UCPH Data Analysis Timepoints

Elise

2024-06-09

Set the right working directory.

```
rm(list = ls())

setwd("C:/Users/elise/Documents/Mémoire/Main/Data/Templates/UCPH")
platform <- "UCPH"
```

Data importation

Reimport the data sets extracted from the Data Preparation and Data Analysis R Markdown.

```
list.files()
```

```
## [1] "endpoint.txt"     "plant_info.txt"   "S_timeseries.txt" "T_timeseries.txt"
## [5] "timeseries.txt"
```

```
plant_info <- read.table("plant_info.txt", header = TRUE, sep = "\t")
endpoint <- read.table("endpoint.txt", header = TRUE, sep = "\t")
S_timeseries <- read.table("S_timeseries.txt", header = TRUE, sep = "\t")

# plant_info
plant_info <- lapply(plant_info, factor)

# endpoint
matching_cols <- intersect(names(endpoint), names(plant_info))
endpoint[, matching_cols] <- lapply(endpoint[, matching_cols], factor)
endpoint$Date <- date(endpoint$Date)
endpoint$Timestamp <- NA

# timeseries
# No data for UCPH

# S_timeseries
matching_cols <- intersect(names(S_timeseries), names(plant_info))
S_timeseries[, matching_cols] <- lapply(S_timeseries[, matching_cols], factor)
S_timeseries$Timestamp <- as.POSIXct(S_timeseries$Timestamp, format = "%Y-%m-%d %H:%M:%
S")
S_timeseries$Date <- date(S_timeseries$Date)

# T_timeseries
# No data

platform <- "UCPH"

# endpoint
df <- endpoint[,colSums(is.na(endpoint))<nrow(endpoint)]
genotype_index <- which(colnames(df) == "Genotype")
variables <- colnames(df[, c(3:(genotype_index - 1))]) # We remove the two first column
s that are "Unit.ID" and "Date"

# timeseries
# no data

# S_timeseries
df_S_timeseries <- S_timeseries[,colSums(is.na(S_timeseries))<nrow(S_timeseries)]
genotype_index <- which(colnames(df_S_timeseries) == "Genotype")
variables_S <- "S_Height_cm"

# T_timeseries
# no data

print(paste(platform, ": The variables for endpoint are", paste(variables, collapse =
", "), sep = " "))
```

```
## [1] "UCPH : The variables for endpoint are DW_shoot_g, DW_root_g"
```

```
print(paste(platform, ": The variables for S_timeseries are", paste(variables_S, collap
se = ", "), sep = " "))
```

```
## [1] "UCPH : The variables for S_timeseries are S_Height_cm"
```

```
endpoint$Plant_type <- substr(endpoint$Genotype, nchar(as.character(endpoint$Genotyp
e)), nchar(as.character(endpoint$Genotype)))
S_timeseries$Plant_type <- substr(S_timeseries$Genotype, nchar(as.character(S_timeserie
s$Genotype)), nchar(as.character(S_timeseries$Genotype)))
```

Get the cleaned endpoint data

```
endpoint_clean <- endpoint
# Run the function on the dataset for all the variables
endpoint_clean <- detect_replace_outliers_by_genotype(endpoint_clean)



# We add a Plant_type variable that is H or L, with T being L
S_timeseries$Plant_type <- substr(S_timeseries$Genotype, nchar(as.character(S_timeserie
s$Genotype)), nchar(as.character(S_timeseries$Genotype)))

S_timeseries$Plant_type <- ifelse(S_timeseries$Plant_type %in% c("T", "L"), "Line",
                                  ifelse(S_timeseries$Plant_type == "H", "Hybrid", S_ti
meseries$Plant_type))
```

Time point objects

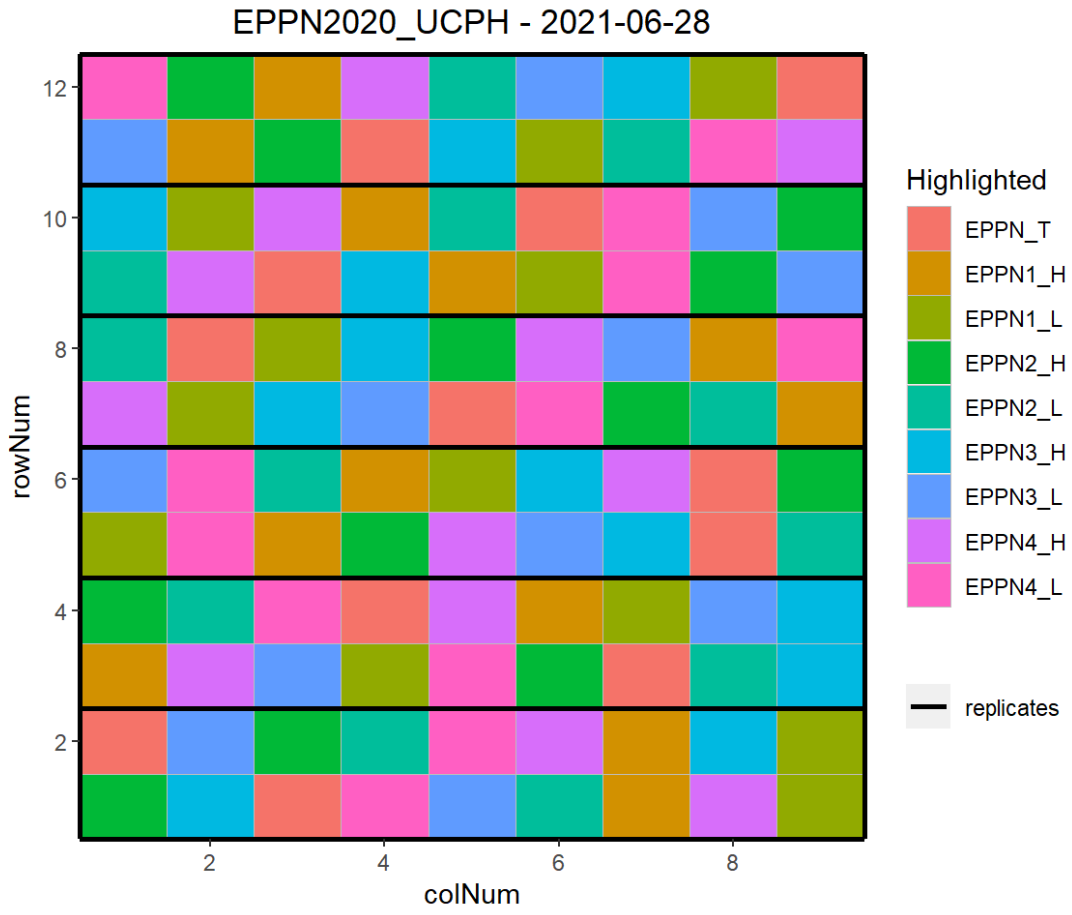
Generation of the timePoints objects using the function “createTimePoints”.

```
timePoint_endpoint <- createTimePoints(dat = endpoint,
                                       experimentName = "EPPN2020_UCPH",
                                       genotype = "Genotype",
                                       timePoint = "Date",
                                       plotId = "Unit.ID",
                                       rowNum = "Row",
                                       colNum = "Column",
                                       repId = "Replication")

timePoint_endpoint_clean <- createTimePoints(dat = endpoint_clean,
                                       experimentName = "EPPN2020_UCPH",
                                       genotype = "Genotype",
                                       timePoint = "Date",
                                       plotId = "Unit.ID",
                                       rowNum = "Row",
                                       colNum = "Column",
                                       repId = "Replication")
S_timeseries <- S_timeseries %>%
  group_by(Date, Unit.ID) %>%
  slice(1) %>%  # Keep the first line of every duplicate
  ungroup()

timePoint_S <- createTimePoints(dat = S_timeseries,
                                experimentName = "EPPN2020_UCPH",
                                genotype = "Genotype",
                                timePoint = "Date",
                                plotId = "Unit.ID",
                                rowNum = "Row",
                                colNum = "Column",
                                repId = "Replication",
                                addCheck = TRUE,
                                checkGenotypes = "EPPN_T")
```

Gentoypic layout

Check the layout of the platforms' genotypes.

```
genotypes_list <- as.character(unique(endpoint$Genotype))

plot(timePoint_endpoint,
     plotType = "layout",
     highlight =  genotypes_list,
     showGeno = FALSE)
```



1. endpoint

Comparisons between raw and cleaned data

View timePoint object.

```
summary(timePoint_endpoint)
```

```
## timePoint_endpoint contains data for experiment EPPN2020_UCPH.
##
## It contains 1 time points.
## First time point: 2021-06-28
## Last time point: 2021-06-28
##
## No check genotypes are defined.
```

```
getTimePoints(timePoint_endpoint)
```

```
##   timeNumber  timePoint
## 1          1 2021-06-28
```

Count the number of observations per trait.

```
traits <- variables

for (trait_name in traits) {
  print(paste("How many data observations for", trait_name))
  num_observations <- countValid(timePoint_endpoint, trait_name)
  print(num_observations)
}
```

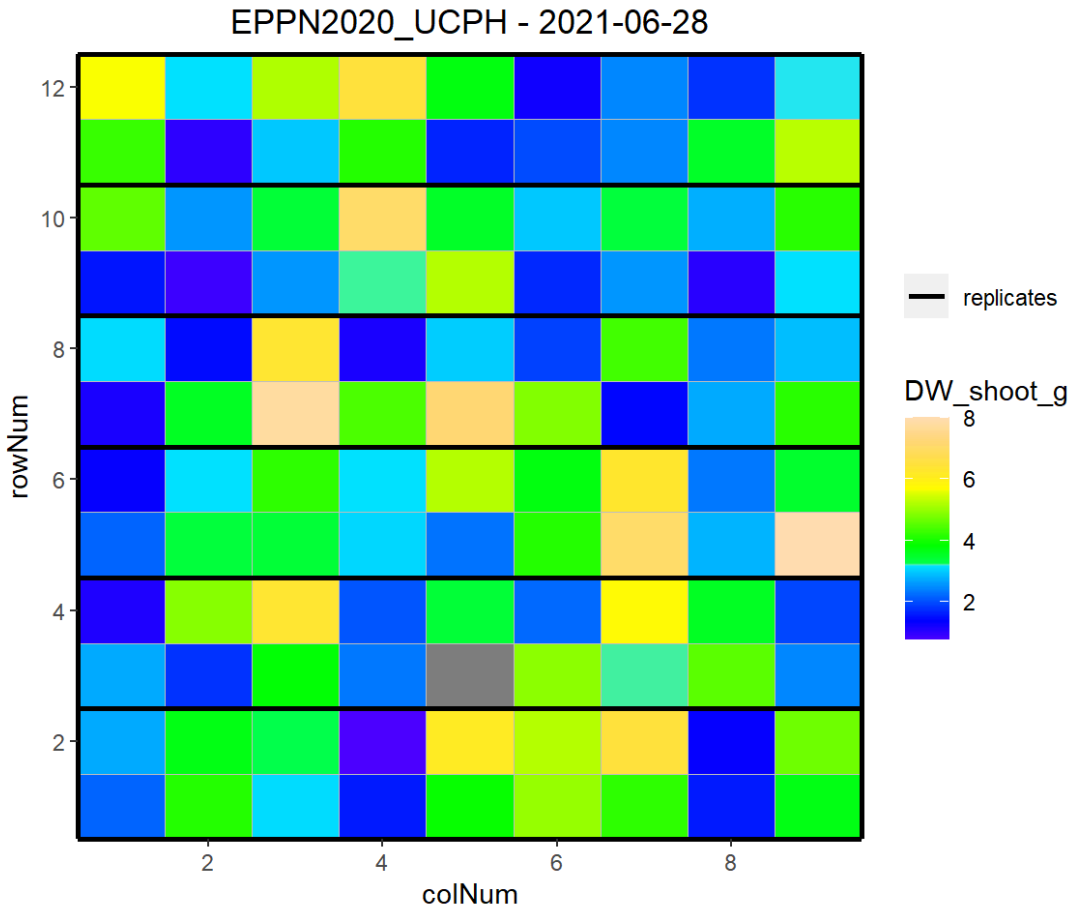
```
## [1] "How many data observations for DW_shoot_g"
## 2021-06-28
##        107
## [1] "How many data observations for DW_root_g"
## 2021-06-28
##        108
```

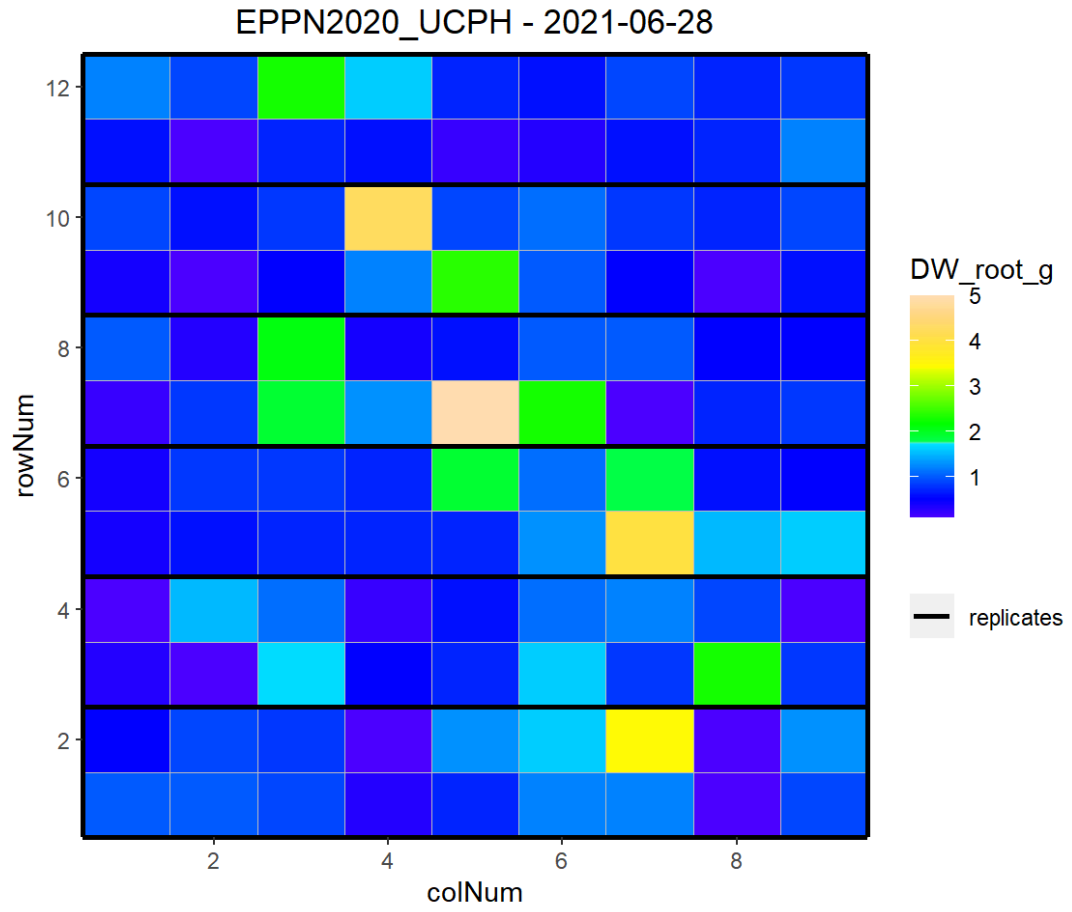
```
for (trait_name in traits) {
  print(paste("How many cleaned data observations for", trait_name))
  num_observations <- countValid(timePoint_endpoint_clean, trait_name)
  print(num_observations)
}
```

```
## [1] "How many cleaned data observations for DW_shoot_g"
## 2021-06-28
##        101
## [1] "How many cleaned data observations for DW_root_g"
## 2021-06-28
##        104
```

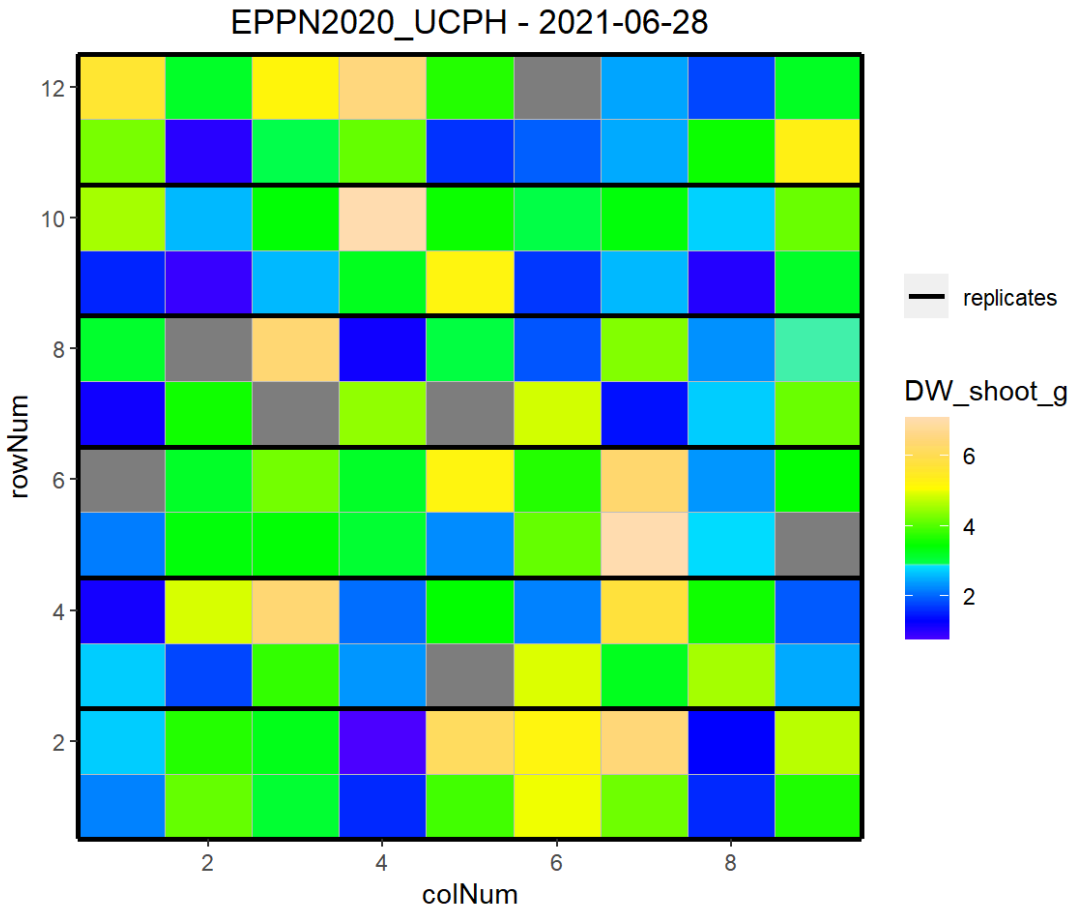
Check the heatmap of the data at harvest

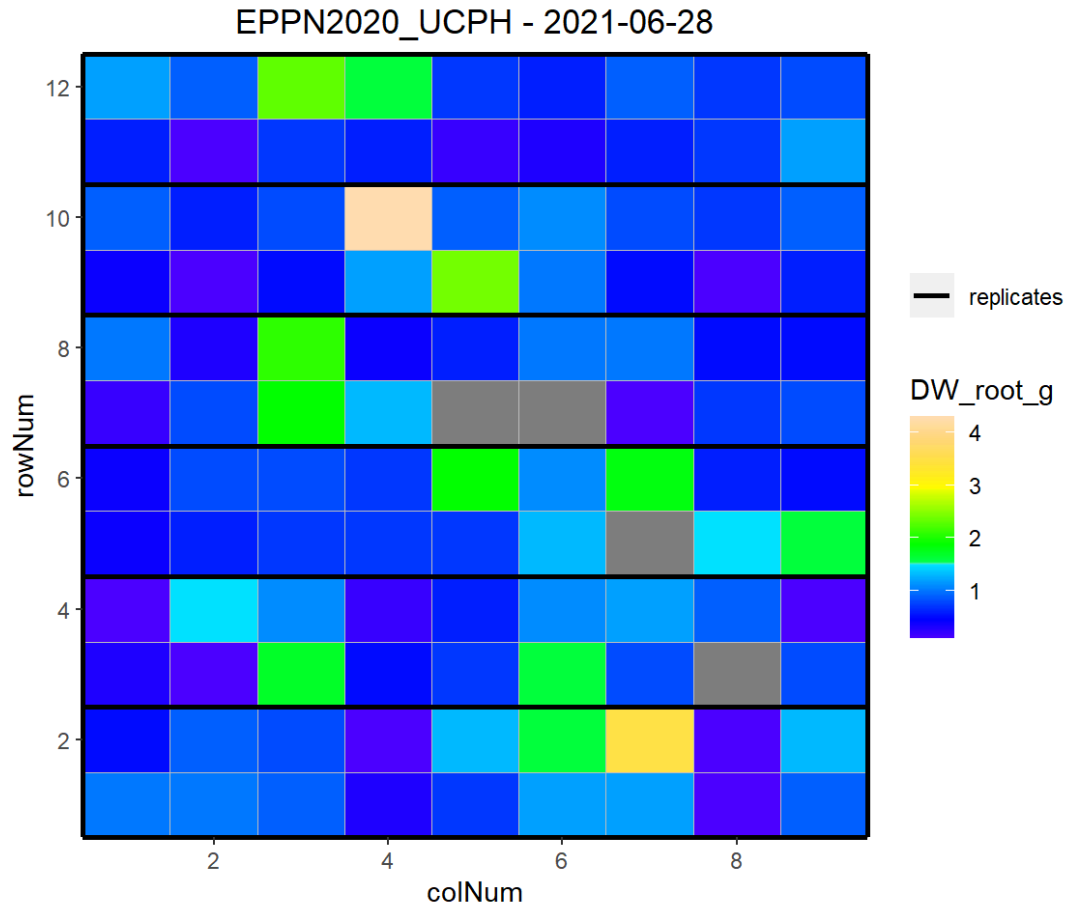
```
for (trait_name in traits) {
  plot(timePoint_endpoint,
     plotType = "layout",
     timePoints = 1,
     traits = trait_name)
}
```


EPPN2020_UCPH - 2021-06-28
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
DW_shoot_g
8
6
4
2


EPPN2020_UCPH - 2021-06-28
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
DW_root_g
5
4
3
2
1
replicates

```
for (trait_name in traits) {
  plot(timePoint_endpoint_clean,
     plotType = "layout",
     timePoints = 1,
     traits = trait_name)
}
```





2. S_timeseries

Raw data

View timePoint object

```
summary(timePoint_S)
```

```
## timePoint_S contains data for experiment EPPN2020_UCPH.
##
## It contains 17 time points.
## First time point: 2021-06-13
## Last time point: 2021-06-29
##
## The following genotypes are defined as check genotypes: EPPN_T.
```

```
getTimePoints(timePoint_S)
```

```
##    timeNumber  timePoint
## 1           1 2021-06-13
## 2           2 2021-06-14
## 3           3 2021-06-15
## 4           4 2021-06-16
## 5           5 2021-06-17
## 6           6 2021-06-18
## 7           7 2021-06-19
## 8           8 2021-06-20
## 9           9 2021-06-21
## 10         10 2021-06-22
## 11         11 2021-06-23
## 12         12 2021-06-24
## 13         13 2021-06-25
## 14         14 2021-06-26
## 15         15 2021-06-27
## 16         16 2021-06-28
## 17         17 2021-06-29
```

```
num_timepoints <- getTimePoints(timePoint_S)
```

Count the number of observations per trait and time point

We focus on the Height [cm] and Leaf area, because these are the two most common among the platforms.

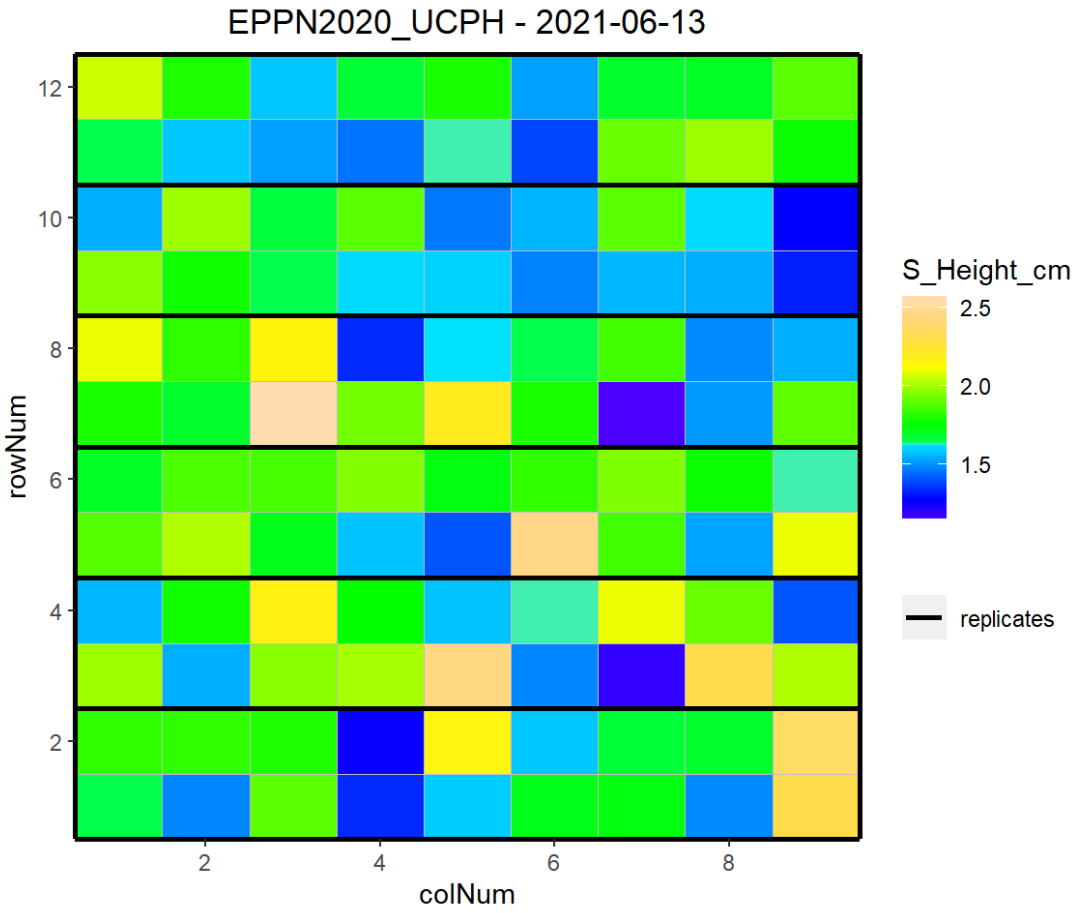
Height is computed for 6 platforms out of 9 and area for 4 out of 9.

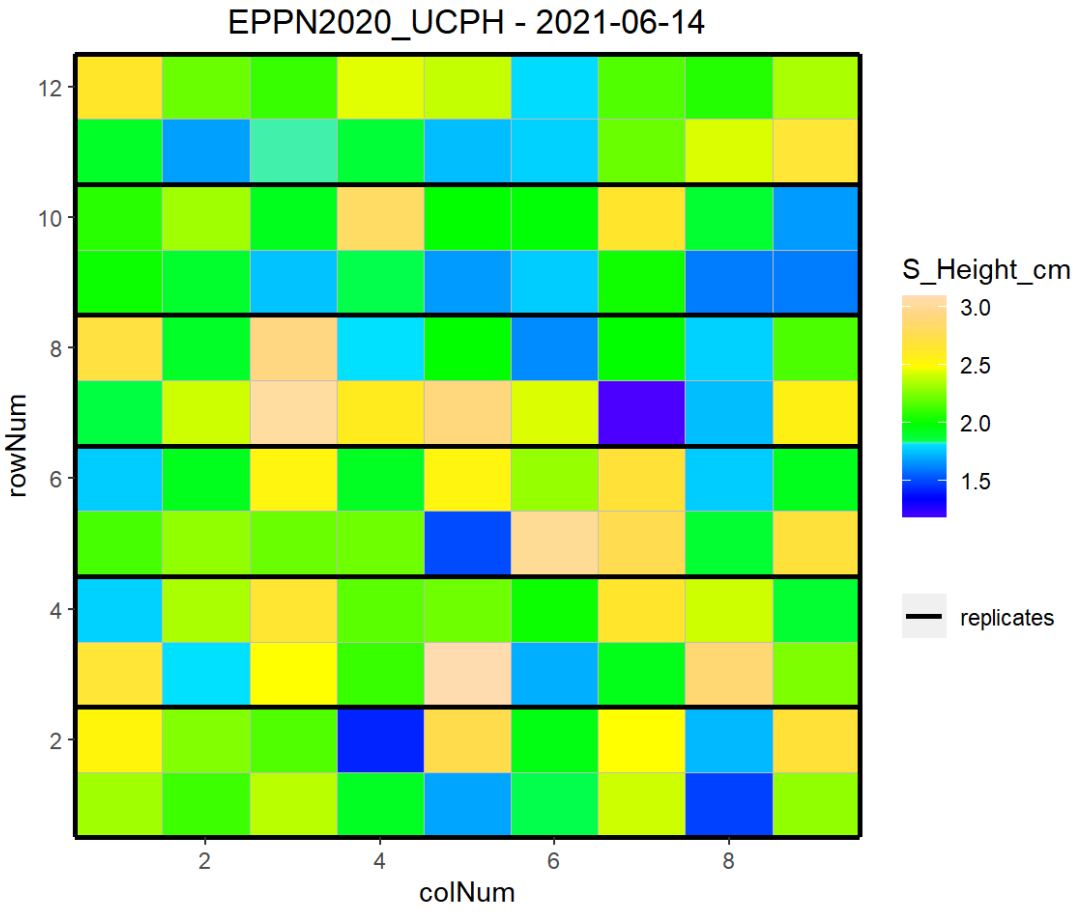
```
var_voulues <- c(variables_S[1])
traits <- var_voulues

for (trait_name in traits) {
  print(paste("How many observations for", trait_name))
  valid_count <- countValid(timePoint_S, trait_name)
  print(valid_count)
}
```

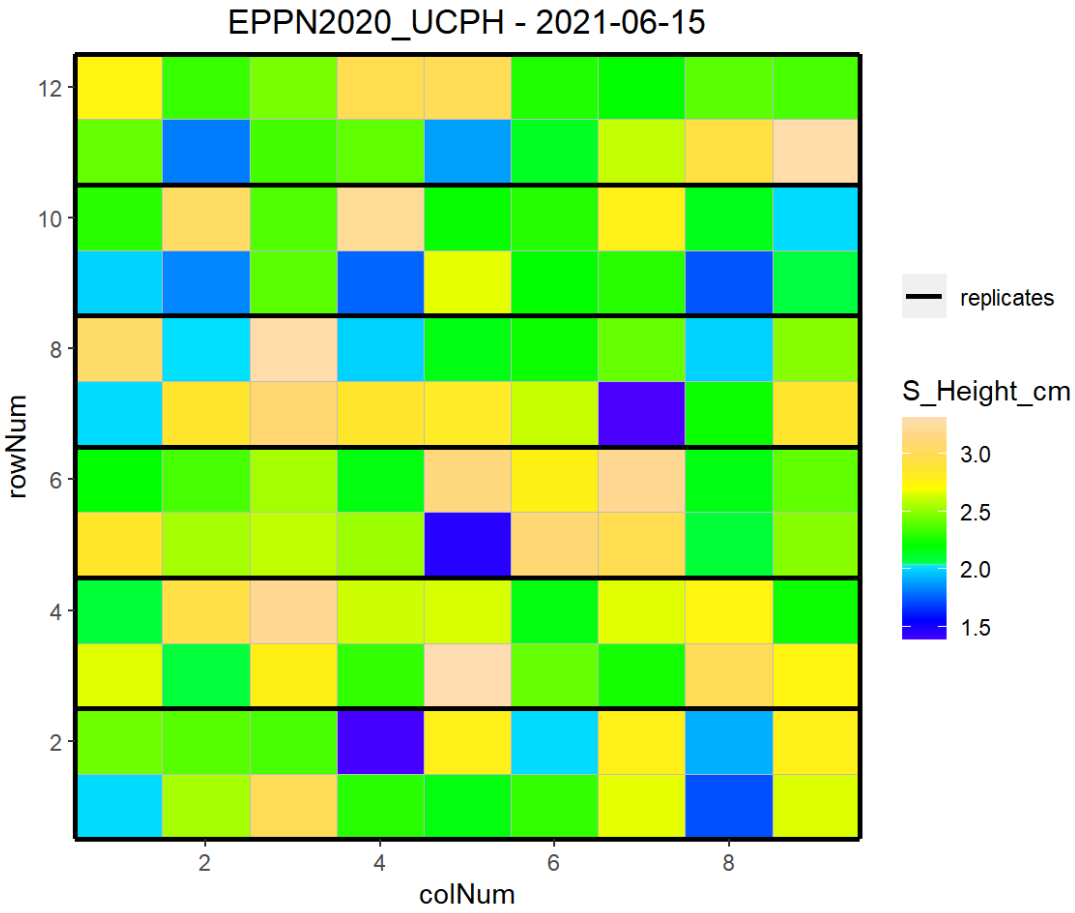
```
## [1] "How many observations for S_Height_cm"
## 2021-06-13 2021-06-14 2021-06-15 2021-06-16 2021-06-17 2021-06-18 2021-06-19
##        108        108        108        108        108        108        108
## 2021-06-20 2021-06-21 2021-06-22 2021-06-23 2021-06-24 2021-06-25 2021-06-26
##        108        108        108        108        108        108        108
## 2021-06-27 2021-06-28 2021-06-29
##         66        108         84
```

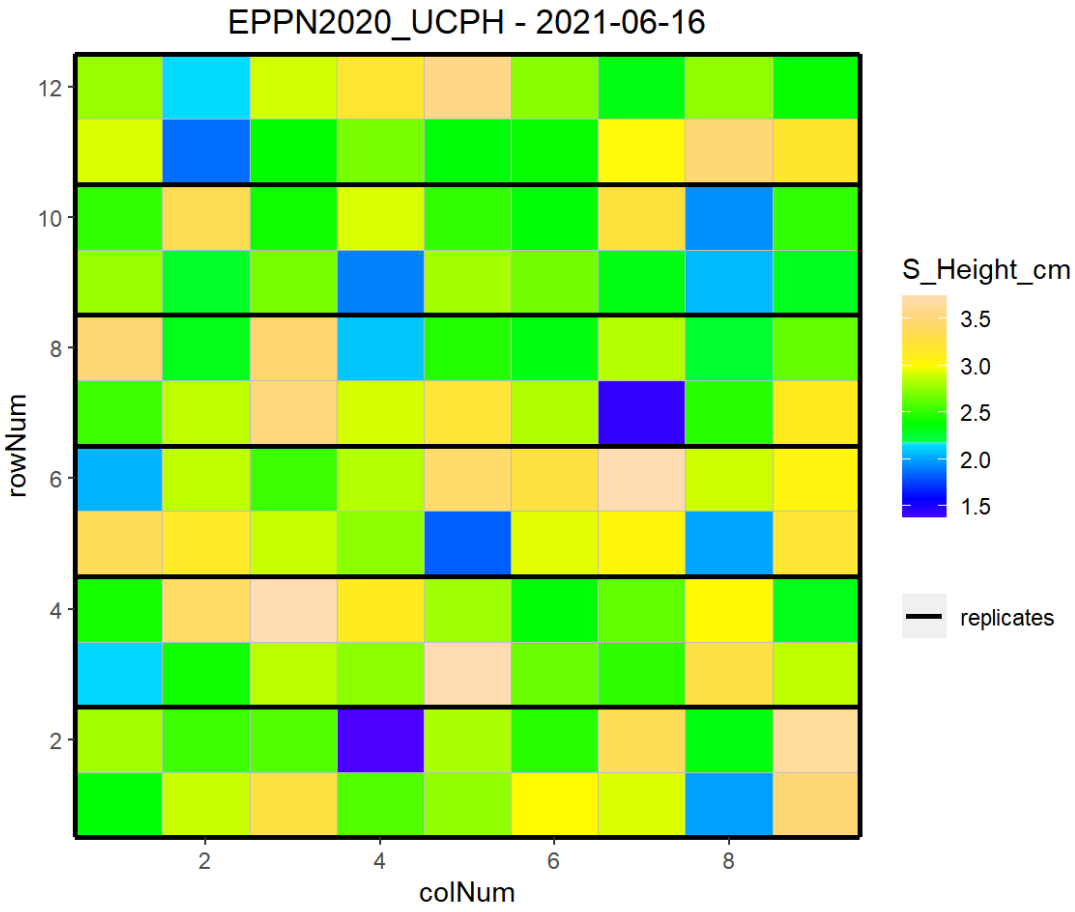
Check the heatmap of the raw data per time point

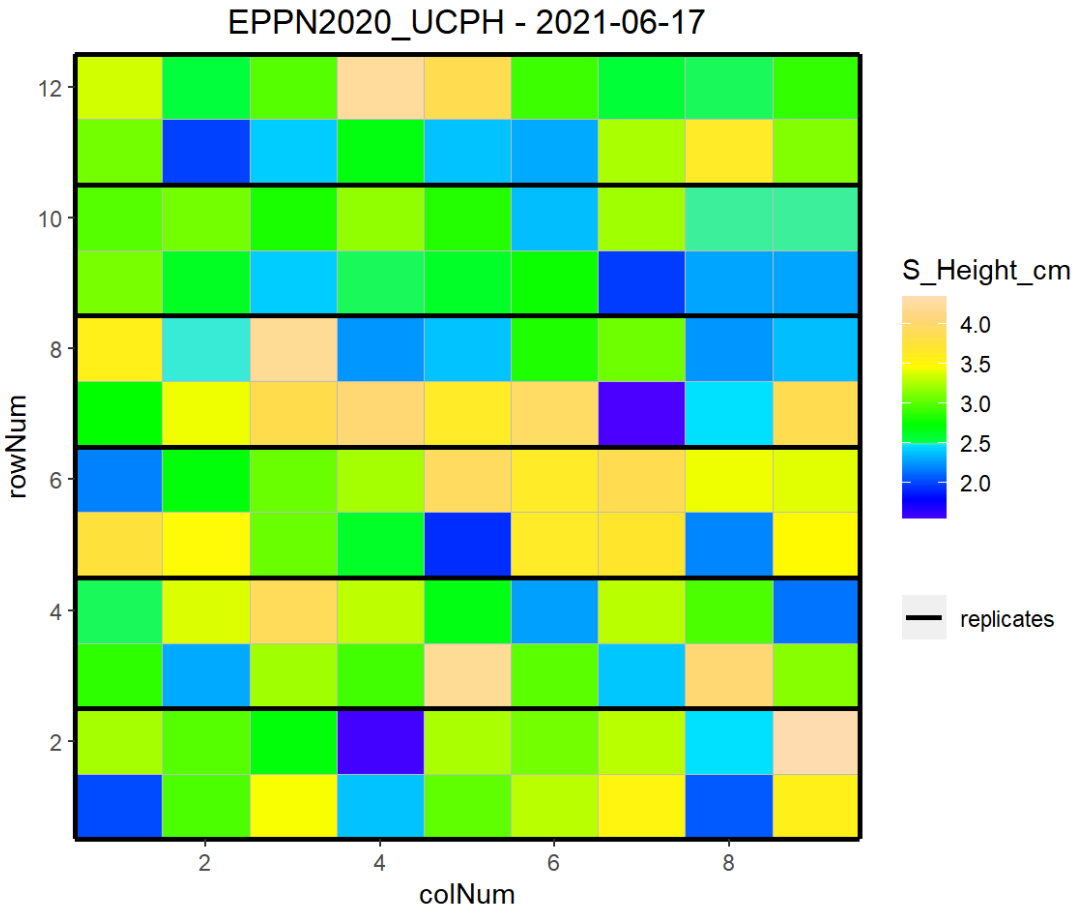
```
for (trait_name in traits) {
  for (tp in 1:length(num_timepoints$timeNumber)) {
    plot(timePoint_S,
         plotType = "layout",
         timePoints = tp,
         traits = trait_name)
  }
}
```

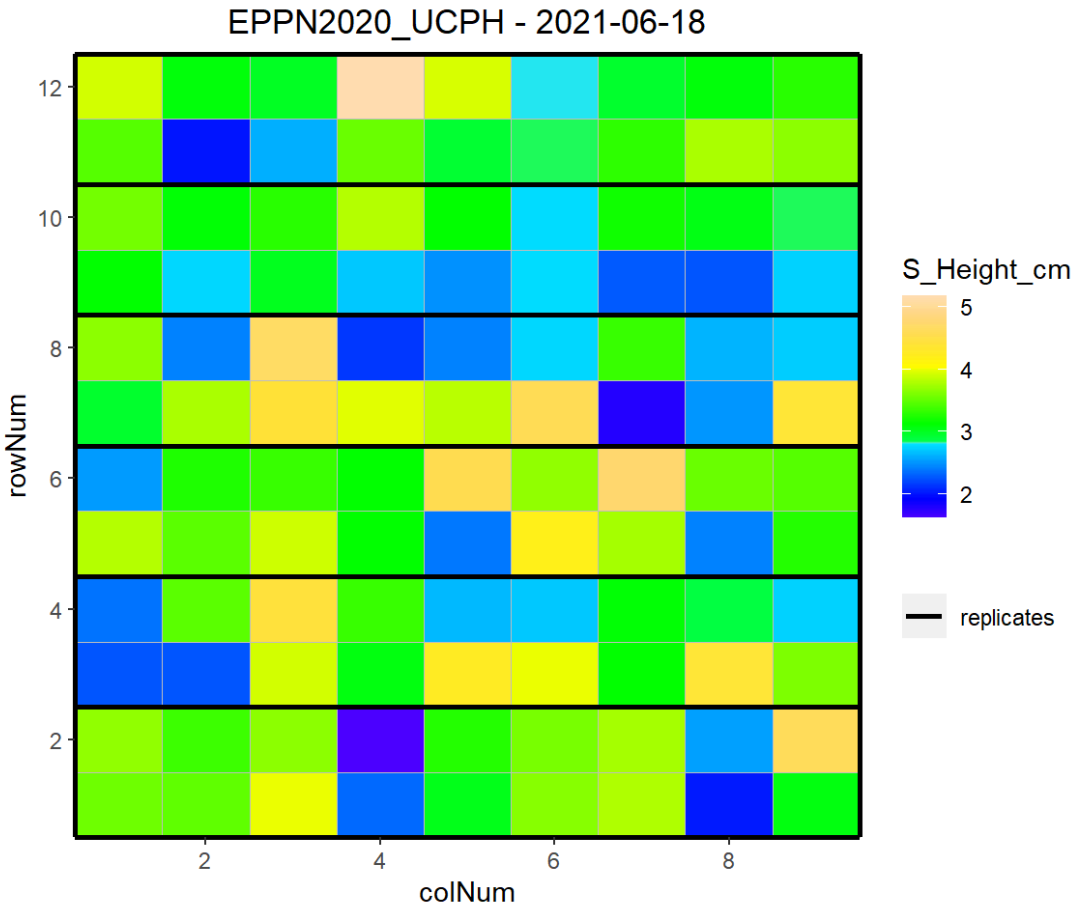

EPPN2020_UCPH - 2021-06-13
rowNum
colNum
S_Height_cm
2.5
2.0
1.5
replicates

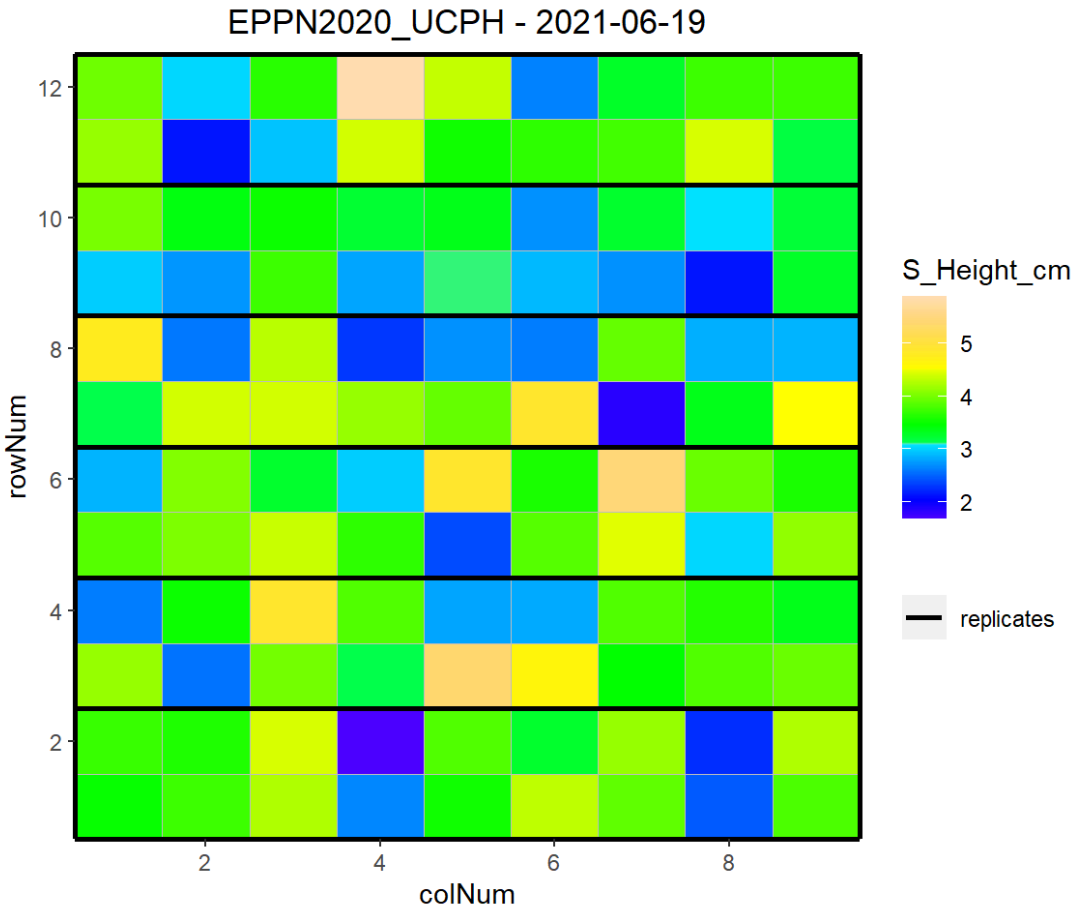

EPPN2020_UCPH - 2021-06-14
rowNum
colNum
S_Height_cm
3.0
2.5
2.0
1.5
replicates

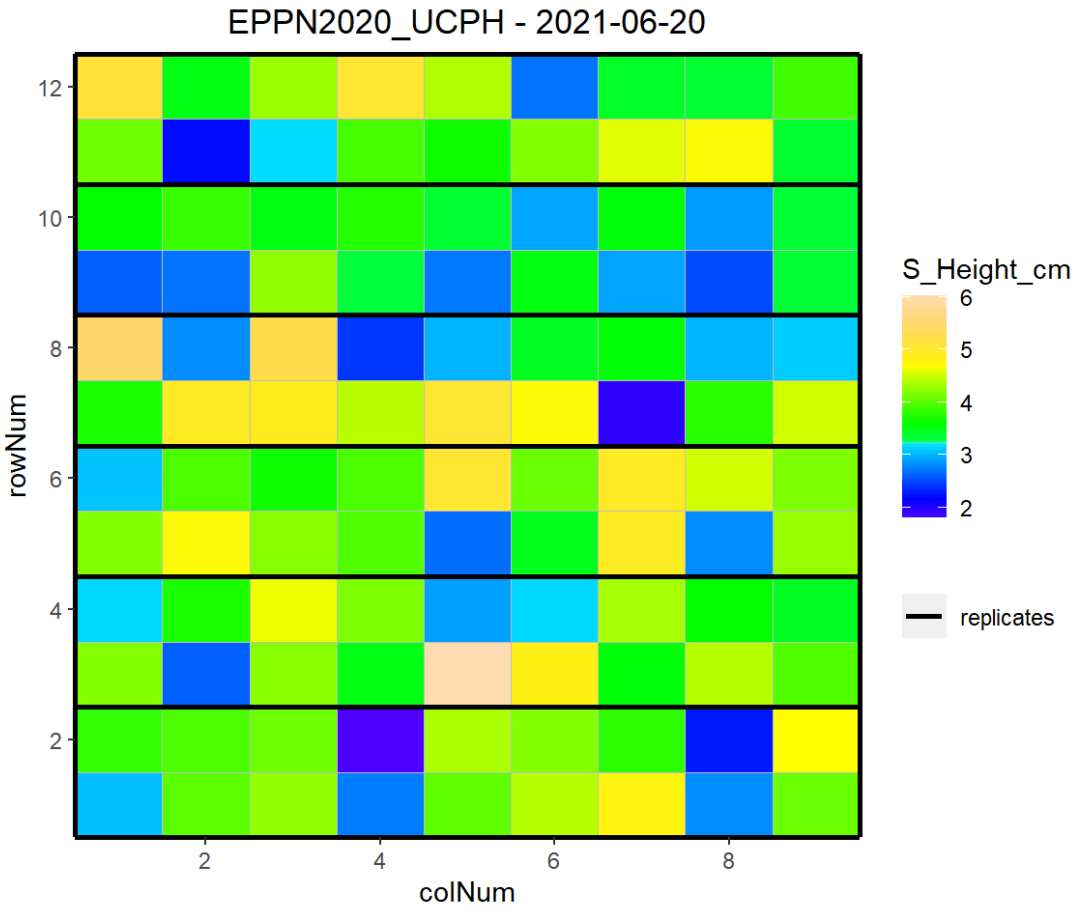

EPPN2020_UCPH - 2021-06-15
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
3.0
2.5
2.0
1.5

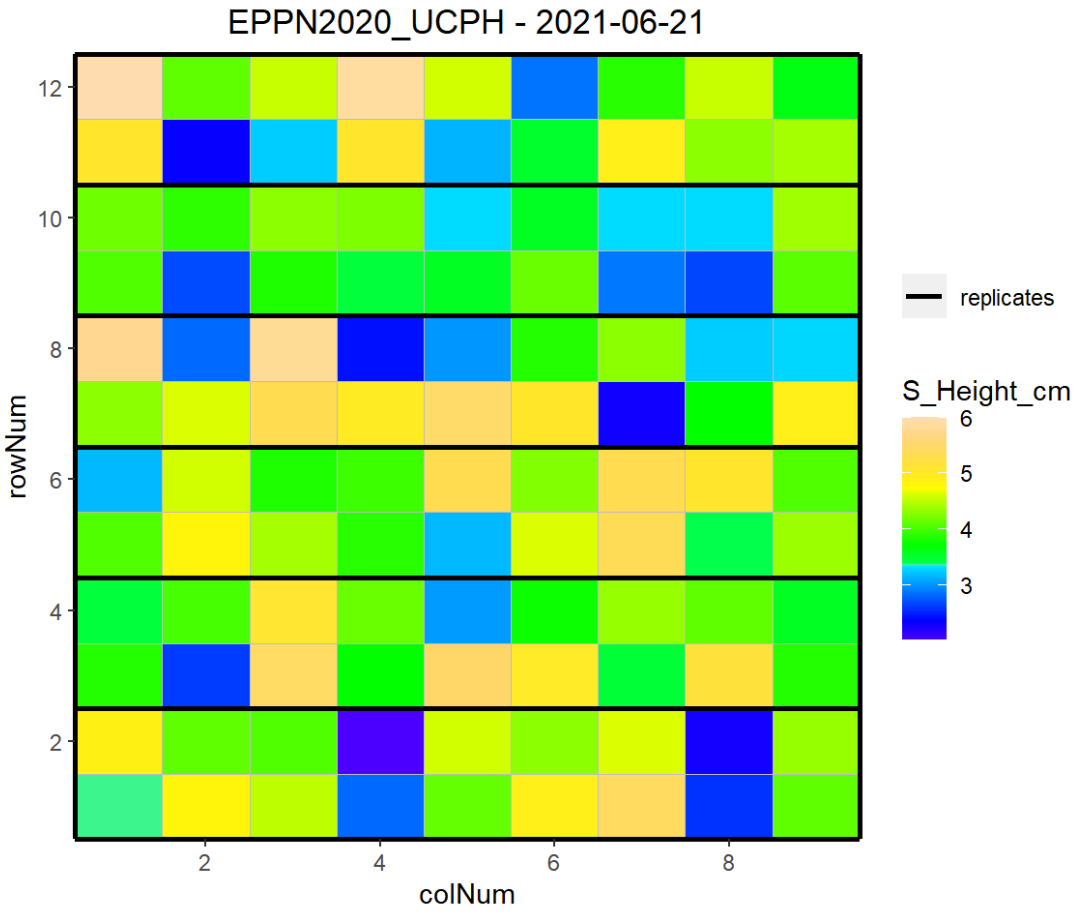

EPPN2020_UCPH - 2021-06-16
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
3.5
3.0
2.5
2.0
1.5
replicates

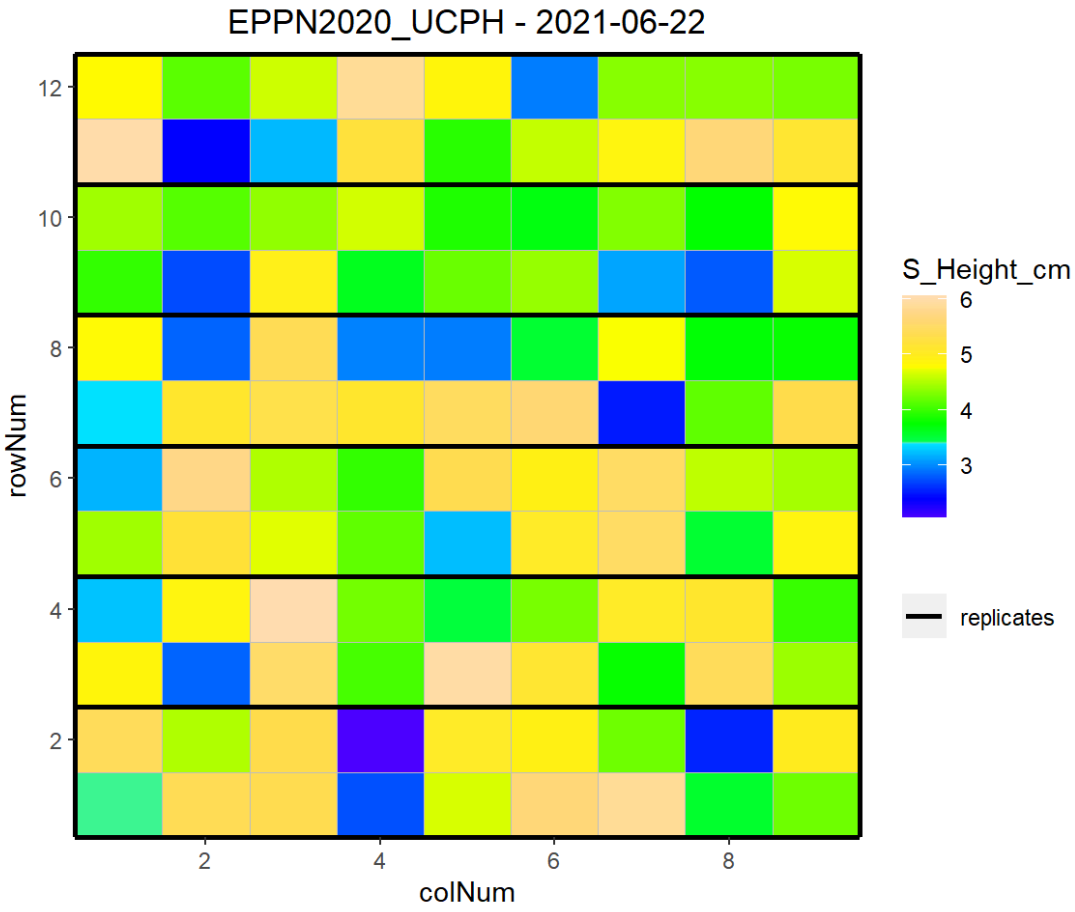

EPPN2020_UCPH - 2021-06-17
rowNum
colNum
S_Height_cm
4.0
3.5
3.0
2.5
2.0
replicates

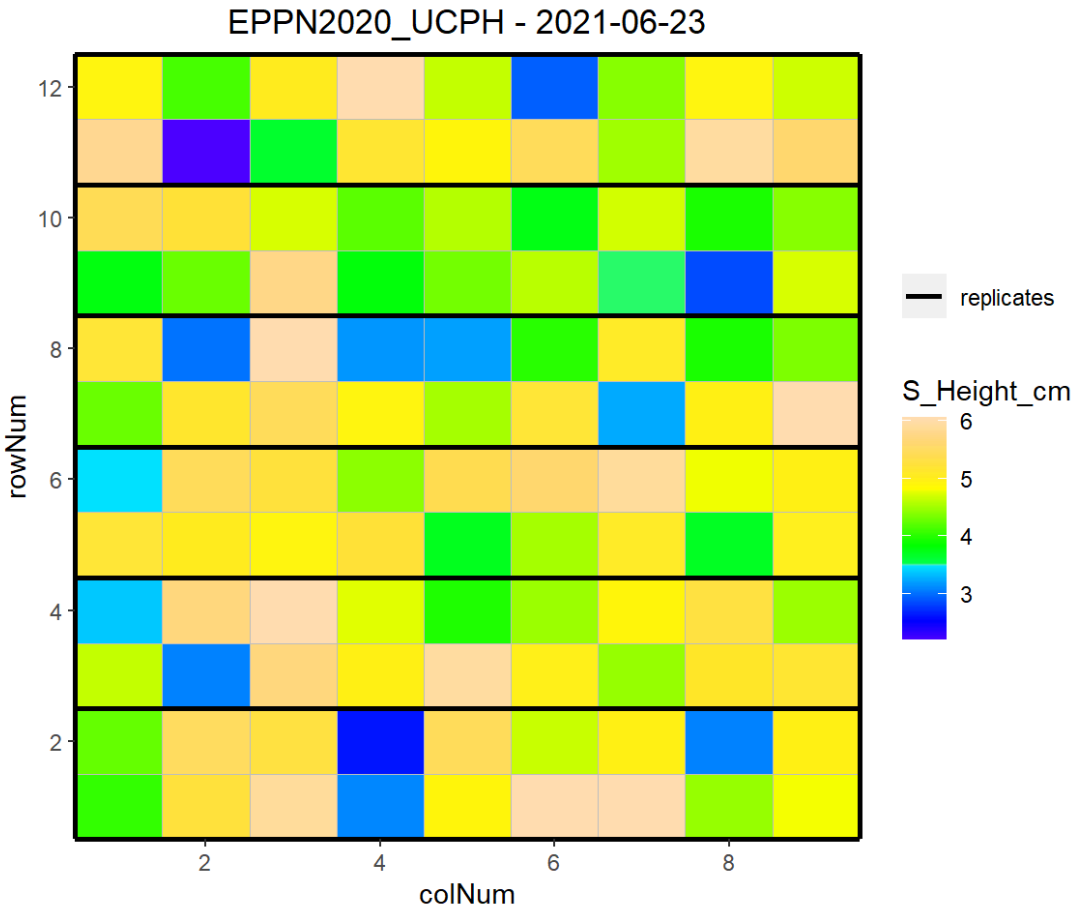

EPPN2020_UCPH - 2021-06-18
rowNum
colNum
S_Height_cm
5
4
3
2
replicates

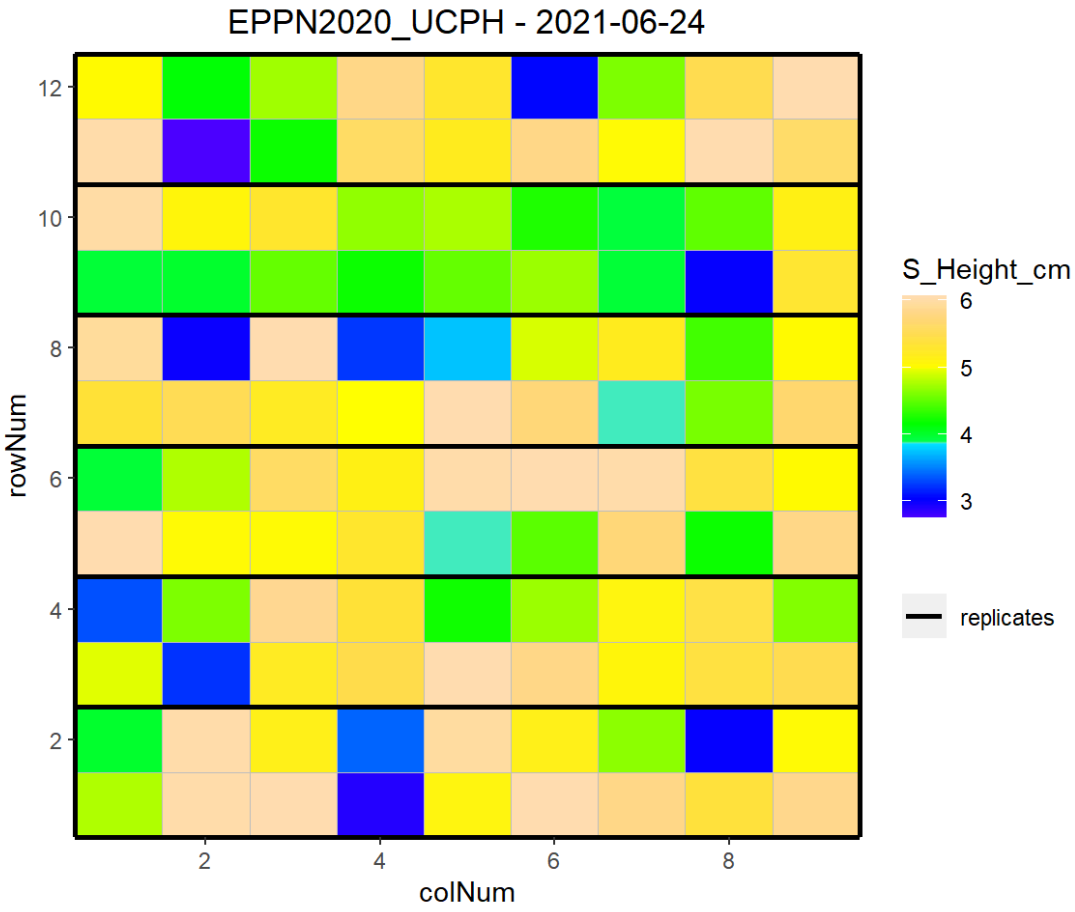

EPPN2020_UCPH - 2021-06-19
rowNum
colNum
S_Height_cm
5
4
3
2
replicates

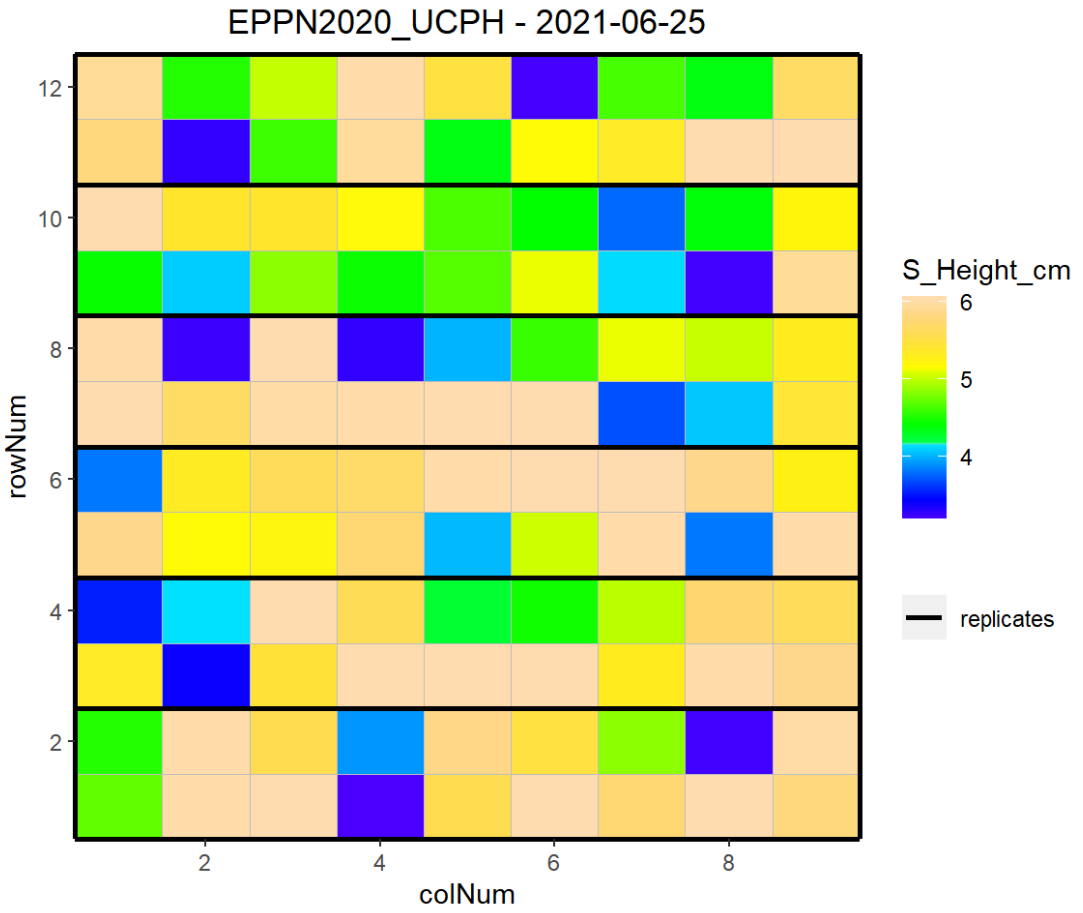

EPPN2020_UCPH - 2021-06-20
rowNum
colNum
S_Height_cm
6
5
4
3
2
replicates

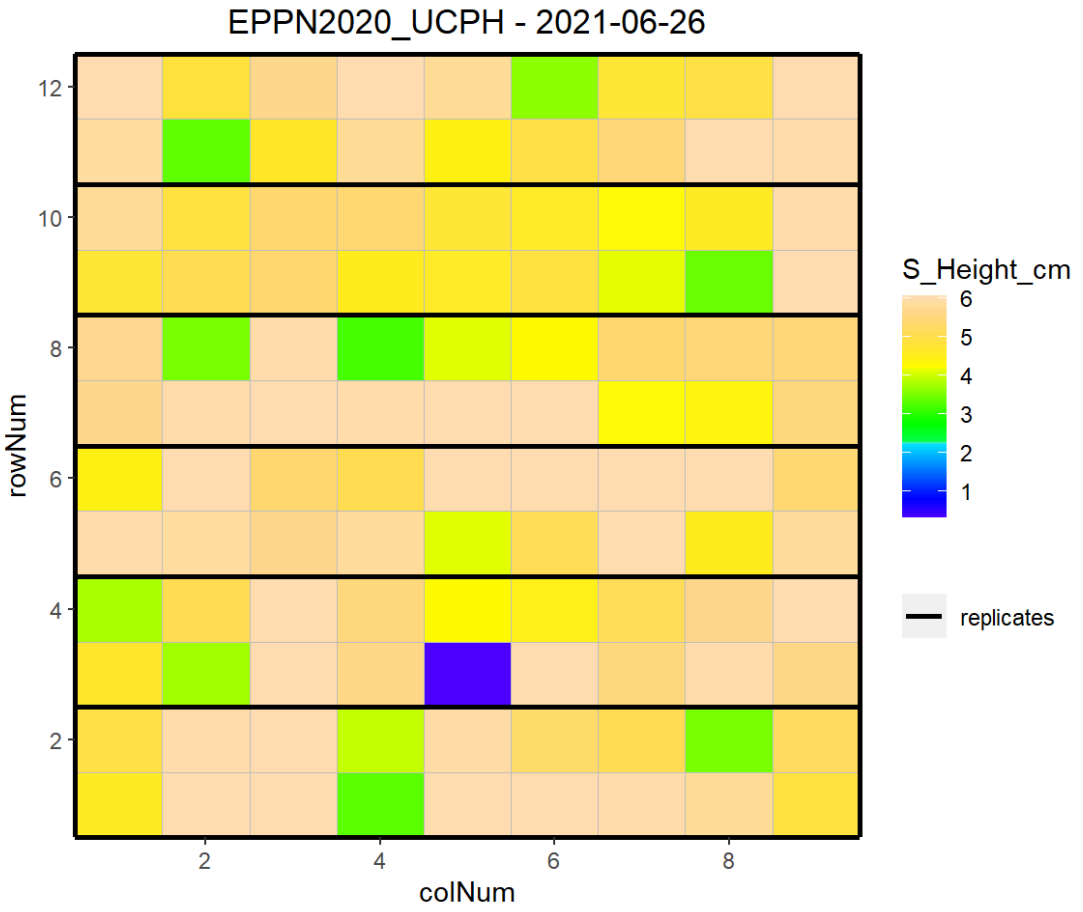

EPPN2020_UCPH - 2021-06-21
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
6
5
4
3

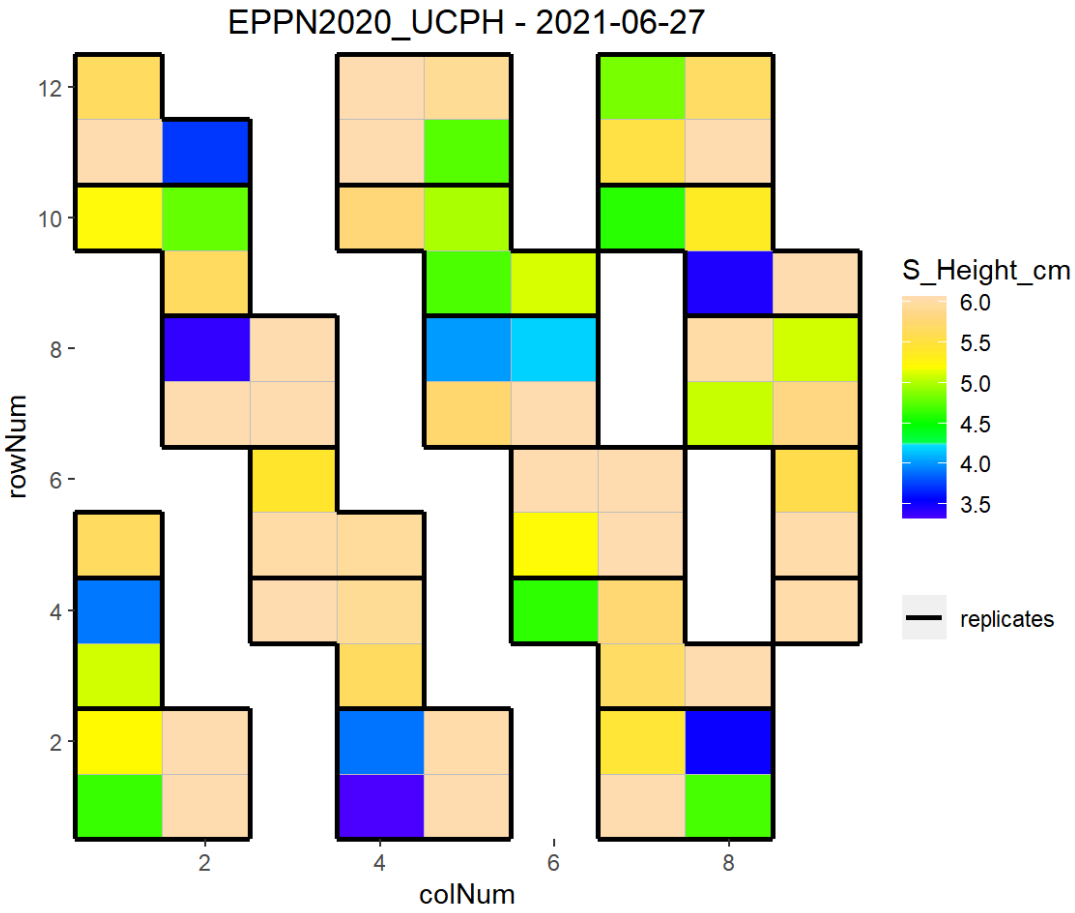

EPPN2020_UCPH - 2021-06-22
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
6
5
4
3
replicates

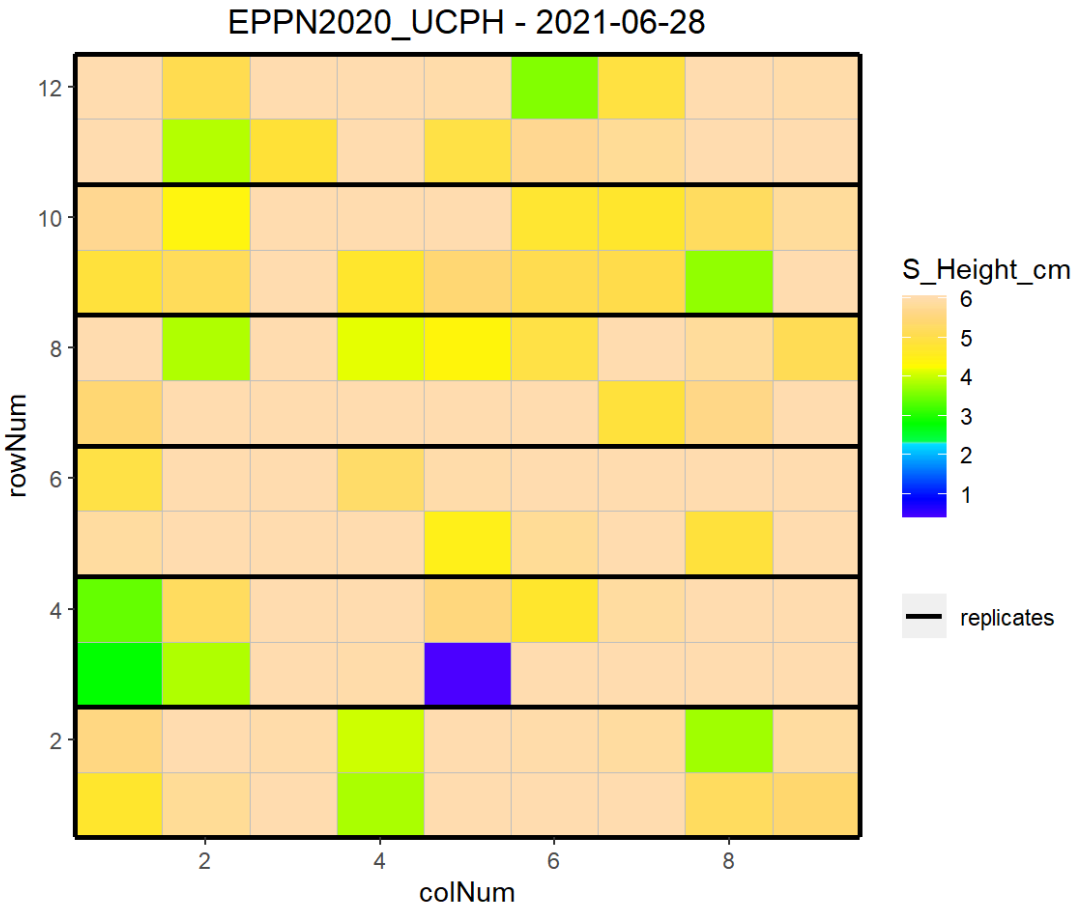

EPPN2020_UCPH - 2021-06-23
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
6
5
4
3

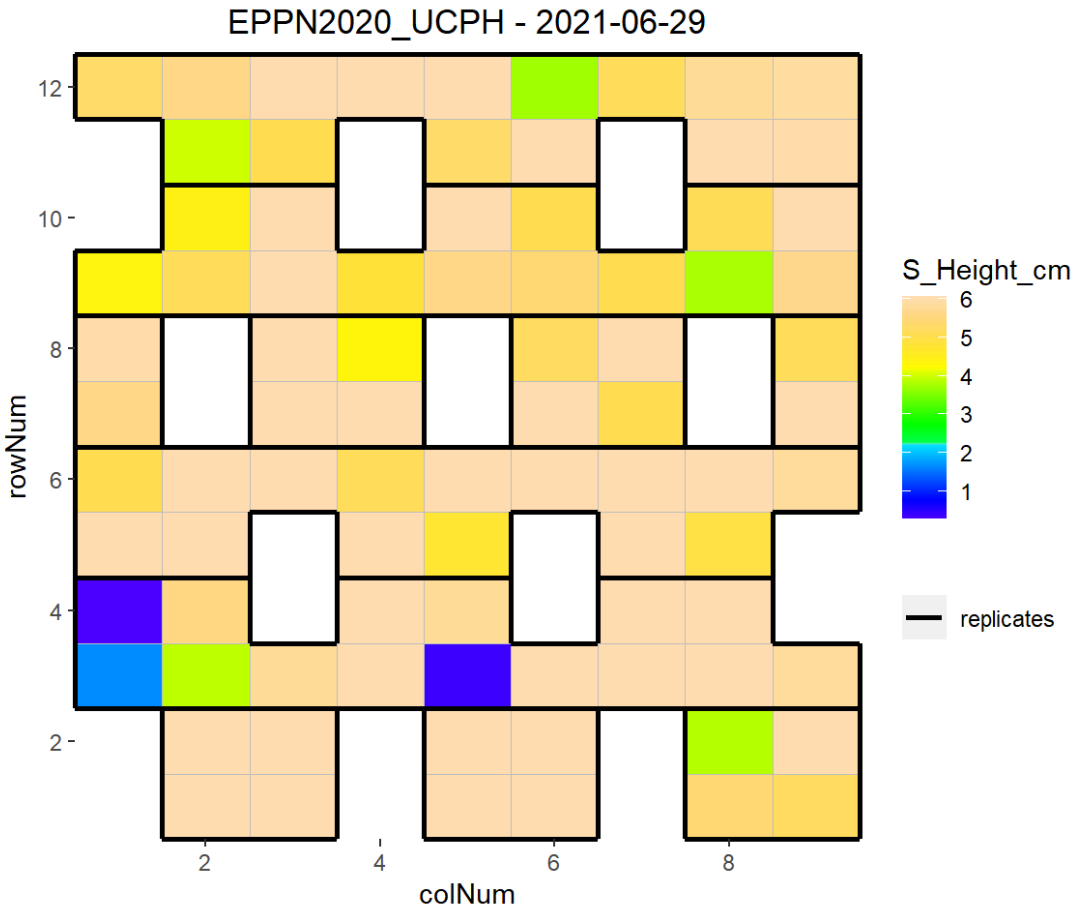

EPPN2020_UCPH - 2021-06-24
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
6
5
4
3
replicates


EPPN2020_UCPH - 2021-06-25
rowNum
colNum
S_Height_cm
6
5
4
replicates


EPPN2020_UCPH - 2021-06-26
rowNum
colNum
S_Height_cm
6
5
4
3
2
1
replicates

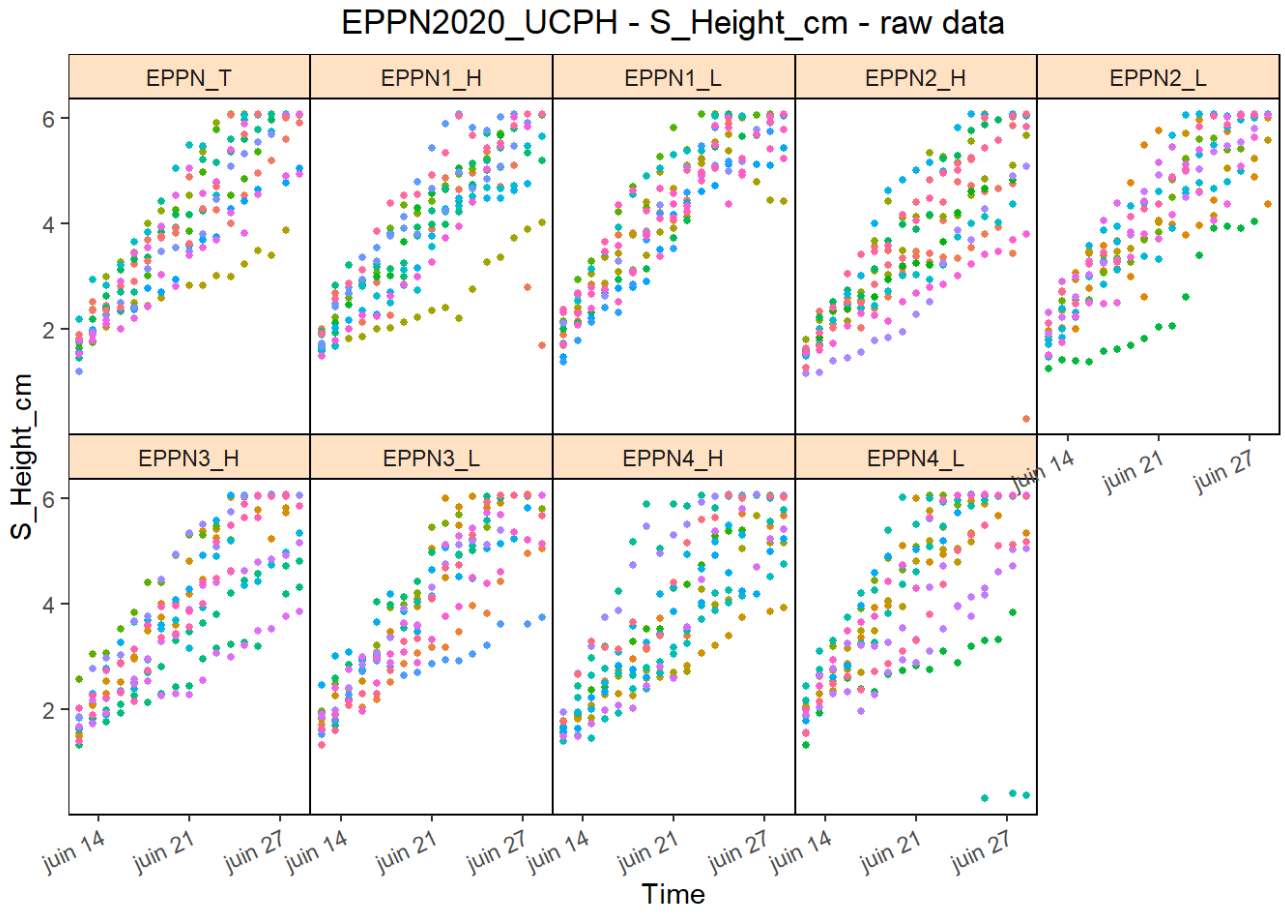

EPPN2020_UCPH - 2021-06-27
rowNum
colNum
S_Height_cm
6.0
5.5
5.0
4.5
4.0
3.5
replicates


EPPN2020_UCPH - 2021-06-28
rowNum
colNum
S_Height_cm
6
5
4
3
2
1
replicates



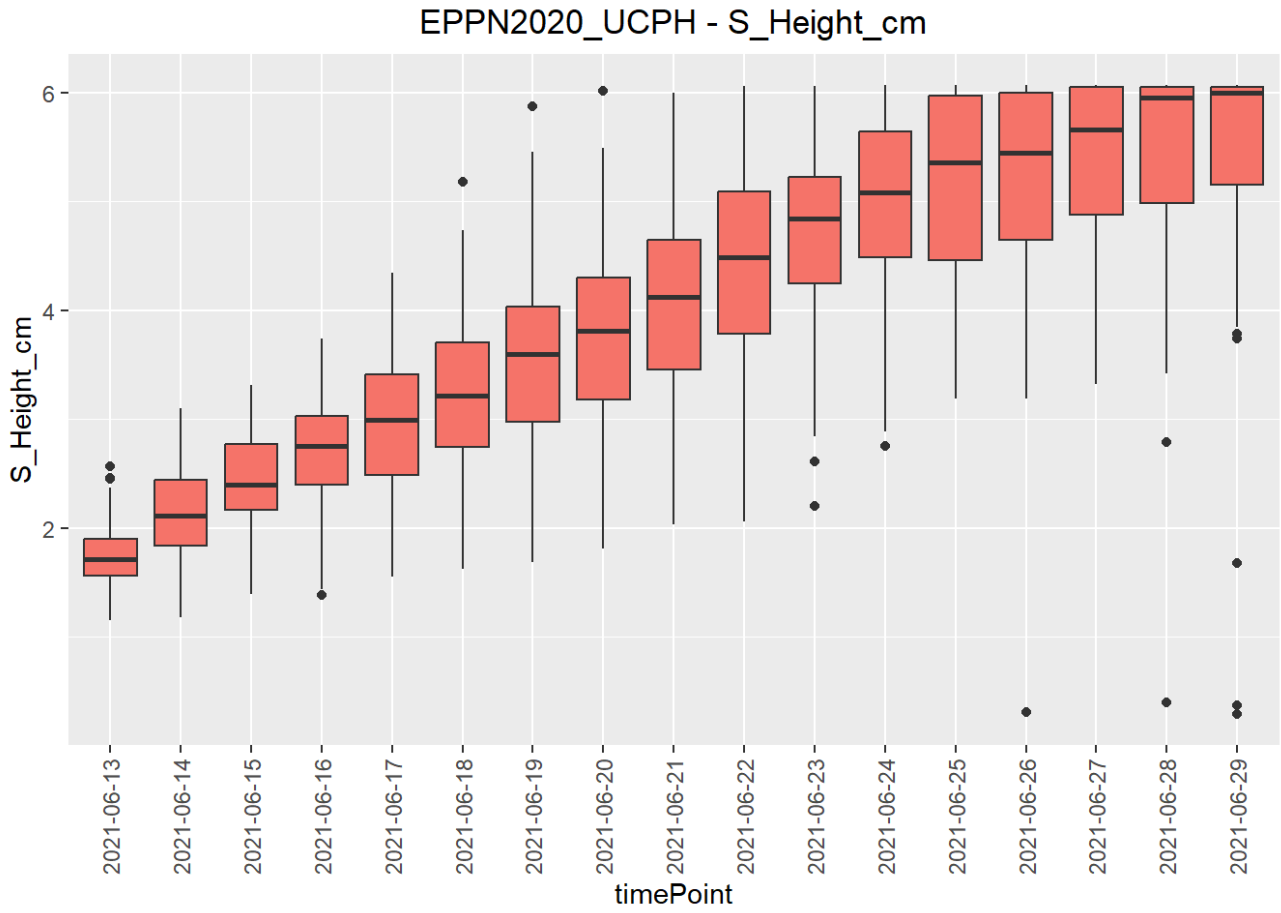
Check time course of raw data per time point

```
for (trait_name in traits) {
  plot(timePoint_S,
     traits = trait_name,
     plotType = "raw")
}
```



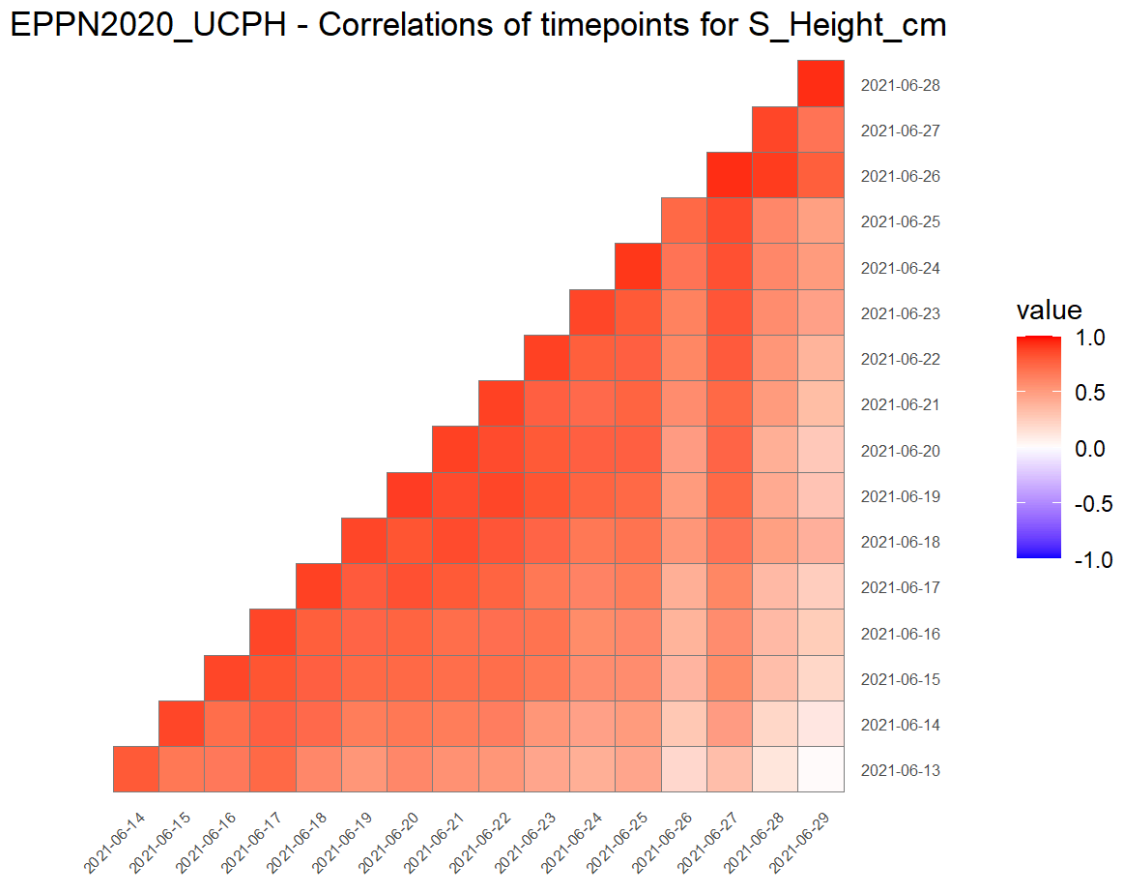
Check the boxplots of raw data per time point

```
for (trait_name in traits) {
  plot(timePoint_S,
     plotType = "box",
     traits = trait_name)
}
```



Check the correlation plots of raw data per time point

```
for (trait_name in traits) {
  plot(timePoint_S,
     plotType = "cor",
     traits = trait_name)
}
```

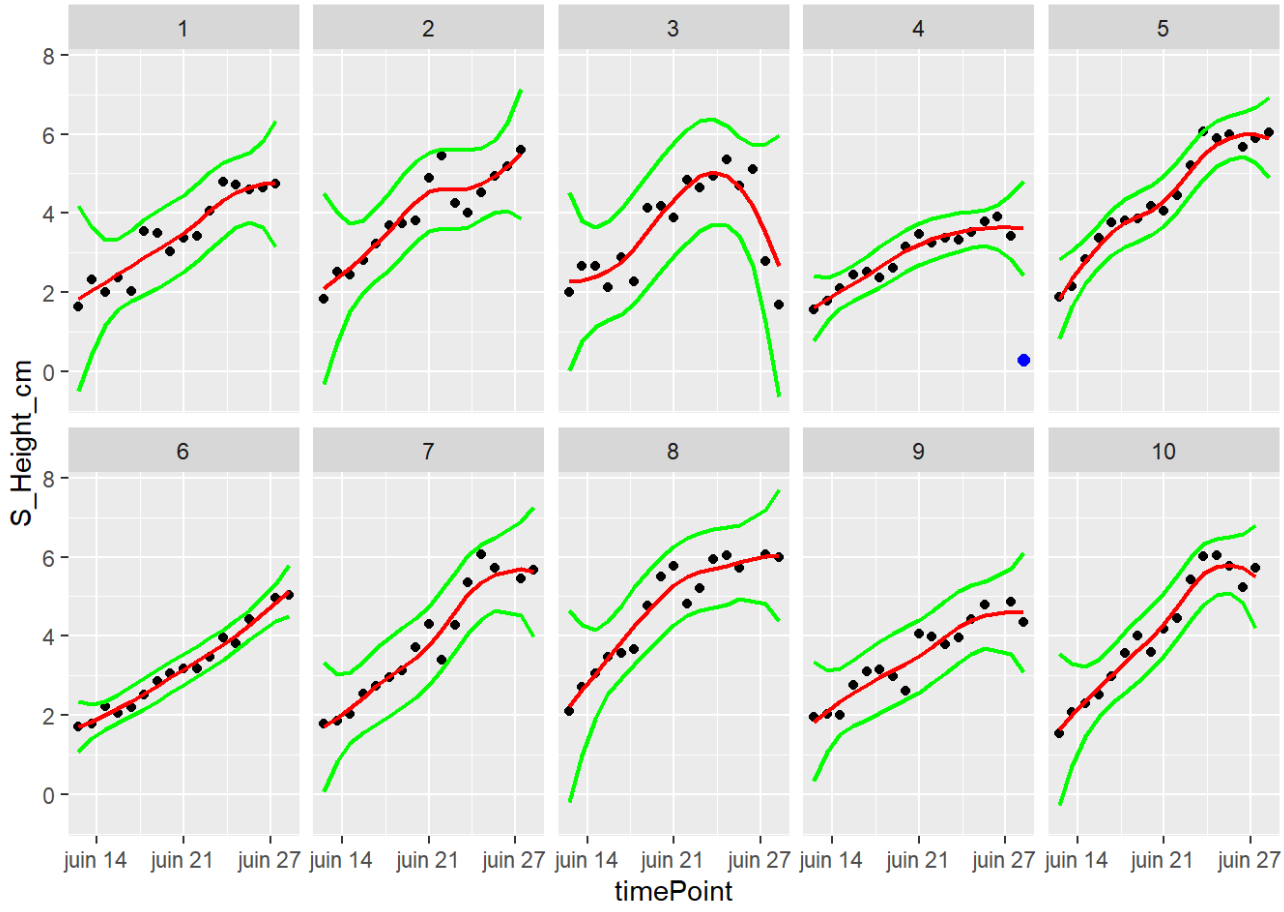


1. Detection of outliers for single observations

Using the SingleOut detect and single functions. We select a subset of plants to adjust the settings for the confIntSize and nnLocfit.

```
plantSel<- c(1,2,3,4,5,6,7,8,9,10)

ci <- 5 # confidence interval
nn <- 0.8 # nearest neighbor
ce <- FALSE
```

```
for (trait_name in traits) {
  variable_name <- paste0("Single_test_", trait_name)

  single_test <- detectSingleOut(
    TP = timePoint_S,
    trait = trait_name,
    plotIds = plantSel,
    confIntSize = ci,
    nnLocfit = nn,
    checkEdges = TRUE # check for outlier values in start and end of experiment
  )

  assign(variable_name, single_test)

  plot(single_test, outOnly = FALSE)
}
```



We can then run on all plants of the data set.

```
for (trait_name in traits) {
  single_test_object_name <- paste0("Single_test_", trait_name)
  Single_test <- get(single_test_object_name)
    if (any(Single_test$outlier == 1)) {
    outliers_count <- with(Single_test[Single_test$outlier == 1,], table(timePoint))
    print(trait_name)
    print(outliers_count)

    Single_outliers <- removeSingleOut(timePoint_S, Single_test)
    assign(paste0("Single_outliers_", trait_name), Single_outliers)

    readr::write_tsv(Single_test, sprintf("%s/single_outliers_%s.tsv", datadir, trait_n
ame))
  } else {
    cat("No outlier for", trait_name, "\n")
  }
}
```

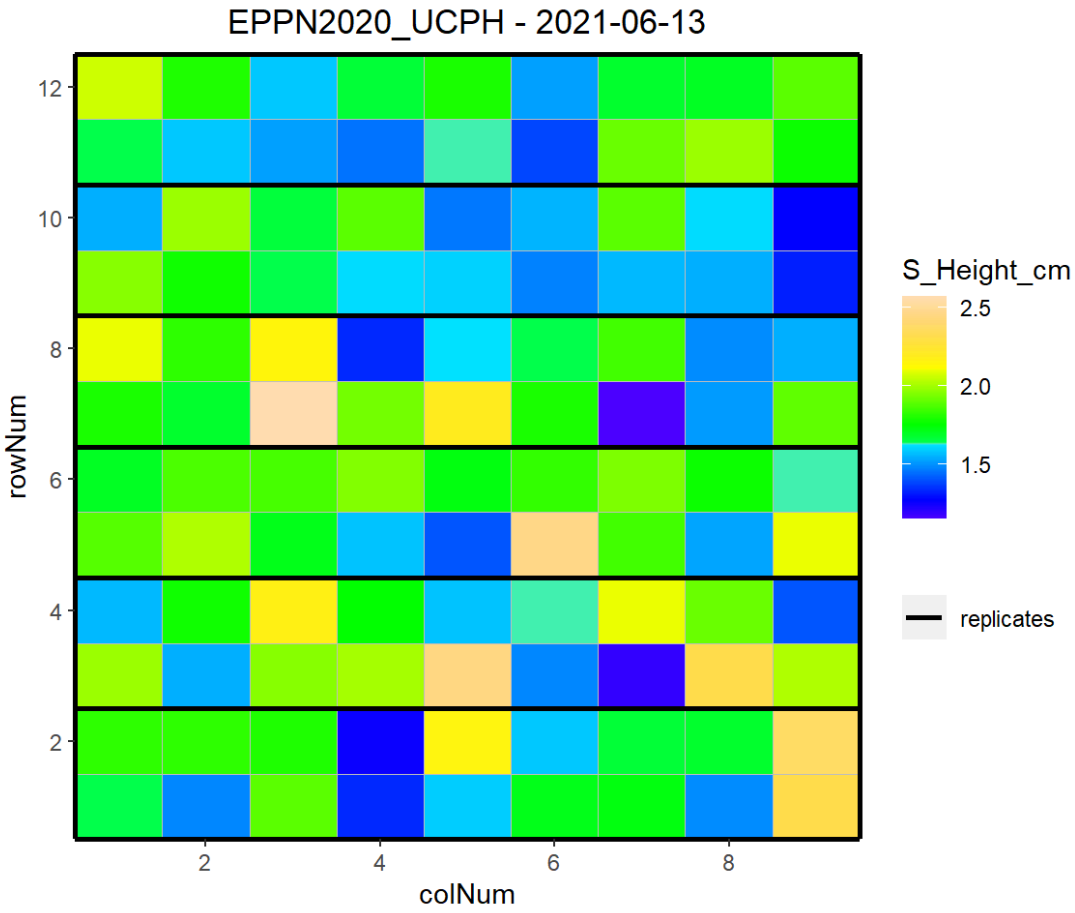
```
## [1] "S_Height_cm"
## timePoint
## 2021-06-29
##          1
```

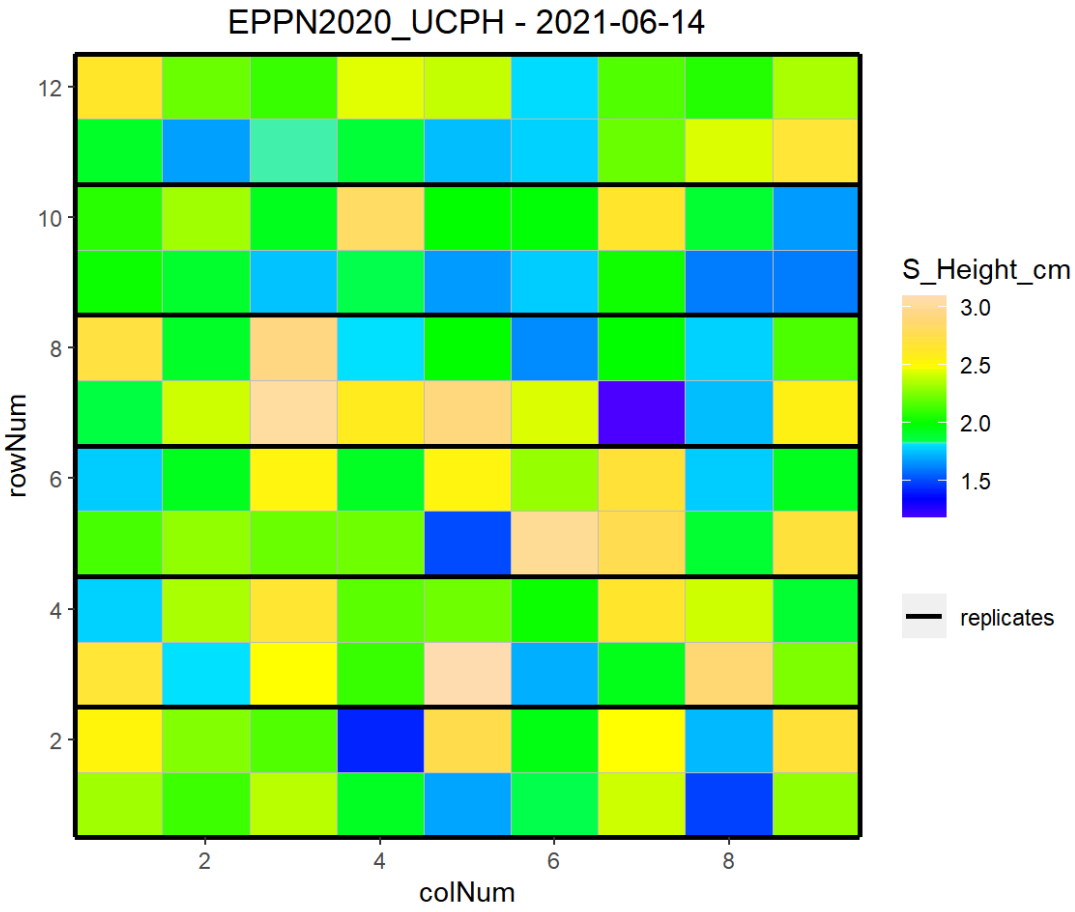
Data visualisation after single observations outliers removal

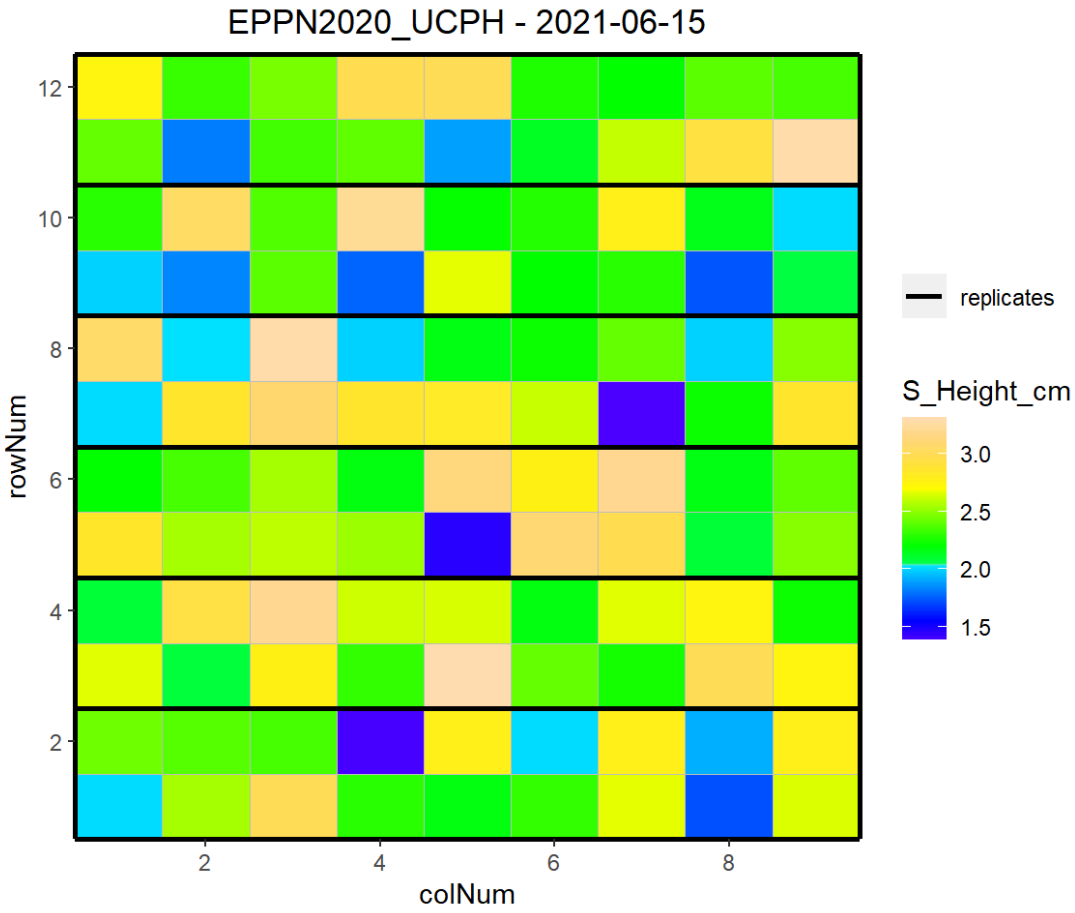
Heatmap of data

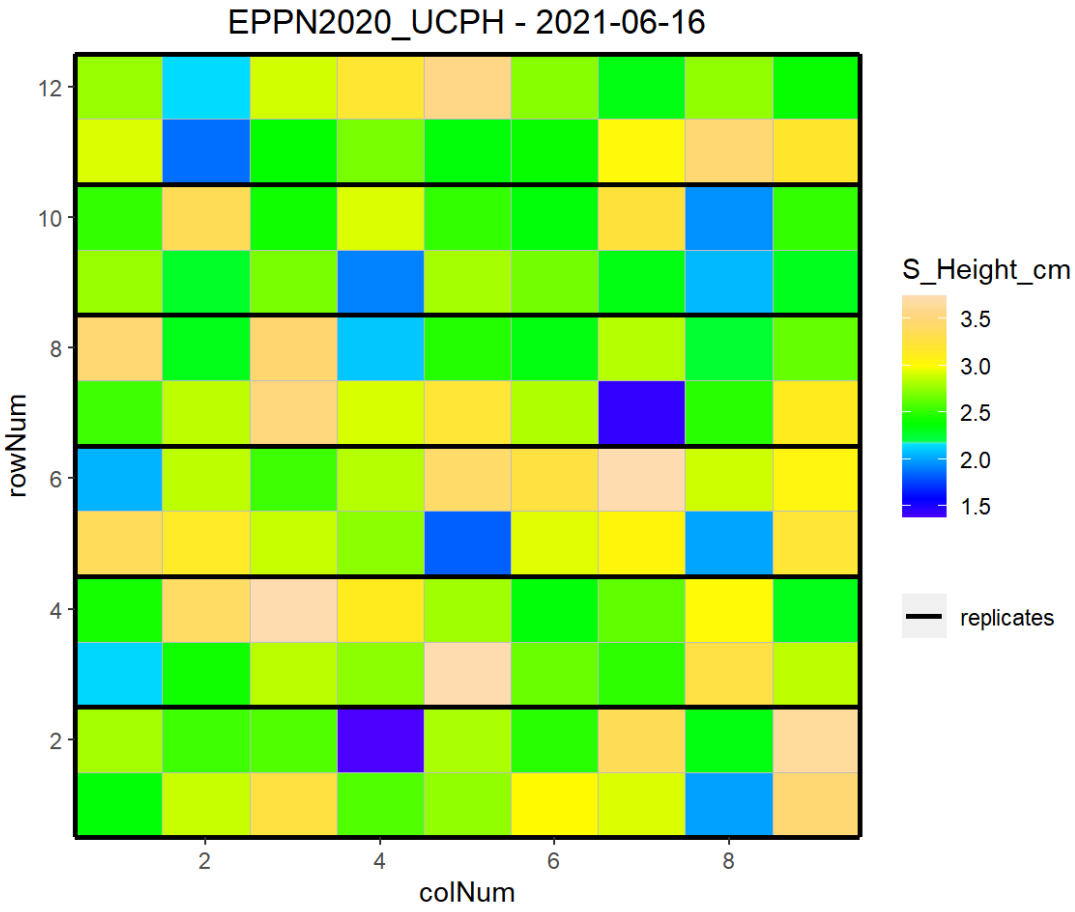
Check the heatmap of the data with outliers detection at all the time points.

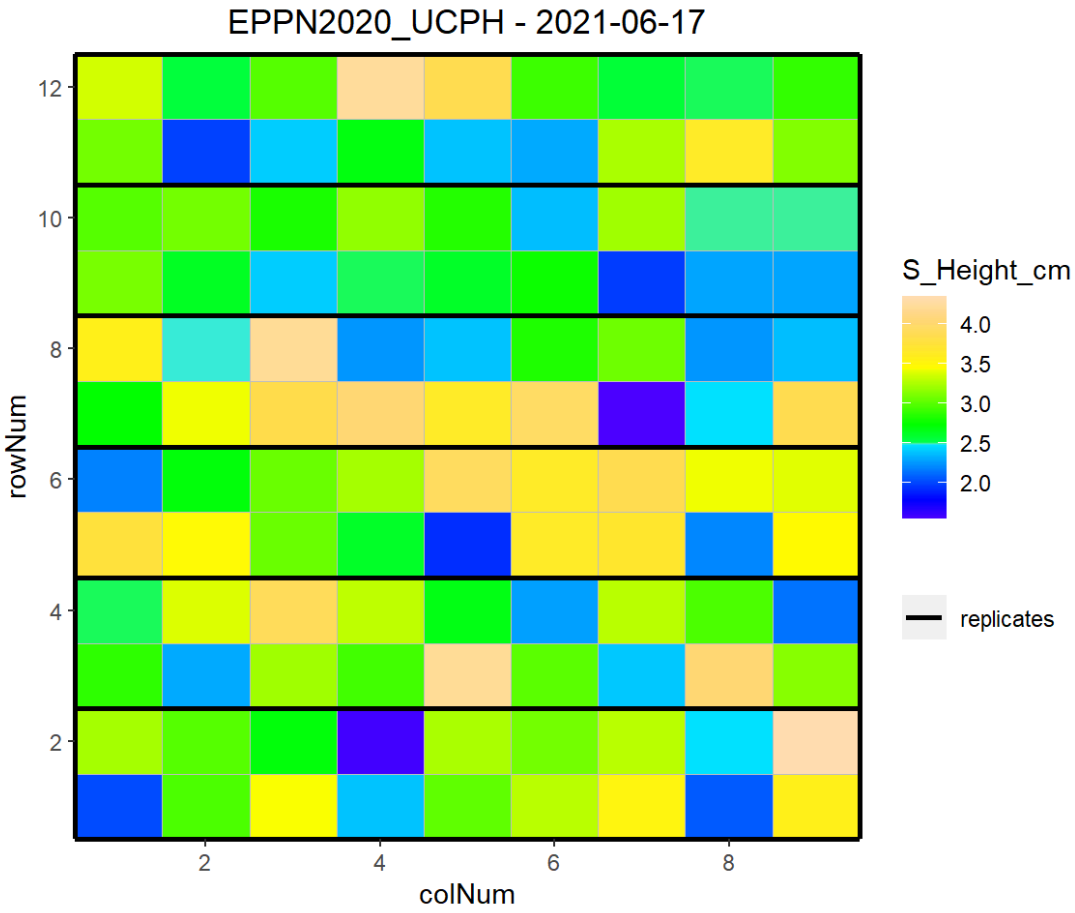
```
for (trait_name in traits) {
  single_outliers_name <- paste0("Single_outliers_", trait_name)

  if (exists(single_outliers_name)) {
    Single_outliers <- get(single_outliers_name)

    for (tp in 1:length(num_timepoints$timeNumber)) {
      plot(Single_outliers,
           plotType = "layout",
           timePoints = tp,
           traits = trait_name)
    }
  } else {
    cat("No object Single_outliers found for trait", trait_name, "\n")
  }
}
```

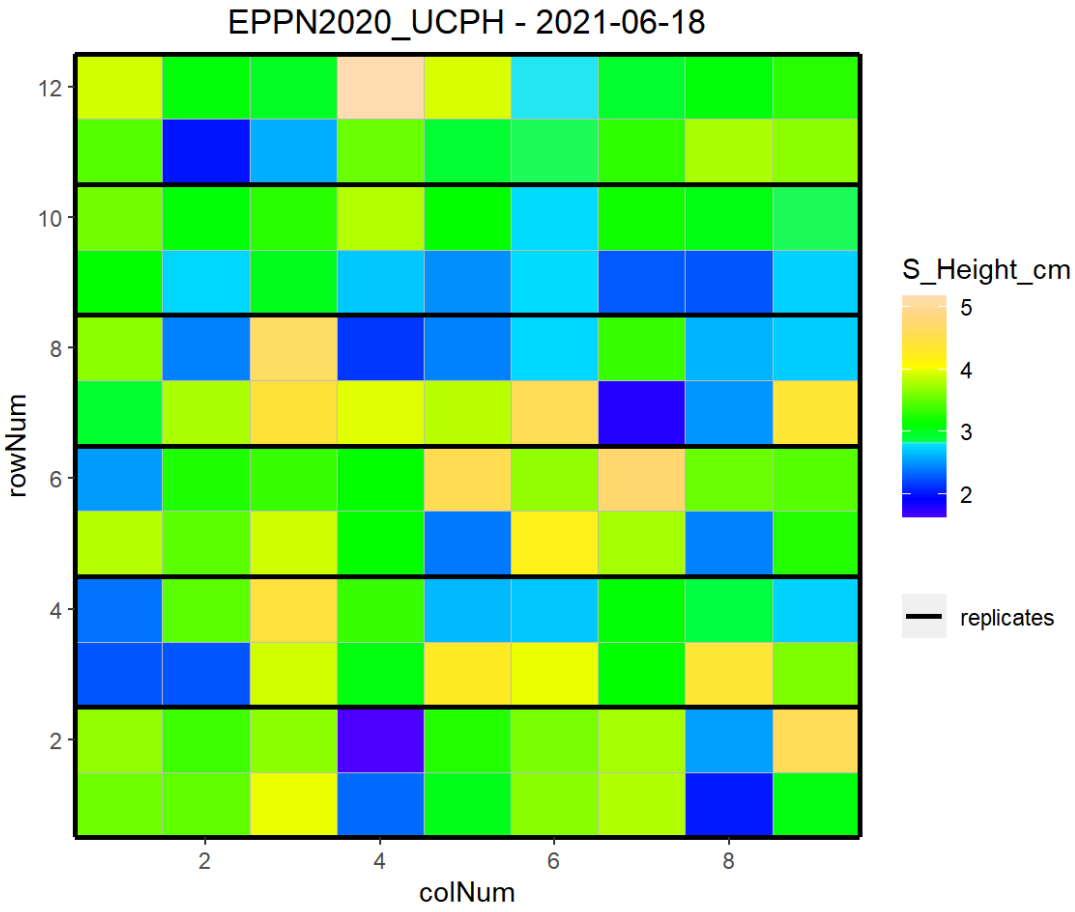

EPPN2020_UCPH - 2021-06-13
rowNum
colNum
S_Height_cm
2.5
2.0
1.5
replicates

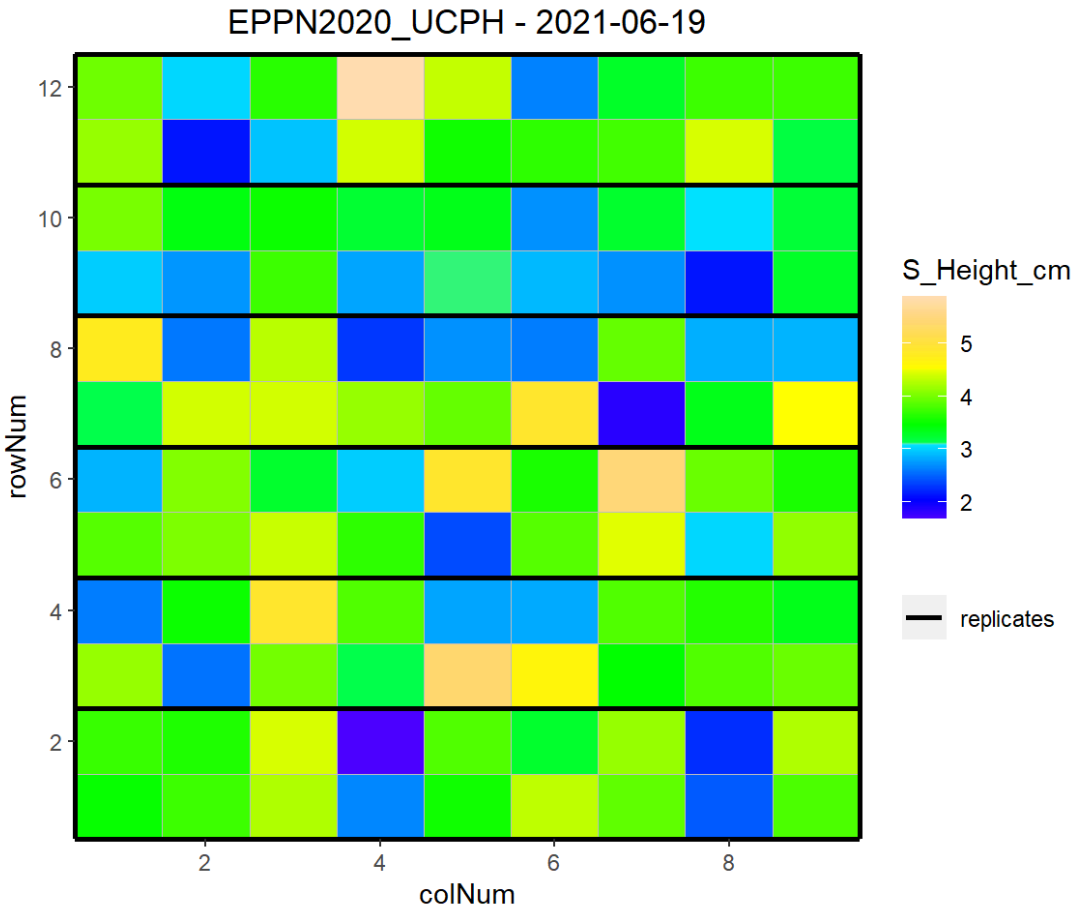

EPPN2020_UCPH - 2021-06-14
rowNum
colNum
S_Height_cm
3.0
2.5
2.0
1.5
replicates

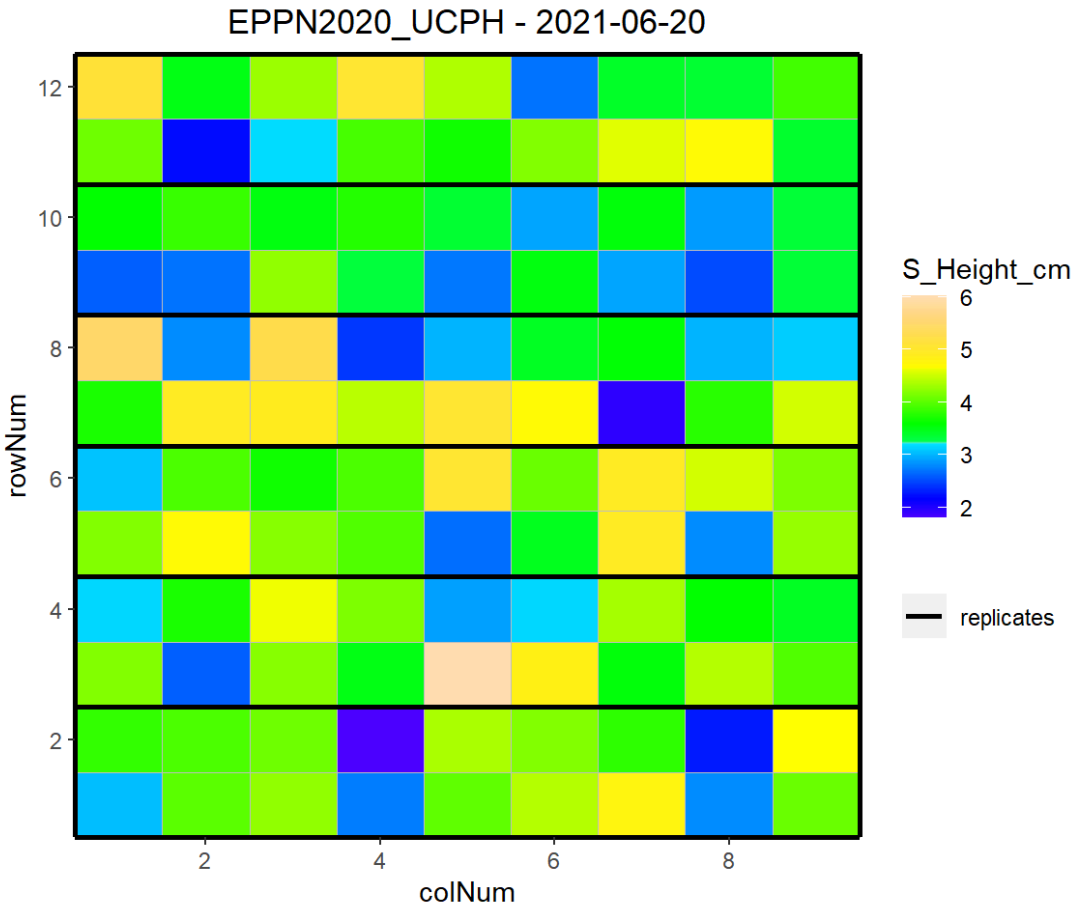

EPPN2020_UCPH - 2021-06-15
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
3.0
2.5
2.0
1.5

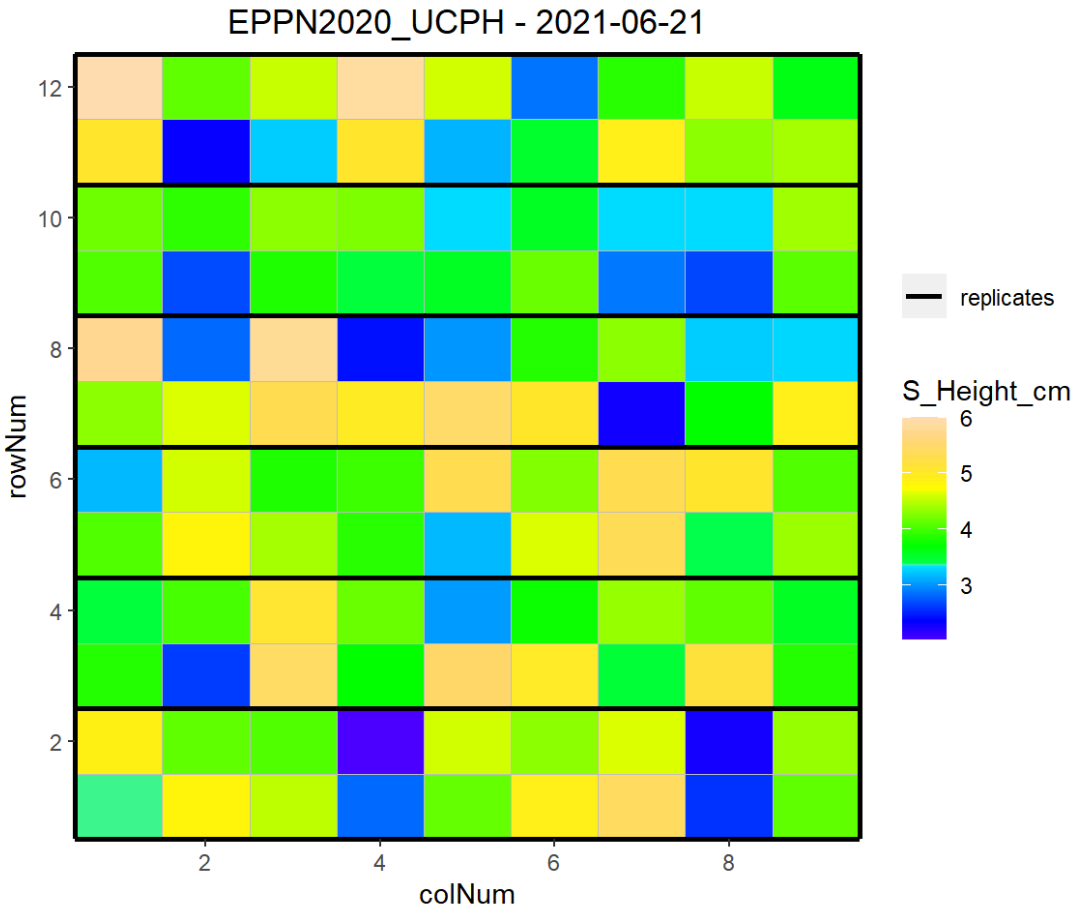

EPPN2020_UCPH - 2021-06-16
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
3.5
3.0
2.5
2.0
1.5
replicates

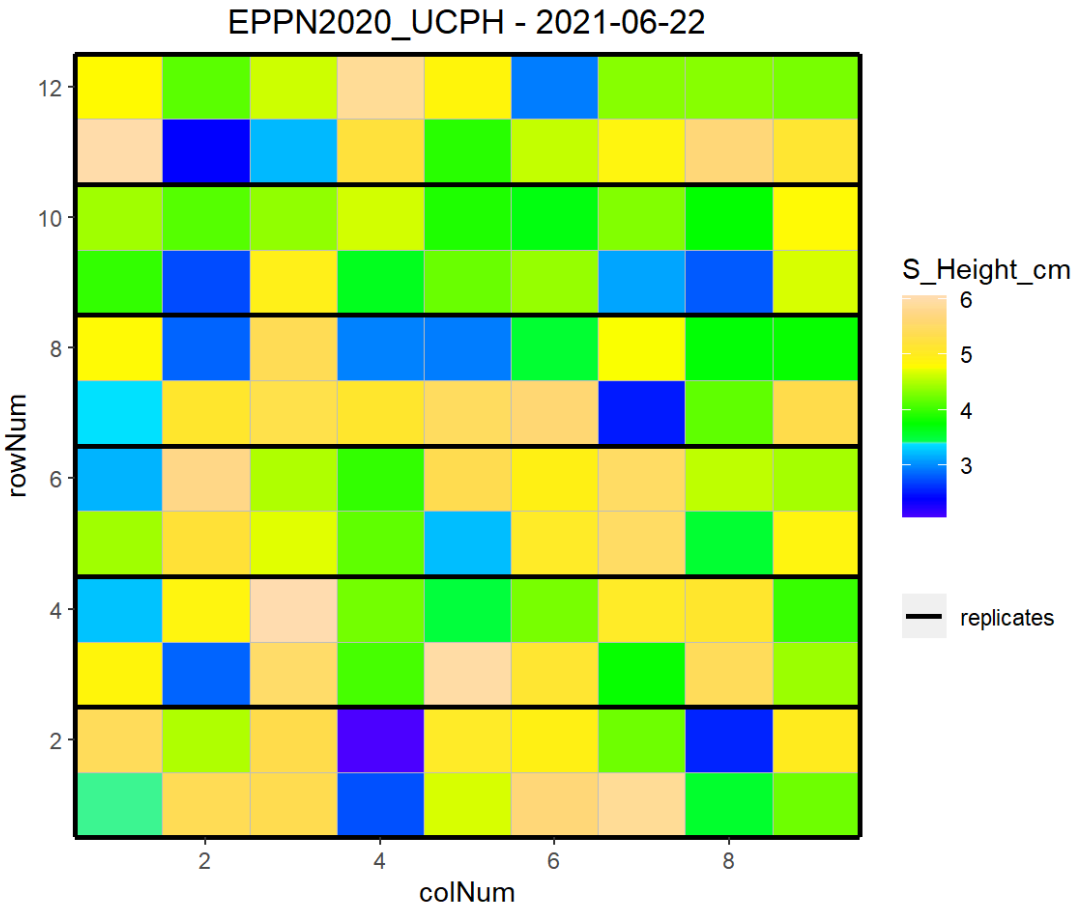

EPPN2020_UCPH - 2021-06-17
rowNum
colNum
S_Height_cm
4.0
3.5
3.0
2.5
2.0
replicates

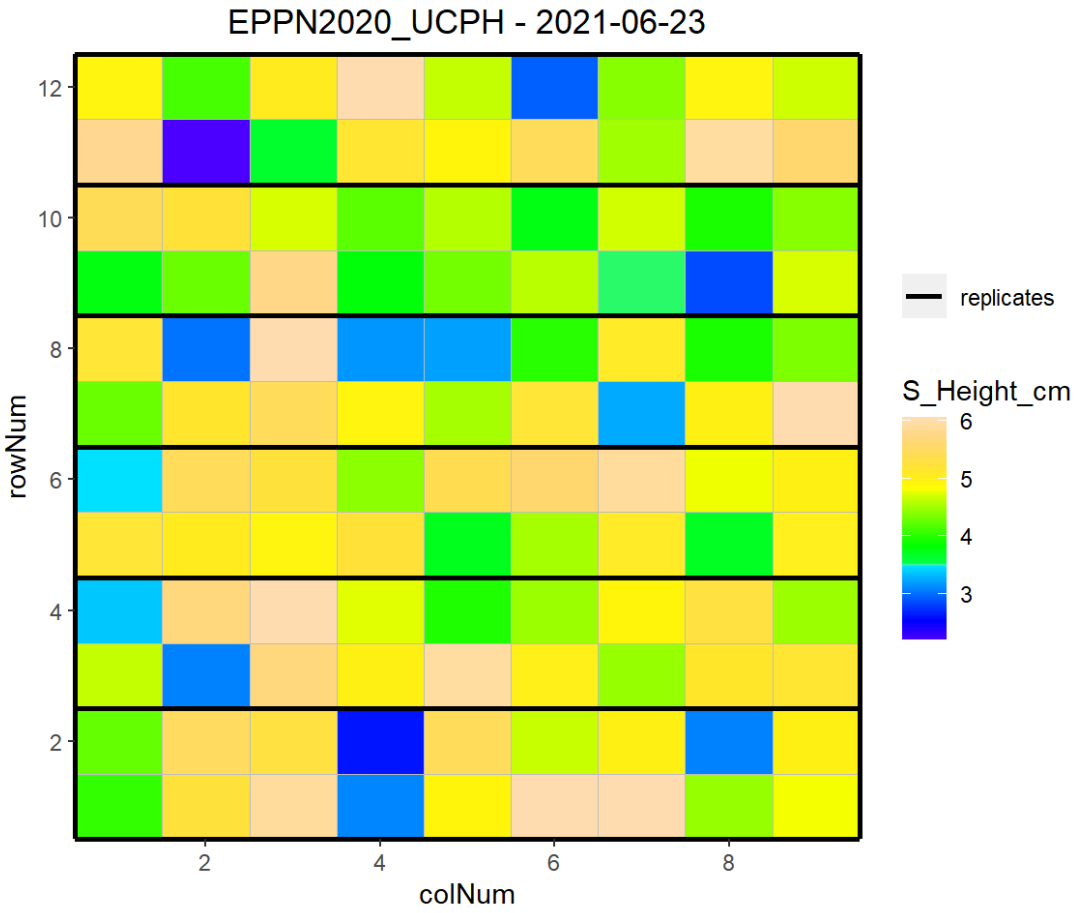

EPPN2020_UCPH - 2021-06-18
rowNum
colNum
S_Height_cm
5
4
3
2
replicates

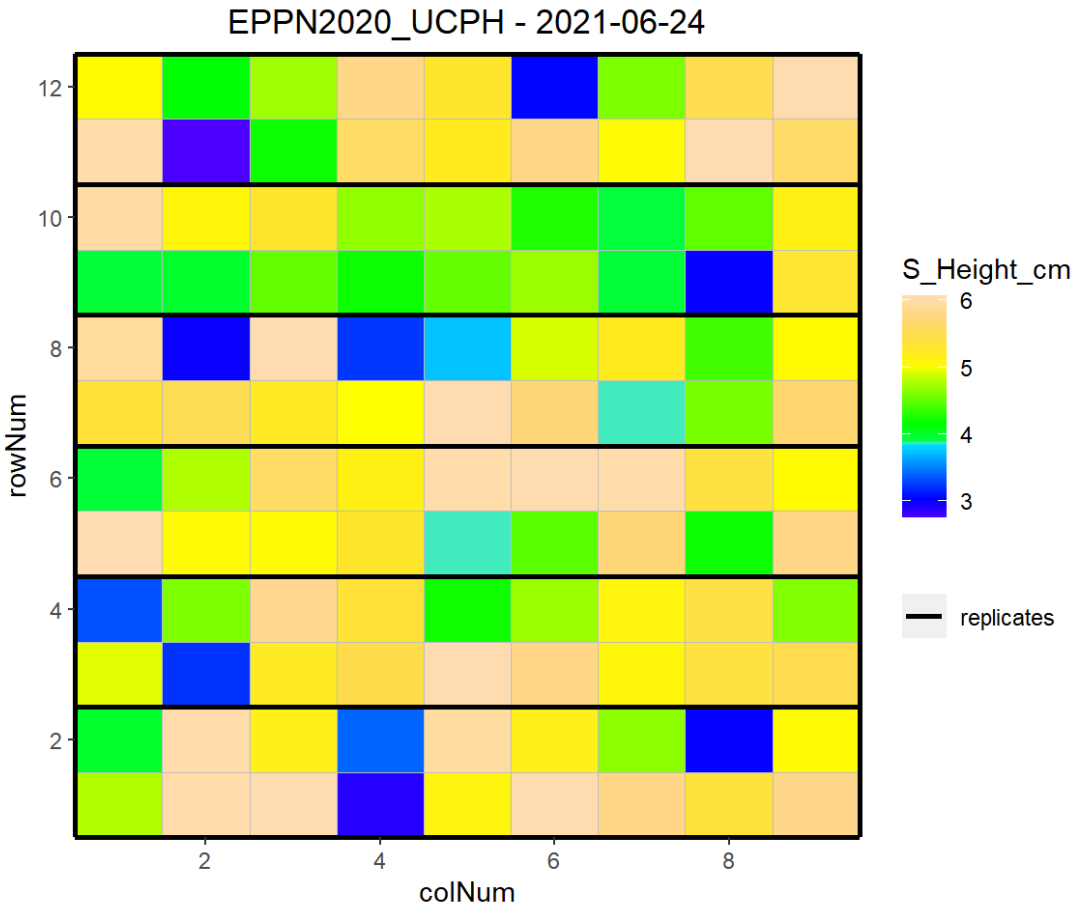

EPPN2020_UCPH - 2021-06-19
rowNum
colNum
S_Height_cm
5
4
3
2
replicates

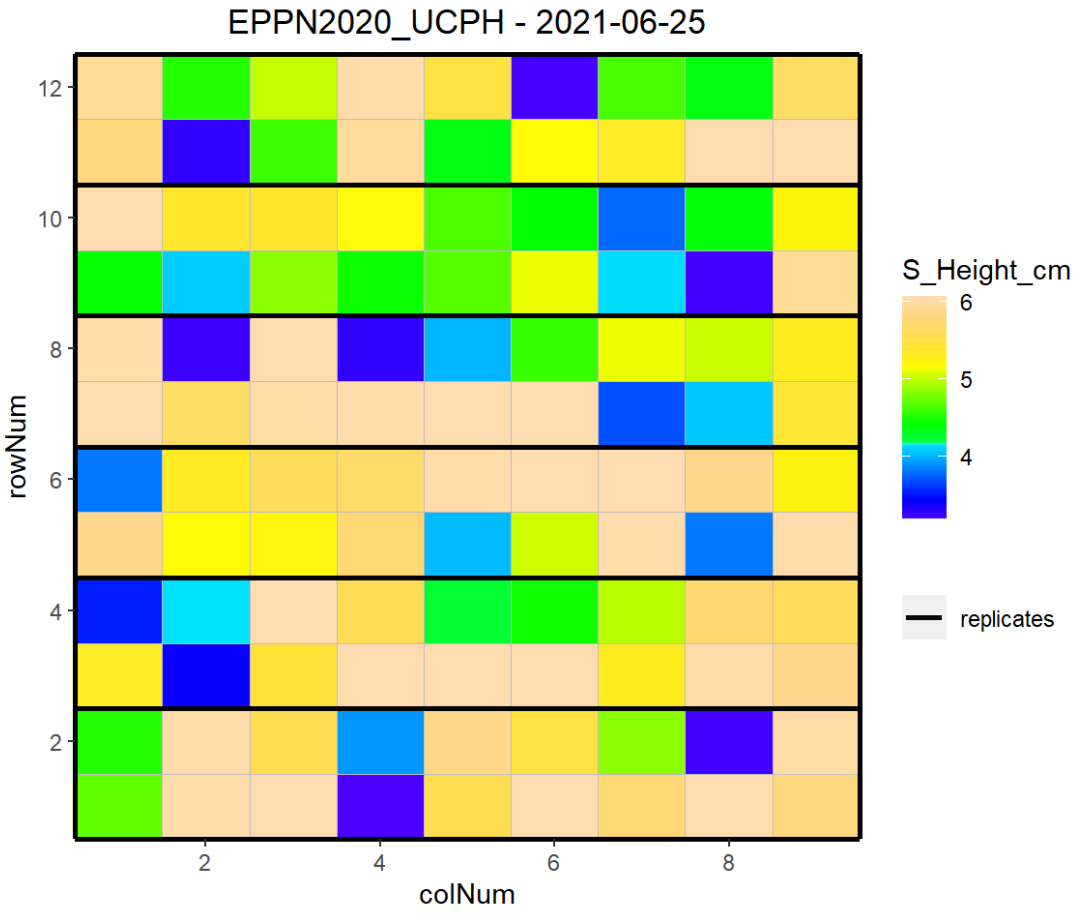

EPPN2020_UCPH - 2021-06-20
rowNum
colNum
S_Height_cm
6
5
4
3
2
replicates

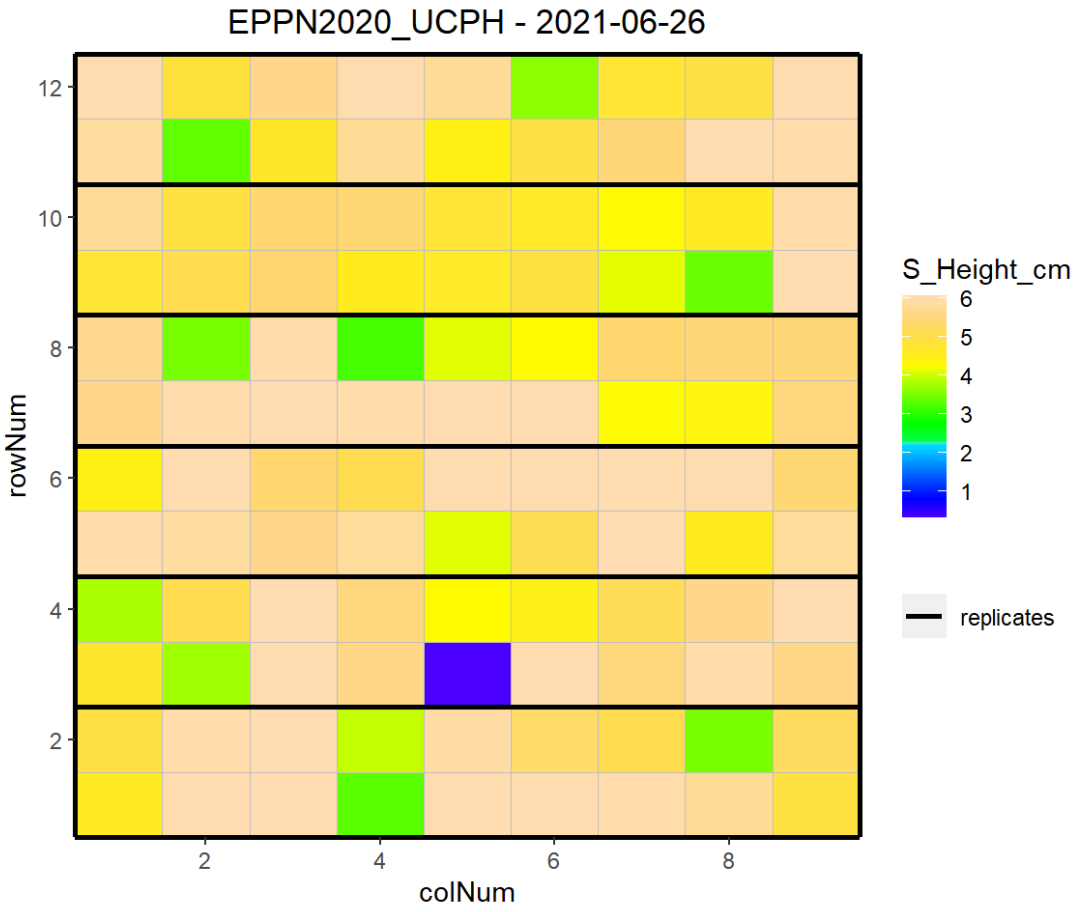

EPPN2020_UCPH - 2021-06-21
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
6
5
4
3

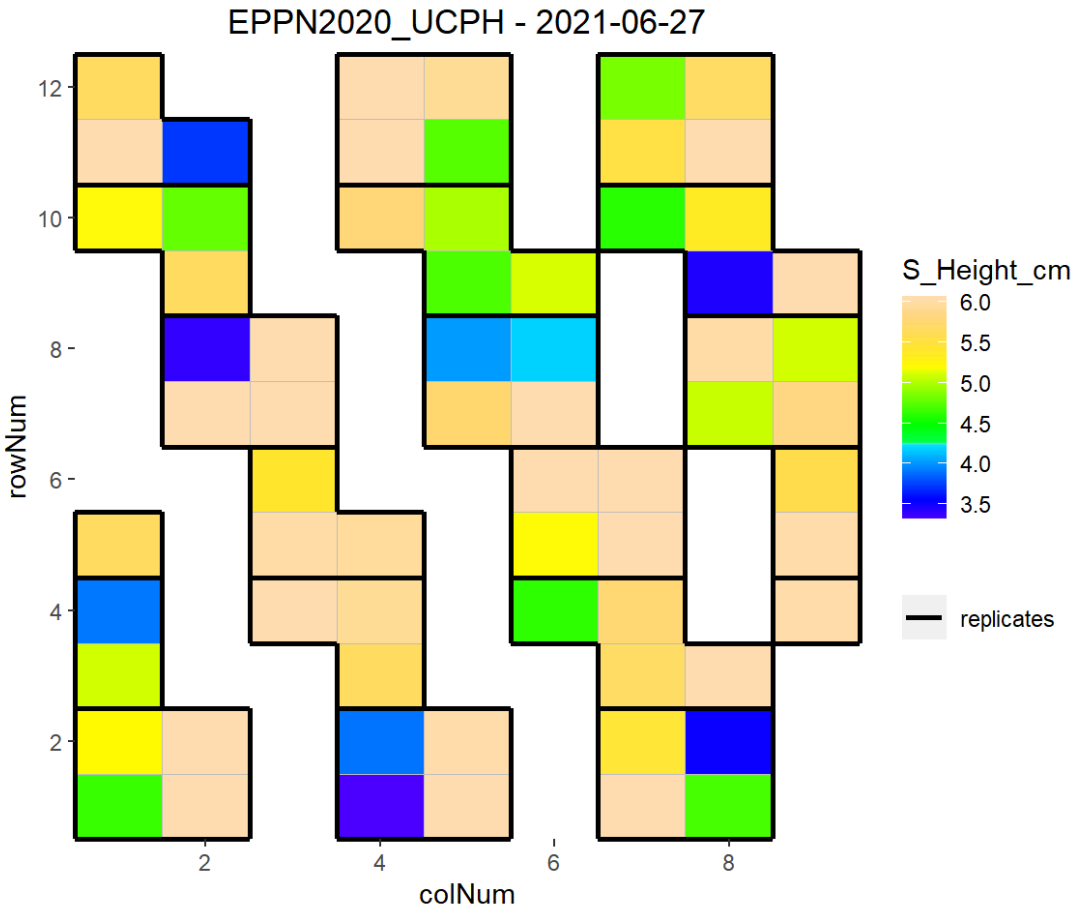

EPPN2020_UCPH - 2021-06-22
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
6
5
4
3
replicates

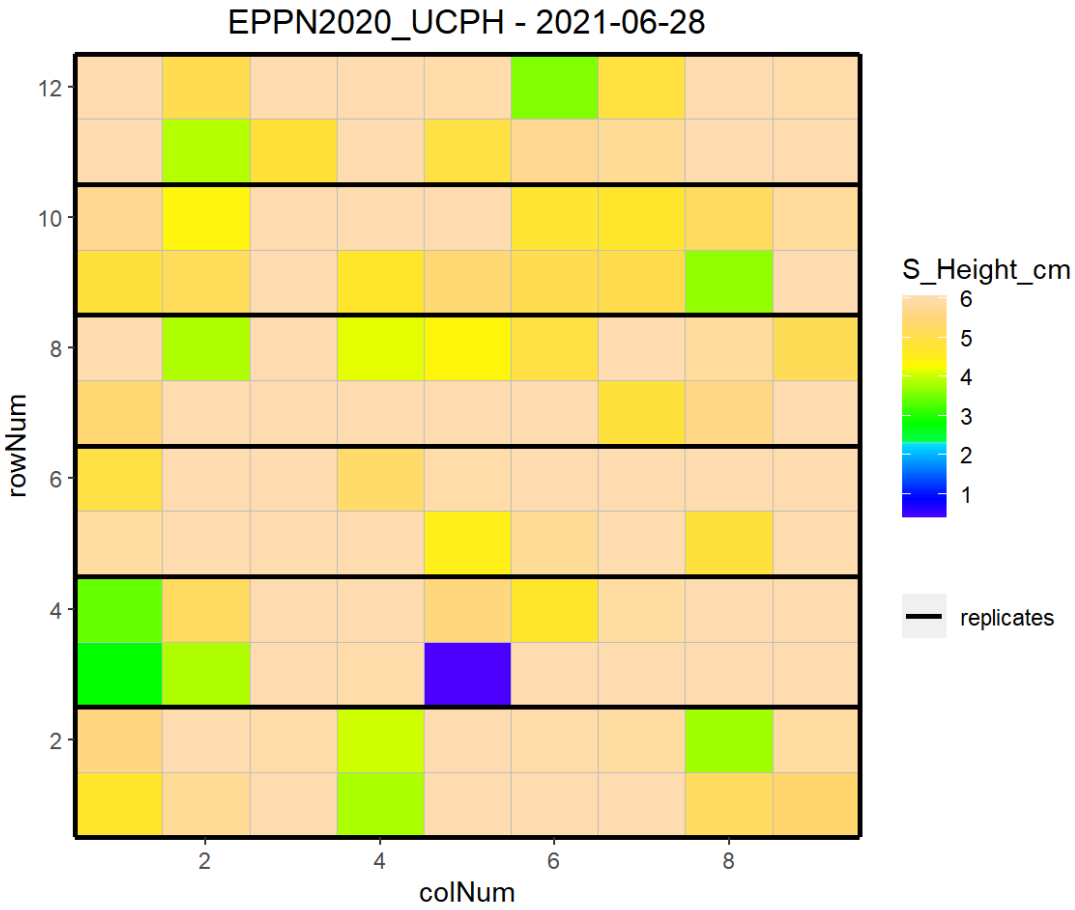

EPPN2020_UCPH - 2021-06-23
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
replicates
S_Height_cm
6
5
4
3

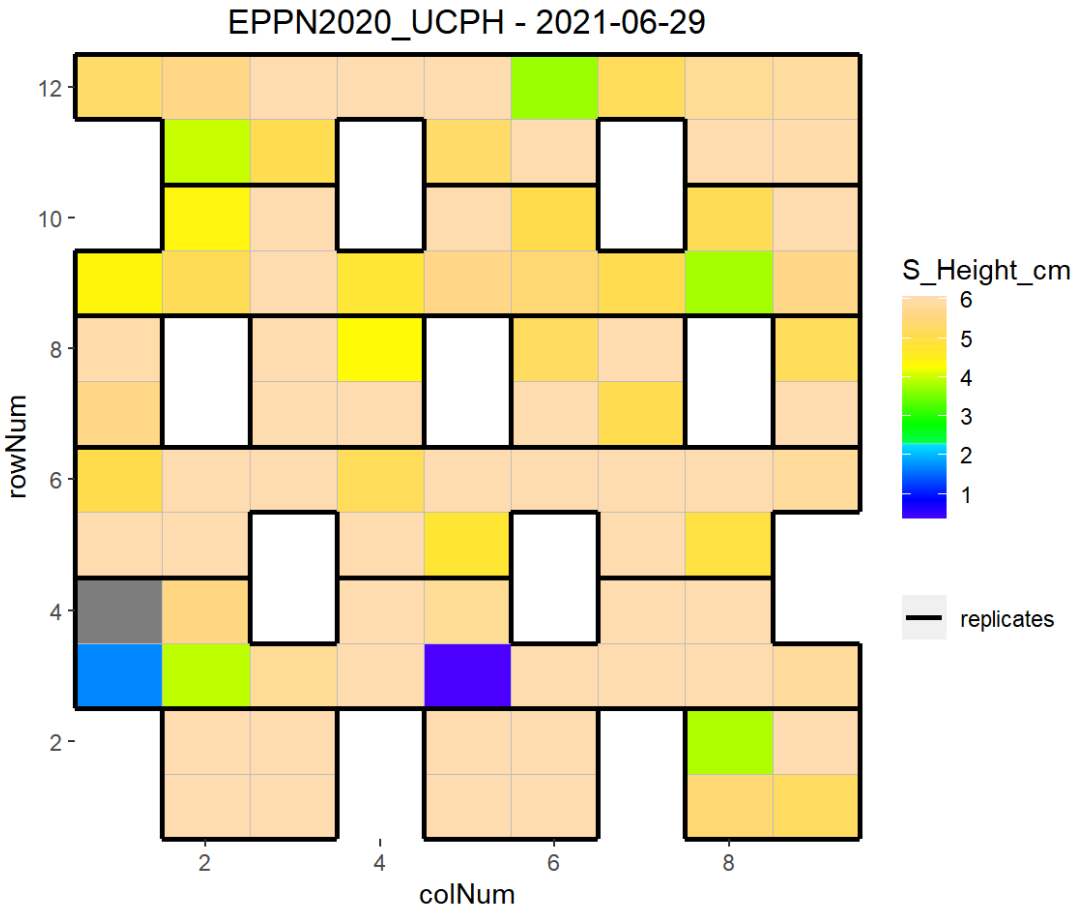

EPPN2020_UCPH - 2021-06-24
rowNum
colNum
12
10
8
6
4
2
2
4
6
8
S_Height_cm
6
5
4
3
replicates


EPPN2020_UCPH - 2021-06-25
rowNum
colNum
S_Height_cm
6
5
4
replicates


EPPN2020_UCPH - 2021-06-26
rowNum
colNum
S_Height_cm
6
5
4
3
2
1
replicates

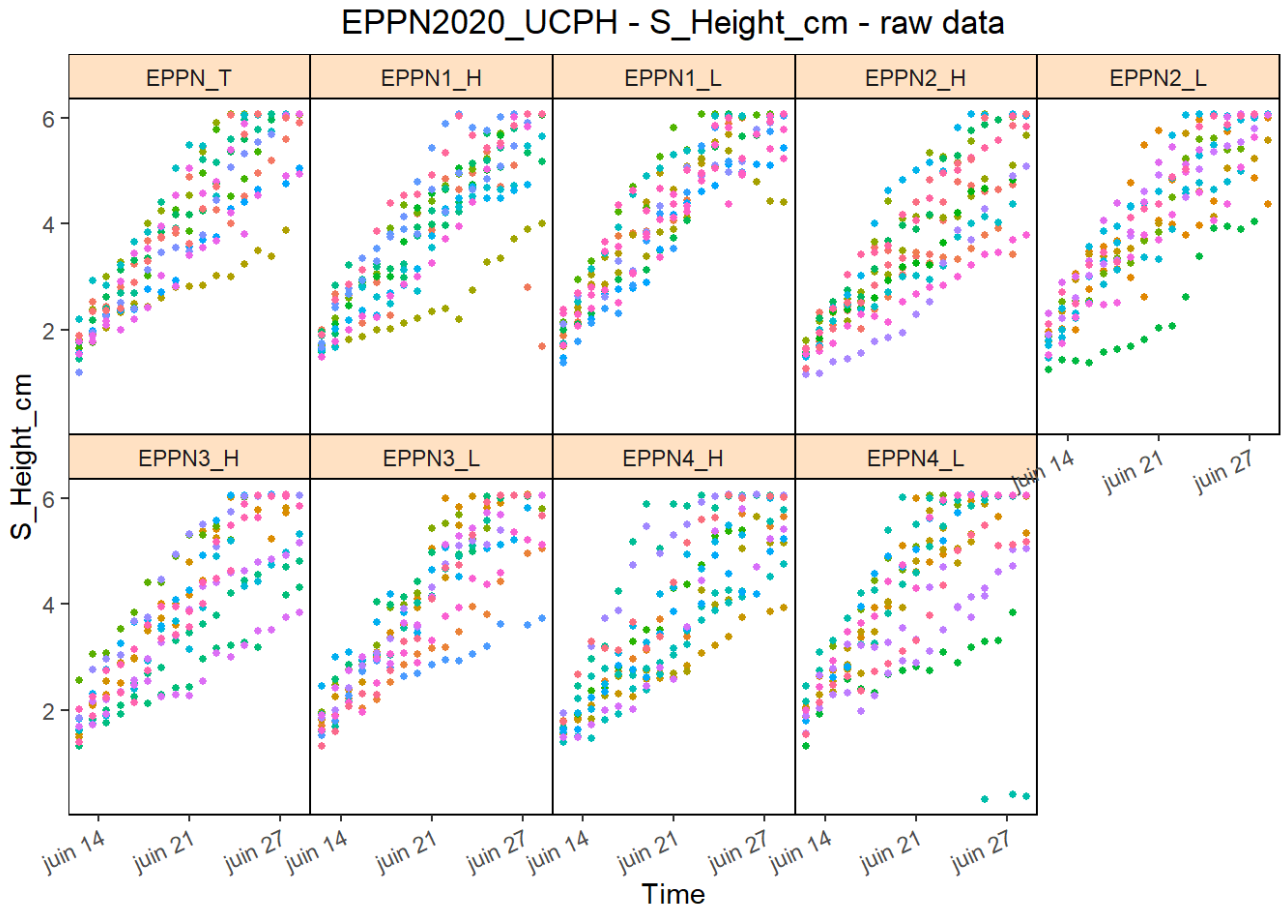

EPPN2020_UCPH - 2021-06-27
rowNum
colNum
S_Height_cm
6.0
5.5
5.0
4.5
4.0
3.5
replicates

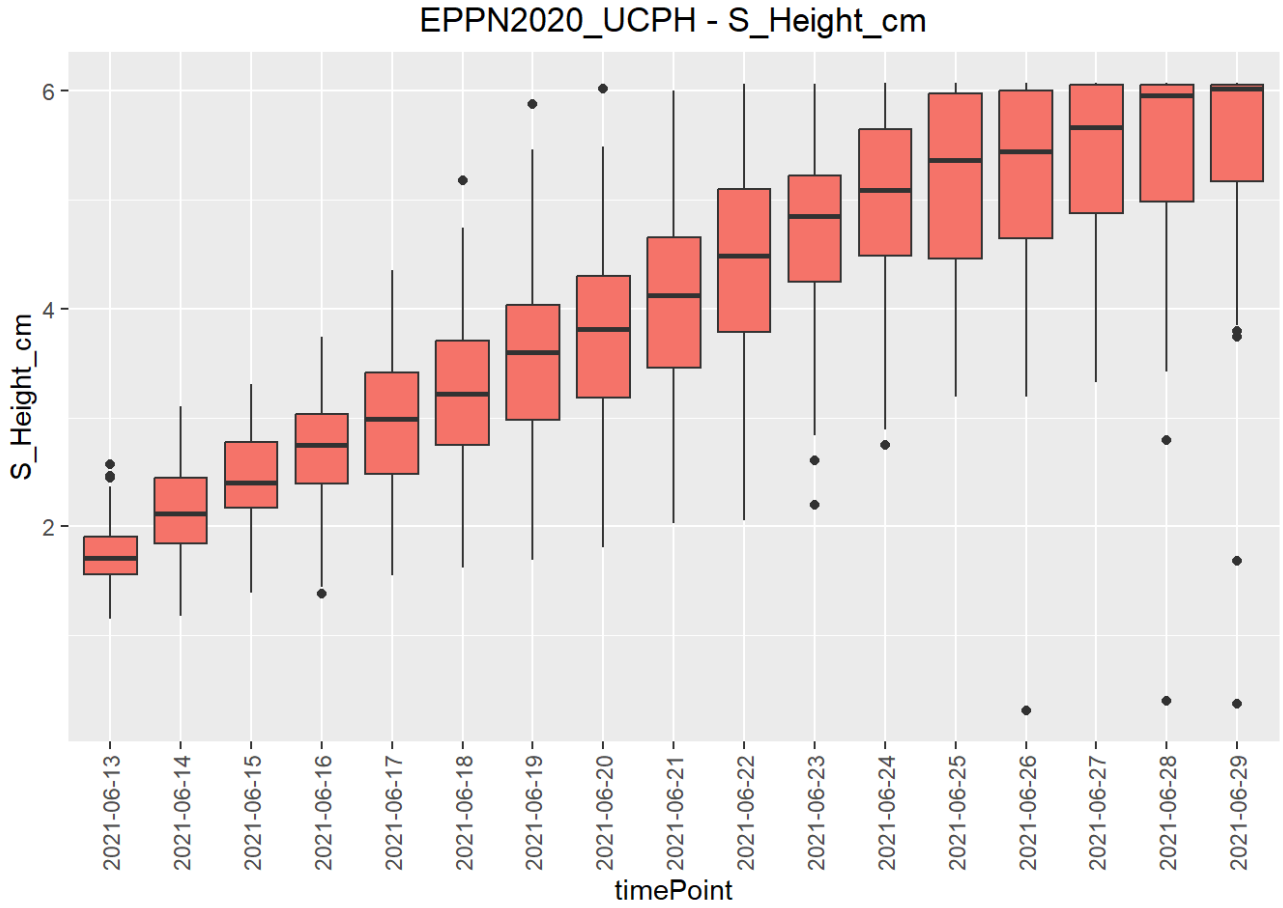

EPPN2020_UCPH - 2021-06-28
rowNum
colNum
S_Height_cm
6
5
4
3
2
1
replicates

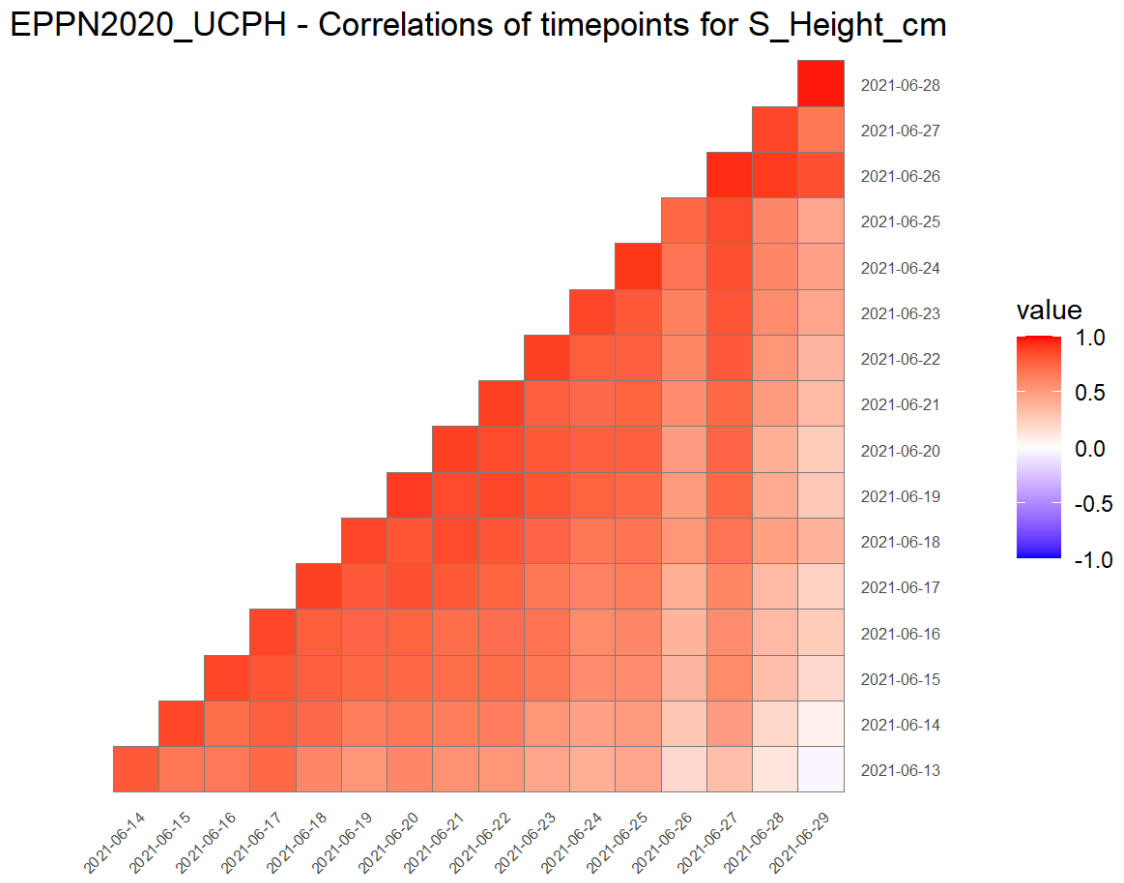


Time course, boxplots and correlation plots of data

```
for (trait_name in traits) {
  single_outliers_name <- paste0("Single_outliers_", trait_name)

  if (exists(single_outliers_name)) {
    Single_outliers <- get(single_outliers_name)

    plot(Single_outliers,
         traits = trait_name,
         plotType = "raw")

    plot(Single_outliers,
         plotType = "box",
         traits = trait_name)

    plot(Single_outliers,
         plotType = "cor",
         traits = trait_name)

  } else {
    cat("No Single_outliers object found for trait", trait_name, "\n")
  }
}
```


EPPN2020_UCPH - S_Height_cm - raw data
EPPN_T
EPPN1_H
EPPN1_L
EPPN2_H
EPPN2_L
EPPN3_H
EPPN3_L
EPPN4_H
EPPN4_L
S_Height_cm
Time
juin 14
juin 21
juin 27


EPPN2020_UCPH - S_Height_cm
S_Height_cm
timePoint
2021-06-13
2021-06-14
2021-06-15
2021-06-16
2021-06-17
2021-06-18
2021-06-19
2021-06-20
2021-06-21
2021-06-22
2021-06-23
2021-06-24
2021-06-25
2021-06-26
2021-06-27
2021-06-28
2021-06-29



2. Correction for spatial trends

Fit a model for all time points with no extra fixed effects.

```
for (trait_name in traits) {
  single_outliers_name <- paste0("Single_outliers_", trait_name)

  if (exists(single_outliers_name)) {
    Single_outliers <- get(single_outliers_name)

    assign(paste0("modTP_", trait_name),
           fitModels(TP = Single_outliers,
                     trait = trait_name,
                     geno.decomp = "Plant_type"))
  } else {
    assign(paste0("modTP_", trait_name),
           fitModels(TP = timePoint_S,
                     trait = trait_name,
                     geno.decomp = "Plant_type"))
  }
}
```

```
## 2021-06-13
```

```
## 2021-06-14
```

```
## 2021-06-15
```

```
## 2021-06-16
```

```
## 2021-06-17
```

```
## 2021-06-18
```

```
## 2021-06-19
```

```
## 2021-06-20
```

```
## 2021-06-21
```

```
## 2021-06-22
```

```
## 2021-06-23
```

```
## 2021-06-24
```

```
## 2021-06-25
```

```
## 2021-06-26
```

```
## 2021-06-27
```

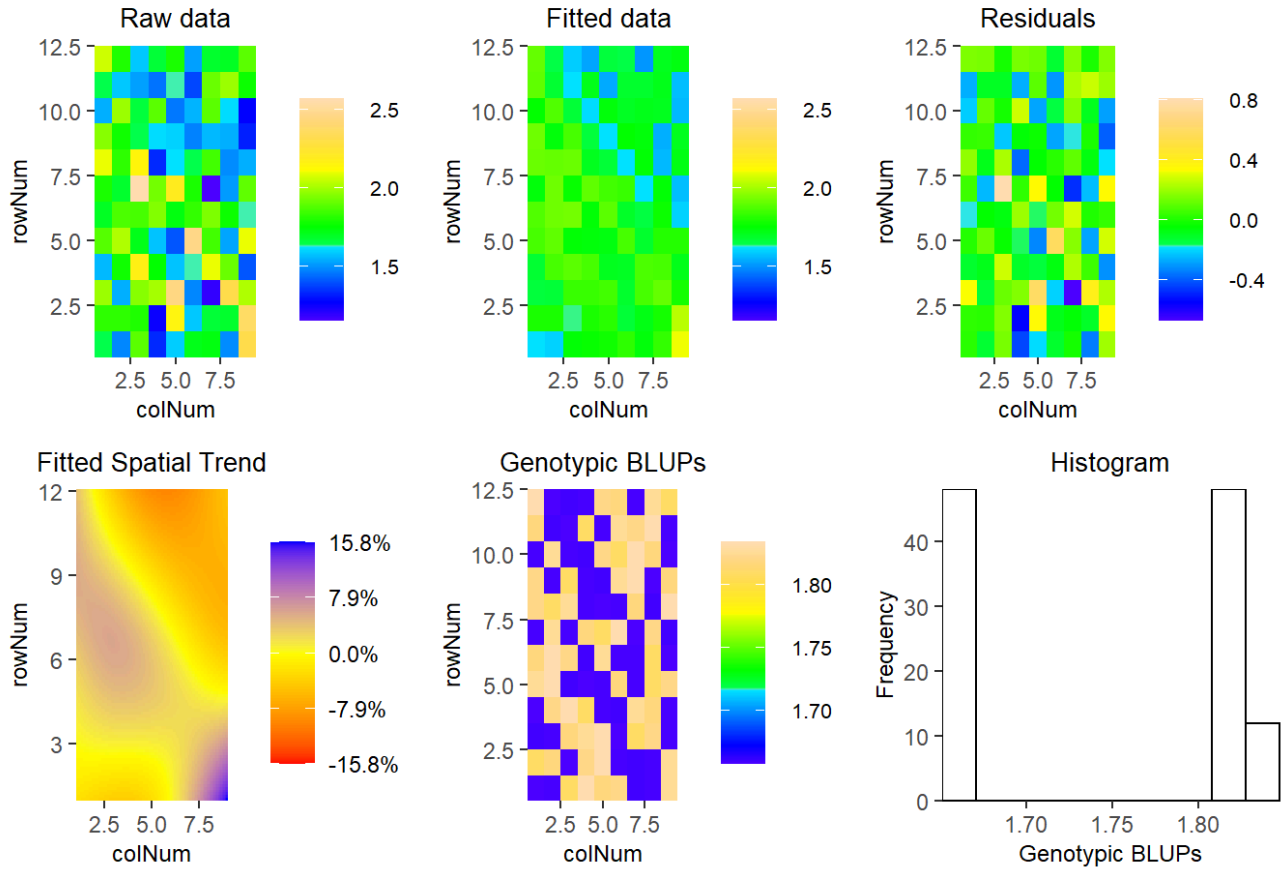
```
## 2021-06-28
```

```
## 2021-06-29
```

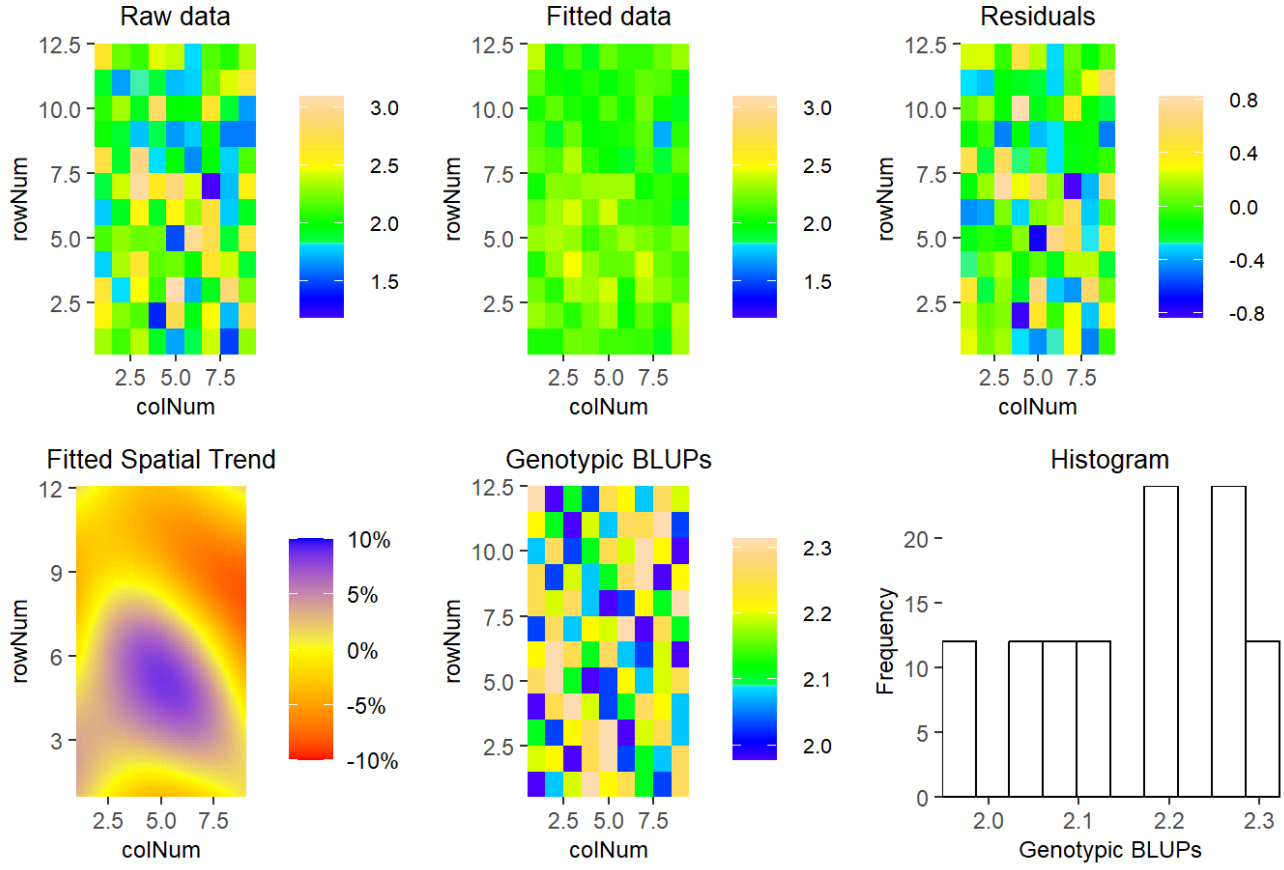
Model visualisation

```
for (trait_name in traits) {
  mod_name <- paste0("modTP_", trait_name)

  if (exists(mod_name)) {
    mod <- get(mod_name)

    for (tp in 1:length(num_timepoints$timeNumber)) {
      plot(mod,
           timePoints = tp,
           plotType = "spatial",
           spaTrend = "percentage")
    }

    gif_file <- sprintf("%s/%s_mod.gif", datadir, trait_name)

    plot(mod,
         plotType = "timeLapse",
         spaTrend = "percentage",
         outFile = gif_file)
  } else {
    cat("No model found for", trait_name, "\n")
  }
}
```

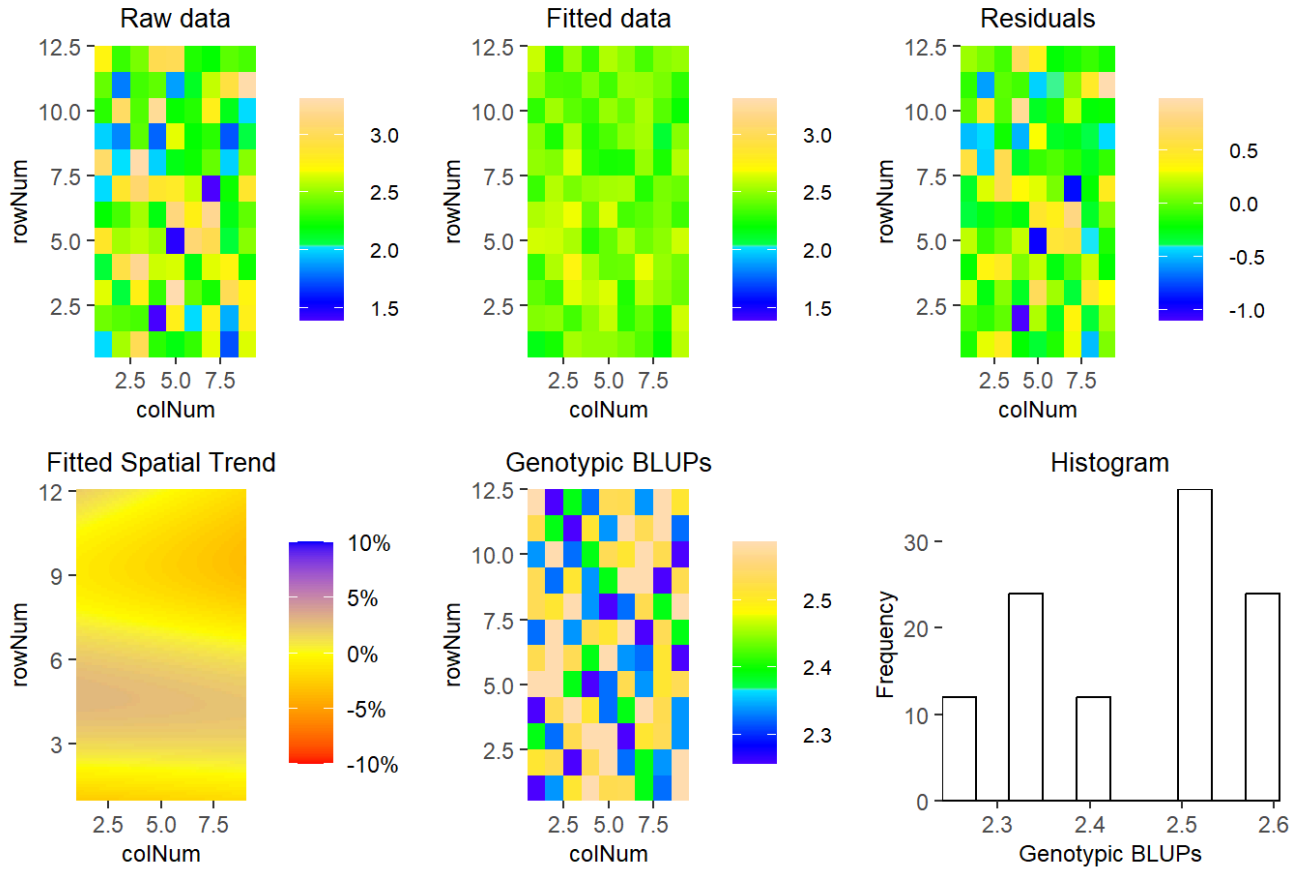
EPPN2020_UCPH - S_Height_cm - 2021-06-13



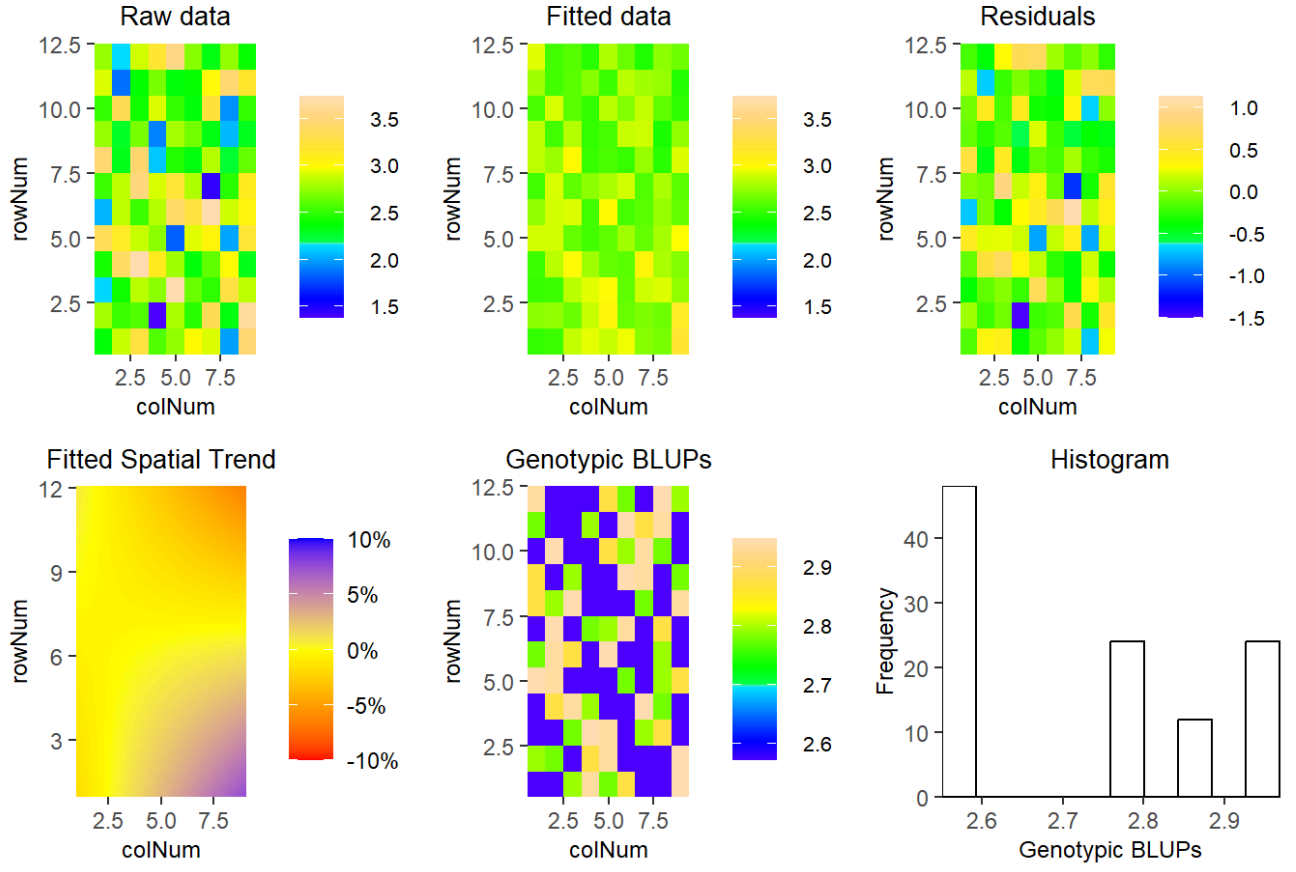
EPPN2020_UCPH - S_Height_cm - 2021-06-14



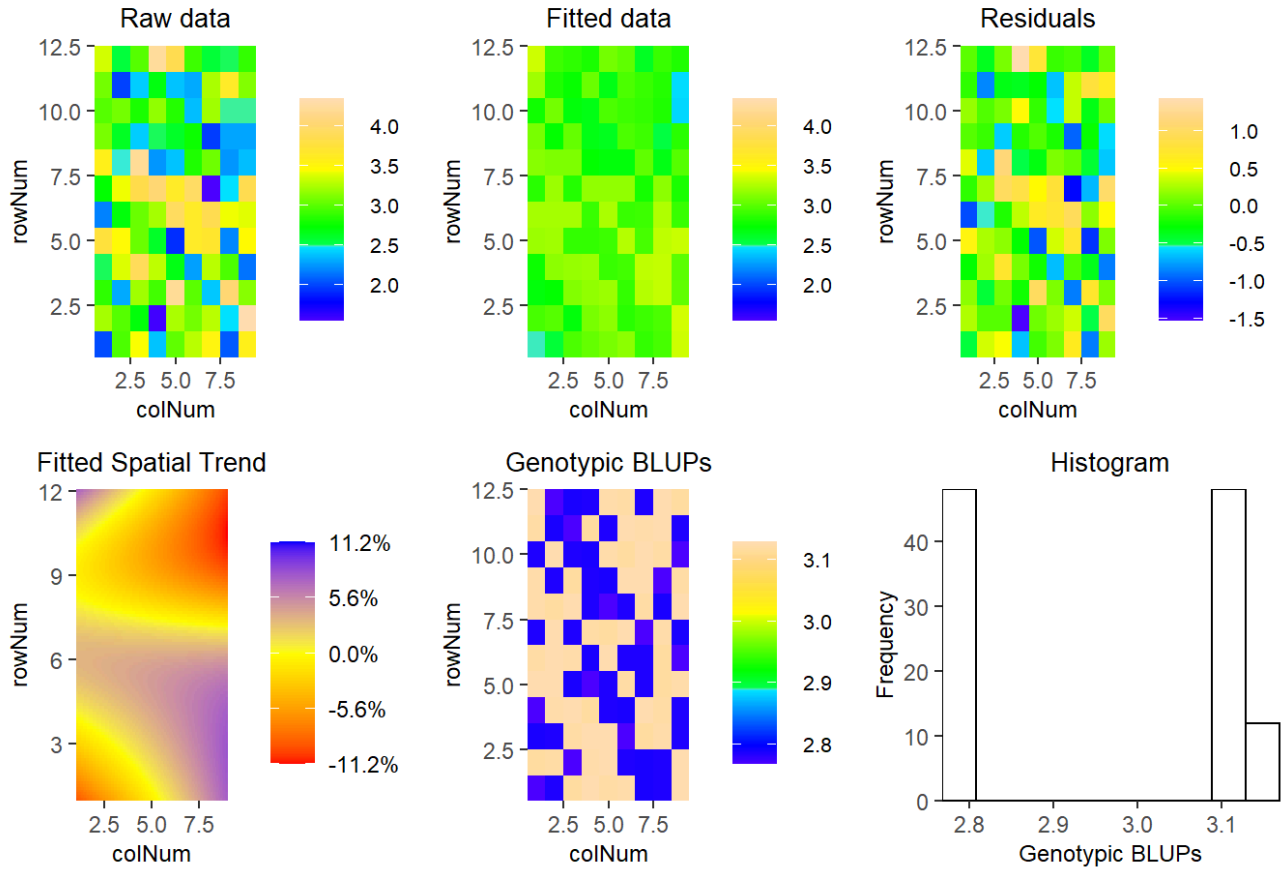
EPPN2020_UCPH - S_Height_cm - 2021-06-15



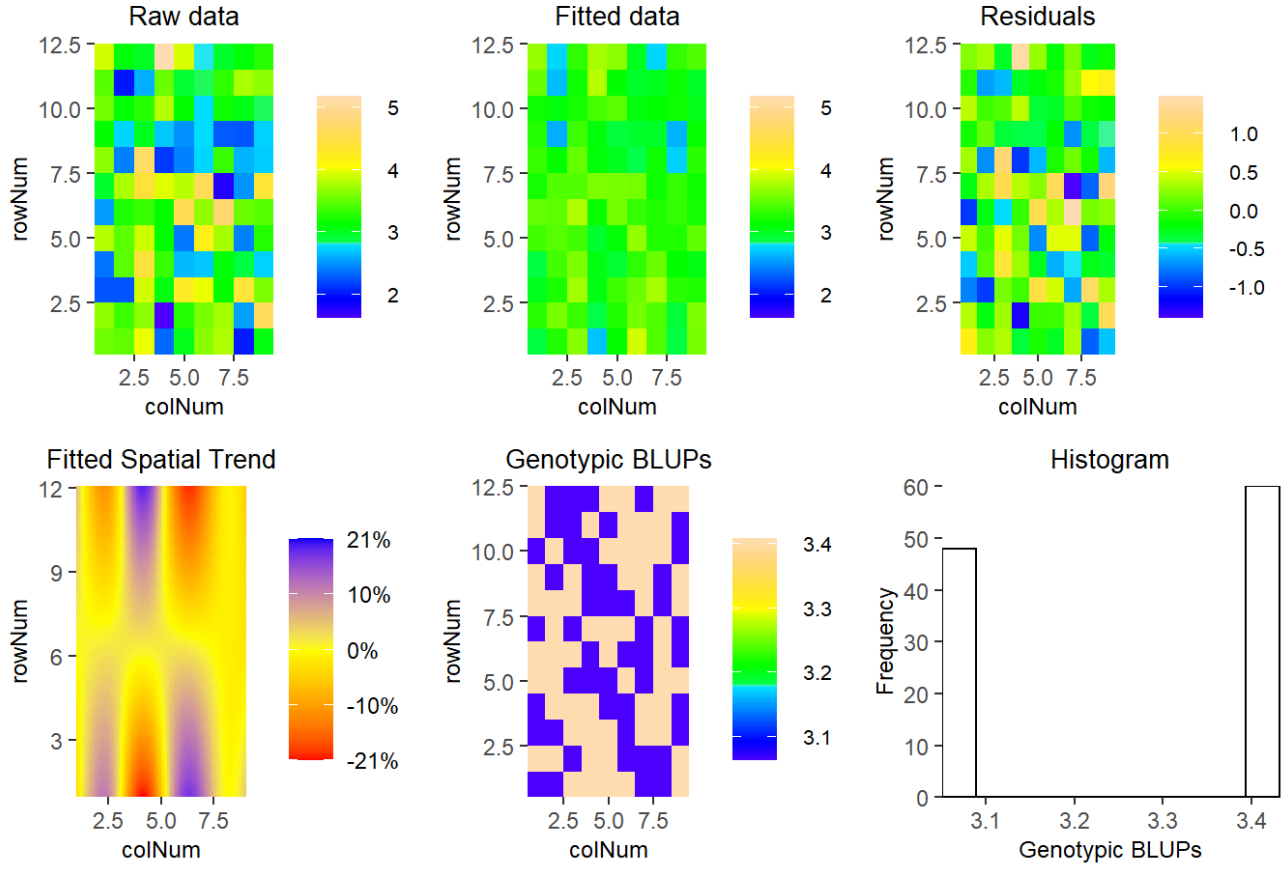
EPPN2020_UCPH - S_Height_cm - 2021-06-16



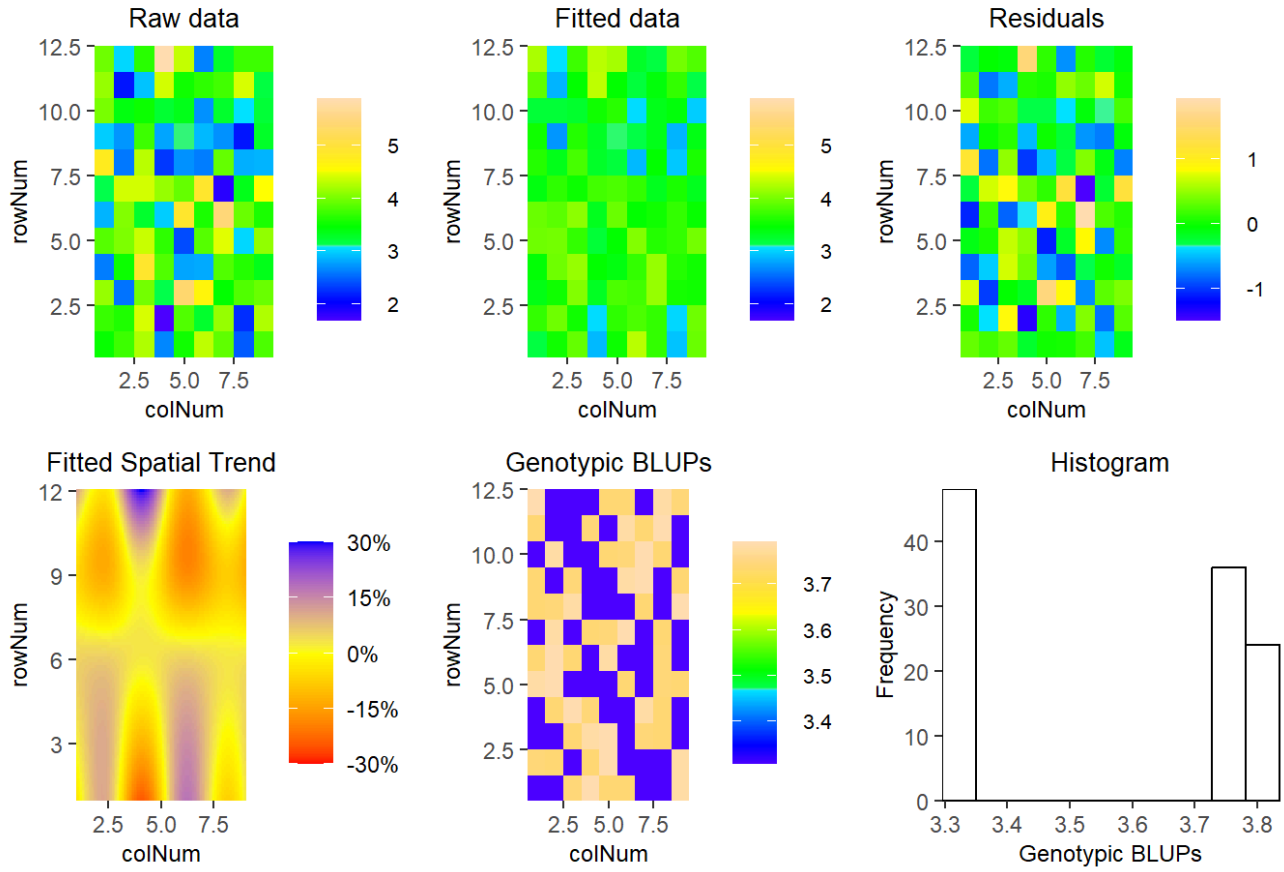
EPPN2020_UCPH - S_Height_cm - 2021-06-17



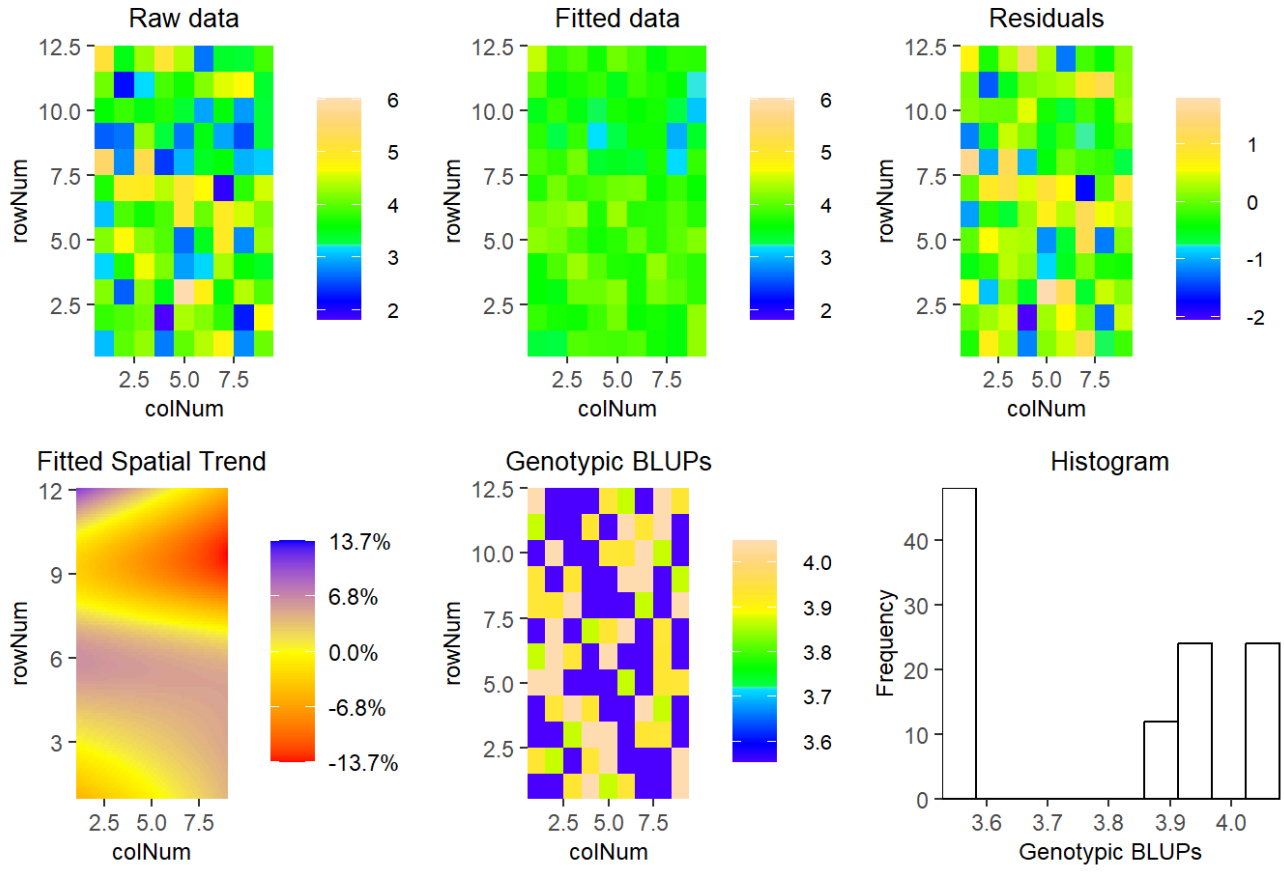
EPPN2020_UCPH - S_Height_cm - 2021-06-18



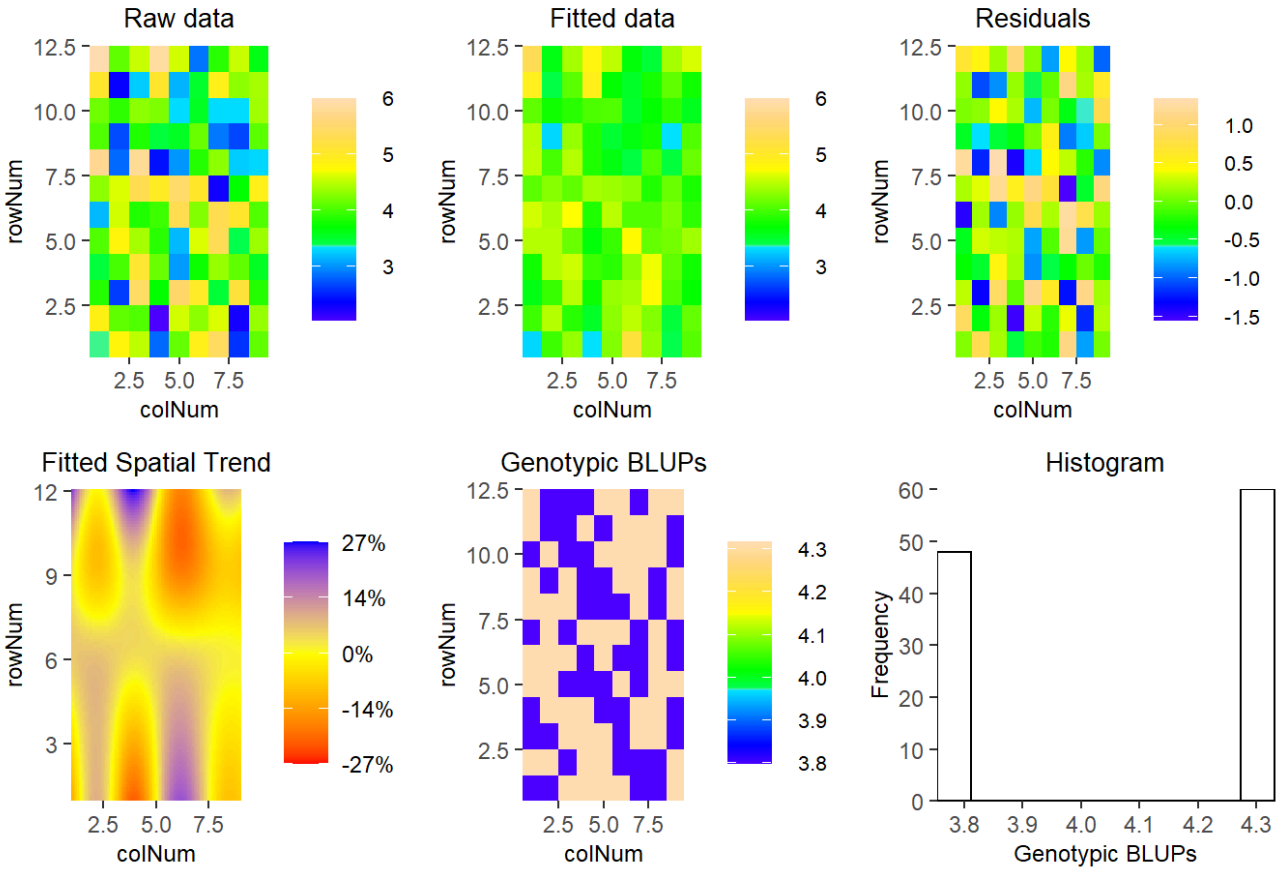
EPPN2020_UCPH - S_Height_cm - 2021-06-19



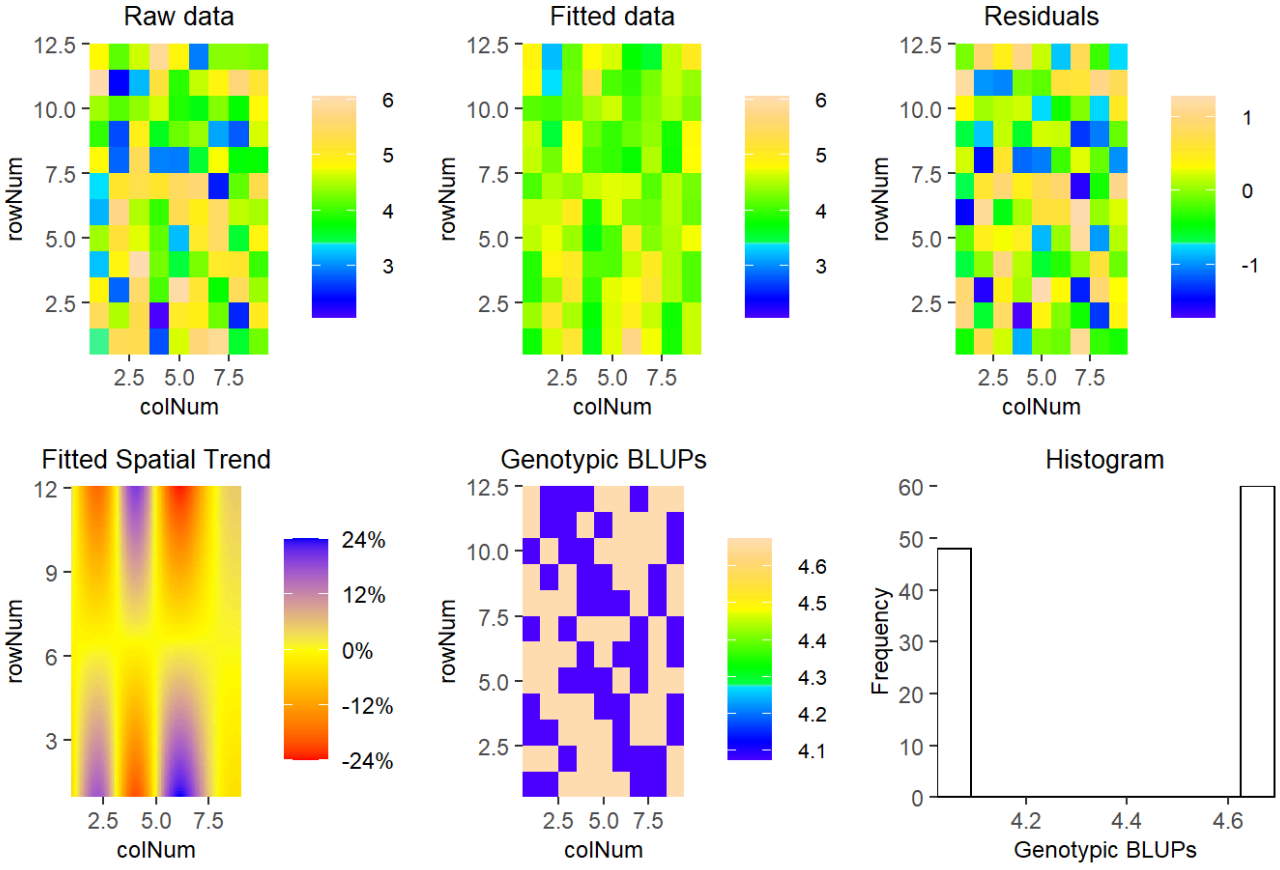
EPPN2020_UCPH - S_Height_cm - 2021-06-20



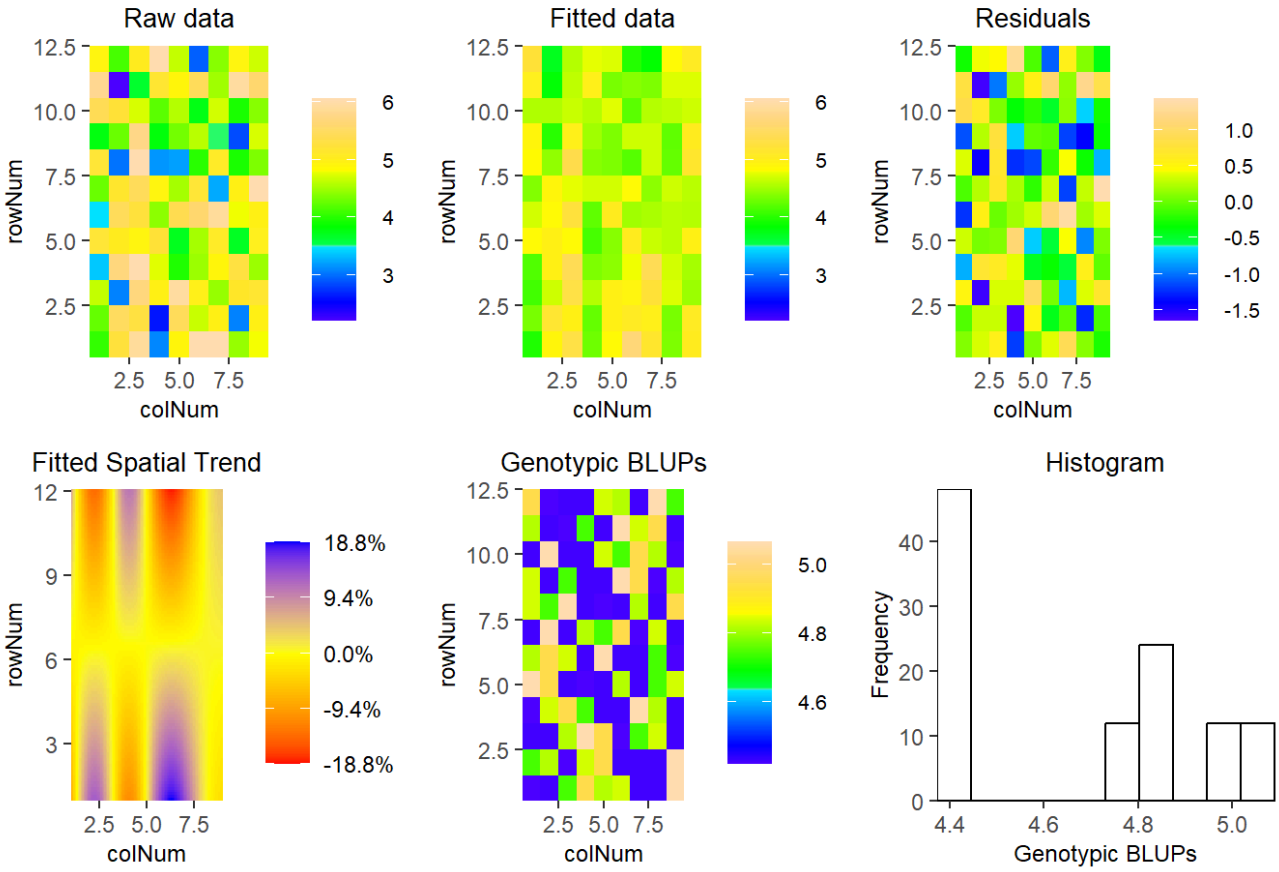
EPPN2020_UCPH - S_Height_cm - 2021-06-21



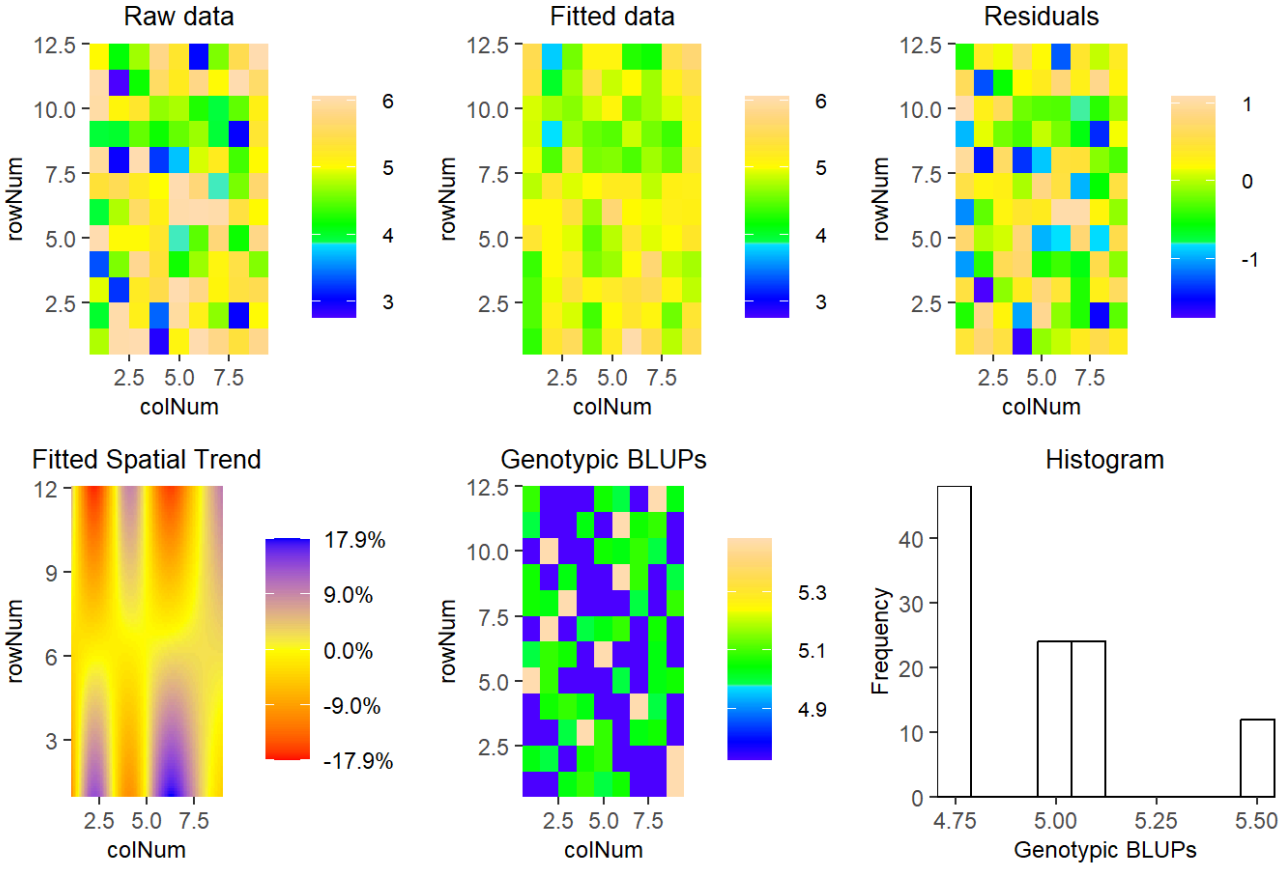
EPPN2020_UCPH - S_Height_cm - 2021-06-22



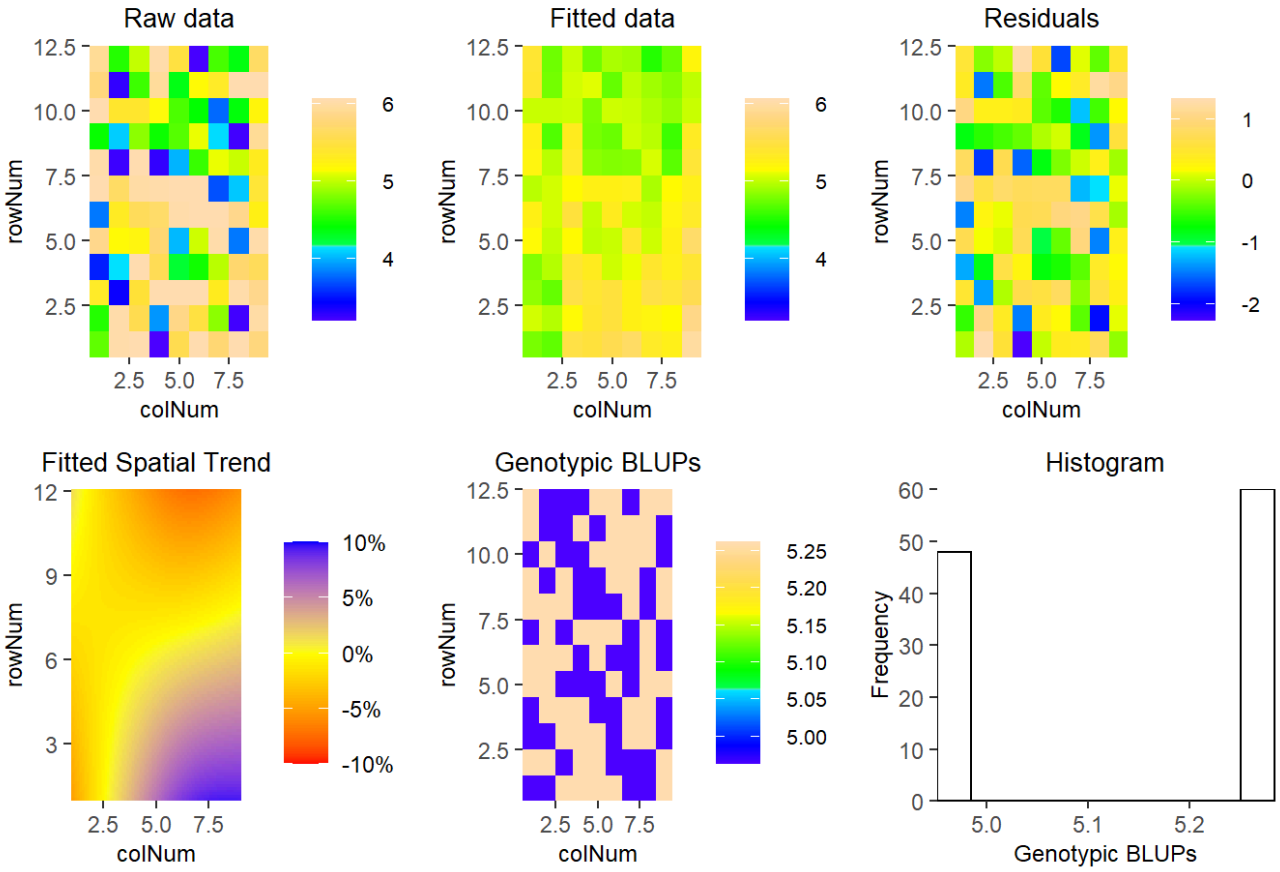
EPPN2020_UCPH - S_Height_cm - 2021-06-23



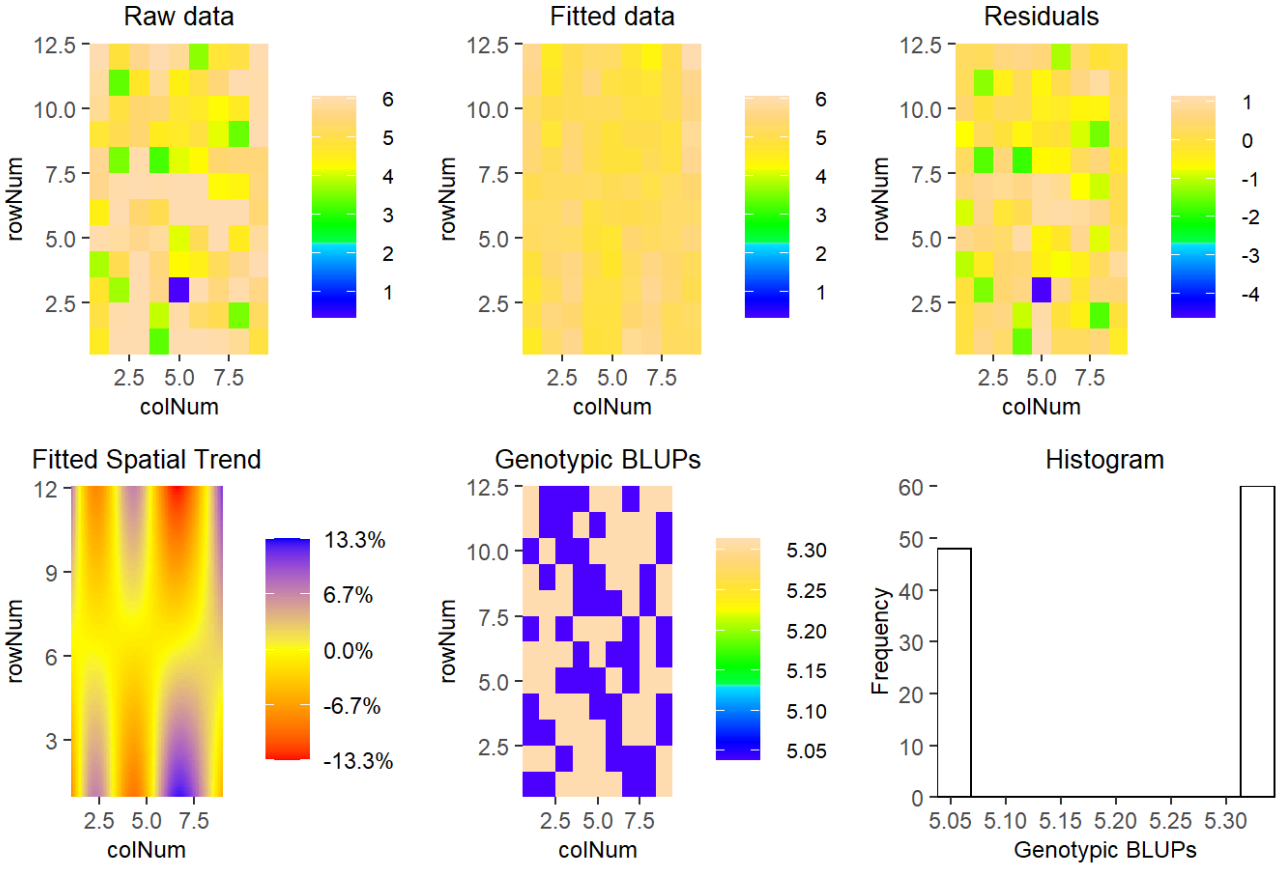
EPPN2020_UCPH - S_Height_cm - 2021-06-24

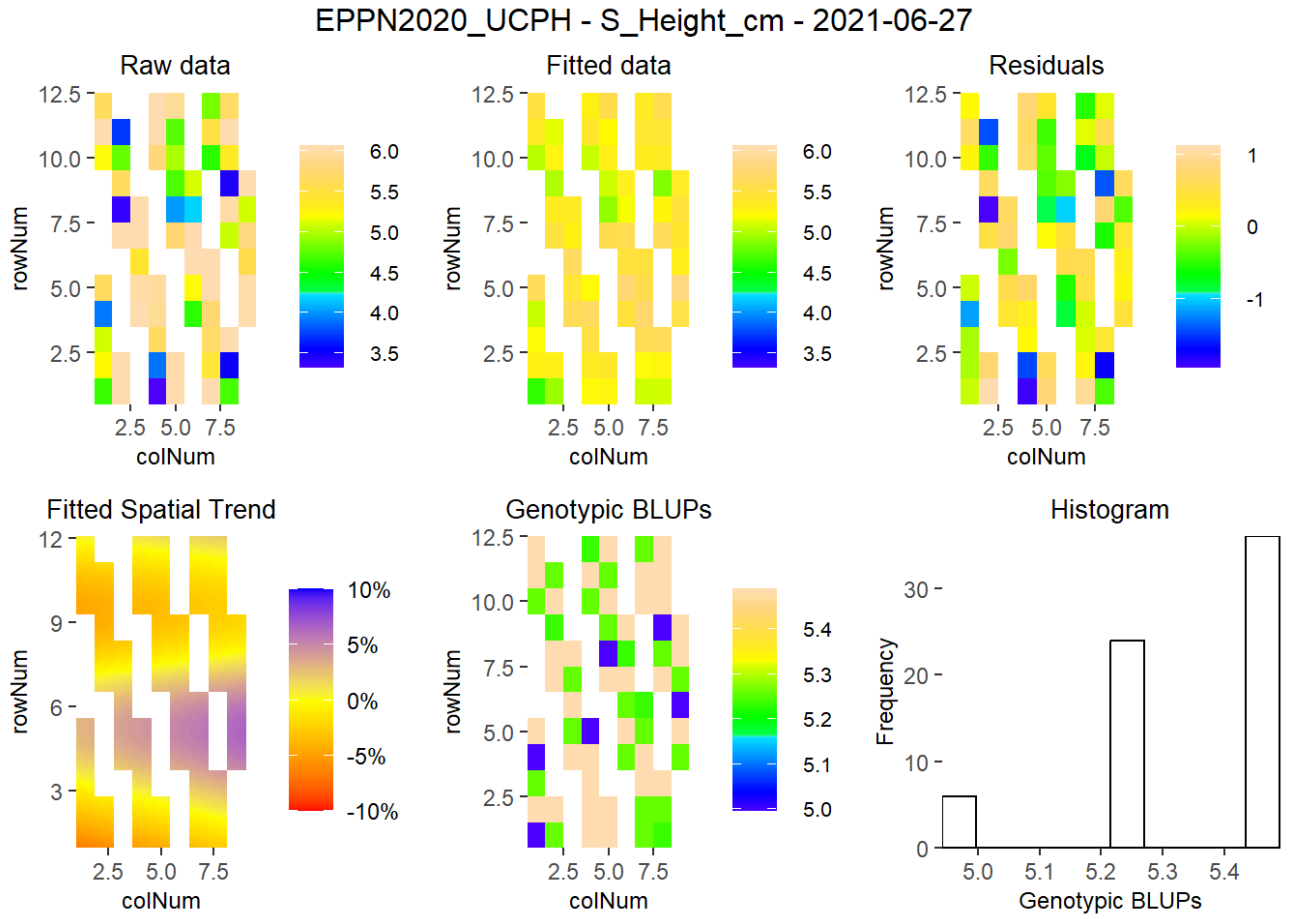


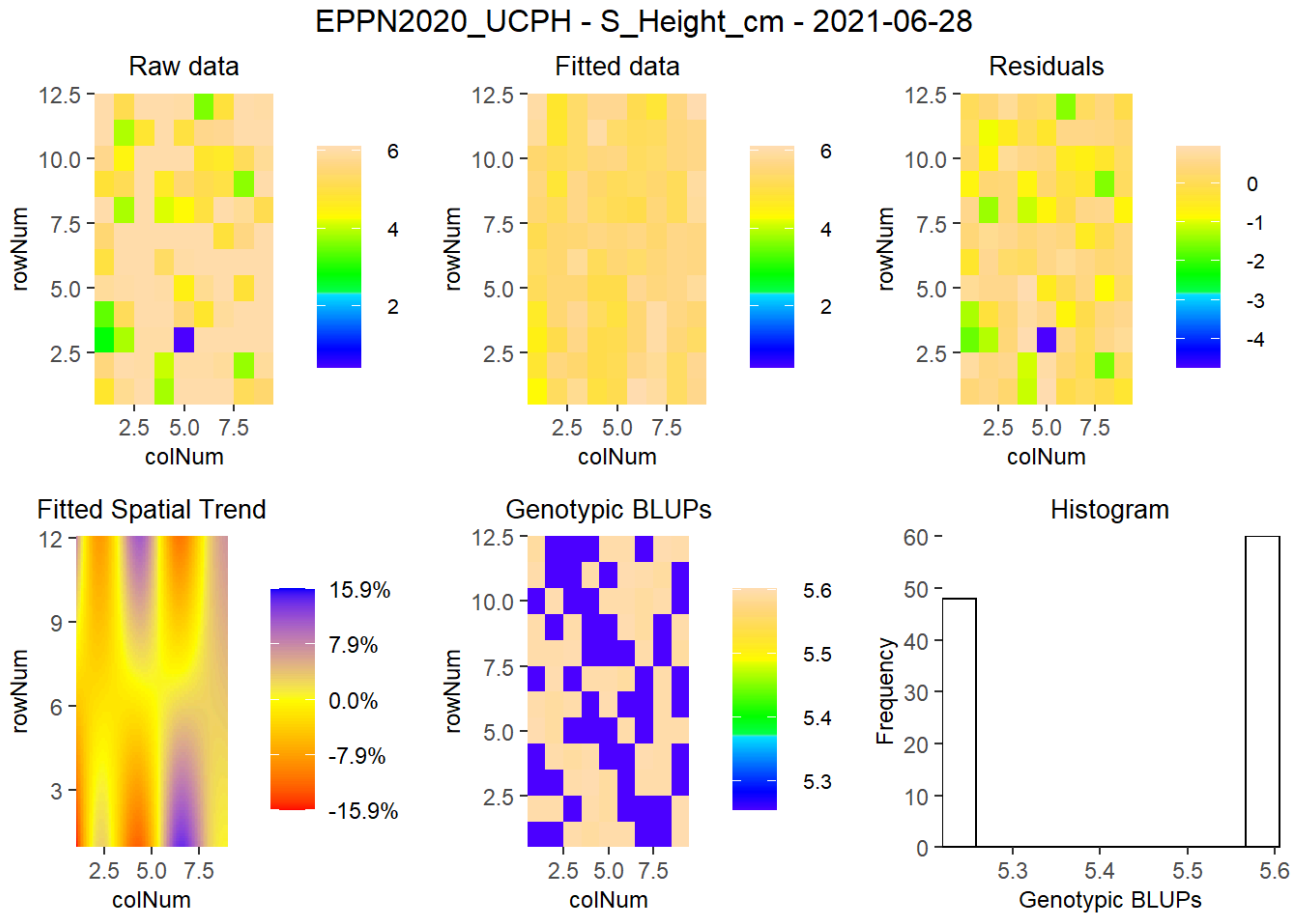
EPPN2020_UCPH - S_Height_cm - 2021-06-25



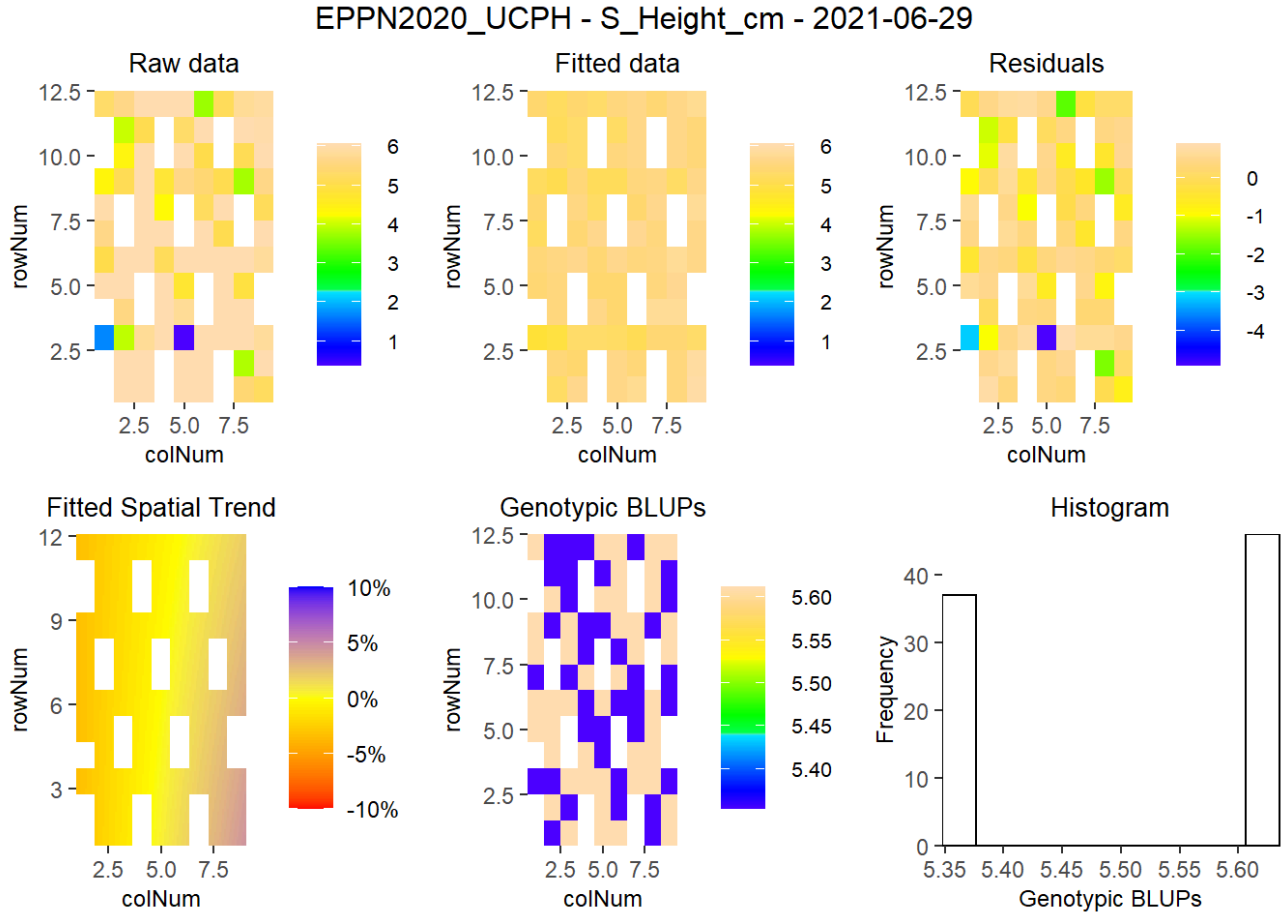
EPPN2020_UCPH - S_Height_cm - 2021-06-26





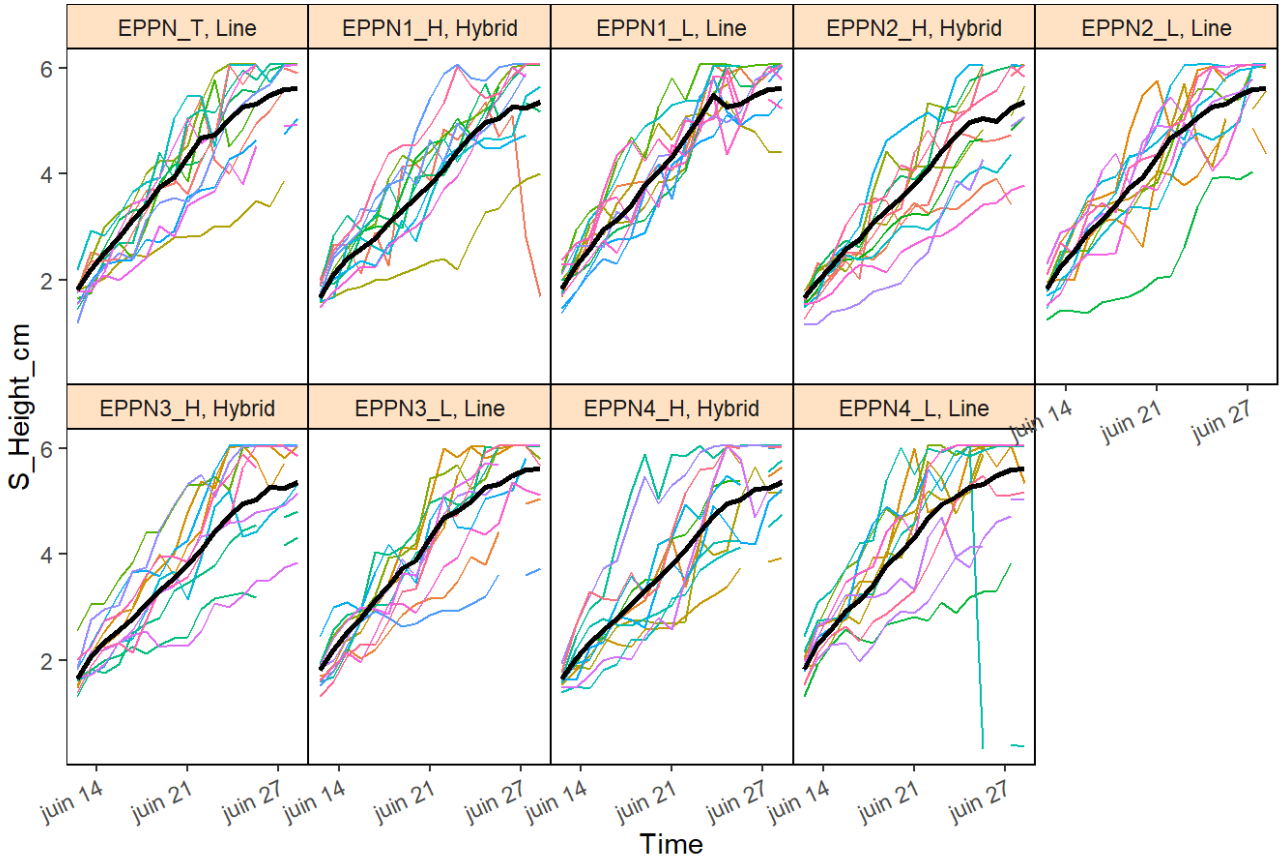



```
## Output at: C:/Users/elise/Documents/Mémoire/Main/Data/Extracted/S_Height_cm_mod.gif
```

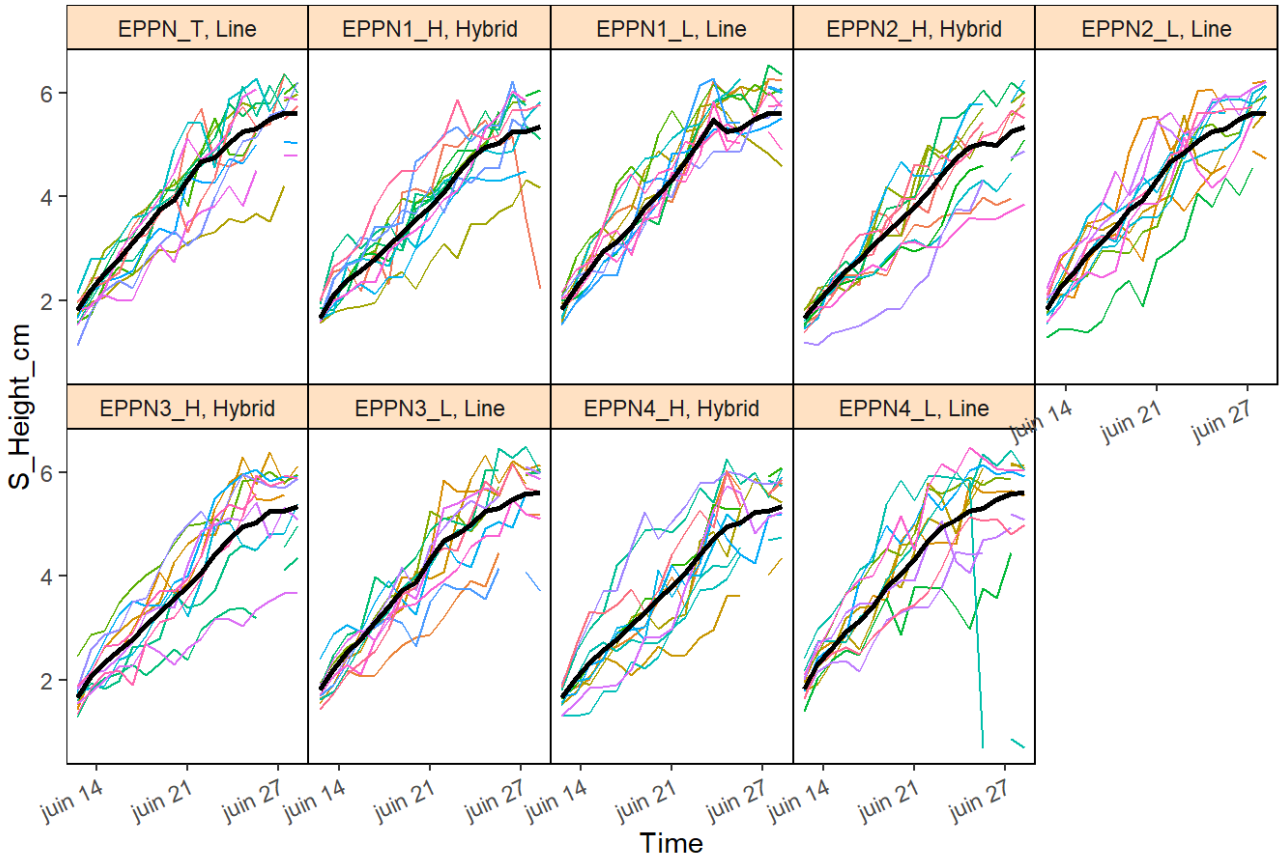


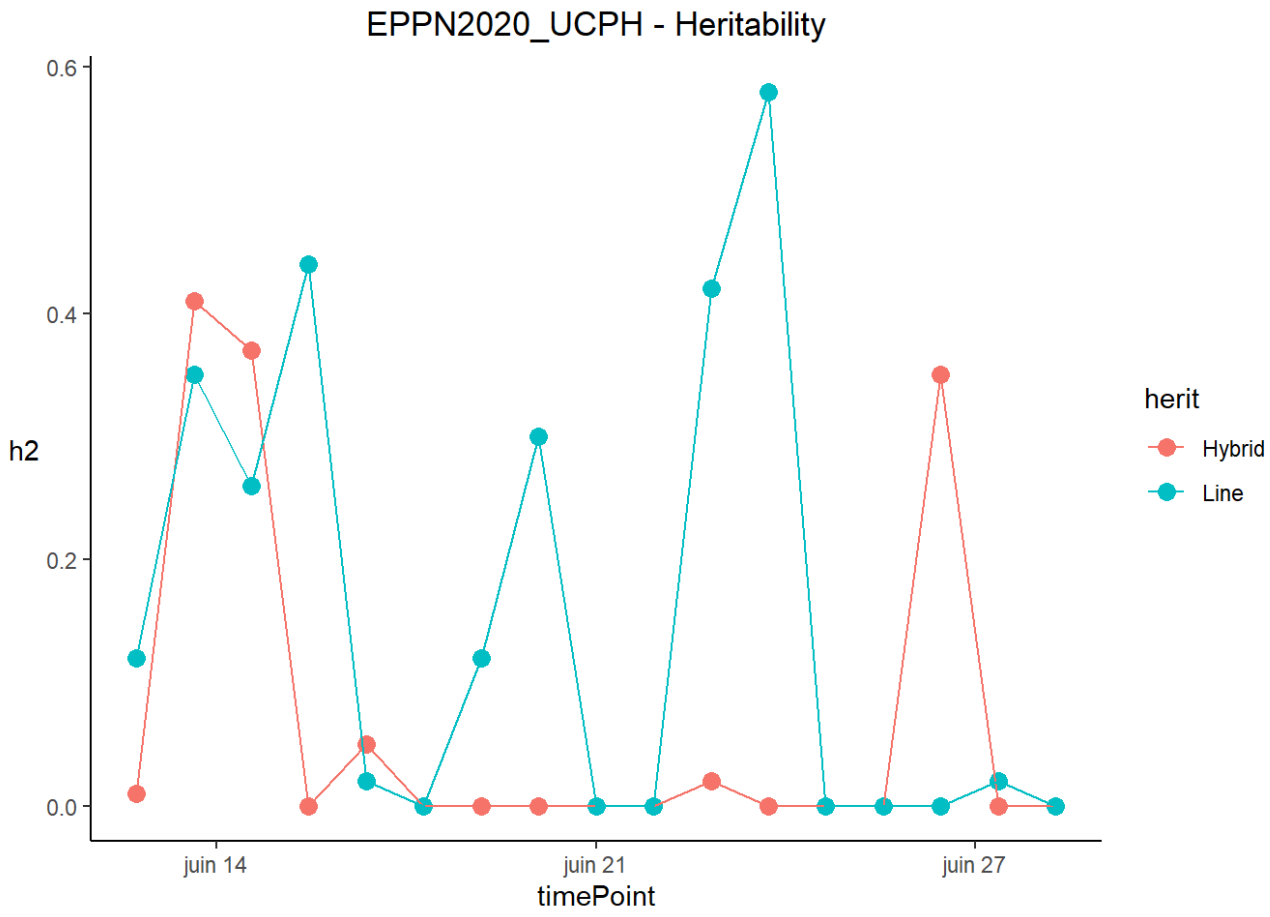

```
for (trait_name in traits) {
  mod_name <- paste0("modTP_", trait_name)
    if (exists(mod_name)) {
    mod <- get(mod_name)

    plot(mod,
         plotType = "rawPred",
         plotLine = TRUE)

    plot(mod,
         plotType = "corrPred",
         plotLine = TRUE)

    plot(mod,
         plotType = "herit",
         yLim = c(0, 0.5))

    plot(mod,
         plotType = "effDim",
         EDType = "ratio",
         yLim = c(0, 1))

    plot(mod,
         plotType = "corrPred")
  } else {
    cat("No model found for the trait", trait_name, "\n")
  }
}
```

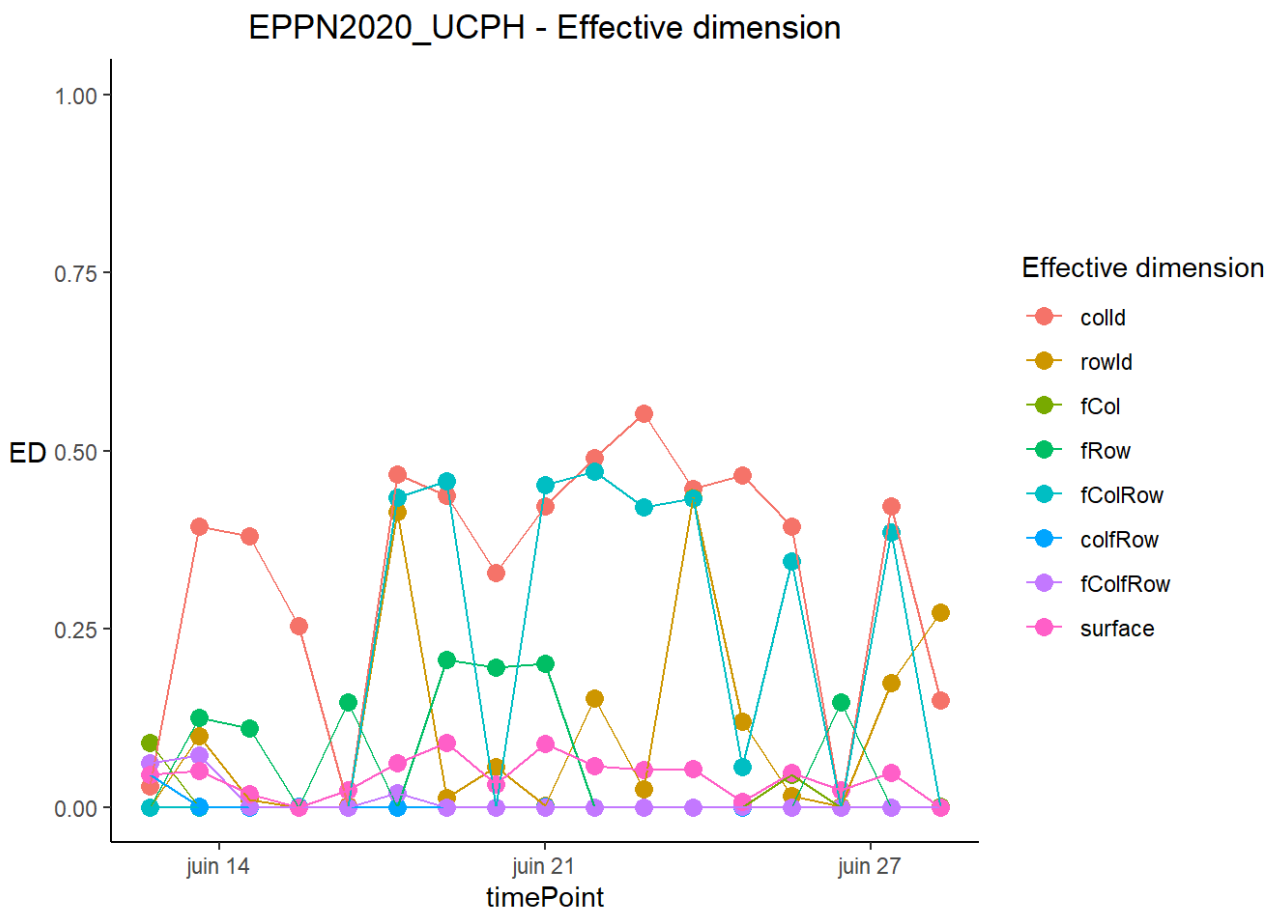
EPPN2020_UCPH - genotypic prediction + raw data

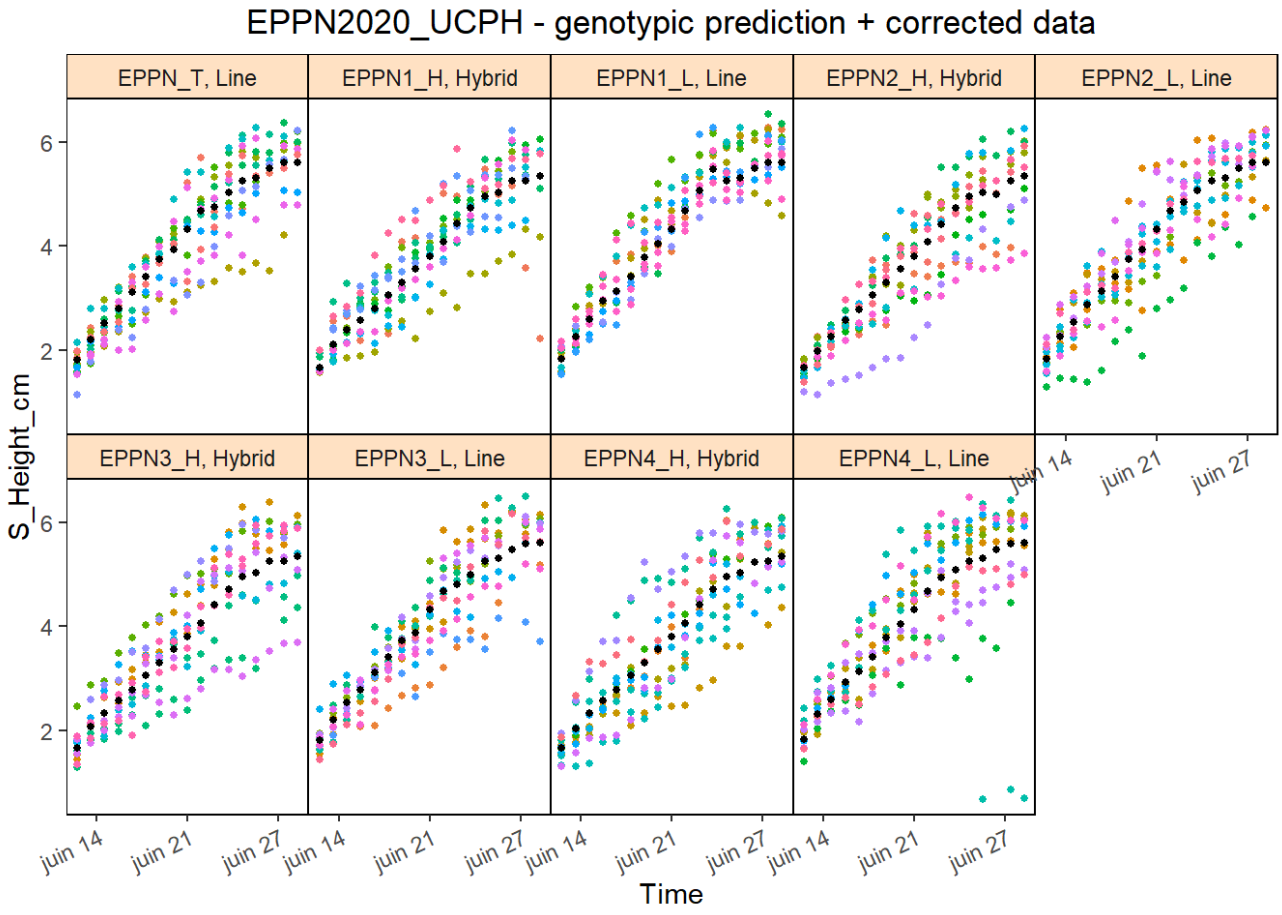


EPPN2020_UCPH - genotypic prediction + corrected data




EPPN2020_UCPH - Heritability
h2
0.6
0.4
0.2
0.0
juin 14
juin 21
juin 27
timePoint
herit
Hybrid
Line


EPPN2020_UCPH - Effective dimension
ED
1.00
0.75
0.50
0.25
0.00
juin 14
juin 21
juin 27
timePoint
Effective dimension
colId
rowId
fCol
fRow
fColRow
colfRow
fColfRow
surface



3. Outlier detection for series of observations

By using the splines.

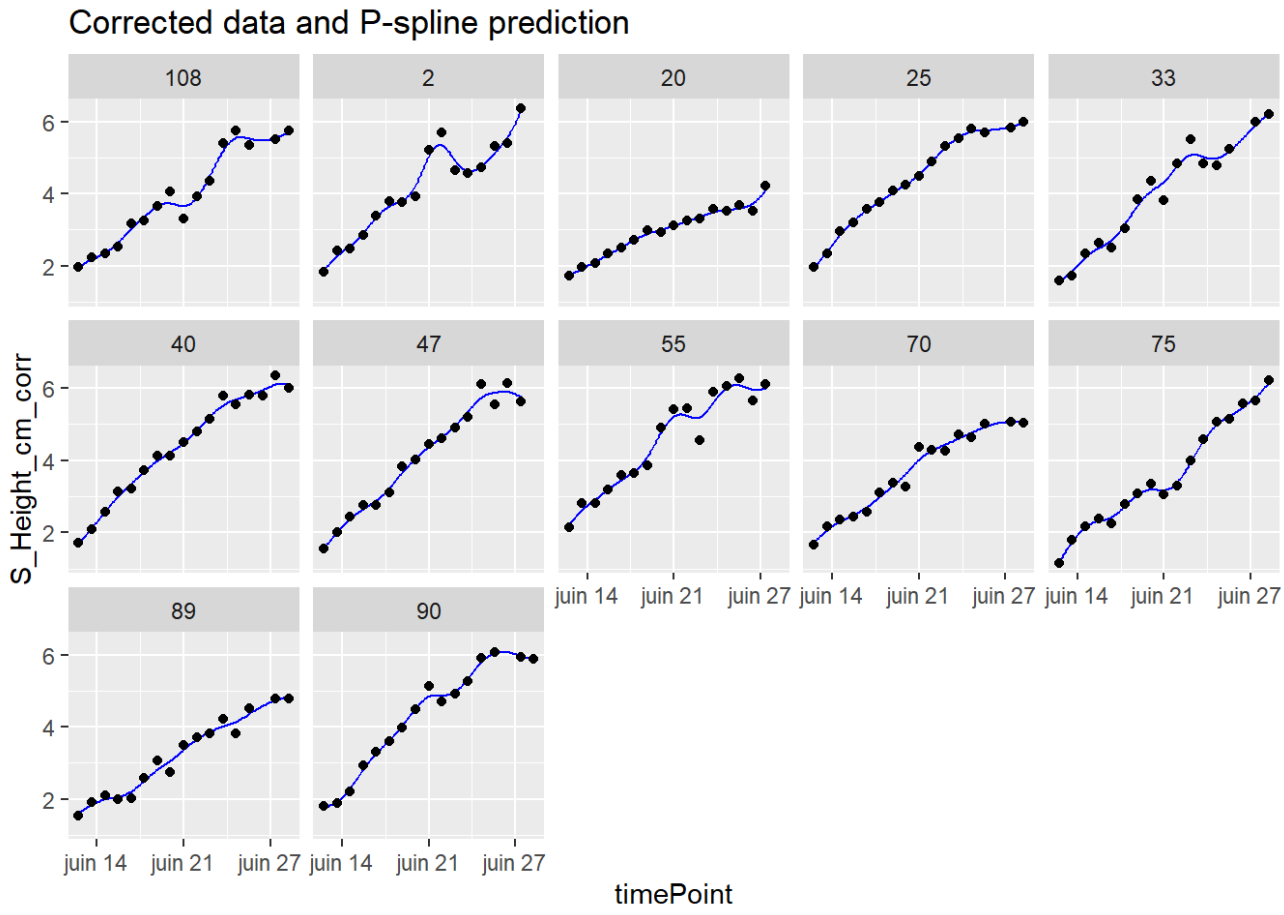
fitModels

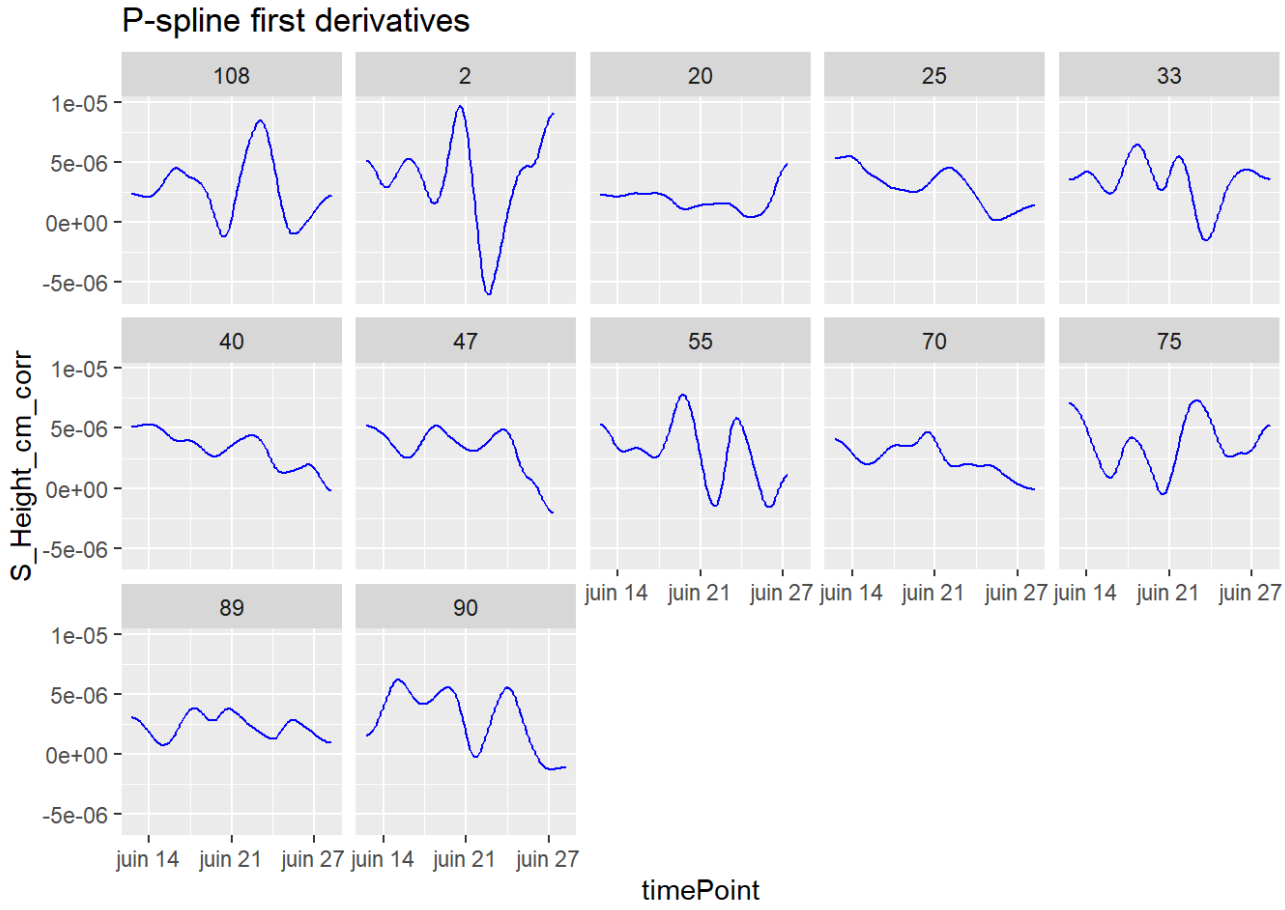
```
for (trait_name in traits) {
  Spatial_Corrected_name <- paste0("Spatial_Corrected_", trait_name)
  modTP_name <- paste0("modTP_", trait_name)
}
```

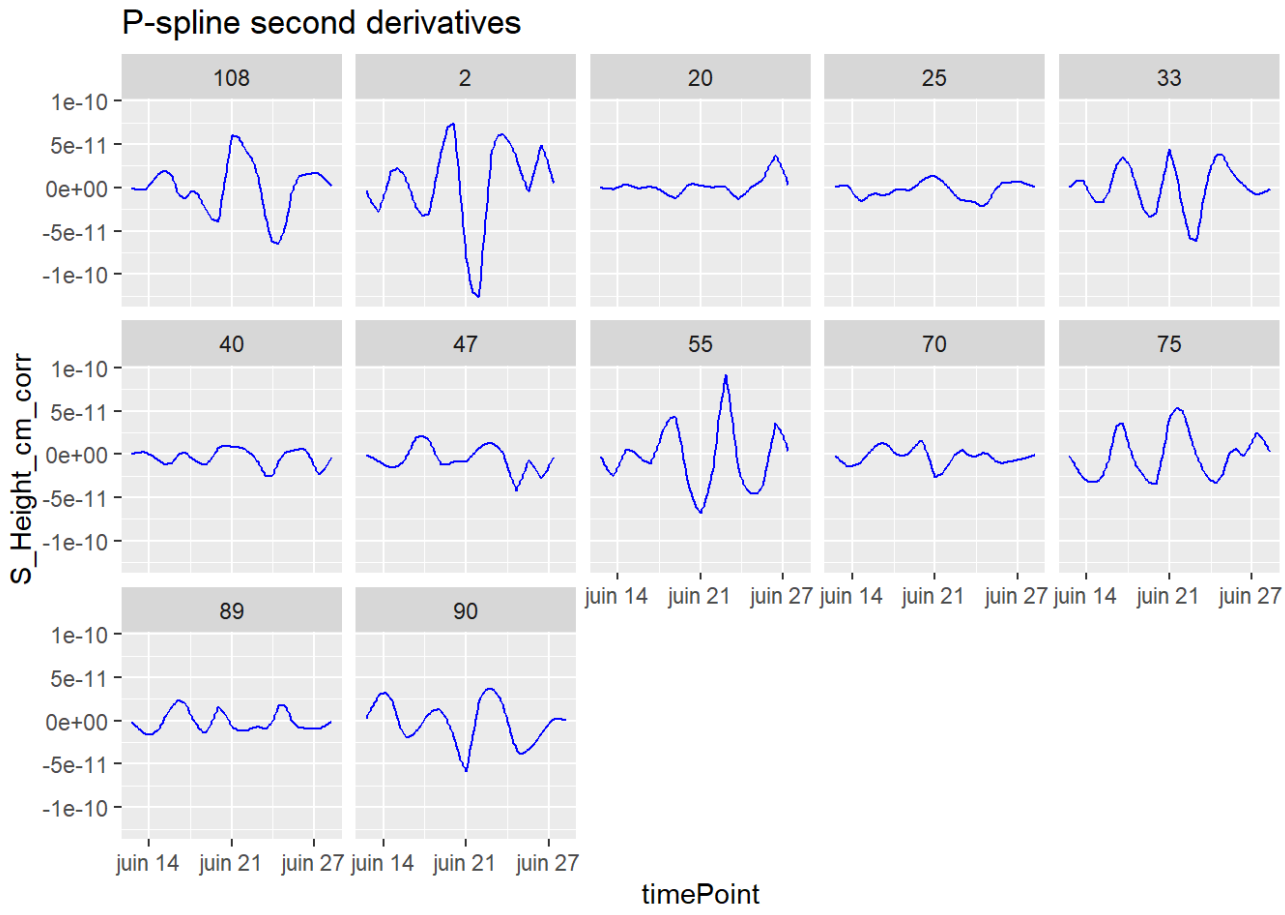
```
knots <- c(30)
mintimepoints <- c(9) # Minimal number of observations

for (trait_name in traits) {
  # Nom de la variable pour les données corrigées
  Spatial_Corrected_name <- paste0("Spatial_Corrected_", trait_name)
  modTP_name <- paste0("modTP_", trait_name)

  # Vérifier si le modèle existe
  if (exists(modTP_name)) {
    modTP <- get(modTP_name)
    Spatial_Corrected <- getCorrected(modTP)
    assign(Spatial_Corrected_name, Spatial_Corrected)

    # Ajuster les splines pour les données corrigées
    fit.spline <- fitSpline(inDat = Spatial_Corrected,
                            trait = paste0(trait_name, "_corr"),
                            knots = knots,
                            minNoTP = mintimepoints)

    # Extraire les tables de valeurs prédites et coefficients de splines
    predDat_name <- paste0("predDat_", trait_name)
    coefDat_name <- paste0("coefDat_", trait_name)

    assign(predDat_name, fit.spline$predDat)
    assign(coefDat_name, fit.spline$coefDat)
  } else {
    cat("No model found for", trait_name, "\n")
  }
}
```

Plot the splines for a plant selection

```
for (trait_name in traits) {
  plantSel <- S_timeseries[grepl('EPPN_T', S_timeseries$Genotype), "Unit.ID", drop = TR
UE]
  plantSel <- as.character(plantSel)

  plot(fit.spline,
       plotIds = plantSel,
       plotType = "predictions",
       main = paste("Predictions for", trait_name))

  plot(fit.spline,
       plotIds = plantSel,
       plotType = "derivatives",
       main = paste("Derivatives for", trait_name))

  plot(fit.spline,
       plotIds = plantSel,
       plotType = "derivatives2",
       main = paste("Second Derivatives for", trait_name))
}
```


Corrected data and P-spline prediction
108
2
20
25
33
40
47
55
70
75
89
90
S_Height_cm_corr
2
4
6
juin 14
juin 21
juin 27
timePoint

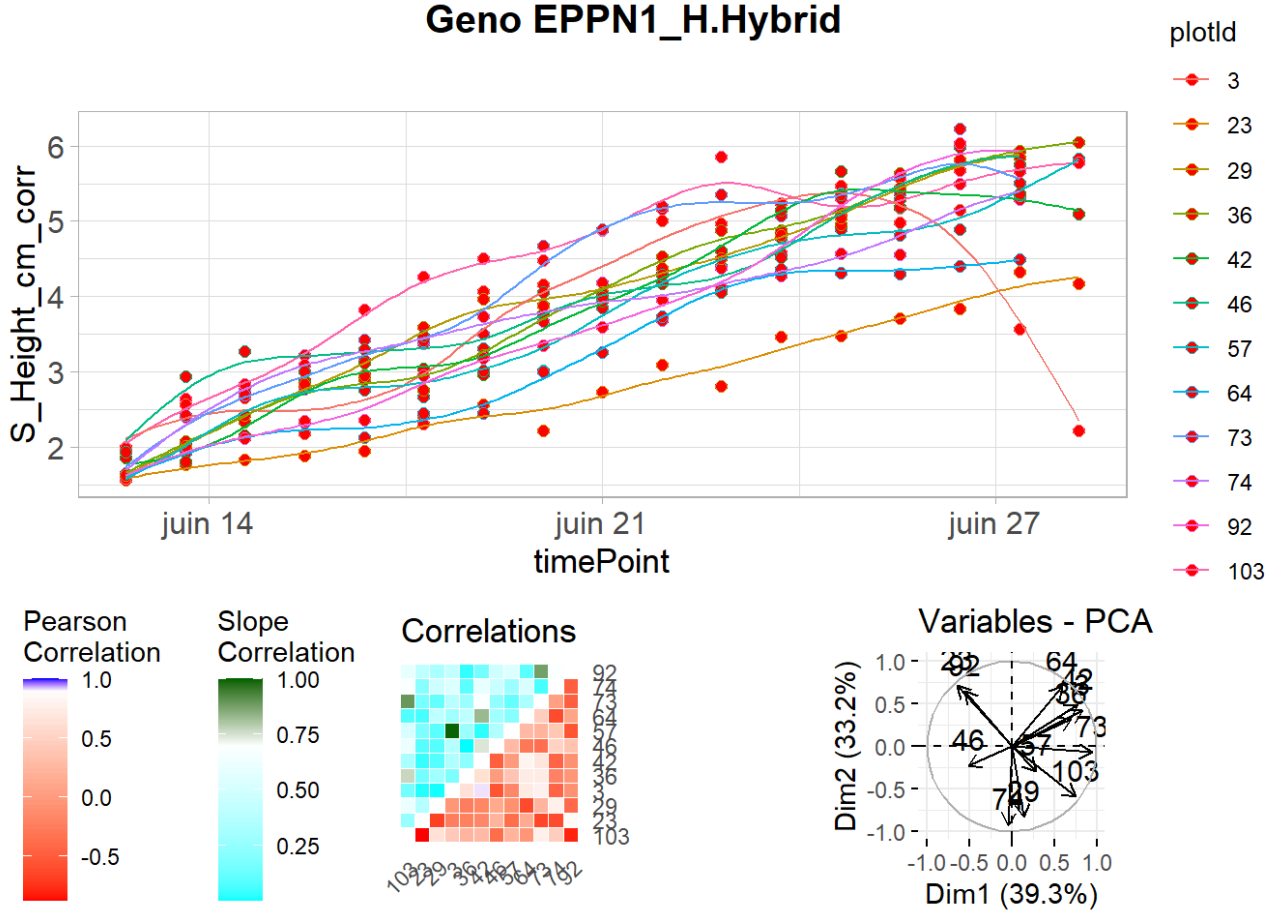

P-spline first derivatives
108
2
20
25
33
40
47
55
70
75
89
90
S_Height_cm_corr
1e-05
5e-06
0e+00
-5e-06
juin 14
juin 21
juin 27
timePoint



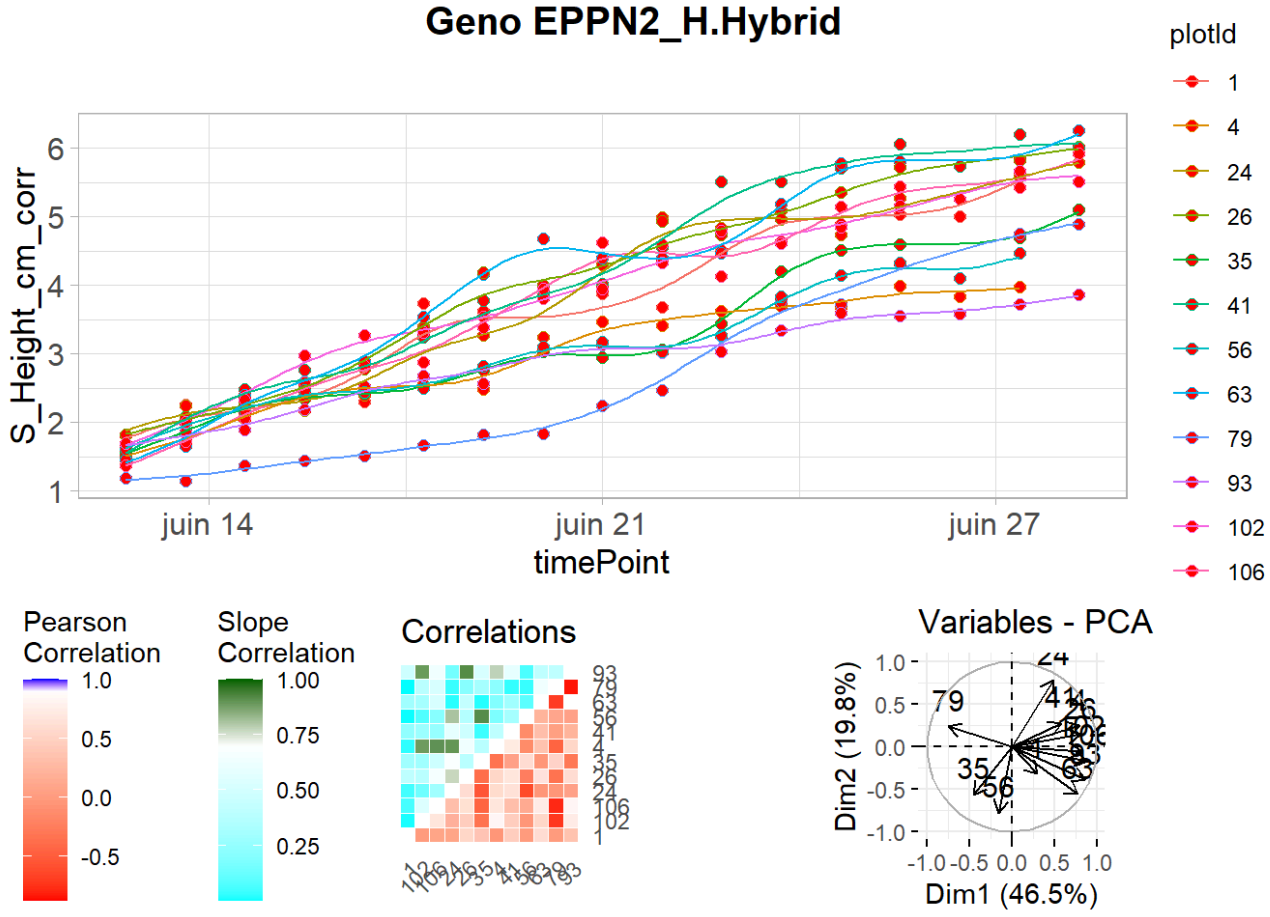
detectSerieOut

```
thrCor <- c(0.9) # correlation threshold
thrPca <- c(30) # pca angle threshold
thrSlope <- c(0.7) # slope threshold

for (trait_name in traits) {
  Spatial_Corrected_name <- paste0("Spatial_Corrected_", trait_name)
  predDat_name <- paste0("predDat_", trait_name)
  coefDat_name <- paste0("coefDat_", trait_name)

  if (exists(Spatial_Corrected_name) && exists(predDat_name) && exists(coefDat_name)) {
    Spatial_Corrected <- get(Spatial_Corrected_name)
    predDat <- get(predDat_name)
    coefDat <- get(coefDat_name)

    Series_test <- detectSerieOut(corrDat = Spatial_Corrected,
                                  predDat = predDat,
                                  coefDat = coefDat,
                                  trait = paste0(trait_name, "_corr"),
                                  thrCor = thrCor,
                                  thrPca = thrPca,
                                  thrSlope = thrSlope,
                                  geno.decomp = "geno.decomp")

    plot(Series_test, genotypes = levels(factor(Series_test$genotype)))

    assign(paste0("Series_test_", trait_name), Series_test)

    assign(paste0("Spatial_Corrected_Out_", trait_name), Spatial_Corrected)
  } else {
    cat("No corrected data or prediction data found for", trait_name, "\n")
  }
}
```

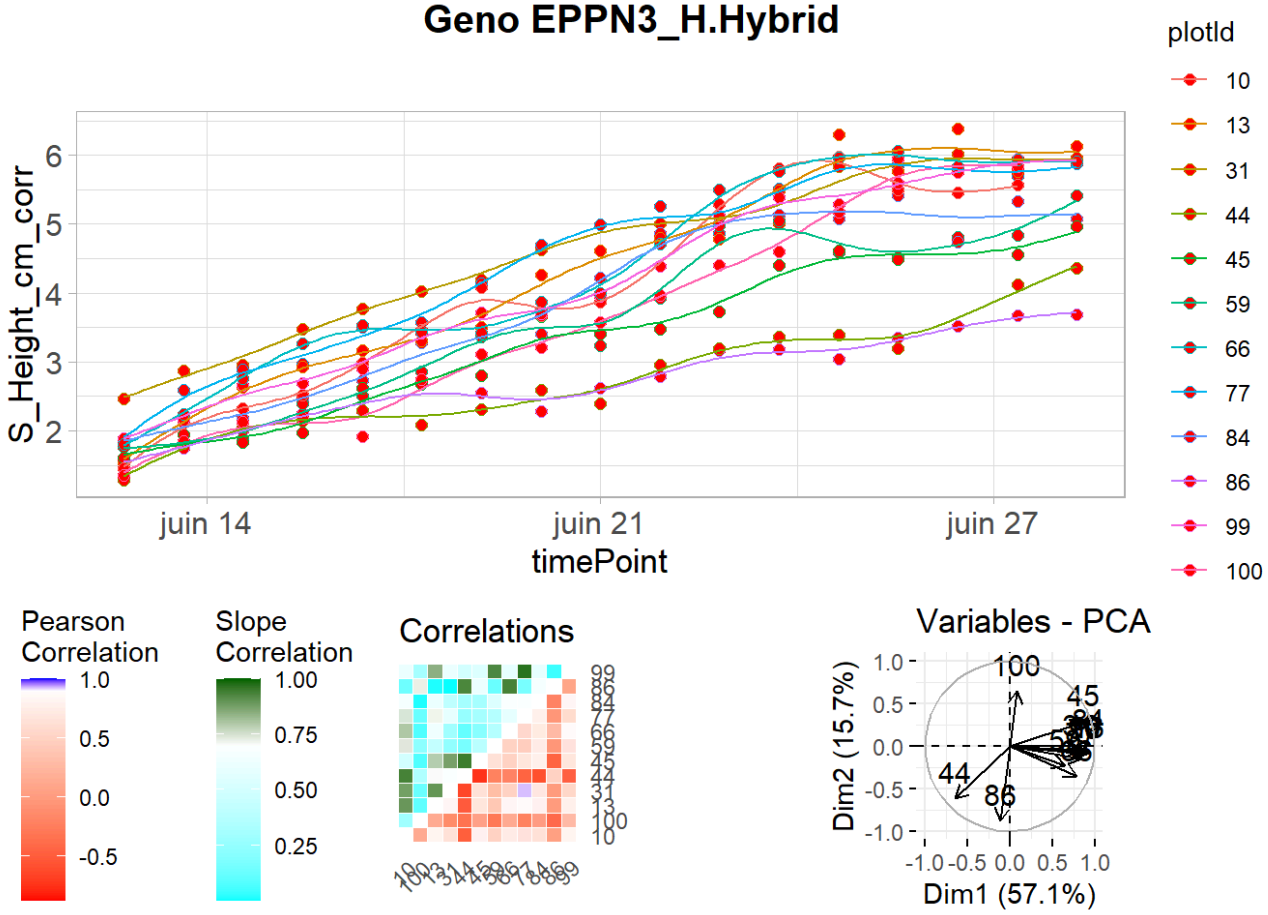
Geno EPPN1_H.Hybrid



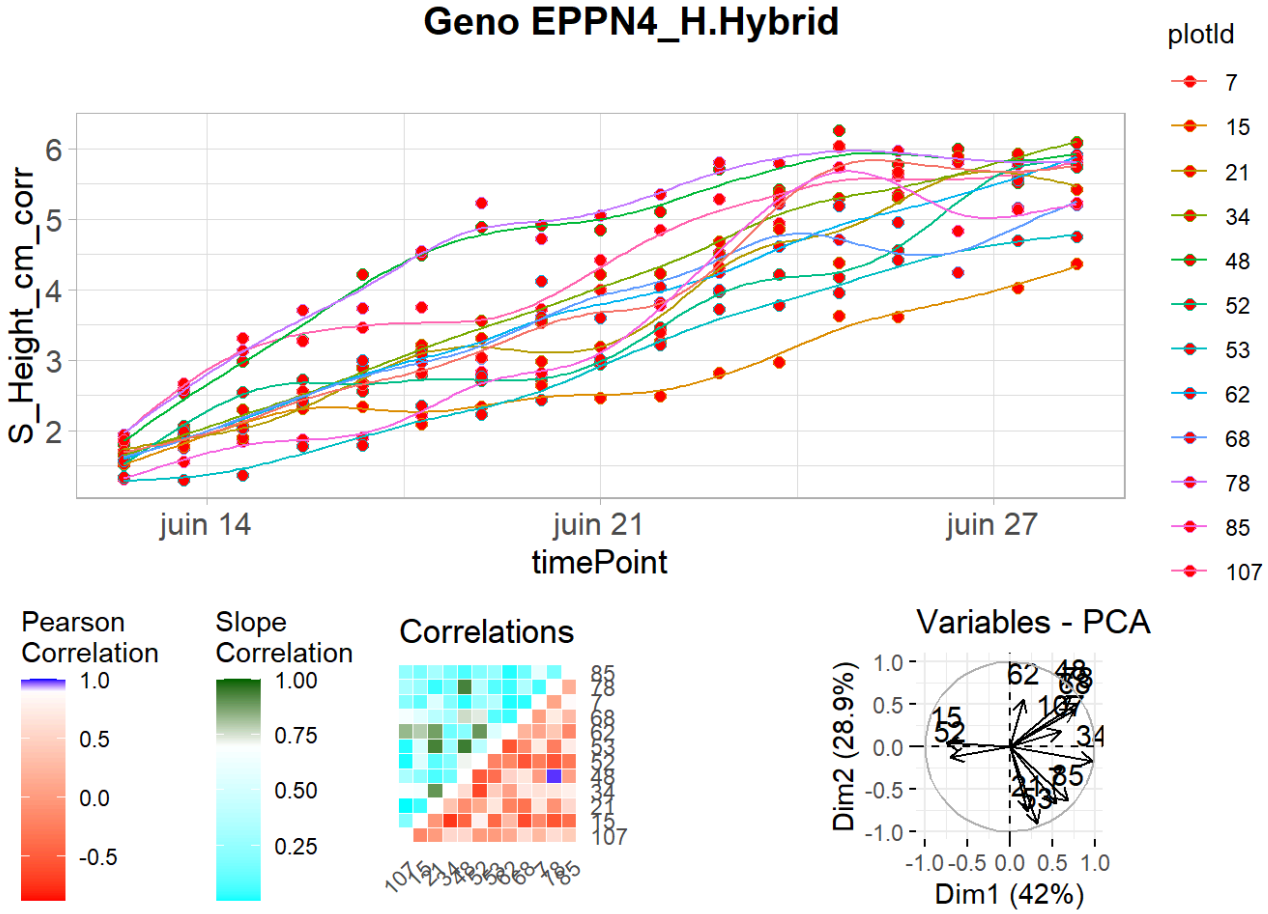
Geno EPPN2_H.Hybrid



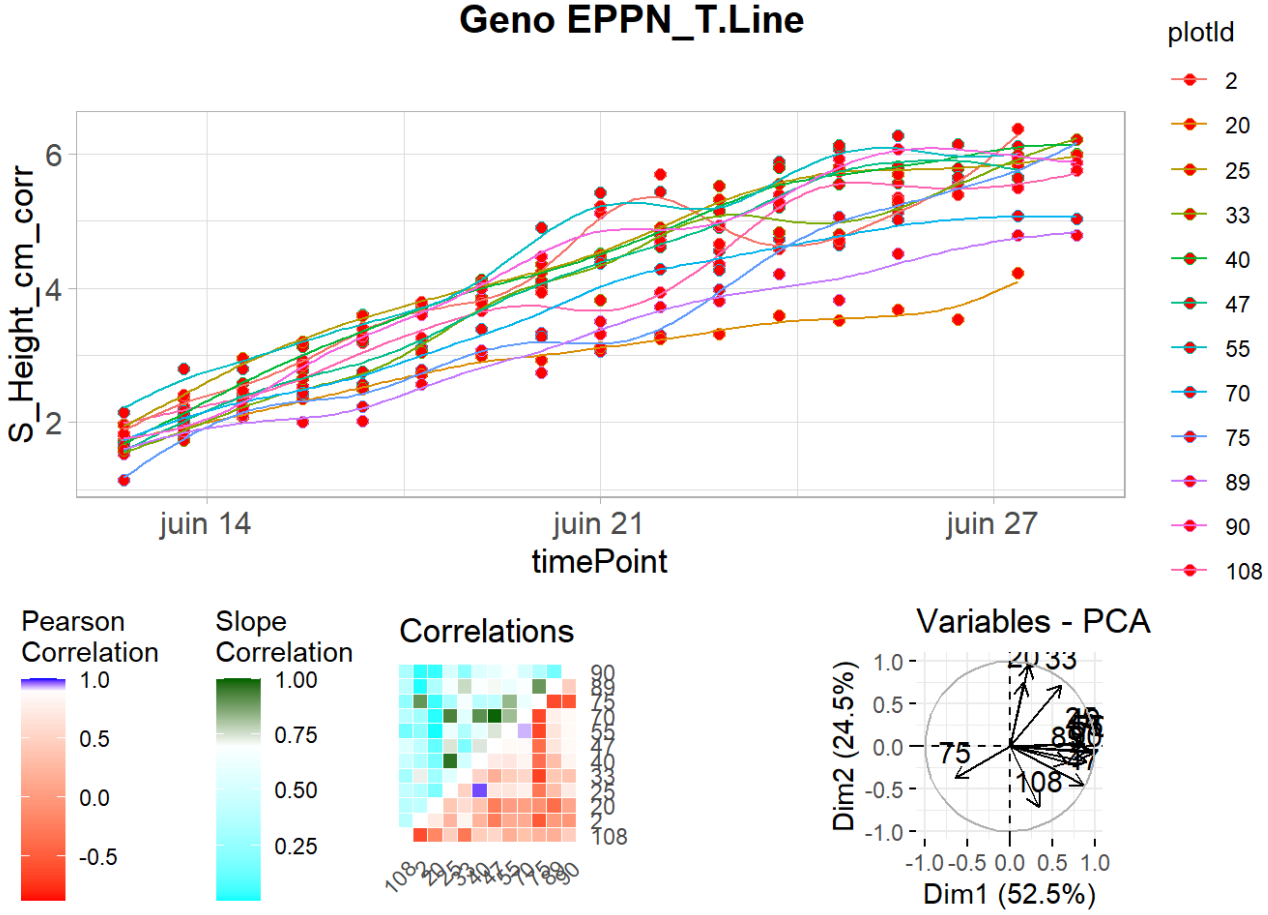
Geno EPPN3_H.Hybrid



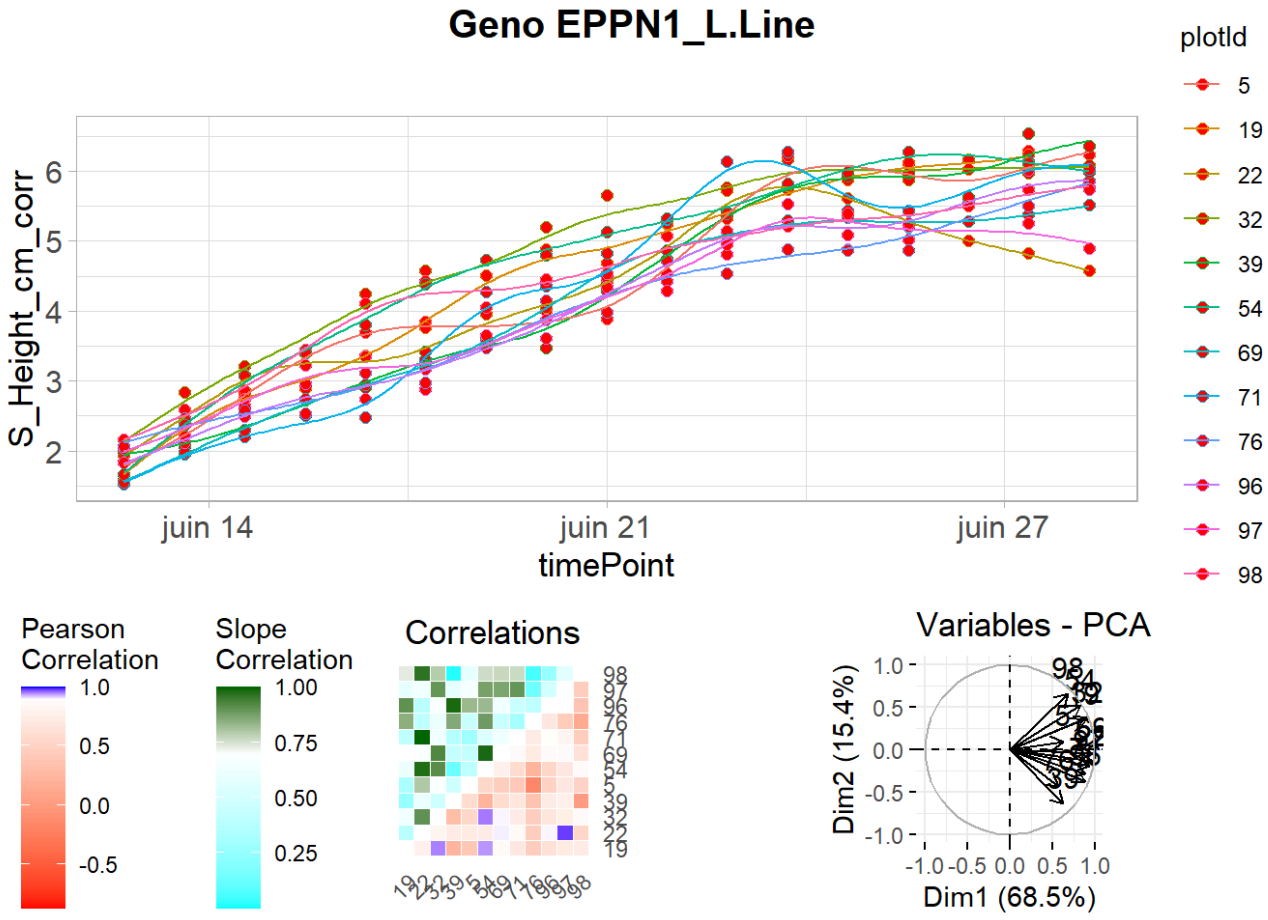
Geno EPPN4_H.Hybrid



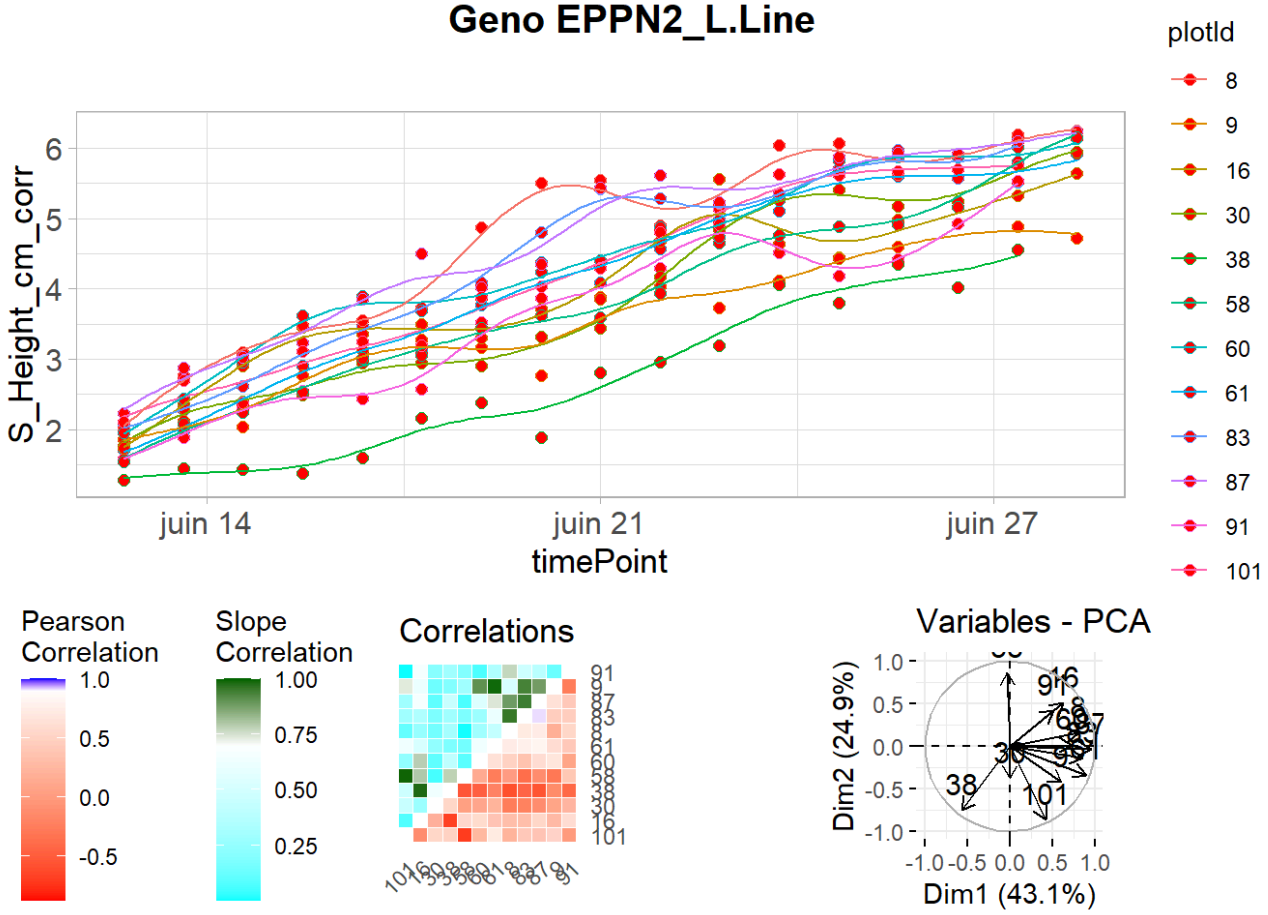
Geno EPPN_T.Line



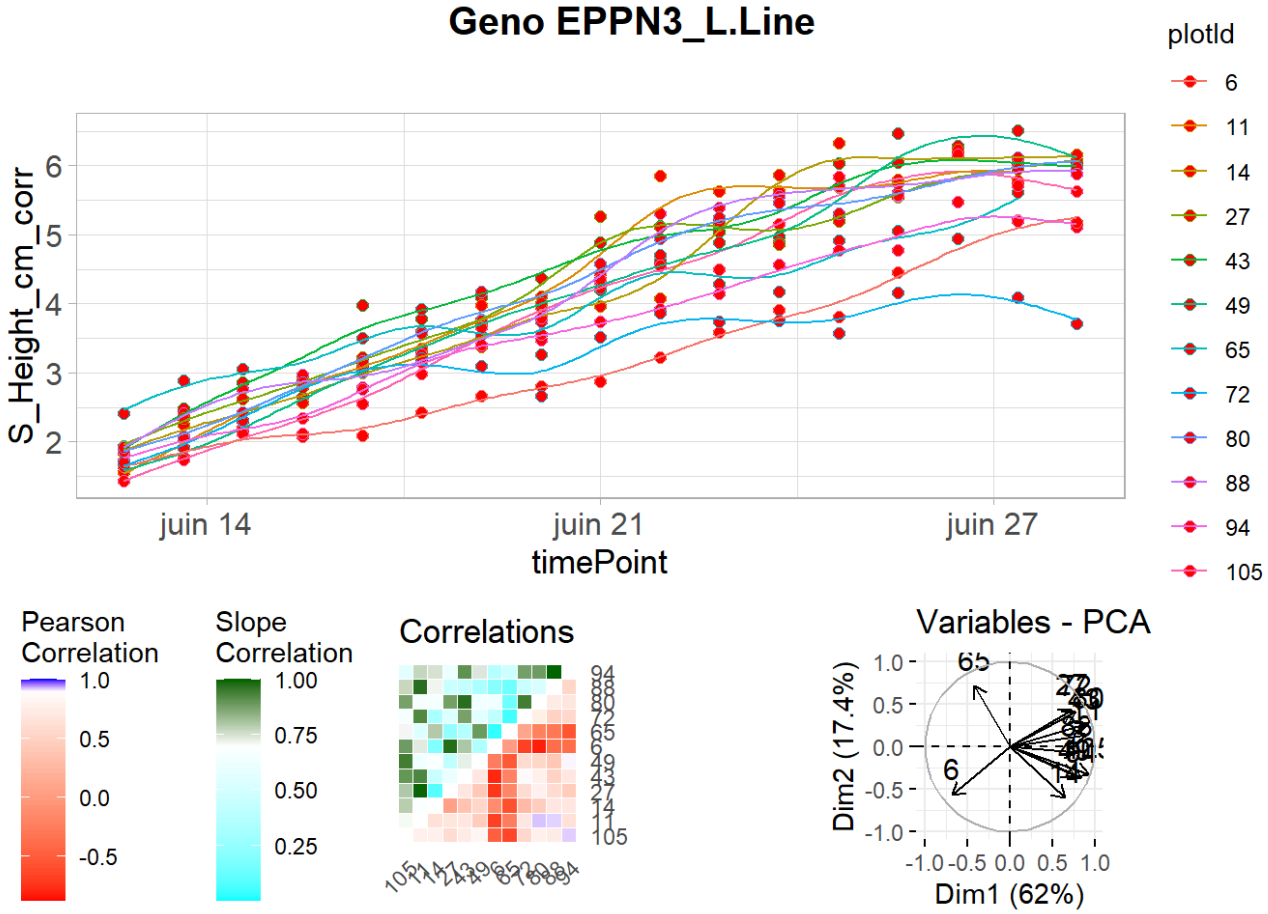
Geno EPPN1_L.Line

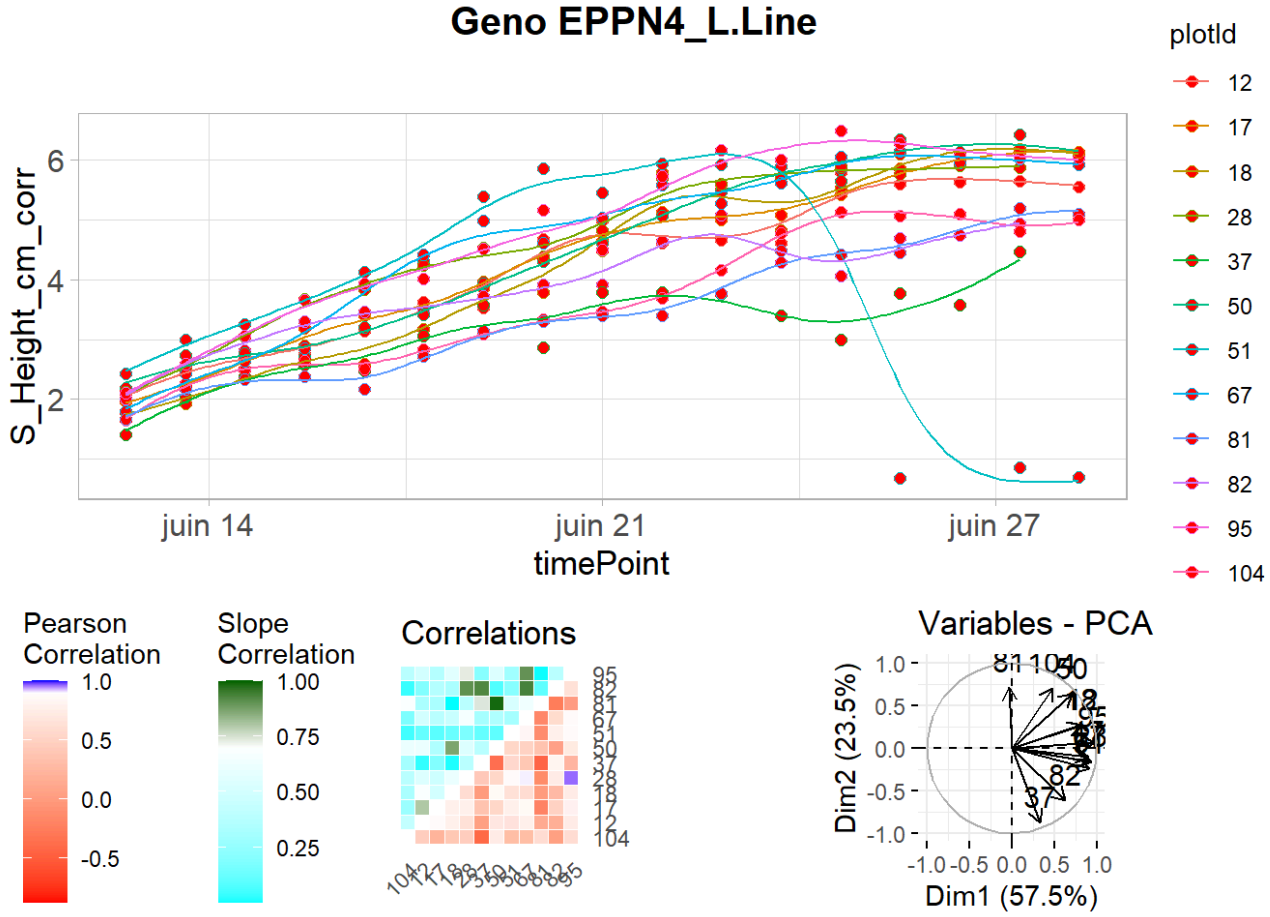


Geno EPPN2_L.Line



Geno EPPN3_L.Line





removeSerieOut

```
for (trait_name in traits) {
  # Nom de la variable pour les données corrigées
  Spatial_Corrected_name <- paste0("Spatial_Corrected_", trait_name)
  Series_test_name <- paste0("Series_test_", trait_name)

  # Extraire les données corrigées et les résultats des séries
  if (exists(Spatial_Corrected_name) && exists(Series_test_name)) {
    Spatial_Corrected <- get(Spatial_Corrected_name)
    Series_test <- get(Series_test_name)

    # Supprimer les outliers de la série
    Spatial_Corrected_Out <- removeSerieOut(dat = Spatial_Corrected, serieOut = Series_
test)

    # Assigner le résultat à une nouvelle variable
    assign(paste0("Spatial_Corrected_Out_", trait_name), Spatial_Corrected_Out)
  } else {
    cat("No corrected data or series test data found for", trait_name, "\n")
  }
}
```

```
for (trait_name in traits) {
  Spatial_Corrected_Out_name <- paste0("Spatial_Corrected_Out_", trait_name)

  if (exists(Spatial_Corrected_Out_name)) {
    Spatial_Corrected_Out <- get(Spatial_Corrected_Out_name)
    output_file <- sprintf("%s/timeSeriesOutliers_%s.tsv", datadir, trait_name)
    readr::write_tsv(Spatial_Corrected_Out, output_file)

    cat("Data written to:", output_file, "\n")
  } else {
    cat("No corrected data found for", trait_name, "\n")
  }
}
```

```
## Data written to: C:/Users/elise/Documents/Mémoire/Main/Data/Extracted/timeSeriesOutl
iers_S_Height_cm.tsv
```

4. With the cleaned data, re-do the spatial correction

This is used to compare the values before and after.

Need to write a for loop for all the variables.

For S_Height_cm

```
trait_name <- "S_Height_cm"

timePoint_S_2 <- createTimePoints(dat = Spatial_Corrected_Out_S_Height_cm,
                                  experimentName = "EPPN2020_clean",
                                  genotype = "genotype",
                                  timePoint = "timePoint",
                                  plotId = "plotId",
                                  rowNum = "rowId",
                                  colNum = "colId")

## Correct for spatial
modTP_2 <- fitModels(TP = timePoint_S_2,
                           trait = trait_name,
                           geno.decomp = c("geno.decomp"))
```

```
## 2021-06-13
```

```
## 2021-06-14
```

```
## 2021-06-15
```

```
## 2021-06-16
```

```
## 2021-06-17
```

```
## 2021-06-18
```

```
## 2021-06-19
```

```
## 2021-06-20
```

```
## 2021-06-21
```

```
## 2021-06-22
```

```
## 2021-06-23
```

```
## 2021-06-24
```

```
## 2021-06-25
```

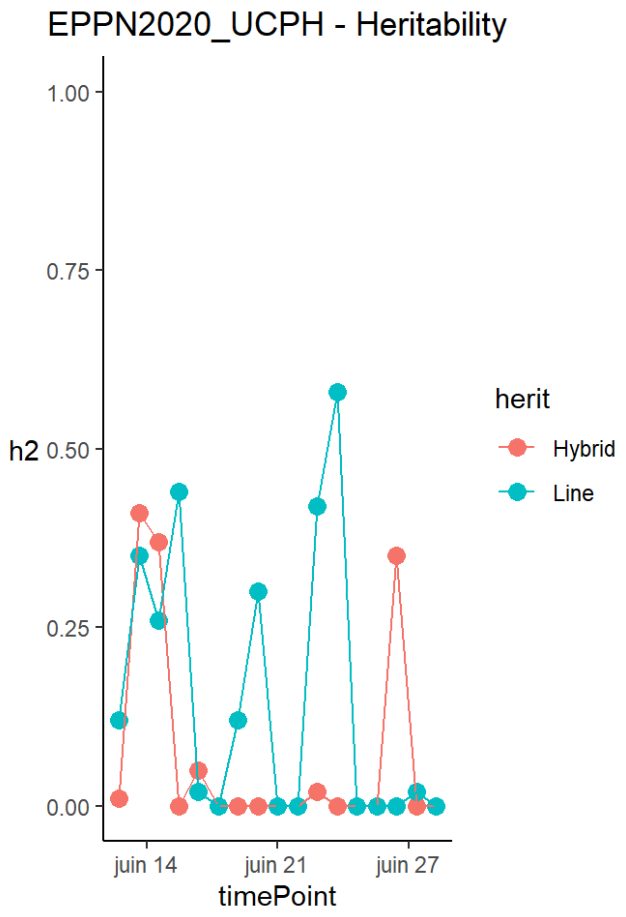
```
## 2021-06-26
```

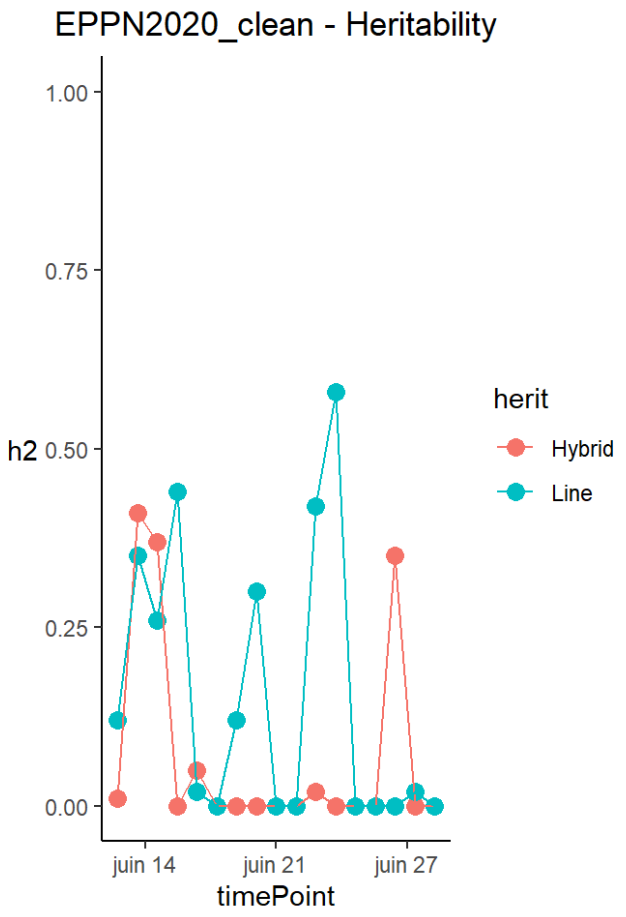
```
## 2021-06-27
```

```
## 2021-06-28
```

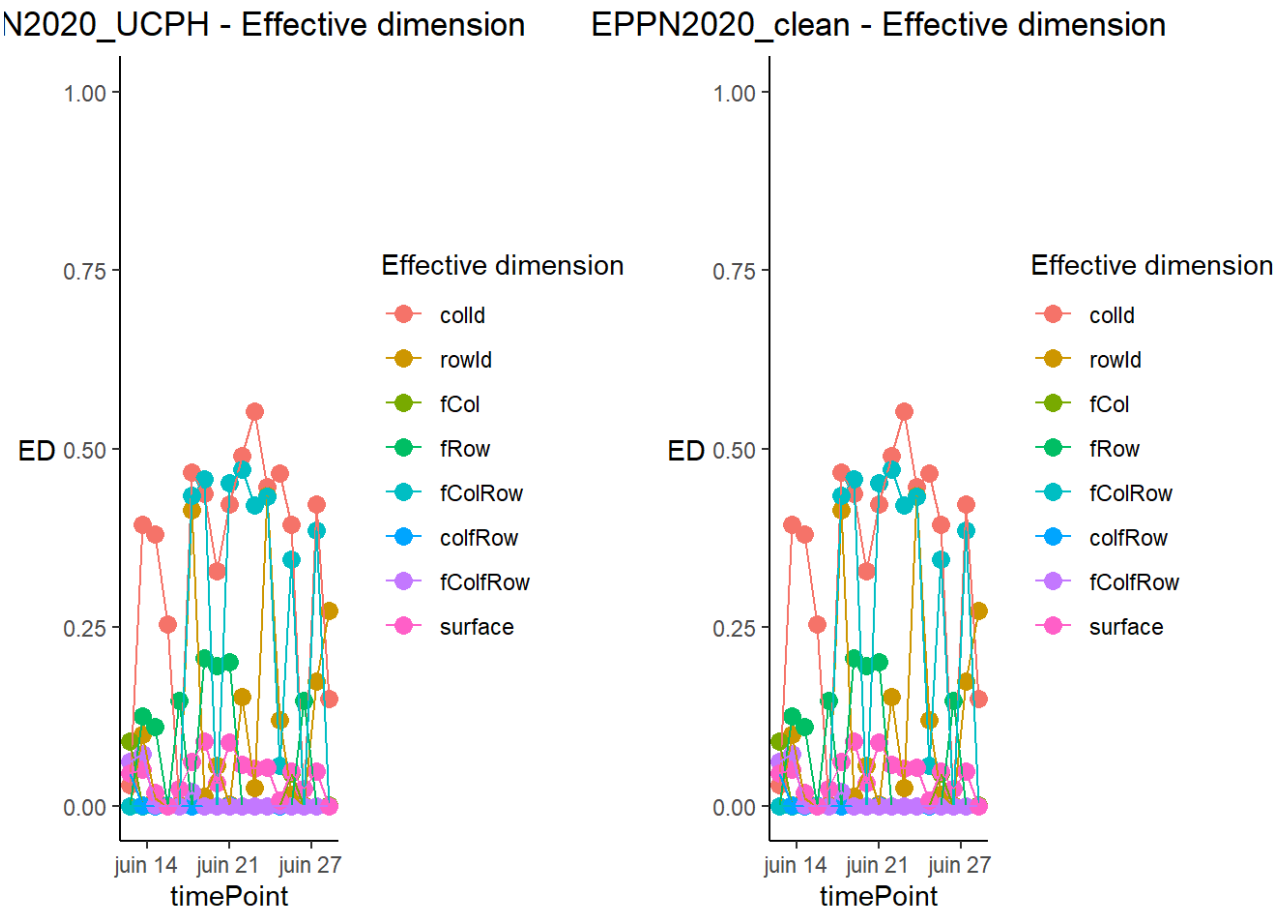
```
## 2021-06-29
```

```
## Extract corrected values
Spatial_Corrected_2 <- getCorrected(modTP_2)

#Check the values before and after:

h21<-plot(modTP, output = FALSE,
          plotType = "herit",
          yLim = c(0.4,1))

h22<-plot(modTP_2, output = FALSE,
          plotType = "herit",
          yLim = c(0.4,1))

grid.arrange(h21, h22,  nrow = 1)
```



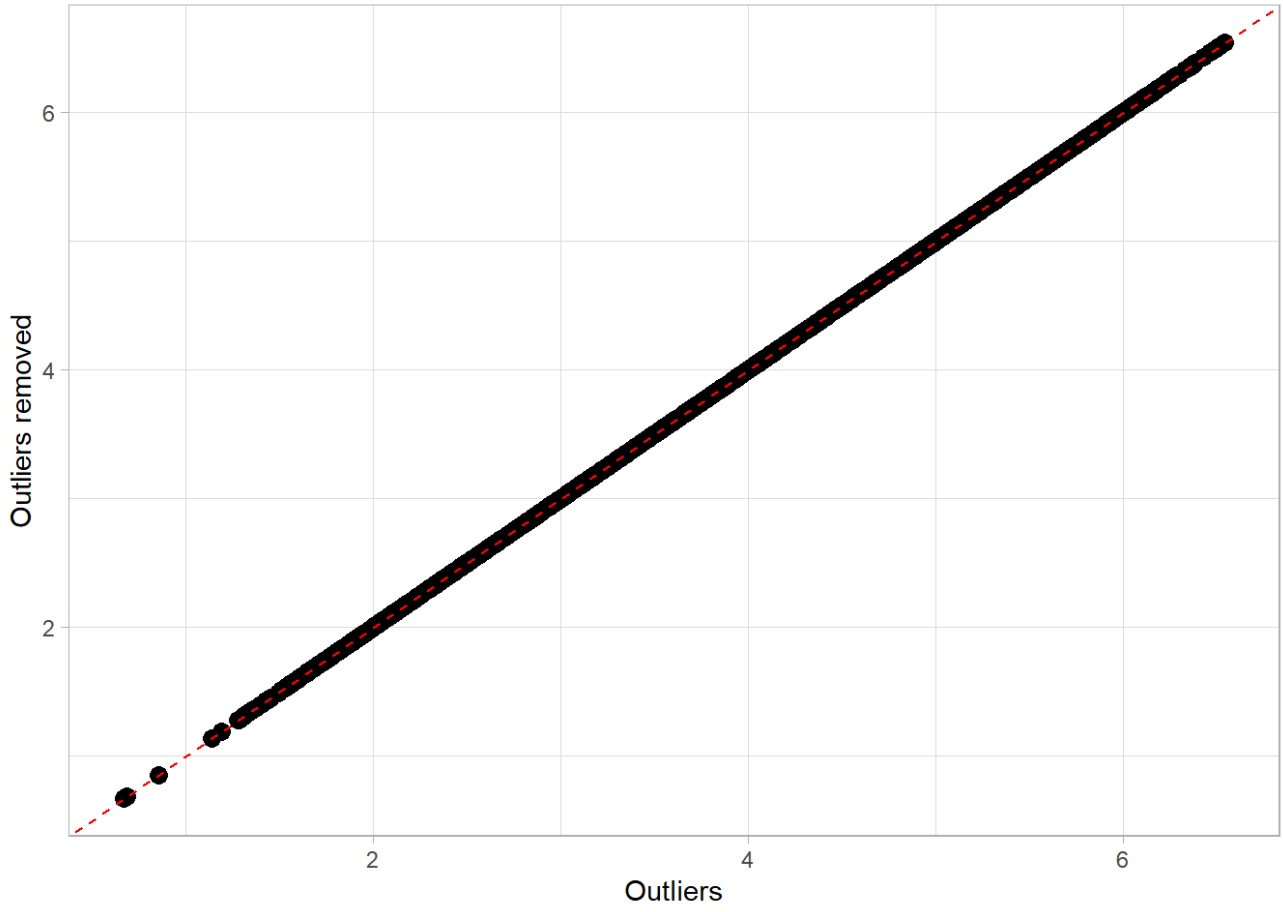



```
#
ed2<-plot(modTP_2,output = FALSE,
          plotType = "effDim",
          EDType = "ratio",
          yLim = c(0,1))

ed1<-plot(modTP,output = FALSE,
          plotType = "effDim",
          EDType = "ratio",
          yLim = c(0,1))

grid.arrange(ed1, ed2,  nrow = 1)
```




```
spatial_Corr <-
  inner_join(Spatial_Corrected_S_Height_cm[,c("timeNumber","plotId",paste0(trait_name,
"_corr"))],
             Spatial_Corrected_2[,c("timeNumber","plotId",paste0(trait_name, "_cor
r"))],
             by = c("timeNumber","plotId") )


col1 <- paste0(trait_name, "_corr.x")
col2 <- paste0(trait_name, "_corr.y")
ggplot(spatial_Corr, aes(x = .data[[col1]], y = .data[[col2]])) +
  geom_point(size=3) +
  geom_abline(slope=1, intercept = 0,col="red",lty=2) +
  ylab("Outliers removed") + xlab("Outliers") + theme_light()
```

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




```
# fit Spline again
fit.spline_2 <- fitSpline(inDat = Spatial_Corrected_2,
                                trait = paste0(trait_name, "_corr"),
                                knots = 30,
                                minNoTP = 9)
```

Estimation of parameter from time series

```
param1bis <- estimateSplineParameters(x = fit.spline_2,
                            estimate = "derivatives",
                            what = "max")
head(param1bis)
```

```
##   genotype geno.decomp plotId max_derivatives max_timeNumber
## 1   EPPN_T        Line    108    8.427168e-06         892800
## 2   EPPN_T        Line      2    9.671293e-06         652800
## 3   EPPN_T        Line     20    4.874353e-06        1296000
## 4   EPPN_T        Line     25    5.491787e-06          96000
## 5   EPPN_T        Line     33    6.457257e-06         470400
## 6   EPPN_T        Line     40    5.298470e-06         115200
##         max_timePoint
## 1 2021-06-23 08:00:00
## 2 2021-06-20 13:20:00
## 3 2021-06-28 00:00:00
## 4 2021-06-14 02:40:00
## 5 2021-06-18 10:40:00
## 6 2021-06-14 08:00:00
```

```
param1bis[,'genotype'] <- factor( param1bis[, 'genotype'],
                                  levels = genotypes_list)

# Visualize the variability
ggplot(param1bis,
       aes(x = genotype, y = max_derivatives)) +
  geom_boxplot(na.rm = TRUE) +
  ylab("Max growth rate") +
  theme_classic()
```