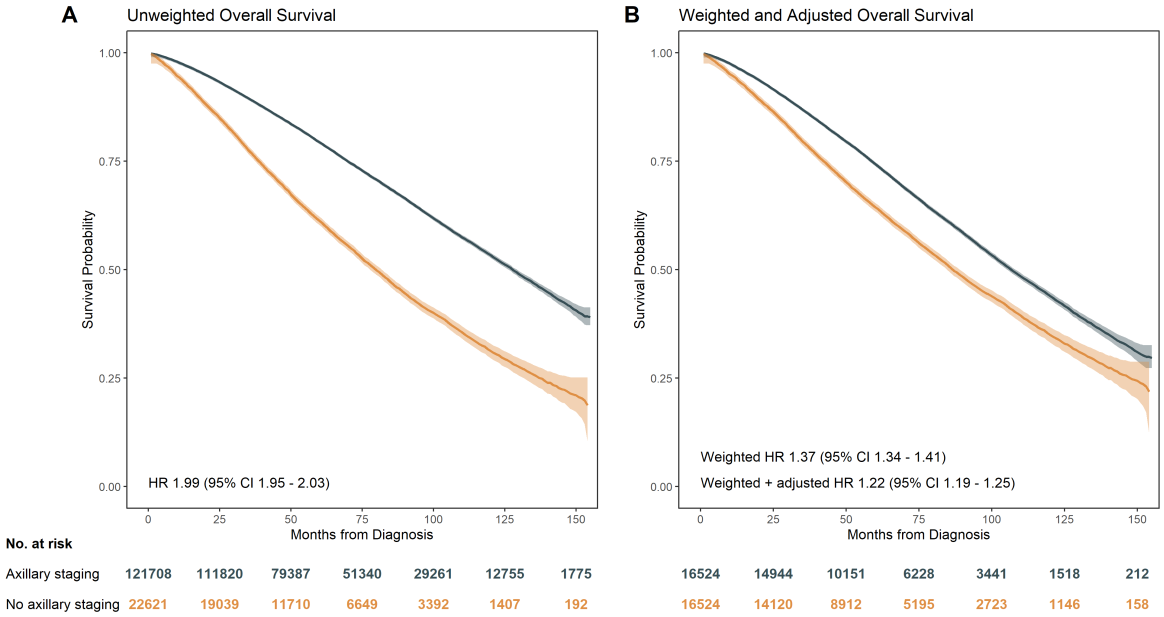
Introduction to ggplot2: Creating simple publication-quality figures

Matthew Castelo

June, 2021

This tutorial will introduce ggplot2 and allow you to quickly generate publication-quality figures from clean datasets. The ggplot2 package is extremely flexible and is suitable for a variety of users. From researchers who wish to create clean, high-resolution figures that complement analyses performed in other software, to those wanting to create complex, annotated figures that convey a greater amount of information.

Some examples of ggplot2 figures are presented below:



No. at risk	0	25	50	75	100	125	150	0	25	50	75	100	125	150
Axillary staging	121708	111820	79387	51340	29261	12755	1775	16524	14944	10151	6228	3441	1518	212
No axillary staging	22621	19039	11710	6649	3392	1407	192	16524	14120	8912	5195	2723	1146	158

Figure 1: Panelled Kaplan-Meier curves with at-risk tables shown below and annotations.

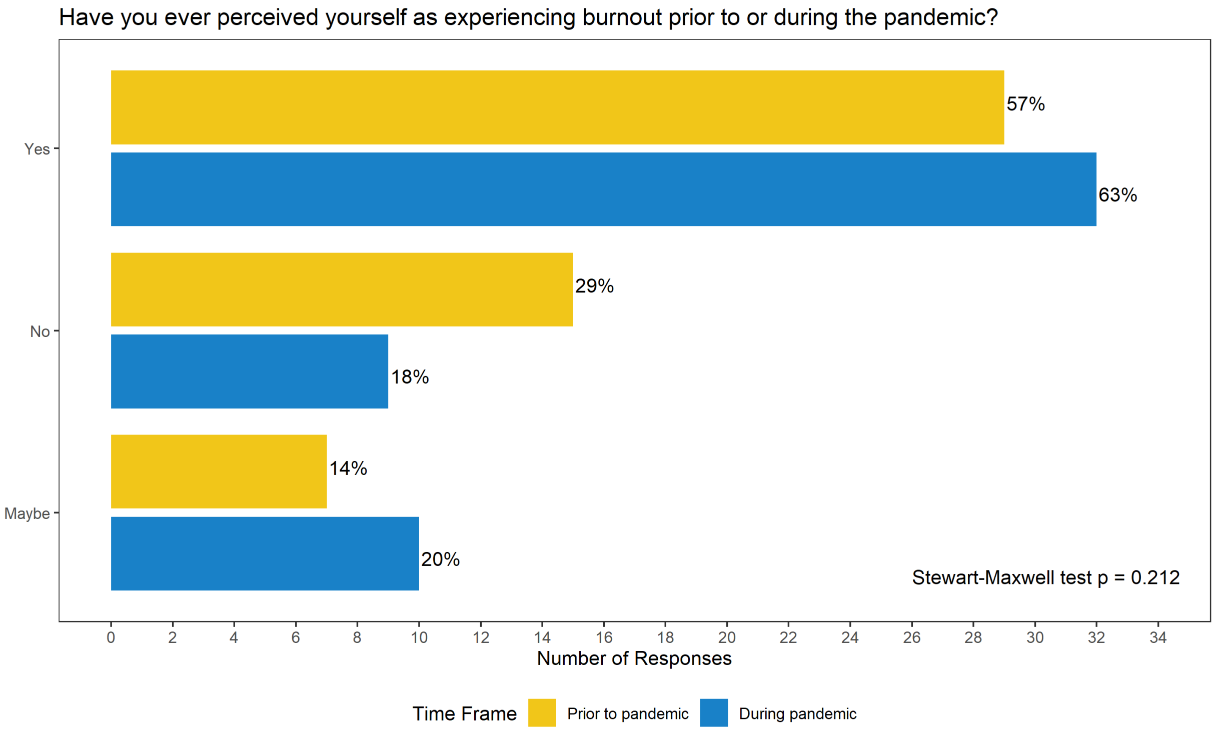
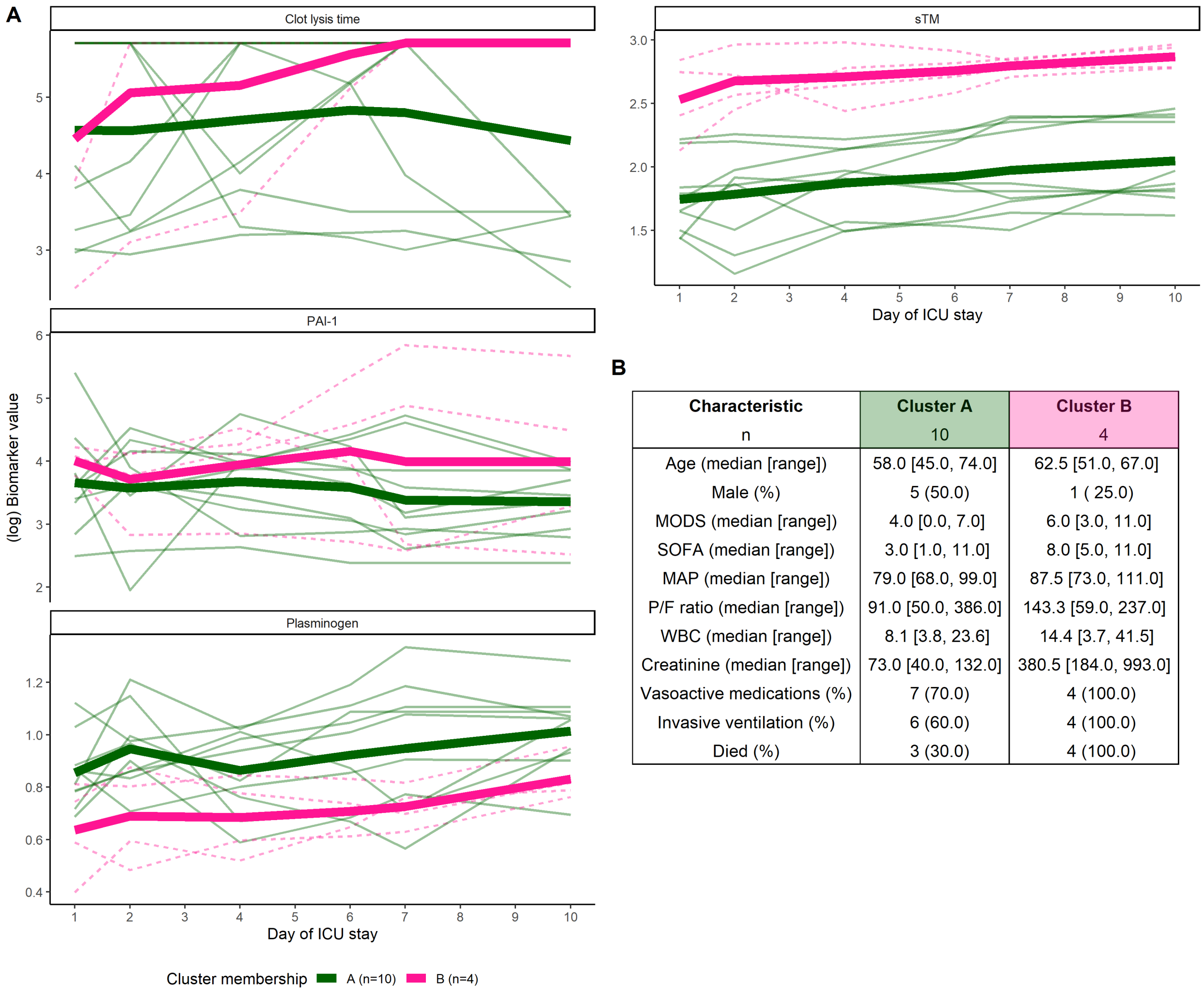


Figure 2: Bar chart showing before/after survey data with annotations.



Characteristic	Cluster A	Cluster B
n	10	4
Age (median [range])	58.0 [45.0, 74.0]	62.5 [51.0, 67.0]
Male (%)	5 (50.0)	1 (25.0)
MODS (median [range])	4.0 [0.0, 7.0]	6.0 [3.0, 11.0]
SOFA (median [range])	3.0 [1.0, 11.0]	8.0 [5.0, 11.0]
MAP (median [range])	79.0 [68.0, 99.0]	87.5 [73.0, 111.0]
P/F ratio (median [range])	91.0 [50.0, 386.0]	143.3 [59.0, 237.0]
WBC (median [range])	8.1 [3.8, 23.6]	14.4 [3.7, 41.5]
Creatinine (median [range])	73.0 [40.0, 132.0]	380.5 [184.0, 993.0]
Vasoactive medications (%)	7 (70.0)	4 (100.0)
Invasive ventilation (%)	6 (60.0)	4 (100.0)
Died (%)	3 (30.0)	4 (100.0)

Figure 3: Complex figure with plots and an inset table.

First, you should have R and RStudio installed. Next, install the **tidyverse** package by running the following command:

```
install.packages("tidyverse")
```

The **tidyverse** package will install a number of smaller packages, including **ggplot2** for plotting, and **dplyr** for data manipulation. Once installed, packages need to be loaded every time RStudio is launched. It is typical to load your required packages and set your working directory in the first code chunk.

```
library(tidyverse)

#The working directory saves the following path when loading files, saving plots, etc
setwd("C:/Users/matth/Documents/R/R code and education/Tutorials/")
```

We will be working with a simulated dataset representing 80 generic cancer patients. The dataset can be loaded directly from GitHub so all the subsequent code will run on your computer.

```
urlRemote  <- "https://raw.githubusercontent.com/"
pathGithub <- "mcas-surg/Tutorials/main/Datasets/"
fileName   <- "generic_cancer.csv"

cancer <- read_csv(paste0(urlRemote, pathGithub, fileName))
```

We now have our dataset saved to the *cancer* object. The dataset can be inspected with some useful functions. `head()` will show by default just the first 5 rows.

```
head(cancer)
```

```
## # A tibble: 6 x 13
##   male   pt_id   age high_grade adverse_tumour_marker tumour_size diabetes
##   <chr>  <dbl> <dbl> <chr>      <chr>                       <dbl> <chr>
## 1 Male       1    62 No         No                           0.41 No
## 2 Male       2    57 No         No                           2.84 No
## 3 Female     3    66 No         No                           3.35 No
## 4 Male       4    54 No         No                           4.63 No
## 5 Male       5    49 No         No                           1.29 No
## 6 Male       6    63 No         Yes                          1.99 No
## # ... with 6 more variables: heart_disease <chr>, hospital <chr>,
## #   extended_resection <chr>, postop_complication <chr>, vital_status <chr>,
## #   follow_up <dbl>
```

The `glimpse()` function will show the dimensions of the dataset, each column, and the variable type (numeric, character, etc.).

```
glimpse(cancer)
```

```
## Rows: 80
## Columns: 13
## $ male                  <chr> "Male", "Male", "Female", "Male", "Male", "Male"~
## $ pt_id                 <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1~
## $ age                   <dbl> 62, 57, 66, 54, 49, 63, 54, 72, 57, 65, 63, 54, ~
## $ high_grade             <chr> "No", "No", "No", "No", "No", "No", "No", "No", ~
## $ adverse_tumour_marker <chr> "No", "No", "No", "No", "No", "Yes", "No", "No",~
## $ tumour_size           <dbl> 0.41, 2.84, 3.35, 4.63, 1.29, 1.99, 1.86, 1.01, ~
## $ diabetes              <chr> "No", "No", "No", "No", "No", "No", "No", "No", ~
## $ heart_disease         <chr> "No", "No", "No", "Yes", "No", "No", "No", "No",~
## $ hospital              <chr> "Community hospital", "Community hospital", "Com~
## $ extended_resection    <chr> "Yes", "No", "No", "No", "No", "Yes", "No", "No"~
## $ postop_complication   <chr> "Yes", "No", "Yes", "Yes", "No", "Yes", "Yes", "~
## $ vital_status          <chr> "Alive", "Died", "Died", "Alive", "Alive", "Aliv~
## $ follow_up             <dbl> 58, 52, 33, 58, 56, 41, 64, 32, 30, 40, 29, 29, ~
```

It is sometimes helpful to open the dataset in a “spreadsheet” format. This can be accomplished using `View()`.

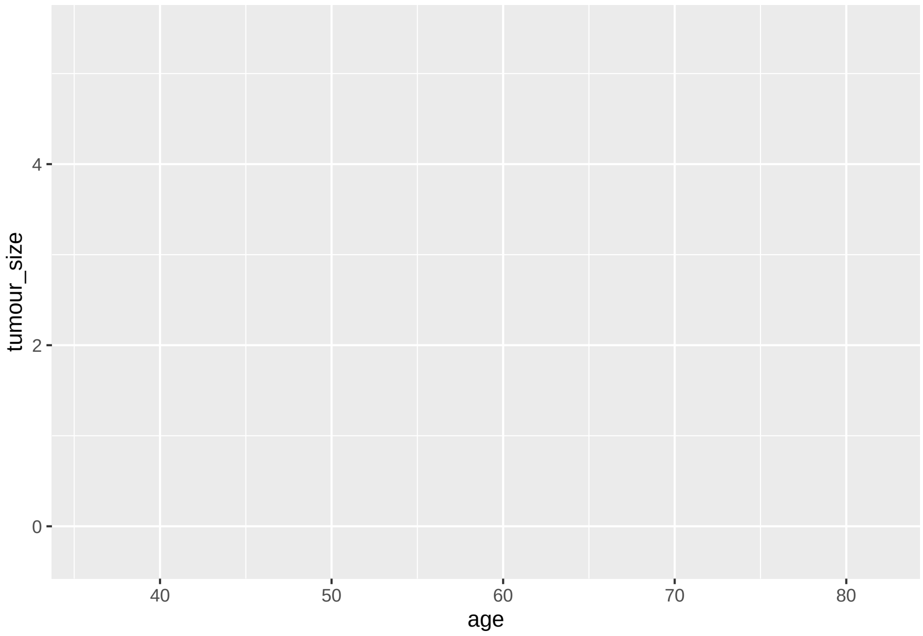
We note our dataset has 80 observations and 13 variables.

	level	Overall
n		80
male (%)	Female	36 (45.0)
	Male	44 (55.0)
age (mean (SD))		60.54 (7.51)
high_grade (%)	No	60 (75.0)
	Yes	20 (25.0)
adverse_tumour_marker (%)	No	59 (73.8)
	Yes	21 (26.2)
tumour_size (mean (SD))		2.82 (1.25)
diabetes (%)	No	70 (87.5)
	Yes	10 (12.5)
heart_disease (%)	No	51 (63.7)
	Yes	29 (36.2)
hospital (%)	Academic hospital	38 (47.5)
	Community hospital	42 (52.5)
extended_resection (%)	No	44 (55.0)
	Yes	36 (45.0)
postop_complication (%)	No	50 (62.5)
	Yes	30 (37.5)
vital_status (%)	Alive	30 (37.5)
	Died	50 (62.5)
follow_up (mean (SD))		40.38 (12.18)

Let's start with a simple scatter plot of age versus tumour size using **ggplot2**. The dataset is passed to ggplot using a pipe (%>%). This symbol links commands and functions to each other, allowing more efficient coding.

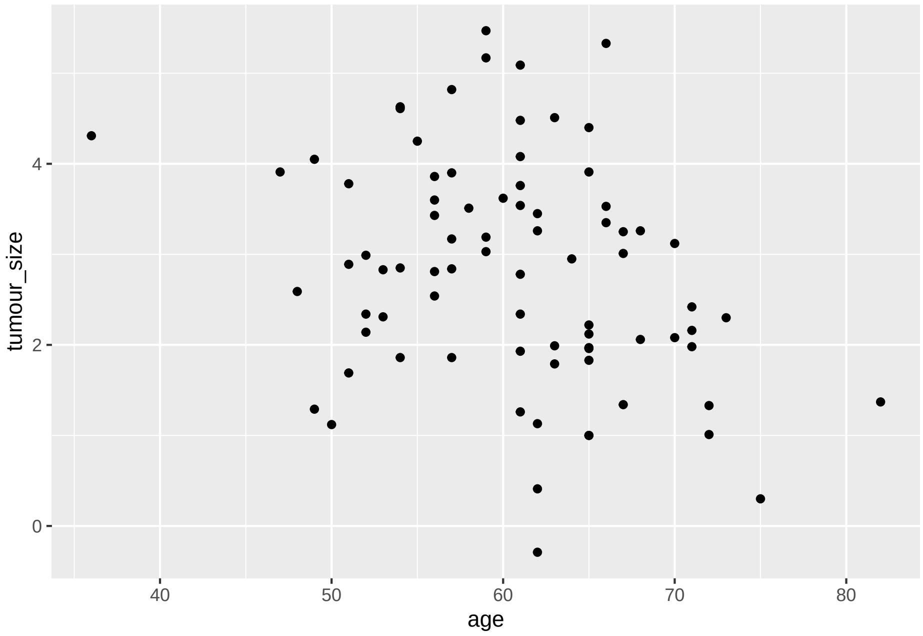
A ggplot is build in layers. First we pass the dataset to `ggplot()`, and define our aesthetics. Aesthetics (`aes()`) link variables in our dataset to visual elements on the plot. For example, color, position, size, transparency etc.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size))
```



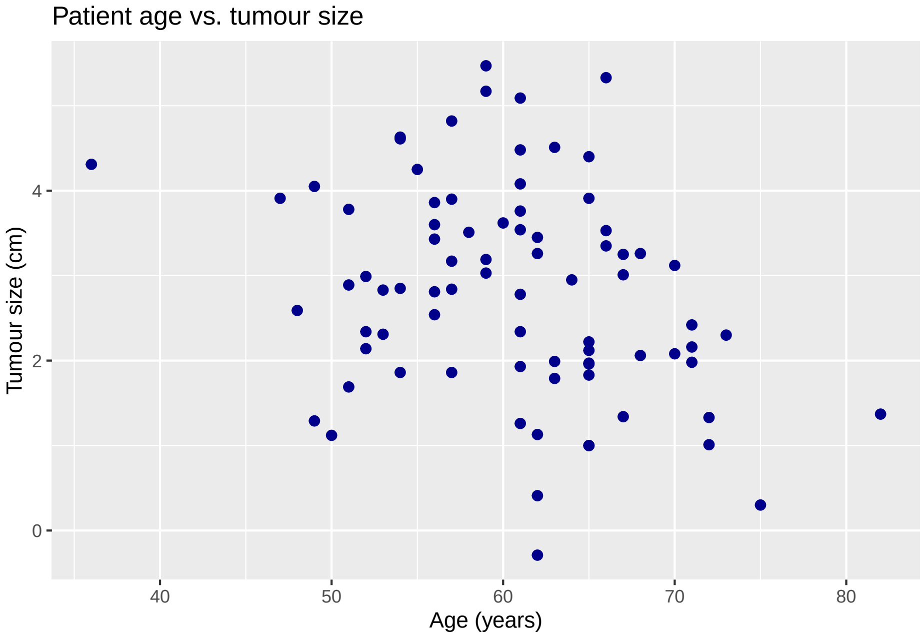
This doesn't do much on its own. Now we can add layers. These will automatically inherit the relationships we defined as aesthetics. The layer `geom_point()` will make our scatter plot.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size))+
  geom_point()
```



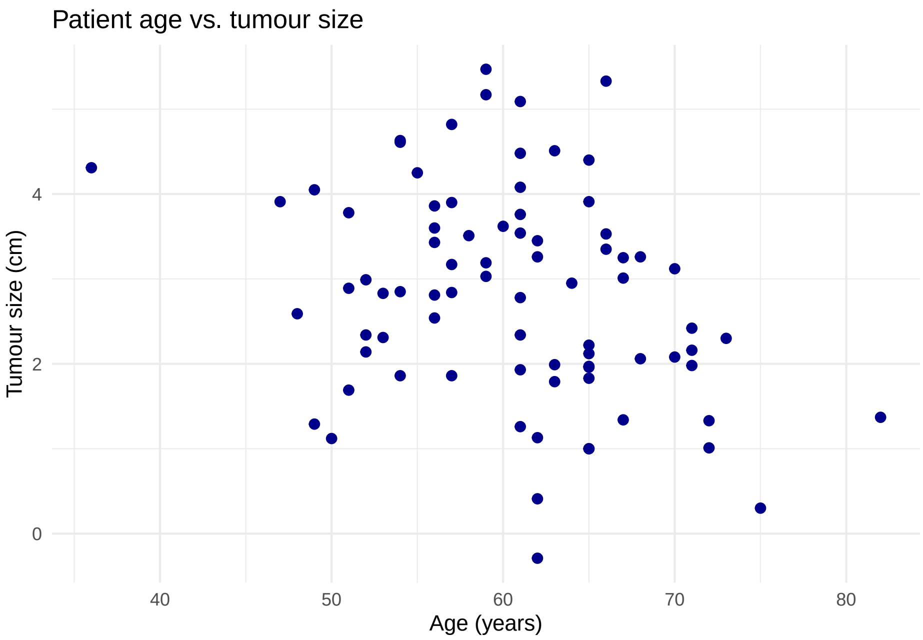
We can manually increase the size of the points and add colour. Let's also change the name of the axes and add a title using `labs()`.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size))+
  geom_point(size = 2, colour = "darkblue")+
  labs(title = "Patient age vs. tumour size",
       x = "Age (years)",
       y = "Tumour size (cm)")
```



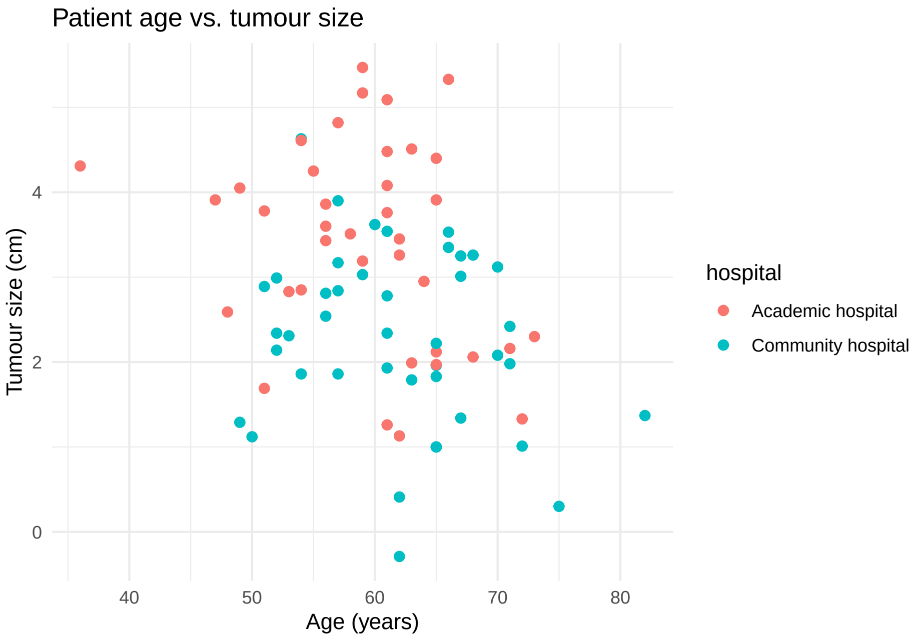
There are many built-in themes for **ggplot2**. The black and white theme `+theme_bw()` is popular, as is the minimal theme `+theme_minimal()`. Note when adding layers to a ggplot, we use a `+` rather than a pipe. Also note the code format. Place each layer on a new line to improve readability.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size))+
  geom_point(size = 2, colour = "darkblue")+
  labs(title = "Patient age vs. tumour size",
       x = "Age (years)",
       y = "Tumour size (cm)")+
  theme_minimal()
```



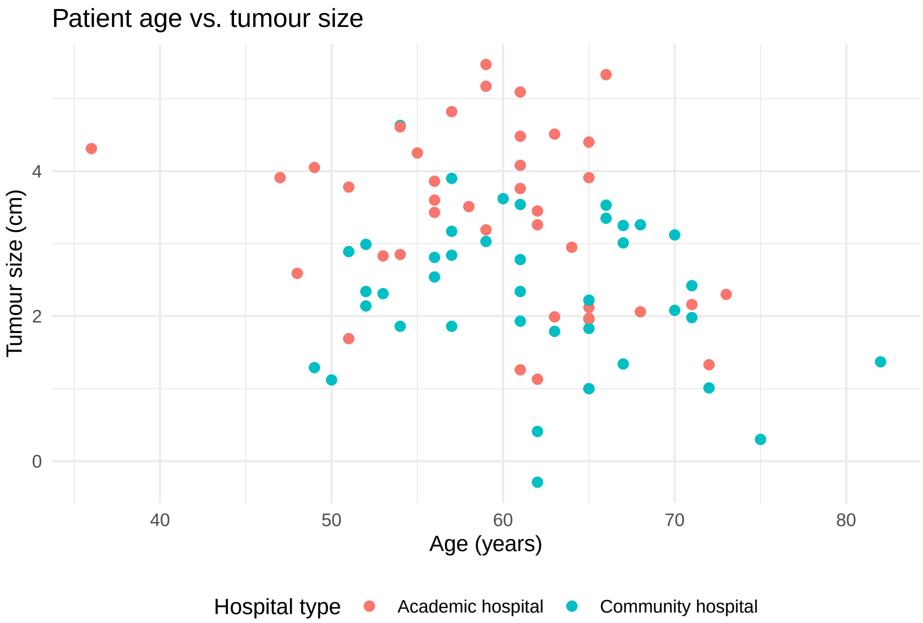
There appears to be a weak relationship between smaller tumours and older age. We can add additional information by colouring points by hospital type. This is done in the aesthetics.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size, colour = hospital))+
  geom_point(size = 2)+
  labs(title = "Patient age vs. tumour size",
       x = "Age (years)",
       y = "Tumour size (cm)")+
  theme_minimal()
```



A legend is automatically generated, and the default **ggplot2** colours are used. We can change the legend name in `labs()` and the position by manually changing the theme. Let's do that now.

```
cancer %>%
  ggplot(aes(x = age, y = tumour_size, colour = hospital))+
  geom_point(size = 2)+
  labs(title = "Patient age vs. tumour size",
       x = "Age (years)",
       y = "Tumour size (cm)",
       colour = "Hospital type")+
  theme_minimal()+
  theme(legend.position = "bottom")
```

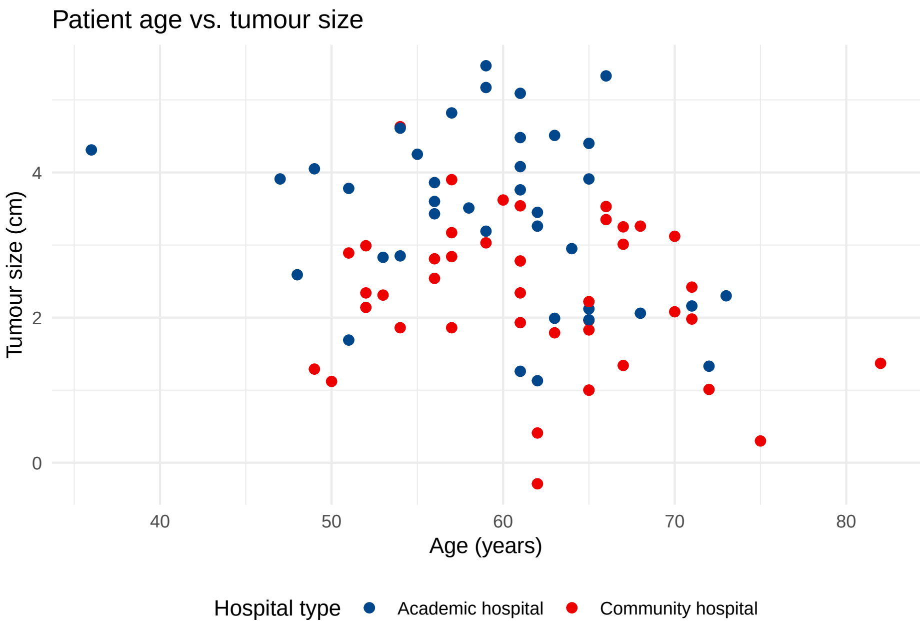


tidyverse contains a useful package called **ggsci**. It contains the colour palettes for several major academic journals (and Star Trek!). The full list of palettes are available here: https://cran.r-project.org/web/packages/ggsci/vignettes/ggsci.html

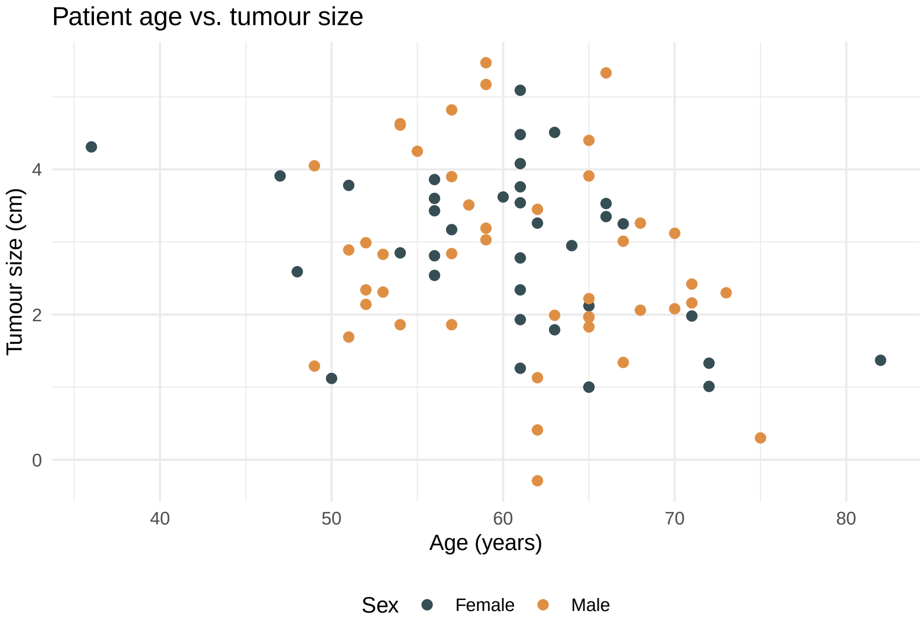
You should never submit figures with the default **ggplot2** colours as they are highly recognizable. Let's plan to submit to the Lancet.

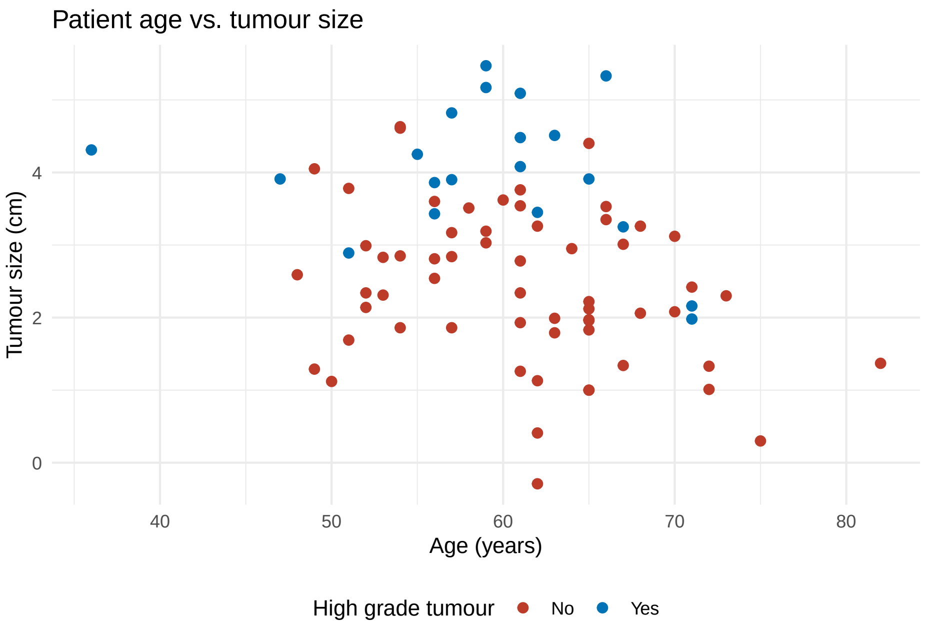
```
library(ggsci)

cancer %>%
  ggplot(aes(x = age, y = tumour_size, colour = hospital))+
  geom_point(size = 2)+
  labs(title = "Patient age vs. tumour size",
       x = "Age (years)",
       y = "Tumour size (cm)",
       colour = "Hospital type")+
  theme_minimal()+
  theme(legend.position = "bottom")+
  scale_colour_lancet()
```



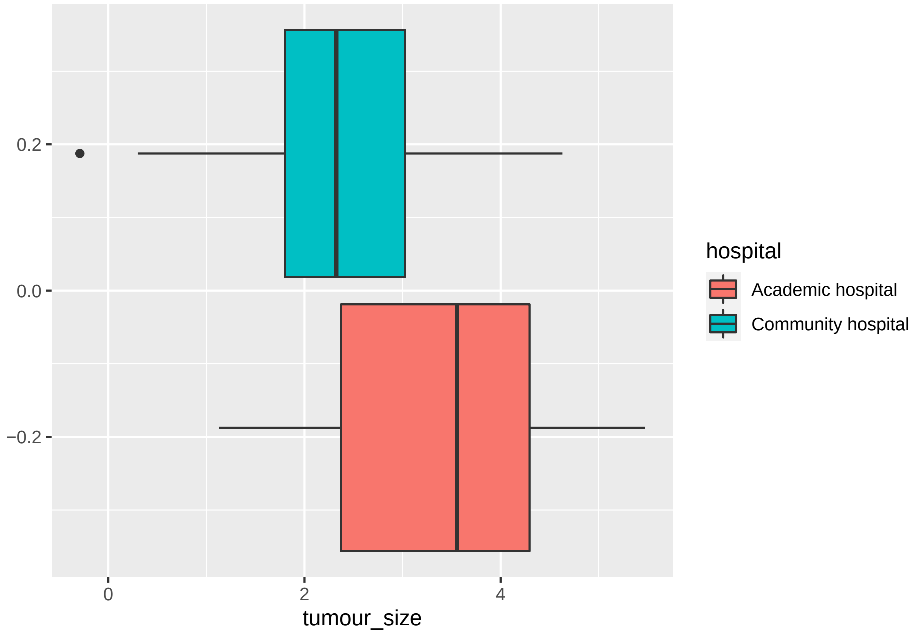
We can also see it appears the academic hospital treats patients with larger tumours. Let's explore two other categorical variables - sex and tumour grade. By simply switching `colour = hospital` to `colour = male` for example. Remember to change the legend title.





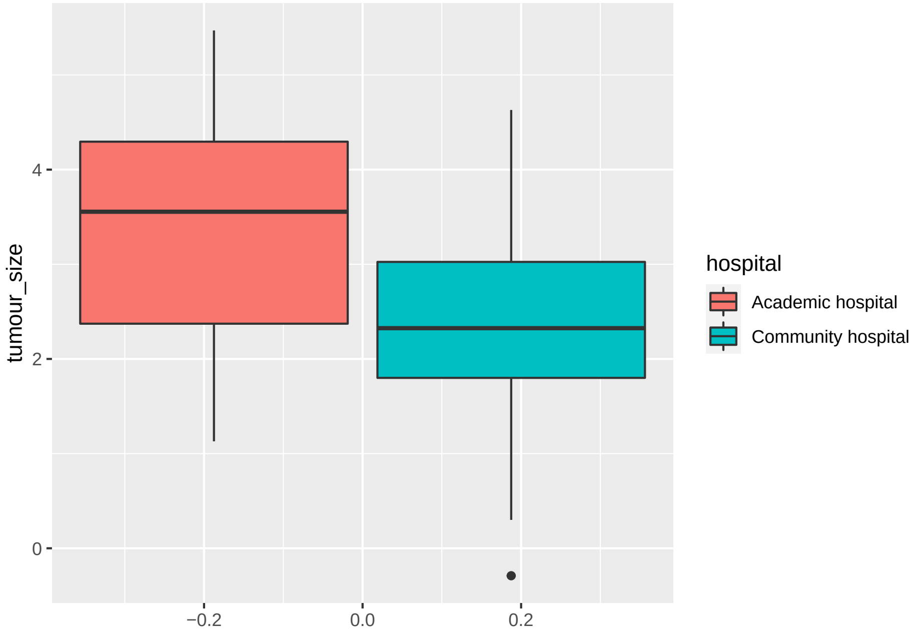
These relationships might be better expressed as a boxplot.

```
cancer %>%
  ggplot(aes(x = tumour_size, fill = hospital))+
  geom_boxplot()
```



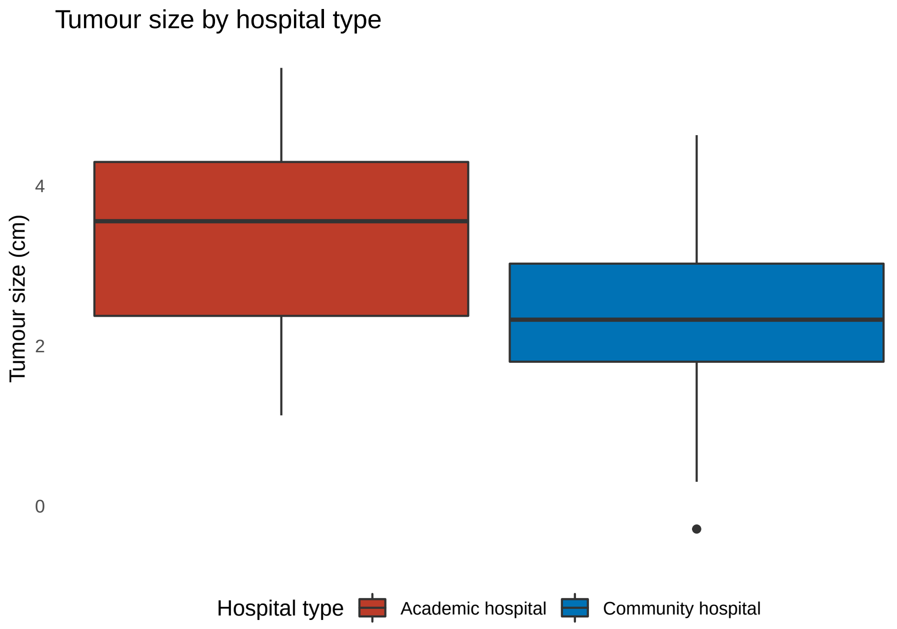
Note we use `fill =` rather than `colour =` as this will only change the line colour rather than fill in the boxplot. By default boxplots are horizontal. We can change this with `+coord_flip()`.

```
cancer %>%
  ggplot(aes(x = tumour_size, fill = hospital))+
  geom_boxplot()+
  coord_flip()
```



Let’s make our usual visual changes. We will also remove the gridlines in the background using `theme()`.

```
cancer %>%
  ggplot(aes(x = tumour_size, fill = hospital))+
  geom_boxplot()+
  coord_flip()+
  theme_minimal()+
  labs(title = "Tumour size by hospital type",
       x = "Tumour size (cm)",
       fill = "Hospital type")+
  theme_minimal()+
  theme(legend.position = "bottom",
        panel.grid = element_blank(),
        axis.text.x = element_blank())+
  scale_fill_nejm()
```



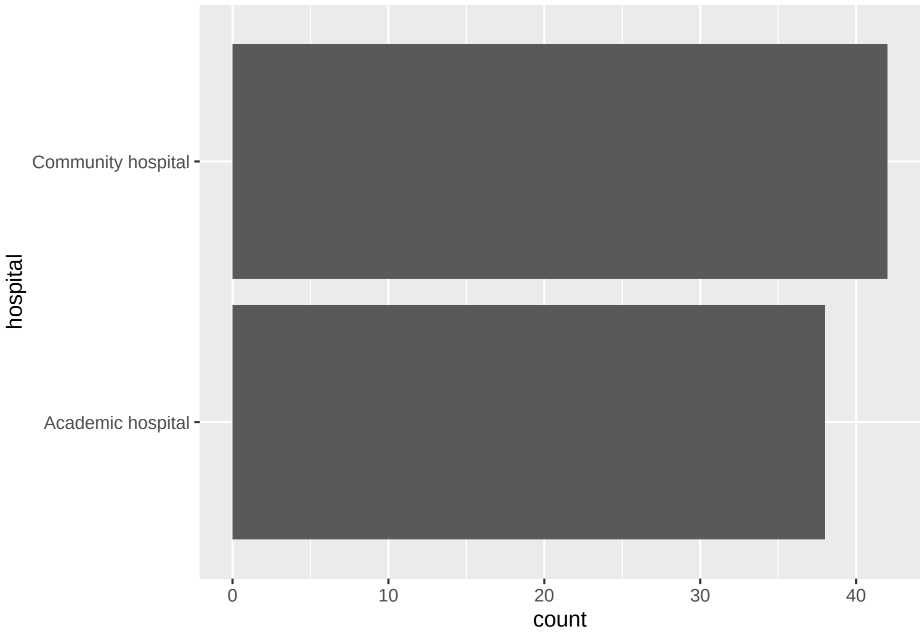
There are a few complexities that are important to point out. Because we used the fill aesthetic, we had to change colour to fill in `labs()` and in our custom colour palette. This changes from `scale_colour_nejm()` to `scale_fill_nejm()`. Setting elements of the plot to `= element_blank()` in `theme()` will remove them. Finally, when we flip boxplots, the axis name remains the same (i.e. tumour size is still the x-axis).

Assuming we are happy with the plot, we can save it as a high-quality png file for publication. The `ggsave()` function allows us to manually set the plot size, file type (png, TIFF, etc), and quality (300 dpi is typical for publication).

```
#If you have not set the working directory, include the full file path
ggsave("my_plot.png",
       dpi = 300, dev = "png",
       height = 15, width = 15, units = "cm")
```

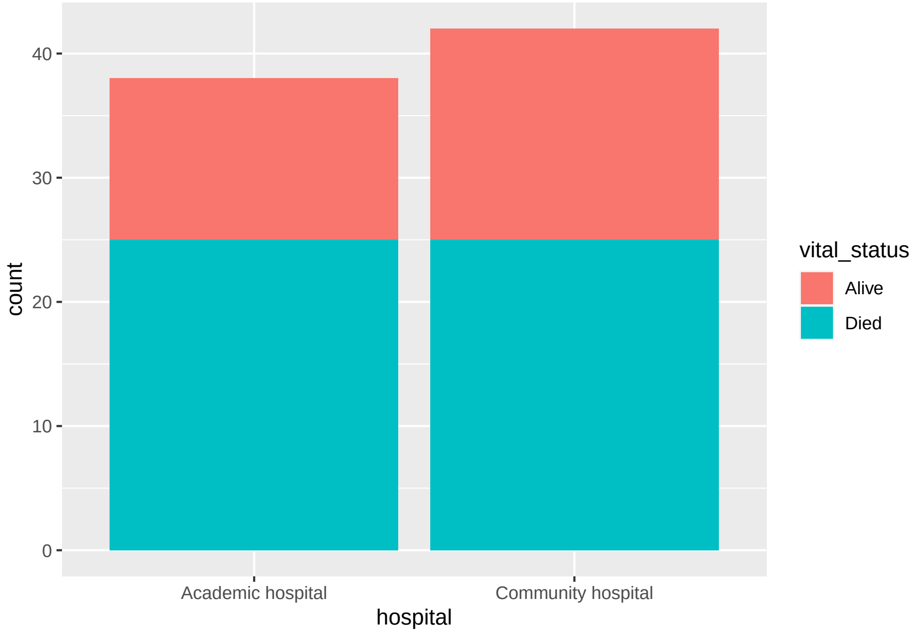
Let's explore bar charts. First, showing the number of patients in each hospital type with the new layer `+geom_bar`.

```
cancer %>%
  ggplot(aes(y = hospital))+
  geom_bar()
```



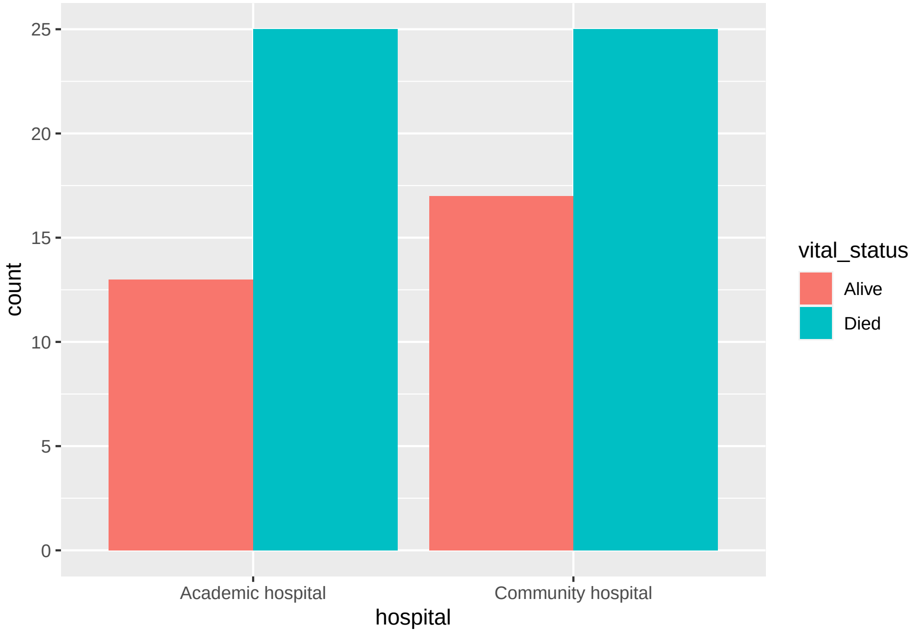
Adding a fill aesthetic showing the number of patients who died vs. survived, and flipping the bars vertical.

```
cancer %>%
  ggplot(aes(y = hospital, fill = vital_status))+
  geom_bar()+
  coord_flip()
```



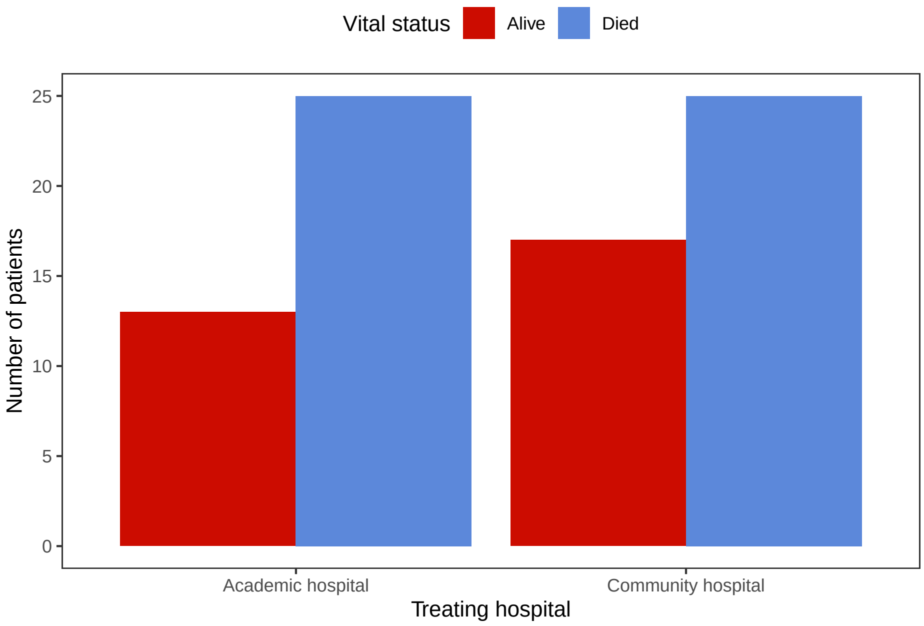
The position of the categories can be changed from stacked (default), to dodged.

```
cancer %>%
  ggplot(aes(y = hospital, fill = vital_status))+
  geom_bar(position = "dodge")+
  coord_flip()
```



Let’s improve the overall visuals, as we’ve seen before. This time we can use the Star Trek Palette, and the black and white theme. We will also put the legend on top.

```
cancer %>%
  ggplot(aes(y = hospital, fill = vital_status))+
  geom_bar(position = "dodge")+
  coord_flip()+
  theme_bw()+
  labs(y = "Treating hospital",
       x = "Number of patients",
       fill = "Vital status")+
  scale_fill_startrek()+
  theme(legend.position = "top",
        panel.grid = element_blank())
```

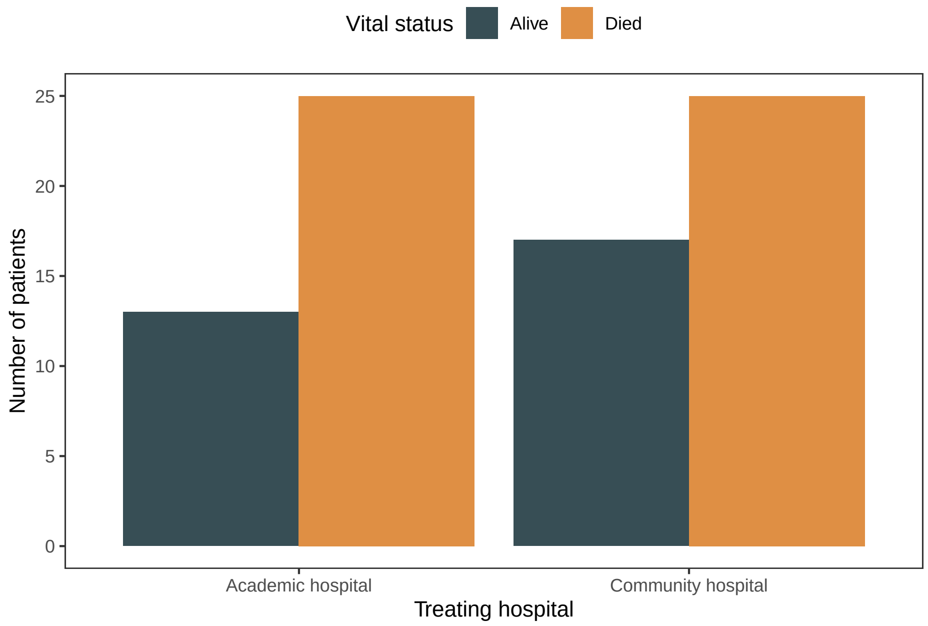


You can assign ggplot code to an object, similar to a dataset. This plot can be used in other tasks, or you can add additional layers to it. Sometimes this can make your code more efficient. For example, the previous plot code can be assigned to the object *plot_1*.

```
plot_1 <- cancer %>%
  ggplot(aes(y = hospital, fill = vital_status))+
  geom_bar(position = "dodge")+
  coord_flip()+
  theme_bw()+
  labs(y = "Treating hospital",
       x = "Number of patients",
       fill = "Vital status")+
  scale_fill_startrek()+
  theme(legend.position = "top",
        panel.grid = element_blank())
```

If we wish to change the color theme to JAMA, we only have to add that single line of code to *plot_1*.

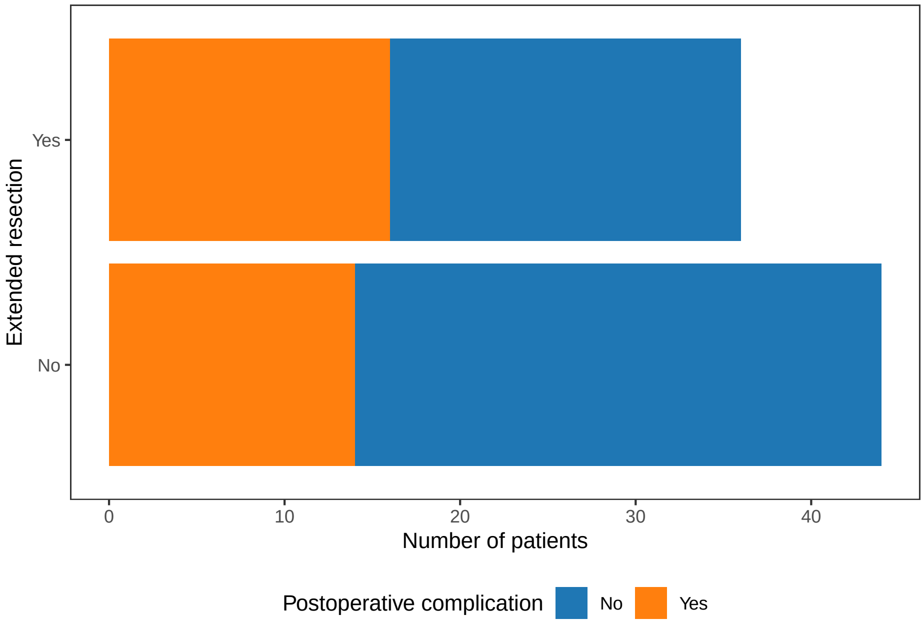
```
plot_1 +
  scale_fill_jama()
```



Continuous axes can be modified with two main arguments: `limits` which controls the upper and lower limit, and `breaks` which controls the size of breaks. Consider the following bar chart of extended resection for cancer versus postoperative complications.

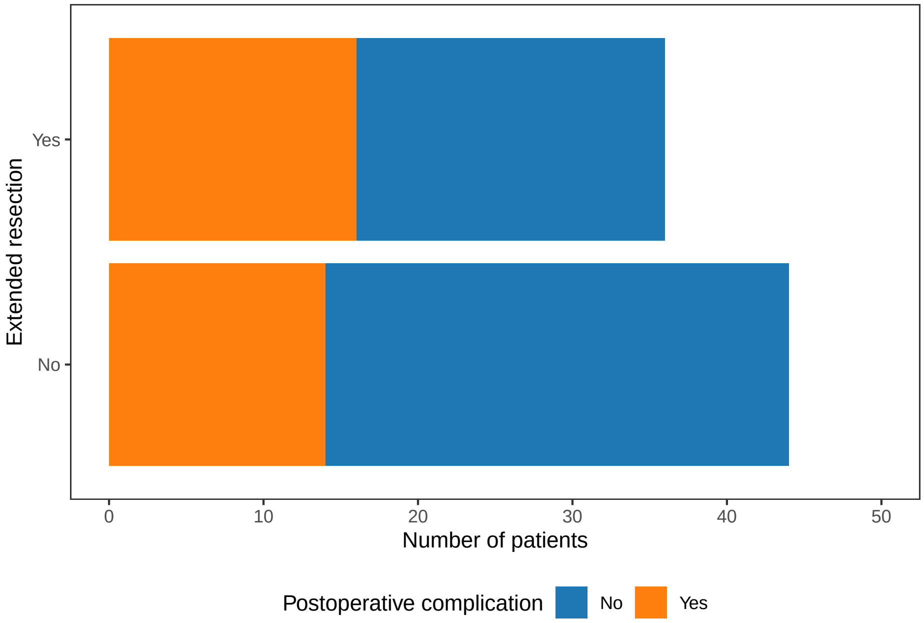
```
plot_2 <- cancer %>%
  ggplot(aes(y = extended_resection, fill = postop_complication))+
  geom_bar()+
  theme_bw()+
  labs(y = "Extended resection",
       x = "Number of patients",
       fill = "Postoperative complication")+
  scale_fill_d3()+
  theme(legend.position = "bottom",
        panel.grid = element_blank())

plot_2
```



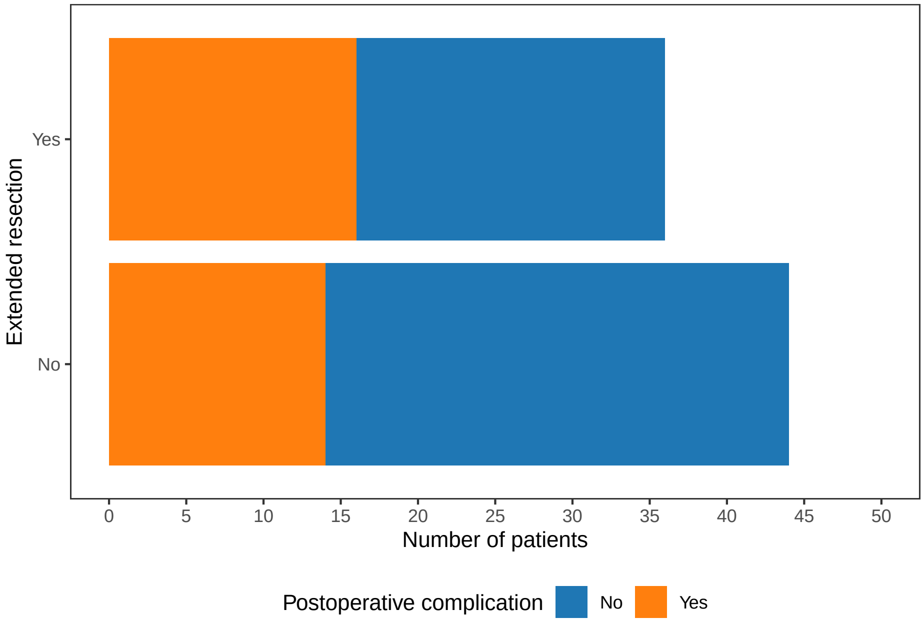
If we wanted to modify the x-axis so the upper limit was 50, we would add the following.

```
plot_2+
  scale_x_continuous(limits = c(0, 50))
```



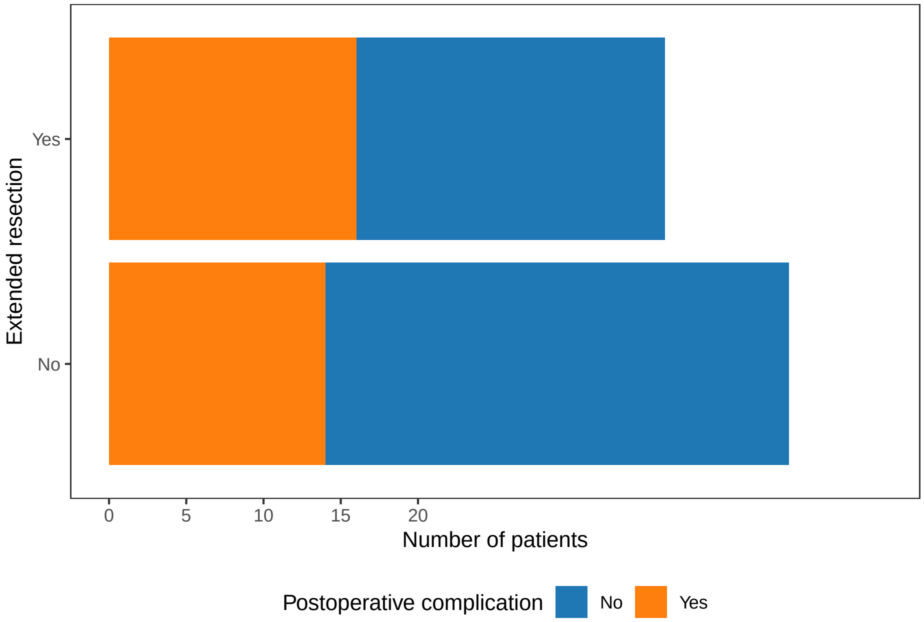
If we would prefer breaks of 5, we can manually set the breaks. It is important to note, the values of `breaks` can be greater than the range set by `limits` but it cannot be smaller without suppressing some values on the axis.

```
plot_2+
  scale_x_continuous(limits = c(0, 50),
                     breaks = seq(0, 50, 5))
```



For example, if we inadvertently set the range of `breaks` smaller than `limits`.

```
plot_2+
  scale_x_continuous(limits = c(0, 50),
                     breaks = seq(0, 20, 5))
```



The piece of code `seq(0, 50, 5)` simply means create a sequence of values starting from 0 going to 50, increasing by 5. Observe the following examples of sequences.

```
seq(1, 10, 1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(100, 200, 50)
```

```
## [1] 100 150 200
```

This was an introduction to creating simple high-quality figures that may complement analyses performed with other statistical software. In later tutorials we will explore other common layers, such as `geom_line` and `geom_ribbon`.