Week 3: Data visualization with ggplot

SOC6302 Winter 2023

Table of contents

1 By the end of this lab you should know

- how to create your own tibble (dataset)
- `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%`

- faceting

2 Creating your own dataset

Note that you can also create your own dataset using the `tibble` function. Note that each column is defined as a vector:

```
library(tidyverse)
my_fruit <- tibble(fruit = c("banana", "apple", "pear"),
                   count = c(2,4,1))
my_fruit
```

```
# A tibble: 3 x 2
  fruit  count
  <chr>  <dbl>
1 banana     2
2 apple      4
3 pear       1
```

3 Read in the data

We'll be using the GSS and country indicators data set.

```
library(tidyverse)

# Read the files
gss <- read_csv("../data/gss.csv")
country_ind <- read_csv("../data/country_indicators.csv")
```

4 `ggplot`

`ggplot` is a powerful visualization package. It provides many options to make beautiful graphs, maps, plots of all sorts. We will look at some important graph types today.

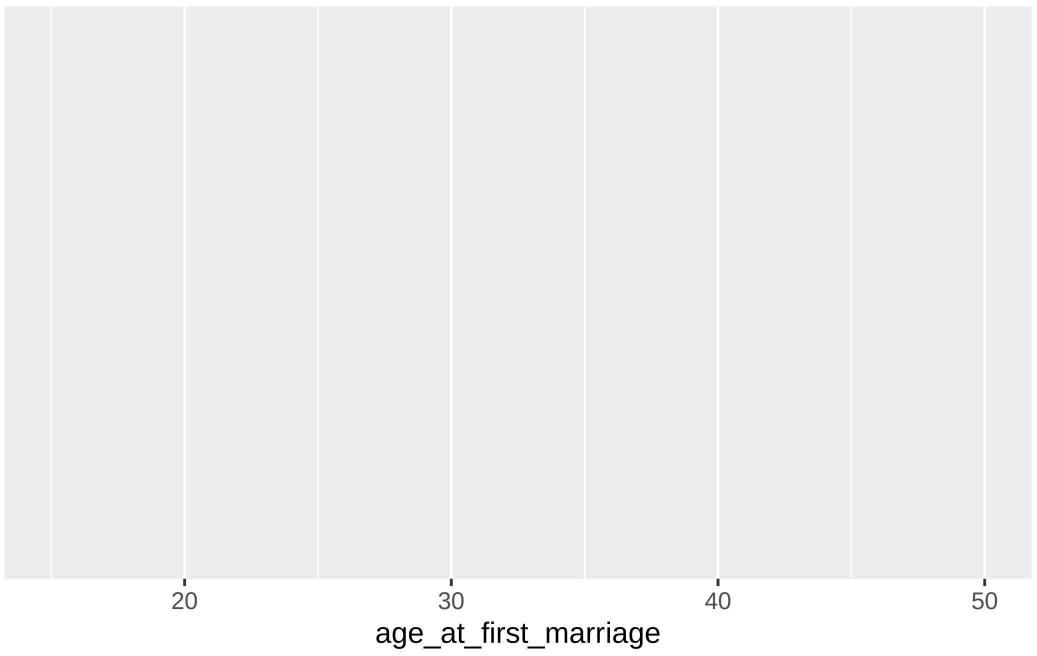
4.1 Histograms

`ggplot` works in layers:

- The first piece is the `ggplot` function, where you specify
  - the data set where the data to be plotted are contained
  - the `aes` function, where you specify the variables to be plotted
- We then specify what type of plot to make, which are functions prefixed by `geom_`
- We can then customize titles, labels, themes, etc

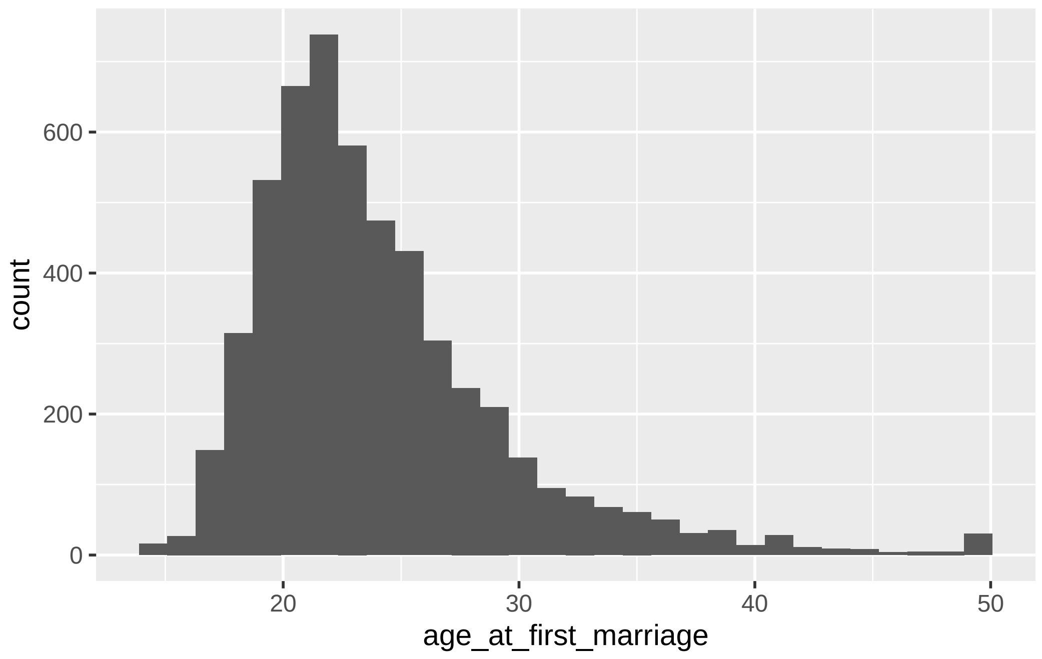
So for a histogram of ages at first marriage in the GSS, we start with specifying the dataset and variable:

```
ggplot(data = gss, aes(age_at_first_marriage))
```



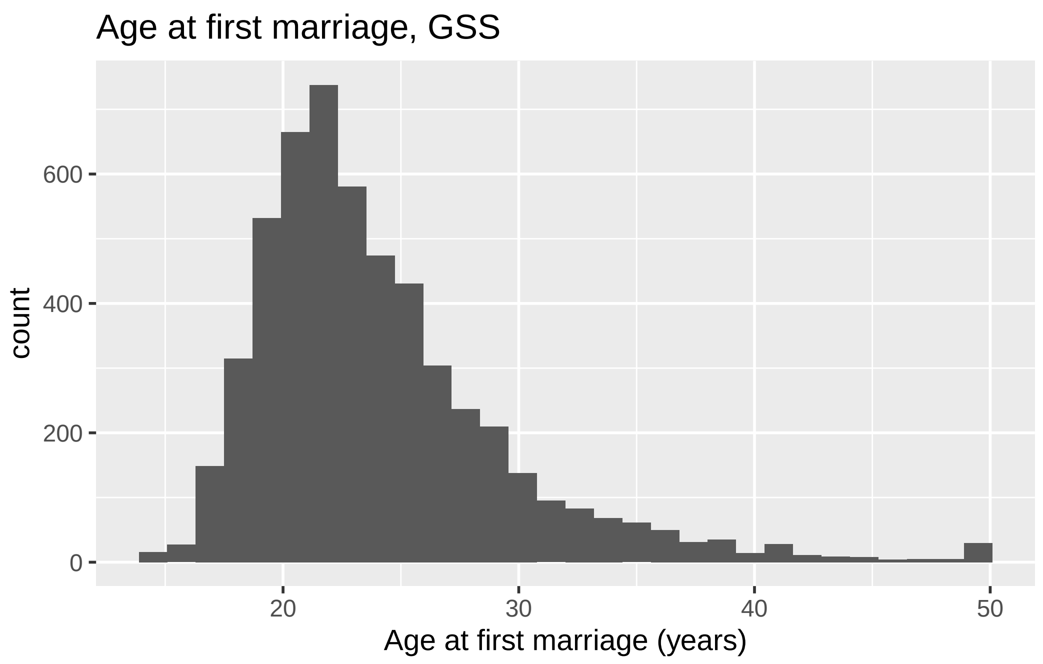
Note this is just a blank box, but it has the right x axis. Add a histogram:

```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram()
```



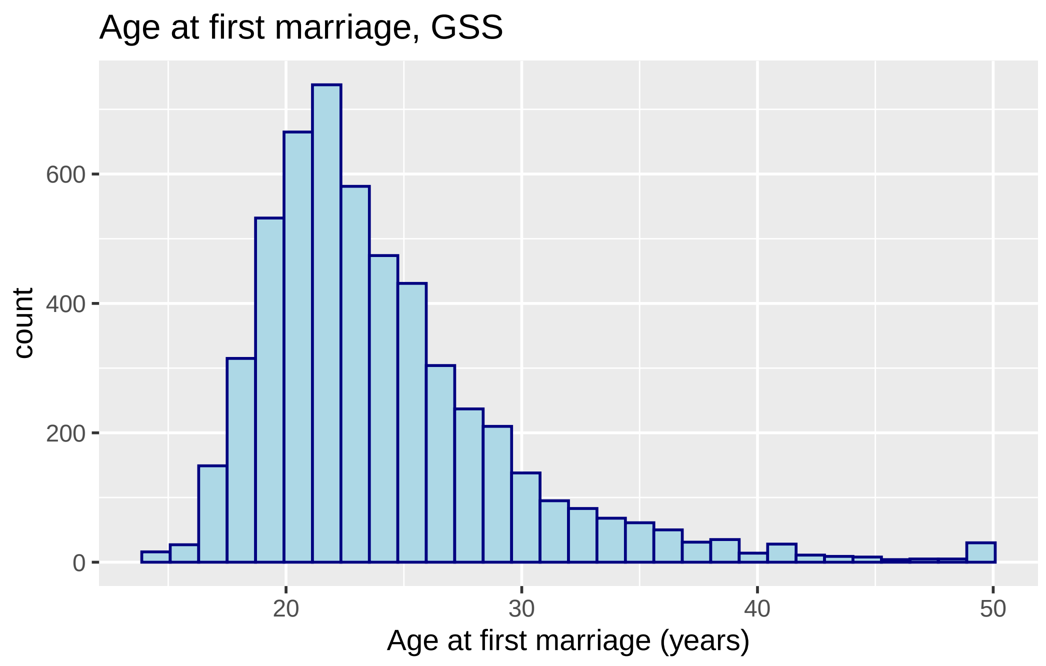
Customize the labels:

```
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram() +
  labs(title = "Age at first marriage, GSS", x = "Age at first marriage (years)")
```



Now we can change the color of the bars. Note for histograms, bar chats, box plots, `fill` is the main color choice (`color` changes the outline)

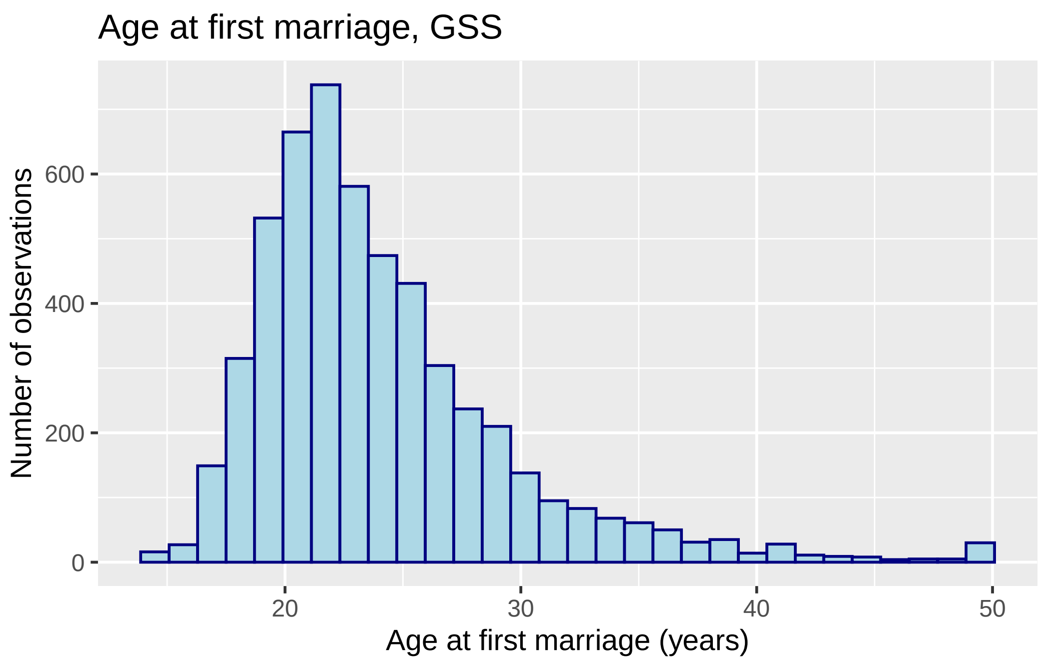
```r
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy") +
  labs(title = "Age at first marriage, GSS", x = "Age at first marriage (years)")
```



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

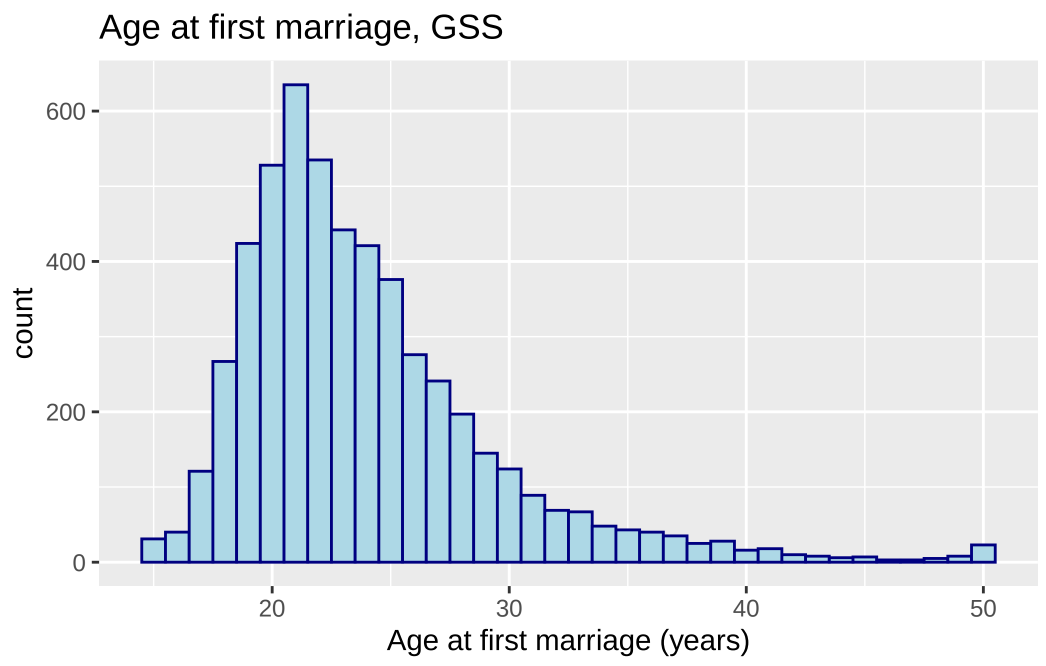
```
my_plot <- ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy")+
  labs(title = "Age at first marriage, GSS", x = "Age at first marriage (years)")

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

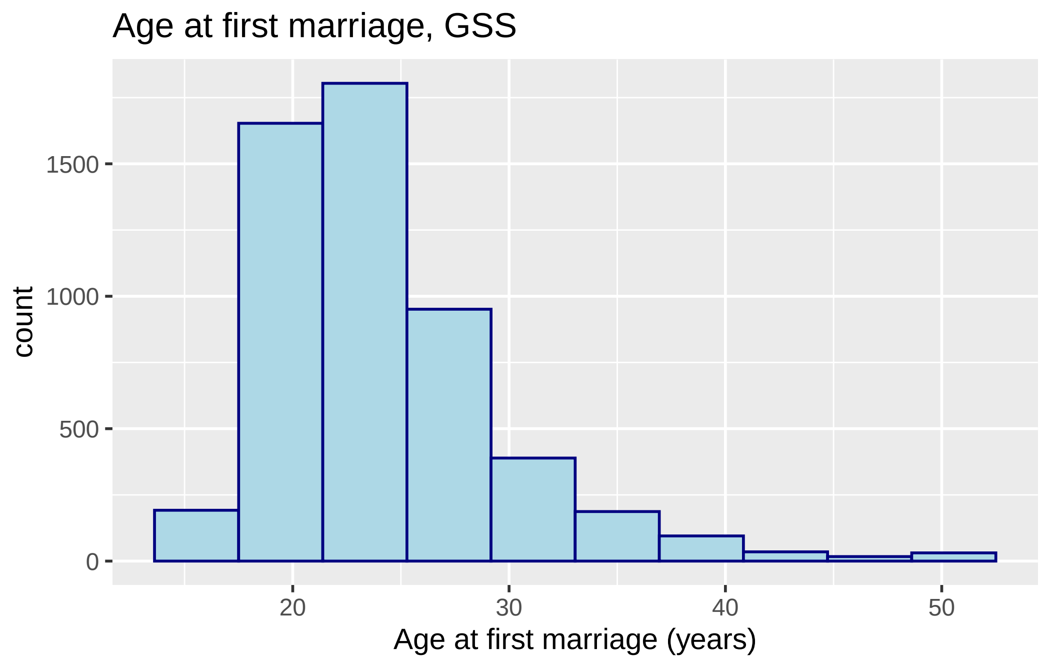


Histograms select a `binwidth` or section of the data and then count how many of the observations fall within that. Histograms look different depending on the size of the bins. You can also supply the number of bins that you want to create.

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

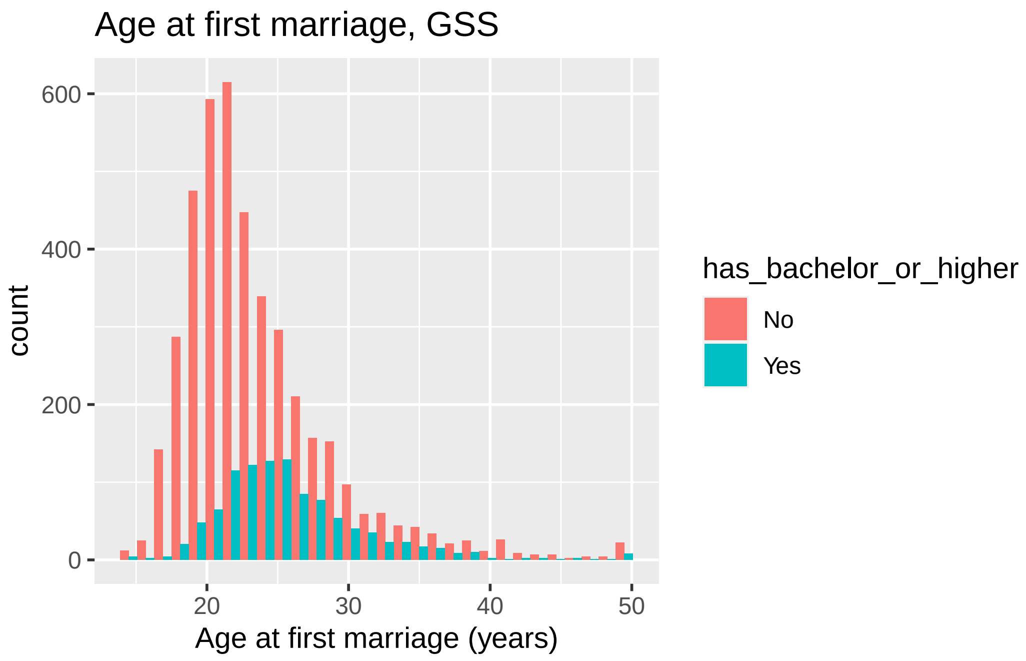


```r
ggplot(data = gss, aes(age_at_first_marriage)) +
  geom_histogram(fill = "lightblue", color = "navy", bins = 10)+
  labs(title = "Age at first marriage, GSS", x = "Age at first marriage (years)")
```



We can also plot by another variable to compare the plots by the categories of the variable. For example, we look at plots by whether or not people have at least a bachelor degree:

```
ggplot(data = gss |> drop_na(has_bachelor_or_higher), aes(age_at_first_marriage , fill = h
  geom_histogram(position = 'dodge') +
  labs(title = "Age at first marriage, GSS", x = "Age at first marriage (years)")
```



Importantly, note that the fill color is now specified in the `aes` function, **because it depends on a variable**. Also note that when specifying the data, we have dropped the NAs in the `has_bachelor_or_higher` variable.

4.2 Bar charts

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

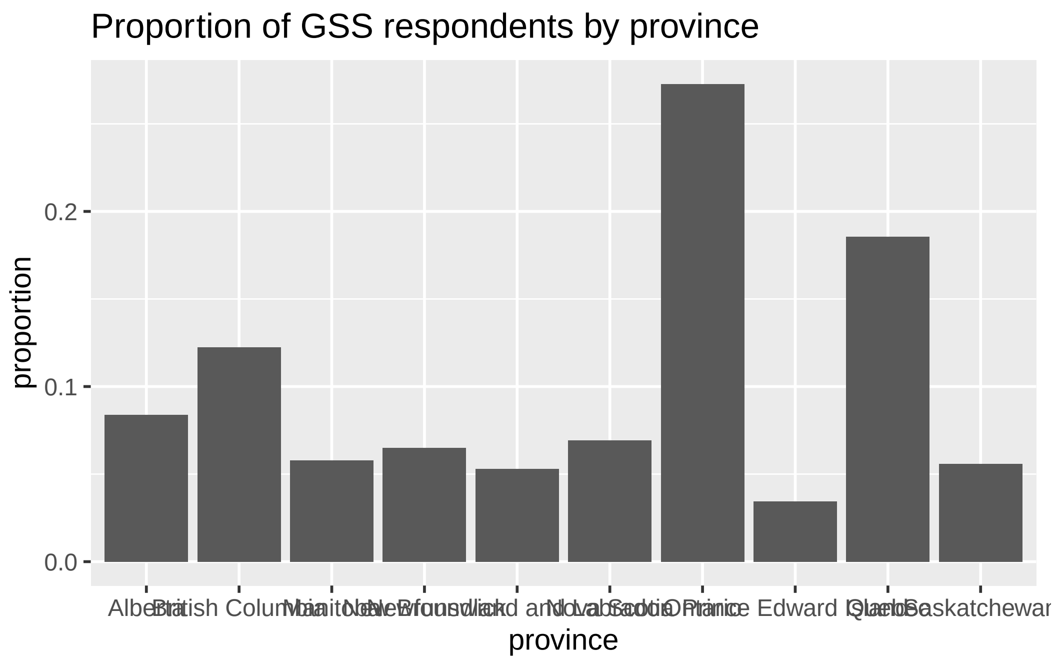
```
resp_by_prov <- gss |>
  group_by(province) |>
  tally() |>
  mutate(prop = n / sum(n))
```

```
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") +
  labs(title = "Proportion of GSS respondents by province", y = "proportion")
```

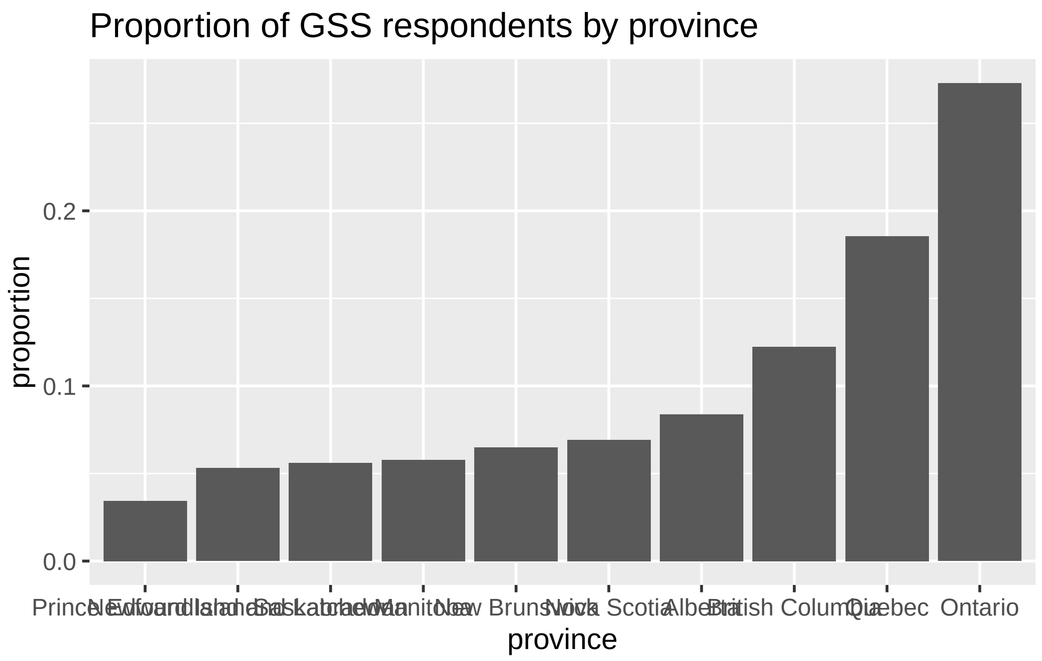


There are a few things here that would be nice to fix. Firstly, the categories are ordered alphabetically, which is the default. It would be better visually to order by proportion. We can do this using the `fct_reorder` function to alter (mutate) the province variable.

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

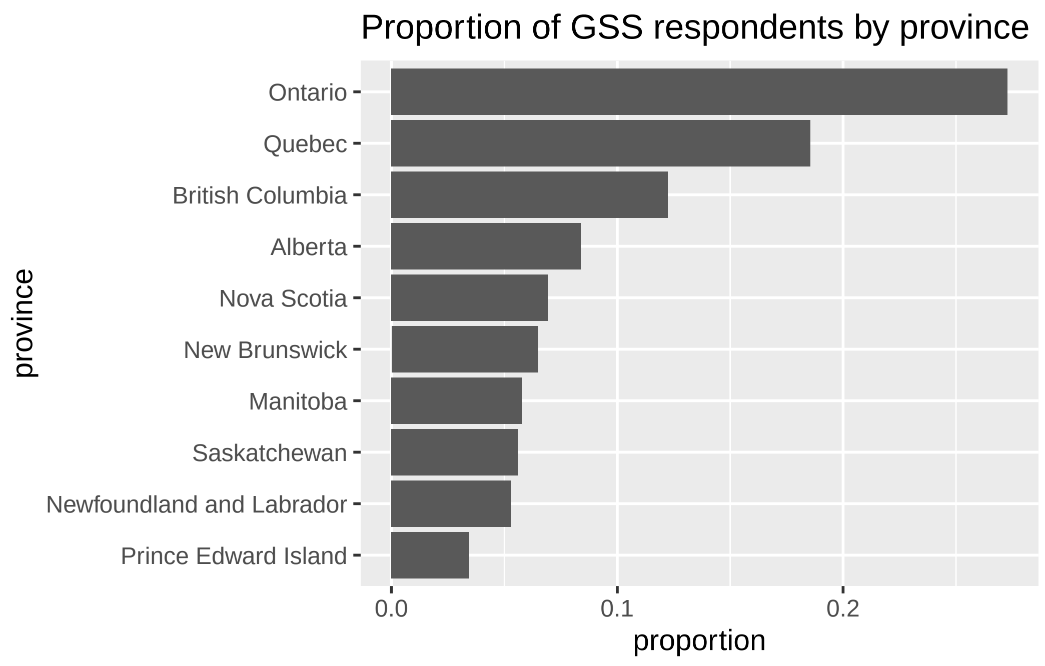
Now try plotting again.

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



To improve readability, could change to horizontal bar chart.

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

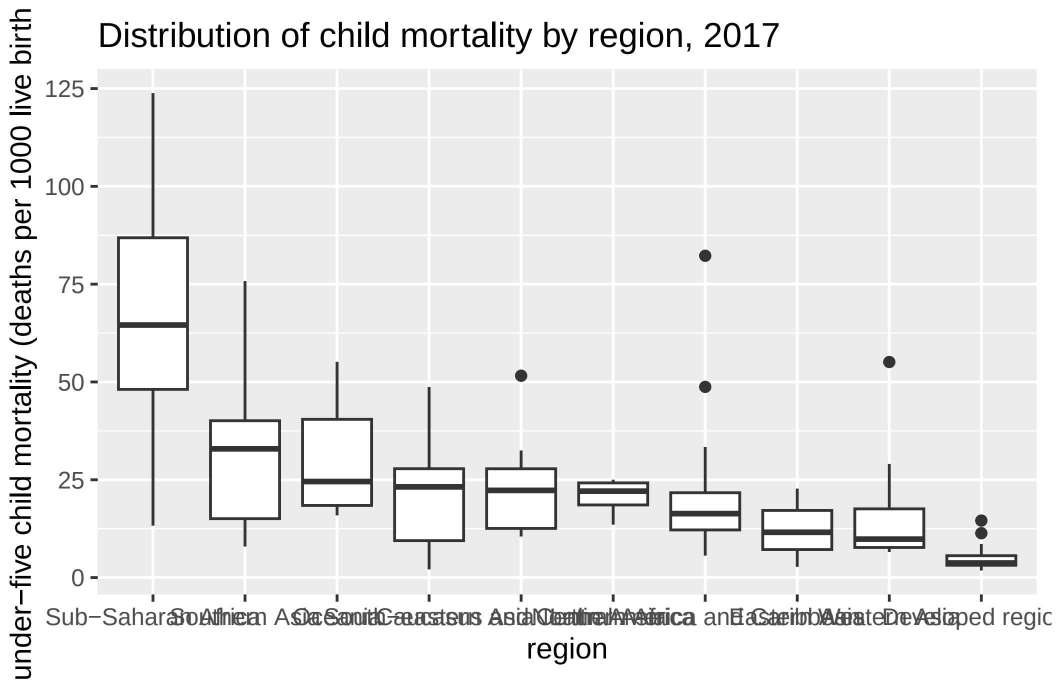


4.3 Box plots

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

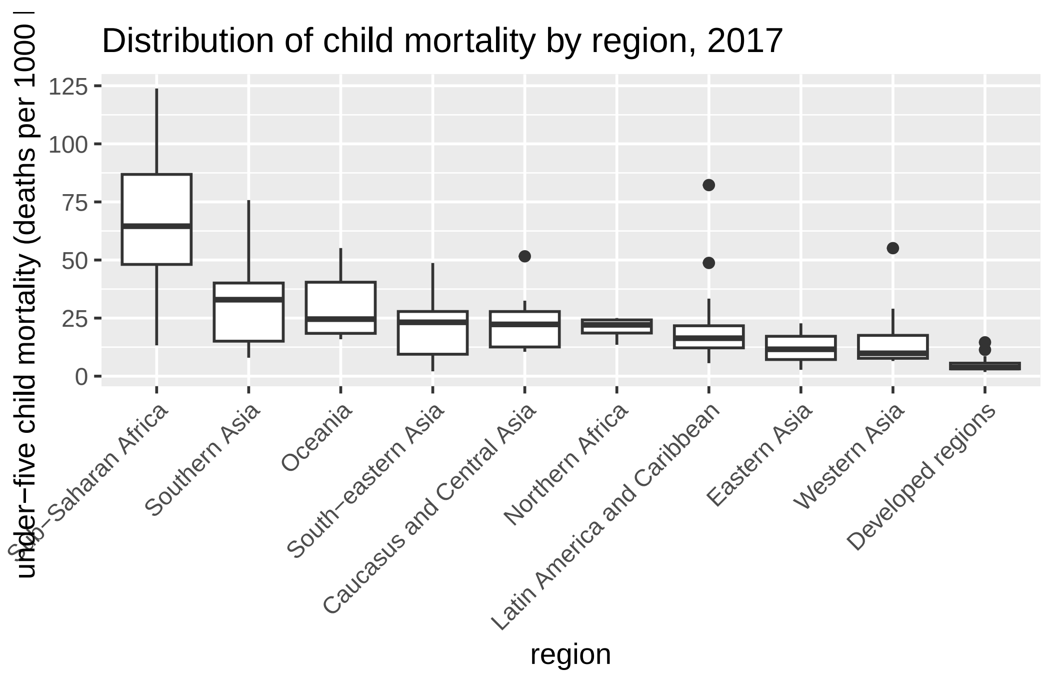
```
country_ind_2017 <- country_ind |>
  filter(year==2017) |>
  mutate(region = fct_reorder(region, -child_mort)) # descending order

ggplot(data = country_ind_2017, aes(x = region, y = child_mort)) +
  geom_boxplot() +
  labs(title = "Distribution of child mortality by region, 2017", y = "under-five child mo
```



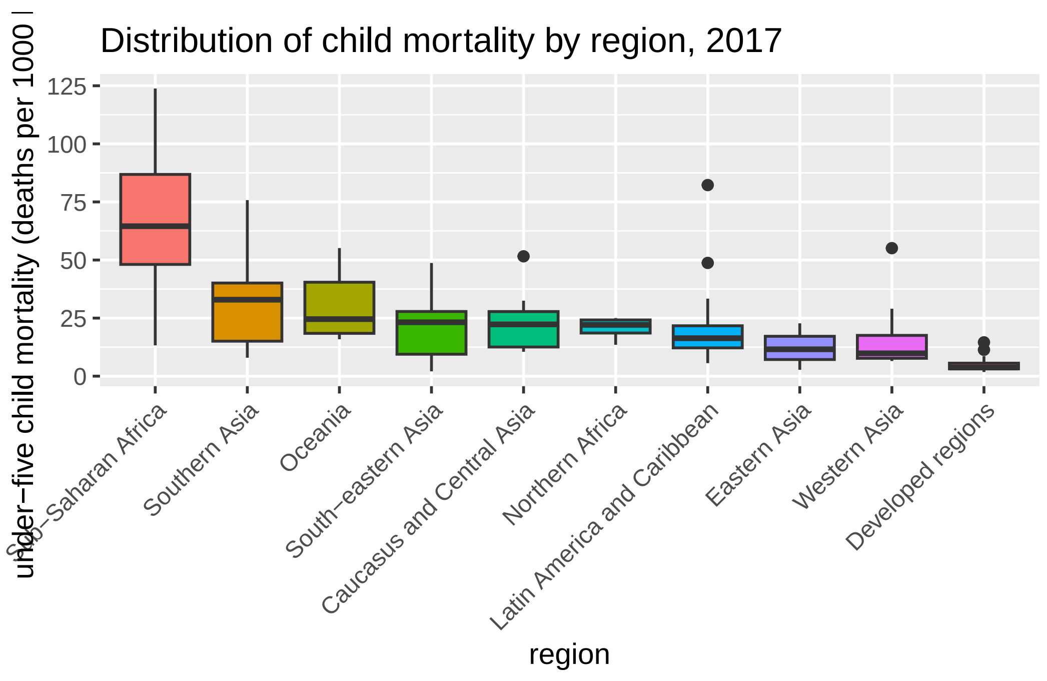
The labels on the x axis are hard to read. We could do the same as last time (switch to horizontal), or we can change the alignment of the labels:

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



4.4 Line graphs

Let's look at the mean life satisfaction 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)))

#check
gss |> select(age, age_group)
```

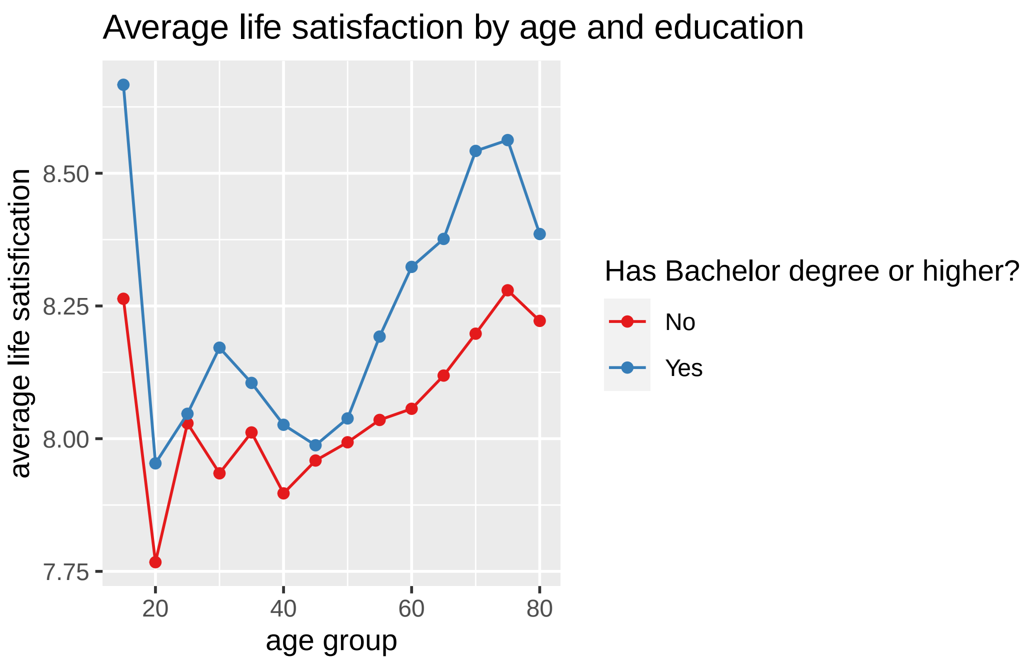
```
# A tibble: 20,602 x 2
     age age_group
   <dbl>     <dbl>
 1  52.7        50
 2  51.1        50
 3  63.6        60
 4  80          80
 5  28          25
 6  63          60
 7  58.8        55
 8  80          80
 9  63.8        60
10  25.2        25
# ... with 20,592 more rows
```

Now let's calculate the average of the 'life satisfaction' variable by age group and whether or not they had at least a bachelor's degree. This involves a `group_by` by two variables:

```
life_satis_age_bach <- gss |>
  drop_na(has_bachelor_or_higher) |>
  group_by(age_group, has_bachelor_or_higher) |>
  summarise(mean_life_satis = mean(feelings_life, na.rm = TRUE))
```

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

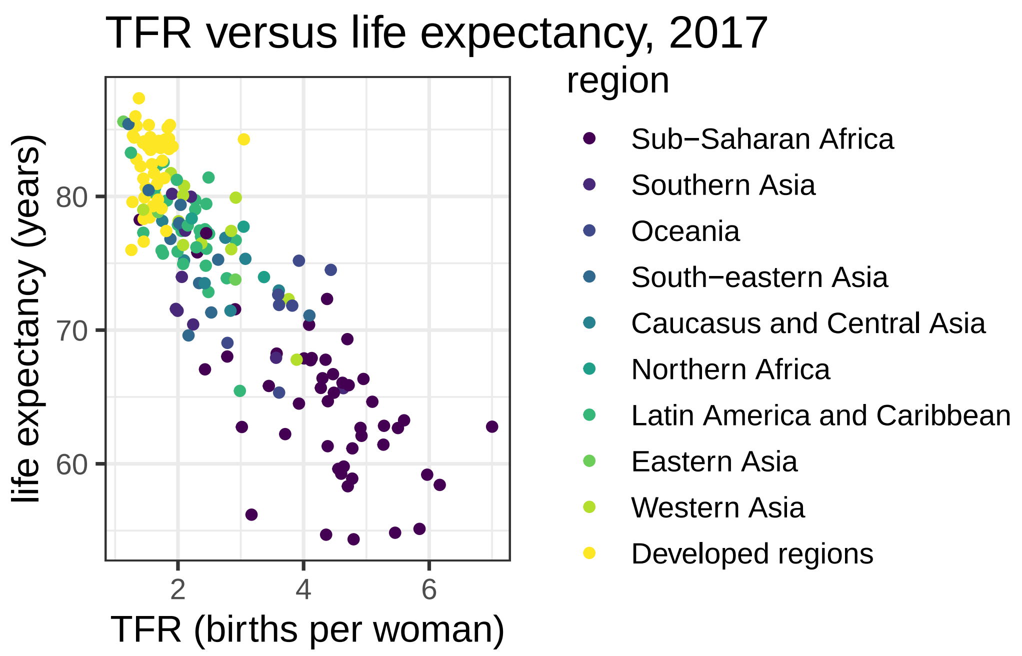
```
ggplot(data = life_satis_age_bach, aes(x = age_group,
                                       y = mean_life_satis,
                                       colour = has_bachelor_or_higher)) +
  geom_point() +
  geom_line() +
  scale_color_brewer(palette = "Set1", name = "Has Bachelor degree or higher?") + # change
  labs(title = "Average life satisfaction by age and education", x = "age group", y = "ave
```



4.5 Scatter plots

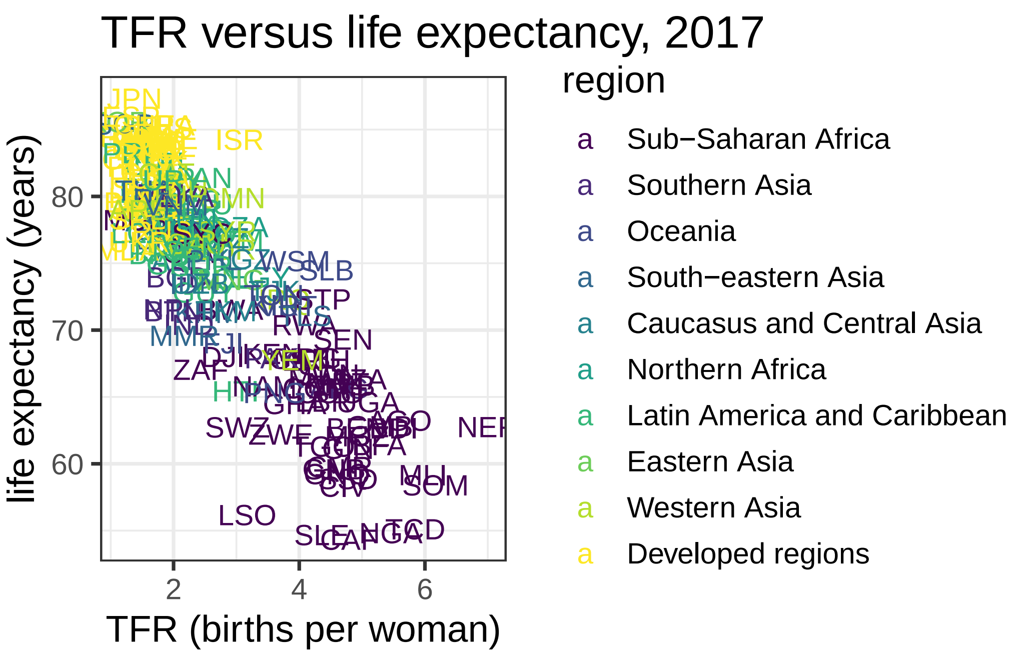
Let's use the country indicators dataset here. The example in the lecture slides is life expectancy versus TFR. We also used a new colour palette called `virdis`, these colours palettes are designed to be viewable in black and white as well.

```
ggplot(country_ind_2017, aes(tfr, life_expectancy, color = region,)) +
geom_point() +
labs(title = "TFR versus life expectancy, 2017", y = "life expectancy (years)", x = "TFR
theme_bw(base_size = 14) +
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)
geom_text() +
labs(title = "TFR versus life expectancy, 2017", y = "life expectancy (years)", x = "TFR
theme_bw(base_size = 14)+
scale_color_viridis_d()
```



4.6 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 whether or not the respondent has at least a bachelor degree, 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:

```r
gss |>
  select(place_birth_canada) |>
  unique()
```

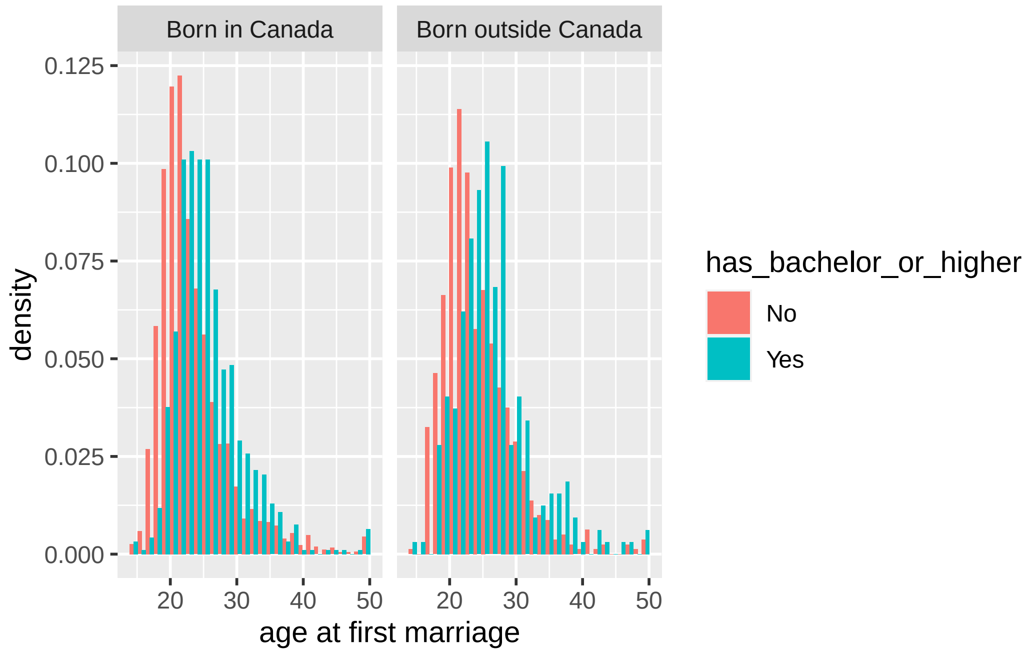
```
# A tibble: 4 x 1
  place_birth_canada
  <chr>
1 Born in Canada
2 Born outside Canada
3 <NA>
4 Don't know
```

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")) |>
  drop_na(has_bachelor_or_higher) # also remove the NAs from the education variable
```

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 = has_bachelor_or_higher)) +
  geom_histogram(position = 'dodge', aes(y = ..density..)) +
  facet_wrap(~place_birth_canada) +
  xlab("age at first marriage")
```



5 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.