100% charts are best used to compare proportions between groups. They are also known as 100% stacked area charts.

The area in 100% charts do not overlap because they are cumulative at each point. The y-axis of a 100% chart is a value percentage, running from 0% to 100%. The x-axis are the categories being measured against, which could represent dates or times e.g. years, quarters, months, etc.

When should you use them?

100% charts are **most** suitable when:

- your values can be measured by the percentage of the whole
- the total amount (100%) is crucial to how the data is broken down
- you are looking to see how multiple values develop over time
- there are considerably large differences in your values
- you have many dates as points of comparison.

When should you use another chart?

100% charts are **least** suitable when:

- you are looking to display one value overtaking another value. While a 100% chart will reflect this change in data, a different type of chart, such as a line chart, will show this change in a clearer fashion.

- you have many values that you are comparing, or the values are too similar. With too many values for comparison, a 100% chart may become difficult to read.

- the changes in your data are not significant when compared to the 100% total.

100% don't always have to be square or rectangle shapes. Explore Google's interactive Music timeline 100% chart. What do you think are the strengths and weaknesses of this graph?

Give it a go!

In this exercise, you're going to create your first graphic plot using **ggplot2**. You'll do this incrementally, where the plot slowly becomes more complex as you get closer to its desired appearance.

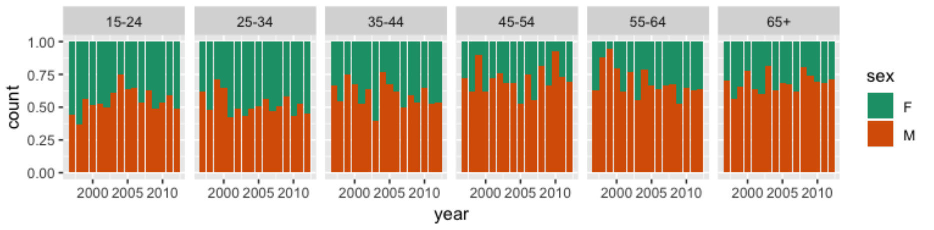
Before you begin, download the data set for tuberculosis (TB) notifications and then place it in a subfolder of your project folder on your computer. If you haven't already, open RStudio on your computer, return to this step and then follow along.

```r
library(tidyverse)
tb <- read_rds("data/tb_tidy.rds")
```

```
tb_au <- filter(tb,
                country == "Australia",
                !is.na(age_group))
tb_au
```

```
## # A tibble: 192 x 6
##    country   iso3   year count sex   age_group
##    <chr>     <chr> <dbl> <dbl> <fct> <fct>
##  1 Australia AUS    1997     8 M     15-24
##  2 Australia AUS    1998    11 M     15-24
##  3 Australia AUS    1999    13 M     15-24
##  4 Australia AUS    2000    16 M     15-24
##  5 Australia AUS    2001    23 M     15-24
##  6 Australia AUS    2002    15 M     15-24
##  7 Australia AUS    2003    14 M     15-24
##  8 Australia AUS    2004    18 M     15-24
##  9 Australia AUS    2005    32 M     15-24
## 10 Australia AUS    2006    33 M     15-24
## # ... with 182 more rows
```

Once you've completed each step in the exercise below, you'll have plotted a 100% chart that compares the proportion of TB cases in each age group by sex. Your finished chart should resemble the following:



Focus of 100% charts

There are three key things learnt from this chart:

- The focus is on **proportion** in each category.
- Across (almost) all ages, and years, the proportion of males having TB is higher than females.
- These proportions tend to be higher in the older age groups, for all years.

Building your own `ggplot`

The first code that's used when making any **`ggplot2`** graphic is the function **`ggplot()`**. Usually, you pass a **data set** and an **aesthetic mapping** as arguments to the function.

An **aesthetic mapping**, which is sometimes called a **variable mapping**, is constructed with the function **`aes()`**. With **`aes()`**, you're telling **`ggplot()`** which variable in your data should correspond to a particular graphical element to be drawn.

Read and review

Consider the following example:

```
p <- ggplot(tb_au, aes(x = year, y = count, fill = sex))
```

The first argument provided is the name of the data, **`tb_au`**. The variable mapping consists of **three** elements, where the variable:

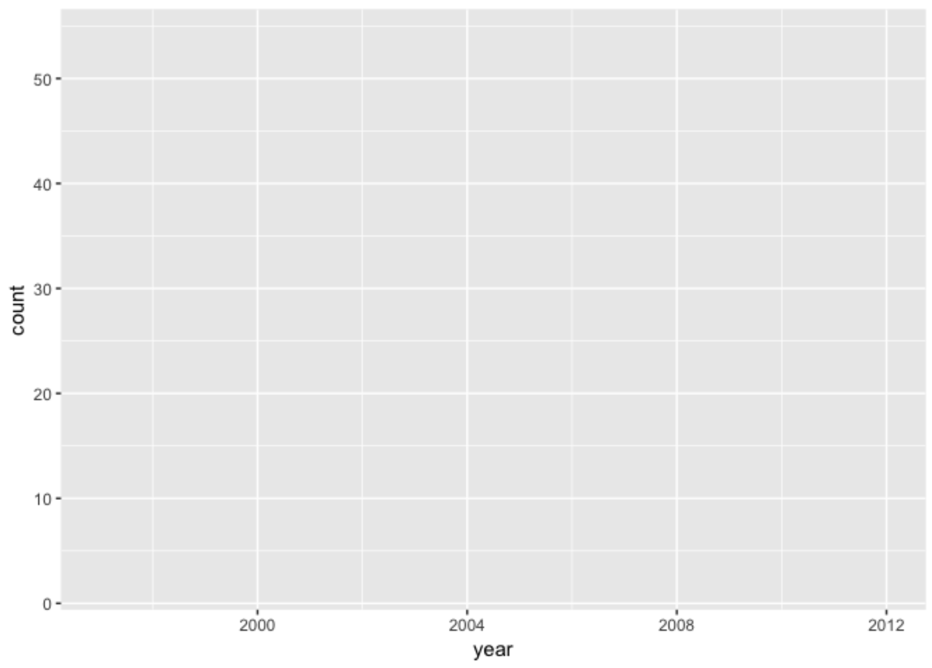
- **year** will be mapped to the x-axis,
- **count** will be mapped to the y-axis

- **sex** will be mapped to the fill element (it will be the colour)

The result is saved to an object called **p**. What does it look like if you print it from RStudio on your computer?

Enter the following in RStudio to find out.

```
p
```



You have a blank plot! The **year** and **count** have been mapped to the axes, although there's no fill.

Draw a geometrical shape

In order to draw something you need a **geom**; this refers to the geometrical shape (think a point or a rectangle) that will be drawn.

You want to draw bars, so you can add **geom_bar()** to your plot object **p**. Copy and run the following code chunk in RStudio on your computer.

```
p <- p + geom_bar(stat = "identity", position = "fill")
```

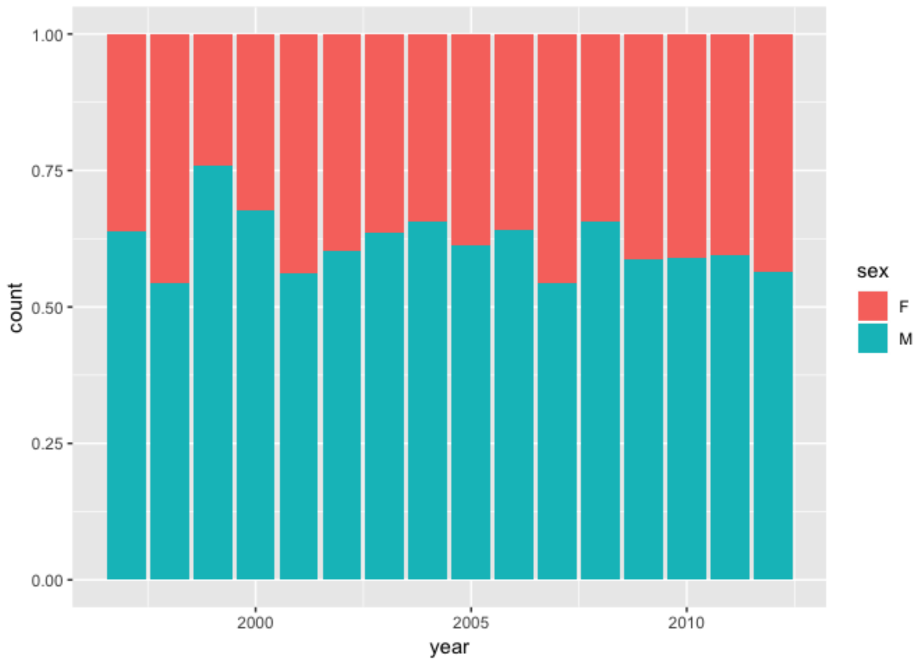
When new grammar elements are added to a plot object using **ggplot2**, they are included using the **+** operator.

Here, you have added **geom_bar()** with two arguments **stat = "identity"** and **position = "fill"**. The former refers to any transformation that will be applied to the **count** column. You have already counted how many TB incidences are in each combination of categories, so **stat = "identity"** says no need to compute the count setting.

The latter argument refers to how the bars are placed. You're mostly interested in proportions between sex, over years, separately by age. The **position = "fill"** option in **geom_bar** sets the heights of the bars to be all at 100%. It ignores counts, and emphasises the proportion of males and females.

Show the result by entering the following in RStudio:

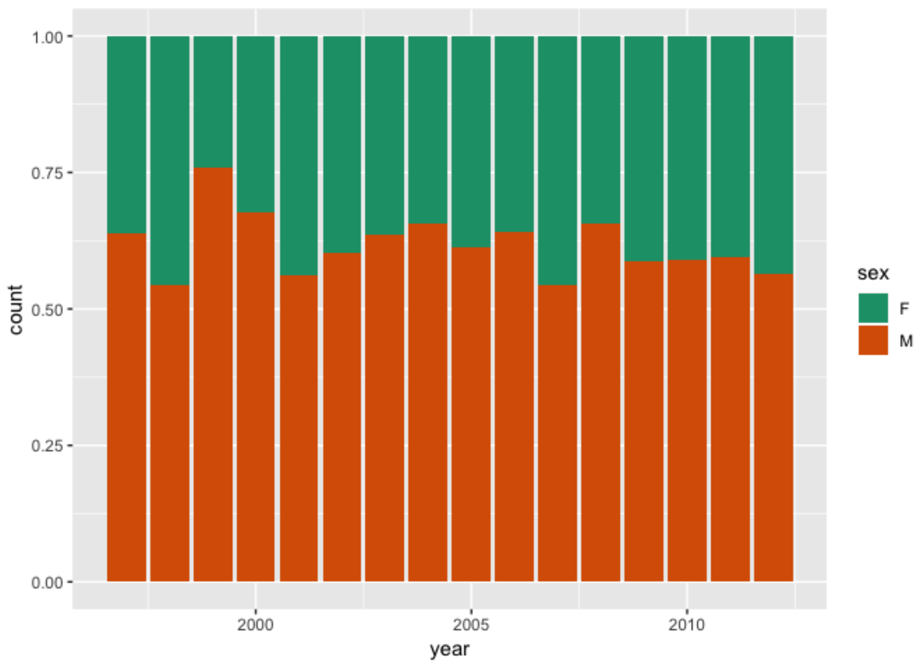
```
p
```



Now you have bars that are filled by the proportion of male and female TB cases for each year. You might have noticed that **`ggplot`** automatically draws a legend.

You may want to consider changing the colour of the bars to accommodate colour blindness. You can do this by adding a **`scale`** element.

```
p <- p + scale_fill_brewer(palette = "Dark2")
p
```

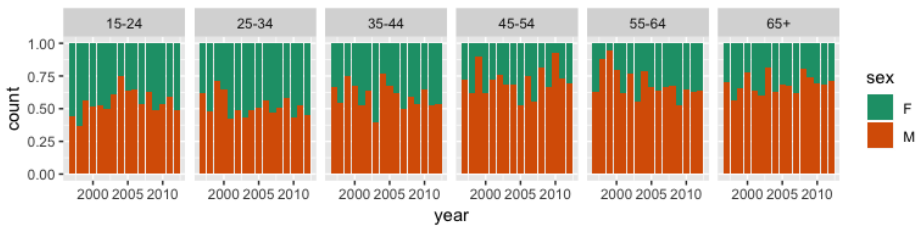


For this exercise, modify the fill colours by using the colour palettes from Cynthia Brewer. The argument **`palette = "Dark2"`** refers to the name of the colour palette. You can find all of the colour brewer palettes on colorbrewer2.org.

Finishing the chart

The final step is to create a bar chart for each age group present in the TB data, which you can do by adding a **`facet`** element. Do this with the following code chunk:

```
p <- p + facet_grid(. ~ age_group)
p
```

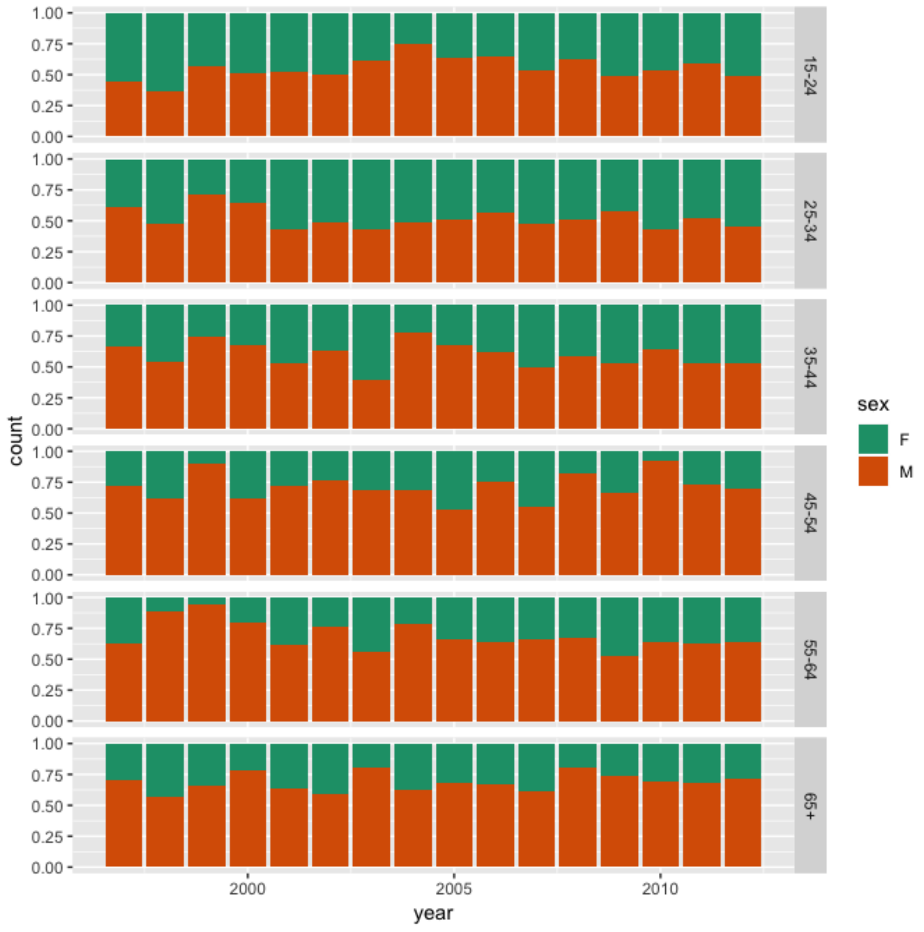


What’s a `facet`?

A **`facet`** creates subplots for each category (or combination of categories) laid out in a grid. The argument that constructs the facets is **`. ~ age_group`**.

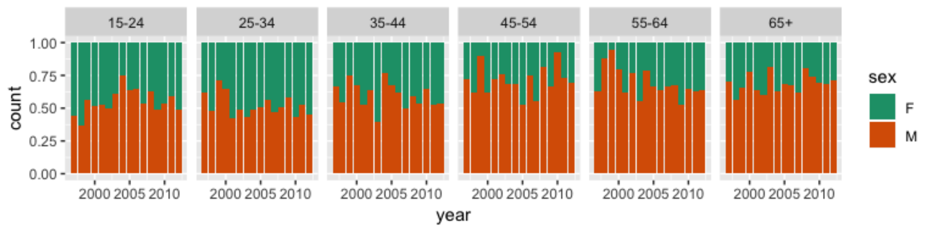
This is called a **formula** and is of the form **left-hand side ~ right-hand side**. The **~** is called a tilde and is usually located on the top left of most keyboards. How you place variables within a formula changes the way subplots are laid out on the screen.

```
p <- p + facet_grid( age_group ~ .)
p
```



To summarise, you have created your first **`ggplot`** one layer at a time. Once you become more confident you will be able to start building everything up at once. Your completed chart should resemble the following:

```
p <- ggplot(tb_au, aes(x = year, y = count, fill = sex)) +
  geom_bar(stat = "identity", position = "fill") +
  facet_grid(~ age_group) +
  scale_fill_brewer(palette="Dark2")
p
```



Tell us how you went

Within the **Comments**, share with other learners your experience of creating your first graphic plot using **`ggplot2`**.

In the next step, you will learn how changing parts of the code can modify the appearance of the plot, and you will learn how to go from 100% chart to a regular bar chart.