Case Study Migraine

Migraine case study

In december 2016 the company “BB” launched the app “MigraineTracker”. This app was developed to collect data from people suffering from migraine. Users are able to create a record in this app after they had a sick headache. So far this app has 4485 users and they created int total 22645 entries in the year 2017.

As we are not able to analyze all the 22645 records manually, we are glad about the opportunity to use R to get some information about the data produced in 2017.

With this case study you should practice the concepts you learned so far. At the same time a few new features are shown within this exercise that will give you an idea of additional opportunities provided by R. The content of the dataset used here was inspired by some basic knowledge about migraine and existing apps. However, this is an artificial dataset, all values were sampled randomly.

Loading the data

So far we always loaded the datasets from local files. However, `read_csv` for example can also take an URL as filepath to retrieve files from the Internet.

```
  records <- read_csv("https://cbdm.uni-mainz.de/files/2019/03/records.csv.gz")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   user.ID = col_double(),
##   month = col_double(),
##   day = col_double(),
##   beta.blocker = col_character(),
##   pain.level = col_double(),
##   duration = col_double(),
##   visual.disorder = col_character()
## )
```

```
  records
```

```
## # A tibble: 22,647 x 8
##       X1 user.ID month   day beta.blocker pain.level duration
##    <dbl>   <dbl> <dbl> <dbl> <chr>             <dbl>    <dbl>
##  1     1       4     6     4 yes                   4        1
##  2     2    3286     1     4 no                    3        1
##  3     3    3048     5    29 yes                   3        3
##  4     4    3290     4    28 yes                   4        2
##  5     5    4363     4    17 yes                   3        1
##  6     6    3332     6     6 yes                   3        5
##  7     7     851    12     8 no                    5        7
##  8     8    4247     6    11 yes                   4        2
##  9     9    1674     4    28 yes                   3        1
## 10    10     616     8    17 yes                   4        4
## # ... with 22,637 more rows, and 1 more variable: visual.disorder <chr>
```

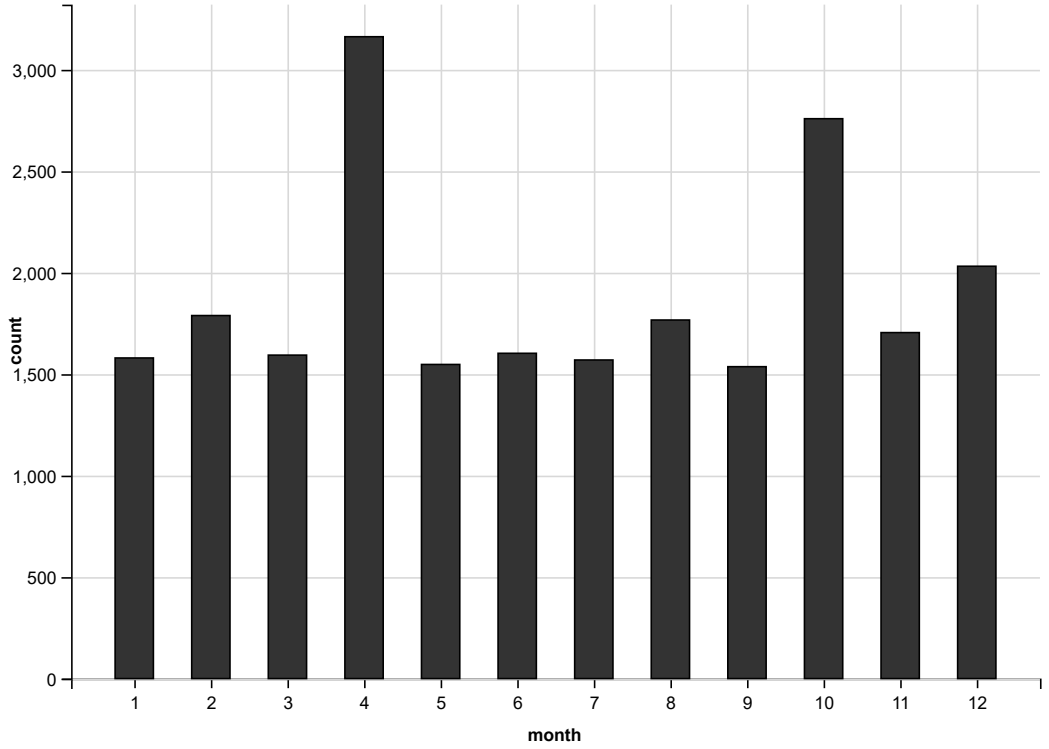
A record consists basically of the users ID, the date and some information about the sick headache.

- **user.ID:** A numeric ID for each subject.
- **month:** The month of the date. 1 is Jan, 12 is Dec.
- **day:** The day of the date.
- **beta.blocker:** “yes” or “no” whether beta blockers were used or not.
- **pain.level:** A value between 1 and 5, while 5 means strongest pain.
- **duration:** The duration of the sick headache in days.
- **visual.disorder:** “yes” or “no” whether the patient had visual disorders.

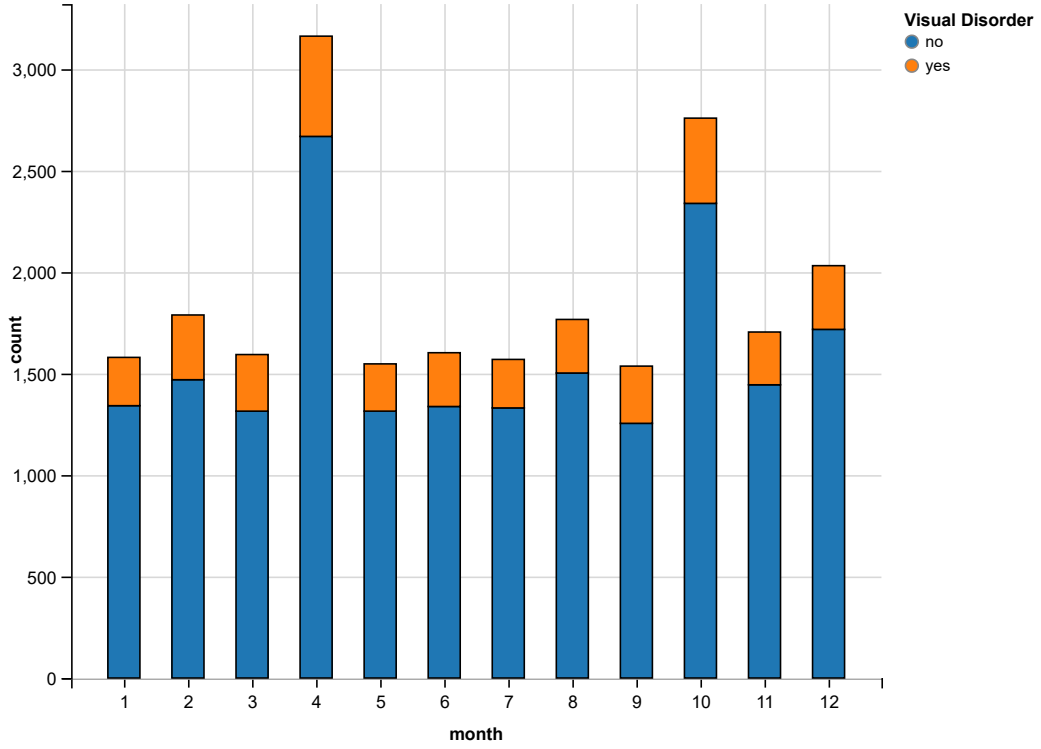
Remember, you must load the packages `tidyverse` and `ggvis` .

Exercise 6.1 - The records

1. Column `x` looks like being not interesting. Kick it out!
2. Plot a histogram showing the count of records for the different months. What is the month with the most records?



3. Include also information about visual disorder to the bar plot. Fill the bars with information from the column about visual disorders and add a legend. Is there a difference between the months with respect to this information?



4. The user support reported problems with a certain user account. It is about the user with ID `18`. Display all records from user `18`. The records are not sorted, but isn't there something strange?

```
## # A tibble: 9 x 7
##   user.ID month   day beta.blocker pain.level duration visual.disorder
##     <dbl> <dbl> <dbl> <chr>             <dbl>    <dbl> <chr>
## 1      18     6     3 yes                   1        1 no
## 2      18     4    12 yes                   1        2 no
## 3      18     1    19 yes                   5        2 no
## 4      18    12    31 yes                   1        3 no
## 5      18    11     7 yes                   2        2 no
## 6      18     8     1 no                    5        6 no
## 7      18    11     7 yes                   5        5 no
## 8      18     6    12 yes                   4        3 no
## 9      18     6    12 yes                   4        1 no
```

Exercise 6.2 - The users

So far we don't have any information about the users. Please load another dataset from this URL "https://cbdm.uni-mainz.de/files/2019/03/users.csv.gz (https://cbdm.uni-mainz.de/files/2019/03/users.csv.gz)" and name it users.

```
## # A tibble: 4,485 x 3
##    user.ID error           parent.suffers
##      <dbl> <chr>           <chr>
##  1       1 40-50/44;male   yes
##  2       2 60-70/68;male   yes
##  3       3 50-60/59;female yes
##  4       4 40-50/44;female yes
##  5       5 70-80/72;male   yes
##  6       6 80-90/85;male   yes
##  7       7 20-30/23;female yes
##  8       8 40-50/49;male   yes
##  9       9 30-40/34;male   yes
## 10      10 60-70/64;male   yes
## # ... with 4,475 more rows
```

As you can see there is an error column that seems to be badly formatted. The idea was to get one column for the age range (described as `from-to`), another column for the age (after `/` character), and another column for the gender (separated by `;`).

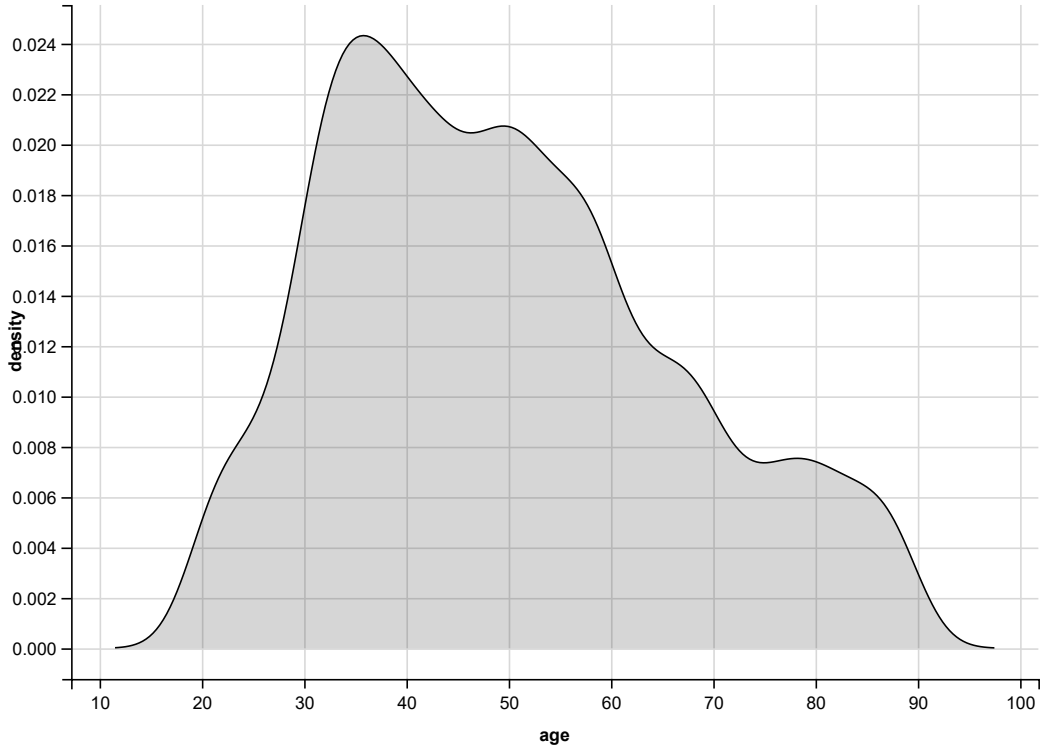
1. Please separate the column `error` into `ageRE` and `gender`.

```
## # A tibble: 4,485 x 4
##    user.ID ageRE    gender parent.suffers
##      <dbl> <chr>    <chr>  <chr>
##  1       1 40-50/44 male   yes
##  2       2 60-70/68 male   yes
##  3       3 50-60/59 female yes
##  4       4 40-50/44 female yes
##  5       5 70-80/72 male   yes
##  6       6 80-90/85 male   yes
##  7       7 20-30/23 female yes
##  8       8 40-50/49 male   yes
##  9       9 30-40/34 male   yes
## 10      10 60-70/64 male   yes
## # ... with 4,475 more rows
```

2. Now separate ageRE into `ageRange` and `age` and take care about the type of the columns.

```
## # A tibble: 4,485 x 5
##    user.ID ageRange   age gender parent.suffers
##      <dbl> <chr>    <dbl> <chr>  <chr>
##  1       1 40-50       44 male   yes
##  2       2 60-70       68 male   yes
##  3       3 50-60       59 female yes
##  4       4 40-50       44 female yes
##  5       5 70-80       72 male   yes
##  6       6 80-90       85 male   yes
##  7       7 20-30       23 female yes
##  8       8 40-50       49 male   yes
##  9       9 30-40       34 male   yes
## 10      10 60-70       64 male   yes
## # ... with 4,475 more rows
```

3. Plot the density of the counts of users for the different age. Can you find an explanation for this distribution?



Exercise 6.3 - Join

In this case study the user information was disconnected from the records. After the information was made anonymous we got it back, but now we would like to have a dataset including both the records and the user information. This will be done by using the function left_join(). Do you have an idea by which column in common we could join the two datasets?

Finally, in this case, the function will do it automatically, because there is only one opportunity.

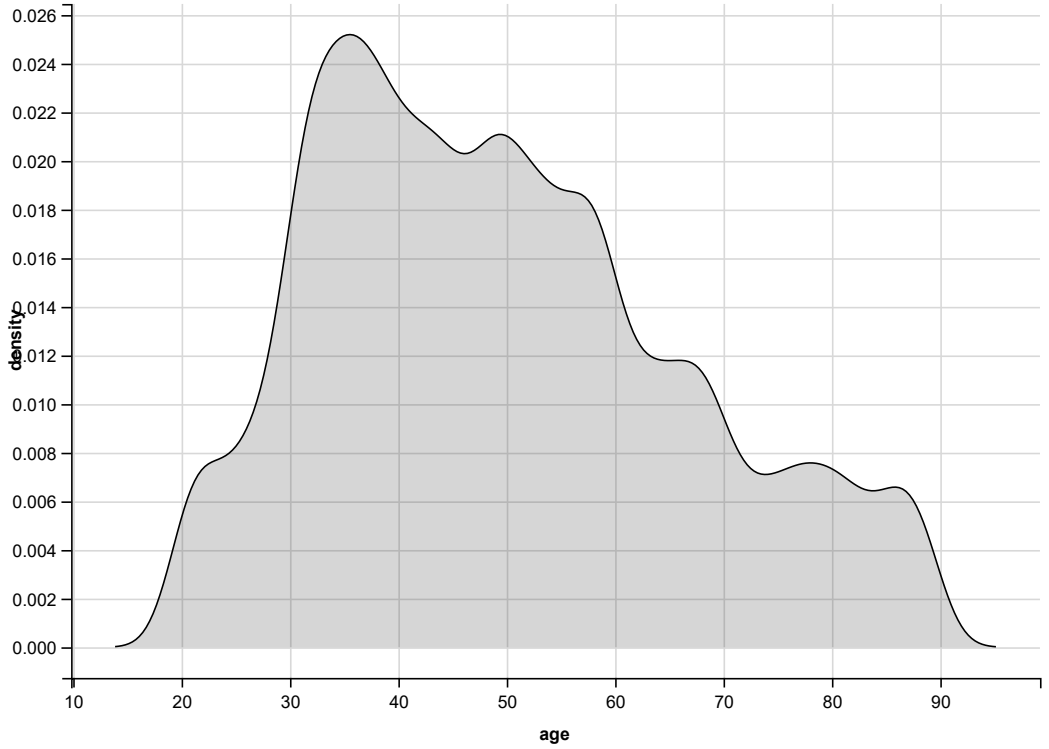
```
joined <- left_join(records, users)
```

```
## Joining, by = "user.ID"
```

```
joined
```

```
## # A tibble: 22,647 x 11
##    user.ID month   day beta.blocker pain.level duration visual.disorder
##      <dbl> <dbl> <dbl> <chr>             <dbl>    <dbl> <chr>
##  1       4     6     4 yes                   4        1 yes
##  2    3286     1     4 no                    3        1 no
##  3    3048     5    29 yes                   3        3 no
##  4    3290     4    28 yes                   4        2 no
##  5    4363     4    17 yes                   3        1 no
##  6    3332     6     6 yes                   3        5 no
##  7     851    12     8 no                    5        7 no
##  8    4247     6    11 yes                   4        2 yes
##  9    1674     4    28 yes                   3        1 yes
## 10     616     8    17 yes                   4        4 no
## # ... with 22,637 more rows, and 4 more variables: ageRange <chr>,
## #   age <dbl>, gender <chr>, parent.suffers <chr>
```

1. For the new table, plot again the density for the ages.



Exercise 6.4 - The score

Combining some basic knowledge about variables, vectors and functions we can create a score for each record in the dataset `joined`. In the following code lines you'll see how to make use of variables to get the maximum pain level and maximum duration ever recorded.

```
plMax <- max(joined %>% select(pain.level))
plMax
```

```
## [1] 5
```

1. Do something similar to get the maximum duration (dMax).

```
## [1] 7
```

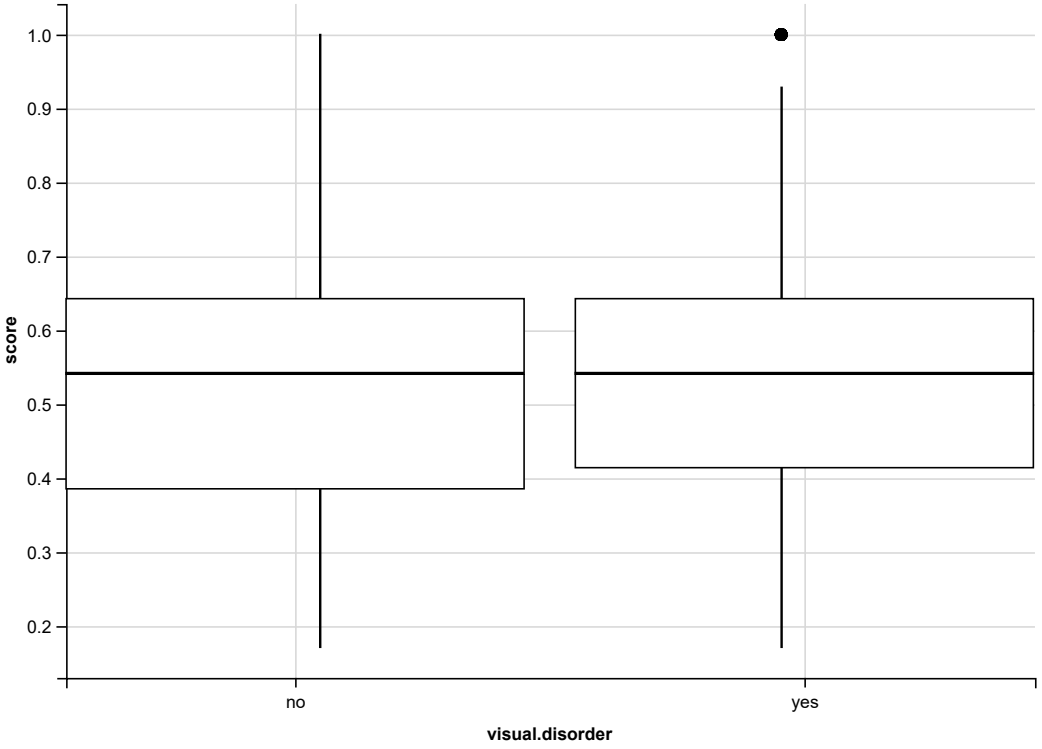
Using these two variables each value for the pain.level and for the duration can be transformed as a percentage of the maximum. We will make use of the maximums to calculate a score.

2. Add a new column to the `joined` table that contains the score for each record. The score is the average of percentage of pain level and percentage of duration.

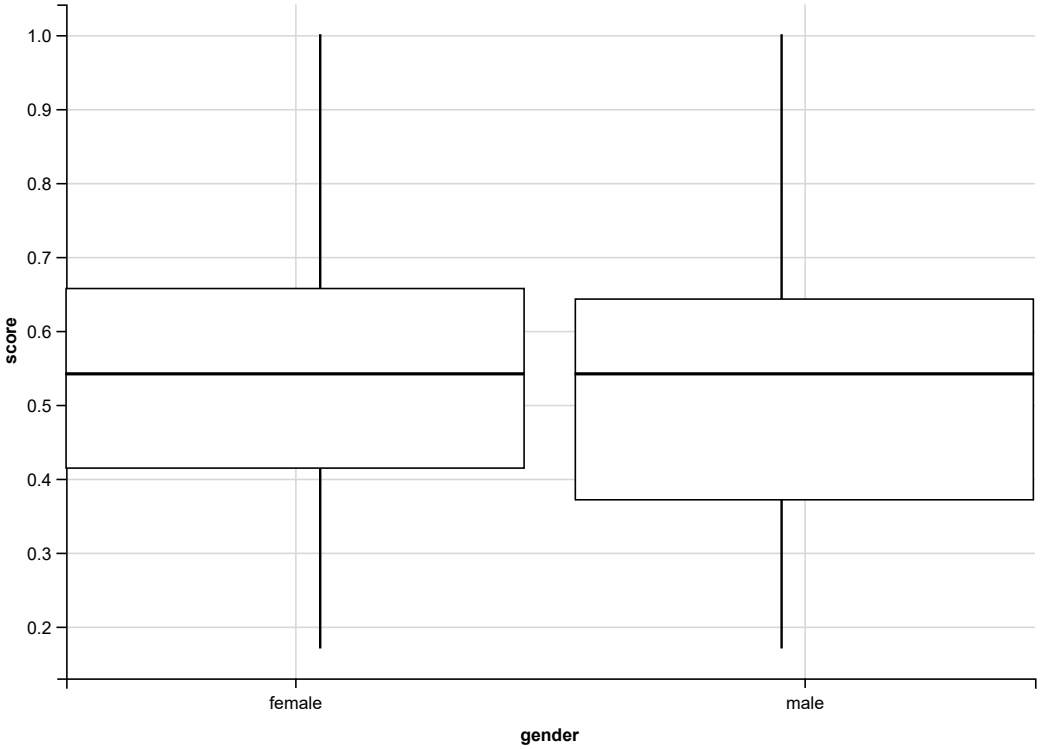
```
joined %>%
  select(user.ID, pain.level, duration, score)
```

```
## # A tibble: 22,647 x 4
##    user.ID pain.level duration score
##      <dbl>      <dbl>    <dbl> <dbl>
##  1       4          4        1 0.471
##  2    3286          3        1 0.371
##  3    3048          3        3 0.514
##  4    3290          4        2 0.543
##  5    4363          3        1 0.371
##  6    3332          3        5 0.657
##  7     851          5        7 1
##  8    4247          4        2 0.543
##  9    1674          3        1 0.371
## 10     616          4        4 0.686
## # ... with 22,637 more rows
```

3. Show the boxplot of the score with visual disorder on the x axis.



4. Show the boxplot of the score with gender on the x axis.



Here we see a small difference between female and male. Now the question is whether this difference is significant. As the number of records from female and male are not equal we should use the Wilcoxon's test to check whether the values from these two sets are significantly different. R provides the function `wilcox.test` that can be used for paired and unpaired tests. The default setting is unpaired, thus in our case it is as simple as this:

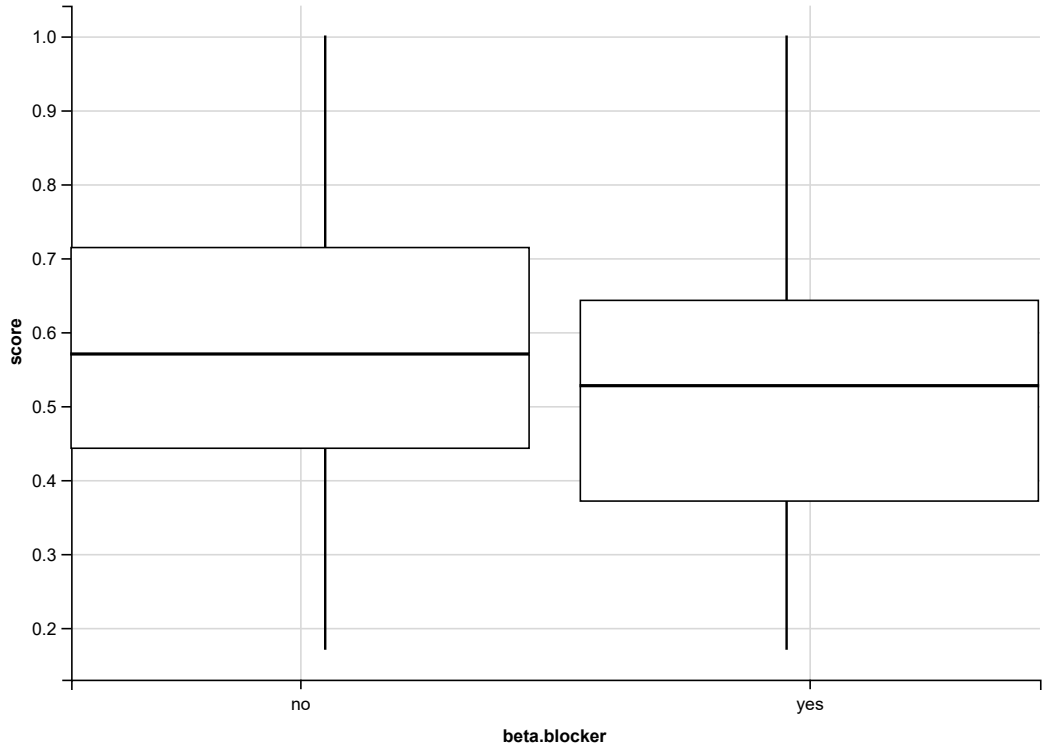
```
wilcox.test(score~gender, data=joined)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  score by gender
## W = 55594000, p-value = 7.54e-09
## alternative hypothesis: true location shift is not equal to 0
```

As the p-value is way lower than the frequently used thresholds 5%, 3%, or 1% we can conclude that the score significantly differs between female and male.

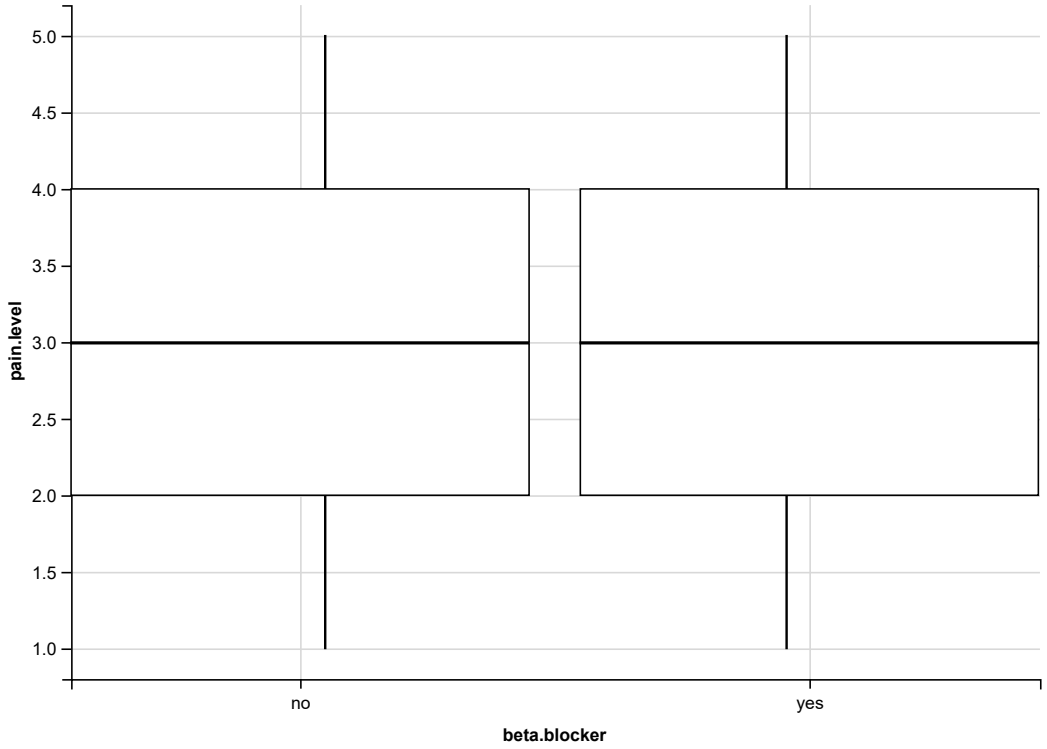
Exercise 6.5 - A detailed look at the beta blockers

1. Show the boxplot of the score with beta blocker on the x axis.

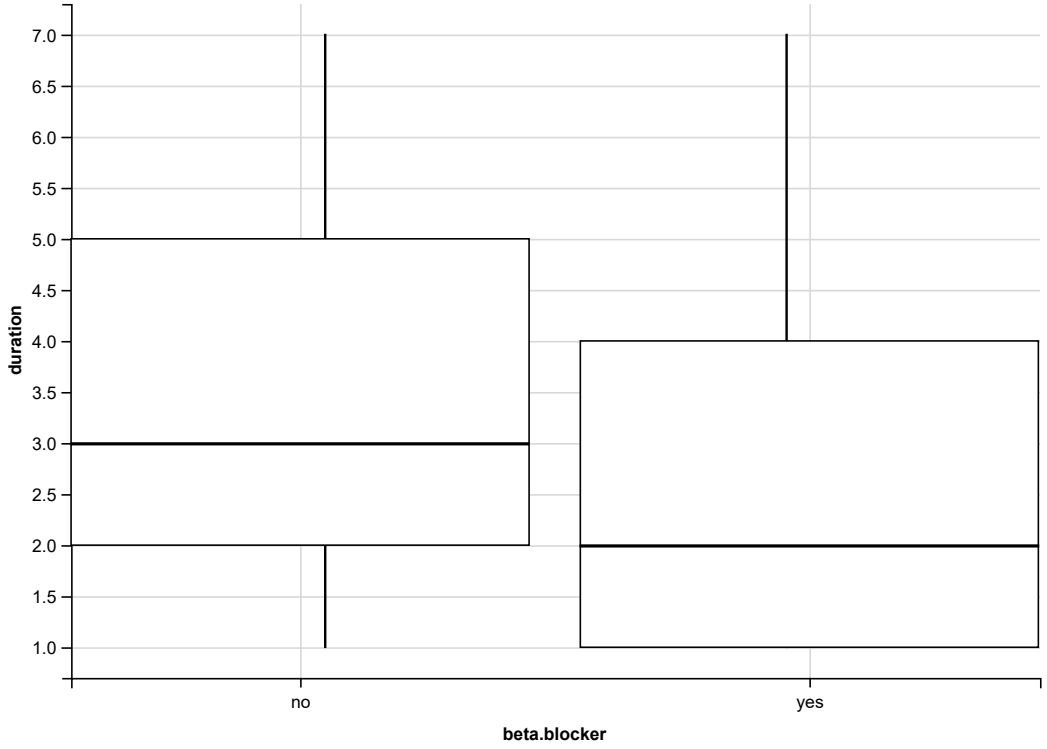


2. As we created the score, we know that it is a combination out of the pain level and the duration. Use again the boxplot to see whether the beta blockers have a stronger effect to the duration or the pain level.

Pain Level:



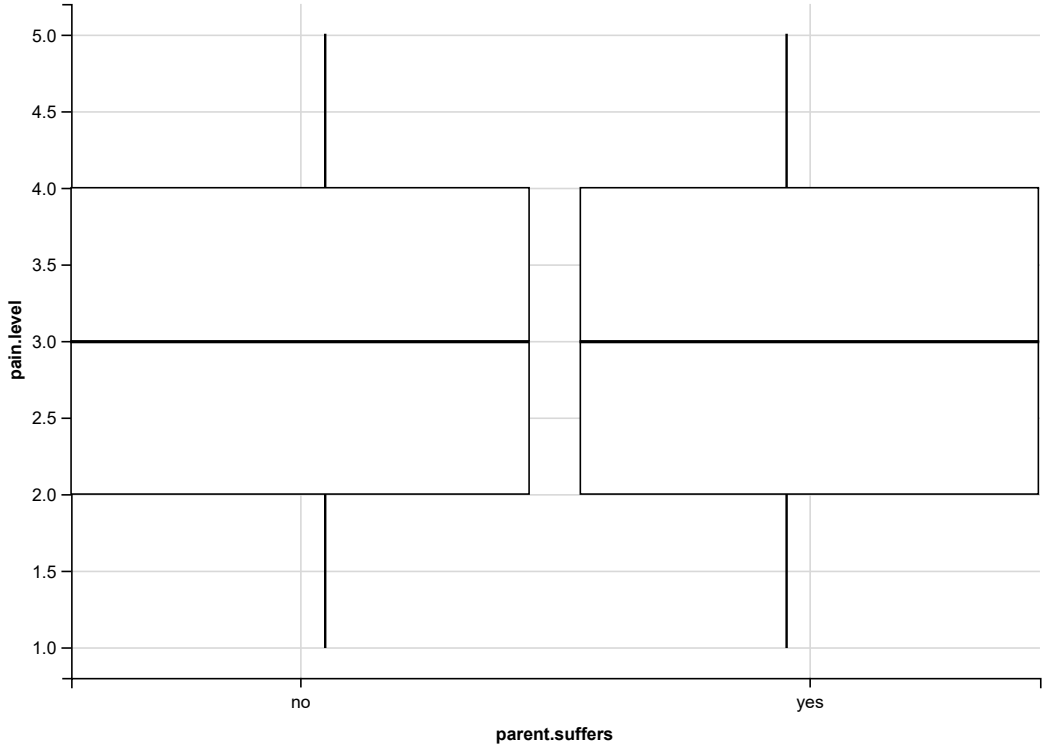
Duration:



3. What is the p-value of the Wilcoxon test comparing the duration with and without beta blockers?

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  duration by beta.blocker
## W = 61102000, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

4. Does the column parent.suffers have an impact to the duration or pain level? Find it out by plotting and statistical testing.




```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pain.level by parent.suffers
## W = 43412000, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Exercise 6.6 - Count

The findings from the previous exercise indicate that the beta blockers have a strong impact to the duration of a sick headache. Let's use `count` to get some information about the categorical features like ageRange, gender, beta.blocker, parent.suffers.

The function `count` can be used to simply check the number of records with and without visual disorders:

```
  joined %>%
    count(visual.disorder)
```

```
## # A tibble: 2 x 2
##   visual.disorder     n
##   <chr>           <int>
## 1 no              19040
## 2 yes              3607
```

When combined with `grouped_by` we can have a closer look to dependensies between the different columns:

```
  joined %>%
    group_by(beta.blocker) %>%
    count(visual.disorder)
```

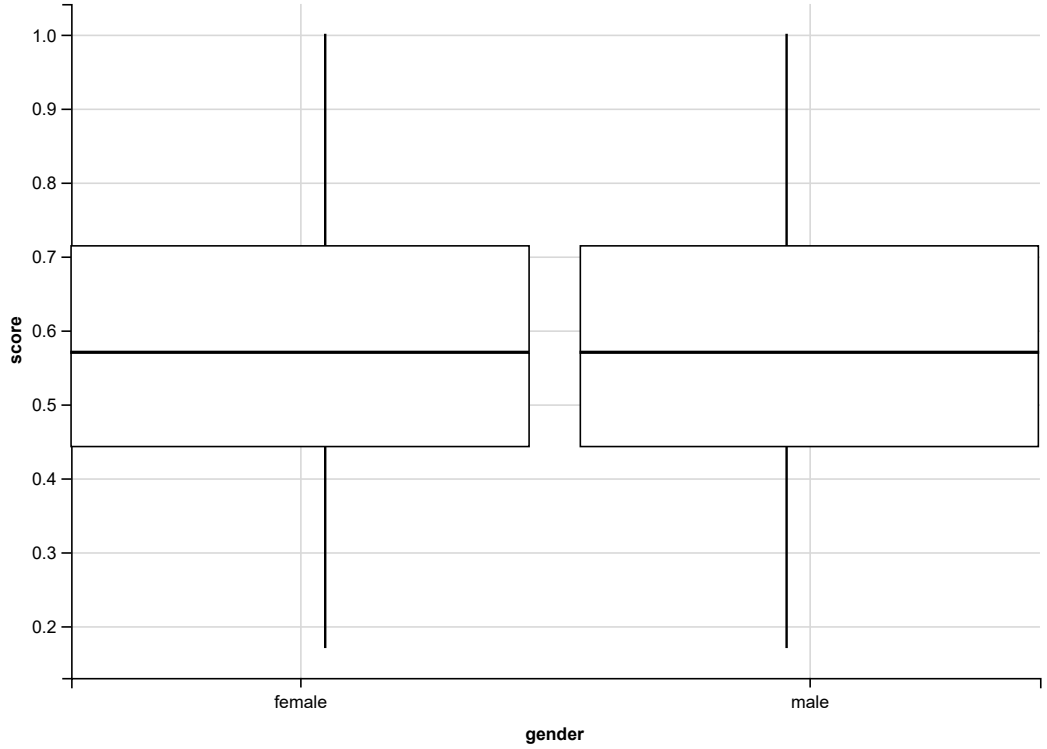
```
## # A tibble: 4 x 3
## # Groups:   beta.blocker [2]
##   beta.blocker visual.disorder     n
##   <chr>        <chr>           <int>
## 1 no           no               4837
## 2 no           yes              1256
## 3 yes          no              14203
## 4 yes          yes              2351
```

1. Let's see whether females or males use the beta blockers more often.

```
## # A tibble: 4 x 3
## # Groups:   gender [2]
##   gender beta.blocker     n
##   <chr>  <chr>        <int>
## 1 female no            2942
## 2 female yes           3670
## 3 male   no            3151
## 4 male   yes          12884
```

It looks like men use the beta blockers more frequently. Before we found out that men have a slightly lower score. Furthermore, we have seen that the beta blockers have a strong impact to the duration of a sick headache and the duration is a part of the score.

2. Replot the boxplot having the gender on the x-axis and the score on the y-axis. But this time only with records with no beta blockers.



3. What about the statistical test?

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  score by gender
## W = 4597000, p-value = 0.5787
## alternative hypothesis: true location shift is not equal to 0
```

Exercise 6.7 - The parents and age

What about the information about the parents. Do patients have a higher or lower pain level if their mother or father has migraine?

1. Calculate the mean pain level for the two groups defined by parent.suffers.

```
## # A tibble: 2 x 2
##   parent.suffers avg_pl
##   <chr>           <dbl>
## 1 no               3.09
## 2 yes              3.28
```

2. Calculate the mean duration for the different age ranges.

```
## # A tibble: 7 x 2
##   ageRange avg_d
##   <chr>    <dbl>
## 1 20-30     2.88
## 2 30-40     2.90
## 3 40-50     2.94
## 4 50-60     2.86
## 5 60-70     2.84
## 6 70-80     2.89
## 7 80-90     2.93
```

Exercise 6.8 - Final set with the date

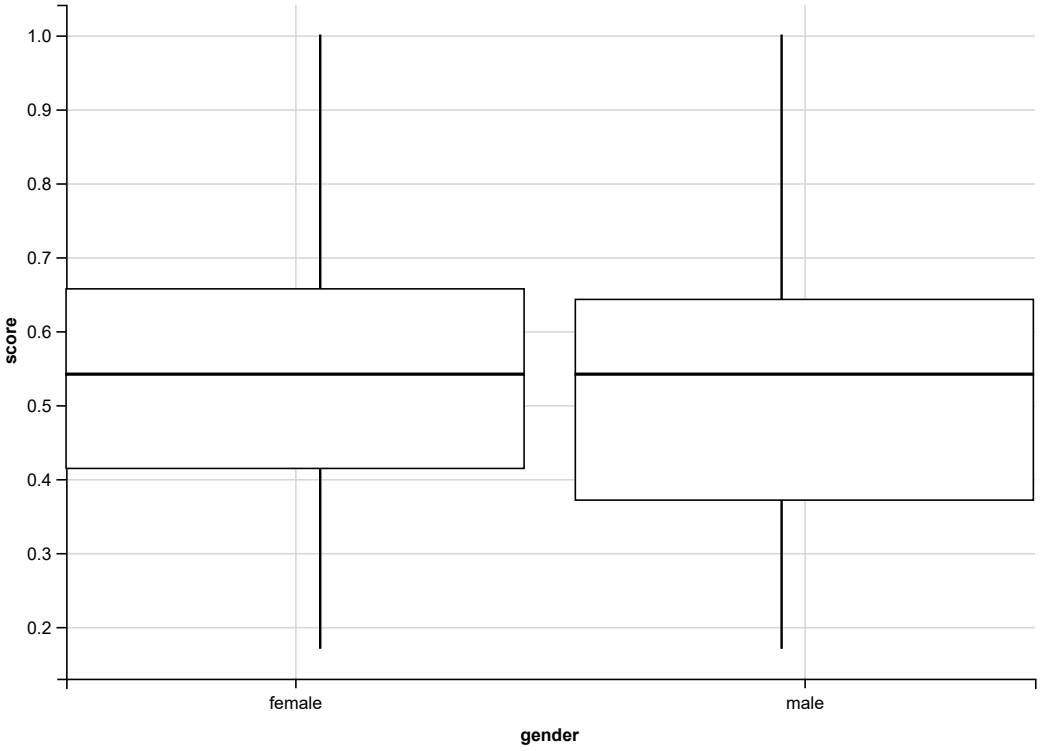
There is already data from 2018 but in our datasets there is no information about the year. So it could be difficult to differentiate. Add another column with the year (it is always 2017). Add also a new column that describes the data in the format `DAY.MONTH.YEAR`. Assign this dataset to the variable `final`.

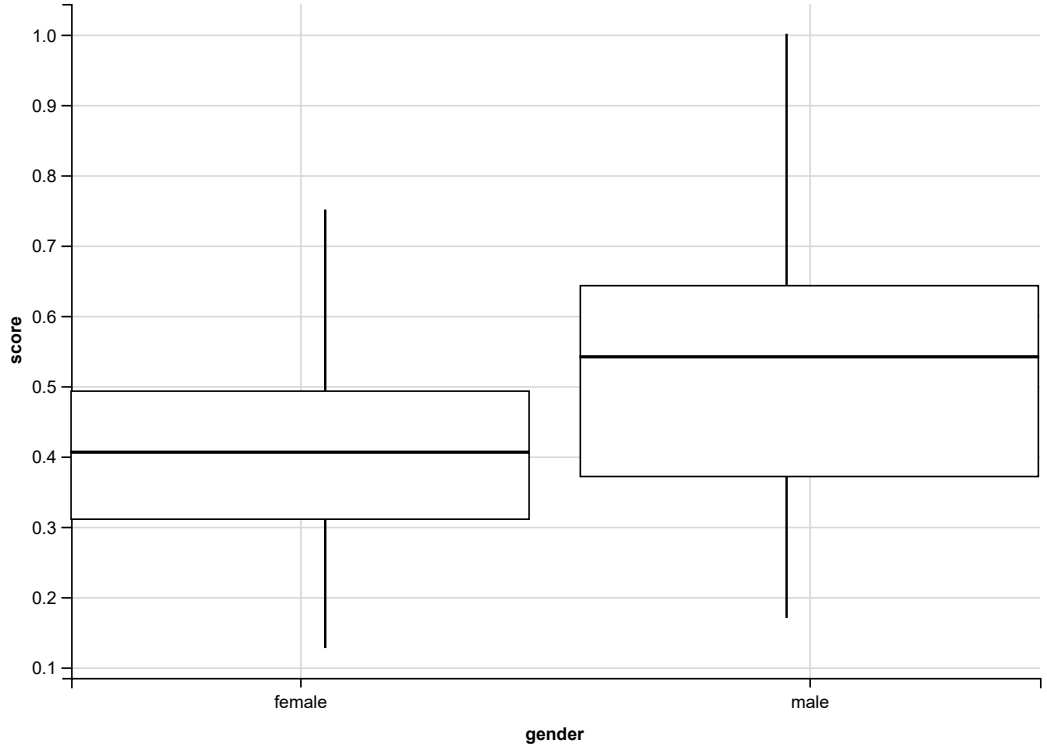
```
final %>%
  select(-c(beta.blocker, parent.suffers, duration, pain.level, visual.disorder, ageRange))
```

```
## # A tibble: 22,647 x 5
##    user.ID date        age gender score
##      <dbl> <chr>     <dbl> <chr>  <dbl>
##  1       4 4.6.2017     44 female 0.471
##  2    3286 4.1.2017     67 male   0.371
##  3    3048 29.5.2017    35 male   0.514
##  4    3290 28.4.2017    78 male   0.543
##  5    4363 17.4.2017    32 male   0.371
##  6    3332 6.6.2017     52 male   0.657
##  7     851 8.12.2017    45 male   1
##  8    4247 11.6.2017    42 male   0.543
##  9    1674 28.4.2017    37 female 0.371
## 10     616 17.8.2017    58 male   0.686
## # ... with 22,637 more rows
```

Exercise 6.9 - Cheating

Basically you are not that satisfied with the whole study. It is nice to show that the scores are significantly different between women and men, but you expected to see a stronger difference in the boxplot. Moreover, everything seems to be driven by the beta blockers. Only now you are allowed to cheat :) Use `ifelse` to artificially decrease the score of either women or men. By which extend you do this and for which gender, it is up to you. After this “artificial manipulation” plot the boxplot again. Is the p-value still good?






```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  score by gender
## W = 32803000, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Exercise 6.10 - Rank

1. Rank users by their mean pain level, the highest value on top.

```
## # A tibble: 4,451 x 2
##    user.ID avg_pl
##      <dbl>  <dbl>
##  1      13      5
##  2      61      5
##  3      68      5
##  4      83      5
##  5      85      5
##  6     185      5
##  7     285      5
##  8     289      5
##  9     337      5
## 10     381      5
## # ... with 4,441 more rows
```

Exercise 6.11 - Tidy up

There is new data from new users. On the website you will find the file `users2018.tsv.gz` . Load it via the URL.

```
## # A tibble: 200 x 5
##    user.ID gender parent.suffers error1   error2
##      <dbl> <chr>  <chr>          <chr>    <chr>
##  1    4486 female yes            age      60
##  2    4486 female yes            ageRange 60-70
##  3    4487 female no             age      88
##  4    4487 female no             ageRange 80-90
##  5    4488 male   no             age      81
##  6    4488 male   no             ageRange 80-90
##  7    4489 female no             age      25
##  8    4489 female no             ageRange 20-30
##  9    4490 male   no             age      39
## 10    4490 male   no             ageRange 30-40
## # ... with 190 more rows
```

1. Check what is wrong with it and make it tidy. It should look like this:

```
## # A tibble: 100 x 5
##    user.ID gender parent.suffers age   ageRange
##      <dbl> <chr>  <chr>          <chr> <chr>
##  1    4486 female yes            60    60-70
##  2    4487 female no             88    80-90
##  3    4488 male   no             81    80-90
##  4    4489 female no             25    20-30
##  5    4490 male   no             39    30-40
##  6    4491 male   no             36    30-40
##  7    4492 male   no             20    20-30
##  8    4493 female no             52    50-60
##  9    4494 male   no             70    70-80
## 10    4495 female no             79    70-80
## # ... with 90 more rows
```

One of the data scientists created a file about the average duration from different age ranges in the different years. On the website you will find the file `AVGduration.tsv.gz` . Load it via the URL.

```
## # A tibble: 6 x 4
##   ageRange `2017` `2018` `2019`
##   <chr>     <dbl>  <dbl>  <dbl>
## 1 20-30      2.87   2.56   1.99
## 2 30-40      2.9    2.74   2.87
## 3 40-50      2.93   2.98   2.43
## 4 50-60      2.85   2.64   2.32
## 5 70-80      2.89   2.16   2.14
## 6 80-90      2.92   2.68   2.23
```

2. Each record should have its own line. Check what is wrong with it and make it tidy.

```
## # A tibble: 18 x 3
##    ageRange year  avgDuration
##    <chr>    <chr>       <dbl>
##  1 20-30    2017         2.87
##  2 30-40    2017         2.9
##  3 40-50    2017         2.93
##  4 50-60    2017         2.85
##  5 70-80    2017         2.89
##  6 80-90    2017         2.92
##  7 20-30    2018         2.56
##  8 30-40    2018         2.74
##  9 40-50    2018         2.98
## 10 50-60    2018         2.64
## 11 70-80    2018         2.16
## 12 80-90    2018         2.68
## 13 20-30    2019         1.99
## 14 30-40    2019         2.87
## 15 40-50    2019         2.43
## 16 50-60    2019         2.32
## 17 70-80    2019         2.14
## 18 80-90    2019         2.23
```