STAT 451 - Visualizing Data - Autumn 2025

Ariane Ducellier

10/07/2025

Tutorial Ggplot part 1

Today, we are going to focus on basic ggplot2 functions. For more on ggplot, I recommend this book:

Applied data visualization with R and ggplot2 : Create useful, elaborate, and visually appealing plots Moulik, Tania 2018; Birmingham, UK : Packt Publishing Ltd.

We will need the following R libraries:

```
library(tidyverse)
library(ggpubr)
library(gridExtra)
library(Lock5Data)
```

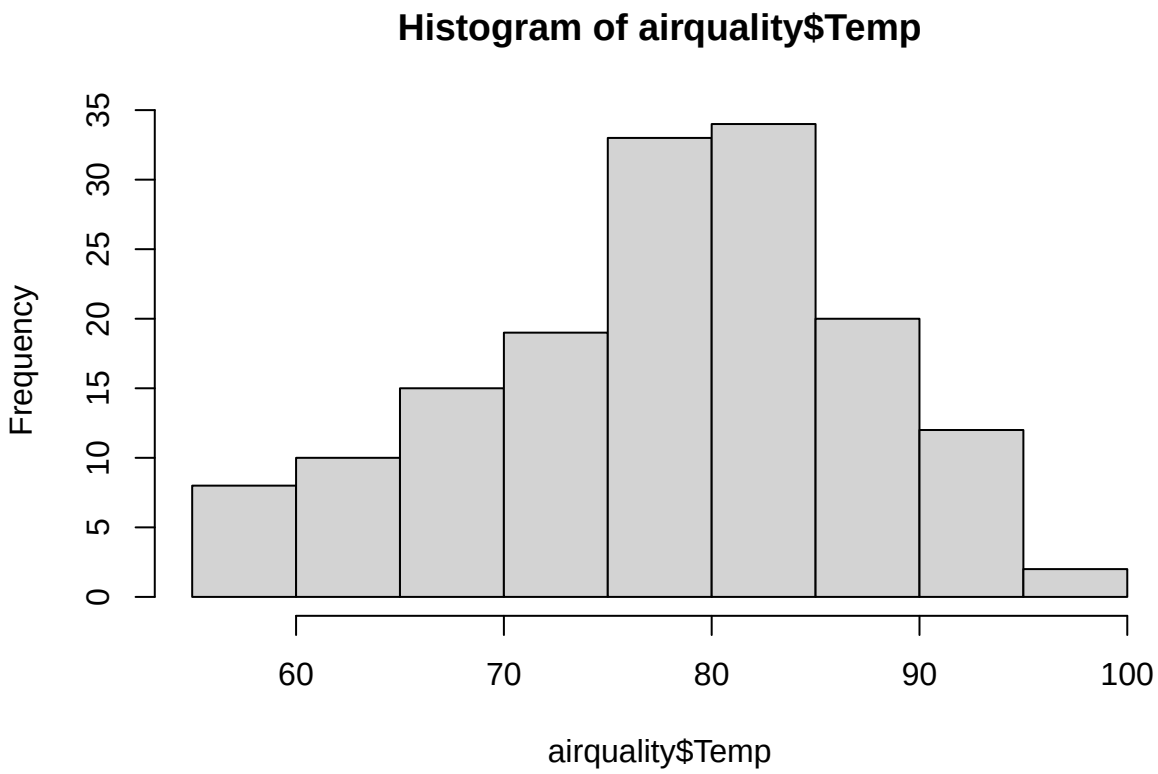
Basic plotting in ggplot2

We will first review basic types of data visualization.

Histograms

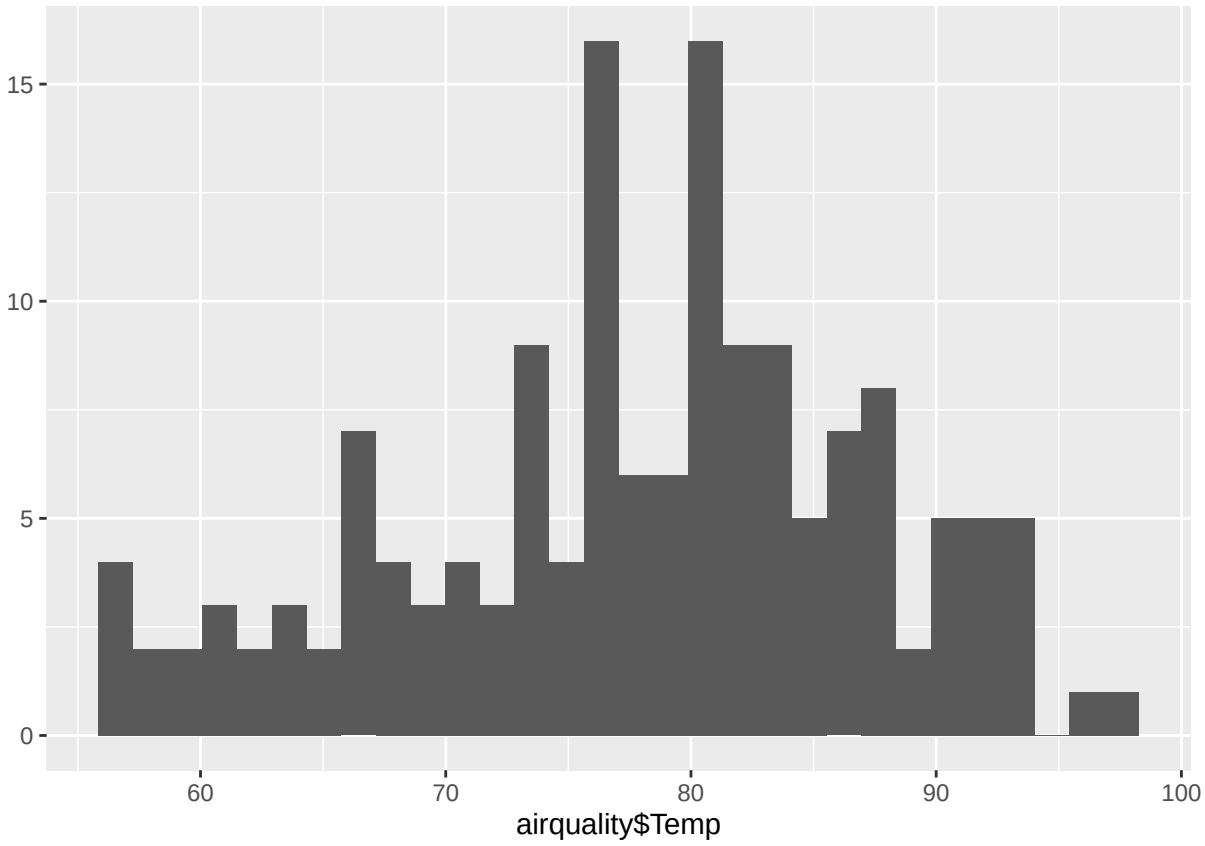
The default way on plotting with R is to call a plotting function and pass the data to be plotted as an argument.

```
hist(airquality$Temp)
```



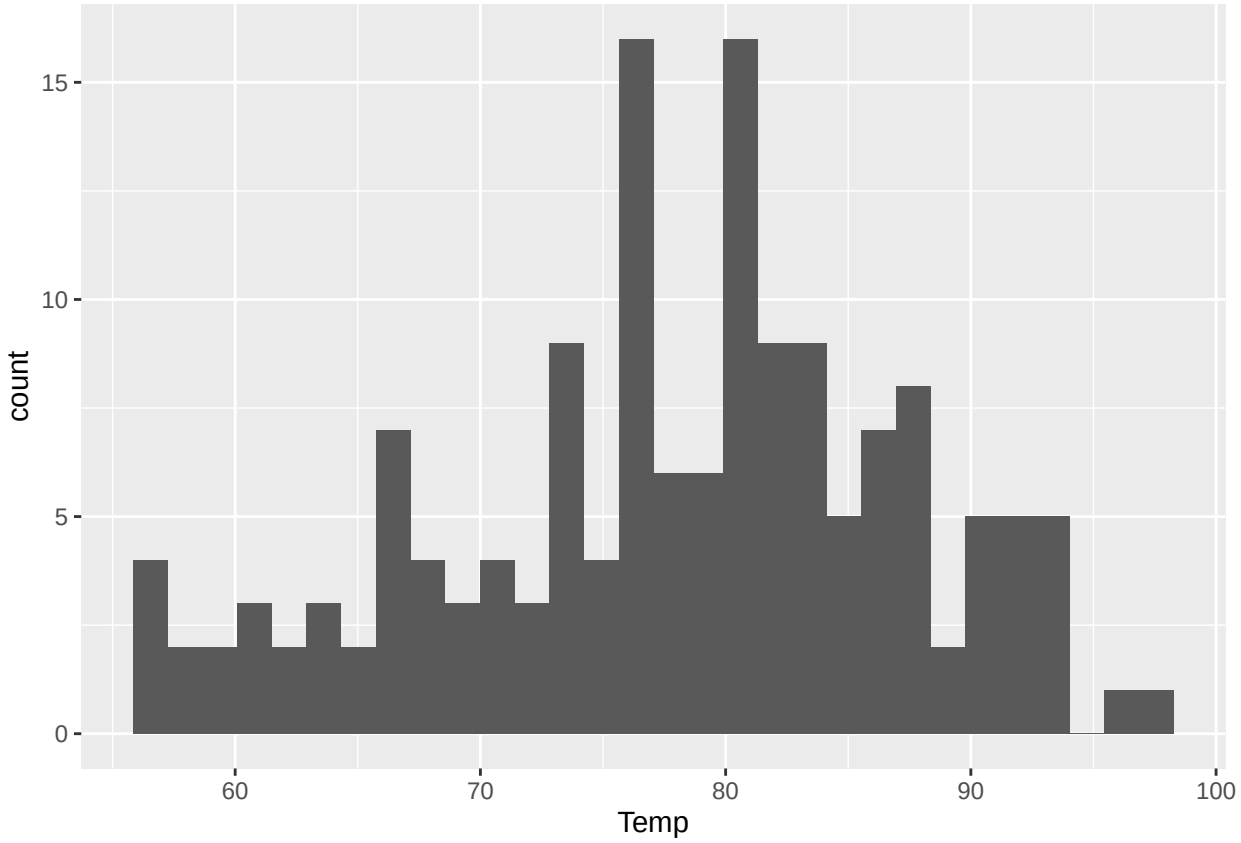
Similar (deprecated) basic functions exist in ggplot2:

```
qplot(airquality$Temp)
```



However, users are encouraged to use the full ggplot2 capabilities for their plots. Each plotting script starts by declaring the data set and the columns to be used for the plotting, and then call different ggplot2 functions to add elements to the data visualization.

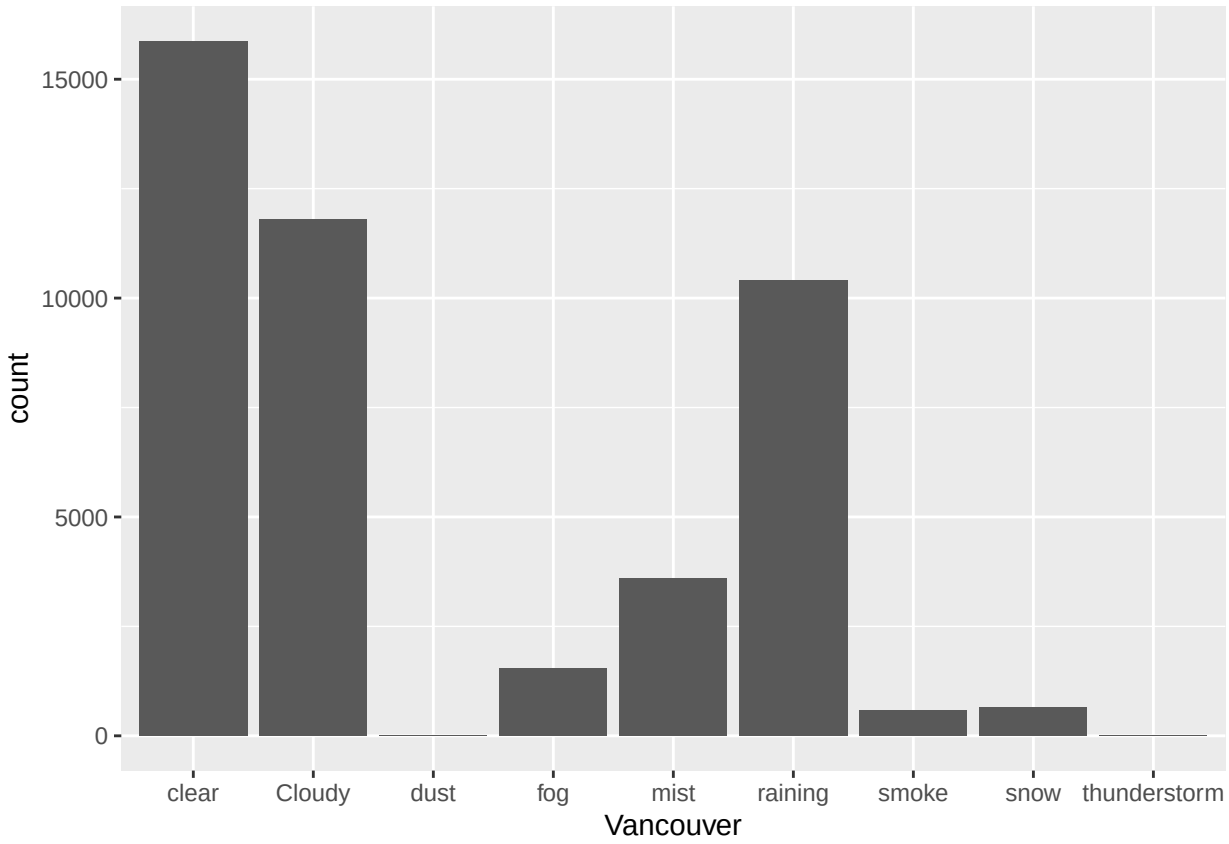
```
ggplot(airquality, aes(x=Temp)) +
  geom_histogram()
```



Bar plots

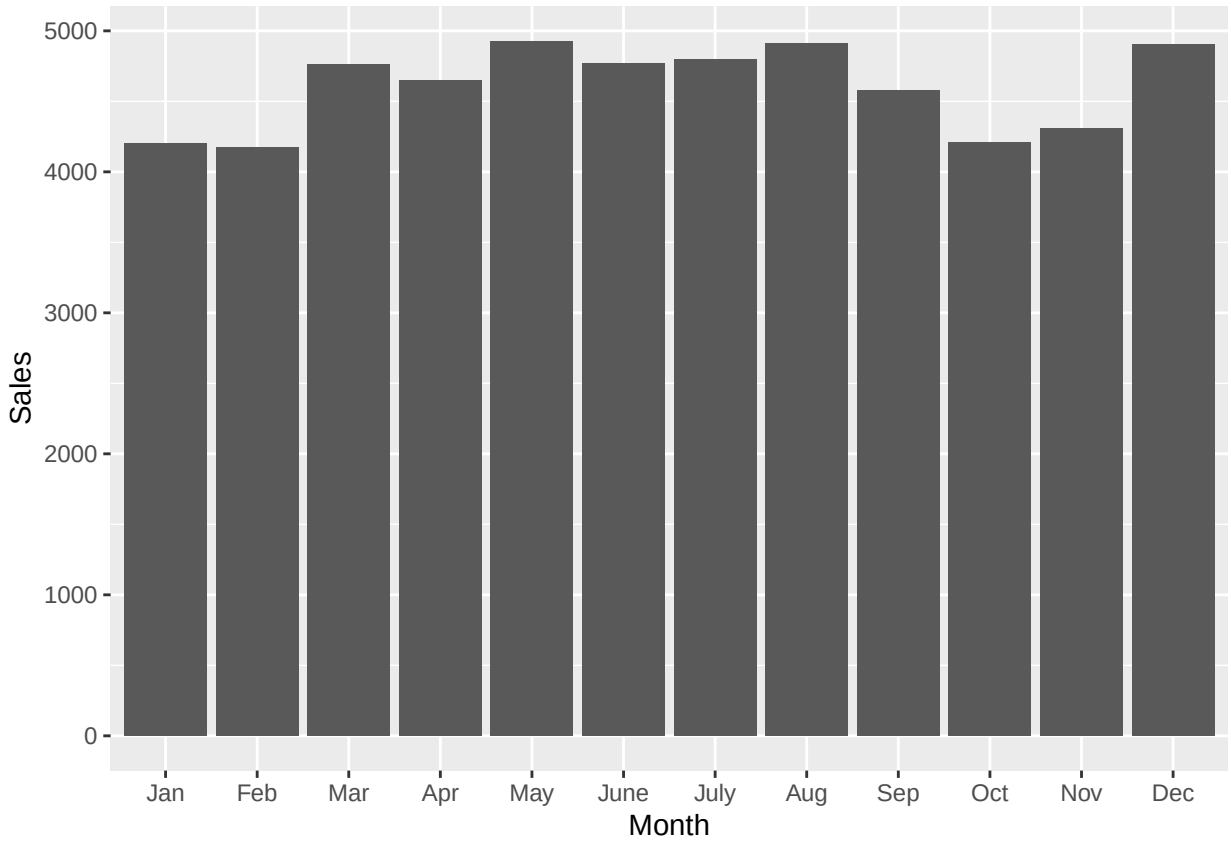
You may want to visualize the number of occurrences of each value of a categorical variable within a data set. You then only need to specify the x coordinates.

```
df_desc <- read_csv("../../data/historical-hourly-weather-data/weather_description.csv")
ggplot(df_desc, aes(x=Vancouver)) +
  geom_bar()
```



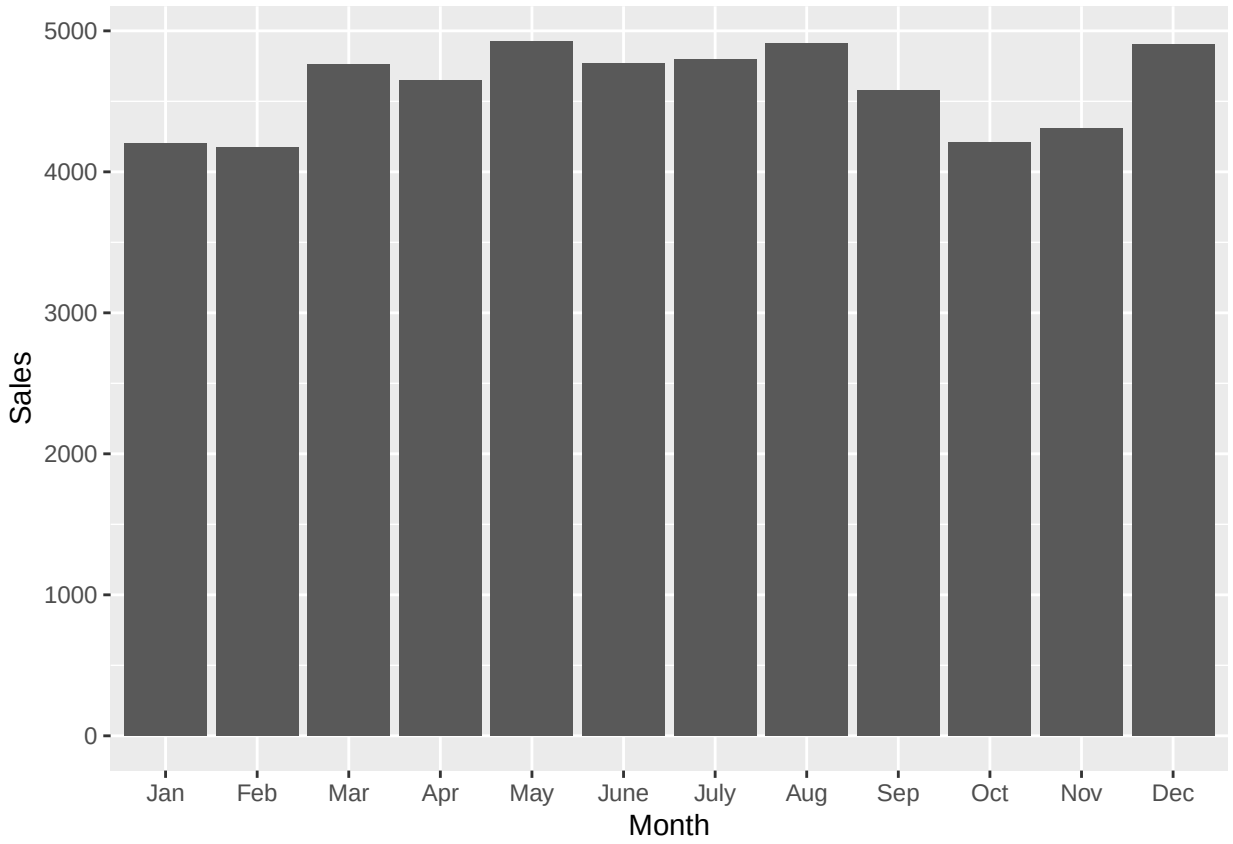
If you want to visualize the sum of the values assciated with with category, you need to also specify the variable where you look at the values.

```
df <- na.omit(RetailSales)
months_of_the_year <- c("Jan", "Feb", "Mar", "Apr", "May", "June",
                        "July", "Aug", "Sep", "Oct", "Nov", "Dec")
ggplot(df) +
  geom_bar(aes(x=factor(Month, months_of_the_year), y=Sales), stat="identity") +
  xlab("Month")
```



An alternative is to use geom_col.

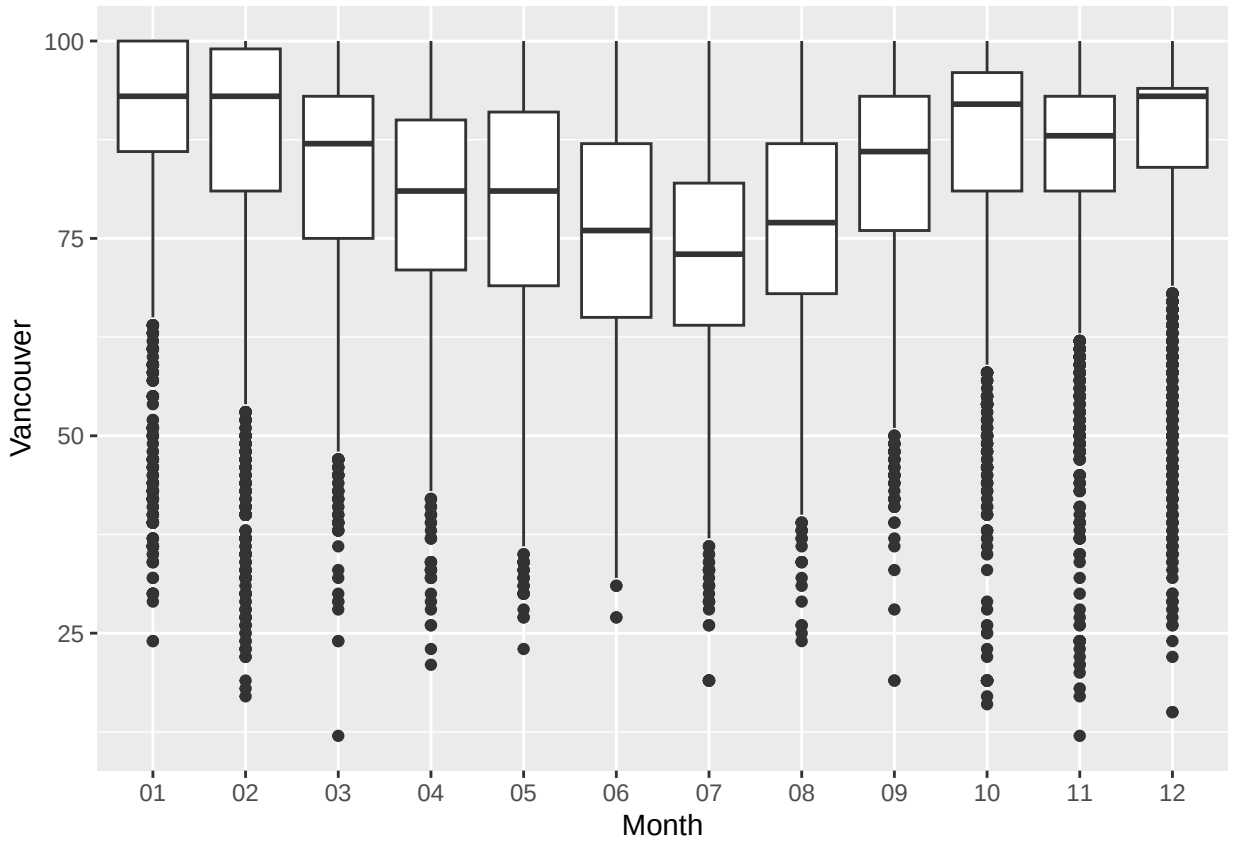
```
ggplot(df) +
  geom_col(aes(x=factor(Month, months_of_the_year), y=Sales)) +
  xlab("Month")
```



Box plots

To make boxplots, we use a similar syntax using a categorical variable for the x axis and a quantitative variable for the y axis. You may have to transform your date variable to transform into a categorical variable.

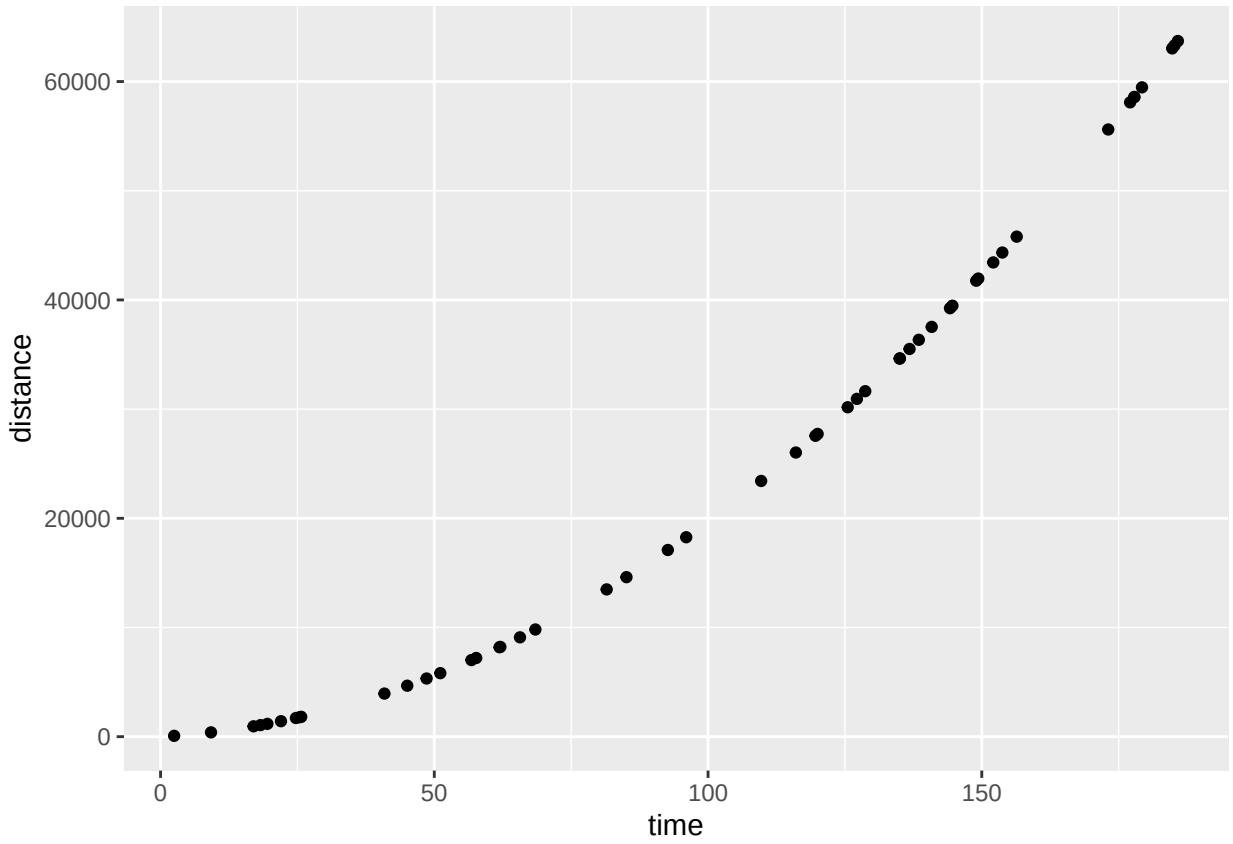
```
df_hum <- read_csv("../../data/historical-hourly-weather-data/humidity.csv")
df_hum$datetime <- as.character(df_hum$datetime)
df_hum$Month <- substr(df_hum$datetime, 6, 7)
ggplot(df_hum, aes(x=Month, y=Vancouver)) +
  geom_boxplot()
```



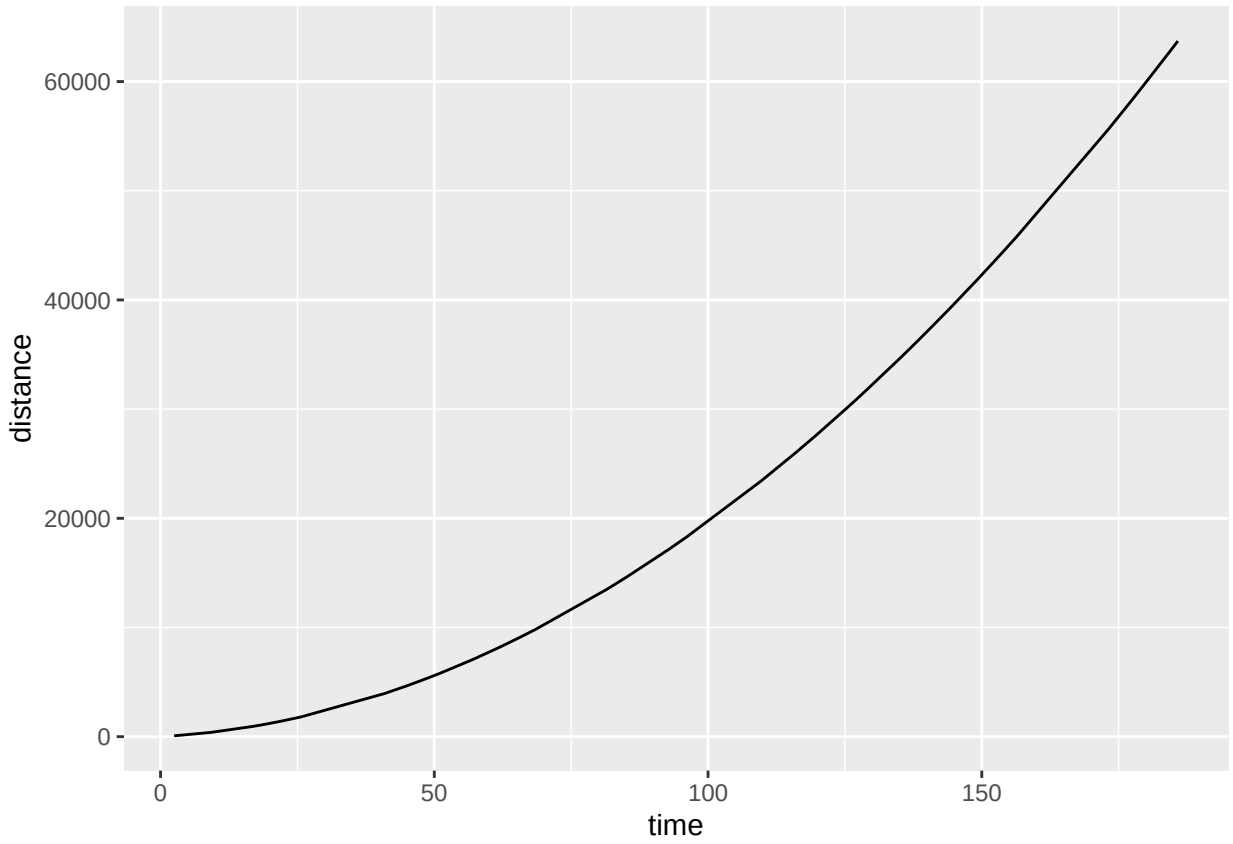
Scatter plots and line plots

To make scatter plots and line plots, we use a similar syntax, using geom_point for scatter plots and geom_line for line plots.

```
a = 3.4
v0 = 27
time <- runif(50, min=0, max=200)
distance <- sapply(time, function(x) v0 * x + 0.5 * a * x^2)
df <- data.frame(time,distance)
ggplot(df, aes(x=time, y=distance)) +
  geom_point()
```



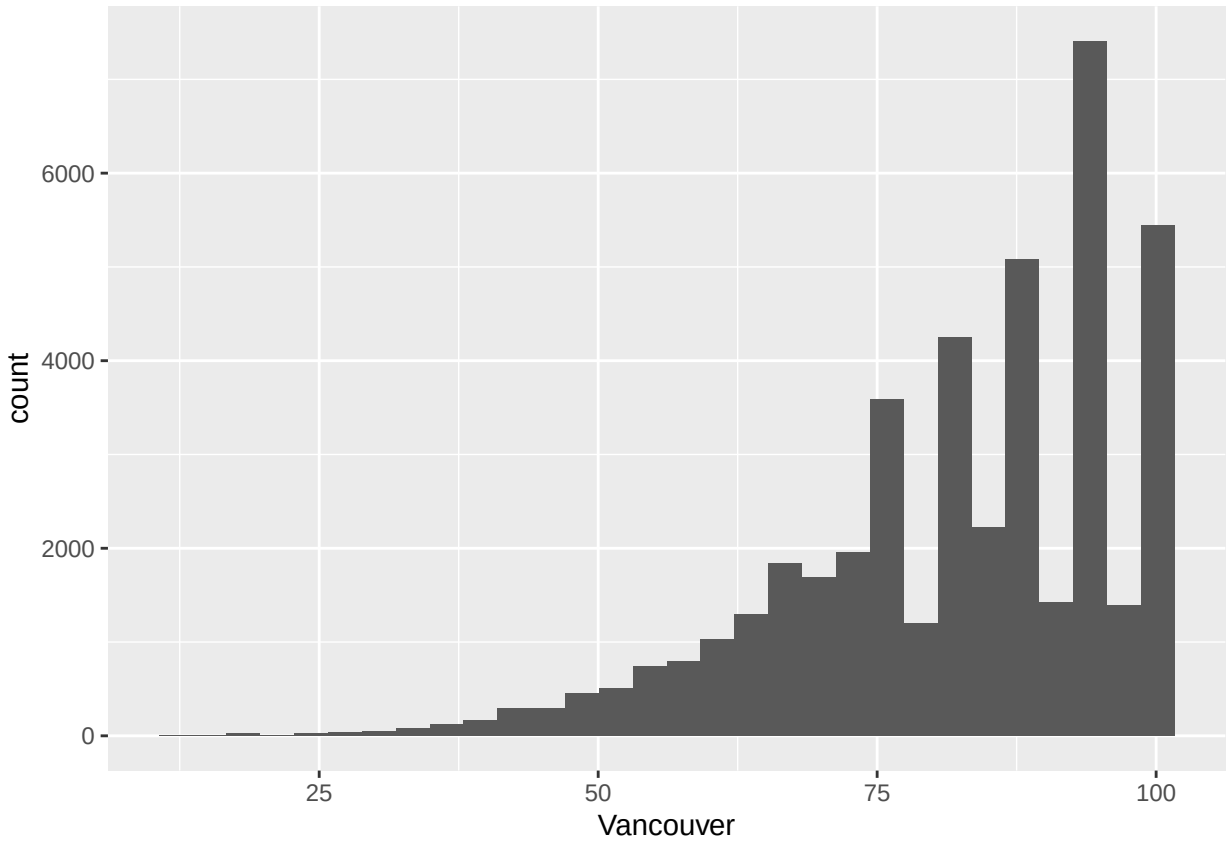

```
ggplot(df, aes(x=time, y=distance)) +
  geom_line()
```



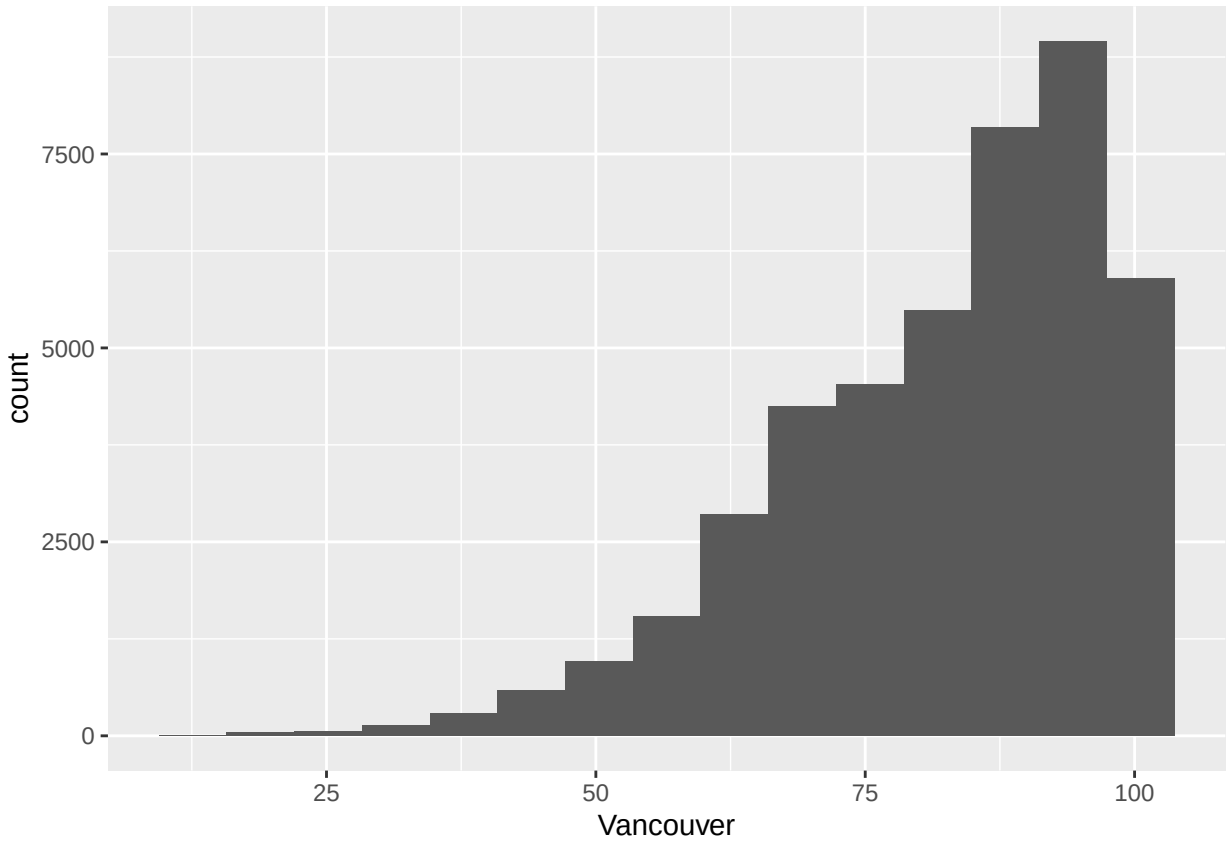
Changing histogram defaults and adding aesthetics

We will now review basic functions to customize the plots and add context, such as title and labels, to an histogram plot.

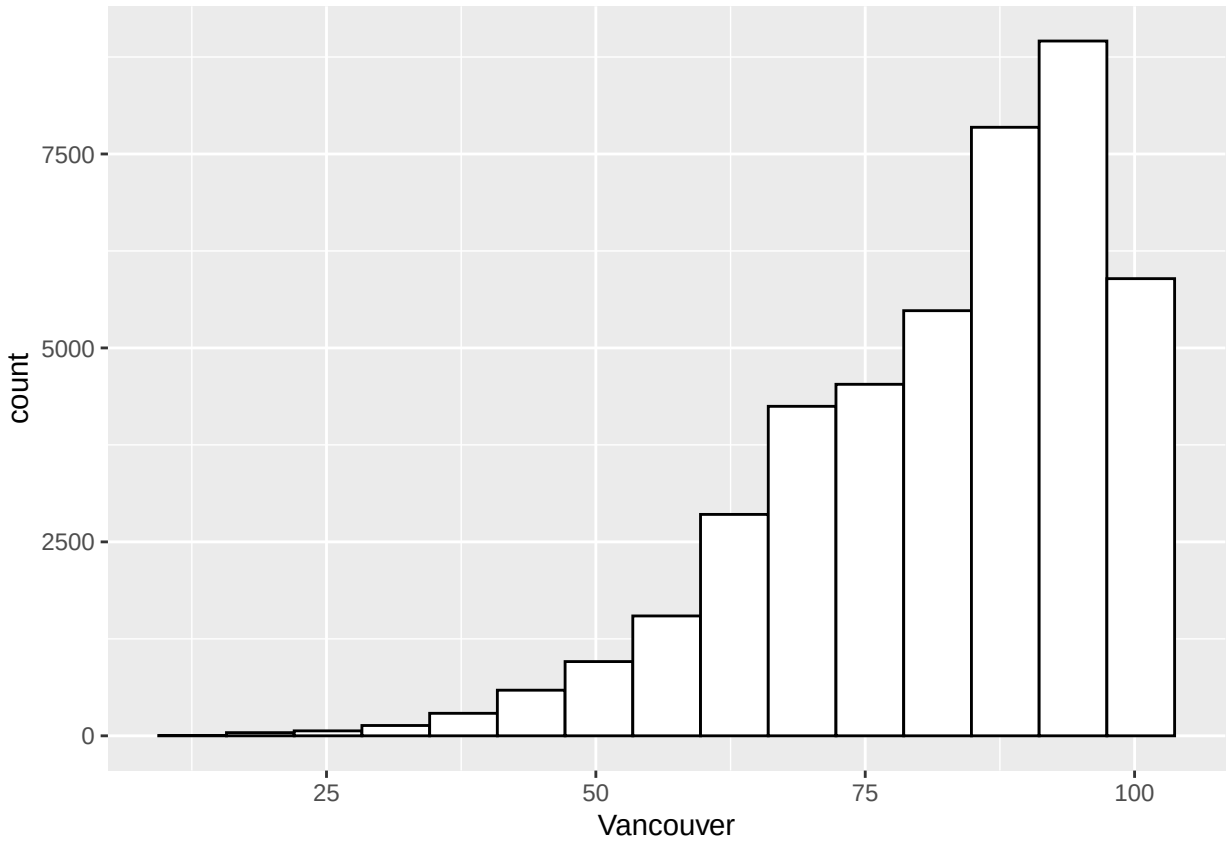
```
df_hum <- read_csv("../../data/historical-hourly-weather-data/humidity.csv")
ggplot(df_hum, aes(x=Vancouver)) +
  geom_histogram()
```



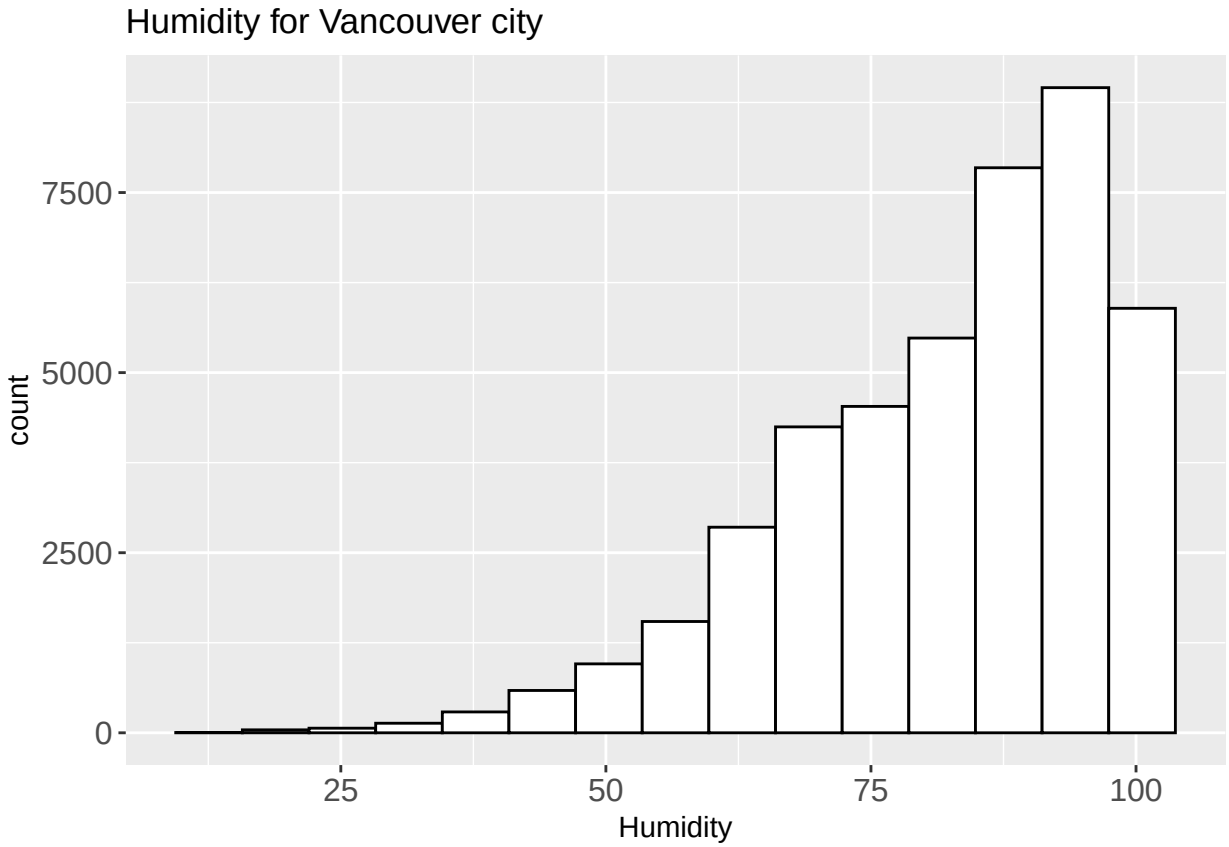

```
ggplot(df_hum, aes(x=Vancouver)) +
  geom_histogram(bins=15)
```




```
ggplot(df_hum, aes(x=Vancouver)) +
  geom_histogram(bins=15, fill="white", color=1)
```



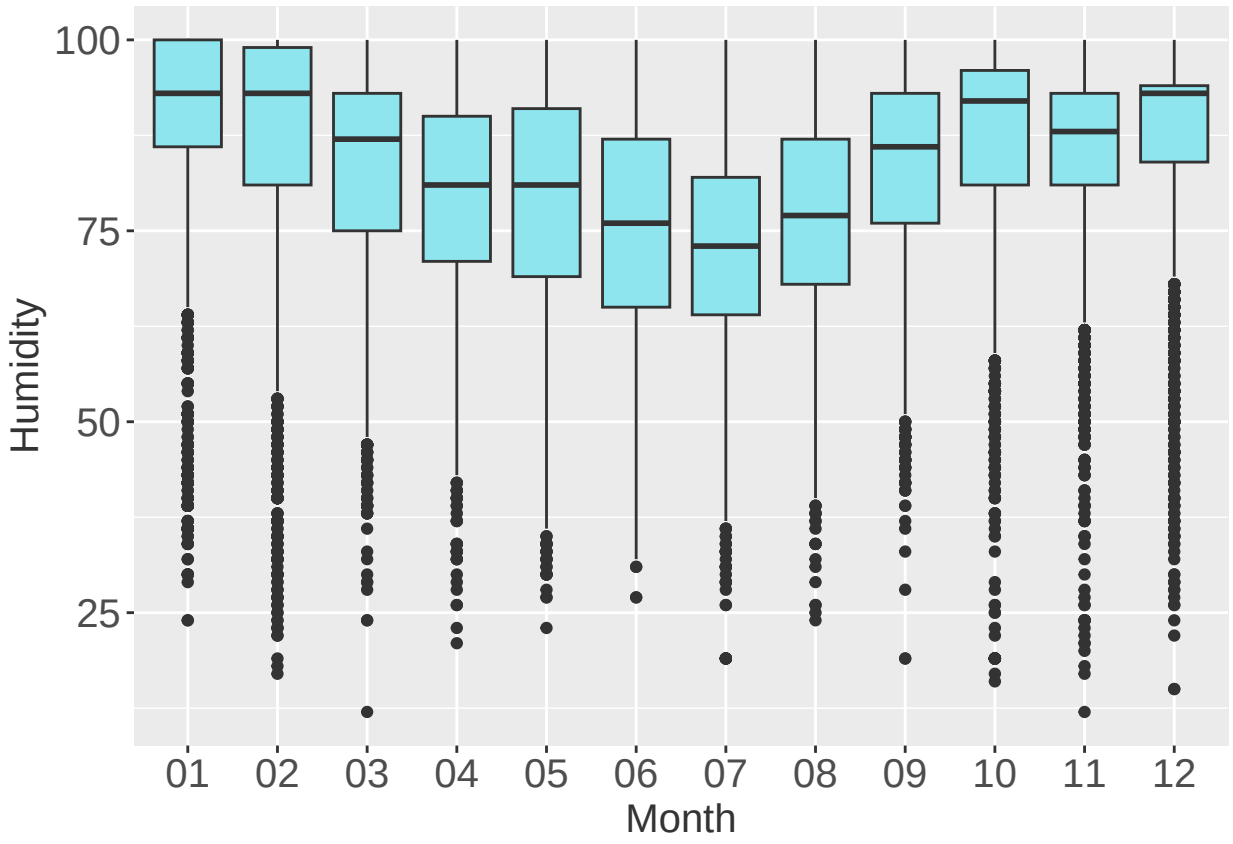

```
ggplot(df_hum, aes(x=Vancouver)) +
  geom_histogram(bins=15, fill="white", color=1) +
  ggtitle("Humidity for Vancouver city") +
  xlab("Humidity") +
  theme(axis.text.x=element_text(size=12),
        axis.text.y=element_text(size=12))
```



Changing boxplot defaults and adding aesthetics

Th same ggplot commands can be used to customize a boxplot.

```
df_hum <- read_csv("../../data/historical-hourly-weather-data/humidity.csv")
df_hum$datetime <- as.character(df_hum$datetime)
df_hum$Month <- substr(df_hum$datetime, 6, 7)
ggplot(df_hum, aes(x=Month, y=Vancouver)) +
  geom_boxplot(color="gray20", fill="cadetblue2") +
  ylab("Humidity") +
  theme(axis.text.x=element_text(size=15),
        axis.text.y=element_text(size=15),
        axis.title.x=element_text(size=15, color="gray20"),
        axis.title.y=element_text(size=15, color="gray20"))
```



Grammar of graphics and visual components

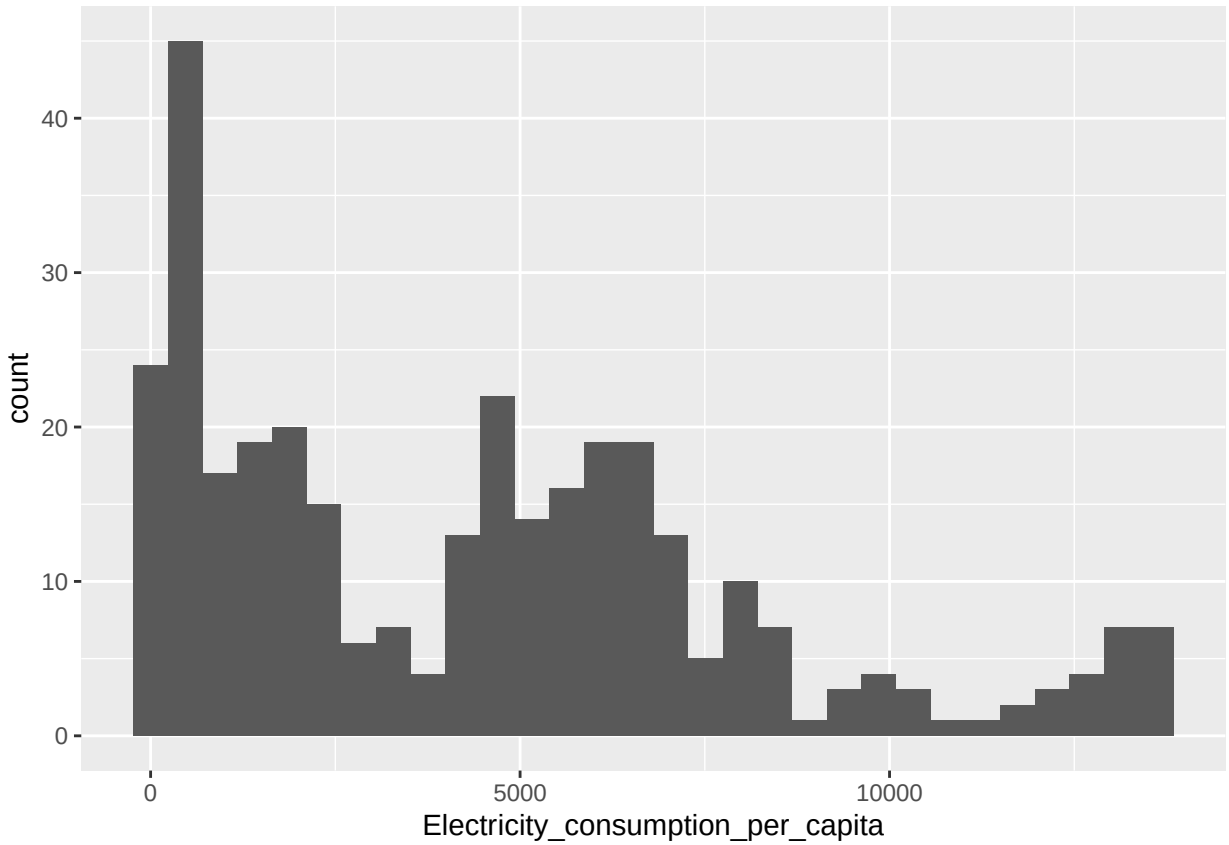
To make more complex plots by adding contextual elements and customizing the properties of the plot, we add an additional function to the existing plot. We will see today some of these additional plotting function and continue exploring them in the next tutorial.

Layers

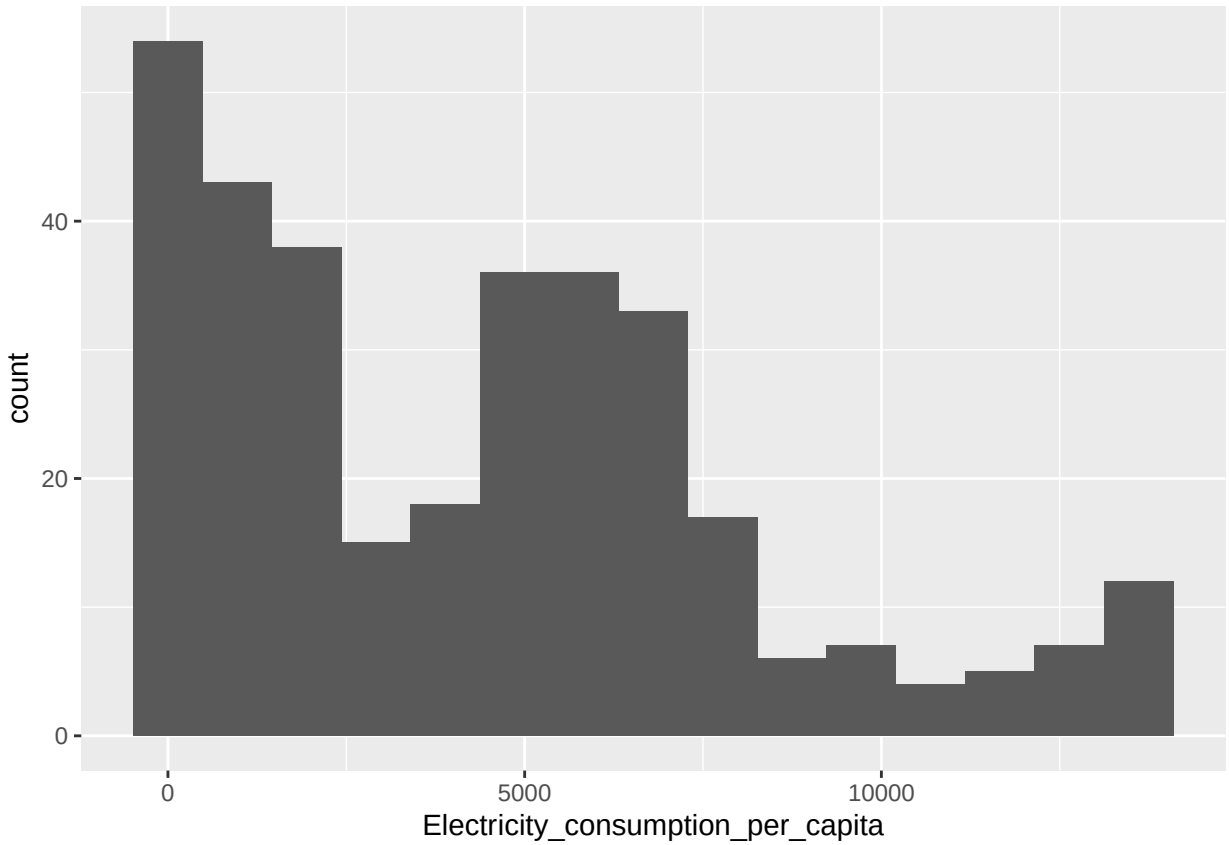
Each ggplot plot is built up as a layer. Layers are made up of geometric objects, together with their statistical transformations and their thematic aspects. Thus each plot can be thought of as a separate variable, to which further transformations can be applied.

We can thus create an initial plot and then customize it in different manners by applying different ggplot functions to the initial output. Let us look at an example.

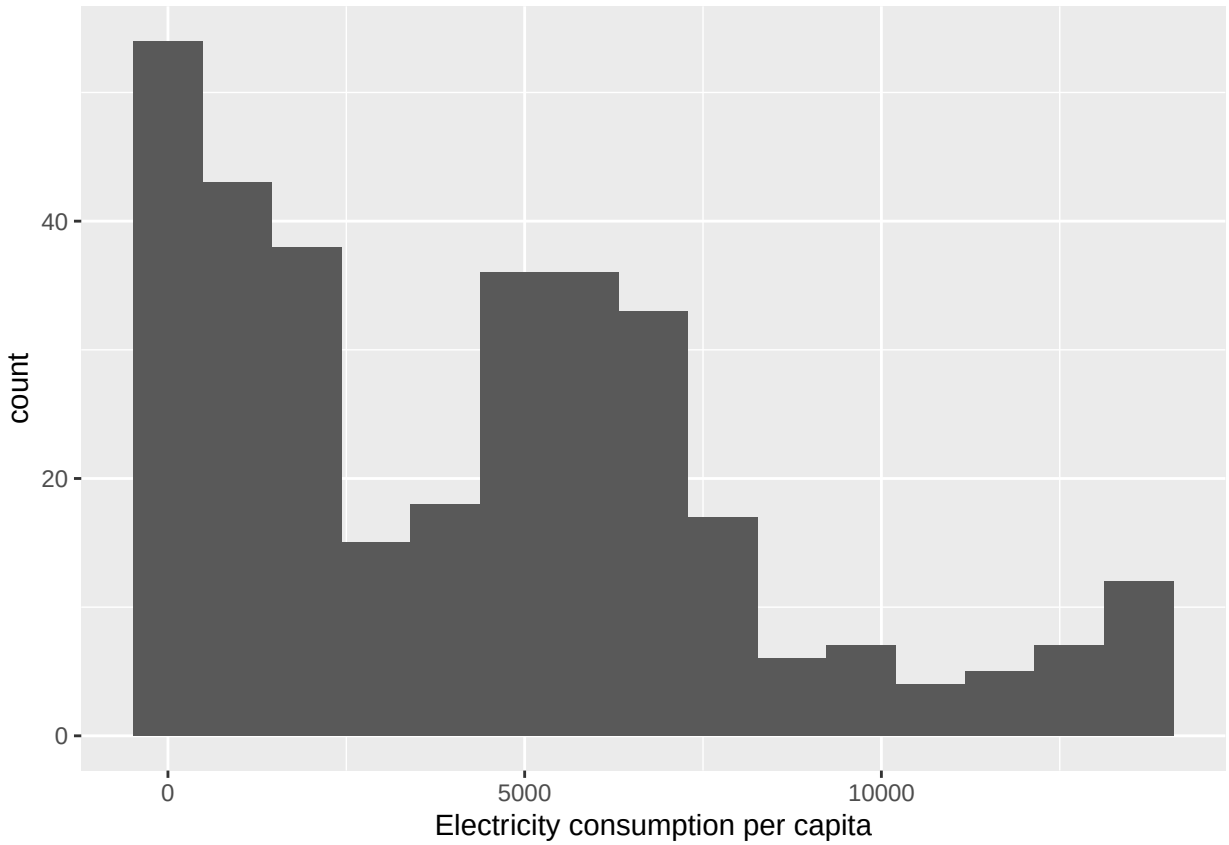
```
df <- read_csv("../../data/gapminder-data.csv")
p1 <- ggplot(df, aes(x=Electricity_consumption_per_capita))
p2 <- p1 +
  geom_histogram()
p2
```




```
p3 <- p1 +
  geom_histogram(bins=15)
p3
```



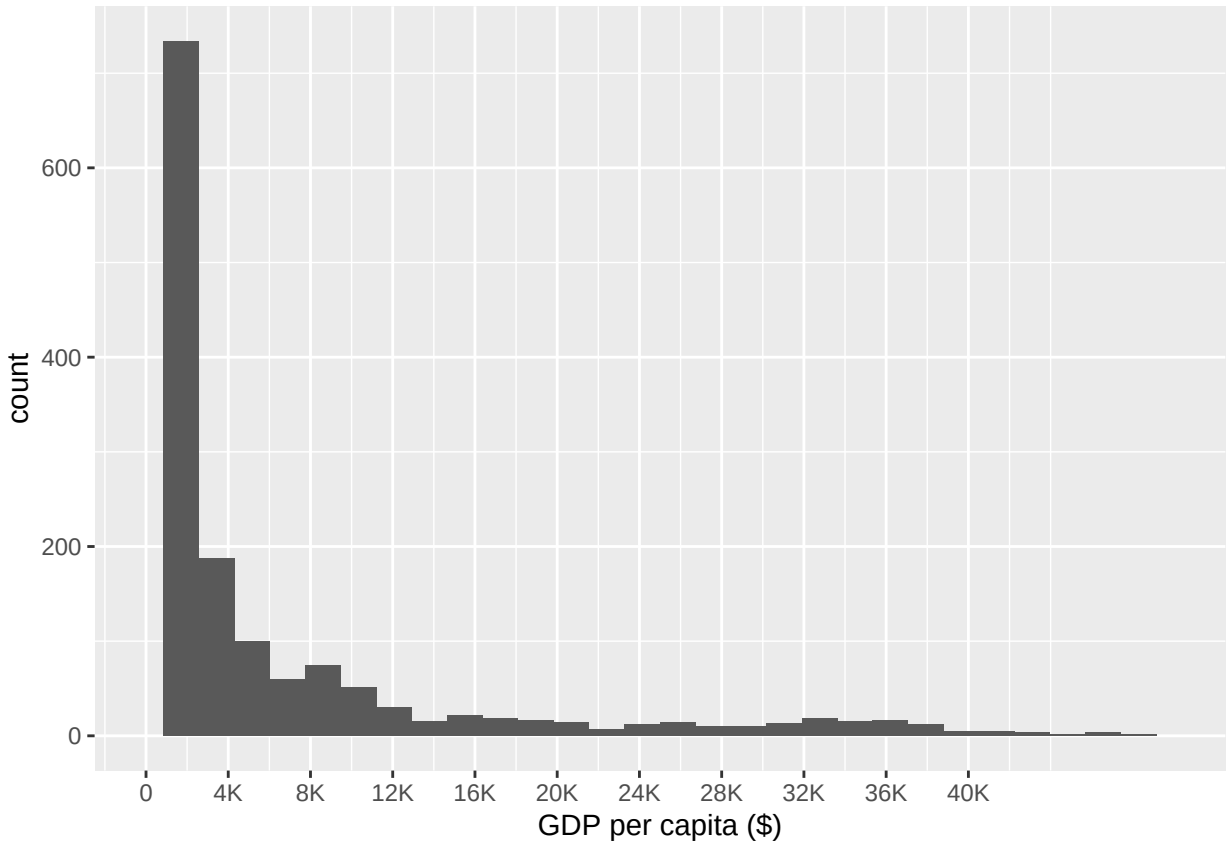

```
p4 <- p3 +
  xlab("Electricity consumption per capita")
p4
```



Scales

Axes scales can be modified using the scale_x_continuous (or scale_y_continuous) and scale_x_discrete (or scale_y_discrete) functions.

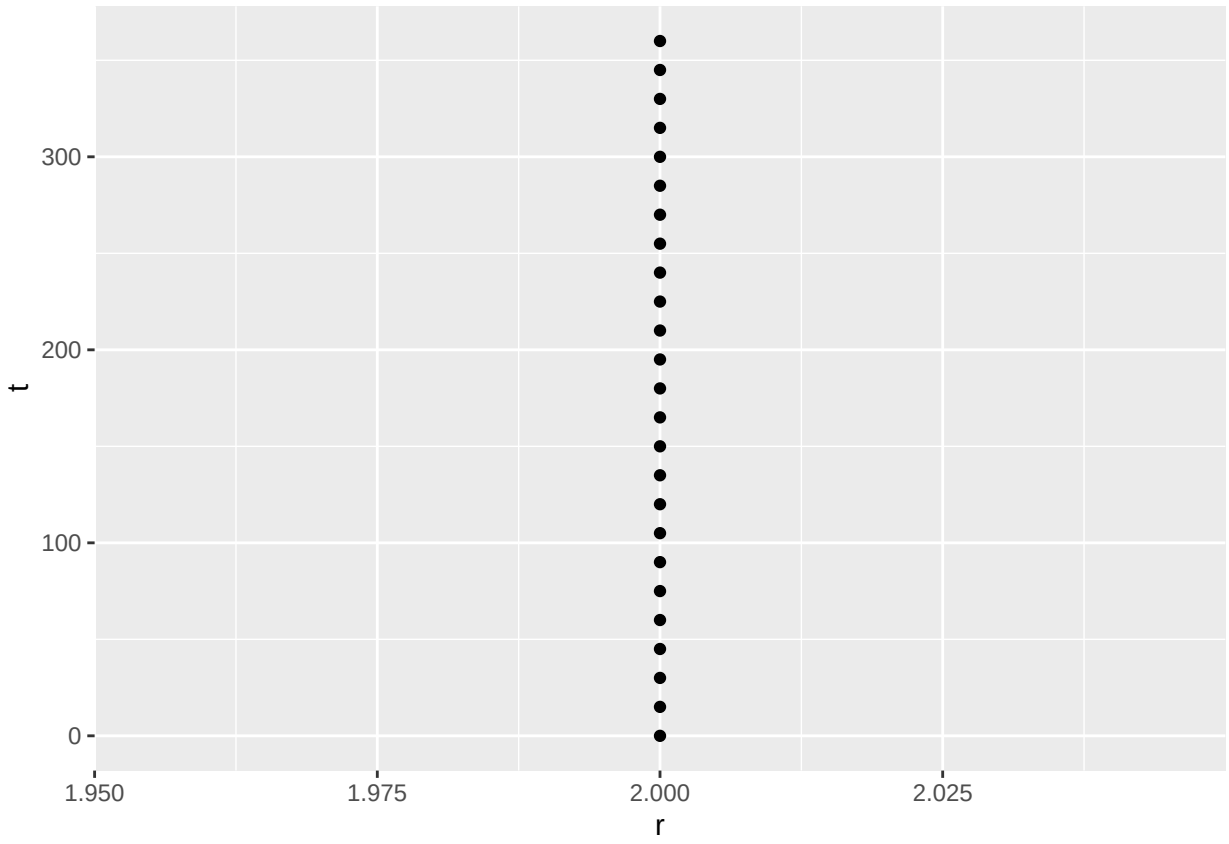
```
df <- read_csv("../../data/gapminder-data.csv")
p1 <- ggplot(df, aes(x=gdp_per_capita))
p2 <- p1 +
  geom_histogram()
p3 <- p2 +
  scale_x_continuous(name='GDP per capita ($)',
                     limits=c(0, 50000),
                     breaks=seq(0, 40000, 4000),
                     labels=c('0', '4K', '8K', '12K', '16K', '20K',
                              '24K', '28K', '32K', '36K', '40K'))
p3
```



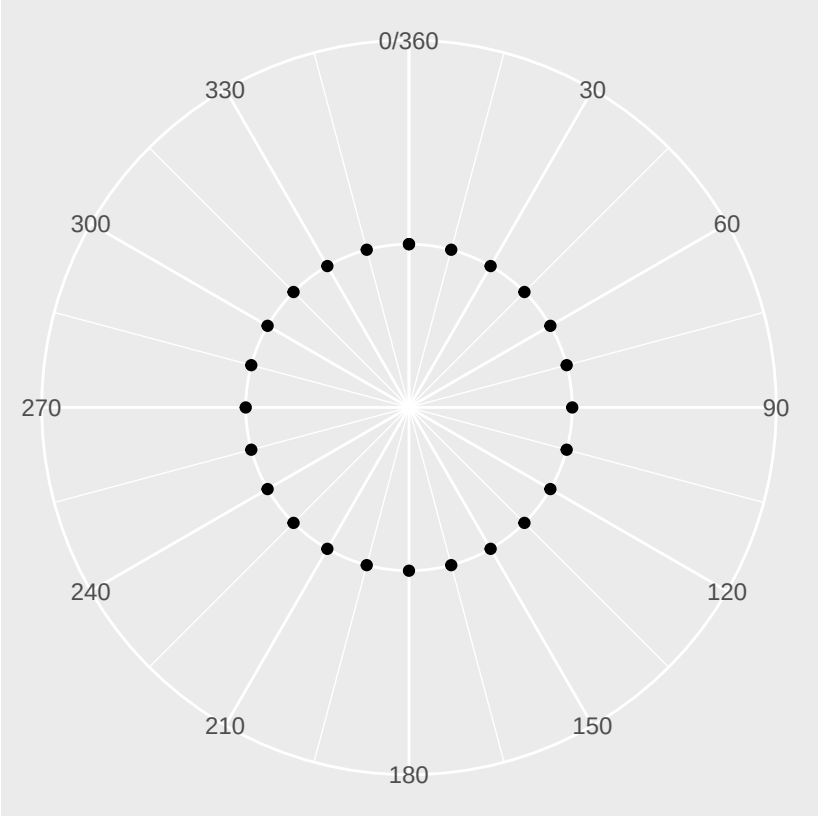
Polar coordinates

The initial plot can be transformed by applying polar coordinates.

```
t <- seq(0, 360, by=15)
r <- 2
p1 <- qplot(r, t)
p1
```



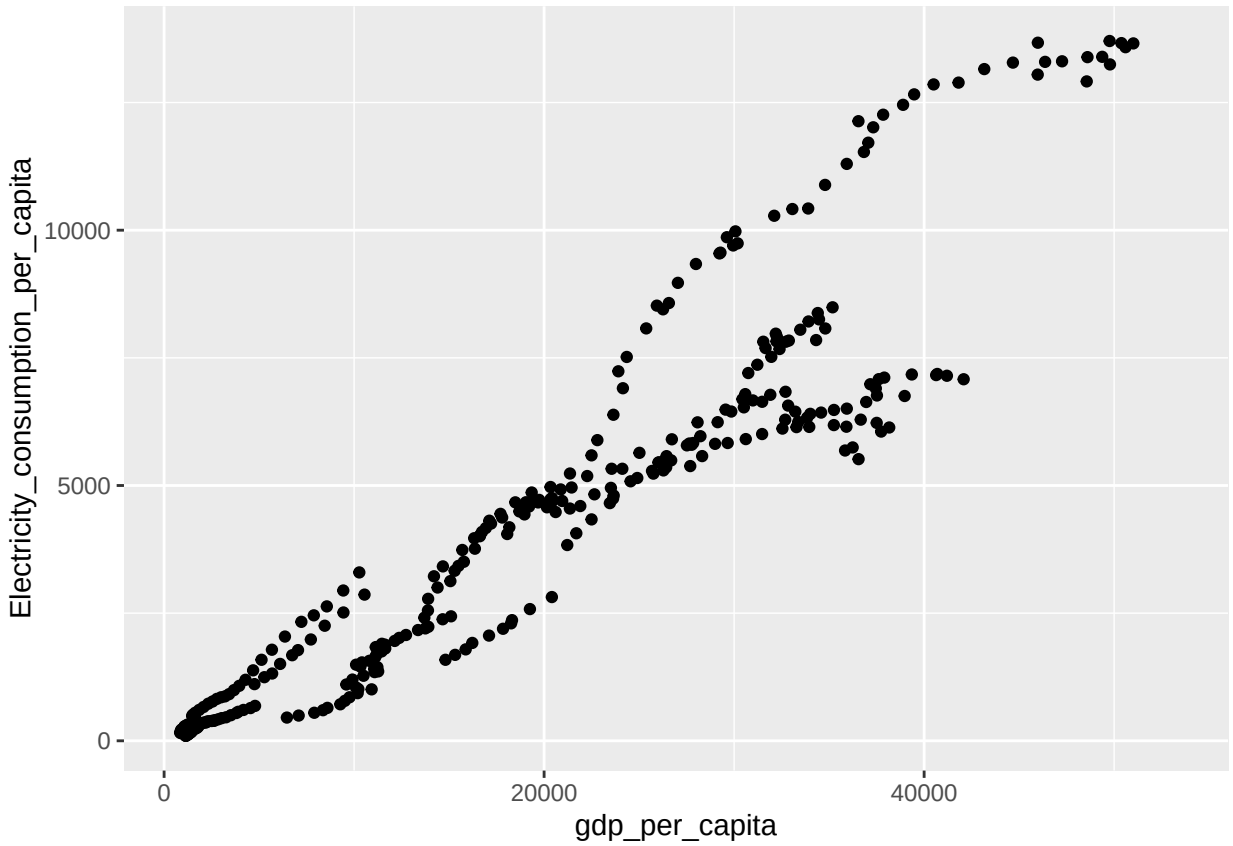

```
p1 +
  coord_polar(theta="y") +
  scale_y_continuous(breaks=seq(0, 360, 30)) +
  theme(axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Facets

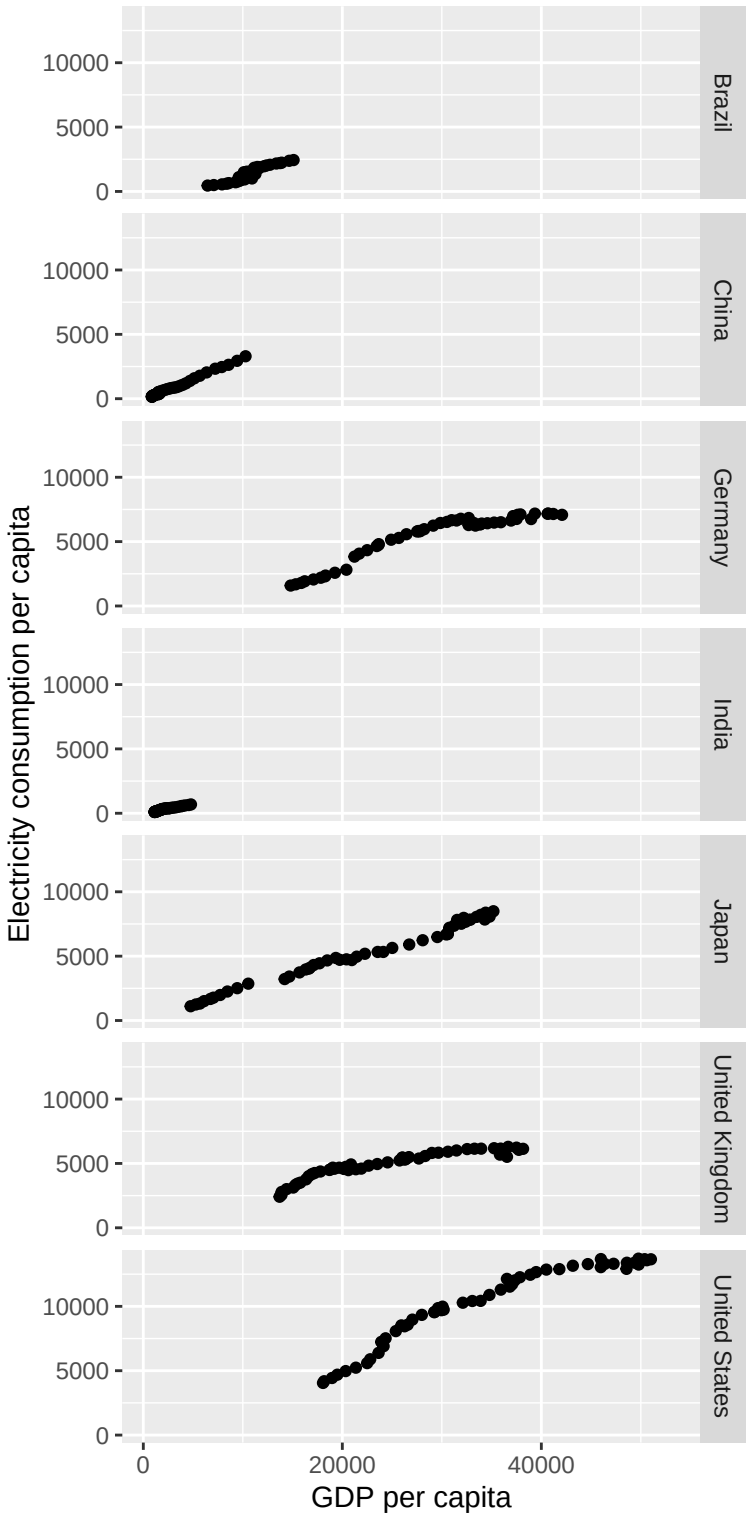
Trellis displays allows us to create subplots for each value of a categorical variable. This is usually useful for data exploration. We first need to create the general type of plot that we want to have.

```
df <- read_csv("../../data/gapminder-data.csv")
p <- ggplot(df, aes(x=gdp_per_capita, y=Electricity_consumption_per_capita)) +
  geom_point()
p
```



If we have only one categorical variable, we may want to have one row per value of the categorical variable, which will give a single column with one plot per row.

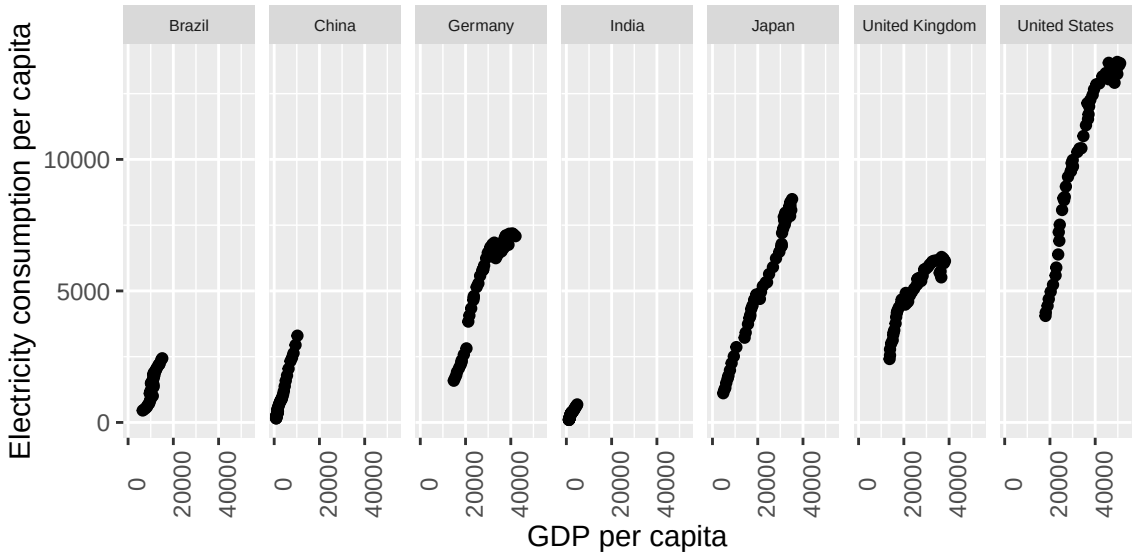
```
p + facet_grid(Country ~ .) +
  xlab("GDP per capita") +
  ylab("Electricity consumption per capita")
```



Alternatively, we may want to have one column per value of the categorical variable, which will give a single row with one plot per column.

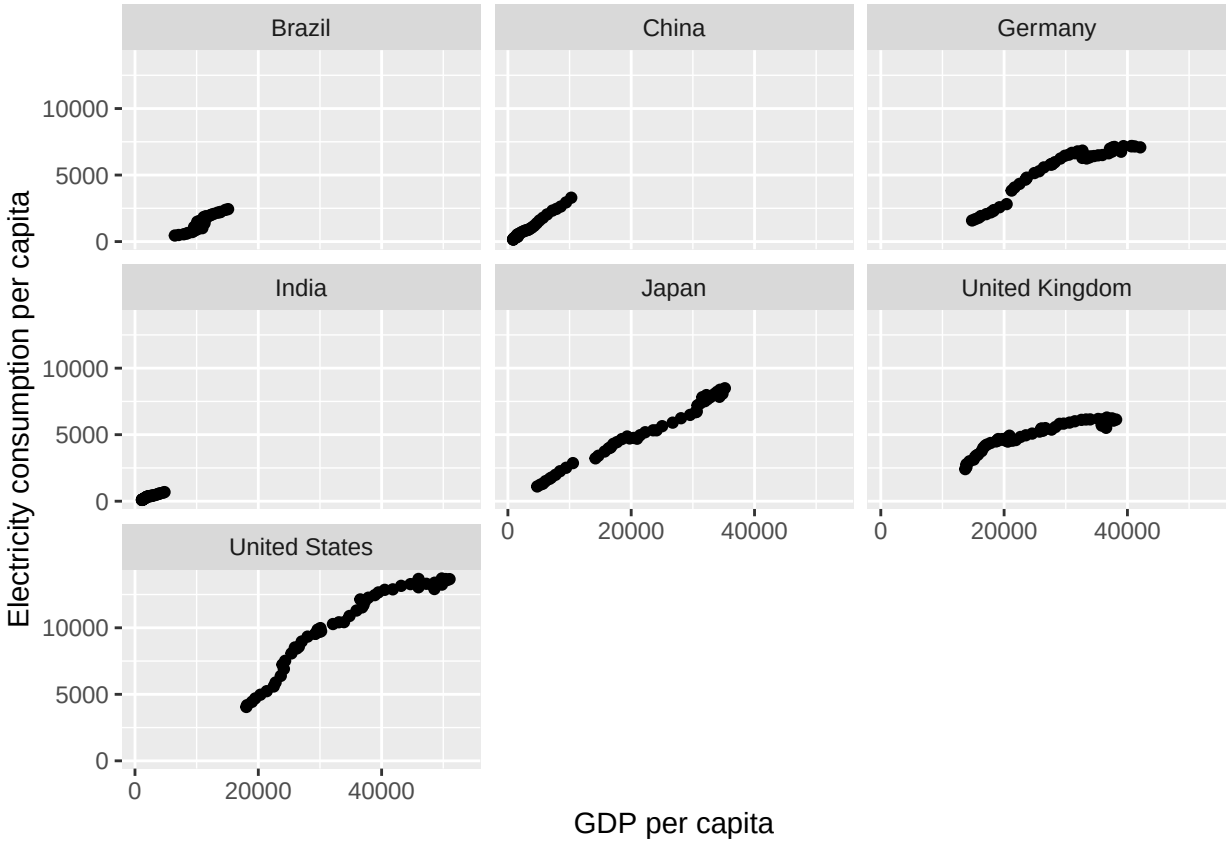
```
p + facet_grid(. ~ Country) +
  xlab("GDP per capita") +
  ylab("Electricity consumption per capita") +
```

```
  theme(axis.text.x=element_text(angle=90),
        strip.text.x=element_text(size=6))
```



Alternatively, if there is a single categorical variable, we can put the subplots into a grid which size will be determined by the number of distinct values taken by this variable.

```
p + facet_wrap(~Country) +
  xlab("GDP per capita") +
  ylab("Electricity consumption per capita")
```

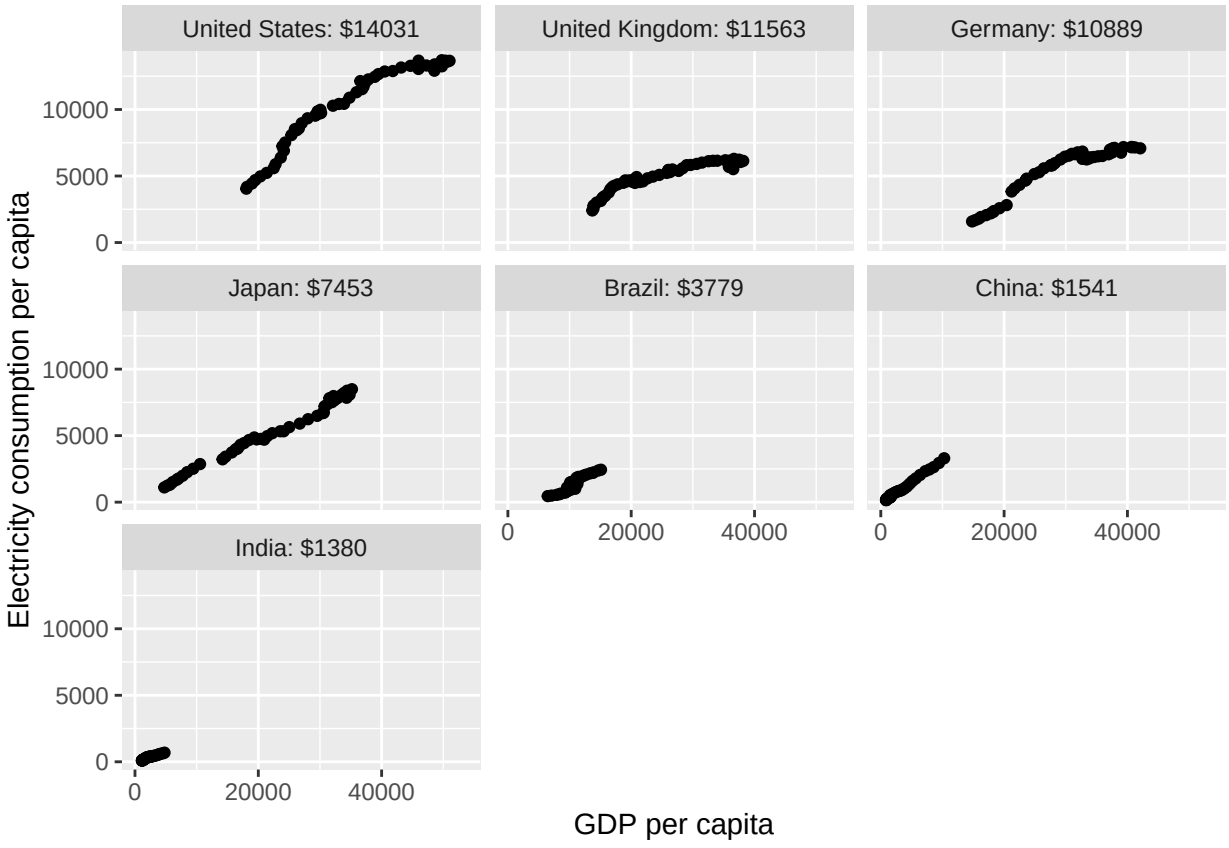


By default, the countries will be ordered in the grid by alphabetical order. This may not the easiest way for the viewer to perceive some underlying phenomena in the data set. This is how you can reorder the countries by decreasing average GDP per capita.

```
ordered_countries <- df %>%
  group_by(Country) %>%
  summarize(mean = round(mean(gdp_per_capita))) %>%
  arrange(desc(mean)) %>%
  mutate(labels = str_c(Country, ": $", mean))
country.labs <- ordered_countries$labels
names(country.labs) <- ordered_countries$Country

df_ordered <- df %>%
  mutate(Country = factor(Country, levels=ordered_countries$Country))

ggplot(df_ordered, aes(x=gdp_per_capita, y=Electricity_consumption_per_capita)) +
  geom_point() +
  facet_wrap(~Country,
             labeller=labeller(Country = country.labs)) +
  xlab("GDP per capita") +
  ylab("Electricity consumption per capita")
```

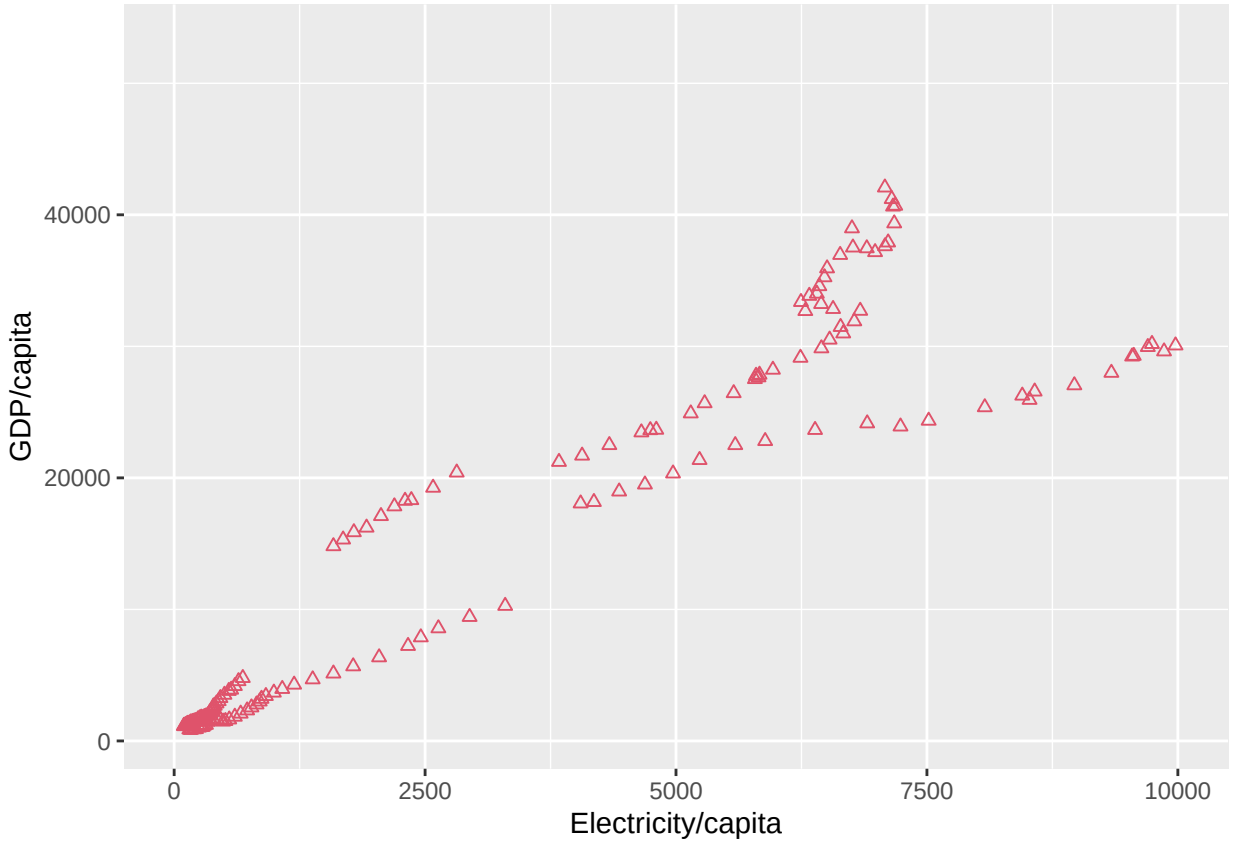


Shapes and colors

You can either change the shape and color for the entire plot.

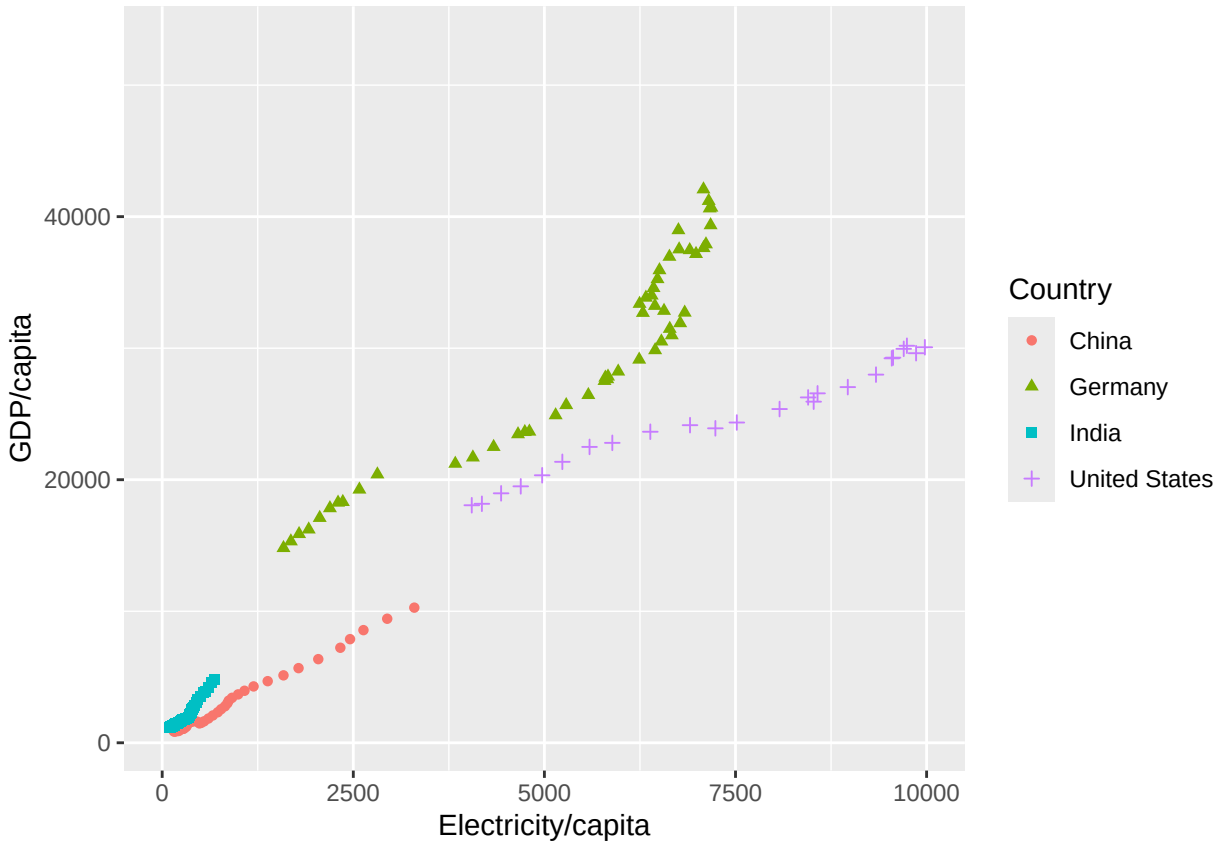
```
dfs <- subset(df, Country %in% c("Germany", "India", "China", "United States"))
ggplot(dfs, aes(x=Electricity_consumption_per_capita, y=gdp_per_capita)) +
```

```
  geom_point(color=2, shape=2) +
  xlim(0, 10000) +
  xlab("Electricity/capita") +
  ylab("GDP/capita")
```



or use shape and color as visual cues for a categorical variable.

```
ggplot(dfs, aes(x=Electricity_consumption_per_capita, y=gdp_per_capita)) +
  geom_point(aes(color=Country, shape=Country)) +
  xlim(0, 10000) +
  xlab("Electricity/capita") +
  ylab("GDP/capita")
```


GDP/capita
40000
20000
0
0
2500
5000
7500
10000
Electricity/capita
Country
China
Germany
India
United States