Identification and Analysis of Candidate Kinases for a Function in Gravitropic Signalling Using a Novel Phenotyping Pipeline

Lukas Karl Brenton Müller, BSc

Submitted to the

University of Nottingham

for the degree of Doctor of Philosophy

July 2018

Abstract

Roots use gravity to guide their growth (gravitropism). Gravitropism involves the establishment of an auxin gradient across the root, which is driven by polar auxin transport (PAT). Auxin acts as a growth inhibitor, triggering a differential growth response that results in root bending towards gravity.

Using a three-step process, my aim was to identify and investigate *Arabidopsis thaliana* kinases likely to function in gravitropic signalling (candidate kinases). First, I utilised my Kinase Subsequence SELECTion (KSselect) algorithm to identify candidate kinases. Second, transcriptional and translational fusions are used to characterise candidate kinases. Third, I used our Quantitative Gravitropic Analysis Pipeline(qGAP) to phenotype candidate kinase mutants. My research focused on the D6PKL3 kinase and the redundant CIPK4 and CIPK7 (CIPK4/7) kinases.

D6PKL3 is identified as auxin-inducible and closely related to D6PK, which regulates PAT by targeting PIN proteins. Although KSselect predicts D6PKL3 and D6PK functional divergence, further analysis suggests that both target PINs. D6PKL3 is expressed in columella cells (continuous) and the basal meristem (auxin-induced). In the basal meristem, D6PKL3 does not co-localise with PINs, allowing for the possibility that D6PKL3 targets the ABCB1 transporter. The qGAP results exclude a non-redundant function of D6PKL3 in gravitropism.

CIPK4/7 are identified by KSselect as putative orthologs of ZmCIPK15, which is associated with the regulation of crown root angle in *Zea mays*.

Transcriptional fusions locate CIPK4/7 in the maturation zone and the basal meristem in *Arabidopsis thaliana*. Although the auxin inducible CIPK7 is expressed predominantly over CIPK4 in the basal meristem, the qGAP results conclude that neither function in gravitropic signalling. I then extended my CIPK4/7 research to the cold stress response and found that the CIPK4/7 mutants did not show a cold-conditional phenotype.

My research addresses the challenges faced by gravitropic research, and offers novel tools including KSselect and qGAP. These tools are timely and can be adapted by researchers in a variety of fields.

Dedication

For my daughter Harriet.

My little birdie.

Acknowledgement

This thesis deviates from a traditional Plant Science thesis. It uses tools (e.g. Jupyter notebooks) that have been well established in other fields, but haven't found their way into Plant Science. These new tools are combined with traditional Plant Science approaches. The tools used in this thesis were chosen as the most suitable to answer the questions set at the outset of my research.

Firstly, I want to thank my supervisors Malcolm Bennett, Ranjan Swarup and Darren Wells for guiding me, challenging me and allowing me to explore new ways of addressing my research.

Secondly, I want to thank colleagues and staff for their relentless help, including Nicholas Redman for kindly sharing his image data for the *pgm1* analysis.

Thirdly, I want to thank my A-level teachers Annemarie and Michael Stephan for helping me ignite an interest in the sciences.

Fourthly, I want to thank my parents Wolfgang and Christa Müller for their love and support.

Finally, I want to thank my partner Carrie Peploe for helping me through the dark hours.

List Of Abbreviations

AL : Activation Loop (of kinase domain)

ARBC : Auxin Responsive Backup Circuit

ARF : Auxin Response Factor

AS : Activation Segment (of kinase domain)

AVV : Angle Versus Vertical

BLUP : Best Linear Unbiased Predictions (part of LMM)

DoS : Domains of Specificity (of kinase domain)

ER : Endoplasmic Reticulum

GR : Growth Rate

GSA : Gravitropic Set-point Angle

GWAS : Genome Wide Association Study

HM : Hydrophobic Motif (of kinase)

LMM : Linear Mixed Model

MC-group : Minimal Charge-group (of PIN phospho-sites)

MDD : Minimal Detectable Difference

MSA : Multiple Sequence Alignments

NaN : Not a Number

PA : Phosphatide Acid

PH : Pleckstrin Homology

PM : Plasma Membrane

PRBP : Phospho-site Residue "Binding Pocket" (of kinase domain)

PSSM : Position-Specific Scoring Matrix (of kinase domain)

R-spine : Regulatory Spine (of kinase domain)

SC1-group : Selective Charge 1-group (of PIN phospho-sites)

SC2-group : Selective Charge 2-group (of PIN phospho-sites)

SD : Standard Deviation

SE : Standard Error

Contents

1 Introduction **1**

1.1 Gravitropism 1

1.1.1 Gravitropism in Roots 1

1.1.2 Graviperception in Roots 3

1.1.3 Re-configuration of the Auxin Gradient in the Elongation Zone and the Gravitropic Response 10

1.1.4 Methods in Gravitropic Research 16

1.2 Kinases in Gravitropism 19

1.2.1 Kinase Function and Regulation 19

1.2.2 AGC-Kinases in Gravitropism 22

1.2.3 CIPK-Kinases in Gravitropism 31

1.3 Aims and Objectives 36

2 Methods **38**

2.1 Molecular Methods 38

2.1.1 DNA extraction 38

2.1.2 Polymerase Chain Reaction 39

2.1.3 Gel electrophoresis 41

2.1.4 Gel extraction 41

2.1.5 Plasmid Methods 41

2.1.6 Restriction 42

2.1.7 Ligation 43

2.1.8 Transformation and Selection of Bacteria 44
2.2 Plant Methods 45
2.2.1 Growing *Arabidopsis thaliana* 45
2.2.2 *Arabidopsis thaliana* Transformation 46
2.2.3 *Arabidopsis thaliana* Crossing 47
2.2.4 *Arabidopsis thaliana* Seed Harvesting 47
2.2.5 Histochemical GUS Assays of Seedlings 47
2.3 Gravitropic Assays 48
2.3.1 Gravitropic Assay 48
2.3.2 Cold-Conditional Gravitropic Assay 48
2.4 Bioinformatic Methods 48
2.4.1 Analysis and Programming Software 48
2.4.2 KSselect Module and Notebooks 49
2.4.3 qGAP Module and Notebooks 50
2.4.4 PRBPviz Notebooks 53
2.4.5 ETE3 Data Visualisation - KSviz, Auxin Induction and Abiotic Stress 54

3 Phylogenetic Analysis to Infer Kinase Function - A Novel Screening Approach 55
3.1 Introduction 55
3.2 Auxin-Inducible AGC-Kinases 59
3.3 Phylogeny of AGC-Kinases 63
3.4 Phylogeny of Auxin-Inducible CIPK-Kinases 67
3.5 Conclusion 70

4 Characterisation and Phenotyping of D6PKL3 74
4.1 Introduction 74
4.2 Prediction of D6PK Phospho-Site Preferences 78
4.3 D6PKL3 Expression and Targeting 88

4.4 Gravitropic Response of D6PK Mutants 94
4.5 Conclusion 100

5 Characterisation and Phenotyping of CIPK4 and CIPK7 103
5.1 Introduction 103
5.2 CIPK4 and CIPK7 Expression, Targeting and Gravitropic Phenotyping 105
5.3 CIPK Stress Expression Screen and Conditional Gravitropic Phenotyping 112
5.4 Conclusion 118

6 The Quantitative Gravitropic Analysis Pipeline (qGAP) - A Novel and Adaptable Phenotyping Tool 120
6.1 Introduction 120
6.2 Reading the RootTrace Output 123
6.3 Statistical Analysis of RootTrace Output 126
6.4 Removal of Outliers 127
6.5 Conclusion 133

7 Conclusion 135

References 142

Published Papers 162

A Supplemental Material 163
S1 Supplemental Figures 163
S1.1 Supplemental Figures for Chapter 2: Methods 163
S1.2 Supplemental Figures for Chapter 3: Phylogenetic Analysis to Infer Kinase Function - A Novel Screening Approach 169

S1.3 Supplemental Figures for Chapter 4: Characterisation and Phenotyping of D6PKL3 187
S1.4 Supplemental Figures for Chapter 5: Characterisation and Phenotyping of CIPK4 and CIPK7 199
S1.5 Supplemental Figures for Chapter 6: The Quantitative Gravitropic Analysis Pipeline (qGAP) - A Novel and Adaptable Phenotyping Tool 224
S2 Supplemental Code 228
S2.1 Jupyter Notebooks 228
S2.2 Modules 288

List of Figures

1.1 Schematic representation of a cross-section of an *Arabidopsis thaliana* root apex. Cell types are colour coded as indicated in the legend. Different tiers of columella cells are indicated by S1, story 1; S2, story 2; S3, story 3; and PC, peripheral cells. Adapted from Ubeda-Tomas *et al.*, 2012. 4

1.2 Schematic representation of the subcellular organisation of an *Arabidopsis thaliana* columella cell and an indication of its position in the root tip. Subcellular structures are indicated by Ac, actin microfilaments; Am, amyloplasts; ER, endoplasmic reticulum; N, nucleus; and V, vacuoles. Adapted from Ubeda-Tomas *et al.* (2012). 6

1.3 Time course of pH-changes in S2 columella cells and an indication of their position within the root tip. False colour images based on the monitoring of BCECF-Dextran microinjection signal. Time steps given in minutes after gravistimulation. Left side shows signal from vertical cells. Scale given in pH units. Adapted from Ubeda-Tomas *et al.* (2012) and Fasano *et al.* (2001). 7

1.4 Schematic representation of the reverse fountain model during gravistimulation. Grey arrows indicate direction and extent of auxin flux. Cell types are colour coded. Grey, S1 and S2 columella; yellow, lateral root cap; blue, epidermis. Arrow on right hand side indicates gravity vector (g). Adapted from Swarup *et al.* (2005). 9

1.5 PIN3 transcytosis in S2 columella cells upon gravistimulation. (A) PIN3-GFP signal after cyclohexamide (CHX) treatment, before gravistimulation. (B) Gravistimulation and subsequent photobleaching (white boxes) of CHX-treated root. Confocal images show PIN3-GFP signal at 0 and 30 minutes after gravistimulation. GFP signal in bottom photobleached area after 30 minutes shows PIN3 transcytosis. Adapted from Ubeda-Tomas *et al.* (2012) and Kleine Vehn *et al.* (2010). . 10

1.6 PIN transporter function during gravitropism as suggested by expression domains. (A) Schematic of the gravi-stimulated root apex. PIN proteins with a proposed function in polar transport are indicated. (B) Expression of PIN proteins in the root apex, as shown by GFP signal of translational fusions. Adapted from Swarup *et al.* (2005) and Yan *et al.* (2013). 11

1.7 Schematic of the genomic auxin response based on ARF and AUX/IAA transcription factors. ARF transcriptional activators (type I) and ARF transcriptional repressors (type II and III) bind auxin response elements to regulate gene expression. Both type I and II ARFs form homodimers (dashed circles), while only type I ARFs form heterodimers with AUX/IAA transcriptional repressors (dashed line). Binding of type I ARFs by AUX/IAAs counteract their transcriptional activation. Repression of type I ARFs by AUX/IAAs is relieved under elevated auxin conditions via auxin-dependent degradation of AUX/IAAs, which is facilitated by TIR1/ABFs. . . 16

1.8 Venn diagram of the research methodology used in 51 papers from 2000 until 2018. Statistical tests are given in braces, if provided. Both scatter plots and bar charts usually give standard error error-bars. Star (*) indicates standard deviation error-bars. Publications in overlapping regions use multiple methods. See References for full references. 18

1.9 (A) Akt/PKB kinase domain 3D-model in complex with a substrate peptide (GSK3, orange) (PDB ID: 1O6K; Yang *et al.*, 2002). Domains of Specificity (DoS) that scored higher than 0.9 in the KINspect algorithm are highlighted (cyan). Arrows (red) outline three canals formed by these DoS. See (B-F) for longitudinal and (H-K) for transversal slices of (A). From Creixell *et al.* (2015). 21

1.10 (A) Conserved 3D structure of AGC-VIII kinase domain. Yellow indicates N-lobe and blue indicates C-lobe. Position of αC-helix is highlighted. (B) Phosphorylated activation loop binding to Arg in the VIb subdomain of the C-lobe, which allows for formation of the R-spine (grey shell). Position of αC-helix indicated (C) Conserved hydrophobic pockets (1-4) surrounding the αC-helix. Adapted from Rademacher and Offringa (2012) and Taylor and Kornev (2011). 23

1.11 (A) Schematic of the AGC C-tail highlighting regions that interact with the kinase domain. Yellow circles represent conserved PxxP and hydrophobic motifs (F). (B) Activation of AGC kinases by docking to PDK1 via the hydrophobic motif (yellow circles) located on the C-tail (red line). PDK1 phosphorylates (red dot) the activation loop (blue line) of the interacting AGC kinase. Following phosphorylation by PDK1, the C-tail of the AGC kinase binds its own kinase domain. Adapted from Kannan *et al.* (2007). 24

1.12 Phylogenetic tree of AGC-VIII kinases. Ticks indicate conservation in green algae, mosses, ferns and seeded plants. Right-hand column states whether the hydrophobic motif is absent (none) or whether it is the consensus sequence (FxxF). PHOT2 and D6PKL3 are the only AGC-VIII kinases to retain a diverged hydrophobic motif (IxxF). Adapted from Galvan-Ampudia and Offringa (2007) and Zegzouti *et al.* (2006a). 26

1.13 (A) Subcellular localisation of full-length PHOT2 protein (P2G) and various truncated PHOT2 proteins (P2NG, P2CG and P2C(D720/N)G) five minutes after blue light irradiation. Full-length PHOT2 (P2G) accumulates at PM and Golgi. The N-tail domain (P2NG) is observed throughout the cytoplasm. The C-tail and kinase domain (P2CG) accumulates at Golgi and faintly at the PM. The kinase-dead/unphosphorylated kinase domain plus C-tail (P2C(D720/N)G) shows a more pronounced accumulation at the PM than P2CG. (B) Schematic of the conformational change in the activation segment after phosphorylation. Adapted from Kong *et al.* (2006) and Taylor *et al.* (2012). 28

1.14 Evidence that PID-mediated phosphorylation of PINs targets them to the shoot-ward PM. (A) PIN2 immunolocalisation in WT (top), *pid9* (bottom left), and 35S::PID line (bottom right). The *pid9* mutant shows accumulation of PIN2 in endomembrane structures of root cells. The over-expression line shows root- to shootward localisation shift of PIN2 in root cortex. (B) PIN1 translational fusions in WT (top) and 35S::PID background (bottom). The over-expression line shows shoot- to rootward localisation shift. (C) Phospho-mimic (top) and phospho-mutant (bottom) for PID phosphosites of PIN1 translational fusions. Phospho-mimic accumulates at shoot-ward PM of shoot apical meristem cells, while phospho-mutant accumulates at rootward PM of shoot apical meristem cells. Adapted from Friml *et al.* (2004), Huang *et al.* (2010), and Sukumar *et al.* (2009). 30

1.15 (A) Conserved 2D structure of CIPK kinases. NAF functions as auto inhibitory sequence. PPI interacts with protein phosphatase.(B) Conserved 3D structure of CIPK kinase domain illustrating that the hydrophobic pockets are occupied by the NAF domain, which inhibits the kinase. C and N-terminals are indicated. (C) Conservation of the hydrophobic motif in 13 *Arabidopsis thaliana* CIPK junction domains. Adapted from Chaves-Sanjuan *et al.* (2014). 33

1.16 (A) Schematic of CIPK kinase structure. (B) Schematic of CBL proteins interacting with the NAF domain of CIPK proteins, which relieves the CIPK kinase domain (C- and N-lobe) from inhibition by the NAF domain. (C) Schematic of CBL protein structure showing the four EF-hands and indicating the variable N-tail (orange). (D) Sequence alignments of the N-tail of all *Arabidopsis thaliana* CBL proteins. Sequences cluster into three clades: long N-tail (CBL6,3,2,10), myrosylated/palmitoylated short N-tail (CBL4,1,9,5) and non-myrosylated/palmitoylated short N-tail(CBL8,7). The three CBL protein clades show conserved subcellular localisation to the tonoplast (long N-tail), PM (myrosylated/palmitoylated short N-tail), and the cytosol and nucleus (non- myrosylated/palmitoylated short N-tail); as demonstrated by GFP/OFP fusions in protoplasts. Adapted from Chaves-Sanjuan *et al.* (2014), Chaves-Sanjuan *et al.* (2013), and Batistic *et al.* (2010). 34

2.1 The formula used to calculate the volume of the reactants used in the ligation reaction. 43

3.1 Schematic of the KSselect algorithm. KSselect uses a multiple sequence alignment (or a Phytozome (csv) output) of full protein sequences as input. From this it generates multiple sequence alignments of the kinase domain and the domains of specificity. The phylogenetic trees generated from these alignments can be used for an in-depth analysis of the functional and evolutionary relationship of the kinases of interest. 57

3.2 Unrooted phylogentic tree (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their ARF7/19-dependent auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3. Scale bar: Mean number of substitutions per site. (see Section 2.4.5) 61

3.3 Unrooted *Arabidopsis thaliana* AGC-VIII kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site. 64

3.4 Unrooted *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site. 65

3.5 Unrooted DoS-dendrogram of *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinases (same as Figure 3.4c). Sequence detail shows a subset of domains of specificity (DoS). Red arrow indicates a conserved change (S/T to P) in one DoS of D6PKL3 and D6PKL3-related kinases. Generated using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site. 66

3.6 Unrooted *S. moellendorffii, Z. mays, A. lyrata, A. halleri, A. thaliana, B. distachyon, C. rubella, G. max, V. vinifera, O. sativa, P. trichocarpa, P. patens* and *S. lycopersicum* CIPK4/7 kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site. 68

3.7 Unrooted phylogenetic tree (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their ARF7/19-dependent auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3 (see Section 2.4.5). 71

4.1 Promoter-GUS expression analyses of the D6PK-family. (A,D,G,J) GUS staining of 7-day-old seedlings. (B,E,H,K) GUS staining of root tips. (C,F,I,L) GUS staining of lateral root initiation sites. (K) Only D6PKL3 accumulates in columella cells of the root tip . (E) D6PKL1 is expressed ubiquitously throughout the root tip, including the columella cells. (B, H) D6PK and D6PKL2 show no expression in columella cells. From Zourelidou *et al.* (2009) 76

4.2 Schematic of PID and D6PK group functions based on published results, following the linear model of kinase signal transduction. Graph (bottom) indicates the suggested all-or-nothing PIN-activation by these kinases. 77

4.3 [Part 1] Schematic representation of the 3D structure of the PKA kinase (grey) and its substrate (orange). Subfigures (a-c) show the ”binding-pockets” / PRBPs (purple) for each phosphorylation site residue (P-3, P-2 and P-1)(green). Generated using Chimera. 79

4.4 [Part 2] Schematic representation of the 3D structure of the PKA kinase (grey) and its substrate (orange). Subfigures (a-d) show the ”binding-pockets” / PRBPs (purple) for each phosphorylation site residue (P-0, P+1, P+2 and P+3) (green). Generated using Chimera. 80

4.5 Balloon plots of Phospho-site Residue Binding Pockets (PRBPs) for PIDs and D6PKs. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 82

4.6 Balloon plots of phosphorylation sites on PINs (S1, S2,...). Colours indicate charge and circle diameter indicates molecular weight of residue. Generated using PRBPviz (see Section 2.4.4) 84

4.7 Schematic of D6PK and PID group function based on PRBPviz results. PRBPviz results suggest multiple target phospho-site types (MC-group, SC1-group and SC2-group) for one group of kinase and cross talk between the kinases (shared target phospho-site types). Dashed lines indicate potential Auxin Responsive Backup Circuit formed by D6PKL3 and other kinases. Graph (bottom) indicates unknown gradual PIN-activation by D6PK kinases. Dashed curve indicates PIN-activation expected, if ARBC is formed. 87

4.8 D6PKL3 expression after 4 hours of auxin treatment (bottom) and under control conditions (top). Under control conditions, D6PKL3 is mainly expressed in the columella cells. After 4 hours of auxin treatment, D6PKL3 expression is shifted to the peripheral columella cells and is additionally expressed in the basal meristem, as shown by promoter GUS fusion. [bar = 75μm] 90

4.9 Translational fusions of (a) D6PKL3 and (b) ABCB1 show overlapping expression in the root tip. D6PKL3 and ABCB1 appear to co-localise at the rootward plasma membrane in the basal meristem. In columella cells, D6PKL3 mainly localises to intracellular structures. ABCB1-YFP image from Henrichs *et al.* (2012). [bar = 50μm] 91

4.10 Balloon plot of phosphorylation sites on the ABCB1 linker region (S616, S621,...). Colour indicates charge and circle diameter indicates molecular weight of residue. Generated using PRBPviz (see Section 2.4.4) 93

4.11 Schematic of D6PKL3 function in columella cells (left) and basal root meristem (right). Expression, targeting and PRBPviz results for D6PKL3 suggest that D6PKL3 has two targets: PIN in the columella cells and ABCB1 in the basal mersitem. Dashed lines indicate potential Auxin Responsive Backup Circuit formed by D6PKL3 and D6PKL1 in the basal meristem. Graphs (bottom) indicate unknown gradual PIN-activation and ABCB1-inhibition by D6PK kinases. Dashed curve indicates ABCB1-inhibition expected, if ARBC is formed. 95

4.12 Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* triple mutant and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. See Table 4.1 for corresponding summary of linear mixed model. Time is given in minutes and angle in degree. 98

5.1 CIPK4::GUS expression analysis. (a) CIPK4 shows root-specific expression. (b) CIPK4 expression found in the basal meristem of root tip. (c) CIPK4 expression found in the maturation zone, but not in young lateral roots (see Section 2.2.5 for GUS-protocol and Supplemental Fig. **??** for construct details). 108

5.2 CIPK7::GUS expression analysis. (a) CIPK7 shows root-specific expression. (b) CIPK7 expression found in the basal meristem of root tip. (c) CIPK7 expression found in the maturation zone and in young lateral roots (see Section 2.2.5 for GUS-protocol and Supplemental Fig. **??** for construct details). 109

5.3 CIPK7 appears to localise to (a) intracellular structures (potentially the tonoplast) in the basal meristem (root tip) and (b) to the PM in the maturation zone. CIPK7::CIPK7-GFP viewed under the confocal microscope (see Supplemental Fig. **??** for construct details). [bar = 75μm] 110

5.4 Gravitropic kinetics of Col-0 and *cipk4-1*, *cipk4-2*, *cipk7-1* and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. See Table 5.1 for corresponding summary of linear mixed model. Time is given in minutes and angle in degree. 112

5.5 Unrooted phylogenetic tree (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their cold-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3 (see Section 2.4.5). Scale bar: Mean number of substitutions per site. 114

5.6 Root growth rates of *cipk4-1, cipk7-1, cipk4-1 cipk7-1* and isozygous wild-type control following control- (top) and cold-treatment (bottom) analysed using qGAP. Dashed lines show 95% confidence interval based on linear mixed model. Isozygous wild-type control root growth under control conditions is superimposed on every panel. Time is given in minutes and growth in pixel. 117

6.1 Schematic of the qGAP workflow. Middle (blue) indicates existing tools (Imaging Robot, RootTrace) plus the novel anaylsis pipeline (qGAP module and qGAP Jupyter notebooks). Left (black and green) indicates output files and directory structure. Right (red) indicates analysis output. IR-robot produces time-course images of different plates organised into folders (e.g. Plate1). We use RootTrace to analyse these images a quarter of a plate at a time, and save it to a new subfolder (e.g. Q1). The qGAP read module generates one data-frame for angle versus vertical (AVV) measurements and one for growth rate (GR) measurements. These data-frames are analysed using the qGAP Jupyter notebooks (see text for detail). 122

6.2 Root growth rate and gravitropic kinetic plots suggest that both change linearly with time for the observed time-frame. 128

6.3 Histograms of differences in slope from grand mean as estimated by BLUP (qGAP example dataset). Lines indicate 3*standard deviation (red) and 1.5*standard deviation (purple) thresholds, which are given as parameters in qGAP_filt.ipynb 130

6.4 Power curves of the mixed models based on (a, b) the full dataset, (c, d) the 3*standard deviation filtered dataset and (e, f) the 1.5*standard deviation filtered dataset. Power was estimated using simulations, with the Col-0 model as the baseline. The Effect.size at 80% Power is often defined as the Minimal Detectable Difference (MDD). Power analysis simulations were ran using simr. 132

S1 Schematic of Green Gate cloning strategy. See Lampropoulos *et al.* (2013) for details of Green Gate cloning protocol. Kozak sequences (red) were introduced using a modified B dummy, a modified GUS C-element, and via the primers used to amplify coding sequences (CDSs). This cloning strategy was used for CIPK4:GUS, CIPK7:GUS and CIPK7:CIPK7-GFP. 164

S2 Linear Mixed Model (LMM) Best Linear Unbiased Predictions (BLUPs) superimposed on scatter plots for the *pgm1* mutant and control (Col-0) showing root angle versus time. . 165

S3 Linear Mixed Model (LMM) Best Linear Unbiased Predictions (BLUPs) superimposed on scatter plots for the *pgm1* mutant and control (Col-0) showing root growth versus time. 166

S4 Gravitropic kinetics of *pgm1* mutant and control (Col-0) lines analysed using qGAP. Dashed lines give 95% confidence interval based on a linear mixed model. 167

S5 Power curve for the mixed model based on the *pgm1* dataset. Power was estimated using simulations, with the Col-0 model as the baseline. The Effect.size at 80% Power is often defined as the Minimal Detectable Difference (MDD). Power analysis simulations were run using simr. 168

S6 Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their ARF7/19-dependent auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 170

S7 Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 172

S8 Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 174

S9 (a) DoS-dendrogram and (b) full sequence phylogeny of *Arabidopsis thaliana* AGC-VIII kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5) 176

S10 (a) DoS-dendrogram and (b) full sequence phylogeny of *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5) 177

S11 *Arabidopsis thaliana* CIPK kinase phylogenies based on different sub-sequences of the full protein sequence (a). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5) 178

S12 (a) DoS-dendrogram and (b) full sequence phylogeny of *Arabidopsis thaliana* CIPK kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5) 179

S13 (a) DoS-dendrogram and (b) full sequence phylogeny of *S. moellendorffii, Z. mays, A. lyrata, A. halleri, A. thaliana, B. distachyon, C. rubella, G. max, V. vinifera, O. sativa, P. trichocarpa, P. patens* and *S. lycopersicum* CIPK4/7 kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5) 180

S14 Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their ARF7/19-dependent auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 181

S15 Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 183

S16 Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 185

S17 Balloon plot of the P+3 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 188

S18 Balloon plot of the P+2 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 189

S19 Balloon plot of the P+1 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 190

S20 Balloon plot of the P-0 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 191

S21 Balloon plot of the P-1 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 192

S22 Balloon plot of the P-2 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 193

S23 Balloon plot of the P-3 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4) 194

S24 Auxin-induction of D6PKL3 in the root as determined by qPCR. TUB and UBA are used as technical controls. AUF1 is used as a positive control for auxin-induction. 195

S25 Additional images of GUS-stained roots after 4 hours of auxin treatment (bottom) and under control conditions (top). Under control conditions, D6PKL3 is mainly expressed in the columella cells. After 4 hours of auxin treatment, D6PKL3 expression is shifted to the peripheral columella cells and is additionally expressed in the basal meristem, as shown by promoter GUS fusion. 195

S26 Root growth rate of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and growth in pixel. 196

S27 Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The full "D6PKL3init"-dataset was used. Time is given in minutes and angle in degree. 197

S28 Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 1.5*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and angle in degree. 198

S29 Yeast two hybrid interaction assays for CIPKs and CBL proteins. Adapted from Albret *et al.* (2001) and Kolukisagolu *et al.* (2004). 199

S30 Isolation of *cipk4* and *cipk7* knock-out mutants. (A) Location of the two T-DNA inserts selected for each gene. (B) Quantitative reverse transcription PCR results for root samples of the selected T-DNA insert lines and two Col-0 lines. TUB-5 sequence was used as standard. TUB-3 sequence included as control for RNA quality (undigested). UBA sequence served as negative control. The *cipk4-1* and *cipk7-1* mutants were identified as knock-out mutants. 200

S31 Growth rates of Col-0, *cipk4-1*, *cipk4-2*, *cipk7-1*, and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. Time is given in minutes and growth in pixel. 201

S32 Histograms of differences in slope from grand mean as estimated by BLUP (CIPK dataset). Lines indicate 3*standard deviation (red) and 1.5*standard deviation (purple) thresholds, which are given as parameters in qGAP_filt.ipynb . . . 201

S33 Gravitropic kinetics of Col-0, *cipk4-1*, *cipk4-2*, *cipk7-1*, and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and growth in pixel. 202

S34 Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (CIPK full dataset). 203

S35 Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (CIPK 3*standard deviation filtered dataset) 204

S36 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their drought-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 205

S37 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their genotoxicity-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 207

S38 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their heat-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 209

S39 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their osmotic-stress-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 211

S40 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their oxidative-stress-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 213

S41 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their salt-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) 215

S42 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their UVB-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 217

S43 Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their wounding-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5) . . 219

S44 (A) Gravitropic kinetics and (B) root tip elongation of Col-0 root after 0, 4, 6, 8, 12, and 24 hours of cold treatment. (C) Auxin response after 12 hours of cold treatment compared to untreated roots as shown by IAA2-GUS auxin marker line. (D) Accumulation of PIN2-GFP in BFA-compartments is absent in roots subjected to cold treatment for 12 hours. (E) Rate of shootward auxin transport is inhibited after 12 hours of cold treatment as measured by transport of radioactively labelled IAA. Adapted from Shibasaki *et al.* (2009) 220

S45 Gravitropic kinetics for *cipk4-1, cipk7-1, cipk4-1 cipk7-1* mutants and isozygous wild-type control after control (top) and cold treatment (bottom) analysed using qGAP. Dashed lines show 95% confidence interval based on linear mixed model. Time is given in minutes and angle in degree times 100. . . . 221

S46 Summary of gravitropic kinetic linear mixed model of CIPK mutants under cold and control treatment. Time is given in minutes and angle in degree times 100. 222

S47 Summary of growth rate linear mixed model of CIPK mutants under cold and control treatment. Time is given in minutes and growth in pixel. 223

S48 Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (full "D6PKinit"/qGAP example dataset). 225

S49 Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (3*standard deviation filtered "D6PKinit"/qGAP example dataset). 226

S50 Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (1.5*standard deviation filtered "D6PKinit"/qGAP example dataset). 227

List of Tables

2.1 Buffers 38

2.2 Standard PCR 39

2.3 Genomic DNA PCR 40

2.4 Restriction Digest 42

2.5 GUS Solutions 47

4.1 Summary of Linear Mixed Model for 3*standard deviation filtered "D6PKL3init"-dataset. See Figure 4.12 for corresponding plot 99

5.1 Summary of Linear Mixed Model for CIPK gravitropic kinetics. See Figure 5.4 for corresponding plot 111

6.1 Example of unfiltered RootTrace measurements (columns 2 and 3), dataset after automatic filters applied by qgap_read (columns 4 and 5) and corresponding RootTrace trace images (column 6). 124

6.2 Comparison of statistical test suitable for analysis of gravitropic kinetic data 127

S1 Details of CIPK mutants. 163

S2 GreenGate primers. 163

S3 qPCR primers. 163

S4 Summary of Linear Mixed Model for *pgm1* dataset. 164

S5 Summary of linear mixed model of root growth data for 3*standard deviation filtered ”D6PKL3init”-dataset 187

S6 Summary of linear mixed model of gravitropic kinetics for full ”D6PKL3init”-dataset 187

S7 Summary of linear mixed model of gravitropic kinetics for 1.5*standard deviation filtered ”D6PKL3init”-dataset 187

S8 Summary of growth rate linear mixed model of full CIPK-dataset. 199

S9 Summary of linear mixed model of gravitropic kinetic data for 3*standard deviation filtered CIPK-dataset. 200

Chapter 1

Introduction

1.1 Gravitropism

1.1.1 Gravitropism in Roots

The primary functions of plant roots are anchorage and resource capture, which must be fulfilled in a spatially heterogeneous environment, i.e. the soil. Plant survival is therefore dependent upon the ability of roots to adapt their growth in response to a range of abiotic stimuli. Tropic movements contribute to this plasticity of root growth. In contrast to nastic movements, tropic growth responses are dependent on the direction of the stimulus. Plants exhibit tropic movements in response to gravity, light, touch and water. Gravitropism, the tropic response to gravity, is considered the most fundamental tropic movement of roots. Primary roots grow downward following the gravity vector, whereas lateral roots grow at an angle to the gravity vector, called the gravitropic set-point angle (GSA). The GSA is influenced by genotype, developmental stage, and the environment (Digby and Fern, 2002; Bai *et al.*, 2013).

It has been suggested that the tropic responses to light, touch, and water (photo-, thigmo- and hydrotropism) need to override the gravitropic response in order to instruct a growth direction (reviewed in Wyatt and Kiss,

2013). Tropic responses can be broken down into three sequential steps: perception, signal transduction, and growth response. Signal transduction of gravitropism can be further split into early signal transduction and signal transduction downstream of an auxin gradient (reviewed in Baldwin *et al.*, 2013).

The molecular basis of gravitropism is thought to be largely conserved across the plant kingdom. However, the root systems of different species vary extensively. For example, the root system of a mature *Arabidopsis thaliana* plant solely consists of a primary root and lateral roots, whereas a mature *Zea mays* root system is dominated by shoot-born roots. The following paragraph describes the molecular basis of the gravitropic response in *Arabidopsis thaliana* primary roots. However, in other species and other root systems this is expected to differ.

Gravity is perceived at the root tip by the columella cells. Columella cells characteristically contain starch filled plastids, termed amyloplasts (Swarup *et al.*, 2013). Amyloplasts sediment upon root reorientation and are thought to trigger changes in cytosolic pH and calcium. Subsequent transcytosis of auxin transporters (PINs) occurs, which results in the accumulation of PINs on the bottom plasma membrane (PM)(Kleine-Vehn *et al.*, 2010). Additionally, direct activation of PINs on the bottom PM is thought to precede transcytosis. Both transcytosis and direct activation of PINs are thought to drive the establishment of a lateral auxin gradient. An auxin gradient is established in the root tip and is then re-configured shootward in the elongation zone, where high levels of auxin inhibit growth. This re-configuration also involves auxin transporters and changes in cytosolic pH and calcium. Once the lateral auxin gradient is re-configured in the elongation zone it is thought to trigger a differential cell growth response, resulting in root bending downward following the gravity vector.

1.1.2 Graviperception in Roots

1.1.2.1 Gravi-sensing in Roots

As early as 1966, the site of gravity perception was mapped to the root tip, with crude removal of the root cap abolishing the gravitropic response (Juniper *et al.*, 1966). A more sophisticated laser ablation approach was able to fine-map gravity perception to the columella cell files (Blancaflor *et al.*, 1998).

Columella cells are a block of cells in the centre of the root cap, which are flanked by the lateral root cap (see Figure 1.1). They originate from the columella initials, which are adjacent to the quiescent centre (Arnaud *et al.*, 2010). Columella cells lay adjacent to the columella initials and form four vertical tiers: story 1 (S1), S2, S3, and the peripheral cell layer (see Figure 1.1) (Sack and Kiss, 1989). In a midsection of the root tip, four rows of columella cells are apparent. However, the three dimensional shape is complex.

The different cell files of the columella cells contribute differentially to gravity perception (Blancaflor *et al.*, 1998). The S2 tier makes the biggest contribution, while the peripheral cell layer appears to have no contribution to root gravitropism. Furthermore, the central columella cells contribute more to gravity perception than the columella cells closer to the lateral root cap (flank columella).

1.1.2.2 Starch-Statolith Hypothesis

Characteristically, all columella cells contain starch filled plastids, also called amyloplasts or statoliths. Upon reorientation of columella cells, amyloplasts sediment towards the lower PM. The starch-statolith hypothesis predicts that this sedimentation is the primary physical signal of gravitropism.

Three major lines of evidence support this hypothesis. Firstly, the starchless *phosphoglucomutase-1 (pgm-1)* mutant shows a highly attenuated

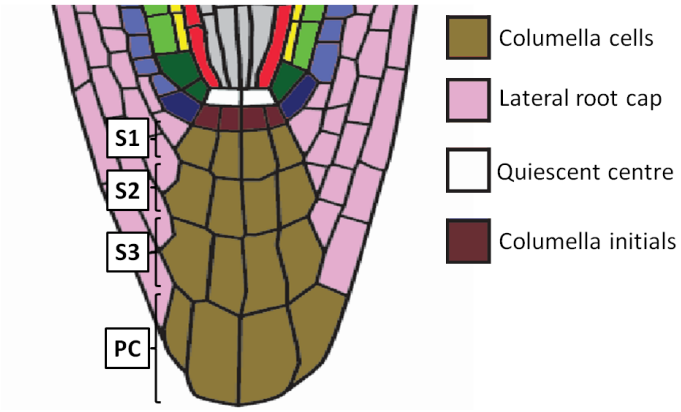


Figure 1.1: Schematic representation of a cross-section of an *Arabidopsis thaliana* root apex. Cell types are colour coded as indicated in the legend. Different tiers of columella cells are indicated by S1, story 1; S2, story 2; S3, story 3; and PC, peripheral cells. Adapted from Ubeda-Tomas *et al.*, 2012.

gravitropic response (Band *et al.*, 2012; Wolverton *et al.*, 2011). A normal gravitropic response can be induced in *pgm-1* by introducing a hypergravity stimulus, further suggesting that starch is not essential for gravity sensing but rather promotes amyloplast sedimentation (Fitzelle and Kiss, 2001). Secondly, repositioning amyloplasts towards the lateral wall of columella cells via the application of a high-gradient magnetic field (HGMF) induces a response that resembles the gravitropic response (Kondrachuk and Hasenstein, 2001). Thirdly, the higher the amyloplast sedimentation velocity in a columella cell, the more it contributes to gravity perception. The S2 tier of columella cells, which play a major role gravitropism, show the highest sedimentation velocity. Flank columella cells show a lower velocity than the central columella cells, which is consistent with their subordinate function in gravitropic signalling (Blancaflor *et al.*, 1998).

Amyloplast-movement in columella cells has been described in relation to the endoplasmic reticulum (ER). The ER of columella cells shows a very distinct cup-shaped morphology, lining the root-ward and lateral PM (Zheng and Steahelin, 2001). Upon reorientation, amyloplasts first move away from the root-ward ER, a movement that is thought to be involved in gravity perception. Following this initial repulsion, amyloplasts sediment

to the new bottom (Leitz *et al.*, 2009).

1.1.2.3 Actin and Gravi-sensing

As the sedimentation of amyloplasts is not severely obstructed by other subcellular structures (Leitz *et al.*, 2009; Wang *et al.*, 2007) (see Figure 1.2), the actin cytoskeleton is thought to facilitate the saltatory movement of amyloplasts (see Section 1.1.2.2). Furthermore, the actin cytoskeleton plays another role in gravitropism, that of transducing the physical strain of amyloplast sedimentation to the PM or the ER membrane.

Two lines of evidence support this function. Firstly, the first observable changes to gravity occur within two minutes (Blancaflor *et al.*, 1998). However, it takes three and a half minutes for the statoliths to reach the lower ER (Leitz *et al.*, 2009). Secondly, repeated short gravistimuli can induce gravitropic bending without amyloplast sedimentation (Heijnowicz *et al.*, 1998).

Assuming transducing amyloplast sedimentation is the only role of the actin cytoskeleton, actin should positively regulate gravitropism. However, treatment with actin depolymerising drugs leads to an enhanced gravitropic response (Hou *et al.*, 2004), indicating a negative regulatory function. It is important to consider that actin depolymerisation also affects other cell biological processes that are important for gravitropism.

1.1.2.4 pH Changes in Early Gravitropic Signalling

The transient alkalisation of the columella cytoplasm is the earliest detectable chemical signal in root gravitropism. Scott and Allen (1999) found that the pH of lower S2 columella cells increased by 0.4 units within 55 seconds following gravistimulation (see Fig. 1.3). This cytoplasmic alkalisation is associated with an apoplastic acidification, suggesting that protons

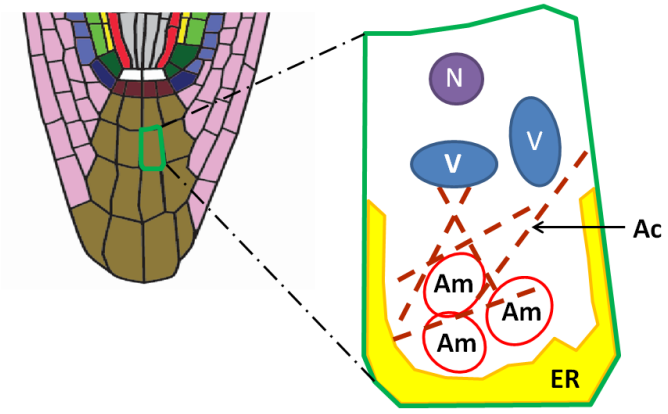


Figure 1.2: Schematic representation of the subcellular organisation of an *Arabidopsis thaliana* columella cell and an indication of its position in the root tip. Subcellular structures are indicated by Ac, actin microfilaments; Am, amyloplasts; ER, endoplasmic reticulum; N, nucleus; and V, vacuoles. Adapted from Ubeda-Tomas *et al.* (2012).

are extruded across the PM (Fasano *et al.*, 2001).

Four lines of evidence suggest a marginal role of cytoplasmic alkalisation in gravitropism. Firstly, the release of caged protons to keep the pH in columella cells constant delays the gravitropic response, but does not abolish it (Fasano *et al.*, 2001). Secondly, cytoplasmic alkalisation is reduced in the starchless *pgm1-1* mutant, which shows a reduced gravitropic response (Fasano *et al.*, 2001). Thirdly, treatment with actin depolymerising drugs, which enhances the gravitropic response, prolongs the cytoplasmic alkalisation (Hou *et al.*, 2004). Fourthly, cytoplasmic alkalisation is absent in the *altered response to gravity1* (*arg1*) mutant, which shows a delayed gravitropic response in roots (Boonsirichai *et al.*, 2003).

1.1.2.5 Calcium Changes in Early Gravitropic Signalling

The transient alkalisation of the columella cytoplasm is thought to be global throughout the cell. Unlike protons, calcium ions generate gradients across single cells, with their diffusion inhibited by calcium-binding proteins (Perbal and Driss-Ecole, 2003).

Two lines of evidence suggest that amyloplast sedimentation induces

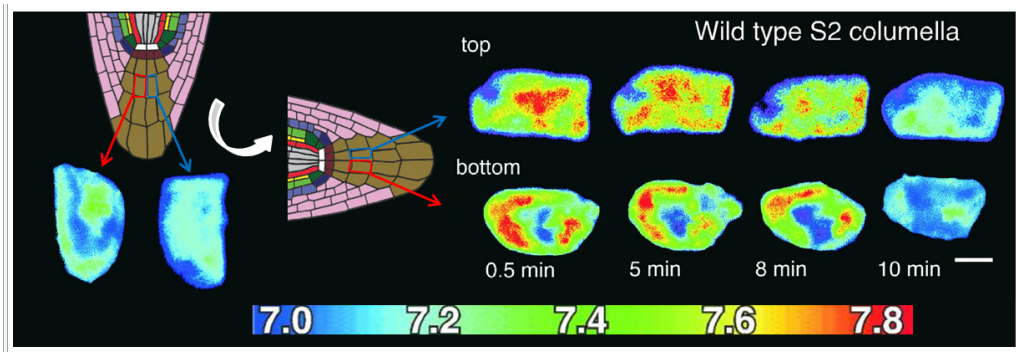


Figure 1.3: Time course of pH-changes in S2 columella cells and an indication of their position within the root tip. False colour images based on the monitoring of BCECF-Dextran microinjection signal. Time steps given in minutes after gravistimulation. Left side shows signal from vertical cells. Scale given in pH units. Adapted from Ubeda-Tomas *et al.* (2012) and Fasano *et al.* (2001).

gravity-specific calcium elevations in columella cells. Firstly, the distinct cup-shaped ER of columella cells is confined to the cell cortex (see Figure 1.2) (Zheng and Steahelin, 2001). This morphology allows directional sensing of amyloplast-sedimentation via ER deformation, suggesting a role in gravity sensing. As the ER is a major calcium store (Stael *et al.*, 2012), it is possible that calcium contributes to early gravitropic signalling. Secondly, the application of calcium chelators and drugs that inhibit mechanosensitive ion channels are shown to inhibit root gravitropism (reviewed in Fasano *et al.*, 2002).

However, two different approaches were unable to detect gravity-specific calcium elevations in columella cells. In the first approach, a luminescence-based sensor aequorin was ubiquitously expressed to measure calcium transients of whole seedlings upon gravistimulation. Gravity-specific calcium transients peaking 40 to 60 seconds after gravistimulation were detected, but the method did not allow for aequorin-reconstitution in the root (Plieth and Trewavas, 2002; Toyota *et al.*, 2008). In the second approach, calcium elevations were imaged using the FRET-based sensor YC3.6. This sensor revealed gravity-induced calcium elevation in the lower lateral root cap, which plateau approximately two minutes after gravistimulation. How-

ever, no calcium elevations were detected in columella cells (S. Gilroy, pers. comm.).

1.1.2.6 Establishment of the Auxin Gradient

1.1.2.6.1 Auxin in Gravitropism The roles of actin, pH, and calcium changes in early gravitropic signalling remain elusive. However, it is known that the final output of graviperception is a lateral auxin gradient. Band *et al.* (2012) show that auxin accumulates on the lower side of the root tip five minutes after gravistimulation.

Two lines of evidence suggest that this auxin gradient is essential for gravitropic signalling. Firstly, treatment with the auxin transport inhibitor naphthylphthalamic acid (NPA) inhibits the gravitropic response of roots upon a 90° reorientation (Rashotte *et al.*, 2000). Secondly, a loss-of-function mutation in the auxin influx carrier AUX1 causes the root to be agravitropic (Marchant *et al.*, 1999).

According to the reverse fountain model (see Figure 1.4), auxin from the shoot arrives at the root tip mainly via mass flow through the phloem. Carriers unload this auxin and an auxin maximum is formed in the columella initials (Sabatini *et al.*, 1999). Columella cells redirect the auxin flux laterally, and auxin is transported shoot-wards in the lateral root cap and the epidermis via polar auxin transport (see Section 1.1.3.1) (Swarup *et al.*, 2005).

In vertical primary root tips, auxin transport on either side of the columella cells occurs at an equal rate. Upon gravistimulation, auxin is preferentially transported out of the columella cells through their bottom PM, thereby establishing an auxin gradient across the root tip (see Figure 1.4).

1.1.2.6.2 The Role of Auxin Carriers in Gradient Formation Auxin transport is increased across the lower columella PM following gravistimulation. This can be explained by an increase in auxin transporter abundance

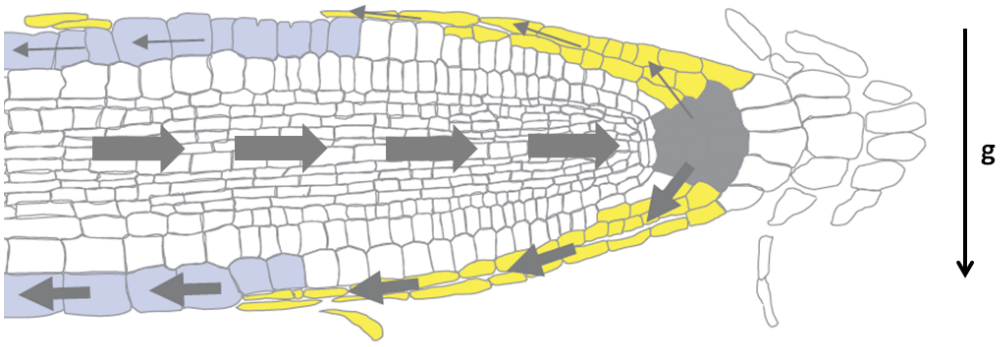


Figure 1.4: Schematic representation of the reverse fountain model during gravistimulation. Grey arrows indicate direction and extent of auxin flux. Cell types are colour coded. Grey, S1 and S2 columella; yellow, lateral root cap; blue, epidermis. Arrow on right hand side indicates gravity vector (g). Adapted from Swarup *et al.* (2005).

and/or activity compared to the upper columella PM. Four auxin transporters are expressed in the columella domain: AUX1, PIN3, PIN4, and PIN7.

AUX1 is expressed in the S2 layer and its localisation to the PM is dynamic, which suggests that it can modulate auxin redistribution. However, no evidence shows that AUX1 is polarly distributed in columella cells (Swarup *et al.*, 2001).

On the contrary, PIN3 and PIN7, which localise symmetrically in vertical columella cells, have been shown to accumulate on the lower PM upon gravistimulation (Friml *et al.*, 2002; Kleine-Vehn *et al.*, 2010). The PIN3 proteins accumulating at the lower PM originate from endomembrane-stores, which are rapidly replenished by gravity-induced PIN3 endocytosis. This process is referred to as PIN3 transcytosis (Kleine-Vehn *et al.*, 2010). PIN3 transcytosis is thought to facilitate the establishment of the lateral auxin gradient. However, there are no reports of PIN3 transcytosis occurring at a rate fast enough to account for the rapidly forming auxin gradient. An auxin gradient is predicted to be established five minutes after gravistimulation (Band *et al.*, 2010). However, Kleine-Vehn *et al.* (2010) show PIN3

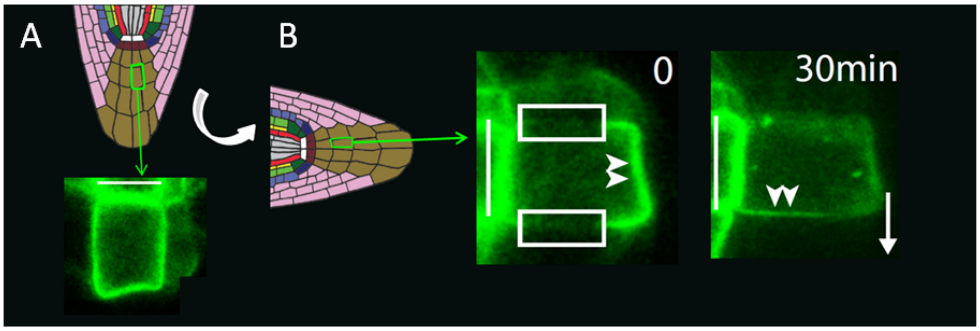


Figure 1.5: PIN3 transcytosis in S2 columella cells upon gravistimulation. (A) PIN3-GFP signal after cyclohexamide (CHX) treatment, before gravistimulation. (B) Gravistimulation and subsequent photobleaching (white boxes) of CHX-treated root. Confocal images show PIN3-GFP signal at 0 and 30 minutes after gravistimulation. GFP signal in bottom photobleached area after 30 minutes shows PIN3 transcytosis. Adapted from Ubeda-Tomas *et al.* (2012) and Kleine Vehn *et al.* (2010).

transcytosis 15 minutes after gravistimulation (see Figure 1.5).

1.1.3 Re-configuration of the Auxin Gradient in the Elongation Zone and the Gravitropic Response

1.1.3.1 Locations of Re-configuration and Response

The gravitropic response takes place in the elongation zone, which is positioned shootward of the root cap. Bending of the elongation zone is caused by the inhibition of cell elongation on the lower side relative to the upper side. Auxin accumulation at the lower side, which inhibits cell elongation, is essential for this differential growth response between upper and lower side. Swarup *et al.* (2005) were able to further fine-map the gravi-responsive tissue. Blocking the auxin response specifically in the epidermis of the elongation zone abolishes the gravitropic response. This suggests that the epidermis is the primary site involved in the differential growth response, and that it regulates the elongation of underlying tissues.

Early gravitropic signalling ends with the establishment of an auxin gradient across the columella cells in the root cap (see Section 1.1.2.1).

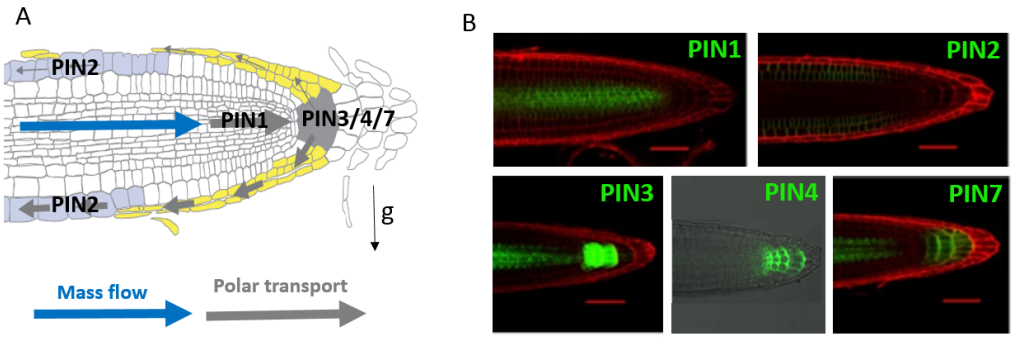


Figure 1.6: PIN transporter function during gravitropism as suggested by expression domains. (A) Schematic of the gravi-stimulated root apex. PIN proteins with a proposed function in polar transport are indicated. (B) Expression of PIN proteins in the root apex, as shown by GFP signal of translational fusions. Adapted from Swarup *et al.* (2005) and Yan *et al.* (2013).

This gradient is subsequently re-configured in the elongation zone before the gravitropic response occurs. One main facilitator of this shootward re-configuration is the auxin influx carrier AUX1. The *aux1* knock-out mutant is agravitropic and can be re-complemented by AUX1 expression in the lateral root cap and root epidermis (Swarup *et al.*, 2005). AUX1 expression in the columella cells is therefore not essential for the gravitropic response, and AUX1 mainly functions in the re-configuration of the auxin gradient (Swarup *et al.*, 2005). It is also evident that auxin transport via the lateral root cap and root epidermis is essential for the re-configuration of the auxin gradient (see Figure 1.6).

1.1.3.2 Re-configuration of the Auxin Gradient

AUX1 is a major auxin transporter involved in the re-configuration of the auxin gradient (see Section 1.1.3.1). However, AUX1 is not polarly localised in the lateral root cap nor the root epidermis, meaning it cannot be responsible for the directionality of auxin transport.

Three other auxin transporters are reportedly expressed in these tissues: PIN2, ABCB1, and ABCB4. Two lines of evidence suggest that the PIN2 auxin efflux carrier functions as the key player in directing shootward re-

configuration of the auxin gradient. Firstly, PIN2 shows a more distinct polar localisation than ABCB1 and ABCB4. It accumulates on the shootward PM in the lateral root cap and the root epidermis, consistent with a function in the shootward re-configuration of the auxin gradient. Secondly, the *pin2* mutant is largely agravitropic. In contrast, the *abcb1* mutant exhibits wild-type root gravitropic kinetics, while the *abcb4* mutant has a 50% reduction in its root bending rate. The residual gravitropic response exhibited by the *pin2* mutant is likely due to the expression of PIN1 in the PIN2 domain (Vieten *et al.*, 2005).

In addition to being the key player in instructing the direction of auxin transport, PIN2 regulates the auxin transport capacity during auxin gradient re-configuration. Gravi-stimulation triggers PIN2 degradation on the upper side of the root, creating a differential auxin transport capacity across the gravistimulated root (Abas *et al.*, 2006). This PIN2-mediated increase in auxin transport on the lower side of a gravistimulated root is essential for gravitropic signalling. A point mutation rendering PIN2 resistant to proteolysis (*wav6-52*) abolishes the degradation of PIN2 on the upper side of a gravistimulated root, causing the root to be agravitropic. Consistent with its resistance to proteolysis, in *wav6-52* the mutated PIN2 protein accumulates in endomembrane structures (Abas *et al.*, 2006). Proteolysis of PIN2, and the resulting differential in auxin transport capacity across the gravistimulated root appear to be vital in gravitropic signalling.

1.1.3.3 Gravitropic Reponse

1.1.3.3.1 Non-Transcriptional Response The gravitropic response takes place in the root elongation zone (see Section 1.1.3.2). The auxin gradient, which is initiated in the columella cells, is re-configured in the elongation zone within six minutes of gravi-stimulation. Root bending downward following the gravity vector can be detected 10 minutes after

gravi-stimulation (Band *et al.*, 2012), which is four minutes after the auxin gradient has reached the elongation zone. This time-frame is too short for auxin to act via transcriptional regulation, as it takes a minimum of 15 minutes for auxin-induced genes to be detectable (translation takes additional time) (Kimbrough *et al.*, 2004). The initial gravitropic response must therefore be controlled by a non-transcriptional auxin response.

Two key steps of the non-transcriptional auxin response have been reported. The earliest detected response to auxin is an increase in cytosolic calcium levels. This increase in cytosolic calcium triggers alkalisation of the apoplast, which in turn is thought to change the enzymatic activity of cell-wall remodelling enzymes (Barbez *et al.*, 2017).

Three lines of evidence identify the increase in cytosolic calcium levels and apoplast alkalisation as steps of the non-transcriptional auxin response. Firstly, the global application of 100nM of indole-3-acetic acid onto the root leads to an increase in cytosolic calcium levels (11 seconds) and alkalisation of the apoplast (15 seconds) (Mohnshausen *et al.*, 2010). Secondly, auxin application to the outermost root tip elicits a calcium wave propagating shootwards. A rise in the apoplastic pH propagates along the root in roughly the same time-frame. The speed of both is consistent with the rate of auxin transport, and both are dependent on auxin transport (Mohnshausen *et al.*, 2010).

Thirdly, gravistimulation triggers a shootward mobilised calcium wave and apoplastic pH gradient that propagates at the approximate rate of the auxin gradient. Both are shown to be dependent on auxin transport (Mohnshausen *et al.*, 2010; S. Gilroy, unpublished)

Auxin-induced alkalisation of the apoplast is placed downstream of the increase in cytosolic calcium. This is based on the fact that apoplast alkalisation is blocked by treatment with a calcium channel blocker. Changes in the pH of the apoplast regulates the enzymatic activity of xyloglucan endo-

transglycosylase/hydrolase (XTH) and expansins (EXP), which are both families of cell-wall remodelling enzymes (see Section 1.1.3.3.2).

1.1.3.3.2 Transcriptional Response In contrast to the non-transcriptional auxin response that is not well understood, the key players involved in the transcriptional auxin response are well characterised.

Transcriptional regulation by auxin relies on a subset of AUXIN RESPONSE FACTOR (ARF) transcription factors. ARF1 was identified biochemically based on its ability to bind the auxin response element TGTCTC, which is located within the promoters of early auxin-induced genes. Based on sequence homology, 22 full-length ARFs are encoded in the *Arabidopsis thaliana* genome. There are three types of *Arabidopsis thaliana* ARF proteins: type I (ARF5,6,7,8,19), type II (ARF1,2,4,9-12,14-16,18,20-22), and type III (ARF3,13,17). All three types are characterised by an N-terminal B3-type DNA-binding domain (Guifoyle and Hagen, 2007) (see Figure 1.7). Type I and II ARFs contain a C-terminal PB1 protein-protein interaction domain (Guifoyle, 2015). Type II and III ARFs function as transcriptional repressors, while type I ARFs function as transcriptional activators. Type I and II ARFs are thought to dimerise preferentially with ARF proteins of the same type (Guifoyle and Hagen, 2007) (see Figure 1.7). Furthermore, type I ARFs strongly interact with AUX/IAA proteins, which like type I and II ARFs contain a PB1 protein-protein interaction domain. When bound to type I ARFs, AUX/IAA proteins function as transcriptional repressors, counteracting type I ARFs.

Auxin promotes the binding of TRANSPORT INHIBITOR RESPONSE 1 (TIR1) and AUXIN BINDING F-BOX (ABF) proteins to AUX/IAA proteins at domain II (DII). TIR1 and ABFs are part of the SCF E3 ubiquitin ligase complex, which catalyses the ubiquitylation of bound substrates, i.e. AUX/IAA proteins. Consequently, at higher auxin concentrations, AUX/IAA proteins become increasingly ubiquitinated, which targets them

for degradation by the 26S proteasome. This releases type I ARFs from repression by AUX/IAA proteins, thus initiating auxin-induced gene expression (see Figure 1.7).

Two redundant type I ARF proteins, ARF7 and ARF19, are thought to facilitate the genomic auxin response during gravitropism (Weijers *et al.*, 2005). ARF7 and ARF19 are known to synergistically control the expression of xyloglucan endo-transglycosylase/hydrolase 18 (XTH18), in addition to other cell-wall remodelling enzymes (Okushima *et al.*, 2005). Therefore, it appears that the initial cell-wall remodelling that occurs as part of the non-transcriptional response is continued and modified by the transcriptional response (Swarup *et al.*, 2013).

It is known that auxin regulates plant growth both transcriptionally and non-transcriptionally. Fedrych *et al.* (2016) show that the canonical TIR1-AUX/IAA-ARF module has a non-transcriptional signalling component. It is not clear, whether this non-transcriptional component of auxin signalling is also involved in root gravitropism.

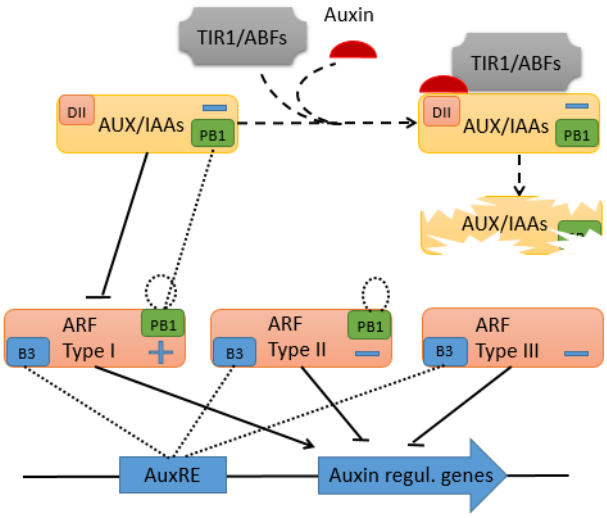


Figure 1.7: Schematic of the genomic auxin response based on ARF and AUX/IAA transcription factors. ARF transcriptional activators (type I) and ARF transcriptional repressors (type II and III) bind auxin response elements to regulate gene expression. Both type I and II ARFs form homodimers (dashed circles), while only type I ARFs form heterodimers with AUX/IAA transcriptional repressors (dashed line). Binding of type I ARFs by AUX/IAAs counteract their transcriptional activation. Repression of type I ARFs by AUX/IAAs is relieved under elevated auxin conditions via auxin-dependent degradation of AUX/IAAs, which is facilitated by TIR1/ABFs.

1.1.4 Methods in Gravitropic Research

Reviews on gravitropic research often quote Charles Darwin and his son (Darwin and Darwin, 1880) as the pioneers of gravitropic research. It is often ignored that Thomas Knight published a letter discussing root gravitropism 74 years before the Darwins (Knight, 1806). Thomas Knight published a description of the root gravitropic response. The Darwins included a drawing of a *Vicia faba* radicle (embryonic root) bending towards gravity. Simple images, like Darwin's drawings, are sufficient to document the gravitropic phenotype of very severe mutants. However, gravitropic research often requires the quantification of root gravitropism.

Over the past two decades three approaches of reporting root gravitropic behaviour have been widely adopted in this field (see Figure 1.8). The first approach is often referred to as a gravitropism diagram. Gravitropism

diagrams do not quantify the gravitropic response nor do they allow allow for statistical analysis. To create a gravitropism diagram each gravistimulated root is assigned one of (typically) 12 sectors on a circle (30 degree per sector). On each sector a bar is drawn, the length of which represents the percentage of seedlings whose root tip is in that sector. Only one time-point after gravistimulation is shown in such a gravitropism diagram. The second approach is a simple bar chart showing the mean angle of root tips at a certain time-point after gravistimulation. Error bars typically show plus/minus standard error (SE). This approach is often accompanied by a statistical analysis, often using t-tests (see Figure 1.8). The third approach is a extension of the second approach that looks at multiple time-points, which are shown in a scatter plot. Like for the bar charts, error bars in the scatter plots usually represent plus/minus SE and t-tests are the statistical test of choice (although rarely provided) (see Figure 1.8).

While gravitropic diagrams provide a clear visualisation of the gravitropic behaviour of a root, they are themselves not sufficient as they do not include a statistical analysis. The error bars typically provided for bar charts and scatter plots are not sufficient to indicate statistically significant differences. Instead plus/minus two-times SE error bars might be used to indicate the 95% confidence interval. However, a statistical analysis is be preferable. Neither of the approaches is typically accompanied by an parallel analysis of the root growth rate. It therefore often remains elusive whether a detected gravitropic defect is only a secondary effect of a defect in growth rate. Further, the statistical test used in scatter plots only consider single time points and not the whole time series dataset. The omission of growth rate and the lack of a approach that includes all time-series data-points in the statistical mark gaps the approaches currently used in gravitropic research.

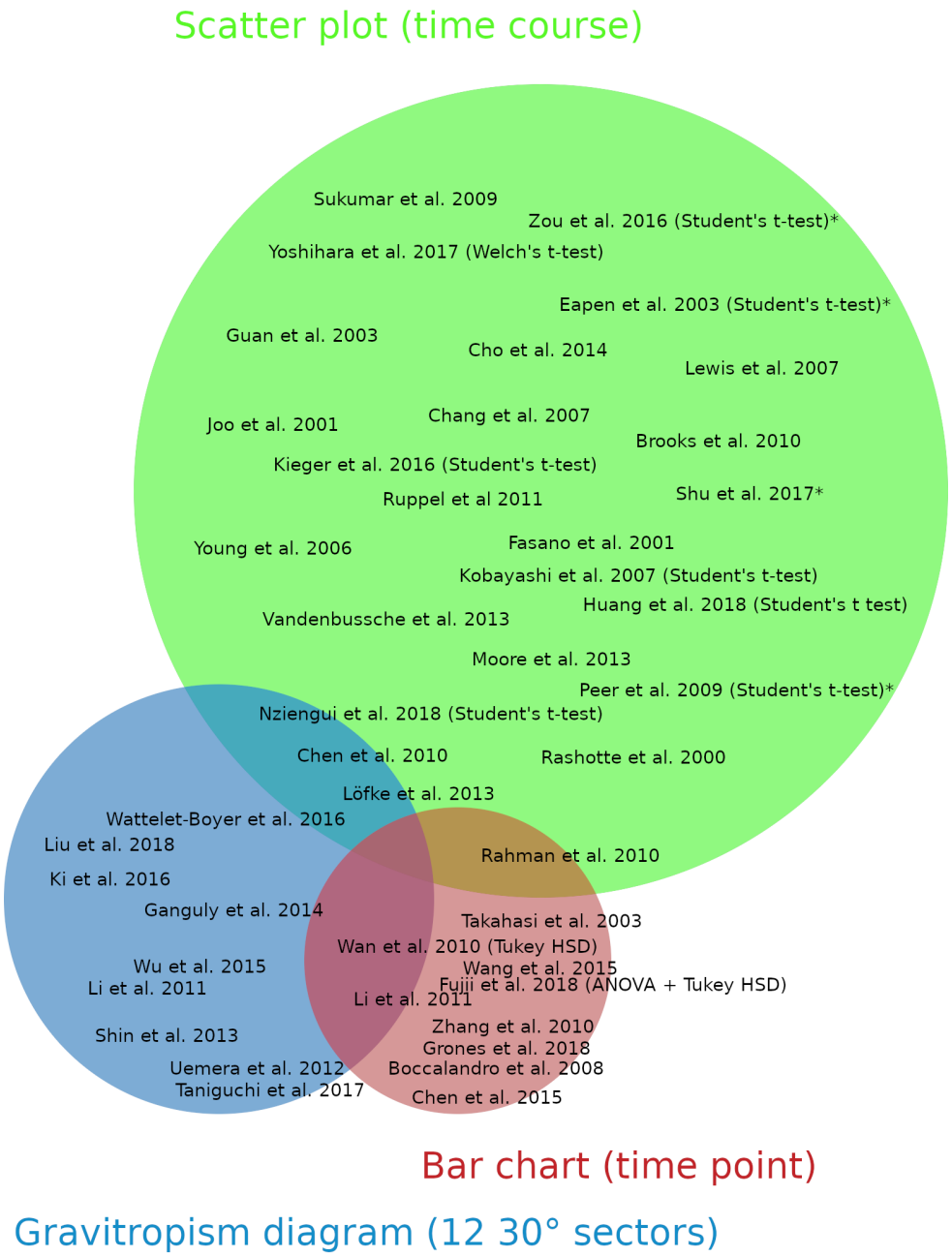


Figure 1.8: Venn diagram of the research methodology used in 51 papers from 2000 until 2018. Statistical tests are given in braces, if provided. Both scatter plots and bar charts usually give standard error error-bars. Star (*) indicates standard deviation error-bars. Publications in overlapping regions use multiple methods. See References for full references.

1.2 Kinases in Gravitropism

1.2.1 Kinase Function and Regulation

1.2.1.1 Function and Target-Specificity of Eukaryotic Kinases

Kinases function in signal transduction by catalysing the transfer of a phosphate from ATP or GTP to a serine (S), threonine (T), or tyrosine (Y) on a substrate protein. The added phosphate modifies the function of the substrate by changing its conformation and/or its interaction with other proteins or ligands. Protein kinase function is tightly regulated on two levels: activity and specificity (Sections 1.2.1.2, 1.2.2.1, and 1.2.3.1). Kinase specificity relies on two mechanisms. The first mechanism is the recognition of a consensus sequence around the phosphorylated residue, subsequently referred to as the phosphorylation site (phospho-site). Phospho-site recognition (preference) is facilitated by a subset of amino acids in the kinase domain. The second mechanism ensuring kinase specificity is the binding domains outside of the kinase domain (kinase) to domains distal to the phospho-site (substrate). This mechanism is often referred to as docking (Turk, 2008).

Kinase phospho-site recognition has been well researched. One popular methodology is the generation of Position-Specific Scoring Matrices (PSSMs) that capture the phospho-site preference of a kinase domain. Using structural studies that capture kinase domains interacting with target phospho-sites, Brinkworth *et al.* (2004) were able to generate an algorithm to predict the optimal phospho-site for any given kinase domain. This algorithm makes use of amino acids adjacent to the substrate phospho-site in a kinase-substrate complex, which are thought to form Phospho-site Residue "Binding Pockets" (PRBPs). As the sequence and structure of kinase domains is strongly conserved, the structure of one kinase-substrate complex is sufficient to pin-point which amino acids form PRBPs.

For a decade it remained unknown whether residues distal to the PRBPs are also involved in phospho-site recognition. In a critical paper, Creixell *et al.* (2015) utilise a novel evolutionary algorithm that scores every kinase domain residue based on its importance in phospho-site recognition. Residues with a high score (Domains of Specificity / DoS) are highlighted in Figure 1.9. The algorithm utilises a dataset of kinases and their experimentally determined PSSMs. DoS are determined based on the logic that kinases, which exhibit similarity in the residues that are important for phospho-site recognition, have similar PSSMs. The authors were able to generate a phylogeny of the human kinome based on DoS, which reflects how human kinases are related in terms of peptide specificity.

1.2.1.2 Regulation of Eukaryotic Kinases

The role of eukaryotic Ser/Thr protein kinases in triggering signalling cascades requires them to be tightly regulated. Some regulatory mechanisms are shared by all eukaryotic Ser/Thr protein kinases, whereas others are specific to certain groups or families of kinases. The activity of eukaryotic Ser/Thr protein kinases depends on a regulatory-spine (R-spine) of hydrophobic residues, which bridges the catalytic cleft between the N-lobe and the C-lobe (see Figure 1.10B). The R-spine consists of four hydrophobic residues, two from the N-lobe, one from the C-lobe, and one from the Activation Loop (AL), which is a structurally diverse section of the C-lobe (Taylor and Kornev, 2011).

A kinase domain can divided into 12 subdomains. These subdomains were identified based on their primary sequence conservation, with the unique functions of each subdomain being subsequently identified. The AL is part of the Activation Segment (AS), which is defined as the sequence between the conserved DFG/D motif in subdomain VII and the conserved APE motif in subdomain VIII. In most kinases, phosphorylation of a spe-

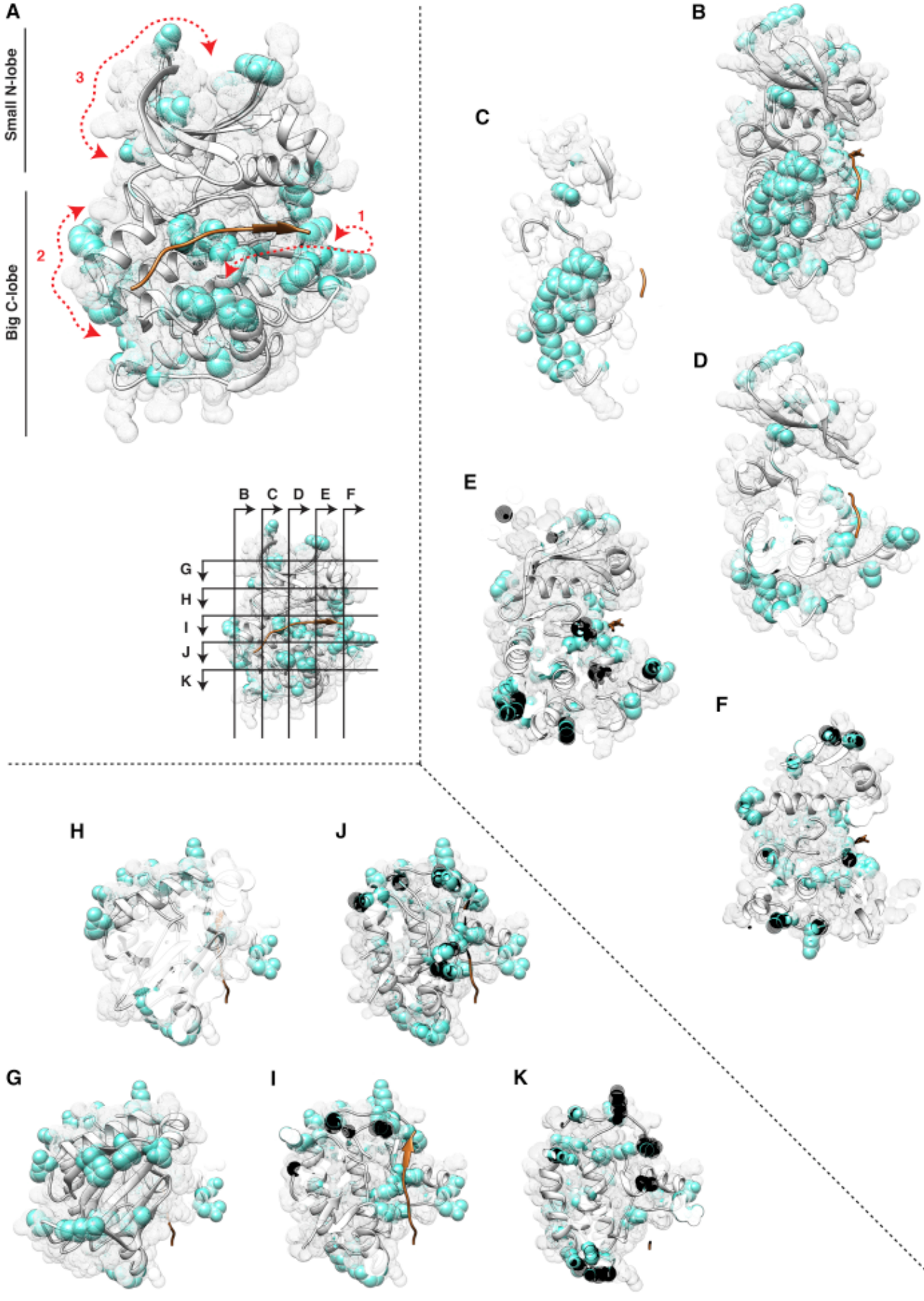


Figure 1.9: *(legend on next page)*

Figure 1.9: (A) Akt/PKB kinase domain 3D-model in complex with a substrate peptide (GSK3, orange) (PDB ID: 1O6K; Yang *et al.*, 2002). Domains of Specificity (DoS) that scored higher than 0.9 in the KINspect algorithm are highlighted (cyan). Arrows (red) outline three canals formed by these DoS. See (B-F) for longitudinal and (H-K) for transversal slices of (A). From Creixell *et al.* (2015).

cific residue on the AL will initiate phosphate bridging to an Arg in the VIb subdomain of the C-lobe. Phosphate bridging will position a hydrophobic residue from the AL to allow for formation of the R-spine (see Figure 1.10A, B). Phosphorylation of the AL therefore regulates kinase activity by initiating formation of the R-spine. (Nolen *et al.*, 2004) (Taylor and Kornev, 2011).

Formation of the R-spine, and therefore kinase activity, is also dependent on the correct positioning of a hydrophobic residue within subdomain III of the N-lobe. Subdomain III corresponds to the α-helix (αC) that is positioned towards the catalytic cleft. Comparative surface geometry of eukaryotic Ser/Thr protein kinases revealed two conserved hydrophobic pockets adjacent to the αC-helix (see Figure 1.10A, C). Both pockets require binding by small hydrophobic motifs in order to direct αC-helix positioning for R-spine formation. Although the hydrophobic pockets are conserved in eukaryotic Ser/Thr protein kinases, the way in which they are bound differs between kinase groups and families (Taylor and Kornev, 2011).

1.2.2 AGC-Kinases in Gravitropism

1.2.2.1 Regulation of AGC-Kinases

Most kinases in the AGC group contain a hydrophobic motif (HM) at the C-terminus. The HM binds one of the hydrophobic pockets adjacent to the αC-helix (see Section 1.2.1.2). Not only does this interaction contribute to the formation of the R-spine, but it also wraps the region between the kinase domain and the HM (C-tail) tightly around both kinase lobes (see

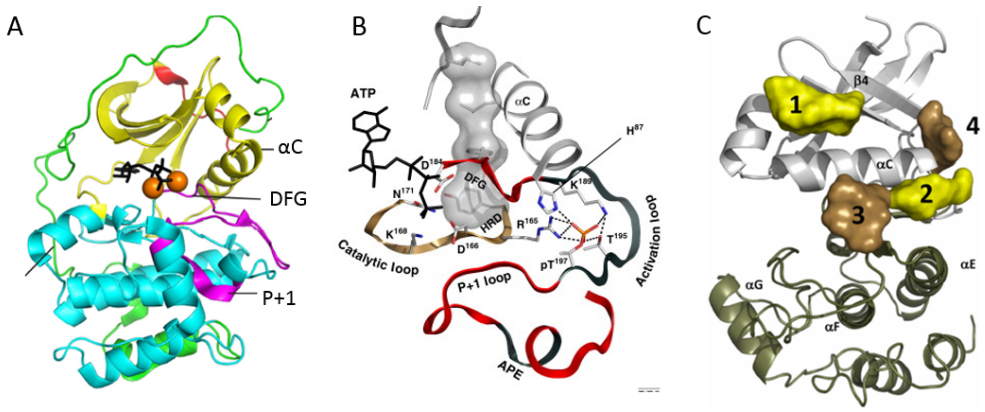


Figure 1.10: (A) Conserved 3D structure of AGC-VIII kinase domain. Yellow indicates N-lobe and blue indicates C-lobe. Position of αC-helix is highlighted. (B) Phosphorylated activation loop binding to Arg in the VIb subdomain of the C-lobe, which allows for formation of the R-spine (grey shell). Position of αC-helix indicated (C) Conserved hydrophobic pockets (1-4) surrounding the αC-helix. Adapted from Rademacher and Offringa (2012) and Taylor and Kornev (2011).

Figure 1.11). In AGC kinases, this C-tail has co-evolved with the kinase domain in a manner that makes catalysis dependent on certain residues of the C-tail (Kannan *et al.*, 2007).

In addition to binding the hydrophobic pocket of the same protein, the HM motif may also bind the hydrophobic pockets of other AGC kinases (see Figure 1.11). In mammals, interaction between the hydrophobic pocket of the AGC kinase PDK1 with the HM motif of a different AGC kinase is a highly conserved mechanism of AGC-kinase regulation. Upon binding to the HM domain of target AGC kinases, PDK1 phosphorylates the AL to prime the kinase for catalysis (see Figure 1.11B). In plants, PDK1 is only fully activated upon interaction with phosphatide acid (PA), a lipid found in specific domains of the PM (Anthony *et al.*, 2004). PA binding is facilitated by a pleckstrin homology (PH) domain in the C-tail of PDK1, which acts as a repressor when not bound to PA (Rommel *et al.*, 2010).

Most mammalian AGC-kinases are targeted to specific domains of the PM by PDK1-interaction. Other AGC kinases co-localise with PDK1 independently. Interestingly, the mammalian AGC kinase PKB lacks the HM

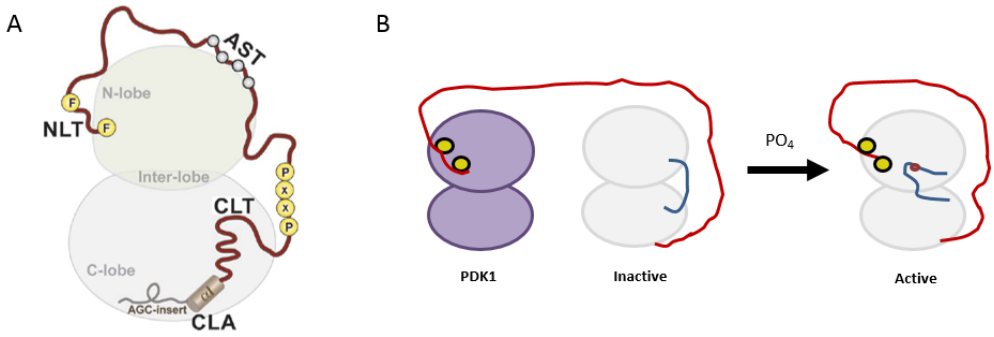


Figure 1.11: (A) Schematic of the AGC C-tail highlighting regions that interact with the kinase domain. Yellow circles represent conserved PxxP and hydrophobic motifs (F). (B) Activation of AGC kinases by docking to PDK1 via the hydrophobic motif (yellow circles) located on the C-tail (red line). PDK1 phosphorylates (red dot) the activation loop (blue line) of the interacting AGC kinase. Following phosphorylation by PDK1, the C-tail of the AGC kinase binds its own kinase domain. Adapted from Kannan *et al.* (2007).

motif and instead activation by PDK1 occurs via an alternative mechanism. This mechanism is dependent on co-localisation, which is facilitated by the PH-domains present in both PDK1 and PKB (Rommel *et al.*, 2010). A similar mechanism could be present in plants.

1.2.2.2 Phylogeny of AGC-Kinases

AGC-VIII kinases, which were identified in flowering plants, contain a characteristic insertion domain in their activation segment (see Section 1.2.1.2). Orthologs of the AGC-VIII kinase PHOTOTROPIN2 (PHOT2) were subsequently found in green algae. PHOT2 is involved in chloroplast movement in response to light, a mechanism essential to both plants and unicellular green algae. Other AGC-VIII kinases have functions associated with multicellularity and are therefore thought to have evolved from an ancestral PHOT2 (Galvan-Ampudia and Offringa, 2007). PHOT2 is a specialised AGC-VIII kinase that functions as a photoreceptor. It contains two flavin mononucleotide (FMN) binding domains positioned N-terminal to the PHOT2 kinase domain. These FMN domains undergo a conformational change upon blue light irradiation, which activates the kinase (Christie, 2007). All other

AGC-VIII kinases, excluding PHOT1, do not function as photoreceptors and have lost their FMN-binding domains (Galvan-Ampudia and Offringa, 2007). It is likely that novel plant-specific regulatory domains have replaced these N-tail domains.

The AGC-VIII kinases form four different subfamilies based on their kinase domain sequence (see Figure 1.12). PHOT2 and the homologous PHOT1 make up the AGC-VIII-4 subfamily (Galvan-Ampudia and Offringa, 2007). Kinases in the AGC-VIII-2 subfamily have diverged from all other AGC-VIII kinases and do not contain the conserved AEP motif in the AGC-VIII activation segment insert. Two kinase in the AGC-VIII-2 subfamily, OXI1 and UCN (AGC2-3), offer a good example of how AGC-VIII kinases function in a variety of signalling pathways. OXI1 is involved in oxidative burst signalling, while UCN functions in the regulation of cell growth in the integuments (Galvan-Ampudia and Offringa, 2007). The remaining two subfamilies of AGC-VIII kinases, AGC-VIII-1 and AGC-VIII-3, are more closely related to AGC-VIII-4 (PHOT2 and PHOT1)(Galvan-Ampudia and Offringa, 2007) (see Figure 1.12).

AGC kinase activation does not rely on the binding of other proteins to the kinase domain hydrophobic pockets. Instead, a hydrophobic motif at the N-terminal of AGC kinases fulfils this function (see Section 1.2.2.1). This feature is unique to AGC kinases and allows them to function independently of certain regulatory pathways that are essential to other eukaryotic protein kinases (EPKs). AGC-VIII kinases specifically are capable of auto-phosphorylating their own activation loop (AL). Plant AGC-VIII kinases have evolved from an ancestral form of the photoreceptor PHOT2. Unlike most other AGC kinases, PHOT2 is not activated by PDK1 and is instead capable of auto-phosphorylating its own AL (Christie, 2007). Auto-phosphorylation of the AL is conserved in members of the AGC-VIII-1 (KIPK) and AGC-VIII-3 (PID, WAG1 and WAG2) subfamilies (Zegzouti

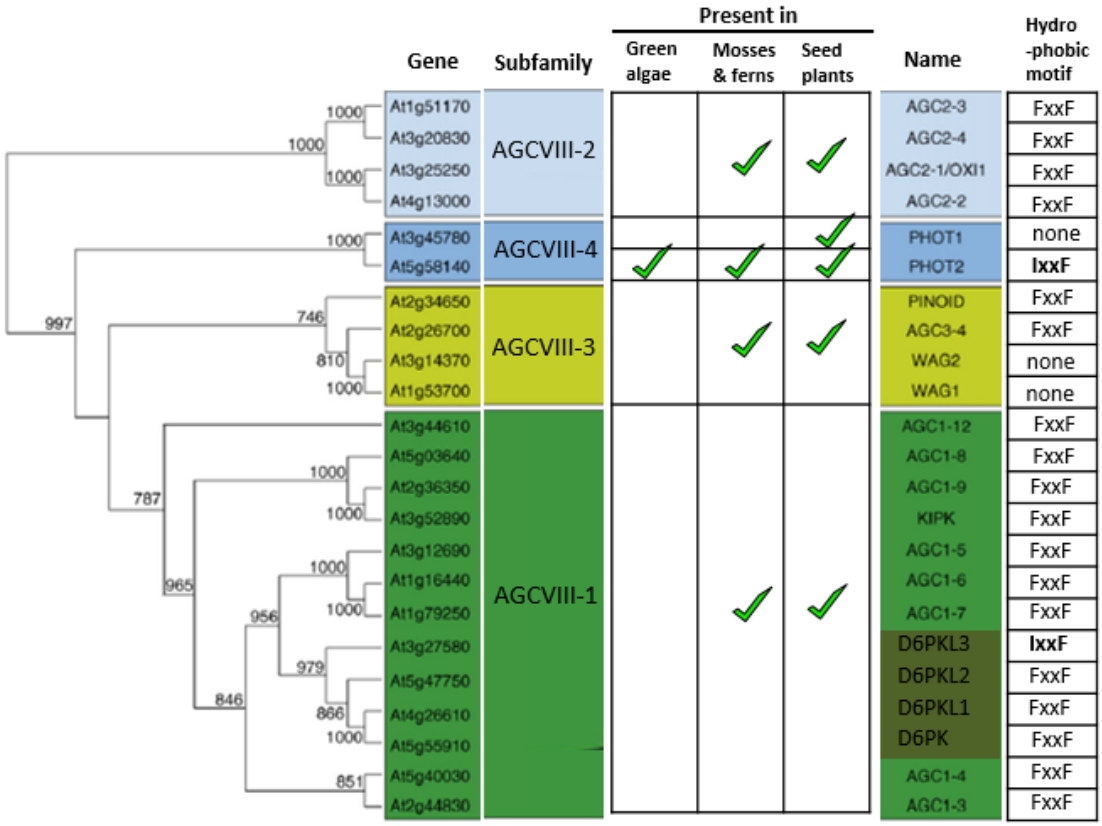


Figure 1.12: Phylogenetic tree of AGC-VIII kinases. Ticks indicate conservation in green algae, mosses, ferns and seeded plants. Right-hand column states whether the hydrophobic motif is absent (none) or whether it is the consensus sequence (FxxF). PHOT2 and D6PKL3 are the only AGC-VIII kinases to retain a diverged hydrophobic motif (IxxF). Adapted from Galvan-Ampudia and Offringa (2007) and Zegzouti *et al.* (2006a).

et al., 2006a). AL auto-phosphorylation allows AGC-VIII kinases to bypass another common regulatory mechanism of EPKs, in addition to the binding of their own hydrophobic motif. The primary regulatory mechanism of AGC-VIII kinases is thought to be protein targeting, which limits substrate accessibility.

Consistent with the notion that protein targeting is central for the regulation of AGC-VIII kinases, it appears to play a key role in PHOT2 signalling. Blue light induced auto-phosphorylation of PHOT2 coincides with the dissociation of PHOT2 from the PM and subsequent association with the Golgi apparatus. Binding of PHOT2 to the Golgi apparatus is thought to facilitate its co-localisation with substrates (Christie, 2007).

1.2.2.3 Targeting of AGC-Kinases

Protein targeting, a key mechanism regulating PHOT2, appears to be a conserved regulatory mechanism of AGC-VIII kinases (see Section 1.2.2.1). The domains facilitating protein targeting have been mapped for PID and ADI3 (tomato ortholog AGC1-3) (Gray *et al.*, 2013). The AGC-VIII-specific activation loop (AL) inserts of PID and ADI3 are sufficient for targeting to the PM and nucleus, respectively (Zegzouti *et al.*, 2006a, Ek-Ramos *et al.*, 2010). Three lines of evidence suggest that the AGC-VIII-specific AL insert of PHOT2 dynamically targets it to both the PM and Golgi apparatus. Firstly, the majority of the PHOT2 N-tail is not required for PM and Golgi targeting (see Figure 1.13A) (Kong *et al.*, 2006). Secondly, unphosphorylated (kinase dead) PHOT2 preferentially localises to the PM (in an N-tail independent manner) (see Figure 1.13A) (Kong *et al.*, 2006). Thirdly, the AL undergoes a conformational change upon phosphorylation, which likely alters the conformation of the AGC-VIII-specific AL insert (see Figure 1.13B) (Taylor *et al.*, 2012). Conformational changes in the insert could explain how phosphorylation effects the protein targeting of PHOT2.

Although the AL insert is the primary contributor to protein-targeting, the highly variable N-tails of AGC-VIII kinases potentially contain additional regulatory domains involved in protein targeting. Sequences at the AGC-VIII C-tail are subject to evolutionary constrains due complementation of the kinase domain (see Section 1.2.2.1). This makes it less likely that the C-tail contains regulatory regions, apart from the hydrophobic motif (HM). Unlike PHOT2, most AGC-VIII kinases contain a consensus HM involved in PDK1 interaction (see Figure 1.12) (Zegzouti *et al.*, 2006a). The HMs of PID and KIPK interact with PDK1; however, only in the case of PID does this interaction lead to an increase in the phosphorylation of the PID AL (Zegzouti *et al.*, 2006b, Zegzouti *et al.*, 2006a). PDK1 is a PM-associated protein and PDK1-interaction may contribute to the tar-

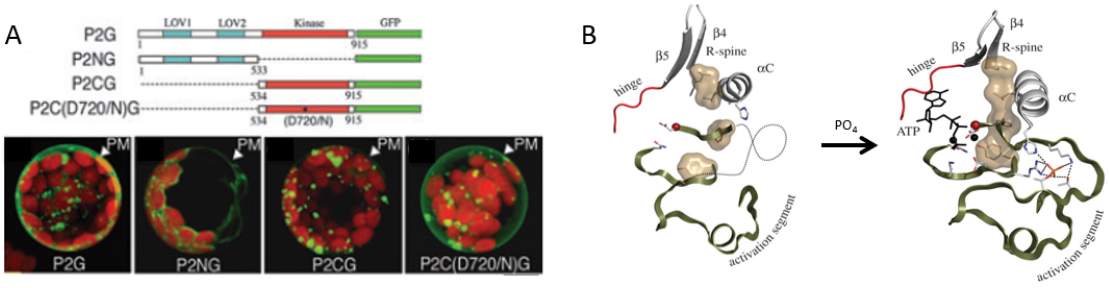


Figure 1.13: (A) Subcellular localisation of full-length PHOT2 protein (P2G) and various truncated PHOT2 proteins (P2NG, P2CG and P2C(D720/N)G) five minutes after blue light irradiation. Full-length PHOT2 (P2G) accumulates at PM and Golgi. The N-tail domain (P2NG) is observed throughout the cytoplasm. The C-tail and kinase domain (P2CG) accumulates at Golgi and faintly at the PM. The kinase-dead/unphosphorylated kinase domain plus C-tail (P2C(D720/N)G) shows a more pronounced accumulation at the PM than P2CG. (B) Schematic of the conformational change in the activation segment after phosphorylation. Adapted from Kong *et al.* (2006) and Taylor *et al.* (2012).

geting of some AGC-VIII kinases to the PM (see Section 1.2.2.1). D6PKs have a highly conserved AL insert, suggesting that protein targeting is conserved in the clade. The founding member, D6PK, is specifically targeted to the root-ward PM (Zourelidou *et al.*, 2009), making it possible that the other D6PKs are also targeted to this domain. D6PKL3 lacks the consensus C-terminal HM (see Figure 1.12), which suggests that this kinase is regulated in a PDK1-independent manner. However, like all other D6PKs, D6PKL3 phosphorylation is significantly up-regulated by PDK1 (Zegzouti *et al.*, 2006a).

1.2.2.4 The Role of PINOID in Gravitropism

1.2.2.5 PINOID-Mediated Phosphorylation of PIN2 in Auxin Gradient Re-configuration

The auxin efflux carrier PIN2 is important for the rate and direction of auxin transport during the re-configuration of the auxin gradient (see Section 1.1.3.2). Both functions are dependent on the polar targeting of PIN2 to the shootward PM, which is thought to be controlled by PINOID (PID).

Four lines of evidence suggest that PID-mediated phosphorylation of PIN2, and other PIN proteins, targets it to the shootward PM (Friml *et al.*, 2004). Firstly, the over-expression of PID leads to a root- to shootward localisation shift of PIN2 in the root cortex (see Figure 1.14A). Similarly, PID over-expression results in a PIN1 root- to shootward localisation shift in the root stele (Friml *et al.*, 2004). Secondly, in the shoot apical meristem of PID over-expression lines, PIN1 shows a shoot- to rootward localisation shift (Friml *et al.*, 2004) (see Figure 1.14B). Thirdly, PIN1 mutant proteins that cannot be phosphorylated by PID (phospho-mutant) are targeted to the rootward PM in the apical meristem. In contrast, PIN1 mutant proteins that behave like PID-phosphorylated PIN1 (phospho-mimic) are targeted to the shootward PM of shoot apical meristem cells (Huang *et al.*, 2010)(see Figure 1.14C). Fourthly, in the *pid-9* loss of function mutant, PIN2 accumulates in endomembrane structures (see Figure 1.14A) (Sukumar *et al.*, 2009).

The increased PIN2 accumulation on the shootward PM on the bottom side of a gravi-stimulated root is thought of be auxin-dependent (see Section 1.1.3.2) (Abas *et al.*, 2006). PID expression is highly auxin-inducible, suggesting auxin up-regulates its own rate of transport via the pronounced shootward targeting of PIN2 (Benjamins *et al.*, 2003). Indeed, the *pid-9* loss of function mutant shows an approximate 50% reduction in the gravitropic bending rate (Huang *et al.*, 2010). The weak phenotype of the *pid-9* mutant could be due to redundancy with PID-LIKE1, WAG1 and/or WAG2, which are assumed to have a conserved biochemical function (Haga *et al.*, 2014).

The primary function of the PID-family of AGC-VIII kinases has long been thought to be the targeting of PIN proteins to the shoot-ward PM. However, PID-mediated phosphorylation of PIN proteins has recently been shown to directly activate PIN proteins (Zourelidou *et al.*, 2014). Indeed the PID loss of function mutants and PID overexpression lines have altered

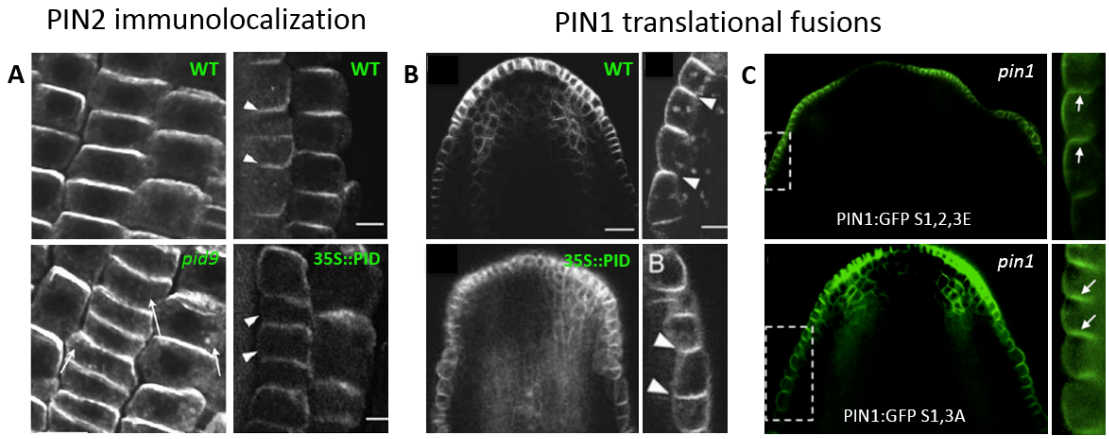


Figure 1.14: Evidence that PID-mediated phosphorylation of PINs targets them to the shoot-ward PM. (A) PIN2 immunolocalisation in WT (top), *pid9* (bottom left), and 35S::PID line (bottom right). The *pid9* mutant shows accumulation of PIN2 in endomembrane structures of root cells. The over-expression line shows root- to shootward localisation shift of PIN2 in root cortex. (B) PIN1 translational fusions in WT (top) and 35S::PID background (bottom). The over-expression line shows shoot- to rootward localisation shift. (C) Phospho-mimic (top) and phospho-mutant (bottom) for PID phospho-sites of PIN1 translational fusions. Phospho-mimic accumulates at shoot-ward PM of shoot apical meristem cells, while phospho-mutant accumulates at rootward PM of shoot apical meristem cells. Adapted from Friml *et al.* (2004), Huang *et al.* (2010), and Sukumar *et al.* (2009).

PIN activity and auxin distribution. Irrespective of whether PID-mediated phosphorylation targets PIN2 to the shootward PM or activates it on the shootward PM, PID could contribute to the up-regulation of auxin transport on the lower side of the gravi-stimulated root.

1.2.2.6 The Non-Transcriptional Regulation of PINOID in Early Auxin Gradient Re-configuration

PID is important for the increase of PIN2 accumulation at the shootward PM on the bottom side of a gravi-stimulated root. Auxin-induction of PID expression could contribute to the up-regulation of auxin transport on the bottom side of the gravi-stimulated root. However, this transcriptional control of PID is too slow to account for the early stages of auxin gradient re-configuration. The auxin gradient is thought to reach the elongation zone

roughly six minutes after gravi-stimulation. In comparison, PID transcription is induced roughly 15 minutes after auxin treatment. It is therefore only possible that auxin-induced PID expression is involved in the later stages of auxin gradient re-configuration.

It is not plausible that transcriptional regulation by auxin (transcriptional response) is involved in regulating the earlier stages of auxin gradient re-configuration. Therefore, these earlier stages must be regulated via the non-genomic auxin response. An increase in cytosolic calcium is a well-known part of the non-transcriptional auxin response (see Section 1.1.3.3.1). Benjamins *et al.* (2003) provide two lines of evidence that PID activity is regulated by calcium. Firstly, PID over-expression lines treated with calcium channel blockers (calcium response inhibitors) exhibit an increased tendency of root meristem collapse. Secondly, TCH3, a calmodulin-related protein, interacts with PID in a calcium-dependent manner. It is therefore possible that PID is not only part of the genomic auxin response, but also the non-genomic auxin response.

1.2.3 CIPK-Kinases in Gravitropism

1.2.3.1 Regulation of CIPK-Kinases

The kinase domains of CaMK kinases are thought to have diverged from other eukaryotic protein kinases to accommodate for auto inhibitory sequences (AISs) in the catalytic cleft (Taylor and Kornev, 2011) (see Section 1.2.1.2). CaMK kinases consist of two main families: the CaMK-I family and the SNF1/AMPK family (Hanks and Hunter, 1995). The founding members of both kinase families, human CaMK1 and AMPK, are regulated by AISs (Goldberg *et al.*, 1996, Chen *et al.*, 2009). In plant specific CIPKs, which are part of the SNF1/AMPK family, the NAF domain in the C-tail functions as AIS (Guo *et al.*, 2001) (see Figure 1.15).

The NAF domain forms a helix-loop-helix segment, reminiscent of the

CaMK1 AIS (see Figure 1.15B). The first helix of both the NAF domain and the CaMK1-AIS domain binds at the substrate binding site. The second helix of both AIS domains binds within proximity of the ATP binding pocket (Chaves-Sanjuan *et al.*, 2014)(see Figure 1.15B). It is likely that these similarities are the result of convergent evolution. AMPK, which is more closely related to CIPK than CaMK1, contains a structurally different AIS (Chen *et al.*, 2009). Consistent with their analogous nature, the NAF and CaMK1-AIS domains are regulated by different mechanisms. The CaMK1 kinase is regulated by calmodulin, which binds in proximity of the AIS to inhibit kinase domain binding (Yokokura *et al.*, 1995). The NAF domain, on the other hand, binds directly to CBL proteins, and this interaction is thought to relieve the CIPK kinase from NAF-inhibition in a calcium-dependent manner (Albrecht *et al.*, 2001).

Next to the NAF domain, the C-tail contains a conserved protein phosphatase interaction (PPI) motif (see Figure 1.15A). The PPI motif was identified in the CIPK kinase SALT OVERLY SENSITIVE (SOS2) as the site of interaction with the protein phosphatase ABA-INSENSITIVE2 (ABI2). A mutation that renders ABI2 incapable of interacting with SOS2 makes the plant more tolerant to salt shock. However, the mechanism of ABI2-SOS2 crosstalk remains elusive (Ohta *et al.*, 2003). The N-tail of CIPKs is very short and has been largely ignored.

Although the kinase domains of CIPK kinases have diverged from other eukaryotic protein kinases (EPKs), they contain conserved regulatory features (see Section 1.2.1.2). Like most EPKs, CIPKs are activated by phosphorylation of the activation loop. The CIPK activation loop contains three phosphorylation sites that have been implicated in the regulation of CIPK activity (Chaves-Sanjuan *et al.*, 2014).

Additionally, CIPKs resemble other EPKs by possessing two hydrophobic pockets adjacent to the αC-helix. Interestingly, in roughly half of CIPK

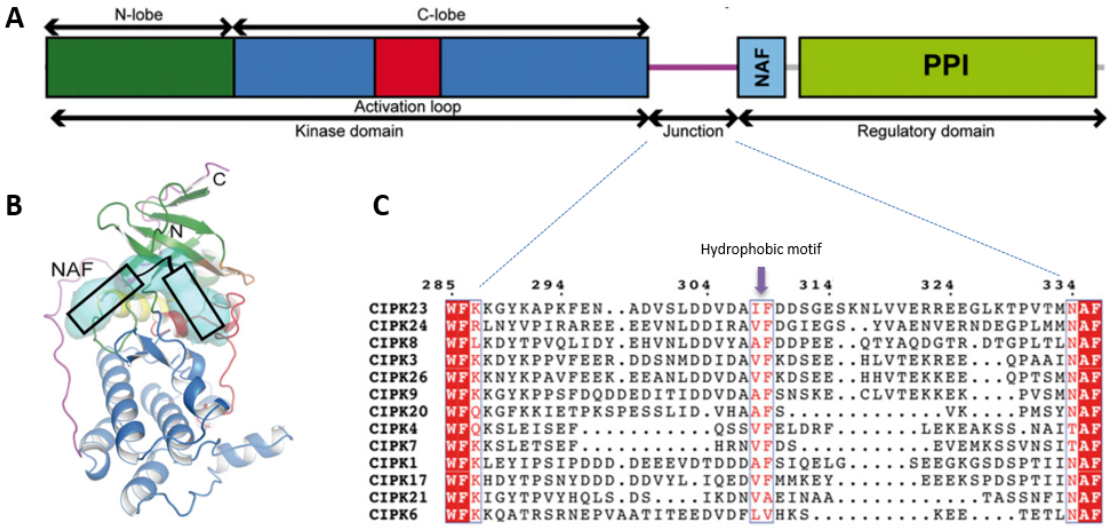


Figure 1.15: (A) Conserved 2D structure of CIPK kinases. NAF functions as auto inhibitory sequence. PPI interacts with protein phosphatase.(B) Conserved 3D structure of CIPK kinase domain illustrating that the hydrophobic pockets are occupied by the NAF domain, which inhibits the kinase. C and N-terminals are indicated. (C) Conservation of the hydrophobic motif in 13 *Arabidopsis thaliana* CIPK junction domains. Adapted from Chaves-Sanjuan *et al.* (2014).

kinases, one of these hydrophobic pockets is filled by a hydrophobic motif (HM) that is located in the C-tail (Chaves-Sanjuan *et al.*, 2014). This mechanism resembles the characteristic AGC kinase regulation by a C-terminal HM (see Section 1.2.1.2). This resemblance could reflect how closely related CaMK and AGC kinases are (Hanks and Hunter, 1995). However, the CIPK HM does have a different sequence and is positioned differently; between the kinase domain and the NAF domain rather than at the C-terminus (AGC group) (see Figure 1.15A,C).

1.2.3.2 Targeting of CIPK-Kinases

Regulation by an AIS is a conserved feature of CaMK kinases. The AIS of CIPKs is the NAF domain. Although the NAF domain structurally resembles other CaMK AISs, its regulation is unique to CIPKs. The NAF domain is regulated by interaction with plant specific calcium-binding proteins; the CBLs (Albrecht *et al.*, 2001) (see Figure 1.16B). All CBLs harbour four

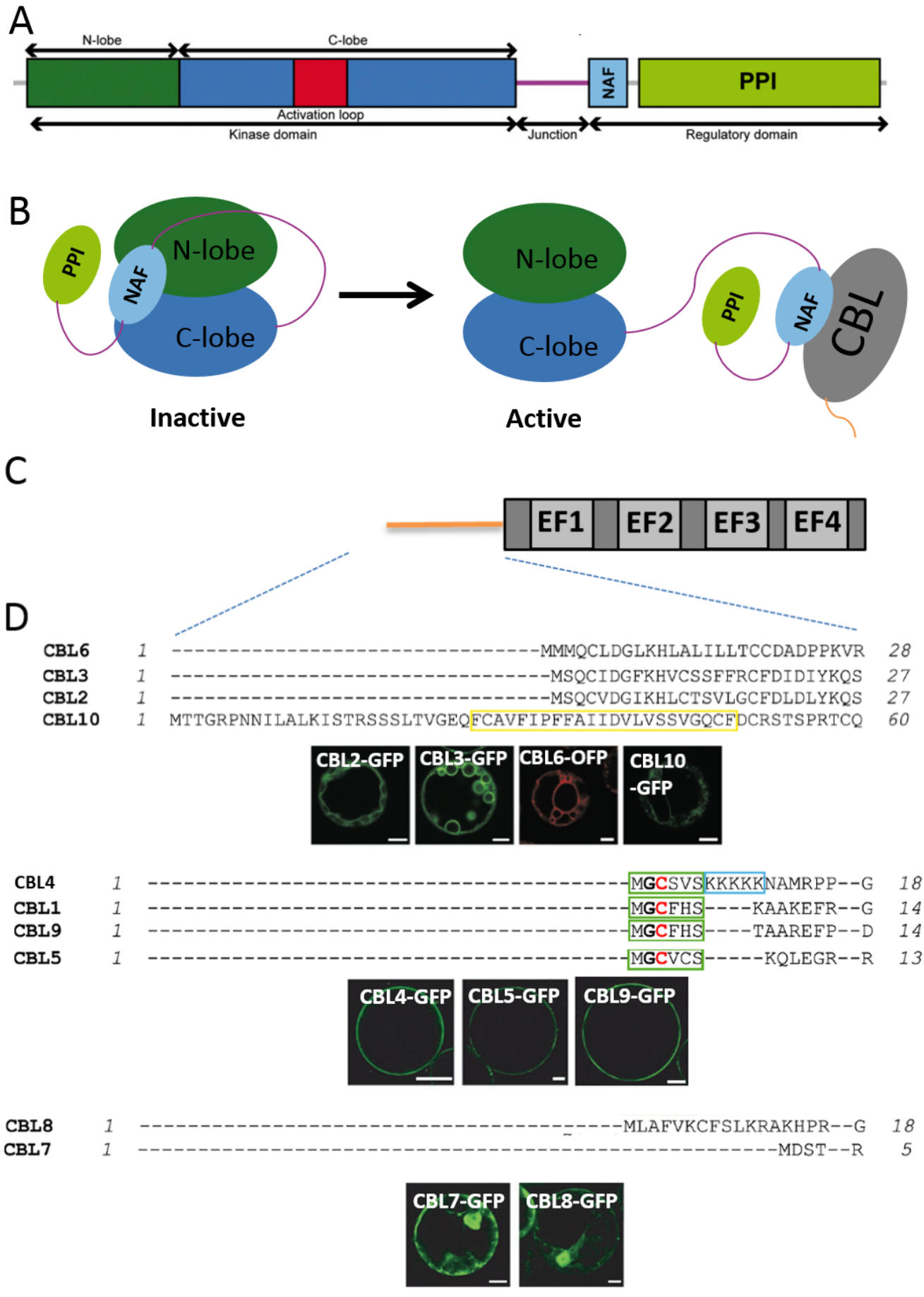


Figure 1.16: *(legend on next page)*

Figure 1.16: (A) Schematic of CIPK kinase structure. (B) Schematic of CBL proteins interacting with the NAF domain of CIPK proteins, which relieves the CIPK kinase domain (C- and N-lobe) from inhibition by the NAF domain. (C) Schematic of CBL protein structure showing the four EF-hands and indicating the variable N-tail (orange). (D) Sequence alignments of the N-tail of all *Arabidopsis thaliana* CBL proteins. Sequences cluster into three clades: long N-tail (CBL6,3,2,10), myrosylated/palmitoylated short N-tail (CBL4,1,9,5) and non- myrosylated/palmitoylated short N-tail(CBL8,7). The three CBL protein clades show conserved subcellular localisation to the tonoplast (long N-tail), PM (myrosylated/palmitoylated short N-tail), and the cytosol and nucleus (non- myrosylated/palmitoylated short N-tail); as demonstrated by GFP/OFP fusions in protoplasts. Adapted from Chaves-Sanjuan *et al.* (2014), Chaves-Sanjuan *et al.* (2013), and Batistic *et al.* (2010).

elongation factor-hands (EF-hands), which are conserved calcium-binding domains (see Figure 1.16C). Calcium regulates the activation of CIPKs via CBLs in two different ways. In CIPK1, calcium initiates CBL1-binding. Whereas in CIPK24, CBL4 is constitutively bound to the kinase. Independent of the mode of interaction, activity of the CIPK-CBL complex is dependent on calcium levels. EF-hands can be classed as canonical (more conserved) or non-canonical (less conserved). All ten *Arabidopsis thaliana* CBLs contain non-canonical EF-hands, however, only CBL1, CBL8, CBL9 and CBL10 contain canonical EF-hands. Canonical EF-hands have a higher affinity for calcium, suggesting that CBLs differ in their calcium-sensitivity. CBLs lacking canonical EF-hands are among the plant calcium-binding proteins with the lowest calcium-sensitivity (Sanchez-Barrena *et al.*, 2013).

The low calcium-sensitivity of some CBLs is linked to their subcellular localisation. CBL2, CBL3, CBL6 and CBL10, which have one (CBL10) or no (CBL2,3,6) canonical EF-hands, localise to the tonoplast (Batistic *et al.*, 2010) (see Figure 1.16D). In plants, the vacuole is a calcium store and the release of calcium from this compartment results in high calcium elevations in the proximity of the tonoplast. Interestingly, the tonoplast-localised CBLs form a phylogenetically distinct group within the CBLs as they contain an extended N-terminal domain. The N-terminal domain of CBL10 shows an

additional N-terminal extension, which is thought to accommodate a transmembrane helix. The remaining six CBLs form a separate phylogenetic group characterised by a short N-tail. Four of the short N-tail CBLs contain a myrosylation-palmitoylation motif, where a myrosylated glycine is adjacent to a palmitoylated cysteine (Sanchez-Barrena *et al.*, 2013). This dual fatty acyl modification determines the localisation of CBL1, CBL4, CBL5 and CBL9 to the PM (Batistic *et al.*, 2008) (see Figure 1.16D). The short N-tail CBL7 and CBL8 lack the myrosylation-palmitoylation motif and are localised to the cytosol and the nucleus (Batistic *et al.*, 2010)(see Figure 1.16D).

1.3 Aims and Objectives

Compared to other plant movements, gravitropism is initiated in a short time-frame. The gravitropic response is well studied at organ level, but less is known at the intracellular level. Kinases are capable of triggering rapid signal amplification and are therefore likely to fulfil intracellular functions during gravitropic signalling. One prominent example of a kinase functioning in gravitropic signalling is PID, which is thought to regulate PIN2 localisation during auxin gradient re-configuration. It is likely that multiple other kinases function in gravitropic signalling.

The primary aim of this thesis was to identify and evaluate kinases likely to function in gravitropic signalling (candidate kinases). The identification of candidate kinases and the subsequent gravitropic phenotyping required the creation of new tools. A large proportion of the thesis is therefore dedicated to introducing these tools.

The primary aim can be split into four sub-aims:

- **Aim 1:** To generate and employ a screening tool for the identification of kinases that are likely to function in gravitropism (see Chapter 3).

In Chapter 3 I identify three candidate kinases, D6PKL3, CIPK4 and CIPK7. D6PKL3 is identified as a candidate kinase based on its relationship to D6PK, a kinase known to function in auxin transport. CIPK4 and CIPK7 are identified as likely orthologs of maize CIPK15 (ZmCIPK15), which is a kinase associated with crown root angle. Preliminary analysis of the gravitropic phenotype of the corresponding mutants did not discover any gravitropic defect, which leads me to my second sub-aim:

- **Aim 2:** To generate a tool capable of detecting subtle and/or conditional gravitropic phenotypes (see Chapter 6).

The Introduction (Chapter 1) further revealed a clear lack of a analysis method that allows to for the analysis of time-series data of the gravitropic response in one statistical test. 'Aim 2' addresses this and other shortcomings in existing tools. The tool generated for 'Aim 2' was then used in conjunction with molecular and bioinformatic methods to address 'Aim 3' and 'Aim 4'.

- **Aim 3:** To determine whether D6PKL3 is involved in gravitropic signalling (see Chapter 4).

The Introduction (Chapter 1) showed the lack of knowledge about the molecular mechanism underlying the direct activation of PIN proteins in columella cells, which is needed to explain the rapid generation of the auxin gradient. 'Aim 3' directly addresses this gap in our understanding by assessing D6PKL3, which is likely to function in this pathway.

- **Aim 4:** To determine whether CIPK4 and/or CIPK7 are involved in gravitropic signalling (see Chapter 5).

In Chapter 3 I identify CIPK4/7 as putative orthologs of ZmCIPK15, which is known to be associated with the GSA of crown roots in maize (J. Lynch, personal comm.). GSA establishment uses pathways related to gravitropism (Roychoudhry *et al.*, 2013).

Chapter 2

Methods

2.1 Molecular Methods

2.1.1 DNA extraction

A small leaf sample was taken into a 1.5mL Eppendorf tube and flash frozen in liquid nitrogen. Samples were ground with sterile pestle, adding 100 μL of DNA Extraction Buffer (see Table 2.1). An additional 500μL of DNA Extraction Buffer were added and the samples and vortexed for five seconds. The samples were centrifuged at 13000 rpm for five minutes and the supernatant (500μL) transferred into a new Eppendorf tube. Equal volume (500μL) of isopropanol was added and mixed via inverting. Samples were stored at -20°C for 30 min and centrifuged at 13000 rpm for five minutes. The supernatant was discarded and the pellet washed with 500μL of 70% ethanol (re-centrifuged). Pellets were dried and re-suspended in 50μL sterile water.

Table 2.1: Buffers

DNA extraction buffer	200mM Tris-HCl, 250mM NaCl, 25mM EDTA, 0.5% SDS

Table 2.2: Standard PCR

DNA sample	1.0 μL
5x Mango buffer mix	2.0 μL
MgCl solution (50mM)	0.5 μL
Deoxynucleoside triphosphate (dNTP) mix (10 mM)	0.2 μL
Forward primer working solution (10 μM)	0.5 μL
Reverse primer working solution (10 μM)	0.5 μL
TAQ DNA polymerase (housemade)	0.2 μL
Sterile water	5.1 μL
	= 10.0 μL

2.1.2 Polymerase Chain Reaction

The PCR recipe is given in Table 2.2. The reaction mix was kept on ice whilst mixing. The TAQ DNA polymerase was added last.

PCR program was set up as follows (Gene Amp, PCR Systems 9700, Applied Biosystems):

94°C for 2 minutes

Start cycle (x35)

94°C for 30 seconds

54°C for 30 seconds

72°C for 1 minute

End cycle

72°C for 10 minutes

Hold at 4°C

2.1.2.1 Colony Polymerase Chain Reaction

For colony PCR the PCR protocol was followed (see Section 2.1.2), with 1.0 μL of "colony solution" being added instead of the DNA. The "colony solution" was made by touching a colony with a yellow pipette tip and then immersing in 10μL sterile water.

Table 2.3: Genomic DNA PCR

Arabidopsis thaliana (Col) whole seedling cDNA	1.0 μL
5x Pfx buffer mix	4.0 μL
Forward primer working solution (10 μM)	1.0 μL
Reverse primer working solution (10 μM)	1.0 μL
Pfx DNA polymerase	0.2 μL
Deoxynucleoside triphosphate (dNTP) mix (10 mM)	0.4 μL
Sterile water	12.4 μL
	= 20.0 μL

2.1.2.2 Polymerase Chain Reaction for Amplification of Sequences of Interest

The PCR reaction mix to amplify sequences of interest for cloning from *Arabidopsis thaliana* (Col-0) whole seedling gDNA (a gift from Dr. Ranjan Swarup) is shown in Table 2.3. The PCR reaction mix was mixed on ice. Pfx DNA polymerase was added second to last and gDNA added last. See Supplemental Table S2 for primers used to amplify from genomic DNA. Amplified sequences were used in the GreenGate cloning system (Lampropoulos *et al.*, 2013).

2.1.2.3 Reverse Transcription Quantitative Polymerase Chain Reaction (RT-qPCR)

Total RNA was extracted from whole seedlings (mutant characterisation) or roots (auxin induction), using the Qiagen RNeasy Plant Mini Kit. Poly(dT) cDNA was generated from 3 μg total RNA using the Transcriptor First Strand cDNA Synthesis Kit (Roche) by pooling three RT reactions. SYBR Green Sensimix (Quantace) was used to carry out Quantitative Polymerase Chain Reaction (qPCR) using 96-well optical reaction plates.

PCR program was as follows:

95°C for 5 minutes

Start cycle (x40)

95°C for 10 seconds

60°C for 30 seconds

72°C for 1 minute

End cycle

The results shown in figures are triplicates of qPCR reactions. See Supplemental Table S3 for primers used in qPCRs.

2.1.3 Gel electrophoresis

DNA samples were run on 1% agarose gel containing 1 μL/ mL ethidium bromide. Loading Buffer (6x) was added to DNA samples to create a 1x buffered solution, of which 6-12 μL was loaded onto the gel. 1 μL of 1kb plus ladder (Invitrogen TM) was loaded into a separate well. If the gel was to be used for gel extraction an empty well was left between the constructs. Gel ran at 90-100 V for 20 to 30 minutes.

2.1.4 Gel extraction

Band excision was carried out on a UV trans-illuminator. Excised bands were weighed before using the Gen Elute TM Gel Extraction Kit (Sigma-Aldrich, http://www.sigmaaldrich.com). This protocol involves the excised bands being solubilised, loaded onto a pre-treated binding column, and washed. The DNA was eluted in 30 μL of water.

2.1.5 Plasmid Methods

2.1.5.1 Plasmid Storage

Plasmids were stored in elution buffer at -20°C for direct use in a variety of protocols (working solution). For rapid recovery of working solutions, plasmids were also kept in *Escherichia coli* cells suspended in 16,66% glycerol at -70°C ("plasmid stocks"). "Plasmid stocks" were prepared by adding 400 μL of 50% glycerol to 800 μL overnight *E. coli* cultures in Luria Bertani (LB) medium. Working solutions of plasmid were generated following the

Table 2.4: Restriction Digest

DNA sample	x μL	x μL
10x NEBuffer 4	2 μL	4 μL
Restriction enzyme 1 (BioLabs)	1 μL	2 μL
Restriction enzyme 2 (BioLabs)	1 μL	2 μL
Sterile Water	16-x μL	32-x μL
	= 20 μL	= 40 μL

selection of bacteria (see Section 2.1.8.3).

2.1.5.2 Plasmid Isolation

Plasmid were isolated from overnight cultures (5-6 mL) using the NucleoSpin Plasmid (Machery-Nagel). The protocol given in the enclosed User Manual (http://www.mn-net.com) was followed. Protocol consists of the harvested cells being lysed, washed and loaded onto a column for further washes. The DNA was eluted in 50 μL of water.

2.1.6 Restriction

2.1.6.1 Restriction digest

Restriction digests were set up in 1.5 mL Eppendorf tubes. The two recipes used for restriction digest are shown in Table 2.4. The 20 μL reaction volume (with 1 μL of DNA) was used for analytical purposes (confirmation of DNA sequence). The 40 μL reaction volume was used in preparation for ligation reactions.

Reactions were incubated at 37°C for 3 hours. Subsequently, the restriction enzymes were inactivated at 65°C for 10 minutes. However, in the case of a digestion for a ligation reaction the plasmid was subjected to dephosphorylation before heat inactivation (see Section 2.1.6.2).

$$m_i = \frac{m_v}{M_v} * 3 * M_i$$

Figure 2.1: The formula used to calculate the volume of the reactants used in the ligation reaction.

2.1.6.2 Dephosphorylation

Dephosphorylation of DNA samples was carried out as part of the restriction digest (see Section 2.1.6.1). Following a 3 hour incubation, 2 μL of CIAP Phosphatase (Thermo Scientific) was added to a 40 μL reaction mix. The mix was then incubated at 37°C for 20 minutes and heat inactivated at 65°C for 10 minutes.

2.1.6.3 DNA Extraction from Restriction Digest

The GenElute TM PCR Clean Up Kit from Sigma-Aldrich was used for DNA extraction from restriction reactions. The protocol includes the reaction mixes being loaded onto a pre-treated binding column, washed and the DNA sample being eluted in 25 μL of water.

2.1.7 Ligation

A 20 μL ligation reaction mix normally contained 40ng of vector DNA and have a molar ratio (MR) of 1:3 between vector (v) and insert (i). The mass (m) of the insert was calculated using Formula 2.1. The calculations assume a linear correlation between the length of the DNA and its molecular weight (M).

The ligation mixes were incubated in Eppendorf tubes at room temperature for 2 hours or at 4°C overnight.

2.1.8 Transformation and Selection of Bacteria

2.1.8.1 *E. coli* transformation

An 80 μL aliquot of competent *E. coli* cells was added to a 20 μL ligated plasmid. The solution was mixed by inverting the Eppendorf tubes and then put on ice. After 20 minutes on ice a 42°C heat shock was given for 90 seconds and the mix put back on ice for 20 minutes. 1 mL LB medium was added and the cells were left to recover at 37°C under constant shaking (200 rpm) for 1 hour. Aliquots of 150 μL of transformed *E. coli* cells were transferred onto LB plates containing the appropriate antibiotic (kanamycin 50 μg/mL, hygromycin 50 μg/mL, or riffamycin 50 μg/mL)

2.1.8.2 *Agrobacterium tumefaciens* transformation

For *Agrobacterium tumefaciens* transformation 1.5 μL of the plasmid of interest was added to 40 μL of competent *Agrobacterium tumefaciens* C58 cells. The solution was mixed by gentle pipetting. The mix was then transferred into a cool electroporation cuvette, which was inserted into a MicroPulser set to an apparatus specific setting for *A. tumefaciens*. An electric pulse was applied to the solution and the time of the pulse recorded. 1 mL LB medium was added to the cuvette and the cells allowed to recover at 28°C for 3 hours. Aliquots of 150 μL of transformed *A. tumefaciens* cells were transferred onto LB plates containing the appropriate antibiotic (kanamycin 50 μg/mL, hygromycin 50 μg/mL, or riffamycin 50 μg/mL)

2.1.8.3 Selection of Bacteria

Plates containing the transformed *E. coli* or *A. tumefaciens* (see Sections 2.1.8.1 and 2.1.8.2) were incubated at 37°C overnight or at 28°C for 2 days, respectively. The colonies were used for colony PCR (see Section 2.1.2.1). They were also used to inoculate 5-6 mL overnight cultures of liquid LB medium containing the appropriate antibiotic (50mL falcon tube).

Overnight cultures were used to isolate the plasmid (see Section 2.1.5.2).

2.2 Plant Methods

2.2.1 Growing *Arabidopsis thaliana*

2.2.1.1 Lines Described in this Thesis

See Supplemental Table S1 and Zourelidou *et al.* (2009) for lines described in this thesis.

2.2.1.2 Preparation of *Arabidopsis thaliana* Growth Media

Agar-plates typically consisted of 1/2 MS medium (0.215% Murashige and Skoog Basal Salt, 1% Agar). The pH-value of the solution was adjusted to 6.2 by titrating 0.1 M potassium hydroxide. If required, the appropriate antibiotic was added after autoclaving (kanamycin 50 μg/mL, hygromycin 50 μg/mL, riffamycin 50 μg/mL). Roughly 70 mL of the medium was poured into Petri dishes, which were stored at 4°C.

2.2.1.3 Sowing of *Arabidopsis thaliana* Seeds

Arabidopsis thaliana seeds were aliquoted into 1.5 mL Eppendorf tubes and 0.5 mL of bleach solution (0.5 mL commercial bleach in 9.5 mL water) was added to each tube and mixed. After 6 minutes, 1 mL of 0.1% Trition solution (10 μL Triton in 100 mL water) was added to the tubes. The supernatant was removed and 1.5 mL of 0.1% Trition solution added. This step was repeated twice.

Two methods were used for sowing the *Arabidopsis thaliana* seeds onto agar-plates (see Section 2.2.1.4). In the first protocol, the seeds were flushed out of the Eppendorf tubes onto filter papers with 70% ethanol, left to dry and scraped onto agar-plates. In the second protocol, the seeds were

suspended in sterile water and sown individually with a pipette (only used for gravitropic experiments, described in Section 2.3.1).

2.2.1.4 Growing *Arabidopsis thaliana* on Plates

The seeds sown on agar-plates were stratified at 4°C for at least 24 hours directly after sowing (see Section 2.2.1.3). The plates were then transferred to a 21°C growth room with either continuous lighting (seed bulking or crossing) or a 16/8h day/night cycle (gravitropic assays).

2.2.1.5 Growing *Arabidopsis thaliana* in the Greenhouse

All *Arabidopsis thaliana* plants grown for transformation (see Section 2.2.2), crossing (see Section 2.2.3) or seed bulking were transferred to moist Levington M3 compost containing imidacloprid. Four plants were grown in each 9 cm diameter plastic pot. A plastic sleeve was put around each pot. The greenhouse was kept at approximately 22°C with a 12 hour light photoperiod.

2.2.2 *Arabidopsis thaliana* Transformation

Agrobacterium tumefaciens C58 "plasmid stocks" (see Section 2.1.5.1) were used to inoculate 3 mL of LB medium containing the appropriate antibiotic(s) (kanamycin 50 μg/mL or rifamycin 50 μg/mL) ("starter culture"). The "starter cultures" were incubated at 28°C under constant shaking (200 rpm) for 24 hours. 2 mL of the starter cultures were used to inoculate 100 mL of LB medium containing the appropriate antibiotic(s). These cultures were kept in sterile 500 mL Erlenmeyer flasks, which were incubated at 28°C under constant shaking (200 rpm) until the OD600 reached 0.8 to 1.2. Then 5g of Sucrose and 50 μL of Silwet were added and mature *Arabidopsis thaliana* plants were dipped into the culture for around 12 seconds. Following dipping, sleeves were placed around the plants and sealed. After 4

Table 2.5: GUS Solutions

GUS staining solution	1 mM X-gluc (5-bromo-4-chloro-3-indolyl D-glucuronide), 2 mM K3Fe(CN)6, 2 mM K4Fe(CN)6
GUS clearing solution	5% formaldehyde, 5% acetic acid, 50% ethanol

days the sleeves were opened. The plants were kept as described in Section 2.2.1.5 until harvest.

2.2.3 *Arabidopsis thaliana* Crossing

A single inflorescence of the female parent was moved outside of the sleeve surrounding the pot and all siliques, open flowers, and the apical meristem removed. The sepals, petals, and anthers of the remaining floral buds were removed with fine forceps. Mature anthers were excised from the male parent plants and tapped onto the exposed carpel of the female parent plants. The crossed inflorescence was labelled and attached to the outside of the sleeve.

2.2.4 *Arabidopsis thaliana* Seed Harvesting

Arabidopsis thaliana seeds were harvested once all or most siliques were dehiscent. The aerial part of the plant was ground on paper and the debris removed.

2.2.5 Histochemical GUS Assays of Seedlings

Arabidopsis thaliana seedlings were incubated in GUS staining solution (Table 2.5) at 37°C until one line and/or treatment showed blue staining. Seedlings were then submerged in GUS clearing solution (Table 2.5) for 30 minutes to 1 hour. Finally they were repeatedly washed in 70% ethanol.

2.3 Gravitropic Assays

2.3.1 Gravitropic Assay

Plants were grown in a growth room as described in Section 2.2.1.4. Five day old seedlings were placed on the CPIB AutoCAM robotic platform and pictures automatically taken every 15 or 20 minutes by a Canon Photoshot A620 digital camera. After a minimum of 15 minutes post-transfer (to the robotic platform) the plates were turned 90°. The robotic platform was near-infrared backlit and the growth room had a 16/8h day/night cycle.

2.3.2 Cold-Conditional Gravitropic Assay

The experimental set-up was as described in Section 2.3.1, with the addition of a pretreatment. A cold or a control pretreatment was applied for six hours prior to gravi-stimulation. Plates were either kept in the cold room (cold pretreatment, −4°C) or on the lab bench (control pretreatment, room temperature) under illumination by mobile growth lamps.

2.4 Bioinformatic Methods

2.4.1 Analysis and Programming Software

All data processing and statistical analysis was conducted in the Jupyter Notebook environment as part of the Anaconda distribution. The programming languages Python and R were used for data processing and statistical analysis, respectively. Complex data processing functions were written as Python modules and imported into Jupyter Notebooks. Python modules were written in a functional programming style and require multiple open source packages (see Sections 2.4.2 and 2.4.3). Testing was conducted in the Python Tools for Visual Studio 2015 environment.

Jupyter Notebooks were either run on a remote Linux server or locally

on Windows. Only PRBPviz notebooks were run locally as they have a short runtime and all of the required packages run on Windows (see Section 2.4.4).

2.4.2 KSselect Module and Notebooks

The Kinase Subsequence SELECTion (KSselect) module offers multiple functions for reading Phytozome queries (Goodstein *et al.*, 2012) and manipulating multiple sequence alignments (MSAs). KSselect was run in Jupyter notebooks, which serve as documentation of the parameters used.

The KSselect workflow is as follows. Firstly, a Phytozome query is read into a Biophyton MSA object (Cock *et al.*, 2009). Secondly, a MSA object is modified using the 'cut_out_aligned_to_domain' function. I supply this function with a kinase domain sequence (e.g. PKA-alpha), which is used to generate a MSA of the kinase domain. Thirdly, using the function 'take_columns_biopy_alignment' amino acids from specific positions are extracted out of the kinase domain MSA. This function is provided with the positions of the Domains of Specificity (DoS) to generate a DoS MSA (see Supplemental Code S2.1.2 and S2.2.1).

One challenge when creating DoS MSAs is that DoS are defined with reference to an MSA of human kinases. To address this, the KSselect module uses two functions to convert the indices of the human kinase MSA to the indices of the MSA of interest. The first function 'get_closest_relative_to_sequence_from_alignment' takes a sequence from the MSA of interest and finds the most similar sequence in the human kinase MSA (reference sequence). The kinases of interest are then realigned with the reference sequence. The second function 'convert_indices_alignment_to_alignment' converts the indices of the human kinase MSA (DoS indices) into the indices of the MSA of interest. This function makes use of the reference sequence that is present in both MSAs (see Supplemental Code S2.1.2 and S2.2.1).

The three types of MSA were written to FASTA files, which were used as input for the generation of phylogenetic trees and dendrograms using RaxML (Stamatakis, 2014). One challenge faced in the creation of DoS dendrograms is that there is no precedence as to how an outgroup might be chosen. The lack of an outgroup means DoS dendrograms have to be displayed as unrooted trees, which limits the ways they can be interpreted. Unrooted phylogentic trees do not make assumptions about common ancestry and therefore can only reveal relatedness of leaf nodes and not their evolutionary history. To be able to compare DoS dendrograms to the full sequence and kinase domain phylogenetic trees, the later needed to be presented as unrooted trees as well. Further it remains unclear which method is preferable for generating DoS dendrograms; Neighbor-Joining, Maximum Parsimony, Maximum Likelihood or Markov Chain Monte Carlo. Criexell *et al.* (2015) use Neighbor-Joining to generate DoS dendrograms, however this choice might simply reflect the large dataset the authors worked with. I chose RaxML for its ability to use both maximum parsimony and maximum likelihood. The generation of trees can be time consuming and therefore RaxML was run on a Linux server. In this thesis only the maximum parsimony method was used. Future work is needed to assess which tree inference method is optimal for DoS dendrograms. The tree inference step of KSselect still has be improved and the current limitation have to be considered when interpreting the KSselect output.

In this thesis the KSselect functions are only used to analyse kinase domains and DoSs. However, the KSselect functions can easily be adapted to extract other domains or single amino acids from a MSA.

2.4.3 qGAP Module and Notebooks

The Quantitative Gravitropic Analysis Pipeline (qGAP) consists of a Python module, Jupyter notebooks, an imaging robot and the RootTrace software.

Refer to Chapter 6 for the workflow of the qGAP module and notebooks. For instructions on how to use RootTrace for gravitropic assays see Müller *et al.* (2018).

The qGAP module offers two main functions. The first function 'read_roottrace_output uses a CSV file to combine the output files from the RootTrace software, which are located in different sub-folders. This function returns one or two data-frames, depending on the parameters set in the RootTrace software. One data-frame is returned if RootTrace only measured root growth. Two data-frames are returned if RootTrace measures both root growth and angle. To ensure this function is composable, the output is wrapped in a maybe monad (functional programming structure, see Supplemental Code S2.2.2 and S2.1.1). The second function 'filter_dfs' takes both data frames (growth and angle) and processes them in parallel. This function is composed of smaller functions, each filtering and adapting the values of the relevant data frame(s). The parameters for these smaller functions can be modified via the parameters of the 'filter_dfs' function, which are as follows (see also Supplemental Code S2.2.2 and S2.1.1):

- **mb_gr_df:** Combined data frame of root tip position data wrapped in maybe.
- **mb_avv_df:** Combined data frame of root tip angle data wrapped in maybe.
- **time_duration:** Last time point to include in data frame.
- **gr_min_tracel_thresh:** Lower boundary of allowed distance between two subsequent root tip position data points before changing all subsequent entries to NaN.
- **gr_max_tracel_thresh:** Upper boundary of allowed distance between two subsequent root tip position data points before stopping the conversion to NaN.
- **remove_nan_thresh:** Allowed number of missing values per root.

- **remove_overall_quantile:** Boolean as to whether to cut out roots that have data points outside the overall quantile.
- **upper_quantile:** Parameter for upper boundary of remove_overall_quantile.
- **lower_quantile:** Parameter for lower boundary of remove_overall_quantile.
- **remove_wildtype_quantile:** Boolean as to whether to cut out roots that have data points outside the quantile of data point of wt_id.
- **wt_upper_quantile:** Parameter for upper boundary of remove_wildtype_quantile.
- **wt_lower_quantile:** Parameter for lower boundary of remove_wildtype_quantile.
- **wt_id:** String of wild-type id.
- **initial_angle_tuple:** Tuple of upper and lower threshold of allowed angles at time point 0.
- **zero_growthrate:** Boolean as to whether to subtract all root tip position data by the initial value (resulting in growth rate).
- **adjust_angle:** Boolean as to whether to add 90 to root tip angle data (resulting in angles corresponding to experimental set-up).

To provide a proof of concept (positive control) for qGAP I analysed the *pgm1* mutant, which exhibits a severe gravitropic defect (image data kindly provided by Nicholas Redman). The *pgm1* dataset shows little variation in growth rate (see Supplemental Fig. S3), meaning that the qGAP filter functions for outlier removal and sub-setting are not needed to increase the sensitivity of qGAP (MDD, data not shown). The filter functions are not needed for qGAP to detect the *pgm1* defect (see Supplemental Tab. S4 and Supplemantal Fig. S4). This *pgm1* analysis demonstrates the value of the MDD as a measurement of sensitivity.

2.4.4 PRBPviz Notebooks

The KSselect algorithm predicts differences in target specificity based on amino acids proximal and distal to the substrate binding site (domains of specificity / DoS). The Phospho-site Residue "Binding Pockets" visualisation (PRBPviz) notebook complements KSselect by allowing me to make predictions about the phospho-site preference of kinases. PRBPviz is based on PREDIKIN, an algorithm by Brinkworth *et al.* (2003). Both PREDIKIN and PRBPviz ignore distal DoS, which are accounted for in KSselect. PREDIKIN predicts the optimum phospho-site for a given kinase domain (see Section 4). In contrast, PRBPviz predicts likely target phosph-sites by comparing the chemical properties of proximal DoS and possible target phospho-sites.

The PRBPviz notebook consists of two parts (see Supplemental Code S2.1.3). In the first part, the PRBPs of AGC-VIII kinases are analysed. These PRBPs are extracted out of a kinase domain multiple sequence alignment (MSA) to generate a new MSA. When this new MSA is created, KSselect functions (see Section 2.4.2) are used to map the PRBP positions (Brinkworth *et al.*, 2003) to the corresponding MSA positions. These mapped positions are then added to the Supplemental Table found in Brinkworth *et al.* (2003). A subsequent data frame consisting of all amino acids from one PRBP, their PKA positions and their chemical properties is then made available to R. This data frame is utilised when creating balloon plots showing the molecular weight (size) and charge (color) of amino acids. The R package 'ggpubr' was used to generate these balloon plots.

In the second part, PRBPviz is used to analyse PIN protein phospho-sites. Phospho-sites on PIN proteins are collated from a MSA of PIN proteins, using the known PIN1 sites as a reference (Barbosa *et al.*, 2018). Sites that do not contain a serine at P-0 are excluded. Balloon plots (one for each PIN) were created as previously noted.

2.4.5 ETE3 Data Visualisation - KSviz, Auxin Induction and Abiotic Stress

The ETE3 tree drawing engine is part of the ETE3 tool kit (Huerta-Cepas *et al.*, 2016) that automates the manipulation, analysis, and visualisation of phylogenetic trees. Throughout this thesis I utilised the highly flexible ETE3 tree drawing engine in two ways.

Firstly, ETE3 allows for the visualisation of trees created by the KSselect algorithm. The main benefit of ETE3 with regards to the KSselet data is that it can display whole protein topology or sequence details alongside the trees. This additional information allows readers to see the differences in full proteins and domains of specificity. I will subsequently refer to the notebooks visualising the KSselect output as KSviz notebooks (see Supplemental Code S2.1.4).

Secondly, ETE3 allows for the display of expression data in conjunction with any phylogenetic tree. I utilised this feature twice throughout this thesis. Firstly, I generated heat maps for auxin expression of kinase families (AGC-VIII and CIPK). Secondly, I created heat maps for CIPK kinase expression following various stress stimuli (see Supplemental Code S2.1.4).

Chapter 3

Phylogenetic Analysis to Infer Kinase Function - A Novel Screening Approach

3.1 Introduction

Phylogenetic mapping of the conserved protein kinase catalytic domains can serve as a useful first step in the functional characterization of these newly identified family members.

Hanks et al. 1988

Thirty years ago the pioneers in the classification of kinases set the standard that would dominate the field for the coming decades. They realised that the kinase domain phylogeny more accurately reflects the functional relationship between kinases than phylogeny based on the full protein sequence. In 2015, Creixell *et al.* improved the functional mapping of kinases by generating dendrograms based on domains of specificity (DoS) (domains here refers to single amino acids and not protein domains). DoS are a group of residues within the kinase domain that are thought to facilitate the target-specificity of kinases (see Section 1.2.1.1). Here we generate DoS

dendrograms for plant kinases for the first time.

DoS dendrograms as used by Creixell *et al.* (2015) are the most accurate in silico method for comparing target-specificity within groups of kinases (see Section 1.2.1.1). I decided to use DoS dendrograms to assess redundancy between plant kinases on a biochemical (i.e. target-specificity) level within a species. This approach therefore allows me to more accurately predict the function and evolutionary history of individual plant kinases. In addition, this approach provides an invaluable tool for predicting kinase orthologs across species.

The first step taken in generating DoS dendrograms for plant kinases was constructing an algorithm that enabled me to map the DoS in plant sequences. To map DoS in sequences first I had to isolate the kinase domain from the full-protein sequence (see Figure 3.1). The algorithm therefore represents a sequential selection of subsequences. For this reason I named the algorithm Kinase Subsequence SELECTION (KSselect).

In addition to generating DoS dendrograms, KSselect also generates the corresponding full sequence and kinase domain phylogenies (see Figure 3.1). The full sequence phylogenies show the evolutionary relationship of the whole protein, however they are highly influenced by domains outside of the kinase domain. Because domains outside of the kinase domain primarily function in protein regulation and targeting, full sequence phylogenies reflect similarities in these two processes (see Section 1.2). The traditional approach to show the functional relationship of kinases is to analyse the kinase domain phylogeny. However, I consider DoS dendrograms as a more accurate representation of the functional relationship between kinases. The kinase domain phylogenies are included only to show the evolutionary relationship between the kinase domains.

Creixell *et al.* (2015) set no precedent for the selection of outgroups for DoS dendrograms. Generating a protocol for outgroup selection is not



Figure 3.1: Schematic of the KSselect algorithm. KSselect uses a multiple sequence alignment (or a Phytozome (csv) output) of full protein sequences as input. From this it generates multiple sequence alignments of the kinase domain and the domains of specificity. The phylogenetic trees generated from these alignments can be used for an in-depth analysis of the functional and evolutionary relationship of the kinases of interest.

within the scope of this thesis. I was therefore restricted to visualising the DoS dendrograms as unrooted trees. For the sake of consistency, both the full sequence phylogenies and the kinase domain phylogenies are likewise displayed as unrooted trees. The consequence of using unrooted trees is that no conclusions can be drawn regarding the evolutionary history of the sequences. My analysis is therefore restricted to investigating the relationship between the different sequences (see Section 2.4.2).

The three different outputs of the KSselect algorithm will allow me to gain insights by comparing how kinases cluster in each tree. Firstly, differences in the full sequence phylogeny compared to the kinase domain phylogeny could suggest that domains outside of the kinase domain have undergone duplication, divergence and/or recombination. This would indicate that regulation and/or targeting of these kinases may not have evolved in parallel to the kinase domain. Secondly, differences in the DoS dendrogram compared to the kinase domain phylogeny could suggest that the kinase domain has been under selective pressure to change its target specificity. This putative independent evolution of the DoS domains from the kinase domains might be the result of selective pressure to avoid cross-talk between

pathways.

The juxtaposition of the full sequence phylogeny, the kinase domain phylogeny and the DoS dendrogram gives us a good estimation of which kinases function redundantly. If a group of kinases clusters in all three trees, these kinases possibly function redundantly. A change in only the full sequence phylogeny could suggest that kinases in the cluster have diverged in their regulation and/or targeting mechanisms. A change in only the DoS dendrogram could suggest that kinases in the cluster have diverged in their target specificity.

When analysing the full sequence phylogenies generated by KSselect, it is vital to take into account that these phylogenies represent an average of all changes in the protein sequence, including linker regions. Different mutation rates in different domains plus variable linker region lengths will skew the phylogenetic tree. Therefore, divergence of a kinase in the full sequence phylogeny does not necessarily suggest a change in protein regulation and/or targeting. If a kinase does diverge on the full sequence phylogeny, a specific script (KSviz) is available to further analyse how the protein sequence has diverged (see Section 2.4.5). KSviz generates a schematic sequence alignment which highlights overlapping domains outside the kinase domain. This gives an estimation of where linker regions have changed in length (see e.g. Supplemental Fig. S9).

Another notable factor when using the KSselect algorithm is that DoS dendrograms are based on the BLOSUM-62 substitution matrics. This means that any changes in the charge of DoS residues will not be given enough weight when generating the dendrogram. To address this issue I can again use the KSviz script to visualise the sequence alignment of all DoS (see e.g. Supplemental Fig. S9). In a later chapter I will introduce an alternative script to KSviz that can be used to visualise the charge of certain DoS residues.

In this chapter I demonstrate two applications of the KSselect algorithm. In both applications KSselect improves our understanding of the functional relationship between kinases. In the first application I select candidate kinases for gravitropic signalling based on their ARF7/ARF19-dependent auxin inducibility. I use ARF7/ARF19-dependent auxin inducibility as a selection criterion because both transcription factors are key in gravitropic signalling (Okushima *et al.*, 2005). I use KSselect to evaluate each candidate kinase by searching for functional homologs and evaluating their functional relationship to other kinases. As PINOID is known to play a crucial role in gravitropic signalling (see Section 1.2.2.4), I chose the entire AGC family of kinases to demonstrate this approach. In the second application I use the KSselect algorithm to find potential orthologs of a kinase of interest. The kinase of interest in my case is maize CIPK4, which has been associated with root angle. CIPK kinases are characterised by calcium-regulation, which is thought to play a key role in gravitropic signalling (see Section 1.1.2.5).

3.2 Auxin-Inducible AGC-Kinases

[...] Redundant isoforms are not coregulated in their expression patterns.

Kafri et al. 2009

Functional redundancy is often defined as two homologous genes that compensate for each others loss by having partially overlapping functions. In yeast, redundant kinases can be found in most signal transduction pathways. Karfi *et al.* (2006) propose a mechanisms by which functional overlap facilitates or contributes to signal transduction. Redundant genes can form responsive backup circuits, in which one gene, the controller, is induced and negatively regulates the other gene, the responder. In this section I will, for the first time, assess *Arabidopsis thaliana* AGC-VIII kinases for potential auxin responsive backup circuits. For now I will continue with the tradi-

tional assumption that the redundant kinases form kinase domain clusters, and potential controllers will be auxin inducible clade members. In later sections I will critically review whether the seen clades are truly redundant.

I used the timeline dataset for ARF7/19-dependent auxin induction in roots (E-MTAB-1354, processed by M. Wilson) to calculate ARF7/19-dependent and -independent auxin induction in both the meristem (see Supplemental Fig. S6 and S8) and the elongation zone (see Figure 3.2 and Supplemental Fig. S7). We compared the data by arranging the AGC-VIII kinase entries in accordance with the kinase domain phylogeny and by plotting the data points as a heatmap. To get a first estimate of functional redundancy, I look at this heatmap alongside the kinase domain phylogenetic tree (see Figure 3.2).

The auxin induction dataset consists of six data points: 0, 15, and 30 minutes and one, two, four, and eight hours post auxin treatment. The first timepoint marks the earliest point at which the mRNA of early auxin responsive genes is detectable. I used the first five timepoints to assess the auxin inducibility of gravitropic candidate genes (see Figure 3.2). The sixth timepoint was not relevant in the assessment because a gravitropic response to a 90 degree stimulus is completed within 6 hours. I used ARF7/19 dependent auxin inducibility as a selection criterion for gravitropic candidate kinases.

The highest level of induction is shown by PID (see Figure 3.2), a kinase playing a central role in gravitropism. The auxin induction of PID only plays a functional role in the later stages of auxin gradient re-configuration. However, PID is also thought to play a role in the early non-genomic stages (see Section 1.2.2.4). This sets the precedent for using ARF7/19-dependent auxin induction as a criterion for selecting candidate kinases involved in both the transcriptomic and non-transcriptomic stages of gravitropism. Within the clade in which PID is found, which also includes PID2,

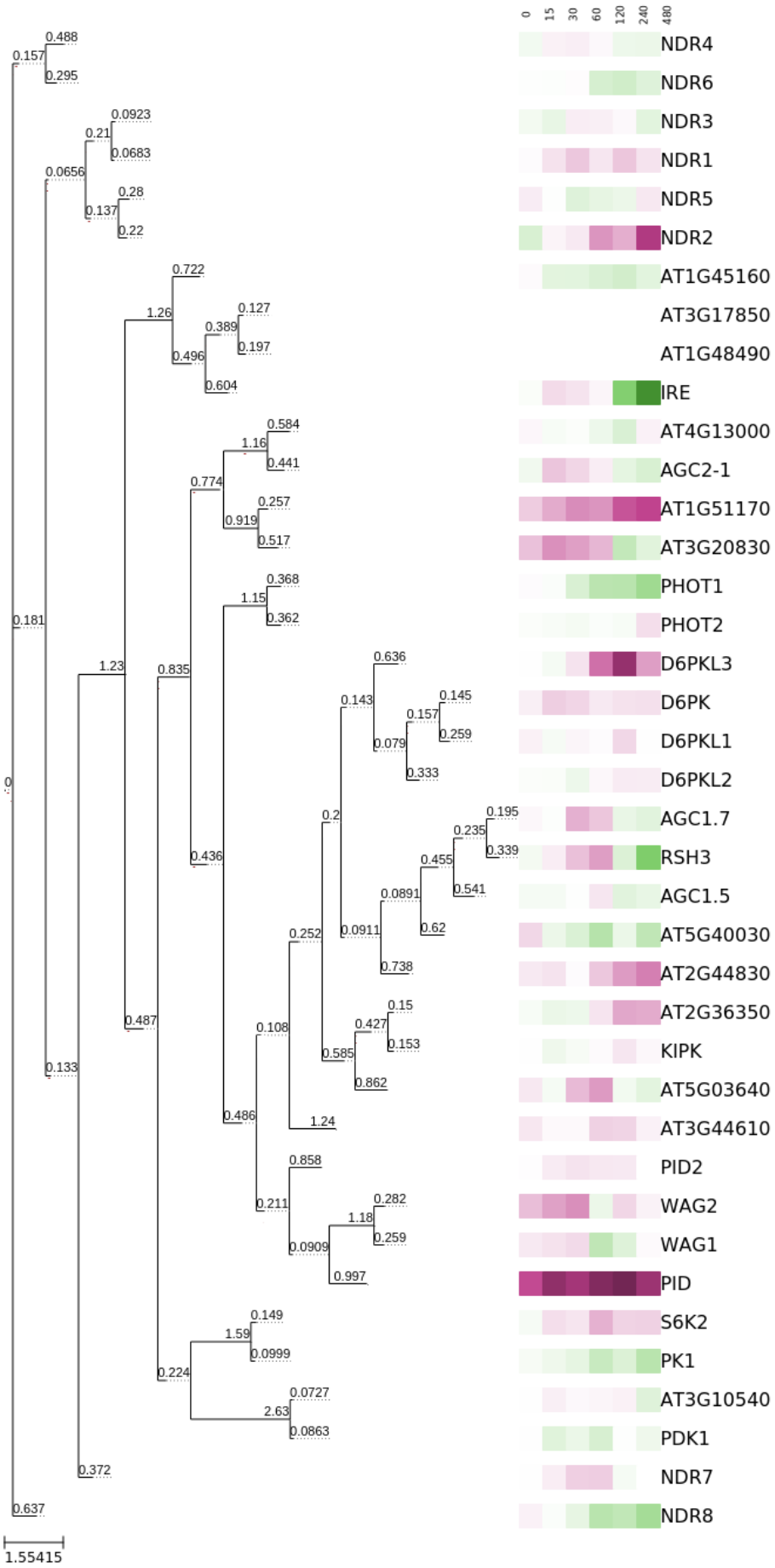


Figure 3.2: *(legend on next page)*

Figure 3.2: Unrooted phylogentic tree (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their ARF7/19-dependent auxin-regulated expression in the root elongation zone. Auxin expression heatmaps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3. Scale bar: Mean number of substitutions per site. (see Section 2.4.5)

WAG1 and WAG2, only PID shows high auxin inducibility(see Figure 3.2). Seeing auxin inducibility in only one member of a clade is a distinctive pattern that is repeated in other clades of both AGC and other kinases. PID is thought to be partially redundant with other members in its clade, suggesting it could function as a controller in an auxin responsive backup circuit. In the following chapter I will address this potential redundancy further (see Chapter 4). PID has long been thought to function in the apical and basal targeting of PIN auxin efflux carriers. Recently it has been shown that it functions as an activator of PIN proteins (see Section 1.2). Its role in root gravitropism could therefore simply be the acceleration of auxin transport on the lower side by activating PIN2. In this simple model auxin could up-regulate PID function on a transcriptional and protein level.

From the remainder of the AGC-VIII kinases in the dataset I selected D6PKL3 as input for our KSselect algorithm to evaluate redundancy relationships to other kinases. D6PKL3 showed the second highest level of auxin induction, after PID (see Figure 3.2). D6PKL3 can be found in the same clade as D6PK. Like PID, D6PK has been show to activate PIN auxin efflux carriers (Zourelidou *et al.*, 2014). In addition, both the PID and D6PK clades are thought to consist of redundant members. One could therefore hypothesise that D6PKL3 is functioning as a controller in an auxin responsive backup circuit. However, the following section shows that the D6PK clade is likely to be functioning non-redundantly.

3.3 Phylogeny of AGC-Kinases

The high degree of sequence conservation between these proteins strongly suggests that they have redundant function, and throughout this report we will refer to these four kinases collectively as D6PKs.

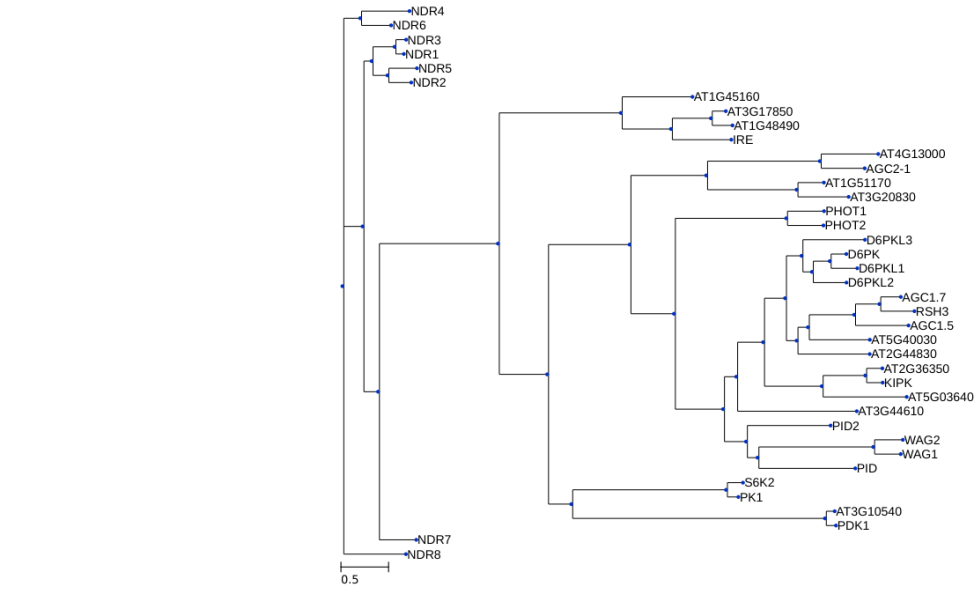
Zourelidou et al. 2009

When the clade of D6-Protein Kinases (D6PKs) was first described, D6PK was arbitrarily chosen as the founding member. Since the original paper, all four D6PK kinases have been treated as functionally redundant, based on their homology in the kinase domain. Here I present results that bring the assumed redundancy into question.

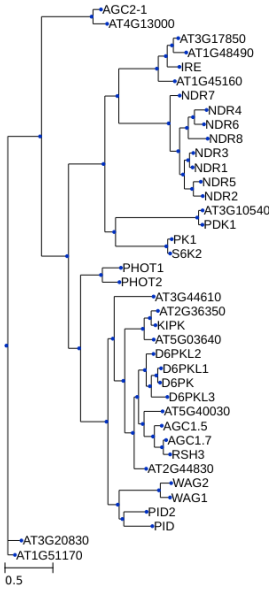
I used my KSselect algorithm to examine the AGC-VIII kinases in more detail (see Section 3.1). KSselect creates the full sequence phylogeny, the kinase domain phylogeny (activation loop insert excluded), and a DoS dendrogram. In the three trees, the D6PK clade (as determined in the kinase domain phylogeny) shows a different relationship (see Figure 3.3). In both the full sequence and kinase domain phylogeny the D6PKs cluster together, however in the DoS dendrogram this cluster is disrupted (see Figure 3.3).

The phylogenetic relationships of the D6PKs in the kinase domain phylogeny differ compared to the full sequence phylogeny. In the kinase domain phylogeny D6PKL2 kinase shows the closest similarity to the rest of the AGC family. In contrast, in the full sequence phylogeny D6PKL3 is located at the boarder of the clade (see Figure 3.3 and Supplemental Fig. S9).

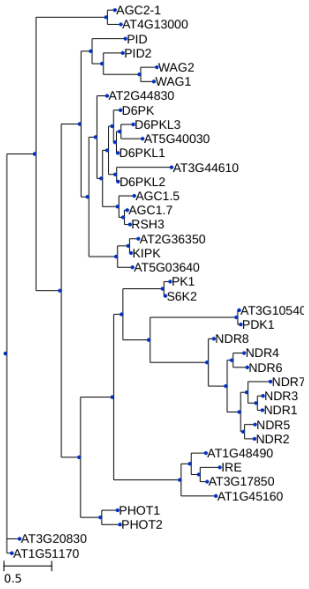
In contrast to the PIDs, the D6PK clade is disrupted in the DoS dendrogram. Specifically, D6PKL2 and D6PKL3 show a closer relationship to unidentified genes (see Figure 3.3 and Supplemental Fig. S9). This suggests that there was a selective pressure on the D6PK kinases to evolve different phospho-site preferences. This evolutionary process is thought to eliminate cross-talk between the kinases, which is contradictory to the assumed redundancy between these kinases.



(a) Full sequence

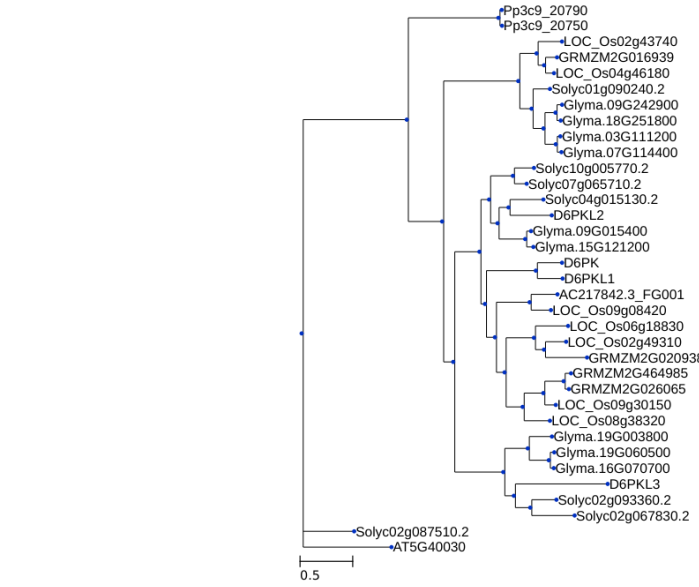


(b) Kinase domain

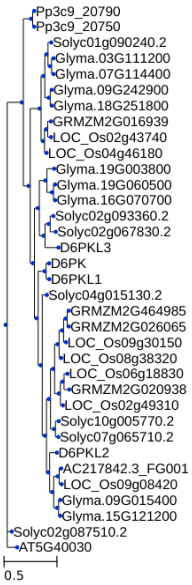


(c) Domains of specificity

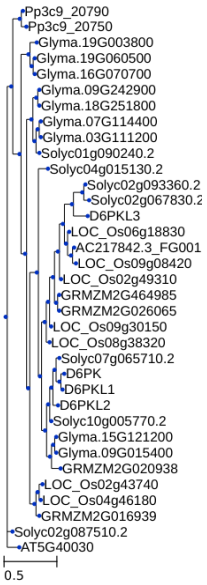
Figure 3.3: Unrooted *Arabidopsis thaliana* AGC-VIII kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site.



(a) Full sequence



(b) Kinase domain



(c) Domains of specificity

Figure 3.4: Unrooted *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site.

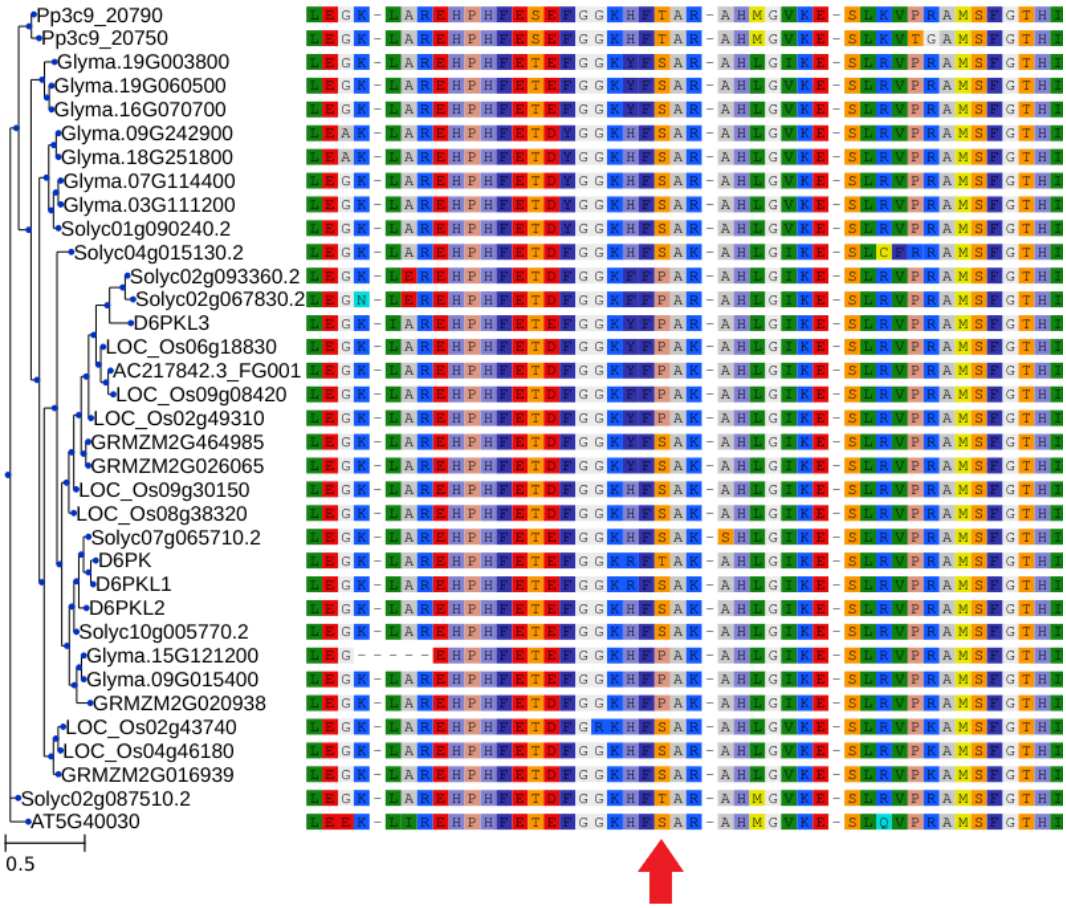


Figure 3.5: Unrooted DoS-dendrogram of *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinases (same as Figure 3.4c). Sequence detail shows a subset of domains of specificity (DoS). Red arrow indicates a conserved change (S/T to P) in one DoS of D6PKL3 and D6PKL3-related kinases. Generated using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site.

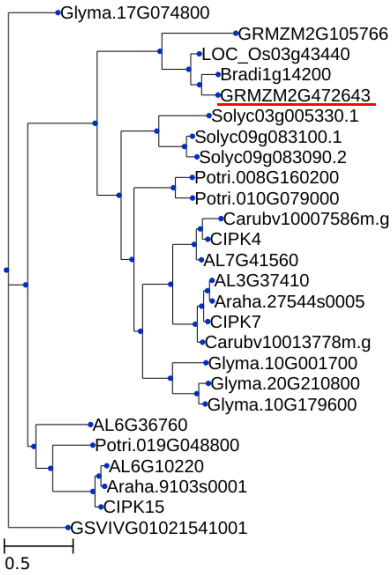
Next, I looked at the phylogeny of the D6PK clade across species (see Figure 3.4 and 3.5). From the kinase domain phylogeny it seems possible that D6PKL2 and D6PKL3 are the oldest members of the clade. D6PK and D6PKL1 could have potentially arisen from a more recent gene duplication event. This pattern is also clearly present in the full sequence tree, where D6PKL3 forms an even more distinct cluster with its potential analogs (see Figure 3.4 and Supplemental Fig. S10). Strikingly, D6PKL3 is positioned distal to the other D6PKs in the DoS dendrogram. In the DoS dendrogram D6PK, D6PKL1, and D6PKL2 cluster together, while D6PKL3 falls into a separate cluster with analogs primarily from monocot species (see Figure 3.4 and Supplemental Fig. S10). This pattern suggests that there might have been a strict selective pressure on D6PKL3 (the ancestral D6PK). This could have ensured the conservation of D6PKL3 target specificity, while the other D6PK kinases were able to diverge. While inspection of DoS sequences across species reveals a high conservation, there is a clear and conserved amino acid change in one of the DoS in D6PKL3 and multiple D6PKL3-related kinases (see Figure 3.5).

3.4 Phylogeny of Auxin-Inducible CIPK-Kinases

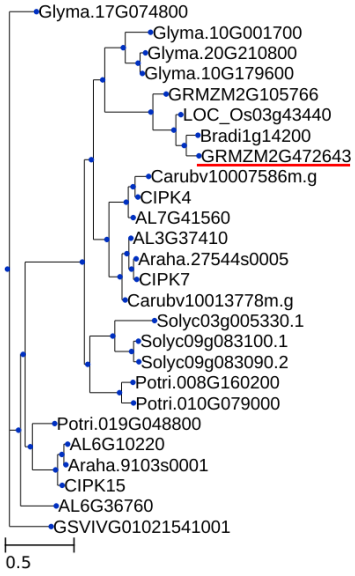
[...] There is a need to prioritize translational research as an important component of bench scientists' goals of research.

Mattoo et al. 2009

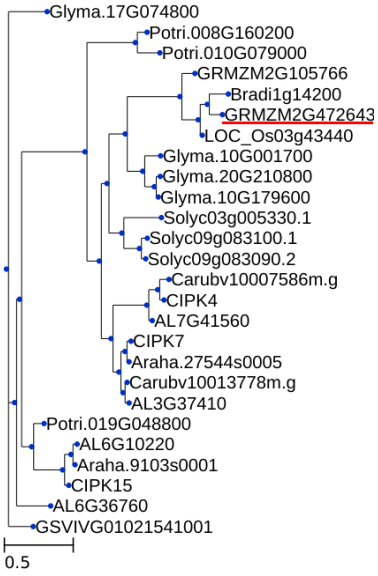
Translational research allows us to take research carried out on model species under laboratory conditions and apply it to crop species in the field. This is a potential area where my KSselect algorithm could be utilised to discover crop orthologs of a kinase of interest in a model species. Here I utilise KSselect to discover putative orthologous genes of a maize kinase in *Arabidopsis thaliana*, i.e. translational research in reverse.



(a) Full sequence



(b) Kinase domain



(c) Domains of specificity

Figure 3.6: Unrooted *S. moellendorffii, Z. mays, A. lyrata, A. halleri, A. thaliana, B. distachyon, C. rubella, G. max, V. vinifera, O. sativa, P. trichocarpa, P. patens* and *S. lycopersicum* CIPK4/7 kinase phylogenetic trees/dendrograms based on full sequence (a), kinase domain (b) and domains of specificity (c). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5). Scale bar: Mean number of substitutions per site.

The kinase of interest, *Zea mays* CBL-interacting protein kinases 15 (ZmCIPK15, GRMZM2G472643), was discovered in a genome-wide association study (GWAS). ZmCIPK15 was found to be associated with a change in crown and seminal root angle (J. Lynch, pers. comm.). All CIPKs are calcium-regulated and interact with calcineurin B-like (CBL) proteins (see Section 1.2.3.1). Calcium is thought to have a dual role in gravitropism. Firstly, it is thought to signal in graviperception downstream of amyloplast displacement, but its precise role remains elusive (see Section 1.1.2.5). Secondly, calcium is the primary signal facilitating the non-transcriptional auxin response (see Section 1.1.3.3.1). ZmCIPK15 could act downstream of either signals to facilitate guidance of lateral root angle. I used KSselect to find the putative functional ortholog of ZmCIPK15 in *Arabidopsis thaliana*.

The KSselect output shows a conserved pattern across all three trees (full sequence, kinase domain, and domains of specificity), with ZmCIPK15 (GRMZM2G472643) consistently located within a clade of monocot CIPKs (see Figure 3.6). Core eudicot kinases from various plant families: Malpighiales (*P. trichocarpa*), Brassicales (*A.thalina, A. halleri, C.rubella*), Fabidae (*G. max*), and Asterid (*S. lycopersicum*) are seperate to this monocot clade (see Figure 3.6 and Supplemental Fig. S13). This could indicate that the ancestral kinase of ZmCIPK15 diversified after the monocot-dicot split, which could suggest that there is no clear putative ortholog of ZmCIPK15 in dicot species.

CIPKs from the core eudicots form species-specific clusters (see Figure 3.6 and Supplemental Fig. S13). This could suggest that the kinases in each species function redundantly, because neither the regulatory domains (full sequence), nor the target specificity have substantially diverged. Sequences of the CIPKs show a high level of sequence similarity (see Figure 3.6 and Supplemental Fig. S13), suggesting that there is a high conservation of function across the whole subclade. Redundancy of CIPK4 and CIPK7 is

also reflected in the KSselect output for all *Arabidopsis thaliana* CIPKs, with the pair forming a separate cluster in all three trees (see Supplemental Fig. S11 and S12). However, in the DoS-dendrogram CIPK21 joins the CIPK4/CIPK7-cluster. This might suggest that there was a selective pressure for the target specificity of CIPK21 to resemble that of CIPK4 and CIPK7. This similarity in target specificity offers the possibility of cross talk between the CIPK21 and the CIPK4/CIPK7 pathway. CIPK21 is involved in the salt stress response. From the KSselect output it is clear that there is no CIPK4 ortholog in *A. halleri*, a salt-adapted species related to *Arabidopsis thaliana* (see Supplemental Fig. S11). It is tempting to speculate that *A. halleri* has lost CIPK4 in order to reduce cross talk with CIPK21, thereby improving its growth under salt stress conditions.

It is clear from the KSselect output that CIPK4 and CIPK7 are biochemically redundant kinases. As previously observed, in AGC kinases one gene from each redundant cluster stood out as being auxin inducible (see Section 3.2). This is a reoccurring pattern seen within the CIPKs, including the CIPK4/CIPK7 cluster (see Figure 3.7 and Supplemental Fig. S14 to S16). CIPK7 is the auxin inducible isoform, while CIPK4 is the non-inducible isoform (see Figure 3.7). The pair offer a simple model to investigate a possible auxin responsive backup circuit, with CIPK7 as the controller and CIPK4 as the responder. Interestingly, the pattern of CIPK7 auxin-induction resembles that of D6PKL3, with maximum expression at four hours (see Figure 3.7) (see Section 3.2). This suggests that the expression of both CIPK7 and D6PKL3 could be controlled by the same secondary auxin-response transcription factors, which may be specific to the gravitropic response.

3.5 Conclusion

[...] Selective forces [...] acted on the system as a whole during evolution.

Jin and Pawson, 2012

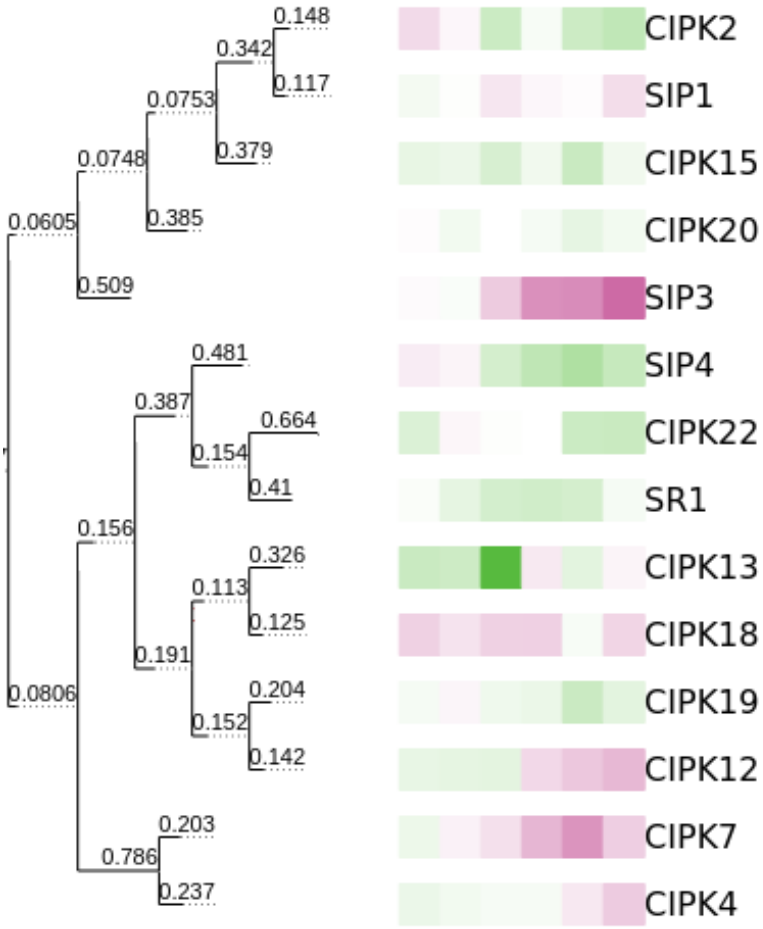


Figure 3.7: Unrooted phylogenetic tree (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their ARF7/19-dependent auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3 (see Section 2.4.5).

Using DoS dendrograms of the KSselect algorithm as a proxy for kinase target specificity, I was able to speculate as to whether selective pressures acted to conserve or diverge the target specificity of our kinases of interest.

In instances where the DoS dendrogram diverged from the kinase domain phylogeny, selection pressures may have resulted in either an accumulation or a prevention of mutations in the domains of specificity. In the case of D6PKL3 I propose that selection pressures have ensured that its target specificity remained similar to that of its ancestral kinase. The target speci-

ficities of the other D6PK members are thought to have diverged. In the case of CIPK21, it is likely that selection pressures have acted on CIPK21 target specificity to resemble that of CIPK4 and CIPK7.

In contrast, if kinase positioning in the DoS dendrogram and the kinase domain phylogeny is similar, I took this as a hallmark of biochemical redundancy in target specificity. Biochemically redundant kinases could be functionally non-redundant due to differences in regulatory and targeting domains outside of the kinase domain.

Because redundant target-specificity does not necessarily imply functional redundancy, the KSselect algorithm includes the full sequence phylogeny. This was taken as a proxy for kinase regulation and targeting. Differences between the full sequence phylogeny and the kinase domain phylogeny need to be interpreted with caution. This is because regions linking the protein domains (linkers) are of variable lengths, which skews sequence relationships seen in the full-sequence phylogeny. Nonetheless, similarities between full sequence phylogeny and the kinase domain phylogeny still suggest conservation within regulatory and targeting mechanisms. Future versions of KSselect will include separate phylogenies for every domain present in a full kinase sequence.

Throughout this chapter I have used the KSselect algorithm to predict kinase redundancy. This combined with the phylogeny-centric visualisation of auxin induction led us to hypothesise an auxin responsive backup circuit for the redundant CIPK4 and CIPK7. However, for D6PKL3 we predict a difference in target specificity compared to the other D6PKs. D6PKL3 is therefore likely non-redundant in regards to its target phosphorylation sites (phospho-sites). However, D6PKL3 can still be functionally redundant with other D6PKs if the target protein contains phospho-sites for both.

In this chapter I have focused on the evolution of kinase regulation, targeting and target specificity. However, in the following chapters I will

add another level of complexity by looking at the phospho-sites that have co-evolved with these kinases. The main focus of subsequent chapters will be the quantitative phenotyping of mutant lines for root gravitropic defects. Because I hypothesise auxin responsive backup circuits between various kinases, single knock-out mutants are likely to exhibit only slight shifts in gravitropic kinetics. It is therefore pivotal to quantitatively measure the gravitropic response of all mutant lines.

Chapter 4

Characterisation and Phenotyping of D6PKL3

4.1 Introduction

[Protein kinase C (PKC) isoforms] are coexpressed and respond to similar stimulatory agonists in many cell types. However, there is growing evidence that individual PKC isoforms subserve unique (and in some cases opposing) functions in cells [...].

S. Steinberg (2008)

The auxin transport field is going through a paradigm shift. It has long been thought that PIN protein trafficking is the primary mechanism regulating the capacity and direction of polar auxin transport during plant growth and development. However, recent findings suggest that D6-PROTEIN KINASEs (D6PKs) and PINOID (PID) kinases directly activate PIN proteins. In an oocyte-based auxin transport assay, the expression of PIN3 or PIN1 did not result in auxin transport. However, when co-expressed with D6PK a significant increase in auxin transport was detectable. Furthermore, PIN1 was shown to be activated by PINOID (PID) and WAVY ROOT GROWTH 2 (WAG2) (Zourelidou *et al.*, 2014). D6PK, PID, and WAG2 belong to the

AGC-VIII kinase family. D6PK is the founding member of a protein group encompassing D6PK, D6PK-LIKE1 (D6PKL1), D6PKL2, and D6PKL3. PID and WAG2 belong to a different group that also contains WAG1 and PID-LIKE1.

The D6PKs have been considered redundant kinases since they were first characterised (Zourelidou *et al.*, 2009). The authors used the term redundancy to describe three observations. Firstly, Zourelidou *et al.* (2009) found high sequence conservation in the kinase domains of all D6PKs. From this they assumed that all D6PKs have the the same biochemical function, meaning they have the same phospho-site preference and target the same proteins. Secondly, they show that the expression domains of D6PKs overlap, meaning at least two D6PKs are present in the majority of cells (see Figure 4.1). This justifies the assumption of biochemical redundancy in a multicellular context. Thirdly, they show that at organism level the phenotypic defects of single knock-out mutants are minute, with defects becoming more severe with an increasing level of mutant complexity (double, triple, quadruple). This is indicative of a different type of redundancy, which we will refer to as phenotypic redundancy. It is clear that the authors use redundancy as an umbrella term for two different processes, biochemical redundancy and phenotypic redundancy. Based on the above observations of redundancy the authors suggest that the biochemical function, which was determined for the founding member (D6PK), is conserved across the entire D6PK group.

Zourelidou *et al.* (2014) present evidence suggesting that D6PK preferentially phosphorylates the S4 and S5 phospho-sites on PIN proteins (see Section 1.1). This is in contrast to PID, which is suggested to preferentially phosphorylate the S1, S2, and S3 phospho-sites. The phospho-site preference of PID is thought to be conserved throughout the PID clade. Based on published data, the minimal model for D6PK and PID signalling

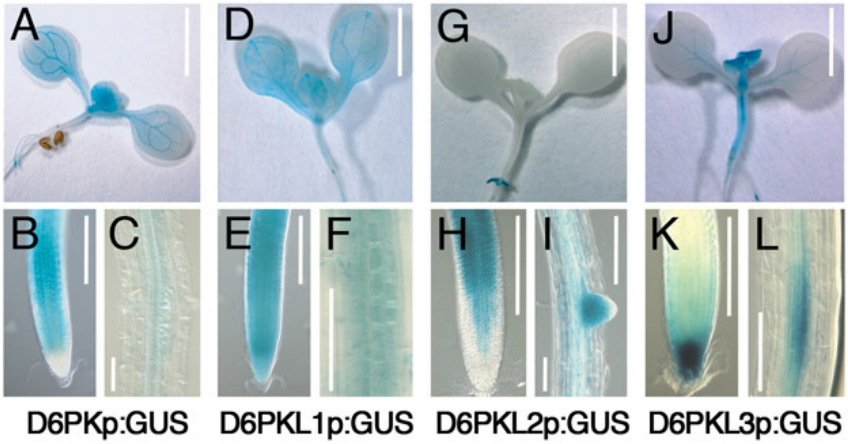


Figure 4.1: Promoter-GUS expression analyses of the D6PK-family. (A,D,G,J) GUS staining of 7-day-old seedlings. (B,E,H,K) GUS staining of root tips. (C,F,I,L) GUS staining of lateral root initiation sites. (K) Only D6PKL3 accumulates in columella cells of the root tip . (E) D6PKL1 is expressed ubiquitously throughout the root tip, including the columella cells. (B, H) D6PK and D6PKL2 show no expression in columella cells. From Zourelidou *et al.* (2009)

is a simple transducer model in which each group of kinases redundantly phosphorylates a specific subset of PIN phospho-sites (see Figure 4.2).

In the first chapter, I classified D6PKL3 as a candidate gene for gravitropic signalling. This was based on two characteristics. Firstly, D6PKL3 is one of few auxin inducible AGC-VIII kinases. Secondly, the close relationship of D6PKL3 to D6PK suggests that it also plays a role in auxin transport regulation. However, results from the KSselect algorithm suggest that the target specificity of D6PK has diverged from D6PKL3. This implies that the role of D6PK in auxin transport regulation is not conserved in D6PKL3.

In this chapter I will address the role of D6PKL3 in root gravitropism. Firstly, I employ a bioinformatic approach to make predictions about the target specificity of D6PKL3. This enables me to assess whether D6PKL3 phosphorylates PIN proteins. This was a crucial line of enquiry following the KSselect results, which suggest that D6PKL3 does not phosphorylate PIN proteins. Discovering whether or not D6PKL3 is able to phosphory-

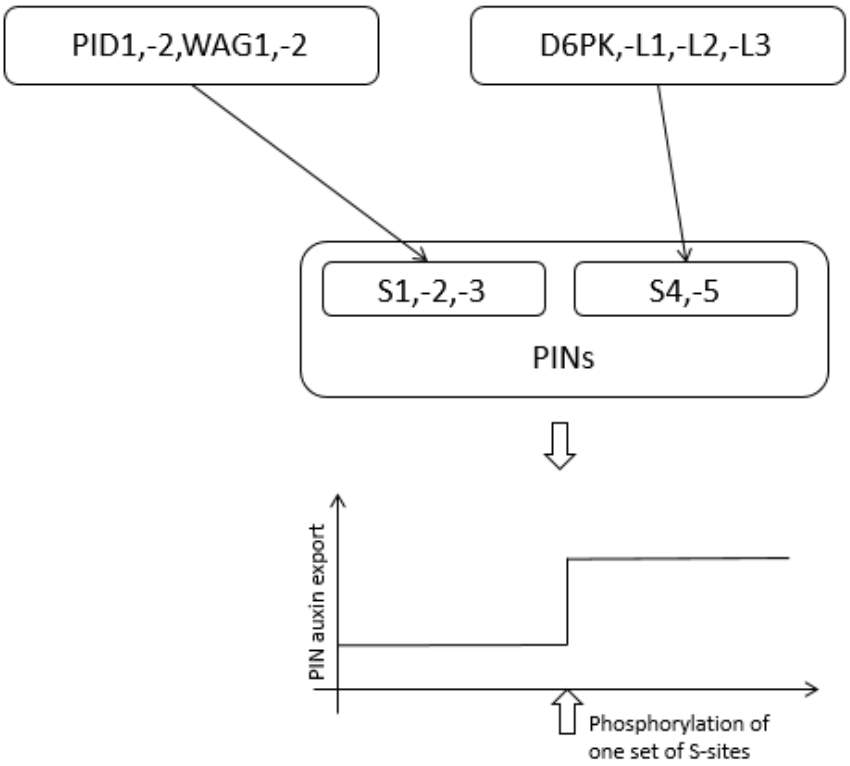


Figure 4.2: Schematic of PID and D6PK group functions based on published results, following the linear model of kinase signal transduction. Graph (bottom) indicates the suggested all-or-nothing PIN-activation by these kinases.

late PINs will guide my research on its role in gravitropism. Secondly, I use molecular biological methods to explore the function of D6PKL3 on a tissue and cellular level. Primarily I show where D6PKL3 auxin induction is localised in the root. Furthermore I show D6PKL3 localisation in different cell types of its expression domain. Both insights help us to generate a model for D6PKL3 function in the root. Thirdly, I conduct an in-depth gravitropic phenotyping of a *d6pkl3* knock-out mutant. Based on our insights on D6PKL3 function, we considered it necessary that our gravitropic phenotyping approach was both highly sensitive and flexible. For this reason I created a new phenotyping pipeline that extended the tool-kit available at our laboratory.

4.2 Prediction of D6PK Phospho-Site Preferences

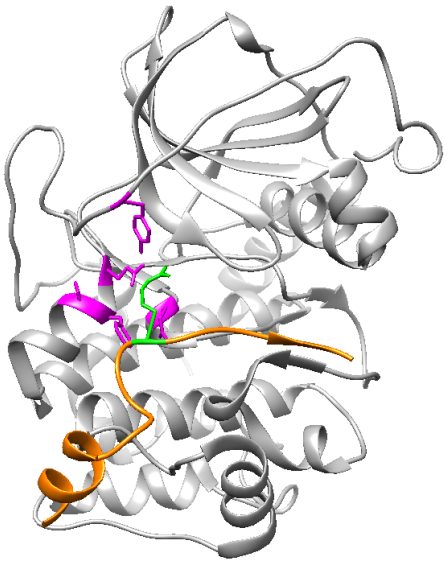
Whereas D6PK appeared to have a preference for the S4 and S5 sites in PIN1 and PIN3, PID preferentially phosphorylated the previously identified S1 to S3 phosphosites.

Zourelidou et al. 2014

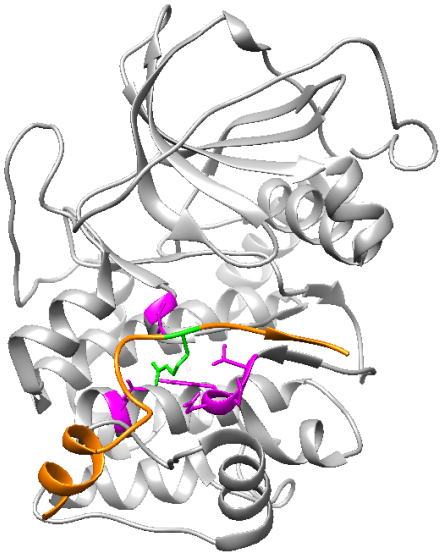
Past publications suggest that the function of both the D6PK and PID family of kinases resemble a transducer model. D6PKs and PIDs are thought to phosphorylate different sets of phospho-sites on PIN proteins. This model is consistent with the traditional linear view of kinase signalling, in which each group of kinases (isoforms) has one specific function. At the level of target specificity, this linearity is reflected by each group of isoforms having a high affinity to a specific phospho-site or set of phospho-sites. In a linear model, this phospho-site affinity is not shared with other kinases. Phospho-site in this context refers to the amino acids in close proximity to the phosphorylated residue.

In the first results chapter I present evidence that this linear view of kinase function might not apply to the D6PK group of kinases. My KSselect algorithm predicts that the target specificity of D6PKL3 is different from the rest of the D6PK kinases. More specifically, the other D6PK kinases have diverged away from D6PKL3. It is therefore possible that kinase function is not conserved across the D6PK kinases and that D6PKL3 does not target PIN proteins.

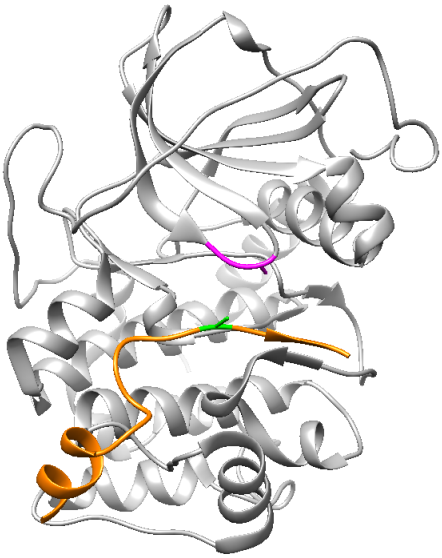
At the outset of my analysis of the role of D6PKL3 in gravitropism it was pivotal to further characterise the target specificity of D6PKL3. The KSselect results show that D6PKL3 has an ancestral target specificity, which is not shared with the other D6PKs (see Chapter 3). The KSselect algorithm predicts differences in target specificity based on amino acids proximal



(a) PKA P-3

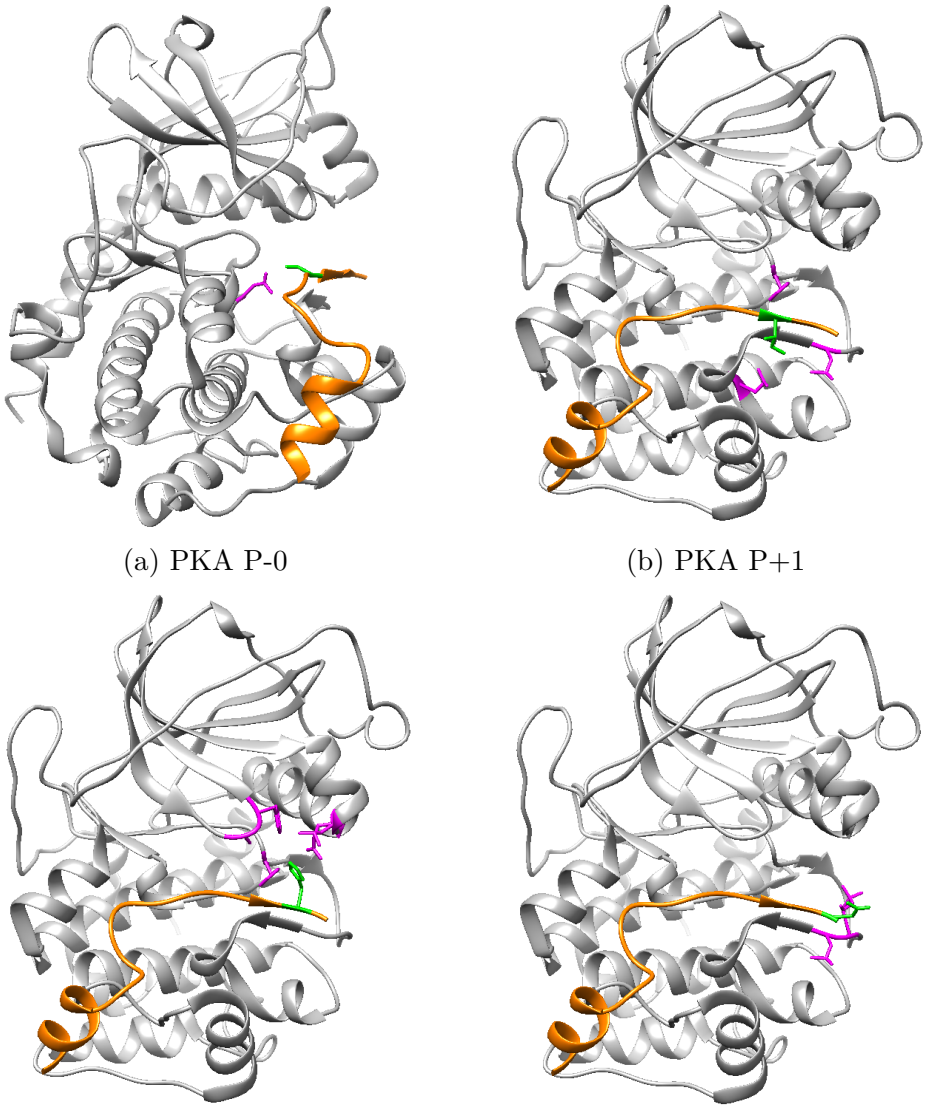


(b) PKA P-2



(c) PKA P-1

Figure 4.3: [Part 1] Schematic representation of the 3D structure of the PKA kinase (grey) and its substrate (orange). Subfigures (a-c) show the "binding-pockets" / PRBPs (purple) for each phosphorylation site residue (P-3, P-2 and P-1)(green). Generated using Chimera.



(a) PKA P-0 (b) PKA P+1

(c) PKA P+2 (d) PKA P+3

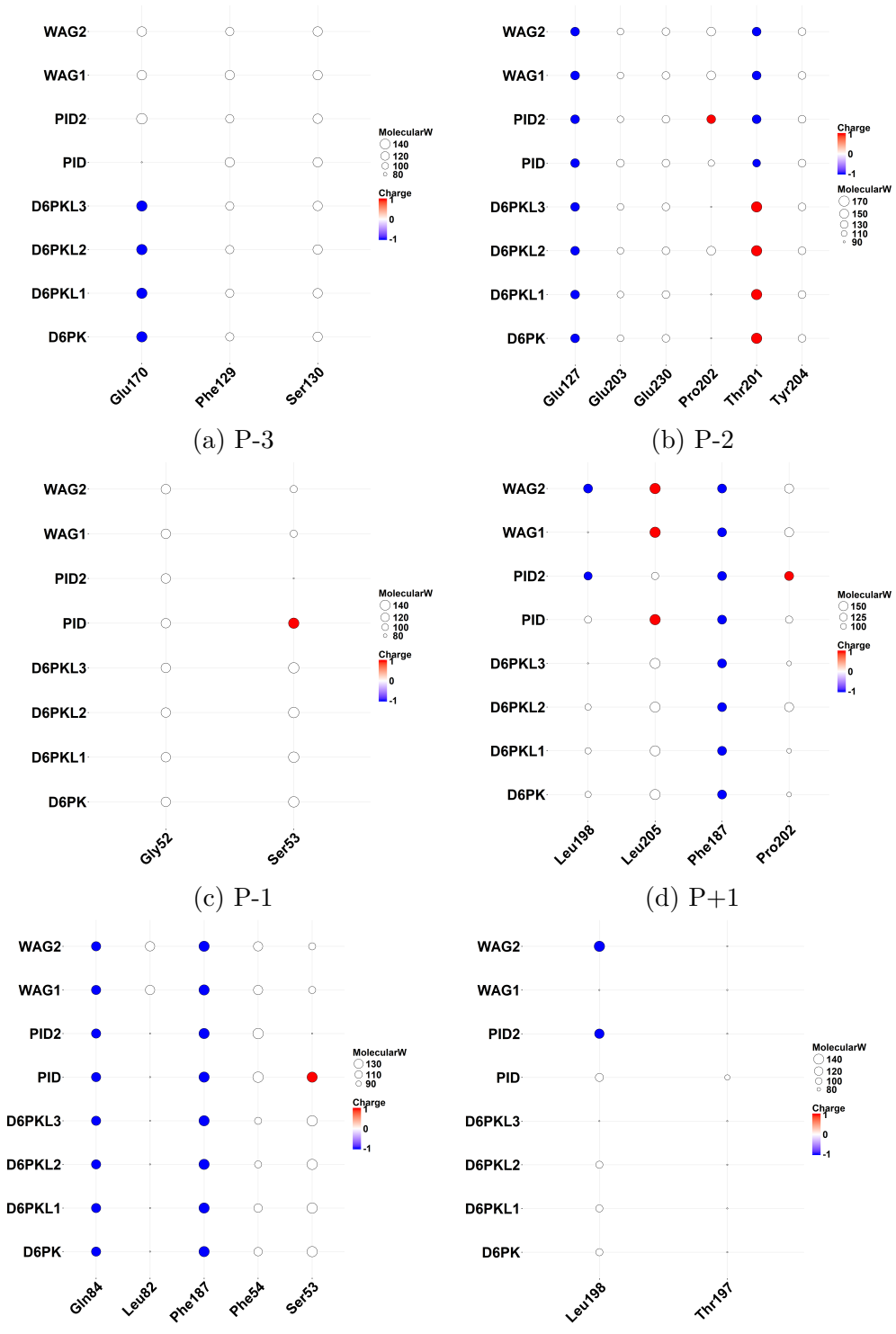
Figure 4.4: [Part 2] Schematic representation of the 3D structure of the PKA kinase (grey) and its substrate (orange). Subfigures (a-d) show the ”binding-pockets” / PRBPs (purple) for each phosphorylation site residue (P-0, P+1, P+2 and P+3) (green). Generated using Chimera.

and distal to the substrate binding site (domains of specificity/DoS). It is however unable to make predictions about the phospho-site preference of a kinase.

In order to make prediction about the phospho-site preference of D6PKL3 and other kinases, I generated the Phospho-site Residue "Binding Pockets" visualisation (PRBPviz) script (see Section 2.4.4). PRBPviz is an adaptation of an algorithm (PREDIKIN) by Brinkworth *et al.* (2003) that predicts the optimum phospho-site for a given kinase domain. Unlike KSselect, PRBPviz disregards distal amino acids involved in phospho-site preference (distal DoS). PRBPviz uses the chemical properties of proximal DoS to make predictions about likely target phospho-sites.

Because PRBPviz is based on PREDIKIN (Brinkworth *et al.*, 2003), it is important to first understand this original algorithm. Part of the PREDIKIN algorithm deals with determining which amino acids in the kinase domain are structurally close to single residues in the target phospho-site. Amino acids on the kinase domain that are found to be in proximity to one residue of the phospho-site are assumed to function as a binding pocket of this residue (see Figure 4.3 and 4.4). I will subsequently refer to these kinase domain amino acids as Phospho-site Residue "Binding Pockets" (PRBPs). I named the PRBPs following the conventional nomenclature for phospho-site residues, which is P-0 (phosphorylated residue), P-1 (residue N-terminal to P-0) and P+1 (residue C-terminal to P-0) (see Figure 4.3 and 4.4). PREDIKIN (Brinkworth *et al.*, 2003) uses the crystal structure of Protein Kinase A (PKA) in complex with a substrate peptide to determine the PRBPs for the PKA kinase (see Figure 4.3 and 4.4). Due to the high structural conservation between all eukaryotic kinase domains, these PRBPs can be extrapolated to all eukaryotic kinases.

PREDIKIN uses the chemical properties of the PRBPs to predict the optimum phospho-site for a given kinase. In contrast, PRBPviz plots the



(a) P-3 (b) P-2

(c) P-1 (d) P+1

(e) P+2 (f) P+3

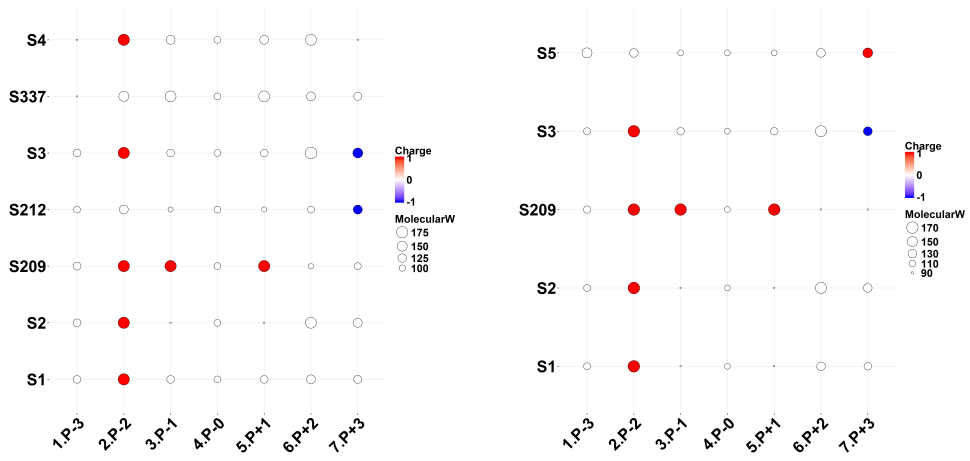
Figure 4.5: Balloon plots of Phospho-site Residue Binding Pockets (PRBPs) for PIDs and D6PKs. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)

chemical properties of the PRPBs as a balloon plot (e.g. Figure 4.5). These plots show the charge (at physiological pH) and molecular weight of the amino acid side chain as colour and diameter, respectively. These plots provide a tool to predict the phospho-site preference of a kinase domain.

Both the D6PK and PID group of kinases were used as input data for the PRBPviz script. For these kinases, PRBPviz generates one balloon plot for each PRBP (see Figure 4.5). All PRBPs of the D6PK clade had mostly conserved chemical properties. This suggests that the divergence of D6PKL3 target specificity (see Chapter 3) is likely caused by distal DoS. In contrast, there is a clear divergence between the D6PK and the PID group of kinases. One difference is that the PID kinases contain a positively charged amino acid in the P+1 PRBPs (see Figure 4.5d), not present in the D6PK kinases (see Figure 4.5d). Whereas the charge of PRBPs residues in D6PKs was conserved, this was not observed for the PID kinases. For example, only PID contains a positively charged residue in its P-1 PRBP (see Figure 4.5c). PRBPviz balloon plots for the entire AGC-VIII family can be found in the Supplemental Figures (see S17 - S23).

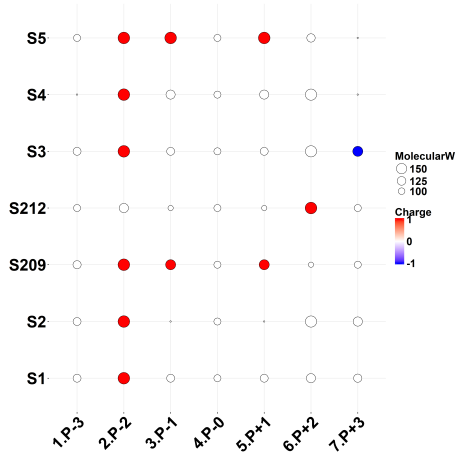
In addition to generating PRBP balloon plots (e.g. Figure 4.5), PRBPviz can also generate phospho-site balloon plots (e.g. Figure 4.6). Examining PRBP and phospho-site balloon plots in parallel reveals the true potential of PRBPviz.

I generated phospho-site balloon plots for all known PIN phospho-sites using PRBPviz (see Figure 4.6). Each PIN balloon plot shows the phospho-site residues of one PIN protein. This is different to the kinase balloon plots that show the residues of one PRBP across different kinases. Each PIN protein contains a subset of eight known phospho-sites, which are partially conserved across PINs (see Figure 4.6). Examining both the D6PK/PID and the PIN balloon plots will help us understand how differences in PRBPs relate to known and potential target phospho-sites.

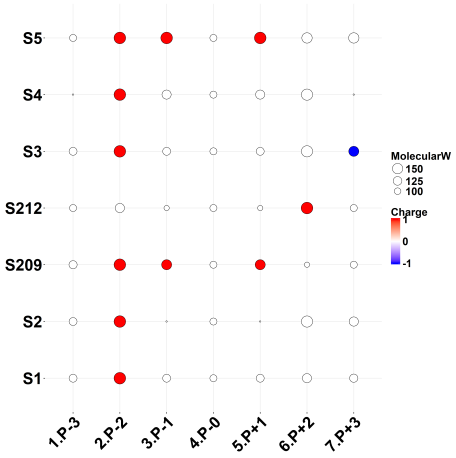


(a) PIN1

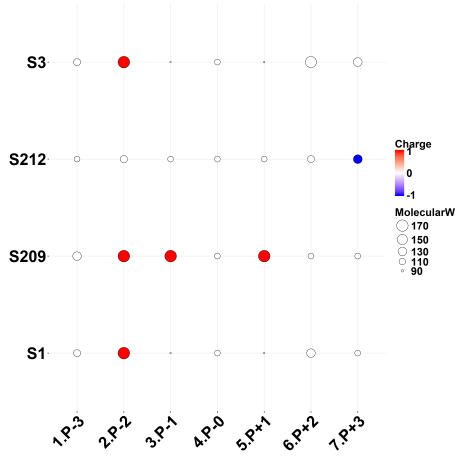
(b) PIN2



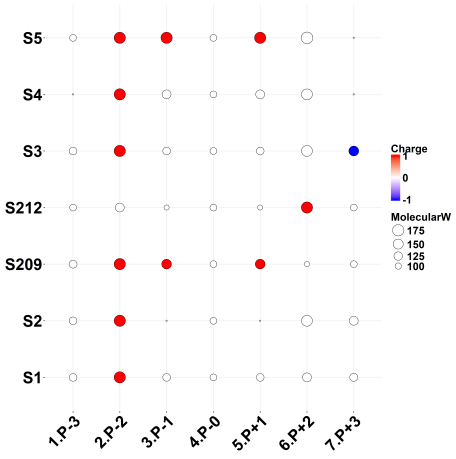
(c) PIN3



(d) PIN4



(e) PIN6



(f) PIN7

Figure 4.6: Balloon plots of phosphorylation sites on PINs (S1, S2,...). Colours indicate charge and circle diameter indicates molecular weight of residue. Generated using PRBPviz (see Section 2.4.4)

The most important residue determining target specificity of a kinase domain is the residue corresponding to Glu127 in PKA (P-2 PRBP). This residue is conserved and negatively charged throughout D6PKs and PIDs (see Figure 4.5). A negative charge at Glu127 confers a requirement for a positive residue at P-2 (Goldberg *et al.*, 1996). Most PIN phospho-sites fulfil this requirement with the exception of S5 in PIN2 (see Figure 4.6b) and S212 in PIN1, PIN3, PIN4, PIN6, and PIN7 (see Figure 4.6). The phospho-sites S5 in PIN2 and S212 are therefore not phosphorylated by any PID or D6PK kinase.

Unlike the residue at Glu127, it is not known whether other PRBP residues confer a charge requirement. The plots can however be used to predict the phospho-site preferences of a kinase. PIN phospho-sites can be split into three groups based on their charge pattern. The first group consists of the S1, S2 and S4 phospho-sites of every PIN protein. It also includes the S3 phospho-site of PIN6. This group is characterised by the absence of charged residues at any position except the P-2 position (positive) (see Figure 4.6). I will subsequently refer to this group as the Minimal Charge group (MC-group). The second group comprises of phospho-sites with a positive charge at P-2, P-1 and P+1. These phospho-sites include the S5 and S209 sites of all PIN protein, with the exception of S5 in PIN2 (see Figure 4.6). I will subsequently refer to this group as the Selective Charge 1 group (SC1-group). The third group consists of the S3 site of all PINs, with the exception of PIN6. It is characterised by a positive charge at P-2 and a negative charge at P+3 (see Figure 4.6). I will subsequently refer to this group as the Selective Charge 2 group (SC2-group).

The PIN phospho-site preferences of PID and D6PK were investigated by Zourelidou *et al.* (2014). The most striking difference between PID and D6PK is how they phosphorylate a mutated PIN1 protein (deletion of S1, S2 and S3 phospho-sites / $PIN1^{S1-S3}$). In $PIN1^{S1-S3}$, PID phosphorylation is

reduced by approximately 80%, compared to a reduction of 20% for D6PK (Zourelidou *et al.*, 2014). This mutated PIN1^{S1-S3} protein only contains one MC-group phospho-site (S4) and one SC1-group phospho-site (S209) (see Figure 4.6a).

Differences in the phosphorylation of PIN1^{S1-S3} by PID and D6PK were thought to be a result of PID having a phospho-site preference for S1, S2, and S3 (Zourelidou *et al.*, 2014). My PRBPviz results for D6PK/PID and PIN1 allow me to formulate a more elaborate hypothesis to explain these differences in PIN1^{S1-S3} phosphorylation. The PIN1^{S1-S3} MC-group phospho-site (S4) does not carry any charge except at its P-2 residue, and is therefore compatible with both PID and D6PK PRBPs (see Figure 4.5b). I hypothesise that MC-group phospho-sites are general phospho-sites, which are targeted equally by D6PKs and PIDs. In contrast, the PIN1^{S1-S3} SC1-group phospho-site (S209) is positively charged at residues P-1 and P+1, which repulses PIDs positively charged P-1 and P+1 PRBPs (see Figure 4.5c and 4.5d). I hypothesise that the SC1-group phospho-sites are only targeted by D6PK kinases (see Figure 4.7). Given the kinase preferences predicted by PRBPviz, PIN1^{S1-S3} is phosphorylated by D6PK at two sites and by PID at only one site, which is consistent with the experimentally determined differences in phosphorylation

In addition to explaining the observed differences in PIN1^{S1-S3} phosphorylation between D6PK and PID (Zourelidou *et al.*, 2014), PRBPviz is capable of making predictions regarding the phospho-site preferences of other AGC-VIII kinases. I hypothesise that PID2 and WAG2 are repulsed from the S3 site of PINs (except PIN6), which is negative at P+3 (see Figure 4.6). The P+3 binding pockets of PID2 and WAG2 also contain a negative residue, which could prevent binding at these phospho-sites (see Figure 4.5f).

PRBPviz is a reductionist approach that focuses only on the proximal

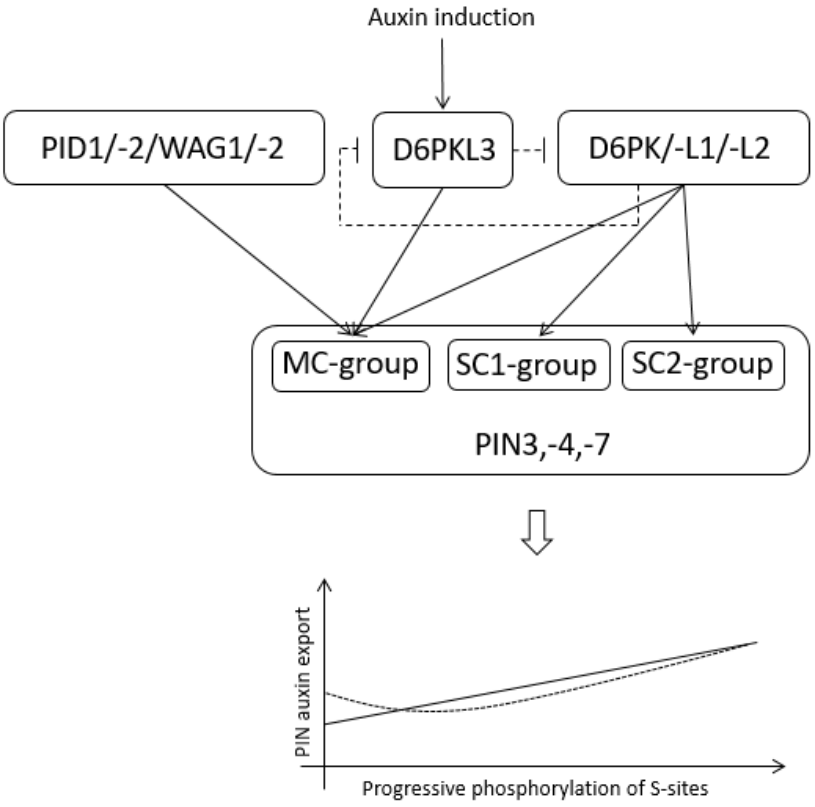


Figure 4.7: Schematic of D6PK and PID group function based on PRBPviz results. PRBPviz results suggest multiple target phospho-site types (MC-group, SC1-group and SC2-group) for one group of kinase and cross talk between the kinases (shared target phospho-site types). Dashed lines indicate potential Auxin Responsive Backup Circuit formed by D6PKL3 and other kinases. Graph (bottom) indicates unknown gradual PIN-activation by D6PK kinases. Dashed curve indicates PIN-activation expected, if ARBC is formed.

DoS when predicting the target-specificity of kinases. Nevertheless, it provides unprecedented insights into the molecular mechanisms governing kinase phospho-site specificity. The most unconventional aspect of our hypothesis is the existence of general phosphorylation sites (MC-group sites), which are equally phosphorylated by PID and D6PK kinases. These general MC-group sites add another level of redundancy to auxin transport signalling (additional to the redundancy at kinase level). Phosphorylation at MC-group sites therefore represents a point of cross-talk between PIDs and D6PKs. This cross-talk contradicts the linear model of signal transduction (see Figure 4.7).

The existence of general MC-group sites on PIN proteins also implies that D6PKL3 is likely targeting PIN proteins. The unique target specificity of D6PKL3, predicted by KSselect, appears to result primarily from differences in distal DoS. We hypothesise that phospho-sites targeted by D6PKL3 show a structural, rather than a charge, difference.

4.3 D6PKL3 Expression and Targeting

Pathways are traditionally drawn as separated linear entities. However, this rather reflects the history of how they were discovered than their real functional context. It has turned out that pathways are extensively connected and embedded in networks. The economy of evolution has produced a multitude of links and crosstalk between pathways and even led to the reuse of protein components in different pathways. Thus, the specificity of biological responses is largely generated by the combinatorial integration of pathway crosstalk and the versatility of component function.

Kolch et al. (2005)

The PRBPviz results suggest that PINs carry general phosphorylation sites (MC-group sites) that have equal affinity to both D6PKs and PIDs. This redundancy at phospho-site level suggests that PINs are targeted by D6PKL3. It is therefore likely that D6PKL3 plays a role in auxin transport, which is a key insight into its possible role in gravitropic signalling.

The existence of MC-group sites adds complexity to the D6PKL3 network. In this section I present results suggesting that D6PKL3 is phosphorylating another protein, in addition to PINs. Having multiple target proteins represents another deviation from the linear model. This suggests that D6PKL3 must be viewed in a systemic way as part of a network, allowing for both cross-talk and multiple targets.

The gravitropic response of roots is characteristically split into three

phases that are spatially separated. Firstly, gravi-sensing takes place in the columella cells. Secondly, auxin gradient re-configuration occurs in the lateral root cap and meristem epidermis. Thirdly, the gravitropic response takes place in the elongation zone (see Section 1.1). D6PKL3 is known to be highly expressed in columella cells (Zourelidou *et al.*, 2009). The most obvious hypothesis is that D6PKL3 functions in gravi-sensing. It is unclear why such a role would require D6PKL3 to be auxin inducible, and this may suggest other roles or layers of regulation that we don't as yet understand. As local auxin levels elevate during gravitropic signalling, auxin induction of D6PKL3 is likely a key factor in D6PKL3 function.

To investigate the function of D6PKL3 auxin induction I treated our promoter GUS fusion line with the synthetic auxin NAA (see Section 2.2.5). This allows me to see where auxin induction of D6PKL3 takes place. Auxin-induced D6PKL3 expression was observed in the basal meristem, which represents a new expression domain in addition to the columella cells (see Figure 4.8 and Supplemental Fig. S25). Expression in the columella cells only shifted slightly to the peripheral cells (see Figure 4.8 and Supplemental Fig. S25). D6PKL3 is co-expressed with D6PKL1 in both the columella cells and basal meristem (see Figure 4.1), which could suggest that both function redundantly in these domains.

D6PKL3 has two different expression domains, the auxin inducible basal meristem domain and the columella cell domain. The KSselect algorithm predicts that protein targeting is conserved across the D6PK group. D6PK, the founding member of the D6PK group, has been shown to localise to the rootward membrane in most cells (Zourelidou *et al.*, 2009). However, it was not previously known whether rootward targeting is conserved in D6PKL3. I therefore analysed the signal of our D6PKL3-YFP translational fusion line to determine D6PKL3 localisation. In the basal meristem, D6PKL3 seems to be targeted to the rootward plasma membrane (PM), which is consistent

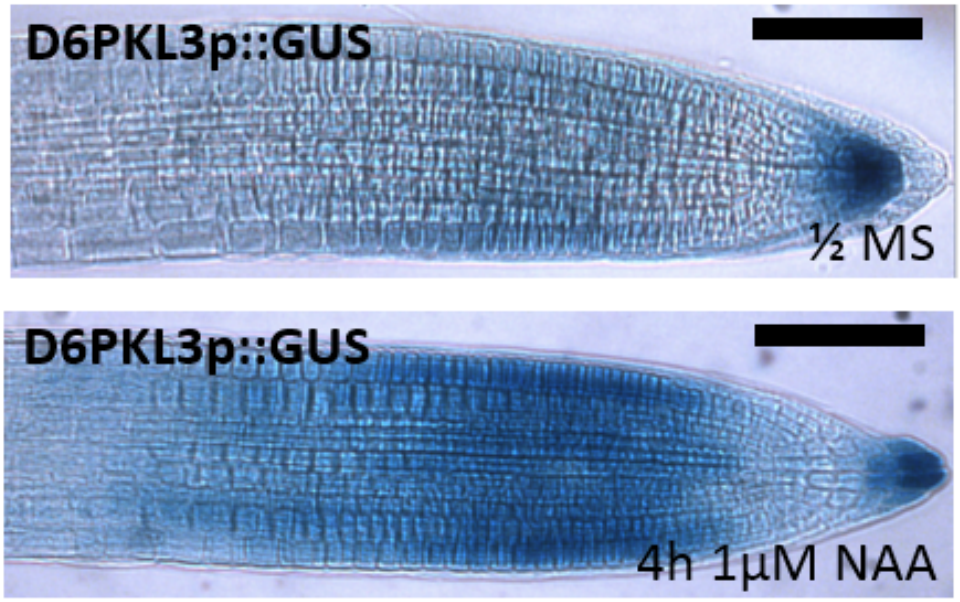
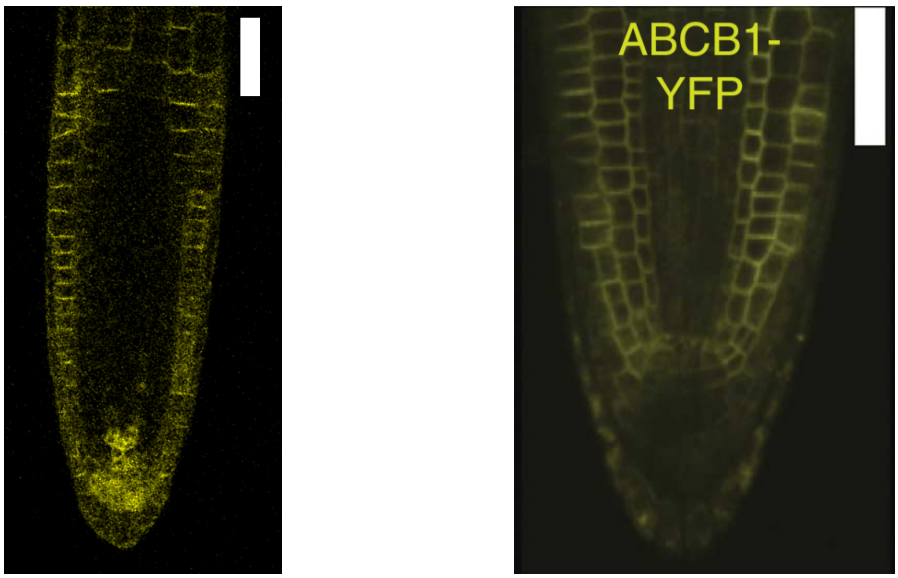


Figure 4.8: D6PKL3 expression after 4 hours of auxin treatment (bottom) and under control conditions (top). Under control conditions, D6PKL3 is mainly expressed in the columella cells. After 4 hours of auxin treatment, D6PKL3 expression is shifted to the peripheral columella cells and is additionally expressed in the basal meristem, as shown by promoter GUS fusion. [bar = 75μm]

with D6PK localisation in other cell types (see Figure 4.9a). In columella cells D6PKL3 accumulates intracellularly, meaning it does not accumulate on a particular side of the PM (see Figure 4.9a).

D6PKL3 shows different targeting in columella cells and the basal meristem. In columella cells the apolar localisation of D6PKL3 is consistent with its function in PIN-phosphorylation. In a vertically growing root, the three PINs that are expressed in columella cells (PIN3, PIN4, and PIN7) also localise in an apolar fashion (Kleine-Vehn *et al.*, 2010). I therefore hypothesise that in columella cells D6PKL3 and D6PKL1 redundantly phosphorylate PIN3, PIN4, and PIN7 (see Figure 4.11). This model for D6PKL3 function in columella cells is not linear due to cross-talk with PIDs expressed in this domain.

D6PKL3 localisation in columella cells is consistent with its function as a PIN activating kinase. In contrast, in the basal meristem D6PKL3 does



(a) D6PKL3::D6PKL3:YFP (b) ABCB1::ABCB1:YFP

Figure 4.9: Translational fusions of (a) D6PKL3 and (b) ABCB1 show overlapping expression in the root tip. D6PKL3 and ABCB1 appear to co-localise at the rootward plasma membrane in the basal meristem. In columella cells, D6PKL3 mainly localises to intracellular structures. ABCB1-YFP image from Henrichs *et al.* (2012). [bar = 50μm]

not appear to obviously co-localise with any PIN proteins. In the basal meristem, D6PKL3 is predominantly expressed in the epidermis where it localises to the rootward PM. PIN2 is the only PIN that is expressed in the root epidermis, where is localises to the shootward PM. Therefore, D6PKL3 does not co-localise with any PIN in the basal-meristem epidermis.

It is a common characteristic of kinases to have multiple targets (Jordan *et al.*, 2000). When a kinase has multiple targets, the target proteins are often part of a protein complex. I therefore searched through the literature for auxin transport-related proteins that are phosphorylated by AGC-VIII kinases. A particular group of proteins, the ABC (ATP-binding casset) transporters, stood out due to their interaction with PINs on the PM (Blakeslee *et al.*, 2007).

Two lines of evidence point to a subset of ABCB transporters as being likely D6PKL3 targets. Firstly, the ancestral AGC-VIII kinase, PHOT1, inhibits ABCB19 function via direct phosphorylation. D6PKL3, also an AGC-

VIII kinase, could have conserved this function (Christie *et al.*, 2011). Secondly, the AGC-VIII kinase PID has been shown to phosphorylate ABCB1, meaning that a kinase closely related to D6PKL3 could have conserved the ability to phosphorylate ABCBs (Henrichs *et al.*, 2012).

Of the three *Arabidopsis thaliana* ABCBs (ABCB1, ABCB4 and ABCB19) that have a well characterised role in auxin transport, ABCB1 expression most resembles that of D6PKL3 in the basal meristem (Cho and Cho, 2013; Titapiwatanakun and Murphy, 2009)(see Figure 4.9). ABCB4 and ABCB19 expression partially overlap with ABCB1 (Titapiwatanakun and Murphy, 2009). In the basal meristem, all three ABCBs localise apolarly. I hypothesise that upon an increase in auxin response, D6PKL3 locally inhibits ABCB1, ABCB4 and ABCB19 on the rootward PM. The resulting reduction in rootward auxin transport leads to an increase in shootward auxin transport.

ABCBs are known to be regulated via phosphorylation in their linker regions (also know as B-regions) (Henrichs *et al.*, 2012). PID has been suggested to phosphorylate ABCB1 at the linker region phospho-site S634 (Henrichs *et al.*, 2012). To evaluate whether any phoshpo-site in the ABCB1 linker region can be phosphorylated by D6PKL3, I used our PRBPviz to analyse all ABCB1 linker region phospho-sites (see Section 2.4.4). My results show that there are six phospho-sites in the ABCB1 linker region that carry a negative charge at P-2, which is the basic requirement for phosphorylation by AGC-VIII kinases (see Figure 4.10). The six phospho-sites are S621, S622, S626, S631, S641, and S653. Two of these sites, S631 and S641, show the same charge pattern as the Minimal Charge (MC)-group phospho-sites on PIN proteins (see Figure 4.10). D6PKL3 could phosphorylate ABCB1 at these two sites. I will subsequently also refer to S631 and S641 as MC-group sites.

My PRBPviz results for ABCB1 linker region phospho-sites bring pre-

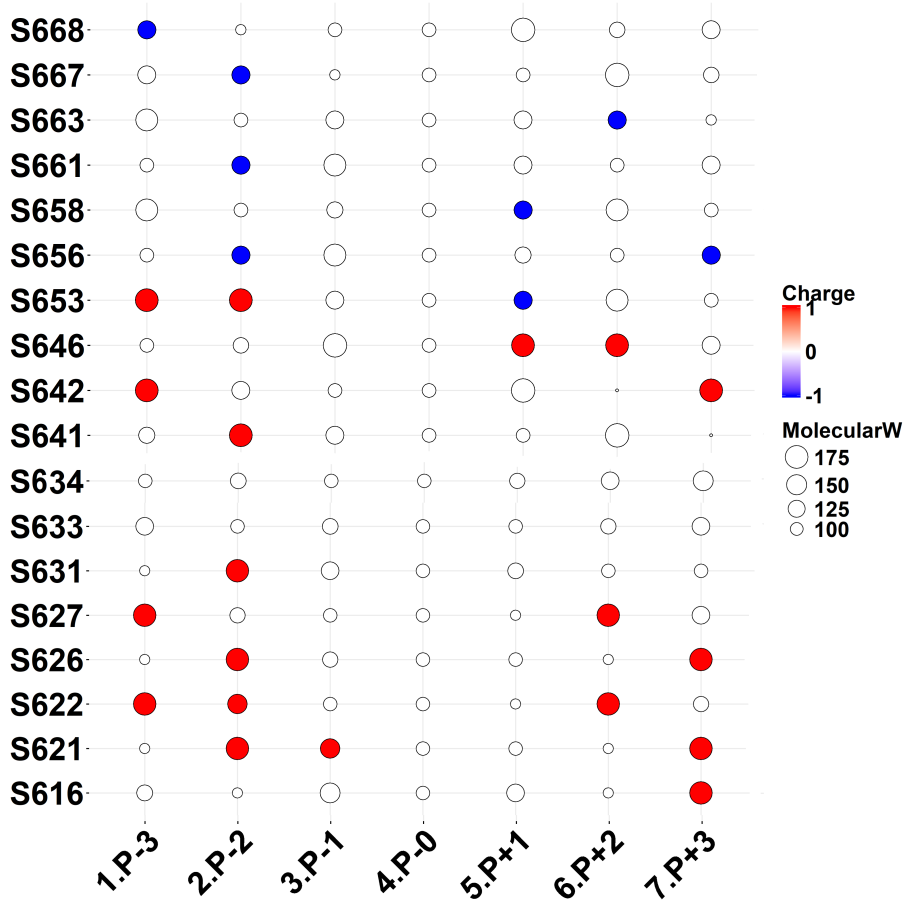


Figure 4.10: Balloon plot of phosphorylation sites on the ABCB1 linker region (S616, S621,...). Colour indicates charge and circle diameter indicates molecular weight of residue. Generated using PRBPviz (see Section 2.4.4)

vious findings regarding PID into question. PID was assumed to phosphorylate ABCB1 at the S634 site (Henrichs *et al.*, 2012). I argue, however, that this phosphorylation occurs at the S631 site. This argument is based on two observations. Firstly, the S634 site does not carry a negative charge at P-2, which is required for targeting by PID (see Figure 4.10). Secondly, the MC-group site S631 is found within the same protein fragment that was used as evidence for PID phosphorylation at S634 (Henrichs *et al.*, 2012).

As PHOT1 inihibts ABCB19, I hypothesise that D6PKL3 has conserved the ability to negatively regulate ABCBs. The rootward localisation of D6PKL3 would therefore accelerate shootward auxin transport. The auxin inducible D6PKL3 is co-expressed with D6PKL1 in the epidermis of the basal meristem. I therefore hypothesise that the acceleration of shoot-

ward auxin transport is regulated by an Auxin Responsive Backup Circuit (ARBC) that is formed by D6PKL3 (controller) and D6PKL1 (responder) (see Figure 4.11) (see Chapter 3).

My hypothetical model of the D6PKL3 network shows a dual function for D6PKL3 in gravitropism (see Figure 4.11). First, D6PKL3 phosphorylates PIN3, PIN4 and PIN7 in columella cells, which potentially contributes to the establishment of the auxin gradient (see Section 1.1). Secondly, D6PKL3 phosphorylates ABCB1 in the basal meristem epidermis, which likely contributes to auxin gradient re-configuration (see Section 1.1). It is likely that both D6PKL3 functions are performed redundantly by D6PKL1, which is co-expressed with D6PKL3 in both columella cells and the basal meristem (see Figure 4.11).

4.4 Gravitropic Response of D6PK Mutants

Testing model prediction requires the continuous production of relevant quantitative biological data. This is a major challenge for biologists, but constant innovations (Ehrhardt and Frommer, 2012) are opening fantastic opportunities for plant scientists.

A. Middleton (2012)

I hypothesise that D6PKL3 plays a dual role in gravitropism. Firstly, I predict that auxin induction of D6PKL3 in the basal meristem contributes to gravitropic kinetics. In this expression domain, D6PKL3 localises to the rootward PM, where I hypothesise it inhibits ABCB1 via phosphorylation. Auxin induction of D6PKL3 therefore inhibits rootward auxin efflux in the epidermis of the basal meristem. This accelerates shootward (basipetal) auxin transport (BAT) through the basal meristem, which increases auxin flux to the elongation zone during gravitropism. Secondly, I predict that D6PKL3 contributes to the establishment of the auxin gradient in columella

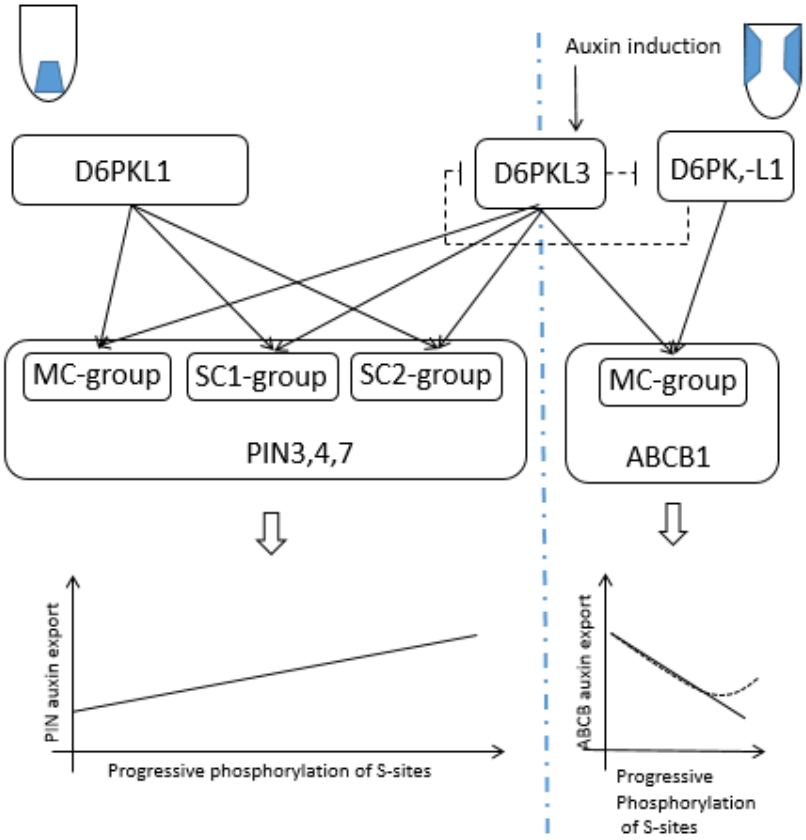


Figure 4.11: Schematic of D6PKL3 function in columella cells (left) and basal root meristem (right). Expression, targeting and PRBPviz results for D6PKL3 suggest that D6PKL3 has two targets: PIN in the columella cells and ABCB1 in the basal mersitem. Dashed lines indicate potential Auxin Responsive Backup Circuit formed by D6PKL3 and D6PKL1 in the basal meristem. Graphs (bottom) indicate unknown gradual PIN-activation and ABCB1-inhibition by D6PK kinases. Dashed curve indicates ABCB1-inhibition expected, if ARBC is formed.

cells. Here, D6PKL3 localises apolarly, which is consistent with its hypothesised function in phosphorylating PIN3, PIN4 and PIN7.

Three observations led us to hypothesise that a *d6pkl3* knock-out mutant would exhibit a subtle gravitropic defect. Firstly, in both expression domains D6PKL3 is co-expressed with D6PKL1, meaning the two could function partially redundant. Secondly, I hypothesise that in the basal meristem, D6PKL3 forms an Auxin Responsive Backup Circuit (ARBC) with D6PKL1. A *d6pkl3* mutant would therefore only lack the ”responder” component of the ARBC, while still maintaining the ”regulator” function.

Thirdly, another molecular mechanism involving PIN2 exists by which auxin accelerates its own transport to the elongation zone during gravitropism (see Section 1.1). This mechanism could act redundantly with D6PKL3 in the basal meristem.

My phenotyping approach therefore had to be able to detect subtle differences in the gravitropic kinetics. To this end, I implemented the Quantitative Gravitropic Analysis Pipeline (qGAP) (see Chapter 6). The qGAP is fast, sensitive and flexible. It uses time-series images that are analysed using the RootTrace software, the qGAP module, and the qGAP Jupyter notebooks (see Chapter 6). The main feature enabling qGAP to discern subtle differences in gravitropic kinetics from biological variation is its high throughput nature. Additionally, qGAP offers an approach to deal with the biological variation in a dataset (see Chapter 6). One typical source of variation in a qGAP dataset is a wide range of seedling ages. Seedling age (developmental stage) strongly influences gravitropic kinetics. The determination of the developmental stage of a seedling using images is difficult and manual examination of the plates would be required. Using qGAP avoids this laborious process by using root growth rate as a proxy for seedling age. Although outliers for root growth rate can be removed from the dataset, we consider it good practice to also show the whole dataset analysis.

The qGAP data analysis can therefore be split into four logical steps, each contained in a separate notebook. Firstly, the data is read and initial filter functions are applied (qGAP_read.ipynb). Secondly, an initial mixed model analysis is run on the whole dataset (qGAP_init.ipynb). Thirdly, using data from the ”qGAP_init”-notebook, the dataset is filtered based on root growth rate (qGAP_filt.ipynb). Fourthly, a final mixed model analysis is run on the filtered dataset (qGAP_fin.ipynb) (see Chapter 6).

In previous sections I began uncovering the complexity of the D6PKL3 network. The different levels of D6PKL3 network complexity suggest that

mathematical or computational models will be required to determine the biological function of D6PKL3. The parametrisation of these models requires quantitative data, meaning it was pivotal that my gravitropic phenotyping was quantitative in nature. By using linear mixed models (LMMs), qGAP is able to provide a quantitative description of gravitropic kinetics (see Chapter 6).

My first qGAP analysis of gravitropic kinetics compared the *d6pkl3* mutant, the *d6pk d6pkl1 d6pkl2* triple mutant, and the wild type line (Col-0)("D6PKL3init"-dataset)(see Figure 4.12). The qGAP automatically analyses growth rate in parallel to gravitropic kinetics. The experimental design followed a standard CPIB protocol for gravitropic assays, with a seedling age of five days at the time of analysis (see Section 2.3.1). The "qGAP-read"-notebook filter function was mainly ran with its default parameters, including an initial angle range of -10 to 10 degrees (standard CPIB protocol). Optimum values for trace loss parameters were determined via trial and error (see Chapter 6).

Analysing D6PKL3init plots for Angle vs Time and Growth vs Time (qGAP_read.ipynb) show that both the gravitropic and growth kinetics follow a linear function (see Chapter 6). Root growth rate and bending rate were therefore analysed using linear mixed-effect regression, LMER, using the lmerTest package in R (Kuznetsova *et al.*, 2017). This analysis is contained in the "qGAP_init"-notebook (whole dataset), and in the "qGAP_fin"-notebook (growth rate filtered dataset). In both notebooks, fixed effects are given as Time and Line.ID (with interaction terms). As random effects we chose intercepts for Root.ID, and by-Root.ID random slopes for the effect of Time (see Listing 4.1).

Listing 4.1: Linear Mixed Model R Code

```
mlm <- lmer(Growth~Line*Time + (Time|Plate.ID:Root.ID), data=df)
```

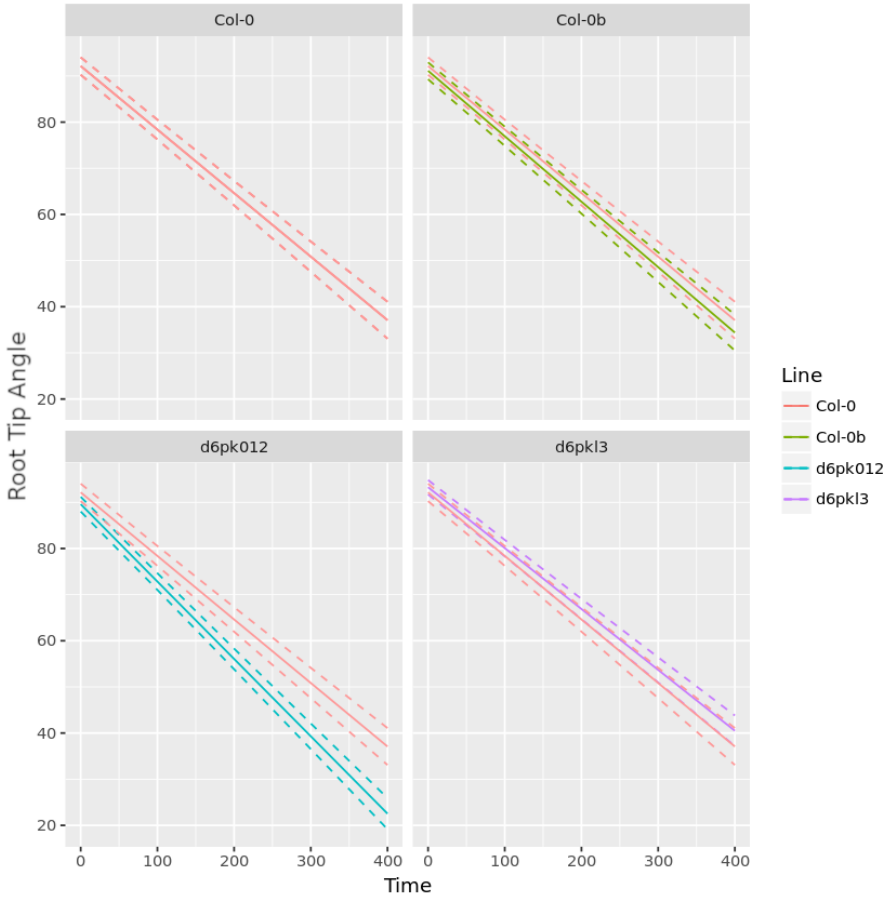


Figure 4.12: Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* triple mutant and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. See Table 4.1 for corresponding summary of linear mixed model. Time is given in minutes and angle in degree.

In the qGAP method chapter (see Chapter 6), I compare the effect of the standard deviation parameter given in the "qGAP-filter" notebook. I show that both outlier removal (3*standard deviation/3*std) and the selection of similar growth rates (1.5*standard deviation/1.5*std) resulted in a similar increase in the power of the mixed model (compared to the whole dataset). The increase in power in both the 3*std and 1.5*std datasets is due to the exclusion of the youngest and oldest seedlings. Delayed germination is likely the cause of high variation in seedling age. High variation in seedling age skews the analysis of gravitropic kinetics, as seedlings show a different gravitropic response depending on their developmental age. I consider the 3*std

Table 4.1: Summary of Linear Mixed Model for 3*standard deviation filtered "D6PKL3init"-dataset. See Figure 4.12 for corresponding plot

	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	92.15	0.96	96.47	95.66	0.00
LineCol-0b	-1.26	1.33	95.43	-0.95	0.34
Lined6pk012	-2.56	1.26	96.13	-2.03	0.05
Lined6pkl3	1.12	1.25	95.74	0.90	0.37
Time	-0.14	0.01	95.71	-25.54	0.00
LineCol-0b:Time	0.00	0.01	95.17	0.36	0.72
Lined6pk012:Time	-0.03	0.01	95.53	-4.25	0.00
Lined6pkl3:Time	0.01	0.01	95.30	0.82	0.41

dataset to most accurately represent the effect of genotype on gravitropic kinetics, as it has increased test power while conserving sample size. For the sake of brevity I will only report the results from the analysis of the 3*std dataset (refer to the Supplemental Figure S27 and Supplemental Table S6 for the results for the whole dataset).

Visual inspection of residual plots did not reveal any obvious deviations from homoscedasticity or normality. P-values were obtained by likelihood ratio tests of the full model against the model with the effect in question removed. The 'lmerTEST' package presents the significance of each fixed effect in a ANOVA-style table. However, each coefficient (intercept or slope) represents the mean of all roots, with each root having its own intercept and slope. This is determined by the random effects specified in the mixed model (see Listing 4.1). For the sake of brevity, here I only present the F tests of the LMM analysis results (Satterthwaite degrees of freedom approximation).

There was no significant difference in the growth rates of *d6pkl3*, *d6pk d6pkl1 d6pkl2*, and both Col-0 lines (see Supplemental Fig. S26 and Supplemental Tab. S5). This is advantageous to my analysis as it rules out differences in growth rate as being the underlying cause of differences in gravitropic kinetics. Similarly, there is no significant difference in the intercepts of the bending rate regression (see Supplemental Fig. S26 and Supplemental Tab. S5). This is also beneficial for my analysis, because I aim to compare the gravitropic kinetics of seedlings close to 90 degrees from

vertical at time-point zero.

In *d6pk d6pkl1 d6pkl2*, the increase in angle over time (slope) was -0.03 degree/minute, which was significantly lower than Col-0 (p = 0.00) (see Figure 4.12 and Table 4.1). The triple mutant therefore exhibits accelerated gravitropic kinetics. Neither *d6pkl3* nor the second Col-0 line (Col-0b) show significant differences in angle over time (slope) compared to Col-0 (see Figure 4.12 and Table 4.1). A knock-out mutant of D6PKL3 therefore does not affect gravitropic kinetics. The minimal detectable difference for *d6pkl3* was around -0.015 degree/minute (see Section 6).

I have shown with high sensitivity that the D6PKL3 single mutant does not show a gravitropic phenotype. It is likely that in both expression domains, D6PKL3 functions redundantly with D6PKL1.

4.5 Conclusion

Individual pathways transmit signals along linear tracts resulting in regulation of discrete cell functions. [...] However, as increasingly larger numbers of cell signaling components and pathways are being identified and studied, it has become apparent that these linear pathways are not free-standing entities but parts of larger networks. [...] There are two general classes of [network-interconnections]: junctions, which are signal integrators and nodes, which split the signal and route them to multiple outputs. An immediate complexity that arises from these definitions is that they are not mutually exclusive properties for signalling components.

J. D. Jordan et al. (2000)

The well established canalisation hypothesis states that auxin can modulate its own transport and the direction of that transport (Sachs, 1988). A prominent example of the canalisation hypothesis is found during the reconnection of vascular strands as a regenerative response to wounding. During

this response, auxin is channelled through non-vascular tissue, priming it to become connective vascular strands (Sachs, 2000; Berleth and Sachs, 2001). This example illustrates the positive feedback of auxin on its own transport, plus the directional polarisation of auxin transport relative to its source.

Auxin transport is a key concept in gravitropic signalling, however links to the canalisation hypothesis are rarely drawn. One exception is the increased PIN2 accumulation on shoot-ward PM in the bottom side root epidermis cells during the gravitropic response (see Section 1.1). We predict that D6PKL3 auxin-dependently inhibits ABCB1 in the basal meristem. This hypothesised function of D6PKL3 is potentially another molecular mechanism underlying the canalisation hypothesis.

All of my observations suggest that D6PKL3 functions in a highly complex network. Firstly, D6PKL3 targeting to the root basal meristem epidermis suggests that the kinase targets ABCBs in addition to PINs. Secondly, D6PKL3 expression overlaps with D6PKL1, suggesting cross-talk within the D6PKs. Thirdly, the predicted targets of D6PKL3 (PINs and ABCBs) carry non-selective Minimal Charge (MC) phospho-sites, which suggest cross-talk with kinases outside of the D6PK-clade. The different levels of complexity of the D6PKL3 network suggests that mathematical or computational models will be necessary to simulate the behaviour of this network. The parametrisation of these models requires quantitative data. I therefore designed our phenotyping platform to be quantitative (qGAP). However, despite the high sensitivity of qGAP, I was unable to detect a gravitropic defect for the *d6pkl3* knock-out mutant.

The absence of a gravitropic phenotype in the *d6pkl3* mutant suggests that D6PKL3 either functions in a different pathway, or that its function in gravitropism is being carried out redundantly by another/other kinase(s). D6PKL3 expression, targeting, and predicted phospho-site preference is also consistent with a function in root meristem maintenance. However, a com-

mon phenotype of meristem maintenance mutants is a defect in root growth. The qGAP results show with high sensitivity that *d6pkl3* does not exhibit any defect in root growth. D6PKL3 is therefore unlikely to play a role in root meristem maintenance or redundancy with another kinase masks its phenotype.

Signalling components involved in auxin transport have been shown to exhibit a high level of redundancy. A prominent example of this is the compensatory expression of PIN proteins in PIN single mutants. If one PIN carrier is knocked out, other PIN genes are expressed in its domain to fulfil the function of the absent PIN protein (Vienten *et al.*, 2005). Consistent with the redundancy in other aspects of auxin transport signalling, the D6PK and the PID group of kinases are thought to be functionally redundant within their own group. In this chapter I predict the existence of general phospho-sites (MC-group sites), which represent another level of redundancy in the auxin transport signalling machinery. The landscape of PIN phospho-sites uncovered in this chapter suggest that PIN proteins act as a nexus in auxin transport signalling, potentially being targeted by a plethora of kinases.

Chapter 5

Characterisation and Phenotyping of CIPK4 and CIPK7

5.1 Introduction

A Genome Wide Association Study (GWAS) combined with mutant phenotyping found that the *Zea mays* CBL-Interacting Kinase 15 (ZmCIPK15) gene is associated with a crown root angle / gravitropic set-point angle (GSA) phenotype (J. Lynch, personal comm.). CIPK kinases characteristically signal in stress response pathways (see Section 1.2.3.1), however the crown root angle phenotype of *ZmCIPK15* is not dependent on a stress condition. The resources available in *Zea mays* to further characterise the ZmCIPK15 function are limited. There are more resources and protocols available in the model plant *Arabidopsis thaliana*. In the first chapter, the KSselect algorithm identified both *Arabidopsis thaliana* CIPK4 and CIPK7 as possible orthologs of ZmCIPK15. The KSselect results further show that kinases related to CIPK4 and CIPK7 form an ancient clade (CIPK4/7-clade). For each species examined by KSselect, only a limited number of

CIPK4/7-clade kinases were present (see Chapter 3). This suggests that the CIPK4/7-clade kinases share an ancient function. It is therefore likely that ZmCIPK15 function is also carried out by CIPK4 and/or CIPK7, meaning we are able to utilise the resources and protocols available in *Arabidopsis thaliana* to further investigate ZmCIPK15 function.

I was unable to find any literature regarding the molecular mechanisms of ZmCIPK15. On the other hand there are multiple publications addressing the protein interactions between CBL proteins and *Arabidopsis thaliana* CIPK4/CIPK7. Both CIPK4 and CIPK7 were shown to interact with CBL2 and CBL3 in a yeast two hybrid assay (see Supplemental Fig. S29) (Albrecht *et al.*, 2001). Furthermore, CBL1 interacts with CIPK7 but not with CIPK4 (see Supplemental Fig. S29). The closely related CIPK13 and CIPK18 also exhibit a differential interaction with CBL1, suggesting that CBL-interaction partners do not correlate with phylogenetic relationship (Kolukisaoglu *et al.*, 2004). The interaction between CIPK7 and CBL1 was further confirmed by in-vitro and in-vivo pull downs (Huang *et al.*, 2011). Based on their identified CBL partners, both CIPK4 and CIPK7 are predicted to localise to the tonoplast (CBL2 and CBL3), with CIPK7 also thought to localise to the PM (CBL1) (Batistic *et al.*, 2010) (see Section 1.2.3.2). The predicted dual localisation of CIPK7 is reminiscent of the dual localisation of CIPK24/SOS2 via interaction with CBL4/SOS3 (PM) and CBL10 (vacuole) (Kim *et al.*, 2007). Subcellular localisation studies of CIPK4 and CIPK7 in cell cultures, however, remain inconclusive, as their localisation is masked by a predominant cytoplasmic signal (Batistic *et al.*, 2010).

The dual localisation of CIPK7 is indicative of two different sets of targets, one on the tonoplast and one on the PM. PM-localisation of CIPK7 is consistent with a function in auxin gradient re-configuration (see Section 1.1.3.3.1). Additionally, CIPK7 is auxin inducible, unlike CIPK4. CIPK7

therefore seems likely to be the ortholog of ZmCIPK15. However, the KSselect algorithm predicts that CIPK4 and CIPK7 are biochemically redundant based on their proximity in the DoS dendrogram (see Chapter 3). It it therefore possible that CIPK4 and CIPK7 function redundantly and share the orthologous function of ZmCIPK15. Furthermore, it is possible that CIPK4 and CIPK7 form an Auxin Responsive Backup Circuit (ARBC) (see Chapter 3)

In this chapter I will address the functions of CIPK4 and CIPK7 in primary root gravitropism, which uses pathways that are related to GSA establishment (Roychoudhry *et al.*, 2013). I hope that the insights gained will shed light on the molecular mechanisms underlying the root angle phenotype associated with ZmCIPK15 (J. Lynch, personal comm.). The first section of this chapter will address the expression and targeting of CIPK4 and CIPK7, as this information is currently limited. To explore CIPK4/CIPK7 expression and targeting I utilise transcriptional and translational fusions, respectively. In the second section, I apply our highly sensitive Quantitative Gravitropic Analysis Pipeline (qGAP) in an attempt to find a phenotype connected to the ZmCIPK15 crown root angle phenotype in *Arabidopsis thaliana*. In the final section I use a more traditional approach to address the function of CIPK4 and CIPK7 in stress response signalling.

5.2 CIPK4 and CIPK7 Expression, Targeting and Gravitropic Phenotyping

Based on the phylogenetic trees and dendrograms generated by KSselect, CIPK4 and CIPK7 are likely to be biochemically redundant (see Section 3.4). This interpretation of the KSselect output should be viewed with caution. In a multicellular organism, two genes with the same function can be expressed in different domains, meaning the genes are not necessarily

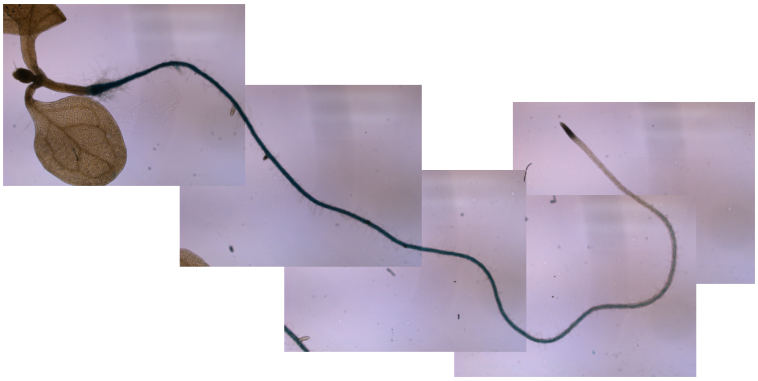
redundant. Before addressing the function of CIPK4 and CIPK7 in gravitropic signalling, it was important to establish their expression patterns. These expression patterns also give further evidence as to what role these kinases are potentially playing in gravitropism.

To determine the expression domains of CIPK4 and CIPK7, I generated GUS-promoter fusions for both genes (see Figure **??**). The GUS signal of both the CIPK4 and CIPK7 lines overlapped, with both being expressed in the root basal meristem and the root maturation zone (see Figure 5.1 and 5.2). CIPK4 and CIPK7 expression was root-specific, which is consistent with a function orthologous to ZmCIPK15 (see Figure 5.1a and 5.2a). Although the expression of CIPK4 and CIPK7 overlap, strength of the GUS signal is highly dependent on the gene and expression domain (meristem or maturation zone). CIPK4 is highly expressed in the maturation zone, but its expression is barely detectable in the meristem (see Figure 5.1). In contrast, CIPK7 expression is visibly lower than CIPK4 expression in the maturation zone, while it is highly expressed in the meristem (see Figure 5.2). The overlapping expression of CIPK4 and CIPK7 suggests that both function redundantly and form an ARBC. However, the enrichment of CIPK4 in the maturation zone and CIPK7 in the meristem suggest some degree of divergence between the kinases.

In addition to CIPK7 being expressed in the maturation zone and the meristem, CIPK7 is also predicted to localise to two different membranes (tonoplast and PM) (see Section 1.2.3.2). Because CIPK7 localisation *in planta* remains inconclusive (see Section 5.1), I generated the translational CIPK7-GFP fusion expressed under its native promoter (see Figure **??**). My transgenic lines show that CIPK7-GFP is present in the maturation zone and the meristem (see Figure 5.3), which is consistent with the expression domains determined by my GUS-promoter lines (see Figure 5.2). In the maturation zone, CIPK7-GFP appears to localise to the PM, whereas in the

meristem it accumulates in the cytosol and at undetermined intracellular domains (see Figure 5.3). It is important to note that CIPK7-GFP signal is significantly weaker in the meristem than in the maturation zone (see Figure 5.3). As CIPK7 is predicted to localise to the tonoplast, I hypothesise that the undetermined intracellular structures in the meristem are on the tonoplast membrane. The difference in the localisation of CIPK7 between both its expression domains suggests that the protein carries out different functions in each domain. A protein that has two different functions is known as a moon-lighting protein (Jeffery, 1999).

Of the two expression domains of CIPK4 and CIPK7, only the meristem is relevant in regards to a potential function in gravitropic signalling. In the meristem, CIPK7 expression is substantially higher than CIPK4, suggesting that CIPK7 could play the predominant role (over CIPK4) in gravitropic signalling. Translational CIPK7-GFP fusions suggest that, in the meristem, CIPK7 localises to intracellular structures, possibly the tonoplast (see Figure 5.3). The localisation of CIPK7 to the tonoplast, which is a major store of calcium in cells, would be consistent with a function downstream of the calcium elevations that accompany the auxin gradient during gravitropism (see Section 1.1.3.3.1) (Sanders *et al.*, 2002).



(a) Whole seedling

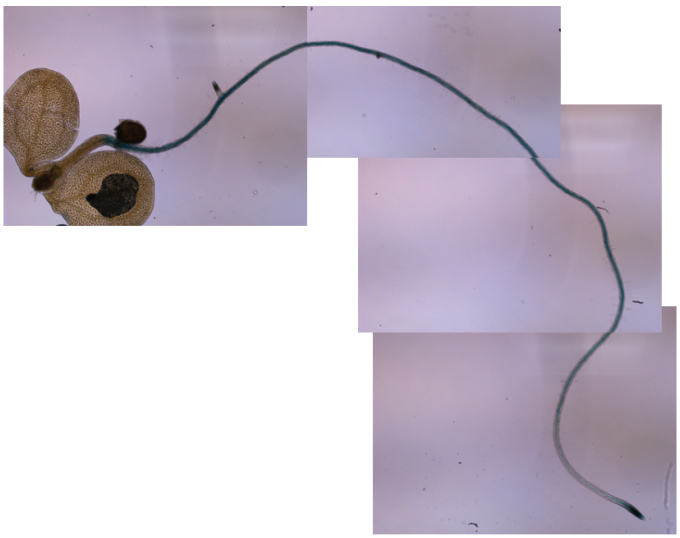


(b) Root tip

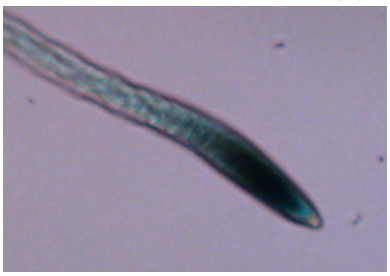


(c) Young lateral root

Figure 5.1: CIPK4::GUS expression analysis. (a) CIPK4 shows root-specific expression. (b) CIPK4 expression found in the basal meristem of root tip. (c) CIPK4 expression found in the maturation zone, but not in young lateral roots (see Section 2.2.5 for GUS-protocol and Supplemental Fig. **??** for construct details).



(a) Whole seedling

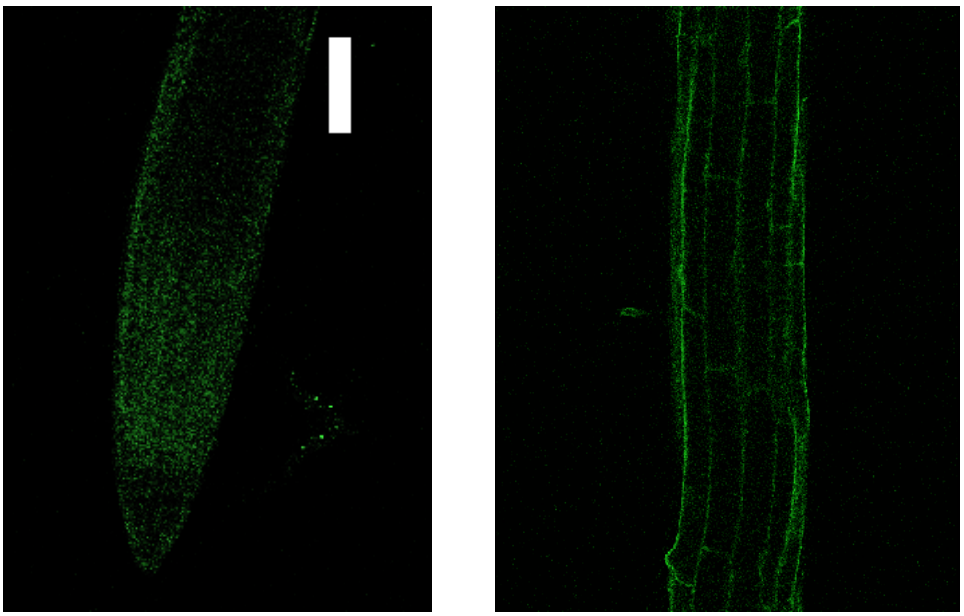


(b) Root tip



(c) Young lateral root

Figure 5.2: CIPK7::GUS expression analysis. (a) CIPK7 shows root-specific expression. (b) CIPK7 expression found in the basal meristem of root tip. (c) CIPK7 expression found in the maturation zone and in young lateral roots (see Section 2.2.5 for GUS-protocol and Supplemental Fig. **??** for construct details).



(a) Root tip (b) Maturation zone

Figure 5.3: CIPK7 appears to localise to (a) intracellular structures (potentially the tonoplast) in the basal meristem (root tip) and (b) to the PM in the maturation zone. CIPK7::CIPK7-GFP viewed under the confocal microscope (see Supplemental Fig. **??** for construct details). [bar = 75μm]

The auxin-inducibility and high expression of CIPK7 in the root meristem suggests that CIPK7 plays a predominant role, over CIPK4, in gravitropic signalling. However, KSselect predicts a biochemical redundancy between CIPK4 and CIPK7, in addition to GUS-promoter lines showing an overlapping expression in the root meristem. CIPK4 and CIPK7 may therefore function redundantly in gravitropic signalling. To test whether CIPK4 and/or CIPK7 function in gravitropic signalling, I generated single and double knock-out mutants. Homozygosity of these lines (see Supplemental Fig. S30) was confirmed using the Genotyping PCR protocol (see Methods 2.1). Additionally, RT-qPCR confirmed *cipk4-1* and *cipk7-1* as full knock-out mutants (see Methods 2.1)(see Supplemental Fig. S30). Based on the qPCR results, we generated the *cipk4-1 cipk7-1* double mutant, which is a full knock-out of both genes.

I used qGAP to analyse the gravitropic phenotypes of *cipk4-1*, *cipk7-1*, and *cipk4-1 cipk7-1*. Neither of the single nor the double mutant showed a significant reduction in root growth rate compared to wild type (see Supplemental Fig. S31). Similarly, neither of the single nor the double mutant showed a significant reduction in root gravitropic kinetics compared to wild type (see Figure 5.4, Table 5.1 and Supplemental Fig. S33). This suggests that neither CIPK4 nor CIPK7 functions in gravitropic signalling.

Table 5.1: Summary of Linear Mixed Model for CIPK gravitropic kinetics. See Figure 5.4 for corresponding plot

	Estimate	Std. Error	df	t value	Pr($>\|t\|$)
(Intercept)	93.12	1.25	92.37	74.21	0.00
LineCipk4-1	1.52	1.73	92.68	0.88	0.38
LineCipk4-2	1.79	1.67	92.79	1.08	0.28
LineCipk7-1	-2.17	1.67	93.02	-1.30	0.20
LineCipk7-2	-3.07	1.64	92.71	-1.88	0.06
Time	-0.13	0.01	91.20	-15.48	0.00
LineCipk4-1:Time	0.01	0.01	91.37	0.61	0.54
LineCipk4-2:Time	0.05	0.01	91.32	4.00	0.00
LineCipk7-1:Time	0.01	0.01	91.77	1.22	0.23
LineCipk7-2:Time	0.01	0.01	91.48	0.65	0.52

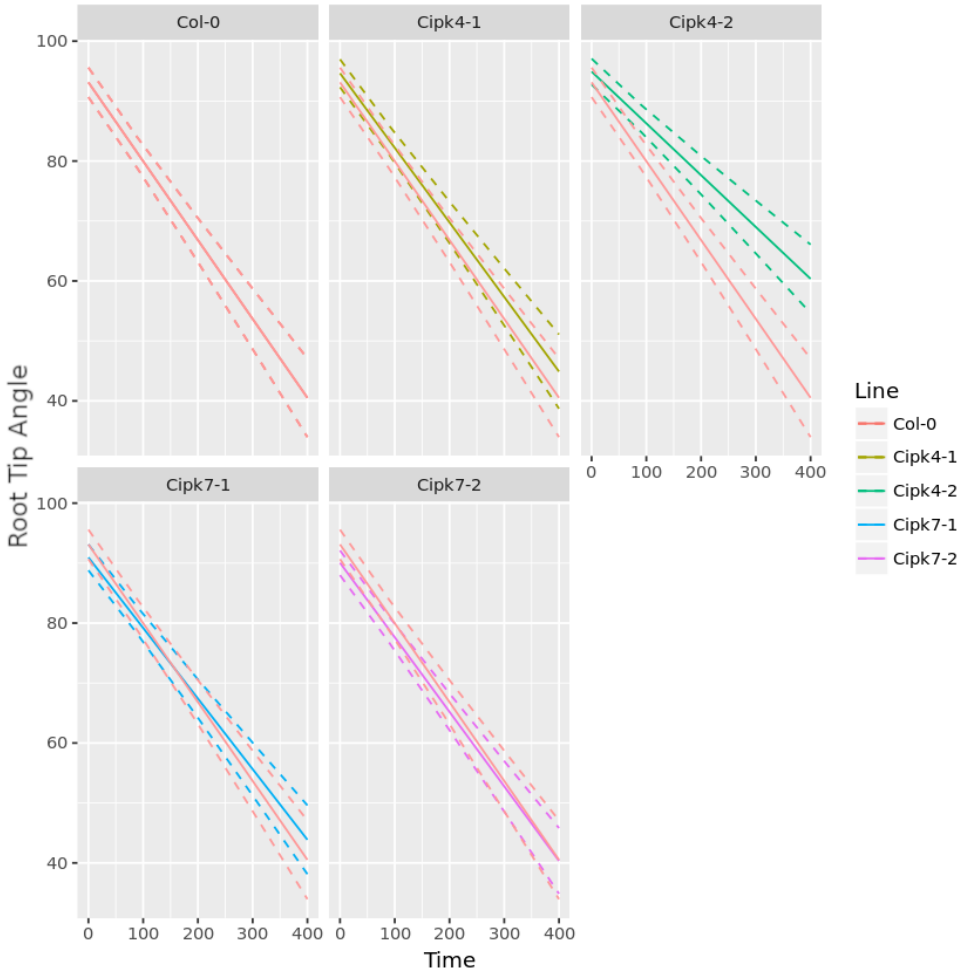


Figure 5.4: Gravitropic kinetics of Col-0 and *cipk4-1*, *cipk4-2*, *cipk7-1* and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. See Table 5.1 for corresponding summary of linear mixed model. Time is given in minutes and angle in degree.

5.3 CIPK Stress Expression Screen and Conditional Gravitropic Phenotyping

All characterised *Arabidopsis thaliana* CIPKs are known to function in stress response signalling (Li *et al.*, 2014). It is therefore possible that CIPK4 and CIPK7 signal downstream of a stress in the root meristem. The traditional approach to identifying which stress response pathway(s) a specific CIPK is functioning in is to determine which stress induces the expression of the CIPK of interest. I therefore generated expression heat-maps for all *Arabidopsis thaliana* CIPKs for multiple stress responses. The data I used to

generate these heat-maps was obtained by micro-array transcriptional profiling following stress treatments (Kilian *et al.*, 2007). Expression data was available for the following stress stimuli: cold, drought, genotoxicity, heat, osmotic, oxidative, salt, UVB, and wounding stress. CIPK4 and CIPK7 were only consistently up-regulated in response to cold stress, suggesting they function downstream of cold-stress (see Figure 5.5 and Supplemental Fig. S36 to S43). CIPK7 is more strongly induced by cold stress than CIPK4 (see Figure 5.5).

Cold shock treatment is known to delay the root gravitropic response and inhibit root elongation (Shibasaki *et al.*, 2009) (see Supplemental Fig. S44A,B). Cold stress increases an auxin response in the cortex and the epidermis of the root tip (Shibasaki *et al.*, 2009) (see Supplemental Fig. S44C). This is due to an inhibition of shoot-ward auxin transport as a result of the cold stress response (see Supplemental Fig. S44E). Down-regulation of shoot-ward auxin transport following cold shock treatment has been attributed to the inhibition of PIN2 intracellular cycling (see Supplemental Fig. S44D). The intracellular cycling of other proteins, however, remains unaffected, eliminating the possibility that cold treatment affects protein trafficking in general (Shibasaki *et al.*, 2009). I hypothesise that CIPK4 and CIPK7 function down-stream of cold stress to attenuate the gravitropic response by inhibiting PIN2 intracellular cycling. If CIPK4 and CIPK7 function in the cold-induced attenuation of gravitropism, the *cipk4 cipk7* double mutant should exhibit a cold-conditional gravitropic phenotype. A cold-conditional phenotype is a change in gravitropic kinetics that is only apparent after or during cold stimulus. In the next section I will utilise the flexibility of qGAP to investigate the cold-conditional gravitropic phenotype of *cipk4 cipk7* and its respective single mutants.

I hypothesise that CIPK4 and CIPK7 signal downstream of a cold stimulus to attenuate the gravitropic response. Cold stress is thought to af-

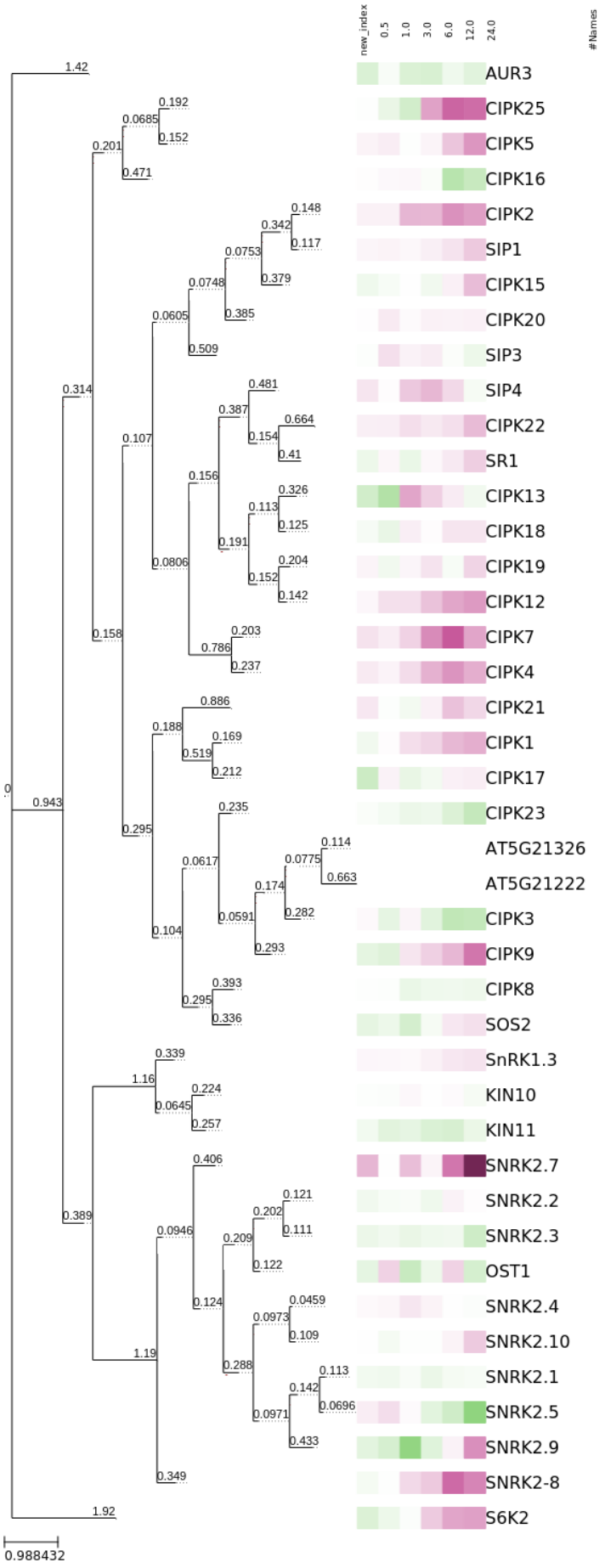


Figure 5.5: *(legend on next page)*

Figure 5.5: Unrooted phylogenetic tree (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their cold-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black numbers indicate branch length. Figure created using ete3 (see Section 2.4.5). Scale bar: Mean number of substitutions per site.

fect root elongation rate and gravitropic kinetics by inhibiting shoot-ward auxin transport (Shibasaki *et al.*, 2009). It is not know whether changes in gravitropic kinetics is a secondary effect of the reduction in root growth. Further, it is not know what signalling components act downstream of the cold stimulus to facilitate the inhibition of shoot-ward auxin transport. The expression of CIPK4 and CIPK7 in the meristem qualifies both genes as possible signalling components in the cold-induced attenuation of gravitropic kinetics. If one or both genes signal in this pathway, the single or double mutant should exhibit a cold-conditional gravitropic phenotype. A cold-conditional gravitropic phenotype is a phenotype that is only seen when a cold treatment is applied.

Two requirements had to be met by my gravitropic phenotyping approach (qGAP). The first requirement was the ability to detect conditional gravitropic phenotypes. The second requirement was the ability to exclude root growth as an impacting factor on gravitropic kinetics. The qGAP automatically analyses root growth in parallel to gravitropic kinetics. Therefore, differences in root growth rates are quantified and can confirm whether changes in gravitropic kinetics are primary or secondary. The ability of qGAP to analyse conditional gravitropic phenotypes is due to its extendibility. The qGAP owes its extendibility to the use of Linear Mixed Models (LMMs), which can accommodate additional variables (e.g. cold treatment) in the analysis of gravitropic kinetics (see Listing 5.1) (see Chapter 6).

Listing 5.1: Linear Mixed Model R Code (with Treatment)

```
mlm <- lmer(Growth~Line*Time*Treatment + (Time|Plate.ID:Root.ID),
```

```
↪ data=df)
```

We used qGAP to analyse the cold-conditional gravitropic phenotypes of *cipk4-1*, *cipk7-1*, and *cipk4-1 cipk7-1* (see Supplemental Fig. S45 and Supplemental Tab. S46). Consistent with previous qGAP results (see Section 5.2), neither of the single nor double mutant shows a significant change in root gravitropic kinetics compared to wild type under control conditions. We were unable to detect a cold-conditional gravitropic phenotype for any of the lines (see Supplemental Fig. S45 and Supplemental Tab. S46). Furthermore, we were unable to reproduce the previously observed cold-induced attenuation of the gravitropic response (Shibasaki *et al.*, 2009). The observed reduction in root growth in *cipk4-1* and *cipk4-1 cipk7-1* under control conditions was not consistent with previous qGAP results (see Figure 5.6, Supplemental Tab. S47 and Supplemental Fig. S31). It is therefore important to interpret the gravitropic kinetic results for *cipk4-1* and *cipk4-1 cipk7-1* with caution, as they might be affected by differences in root growth. In contrast, the wild type and *cipk7-1* lines show no differences in root growth under control conditions, making their gravitropic kinetic analyses conclusive (see Figure 5.6 and Supplemental Tab. S47).

Contrary to the gravitropic kinetics, which are not reduced by cold treatment, we found a cold-induced attenuation of root growth (see Figure 5.6 and Supplemental Tab. S47). We therefore hypothesise that, under our experimental conditions (see Methods 2.3.2), cold triggers a reduction in root growth that is decoupled from the gravitropic kinetics. However, neither wild type nor *cipk7-1*, which deliver reproducible root growth measurements under control conditions, exhibit a cold-conditional root growth phenotype (see Figure 5.6 and Supplemental Tab. S47).

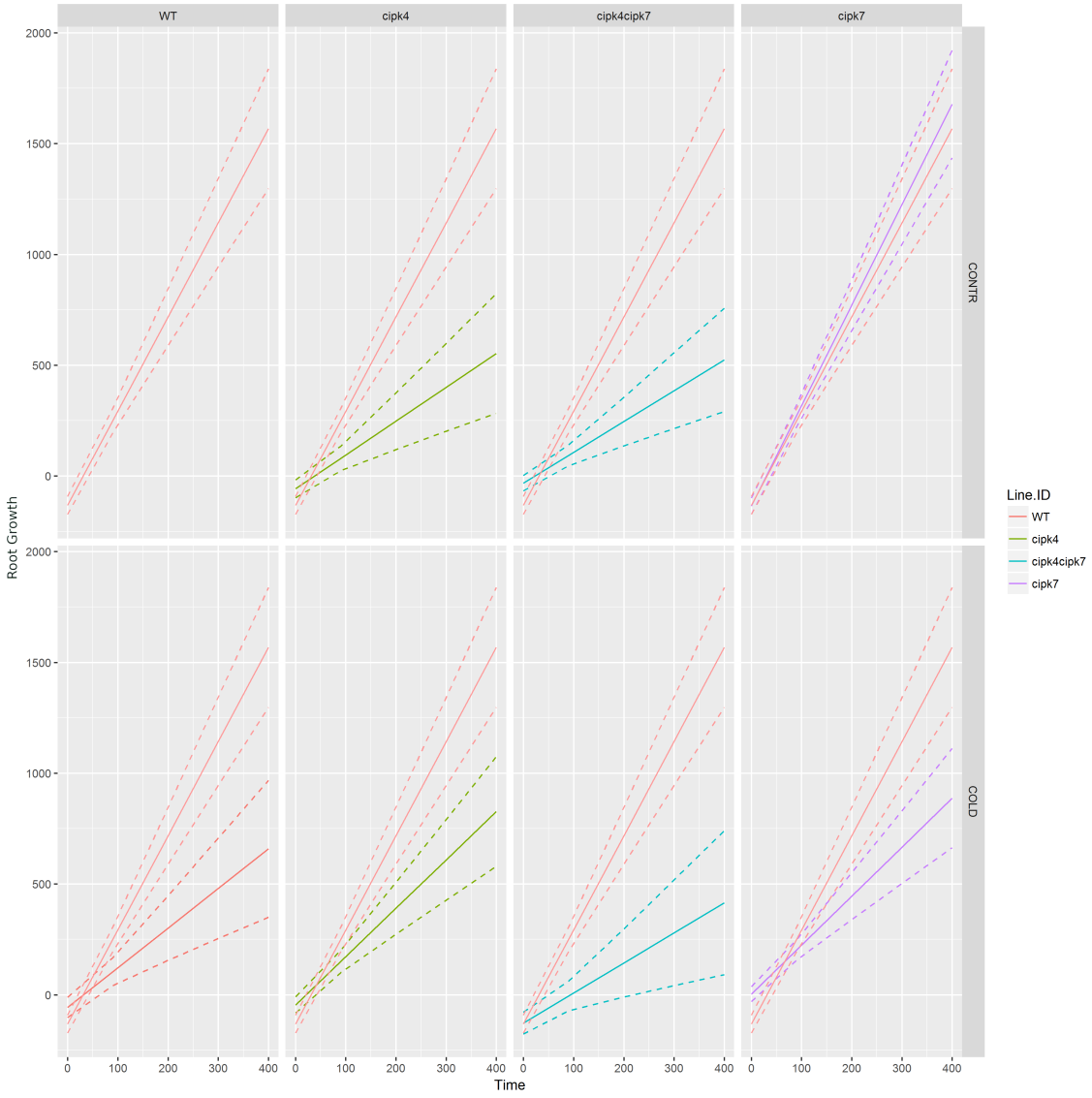


Figure 5.6: Root growth rates of *cipk4-1, cipk7-1, cipk4-1 cipk7-1* and isozygous wild-type control following control- (top) and cold-treatment (bottom) analysed using qGAP. Dashed lines show 95% confidence interval based on linear mixed model. Isozygous wild-type control root growth under control conditions is superimposed on every panel. Time is given in minutes and growth in pixel.

5.4 Conclusion

Zea mays ZmCIPK15 was identified as a gene associated with crown root angle (J Lynch, personal comm.). Our KSselect algorithm proposes *Arabidopsis thaliana* CIPK4 and CIPK7 as possible functional orthologs of ZmCIPK15 (see Chapter 3). Furthermore, KSselect predicts a functional redundancy between CIPK4 and CIPK7, which is further supported by their overlapping expression in the root basal meristem and the maturation zone. Of CIPK4 and CIPK7, only the latter is auxin inducible. I therefore hypothesise that CIPK4 and CIPK7 function redundantly in an ARBC, which allows for a fine tuned response to auxin (see Chapter 3). The predominant expression of CIPK7 over CIPK4 in the basal meristem suggests that CIPK7 plays a more crucial role in the hypothesised ARBC.

My results suggest that neither CIPK4 nor CIPK7 function in gravitropic signalling. The qGAP was not able to detect any differences in the gravitropic kinetics between the *cipk4-1*, *cipk7-1*, and *cipk4-1 cipk7-1* mutants. Further, the KSselect places CIPK4, CIPK7, and ZmCIPK15 in an ancient clade, making it unlikely that they share redundancy with other *Arabidopsis thaliana* kinases. I can therefore assume that the gravitropic phenotype of *cipk4-1 cipk7-1* is not masked by a redundant kinase.

As CIPK4 and CIPK7 do not function in gravitropic signalling, I further assessed whether they function in any stress response. The up-regulation in the expression of a CIPK kinase in response to a certain stress can indicate its involvement in the associated stress response pathway. My stress response expression heat-maps highlight a possible role of CIPK4 and CIPK7 in the cold response pathway. Further, these heat-maps, which show all *Arabidopsis thaliana* CIPKs, serve as a resource to investigate other CIPKs (see Figure 5.5 and Supplemental Figures S36 to S43).

The extendibility of qGAP allowed me to analyse the cold-conditional gravitropic phenotype of our CIPK mutants. I was unable to reproduce

the previously described cold-induced attenuation of gravitropic kinetics (Shibasaki *et al.*, 2009). However, qGAP detected a cold-induced reduction in root growth. The wild-type and *cipk7-1* lines showed a similar reduction in root growth in response to a cold stimulus. Results for the *cipk4-1* and *cipk4-1 cipk7-1* lines remain inconclusive, with both showing a reduction in root growth under control conditions. Further experiments are required to assess whether CIPK4 and CIPK7 function redundantly in the cold-induced reduction of root growth. The analysis of cold-conditional gravitropic kinetics clearly demonstrates the benefits of qGAPs ability to analyse root growth in parallel with gravitropic kinetics.

Chapter 6

The Quantitative Gravitropic Analysis Pipeline (qGAP) - A Novel and Adaptable Phenotyping Tool

6.1 Introduction

Multiple *Arabidopsis thaliana* mutants with impaired gravitropic responses have been characterised over past decades. The first gravitropic mutants discovered exhibit apparent phenotypes that are easily analysed using a t-test on root angles at an arbitrary time-point after gravi-stimulation. In recent years there has been a trend towards high-throughput time-course analyses of the gravitropic response (e.g. Wells *et al.*, 2012). The likely reason for this trend is that *Arabidopsis thaliana* single mutants with strong gravitropic phenotypes have been exhaustively discovered.

A major challenge faced by gravitropic research is that mutant phenotypes are likely to be subtle and/or conditional. Subtle gravitropic phenotypes can be masked by biological variation in gravitropic assays. High-

throughput time course analysis is used to increase the sensitivity of the analysis, however even for time course data the current approaches only allow for the statistical analysis of single time points (see Section 1.1.4). Our Centre for Integrative Plant Biology (CPIB) has developed RootTrace, a software for the semi-automatic measurements of root angle and root growth from time-course images (French *et al.*, 2009). CPIB has also build an imaging robot that automatically generates time-course images for up to 400 seedlings (Wells *et al.*, 2012). Combining both tools allows for a high-throughput time-course analysis of gravitropism.

Analysing the output of the RootTrace software manually is very time consuming. Three factors contribute to the manual processing of the RootTrace output being the bottleneck of the analysis. Firstly, the RootTrace software is prone to errors, none of which are explicitly annotated in the output files. These erroneous values must be manually removed from the dataset, which involves the visual inspection of trace images. Secondly, due to the experimental set-up (see Methods 2.3), the RootTrace output data for a specific line (and treatment) is split over multiple sub-directories. The output data therefore has to be manually combined from different sub-directories. Thirdly, selecting a subset of roots to be analysed based on initial angle or growth rate involves laborious manual inspection of the dataset.

To automate the analysis of the RootTrace output, I have written a Python module and Jupyter notebooks (see Figure 6.1). Combining the existing CPIB tool-set with my Python module and Jupyter notebooks creates a new high-throughput analysis pipeline, which I will subsequently refer to as the Quantitative Gravitropic Analysis Pipeline (qGAP). The qGAP not only automates the pre-processing of the RootTrace output, but also provides blueprints for the statistical analysis of the dataset (see Section 2.4.3). These blueprints allow researchers to minimise the MDD of the gravitropic kinetic analysis. Small MDDs allow for the discovery of subtle

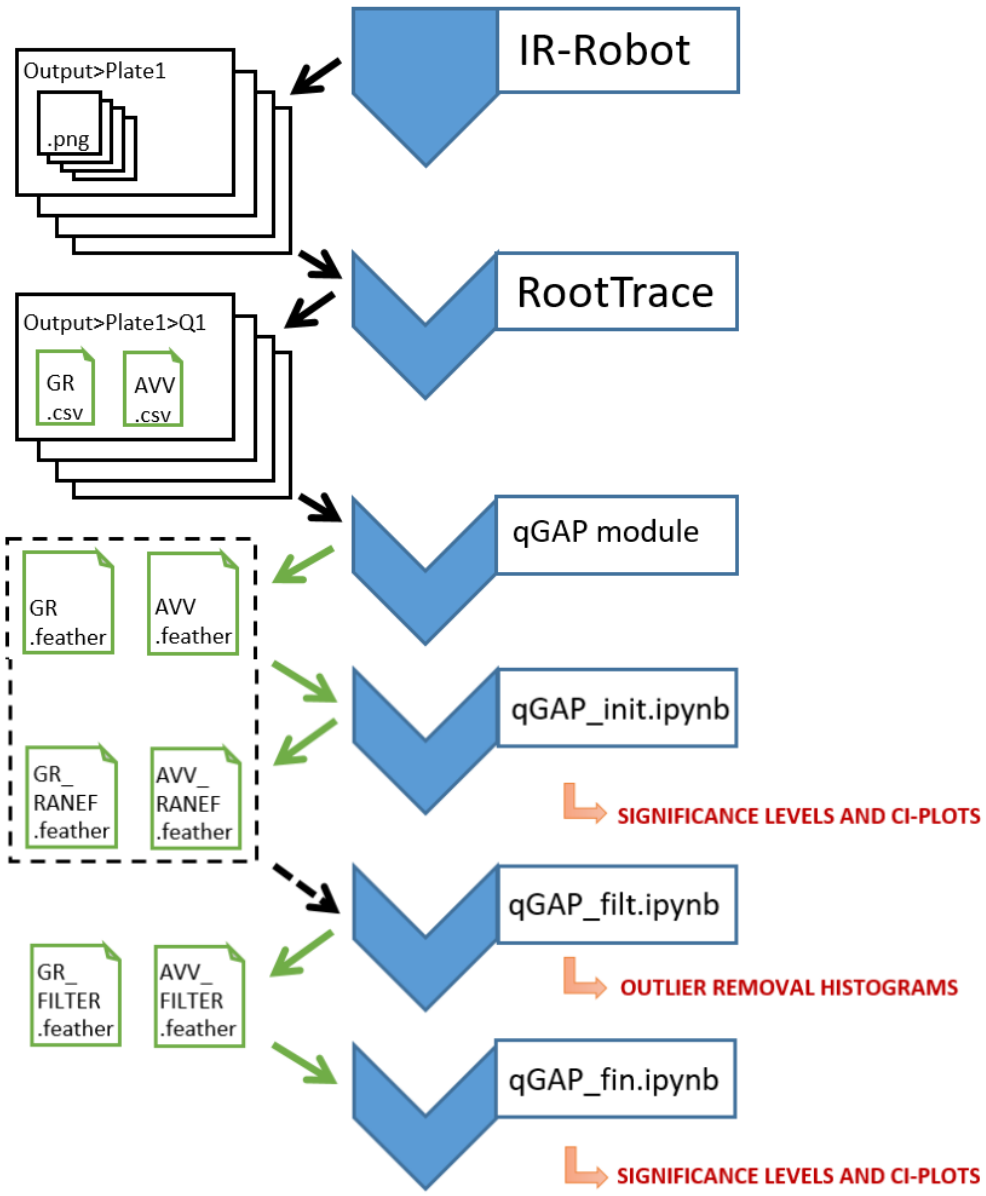


Figure 6.1: Schematic of the qGAP workflow. Middle (blue) indicates existing tools (Imaging Robot, RootTrace) plus the novel anaylsis pipeline (qGAP module and qGAP Jupyter notebooks). Left (black and green) indicates output files and directory structure. Right (red) indicates analysis output. IR-robot produces time-course images of different plates organised into folders (e.g. Plate1). We use RootTrace to analyse these images a quarter of a plate at a time, and save it to a new subfolder (e.g. Q1). The qGAP read module generates one data-frame for angle versus vertical (AVV) measurements and one for growth rate (GR) measurements. These data-frames are analysed using the qGAP Jupyter notebooks (see text for detail).

gravitropic phenotypes. The qGAP is also capable of assessing the effect of different treatments on gravitropic kinetics. This allows for the analysis of conditional mutants.

In this chapter I will demonstrate how qGAP allows for the discovery of subtle and conditional gravitropic mutant phenotypes. In the first section I will show how qGAP automatically detects erroneous data-points. In the second section I will justify my decision to use mixed models to analyse gravitropic kinetics. In the third section I will evaluate whether the selection of roots based on seedling age enhances the sensitivity of the mixed model. To my knowledge qGAP is the first example of mixed models, which have recently gained a lot of traction, being applied to gravitropism.

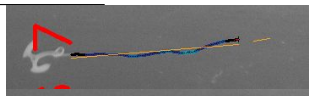
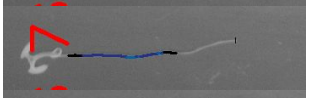
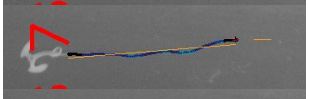
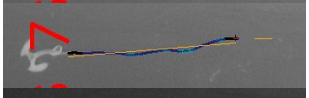
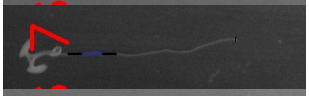
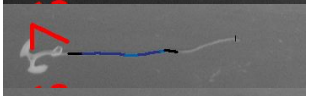
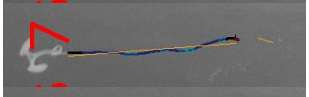
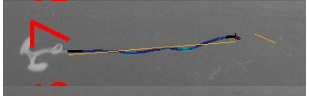
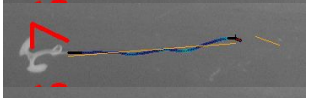
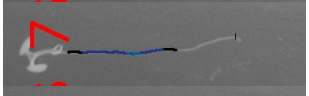
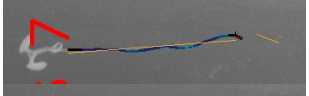
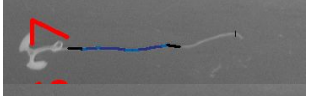
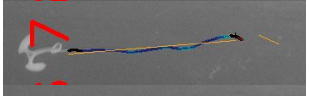
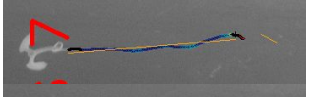
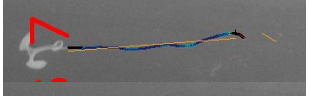
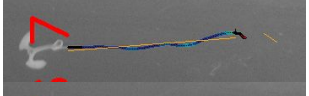
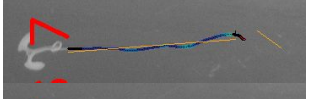
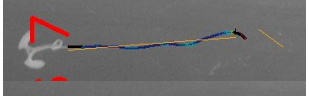
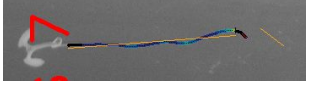
6.2 Reading the RootTrace Output

While the RootTrace software allows for the semi-automatic measurement of root angle and root growth, the analysis of the output data still requires time-consuming manual inspection and annotation. To speed up this analysis, I have generated the qGAP Python module to automate the inspection and annotation of the RootTrace output (see Figure 6.1).

The nature of our experimental set-up (multiple split plates) means that the RootTrace data for one line (and treatment) is split over multiple sub-directories. The qGAP module offers the qGAP_ read function, which automates the annotation and collection of the RootTrace output. This function takes the RootTrace output directory and a layout file (.csv) as parameters and generates one annotated data-frame (Pandas long). The qGAP_read function generates data-frames for root growth and root angle simultaneously. It uses a functional programming structure to account for a possible absence of the angle data-frame (data-frames wrapped in maybe monad)(see Section 2.4.3).

The qGAP module further offers the qGAP_filter function, which au-

Table 6.1: Example of unfiltered RootTrace measurements (columns 2 and 3), dataset after automatic filters applied by qgap_read (columns 4 and 5) and corresponding RootTrace trace images (column 6).

	RootTrace		qGAP		
Time	Growth	Angle	Growth	Angle	Image
0	203.599	7.0	0.0	97.0	
20	130.461	999.0	NaN	NaN	
40	206.613	3.0	3.014	93.0	
60	206.772	1.0	3.173	91.0	
80	57.545	999.0	NaN	NaN	
100	131.556	999.0	NaN	NaN	
120	208.529	-16.0	4.930	74.0	
140	208.874	-22.0	5.275	68.0	
160	210.020	-19.0	6.421	71.0	
180	130.966	999.0	NaN	NaN	
200	211.493	-22.0	7.894	68.0	
220	135.215	999.0	NaN	NaN	
240	213.134	-23.0	9.535	67.0	
260	214.997	-29.0	11.398	61.0	
280	214.764	-32.0	11.165	58.0	
300	217.133	-33.0	13.534	57.0	
320	216.952	-35.0	13.353	55.0	
340	217.778	-35.0	14.179	55.0	
360	218.858	-36.0	15.259	54.0	

tomates the detection of RootTrace errors. Two types of errors lead to erroneous values in the RootTrace output. Firstly, root tracing can stop before it reaches the root tip. I will refer to this type of error as a trace-loss error. Trace-loss errors can be initial (at the first time-point) or intermittent (at later time-points) (see Table 6.1). Secondly, errors can occur in the angle measurement. I will refer to this type of error as an angle measurement error. Angle measurement errors can result from failed measurements, or measurements taken from an incorrect baseline. The qGAP_filter function labels erroneous values, resulting from trace-loss errors or angle measurement errors, as missing values (Not a Number / NaN).

The qGAP_filter function pre-processes the angle and growth data-frames in two additional ways. Firstly, roots with initial trace-loss are excluded, following CPIB standard protocol. Secondly, roots with a number of NaNs exceeding a set threshold are excluded, due to their negative impact on subsequent statistical analyses (see Section 2.4.3).

Calling both the qGAP_read and the qGAP_filter function from a Jupyter notebook environment has three primary advantages. Firstly , Jupyter notebooks serve as a record of the parameters given in any run of the analysis pipeline. Secondly, Jupyter notebooks can be hosted on a cloud server and made available to collaborators via Jupyter Hub (see Supplemental Code S2.1.1). Collaborators can interactively modify, run and save these notebooks in most web browsers. Thirdly, an R kernel exists for Jupyter notebooks, meaning that the statistical analysis (see Section 6.3) can be run in the same environment.

Although it is possible to run both Python and R in the same notebook, running the entire qGAP analysis in one notebook would result in overbearing notebooks. I therefore split the analysis into four notebooks, using the feather format to share data between these notebooks.

The final step of RootTrace output pre-processing is the conversion of

data-frames (generated by calling qGAP_read and qGAP_filter) into the long format (see Supplemental Code S2.1.1). The converted data-frames are then saved in "feather" format, which can be read by R for statistical analysis. In the next section I will justify my statistical model of choice.

6.3 Statistical Analysis of RootTrace Output

The qGAP module automates the pre-processing of the RootTrace output. This reduces the time it takes to convert the RootTrace output to an R data-frame from hours to just a few minutes. In addition to the qGAP module, qGAP provides a blueprint for the statistical analysis of a typical qGAP dataset, in the form of Jupyter notebooks.

As a qGAP dataset consists of repeated measures of individual roots, only three statistical tests are suitable. Firstly, a single time-point in the dataset can be analysed using ANalyis Of VAriance (ANOVA). However, this approach ignores the majority of the dataset (94% in my set-up) (see Table 6.2). Secondly, repeated measure ANalyis of COVAriance (rm-ANCOVA) can be used to analyse the majority of data-points. However, all roots with a NaN value must be excluded. In a typical qGAP dataset this will result in only a small reduction in the number of data-points. However, power analysis of rm-ANCOVA has not yet been implemented in any R package. Power analysis is required to calculate the MDD, which is a measure of the sensitivity of a statistical test (see Table 6.2). Thirdly, the entire dataset can be analysed using mixed models. Mixed models have three primary advantages over rm-ANCOVA: they allow for the analysis of additional explanatory variables (fixed effects) and their interactions (see Chapter 5); the relation between angle/growth and time does not have to be linear; and the data does not have to be normally distributed (generalised

models exist) (see Table 6.2).

Table 6.2: Comparison of statistical test suitable for analysis of gravitropic kinetic data

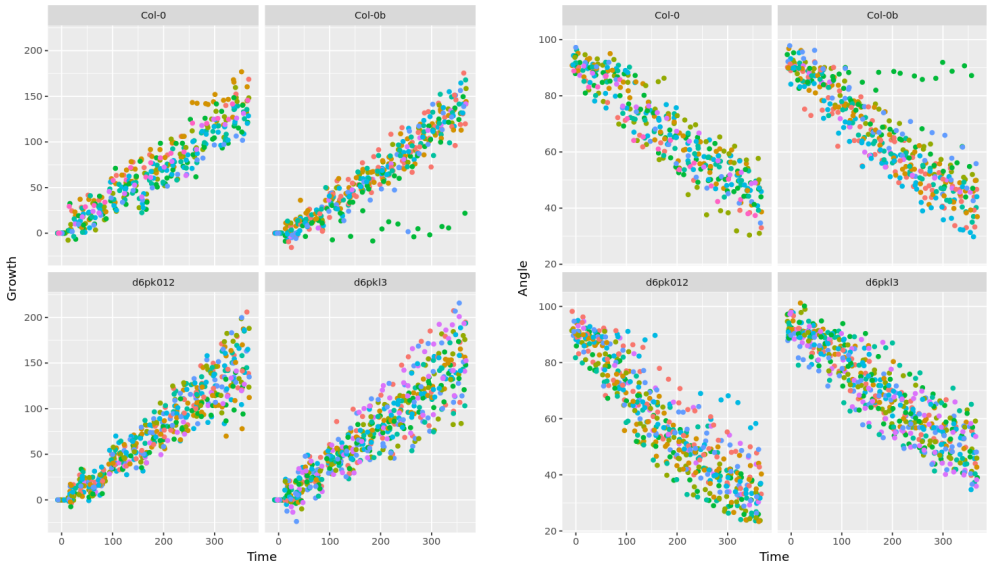
	ANOVA on Time-Point	Repeated-Measure ANCOVA	Mixed Model
All data analysed	**No**	**Yes***	**Yes**
Power analysis library	**Yes**	**No**	**Yes**
Flexibility			
multiple explanatory variables	Yes	No	Yes
non-normal tests	No	No	Yes
non-linear relationship	No	No	Yes

*except roots with missing data

Because of the multiple advantages of mixed models over ANOVA and rm-ANCOVA, mixed models are the statistical test of choice for qGAP. I will subsequently use an example dataset to demonstrate the use of qGAP. The example dataset was obtained via a gravitropic assay of four lines, two Columbia wild-type lines, the *d6pkl3* single mutant and the *d6pkl0 d6pkl1 d6pkl2* triple mutant. Inspection of the resulting scatter plots suggests that both angle and growth change linearly with time (see Figure 6.2). I can therefore apply Linear Mixed Models (LMMs) in this qGAP analysis of gravitropic kinetics and root growth. I am confident that LMMs will be suitable for the majority of qGAP datasets. To conduct LMM analyses, qGAP utilises the lmer function of the lmerTest package (R language) (Kuznetsova *et al.*, 2017), which includes automatic post-hoc analysis. Angle/Growth and Line were chosen as fixed effects, while Plate.ID and Root.ID were chosen as random effect (see Chapter 4). The example dataset further exemplifies that a qGAP dataset fulfils the requirements for LMM analysis (see Supplemental Figures S48 to S50).

6.4 Removal of Outliers

Past gravitropic assays have shown that initial root angle influences gravitropic kinetics (CPIB, unpublished). Seedlings with a root angle deviating



(a) Root growth rate (b) Gravitropic kinetics

Figure 6.2: Root growth rate and gravitropic kinetic plots suggest that both change linearly with time for the observed time-frame.

from 90 degrees experience a different gravitropic stimulus, compared to those at roughly 90 degrees. It is therefore standard procedure to select seedling based on their initial root angle (80-100 degrees). The qGAP_filter function (see Section 6.2) implements the removal of roots with an initial angle deviating from 90 degrees at a given threshold.

Another factor influencing gravitropic kinetics is seedling age (Brooks *et al.*, 2010). Non-synchronous germination results in seedling age variation within a qGAP dataset. Dealing with this variation manually requires laborious inspection of the plates at the time of germination. To implement the automatic selection of seedlings within a certain age range, qGAP uses root growth rate as a proxy for seedling age. Root growth rate positively correlates with seedling age (Brooks *et al.*, 2010). Root growth rate can be estimated using the Best Linear Unbiased Predictions (BLUPs) of random effects taken from the root growth mixed model (see Section 2.4.3).

To select for seedlings based on age, qGAP offers the qGAP-filter Jupyter notebook. This notebook implements the removal of seedlings with a root growth rate (BLUP) outside of a certain threshold surrounding the mean

growth rate. Depending on the parameter given in the qGAP filter notebook, seedling removal based on growth rate is either in the form of outlier removal (3*standard deviation) or dataset sub-setting (1.5*standard deviation) (see Figure 6.3).

The removal of outliers from biological data is often met with criticism. Critics argue that an outlier, if not due to a measurement error, is part of the phenotype and should therefore not be removed. However, outliers can have a negative impact on statistical analyses, e.g. outliers can violate the requirements of a statistical test. When removing outliers it is considered good practice to report the statistical analysis of the full dataset in addition to the filtered dataset. The qGAP therefore implements a two-step analysis, an initial analysis of the full dataset (qGAP_init.ipynb) and a further analysis of the filtered dataset (qGAP_fin.ipynb).

The sub-setting of data based on root growth may also be met with scepticism due to two main objections. Firstly, the variation in root growth might be part of the phenotype of a line. Secondly, root growth rate might not be an accurate proxy for seedling age. Despite both objections, not selecting roots based on seedling age could skew the analysis, as seedlings of different ages exhibit entirely different gravitropic kinetics.

Before removing outliers or sub-setting the dataset, it is important to clarify the distribution of the growth rates (slope BLUP of root growth mixed model) within each line. I therefore generated growth rate histograms for each line (see Figure 6.3). These histograms show that the Col-0b data contains an outlier, and that the growth rates of *d6pkl3* and *d6pk0 d6pkl1 d6pkl2* exhibit increased variability compared to Col-0. The two thresholds for root growth-based selection are 3*standard deviation (outlier removal) and 1.5*standard deviation (sub-setting), which are indicated on the histograms (see Figure 6.3).

I used simulation-based power analysis (simr package) to generate power

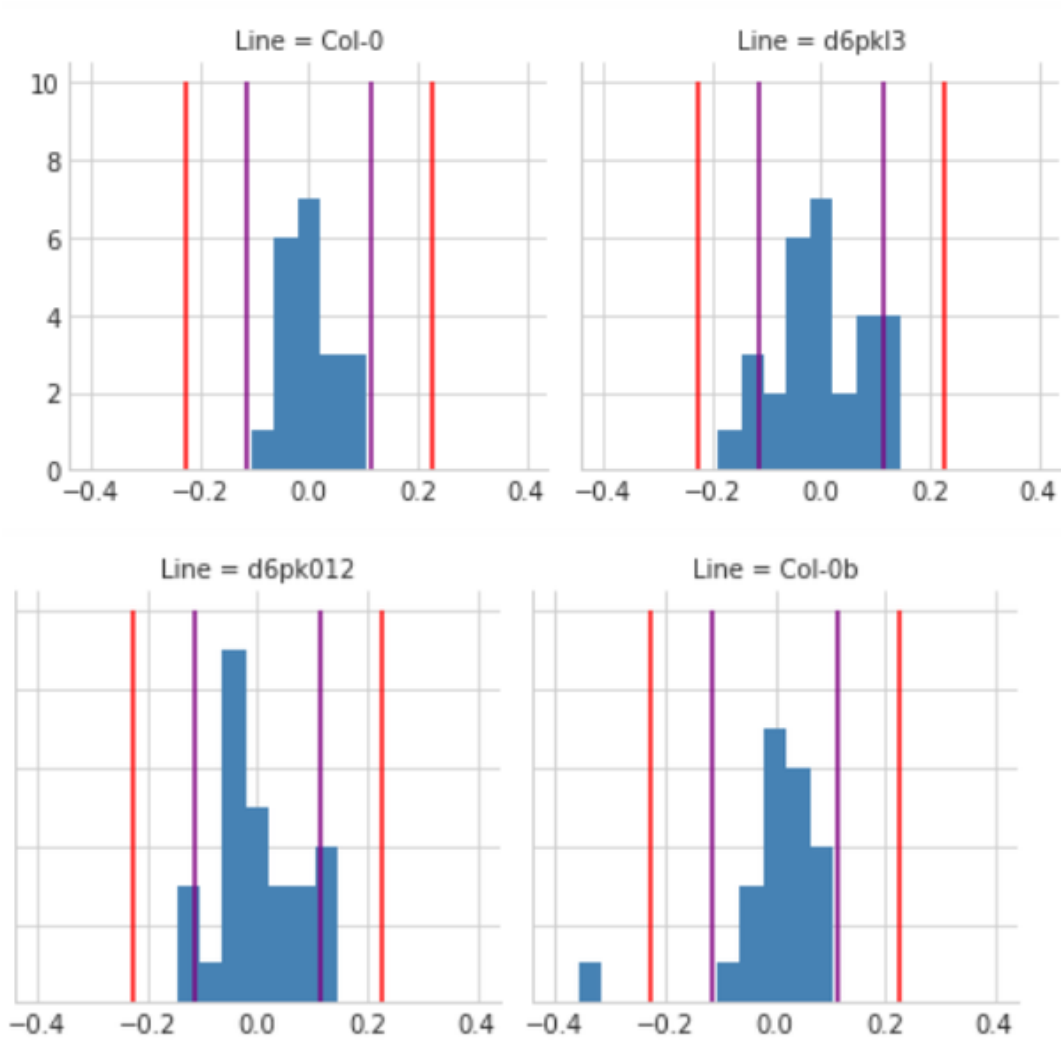
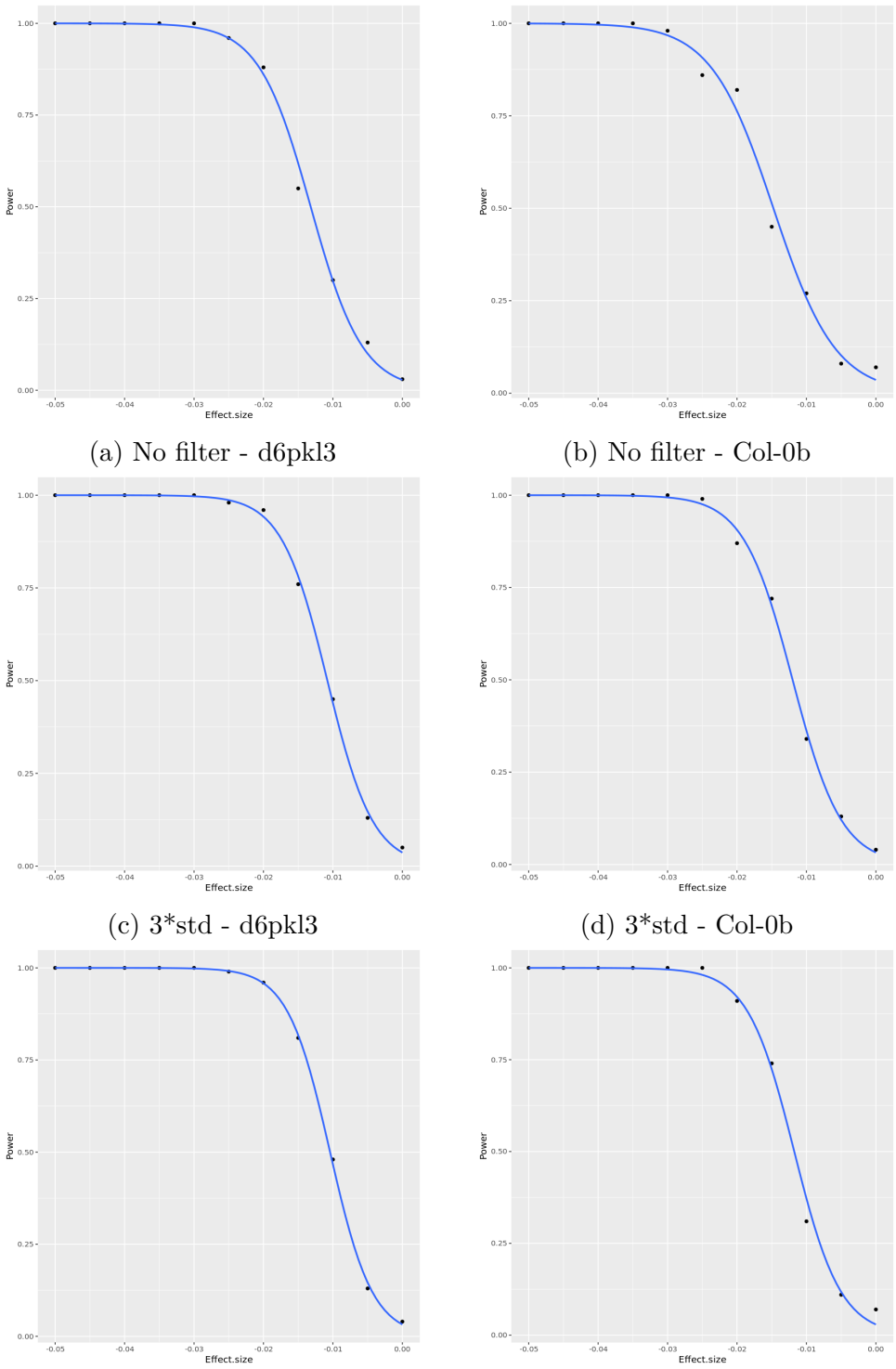


Figure 6.3: Histograms of differences in slope from grand mean as estimated by BLUP (qGAP example dataset). Lines indicate 3*standard deviation (red) and 1.5*standard deviation (purple) thresholds, which are given as parameters in qGAP_filt.ipynb

curves for the whole dataset, the outlier removal dataset and the sub-setting dataset (see Figure 6.4). Minimal detectable difference (MDD) is often defined as the effect size at 80% power of a statistical test. Outlier removal decreased the MDD of the mixed model, when compared to the mixed model of the whole dataset (see Effect.size at 80% Power in Figure 6.4). Sub-setting results in a further reduction in the MDD in *d6pkl3*, when compared to the corresponding outlier removal mixed model (see Effect.size at 80% Power in Figure 6.4). However, sub-setting did not change the MDD in Col-0b, when compared to the corresponding outlier removal mixed model

(see Effect.size at 80% Power in Figure 6.4). This demonstrates that outlier removal and dataset sub-setting increase the sensitivity (MDD) of qGAPs gravitropic kinetic mixed models.



(a) No filter - d6pkl3

(b) No filter - Col-0b

(c) 3*std - d6pkl3

(d) 3*std - Col-0b

(e) 1.5*std - d6pkl3

(f) 1.5*std - Col-0b

Figure 6.4: Power curves of the mixed models based on (a, b) the full dataset, (c, d) the 3*standard deviation filtered dataset and (e, f) the 1.5*standard deviation filtered dataset. Power was estimated using simulations, with the Col-0 model as the baseline. The Effect.size at 80% Power is often defined as the Minimal Detectable Difference (MDD). Power analysis simulations were ran using simr.

6.5 Conclusion

The qGAP Python module and Jupyter notebooks provide the necessary tools to make qGAP a high-throughput analysis pipeline. The pipeline addresses the need for a flexible and highly sensitive analysis pipeline for gravitropic kinetic data. The qGAP was designed to allow for modifications and enhancements, ensuring that it is adaptable and not restricted to the CPIB toolset.

Currently, the qGAP Python module and Jupyter notebooks are specific to the RootTrace software. Although RootTrace is a well established tool for the measurement of angle and growth (French *et al.*, 2009), it has limitations. One limitation is that the "tip points to fit" parameter of RootTrace determines the noise level and accuracy of the resulting measurements (Müller *et al.*, 2018). Depending on the parameter given, a reduction in noise will reduce the accuracy, and vice versa. To enable researchers to adapt qGAP to different image analysis softwares, I have implemented the qGAP Python module following the functional programming paradigm. When using the functional programming paradigm, the logic behind qGAPs composition is transparent. This ensured qGAP is accessible for the modification of individual functions.

The qGAP uses root growth as a proxy for seedling age. Root selection based on seedling age is accomplished by removing root growth outliers. This requires a similar distribution of root growth rates in all the lines being analysed, meaning qGAP is unable to select based on seedling age if root growth rates do not overlap. Scholler *et al.* (2018) address this problem by using growth rate normalization. In future versions of qGAP I plan to implement a gravitropic analysis that accounts for differences in root growth rates.

Users of qGAP can choose not to apply the growth rate filter, for example if there is little variation in the growth rate (e.g. *pgm1* dataset, Section 2.4).

The MDD, which is determined by the power analysis, serves a measurement of sensitivity. This allows researchers to design further experiments with the aim of minimising the MDD and therefore maximising the sensitivity.

Jupyter notebooks, which are popular in other scientific disciplines, are rarely used in plant science. While designing qGAP I saw distinct advantages of using Jupyter notebooks (see Section 6.1). My hope is that qGAP will pave the way for the adoption of Jupyter notebooks in CPIB, and potentially plant sciences as a whole.

Chapter 7

Conclusion

Research into root gravitropism has been ongoing for over two centuries (Knight, 1806) and although this field can be viewed as purely academic, its findings can have a significant impact on agriculture. Ongoing gravitropic research continues to uncover the complexity of the underlying mechanisms involved. This thesis not only furthers our understanding of root gravitropism, but also challenges previous assumptions and conventions. As noted in my literature review, while we posses a substantial understanding of the slower mechanisms that generate the auxin gradient (PIN transcytosis), there remains a gap in our knowledge regarding the faster mechanisms (direct activation of PIN proteins)(see Chapter 1). It is likely that changes in the direct activation of PIN proteins would only elicit subtle defects in the gravitropic response due to redundancy with PIN transcytosis. Kinases are capable of rapid signal amplification, making them good candidates for the direct activation of PIN proteins. The aim of this thesis was to identify and assess candidate kinases for a role in gravitropic signalling. This primary aim was further divided into four sub-aims:

- **Aim 1:** To generate and employ a screening tool for the identification of kinases that are likely to function in gravitropism (see Chapter 3).
- **Aim 2:** To generate a tool capable of detecting subtle and/or condi-

tional gravitropic phenotypes (see Chapter 6).

- **Aim 3:** To determine whether D6PKL3 is involved in gravitropic signalling (see Chapter 4).
- **Aim 4:** To determine whether CIPK4 and/or CIPK7 are involved in gravitropic signalling (see Chapter 5).

To screen for kinases with a possible function in gravitropism ('Aim 1'), I needed to investigate the relationship between kinases in regards to their target specificity. This is conventionally achieved by examining kinase domain phylogenies. However, Creixell et al. (2015) created a more accurate methodology, which I applied to plant kinases for the first time by using my KSselect algorithm. The KSselect results combined with auxin expression profiles lead to the identification of two candidate kinases for a role in gravitropic signalling, D6PKL3 and CIPK4/7. D6PKL3 was identified as a candidate kinase based on its relationship to D6PK, a kinase known to function in auxin transport. CIPK4 and CIPK7 were identified as likely orthologs of maize CIPK15 (ZmCIPK15). ZmCIPK15 is associated with changes in crown root angle, which is a phenotype linked to gravitropism (J. Lynch, personal comm.). For both D6PKL3 and CIPK4/7, auxin inducibility was used as a selection criterion. KSselect was then used to predict the functional relationship between kinases by determining similarities in phospho-site preference. Subsequent analysis revealed that neither D6PKL3 nor CIPK4/7 are likely to function non-redundantly in gravitropism (see Chapter 4/5).

An improvement to my screening method would be including additional selection criteria such as high expression levels in columella cells, the root meristem and/or the elongation zone. My KSselect algorithm extends on the KINspect algorithm (Creixell *et al.*, 2015) to make it applicable to plant kinases. The KINspect algorithm accurately predicts the functional relationship of human kinases (Creixell *et al.*, 2015). There are no results

indicating that the KINspect methodology cannot be applied to plants. However, it is important to confirm the KSselect results using molecular biological work (e.g. phosphorylation assays). Furthermore, neither KINspect nor KSselect implement a protocol for outgroup selection, meaning DoS dendrograms cannot be rooted. Until such a protocol is established, the interpretation of DoS dendrograms is limited to the functional relationship between kinases. An outgroup selection protocol would enable us to draw conclusions about the evolutionary history of kinase target specificity.

Preliminary analysis of the D6PKL3 and CIPK4/7 mutants did not uncover any gravitropic defects. However, it is possible that subtle defects were not detected by this preliminary analysis. I therefore created the highly sensitive Quantitative Gravitropic Analysis Pipeline (qGAP) ('Aim 2'). Compared to previously available gravitropic analysis approaches, qGAP has three major advantages. Firstly, its ability to analyse all data points in one statistical test improves the sensitivity of the analysis. A measurement of this sensitivity is provided by the Minimal Detectable Difference (MDD) (see Chapter 6). Secondly, qGAP allows us to analyse the effect a treatment has on the gravitropic response of mutants (conditional phenotypes) (see Chapter 5). Thirdly, qGAP reduces the time it takes to analyse gravitropic time-series data from hours to minutes.

The adoption of qGAP as a standard protocol could be hindered by its requirement for the programming languages Python and R. Biologists are rarely proficient in these programming languages. In addition, setting up the required programming environments could present a challenge, which is why I demonstrate qGAP in Jupyter notebooks (see Section 6.1). Jupyter notebooks can be accessed by most web browsers, thus avoiding the need to set up programming environments. In future versions of qGAP I will include Jupyter dashboards, which will make qGAP usable without the knowledge of programming languages. In this thesis I used qGAP to analyse D6PKL3

and CIPK4/7 mutants.

D6PKL3 was discovered as a candidate kinase by my screen and in Chapter 4 I attempt to elucidate its role in gravitropism ('Aim 3'). It's possible that D6PKL3 is involved in the molecular mechanisms facilitating direct activation of PIN proteins in columella cells (see Introduction).Transcriptional (GUS) and translational (GFP/YFP) fusions were used to investigate D6PKL3 expression and targeting. My results confirm the previously described enrichment of D6PKL3 in columella cells (Zourelidou *et al.*, 2009). Furthermore, auxin treatment revealed an additional auxin inducible expression domain in the basal meristem. The translational fusions suggest that D6PKL3 is mainly bound to intracellular stores in columella cells, but localises to the rootward PM in the basal meristem. qGAP results for the *d6pkl3* knockout mutant reveal no detectable gravitropic defect, meaning I can exclude a non-redundant function of D6PKL3 in gravitropic signalling. It is therefore unlikely that D6PKL3 is functioning in the direct activation of PINs during early gravitropic signalling.

Further characterisation of D6PKL3 should focus on utilising qGAP to phenotype different double and triple mutant combinations of *d6pkl3*. The KSselect results suggest that, in terms of target specificity (DoS dendrograms), there are uncharacterised kinases that are more closely related to D6PKL3 than the other D6PK-clade kinases. These uncharacterised kinases should arguably be prioritised over the D6PK-clade kinases when phenotyping the double and triple mutant combinations of *d6pkl3*.

The proximity of D6PKL3 to these uncharacterised kinases in the DoS-dendrograms casts doubt on whether D6PKL3 is capable of phosphorylating PIN proteins. To investigate this I created PRBPviz, which predicts the phospho-site preference of a kinase based on kinase domain characteristics. In this thesis PRBPviz predicted the existence of general phospho-sites (MC-group) on PIN proteins, which are targeted equally by all D6PKs and

PIDs. My hypothesis is that redundancy at the phospho-site level (MC-group, PRBPviz) takes precedence over D6PKL3 divergence at protein level (KSselect). The existence of general phospho-sites (MC-group) on PIN proteins impacts the current model of polar auxin transport. Consistent with the traditional linear model of kinase function, PIDs and D6PKs are considered non-redundant. However, the presence of MC-group phospho-sites challenges this traditional model, as it allows for cross talk between D6PKs and PIDs.

Predictions made by PRBPviz lay substantial groundwork for the further characterisation of PIN phospho-sites. In future work, it is important that predictions made by PRBPviz are confirmed using biochemical assays (e.g. phosphorylation assays). Once confirmed, PRBPviz can be used to discover candidate target phospho-sites for a variety of kinases. One example of this would be the discovery of an MC-group phospho-site, which would indicate phosphorylation by AGC kinases.

CIPK4 and CIPK7 were identified as candidate kinases (KSselect). In Chapter 5 I attempt to elucidate the role of these kinases in gravitropism ('Aim 4'). In Chapter 3 I discovered that CIPK7 is auxin inducible, but CIPK4 is not. Interestingly, CIPK7 exhibits a similar temporal profile of auxin-induction to D6PKL3 (see Chapter 3). It is therefore possible that CIPK7 and D6PKL3 are regulated by the same secondary auxin-response transcription factors. To further characterise CIPK4/7 I used transcriptional (GUS) and translational (GFP/YFP) fusions to investigate CIPK4/7 expression and targeting. The qGAP results for CIPK4/7 allow me to exclude the possibility that these genes are functioning non-redundantly in gravitropism. Both CIPK4 and CIPK7 are placed in an isolated clade by KSselect, suggesting their function is not masked by redundant genes. It is therefore unlikely that the lack of a phenotype in the qGAP results is due to redundancy. This result demonstrates the resourcefulness of KSselect in

interpreting qGAP results.

Following the conclusion that neither CIPK4 nor CIPK7 function in gravitropic signalling, I investigated a possible role of these kinases in a variety of stress responses. I found that CIPK4/7-expression is exclusively up-regulated in response to cold stress, suggesting they function in the cold stress response. The extendibility of qGAP allowed me to determine that neither the *cipk4-1*, the *cipk7-1*, nor the *cipk4-1 cipk7-1* exhibit a cold conditional gravitropic phenotype.

As CIPK4 and CIPK7 do not function in gravitropic signalling, future work should focus on lateral and shoot-born roots in *Arabidopsis thaliana*. *ZmCIPK15* exhibits a defect in the angle of crown roots (J. Lynch, *personal comm.*), which are shoot-born. CIPK7, unlike CIPK4, is expressed in young lateral roots(see Chapter 5) and future work should assess whether the *cipk7-1* mutant exhibits a defect in the Gravitropic Set-point Angle (GSA) of lateral roots. In addition, CIPK7 expression in shoot-born roots should be analysed. The observed attenuation of gravitropic kinetics in response to cold (Shibasaki *et al.*, 2009) was not reproduced in my results. The cold stress response is highly influenced by light conditions (Shibasaki *et al.*, 2009), meaning future research must provide a detailed record of the light conditions during an experiment.

To address the four aims of this thesis I employed a three-step process. The first step was screening for candidate kinases using my KSselect algorithm combined with auxin expression profiles ('Aim 1'). For the second step I used bioinformatic and molecular biological methods to further characterise each candidate kinase ('Aim 3/4'). The third step was the gravitropic phenotyping of candidate kinase mutants using my highly sensitive and extendible Quantitative Gravitropic Analysis Pipeline(qGAP)('Aim 3/4'). The qGAP provides crucial features that were missing in the available analysis methods ('Aim 2'). KSselect not only provides a novel approach for

identifying candidate kinases, but also helped me interpret the qGAP results. For example, KSselect allowed me to exclude the possibility that the *cipk4-1 cipk7-1* phenotype is masked by redundant genes. The qGAP allowed me to analyse my D6PK and CIPK4/7 mutants with an unprecedented degree of sensitivity. Unlike existing phenotyping approaches, qGAP is able to discern between no gravitropic defect and a gravitropic defect that is too subtle to be detected with high certainty. The key value of this thesis is the adaptability of KSselect and qGAP to a variety of research questions, meaning they could have a significant impact on plant sciences.

References

Abas, L., Benjamins, R., Malenica, N., Paciorek, T., Wisniewska, J., et al. 2006. Intracellular trafficking and proteolysis of the Arabidopsis auxin-efflux facilitator PIN2 are involved in root gravitropism. *Nat Cell Biol.* **8**(3):249-56.

Albrecht, V., Ritz, O., Linder, S., Harter, K. and Kudla, J. 2001. The NAF domain defines a novel protein-protein interaction module conserved in Ca regulated kinases. *The EMBO Journal* **20**(5): 1051-1063.

Altschul, S. F., Gish, W., Miller, W., Myers, E, W and Lipman, D, J. 1990. Basic local alignment search tool. *Journal of Molecular Biology* **215**(3): 403-410.

Anthony, R. G et al. 2004. A protein kinase target of a PDK1 signalling pathway is involved in root hair growth in Arabidopsis. *The EMBO Journal* **23**: 572-581.

Arnaud, C., Bonnot, C., Desnos, T. and Nussaume, L. 2010. The root cap at the forefront. *Comptes Rendus Biologies* **333**: 335-343.

Bai, H., Murali, B., Barber, K. and Wolverton, C. 2013. Low phosphate alters lateral root setpoint angle and gravitropism. American Journal of Botany **100**(1): 175-182.

Baldwin, K. L., Strohm, A. K. and Masson, P. H. 2013. Gravity sensing and signal transduction in vascular plant primary roots. *American Journal of Botany* **100**(1): 126-142.

Band, L. R., Wells, D. M., Larrieu, A., Sun, J., Middleton, A. M., French, A. P., Brunoud, G. et al. 2012 . Root gravitropism is regulated by a transient lateral auxin gradient controlled by a tipping-point mechanism. *Proceedings of the National Academy of Sciences* **109**: 4668-4673.

Barbez, E., Dunser, K., Gaidora, A., Lendl, T., and Busch, W. 2017. Auxin steers root cell expansion via apoplastic pH regulation in Arabidopsis thaliana. *PNAS* **114**(24):E4884-E4893.

Barbosa, I. C. R., Hammes, U. Z. and Schwechheimer, C. 2018. Activation and Polarity Control of PIN-FORMED Auxin Transporters by Phosphorylation. *Trends in Plant Science* **23**(6): 523-538.

Batistic, O., Sorek, N., Schultke, S., Yalovsky, S. and Kudla, J. 2008. Dual Fatty Acyl Modification Determines the Localization and Plasma Membrane Targeting of CBL/CIPK Ca21 Signaling Complexes in Arabidopsis. *The Plant Cell* **20**: 1346-1362.

Batistic, O., Waadt, R., Steinhorst, L., Held, K. and Kudla, J. 2010. CBL-mediated targeting of CIPKs facilitates the decoding of calcium signals emanating from distinct cellular stores. *The Plant Journal* **61**: 211-222.

Blakeslee, J. J., Bandyopadhyay, A., Lee, O. R., Mravec, J., Titapiwatanakun, B. et al., 2007. Interactions among PIN-FORMED and P-Glycoprotein Auxin Transporters in Arabidopsis. *The Plant Cell* **19**: 131-147.

Blancaflor, E. B., Fasano, J. M. and Gilroy, S. 1998. Mapping the Functional Roles of Cap Cells in the Response of Arabidopsis Primary Roots to Gravity. *Plant Physiology* **116**, 213-222.

Boccalandro, H.E., De Simone, S.N., Bergmann-Honsberger, A., Schepens, I., Fankhauser, C. and Casal, J.J. 2008. PHYTOCHROME KINASE SUBSTRATE1 regulates root phototropism and gravitropism. *Plant Physiol.* **146**(1):108-15.

Boonsirichai, K., Guan, C., Chen, R. and Masson, P. H. 2002. Root Gravitropism: An Experimental Tool to Investigate Basic Cellular and Molecular Processes Underlying Mechanosensing and Signal Transmission in Plants. *Annual Review of Plant Biology* **53**, 421-447.

Boonsirichai, K., Sedbrook , J. C., Chen, R., Gilroy, S. and Masson, P. H. 2003. Altered Response To Gravity is a peripheral membrane protein that modulates gravity-induced cytoplasmic alkalinisation and lateral auxin transport in plant statocytes. *Plant Cell* **15**: 2612-2625.

Brady, S. M., Orlando, D. A. et al. 2007. A high-resolution root spatiotemporal map reveals dominant expression patterns. *Science* **318**(5851): 801-806.

Brinkworth, R. I., Breinl, R. A. and Kobe, B. 2003. Structural basis and prediction of substrate specificity in protein serine/threonine kinases. *Proc Natl Acad Sci U S A* **100**(1): 74-79.

Brooks, T. L., Miller, N. D. and Spalding, E. P. 2010. Plasticity of Arabidopsis Root Gravitropism throughout a Multidimensional Condition Space Quantified by Automated Image Analysis. *Plant Physiology* **152**: 206-216.

Chaves Sanjuan, A. et al. 2014. Structural basis of the regulatory mechanism of the plant CIPK family of protein kinases controlling ion homeostasis and abiotic stress. *PNAS* E4532-E4541.

Chen, L., Jiao, Z. H., Zheng, L. S., Zhang, Y. Y., Xie, S. T., Wang, Z. X. and Wu, J. W. 2009. Structural insight into the autoinhibition mechanism of AMP-activated protein kinase. *Nature* **459**: 1146-1150.

Chen, H. and Xiong, L. 2010. Myo-Inositol-1-phosphate synthase is required for polar auxin transport and organ development. *J Biol Chem.* **285**(31):24238-47.

Chen, X., Gu, Z., Xin, D., Hao, L., Liu, C., Huang, J., Ma, B. and

Zhang, H. 2011. Identification and characterization of putative CIPK genes in maize. *Journal of Genetics and Genomics* **38**: 77-87.

Chen, I., Wang, Q. Q., Zhou, I., Ren, F., Li, D. D. and Li, X. B. 2013. Arabidopsis CBL-interacting protein kinase (CIPK6) is involved ion plant reponse to salt/osmotic stress and ABA. *Molecular and Biological Reports* **40**(8): 4759-4767.

Chen, T., Cui, P. and Xiong, L. 2015. The RNA-binding protein HOS5 and serine/arginine-rich proteins RS40 and RS41 participate in miRNA biogenesis in Arabidopsis. *Nucleic Acids Res.* **43**(17):8283-98.

Cho, M. and Cho, H. T. 2013. The function of ABCB transporters in auxin transport. *Plant Signaling and Behavior* **8**: 2.

Cho, M., Henry, E.M., Lewis, D.R., Wu, G., Muday, G.K. and Spalding, E.P. 2014. Block of ATP-binding cassette B19 ion channel activity by 5-nitro-2-(3-phenylpropylamino)-benzoic acid impairs polar auxin transport and root gravitropism. *Plant Physiol.* **166**(4):2091-9.

Chong, K. and Zhuang, X. 2007. ARF-GTPase as a Molecular Switch for Polar Auxin Transport Mediated by Vesicle Trafficking in Root Development. *Plant Signal Behav.* **2**(2):101-2.

Creixell, P., Palmeri, A., Miller, C. J., Lou, H. J., Santini, C. C. et al., 2015. Unmasking Determinants of Specificity in the Human Kinome. *Cell* **163**: 187-201.

Christie, J. M. 2007. Phototropin Blue-Light Receptors. *Annual Review of Plant Biology* **58**: 21-45.

Christie, J. M., Yang, H., Richter, G. L., Sullivan, S., Thomson, C. E., et al. 2011. phot1 Inhibition of ABCB19 Primes Lateral Auxin Fluxes in the Shoot Apex Required For Phototropism. *PLoS Biol* **9**(6): e1001076.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J.

et al. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**(11): 1422-1423.

Collings, D. A., Zsuppan G., Allen N. S., and Blancaflor, E. B. 2001. Demonstration of prominent actin filaments in the root columella. *Planta* **212**: 392-403.

DAngelo, C., et al. 2006. Alternative complex formation of the Ca regulated protein kinase CIPK1 controls abscisic acid-dependent and independent stress responses in Arabidopsis. *The Plant Journal* **48**: 857-872.

Darwin, C. R. 1880. The power of movement in plants. London: John Murray. *http://darwin-online.org.uk/content/frameset?itemID=F1325viewtype=textpageseq=1*

Dhonukshe, P., Tanaka, H., Goh, T., Ebine, K., Mahonen, A. P., Prasad, K., Blilou, I., Geldner, N., Xu, J., Uemura, T., Chory, J., Ueda, T., Nakano, A.,Scheres, B., and Friml, J. 2008. Generation of cell polarity in plants links endocytosis, auxin distribution and cell fate decisions. *Nature* **456**(7224): 962-966.

Dhonukshe, P., et al. 2010. Plasma membrane-bound AGC3 kinases phosphorylate PIN auxin carriers at TPRXS(N/S) motifs to direct apical PIN recycling. *Development* **137**: 3245-3255.

Digby, J., and Firn, R. D. 2002. Light modulation of the gravitropic setpointangle (GSA). *Journal of Experimental Botany* **53**: 377-381.

Eapen, D., Barroso, M.L., Campos, M.E., Ponce, G., Corkidi, G. et al. 2003. A no hydrotropic response root mutant that responds positively to gravitropism in Arabidopsis. *Plant Physiol.* **131**(2):536-46.

Ek Ramos, M. J., Avila, J., Cheng, C., Martin, G. B. and Devarenne, T. P. 2010. The T-loop Extension of the Tomato Protein Kinase AvrPto-dependent Pto-interacting Protein 3 (Adi3) Directs Nuclear Localization

for Suppression of Plant Cell Death. *The Journal of Biological Chemistry* **285**(23): 17584-17594.

Fasano, J. M., Massa, G. D. and Gilroy S. 2002. Ionic signalling in plant responses to gravity and touch. *Journal of Plant Growth Regulation* **21**(2): 71-88.

Fasano, J. M., Swanson, S. J. Blancaflor, E. B. Dowd, P. E. Kao, T and Gilroy, S. 2001. Changes in Root Cap pH Are Required for the Gravity Response of the Arabidopsis Root. *The Plant Cell* **13**: 907-921.

Fendrych, M., Leung, J., Friml, J. 2016. TIR1/AFB-Aux/IAA auxin perception mediates rapid cell wall acidification and growth of Arabidopsis hypocotyls. *Elife* **5**: e19048.

Fitzelle, K. J. and Kiss, J. Z. 2001. Restoration of gravitropic sensitivity in starch-deficient mutants of Arabidopsis by hypergravity. *Journal of Experimental Botany* **52**: 265-275.

French, A., Ubeda-Tomas, S., Holman, T. J., Bennett, M. J. and Pridmore, T. 2009. High-Throughput Quantfication of Root Growth Using a Novel Image-Analysis Tool. *Plant Physiology* **150**: 1784-1795.

Friml, J., Wisniewska, J., Benkova, E., Mendgen, K. and Palme, K. 2002. Lateral relocation of auxin efflux regulator PIN3 mediates tropism in Arabidopsis. *Nature* **415**, 806-809.

Fujii, N., Miyabayashi, S., Sugita, T., Kobayashi, A., Yamazaki, C. et al. 2018. Root-tip-mediated inhibition of hydrotropism is accompanied with the suppression of asymmetric expression of auxin-inducible genes in response to moisture gradients in cucumber roots. *PLoS One.* **13**(1):e0189827.

Galvan-Ampudia, C. S. and Offringa, R. 2007. Plant evolution: AGC kinases tell the auxin tale. *Trends in Plant Science* **12**(12).

Ganguly, A., Park, M., Kesawat, M.S. and Cho, H.T. 2014. Functional

Analysis of the Hydrophilic Loop in Intracellular Trafficking of Arabidopsis PIN-FORMED Proteins. *Plant Cell.* **26**(4):1570-1585.

Gish, W and States, D, J. 1993. Identification of protein coding regions by database similarity search. *Nature Genetics* **3**(3): 266-72.

Goldberg, J., Nairn, A. C. and Kuriyan, J. 1996. Structural Basis for the Autoinhibition of Calcium/Calmodulin-Dependent Protein Kinase I. *Cell* **84**: 875-887.

Goodstein, D. M., Shu, S., Howson, R., Neupane, R., Hayes, R. D. et al. 2012. Phytozome: a comparative platform for green plant genomics. *Nucleic Acids Res.* **40**: D1178-D1186.

Gray, J. W., Dittrich, A. C. N., Chen, S., Avila, J., Giavalisco, P. and Devarenne, T. P. 2013. Two Pdk1 phosphorylation sites on the plant cell death suppressor Adi3 contribute to substrate phosphorylation. *Biochimica et Biophysica Acta* **1834**(6): 1099-1106.

Grones, P., Abas, M., Hajny, J., Jones, A., Waidmann, S., Kleine-Vehn, J. and Friml, J. 2018. PID/WAG-mediated phosphorylation of the Arabidopsis PIN3 auxin transporter mediates polarity switches during gravitropism. *Sci Rep.* **8**(1):10279.

Guan, C., Rosen, E.S., Boonsirichai, K., Poff, K.L. and Masson, P.H. 2003. The ARG1-LIKE2 gene of Arabidopsis functions in a gravity signal transduction pathway that is genetically distinct from the PGM pathway. *Plant Physiol.* **133**(1):100-12.

Guo, Y., Halfter, U., Ishitani, M. and Zhu J. K. 2001. Molecular Characterization of Functional Domains in the Protein Kinase SOS2 That Is Required for Plant Salt Tolerance. *The Plant Cell* **13**: 1383-1399.

Haga, K., Hayashi, K. and Sakai, T. 2014. PINOID AGC Kinases Are Necessary for Phytochrome-Mediated Enhancement of Hypocotyl Phototropism in Arabidopsis. *Plant Physiol.* **166**(3): 1535-1545.

Hanks, S. K., Quinn, A. M. and Hunter, T. 1988. The protein kinase family: conserved features and deduced phylogeny of the catalytic domains. *Science* **241**(4861):42-52.

Hanks, S. K. and Hunter, T. 1995. The eukaryotic protein kinase superfamily: kinase (catalytic) domain structure and classification. *The FASEB Journal* **9**: 576-596.

Heijnowicz, Z., Sondag, C., Alt, W. and Sievers, A. 1998. Temporal course of graviperception in intermittently stimulated cress roots. *Plant, Cell and Environment* **21**(12): 1293-300.

Henrichs, S., Wang1, B., Fukao, Y., Zhu, J., Charrier, L. et al. 2012. Regulation of ABCB1/PGP1-catalysed auxin transport by linker phosphorylation. *The EMBO Journal* **31**: 2965-2980.

Hou, G., Kramer, V. L., Wang, Y., Chen, R., Perbal, G., Gilroy, S. and Blancaflor, E. B. 2004. The promotion of gravitropism in Arabidopsis roots upon actin disruption is coupled with the extended alkalinization of the columella cytoplasm and a persistent lateral auxin gradient. *The Plant Journal* **39**(1): 113-125.

Huang, F., Zago, M. K., Abas, L., van Marion, A., Galvan-Ampudia, C. S. and Offringa, R. 2010. Phosphorylation of Conserved PIN Motifs Directs Arabidopsis PIN1 Polarity and Auxin Transport. *Plant Cell* **22**(4): 1129-1142.

Huang, C., Ding, S., Zhang, H., Du, H. and An, L. 2011. CIPK7 is involved in cold response by interacting with CBL1 in Arabidopsis thaliana. *Plant Science* **181**: 57-64.

Huang, G., Liang, W., Sturrock, C.J., Pandey, B.K., Giri, J., et al. 2018. Rice actin binding protein RMD controls crown root angle in response to external phosphate. *Nat Commun.* **9**(1):2346.

Huerta-Cepas, J., Serra, F. and Bork, P. 2016. ETE 3: Reconstruction, analysis and visualization of phylogenomic data. *Mol Biol Evol.*

Jeffery, C. J. 1999. Moonlighting proteins. *Trends Biochem Sci.* **24**(1): 8-11.

Jin, J. and Pawson, T. 2012. Modular evolution of phosphorylation-based signalling systems. *Philos Trans R Soc Lond B Biol Sci.* **367**(1602): 2540-55.

Joo, J.H., Bae, Y.S. and Lee J.S. 2001. Role of auxin-induced reactive oxygen species in root gravitropism. *Plant Physiol.* **126**(3):1055-60.

Jordan, J. D., Landau, E. M. and Iyengar, R. 2000. Signalling Networks: The Origins of Cellular Multitasking. *Cell* **103**(2): 193-200.

Juniper, B. E., Gorves, S., Landau Schachar, B., Audus, L. J. 1966. Root cap and the perception of gravity. *Nature* **209**, 93-94.

Kafri, R., Levy, M. and Pilpel, Y. 2006. The regulatory utilization of genetic redundancy through responsive backup circuits. *PNAS* **103**(31): 11653-11658.

Kafri, R. 2009. Genetic Redundancy: New Tricks for Old Genes. *Cell* **136**(3): 389-392.

Kannan, N., Haste, N., Taylor, S. S. and Neuwald, A. F. 2007. The hallmark of AGC kinase functional divergence is its C-terminal tail, a cis-acting regulatory module. *PNAS* **104**(4): 1272-1277.

Ki, D., Sasayama, D. and Cho H.T. 2016. The M3 Phosphorylation Site Is Required for Trafficking and Biological Roles of PIN-FORMED1, 2, and 7 in Arabidopsis. *Front Plant Sci.* **7**:1479.

Kilian, J., Whitehead, D., Horak, J., Wanke, D., Weinl, S., et al. 2007. The AtGenExpress global stress expression data set: protocols, evaluation and model data analysis of UV-B light, drought and cold stress responses. *Plant J.* **2**:347-63.

Kim, B. G. et al. 2007. The calcium sensor CBL10 mediates salt tolerance by regulating ion homeostasis in Arabidopsis. *The Plant Journal* **52**: 473-484.

Kleine-Vehn, J., Ding, Z., Jones, A. R., Tasak, M., Morita, M. T. and Friml, J. 2010. Gravity-induced PIN transcytosis for polarization of auxin fluxes in gravity-sensing root cells. *PNAS* Available at: www.pnas.org/cgi/doi/10.1073/pnas.1013145107 [Accessed 30 June 2014].

Kleist, T. J., Spencley, A. L. and Luan, S. 2014. Comparative phylogenomics of the CBL-CIPK calcium-decoding network in the moss Physcomitrella, Arabidopsis, and other green lineages. *Frontiers in Plant Science* **5**(187): 1-17.

Knight, T. 1806. On the direction of the radicle and germen during the vegetation of seeds. *Philosophical Transactions of the Royal Society of London, Biological Sciences* **99**: 108-120.

Kobayashi, A., Takahashi, A., Kakimoto, Y., Miyazawa, Y., Fujii, N., Higashitani, A. and Takahashi, H. 2007. A gene essential for hydrotropism in roots. *Proc Natl Acad Sci U S A.* **104**(11):4724-9.

Kolch, W., Calder, M. and Gilbert, D. 2005. When kinases meet mathematics: the systems biology of MAPK signalling. *FEBS Lett.* **579**(8):1891-5.

Kolukisaoglu, U., Weinl, S., Blazevic, D., Batistic, O. and Kudla, J. 2004. Calcium Sensors and Their Interacting Protein Kinases: Genomics of the Arabidopsis and Rice CBL-CIPK Signaling Networks. *Plant Physiology* **134**: 43-58.

Kondrachuk, A. V. and Hasenstein, K. H. 2001. The effects of HGMFs on the plant gravisensing system. *Advances in Space Research* **27**(5): 1001-1005.

Kong, S. G., Suzuki, T., Tamura, K., Mochizuki, N., Hara-Nishimura,

I. and Nagatani, A. 2006. Blue light-induced association of phototropin 2 with the Golgi apparatus. *The Plant Journal* **45**: 994-1005.

Krieger, G., Shkolnik, D., Miller, G. and Fromm, H. 2016. Reactive Oxygen Species Tune Root Tropic Responses. *Plant Physiol.* **172**(2):1209-1220.

Krzeszowlec, W. and Gabrys, H. 2007. Phototropin Mediated Relocation of Myosins in Arabidopsis thaliana. *Plant Signalling and Behaviour* **2**(5): 333-336.

Kuznetsova, A., Brockhoff, P. B. and Christensen, R. H. B. 2017. lmerTest Package: Tests in Linear Mixed Effects Models. *Journal of Statistical Software* **82**(13).

Lampropoulos, A., Sutikovic, Z., Wenzl, C., Maegele, I., Lohmann, J. U. and Forner, J. 2013. GreenGate - A Novel, Versatile, and Efficient Cloning System for Plant Transgenesis. *Elife* **8**(12): e83043.

Leitz, G., Kang, B., Schoenwaelder, M. E. A. and Staehelin, L. A. 2009. Statolith Sedimentation Kinetics and Force Transduction to the Cortical Endoplasmic Reticulum in Gravity-Sensing Arabidopsis Columella Cells. *The Plant Cell* **21**: 843-860.

Lewis, D.R., Miller, N.D., Splitt, B.L., Wu, G. and Spalding, E.P. 2007. Separating the roles of acropetal and basipetal auxin transport on gravitropism with mutations in two Arabidopsis multidrug resistance-like ABC transporter genes. *Plant Cell.* **19**(6):1838-50.

Li, Y., Dai, X., Cheng, Y. and Zhao, Y. 2011. NPY genes play an essential role in root gravitropic responses in Arabidopsis. *Mol Plant.* **4**(1):171-9.

Liu, F., Hu, W. and Vierstra, R.D. 2018. The Vacuolar Protein Sorting-38 Subunit of the Arabidopsis Phosphatidylinositol-3-Kinase Complex Plays Critical Roles in Autophagy, Endosome Sorting, and Gravitropism. *Front Plant Sci.* **9**:781.

Lofke, C., Zwiewka, M., Heilmann, I., Van Montagu, M.C., Teichmann, T. and Friml, J. 2013. Asymmetric gibberellin signaling regulates vacuolar trafficking of PIN auxin transporters during root gravitropism. *Proc Natl Acad Sci U S A.* **110**(9):3627-32.

Marchant, A., Kargul , J., May , S. T., Muller, P., Delbarre, A., Perrotrechenmann, C., and Bennett, M. J. 1999. AUX1 regulates root gravitropism in Arabidopsis by facilitating auxin uptake within root apical tissues. *EMBO Journal* **18**: 2066-2073.

Massa, G. D. and Gilroy, S. 2003. Touch modulates gravity sensing to regulate the growth of primary roots of Arabidopsis thaliana. *Plant Journal* **33**(3): 435-445.

Mattoo, A. K. 2014. Translational research in agricultural biology-enhancing crop resistivity against environmental stress alongside nutritional quality. *Frontiers in Chemistry: Agricultural Biological Chemistry* **2**: 30.

Middleton, A. M., Farcot, E., Owen, M. R. and Vernoux, T. 2012. Modelling Regulatory Networks to Understand Plant Development: Small Is Beautiful. *The Plant Cell* **24**: 3876-3891.

Monshausen, G. B., Miller, N. D., Murphy, A. S. and Gilroy, S. 2011. Dynamics of auxin-dependent Ca and pH signaling in root growth revealed by integrating high-resolution imaging with automated computer vision-based analysis. *The Plant Journal* **65**: 309-318.

Moore, C.R., Johnson, L.S., Kwak, I.Y., Livny, M., Broman, K.W. and Spalding, E.P. 2013. High-throughput computer vision introduces the time axis to a quantitative trait map of a plant growth response. *Genetics.* **195**(3):1077-86.

Nolen, B., Taylor, S. and Ghosh, G. 2004. Regulation of Protein Kinases: Controlling Activity through Activation Segment Conformation. *Molecular Cell* **15**: 661-675.

Nziengui, H., Lasok, H., Kochersperger, P., Ruperti, B., Rebeille, F. et al. 2018. Root Gravitropism Is Regulated by a Crosstalk between para-Aminobenzoic Acid, Ethylene, and Auxin. *Plant Physiol.* **178**(3):1370-1389.

Ohta, M., Guo, Y., Halfter, U. and Zhu, J. K. 2003. A novel domain in the protein kinase SOS2 mediates interaction with the protein phosphatase 2C ABI2. *PNAS* **100**(20): 11771-11776.

Okushima, Y., Overvoorde, P. J., Arima, K., Alonso, J. M., Chan, A. et al. 2005. Functional Genomic Analysis of the AUXIN RESPONSE FACTOR Gene Family Members in Arabidopsis thaliana: Unique and Overlapping Functions of ARF7 and ARF19. *Plant Cell.* **17**(2):444-463.

Peer, W.A., Hosein, F.N., Bandyopadhyay, A., Makam, S.N., Otegui, M.S., et al. 2009. Mutation of the membrane-associated M1 protease APM1 results in distinct embryonic and seedling developmental defects in Arabidopsis. *Plant Cell.* **21**(6):1693-721.

Perbal, G. and Driss-Ecole D. 2003. Mechanotransduction in gravisensing Cells. *Trends in Plant Science* **8**(10) 322-344.

Plieth, C. and Trewavas, A. J. 2002. Reorientation of Seedlings in the Earths Gravitational Field Induces Cytosolic Calcium Transients. *Plant Physiology* **129**: 786-796.

Rademacher, E. H. and Offringa, R. 2012. Evolutionary adaptations of plant AGC kinases: from light signaling to cell polarity regulation. *Frontiers in Plant Science* **3**(250).

Rahman, A., Takahashi, M., Shibasaki, K., Wu, S., Inaba, T. et al. 2010. Gravitropism of Arabidopsis thaliana roots requires the polarization of PIN2 toward the root tip in meristematic cortical cells. *Plant Cell.* **22**(6):1762-76.

Rashotte, A. M., Brady, S. R., Reed, R. C., Ante, S. J. and Muday, G. K. 2000. Basipetal Auxin Transport Is Required for Gravitropism in Roots

of Arabidopsis. *Plant Physiology* **122**: 481-490.

Rommel, C., Vanhaesebroeck, B. and Vogt, P. K. (eds). 2010. *Phosphoinositide 3-kinase in Health and Disease* Vol 1. Springer-Verlag Berlin Heidelberg, pp. 14-16.

Roychoudhry, S., Del Bianco, M., Kieffer, M. and Kepinski, S. 2013. Auxin controls gravitropic setpoint angle in higher plant lateral branches. *Curr Biol.* **23**(15):1497-504.

Ruppel, N.J., Logsdon, C.A., Whippo, C.W., Inoue, K. and Hangarter, R.P. 2011. A mutation in Arabidopsis seedling plastid development1 affects plastid differentiation in embryo-derived tissues during seedling growth. *Plant Physiol.* **155**(1):342-53.

Sabatini, S., Beis, D., Wolkenfelt, H., Murfett, J., Guilfoyle, T., Malamy, J., Benfey, P., Leyser, O., Bechtold, N., Weisbeek, P. and Scheres, B. 1999. An Auxin-Dependent Distal Organizer of Pattern and Polarity in the Arabidopsis Root. *Cell* **99**: 463-472.

Sack, F. D. and Kiss, J. Z. 1989. Root cap structure in wild type and in a starchless mutant of Arabidopsis. *American Journal of Botany* **76**: 454-464.

Sanchez-Barrena, M. J., Martinez-Ripoll, M. and Albert, A. 2013. Structural Biology of a Major Signaling Network that Regulates Plant Abiotic Stress: The CBL-CIPK Mediated Pathway. *International Journal of Molecular Sciences* **14**: 5734-5749.

Sanders, D., Pelloux, J., Brownlee, C. and Harper, J. F. 2002. Calcium at the crossroads of signalling. *Plant Cell* **14**: 401-417.

Scholler, M., Sarkel, E., Kleine-Vehn, J. and Feraru, E. 2018. Growth Rate Normalization Method to Assess Gravitropic Root Growth. *Methods Mol Biol.* **1761**:199-208.

Scott, A. M. and Allen, N. S. 1999. Changes in Cytosolic pH within Arabidopsis Root Columella Cells Play a Key Role in the Early Signalling Pathway for Root Gravitropism. *Plant Physiology* **121**: 1291-1298.

Shi, Q., Zhang, H., Song, X., Jiang, Y., Liang, R. and Li, G. Functional Characterization of the Maize Phytochrome-Interacting Factors PIF4 and PIF5. *Front Plant Sci.* **8**:2273.

Shibasaki, K., Uemura, M., Tsurumi, S. and Rahman, A. 2009. Auxin Reponse in Arabidopsis under Cold Stress: Underlying Molecular Mechanisms. *The Plant Cell* **21**: 3823-3838.

Shin, K., Lee, R.A., Lee, I., Lee, S., Park, S.K. and Soh, M.S. 2013. Genetic identification of a second site modifier of ctr1-1 that controls ethylene-responsive and gravitropic root growth in Arabidopsis thaliana. *Mol Cells.* **36**(1):88-96.

Stamatakis, A. 2014. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30**(9): 1312-1313.

Stanga, J. P., Boonsirichai, K., Sedbrook, J. C., Otegui, M. S and Masson, P. H. 2009. A role for the TOC complex in Arabidopsis root gravitropism. *Plant Physiology* **149**: 1896-1905.

Steinber, S. F. 2008. Structural Basis of Protein Kinase C Isoform Function. *Physiol Rev.* **88**(4): 1341-1378.

Sukumar, P., Edwards, K.S., Rahman, A., Delong, A. and Munday G.K. 2009. PINOID kinase regulates root gravitropism through modulation of PIN2-dependent basipetal auxin transport in Arabidopsis. *Plant Physiol.* **175**(2):722-35.

Swarup, R., Friml, J., Marchant, A., Ljung, K., Sandberg, G., Palme, K. and Bennett, M. 2001. Localization of the auxin permease AUX1 suggests two functionally distinct hormone transport pathways operate in the Arabidopsis root apex. *Genes and Development* **15**: 2648-2653.

Swarup, R., Kramer, E. M., Perry, P., Knox, K., Leyser, O., Haseloff, J., Beemster G. T. S., Bhalerao, R. and Bennett, M. J. 2005. Root gravitropism requires lateral root cap and epidermal cells for transport and response to a mobile auxin signal. *Nature Cell Biology* **7**(11): 1057-1065.

Swarup, R., Wells, D. and Bennett, M. J. 2013. Root gravitropism. In: Eshel, A., Beeckman, T. (eds). *The Hidden half* IV ed: CRC press, pp. 19-34.

Takahashi, N., Yamazaki, Y., Kobayashi, A., Higashitani, A. and Takahashi, H. 2003. Hydrotropism interacts with gravitropism by degrading amyloplasts in seedling roots of Arabidopsis and radish. *Plant Physiol.* **132**(2):805-10.

Taniguchi, M., Furutani, M., Nishimura, T., Nakamura, M., Fushita, T., et al. 2017. The Arabidopsis LAZY1 Family Plays a Key Role in Gravity Signaling within Statocytes and in Branch Angle Control of Roots and Shoots. *Plant Cell.* **29**(8):1984-1999.

Taylor, S. S, and Kornev, A. P. 2011. Protein Kinases: Evolution of Dynamic Regulatory Proteins. *Trends in Biochemical Science* **36**(2): 65-77.

Taylor, S. S., Keshwani, M. M., Steichen, J. M. and Kornev, A. P. 2012. Evolution of the eukaryotic protein kinases as dynamic molecular switches. *Philosophical Transactions of the Royal Society* **367**: 2517-2528.

Titapiwatanakun, B. and Murphy, A. S. 2008. Post-transcriptional regulation of auxin transport proteins: cellular trafficking, protein phosphorylation, protein maturation, ubiquitination, and membrane composition. *Journal of Experimental Botany* 60(4): 1093-1107.

Toyota, M., Furuichi, T., Tatsumi, H. and Sokabe, M. 2008. Cytoplasmic Calcium Increases in Response to Changes in the Gravity Vector in Hypocotyls and Petioles of Arabidopsis Seedlings. *Plant Physiology* **146**:

505-514.

Turk, B. E. 2008. Understanding and exploiting substrate recognition by protein kinases. *Curr Opin Chem Biol.* **12**(1):4-10.

Ubeda-Tomas, S., Beemster, G. T. S. and Bennett, M. J. 2012. Hormonal regulation of root growth: integrating local activities into global behaviour. *Trends in Plant Science* **17**(6).

Uemura, T., Kim, H., Saito, C., Ebine, K., Ueda, T., Schulze-Lefert, P. and Nakano, A. 2012. Qa-SNAREs localized to the trans-Golgi network regulate multiple transport pathways and extracellular disease resistance in plants. *Proc Natl Acad Sci U S A.* **109**(5):1784-9.

Vandenbussche, F., Smalle, J., Le, J., Saibo, N.J., De Paepe, A., et al. 2003. The Arabidopsis mutant alh1 illustrates a cross talk between ethylene and auxin. *Plant Physiol.* **131**(3):1228-38.

Vieten, A., Vanneste, S., Wisniewska, J., Benkova, E., Benjamins, R., Beeckman, T., Luschnig, C. and Friml, J. (2005). Functional redundancy of PIN proteins is accompanied by auxin-dependent cross-regulation of PIN expression. *Development* **132**: 4521-4531.

Wan, J., Zhang, P., Wang, R., Sun, L., Wang, W. et al. 2018. UV-B Radiation Induces Root Bending Through the Flavonoid-Mediated Auxin Pathway in Arabidopsis. *Front Plant Sci.* **9**:618.

Wang, H.Z., Yang, K.Z., Zou, J.J., Zhu, L.L., Xie, Z.D., et al. 2015. Transcriptional regulation of PIN genes by FOUR LIPS and MYB88 during Arabidopsis root gravitropism. *Nat Commun.* **6**:8822.

Wattelet-Boyer, V., Brocard, L., Jonsson, K., Esnay, N., Joubes, J. et al. 2016. Enrichment of hydroxylated C24- and C26-acyl-chain sphingolipids mediates PIN2 apical sorting at trans-Golgi network subdomains. *Nat Commun.* **7**:12788.

Wells, D. M., French, A. P., Naeem, A., Ishaq, O., Traini, R. et al. 2012. Recovering the dynamics of root growth and development using novel image acquisition and analysis methods. *Philos Trans R Soc Lond B Biol Sci.* **367**(1595): 1517-1524.

Willige, B. C, Ahlers, S., Zourelidou, M., Barbosa, I. C., Demarsy, E., Trevisan, M., Davis, P. A., Roelfsema, M. R., Hangarter, R., Fankhauser, C. and Schwechheimer, C. 2013. D6PK AGCVIII kinases are required for auxin transport and phototropic hypocotyl bending in Arabidopsis. *Plant Cell* **25**(5): 1674-1688.

Wolverton, C., Paya, A. M. and Toska, J. 2011. Root cap angle and gravitropic response rate are uncoupled in the Arabidopsis pgm-1 mutant. *Physiologia Plantarum* **141**: 373-382.

Wyatt, S. E. and Kiss, J. Z. 2013. Plant tropisms: From Dariwn to the International Space Station. *American Journal of Botany* **100**(1): 1-3.

Yamanaka, S., Blau, H, M, 2010. Nuclear reprogramming to a pluripotent state by three approaches. *Nature* **465**: 704-712.

Yan, D.W., Wang, J., Yuan T.T., Hong, L.W., Gao, X., et al. 2013. Perturbation of Auxin Homeostasis by Overexpression of Wild-Type IAA15 Results in Impaired Stem Cell Differentiation and Gravitropism in Roots. *PLoS ONE* **8**(3):e58103.

Yokokura, H., Picciotti, M. R., Nairn, A. C. and Hidaka, H. 1995. The Regulatory Region of Calcium/Calmodulin-dependent Protein Kinase I Contains Closely Associated Autoinhibitory and Calmodulin-binding Domains. The Journal of Biological Chemistry **270**(40): 23851-23859.

Yoshihara T. and Spalding E.P. 2017. LAZY Genes Mediate the Effects of Gravity on Auxin Gradients and Plant Architecture. *Plant Physiol.* **175**(2):959-969.

Young, L.S., Harrison, B.R., Narayana Murthy, U.M., Moffatt, B.A.,

Gilroy, S. and Masson, P.H. 2006. Adenosine kinase modulates root gravitropism and cap morphogenesis in Arabidopsis. *Plant Physiol.* **142**(2):564-73.

Zegzouti, H. et al. 2006a. Structural and Functional Insights into the Regulation of Arabidopsis AGC VIIIa Kinases. *The Journal of Biological Chemistry* **281**(46): 35520-35530.

Zegzouti, H., Anthony, R. G., Jahchan, N., Bogre, L. and Christensen, S. K. 2006b. Phosphorylation and activation of PINOID by the phospholipid signaling kinase 3-phosphoinositide-dependent protein kinase 1 (PDK1) in Arabidopsis. *PNAS* **103**(16): 6404-6409.

Zhang, J., Nodzynski, T., Pencik, A., Rolcik, J. and Friml J. 2010. PIN phosphorylation is sufficient to mediate PIN polarity and direct auxin transport. *Proc Natl Acad Sci U S A.* **107**(2):918-22.

Zheng, H. Q. and Staehelin L. A. 2001. Nodal Endoplasmic Reticulum, a Specialized Form of Endoplasmic Reticulum Found in Gravity-Sensing Root Tip Columella Cells. *Plant Physiology* **125**: 252-265.

Zou, J.J., Zheng, Z.Y., Xue, S., Li, H.H., Wang, Y.R. and Le, J. 2016. The role of Arabidopsis Actin-Related Protein 3 in amyloplast sedimentation and polar auxin transport in root gravitropism. *J Exp Bot.* **67**(18):5325-5337.

Zourelidou, M., Muller, I., Willige, B, C., Nill, C., Jikumaru, Y., Li H and Schwechheimer. C. 2009. The polarly localized D6 PROTEIN KINASE is required for efficient auxin transport in Arabidopsis thaliana. *Development* **136**, 627-636.

Zourelidou, M., Absmanner, B., Weller, B., Barbosa, I. C. R., Willige, B. C., et al. 2014. Auxin efflux by PIN-FORMED proteins is activated by two different protein kinases, D6 PROTEIN KINASE and PINOID. *eLife* **3**: e02860.

Zulawski, M. and Schulze W. X. 2015. The Plant Kinome. In: Schulze W. X. (ed). *Plant Phosphoproteomics*, Humana press, pp. 1-24.

Published Papers

Müller, L., Bennett, M. J., French, A., Wells, D. M. and Swarup, R. 2018. Root Gravitropism: Quantification, Challenges, and Solutions. *Methods Mol Biol.* **1761**: 103-112.

Huang, G., Liang, W., Sturrock, C. J., Pandey, B. K., Giri, J., Mairhofer, S., Wang, D., Müller, L., Tan, H., York, L. M., Yang, J., Song, Y., Kim, Y-J., Qiao, Y., Xu, J., Kepinski, S., Bennett, M. J. and Zhang, D. 2018. Rice actin binding protein RMD controls crown root angle in response to external phosphate. *Nature Communications* **9**(1): 2346.

Appendix A

Supplemental Material

S1 Supplemental Figures

S1.1 Supplemental Figures for Chapter 2: Methods

Table S1: Details of CIPK mutants.

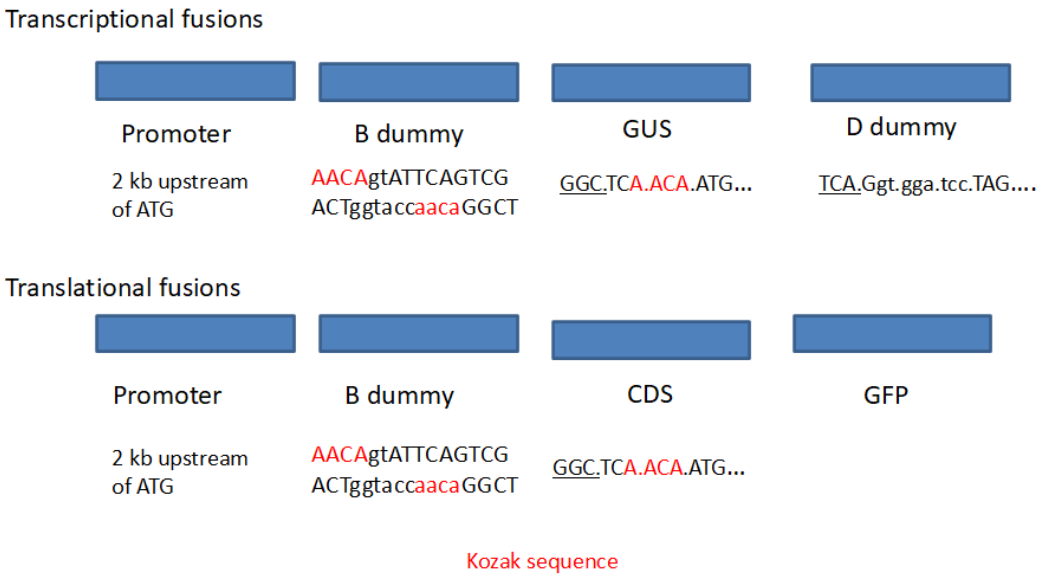
Gene name	T-DNA line	Name	Location
CIPK4	SALK_009893	*cipk4-1*	Exon
CIPK4	SALK_046245	*cipk4-2*	1000- prom
CIPK7	SALK_055294	*cipk7-1*	Exon
CIPK7	SALK_072998	*cipk7-2*	300- UTR

Table S2: GreenGate primers.

Primer name	Sequence
CIPK4pro_FW	AACAGGTCTCGACCTtggaaacagtttctctcatgttgg
CIPK4pro_RV	ACCAGGTCTCGTGTTtgtttgtggtggttgttgcg
CIPK7pro2_FW	AACAGGTCTCGCgtctcttaagtatgaaggtgttgtag
CIPK7pro2_RV	ACCAGGTCTCGTGTTggcggcggcggaagg
D6L3pro_FW	AACAGGTCTCGACCTcttgaaaggctaatttgatgatcaacg
D6L3pro_RV	ACCAGGTCTCGTGTTtgaaggtgttcgacaaaattcctg
CIPK4cds2_FW	ACCAGGTCTCGTgTCCGACTTTGGTCTATCG
CIPK4cds2_RV	AACAGGTCTCTCTGAATTGTGCCATGAGAGCACAATCTCC
CIPK7cds2_FW	ACCAGGTCTCGTgTCTGACTTTGGTTTATCAGCTTTACC
CIPK7cds2_RV	AACAGGTCTCTCTGACATGATGTCATTGTGCCATGAAAG
D6L3cds3_FW	ACCAGGTCTCGCCAaAGATTATCAAAGGAGAAGGTCATGG
D6L3cds3_RV	AACAGGTCTCTCTGAGAAGAAATCAATTTCCAAATAATTACCAGAAGG

Table S3: qPCR primers.

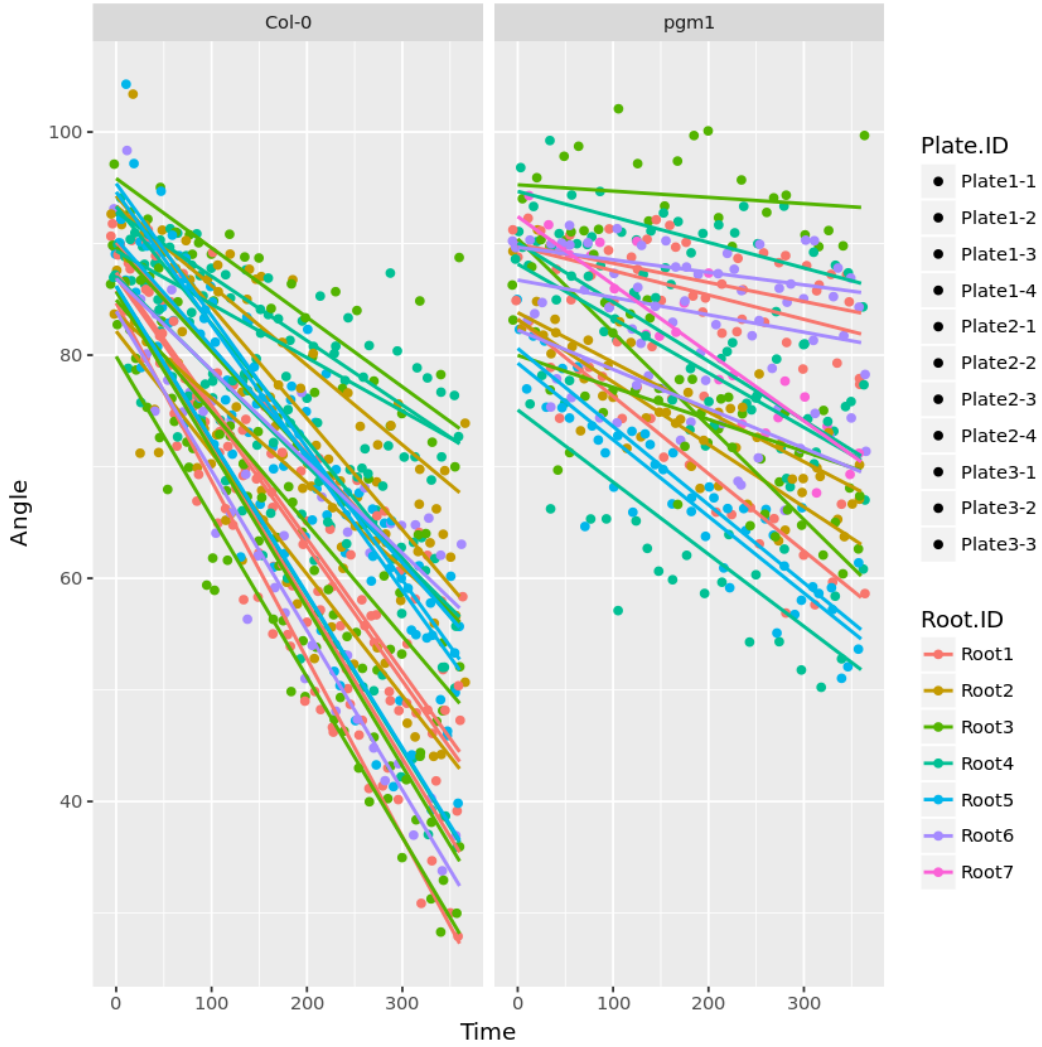
Primer name	Sequence
D6PKL3_q_LP	TTCTTGTTTGGTTATGGAGTTTTG
D6PKL3_q_RP	AGCAAGAAGAACTTCAGCAACAT
CIPK7_q_LP	aggtttcgggtgttgaagtg
CIPK7_q_RP	cataacctaacttctccccaatca
CIPK4_q_LP	gatctttcggggttgtttga
CIPK4_q_RP	cactctcgccgtaaacctct



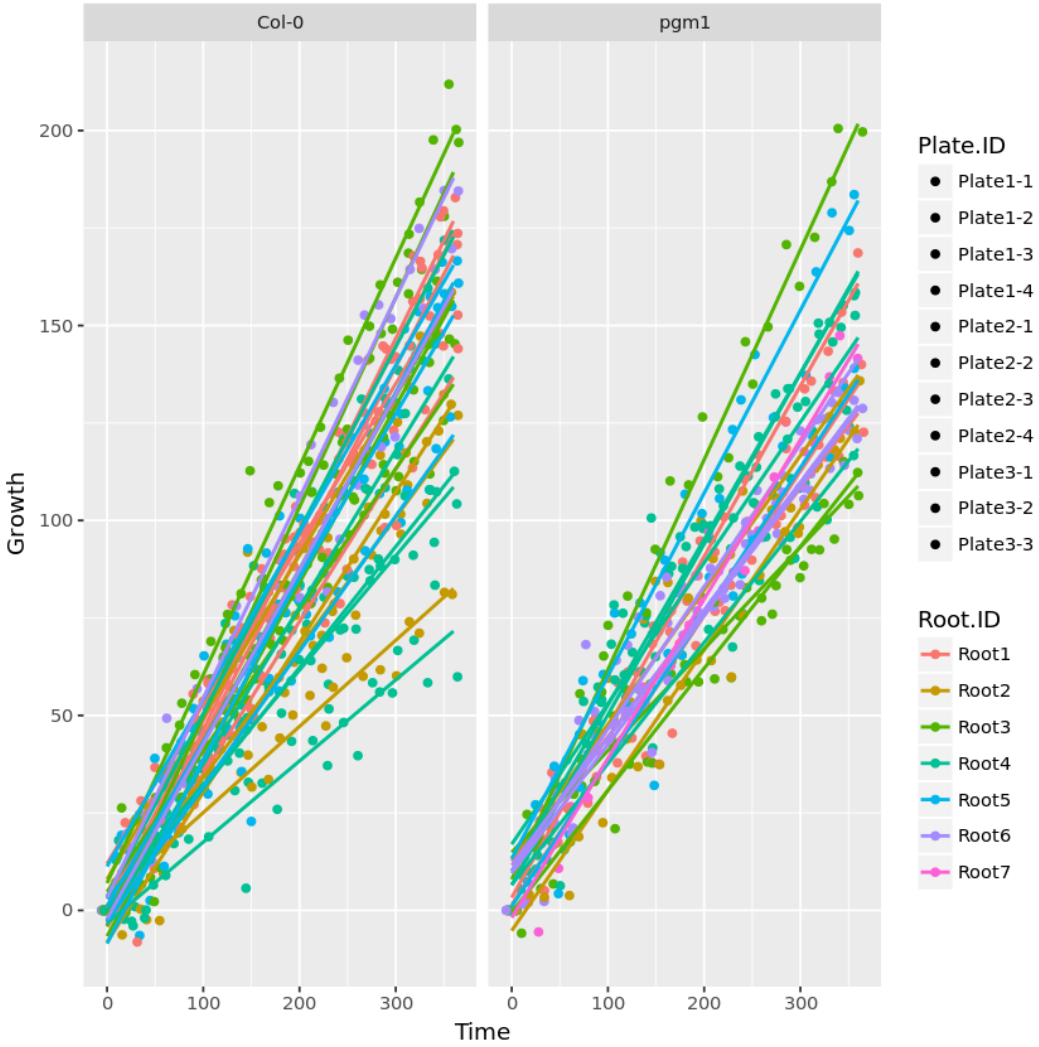
Supplemental Material, Figure S1: Schematic of Green Gate cloning strategy. See Lampropoulos *et al.* (2013) for details of Green Gate cloning protocol. Kozak sequences (red) were introduced using a modified B dummy, a modified GUS C-element, and via the primers used to amplify coding sequences (CDSs). This cloning strategy was used for CIPK4:GUS, CIPK7:GUS and CIPK7:CIPK7-GFP.

Table S4: Summary of Linear Mixed Model for *pgm1* dataset.

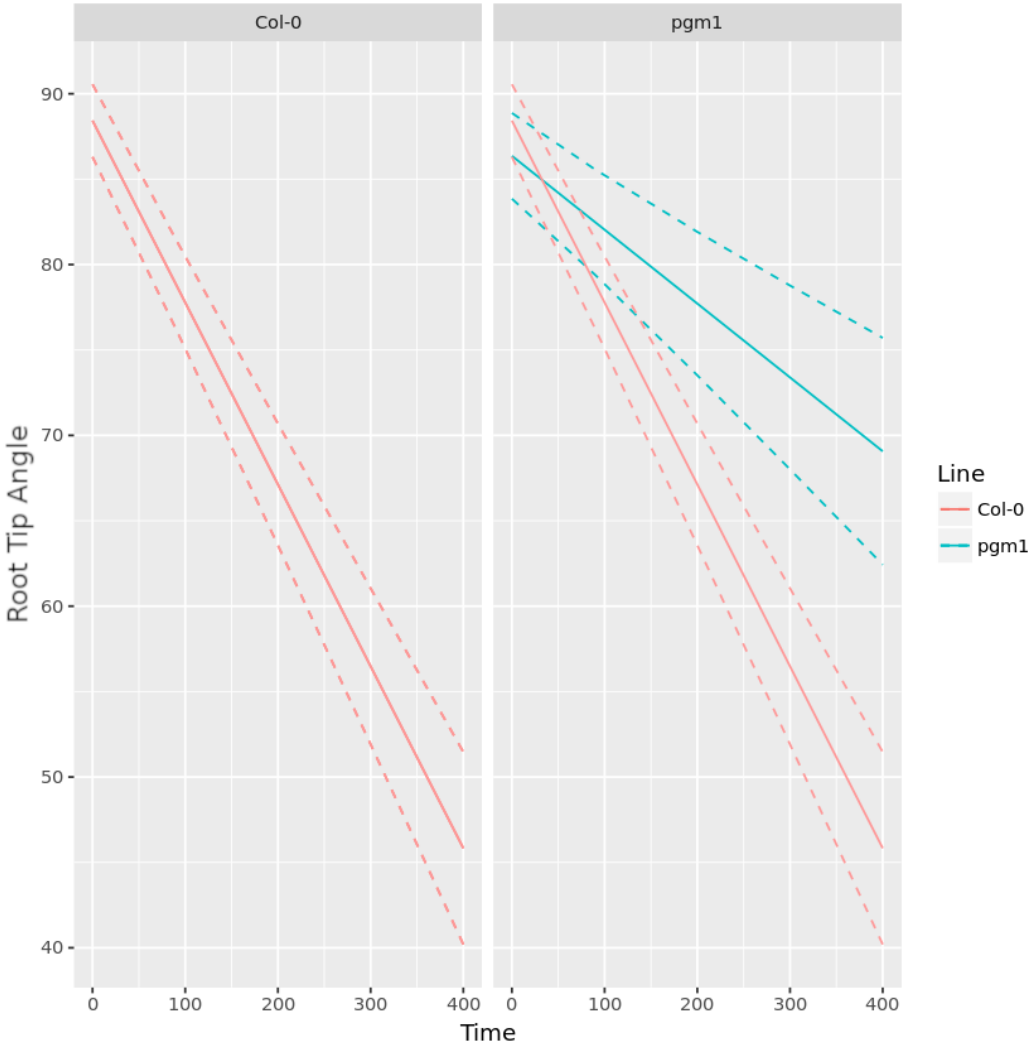
	Estimate	Std. Error	df	t value	Pr(¿—t—)
(Intercept)	88.43	1.08	40.96	81.56	0.00
Linepgm1	-2.06	1.68	41.06	-1.23	0.23
Time	-0.11	0.01	40.98	-17.92	0.00
Linepgm1:Time	0.06	0.01	41.03	6.88	0.00



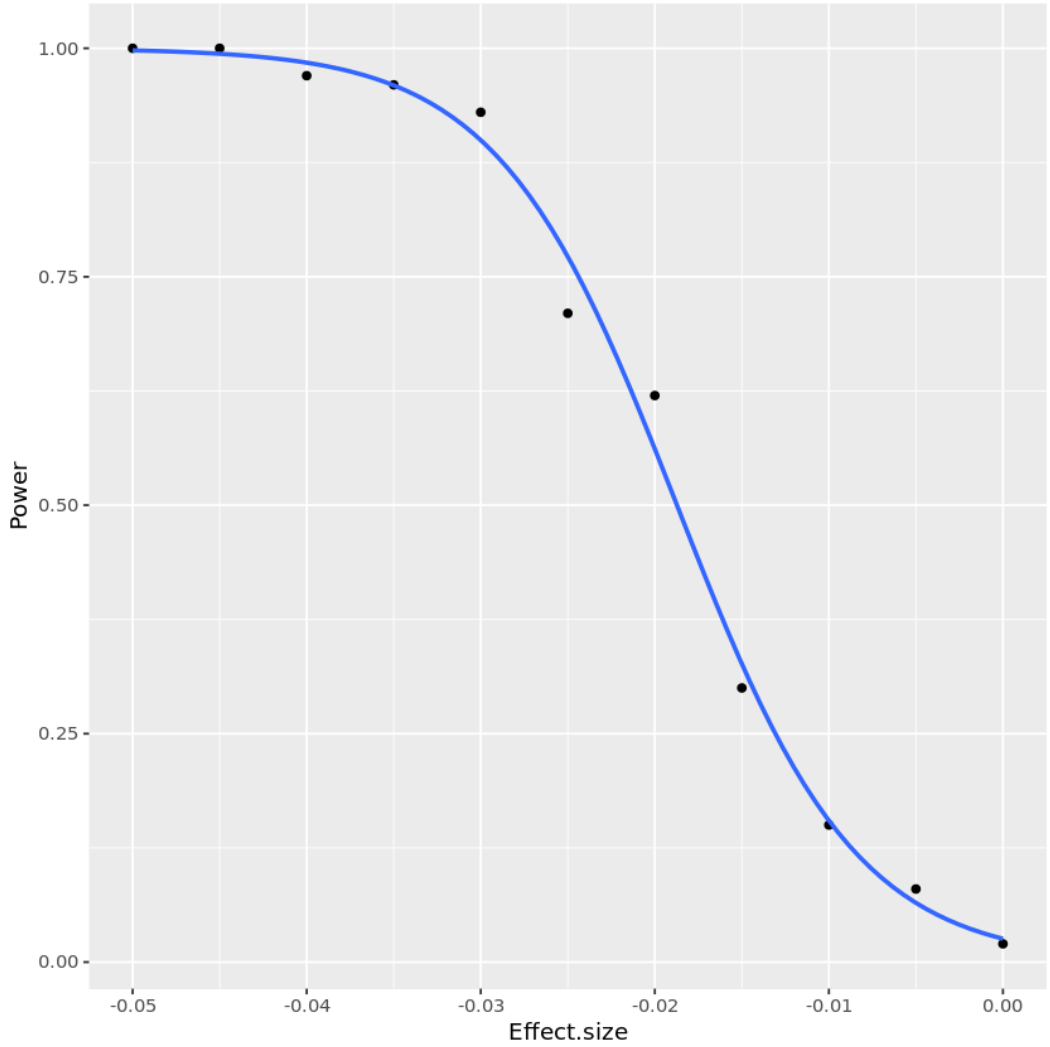
Supplemental Material, Figure S2: Linear Mixed Model (LMM) Best Linear Unbiased Predictions (BLUPs) superimposed on scatter plots for the *pgm1* mutant and control (Col-0) showing root angle versus time.



Supplemental Material, Figure S3: Linear Mixed Model (LMM) Best Linear Unbiased Predictions (BLUPs) superimposed on scatter plots for the *pgm1* mutant and control (Col-0) showing root growth versus time.

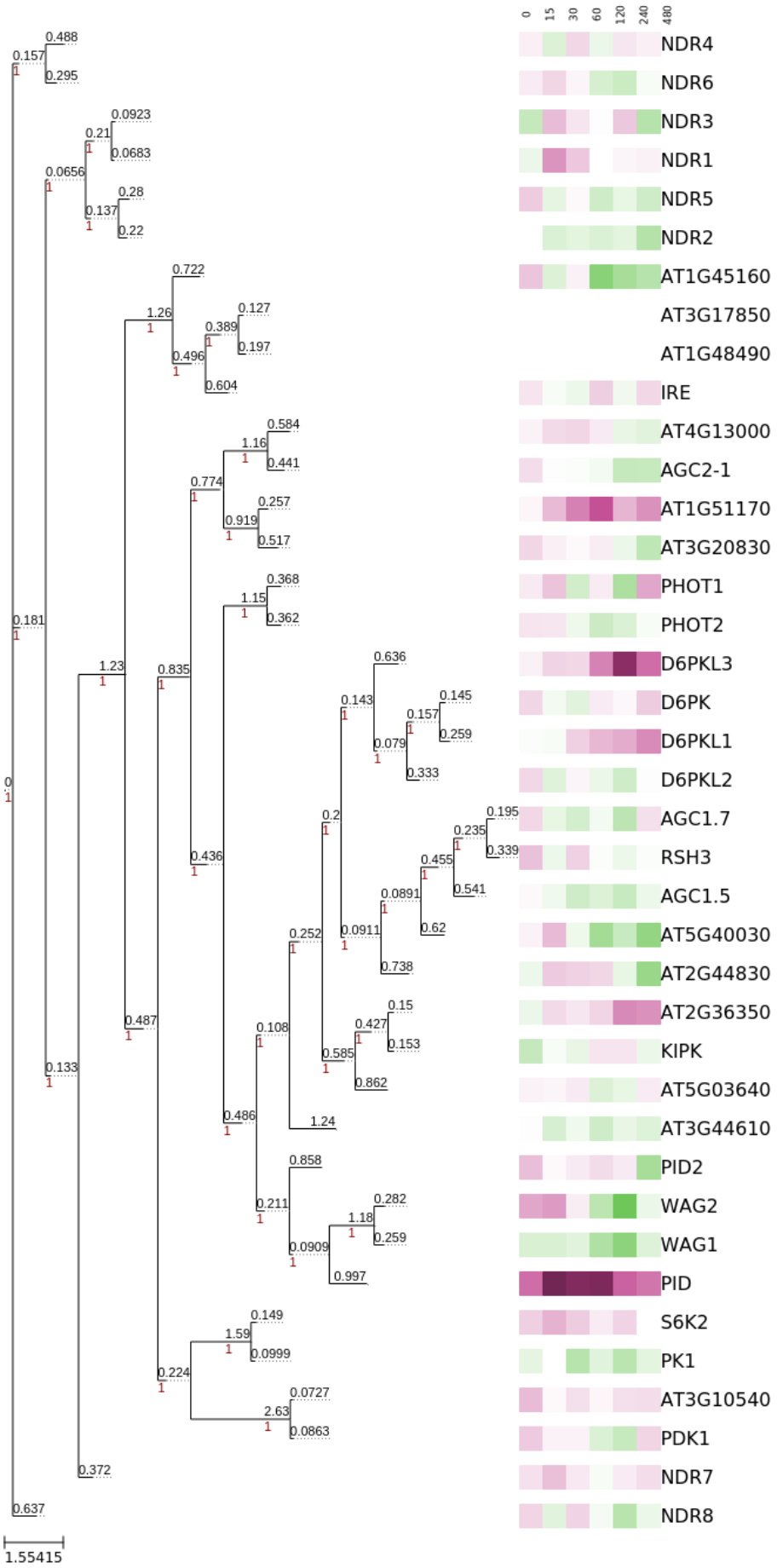


Supplemental Material, Figure S4: Gravitropic kinetics of *pgm1* mutant and control (Col-0) lines analysed using qGAP. Dashed lines give 95% confidence interval based on a linear mixed model.



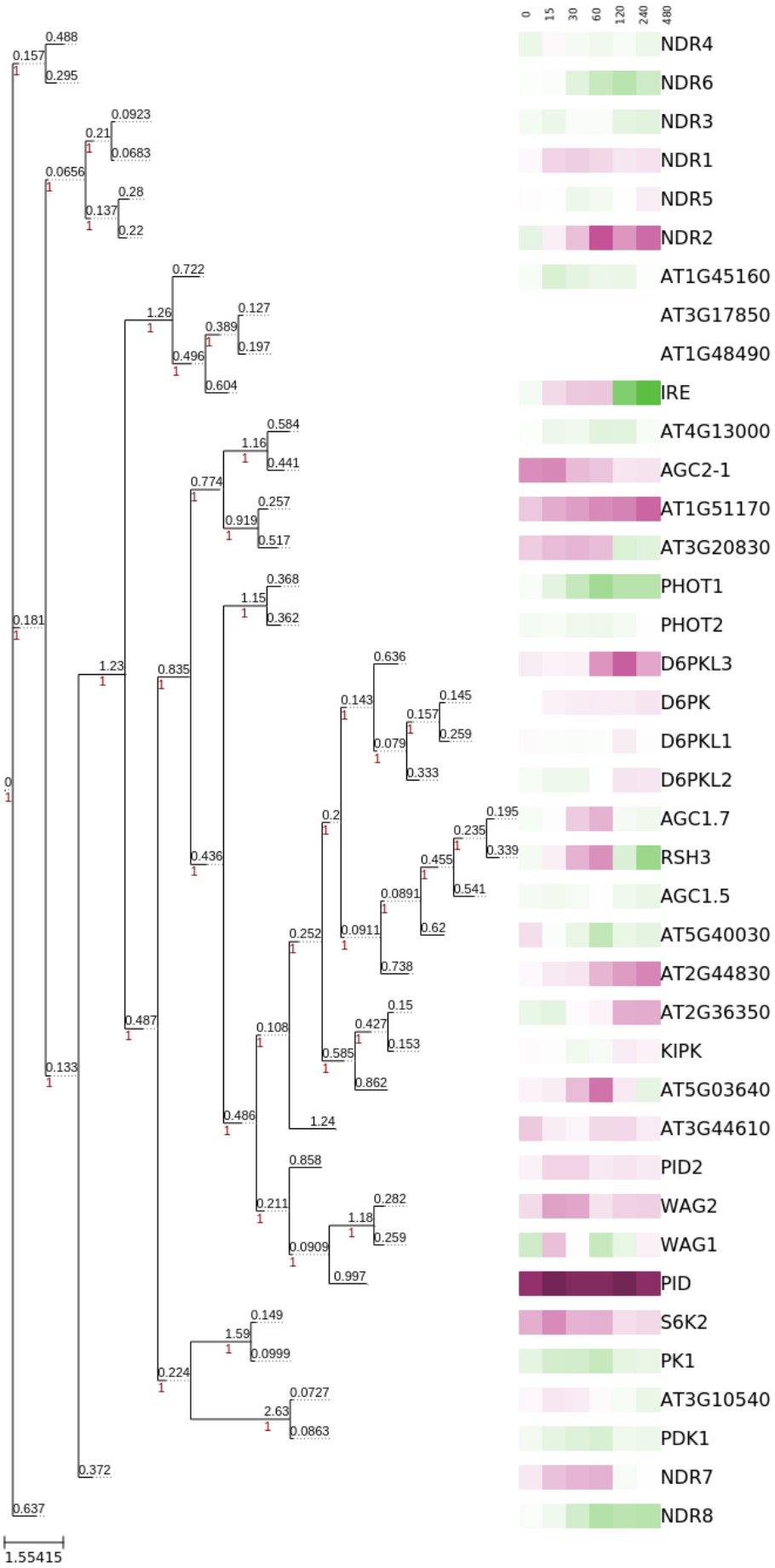
Supplemental Material, Figure S5: Power curve for the mixed model based on the *pgm1* dataset. Power was estimated using simulations, with the Col-0 model as the baseline. The Effect.size at 80% Power is often defined as the Minimal Detectable Difference (MDD). Power analysis simulations were run using simr.

S1.2 Supplemental Figures for Chapter 3: Phylogenetic Analysis to Infer Kinase Function - A Novel Screening Approach



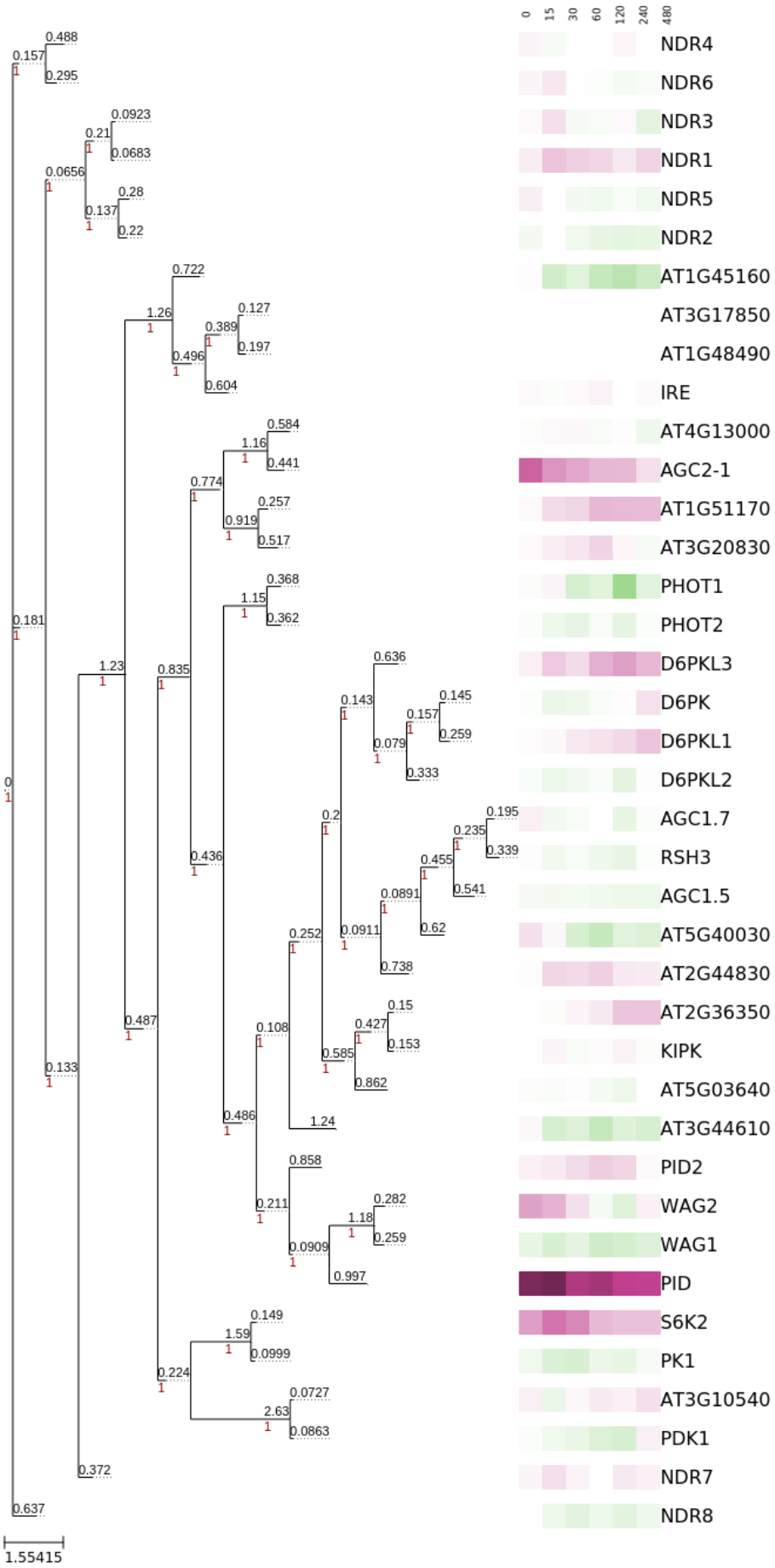
Supplemental Material, Figure S6: *(legend on next page)*

Supplemental Material, Figure S6: Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their ARF7/19-dependent auxin-regulated expression in the root meristem. Auxin expression heatmaps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



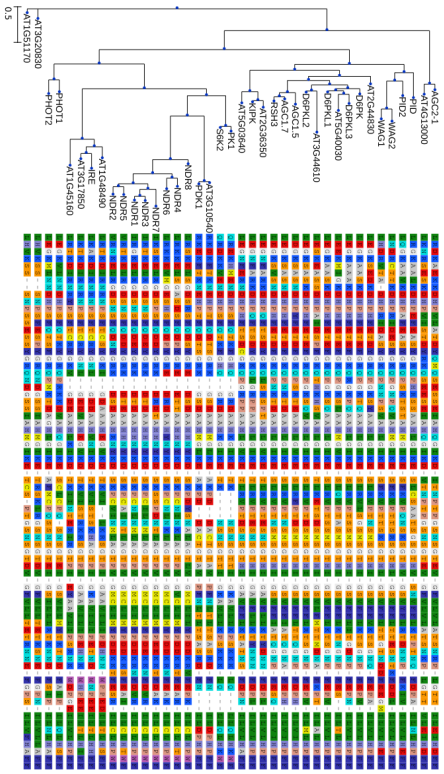
Supplemental Material, Figure S7: *(legend on next page)*

Supplemental Material, Figure S7: Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)

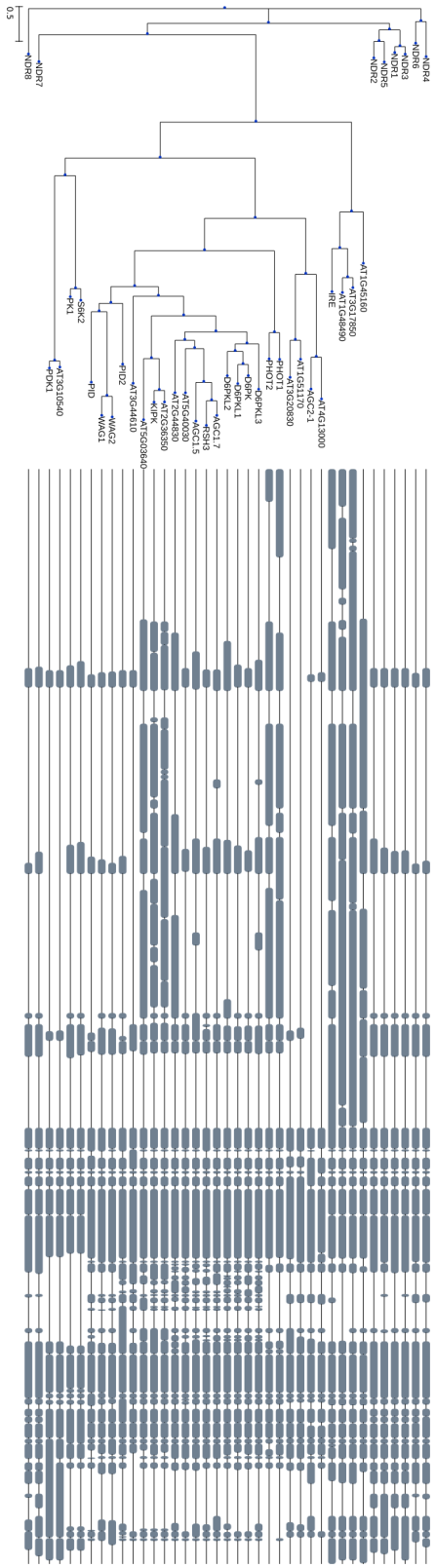


Supplemental Material, Figure S8: *(legend on next page)*

Supplemental Material, Figure S8: Phylogeny (kinase domain) of all *Arabidopsis thaliana* AGC-VIII kinases showing their auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)

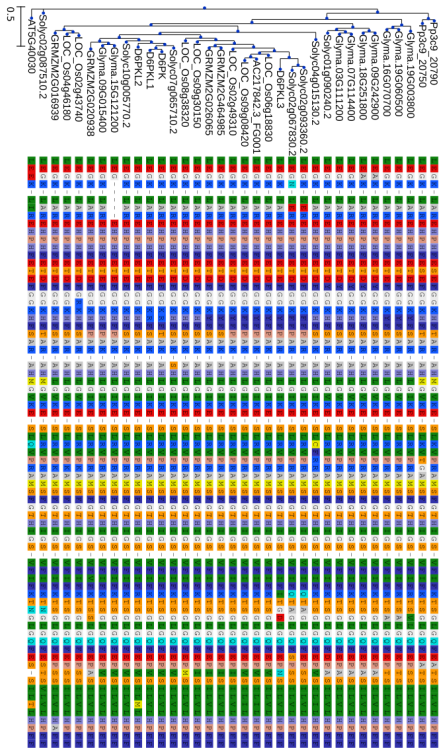


(a) Domains of specificity

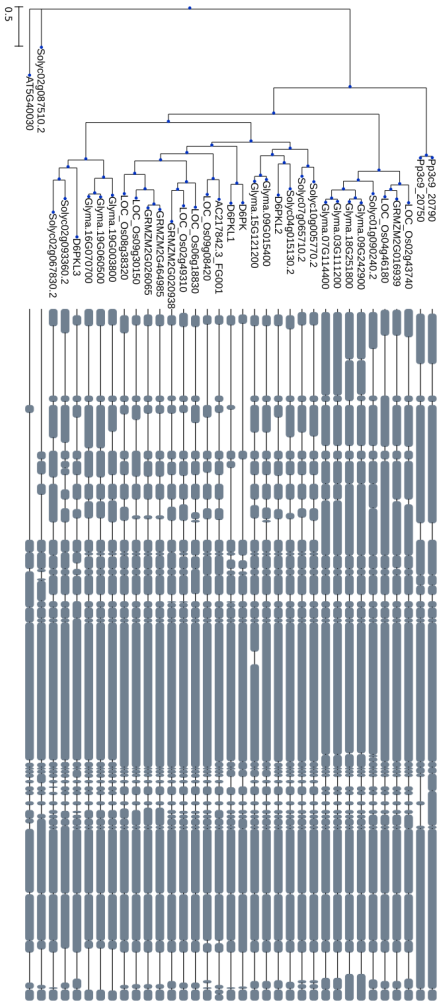


(b) Full sequence

Supplemental Material, Figure S9: (a) DoS-dendrogram and (b) full sequence phylogeny of *Arabidopsis thaliana* AGC-VIII kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5)

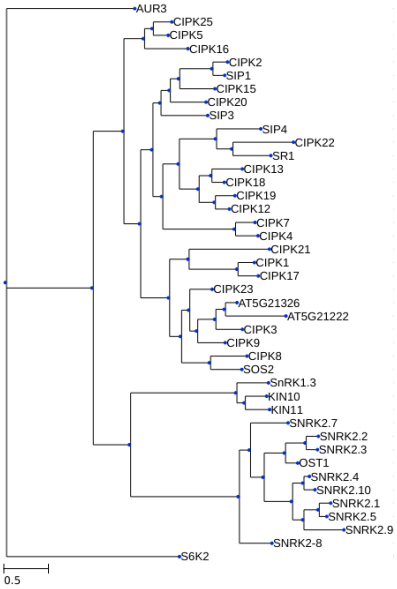


(a) Domains of specificity

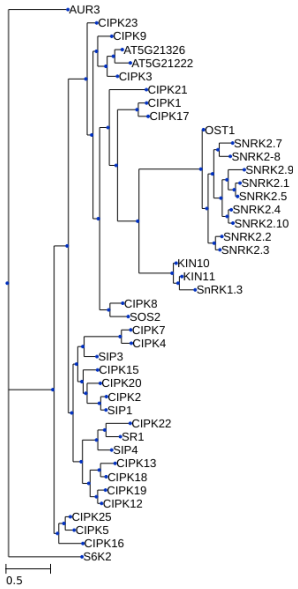


(b) Full sequence

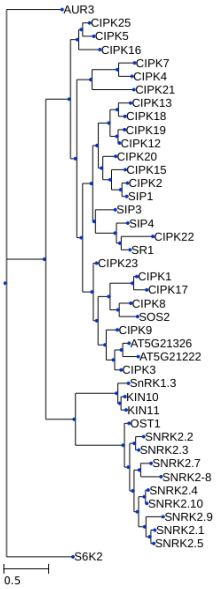
Supplemental Material, Figure S10: (a) DoS-dendrogram and (b) full sequence phylogeny of *Z. mays, A. thaliana, G. max, O. sativa, P. patens* and *S. lycopersicum* D6PK kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5)



(a) Full sequence

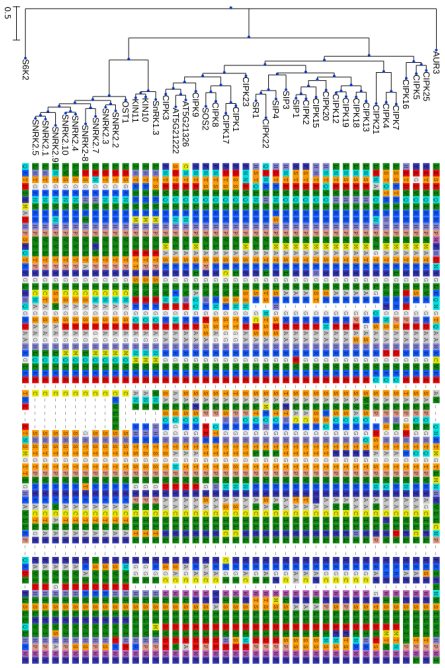


(b) Kinase domain

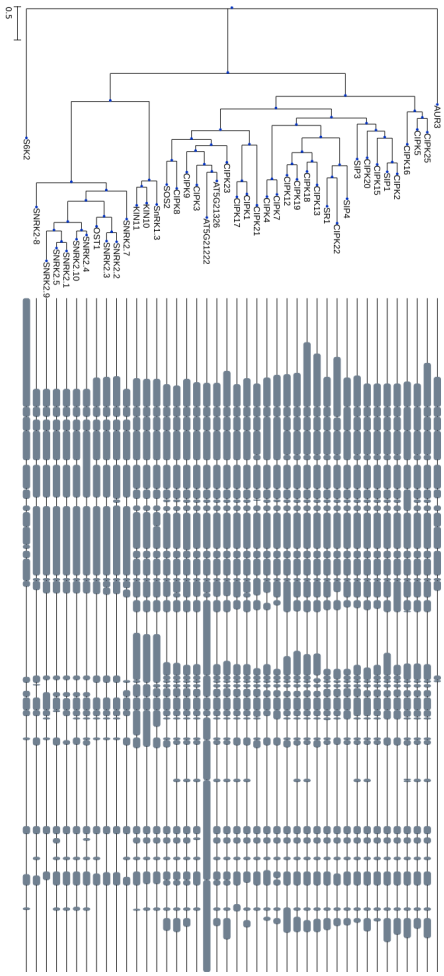


(c) Domains of specificity

Supplemental Material, Figure S11: *Arabidopsis thaliana* CIPK kinase phylogenies based on different sub-sequences of the full protein sequence (a). Alignments generated by KSselect. Trees constructed with RaxML and visualised using KSviz (see Section 2.4.5)

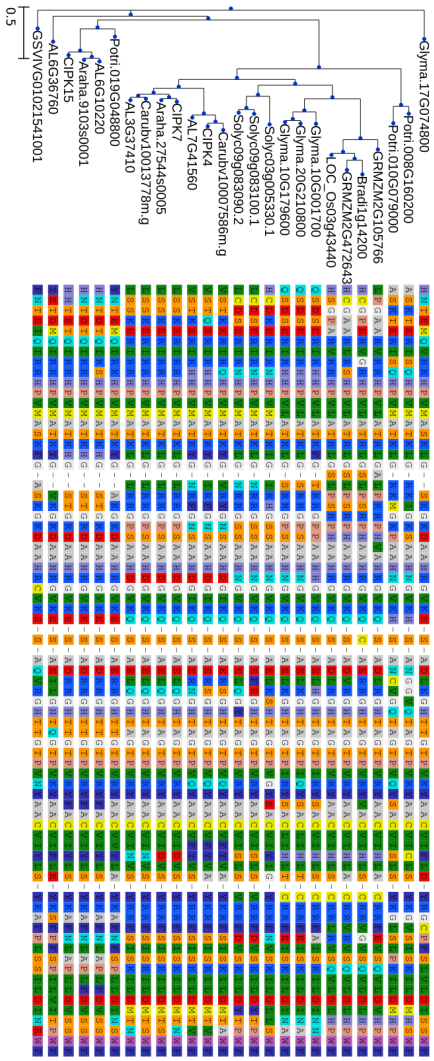


(a) Domains of specificity

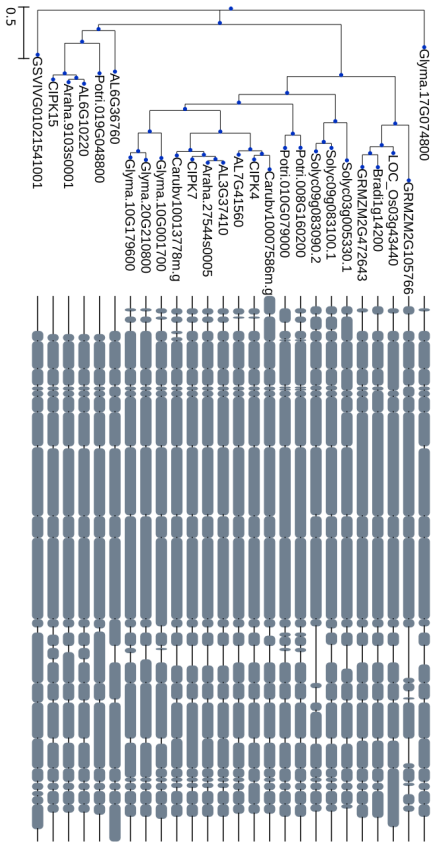


(b) Full sequence

Supplemental Material, Figure S12: (a) DoS-dendrogram and (b) full sequence phylogeny of *Arabidopsis thaliana* CIPK kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5)

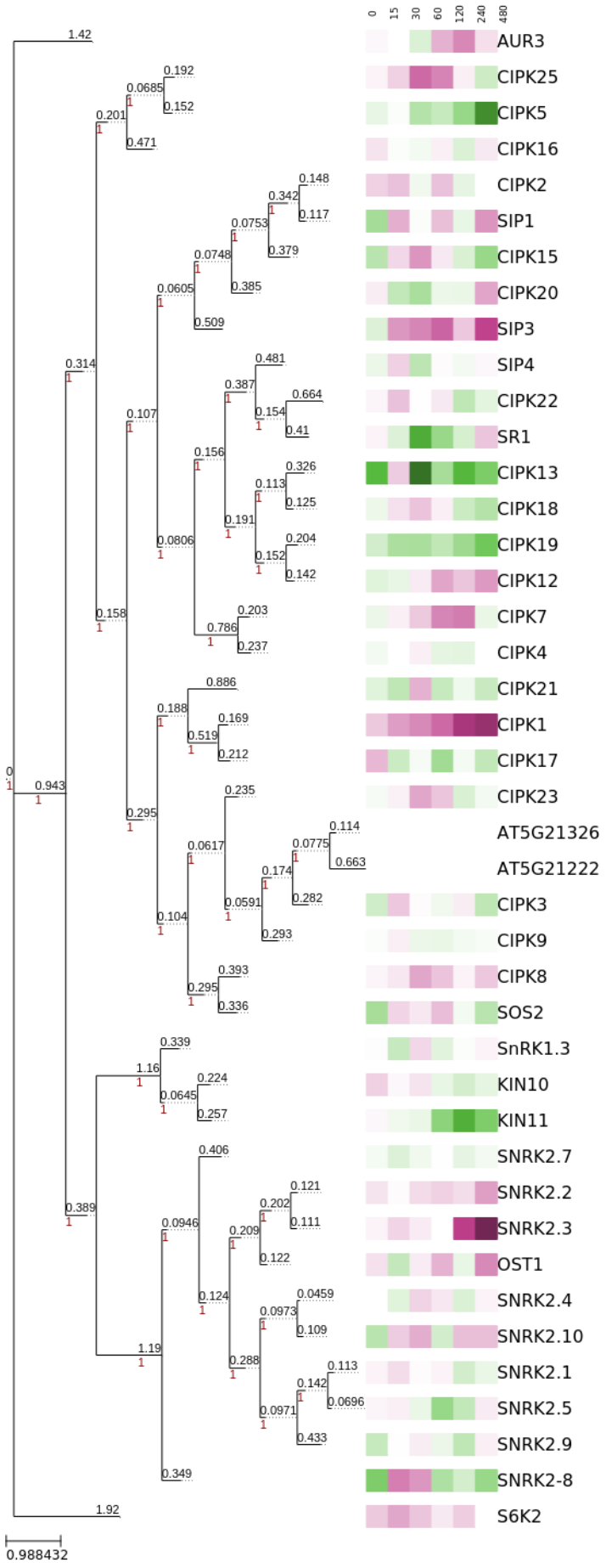


(a) Domains of specificity



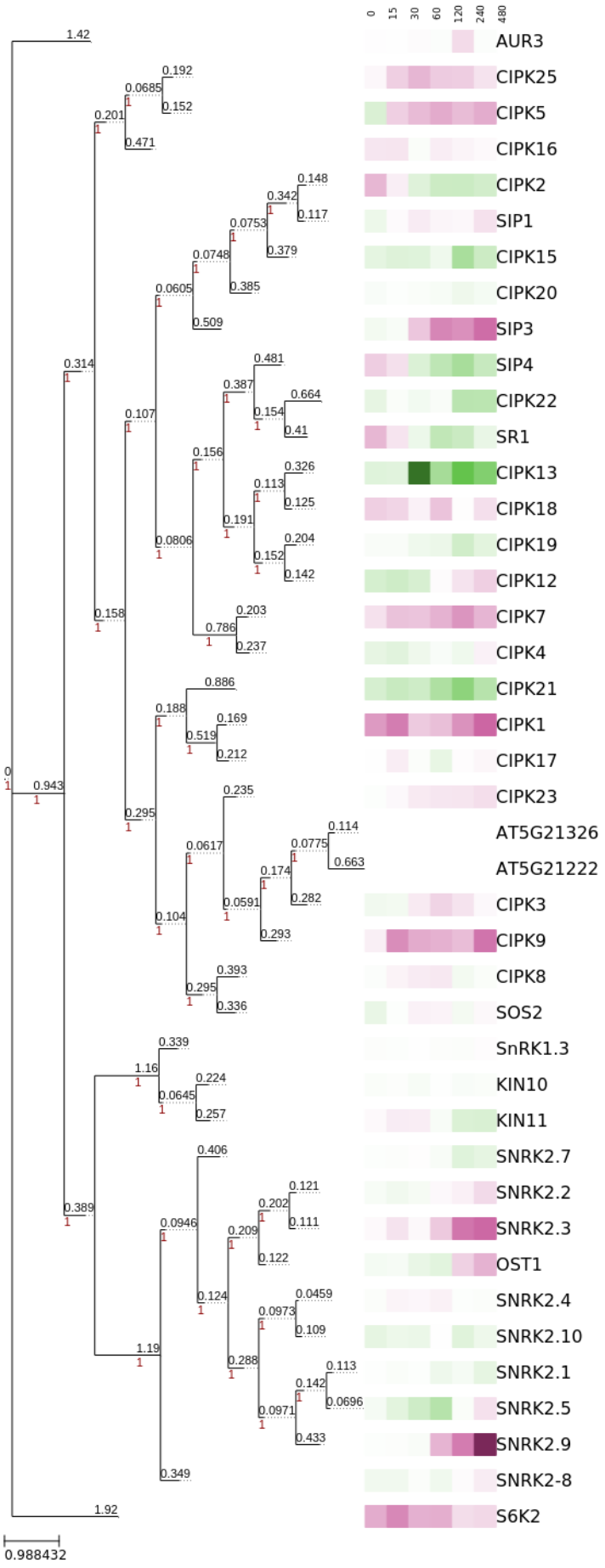
(b) Full sequence

Supplemental Material, Figure S13: (a) DoS-dendrogram and (b) full sequence phylogeny of *S. moellendorffii, Z. mays, A. lyrata, A. halleri, A. thaliana, B. distachyon, C. rubella, G. max, V. vinifera, O. sativa, P. trichocarpa, P. patens* and *S. lycopersicum* CIPK4/7 kinases with sequence detail (DoS) and sequence schematic (full sequence). Generated using KSviz (see Section 2.4.5)



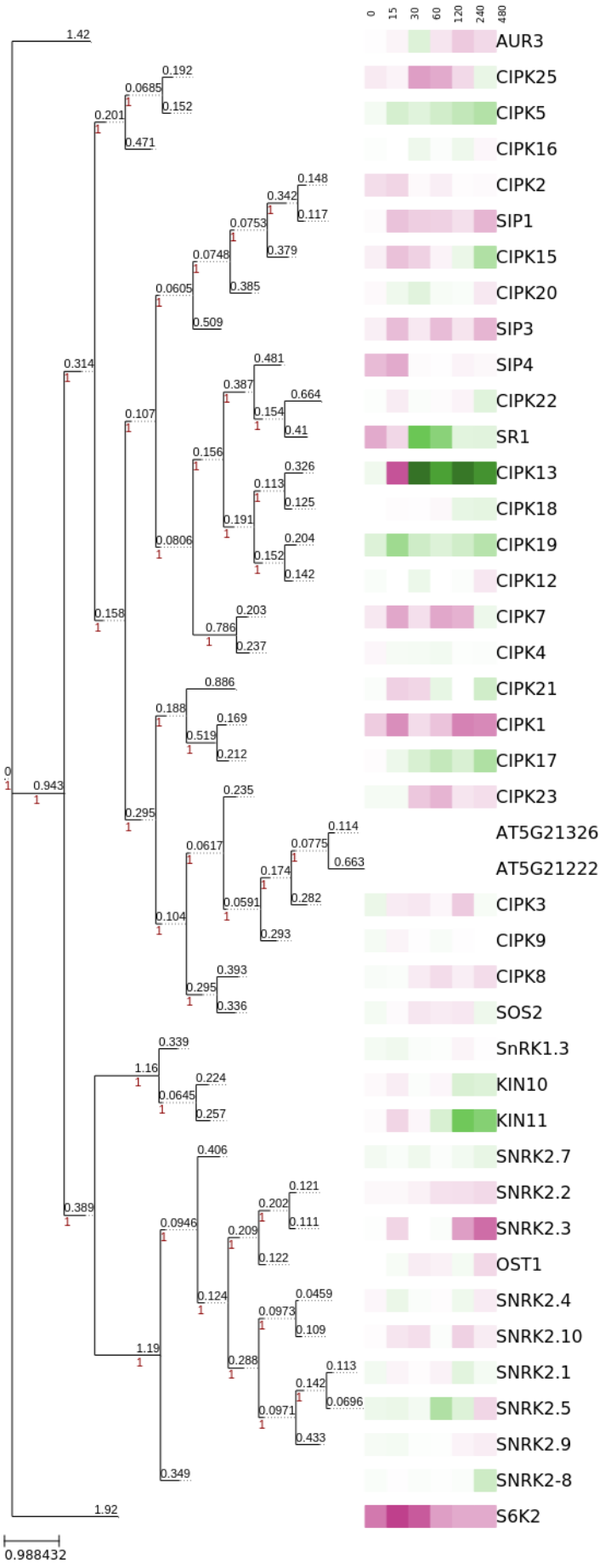
Supplemental Material, Figure S14: *(legend on next page)*

Supplemental Material, Figure S14: Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their ARF7/19-dependent auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



Supplemental Material, Figure S15: *(legend on next page)*

Supplemental Material, Figure S15: Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their auxin-regulated expression in the root elongation zone. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



Supplemental Material, Figure S16: *(legend on next page)*

Supplemental Material, Figure S16: Phylogeny (kinase domain) of all *Arabidopsis thaliana* CIPK kinases showing their auxin-regulated expression in the root meristem. Auxin expression heat-maps are based on CPIB micro-array (E-MTAB-1354, processed by M. Wilson). Green indicates up-regulation and purple down-regulation. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)

S1.3 Supplemental Figures for Chapter 4: Characterisation and Phenotyping of D6PKL3

Table S5: Summary of linear mixed model of root growth data for 3*standard deviation filtered "D6PKL3init"-dataset

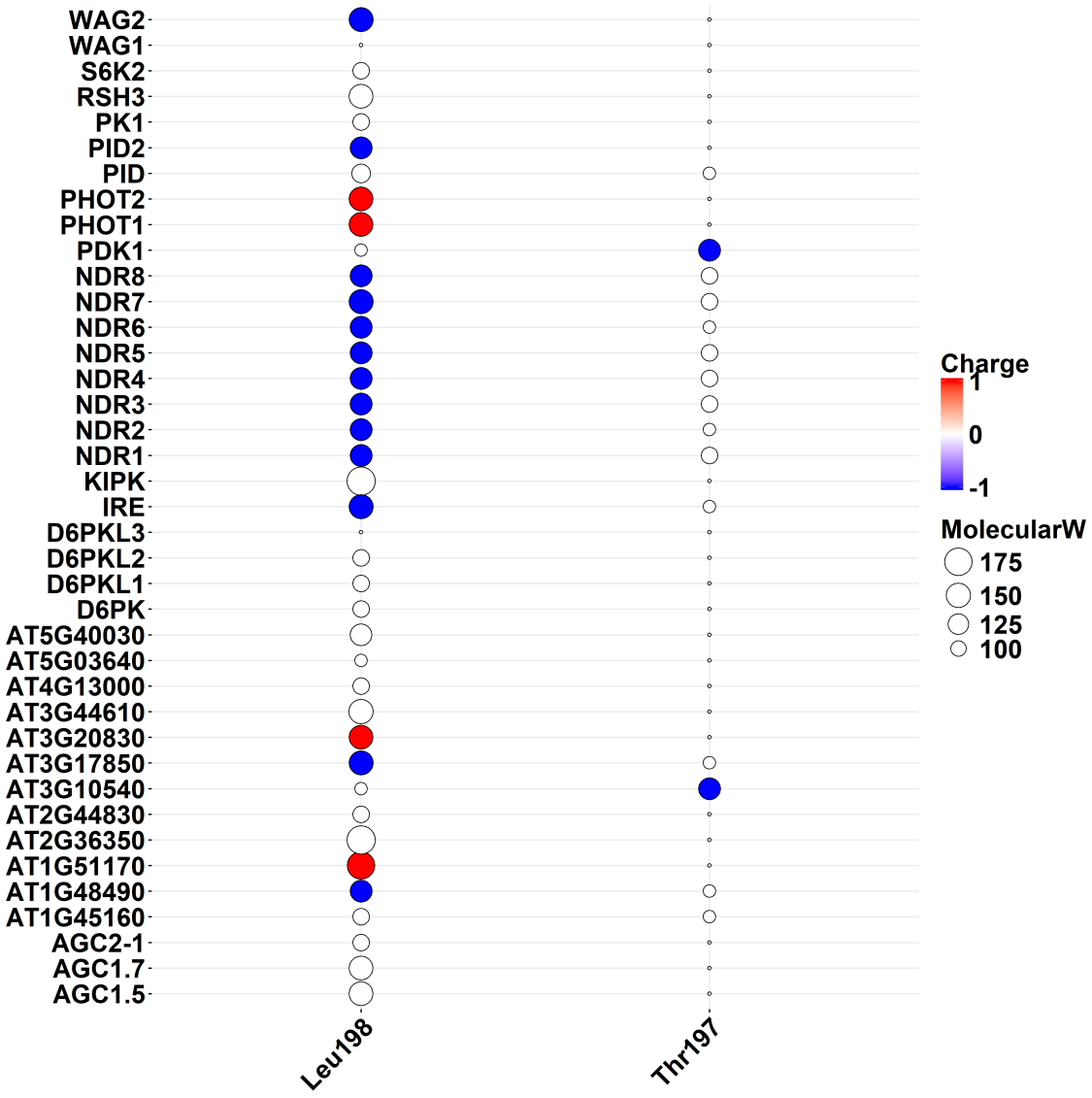
	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	0.02	1.58	95.48	0.01	0.99
LineCol-0b	-9.00	2.19	94.15	-4.11	0.00
Lined6pk012	-4.77	2.06	95.25	-2.31	0.02
Lined6pkl3	-3.77	2.04	94.70	-1.84	0.07
Time	0.39	0.02	94.40	24.24	0.00
LineCol-0b:Time	0.01	0.02	94.11	0.59	0.56
Lined6pk012:Time	0.04	0.02	94.32	1.97	0.05
Lined6pkl3:Time	0.05	0.02	94.20	2.19	0.03

Table S6: Summary of linear mixed model of gravitropic kinetics for full "D6PKL3init"-dataset

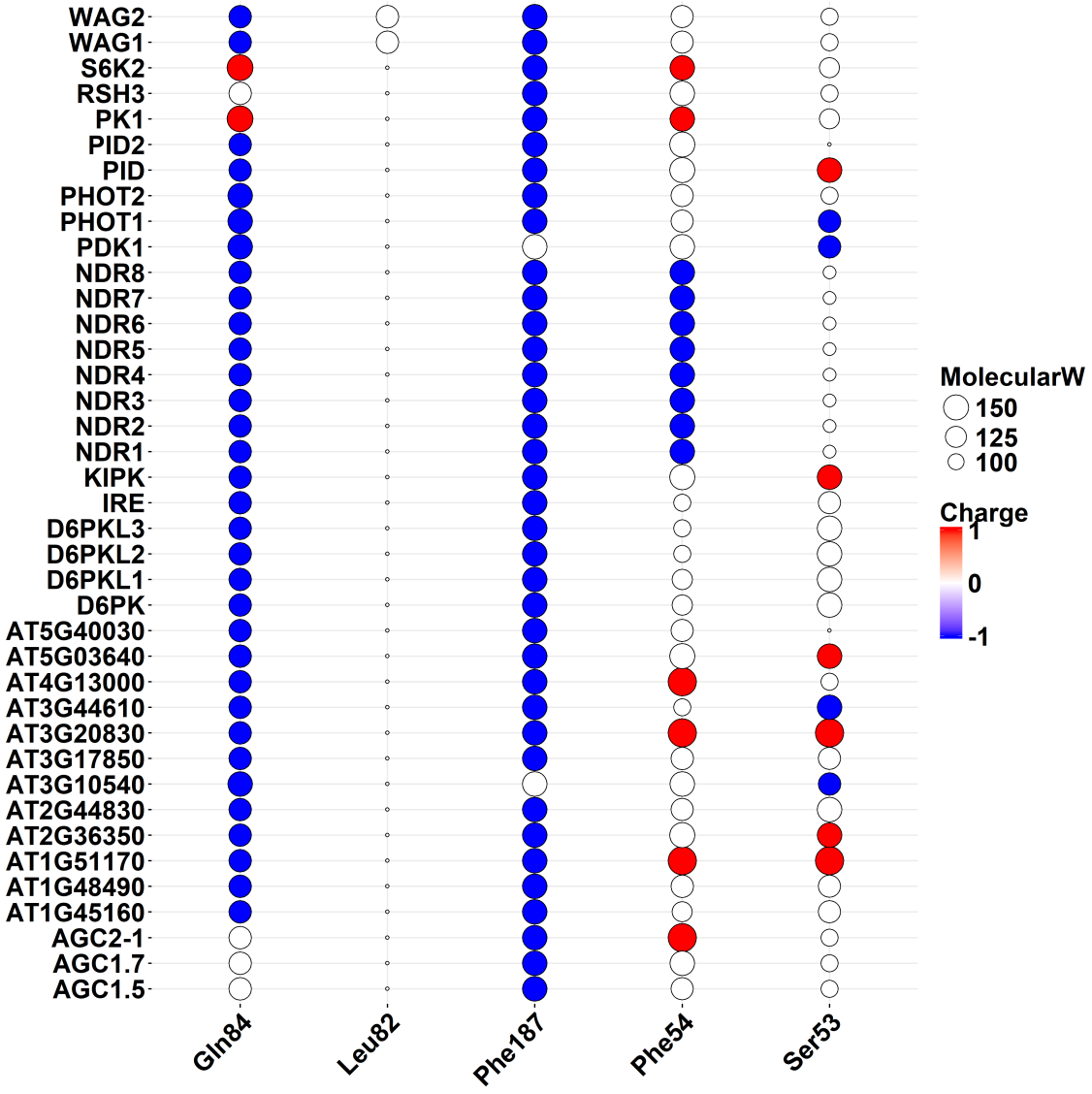
	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	92.15	0.96	96.47	95.66	0.00
LineCol-0b	-1.26	1.33	95.43	-0.95	0.34
Lined6pk012	-2.56	1.26	96.13	-2.03	0.05
Lined6pkl3	1.12	1.25	95.74	0.90	0.37
Time	-0.14	0.01	95.71	-25.54	0.00
LineCol-0b:Time	0.00	0.01	95.17	0.36	0.72
Lined6pk012:Time	-0.03	0.01	95.53	-4.25	0.00
Lined6pkl3:Time	0.01	0.01	95.30	0.82	0.41

Table S7: Summary of linear mixed model of gravitropic kinetics for 1.5*standard deviation filtered "D6PKL3init"-dataset

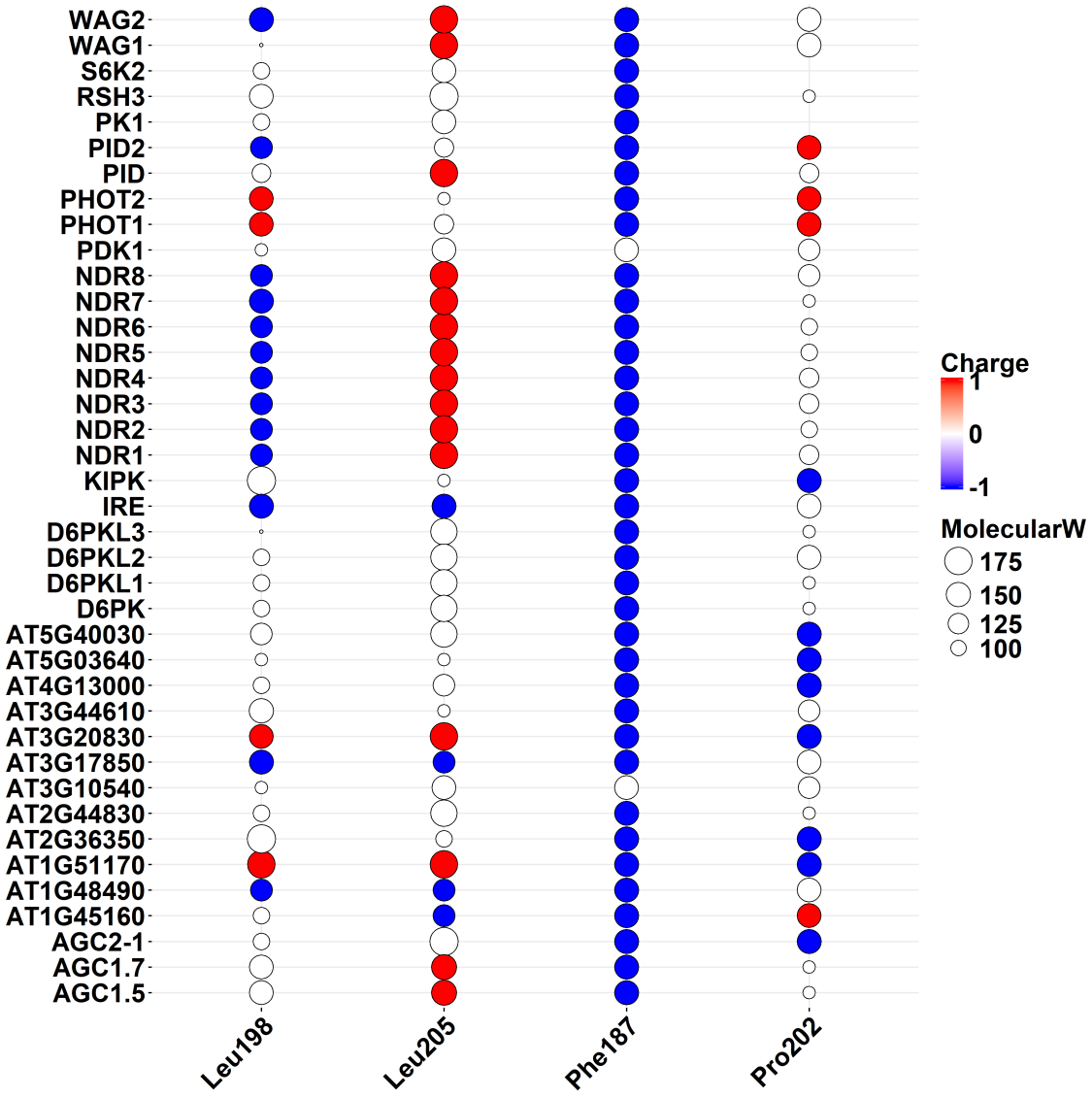
	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	92.15	0.95	84.30	96.60	0.00
LineCol-0b	-1.03	1.33	83.37	-0.78	0.44
Lined6pk012	-1.85	1.29	84.13	-1.43	0.16
Lined6pkl3	1.92	1.31	83.64	1.46	0.15
Time	-0.14	0.00	84.16	-34.12	0.00
LineCol-0b:Time	-0.00	0.01	83.39	-0.74	0.46
Lined6pk012:Time	-0.03	0.01	84.03	-5.69	0.00
Lined6pkl3:Time	0.00	0.01	83.56	0.52	0.61



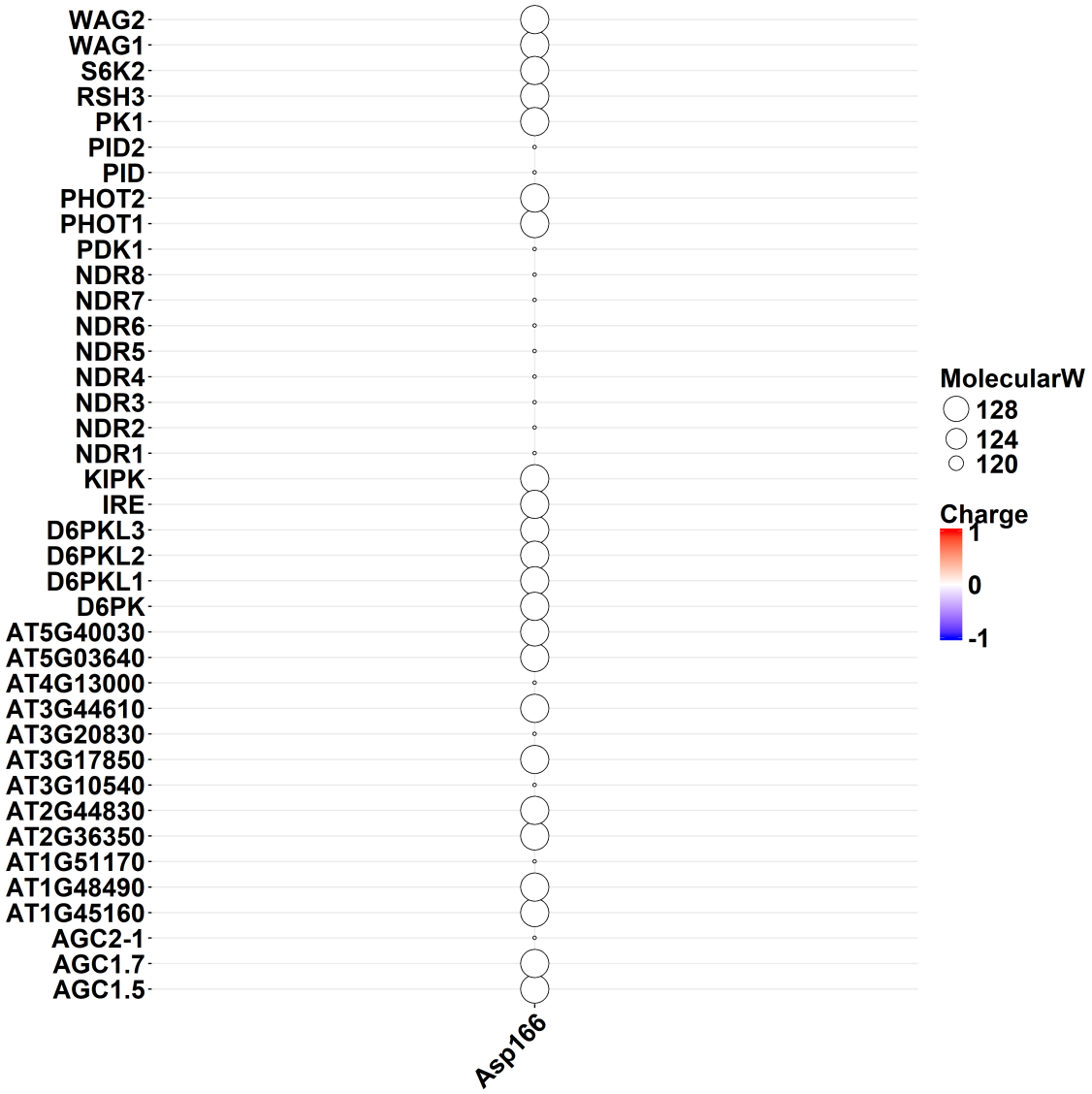
Supplemental Material, Figure S17: Balloon plot of the P+3 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



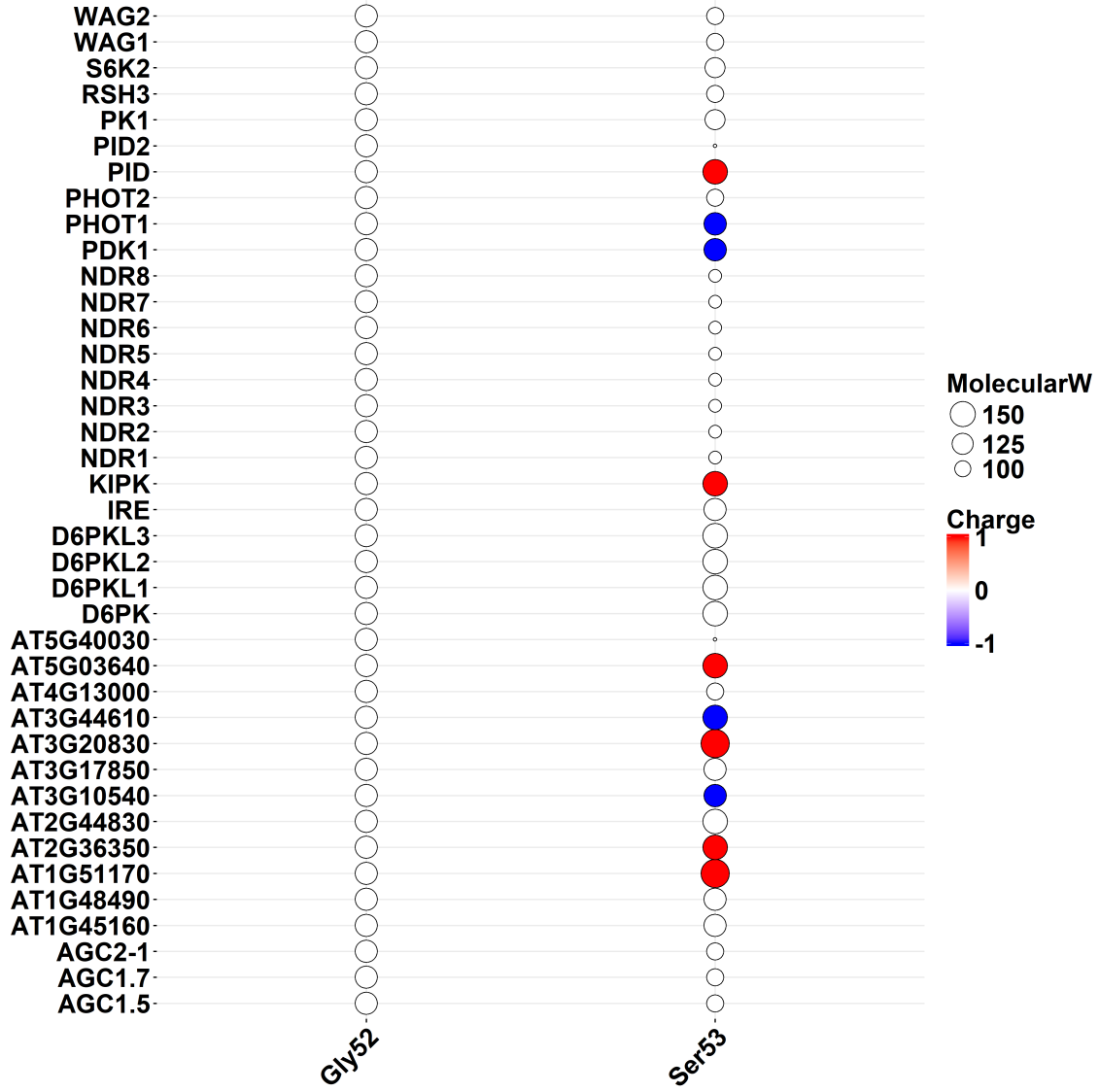
Supplemental Material, Figure S18: Balloon plot of the P+2 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



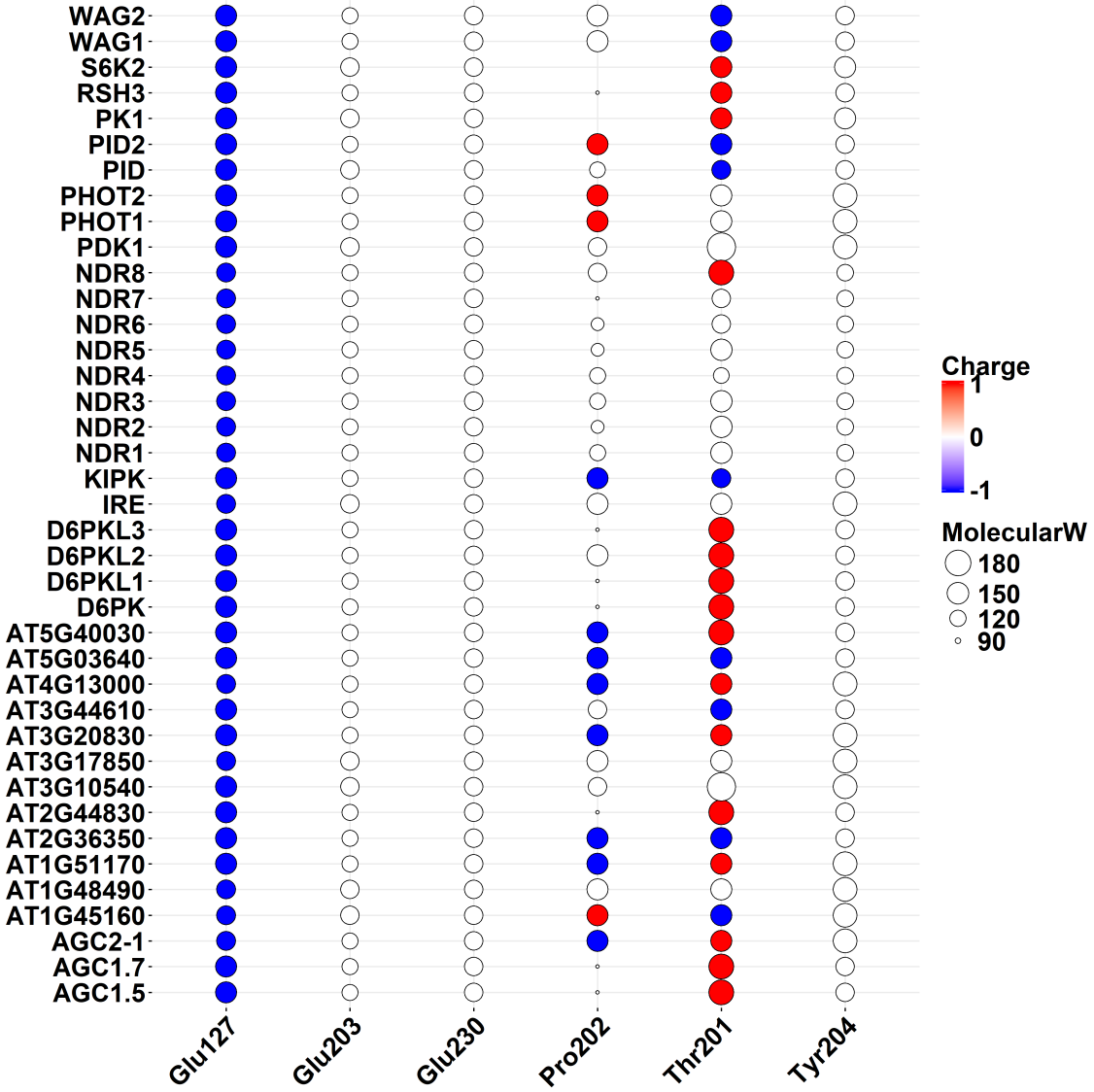
Supplemental Material, Figure S19: Balloon plot of the P+1 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



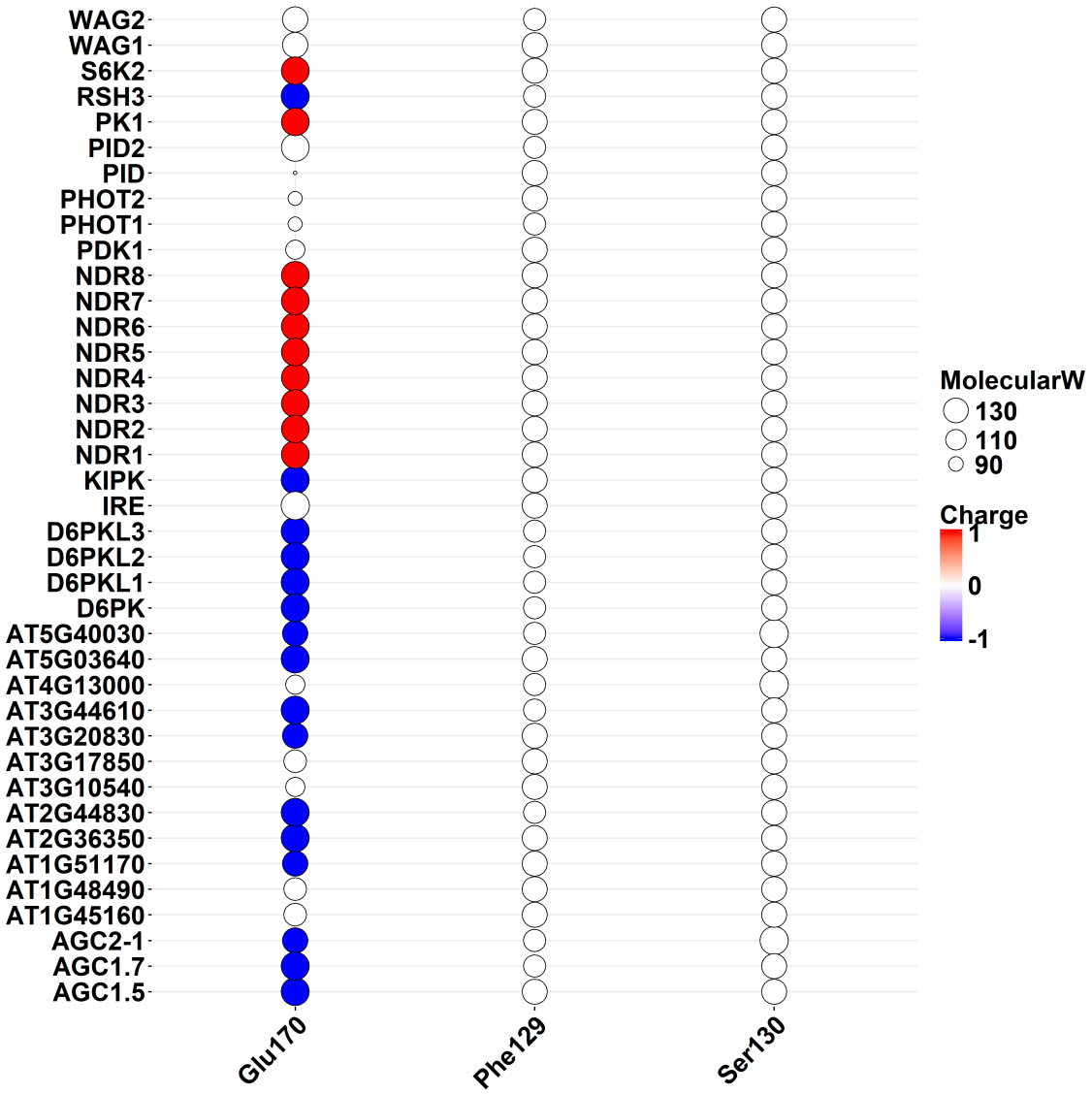
Supplemental Material, Figure S20: Balloon plot of the P-0 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



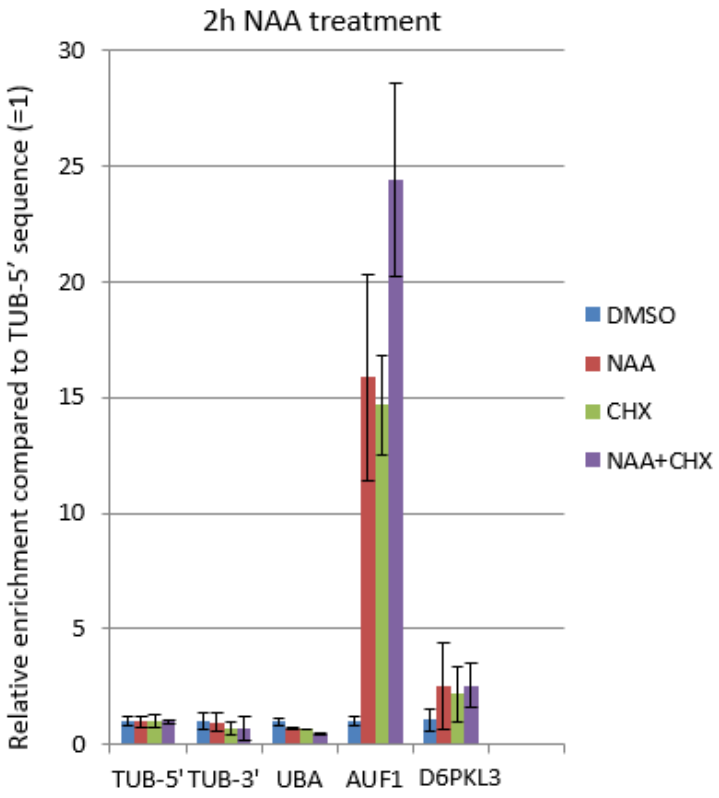
Supplemental Material, Figure S21: Balloon plot of the P-1 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



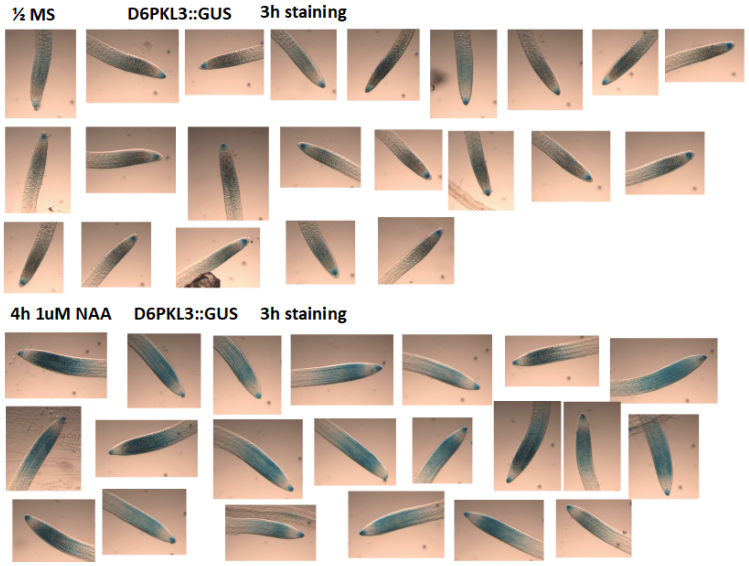
Supplemental Material, Figure S22: Balloon plot of the P-2 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



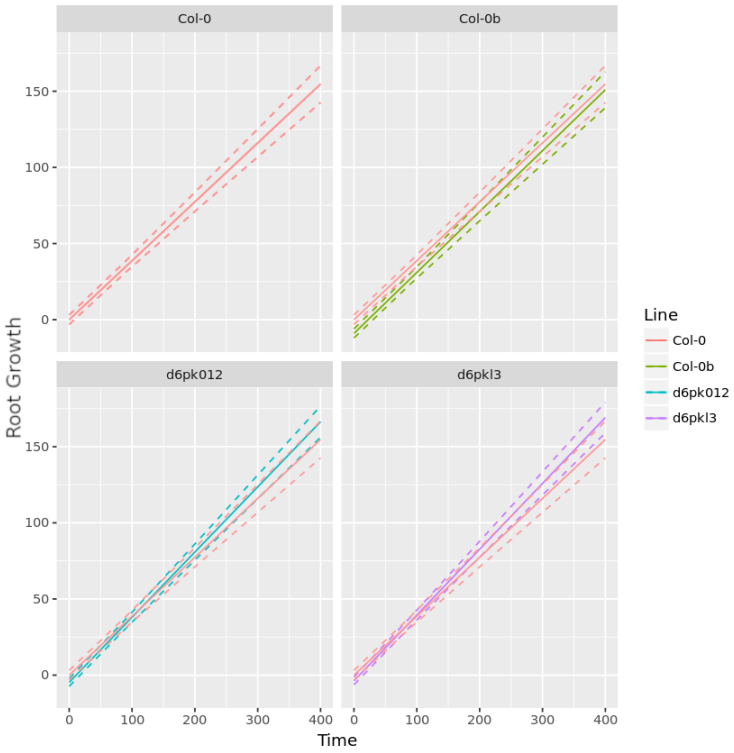
Supplemental Material, Figure S23: Balloon plot of the P-3 Phospho-site Residue Binding Pocket of all AGC-VIII kinases. Colour indicates charge and circle diameter indicates molecular weight of residue. Positions are given by the corresponding amino acid in PKA. Generated using PRBPviz (see Section 2.4.4)



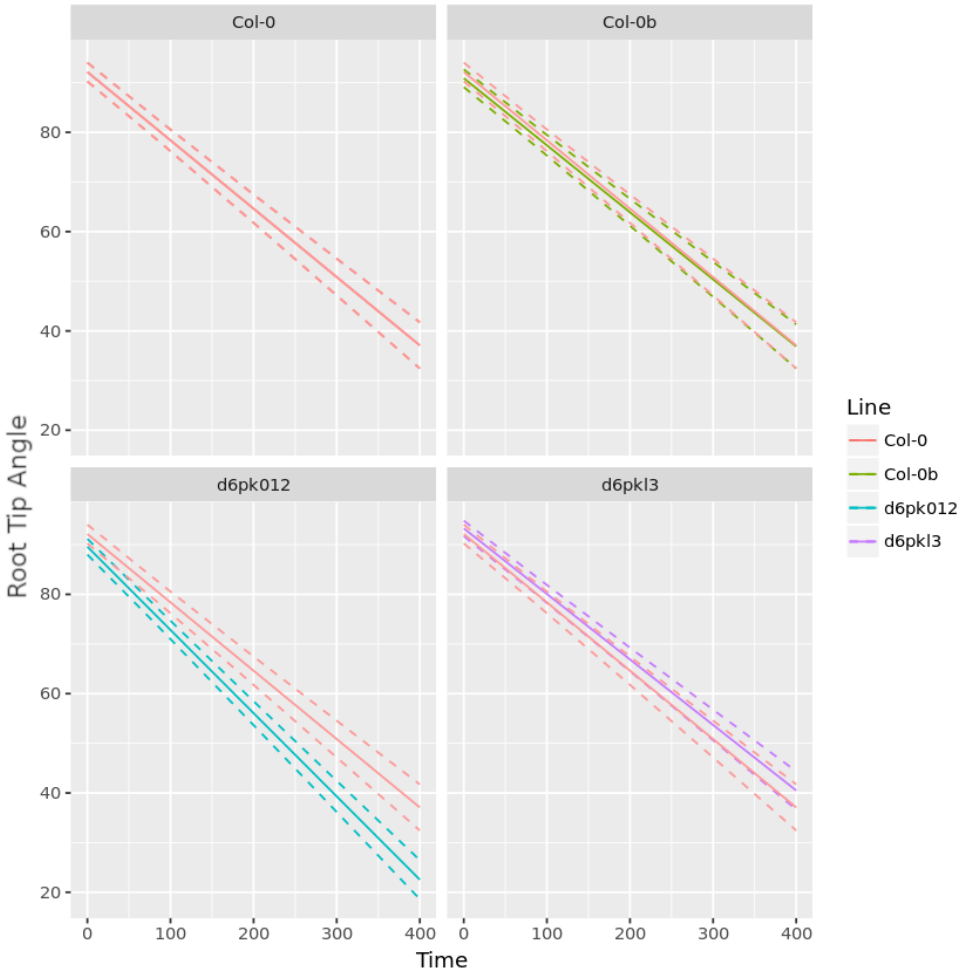
Supplemental Material, Figure S24: Auxin-induction of D6PKL3 in the root as determined by qPCR. TUB and UBA are used as technical controls. AUF1 is used as a positive control for auxin-induction.



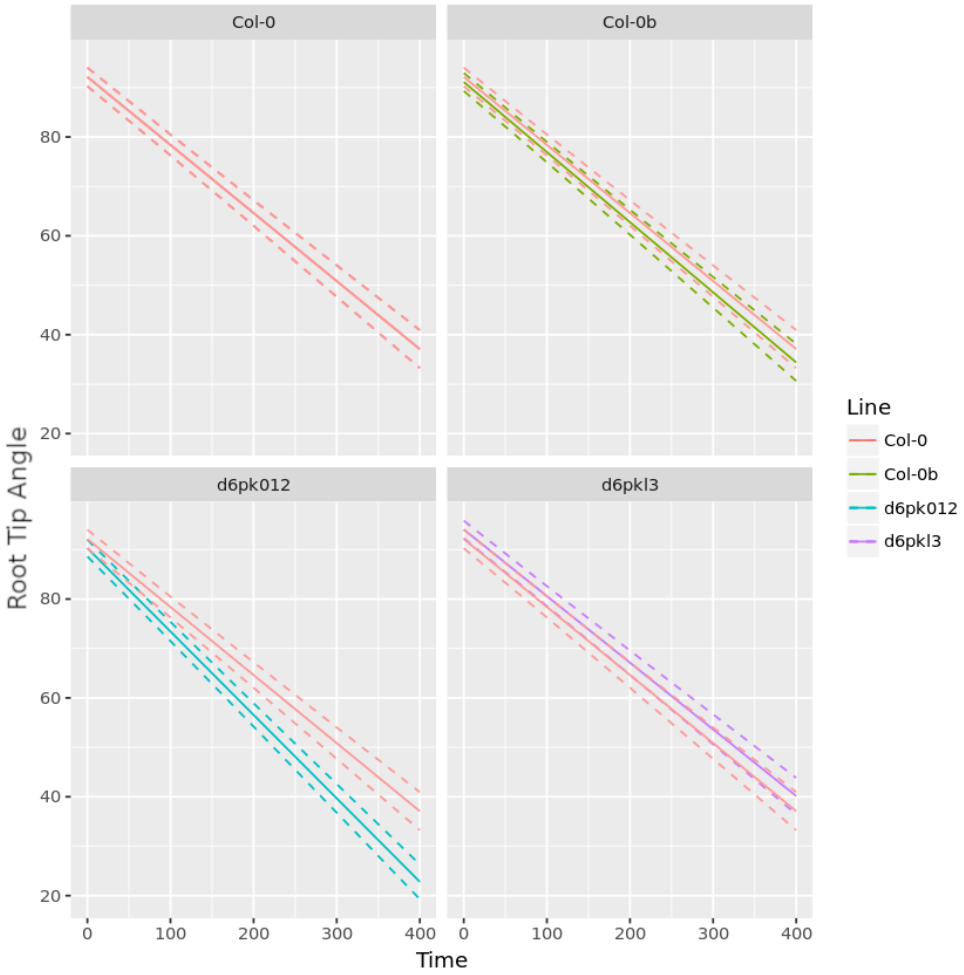
Supplemental Material, Figure S25: Additional images of GUS-stained roots after 4 hours of auxin treatment (bottom) and under control conditions (top). Under control conditions, D6PKL3 is mainly expressed in the columella cells. After 4 hours of auxin treatment, D6PKL3 expression is shifted to the peripheral columella cells and is additionally expressed in the basal meristem, as shown by promoter GUS fusion.



Supplemental Material, Figure S26: Root growth rate of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and growth in pixel.



Supplemental Material, Figure S27: Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The full "D6PKL3init"-dataset was used. Time is given in minutes and angle in degree.

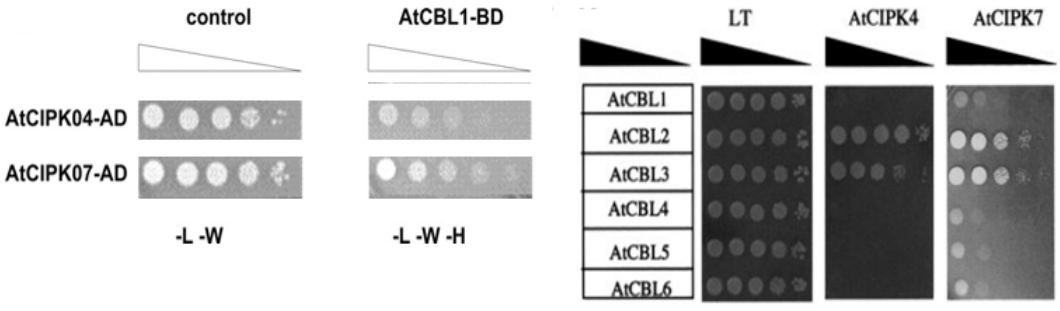


Supplemental Material, Figure S28: Gravitropic kinetics of *d6pkl3* mutant, *d6pk d6pkl1 d6pkl2* mutant, and two Col-0 lines analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 1.5*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and angle in degree.

S1.4 Supplemental Figures for Chapter 5: Characterisation and Phenotyping of CIPK4 and CIPK7

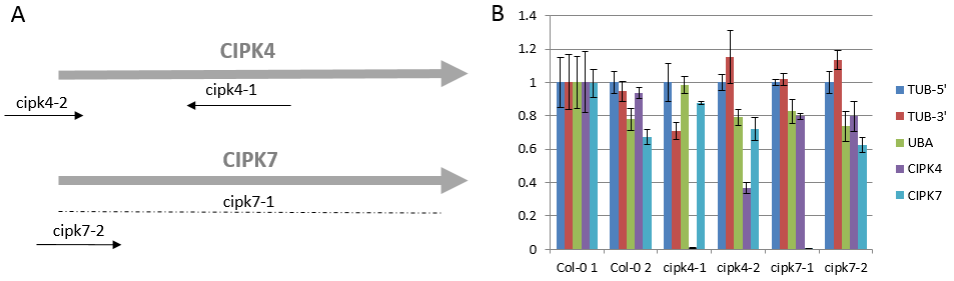
Table S8: Summary of growth rate linear mixed model of full CIPK-dataset.

	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	-7.88	2.67	85.73	-2.95	0.00
LineCipk4-1	2.20	3.68	86.27	0.60	0.55
LineCipk4-2	11.31	3.55	86.45	3.18	0.00
LineCipk7-1	7.05	3.56	86.99	1.98	0.05
LineCipk7-2	8.47	3.48	86.37	2.43	0.02
Time	0.36	0.02	88.74	15.91	0.00
LineCipk4-1:Time	-0.04	0.03	88.96	-1.20	0.23
LineCipk4-2:Time	-0.11	0.03	88.89	-3.74	0.00
LineCipk7-1:Time	0.04	0.03	89.51	1.24	0.22
LineCipk7-2:Time	0.04	0.03	89.13	1.26	0.21



(a) (b)

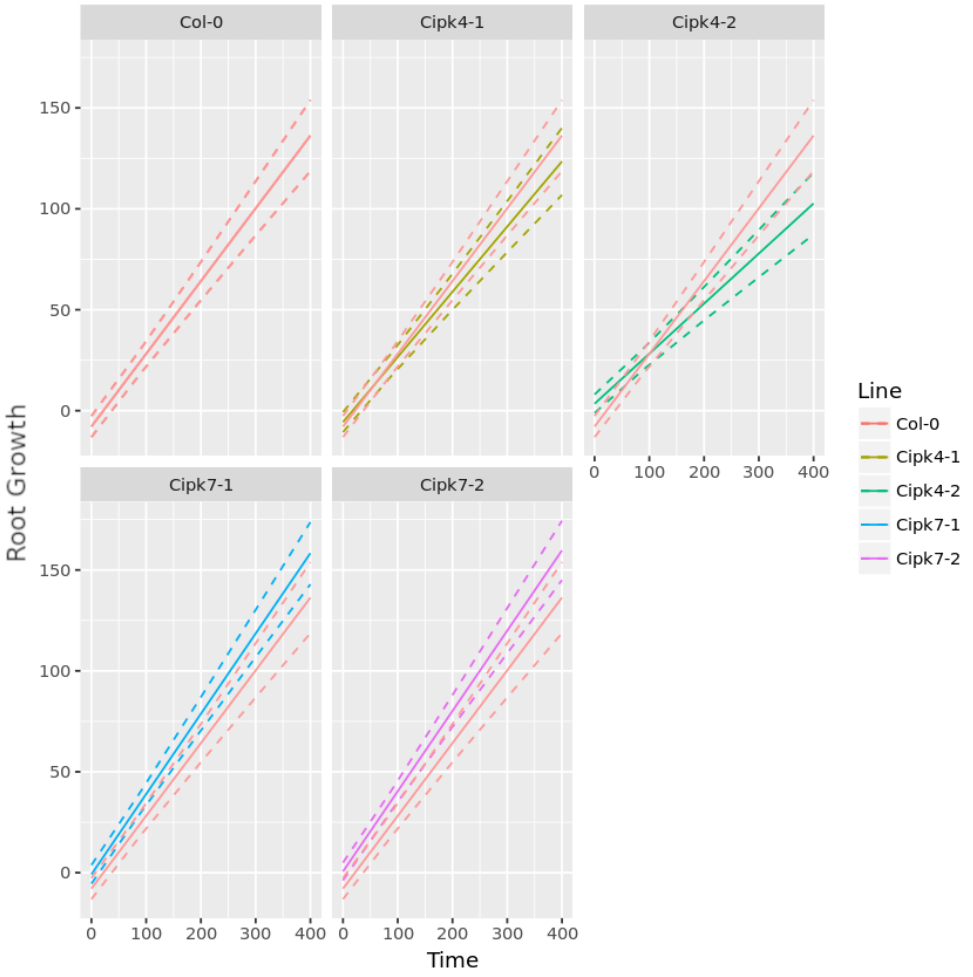
Supplemental Material, Figure S29: Yeast two hybrid interaction assays for CIPKs and CBL proteins. Adapted from Albret *et al.* (2001) and Kolukisagolu *et al.* (2004).



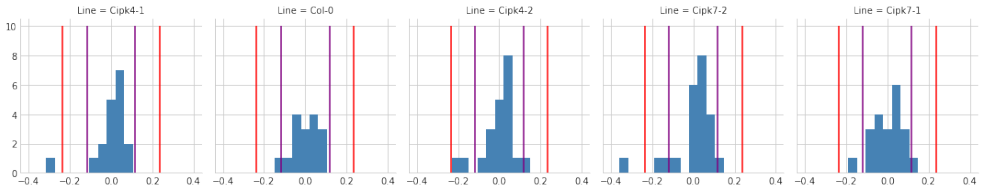
Supplemental Material, Figure S30: Isolation of *cipk4* and *cipk7* knock-out mutants. (A) Location of the two T-DNA inserts selected for each gene. (B) Quantitative reverse transcription PCR results for root samples of the selected T-DNA insert lines and two Col-0 lines. TUB-5 sequence was used as standard. TUB-3 sequence included as control for RNA quality (undigested). UBA sequence served as negative control. The *cipk4-1* and *cipk7-1* mutants were identified as knock-out mutants.

Table S9: Summary of linear mixed model of gravitropic kinetic data for 3*standard deviation filtered CIPK-dataset.

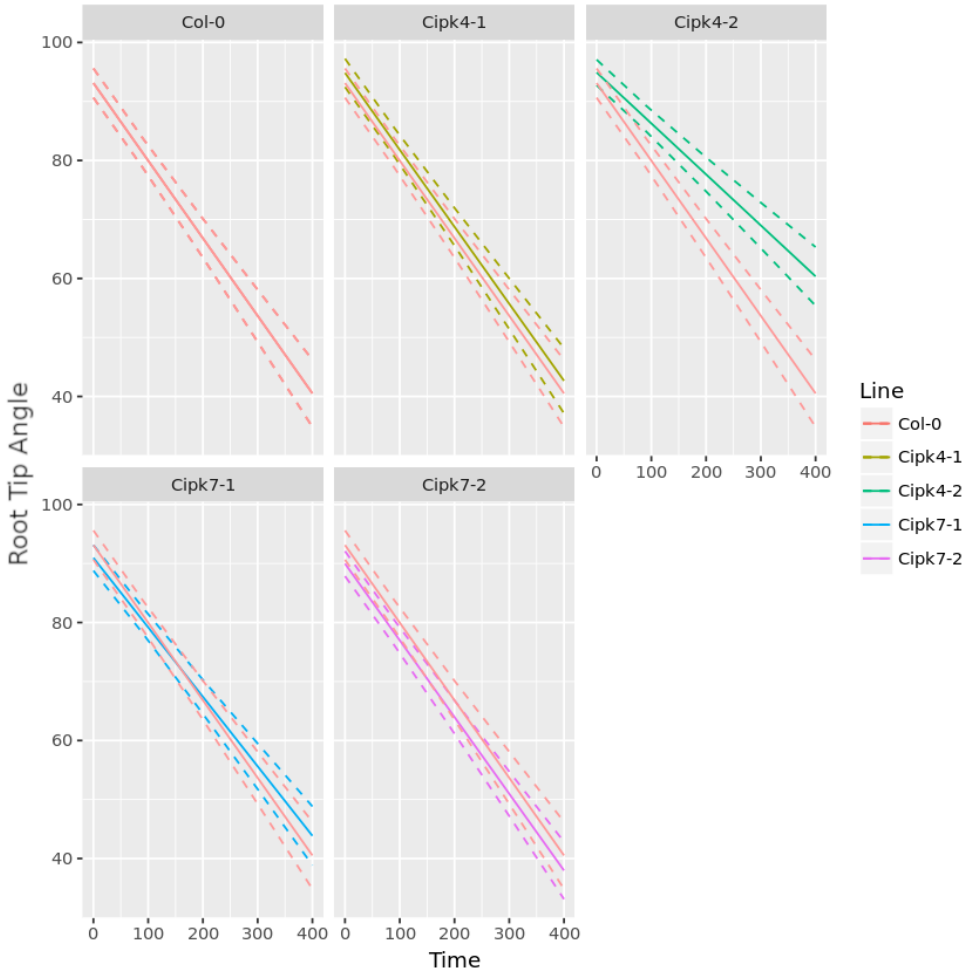
	Estimate	Std. Error	df	t value	Pr(> |t|)
(Intercept)	93.12	1.26	90.44	73.68	0.00
LineCipk4-1	1.73	1.76	90.38	0.98	0.33
LineCipk4-2	1.80	1.68	90.84	1.07	0.29
LineCipk7-1	-2.16	1.68	91.06	-1.29	0.20
LineCipk7-2	-3.15	1.66	90.57	-1.89	0.06
Time	-0.13	0.01	90.88	-17.82	0.00
LineCipk4-1:Time	0.00	0.01	90.90	0.10	0.92
LineCipk4-2:Time	0.05	0.01	91.04	4.60	0.00
LineCipk7-1:Time	0.01	0.01	91.61	1.40	0.17
LineCipk7-2:Time	0.00	0.01	91.03	0.15	0.88



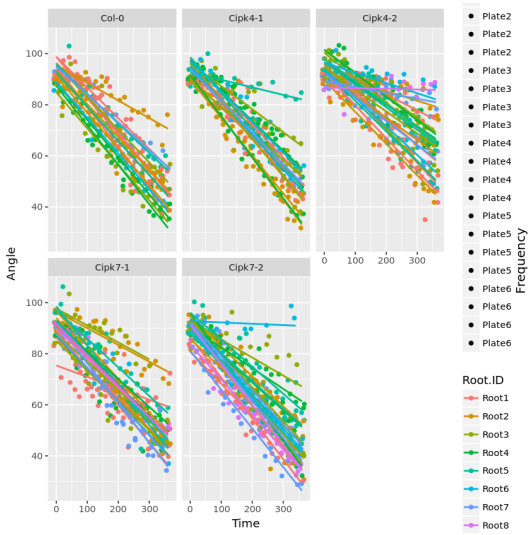
Supplemental Material, Figure S31: Growth rates of Col-0, *cipk4-1*, *cipk4-2*, *cipk7-1*, and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. Time is given in minutes and growth in pixel.



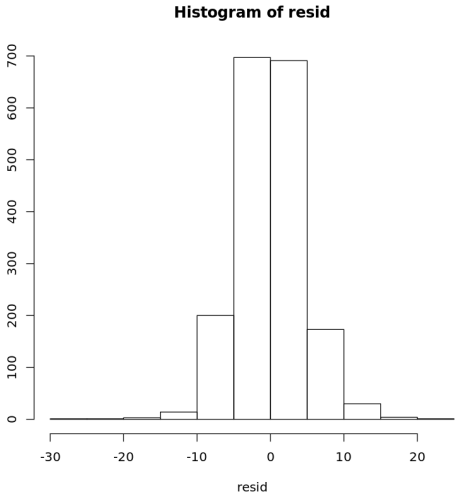
Supplemental Material, Figure S32: Histograms of differences in slope from grand mean as estimated by BLUP (CIPK dataset). Lines indicate 3*standard deviation (red) and 1.5*standard deviation (purple) thresholds, which are given as parameters in qGAP_filt.ipynb



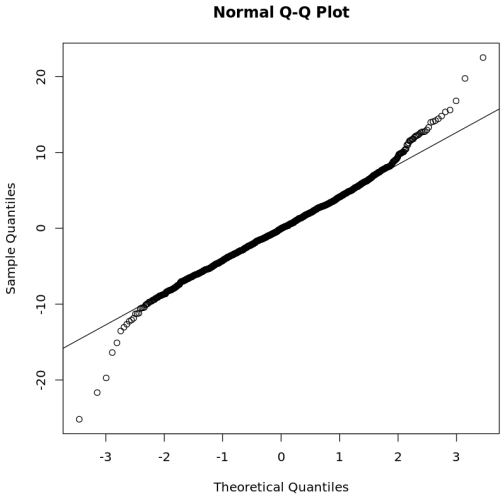
Supplemental Material, Figure S33: Gravitropic kinetics of Col-0, *cipk4-1*, *cipk4-2*, *cipk7-1*, and *cipk7-2* mutants analysed using qGAP. Dashed lines give 95% confidence interval based on linear mixed model. The growth rate outliers were removed from the dataset using 3*standard deviation as parameter in qGAP_filt.ipynb. Time is given in minutes and growth in pixel.



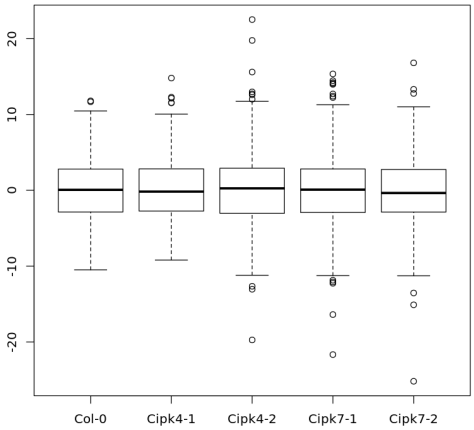
(a) LMM BLUPs superimposed on scatter-plots



(b) Histogram of Residuals

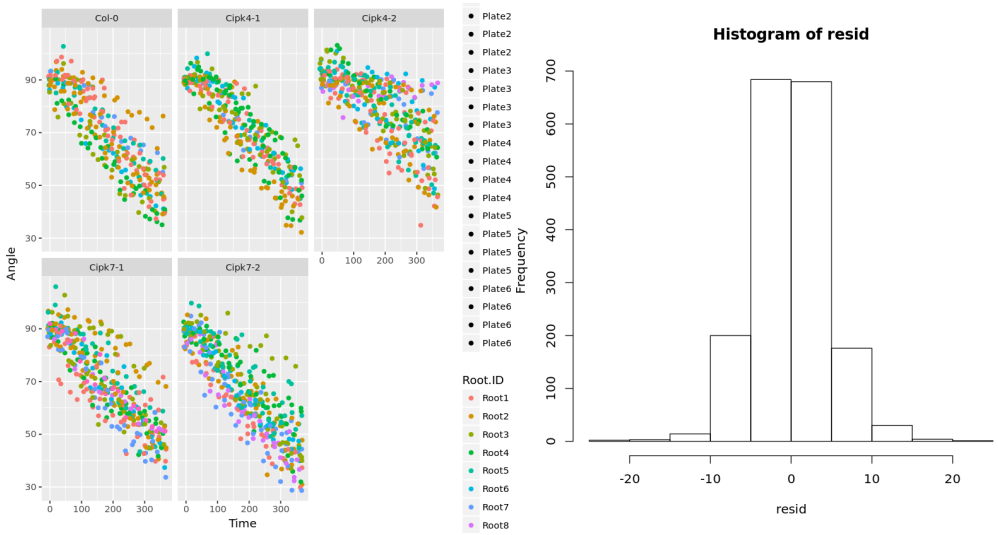


(c) QQ-plot of Residuals



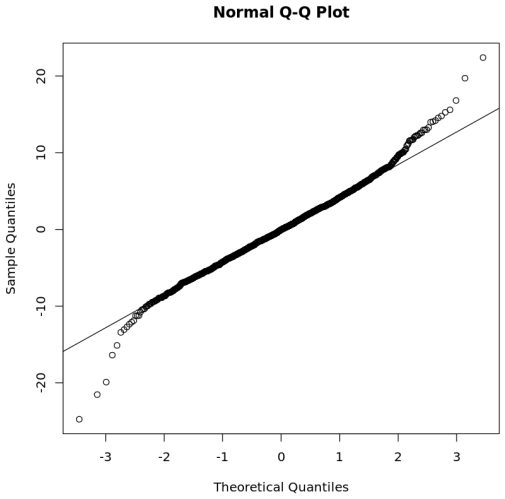
(d) Boxplot of Residuals

Supplemental Material, Figure S34: Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (CIPK full dataset).



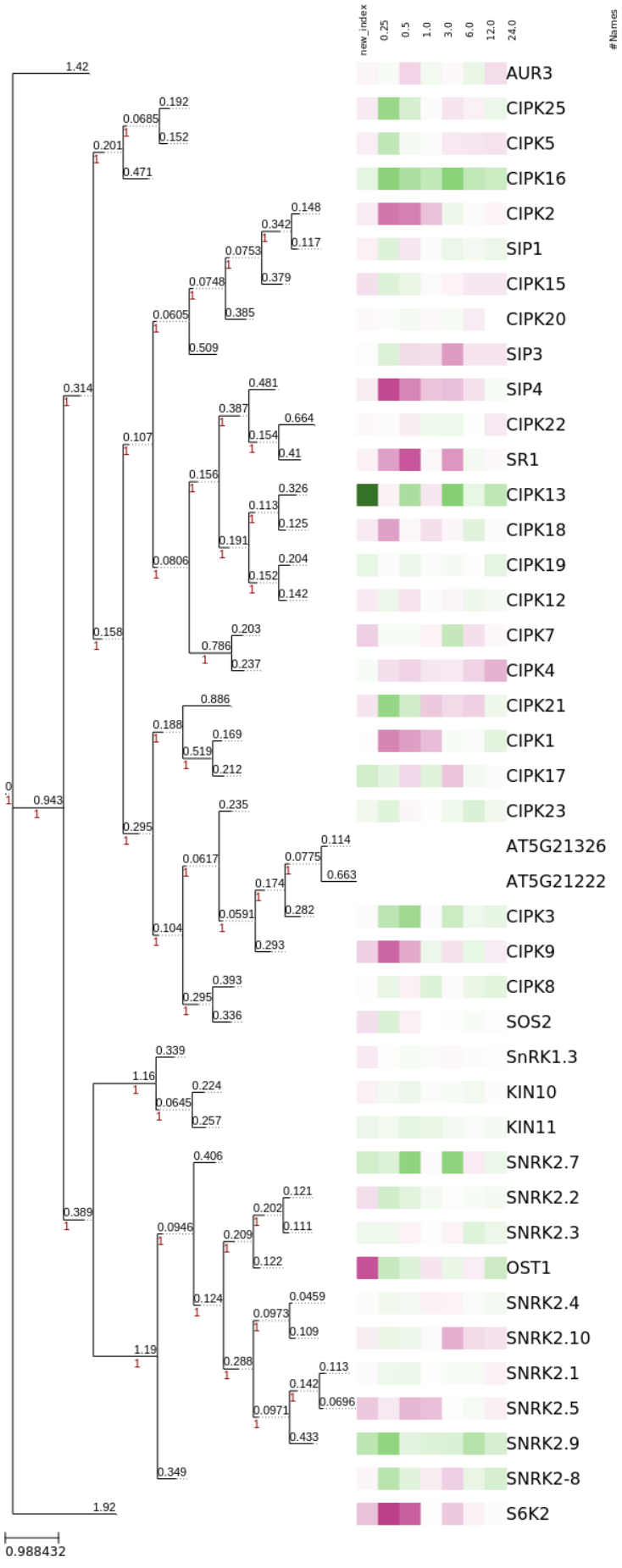
(a) LMM BLUPs superimposed on scatter-plots

(b) Histogram of Residuals



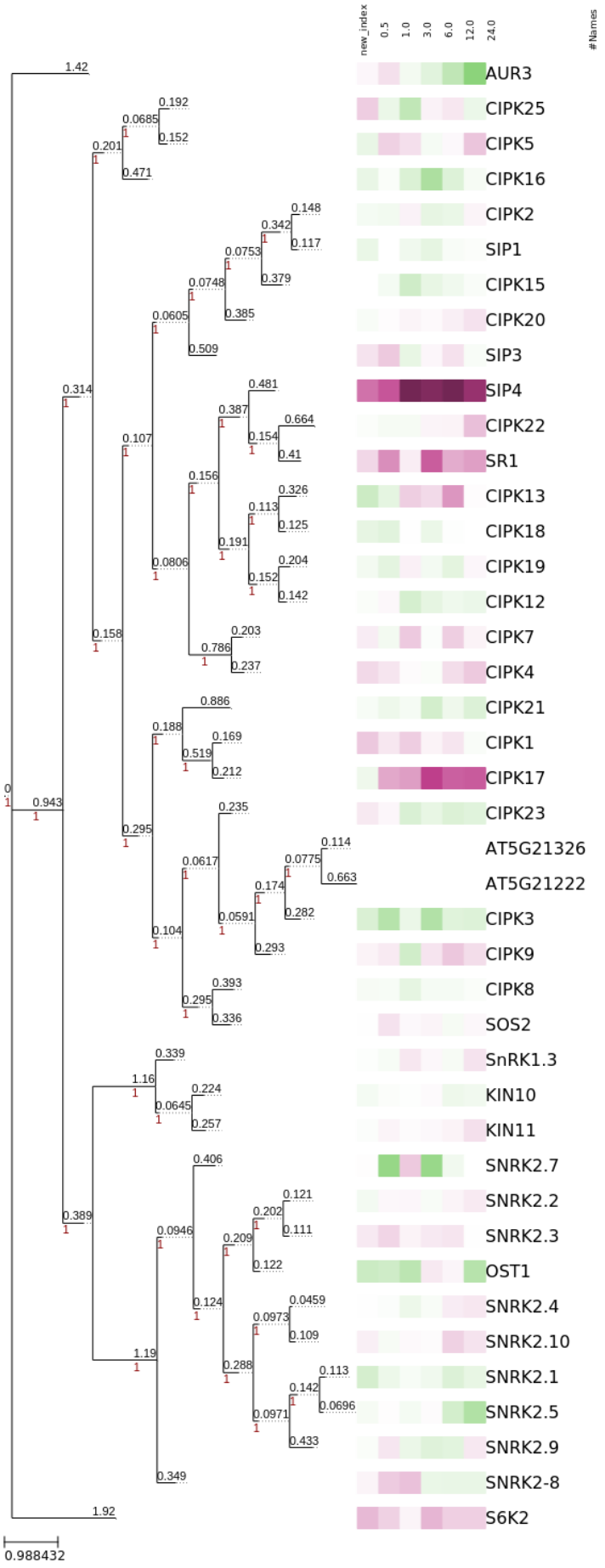
(c) QQ-plot of Residuals

Supplemental Material, Figure S35: Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (CIPK 3*standard deviation filtered dataset) .



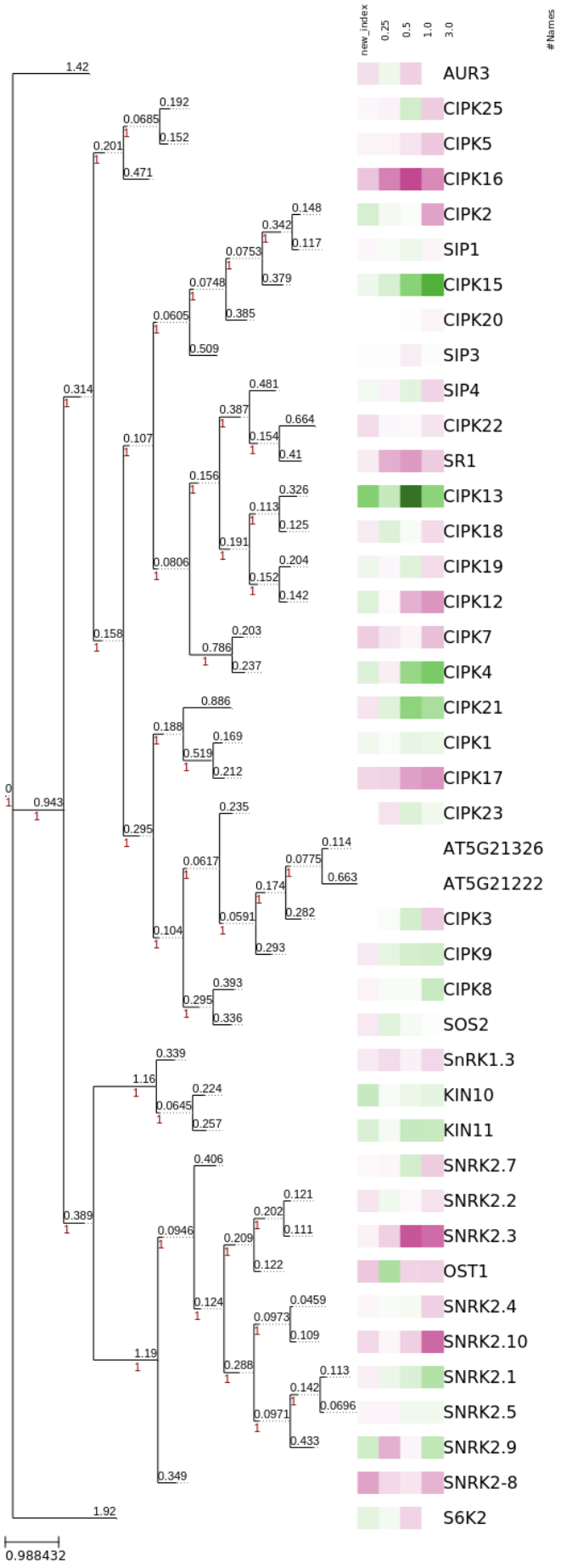
Supplemental Material, Figure S36: *(legend on next page)*

Supplemental Material, Figure S36: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their drought-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



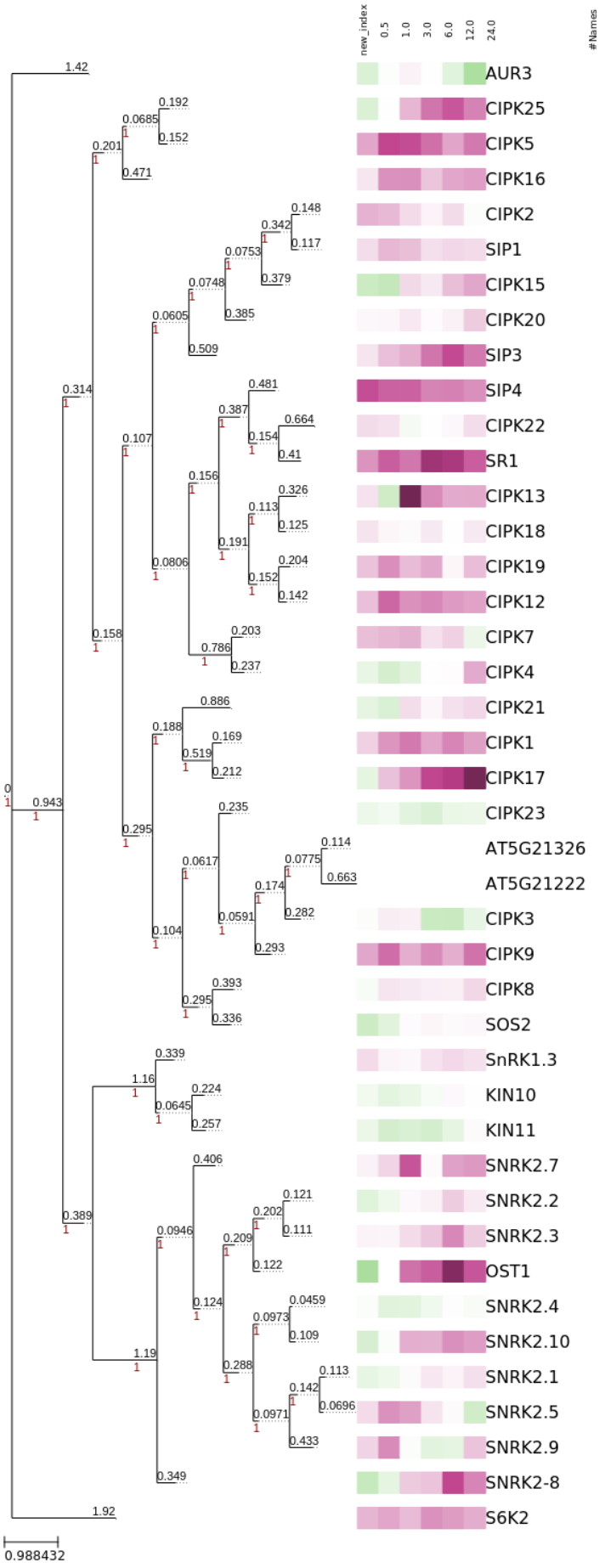
Supplemental Material, Figure S37: *(legend on next page)*

Supplemental Material, Figure S37: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their genotoxicity-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



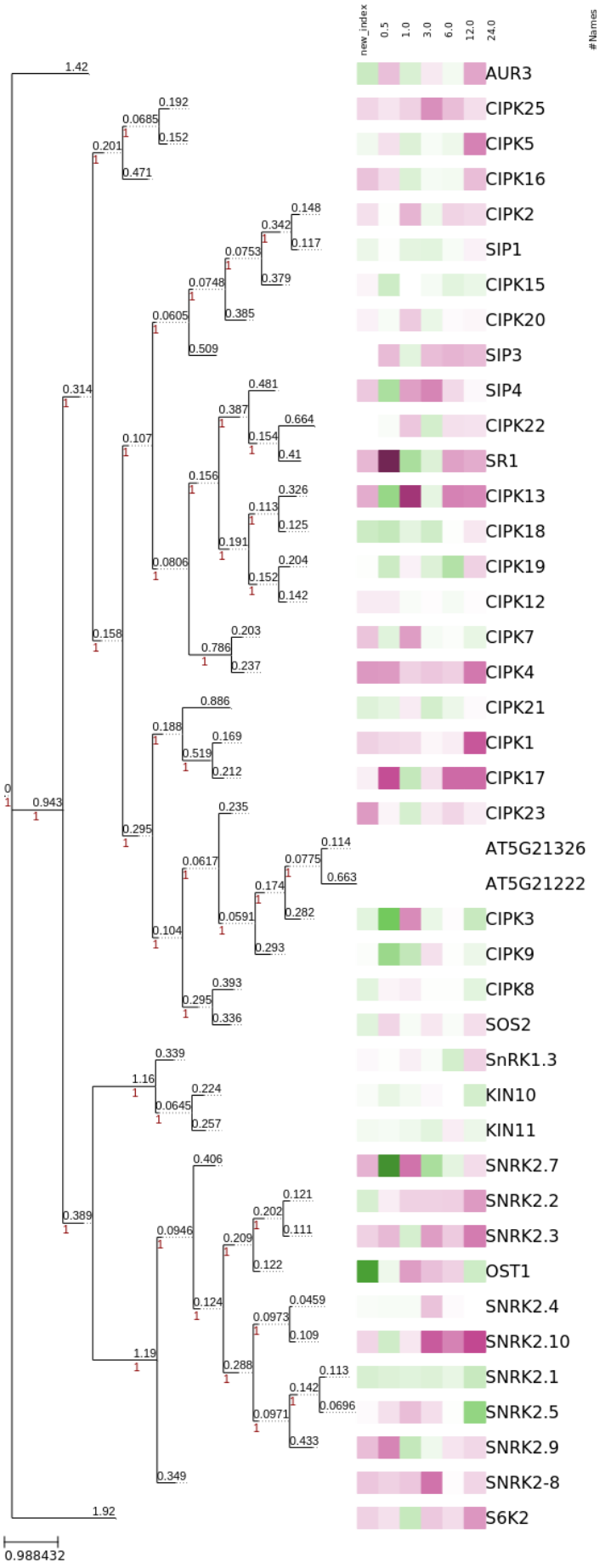
Supplemental Material, Figure S38: *(legend on next page)*

Supplemental Material, Figure S38: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their heat-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



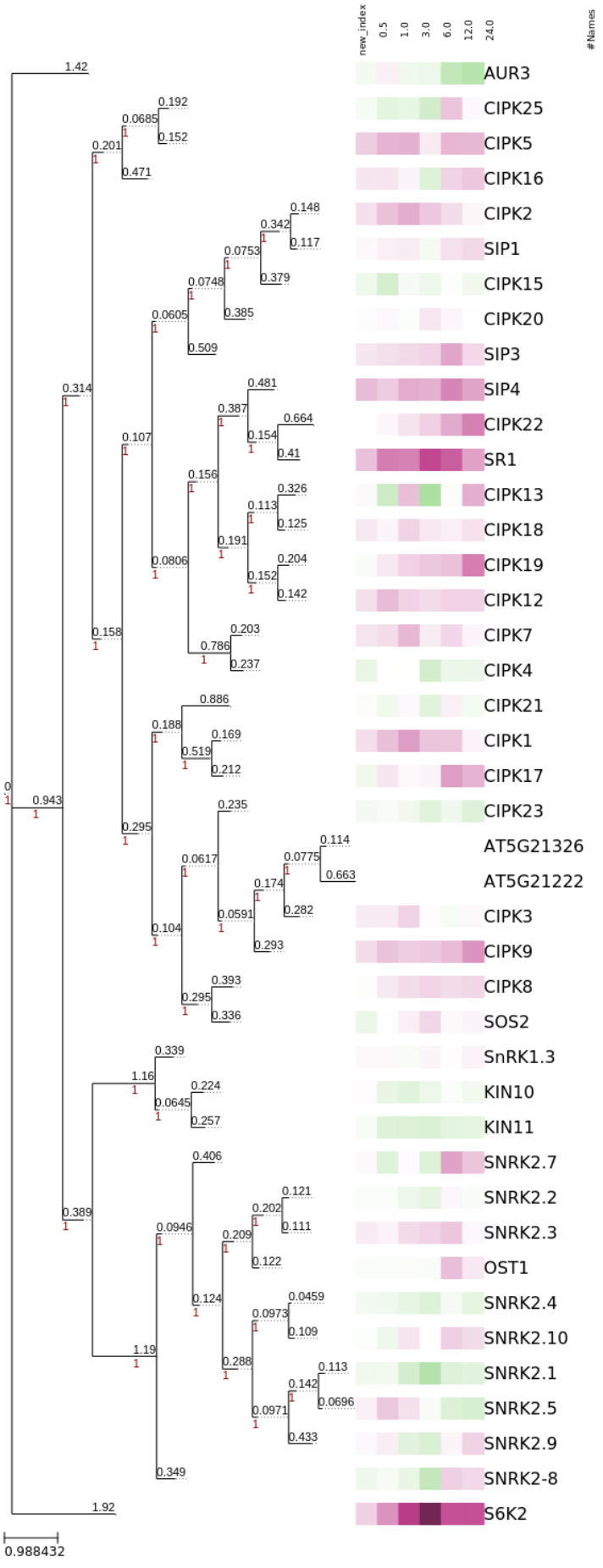
Supplemental Material, Figure S39: *(legend on next page)*

Supplemental Material, Figure S39: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their osmotic-stress-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



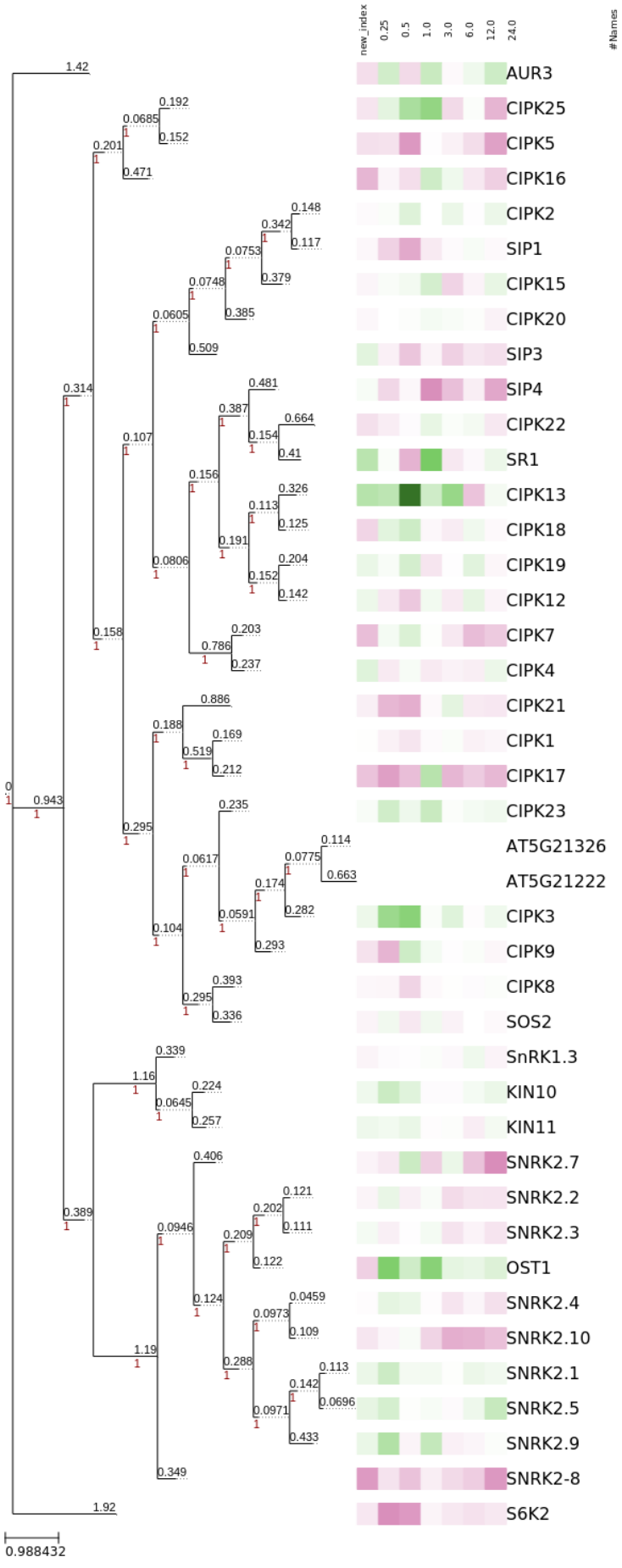
Supplemental Material, Figure S40: *(legend on next page)*

Supplemental Material, Figure S40: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their oxidative-stress-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



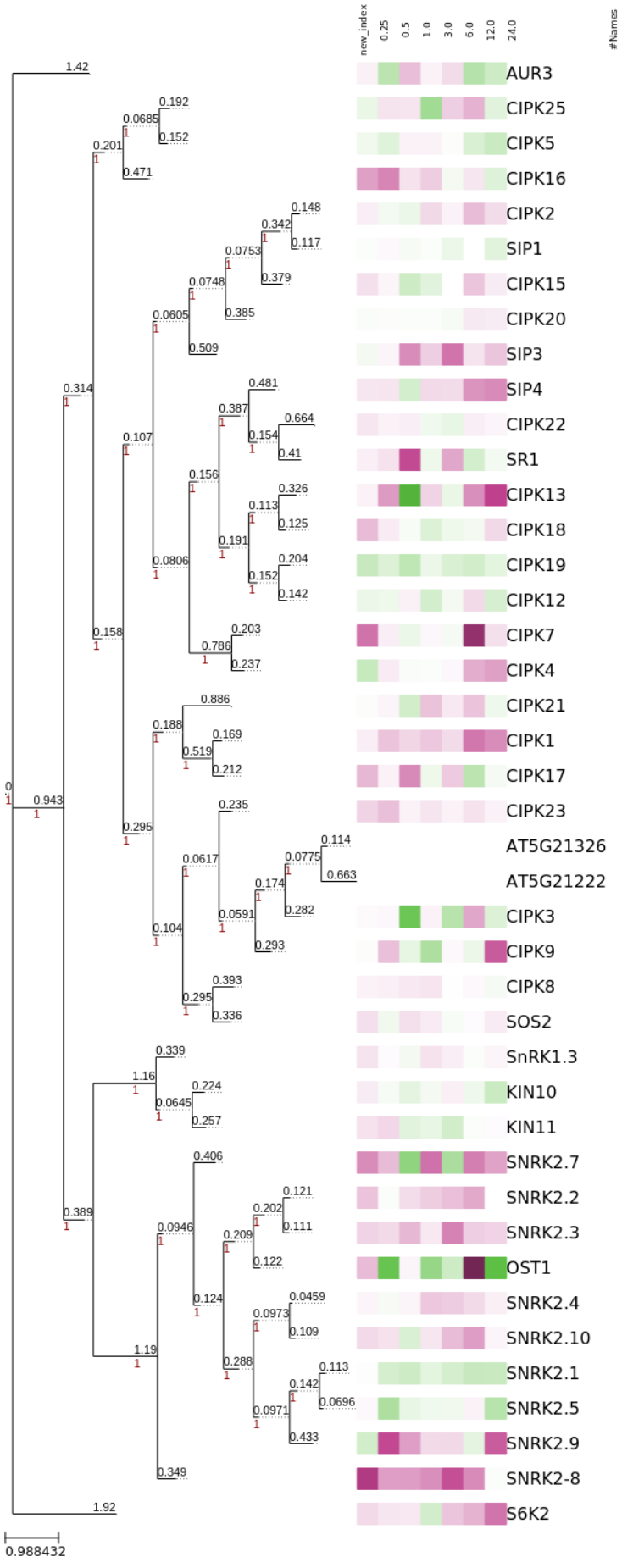
Supplemental Material, Figure S41: *(legend on next page)*

Supplemental Material, Figure S41: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their salt-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



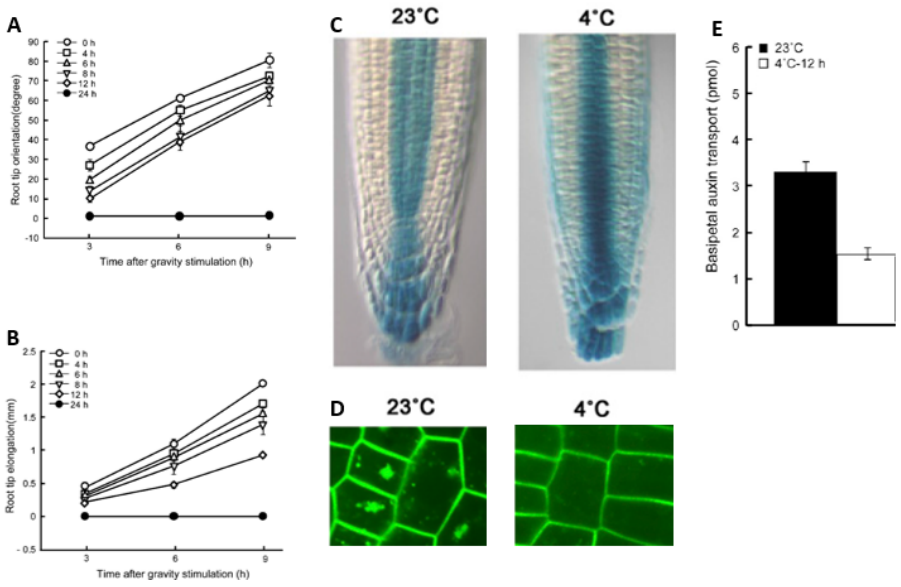
Supplemental Material, Figure S42: *(legend on next page)*

Supplemental Material, Figure S42: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their UVB-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)

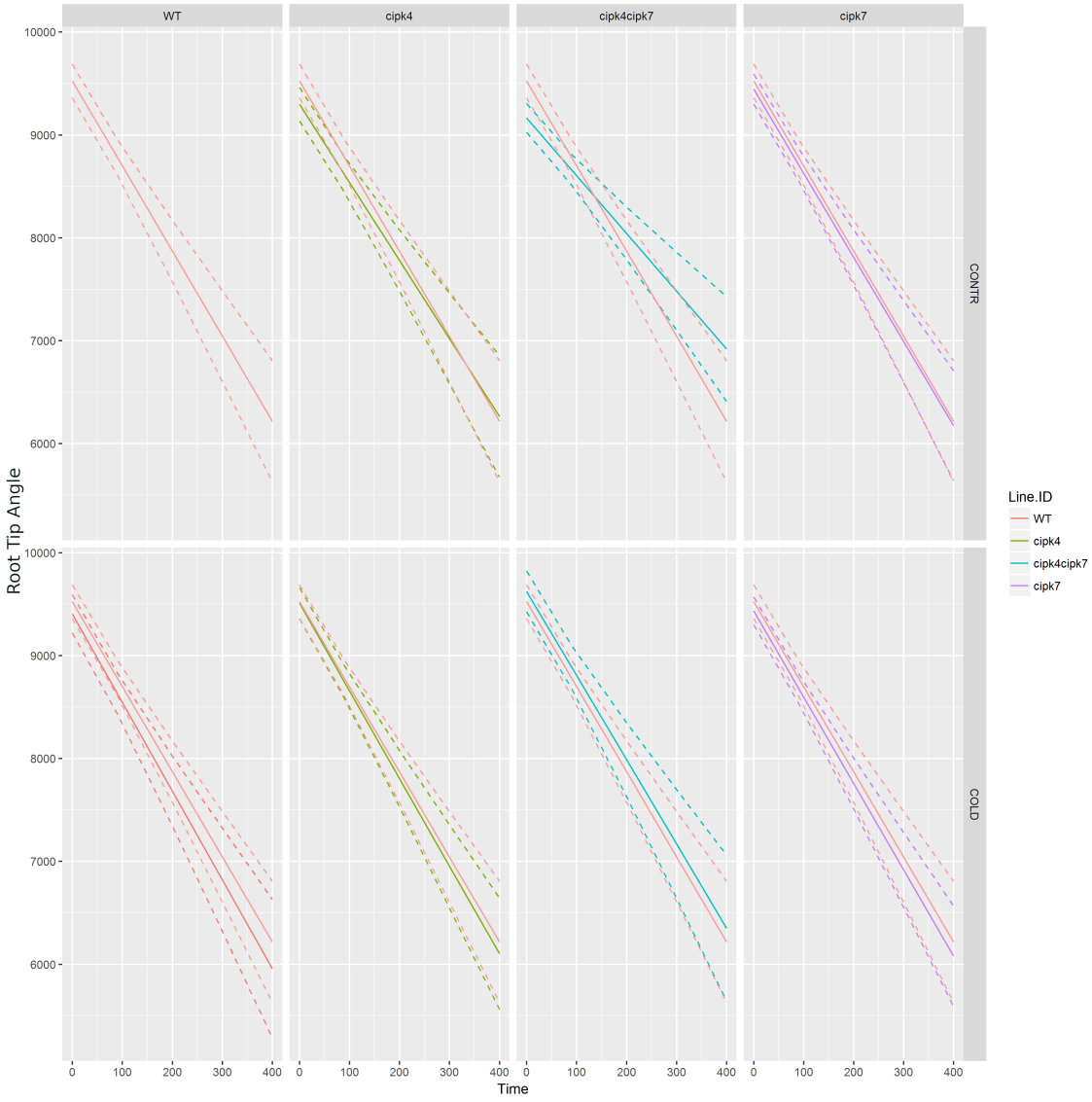


Supplemental Material, Figure S43: *(legend on next page)*

Supplemental Material, Figure S43: Phylogeny (full protein) of selected *Arabidopsis thaliana* CIPK kinases showing their wounding-induced expression in the root. Expression heat-maps are based on micro-array data from Kilian *et al.* (2007). Green indicates up-regulation and purple down-regulation. Gradients for green and purple are based on the maximum and minimum values of the dataset. Black and red numbers indicate branch length and support, respectively. Figure created using ete3 (see Section 2.4.5)



Supplemental Material, Figure S44: (A) Gravitropic kinetics and (B) root tip elongation of Col-0 root after 0, 4, 6, 8, 12, and 24 hours of cold treatment. (C) Auxin response after 12 hours of cold treatment compared to untreated roots as shown by IAA2-GUS auxin marker line. (D) Accumulation of PIN2-GFP in BFA-compartments is absent in roots subjected to cold treatment for 12 hours. (E) Rate of shootward auxin transport is inhibited after 12 hours of cold treatment as measured by transport of radioactively labelled IAA. Adapted from Shibasaki *et al.* (2009)



Supplemental Material, Figure S45: Gravitropic kinetics for *cipk4-1, cipk7-1, cipk4-1 cipk7-1* mutants and isozygous wild-type control after control (top) and cold treatment (bottom) analysed using qGAP. Dashed lines show 95% confidence interval based on linear mixed model. Time is given in minutes and angle in degree times 100.

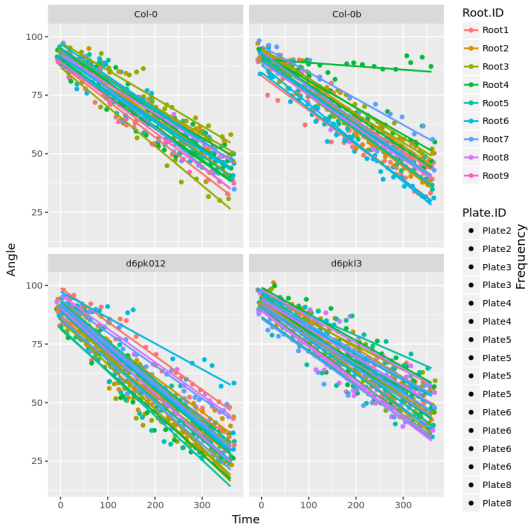
	RootTipAngle		
	B	*CI*	*p*
Fixed Parts			
(Intercept)	9523.98	9361.09 – 9686.86	<.001
Line.IDcipk4	-225.29	-456.03 – 5.45	.057
Line.IDcipk4cipk7	-358.18	-573.23 – -143.13	.001
Line.IDcipk7	-79.54	-299.20 – 140.11	.479
Time	-8.27	-9.84 – -6.70	<.001
TreatmentCOLD	-118.57	-365.71 – 128.58	.348
Line.IDcipk4:Time	0.68	-1.54 – 2.90	.550
Line.IDcipk4cipk7:Time	2.65	0.58 – 4.72	.013
Line.IDcipk7:Time	0.10	-2.02 – 2.21	.930
Line.IDcipk4:TreatmentCOLD	328.59	-3.33 – 660.51	.054
Line.IDcipk4cipk7:TreatmentCOLD	576.15	229.15 – 923.15	.001
Line.IDcipk7:TreatmentCOLD	108.34	-209.40 – 426.08	.505
Time:TreatmentCOLD	-0.35	-2.73 – 2.03	.774
Line.IDcipk4:Time:TreatmentCOLD	-0.58	-3.77 – 2.62	.723
Line.IDcipk4cipk7:Time:TreatmentCOLD	-2.22	-5.54 – 1.11	.193
Line.IDcipk7:Time:TreatmentCOLD	0.14	-2.92 – 3.20	.930
Random Parts			
σ^2		161068.014	
$\tau_{00,\ Plate.ID:Root.ID}$		148114.501	
ρ_{01}		-0.318	
$N_{Plate.ID:Root.ID}$		226	
$ICC_{Plate.ID:Root.ID}$		0.479	
Observations		4230	
R^2 / Ω_0^2		.908 / .908	

Supplemental Material, Figure S46: Summary of gravitropic kinetic linear mixed model of CIPK mutants under cold and control treatment. Time is given in minutes and angle in degree times 100.

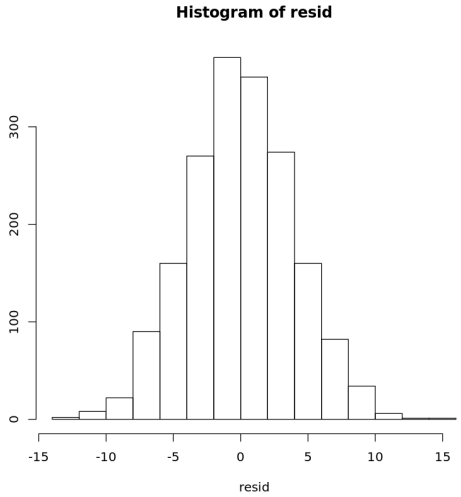
	Growth		
	B	*CI*	*p*
Fixed Parts			
(Intercept)	-131.88	-171.92 – -91.85	<.001
Line.IDcipk4	74.40	17.64 – 131.16	.011
Line.IDcipk4cipk7	99.93	47.07 – 152.79	<.001
Line.IDcipk7	-3.73	-57.76 – 50.31	.893
Time	4.25	3.53 – 4.97	<.001
TreatmentCOLD	75.39	14.62 – 136.15	.016
Line.IDcipk4:Time	-2.72	-3.75 – -1.70	<.001
Line.IDcipk4cipk7:Time	-2.86	-3.81 – -1.90	<.001
Line.IDcipk7:Time	0.29	-0.69 – 1.26	.566
Line.IDcipk4:TreatmentCOLD	-63.49	-145.14 – 18.16	.129
Line.IDcipk4cipk7:TreatmentCOLD	-170.92	-256.43 – -85.42	<.001
Line.IDcipk7:TreatmentCOLD	63.36	-14.79 – 141.50	.113
Time:TreatmentCOLD	-2.46	-3.56 – -1.36	<.001
Line.IDcipk4:Time:TreatmentCOLD	3.11	1.64 – 4.59	<.001
Line.IDcipk4cipk7:Time:TreatmentCOLD	2.43	0.89 – 3.96	.002
Line.IDcipk7:Time:TreatmentCOLD	0.14	-1.27 – 1.55	.850
Random Parts			
σ^2		14557.182	
$\tau_{00,\ Plate.ID:Root.ID}$		8006.027	
ρ_{01}		-0.560	
$N_{Plate.ID:Root.ID}$		226	
$ICC_{Plate.ID:Root.ID}$		0.355	
Observations		4230	
R^2 / Ω_0^2		.949 / .949	

Supplemental Material, Figure S47: Summary of growth rate linear mixed model of CIPK mutants under cold and control treatment. Time is given in minutes and growth in pixel.

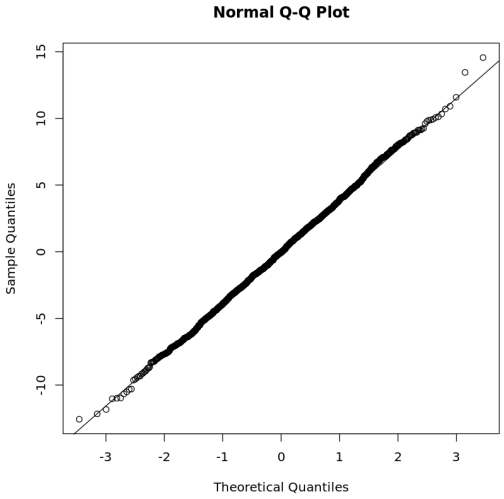
S1.5 Supplemental Figures for Chapter 6: The Quantitative Gravitropic Analysis Pipeline (qGAP) - A Novel and Adaptable Phenotyping Tool



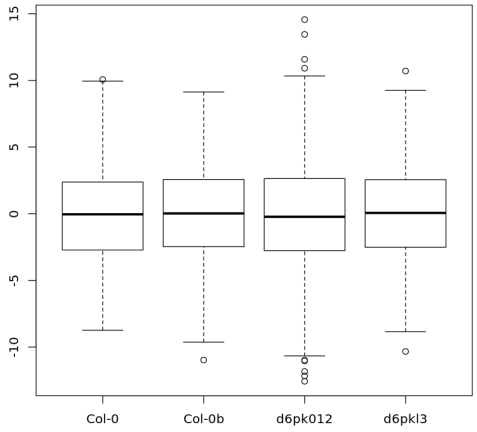
(a) LMM BLUPs superimposed on scatter-plots



(b) Histogram of Residuals

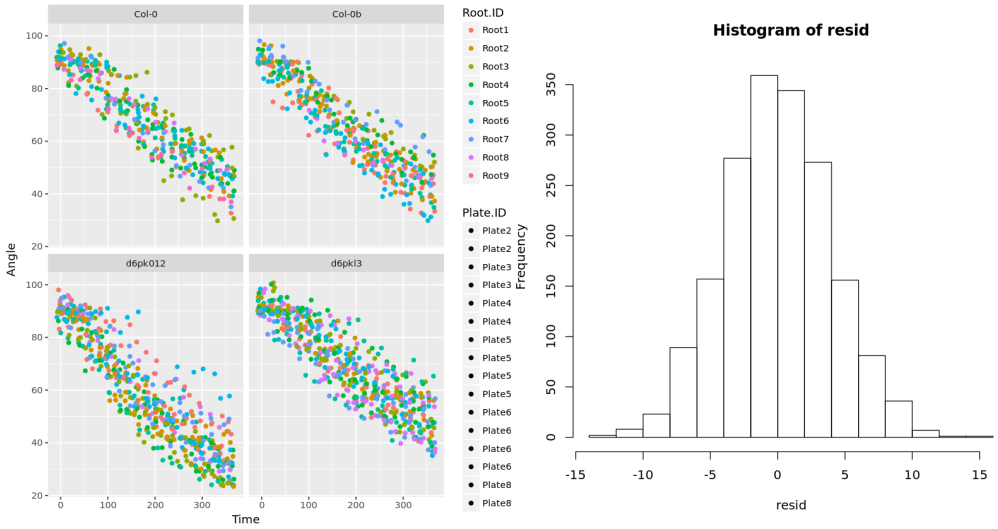


(c) QQ-plot of Residuals



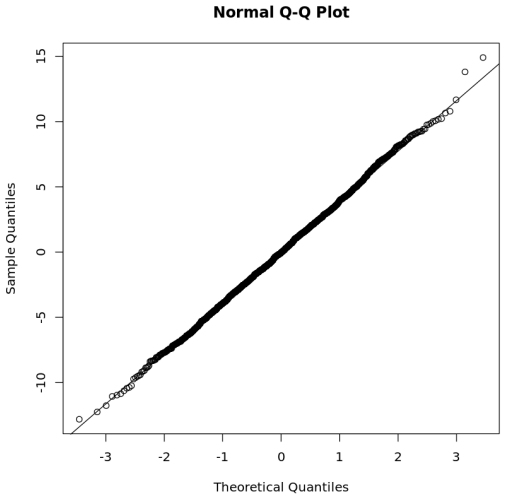
(d) Boxplot of Residuals

Supplemental Material, Figure S48: Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (full "D6PKinit"/qGAP example dataset).



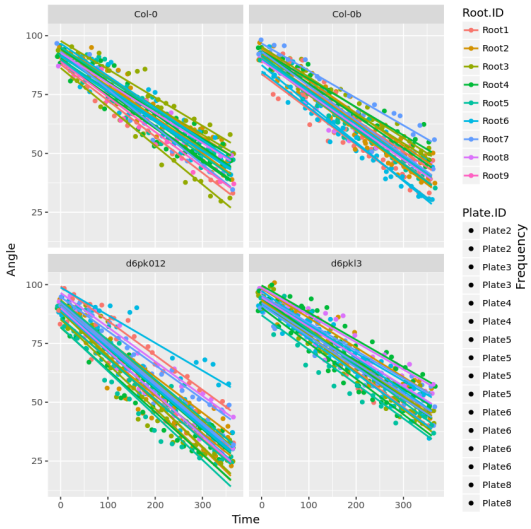
(a) LMM BLUPs superimposed on scatter-plots

(b) Histogram of Residuals

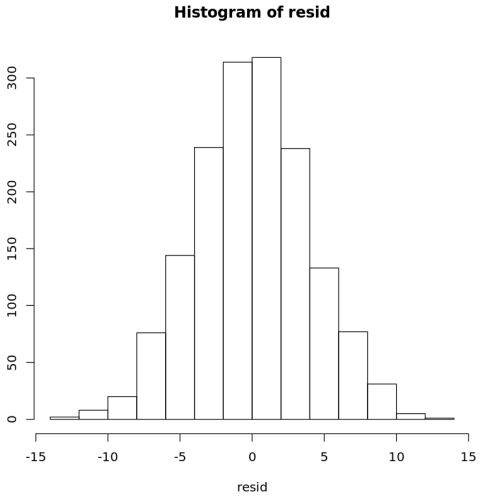


(c) QQ-plot of Residuals

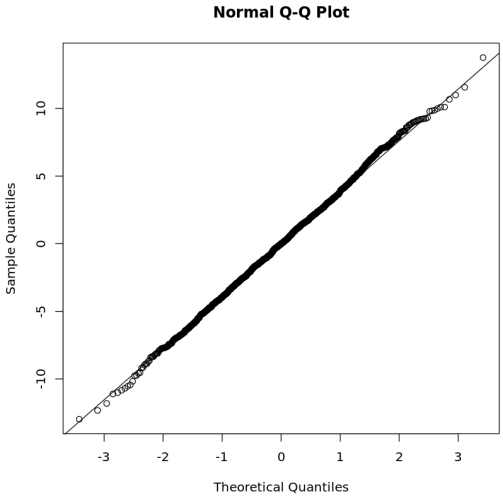
Supplemental Material, Figure S49: Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (3*standard deviation filtered "D6PKinit"/qGAP example dataset).



(a) LMM BLUPs superimposed on scatter-plots



(b) Histogram of Residuals



(c) QQ-plot of Residuals

Supplemental Material, Figure S50: Plots showing that the requirements of linear mixed models are met by the gravitropic kinetic data (1.5*standard deviation filtered "D6PKinit"/qGAP example dataset).

S2 Supplemental Code

S2.1 Jupyter Notebooks

S2.1.1 qGAP-notebooks

Listing A.1: qgap_read.ipynb

```
# In[1]:


ROOTTRACE_OUTPUT_DIR = '/home/lukas/data/LukasDataSelect/
    ↪ WDgravitrop040314'
RUN_ID = 'WDgravitrop040314_01'


# In[2]:


TIME_INTERVAL = 20
TIME_DURATION = 360
MIN_GROWTHRATE_NOT_TRACELOSS = 0.01
MAX_GROWTHRATE_NOT_TRACELOSS = 6 #was 30
MAX_NUM_NAN_PER_ROOT = 10
UPPER_QUARTILE = .95
LOWER_QUARTILE = .05
ADJUST_GROWTHRATE_TO_WT = False
ADJUST_UPPER_QUARTILE = .99
ADJUST_LOWER_QUARTILE = .01
INITIAL_ANGLE_RANGE_TUPLE = (-9, 9)
NAME_WT = 'Col-0'

# Edit GravitropicAnalysisTemplate.csv and save as LAYOUT.csv


# In[3]:


PROJECTS_PATH = '/home/lukas/lkbm/projects'
LAYOUT_CSV = ROOTTRACE_OUTPUT_DIR + '/LAYOUT.csv'


# In[4]:
```

```
import pandas as pd
pd.set_option('display.max_rows', None, 'display.max_columns', None)
import numpy
import sys
import feather
import seaborn as sns
import matplotlib.pylab as plt

sys.path.append(PROJECTS_PATH)

import qGAP.io
import qGAP.filters4
import util.maybe as mb
from util.pandasutil2 import create_pairw_id_df_dict
from qGAP.formulaformat import format_df_for_formula


# Read RootTrace csv

# In[5]:


mb_GR_df, mb_AVV_df= qGAP.io.read_roottrace_output (
    ↪ rootnav_output_directory = ROOTTRACE_OUTPUT_DIR,
gr_identifier_str = '.bmp_data.csv',
avv_identifier_str = 'tip_angles_to_vertical.csv',
layout_csv = LAYOUT_CSV,
time_interval = TIME_INTERVAL)

display(mb_GR_df.value)

display(mb_AVV_df.value)



#Filter growth rate dataframe or both dataframes in parallel

# In[6]:


filtered_df_tuple = qGAP.filters4.filter_dfs(mb_gr_df = mb_GR_df,
mb_avv_df = mb_AVV_df,
time_duration = TIME_DURATION,
gr_min_tracel_thresh = MIN_GROWTHRATE_NOT_TRACELOSS,
gr_max_tracel_thresh = MAX_GROWTHRATE_NOT_TRACELOSS,
remove_nan_thresh = MAX_NUM_NAN_PER_ROOT,
remove_overall_quantile = False,
upper_quantile = UPPER_QUARTILE,
```

```
lower_quantile = LOWER_QUARTILE,
remove_wildtype_quantile = ADJUST_GROWTHRATE_TO_WT,
wt_upper_quantile = ADJUST_UPPER_QUARTILE,
wt_lower_quantile = ADJUST_LOWER_QUARTILE,
wt_id = NAME_WT,
initial_angle_tuple = INITIAL_ANGLE_RANGE_TUPLE)


GR_mb_df = filtered_df_tuple[0]

AVV_mb_df = filtered_df_tuple[1]

display(GR_mb_df.value)

display(AVV_mb_df.value)


# The rest of the notebook is kept as a script to show the statisitcs being
    ↪ applied

# In[23]:



GR_df = GR_mb_df.value

analysis_type = 'Growth'
df_T = GR_df.T
df_T.index.name = 'Time'
df_T = df_T.reset_index()
df_T['Time'] = df_T['Time'].astype(float)
df_T = df_T.set_index('Time')
df_T = df_T.stack([0, 1, 2]).reset_index()
df_T.columns = ['Time', 'Plate.ID', 'Root.ID', 'Line', analysis_type]
df_T["Growth"] = df_T["Growth"].map(lambda x: x*10)
df_T = df_T.loc[df_T['Line'].isin(["Col-0", "Col-0b", "d6pkl3", "d6pk012
    ↪ "])]
print(df_T.shape)
g = sns.FacetGrid(df_T, col="Line", col_wrap=2, hue="Root.ID")
g = (g.map(plt.scatter, "Time", analysis_type, edgecolor="w")
     .add_legend())


feather.write_dataframe(df_T, RUN_ID + '_GR.feather')



# In[25]:
```

```
AVV_df = AVV_mb_df.value
analysis_type = 'Angle'
df_T = AVV_df.T
df_T.index.name = 'Time'
df_T = df_T.reset_index()
df_T['Time'] = df_T['Time'].astype(float)
df_T = df_T.set_index('Time')
df_T = df_T.stack([0, 1, 2]).reset_index()
df_T.columns = ['Time', 'Plate.ID', 'Root.ID', 'Line', analysis_type]
df_T["Angle"] = df_T["Angle"].map(lambda x: x*1)
df_T = df_T.loc[df_T['Line'].isin(["Col-0", "Col-0b", "d6pkl3", "d6pk012
    ↪ "])]
print(df_T.shape)
g = sns.FacetGrid(df_T, col="Line", col_wrap=2, hue="Root.ID")
g = (g.map(plt.scatter, "Time", analysis_type, edgecolor="w")
     .add_legend())


feather.write_dataframe(df_T, RUN_ID + '_AVV.feather')
```

Listing A.2: qgap_lmlinit.ipynb

```
#In []:
ROOTTRACE.OUTPUT.DIR <- '/home/lukas/data/LukasDataSelect/
    ↪ WDgravitrop040314'
RUN.ID <- 'WDgravitrop040314_01'

#In []:
library("ggplot2")
library("lmerTest")
library("effects")
library("sjPlot")
library("simr")
library("HLMdiag")
library("pwr")
library("feather")
library("binom")
library("car")
library("xtable")
library("drc")

#In []:
path <- paste(c(RUN.ID, '_AVV.feather'), collapse="")
```

```
df_T <- read_feather(path)

df <- within(df_T,{Line<-factor(Line);
                   Plate.ID<-factor(Plate.ID);
                  Root.ID<-factor(Root.ID)})
df$Line <- relevel(df$Line, ref='Col-0')

df

plot_1 <- ggplot(df,
                 aes(Time,Angle,colour=Root.ID,fill=Plate.ID))+ #,fill=
                     ↪ Line
                 geom_jitter() +
                 facet_wrap(~Line)
plot_1

#In []:
mlm <- lmer(Angle~Line*Time + (Time|Plate.ID:Root.ID), data=df) #
    ↪ Growth~0+Line
sum <- summary(mlm)
ee <- Effect(c('Line','Time'),mlm)
mlm
sum

#In []:
col_mlm <- mlm
d6l3_mlm <- mlm
d6012_mlm <- mlm

#In []:
df$fit <- predict(mlm)
plot_1 <- ggplot(df,
                 aes(Time,Angle,colour=Root.ID,fill=Plate.ID))+ #,fill=
                     ↪ Line
                 geom_jitter() +
                 facet_wrap(~Line) +
       geom_line(aes(y=fit), size=0.8)

plot_1

#ranef(mlm)

#In []:
resid <- residuals(mlm)
#resid
hist(resid)
```

```
#In []:
qqnorm(resid)
qqline(resid)

#In []:
boxplot(residuals(mlm) ~ df$Line)
leveneTest(residuals(mlm) ~ df$Line)

#In []:
doTest(mlm, fixed("Lined6pkl3:Time", "t"))

doTest(mlm, fixed("Line:Time", "anova"))


#In []:
d <- NULL


for(i in seq(0, -0.05, by=-0.005)){
        fixef(col_mlm)["LineCol-0b:Time"] <- i
        x <- powerSim(col_mlm, fixed("LineCol-0b:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <- sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
             cat("<NA>")
             return()
         }

         interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
             ↪ c("mean", "lower", "upper")]
         power <- interval$mean

    d <- rbind(d, data.frame(i, power))

    }

colnames(d) <- c("Effect.size", "Power")

print(d)

plot_1 <- ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                     method.args = list(family = "binomial"),
                     se = FALSE)
```

```
plot_1



#In []:
d <- NULL


for(i in seq(0, -0.05, by=-0.005)){
        fixef(d6012_mlm)["Lined6pk012:Time"] <- i
        x <- powerSim(d6012_mlm, fixed("Lined6pk012:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <- sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
              cat("<NA>")
              return()
         }

         interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
             ↪ c("mean", "lower", "upper")]
         power <- interval$mean

    d <- rbind(d, data.frame(i, power))

    }

colnames(d) <- c("Effect.size", "Power")

print(d)

plot_1 <- ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                     method.args = list(family = "binomial"),
                     se = FALSE)

plot_1



#In []:
d <- NULL


for(i in seq(0, -0.05, by=-0.005)){
        fixef(d6l3_mlm)["Lined6pkl3:Time"] <- i
```

```
        x <- powerSim(d6l3_mlm, fixed("Lined6pkl3:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <- sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
             cat("<NA>")
             return()
         }

         interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
             ↪ c("mean", "lower", "upper")]
         power <- interval$mean

    d <- rbind(d, data.frame(i, power))

    }

colnames(d) <- c("Effect.size", "Power")

print(d)

plot_1 <- ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                     method.args = list(family = "binomial"),
                     se = FALSE)

plot_1

#In []:
x <- powerSim(mlm, fixed("LineCol-0b:Time", "t"), nsim=100)
print(x)
z <- sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <- interval$mean
print(power)

x <- powerSim(mlm, fixed("Lined6pk012:Time", "t"), nsim=100)
print(x)
```

```
z <− sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <− binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <− interval$mean
print(power)

#In []:
x <− powerSim(mlm, fixed("Lined6pkl3:Time", "t"), nsim=100)
print(x)
z <− sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <− binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <− interval$mean
print(power)

#In []:
GR.ranef <− as.data.frame(ranef(mlm))
GR.ranef
path.ranef <− paste(c(RUN.ID, '_AVV_ranef.feather'), collapse="")
write_feather(GR.ranef, path.ranef)

#In []:
df <− as.data.frame(ee)
wt_df <− df[df$Line == 'Col−0',]
df$Line <− relevel(df$Line, ref='Col−0')

#In []:
plot_1 <− ggplot(df,
                 aes(Time,fit,colour=Line,fill=Line))+ #,fill=Line
                 geom_line()+
                 geom_line(aes(y=lower), linetype = 'dashed')+
                 geom_line(aes(y=upper), linetype = 'dashed')+
                 facet_wrap(~Line)+
                 annotate(geom='line', x = wt_df$Time , y = wt_df$fit, colour
```

```
                ↪  = '#FF9999')+
            annotate(geom='line', x = wt_df$Time , y = wt_df$upper,
                ↪ linetype = 'dashed', colour = '#FF9999')+
            annotate(geom='line', x = wt_df$Time , y = wt_df$lower,
                ↪ linetype = 'dashed', colour = '#FF9999')

plot_1

#In []:
plot_2 <- sjp.lmer(mlm, y.offset = 0.4, facet.grid = TRUE, geom.size =
    ↪ .001) #, sort.est = 'Time'
plot_2

#In []:
sum

#In []:
print(xtable(coef(sum)))
```

Listing A.3: qgap_lmlfilt.ipynb

```
# In[1]:

ROOTTRACE_OUTPUT_DIR = 'C://LukasData/WDgravitrop040314'
RUN_ID = 'WDgravitrop040314_01'


# In[2]:

TIME_INTERVAL = 20
TIME_DURATION = 360
MIN_GROWTHRATE_NOT_TRACELOSS = 0.01
MAX_GROWTHRATE_NOT_TRACELOSS = 30
MAX_NUM_NAN_PER_ROOT = 3
UPPER_QUARTILE = .95
LOWER_QUARTILE = .05
ADJUST_GROWTHRATE_TO_WT = False
ADJUST_UPPER_QUARTILE = .99
ADJUST_LOWER_QUARTILE = .01
INITIAL_ANGLE_RANGE_TUPLE = (-9, 9)
NAME_WT = 'Col-0'

# Edit GravitropicAnalysisTemplate.csv and save as LAYOUT.csv


# In[3]:
```

```
PROJECTS_PATH = '/home/lukas/lkbm/projects'


# In[4]:


import pandas
import pandas as pd
pd.set_option('display.max_rows', None, 'display.max_columns', None)
import numpy as np
import sys
import feather
from IPython.display import display, HTML, Image
import matplotlib.pylab as plt
import seaborn as sns
from scipy.stats import norm

sys.path.append(PROJECTS_PATH)

import qGAP.io
import qGAP.filters4
import util.maybe as mb
from util.pandasutil2 import create_pairw_id_df_dict
from qGAP.formulaformat import format_df_for_formula
from qGAP.plot4 import plot_linear_model_and_ancova
import pandas as pd



# In[5]:



GR_ranef_df = feather.read_dataframe(RUN_ID + "_GR_ranef.feather")

print(GR_ranef_df)

#select slope values

GR_slope_ranef_df = GR_ranef_df.loc[GR_ranef_df['term'].isin(["Time"])]

GR_slope_ranef_df["slope_diff_to_GM"] = GR_slope_ranef_df["condval"]

#plot histogram of slope GR ranef

n, bins, patches = plt.hist(GR_slope_ranef_df.loc[:,"slope_diff_to_GM"], 40,
```

```
    ↪  density=True, facecolor='g', alpha=0.75, histtype='stepfilled',
    ↪ range=(−0.20, 0.20))

mu, std = norm.fit(GR_slope_ranef_df.loc[:,"slope_diff_to_GM"])

plt.xlabel('Slope')
plt.ylabel('Frequence')
plt.axis([−0.40, 0.40, 0, 40])
plt.grid(True)
#line
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Fit␣results:␣mu␣=␣%.2f,␣␣std␣=␣%.2f" % (mu, std)
plt.title(title)

plt.show()



# In[6]:


#add line id to GR ranef dataframe

GR_df = feather.read_dataframe(RUN_ID + "_GR.feather")

GR_df["grp"] = GR_df.apply(lambda x: x["Plate.ID"] + ":" + x["Root.ID
    ↪ "], axis = 1)

merged = pd.merge(GR_slope_ranef_df, GR_df, on='grp')

merged_o = merged.loc[merged['Time'].isin([0])]

GR_slope_ranef_lineid_df = merged_o.loc[:,["grp", "Line", "
    ↪ slope_diff_to_GM"]]


three_std = 3∗GR_slope_ranef_lineid_df.slope_diff_to_GM.std()

onefive_std = 1.5∗GR_slope_ranef_lineid_df.slope_diff_to_GM.std()




GR_slope_ranef_lineid_df.loc[:,"3std:mask"] = np.abs(
    ↪ GR_slope_ranef_lineid_df.slope_diff_to_GM−
```

```
    ↪ GR_slope_ranef_lineid_df.slope_diff_to_GM.mean())<=three_std

GR_slope_ranef_lineid_df.loc[:,"1x5std:mask"] = np.abs(
    ↪ GR_slope_ranef_lineid_df.slope_diff_to_GM-
    ↪ GR_slope_ranef_lineid_df.slope_diff_to_GM.mean())<=onefive_std



print(GR_slope_ranef_lineid_df)

g = sns.FacetGrid(GR_slope_ranef_lineid_df, col="Line")

bins = np.linspace(-0.40, 0.40, 20)

g.map(plt.hist, "slope_diff_to_GM", color = "steelblue", bins = bins, lw =
    ↪ 0)
g.map(plt.vlines, x = three_std, ymin = 0, ymax = 10, color = "red")
g.map(plt.vlines, x = -three_std, ymin = 0, ymax = 10, color = "red")
g.map(plt.vlines, x = onefive_std, ymin = 0, ymax = 10, color = "purple"
    ↪ )
g.map(plt.vlines, x = -onefive_std, ymin = 0, ymax = 10, color = "
    ↪ purple")

GR_slope_ranef_std_df = GR_slope_ranef_lineid_df.loc[:,["grp", "3std:mask"
    ↪ , "1x5std:mask"]]


# In[7]:


GR_df = feather.read_dataframe(RUN_ID + "_GR.feather")

GR_df["grp"] = GR_df.apply(lambda x: x["Plate.ID"] + ":" + x["Root.ID
    ↪ "], axis = 1)

merged_std = pd.merge(GR_df, GR_slope_ranef_std_df, on='grp')

merged_three_std = merged_std[merged_std["3std:mask"]]

merged_onefive_std = merged_std[merged_std["1x5std:mask"]]

print(merged_three_std)

print(merged_onefive_std)

feather.write_dataframe(merged_three_std, RUN_ID + "_GR_3std.feather")

feather.write_dataframe(merged_onefive_std, RUN_ID + "_GR_1x5std.
```

```
    ↪ feather")


# In[8]:

AVV_df = feather.read_dataframe(RUN_ID + "_AVV.feather")

print(AVV_df.shape)

AVV_df["grp"] = AVV_df.apply(lambda x: x["Plate.ID"] + ":" + x["Root.
    ↪ ID"], axis = 1)

merged_std = pd.merge(AVV_df, GR_slope_ranef_std_df, on='grp')

merged_three_std = merged_std[merged_std["3std:mask"]]

merged_onefive_std = merged_std[merged_std["1x5std:mask"]]

#print(merged_three_std)

print(merged_three_std.shape)

#print(merged_onefive_std)

print(merged_onefive_std.shape)

feather.write_dataframe(merged_three_std, RUN_ID + "_AVV_3std.feather
    ↪ ")

feather.write_dataframe(merged_onefive_std, RUN_ID + "_AVV_1x5std.
    ↪ feather")
```

Listing A.4: qgap_lmlfin.ipynb

```
#In []:
ROOTTRACE.OUTPUT.DIR <- '/home/lukas/data/LukasDataSelect/
    ↪ WDgravitrop040314'
RUN.ID <- 'WDgravitrop040314_01'

#In []:
library("ggplot2")
library("lmerTest")
library("effects")
library("sjPlot")
library("simr")
library("HLMdiag")
library("pwr")
```

```
library("feather")
library("binom")
library("xtable")

#In []:
path <- paste(c(RUN.ID, '_AVV_3std.feather'), collapse="")

df_T <- read_feather(path)

df <- within(df_T,{Line<-factor(Line);
                   Plate.ID<-factor(Plate.ID);
                  Root.ID<-factor(Root.ID)})
df$Line <- relevel(df$Line, ref='Col-0')

df

plot_1 <- ggplot(df, aes(Time,Angle,colour=Root.ID,fill=Plate.ID))+
    ↪ geom_jitter() + facet_wrap(~Line)
plot_1

#In []:
mlm <- lmer(Angle~Line*Time + (Time|Plate.ID:Root.ID), data=df) #
    ↪ Growth~0+Line
sum <- summary(mlm)
ee <- Effect(c('Line','Time'),mlm)
mlm
sum

#In []:
col_mlm <- mlm
d6l3_mlm <- mlm
d6012_mlm <- mlm

#In []:
df$fit <- predict(mlm)
plot_1 <- ggplot(df,
                 aes(Time,Angle,colour=Root.ID,fill=Plate.ID))+ #,fill=
                     ↪ Line
                 geom_jitter() +
                 facet_wrap(~Line) +
       geom_line(aes(y=fit), size=0.8)

plot_1

ranef(mlm)

#In []:
resid <- residuals(mlm)
```

```
#resid
hist(resid)

#In []:
qqnorm(resid)
qqline(resid)

#In []:
d <- NULL


for(i in seq(0, -0.05, by=-0.005)){
        fixef(col_mlm)["LineCol-0b:Time"] <- i
        x <- powerSim(col_mlm, fixed("LineCol-0b:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <- sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
             cat("<NA>")
             return()
         }

         interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
             ↪ c("mean", "lower", "upper")]
         power <- interval$mean

    d <- rbind(d, data.frame(i, power))

    }

colnames(d) <- c("Effect.size", "Power")

print(d)

plot_1 <- ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                 method.args = list(family = "binomial"),
                 se = FALSE)

plot_1


#In []:
d <- NULL
```

```
for(i in seq(0, −0.05, by=−0.005)){
        fixef(d6012_mlm)["Lined6pk012:Time"] <− i
        x <− powerSim(d6012_mlm, fixed("Lined6pk012:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <− sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
             cat("<NA>")
             return()
         }

         interval <− binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
             ↪ c("mean", "lower", "upper")]
         power <− interval$mean

    d <− rbind(d, data.frame(i, power))

    }

colnames(d) <− c("Effect.size", "Power")

print(d)

plot_1 <− ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                     method.args = list(family = "binomial"),
                     se = FALSE)

plot_1


#In []:
d <− NULL


for(i in seq(0, −0.05, by=−0.005)){
        fixef(d6l3_mlm)["Lined6pkl3:Time"] <− i
        x <− powerSim(d6l3_mlm, fixed("Lined6pkl3:Time", "t"), nsim
            ↪ =100)
        print(x)
        z <− sum(x$pval < x$alpha, na.rm=TRUE)
         if(is.na(z) || is.na(x$n) || (x$n==0)) {
             cat("<NA>")
             return()
         }
```

```
        interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[
            ↪ c("mean", "lower", "upper")]
        power <- interval$mean

    d <- rbind(d, data.frame(i, power))

    }

colnames(d) <- c("Effect.size", "Power")

print(d)

plot_1 <- ggplot(d,
                 aes(Effect.size,Power))+ #,fill=Line
                 geom_point()+
                 geom_smooth(method = "glm",
                     method.args = list(family = "binomial"),
                     se = FALSE)

plot_1

#In []:
x <- powerSim(mlm, fixed("Line:Time", "t"), nsim=100)
print(x)
z <- sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <- interval$mean
print(power)

x <- powerSim(mlm, fixed("LineCol-0b:Time", "t"), nsim=100)
print(x)
z <- sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
```

```
    ↪ mean", "lower", "upper")]
power <- interval$mean
print(power)

#In []:
x <- powerSim(mlm, fixed("Lined6pk012:Time", "t"), nsim=100)
print(x)
z <- sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <- interval$mean
print(power)

#In []:
x <- powerSim(mlm, fixed("Lined6pkl3:Time", "t"), nsim=100)
print(x)
z <- sum(x$pval < x$alpha, na.rm=TRUE)

if(is.na(z) || is.na(x$n) || (x$n==0)) {

    cat("<NA>")
    return()
}

interval <- binom.confint(z, x$n, 0.95, getSimrOption("binom"))[c("
    ↪ mean", "lower", "upper")]
power <- interval$mean
print(power)

#In []:
df <- as.data.frame(ee)
wt_df <- df[df$Line == 'Col-0',]
df$Line <- relevel(df$Line, ref='Col-0')

plot_1 <- ggplot(df,
                 aes(Time,fit,colour=Line,fill=Line))+ #,fill=Line
                 geom_line()+
                 geom_line(aes(y=lower), linetype = 'dashed')+
                 geom_line(aes(y=upper), linetype = 'dashed')+
                 facet_wrap(~Line)+
                 annotate(geom='line', x = wt_df$Time , y = wt_df$fit, colour
```

```
                ↪  = '#FF9999')+
            annotate(geom='line', x = wt_df$Time , y = wt_df$upper,
                ↪ linetype = 'dashed', colour = '#FF9999')+
            annotate(geom='line', x = wt_df$Time , y = wt_df$lower,
                ↪ linetype = 'dashed', colour = '#FF9999')

plot_1

#In []:
plot_2 <- sjp.lmer(mlm, y.offset = 0.4, facet.grid = TRUE, geom.size =
    ↪ .001) #, sort.est = 'Time'
plot_2

#In []:
sum

#In []:
print(xtable(coef(sum)))
```

S2.1.2 KSselect-notebook

Listing A.5: ksselect.ipynb

```
#KSselect a.k.a. KPR

# In[1]:


INFILE = '/media/usb0/infiles/csv/D6PKNDRatPROT111117.csv'


# Notes for Ipython:
# interrupt kernel does not work if modules like numpy were called
#

# In[2]:


INFILE_NO_OF_SUBDIR = 4
PROJECTS_PATH = '/lkbm/projects/kpr/src'
EXE_PATH = '/lkbm/projects/kpr/exe'
RES_PATH = '/lkbm/projects/kpr/res'
DATA_PATH = '/lkbm/data/kpr'
ID_STRUCT_PROT = 'CIPK7'
```

```
# In[3]:


import re
import numpy
#from subprocess import Popen, PIPE
import subprocess as sp
import Bio as bio

import os
import sys
import pickle

sys.path.append(PROJECTS_PATH)

"""
module_path = os.path.abspath(os.path.join('...'))
if module_path not in sys.path:
    sys.path.append(module_path)
"""

import KinasePhyloRemote4 as kpr



# In[4]:


additional_annotation_dict = {
    'NDR1': 'AT4G14350',
    'NDR2': 'AT1G03920',
    'NDR3': 'AT3G23310',
    'NDR4': 'AT2G19400',
    'NDR5': 'AT2G20470',
    'NDR6': 'AT4G33080',
    'NDR7': 'AT1G30640',
    'NDR8': 'AT5G09890',
    'D6PK': 'AT5G55910',
    'D6PKL1': 'AT4G26610',
    'D6PKL2': 'AT5G47750',
    'D6PKL3': 'AT3G27580'
     }


# def cut_out_aligned_to_domain (biopy_alignment, domain_id):
#
# return genes_domain_id_cut_sequences_MSA
```

```
#
# def get_closest_relative_to_sequence_from_alignment (biopy_seqrec,
    ↪ biopy_MSA):
#
# return seqrec
#
#
# def convert_indices_alignment_to_alignment(list_of_indices,
    ↪ biopy_MSA_from, biopy_MSA_to, id_in_common):
#
# return converted_index_list
#
#
# def take_columns_biopy_alignment (index_list, biopy_MSA):
#
# return biopy_MSA_subset

# In[5]:



DOMAIN_FASTA_PATH = RES_PATH + '/PKAinFASTA.txt'

DOMAINS_ALINGMENT_PATH = RES_PATH + '/KSalign.fa'

KINSPECT_CSV_PATH = RES_PATH + '/KScsv.csv'

#executables

MUSCLE_PATH = EXE_PATH + '/muscle_i86linux32'

PHYML_PATH = EXE_PATH + '/PhyML_linux32'


GOIs_csv_path = INFILE

run_id = re.split('[/|.]', INFILE)[INFILE_NO_OF_SUBDIR + 1] + '2'


#output files:

node_atg_dict_path = DATA_PATH + '/{}_node_atg_dict.pickle'.format(
    ↪ run_id)

DoS_lists_dict_path = DATA_PATH + '/{}_DoS_list_dict.pickle'.format(
    ↪ run_id)
```

```
full_sequence_path_dict = {

"fasta" : DATA_PATH + '/{}.fa'.format(run_id),

"aln" : DATA_PATH + '/{}.aln'.format(run_id),

"phy" : DATA_PATH + '/{}.phy'.format(run_id),

}

kinase_domain_path_dict = {

"UC.fasta" : DATA_PATH + '/{}_KD_UC.fasta'.format(run_id),

"UC.aln" : DATA_PATH + '/{}_KD_UC.aln'.format(run_id),

"aln" : DATA_PATH + '/{}_KD.aln'.format(run_id),

"phy" : DATA_PATH + '/{}_KD.phy'.format(run_id),

}

domains_of_specificity_path_dict = {

"UC.fasta" : DATA_PATH + '/{}_KD_DoS_UC.fasta'.format(run_id),

"UC.aln" : DATA_PATH + '/{}_KD_DoS_UC.aln'.format(run_id),

"aln" : DATA_PATH + '/{}_KD_DoS.aln'.format(run_id),

"phy" : DATA_PATH + '/{}_KD_DoS.phy'.format(run_id),

}



# In[6]:


node_atg_dict = kpr.read_phytozome_csv_to_MSA_plus_dicts (INFILE,
    ↪ full_sequence_path_dict["fasta"], additional_annotation_dict =
    ↪ additional_annotation_dict)

display(node_atg_dict)


with open(node_atg_dict_path, 'wb') as handle:
```

```
    pickle.dump(node_atg_dict, handle, protocol=pickle.
        ↪ HIGHEST_PROTOCOL)



# Make alignment and tree of genes of interest

# In[ ]:


print("External process MUSCLE started")

popen_cline = [MUSCLE_PATH,
                "-in", full_sequence_path_dict['fasta'],
                "-out", full_sequence_path_dict['aln'],
                "-clw",
                "-loga", run_id + "_full_seq_kpr_" + "MUSCLElog.txt",
                "-verbose",
                "-nocore"]

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()

print("External process MUSCLE ended")

bio.AlignIO.convert(full_sequence_path_dict['aln'], "clustal",
    ↪ full_sequence_path_dict['phy'], "phylip-relaxed")

"""
popen_cline = [PHYML_PATH,
                "--input", full_sequence_path_dict['phy'],
                "--datatype", 'aa',
                "--model", 'Blosum62',
                "--bootstrap", "3"] #was 100

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()
"""


# Make alingment of genes and domain

# In[6]:
```

```
#global GOIs_seqrec_list

GOIs_seqrec_list = list(bio.SeqIO.parse(full_sequence_path_dict["fasta"], "
    ↪ fasta"))

#global DOI_seqrec

DOI_seqrec = bio.SeqIO.read(DOMAIN_FASTA_PATH, "fasta")

GOIs_seqrec_list.append(DOI_seqrec)

bio.SeqIO.write(GOIs_seqrec_list, kinase_domain_path_dict["UC.fasta"], "
    ↪ fasta")

print("External process MUSCLE -started")

popen_cline = [MUSCLE_PATH,
               "-in", kinase_domain_path_dict["UC.fasta"],
               "-out", kinase_domain_path_dict["UC.aln"],
               "-clw",
               "-loga", run_id + "_KD_kpr_" + "MUSCLElog.txt",
               "-verbose",
               "-nocore"]

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()

print("External process MUSCLE -ended")


# Cut away parts of genes that do not align with domain

# In[7]:



#global GOIs_DOI_MSA

GOIs_DOI_MSA = bio.AlignIO.read (kinase_domain_path_dict["UC.aln"],
    ↪ "clustal")

GOIs_DOI_MSA_cut = kpr.cut_out_aligned_to_domain (GOIs_DOI_MSA,
    ↪ DOI_seqrec.id)

GOIs_MSA_cut = kpr.remove_id_from_MSA (GOIs_DOI_MSA_cut,
    ↪ DOI_seqrec.id)
```

```
bio.AlignIO.write (GOIs_MSA_cut, kinase_domain_path_dict["aln"], "
    ↪ clustal")


bio.AlignIO.convert(kinase_domain_path_dict["aln"], "clustal",
    ↪ kinase_domain_path_dict["phy"], "phylip-relaxed")


"""
popen_cline = [PHYML_PATH,
               "--input", kinase_domain_path_dict["phy"],
               "--datatype", 'aa',
               "--model", 'Blosum62',
               "--bootstrap", "3"] #was100

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()
"""


# Get kinase domain sequence used in KINspec paper that is closest to
    ↪ first item in genes_of_interest_MSA_list (keeping alignment
    ↪ produced in the paper)

# In[8]:



#global GOI_seqrec

GOI_seqrec = GOIs_seqrec_list[0]

#global DOIs_MSA

DOIs_MSA = bio.AlignIO.read(DOMAINS_ALINGMENT_PATH, "fasta")

#global closest_relative_of_GOI_in_DOIs_seqrec

closest_relative_of_GOI_in_DOIs_seqrec = kpr.
    ↪ get_closest_relative_to_sequence_from_alignment (GOI_seqrec,
    ↪ DOIs_MSA)

#THIS TAKES LONG


# Make alignment of KINspect_closest and genes_of_interest_MSA_list
```

```
# In[9]:



GOIs_seqrec_list.append(closest_relative_of_GOI_in_DOIs_seqrec)

bio.SeqIO.write(GOIs_seqrec_list, domains_of_specificity_path_dict["UC.
    ↪ fasta"], "fasta") #here pka ends up in

print("External process MUSCLE -started")

popen_cline = [MUSCLE_PATH,
               "-in", domains_of_specificity_path_dict["UC.fasta"],
               "-out", domains_of_specificity_path_dict["UC.aln"],
               "-clw",
               "-loga", run_id + "_DOS_kpr_" + "MUSCLElog.txt",
               "-verbose",
               "-nocore"]

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()

print("External process MUSCLE -ended")

GOIs_DOI_MSA = bio.AlignIO.read (domains_of_specificity_path_dict["UC
    ↪ .aln"], "clustal")

#global GOIs_DOI_MSA

GOIs_DOI_MSA = kpr.remove_id_from_MSA (GOIs_DOI_MSA,
    ↪ DOI_seqrec.id)


# In[27]:


"""
Get DOS inices from supplemental data
"""

KINspect_csv = numpy.genfromtxt(KINSPECT_CSV_PATH, delimiter=",
    ↪ ")

#global DOS_index_list

DOS_index_list = numpy.where(KINspect_csv[:,1] > 0.8)
```

```
DOS_index_list = DOS_index_list[0]

#i need to delete entries that do not correspond


# In[33]:



"""
---
"""

converted_DOS_index_list, corresponding_DoS_index_list = kpr.
    ↪ convert_indices_alignment_to_alignment(DOS_index_list,
    ↪ DOIs_MSA, GOIs_DOI_MSA,
    ↪ closest_relative_of_GOI_in_DOIs_seqrec.id)

DoS_lists_dict = {"converted_DOS_index_list": converted_DOS_index_list,
                  "corresponding_DoS_index_list":
                      ↪ corresponding_DoS_index_list}



if ID_STRUCT_PROT != None:

    for seqrec in GOIs_DOI_MSA:
        if seqrec.id == ID_STRUCT_PROT:
            struct_prot_map = kpr.create_map_gapped_ungapped(seqrec)

    struct_prot_DoS = []

    for i in converted_DOS_index_list:
        try:
            ungapped_i = structure_protein_map[i]
        except:
            continue
        struct_prot_DoS.append(ungapped_i)

    DoS_lists_dict["struct_prot_DoS"] = struct_prot_DoS

    print (struct_prot_DoS)

with open(DoS_lists_dict_path, 'wb') as handle:
    pickle.dump(DoS_lists_dict, handle, protocol=pickle.
        ↪ HIGHEST_PROTOCOL)
```

```
GOIs_DOI_MSA_DOSsubset = kpr.take_columns_biopy_alignment (
    ↪ converted_DOS_index_list, GOIs_DOI_MSA)

GOIs_MSA_DOSsubset = kpr.remove_id_from_MSA (
    ↪ GOIs_DOI_MSA_DOSsubset,
    ↪ closest_relative_of_GOI_in_DOIs_seqrec.id)

"""
Make MSA object of onkly the amino acids of the DoS indices
"""

bio.AlignIO.write (GOIs_MSA_DOSsubset,
    ↪ domains_of_specificity_path_dict["aln"], "clustal")

bio.AlignIO.convert(domains_of_specificity_path_dict["aln"], "clustal",
    ↪ domains_of_specificity_path_dict["phy"], "phylip-relaxed")

"""
popen_cline = [PHYML_PATH,
               "--input", domains_of_specificity_path_dict["phy"],
               "--datatype", 'aa',
               "--model", 'Blosum62',
               "--bootstrap", "3"] #was 100

child_process = sp.Popen(popen_cline, stdout=sp.PIPE)

stdout_str, stderr_str = child_process.communicate()
"""
```

S2.1.3 PRBPviz-notebook

Listing A.6: prbp_viz.ipynb

```
# coding: utf-8

# In[1]:


PROJECTS_PATH = 'C://Users/Christa/lkbm/projects/kpr/src'


# In[2]:
```

```
# %load /lkbm/projects/eteforkpr/ETEforKPR.py
# Import Tree instance and faces module
#from ete3 import Tree, faces, TreeStyle
from Bio import AlignIO,SeqIO
from funcy import func_partial as partial
import re
import pickle
import pandas as pd
import numpy as np
#from ete3 import Tree, faces, TreeStyle, NodeStyle
#from ete3 import ClusterTree, RectFace, AttrFace, ProfileFace,
    ↪ TextFace
#from ete3.treeview.faces import add_face_to_node
import colorsys
from IPython.display import display, HTML,Image
import matplotlib.pylab as plt
import matplotlib.image as mpimg
from matplotlib import gridspec
import ipywidgets as widgets
import subprocess as sp
import sys
sys.path.append(PROJECTS_PATH)

"""
module_path = os.path.abspath(os.path.join('...'))
if module_path not in sys.path:
    sys.path.append(module_path)
"""

import KinasePhyloRemote4 as kpr

get_ipython().magic('load_ext_RWinOut')
get_ipython().magic('R_require("ggplot2")')

#from ete3fork import faces
#from ete3fork.coretype.tree import Tree
#from ete3fork.treeview.main import TreeStyle


# In[3]:


RUN_ID = "D6PKkdAT2511172"

alignment_path = RUN_ID + '_KD.aln'

brinkworth_df_path = 'brinkworthdosada4.csv'
```

```
charge_subst_matrix_df_path = 'aachargearray.csv'#'AAchargeCHANGE.
    ↪ csv'

amino_acid_propert_df_path = 'aaprop.csv'

pin_fasta_path = 'ARApin.fa'

blosum62_subst_matrix_df_path = 'blosum62.csv'


# In[4]:


Pminus3_path = 'Pminus3.png'

Pminus2_path = 'Pminus2.png'

Pminus1_path = 'Pminus1.png'

Pminus0_path = 'Pminus0.png'

Pplus1_path = 'Pplus1.png'

Pplus2_path = 'Pplus2.png'

Pplus3_path = 'Pplus3.png'

PIN_path= 'PIN.png'

PIN1_path= 'PIN1.png'

PIN2_path= 'PIN2.png'

PIN3_path= 'PIN3.png'

PIN4_path= 'PIN4.png'

PIN5_path= 'PIN5.png'

PIN6_path= 'PIN6.png'

PIN7_path= 'PIN7.png'



get_ipython().magic('Rpush Pminus3_path')
```

```
get_ipython().magic('Rpush_Pminus2_path')

get_ipython().magic('Rpush_Pminus1_path')

get_ipython().magic('Rpush_Pminus0_path')

get_ipython().magic('Rpush_Pplus1_path')

get_ipython().magic('Rpush_Pplus2_path')

get_ipython().magic('Rpush_Pplus3_path')

get_ipython().magic('Rpush_PIN_path')

get_ipython().magic('Rpush_PIN1_path')

get_ipython().magic('Rpush_PIN2_path')

get_ipython().magic('Rpush_PIN3_path')

get_ipython().magic('Rpush_PIN4_path')

get_ipython().magic('Rpush_PIN5_path')

get_ipython().magic('Rpush_PIN6_path')

get_ipython().magic('Rpush_PIN7_path')


# In[5]:


Biopy_MSA = AlignIO.read(alignment_path, 'clustal')



#Create Pandas Dataframe with multiple sequence alignment (MSA) of
    ↪ D6PKs and PIDs (/AGCs).

# In[6]:

DOS_msa_dict = kpr.biopy_MSA_to_dict_np_arrays(Biopy_MSA)

DOS_msa_dict = DOS_msa_dict[0]

DoS_msa_df = pd.DataFrame(DOS_msa_dict)

DoS_msa_df.index.name = RUN_ID + '_KD'
```

```
DoS_msa_df_ = DoS_msa_df.loc[: , ['WAG1', 'WAG2', 'PID', 'PID2', '
    ↪ D6PK', 'D6PKL1', 'D6PKL2', 'D6PKL3']]

#or:
#DoS_msa_df = DoS_msa_df.iloc[:, 31:]
#DoS_msa_df = DoS_msa_df.iloc[:, 6:31]

DoS_msa_df = DoS_msa_df.reset_index()

display(DoS_msa_df)


#Reading in data from Brinkworth et al 200x

# In[7]:


DoS_ref_df = pd.DataFrame.from_csv(brinkworth_df_path)

display (DoS_ref_df)


#Merge Brinkworth data with MSA

# In[8]:


DoS_merge_df = pd.merge(DoS_msa_df, DoS_ref_df, how='right', on=[
    ↪ RUN_ID + '_KD'])

DoS_merge_df = DoS_merge_df.set_index(list(DoS_ref_df.columns.values))

display (DoS_merge_df)


#Group merged dataframe by Phospho-site Residue Binding Pockets (
    ↪ PRBPs given by Brinkworth data)

# In[9]:


DoS_merge_df_gb = DoS_merge_df.groupby('P-site')

display(DoS_merge_df_gb.get_group('P-3').T)


#Load amino acid properties dataframe
```

```
# In[10]:


aaprop_df = pd.DataFrame.from_csv('aaprop.csv', encoding='iso-8859-1'
    ↪ )

display (aaprop_df )


#Create sub dataframes for ONE PRBPs form all kinases!
#Further, get aa properties for each residue in this PRBP for every
    ↪ kinase

# In[11]:


def prepare_df(psite):

    df = DoS_merge_df_gb.get_group(psite)

    df.columns.name = 'ProteinID'

    df_stacked = df.stack()

    df_stacked = df_stacked.to_frame(name='1Letter').reset_index()

    stacked_merge_df = pd.merge(df_stacked, aaprop_df, how='left', on='1
        ↪ Letter')

    stacked_merge_df = stacked_merge_df.dropna()

    return stacked_merge_df

Pminus3_df = prepare_df('P-3')

Pminus2_df = prepare_df('P-2')

Pminus1_df = prepare_df('P-1')

Pminus0_df = prepare_df('P-0')

Pplus1_df = prepare_df('P+1')

Pplus2_df = prepare_df('P+2')

Pplus3_df = prepare_df('P+3')
```

```
get_ipython().magic('Rpush_Pminus3_df')

get_ipython().magic('Rpush_Pminus2_df')

get_ipython().magic('Rpush_Pminus1_df')

get_ipython().magic('Rpush_Pminus0_df')

get_ipython().magic('Rpush_Pplus1_df')

get_ipython().magic('Rpush_Pplus2_df')

get_ipython().magic('Rpush_Pplus3_df')

# Use ggpubr from R to create ballon plot

# In[12]:

### use %%R (R magic) to run R code given in "PRBPviz R-Cell 2"
    ↪ for all different P-sites


#Read in MSA of all Arabidopsis PINs (PIN-MSA)

# In[20]:


PIN_aln = AlignIO.read ('ARApin.aln', "clustal")

PIN_aln_dict_arrays_annot_tuple = kpr.biopy_MSA_to_dict_np_arrays(
    ↪ PIN_aln)

PIN_aln_df = pd.DataFrame(PIN_aln_dict_arrays_annot_tuple[0])


# Use KSselect functions to get all PIN phospho-sites postions in the
    ↪ PIN-MSA using PIN1 phospho-site positions!

# This will be per PIN

# In[20]:

PIN1_S_sites_dict = {'S5': 215,
                     'S1': 231,
                     'S2': 252,
                     'S4': 271,
                     'S3': 290,
```

```
                    'S209': 209,
                    'S212': 212,
                    'S337': 337
                   }

PIN1_S_sites_df = pd.DataFrame(PIN1_S_sites_dict, index = ['PIN1-
    ↪ indices-unaligned'])

PIN1_S_sites_df = PIN1_S_sites_df.T

#--

PIN_map_gapped_ungapped_dict = {}

for seqrec in PIN_aln:

    map_gapped_ungapped = kpr.create_map_gapped_ungapped(seqrec)

    PIN_map_gapped_ungapped_dict[seqrec.id] = map_gapped_ungapped

#--

PIN1_map_ungapped_gapped = {}

for k in PIN_map_gapped_ungapped_dict['PIN1.']:
    PIN1_map_ungapped_gapped[PIN_map_gapped_ungapped_dict['PIN1.'
        ↪ ][k]] = k

#--

def look_up_entry_in_map (entry, map_dict):
    try:
        ret = map_dict[entry]

    except:
        ret = np.NaN

    return ret

PIN1_S_sites_df['PIN1-indices-aligned'] = PIN1_S_sites_df['PIN1-indices
    ↪ -unaligned'].apply(partial(look_up_entry_in_map, map_dict =
    ↪ PIN1_map_ungapped_gapped))

for k in PIN_map_gapped_ungapped_dict:
    PIN1_S_sites_df[k] = PIN1_S_sites_df['PIN1-indices-aligned'].apply(
        ↪ partial(look_up_entry_in_map, map_dict =
        ↪ PIN_map_gapped_ungapped_dict[k]))
```

```
PIN1_S_sites_df = PIN1_S_sites_df.drop(['PIN1-indices-unaligned', 'PIN1
    ↪ -indices-aligned'], 1)

PIN1_S_sites_df = PIN1_S_sites_df.stack().to_frame('4.P-0')

display(PIN1_S_sites_df)



#

#In[21]:

P_site_dict = {
                  '1.P-3': -3,
                  '2.P-2': -2,
                  '3.P-1': -1,
                  '5.P+1': -1,
                  '6.P+2': 2,
                  '7.P+3': 3,
}

for k in P_site_dict:
    PIN1_S_sites_df[k] = PIN1_S_sites_df['4.P-0'] + P_site_dict[k]


PIN1_S_sites_df = PIN1_S_sites_df[['1.P-3', '2.P-2', '3.P-1', '4.P-0', '5.
    ↪ P+1', '6.P+2', '7.P+3']]

PIN1_S_sites_df = PIN1_S_sites_df.reindex(PIN1_S_sites_df.index.rename(['
    ↪ S-site', 'protein']))

PIN_aln_df = PIN_aln_df.T

PIN_list = list(PIN_aln_df.index)

#--

PIN_dict_raw = SeqIO.to_dict(SeqIO.parse('ARApin.fa', "fasta"))

PIN_dict = {}

for k in PIN_dict_raw:
    seq = PIN_dict_raw[k]._seq._data
    PIN_dict[k] = np.array(list(seq))
```

```
AA_list = []

def create_look_up_aas_func(seq_series):
    def look_up_aas (index):
        return seq_series[int(index-1)]
    return look_up_aas


for pin in PIN_list:
    df = PIN1_S_sites_df.groupby('protein').get_group(pin)

    aas = PIN_dict[pin]

    look_up_aas = create_look_up_aas_func(aas)

    df = df.apply(lambda series: series.apply(look_up_aas))

    AA_list.append (df)

AA_df = pd.concat(AA_list)

# cut out all entries where phosphorylated residue is not S

AA_df = AA_df[AA_df.loc[:,'4.P-0'] == 'S']

display(AA_df)


#

# In[22]:

AA_df.columns.name = 'P-site'

AA_df_stacked = AA_df.stack()

AA_df_stacked = AA_df_stacked.to_frame(name='1Letter').reset_index()

AA_df_stacked_merge = pd.merge(AA_df_stacked, aaprop_df, how='left',
    ↪ on='1Letter')

AA_df_stacked_merge = AA_df_stacked_merge.dropna()

display(AA_df_stacked_merge)

AA_df_stacked_merge['id'] = AA_df_stacked_merge['protein'] +
    ↪ AA_df_stacked_merge['S-site']
```

```
get_ipython().magic('Rpush AA_df_stacked_merge')


#Get dataframes for all phospho-site in each PIN

# In[23]:


pin_gb = AA_df_stacked_merge.groupby('protein')


PIN1_df = pin_gb.get_group('PIN1.')

PIN2_df = pin_gb.get_group('PIN2.')

PIN3_df = pin_gb.get_group('PIN3.')

PIN4_df = pin_gb.get_group('PIN4.')

PIN6_df = pin_gb.get_group('PIN6.')

PIN7_df = pin_gb.get_group('PIN7.')



get_ipython().magic('Rpush PIN1_df')

get_ipython().magic('Rpush PIN2_df')

get_ipython().magic('Rpush PIN3_df')

get_ipython().magic('Rpush PIN4_df')

get_ipython().magic('Rpush PIN6_df')

get_ipython().magic('Rpush PIN7_df')


# In[24]:

### use %%R (R magic) to run R code given in "PRBPviz R-Cell 2"
    ↪ for all different PINs
```

Listing A.7: PRBPviz R-Cell 1

```
library(ggplot2)
library(ggpubr)
```

```
png(Pminus3_path,width = 30, height = 30, units ="cm", res = 350)
plot_1 <- ggballoonplot(Pminus3_df, x = "Residue", y = "ProteinID",
                  size = "MolecularW", fill = "Charge", font.tickslab = c(20,
                      ↪ 'bold', 'black'),
                              font.legend = c(20, 'bold', 'black')) +
    scale_fill_gradientn(colours = c("blue", "white", "red"),breaks = c(-1,
        ↪  0, 1), limits = c(-1,1))

print(plot_1)

dev.off()
```

Listing A.8: PRBPviz R-Cell 2

```
library(ggplot2)
library(ggpubr)

png(PIN1_path,width = 30, height = 30, units ="cm", res = 350)
plot_1 <- ggballoonplot(PIN1_df, x = "P.site", y = "S.site",
                  size = "MolecularW", fill = "Charge", font.tickslab = c(30,
                      ↪ 'bold', 'black'),
                              font.legend = c(20, 'bold', 'black')) +
    scale_fill_gradientn(colours = c("blue", "white", "red"),breaks = c(-1,
        ↪  0, 1), limits = c(-1,1))

print(plot_1)

dev.off()
```

S2.1.4 ETE3-notebooks

Listing A.9: ks_viz.ipynb

```
# coding: utf-8

# In[1]:


# %load /lkbm/projects/eteforkpr/ETEforKPR.py
# Import Tree instance and faces module
#from ete3 import Tree, faces, TreeStyle
from Bio import AlignIO
from funcy import func_partial as partial
```

```
import re
import pickle
import pandas as pd
import numpy as np
from ete3 import Tree, faces, TreeStyle, NodeStyle
from ete3 import ClusterTree, RectFace, AttrFace, ProfileFace, TextFace
from ete3.treeview.faces import add_face_to_node
import colorsys
from IPython.display import display, HTML,Image
import matplotlib.pylab as plt
import matplotlib.image as mpimg
from matplotlib import gridspec
import ipywidgets as widgets


#from ete3fork import faces
#from ete3fork.coretype.tree import Tree
#from ete3fork.treeview.main import TreeStyle


# In[2]:


RUN_ID = "CIPKatPROT111117"
DATA_DIR = '/lkbm/data/eteforkpr/'

RES_DIR = '/lkbm/projects/eteforkpr/res/'
KINASE_VIZ_FULL_PATH = RES_DIR + 'KINASEviz/kinaseDOSviz3a.
    ↪ png'
KINASE_VIZ_KD_PATH = RES_DIR + 'KINASEviz/kinaseDOSviz3b.
    ↪ png'
KINASE_VIZ_DOS_PATH = RES_DIR + 'KINASEviz/kinaseDOSviz3c.
    ↪ png'

NODE_ATG_DICT_PATH = DATA_DIR + '{}_node_atg_dict.pickle'.
    ↪ format(RUN_ID)


# In[3]:


#input files:

tree_path_dict = {

"full_sequence" : DATA_DIR + '{0}full/RAxML_bestTree.{0}full'.format(
    ↪ RUN_ID),
```

```
"kinase_domain" : DATA_DIR + '{0}kd/RAxML_bestTree.{0}kd'.format(
    ↪ RUN_ID),

"domains_of_specificity" : DATA_DIR + '{0}dos/RAxML_bestTree.{0}dos
    ↪ '.format(RUN_ID),

}
###phyml give newick format back, there is a .txt missing at the end.

aln_path_dict = {

"full_sequence" : DATA_DIR + '{}.aln'.format(RUN_ID),

"kinase_domain" : DATA_DIR + '{}_KD.aln'.format(RUN_ID),

"domains_of_specificity" : DATA_DIR + '{}_KD_DoS.aln'.format(RUN_ID
    ↪ )

}


# In[4]:


Biopy_MSA_dict = {}

for key in aln_path_dict:

    Biopy_MSA = AlignIO.read(aln_path_dict[key], 'clustal')

    Biopy_MSA_dict[key] = Biopy_MSA



# In[5]:


#layout functions, think of them as to be used with apply or map


def layout_pattern (node, id_species_dict = None, Biopy_MSA = None,
    ↪ seq_format = "()"): # 'seq_format = "seq"' for closeup

    if node.is_leaf():

        if id_species_dict != None:

            #longNameFace = faces.TextFace(name_dict[node.name])
```

```
            #faces.add_face_to_node(longNameFace, node, column=0)

            imageFace = faces.ImgFace(species_png_path_dict[
                ↪ id_species_dict[node.name]])
            #imageFace.margin_border.width = 1
            #imageFace.margin_bottom = 5
            #imageFace.margin_top = 5
            #imageFace.margin_right = 10
            faces.add_face_to_node(imageFace, node, column=0)

            speciesFace = faces.TextFace(id_species_dict[node.name], fsize
                ↪  = 10, fgcolor="#009000")
            #imageFace.margin_border.width = 1
            #imageFace.margin_bottom = 5
            #imageFace.margin_top = 5
            #imageFace.margin_right = 10
            faces.add_face_to_node(speciesFace, node, column=1)

            idFace = faces.TextFace(node.name, fsize = 2)
            #imageFace.margin_border.width = 1
            #imageFace.margin_bottom = 5
            #imageFace.margin_top = 5
            #imageFace.margin_right = 10
            faces.add_face_to_node(idFace, node, column=1)


            for seqrec in Biopy_MSA:

                if seqrec.id == node.name:

                    seqFace = SeqMotifFace(seqrec._seq._data, seq_format
                        ↪ =seq_format) #fsize?
                    faces.add_face_to_node(seqFace, node, column=0,
                        ↪ aligned=True)

        if id_species_dict == None:

            for seqrec in Biopy_MSA:

                if seqrec.id == node.name:

                    seqFace = faces.SeqMotifFace(seqrec._seq._data,
                        ↪ seq_format=seq_format) #fsize?
                    #margin_top, margin_ bottom, border.margin?
                    faces.add_face_to_node(seqFace, node, column=0,
                        ↪ aligned=True) #ud
```

```
# In[6]:


#partailly applied fuinctions:

layout_function_dict = {

'full_sequence' : partial(layout_pattern,
                          id_species_dict = None,
                          Biopy_MSA = Biopy_MSA_dict['full_sequence'],
                          seq_format = "()"),

'kinase_domain' : partial(layout_pattern,
                          id_species_dict = None,
                          Biopy_MSA = Biopy_MSA_dict['kinase_domain'
                              ↪ ],
                          seq_format = "seq"),

'domains_of_specificity' : partial(layout_pattern,
                                   id_species_dict = None,
                                   Biopy_MSA = Biopy_MSA_dict['
                                       ↪ domains_of_specificity'],
                                   seq_format = "seq")

}


# In[8]:


out1 = widgets.Output()
out2 = widgets.Output()
out3 = widgets.Output()

tab_contents = [out1, out2, out3]
tab_names = ['Full_Protein', 'Kinase_Domain', 'Domains_of_Specificty']
tab = widgets.Tab()
tab.children = tab_contents
for i, name in enumerate (tab_names):
    tab.set_title(i, name)
tab


# In[7]:
```

```
# script:::
for key in tree_path_dict:

    t = Tree(tree_path_dict[key]) # format=1 -> internal node names
        ↪ are loaded
    # t.link_to_alignment(alg_path_dict[key]) this seems to restrict to
        ↪ closeup seq

    ts = TreeStyle()
    ts.scale = 100
    ts.layout_fn = layout_function_dict[key]
    t.render(DATA_DIR + RUN_ID + '_' + key + "_tree_fig3.png",
        ↪ tree_style = ts, units = 'mm', h = 50, dpi = 350) #ud
```

Listing A.10: auxin_ete3.ipynb

```
# coding: utf-8

# Script for generating figures combining phylogenetic tree with data
    ↪ from Kilian et al. 2008

# In[1]:


from Bio import AlignIO
from funcy import func_partial as partial
import re
import pickle
import pandas as pd
import numpy as np
from ete3 import Tree, faces, TreeStyle, NodeStyle
from ete3 import ClusterTree, RectFace, AttrFace, ProfileFace, TextFace
from ete3.treeview.faces import add_face_to_node
import colorsys
from IPython.display import display, HTML


# In[2]:


RUN_ID = "CIPKatPROT111117"
TREE_PATH = '/lkbm/data/eteforkpr/{0}full/RAxML_bestTree.{0}full'.
    ↪ format(RUN_ID)
WORKING_DIR = '/projects/eteforkpr/'
DATA_DIR = '/media/usb0/data/eteforkpr/'
```

```
PATH_TO_AUXIN_MS_DATA = '/media/usb0/infiles/expr/IAA-lookup
    ↪ -ms.csv'
PATH_TO_AUXIN_EZ_DATA = '/media/usb0/infiles/expr/IAA-lookup
    ↪ -ez.csv'
NODE_ATG_DICT_PATH = '/lkbm/data/eteforkpr/{}_node_atg_dict.
    ↪ pickle'.format(RUN_ID)


# In[3]:


auxin_ms_expression_data = pd.read_csv(PATH_TO_AUXIN_MS_DATA)

auxin_ez_expression_data = pd.read_csv(PATH_TO_AUXIN_EZ_DATA)

def process_auxin_data (auxin_expression_data):

    auxin_expression_data = (auxin_expression_data.set_index('Unnamed:␣
        ↪ 0')
        .T
    .reset_index())

    auxin_expression_data['genotype'] = auxin_expression_data.apply((
        ↪ lambda row: row['index'].split('_')[0]), axis = 1)

    auxin_expression_data['timepoint'] = auxin_expression_data.apply((
        ↪ lambda row: int(row['index'].split('_')[1])), axis = 1)

    auxin_expression_data['treatment'] = auxin_expression_data.apply((
        ↪ lambda row: row['index'].split('_')[2]), axis = 1)

    auxin_expression_data = (auxin_expression_data.drop('index', axis =
        ↪ 1)
    .set_index(['genotype','treatment','timepoint']))

    auxin_expression_data_genot = auxin_expression_data.groupby('
        ↪ genotype')

    auxin_expression_data_wt = auxin_expression_data_genot.get_group('
        ↪ WT')
    auxin_expression_data_wt.index = auxin_expression_data_wt.index.
        ↪ droplevel()

    auxin_expression_data_arf = auxin_expression_data_genot.get_group('
        ↪ arf')
    auxin_expression_data_arf.index = auxin_expression_data_arf.index.
        ↪ droplevel()
```

```
    auxin_expression_data_wt_treatm = auxin_expression_data_wt.groupby
        ↪ ('treatment')

    auxin_expression_data_wt_cont = auxin_expression_data_wt_treatm.
        ↪ get_group('cont')
    auxin_expression_data_wt_cont.index = auxin_expression_data_wt_cont.
        ↪ index.droplevel()

    auxin_expression_data_wt_iaa = auxin_expression_data_wt_treatm.
        ↪ get_group('iaa')
    auxin_expression_data_wt_iaa.index = auxin_expression_data_wt_iaa.
        ↪ index.droplevel()

    auxin_expression_data_arf_treatm = auxin_expression_data_arf.groupby
        ↪ ('treatment')

    auxin_expression_data_arf_cont = auxin_expression_data_arf_treatm.
        ↪ get_group('cont')
    auxin_expression_data_arf_cont.index = auxin_expression_data_arf_cont
        ↪ .index.droplevel()

    auxin_expression_data_arf_iaa = auxin_expression_data_arf_treatm.
        ↪ get_group('iaa')
    auxin_expression_data_arf_iaa.index = auxin_expression_data_arf_iaa.
        ↪ index.droplevel()

    auxin_induction_wt = auxin_expression_data_wt_iaa -
        ↪ auxin_expression_data_wt_cont

    auxin_induction_arf = auxin_expression_data_arf_iaa -
        ↪ auxin_expression_data_arf_cont

    arf_dep_auxin_induction = auxin_induction_wt - auxin_induction_arf


    return auxin_induction_wt.sort_index().T, arf_dep_auxin_induction.
        ↪ sort_index().T


auxin_induction_ms, arf_dep_auxin_induction_ms = process_auxin_data(
    ↪ auxin_ms_expression_data)

auxin_induction_ez, arf_dep_auxin_induction_ez = process_auxin_data(
    ↪ auxin_ez_expression_data)



# In[4]:
```

```
def look_up_agi (row, atg_node_dict):
    atg_code = row.loc['Unnamed:_0']
    name = atg_node_dict[atg_code]
    return name


def prepare_dataframe_for_ete (df, node_atg_dict):

    atg_list = []

    atg_node_dict = {}

    for k in node_atg_dict:
        atg_list.append(node_atg_dict[k])
        atg_node_dict[node_atg_dict[k]] = k


    df_cut = (df.dropna(axis = 1)
              .loc[atg_list,:]
              .reset_index())

    look_up_agi_pa = partial(look_up_agi, atg_node_dict = atg_node_dict)

    df_cut["#Names"] = df_cut.apply(look_up_agi_pa, axis = 1)

    header = list(df.columns.values)

    df_cut = (df_cut.set_index("#Names")
                    .drop('Unnamed:_0', axis = 1) #set index before???
                    .to_csv(None, sep="\t", float_format="%.2f"))

    return df_cut, header


with open(NODE_ATG_DICT_PATH, 'rb') as handle:
    node_atg_dict = pickle.load(handle)

node_atg_dict = node_atg_dict[0]



ai_ms, ai_ms_header = prepare_dataframe_for_ete(auxin_induction_ms,
    ↪ node_atg_dict)
arf_dep_ai_ms, arf_dep_ai_ms_header = prepare_dataframe_for_ete (
    ↪ arf_dep_auxin_induction_ms, node_atg_dict)
```

```
ai_ez, ai_ez_header = prepare_dataframe_for_ete (auxin_induction_ez,
    ↪ node_atg_dict)
arf_dep_ai_ez, arf_dep_ai_ez_header = prepare_dataframe_for_ete (
    ↪ arf_dep_auxin_induction_ez, node_atg_dict)



#This code is from blog.genesino.com/2016/07/ete/

# In[5]:


nameFace = AttrFace("name", fsize=12) #Set leaf node attribute

def setup_heatmap(tree, tree_style, header, center_value=0.0, nameMap
    ↪ ={}, nameLabel = '', color_up=0.7, color_down=0.2, color_center=
    ↪ "white"):
        DEFAULT_COLOR_SATURATION = 0.5
        BASE_LIGHTNESS = 0.7
        def gradient_color(value, max_value, saturation=0.5, hue=0.1):
                def rgb2hex(rgb):
                        return '#%02x%02x%02x' % rgb
                def hls2hex(h, l, s):
                        return rgb2hex( tuple(map(lambda x: int(x*255),
                            ↪  colorsys.hls_to_rgb(h, l, s))))

                lightness = 1 - (value * BASE_LIGHTNESS) /
                    ↪ max_value
                return hls2hex(hue, lightness,
                    ↪ DEFAULT_COLOR_SATURATION)


        # Calculate max gradient value from the ClusterTree matrix
        maxv = abs(center_value - tree.arraytable._matrix_max)
        minv = abs(center_value - tree.arraytable._matrix_min)
        if center_value <= tree.arraytable._matrix_min:
                MAX_VALUE = minv + maxv
        else:
                MAX_VALUE = max(maxv, minv)

        # Add heatmap colors to tree
        cols_add_before_heat = 0
        if nameMap:
                cols_add_before_heat = 1
        for lf in tree:
                if nameMap:
                        longNameFace = faces.TextFace(nameMap.get(lf.
                            ↪ name, lf.name))
```

```
                        lf.add_face(longNameFace, column=0, position="
                            ↪ aligned")

                for i, value in enumerate(getattr(lf, "profile", [])):
                        if value > center_value:
                                color = gradient_color(abs(center_value -
                                    ↪  value), MAX_VALUE, hue=
                                    ↪ color_up)
                        elif value < center_value:
                                color = gradient_color(abs(center_value -
                                    ↪  value), MAX_VALUE, hue=
                                    ↪ color_down)
                        else:
                                color = color_center
                        lf.add_face(RectFace(20, 20, color, color), position
                            ↪ ="aligned", column=i+
                            ↪ cols_add_before_heat)
                        # Uncomment to add numeric values to the
                            ↪ matrix
                        #lf.add_face(TextFace("%0.2f "%value, fsize=5),
                            ↪ position="aligned", column=i)
                lf.add_face(nameFace, column=i+cols_add_before_heat+1,
                    ↪  position="aligned")

        if nameMap and nameLabel:
                nameF = TextFace(nameLabel, fsize=7)
                #nameF.rotation = -90
                tree_style.aligned_header.add_face(nameF, column=0)
        # Add header
        for i, name in enumerate(header):
                nameF = TextFace(name, fsize=7)
                nameF.rotation = -90
                tree_style.aligned_header.add_face(nameF, column=i+
                    ↪ cols_add_before_heat)


# Auxin induction meristem

# In[6]:


t = ClusterTree(TREE_PATH, ai_ms, ai_ms_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True
```

```
setup_heatmap(t, ts, ai_ms_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name="%%inline", tree_style=ts)


# Arf dependent auxin induction meristem:

# In[7]:


t = ClusterTree(TREE_PATH, arf_dep_ai_ms, arf_dep_ai_ms_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, arf_dep_ai_ms_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name="%%inline", tree_style=ts)


# Auxin induction elongaton zone

# In[8]:



t = ClusterTree(TREE_PATH, ai_ez, ai_ez_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, ai_ez_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name="%%inline", tree_style=ts)

Arf dependent auxin induction elongation zone
# In[9]:


t = ClusterTree(TREE_PATH, arf_dep_ai_ez, arf_dep_ai_ez_header)
```

```
ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, arf_dep_ai_ez_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name="%%inline", tree_style=ts)
```

Listing A.11: abio_ete3.ipynb

```
# coding: utf-8

# Script for generating figures combining phylogenetic tree with data
    ↪ from Kilian et al. 2008

# In[1]:


from Bio import AlignIO
from funcy import func_partial as partial
import re
import pickle
import pandas as pd
import numpy as np
from ete3 import Tree, faces, TreeStyle, NodeStyle
from ete3 import ClusterTree, RectFace, AttrFace, ProfileFace, TextFace
from ete3.treeview.faces import add_face_to_node
import colorsys
from IPython.display import display, HTML


# In[2]:


RUN_ID = "CIPKatPROT111117"
TREE_PATH = '/lkbm/data/eteforkpr/{0}full/RAxML_bestTree.{0}full'.
    ↪ format(RUN_ID)
WORKING_DIR = '/projects/eteforkpr/'
DATA_DIR = '/lkbm/eteforkpr/'
PATH_TO_ABIO_STRESS_DATA = '/media/usb0/infiles/expr/
    ↪ abioSTRESSdata.txt'
PATH_TO_ABIO_STRESS_INDEX = '/lkbm/infiles/expr/
    ↪ abioSTRESSindexPROC.txt'
NODE_ATG_DICT_PATH = '/lkbm/data/eteforkpr/{}_node_atg_dict.
```

```
    ↪ pickle'.format(RUN_ID)

#output paths

cold_root_path = '/lkbm/data/eteforkpr/{0}cold_root.png'.format(
    ↪ RUN_ID)

osm_root_path = '/lkbm/data/eteforkpr/{0}osm_root.png'.format(
    ↪ RUN_ID)

salt_root_path = '/lkbm/data/eteforkpr/{0}salt_root.png'.format(RUN_ID
    ↪ )

drought_root_path = '/lkbm/data/eteforkpr/{0}drought_root.png'.format(
    ↪ RUN_ID)

genot_root_path = '/lkbm/data/eteforkpr/{0}genot_root.png'.format(
    ↪ RUN_ID)

oxi_root_path = '/lkbm/data/eteforkpr/{0}oxi_root.png'.format(RUN_ID)

uvb_root_path = '/lkbm/data/eteforkpr/{0}uvb_root.png'.format(
    ↪ RUN_ID)

wound_root_path = '/lkbm/data/eteforkpr/{0}wound_root.png'.format(
    ↪ RUN_ID)

heat_root_path = '/lkbm/data/eteforkpr/{0}heat_root.png'.format(
    ↪ RUN_ID)


# In[3]:


abiotic_expression_data = pd.read_csv(PATH_TO_ABIO_STRESS_DATA,
    ↪ sep = '\t')

abiotic_expression_data = abiotic_expression_data[abiotic_expression_data['
    ↪ AGI_code'].notnull()]

abiotic_expression_data = abiotic_expression_data[abiotic_expression_data['
    ↪ AGI_code'].str.contains("At")]

abiotic_expression_data = (abiotic_expression_data.drop('Affy_Identifier',
    ↪ axis = 1)
                                                    .set_index('AGI_code')
                                                    .T
                                                    .reset_index())
```

```
# In[4]:


abiotic_expression_index = pd.read_csv(
    ↪ PATH_TO_ABIO_STRESS_INDEX, delim_whitespace=True)

def clean_exp_ID (row):
    return row.loc['Ccode'][1:]

abiotic_expression_index = abiotic_expression_index.drop('x', axis = 1)

abiotic_expression_index['index'] = abiotic_expression_index.apply(
    ↪ clean_exp_ID, axis = 1)

abiotic_expression_index = abiotic_expression_index.drop('Ccode', axis =
    ↪ 1)


# In[5]:


abiotic_expression_indexed_data = pd.merge(abiotic_expression_index,
    ↪ abiotic_expression_data, on='index')

abiotic_expression_indexed_data = abiotic_expression_indexed_data.
    ↪ set_index(list(abiotic_expression_index.columns))


# In[6]:


root_data = abiotic_expression_indexed_data.groupby('Organ').get_group('
    ↪ Roots')

root_data_grouped = root_data.groupby('Treatment')

contr_root_data = root_data_grouped.get_group('Control')
contr_root_data = contr_root_data.groupby('Hours').agg(np.mean)

def process_treatm_root_data (treatment_name, node_atg_dict):
    treatm_root_data = root_data_grouped.get_group(treatment_name)
    treatm_root_data = treatm_root_data.groupby('Hours').agg(np.mean)
    treatm_root_data = treatm_root_data - contr_root_data
    treatm_root_data = (treatm_root_data.dropna()
                                        .T
                                        .reset_index())
```

```
    treatm_root_data['new_index'] = treatm_root_data['index'].apply(
        ↪ lambda s: s.upper())

    treatm_root_data = (treatm_root_data.set_index('new_index')
                                        .drop ('index', axis = 1))

    atg_list = []

    atg_node_dict = {}

    for k in node_atg_dict:
        atg_list.append(node_atg_dict[k])
        atg_node_dict[node_atg_dict[k]] = k


    treatm_root_data_cut = (treatm_root_data.loc[atg_list,:]
                                            .reset_index())

    treatm_root_data_cut["#Names"] = treatm_root_data_cut['new_index'].
        ↪ apply(lambda n: atg_node_dict[n])

    header = list(treatm_root_data_cut.columns.values)

    treatm_root_data_cut = (treatm_root_data_cut.set_index("#Names")
                                                .drop('new_index', axis =
                                                    ↪  1)
                                                .to_csv(None, sep="\t",
                                                    ↪ float_format="
                                                    ↪ %.2f"))


    return treatm_root_data_cut, header


with open(NODE_ATG_DICT_PATH, 'rb') as handle:
    node_atg_dict = pickle.load(handle)

node_atg_dict = node_atg_dict[0]

cold_root_data, cold_root_header = process_treatm_root_data('Cold',
    ↪ node_atg_dict)

osm_root_data, osm_root_header = process_treatm_root_data('
    ↪ OsmoticStress', node_atg_dict)

salt_root_data, salt_root_header = process_treatm_root_data('SaltStress',
    ↪ node_atg_dict)
```

```
drought_root_data, drought_root_header = process_treatm_root_data('
    ↪ DroughtStress', node_atg_dict)

genot_root_data, genot_root_header = process_treatm_root_data('
    ↪ GenotoxicStress', node_atg_dict)

oxi_root_data, oxi_root_header = process_treatm_root_data('
    ↪ OxidativeStress', node_atg_dict)

uvb_root_data, uvb_root_header = process_treatm_root_data('UV-BStress'
    ↪ , node_atg_dict)

wound_root_data, wound_root_header = process_treatm_root_data('
    ↪ WoundingStress', node_atg_dict)

heat_root_data, heat_root_header = process_treatm_root_data('HeatStress',
    ↪  node_atg_dict)



##This code is from blog.genesino.com/2016/07/ete/

# In[7]:


nameFace = AttrFace("name", fsize=12) #Set leaf node attribute

def setup_heatmap(tree, tree_style, header, center_value=0.0, nameMap
    ↪ ={}, nameLabel = '',
                                        color_up=0.7, color_down=0.2,
                                            ↪ color_center="white"):
        DEFAULT_COLOR_SATURATION = 0.5
        BASE_LIGHTNESS = 0.7
        def gradient_color(value, max_value, saturation=0.5, hue=0.1):
                def rgb2hex(rgb):
                        return '#%02x%02x%02x' % rgb
                def hls2hex(h, l, s):
                        return rgb2hex( tuple(map(lambda x: int(x*255),
                            ↪  colorsys.hls_to_rgb(h, l, s))))

                lightness = 1 - (value * BASE_LIGHTNESS) /
                    ↪ max_value
                return hls2hex(hue, lightness,
                    ↪ DEFAULT_COLOR_SATURATION)


        # Calculate max gradient value from the ClusterTree matrix
        maxv = abs(center_value - tree.arraytable._matrix_max)
```

```
        minv = abs(center_value - tree.arraytable._matrix_min)
        if center_value <= tree.arraytable._matrix_min:
                MAX_VALUE = minv + maxv
        else:
                MAX_VALUE = max(maxv, minv)

        # Add heatmap colors to tree
        cols_add_before_heat = 0
        if nameMap:
                cols_add_before_heat = 1
        for lf in tree:
                if nameMap:
                        longNameFace = faces.TextFace(nameMap.get(lf.
                            ↪ name, lf.name))
                        lf.add_face(longNameFace, column=0, position="
                            ↪ aligned")

                for i, value in enumerate(getattr(lf, "profile", [])):
                        if value > center_value:
                                color = gradient_color(abs(center_value -
                                    ↪  value), MAX_VALUE, hue=
                                    ↪ color_up)
                        elif value < center_value:
                                color = gradient_color(abs(center_value -
                                    ↪  value), MAX_VALUE, hue=
                                    ↪ color_down)
                        else:
                                color = color_center
                        lf.add_face(RectFace(20, 20, color, color), position
                            ↪ ="aligned", column=i+
                            ↪ cols_add_before_heat)
                        # Uncomment to add numeric values to the
                            ↪ matrix
                        #lf.add_face(TextFace("%0.2f "%value, fsize=5),
                            ↪ position="aligned", column=i)
                lf.add_face(nameFace, column=i+cols_add_before_heat+1,
                    ↪  position="aligned")

        if nameMap and nameLabel:
                nameF = TextFace(nameLabel, fsize=7)
                #nameF.rotation = -90
                tree_style.aligned_header.add_face(nameF, column=0)
        # Add header
        for i, name in enumerate(header):
                nameF = TextFace(name, fsize=7)
                nameF.rotation = -90
                tree_style.aligned_header.add_face(nameF, column=i+
                    ↪ cols_add_before_heat)
```

```
# Cold stress data:

# In[8]:


t = ClusterTree(TREE_PATH, cold_root_data, cold_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, cold_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=cold_root_path, tree_style=ts)


# Osmotic stress data:

# In[9]:


t = ClusterTree(TREE_PATH, osm_root_data, osm_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, osm_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=osm_root_path, tree_style=ts)


# Salt stress data:

# In[10]:


t = ClusterTree(TREE_PATH, salt_root_data, salt_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
```

```
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, salt_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=salt_root_path, tree_style=ts)
```

Drought stress data:

```
# In[11]:


t = ClusterTree(TREE_PATH, drought_root_data, drought_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, drought_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=drought_root_path, tree_style=ts)


# Genotoxic stress data:

# In[12]:


t = ClusterTree(TREE_PATH, genot_root_data, genot_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, genot_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=genot_root_path, tree_style=ts)


# Oxidative stress data:

# In[13]:
```

```
t = ClusterTree(TREE_PATH, oxi_root_data, oxi_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, oxi_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=oxi_root_path, tree_style=ts)


# UV−B stress data:

# In[14]:


t = ClusterTree(TREE_PATH, uvb_root_data, uvb_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, uvb_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=uvb_root_path, tree_style=ts)


# Wound stress data:

# In[15]:


t = ClusterTree(TREE_PATH, wound_root_data, wound_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, wound_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=wound_root_path, tree_style=ts)
```

```
# Heat stress data:

# In[16]:


t = ClusterTree(TREE_PATH, heat_root_data, heat_root_header)

ts = TreeStyle()
ts.show_leaf_name = False
ts.show_branch_length = True
ts.show_branch_support = True

setup_heatmap(t, ts, heat_root_header, center_value=0, color_up=0.9,
    ↪ color_down=0.3, color_center="white")

t.render(file_name=heat_root_path, tree_style=ts)
```

S2.2 Modules

S2.2.1 lkbm-KSselect

Listing A.12: KinasePhyloRemote.py

```
"""
KINASE SUBSEQUENCE SELECTION (KSselect) a.k.a. KINASE
    ↪ PHYLO REMOTE
uses Subprocess instead of Biopython to get more control over
    ↪ subprocesses (Error logs, etc.)

"""


from subprocess import Popen, PIPE
import argparse
import re
from collections import OrderedDict

#specific modules:
from Bio import SeqIO
from Bio import AlignIO
from Bio import SeqRecord
from Bio import Seq
from Bio import Align
from Bio.Align import MultipleSeqAlignment
from Bio.Align.Applications import MuscleCommandline
```

```
from Bio import pairwise2
from Bio.Alphabet import IUPAC, Gapped
from Bio import Phylo
from Bio.Phylo.Applications import PhymlCommandline
from Bio.SeqIO.FastaIO import FastaWriter
import numpy
import pandas as pd
from funcy import func_partial


"""
Basic functions to convert to an from Biopython objects
"""

def biopy_MSA_to_dict_np_arrays (biopy_MSA):

    dict_id_seqrec_array = OrderedDict([])

    for seqrec in biopy_MSA:
        seqlist = []
        for aa in seqrec._seq._data:
            seqlist.append(aa)
        seqarray = numpy.array (seqlist)
        dict_id_seqrec_array[seqrec.id] = seqarray

    annot = biopy_MSA.annotations

    return (dict_id_seqrec_array, annot) #type tuple

def dict_np_arrays_to_biopy_MSA (dict_np_arrays, annotations):

    seqrec_list = []

    for gene_id in dict_np_arrays :
        sequence = ''.join(dict_np_arrays[gene_id])
        seqrec = SeqRecord.SeqRecord(Seq.Seq(sequence, Gapped(IUPAC
            ↪ .protein, "-")), id= gene_id)
        seqrec_list.append(seqrec)
         #am i swapping stuff here???? it seems!!!


    genes_domain_cut_MSA = MultipleSeqAlignment(seqrec_list, Gapped(
        ↪ IUPAC.protein, "-"), annotations)

    return genes_domain_cut_MSA

def remove_id_from_MSA (biopy_MSA, id_to_remove):
```

```
    seqrec_list = []

    for seqrec in biopy_MSA:
        if seqrec.id != id_to_remove:
            seqrec_list.append(seqrec)

    biopy_MSA_cut = MultipleSeqAlignment(seqrec_list, annotations =
        ↪ biopy_MSA.annotations)

    return biopy_MSA_cut

"""
Basic functions to convert indeces of gapped to and from indices of
    ↪ ungapped sequences
"""
def create_map_gapped_ungapped (biopy_seq_gapped):

    counter = 0

    map_gapped_ungapped = {}

    for c, x in enumerate(biopy_seq_gapped._seq._data):
        if x != '-':
            counter = counter + 1
            map_gapped_ungapped[c+1] = counter

    return map_gapped_ungapped


def create_map_ungapped_gapped (biopy_seq_gapped):

    sequence_np_array = numpy.array(list(biopy_seq_gapped._seq._data))

    index_not_gap = numpy.where (sequence_np_array != '-')

    mask_not_gap = sequence_np_array != '-'

    ungapped_sequence = sequence_np_array [mask_not_gap]

    map_ungapped_gapped = dict(zip(list(range(1,len(ungapped_sequence
        ↪ )+1)), index_not_gap[0]+1))

    return map_ungapped_gapped

"""
Advanced funtions to modify and analyse biopython msa objects
"""
```

```
def read_phytozome_csv_to_MSA_plus_dicts (infile_path, outfile_path,
    ↪ additional_annotation_dict):
    """
        Takes a .csv file that is given by the Phytozome website and
            ↪ writes it to a .fasta file
        Returns a note_atg_dict, which is DECREPATED!
    This function could be extended with bool keep_splice_variants, which
        ↪  would need a
        recursive function to name the splice variants with numbering
    """

    df = pd.read_csv(infile_path,
                     header = None,
                     names = ["Primary_Identifier",
                              "Defline",
                              "Organism_Name",
                              "Symbol",
                              "Length",
                              "Chrom._Identifier",
                              "Chrom._Loc._Start",
                              "Chrom._Loc._End",
                              "Sequence"],
                     dtype = str)

    def insert_additional_annotation(row, additional_annotation_dict):

        atg_list = []

        atg_name_dict = {}

        for k in additional_annotation_dict:
            atg_list.append(additional_annotation_dict[k])
            atg_name_dict[additional_annotation_dict[k]] = k

        atg_code = row.loc["Primary_Identifier"]

        if atg_code in atg_name_dict:
            row['Symbol'] = atg_name_dict[atg_code]

        return row

    insert_additional_annotation_p = func_partial(
        ↪ insert_additional_annotation, additional_annotation_dict =
        ↪ additional_annotation_dict)

    df = df.apply(insert_additional_annotation_p, axis = 1)

    handle = open(outfile_path, "w")
```

```
    writer = FastaWriter(handle)

    seqrec_list = []

    node_atg_dict = {}

    def _check_for_splice_var_pi (row):
        def _check_for_splice_var (seqrec):
            return seqrec.id == row.loc['Primary_Identifier']
        return _check_for_splice_var

    def _check_for_splice_var_sy (row):
        def _check_for_splice_var (seqrec):
            return seqrec.id == row.loc['Symbol']
        return _check_for_splice_var

    sy_mask = pd.notnull(df['Symbol'])

    sy_df = df[sy_mask]

    print (sy_df)


    pi_mask = pd.isnull(df['Symbol'])

    pi_df = df[pi_mask]

    print (pi_df)


    for index, row in sy_df.iterrows():
        sequence = row.loc['Sequence']
        check_for_splice_var_sy = _check_for_splice_var_sy (row)
        bool_list_sy =list( map (check_for_splice_var_sy, seqrec_list))
        if any(bool_list_sy):
            continue
        else:
            seqrec = SeqRecord.SeqRecord(Seq.Seq(sequence,
                                                 Gapped(IUPAC.protein,
                                                     ↪ "-")),
                                                 id = str(row.loc['
                                                     ↪ Symbol']))
        seqrec_list.append(seqrec)
        node_atg_dict[row.loc['Symbol']] = row.loc['Primary_Identifier'] #
            ↪ ?

    for index, row in pi_df.iterrows():
        sequence = row.loc['Sequence']
```

```
            check_for_splice_var_pi = _check_for_splice_var_pi (row)
            bool_list_pi =list( map (check_for_splice_var_pi, seqrec_list))
            print (bool_list_pi)
            if any(bool_list_pi):
                continue
            else:
                seqrec = SeqRecord.SeqRecord(Seq.Seq(sequence,
                                                     Gapped(IUPAC.protein,
                                                         ↪ "-")),
                                                     id = str(row.loc["
                                                         ↪ Primary␣
                                                         ↪ Identifier"]))
            seqrec_list.append(seqrec)
            node_atg_dict[row.loc["Primary_Identifier"]] = row.loc['Primary␣
                ↪ Identifier']

    print (seqrec_list)
    writer.write_file(seqrec_list)
    handle.close()

    return node_atg_dict

def cut_out_aligned_to_domain (biopy_alignment, domain_id):
    """
        Function takes an biopy MSA object and returns a biopy MSA
            ↪ object, where the regions aligning with the given
        domain_id are cut off
        """

    genes_domain_dict_np_arrays, MSA_annotation =
        ↪ biopy_MSA_to_dict_np_arrays (biopy_alignment)

    domain_aligned_array = genes_domain_dict_np_arrays [domain_id]

    domain_mask = domain_aligned_array != '-'

    genes_domain_id_cut_sequences_dict = {}

    for id in genes_domain_dict_np_arrays:

        np_array = genes_domain_dict_np_arrays [id]

        gene_domain_cut = np_array[domain_mask]

        genes_domain_id_cut_sequences_dict [id] = gene_domain_cut

    genes_domain_id_cut_sequences_MSA = dict_np_arrays_to_biopy_MSA
        ↪ (genes_domain_id_cut_sequences_dict, MSA_annotation)
```

```
    return genes_domain_id_cut_sequences_MSA

def get_closest_relative_to_sequence_from_alignment (biopy_seqrec,
    ↪ biopy_MSA):
    """
        Uses biopy pairwise alignment to find the closest related sequence
            ↪  to biopy_seqrec in a biopy MSA
        """

    domains_unaligned = []

    for sequence in biopy_MSA:
        seqq = biopy_seqrec._seq._data.replace('-','')
        idd = sequence.id
        domains_unaligned.append(SeqRecord.SeqRecord(Seq.Seq(seqq,
            ↪ Gapped(IUPAC.protein, "-")), id= idd))

    domain_score_dict = {}

    for sequence in domains_unaligned:
        domain_score = pairwise2.align.globalxx(biopy_seqrec.seq,
            ↪ sequence.seq, score_only = True)
        domain_score_dict [sequence.id] = domain_score

    domains_closest_id = max(domain_score_dict, key=lambda i:
        ↪ domain_score_dict[i])

    for sequence in biopy_MSA:
        if sequence.id == domains_closest_id:
            domains_closest_aligned = sequence._seq._data

    seqrec = SeqRecord.SeqRecord(Seq.Seq(domains_closest_aligned,
        ↪ Gapped(IUPAC.protein, "-")), id= domains_closest_id)

    return seqrec


def convert_indices_alignment_to_alignment(list_of_indices,
    ↪ biopy_MSA_from, biopy_MSA_to, id_in_common):
    """
        Uses a common sequence (id_in_common) to convert a list of
            ↪ indices for one MSA (from) to that of another (to)
        """

    for seqrec in biopy_MSA_from:
        if seqrec.id == id_in_common:
            seqrec_from = seqrec
```

```
    for seqrec in biopy_MSA_to:
        if seqrec.id == id_in_common:
            seqrec_to = seqrec

    map_gapped_ungapped = create_map_gapped_ungapped (seqrec_from)

    map_ungapped_gapped = create_map_ungapped_gapped (seqrec_to)

    converted_index_list = []

    corresponding_DoS_index_list = []

    for i in list_of_indices:

        try:
            ungapped_i = map_gapped_ungapped[i]
            corresponding_DoS_index_list.append(i)
        except:
            continue
        gapped_i = map_ungapped_gapped[ungapped_i]
        converted_index_list.append(gapped_i)

    return converted_index_list, corresponding_DoS_index_list

def take_columns_biopy_alignment (index_list, biopy_MSA):
    """
        Takes "columns" of given indices from biopy MSA to create new
            ↪ biopy MSA
        """


    dict_np_arrays, MSA_annotation = biopy_MSA_to_dict_np_arrays (
        ↪ biopy_MSA)

    index_list_minus1 = [x - 1 for x in index_list]

    dict_np_arrays_subset = {}

    for i in dict_np_arrays:
        array_subset = numpy.take(dict_np_arrays[i], index_list_minus1)
        dict_np_arrays_subset[i] = array_subset

    biopy_MSA_subset = dict_np_arrays_to_biopy_MSA (
        ↪ dict_np_arrays_subset, MSA_annotation)

    return biopy_MSA_subset
```

S2.2.2 lkbm-qGAP

Listing A.13: io.py

```
import sys

sys.path.append('C://Users/stxlkml/lkbm/projects')

from util.funcutil import lift_to_mb, args_to_tuple
from util.ioutil import get_mb_relative_path_list_and_ids
from funcy import rcompose, func_partial
from functional.pipeline import Sequence as fn_seq
import pandas as pd


def _read_relative_filepaths (relpaths, ids, time_interval):
    """Private function used in read_rootrace_output"""

    def __read_rootnav_csv_to_df (filepath):
        df = pd.read_csv(filepath,
                         skiprows = [0],
                         header = None)
        del df[0]

        column_name_list = []
        n_columns = list(range(1, (len(df.columns)+1)))
        for column_n in n_columns:
            column_name = "Root" + str(column_n)
            column_name_list.append(column_name)

        key_list = list(range(0, (len(df.index)*time_interval), time_interval
            ↪ ))

        df.columns = column_name_list
        df.loc[:,'Time'] = key_list
        df2 = df.set_index('Time').T
        df2.index.name = 'root'
        return df2

    files = fn_seq(relpaths)\
                    .map(__read_rootnav_csv_to_df)\
                    .to_list()

    df = pd.concat(files, keys = ids, names = ['plate-id'])

    return (df, ids)
```

```
def _add_line_index (df, ids, layout_csv):
    """Private function used in read_rootrace_output"""

    layout_df = pd.read_csv(layout_csv)
    layout_df = layout_df.set_index(['plate-id'])
    layout_df = layout_df[layout_df.index.isin(ids)] #need to get this id...
        ↪ how????


    def __add_index_line (df, layout_df):
        df = (df.reset_index()
                .join(layout_df,on=['plate-id'])
                .set_index(df.index.names)
                .set_index('line', append = True))
        return (df)

    return __add_index_line(df, layout_df)

def read_roottrace_output (rootnav_output_directory = None,
                           gr_identifier_str = None,
                           avv_identifier_str = None,
                           layout_csv = None,
                           time_interval = None):
        """Function mainly takes RootTrace output root directory,
            ↪ layout.csv filename and
        time interval of images taken and returns two maybe dataframes:
            ↪ growth rate and angle versus vertical"""


    read_relative_filepaths = func_partial(_read_relative_filepaths,
        ↪ time_interval = time_interval)
    add_line_index = func_partial(_add_line_index, layout_csv = layout_csv
        ↪ )

    read_relative_filepaths = args_to_tuple(read_relative_filepaths)
    add_line_index = args_to_tuple(add_line_index)

    read_relative_filepaths = lift_to_mb(read_relative_filepaths)
    add_line_index = lift_to_mb(add_line_index)

    read_files = rcompose(
                  get_mb_relative_path_list_and_ids,
                  read_relative_filepaths,
                  add_line_index)

    mb_gr_df = read_files(rootnav_output_directory, gr_identifier_str)
```

```
    mb_avv_df = read_files(rootnav_output_directory, avv_identifier_str)

    return mb_gr_df, mb_avv_df
```

Listing A.14: filter.py

```
"""
Add a docstring


"""
import sys

sys.path.append('C://Users/stxlkml/lkbm/projects')

import util.maybe as mb #requires maybe in util folder
from util.funcutil import lift_to_mb, args_to_tuple
from util.ioutil import get_mb_relative_path_list_and_ids
from funcy import rcompose, func_partial
from functional.pipeline import Sequence as fn_seq
import pandas as pd
import numpy
#these might not all be required

#Processing helper functions

@lift_to_mb
def _remove_mask (df, mask):
    return df[mask]

@lift_to_mb
def _nan_mask (df, mask):
    return df.where(mask, other = numpy.nan)

#Processing Step1: Choose time interval

def _cut_time (mb_df_gr, mb_df_avv, time_duration):
    """Add a docsting"""
    @lift_to_mb
    def __cut_time (df):
        df = df.loc[:, df.columns <= time_duration]
        return df
    return (__cut_time(mb_df_gr), __cut_time(mb_df_avv)) #this is not
        ↪ working!!!
```

```
#Processing Step2: Deal with RootTrace errors

def _filter_traceloss (mb_df_gr, mb_df_avv, gr_min_tracel_thresh):
    """
    Remove all traces that do not progress
    (indicative of overall traceloss)
    """
    @lift_to_mb
    def _generate_tracel_mask (gr_df):
        def _scan_root_for_traceloss (gr_series):
            gr_series = gr_series[gr_series.notnull()]
            diff_series = gr_series.diff().abs()[1:]
            mask = diff_series > gr_min_tracel_thresh
            if mask.any():
                return True
            else:
                return False
        tracel_mask = gr_df.apply (_scan_root_for_traceloss, axis = 1)
        return tracel_mask

    tracel_mask = _generate_tracel_mask(mb_df_gr)

    return (_remove_mask(mb_df_gr, tracel_mask), _remove_mask(
        ↪ mb_df_avv, tracel_mask))


def _filter_initial_traceloss (mb_df_gr, mb_df_avv, gr_max_tracel_thresh):
    """
    Remove all traces where there is a sudden increase in lenght
    (indicative of initial traceloss)
    """
    @lift_to_mb
    def _generate_init_tracel_mask (gr_df):
        def _scan_root_for_init_tracel (gr_series):
            gr_series = gr_series[gr_series.notnull()]

            diff_series = gr_series.diff()[1:]

            thresh_mask = numpy.logical_or(-gr_max_tracel_thresh >
                ↪ diff_series,
                                           diff_series >
                                               ↪ gr_max_tracel_thresh)


            thresh_series = diff_series[thresh_mask]

            if len(thresh_series) == 0:
```

```
                return True

            elif thresh_series.iloc[0] < -gr_max_tracel_thresh:

                return True

            elif thresh_series.iloc[0] > gr_max_tracel_thresh:

                return False

        init_tracel_mask = gr_df.apply (_scan_root_for_init_tracel, axis =
            ↪ 1)

        return init_tracel_mask

    init_tracel_mask = _generate_init_tracel_mask(mb_df_gr)

    return (_remove_mask(mb_df_gr, init_tracel_mask), _remove_mask(
        ↪ mb_df_avv, init_tracel_mask))


def _filter_intermittent_traceloss (mb_df_gr, mb_df_avv,
    ↪ gr_max_tracel_thresh):
    """
    Convert all values to missing between sudden decreases and increases
        ↪ in lenght
    (indicative of intermittent traceloss)
    """
    def _generate_interm_tracel_mask (gr_df):

        mask_masks = []

        for index, gr_series in gr_df.iterrows():

            diff_series = gr_series.diff()[1:]

            mask = [True]

            status = 'on'

            for i in list(diff_series.values):

                if i < -gr_max_tracel_thresh:
                    status = 'off'
                elif i > gr_max_tracel_thresh:
                    status = 'on'

                if status == 'on':
```

```
                    mask.append(True)

                elif status == 'off':
                    mask.append(False)

            mask_masks.append(mask)

        return(mask_masks)

    interm_tracel_mask = _generate_interm_tracel_mask(mb_df_gr.value)

    return (_nan_mask(mb_df_gr, interm_tracel_mask), _nan_mask(
        ↪ mb_df_avv, interm_tracel_mask))


def _filter_missing_avv_value (mb_df_gr, mb_df_avv):
    """
    Filter out measurement errors that result in 999 being recorded
    """
    if mb_df_avv != mb.Nothing:
        @lift_to_mb
        def _generate_missing_avv_value_mask (df_avv):
            missing_avv_value_mask = df_avv != 999
            return missing_avv_value_mask

        missing_avv_value_mask = _generate_missing_avv_value_mask (
            ↪ mb_df_avv)

        return (_nan_mask(mb_df_gr, missing_avv_value_mask), _nan_mask
            ↪ (mb_df_avv, missing_avv_value_mask))

    else:
        return (mb_df_gr, mb_df_avv)

def _filter_wrong_avv_value (mb_df_gr, mb_df_avv):
    """
        Filter out measurement errors that result from RootTrace using
            ↪ the wrong baseline
        """
    if mb_df_avv != mb.Nothing:
        df = mb_df_avv.value
        df = df.where(df != 179, other = -1)
        df = df.where(df != 178, other = -2)
        df = df.where(df != -179, other = 1)
        df = df.where(df != -178, other = 2)
        return (mb_df_gr, mb.Just(df))
    else:
        return (mb_df_gr, mb_df_avv)
```

```
def _filter_nan_thresh (mb_df_gr, mb_df_avv, nan_thresh):
    """
        Filter out roots that have a number of NaN values above the
            ↪ given trashhold
        """
    @lift_to_mb
    def __dropna_thresh (df):
        number_of_datapoints = df.shape[1] - nan_thresh
        df = df.dropna(thresh = number_of_datapoints)
        return df
    return (__dropna_thresh(mb_df_gr), __dropna_thresh(mb_df_avv))

#Processing Step3: Deal with seedling variation

def _filter_overall_gr_quantile (mb_df_gr, mb_df_avv, upper_quantile,
    ↪ lower_quantile):
    """
        Filters out roots whose growth is above or below a given quantile
            ↪ at any timepoint
        """


    @lift_to_mb
    def _generate_overall_gr_quantile_mask (df_gr):
        percentile_df = df_gr.quantile([upper_quantile, lower_quantile])
        percentile_mask = df_gr.apply(lambda x: numpy.logical_and ((x >
            ↪  percentile_df.loc[lower_quantile,x.name]),
                                                                   (x <
                                                                       ↪ percentile_df
                                                                       ↪ .
                                                                       ↪ loc
                                                                       ↪ [
                                                                       ↪ upper_quant
                                                                       ↪ ,
                                                                       ↪ x
                                                                       ↪ .
                                                                       ↪ name
                                                                       ↪ ])
                                                                       ↪ )
                                                                       ↪ )
                                                                       ↪
        percentile_mask_collated = percentile_mask.apply(lambda x:
            ↪ numpy.logical_and.reduce(x), axis = 1)
        return percentile_mask_collated

    overall_gr_quantile_mask = _generate_overall_gr_quantile_mask (
        ↪ mb_df_gr)
```

```
    return (_remove_mask(mb_df_gr, overall_gr_quantile_mask),
        ↪ _remove_mask(mb_df_avv, overall_gr_quantile_mask))


def _filter_wildtype_gr_quantile (mb_df_gr, mb_df_avv, wt_upper_quantile,
    ↪ wt_lower_quantile, wt_id = None):
    """
    DECREPATED! This filter was too stringent.
    """
    @lift_to_mb
    def _generate_wildtype_gr_quantile_mask (df_gr):
        wt_df = df_gr.groupby('line').get_group(wt_id)
        percentile_df = wt_df.quantile([wt_upper_quantile,
            ↪ wt_lower_quantile])
        percentile_mask = df_gr.apply(lambda x: numpy.logical_and ((x >
            ↪  percentile_df.loc[wt_lower_quantile,x.name]),
                                                                   (x <
                                                                       ↪ percentile_df
                                                                       ↪ .
                                                                       ↪ loc
                                                                       ↪ [
                                                                       ↪ wt_upper_qu
                                                                       ↪ ,
                                                                       ↪ x
                                                                       ↪ .
                                                                       ↪ name
                                                                       ↪ ])
                                                                       ↪ )
                                                                       ↪ )
                                                                       ↪
        percentile_mask_collated = percentile_mask.apply(lambda x:
            ↪ numpy.logical_and.reduce(x), axis = 1)
        return percentile_mask_collated

    wildtype_gr_quantile_mask = _generate_wildtype_gr_quantile_mask (
        ↪ mb_df_gr)

    return (_remove_mask(mb_df_gr, wildtype_gr_quantile_mask),
        ↪ _remove_mask(mb_df_avv, wildtype_gr_quantile_mask))


def _filter_initial_angle_thresh (mb_df_gr, mb_df_avv, initial_angle_tuple):
    """
        Filter roots based on their initial angle (CPIB standart protocol
            ↪ is −10 to 10, which corresponds to 80 to 100.
    """
    if mb_df_avv != mb.Nothing:
        initial_angle_mask = numpy.logical_and(initial_angle_tuple[0] <
```

```
            ↪ mb_df_avv.value[0],
                                                mb_df_avv.value[0] <
                                                    ↪ initial_angle_tuple
                                                    ↪ [1])
        return (_remove_mask(mb_df_gr, initial_angle_mask),
            ↪ _remove_mask(mb_df_avv, initial_angle_mask))
    else:
        return (mb_df_gr, mb_df_avv)
#Processing Step4: Prepare data for visualisation

def _adjust_growthrate (mb_df_gr, mb_df_avv):
    """
        Substracts growth datapoints by initial value, giving the distance
            ↪ grown after the first timepoint
        """
    df = mb_df_gr.value
    GR_starting_positions = df.iloc[:,0]
    df = df.subtract(GR_starting_positions, axis = 0, level = 'id')
    return (mb.Just(df), mb_df_avv)


def _adjust_angle (mb_df_gr, mb_df_avv):
    """
        Adjusts angle datapoints to reflect experimental setup
        """
    @lift_to_mb
    def __adjust_angle(df):
        return df + 90
    return (mb_df_gr, __adjust_angle(mb_df_avv))


def filter_dfs (mb_gr_df = None,
                    mb_avv_df = None,
                    time_duration = 0,
                    gr_min_tracel_thresh = 0,
                    gr_max_tracel_thresh = 0,
                    remove_nan_thresh = 0,
                    remove_overall_quantile = True,
                        upper_quantile = .95,
                        lower_quantile = .05,
                    remove_wildtype_quantile = True,
                        wt_upper_quantile = .95,
                        wt_lower_quantile = .05,
                        wt_id = 'Col-0',
                    initial_angle_tuple = (0,0),
                    zero_growthrate = True,
                    adjust_angle = True):
```

```
    """
    Applies filter function to the both given maybe dataframes. Uses
        ↪ functional composition to compose private functions
    """

    filter_function_list = []

    #Processing Step1: Choose time interval

    if time_duration == 0:
        pass
    else:
        cut_time = func_partial(_cut_time,
                                            time_duration = time_duration)
        cut_time = args_to_tuple(cut_time)
        filter_function_list.append(cut_time)

    #Processing Step2: Deal with RootTrace errors

    if gr_min_tracel_thresh == 0:
        pass
    else:
        filter_traceloss = func_partial(_filter_traceloss,
                                            gr_min_tracel_thresh =
                                                ↪ gr_min_tracel_thresh)
        filter_traceloss = args_to_tuple(filter_traceloss)
        filter_function_list.append(filter_traceloss)



    if gr_max_tracel_thresh == 0:
        pass
    else:
        filter_initial_traceloss = func_partial(_filter_initial_traceloss,
                                                    gr_max_tracel_thresh =
                                                        ↪ gr_max_tracel_thresh
                                                        ↪ )
        filter_initial_traceloss = args_to_tuple(filter_initial_traceloss)
        filter_function_list.append(filter_initial_traceloss)



        filter_intermittent_traceloss = func_partial(
            ↪ _filter_intermittent_traceloss,
                                                    gr_max_tracel_thresh
                                                        ↪ =
                                                        ↪ gr_max_tracel_thresh
                                                        ↪ )
```

```
        filter_intermittent_traceloss = args_to_tuple(
            ↪ filter_intermittent_traceloss)
        filter_function_list.append(filter_intermittent_traceloss)



    filter_missing_avv_value = args_to_tuple(_filter_missing_avv_value)
    filter_function_list.append(filter_missing_avv_value)

    filter_wrong_avv_value = args_to_tuple(_filter_wrong_avv_value)
    filter_function_list.append(filter_wrong_avv_value)


    if remove_nan_thresh == 0:
        pass
    else:
        filter_nan_thresh = func_partial(_filter_nan_thresh,
                                         nan_thresh =
                                             ↪ remove_nan_thresh)
        filter_nan_thresh = args_to_tuple(filter_nan_thresh)
        filter_function_list.append(filter_nan_thresh)


    if remove_overall_quantile == False:
        pass
    else:
        filter_overall_gr_quantile = func_partial(_filter_overall_gr_quantile,
                                                  upper_quantile =
                                                      ↪ upper_quantile,
                                                  lower_quantile =
                                                      ↪ lower_quantile)
        filter_overall_gr_quantile = args_to_tuple(filter_overall_gr_quantile)
        filter_function_list.append(filter_overall_gr_quantile)


    if remove_wildtype_quantile == False:
        pass
    else:
        filter_wildtype_gr_quantile = func_partial(
            ↪ _filter_wildtype_gr_quantile,
                                                  wt_upper_quantile =
                                                      ↪ wt_upper_quantile
                                                      ↪ ,
                                                  wt_lower_quantile =
                                                      ↪ wt_lower_quantile
                                                      ↪ ,
                                                  wt_id = wt_id)
        filter_wildtype_gr_quantile = args_to_tuple(
```

```
            ↪ filter_wildtype_gr_quantile)
        filter_function_list.append(filter_wildtype_gr_quantile)


    if initial_angle_tuple == (0,0):
        pass
    else:
        filter_initial_angle_thresh = func_partial(_filter_initial_angle_thresh,
                                                   initial_angle_tuple =
                                                       ↪ initial_angle_tuple
                                                       ↪ )
        filter_initial_angle_thresh = args_to_tuple(filter_initial_angle_thresh
            ↪ )
        filter_function_list.append(filter_initial_angle_thresh)

    #Processing Step4: Prepare data for visualisation

    if zero_growthrate == False:
        pass
    else:
        adjust_growthrate = args_to_tuple(_adjust_growthrate)
        filter_function_list.append (adjust_growthrate)
    if adjust_angle == False:
        pass
    else:
        adjust_angle = args_to_tuple(_adjust_angle)
        filter_function_list.append (adjust_angle)



    _filter_dataframes = rcompose(*filter_function_list)

    filtered_dfs_tuple = _filter_dataframes((mb_gr_df, mb_avv_df))

    return filtered_dfs_tuple
```

Listing A.15: formulaformat.py

```
import pandas as pd

def format_df_for_formula (df):
    """
        Formats dataframe to long format to be used in statistical
            ↪ packages (e.g. lmerTEST)
        """
    df_T = (df.reset_index()
```

```
                .set_index(['line'])
                .T[5::])

    df_T.index.name = 'Time'
    df_T = (df_T.reset_index()
                .astype(float))

    df_melt = pd.melt(df_T, id_vars=["Time"],
                      var_name="Line", value_name="Angle")

    return df_melt
```

S2.2.3 lkbm-util

Listing A.16: funcutil.py

```
"""
Utility functions for funcy.rcompose and util.maybe
"""
import sys

sys.path.append('C://Users/stxlkml/lkbm/projects')

import util.maybe as mb #requires maybe in util folder

def args_to_tuple (fn):
    def wrapper (args):
        return fn(*args)
    return wrapper

def lift_to_mb (fn):
    return mb.lift(fn)
```

Listing A.17: ioutil.py

```
import sys

sys.path.append('C://Users/stxlkml/lkbm/projects')

import util.maybe as mb #requires maybe in util folder
from functional.pipeline import Sequence as fn_seq
from funcy import rcompose
import itertools as it
```

```
import os

def get_mb_relative_path_list_and_ids (root_directory_str, file_identifier_str):
    """
        Takes a root diretory and a identifier for the files in it and returns
            ↪ maybe tuple:
        list of paths for each file and corresponsing ids (file path fused
            ↪ with "-"

    [str], [str] => Just[tuple]
    """
    def relpath_and_join (x, y):
        return os.path.relpath(os.path.join(x, y))

    l = fn_seq(it.starmap(lambda *x: x, os.walk(root_directory_str)))\
                .filter(lambda x: x[1] == [])\
                .to_list()

    dirpaths = fn_seq(l).map(lambda x : x[0])\
                            .to_list()

    filenames = fn_seq(l).map(lambda x: x[2])\
                                .map(lambda x: list(filter(lambda s :
                                    ↪ file_identifier_str in s, x)))\
                                .to_list()

    if fn_seq(filenames).for_all(lambda x: x == []):
        return mb.Nothing

    else:
        where_no_rec = fn_seq(filenames).map(lambda x: x != [])
        filenames = fn_seq(it.compress(filenames, where_no_rec)).flatten().
            ↪ to_list()
        dirpaths = list(it.compress(dirpaths, where_no_rec))
        relpaths = list(map(relpath_and_join, dirpaths, filenames))
        ids = list(map(lambda x: '-'.join(x.split('\\')[1:]), dirpaths))
        return mb.Just((relpaths, ids)) #this makes sense from lift sc
            ↪ lm220917
```

Listing A.18: maybe.py

```
# maybe.py - a Pythonic implementation of the Maybe monad
# Copyright (C) 2014. Senko Rasic <senko.rasic@goodcode.io>


__all__ = ['Maybe', 'Just', 'Nothing', 'lift']
```

```
def lift(fn):
    """lift a function to Maybe monad
    Converts a plain function to one that can accept Maybe arguments
    (positional or keyword) and returns a Maybe return value.
    The returning function returns Nothing if any of the arguments is
        ↪ Nothing.
    Otherwise, it returns Just(<return value of the original function>).
    The converted function accepts both Maybe (which are unpacked)
        ↪ and
    non-Maybe arguments (which are used as-is).
    Examples:
    >>> maybe_abs = lift(abs)
    >>> maybe_abs(Just(-1))
    Just(1)
    >>> maybe_abs(Nothing)
    Nothing
    >>> maybe_abs(0)
    Just(0)
    """

    def wrapper(*args, **kwargs):
        def lift_arg(arg):
            return arg.value if isinstance(arg, Maybe) else arg

        largs = []
        for arg in args:
            if arg is Nothing:
                return Nothing
            else:
                largs.append(lift_arg(arg))
        lkwargs = {}
        for k, v in kwargs.items():
            if v is Nothing:
                return Nothing
            else:
                lkwargs.append(lift_arg(arg))
        return Maybe(fn(*largs, **lkwargs))
    wrapper.__name__ = fn.__name__
    wrapper.__doc__ = fn.__doc__
    return wrapper


class Maybe(object):
    """Maybe monad
    Wraps a value (or nonexistence of the value) inside an opaque
        ↪ container.
    The container can be used in place of the actual value in all
```

```
    ↪ operations
(including mathematical and logical operations, attribute access and
function calls).
The monad is either ‘Nothing‘ (a singleton object like None, signalling
the nonexistence of any value) or ‘Just(value)‘ (a wrapper around the
actual value).
All operations on the monad are short−circuited to return ‘Nothing‘
    ↪ if
the monad is ‘Nothing‘, and to perform the operation on the actual
    ↪ value
if the monad is ‘Just(value)‘. This allows for easy chaining of
    ↪ operations
on a value without checking for result existence or validity after each
step.
Nothing is a singleton and different from any Just value:
>>> Nothing is (Nothing + Nothing)
True
>>> Nothing is Just(None)
False
Math/logical operators work as expected:
>>> assert (Nothing == 1) is Nothing
>>> assert Nothing + Just(1) is Nothing
>>> assert Nothing − 1 is Nothing
>>> assert 2 ∗ Nothing is Nothing
>>> assert Just(2) / Nothing is Nothing
>>> assert Just(10) % 3 == Just(1)
>>> assert Just(5) > 3
>>> assert Just(5) == 5
>>> assert Just(5) < 6
>>> assert Just(5) != 6
Note that Nothing in boolean context is always False (for easier
    ↪ checking
without explicit cast to ‘bool‘), while Just is False or True depending
on its value:
>>> bool(Nothing)
False
>>> bool(1)
True
>>> bool(0)
False
Accessing attributes on Maybe either short−cuts to Nothing or
    ↪ accesses
the corresponding attributes in the contained value:
>>> assert Nothing.foo is Nothing
>>> assert Just('hello').upper() >> str == 'HELLO'
Maybe is a callable and will either short−circuit for Nothing, or call
the contained value:
>>> assert Nothing() is Nothing
```

```
>>> assert Just(lambda: 42)() >> int == 42
The above two facts, combined, mean that we can do something like:
>>> Just("hello world").upper()
Just('HELLO WORLD')
Right-shift operator (>>) can be used in the same way as monadic
    ↪ bind
in Haskell (>>=):
>>> Just(5) >> (lambda x: Just(x * 2))
Just(10)
In Python, this is useful for getting the value back from the Monad,
    ↪ and
at the same time verifying that it can be cast to a specific type:
>>> Just(5) >> int
5
>>> Just('hello') >> int
Traceback (most recent call last):
...
ValueError: invalid literal for int() with base 10: 'hello'
Often it is useful to get either the actual value (if any) or a default
value in case of Nothing. Although this can be done with >>, there's
    ↪ a
nicer and shorter syntax for it using the bitwise or operator (|):
>>> Just(5) | 0
5
>>> Nothing | 0
0
When cast to string or unicode, Nothing will evaluate to an empty
    ↪ string,
and Just will just call the to-string conversion methods of the
    ↪ contained
value.
>>> str(Nothing)
''
>>> unicode(Nothing)
u''
And finally, here's proof 'Maybe' really is a monad (required axioms
    ↪ hold):
>>> assert (Just(42) >> int) == 42 # (return x) >>= f <=> f x
>>> assert (Just('hello') >> str) == 'hello' # (return x) >>= f
    ↪ <=> f x
>>> assert (Just(42) >> Just) == Just(42) # m >>= return <=>
    ↪  m
>>> assert (Just('hello') >> Just) == Just('hello') # m >>=
    ↪ return <=> m
>>> # (m >>= f) >>= g <=> m >>= (lambda x -> (f x >>= g
    ↪ ))
>>> assert ((Just(-1) >> (lambda x: Just(abs(x))) >> str) ==
... Just(-1) >> (lambda x: str(abs(x))) == '1')
```

```
    """
    class __metaclass__(type):

        ops = ['add', 'sub', 'mul', 'div', 'mod', 'eq', 'ne', 'lt', 'gt']

        def __new__(mcs, name, bases, dct):
            import operator

            def reverse(fn):
                wrapper = lambda x, y: fn(y, x)
                wrapper.__name__ = fn.__name__
                wrapper.__doc__ = fn.__doc__
                return wrapper

            for opname in mcs.ops:
                op = getattr(operator, opname)
                dct['__' + opname + '__'] = lift(op)
                dct['__r' + opname + '__'] = lift(reverse(op))

            return type.__new__(mcs, name, bases, dct)


    def __init__(self, value):
        self.value = value

    def __repr__(self):
        return 'Nothing' if self is Nothing else 'Just(%s)' % repr(self.value
            ↪ )

    def __str__(self):
        return '' if self is Nothing else unicode(self.value)

    def __unicode__(self):
        return u'' if self is Nothing else unicode(self.value)

    def __nonzero__(self):
        return False if self is Nothing else bool(self.value)

    def __call__(self, *args, **kwargs):
        return Nothing if self is Nothing \
            else Maybe(self.value(*args, **kwargs))

    def __getattr__(self, name):
        return Nothing if self is Nothing \
            else Maybe(getattr(self.value, name))

    def __getitem__(self, key_or_slice):
        return Nothing if self is Nothing \
```

```
            else Maybe(self.value.__getitem__(key_or_slice))

    def __rshift__(self, fn):
        if self is Nothing:
            return Nothing
        else:
            return fn(self.value)

    def __or__(self, other):
        if self is Nothing:
            return other
        else:
            return self.value

Nothing = Maybe(None) # actual value here doesn't matter at all
Just = Maybe


if __name__ == '__main__':
    import doctest
    doctest.testmod()
```