Designing graphs

There are 2 important considerations when designing a graph:

1. It should have something to say.
    - Decide what information you want your graph to display.
2. It should be easy to interpret.
    - Simplicity is key!
    - Sufficiently large fonts, all axes labelled, clear legends, etc.

Data visualisation in **R**

The `ggplot2` package

- **R** has several systems for making graphs, but `ggplot2` is one of the most elegant and versatile.

Install and load the `ggplot2` package

```r
install.packages("ggplot2")
library(ggplot2)
```

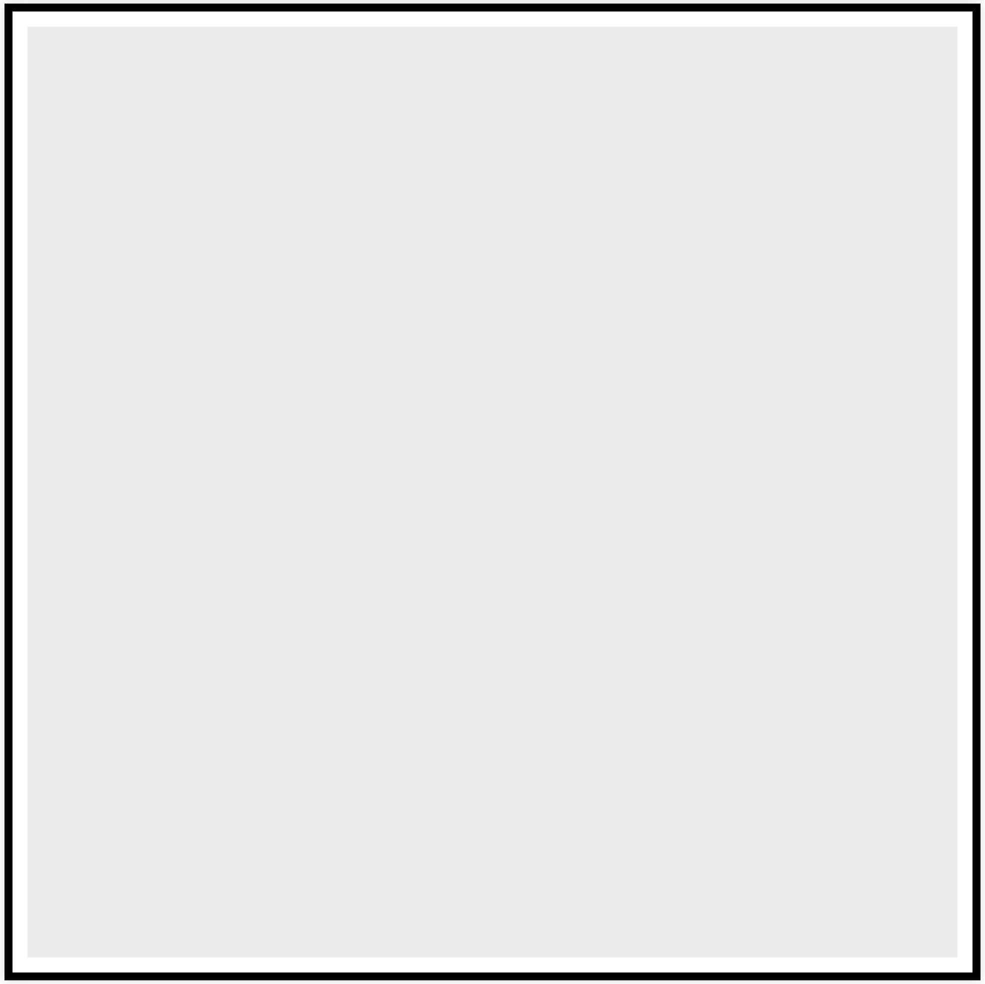
Creating a new plot

We can initialize a new plot with the `ggplot()` function:

```r
ggplot()
```

Creating a new plot

```r
ggplot()
```



Creating a new plot

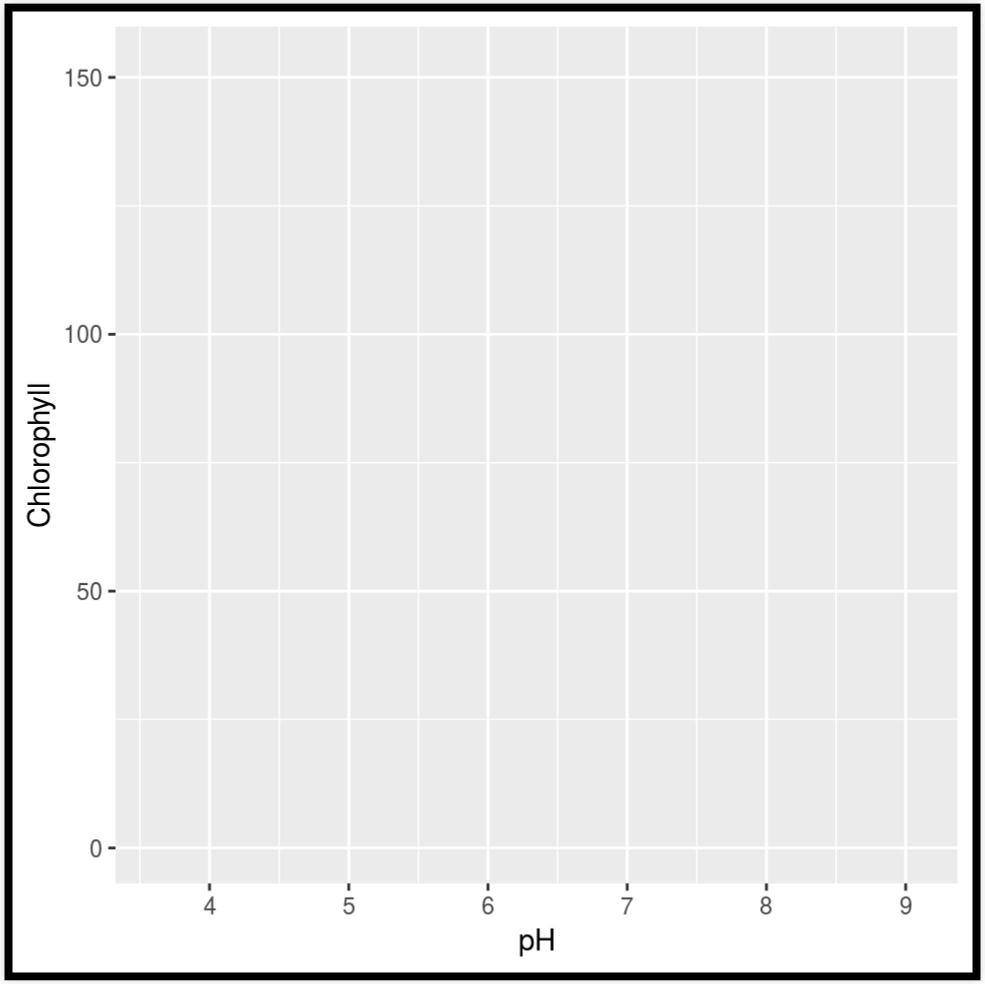
To define the coordinate system, we need to provide 3 pieces of information:

1. Data set (`lake.df`)
2. Which variable to plot on the x axis (`pH`)
3. Which variable to plot on the y axis (`Chlorophyll`)

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll))
```

Creating a new plot

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll))
```



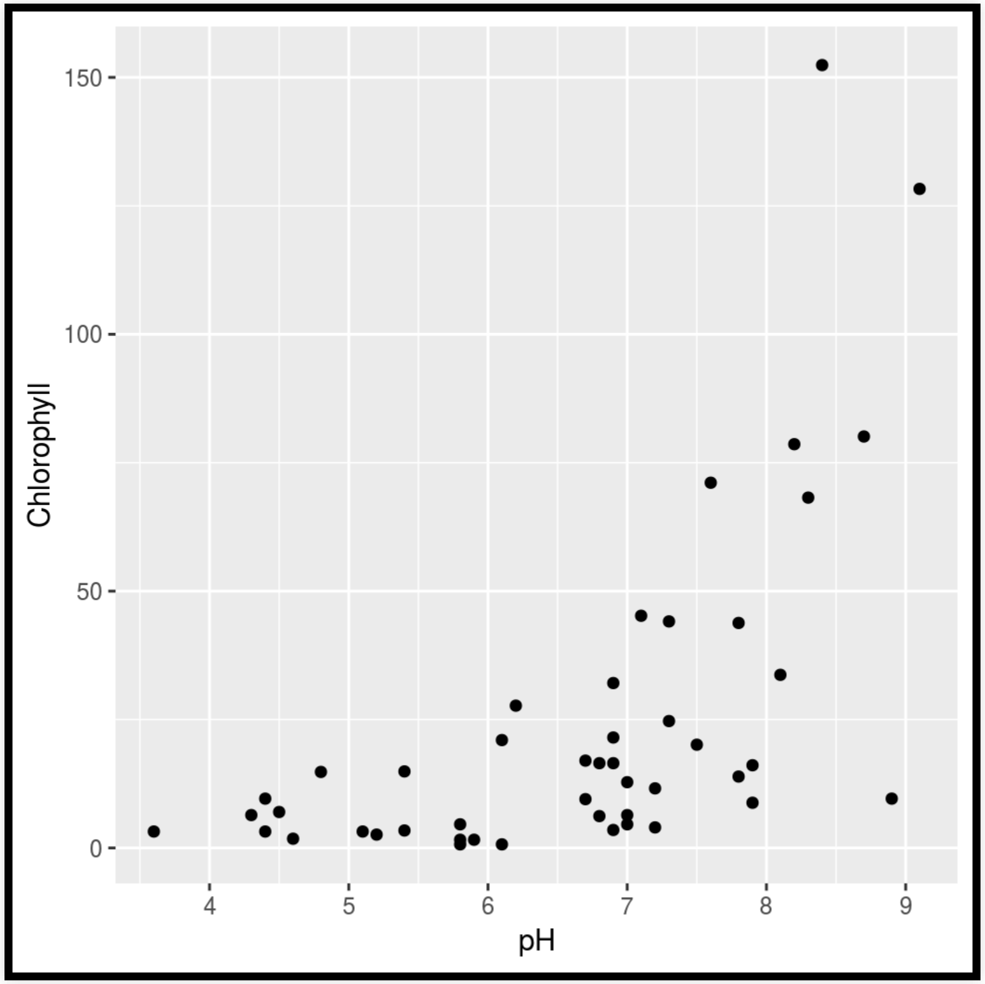
Creating a scatterplot

Now that we have initialized our canvas, we will tell **R** that we want to add (using `+`) points (`geom_point()`) at the coordinates to create a scatterplot.

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point()
```

Creating a scatterplot

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point()
```



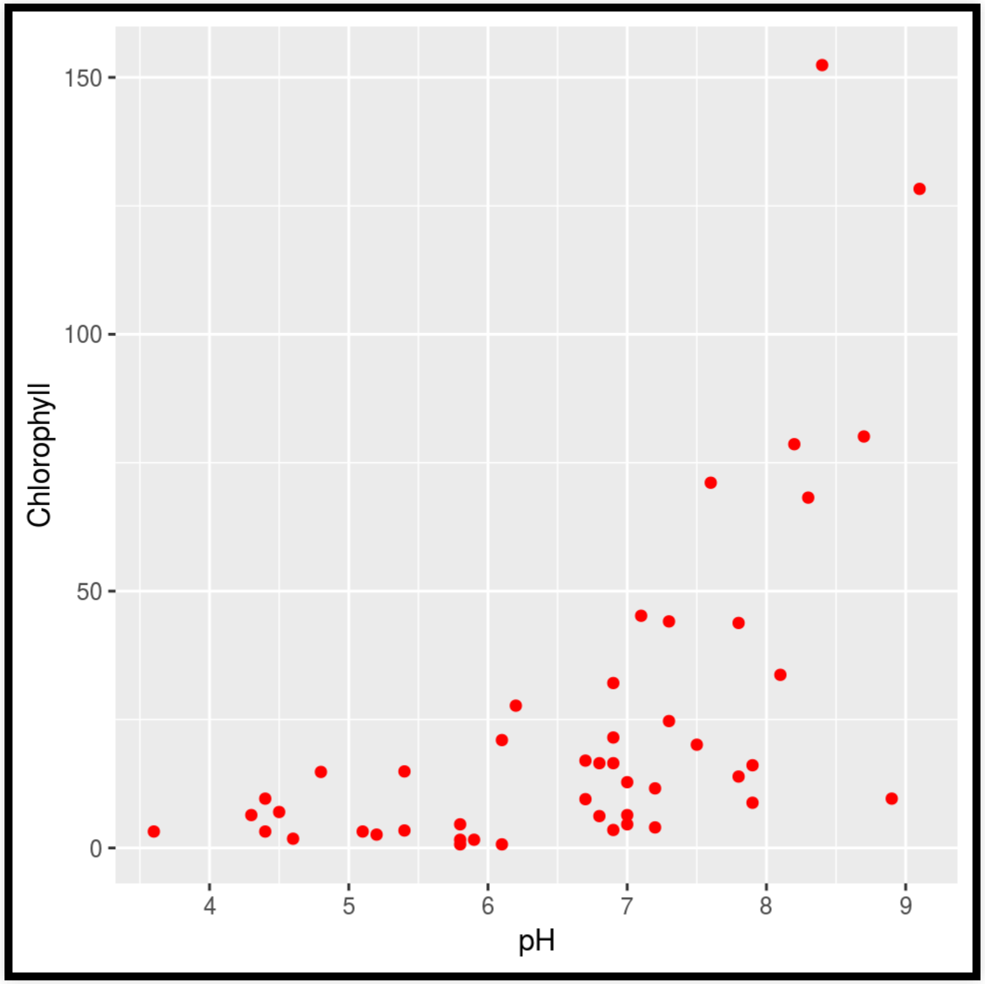
Changing the colour manually

We can change the colour of the points easily using the `colour` argument:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(colour = "red")
```

Changing the colour manually

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(colour = "red")
```



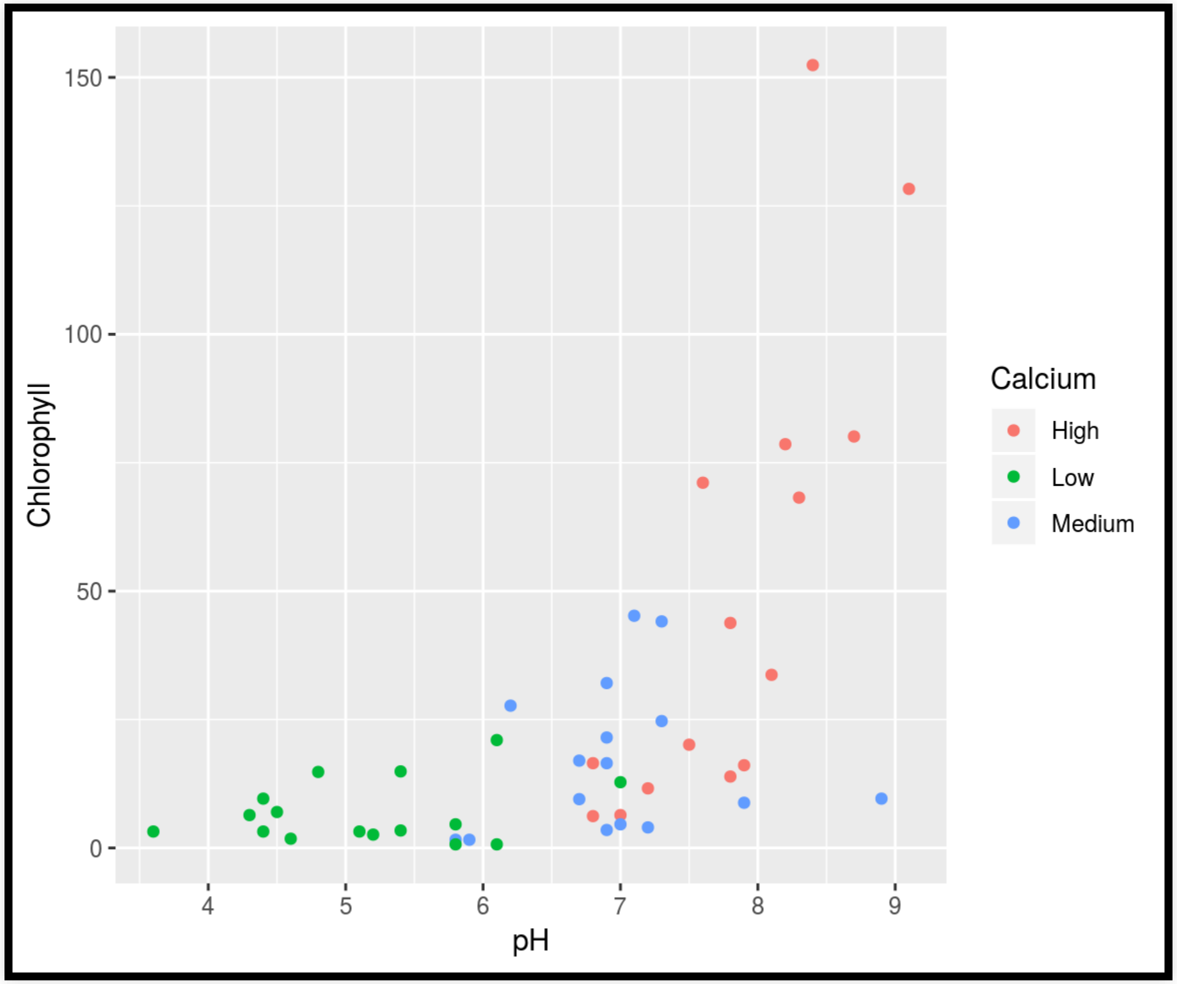
Changing the colour automatically

We can use a variable as the colour. When we do this, we need to put it inside `aes()`. For example, we can colour the points by calcium level:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium))
```

Changing the colour automatically

```
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium))
```



Changing the shape manually

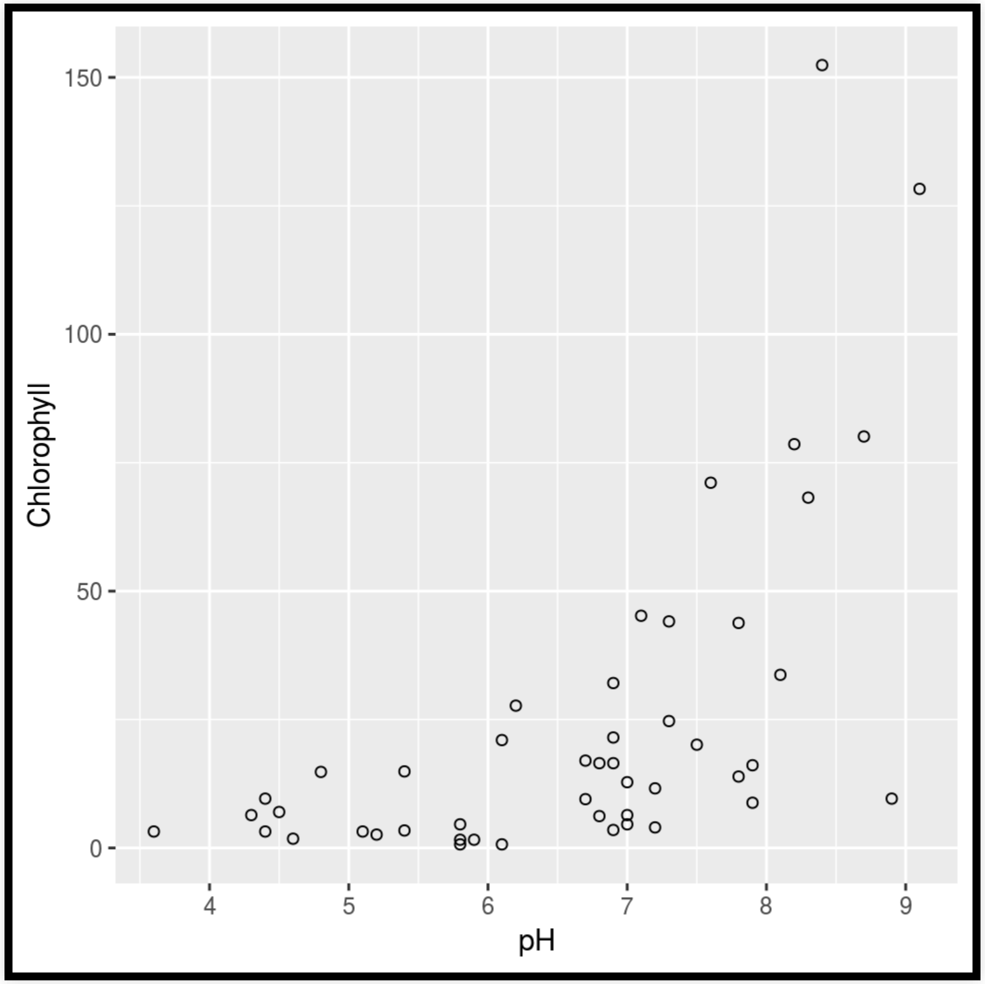
We can change the shape of the points easily using the `shape` argument:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(shape = 1)
```

See here for the various plotting symbols you can use.

Changing the shape manually

```
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(shape = 1)
```



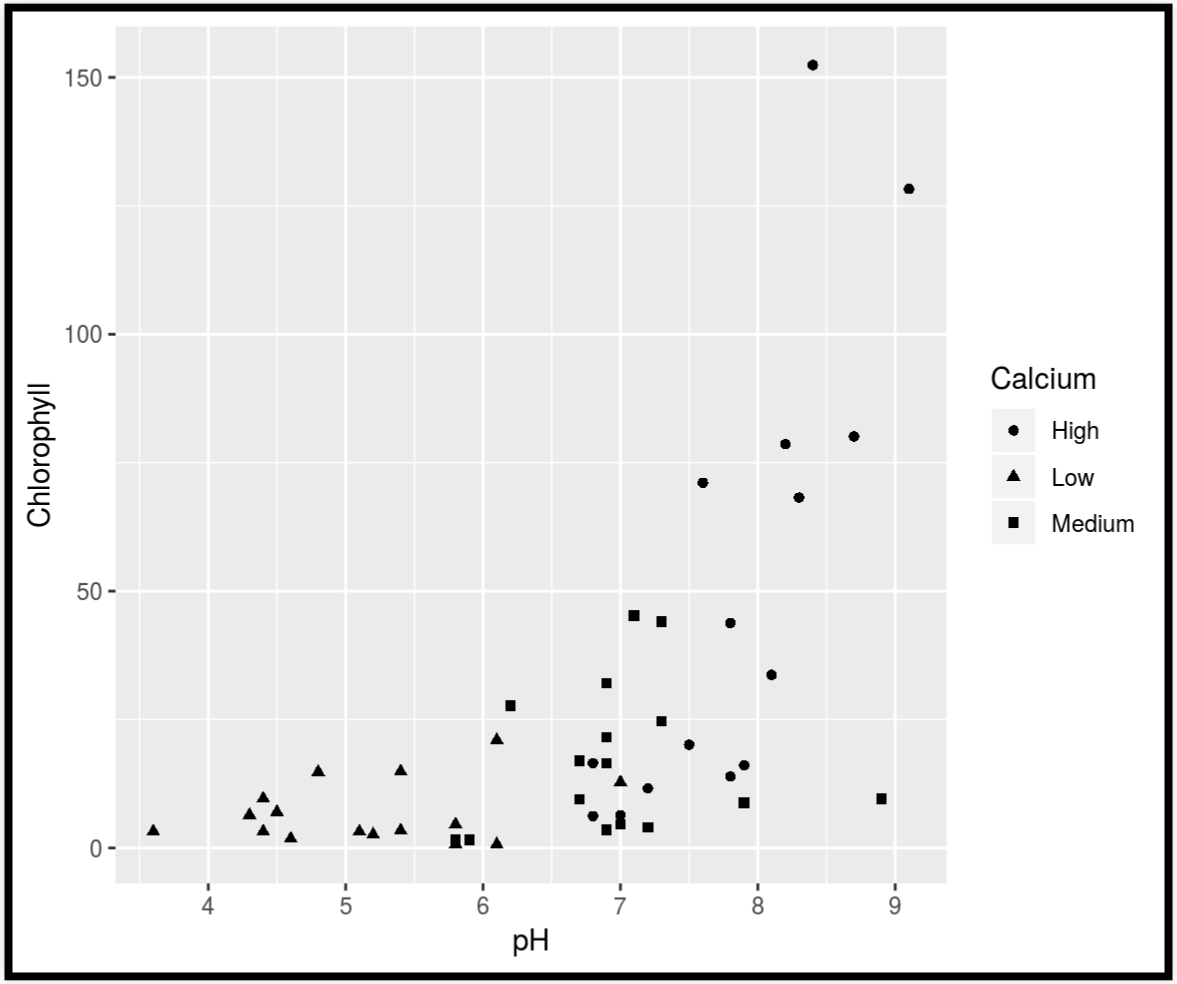
Changing the shape automatically

We can change the shape of the points to represent the 3 different calium levels:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(shape = Calcium))
```

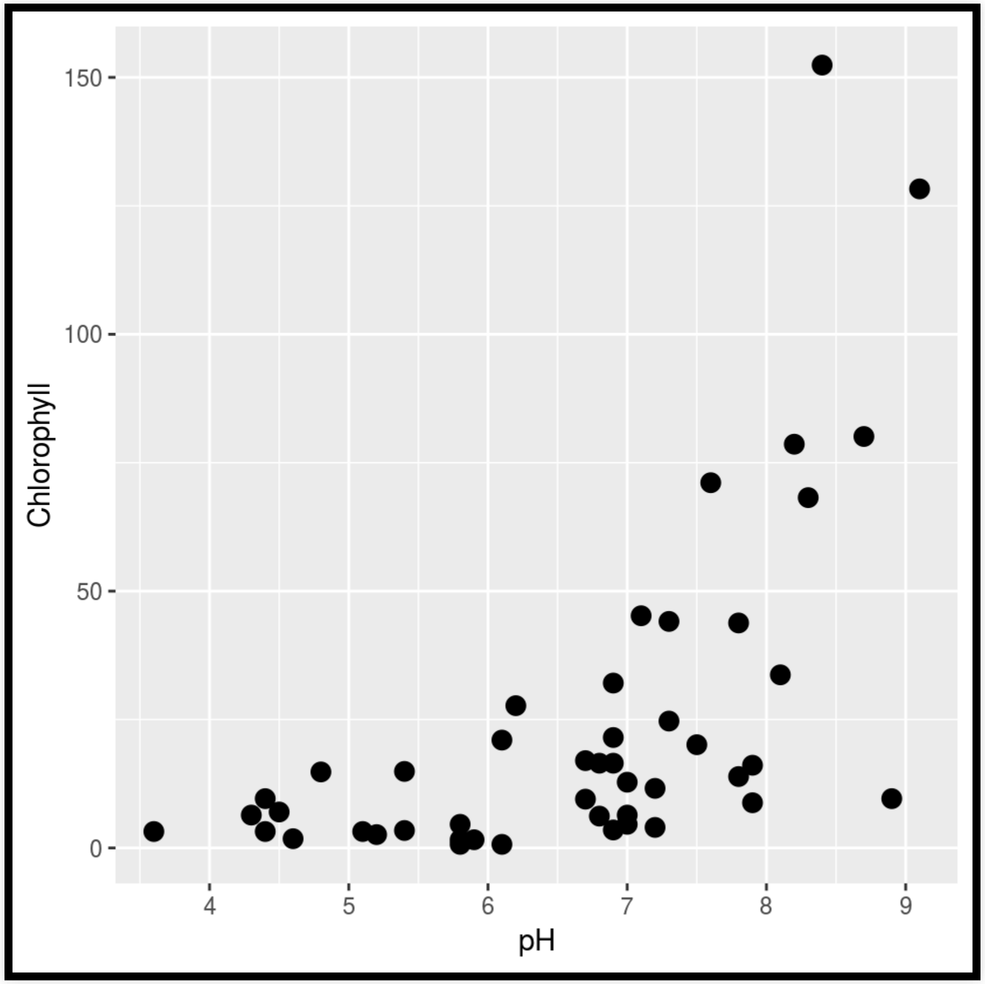
Changing the shape automatically

```
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(shape = Calcium))
```



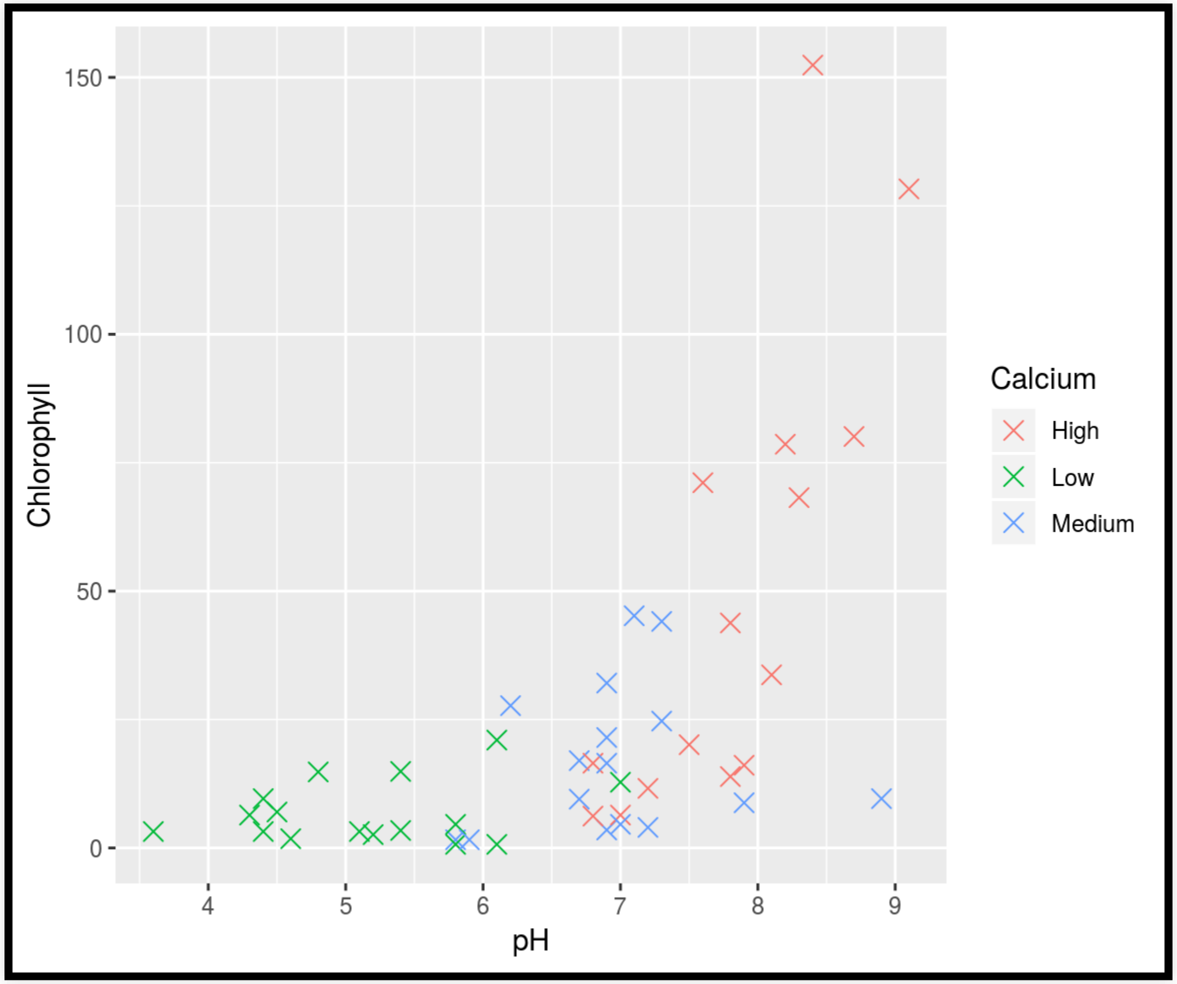
Changing the point size

```
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(size = 3)
```



Combining shape, colour and size

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4)
```

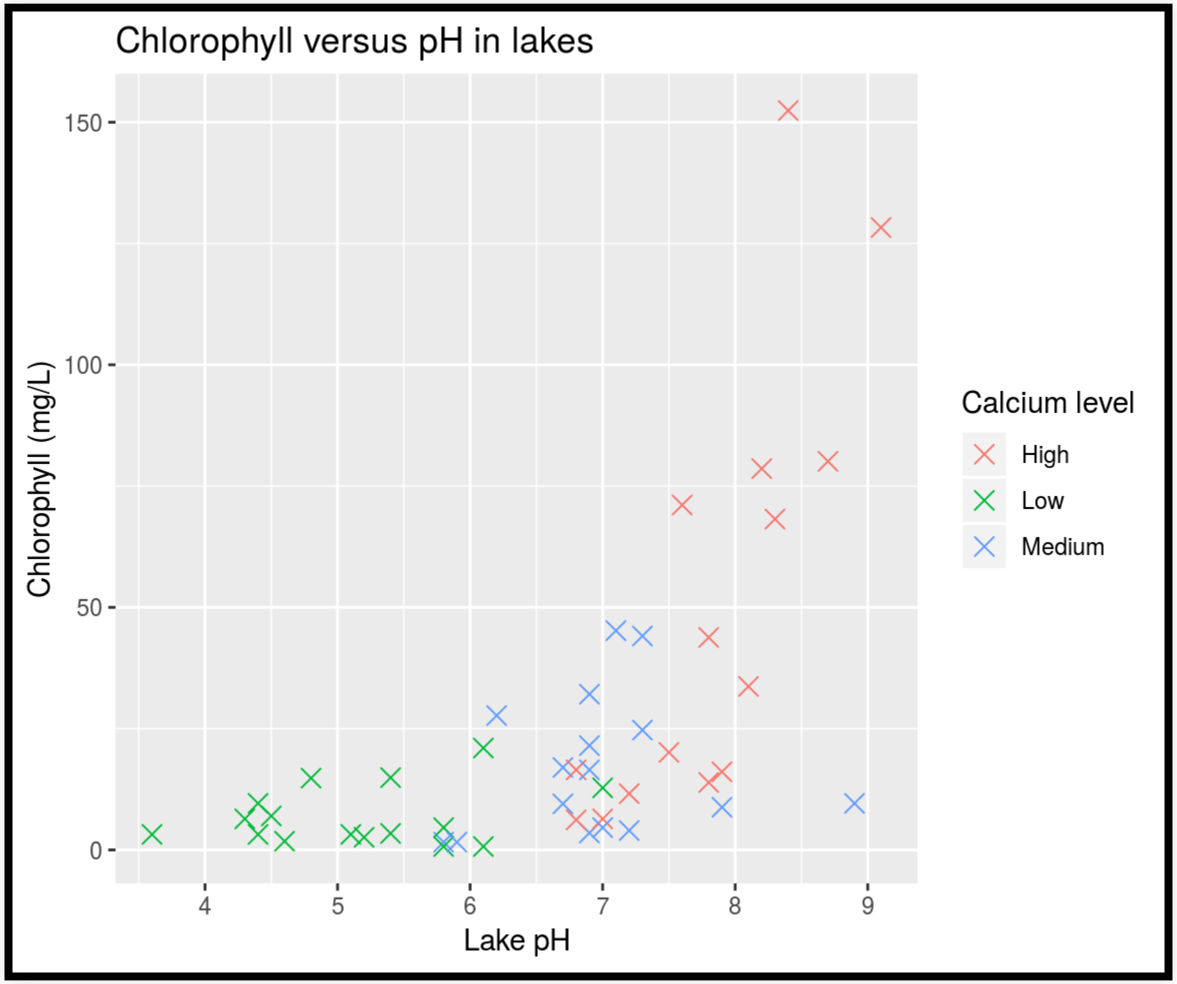


Modify labels

We can modify the labels of the plot by adding `labs()`:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4) +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       colour = "Calcium level")
```

Modify labels



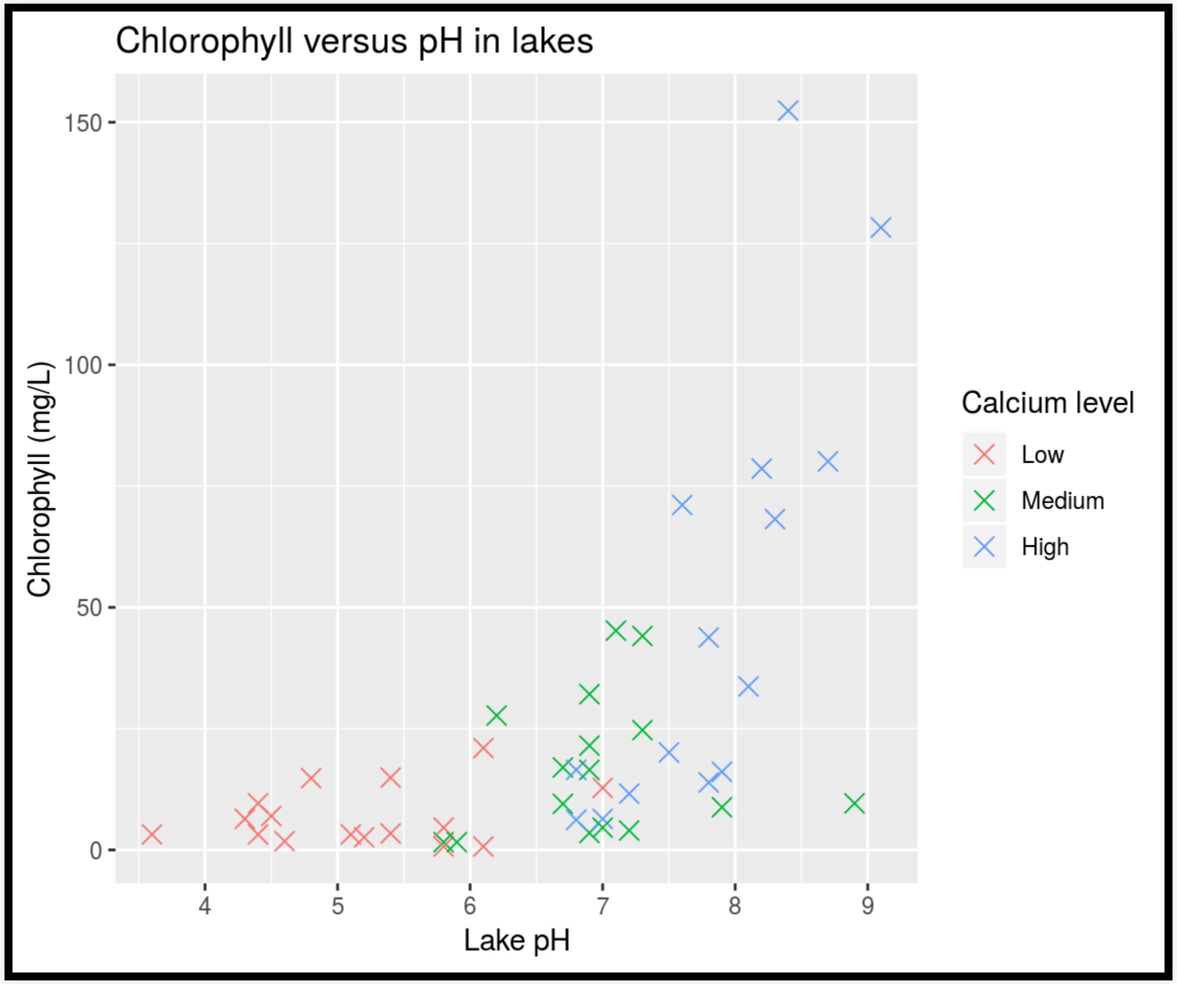
Modify the legend order

- **R** does not understand that “Medium” naturally comes after “Low” and before “High”.
- **R** coerces *character* vectors to *factors*, which uses alphabetical ordering of the levels by default.
- We have to manually change the `lake.df$Calcium` *character* vector into a *factor* and specify the order of the levels ourselves:

```r
# Change Calcium variable to a factor and specify levels
lake.df$Calcium = factor(lake.df$Calcium,
                         levels = c("Low", "Medium", "High"))

# Check the levels
levels(lake.df$Calcium)
```

```
#R:   [1] "Low"     "Medium" "High"
```

Modify the legend order

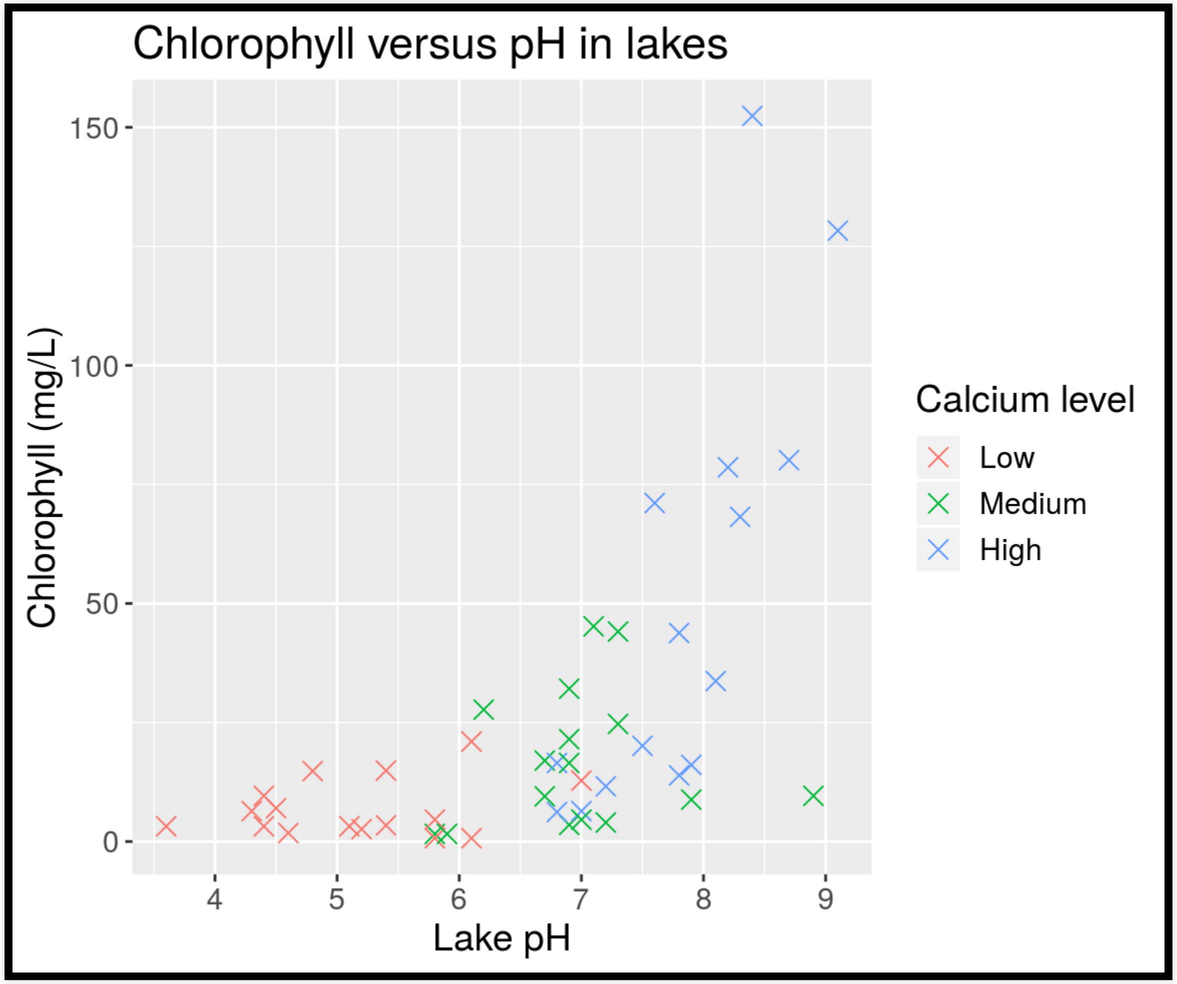


Modify components of the theme

We can modify many other components of this plot by adding `theme()` to our code:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4) +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       colour = "Calcium level") +
  theme(text = element_text(size = 14))
```

Modify components of the theme



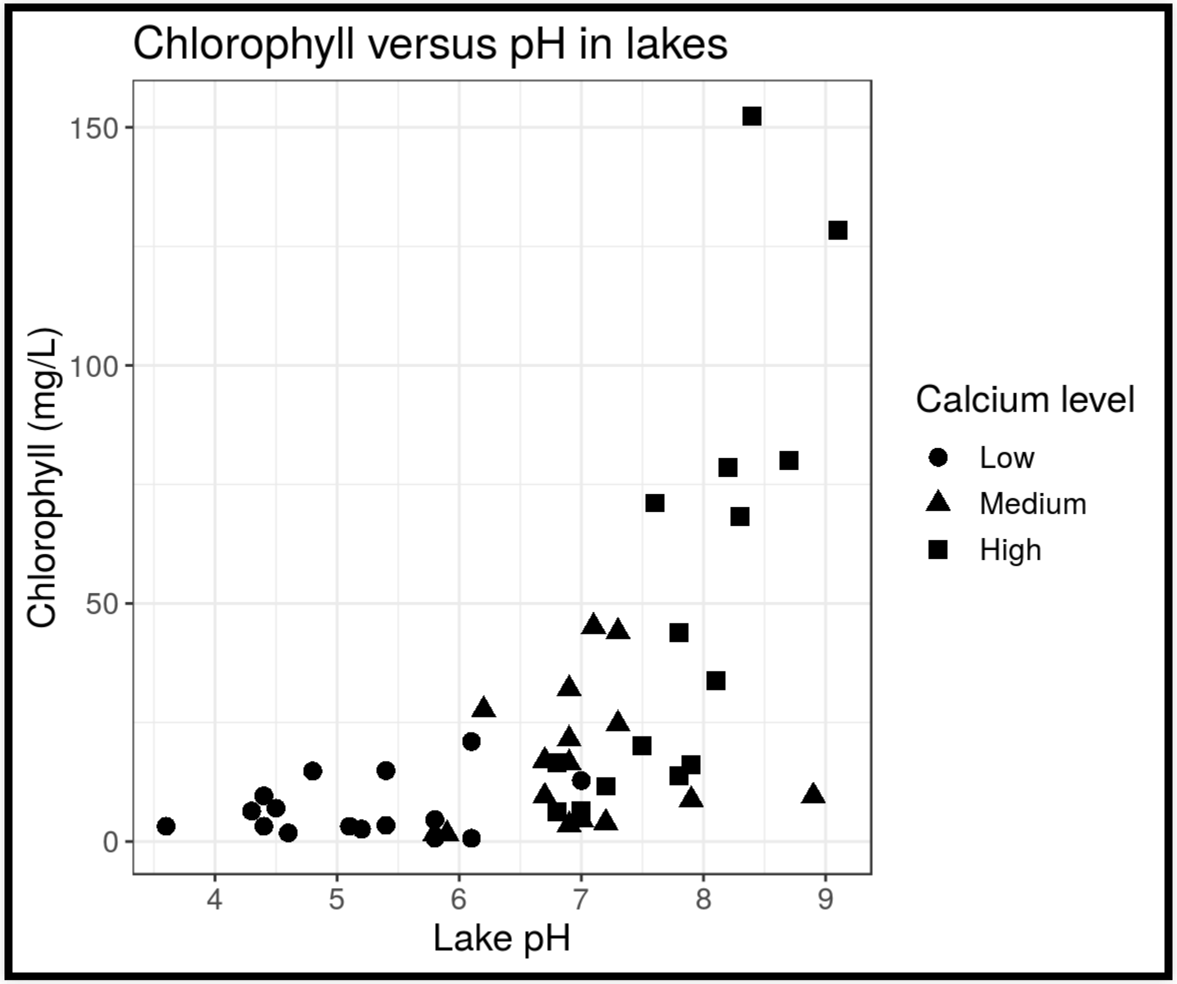
Modify the entire theme

You may prefer a black and white figure, rather than colour. We can change the entire theme to a black and white one easily by adding `theme_bw()`, and using `shape` instead of `colour` to differentiate the calcium levels:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(shape = Calcium), size = 3) +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       shape = "Calcium level") +
  theme_bw() +
  theme(text = element_text(size = 14))
```

Check out the complete list of available pre-defined themes here.

Modify the entire theme

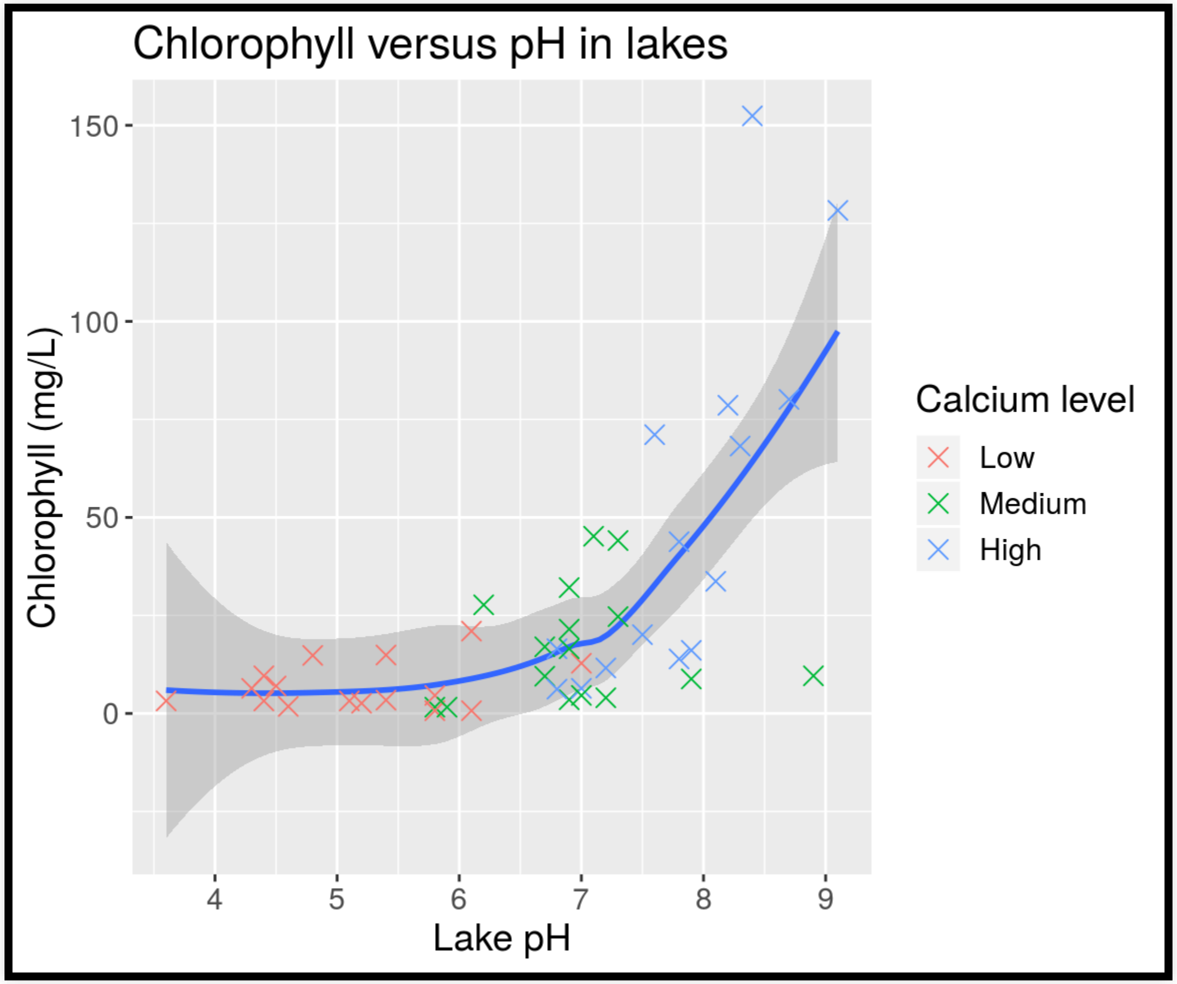


Add a loess smooth

We can easily add a smooth to a plot using `geom_smooth()`, which defaults to a loess smooth (`method = "loess"`) with a standard error (`se = TRUE`) region.

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4) +
  geom_smooth() +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       colour = "Calcium level") +
  theme(text = element_text(size = 14))
```

Add a loess smooth

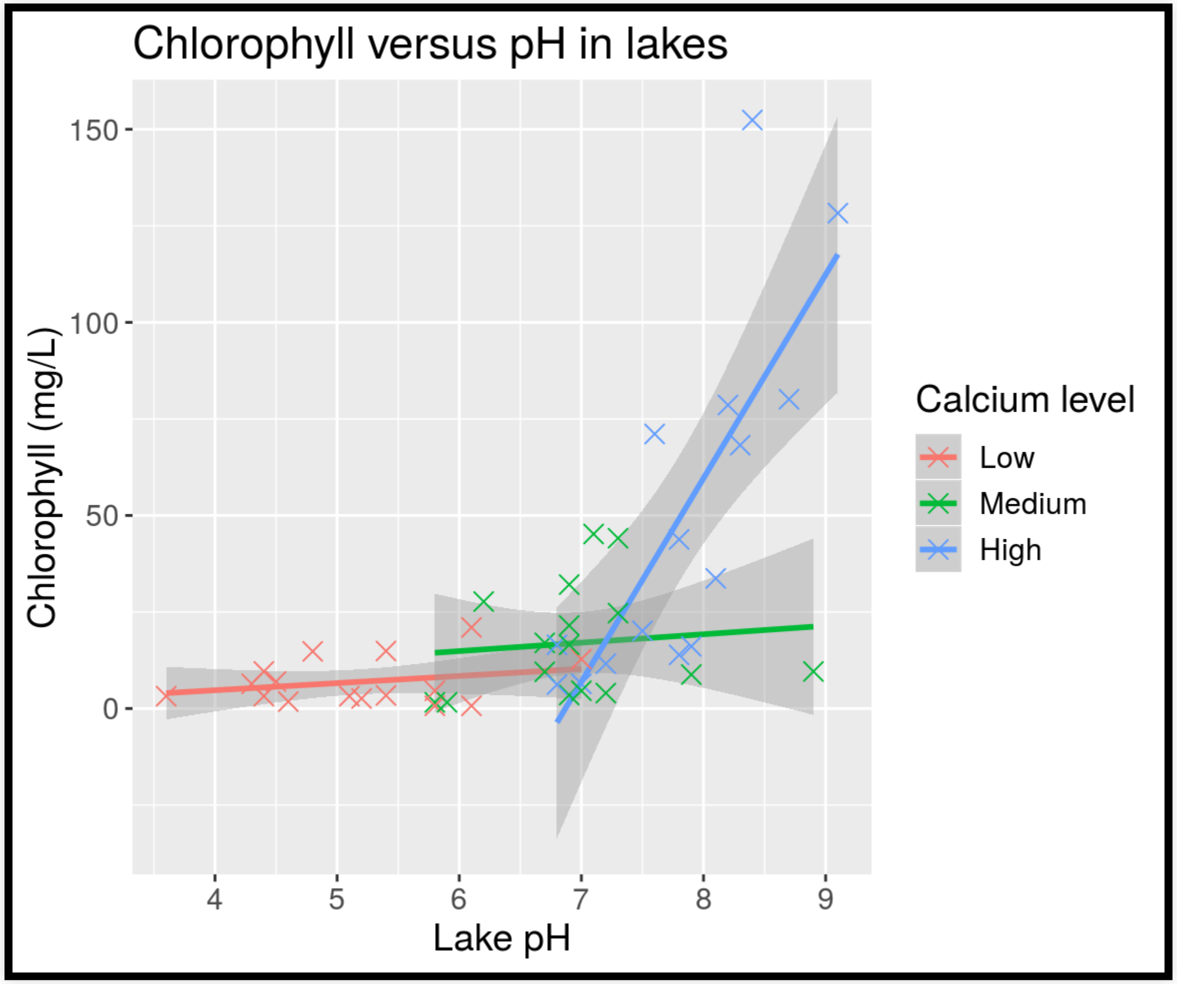


Multiple linear smooths

- Linear smooths can be produced with `method = "lm"` (instead of `"loess"`).
- Can include smooths for each calcium level by adding `aes(colour = Calcium)`.

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4) +
  geom_smooth(method = "lm", aes(colour = Calcium)) +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       colour = "Calcium level") +
  theme(text = element_text(size = 14))
```

Multiple linear smooths

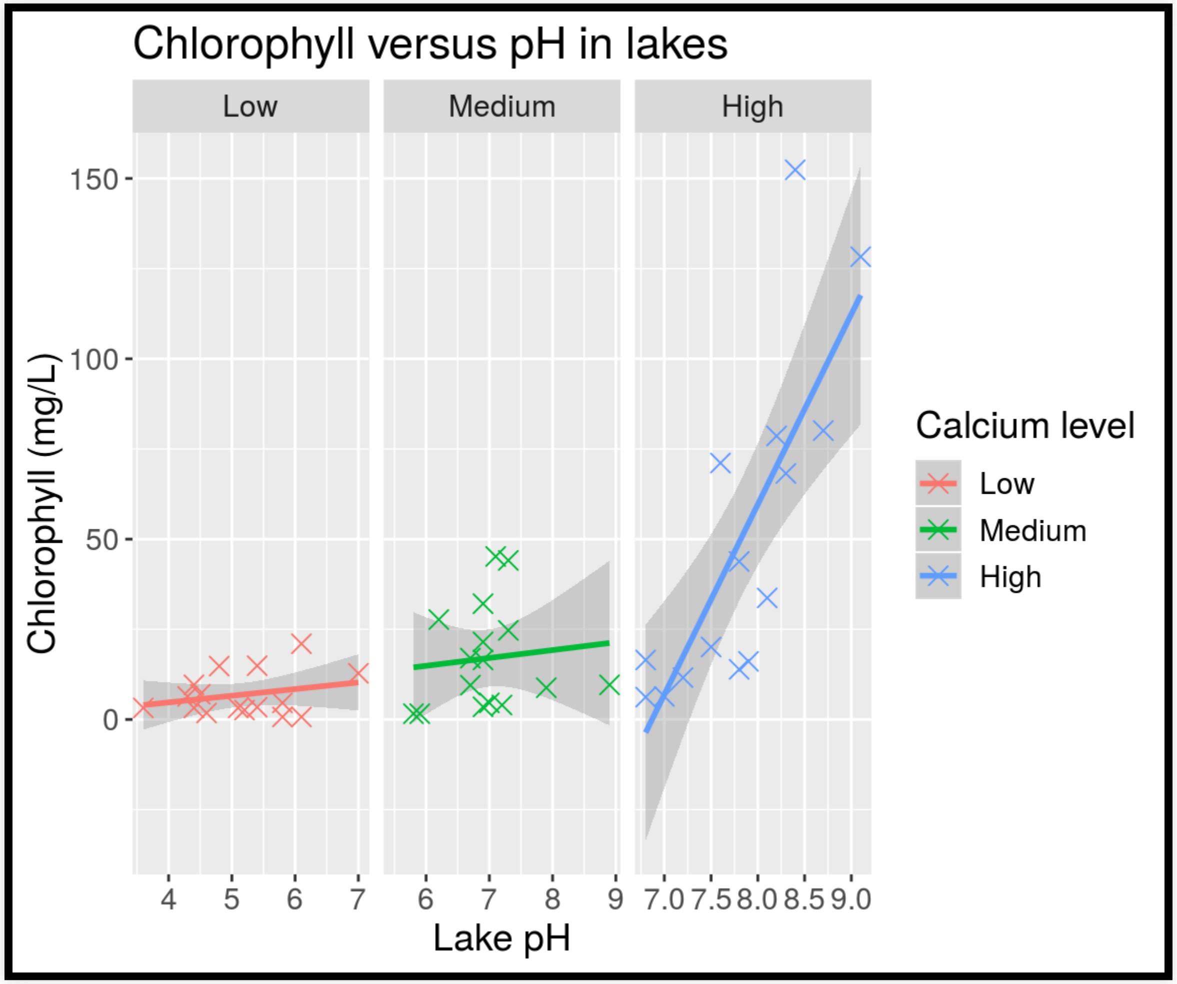


Faceting

A single plot can be separated into facets depending on the level of a factor by adding `geom_wrap`:

```r
ggplot(data = lake.df,
       mapping = aes(x = pH, y = Chlorophyll)) +
  geom_point(aes(colour = Calcium), size = 3, shape = 4) +
  geom_smooth(method = "lm", aes(colour = Calcium)) +
  facet_wrap(~Calcium, scales = "free_x") +
  labs(x = "Lake pH", y = "Chlorophyll (mg/L)",
       title = "Chlorophyll versus pH in lakes",
       colour = "Calcium level") +
  theme(text = element_text(size = 14))
```

Faceting



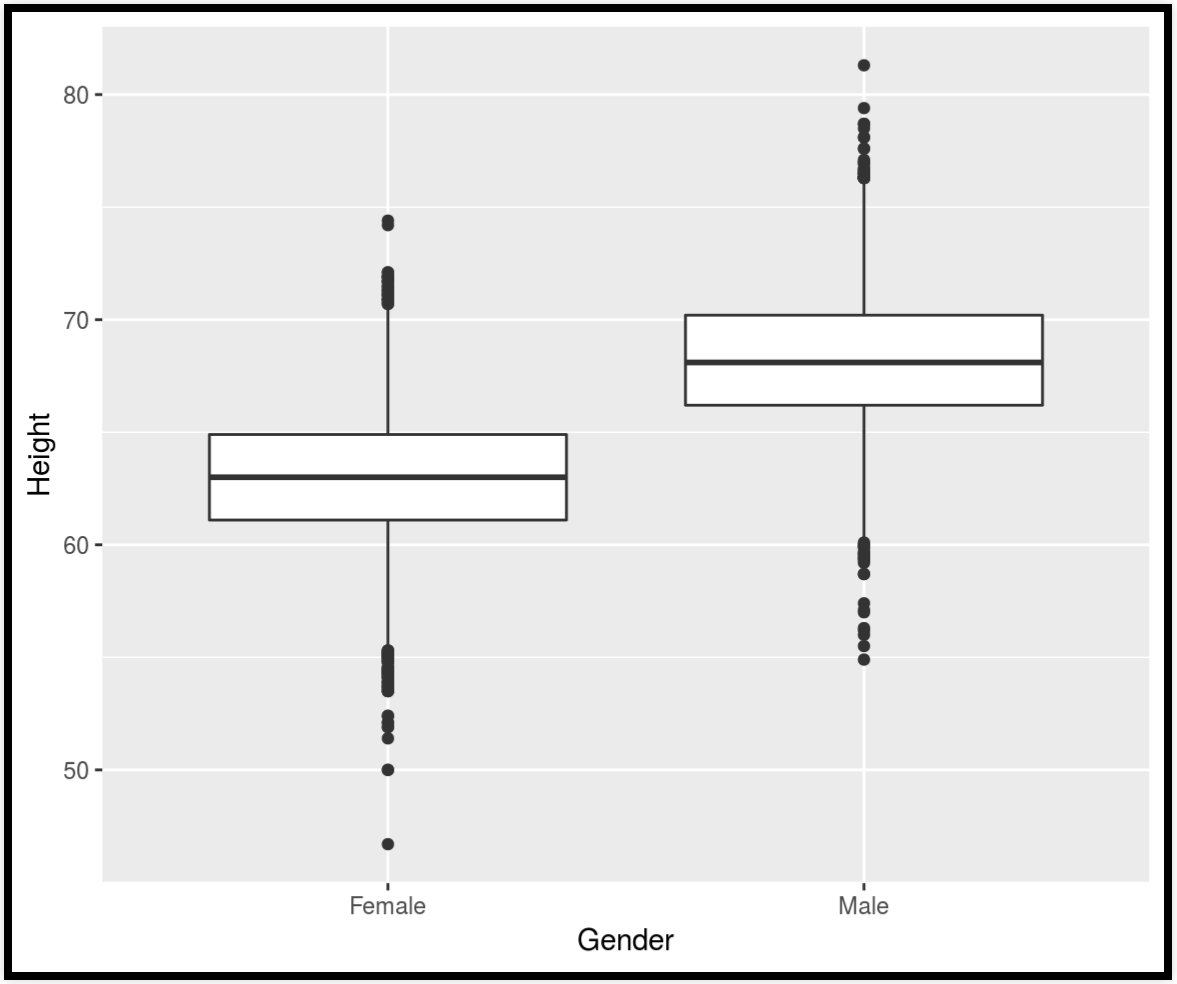
Chlorophyll versus pH in lakes
Low
Medium
High
150
100
50
0
Chlorophyll (mg/L)
4
5
6
7
6
7
8
9
7.0
7.5
8.0
8.5
9.0
Lake pH
Calcium level
Low
Medium
High

Boxplots

- We need to have a discrete variable on the x axis, and a continuous variable on the y axis.
- The code is very similar, except we now use `geom_boxplot`, instead of `geom_point`:

```r
ggplot(data = patient.df,
       mapping = aes(x = Gender, y = Height)) +
  geom_boxplot()
```

Boxplots

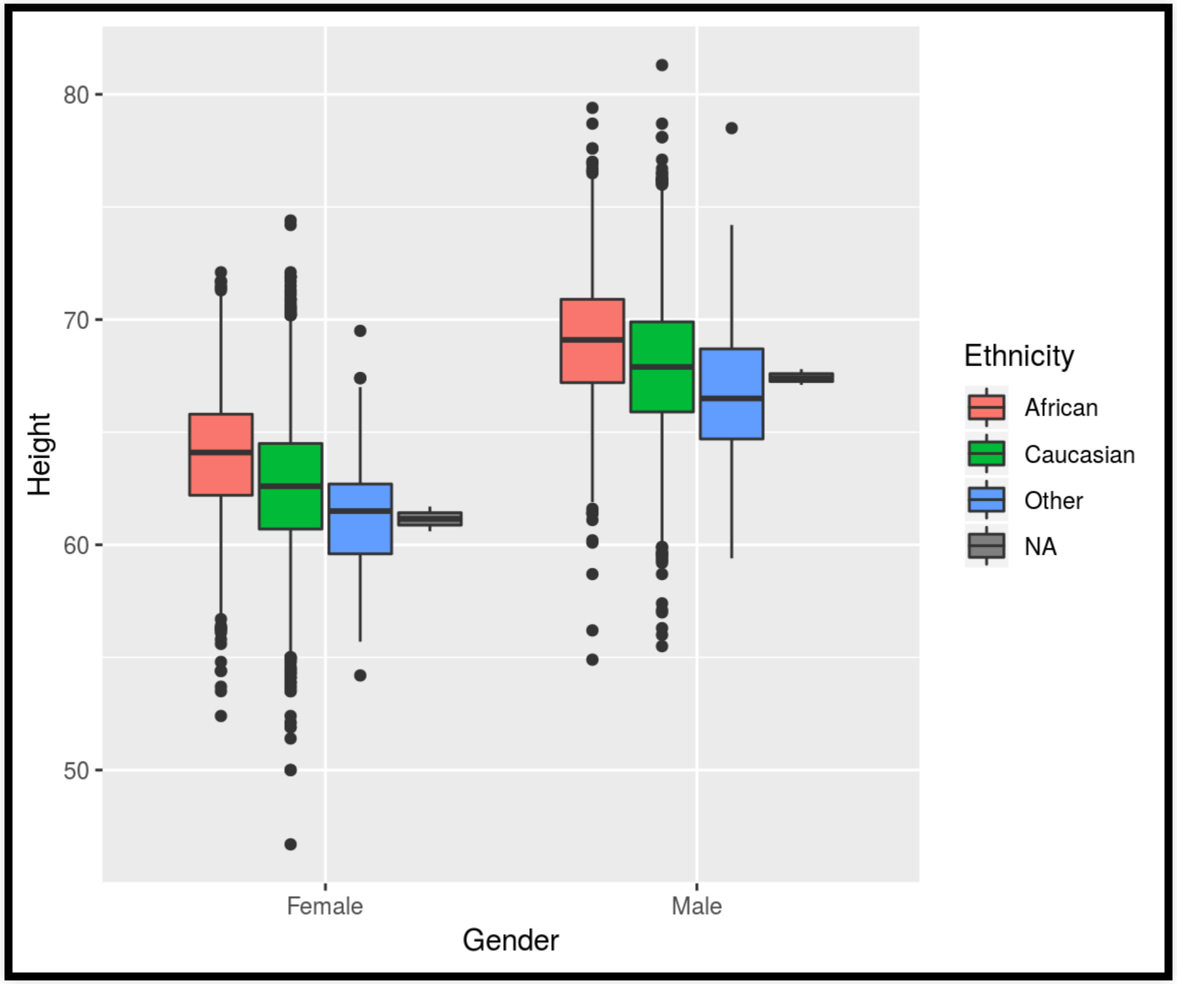


Side-by-side boxplots

- We can use the `fill` (or `colour`) argument to create side-by-side boxplots:

```r
ggplot(data = patient.df,
       mapping = aes(x = Gender, y = Height)) +
  geom_boxplot(aes(fill = Ethnicity))
```

Side-by-side boxplots

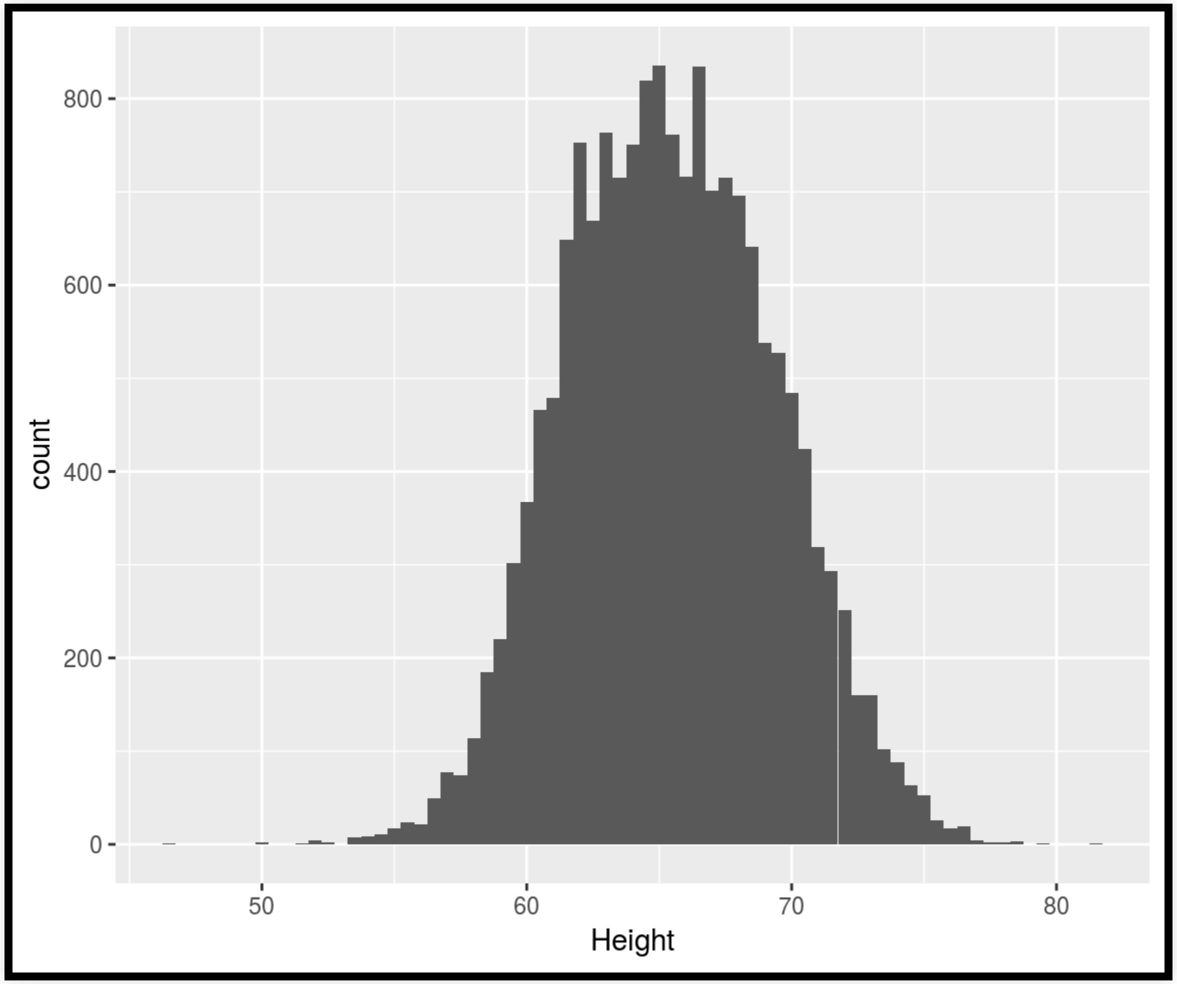


Histogram

- A histogram (`geom_histogram`) only needs an `x` aesthetic.
- You can change the binwidth with the `binwidth` argument.

```
ggplot(data = patient.df,
       mapping = aes(x = Height)) +
  geom_histogram(binwidth = .5)
```

Histogram

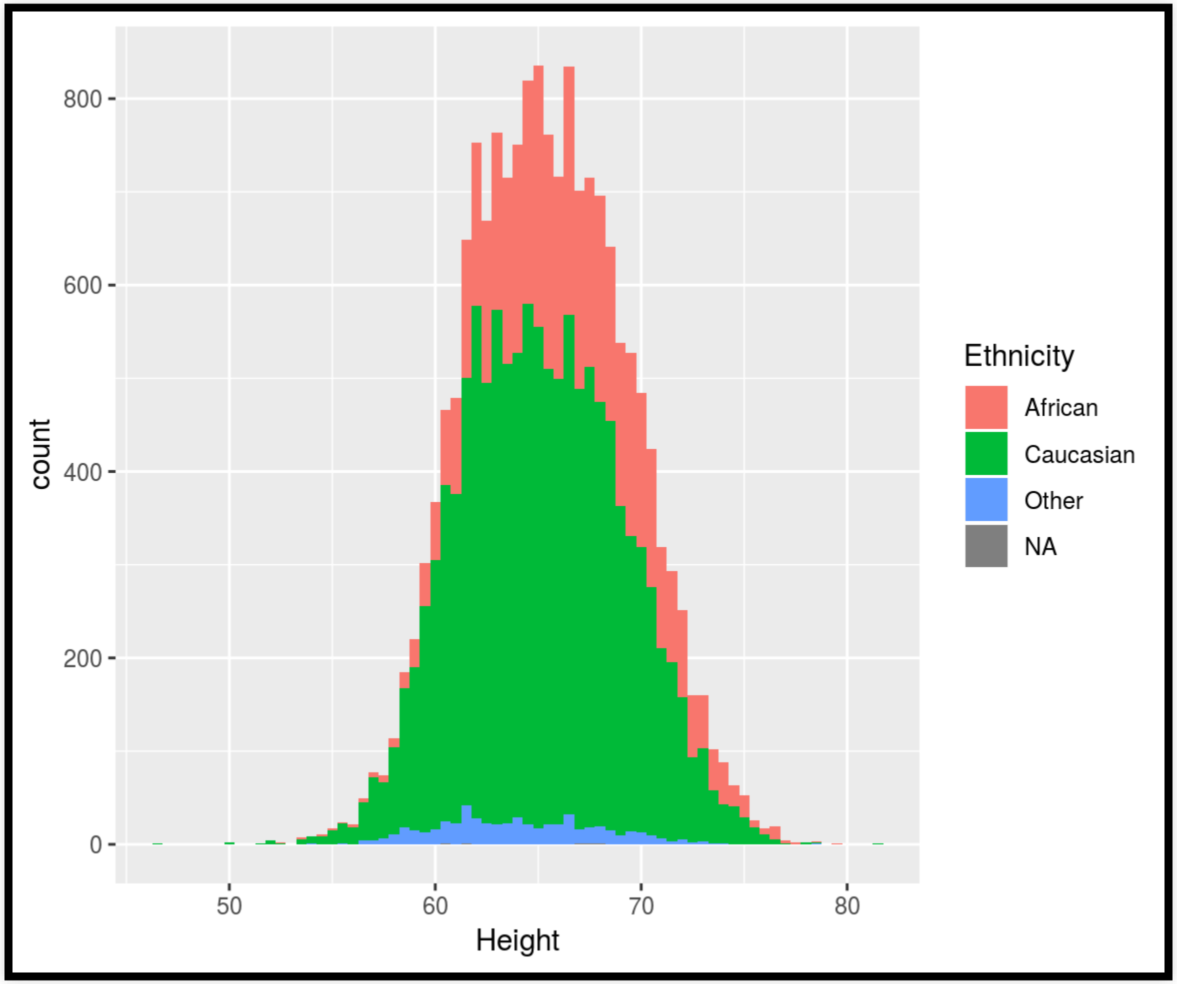


Histogram with `fill`

- We can use `fill` for histograms as well:

```
ggplot(data = patient.df,
       mapping = aes(x = Height)) +
  geom_histogram(aes(fill = Ethnicity), binwidth = .5)
```

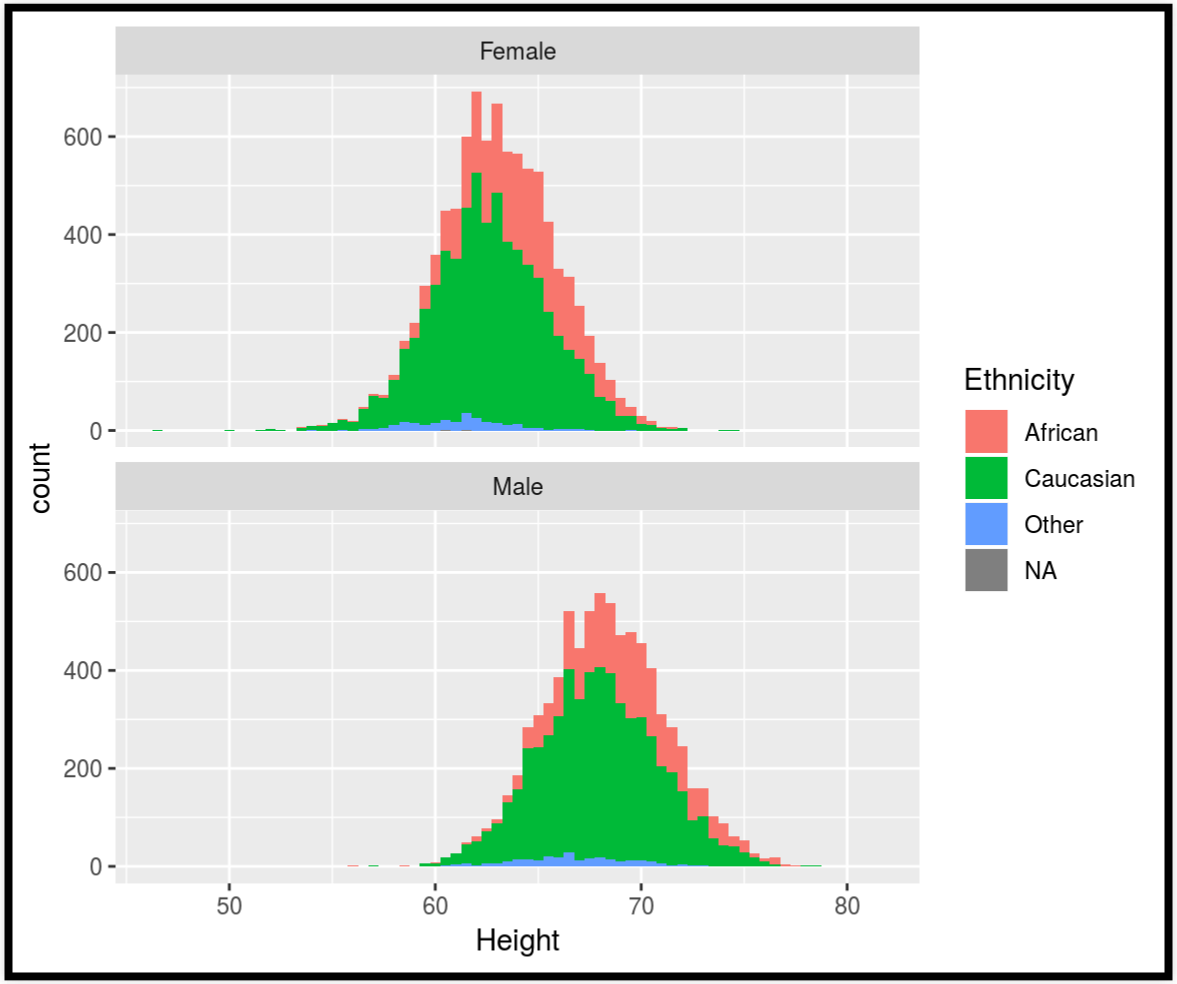
Histogram with `fill`



Faceted histogram with `fill`

```r
ggplot(data = patient.df,
       mapping = aes(x = Height)) +
  geom_histogram(aes(fill = Ethnicity), binwidth = .5) +
  facet_wrap(~Gender, ncol = 1)
```

Faceted histogram with `fill`



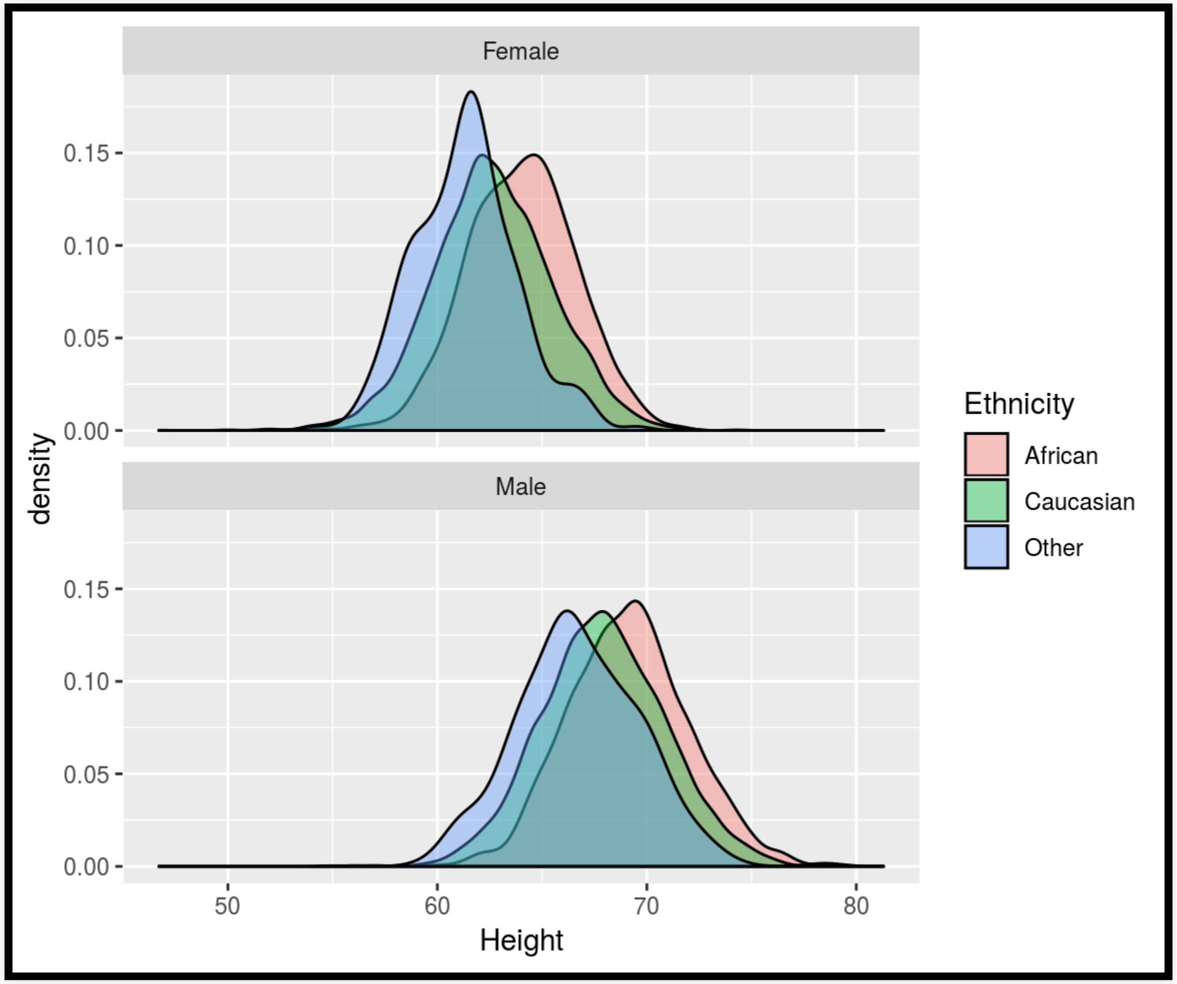
Faceted density plot with `fill`

We can substitute `geom_histogram()` from the previous code with `geom_density()` to create a density plot.

- Transparency is controlled with the `alpha` argument

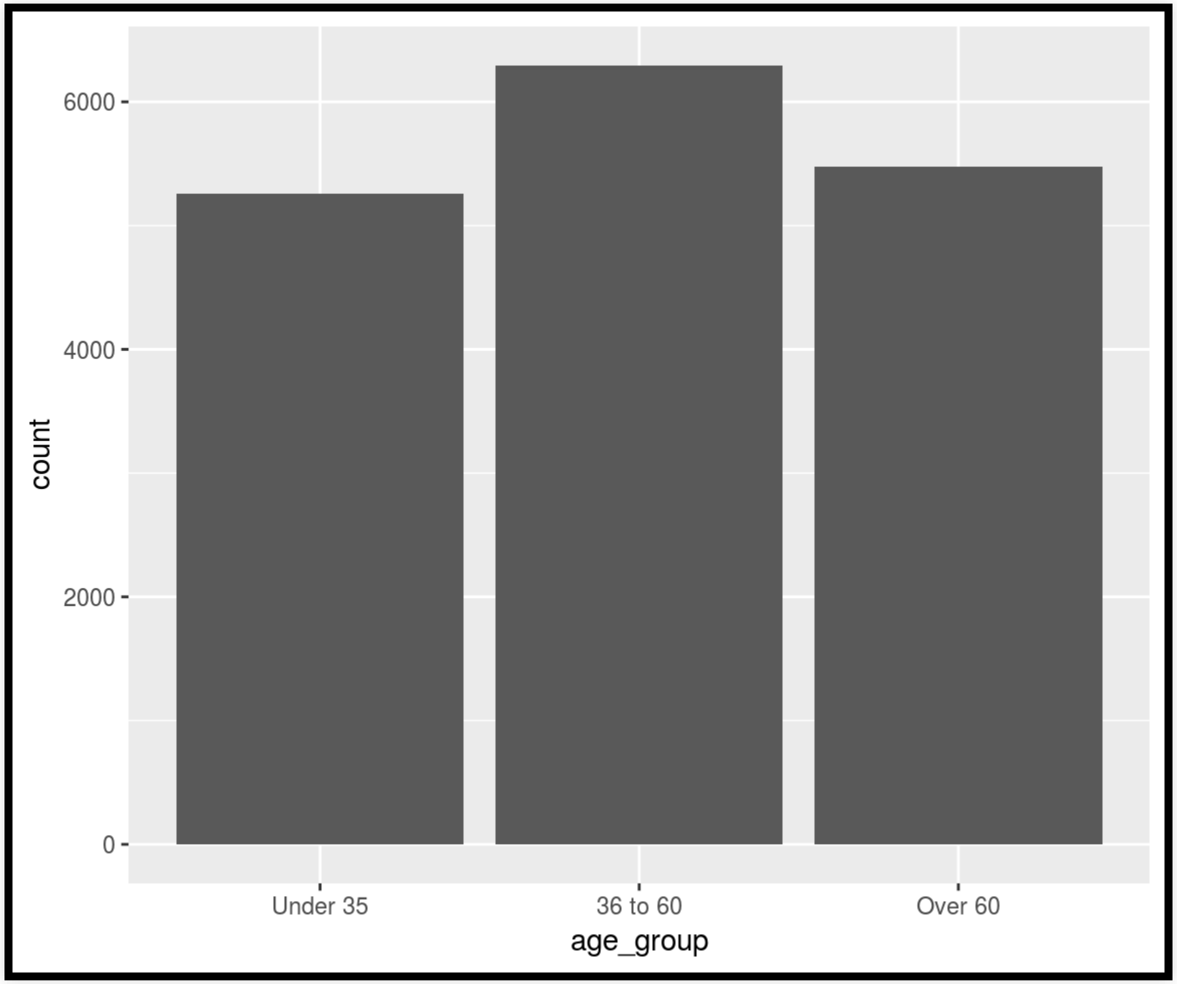
```r
# Remove missing values from Ethnicity
patient_2.df = subset(patient.df, !is.na(Ethnicity))

ggplot(data = patient_2.df,
       mapping = aes(x = Height)) +
  geom_density(aes(fill = Ethnicity), alpha = .4) +
  facet_wrap(~Gender, ncol = 1)
```

Faceted density plot with `fill`



Barplot

```
ggplot(data = patient.df,
       mapping = aes(x = age_group)) +
  geom_bar()
```

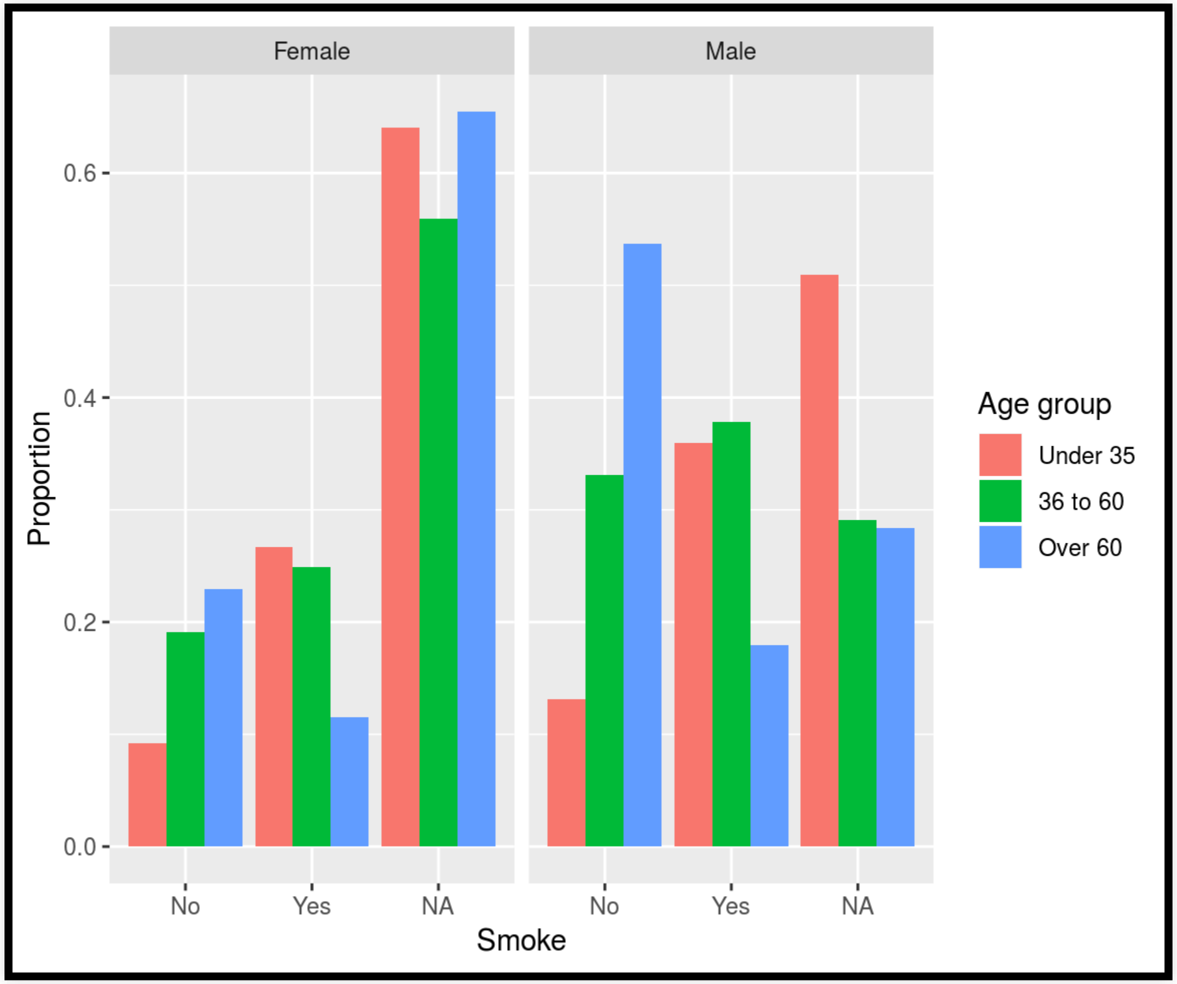


Faceted, side-by-side barplot of proportions

- The proportions add to 1 within each age group (`group = age_group`) and each facet.

```r
ggplot(data = patient.df,
       mapping = aes(x = Smoke, y = ..prop.., group = age_group)) +
  geom_bar(aes(fill = age_group), position = "dodge") +
  facet_wrap(~Gender) +
  labs(y = "Proportion",
       fill = "Age group")
```

Faceted, side-by-side barplot of proportions



Summary

Plot type	Function
Scatterplot	`geom_point`
Barplot	`geom_bar`
Histogram	`geom_histogram`
Boxplot	`geom_boxplot`

The following can be added or changed in any plot:

- Facets
- Legends
- Themes
- Transparency
- Labels and other options