

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

Chris Brien

02 July, 2022

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.

## 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 <- getImageData(data=RiceRaw.dat, smarthouse.lev=c("NE","NW"))

longi.dat <- designFactors(data = longi.dat, insertName = "Reps",
                           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

```
# 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)

```

```

## Snapshot.ID.Tag DAP Smarthouse Lane Position xDAP Snapshot.Time.Stamp
## 1 045727-C 28 NE 13 2 28 2015-02-18 05:31:00
## 2 045727-C 30 NE 13 2 30 2015-02-20 05:23:00
## 3 045727-C 31 NE 13 2 31 2015-02-21 05:23:00
## 4 045727-C 32 NE 13 2 32 2015-02-22 05:23:00
## 5 045727-C 33 NE 13 2 33 2015-02-23 05:24:00
## 6 045727-C 34 NE 13 2 34 2015-02-24 10:15:00
## Hour Reps Zone cZone SHZone ZLane ZMainunit Subunit cMainPosn cPosn
## 1 5.516667 1 1 -1 1 1 1 1 -10.5 -11
## 2 5.383333 1 1 -1 1 1 1 1 -10.5 -11
## 3 5.383333 1 1 -1 1 1 1 1 -10.5 -11
## 4 5.383333 1 1 -1 1 1 1 1 -10.5 -11
## 5 5.400000 1 1 -1 1 1 1 1 -10.5 -11
## 6 10.250000 1 1 -1 1 1 1 1 -10.5 -11
## Genotype.ID Treatment.1 Weight.Before Weight.After Water.Amount WU PSA
## 1 121146 Control 4013 4032 22 NA 55.311
## 2 121146 Control 4062 4085 26 -30 80.130
## 3 121146 Control 4040 4085 48 45 94.788
## 4 121146 Control 4032 4086 56 53 108.613
## 5 121146 Control 4027 4086 61 59 133.677
## 6 121146 Control 4012 4086 76 74 157.847
## PSA.SV1 PSA.SV2 PSA.TV Boundary.Points.To.PSA.Ratio.SV1
## 1 11.307 15.456 28.548 0.563633
## 2 25.816 21.768 32.546 0.342539
## 3 31.627 23.604 39.557 0.346413
## 4 37.702 30.704 40.207 0.342316
## 5 39.861 40.317 53.499 0.376308
## 6 48.086 46.873 62.888 0.320904
## Boundary.Points.To.PSA.Ratio.SV2 Boundary.Points.To.PSA.Ratio.TV
## 1 0.371442 0.233571
## 2 0.439085 0.220304
## 3 0.411922 0.230477
## 4 0.365262 0.228343
## 5 0.346876 0.222883
## 6 0.367504 0.219342
## Caliper.Length.SV1 Caliper.Length.SV2 Caliper.Length.TV Compactness.SV1
## 1 736.872 792.324 888.821 0.0491248
## 2 728.754 785.611 797.924 0.0994986
## 3 779.808 889.427 797.332 0.1061870
## 4 956.613 896.909 861.304 0.1008410
## 5 1076.500 1123.540 1202.150 0.0809829
## 6 1016.490 1235.360 1277.480 0.1033860
## Compactness.SV2 Compactness.TV Convex.Hull.PSA.SV1 Convex.Hull.PSA.SV2

```

```
## 1      0.0815964      0.1069410      230.169      189.420
## 2      0.0683533      0.1051310      259.461      318.463
## 3      0.0658089      0.1030830      297.843      358.675
## 4      0.0733596      0.0916347      373.877      418.541
## 5      0.0796217      0.0908369      492.215      506.357
## 6      0.0621912      0.0986004      465.109      753.692
## Convex.Hull.PSA.TV Center.Of.Mass.Y.SV1 Center.Of.Mass.Y.SV2
## 1      266.952      1822.21      1785.29
## 2      309.575      1809.31      1808.03
## 3      383.738      1815.14      1826.45
## 4      438.775      1827.60      1874.88
## 5      588.957      1823.38      1861.62
## 6      637.807      1830.35      1843.54
## Max.Dist.Above.Horizon.Line.SV1 Max.Dist.Above.Horizon.Line.SV2 xDAT
## 1      612      626 -1
## 2      637      636 1
## 3      591      628 2
## 4      650      691 3
## 5      599      618 4
## 6      695      707 5
## DAP.diffs PSA.AGR PSA.RGR PSA.WUI WUI.cum WU.cum
## 1      NA      NA      NA      NA      NA      NA
## 2      2 12.4095 0.1853393 -0.8273000 -2.6710000 -30
## 3      1 14.6580 0.1679925 0.3257333 6.3192000 15
## 4      1 13.8250 0.1361483 0.2608491 1.5972500 68
## 5      1 25.0640 0.2076353 0.4248136 1.0525748 127
## 6      1 24.1700 0.1661998 0.3266216 0.7853085 201
```

## 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

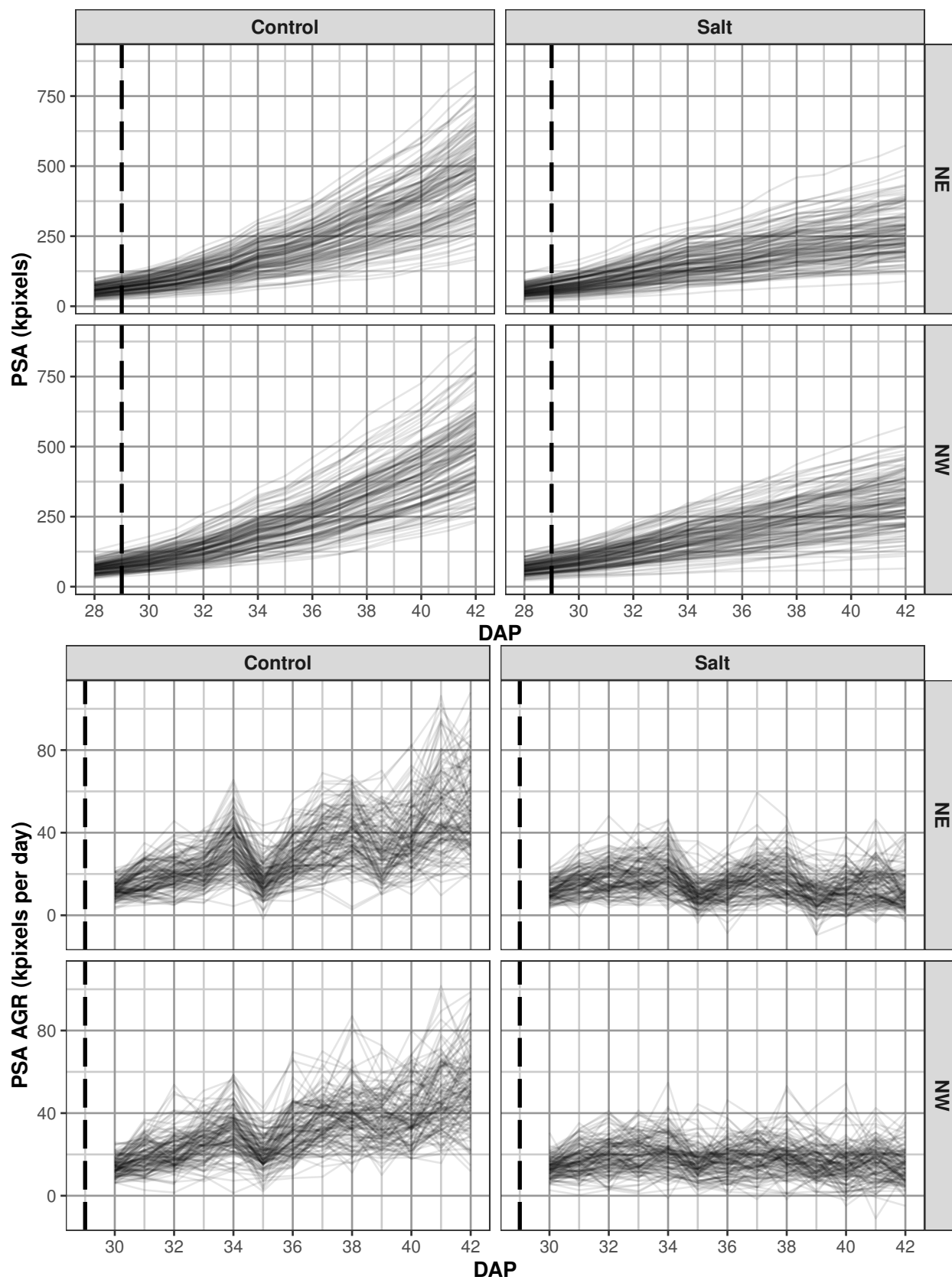
## Warning in log(PGR(x, time.diffs, lag = lag)): NaNs produced
# Finalize longi.dat
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag, xDAP), ])
```

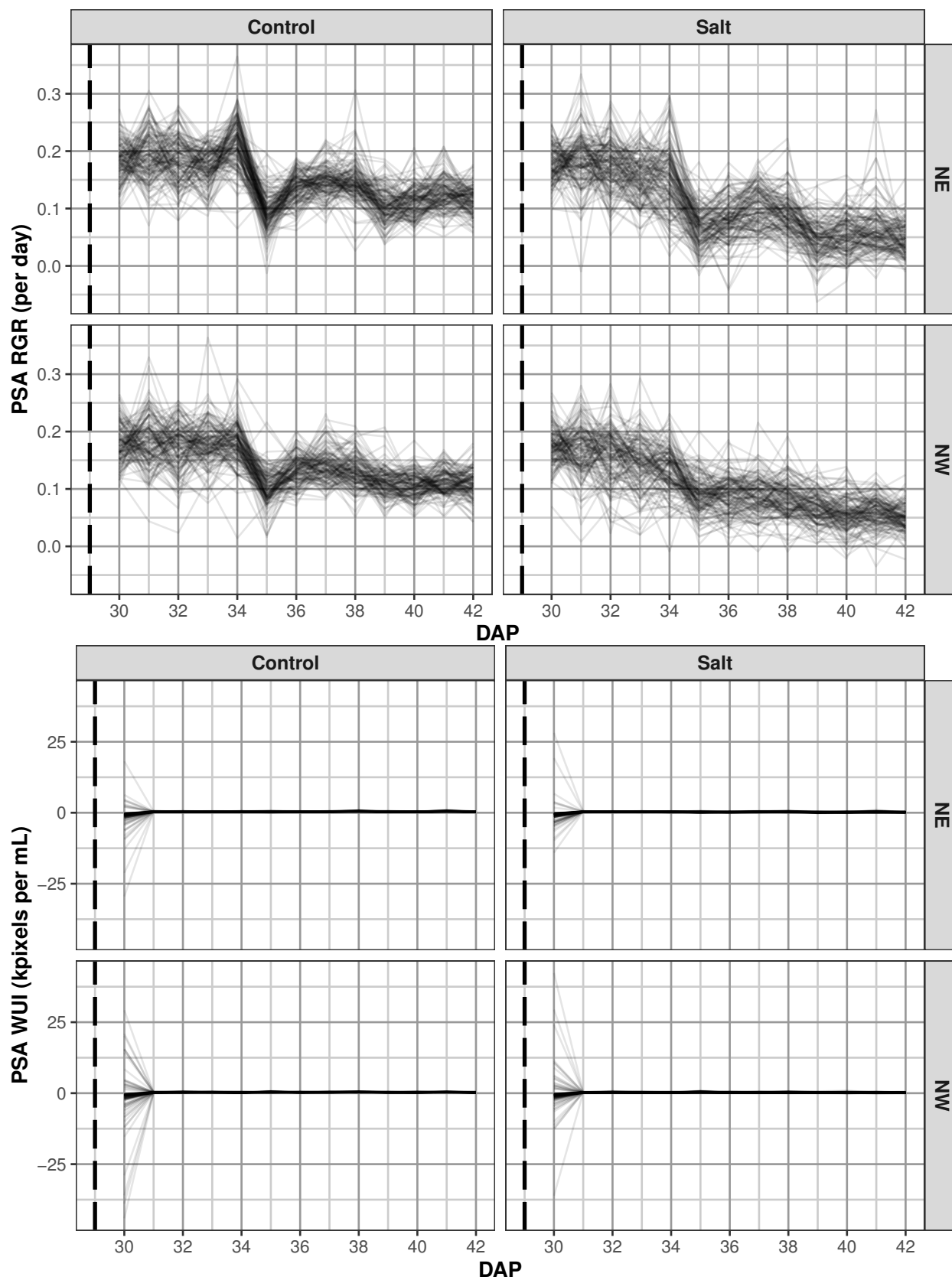
## Step 2(b): Compare plots of unsmoothed and smoothed longitudinal data

```
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

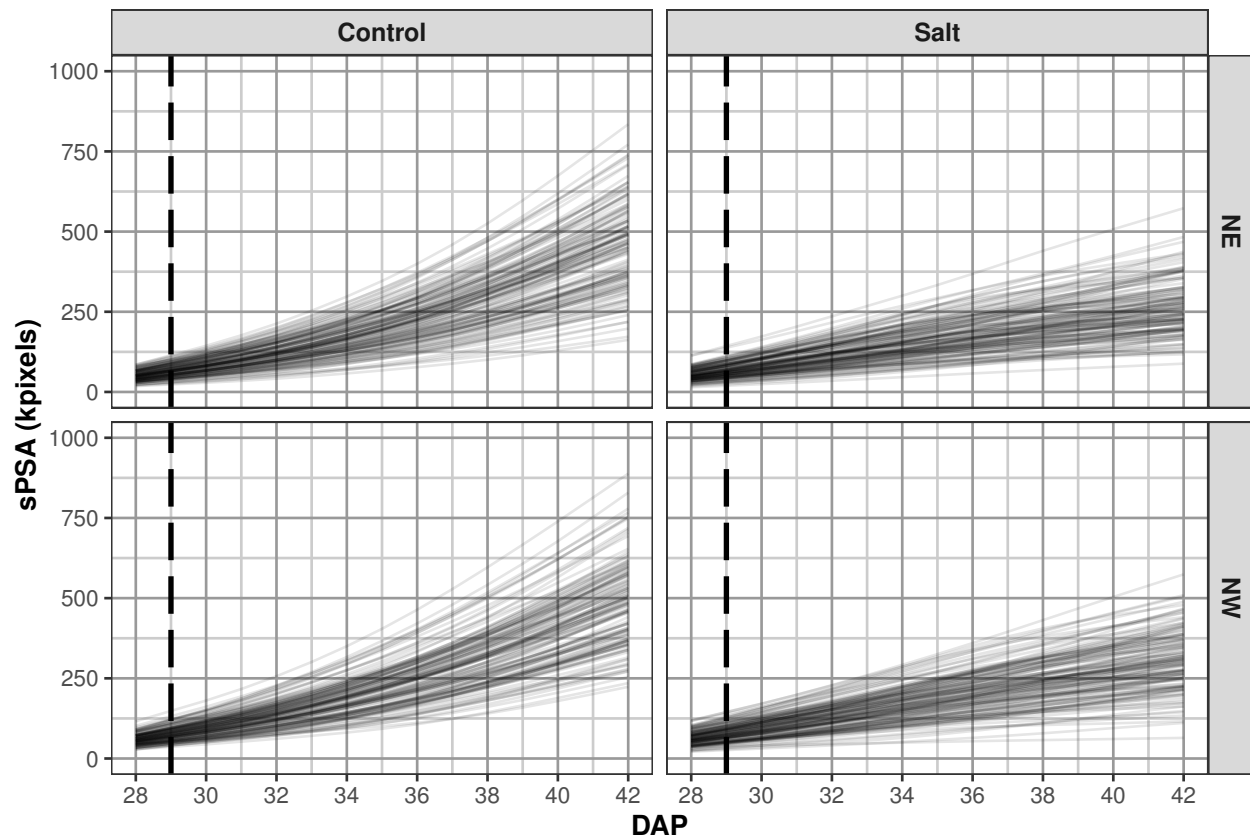
```
for (k in 1:nresp)
{
  plt <- plotProfiles(data = longi.dat, response = responses.longi[k],
                     y.title = titles[k], times = "DAP",
                     facet.x = "Treatment.1", facet.y = "Smarthouse",
                     printPlot=FALSE)
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", size=1) +
    scale_x_continuous(breaks=seq(28, 42, by=2))
  print(plt)
}
```

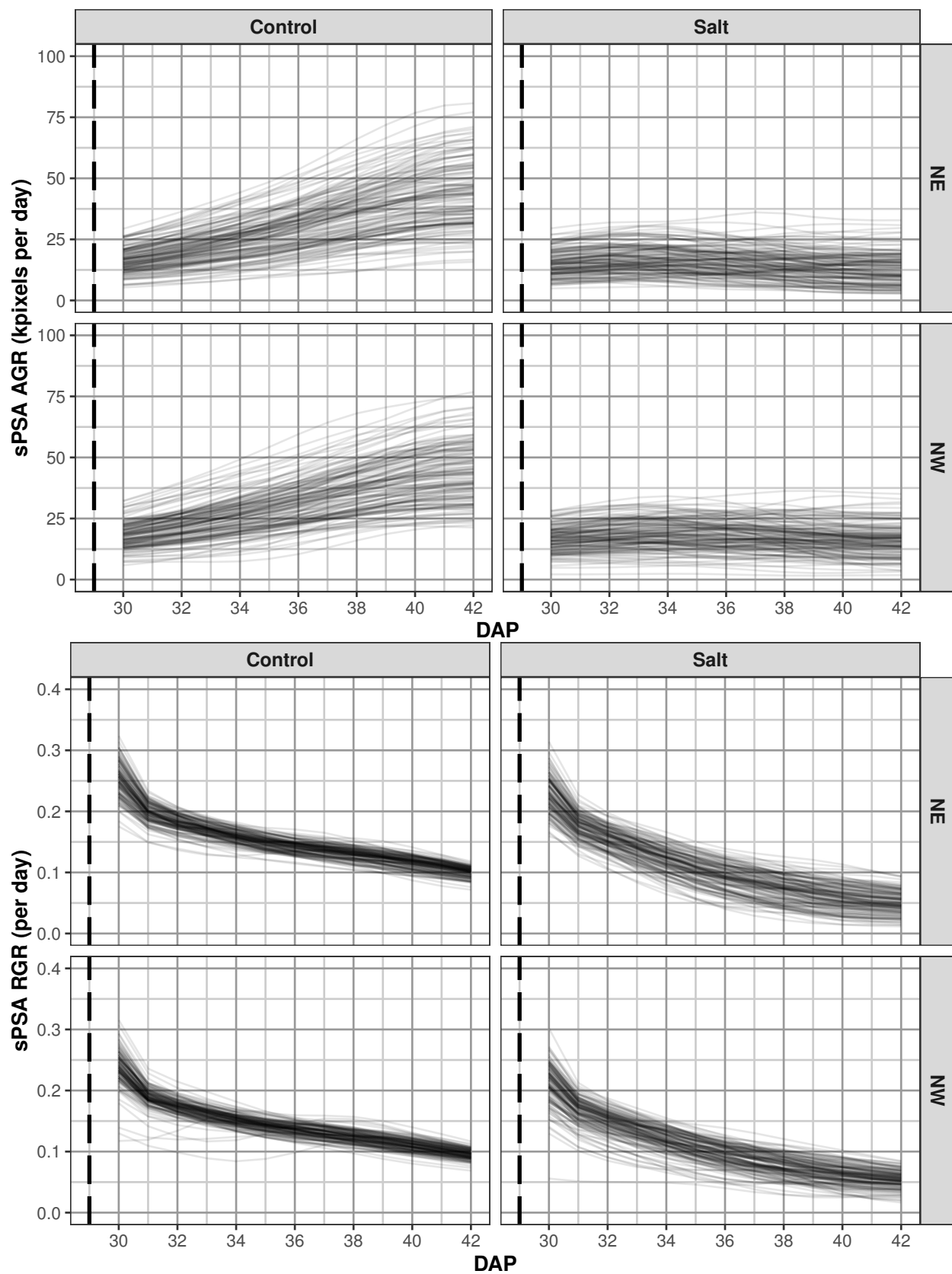




## Plot smoothed profiles for all longitudinal responses

```
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",
    printPlot=FALSE)
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", size=1) +
    scale_x_continuous(breaks=seq(28, 42, by=2)) +
    scale_y_continuous(limits=limits[[k]])
  print(plt)
}
```







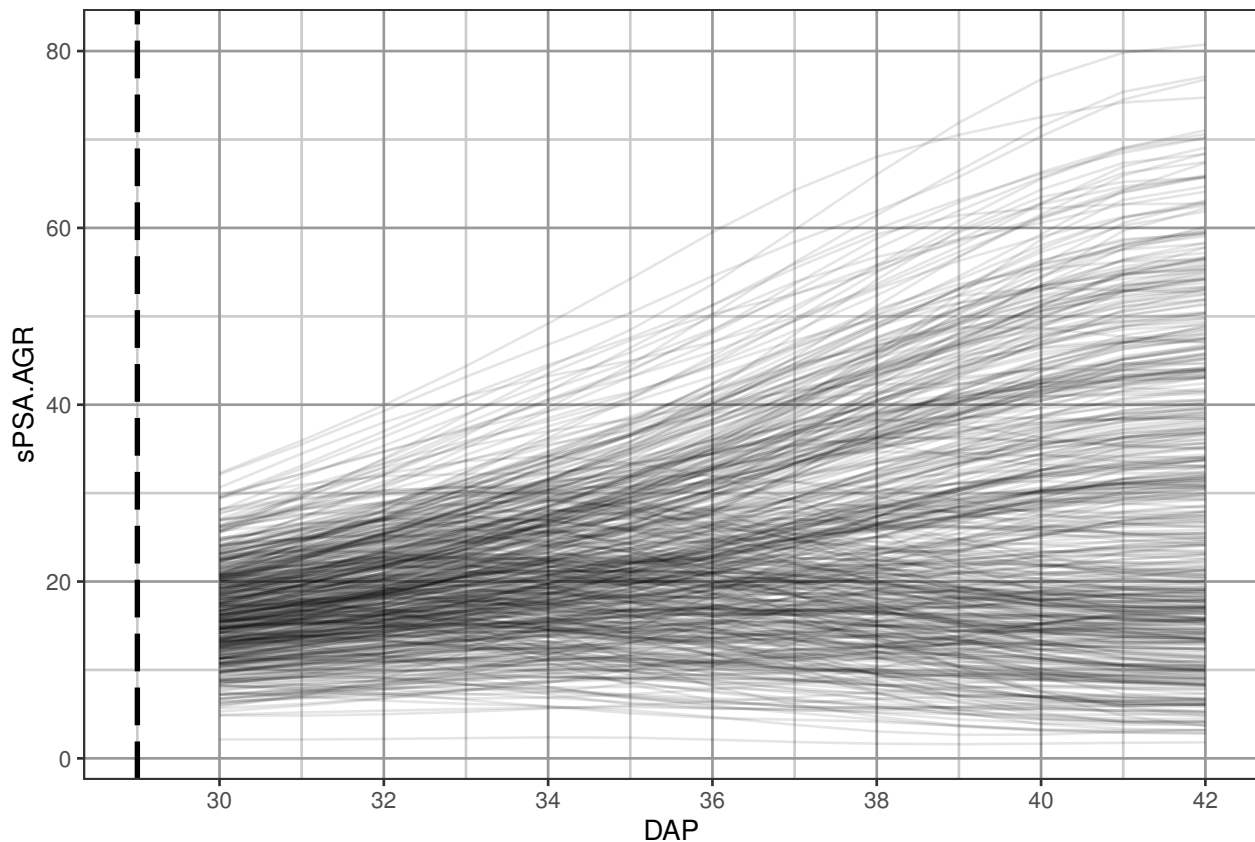
### Step 3: Choose the smoothing method and DF

This step has been omitted.

### Step 4: Identify potential outliers 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 anomalous 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", "DAP")
anomalous <- plotAnom(longi.dat, response=response, lower=2.5, start.time=40,
                     times = "DAP", vertical.line=29, breaks=seq(28, 42, by=2),
                     whichPrint=c("innerPlot"), y.title=response)
```



```
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_string(x = "DAP",
                                                         y = response,
                                                         label="Snapshot.ID.Tag"),
                                              size=3, hjust=0.7, vjust=0.5)

print(outerPlot)
}

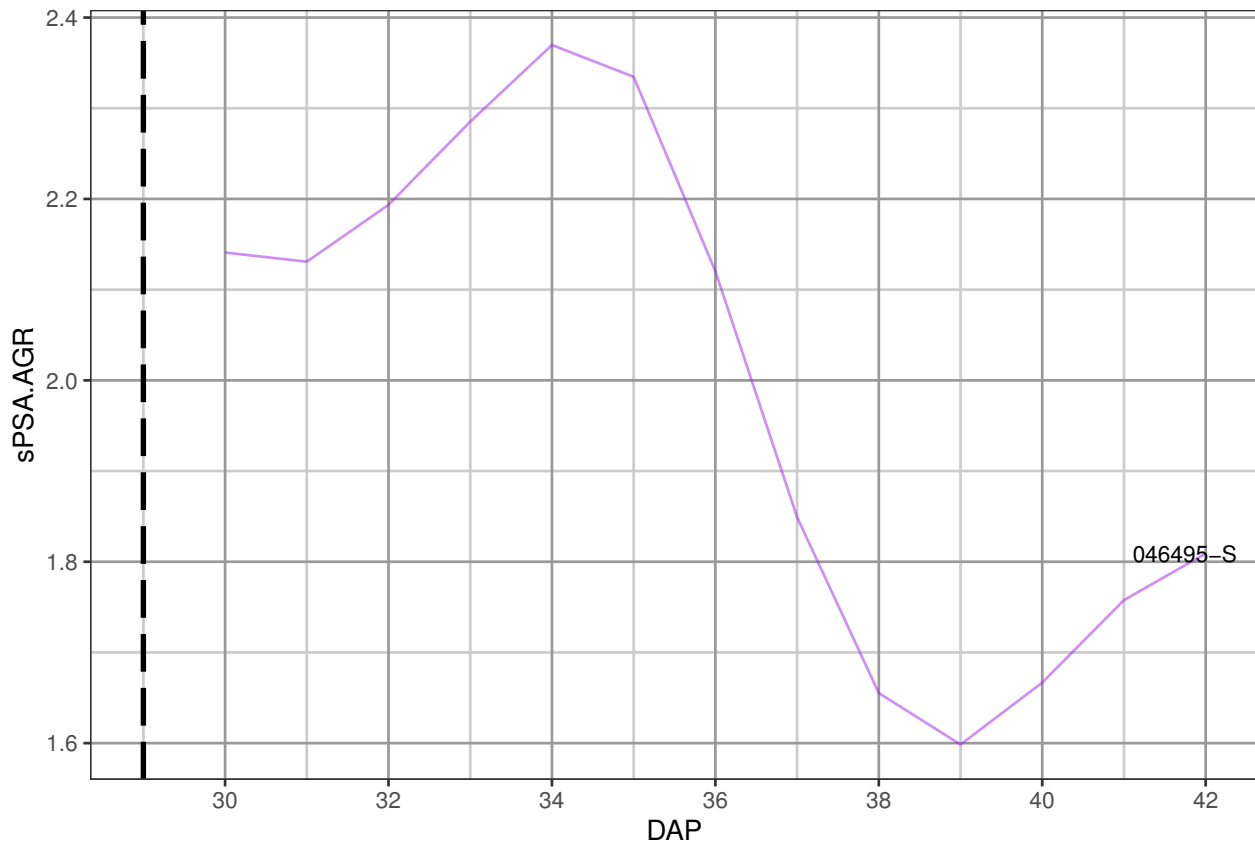
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
##      Snapshot.ID.Tag Smarthouse Lane Position Treatment.1 Genotype.ID DAP
## 6608      046495-S      NW      22      10      Salt      120952 42
##      sPSA.AGR
## 6608 1.809133
```



## 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

```
cart.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")]
cart.dat <- cart.dat[do.call(order, cart.dat), ]
```

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

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

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

# Calculate water index over whole period
cart.dat <- merge(cart.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))
  {
    cart.dat <- merge(cart.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))
{
  cart.dat <- merge(cart.dat,
                    byIndv4Intvl_WaterUse(data = longi.dat,
                                           water.use = "WU", responses = "PSA",
                                           times = "DAP",
                                           trait.types = c("WU","WUR","AGR","WUI"),
                                           start.time = DAP.starts[k],
                                           end.time = DAP.stops[k],
                                           suffix.interval = suffices[k]),
                    by = "Snapshot.ID.Tag")
}

cart.dat <- with(cart.dat, cart.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="."))

```

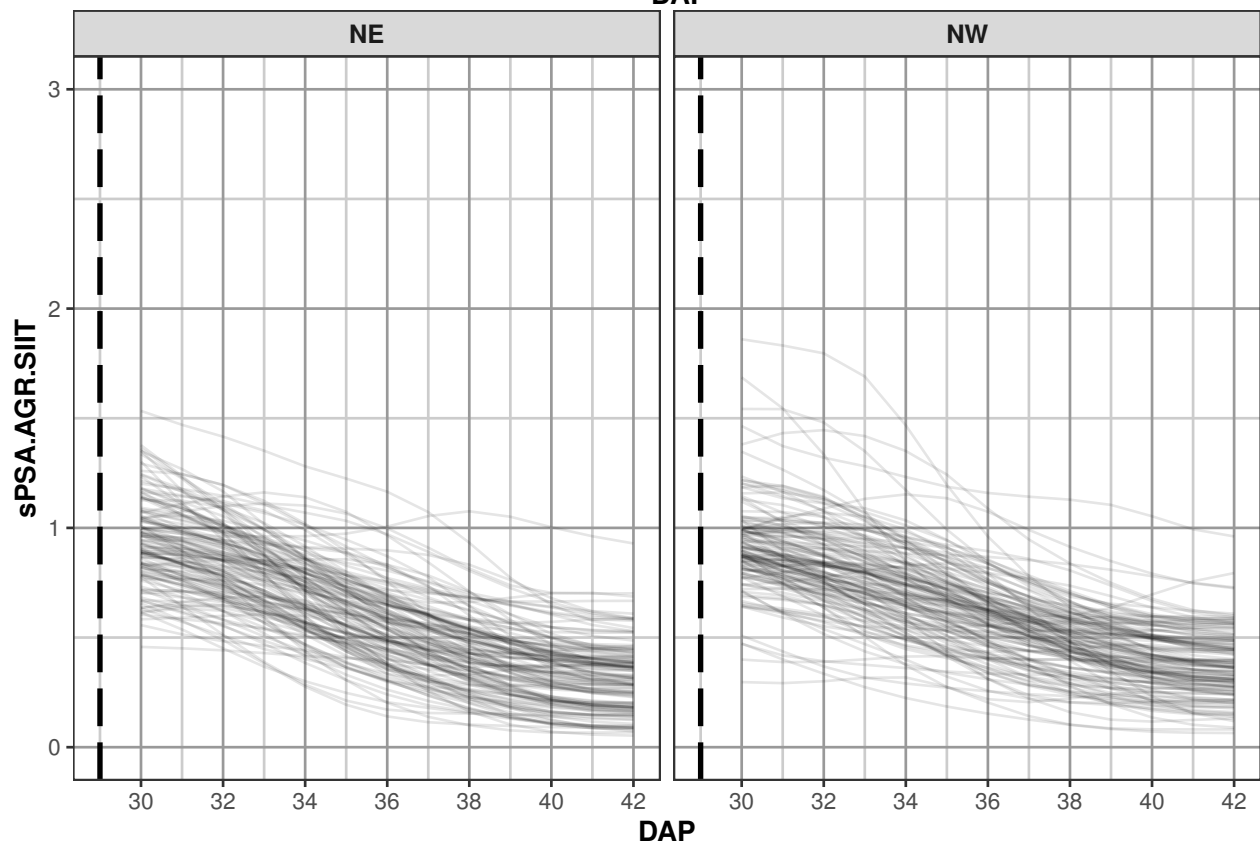
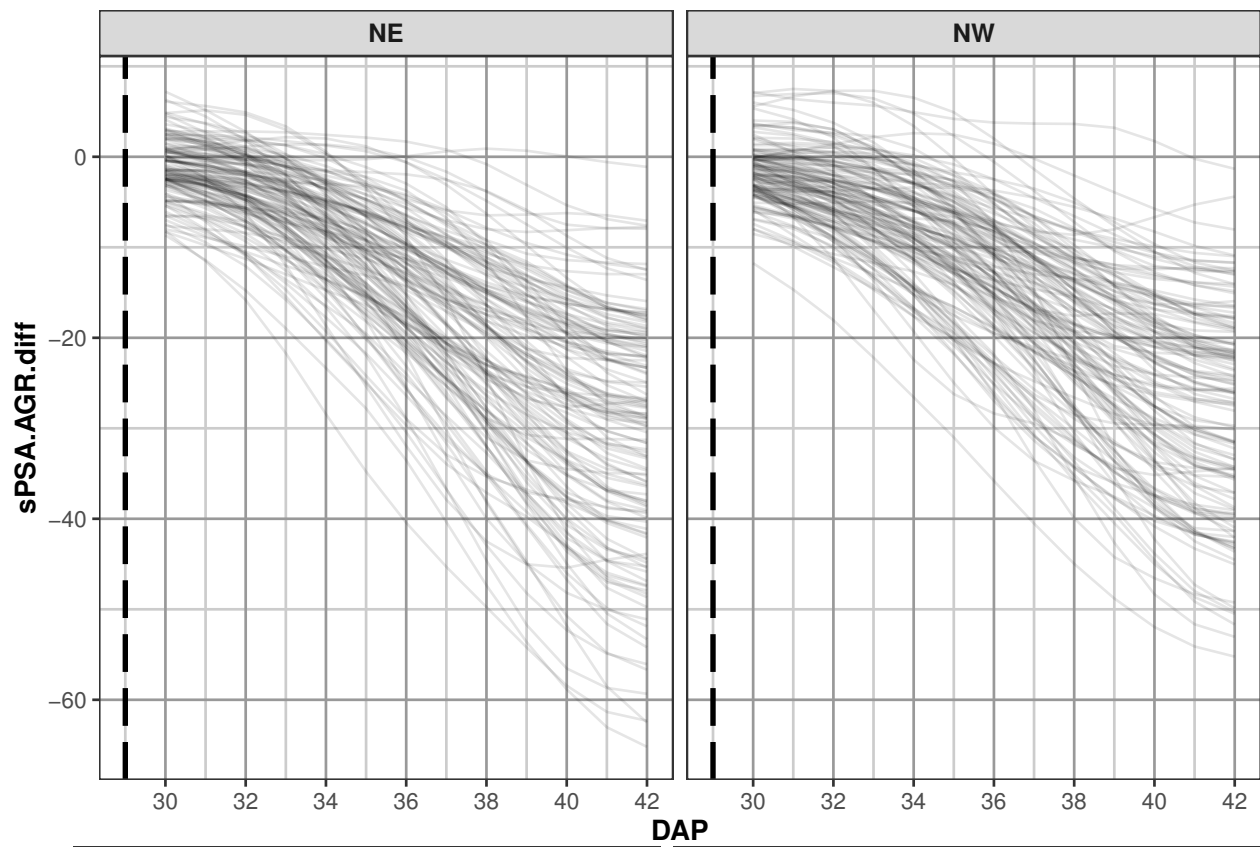
```

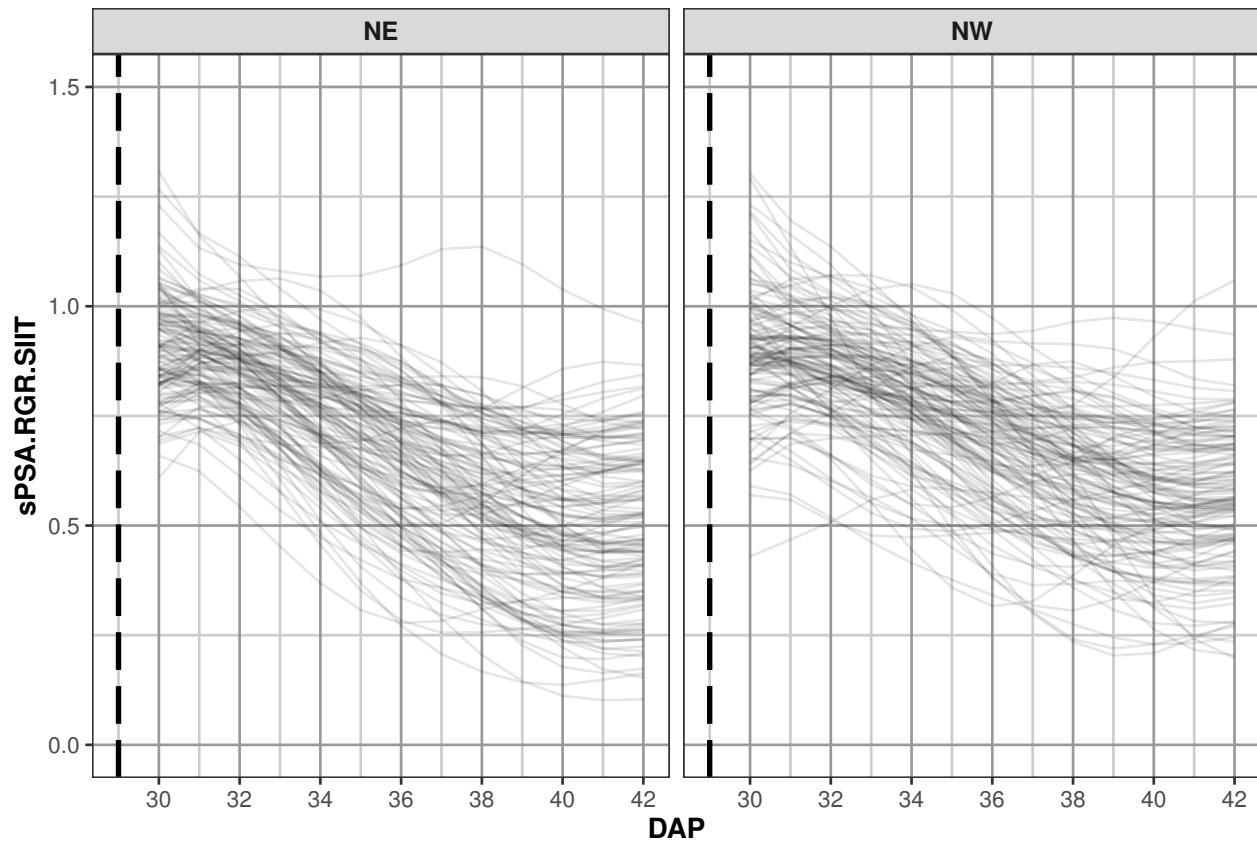
longi.SIIT.dat <-
  twoLevelOpcreate(data = longi.dat, responses = responses.GR, suffices.treatment=c("C","S"),
    operations = c("-", "/", "/"), suffices.results = suffices.results,
    columns.retained = cols.retained,
    by = c("Smarthouse","Zone","ZMainunit","DAP"))
longi.SIIT.dat <- with(longi.SIIT.dat,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

# 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=".", printPlot=FALSE, )
  plt <- plt + geom_vline(xintercept=29, linetype="longdash", size=1) +
    scale_x_continuous(breaks=seq(28, 42, by=2)) +
    scale_y_continuous(limits=limits[[k]])
  print(plt)
}

```





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="."))

cart.SIIT.dat <- twoLevel0pcreate(data = cart.dat, responses = responses.SIITinterval,
                                suffices.treatment=c("C","S"),
                                suffices.results="SIIT",
                                columns.suffixed="Snapshot.ID.Tag")

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

## sPSA.RGR.31to35.SIIT
## Min. :0.4240
## 1st Qu.:0.7240
## Median :0.8033
## Mean :0.7940
## 3rd Qu.:0.8720
## Max. :1.0789

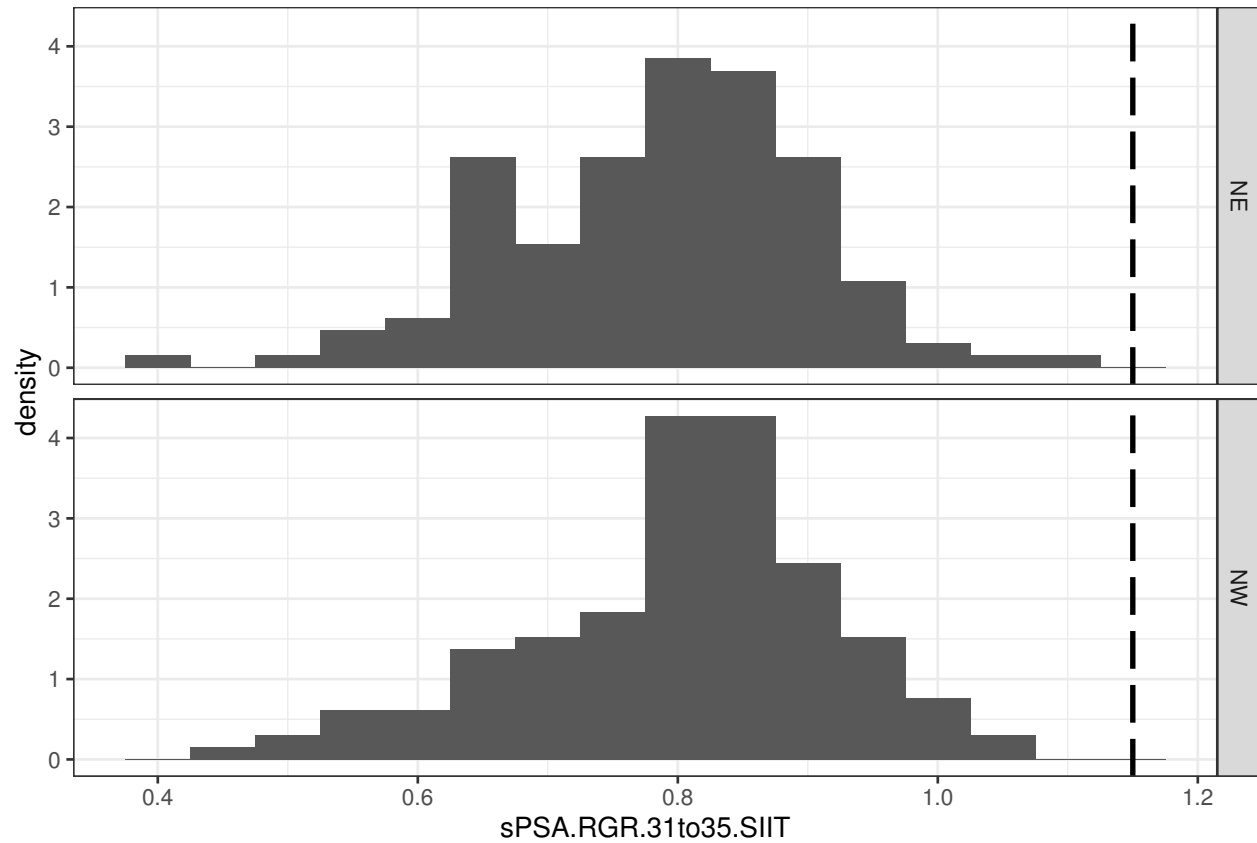
big.SIIT <- with(tmp, tmp[tmp[SIIT] > 1.15, c("Snapshot.ID.Tag.C", "Genotype.ID",
                                             paste(response, "C", sep="."),
                                             paste(response, "S", sep="."), SIIT)])

if (nrow(big.SIIT) > 1)
  big.SIIT <- big.SIIT[order(big.SIIT[SIIT]),]
```

```
print(big.SIIT)
```

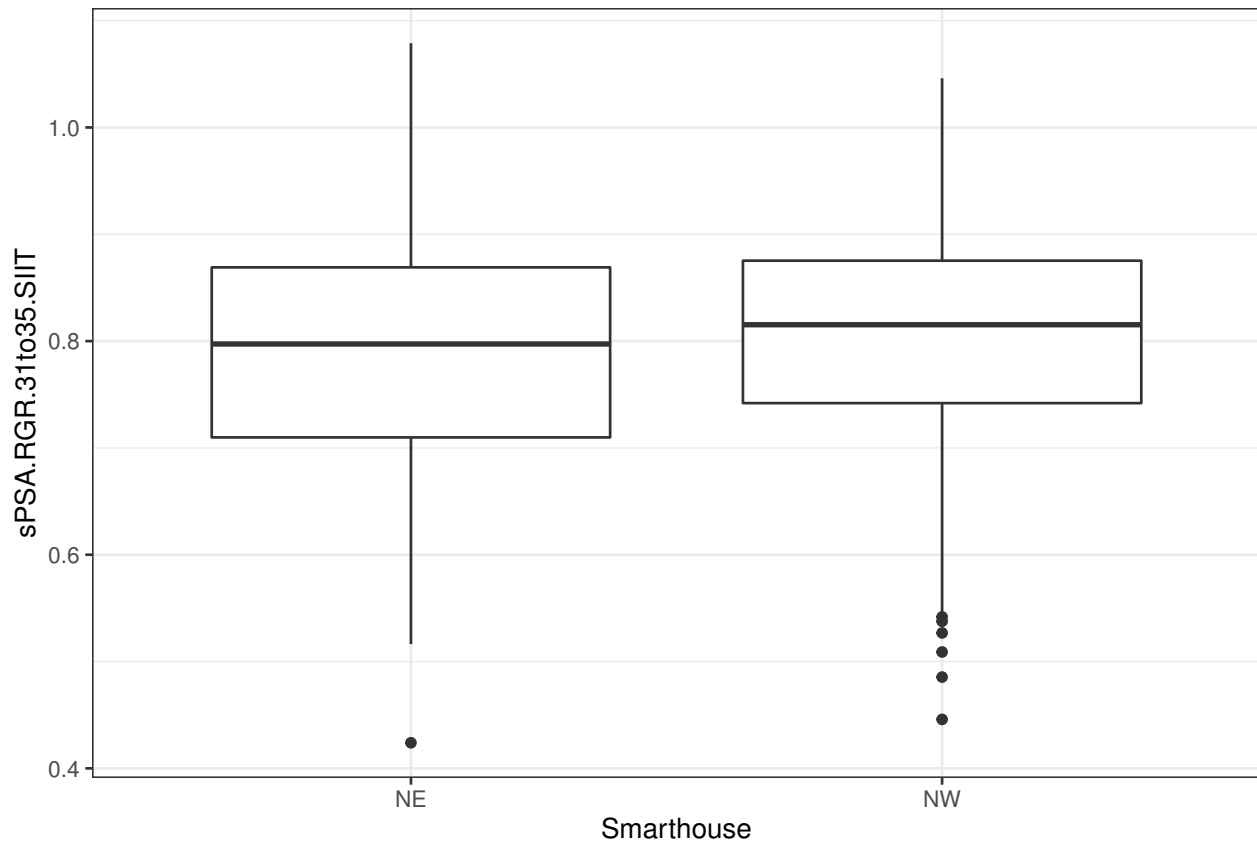
```
## [1] Snapshot.ID.Tag.C      Genotype.ID      sPSA.RGR.31to35.C
## [4] sPSA.RGR.31to35.S      sPSA.RGR.31to35.SIIT
## <0 rows> (or 0-length row.names)
```

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



```
plt <- ggplot(tmp, aes_string(x="Smarthouse", y=SIIT)) +
  geom_boxplot() + theme_bw()
print(plt)
```





```
remove(tmp)
```

## Save image

```
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., 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>.