02 - Data Manipulation

David Vichansky [6819516]

12-11-2020

Here is an example file you can write.

First, load the packages:

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages ---------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(haven)
library(readxl)
library(tinytex)
```

1. Objects and classes

```
object_1 <- 1:5
object_2 <- 1L:5L
object_3 <- "-123.456"
object_4 <- as.numeric(object_2)
object_5 <- letters[object_1]
object_6 <- as.factor(rep(object_5, 2))
object_7 <- c(1, 2, 3, "4", "5", "6")
```

```
df <- data.frame(object_1, object_2)
sapply(df, class)
```

```
##  object_1  object_2
## "integer" "integer"
```

```
# creating dataframe for all objects does not work due to differing row lengths
class(object_1)
```

```
## [1] "integer"
```

```
class(object_2)
```

```
## [1] "integer"
```

```
class(object_3)
```

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

```
class(object_4)
```

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

```
class(object_5)
```

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

```
class(object_6)
```

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

```
class(object_7)
```

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

```
# Here is a comment about the class of object_1
```

2. Convert to numeric

```
object_7 <- as.numeric(object_7)
class(object_7)
```

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

3. Create list titled 'objects'

```
objects <- list(object_1,object_2,object_3,object_4,object_5,object_6,object_7)
objects
```

```
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
## [1] 1 2 3 4 5
##
## [[3]]
## [1] "-123.456"
##
```

```
## [[4]]
## [1] 1 2 3 4 5
##
## [[5]]
## [1] "a" "b" "c" "d" "e"
##
## [[6]]
##  [1] a b c d e a b c d e
## Levels: a b c d e
##
## [[7]]
## [1] 1 2 3 4 5 6
```

4. Create data-frame

```
dataframe <- data.frame(object_1, object_2, object_5)
dataframe
```

```
##   object_1 object_2 object_5
## 1        1        1        a
## 2        2        2        b
## 3        3        3        c
## 4        4        4        d
## 5        5        5        e
```

5. Determine size of data-frame

```
ncol(dataframe)
```

```
## [1] 3
```

```
nrow(dataframe)
```

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

6. Read csv file

```
apps <- read.csv("data/googleplaystore.csv", header = TRUE)
#too large to print in pdf
#apps
```

7. Did any column get a variable type you did not expect?

```
lapply(apps, class)
```

```
## $App
## [1] "factor"
##
## $Category
## [1] "factor"
```

```
##
## $Rating
## [1] "numeric"
##
## $Reviews
## [1] "integer"
##
## $Size
## [1] "factor"
##
## $Installs
## [1] "factor"
##
## $Type
## [1] "factor"
##
## $Price
## [1] "factor"
##
## $Content.Rating
## [1] "factor"
##
## $Genres
## [1] "factor"
##
## $Last.Updated
## [1] "factor"
##
## $Current.Ver
## [1] "factor"
##
## $Android.Ver
## [1] "factor"
```

We suppose the variable type "factor" is unexpected in the sense we haven not worked with this yet in this example, "numeric" and "integer" we know the functionalities of those two data types.

8. First row of data-frame 'apps'.

```
head(apps)
```

```
##                                                 App       Category Rating
## 1     Photo Editor & Candy Camera & Grid & ScrapBook ART_AND_DESIGN    4.1
## 2                               Coloring book moana ART_AND_DESIGN    3.9
## 3 U Launcher Lite - FREE Live Cool Themes, Hide Apps ART_AND_DESIGN    4.7
## 4                             Sketch - Draw & Paint ART_AND_DESIGN    4.5
```

```
## 5              Pixel Draw - Number Art Coloring Book ART_AND_DESIGN    4.3
## 6                         Paper flowers instructions ART_AND_DESIGN    4.4
##   Reviews Size    Installs Type Price Content.Rating                     Genres
## 1     159  19M     10,000+ Free     0       Everyone                Art & Design
## 2     967  14M    500,000+ Free     0       Everyone Art & Design;Pretend Play
## 3   87510 8.7M  5,000,000+ Free     0       Everyone                Art & Design
## 4  215644  25M 50,000,000+ Free     0           Teen                Art & Design
## 5     967 2.8M    100,000+ Free     0       Everyone   Art & Design;Creativity
## 6     167 5.6M     50,000+ Free     0       Everyone                Art & Design
##       Last.Updated        Current.Ver  Android.Ver
## 1  January 7, 2018              1.0.0 4.0.3 and up
## 2 January 15, 2018              2.0.0 4.0.3 and up
## 3   August 1, 2018              1.2.4 4.0.3 and up
## 4     June 8, 2018 Varies with device   4.2 and up
## 5    June 20, 2018                1.1   4.4 and up
## 6   March 26, 2017                1.0   2.3 and up
```

9. Repeating steps '5', '6' and '7' for another data set.

```
# load data set
student <- read.csv("data/students.csv", header = TRUE)
#too large to print in pdf
#student

# check dimension of dataframe
ncol(student)
```

```
## [1] 3
```

```
nrow(student)
```

```
## [1] 37
```

```
# check 'class' of variables
lapply(student, class)
```

```
## $ï..student_number
## [1] "integer"
##
## $grade
## [1] "numeric"
##
## $programme
## [1] "factor"
```

```
# 'tail' and 'View' of data set
tail(student)
```

```
##    ï..student_number    grade programme
```

```
## 32           5062746 7.426839         B
## 33           6560954 7.038757         B
## 34           6120285 6.713390         A
## 35           6553913 8.244182         A
## 36           4181101 5.624090         B
## 37           4639846 4.844375         A
```

```
View(student) # opens data set in seperate window
```

10. Create summary of 'student' data set.

```
summary(student)
```

```
##  ï..student_number     grade       programme
##  Min.   :4011659   Min.   :4.844   A:19
##  1st Qu.:4862862   1st Qu.:6.390   B:18
##  Median :6000241   Median :7.151
##  Mean   :5686729   Mean   :6.991
##  3rd Qu.:6553913   3rd Qu.:7.573
##  Max.   :6997130   Max.   :9.291
```

11. Filter students with a grade lower than 5.5.

```
filter(student, grade < 5.5)
```

```
##   ï..student_number    grade programme
## 1           6114656 5.159602         A
## 2           5265402 5.487652         B
## 3           4639846 4.844375         A
```

12. Filter for students with a grade higher than 8 from programme A.

```
filter(student, grade > 8.0, programme == "A")
```

```
##   ï..student_number    grade programme
## 1           6352581 8.091947         A
## 2           6165611 8.019184         A
## 3           4133949 8.397477         A
## 4           4011659 8.943093         A
## 5           6553913 8.244182         A
```

13. Sort students from programme A are on top of the data frame and within the programmes the highest grades come first.

```
arrange(student, programme, -grade)
```

```
##    ï..student_number    grade programme
## 1           4011659 8.943093         A
## 2           4133949 8.397477         A
## 3           6553913 8.244182         A
```

```
## 4            6352581 8.091947         A
## 5            6165611 8.019184         A
## 6            6997130 7.754851         A
## 7            4862862 7.707184         A
## 8            6562582 7.572885         A
## 9            4483974 7.463982         A
## 10           5128923 7.260540         A
## 11           6827756 6.802390         A
## 12           6040650 6.754859         A
## 13           6580486 6.728600         A
## 14           6120285 6.713390         A
## 15           5117250 6.540712         A
## 16           6207923 6.001035         A
## 17           4096023 5.916181         A
## 18           6114656 5.159602         A
## 19           4639846 4.844375         A
## 20           6375078 9.290566         B
## 21           5625916 7.900654         B
## 22           4353370 7.566583         B
## 23           5210665 7.511415         B
## 24           5062746 7.426839         B
## 25           6889790 7.330978         B
## 26           5977188 7.256302         B
## 27           6921600 7.193169         B
## 28           4668787 7.150634         B
## 29           6560954 7.038757         B
## 30           6561723 6.573150         B
## 31           6960778 6.421787         B
## 32           5687211 6.389711         B
## 33           6000241 6.081056         B
## 34           4807286 5.963946         B
## 35           4181101 5.624090         B
## 36           6301091 5.527946         B
## 37           5265402 5.487652         B
```

14. Show only the student_number and programme columns from the students dataset.

```
select(student, ï..student_number, programme)
```

```
##    ï..student_number programme
## 1            5117250         A
## 2            6562582         A
## 3            6000241         B
## 4            4862862         A
## 5            6561723         B
## 6            5625916         B
```

```
## 7            4096023         A
## 8            6114656         A
## 9            5265402         B
## 10           5977188         B
## 11           6889790         B
## 12           4807286         B
## 13           6352581         A
## 14           6207923         A
## 15           5210665         B
## 16           6040650         A
## 17           4353370         B
## 18           4483974         A
## 19           6997130         A
## 20           6827756         A
## 21           6580486         A
## 22           6165611         A
## 23           4133949         A
## 24           5687211         B
## 25           5128923         A
## 26           6921600         B
## 27           6375078         B
## 28           4668787         B
## 29           4011659         A
## 30           6960778         B
## 31           6301091         B
## 32           5062746         B
## 33           6560954         B
## 34           6120285         A
## 35           6553913         A
## 36           4181101         B
## 37           4639846         A
```

15. Change the codes in the programme column of the students dataset to their names.

```
students_recoded <- student %>% mutate(programme=recode(programme,
                          `A`="Science",
                          `B`="Social Science"))
#too large to print
# students_recoded
```

16. Create a data processing pipeline that (a) loads the apps dataset, (b) parses the number of installs as 'Downloads' variable using mutate and parse_number(), (c) shows only apps with more than 500 000 000 downloads, (d) orders them by rating (best on top), and (e) shows only the relevant columns (you can choose which are relevant, but select at least the Rating and Category variables). Save the result under the name popular_apps.

```
popular_apps <- read.csv("data/googleplaystore.csv") %>%
  filter(Installs == "500,000,000+") %>%
  mutate(Downloads=parse_number("500,000,000+")) %>%
  arrange(Rating) %>%
  select(Rating, Category, Content.Rating, Genres)

#too large to print
# popular_apps
```

17. Show the median, minimum, and maximum for the popular apps dataset.

```
popular_apps %>%
  summarise(
    mean = mean(Rating),
    variance = var(Rating),
    min = min(Rating),
    max = max(Rating)
  )
```

```
##   mean  variance min max
## 1 4.35 0.0228169   4 4.7
```

18. Add the median absolute deviation to the summaries.

```
mad <- function(x) {
  median(abs(x - median(x)))
}

popular_apps %>% summarise(mad = mad(Rating))
```

```
##   mad
## 1 0.1
```

19. Create a grouped summary of the ratings per category.

```
popular_apps %>%
  group_by(Category) %>%
  summarise(
    mean = mean(Rating),
    variance = var(Rating),
    min = min(Rating),
    max = max(Rating)
  )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 9 x 5
##   Category            mean variance   min   max
```

```
##   <fct>              <dbl>    <dbl> <dbl> <dbl>
## 1 COMMUNICATION       4.35  0.0176    4.2   4.6
## 2 FAMILY              4.4   0.01      4.3   4.5
## 3 GAME                4.36  0.00507   4.3   4.5
## 4 HEALTH_AND_FITNESS  4.3  NA         4.3   4.3
## 5 NEWS_AND_MAGAZINES  4.35  0.003     4.3   4.4
## 6 PRODUCTIVITY        4.34  0.0227    4.1   4.5
## 7 SOCIAL              4.1   0.0240    4     4.3
## 8 TOOLS               4.44  0.0453    4.2   4.7
## 9 VIDEO_PLAYERS       4.5   0         4.5   4.5
```

20. Create a summary based on the Google play store apps dataset. We create a pivot table.

```
#popular123 <- popular_apps %>%
#  group_by(Category, Content.Rating) %>%
#  summarise(
#    mean = mean(Rating)
#  )
#
#pivot_wider(popular, id_cols = Category, names_from = 'Content.Rating', values_from =


# create pivot table to show the mean rating by app 'category' to then compare across
popular_apps %>%
  group_by(Category, Content.Rating) %>%
  summarise(
    mean = mean(Rating)
  ) %>%
  pivot_wider(id_cols = Category, names_from = 'Content.Rating', values_from = mean)
```

```
## `summarise()` regrouping output by 'Category' (override with `.groups` argument)
```

```
## # A tibble: 9 x 5
## # Groups:   Category [9]
##   Category           Everyone  Teen `Everyone 10+` `Mature 17+`
##   <fct>                 <dbl> <dbl>          <dbl>        <dbl>
## 1 COMMUNICATION          4.32   4.5            NA           NA
## 2 FAMILY                 4.4   NA              NA           NA
## 3 GAME                   4.36  NA              NA           NA
## 4 HEALTH_AND_FITNESS     4.3   NA              NA           NA
## 5 NEWS_AND_MAGAZINES    NA     NA               4.4          4.3
## 6 PRODUCTIVITY           4.34  NA              NA           NA
## 7 SOCIAL                NA      4.1            NA           NA
## 8 TOOLS                  4.44  NA              NA           NA
## 9 VIDEO_PLAYERS          4.5   NA              NA           NA
```