Data Wrangling 101

for BCS 206

Fall 2019

Contents

1 Goals for next two weeks

- Thinking about workflow in R:
  - Version control
  - R Markdown
- Data wrangling: Turning the data into the form you need (*dplyr*)
- Data visualization:
  - General principles
  - How to plot in R (*ggplot*, *plotly*)

We only have a relatively short time, so we will focus on learning what tools are available and on *examples* of use (rather than an in-depth tutorial). There are great online tutorials and cheatsheets that contain further information.

2 Preliminaries

2.1 Version control

RStudio makes version control, data backup, and data sharing easy (e.g., via Github.com). To use it, download and install git on your computer. Get a free github.com or bitbucket.com account. You only have to do this once.

Then, for each project, create a new project in RStudio and link it to the remote repository (select "Create project" > "Version control"). You will have to enter a URL for the remote repository, which you get, for example, at github.com under the repository's main page by clicking the "Clone or download button".

For step by step instructions, see:

- Setting up RStudio for version control
- RStudio help on version control
- Reverting a file to an earlier version

2.2 Reproducibility and literate coding

R and RStudio support reproducibility oriented literate coding via Sweave and Knitr: lab books, presentations, and papers can weave/knit together data, code, and text. The document you share contains the code needed to create its outputs (figures, tables, etc.). This is achieved by combining latex or R markdown with R code (or, for that matter, code from other programming languages). For an excellent video-based introduction, see this tutorial on R markdown. *This document is R markdown compiled with RStudio's knitr.

3 Data wrangling

The *R* libraries *dplyr* provide us with efficient ways to transform ('wrangle') our data tables. The library *magrittr* let's us concatenate these operations in transparent and easy to read code.

3.1 An example data set

We will illustrate the use of *dplyr* with the following data from an experiment with a 2AFC task in three within-subject conditions (A, B, C), for which we have extracted correctness (1 = correct; 0 = incorrect) and reaction times (RT):

```r
summary(d)
```

```
##  condition     trial          subject        correct
##  A:2688    Min.   : 1.00   1      : 192   Min.   :0.0000
##  B:2688    1st Qu.:16.75   2      : 192   1st Qu.:0.0000
##  C:2688    Median :32.50   3      : 192   Median :1.0000
##            Mean   :32.50   4      : 192   Mean   :0.6198
##            3rd Qu.:48.25   5      : 192   3rd Qu.:1.0000
##            Max.   :64.00   6      : 192   Max.   :1.0000
##                              (Other):6912
##        RT
##  Min.   : 188.6
##  1st Qu.: 421.1
##  Median : 567.0
##  Mean   : 766.1
##  3rd Qu.:1064.8
##  Max.   :2548.9
##
```

```r
glimpse(d)
```

```
## Observations: 8,064
## Variables: 5
## $ condition <fct> A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A...
## $ trial     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ subject   <fct> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
## $ correct   <int> 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0...
## $ RT        <dbl> 419.679, 546.658, 367.420, 345.559, 434.292, 756.464...
```

3.2 Dplyr's verbs

Dplyr has 'verbs' like filter, select, summarize, mutate, transmute, etc. to let use conduct operations on our data, and reshape the data frame into the format we need. We can use dplyr, for example, to calculate the proportion correct answers in our experiment by using *summarise*.

```r
summarise(d, meanCorrect = mean(correct))
```

```
## # A tibble: 1 x 1
##   meanCorrect
##         <dbl>
## 1       0.620
```

Or just for condition A:

```r
d.A = filter(d, condition == "A")
summarise(d.A, meanCorrect = mean(correct))
```

```
## # A tibble: 1 x 1
##   meanCorrect
##         <dbl>
## 1       0.488
```

3.3 Maggritr's pipes

Here we will use only of the 'pipes' magrittr provides:

- x %>% f: takes x and hands it to the function f on the right, as f's first argument
- x %<>% f1 %>% f2 %>% etc.: takes x hands it to f1, takes the output of f1 and hands it to f2, etc. And since the first pipe was %<>% (rather than just %>%), the final result will be written back into x.

3.4 Putting it together: Wrangling through pipes

Remember how we got the mean proportion correct for just Condition A?

Figure 1: Magritt's pipe

Figure 2: Magrittr's pipe

```
d.A = filter(d, condition == "A")
summarise(d.A, meanCorrect = mean(correct))
```

```
## # A tibble: 1 x 1
##   meanCorrect
##         <dbl>
## 1       0.488
```

This is inelegant and hard to read. Pipes let us make this more transparent:

```
d %>%
  filter(condition == "A") %>%
  summarise(meanCorrect = mean(correct))
```

```
## # A tibble: 1 x 1
##   meanCorrect
##         <dbl>
## 1       0.488
```

And this advantage becomes even clearer, the more operationgs we concatenate. For example, *group_by* is an elegant operator that tells the pipes to conduct all subsequent operations for each of the groups (and then put all the separate outcomes back together into a single data frame). So if we want the proportion correct for all groups:

```
d %>%
  group_by(condition) %>%
  summarise(meanCorrect = mean(correct))
```

```
## # A tibble: 3 x 2
##   condition meanCorrect
##   <fct>           <dbl>
## 1 A               0.488
## 2 B               0.572
## 3 C               0.799
```

4 Exercises

How can we:

- View the entire data set? (*View*)
- Calculate the by-subject averages for all three conditions? (*group_by*, *summarise*)
- Calculate the by-subject standard deviations around those averages? (*group_by*, *summarise*)
- Attach this information (the averages and SDs) to each row of the present data.frame? (*group_by*, *mutate*)
- Determine whether RTs were on average faster for correct, as compared to incorrect, trials?
- Add a column for log-transformed RTs to the data set?
- Remove the old column for raw RTs? (*select*)
- Sort the data by log-transformed reaction times? (*arrange*)

Say we further have an additional data frame with information about our subjects:

```
## Source: local data frame [42 x 3]
## Groups: <by row>
##
## # A tibble: 42 x 3
##    subject gender   age
##    <fct>   <chr>  <dbl>
##  1 1       male      20
##  2 2       female    19
##  3 3       male      20
##  4 4       male      20
##  5 5       male      18
##  6 6       male      20
##  7 7       male      20
##  8 8       male      20
##  9 9       male      19
## 10 10      female    19
## # ... with 32 more rows
```

- How can we join the information from the two data sources together? (*left_join*)

```
## Source: local data frame [42 x 3]
## Groups: <by row>
##
## # A tibble: 42 x 3
##    subject gender   age
##    <fct>   <chr>  <dbl>
##  1 1       male      20
##  2 2       female    19
##  3 3       male      20
##  4 4       male      20
##  5 5       male      18
##  6 6       male      20
##  7 7       male      20
##  8 8       male      20
##  9 9       male      19
## 10 10      female    19
## # ... with 32 more rows
```

```
## Joining, by = "subject"
```

```
## # A tibble: 8,064 x 7
##    condition trial subject correct    RT gender   age
##    <fct>     <int> <fct>     <int> <dbl> <chr>  <dbl>
##  1 A             1 1             0  420. male      20
##  2 A             1 2             1  547. female    19
##  3 A             1 3             1  367. male      20
##  4 A             1 4             1  346. male      20
##  5 A             1 5             1  434. male      18
##  6 A             1 6             0  756. male      20
##  7 A             1 7             0  493. male      20
##  8 A             1 8             1  485. male      20
##  9 A             1 9             0  302. male      19
## 10 A             1 10            1  351. female    19
## # ... with 8,054 more rows
```

5 Case Study I: (Rucci group)

5.1 Design

5.2 Loading data from .csv file

```
## Parsed with column specification:
## cols(
##   SMI_timestamp = col_double(),
##   Unity_timestamp = col_double(),
##   head_pos_x = col_double(),
##   head_pos_y = col_double(),
##   head_pos_z = col_double(),
##   head_rot_x = col_double(),
##   head_rot_y = col_double(),
##   head_rot_z = col_double(),
##   head_rot_w = col_double(),
##   gaze_l_x = col_double(),
##   gaze_l_y = col_double(),
##   gaze_l_z = col_double(),
##   gaze_r_x = col_double(),
##   gaze_r_y = col_double(),
##   gaze_r_z = col_double(),
##   session_idx = col_double(),
##   stimulus_deg = col_double()
## )
```

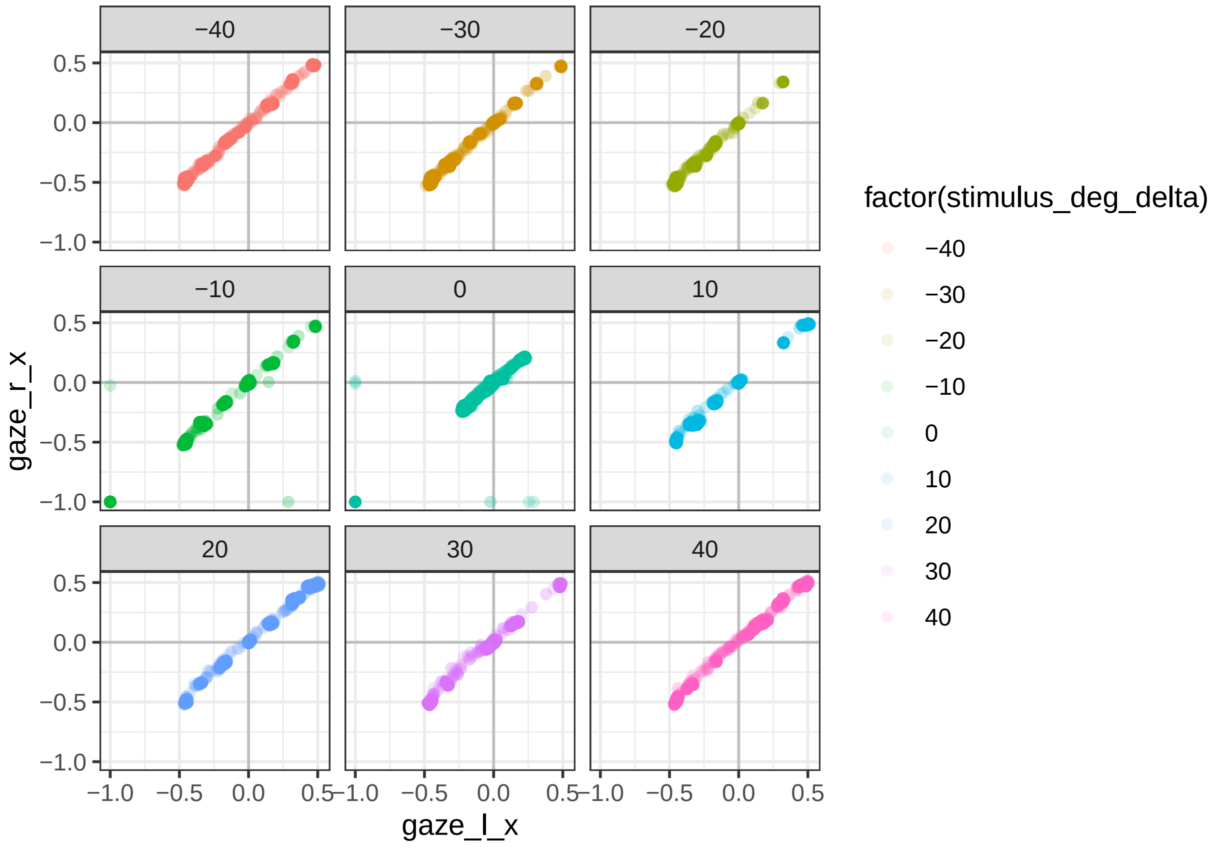
```
##  SMI_timestamp       Unity_timestamp     head_pos_x        head_pos_y
##  Min.   :0.000e+00   Min.   :  1.515   Min.   :-0.4457   Min.   :0.9623
##  1st Qu.:4.022e+10   1st Qu.: 44.985   1st Qu.:-0.4442   1st Qu.:0.9681
##  Median :7.076e+10   Median : 75.524   Median :-0.4434   Median :0.9685
##  Mean   :7.041e+10   Mean   : 75.146   Mean   :-0.4433   Mean   :0.9686
##  3rd Qu.:1.013e+11   3rd Qu.:106.064   3rd Qu.:-0.4427   3rd Qu.:0.9690
##  Max.   :1.319e+11   Max.   :136.625   Max.   :-0.4330   Max.   :0.9721
##    head_pos_z        head_rot_x          head_rot_y
##  Min.   :-0.09231   Min.   :-0.066826   Min.   :0.6681
##  1st Qu.:-0.08253   1st Qu.:-0.055539   1st Qu.:0.6983
##  Median :-0.07953   Median :-0.047245   Median :0.6990
##  Mean   :-0.08053   Mean   :-0.049261   Mean   :0.6987
##  3rd Qu.:-0.07828   3rd Qu.:-0.044043   3rd Qu.:0.6996
##  Max.   :-0.07600   Max.   : 0.007768   Max.   :0.7042
##    head_rot_z          head_rot_w       gaze_l_x
##  Min.   :-0.004449   Min.   :0.7089   Min.   :-1.000000
##  1st Qu.: 0.039071   1st Qu.:0.7113   1st Qu.:-0.342505
##  Median : 0.039743   Median :0.7122   Median :-0.159412
##  Mean   : 0.039393   Mean   :0.7125   Mean   :-0.126632
##  3rd Qu.: 0.040691   3rd Qu.:0.7133   3rd Qu.: 0.007257
##  Max.   : 0.044881   Max.   :0.7440   Max.   : 0.514591
##     gaze_l_y           gaze_l_z         gaze_r_x
##  Min.   :-1.000000   Min.   :-1.0000   Min.   :-1.000000
##  1st Qu.:-0.015963   1st Qu.: 0.8959   1st Qu.:-0.355706
##  Median :-0.005040   Median : 0.9719   Median :-0.157496
##  Mean   :-0.033892   Mean   : 0.9016   Mean   :-0.135263
##  3rd Qu.: 0.004665   3rd Qu.: 0.9947   3rd Qu.: 0.009525
##  Max.   : 0.096892   Max.   : 1.0000   Max.   : 0.516798
##     gaze_r_y            gaze_r_z        session_idx     stimulus_deg
##  Min.   :-1.0000000   Min.   :-1.0000   Min.   : 0.00   Min.   :-30.000
##  1st Qu.:-0.0200058   1st Qu.: 0.8845   1st Qu.: 9.00   1st Qu.:-20.000
##  Median :-0.0081834   Median : 0.9708   Median :39.00   Median :  0.000
##  Mean   :-0.0360727   Mean   : 0.8959   Mean   :41.24   Mean   : -6.309
##  3rd Qu.: 0.0000424   3rd Qu.: 0.9955   3rd Qu.:70.00   3rd Qu.:  0.000
##  Max.   : 0.1120100   Max.   : 1.0000   Max.   :96.00   Max.   : 30.000
```

```
## Joining, by = c("session_idx", "stimulus_deg")
```

```
##  SMI_timestamp       Unity_timestamp     head_pos_x        head_pos_y
##  Min.   :0.000e+00   Min.   :  1.515   Min.   :-0.4457   Min.   :0.9623
##  1st Qu.:4.022e+10   1st Qu.: 44.985   1st Qu.:-0.4442   1st Qu.:0.9681
##  Median :7.076e+10   Median : 75.524   Median :-0.4434   Median :0.9685
##  Mean   :7.041e+10   Mean   : 75.146   Mean   :-0.4433   Mean   :0.9686
##  3rd Qu.:1.013e+11   3rd Qu.:106.064   3rd Qu.:-0.4427   3rd Qu.:0.9690
##  Max.   :1.319e+11   Max.   :136.625   Max.   :-0.4330   Max.   :0.9721
##
##    head_pos_z        head_rot_x          head_rot_y
##  Min.   :-0.09231   Min.   :-0.066826   Min.   :0.6681
##  1st Qu.:-0.08253   1st Qu.:-0.055539   1st Qu.:0.6983
##  Median :-0.07953   Median :-0.047245   Median :0.6990
##  Mean   :-0.08053   Mean   :-0.049261   Mean   :0.6987
##  3rd Qu.:-0.07828   3rd Qu.:-0.044043   3rd Qu.:0.6996
##  Max.   :-0.07600   Max.   : 0.007768   Max.   :0.7042
##
##    head_rot_z          head_rot_w       gaze_l_x
##  Min.   :-0.004449   Min.   :0.7089   Min.   :-1.000000
##  1st Qu.: 0.039071   1st Qu.:0.7113   1st Qu.:-0.342505
##  Median : 0.039743   Median :0.7122   Median :-0.159412
##  Mean   : 0.039393   Mean   :0.7125   Mean   :-0.126632
##  3rd Qu.: 0.040691   3rd Qu.:0.7133   3rd Qu.: 0.007257
##  Max.   : 0.044881   Max.   :0.7440   Max.   : 0.514591
##
##     gaze_l_y            gaze_l_z          gaze_r_x
##  Min.   :-1.000000   Min.   :-1.0000   Min.   :-1.000000
##  1st Qu.:-0.015963   1st Qu.: 0.8959   1st Qu.:-0.355706
##  Median :-0.005040   Median : 0.9719   Median :-0.157496
##  Mean   :-0.033892   Mean   : 0.9016   Mean   :-0.135263
##  3rd Qu.: 0.004665   3rd Qu.: 0.9947   3rd Qu.: 0.009525
##  Max.   : 0.096892   Max.   : 1.0000   Max.   : 0.516798
##
##     gaze_r_y             gaze_r_z        session_idx     stimulus_deg
##  Min.   :-1.0000000   Min.   :-1.0000   Min.   : 0.00   Min.   :-30.000
##  1st Qu.:-0.0200058   1st Qu.: 0.8845   1st Qu.: 9.00   1st Qu.:-20.000
##  Median :-0.0081834   Median : 0.9708   Median :39.00   Median :  0.000
##  Mean   :-0.0360727   Mean   : 0.8959   Mean   :41.24   Mean   : -6.309
##  3rd Qu.: 0.0000424   3rd Qu.: 0.9955   3rd Qu.:70.00   3rd Qu.:  0.000
##  Max.   : 0.1120100   Max.   : 1.0000   Max.   :96.00   Max.   : 30.000
##
##  prev_stimulus_deg stimulus_deg_delta
##  Min.   :-30.000   0      :5081
##  1st Qu.:-20.000   -10    :3659
##  Median :  0.000   -20    :2756
##  Mean   : -5.976   40     :2752
##  3rd Qu.:  0.000   -40    :2748
##  Max.   : 30.000   -30    :2748
##                    (Other):8236
```

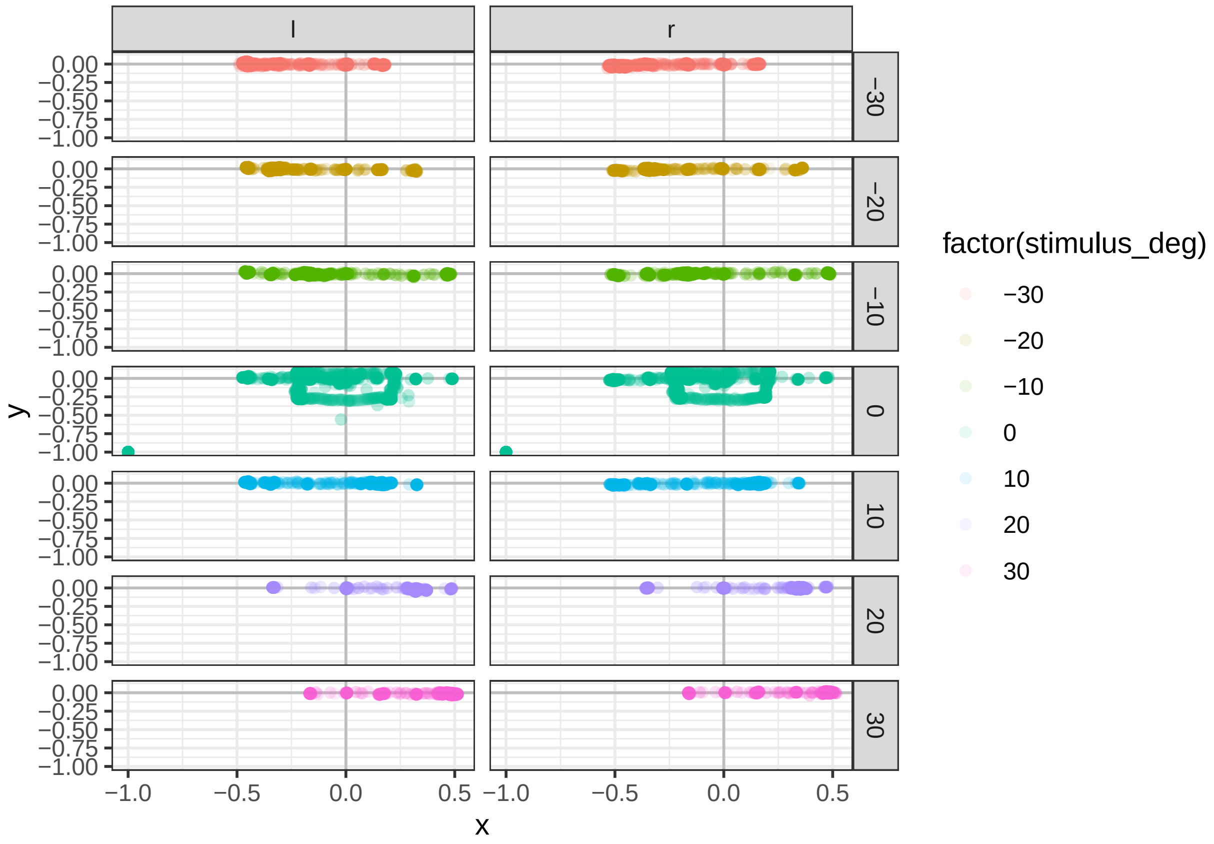
5.3 Plotting the data

5.3.1 Plotting left and right eye's x coordinate by stimulus degree



5.3.2 Plotting left and right eye's x and y coordinates by stimulus degree

For this we first need to transform that data so that we have separate rows for gaze information about the left and right eye. Then we can plot the data in a way very similar to the plot in the previous section.



6 Case Study II: visual decision-making (Haefner group)

This group seeks to replicate Herce Castañón et al. (2019).

6.1 Design

The design of the present study crossed two levels of contrast (Low = 15%, High = 60%), 3 levels of variance (0, 4, 10), and how the trials in the block were cued (L = left, R = right, N = uncued), for a total of 2 x 3 x 3 = 18 within-subject conditions.

6.2 Loading data from MatLab

The data are stored in a MatLab (.mat) file. The file contains one matrix with fields: participant, exp(eriment), stimuli and response. Within each field, there is further information. The important information seems to be in the response field. Some of the important parts include:

- responseRight: the response of the subject (0 for CCW, 1 for CW, w.r.t horizontal)
- correct: what the correct answer is (0 for CCW, 1 for CW, w.r.t horizontal)
- accuracy: whether subject got the correct answer (1) or not (0)
- reaction time: time in seconds the subject took to answer
- confidence: whether the subject was confident in their answer (1) or not (-1)
- cue: whether the cue on that trials is left (-1), right (1), or no cue (0)
- contrast: the contrast of the gabor patch on that trial
- variance: variability in the orientation of gratings of gabor patches on that trial

Fig. 2

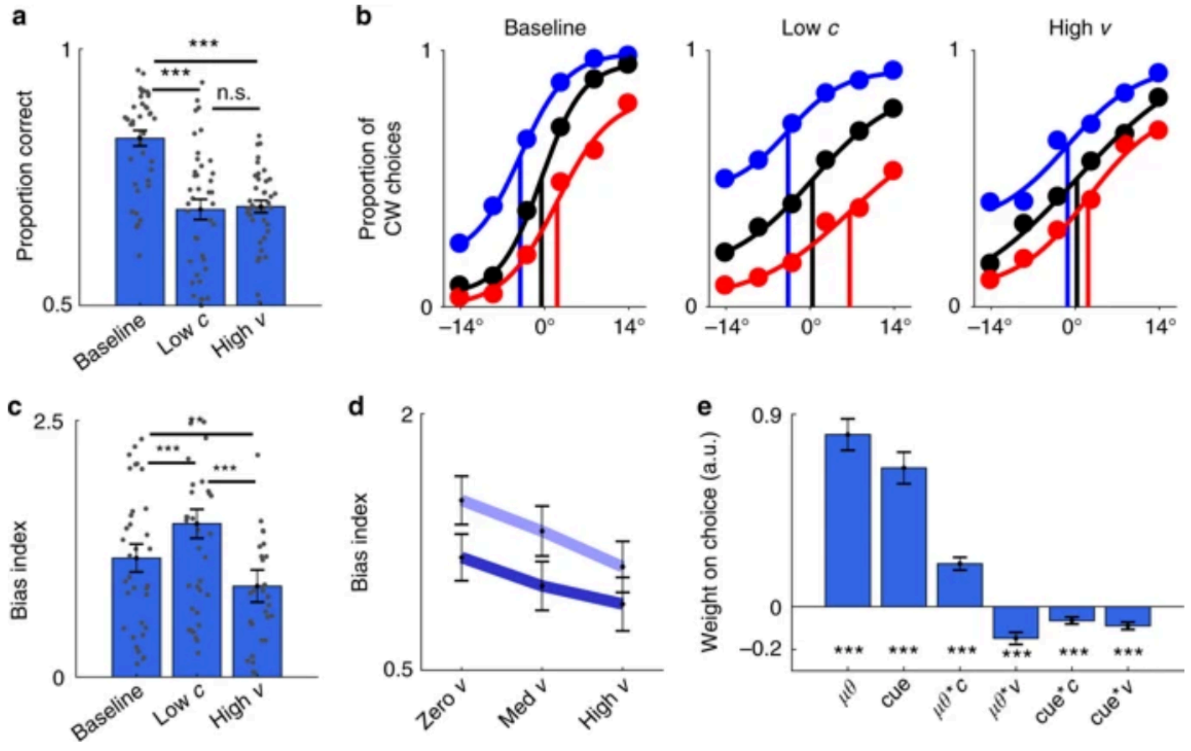


Figure 3: Figure 2 from Herce Castañón et al. (2019)

- isCuedBlock: whether a block (of trials) will have cues (1) or no cues (0)

```r
# Load a matlab file and extract the "data" matrix out of it
d.haefner = readMat("./data/Haefner/uncertaintyV1-subject18-1-EarlyQuit.mat")
d.haefner = d.haefner[["data"]][,,1][["response"]][,,1]
d.haefner[["trueOrientaions"]] <- NULL

# Look at what we've imported.
# NB: str() gives your the structure of an R object
str(d.haefner)

d.haefner %<>%
  map(.f = function(x) c(x)) %>%
  as_tibble()

# The data we have are preliminary pilot data from one of the
# experimenters, and that run did contain all trials. We omit
# all the trials with missing information.
d.haefner %<>%
  na.omit()

# Add the definition of the three conditions of interest in the
# original paper
d.haefner %<>%
  mutate(
    condition = case_when(
        variance == min(variance) & contrast == max(contrast) ~ "baseline",
        variance == max(variance) & contrast == max(contrast) ~ "high variance",
        variance == min(variance) & contrast == min(contrast) ~ "low contrast",
        T ~ ""
    )
  )
```

Now that we've imported the data into an R data frame (or *tibble*), let's have a look at it. First, we can get a general idea of the data by using str() (for structure) or print():

```
## # A tibble: 864 x 12
##    randSeed responseRight correct accuracy reactionTime confidence
##       <dbl>         <dbl>   <dbl>    <dbl>        <dbl>      <dbl>
##  1   2.20e8             0       1        0        0.708         -1
##  2   2.20e8             1       0        0        0.609         -1
##  3   2.20e8             0       1        0        1.73           0
##  4   2.20e8             1       0        0        0.684          0
##  5   2.20e8             0       1        0        0.550         -1
##  6   2.20e8             1       1        1        0.565         -1
##  7   2.20e8             0       1        0        0.492         -1
##  8   2.20e8             0       0        1        0.994         -1
##  9   2.20e8             0       1        0        0.872          0
## 10   2.20e8             0       1        0        0.782          1
## # ... with 854 more rows, and 6 more variables: isCuedBlock <dbl>,
## #   cue <dbl>, orientationMean <dbl>, contrast <dbl>, variance <dbl>,
## #   condition <chr>
```

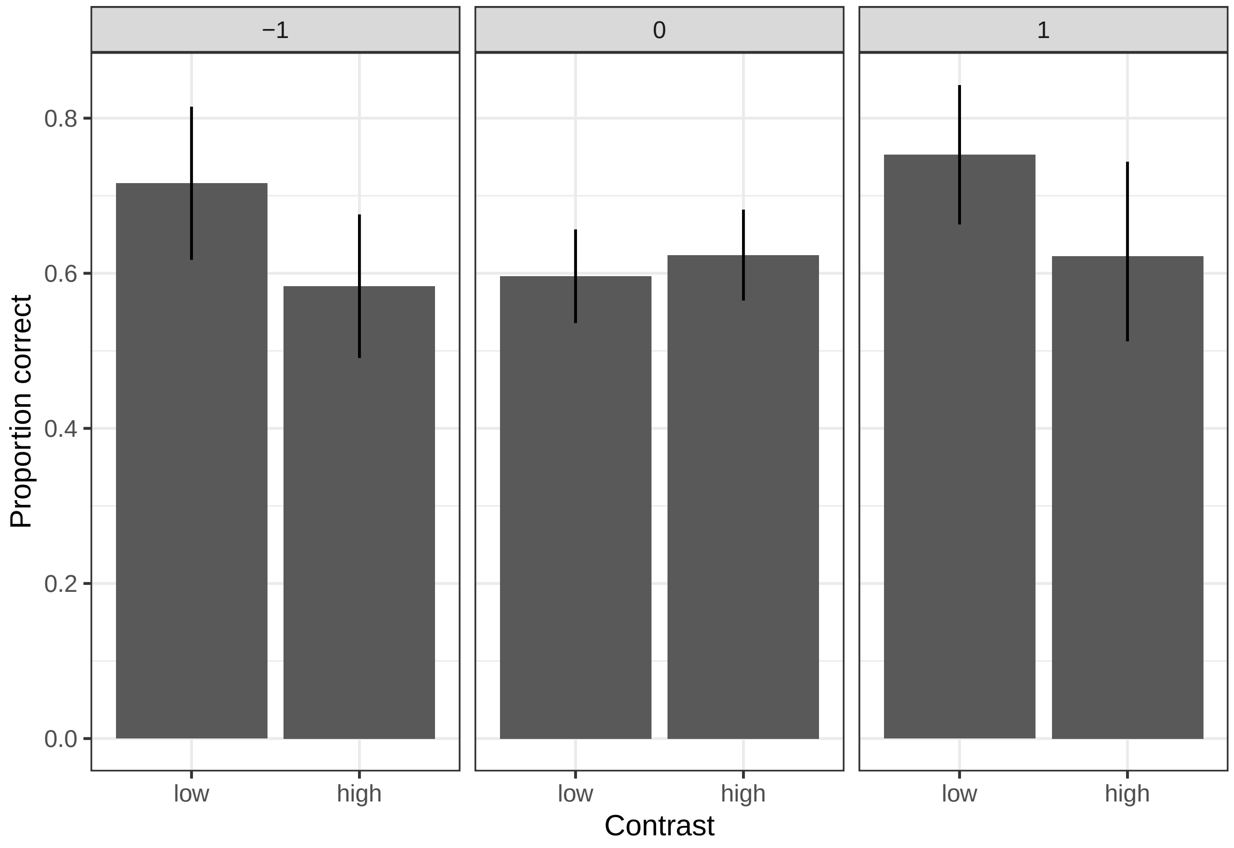
To instead get a summary of the data:

```
##     randSeed         responseRight      correct          accuracy
##  Min.   :220286057   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:220377520   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :220475950   Median :1.000   Median :1.0000   Median :1.0000
##  Mean   :220480775   Mean   :0.559   Mean   :0.5081   Mean   :0.6319
##  3rd Qu.:220587872   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :220674325   Max.   :1.000   Max.   :1.0000   Max.   :1.0000
##   reactionTime      confidence       isCuedBlock          cue
##  Min.   :0.1983   Min.   :-1.0000   Min.   :0.0000   Min.   :-1.00000
##  1st Qu.:0.4322   1st Qu.:-1.0000   1st Qu.:0.0000   1st Qu.: 0.00000
##  Median :0.5677   Median :-1.0000   Median :0.0000   Median : 0.00000
##  Mean   :0.6970   Mean   :-0.5289   Mean   :0.4167   Mean   :-0.02083
##  3rd Qu.:0.8143   3rd Qu.: 0.0000   3rd Qu.:1.0000   3rd Qu.: 0.00000
##  Max.   :2.9974   Max.   : 1.0000   Max.   :1.0000   Max.   : 1.00000
##  orientationMean        contrast         variance      condition
##  Min.   :-26.67646   Min.   :0.1500   Min.   : 0.00   Length:864
##  1st Qu.: -5.95720   1st Qu.:0.1500   1st Qu.: 4.00   Class :character
##  Median :  0.15682   Median :0.1500   Median : 4.00   Mode  :character
##  Mean   :  0.08092   Mean   :0.3734   Mean   : 4.66
##  3rd Qu.:  5.95273   3rd Qu.:0.6000   3rd Qu.:10.00
##  Max.   : 25.98183   Max.   :0.6000   Max.   :10.00
```

6.3 Figure 2 from Herce Castañón et al.

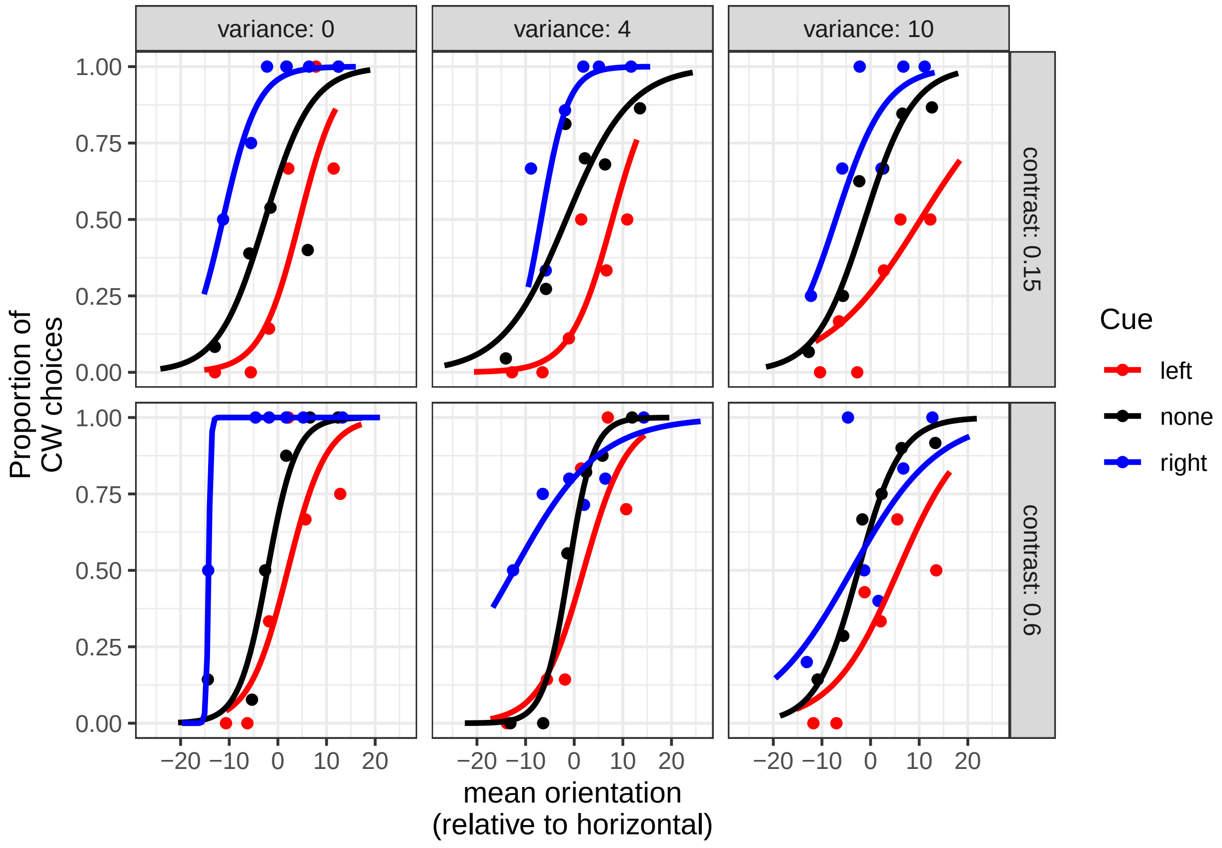
6.3.1 Panel A

We begin by plotting the proportion of correct choices for *all* conditions:

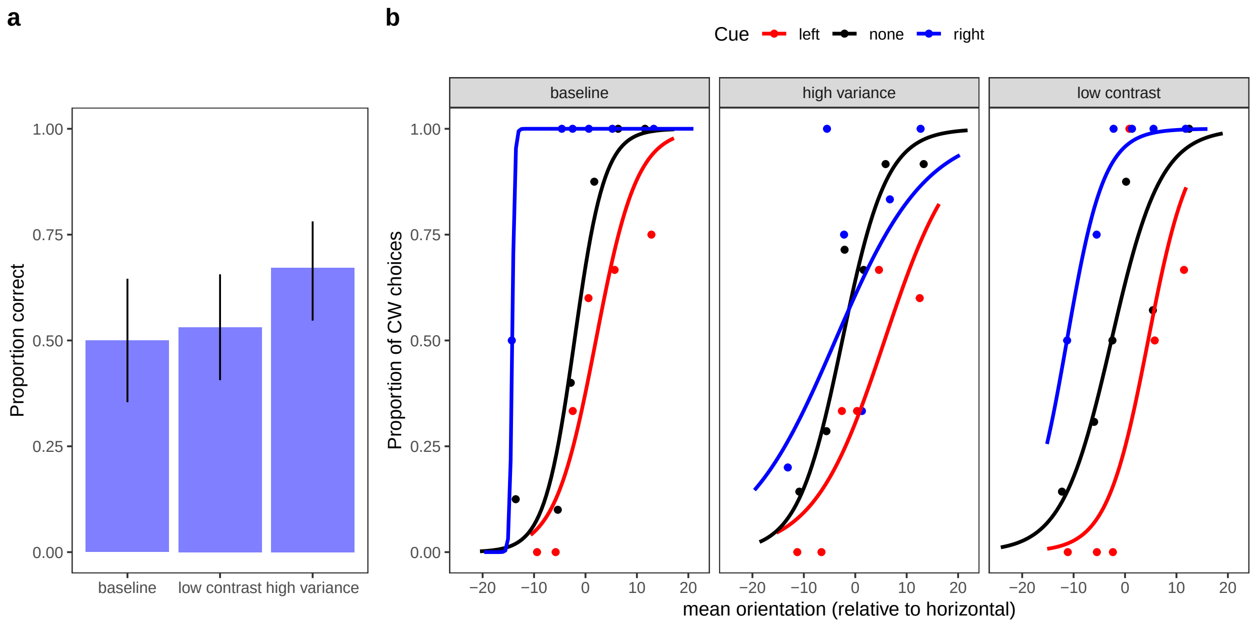


6.3.2 Panel B

We begin by plotting the proportion of CW choices for *all* conditions:



6.3.3 Panel A and B together



7 Case Study III: (Huxlin group)

7.1 Load data from Excel files

8 Session info

```
## - Session info ----------------------------------------------------------
##  setting  value
##  version  R version 3.6.0 (2019-04-26)
##  os       macOS High Sierra 10.13.6
##  system   x86_64, darwin15.6.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2019-11-06
##
## - Packages --------------------------------------------------------------
##  package      * version  date       lib source
##  acepack        1.4.1    2016-10-29 [1] CRAN (R 3.6.0)
##  assertthat     0.2.1    2019-03-21 [1] CRAN (R 3.6.0)
##  backports      1.1.5    2019-10-02 [1] CRAN (R 3.6.0)
##  base64enc      0.1-3    2015-07-28 [1] CRAN (R 3.6.0)
##  broom          0.5.2    2019-04-07 [1] CRAN (R 3.6.0)
##  callr          3.3.2    2019-09-22 [1] CRAN (R 3.6.0)
##  cellranger     1.1.0    2016-07-27 [1] CRAN (R 3.6.0)
##  checkmate      1.9.4    2019-07-04 [1] CRAN (R 3.6.0)
##  cli            1.1.0    2019-03-19 [1] CRAN (R 3.6.0)
##  cluster        2.1.0    2019-06-19 [1] CRAN (R 3.6.0)
##  colorspace     1.4-1    2019-03-18 [1] CRAN (R 3.6.0)
##  cowplot      * 1.0.0    2019-07-11 [1] CRAN (R 3.6.0)
##  crayon         1.3.4    2017-09-16 [1] CRAN (R 3.6.0)
##  data.table     1.12.6   2019-10-18 [1] CRAN (R 3.6.0)
##  desc           1.2.0    2018-05-01 [1] CRAN (R 3.6.0)
##  devtools       2.2.1    2019-09-24 [1] CRAN (R 3.6.0)
##  digest         0.6.22   2019-10-21 [1] CRAN (R 3.6.0)
##  dplyr        * 0.8.3    2019-07-04 [1] CRAN (R 3.6.0)
##  ellipsis       0.3.0    2019-09-20 [1] CRAN (R 3.6.0)
##  evaluate       0.14     2019-05-28 [1] CRAN (R 3.6.0)
##  fansi          0.4.0    2018-10-05 [1] CRAN (R 3.6.0)
##  forcats      * 0.4.0    2019-02-17 [1] CRAN (R 3.6.0)
##  foreign        0.8-72   2019-08-02 [1] CRAN (R 3.6.0)
##  Formula        1.2-3    2018-05-03 [1] CRAN (R 3.6.0)
##  fs             1.3.1    2019-05-06 [1] CRAN (R 3.6.0)
##  generics       0.0.2    2018-11-29 [1] CRAN (R 3.6.0)
##  ggplot2      * 3.2.1    2019-08-10 [1] CRAN (R 3.6.0)
##  glue           1.3.1    2019-03-12 [1] CRAN (R 3.6.0)
##  gridExtra      2.3      2017-09-09 [1] CRAN (R 3.6.0)
##  gtable         0.3.0    2019-03-25 [1] CRAN (R 3.6.0)
##  haven          2.1.1    2019-07-04 [1] CRAN (R 3.6.0)
##  Hmisc          4.2-0    2019-01-26 [1] CRAN (R 3.6.0)
##  hms            0.5.2    2019-10-30 [1] CRAN (R 3.6.0)
##  htmlTable      1.13.2   2019-09-22 [1] CRAN (R 3.6.0)
##  htmltools      0.4.0    2019-10-04 [1] CRAN (R 3.6.0)
##  htmlwidgets    1.5.1    2019-10-08 [1] CRAN (R 3.6.0)
##  httr           1.4.1    2019-08-05 [1] CRAN (R 3.6.0)
##  jsonlite       1.6      2018-12-07 [1] CRAN (R 3.6.0)
##  knitr          1.25     2019-09-18 [1] CRAN (R 3.6.0)
##  labeling       0.3      2014-08-23 [1] CRAN (R 3.6.0)
##  lattice        0.20-38  2018-11-04 [1] CRAN (R 3.6.0)
##  latticeExtra   0.6-28   2016-02-09 [1] CRAN (R 3.6.0)
##  lazyeval       0.2.2    2019-03-15 [1] CRAN (R 3.6.0)
##  lifecycle      0.1.0    2019-08-01 [1] CRAN (R 3.6.0)
##  lubridate      1.7.4    2018-04-11 [1] CRAN (R 3.6.0)
##  magrittr     * 1.5      2014-11-22 [1] CRAN (R 3.6.0)
##  Matrix         1.2-17   2019-03-22 [1] CRAN (R 3.6.0)
##  memoise        1.1.0    2017-04-21 [1] CRAN (R 3.6.0)
##  modelr         0.1.5    2019-08-08 [1] CRAN (R 3.6.0)
##  munsell        0.5.0    2018-06-12 [1] CRAN (R 3.6.0)
##  nlme           3.1-141  2019-08-01 [1] CRAN (R 3.6.0)
##  nnet           7.3-12   2016-02-02 [1] CRAN (R 3.6.0)
##  openxlsx     * 4.1.2    2019-10-29 [1] CRAN (R 3.6.0)
##  pillar         1.4.2    2019-06-29 [1] CRAN (R 3.6.0)
##  pkgbuild       1.0.6    2019-10-09 [1] CRAN (R 3.6.0)
##  pkgconfig      2.0.3    2019-09-22 [1] CRAN (R 3.6.0)
##  pkgload        1.0.2    2018-10-29 [1] CRAN (R 3.6.0)
##  plotly       * 4.9.0    2019-04-10 [1] CRAN (R 3.6.0)
##  plyr           1.8.4    2016-06-08 [1] CRAN (R 3.6.0)
##  prettyunits    1.0.2    2015-07-13 [1] CRAN (R 3.6.0)
##  processx       3.4.1    2019-07-18 [1] CRAN (R 3.6.0)
##  ps             1.3.0    2018-12-21 [1] CRAN (R 3.6.0)
##  purrr        * 0.3.3    2019-10-18 [1] CRAN (R 3.6.0)
##  R.matlab     * 3.6.2    2018-09-27 [1] CRAN (R 3.6.0)
##  R.methodsS3    1.7.1    2016-02-16 [1] CRAN (R 3.6.0)
##  R.oo           1.22.0   2018-04-22 [1] CRAN (R 3.6.0)
##  R.utils        2.9.0    2019-06-13 [1] CRAN (R 3.6.0)
##  R6             2.4.0    2019-02-14 [1] CRAN (R 3.6.0)
##  RColorBrewer   1.1-2    2014-12-07 [1] CRAN (R 3.6.0)
##  Rcpp           1.0.2    2019-07-25 [1] CRAN (R 3.6.0)
##  readr        * 1.3.1    2018-12-21 [1] CRAN (R 3.6.0)
##  readxl         1.3.1    2019-03-13 [1] CRAN (R 3.6.0)
##  remotes        2.1.0    2019-06-24 [1] CRAN (R 3.6.0)
##  reshape2       1.4.3    2017-12-11 [1] CRAN (R 3.6.0)
##  rlang          0.4.1    2019-10-24 [1] CRAN (R 3.6.0)
##  rmarkdown      1.16     2019-10-01 [1] CRAN (R 3.6.0)
##  rpart          4.1-15   2019-04-12 [1] CRAN (R 3.6.0)
##  rprojroot      1.3-2    2018-01-03 [1] CRAN (R 3.6.0)
##  rstudioapi     0.10     2019-03-19 [1] CRAN (R 3.6.0)
##  rvest          0.3.4    2019-05-15 [1] CRAN (R 3.6.0)
##  scales         1.0.0    2018-08-09 [1] CRAN (R 3.6.0)
##  sessioninfo    1.1.1    2018-11-05 [1] CRAN (R 3.6.0)
##  stringi        1.4.3    2019-03-12 [1] CRAN (R 3.6.0)
##  stringr      * 1.4.0    2019-02-10 [1] CRAN (R 3.6.0)
##  survival       2.44-1.1 2019-04-01 [1] CRAN (R 3.6.0)
##  testthat       2.2.1    2019-07-25 [1] CRAN (R 3.6.0)
##  tibble       * 2.1.3    2019-06-06 [1] CRAN (R 3.6.0)
##  tidyr        * 1.0.0    2019-09-11 [1] CRAN (R 3.6.0)
##  tidyselect     0.2.5    2018-10-11 [1] CRAN (R 3.6.0)
##  tidyverse    * 1.2.1    2017-11-14 [1] CRAN (R 3.6.0)
##  usethis        1.5.1    2019-07-04 [1] CRAN (R 3.6.0)
##  utf8           1.1.4    2018-05-24 [1] CRAN (R 3.6.0)
##  vctrs          0.2.0    2019-07-05 [1] CRAN (R 3.6.0)
##  viridisLite    0.3.0    2018-02-01 [1] CRAN (R 3.6.0)
##  withr          2.1.2    2018-03-15 [1] CRAN (R 3.6.0)
##  xfun           0.10     2019-10-01 [1] CRAN (R 3.6.0)
##  xml2           1.2.2    2019-08-09 [1] CRAN (R 3.6.0)
##  yaml           2.2.0    2018-07-25 [1] CRAN (R 3.6.0)
##  zeallot        0.1.0    2018-01-28 [1] CRAN (R 3.6.0)
##  zip            2.0.4    2019-09-01 [1] CRAN (R 3.6.0)
##
## [1] /Library/Frameworks/R.framework/Versions/3.6/Resources/library
```