

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

Chris Brien

17 March, 2020

This example is taken from Brien et al. (2020), who used it to illustrate the five steps of the method they describe for smoothing and extracting traits (SET). More details on the rationale for the steps used in this process are available in the Methods section of Brien et al. (2020).

## Initialize

### Set up characters for variable names and titles

```
responses <- c("Area", paste("Area", c("AGR", "RGR"), sep = "."))
responses.smooth <- c("Area.smooth", paste("Area.smooth", c("AGR", "RGR"), sep = "."))
responses.logis <- paste(responses.smooth, "Logistic", sep = ".")
resptitles <- c("PSA", "PSA AGR", "PSA RGR")
resptitles.smooth <- c("sPSA", "sPSA AGR", "sPSA RGR")
respunits <- c("(kpixels)", "(kpixels / day)", "( / day)")
y.titles <- c("PSA (kpixels)", "PSA AGR (kpixels / day)", "PSA RGR ( / day)")
names(y.titles) <- responses.smooth
ypred.titles <- paste0("Predicted s", y.titles)
names(ypred.titles) <- responses.smooth
pred.type <- c("Predicted", "Backtransformed predicted")
devn.titles <- c("PSA deviation (kpixels)", "PSA AGR deviation (kpixels / day)",
                "PSA RGR deviation ( / day)")
names(devn.titles) <- responses.smooth
x.title <- "DAP"
# Specify time intervals of homogeneous growth dynamics
DAP.cart <- c(18,22,27,33,39,43,51)
DAP.starts <- DAP.cart[-length(DAP.cart)]
DAP.ends <- DAP.cart[-1]
DAP.mids <- (DAP.starts + DAP.ends)/2
#Functions to label the plot facets
labelAMF <- as_labeller(function(lev) paste(lev, "AMF"))
labelZn <- as_labeller(function(lev) paste("Zn:", lev, "mg/kg"))
vline.water <- list(geom_vline(xintercept=39, linetype="longdash",
                             alpha = 0.5, size=0.6))
x.axis <- list(scale_x_continuous(breaks = seq(17, 51, by = 2)),
              theme(axis.text.x = element_text(angle = 90),
                    panel.grid.minor.x = element_blank()))
vline.DAP.intvl <- list(geom_vline(xintercept=DAP.starts[-1], linetype="longdash",
                                 alpha = 0.5, size=0.75))
theme.profile <- list(vline.DAP.intvl,x.axis)
```

## Step 1: Import, select and derive longitudinal data

In this step, the aim is to produce the data.frame `longit.dat` that contains the imaging variables, observed growth rates, covariates and factors. The growth rates are calculated from the observed data by differencing consecutive observations for a plant..

### Load the pre-prepared data

```
data(tomato.dat)
```

### Copy the data to preserve the original data.frame

```
longit.dat <- tomato.dat
```

### Add continuous growth rates for raw data

```
longit.dat <- splitContGRdiff(longit.dat, responses = responses[1],
                             INDICES="Snapshot.ID.Tag", which.rates = c("AGR", "RGR"),
                             times.factor="DAP")
```

## Step 2: Exploratory analysis

We begin by trying direct smoothing of the observed PSA with smoothing  $DF = 5$ . The growth rates are calculated from the smoothed data by difference, rather than from the spline derivatives. This matches the method used for the observed data.

### Add smoothed PSA for logarithmic smoothing with $DF = 6$ to a temporary file

```
SET.dat <- longit.dat
SET.dat <- splitSplines(SET.dat, response = responses[1], x = "xDAP",
                       INDICES = "Snapshot.ID.Tag",
                       smoothing.method = "log", df = 6)
```

### Add growth rates to SET.dat

```
SET.dat <- splitContGRdiff(SET.dat, responses = responses.smooth[1],
                          INDICES="Snapshot.ID.Tag", which.rates = c("AGR", "RGR"),
                          times.factor="DAP")
```

### Plot the PSA traits

```
for (k in 1:length(responses))
{
  plotLongitudinal(SET.dat, x = "xDAP+34", response = responses[k],
                   xname = "xDAP", x.title = "DAP", y.title = paste(resptitles[k], respunits[k]),
                   facet.x = ".", facet.y = ".", alpha = 0.4,
                   colour.column = "Zn", addMediansWhiskers = TRUE,
                   ggplotFuncs = c(list(facet_grid(. ~ AMF,
                                                labeller = labeller(AMF = labelAMF))),
                                   x.axis, vline.water))
  plotLongitudinal(SET.dat, x = "xDAP+34", response = responses.smooth[k],
```

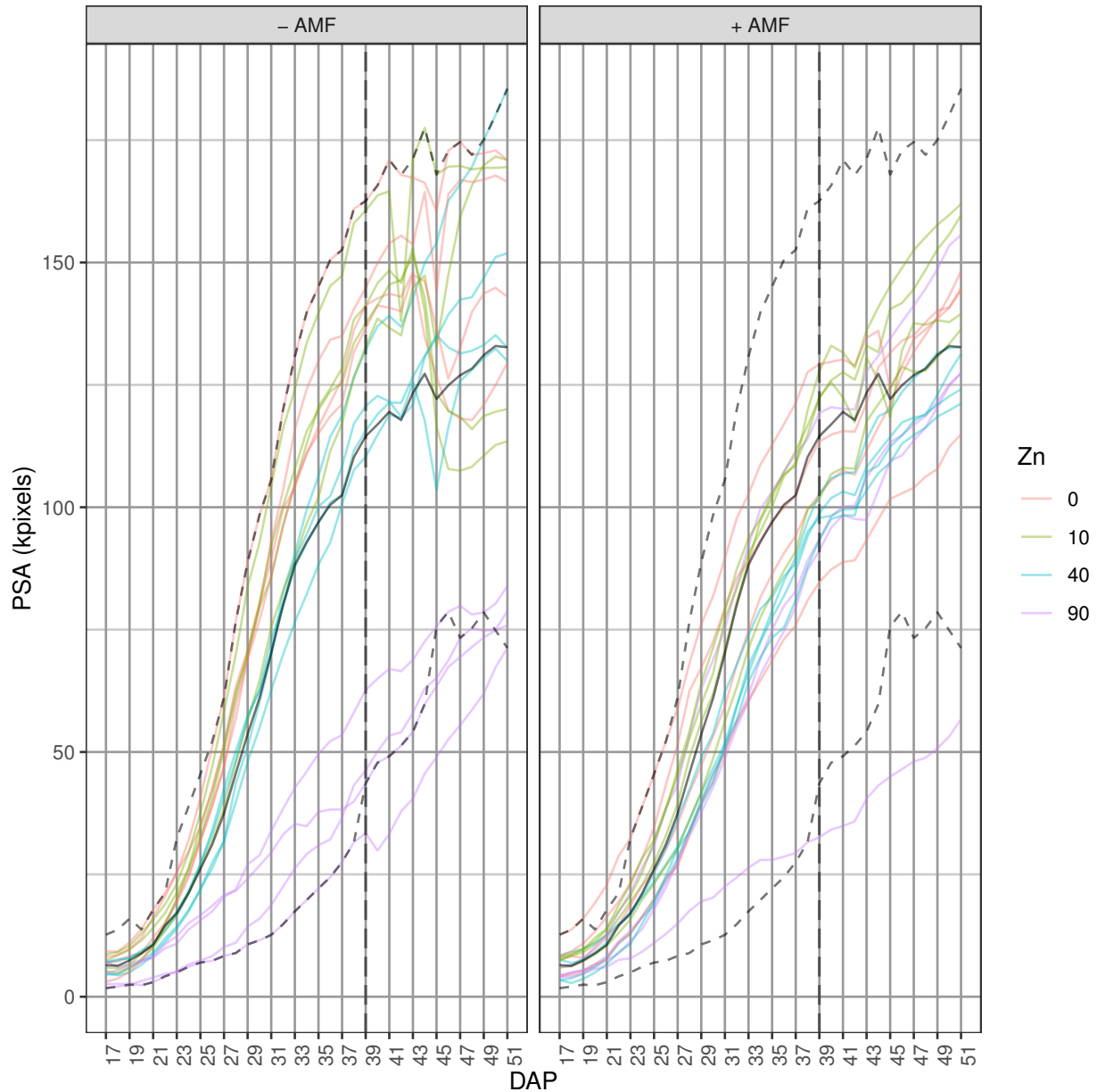
```

xname = "xDAP", x.title = "DAP",
y.title = paste(resptitles.smooth[k], respunits[k]),
facet.x = ".", facet.y = ".", alpha = 0.4,
colour.column = "Zn", addMediansWhiskers = TRUE,
ggplotFuncs = c(list(facet_grid(. ~ AMF,
                        labeller = labeller(AMF = labelAMF))),
                x.axis, vline.water))
}

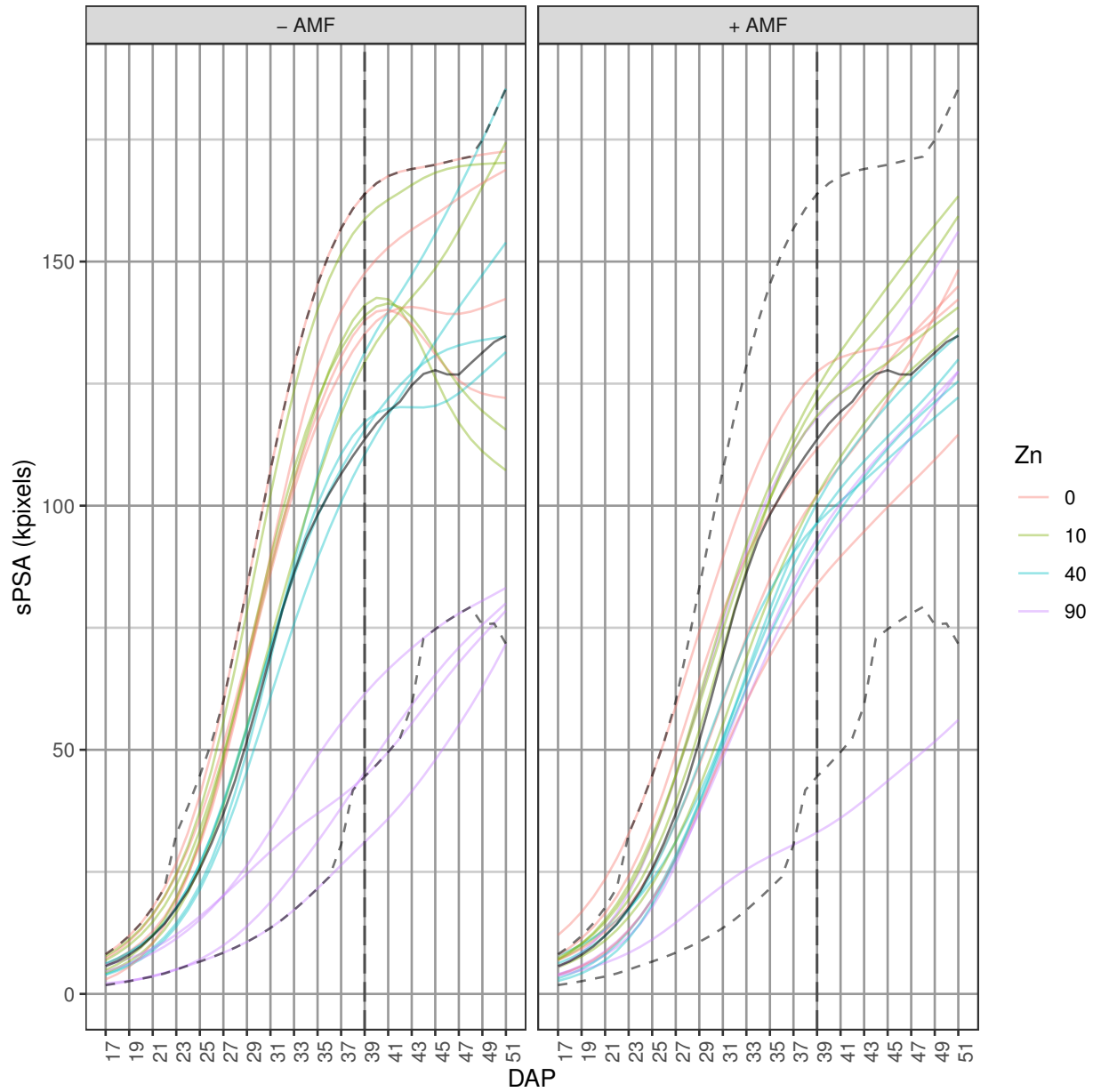
```

## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses[k], : x is xDAP+34 and xname is xDAP  
## Is xname the name of the column from which x is derived?

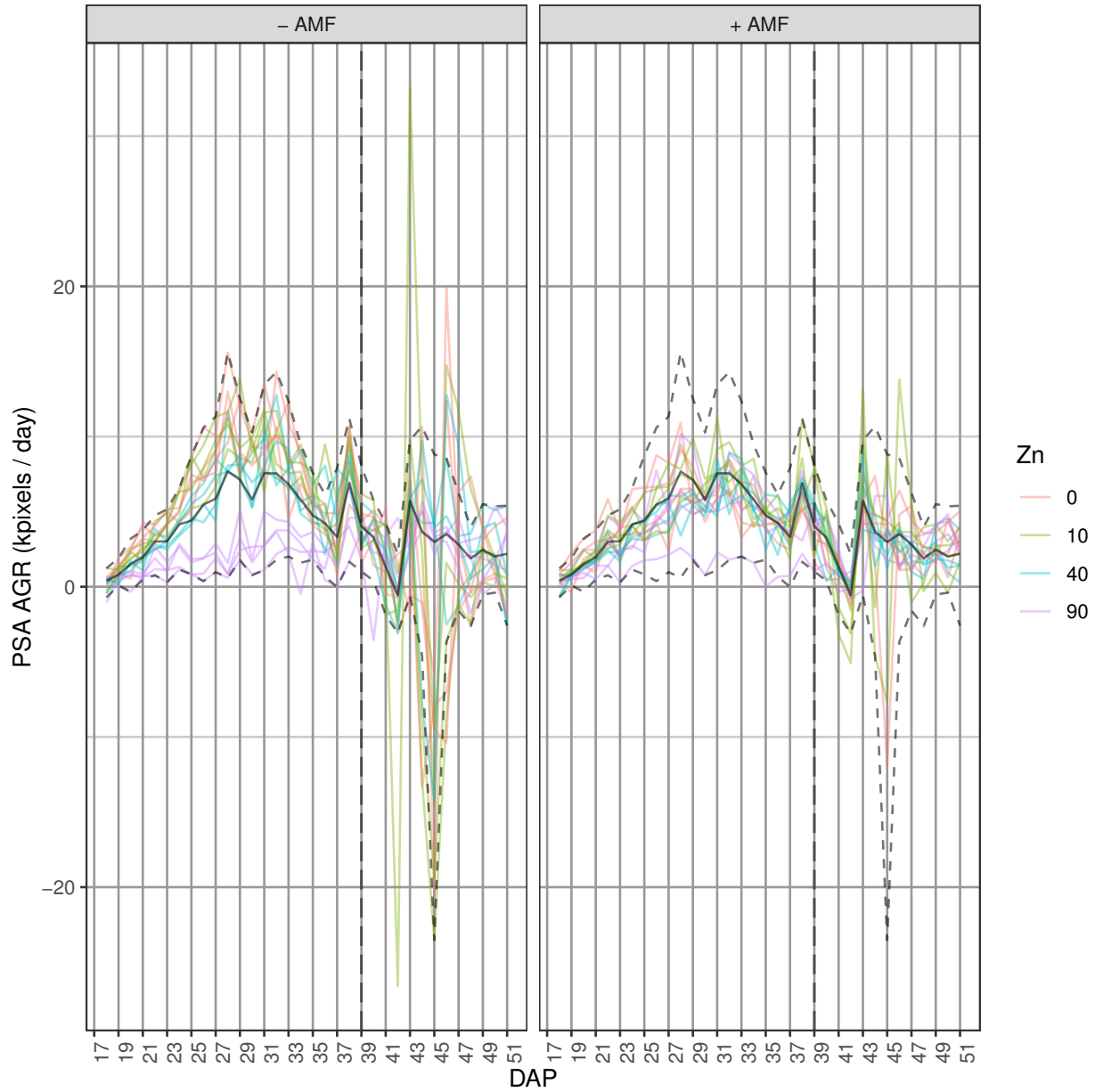
## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses.smooth[k], : x is xDAP+34 and xname is xDAP  
## Is xname the name of the column from which x is derived?



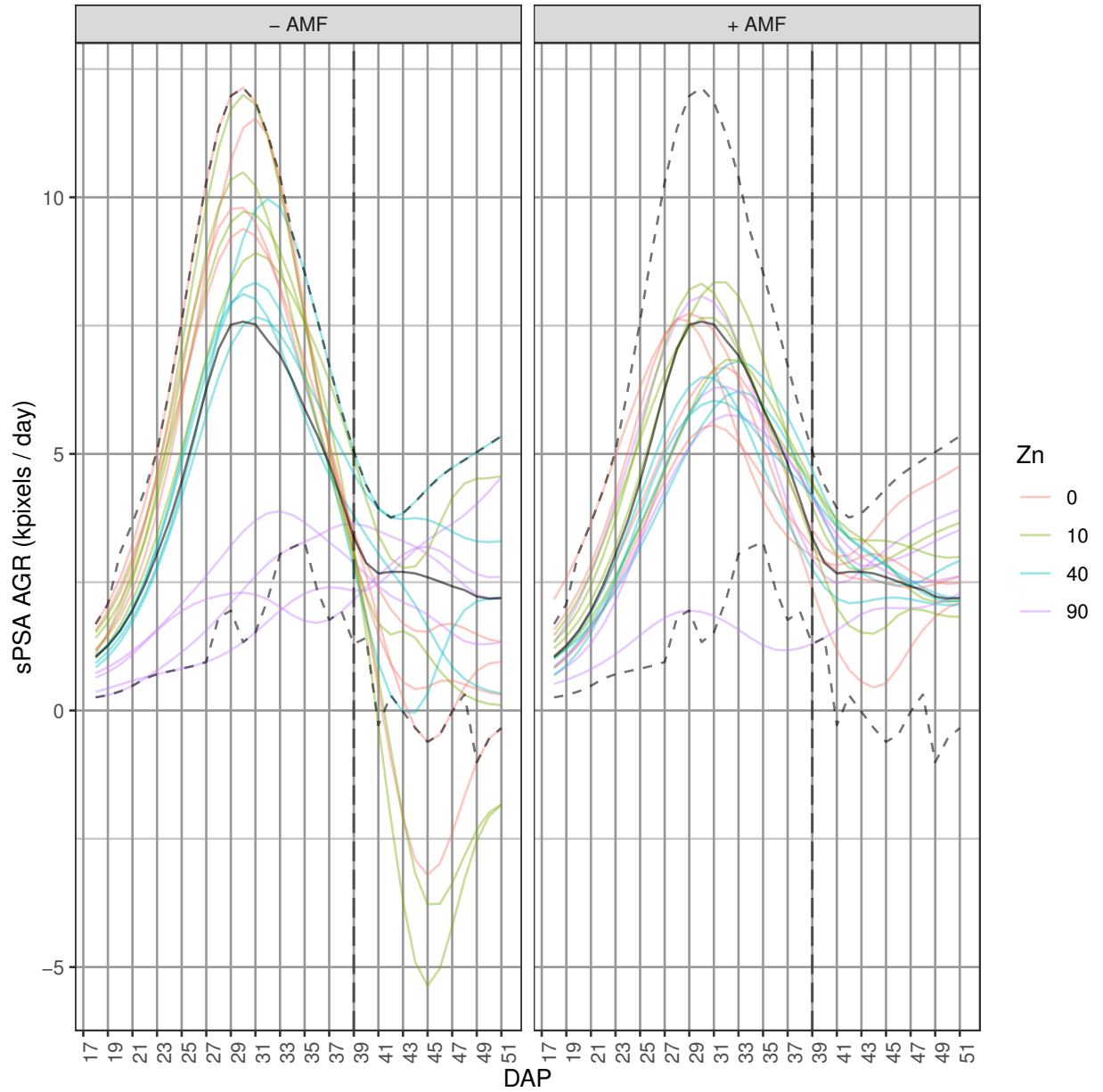
```
## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses[k], : x is xDAP+34 and xname
## Is xname the name of the column from which x is derived?
```



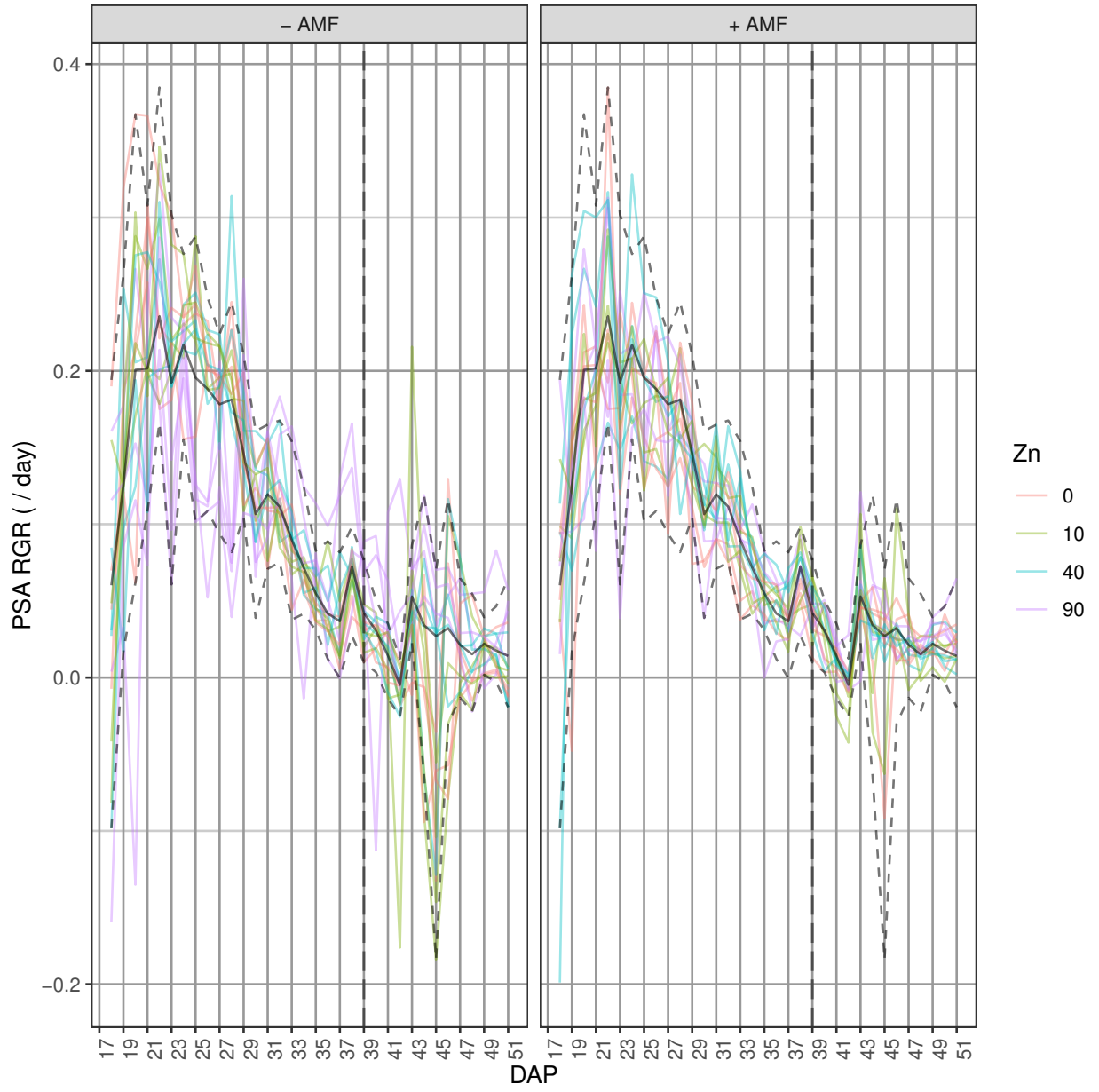
```
## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses.smooth[k], : x is xDAP+34 and
## Is xname the name of the column from which x is derived?
```

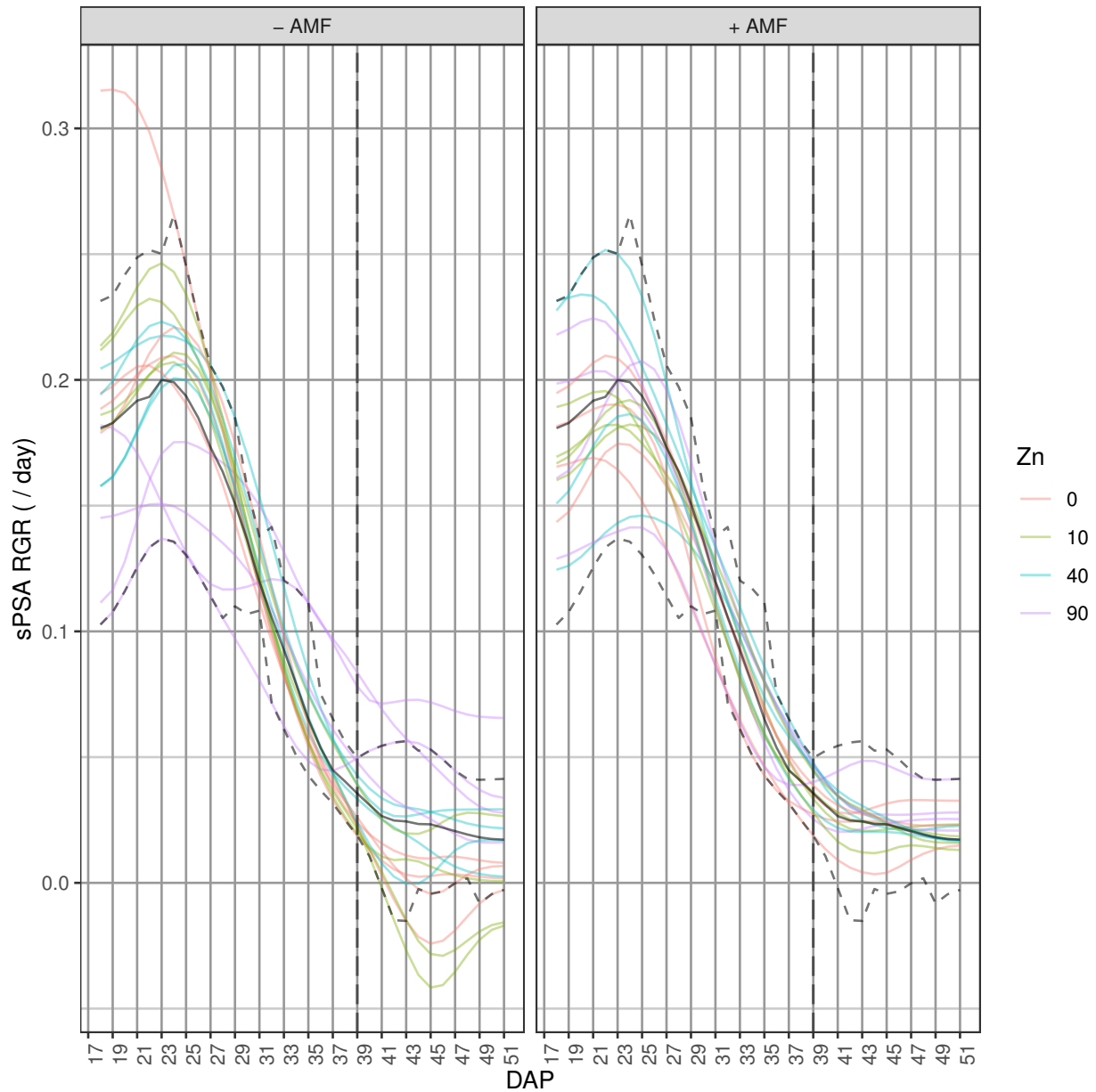


```
## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses[k], : x is xDAP+34 and xname
## Is xname the name of the column from which x is derived?
```



```
## Warning in plotLongitudinal(SET.dat, x = "xDAP+34", response = responses.smooth[k], : x is xDAP+34 and
## Is xname the name of the column from which x is derived?
```





### Step 3: Choose smoothing DF and method

Fit three-parameter logistic curves

Organize non-missing data into a grouped object

```
logist.sub <- na.omit(longit.dat)
logist.grp <- groupedData(Area ~ xDAP | Snapshot.ID.Tag,
                          data = logist.sub)
```



## Fit the logistics and obtain fitted values

```
logist.lis <- nlsList(SSlogis, logist.grp)
logist.sub$Area.smooth <- fitted(logist.lis)
```

## Calculate the growth rates from the logistic fits

```
logist.sub <- splitContGRdiff(logist.sub, responses = responses.smooth[1],
                             INDICES="Snapshot.ID.Tag", which.rates = c("AGR", "RGR"),
                             times.factor="DAP")
names(logist.sub)[match(responses.smooth, names(logist.sub))] <- responses.logis
```

## Probe the smoothing methods and DF

```
smooth.dat <- probeSmoothing(data = longit.dat, response = "Area",
                             xname = "xDAP", times.factor="DAP",
                             facet.x = ".", facet.y = ".",
                             smoothing.methods = c("dir", "log"),
                             df = c(4:6,12), x="xDAP", get.rates = TRUE,
                             which.plots = "none", deviations.plots = "none")
```

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

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

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

```
SET.dat <- merge(longit.dat, smooth.dat)
SET.dat <- merge(SET.dat, logist.sub, all.x = TRUE)
```

## Plot the median deviations plots

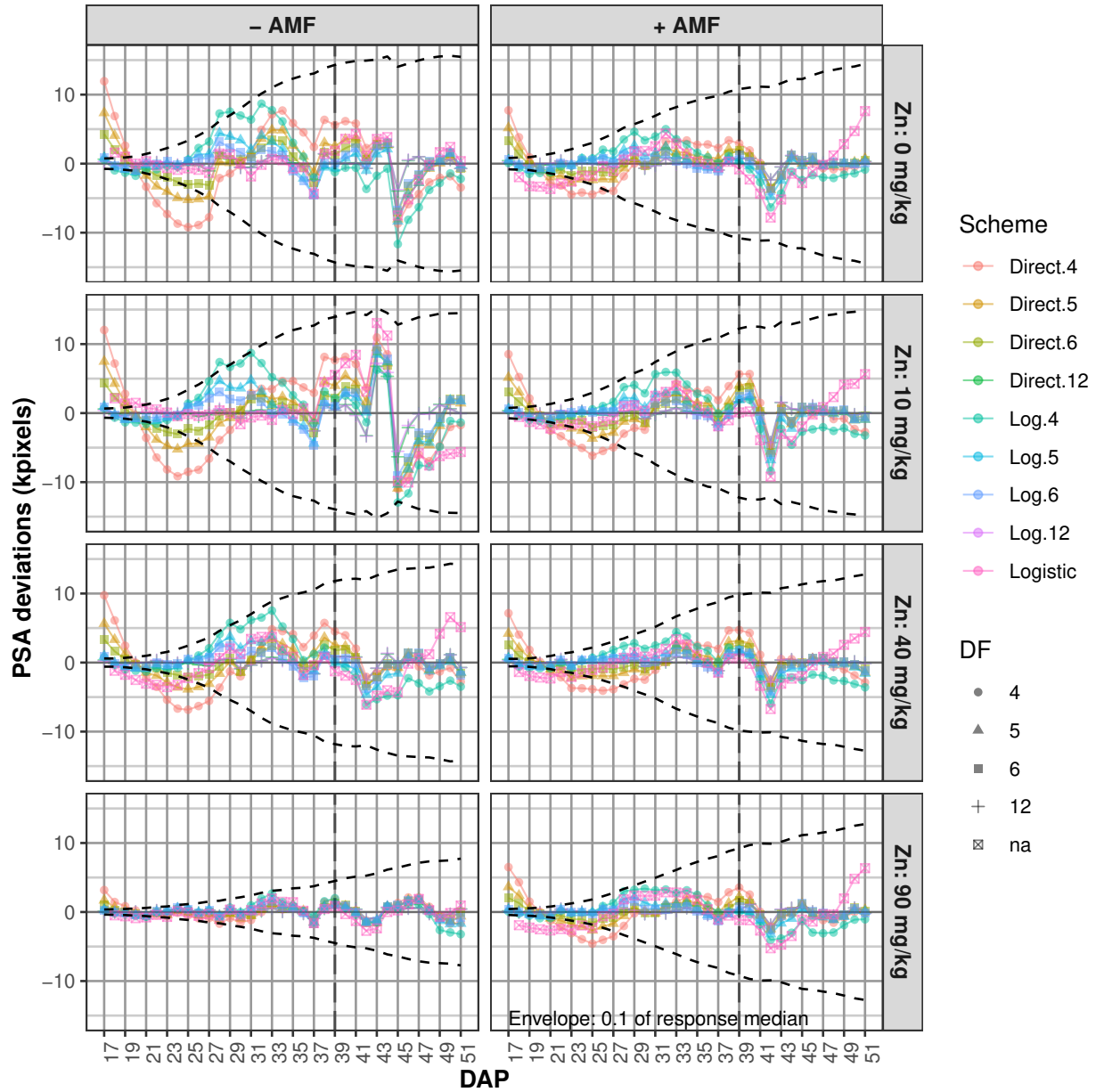
```
propn.types = c(0.1, 0.5, 0.75)
devn.scales <- list()
devn.scales$Area.smooth <- scale_y_continuous(#limits = c(-15,15),
                                             breaks = seq(-10,10,by=10))
devn.scales$Area.smooth.AGR <- scale_y_continuous(limits = c(-10,10),
                                                  breaks = seq(-10,10,by=5))
devn.scales$Area.smooth.RGR <- scale_y_continuous(limits = c(-0.75,0.25),
                                                  breaks = round(seq(-0.75,0.25,by=0.25), 2))

for (k in 1:length(responses))
  plotMedianDeviations(data = SET.dat,
                       response =responses[k],
                       response.smoothed = responses.smooth[k],
                       extra.smooths = "Logistic",
                       x = "xDAP+34", xname = "xDAP", x.title = x.title,
                       smoothing.methods = c("dir", "log"), df = c(4:6,12) ,
                       y.titles = paste(resptitles[k], "deviations", respunits[k]),
                       facet.x = "AMF", facet.y = "Zn",
                       trait.types = "response",
                       propn.types = propn.types[k],
                       labeller = labeller(Zn = labelZn,
                                           AMF = labelAMF),
```

```
ggplotFuncsMedDevn = c(devn.scales[k], vline.water, x.axis))
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```

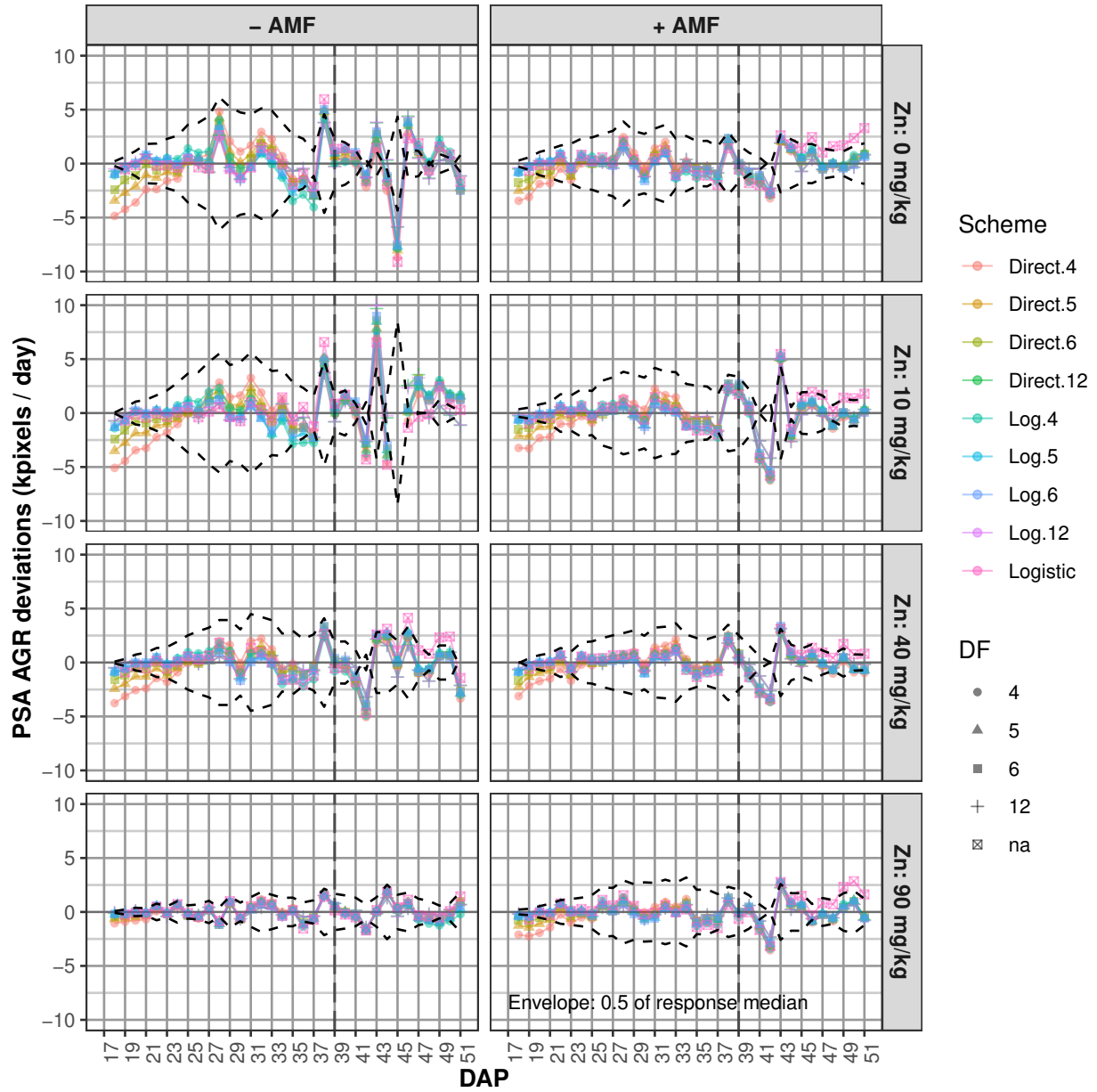
```
## Warning: Removed 8 rows containing missing values (geom_point).
```



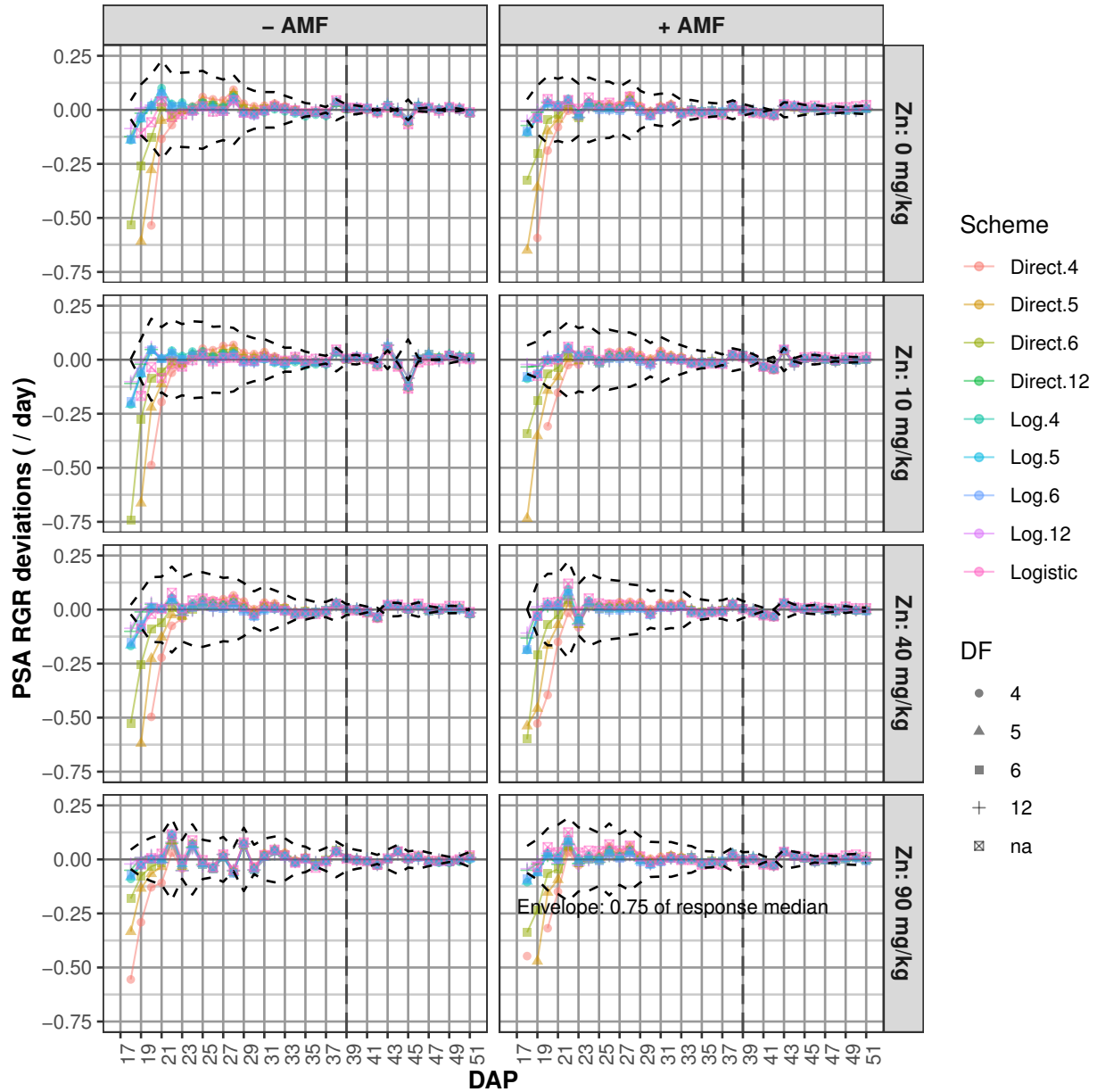
```
## Warning: Removed 10 rows containing missing values (geom_path).
```

```
## Warning: Removed 89 rows containing missing values (geom_point).
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```



```
## Warning: Removed 13 rows containing missing values (geom_path).
## Warning: Removed 95 rows containing missing values (geom_point).
## Warning: Removed 2 rows containing missing values (geom_path).
```



Plot the profile plots comparing direct and log smoothing for  $DF = 6$

```
probeSmoothing(data = SET.dat, response = responses[1],
  x = "xDAP+34", xname = "xDAP", x.title = x.title,
  times.factor="DAP",
  facet.x = ".", facet.y = "AMF",
  smoothing.methods = c("dir", "log"),
  df = 6, get.rates = TRUE,
  colour.column = "Zn", alpha = 0.5,
  which.plots = "methods+rawcompare", addMediansWhiskers = TRUE,
  deviations.plots = "none",
  labeller = labeller(Zn = labelZn,
    AMF = labelAMF),
```

```
ggplotFuncs = theme.profile)
```

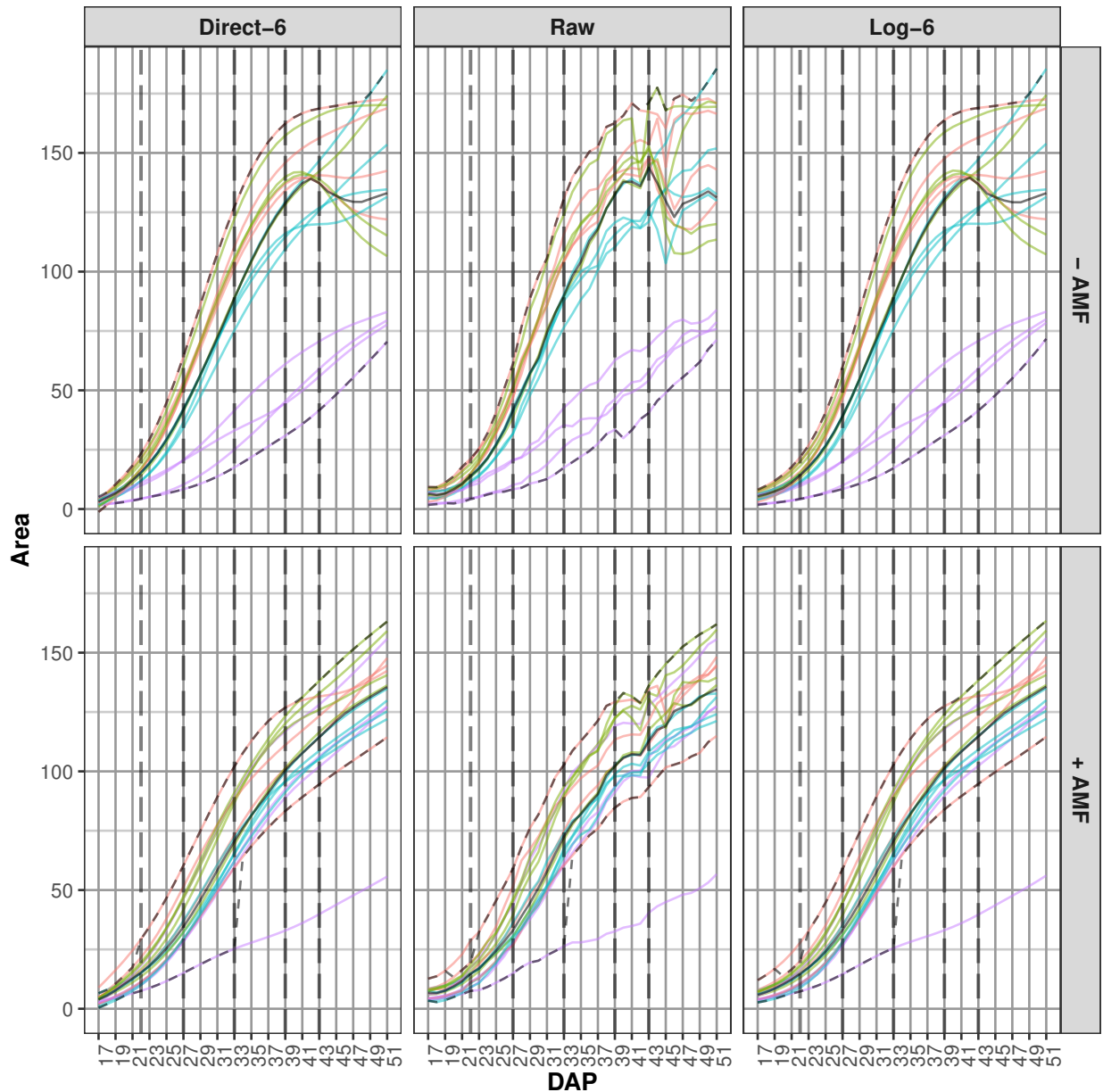
```
## Warning in log(PGR(x, time.diffs, lag = lag)): NaNs produced
```

```
## Warning in plotLongitudinal(data = tmp.sm, x = x, xname = xname, response = response, : x is xDAP+34
```

```
## Is xname the name of the column from which x is derived?
```

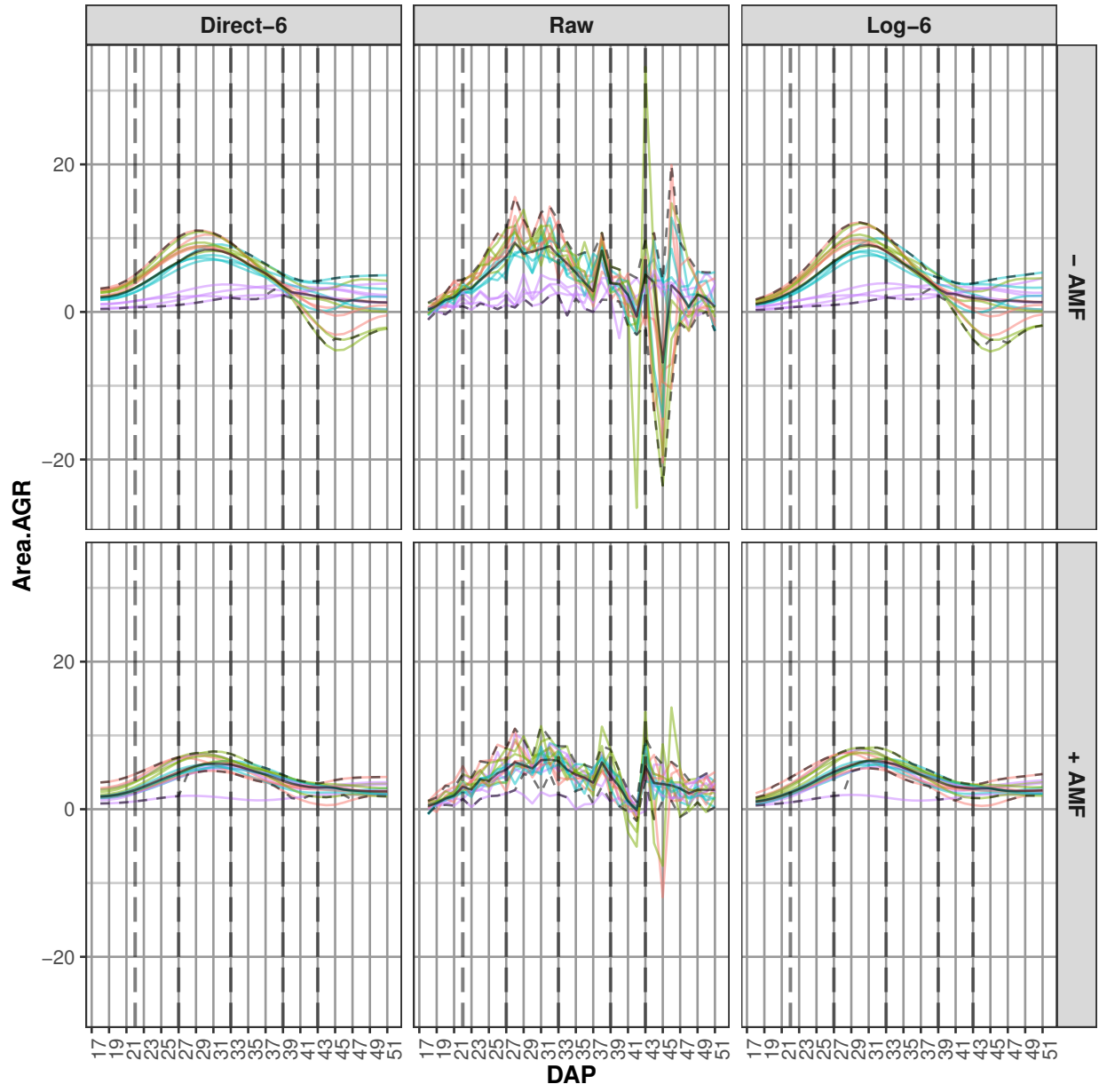
```
## Warning in plotLongitudinal(data = tmp.sm, x = x, xname = xname, response = response, : x is xDAP+34
```

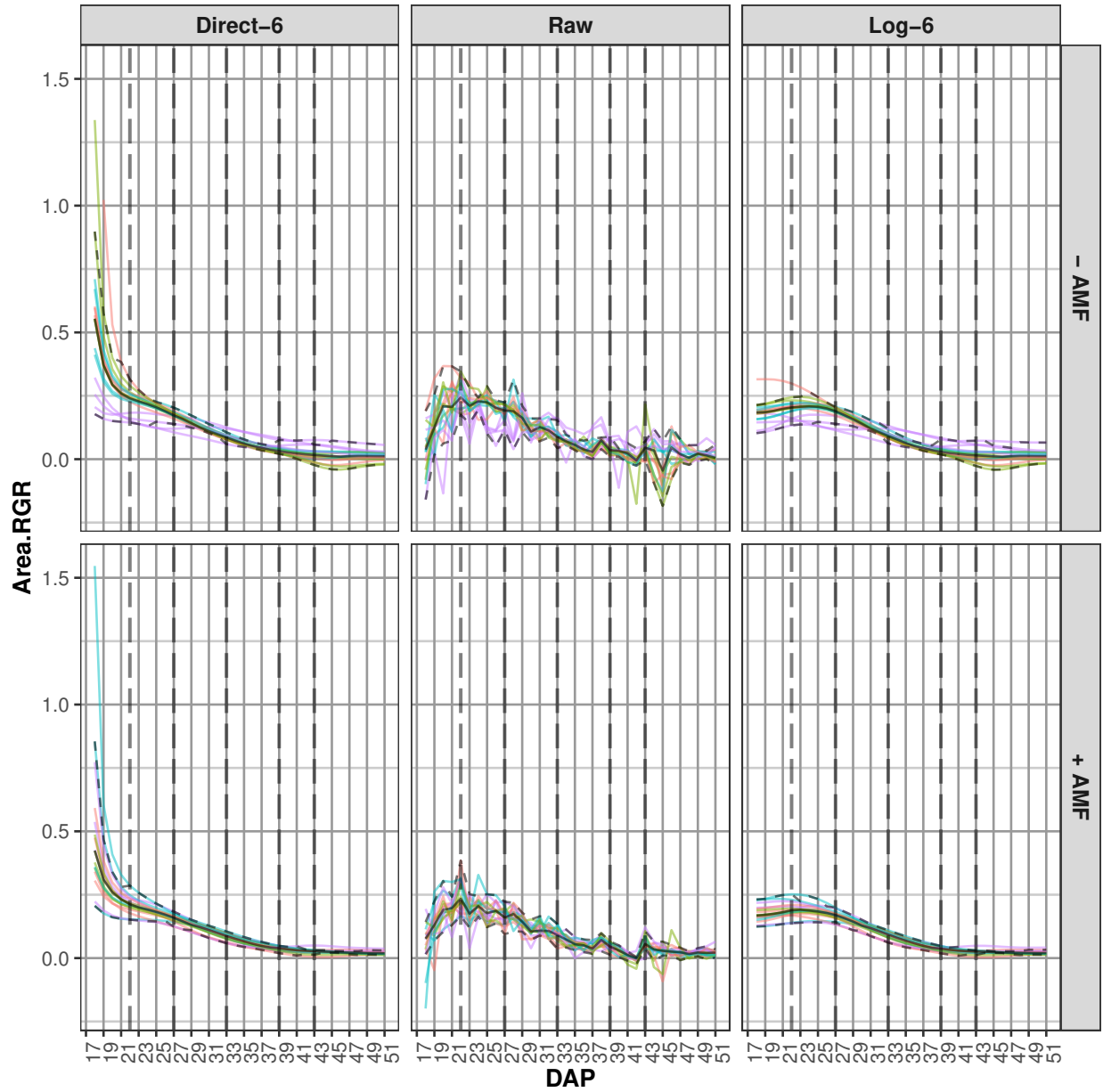
```
## Is xname the name of the column from which x is derived?
```



```
## Warning in plotLongitudinal(data = tmp.sm, x = x, xname = xname, response = response, : x is xDAP+34
```

```
## Is xname the name of the column from which x is derived?
```





## Step 4: Identify potential outliers and remove if justified

A plant was identified as slow growing. Even though its pot had been inoculated with AMF, it had low AMF root colonization and a random mutated shoot phenotype, which could explain why its behaviour was consistent with a plant that was not inoculated with AMF. We omit the it from further analysis.

### Omit responses for the outlier plant

```
omit <- with(longit.dat, Zn==90 & AMF=="+" & Block ==4)
responses.all <- names(longit.dat)[match("Weight.After", names(longit.dat)):length(longit.dat)]
longit.dat[omit, responses.all] <- NA
```

## Step 3 (cont'd): Choose smoothing DF and method

Add smoothed PSA for log-smoothing with  $DF = 6$

```
longit.dat <- splitSplines(longit.dat, response = responses[1], x = "xDAP",
                          INDICES = "Snapshot.ID.Tag",
                          smoothing.method = "log", df = 6)
```

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

Add growth rates to longit.dat

```
longit.dat <- splitContGRdiff(longit.dat, responses = responses.smooth[1],
                              INDICES="Snapshot.ID.Tag", which.rates = c("AGR", "RGR"),
                              times.factor="DAP")
```

## Step 5: Per-cart trait formulation and extraction

In this step, traits that have a single-value for each sharp are created.

The time intervals of homogeneous growth dynamics (set in global.r)

```
(suffices <- paste(DAP.starts, DAP.ends, sep="to"))
```

```
## [1] "18to22" "22to27" "27to33" "33to39" "39to43" "43to51"
```

Commence cart.dat

```
nidcols <- match("Weight.After", names(longit.dat))-1
idcols.cart <- names(longit.dat)[1:nidcols]
cart.dat <- longit.dat[longit.dat$DAP == DAP.cart[1], idcols.cart]
cart.dat <- cart.dat[do.call(order, cart.dat), ]
```

Populate cart.dat at prescribed time-points

```
for (day in DAP.cart)
  cart.dat <- cbind(cart.dat,
                   getTimesSubset(responses.smooth, data = longit.dat,
                                   times.factor = "DAP", which.times = day,
                                   suffix = as.character(day)))
```

Populate cart.dat and cart.all.intervals at prescribed intervals

```
for (r in responses.smooth[1])
{
  for (k in 1:length(suffices))
  {
    cart.dat <- merge(cart.dat,
                     intervalGRdiff(r,
                                     which.rates = c("AGR", "RGR"),
                                     times.factor = "DAP",
```



```

        start.times = DAP.starts[k],
        end.times = DAP.ends[k],
        suffix.interval = suffices[k],
        data = longit.dat),
    by = "Snapshot.ID.Tag")
}
}

```

## Finalise

```

cart.dat <- with(cart.dat, cart.dat[order(Snapshot.ID.Tag), ])
head(cart.dat)

```

```

## Snapshot.ID.Tag DAP Lane Position xDAP DAP.diffs xPosn Block Cart AMF Zn
## 1 061472 18 6 5 -16 1 -8.5 1 1 - 0
## 2 061473 18 6 6 -16 1 -7.5 1 2 + 10
## 3 061474 18 6 7 -16 1 -6.5 1 3 - 90
## 4 061475 18 6 8 -16 1 -5.5 1 4 + 40
## 5 061476 18 6 9 -16 1 -4.5 1 5 + 90
## 6 061477 18 6 10 -16 1 -3.5 1 6 - 40
## Treatments Area.smooth.18 Area.smooth.AGR.18 Area.smooth.RGR.18
## 1 -,0 9.807324 1.6845547 0.1884583
## 2 +,10 8.162253 1.2076831 0.1601212
## 3 -,90 2.420749 0.2552224 0.1114134
## 4 +,40 8.903409 1.0421415 0.1244864
## 5 +,90 4.741824 0.7045944 0.1608632
## 6 -,40 4.983090 0.9214957 0.2044746
## Area.smooth.22 Area.smooth.AGR.22 Area.smooth.RGR.22 Area.smooth.27
## 1 21.757197 4.0535127 0.2061718 59.936608
## 2 16.096746 2.6186640 0.1775523 39.409717
## 3 4.191419 0.6203235 0.1601669 9.954592
## 4 15.116144 1.9650536 0.1392586 31.134433
## 5 9.600854 1.6661982 0.1906121 26.442502
## 6 11.630394 2.2636744 0.2164589 33.821095
## Area.smooth.AGR.27 Area.smooth.RGR.27 Area.smooth.33 Area.smooth.AGR.33
## 1 10.292807 0.1884140 128.71939 10.245261
## 2 6.243039 0.1724667 86.91303 8.071984
## 3 1.560483 0.1705038 24.72807 3.047349
## 4 4.142143 0.1427631 64.69042 6.223293
## 5 4.704429 0.1959074 62.49983 6.215004
## 6 6.288936 0.2057300 88.28055 9.794637
## Area.smooth.RGR.33 Area.smooth.39 Area.smooth.AGR.39 Area.smooth.RGR.39
## 1 0.08294015 163.83401 3.038382 0.01871961
## 2 0.09747422 121.36293 3.981731 0.03335872
## 3 0.13151559 45.37888 3.643014 0.08368594
## 4 0.10114846 96.64198 4.233163 0.04479083
## 5 0.10473886 93.25406 4.156842 0.04559948
## 6 0.11760066 131.36934 5.048666 0.03918904
## Area.smooth.43 Area.smooth.AGR.43 Area.smooth.RGR.43 Area.smooth.51
## 1 168.95836 0.5494274 0.00325715 172.57053
## 2 133.31548 2.7296263 0.02068746 159.31254
## 3 59.16752 3.3001120 0.05739157 80.01691

```

## 4	109.11412	2.6781128	0.02485037	129.95768
## 5	106.62596	2.9949757	0.02849065	127.50507
## 6	147.30562	3.8417482	0.02642624	185.43136
##	Area.smooth.AGR.51	Area.smooth.RGR.51	Area.smooth.AGR.18to22	
## 1	0.3070212	0.00178069	2.9874681	
## 2	3.6586227	0.02323287	1.9836231	
## 3	2.1967173	0.02783704	0.4426675	
## 4	2.9164823	0.02269743	1.5531839	
## 5	2.6213027	0.02077269	1.2147575	
## 6	5.3449009	0.02924772	1.6618259	
##	Area.smooth.RGR.18to22	Area.smooth.AGR.22to27	Area.smooth.RGR.22to27	
## 1	0.1992038	7.635882	0.2026686	
## 2	0.1697742	4.662594	0.1790791	
## 3	0.1372406	1.152635	0.1729989	
## 4	0.1323323	3.203658	0.1445102	
## 5	0.1763576	3.368330	0.2026241	
## 6	0.2118929	4.438140	0.2134926	
##	Area.smooth.AGR.27to33	Area.smooth.RGR.27to33	Area.smooth.AGR.33to39	
## 1	11.463797	0.1273912	5.852438	
## 2	7.917219	0.1318159	5.741651	
## 3	2.462247	0.1516509	3.441801	
## 4	5.592665	0.1218831	5.325260	
## 5	6.009554	0.1433652	5.125705	
## 6	9.076575	0.1599058	7.181466	
##	Area.smooth.RGR.33to39	Area.smooth.AGR.39to43	Area.smooth.RGR.39to43	
## 1	0.04020318	1.281087	0.007699625	
## 2	0.05564626	2.988136	0.023483206	
## 3	0.10118461	3.447160	0.066331488	
## 4	0.06690002	3.118033	0.030345249	
## 5	0.06669396	3.342975	0.033499857	
## 6	0.06624883	3.984070	0.028624182	
##	Area.smooth.AGR.43to51	Area.smooth.RGR.43to51		
## 1	0.4515206	0.002644212		
## 2	3.2496329	0.022268701		
## 3	2.6061736	0.037733154		
## 4	2.6054456	0.021851825		
## 5	2.6098887	0.022353640		
## 6	4.7657177	0.028771915		

## Save the cart.data and the workspace

```
save(cart.dat, file="../data/cart.dat.rda")
load(file="../data/cart.dat.rda")
save.image("Tomato.RData")
```

## Reference

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>.