FZJ Data Analysis Timepoints

Elise

2024-06-09

Set the right working directory.

```
rm(list = ls())

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

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")
timeseries <- read.table("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
matching_cols <- intersect(names(timeseries), names(plant_info))
timeseries[, matching_cols] <- lapply(timeseries[, matching_cols], factor)
timeseries$Date <- date(timeseries$Date)
timeseries$Timestamp <- NA

platform <- "FZJ"

# 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
variables_t <- c("Manual_Plant_height_cm")


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

```
## [1] "FZJ : The variables for timeseries are Manual_Plant_height_cm"
```

```
endpoint$Plant_type <- substr(endpoint$Genotype, nchar(as.character(endpoint$Genotyp
e)), nchar(as.character(endpoint$Genotype)))
timeseries$Plant_type <- substr(timeseries$Genotype, nchar(as.character(timeseries$Geno
type)), nchar(as.character(timeseries$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
timeseries$Plant_type <- substr(timeseries$Genotype, nchar(as.character(timeseries$Geno
type)), nchar(as.character(timeseries$Genotype)))

timeseries$Plant_type <- ifelse(timeseries$Plant_type %in% c("T", "L"), "Line",
                                ifelse(timeseries$Plant_type == "H", "Hybrid", timese
ries$Plant_type))
```

Time point objects

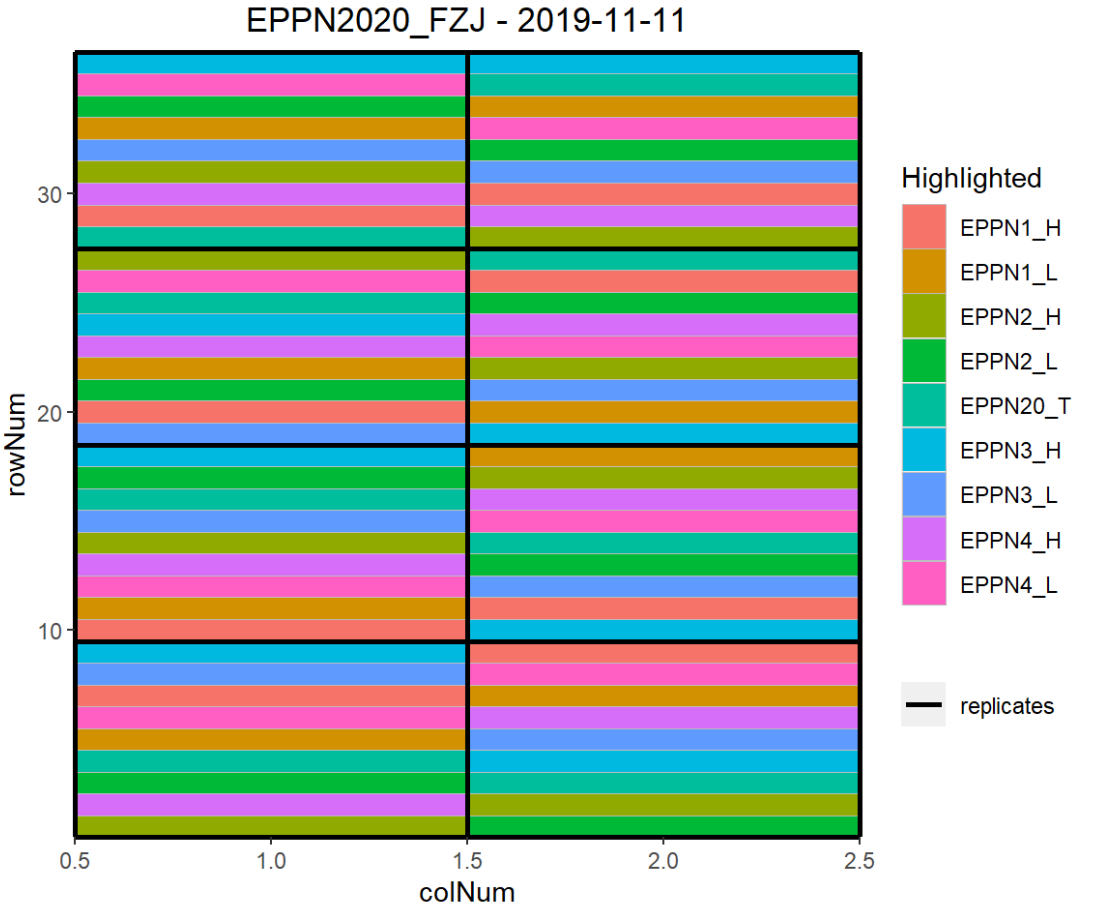
Generation of the timePoints objects using the function “createTimePoints”.

```
timePoint_endpoint <- createTimePoints(dat = endpoint,
                                       experimentName = "EPPN2020_FZJ",
                                       genotype = "Genotype",
                                       timePoint = "Date",
                                       plotId = "Unit.ID",
                                       rowNum = "Row",
                                       colNum = "Column",
                                       repId = "Replication")

timePoint_endpoint_clean <- createTimePoints(dat = endpoint_clean,
                                       experimentName = "EPPN2020_FZJ",
                                       genotype = "Genotype",
                                       timePoint = "Date",
                                       plotId = "Unit.ID",
                                       rowNum = "Row",
                                       colNum = "Column",
                                       repId = "Replication")

timePoint_S <- createTimePoints(dat = timeseries,
                                experimentName = "EPPN2020_FZJ",
                                genotype = "Genotype",
                                timePoint = "Date",
                                plotId = "Unit.ID",
                                rowNum = "Row",
                                colNum = "Column",
                                repId = "Replication",
                                addCheck = TRUE,
                                checkGenotypes = "EPPN20_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_FZJ.
##
## It contains 1 time points.
## First time point: 2019-11-11
## Last time point: 2019-11-11
##
## No check genotypes are defined.
```

```
getTimePoints(timePoint_endpoint)
```

```
##   timeNumber  timePoint
## 1          1 2019-11-11
```

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)
}
```

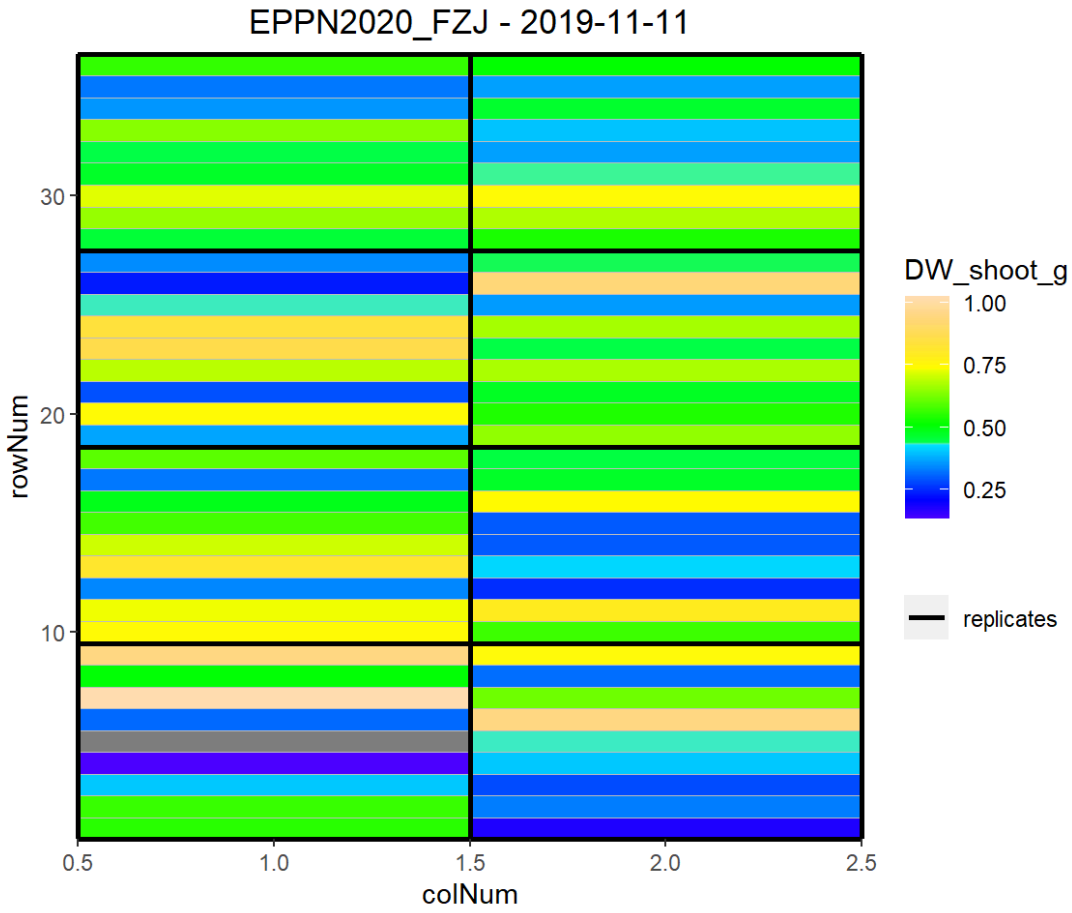
```
## [1] "How many data observations for DW_shoot_g"
## 2019-11-11
##         71
## [1] "How many data observations for FW_shoot_g"
## 2019-11-11
##         69
## [1] "How many data observations for DW_root_g"
## 2019-11-11
##         72
## [1] "How many data observations for Root_length_cm"
## 2019-11-11
##         72
## [1] "How many data observations for Root_number"
## 2019-11-11
##         67
## [1] "How many data observations for Root_angle"
## 2019-11-11
##         42
```

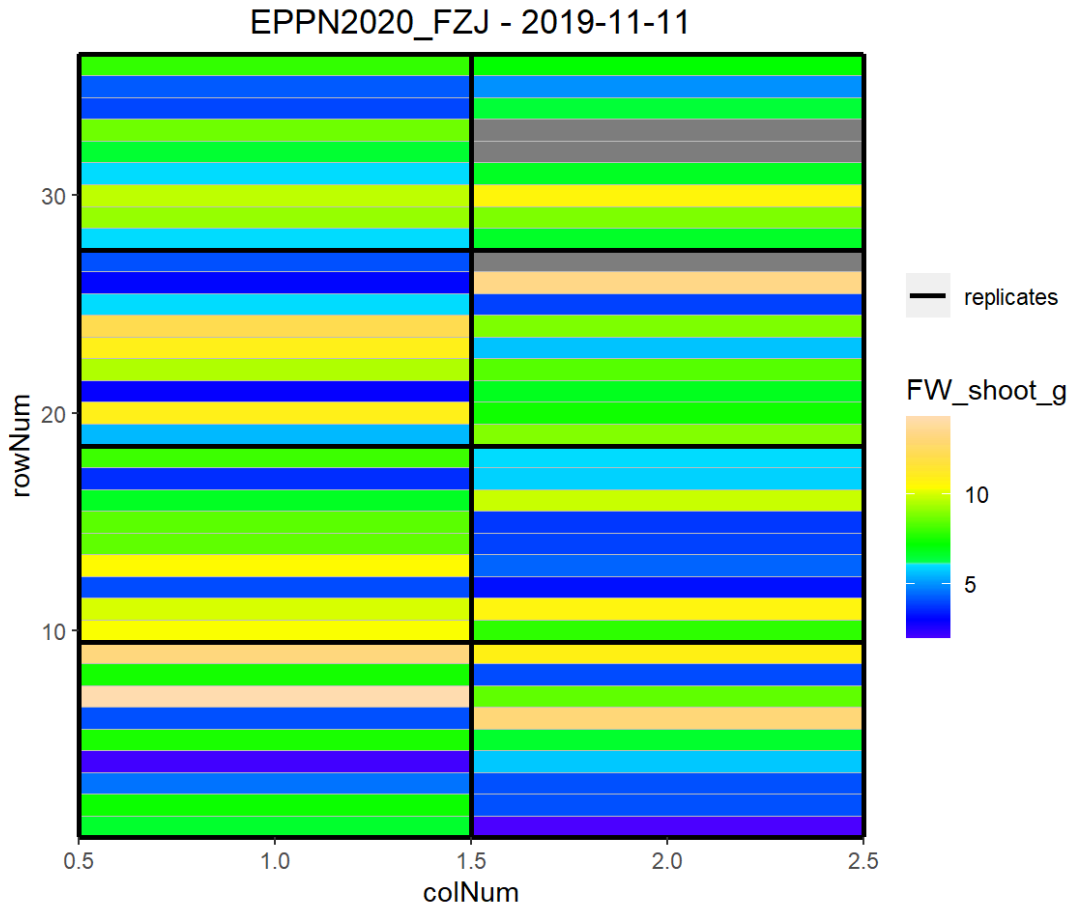
```
for (trait_name in traits) {
  print(paste("How many cleaned data observations for", trait_name))
  num_observations <- countValid(timePoint_endpoint_clean, trait_name)
  print(num_observations)
}
```

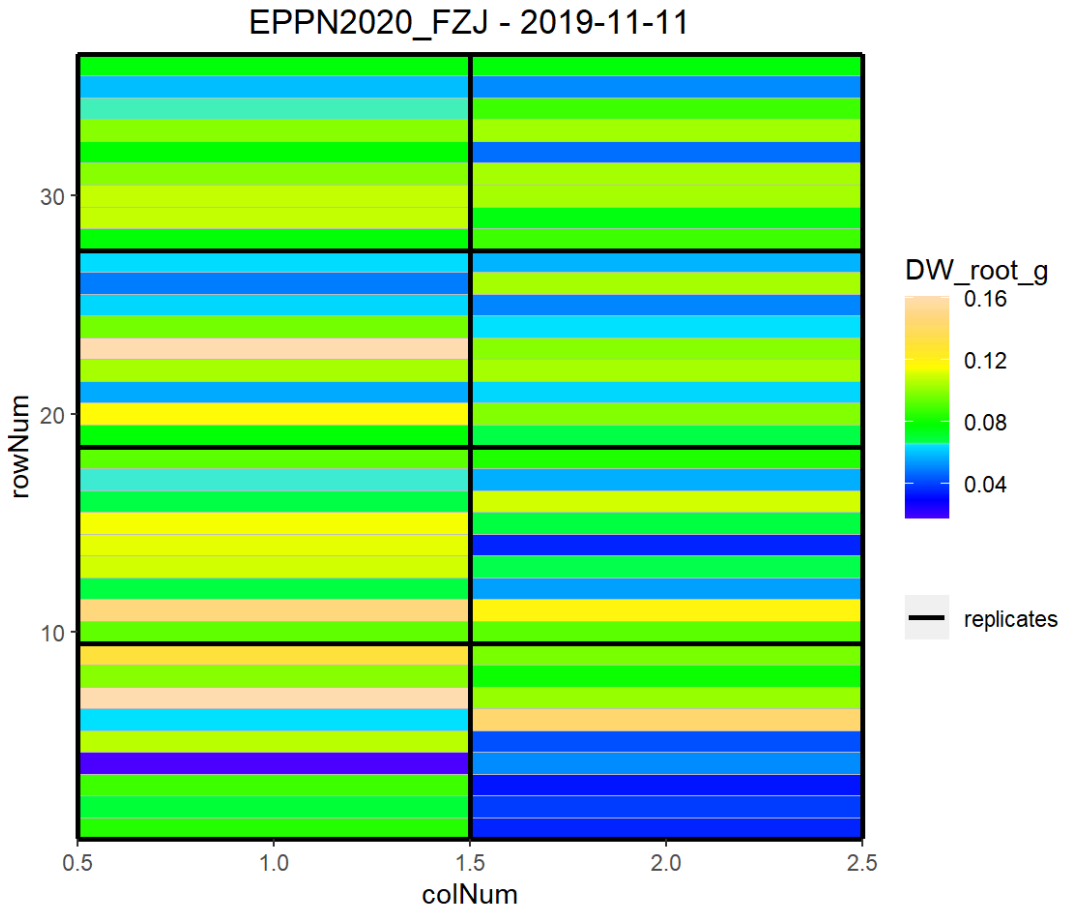
```
## [1] "How many cleaned data observations for DW_shoot_g"
## 2019-11-11
##         65
## [1] "How many cleaned data observations for FW_shoot_g"
## 2019-11-11
##         61
## [1] "How many cleaned data observations for DW_root_g"
## 2019-11-11
##         69
## [1] "How many cleaned data observations for Root_length_cm"
## 2019-11-11
##         71
## [1] "How many cleaned data observations for Root_number"
## 2019-11-11
##         63
## [1] "How many cleaned data observations for Root_angle"
## 2019-11-11
##         40
```

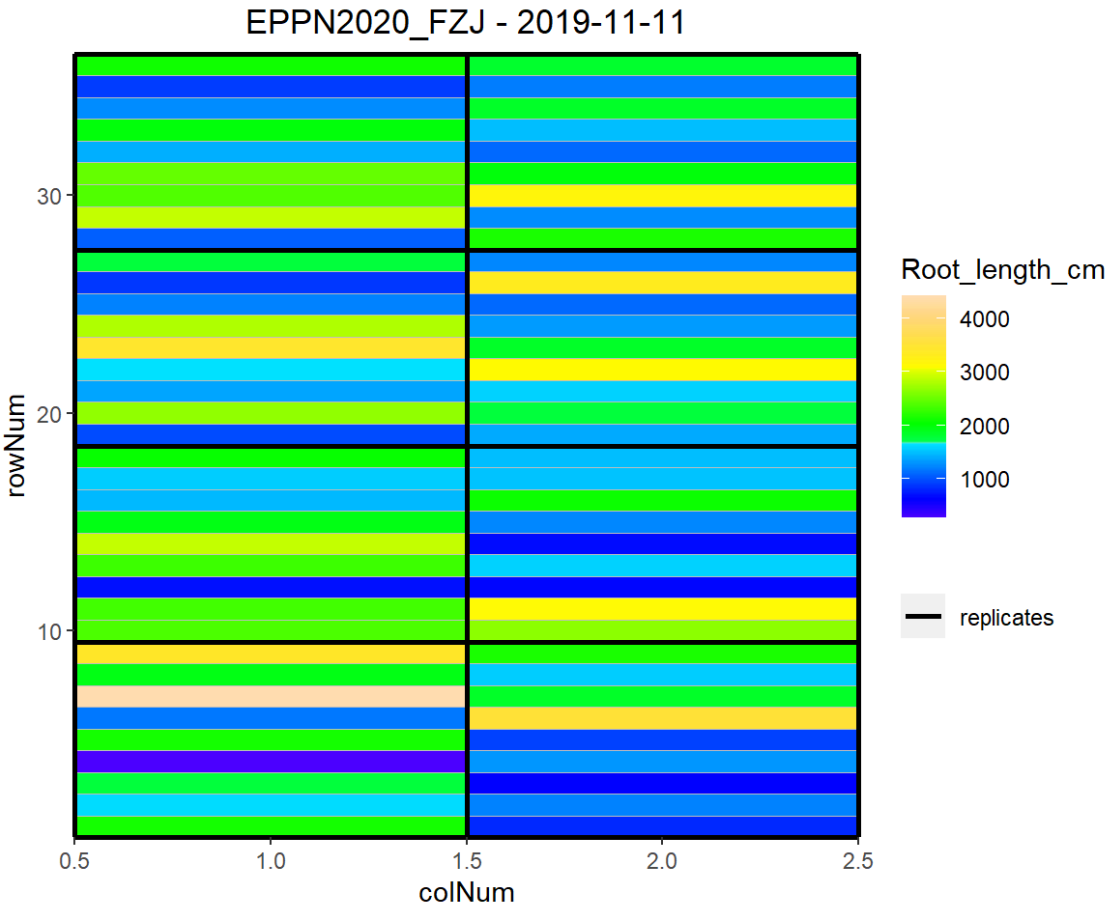
Check the heatmap of the data at harvest

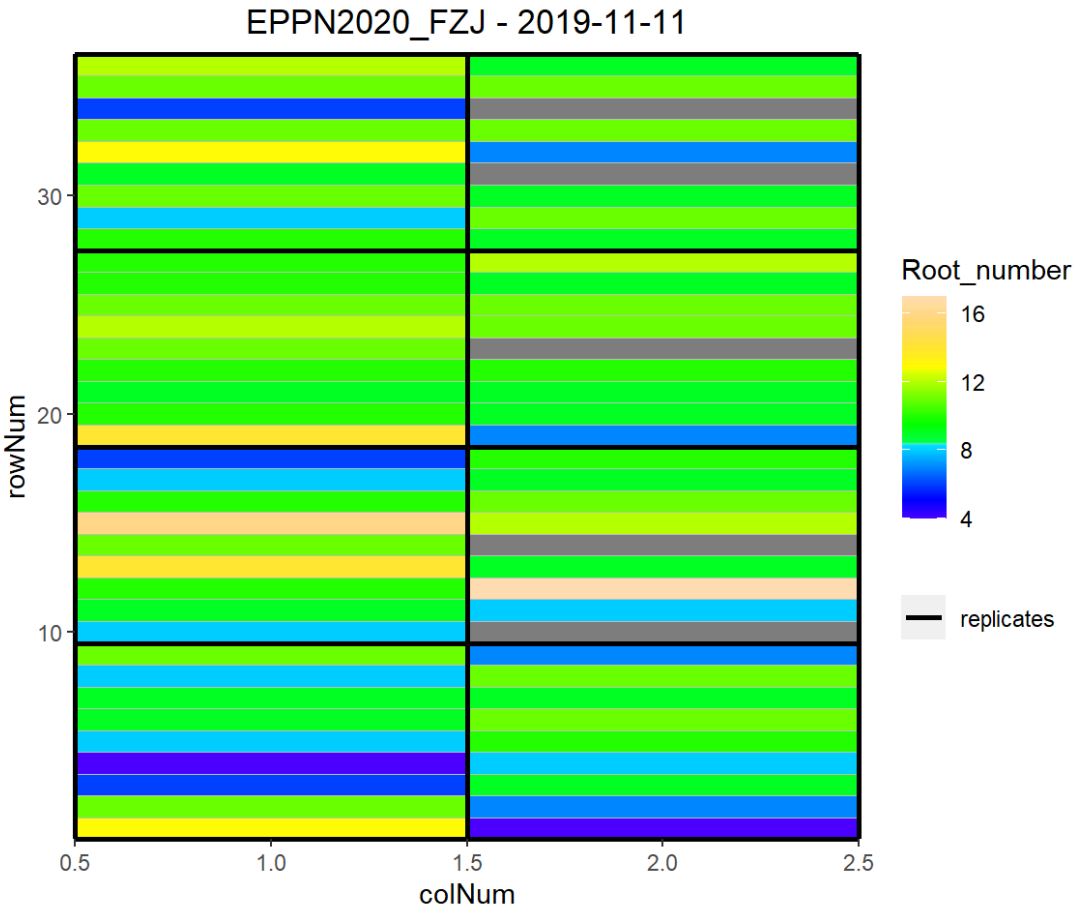
```
for (trait_name in traits) {
  plot(timePoint_endpoint,
     plotType = "layout",
     timePoints = 1,
     traits = trait_name)
}
```

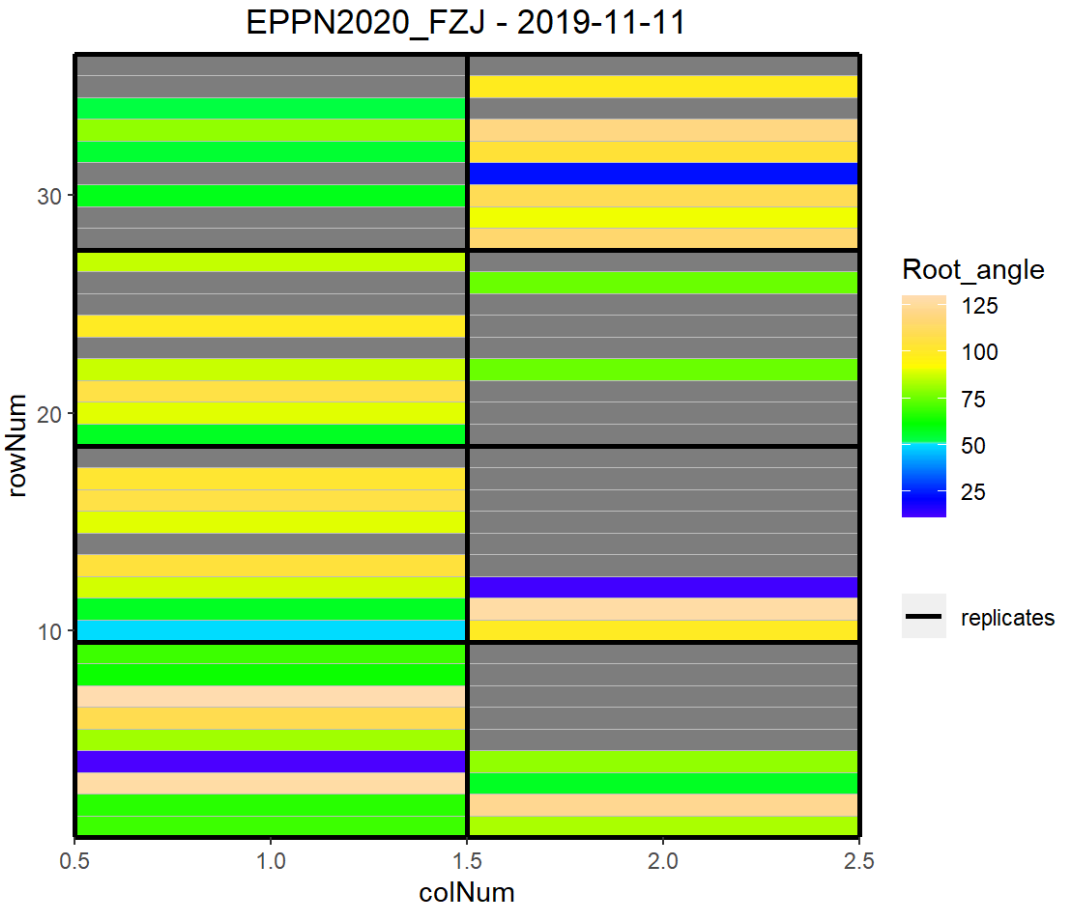

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
DW_shoot_g
1.00
0.75
0.50
0.25
replicates


EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
replicates
FW_shoot_g
10
5

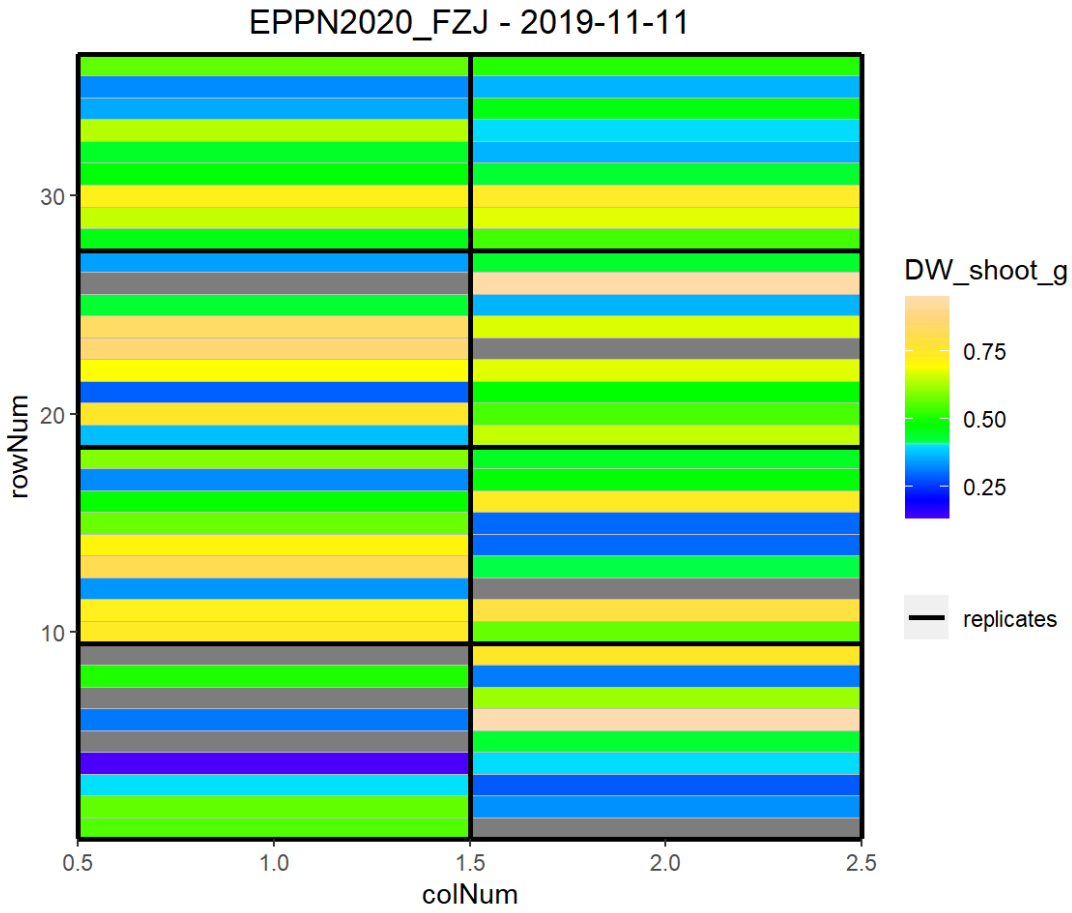

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
DW_root_g
0.16
0.12
0.08
0.04
replicates

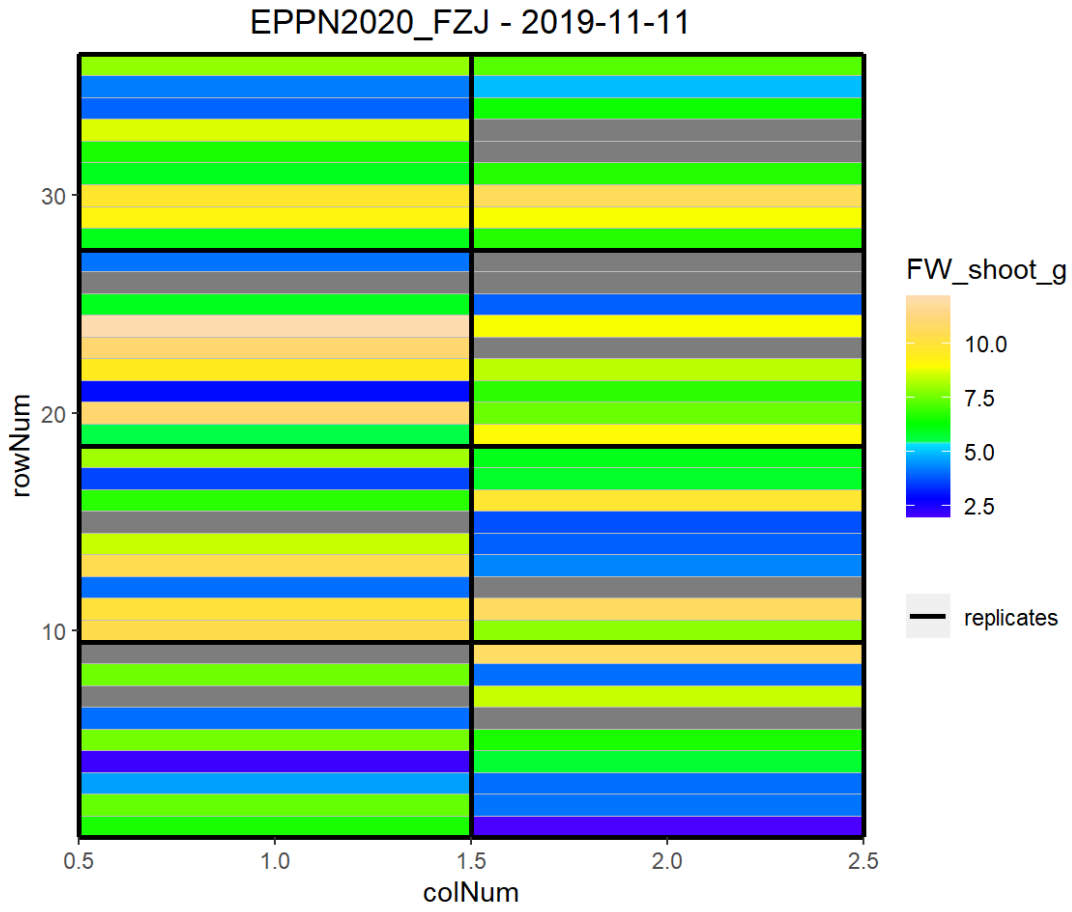

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Root_length_cm
4000
3000
2000
1000
replicates

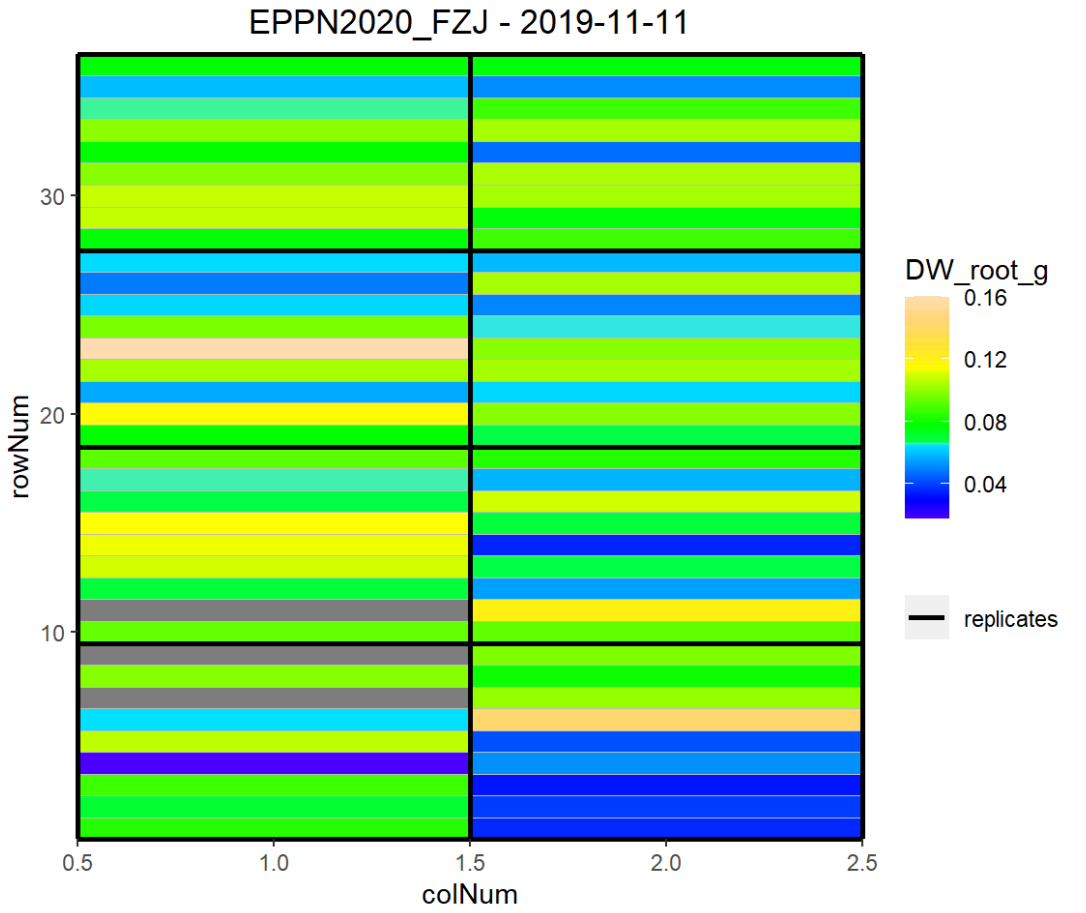

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Root_number
16
12
8
4
replicates

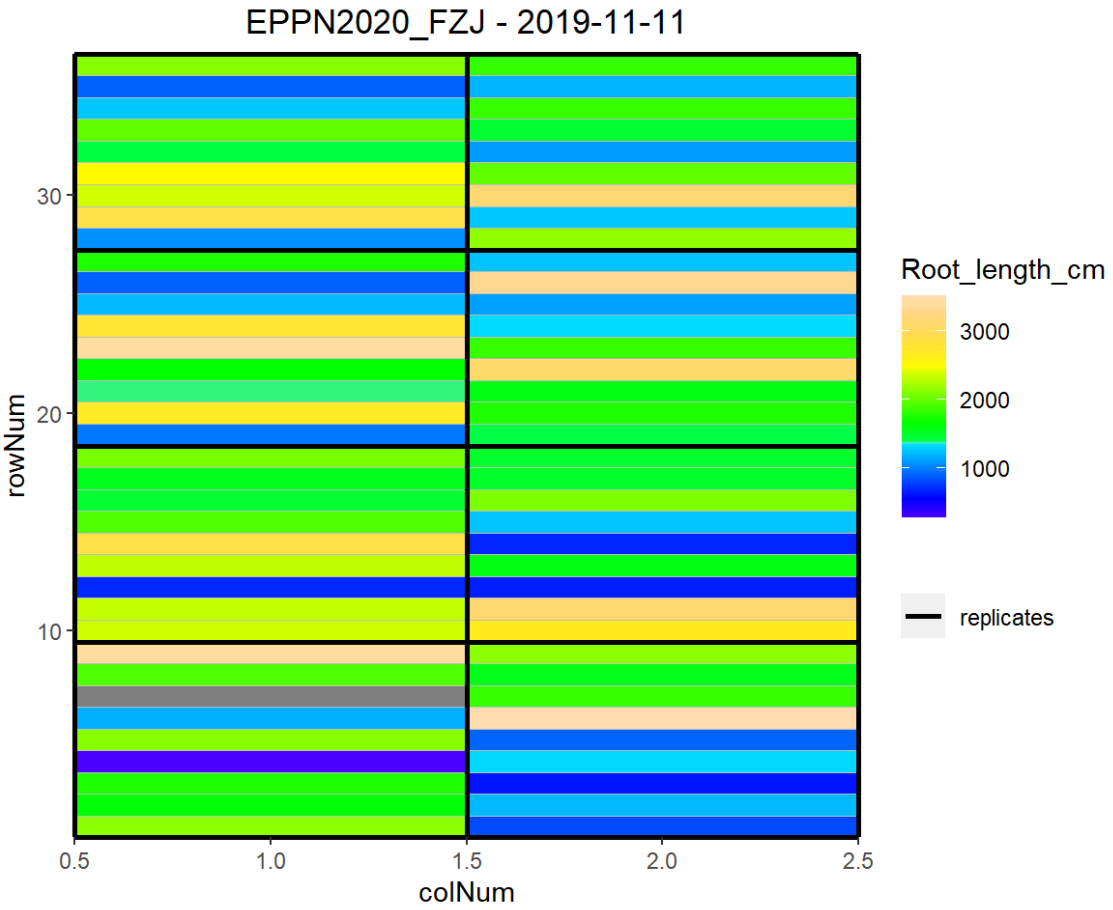

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Root_angle
125
100
75
50
25
replicates

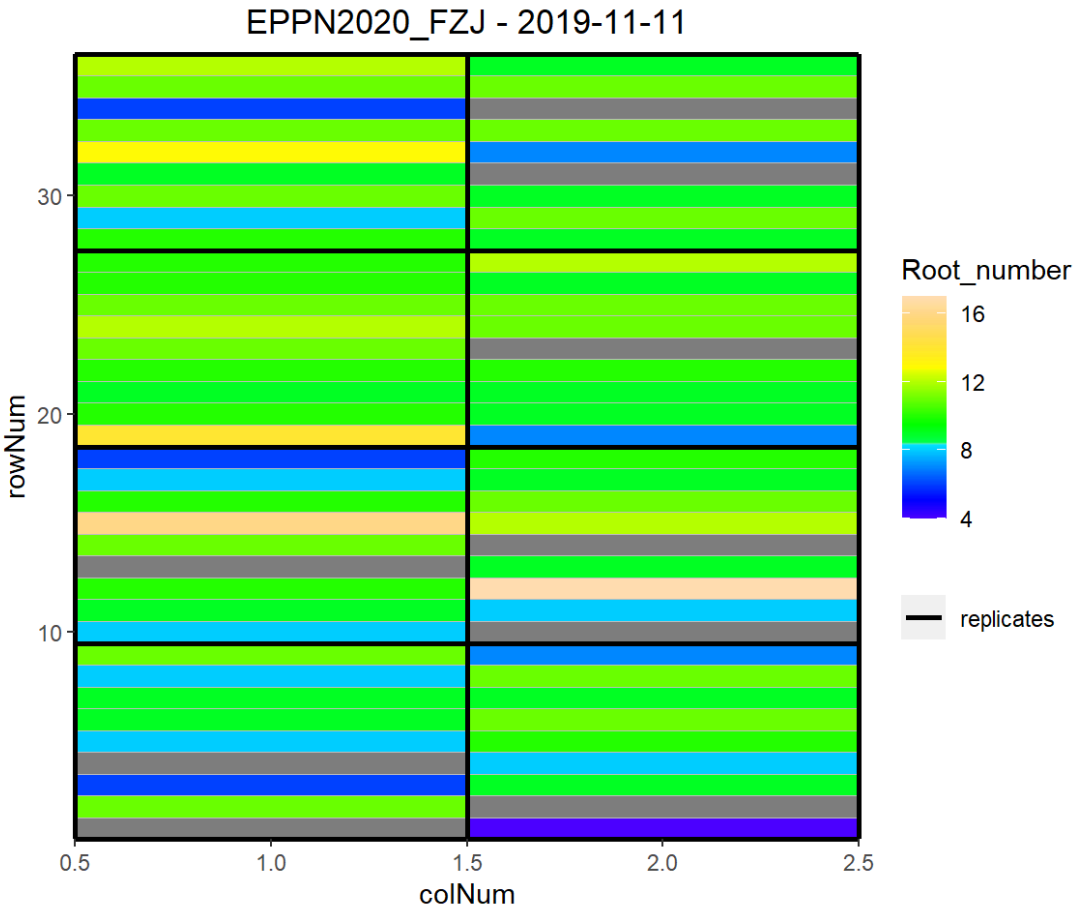
```
for (trait_name in traits) {
  plot(timePoint_endpoint_clean,
     plotType = "layout",
     timePoints = 1,
     traits = trait_name)
}
```

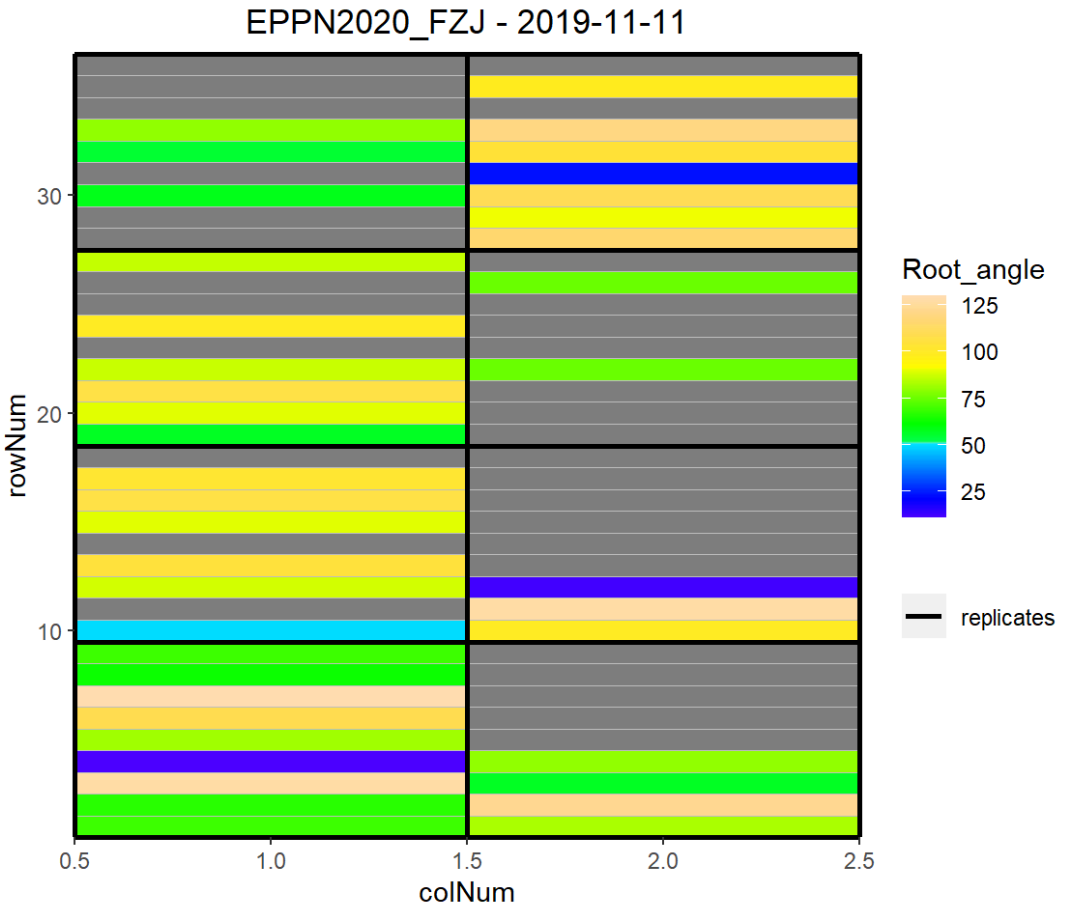

EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
DW_shoot_g
0.75
0.50
0.25
replicates


EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
FW_shoot_g
10.0
7.5
5.0
2.5
replicates


EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
DW_root_g
0.16
0.12
0.08
0.04
replicates


EPPN2020_FZJ - 2019-11-11
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Root_length_cm
3000
2000
1000
replicates





2. timeseries

Raw data

View timePoint object

```
summary(timePoint_S)
```

```
## timePoint_S contains data for experiment EPPN2020_FZJ.
##
## It contains 5 time points.
## First time point: 2019-10-28
## Last time point: 2019-11-11
##
## The following genotypes are defined as check genotypes: EPPN20_T.
```

```
getTimePoints(timePoint_S)
```

```
##   timeNumber  timePoint
## 1          1 2019-10-28
## 2          2 2019-10-31
## 3          3 2019-11-04
## 4          4 2019-11-07
## 5          5 2019-11-11
```

```
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.

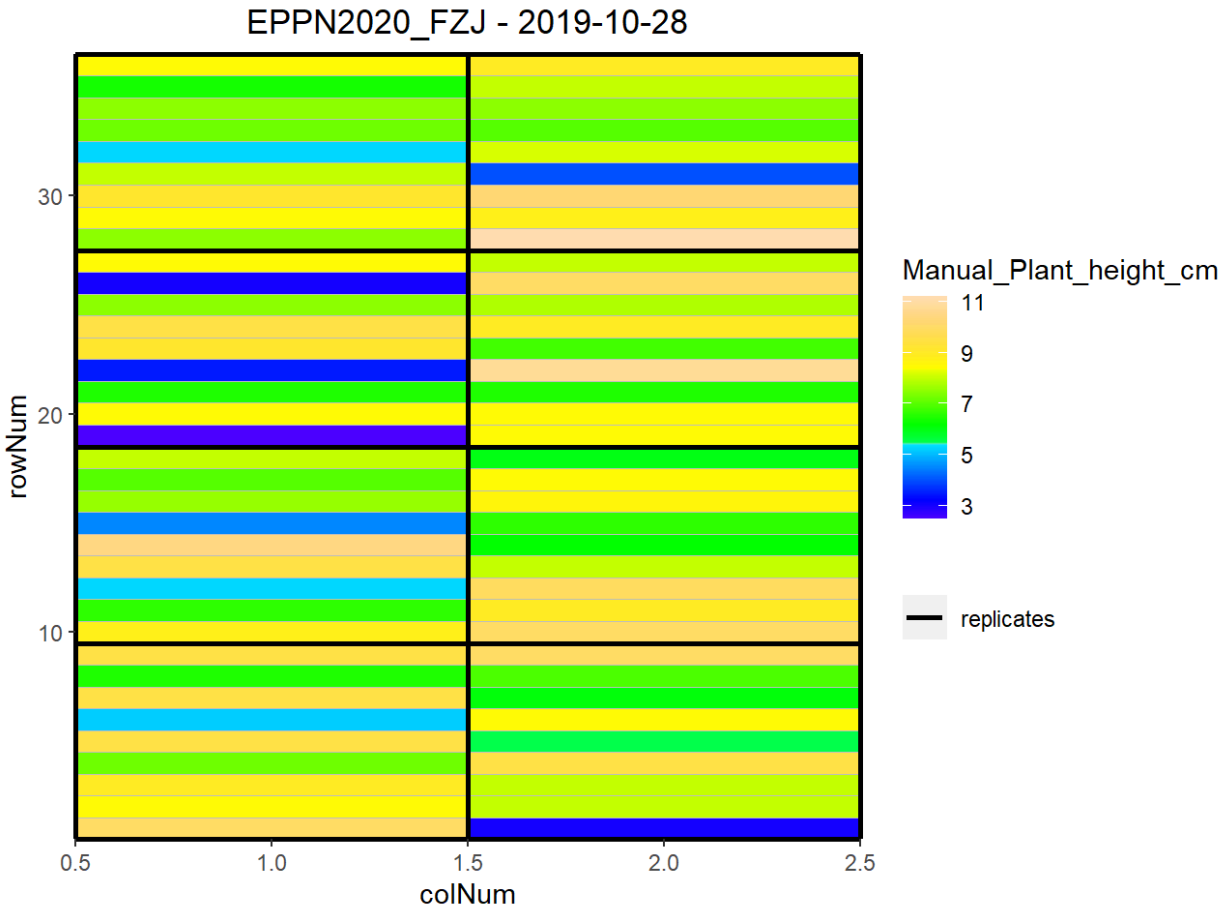
Height is computed for 6 platforms out of 9 and area for 4 out of 9.

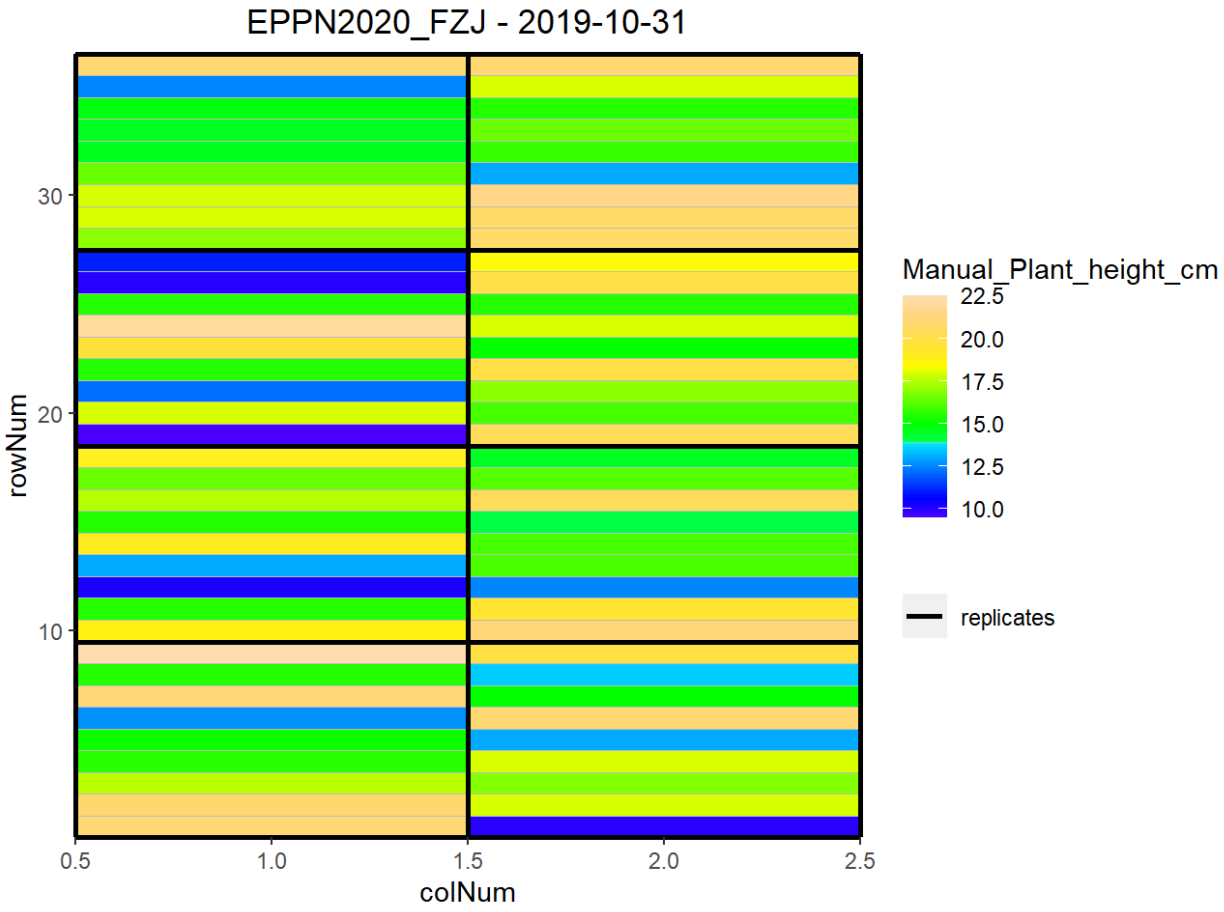
```
var_voulues <- c(variables_t)
traits <- var_voulues

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

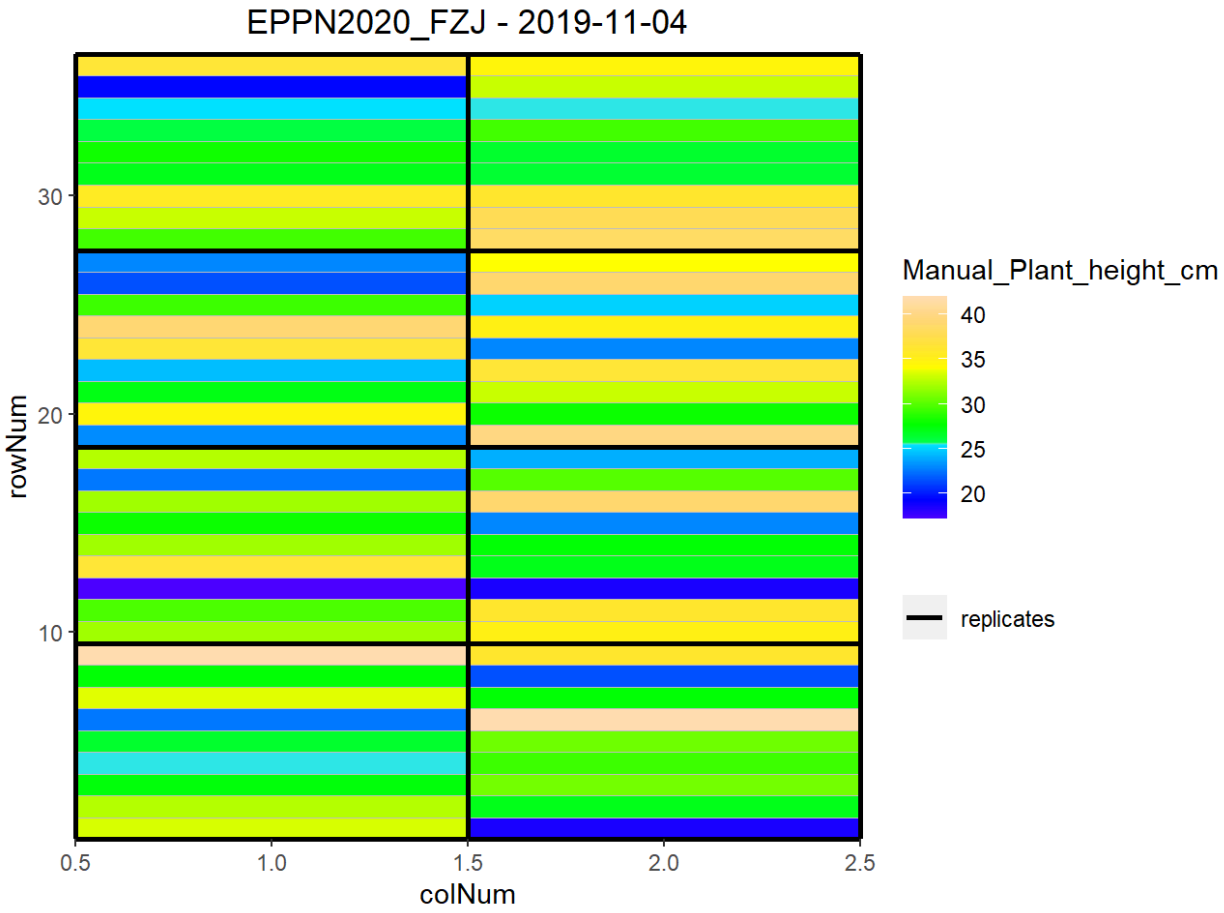
```
## [1] "How many observations for Manual_Plant_height_cm"
## 2019-10-28 2019-10-31 2019-11-04 2019-11-07 2019-11-11
##         72         72         72         72         70
```

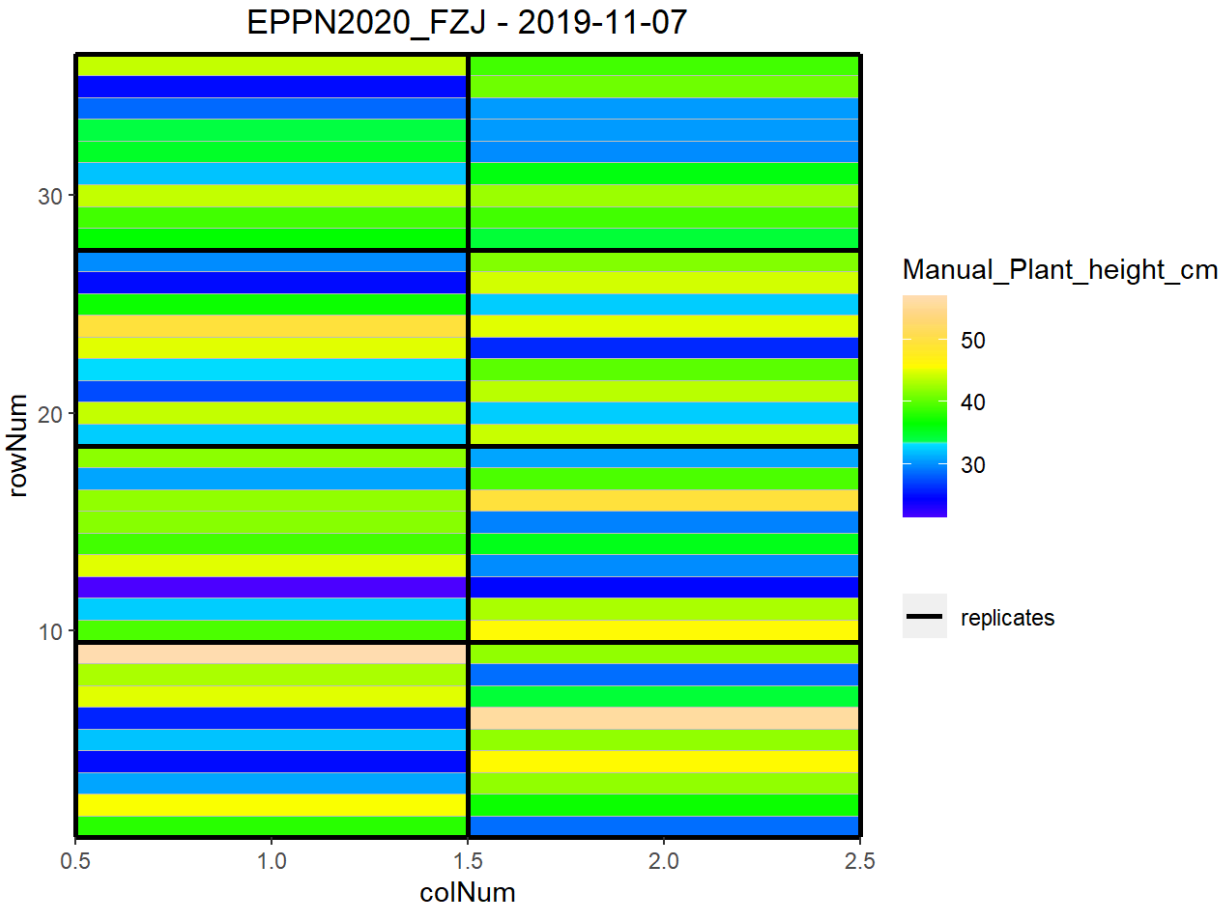
Check the heatmap of the raw data per time point

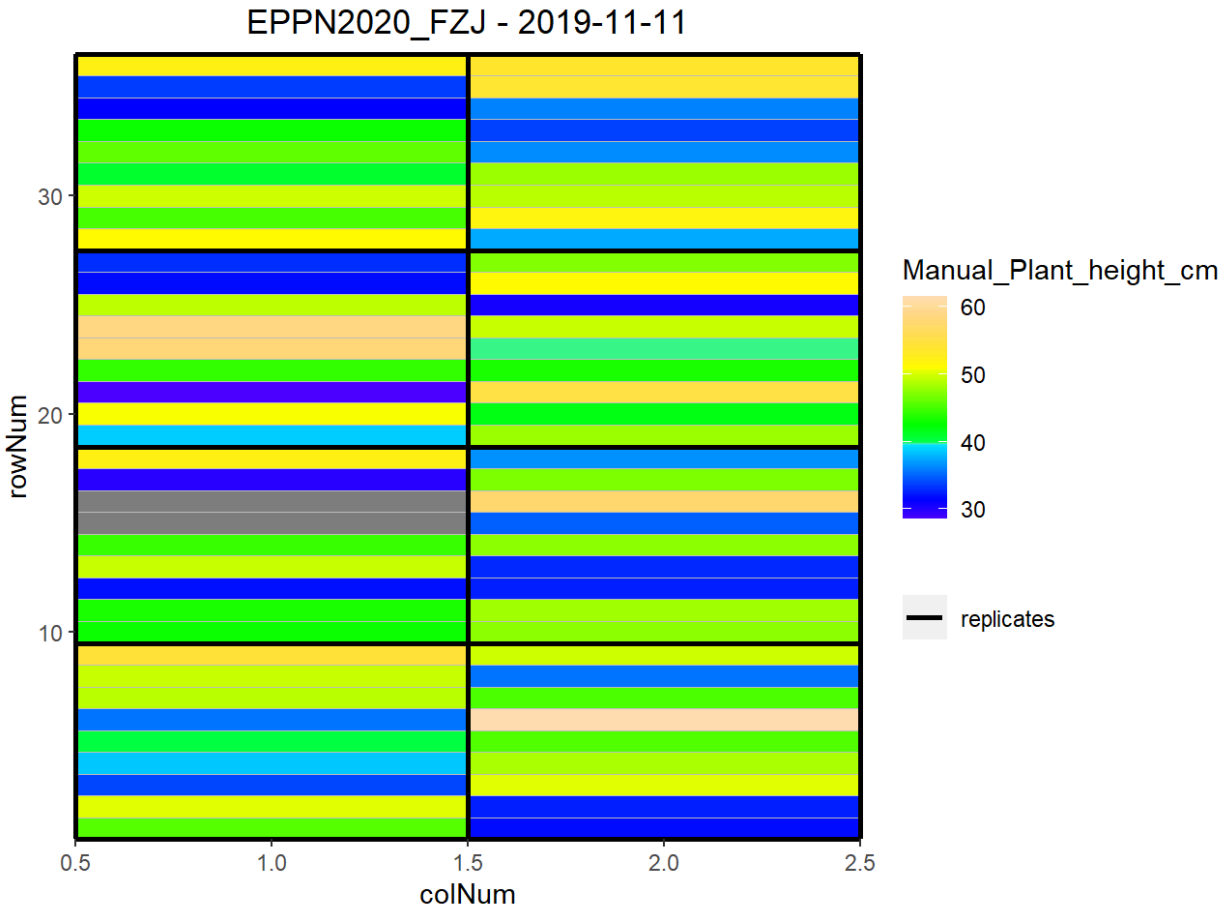
```
for (trait_name in traits) {
  for (tp in 1:length(num_timepoints$timeNumber)) {
    plot(timePoint_S,
         plotType = "layout",
         timePoints = tp,
         traits = trait_name)
  }
}
```


EPPN2020_FZJ - 2019-10-28
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Manual_Plant_height_cm
11
9
7
5
3
replicates


EPPN2020_FZJ - 2019-10-31
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Manual_Plant_height_cm
22.5
20.0
17.5
15.0
12.5
10.0
replicates

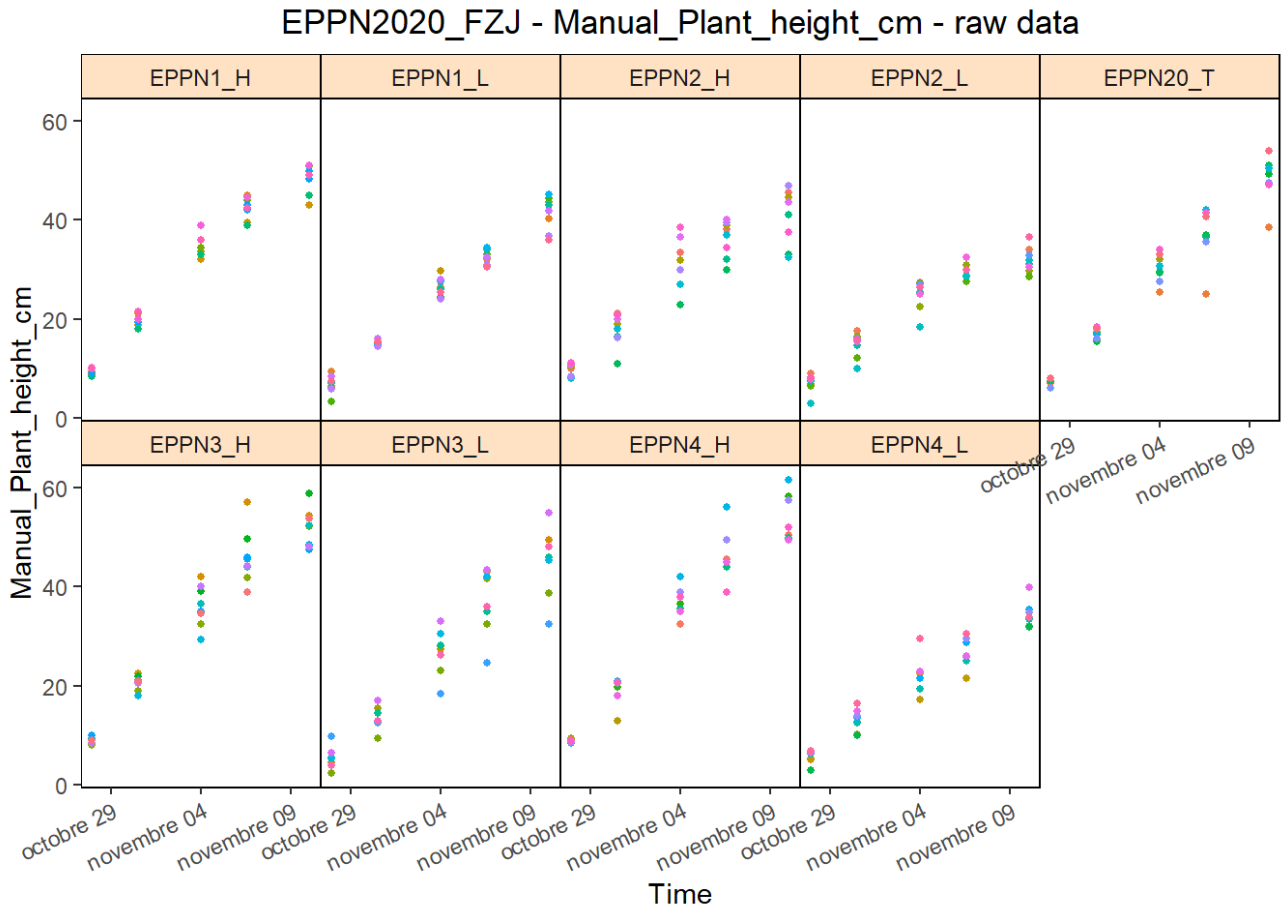

EPPN2020_FZJ - 2019-11-04
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Manual_Plant_height_cm
40
35
30
25
20
replicates


EPPN2020_FZJ - 2019-11-07
rowNum
30
20
10
0.5
1.0
1.5
2.0
2.5
colNum
Manual_Plant_height_cm
50
40
30
replicates



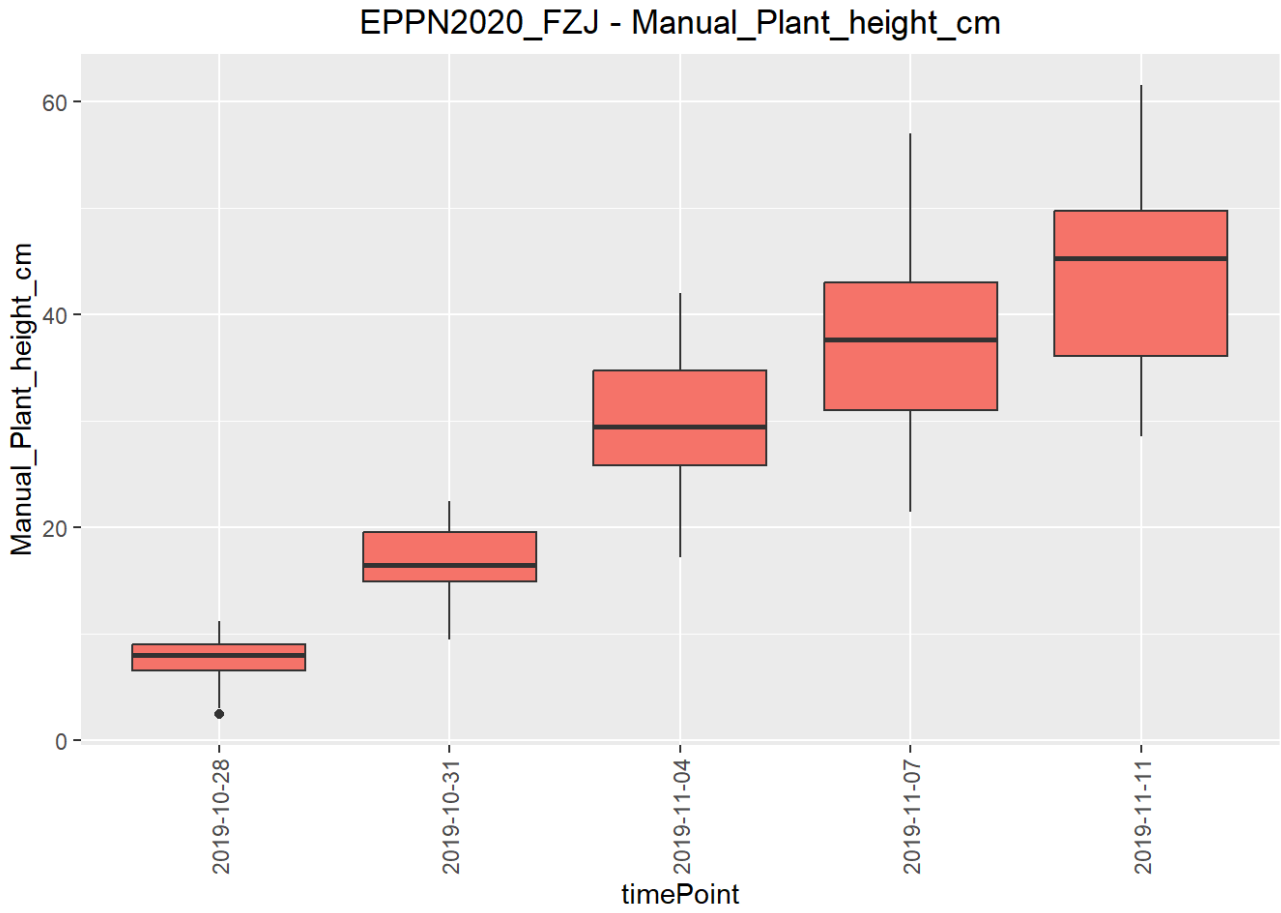
Check time course of raw data per time point

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



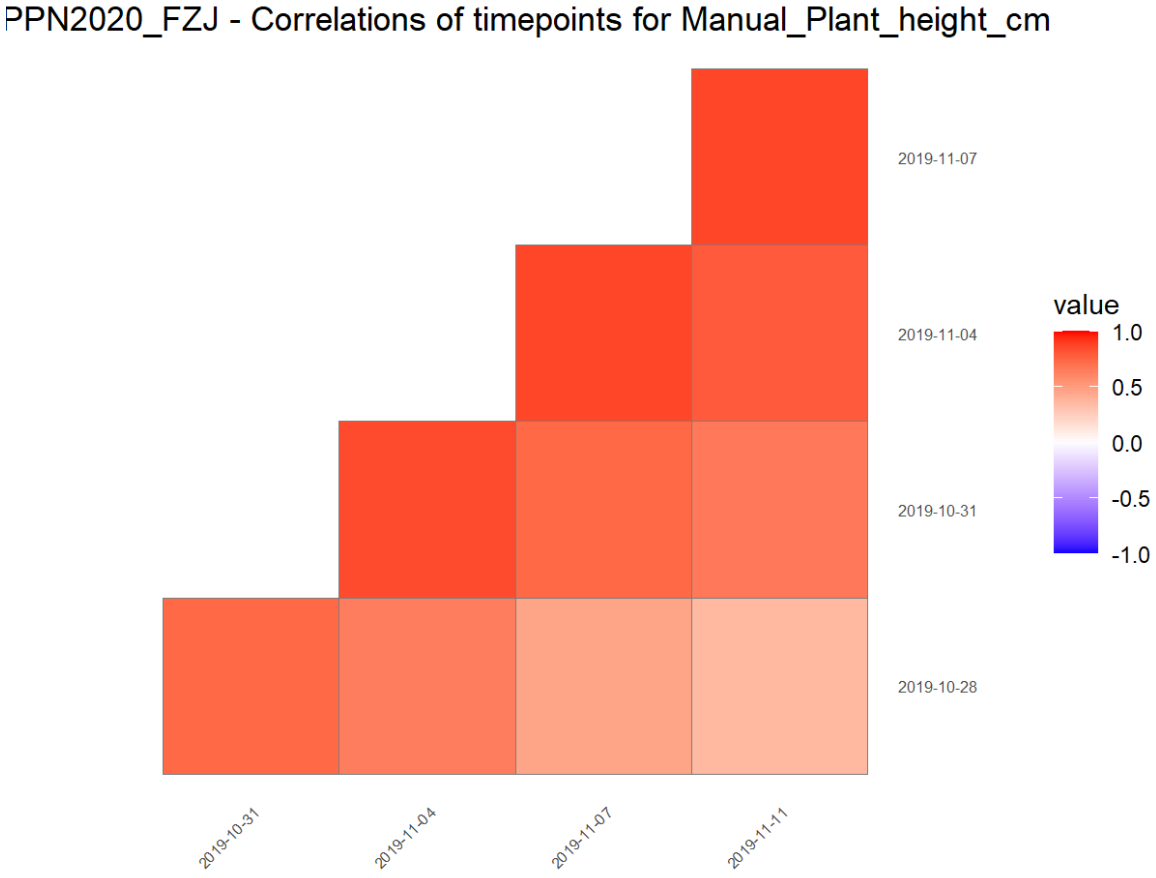
Check the boxplots of raw data per time point

```
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)
#}

print("Erreur dans detectSingleOut(TP = timePoint_S, trait = trait_name, plotIds = plan
tSel,  :
  Not enough data points (<= 6) for any of the plots.")
```

```
## [1] "Erreur dans detectSingleOut(TP = timePoint_S, trait = trait_name, plotIds = pla
ntSel,  : \n  Not enough data points (<= 6) for any of the plots."
```

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

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")
  }
}
```

```
## No object Single_outliers found for trait Manual_Plant_height_cm
```

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")
  }
}
```

```
## No Single_outliers object found for trait Manual_Plant_height_cm
```

2. Correction for spatial trends

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

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")
  }
}
```

```
## No model found for Manual_Plant_height_cm
```

```
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")
  }
}
```

```
## No model found for the trait Manual_Plant_height_cm
```

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")
  }
}
```

```
## No model found for Manual_Plant_height_cm
```

Plot the splines for a plant selection

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")
  }
}
```

```
## No corrected data or prediction data found for Manual_Plant_height_cm
```

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")
  }
}
```

```
## No corrected data or series test data found for Manual_Plant_height_cm
```

```
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")
  }
}
```

```
## No corrected data found for Manual_Plant_height_cm
```

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

Estimation of parameter from time series