UCPH Data Analysis

Elise

2024-06-09

Set the right working directory.

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

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")
S_timeseries <- read.table("S_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
# No data

# 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
```

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

```
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 3 first columns
that are "Unit.ID" and "Date" etc

# 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"
```

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

1. Endpoint dataframe

A. Exploration of data

Exploration tables using the rstatix, janitor and skimr packages

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

```
##   Genotype  n
## 1   EPPN_T 12
## 2  EPPN1_H 12
## 3  EPPN1_L 12
## 4  EPPN2_H 12
## 5  EPPN2_L 12
## 6  EPPN3_H 12
## 7  EPPN3_L 12
## 8  EPPN4_H 12
## 9  EPPN4_L 12
```

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

```
##  Genotype/Column          1           2           3           4           5
##           EPPN_T  8.3%  (1)  8.3%  (1) 16.7%  (2) 16.7%  (2)  8.3%  (1)
##          EPPN1_H  8.3%  (1)  8.3%  (1) 16.7%  (2) 16.7%  (2)  8.3%  (1)
##          EPPN1_L  8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1)  8.3%  (1)
##          EPPN2_H 16.7%  (2)  8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1)
##          EPPN2_L 16.7%  (2)  8.3%  (1)  8.3%  (1)  8.3%  (1) 16.7%  (2)
##          EPPN3_H  8.3%  (1)  8.3%  (1)  8.3%  (1) 16.7%  (2)  8.3%  (1)
##          EPPN3_L 16.7%  (2)  8.3%  (1)  8.3%  (1)  8.3%  (1)  8.3%  (1)
##          EPPN4_H  8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1) 16.7%  (2)
##          EPPN4_L  8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1) 16.7%  (2)
##            Total 11.1% (12) 11.1% (12) 11.1% (12) 11.1% (12) 11.1% (12)
##           6           7           8           9
##   8.3%  (1)  8.3%  (1) 16.7%  (2)  8.3%  (1)
##   8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1)
##  16.7%  (2)  8.3%  (1)  8.3%  (1) 16.7%  (2)
##   8.3%  (1)  8.3%  (1)  8.3%  (1) 16.7%  (2)
##   8.3%  (1)  8.3%  (1) 16.7%  (2)  8.3%  (1)
##   8.3%  (1) 16.7%  (2)  8.3%  (1) 16.7%  (2)
##  16.7%  (2)  8.3%  (1) 16.7%  (2)  8.3%  (1)
##  16.7%  (2)  8.3%  (1)  8.3%  (1)  8.3%  (1)
##   8.3%  (1) 16.7%  (2)  8.3%  (1)  8.3%  (1)
##  11.1% (12) 11.1% (12) 11.1% (12) 11.1% (12)
```

```
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        7
##       EPPN_T 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN1_H 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN1_L 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN2_H 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN2_L 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN3_H 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN3_L 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN4_H 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##      EPPN4_L 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##        Total 8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9)
##         8        9       10       11       12
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1) 8.3% (1)
##  8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9) 8.3% (9)
```

```
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 DW_shoot_g   107 0.768     8   3.22  2.00 3.46  1.65   0.16 0.317
## 2 DW_root_g    108 0.1       5   0.8   0.7  0.982 0.832  0.08 0.159
```

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

Data summary

Name	endpoint[variables]
Number of rows	108
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
DW_shoot_g	1	0.99	3.46	1.65	0.77	2.3	3.22	4.3	8	▆▇▅▂▁
DW_root_g	0	1.00	0.98	0.83	0.10	0.5	0.80	1.2	5	▇▂▁▁▁

Data visualization

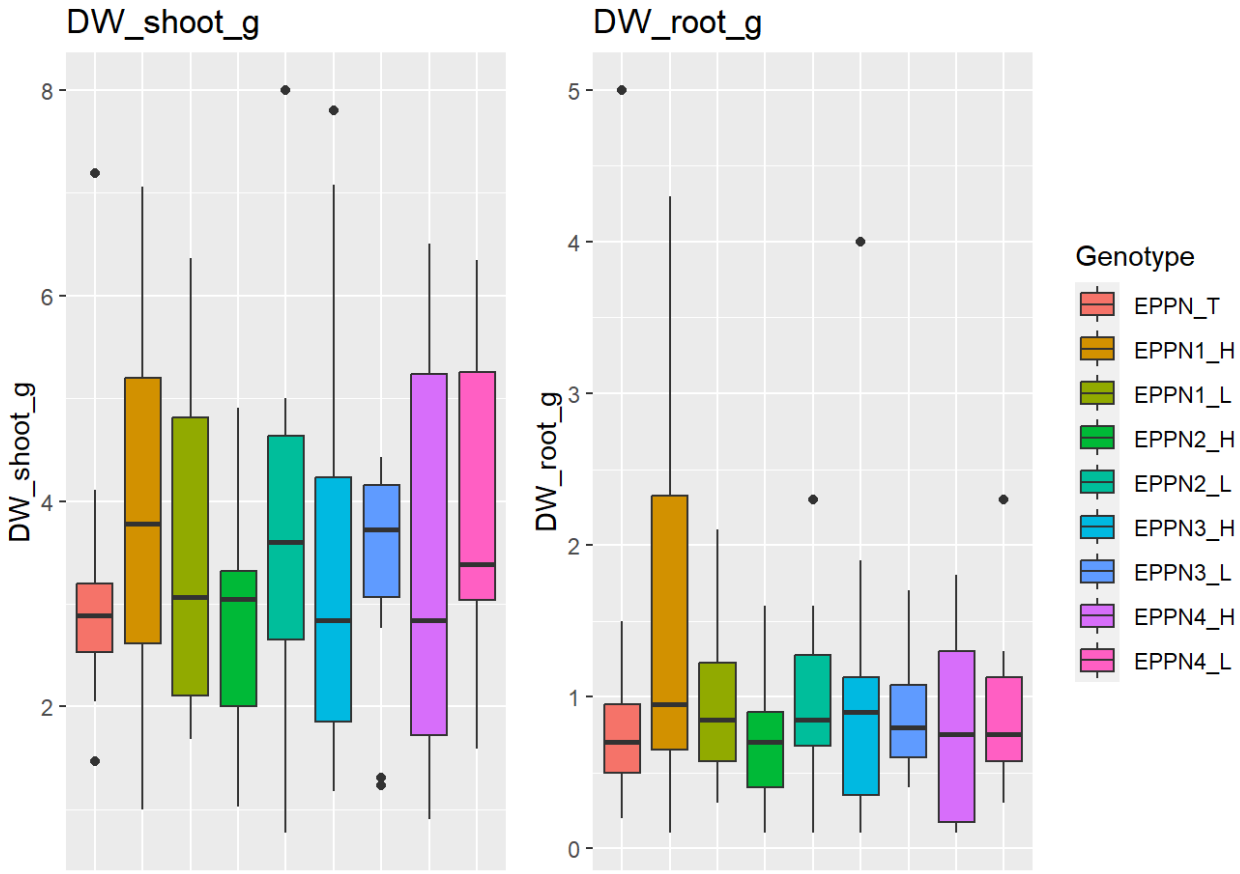
Using several functions that are located in the functions.R script

Boxplots

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

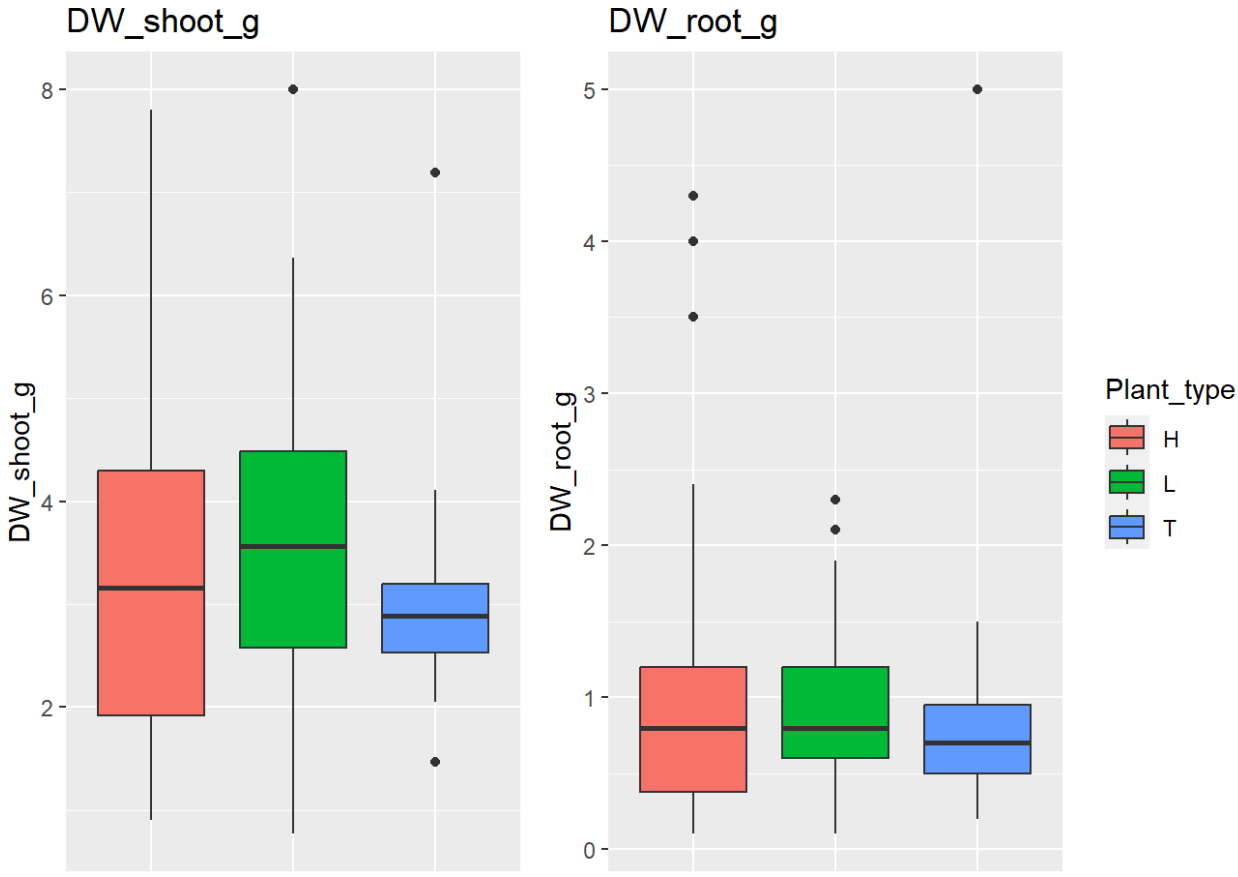
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this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

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




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

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



Correlation plots

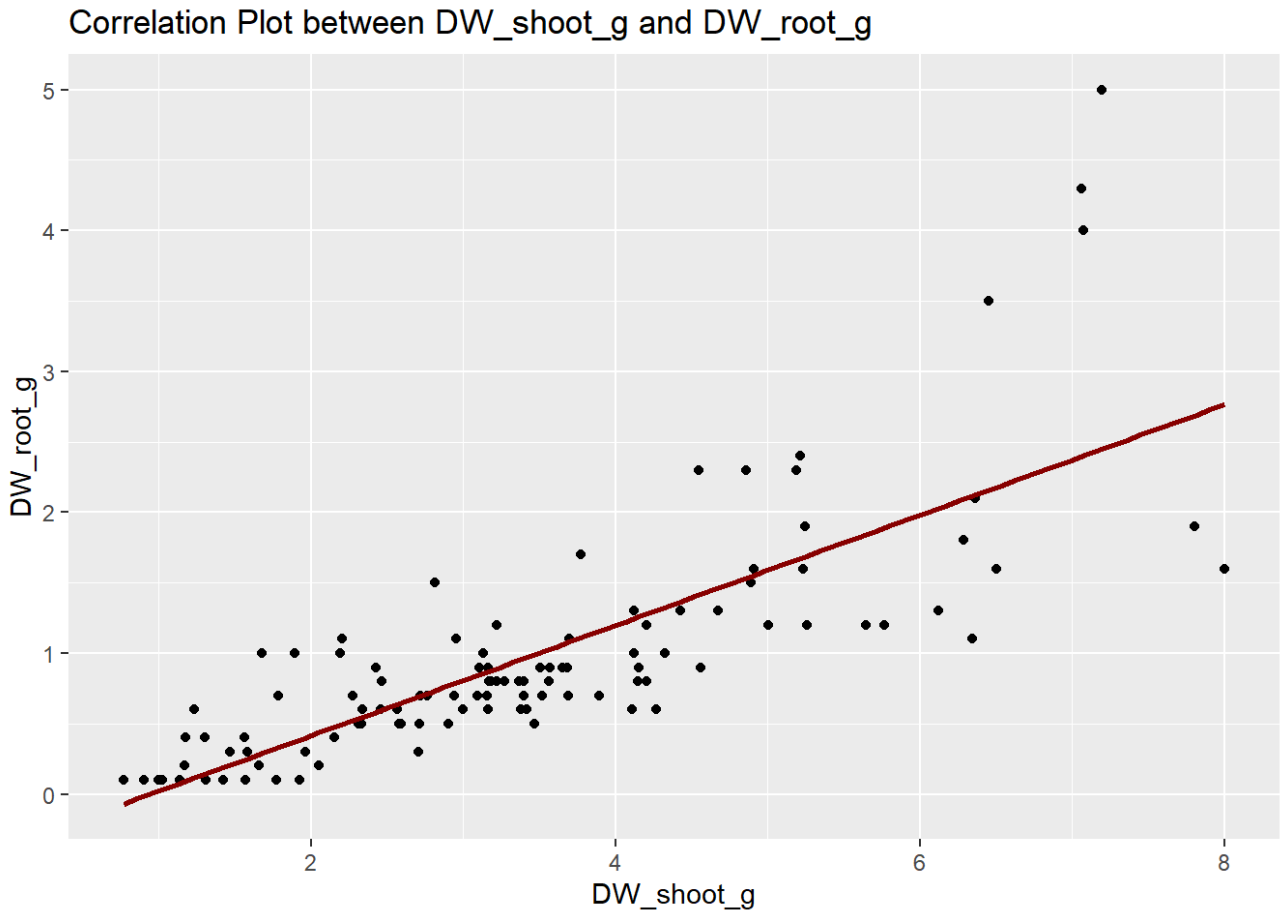
```
for (i in 1:(length(variables) - 1)) {
  for (j in (i + 1):length(variables)) {
    calculate_correlation_plot(endpoint, variables[i], variables[j])
  }
}
```

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

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

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

```
## Warning: Removed 108 rows containing missing values (`geom_text()`).
```

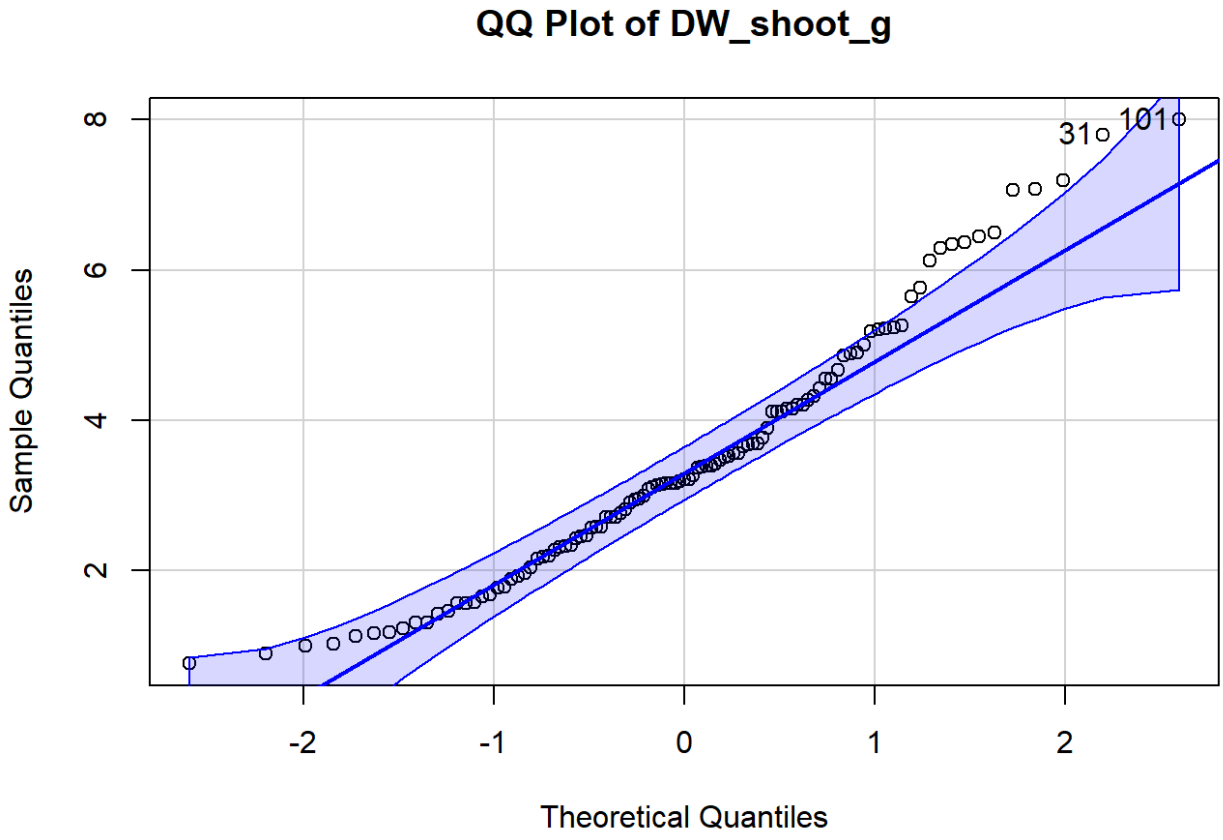


B. Normality hypothesis and outlier detection

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.
```

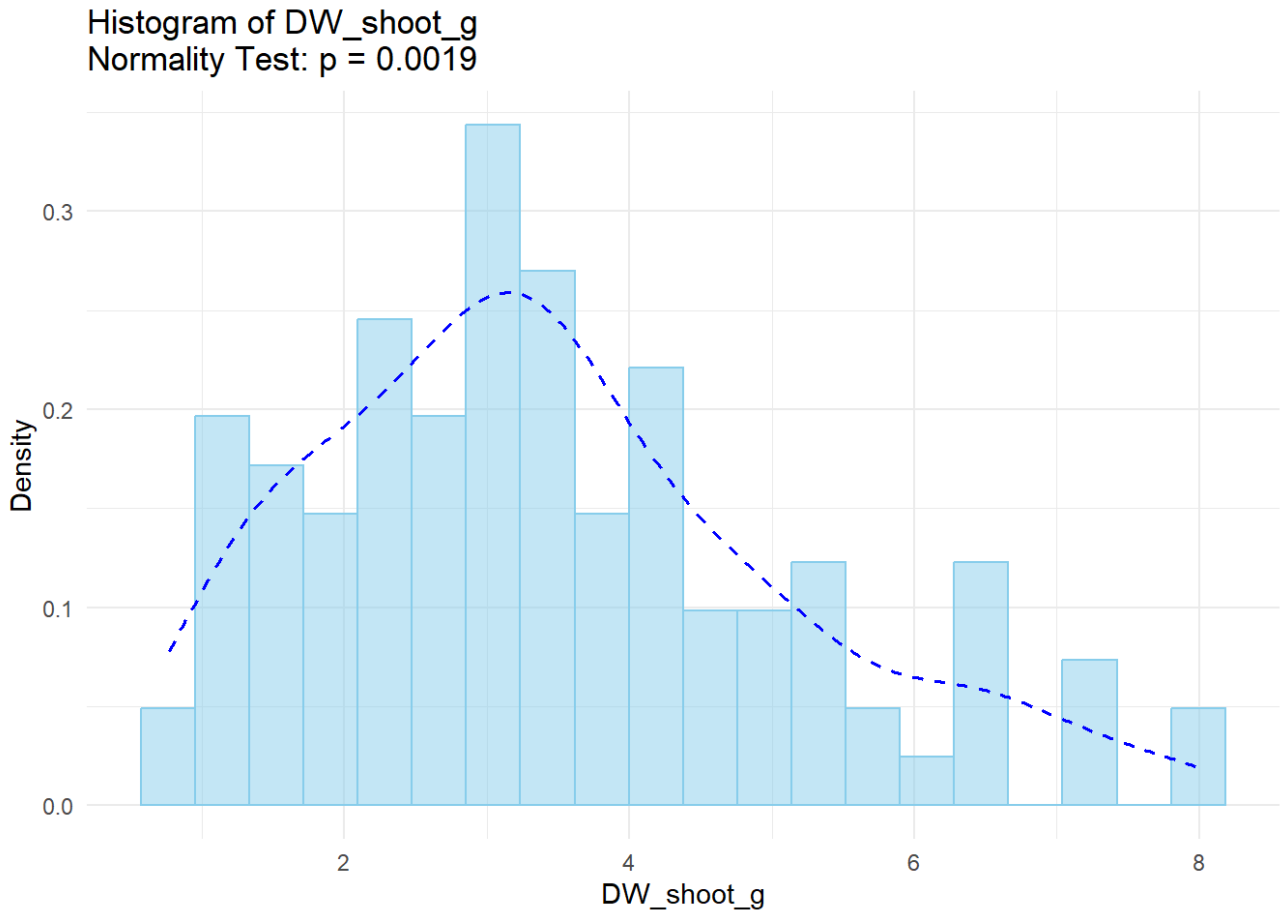



```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

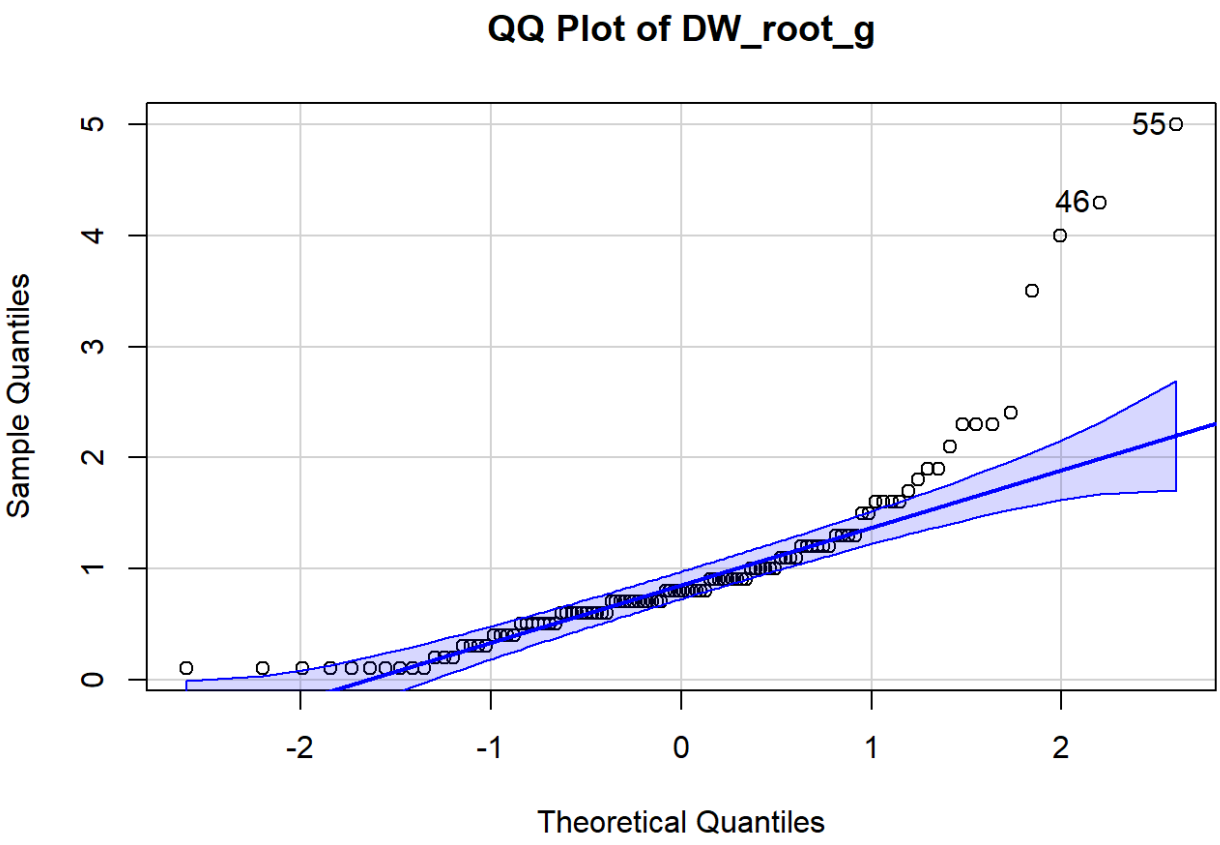
```
## Warning: Removed 1 rows containing non-finite values (`stat_bin()`).
```

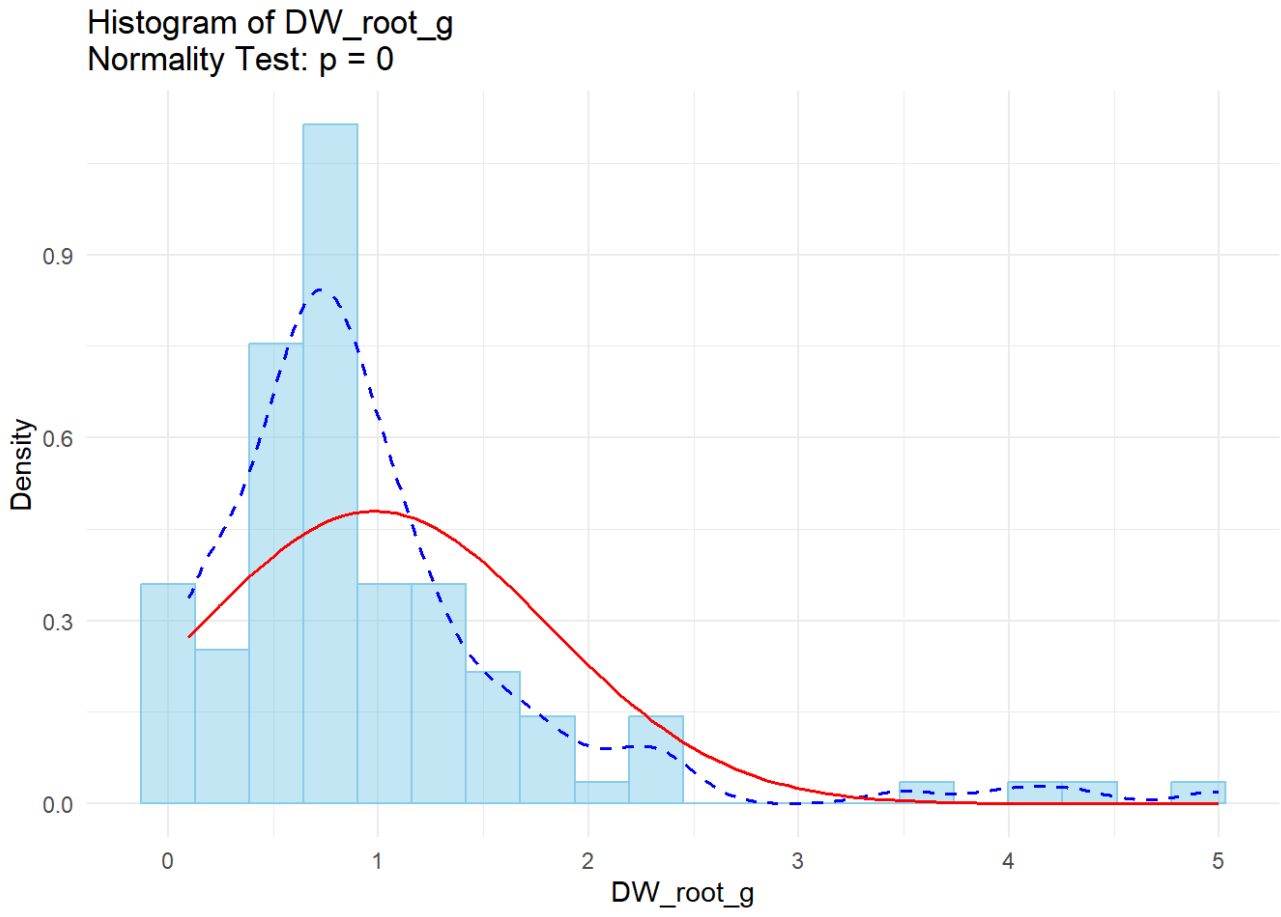
```
## Warning: Removed 1 rows containing non-finite values (`stat_density()`).
```

```
## Warning: Removed 101 rows containing missing values (`geom_function()`).
```


Histogram of DW_shoot_g
Normality Test: p = 0.0019
Density
0.0
0.1
0.2
0.3
2
4
6
8
DW_shoot_g

```
## [1] 101  31
```






```
## [1] 55 46
```

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.

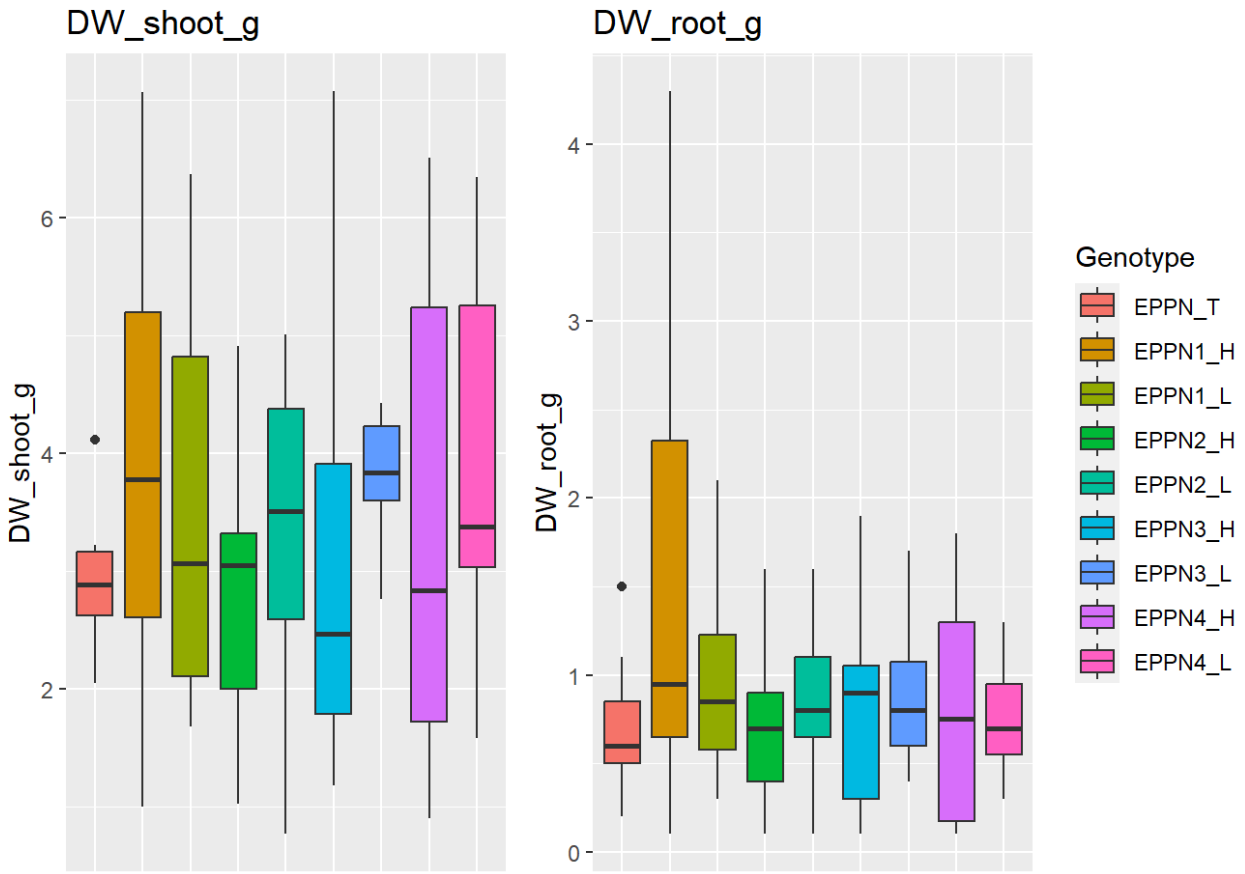
```
endpoint_clean <- endpoint
# Run the function on the dataset for all the variables
endpoint_clean <- detect_replace_outliers_by_genotype(endpoint_clean)
```

Boxplots after outlier detection

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

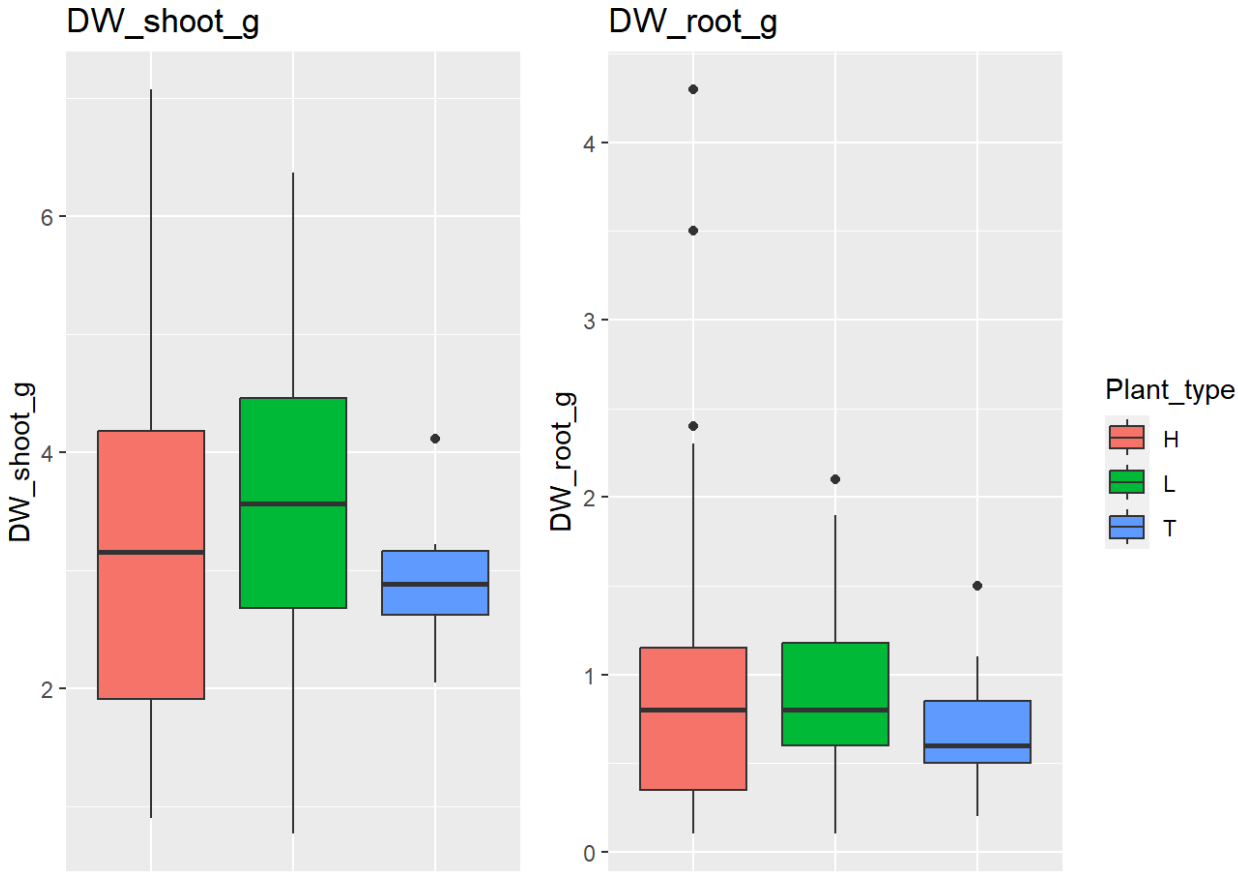
```
## Warning: Removed 7 rows containing non-finite values (`stat_boxplot()`).
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_boxplot()`).
```




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

```
## Warning: Removed 7 rows containing non-finite values (`stat_boxplot()`).
## Removed 4 rows containing non-finite values (`stat_boxplot()`).
```



Violin and sina plots after outlier detection

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

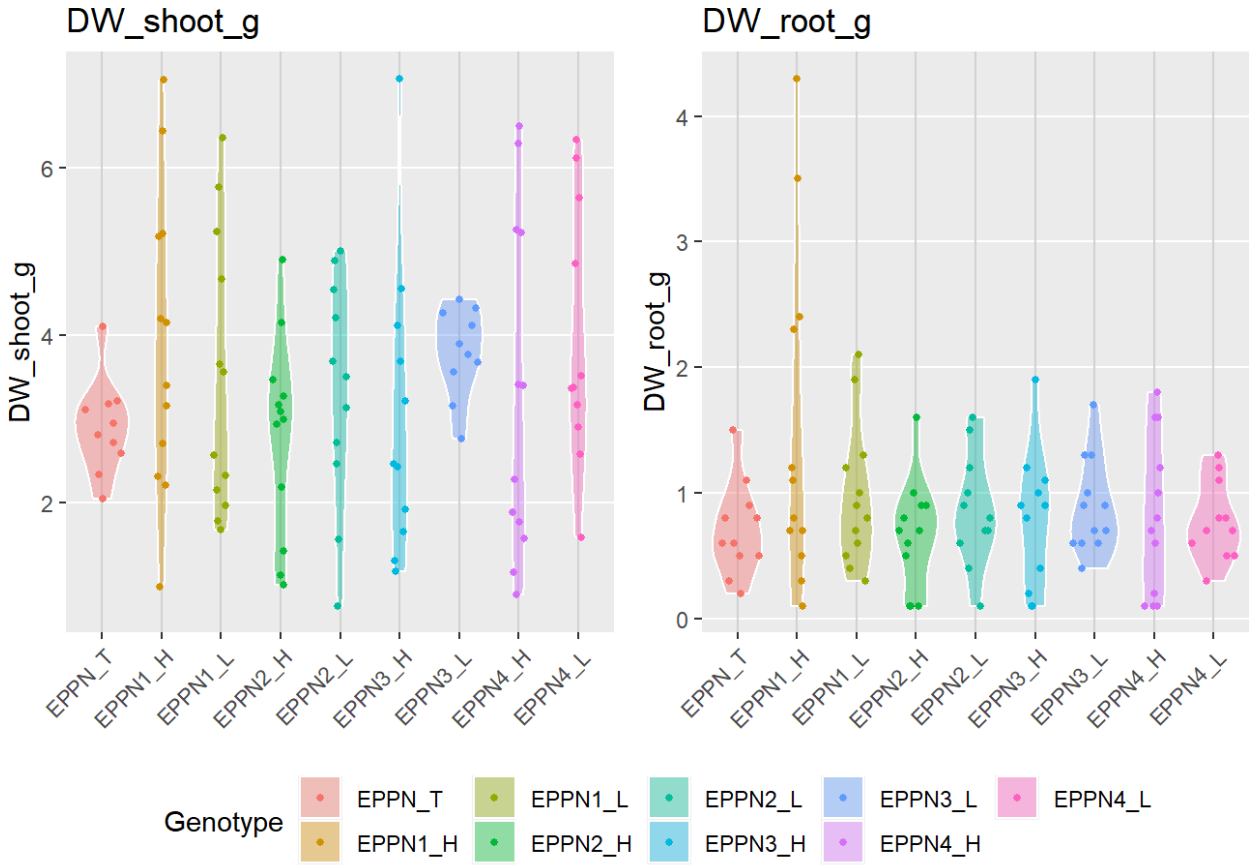
```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## ℹ Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_sina()`).
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_sina()`).
```



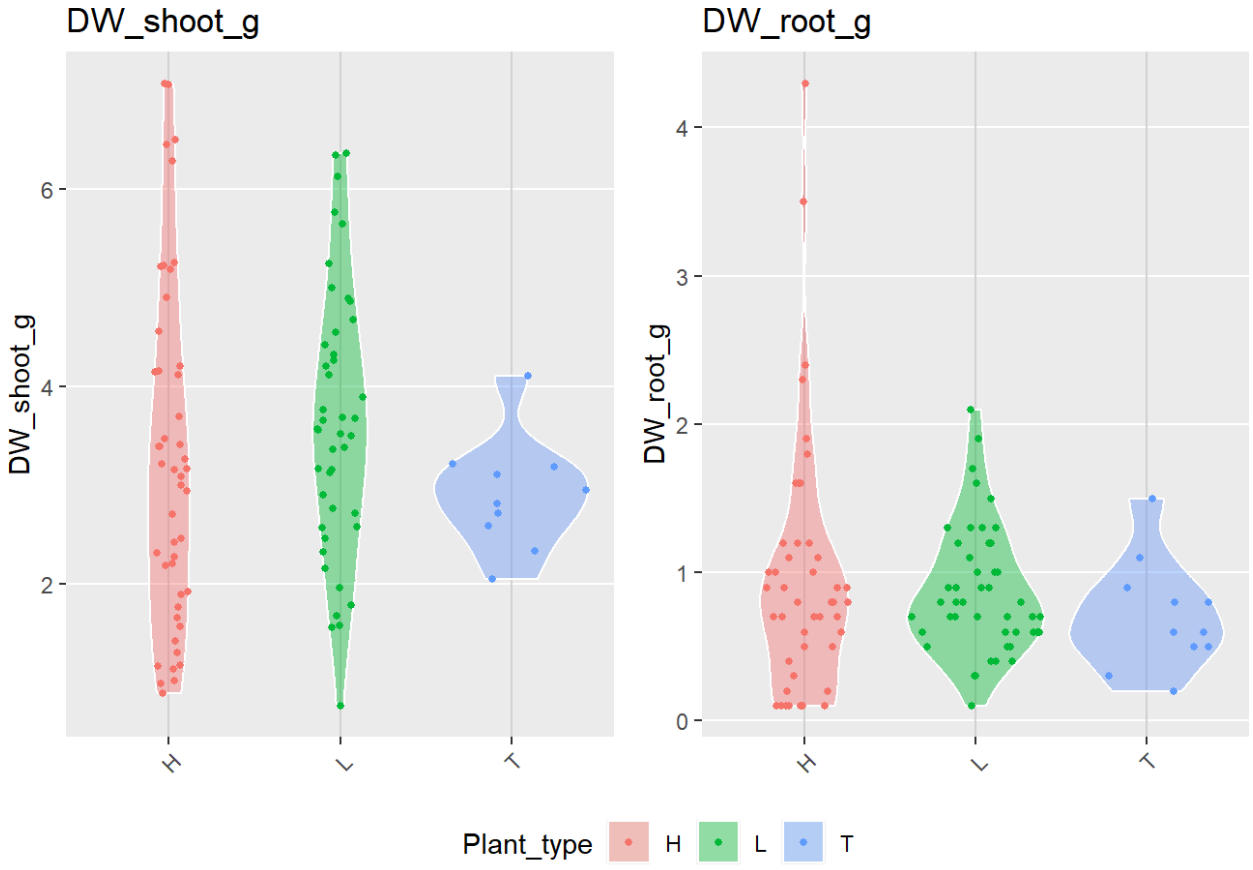

```
create_violin_plots(endpoint_clean, variables, "Plant_type")
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_sina()`).
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_ydensity()`).
```

```
## Warning: Removed 4 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	108
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
DW_shoot_g	7	0.94	3.40	1.49	0.77	2.33	3.22	4.21	7.07	▃▇▅▂▁
DW_root_g	4	0.96	0.89	0.66	0.10	0.50	0.80	1.10	4.30	▇▃▁▁▁

```
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: DW_shoot_g
## # A tibble: 9 × 4
##   Genotype  mean std.dev n_missing
##   <fct>    <dbl>   <dbl>     <int>
## 1 EPPN4_L   3.95   1.56          1
## 2 EPPN1_H   3.92   1.81          0
## 3 EPPN3_L   3.80   0.532         2
## 4 EPPN1_L   3.48   1.66          0
## 5 EPPN2_L   3.32   1.36          1
## 6 EPPN4_H   3.31   2.03          0
## 7 EPPN3_H   3.06   1.75          1
## 8 EPPN_T    2.91   0.565         2
## 9 EPPN2_H   2.81   1.18          0
##
## Summary for: DW_root_g
## # A tibble: 9 × 4
##   Genotype  mean std.dev n_missing
##   <fct>    <dbl>   <dbl>     <int>
## 1 EPPN1_H  1.49    1.34          0
## 2 EPPN1_L  0.975   0.567         0
## 3 EPPN3_L  0.892   0.378         0
## 4 EPPN2_L  0.864   0.448         1
## 5 EPPN4_H  0.817   0.629         0
## 6 EPPN3_H  0.782   0.549         1
## 7 EPPN4_L  0.773   0.313         1
## 8 EPPN_T   0.709   0.370         1
## 9 EPPN2_H  0.667   0.438         0
```

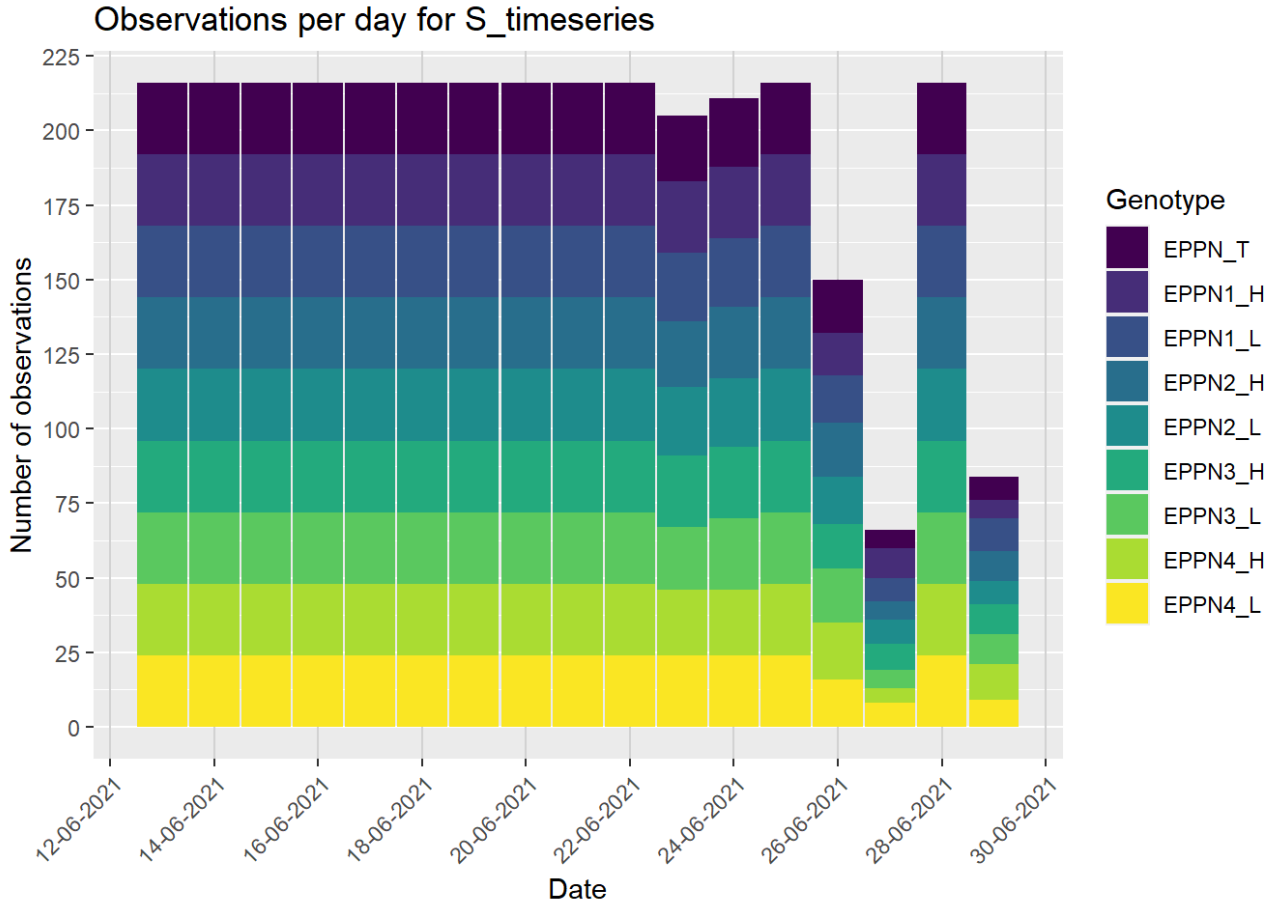
2. Exploration of the timeseries data

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

```
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_ti
meseries") +
  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
```



A. Exploration of the timeseries dataframe

Scatter plots by Genotype

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

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

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

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

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

Scatter plots for all genotypes by water treatment

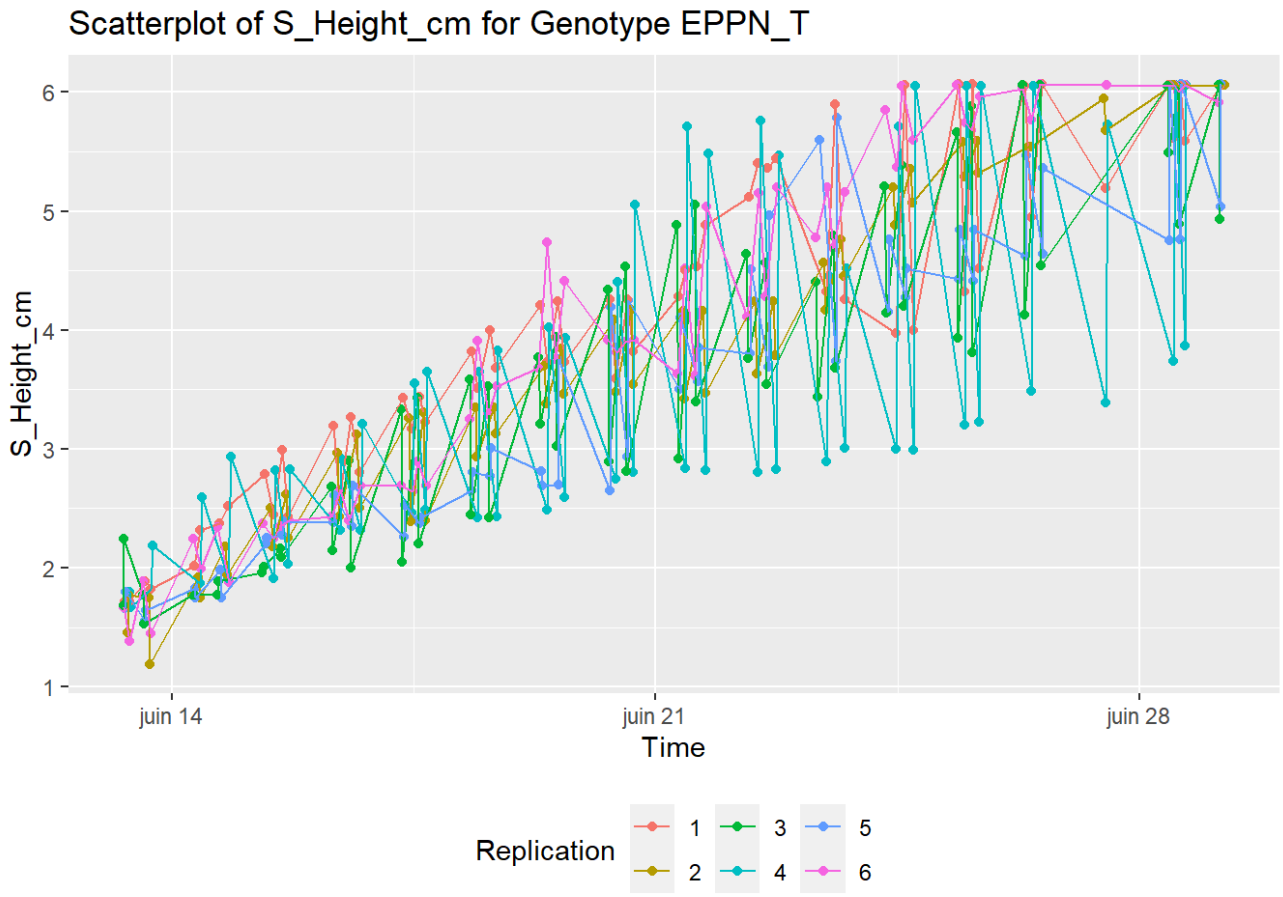
```
print(paste0("No data for", platform))
```

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

B. Exploration of the S_timeseries dataframe

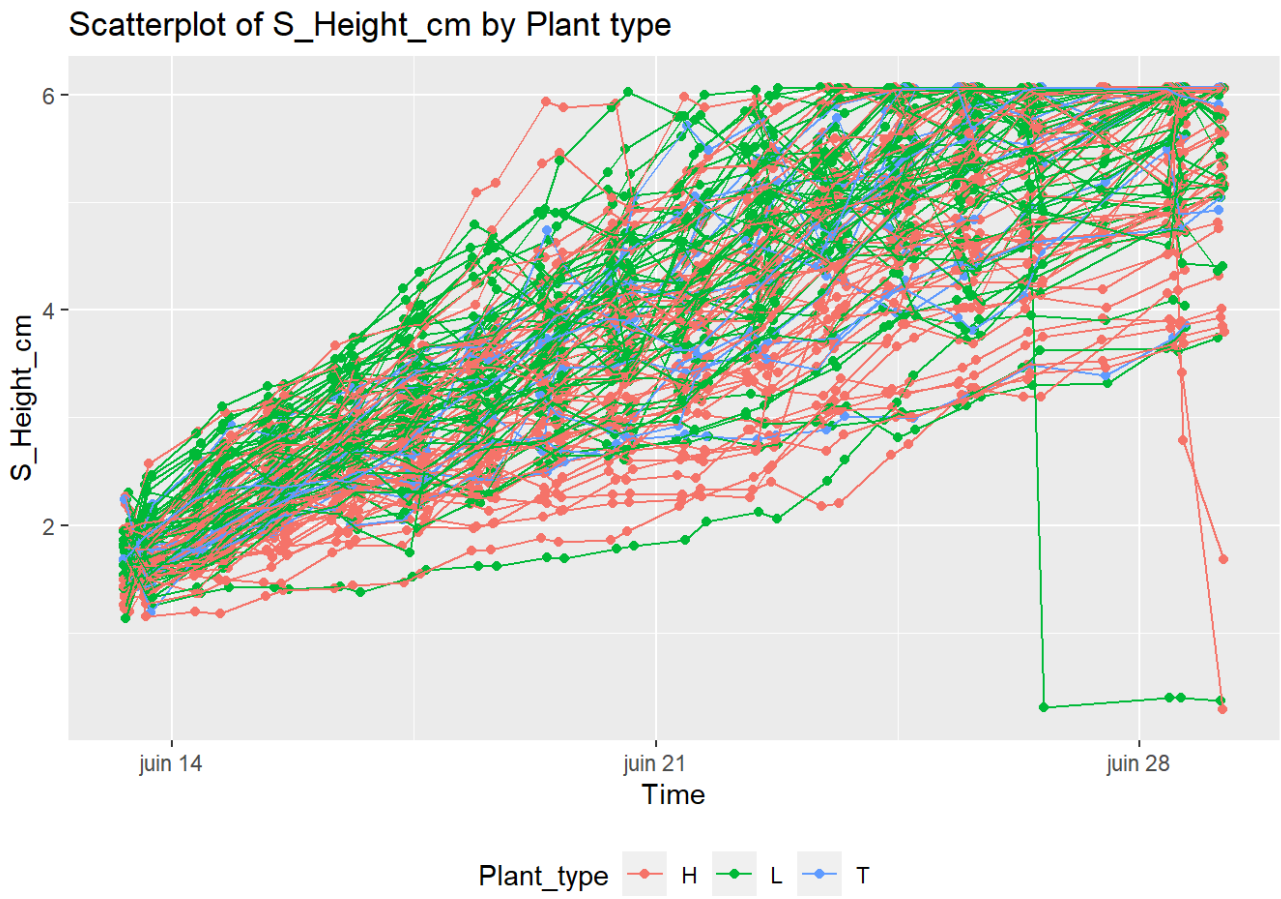
Scatter plots by Genotype

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

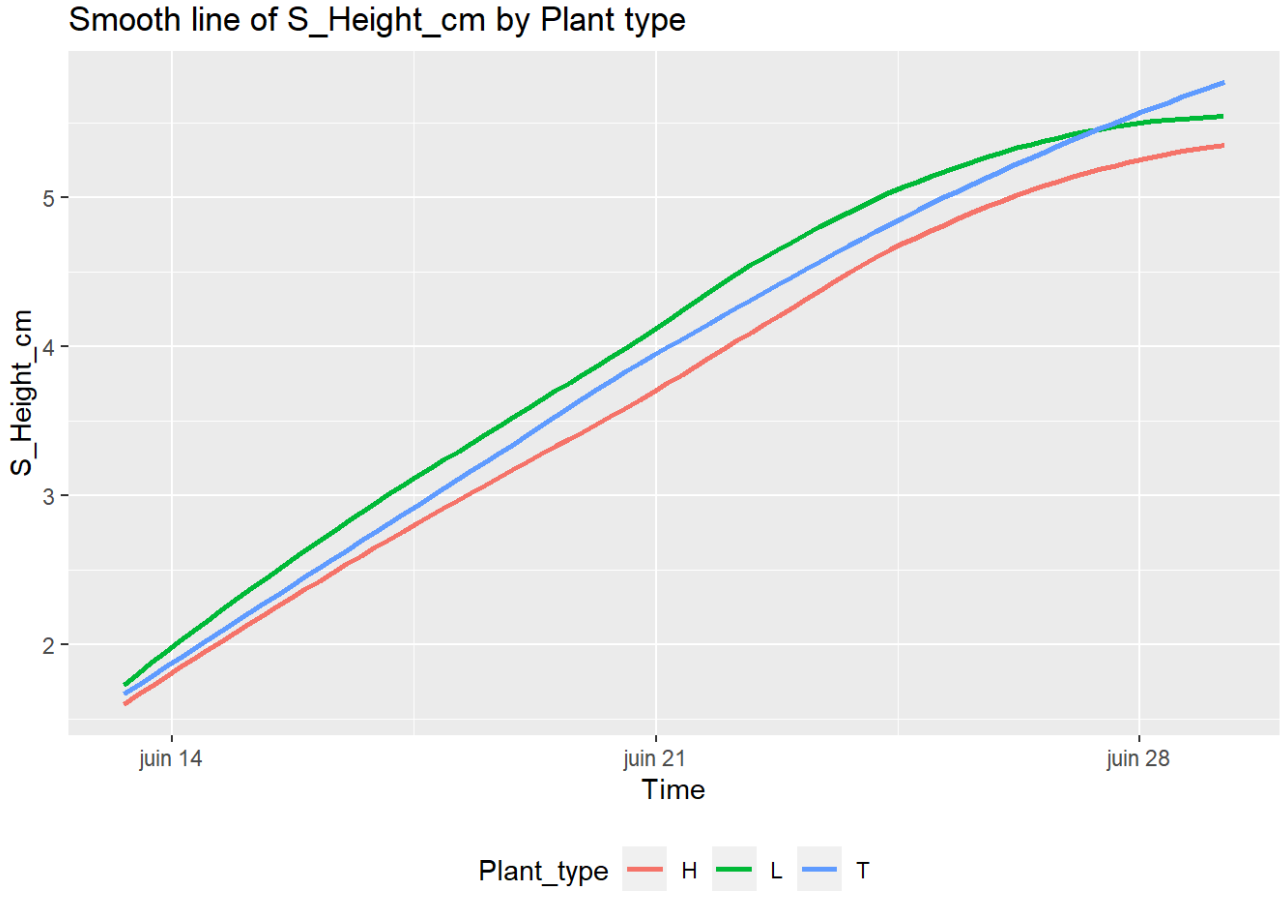


Scatterplots for all genotypes by Plant type (Hybride, Line, EPPN20_T) with smooth line.

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

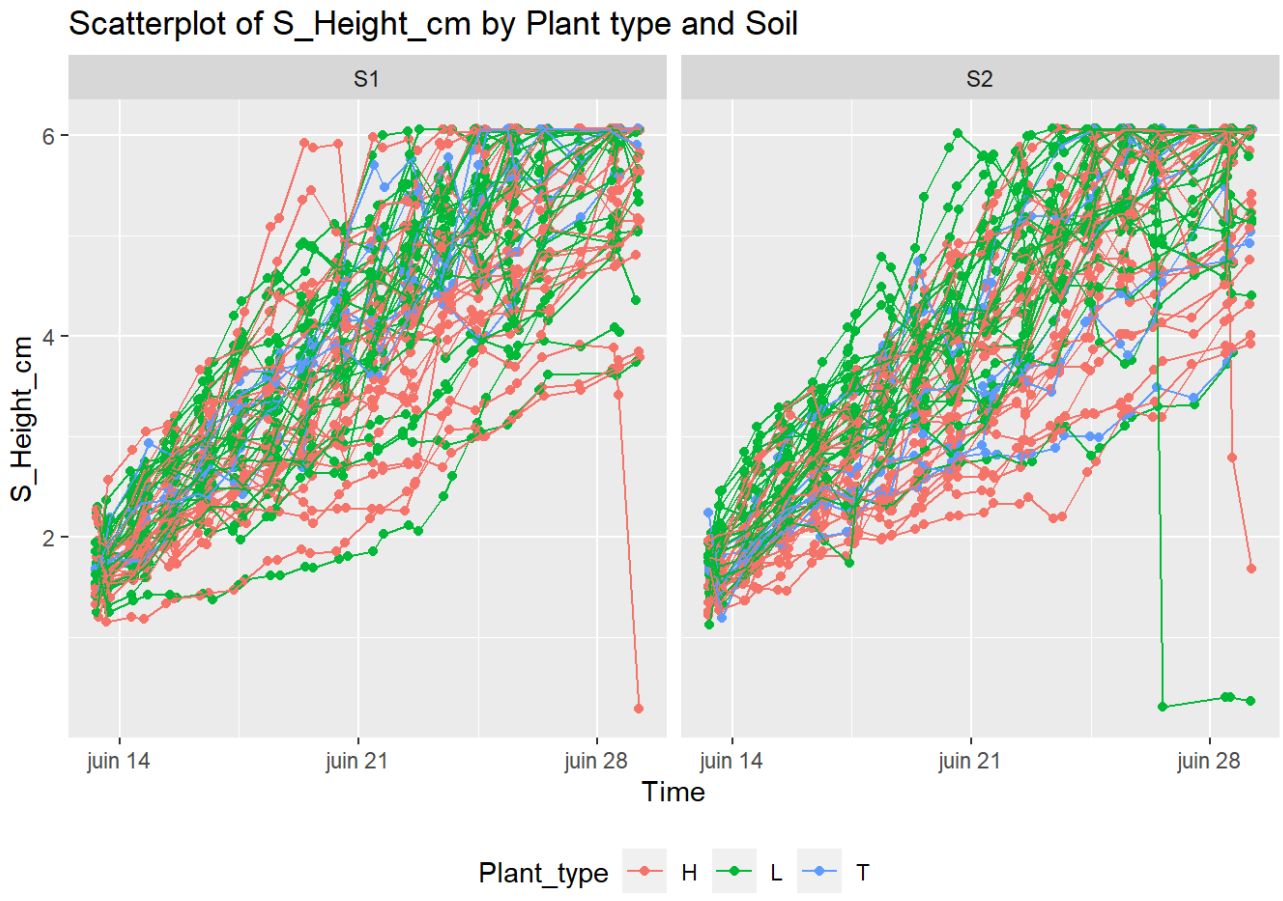



```
## `geom_smooth()` using formula = 'y ~ x'
```

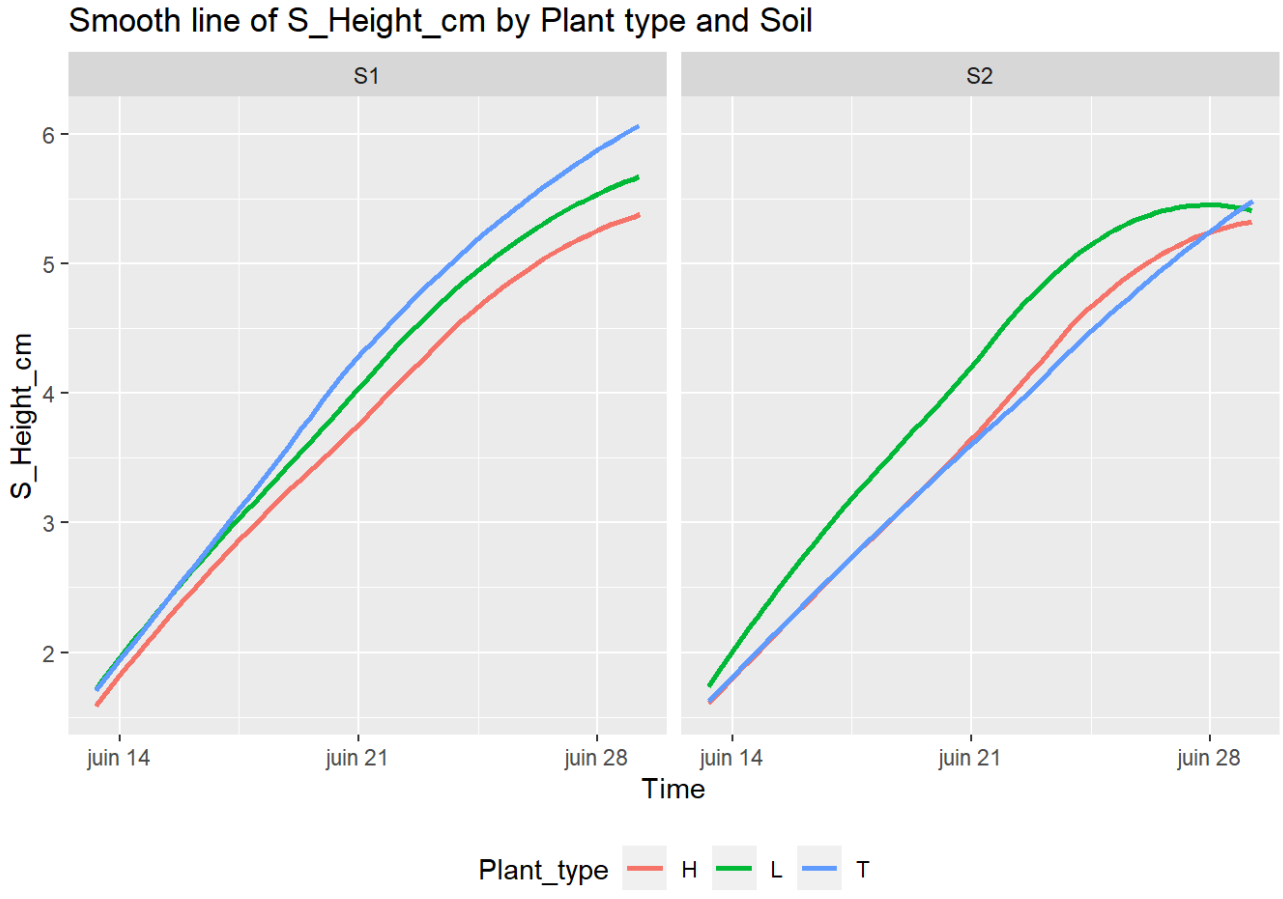


Scatter plots for all genotypes by water treatment

```
plot_scatter_with_smooth_water(S_timeseries, variables_S)
```




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



C. Exploration of the T_timeseries dataframe

Scatter plots by Genotype

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

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

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

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

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

Scatter plots for all genotypes by water treatment

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

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