Because, obviously, I am not going to make my son’s actual weight data available on the internet, I created some mock data, similar to real-world examples (weight in *g*):

library(readr) library(lubridate)

weight\_measures <- read\_delim("weight\_measures.csv",

delim = ",",

col\_types = list(col\_date(format = "%d.%m.%Y"),

col\_double())) head(weight\_measures)

## # A tibble: 6 x 2

|  |  |  |  |
| --- | --- | --- | --- |
| ##  ## |  | date | weight |
| ## | 1 | 2019-08-12 | 3168 |
| ## | 2 | 2019-08-16 | 3126 |
| ## | 3 | 2019-08-17 | 3138 |
| ## | 4 | 2019-08-18 | 3348 |
| ## | 5 | 2019-08-20 | 3286 |
| ## | 6 | 2019-08-22 | 3626 |

In the Baby Weight Shiny app, you can upload this mock data or your own data. Then choose:

whether the columns in your .csv are separated by **comma, semicolon or tabulator**

whether you want to show only reference values for the **first 13 weeks** or for the **first 5 years** (only reference values corresponding to your measurement dates will be shown)

whether your weight data is given in **gramm or kilogramm** and whether you want to display reference data for **girls or boys**

You can then check to see if the data looks correct (tab: *Data*). The tabs *Curve* and *Barchart* show two plots. Below, I’ll describe how I prepared the data for plotting.

# Data preparation

First, I combined my weight measures with the reference tables from WHO and then wrote the below Functions and R scripts.

Function – combine\_measures.R

|  |
| --- |
| combine\_measures\_who <- function(weight\_measures, |
|  | p\_0\_13, p\_0\_5, |
|  | age\_range, # "0\_13" or "0\_5" |
|  | weight\_in, # "g" or "kg" |
|  | gender) { # "boy" or "girl" |
|  |  |
|  | # if weight in g convert to kg |
|  | if (weight\_in == "g") { |
|  | weight\_measures <- weight\_measures %>% |
|  | mutate(weight = weight / 1000) |
|  | } |
|  |  |
|  | # get first and last date in table |
|  | reference\_date <- weight\_measures$date[[1]] |
|  | end\_date <- weight\_measures$date[[nrow(weight\_measures)]] + 31 |
|  |  |
|  | # get weight at birth for calculating the reference percentile |
|  | reference\_weight <- weight\_measures$weight[[1]] |
|  |  |
|  | # filter for input age range & sex |
|  | if (age\_range == "0\_13") { |
|  | if (gender == "boy") { |
|  | who <- p\_0\_13 %>% |
|  | filter(gender == "boys") |
|  |  |
|  | } else if (gender == "girl") { |
|  | who <- p\_0\_13 %>% |
|  | filter(gender == "girls") |
|  |  |
|  | } else { |
|  | stop('"gender" either has to be "boy" or "girl"') |
|  | } |
|  |  |
|  | who <- who %>% |
|  | mutate(date = rep(seq(from = reference\_date, by = "week", length.out = 14), 15), |
|  | ref = "WHO") |
|  |  |
|  | who\_f <- who %>% |
|  | filter(Week == 0) |
|  |  |
|  | } else if (age\_range == "0\_5") { |
|  | if (gender == "boy") { |
|  | who <- p\_0\_5 %>% |
|  | filter(gender == "boys") |
|  |  |
|  | } else if (gender == "girl") { |
|  | who <- p\_0\_5 %>% |
|  | filter(gender == "girls") |
|  |  |
|  | } else { |
|  | stop('"gender" either has to be "boy" or "girl"') |
|  | } |
|  |  |
|  | who <- who %>% |
|  | mutate(date = rep(seq(from = reference\_date, by = "month", length.out = 61), 15), |
|  | ref = "WHO") %>% |
|  | filter(date <= !!paste(end\_date)) |
|  |  |
|  | who\_f <- who %>% |
|  | filter(Month == 0) |
|  |  |
|  | } else { |
|  | stop('"age\_range" either has to be "0\_13" or "0\_5"') |
|  | } |
|  |  |
|  | who <- weight\_measures %>% |
|  | mutate(ref = "measurement", |
|  | percentile = "measurement", |
|  | starting\_p = who\_f[which.min(abs(who\_f$weight - reference\_weight)), ]$percentile) %>% |
|  | bind\_rows(who) |
|  |  |
|  | return(who) |
|  | } |

Function – Combine\_measures\_who\_2.R

|  |
| --- |
| combine\_measures\_who\_2 <- function(weight\_measures, |
|  | weight\_in, # "g" or "kg" |
|  | gender) { # "boy" or "girl" |
|  |  |
|  | if (weight\_in == "g") { |
|  | weight\_measures <- weight\_measures %>% |
|  | mutate(weight = weight / 1000) |
|  | } |
|  |  |
|  | reference\_date <- weight\_measures$date[[1]] |
|  | end\_date <- weight\_measures$date[[nrow(weight\_measures)]] + 31 |
|  |  |
|  | reference\_weight <- weight\_measures$weight[[1]] |
|  |  |
|  | p\_0\_13\_2 <- p\_0\_13 %>% |
|  | mutate(date = rep(seq(from = reference\_date, by = "week", length.out = 14), 30), |
|  | ref = "WHO") |
|  |  |
|  | p\_0\_5\_2 <- p\_0\_5 %>% |
|  | mutate(date = rep(seq(from = reference\_date, by = "month", length.out = 61), 30), |
|  | ref = "WHO") |
|  |  |
|  | who <- p\_0\_13\_2 %>% |
|  | bind\_rows(p\_0\_5\_2) %>% |
|  | filter(date <= !!paste(end\_date)) |
|  |  |
|  | if (gender == "boy") { |
|  | who <- who %>% |
|  | filter(gender == "boys") |
|  |  |
|  | } else if (gender == "girl") { |
|  | who <- who %>% |
|  | filter(gender == "girls") |
|  |  |
|  | } else { |
|  | stop('"gender" either has to be "boy" or "girl"') |
|  | } |
|  |  |
|  | who\_f <- who %>% |
|  | filter(Week == 0) |
|  |  |
|  | who <- weight\_measures %>% |
|  | mutate(ref = "measurement", |
|  | percentile = "measurement", |
|  | starting\_p = who\_f[which.min(abs(who\_f$weight - reference\_weight)), ]$percentile) %>% |
|  | bind\_rows(who) |
|  |  |
|  | return(who) |
|  | } |

Function – elapsed\_months.R

|  |
| --- |
| elapsed\_months <- function(end\_date, start\_date) { |
|  | ed <- as.POSIXlt(end\_date) |
|  | ed\_2 <- ed$year \* 12 + ed$mon |
|  |  |
|  | sd <- as.POSIXlt(start\_date) |
|  | sd\_2 <- sd$year \* 12 + sd$mon |
|  |  |
|  | mo <- ed\_2 - sd\_2 |
|  | mo <- ifelse(day(ed) >= day(sd), mo + 1, mo) |
|  | return(mo) |
|  | } |

Shiny\_App.R

|  |
| --- |
| library(shiny) |
|  | library(shinythemes) |
|  |  |
|  | library(readr) |
|  | library(tidyverse) |
|  | library(plotly) |
|  | library(modelr) |
|  | library(lubridate) |
|  |  |
|  | source("functions/combine\_measures.R") |
|  | source("functions/elapsed\_months.R") |
|  |  |
|  | # Define UI for data upload app ---- |
|  | ui <- fluidPage( |
|  | theme = shinytheme("paper"), |
|  |  |
|  | # App title ---- |
|  | titlePanel("WHO Baby Weight Reference"), |
|  |  |
|  | helpText("Compare your baby's weight with WHO references for boys and girls. Upload your weight measurements below:"), |
|  |  |
|  | # Sidebar layout with input and output definitions ---- |
|  | sidebarLayout( |
|  |  |
|  | # Sidebar panel for inputs ---- |
|  | sidebarPanel( |
|  |  |
|  | # Input: Select a file ---- |
|  | fileInput("file1", "Choose CSV File", |
|  | multiple = FALSE, |
|  | accept = c("text/csv", |
|  | "text/comma-separated-values,text/plain", |
|  | ".csv")), |
|  |  |
|  | # Input: Select separator ---- |
|  | radioButtons("sep", "Separator", |
|  | choices = c(Comma = ",", |
|  | Semicolon = ";", |
|  | Tab = "\t"), |
|  | selected = ","), |
|  |  |
|  | radioButtons("age\_range", "Age range", |
|  | choices = c('Weeks 1 through 13' = "0\_13", |
|  | 'Years 1 through 5' = "0\_5"), |
|  | selected = "0\_5"), |
|  |  |
|  | radioButtons("weight\_in", "Weight in", |
|  | choices = c(Gramm = "g", |
|  | Kilogramm = "kg"), |
|  | selected = "g"), |
|  |  |
|  | radioButtons("gender", "Gender", |
|  | choices = c(Boy = "boy", |
|  | Girl = "girl"), |
|  | selected = "boy") |
|  |  |
|  | ), |
|  |  |
|  | # Main panel for displaying outputs ---- |
|  | mainPanel( |
|  |  |
|  | tabsetPanel(type = "tabs", |
|  | tabPanel("Data", |
|  | verbatimTextOutput("input\_instructions"), |
|  | DT::dataTableOutput("contents"), |
|  | uiOutput("example\_csv")), |
|  | tabPanel("Curve", |
|  | plotlyOutput("plot\_curve"), |
|  | htmlOutput("correlation")), |
|  | tabPanel("Barchart", |
|  | plotlyOutput("plot\_bar"), |
|  | htmlOutput("weight\_diff")) |
|  | ) |
|  | ) |
|  | ), |
|  |  |
|  | hr(), |
|  | div(class = "footer", |
|  | includeHTML("footer.html") |
|  | ) |
|  | ) |
|  |  |
|  | # Define server logic to read selected file ---- |
|  | server <- function(input, output, session) { |
|  |  |
|  | url <- a("example CSV file with dummy data.", href="https://github.com/ShirinG/who\_baby\_weight\_app/blob/master/data/weight\_measures.csv") |
|  |  |
|  | output$example\_csv <- renderUI({ |
|  | tagList("Follow this link to download an", url) |
|  | }) |
|  |  |
|  | output$input\_instructions <- renderText({ |
|  | "Your weight measurements need to have the following format:\n |
|  | - CSV format with comma, semicolon or tab separator\n |
|  | - two columns: column 1 titled date in format '%d.%m.%Y' (e.g. 28.02.2019)\n |
|  | - column 2 titled weight with measurements in Gramm or Kilogramm\n |
|  | - the first row has to contain birth date and birth weight" |
|  | }) |
|  |  |
|  | output$contents <- DT::renderDataTable({ |
|  |  |
|  | req(input$file1) |
|  |  |
|  | weight\_measures <- read\_delim(input$file1$datapath, |
|  | delim = input$sep |
|  | ) |
|  |  |
|  | return(weight\_measures) |
|  |  |
|  | }) |
|  |  |
|  | weight\_measures <- reactive({ |
|  | req(input$file1) |
|  |  |
|  | weight\_measures <- read\_delim(input$file1$datapath, |
|  | delim = input$sep, |
|  | col\_types = list(col\_date(format = "%d.%m.%Y"), col\_double()) |
|  | ) |
|  |  |
|  | }) |
|  |  |
|  | combine\_measures\_who\_final <- reactive({ |
|  |  |
|  | p\_0\_5 <- read\_csv("data/p\_0\_5.csv") |
|  | p\_0\_13 <- read\_csv("data/p\_0\_13.csv") |
|  |  |
|  | combine\_measures\_who\_final <- combine\_measures\_who(weight\_measures(), |
|  | p\_0\_13, p\_0\_5, |
|  | age\_range = input$age\_range, |
|  | weight\_in = input$weight\_in, |
|  | gender = input$gender) |
|  | }) |
|  |  |
|  | weight\_measures\_all <- reactive({ |
|  | weight\_measures <- weight\_measures() |
|  |  |
|  | # add missing dates for calculating weight change per week of life |
|  | reference\_date <- weight\_measures$date[[1]] |
|  | end\_date <- weight\_measures$date[[nrow(weight\_measures)]] |
|  |  |
|  | all\_dates <- seq(from = reference\_date, to = end\_date, by = "day") %>% |
|  | as\_tibble() |
|  | colnames(all\_dates) = "date" |
|  |  |
|  | weight\_measures\_all <- weight\_measures %>% |
|  | full\_join(all\_dates, by = "date") |
|  |  |
|  | ## arrange by date and approximate missing values |
|  | weight\_measures\_all <- weight\_measures\_all %>% |
|  | arrange(date) %>% |
|  | mutate(weight\_approx = approx(weight, n = nrow(.))[[2]]) |
|  |  |
|  | ## add running week number |
|  | nos <- rep(1:ceiling(nrow(weight\_measures\_all)/7), each = 7) |
|  | weight\_measures\_all$week <- nos[1:nrow(weight\_measures\_all)] |
|  |  |
|  | ## calculate sum over week |
|  | weight\_measures\_all <- weight\_measures\_all %>% |
|  | mutate(diff\_day = c(0, diff(weight\_approx, lag = 1)), |
|  | diff\_week = c(rep(0, 7), diff(weight\_approx, lag = 7))) |
|  |  |
|  | # calculate number of elapsed months btw dates |
|  | weight\_measures\_all <- weight\_measures\_all %>% |
|  | mutate(month = elapsed\_months(date, reference\_date)) |
|  |  |
|  | #may I suggest using case\_when instead? to me it seems more readable than a bunch of nested ifelse statements! |
|  | weight\_measures\_all <- weight\_measures\_all %>% |
|  | mutate(color = case\_when( |
|  | month <= 2 & diff\_week >= 170 ~ "ok", |
|  | month == 3 & diff\_week >= 110 ~ "ok", |
|  | month == 4 & diff\_week >= 110 ~ "ok", |
|  | month == 5 & diff\_week >= 70 ~ "ok", |
|  | month == 6 & diff\_week >= 70 ~ "ok", |
|  | month >= 7 & diff\_week >= 40 ~ "ok", |
|  | TRUE ~ "low")) |
|  | }) |
|  |  |
|  | starting\_p <- reactive({ |
|  | ## find reference percentile |
|  | reference\_date <- weight\_measures()$date[[1]] |
|  | starting\_p <- combine\_measures\_who\_final() %>% |
|  | filter(date == reference\_date) %>% |
|  | select(starting\_p) %>% |
|  | .[[1]] %>% |
|  | .[1] |
|  | }) |
|  |  |
|  | test\_curves <- reactive({ |
|  | test\_curves <- combine\_measures\_who\_final() %>% |
|  | filter(percentile == !!paste(starting\_p())) %>% |
|  | left\_join(select(weight\_measures\_all(), date, weight\_approx), by = "date") |
|  | }) |
|  |  |
|  | output$plot\_curve <- renderPlotly({ |
|  |  |
|  | gender <- ifelse(input$gender == "boy", "boys", "girls") |
|  | age <- ifelse(input$age\_range == "0\_13", "weeks 1 through 13", "years 1 through 5") |
|  |  |
|  | plot\_final\_curve <- combine\_measures\_who\_final() %>% |
|  | ggplot(aes(date, weight, |
|  | linetype = ref, |
|  | color = percentile)) + |
|  | geom\_line() + |
|  | geom\_point() + |
|  | labs(x = "Date", |
|  | y = "Weight in kg", |
|  | title = paste("WHO child growth standard percentiles shown for", gender, age)) + |
|  | theme\_bw() + |
|  | theme(legend.position = "bottom") |
|  |  |
|  | ggplotly(plot\_final\_curve) |
|  | }) |
|  |  |
|  | output$correlation <- renderUI({ |
|  | paste("Your reference percentile is:", starting\_p(), |
|  | "\n. Correlation between your measurements and your reference percentile is:", |
|  | round(cor(test\_curves()$weight, test\_curves()$weight\_approx, use = 'complete.obs'), digits = 5)) |
|  | }) |
|  |  |
|  | output$plot\_bar <- renderPlotly({ |
|  |  |
|  | plot\_final\_bar <- weight\_measures\_all() %>% |
|  | ggplot(aes(x = date, y = diff\_week, fill = color)) + |
|  | geom\_bar(stat = "identity") + |
|  | scale\_fill\_brewer(palette = "Set1") + |
|  | labs(x = "Date", |
|  | y = "Weight difference in gramm", |
|  | fill = "Weight gain", |
|  | title = "Weekly weight differences: (Approximated) weight for each day compared to weight at 7 days prior") + |
|  | theme\_bw() |
|  |  |
|  | ggplotly(plot\_final\_bar) |
|  | }) |
|  |  |
|  | output$weight\_diff <- renderUI({ |
|  | url <- a("this German site about breastfeeding", href="https://www.stillkinder.de/gewicht-und-wachstum-von-gestillten-kindern/") |
|  | tagList("Bar colors show whether the weekly weight gain is above (blue) or below (red) the required minimum for BREASTFED babies given by", |
|  | url, |
|  | ". In months 1 and 2, the minimum weight gain should be: 170 g, in months 3 and 4: 110 g, in months 5 and 6: 70 g and from month 7 on: 40 g.") |
|  | }) |
|  |  |
|  | } |
|  | # Run the app ---- |
|  | shinyApp(ui, server) |

library(tidyverse)

source("combine\_measures.R") p\_0\_5 <- read\_csv("p\_0\_5.csv") p\_0\_13 <- read\_csv("p\_0\_13.csv")

combine\_measures\_who\_final <- combine\_measures\_who(weight\_measures,

p\_0\_13, p\_0\_5, age\_range = "0\_5", weight\_in = "g", gender = "boy")

head(combine\_measures\_who\_final)

## # A tibble: 6 x 10

## date weight ref percentile starting\_p Month L M S gender

##

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 1 | 2019-08-12 | 3.17 | measur… | measureme… | P50 | NA | NA | NA | NA |
| ## | 2 | 2019-08-16 | 3.13 | measur… | measureme… | P50 | NA | NA | NA | NA |
| ## | 3 | 2019-08-17 | 3.14 | measur… | measureme… | P50 | NA | NA | NA | NA |
| ## | 4 | 2019-08-18 | 3.35 | measur… | measureme… | P50 | NA | NA | NA | NA |
| ## | 5 | 2019-08-20 | 3.29 | measur… | measureme… | P50 | NA | NA | NA | NA |
| ## | 6 | 2019-08-22 | 3.63 | measur… | measureme… | P50 | NA | NA | NA | NA |

In order to calculate weight data for every full week of life (to make the values comparable with the reference), I am interpolating weight values for every day with missing information (linear approximation).

# add missing dates for calculating weight change per week of life reference\_date <- weight\_measures$date[[1]]

end\_date <- weight\_measures$date[[nrow(weight\_measures)]]

starting\_p <- combine\_measures\_who\_final %>% filter(date == reference\_date) %>% select(starting\_p) %>%

.[[1]] %>%

.[1]

all\_dates <- seq(from = reference\_date, to = end\_date, by = "day") %>% as\_tibble()

colnames(all\_dates) = "date"

weight\_measures\_all <- weight\_measures %>% full\_join(all\_dates, by = "date") %>% arrange(date)

## approximate missing values weight\_measures\_all <- weight\_measures\_all %>%

mutate(weight\_approx = approx(weight, n = nrow(.))[[2]])

## add running week number

nos <- rep(1:ceiling(nrow(weight\_measures\_all)/7), each = 7) weight\_measures\_all$week <- nos[1:nrow(weight\_measures\_all)]

## calculate sum over week weight\_measures\_all <- weight\_measures\_all %>%

mutate(diff\_day = c(0, diff(weight\_approx, lag = 1)), diff\_week = c(rep(0, 7), diff(weight\_approx, lag = 7)))

source("elapsed\_months.R") weight\_measures\_all <- weight\_measures\_all %>%

mutate(month = elapsed\_months(date, reference\_date))

# The curve

The first plot shows a simple lineplot with all reference curves (i.e. all WHO growth percentiles) and my measurements plotted against them. The plot in the Shiny app has been created with **Plotly**, so you can click on the legend to hide/show specific growth percentiles or zoom into specific areas on the plot.

gender <- "boys"

age <- "years 1 through 5"

combine\_measures\_who\_final %>% ggplot(aes(date, weight,

linetype = ref,

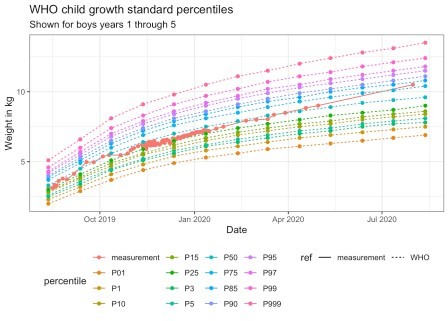
color = percentile)) + geom\_line() +

geom\_point() + labs(x = "Date",

y = "Weight in kg",

title = "WHO child growth standard percentiles", subtitle = paste("Shown for", gender, age)) +

theme\_bw() + theme(legend.position = "bottom")



test\_curves <- combine\_measures\_who\_final %>% filter(percentile == !!paste(starting\_p)) %>%

left\_join(select(weight\_measures\_all, date, weight\_approx), by = "date")

paste("Your reference percentile is:", starting\_p) ## [1] "Your reference percentile is: P50"

paste("Correlation between your measurements and your reference percentile is:", round(cor(test\_curves$weight, test\_curves$weight\_approx, use =

'complete.obs'), digits = 5))

## [1] "Correlation between your measurements and your reference percentile is: 0.98757"

# Approximated missing values

The next plot is not shown in the Shiny app, but here I’ll include it to show how the interpolated weight values look like:

weight\_measures\_all %>% ggplot(aes(x = date)) +

geom\_line(aes(y = weight\_approx), color = "grey") +

geom\_point(aes(y = weight\_approx), color = "blue", size = 1, alpha = 0.6) + geom\_point(aes(y = weight), color = "red", size = 3, alpha = 0.6) + geom\_point(data = weight\_measures\_all %>% distinct(week, .keep\_all = TRUE),

aes(x = date, y = weight\_approx),

color = "grey", size = 2, alpha = 0.6) +

geom\_label(data = weight\_measures\_all %>% distinct(week, .keep\_all = TRUE), aes(x = date, y = weight\_approx, label = week),

nudge\_y = 500, alpha = 0.6) +

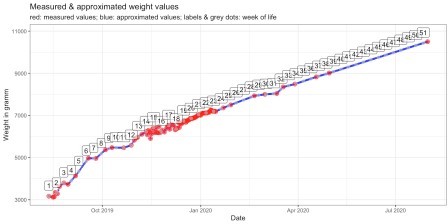
labs(x = "Date",

y = "Weight in gramm",

title = "Measured & approximated weight values",

subtitle = "red: measured values; blue: approximated values; labels & grey dots: week of life") +

theme\_bw()



# Barchart

The barchart, which you can find in the Shiny app, shows weekly weight differences, i.e. for every day the difference in weight compared to 7 days prior is calculated.

Bar colors show whether the weekly weight difference is above (blue) or below (red) the required minimum for **BREASTFED** babies given by this German site about breastfeeding:

in months 1 and 2, the minimum weight gain should be: 170 g in months 3 and 4: 110 g

in months 5 and 6: 70 g and

from month 7 to 12: 40 g

weight\_measures\_all <- weight\_measures\_all %>%

mutate(color = ifelse(month <= 2 & diff\_week >= 170, "ok",

ifelse(month == 3 & diff\_week >= 110, "ok", ifelse(month == 4 & diff\_week >= 110, "ok",

ifelse(month == 5 & diff\_week >= 70, "ok", ifelse(month == 6 & diff\_week >=

70, "ok",

>= 40, "ok", "low")))))))

ifelse(month >=7 & diff\_week

The actual plot in the Shiny app is again created with **Plotly**, so you can interact with the graph there.

weight\_measures\_all %>%

ggplot(aes(x = date, y = diff\_week)) + geom\_bar(aes(fill = color), stat = "identity") +

geom\_label(data = weight\_measures\_all %>% distinct(week, .keep\_all = TRUE), aes(x = date, label = week),

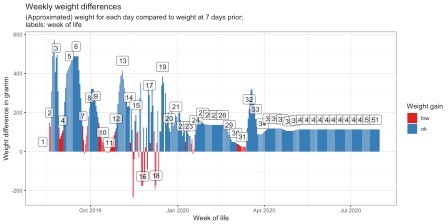
nudge\_y = 50, alpha = 0.6) + scale\_fill\_brewer(palette = "Set1") + labs(x = "Week of life",

y = "Weight difference in gramm", fill = "Weight gain",

title = "Weekly weight differences",

subtitle = "(Approximated) weight for each day compared to weight at 7 days prior;\nlabels: week of life") +

theme\_bw()



sessionInfo()

## R version 4.0.2 (2020-06-22)

## Platform: x86\_64-apple-darwin17.0 (64-bit) ## Running under: macOS Catalina 10.15.6

##

## Matrix products: default

## BLAS: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib

/libRblas.dylib

## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib

/libRlapack.dylib ##

## locale:

## [1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8 ##

## attached base packages:

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

## other attached packages:

## [1] forcats\_0.5.0 stringr\_1.4.0 dplyr\_1.0.2 purrr\_0.3.4

## [5] tidyr\_1.1.2 tibble\_3.0.3 ggplot2\_3.3.2 tidyverse\_1.3.0 ## [9] lubridate\_1.7.9 readr\_1.3.1

##

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

## [1] tidyselect\_1.1.0 xfun\_0.16 haven\_2.3.1 colorspace\_1.4-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | [5] | vctrs\_0.3.3 | generics\_0.0.2 | htmltools\_0.5.0 | yaml\_2.2.1 |
| ## | [9] | utf8\_1.1.4 | blob\_1.2.1 | rlang\_0.4.7 | pillar\_1.4.6 |
| ## | [13] | withr\_2.2.0 | glue\_1.4.2 | DBI\_1.1.0 |  |
| RColorBrewer\_1.1-2 | | |  |  |  |
| ## [17] dbplyr\_1.4.4 | | | modelr\_0.1.8 | readxl\_1.3.1 | lifecycle\_0.2.0 |
| ## [21] munsell\_0.5.0 | | | blogdown\_0.20 | gtable\_0.3.0 |  |
| cellranger\_1.1.0 | | |  |  |  |
| ## | [25] | rvest\_0.3.6 | evaluate\_0.14 | labeling\_0.3 | knitr\_1.29 |
| ## | [29] | fansi\_0.4.1 | broom\_0.7.0 | Rcpp\_1.0.5 | backports\_1.1.9 |
| ## | [33] | scales\_1.1.1 | jsonlite\_1.7.0 | farver\_2.0.3 | fs\_1.5.0 |
| ## | [37] | hms\_0.5.3 | digest\_0.6.25 | stringi\_1.4.6 | bookdown\_0.20 |
| ## | [41] | grid\_4.0.2 | cli\_2.0.2 | tools\_4.0.2 | magrittr\_1.5 |
| ## | [45] | crayon\_1.3.4 | pkgconfig\_2.0.3 | ellipsis\_0.3.1 | xml2\_1.3.2 |
| ## | [49] | reprex\_0.3.0 | rstudioapi\_0.11 | assertthat\_0.2.1 | rmarkdown\_2.3 |
| ## | [53] | httr\_1.4.2 | R6\_2.4.1 | compiler\_4.0.2 |  |