les 6-5

2024-05-06

Logboek kiemgroenten

2024-05-07

Om alvast te kijken hoe we met de data om kunnen gaan, ga ik een basic dataframe maken van 1 van de planten. In deze dataframe komt alleen maar de groei van de plant en de oplossing zout te staan. Met deze data wordt een histogram gemaakt.

Dataset

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(pwr)
set.seed(0)
cress_data <- data.frame(species = rep(c("cress", "alfalfa"), each=400),
                         N = 1:200,
                         batch_n=rep(c(1, 2), each=200),
                         Oplossing=rep(c(0,1,2,3), each=50),
                         cm_groei=c(rnorm(n=50, 5, 1.5),rnorm(n=50, 8,1),rnorm(n=50, 3, 2),rnorm(n=50, 1,
                                    rnorm(n=50, 5.5, 1.6),rnorm(n=50, 7.6,1.2),rnorm(n=50, 2.8, 2.1),rnorm
                                    rnorm(n=50, 3, 1),rnorm(n=50, 4,0.8),rnorm(n=50, 2, 0.5),rnorm(n=50,
                                    rnorm(n=50, 3.2, 1.1),rnorm(n=50, 3.8,1.1),rnorm(n=50, 2.1, 0.4),rnorm

                         n_ontkieming=c(rbinom(50, 1, 0.6), rbinom(50, 1, 0.8), rbinom(50, 1, 0.4), rbinom
                                        rbinom(50, 1, 0.5), rbinom(50, 1, 0.7), rbinom(50, 1, 0.5), rbinom
                                        rbinom(50, 1, 0.7), rbinom(50, 1, 0.9), rbinom(50, 1, 0.3), rbinom
```

```
                                    rbinom(50, 1, 0.4), rbinom(50, 1, 0.8), rbinom(50, 1, 0.6), rbinom

                      n_blaadjes=c(rnorm(n=50, 3, 1),rnorm(n=50, 4, 1),rnorm(n=50, 2, 1),rnorm(n=50, 1
                                   rnorm(n=50, 3, 1),rnorm(n=50, 4, 1),rnorm(n=50, 2, 1),rnorm(n=50, 1
                                   rnorm(n=50, 3, 1),rnorm(n=50, 4, 1),rnorm(n=50, 2, 1),rnorm(n=50, 1
                                   rnorm(n=50, 3, 1),rnorm(n=50, 4, 1),rnorm(n=50, 2, 1),rnorm(n=50, 1
cress_data <- as.tibble(cress_data)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
head(cress_data)
```

```
## # A tibble: 6 x 7
##   species     N batch_n Oplossing cm_groei n_ontkieming n_blaadjes
##   <chr>   <int>   <dbl>     <dbl>    <dbl>        <int>      <dbl>
## 1 cress       1       1         0     6.89            1      2.04
## 2 cress       2       1         0     4.51            1      1.38
## 3 cress       3       1         0     6.99            1      3.82
## 4 cress       4       1         0     6.91            1      3.11
## 5 cress       5       1         0     5.62            0      3.76
## 6 cress       6       1         0     2.69            0      0.694
```

Hierboven is de dataset gegenereerd

Dataset zonder batch-sep

```
# Data genereren zonder batches
cress_new <- cress_data %>%
    select(-batch_n)

# Order de data zodat alle oplossingen bij elkaar staan
cress_new <- cress_new[order(cress_new$Oplossing),]
head(cress_new)
```

```
## # A tibble: 6 x 6
##   species     N Oplossing cm_groei n_ontkieming n_blaadjes
##   <chr>   <int>     <dbl>    <dbl>        <int>      <dbl>
## 1 cress       1         0     6.89            1      2.04
## 2 cress       2         0     4.51            1      1.38
## 3 cress       3         0     6.99            1      3.82
## 4 cress       4         0     6.91            1      3.11
## 5 cress       5         0     5.62            0      3.76
## 6 cress       6         0     2.69            0      0.694
```

```
# Gemiddelde generen voor groei in cm voor elke soort en oplossing
aggregate(cm_groei ~ species + Oplossing,data=cress_data, FUN=mean)
```

```
##   species Oplossing cm_groei
## 1 alfalfa         0 3.103501
## 2   cress         0 5.304877
## 3 alfalfa         1 3.835705
```

```
## 4   cress         1 7.863672
## 5 alfalfa         2 2.018409
## 6   cress         2 2.715076
## 7 alfalfa         3 1.097948
## 8   cress         3 1.124463
```

2024-05-08

Test data structuur

Omdat deze data gesimuleerd is, is dit overbodig. Het moet daarintgen wel in onze echte analyse. Dus verwerk ik dit hier nju

```
head(cress_data)
```

```
## # A tibble: 6 x 7
##   species     N batch_n Oplossing cm_groei n_ontkieming n_blaadjes
##   <chr>   <int>   <dbl>     <dbl>    <dbl>        <int>      <dbl>
## 1 cress       1       1         0     6.89            1      2.04
## 2 cress       2       1         0     4.51            1      1.38
## 3 cress       3       1         0     6.99            1      3.82
## 4 cress       4       1         0     6.91            1      3.11
## 5 cress       5       1         0     5.62            0      3.76
## 6 cress       6       1         0     2.69            0      0.694
```

De correcte datatypen komen voor in de data set, ook is de header correct.

Het volgende wat gecheckt moet worden is of het aantal observaties overeenkomt met de dataset

```
str(cress_data)
```

```
## tibble [800 x 7] (S3: tbl_df/tbl/data.frame)
##  $ species     : chr [1:800] "cress" "cress" "cress" "cress" ...
##  $ N           : int [1:800] 1 2 3 4 5 6 7 8 9 10 ...
##  $ batch_n     : num [1:800] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Oplossing   : num [1:800] 0 0 0 0 0 0 0 0 0 0 ...
##  $ cm_groei    : num [1:800] 6.89 4.51 6.99 6.91 5.62 ...
##  $ n_ontkieming: int [1:800] 1 1 1 1 0 0 1 1 1 1 ...
##  $ n_blaadjes  : num [1:800] 2.04 1.38 3.82 3.11 3.76 ...
```

800 observaties komt overeen met de 800 zaadjes die geteld en geplant zijn.

0,1,2,3 nieuwe concentraties

2024-05-09

- 5 liter water gekookt voor 5 minuten.
- 5, 10, 15 gram zout gewogen
- deze heb ik opgelost in 3x500 ml water
- dit zorgt voor 3 verschillende concentraties: 1,2,3% zout
- Deze oplossingen zijn in een eigen spuitfles gedaan.
- 500 ml water zonder opgeloste zout ook in een eigen spuitfles gedaan.
- 16 petrischaaltjes gelabeled met soort, batch n en % oplossing
- 1 keukenpapiertje per schaaltje 2x dubbel gevouwen en gespoten met de oplossing die bij het schaaltje hoort tot het vochtig is aan de bovenkant.

- keukenpapiertje in schaaltje gedaan en 50 zaadjes er over heen gestrooid.
- zaadjes gespoten met correcte oplossing water na het het strooien.

2024-05-10

- 8:40: papier was droog, dus heb elk bakje 2x bemist ### t-test test Een goede manier om voor elke oplossing een t test te doen tussen de 2 soorten

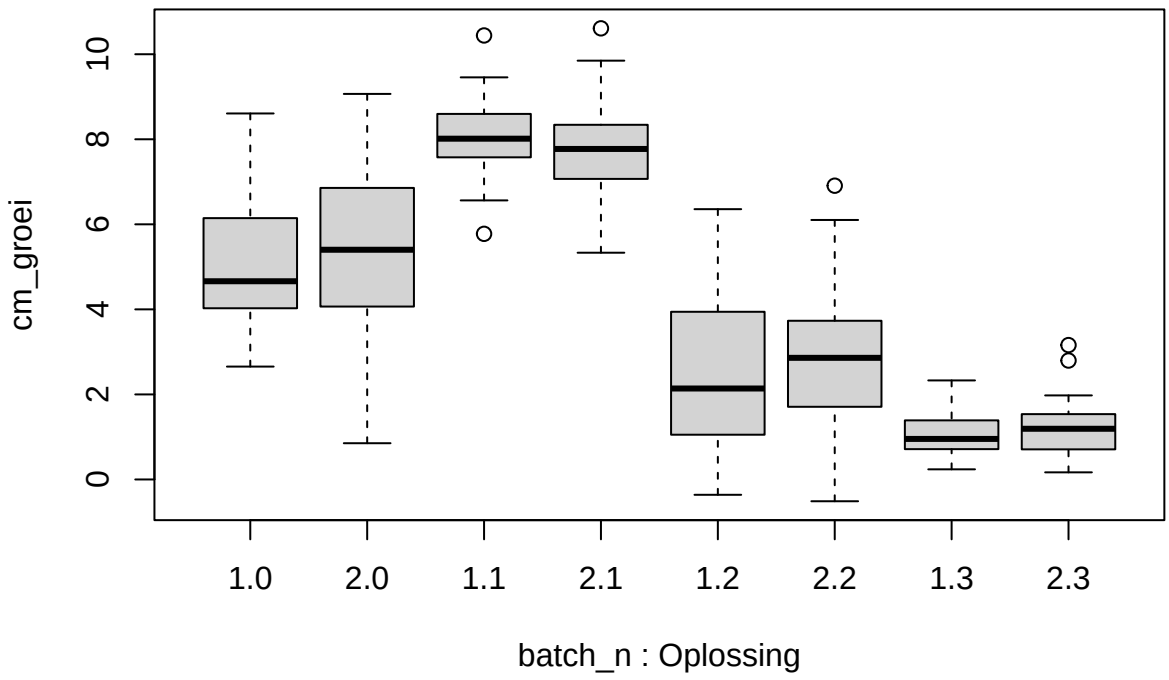
```
cress_data %>%
    filter(Oplossing == 0) %>%
    t.test(cm_groei ~ species, data=.)
```

```
##
##  Welch Two Sample t-test
##
## data:  cm_groei by species
## t = -11.251, df = 179.71, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group alfalfa and group cress is not equal t
## 95 percent confidence interval:
##  -2.587451 -1.815301
## sample estimates:
## mean in group alfalfa   mean in group cress
##              3.103501              5.304877
```

standaard boxplot showcase

Om de spreiding weer te geven tussen de 2 batches op elke oplossing van de cress kan je een boxplot gebruiken, zie hieronder

```
test <- cress_data %>%
    filter(species == "cress")
boxplot(cm_groei ~ batch_n + Oplossing, data=test)
```



2024-05-11

- Omdat het papier uitdroogt spuit ik elke ochtend en avond om 9 uur 2x in elk bakje. Ook laat ik het dekseltje met een klein spleetje open op het bakje, zodat het vocht in het bakje blijft

2024-05-12

Vanwege een fout tijdens het 1e experiment, wat heeft geleid tot extreme uitdroging hebben we besloten overnieuw te beginnen

- zelfde stappen als voorheen
- nu vanaf het begin het dekseltje op het bakje gelaten, zodat de bodem niet uitdroogt.

2024-05-13

- 10 uur in de ochtend 2 keer water gespoten op de zaadjes

sapply test

Er moet vast een manier zijn om een functie meerdere keren te herhalen, en die data op te slaan. Om dit te testen experimenteer ik met lapply, met het uitvoeren van meerdere t-testen

```
# Zo kan je voor elke oplossing kijken of er een significante afwijking is in verhouding met 0%
lijstje <- sapply(1:3, function(x){
    y <-cress_new %>%
        filter(Oplossing == 0 | Oplossing == x) %>%
        filter(species == "cress") %>%
```

```
        t.test(cm_groei ~ Oplossing, data = .)})

# Dit gebeurt er boven ^
x1 <- cress_data %>%
    filter(Oplossing == 0 | Oplossing == 1) %>%
    filter(species == "cress") %>%
    t.test(cm_groei ~ Oplossing, data = .)

x2 <- cress_data %>%
    filter(Oplossing == 0 | Oplossing == 2) %>%
    filter(species == "cress") %>%
    t.test(cm_groei ~ Oplossing, data = .)

x3 <- cress_data %>%
    filter(Oplossing == 0 | Oplossing == 3) %>%
    filter(species == "cress") %>%
    t.test(cm_groei ~ Oplossing, data = .)
```

Zo krijg je als output de t-test uitslagen van het verschil van cm_groei tussen oplossing van 0% en de overige oplossingen.

##2024-05-14 - Plantjes water gegeven om 9 uur.

Proportie

Vandaag hebben we geleerd hoe we om kunnen gaan met percentages, dit is relevant voor onze ontkieming.

```
data_set_ontkieming <- cress_data %>%
    select(n_ontkieming, species, Oplossing)
head(data_set_ontkieming)
```

```
## # A tibble: 6 x 3
##   n_ontkieming species Oplossing
##          <int> <chr>       <dbl>
## 1            1 cress           0
## 2            1 cress           0
## 3            1 cress           0
## 4            1 cress           0
## 5            0 cress           0
## 6            0 cress           0
```

```
ontkieming_table <- table(data_set_ontkieming$species, data_set_ontkieming$Oplossing+data_set_ontkiemin
(ontkieming_table)
```

```
##
##              0   1   2   3   4
##   alfalfa  48  67 139 112  34
##   cress    41  90 129 120  20
```

Nu kunnen we een proptest doen om te kijken of er verschil is tussen de 2 soorten planten

```
str(ontkieming_table)
```

```
##  'table' int [1:2, 1:5] 48 41 67 90 139 129 112 120 34 20
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2] "alfalfa" "cress"
##   ..$ : chr [1:5] "0" "1" "2" "3" ...
```

Om het verschil te kunnen zien tussen de 2 soorten en n ontkiemingen kan je een 2-way chisq-test uitvoeren:

```
(res <- chisq.test(x = ontkieming_table))
```

```
##
##  Pearson's Chi-squared test
##
## data:  ontkieming_table
## X-squared = 8.1986, df = 4, p-value = 0.08457
```

```
cat(sprintf("De p-waarde is groter dan 0.05: %.2f\n", res$p.value))
```

```
## De p-waarde is groter dan 0.05: 0.08
```

De p-waarde is hoger dan 0.05, dit houd in dat er geen significant verschil is tussen de 2 planten soorten. Om te weten of het een relevant verschil is, moet je de effectsterkte berekenen. Voor proportie gebruik ik Cohen's W

```
chi2 <- res$statistic
N <- sum(ontkieming_table)
W <- sqrt(chi2 / N)
df <- res$parameter
cat(sprintf("Cohens W: %.3f", W))
```

```
## Cohens W: 0.101
```

Cohens W ligt bij 0.10, er is dus een zwak verband tussen de invloed op aantal ontkiemde zaadjes tussen soort plant en zoutoplossing.

power

Nu we een conclusie hebben moeten we berekenen of die correct is of niet. Dit kan door de power te berekenen:

```
alpa <- 0.05
pwr.chisq.test(w=W, N=N, df=df, sig.level = alpa)
```

```
##
##      Chi squared power calculation
##
##               w = 0.1012337
##               N = 800
##              df = 4
##       sig.level = 0.05
##           power = 0.6168838
##
## NOTE: N is the number of observations
```

Hier is de power 0.61 Daarmee kunnen we zeggen dat onze conclusie, er is geen sigfnicant verschil gevonden van het aantal ontkiemde zaadjes tussen de 2 planten over de verschillende oplossingen, een 61% kans heeft om correct te zijn.

Hoeveel metingen hebben we nodig om 0.8 power te krijgen, met een W van 0.1?

```
pwr.chisq.test(w = 0.1, df = df, sig.level = alpa, power = 0.80)
```

```
##
##      Chi squared power calculation
##
##               w = 0.1
##               N = 1193.529
```

```
##              df = 4
##       sig.level = 0.05
##           power = 0.8
##
## NOTE: N is the number of observations
```

Dan zouden we dus 1194 planten moeten meten, zo'n 600 per planten soort.

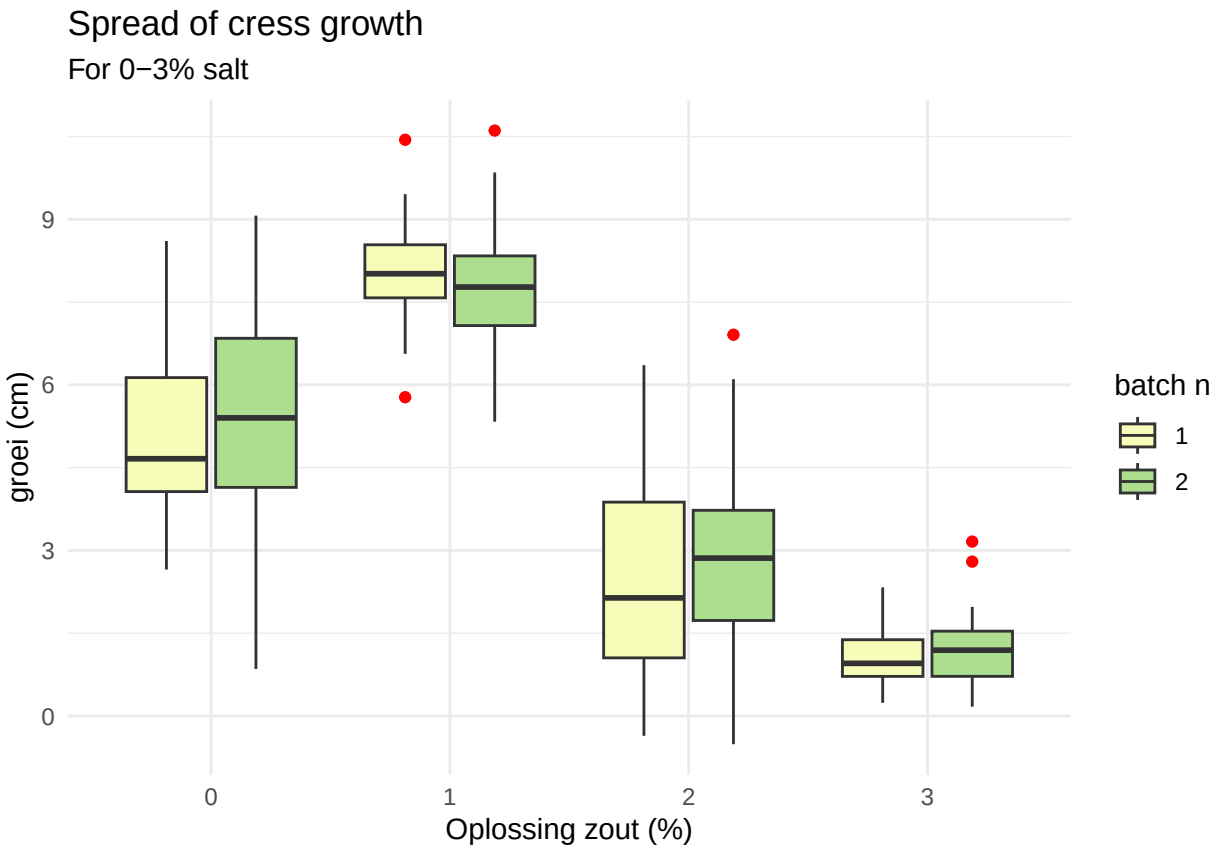
2024-05-15

- 7:30 planten water gegeven

ggplot2 plots

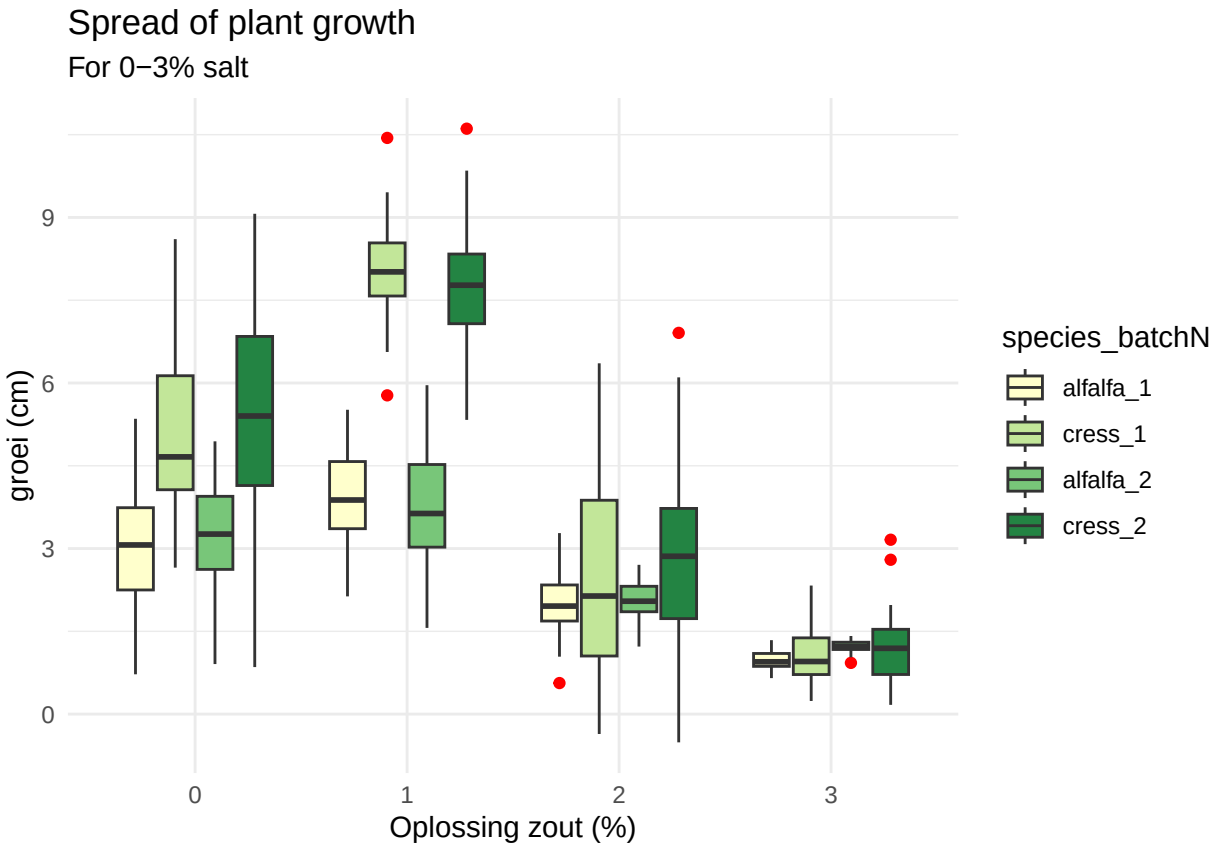
We hebben voor cm_groei een boxplot gemaakt via de basic boxplot functie van R. Graag zou ik alle plots willen maken via de ggplot library.

```
cress_data%>%
    filter(species == "cress") %>%
    ggplot(mapping = aes(x = as.character(Oplossing), y = cm_groei, fill = as.character(batch_n), ))+
    labs(y = "groei (cm)",
         x = "Oplossing zout (%)",
         title = "Spread of cress growth",
         subtitle = "For 0-3% salt",
         fill="batch n")+
    scale_fill_brewer(palette = "YlGn") +
    geom_boxplot(outlier.color = "red",) +
    theme_minimal()
```



Zo is de boxplot wat mooier en makkelijker af te lezen. Hetzelfde kan gedaan worden voor de andere planten soort

```
cress_data %>%
    mutate(species_batch = fct_cross(species, as.character(batch_n), sep = "_")) %>%
    ggplot(mapping = aes(x = as.character(Oplossing), y = cm_groei, fill = species_batch,)) +
        labs(y = "groei (cm)",
         x = "Oplossing zout (%)",
         title = "Spread of plant growth",
         subtitle = "For 0-3% salt",
         fill="species_batchN") +
    scale_fill_brewer(palette = "YlGn") +
    geom_boxplot(outlier.color = "red") +
    theme_minimal()
```



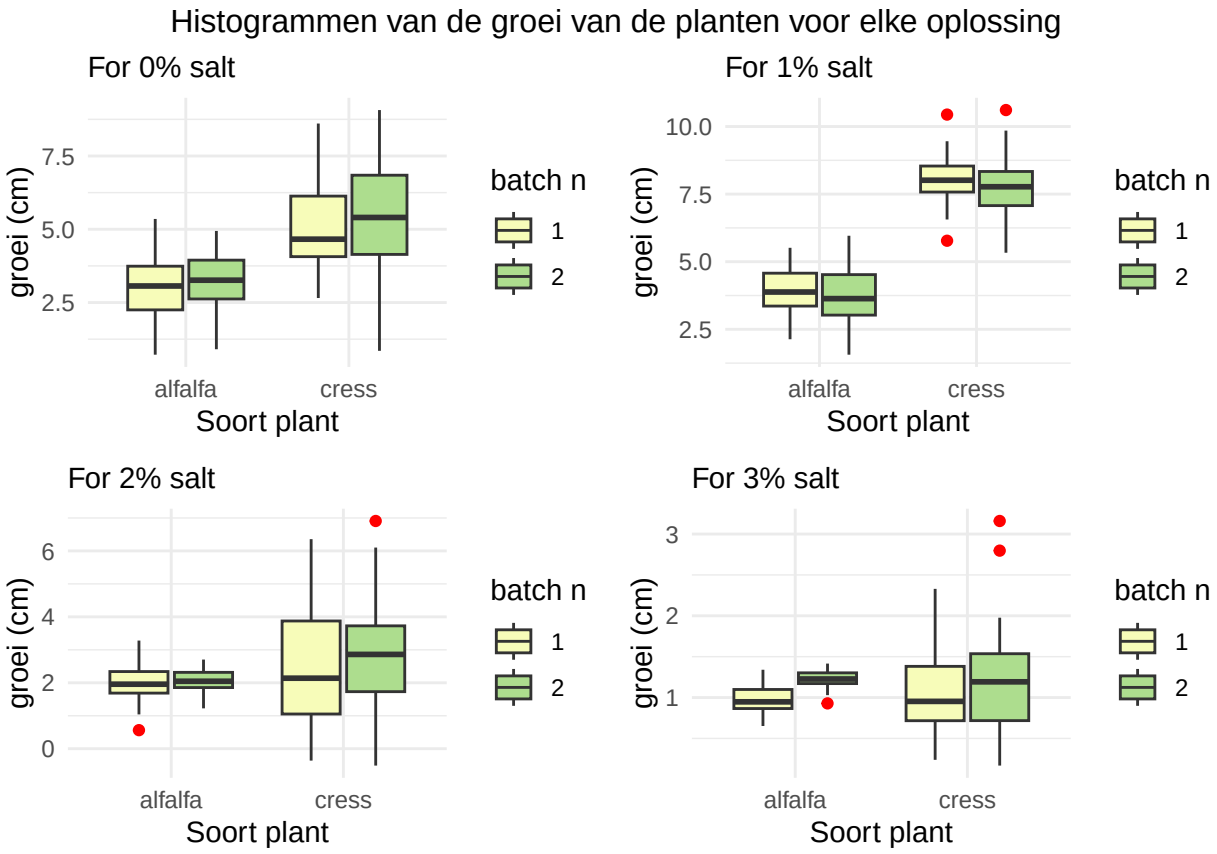
Dit is best onleesbaar, vooral oplossing 3. Een betere manier om dit te doen is dit te plotten is losse charts op 1 pagina. Dit kan met een library: gridExtra

```
plot_data_box <- function (data, variable_oplossing){
    data %>%
        filter(Oplossing == variable_oplossing) %>%
        ggplot(mapping = aes(x = species, y = cm_groei, fill=as.character(batch_n))) +
        labs(y = "groei (cm)",
         x = "Soort plant",
         subtitle = sprintf("For %i%% salt", variable_oplossing),
         fill="batch n") +
        scale_fill_brewer(palette = "YlGn") +
```

```
        geom_boxplot(outlier.color = "red")+
        theme_minimal()


}
```

Functie is gemaakt om makkelijk te plotten

```
p1 <- plot_data_box(cress_data, 0)
p2 <- plot_data_box(cress_data, 1)
p3 <- plot_data_box(cress_data, 2)
p4 <- plot_data_box(cress_data, 3)
grid.arrange(p1, p2, p3 ,p4, top = "Histogrammen van de groei van de planten voor elke oplossing")
```

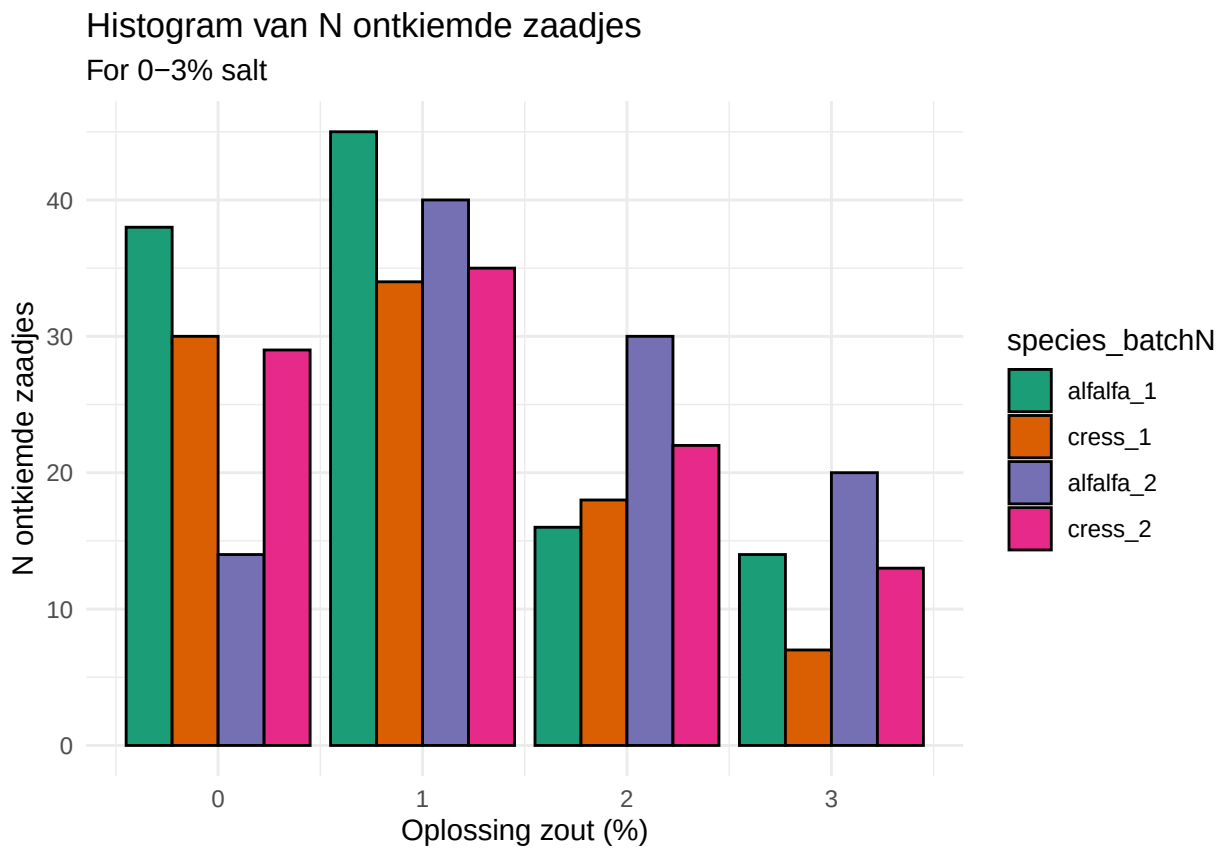


Deze page heeft 4 boxplotjes. Voor elke oplossing geeft het de spreiding van groei in cm weer, voor de soort en batch. De rode puntjes zijn de uitschieters.

Er kan op deze manier ook een histogram gemaakt worden die het aantal ontkiemde zaadjes bevat

```
cress_data %>%
    mutate(species_batch = fct_cross(species, as.character(batch_n), sep="_")) %>%
    filter(n_ontkieming == 1) %>%
    ggplot(mapping = aes(x = Oplossing, fill = species_batch),) +

    labs(y = "N ontkiemde zaadjes",
         x = "Oplossing zout (%)",
         title = "Histogram van N ontkiemde zaadjes",
         subtitle = "For 0-3% salt",
```

```
        fill="species_batchN") +

    scale_fill_brewer(palette = "Dark2") +

    geom_bar(position = "dodge", color = "black")+
    theme_minimal()
```



In deze grafiek zie je het aantal ontkiemde zaadjes per oplossing, voor alle batches en plant soorten.

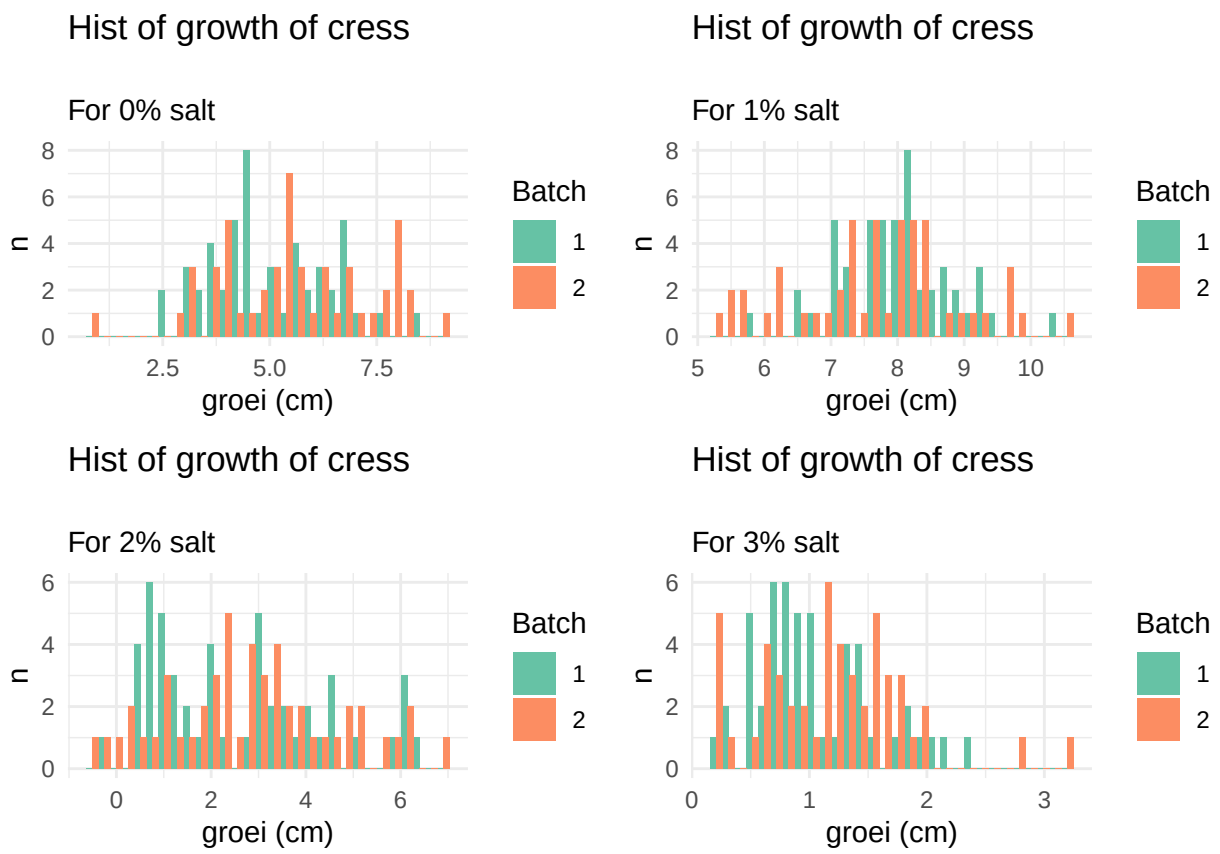
```
plot_data_hist <- function (data, variable_oplossing, v_species){
    data %>%
        filter(Oplossing == variable_oplossing & species == v_species) %>%
        ggplot(mapping = aes(x = cm_groei, fill=as.character(batch_n))) +
        labs(y = "n",
         x = "groei (cm)",
         title = sprintf("Hist of growth of %s\n", v_species),
         subtitle = sprintf("For %i%% salt", variable_oplossing),
         fill="Batch")+

        scale_fill_brewer(palette = "Set2")+

        geom_histogram(position = "dodge",)+
        theme_minimal()}
```

```
p1 <- plot_data_hist(cress_data, 0, "cress")
p2 <- plot_data_hist(cress_data, 1, "cress")
p3 <- plot_data_hist(cress_data, 2, "cress")
```

```
p4 <- plot_data_hist(cress_data, 3, "cress")
```

```
grid.arrange(p1, p2, p3 ,p4)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Hierboven een histogram van de tuinkers, deze histogram geeft de groei in cm weer voor elke zoutoplossing. Ook kan je het verschil zien tussen de 2 batches.

```
averages_data_frame <- aggregate(cbind(cm_groei, n_blaadjes) ~ species+batch_n+Oplossing, data = cress_
colnames(averages_data_frame) <- c("species", "batch_n", "Oplossing", "av_cm_groei", "av_n_blaadjes")
temp <- aggregate(cbind(cm_groei, n_blaadjes) ~ species+batch_n+Oplossing, data = cress_data, FUN = sd)

averages_data_frame$sd_cm_groei <- temp$cm_groei
averages_data_frame$sd_n_blaadjes <- temp$n_blaadjes
head(averages_data_frame)
```

```
##   species batch_n Oplossing av_cm_groei av_n_blaadjes sd_cm_groei sd_n_blaadjes
## 1 alfalfa       1         0    2.979804      3.087327   1.2132633      1.066368
## 2   cress       1         0    5.035896      3.010634   1.3744421      0.945506
## 3 alfalfa       2         0    3.227199      3.081805   1.0631705      1.066072
## 4   cress       2         0    5.573858      2.886641   1.7504073      1.004672
## 5 alfalfa       1         1    3.938455      4.000246   0.7917711      1.069972
## 6   cress       1         1    8.021406      4.016331   0.8569964      1.052283
```

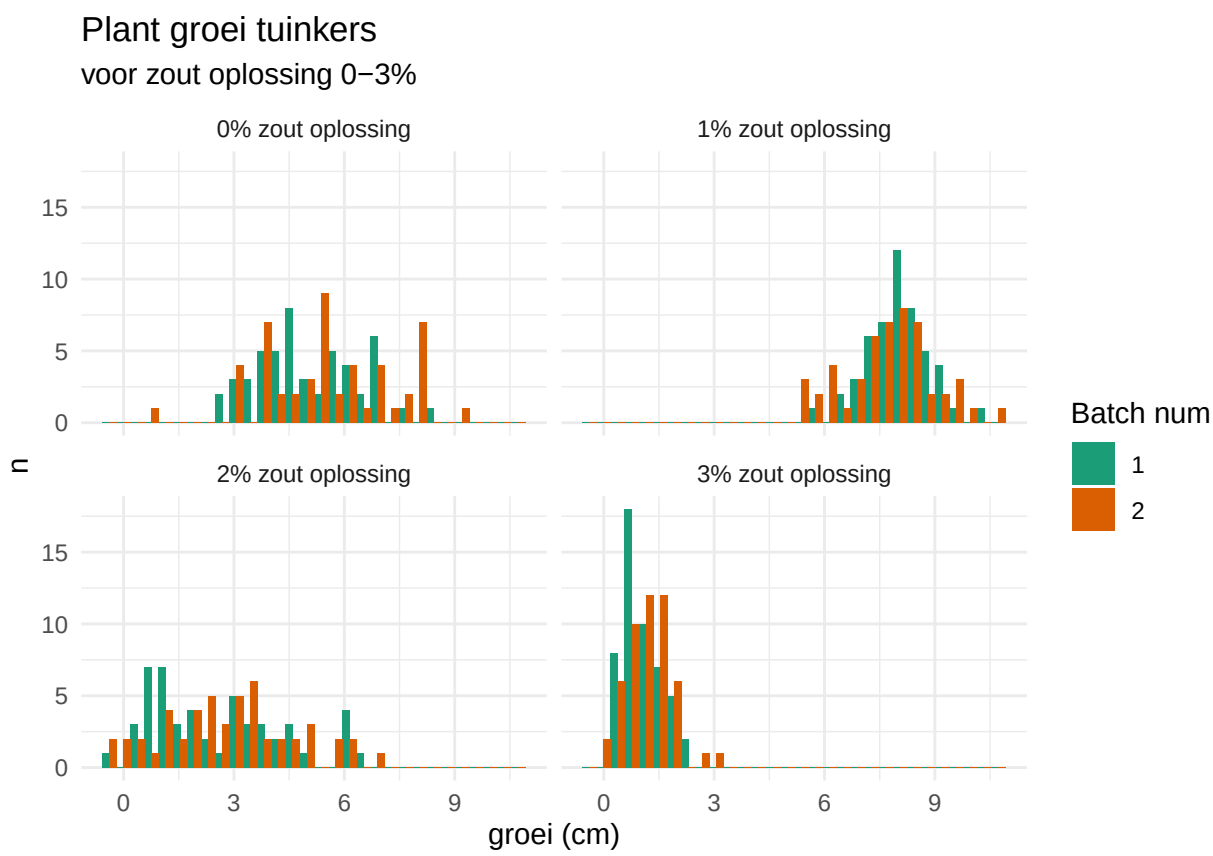
Zo krijg je een dataframe met de gemiddelden en standaard deviaties van cm_groei, en n_blaadjes.

2024-05-16

Facet-wrap gebruiken om makkelijker meerdere plotjes te plotten in 1 chart.

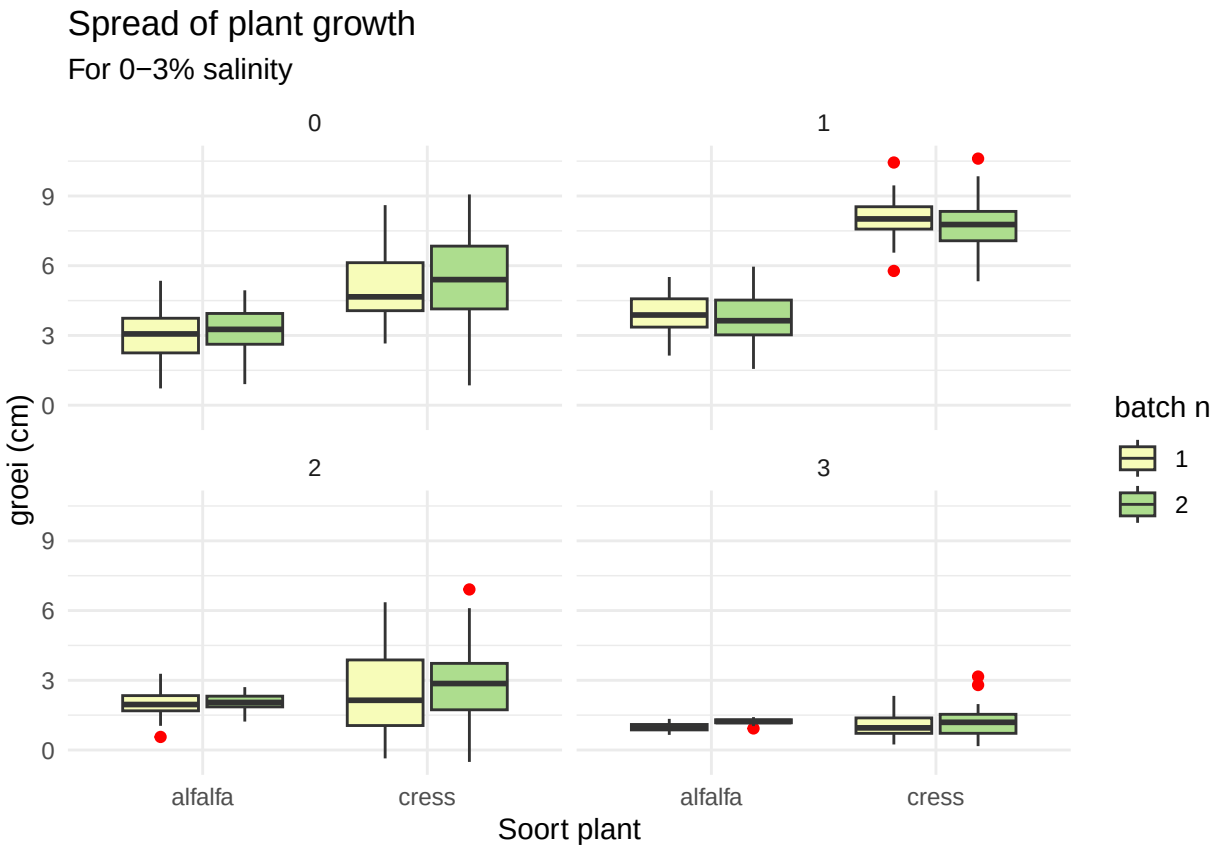
```
cress_data %>%
    filter(species == "cress") %>%
    ggplot(mapping = aes(x = cm_groei, fill=as.character(batch_n))) +
    labs(y = "n",
     x = "groei (cm)",
     title = sprintf("Plant groei tuinkers"),
     subtitle = sprintf("voor zout oplossing 0-3%%"),
     fill="Batch num")+

    scale_fill_brewer(palette = "Dark2")+

    geom_histogram(position = "dodge",) +
    facet_wrap(~ Oplossing,
               labeller = labeller(Oplossing = c("0" = "0% zout oplossing",
                                                 "1" = "1% zout oplossing",
                                                 "2" = "2% zout oplossing",
                                                 "3" = "3% zout oplossing")))+
    theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Hetzelfde kan gedaan worden met de eerder gemaakte boxplotjes

```
cress_data %>%
    ggplot(mapping = aes(x = species, y = cm_groei, fill=as.character(batch_n))) +
    labs(y = "groei (cm)",
         x = "Soort plant",
         title = "Spread of plant growth",
         subtitle = "For 0-3% salinity",
         fill="batch n") +
    scale_fill_brewer(palette = "YlGn") +
    geom_boxplot(outlier.colour = "red") +
    theme_minimal()+
    facet_wrap(~ Oplossing)
```



Wat de library gridExtra in dit geval beter doet dan facet_wrap is dat bij deze plot de y as allemaal dezelfde waarde heeft. Bij gridExtra is dit niet het geval.