Week 2: EDA

Monica Alexander

19/01/2021

By the end of this lab you should know

- how to get the dimensions of a data frame (rows, columns)
- 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. + We are using two additional packages today. You will see the use of `skimr` in a moment, but the `here` package allows us to access our files easier. Since we haven't use these packages before we need to install them. 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.

```r
#install.packages("skimr") # Install new packages
#install.packages("here")

# Call the packages that you are using
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------------------------------------------
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
library(skimr)
library(here)
```

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

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

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   caseid = col_double(),
##   age = col_double(),
##   age_first_child = col_double(),
##   age_youngest_child_under_6 = col_double(),
##   total_children = col_double(),
##   age_start_relationship = col_double(),
##   age_at_first_marriage = col_double(),
##   age_at_first_birth = col_double(),
##   distance_between_houses = col_double(),
##   age_youngest_child_returned_work = col_double(),
##   feelings_life = col_double(),
##   hh_size = col_double(),
##   number_total_children_intention = col_double(),
##   number_marriages = col_double(),
##   fin_supp_child_supp = col_double(),
##   fin_supp_child_exp = col_double(),
##   fin_supp_lump = col_double(),
##   fin_supp_other = col_double(),
##   is_male = col_double(),
##   main_activity = col_logical()
##   # ... with 2 more columns
## )
```

```
## See spec(...) for full column specifications.
```

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

```
## Parsed with column specification:
## cols(
##   country_code = col_character(),
##   country = col_character(),
##   region = col_character(),
##   year = col_double(),
##   tfr = col_double(),
##   life_expectancy = col_double(),
##   child_mort = col_double(),
##   maternal_mort = col_double(),
##   gdp = col_double()
## )
```

Overview summaries of data

Skim is useful for the GSS, because it gives a broad overview of what types of variables the dataset contains.

```r
summary(country_ind)
```

```
##  country_code         country             region               year     
##  Length:1584        Length:1584        Length:1584        Min.   :2009  
##  Class :character   Class :character   Class :character   1st Qu.:2011  
##  Mode  :character   Mode  :character   Mode  :character   Median :2013  
##                                                           Mean   :2013  
##                                                           3rd Qu.:2015  
##                                                           Max.   :2017  
##       tfr        life_expectancy   child_mort      maternal_mort   
##  Min.   :1.131   Min.   :47.02   Min.   :  1.796   Min.   :   2.0  
##  1st Qu.:1.748   1st Qu.:66.92   1st Qu.:  7.955   1st Qu.:  15.0  
##  Median :2.402   Median :75.78   Median : 19.698   Median :  55.0  
##  Mean   :2.886   Mean   :73.25   Mean   : 35.162   Mean   : 179.7  
##  3rd Qu.:3.984   3rd Qu.:79.70   3rd Qu.: 55.609   3rd Qu.: 248.0  
##  Max.   :7.511   Max.   :87.34   Max.   :166.192   Max.   :1450.0  
##       gdp          
##  Min.   :   670.8  
##  1st Qu.:  3591.3  
##  Median : 10869.4  
##  Mean   : 17371.6  
##  3rd Qu.: 24321.4  
##  Max.   :124024.6  
```

```r
skim(gss)
```

Table 1: Data summary

Name	gss
Number of rows	20602
Number of columns	85
Column type frequency:	
character	63
logical	1
numeric	21
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sex	0	1.00	4	6	0	2	0
place_birth_canada	97	1.00	10	19	0	3	0
place_birth_father	203	0.99	10	19	0	3	0
place_birth_mother	47	1.00	10	19	0	3	0
place_birth_macro_region	16457	0.20	4	18	0	6	0
place_birth_province	4289	0.79	6	39	0	11	0
year_arrived_canada	16550	0.20	19	27	0	14	0
province	0	1.00	6	25	0	10	0
region	0	1.00	6	16	0	5	0
pop_center	0	1.00	20	53	0	3	0
marital_status	7	1.00	7	21	0	6	0
aboriginal	3855	0.81	2	10	0	3	0
vis_minority	140	0.99	10	22	0	3	0
age_immigration	17225	0.16	12	14	0	16	0
landed_immigrant	16450	0.20	2	10	0	3	0
citizenship_status	1143	0.94	8	17	0	3	0
education	341	0.98	28	60	0	7	0
own_rent	120	0.99	10	59	0	3	0
living_arrangement	0	1.00	5	51	0	12	0
hh_type	76	1.00	5	40	0	5	0
partner_birth_country	7697	0.63	6	22	0	3	0
partner_birth_province	7883	0.62	6	39	0	12	0
partner_vis_minority	7719	0.63	10	22	0	3	0
partner_sex	20407	0.01	4	10	0	3	0
partner_education	8259	0.60	28	60	0	7	0
average_hours_worked	7166	0.65	6	19	0	6	0
worked_last_week	23	1.00	2	10	0	3	0
partner_main_activity	7907	0.62	5	51	0	10	0
self_rated_health	99	1.00	4	10	0	6	0
self_rated_mental_health	106	0.99	4	10	0	6	0
religion_has_affiliation	282	0.99	10	25	0	3	0
religion_importance	253	0.99	10	20	0	5	0
language_home	448	0.98	6	41	0	8	0
language_knowledge	105	0.99	10	26	0	5	0
income_family	0	1.00	17	21	0	6	0
income_respondent	0	1.00	17	21	0	6	0
occupation	7297	0.65	9	59	0	11	0
childcare_regular	18756	0.09	10	35	0	3	0
childcare_type	19365	0.06	14	38	0	6	0
childcare_monthly_cost	19962	0.03	2	18	0	7	0
ever_fathered_child	13604	0.34	2	10	0	3	0
ever_given_birth	12769	0.38	2	10	0	3	0
number_of_current_union	18600	0.10	11	21	0	4	0
lives_with_partner	0	1.00	2	3	0	2	0
children_in_household	0	1.00	8	22	0	4	0
has_grandchildren	4	1.00	2	3	0	2	0
grandparents_still_living	9	1.00	2	10	0	3	0
ever_married	5	1.00	2	10	0	3	0
current_marriage_is_first	10416	0.49	2	10	0	3	0
religion_participation	199	0.99	10	23	0	6	0
partner_location_residence	18978	0.08	10	36	0	4	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
full_part_time_work	18852	0.08	9	25	0	3	0
time_off_work_birth	18855	0.08	2	10	0	3	0
reason_no_time_off_birth	20283	0.02	5	48	0	10	0
returned_same_job	19451	0.06	2	3	0	2	0
satisfied_time_children	19691	0.04	9	17	0	5	0
provide_or_receive_fin_supp	19578	0.05	10	35	0	5	0
fin_supp_agreement	19937	0.03	5	60	0	5	0
future_children_intention	13438	0.35	6	18	0	6	0
age_diff	10430	0.49	10	42	0	16	0
educ_cat	341	0.98	8	26	0	6	0
partner_educ_cat	8259	0.60	8	26	0	6	0
has_bachelor_or_higher	341	0.98	2	3	0	2	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
main_activity	20602	0	NaN	:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	
caseid	0	1.00	10301.50	5947.43	1.0	5151.25	10301.5	1545
age	0	1.00	52.19	17.75	15.0	37.30	54.2	6
age_first_child	6835	0.67	30.57	17.10	0.0	15.00	32.0	4
age_youngest_child_under_6	18488	0.10	2.41	1.60	0.0	1.00	2.0	4
total_children	19	1.00	1.68	1.49	0.0	0.00	2.0	3
age_start_relationship	18566	0.10	33.63	11.20	18.0	25.00	30.5	4
age_at_first_marriage	15248	0.26	24.10	5.41	15.0	20.50	22.8	2
age_at_first_birth	7865	0.62	26.86	5.42	18.0	22.80	26.4	3
distance_between_houses	19476	0.05	17.13	18.18	0.0	4.00	10.0	2
age_youngest_child_returned_work	19466	0.06	6.59	6.17	0.2	0.50	6.0	1
feelings_life	271	0.99	8.09	1.65	0.0	7.00	8.0	9
hh_size	0	1.00	2.35	1.26	1.0	1.00	2.0	3
number_total_children_intention	12202	0.41	0.90	1.18	0.0	0.00	0.0	2
number_marriages	0	1.00	0.80	0.62	0.0	0.00	1.0	1
fin_supp_child_supp	20057	0.03	0.77	0.42	0.0	1.00	1.0	1
fin_supp_child_exp	20057	0.03	0.34	0.47	0.0	0.00	0.0	1
fin_supp_lump	20057	0.03	0.06	0.23	0.0	0.00	0.0	0
fin_supp_other	20057	0.03	0.06	0.23	0.0	0.00	0.0	0
is_male	0	1.00	0.46	0.50	0.0	0.00	0.0	1
number_total_children_known	0	1.00	0.41	0.49	0.0	0.00	0.0	1
age_group	0	1.00	49.88	17.97	15.0	35.00	50.0	6

Calculating the number of rows, columns

Use `nrow()`, `ncol()` and `dim()`

```r
nrow(gss)
```

```
## [1] 20602
```

```r
ncol(gss)
```

```
## [1] 85
```

```r
dim(gss)
```

```
## [1] 20602    85
```

```r
# or can pipe

gss %>%
  filter(place_birth_canada == "Born in Canada") %>%
  nrow()
```

```
## [1] 16355
```

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

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

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # 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)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # 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)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # 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.

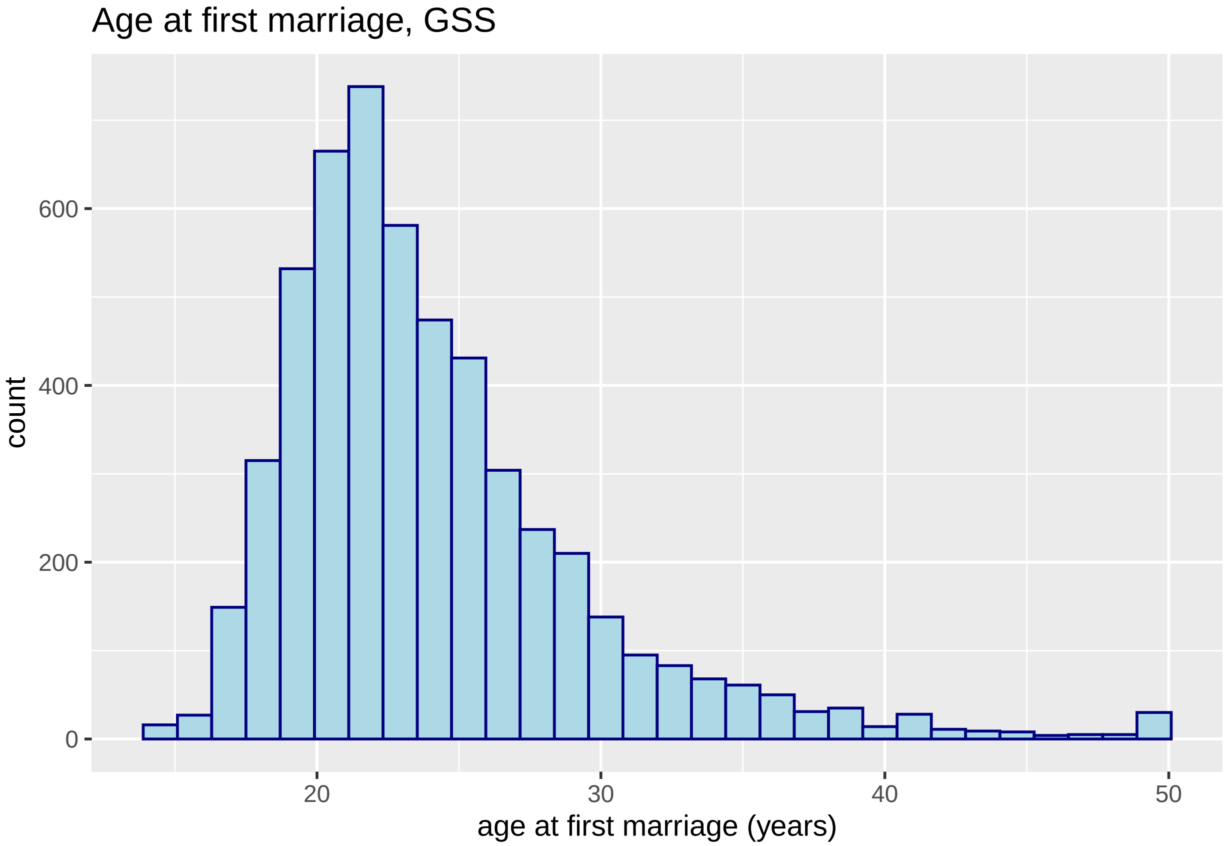
Histograms

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).
```



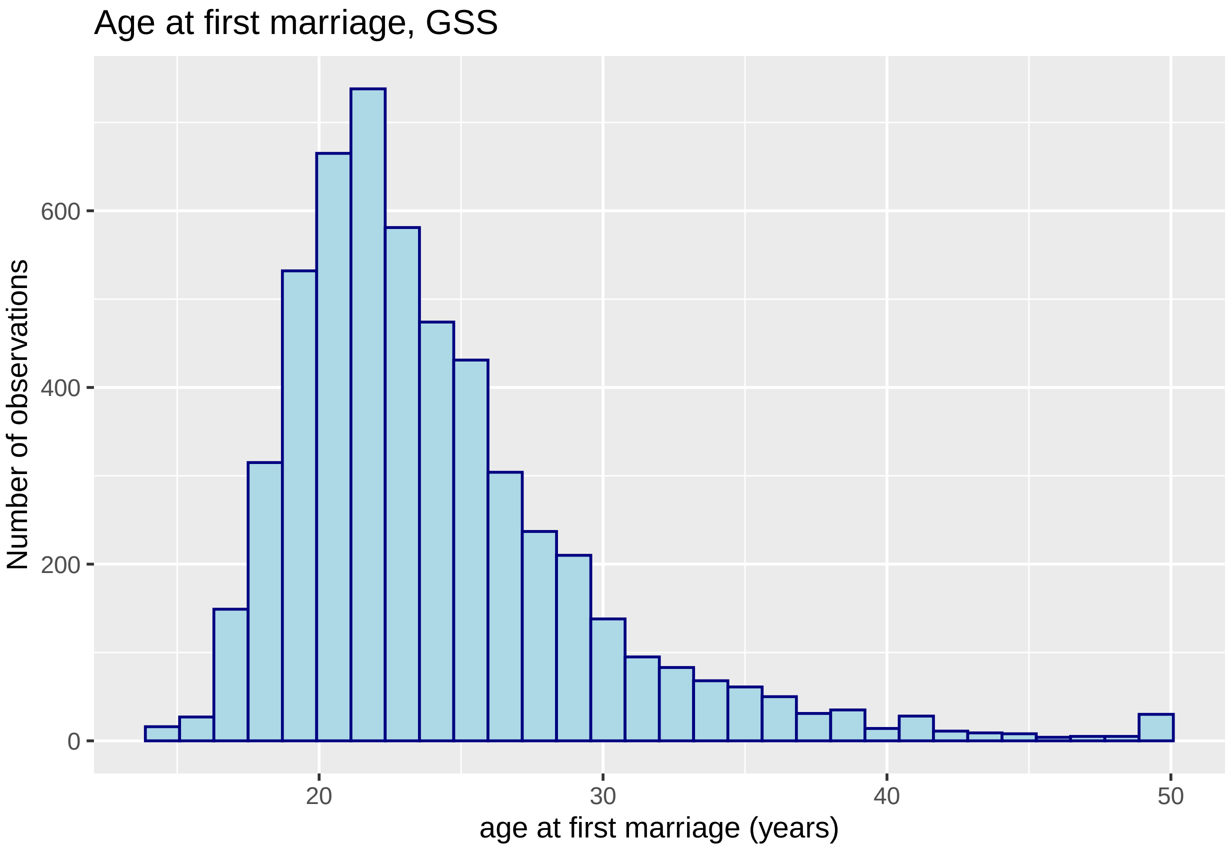
Note that you can also save the plot as an object and then print it

```
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`.
```

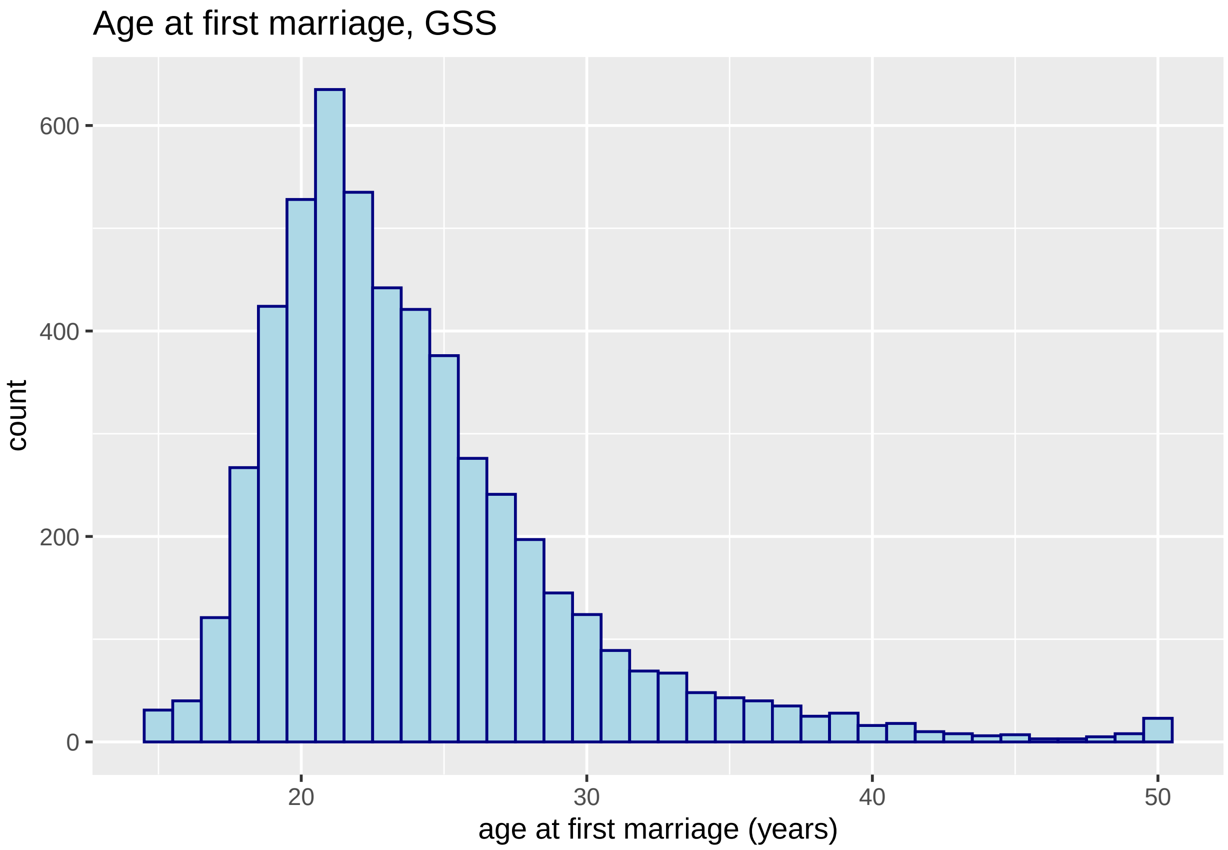
```
## Warning: Removed 15248 rows containing non-finite values (stat_bin).
```



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.

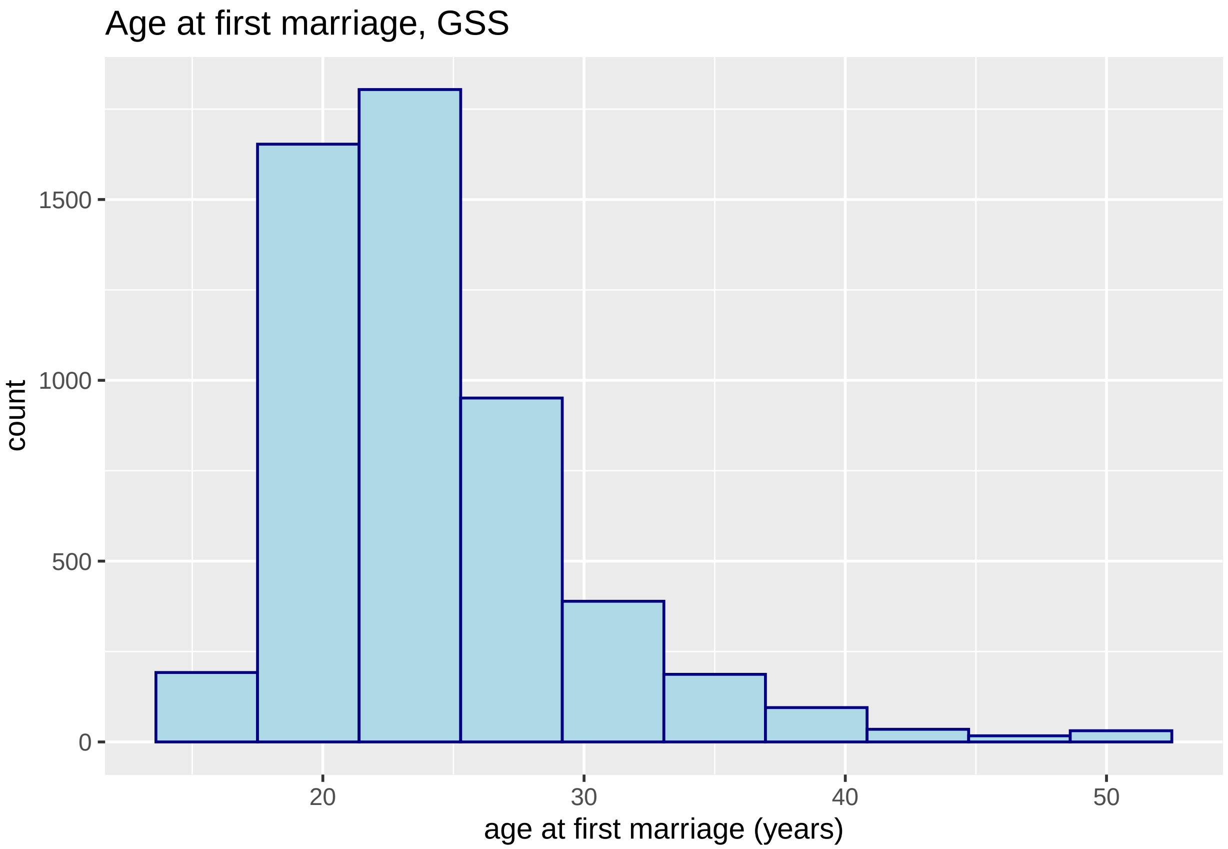
```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy", binwidth = 1) +
  ggtitle("Age at first marriage, GSS") +
  xlab("age at first marriage (years)")
```

```
## 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).
```

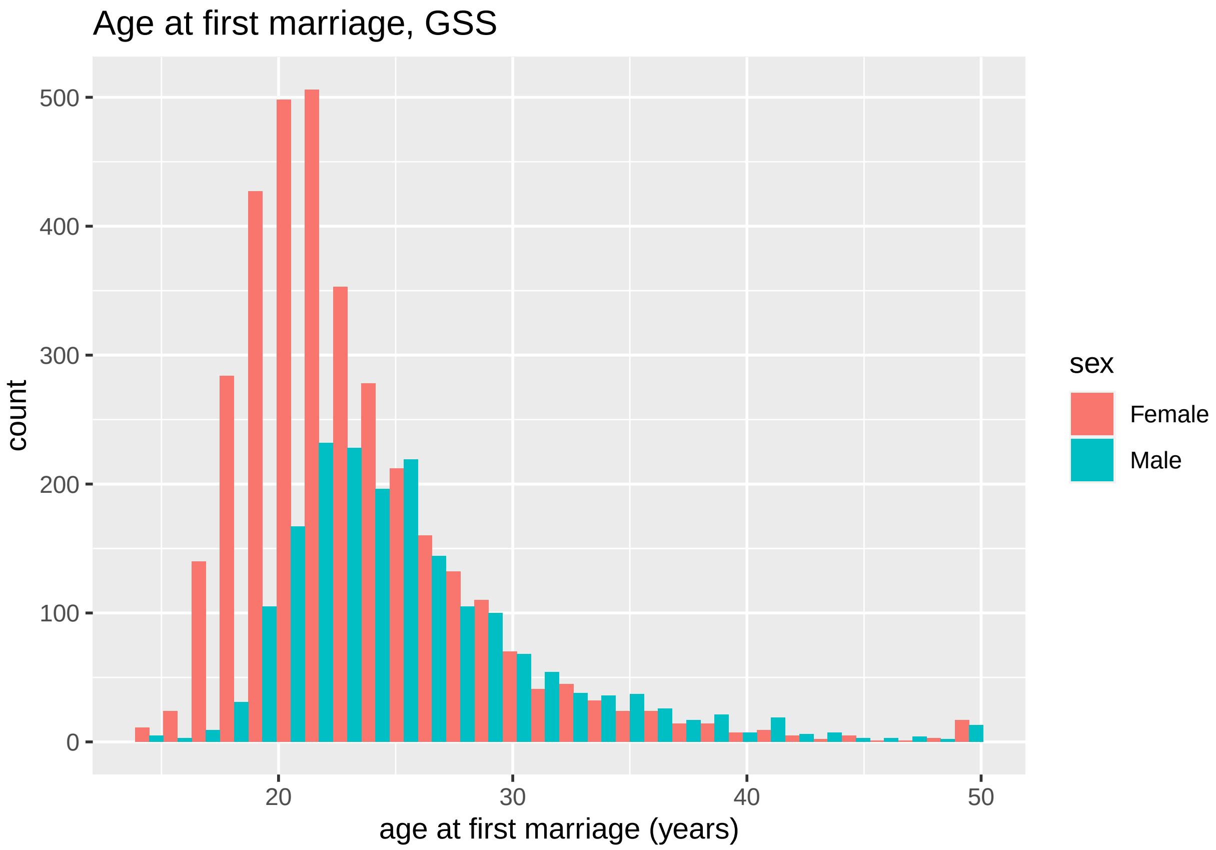


We can also plot by another variable to compare the plots by the categories of the variable. For example, we look at plots by sex:

```
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

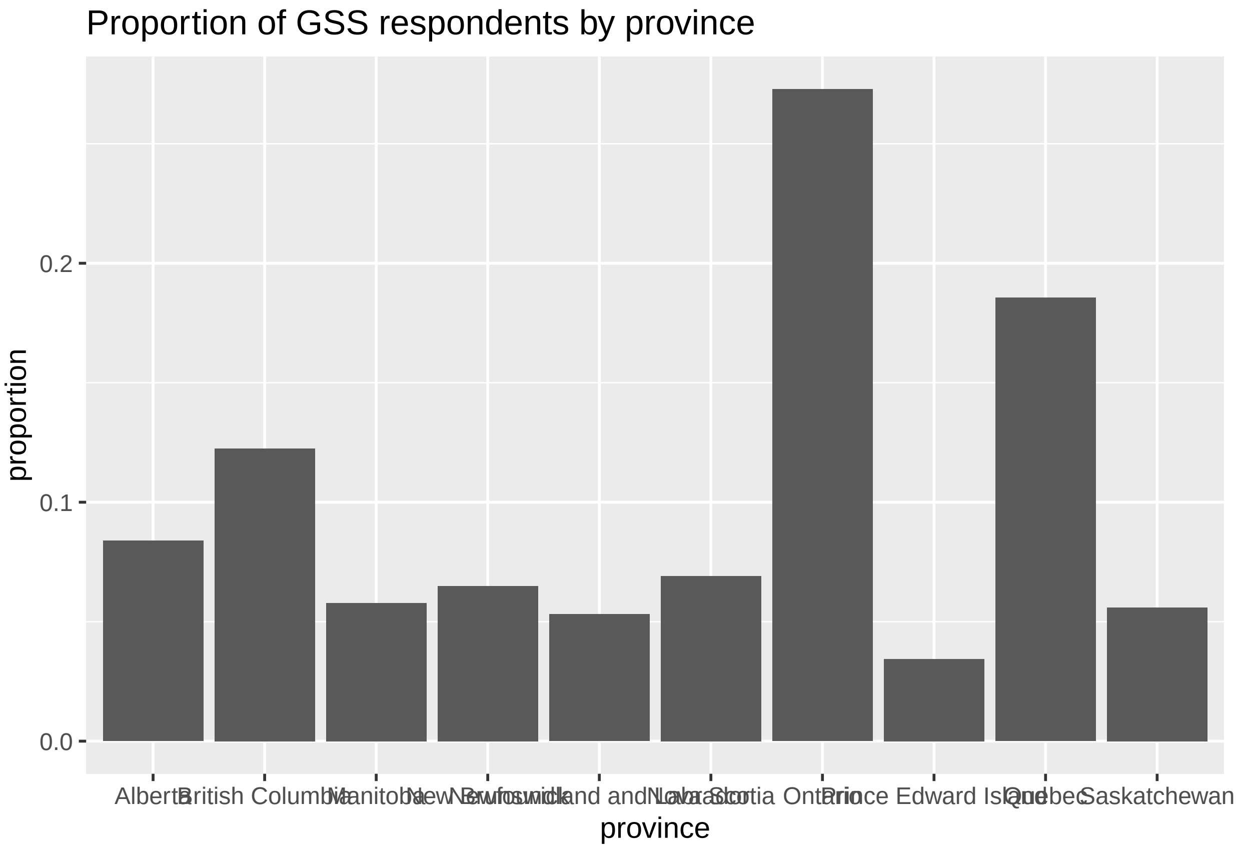
```
resp_by_prov <- gss %>%
  group_by(province) %>%
  tally() %>%
  mutate(prop = n / sum(n))

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")
```

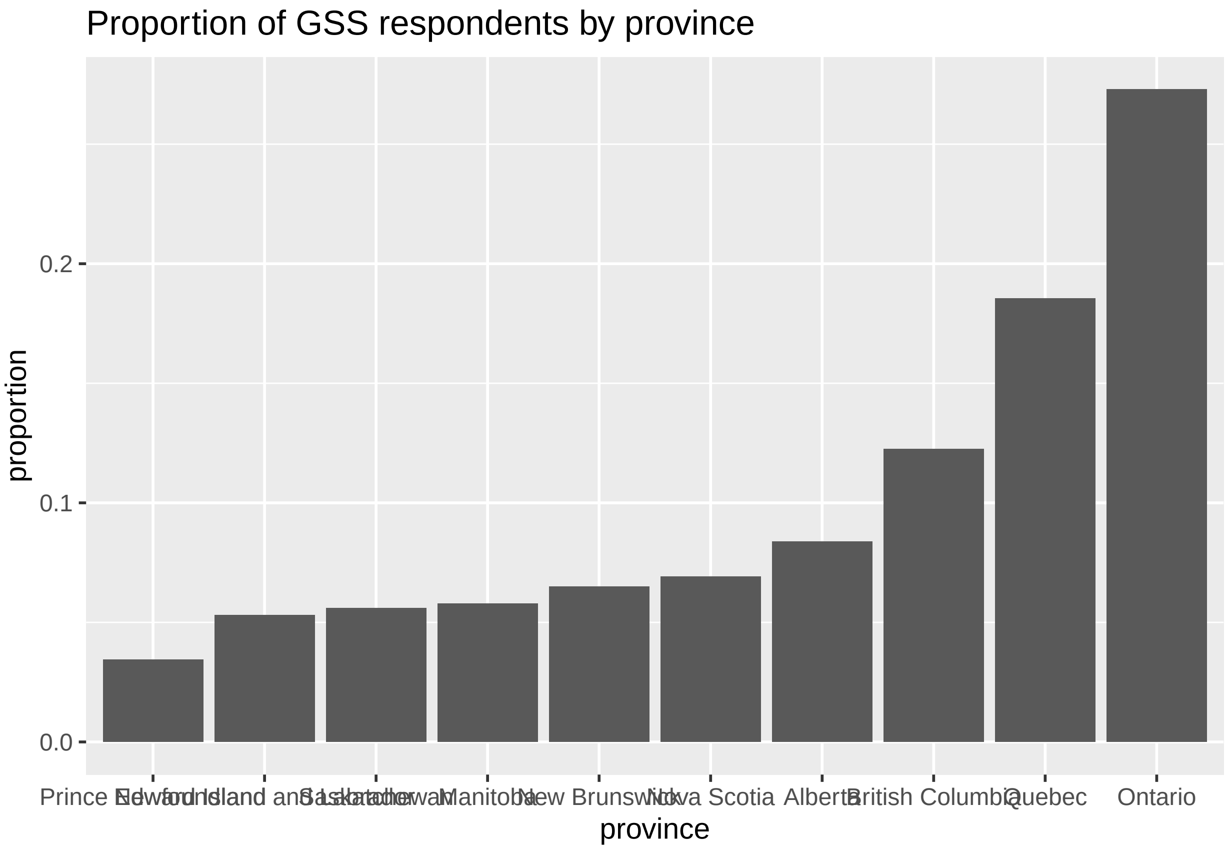


There are a few things here that would be nice to fix. Firstly, the categories are ordered alphabetically, which is the default. It would be better visually to order by proportion. We can do this using the `fct_reorder` function to alter (mutate) the province variable.

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

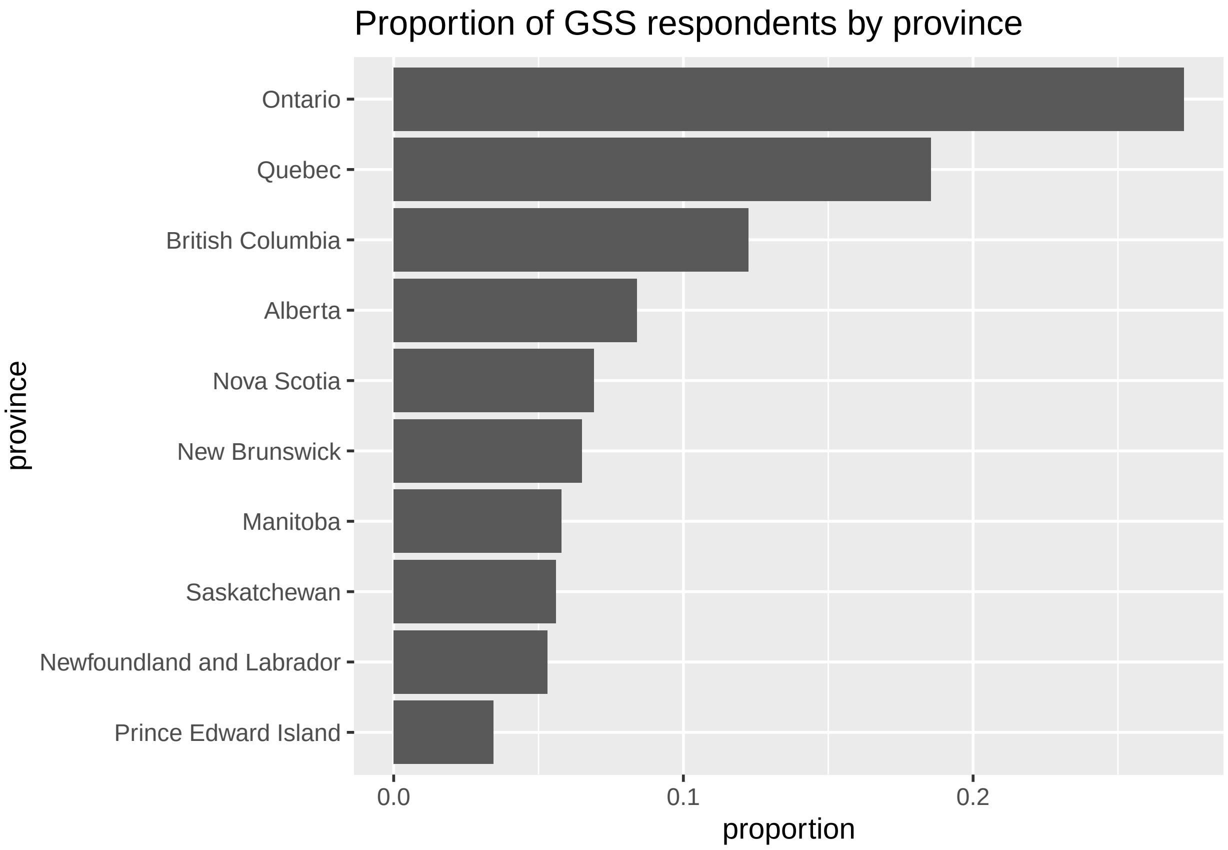
Now try plotting again.

```
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.

```r
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()
```

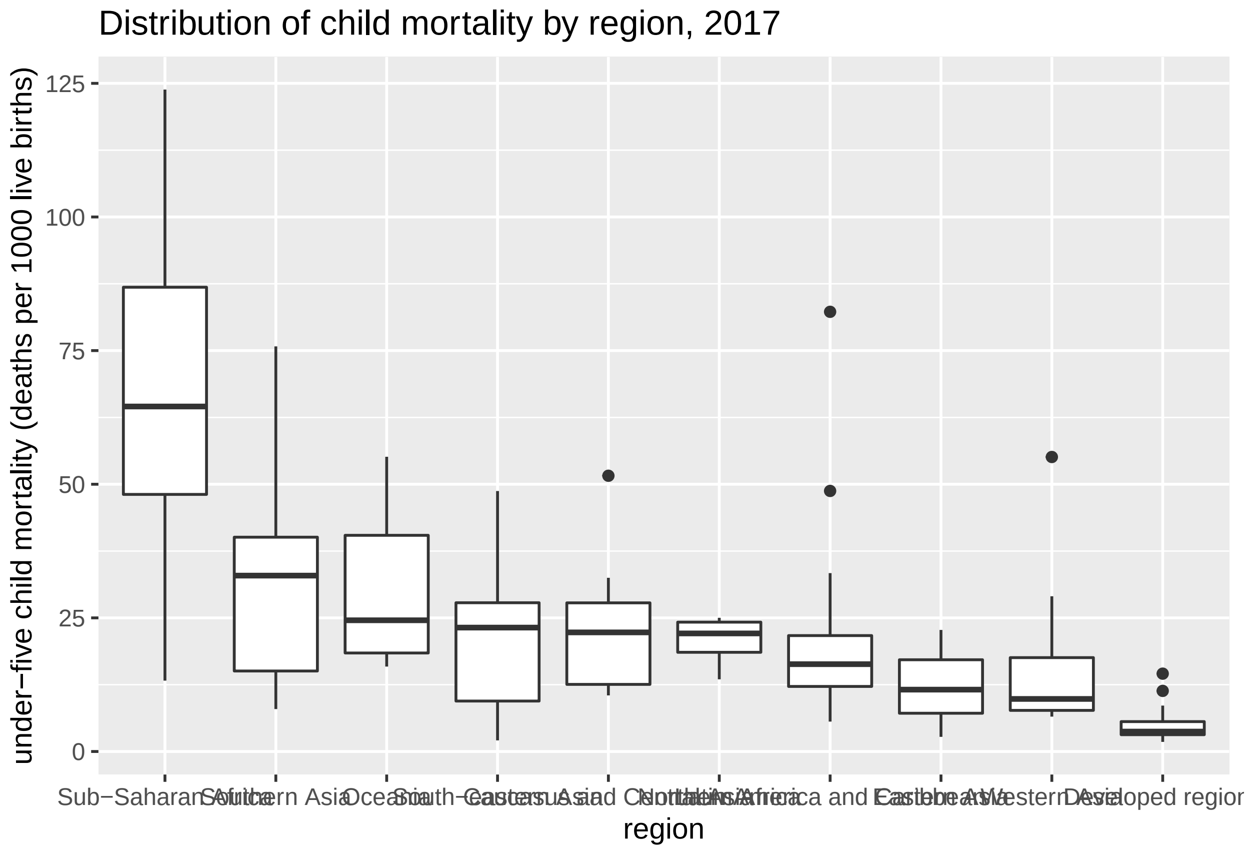


Box plots

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

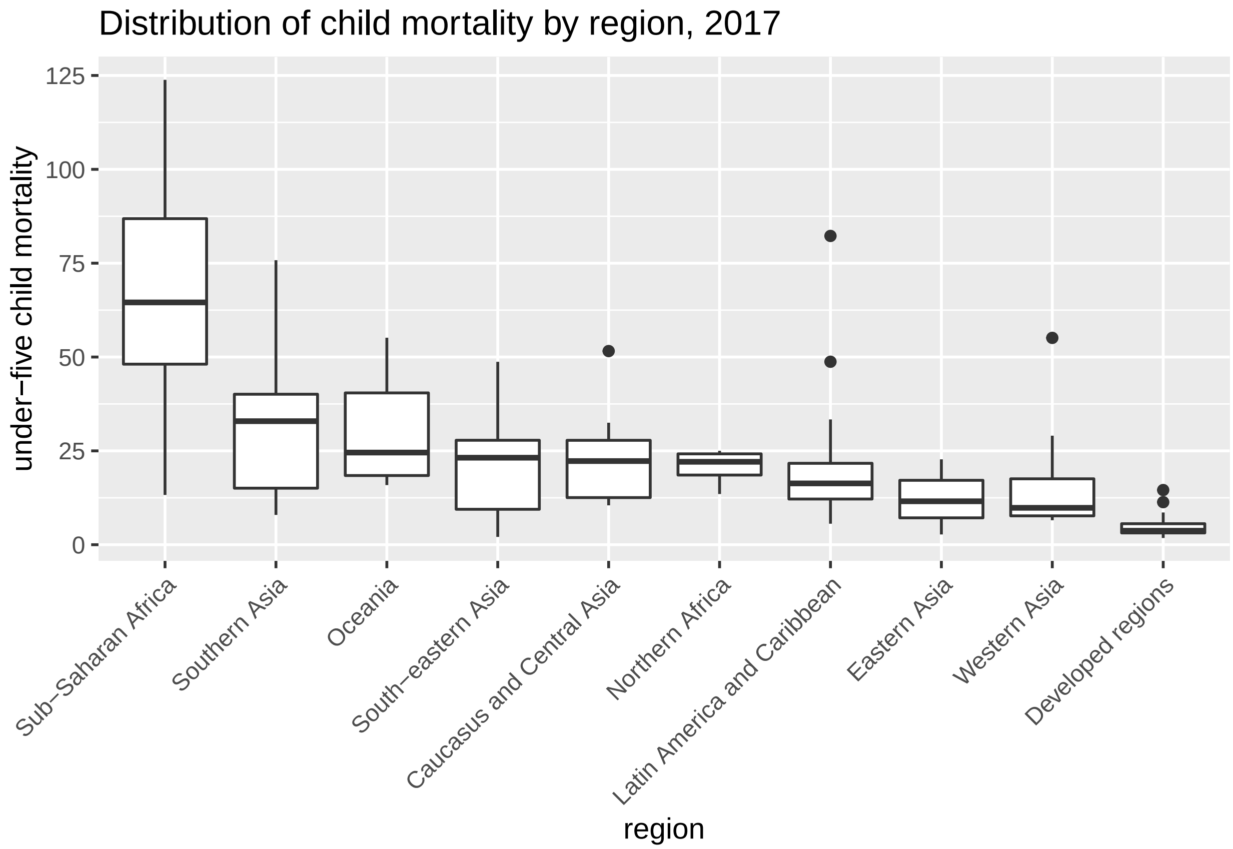
```
country_ind_2017 <- country_ind %>%
  filter(year==2017) %>%
  mutate(region = fct_reorder(region, -child_mort)) # descending order

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")
```



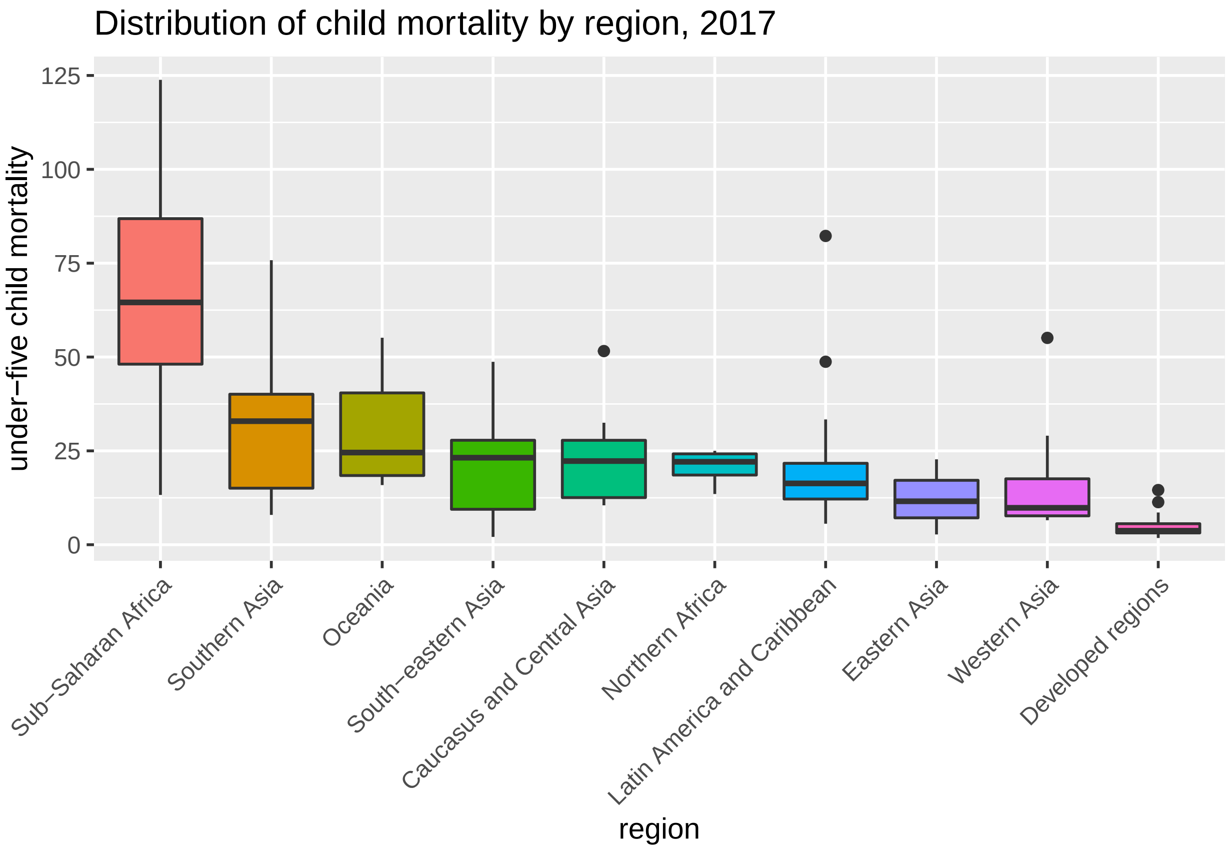
The labels on the x axis are hard to read. We could do the same as last time (switch to horizontal), or we can change the alignment of the labels:

```
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)

```r
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
```

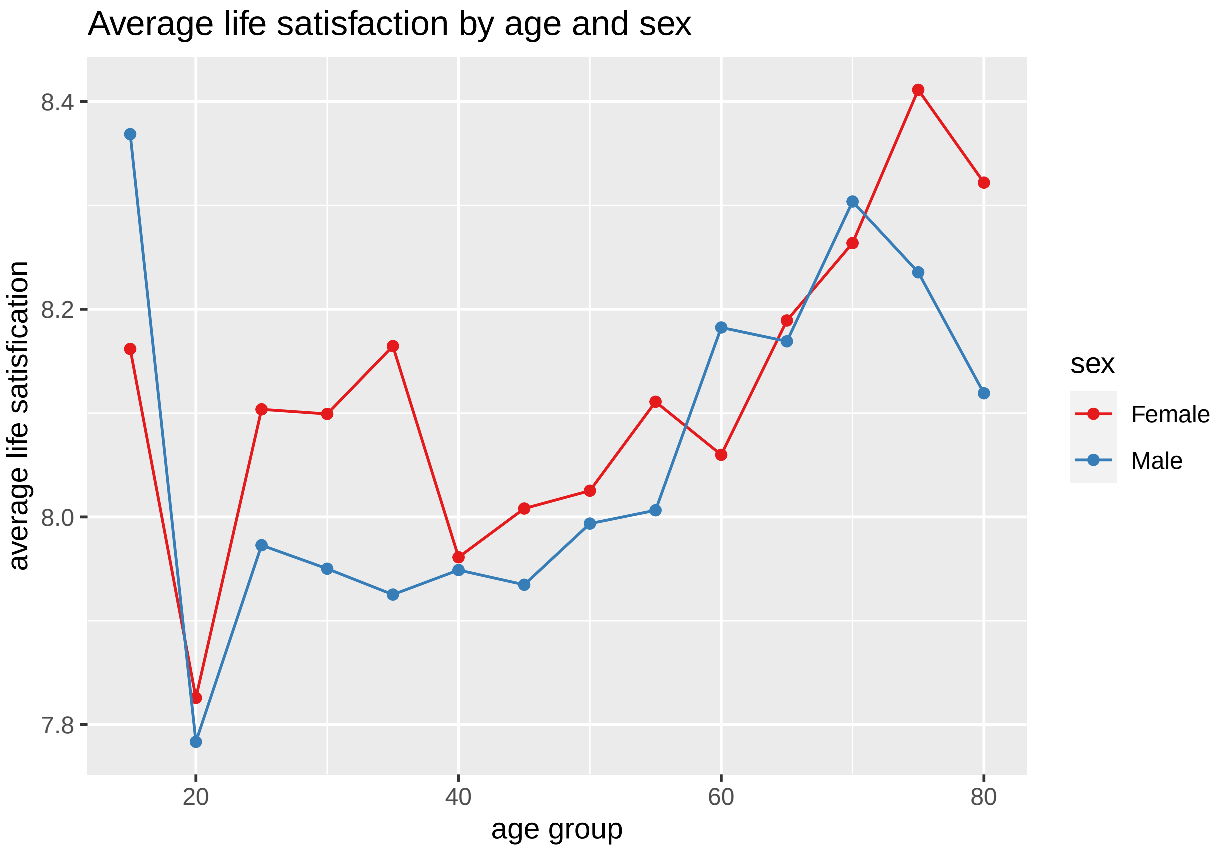
Now let's calculate the average of the 'life satisfaction' variable by age group and sex. This involves a `group_by` by two variables:

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

```
## `summarise()` regrouping output by 'age_group' (override with `.groups` argument)
```

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

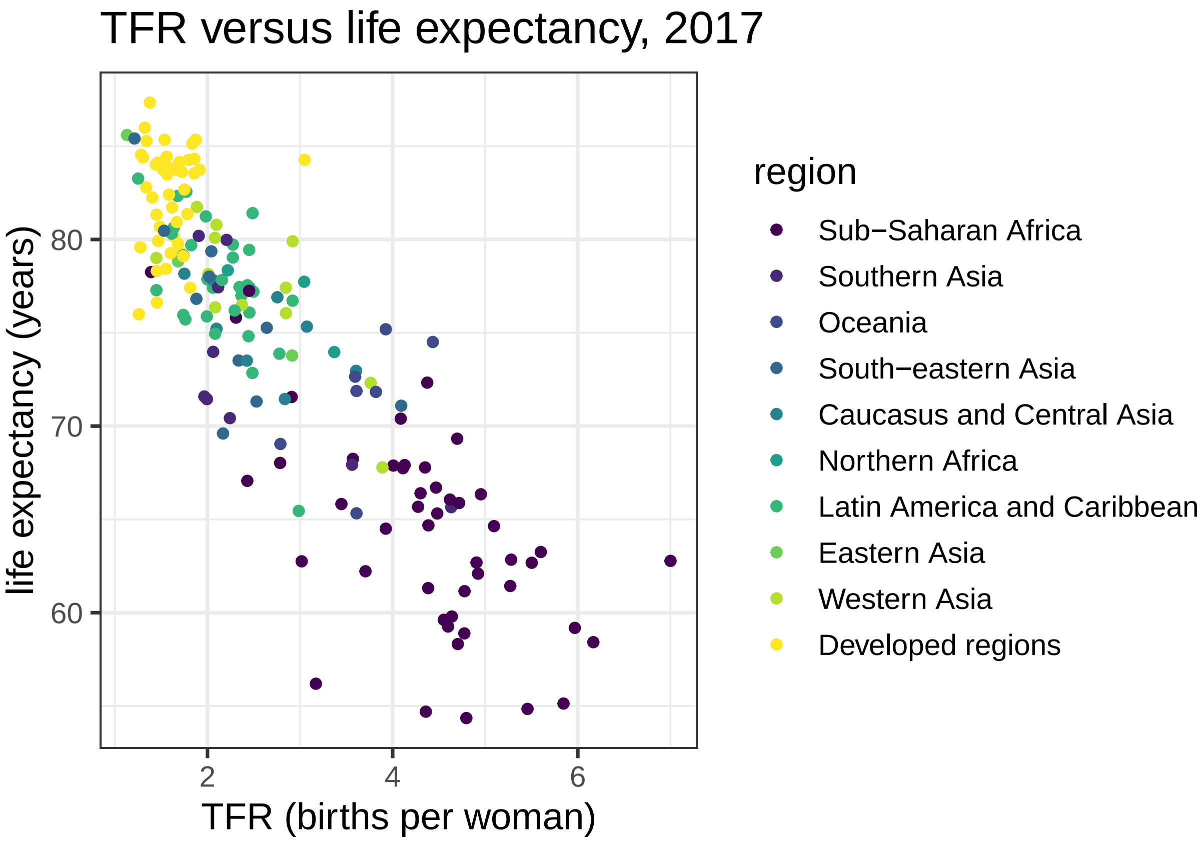
```
ggplot(data = life_satis_age_sex, aes(x = age_group, y = mean_life_satis, colour = sex)) +
  geom_point() +
  geom_line() +
  scale_color_brewer(palette = "Set1") + # change the color scheme
  ylab("average life satisfication") +
  xlab("age group") +
  ggtitle("Average life satisfaction by age and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



Scatter plots

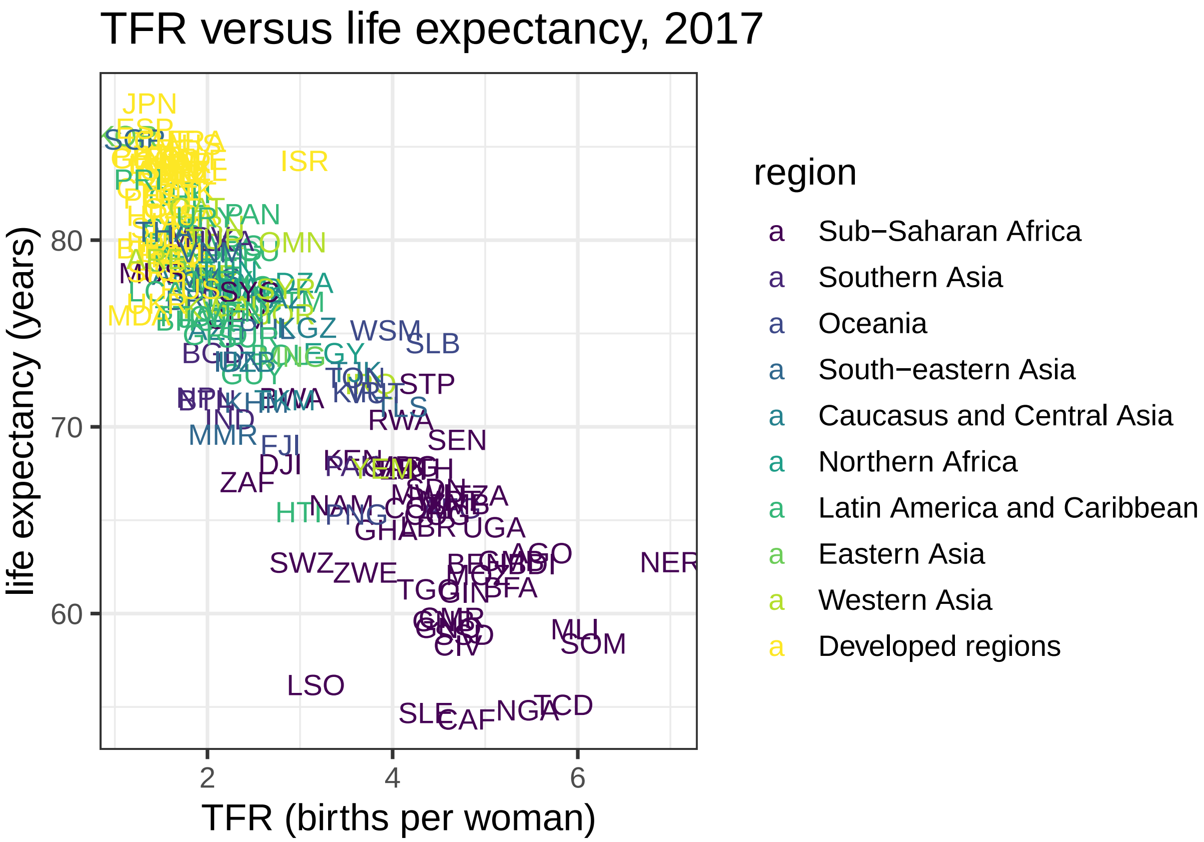
Let's use the country indicators dataset here. The example in the lecture slides is life expectancy versus TFR. We also used a new colour palette called `virdis`, these colours palettes are designed to be viewable in black and white as well.

```r
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)

```r
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:

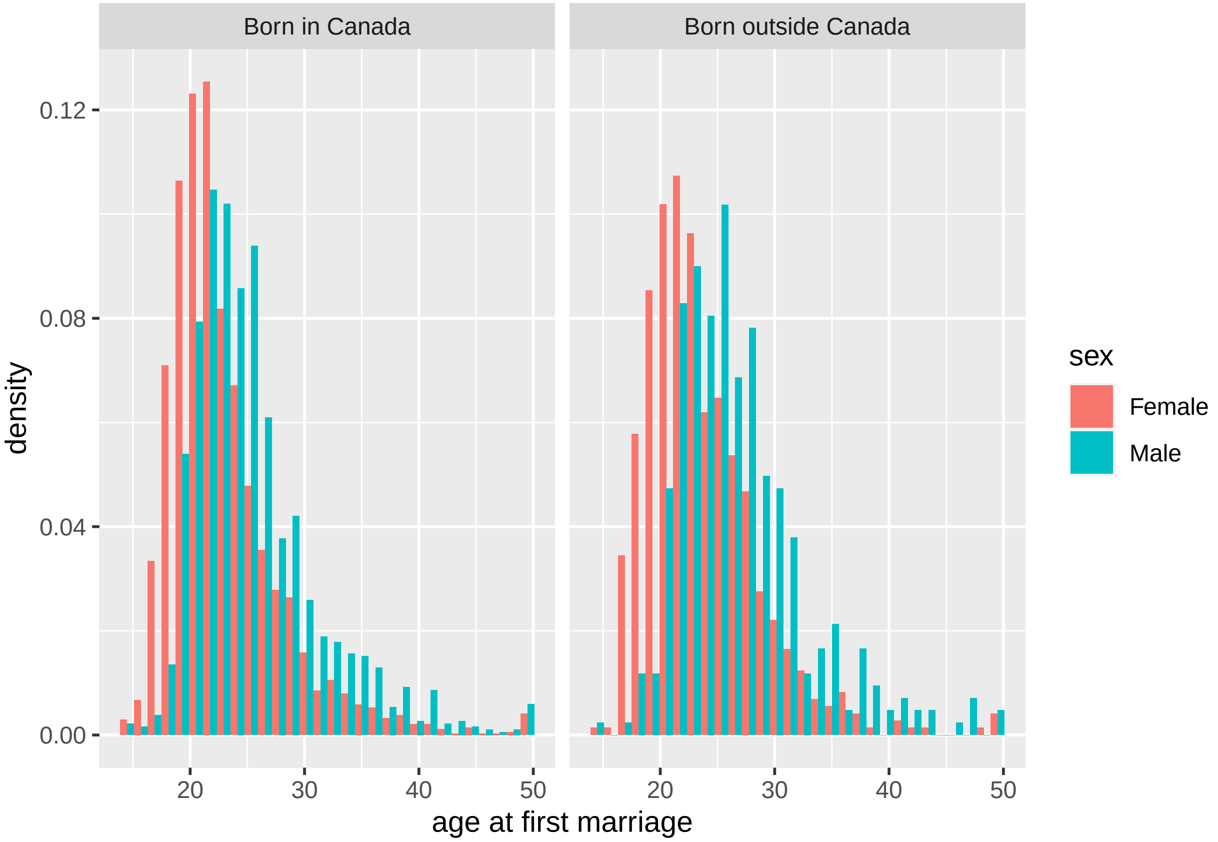
```r
gss_subset <- gss %>%
  filter(place_birth_canada %in% c("Born in Canada", "Born outside Canada"))
```

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

```r
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.