daniel_jackson_module02_R_markdown

Daniel Jackson

September 8th, 2023

Module 02 HW

Chapter 3 HW: In this chapter, you can use the weight data set and perform all the actions covered here: selecting variables, filtering observations and reshaping.

```
# Load all libraries in that will be used in HW
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
library(readr)
library(ggExtra)
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x psych::%+%()    masks ggplot2::%+%()
## x psych::alpha()  masks ggplot2::alpha()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggrepel)

# Read in weight data
weight_df = read_csv('week_2/data/weight.csv')
```

```
## Rows: 6068 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): gender, handedness, race
## dbl (4): subjectid, height, weight, age
##
## 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.
```

```
# Let's select gender, weight and age variables
select(weight_df, gender, weight, age)
```

```
## # A tibble: 6,068 x 3
##    gender weight   age
##    <chr>   <dbl> <dbl>
##  1 male     81.5    41
##  2 male     72.6    35
##  3 male     92.9    42
##  4 male     79.4    31
##  5 male     94.6    21
##  6 male     80.2    39
##  7 male    116.     32
##  8 male     95.4    23
##  9 male     99.5    36
## 10 male     70.2    23
## # i 6,058 more rows
```

```
# Select variables in the range between two variables
select(weight_df, gender:handedness)
```

```
## # A tibble: 6,068 x 4
##    gender height weight handedness
##    <chr>   <dbl>  <dbl> <chr>
##  1 male     178.   81.5 right
##  2 male     170.   72.6 left
##  3 male     174.   92.9 left
##  4 male     166.   79.4 right
##  5 male     191.   94.6 right
##  6 male     172    80.2 right
##  7 male     181   116.  right
```

```
##  8 male     185    95.4 right
##  9 male     178.   99.5 right
## 10 male     181.   70.2 left
## # i 6,058 more rows
```

```
# Select same range of variables above using indices
select(weight_df, 2:5)
```

```
## # A tibble: 6,068 x 4
##    gender height weight handedness
##    <chr>   <dbl>  <dbl> <chr>
##  1 male     178.   81.5 right
##  2 male     170.   72.6 left
##  3 male     174.   92.9 left
##  4 male     166.   79.4 right
##  5 male     191.   94.6 right
##  6 male     172    80.2 right
##  7 male     181   116.  right
##  8 male     185    95.4 right
##  9 male     178.   99.5 right
## 10 male     181.   70.2 left
## # i 6,058 more rows
```

```
# Select variables that start with h. This returns height and handedness variables
select(weight_df, starts_with('h'))
```

```
## # A tibble: 6,068 x 2
##    height handedness
##     <dbl> <chr>
##  1   178. right
##  2   170. left
##  3   174. left
##  4   166. right
##  5   191. right
##  6   172  right
##  7   181  right
##  8   185  right
##  9   178. right
## 10   181. left
## # i 6,058 more rows
```

```
# Select variables that end with e. This returns race and age variables
select(weight_df, ends_with('e'))
```

```
## # A tibble: 6,068 x 2
##      age race
##    <dbl> <chr>
##  1    41 white
##  2    35 white
##  3    42 black
##  4    31 white
##  5    21 black
```

```
##  6    39 white
##  7    32 black
##  8    23 white
##  9    36 white
## 10    23 white
## # i 6,058 more rows
```

```
# Select variables that contains eigh. This returns height and weight
select(weight_df, contains('eigh'))
```

```
## # A tibble: 6,068 x 2
##    height weight
##     <dbl>  <dbl>
##  1   178.   81.5
##  2   170.   72.6
##  3   174.   92.9
##  4   166.   79.4
##  5   191.   94.6
##  6   172    80.2
##  7   181   116.
##  8   185    95.4
##  9   178.   99.5
## 10   181.   70.2
## # i 6,058 more rows
```

```
# Remove the age variable
select(weight_df, -age)
```

```
## # A tibble: 6,068 x 6
##    subjectid gender height weight handedness race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <chr>
##  1     10027 male     178.   81.5 right      white
##  2     10032 male     170.   72.6 left       white
##  3     10033 male     174.   92.9 left       black
##  4     10092 male     166.   79.4 right      white
##  5     10093 male     191.   94.6 right      black
##  6     10115 male     172    80.2 right      white
##  7     10117 male     181   116.  right      black
##  8     10237 male     185    95.4 right      white
##  9     10242 male     178.   99.5 right      white
## 10     10244 male     181.   70.2 left       white
## # i 6,058 more rows
```

```
# Remove multiple variables
select(weight_df, -age, -height, -weight)
```

```
## # A tibble: 6,068 x 4
##    subjectid gender handedness race
##        <dbl> <chr>  <chr>      <chr>
##  1     10027 male   right      white
##  2     10032 male   left       white
##  3     10033 male   left       black
```

```
##  4     10092 male   right      white
##  5     10093 male   right      black
##  6     10115 male   right      white
##  7     10117 male   right      black
##  8     10237 male   right      white
##  9     10242 male   right      white
## 10     10244 male   left       white
## # i 6,058 more rows
```

```
# Remove multiple with range using variable names
select(weight_df, -(gender:weight))
```

```
## # A tibble: 6,068 x 4
##    subjectid handedness   age race
##        <dbl> <chr>      <dbl> <chr>
##  1     10027 right         41 white
##  2     10032 left          35 white
##  3     10033 left          42 black
##  4     10092 right         31 white
##  5     10093 right         21 black
##  6     10115 right         39 white
##  7     10117 right         32 black
##  8     10237 right         23 white
##  9     10242 right         36 white
## 10     10244 left          23 white
## # i 6,058 more rows
```

```
# Now remove multiple with range using indices
select(weight_df, -(2:4))
```

```
## # A tibble: 6,068 x 4
##    subjectid handedness   age race
##        <dbl> <chr>      <dbl> <chr>
##  1     10027 right         41 white
##  2     10032 left          35 white
##  3     10033 left          42 black
##  4     10092 right         31 white
##  5     10093 right         21 black
##  6     10115 right         39 white
##  7     10117 right         32 black
##  8     10237 right         23 white
##  9     10242 right         36 white
## 10     10244 left          23 white
## # i 6,058 more rows
```

```
# Select variables that are character vectors using select_if
select_if(weight_df, where(is.character))
```

```
## # A tibble: 6,068 x 3
##    gender handedness race
##    <chr>  <chr>      <chr>
##  1 male   right      white
```

```
##  2 male   left       white
##  3 male   left       black
##  4 male   right      white
##  5 male   right      black
##  6 male   right      white
##  7 male   right      black
##  8 male   right      white
##  9 male   right      white
## 10 male   left       white
## # i 6,058 more rows
```

```
# Select numeric vectors
select_if(weight_df, where(is.numeric))
```

```
## # A tibble: 6,068 x 4
##    subjectid height weight   age
##        <dbl>  <dbl>  <dbl> <dbl>
##  1     10027   178.   81.5    41
##  2     10032   170.   72.6    35
##  3     10033   174.   92.9    42
##  4     10092   166.   79.4    31
##  5     10093   191.   94.6    21
##  6     10115   172    80.2    39
##  7     10117   181   116.     32
##  8     10237   185    95.4    23
##  9     10242   178.   99.5    36
## 10     10244   181.   70.2    23
## # i 6,058 more rows
```

```
# Rename variable using select()
select(weight_df, participant = subjectid)
```

```
## # A tibble: 6,068 x 1
##    participant
##          <dbl>
##  1       10027
##  2       10032
##  3       10033
##  4       10092
##  5       10093
##  6       10115
##  7       10117
##  8       10237
##  9       10242
## 10       10244
## # i 6,058 more rows
```

```
# Use rename() function to rename variable. This updates name and returns full data frame
rename(weight_df, participant = subjectid)
```

```
## # A tibble: 6,068 x 7
##    participant gender height weight handedness   age race
```

```
##          <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1       10027 male     178.   81.5 right         41 white
##  2       10032 male     170.   72.6 left          35 white
##  3       10033 male     174.   92.9 left          42 black
##  4       10092 male     166.   79.4 right         31 white
##  5       10093 male     191.   94.6 right         21 black
##  6       10115 male     172    80.2 right         39 white
##  7       10117 male     181   116.  right         32 black
##  8       10237 male     185    95.4 right         23 white
##  9       10242 male     178.   99.5 right         36 white
## 10       10244 male     181.   70.2 left          23 white
## # i 6,058 more rows
```

```
# Select observations in weight_df using slice and filter
# Select rows 50, 100, 150, 250, 300, 350
slice(weight_df, c(50, 100, 150, 200, 250, 300, 350))
```

```
## # A tibble: 7 x 7
##   subjectid gender height weight handedness   age race
##       <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
## 1     10497 male     167.   85.6 right         40 white
## 2     10701 male     176.   96.8 right         27 white
## 3     10848 male     193.   80.6 right         21 white
## 4     11161 male     175.   84   right         33 black
## 5     11290 male     161.   56.8 right         30 pacific_islander
## 6     11440 male     175.   88.4 right         29 white
## 7     11758 male     160.   72.5 right         48 white
```

```
# Slice data frame by 1st to 100th row
slice(weight_df, 1:100)
```

```
## # A tibble: 100 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10027 male     178.   81.5 right         41 white
##  2     10032 male     170.   72.6 left          35 white
##  3     10033 male     174.   92.9 left          42 black
##  4     10092 male     166.   79.4 right         31 white
##  5     10093 male     191.   94.6 right         21 black
##  6     10115 male     172    80.2 right         39 white
##  7     10117 male     181   116.  right         32 black
##  8     10237 male     185    95.4 right         23 white
##  9     10242 male     178.   99.5 right         36 white
## 10     10244 male     181.   70.2 left          23 white
## # i 90 more rows
```

```
# Slice data frame to remove rows 100 to 300
slice(weight_df, -(100:300))
```

```
## # A tibble: 5,867 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
```

```
##  1     10027 male     178.   81.5 right         41 white
##  2     10032 male     170.   72.6 left          35 white
##  3     10033 male     174.   92.9 left          42 black
##  4     10092 male     166.   79.4 right         31 white
##  5     10093 male     191.   94.6 right         21 black
##  6     10115 male     172    80.2 right         39 white
##  7     10117 male     181   116.  right         32 black
##  8     10237 male     185    95.4 right         23 white
##  9     10242 male     178.   99.5 right         36 white
## 10     10244 male     181.   70.2 left          23 white
## # i 5,857 more rows
```

```
# Slice data frame to list observation from row 5000 to end of data frame using n()
slice(weight_df, 5000:n())
```

```
## # A tibble: 1,069 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     21449 female   168.   63.3 right         41 black
##  2     21453 female   155    52.6 right         23 hispanic
##  3     21462 female   163    76.6 right         42 white
##  4     21463 female   169.   93.4 left          35 black
##  5     21464 female   156.   66.7 right         30 black
##  6     21482 female   163.   62.2 right         24 hispanic
##  7     21501 female   167.   94.2 right         34 black
##  8     21516 female   160.   64.1 right         26 hispanic
##  9     21524 female   155.   69.2 right         30 black
## 10     21525 female   164.   64.6 right         33 white
## # i 1,059 more rows
```

```
# Use filter to filter observations. This code returns all right handed observations
filter(weight_df, handedness == 'right')
```

```
## # A tibble: 5,350 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10027 male     178.   81.5 right         41 white
##  2     10092 male     166.   79.4 right         31 white
##  3     10093 male     191.   94.6 right         21 black
##  4     10115 male     172    80.2 right         39 white
##  5     10117 male     181   116.  right         32 black
##  6     10237 male     185    95.4 right         23 white
##  7     10242 male     178.   99.5 right         36 white
##  8     10246 male     178    88.2 right         32 white
##  9     10265 male     181.  104.  right         36 black
## 10     10272 male     186.  112.  right         26 white
## # i 5,340 more rows
```

```
# Filter with multiple variables. Let's look at right handed females
filter(weight_df, handedness == 'right', gender == 'female')
```

```
## # A tibble: 1,773 x 7
```

```
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10037 female    156    65.7 right         26 black
##  2     10038 female    166.   53.4 right         21 hispanic
##  3     10042 female    171.   66.3 right         23 white
##  4     10043 female    166    78.2 right         22 black
##  5     10051 female    157.   88.6 right         45 white
##  6     10061 female    164.   73.2 right         21 white
##  7     10070 female    167.   76   right         23 pacific_islander
##  8     10080 female    159    68.4 right         37 white
##  9     10095 female    171.   99   right         33 black
## 10     10101 female    167.   74.2 right         36 black
## # i 1,763 more rows
```

```
# Filter observations by right handed females whose height is less than or equal to 165
filter(weight_df, handedness == 'right'& gender == 'female' & height <= 165)
```

```
## # A tibble: 1,135 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10037 female    156    65.7 right         26 black
##  2     10051 female    157.   88.6 right         45 white
##  3     10061 female    164.   73.2 right         21 white
##  4     10080 female    159    68.4 right         37 white
##  5     10121 female    160.   70.9 right         30 hispanic
##  6     10129 female    165.   62   right         19 white
##  7     10149 female    152.   80   right         31 white
##  8     10173 female    162.   49.1 right         31 white
##  9     10196 female    164.   71.5 right         19 black
## 10     10210 female    158.   70.8 right         23 hispanic
## # i 1,125 more rows
```

```
# Use mutate to add new variable called right_male that shows TRUE when gender is male and handedness i
mutate(weight_df, right_male = ((handedness == 'right') & (gender == 'male')))
```

```
## # A tibble: 6,068 x 8
##    subjectid gender height weight handedness   age race  right_male
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr> <lgl>
##  1     10027 male      178.   81.5 right         41 white TRUE
##  2     10032 male      170.   72.6 left          35 white FALSE
##  3     10033 male      174.   92.9 left          42 black FALSE
##  4     10092 male      166.   79.4 right         31 white TRUE
##  5     10093 male      191.   94.6 right         21 black TRUE
##  6     10115 male      172    80.2 right         39 white TRUE
##  7     10117 male      181   116.  right         32 black TRUE
##  8     10237 male      185    95.4 right         23 white TRUE
##  9     10242 male      178.   99.5 right         36 white TRUE
## 10     10244 male      181.   70.2 left          23 white FALSE
## # i 6,058 more rows
```

```
# Create multiple variables. Create right_female and left_female
mutate(weight_df, right_male = ((handedness == 'right') & (gender == 'male')),
                  left_female = ((handedness == 'left') & (gender == 'female')))
```

```
## # A tibble: 6,068 x 9
##    subjectid gender height weight handedness   age race  right_male left_female
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr> <lgl>      <lgl>
##  1     10027 male     178.   81.5 right         41 white TRUE       FALSE
##  2     10032 male     170.   72.6 left          35 white FALSE      FALSE
##  3     10033 male     174.   92.9 left          42 black FALSE      FALSE
##  4     10092 male     166.   79.4 right         31 white TRUE       FALSE
##  5     10093 male     191.   94.6 right         21 black TRUE       FALSE
##  6     10115 male     172    80.2 right         39 white TRUE       FALSE
##  7     10117 male     181   116.  right         32 black TRUE       FALSE
##  8     10237 male     185    95.4 right         23 white TRUE       FALSE
##  9     10242 male     178.   99.5 right         36 white TRUE       FALSE
## 10     10244 male     181.   70.2 left          23 white FALSE      FALSE
## # i 6,058 more rows
```

```
# Use mutate_all to turn all variables into character vectors
mutate_all(weight_df, as.character)
```

```
## # A tibble: 6,068 x 7
##    subjectid gender height weight handedness age   race
##    <chr>     <chr>  <chr>  <chr>  <chr>      <chr> <chr>
##  1 10027     male   177.6  81.5   right      41    white
##  2 10032     male   170.2  72.6   left       35    white
##  3 10033     male   173.5  92.9   left       42    black
##  4 10092     male   165.5  79.4   right      31    white
##  5 10093     male   191.4  94.6   right      21    black
##  6 10115     male   172    80.2   right      39    white
##  7 10117     male   181    116.2  right      32    black
##  8 10237     male   185    95.4   right      23    white
##  9 10242     male   177.7  99.5   right      36    white
## 10 10244     male   181.1  70.2   left       23    white
## # i 6,058 more rows
```

```
# Use transmute to create right_male variable and return only that new variable
transmute(weight_df, right_male = ((handedness == 'right') & (gender == 'male')))
```

```
## # A tibble: 6,068 x 1
##    right_male
##    <lgl>
##  1 TRUE
##  2 FALSE
##  3 FALSE
##  4 TRUE
##  5 TRUE
##  6 TRUE
##  7 TRUE
##  8 TRUE
##  9 TRUE
## 10 FALSE
## # i 6,058 more rows
```

```r
# Use transmute to create right_male and left_female and return only those variables
transmute(weight_df, right_male = ((handedness == 'right') & (gender == 'male')),
       left_female = ((handedness == 'left') & (gender == 'female')))
```

```
## # A tibble: 6,068 x 2
##    right_male left_female
##    <lgl>      <lgl>
##  1 TRUE       FALSE
##  2 FALSE      FALSE
##  3 FALSE      FALSE
##  4 TRUE       FALSE
##  5 TRUE       FALSE
##  6 TRUE       FALSE
##  7 TRUE       FALSE
##  8 TRUE       FALSE
##  9 TRUE       FALSE
## 10 FALSE      FALSE
## # i 6,058 more rows
```

```r
# Use arrange to sort by gender then by age
arrange(weight_df, gender, age)
```

```
## # A tibble: 6,068 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     26138 female   171.   74.3 right         17 black
##  2     12469 female   182.   88.2 right         18 white
##  3     16084 female   160    69.9 right         18 asian
##  4     18075 female   166.   61.3 left          18 white
##  5     18079 female   154.   57.7 right         18 white
##  6     18134 female   163.   59.8 right         18 hispanic
##  7     18163 female   168.   60.6 right         18 black
##  8     18190 female   166.   58.3 right         18 white
##  9     18193 female   166.   56.4 right         18 white
## 10     18288 female   156.   51.9 right         18 black
## # i 6,058 more rows
```

```r
# Use arrange to sort by subjectid in reverse
arrange(weight_df, desc(subjectid))
```

```
## # A tibble: 6,068 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1    920103 female   164.   61   right         27 hispanic
##  2     29511 female   162.   63.2 right         31 hispanic
##  3     29503 female   164.   76.2 right         40 black
##  4     29502 female   161.   71.7 right         40 hispanic
##  5     29501 female   169.   83.2 right         51 hispanic
##  6     29498 female   172.   76.8 right         25 hispanic
##  7     29497 female   159.   55.8 left          32 white
##  8     29496 female   153.   47.4 right         30 white
##  9     29495 female   160.   69.9 right         30 white
```

```
## 10     29494 female   158.   65.2 right         41 hispanic
## # i 6,058 more rows
```

```
# Use sample_frac to subsample weight_df by randomly sampling 20% of data frame
sample_frac(weight_df, 0.2)
```

```
## # A tibble: 1,214 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     25960 female   156.   80.4 right         46 black
##  2     13872 male     178   102.  right         44 white
##  3     23791 male     179.   87.4 right         27 white
##  4     27918 male     184.  106.  right         21 black
##  5     18385 female   161.   63.3 left          19 white
##  6     10674 male     170.   73   left          24 white
##  7     28168 female   167.   65   right         28 hispanic
##  8     15756 male     176.   87.3 right         33 hispanic
##  9     18088 female   183.   93.1 right         26 white
## 10     16955 male     183   105.  right         42 white
## # i 1,204 more rows
```

```
# Use sample_n to sample specified number of observations. We will do 100 random observations
sample_n(weight_df, 100)
```

```
## # A tibble: 100 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     29287 female   160    71.3 right         28 white
##  2     28258 female   165.   69.4 right         23 white
##  3     28247 female   161.   75.2 right         23 asian
##  4     26638 male     180    81.9 right         28 black
##  5     13733 female   156    60.2 right         46 black
##  6     12821 male     165.   77.6 right         43 white
##  7     23761 male     180.   88.4 right         32 black
##  8     15872 male     170.   82.3 right         26 white
##  9     23664 female   163.   65.8 right         28 white
## 10     28263 female   162.   67.8 left          21 white
## # i 90 more rows
```

```
# Use top_n to select top 100 observations according to age variable. This returns 114 rows because the
top_n(weight_df, 100, age)
```

```
## # A tibble: 114 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     11353 male     190   114.  right         51 white
##  2     11360 male     170.   78.4 right         53 white
##  3     11897 male     184.  102.  right         51 white
##  4     12904 male     171    84   right         51 white
##  5     13712 male     174.   99.8 right         51 hispanic
##  6     13718 male     164.   69.7 right         56 white
##  7     13785 male     169    73.6 right         53 white
```

```
##  8     13871 male     179.   81.2 left          53 white
##  9     13884 male     185.   92.8 right         51 white
## 10     13967 male     173.   73.4 right         53 white
## # i 104 more rows
```

```
# Use top_n to select bottom 100 observations by age
top_n(weight_df, -100, age)
```

```
## # A tibble: 114 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     12247 male     172.   65.7 right         18 white
##  2     12428 male     171.   69.2 left          18 white
##  3     12467 male     176.   62.1 right         18 white
##  4     13070 male     183.   86   right         18 black
##  5     13084 male     173.   72.3 right         18 native_american
##  6     14371 male     179    81.9 left          18 white
##  7     14376 male     181    86.1 right         18 white
##  8     14389 male     179.   84   right         18 white
##  9     14393 male     177.   67   right         18 white
## 10     14488 male     172.   93.5 right         18 white
## # i 104 more rows
```

```
# Use summarize() to calculate summary stats on weight_df. Find mean, median and sd of height
summarize(weight_df,
          mean_height = mean(height, na.rm = TRUE),
          median_height = median(height, na.rm = TRUE),
          sd_height = sd(height, na.rm = TRUE))
```

```
## # A tibble: 1 x 3
##   mean_height median_height sd_height
##         <dbl>         <dbl>     <dbl>
## 1        171.          172.      9.00
```

```
# Use summarize_all to apply summary function to all variables. Apply n_distinct to count unique values
summarize_all(weight_df, n_distinct)
```

```
## # A tibble: 1 x 7
##   subjectid gender height weight handedness   age  race
##       <int>  <int>  <int>  <int>      <int> <int> <int>
## 1      6068      2    481    783          3    42     7
```

```
# Use summarize_if to find mean of all numeric variables
summarize_if(weight_df, is.numeric, ~mean(., na.rm = TRUE))
```

```
## # A tibble: 1 x 4
##   subjectid height weight   age
##       <dbl>  <dbl>  <dbl> <dbl>
## 1    20757.   171.   79.7  29.8
```

```r
# Use summarize_at to find mean, median and sd of weight
summarize_at(weight_df,
             vars(weight),
             list(mean = ~mean(., na.rm = TRUE),
                  median = ~median(., na.rm = TRUE),
                  sd = ~sd(., na.rm = TRUE)
             )
)
```

```
## # A tibble: 1 x 3
##    mean median    sd
##   <dbl>  <dbl> <dbl>
## 1  79.7   78.5  15.7
```

```r
# Calculate same three summary stats for three variables
summarize_at(weight_df,
             vars(weight, height, age),
             list(mean = ~mean(., na.rm = TRUE),
                  median = ~median(., na.rm = TRUE),
                  sd = ~sd(., na.rm = TRUE)
             )
)
```

```
## # A tibble: 1 x 9
##   weight_mean height_mean age_mean weight_median height_median age_median
##         <dbl>       <dbl>    <dbl>         <dbl>         <dbl>      <dbl>
## 1        79.7        171.     29.8          78.5          172.         28
## # i 3 more variables: weight_sd <dbl>, height_sd <dbl>, age_sd <dbl>
```

```r
# Use group_by function to group by gender
weight_by_gender_df = group_by(weight_df, gender)
weight_by_gender_df
```

```
## # A tibble: 6,068 x 7
## # Groups:   gender [2]
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10027 male     178.   81.5 right         41 white
##  2     10032 male     170.   72.6 left          35 white
##  3     10033 male     174.   92.9 left          42 black
##  4     10092 male     166.   79.4 right         31 white
##  5     10093 male     191.   94.6 right         21 black
##  6     10115 male     172    80.2 right         39 white
##  7     10117 male     181   116.  right         32 black
##  8     10237 male     185    95.4 right         23 white
##  9     10242 male     178.   99.5 right         36 white
## 10     10244 male     181.   70.2 left          23 white
## # i 6,058 more rows
```

```r
# We see that there are two groups of genders. Use summarize to this grouped data frame to find average
summarize(weight_by_gender_df, mean = mean(age, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   gender  mean
##   <chr>  <dbl>
## 1 female  28.9
## 2 male    30.2
```

```
# calculate three sumamry stats for age variable of weight_by_gender_df using summary_at
summarize_at(weight_by_gender_df,
             vars(age),
             list(mean = ~mean(., na.rm = TRUE),
                  median = ~median(., na.rm = TRUE),
                  sd = ~sd(., na.rm = TRUE)
             )
)
```

```
## # A tibble: 2 x 4
##   gender  mean median    sd
##   <chr>  <dbl>  <dbl> <dbl>
## 1 female  28.9     27  8.33
## 2 male    30.2     28  8.81
```

```
# Use summarize and group_by together to create new reduced data
summarize(group_by(weight_df, race, age))
```

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

```
## # A tibble: 206 x 2
## # Groups:   race [7]
##    race    age
##    <chr> <dbl>
##  1 asian    17
##  2 asian    18
##  3 asian    19
##  4 asian    20
##  5 asian    21
##  6 asian    22
##  7 asian    23
##  8 asian    24
##  9 asian    25
## 10 asian    26
## # i 196 more rows
```

```
# To ungroup any gropued data, use ungroup() function
ungroup(weight_by_gender_df)
```

```
## # A tibble: 6,068 x 7
##    subjectid gender height weight handedness   age race
##        <dbl> <chr>   <dbl>  <dbl> <chr>      <dbl> <chr>
##  1     10027 male     178.   81.5 right         41 white
##  2     10032 male     170.   72.6 left          35 white
##  3     10033 male     174.   92.9 left          42 black
```

```
##  4     10092 male     166.   79.4 right         31 white
##  5     10093 male     191.   94.6 right         21 black
##  6     10115 male     172    80.2 right         39 white
##  7     10117 male     181   116.  right         32 black
##  8     10237 male     185    95.4 right         23 white
##  9     10242 male     178.   99.5 right         36 white
## 10     10244 male     181.   70.2 left          23 white
## # i 6,058 more rows
```

```
# Reshape weight_df using pivot_longer and pivot_wider.
# Use pivot_longer to create data frame where height and weight are listed in measurement column with c
long_weight_df = pivot_longer(weight_df,
                              cols = contains('eight'),
                              names_to = 'measurement',
                              values_to = 'value')
long_weight_df
```

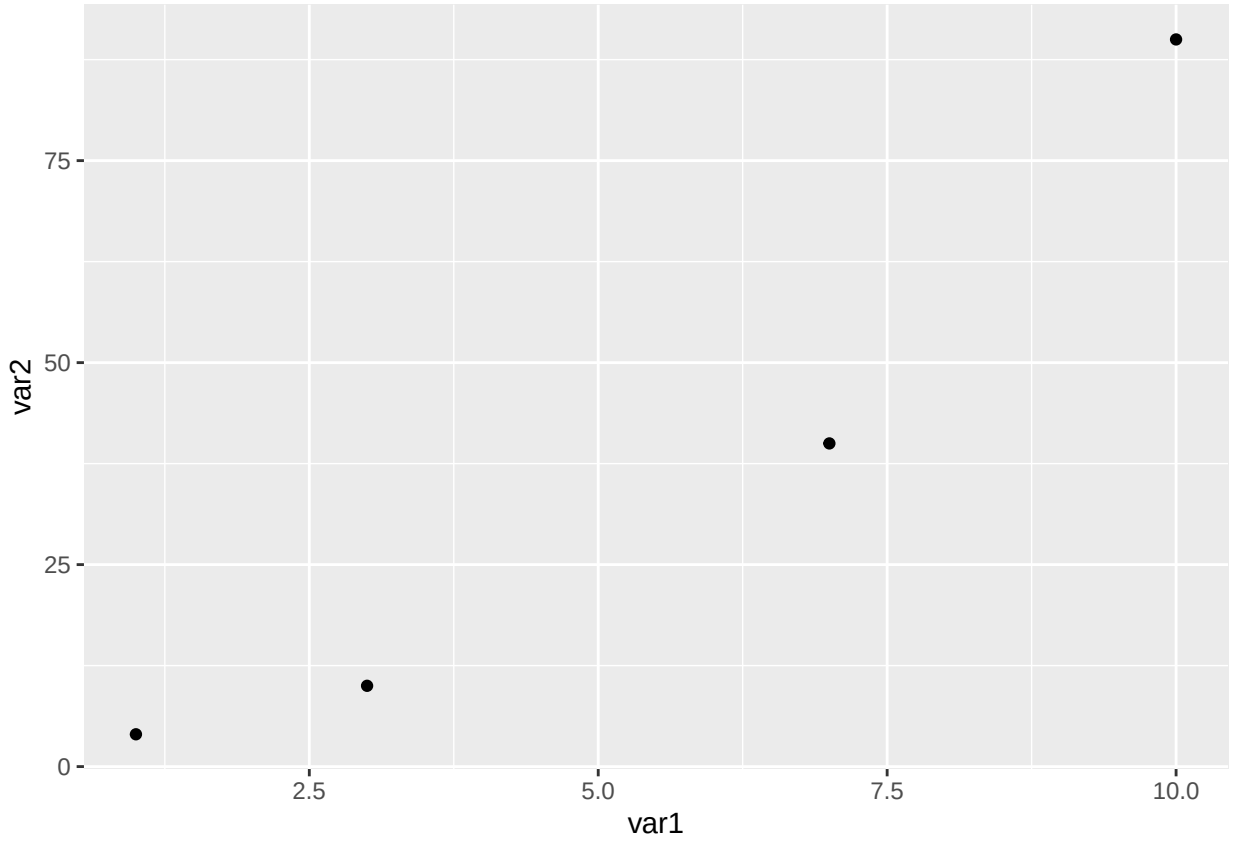
```
## # A tibble: 12,136 x 7
##    subjectid gender handedness   age race  measurement value
##        <dbl> <chr>  <chr>      <dbl> <chr> <chr>       <dbl>
##  1     10027 male   right         41 white height      178.
##  2     10027 male   right         41 white weight       81.5
##  3     10032 male   left          35 white height      170.
##  4     10032 male   left          35 white weight       72.6
##  5     10033 male   left          42 black height      174.
##  6     10033 male   left          42 black weight       92.9
##  7     10092 male   right         31 white height      166.
##  8     10092 male   right         31 white weight       79.4
##  9     10093 male   right         21 black height      191.
## 10     10093 male   right         21 black weight       94.6
## # i 12,126 more rows
```

```
# Let's use pivot_wider to get back to original data frame
wide_weight_df = pivot_wider(long_weight_df,
                             names_from = 'measurement',
                             values_from = 'value')
wide_weight_df
```

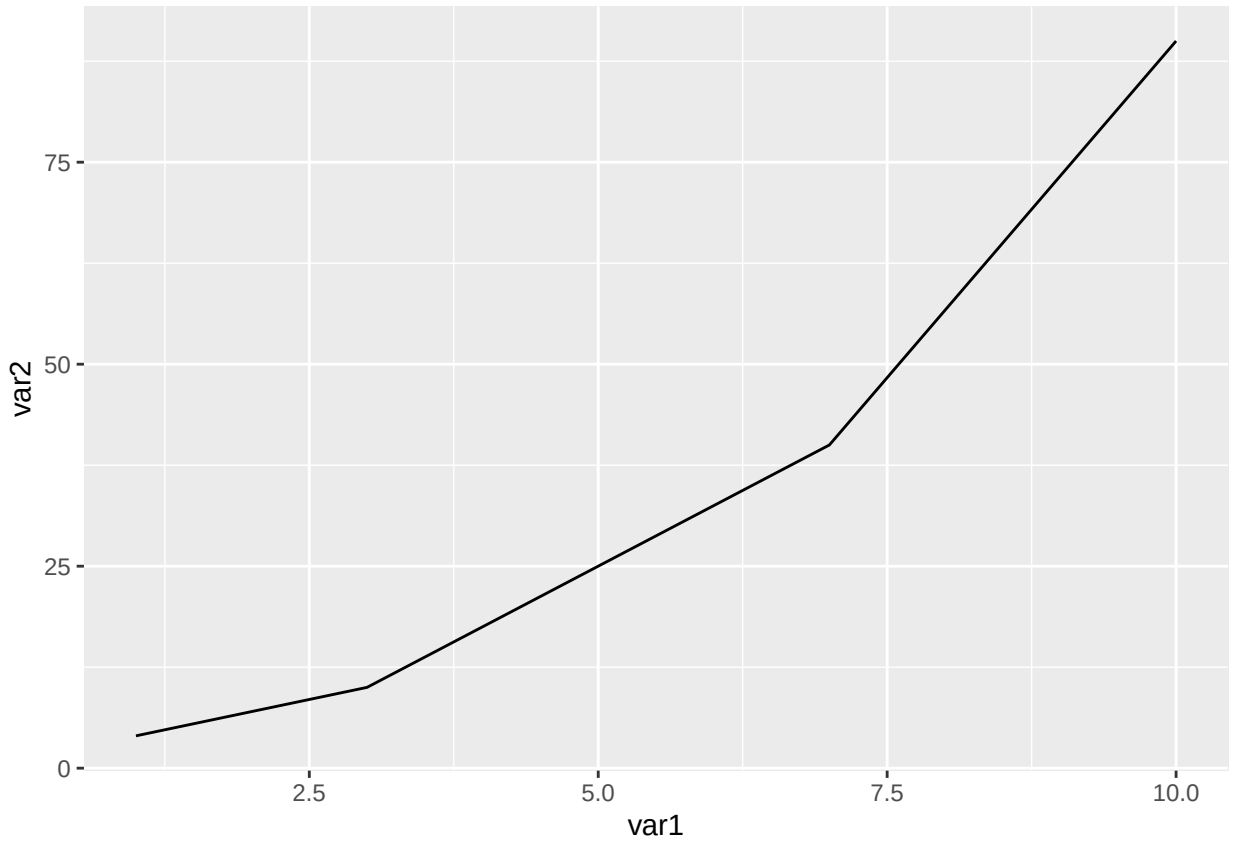
```
## # A tibble: 6,068 x 7
##    subjectid gender handedness   age race  height weight
##        <dbl> <chr>  <chr>      <dbl> <chr>  <dbl>  <dbl>
##  1     10027 male   right         41 white   178.   81.5
##  2     10032 male   left          35 white   170.   72.6
##  3     10033 male   left          42 black   174.   92.9
##  4     10092 male   right         31 white   166.   79.4
##  5     10093 male   right         21 black   191.   94.6
##  6     10115 male   right         39 white   172    80.2
##  7     10117 male   right         32 black   181   116.
##  8     10237 male   right         23 white   185    95.4
##  9     10242 male   right         36 white   178.   99.5
## 10     10244 male   left          23 white   181.   70.2
## # i 6,058 more rows
```

Chapter 4 HW: As with the previous chapter on data wrangling, a valuable exercise based on this chapter is for the reader to use their own data-sets to practice with all the plotting methods that are described in the chapter. It may be that different data sets may be required for different types of plots. See additional datasets below

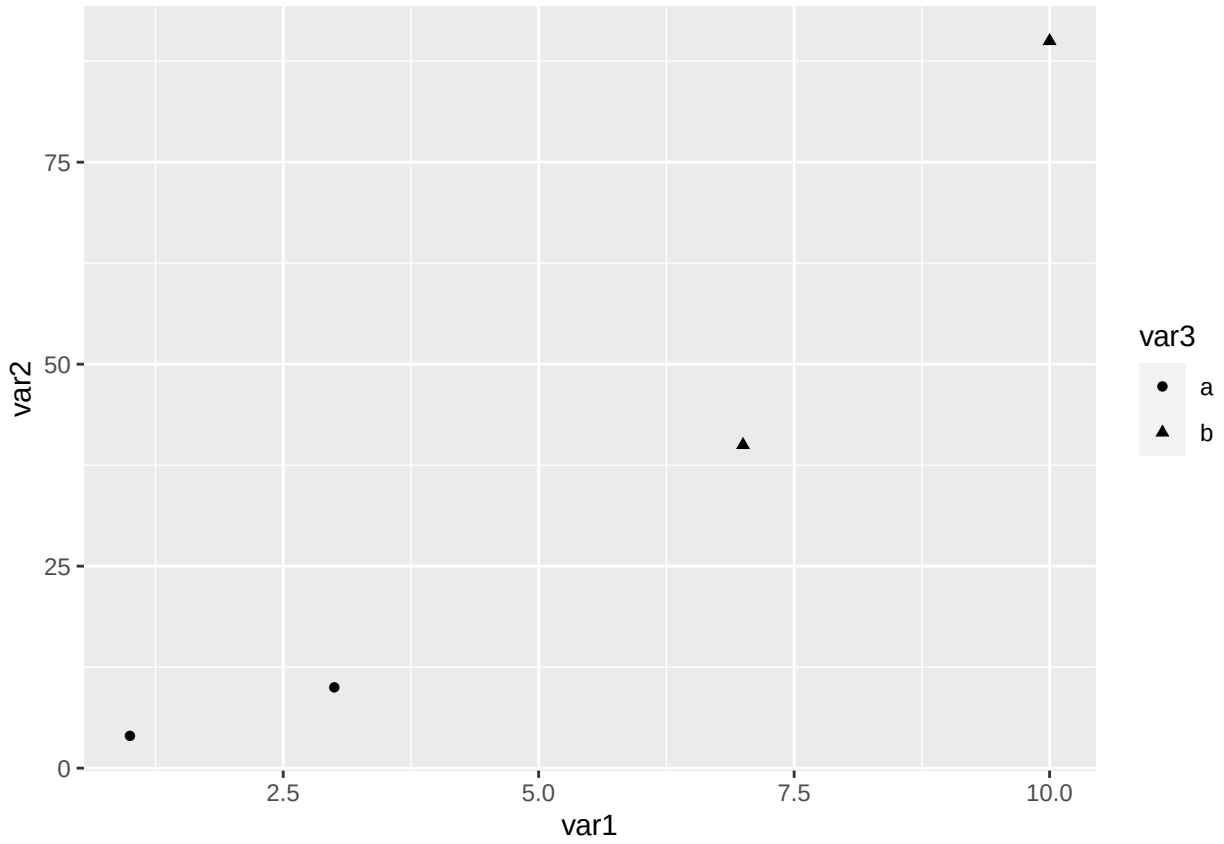
```
# Simple plotting exercies
simple_df = tribble(
  ~var1, ~var2, ~var3,
  1, 4, 'a',
  3, 10, 'a',
  7, 40, 'b',
  10, 90, 'b'
)

# Create scatter plot of simple df with x axis as var1 and y axis as var2
ggplot(simple_df,
       mapping = aes(x = var1, y = var2)) +
         geom_point()
```



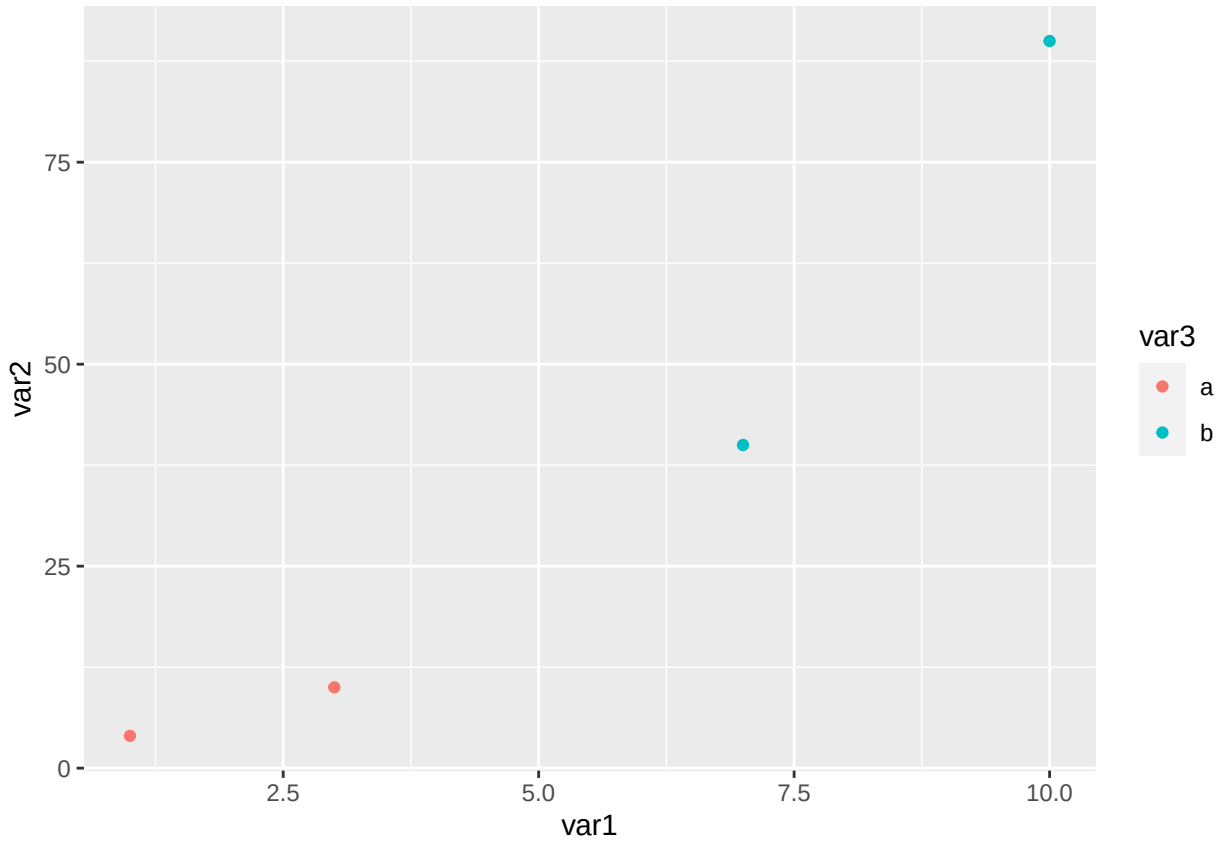

```
# Create line plot
ggplot(simple_df,
       mapping = aes(x = var1, y = var2)) +
  geom_line()
```




```
# Add additional mapping to scatter plot using var3, which is a discrete variable. Since it is discrete
ggplot(simple_df,
       mapping = aes(x = var1, y = var2, shape = var3)) +
  geom_point()
```

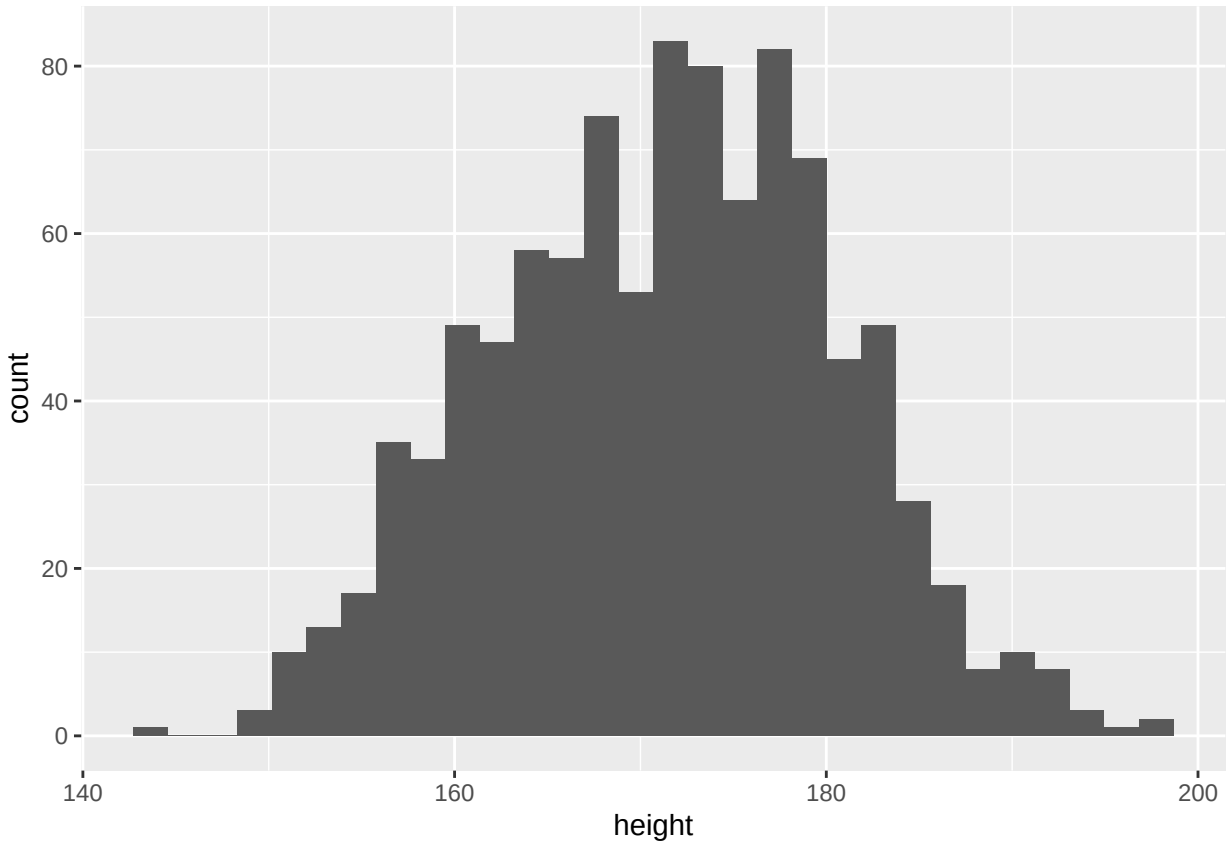



```
# We can represent var3 with colors using color = var3 instead
ggplot(simple_df,
       mapping = aes(x = var1, y = var2, color = var3)) +
  geom_point()
```

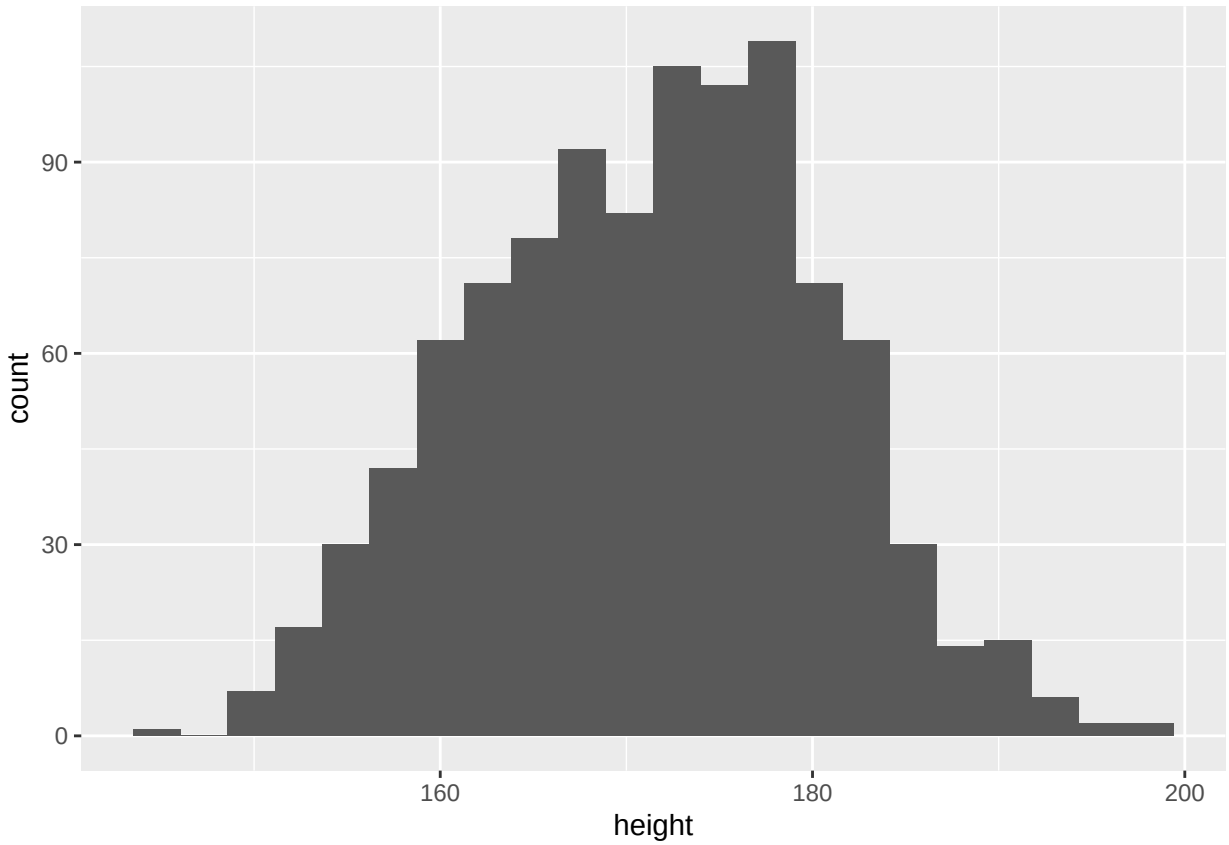



```
# Histograms
# Create histogram using weight_df. Subsample weight_df first to 1000 observations
weight_df_1 = weight_df %>%
  sample_n(1000)
ggplot(weight_df_1,
       mapping = aes(x = height)) +
  geom_histogram()
```

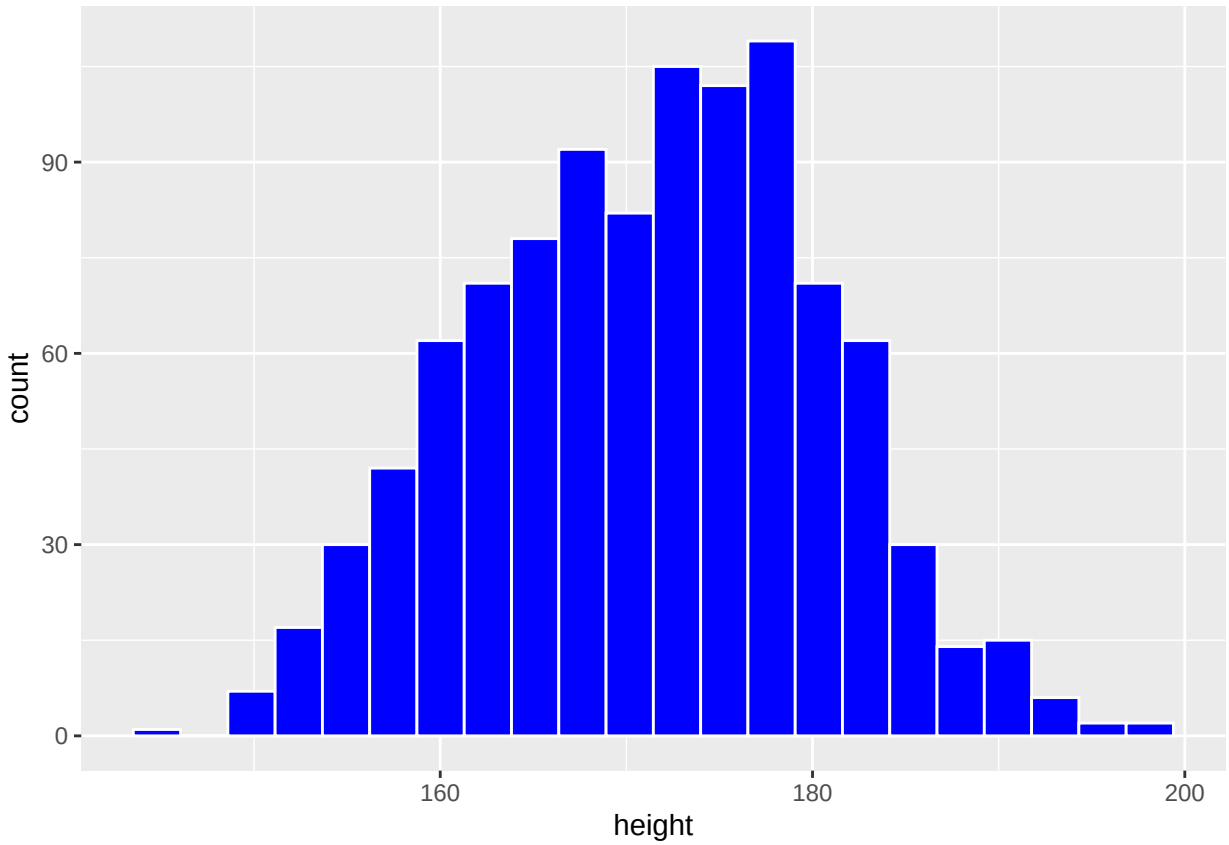
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



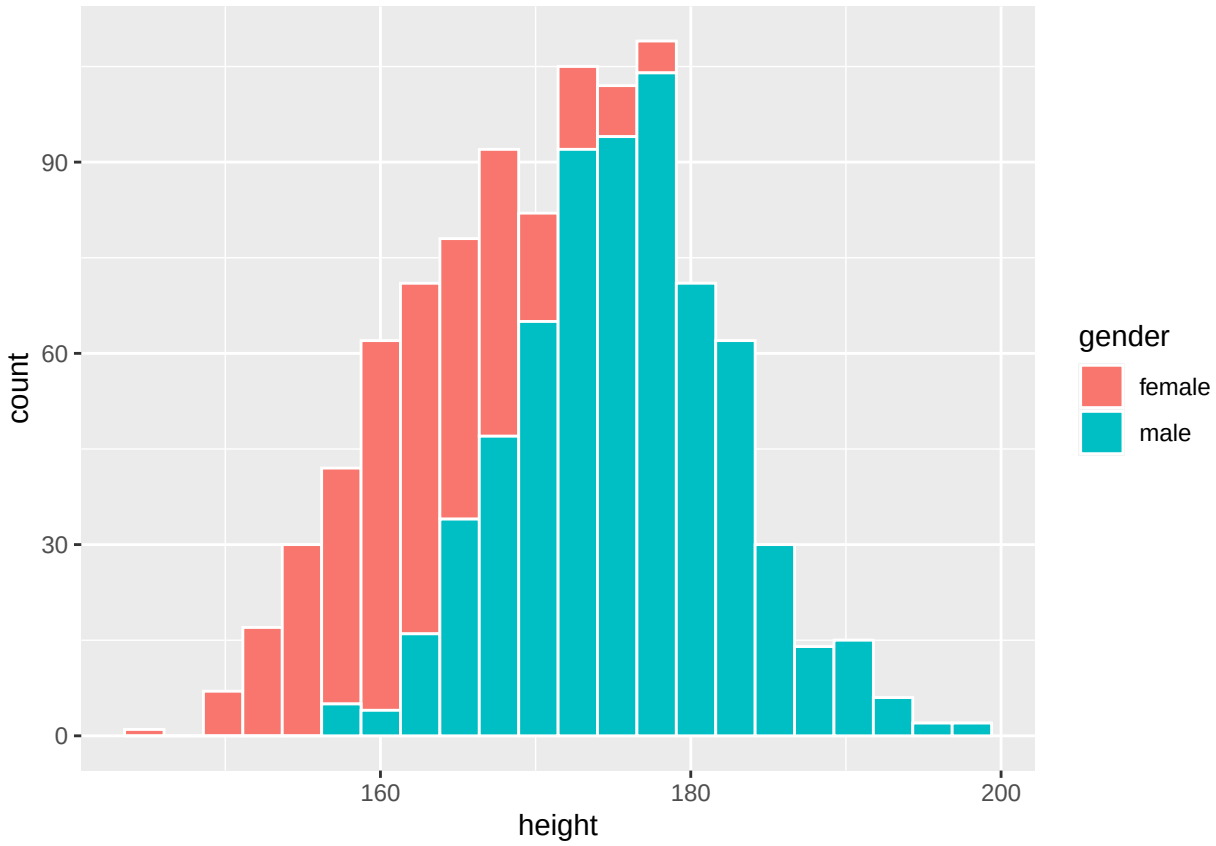

```
# Use binwidth = 2.54 so we can specify width of bin
ggplot(weight_df_1,
       mapping = aes(x = height)) +
  geom_histogram(binwidth = 2.54)
```




```
# Add interior color using fill = '' and add border using color = ''
ggplot(weight_df_1,
       mapping = aes(x = height)) +
  geom_histogram(binwidth = 2.54, color = 'white', fill = 'blue')
```

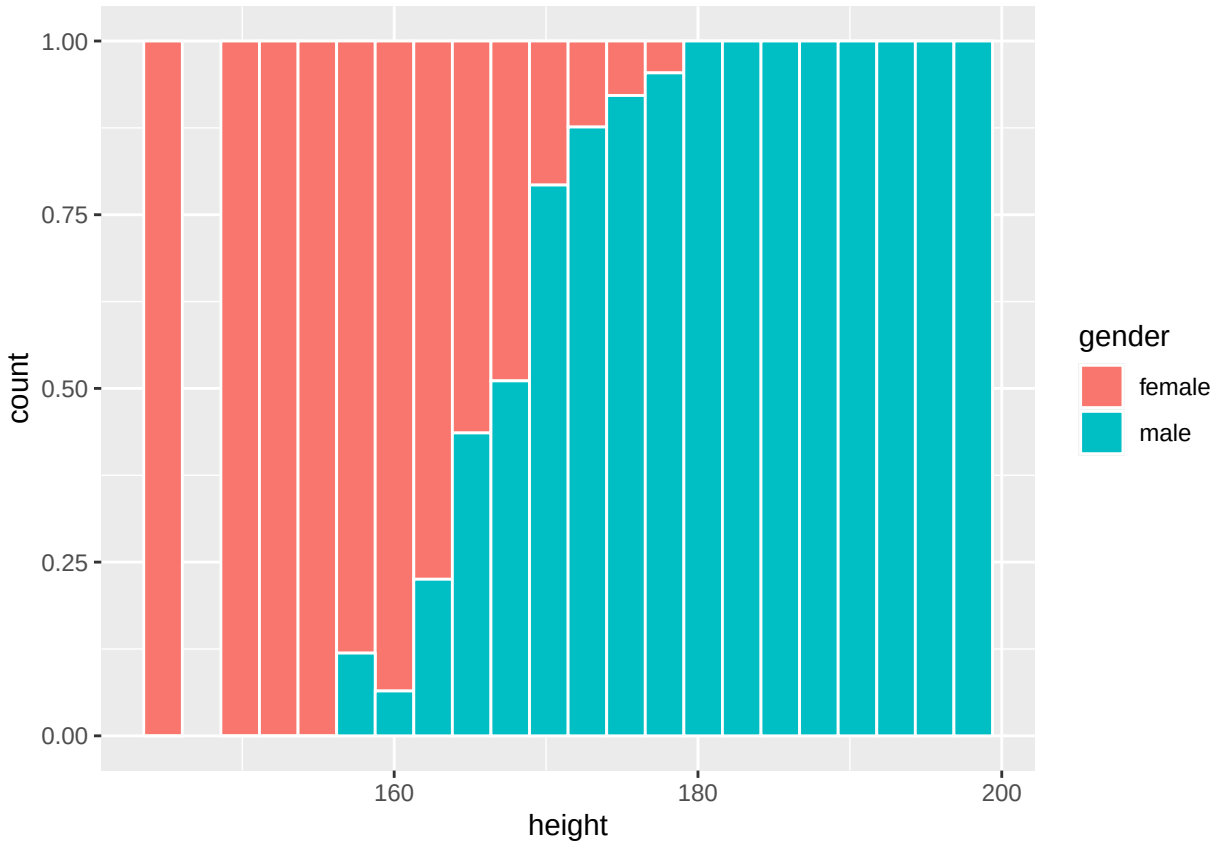



```
# If we speciy that either colour or fill (or maybe both) should be a discrete variable, we obtain stac
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_histogram(binwidth = 2.54, color = 'white')
```

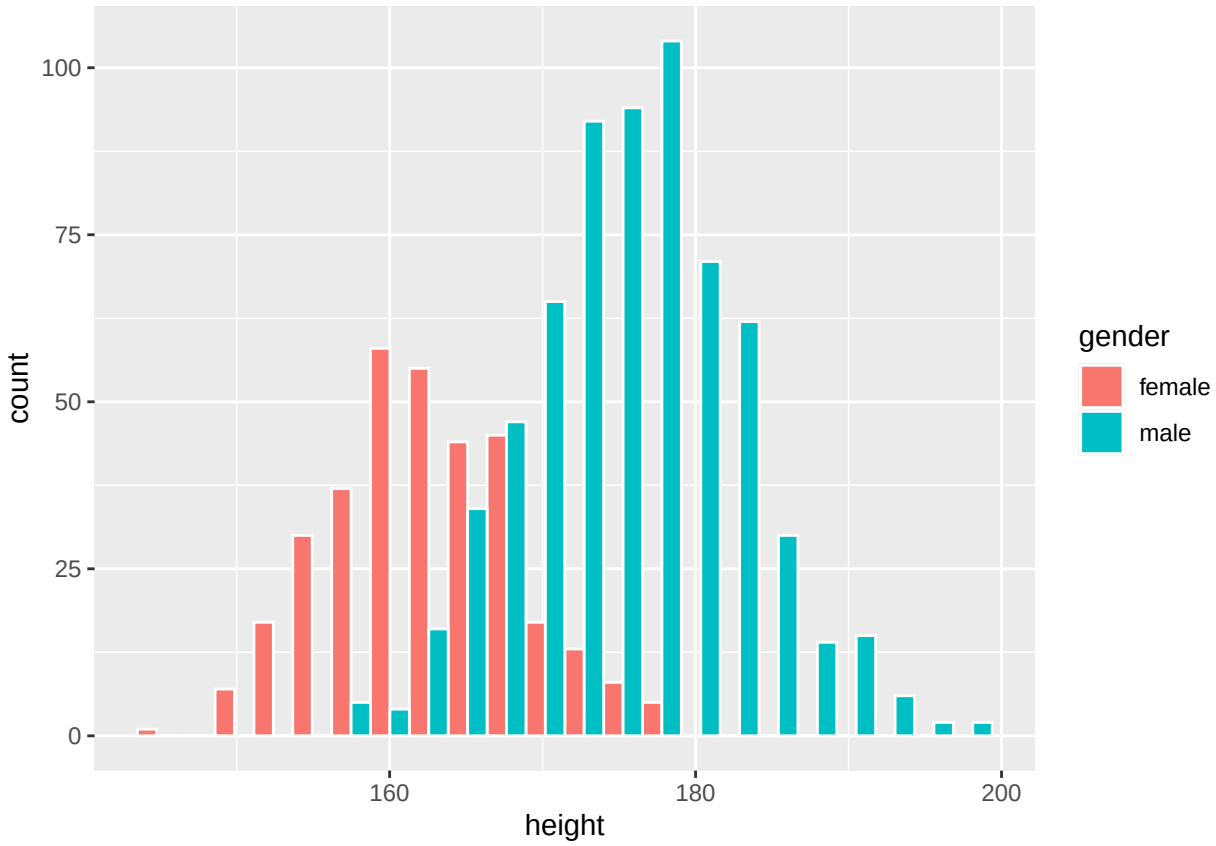



```
# In stacked histogram, each bin codes the proportion of that bin's values that correspond to each valu
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_histogram(binwidth = 2.54, color = 'white', position = 'fill')
```

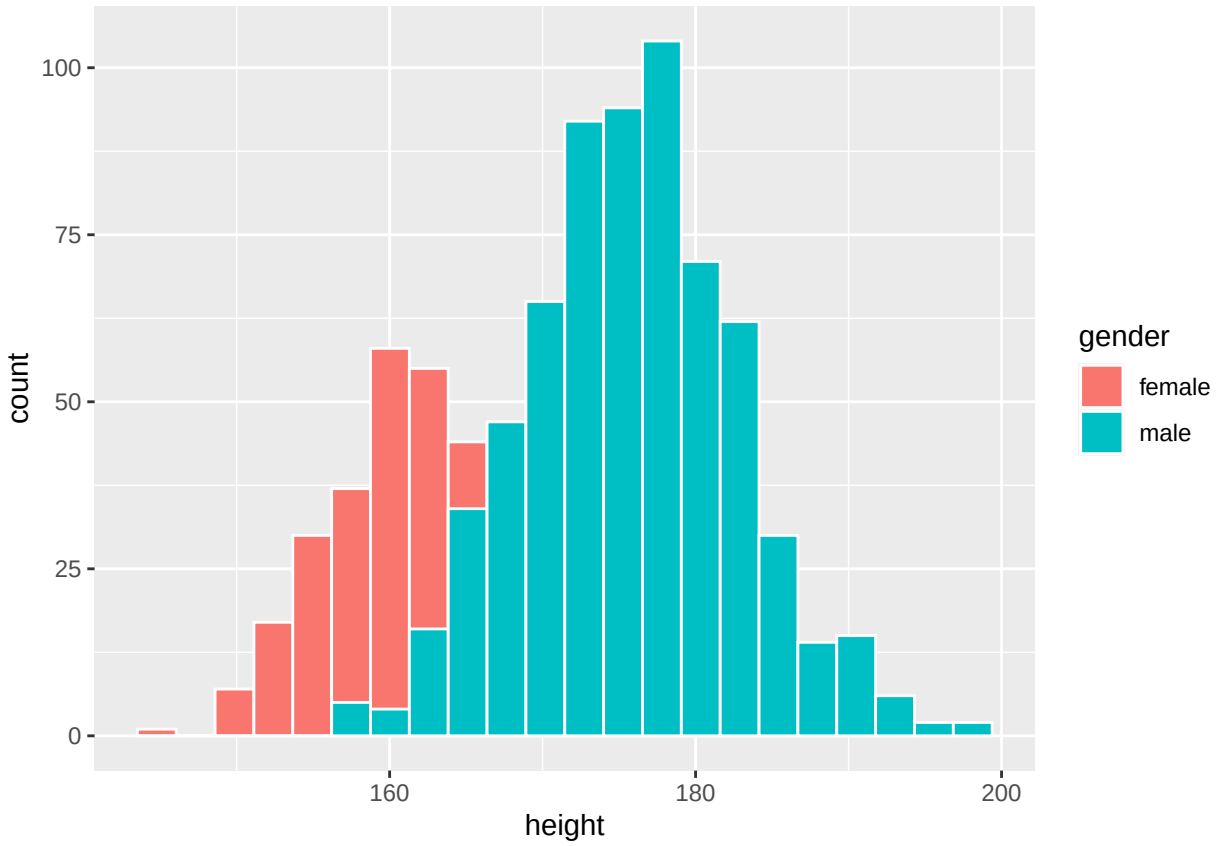
```
## Warning: Removed 2 rows containing missing values (‘geom_bar()‘).
```



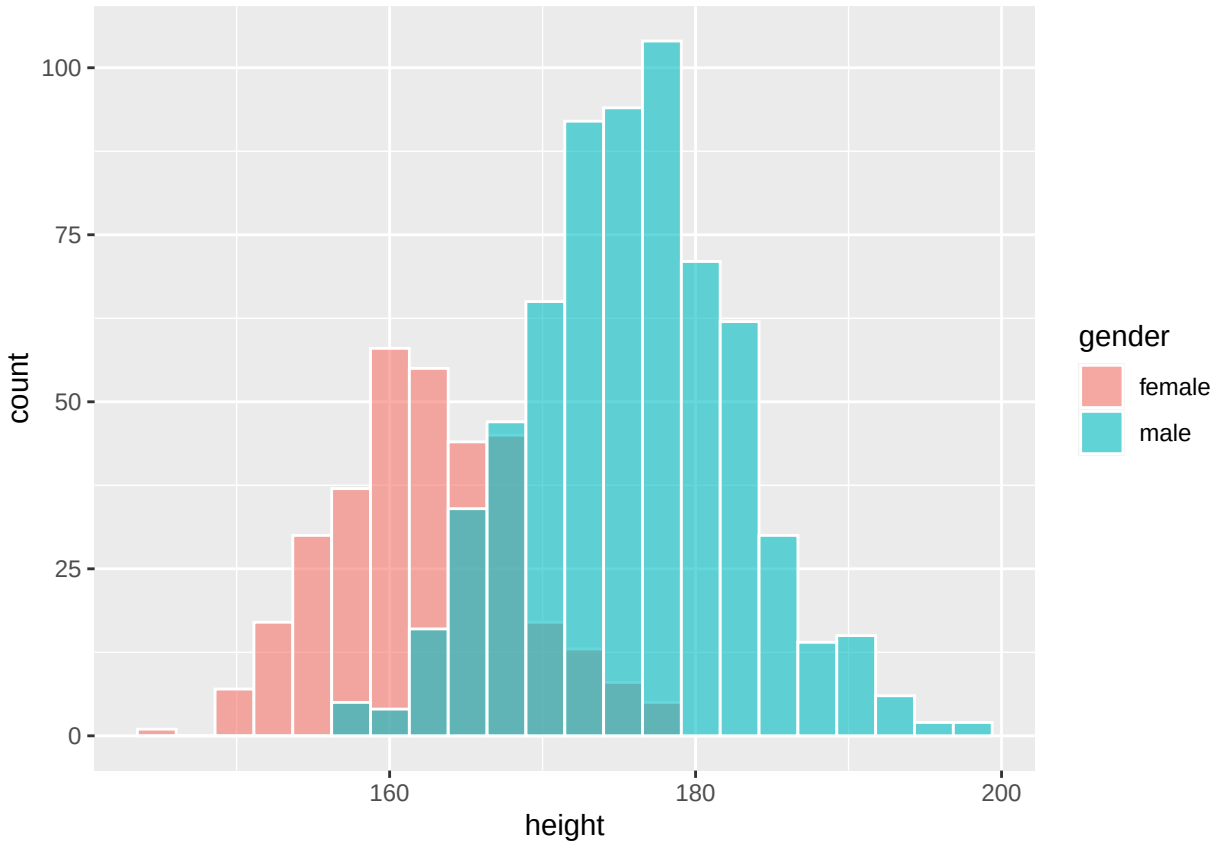

```
# We can create two seperate histograms on the same graph by using position = 'dodge' in geom_histogram
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_histogram(binwidth = 2.54, color = 'white',position = 'dodge')
```



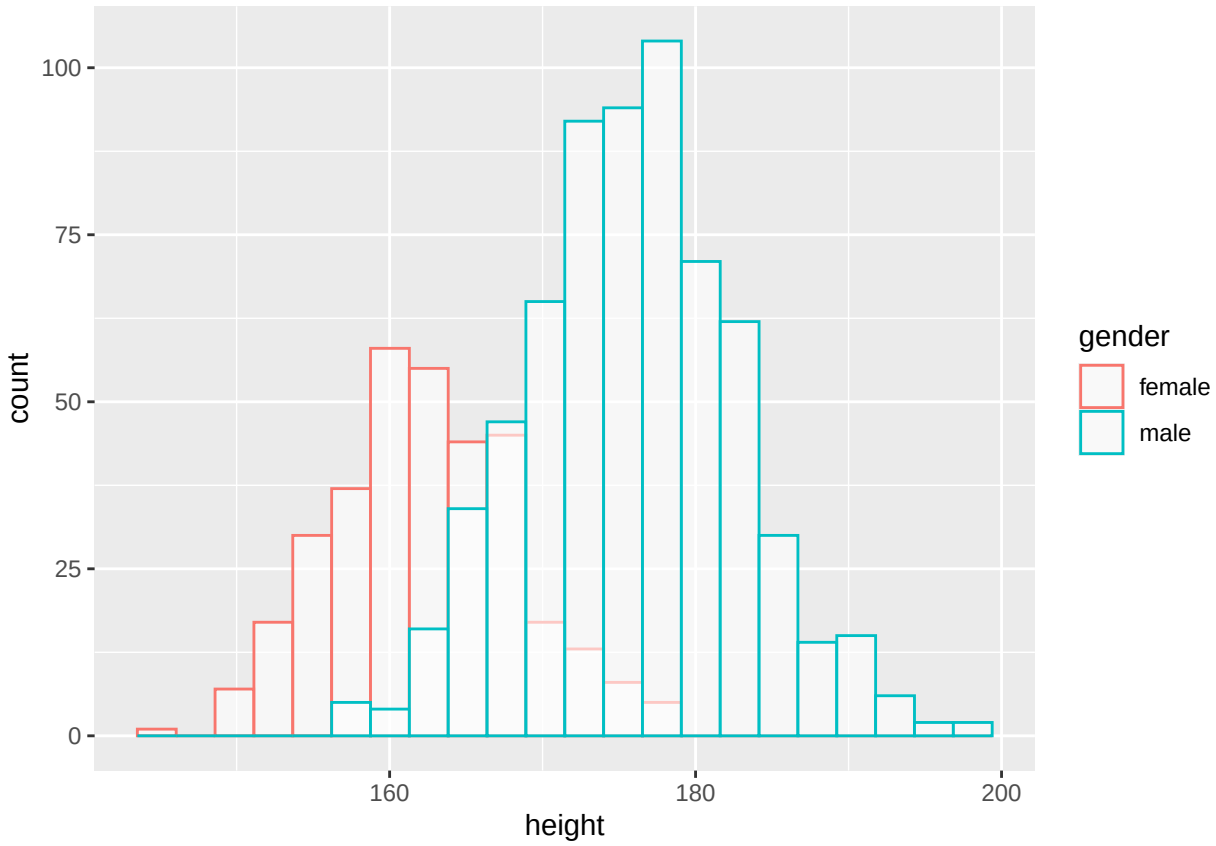

```
# Place bars corresponding to males and females at exact same location using position = 'identity'
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_histogram(binwidth = 2.54, color = 'white',position = 'identity')
```



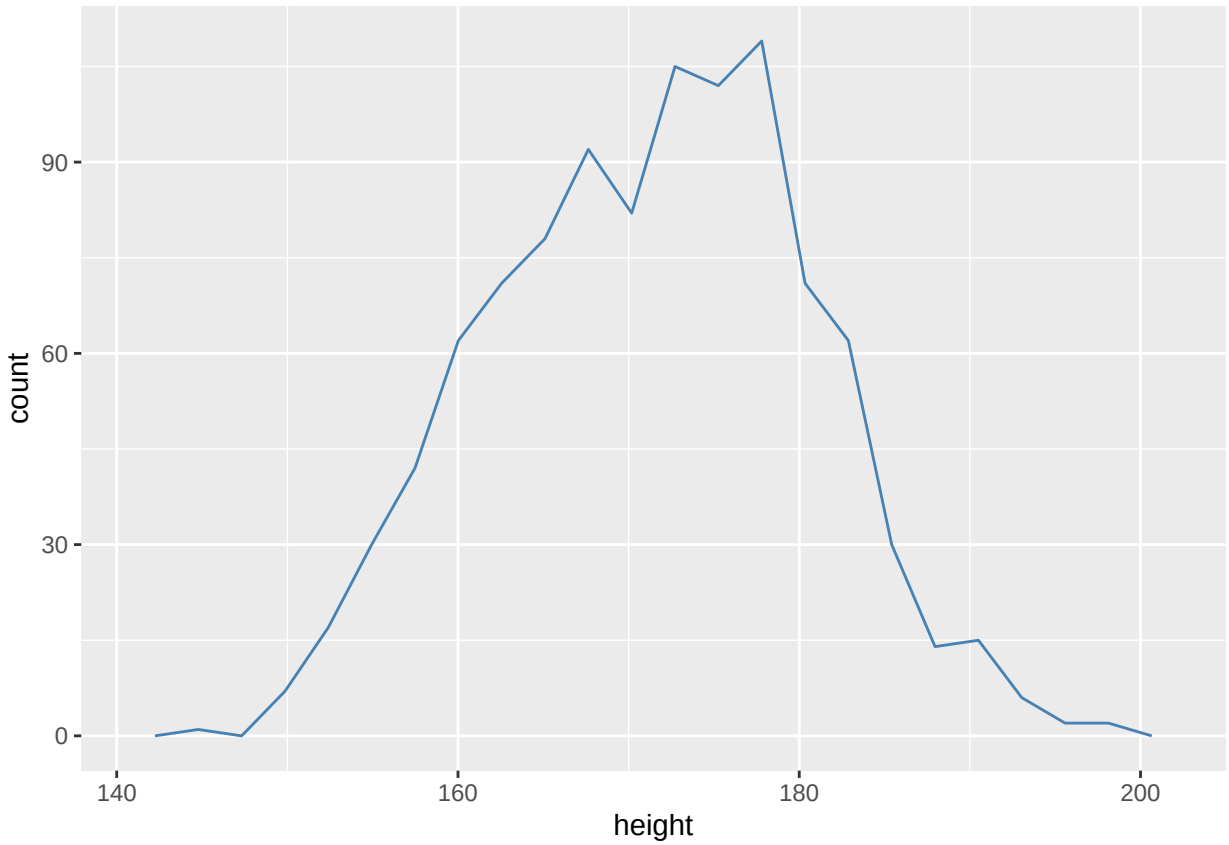

```
# To make it so that we can see hidden female bars behind the male bars (occlustion), use alpha = 0.75
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_histogram(binwidth = 2.54, color = 'white',position = 'identity', alpha = 0.60)
```



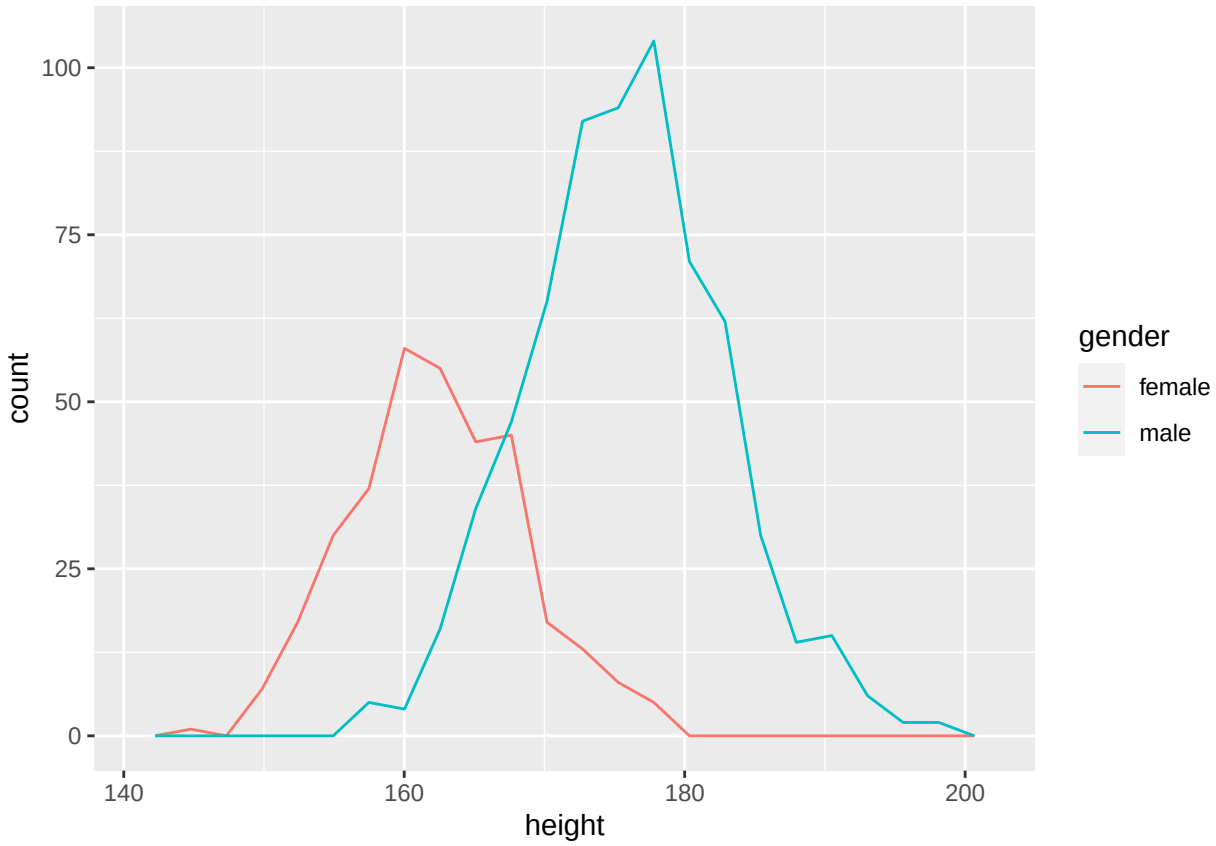

```
# Another way to avoide occlusion is to map gender to color rather than fill, and set fill to white
ggplot(weight_df_1,
       mapping = aes(x = height, color = gender)) +
  geom_histogram(binwidth = 2.54, fill = 'white', position = 'identity', alpha = 0.60)
```



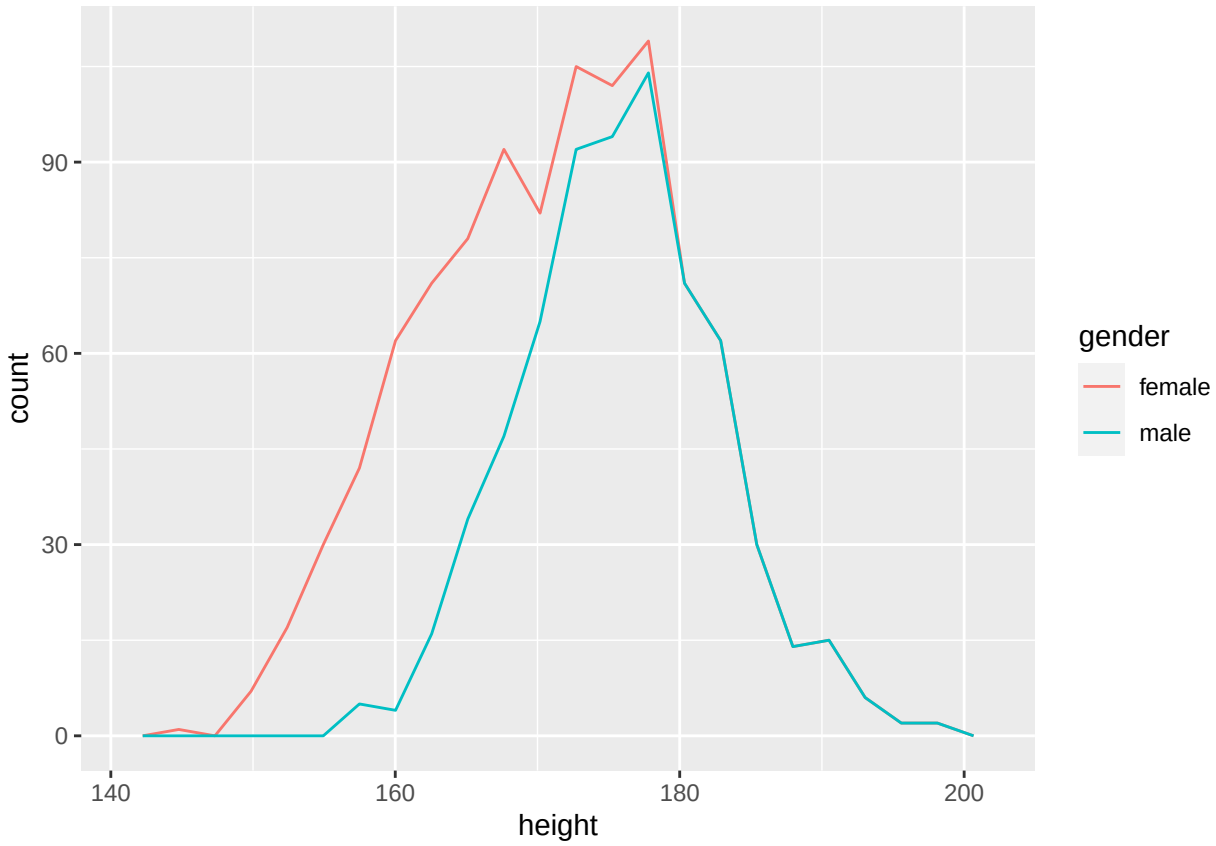

```
# Frequency Polygons
# Create frequency polygon instead of histogram
ggplot(weight_df_1,
       mapping = aes(x = height)) +
  geom_freqpoly(binwidth = 2.54, color = 'steelblue')
```



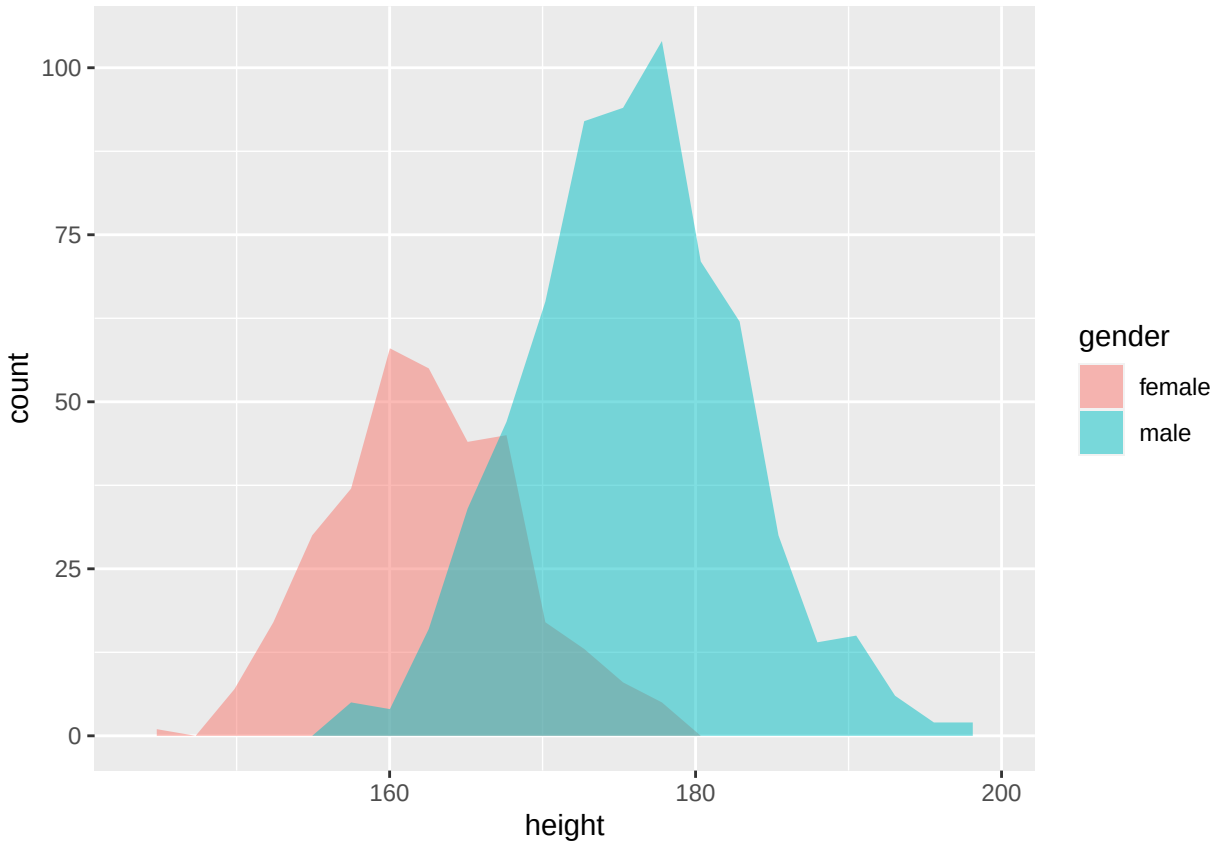

```
# Map gender to color to product two overlaid lines. Equivalent of the histogram using position = 'iden
ggplot(weight_df_1,
       mapping = aes(x = height, color = gender)) +
  geom_freqpoly(binwidth = 2.54)
```



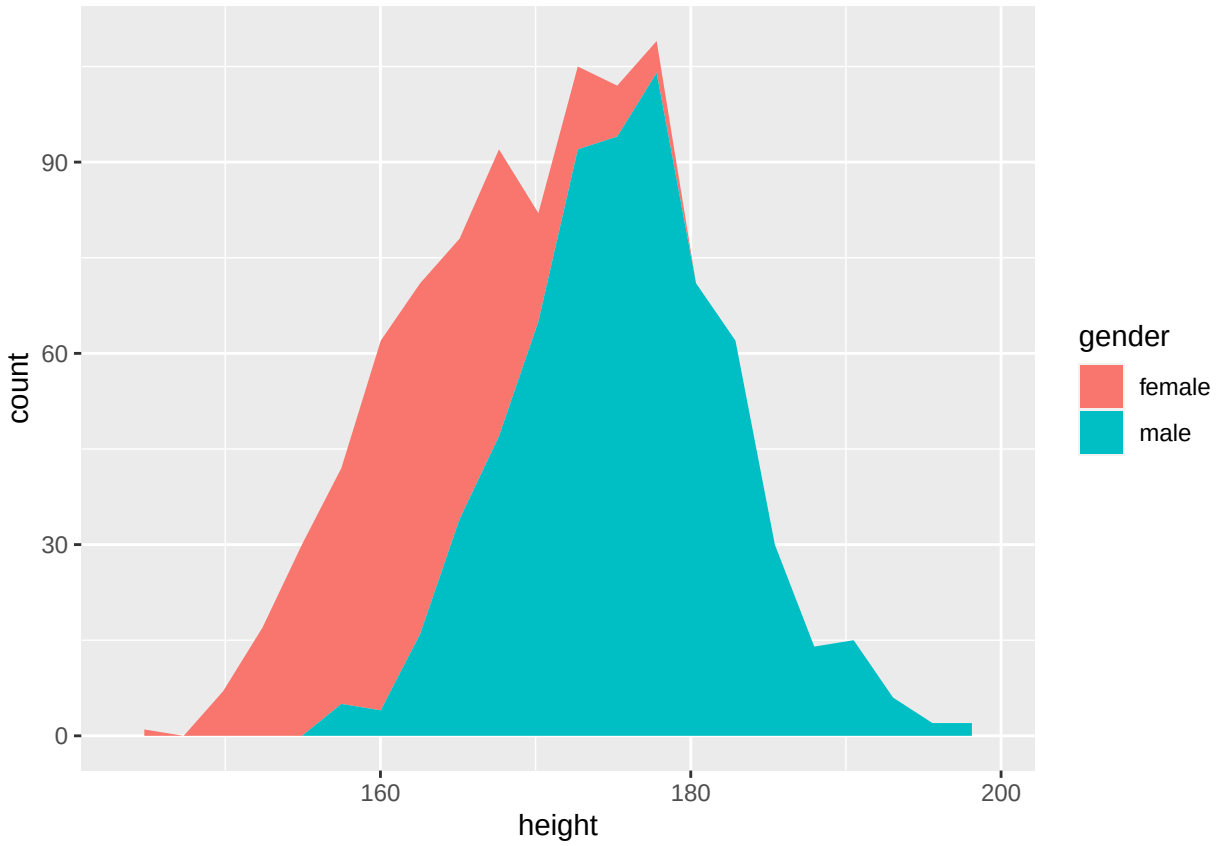

```
# Stack frequency polygon using position = 'stack'
ggplot(weight_df_1,
       mapping = aes(x = height, color = gender)) +
  geom_freqpoly(binwidth = 2.54, position = 'stack')
```



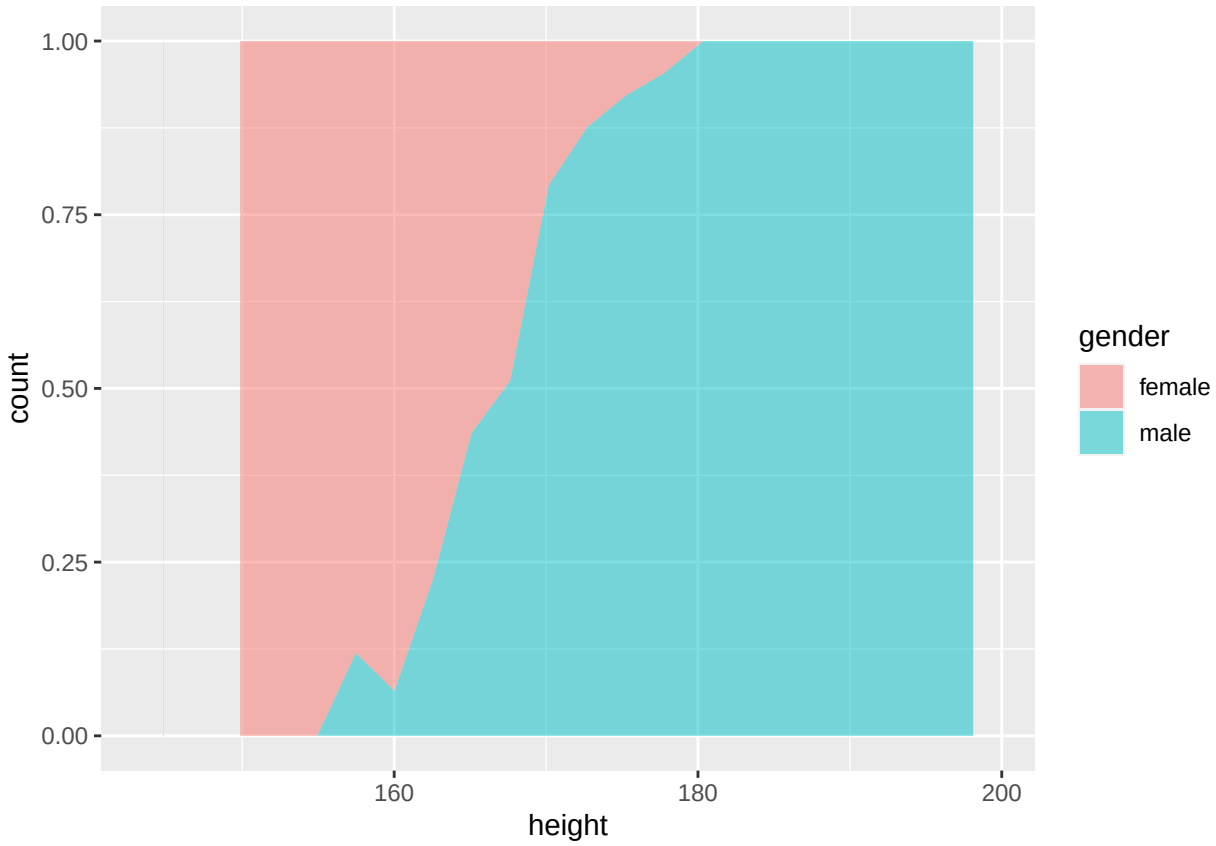

```
# Use geom_area() to create area plot which is filled in frequency polygon
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_area(binwidth = 2.54, position = 'identity', stat = 'bin', alpha = 0.5)
```



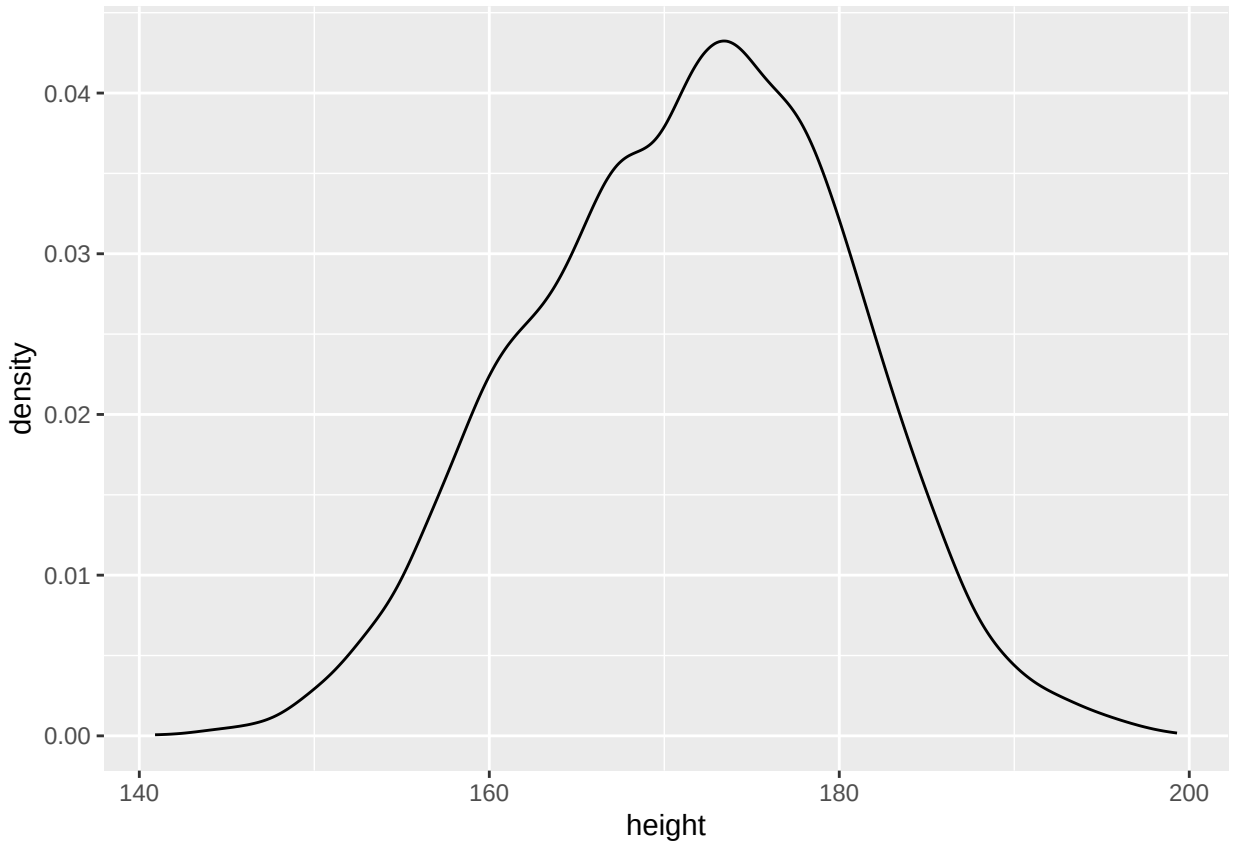

```
# Set position = 'stack' to obtain stacked area plot
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_area(binwidth = 2.54, position = 'stack', stat = 'bin')
```



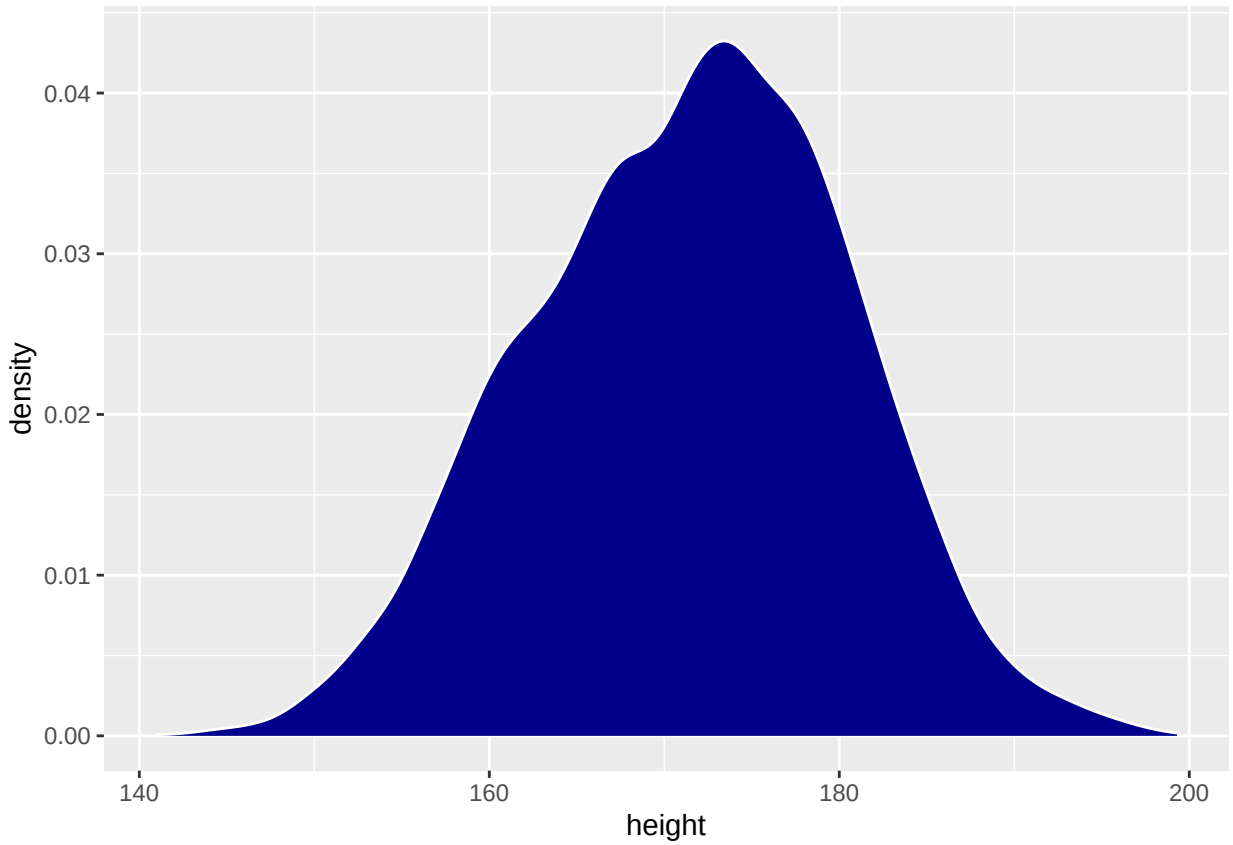

```
# We can also set position = 'fill'
ggplot(weight_df_1,
       mapping = aes(x = height, fill = gender)) +
  geom_area(binwidth = 2.54, position = 'fill', stat = 'bin', alpha = 0.5)
```



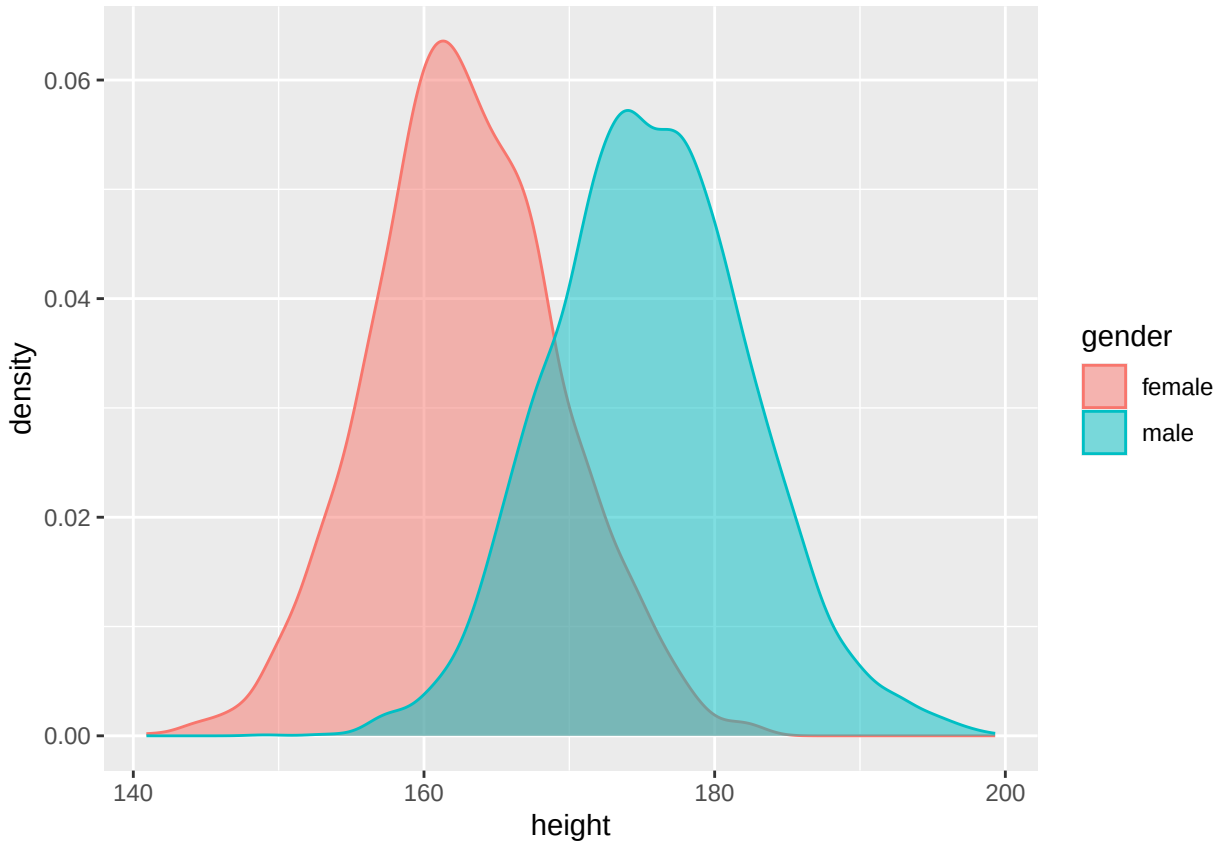

```
# Density Plots
# Density plots use kernel density estimation (KDE) to estimate probability density over the variable.
ggplot(weight_df,
       mapping = aes(x = height)) +
  geom_density()
```



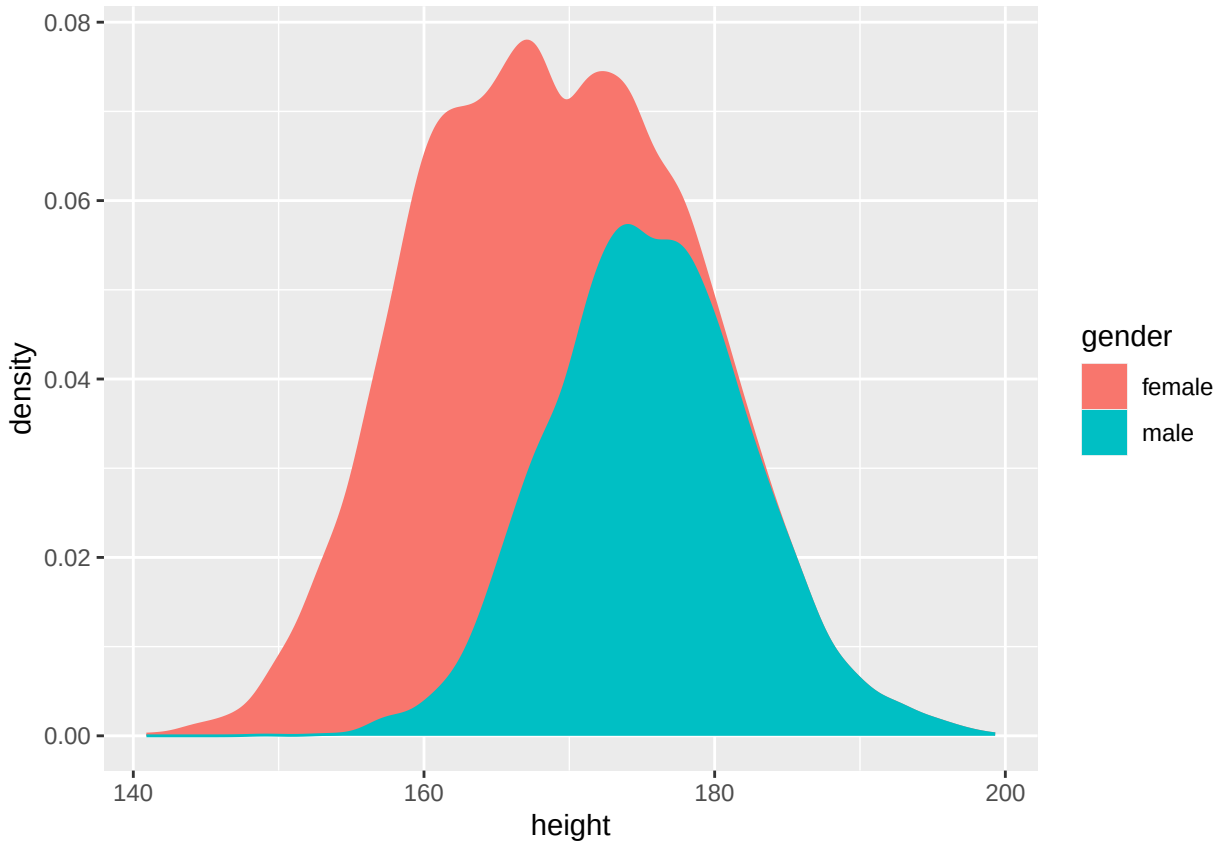

```
# We can change color of border with color = and interior with fill =
ggplot(weight_df,
       mapping = aes(x = height)) +
  geom_density(color = 'white', fill = 'darkblue')
```



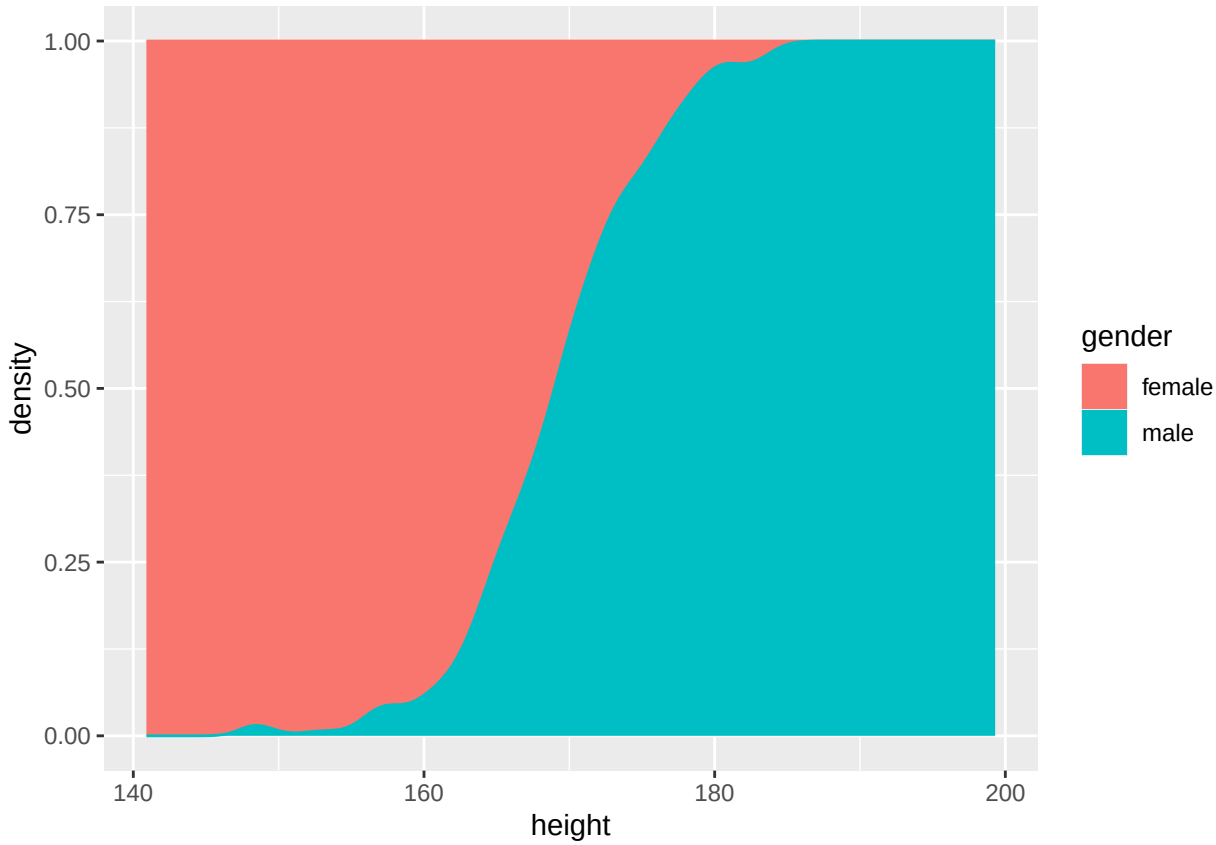

```
# Set both fill and color to gender to plot both denisty plots for each gender.
ggplot(weight_df,
       mapping = aes(x = height, color = gender, fill = gender)) +
  geom_density(alpha = 0.5)
```




```
# Use alpha = 0.5 to avoid occulusion bewteen each plot

# Set position = 'stack' to stack density plots
ggplot(weight_df,
       mapping = aes(x = height, color = gender, fill = gender)) +
  geom_density(position = 'stack')
```



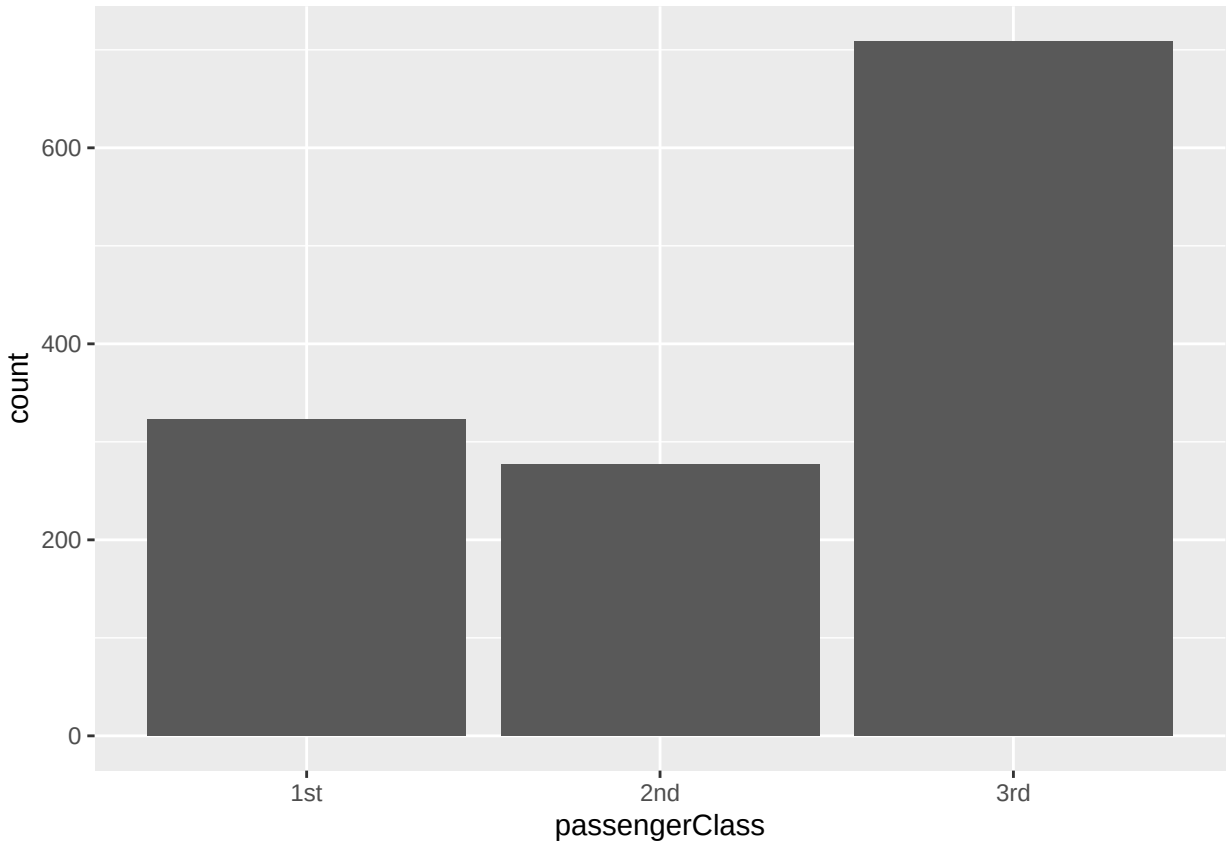

```
# To fill the stacked density plot, set position to 'fill'
ggplot(weight_df,
       mapping = aes(x = height, color = gender, fill = gender)) +
  geom_density(position = 'fill')
```



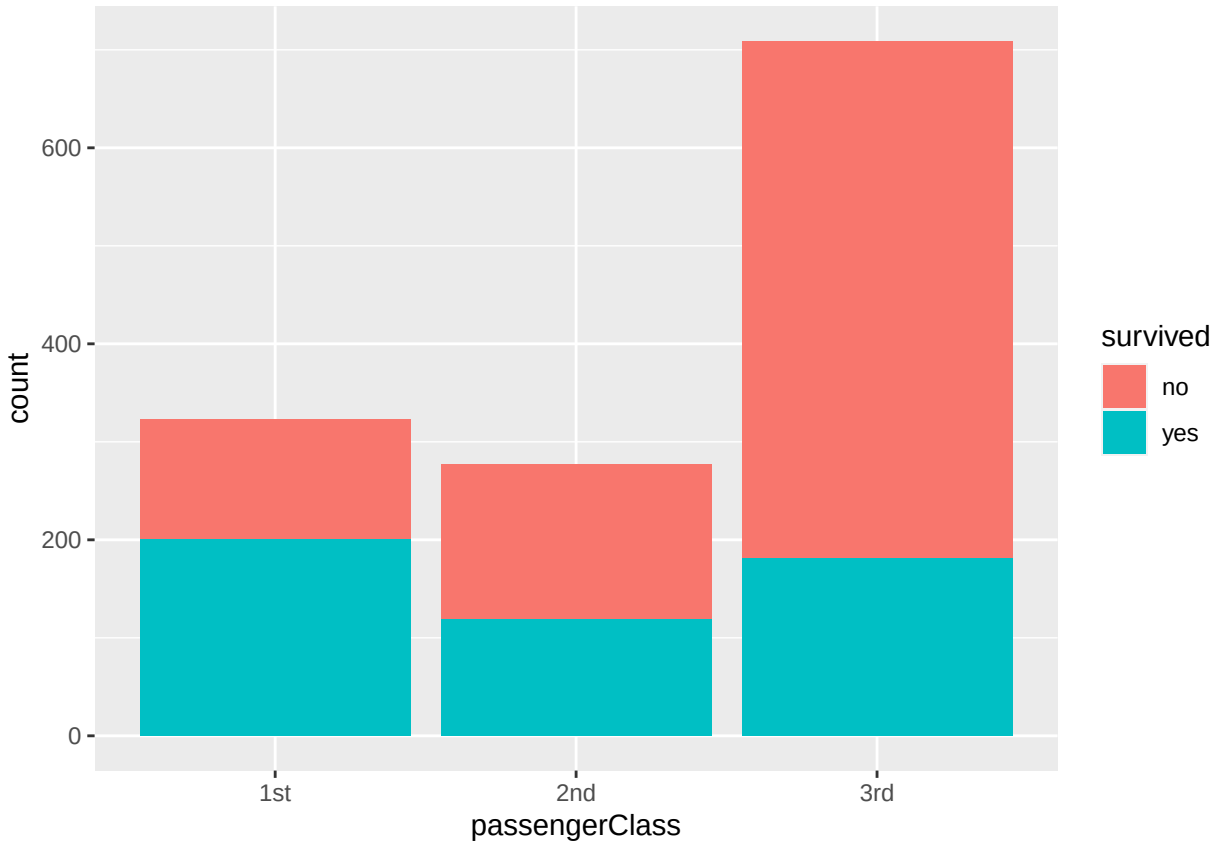

```
# Barplots
# Use barplots for discrete variables. Histograms and Density plots are meant for continuous variables
# Read in TitanticSurvival.csv
titanic_df = read_csv('week_2/data/TitanicSurvival.csv')
```

```
## New names:
## Rows: 1309 Columns: 5
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (4): ...1, survived, sex, passengerClass dbl (1): age
## 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.
## * `` -> `...1`
```

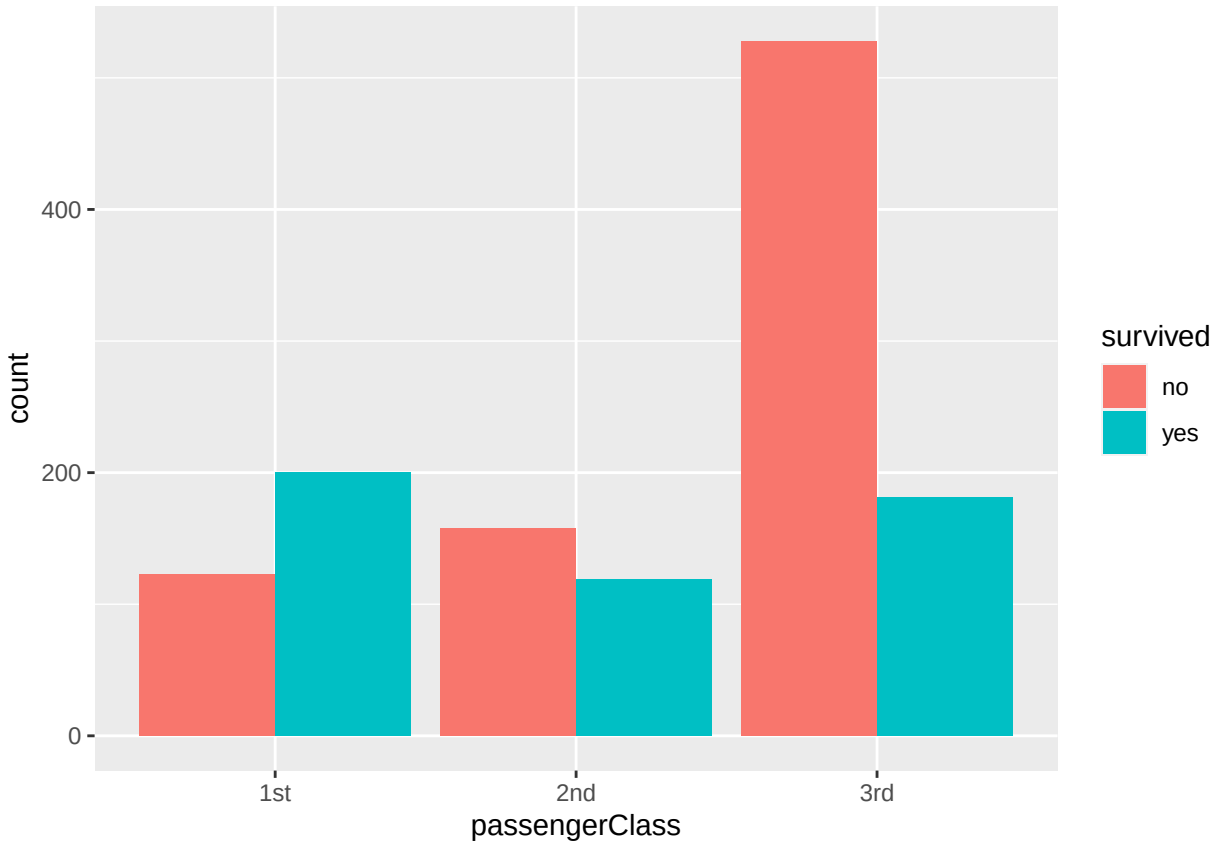
```
# Create barplot of each passenger in each passenger class
ggplot(titanic_df,
       mapping = aes(x = passengerClass)) +
  geom_bar()
```




```
# Use fill property to plot the variable survived in dataframe
ggplot(titanic_df,
       mapping = aes(x = passengerClass, fill = survived)) +
  geom_bar()
```



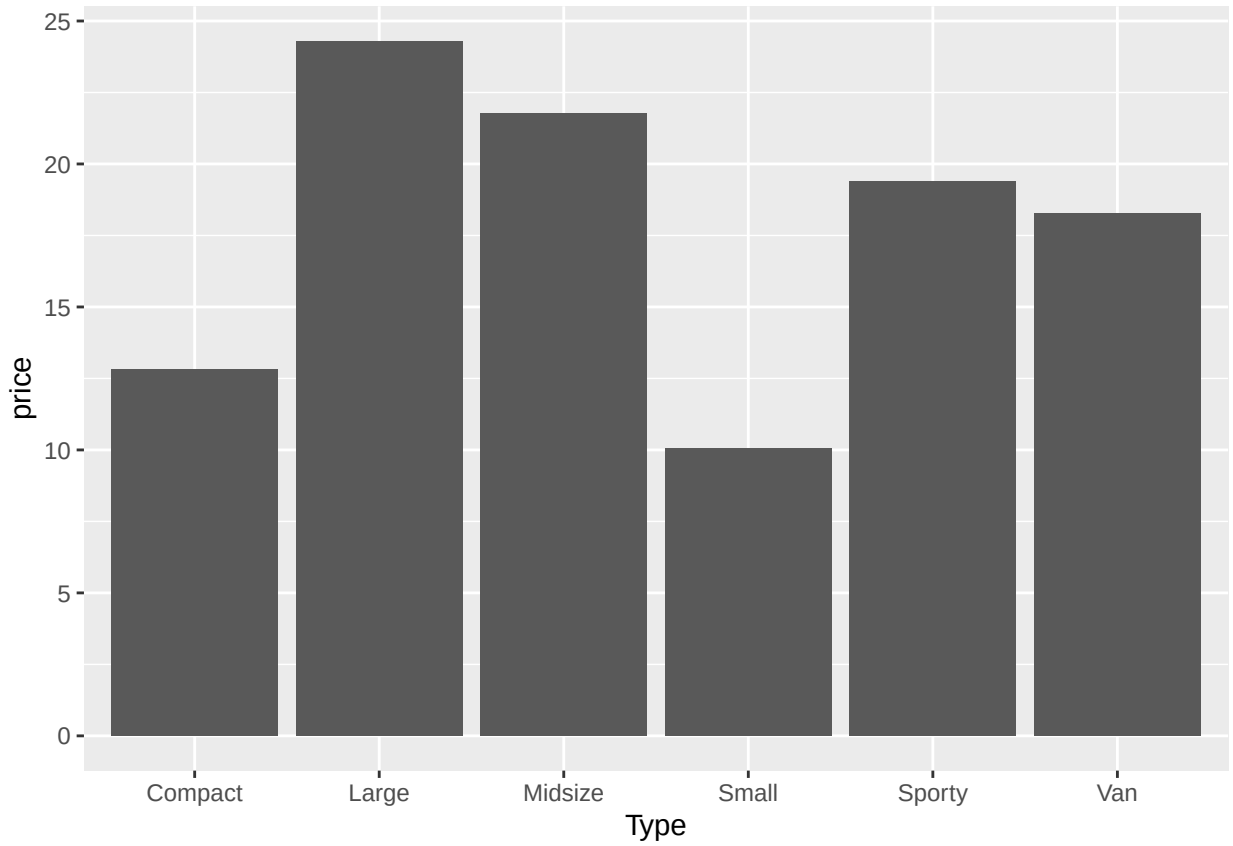

```
# Use position = 'dodge' in geom_bar() to put bars next to each other
ggplot(titanic_df,
       mapping = aes(x = passengerClass, fill = survived)) +
  geom_bar(position = 'dodge')
```




```
# Create barplot where the heights of the bars are given by the value of variable in data frame
# Read in carprice.csv
car_prices_df = read_csv('week_2/data/carprice.csv')
```

```
## New names:
## Rows: 48 Columns: 10
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Type dbl (9): ...1, Min.Price, Price, Max.Price, Range.Price, RoughRange,
## gpm100,...
## 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.
## * `` -> `...1`
```

```
# Let's group by Type variable then summarize by average Price variable
car_prices_avg_price = car_prices_df %>%
  group_by(Type) %>%
  summarize(price = mean(Price))

# Plot barplot where x axis is Type, y axis is price
ggplot(car_prices_avg_price,
       mapping = aes(x = Type, y = price)) +
  geom_bar(stat = 'identity')
```




```
# stat = 'identity' tells the plot that the height of the bars to be actual value of price from data fr

# You can display two discrete variables by mapping the fill variable
# Read in FatRats.csv
fat_rats_df = read_csv('week_2/data/FatRats.csv')
```

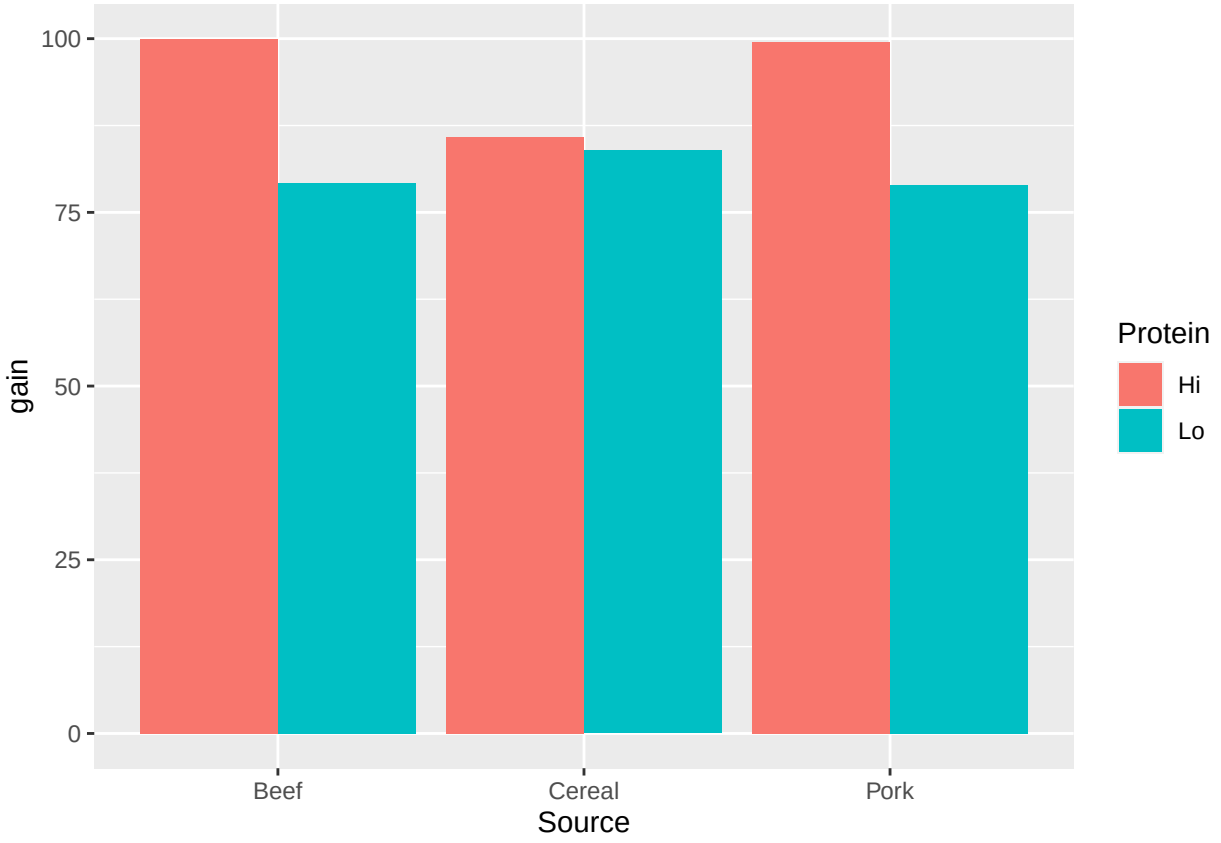
```
## New names:
## Rows: 60 Columns: 4
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): Protein, Source dbl (2): ...1, Gain
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
# Use group_by and summarize() to create smaller data frame
fat_rats_sub_df = fat_rats_df %>%
  group_by(Protein, Source) %>%
  summarize(gain = mean(Gain), se = sd(Gain) / sqrt(n()))
```

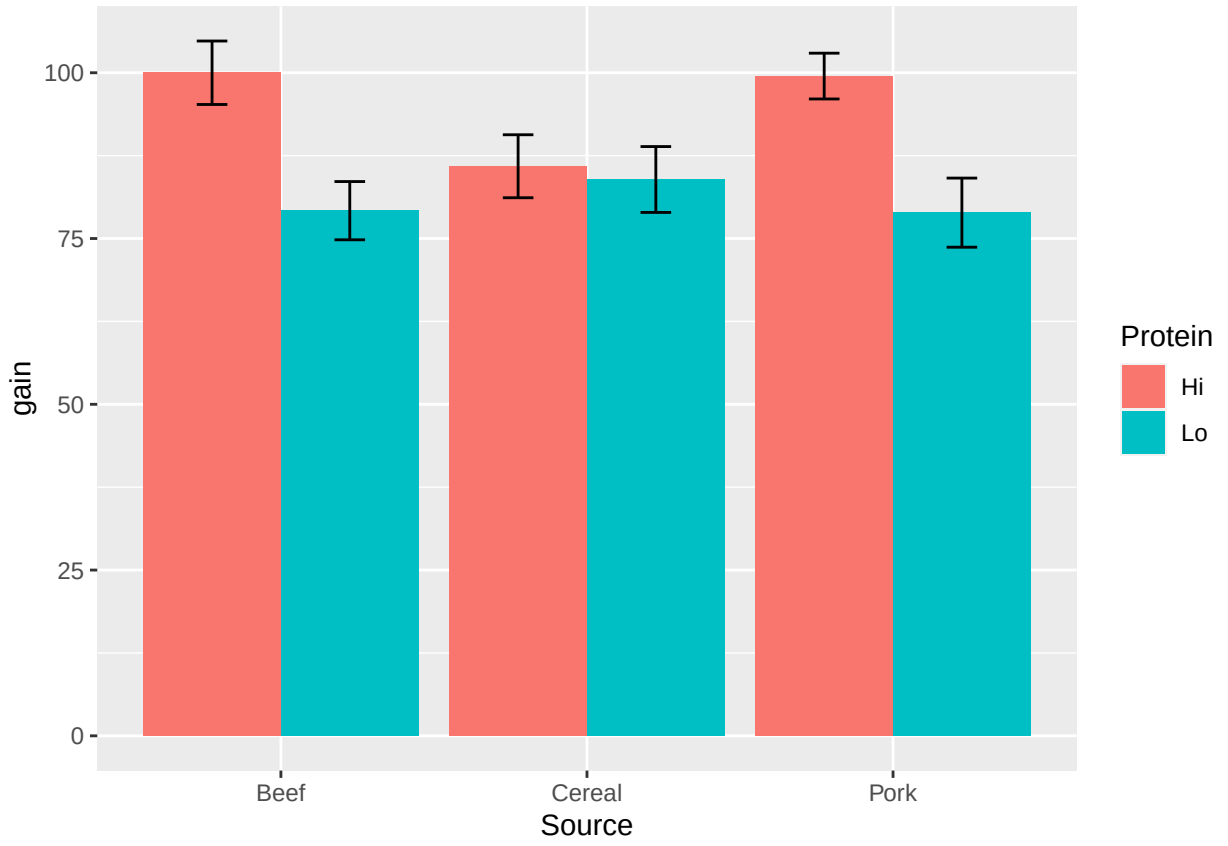
```
## `summarise()` has grouped output by 'Protein'. You can override using the
## `.groups` argument.
```

```
# Plot it with barplot
ggplot(fat_rats_sub_df,
```

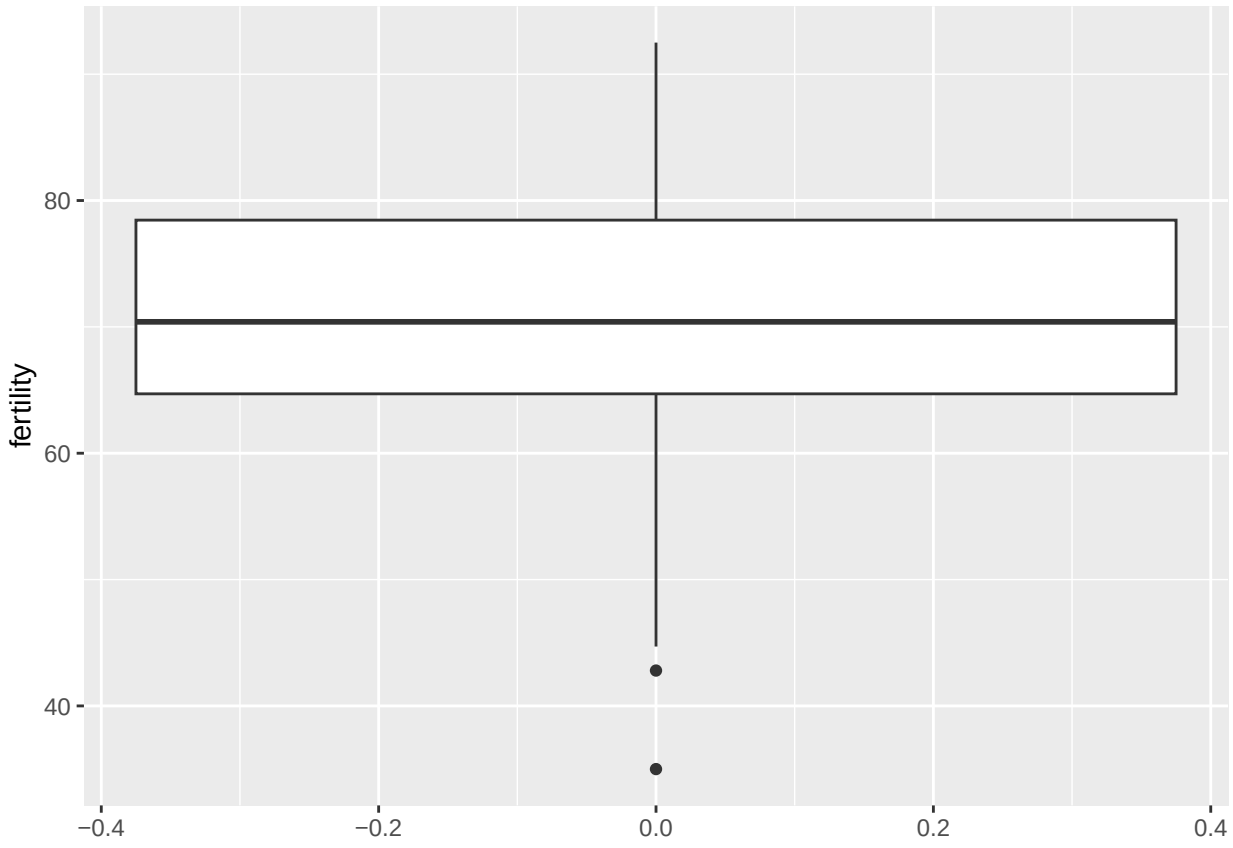
```
      mapping = aes(x = Source, y = gain, fill = Protein)) +
  geom_bar(stat = 'identity', position = 'dodge')
```



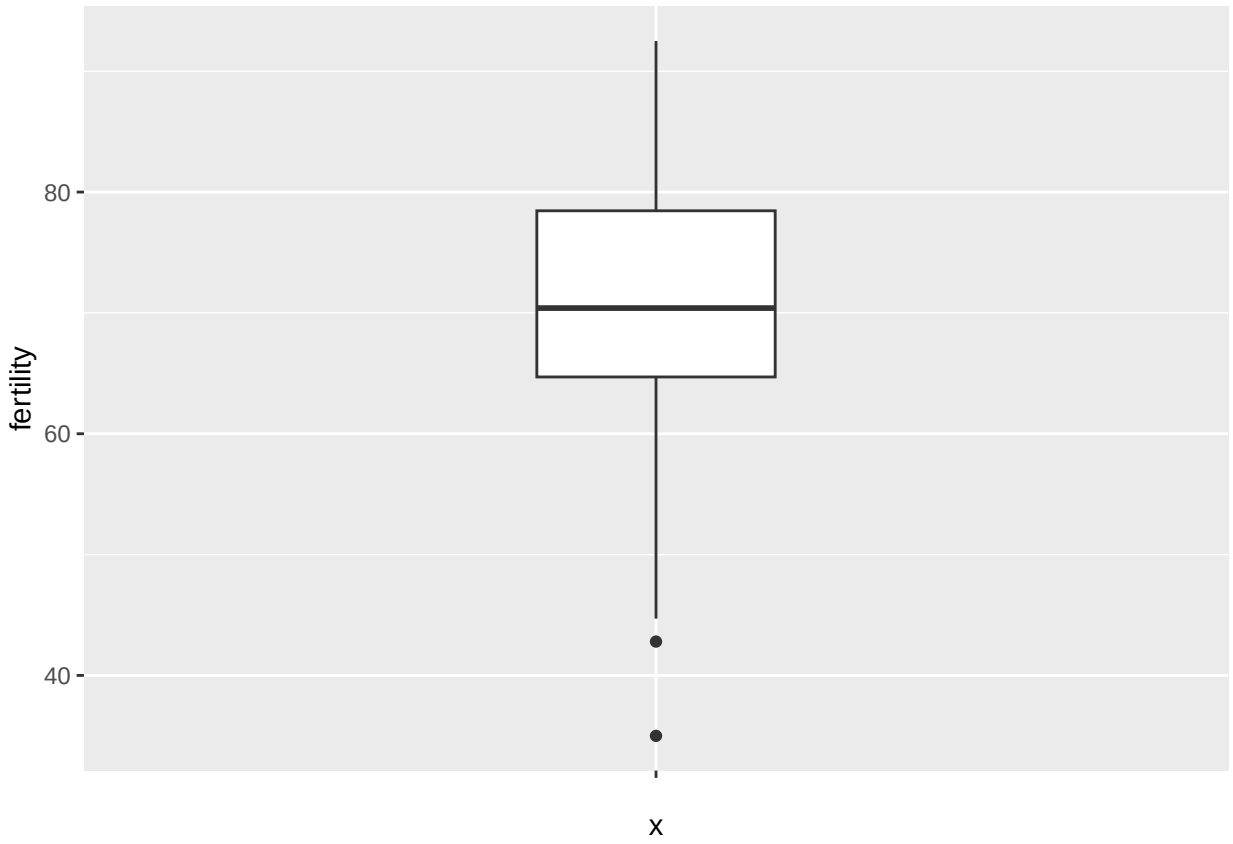

```
# We can add error bars to barplots using geom_errorbar. geom_errorbar must match default width of bars
ggplot(fat_rats_sub_df,
       mapping = aes(x = Source, y = gain, fill = Protein,
                     ymin = gain - se, ymax = gain + se)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  geom_errorbar(width = 0.2, position = position_dodge(width = 0.9))
```



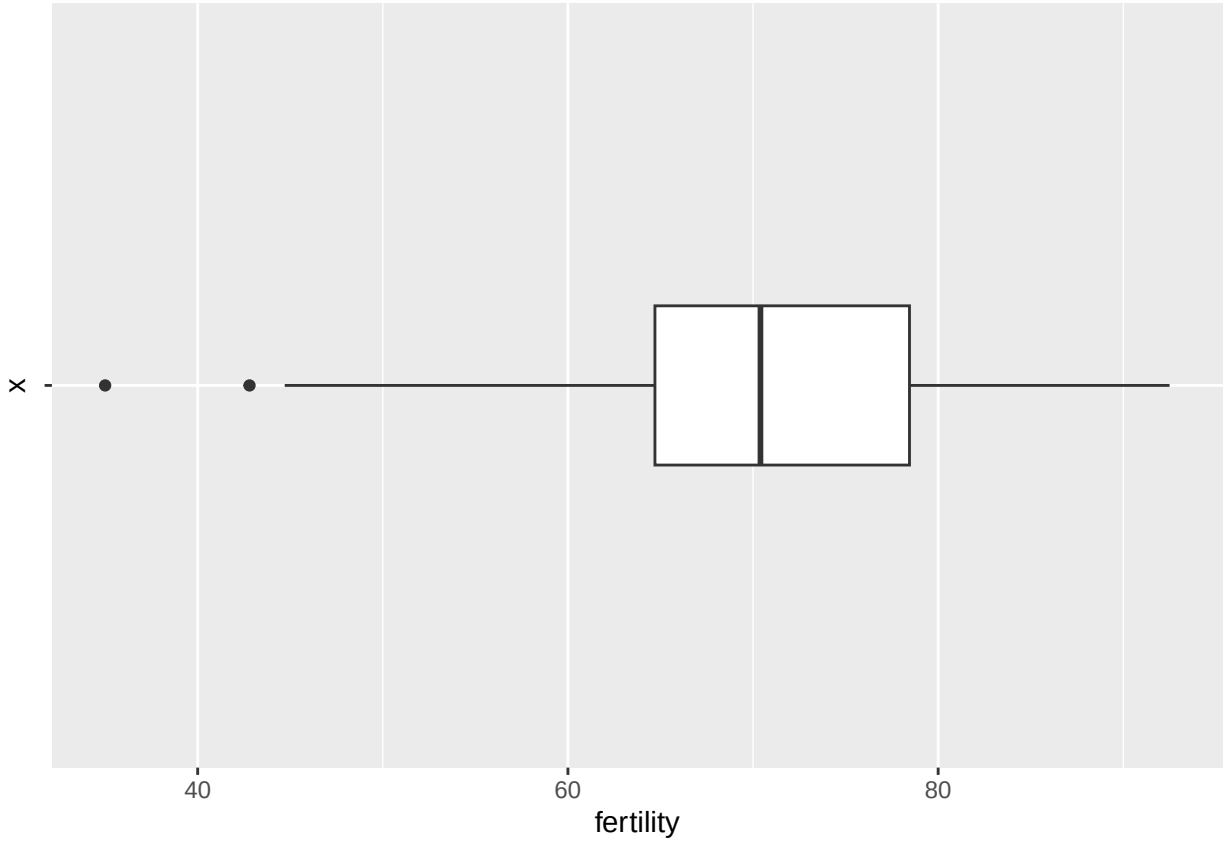

```
# Tukey Boxplots
# Most common subtype and ar the default type implemented in ggplot2 using geom_boxplot()
# Read in swiss data frame
swiss_df = swiss %>%
  rownames_to_column('province') %>%
  rename_all(tolower) %>%
  mutate(catholic = catholic > 50)



# Use Tukey boxplot to display distribution of the fertility variable in swiss dataframe
ggplot(swiss_df,
       mapping = aes(y = fertility)) +
  geom_boxplot()
```



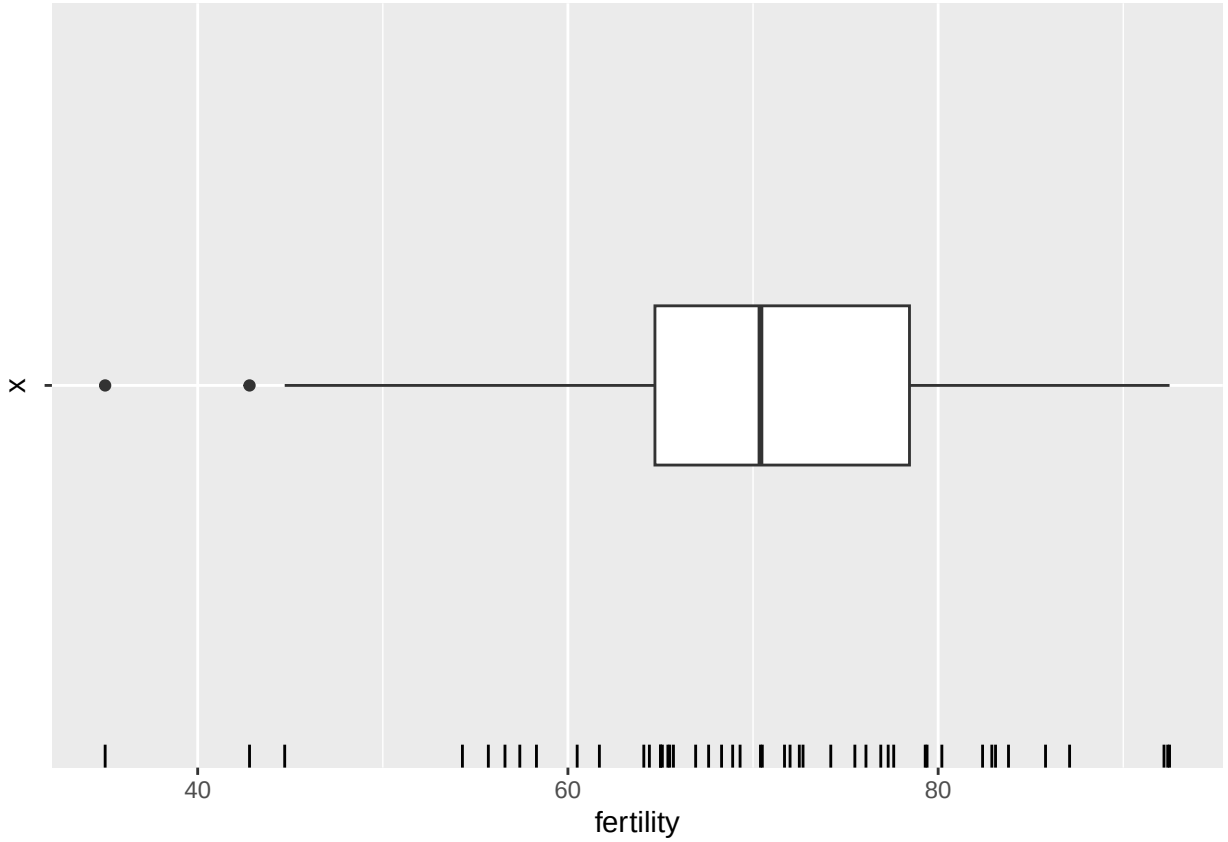

```
# To tell R that the x variable is discrete, use x = ''. This changes width of box plot
ggplot(swiss_df,
       mapping = aes(x = '', y = fertility)) +
  geom_boxplot(width = 0.25)
```



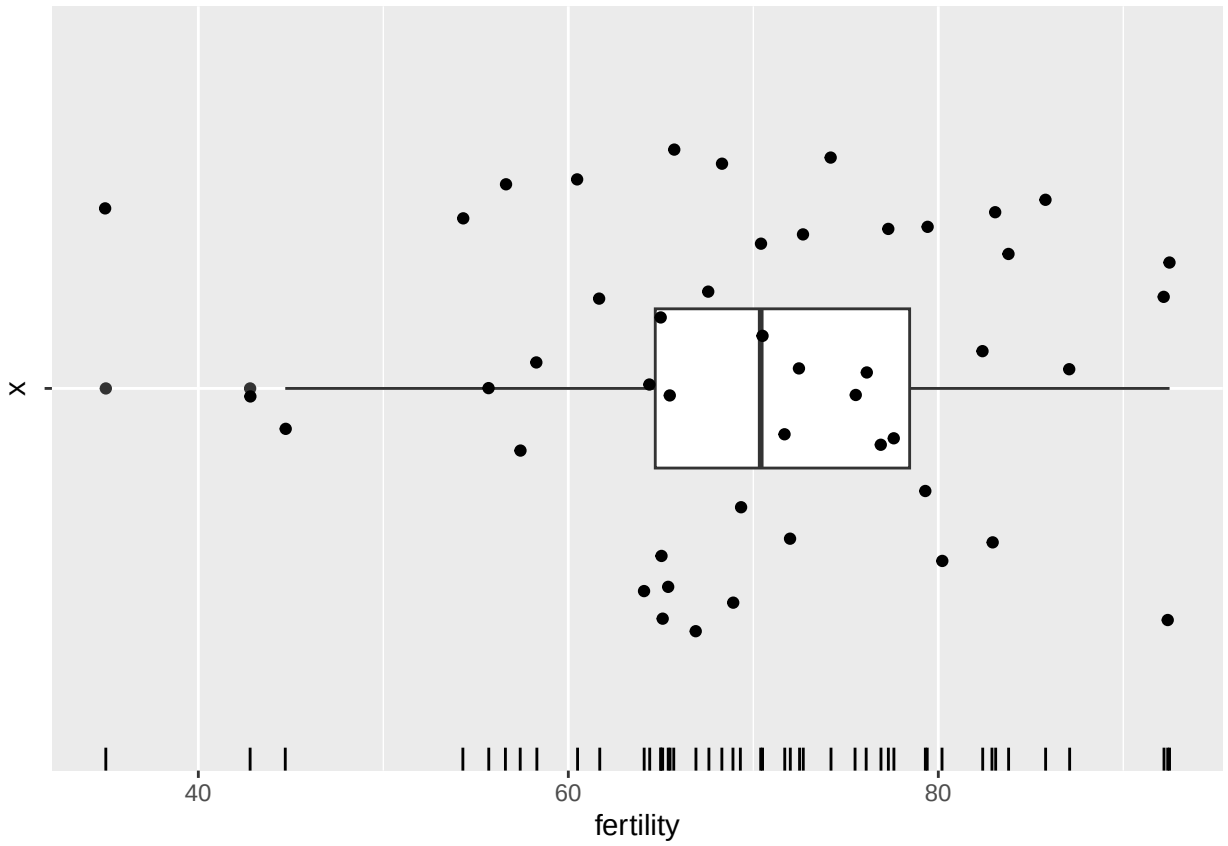

```
# You can convert vertical box plot to horizontal one using coord_flip()
ggplot(swiss_df,
       mapping = aes(x = '', y = fertility)) +
  geom_boxplot(width = 0.25) +
  coord_flip()
```



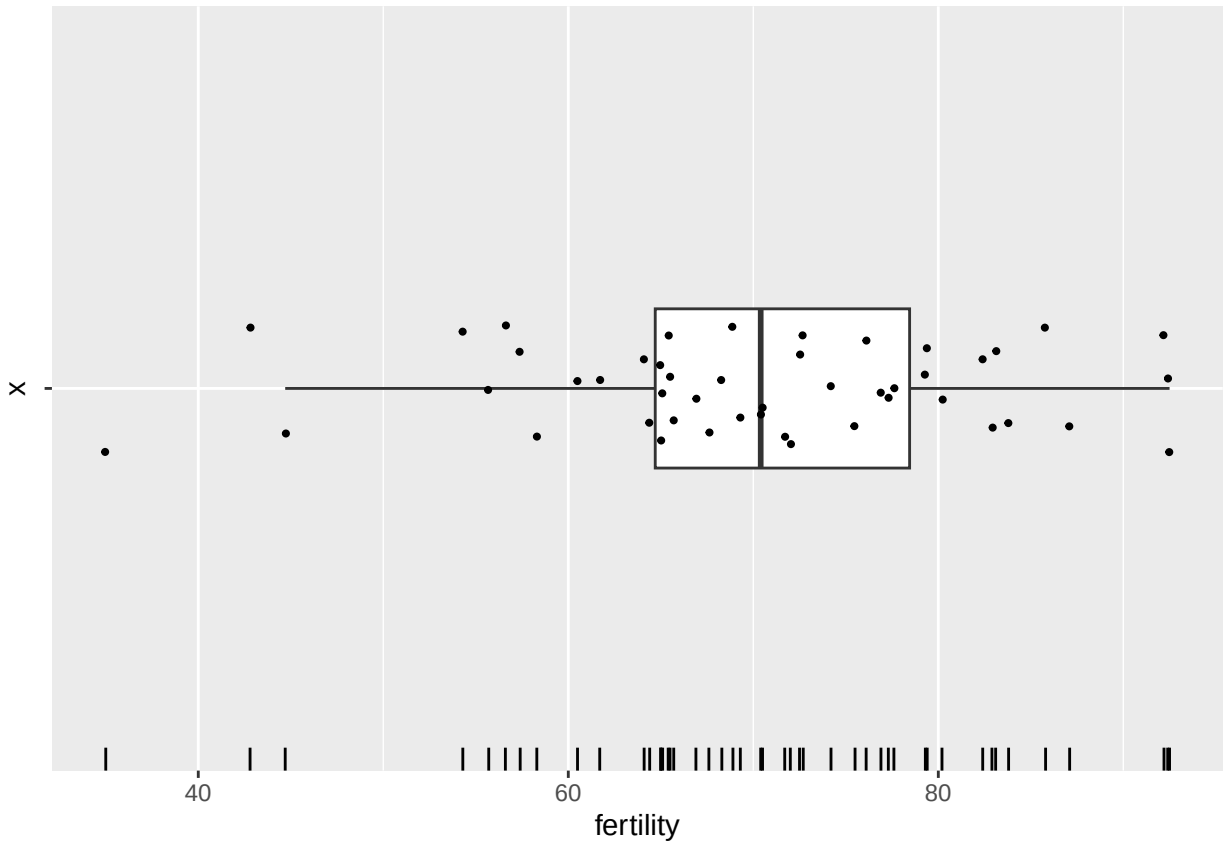

```
# Add rug plot using geom_plot() which displays location of every data point usint a short line
ggplot(swiss_df,
       mapping = aes(x = '', y = fertility)) +
  geom_boxplot(width = 0.25) +
  coord_flip() +
  geom_rug(sides = 'l')
```



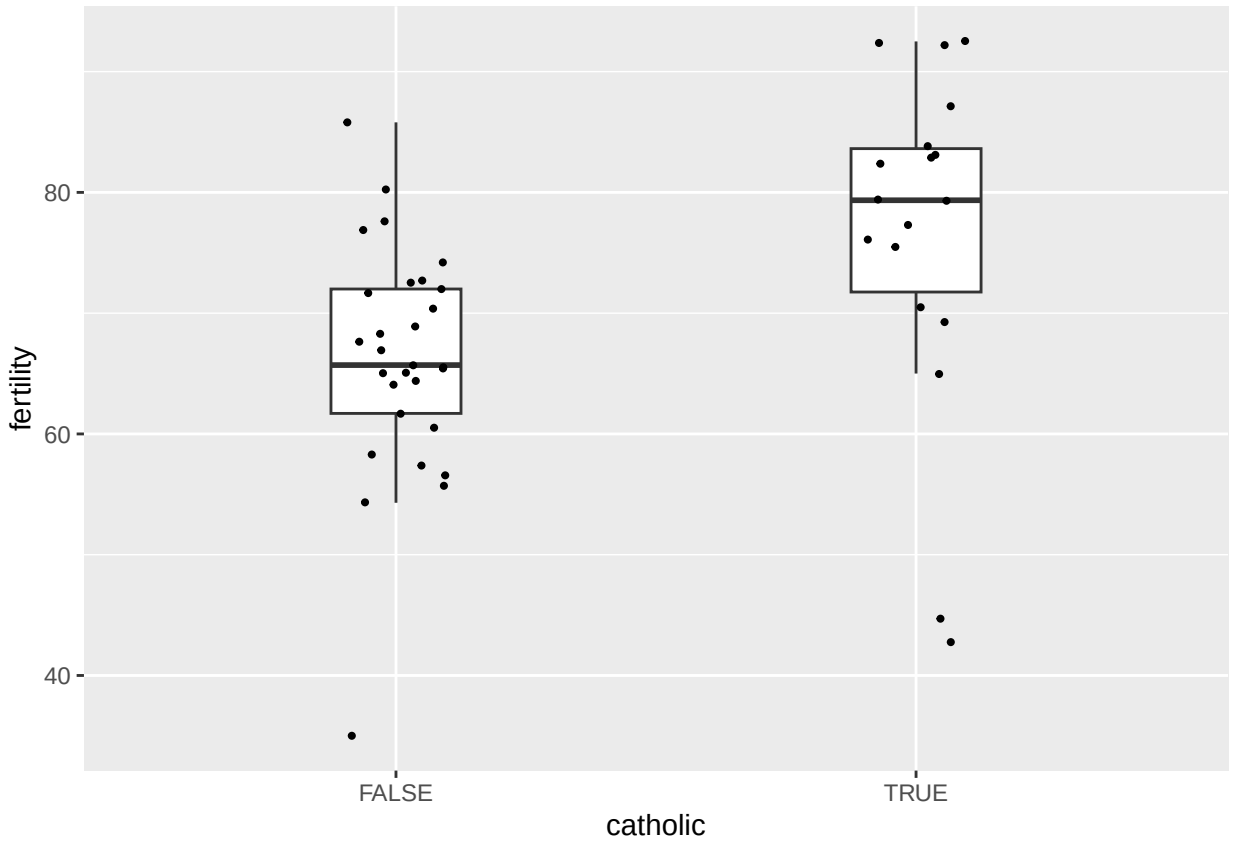

```
# sides = 'l' argument says that we only want rug on left side of graph, which is bottom axis when grap

# Can also add jitter plot which is like a scatter plot of each individual point where each point is pe
ggplot(swiss_df,
       mapping = aes(x = '', y = fertility)) +
  geom_boxplot(width = 0.25) +
  coord_flip() +
  geom_rug(sides = 'l') +
  geom_jitter()
```




```
# Can clean up plot by reducing spread points along vertical axis so they are within width of box. Sinc
ggplot(swiss_df,
       mapping = aes(x = '', y = fertility)) +
  geom_boxplot(width = 0.25, outlier.shape = NA) +
  coord_flip() +
  geom_rug(sides = 'l') +
  geom_jitter(width = 0.1, size = 0.75)
```

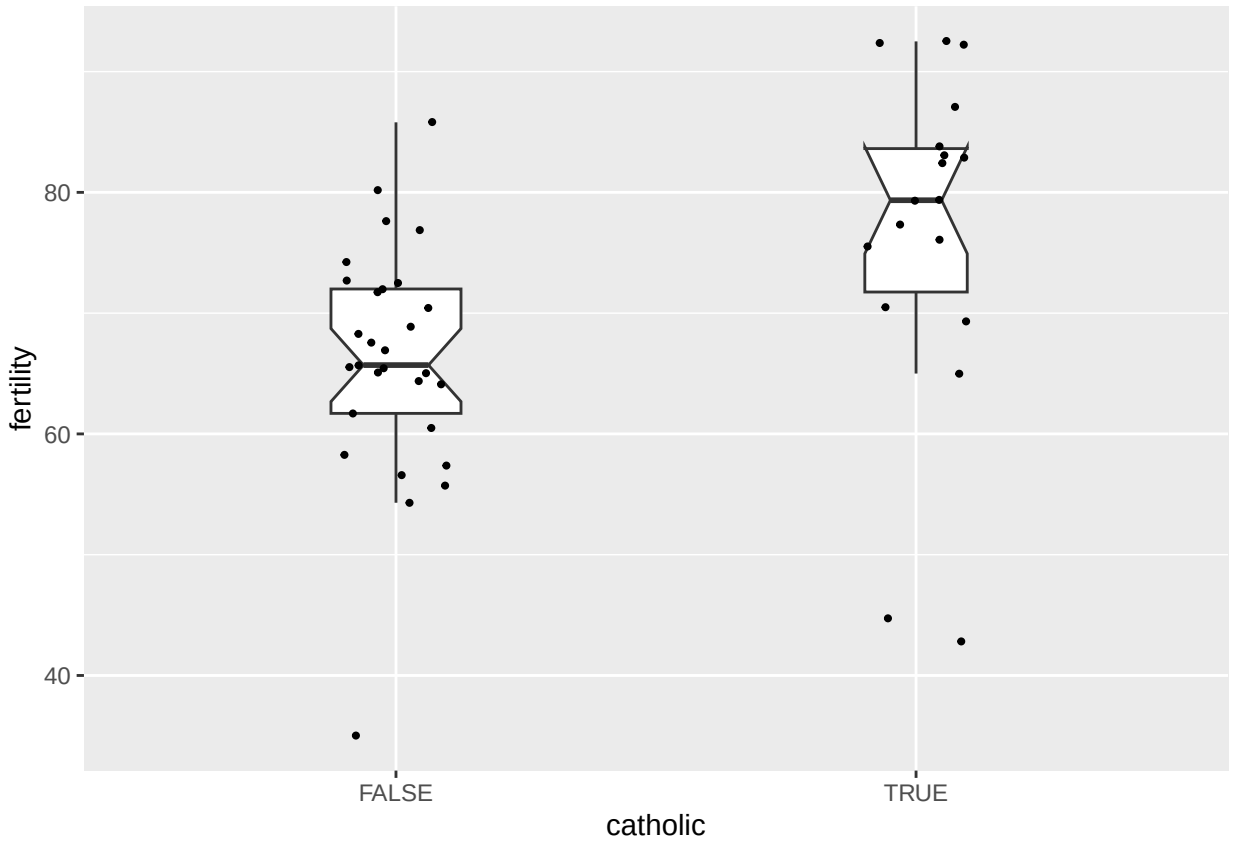



```
# We can map x property to a third variable to display multiple boxplots
ggplot(swiss_df,
       mapping = aes(x = catholic, y = fertility)) +
  geom_boxplot(width = 0.25, outlier.shape = NA) +
  geom_jitter(width = 0.1, size = 0.75)
```

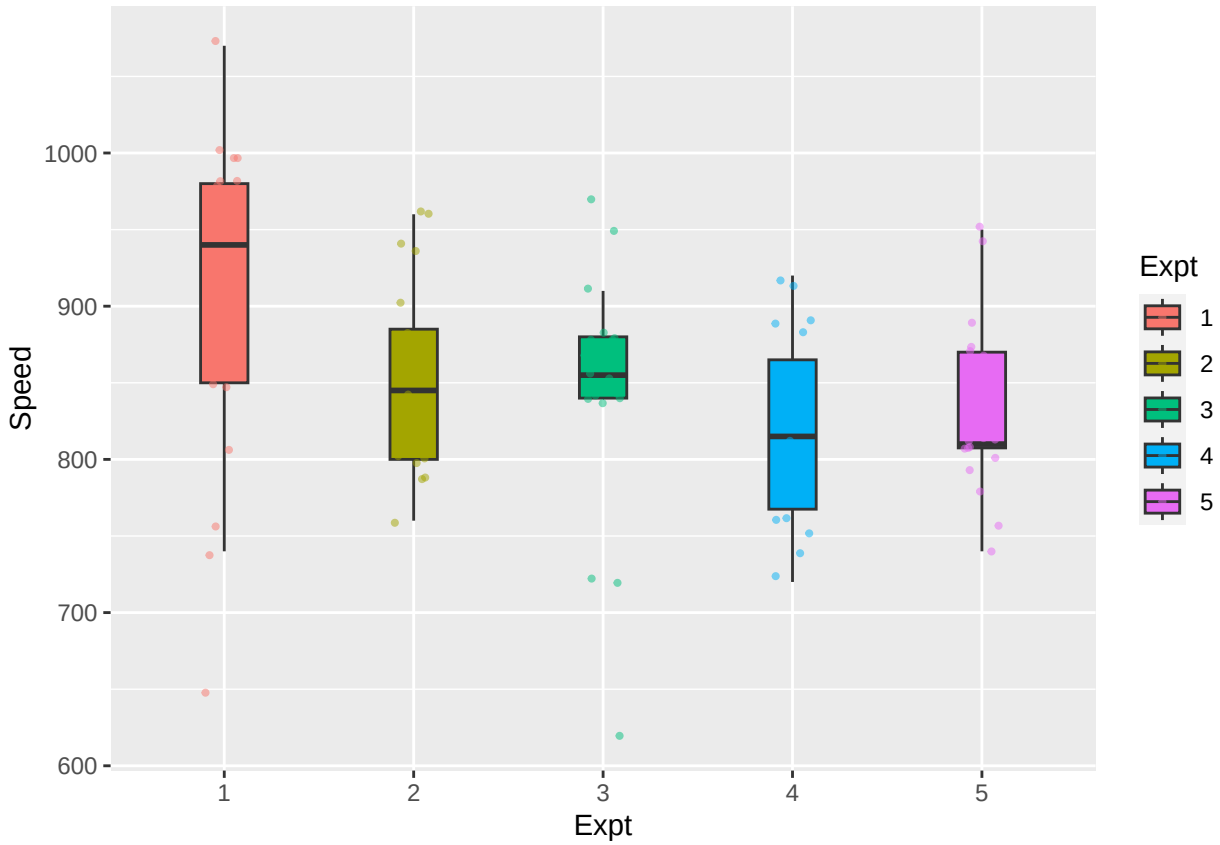



```
# We can scale the width of the box with the square root of the sample size by setting varwidth to TRUE
# We can also set notch to TRUE to provide measure of the uncertainty concerning true value of median
ggplot(swiss_df,
       mapping = aes(x = catholic, y = fertility)) +
  geom_boxplot(width = 0.25, outlier.shape = NA, varwidth = T, notch = T) +
  geom_jitter(width = 0.1, size = 0.75)
```

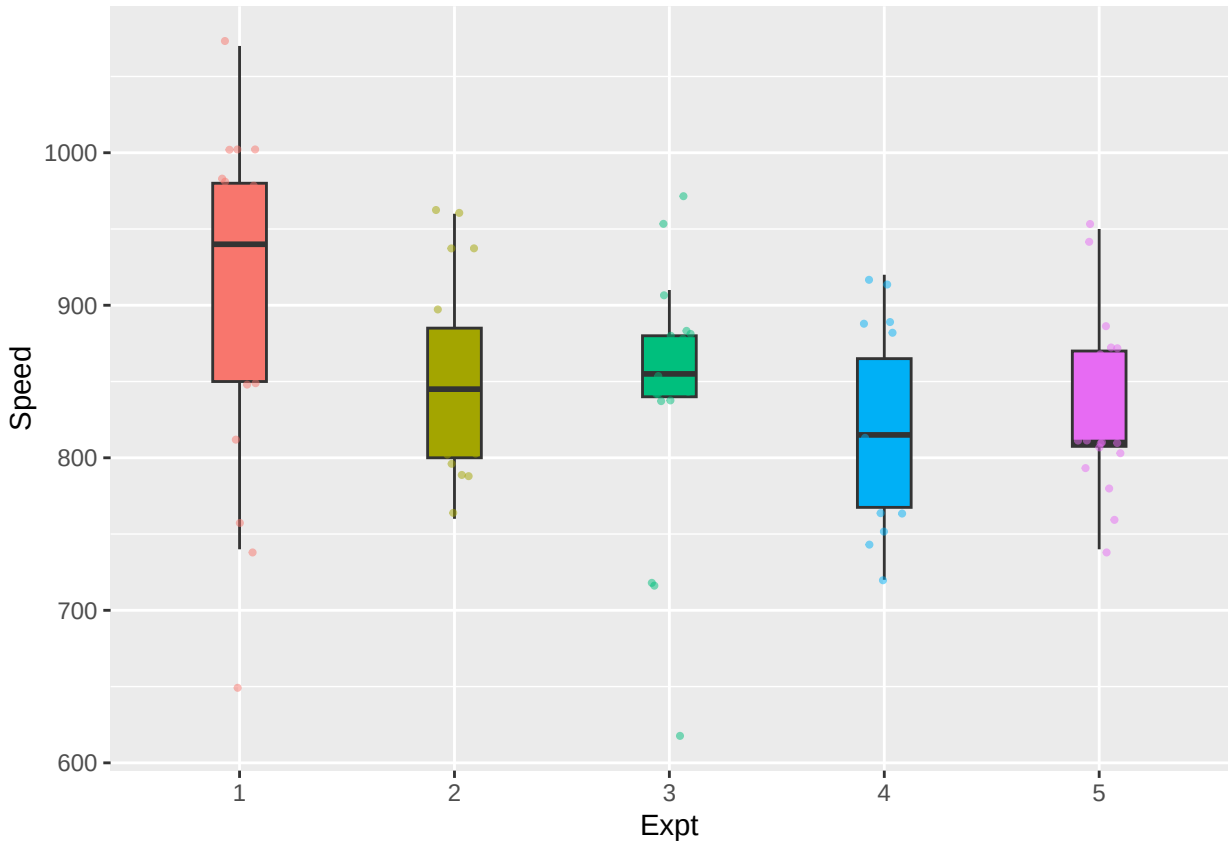
```
## Notch went outside hinges
## i Do you want `notch = FALSE`?
```



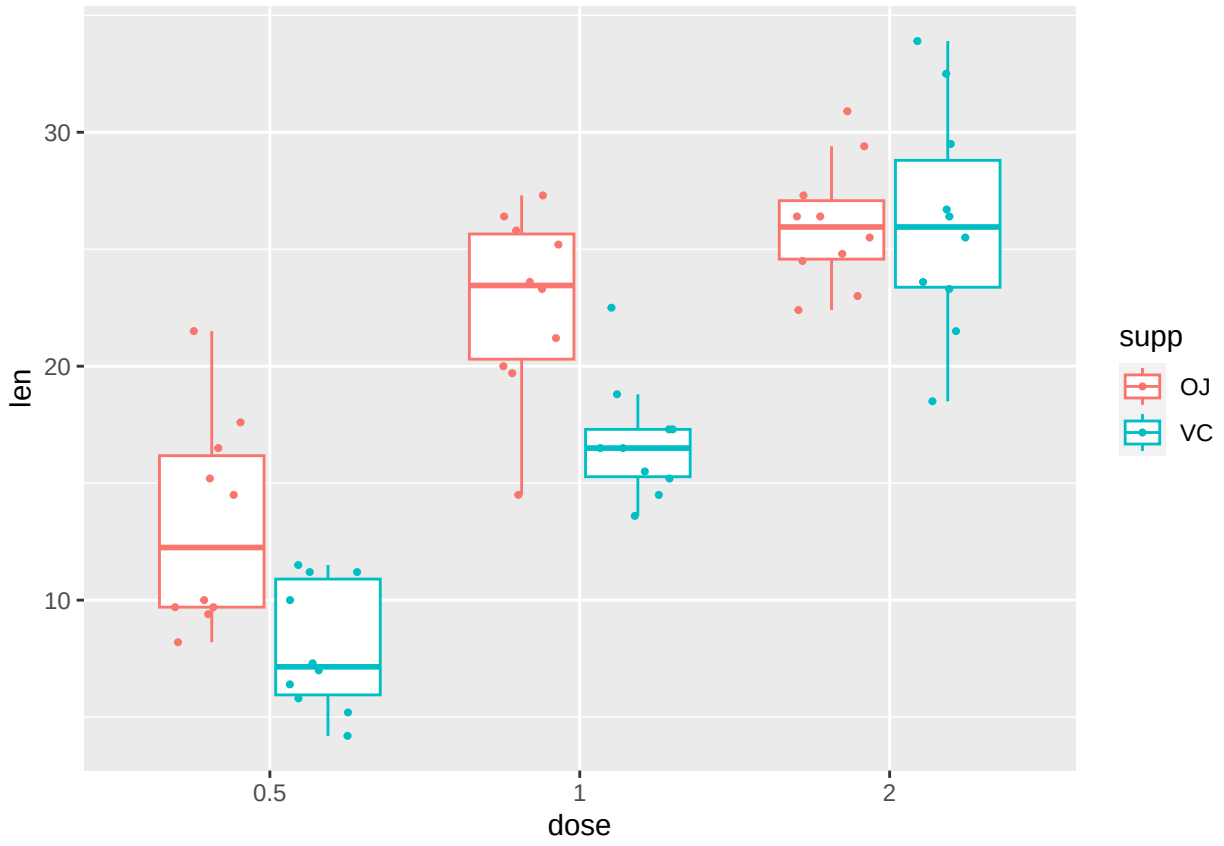

```
# You can color code different box plots to make them more distinguishable using color or fill, or both
# Use built in dataset morley
morley_df = morley
morley_df %>%
  mutate(Expt = as.factor(Expt)) %>%
  ggplot(
    mapping = aes(x = Expt, y = Speed, fill = Expt)) +
  geom_boxplot(width = 0.25, outlier.shape = NA, varwidth = T) +
  geom_jitter(aes(color = Expt), alpha = 0.5, width = 0.1, size = 0.75)
```



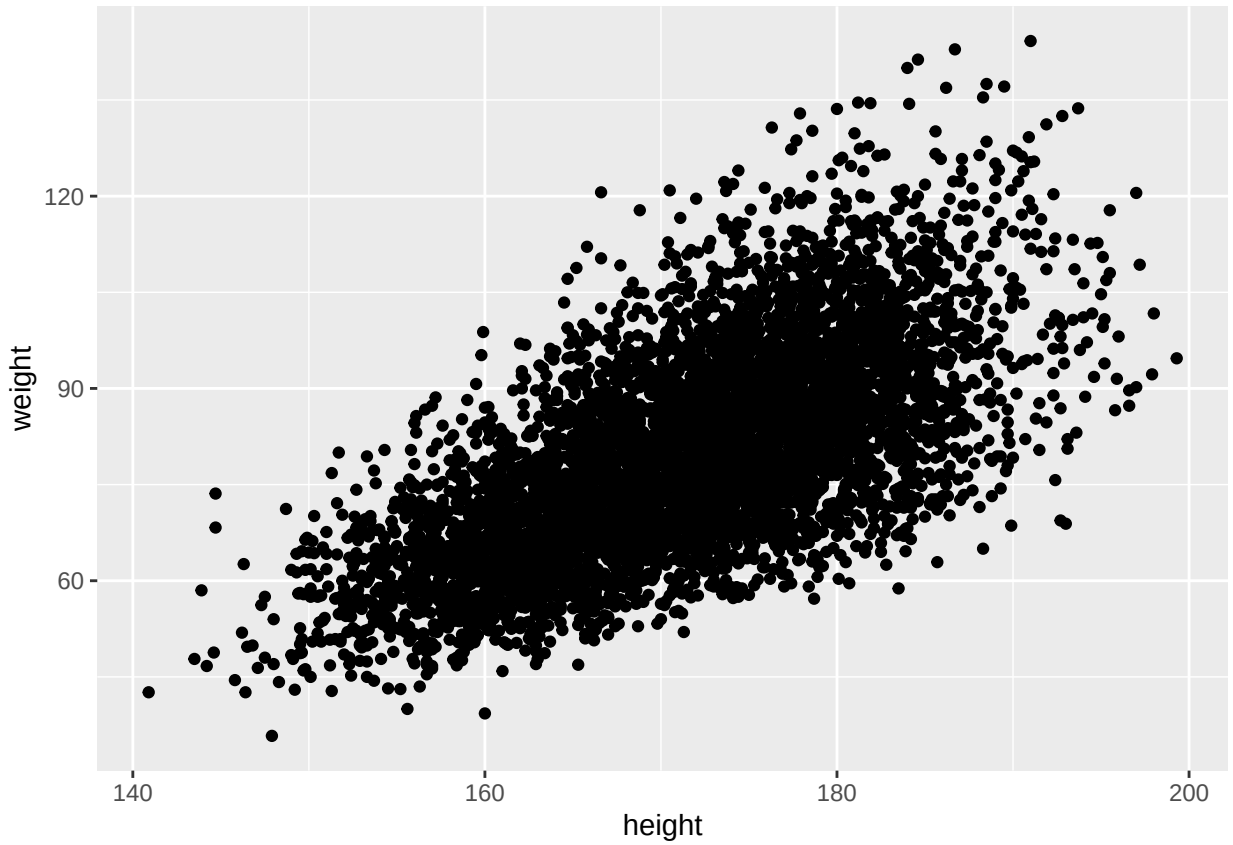

```
# In this example, since box plots are labeled, we can remove legend with legend.position = 'none'
morley_df %>%
  mutate(Expt = as.factor(Expt)) %>%
  ggplot(
    mapping = aes(x = Expt, y = Speed, fill = Expt)) +
  geom_boxplot(width = 0.25, outlier.shape = NA, varwidth = T) +
  geom_jitter(aes(color = Expt), alpha = 0.5, width = 0.1, size = 0.75) +
  theme(legend.position = 'none')
```



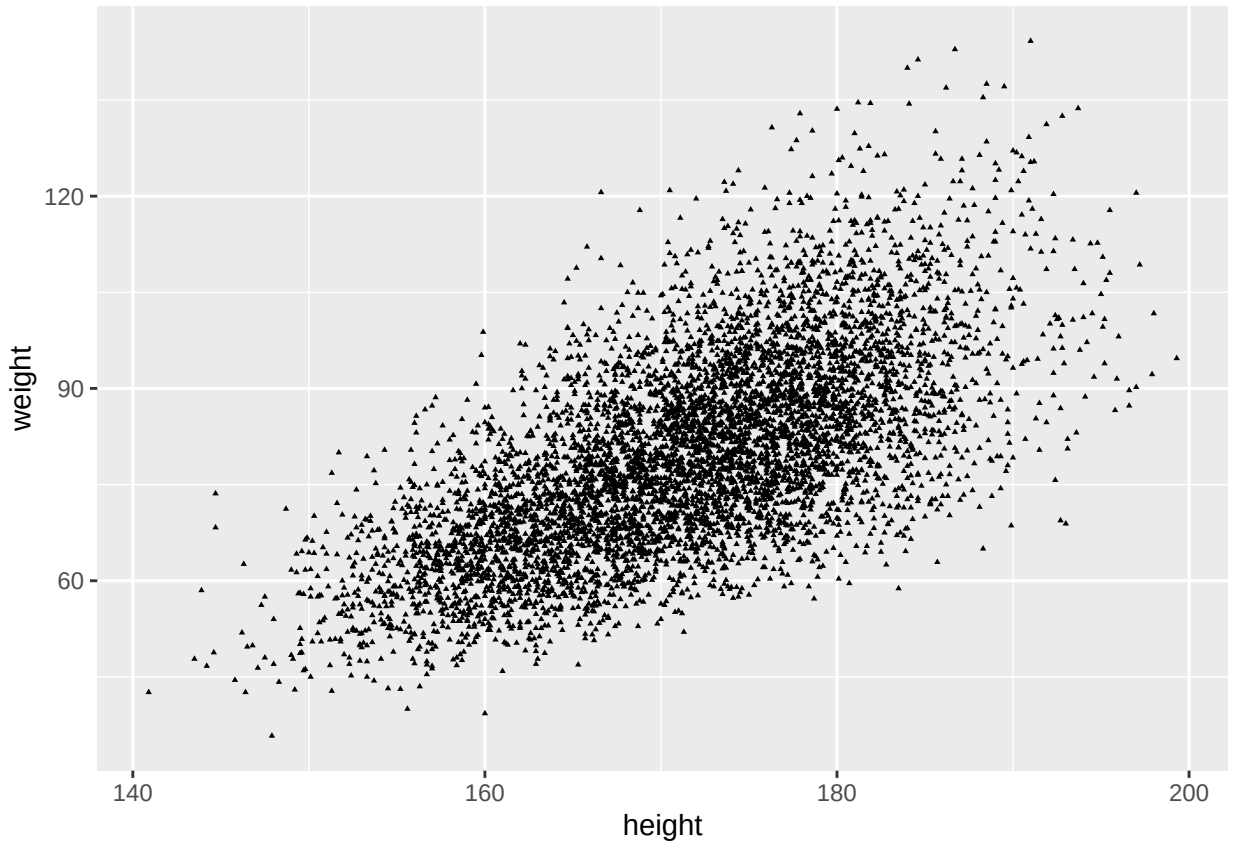

```
# We can use another as the color variable thus dispalying boxplots for each combination of the values
# Read in built in dataset ToothGrowth
tooth_growth_df = ToothGrowth
tooth_growth_df %>%
  mutate(dose = as.factor(dose)) %>%
  ggplot(mapping = aes(x = dose, y = len, color = supp)) +
  geom_boxplot(outlier.shape = NA, varwidth = T) +
  geom_jitter(position = position_jitterdodge(0.5), size = 0.75)
```



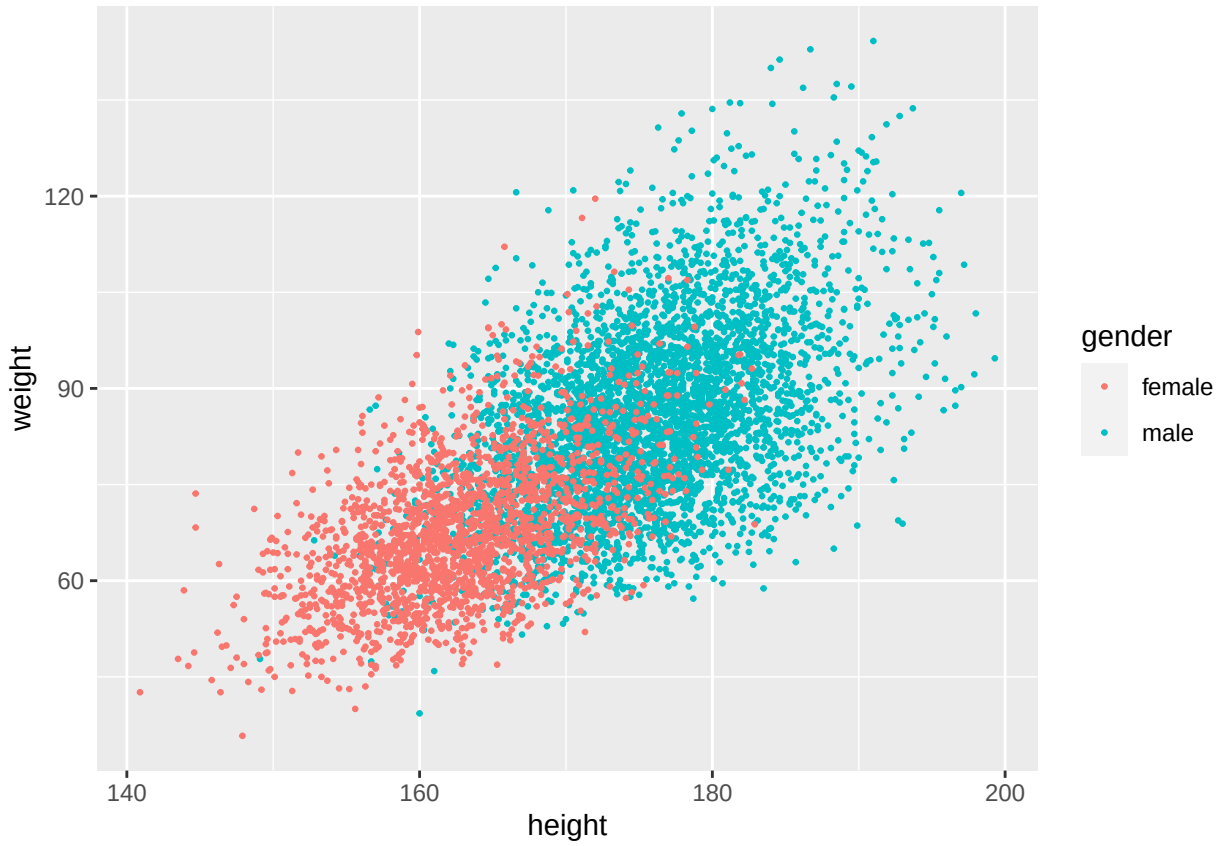

```
# Scatterplots
# More in-depth scatterplots
# Scatterplot of weight_df as function of height
ggplot(weight_df,
       mapping = aes(x = height, y = weight)) +
  geom_point()
```




```
# We can change size and shape of point within geoms()
ggplot(weight_df,
       mapping = aes(x = height, y = weight)) +
  geom_point(size = 0.5, shape = 'triangle')
```



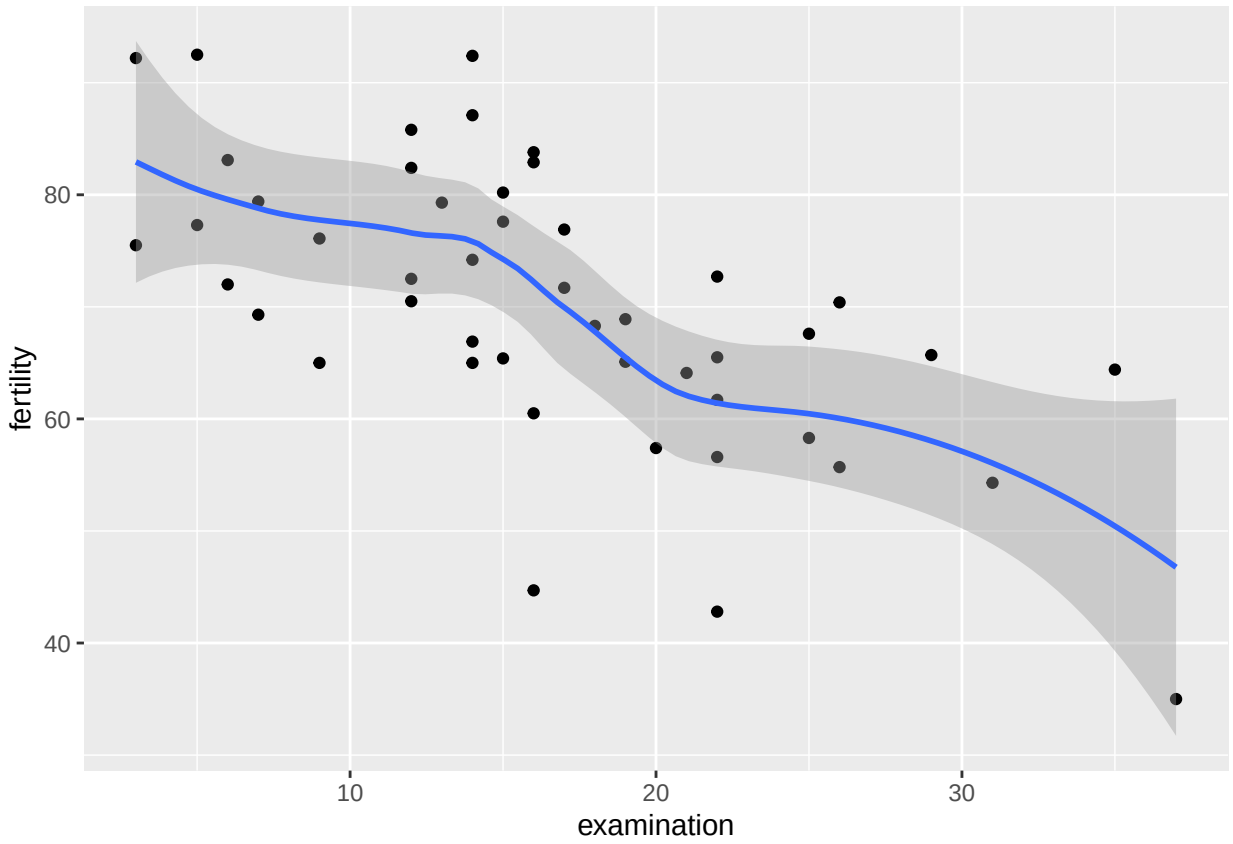

```
# Use another variable, gender, as color within mapping
ggplot(weight_df,
       mapping = aes(x = height, y = weight, color = gender)) +
  geom_point(size = 0.5)
```




```
# Add rug plot to both x and y axis to see distribution over individual variables
ggplot(weight_df,
       mapping = aes(x = height, y = weight, color = gender)) +
  geom_point(size = 0.5) +
  geom_rug(alpha = 0.5, linewidth = 1/10)
# I got a warning message about using 'size =' in geom_rug so I used 'linewidth' instead

# There is a wider set of marginal distributions in scatterplots available in ggMarginal function that
# to use ggExtra, assign plot to variable
p = ggplot(weight_df,
           mapping = aes(x = height, y = weight, color = gender)) +
  geom_point(size = 0.5)
# Use ggMarginal(p) to display it
ggMarginal(p)

# The grouping color is not shown on the density functions shown. Use following code to build histogram
ggMarginal(p, type = 'histogram', groupColour = T, groupFill = T,
           position = 'identity', alpha = 0.5)
```

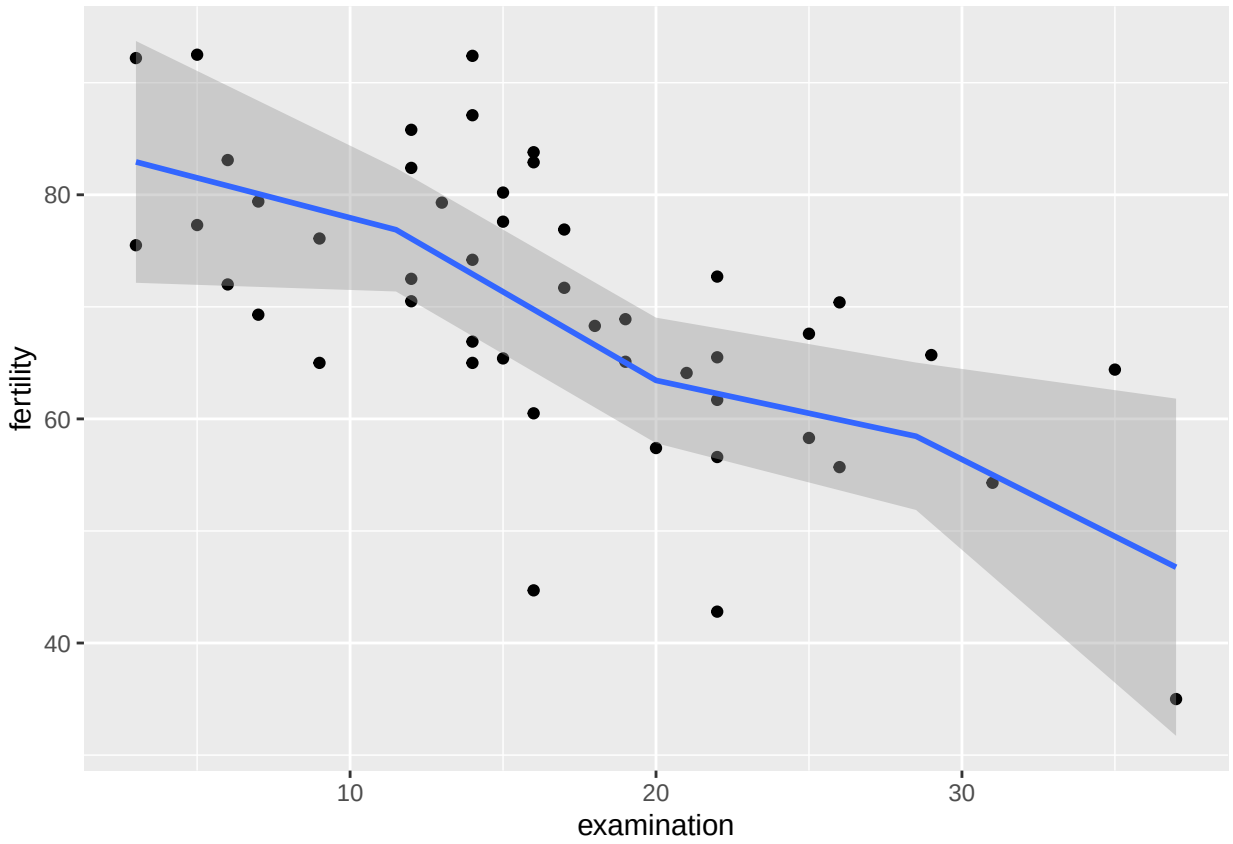

```
# We can add smoothing function. A loess(locally estimated scatterplot smoohting) captures the trends i
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



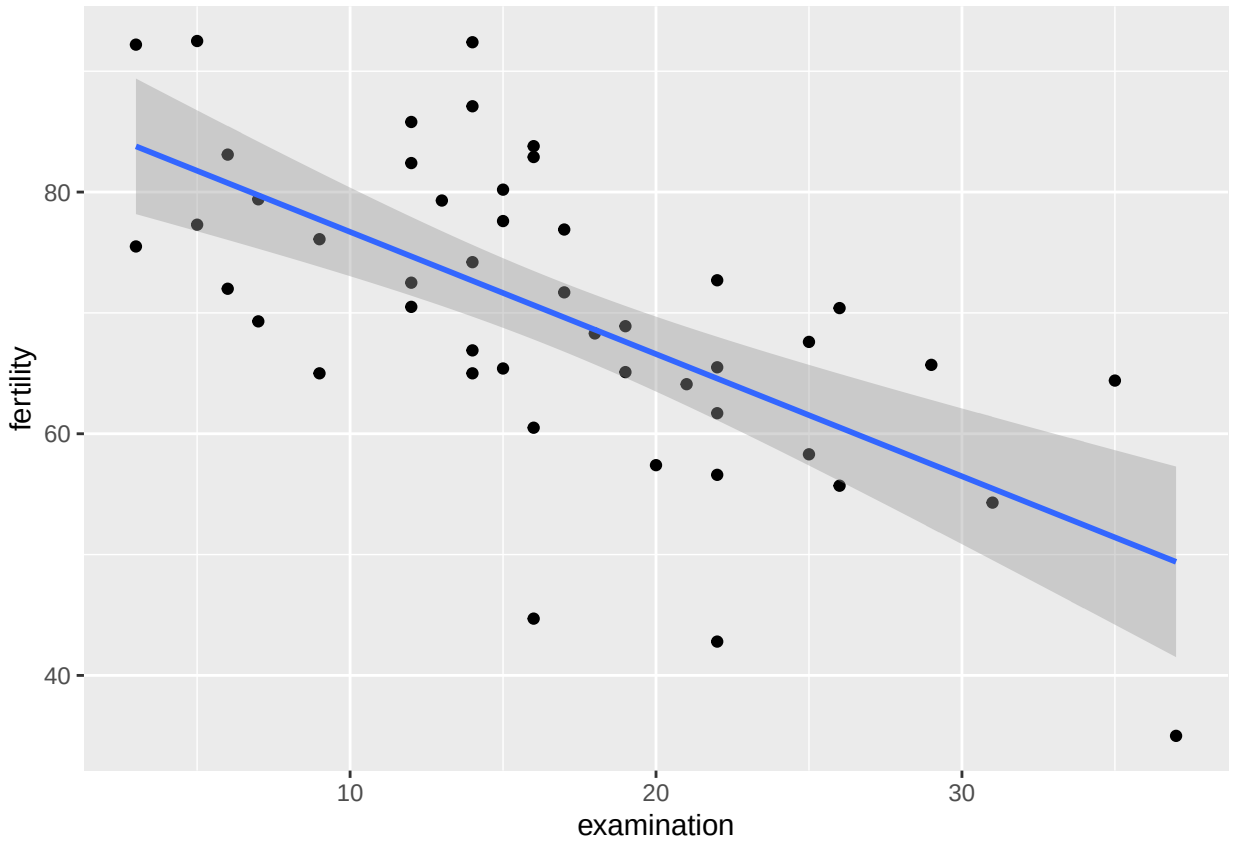

```
# loess smoother uses 80 evenly spaced points in the range of the predictor variable and uses linear re

# We can change the number of evaluations points and the regression function by specying the value of n
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility)) +
  geom_point() + geom_smooth(n = 5, formula = y ~ x)
```

```
## `geom_smooth()` using method = 'loess'
```



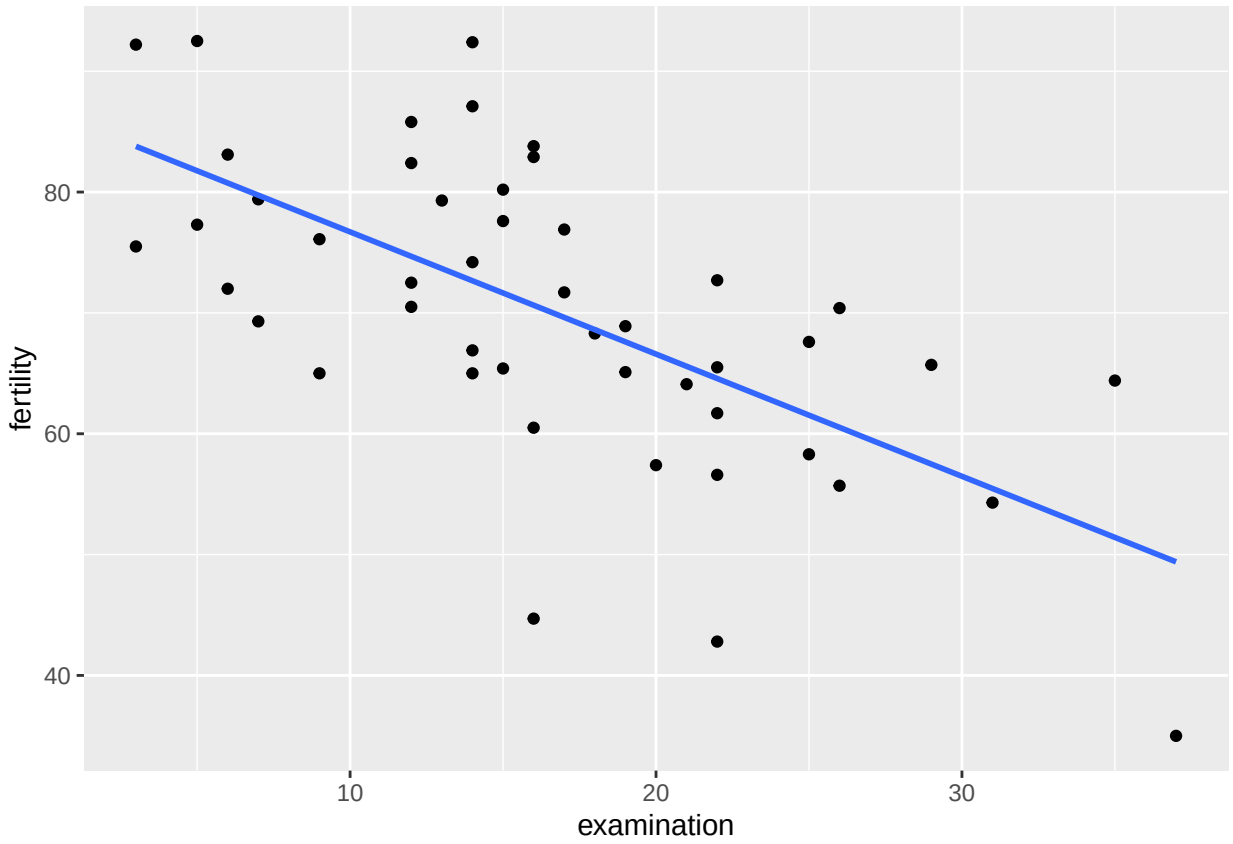

```
# formula = y ~ x used to fit linear regression line to to the data
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility)) +
  geom_point() + geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

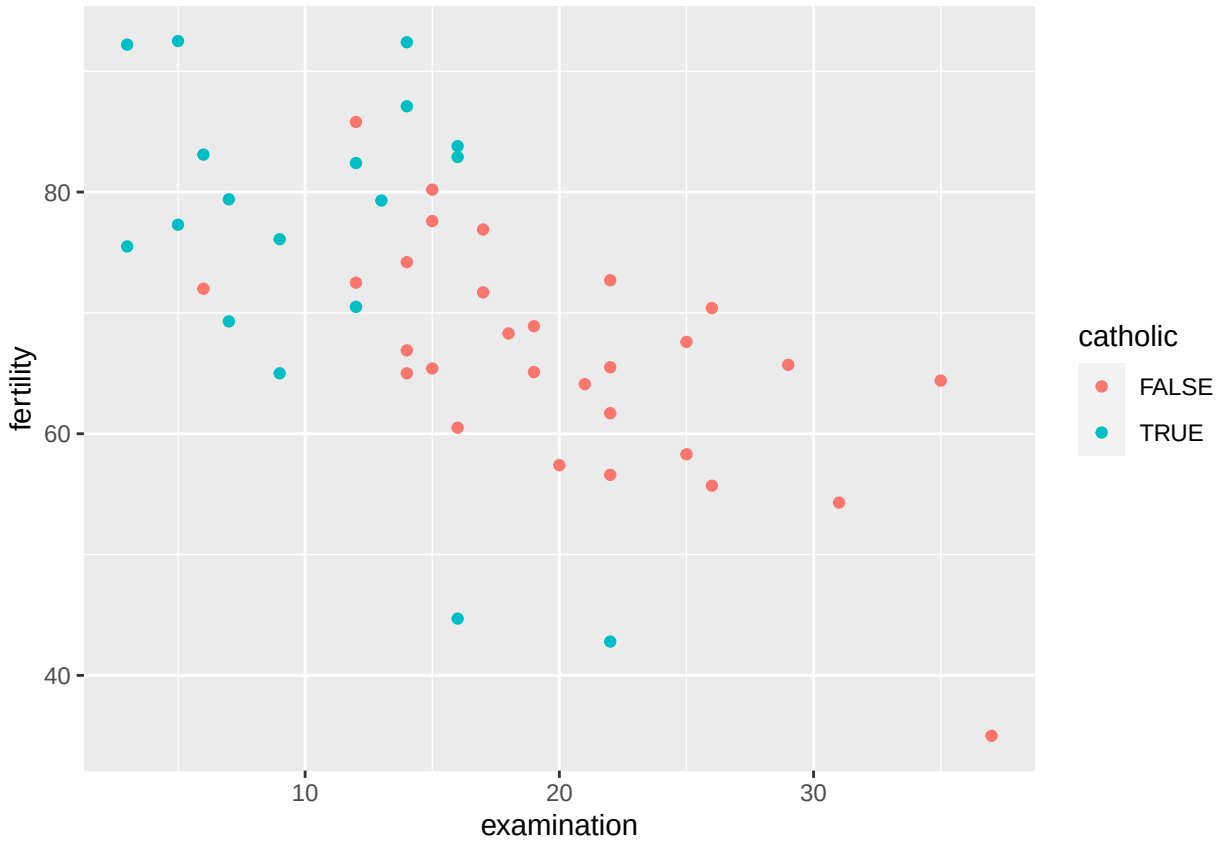



```
# This displays the least squares line of best fit that we would obtain from standard linear regression
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility)) +
  geom_point() + geom_smooth(method = 'lm', se = F)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

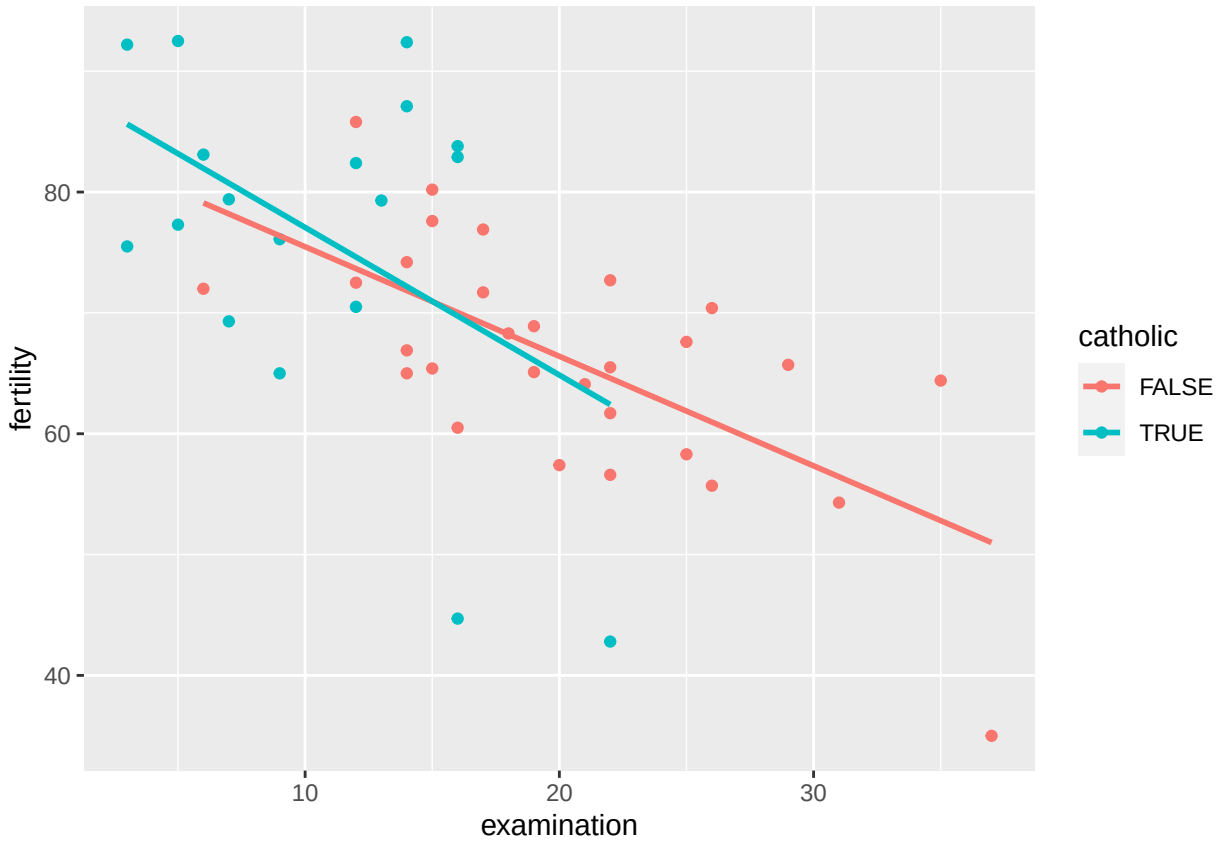



```
# Set color = catholic so each point will color-code whether corresponding province is Catholic or Prot
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility, color = catholic)) +
  geom_point()
```



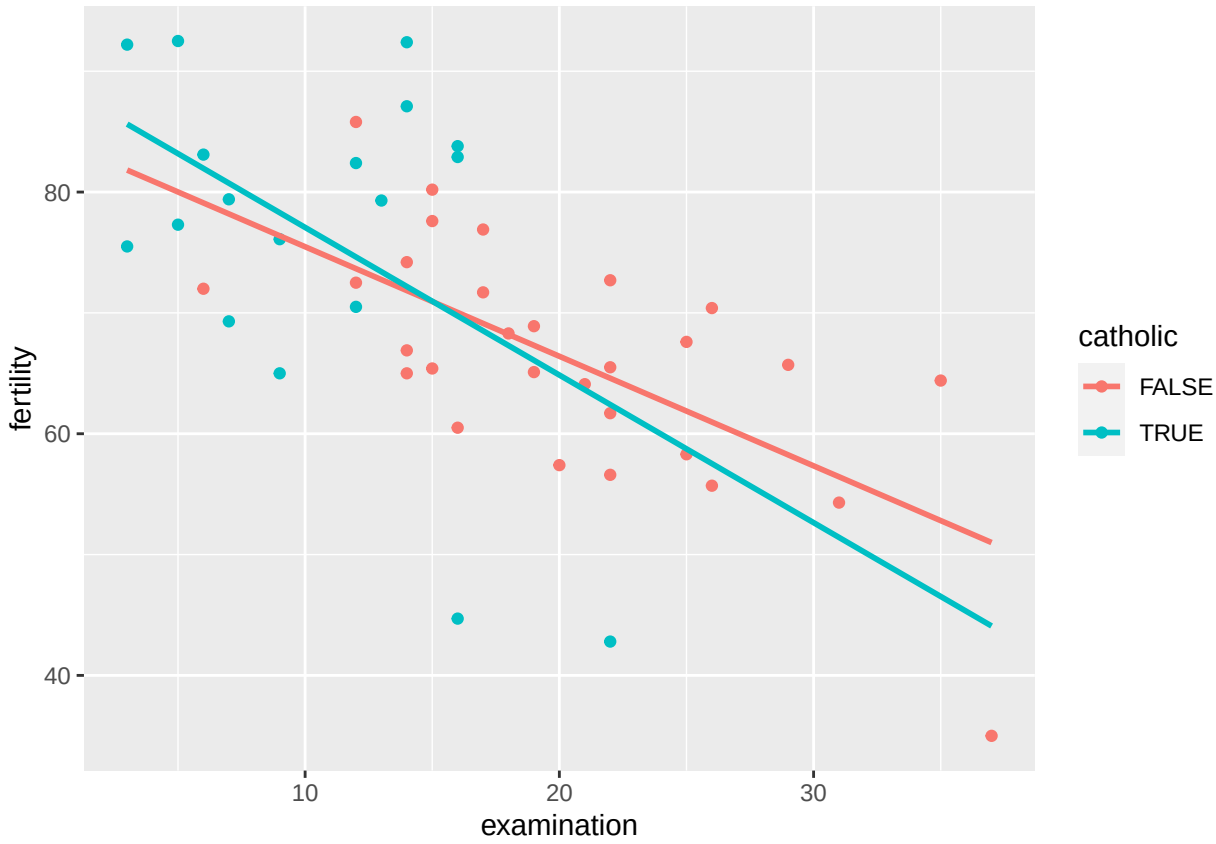

```
# Let's add smoothing curve or line of best fit
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility, color = catholic)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



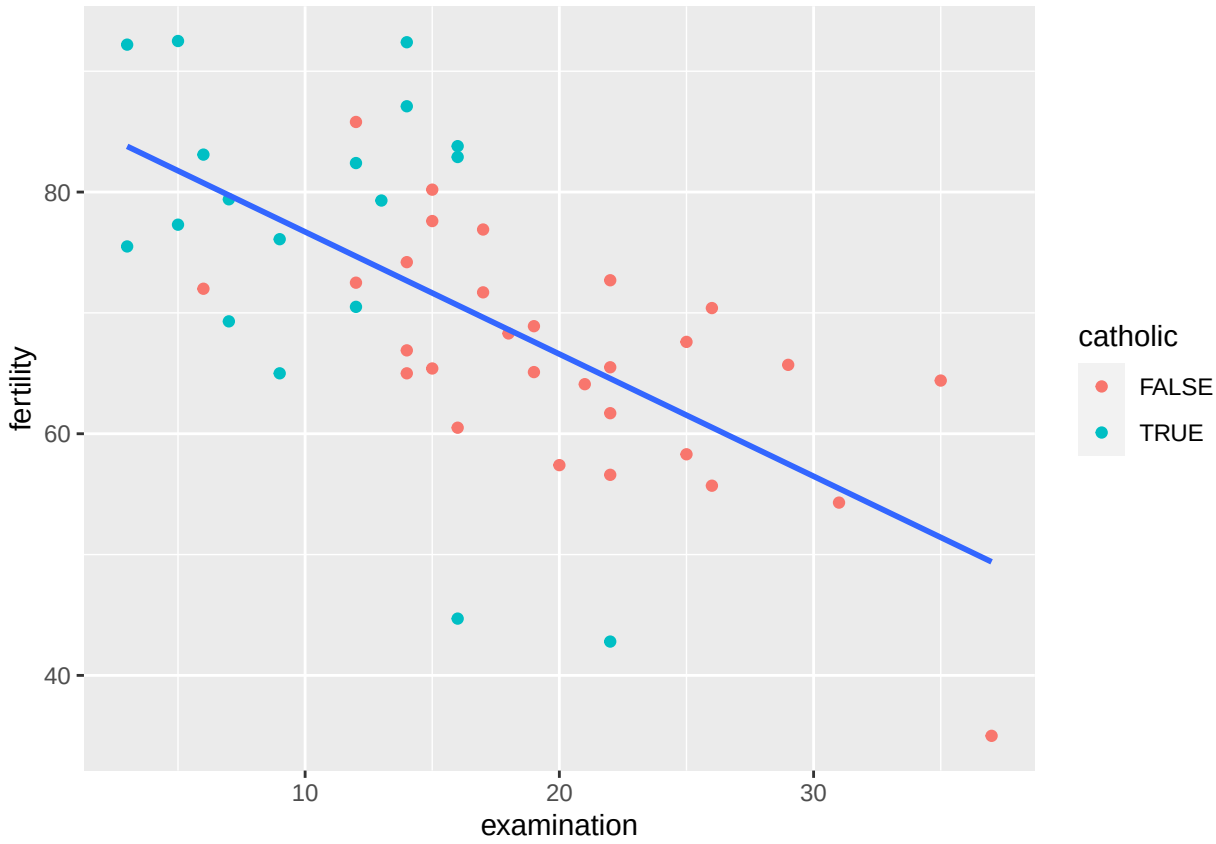

```
# Lines extend only over the range of subet of points. We can extend the line
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility, color = catholic)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F, fullrange = T)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

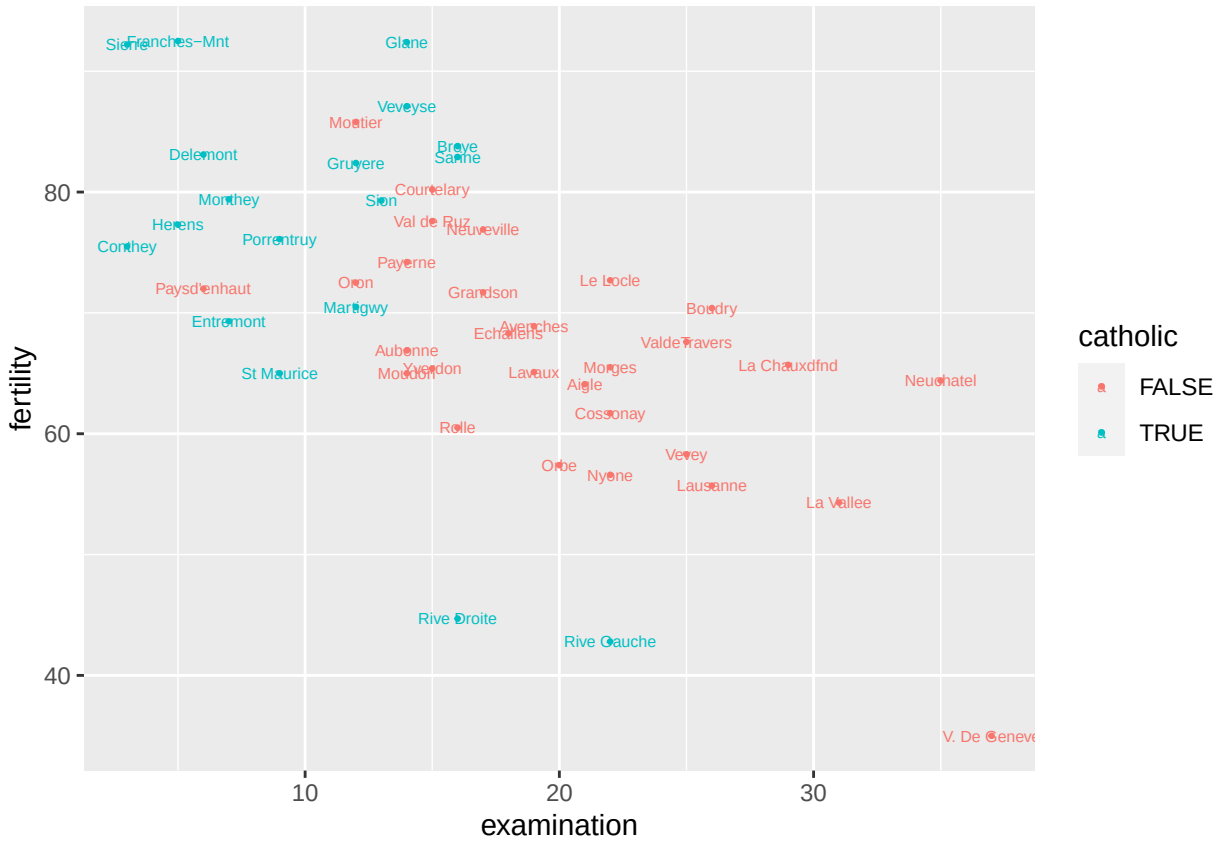



```
# To keep the points color-coded by catholic variable and produce single line of best fit, we need to r
ggplot(swiss_df,
       mapping = aes(x = examination, y = fertility)) +
  geom_point(mapping = aes(color = catholic)) +
  geom_smooth(method = 'lm', se = F, fullrange = T)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

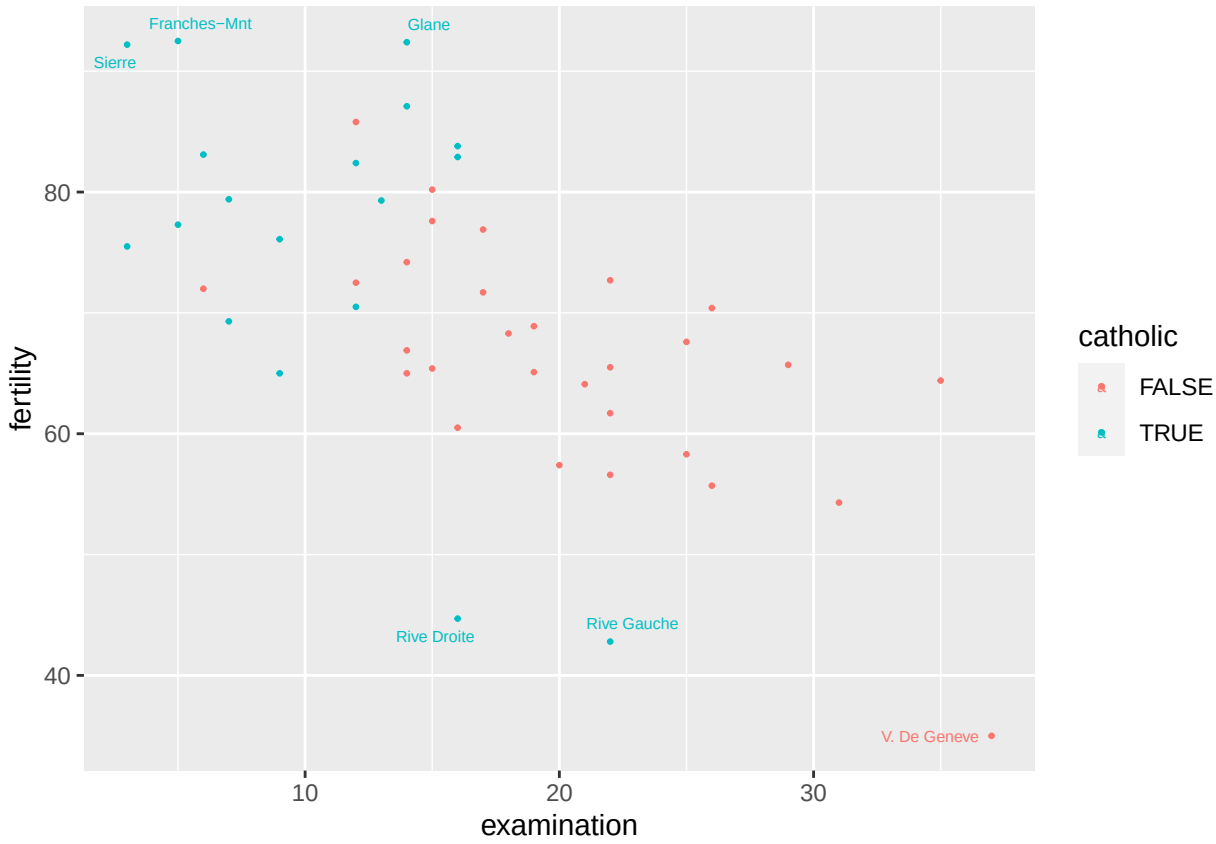



```
# We can add labels by setting the label mapping and using geom_text
swiss_df %>%
  ggplot(mapping = aes(x = examination, y = fertility,
                       label = province, color = catholic)) +
  geom_point(size = 0.5) +
  geom_text(size = 2)
```

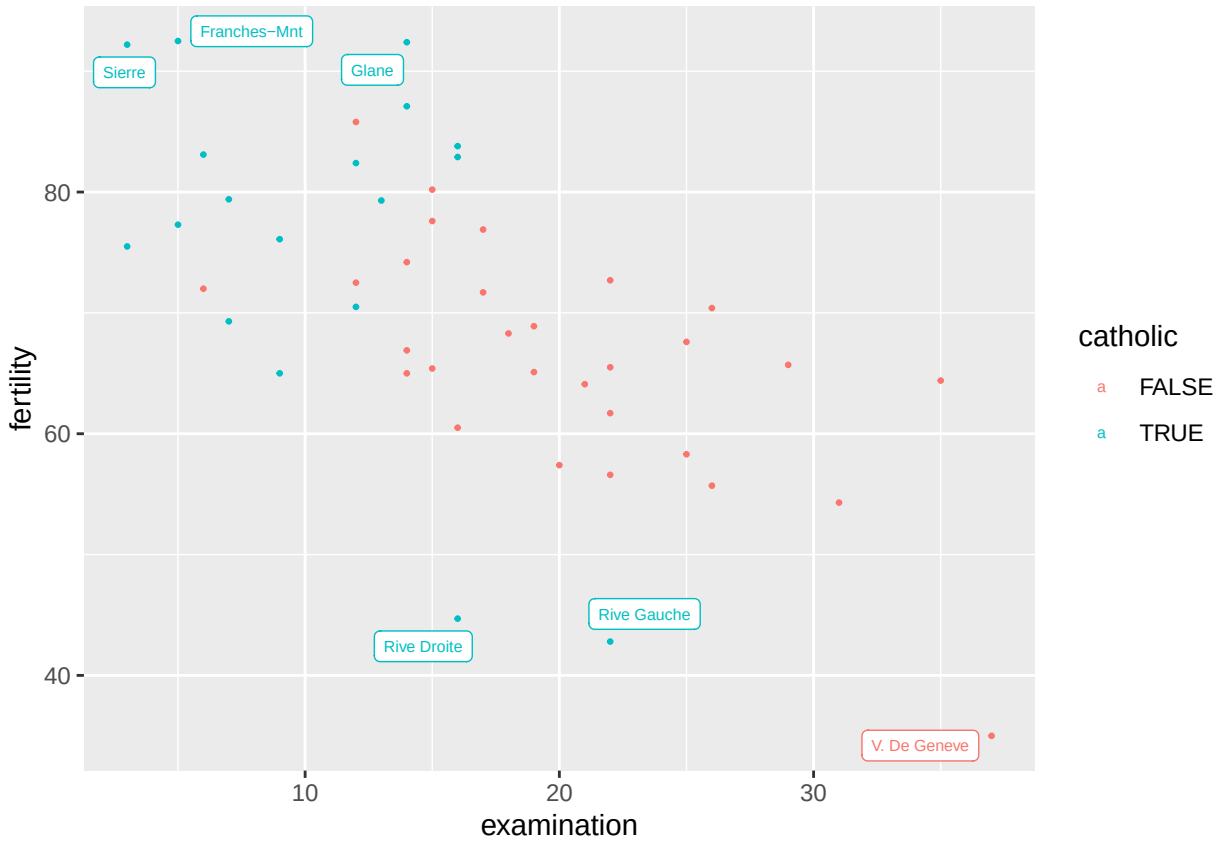



```
# Use geom_text_repel from ggrepel package to can minimize or overcome overlapping labels on plot
swiss_df %>%
  ggplot(mapping = aes(x = examination, y = fertility,
                       label = province, color = catholic)) +
  geom_point(size = 0.5) +
  geom_text_repel(size = 2, segment.alpha = 0.5)
```

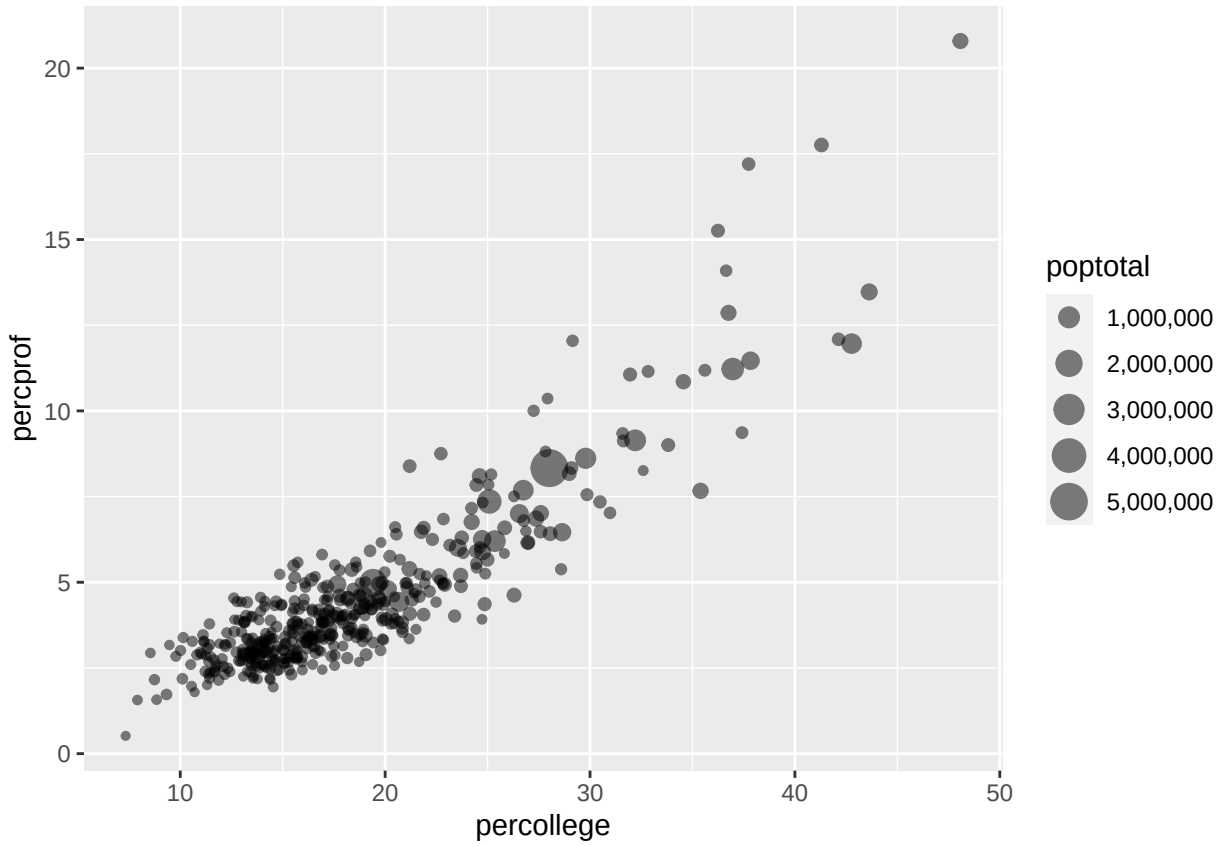

```
# segment.alpha = 0.5 controls transparency of lines connecting lables to points

# Labeling all of our points can get really messy. We can label specific points
# Let's label only thos provinces where fertility rates were very high or very low
swiss_df %>%
  mutate(extreme = ifelse(fertility < 50 | fertility > 90, province, '')) %>%
  ggplot(mapping = aes(x = examination, y = fertility,
                       label = extreme, color = catholic)) +
  geom_point(size = 0.5) +
  geom_text_repel(size = 2)
```



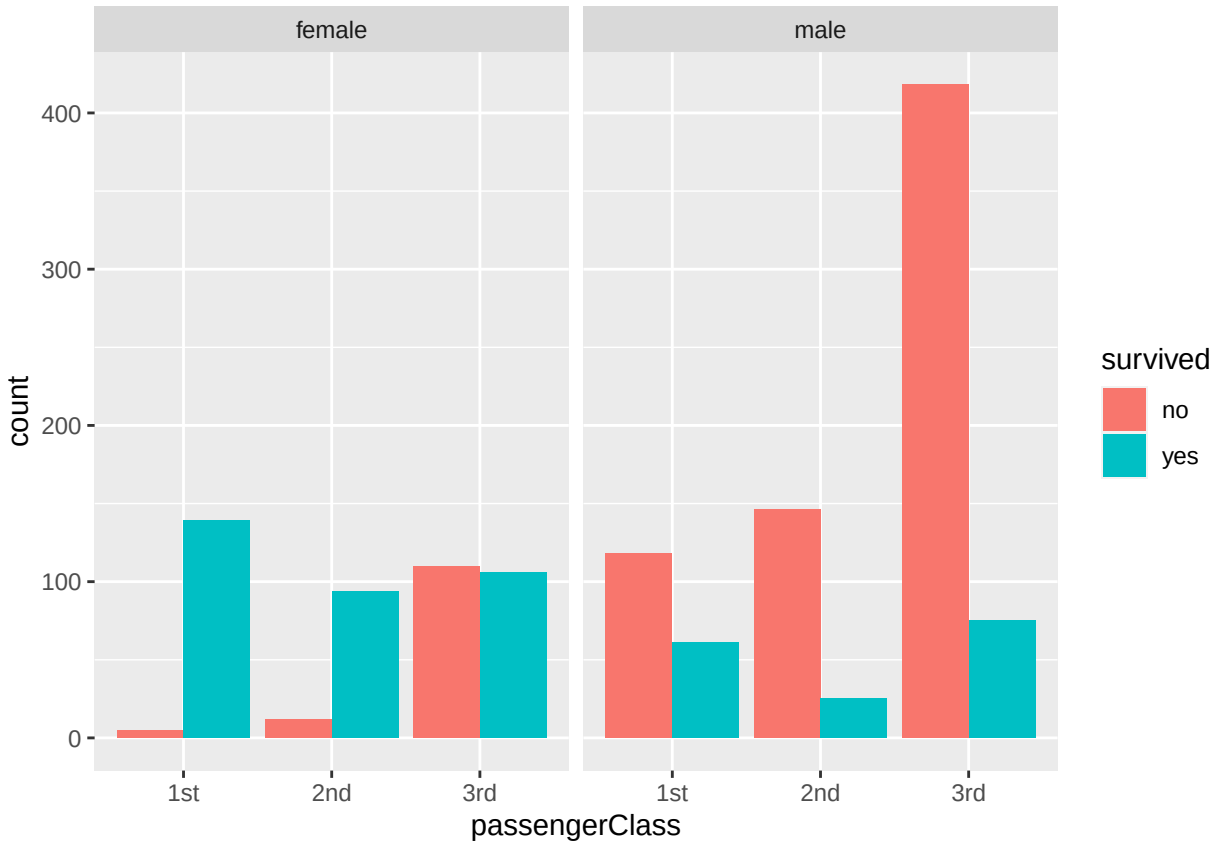

```
# mutate(extreme = ifelse(fertility < 50 | fertility > 90, province, '')) adds new variable extreme whe

# We can also use geom_label or geom_label_repel
swiss_df %>%
  mutate(extreme = ifelse(fertility < 50 | fertility > 90, province, '')) %>%
  ggplot(mapping = aes(x = examination, y = fertility,
                       label = extreme, color = catholic)) +
  geom_point(size = 0.5) +
  geom_label_repel(size = 2)
```




```
# Bubbleplots
# Bubbleplots are scatterplots wehre the size of the point is determind by the value of the third varia
# Use midwest data set from ggplot2
midwest_df = midwest
midwest_df %>%
  ggplot(mapping = aes(x = percollege, y = percprof, size = poptotal)) +
  geom_point(alpha = 0.5) +
  scale_size_continuous(labels = scales::comma_format(scale = 1))
```




```
# Used scale_size_continuous function to change legend format

# Facet plots
# All you to produce multiple related subplots, where each subplot displays some subset of data
ggplot(titanic_df,
       mapping = aes(x = passengerClass, fill = survived)) +
  geom_bar(position = 'dodge') +
  facet_wrap(~sex)
```

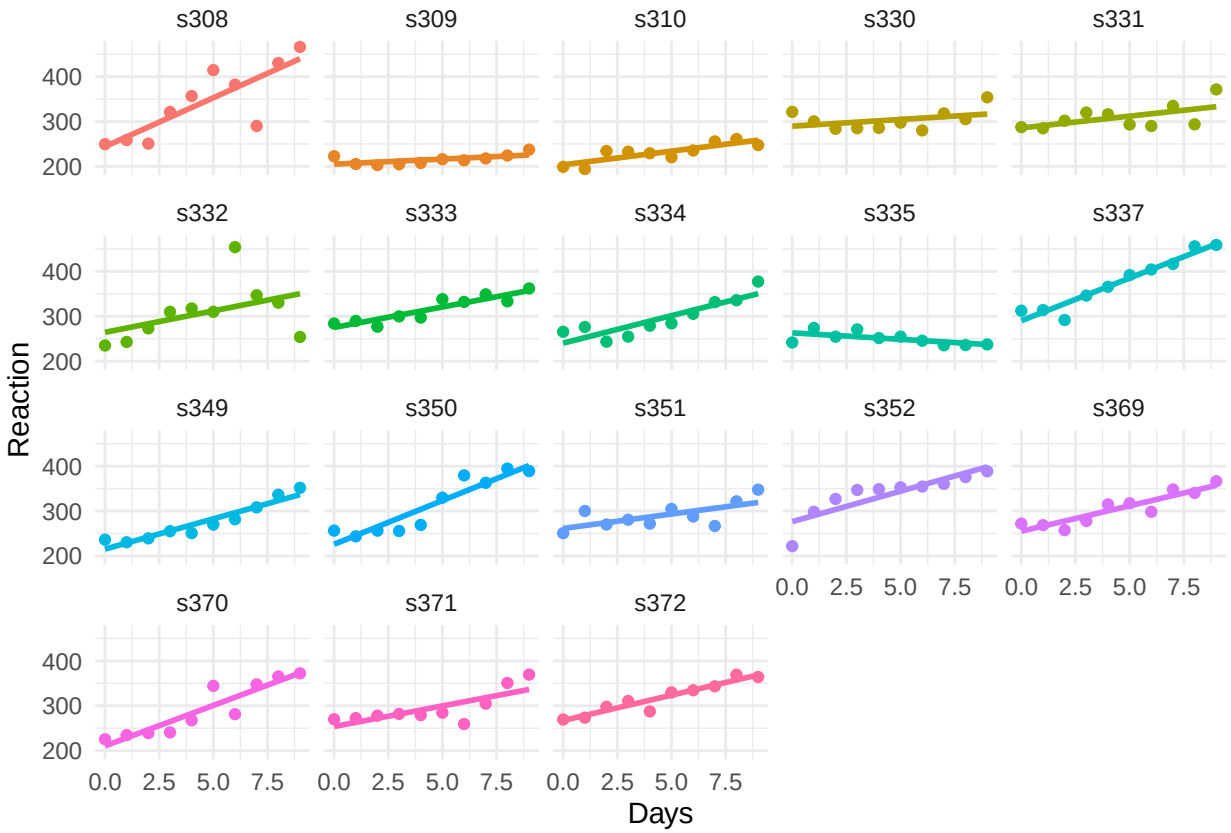



```
# When there is multiple subplots, facet_wrap will wrap the subplots
# Produce one scatter plot from sleepstudy.csv with line of best fit for each of the 18 subjects in the
sleepstudy_df = read_csv('week_2/data/sleepstudy.csv')
```

```
## Rows: 180 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Subject
## dbl (2): Reaction, Days
##
## 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.
```

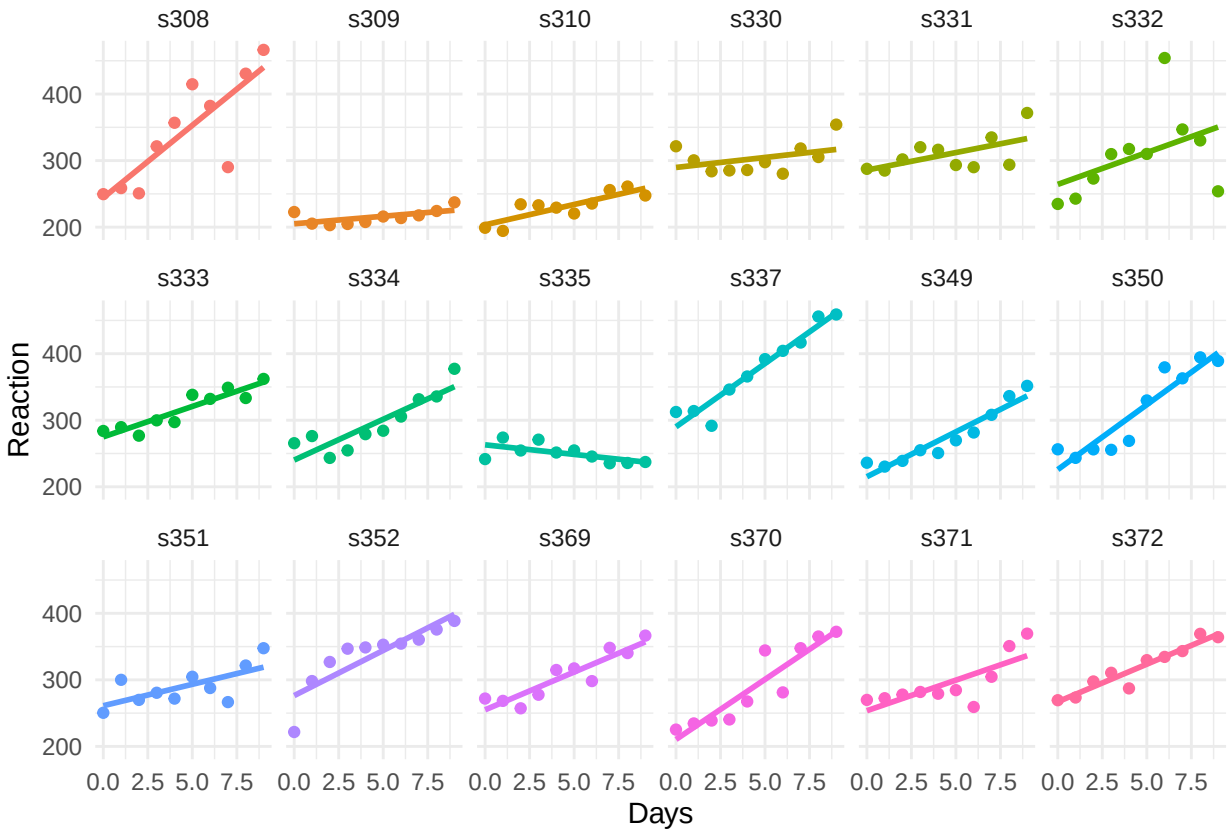
```
ggplot(sleepstudy_df,
       mapping = aes(x = Days, y = Reaction, color = Subject)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F) +
  facet_wrap(~Subject) +
  theme_minimal() +
  theme(legend.position = 'none')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# theme_minimal() sets plots theme to minimal style
# use nrow to specify number of rows to use in facet_wrap
ggplot(sleepstudy_df,
       mapping = aes(x = Days, y = Reaction, color = Subject)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F) +
  facet_wrap(~Subject, nrow = 3) +
  theme_minimal() +
  theme(legend.position = 'none')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Chapter 5 HW: Perform some univariate exploratory analyses. For example, from one or more variables of interest plot their histograms and boxplots, both overall, and when the variable is grouped according to values of another variable. In parallel, calculate summary statistics measures of central tendency, dispersion, skewness, and kurtosis. Compare the plots to the tables of quantities to be able to get a grasp on how certain summaries of the data manifest themselves visualization, and how certain properties of the plots manifest themselves in summary statistics. For example, see whether histograms with long tails correspond to relatively high values of skewness, and vice versa.

```
# Let's perform univariate exploratory analyses on car_prices_df
car_prices_df = car_prices_df %>%
  rename_all(tolower)
summarize(car_prices_df,
          avg_price = mean(price, na.rm = TRUE), # mean
          med_price = median(price, na.rm = TRUE), # median
          sd_price = sd(price, na.rm = TRUE), # standard deviation
          var_price = var(price, na.rm = TRUE), # variance
          mad_price = mad(price, na.rm = TRUE) # median absolute deviation
          )
```

```
## # A tibble: 1 x 5
##   avg_price med_price sd_price var_price mad_price
##       <dbl>     <dbl>    <dbl>     <dbl>     <dbl>
## 1      18.6      16.3     7.82      61.1      5.56
```

```
# Run same summary on prices per vehicle type
car_prices_by_type = group_by(car_prices_df, type)
summarize(car_prices_by_type,
```

```
          avg_price = mean(price, na.rm = TRUE),
          med_price = median(price, na.rm = TRUE),
          sd_price = sd(price, na.rm = TRUE),
          var_price = var(price, na.rm = TRUE),
          mad_price = mad(price, na.rm = TRUE)
          )
```

```
## # A tibble: 6 x 6
##   type    avg_price med_price sd_price var_price mad_price
##   <chr>       <dbl>     <dbl>    <dbl>     <dbl>     <dbl>
## 1 Compact      12.8      13.3     1.69      2.87      2.82
## 2 Large        24.3      20.9     6.34     40.2       3.71
## 3 Midsize      21.8      17.4     8.90     79.2       3.19
## 4 Small        10.0      10.1     1.64      2.69      1.63
## 5 Sporty       19.4      15.5     8.48     71.8       2.15
## 6 Van          18.3      19       1.69      2.84      1.33
```

```
# Range of price
max(car_prices_df$price) - min(car_prices_df$price)
```

```
## [1] 32.7
```

```
# Now by vehicle type
summarize(car_prices_by_type,
          range_price = max(price) - min(price))
```

```
## # A tibble: 6 x 2
##   type    range_price
##   <chr>         <dbl>
## 1 Compact        4.7
## 2 Large         17.7
## 3 Midsize       25.2
## 4 Small          4.8
## 5 Sporty        24
## 6 Van            3.60
```

```
# Quantile range for price
quant_range_price = function(x, lower, upper){
  quantile(x, probs = c(lower, upper)) %>%
    unname() %>%
    diff()
}
# 100% inner range
quant_range_price(car_prices_df$price, lower = 0.0, upper = 1.0)
```

```
## [1] 32.7
```

```
# 90% inner range
quant_range_price(car_prices_df$price, lower = 0.05, upper = 0.95)
```

```
## [1] 26.095
```

```r
# 80% inner range
quant_range_price(car_prices_df$price, lower = 0.1, upper = 0.9)
```

```
## [1] 19.84
```

```r
# Interquantile range
quant_range_price(car_prices_df$price, lower = 0.25, upper = 0.75)
```

```
## [1] 7.25
```

```r
# OR you can do this code
IQR(car_prices_df$price)
```

```
## [1] 7.25
```

```r
# Skewness
library(psych)
psych::skew(car_prices_df$price)
```

```
## [1] 1.153757
```

```r
# Skewness of price by vehicle type
summarize(car_prices_by_type,
          psych::skew(car_prices_by_type$price))
```

```
## # A tibble: 6 x 2
##   type    `psych::skew(car_prices_by_type$price)`
##   <chr>                                     <dbl>
## 1 Compact                                    1.15
## 2 Large                                      1.15
## 3 Midsize                                    1.15
## 4 Small                                      1.15
## 5 Sporty                                     1.15
## 6 Van                                        1.15
```

```r
# Quantile skewness
qskewness = function(x, p = 0.25){
  Q = quantile(x, probs = c(p, 0.5, 1 - p)) %>%
    unname()
  Q_1 = Q[1]; m = Q[2]; Q_u = Q[3]
  ((Q_u - m) - (m - Q_1)) / (Q_u - Q_1)
}
# Quantile skew
qskewness(car_prices_df$price)
```

```
## [1] 0.2206897
```

```r
# Octile skew
qskewness(car_prices_df$price, p = 1/8)
```

```
## [1] 0.3484603
```

```r
# Decile skew
qskewness(car_prices_df$price, p = 1/10)
```

```
## [1] 0.4758065
```

```r
# Quantile skew by type
summarize(car_prices_by_type,
          qskewness(car_prices_by_type$price))
```

```
## # A tibble: 6 x 2
##   type    `qskewness(car_prices_by_type$price)`
##   <chr>                                   <dbl>
## 1 Compact                                 0.221
## 2 Large                                   0.221
## 3 Midsize                                 0.221
## 4 Small                                   0.221
## 5 Sporty                                  0.221
## 6 Van                                     0.221
```

```r
# Nonparametric skew
npskew = function(x){
  (mean(x) - median(x)) / sd(x)
}
npskew(car_prices_df$price)
```

```
## [1] 0.2907691
```

```r
# npskew by type
summarize(car_prices_by_type,
          npskew(car_prices_by_type$price))
```

```
## # A tibble: 6 x 2
##   type    `npskew(car_prices_by_type$price)`
##   <chr>                                <dbl>
## 1 Compact                              0.291
## 2 Large                                0.291
## 3 Midsize                              0.291
## 4 Small                                0.291
## 5 Sporty                               0.291
## 6 Van                                  0.291
```

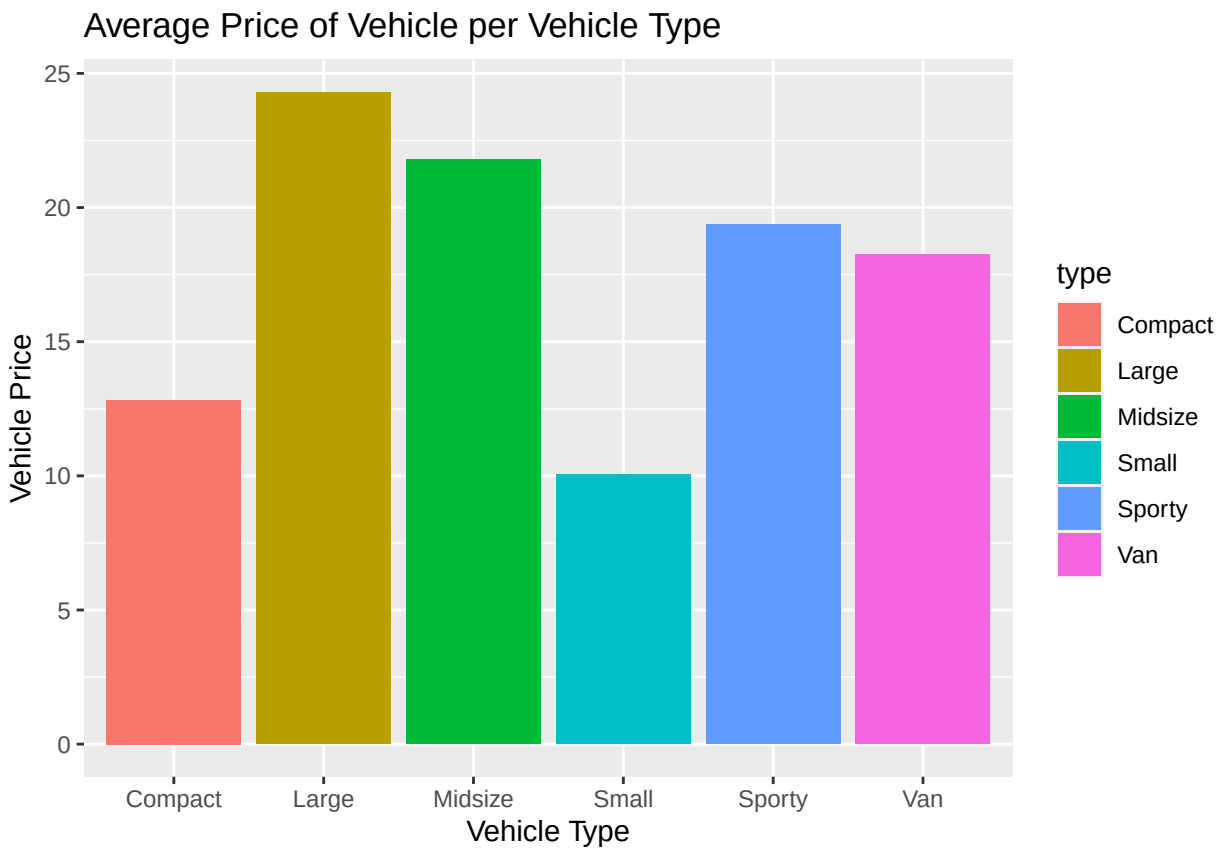
```r
# Kurtosis
kurtosis = function(x){
  z = (x - mean(x)) / sd(x)
  mean(z^4)
}
kurtosis(car_prices_df$price)
```

```
## [1] 3.681784
```

```
# Kurtosis by type
summarize(car_prices_by_type,
          kurtosis(car_prices_by_type$price))
```

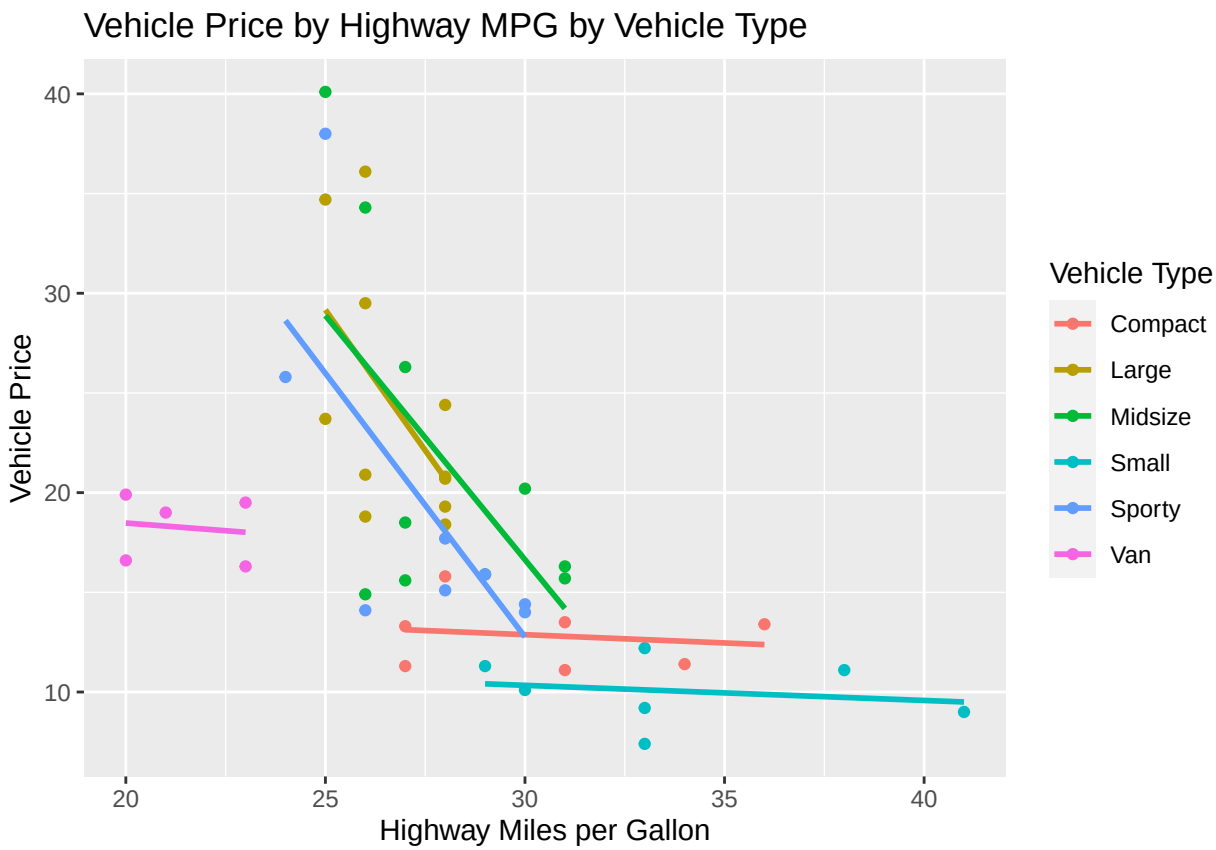
```
## # A tibble: 6 x 2
##   type    `kurtosis(car_prices_by_type$price)`
##   <chr>                                  <dbl>
## 1 Compact                                 3.68
## 2 Large                                   3.68
## 3 Midsize                                 3.68
## 4 Small                                   3.68
## 5 Sporty                                  3.68
## 6 Van                                     3.68
```

```
# Graphical exploration
# Histogram
# Avg price per car per type
ggplot(car_prices_df,
       mapping = aes(x = type, y = price, fill = type)) +
  geom_bar(stat = 'summary', position = 'dodge', fun = 'mean') +
  labs(x = 'Vehicle Type',
       y = 'Vehicle Price',
       title = 'Average Price of Vehicle per Vehicle Type')
```



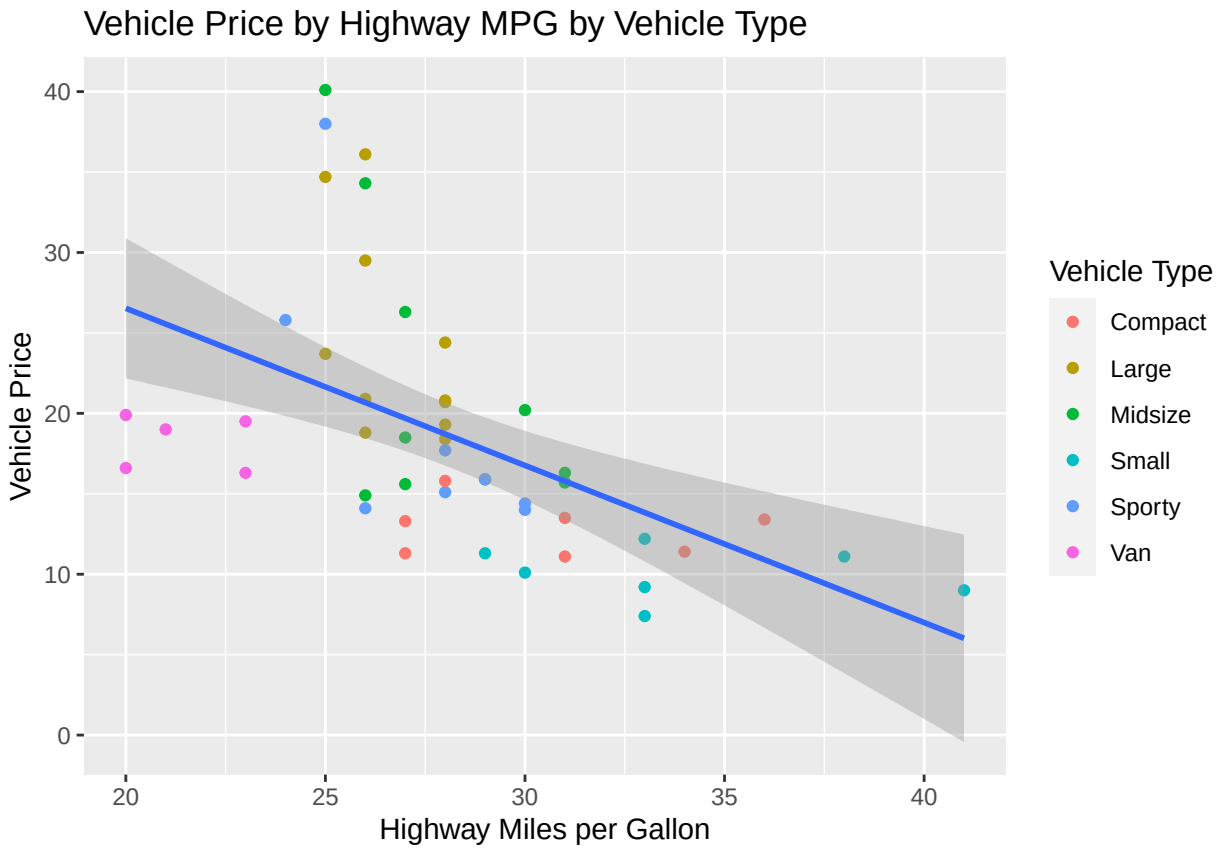
```
# Scatter plot of price of vehicle by mpg.highway by type. Fit linear model to each vehicle type
ggplot(car_prices_df,
       mapping = aes(x = mpg.highway, y = price, color = type)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F) +
  labs(x = 'Highway Miles per Gallon',
       y = 'Vehicle Price',
       title = 'Vehicle Price by Highway MPG by Vehicle Type',
       color = 'Vehicle Type')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



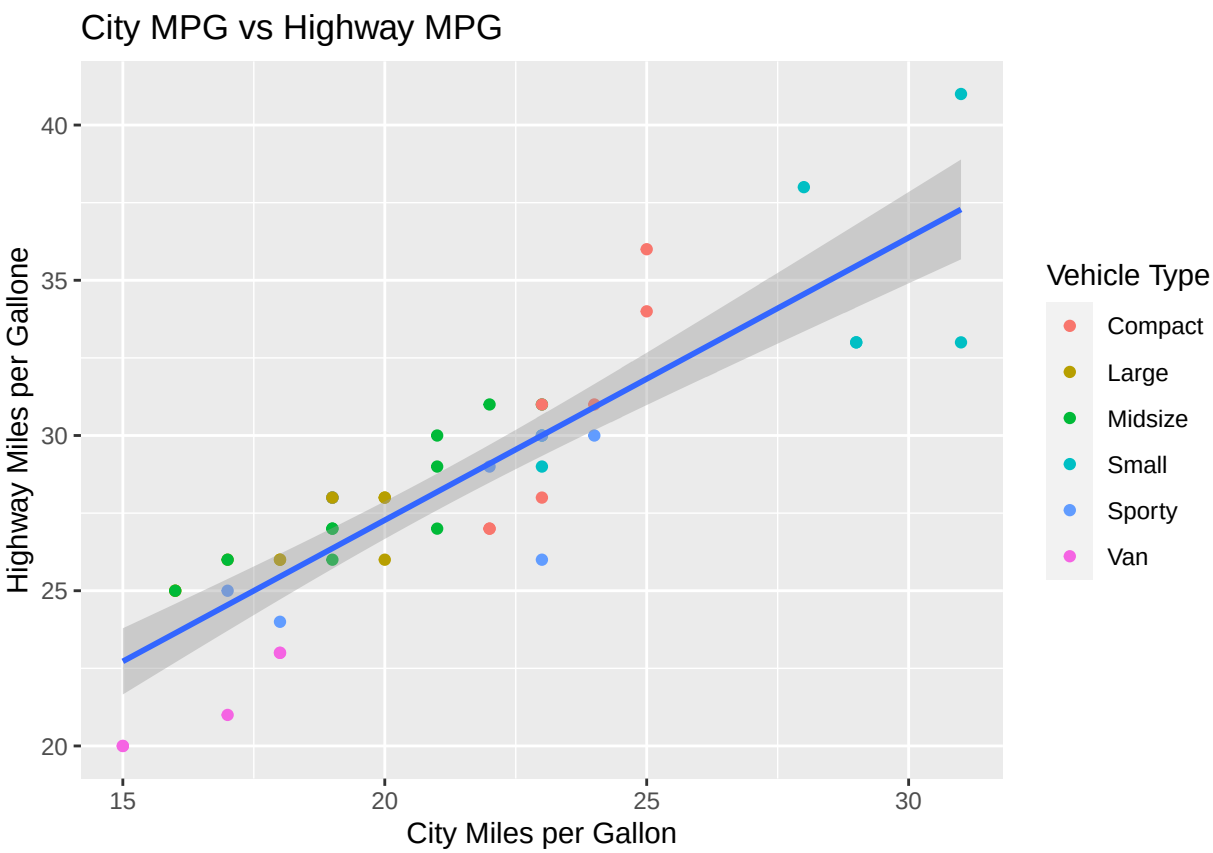

```
# Now with single best fit line for the entire data set
ggplot(car_prices_df,
       mapping = aes(x = mpg.highway, y = price)) +
  geom_point(mapping = aes(color = type)) +
  geom_smooth(method = 'lm', fullrange = T) +
  labs(x = 'Highway Miles per Gallon',
       y = 'Vehicle Price',
       title = 'Vehicle Price by Highway MPG by Vehicle Type',
       color = 'Vehicle Type')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Scatter plot of mpg.city and mpg.highway
ggplot(car_prices_df,
       mapping = aes(x = mpg.city, y = mpg.highway)) +
  geom_point(mapping = aes(color = type)) +
  geom_smooth(method = 'lm', fullrange = T) +
  labs(x = 'City Miles per Gallon',
       y = 'Highway Miles per Gallone',
       title = 'City MPG vs Highway MPG',
       color = 'Vehicle Type')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```


City MPG vs Highway MPG
Highway Miles per Gallone
40
35
30
25
20
15
20
25
30
City Miles per Gallon
Vehicle Type
Compact
Large
Midsize
Small
Sporty
Van