Problem Set 2

Caroline Robbins

2024-09-12

Download dataset of IMDB movies (imdb top 2000 movies.csv) from the following link or from a shared dropbox folder

```
##Set working directory and bring in necessary libraries.
setwd("C:/Users/crobb/Downloads")
library(tidyverse)
```

```
## 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 'tidyr' was built under R version 4.2.3
## Warning: package 'readr' was built under R version 4.2.3
## Warning: package 'purrr' was built under R version 4.2.3
## Warning: package 'dplyr' was built under R version 4.2.3
## Warning: package 'stringr' was built under R version 4.2.3
## Warning: package 'forcats' was built under R version 4.2.3
## Warning: package 'lubridate' was built under R version 4.2.3
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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
```

```
library(tinytex)

##Load the IMDB Movies dataset into the workspace.
raw_data <- read.csv("imdb_top_2000_movies.csv")
```

1. Create a function mins to hrs that would transform number of minutes into number of hours. Apply this function to the variable Duration to create a new variable Duration in hrs

```
mins_to_hrs <- function(x){
  x/60
}
```

```
raw_data$Duration_in_hrs <- mins_to_hrs(raw_data$Duration)
```

2. Using R show how many directors (unique entries) ended up in this list of movies.

```
n_distinct(raw_data$Director)
```

```
## [1] 918
```

```
##918 distinct values for Director, meaning 918 different directors are on this
##list.
```

3. Which director's movies on average have the highest IMDB.Rating score? (To answer this question you can either use a for loop structure or base R functions there is no coding requirement of how you calculate that. But you do have to use R coding to show how you derived the answer to this question.)

```
class("IMDB.Rating")
```

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

```
as.numeric(raw_data$IMDB.Rating)
```

```
##    [1] 9.2 9.0 7.7 8.3 7.4 7.6 8.5 7.7 8.0 7.8 8.5 8.1 7.9 8.1 7.9 8.8 8.0 7.9
##   [19] 7.8 7.7 8.3 5.8 7.1 7.7 6.8 6.4 8.0 8.5 7.2 7.5 7.7 8.5 8.6 8.7 7.4 8.0
##   [37] 9.0 8.3 8.1 7.4 8.4 8.2 8.6 8.3 8.2 7.9 8.0 7.7 7.4 7.0 8.1 7.7 8.1 8.0
##   [55] 7.6 8.4 8.0 8.0 8.1 8.0 7.7 7.2 8.4 7.9 8.2 8.0 7.2 7.9 8.0 8.2 7.4 7.7
##   [73] 6.7 8.4 8.1 7.5 7.6 8.1 7.6 7.6 7.9 7.9 7.7 7.7 8.1 8.1 7.6 7.3 8.3 8.2
##   [91] 6.8 7.4 7.4 7.1 7.9 8.4 7.8 8.3 8.1 7.8 7.7 8.3 8.2 8.0 8.0 7.4 7.8 7.9
##  [109] 7.1 7.6 8.1 8.1 8.3 6.8 7.5 6.9 6.7 7.4 7.5 7.6 7.4 7.7 6.9 6.5 6.8 7.2
##  [127] 7.8 8.0 7.9 8.2 7.7 7.9 7.3 8.4 7.8 7.4 8.1 8.2 8.3 7.1 8.1 8.1 7.5 6.9
##  [145] 7.8 8.3 7.4 7.7 7.4 6.7 7.1 7.2 6.8 7.9 7.5 6.9 7.5 8.1 6.2 7.4 8.6 8.5
##  [163] 8.4 8.4 8.3 8.3 8.2 8.2 8.2 8.1 8.1 8.1 8.2 8.1 8.1 8.0 8.1 8.1 8.1 8.0
##  [181] 7.9 7.9 8.0 8.0 7.3 7.6 8.0 8.0 7.0 6.5 7.8 7.4 6.9 7.0 8.0 8.1 7.2 7.6
##  [199] 6.9 8.1 7.6 8.0 7.3 7.6 8.5 7.8 7.4 8.3 8.3 5.8 8.3 7.2 8.2 7.9 7.5 7.6
##  [217] 8.1 7.8 8.4 7.8 7.9 7.6 7.3 7.8 6.8 7.1 7.5 8.2 7.8 7.6 8.0 7.1 6.9 7.9
##  [235] 8.3 6.7 7.6 6.8 7.3 7.6 7.4 5.8 4.9 7.5 6.9 6.4 6.9 7.0 6.5 7.0 5.9 6.5
##  [253] 6.3 8.3 6.7 7.3 6.6 7.5 7.1 6.6 7.8 7.1 8.1 8.1 8.0 6.9 7.7 6.0 7.2 8.1
##  [271] 8.5 8.2 6.6 8.6 8.6 6.8 7.6 8.5 7.4 6.7 7.0 7.2 8.1 7.4 6.9 6.6 7.3 7.8
##  [289] 7.6 7.5 6.8 7.6 6.9 7.5 7.2 6.7 7.0 7.5 8.4 7.8 7.8 8.4 7.8 8.0 8.3 5.8
##  [307] 7.6 7.9 7.8 7.1 7.0 7.7 7.4 6.6 7.5 7.6 7.1 8.7 8.0 7.7 6.5 7.4 8.1 7.9
##  [325] 6.3 7.8 7.5 7.8 6.3 7.5 7.3 7.8 6.6 7.6 8.2 7.3 7.7 7.1 7.9 7.7 8.2 7.1
##  [343] 8.0 7.7 7.3 7.1 6.4 7.8 6.8 7.3 7.9 7.9 7.2 7.0 7.4 7.2 7.3 8.0 6.4 8.3
##  [361] 7.2 7.6 8.0 8.0 7.2 5.9 7.7 6.9 6.9 7.0 6.7 5.8 5.4 7.7 7.1 7.8 7.3 8.3
##  [379] 6.3 6.1 7.1 7.7 6.9 7.8 8.1 6.7 7.5 7.0 7.5 7.5 6.7 7.0 7.4 7.2 7.5 7.6
##  [397] 7.4 7.3 6.8 6.9 6.1 7.1 6.9 6.7 6.8 6.3 6.9 7.0 7.3 7.2 7.0 7.3 6.1 7.1
##  [415] 6.6 7.7 8.1 8.1 6.8 6.9 5.4 6.5 5.8 5.0 6.4 6.4 8.0 6.8 7.1 7.3 7.5 7.1
##  [433] 7.8 6.2 8.0 7.6 7.8 7.3 6.8 7.3 6.6 7.6 7.8 7.1 8.0 4.5 6.2 7.3 6.5 7.6
##  [451] 6.5 7.7 7.1 7.2 7.3 7.3 7.3 3.7 5.0 7.4 7.7 7.0 6.1 8.0 7.5 6.8 7.6 6.8
##  [469] 6.2 6.2 5.7 7.3 7.9 7.1 6.8 6.4 6.5 6.2 6.2 6.8 6.7 6.5 6.5 5.8 7.2 6.1
##  [487] 6.4 6.4 7.2 7.2 6.7 6.7 7.2 7.2 6.6 6.8 7.2 7.2 7.2 7.4 6.9 6.9 8.3 5.4
##  [505] 7.2 8.0 9.3 7.5 7.3 7.5 7.7 7.9 8.1 9.0 7.6 8.5 8.5 7.6 7.3 8.2 7.7 8.2
##  [523] 7.1 6.8 7.0 7.0 7.6 7.7 7.1 8.3 8.3 6.7 7.7 7.2 7.8 8.5 7.8 7.3 7.2 6.5
##  [541] 7.4 8.6 6.9 6.2 8.1 7.7 6.4 7.8 7.4 6.9 6.2 6.3 7.6 6.5 8.2 7.1 6.3 7.3
##  [559] 7.2 7.5 7.7 7.5 8.0 6.5 7.8 6.9 7.0 4.8 8.8 7.2 7.2 6.6 6.5 6.5 6.7 7.8
##  [577] 6.1 7.5 7.2 7.5 8.9 6.4 6.4 7.9 7.7 7.7 8.1 6.2 6.9 6.3 7.6 8.3 5.3 7.0
##  [595] 8.0 7.1 6.7 7.0 7.6 7.3 7.0 6.6 6.9 5.6 7.1 7.1 7.3 7.3 7.7 6.2 6.8 4.1
##  [613] 6.8 7.4 7.5 6.4 6.1 7.5 6.6 7.7 5.2 7.4 7.8 8.0 7.1 7.1 7.6 7.3 7.2 6.5
```

```
##  [631] 5.8 7.1 7.5 6.0 7.9 7.5 5.8 3.6 5.7 6.9 8.0 8.2 6.8 5.7 8.1 6.9 7.9 7.0
##  [649] 4.8 6.4 8.1 7.4 6.8 5.5 5.0 4.5 7.7 7.3 5.8 8.1 6.0 6.3 6.7 5.9 5.9 6.7
##  [667] 8.5 7.7 7.5 7.7 6.6 6.8 6.8 7.9 6.5 6.2 7.3 5.7 7.1 8.1 7.0 7.2 4.6 6.4
##  [685] 7.3 6.7 7.6 6.4 7.3 7.7 7.5 6.7 6.9 7.3 5.1 6.8 6.9 7.1 6.6 7.3 4.9 7.2
##  [703] 7.9 7.4 7.3 8.1 6.9 7.3 6.7 7.5 7.3 6.0 6.4 3.8 6.5 6.1 7.8 7.3 6.7 7.2
##  [721] 5.6 7.4 7.2 7.5 7.2 7.0 6.1 7.3 6.6 6.7 6.7 6.3 5.4 7.1 6.3 3.9 7.2 5.5
##  [739] 6.5 7.5 6.8 7.2 6.9 6.6 6.7 6.3 5.7 7.2 6.6 6.0 7.1 7.7 6.4 6.8 8.2 7.2
##  [757] 8.0 8.7 5.8 8.1 8.6 8.6 7.7 8.3 6.5 7.9 6.6 6.8 8.5 6.8 6.7 5.4 8.0 8.1
##  [775] 7.5 6.0 8.4 7.2 7.6 5.3 7.0 6.7 7.3 5.9 7.3 6.4 7.3 6.9 8.2 7.5 6.8 7.3
##  [793] 7.0 6.2 7.2 7.6 6.7 6.1 7.5 6.6 7.1 5.6 6.5 7.3 7.1 7.5 7.1 6.4 7.5 5.9
##  [811] 6.7 7.6 6.8 6.3 7.4 7.3 7.7 6.4 7.2 7.2 7.7 6.7 7.3 6.5 4.9 7.1 6.1 7.6
##  [829] 6.7 7.8 7.4 7.9 7.4 6.9 7.5 6.6 7.1 6.5 6.5 7.0 6.4 7.0 7.9 7.3 5.7 6.9
##  [847] 7.1 7.9 8.3 7.6 7.6 7.2 7.1 7.2 6.1 6.8 7.3 6.4 6.4 7.0 6.0 7.7 7.7 8.8
##  [865] 7.0 6.0 8.2 8.1 6.5 6.8 6.1 6.1 6.7 7.8 5.8 6.3 7.4 5.7 7.8 7.4 6.2 7.2
##  [883] 7.9 6.5 6.5 5.8 5.2 7.2 6.8 7.7 6.3 6.4 7.4 7.3 5.9 6.5 7.0 6.9 6.3 7.5
##  [901] 6.8 6.2 5.5 7.7 6.1 6.5 7.6 7.2 8.1 7.3 7.0 7.6 7.4 7.7 7.2 6.3 8.3 6.7
##  [919] 8.0 8.6 8.1 8.1 8.0 8.0 8.1 6.8 6.3 6.1 6.8 7.3 6.5 7.7 6.8 6.5 7.5 7.8
##  [937] 6.6 7.2 7.7 6.6 8.1 7.1 6.7 6.9 6.9 7.5 6.9 7.8 7.0 7.2 6.8 6.4 7.2 6.8
##  [955] 7.3 6.0 7.0 6.6 6.6 6.1 5.8 7.4 5.7 6.7 7.0 3.8 7.0 6.4 7.3 6.6 6.6 6.9
##  [973] 5.4 6.7 5.6 5.9 5.6 6.5 2.5 5.7 6.7 5.3 6.4 6.6 6.6 5.9 4.7 6.3 6.4 6.4
##  [991] 6.3 6.7 5.0 5.3 6.7 6.0 6.8 6.2 6.5 7.6 7.2 7.0 7.1 6.9 7.0 6.7 6.9 8.1
## [1009] 7.7 7.7 8.3 7.8 7.4 7.6 7.1 7.8 7.7 8.1 8.9 9.0 8.2 7.6 7.1 6.6 6.6 7.9
## [1027] 7.4 7.6 6.2 7.1 6.5 7.2 7.4 8.2 7.7 6.1 7.5 7.9 5.7 7.4 7.5 6.7 7.2 5.8
## [1045] 6.9 8.0 7.7 6.6 8.1 7.5 7.7 8.6 7.6 5.5 5.8 8.8 7.5 7.7 5.9 7.6 7.1 4.5
## [1063] 7.3 7.1 7.6 6.5 8.2 8.3 7.3 7.2 8.1 6.8 7.5 7.7 6.3 7.3 7.3 7.9 7.7 8.5
## [1081] 5.5 7.2 5.8 5.8 5.9 6.8 4.4 6.8 7.0 6.2 6.4 7.4 5.5 7.0 7.9 6.7 7.0 7.6
## [1099] 7.3 6.3 6.8 5.1 5.4 7.1 5.8 7.7 5.9 6.1 7.1 6.7 7.4 7.6 6.6 7.9 7.3 6.7
## [1117] 6.7 6.8 6.2 5.7 6.2 6.2 7.2 7.4 5.5 7.3 6.8 6.6 8.0 6.0 6.2 6.7 8.0 5.7
## [1135] 6.8 7.3 7.3 6.4 6.7 5.8 6.3 6.7 6.5 7.0 7.1 6.3 7.6 6.2 6.8 6.6 6.1 5.3
## [1153] 7.0 7.2 6.9 5.6 7.5 5.5 6.1 7.0 6.1 7.5 6.6 7.2 6.4 6.8 6.7 6.8 7.3 7.5
## [1171] 7.3 7.5 7.8 8.1 5.8 6.6 7.3 5.8 7.3 6.2 7.4 7.2 7.6 7.2 5.5 7.2 7.7 6.3
## [1189] 6.3 6.2 5.8 6.2 5.8 8.1 6.4 4.9 6.7 6.5 6.4 2.4 5.5 5.3 5.9 6.7 6.6 6.0
## [1207] 5.6 6.7 7.1 5.2 6.3 8.0 7.2 7.2 6.6 6.6 6.0 6.1 4.9 6.3 5.3 6.2 6.2 5.3
## [1225] 7.1 6.0 6.7 6.2 5.9 5.5 6.5 6.9 6.2 6.2 6.7 5.6 6.2 5.5 6.0 6.3 5.5 6.3
## [1243] 5.5 2.6 6.5 7.1 7.2 7.1 5.6 5.6 6.6 7.2 7.7 8.2 6.9 8.2 7.5 6.9 6.4 8.0
## [1261] 7.4 7.6 7.8 7.4 8.0 6.1 6.8 6.3 7.1 7.0 6.3 7.8 6.5 5.6 7.5 8.1 7.7 7.5
## [1279] 7.8 7.0 6.6 7.4 5.5 7.0 6.0 7.2 6.4 7.9 6.5 8.2 7.5 7.5 7.2 6.5 6.7 5.9
## [1297] 6.9 4.9 6.7 6.8 6.7 6.9 6.6 6.5 7.4 8.4 7.2 7.5 5.7 7.4 7.3 7.3 8.5 8.0
## [1315] 8.2 8.0 7.6 6.9 6.4 7.1 7.7 7.7 4.4 6.7 6.5 7.1 6.2 7.5 7.3 6.3 7.3 6.7
## [1333] 5.5 5.8 7.3 7.7 7.6 6.4 5.7 6.2 7.7 5.9 6.5 7.3 7.0 6.8 8.0 7.2 6.5 7.6
## [1351] 7.2 7.6 6.4 6.5 5.4 5.1 6.5 5.8 7.5 7.1 5.7 4.5 7.2 7.3 6.2 5.0 8.1 8.3
## [1369] 5.6 6.6 6.3 6.8 6.4 6.3 6.6 7.5 5.4 6.0 6.6 5.6 7.3 6.5 7.1 5.7 6.1 7.4
## [1387] 7.0 5.8 5.1 7.8 6.5 6.8 6.2 6.3 6.2 7.7 7.2 6.4 6.9 6.6 6.7 6.2 7.4 4.5
## [1405] 7.2 7.3 7.1 5.5 6.8 6.8 7.1 6.7 5.9 7.2 7.0 6.7 6.9 6.7 8.1 8.2 8.0 7.9
## [1423] 7.5 6.2 7.3 5.9 6.2 5.6 3.8 7.7 7.5 6.9 6.3 6.6 7.3 7.3 5.0 5.7 5.5 6.5
## [1441] 7.2 6.8 5.2 7.2 6.5 6.6 6.7 6.4 7.1 7.1 7.6 4.7 2.4 7.2 6.5 5.4 6.9 5.1
## [1459] 7.0 7.1 6.6 6.3 6.5 7.1 6.1 5.9 6.0 5.9 5.4 6.5 6.2 5.8 5.9 5.8 6.0 5.1
## [1477] 5.5 5.6 6.6 6.9 6.2 5.9 6.1 6.9 3.4 6.6 5.5 5.9 5.9 6.4 4.3 6.5 1.6 6.1
## [1495] 5.7 5.7 6.1 7.1 6.3 7.4 7.0 6.9 8.2 8.0 7.6 7.0 5.7 8.5 9.0 6.3 7.1 7.1
## [1513] 7.2 6.7 7.8 7.1 6.6 7.0 7.9 7.4 7.3 7.6 6.9 7.2 7.1 8.1 6.2 8.2 7.4 6.9
## [1531] 7.7 7.2 8.0 7.8 7.8 6.5 7.1 8.0 6.5 6.1 7.7 8.0 7.1 8.1 7.0 7.7 6.5 6.9
## [1549] 7.3 6.1 8.0 7.6 7.1 6.6 7.2 6.5 7.8 5.4 7.3 7.3 5.6 7.2 8.4 7.1 7.1 6.6
## [1567] 4.7 7.1 7.0 7.9 7.6 5.5 7.4 6.6 6.5 6.8 7.3 6.9 6.6 6.8 7.6 6.2 6.3 6.8
## [1585] 6.9 8.0 7.2 6.0 6.4 7.9 6.6 6.9 6.8 7.5 5.6 6.4 7.1 7.0 6.5 6.4 6.7 6.7
```

```
## [1603] 6.2 6.2 6.2 6.8 4.8 5.9 6.1 7.6 6.8 7.0 6.2 6.5 5.6 6.0 6.3 6.8 6.1 7.4
## [1621] 6.9 6.1 6.8 6.6 3.9 6.5 5.3 6.2 5.9 6.9 6.9 6.2 7.5 7.0 6.8 5.1 6.8 6.4
## [1639] 5.8 6.5 6.6 6.1 6.1 6.7 7.8 7.0 7.0 7.4 8.1 5.8 6.3 3.1 7.2 6.4 7.8 6.2
## [1657] 6.5 7.0 5.8 7.6 6.2 7.1 7.6 6.1 7.2 8.2 6.7 7.4 7.2 7.8 6.4 8.1 7.5 7.8
## [1675] 7.0 6.4 6.2 6.1 7.0 7.1 6.3 5.3 6.0 6.5 7.6 7.4 8.5 7.0 4.6 6.5 5.2 7.0
## [1693] 6.7 6.2 6.8 7.5 7.6 7.4 5.8 5.4 5.1 5.2 6.8 5.7 7.6 7.5 7.8 6.0 6.4 7.3
## [1711] 5.7 6.6 6.5 7.0 5.9 6.8 5.9 5.0 6.7 5.8 6.7 6.8 6.2 6.7 6.4 5.2 5.6 2.4
## [1729] 4.2 5.9 5.5 5.4 6.0 6.7 5.5 6.4 1.5 1.5 1.9 6.1 5.1 6.3 5.9 6.5 7.2 6.2
## [1747] 6.2 5.8 5.9 6.4 7.0 7.0 7.0 7.2 4.5 8.8 7.6 8.4 5.5 7.9 7.8 7.7 7.8 7.9
## [1765] 5.8 7.2 7.9 7.3 7.6 5.7 6.4 7.0 7.5 6.7 6.6 6.8 4.5 7.5 7.3 6.2 8.3 4.0
## [1783] 6.8 5.5 7.6 6.5 7.6 7.1 7.6 6.5 6.5 7.3 7.7 8.1 6.4 6.5 7.6 8.3 5.3 5.3
## [1801] 5.5 7.6 5.2 7.7 6.6 7.6 4.6 6.3 7.5 7.0 6.7 6.6 6.5 6.8 5.3 7.6 7.5 8.2
## [1819] 6.8 7.7 7.3 7.3 6.0 6.5 6.3 6.9 5.5 6.6 7.0 7.9 7.2 6.3 4.7 7.5 5.9 7.8
## [1837] 6.4 7.4 6.6 5.3 6.4 7.2 7.1 6.6 7.4 6.0 6.9 5.9 6.6 5.5 5.8 6.9 7.9 6.8
## [1855] 6.4 4.6 5.5 6.2 7.3 8.1 6.2 6.3 5.8 6.4 6.6 6.4 7.1 7.5 7.4 6.8 4.8 5.1
## [1873] 5.6 7.0 8.1 7.1 6.0 6.4 6.3 5.7 7.2 7.3 6.5 6.2 7.4 6.7 7.7 5.9 6.7 8.2
## [1891] 8.1 8.4 6.8 7.1 8.3 8.0 5.8 6.3 7.8 5.1 7.5 6.3 6.2 6.2 5.8 5.5 7.1 6.2
## [1909] 5.5 6.9 7.3 6.0 7.8 7.9 7.7 7.3 6.4 7.0 6.3 6.2 6.7 6.6 7.0 7.2 8.2 6.0
## [1927] 7.1 6.3 6.8 7.0 7.1 1.7 5.6 5.0 6.3 5.5 7.9 7.2 7.0 7.0 6.0 6.4 6.3 7.0
## [1945] 7.1 6.4 6.8 6.5 7.1 6.1 7.1 6.1 7.5 6.7 7.2 7.2 6.2 6.7 5.3 6.0 6.1 6.4
## [1963] 6.3 6.4 5.6 6.8 5.8 4.7 6.5 6.5 6.3 4.7 6.0 6.1 6.1 4.5 5.9 2.5 6.4 5.4
## [1981] 4.9 5.8 5.3 6.3 5.7 7.2 6.5 6.1 6.5 5.4 6.7 5.8 6.5 6.5 6.3 7.2 5.0 6.5
## [1999] 6.4 6.0
```

```r
director_avg <- raw_data %>%
  group_by(Director) %>%
  summarize(director_rating = mean(IMDB.Rating))
View(director_avg)

top_directors <- director_avg %>%
  slice_max(director_rating, n = 10)
View(top_directors)
```

4. Create a new vector imdb rating that includes a set of unique values of the variable IMDB.Rating. Using R show what the length of this vector is.

```r
imdb_rating <- unique(raw_data$IMDB.Rating)

length(imdb_rating)
```

```
## [1] 66
```

```r
##The length of the vector is 66 values.
```

5. Using a for loop applied to each element of the imdb rating vector, create a new vector with the rounded value of each element to the closest integer (eg., 1.4 → 1).

```r
##Create the empty vector for the loop.
rounded_rating <- c()

##Specify the conditions and effects of the loop.
for (i in imdb_rating){
  imdb_rating = ceiling(i)
  rounded_rating <- c(rounded_rating, imdb_rating)
  print(rounded_rating)
}
```

```
## [1] 10
## [1] 10  9
## [1] 10  9  8
## [1] 10  9  8  9
## [1] 10  9  8  9  8
## [1] 10  9  8  9  8  8
## [1] 10  9  8  9  8  8  9
## [1] 10  9  8  9  8  8  9  8
## [1] 10  9  8  9  8  8  9  8  8
##  [1] 10  9  8  9  8  8  9  8  8  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
```

```
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4
```

```
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5  2
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5  2  2
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5  2  2  4
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5  2  2  4  2
```

```
##View the new vector.
rounded_rating
```

```
##  [1] 10  9  8  9  8  8  9  8  8  9  8  9  6  8  7  7  8  8  9  9  9  9  7  7  8
## [26]  7  7  7  5  6  7  7  6  6  7  5  5  4  6 10  5  9  6  6  5  6  4  6  5  6
## [51]  4  4  3  5  5  3  3  4  5  2  4  5  2  2  4  2
```

Task 2 According to J. Angrist and J.-S. Pischke, what are the four most common questions we should position in the process of inference identification? List all four, you can use your own words.

The four questions J. Angrist and J.-S. Pischke claim we should position several questions during the process of inference identification. The first asks: "What is the causal relationship we are interested in?" The second is: "What experiment could ideally be designed and applied to observe the causal effect we are interested in?" The third question is: "What is the strategy to identify this causal relationship in the data?" The fourth question is: "How are you engaging in statistical inference with regard to your data?"

Directly quoting the authors, the four questions are: 1. What is the causal relationship of interest? 2. What experiment could ideally be used to capture the causal effect of interest? 3. What is your identification strategy? 4. What is your mode of statistical inference?

Task 3 Imagine you are conducting a randomized experiment to measure the impact of a new educational program on student test scores. You have a population of 1000 students, and you want to randomly assign half of them to a treatment group that will use a new program, while the other half will be assigned to a control group that will not use a new program. The test scores for both groups are assumed to follow a normal distribution: • The control group scores follow a normal distribution with a mean of 70 and a standard deviation of 10. • The treatment group scores are expected to increase by 5 points on average due to the program, so their scores follow a normal distribution with a mean of 75 and a standard deviation of 10.

1. Write an R code that would generate a random sample of test scores for both the treatment and control groups, using the normal distribution parameters described above.

```
set.seed(137)
treatment <- sample(c(rep(1, 1000/2), rep(0, 1000/2)), size = 1000, replace =
                      FALSE)

control <- rnorm(treatment == 0, mean = 70, sd = 10)
treat <- rnorm(treatment == 1, mean = 75, sd = 10)
```

2. Simulate the random assignment of 1000 students to the treatment and control groups. Make sure each student has an equal chance of being assigned to either group (Hint: check the code that we worked through in class to get an idea of how you can code that.)

```
set.seed(137)
treatment <- sample(c(rep(1, 1000/2), rep(0, 1000/2)), size = 1000, replace =
                      FALSE)
control <- rnorm(treatment == 0, mean = 70, sd = 10)
treat <- rnorm(treatment == 1, mean = 75, sd = 10)
```

3. Calculate average treatment effect (ATE) of the new program on the test scores

```
##True ATE (treatment mean - control mean)
true_ATE <- mean(75-70)
true_ATE ##5
```

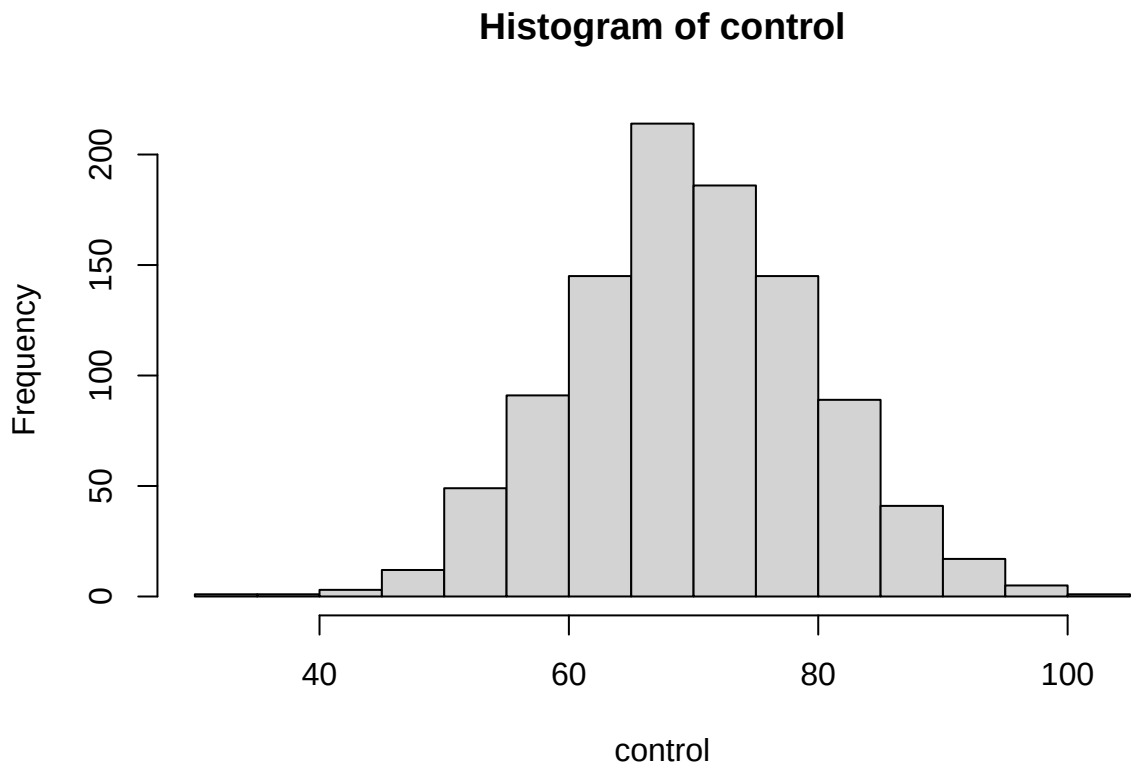
```
## [1] 5
```

```
est_ATE <- mean(treat - control)
est_ATE ##5.288144
```

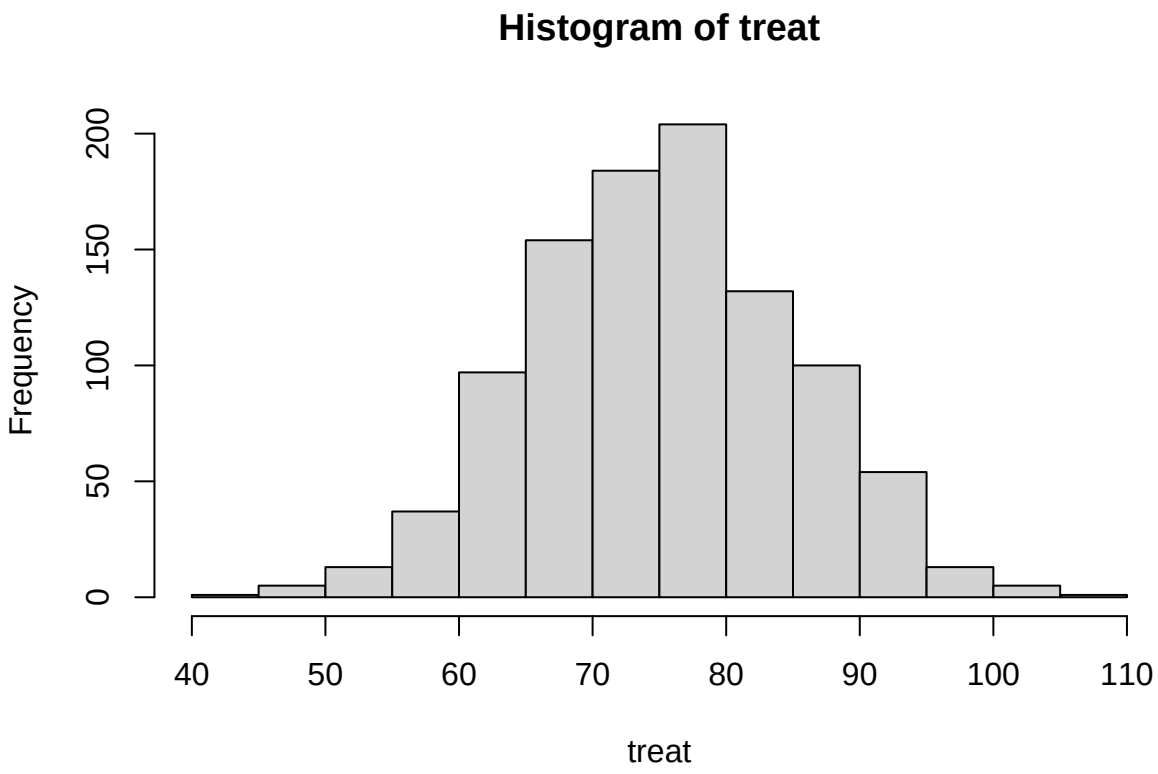
```
## [1] 5.288144
```

4. Create a histogram or a density plot to visualize the distribution of test scores for both — treatment and control — groups. Overlay the plots to compare the distributions.

```
##visualize control hisogram
chist <- hist(control)
```

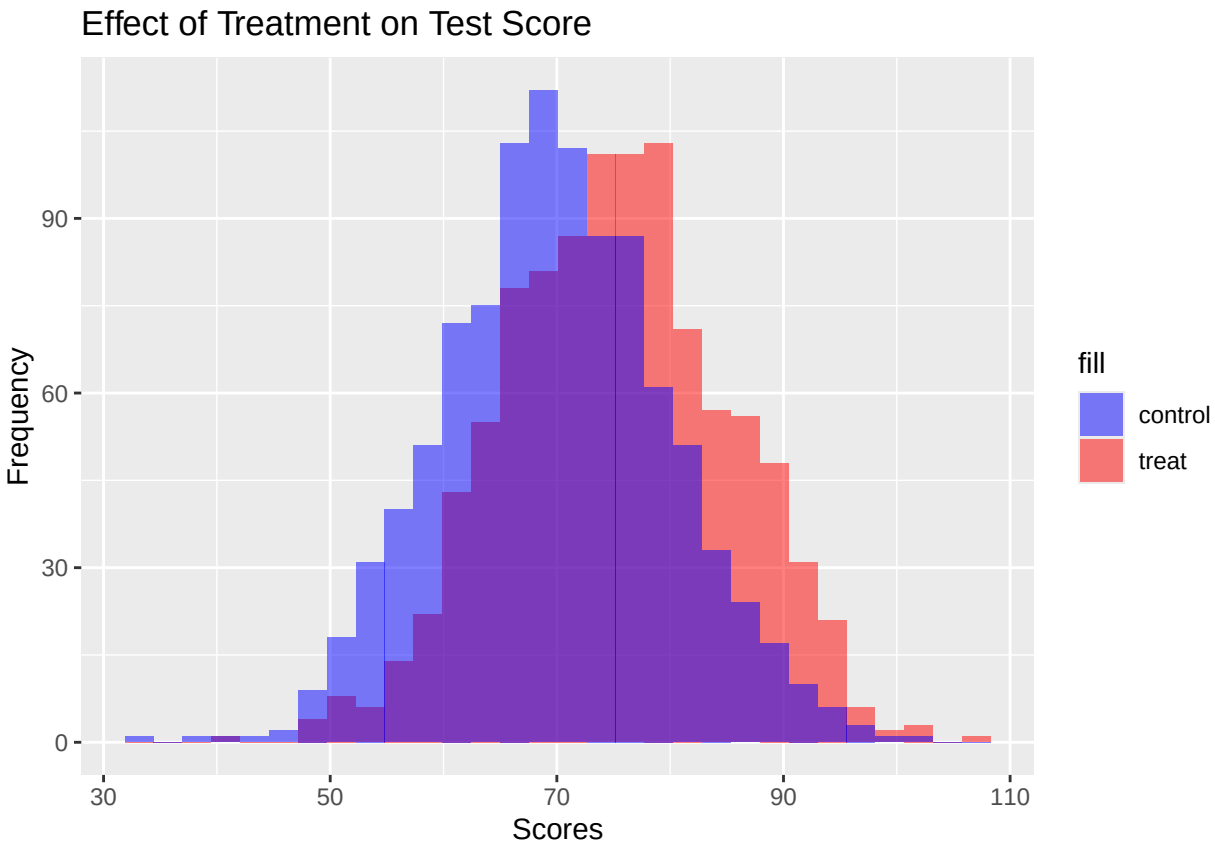



```
##visualize treatment histogram
thist <- hist(treat)
```




```
##Combine both plots
bothist <- ggplot() +
 geom_histogram(aes(x = treat, fill = "treat"), alpha = 0.5) +
 geom_histogram(aes(x = control, fill = "control"), alpha = 0.5) +
 scale_fill_manual(values = c("treat" = "red", "control" = "blue")) +
 labs(title = "Effect of Treatment on Test Score", x = "Scores", y = "Frequency"
      )
print(bothist)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


Effect of Treatment on Test Score
Frequency
90
60
30
0
30
50
70
90
110
Scores
fill
control
treat