The Rice example: illustrating the first five steps for smoothing and extracting traits (SET) using growthPheno

Chris Brien

01 May, 2025

This example is based on the data whose analysis has been published by Al-Tamimi et al. (2016). The five steps of the method for smoothing and extracting traits (SET) described in detail in Brien et al. (2020) is illustrated for this data using `growthPhenO` (Brien, 2025g), a package for the R statistical computing environment (R Core Team, 2025).

Initialize

Step 1: Import, select and derive longitudinal data

Step 1(a): Import the data

```
data(RiceRaw.dat)
```

Step 1(b): Organize the data

Here the imaging variables are selected and covariates and factors added to produce `longi.dat`.

```
longi.dat <- prepImageData(data=RiceRaw.dat,
                           potIDcolumns = c("Genotype.ID", "Treatment.1", "Replicate" ),
                           smarthouse.lev=c("NE","NW"))

longi.dat <- designFactors(data = longi.dat, insertName = "Replicate",
                           nzones = 3, designfactorMethod="StandardOrder")

#'## Particular edits to longi.dat - add Days after treatment (xDAT)
longi.dat$xDAT <- longi.dat$xDAP - 29
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag,DAP), ])
```

Step 1(c): Derive longitudinal traits that result in a value for each observation

```r
# Set responses
responses.image <- c("PSA")
responses.smooth <- paste0("s", responses.image)

# Form growth rates for each observation of a subset of responses by differencing
longi.dat <- byIndv4Times_GRsDiff(longi.dat, responses = responses.image,
                                  times = "DAP",
                                  which.rates = c("AGR","RGR"))

# Form PSA.WUI
longi.dat <- within(longi.dat,
                    PSA.WUI <- WUI(PSA.AGR*DAP.diffs, WU))

# Add cumulative responses
longi.dat <- within(longi.dat,
                    {
                      WU.cum <- unlist(by(WU, Snapshot.ID.Tag,
                                          cumulate, exclude.1st=TRUE))
                      WUI.cum <- PSA / WU.cum
                    })
# Check longi.dat
head(longi.dat)
```

```
##   Smarthouse Lane Position DAP xDAP Snapshot.ID.Tag Snapshot.Time.Stamp
## 1         NE    1        2  28   28        045451-C 2015-02-18 02:14:00
## 2         NE    1        2  30   30        045451-C 2015-02-20 02:14:00
## 3         NE    1        2  31   31        045451-C 2015-02-21 02:14:00
## 4         NE    1        2  32   32        045451-C 2015-02-22 02:14:00
## 5         NE    1        2  33   33        045451-C 2015-02-23 02:14:00
## 6         NE    1        2  34   34        045451-C 2015-02-24 02:14:00
##       Hour Genotype.ID Treatment.1 Replicate Zone cZone SHZone ZLane ZMainunit
## 1 2.233333      121080     Control         1    1    -1      1     1         1
## 2 2.233333      121080     Control         1    1    -1      1     1         1
## 3 2.233333      121080     Control         1    1    -1      1     1         1
## 4 2.233333      121080     Control         1    1    -1      1     1         1
## 5 2.233333      121080     Control         1    1    -1      1     1         1
## 6 2.233333      121080     Control         1    1    -1      1     1         1
##   Subunit cMainPosn cPosn Weight.Before Weight.After Water.Amount  WU     PSA
## 1       1     -10.5   -11          4007         4031           28  NA  57.446
## 2       1     -10.5   -11          4056         4084           32 -25  89.306
## 3       1     -10.5   -11          4036         4083           52  48 100.138
## 4       1     -10.5   -11          4027         4085           61  56 128.323
## 5       1     -10.5   -11          4019         4084           69  66 158.776
## 6       1     -10.5   -11          4014         4083           74  70 182.551
##   PSA.SV1 PSA.SV2 PSA.TV Boundary.Points.To.PSA.Ratio.SV1
## 1  20.912  11.526 25.008                         0.353912
## 2  29.073  21.495 38.738                         0.310735
## 3  27.751  26.835 45.552                         0.354293
## 4  34.697  32.848 60.778                         0.371012
## 5  46.779  37.871 74.126                         0.319823
## 6  48.849  48.794 84.908                         0.328400
##   Boundary.Points.To.PSA.Ratio.SV2 Boundary.Points.To.PSA.Ratio.TV
## 1                         0.454104                        0.197537
```

```
## 2                         0.401396                        0.172182
## 3                         0.332364                        0.174175
## 4                         0.358469                        0.178157
## 5                         0.347179                        0.172517
## 6                         0.290220                        0.163153
##   Caliper.Length.SV1 Caliper.Length.SV2 Caliper.Length.TV Compactness.SV1
## 1            666.013            668.692           704.189       0.0930821
## 2            632.735            729.044           830.812       0.1327200
## 3            731.077            931.028          1104.350       0.0925419
## 4            791.760            878.427          1029.300       0.0969068
## 5            830.360            965.221          1197.530       0.1241550
## 6           1103.050            991.259          1408.310       0.0938637
##   Compactness.SV2 Compactness.TV Convex.Hull.PSA.SV1 Convex.Hull.PSA.SV2
## 1       0.0689923      0.1435880             224.662             167.062
## 2       0.0734412      0.1091450             219.055             292.683
## 3       0.0678337      0.0950009             299.875             395.600
## 4       0.0707469      0.1102850             358.045             464.303
## 5       0.0783589      0.1119250             376.780             483.302
## 6       0.1014870      0.0947390             520.425             480.792
##   Convex.Hull.PSA.TV Center.Of.Mass.Y.SV1 Center.Of.Mass.Y.SV2
## 1            174.165              1841.78              1788.86
## 2            354.921              1837.62              1797.42
## 3            479.490              1826.88              1757.60
## 4            551.097              1798.03              1750.54
## 5            662.283              1796.70              1781.50
## 6            896.231              1809.42              1778.94
##   Max.Dist.Above.Horizon.Line.SV1 Max.Dist.Above.Horizon.Line.SV2 xDAT
## 1                             620                             624   -1
## 2                             543                             541    1
## 3                             642                             633    2
## 4                             736                             823    3
## 5                             658                             652    4
## 6                             639                             633    5
##   DAP.diffs PSA.AGR   PSA.RGR    PSA.WUI    WUI.cum WU.cum
## 1        NA      NA        NA         NA         NA     NA
## 2         2  15.930 0.2206116 -1.2744000 -3.5722400    -25
## 3         1  10.832 0.1144806  0.2256667  4.3538261     23
## 4         1  28.185 0.2480013  0.5033036  1.6243418     79
## 5         1  30.453 0.2129439  0.4614091  1.0950069    145
## 6         1  23.775 0.1395352  0.3396429  0.8490744    215
```

Step 2: Exploratory analysis

Step 2(a): Fit splines to smooth the longitudinal trends in the primary traits and calculate their growth rates

The `smoothing.method` used is `direct` and `df` is set to 4. The growth rates are calculated by difference, rather than from the spline derivatives.

```
# Smooth responses and form growth rates by differences
for (response in c(responses.image, "WU"))
  longi.dat <- byIndv4Times_SplinesGRs(data = longi.dat, response = response,
```

```
                                        response.smoothed = paste0("s", response),
                                        individuals = "Snapshot.ID.Tag", times="DAP",
                                        df = 4)
```

```
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
## Warning in FUN(X[[i]], ...): Need at least 4 distinct x values to fit a spline
## - all fitted values set to NA
```

```
## Warning in log(PGR(x, time.diffs, lag = lag)): NaNs produced
```

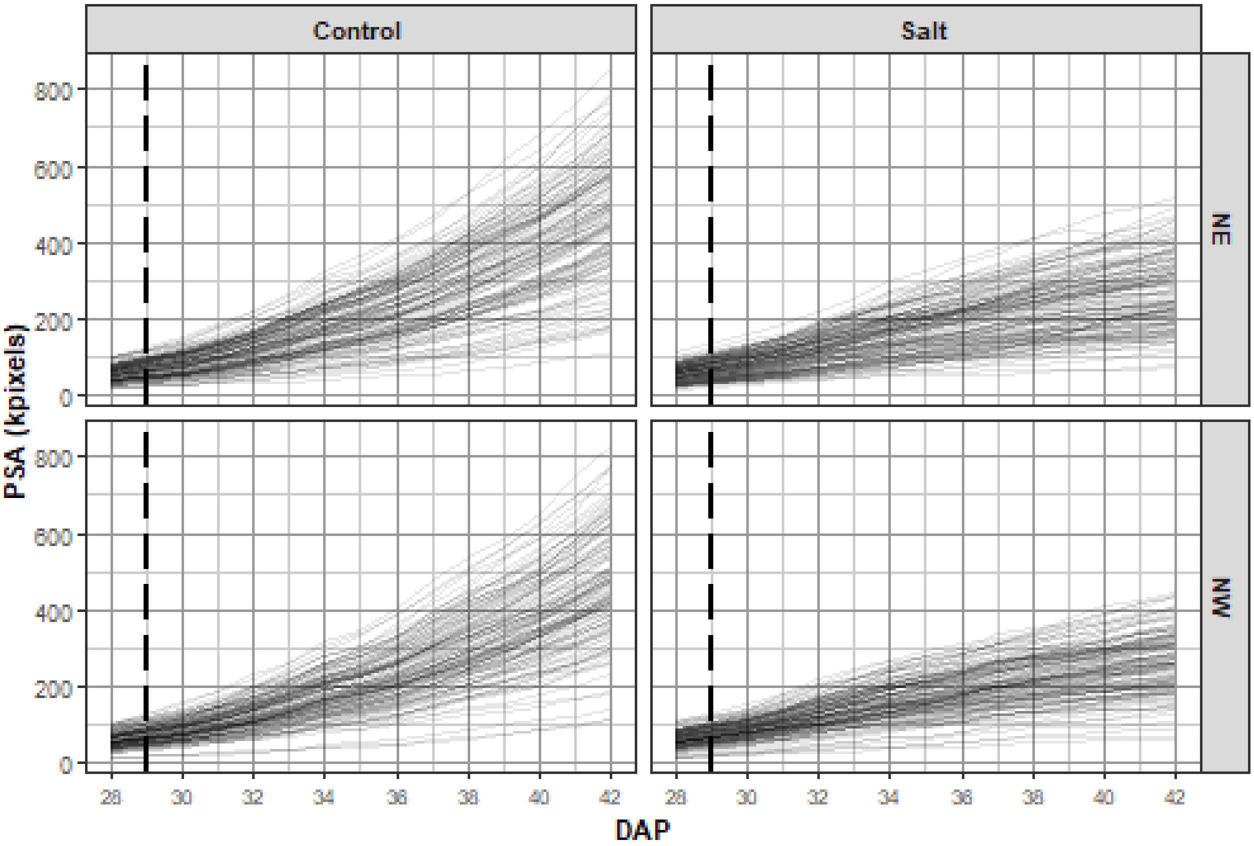
```
# Finalize longi.dat
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag, xDAP), ])
```

Step 2(b): Compare plots of unsmoothed and smoothed longitudinal data

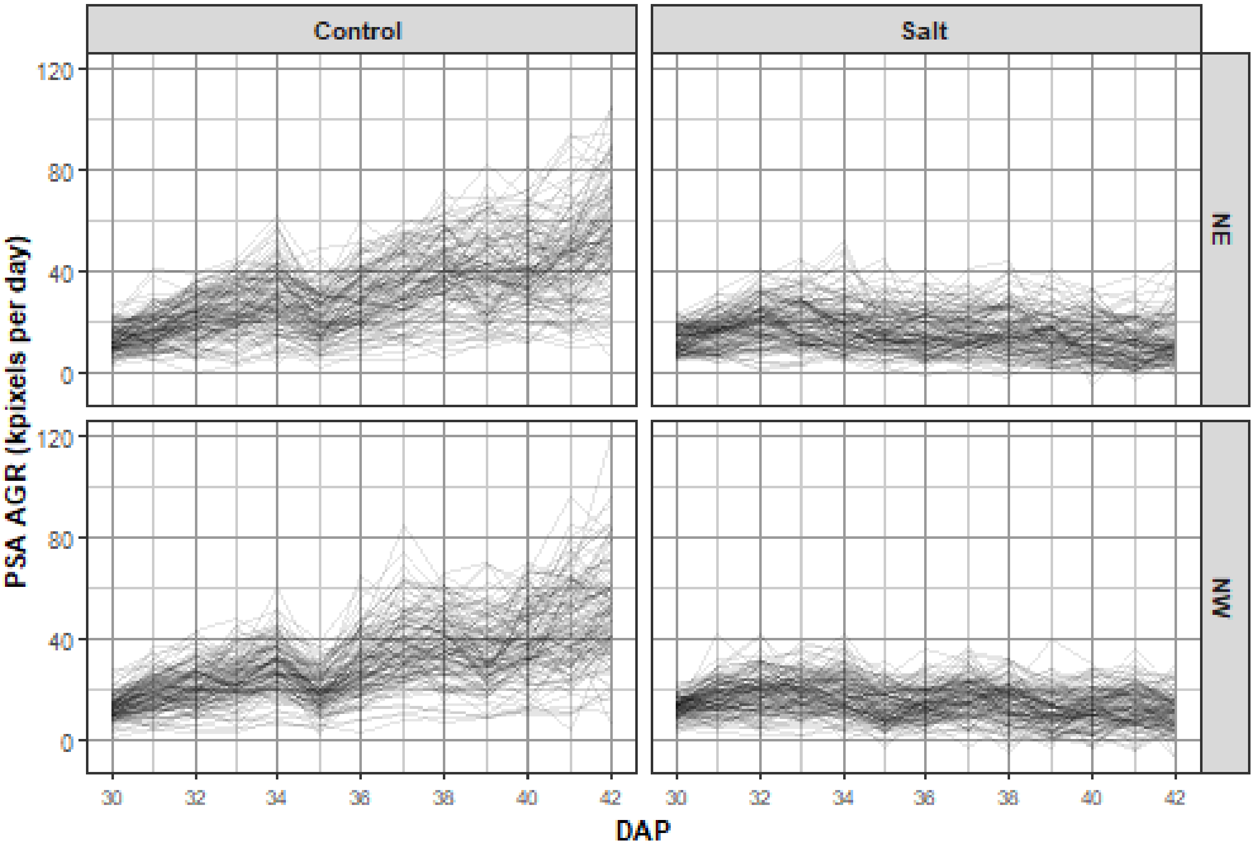
```
responses.longi <- c("PSA","PSA.AGR","PSA.RGR", "PSA.WUI")
responses.smooth.plot <- c("sPSA","sPSA.AGR","sPSA.RGR")
titles <- c("PSA (kpixels)",
            "PSA AGR (kpixels per day)", "PSA RGR (per day)",
            "PSA WUI (kpixels per mL)")
titles.smooth<-paste0("s", titles)
nresp <- length(responses.longi)
```

Plot unsmoothed profiles for all longitudinal responses

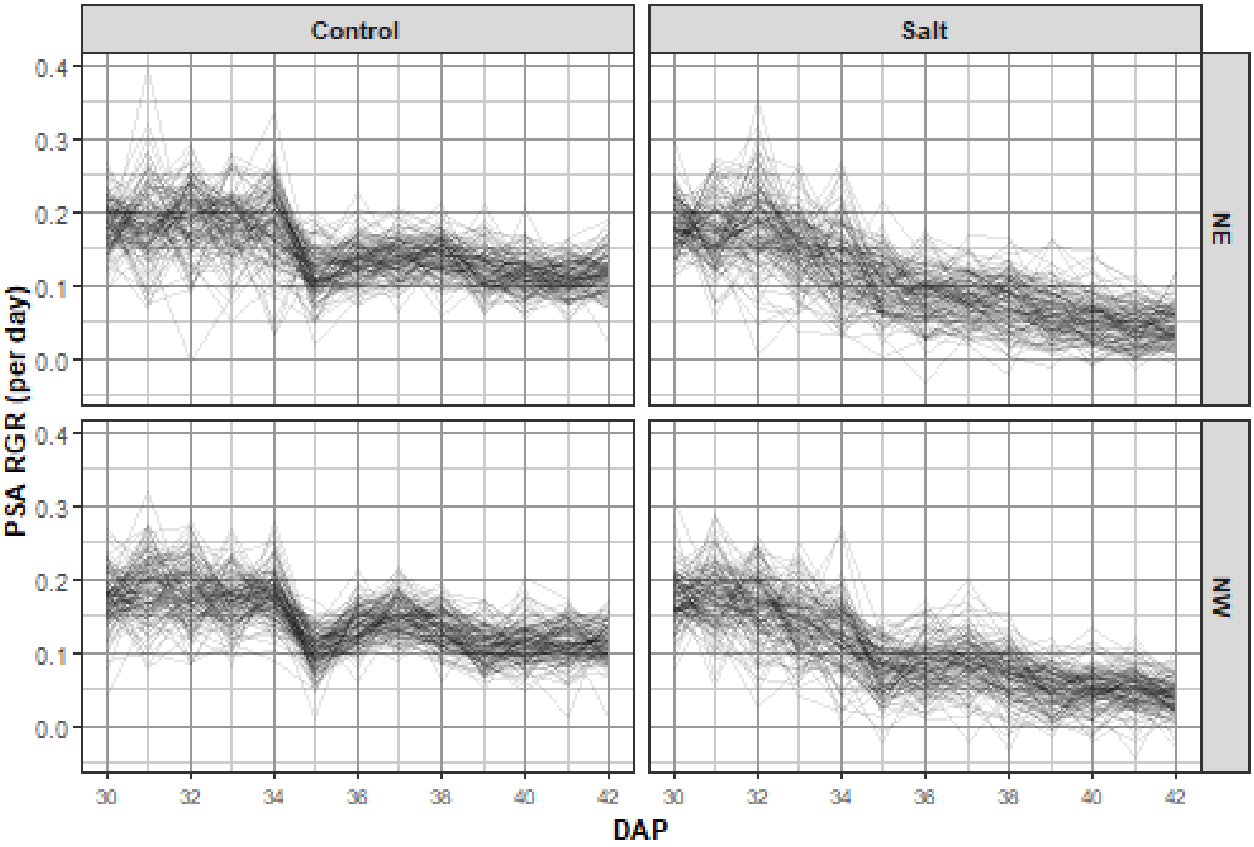
```
for (k in 1:nresp)
{
  plt <- plotProfiles(data = longi.dat, response = responses.longi[k],
                      y.title = titles[k], times = "DAP",
                      facet.x = "Treatment.1", facet.y = "Smarthouse",
                      breaks.spacing.x = 2,
                      printPlot=FALSE)
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", linewidth=1)
  print(plt)
}
```



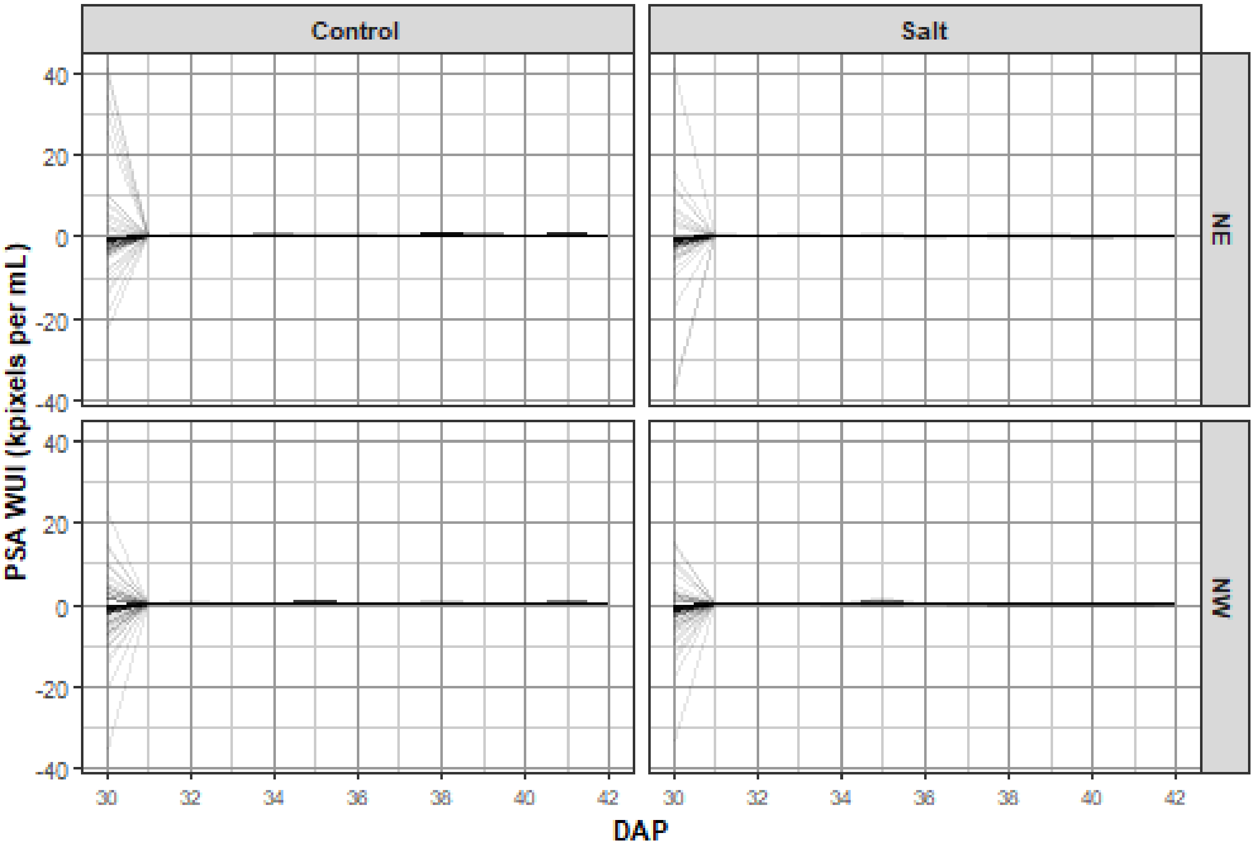
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```

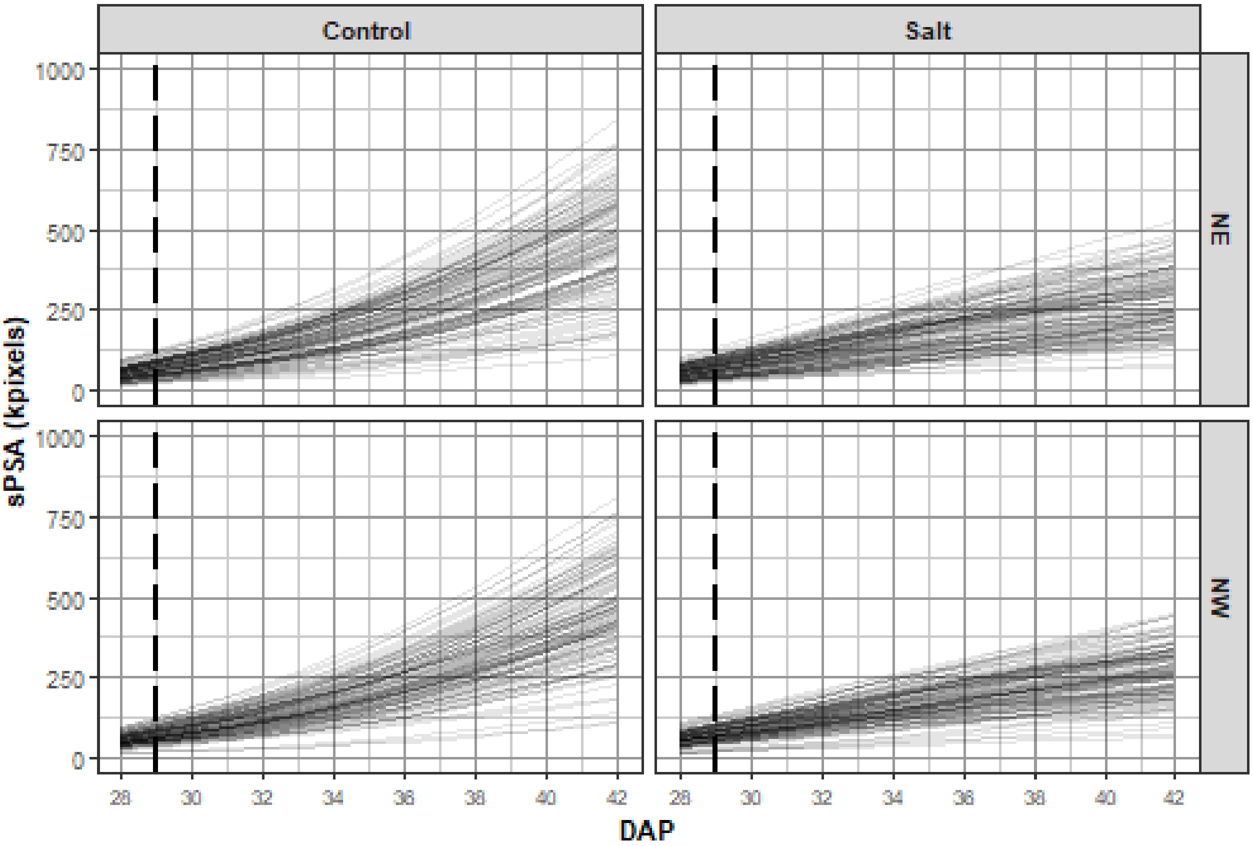


```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```

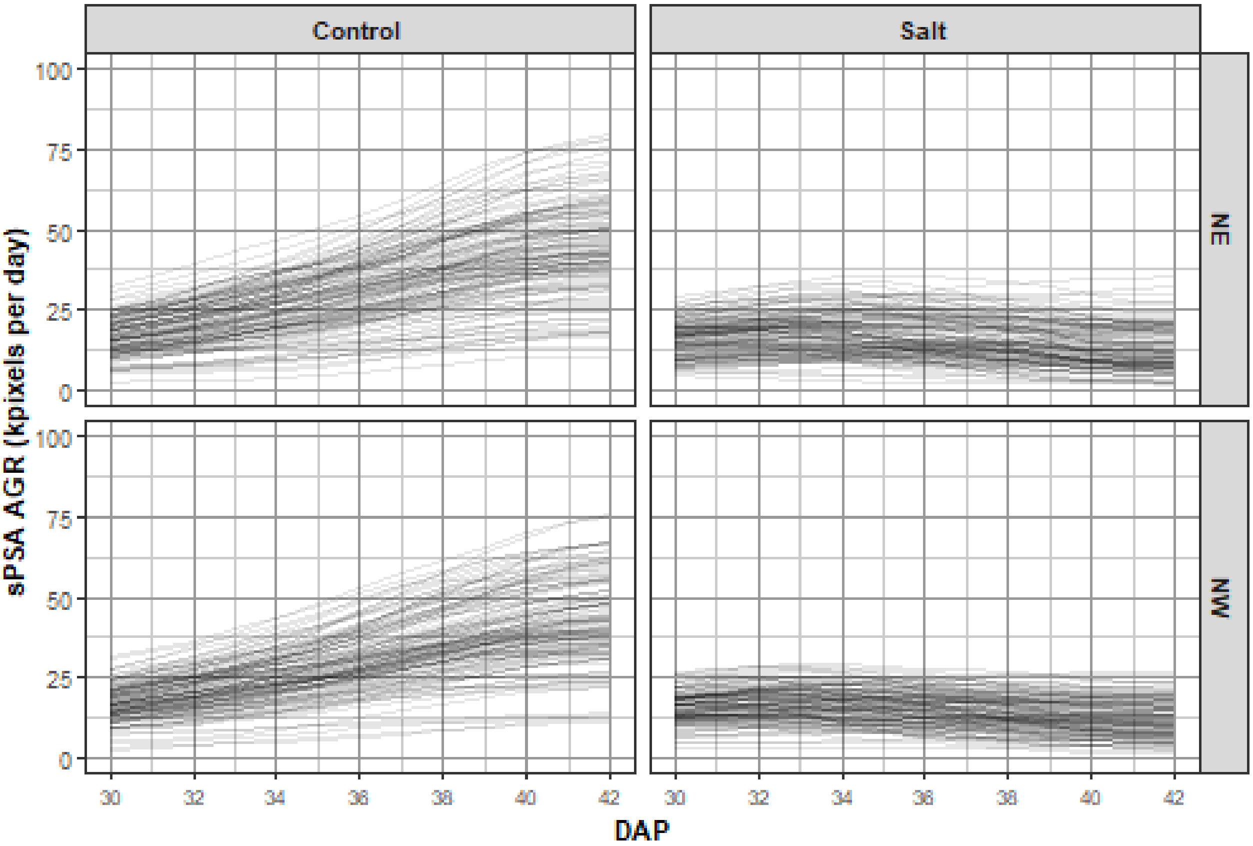


Plot smoothed profiles for all longitudinal responses

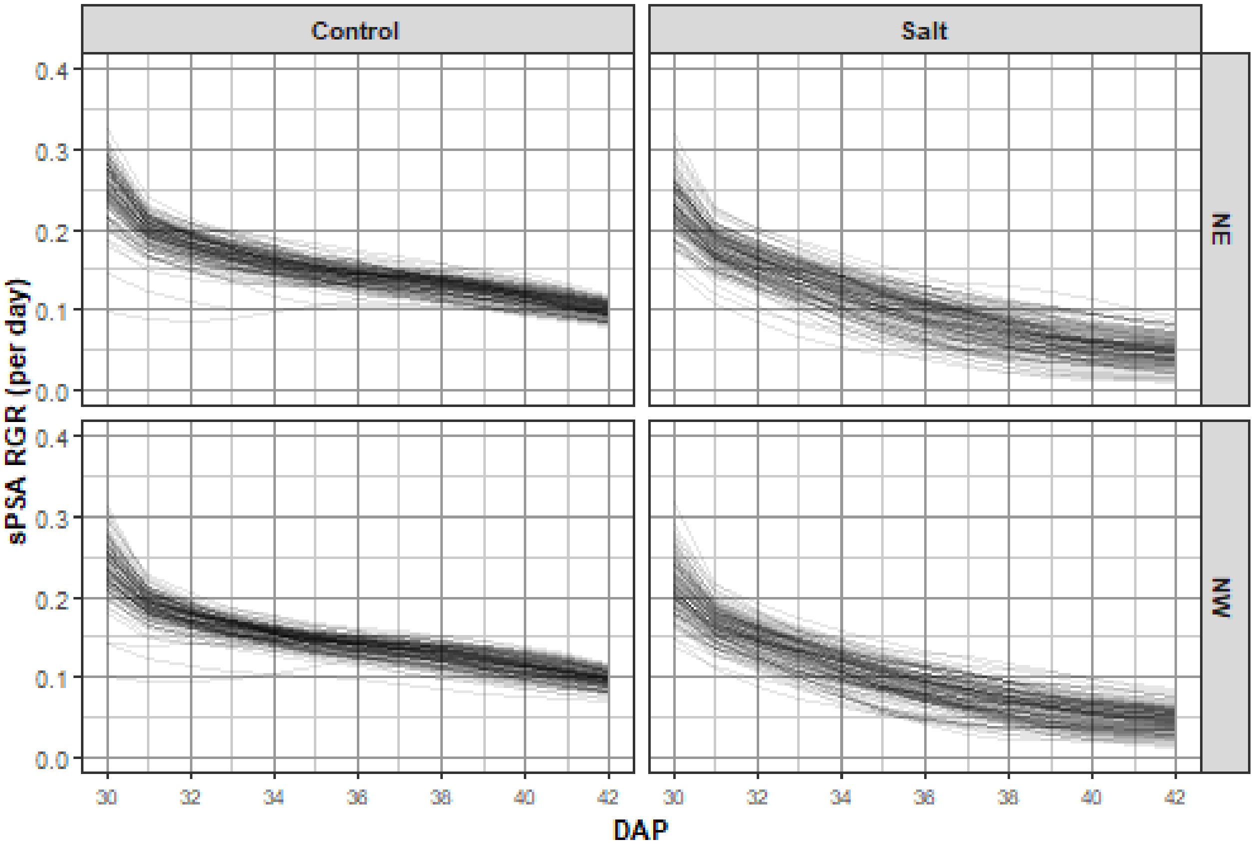
```r
nresp.smooth <- length(responses.smooth.plot)
limits <- list(c(0,1000), c(0,100), c(0.0,0.40))
for (k in 1:nresp.smooth)
{
  plt <- plotProfiles(data = longi.dat, response = responses.smooth.plot[k],
                      y.title = titles.smooth[k], times = "DAP",
                      facet.x = "Treatment.1", facet.y = "Smarthouse",
                      breaks.spacing.x = 2,
                      printPlot=FALSE)
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", linewidth=1) +
               scale_y_continuous(limits=limits[[k]])
  print(plt)
}
```



```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_vline()`).
```



Step 3: Choose the smoothing method and DF

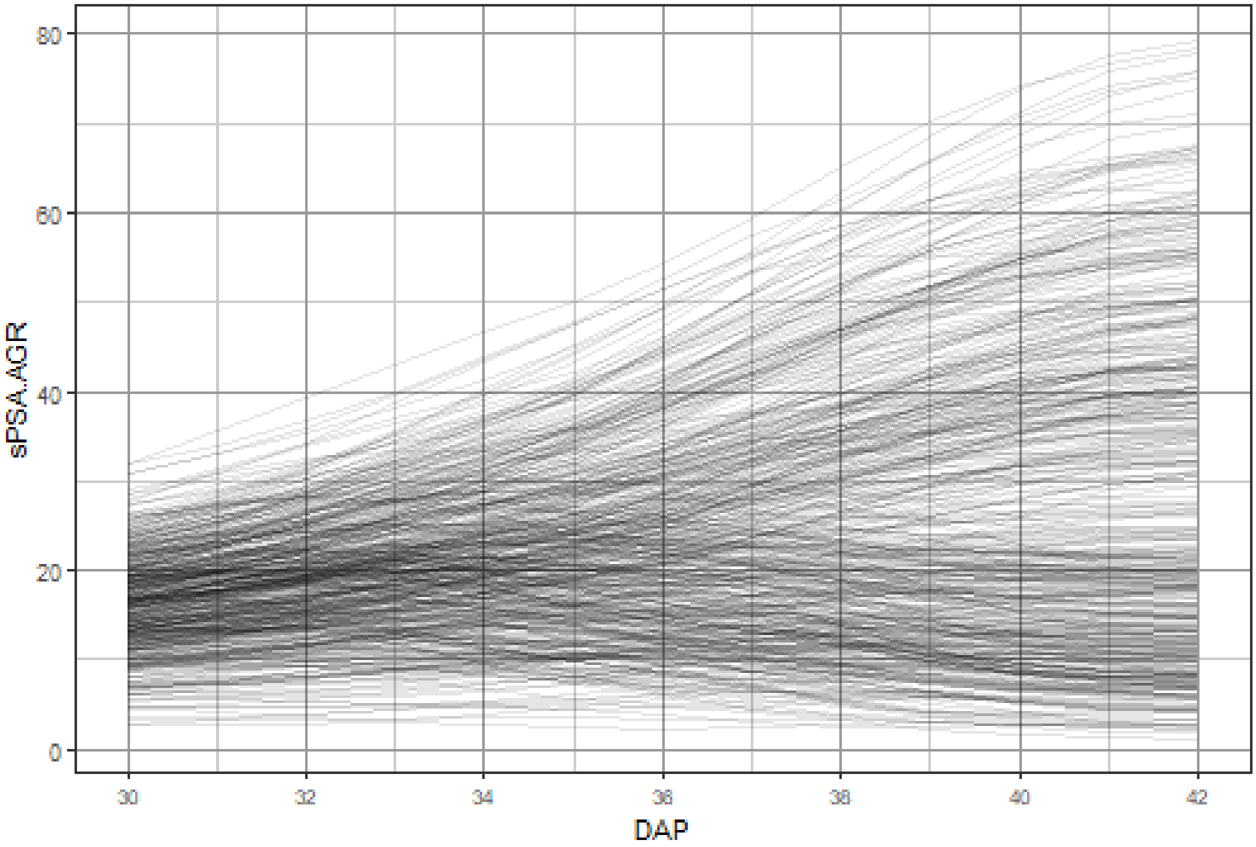
This step has been omitted.

Step 4: Identify potential outlers and clean the data

It has been decided that plants whose smoothed AGR are less than 2.5 after Day 40 are growing so slowly as to be considered anomalous. These plants are identified using `plotAnom`. Their values on Day 42 are printed. The plants are plotted without the anomalous plants followed by a plot of just the anomalous plants. The images of these anomlaous plants were examined and no particular problems were identified with them. They were retained in the data.

```
anom.ID <- vector(mode = "character", length = 0L)
response <- "sPSA.AGR"
cols.output <- c("Snapshot.ID.Tag", "Smarthouse", "Lane", "Position",
                 "Treatment.1", "Genotype.ID", "Replicate", "DAP")
anomalous <- plotAnom(longi.dat, response=response, lower=2.5, start.time=40,
                      times = "DAP", vertical.line=29, breaks.spacing.x = 2,
                      whichPrint=c("innerPlot"), y.title=response)
```

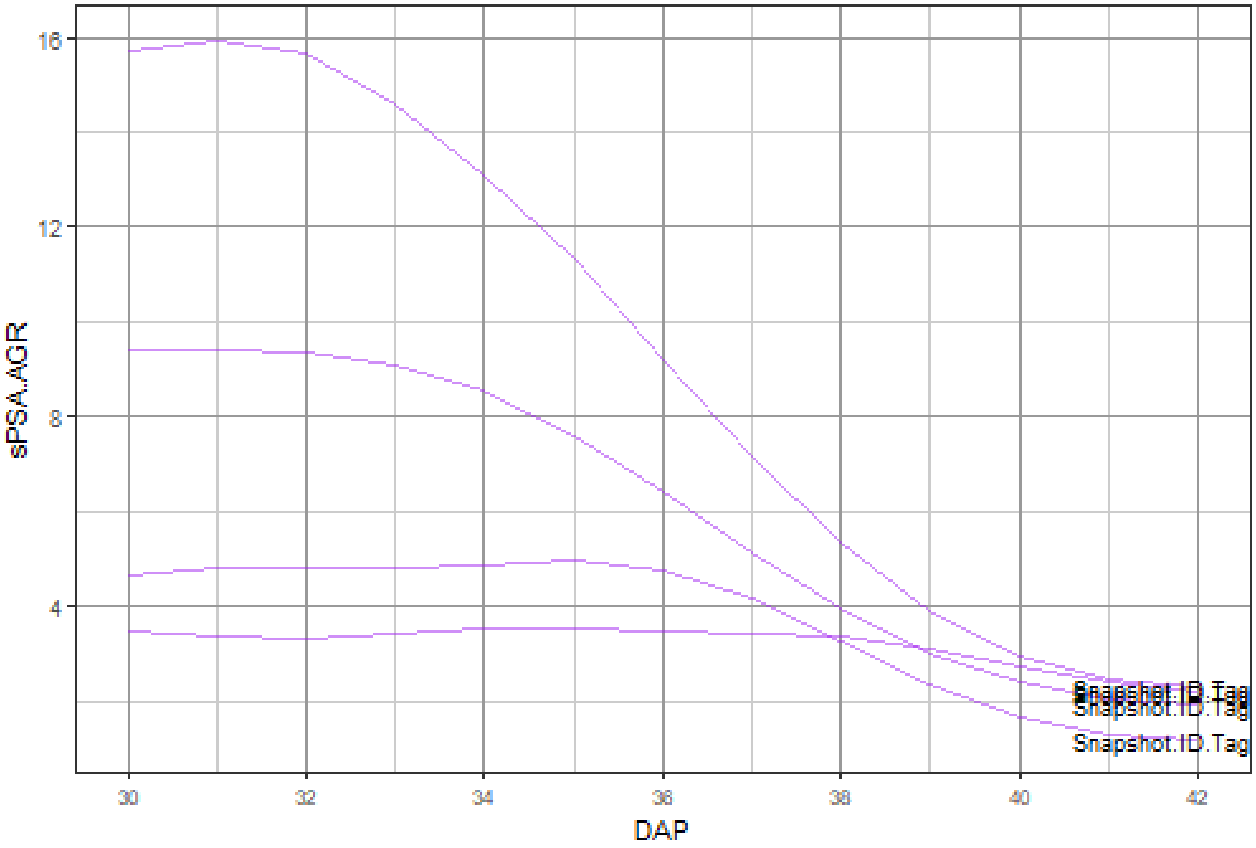
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



```
subs <- subset(anomalous$data, sPSA.AGR.anom & DAP==42)
if (nrow(subs) == 0)
{ cat("\n#### No anomalous data here\n\n")
} else
{
  subs <- subs[order(subs[["Smarthouse"]],subs[["Treatment.1"]], subs[[response]]),]
  print(subs[c(cols.output, response)])
  anom.ID <- unique(c(anom.ID, subs$Snapshot.ID.Tag))
  outerPlot <- anomalous$outerPlot  + geom_text(data=subs,
                                                aes(x = DAP,
                                                    y = .data[[response]],
                                                    label="Snapshot.ID.Tag"),
                                                size=3, hjust=0.7, vjust=0.5)
  print(outerPlot)
}
```

```
##      Snapshot.ID.Tag Smarthouse Lane Position Treatment.1 Genotype.ID Replicate
## 1680        045575-S         NE    6       10        Salt      121701         1
## 2534        045639-S         NE    9        6        Salt      122000         1
## 5586        046144-S         NW    7        5        Salt      121133         1
## 3836        046013-S         NW    1       12        Salt      121852         2
##      DAP sPSA.AGR
## 1680  42 1.926575
## 2534  42 2.297119
## 5586  42 1.199223
## 3836  42 2.216099
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



Step 5: Extract per-cart traits

A range of single-value plant responses are formed in `Snapshot.ID.Tag` order.

#’### Set up intervals

```
DAP.endpts <- c(31,35,38,42)
DAP.starts <- c(31,35,31,38)
DAP.stops   <- c(35,38,38,42)
DAP.mids <- (DAP.starts + DAP.stops)/2
suffices <- paste(DAP.starts, DAP.stops, sep = "to")
```

Step 5(a): Set up a data frame with factors only

```
indv.dat <- longi.dat[longi.dat$DAP == DAP.endpts[1],
                      c("Smarthouse","Lane","Position","Snapshot.ID.Tag",
                        "cPosn","cMainPosn",
                        "Zone","cZone","SHZone","ZLane","ZMainunit", "Subunit",
                        "Genotype.ID","Treatment.1")]
indv.dat <- indv.dat[do.call(order, indv.dat), ]
```

Step 5(b): Get responses based on first and last date.

```
# Observation for first and last date
indv.dat <- cbind(indv.dat, getTimesSubset(data = longi.dat, responses = responses.image,
                                           times = "DAP", which.times = DAP.endpts[1],
                                           suffix = "first"))
indv.dat <- cbind(indv.dat, getTimesSubset(data = longi.dat, responses = responses.image,
                                           times = "DAP",
                                           which.times = DAP.endpts[length(DAP.endpts)],
                                           suffix = "last"))
indv.dat <- cbind(indv.dat, getTimesSubset(data = longi.dat, responses = "WUI.cum",
                                           times = "DAP",
                                           which.times = DAP.endpts[length(DAP.endpts)],
                                           suffix = "last"))
responses.smooth <- paste0("s", responses.image)
indv.dat <- cbind(indv.dat, getTimesSubset(data = longi.dat, responses = responses.smooth,
                                           times = "DAP", which.times = DAP.endpts[1],
                                           suffix = "first"))
indv.dat <- cbind(indv.dat, getTimesSubset(data = longi.dat, responses = responses.smooth,
                                           times = "DAP",
                                           which.times = DAP.endpts[length(DAP.endpts)],
                                           suffix = "last"))

# Growth rates over whole period.
(tottime <- DAP.endpts[length(DAP.endpts)] - DAP.endpts[1]) #= 11
```

```
## [1] 11
```

```
indv.dat <- within(indv.dat,
                   {
                     PSA.AGR.full <- (PSA.last - PSA.first)/tottime
                     PSA.RGR.full <- log(PSA.last / PSA.first)/tottime
                   })

# Calculate water index over whole period
  indv.dat <- merge(indv.dat,
                    byIndv4Intvl_WaterUse(data = longi.dat,
                                          water.use = "WU", response = "PSA",
                                          trait.types = c("WUI","WUR", "WU"),
                                          times = "DAP",
                                          start.time = DAP.endpts[1],
                                          end.time = DAP.endpts[length(DAP.endpts)]),
                    by = c("Snapshot.ID.Tag"))
```

Step 5(c): Add growth rates and water indices for intervals

```
# Growth rates for specific intervals from the smoothed data by differencing
for (r in responses.smooth)
{
  for (k in 1:length(suffices))
```

```
  {
    indv.dat <- merge(indv.dat,
                      byIndv4Intvl_GRsDiff(data = longi.dat, responses = r,
                                           times = "DAP",
                                           which.rates = c("AGR","RGR"),
                                           start.time = DAP.starts[k],
                                           end.time = DAP.stops[k],
                                           suffix.interval = suffices[k]),
                      by = "Snapshot.ID.Tag")
  }
}

# Water indices for specific intervals from the unsmoothed and smoothed data
for (k in 1:length(suffices))
{
    indv.dat <- merge(indv.dat,
                      byIndv4Intvl_WaterUse(data = longi.dat,
                                            water.use = "WU", responses = "PSA",
                                            times = "DAP",
                                            trait.types = c("WU","WUR","WUI"),
                                            start.time = DAP.starts[k],
                                            end.time = DAP.stops[k],
                                            suffix.interval = suffices[k]),
                      by = "Snapshot.ID.Tag")
}

indv.dat <- with(indv.dat, indv.dat[order(Snapshot.ID.Tag), ])
```

Form continuous and interval SIITs

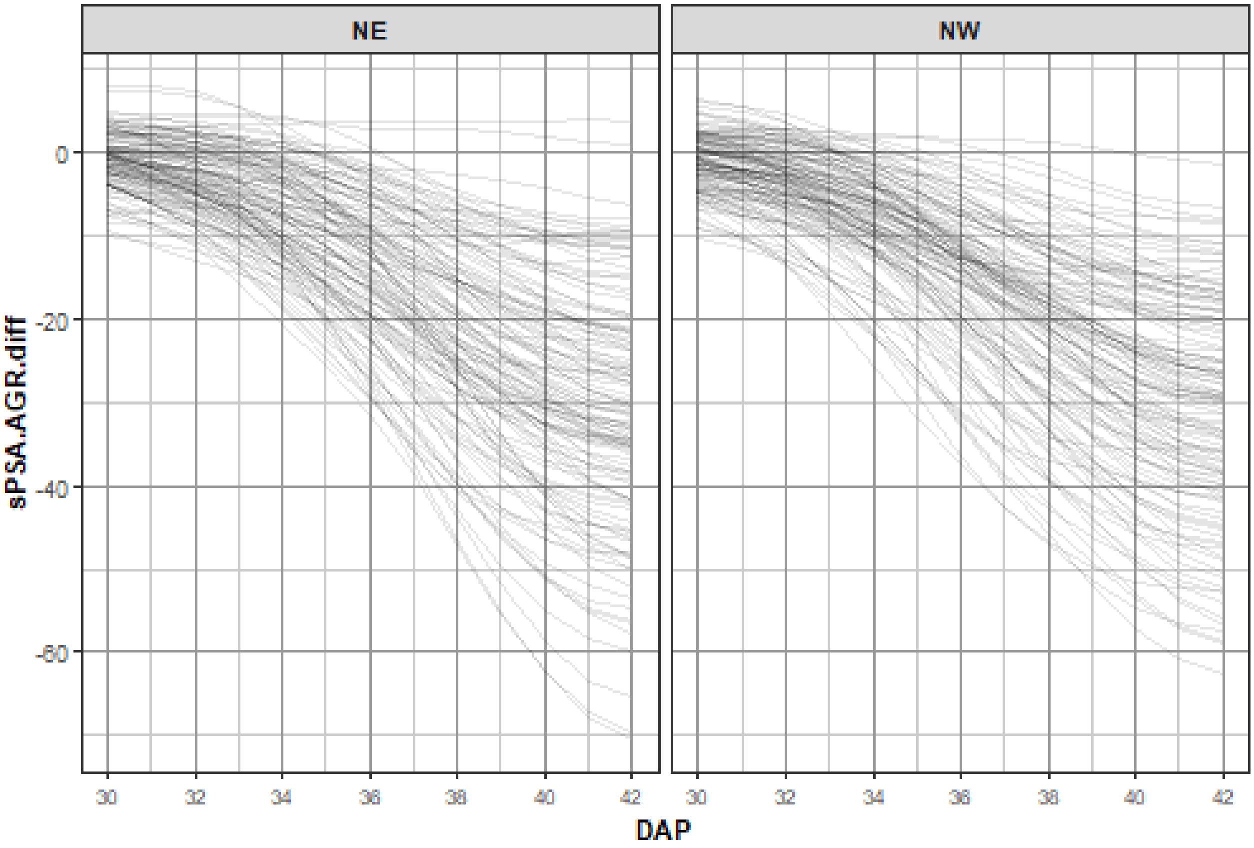
This experiment involved the extra step of calculating a measure of shoot ion-independent tolerance (SIIT) of pairs of plants, control and a salt-treated co-located plants.

Calculate continuous values

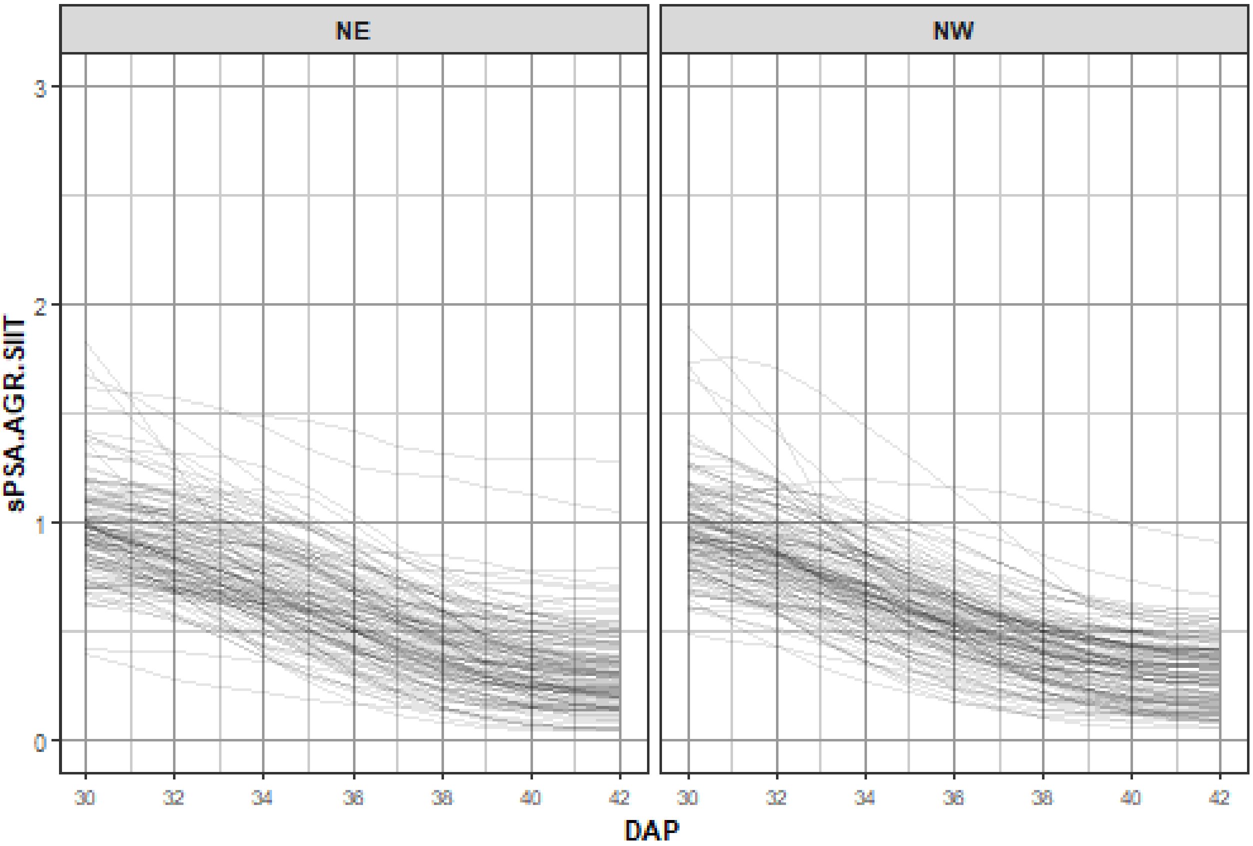
```
cols.retained <-  c("Snapshot.ID.Tag","Smarthouse","Lane","Position",
                    "DAP","Snapshot.Time.Stamp", "Hour", "xDAP",
                    "Zone","cZone","SHZone","ZLane","ZMainunit",
                    "cMainPosn", "Genotype.ID")
responses.GR <- c("sPSA.AGR","sPSA.AGR","sPSA.RGR")
suffices.results <- c("diff", "SIIT", "SIIT")
responses.SIIT <- unlist(Map(paste, responses.GR, suffices.results,sep="."))

longi.SIIT.dat <-
  twoLevelOpcreate(data = longi.dat, responses = responses.GR, suffices.treatment=c("C","S"),
                   operations = c("-", "/", "/"), suffices.results = suffices.results,
                   columns.retained = cols.retained,
                   by = c("Smarthouse","Zone","ZMainunit","DAP"))
longi.SIIT.dat <- with(longi.SIIT.dat,
                       longi.SIIT.dat[order(Smarthouse,Zone,ZMainunit,DAP),])
```

```r
# Plot SIIT profiles
k <- 2
nresp <- length(responses.SIIT)
limits <- with(longi.SIIT.dat, list(c(min(sPSA.AGR.diff, na.rm=TRUE),
                                       max(sPSA.AGR.diff, na.rm=TRUE)),
                                     c(0,3),
                                     c(0,1.5)))
#Plots
for (k in 1:nresp)
{
  plt <- plotProfiles(data = longi.SIIT.dat, times = "DAP",
                      response = responses.SIIT[k],
                      y.title=responses.SIIT[k],
                      facet.x="Smarthouse", facet.y=".",
                      breaks.spacing.x = 2, printPlot=FALSE, )
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", linewidth=1) +
              scale_y_continuous(limits=limits[[k]])
  print(plt)
}
```

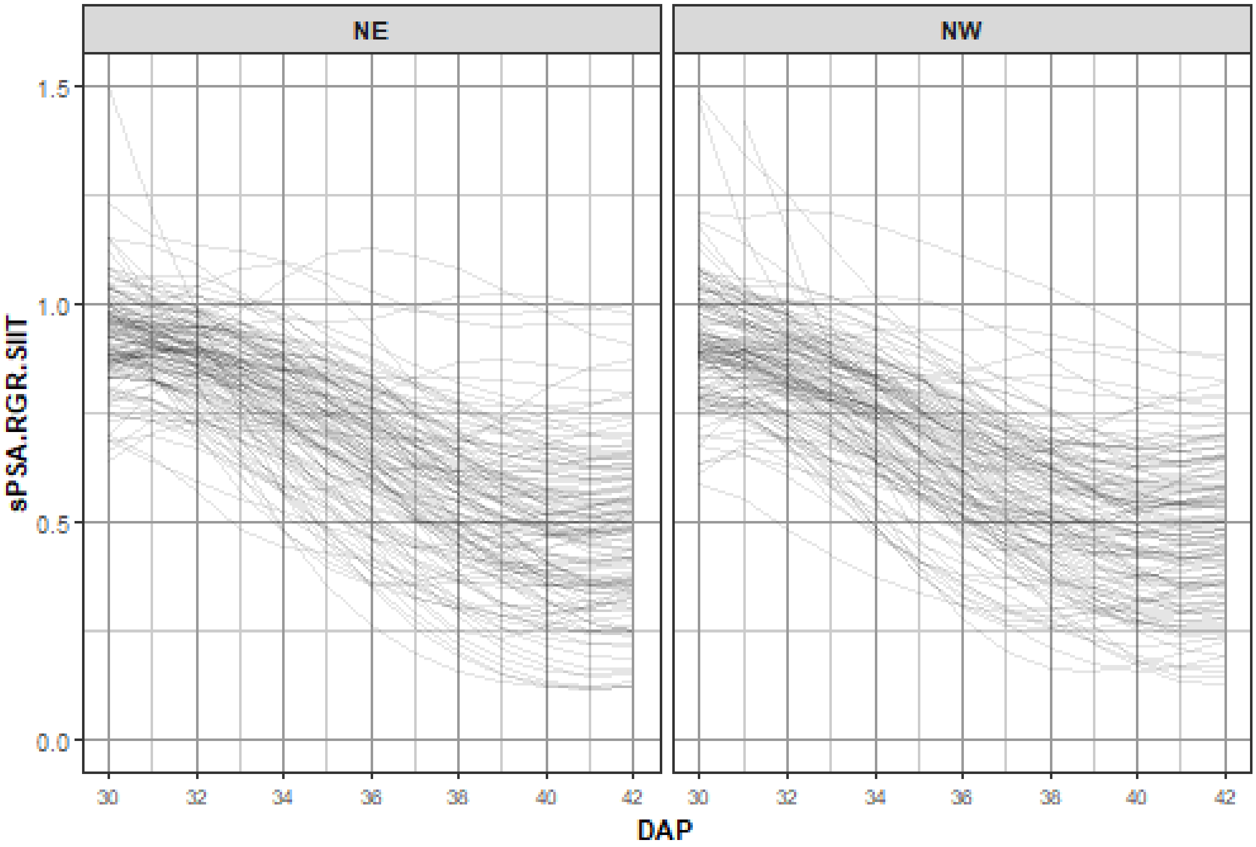
```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_line()‘).
## Removed 2 rows containing missing values or values outside the scale range
## (‘geom_vline()‘).
```



Calculate interval SIITs and check for large values for SIIT for Days 31to35

```
response <- "sPSA.RGR.31to35"
SIIT <- paste(response, "SIIT", sep=".")
responses.SIITinterval <- as.vector(outer("sPSA.RGR", suffices, paste, sep="."))

indv.SIIT.dat <- twoLevelOpcreate(data = indv.dat, responses = responses.SIITinterval,
                                  suffices.treatment=c("C","S"),
                                  suffices.results="SIIT",
                                  columns.suffixed="Snapshot.ID.Tag")

tmp<-na.omit(indv.SIIT.dat)
print(summary(tmp[SIIT]))
```

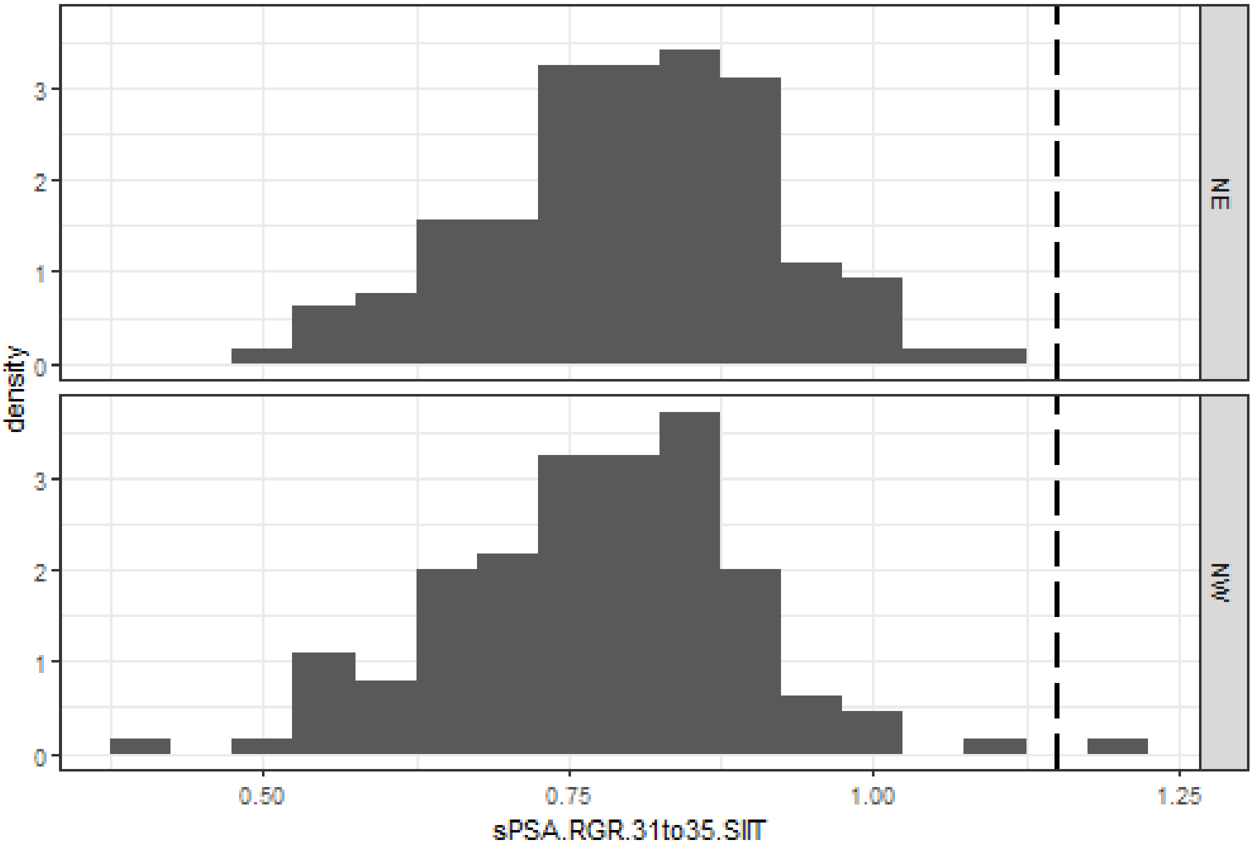
```
##  sPSA.RGR.31to35.SIIT
##  Min.   :0.4077
##  1st Qu.:0.7120
##  Median :0.7961
##  Mean   :0.7876
##  3rd Qu.:0.8663
##  Max.   :1.1885
```

```
big.SIIT <- with(tmp, tmp[tmp[SIIT] > 1.15, c("Snapshot.ID.Tag.C","Genotype.ID",
                                              paste(response,"C",sep="."),
```

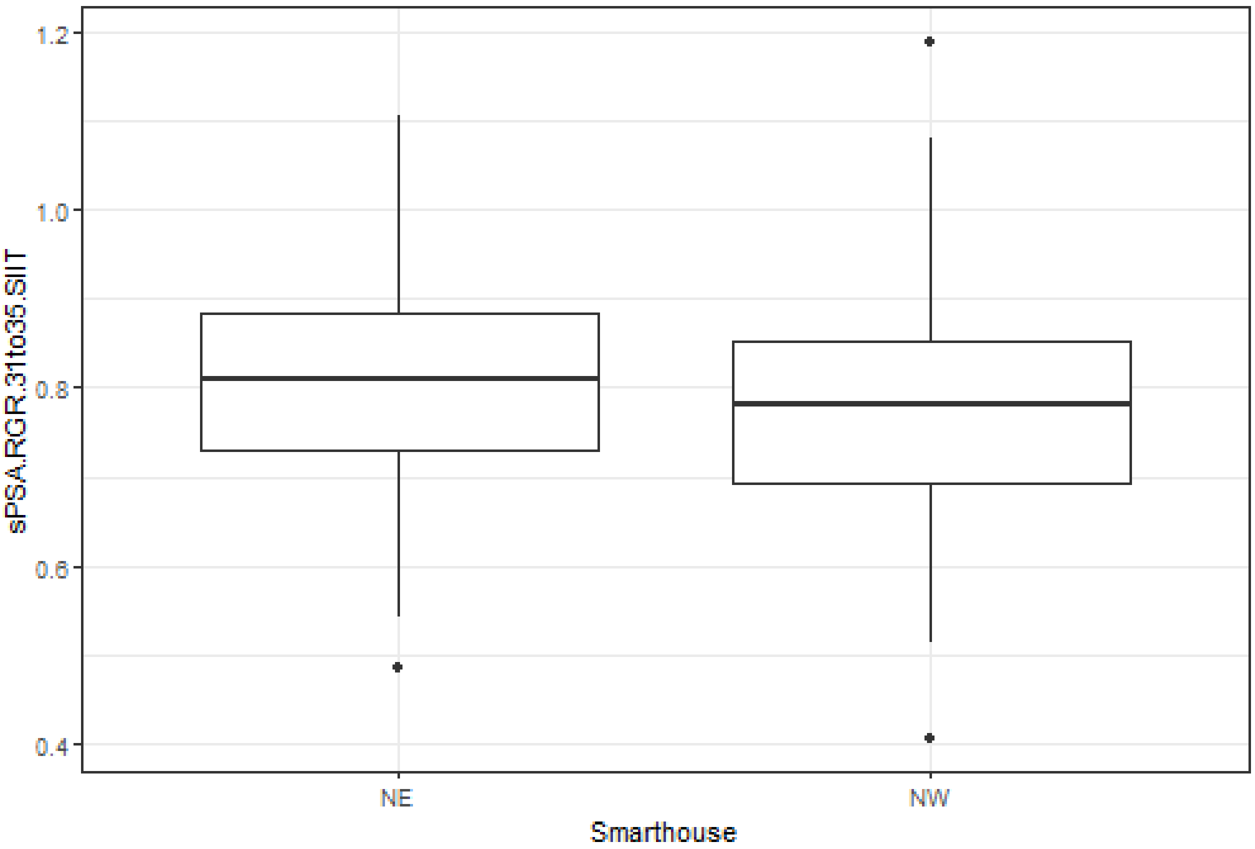
```
                                                    paste(response,"S",sep="."), SIIT)])
if (nrow(big.SIIT) > 1)
  big.SIIT <- big.SIIT[order(big.SIIT[SIIT]),]
print(big.SIIT)
```

```
##     Snapshot.ID.Tag.C Genotype.ID sPSA.RGR.31to35.C sPSA.RGR.31to35.S
## 193          046129-C      122090         0.1310631         0.1557642
##     sPSA.RGR.31to35.SIIT
## 193             1.188467
```

```
plt <- ggplot(tmp, aes(.data[[SIIT]])) +
          geom_histogram(aes(y = after_stat(density)), binwidth=0.05) +
          geom_vline(xintercept=1.15, linetype="longdash", linewidth=1) +
          theme_bw() + facet_grid(Smarthouse ~.)
print(plt)
```



```
plt <- ggplot(tmp, aes(x=Smarthouse, y=.data[[SIIT]])) +
          geom_boxplot() + theme_bw()
print(plt)
```



```r
remove(tmp)
```

Save image

```r
save.image("Rice.RData")
```

References

Al-Tamimi, N, Brien, C.J., Oakey, H., Berger, B., Saade, S., Ho, Y. S., Schmockel, S. M., Tester, M. and Negrao, S. (2016) New salinity tolerance loci revealed in rice using high-throughput non-invasive phenotyping. *Nature Communications*, **7**, 13342.

Brien, C. J. (2025g) `growthPheno`: *Functional Analysis of Phenotypic Growth Data to Smooth and Extract Traits*. Version 3.1.12. https://cran.r-project.org/package=growthPheno.

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J., & Thompson, R. (2023). *ASReml-R reference manual*, Version 4.2. http://asreml.org.

Brien, C., Jewell, N., Garnett, T., Watts-Williams, S. J., & Berger, B. (2020). Smoothing and extraction of traits in the growth analysis of noninvasive phenotypic data. *Plant Methods*, **16**, 36. http://dx.doi.org/10.1186/s13007-020-00577-6.

R Core Team (2025) *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.r-project.org.