# Exploring data #1

# Basic plotting

# Example data—Beijing air quality

Let's read the Beijing data in and clean up the "-999" values:

```
library("readr")
library("dplyr")
```

```
## Warning: package 'dplyr' was built under R version
## 3.5.2
```

```
beijing_pm_raw <- read_csv("data/Beijing_2017_HourlyPM25.csv",
                           skip = 3, na = "-999")
```

# Example data—Beijing air quality

Clean up as before:

```
library(lubridate)
beijing_pm <- beijing_pm_raw %>%
  rename(sample_time = `Date (LST)`,
         value = Value,
         qc = `QC Name`) %>%
  select(sample_time, value, qc) %>%
  mutate(aqi = cut(value,
                   breaks = c(0, 50, 100, 150, 200,
                              300, 500, Inf),
                   labels = c("Good", "Moderate",
                              "Unhealthy for Sensitive Groups",
                              "Unhealthy", "Very Unhealthy",
                              "Hazardous", "Beyond Index"))) %>%
  mutate(sample_time = mdy_hm(sample_time)) %>%
  mutate(heating = sample_time < mdy("03/15/2017"))
```

# Plots

# Plots to explore data

Plots can be invaluable in exploring your data.

Today, we will focus on **useful**, rather than **attractive** graphs, since we are focusing on exploring rather than presenting data.

Next lecture, we will talk more about customization, to help you make more attractive plots that would go into final reports.

# `ggplot` conventions

Here, we'll be using functions from the `ggplot2` library, so you'll need to install that package:

```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R
## version 3.5.2
```

The basic steps behind creating a plot with `ggplot2` are:

1. Create an object of the `ggplot` class, typically specifying the **data** to be shown in the plot;
2. Add on (using +) one or more **geoms**, specifying the **aesthetics** for each; and
3. Add on (using +) other elements to create and customize the plot (e.g., add layers to customize scales or themes or to add facets).

*Note*: To avoid errors, end lines with +, don't start lines with it.

# Plot data

The ggplot function requires you to input a dataframe with the data you will plot. All the columns in that dataframe can be mapped to specific aesthetics within the plot.

```
beijing_pm %>%
  slice(1:3)
```

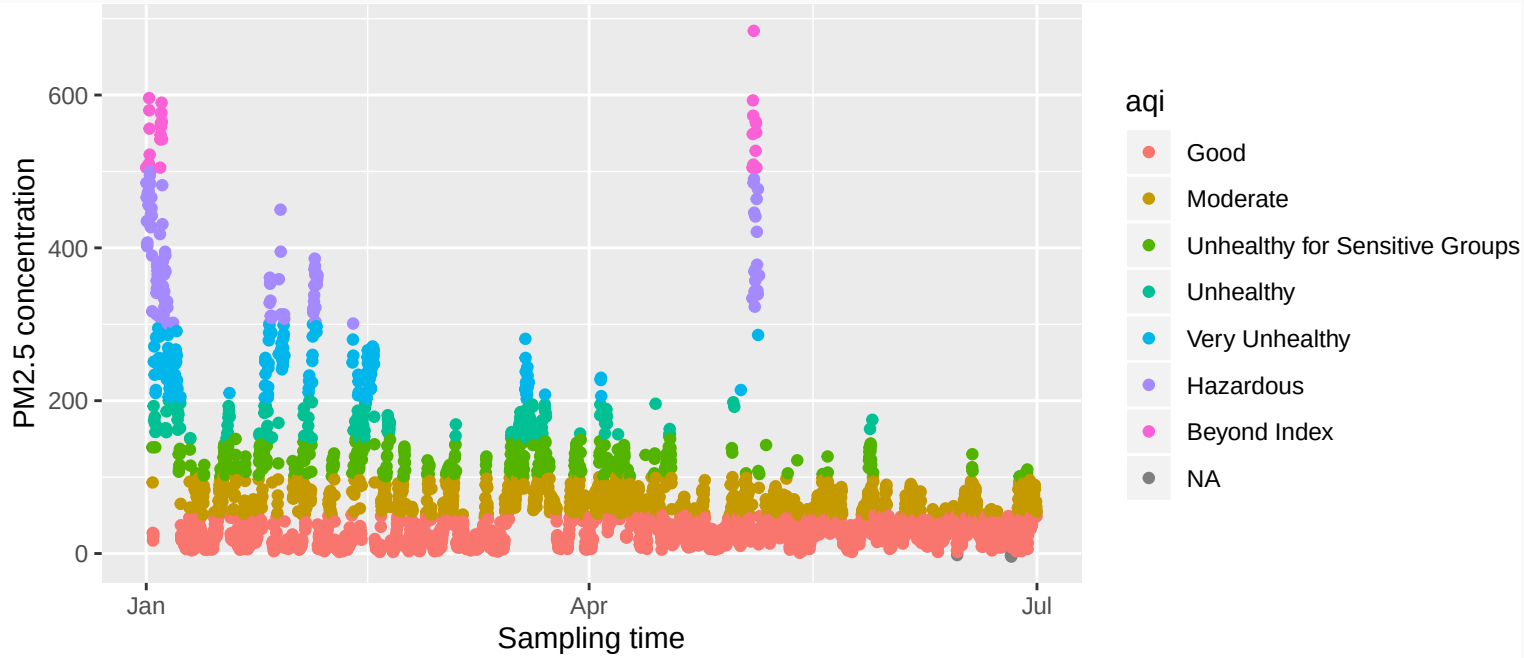
```
## # A tibble: 3 x 5
##   sample_time         value qc    aqi       heating
##   <dttm>              <dbl> <chr> <fct>     <lgl>
## 1 2017-01-01 00:00:00   505 Valid Beyond ~ TRUE
## 2 2017-01-01 01:00:00   485 Valid Hazardo~ TRUE
## 3 2017-01-01 02:00:00   466 Valid Hazardo~ TRUE
```

For example, if we input the beijing_pm dataframe, we would be able to create a plot that shows each sample's sampling time on the x-axis, $PM_{2.5}$ concentration on the y-axis, and AQI by the color of the point.

# Plot aesthetics

**Aesthetics** are plotting elements that can show certain elements of the data.

For example, you may want to create a scatterplot where color shows AQI, x-position shows sampling time, and y-position shows $PM_{2.5}$ concentration.



# Plot aesthetics

In the previous graph, the mapped aesthetics are color, x, and y. In the `ggplot` code, all of these aesthetic mappings will be specified within an `aes` call, which will be nested in another call in the `ggplot` pipeline.

| Aesthetic | `ggplot` abbreviation | `beijing_pm` column |
|---|---|---|
| x-axis position | `x =` | `sample_time` |
| y-axis position | `y =` | `value` |
| color | `color =` | `aqi` |

This is how these mappings will be specified in an `aes` call:

```
# Note: This code should not be run by itself.
# It will eventually be nested in a ggplot call.
aes(x = sample_time, y = value, color = aqi)
```

# Plot aesthetics

Here are some common plot aesthetics you might want to specify:

| Code | Description |
|---|---|
| `x` | Position on x-axis |
| `y` | Position on y-axis |
| `shape` | Shape |
| `color` | Color of border of elements |
| `fill` | Color of inside of elements |
| `size` | Size |
| `alpha` | Transparency (1: opaque; 0: transparent) |
| `linetype` | Type of line (e.g., solid, dashed) |

# Geoms

You will add **geoms** that create the actual geometric objects on the plot. For example, a scatterplot has "points" geoms, since each observation is displayed as a point.

There are `geom_*` functions that can be used to add a variety of geoms. The function to add a "points" geom is `geom_point`.

We just covered three plotting elements:

- Data
- Aesthetics
- Geoms

These are three elements that you will almost always specify when using `ggplot`, and they are sufficient to create a number of basic plots.

# Creating a ggplot object

You can create a scatterplot using ggplot using the following code format:

```
## Generic code
ggplot(data = dataframe) +
  geom_point(mapping = aes(x = column_1, y = column_2,
                           color = column_3))
```

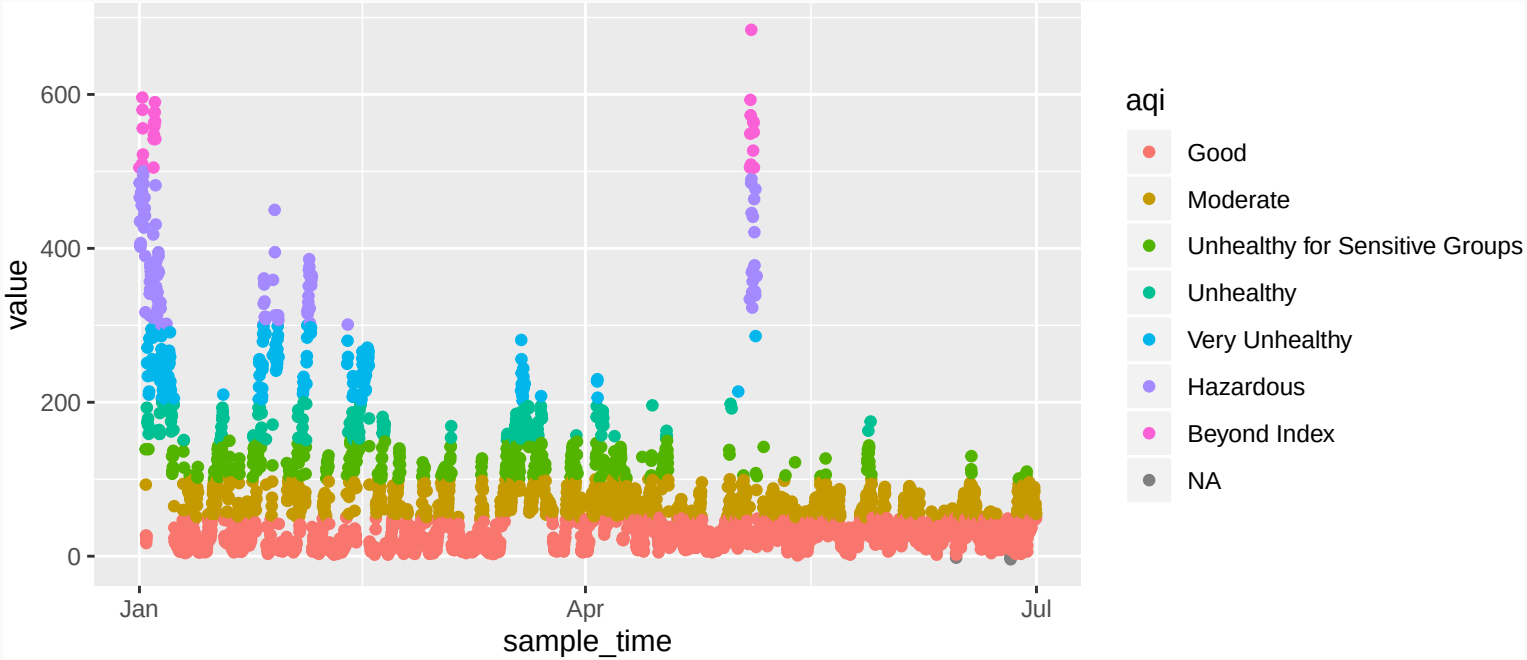
Notice that:

1. The ggplot call specifies the **dataframe** with the data you want to plot
2. A **geom** is added using the appropriate geom_* function for a scatterplot (geom_point).
3. The mappings between columns in the dataframe and **aesthetics** of the geom is specified within an aes call in the mapping argument of the geom_* function call.
4. The aes call includes mappings to two aesthetics that are required from the geom_point geom (x and y) and one that is optional (color).

# Creating a ggplot object

Let's put these ideas together to write the code to create a plot for our example data:

```
ggplot(data = beijing_pm) +
geom_point(mapping = aes(x = sample_time, y = value,
                         color = aqi))
```



# Adding geoms

There are a number of different `geom_*` functions you can use to add geoms to a plot. They are divided between geoms that directly map the data to an aesthetic and those that show some summary or statistic of the data.

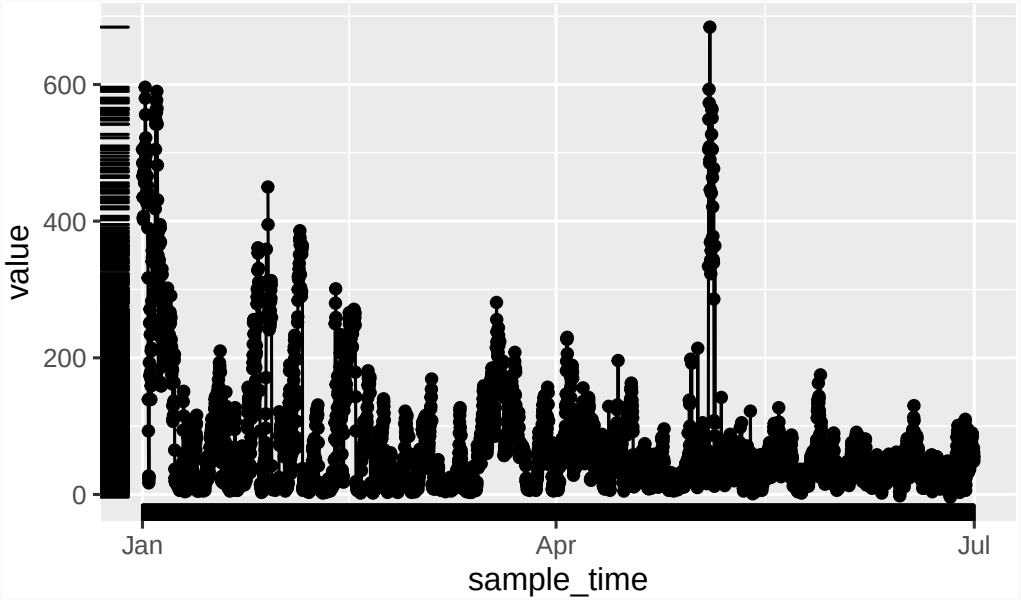
Some of the most common direct-mapping geoms are:

| Geom(s) | Description |
|---|---|
| `geom_point` | Points in 2-D (e.g. scatterplot) |
| `geom_line`, `geom_path` | Connect observations with a line |
| `geom_abline` | A line with a certain intercept and slope |
| `geom_hline`, `geom_vline` | A horizontal or vertical line |
| `geom_rug` | A rug plot |
| `geom_label`, `geom_text` | Text labels |

# Creating a ggplot object

You can add several geoms to the same plot as layers:

```
ggplot(data = beijing_pm) +
geom_point(mapping = aes(x = sample_time, y = value)) +
geom_line(mapping = aes(x = sample_time, y = value)) +
geom_rug(mapping = aes(x = sample_time, y = value))
```



# Creating a ggplot object

You may have noticed that all of these geoms use the same aesthetic mappings (height to x-axis position, weight to y-axis position, and sex to color). To save time, you can specify the aesthetic mappings in the first `ggplot` call. These mappings will then be the default for any of the added geoms.

```
ggplot(data = beijing_pm,
       mapping = aes(x = sample_time, y = value)) +
  geom_point() +
  geom_line() +
  geom_rug()
```

# Creating a ggplot object

Because the first argument of the `ggplot` call is a dataframe, you can also "pipe into" a `ggplot` call:

```
beijing_pm %>%
  ggplot(aes(x = sample_time, y = value)) +
  geom_point() +
  geom_line() +
  geom_rug()
```

# Plot aesthetics

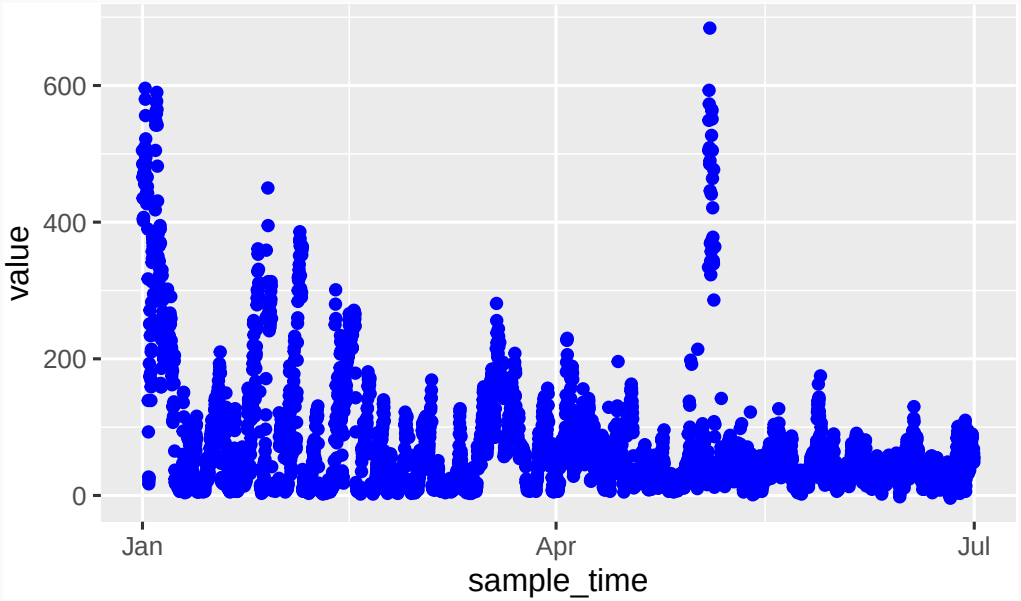
Which aesthetics you must specify in the `aes` call depend on which geom you are adding to the plot.

You can find out the aesthetics you can use for a geom in the "Aesthetics" section of the geom's help file (e.g., `?geom_point`).

Required aesthetics are in bold in this section of the help file and optional ones are not.

# Constant aesthetics

Instead of mapping an aesthetic to an element of your data, you can use a constant value for the aesthetic. For example, you may want to make all the points blue, rather than having color map to AQI:

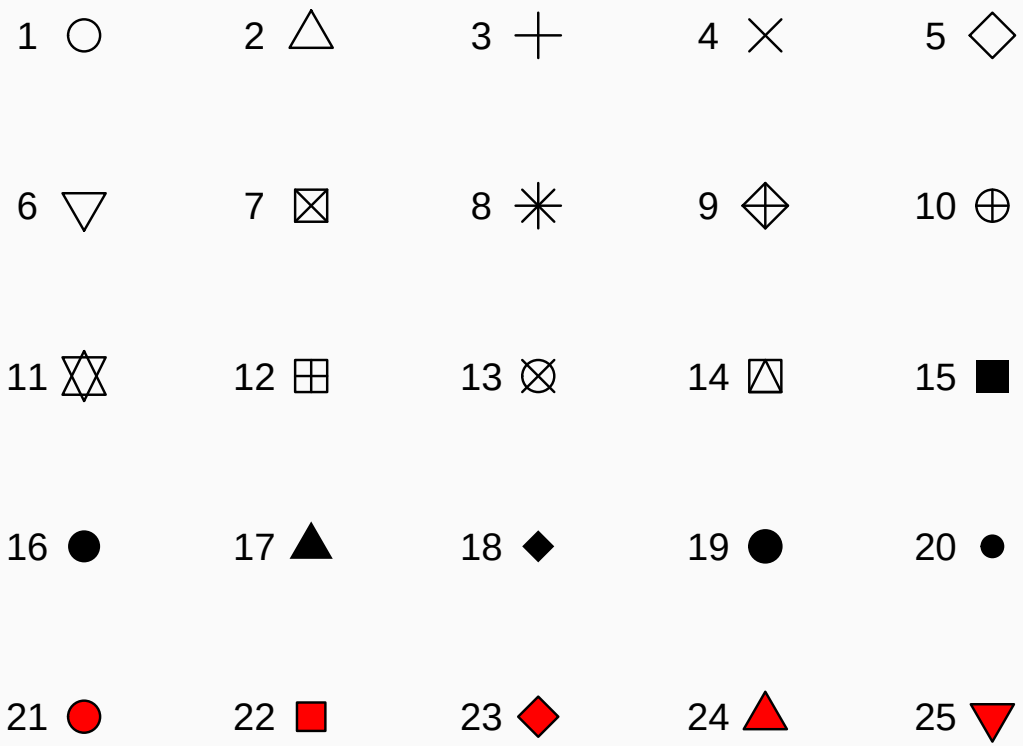



In this case, you can define that aesthetic as a constant for the geom, **outside** of an `aes` statement.

# Constant aesthetics

For example, you may want to change the shape of the points in a scatterplot from their default shape, but not map them to a particular element of the data.

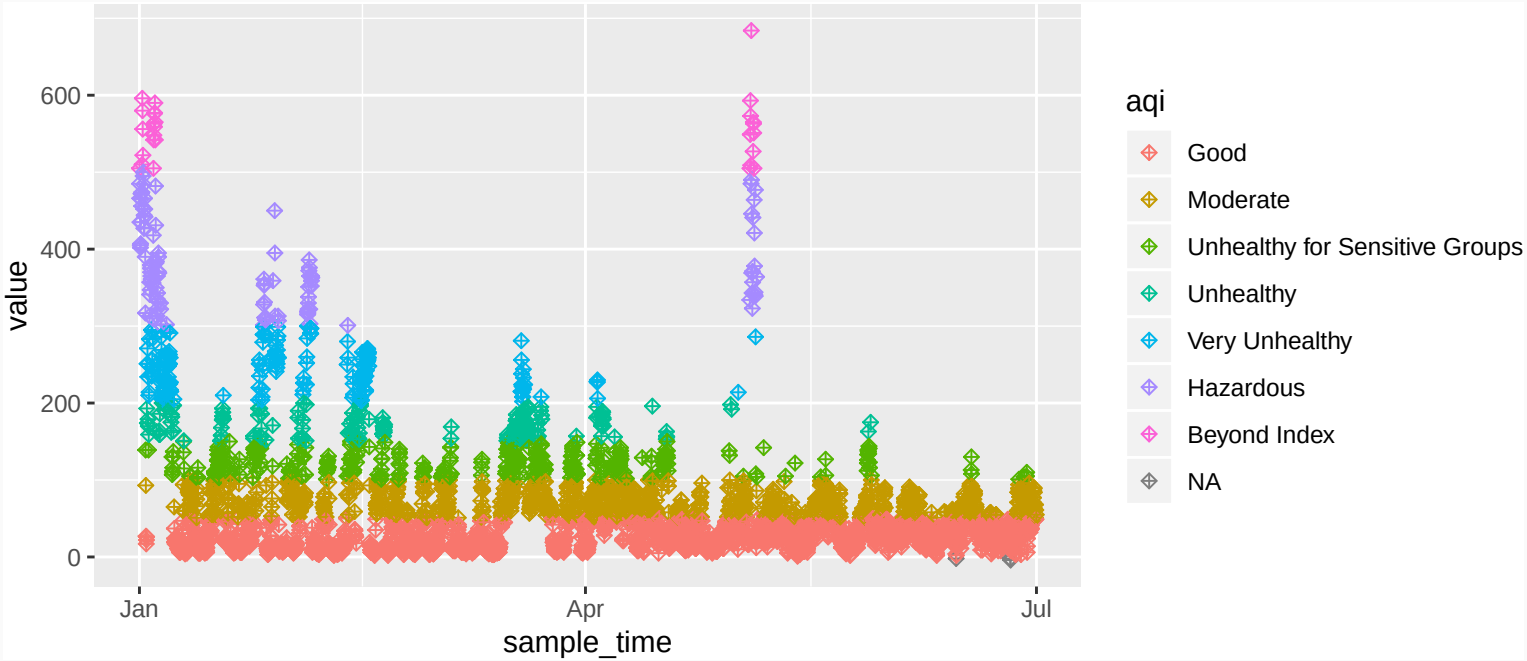
In R, you can specify point shape with a number. Here are the shapes that correspond to the numbers 1 to 25:



# Constant aesthetics

Here is an example of mapping point shape to a constant value other than the default:

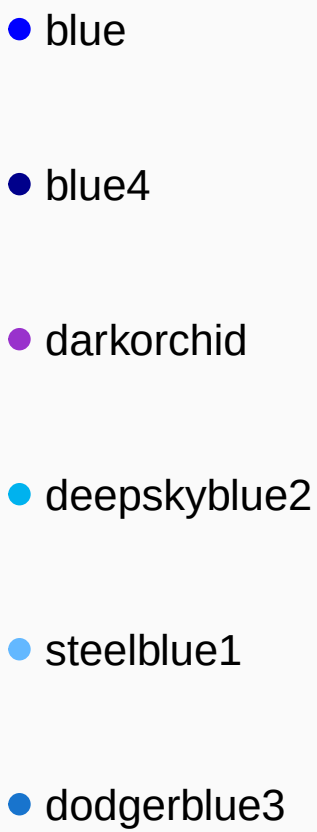
```
ggplot(data = beijing_pm) +
  geom_point(mapping = aes(x = sample_time, y = value,
                           color = aqi),
             shape = 9)
```



# Constant aesthetics

R has character names for different colors. For example:

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

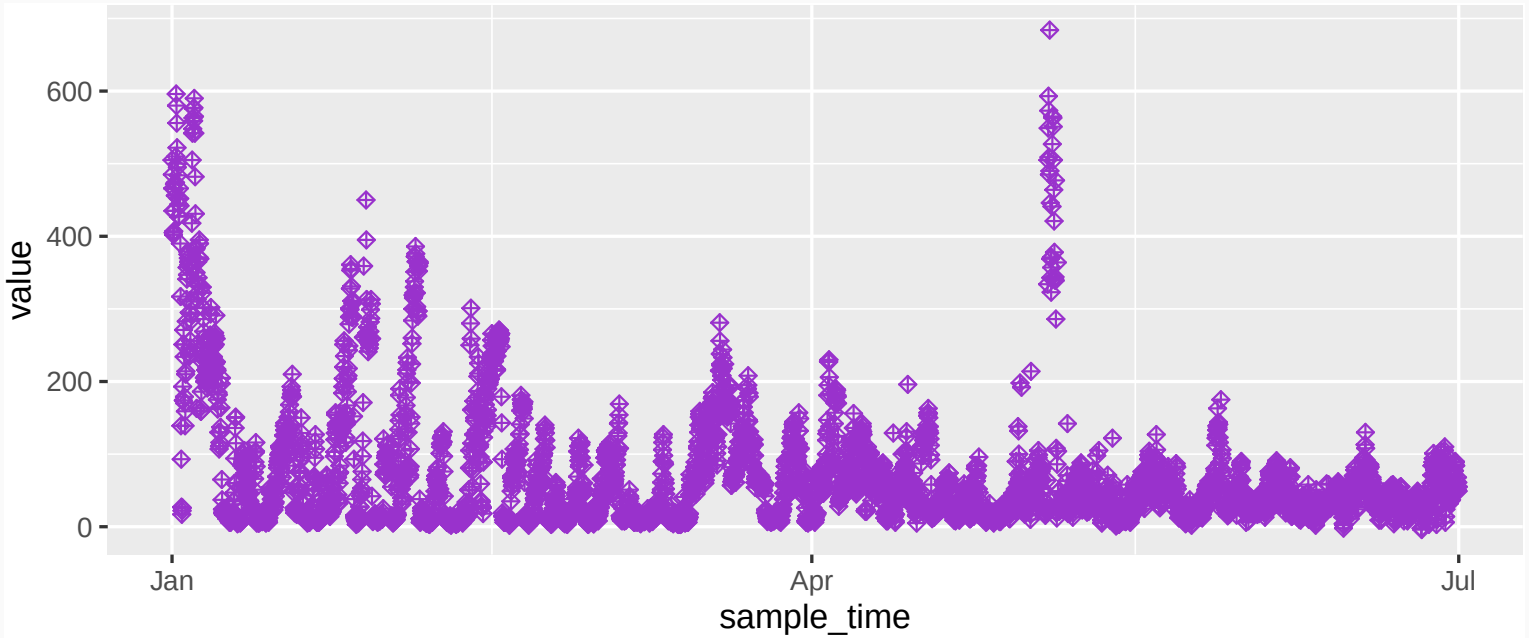



Google "R colors" and search the images to find links to listings of different R colors.

# Constant aesthetics

Here is an example of mapping point shape and color to constant values other than the defaults:

```
ggplot(data = beijing_pm) +
  geom_point(mapping = aes(x = sample_time, y = value),
             shape = 9,
             color = "darkorchid")
```



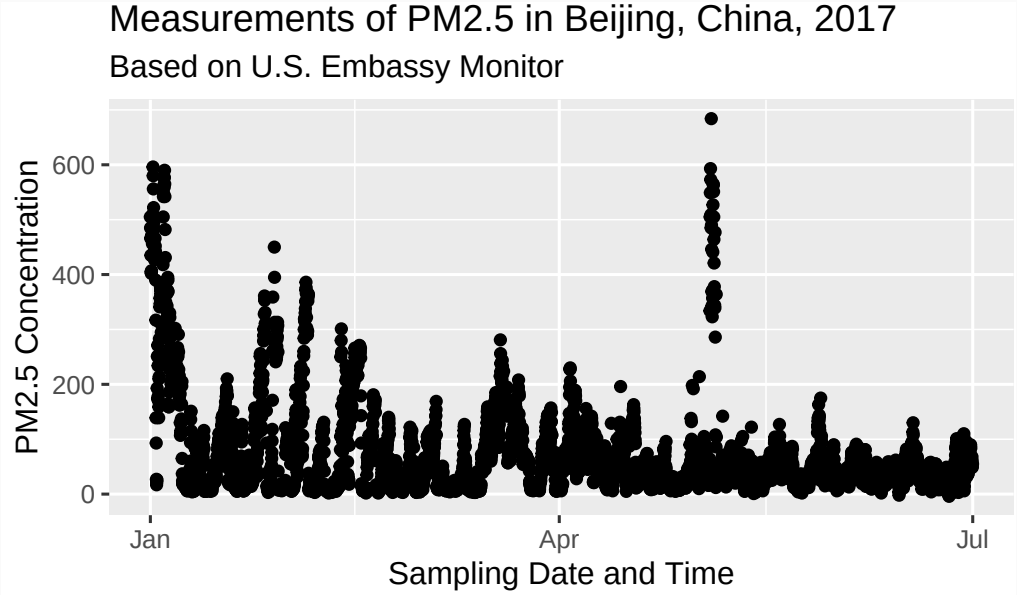
# Useful plot additions

There are also a number of elements that you can add onto a `ggplot` object using +. A few very frequently used ones are:

| Element | Description |
|---|---|
| `ggtitle` | Plot title |
| `xlab`, `ylab`, `labs` | x- and y-axis labels |
| `xlim`, `ylim` | Limits of x- and y-axis |
| `expand_limits` | Include a value in a range |

# Useful plot additions

```
ggplot(data = beijing_pm) +
  geom_point(mapping = aes(x = sample_time, y = value)) +
  labs(x = "Sampling Date and Time",
       y = "PM2.5 Concentration") +
  ggtitle("Measurements of PM2.5 in Beijing, China, 2017",
          subtitle = "Based on U.S. Embassy Monitor")
```



# Adding geoms

There are a number of different `geom_*` functions you can use to add geoms to a plot. They are divided between geoms that directly map the data to an aesthetic and those that show some summary or statistic of the data.

Some of the most common "statistical" geoms are:

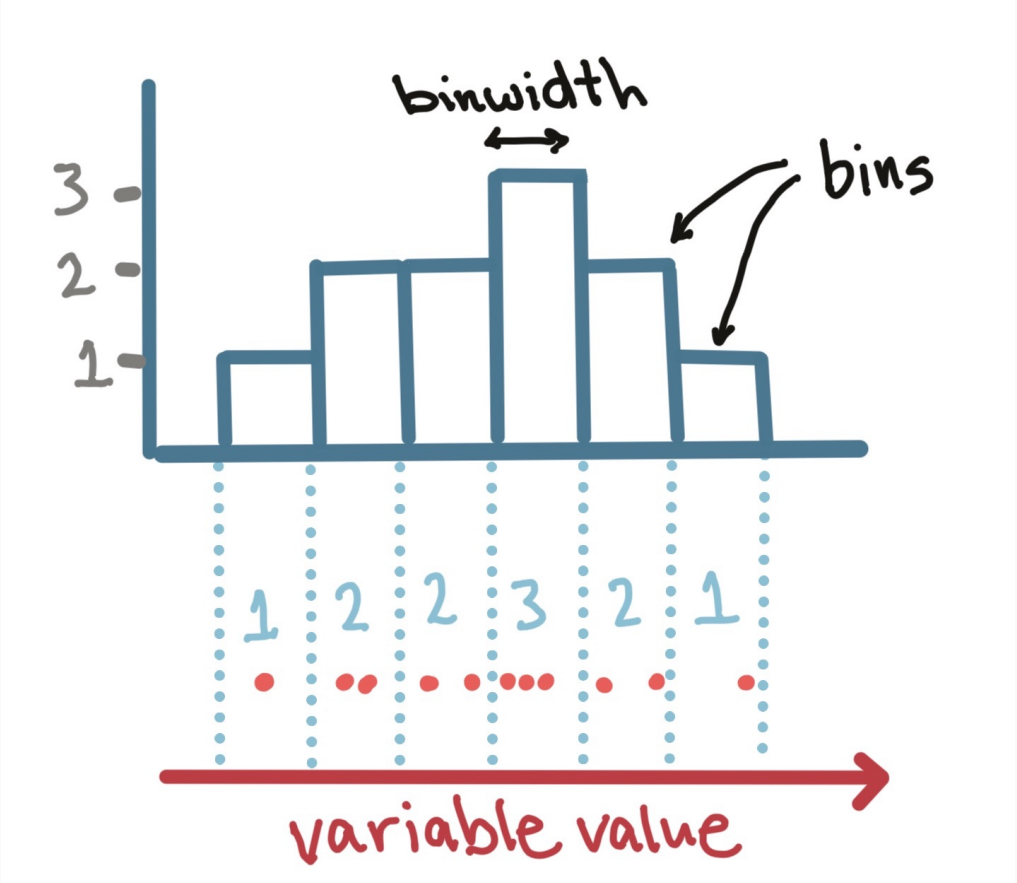
| Geom(s) | Description |
|---|---|
| `geom_histogram` | Show distribution in 1-D |
| `geom_hex`, `geom_density` | Show distribution in 2-D |
| `geom_col`, `geom_bar` | Create bar charts |
| `geom_boxplot`, `geom_dotplot` | Create boxplots and related plots |
| `geom_smooth` | Add a fitted line to a scatterplot |

# Adding geoms

These “statistical” geoms all input the original data and perform some calculations on that data to determine how to plot the final geom. Often, this calculation involves some kind of summarization.

# Adding geoms

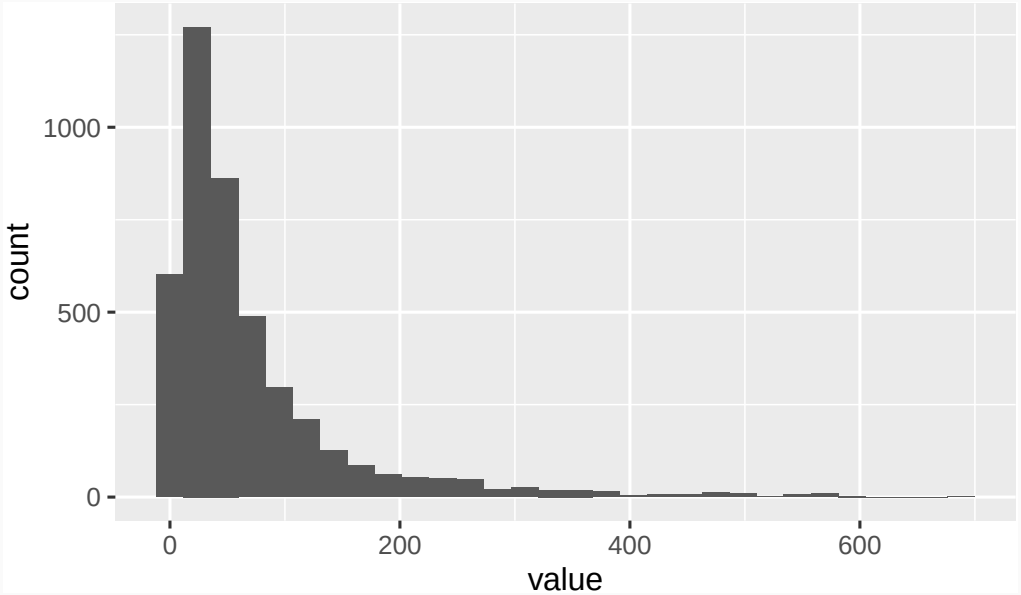
For example, the geom for a histogram (`geom_hist`) divides the data into an evenly-sized set of "bins" and then calculates the number of points in each bin to provide a visualization of how the data is distributed.



# Adding geoms

To plot a histogram of PM_{2.5} concentrations in the Beijing data, run:

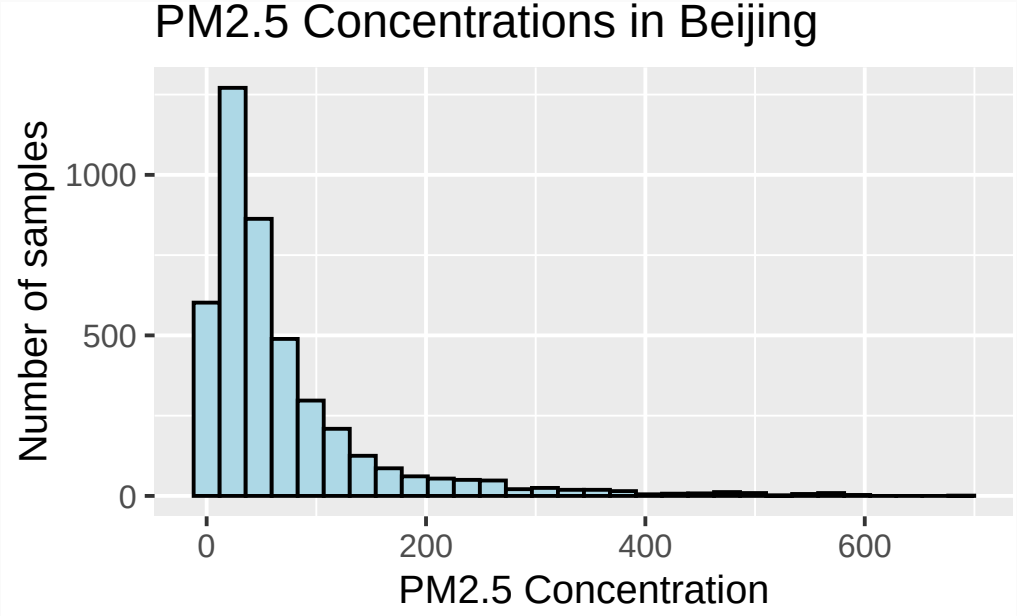
```
ggplot(data = beijing_pm) +
  geom_histogram(aes(x = value))
```



# Histogram example

You can add some elements to the histogram, like `ggtitle`, and `labs`:

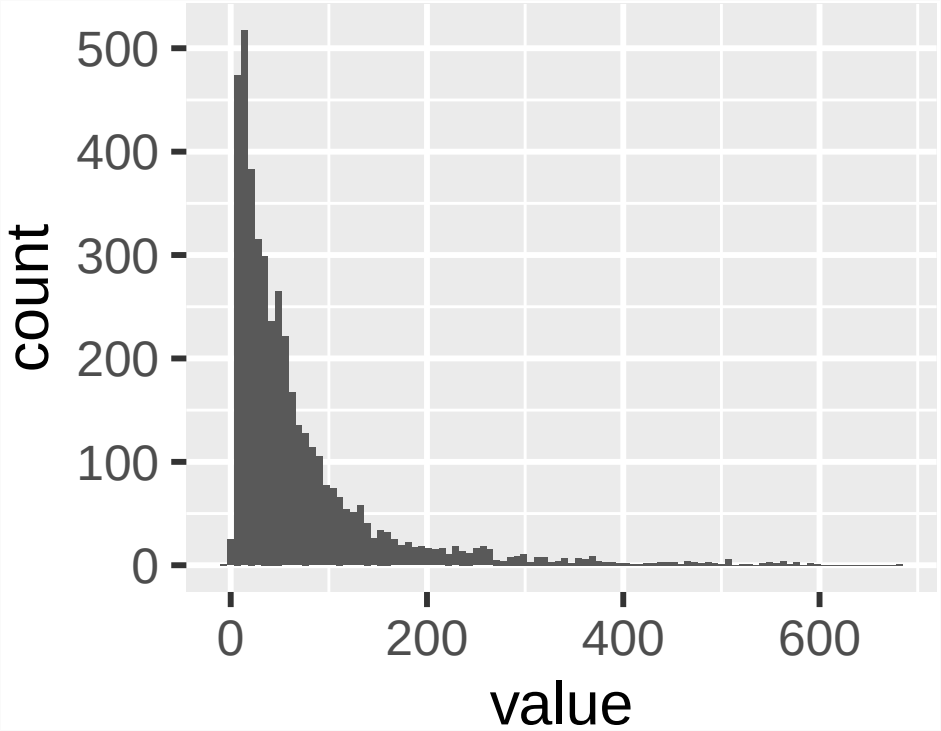
```
ggplot(beijing_pm, aes(x = value)) +
  geom_histogram(fill = "lightblue", color = "black") +
  ggtitle("PM2.5 Concentrations in Beijing") +
  labs(x = "PM2.5 Concentration", y = "Number of samples")
```



# Histogram example

`geom_histogram` also has its own special argument, `bins`. You can use this to change the number of bins that are used to make the histogram:

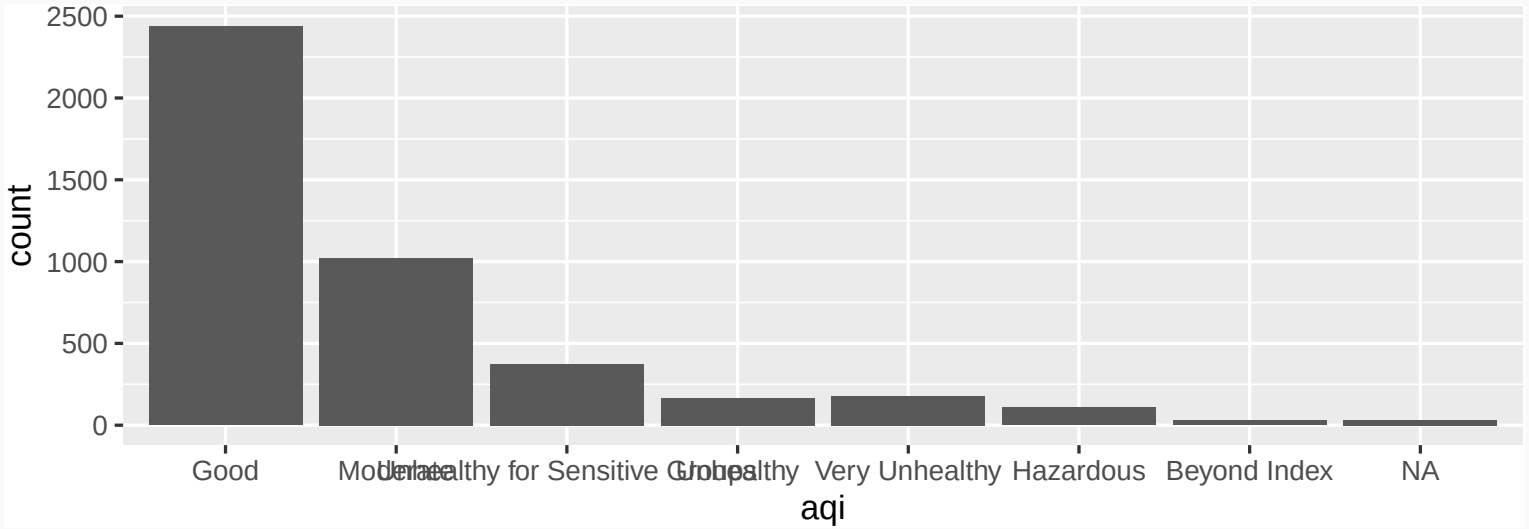
```
ggplot(beijing_pm, aes(x = value)) +
  geom_histogram(bins = 100)
```



# Bar chart

You can use the `geom_bar` geom to create a barchart:

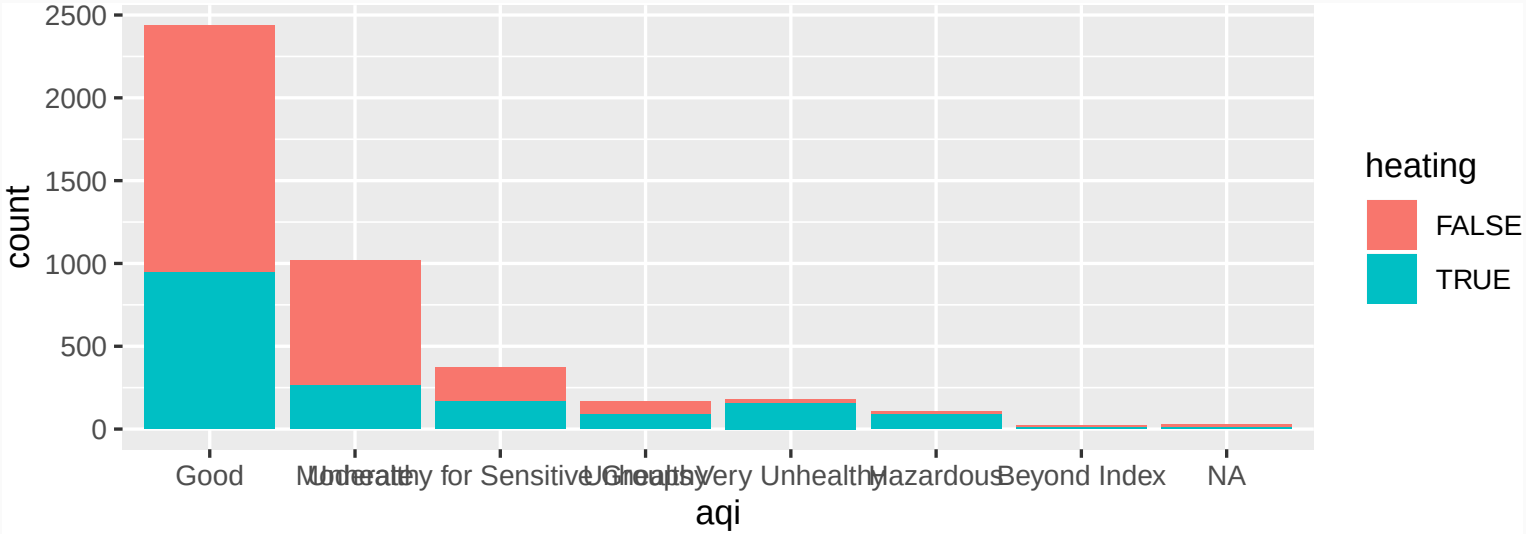
```
ggplot(beijing_pm, aes(x = aqi)) +
  geom_bar()
```



# Bar chart

You can use the `geom_bar` geom to show counts for two factors by using `x` for one and `fill` for the other:

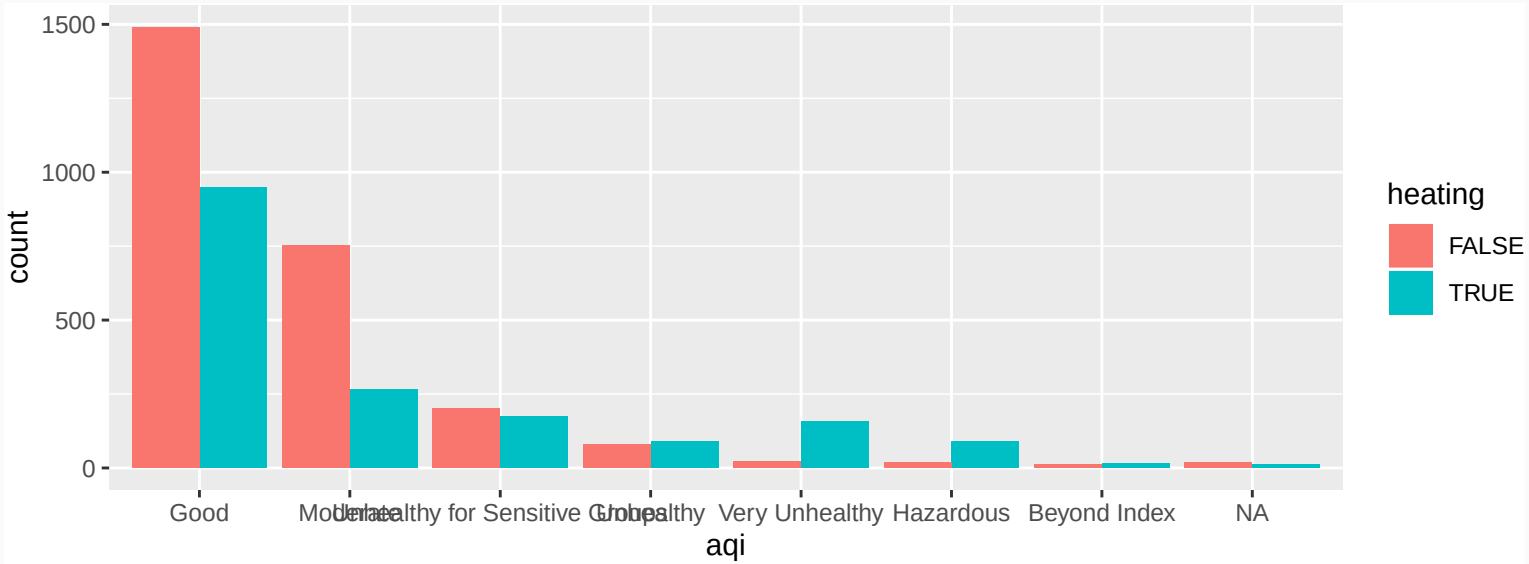
```
ggplot(beijing_pm, aes(x = aqi, fill = heating)) +
  geom_bar()
```



# Bar chart

With the `geom_bar` geom, you can use the `position` argument to change how the bars for different groups are shown (`"stack"`, `"dodge"`, `"fill"`):

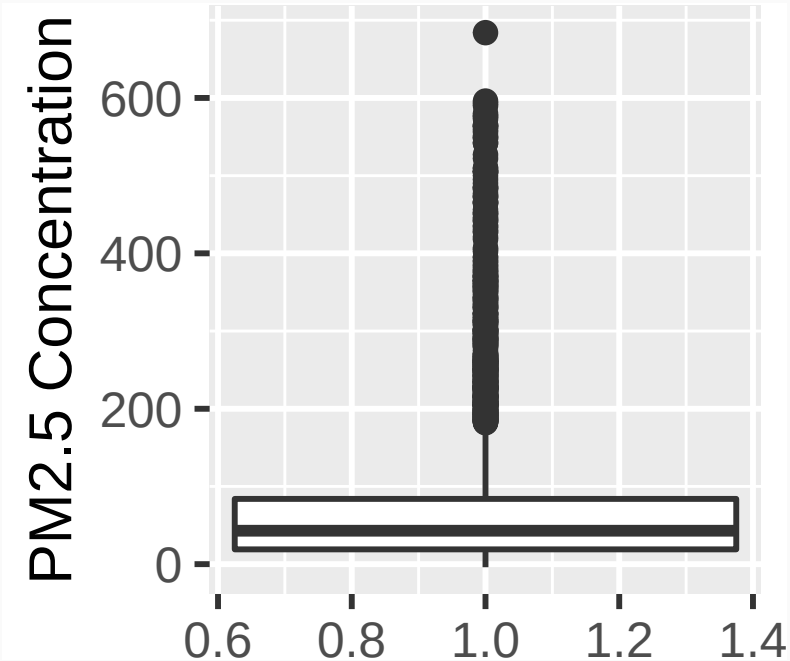
```
ggplot(beijing_pm, aes(x = aqi, fill = heating)) +
  geom_bar(position = "dodge")
```



# Boxplot example

To create a boxplot, you can use `geom_boxplot`:

```
ggplot(beijing_pm, aes(x = 1, y = value)) +
  geom_boxplot() +
  labs(x = "", y = "PM2.5 Concentration")
```



# Boxplot example

You can also do separate boxplots by a factor. In this case, you'll need to include two aesthetics (`x` and `y`) when you initialize the ggplot object.

```
ggplot(beijing_pm, aes(x = aqi, y = value, group = aqi)) +
  geom_boxplot() +
  labs(x = "AQI Category", y = "PM2.5 Concentration")
```