ALSIA Data Analysis

Elise

2024-06-09

Set the right working directory.

```
setwd("C:/Users/elise/Documents/Mémoire/Main/Data/Templates/ALSIA")
```

Data importation

Import the data sets extracted from the Data Preparation 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")
S_timeseries <- read.table("S_timeseries.txt", header = TRUE, sep = "\t")
T_timeseries <- read.table("T_timeseries.txt", header = TRUE, sep = "\t")
```

Convert the columns to factor and date formats.

```
# 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$Timestamp <- as.POSIXct(timeseries$Timestamp, format = "%Y-%m-%d %H:%M:%S")
timeseries$Date <- date(timeseries$Date)

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

Collect the variables of every data template and print the names of the variables. This serves as a double check.

```
platform <- "ALSIA"

# endpoint
df <- endpoint[,colSums(is.na(endpoint))<nrow(endpoint)]
genotype_index <- which(colnames(df) == "Genotype")
variables <- colnames(df[, c(4:(genotype_index - 1))]) # We remove the 3 first columns th
at are "Unit.ID" and "Date" etc

# timeseries
variables_t <- "Wue"

# 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 <- colnames(df_S_timeseries[, c(5:(genotype_index - 1))]) # We remove the thr
ee first columns that are "Unit.ID","Time" and "Date"

# T_timeseries
df_T_timeseries <- T_timeseries[,colSums(is.na(T_timeseries))<nrow(T_timeseries)]
genotype_index <- which(colnames(df_T_timeseries) == "Genotype")
variables_T <- colnames(df_T_timeseries[, c(5:(genotype_index - 1))]) # We remove the thr
ee first columns that are "Unit.ID","Time" and "Date"

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

```
## [1] "ALSIA : The variables for endpoint are FW_shoot_g, Plant_height_cm"
```

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

```
## [1] "ALSIA : The variables for timeseries are Wue"
```

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

```
## [1] "ALSIA : The variables for S_timeseries are S_Height_cm, S_Area_cmsquared, S_Conve
x_hull_area_cmsquared, S_Solidity, S_Leaf_area_cmsquared"
```

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

```
## [1] "ALSIA : The variables for T_timeseries are T_Area_cmsquared, T_Convex_hull_area_c
msquared, T_Solidity"
```

Add a column Plant_type with three levels, H L and T. This variable is useful to test for heterosis effects.

```
endpoint$Plant_type <- substr(endpoint$Genotype, nchar(as.character(endpoint$Genotype)), nchar(as.character(endpoint$Genotype)))
timeseries$Plant_type <- substr(timeseries$Genotype, nchar(as.character(timeseries$Genotype)), nchar(as.character(timeseries$Genotype)))
S_timeseries$Plant_type <- substr(S_timeseries$Genotype, nchar(as.character(S_timeseries$Genotype)), nchar(as.character(S_timeseries$Genotype)))
T_timeseries$Plant_type <- substr(T_timeseries$Genotype, nchar(as.character(T_timeseries$Genotype)), nchar(as.character(T_timeseries$Genotype)))
```

1. Endpoint dataframe

A. Exploration of data

Exploration tables using the rstatix, janitor and skimr packages

```
endpoint %>%
  count(Genotype)
```

```
##   Genotype  n
## 1  EPPN1_H 10
## 2  EPPN1_L 10
## 3  EPPN2_H 10
## 4  EPPN2_L 10
## 5 EPPN20_T 10
## 6  EPPN3_H 10
## 7  EPPN3_L 10
## 8  EPPN4_H 10
## 9  EPPN4_L 10
```

```
endpoint %>%
  tabyl(Genotype, Column) %>%
  adorn_totals("row") %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting() %>%
  adorn_ns() %>%
  adorn_title("combined")
```

```
##  Genotype/Column         14         15         16         17         18
##          EPPN1_H 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN1_L 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN2_H 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN2_L 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##         EPPN20_T 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN3_H 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN3_L 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN4_H 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##          EPPN4_L 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2) 20.0%  (2)
##            Total 20.0% (18) 20.0% (18) 20.0% (18) 20.0% (18) 20.0% (18)
```

```
endpoint %>%
  tabyl(Genotype, Row) %>%
  adorn_totals("row") %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting() %>%
  adorn_ns() %>%
  adorn_title("combined")
```

```
##  Genotype/Row         1         2         3         4         5         6
##       EPPN1_H  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0) 20.0% (2)  0.0% (0)
##       EPPN1_L 10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1) 10.0% (1)
##       EPPN2_H  0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)
##       EPPN2_L  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##      EPPN20_T  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##       EPPN3_H 20.0% (2)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##       EPPN3_L  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##       EPPN4_H 10.0% (1)  0.0% (0) 10.0% (1) 20.0% (2)  0.0% (0)  0.0% (0)
##       EPPN4_L 10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)
##         Total  5.6% (5)  3.3% (3)  4.4% (4)  2.2% (2)  5.6% (5)  3.3% (3)
##          7         8         9        10        11        12        13
##   0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1)
##   0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0) 10.0% (1)
##  10.0% (1)  0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0) 10.0% (1) 20.0% (2)
##  10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1) 10.0% (1)
##   0.0% (0)  0.0% (0) 20.0% (2)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0)
##  10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##   3.3% (3)  2.2% (2)  4.4% (4)  4.4% (4)  3.3% (3)  3.3% (3)  5.6% (5)
##         14        15        16        17        18        19        20
##  10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##   0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1) 10.0% (1) 10.0% (1)
##   0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1)
##   0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0)
##  20.0% (2)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##   0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 20.0% (2)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0) 20.0% (2)  0.0% (0)  0.0% (0)  0.0% (0)
##  10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1) 20.0% (2)
##   4.4% (4)  3.3% (3)  3.3% (3)  3.3% (3)  4.4% (4)  5.6% (5)  4.4% (4)
##         21        22        23        24        25        26
##   0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1)
##  10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##   0.0% (0) 20.0% (2)  0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)
##  10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0) 20.0% (2)  0.0% (0)  0.0% (0)
##   0.0% (0)  0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0) 10.0% (1)
##   0.0% (0) 10.0% (1)  0.0% (0)  0.0% (0)  0.0% (0) 10.0% (1)
##   0.0% (0) 10.0% (1) 10.0% (1)  0.0% (0) 10.0% (1)  0.0% (0)
##   0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)  0.0% (0)
##   2.2% (2)  4.4% (4)  3.3% (3)  3.3% (3)  3.3% (3)  3.3% (3)
```

```
endpoint %>%
  count(Genotype)
```

```
##   Genotype  n
## 1  EPPN1_H 10
## 2  EPPN1_L 10
## 3  EPPN2_H 10
## 4  EPPN2_L 10
## 5 EPPN20_T 10
## 6  EPPN3_H 10
## 7  EPPN3_L 10
## 8  EPPN4_H 10
## 9  EPPN4_L 10
```

```
get_summary_stats(data = endpoint,
                  variables,
                  type = "common")
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(variables)
##
##   # Now:
##   data %>% select(all_of(variables))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 2 × 10
##   variable            n   min   max median   iqr  mean    sd    se    ci
##   <fct>           <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 FW_shoot_g         90  16.5  161.   84.8  60.3  79.1  36.0  3.79  7.53
## 2 Plant_height_cm    90  74    156   114.   32.8 113.   20.9  2.21  4.38
```

```
skim(endpoint[variables])
```

Data summary

Name	endpoint[variables]
Number of rows	90
Number of columns	2
_____	
Column type frequency:	
numeric	2
_____	

Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
FW_shoot_g	0	1	79.14	35.95	16.5	45.20	84.75	105.5	161.1	▆▂▇▆▁
Plant_height_cm	0	1	113.31	20.93	74.0	97.25	114.50	130.0	156.0	▆▇▆▅▂

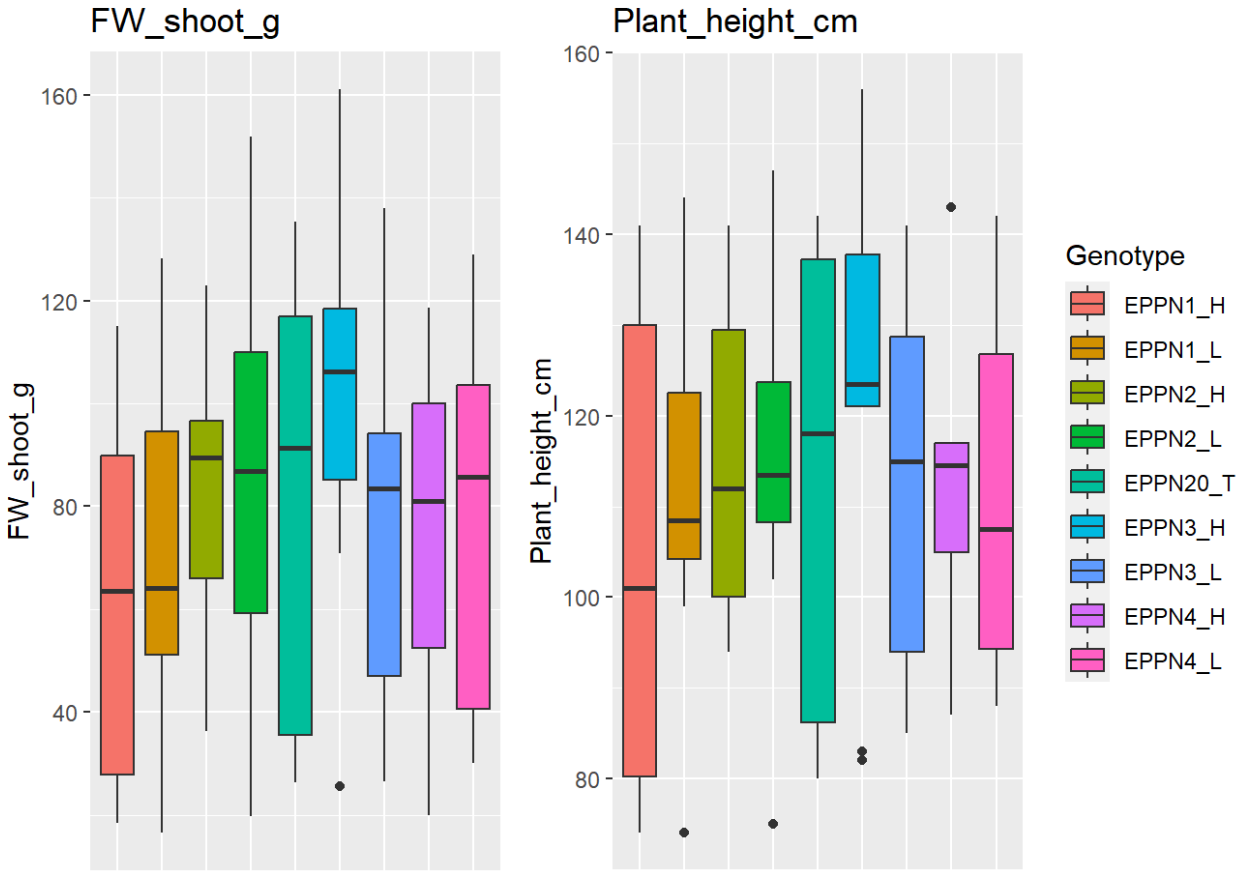
Data visualization

Using several functions that are located in the functions.R script

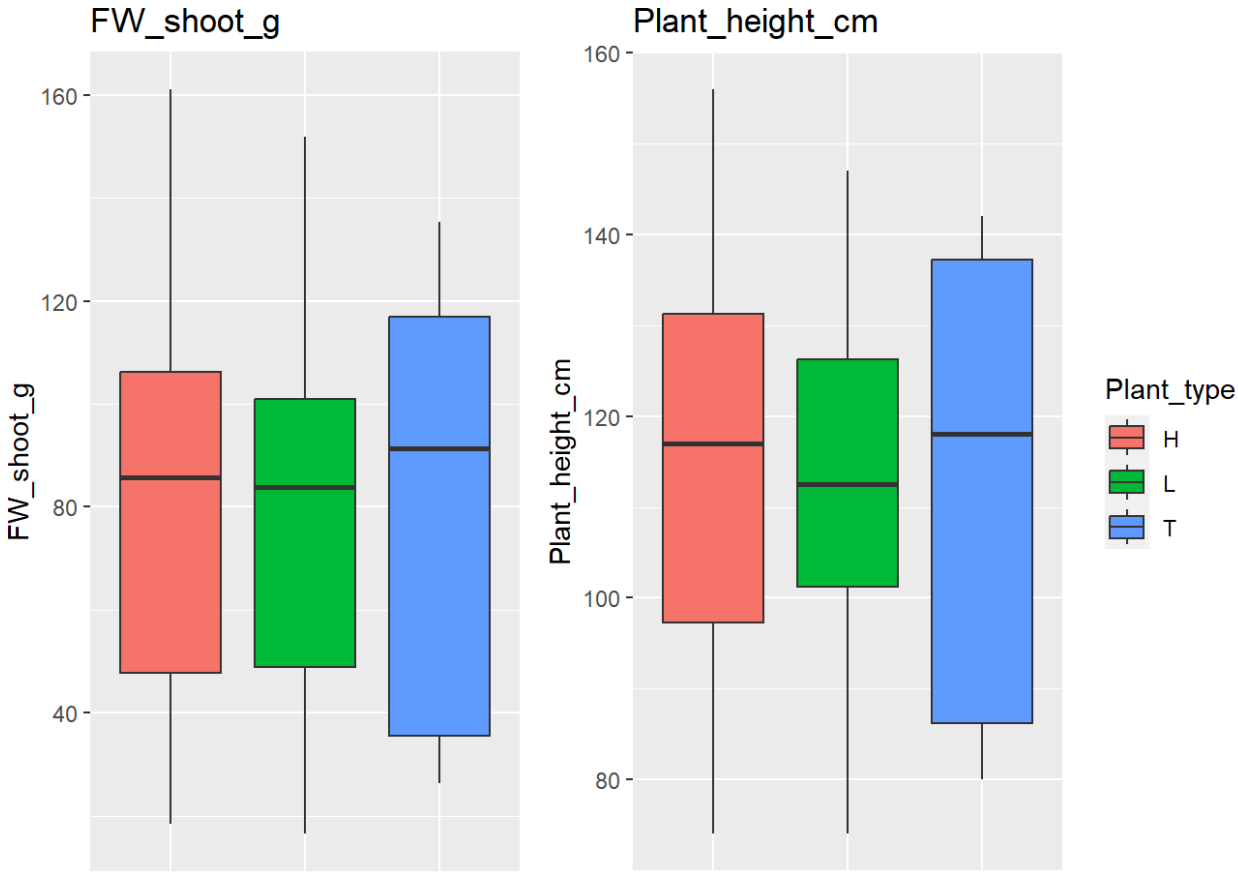
Boxplots

```
create_boxplots(endpoint, variables, "Genotype")
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



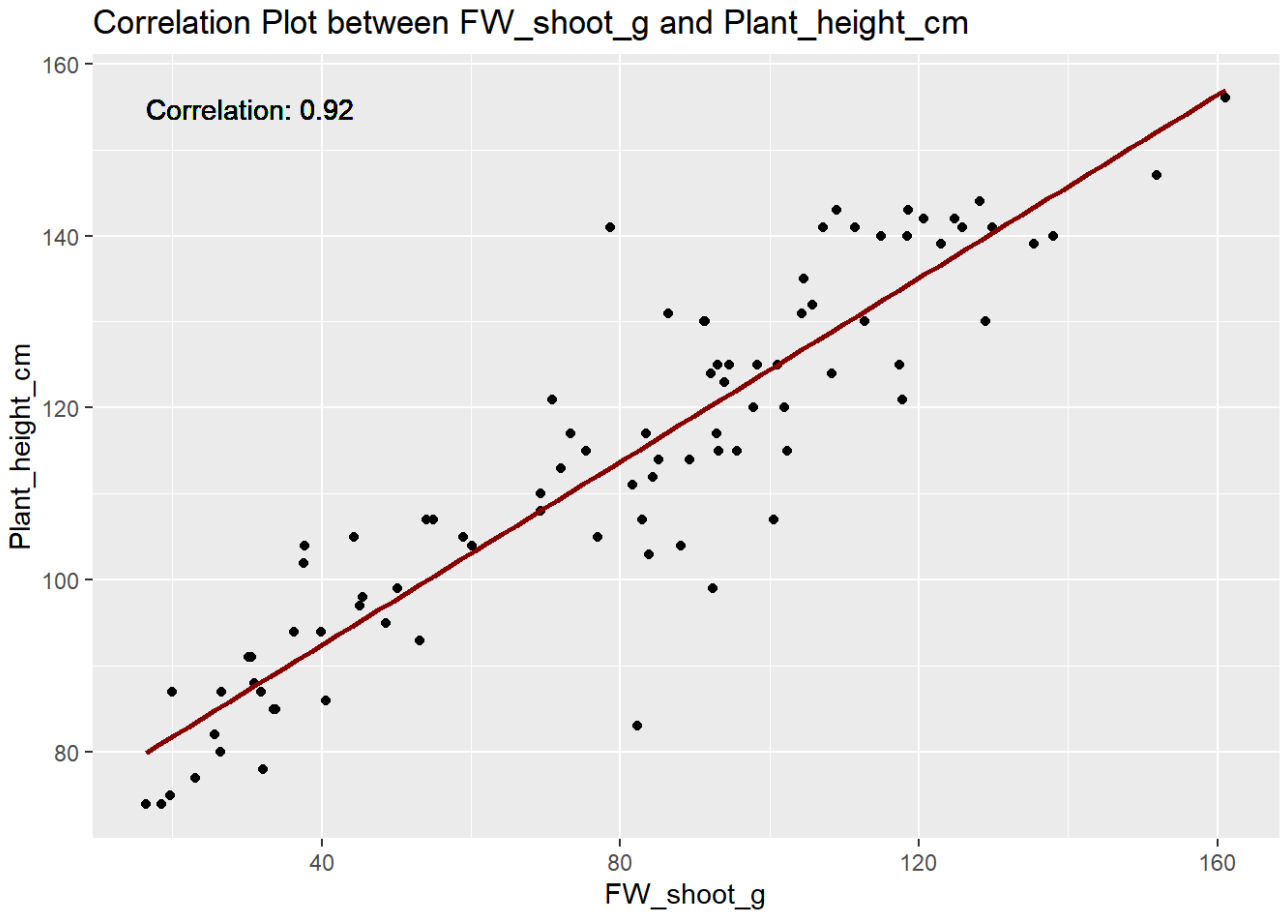

```
create_boxplots(endpoint, variables, "Plant_type")
```



Correlation plots

```
for (i in 1:(length(variables) - 1)) {
  for (j in (i + 1):length(variables)) {
    calculate_correlation_plot(endpoint, variables[i], variables[j])
  }
}
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

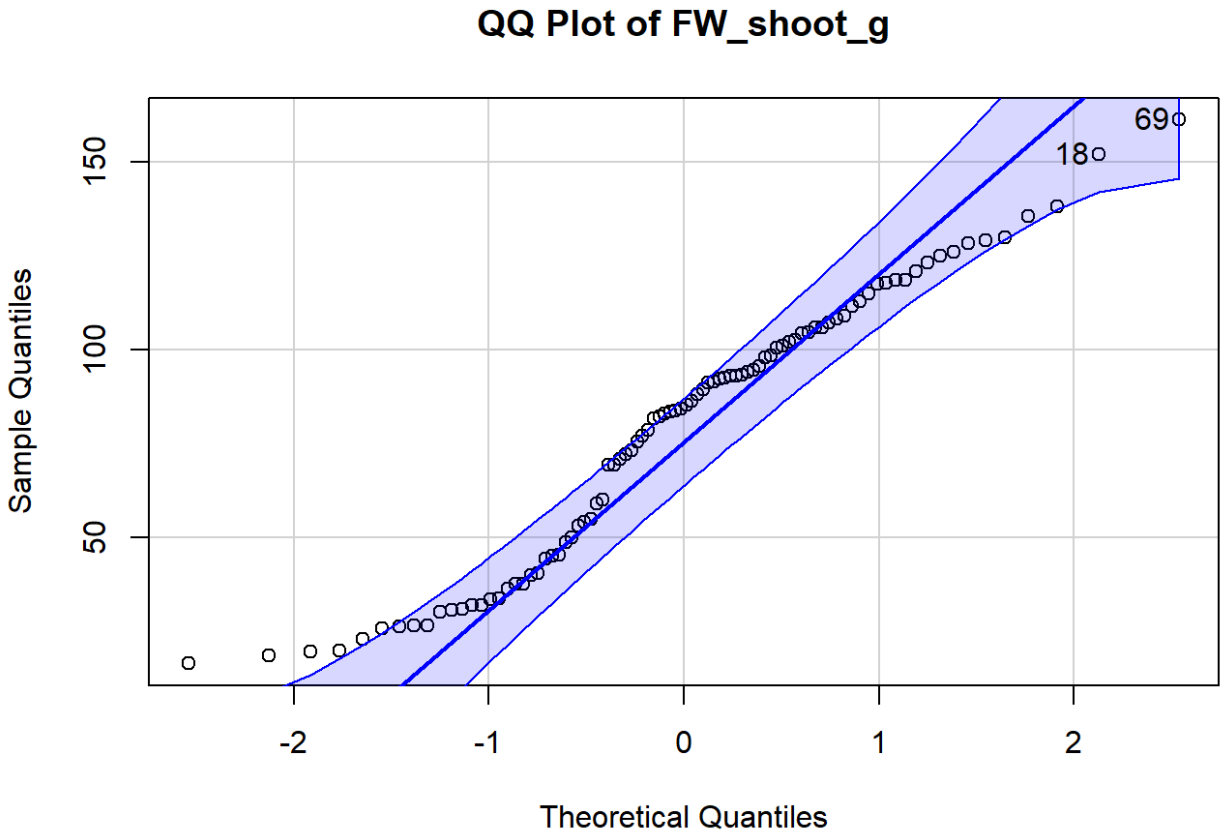


B. Normality hypothesis and outlier detection

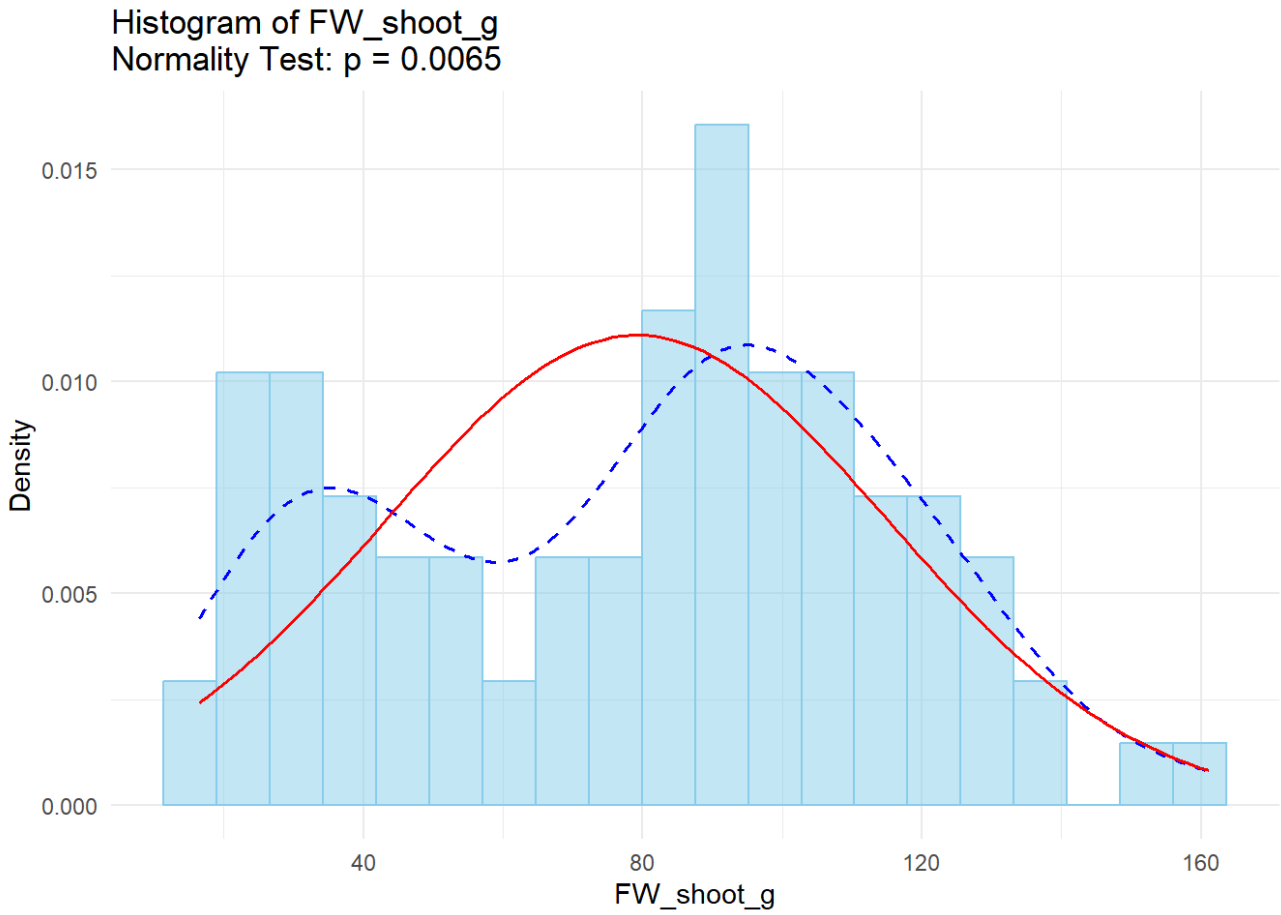
Test for normality hypothesis and plot density histogram. The red curve is the normal distribution, the blue dotted curve is the data density curve.

```
normality_results <- normality_test_histogram(endpoint)
```

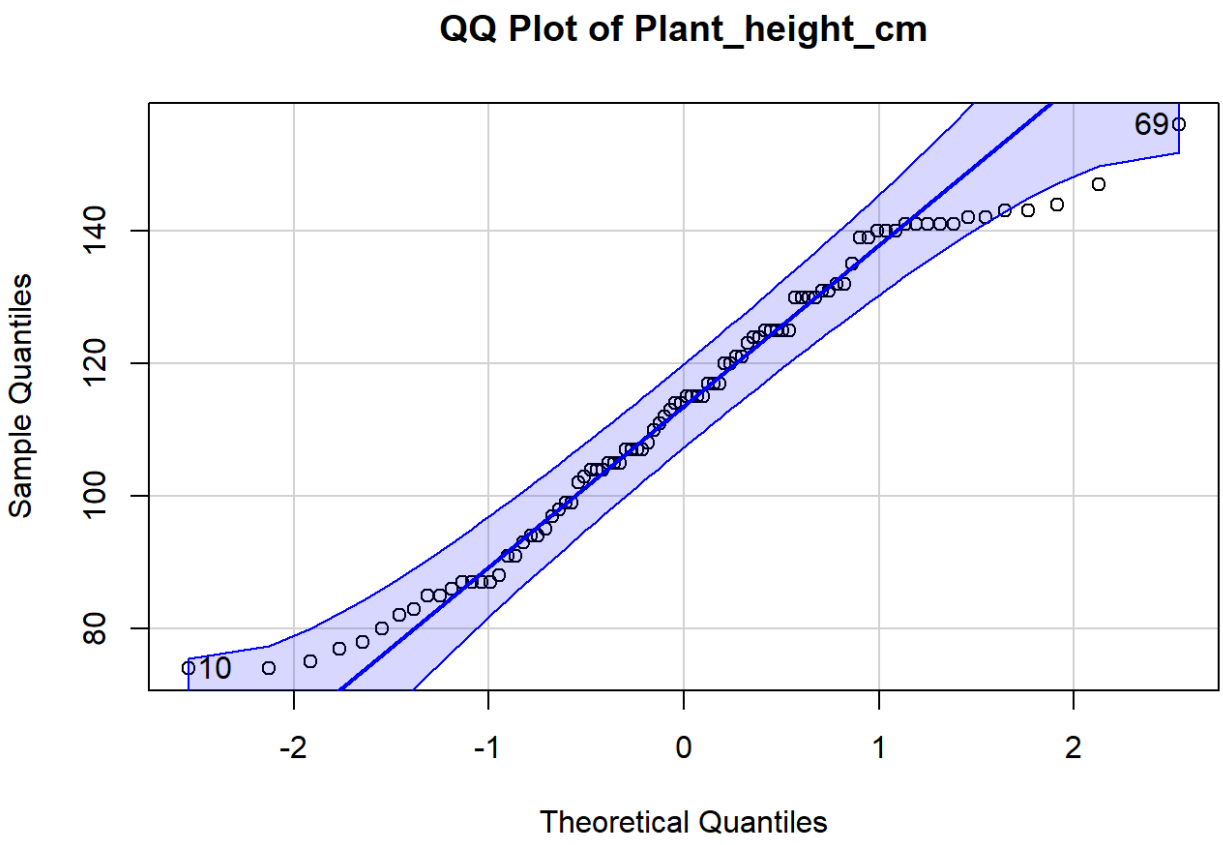
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

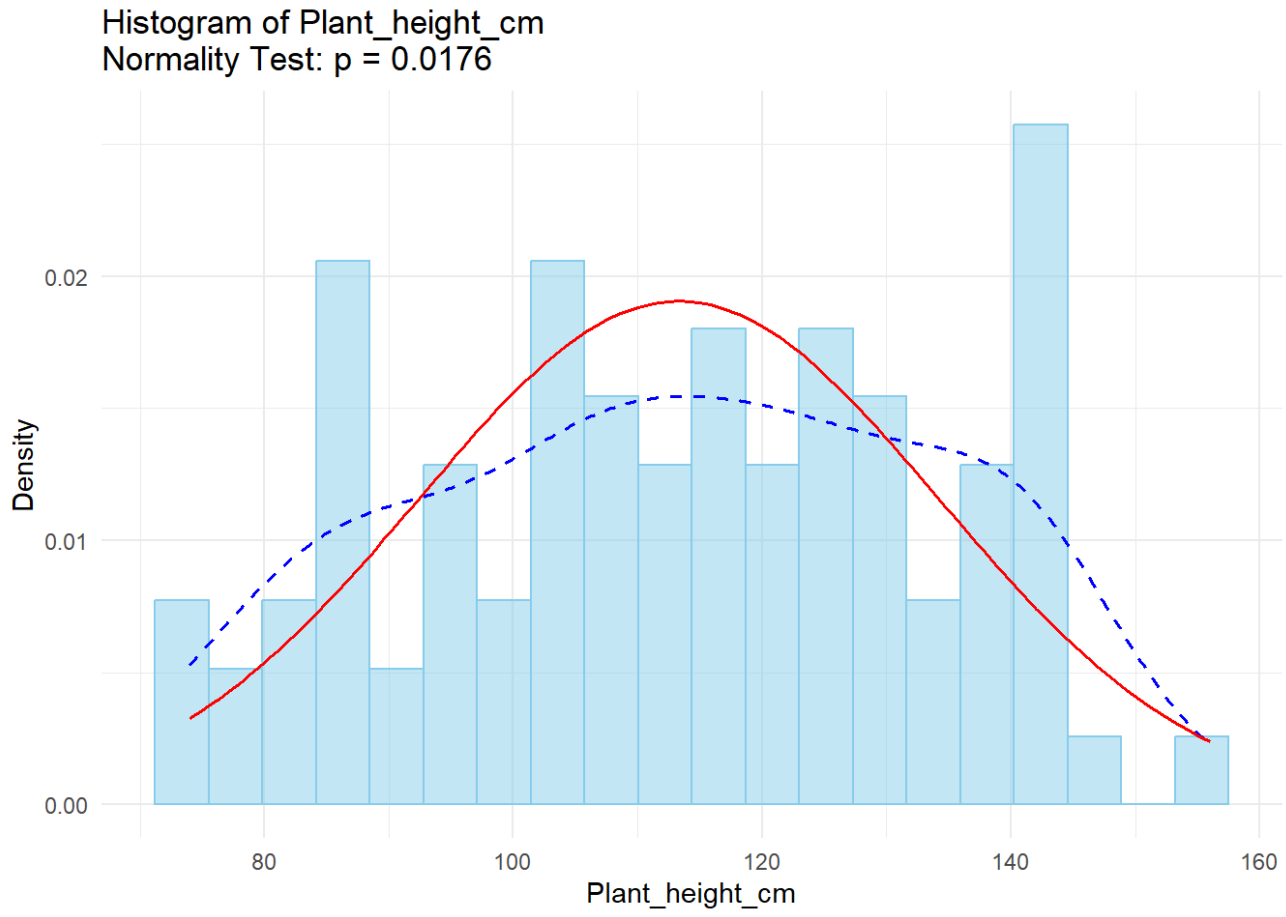



```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Histogram of FW_shoot_g
Normality Test: p = 0.0065
0.015
0.010
0.005
0.000
Density
40
80
120
160
FW_shoot_g

```
## [1] 69 18
```






```
## [1] 69 10
```

Remove the outliers, replacing them with NULL values and normality visual verification.

The function detect_replace_ouliers_by_genotype checks for outlying values, using the Tukey method.

Then run the function on all variables of the dataset.

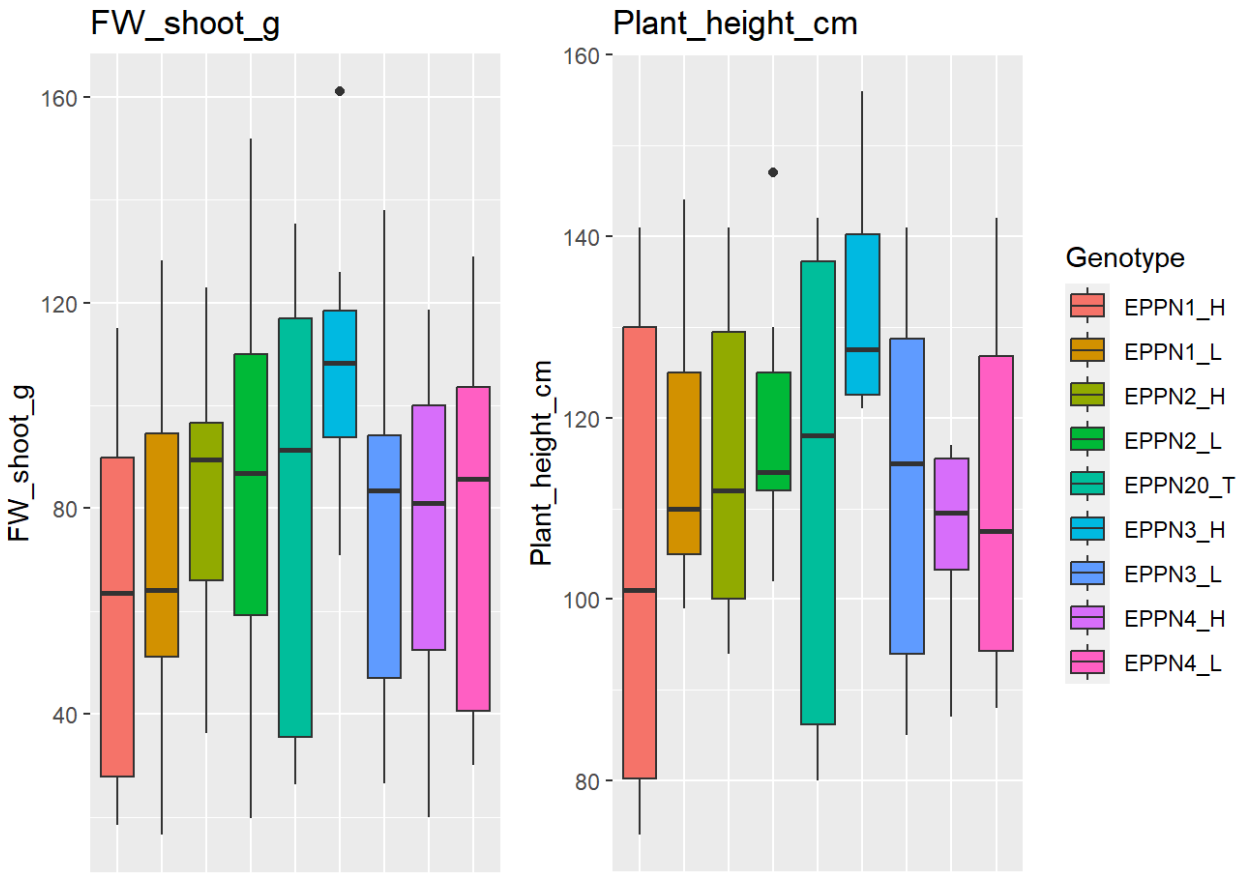
```
endpoint_clean <- endpoint
# Run the function on the dataset for all the variables
endpoint_clean <- detect_replace_outliers_by_genotype(endpoint_clean)
```

Boxplots after outlier detection

```
create_boxplots(endpoint_clean, variables, "Genotype")
```

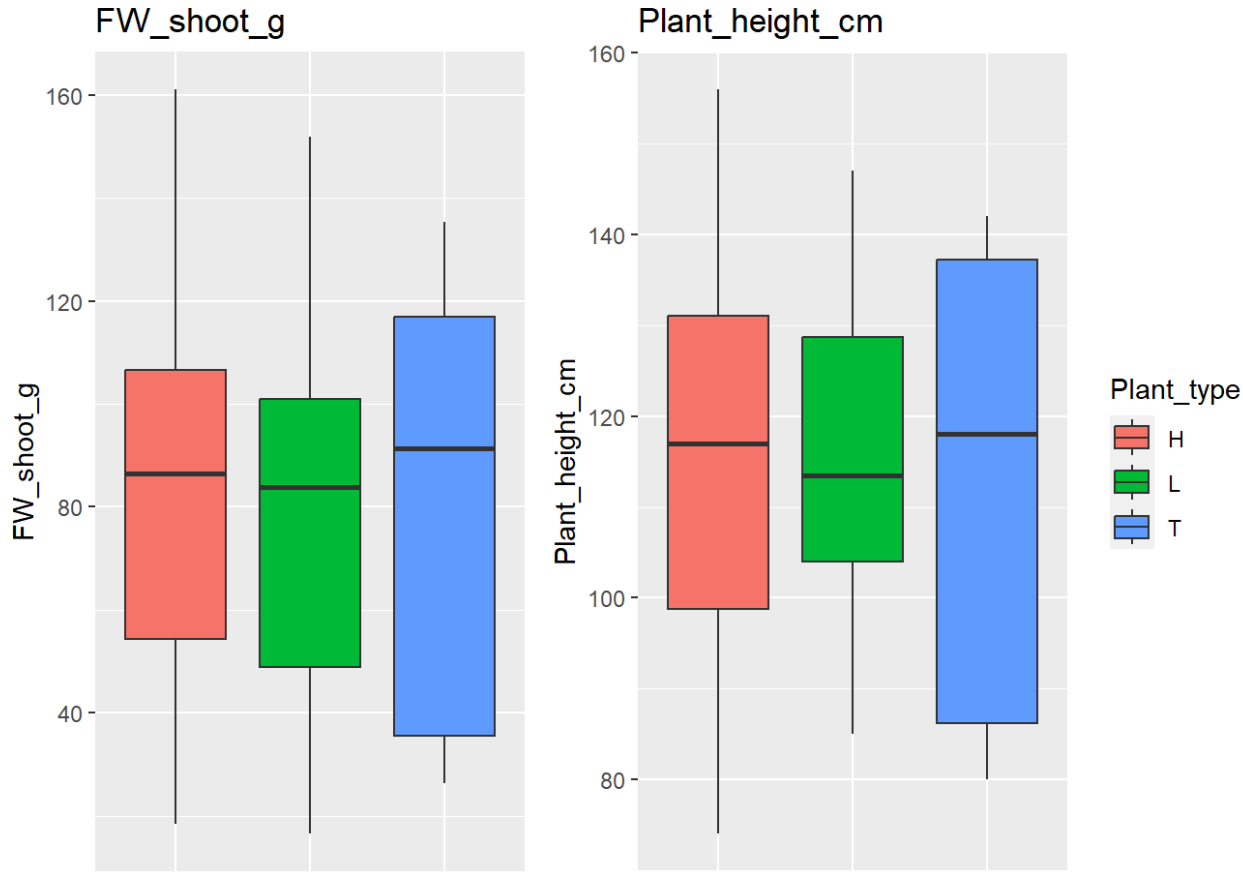
```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_boxplot()`).
```




```
create_boxplots(endpoint_clean, variables, "Plant_type")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
## Removed 6 rows containing non-finite values (`stat_boxplot()`).
```



Violin and sina plots after outlier detection

```
create_violin_plots(endpoint_clean, variables, "Genotype")
```

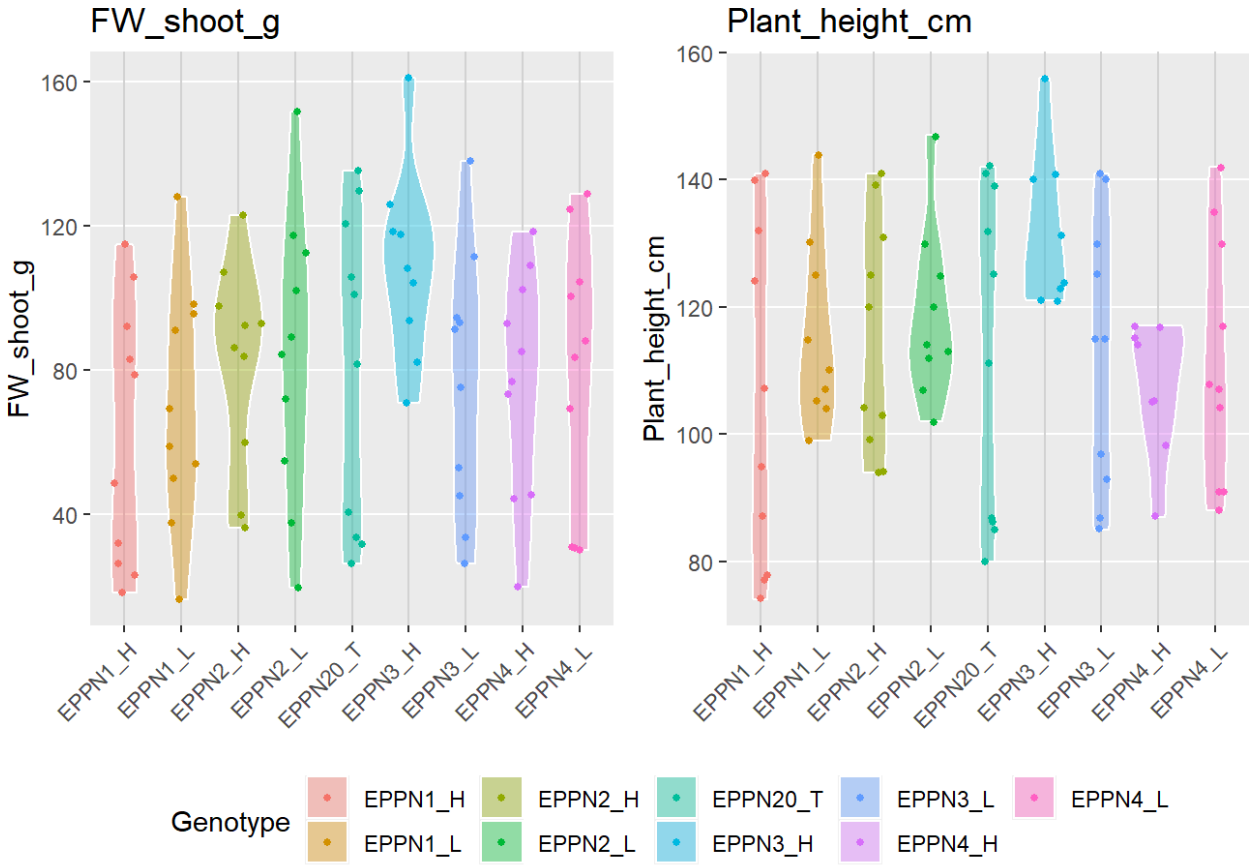
```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## ℹ Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_sina()`).
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_sina()`).
```



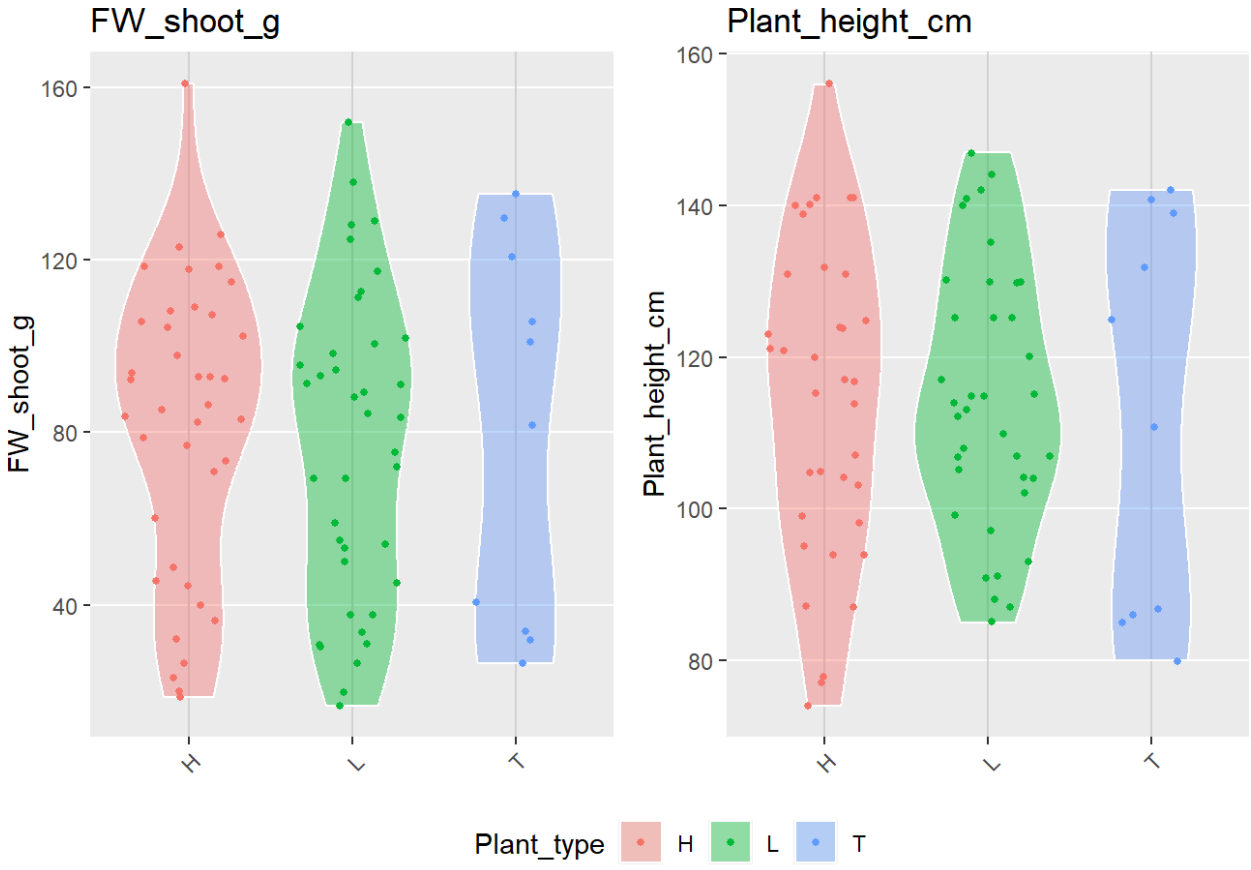

```
create_violin_plots(endpoint_clean, variables, "Plant_type")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_sina()`).
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_sina()`).
```



Exploration statistics for the variables after outlier detection

```
skim(endpoint_clean[variables])
```

Data summary

Name	endpoint_clean[variables]
Number of rows	90
Number of columns	2
_____	
Column type frequency:	
numeric	2
_____	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
FW_shoot_g	1	0.99	79.74	35.7	16.5	45.5	85.2	105.8	161.1	▅▂▇▇▁
Plant_height_cm	6	0.93	114.26	19.7	74.0	99.0	115.0	130.0	156.0	▃▇▇▇▆

```
for (var in variables) {
  cat("\nSummary for:", var, "\n")
  endpoint_clean %>%
    group_by(Genotype) %>%
    summarize(mean    = mean(get(var), na.rm = TRUE),
              std.dev = sd(get(var), na.rm = TRUE),
              n_missing  = sum(is.na(get(var)))) %>%
    arrange(desc(mean)) %>%
    print(n = Inf)
}
```

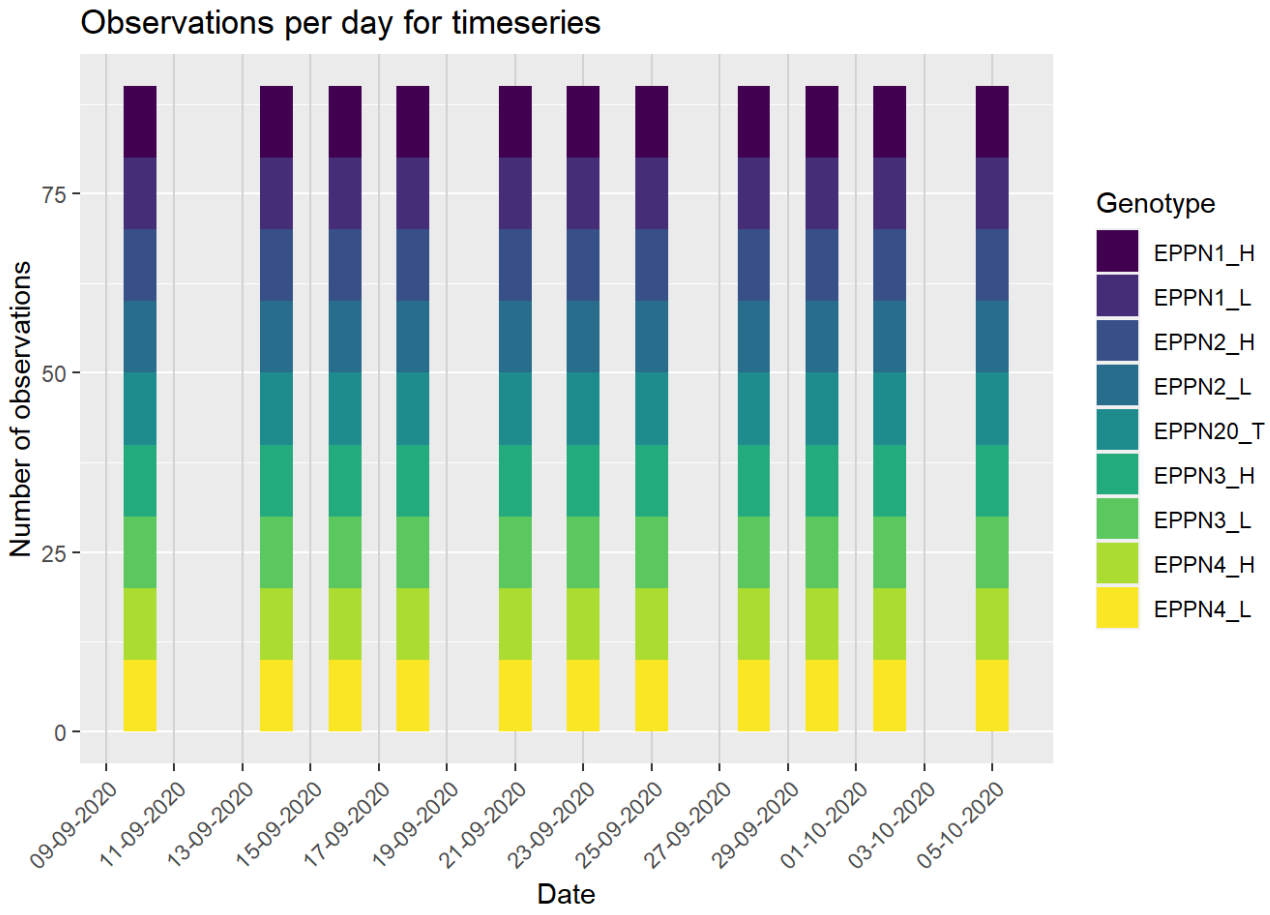
```
##
## Summary for: FW_shoot_g
## # A tibble: 9 × 4
##   Genotype  mean std.dev n_missing
##   <fct>    <dbl>   <dbl>     <int>
## 1 EPPN3_H  109.     26.4         1
## 2 EPPN2_L   84.2    39.7         0
## 3 EPPN2_H   82.0    28.2         0
## 4 EPPN20_T  80.7    43.7         0
## 5 EPPN4_L   79.2    37.9         0
## 6 EPPN4_H   76.8    31.7         0
## 7 EPPN3_L   76.2    36.1         0
## 8 EPPN1_L   70.0    33.3         0
## 9 EPPN1_H   62.4    36.7         0
##
## Summary for: Plant_height_cm
## # A tibble: 9 × 4
##   Genotype  mean std.dev n_missing
##   <fct>    <dbl>   <dbl>     <int>
## 1 EPPN3_H   132.    12.6         2
## 2 EPPN2_L   119.    13.6         1
## 3 EPPN1_L   115.    14.7         1
## 4 EPPN2_H   115     18.4         0
## 5 EPPN20_T  113.    26.0         0
## 6 EPPN3_L   113.    21.3         0
## 7 EPPN4_L   111.    19.3         0
## 8 EPPN4_H   107.    10.7         2
## 9 EPPN1_H   106.    26.9         0
```

2. Exploration of the timeseries data

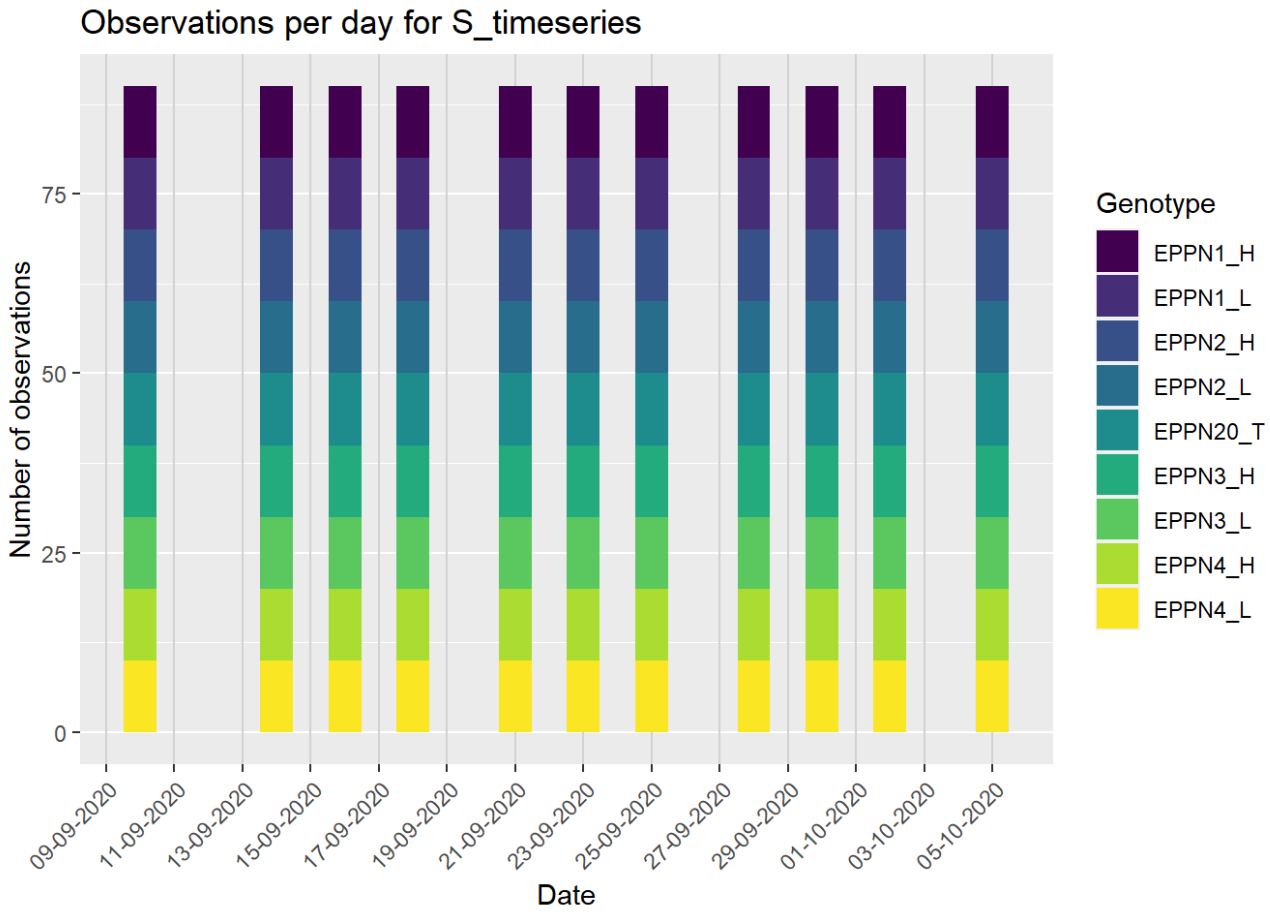
In this part, we look at the timeseries, S_timeseries and T_timeseries datasets, also using several functions, located in the functions.R script.

Number of data observations per day for the traits of the timeseries datasets

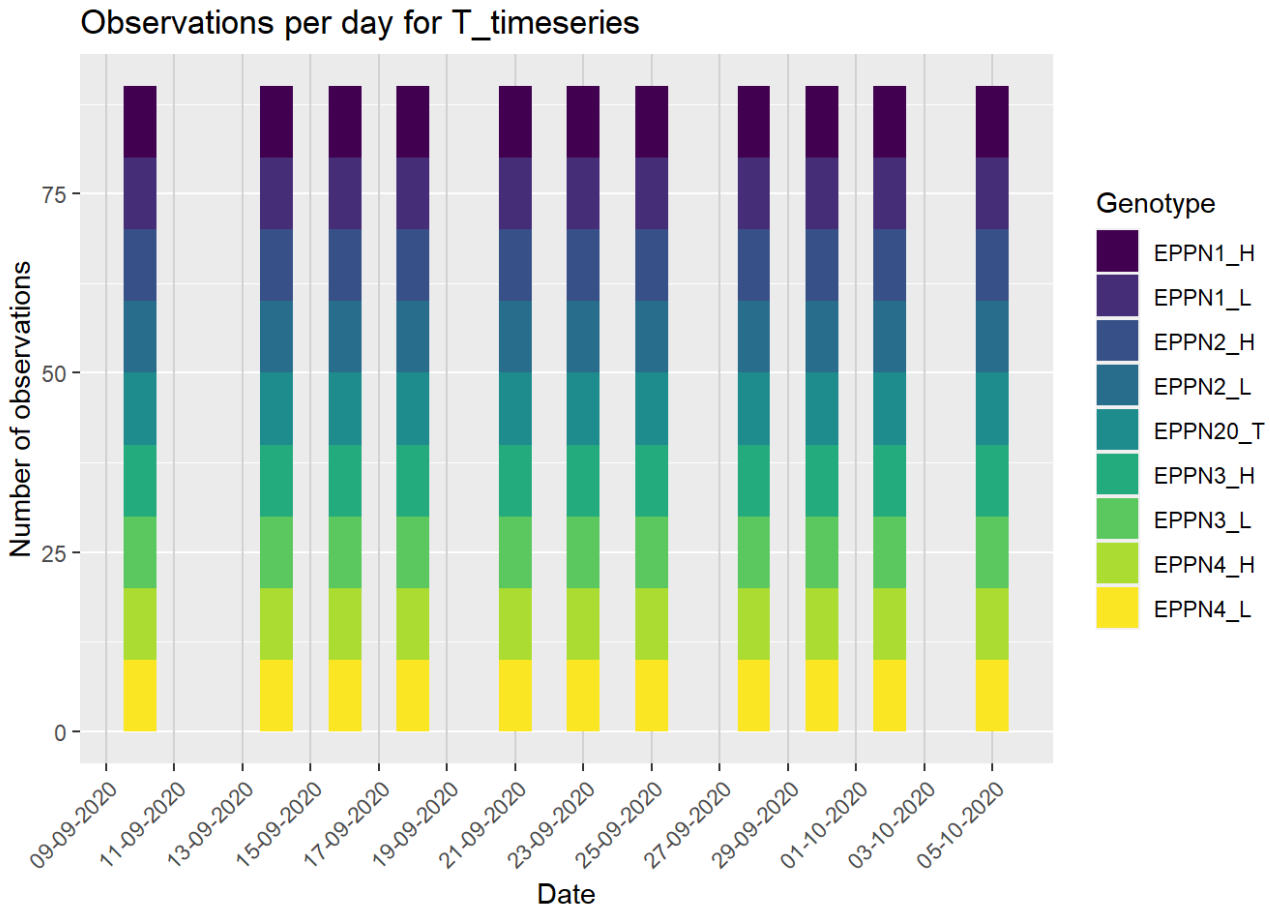
```
h1 <- ggplot(timeseries, aes(x = Date)) +
  geom_bar(aes(fill = Genotype), position = "stack", width = 0.96) +
  scale_fill_viridis_d(option = "D") +
  labs(x = "Date", y = "Number of observations", title = "Observations per day for timese
ries") +
  scale_y_continuous(breaks = seq(from = 0, to = 325, by = 25)) +
  scale_x_date(date_breaks = "2 days", date_labels = "%d-%m-%Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major.x = element_line(color = "lightgray", size = 0.5),
        panel.grid.minor.x = element_blank())

h2 <- ggplot(S_timeseries, aes(x = Date)) +
  geom_bar(aes(fill = Genotype), position = "stack", width = 0.96) +
  scale_fill_viridis_d(option = "D") +
  labs(x = "Date", y = "Number of observations", title = "Observations per day for S_time
series") +
  scale_y_continuous(breaks = seq(from = 0, to = 325, by = 25)) +
  scale_x_date(date_breaks = "2 days", date_labels = "%d-%m-%Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major.x = element_line(color = "lightgray", size = 0.5),
        panel.grid.minor.x = element_blank())

h3 <- ggplot(T_timeseries, aes(x = Date)) +
  geom_bar(aes(fill = Genotype), position = "stack", width = 0.96) +
  scale_fill_viridis_d(option = "D") +
  labs(x = "Date", y = "Number of observations", title = "Observations per day for T_time
series") +
  scale_y_continuous(breaks = seq(from = 0, to = 325, by = 25)) +
  scale_x_date(date_breaks = "2 days", date_labels = "%d-%m-%Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major.x = element_line(color = "lightgray", size = 0.5),
        panel.grid.minor.x = element_blank())
h1
```


Observations per day for timeseries
Number of observations
0
25
50
75
09-09-2020
11-09-2020
13-09-2020
15-09-2020
17-09-2020
19-09-2020
21-09-2020
23-09-2020
25-09-2020
27-09-2020
29-09-2020
01-10-2020
03-10-2020
05-10-2020
Date
Genotype
EPPN1_H
EPPN1_L
EPPN2_H
EPPN2_L
EPPN20_T
EPPN3_H
EPPN3_L
EPPN4_H
EPPN4_L

h2


Observations per day for S_timeseries
Number of observations
0
25
50
75
09-09-2020
11-09-2020
13-09-2020
15-09-2020
17-09-2020
19-09-2020
21-09-2020
23-09-2020
25-09-2020
27-09-2020
29-09-2020
01-10-2020
03-10-2020
05-10-2020
Date
Genotype
EPPN1_H
EPPN1_L
EPPN2_H
EPPN2_L
EPPN20_T
EPPN3_H
EPPN3_L
EPPN4_H
EPPN4_L

h3



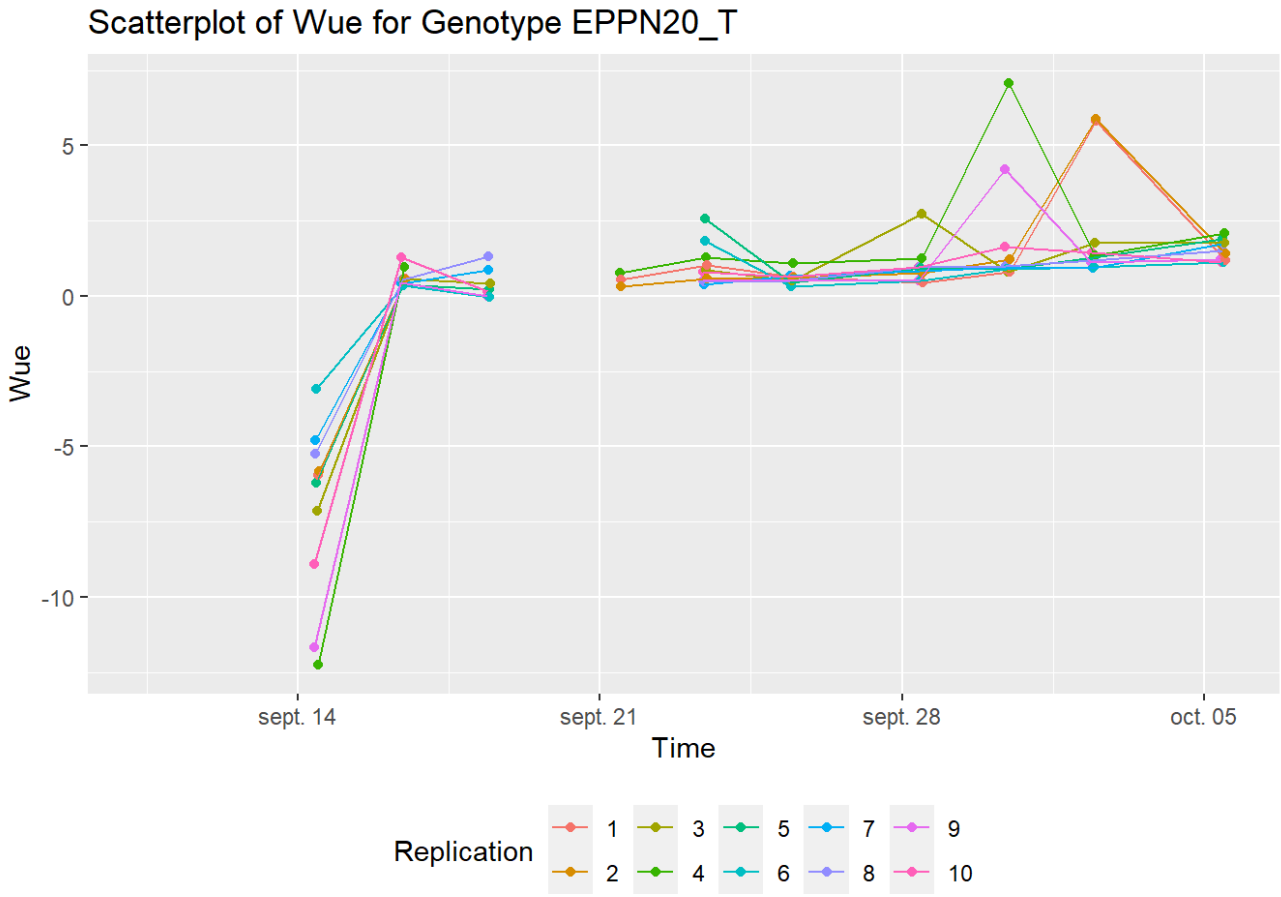
A. Exploration of the timeseries dataframe

Scatter plots by Genotype

```
plot_scatter_by_genotype(timeseries, variables_t, "EPPN20_T")
```

```
## Warning: Removed 20 rows containing missing values (`geom_point()`).
```

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

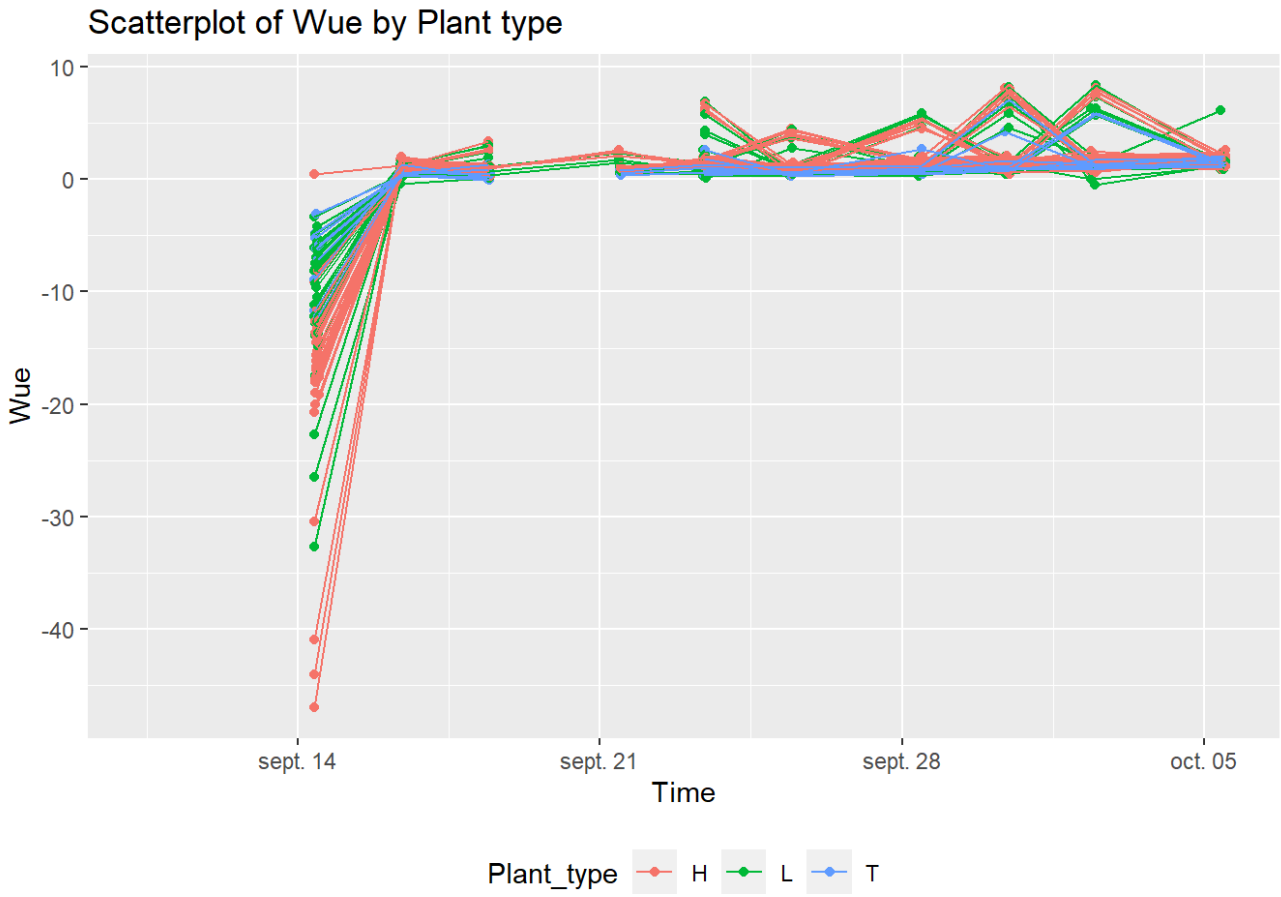


Scatterplots for all genotypes by Plant type (Hybride, Line, EPPN20_T) with smooth line.

```
plot_scatter_with_smooth(timeseries, variables_t)
```

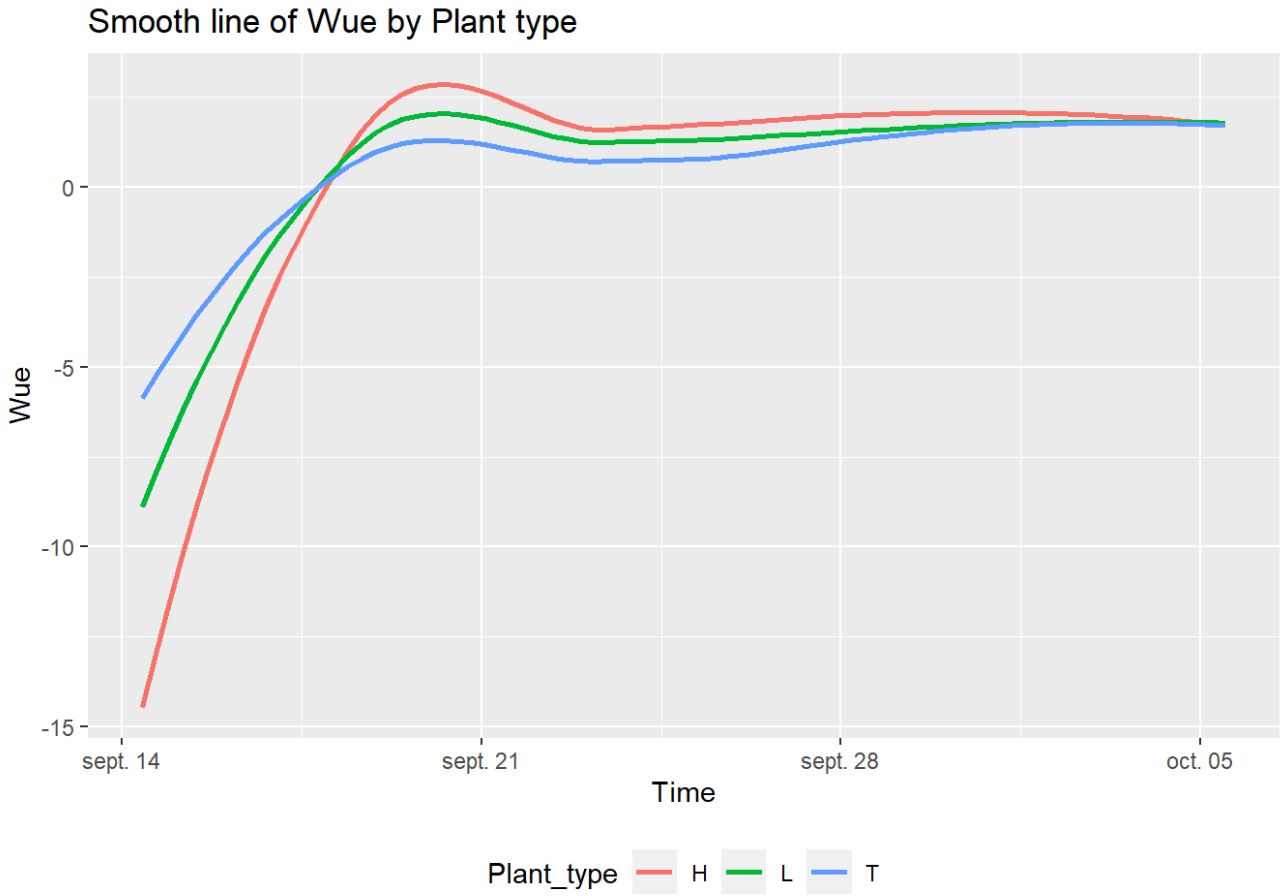
```
## Warning: Removed 173 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 90 rows containing missing values (`geom_line()`).
```




```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 173 rows containing non-finite values (`stat_smooth()`).
```



Scatter plots for all genotypes by water treatment

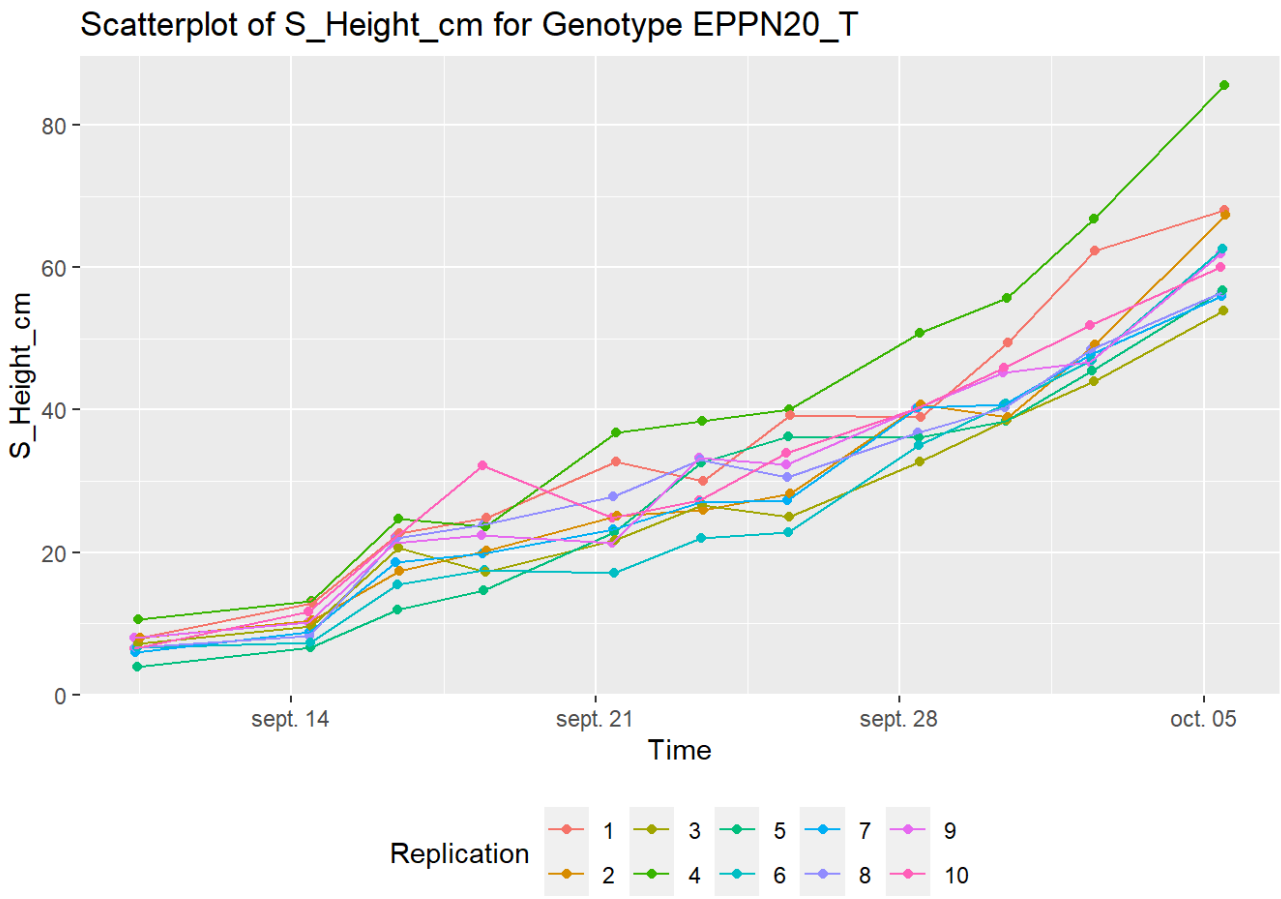
```
print(paste0("No data for", platform))
```

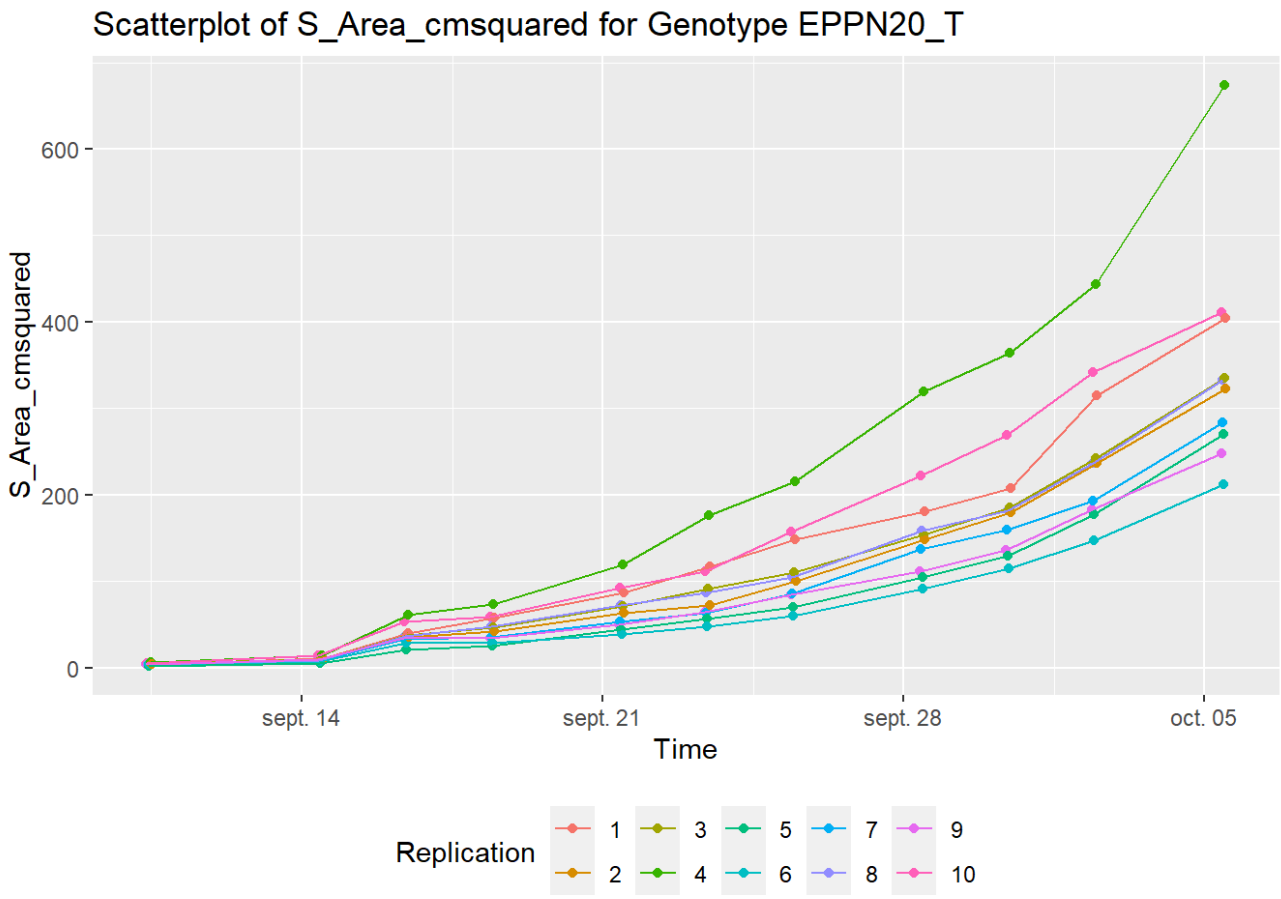
```
## [1] "No data forALSIA"
```

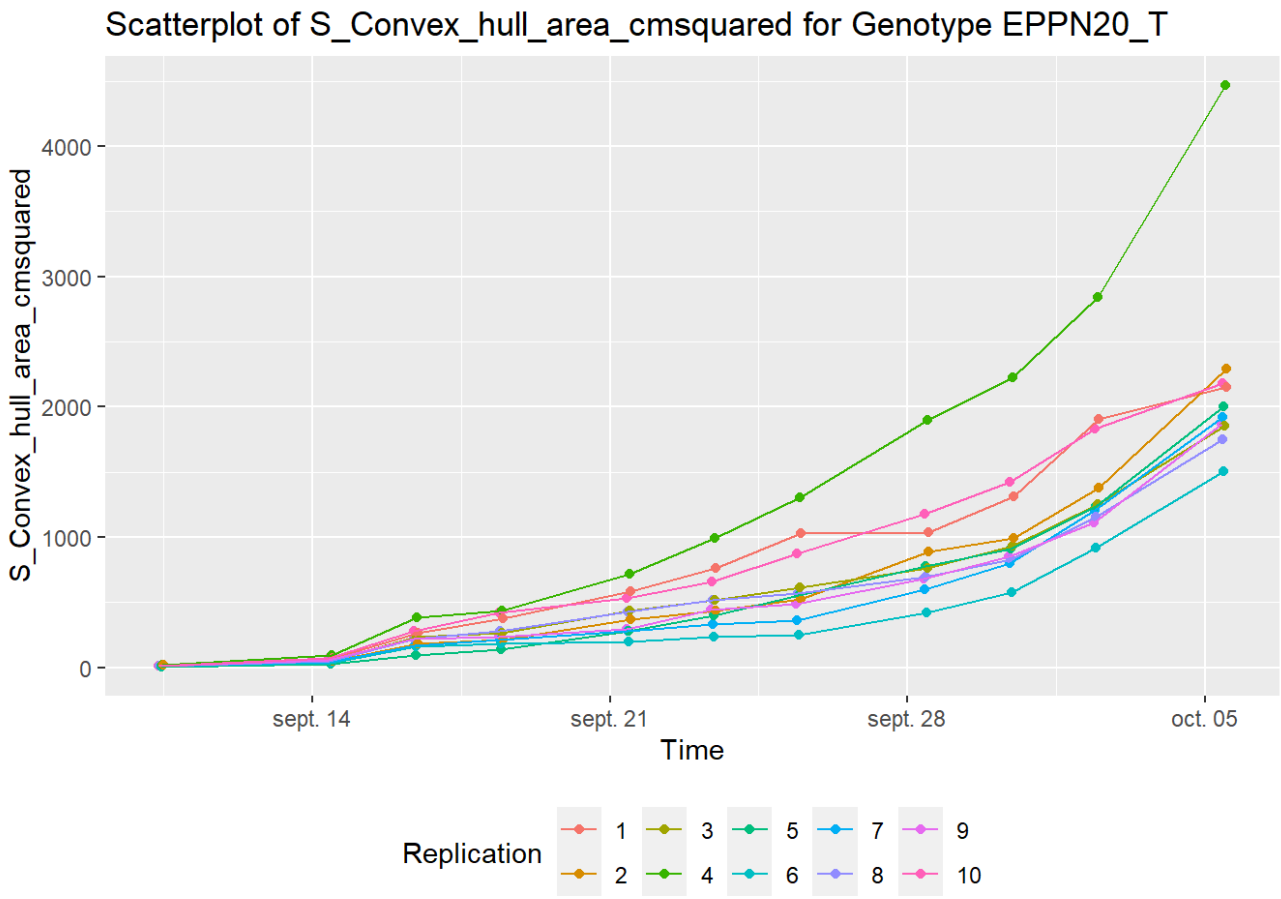
B. Exploration of the S_timeseries dataframe

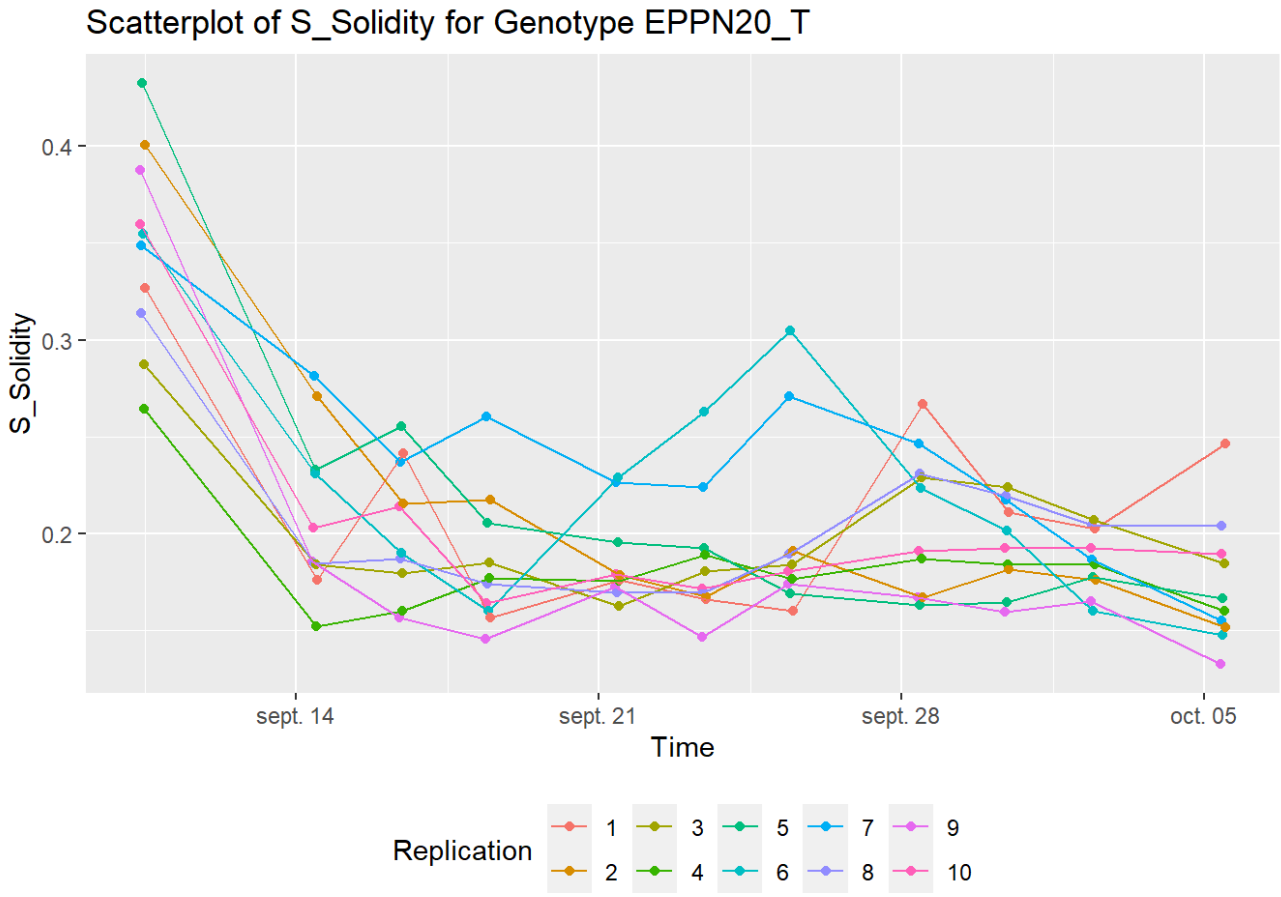
Scatter plots by Genotype

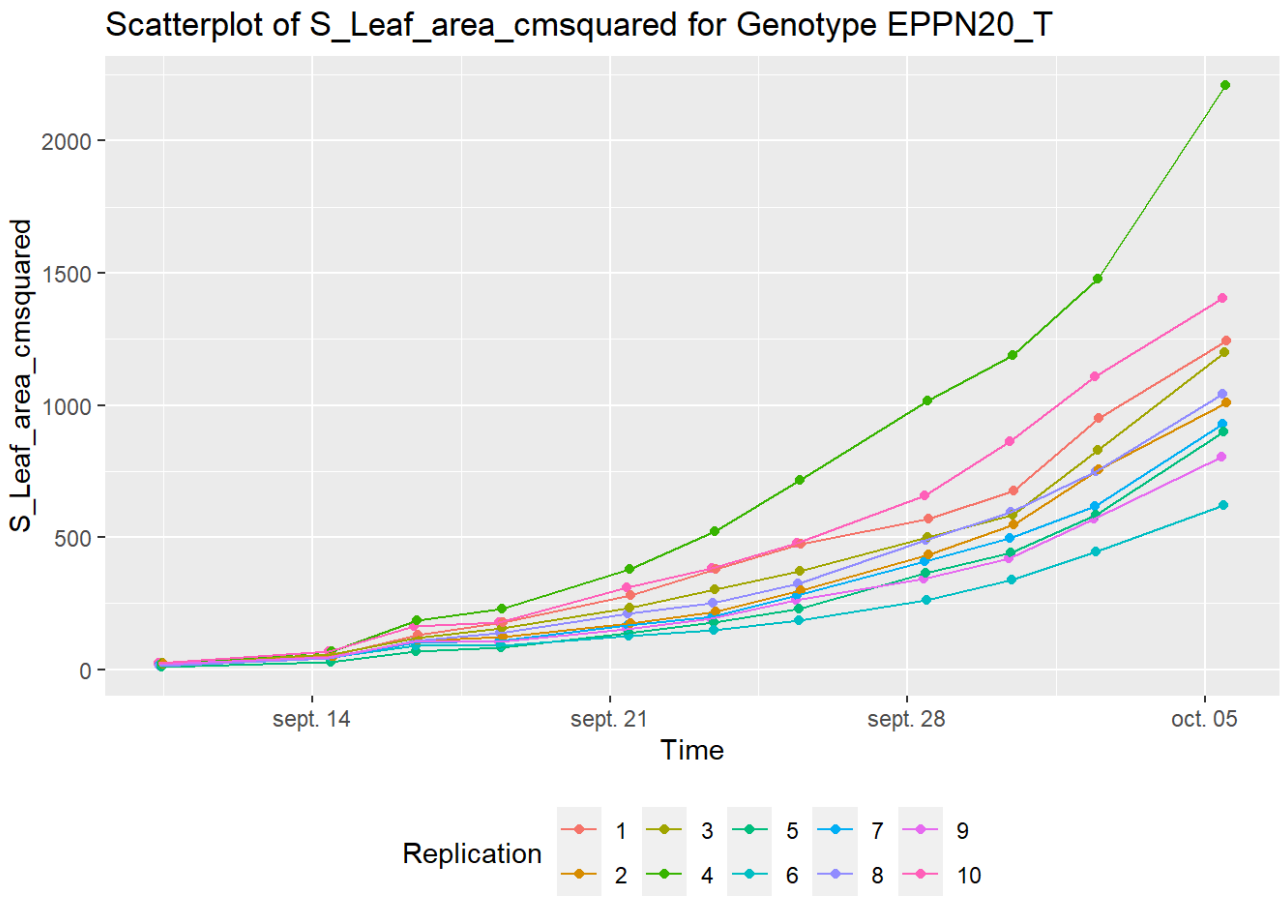
```
plot_scatter_by_genotype(S_timeseries, variables_S, "EPPN20_T")
```


Scatterplot of S_Height_cm for Genotype EPPN20_T
S_Height_cm
0
20
40
60
80
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10


Scatterplot of S_Area_cmsquared for Genotype EPPN20_T
S_Area_cmsquared
0
200
400
600
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10

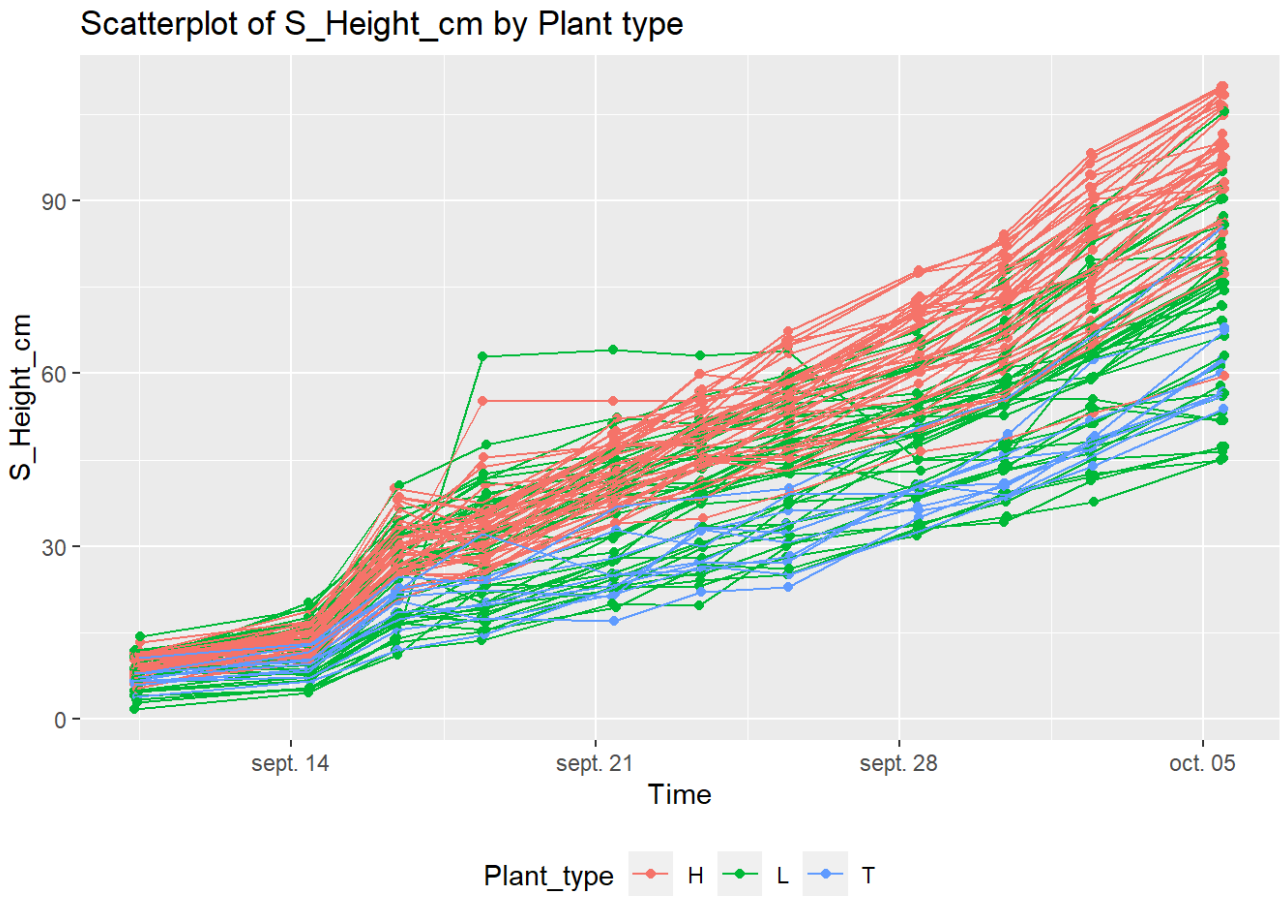

Scatterplot of S_Convex_hull_area_cmsquared for Genotype EPPN20_T
S_Convex_hull_area_cmsquared
4000
3000
2000
1000
0
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10


Scatterplot of S_Solidity for Genotype EPPN20_T
S_Solidity
0.4
0.3
0.2
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10

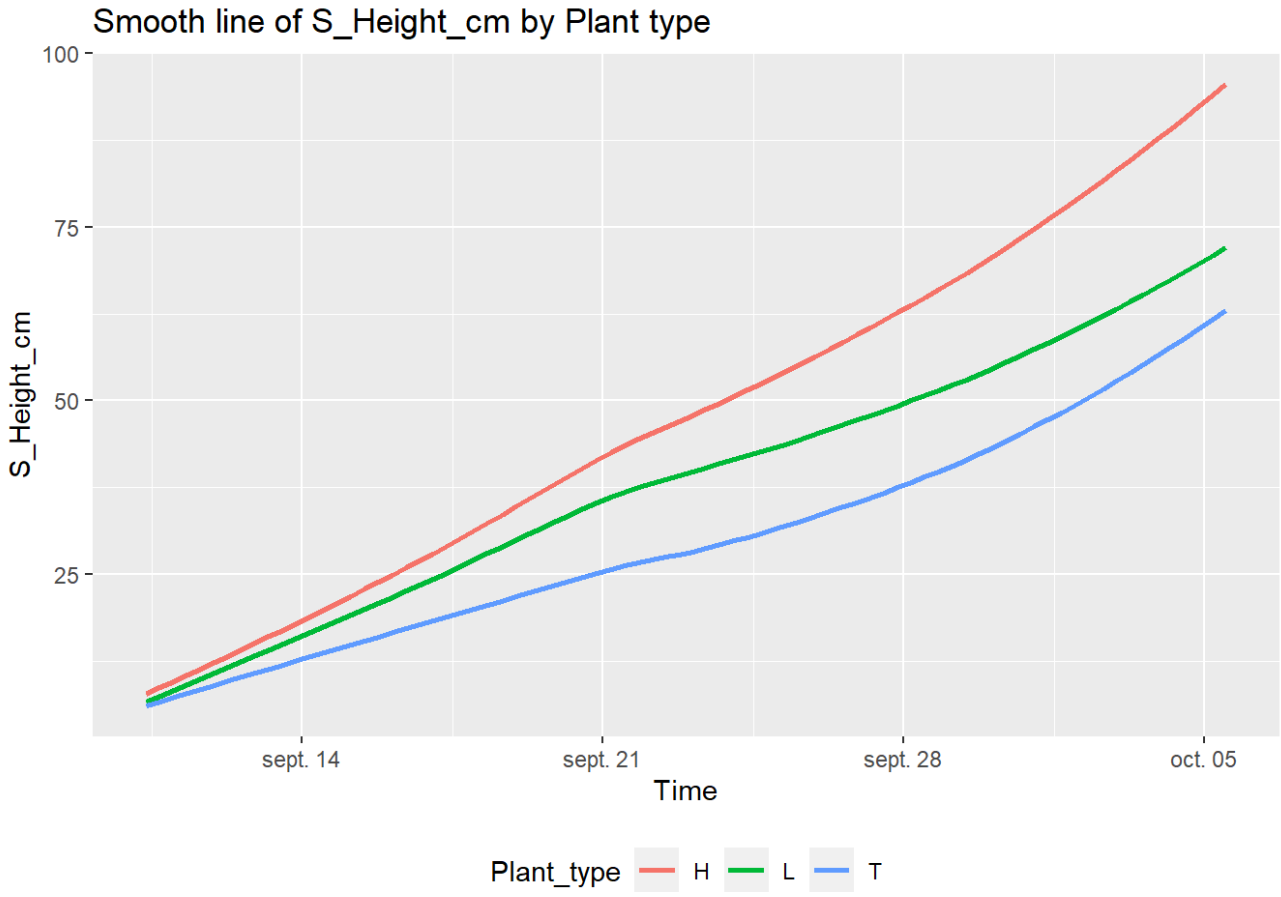


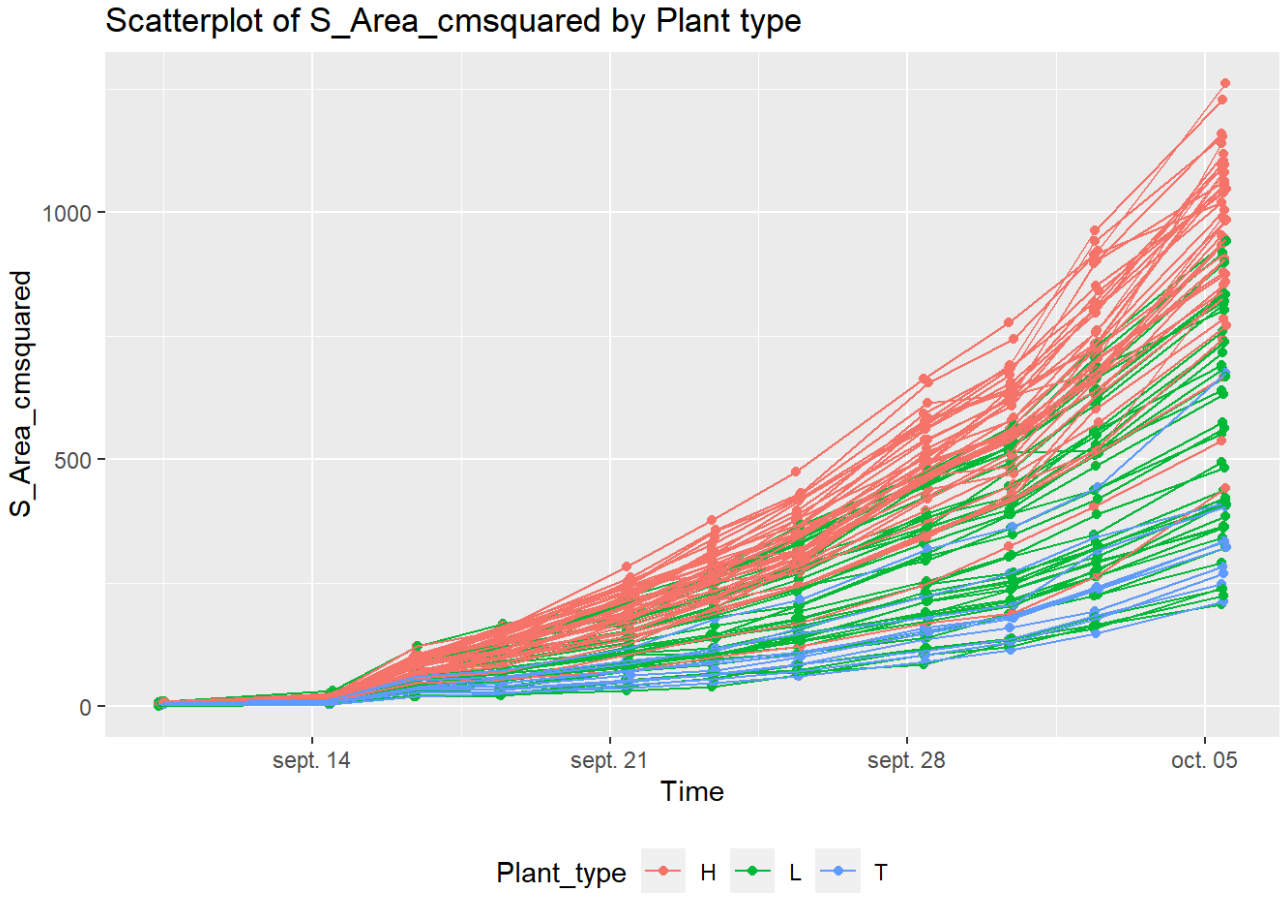
Scatterplots for all genotypes by Plant type (Hybride, Line, EPPN20_T) with smooth line.

```
plot_scatter_with_smooth(S_timeseries, variables_S)
```

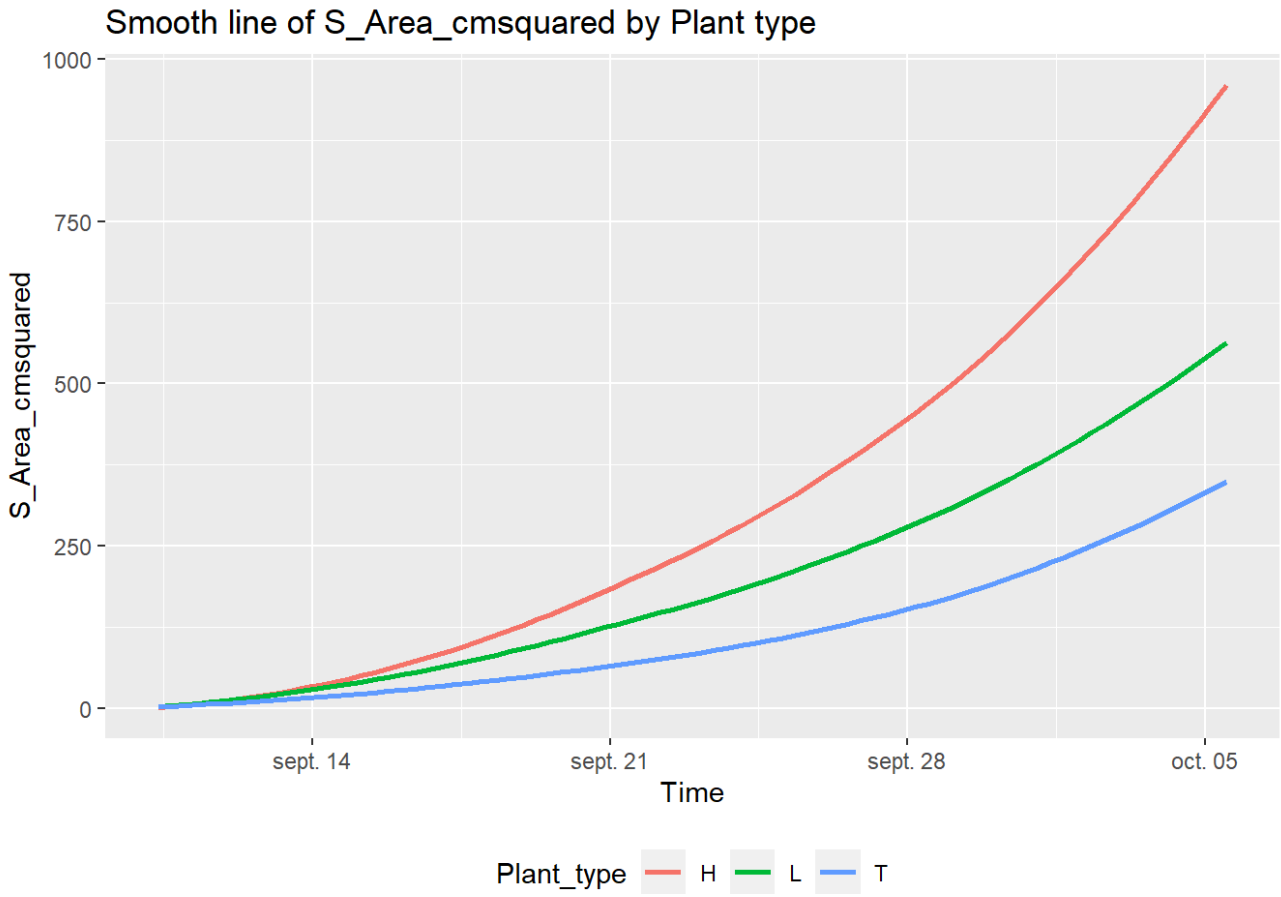


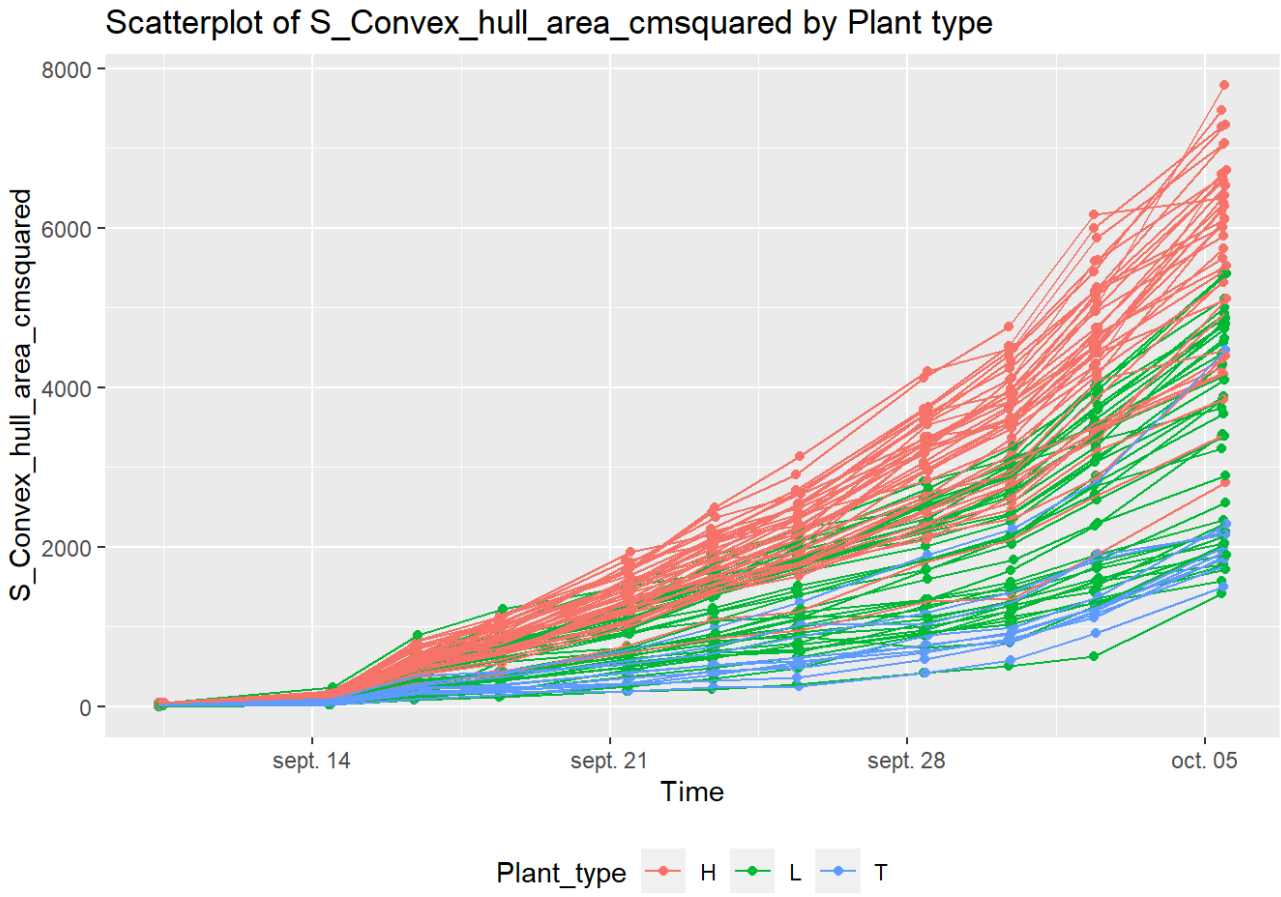
```
## `geom_smooth()` using formula = 'y ~ x'
```



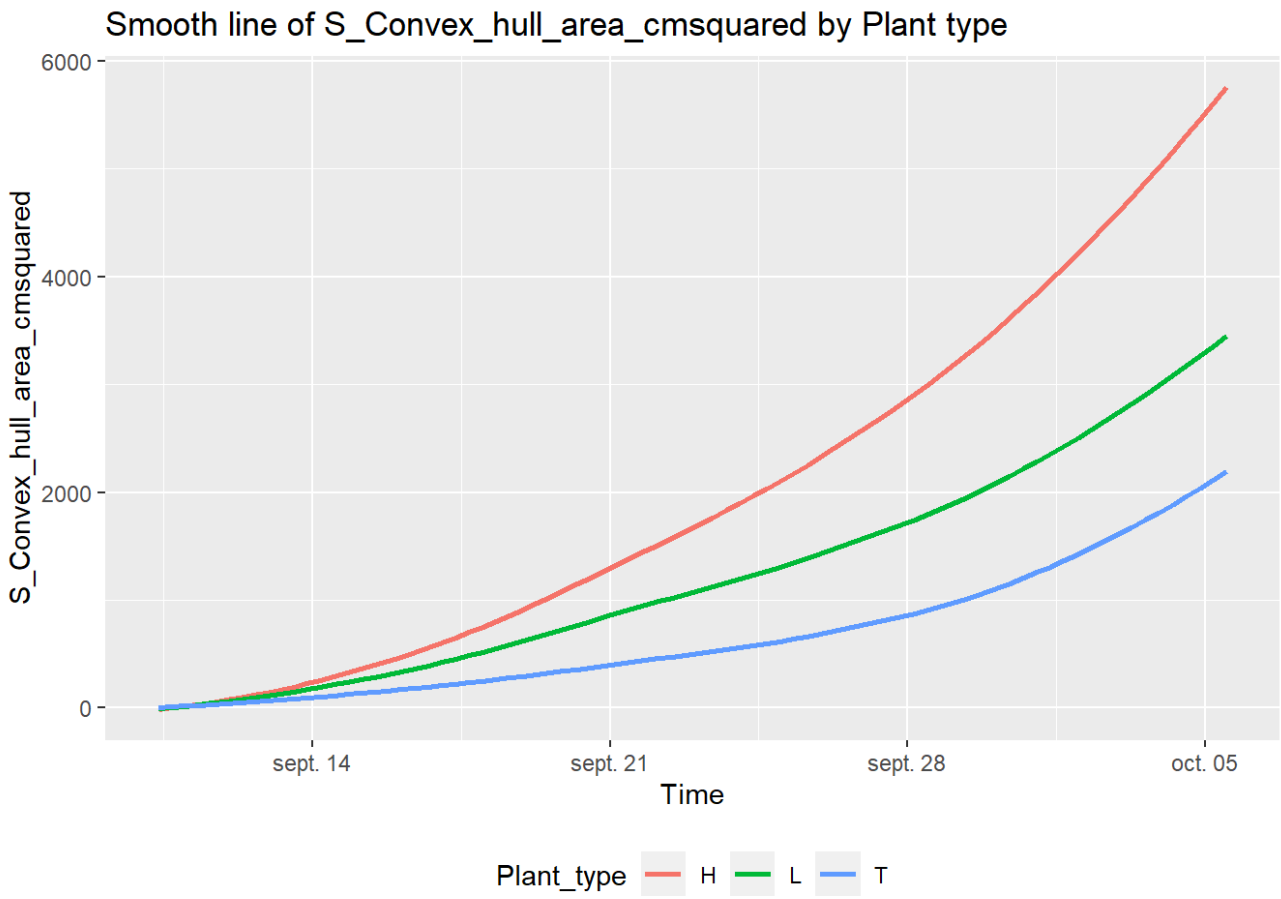


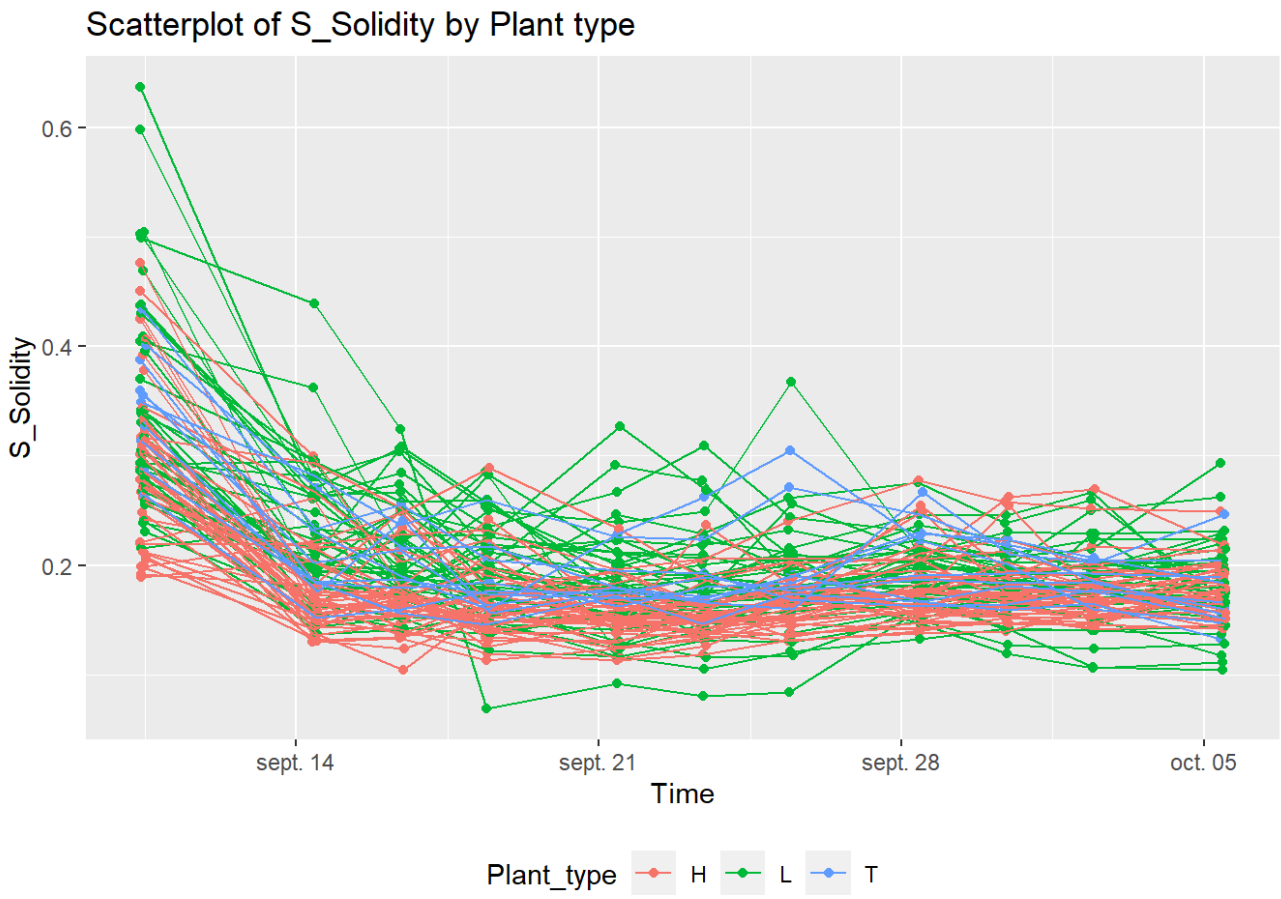

```
## `geom_smooth()` using formula = 'y ~ x'
```



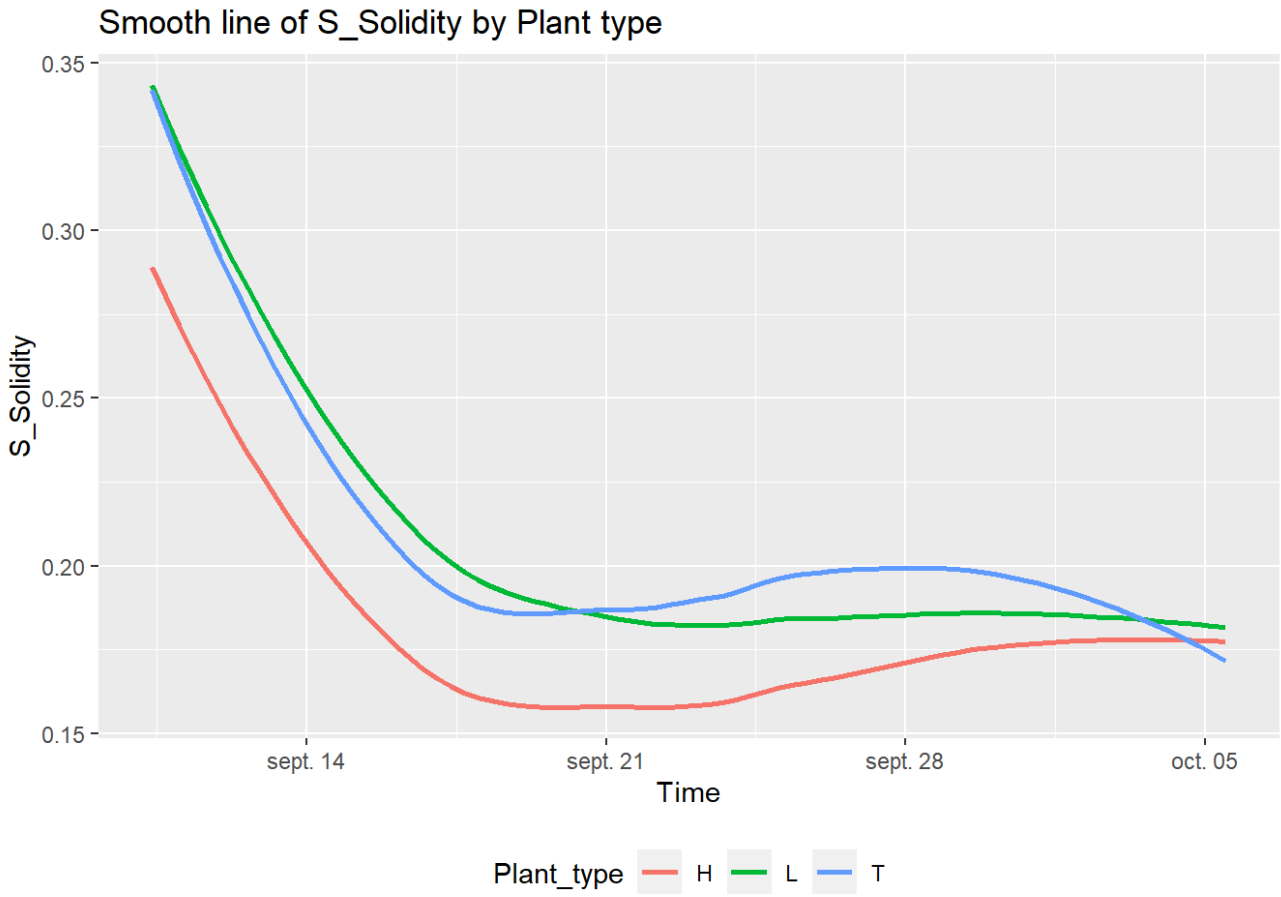


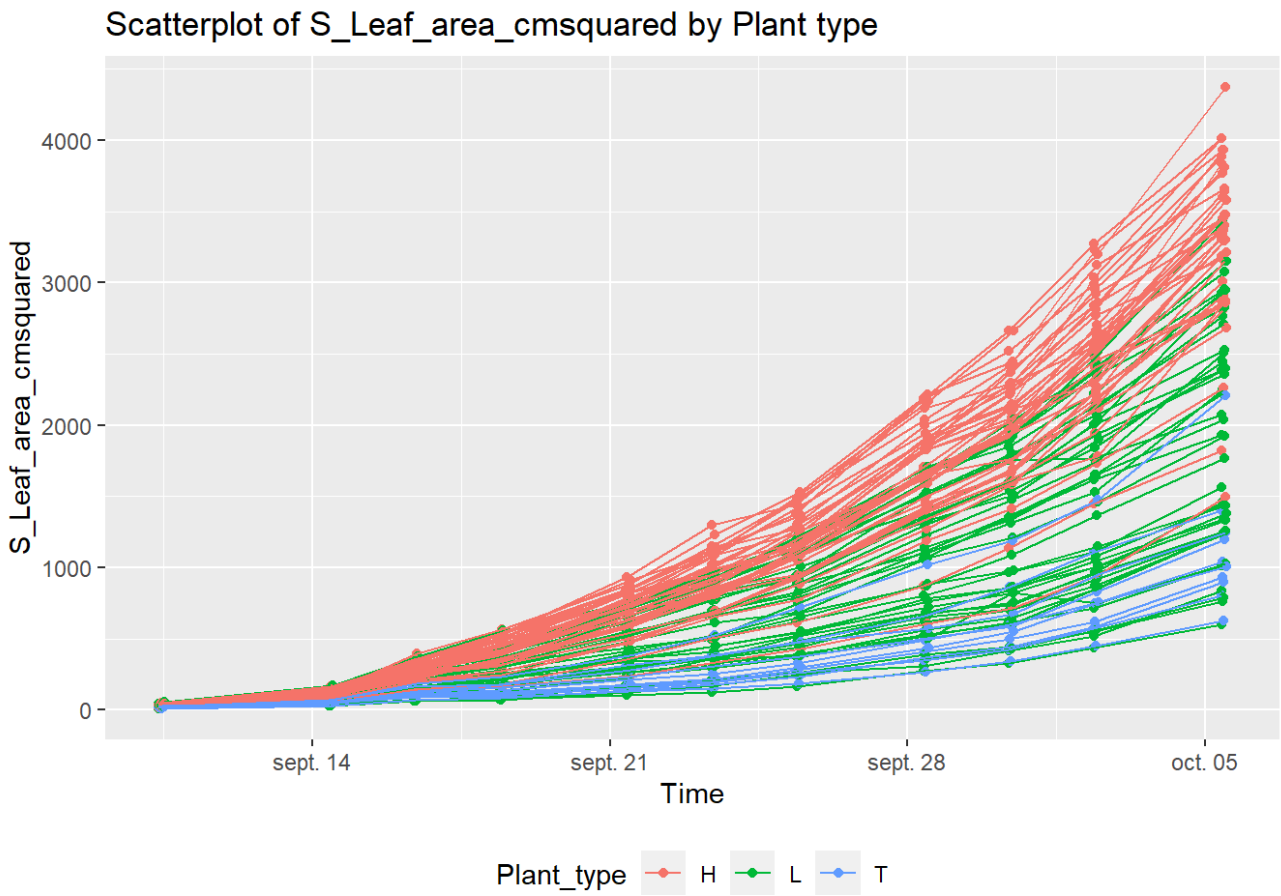

```
## `geom_smooth()` using formula = 'y ~ x'
```



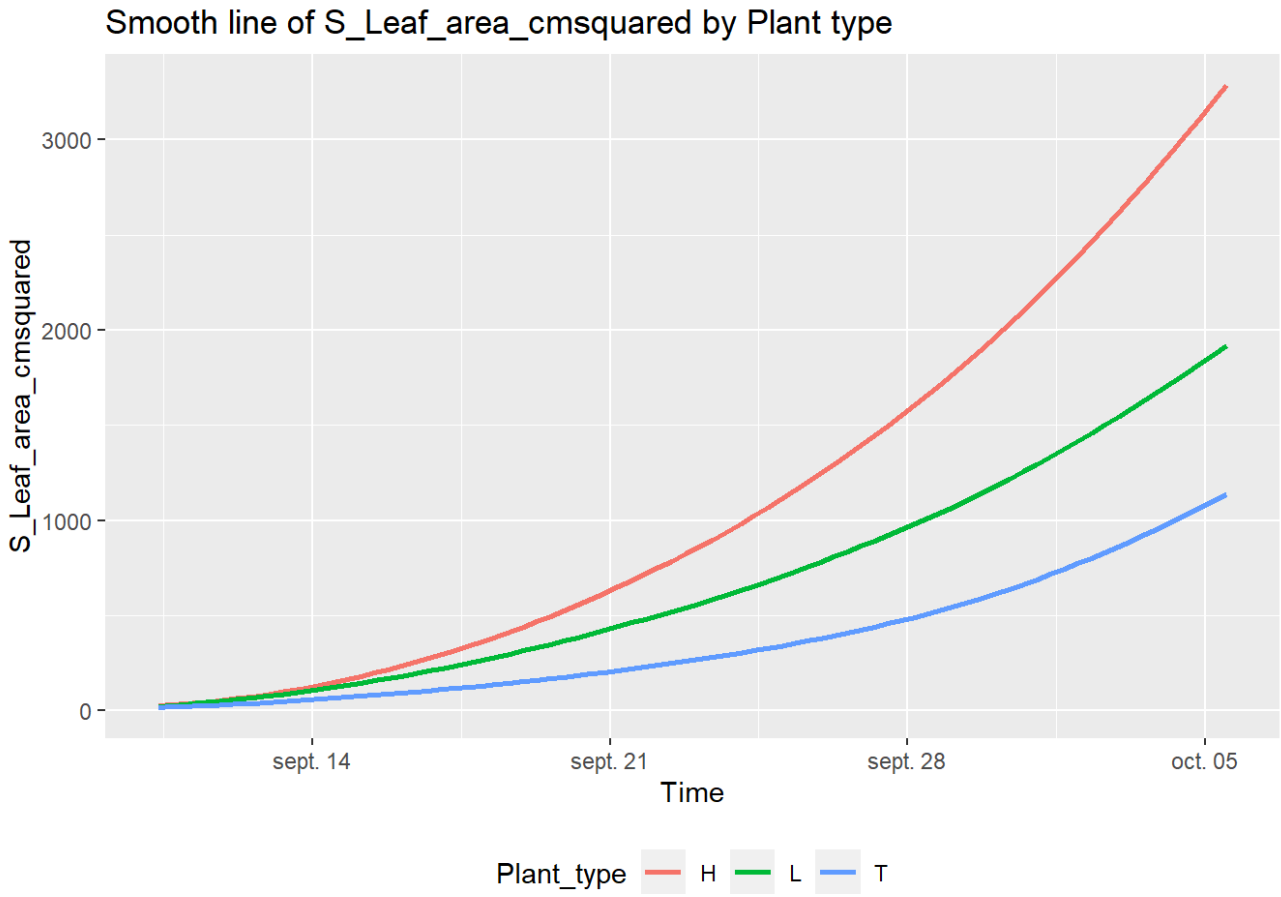



```
## `geom_smooth()` using formula = 'y ~ x'
```






```
## `geom_smooth()` using formula = 'y ~ x'
```



Scatter plots for all genotypes by water treatment

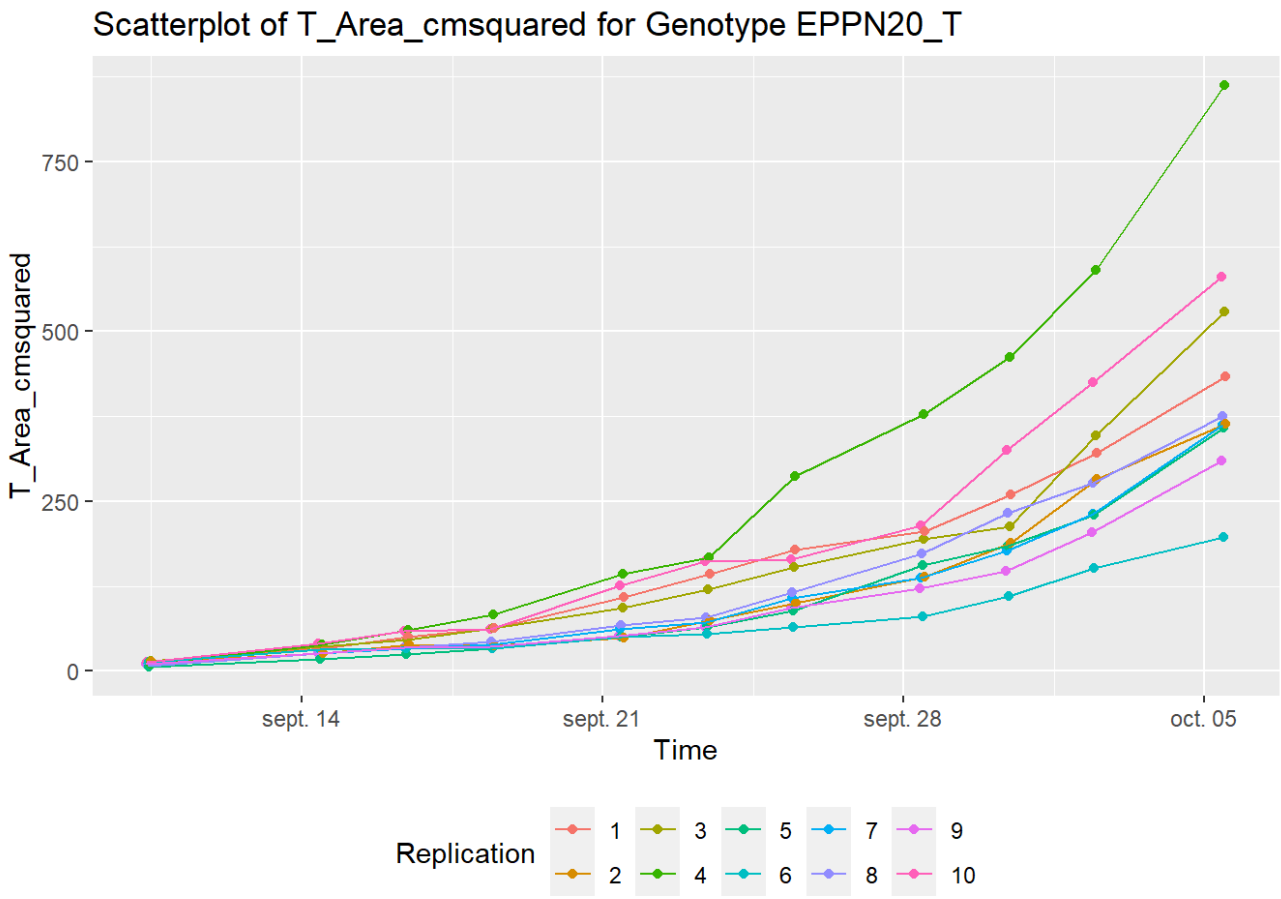
```
print(paste0("No data for", platform))
```

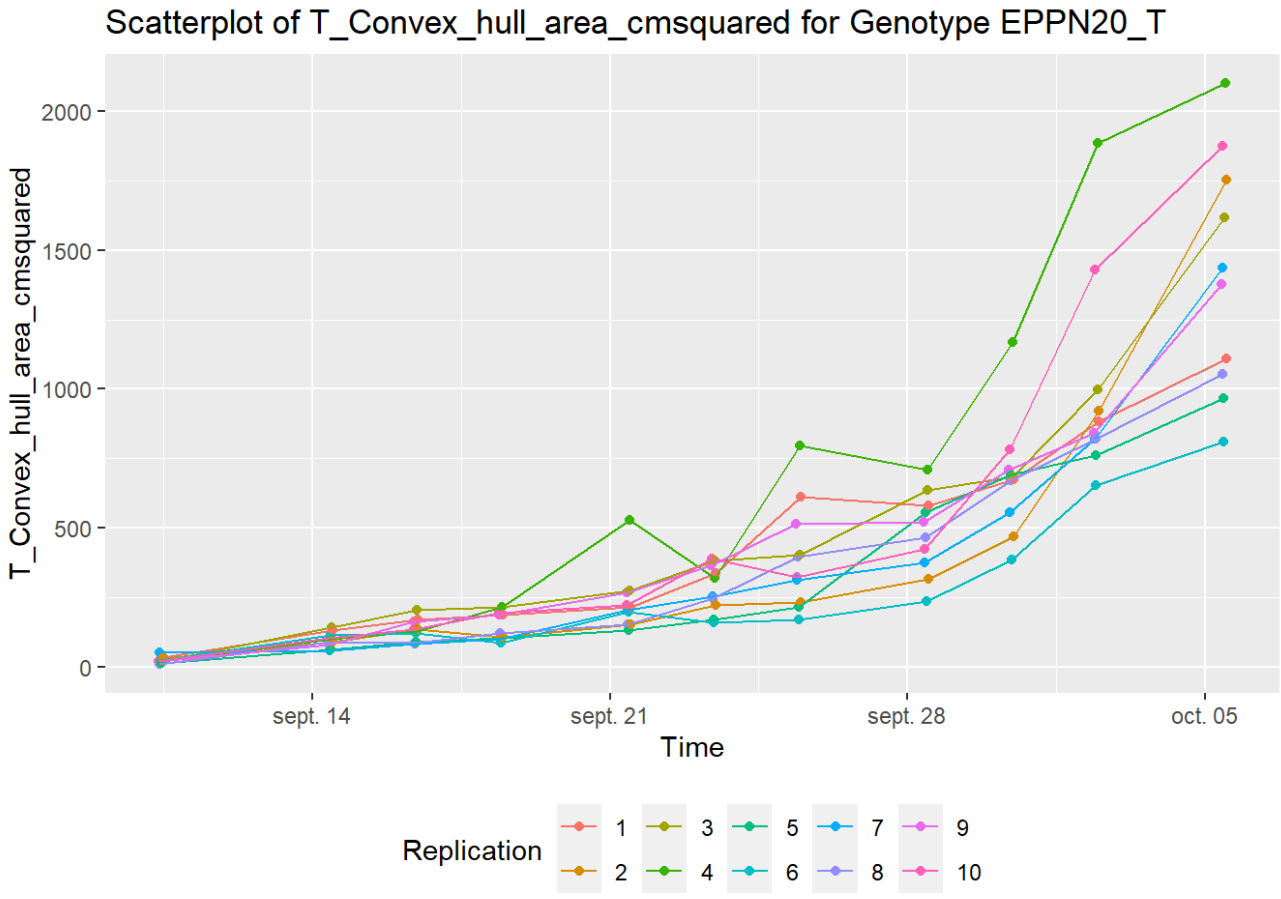
```
## [1] "No data forALSIA"
```

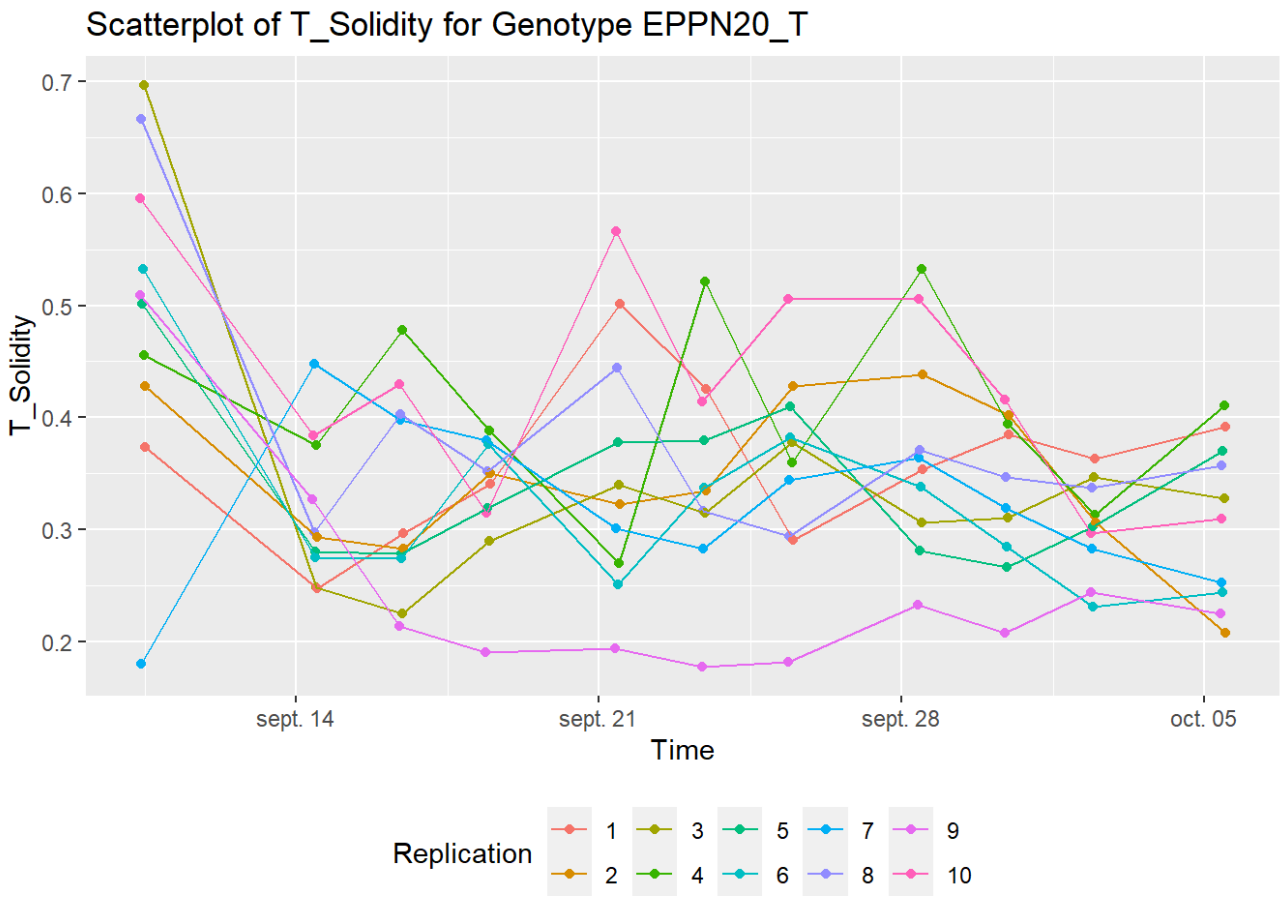
C. Exploration of the T_timeseries dataframe

Scatter plots by Genotype

```
plot_scatter_by_genotype(T_timeseries, variables_T, "EPPN20_T")
```

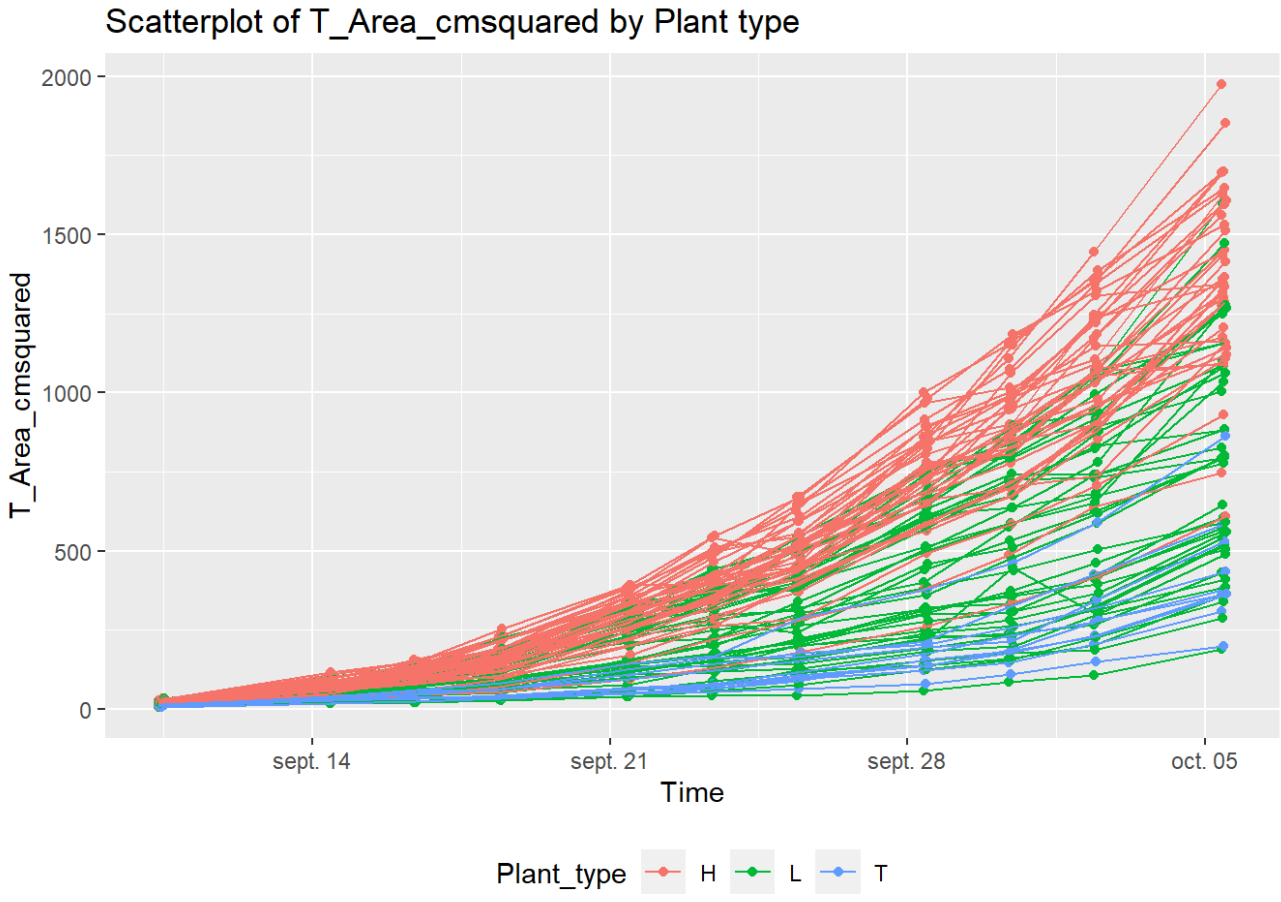

Scatterplot of T_Area_cmsquared for Genotype EPPN20_T
T_Area_cmsquared
750
500
250
0
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10


Scatterplot of T_Convex_hull_area_cmsquared for Genotype EPPN20_T
T_Convex_hull_area_cmsquared
2000
1500
1000
500
0
sept. 14
sept. 21
sept. 28
oct. 05
Time
Replication
1
2
3
4
5
6
7
8
9
10

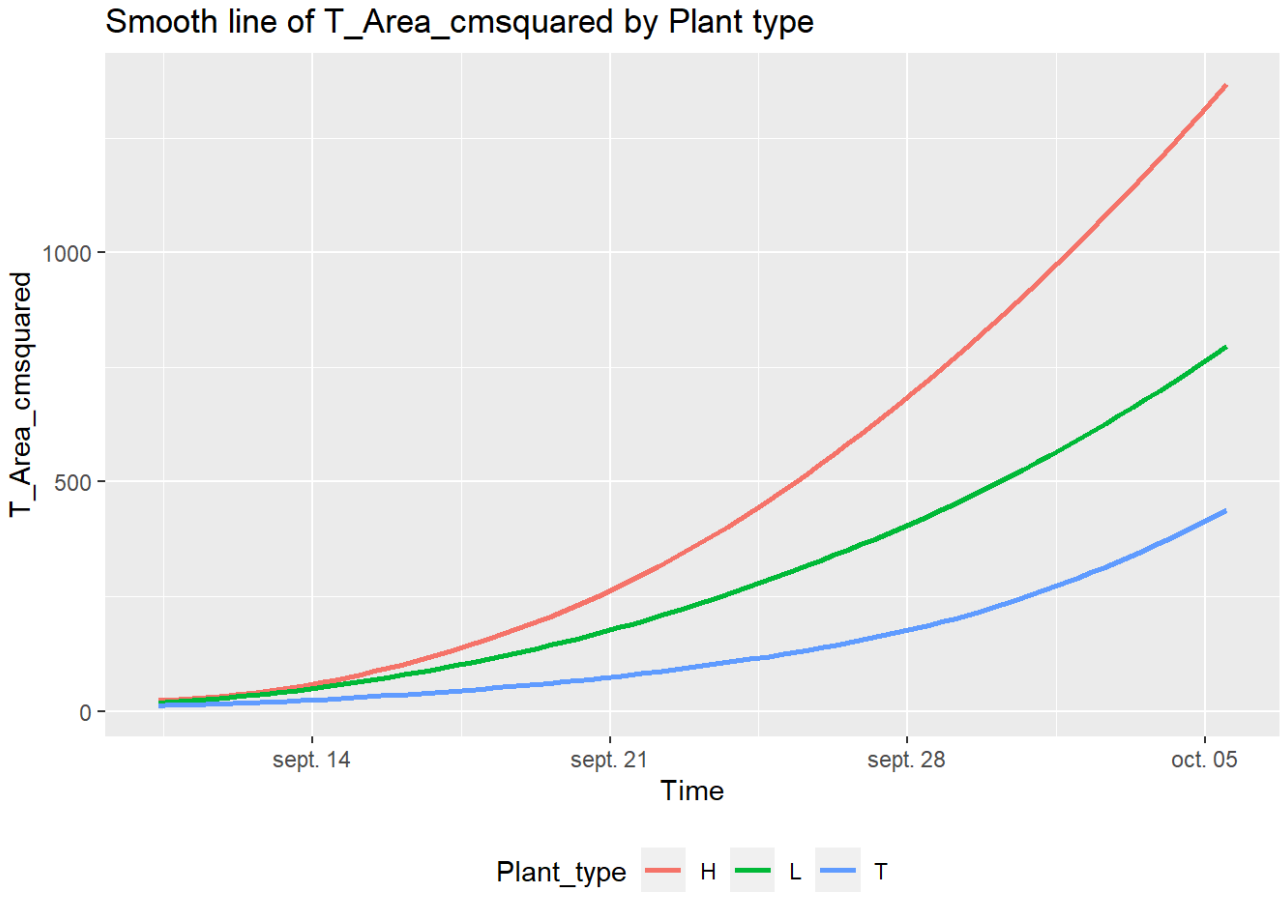


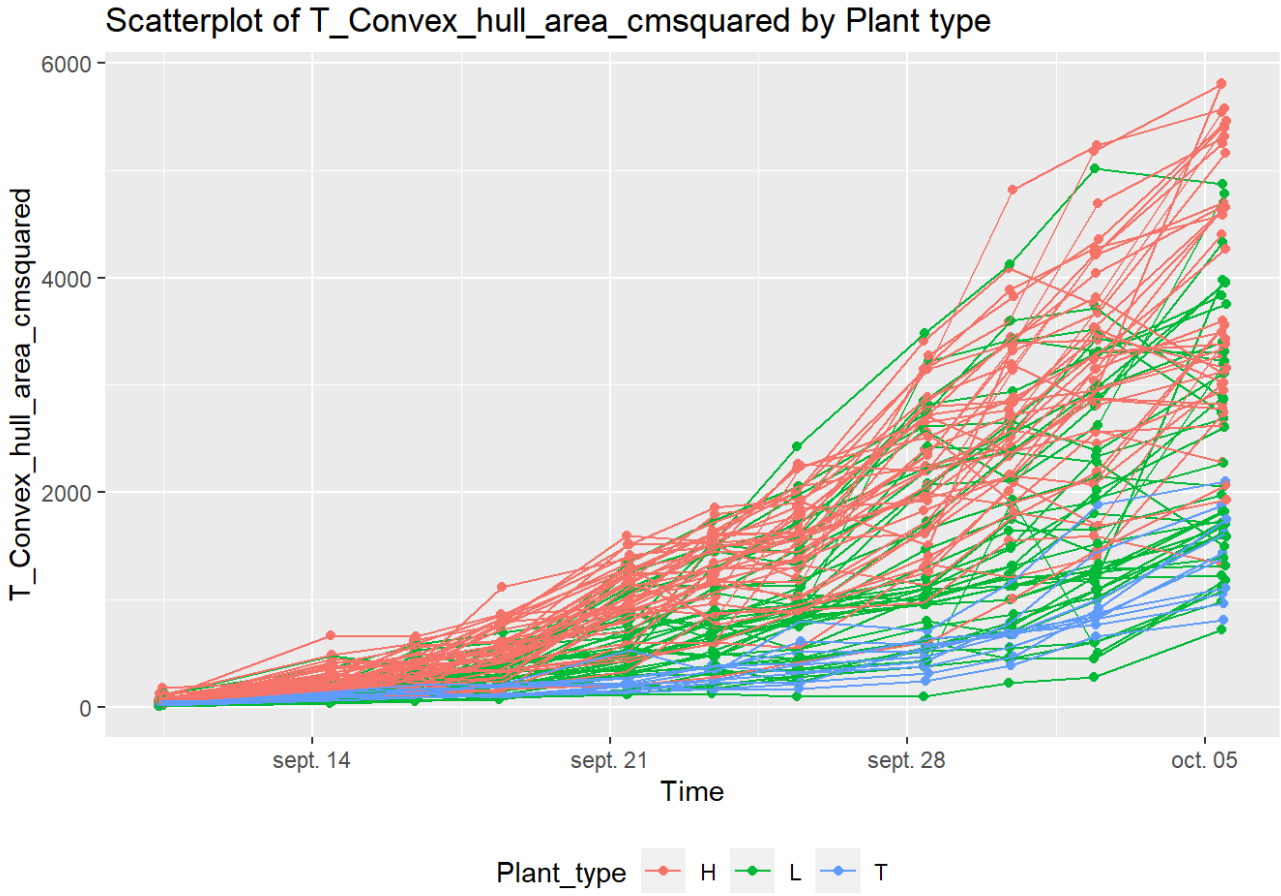
Scatterplots for all genotypes by Plant type (Hybride, Line, EPPN20_T) with smooth line.

```
plot_scatter_with_smooth(T_timeseries, variables_T)
```

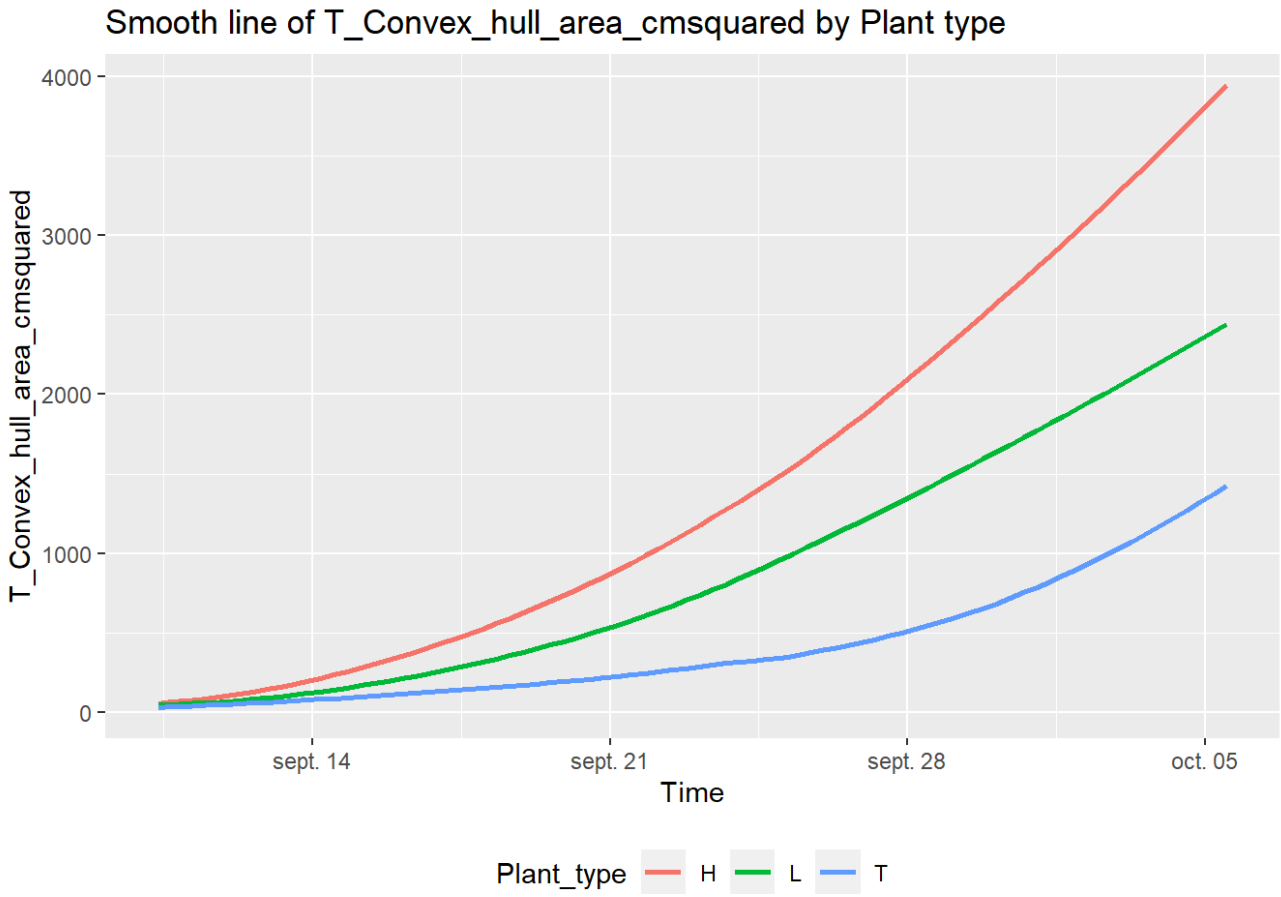


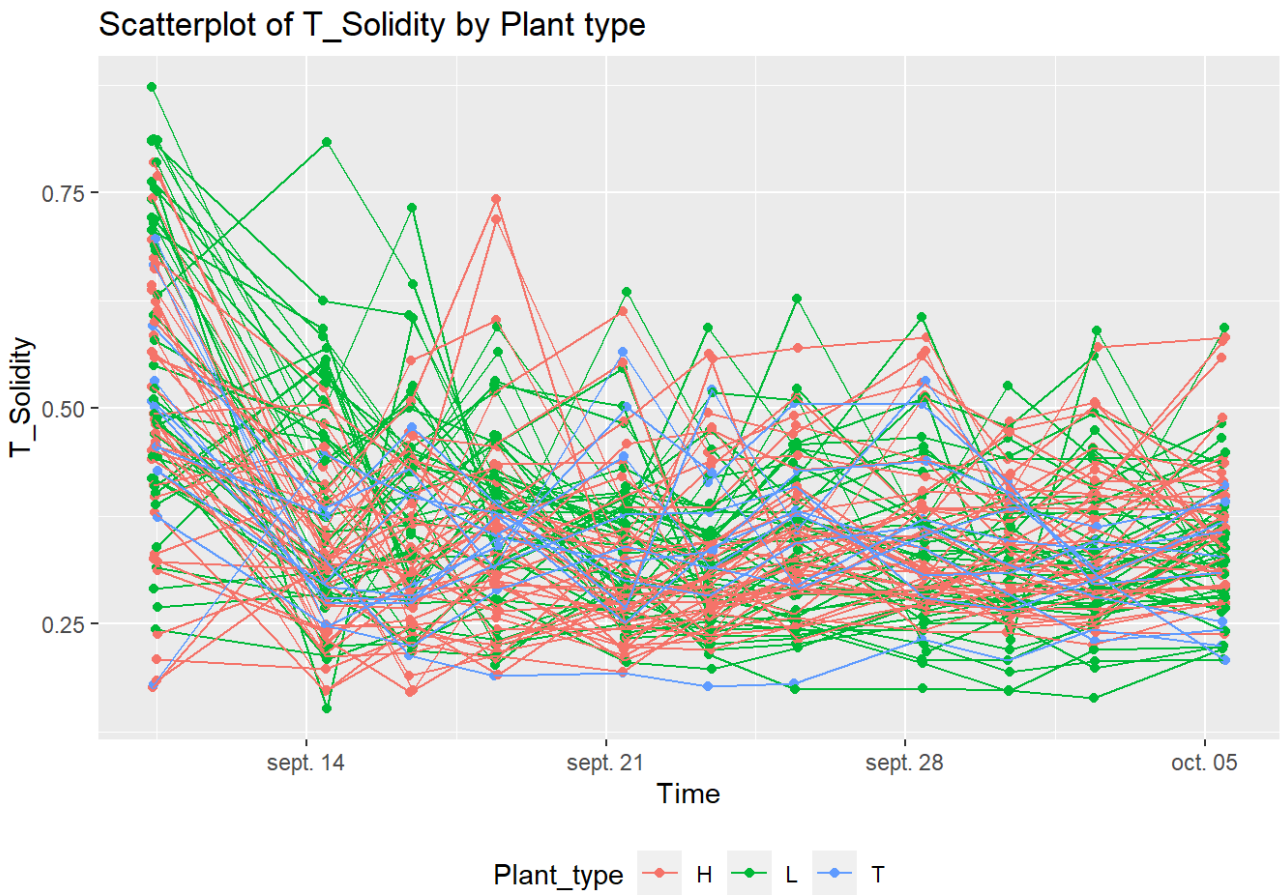
```
## `geom_smooth()` using formula = 'y ~ x'
```



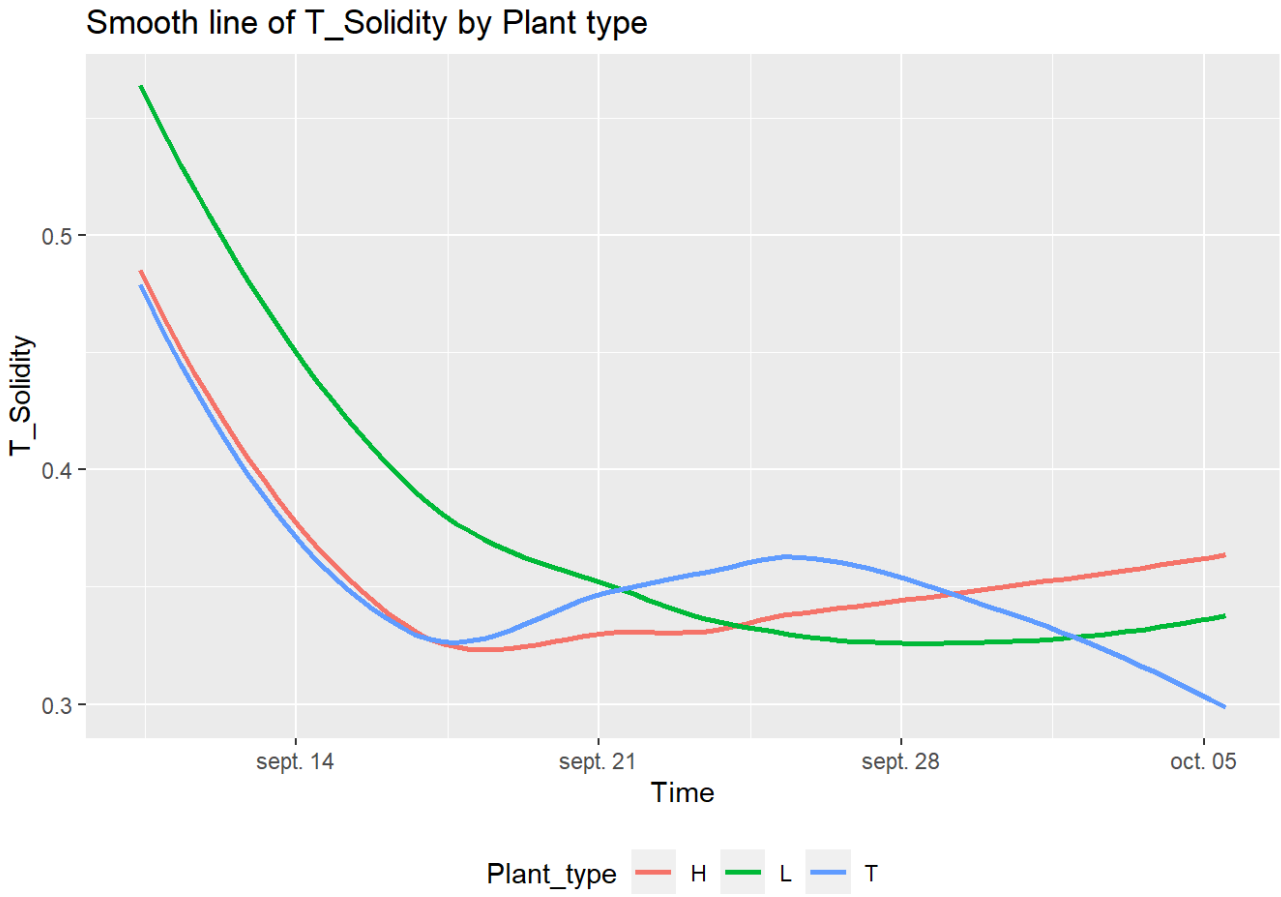



```
## `geom_smooth()` using formula = 'y ~ x'
```






```
## `geom_smooth()` using formula = 'y ~ x'
```



Scatter plots for all genotypes by water treatment

```
print(paste0("No data for", platform))
```

```
## [1] "No data forALSIA"
```