Lesson 7: Chapter 14 & 15

0. Load libraries

```
knitr::opts_chunk$set(echo = TRUE)

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1

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

```
library(rlang)
```

```
##
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':
##
##     %@%, as_function, flatten, flatten_chr, flatten_dbl, flatten_int,
##     flatten_lgl, flatten_raw, invoke, list_along, modify, prepend,
##     splice
```

```
# A library for string operations
library(stringr)

# A library for working with Categorical variables, i.e. factors
library(forcats)


# Load NYC flight dataset
library(nycflights13)
```

Here is the link to package *stringr* and package *forcats*.

1. Chapter 14 Strings

stringr has a lot of functions on string manipulation. Function usually starts with str_

- combining strings: str_c()
- spliting strings: str_split()
- subsetting strings: str_sub()
- detect matches: str_detect()
- replacing matches: str_replace()

```
str_c("x", "y", sep = ", ")
```

```
## [1] "x, y"
```

```
x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 3)
```

```
## [1] "App" "Ban" "Pea"
```

```
x <- c("apple", "banana", "pear")
str_detect(x, "e")
```

```
## [1]  TRUE FALSE  TRUE
```

- matching strings with **regular expressions**

Section $14.3 has a brief introduction about **regular expressions**, which is widely used in many other languages, such as SQL, Python, etc. If you are not familiar with it, I'd recommend read section $14.3 and practice it.

2. Chapter 15 Factors

In most of cases, we'd like to convert character variables/categorical variables into *Factor* in R.

```
x1 <- c("Dec", "Apr", "Jan", "Mar")

sort(x1)
```

```
## [1] "Apr" "Dec" "Jan" "Mar"
```

We can set up a level vector.

```
month_levels <- c(
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
```

Then create a factor from x1 and the levels.

```
y1 <- factor(x1, levels = month_levels)
y1
```

```
## [1] Dec Apr Jan Mar
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
sort(y1)
```

```
## [1] Jan Mar Apr Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

To get familar with gss_cat survey data.

```
str(gss_cat)
```

```
## tibble [21,483 x 9] (S3: tbl_df/tbl/data.frame)
##  $ year   : int [1:21483] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ marital: Factor w/ 6 levels "No answer","Never married",..: 2 4 5 2 4 6 2 4 6 6 ...
##  $ age    : int [1:21483] 26 48 67 39 25 25 36 44 44 47 ...
##  $ race   : Factor w/ 4 levels "Other","Black",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ rincome: Factor w/ 16 levels "No answer","Don't know",..: 8 8 16 16 16 5 4 9 4 4 ...
##  $ partyid: Factor w/ 10 levels "No answer","Don't know",..: 6 5 7 6 9 10 5 8 9 4 ...
##  $ relig  : Factor w/ 16 levels "No answer","Don't know",..: 15 15 15 6 12 15 5 15 15 15 ...
##  $ denom  : Factor w/ 30 levels "No answer","Don't know",..: 25 23 3 30 30 25 30 15 4 25 ...
##  $ tvhours: int [1:21483] 12 NA 2 4 1 NA 3 NA 0 3 ...
```

```
?gss_cat
```

```
## starting httpd help server ... done
```

Use levels() or count() to see the levels in a factor.

```
levels(gss_cat$race)
```

```
## [1] "Other"          "Black"          "White"          "Not applicable"
```

```
levels(gss_cat$rincome)
```

```
##  [1] "No answer"      "Don't know"     "Refused"        "$25000 or more"
##  [5] "$20000 - 24999" "$15000 - 19999" "$10000 - 14999" "$8000 to 9999"
##  [9] "$7000 to 7999"  "$6000 to 6999"  "$5000 to 5999"  "$4000 to 4999"
## [13] "$3000 to 3999"  "$1000 to 2999"  "Lt $1000"       "Not applicable"
```

```
gss_cat %>% count(rincome)
```

```
## # A tibble: 16 x 2
##    rincome            n
##    <fct>          <int>
```

```
##  1 No answer        183
##  2 Don't know       267
##  3 Refused          975
##  4 $25000 or more  7363
##  5 $20000 - 24999  1283
##  6 $15000 - 19999  1048
##  7 $10000 - 14999  1168
##  8 $8000 to 9999    340
##  9 $7000 to 7999    188
## 10 $6000 to 6999    215
## 11 $5000 to 5999    227
## 12 $4000 to 4999    226
## 13 $3000 to 3999    276
## 14 $1000 to 2999    395
## 15 Lt $1000         286
## 16 Not applicable  7043
```

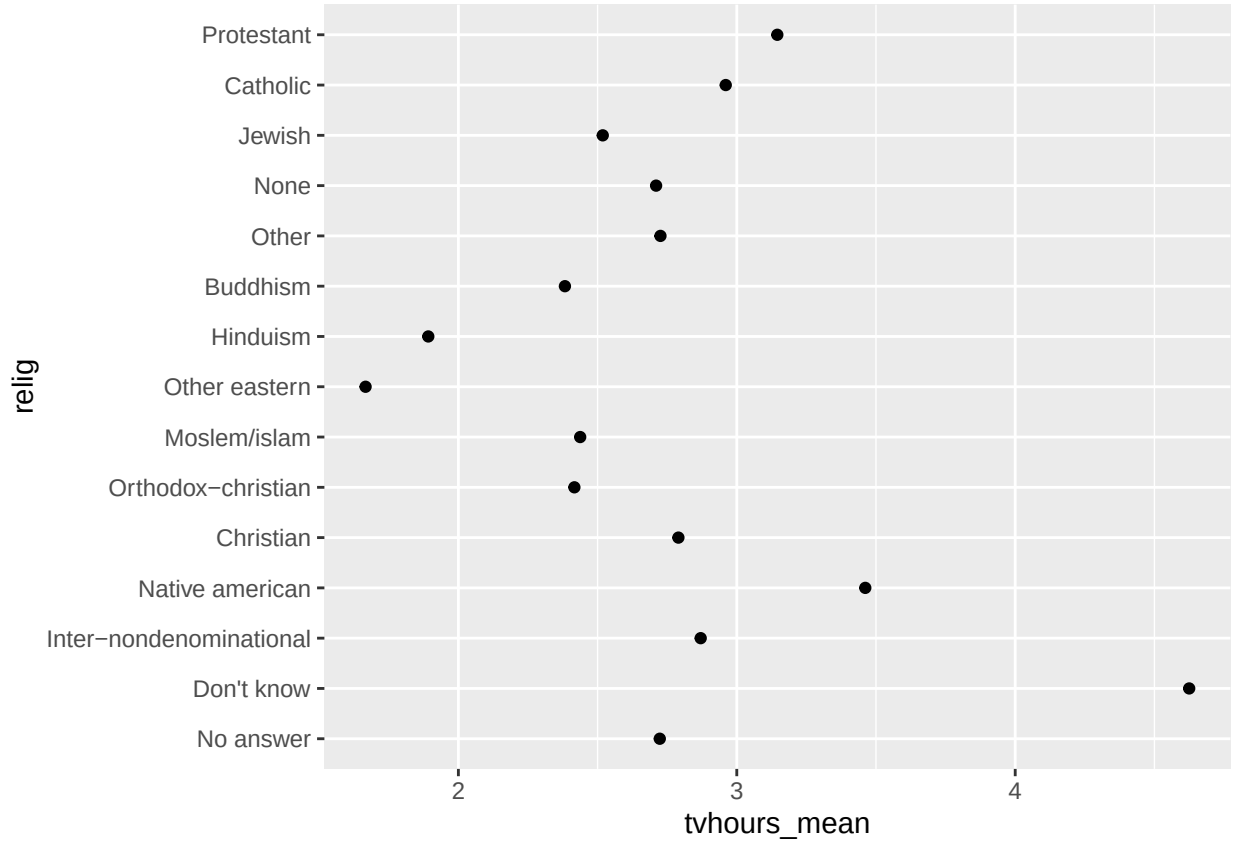
2.1 Modifying factor order in ggplot2

Explore the average number of hours spent watching TV per day across religions:

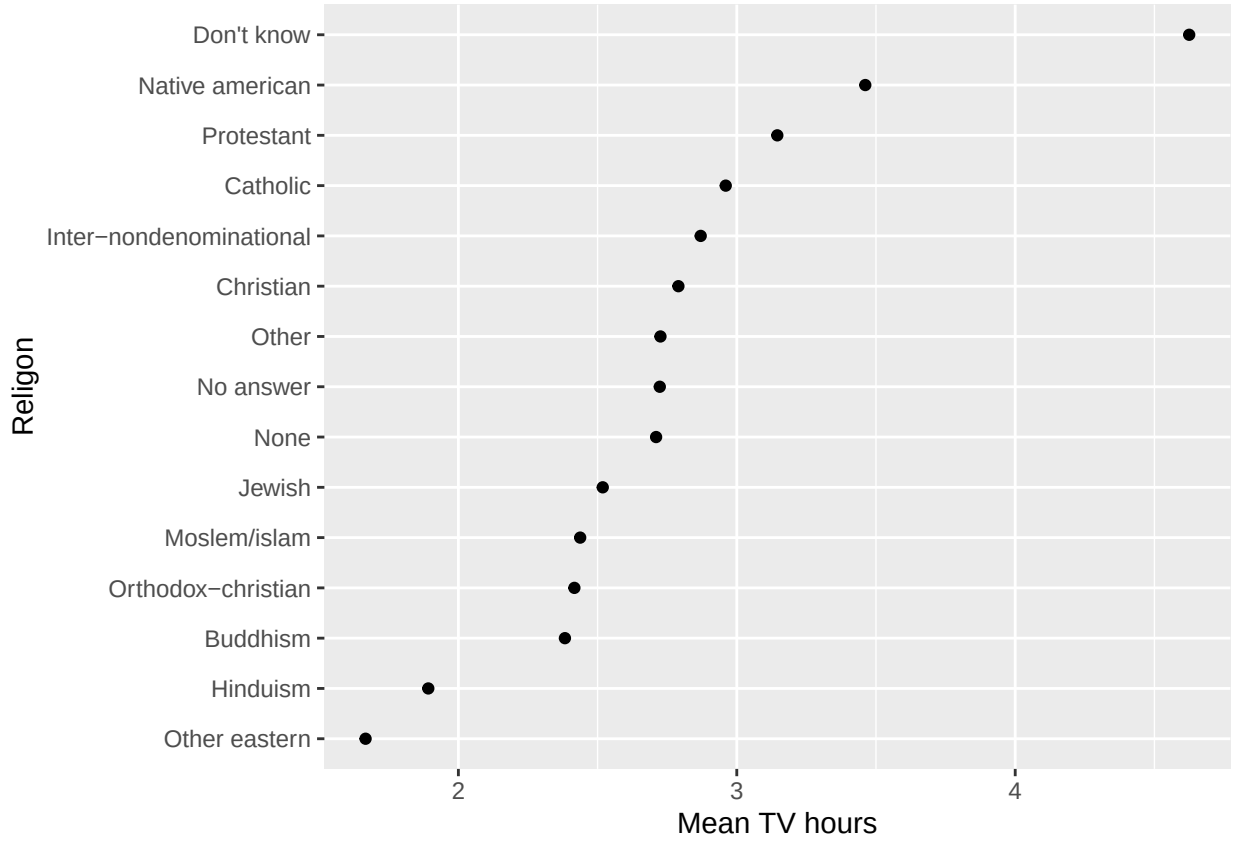
```
relig_summary <- gss_cat %>%
  group_by(relig) %>%
  summarise(
    age_mean = mean(age, na.rm = TRUE),
    tvhours_mean = mean(tvhours, na.rm = TRUE),
    n = n()
  )

View(relig_summary)

ggplot(relig_summary, aes(tvhours_mean, relig)) +
  geom_point()
```

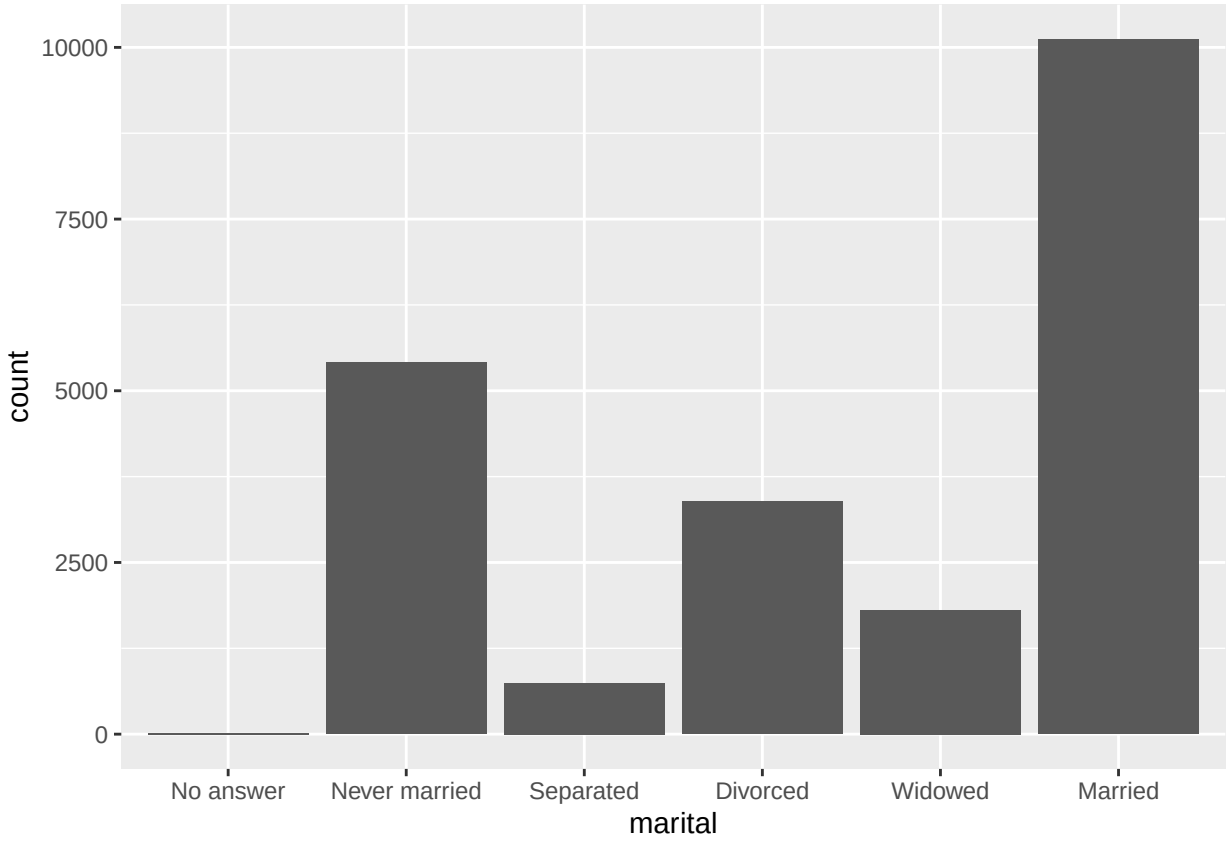



```
ggplot(relig_summary, aes(tvhours_mean, fct_reorder(relig, tvhours_mean))) +
  geom_point() +
  labs(x = "Mean TV hours", y = "Religon")
```



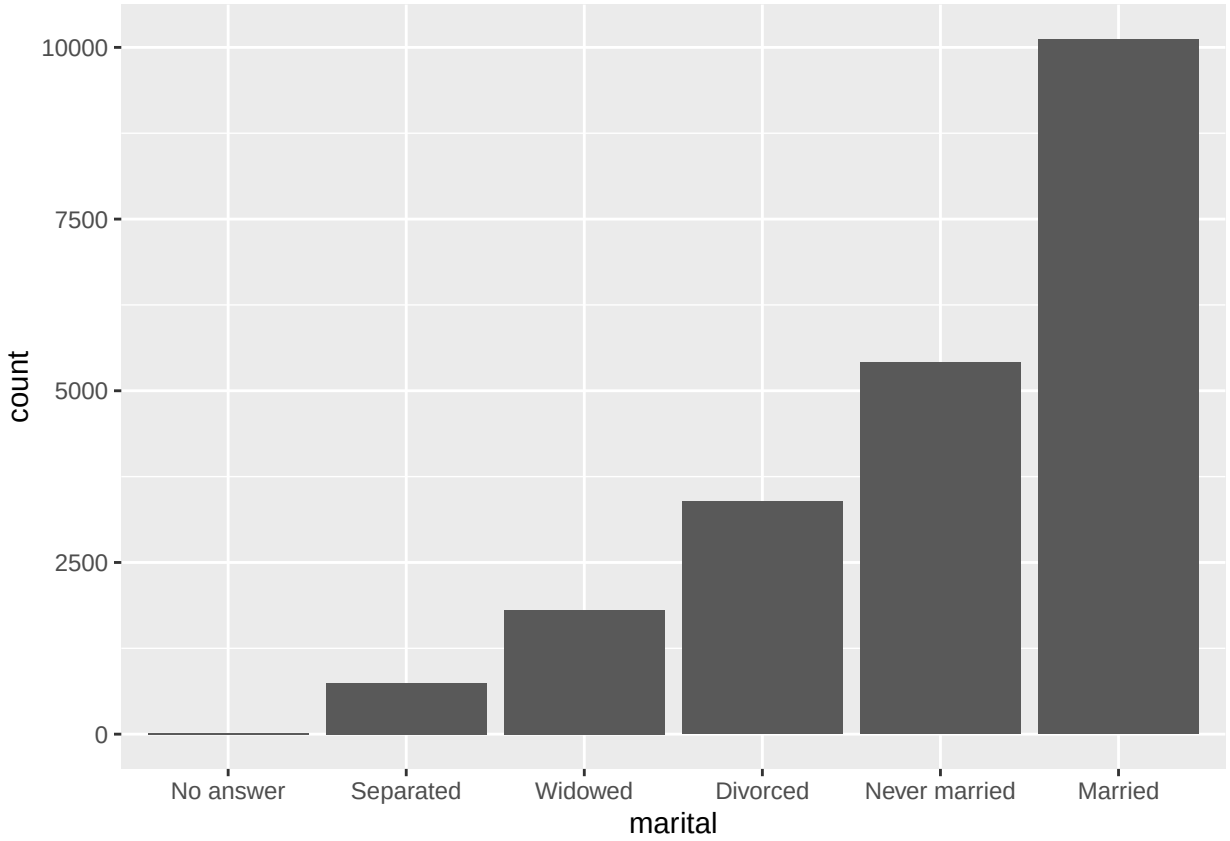
Number of person in each marital status in bar chart.

```
gss_cat %>%
  ggplot(aes(marital)) +
  geom_bar()
```



Make the bars in an upward order.

```
gss_cat %>%
  mutate(marital = marital %>%
           fct_infreq() %>%
           fct_rev()
         ) %>%
  ggplot(aes(marital)) +
    geom_bar()
```


10000
7500
5000
2500
0
count
No answer
Separated
Widowed
Divorced
Never married
Married
marital