

TEND_Subsetting_Data

Me

2023-05-26

```
knitr::opts_chunk$set(root.dir = "C:C:\\Users\\adcre\\OneDrive\\Documents\\R_Programming\\TEND",  
                        echo = TRUE)  
getwd()
```

```
## [1] "C:/Users/adcre/OneDrive/Documents/R_Programming/TEND"
```

Adding libraries/packages

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
## v dplyr      1.1.2      v readr      2.1.4
```

```
## v forcats    1.0.0      v stringr    1.5.0
```

```
## v ggplot2    3.4.2      v tibble     3.2.1
```

```
## v lubridate  1.9.2      v tidyr      1.3.0
```

```
## v purrr      1.0.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()     masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
## Warning: package 'psych' was built under R version 4.2.3
```

```
##
```

```
## Attaching package: 'psych'
```

```
##
```

```
## The following objects are masked from 'package:ggplot2':
```

```
##
```

```
##    %+%, alpha
```

```
##
```

```
##
```

```
## Please cite as:
```

```
##
```

```
## Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
## R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

R Markdown

This markdown will serve as a reference sheet for different subsetting methods, packages, applications, and more. More specifically, how to subset data casewise, listwise, by ID, or by variable. Having a large and complex df could prevent you from gaining insight about specific data. Having the ability to call for certain ID numbers or variables, like gender or age, allows you to use a magnifying glass on your large df. I also plan on learning and practicing substringing values, and being able to remove the “sub” from “sub-xxxx” in the subject ID column. The key differences between substringing and subsetting is that substringing is applied to character strings like text files, and can output specific ranges or positions in a long character string. A subset is more widely used because it can be applied to data frames and other data structures. Based off of certain arguments or criteria, a subset can give you a closer look on specific points in your data structure. Subsets are represented by brackets[], and more of the syntax will be learned with time and practice.

```
# We will first create a df that allows us to visualize subsetting, and learn the syntax used to call f

# Create a data frame
df <- data.frame(
  Name = c("John", "Emily", "David", "Sarah", "Michael"),
  Age = c(25, 32, 28, 35, 30),
  City = c("New York", "London", "Paris", "Tokyo", "Sydney")
)

# Print the data frame
print(df)
```

The first code chunk will be examples of subsetting on a simple data structure that is created within the chunk. In the interest of truly being able to apply these skills, I will use a df imported from Kaggle after this code chunk in order to test it in a real-life situation.

```
##      Name Age   City
## 1   John  25 New York
## 2  Emily  32  London
## 3  David  28   Paris
## 4  Sarah  35   Tokyo
## 5 Michael 30  Sydney
```

```
# Notice the use of brackets[] and a dollar sign$ to indicate that you want to subset, and that you wou

# Subset the data frame by age
subset_df <- df[df$Age > 30, ]

# Print the subsetted data frame
print(subset_df)
```

```
##      Name Age   City
## 2  Emily  32  London
## 4  Sarah  35   Tokyo
```

Let's try subsetting by variables and conditions on a more realistic data set found on Kaggle. We will explore the same data set found in my "Importing_Data_Sets" repo, the Depression Dataset found at this link on Kaggle: <https://www.kaggle.com/datasets/arashnic/the-depression-dataset>

```
# First, I will load in the scores from the selected dataset. It can be done many ways, and should be w
dep_scores <- read_csv("scores.csv", show_col_types = FALSE)
str(dep_scores)
```

This data set is represented as 'unzipped_data' in my Files

```
## spc_tbl_ [55 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ number   : chr [1:55] "condition_1" "condition_2" "condition_3" "condition_4" ...
## $ days     : num [1:55] 11 18 13 13 13 7 11 5 13 9 ...
## $ gender   : num [1:55] 2 2 1 2 2 1 1 2 2 2 ...
## $ age      : chr [1:55] "35-39" "40-44" "45-49" "25-29" ...
## $ afftype  : num [1:55] 2 1 2 2 2 2 1 2 1 2 ...
## $ melanch  : num [1:55] 2 2 2 2 2 2 NA NA NA 2 ...
## $ inpatient: num [1:55] 2 2 2 2 2 2 2 2 2 2 ...
## $ edu      : chr [1:55] "6-10" "6-10" "6-10" "11-15" ...
## $ marriage : num [1:55] 1 2 2 1 2 1 2 1 1 1 ...
## $ work     : num [1:55] 2 2 2 1 2 2 1 2 2 2 ...
## $ madsr1   : num [1:55] 19 24 24 20 26 18 24 20 26 28 ...
## $ madsr2   : num [1:55] 19 11 25 16 26 15 25 16 26 21 ...
## - attr(*, "spec")=
## .. cols(
## ..   number = col_character(),
## ..   days = col_double(),
## ..   gender = col_double(),
## ..   age = col_character(),
## ..   afftype = col_double(),
## ..   melanch = col_double(),
## ..   inpatient = col_double(),
## ..   edu = col_character(),
## ..   marriage = col_double(),
## ..   work = col_double(),
## ..   madsr1 = col_double(),
## ..   madsr2 = col_double()
## .. )
## - attr(*, "problems")=<externalptr>
```

```
# By using str(), I can view the different columns by name. This is essentially a list of the variables
dep_subset <- dep_scores[dep_scores$age == "40-44", ]
print(dep_subset)
```

```
## # A tibble: 5 x 12
##   number      days gender age   afftype melanch inpatient edu   marriage work
##   <chr>      <dbl>  <dbl> <chr>   <dbl>   <dbl>      <dbl> <chr>    <dbl> <dbl>
## 1 condition_2    18      2 40-44     1       2          2 6-10      2      2
## 2 condition_12   12      2 40-44     1       2          2 6-10      2      2
```

```
## 3 condition_18      13      2 40-44      3      2      2 11-15      2      2
## 4 control_8        13      2 40-44      NA      NA      NA <NA>      NA      NA
## 5 control_16       13      2 40-44      NA      NA      NA <NA>      NA      NA
## # i 2 more variables: madsr1 <dbl>, madsr2 <dbl>
```

```
# Notice that we used '==' instead of the logical condition '<' or '>'. So, rather than calling for sub
# What if I wanted to call for a specific ID? There is a 'number' column, and each subject is either re
test_subject1 <- dep_scores[dep_scores$number == "condition_1", ]
control_subject1 <- dep_scores[dep_scores$number == "control_1", ]
View(control_subject1)
View(test_subject1)
```

DURING-Session REFLECTION

POST-Session REFLECTION