Solutions Lecture 12

Exercise 1

Question 1:

First, load `ggplot2` and `dplyr`.

```
#library(tidyverse)
# or
library(ggplot2)
library(dplyr)
```

```
# Load clean irish_polls to environment:
load("irish_polls_clean.RData")
df = irish_polls_clean
names(df)[c(11,12)] <- c("Fianna.Fail", "Sinn.Fein")
```

Question 2:

Even though `Fieldwork.End` is from class `character`, using it to arrange the data frame will work. However, it is more intuitive to first transform this variable to `date` class.

```
class(df$Fieldwork.End)
```

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

Using `dplyr`, first we change the variable type of `Fieldwork.End` from `character` to `date`, then we order (in descending order, based on `Fieldwork.End`) and finally we slice the first 10 rows.

```
sliced_df <- df |>
  mutate(Fieldwork.End = as.Date(Fieldwork.End)) |>
  arrange(desc(Fieldwork.End))

sliced_df <- sliced_df[1:10, ]
```

Question 3

First, we select the relevant columns. Then, we use summarise to get the average scores. Then, we use some dplyr functions to get a convenient format for our df.

Note that NaNs are expected, since not all parties got votes for the selected period.

```r
# This is one way to arrange the data but not the only one:

new_df = sliced_df |>
  summarise(across("Fine.Gael":"Other", mean)) |>
  t() |>
  as.data.frame() |>
  as_tibble(rownames = "Party") |>
  rename(Average = V1)

# Note that the most important lines are the first 2 lines.
# The other lines just arrange the df in a convenient order for us to work on.
# Other alternatives are available.

new_df
```

```
## # A tibble: 12 x 2
##    Party                           Average
##    <chr>                             <dbl>
##  1 Fine.Gael                          25.3
##  2 Fianna.Fail                        17
##  3 Sinn.Fein                          30.7
##  4 Labour.Party                        4.5
##  5 Solidarity.People.Before.Profit     2.8
##  6 Social.Democrats                    4
##  7 Green.Party                         4.4
##  8 Aontú                               2.2
##  9 Renua.Ireland                      NA
## 10 Independent.Alliance               NA
## 11 Independents                       NA
## 12 Other                              NA
```

Question 4

We can just filter for values above 6%.

```r
new_df <- new_df |>
  filter(Average > 6)

new_df
```

```
## # A tibble: 3 x 2
##   Party       Average
##   <chr>         <dbl>
## 1 Fine.Gael      25.3
## 2 Fianna.Fail    17
## 3 Sinn.Fein      30.7
```

To create "other" category, we simply add a new row using `rbind`.

```r
new_row <- data.frame("Party" = "Other", "Average" = 100 - sum(new_df$Average))
```

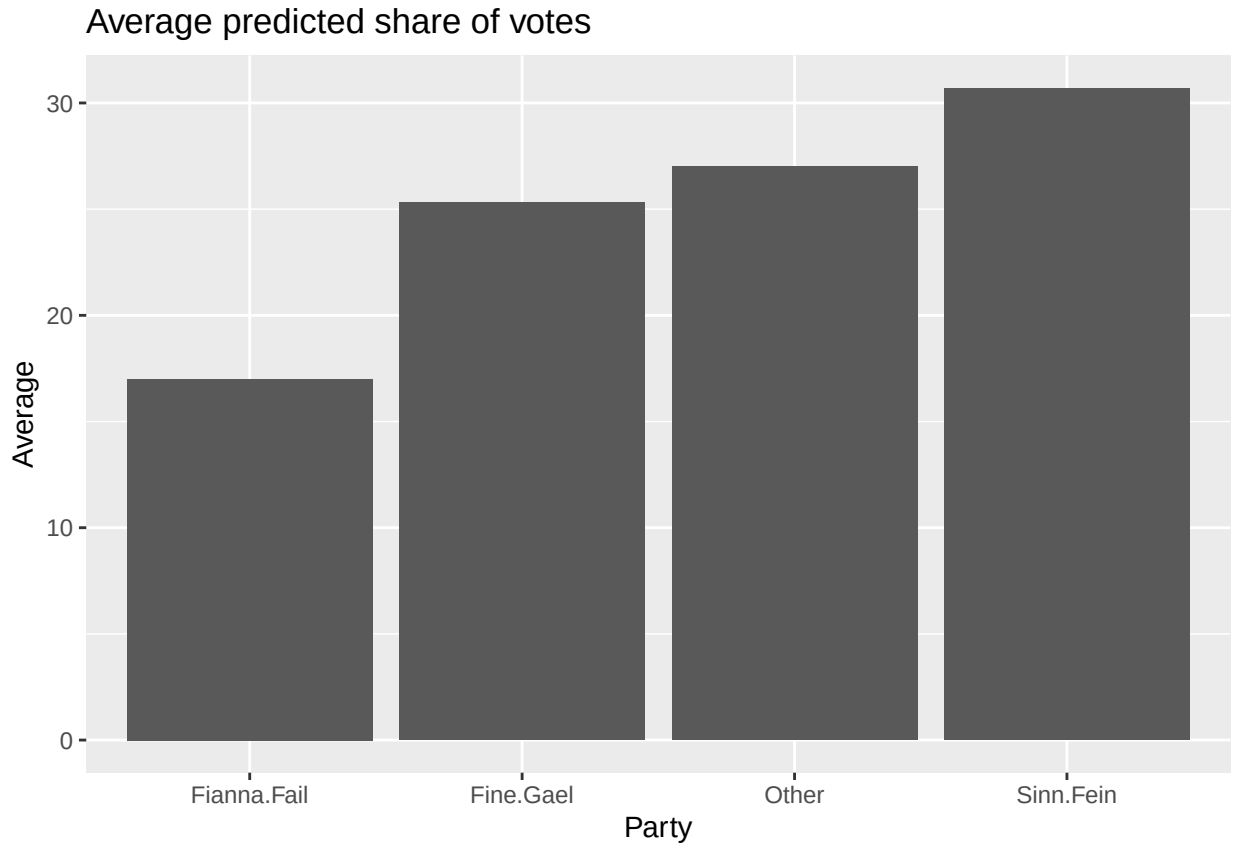
```r
new_df <- rbind(new_df, new_row)

new_df
```

```
## # A tibble: 4 x 2
##   Party       Average
##   <chr>         <dbl>
## 1 Fine.Gael      25.3
## 2 Fianna.Fail    17
## 3 Sinn.Fein      30.7
## 4 Other          27
```

Question 5

A default version of this with a main title can be achieved by the following code.

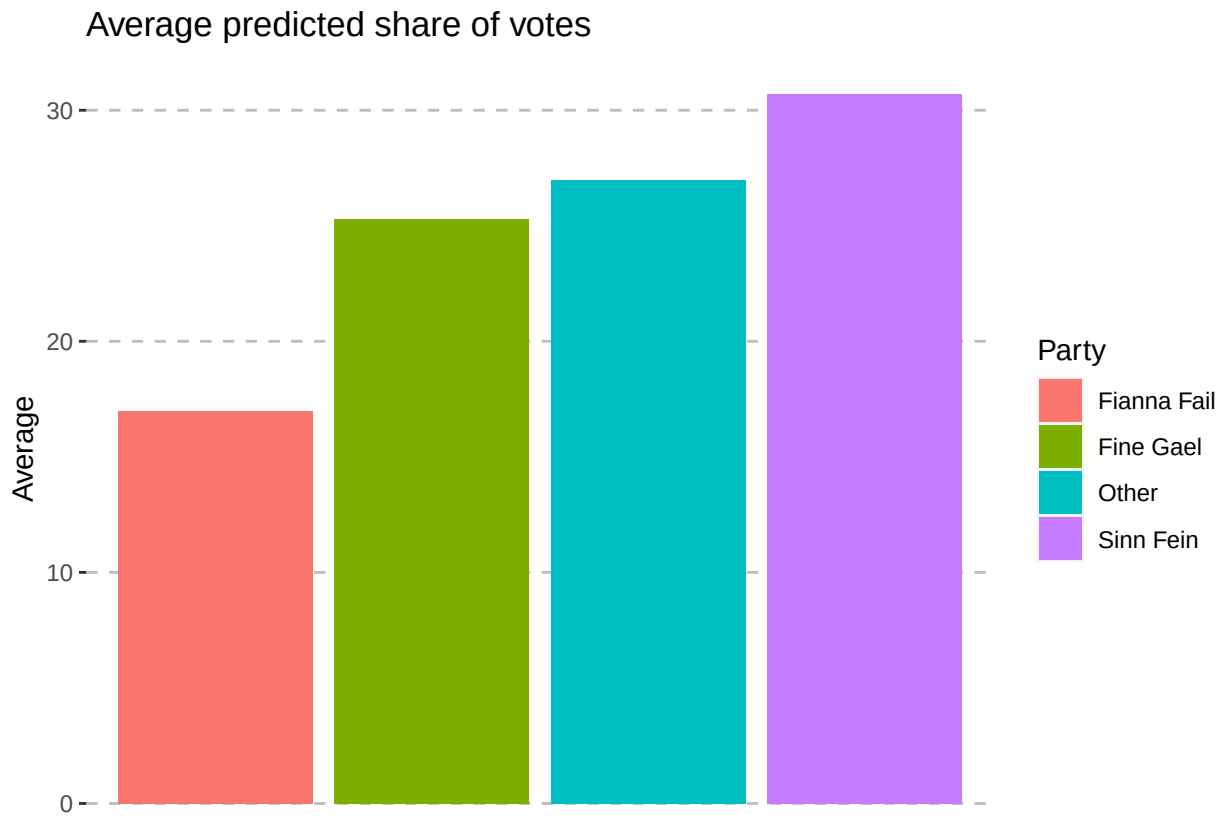
```r
new_df |>
  ggplot(aes(x = Party, y = Average)) +
  geom_bar(stat = "identity") +
  #geom_col() +  #same result
  ggtitle("Average predicted share of votes")
```



Notice that `geom_bar` by default computes the count, we need either to specify the stat to use here or use `geom_col`.

The previous chart is very basic, and can be greatly improved. For instance, we can rename the parties and fill the columns by the party, this will automatically create a legend. Then we remove the x title, text and ticks inside of `theme()` to do not have them two times. Alternatively, legend could be removed and x axis labels preserved! Finally we can make the background white and add a vertical grid.

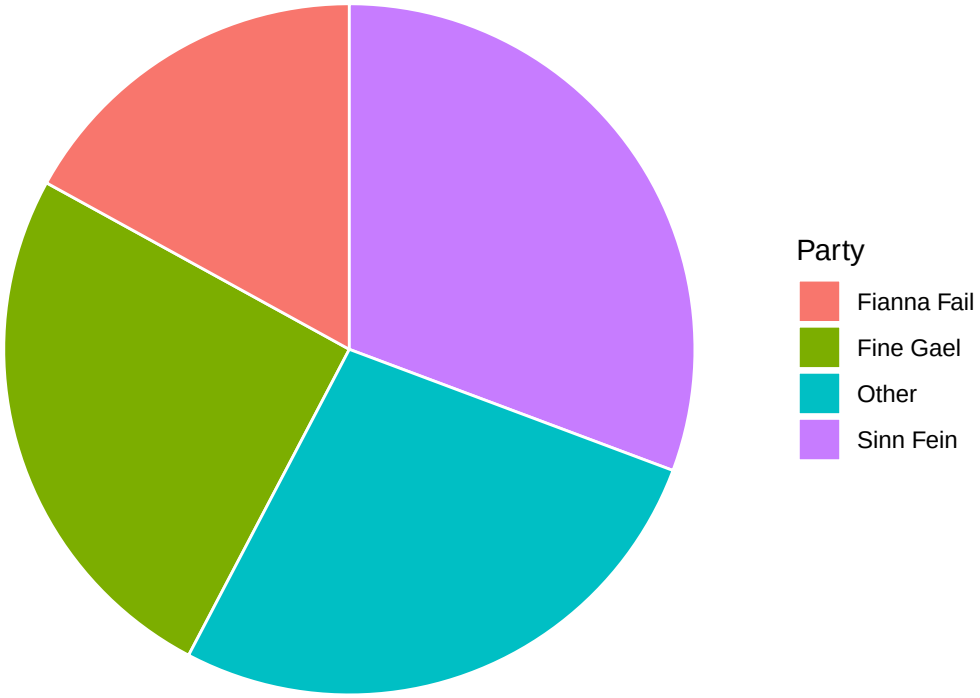
```
new_df <- new_df |>
  mutate(Party = c("Fine Gael", "Fianna Fail",
                   "Sinn Fein", "Other"))
new_df |>
  ggplot(aes(x = Party, y = Average, fill = Party)) +
  geom_col() +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        panel.background = element_rect(fill="white"),
        panel.grid.major = element_line(colour = "grey", linetype = 2),
        panel.grid.major.x = element_blank()) + # Remove vertical grid lines
  ggtitle("Average predicted share of votes")
```



Question 6

```
new_df |>
  ggplot(aes(x = "", y = Average, fill = Party)) +
```

```
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y") +
  theme_void() # remove background, grid, numeric labels
```



Exercise 2

Question 1, 2 & 3:

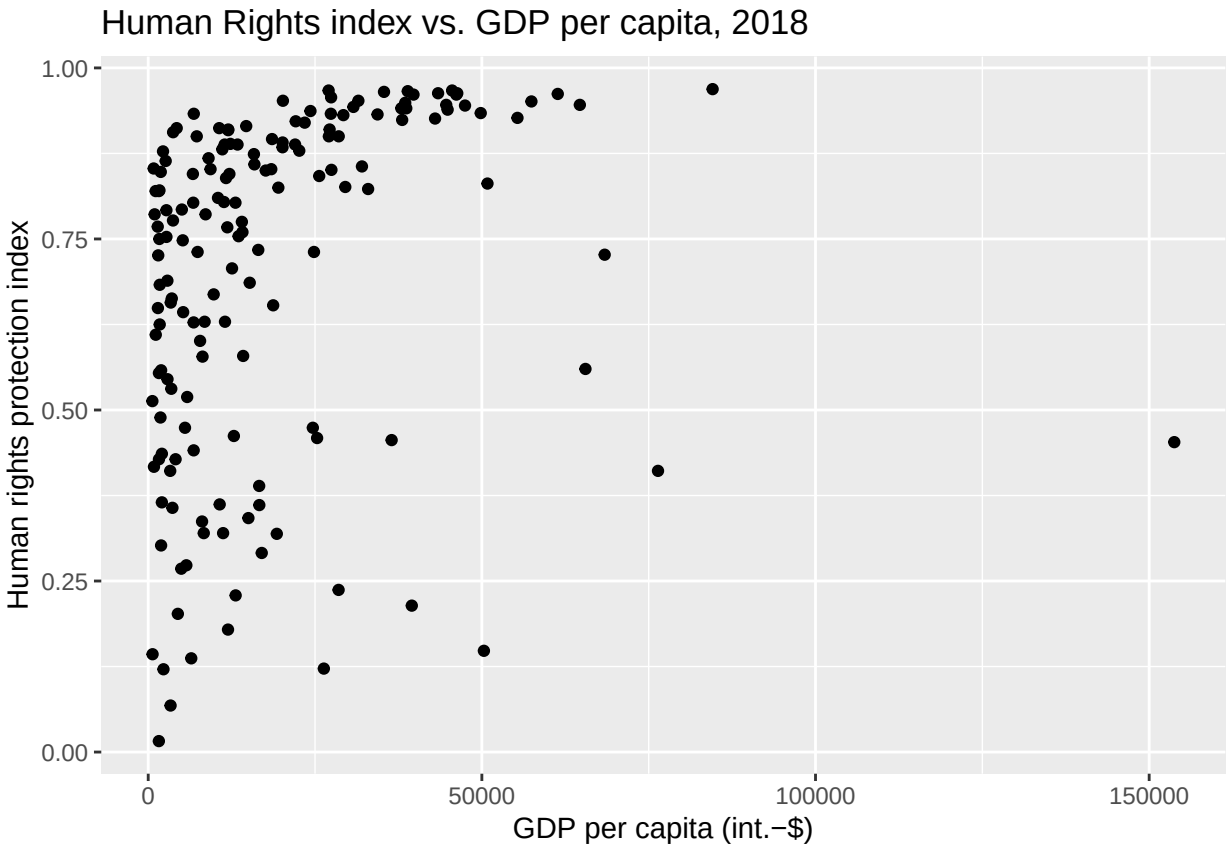
After downloading the data from *Our World in Data*, we can import the in R with:

```
my_data <- read.csv("human-rights-index-vs-gdp-per-capita.csv")
```

Question 4 & 5:

```
my_data <- my_data |>
  rename(HRI = civ_libs_vdem_owid,
         GDPpc = GDP.per.capita,
         Population = Population..historical.estimates. )

my_data |>
  filter(Year == 2018) |>
```

```
  ggplot(aes(y = HRI, x = GDPpc)) +
  geom_point() +
  xlab("GDP per capita (int.-$)") +
  ylab("Human rights protection index") +
  ggtitle("Human Rights index vs. GDP per capita, 2018")
```



Question 6, 7 & 8

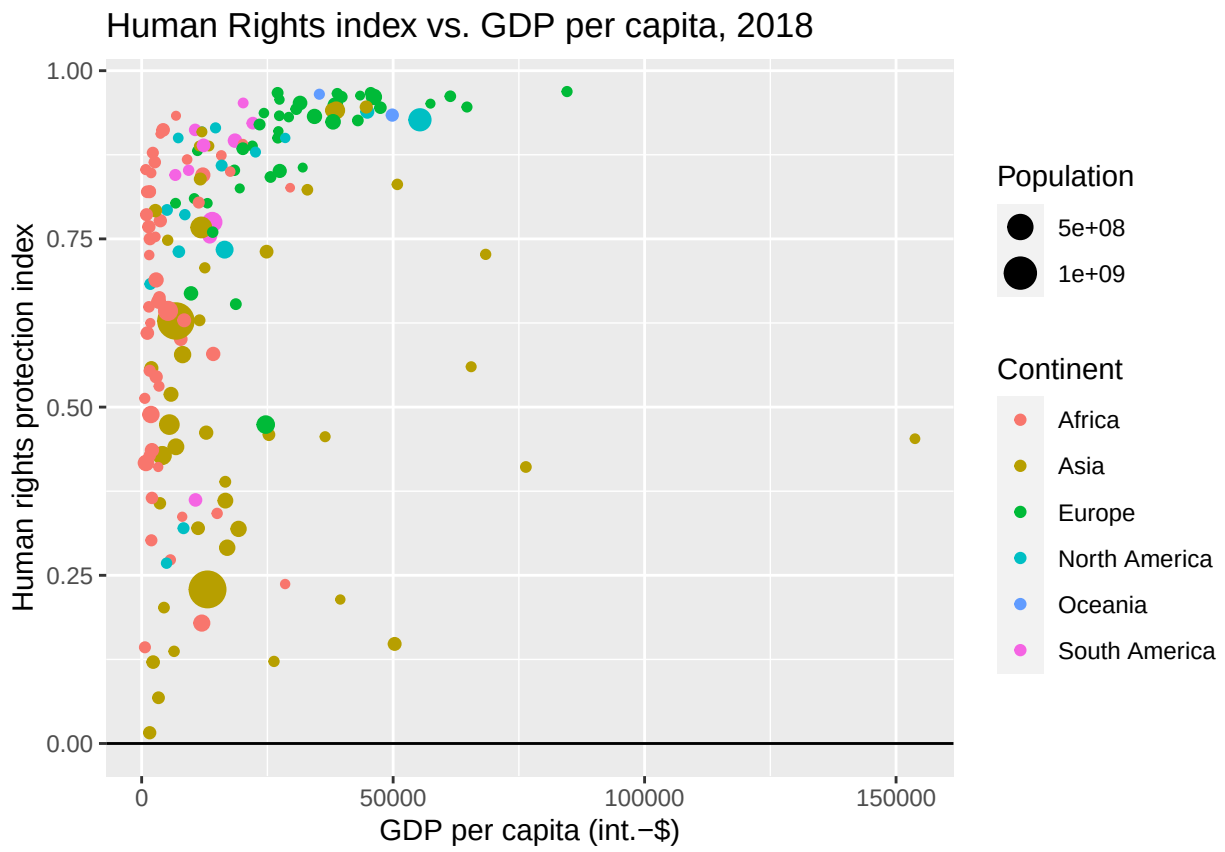
The trickiest part is retrieve the information for `Continent`. One option is reassigning the values for `Continent` in 2015 to 2018. Note that this solution relies in the fact that all the countries in 2018 are included in 2015 and that the 2015 contain information about the continent.

After inspection of the entries without continent information of 2015, it appears these are not souvereign and recognized countries, but parts of countries, categories of countries or continents. Therefore, we remove these from the dataset, since we are interested in the countries.

```
for (country in my_data[my_data$Year == 2018, "Entity"]){
  if (length(my_data[which(my_data$Entity == country & my_data$Year == 2015), "Continent"] != 0)){
      my_data[which(my_data$Entity == country & my_data$Year == 2018), "Continent"] <- my_data[which(my_
  }
  else {
    my_data[which(my_data$Entity == country & my_data$Year == 2018), "Continent"] <- ""
  }
}
```

```
my_data |>
  filter(Year == 2018) |>
  filter(Continent != "") |>
  ggplot(aes(y = HRI, x = GDPpc, color = Continent, size = Population)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  xlab("GDP per capita (int.-$)") +
  ylab("Human rights protection index") +
  ggtitle("Human Rights index vs. GDP per capita, 2018")
```

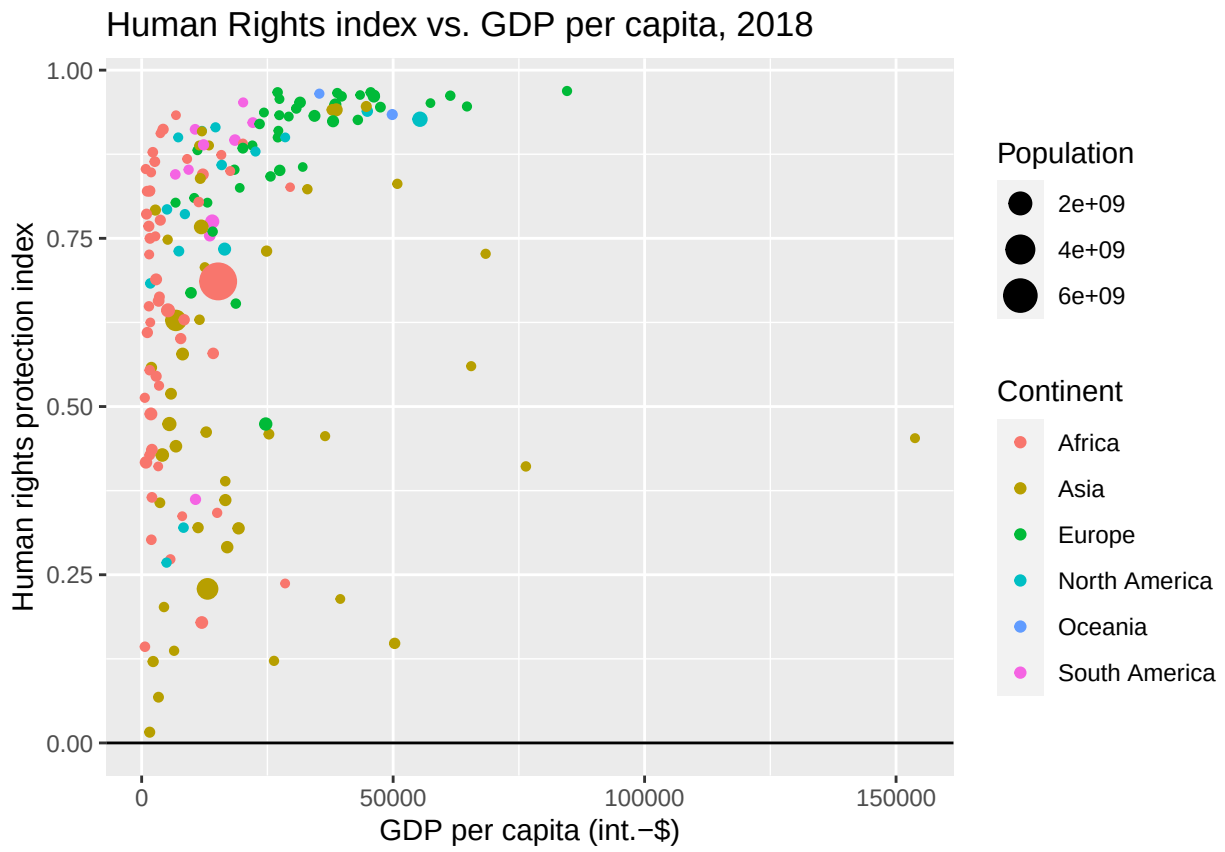
```
## Warning: Removed 80 rows containing missing values (`geom_point()`).
```



Another solution is to use `tidyr`, part of the `tidyverse` package. The function `fill()` does exactly what we are looking for, fills missing values in selected columns using the next or previous entry. To do this we just need to replace the `""` with `NA`. After this we can use `fill` and then just filter and plot as usual.

```
library(tidyr) # If not using tidyverse

my_data |>
  mutate(Continent = ifelse(Continent == "", NA, Continent)) |> # Create NA
  fill(Continent) |>
  filter(Year == 2018) |>
  ggplot(aes(y = HRI, x = GDPpc, color = Continent, size = Population)) +
  geom_point() +
  geom_hline(yintercept = 0) +
```

```
  xlab("GDP per capita (int.-$)") +
  ylab("Human rights protection index") +
  ggtitle("Human Rights index vs. GDP per capita, 2018")
```



Exercise 3

```
library(brolgar)
data(heights)
heights = as.data.frame(heights)
```

Question 1

```
heights |>
  filter(year == 1980) |>
  group_by(continent) |>
  count()
```

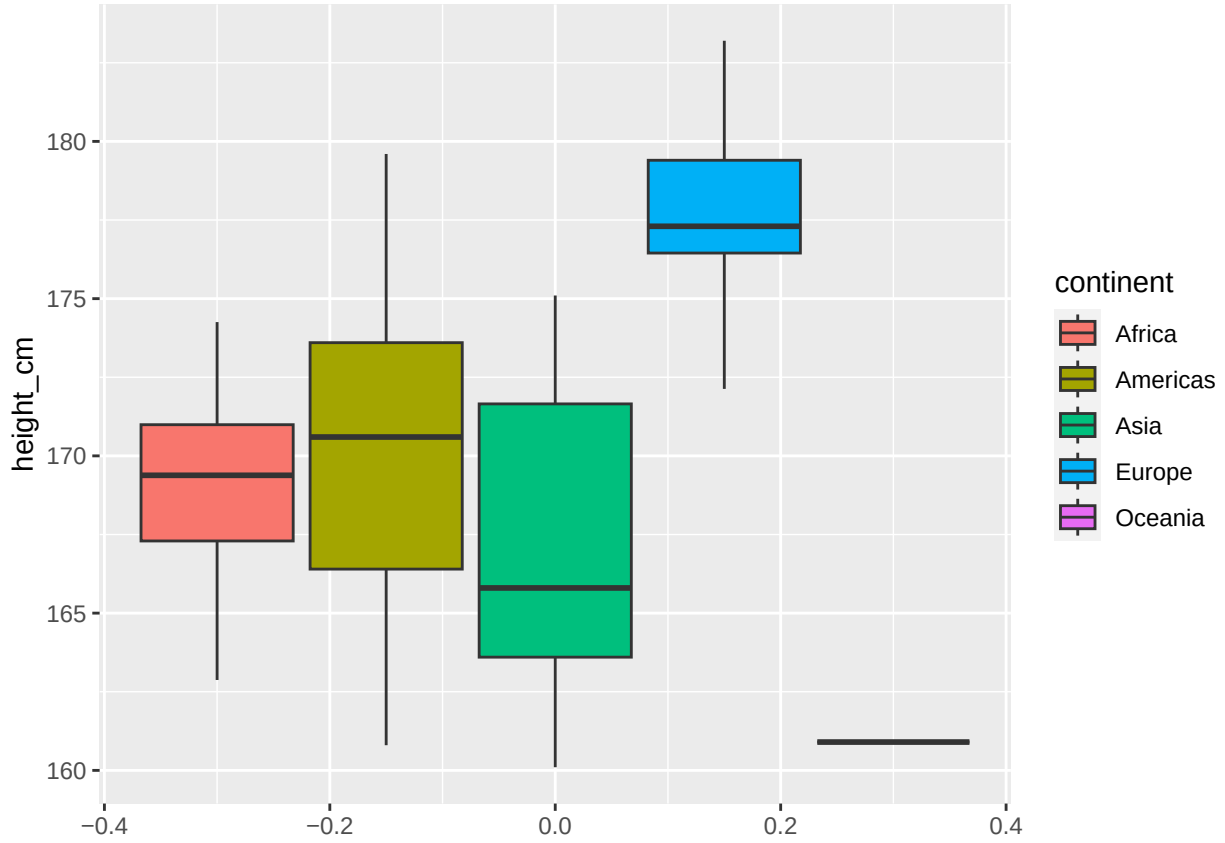
```
## # A tibble: 5 x 2
## # Groups:   continent [5]
##   continent     n
```

```
##   <chr>     <int>
## 1 Africa       34
## 2 Americas     17
## 3 Asia         21
## 4 Europe       19
## 5 Oceania       1
```

Question 2

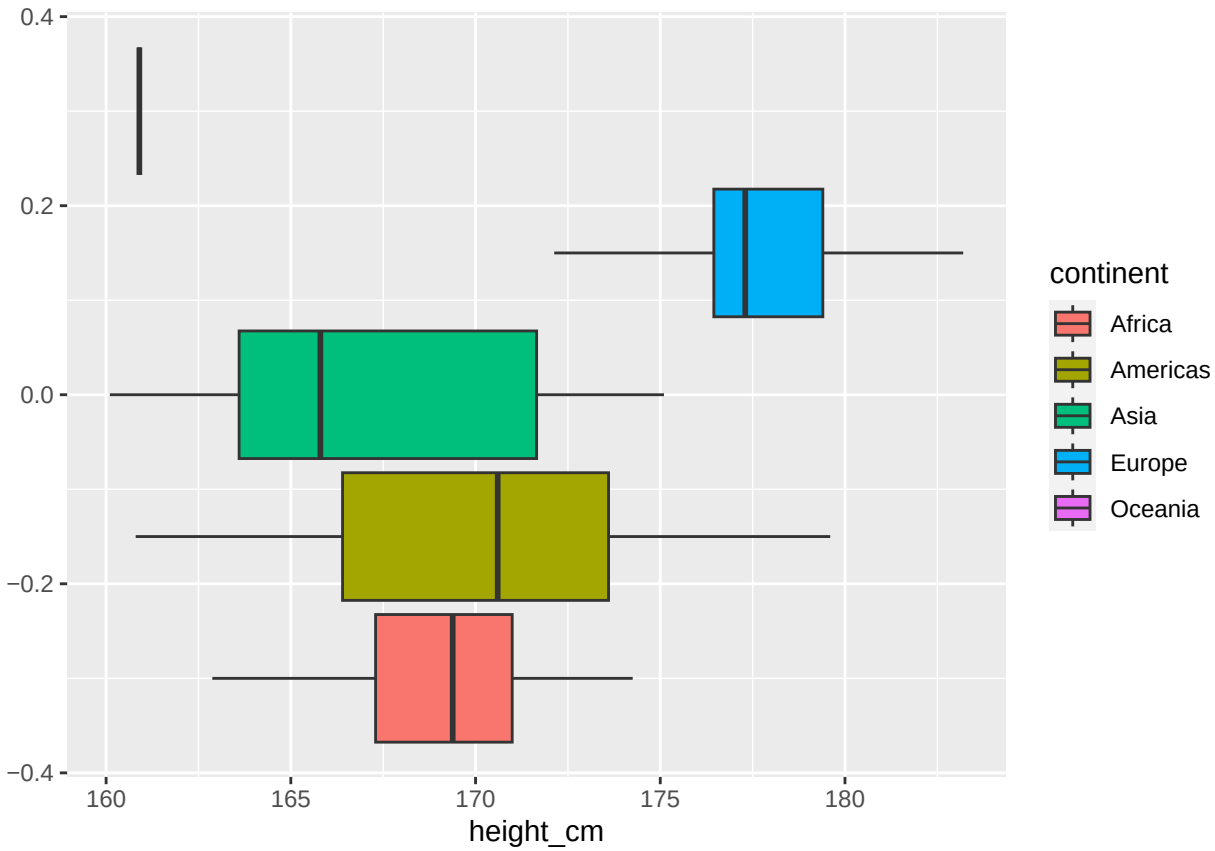
Horizontal:

```
heights |>
  filter(year == 1980) |>
  ggplot(aes(y = height_cm, fill = continent)) +
  geom_boxplot()
```



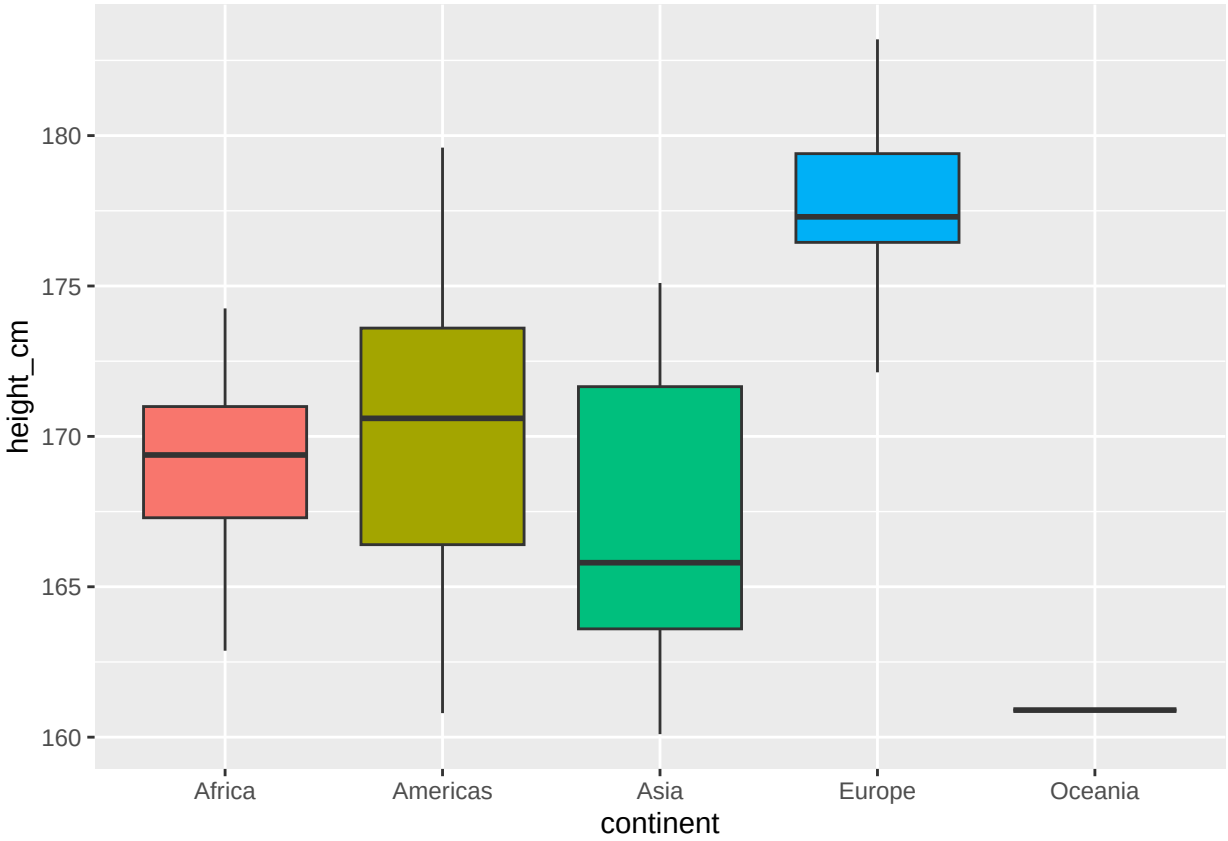
Vertical, this can be achieved either swaping `x` and `y` or using `coord_flip`.

```
heights |>
  filter(year == 1980) |>
  ggplot(aes(y = height_cm, fill = continent)) +
  geom_boxplot() +
  coord_flip()
```

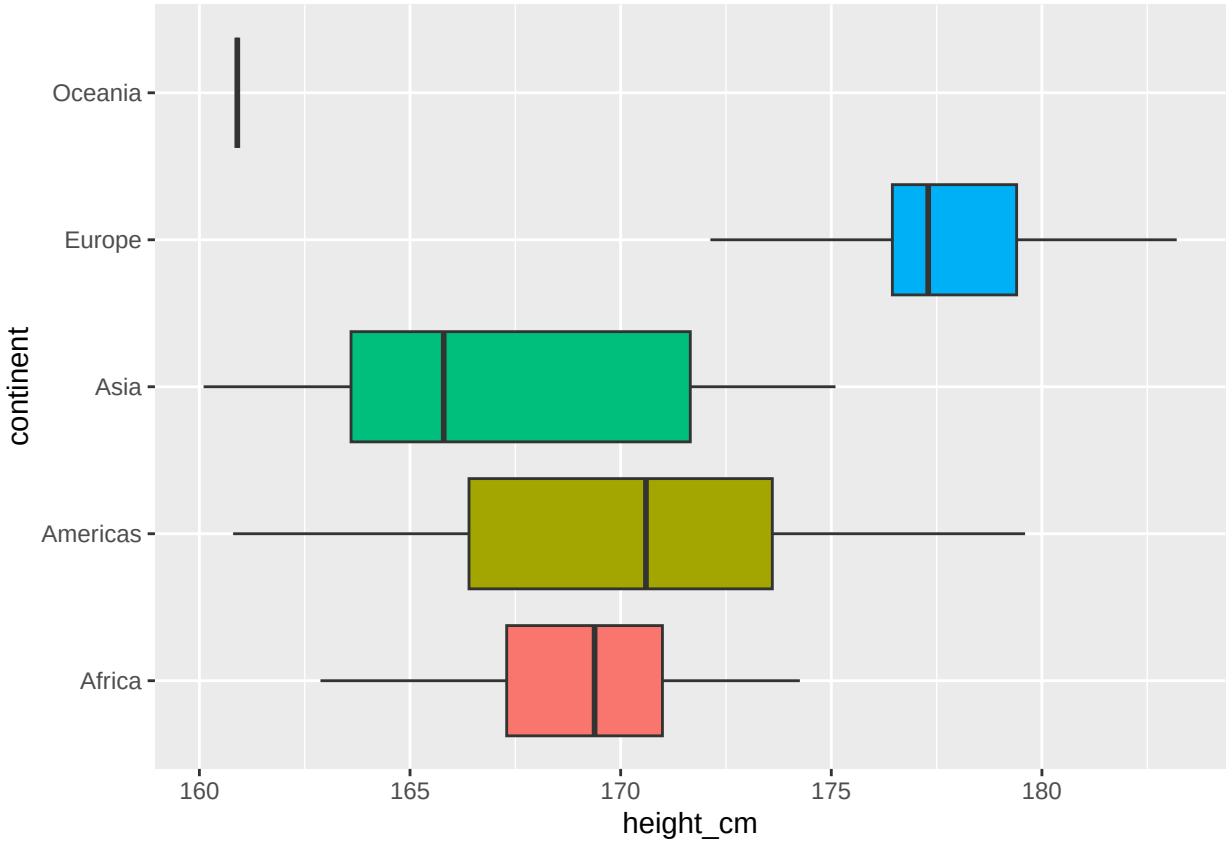


To remove the legend we just need to add `theme(legend.position = "none")`:

```
heights |>
  filter(year == 1980) |>
  ggplot(aes(x = height_cm, y = continent, fill = continent)) +
  theme(legend.position = "none") +
  geom_boxplot() +
  coord_flip()
```




```
heights |>
  filter(year == 1980) |>
  ggplot(aes(x = height_cm, y = continent, fill = continent)) +
  theme(legend.position = "none") +
  geom_boxplot()
```


Oceania
Europe
Asia
Americas
Africa
continent
160
165
170
175
180
height_cm