

Data Preparation: Maize Project

Dr. Luisa Barbanti & Nisia Trisconi | Zurich Data Scientists

October 30, 2023

Contents

| | | |
|----------|--|----------|
| 1 | Freezing Package versions | 3 |
| 2 | Load packages | 3 |
| 3 | Settings | 3 |
| 4 | Getting data | 4 |
| 5 | Remove empty columns/rows | 4 |
| 6 | Overview | 4 |
| 7 | Check data | 5 |
| 7.1 | Check <i>pot</i> | 5 |
| 7.2 | Create <i>pot.fac</i> (new) | 5 |
| 7.3 | Check <i>soil</i> | 6 |
| 7.4 | Create <i>soil.fac</i> (new) | 7 |
| 7.5 | Check <i>well</i> | 7 |
| 7.6 | Create <i>well.fac</i> (new) | 8 |
| 7.7 | Check <i>depth</i> | 9 |
| 7.8 | Check <i>seed.weight</i> | 10 |
| 7.9 | Create <i>seed.weight.grams</i> (new) | 11 |
| 7.10 | Check <i>fungus</i> | 11 |
| 7.11 | Create <i>fungus.fac</i> (new) | 12 |
| 7.12 | Check <i>date.germinated</i> | 12 |
| 7.13 | Create <i>date.germinated.asDate</i> (new) | 13 |
| 7.14 | Check <i>observations</i> | 14 |
| 7.15 | Create <i>obs.time</i> (new) | 15 |
| 7.16 | Create <i>broken</i> (new) | 15 |
| 7.17 | Check <i>height_2022_07_05</i> | 16 |
| 7.18 | Create <i>height_2022_07_05.num</i> (new) | 16 |
| 7.19 | Create <i>plant.found</i> (new) | 19 |
| 7.20 | Check <i>cob_weight</i> | 20 |
| 7.21 | Create <i>cob_weight.num</i> (new) | 20 |
| 7.22 | Check <i>...12</i> | 23 |
| 7.23 | Create <i>germinated.in.lab</i> (new) | 23 |
| 7.24 | Create <i>germinated.in.field</i> (new) | 25 |
| 7.25 | Create <i>germinated.yes</i> (new) | 26 |
| 7.26 | Create <i>days.to.germination</i> (new) | 27 |
| 7.27 | Create <i>days.to.germination.censored</i> (new) | 28 |
| 7.28 | Create <i>seed_coord_y</i> (new) | 29 |

| | | |
|-----------|--|-----------|
| 7.29 | Create <i>seed_coord_x</i> (new) | 31 |
| 7.30 | Create <i>position_field_x</i> (new) | 33 |
| 7.31 | Create <i>position_field_y</i> (new) | 38 |
| 8 | Missing values | 40 |
| 9 | Creating the RDS file | 41 |
| 10 | Session information | 42 |

1 Freezing Package versions

```
## (messages are omitted in this chunk)
##
library(checkpoint)
checkpoint(snapshot_date = "2022-11-15")
```

2 Load packages

```
## (messages are omitted from this chunk)
##
library(dplyr)
library(kableExtra)
library(ggplot2)
library(tidyr)
library(readxl)
library(magrittr)
library(survival)
```

3 Settings

Global settings:

```
theme_set(theme_bw())

if (!dir.exists("Prepared_data_and_models")) {
  dir.create("Prepared_data_and_models")
}
```

4 Getting data

```
d.maize <- read_excel(path = paste0("../..../Original_data/",
                                     "3_cob_weight_filled.xlsx"),
                     sheet = "measurements")
```

New names:

```
* `` -> `...12`
```

5 Remove empty columns/rows

We remove columns that contain only missing values. We use the argument `drop = FALSE` to ensure that we still obtain a `data.frame` result.

```
## Remove empty columns
```

```
## The following columns are empty
```

```
empty.cols <- apply(X = is.na(d.maize), MARGIN = 2, FUN = all)
colnames(d.maize[, empty.cols, drop = FALSE])
```

```
[1] "orientation"
```

```
## Remove
```

```
d.maize <- d.maize[, ! empty.cols, drop = FALSE]
```

6 Overview

```
dim(d.maize)
```

```
[1] 108 11
```

```
head(d.maize)[1:ncol(d.maize)]
```

```
# A tibble: 6 x 11
```

| | pot | soil | well | depth | seed.weight | fungus | date.germinated | observations |
|---|-------|--------------|-------|-------|-------------|--------|-----------------|--------------|
| | <chr> | <chr> | <chr> | <dbl> | <dbl> | <chr> | <chr> | <chr> |
| 1 | A1 | Bio garden a | | 3 | 30 | <NA> | 2022-05-11 | <NA> |
| 2 | A1 | Bio garden b | | 5 | 34 | <NA> | 2022-05-11 | <NA> |
| 3 | A1 | Bio garden c | | 2 | 35 | <NA> | 2022-05-09 | <NA> |
| 4 | A1 | Bio garden d | | 1 | 40 | <NA> | 2022-05-10 | <NA> |
| 5 | A1 | Bio garden e | | 4 | 46 | <NA> | 2022-05-11 | <NA> |
| 6 | A1 | Bio garden f | | 6 | 37 | <NA> | 2022-05-11 | <NA> |

```
# i 3 more variables: height_2022_07_05 <chr>, cob_weight <chr>, ...12 <dbl>
```

```
str(d.maize)
```

```
tibble [108 x 11] (S3: tbl_df/tbl/data.frame)
```

```
$ pot           : chr [1:108] "A1" "A1" "A1" "A1" ...
$ soil          : chr [1:108] "Bio garden" "Bio garden" "Bio garden" "Bio garden" ...
$ well          : chr [1:108] "a" "b" "c" "d" ...
$ depth         : num [1:108] 3 5 2 1 4 6 6 4 5 1 ...
$ seed.weight   : num [1:108] 30 34 35 40 46 37 27 16 23 22 ...
$ fungus        : chr [1:108] NA NA NA NA ...
$ date.germinated : chr [1:108] "2022-05-11" "2022-05-11" "2022-05-09" "2022-05-10" ...
$ observations   : chr [1:108] NA NA NA NA ...
```

```
$ height_2022_07_05: chr [1:108] "217" "131" "143" "194" ...
$ cob_weight       : chr [1:108] "117" "26" "61" "109" ...
$ ...12           : num [1:108] NA NA NA NA NA NA NA NA NA NA ...
```

7 Check data

We first create ‘proto-plot’ objects.

```
gg.density <- ggplot(data = d.maize) +
  guides(alpha = "none") +
  geom_density() +
  geom_rug(alpha = 0.3)
```

7.1 Check *pot*

```
## class
class(d.maize$`pot`)
```

```
[1] "character"
```

```
## overview missing values
```

```
is.na(d.maize$`pot`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
.
TRUE FALSE
0    108
```

```
## number of levels (without missing values)
n_distinct(d.maize$`pot`, na.rm = TRUE)
```

```
[1] 18
```

```
## show the first 10 non-missing values in decreasing order
table(d.maize$`pot`, useNA = "no")
```

```
A1 A2 A3 A4 A5 A6 B1 B2 B3 B4 B5 B6 C1 C2 C3 C4 C5 C6
6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
```

The pots are arranged as follows:

```
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] "C1" "C2" "C3" "C4" "C5" "C6"
[2,] "B1" "B2" "B3" "B4" "B5" "B6"
[3,] "A1" "A2" "A3" "A4" "A5" "A6"
```

7.2 Create *pot.fac* (new)

We change *pot* to a factor and save the new variable in *pot.fac*. We check that levels are mapped correctly.

```
## change pot to factor
d.maize %<>%
  mutate(pot.fac = as.factor(`pot`))
```

```
## check
class(d.maize$`pot.fac`)

[1] "factor"

## levels mapped correctly
d.maize %>%
  select(contains("pot")) %>%
  unique()
```

```
# A tibble: 18 x 2
```

```
  pot pot.fac
  <chr> <fct>
1 A1 A1
2 A2 A2
3 A3 A3
4 A4 A4
5 A5 A5
6 A6 A6
7 B1 B1
8 B2 B2
9 B3 B3
10 B4 B4
11 B5 B5
12 B6 B6
13 C1 C1
14 C2 C2
15 C3 C3
16 C4 C4
17 C5 C5
18 C6 C6
```

```
## number of levels (without missing values)
n_distinct(d.maize$`pot.fac`, na.rm = TRUE)
```

```
[1] 18
```

```
## show how many observations are in each level of pot.fac
table(d.maize$`pot.fac`, useNA = "no")
```

```
A1 A2 A3 A4 A5 A6 B1 B2 B3 B4 B5 B6 C1 C2 C3 C4 C5 C6
 6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
```

7.3 Check soil

We check the soil variable together with instances of missing entries.

```
## class
class(d.maize$`soil`)

[1] "character"

## overview missing values
is.na(d.maize$`soil`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
TRUE FALSE
0 108
```

```
## number of levels (without missing values)
n_distinct(d.maize$`soil`, na.rm = TRUE)
```

```
[1] 4
```

```
## show all non-missing values in decreasing order
table(d.maize$`soil`, useNA = "no") %>%
  sort(decreasing = TRUE)
```

```
herbs Bio garden Composana mixture
36      24      24      24
```

7.4 Create *soil.fac* (new)

Since the *soil* variable is taken as a character, we convert it to a factor and save it in the new variable *soil.fac*. We check that the mapping to a factor happened correctly.

```
d.maize %<>%
  mutate(soil.fac = as.factor(`soil`))
## check
class(d.maize$`soil.fac`)
```

```
[1] "factor"
```

```
## levels mapped correctly
d.maize %>%
  select(contains("soil")) %>%
  unique()
```

```
# A tibble: 4 x 2
  soil      soil.fac
  <chr>    <fct>
1 Bio garden Bio garden
2 herbs     herbs
3 mixture   mixture
4 Composana Composana
```

```
## show all non-missing values
table(d.maize$`soil.fac`, useNA = "no")
```

```
Bio garden Composana herbs mixture
24      24      36      24
```

When we inspect how many observations are recorded in each level we see that the “herbs” soil was used for $36/6 = 6$ pots (divide by 6 since every pot has 6 wells), while all other soil qualities were used for $24/6 = 4$ pots.

7.5 Check *well*

We inspect the *well* variable and notice that it is saved as character. We inspect instances of missing values.

```
## class
class(d.maize$`well`)

[1] "character"

## overview missing values
is.na(d.maize$`well`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()

.
TRUE FALSE
  0    108

## number of levels (without missing values)
n_distinct(d.maize$`well`, na.rm = TRUE)

[1] 6

## show all non-missing values in decreasing order
table(d.maize$`well`, useNA = "no")

 a  b  c  d  e  f
18 18 18 18 18 18
```

7.6 Create *well.fac* (new)

We convert the variable `well` to a factor and check that the mapping is correct. We have a total of 18 pots, and each pot has all 6 wells a, b, c, d, e, f, as shown below.

```
## change well to factor
d.maize %<>%
  mutate(well.fac = as.factor(`well`))
## check
class(d.maize$`well.fac`)

[1] "factor"

## levels mapped correctly
d.maize %>%
  select(contains("well")) %>%
  unique()

# A tibble: 6 x 2
  well well.fac
  <chr> <fct>
1 a     a
2 b     b
3 c     c
4 d     d
5 e     e
6 f     f

## number of levels (without missing values)
n_distinct(d.maize$`well.fac`, na.rm = TRUE)

[1] 6
```

```
## show all non-missing values
table(d.maize$`well.fac`, useNA = "no")
```

```
  a  b  c  d  e  f
18 18 18 18 18 18
```

Each pot is then divided in 6 wells and has the following layout:

```
      [,1] [,2]
[1,] "e"  "f"
[2,] "c"  "d"
[3,] "a"  "b"
```

7.7 Check *depth*

This is the depth at which the seed has been planted in centimeters. The maximum depth of 6cm corresponds to the seed being stuck into the coconut support. We can see that for each depth we have the same amount of observations.

```
class(d.maize$`depth`)
```

```
[1] "numeric"
```

```
summary(d.maize$`depth`)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.0    2.0    3.5    3.5    5.0    6.0
```

```
table(d.maize$`depth`)
```

```
 1  2  3  4  5  6
18 18 18 18 18 18
```

By design, all depth values are present in each pot but are allocated randomly to the wells.

```
d.maize %>%
  filter(pot.fac == "A1") %>%
  select(pot.fac, well.fac, depth)
```

```
# A tibble: 6 x 3
  pot.fac well.fac depth
  <fct>    <fct>    <dbl>
1 A1      a         3
2 A1      b         5
3 A1      c         2
4 A1      d         1
5 A1      e         4
6 A1      f         6
```

```
d.maize %>%
  filter(pot.fac == "B2") %>%
  select(pot.fac, well.fac, depth)
```

```
# A tibble: 6 x 3
  pot.fac well.fac depth
  <fct>    <fct>    <dbl>
```

| | | | |
|---|----|---|---|
| 1 | B2 | a | 3 |
| 2 | B2 | b | 6 |
| 3 | B2 | c | 2 |
| 4 | B2 | d | 1 |
| 5 | B2 | e | 4 |
| 6 | B2 | f | 5 |

7.8 Check *seed.weight*

We inspect `seed.weight` and its distribution. Note that that this variable is recorded in cg (centigrams).

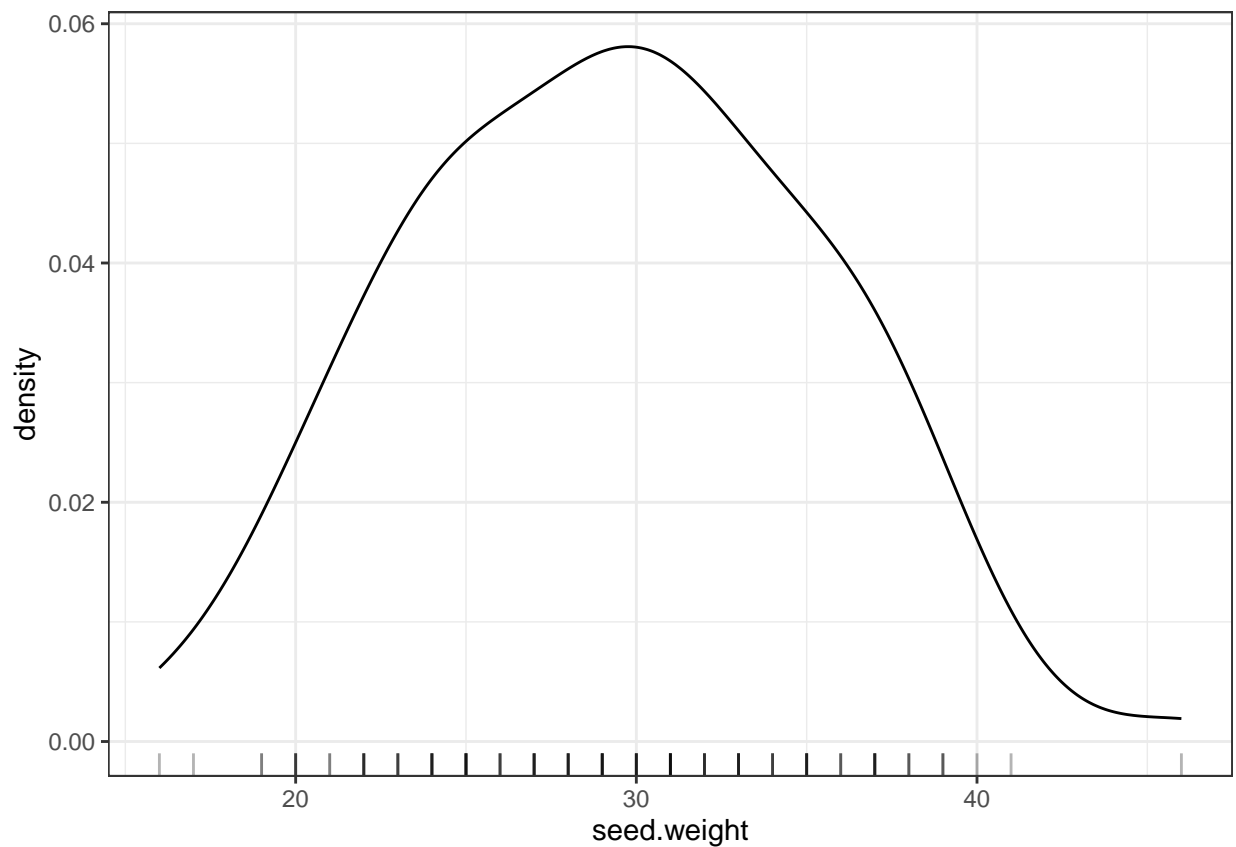
```
class(d.maize$`seed.weight`)
```

```
[1] "numeric"
```

```
summary(d.maize$`seed.weight`)
```

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|------|---------|--------|------|---------|------|
| 16 | 25 | 29 | 29 | 34 | 46 |

```
gg.density +  
aes(x = `seed.weight`, alpha = 0.3)
```



7.9 Create *seed.weight.grams* (new)

We scale the previous quantity to the grams scale by dividing by 100. We select 10 coordinates at random and check that for these coordinates, the conversion happened correctly. The randomly set coordinates depend on a seed which is set in the following code chunk.

The correct conversion rate is obtained by dividing the values by 100. Consequently, this division is the next step.

```
d.maize %<>%
  mutate(seed.weight.grams = 0.01 * `seed.weight`)
## check generalities
class(d.maize$`seed.weight.grams`)

[1] "numeric"

summary(d.maize$`seed.weight.grams`)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.16  0.25   0.29   0.29  0.34   0.46

## levels mapped correctly for some randomly picked observations
## with the following row numbers:
set.seed(1)
(sel <- sample(1:nrow(d.maize), 10))

[1] 68 39  1 34 87 43 14 82 59 51

(d.maize %>%
  select(contains("seed")))[sel,]

# A tibble: 10 x 2
   seed.weight seed.weight.grams
   <dbl>         <dbl>
1      23         0.23
2      31         0.31
3      30         0.3
4      33         0.33
5      26         0.26
6      22         0.22
7      31         0.31
8      26         0.26
9      29         0.29
10     29         0.29
```

7.10 Check *fungus*

The *fungus* variable is initially saved as a character.

```
## class
class(d.maize$`fungus`)

[1] "character"

## overview missing values
is.na(d.maize$`fungus`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```

TRUE FALSE
107      1

## number of levels (without missing values)
n_distinct(d.maize$`fungus`, na.rm = TRUE)

[1] 1

## show all non-missing values in decreasing order
sort(table(d.maize$`fungus`, useNA = "no"), decreasing = TRUE)

yes
1

```

7.11 Create *fungus.fac* (new)

We convert the `fungus` variable to a factor. Since it is only given for one observation, and missing for the rest, we assign "yes" as the value of `fungus.fac` for this one observation and to "no" for the observations with a missing value. We are left with no missing values for this new variable.

```

## replace missing values with "no" and convert to factor
d.maize %<>%
  mutate(fungus.fac = replace_na(fungus, "no") %>%
    as.factor())
## overview missing values
is.na(d.maize$`fungus.fac`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()

```

```

TRUE FALSE
0      108

## number of levels (without missing values)
n_distinct(d.maize$`fungus.fac`, na.rm = TRUE)

[1] 2

## show all non-missing values in decreasing order
table(d.maize$`fungus.fac`, useNA = "no")

no yes
107   1

```

7.12 Check *date.germinated*

The `date.germinated` variable is saved as a character and is missing for some observations.

```

## class
class(d.maize$`date.germinated`)

[1] "character"

## overview missing values
is.na(d.maize$`date.germinated`) %>%

```

```

factor(levels = c(TRUE, FALSE)) %>%
table()

.
TRUE FALSE
  24    84

## number of levels (without missing values)
n_distinct(d.maize$date.germinated, na.rm = TRUE)

[1] 8

## show all non-missing values in decreasing order
table(d.maize$date.germinated, useNA = "no")

2022-05-07 2022-05-08 2022-05-09 2022-05-10 2022-05-11 2022-05-12 2022-05-13
          4          1          6          17          29          13          10
2022-05-14
          4

```

7.13 Create *date.germinated.asDate* (new)

We convert `date.germinated` to a date format and save it into a new variable that we call `date.germinated.asDate`. Later, we check that for 10 randomly selected rows of our dataset, the mapping happened as expected. Some rows contain NA as value for `date.germinated`, meaning that the seed did not germinate in the time span between the seeding in pot and the moment when the well was planted in the field. This remains the same for `date.germinated.asDate`.

```

d.maize %<>%
  mutate(date.germinated.asDate = as.Date(`date.germinated`))
## check class
class(d.maize$date.germinated.asDate)

[1] "Date"

## levels mapped correctly for some randomly picked observations
## with the following row numbers:
(sel <- sample(1:nrow(d.maize), 10))

[1] 97 85 21 54 74 7 73 79 107 37

(d.maize %>%
  select(contains("date.germinated")))[sel,]

# A tibble: 10 x 2
  date.germinated date.germinated.asDate
  <chr>           <date>
1 2022-05-11      2022-05-11
2 2022-05-10      2022-05-10
3 2022-05-07      2022-05-07
4 2022-05-11      2022-05-11
5 <NA>            NA
6 2022-05-11      2022-05-11
7 2022-05-10      2022-05-10
8 <NA>            NA
9 2022-05-11      2022-05-11

```

```

10 <NA>          NA
## number of levels (without missing values)
n_distinct(d.maize$date.germinated.asDate`, na.rm = TRUE)

[1] 8
## show all non-missing values in decreasing order
table(d.maize$date.germinated.asDate`, useNA = "no")

2022-05-07 2022-05-08 2022-05-09 2022-05-10 2022-05-11 2022-05-12 2022-05-13
         4         1         6         17         29         13         10
2022-05-14
         4

```

The transferring of the seeds to the field happened on the day after we have the last germination date. The last germination date is:

```
max(d.maize$date.germinated.asDate, na.rm = TRUE)
```

```
[1] "2022-05-14"
```

That is, the seeds were transferred on

```
max(d.maize$date.germinated.asDate, na.rm = TRUE) + 1
```

```
[1] "2022-05-15"
```

7.14 Check *observations*

We check the `observations` column, where some comments were recorded. Most of the entries are empty, the rest we'll use next on to create some new variables.

```

## class
class(d.maize$observations`)

[1] "character"
## overview missing values
is.na(d.maize$observations`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()

.
TRUE FALSE
101      7
## number of levels (without missing values)
n_distinct(d.maize$observations`, na.rm = TRUE)

[1] 3
## show all non-missing values
table(d.maize$observations`, useNA = "no")

```

```

Broken Broken, morning measurement
      3                          1
morning measurement
      3

```

7.15 Create *obs.time* (new)

Usually, observations about germination date in lab are collected at night. However, from the `observations` column we can see that this has not always been the case. Hence we create a new variable telling us when the observation was collected. It has two levels, `morning` and `night`.

```
obs.time <- rep("night", nrow(d.maize))
obs.time[grepl("morning", d.maize$observations, ignore.case = TRUE)] <- "morning"
d.maize$obs.time <- factor(obs.time, levels = c("morning", "night"))
## class
class(d.maize$`obs.time`)
```

```
[1] "factor"
```

```
head(d.maize$`obs.time`)
```

```
[1] night night night night night night
```

```
Levels: morning night
```

```
## overview missing values
is.na(d.maize$`obs.time`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
.
TRUE FALSE
  0    108
```

```
## show all non-missing values
table(d.maize$`obs.time`, useNA = "no")
```

```
morning  night
      4     104
```

7.16 Create *broken* (new)

In the `observations` column we also have information telling us whether the seed was broken when it was put into the soil. We hence create a special variable for this.

```
broken <- rep(FALSE, nrow(d.maize))
broken[grepl("broken", d.maize$observations, ignore.case = TRUE)] <- TRUE
d.maize$broken <- broken
## class
class(d.maize$`broken`)
```

```
[1] "logical"
```

```
## overview missing values
is.na(d.maize$`broken`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
.
TRUE FALSE
  0    108
```

```
## show all non-missing values
table(d.maize$`broken`, useNA = "no")
```

```
FALSE TRUE
104    4
```

7.17 Check *height_2022_07_05*

The height of the maize plant is saved as a character. We see that some values are missing, meaning that the plant was not there and hence could not be measured, while other values are **not measured**, meaning that there was not enough time to measure the plant.

```
## class
class(d.maize$`height_2022_07_05`)
```

```
[1] "character"
```

```
## overview missing values
is.na(d.maize$`height_2022_07_05`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
.
TRUE FALSE
12      96
```

```
## number of levels (without missing values)
n_distinct(d.maize$`height_2022_07_05`, na.rm = TRUE)
```

```
[1] 48
```

```
## show the first 10 non-missing values in decreasing order
table(d.maize$`height_2022_07_05`, useNA = "no")
```

| | | | | | |
|-----|-----|-----|-----|-----------------|-----|
| 102 | 122 | 125 | 131 | 143 | 150 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 158 | 177 | 191 | 193 | 194 | 195 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 200 | 206 | 210 | 217 | 223 | 228 |
| 1 | 1 | 1 | 2 | 1 | 2 |
| 229 | 231 | 232 | 233 | 235 | 236 |
| 1 | 1 | 1 | 3 | 3 | 1 |
| 239 | 240 | 241 | 242 | 244 | 245 |
| 1 | 2 | 2 | 2 | 2 | 1 |
| 248 | 250 | 251 | 252 | 253 | 255 |
| 1 | 2 | 1 | 1 | 1 | 2 |
| 257 | 258 | 260 | 261 | 263 | 265 |
| 1 | 1 | 1 | 1 | 1 | 2 |
| 270 | 281 | 282 | 66 | 81 not measured | |
| 1 | 1 | 1 | 1 | 1 | 36 |

We will use this information to create a variable indicating whether there was a plant at all, independently of whether its height was measured.

7.18 Create *height_2022_07_05.num* (new)

We convert *height_2022_07_05* to a numeric variable and save it. In this variable, we coerce **not measured** to NA, indicating that the measurement was missing (this is also signalled in the warning). We check the

success of this mapping on a randomly selected subset of our data.

```
d.maize %<>%  
  mutate(height_2022_07_05.num = as.numeric(`height_2022_07_05`))
```

Warning: There was 1 warning in `mutate()`.

i In argument: `height_2022_07_05.num = as.numeric(height_2022_07_05)`.

Caused by warning:

! NAs introduced by coercion

```
## overview missing values
```

```
is.na(d.maize$`height_2022_07_05.num`) %>%  
  factor(levels = c(TRUE, FALSE)) %>%  
  table()
```

```
.  
TRUE FALSE  
48      60
```

```
## values mapped correctly for some randomly picked observations
```

```
## with the following row numbers:
```

```
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 105  89 101  37  34 107  44  79  33  84
```

```
(d.maize %>%  
  select(contains("height")))[sel,]
```

```
# A tibble: 10 x 2
```

| | height_2022_07_05 | height_2022_07_05.num |
|----|-------------------|-----------------------|
| | <chr> | <dbl> |
| 1 | not measured | NA |
| 2 | not measured | NA |
| 3 | not measured | NA |
| 4 | 250 | 250 |
| 5 | 66 | 66 |
| 6 | not measured | NA |
| 7 | <NA> | NA |
| 8 | not measured | NA |
| 9 | 235 | 235 |
| 10 | not measured | NA |

```
## number of levels (without missing values)
```

```
n_distinct(d.maize$`height_2022_07_05.num`, na.rm = TRUE)
```

```
[1] 47
```

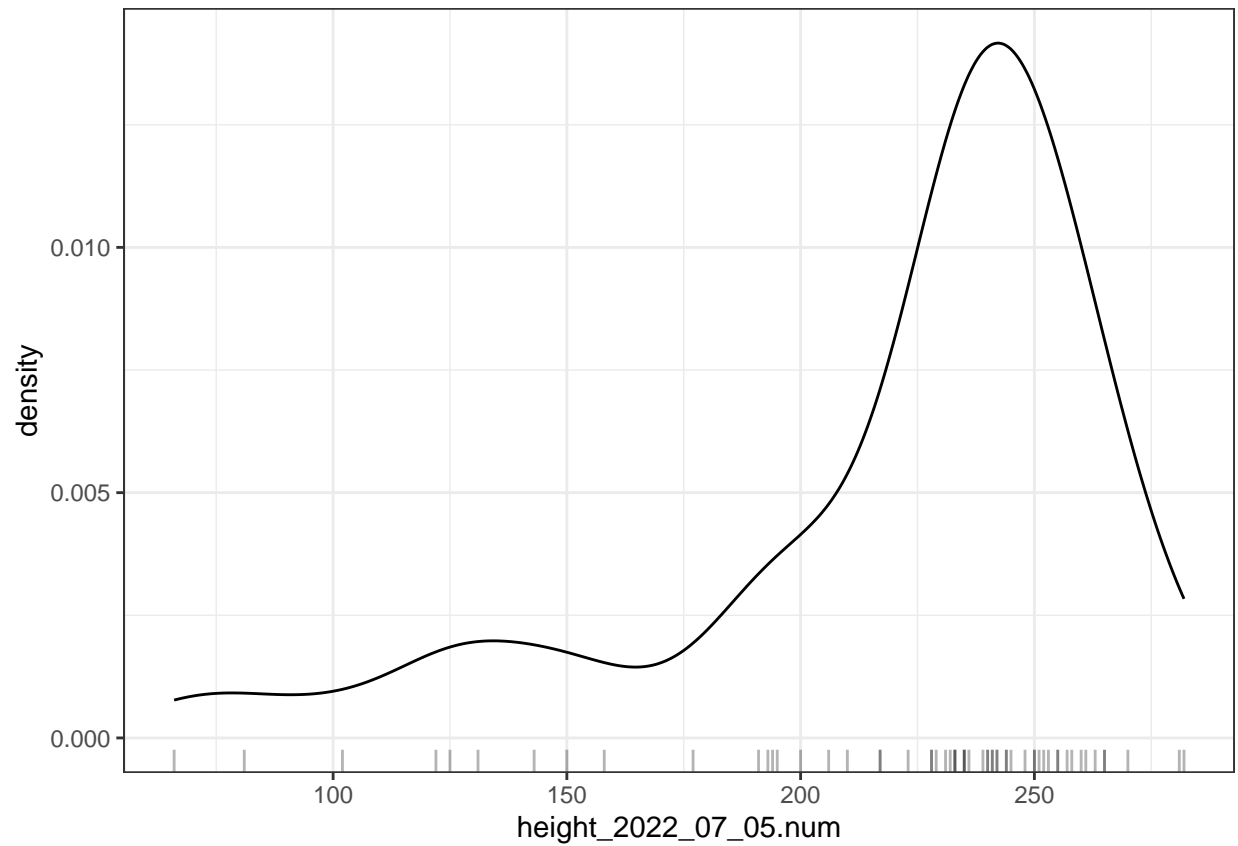
```
## show the first 10 non-missing values in decreasing order
```

```
table(d.maize$`height_2022_07_05.num`, useNA = "no")
```

| | | | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 66 | 81 | 102 | 122 | 125 | 131 | 143 | 150 | 158 | 177 | 191 | 193 | 194 | 195 | 200 | 206 | 210 | 217 | 223 | 228 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 |
| 229 | 231 | 232 | 233 | 235 | 236 | 239 | 240 | 241 | 242 | 244 | 245 | 248 | 250 | 251 | 252 | 253 | 255 | 257 | 258 |
| 1 | 1 | 1 | 3 | 3 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 |
| 260 | 261 | 263 | 265 | 270 | 281 | 282 | | | | | | | | | | | | | |
| 1 | 1 | 1 | 2 | 1 | 1 | 1 | | | | | | | | | | | | | |

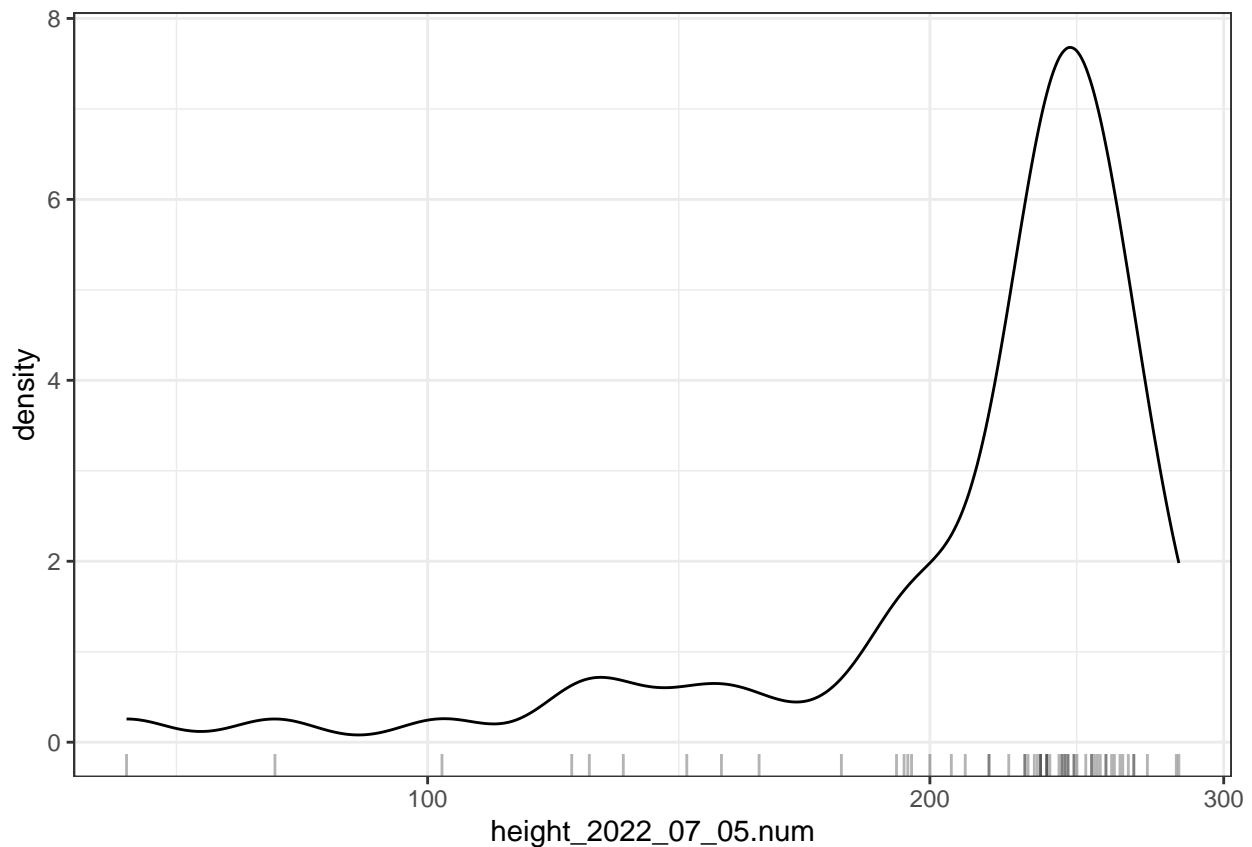
```
gg.density %>%
  d.maize +
  aes(x = `height_2022_07_05.num`, alpha = 0.3)
```

Warning: Removed 48 rows containing non-finite values (`stat_density()`).



```
##
gg.density %>%
  d.maize +
  aes(x = `height_2022_07_05.num`, alpha = 0.3) +
  scale_x_log10()
```

Warning: Removed 48 rows containing non-finite values (`stat_density()`).



7.19 Create *plant.found* (new)

We create a new variable telling us whether the plant was found in the field. Plants that were found in the field have a non-missing value for the variable `height_2022_07_05`. We check that the number of FALSE and TRUE coincides with the number of non-missing and missing values found for the variable `height_2022_07_05`, as well as checking randomly selected rows.

```
d.maize %<>%
  mutate(plant.found = !is.na(`height_2022_07_05`))
## overview missing values
is.na(d.maize$`height_2022_07_05`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
TRUE FALSE
  12    96
```

```
## overview TRUE/FALSE
d.maize$`plant.found` %>%
  # factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
FALSE TRUE
  12    96
```

```
## values mapped correctly for some randomly picked observations
## with the following row numbers:
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 35 70 74 42 38 20 28 103 44 87
```

```
(d.maize %>%
  select(height_2022_07_05, plant.found))[sel,]
```

```
# A tibble: 10 x 2
  height_2022_07_05 plant.found
  <chr>             <lgl>
1 231              TRUE
2 <NA>             FALSE
3 not measured     TRUE
4 125              TRUE
5 257              TRUE
6 265              TRUE
7 244              TRUE
8 not measured     TRUE
9 <NA>             FALSE
10 not measured    TRUE
```

7.20 Check *cob_weight*

The `cob_weight` variable is recorded as a character and presents several missing values, where a cob was not found. These measurements are taken on day 2022-09-16 and are in grams.

```
## class
class(d.maize$cob_weight)
```

```
[1] "character"
```

```
## overview missing values
is.na(d.maize$cob_weight) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```

TRUE FALSE
28      80
```

```
## number of levels (without missing values)
n_distinct(d.maize$cob_weight, na.rm = TRUE)
```

```
[1] 60
```

7.21 Create *cob_weight.num* (new)

We convert `cob_weight.num` to numeric and check that the mapping worked correctly for a randomly selected subset of the rows.

```
d.maize %<>%
  mutate(cob_weight.num = as.numeric(`cob_weight`))
## values mapped correctly for some randomly picked observations
```

```
## with the following row numbers:
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 70 40 44 25 108 39 51 42 6 24
```

```
(d.maize %>%
  select(contains("cob_weight")))[sel,]
```

```
# A tibble: 10 x 2
  cob_weight cob_weight.num
  <chr>      <dbl>
1 <NA>      NA
2 77        77
3 <NA>      NA
4 <NA>      NA
5 94        94
6 165       165
7 105       105
8 <NA>      NA
9 156       156
10 <NA>     NA
```

```
## overview missing values
is.na(d.maize$cob_weight.num) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
.
TRUE FALSE
28      80
```

```
## number of levels (without missing values)
n_distinct(d.maize$cob_weight.num, na.rm = TRUE)
```

```
[1] 60
```

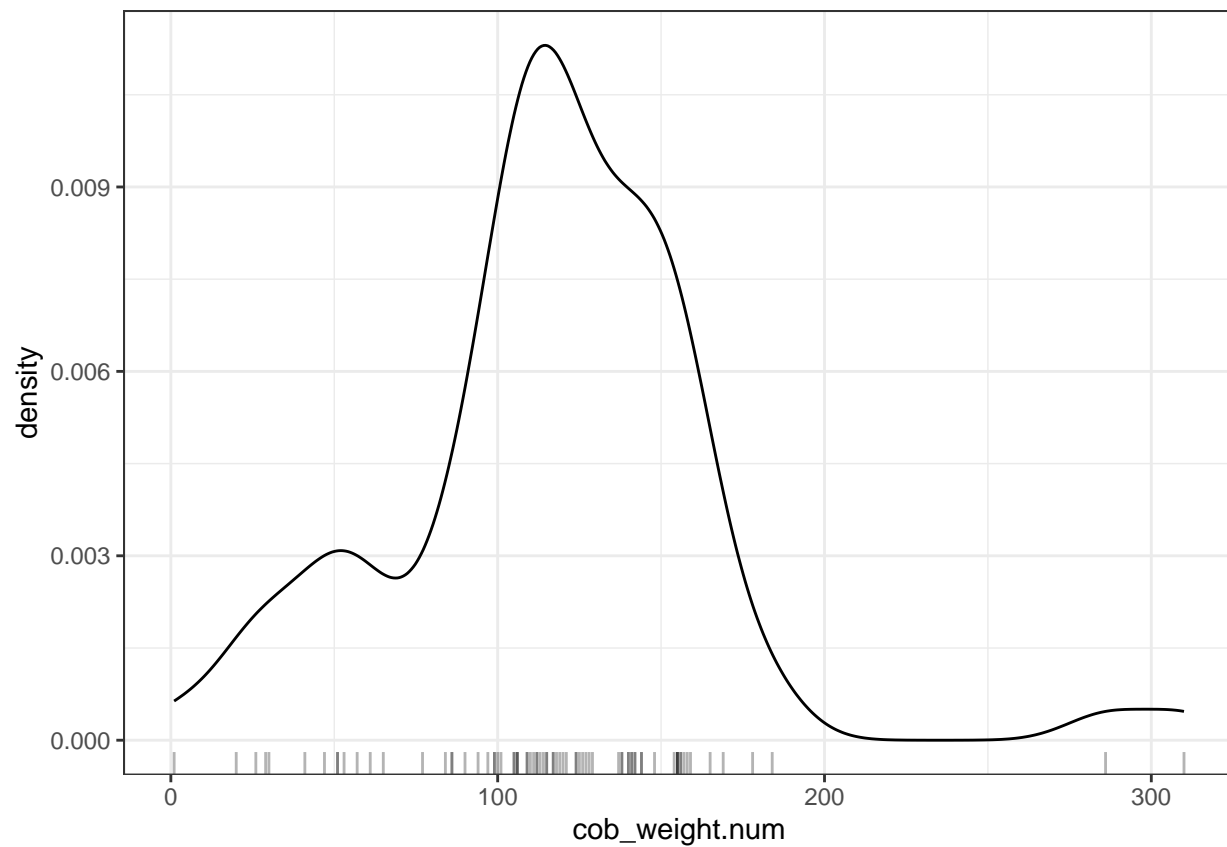
```
## show the first 10 non-missing values in decreasing order
table(d.maize$cob_weight.num, useNA = "no")
```

```

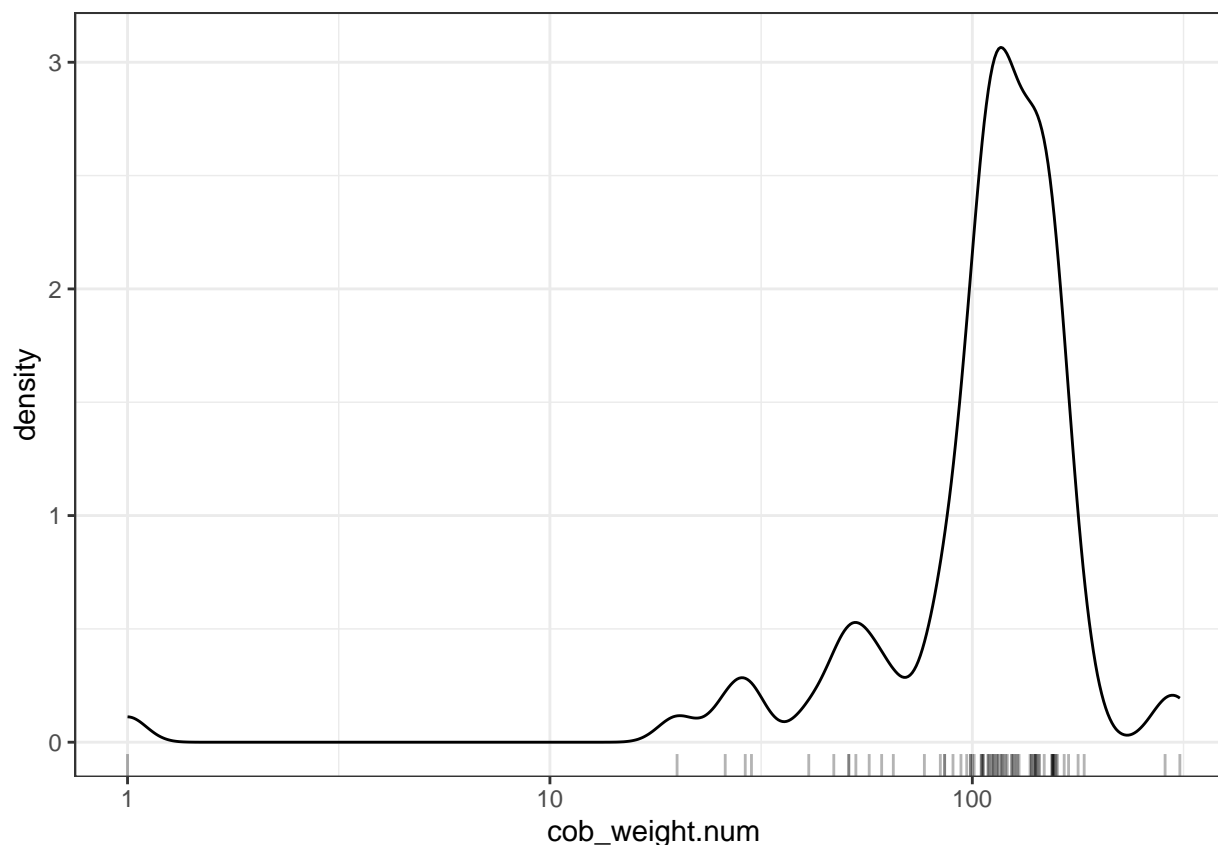
1  20  26  29  30  41  47  51  53  57  61  65  77  84  86  90  94  97  99 100
1   1   1   1   1   1   1   2   1   1   1   1   1   1   2   1   1   1   2   1
101 105 106 109 110 111 112 113 114 115 117 118 119 120 121 124 125 126 127 128
1   2   3   2   1   1   2   1   1   2   2   1   1   1   1   2   1   1   1   1
129 137 138 140 141 142 144 148 154 155 156 157 158 159 165 169 178 184 286 310
1   1   2   2   2   2   2   1   1   4   2   1   1   1   1   1   1   1   1   1
```

We plot the density of this new continuous variable.

```
d.maize %>%
  filter(!is.na(cob_weight.num)) %>%
  ggplot(aes(x = cob_weight.num)) +
  guides(alpha = "none") +
  geom_density() +
  geom_rug(alpha = 0.3)
```



```
d.maize %>%  
  filter(!is.na(cob_weight.num)) %>%  
  ggplot(aes(x = cob_weight.num)) +  
  guides(alpha = "none") +  
  geom_density() +  
  scale_x_log10() +  
  geom_rug(alpha = 0.3)
```



7.22 Check ...12

This row was part of the dataset but contained only one entry and the column name was missing. We thus ignore this row in our analysis.

```
class(d.maize$`...12`)
```

```
[1] "numeric"
```

```
summary(d.maize$`...12`)
```

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. | NA's |
|------|---------|--------|------|---------|------|------|
| 149 | 149 | 149 | 149 | 149 | 149 | 107 |

```
## overview missing values
```

```
is.na(d.maize$`...12`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
TRUE FALSE
107      1
```

7.23 Create *germinated.in.lab* (new)

We create a new binary variable telling us whether the seed germinated in the time it was in the lab, starting from the variable `date.germinated.asDate`. If a seed did not germinated in the lab,

date.germinated.asDate contains a missing value for this variable.

With the following code, missing values are automatically coerced to FALSE, which is what we want.

```
d.maize %<>%  
  mutate(germinated.in.lab = !is.na(`date.germinated.asDate`))  
## class  
class(d.maize$`germinated.in.lab`)
```

```
[1] "logical"
```

```
## values mapped correctly for some randomly picked observations  
## with the following row numbers:  
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 32 14  2 45 18 22 78 65 70 87
```

```
(d.maize %>%  
  select(germinated.in.lab, date.germinated.asDate))[sel,]
```

```
# A tibble: 10 x 2
```

| | germinated.in.lab | date.germinated.asDate |
|----|-------------------|------------------------|
| | <lgl> | <date> |
| 1 | TRUE | 2022-05-13 |
| 2 | TRUE | 2022-05-09 |
| 3 | TRUE | 2022-05-11 |
| 4 | TRUE | 2022-05-10 |
| 5 | TRUE | 2022-05-11 |
| 6 | TRUE | 2022-05-10 |
| 7 | FALSE | NA |
| 8 | TRUE | 2022-05-13 |
| 9 | FALSE | NA |
| 10 | TRUE | 2022-05-12 |

```
## overview missing values
```

```
is.na(d.maize$`germinated.in.lab`) %>%  
  factor(levels = c(TRUE, FALSE)) %>%  
  table()
```

```
.  
TRUE FALSE  
0    108
```

```
## show all non-missing values
```

```
table(d.maize$`germinated.in.lab`, useNA = "no")
```

```
FALSE TRUE  
24    84
```

```
table(d.maize$`germinated.in.lab`, useNA = "no") %>%  
  prop.table() %>%  
  round(digits = 2)
```

```
FALSE TRUE  
0.22  0.78
```

We can see that 22% of the seeds did not germinate in the lab. Note that some seeds did not germinate in the lab but would germinate later on in the field.

7.24 Create *germinated.in.field* (new)

We create a new binary variable telling us whether the seed germinated in the field. We can get this information by looking at all those observations that have not germinated in the lab (`germinated.in.lab == FALSE`) but that have a final height for the plant (`is.na(height_2022_07_05.num) == FALSE`) Note that seeds that have not germinated in field but have not germinated at all are also given a value of `FALSE`.

```
d.maize %<>%  
  mutate(germinated.in.field = (!`germinated.in.lab` & (!is.na(height_2022_07_05.num))))  
## class  
class(d.maize$`germinated.in.field`)
```

```
[1] "logical"
```

```
## values mapped correctly for some randomly picked observations  
## with the following row numbers:  
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 108 70 104 103 75 81 100 13 40 89
```

```
(d.maize %>%  
  select(germinated.in.field,  
         germinated.in.lab,  
         height_2022_07_05.num))[sel,]
```

```
# A tibble: 10 x 3
```

| | germinated.in.field | germinated.in.lab | height_2022_07_05.num |
|----|---------------------|-------------------|-----------------------|
| | <lgl> | <lgl> | <dbl> |
| 1 | FALSE | TRUE | NA |
| 2 | FALSE | FALSE | NA |
| 3 | FALSE | TRUE | NA |
| 4 | FALSE | FALSE | NA |
| 5 | FALSE | TRUE | NA |
| 6 | FALSE | TRUE | NA |
| 7 | FALSE | TRUE | NA |
| 8 | FALSE | TRUE | 217 |
| 9 | TRUE | FALSE | 250 |
| 10 | FALSE | FALSE | NA |

```
## overview missing values
```

```
is.na(d.maize$`germinated.in.field`) %>%  
  factor(levels = c(TRUE, FALSE)) %>%  
  table()
```

```
TRUE FALSE  
0 108
```

```
## show all non-missing values
```

```
table(d.maize$`germinated.in.field`, useNA = "no")
```

```
FALSE TRUE  
102 6
```

```
table(d.maize$`germinated.in.field`, useNA = "no") %>%  
  prop.table() %>%  
  round(digits = 2)
```

```
FALSE TRUE
0.94 0.06
```

Since some seeds did not germinate in the lab but germinated in the field (about 6% of all seeds), the variable `date.germinated.asDate` is actually a truncated observation (because we put NA for those seeds that did not germinate in the lab, but ignore whether they did germinate eventually).

7.25 Create *germinated.yes* (new)

We create a new variable telling us whether the seed germinated at all at some point during the experiment. These are the seeds that for which `germinated.in.field` and `germinated.in.lab` differ. When they are equal, it can only mean that the seed did not germinate in the lab nor in the field, and hence did not germinate at all.

```
d.maize %<>%
  mutate(germinated.yes = (`germinated.in.field` != `germinated.in.lab`))
## class
class(d.maize$`germinated.yes`)
```

```
[1] "logical"
```

```
## values mapped correctly for some randomly picked observations
## with the following row numbers:
(sel <- sample(1:nrow(d.maize), 10))
```

```
[1] 48 89 23 84 103 29 13 22 93 28
```

```
(d.maize %>%
  select(germinated.yes,
         germinated.in.field,
         germinated.in.lab))[sel,]
```

```
# A tibble: 10 x 3
  germinated.yes germinated.in.field germinated.in.lab
  <lgl>          <lgl>          <lgl>
1 TRUE          FALSE          TRUE
2 FALSE         FALSE          FALSE
3 TRUE          FALSE          TRUE
4 TRUE          FALSE          TRUE
5 FALSE         FALSE          FALSE
6 TRUE          FALSE          TRUE
7 TRUE          FALSE          TRUE
8 TRUE          FALSE          TRUE
9 FALSE         FALSE          FALSE
10 TRUE         FALSE          TRUE
```

```
## overview missing values
is.na(d.maize$`germinated.yes`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
TRUE FALSE
0      108
```

```
## number of levels (without missing values)
n_distinct(d.maize$`germinated.yes`, na.rm = TRUE)
```

```
[1] 2
```

```
## show all non-missing values in decreasing order
table(d.maize$`germinated.yes`, useNA = "no")
```

```
FALSE TRUE
  18    90
```

```
table(d.maize$`germinated.yes`, useNA = "no") %>%
  prop.table() %>%
  round(digits = 2)
```

```
FALSE TRUE
  0.17  0.83
```

In total, 17% of all seeds did not germinate at all.

7.26 Create *days.to.germination* (new)

We create a new variable telling us how many days germination took. All seeds have been first watered on 2022-04-30.

Seeds that have not germinated receive a NA value.

We do not censor seeds that have not germinated but instead leave them with missing values.

```
d.maize %<>%
  mutate(days.to.germination = as.numeric(date.germinated.asDate - as.Date("2022-04-30")))
## class
class(d.maize$`days.to.germination`)
```

```
[1] "numeric"
```

```
## overview missing values
is.na(d.maize$`days.to.germination`) %>%
  factor(levels = c(TRUE, FALSE)) %>%
  table()
```

```
TRUE FALSE
  24    84
```

```
## number of levels (without missing values)
n_distinct(d.maize$`days.to.germination`, na.rm = TRUE)
```

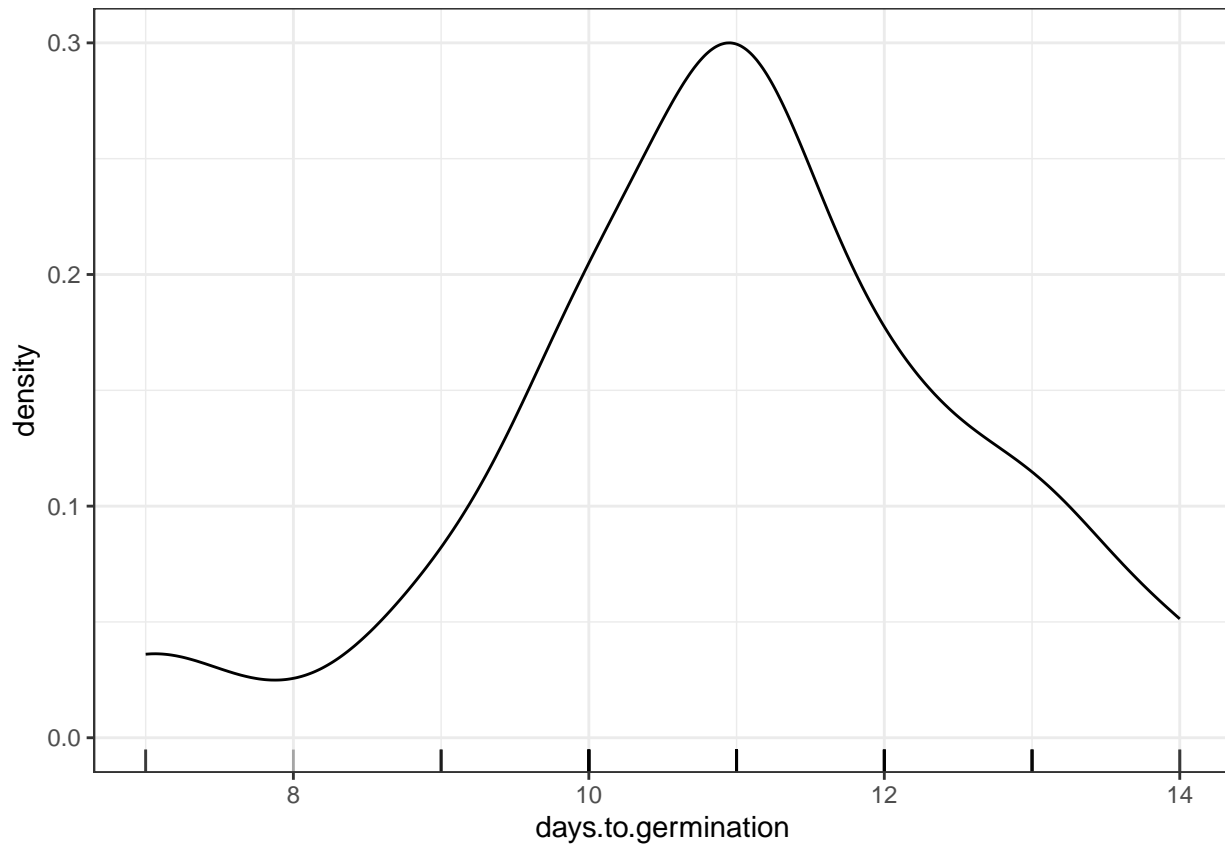
```
[1] 8
```

```
## show all non-missing values in decreasing order
table(d.maize$`days.to.germination`, useNA = "no")
```

```
7  8  9 10 11 12 13 14
4  1  6 17 29 13 10  4
```

We visualize the density

```
d.maize %>%
  filter(!is.na(days.to.germination)) %>%
  ggplot(aes(x = days.to.germination)) +
  guides(alpha = "none") +
  geom_density() +
  geom_rug(alpha = 0.3)
```



7.27 Create *days.to.germination.censored* (new)

We create another variable, *days.to.germination.censored*, which takes into account the fact that the missing values are censored observations.

```
days.to.germination.censored.tmp <- d.maize$days.to.germination
days.to.germination.censored.tmp[is.na(days.to.germination.censored.tmp)] <-
  max(days.to.germination.censored.tmp, na.rm = TRUE)
d.maize <- d.maize %>%
  mutate(days.to.germination.censored = days.to.germination.censored.tmp)
##
## check
d.maize %>%
  select(contains("days.to.germination"), germinated.in.lab) %>%
  unique()
```

```
# A tibble: 9 x 3
```

```
  days.to.germination days.to.germination.censored germinated.in.lab
```

| | <dbl> | <dbl> <lgl> |
|---|-------|-------------|
| 1 | 11 | 11 TRUE |
| 2 | 9 | 9 TRUE |
| 3 | 10 | 10 TRUE |
| 4 | NA | 14 FALSE |
| 5 | 7 | 7 TRUE |
| 6 | 13 | 13 TRUE |
| 7 | 14 | 14 TRUE |
| 8 | 12 | 12 TRUE |
| 9 | 8 | 8 TRUE |

```
## makes sense
```

7.28 Create `seed_coord_y` (new)

We create a new variable encapsulating the coordinate y of each seed in the lab.

The correct English sentence is: “The pots are positioned on three lines, with their long sides along the vertical line. Consequently, there are 9 wells in each column.

This means that the range of values for the variable `seed_coord_y` spans from 1 to 9.

For example, the seed in *pot* “A1” and *well* “a”, will have coordinate y equal to 1; the seed in *pot* “B1” and *well* “c”, will have coordinate y equal to 5.

First of all we create two temporary variables, where we stock the line number inside the pot, and the letter of the pot.

```
library(stringr)
d.maize <- d.maize %>%
  mutate(seed_coord_y.tmp = ifelse(well %in% c("a", "b"),
                                   yes = 1,
                                   no = ifelse(well %in% c("c", "d"),
                                               yes = 2,
                                               no = 3)))

## check
d.maize %>%
  select(seed_coord_y.tmp, well) %>%
  unique()
```

```
# A tibble: 6 x 2
  seed_coord_y.tmp well
      <dbl> <chr>
1         1 a
2         1 b
3         2 c
4         2 d
5         3 e
6         3 f
```

```
## makes sense
```

```
##
```

```
d.maize <- d.maize %>%
  mutate(capital.letter = str_extract(pot,
                                       pattern = "[A-Z]"))
```

```
##
```

```
## check
```

```
set.seed(10)
d.maize %>%
  select(pot, capital.letter) %>%
  unique() %>%
  sample_n(10)
```

```
# A tibble: 10 x 2
  pot capital.letter
<chr> <chr>
1 B5 B
2 B3 B
3 B4 B
4 B6 B
5 B1 B
6 B2 B
7 A6 A
8 C2 C
9 A3 A
10 C1 C
```

```
## makes sense for these 10 observations
```

Now, we can create the variable `seed_coord_y`. if the pot letter is “B”, we add “3” to the value inside the variable `coord_y.tmp`; If the pot letter is “C”, we add “6”.

```
d.maize <- d.maize %>%
  mutate(seed_coord_y = ifelse(capital.letter == "B",
                                yes = seed_coord_y.tmp + 3,
                                no = ifelse(capital.letter == "C",
                                              yes = seed_coord_y.tmp + 6,
                                              no = seed_coord_y.tmp)))

##
## check
set.seed(2023)
d.maize %>%
  select(capital.letter, well, seed_coord_y) %>%
  unique() %>%
  sample_n(10)
```

```
# A tibble: 10 x 3
  capital.letter well seed_coord_y
<chr> <chr> <dbl>
1 C d 8
2 C c 8
3 B c 5
4 B b 4
5 A c 2
6 B d 5
7 B f 6
8 A b 1
9 A a 1
10 C a 7
```

```
## makes sense for these 10 observations
```

```
## we can now remove the two temporary variables
```

```
d.maize <- d.maize %>%
```

```

  select(-c(capital.letter, seed_coord_y.tmp))
##
## double check
table(d.maize$seed_coord_y, useNA = "ifany")

```

```

 1  2  3  4  5  6  7  8  9
12 12 12 12 12 12 12 12 12

```

```

## makes sense: there are 12 seeds for each of the 9 lines

```

7.29 Create *seed_coord_x* (new)

We create a new variable telling us the coordinate x of each seed.

There are 6 pots on each row, and they have their small side on the x axis. Consequently, there are 12 wells on each line.

This means that the value for the variable *seed_coord_x* spans from 1 to 12.

For example, the seed in *pot* “A1” and *well* “a”, will have coordinate x equal to 1; the seed in *pot* “B3” and *well* “d”, will have coordinates x equal to 6.

First of all we create two temporary variables, where we stock the column number inside the pot, and the letter of the pot.

```

d.maize <- d.maize %>%
  mutate(seed_coord_x.tmp = ifelse(well %in% c("a", "c", "e"),
                                   yes = 1,
                                   no = 2))
## check
d.maize %>%
  select(seed_coord_x.tmp, well) %>%
  unique()

```

```

# A tibble: 6 x 2
  seed_coord_x.tmp well
      <dbl> <chr>
1         1 a
2         2 b
3         1 c
4         2 d
5         1 e
6         2 f

```

```

## makes sense
##
d.maize <- d.maize %>%
  mutate(pot.nb = str_extract(pot,
                              pattern = "\\d"))
##
## check
set.seed(2023)
d.maize %>%
  select(pot, pot.nb) %>%
  unique() %>%
  sample_n(10)

```

```
# A tibble: 10 x 2
```

```
  pot pot.nb
  <chr> <chr>
1 C4    4
2 C3    3
3 B3    3
4 B2    2
5 A3    3
6 B4    4
7 B6    6
8 A2    2
9 A1    1
10 C1    1
```

```
## makes sense for these 10 observations
```

Now, we can create the variable `seed_coord_x`. If the pot number is “2”, we add “2” to the value inside the variable `coord_x.tmp`; If the pot number is “3”, we add “4”, and so on.

```
d.maize <- d.maize %>%
  mutate(
    seed_coord_x =
      ifelse(pot.nb == 2,
        yes = seed_coord_x.tmp + 2,
        no = ifelse(pot.nb == 3,
          yes = seed_coord_x.tmp + 4,
          no = ifelse(pot.nb == 4,
            yes = seed_coord_x.tmp + 6,
            no = ifelse(pot.nb == 5,
              yes = seed_coord_x.tmp + 8,
              no = ifelse(pot.nb == 6,
                yes = seed_coord_x.tmp + 10,
                no = seed_coord_x.tmp
              )))
          )))
  )
##
## check
set.seed(2023)
d.maize %>%
  select(pot.nb, well, seed_coord_x) %>%
  unique() %>%
  sample_n(10)
```

```
# A tibble: 10 x 3
```

```
  pot.nb well seed_coord_x
  <chr>  <chr>      <dbl>
1 3      d         6
2 2      b         4
3 5      b        10
4 1      a         1
5 5      e         9
6 3      e         5
7 6      a        11
8 6      b        12
9 1      e         1
10 6     e        11
```

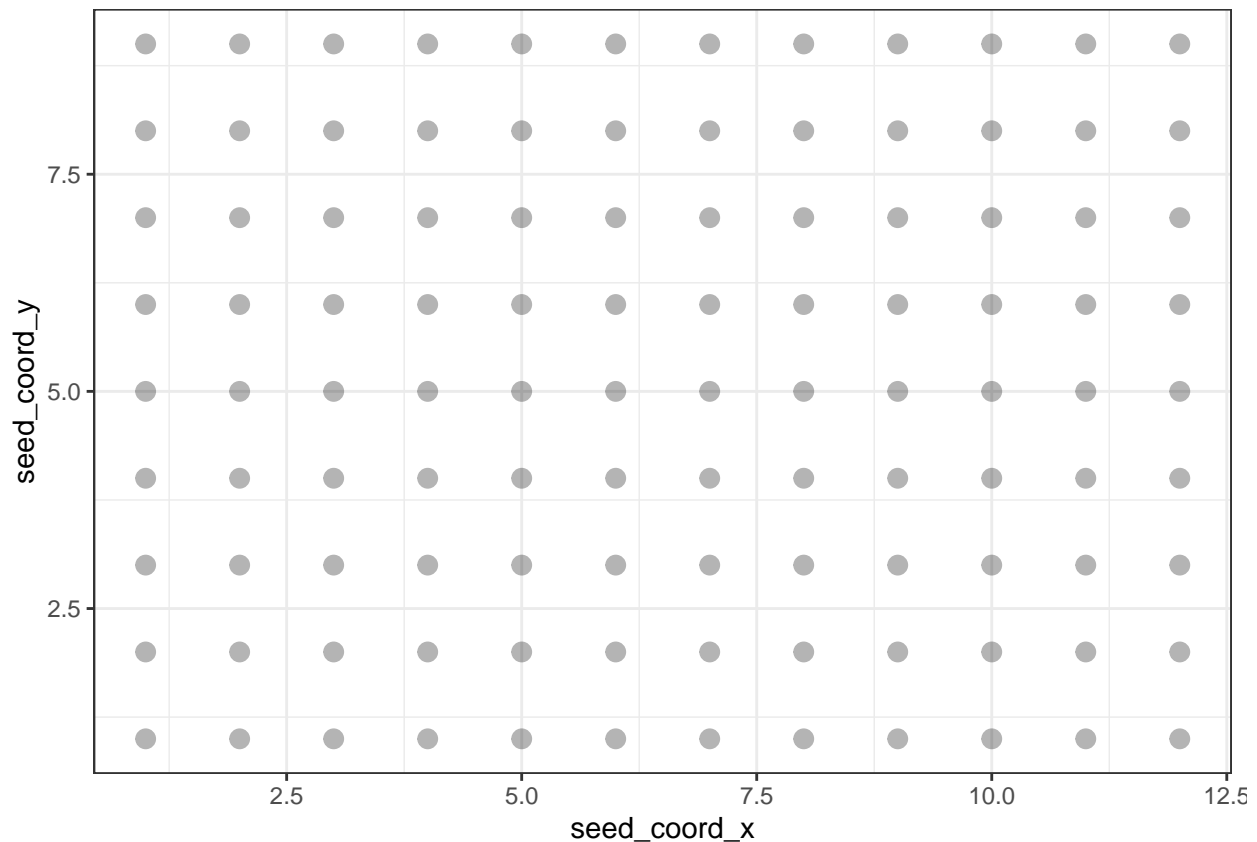
```
## makes sense for these 10 observations
## we can now remove the two temporary variables
d.maize <- d.maize %>%
  select(-c(pot.nb, seed_coord_x.tmp))
##
## double check
table(d.maize$seed_coord_x)
```

```
1  2  3  4  5  6  7  8  9 10 11 12
9  9  9  9  9  9  9  9  9  9  9  9
```

makes sense: there are 9 seeds, for each of the 12 columns.

We do another check, to verify that no seed has been badly coded.

```
ggplot(d.maize, mapping = aes(x = seed_coord_x, y = seed_coord_y)) +
  geom_point(alpha = 0.3, pch = 19, size = 3)
```



7.30 Create *position_field_x* (new)

We create a new variable, *position_field_x*, where we store the coordinate x of the seed, once he has been planted in the field.

The first column is composed only by A1-A4 pots. Starting from the lower left coin, growing.

```
d.maize <- d.maize %>%
  mutate(position_field_x = 0)
##
d.maize$position_field_x[d.maize$pot.fac %in% c("A1", "A2", "A3", "A4")] <- 1
##
## check
d.maize %>%
  select(pot.fac, position_field_x) %>%
  unique()
```

```
# A tibble: 18 x 2
  pot.fac position_field_x
  <fct>         <dbl>
1 A1             1
2 A2             1
3 A3             1
4 A4             1
5 A5             0
6 A6             0
7 B1             0
8 B2             0
9 B3             0
10 B4            0
11 B5            0
12 B6            0
13 C1            0
14 C2            0
15 C3            0
16 C4            0
17 C5            0
18 C6            0
```

```
## makes sense
```

The second column is composed by A5-A6 and B1-B2 pots.

```
d.maize$position_field_x[d.maize$pot.fac %in% c("A5", "A6", "B1", "B2")] <- 2
##
## check
d.maize %>%
  select(pot.fac, position_field_x) %>%
  unique()
```

```
# A tibble: 18 x 2
  pot.fac position_field_x
  <fct>         <dbl>
1 A1             1
2 A2             1
3 A3             1
4 A4             1
5 A5             2
6 A6             2
7 B1             2
8 B2             2
9 B3             0
```

```

10 B4      0
11 B5      0
12 B6      0
13 C1      0
14 C2      0
15 C3      0
16 C4      0
17 C5      0
18 C6      0

```

```
## makes sense
```

The third column is composed by B3-B5 pots and “a”, “b”, “c”, “d” wells of B6 pot.

```

d.maize$position_field_x[d.maize$pot.fac %in% c("B3", "B4", "B5")] <- 3
d.maize$position_field_x[d.maize$pot.fac == "B6" &
  d.maize$well.fac %in% c("a", "b", "c", "d")] <- 3

```

```
##
```

```
## check
```

```

d.maize %>%
  select(pot.fac, position_field_x) %>%
  unique()

```

```
# A tibble: 19 x 2
```

| | pot.fac | position_field_x |
|----|---------|------------------|
| | <fct> | <dbl> |
| 1 | A1 | 1 |
| 2 | A2 | 1 |
| 3 | A3 | 1 |
| 4 | A4 | 1 |
| 5 | A5 | 2 |
| 6 | A6 | 2 |
| 7 | B1 | 2 |
| 8 | B2 | 2 |
| 9 | B3 | 3 |
| 10 | B4 | 3 |
| 11 | B5 | 3 |
| 12 | B6 | 3 |
| 13 | B6 | 0 |
| 14 | C1 | 0 |
| 15 | C2 | 0 |
| 16 | C3 | 0 |
| 17 | C4 | 0 |
| 18 | C5 | 0 |
| 19 | C6 | 0 |

```

d.maize %>%
  filter(pot.fac == "B6") %>%
  select(position_field_x, well) %>%
  unique()

```

```
# A tibble: 6 x 2
```

| | position_field_x | well |
|---|------------------|-------|
| | <dbl> | <chr> |
| 1 | 3 | a |
| 2 | 3 | b |

```
3          3 c
4          3 d
5          0 e
6          0 f
```

```
## make sense
```

The forth column is composed by “e”, “f” wells of B6 pot, C1-C2 pots, and “a”, “b”, “c”, “d”, “e” wells of C3 pot

```
d.maize$position_field_x[d.maize$pot.fac %in% c("C1", "C2")] <- 4
d.maize$position_field_x[d.maize$pot.fac == "B6" &
  d.maize$well.fac %in% c("e", "f")] <- 4
d.maize$position_field_x[d.maize$pot.fac == "C3" &
  d.maize$well.fac %in% c("a", "b", "c", "d", "e")] <- 4

##
## check
d.maize %>%
  select(pot.fac, position_field_x) %>%
  unique()
```

```
# A tibble: 20 x 2
  pot.fac position_field_x
  <fct>      <dbl>
1 A1          1
2 A2          1
3 A3          1
4 A4          1
5 A5          2
6 A6          2
7 B1          2
8 B2          2
9 B3          3
10 B4          3
11 B5          3
12 B6          3
13 B6          4
14 C1          4
15 C2          4
16 C3          4
17 C3          0
18 C4          0
19 C5          0
20 C6          0
```

```
d.maize %>%
  filter(pot.fac == "B6") %>%
  select(position_field_x, well) %>%
  unique()
```

```
# A tibble: 6 x 2
  position_field_x well
  <dbl> <chr>
1          3 a
2          3 b
3          3 c
```

```
4          3 d
5          4 e
6          4 f
```

```
d.maize %>%
  filter(pot.fac == "C3") %>%
  select(position_field_x, well) %>%
  unique()
```

```
# A tibble: 6 x 2
  position_field_x well
      <dbl> <chr>
1             4 a
2             4 b
3             4 c
4             4 d
5             4 e
6             0 f
```

```
## make sense
```

The last column contains “f” well of C3 pot, and C4-C6 pots.

```
d.maize$position_field_x[d.maize$pot.fac %in% c("C4", "C5", "C6")] <- 5
d.maize$position_field_x[d.maize$pot.fac == "C3" &
  d.maize$well.fac %in% c("f")] <- 5
```

```
##
## check
d.maize %>%
  select(pot.fac, position_field_x) %>%
  unique()
```

```
# A tibble: 20 x 2
  pot.fac position_field_x
  <fct>      <dbl>
1 A1          1
2 A2          1
3 A3          1
4 A4          1
5 A5          2
6 A6          2
7 B1          2
8 B2          2
9 B3          3
10 B4         3
11 B5         3
12 B6         3
13 B6         4
14 C1         4
15 C2         4
16 C3         4
17 C3         5
18 C4         5
19 C5         5
20 C6         5
```

```
d.maize %>%
  filter(pot.fac == "C3") %>%
  select(position_field_x, well) %>%
  unique()
```

```
# A tibble: 6 x 2
  position_field_x well
      <dbl> <chr>
1             4 a
2             4 b
3             4 c
4             4 d
5             4 e
6             5 f
```

```
## make sense
```

Moreover, we know that the distance between rows is of 50cm, thus we multiply by this value the coordinates of the seeds.

```
d.maize <- d.maize %>%
  mutate(position_field_x_cm = position_field_x * 50)
##
## check
set.seed(2023)
d.maize %>%
  select(starts_with("position_field_x")) %>%
  sample_n(10)
```

```
# A tibble: 10 x 2
  position_field_x position_field_x_cm
      <dbl>          <dbl>
1             4             200
2             2             100
3             5             250
4             4             200
5             2             100
6             2             100
7             5             250
8             3             150
9             2             100
10            3             150
```

```
## makes sense
```

7.31 Create *position_field_y* (new)

We create a new variable, *position_field_y*, where we store the coordinate y of the seed, once he has been planted in the field.

```
d.maize <- d.maize %>%
  arrange(pot.fac) %>%
  mutate(position_field_y = c(1:24, 1:24, 1:22, 1:19, 1:19))
##
## check
table(d.maize$position_field_y)
```

```

1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  3  3  3  2  2

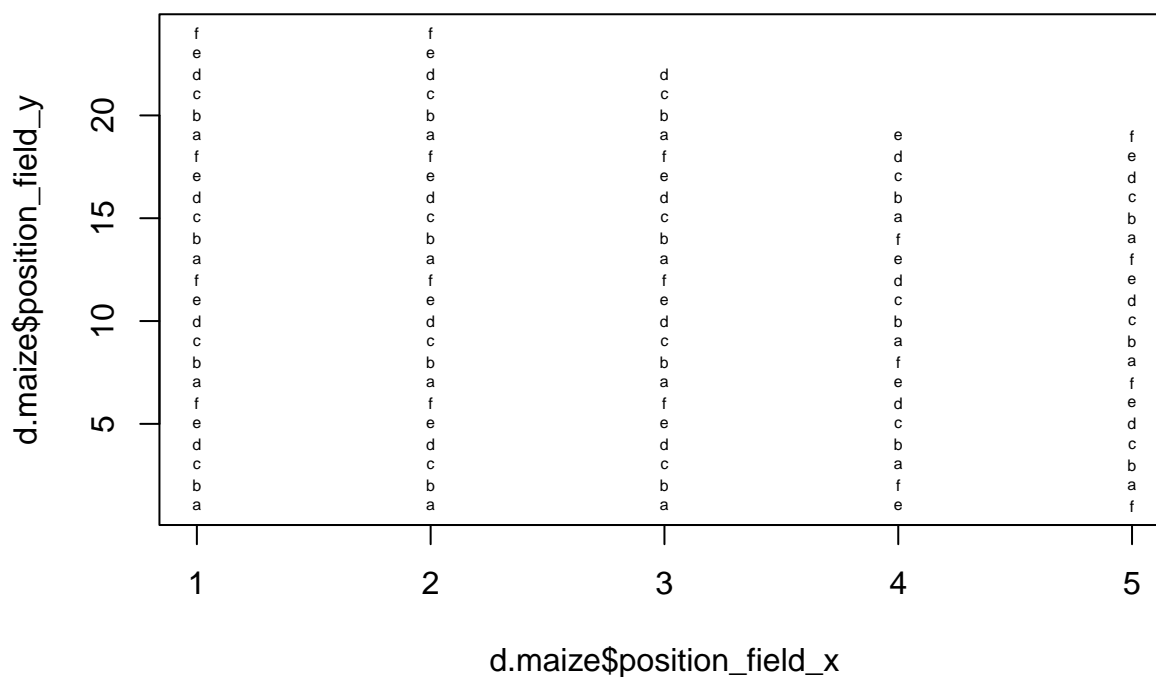
```

We double check with a plot.

```

plot(x = d.maize$position_field_x,
     y = d.maize$position_field_y,
     pch = d.maize$well,
     cex = 0.5)

```



```
## makes sense
```

Moreover, we know that the distance between plants in the same row is of 25cm.

```

d.maize <- d.maize %>%
  mutate(position_field_y_cm = position_field_y * 25)
##
## check
set.seed(2023)
d.maize %>%
  select(starts_with("position_field_y")) %>%
  sample_n(10)

```

```

# A tibble: 10 x 2
  position_field_y position_field_y_cm
      <int>         <dbl>
1         10         250

```

| | | |
|----|----|-----|
| 2 | 23 | 575 |
| 3 | 16 | 400 |
| 4 | 2 | 50 |
| 5 | 2 | 50 |
| 6 | 20 | 500 |
| 7 | 9 | 225 |
| 8 | 17 | 425 |
| 9 | 5 | 125 |
| 10 | 1 | 25 |

```
## makes sense
```

8 Missing values

```
## (warning messages are omitted from this chunk)
##
d.maize %>%
  mutate(across(.cols = everything(), .fns = is.na)) %>%
  tidyr::pivot_longer(cols = everything()) %>%
  mutate(name = factor(x = name, levels = unique(name))) %>%
  group_by(name) %>%
  summarize(n = sum(value),
            n.perc = format(x = round(x = n/n() * 100, digits = 1), nsmall = 1)) %>%
  ungroup() %>%
  rename("Missings" = "n",
        "Missing (%)" = "n.perc") %>%
  kable(caption = "Overview over the number and percentages of missing values",
        label = "tbl_missings",
        booktabs=TRUE,
        longtable = TRUE,
        linesep = c("")) %>%
  kable_styling(font_size = 7,
                latex_options = c("striped", "repeat_header", "hold_position"))
```

Table 1: Overview over the number and percentages of missing values

| name | Missings | Missing (%) |
|------------------------|----------|-------------|
| pot | 0 | 0.0 |
| soil | 0 | 0.0 |
| well | 0 | 0.0 |
| depth | 0 | 0.0 |
| seed.weight | 0 | 0.0 |
| fungus | 107 | 99.1 |
| date.germinated | 24 | 22.2 |
| observations | 101 | 93.5 |
| height_2022_07_05 | 12 | 11.1 |
| cob_weight | 28 | 25.9 |
| ...12 | 107 | 99.1 |
| pot.fac | 0 | 0.0 |
| soil.fac | 0 | 0.0 |
| well.fac | 0 | 0.0 |
| seed.weight.grams | 0 | 0.0 |
| fungus.fac | 0 | 0.0 |
| date.germinated.asDate | 24 | 22.2 |
| obs.time | 0 | 0.0 |
| broken | 0 | 0.0 |
| height_2022_07_05.num | 48 | 44.4 |
| plant.found | 0 | 0.0 |

Table 1: Overview over the number and percentages of missing values (*continued*)

| name | Missings | Missing (%) |
|------------------------------|----------|-------------|
| cob_weight.num | 28 | 25.9 |
| germinated.in.lab | 0 | 0.0 |
| germinated.in.field | 0 | 0.0 |
| germinated.yes | 0 | 0.0 |
| days.to.germination | 24 | 22.2 |
| days.to.germination.censored | 0 | 0.0 |
| seed_coord_y | 0 | 0.0 |
| seed_coord_x | 0 | 0.0 |
| position_field_x | 0 | 0.0 |
| position_field_x_cm | 0 | 0.0 |
| position_field_y | 0 | 0.0 |
| position_field_y_cm | 0 | 0.0 |

9 Creating the RDS file

```
saveRDS(d.maize,
        file = paste0("./Prepared_data_and_models/", "d.maize_PreparedData.RDS"))
write.csv2(x = d.maize,
           fileEncoding = "UTF-8",
           file = paste0("./Prepared_data_and_models/", "d.maize_PreparedData.csv"),
           quote = TRUE,
           row.names = FALSE)
```

10 Session information

```
sessionInfo()
```

```
R version 4.3.1 (2023-06-16)
```

```
Platform: aarch64-apple-darwin20 (64-bit)
```

```
Running under: macOS Sonoma 14.0
```

```
Matrix products: default
```

```
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
```

```
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK vers
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Europe/Zurich
```

```
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] stringr_1.5.0    survival_3.5-5    magrittr_2.0.3    readxl_1.4.3
```

```
[5] tidyr_1.3.0      ggplot2_3.4.4     kableExtra_1.3.4  dplyr_1.1.3
```

```
[9] checkpoint_1.0.2 knitr_1.44
```

```
loaded via a namespace (and not attached):
```

```
[1] Matrix_1.6-1.1    gtable_0.3.4      compiler_4.3.1    webshot_0.5.5
```

```
[5] tidyselect_1.2.0  xml2_1.3.5         splines_4.3.1     systemfonts_1.0.5
```

```
[9] scales_1.2.1      yaml_2.3.7         fastmap_1.1.1     lattice_0.21-8
```

```
[13] R6_2.5.1          labeling_0.4.3     generics_0.1.3    tibble_3.2.1
```

```
[17] munsell_0.5.0     svglite_2.1.2      pillar_1.9.0      rlang_1.1.1
```

```
[21] utf8_1.2.4        stringi_1.7.12     xfun_0.40         viridisLite_0.4.2
```

```
[25] cli_3.6.1         withr_2.5.1        digest_0.6.33     rvest_1.0.3
```

```
[29] grid_4.3.1        rstudioapi_0.15.0  lifecycle_1.0.3   vctrs_0.6.4
```

```
[33] evaluate_0.22     glue_1.6.2         farver_2.1.1      cellranger_1.1.0
```

```
[37] fansi_1.0.5       colorspace_2.1-0   purrr_1.0.2       rmarkdown_2.25
```

```
[41] httr_1.4.7        tools_4.3.1        pkgconfig_2.0.3   htmltools_0.5.6.1
```