Week 4: More EDA

Monica Alexander

29/01/2022

By the end of this lab you should know

- the `group_by` function and how to use it to get summary statistics by group
- how to filter out missing values (NA) in one column or multiple columns, using `!is.na()` or `drop_na()`
- how to calculate the correlation coefficient between two variables
- how to get the number of observations by group
- how to calculate proportions by group
- `ggplot` basics; how to make each of the important types of graphs
  - histogram
  - bar chart
  - boxplot
  - line plot
  - scatter plot
- how to color / fill by group
- `fct_reorder` to reorder categorical values
- selecting only certain values of a variable using `%in%`
- if there's time: faceting

Read in data

Generally there are 3 steps in setting up any R session: 1. Choose the packages you are going to use and tell R to equip them by the `library()` command. + If you are using new packages, you will first need to install them, but remember to remove or comment the `install.packages()` line after you have done this. 2. Next we set our working directory. This tells R where all the files are and how to access them. Since we are using the `here` package as well, we are setting both our working directory and `here` package.
3. Read your files from the appropriate folder. Use `read_csv` command to read the file.

```
#install.packages("here")

# Call the packages that you are using
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   2.1.1     v forcats 0.5.1
```

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

```
library(here)
```

```
## here() starts at /Users/monicaalexander/src/soc6707-2022
```

```
# Read the file
gss <- read_csv(here("data/gss.csv")) # Only include data folder if your file is in a folder called dat
```

```
## Rows: 20602 Columns: 85
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (63): sex, place_birth_canada, place_birth_father, place_birth_mother, p...
## dbl (21): caseid, age, age_first_child, age_youngest_child_under_6, total_ch...
## lgl  (1): main_activity
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
country_ind <- read_csv(here("data/country_indicators.csv"))
```

```
## Rows: 1584 Columns: 9
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): country_code, country, region
## dbl (6): year, tfr, life_expectancy, child_mort, maternal_mort, gdp
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Handling Categorical Data

The group_by function

The group_by function allows you to get key summary statistics by group (levels of a categorical variable). Use in combination with summarize etc that we learnt last week.

e.g. mean life expectancy by region in 2017

```
country_ind %>%
  filter(year == 2017) %>%
  group_by(region) %>%
  summarize(mean_le = mean(life_expectancy))
```

```
## # A tibble: 10 x 2
##    region                       mean_le
##    <chr>                          <dbl>
##  1 Caucasus and Central Asia       75.2
##  2 Developed regions               82.3
##  3 Eastern Asia                    79.4
##  4 Latin America and Caribbean     77.6
##  5 Northern Africa                 76.9
##  6 Oceania                         71.5
##  7 South-eastern Asia              76.1
##  8 Southern Asia                   73.2
##  9 Sub-Saharan Africa              64.3
```

```
## 10 Western Asia                    77.2
```

e.g. mean age and standard deviation by marital status in GSS

```
gss %>%
  group_by(marital_status) %>%
  summarize(mean_age = mean(age),
            sd_age = sd(age)) %>%
  arrange(mean_age)
```

```
## # A tibble: 7 x 3
##   marital_status        mean_age sd_age
##   <chr>                    <dbl>  <dbl>
## 1 Single, never married     38.1  17.2
## 2 Living common-law         44.6  14.5
## 3 Separated                 54.5  13.7
## 4 Married                   54.9  14.8
## 5 Divorced                  61.0  11.4
## 6 <NA>                      65.8  12.9
## 7 Widowed                   73.0   8.47
```

Note that the above table shows the mean and sd of age for when marital status is missing (NA). We may want to remove those. To do this, use the `is.na` function in combination with the `!` (which means "not")

```
gss %>%
  filter(!is.na(marital_status)) %>%
  group_by(marital_status) %>%
  summarize(mean_age = mean(age),
            sd_age = sd(age)) %>%
  arrange(mean_age)
```

```
## # A tibble: 6 x 3
##   marital_status        mean_age sd_age
##   <chr>                    <dbl>  <dbl>
## 1 Single, never married     38.1  17.2
## 2 Living common-law         44.6  14.5
## 3 Separated                 54.5  13.7
## 4 Married                   54.9  14.8
## 5 Divorced                  61.0  11.4
## 6 Widowed                   73.0   8.47
```

Note dealing with missing data is a significant part of data analysis. While in some analysis we decide to exclude missing observations, take a moment and think about why some observations may be missing.

Calculating the correlation coefficient

To calculate the correlation coefficient between two quantitative (numerical/continous) variables, e.g. age and age at first marriage, use the `summarize` function. Notice that we need to remove rows with any NA values before doing the calculation. We can do this using `drop_na()`

```
gss %>%
  select(age, age_at_first_marriage) %>%
  drop_na() %>%
  summarise(correlation = cor(age, age_at_first_marriage))
```

```
## # A tibble: 1 x 1
##   correlation
```

```
##          <dbl>
## 1       -0.154
```

Counts and proportions

Counting the number of observations

Often we would like to include counts of observations in particular groups. To do this, use the `tally()` or `count()` function.

e.g. the number of people by province of residence in the GSS

```
gss %>%
  group_by(province) %>%
  tally()
```

```
## # A tibble: 10 x 2
##    province                      n
##    <chr>                     <int>
##  1 Alberta                    1728
##  2 British Columbia           2522
##  3 Manitoba                   1192
##  4 New Brunswick              1337
##  5 Newfoundland and Labrador  1094
##  6 Nova Scotia                1425
##  7 Ontario                    5621
##  8 Prince Edward Island        708
##  9 Quebec                     3822
## 10 Saskatchewan               1153
```

equivalent:

```
gss %>%
  count(province)
```

```
## # A tibble: 10 x 2
##    province                      n
##    <chr>                     <int>
##  1 Alberta                    1728
##  2 British Columbia           2522
##  3 Manitoba                   1192
##  4 New Brunswick              1337
##  5 Newfoundland and Labrador  1094
##  6 Nova Scotia                1425
##  7 Ontario                    5621
##  8 Prince Edward Island        708
##  9 Quebec                     3822
## 10 Saskatchewan               1153
```

Getting the proportion in each group

Also often useful to get proportion of total in each group:

```
gss %>%
  group_by(province) %>%
  tally() %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 10 x 3
##    province                     n   prop
##    <chr>                    <int>  <dbl>
##  1 Alberta                   1728 0.0839
##  2 British Columbia          2522 0.122
##  3 Manitoba                  1192 0.0579
##  4 New Brunswick             1337 0.0649
##  5 Newfoundland and Labrador 1094 0.0531
##  6 Nova Scotia               1425 0.0692
##  7 Ontario                   5621 0.273
##  8 Prince Edward Island       708 0.0344
##  9 Quebec                    3822 0.186
## 10 Saskatchewan              1153 0.0560
```

equivalent

```
gss %>%
  count(province) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 10 x 3
##    province                     n   prop
##    <chr>                    <int>  <dbl>
##  1 Alberta                   1728 0.0839
##  2 British Columbia          2522 0.122
##  3 Manitoba                  1192 0.0579
##  4 New Brunswick             1337 0.0649
##  5 Newfoundland and Labrador 1094 0.0531
##  6 Nova Scotia               1425 0.0692
##  7 Ontario                   5621 0.273
##  8 Prince Edward Island       708 0.0344
##  9 Quebec                    3822 0.186
## 10 Saskatchewan              1153 0.0560
```

In-class exercise

1. What proportion of age of first marriage is missing?
2. What are the proportion of individuals have worked last week (worked_last_week)? What proportion of this variable is missing?
3. Within non-missing individuals who have worked last week, how many and what proportion worked full-time (full_part_time_work)?

`ggplot`

`ggplot` is a powerful visualization package. It provides many options to make beautiful graphs, maps, plots of all sort. Each example we look at today.

Histograms

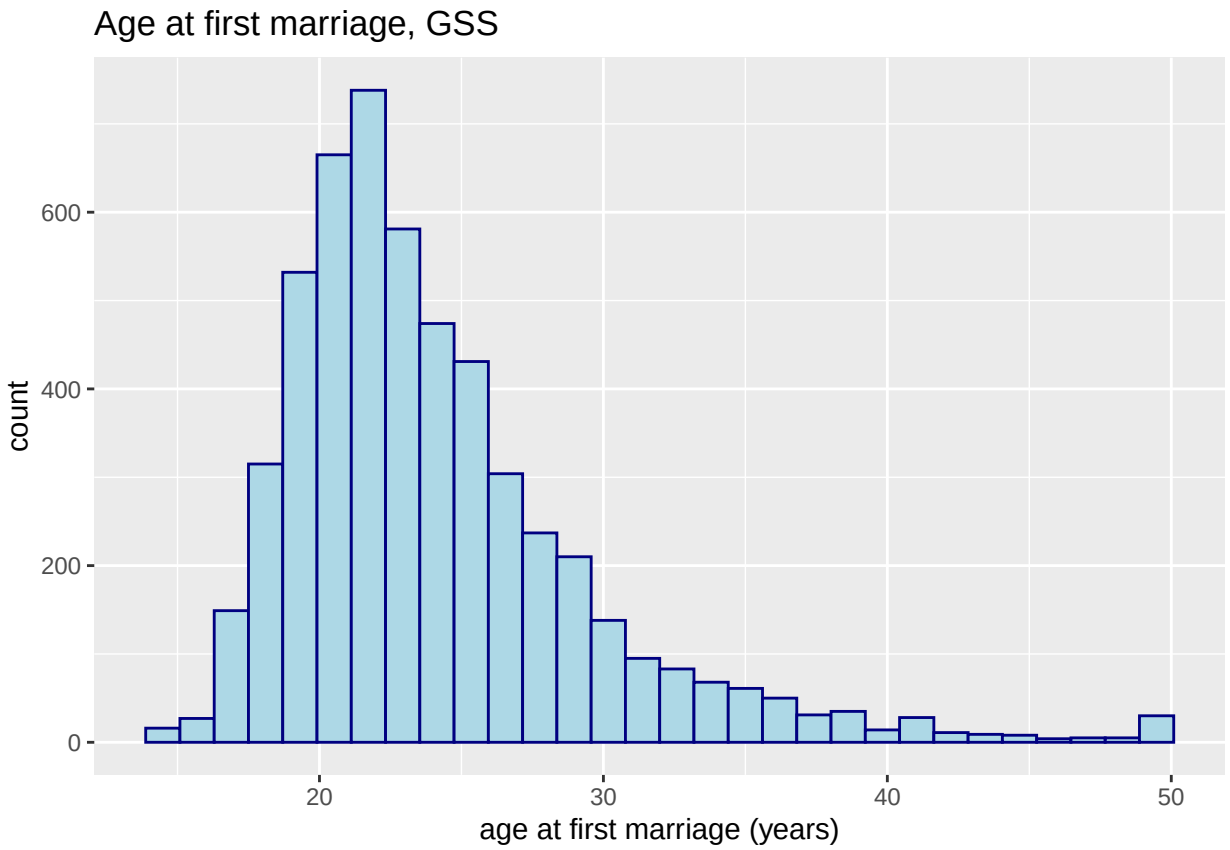
Note for histograms, bar chats, box plots, `fill` is the main color choice (`color` changes the outline)

```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy") +
```

```
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")
```

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

```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```



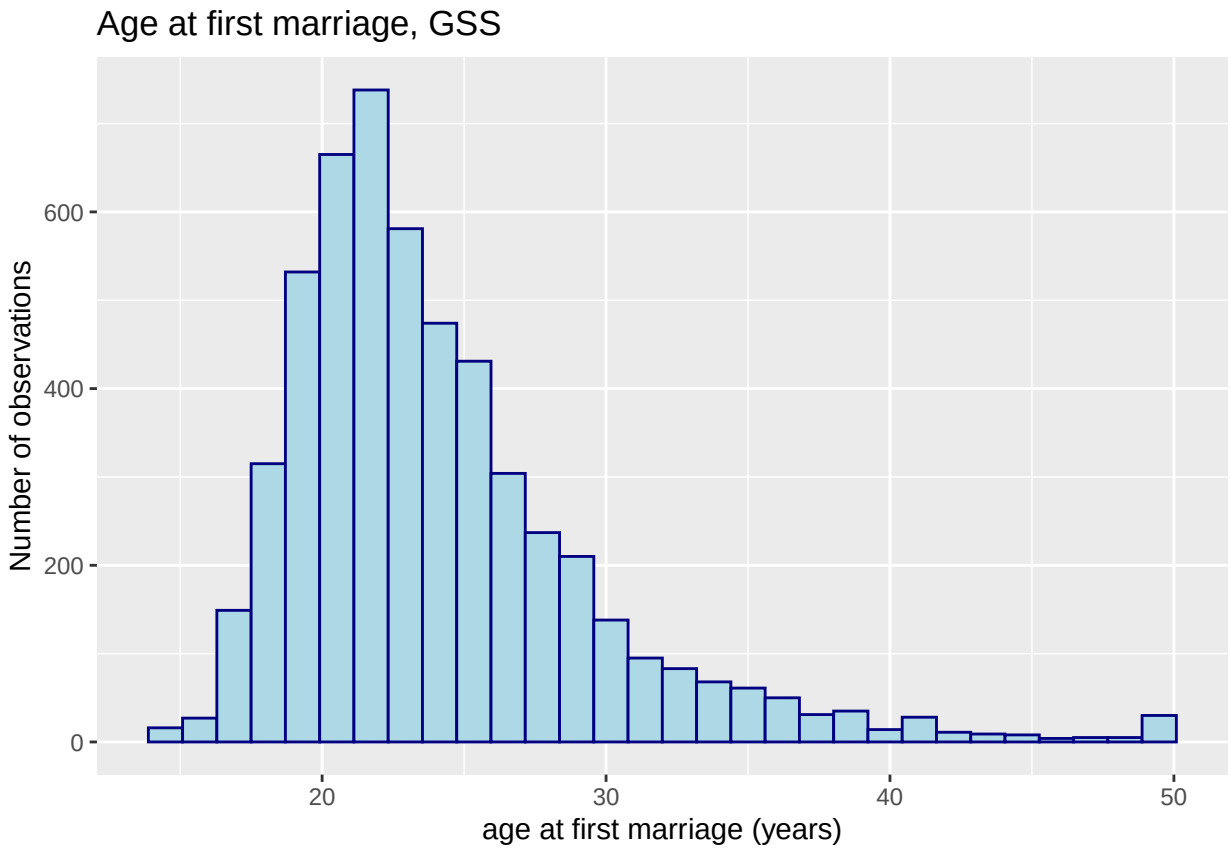
Note that you can also save the plot as an object and then print it

```
my_plot <- ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy")+
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")

# print
my_plot + ylab("Number of observations")
```

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

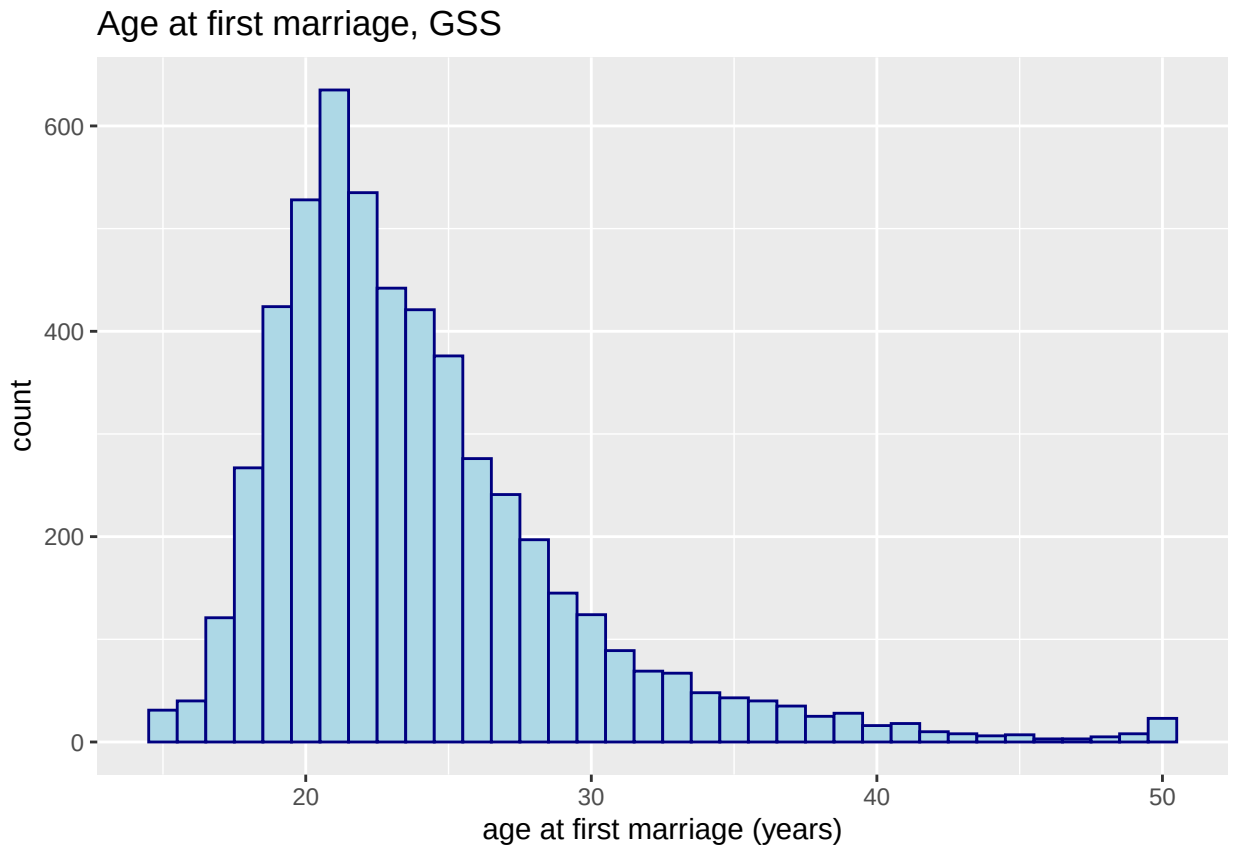
```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```



Histograms select a `binwidth` or section of the data and then count how many of the observations fall within that. Histograms look different depending on the size of the bins. You can also supply the number of bins that you want to create.

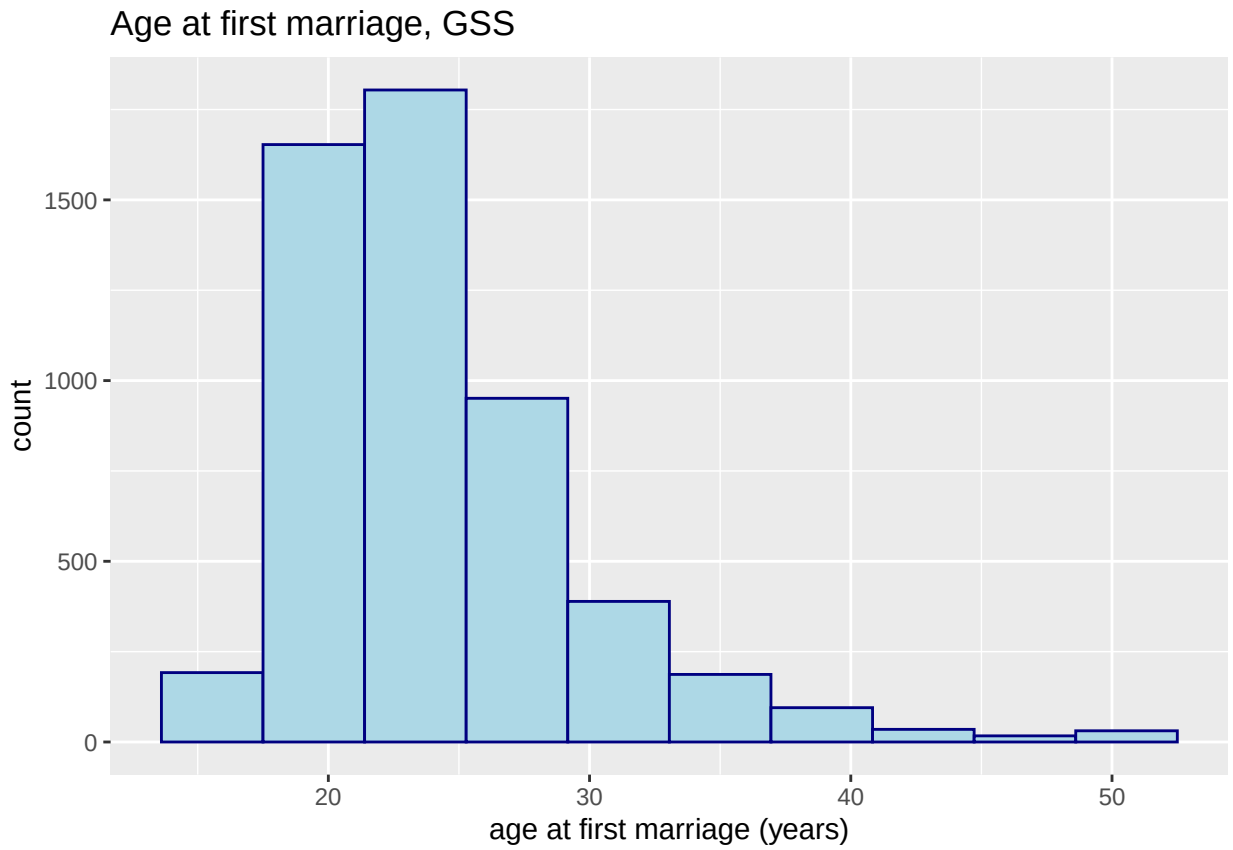
```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy", binwidth = 1) +
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")
```

```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```




```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy", bins = 10)+
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")
```

```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```

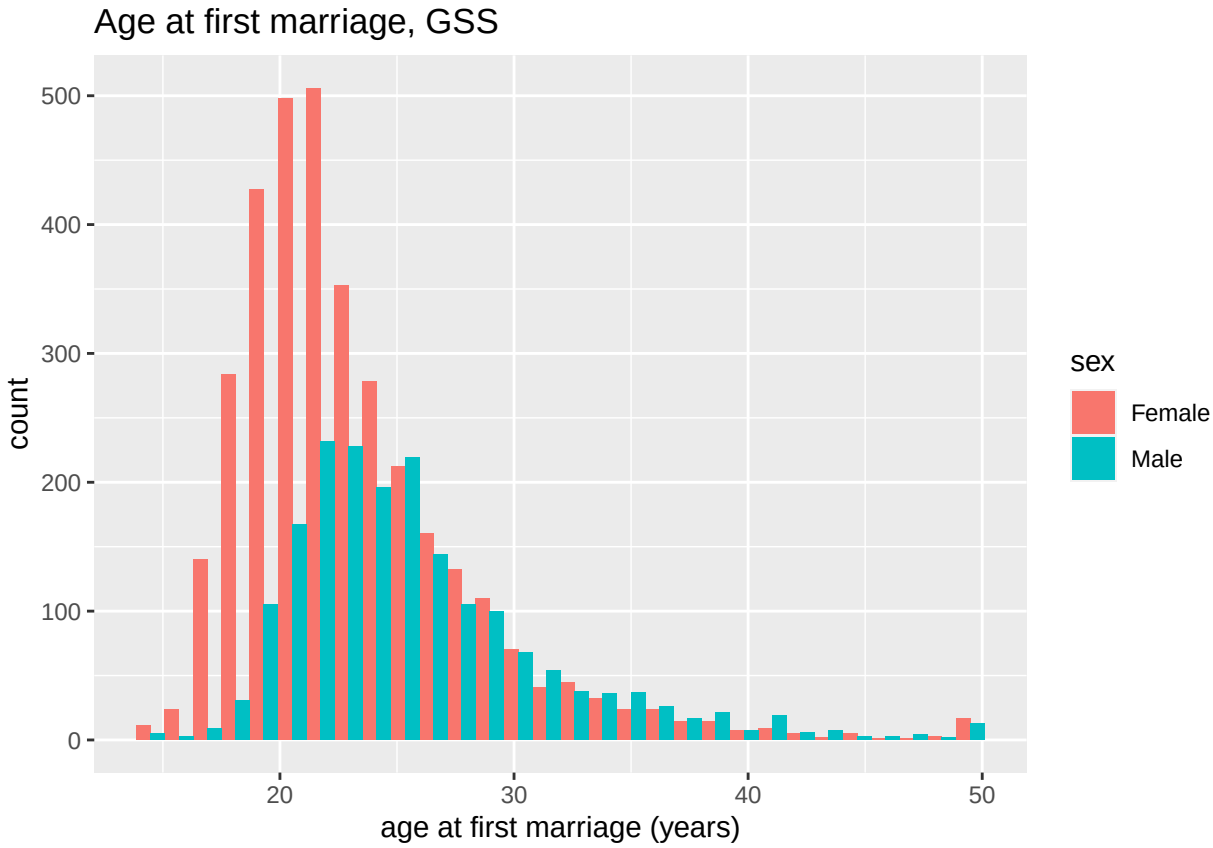


We can also plot by another variable to compare the plots by the categories of the variable. For example, we look at plots by sex:

```
ggplot(data = gss, aes(age_at_first_marriage, fill = sex)) +
  geom_histogram(position = 'dodge') +
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")
```

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

```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```



Bar charts

Let's plot the proportion of respondents by province as a bar chart. First save the proportions as a new data frame

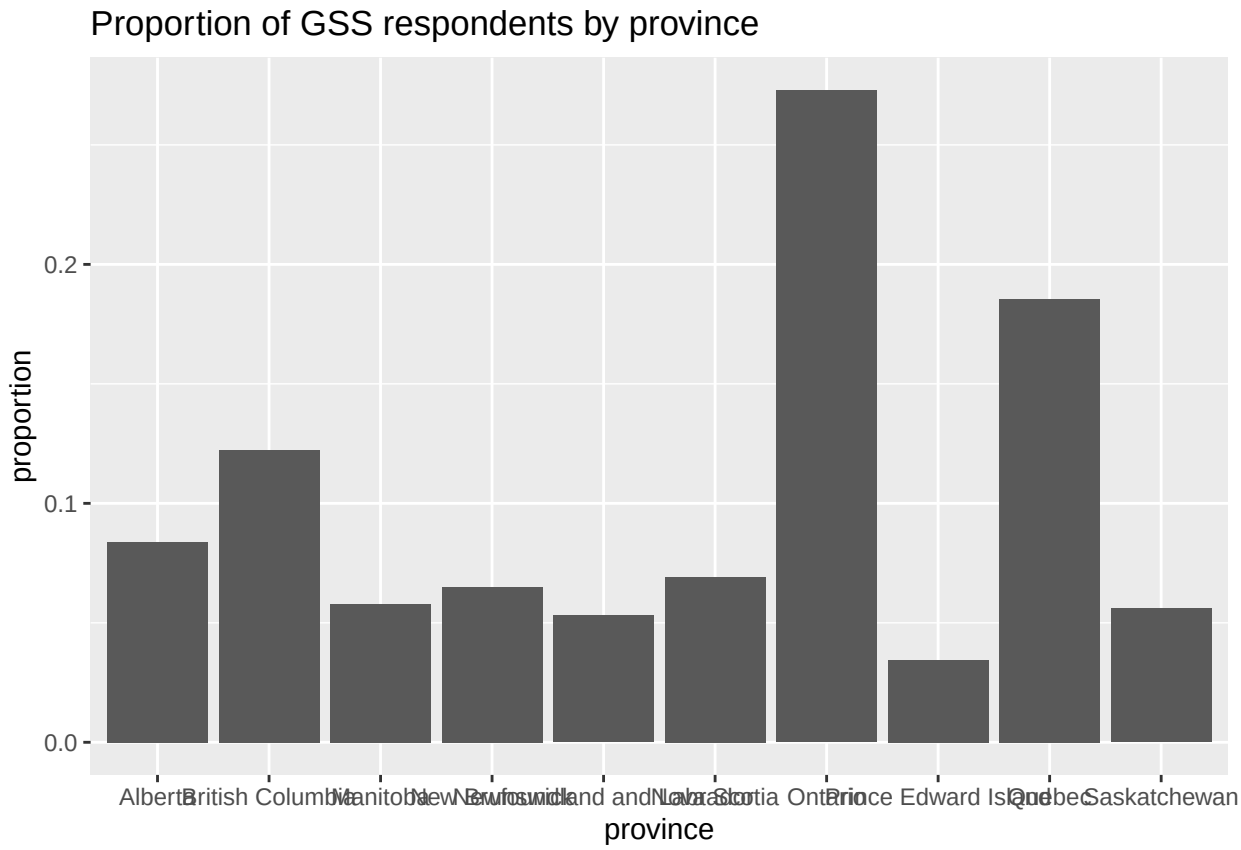
```
resp_by_prov <- gss %>%
  group_by(province) %>%
  tally() %>%
  mutate(prop = n / sum(n))

resp_by_prov
```

```
## # A tibble: 10 x 3
##    province                     n   prop
##    <chr>                    <int>  <dbl>
##  1 Alberta                   1728 0.0839
##  2 British Columbia          2522 0.122
##  3 Manitoba                  1192 0.0579
##  4 New Brunswick             1337 0.0649
##  5 Newfoundland and Labrador 1094 0.0531
##  6 Nova Scotia               1425 0.0692
##  7 Ontario                   5621 0.273
##  8 Prince Edward Island       708 0.0344
##  9 Quebec                    3822 0.186
## 10 Saskatchewan              1153 0.0560
```

Now plot

```
ggplot(data = resp_by_prov, aes(x = province, y = prop)) +
  geom_bar(stat = "identity") +
  ylab("proportion")+
  ggtitle("Proportion of GSS respondents by province")
```

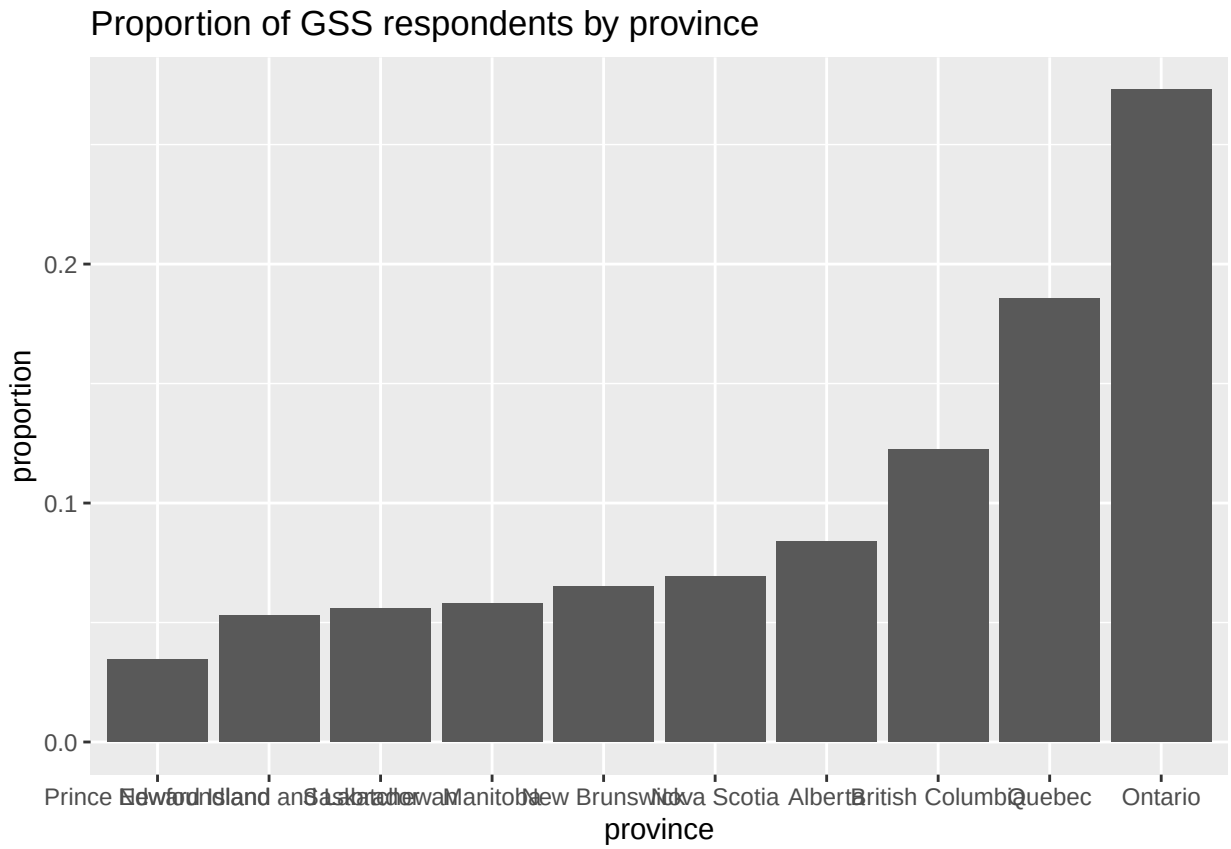


There are a few things here that would be nice to fix. Firstly, the categories are ordered alphabetically, which is the default. It would be better visually to order by proportion. We can do this using the `fct_reorder` function to alter (mutate) the province variable.

```
resp_by_prov <- resp_by_prov %>%
  mutate(province = fct_reorder(province, prop)) # order by proportion
```

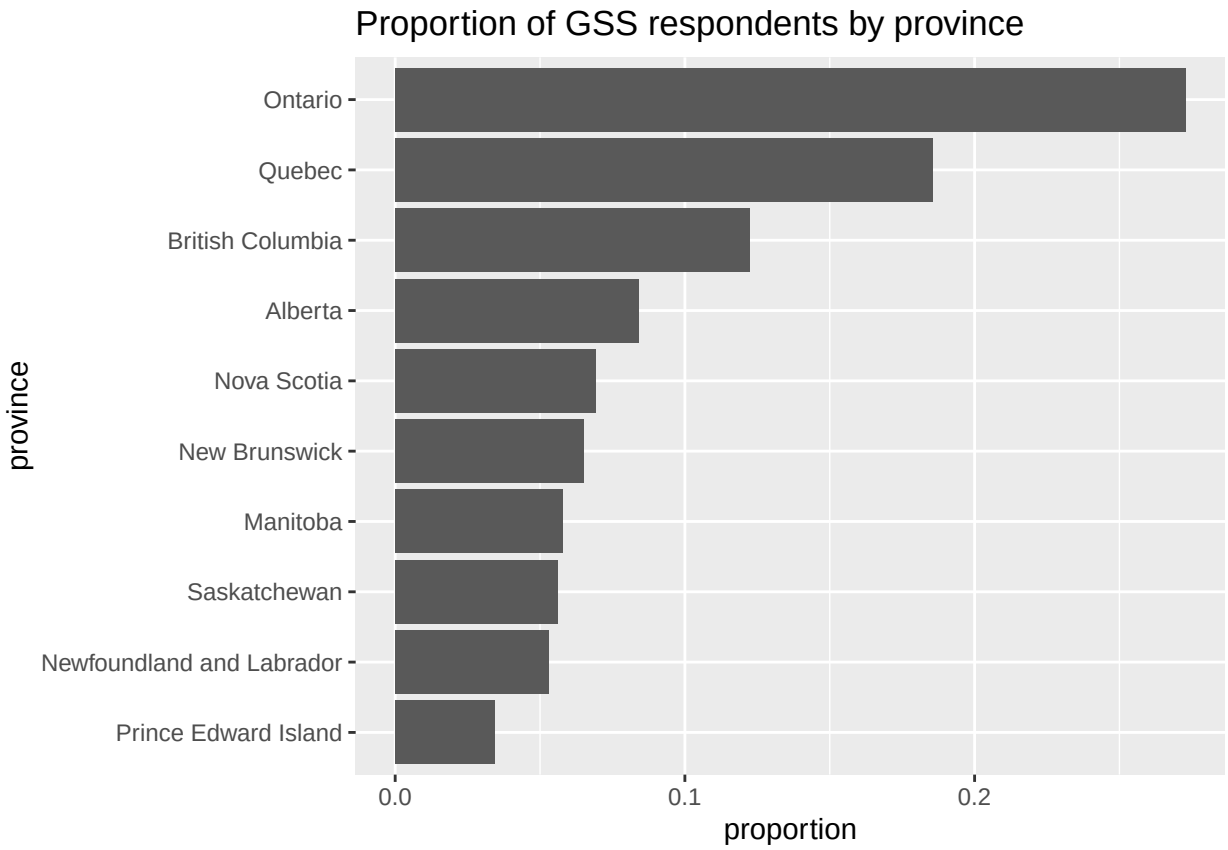
Now try plotting again.

```
ggplot(data = resp_by_prov, aes(x = province, y = prop)) +
  geom_bar(stat = "identity") +
  ylab("proportion")+
  ggtitle("Proportion of GSS respondents by province")
```



To improve readability, could change to horizontal bar chart.

```
ggplot(data = resp_by_prov, aes(x = province, y = prop)) +
  geom_bar(stat = "identity") +
  ylab("proportion")+
  ggtitle("Proportion of GSS respondents by province") +
  coord_flip()
```

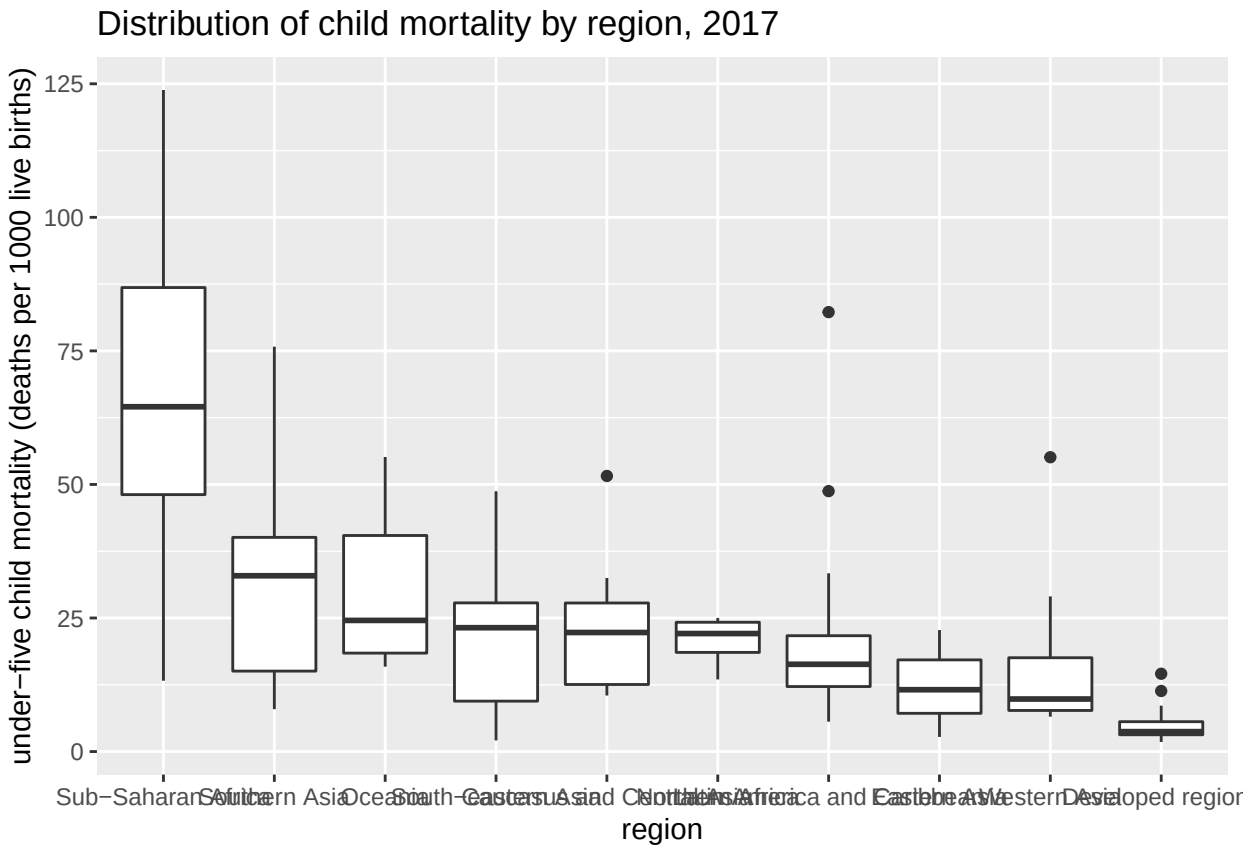


Box plots

Let's use the country indicators dataset here and do boxplots of child mortality in 2017 over regions. Like the bar chart example, best to reorder the regions by the variable we are interested in

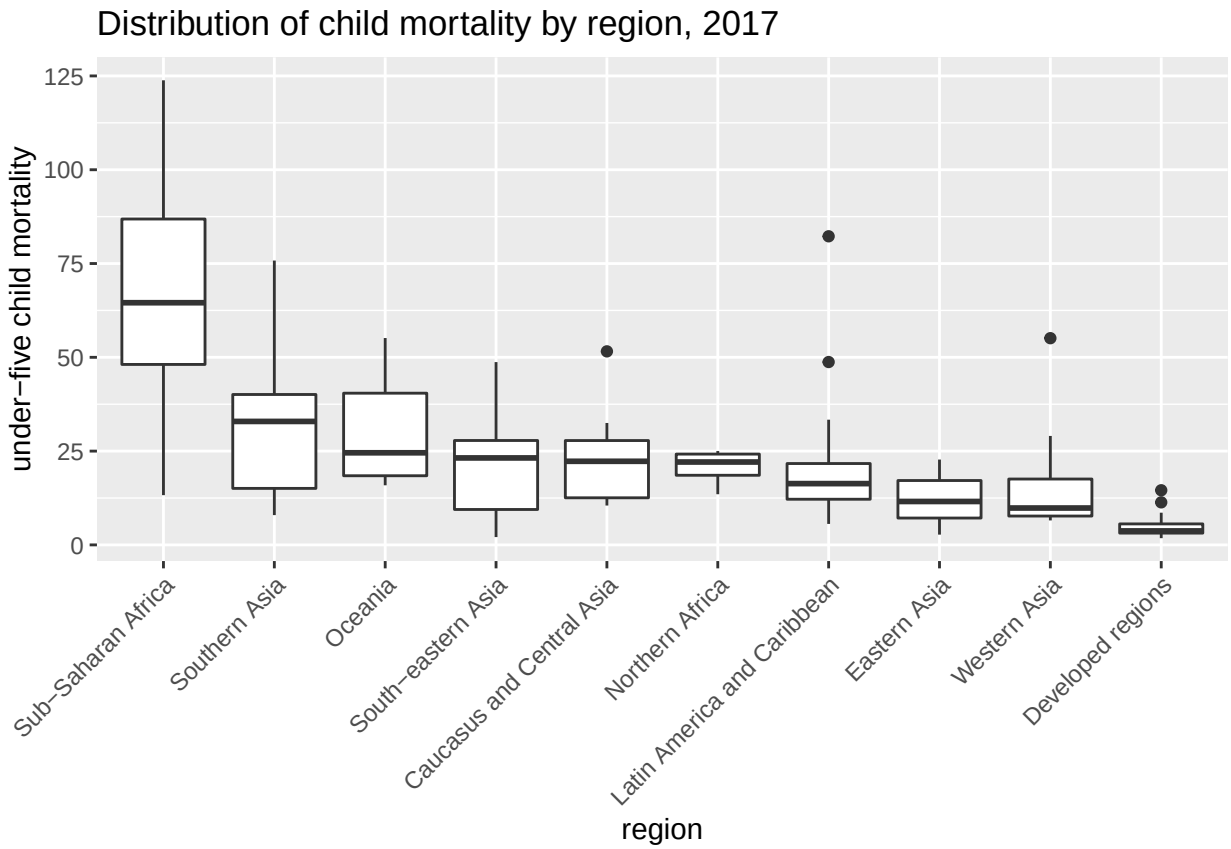
```
country_ind_2017 <- country_ind %>%
  filter(year==2017) %>%
  mutate(region = fct_reorder(region, -child_mort)) # descending order

ggplot(data = country_ind_2017, aes(x = region, y = child_mort)) +
  geom_boxplot() +
  ylab("under-five child mortality (deaths per 1000 live births)") +
  ggtitle("Distribution of child mortality by region, 2017")
```



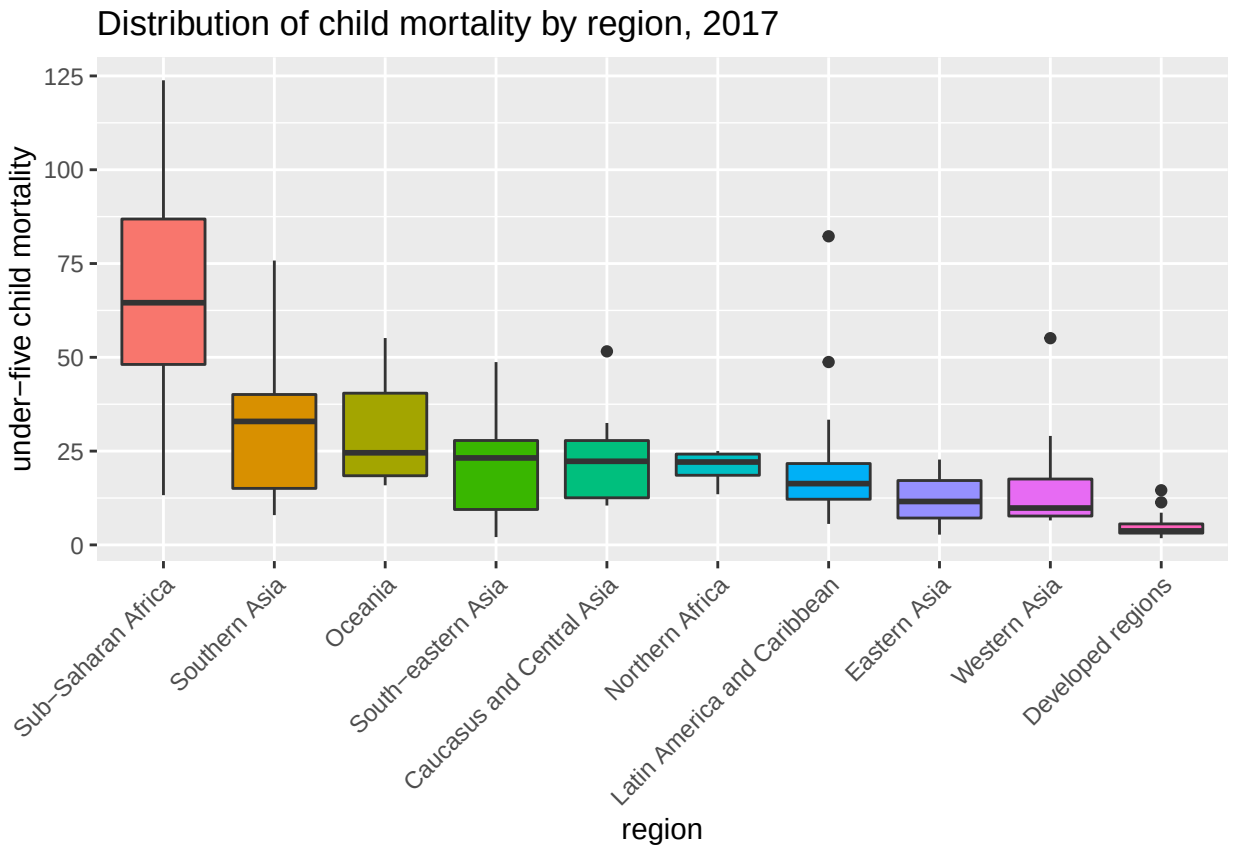
The labels on the x axis are hard to read. We could do the same as last time (switch to horizontal), or we can change the alignment of the labels:

```
ggplot(data = country_ind_2017, aes(x = region, y = child_mort)) +
  geom_boxplot() +
  ylab("under-five child mortality") +
  ggtitle("Distribution of child mortality by region, 2017") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Note if you want to color the boxes, use `fill`, and then remove the legend (not needed)

```
ggplot(data = country_ind_2017, aes(x = region, y = child_mort, fill = region)) +
  geom_boxplot() +
  ylab("under-five child mortality") +
  ggtitle("Distribution of child mortality by region, 2017") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1) ,
        legend.position = 'none')
```



Line graphs

Let's look at the mean age at marriage by age of respondent. Firstly, let's make a new variable in the `gss` dataset that groups people into 5-year age groups. Here's the code to do this:

```
age_groups <- seq(15, 80, by = 5)
gss$age_group <- as.numeric(as.character(cut(gss$age,
                  breaks= c(age_groups, Inf),
                  labels = age_groups,
                  right = FALSE)))

gss %>% select(age, age_group)
```

```
## # A tibble: 20,602 x 2
##      age age_group
##    <dbl>     <dbl>
##  1  52.7        50
##  2  51.1        50
##  3  63.6        60
##  4  80          80
##  5  28          25
##  6  63          60
##  7  58.8        55
##  8  80          80
##  9  63.8        60
## 10  25.2        25
## # ... with 20,592 more rows
```

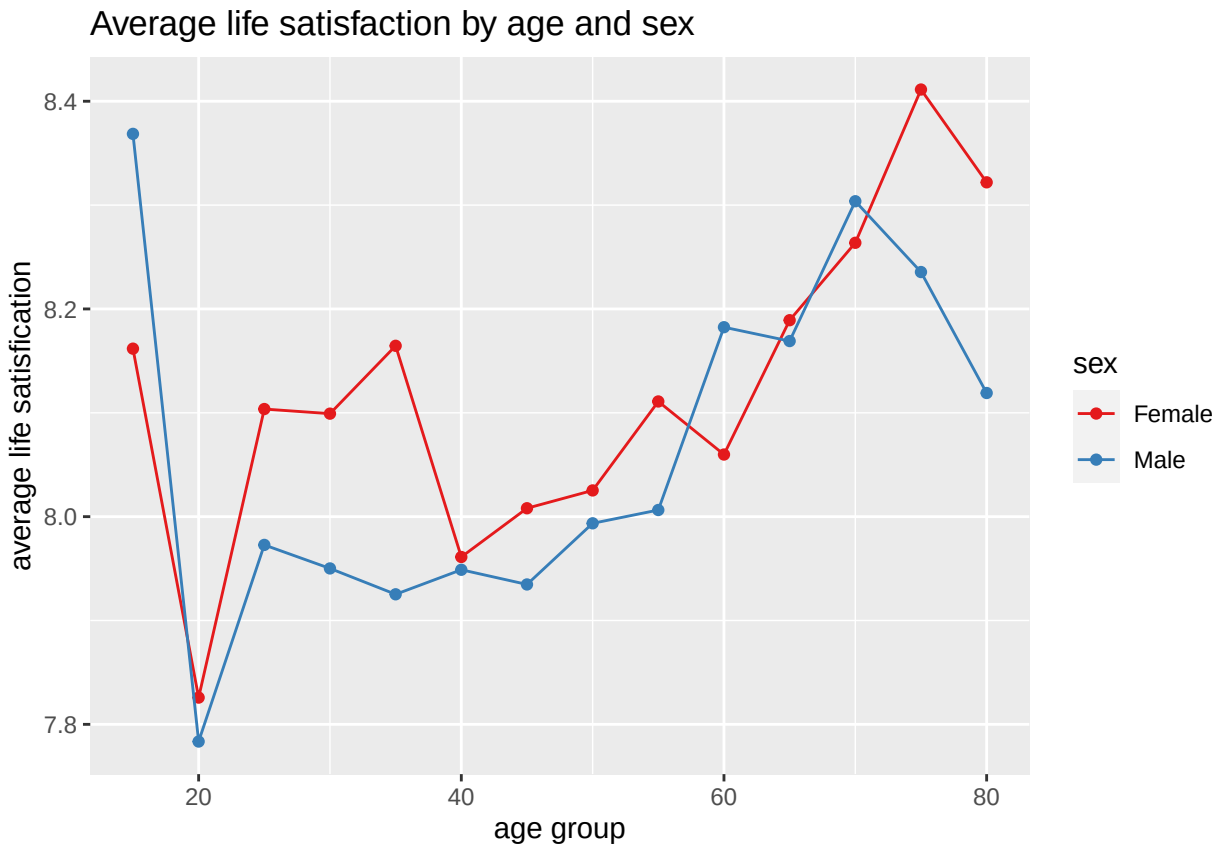
Now let's calculate the average of the 'life satisfaction' variable by age group and sex. This involves a `group_by` by two variables:

```
life_satis_age_sex <- gss %>%
  group_by(age_group, sex) %>%
  summarise(mean_life_satis = mean(feelings_life, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'age_group'. You can override using the `.groups` argument.
```

Plot as a line chart over age, coloring by sex, for this example we use a different colour palette called "Set1":

```
ggplot(data = life_satis_age_sex, aes(x = age_group, y = mean_life_satis, colour = sex)) +
  geom_point() +
  geom_line() +
  scale_color_brewer(palette = "Set1") + # change the color scheme
  ylab("average life satisfication") +
  xlab("age group") +
  ggtitle("Average life satisfaction by age and sex")
```

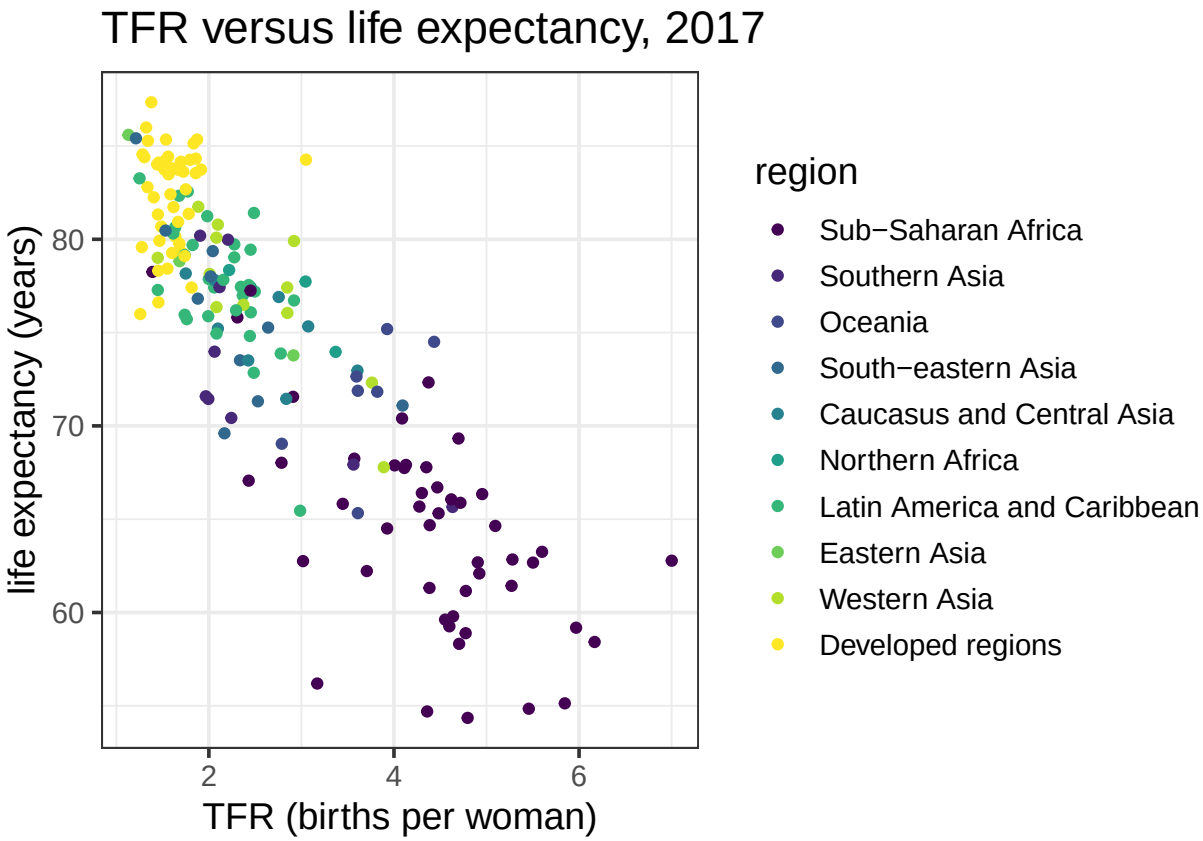


Scatter plots

Let's use the country indicators dataset here. The example in the lecture slides is life expectancy versus TFR. We also used a new colour palette called `virdis`, these colours palettes are designed to be viewable in black and white as well.

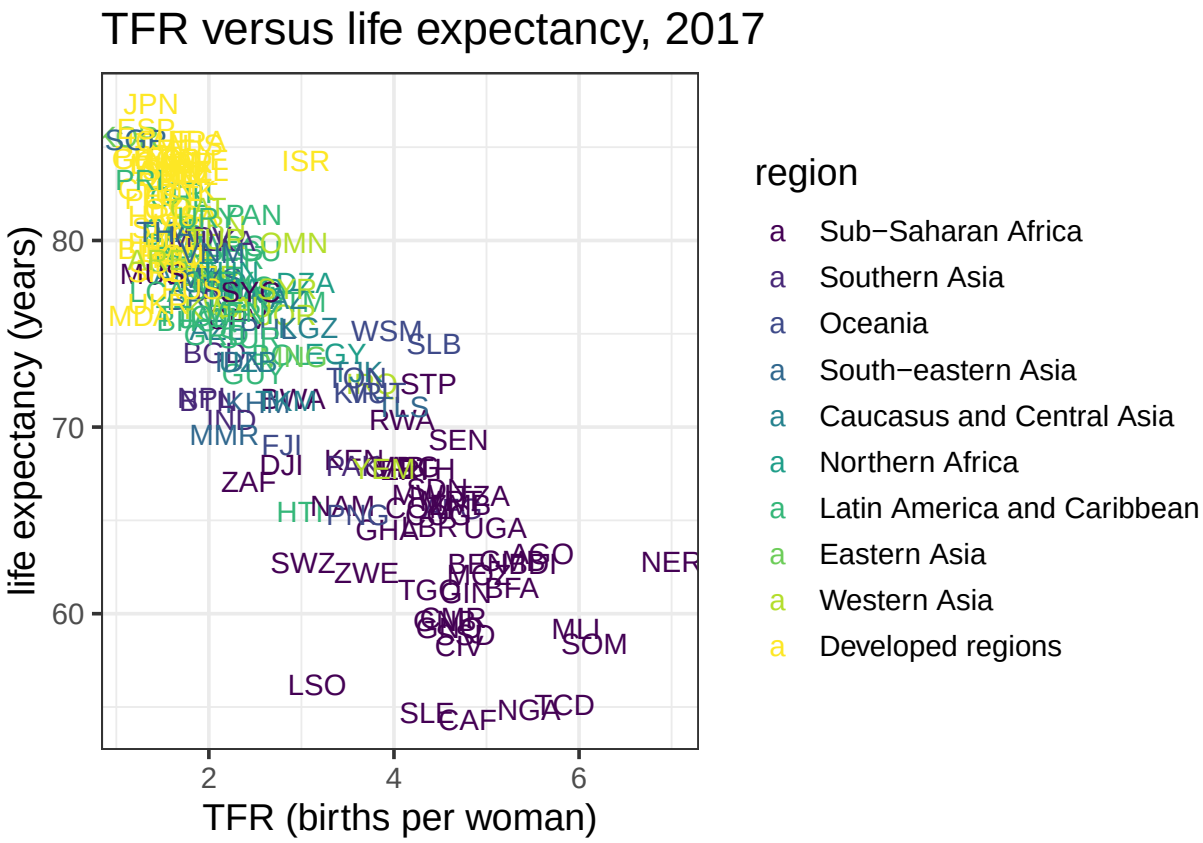
```
  ggplot(country_ind_2017, aes(tfr, life_expectancy, color = region,)) +
  geom_point() +
  ggtitle("TFR versus life expectancy, 2017")+
  theme_bw(base_size = 14) +
```

```
ylab("life expectancy (years)") +
xlab("TFR (births per woman)") +
scale_color_viridis_d()
```



Instead of dots could have country codes (although becomes hard to read, but easy to see outliers)

```
ggplot(country_ind_2017, aes(tfr, life_expectancy, color = region, label = country_code)) + # adding
geom_text() +
ggtitle("TFR versus life expectancy, 2017")+
theme_bw(base_size = 14)+
ylab("life expectancy (years)") +
xlab("TFR (births per woman)") +
scale_color_viridis_d()
```



Faceting

Changing the color and fills is useful to show one other variable on a graph. For more complicated set-ups, faceting graphs by an additional variable becomes useful.

For example let's go back to plotting a histogram of age at first marriage by sex, but also add in whether or not the respondent was born in Canada. First, look at the unique values of the `place_birth_canada` variable:

```
gss %>%
  select(place_birth_canada) %>%
  unique()
```

```
## # A tibble: 4 x 1
##   place_birth_canada
##   <chr>
## 1 Born in Canada
## 2 Born outside Canada
## 3 <NA>
## 4 Don't know
```

For now, filter the data to only include the first two categories. To do this, use the `%in%` function within filter:

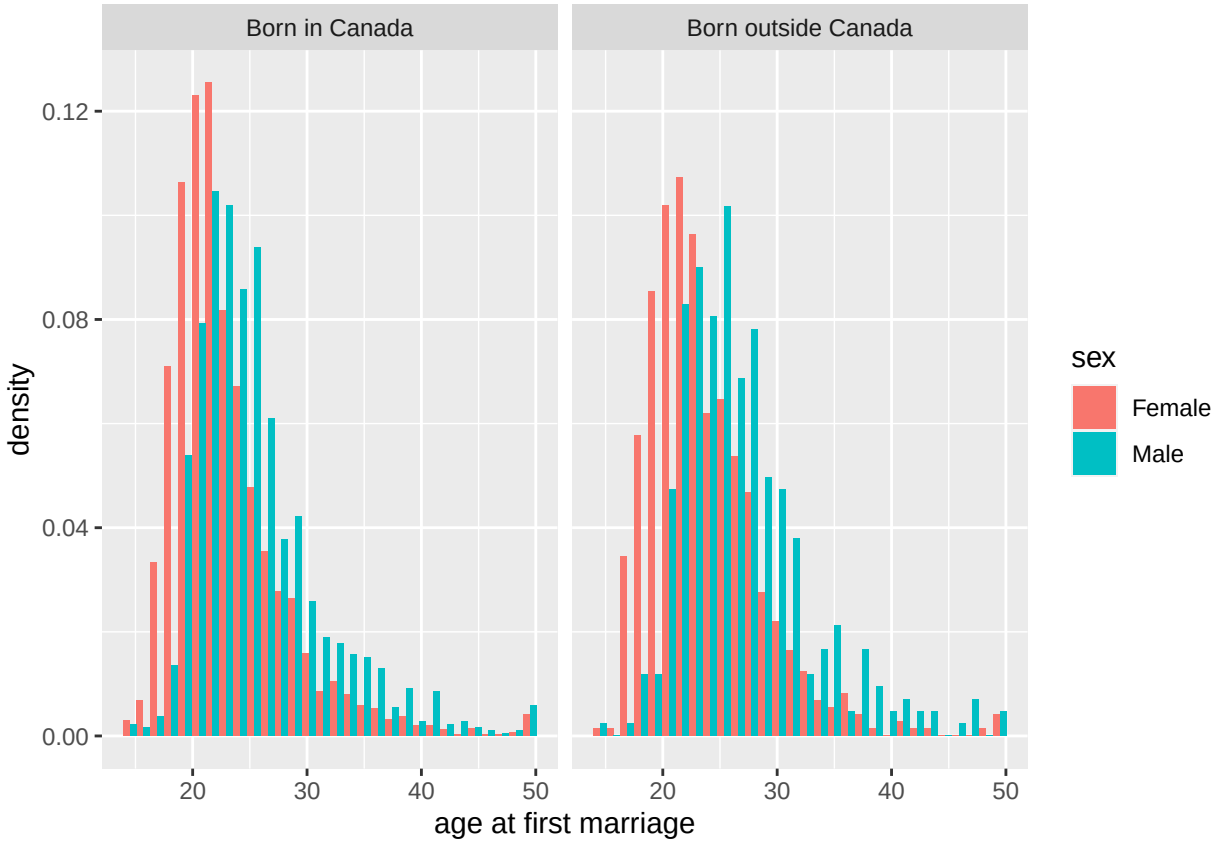
```
gss_subset <- gss %>%
  filter(place_birth_canada %in% c("Born in Canada", "Born outside Canada"))
```

Now plot the histograms as before, but now also facet by place of birth. Note we are plotting the density here.

```
ggplot(data = gss_subset, aes(age_at_first_marriage, fill = sex)) +
  geom_histogram(position = 'dodge', aes(y = ..density..)) +
  facet_wrap(~place_birth_canada) +
  xlab("age at first marriage")
```

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

```
## Warning: Removed 15137 rows containing non-finite values (stat_bin).
```



Review Questions

1. Using the country_indicator dataset, create a scatter plot of GDP over life expectancy by region for the year 2014. Edit the labels, set a title, and make sure the graph is color-coded.
2. Using the GSS dataset, create a bar graph of non-missing values for the province of birth (‘place_birth_province) and then arrange the proportions from high to low. Make sure to color code and make all labels are readable.