Workshop #3: dplyr & ggplot

Lucy Liu

Introduction

We are going to learn about 2 packages in R called dplyr and ggplot. We call R by itself 'base R' and it comes with a lot of functions. You can also add to these functions by installing packages - you can think of this as like installing apps on your phone - it adds extra functions.

The dplyr package gives you functions for manipulating your data - like what we did last week with square brackets where we subsetted part of our data. The ggplot package gives you functions for making plots and it is one of the most popular packages! They were both written by the same person and they work well together.

Shared doc

We are going to use this shared google doc for today. It can be found at the url. First, if you don't see this outline panel, click View > show document outline.

All the notes for what I am covering today can be found here. I am not going to cover everything here but if you are interested you can go through these in your own time.

Installation and data import

To start off with we are going to install these 2 packages and import the data that we will play with today.

Installation

There are 2 steps to installing a package.

This command install.packages downloads the package onto your computer (if you are using the server, you don't need to do this as it would already have been done). You only need to do this once.

The command library 'opens' this package up and you need to do this every time you open up R. People using the server you will need to use this command.

Data import

The data that we are using today can be found here. First, let's take a look at it - what symbol is separating each data value?? What command can you use to read this in (there are several options).

So I want you to save the data as a text file - give it a name - and put it in the folder where you have been putting all your workshop things. Remember this folder because we are also going to set this folder as our working directory.

Does anyone remember how to do this?

Install the packages and save and read in the data and save as a variable called 'patients'. ***

```
patients <- read.csv("patients.txt")
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(ggplot2)
```

We can now have a look at our data. It is a dataframe (recall a dataframe is a data structure - a container for our dat. It is 2D - it has row and columns and just like an excel file)

DPLYR

A vital function in dplyr is called the 'pipe' %>% You can think about this as a pipe, that sends data through to the next function. We'll use the function select to demonstrate this.

Select

The select() function let's you select columns in your data.

The dplyr function is fairly intuitive. What do you think happens when I do this?

```
patients %>%
  select(Name)
```

What happening here, is that we start with our data, which we have 'saved' as a variable called patients, we are piping the data to the first function called select which selects columns.

What if I did this?

```
patients %>%
  select(Name)%>%
  head()
```

What if I did this?

```
patients %>%
  select(-Name) %>%
  head()
```

What if I did this?

```
patients %>%
  select(Name:Sex) %>%
  head()
```

What if I did this?

```
patients %>%
  select(ends_with("t")) %>%
  head()
```

Challenge ***

Filter

filter() function let's you filter for certain rows.

```
patients %>%
  filter(Sex == 'Male') %>%
  head()
```

What happens if I do this?

```
patients %>%
  filter(Name != 'Michael') %>%
  head()
```

What if I do this?

```
patients %>%
  filter(Sex == 'Male', Pet == 'Dog') %>%
  head()
```

You can also use & here. It is exactly the same as a comma.

What if I do this?

```
patients %>%
  filter(Height > '170' | Pet == 'Dog') %>%
  head()
```

Does anyone know what | means? Does everyone know where | is?

What if I do this?

```
patients %>%
  filter(Pet %in% c("Dog", "Bird")) %>%
  head()
```

The %in% function finds matches. So it is looks for all the rows where under Pets, is Dog or bird.

Recall we used '==' earlier. The difference is that you use '==' when you are looking for 1 thing, e.g. just rows with dog. Use %in% when you are looking for more than one thing, e.g. rows with dog and bird

The reason is a bit complicated but if you are interested you can ask one of our lovely helpers in the break.

Now what if I did this?

```
patients %>%
  filter(Pet %in% c("Dog", "Bird")) %>%
  select(Name:Sex) %>%
  head()
```

Mutate

We'll learn about 1 other function and then we'll go to a challenge.

Mutate creates new columns. You use it like this:

```r
patients %>%
  mutate(height_in_m = Height/100) %>%
  head()
```

I've created a new column. It's added to the end and the name of it is what I have given it here. I can change this and it would change the name of the column.

Challenge ***

Now you may have noticed something funny about the last challenge. R performs these functions in sequence - so it does this one first and then this.

In the last challenge if you selected just the columns name and sex, you end up with a dataframe that is just like this:

```r
patients %>%
  select(Name, Sex) %>%
  head()
```

You don't have the other columns so you can't can't filter for the rows where BMI is >25 say.

An easy way to fix this is to first filter the rows where BMI is >25 and smokes is TRUE and then to select the columns.

group_by & summarise

These 2 functions are really useful and you generally want to use them together. Let's first look at what summarise does by itself.

```r
patients %>%
  summarise(mean_height = mean(Height))
```

It's taken all the heights and given us the mean.

Now let's have a look at what it does when you use it with group_by.

```r
patients %>%
  group_by(Sex) %>%
  summarise(mean_height = mean(Height))
```

So it's group the rows into the males and females, and then it has given us the mean height for each group.

What if I did this?

```r
patients %>%
  group_by(Sex, Race) %>%
  summarise(mean_height = mean(Height), mean_weight = mean(Weight))
```

Let's get on to some challenges but first I just want to show you something useful.

```r
a <- c(TRUE, FALSE, FALSE, TRUE, TRUE)
mean(a)
```

What has it done? Remember that Trues and falses are stored as 1's and 0's in R. 1 is TRUE and 0 is FALSE. So it is taking the average of Trues - 60% of these are TRUE's.

This will help you with the challenges.

Challenge ***

Add BMI to your data!

```
patients <- patients %>%
  mutate(BMI = Weight/(Height/100)^2)
```

BREAK ***

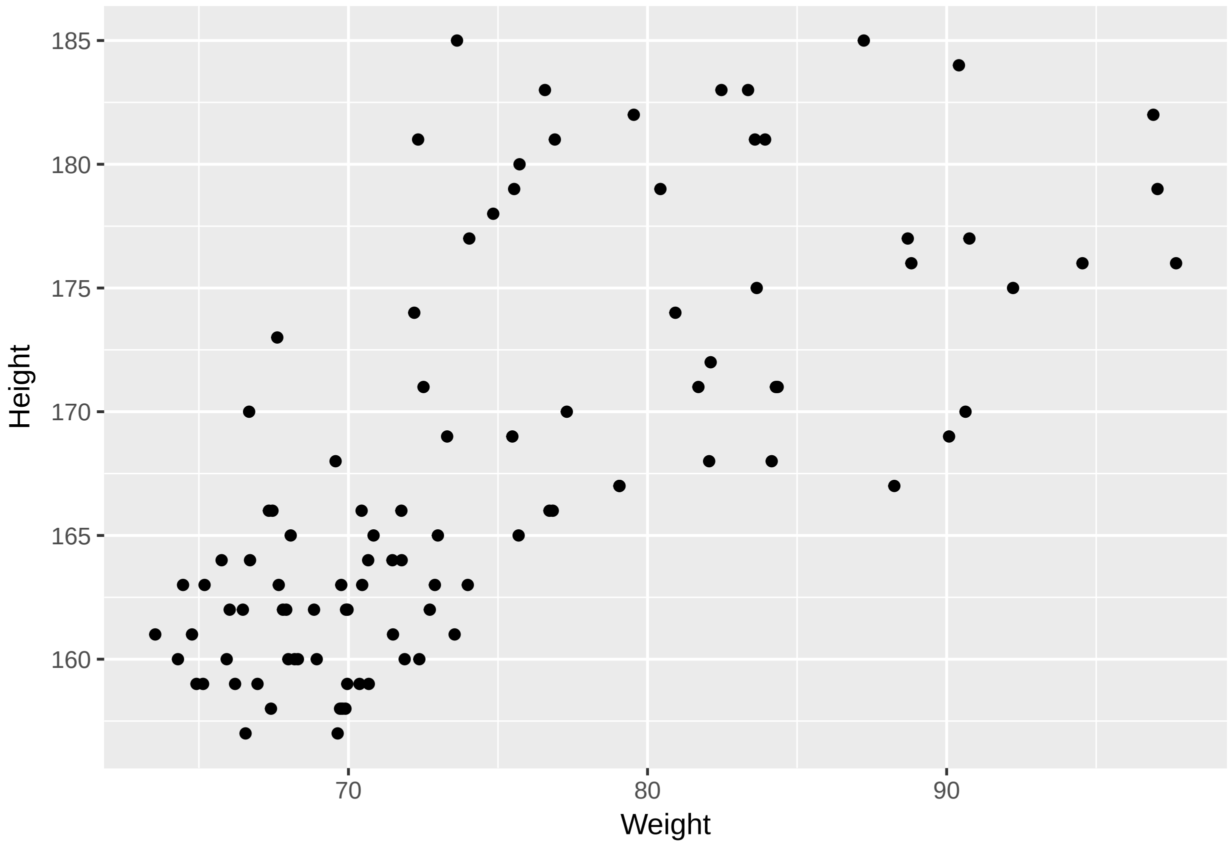
GGPLOT

ggplot is one of the most popular packages in R and you can make nice looking plots with it.

Let's start with the basic anatomy of ggplot code - this is how you give R the instructions to make your plot. The plus is important.

So if I wanted a scatter plot of height on y and weight on x - how would I do this?

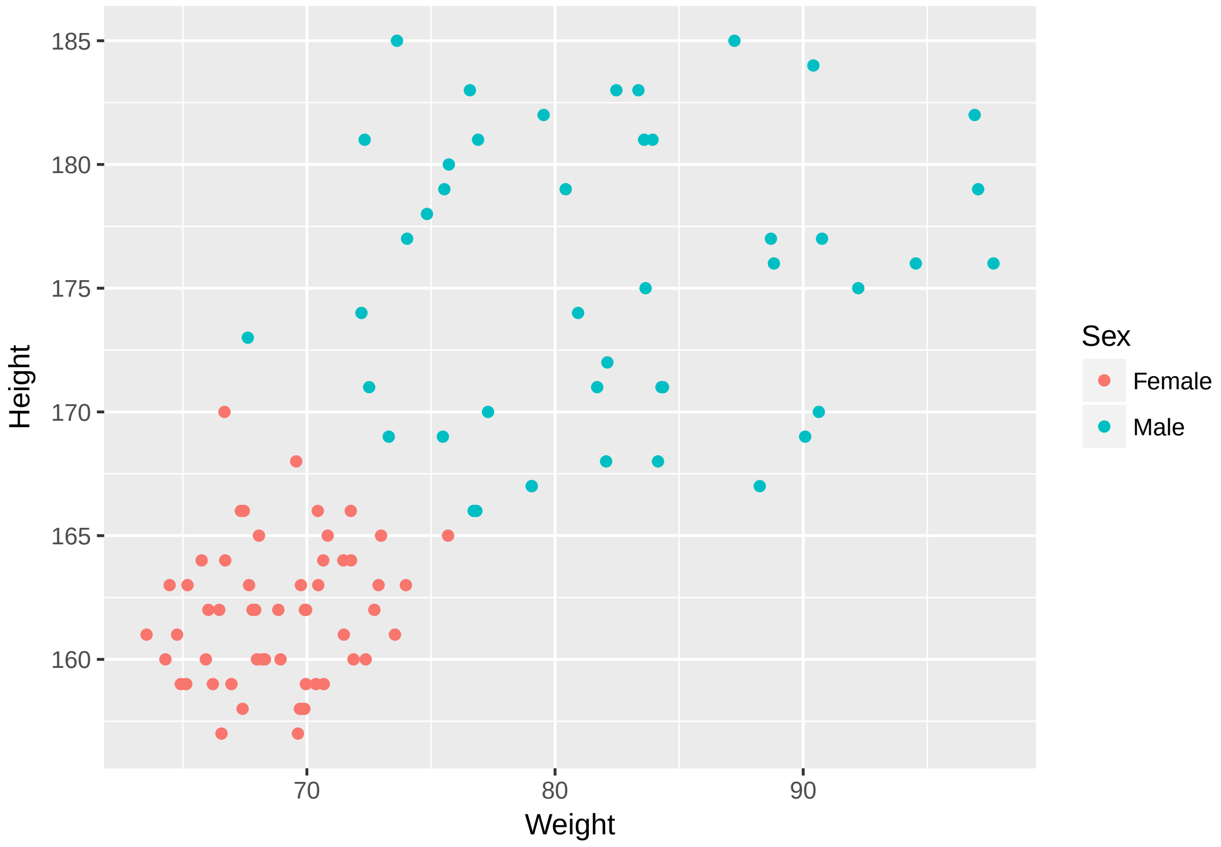
```
library(ggplot2)
ggplot(patients, aes(x=Weight, y=Height)) +
  geom_point()
```



Now there are a few other things that we can add to aes here - this means that we can add extra variables to our scatter plot. They are (show notes page) size, shape and colour.

What do you think will happen here?

```
ggplot(patients, aes(x=Weight, y=Height, colour = Sex)) +
  geom_point()
```



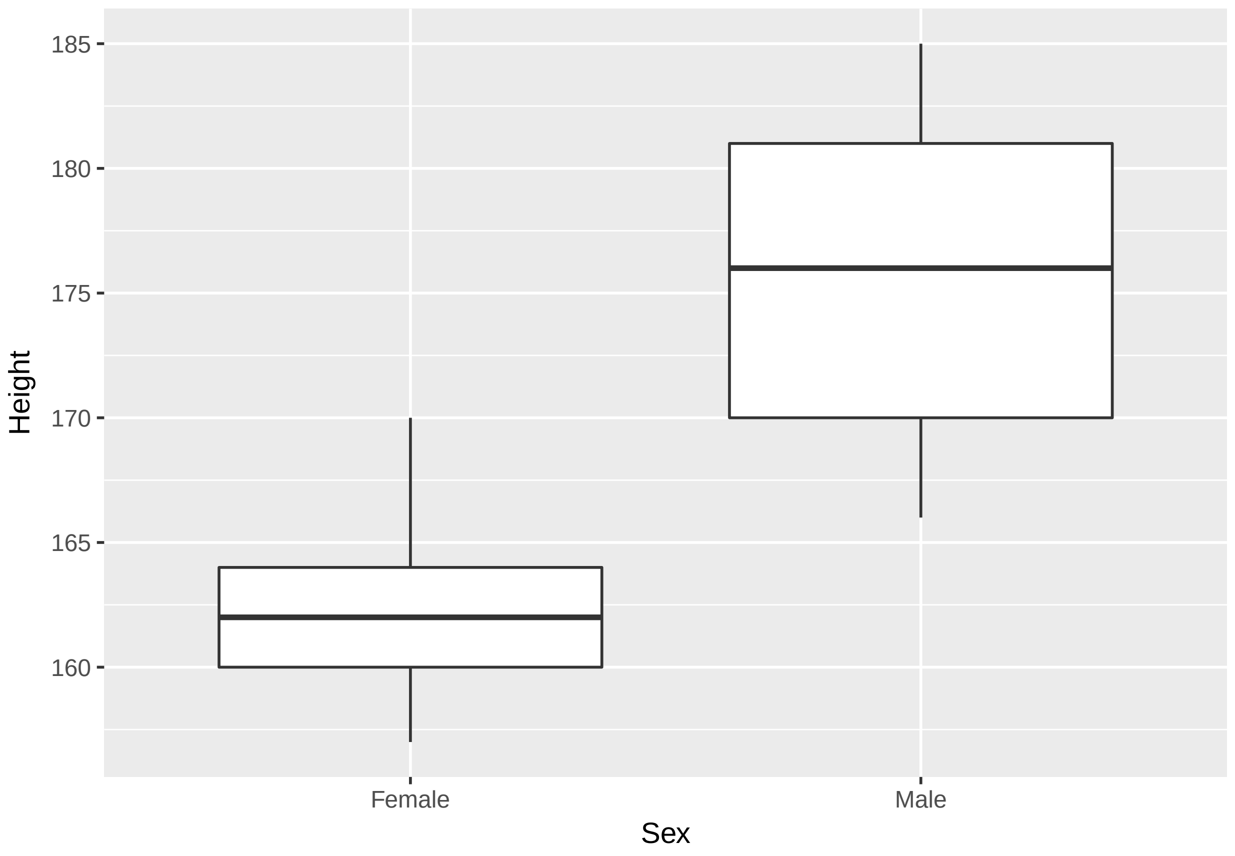
Challenge ***

Now let's have some fun with difference types of plots.

Boxplot

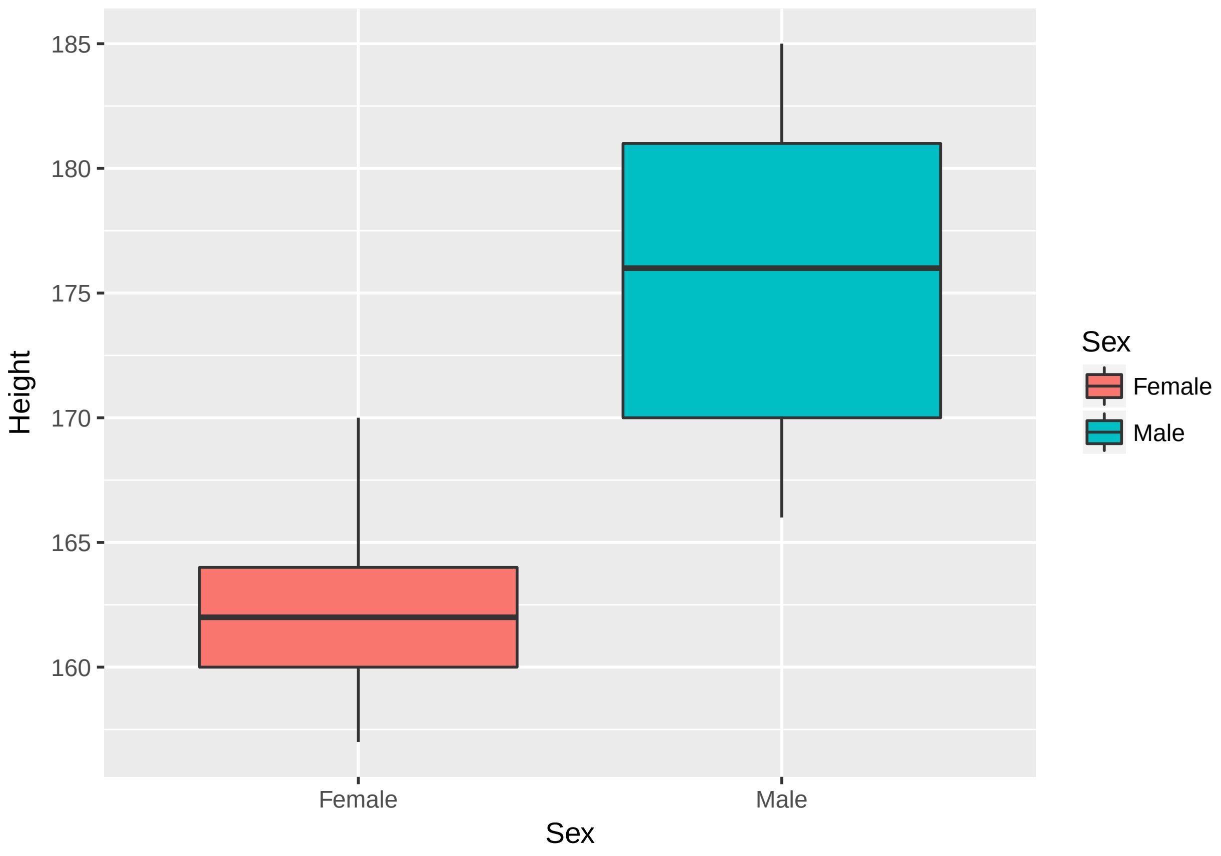
How would I make a boxplot of heights for each sex? (draw on whiteboard)

```
ggplot(patients, aes(y=Height, x= Sex)) +
  geom_boxplot()
```



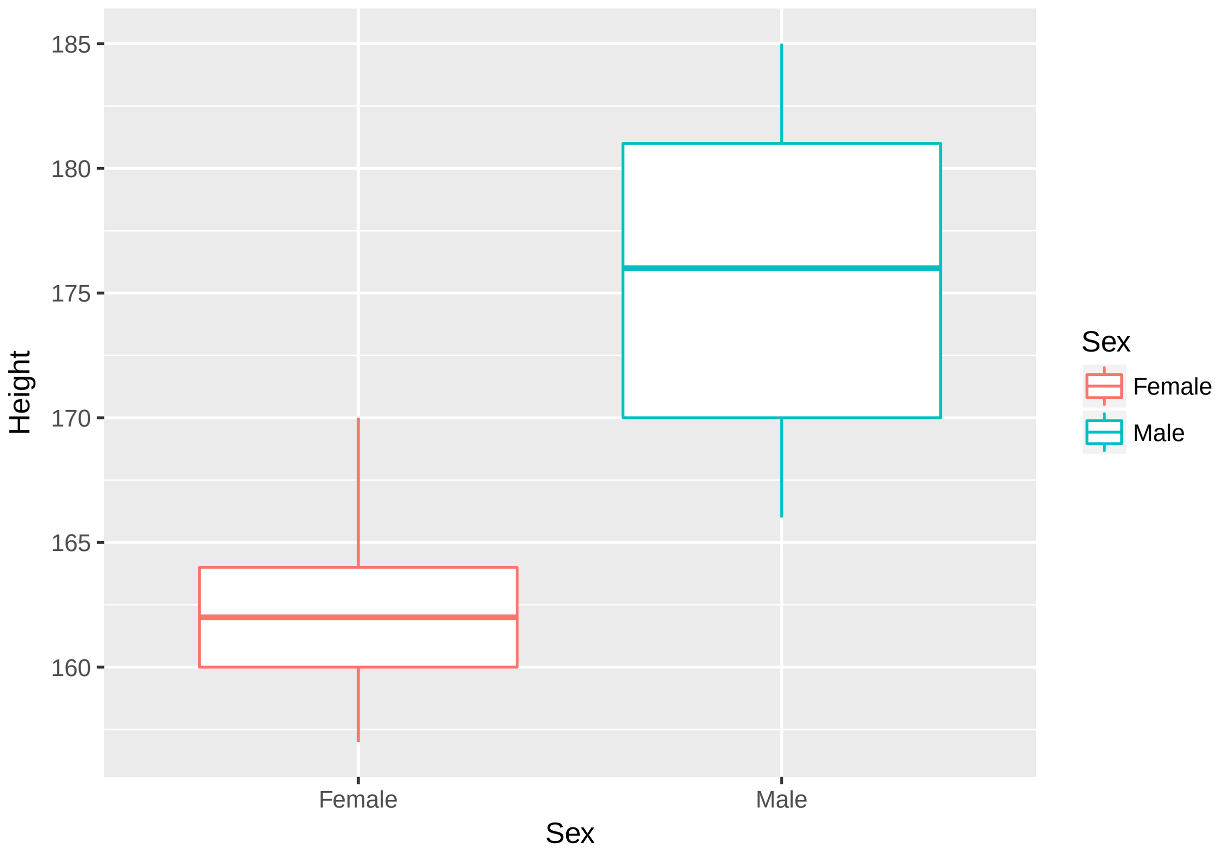
If I also wanted the male and female box to be different coloured, I use fill - as in fill these in different colours.

```
ggplot(patients, aes(y=Height, x= Sex, fill = Sex)) +
  geom_boxplot()
```



Why not use colour? Well weirdly, colour does something sligtly different.

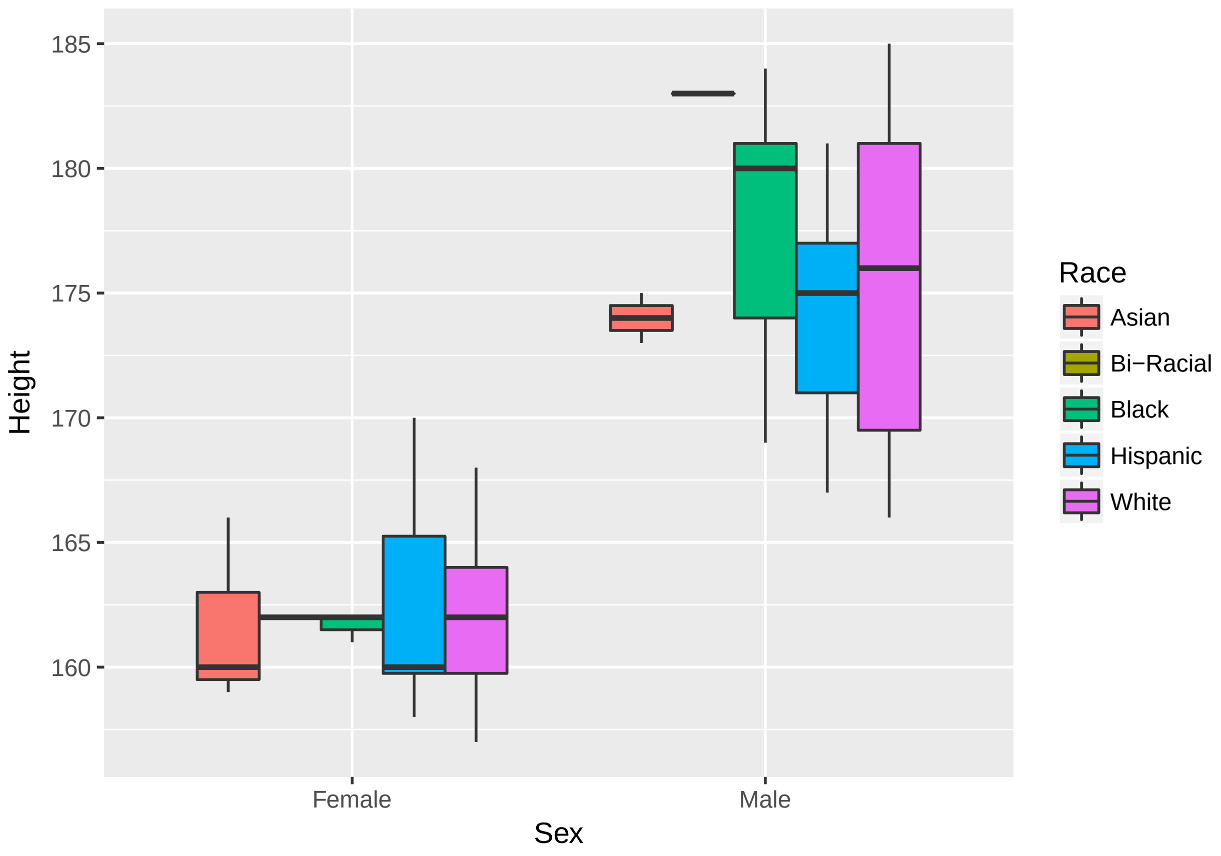
```
ggplot(patients, aes(y=Height, x= Sex, colour = Sex)) +
  geom_boxplot()
```



It actually colours in the border whereas fill - fills in on the inside.

You can also get a side-by-side plot using fill.

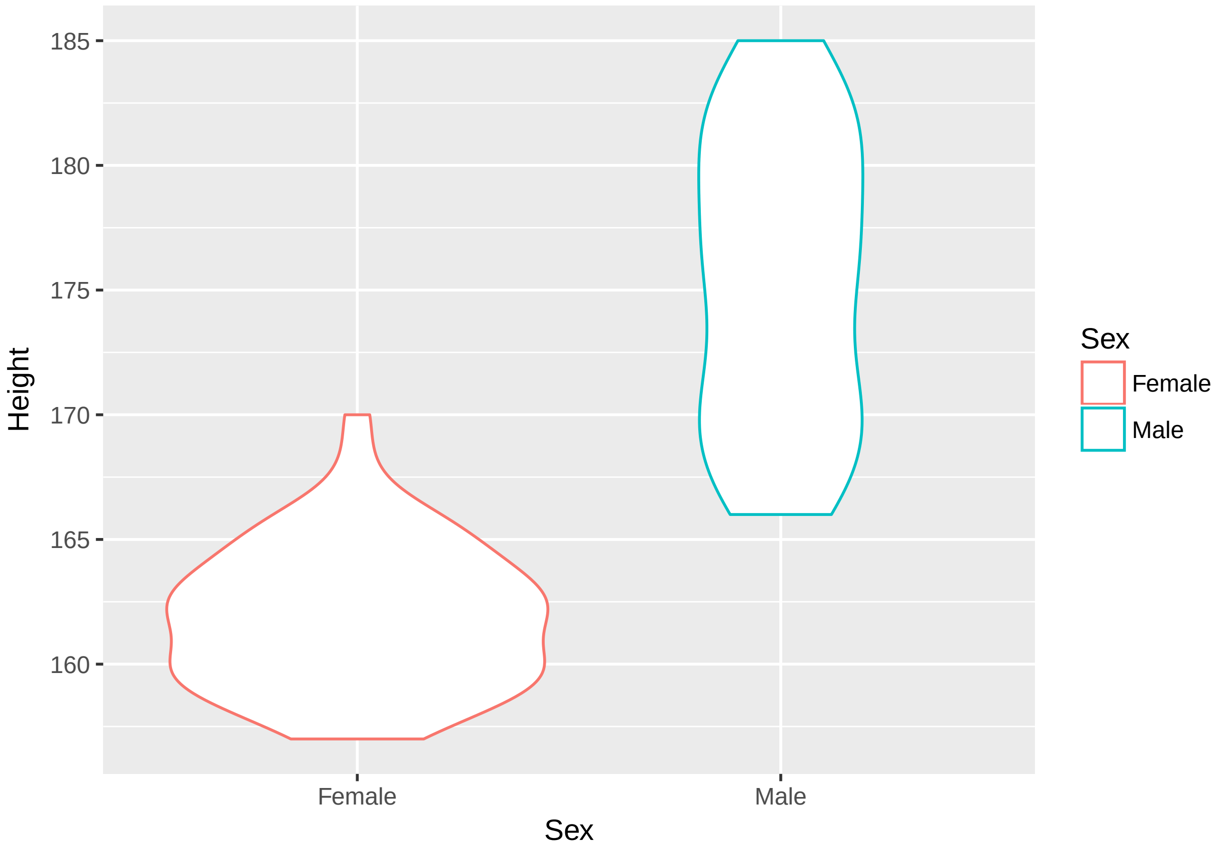
```
ggplot(patients, aes(y=Height, x= Sex, fill = Race)) +
  geom_boxplot()
```



Violin plot

A popular variation of the box plot is the violin plot. Instead of geom_boxplot, I put in geom_violin.

```
ggplot(patients, aes(y=Height, x= Sex, colour = Sex)) +
  geom_violin()
```

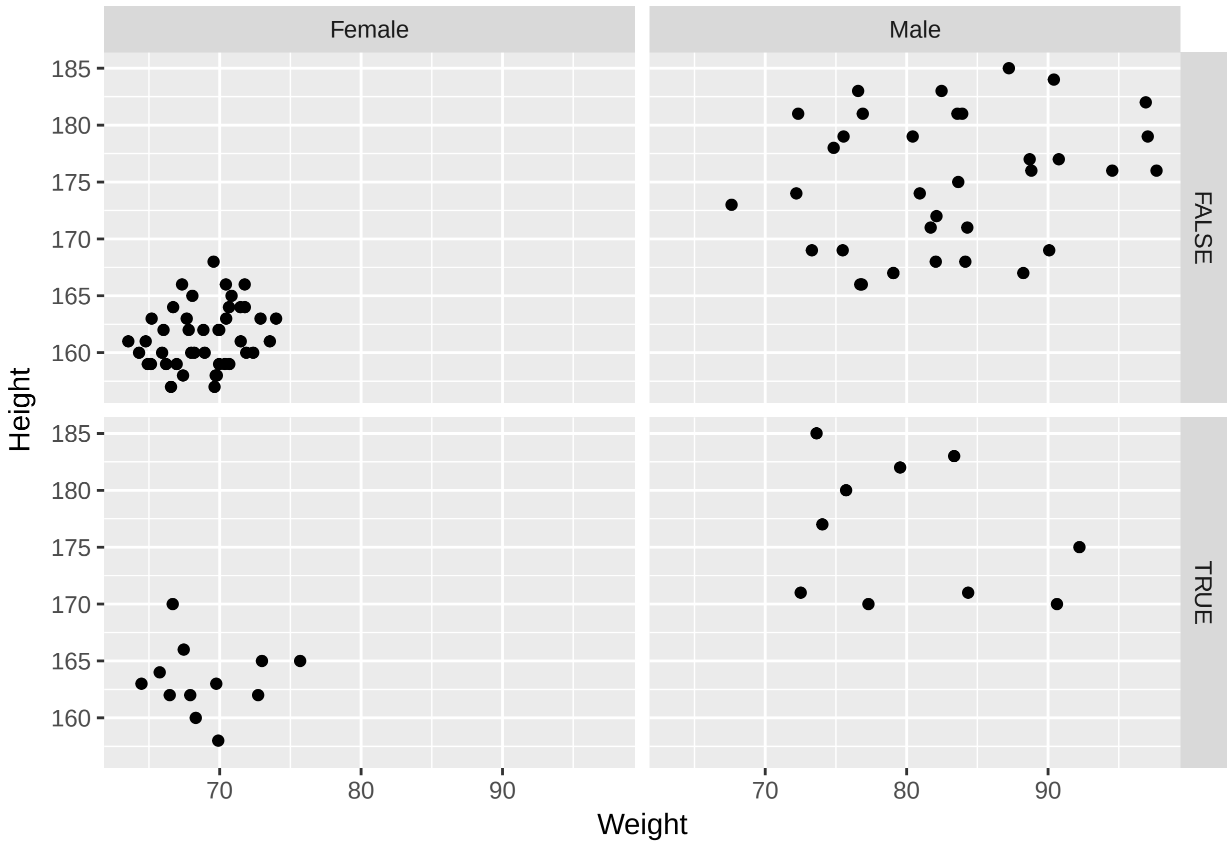


In a volin plot the width of the violin is proportion to the density of data - i.e. the width is small here thus there are few patients that are 170cm tall but here around 160 a lot of patients are this height.

Faceting

Faceting splits your graph into separate panels of graphs.

```r
ggplot(patients, aes(x= Weight, y = Height)) +
  geom_point() +
  facet_grid(Smokes~Sex)
```



What happens if I switch smokes and sex around?

Challenge ***

Bar plots

Bar plots are pretty popular.

I want a bar plot of mean heights for each race. What I need to do is to calculate the mean height for each race - I want a dataframe that looks like this.

We can do this using dplyr - does anyone remember how? What functions do we use?

```
patients %>%
  group_by(Race) %>%
  summarise(mean_height = mean(Height))
```

```
## # A tibble: 5 x 2
##   Race      mean_height
##   <fct>           <dbl>
## 1 Asian            167.
## 2 Bi-Racial        172.
## 3 Black            172.
## 4 Hispanic         169.
## 5 White            167.
```

Now I want to plot this on a bar plot. Remember that I said these 2 packages work well together? Well you can directly 'pipe' this data to ggplot! This actually makes your work less messy - you don't have to save it as a variable first - remember last week when we often ended up saving many many variables - patients_smokes, patients_females, patients_smokes_females.

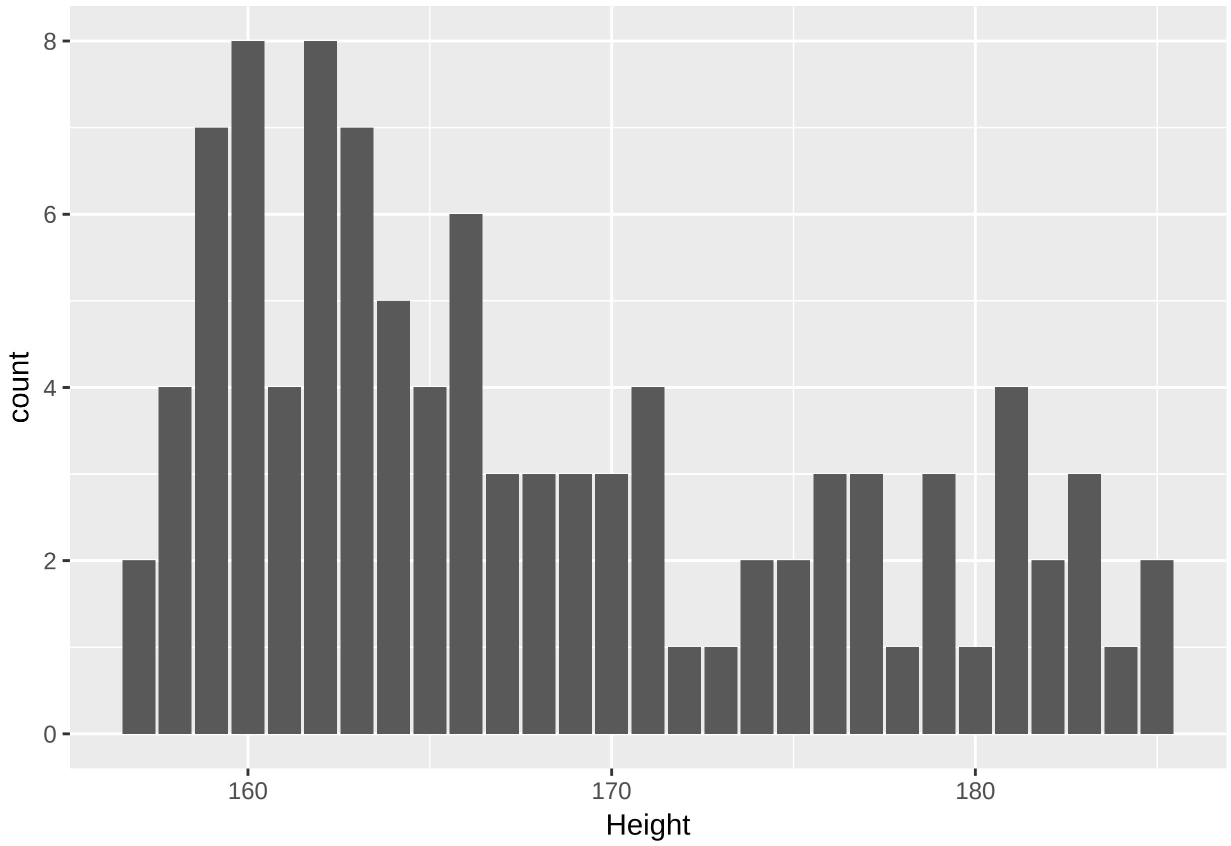
So now we don't have to tell ggplot what data do use - because the data is being piped in from here. The rest of the command is the same. How would I make a bar plot like this? Remember in dplyr we use the %>% and in ggplot we change to + - watch out for this!

```
patients %>%
  group_by(Race) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race)) +
  geom_bar()
```

We have an error! The message is pretty cryptic.

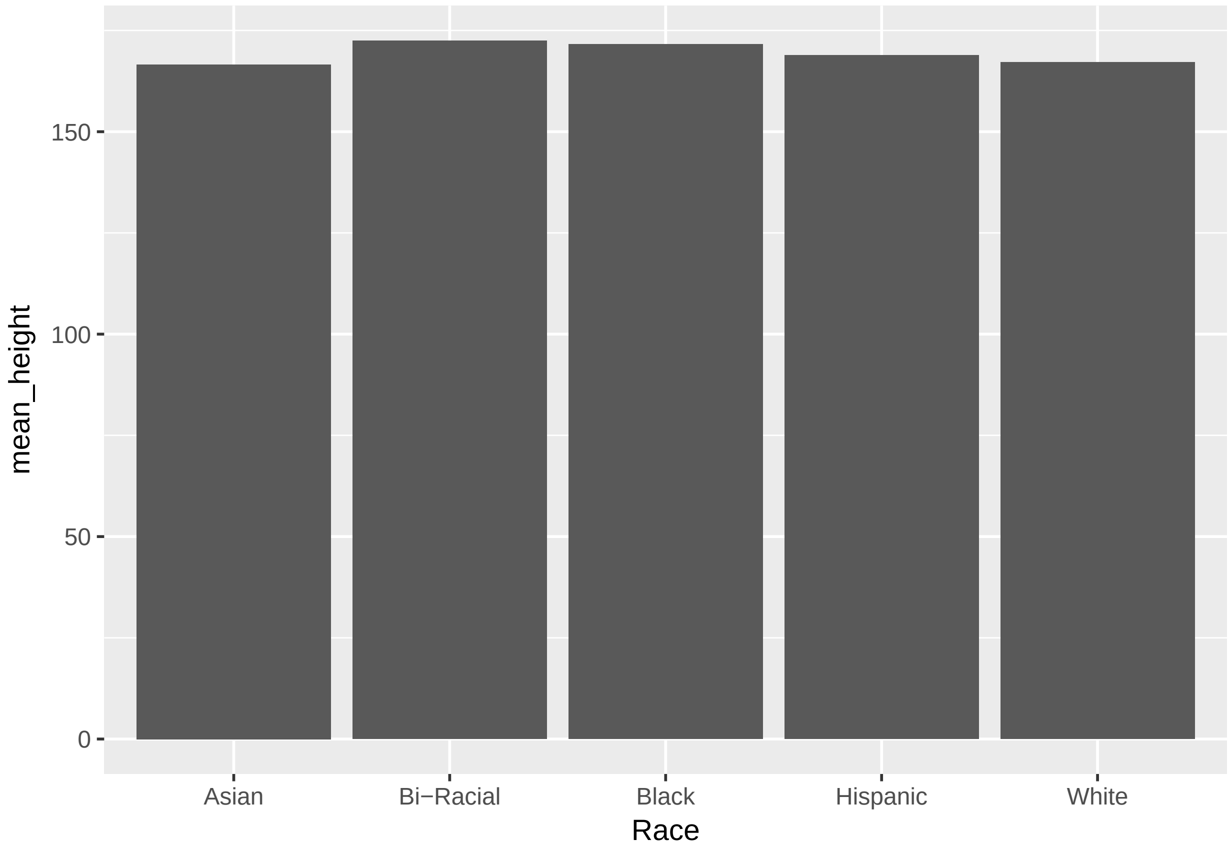
The reason is easier to understand if I show you a barplot that doesn't give an error.

```
ggplot(patients, aes(x=Height)) +
  geom_bar()
```



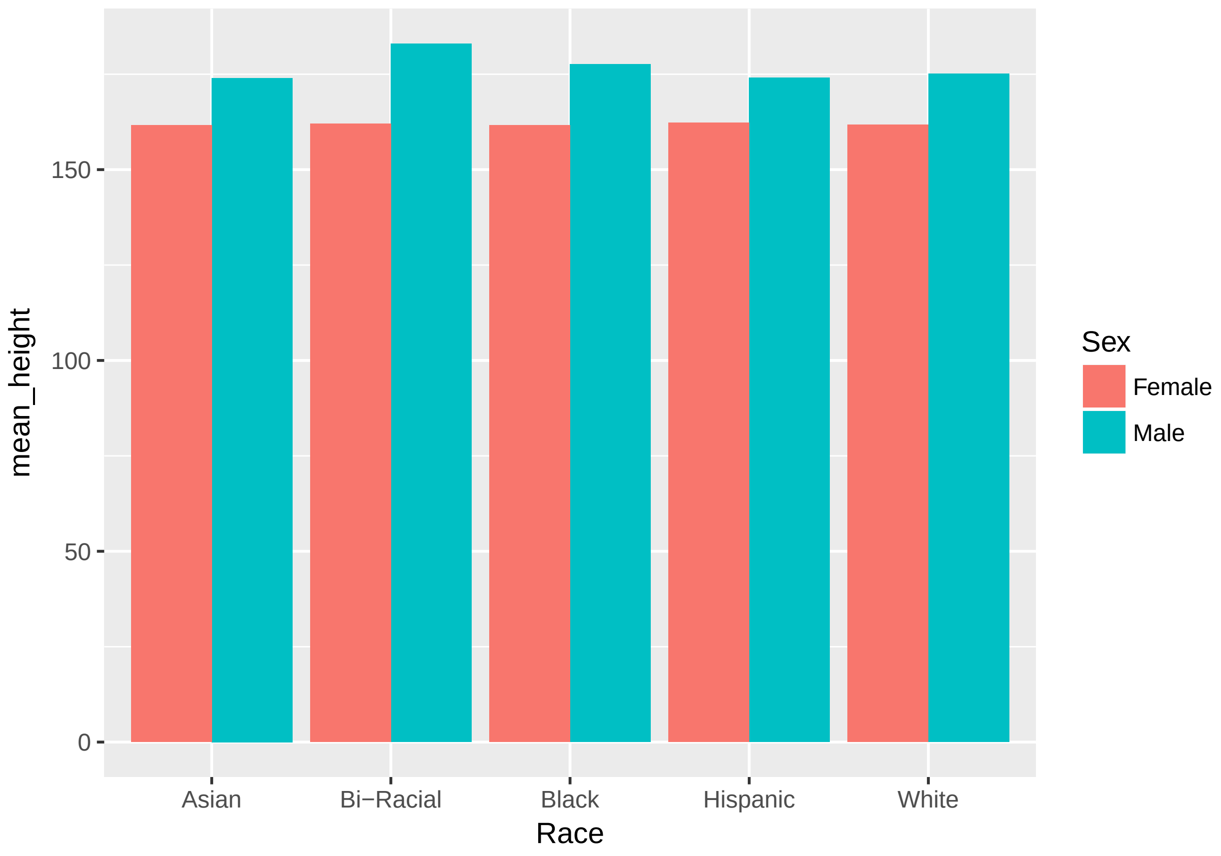
What has it plotted? y = number of patients that had a height of 157, 158..
What this function has done, is automatically counted how many houses/rows have 1 bd, 2 bd...
Notice how we only gave it 1 variable - because it's just counting up how many times there is a 1 bd house and putting the count herre - in the y axis.
Lets look at our previous data again. We are not plotting how many - we want it to plot actually this number here. We can tell it to do so with stat = "identity"

```
patients %>%
  group_by(Race) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race)) +
  geom_bar(stat = 'identity')
```



Challenge ***

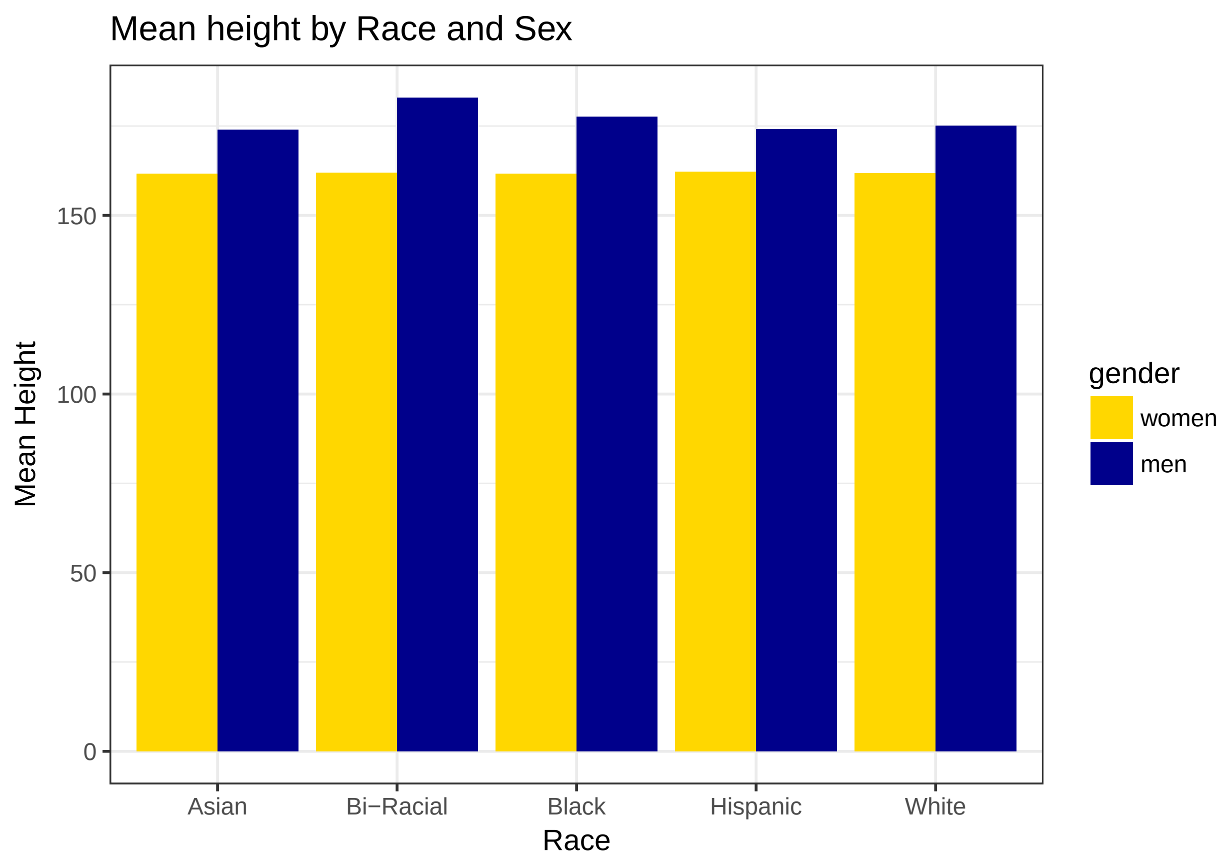
```
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge')
```



Modifying your plot

Adding labels and changing colours

```
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue"), name = 'gender', labels = c('women', 'men')) +
  theme_bw()
```



You can also add THEMES!

Saving plots

There are 2 ways to save plots. Via Export > save as image.

Or using code. Change to .pdf or .png for the type of plot you want.

```
pdf("myPlot.pdf", width = 10, height = 8, units = "px")

#bmp()
#jpeg()
#png()


#make your plot here
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue"))

dev.off
```

This way is much easier if you are making several plots - you don't have to click through and save every plot.

Challenge: Save your plot and then do challenge 8. ***

Scales

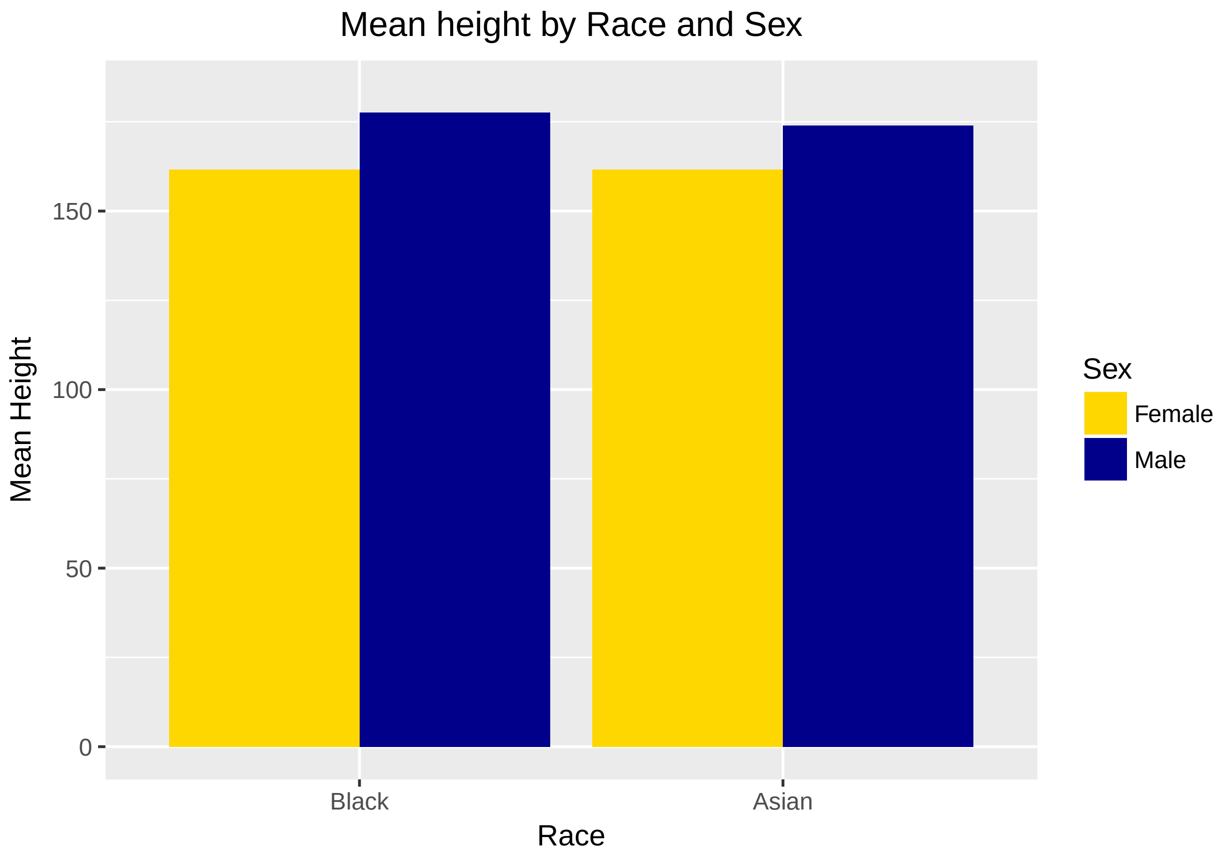
The last thing we will cover is scales - changing the axis.

Discrete axis

So here we have a discrete axis - they are categories.

```
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue")) +
  scale_x_discrete(limit = c("Black", "Asian"))
```

```
## Warning: Removed 6 rows containing missing values (geom_bar).
```

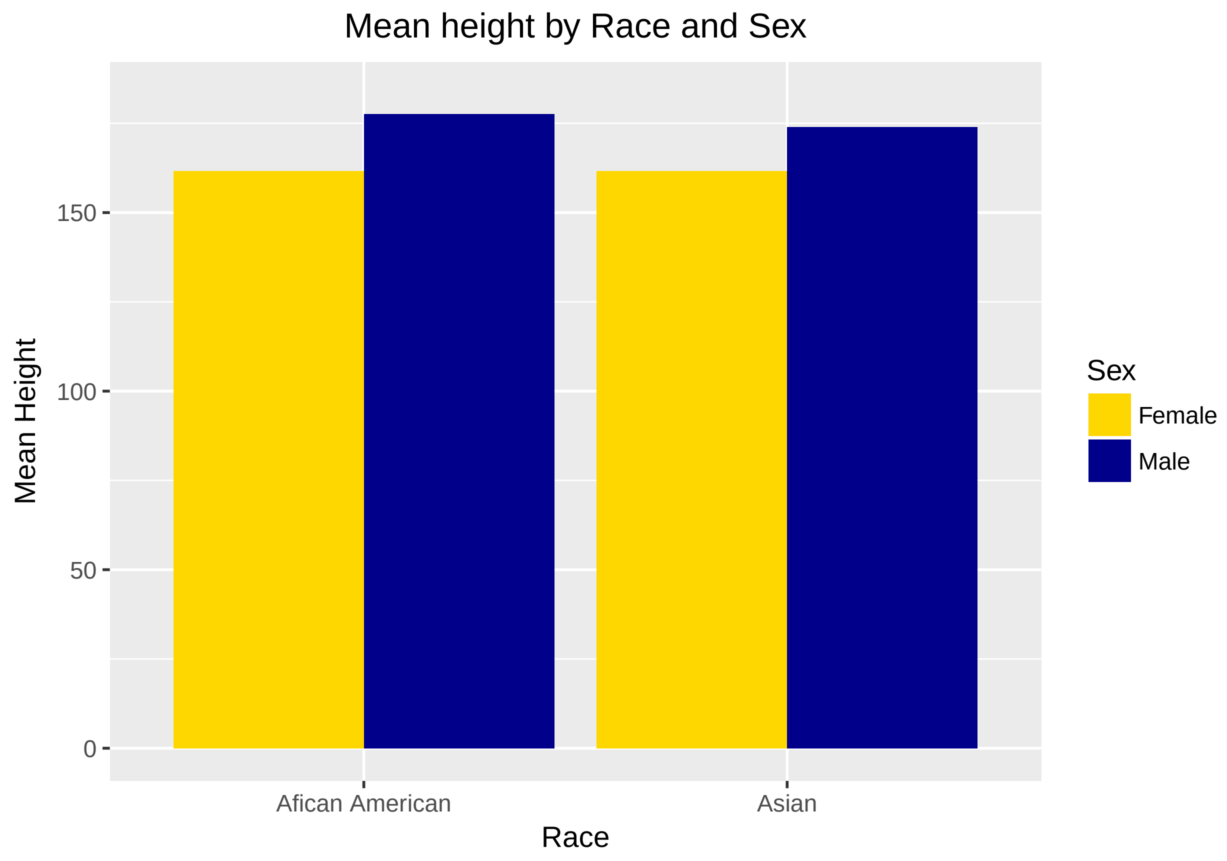


You can also control what order they appear in.

You can also change the names.

```
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue"))+
  scale_x_discrete(limit = c("Black", "Asian"), labels = c("Afican American", 'Asian'))
```

```
## Warning: Removed 6 rows containing missing values (geom_bar).
```



Continuous axis

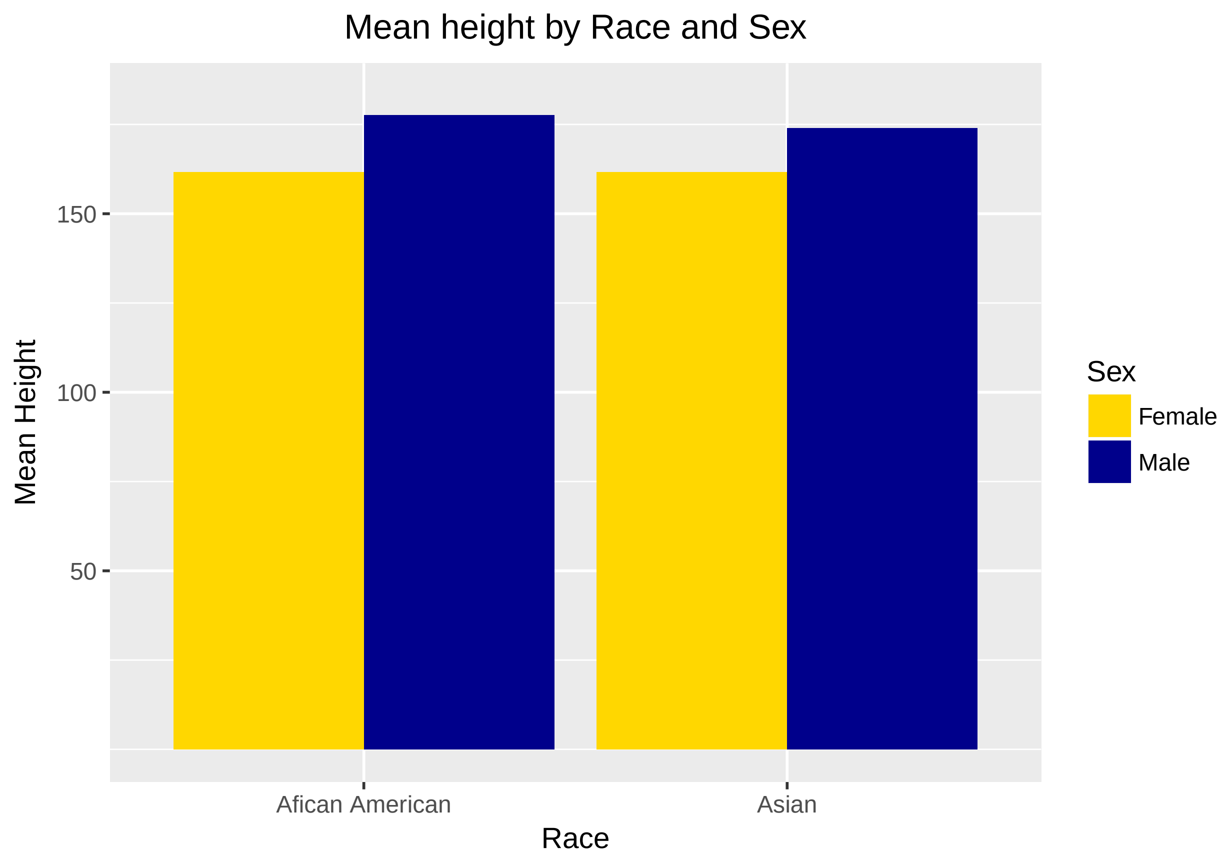
The y axis here is continuous.

You can change the ‘breaks’.

```
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue")) +
```

```r
  scale_x_discrete(limit = c("Black", "Asian"), labels = c("Afican American", 'Asian')) +
  scale_y_continuous(breaks = c(50,100,150))
```

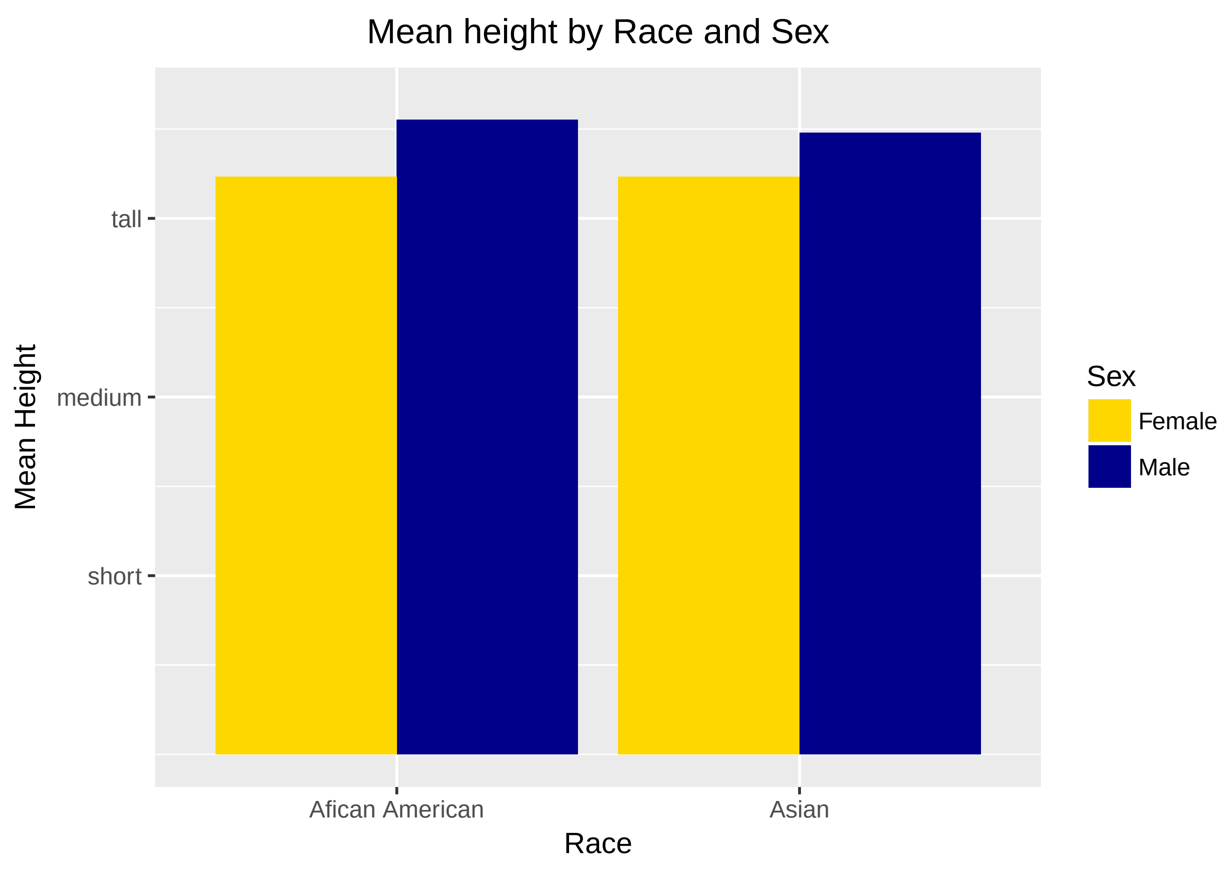
```
## Warning: Removed 6 rows containing missing values (geom_bar).
```



You can also change the labels - it doesn't make sense to do so here but just for demonstration.

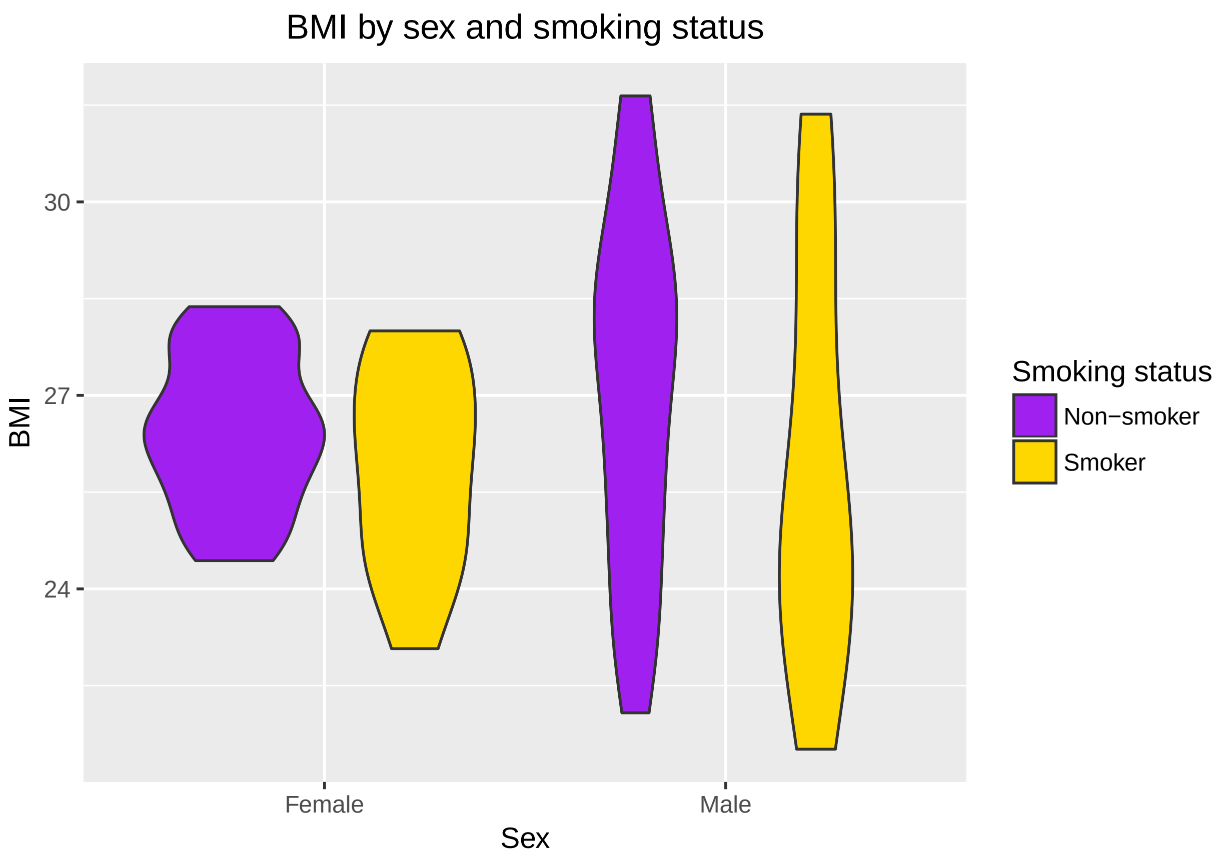
```r
patients %>%
  group_by(Race, Sex) %>%
  summarise(mean_height = mean(Height)) %>%
  ggplot(aes(y=mean_height, x= Race, fill=Sex)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  labs(title = 'Mean height by Race and Sex', y= "Mean Height") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("gold", "darkblue")) +
  scale_x_discrete(limit = c("Black", "Asian"), labels = c("Afican American", 'Asian')) +
  scale_y_continuous(breaks = c(50,100,150), labels = c('short', 'medium', 'tall'))
```

```
## Warning: Removed 6 rows containing missing values (geom_bar).
```



Challenge 9 ***

```
ggplot(patients, aes(y=BMI, x = Sex, fill = Smokes)) +
  geom_violin() +
  labs(title = 'BMI by sex and smoking status') +
  theme(plot.title = element_text(hjust = 0.5)) +
   scale_fill_manual(values = c("purple", "gold"), name = 'Smoking status', label = c("Non-smoker", 'Sm
```


BMI by sex and smoking status
30
27
24
BMI
Female
Male
Sex
Smoking status
Non−smoker
Smoker