The graphical system

“Processing large dataset with R”

Introduction to ggplot2

We will be creating plots using the **ggplot2** package.

```
> library(dplyr)
> library(ggplot2)
```

There are also other packages for creating graphics such as **grid** and **lattice**. We chose to use **ggplot2** in this book because it breaks plots into components in a way that permits beginners to create relatively complex and aesthetically pleasing plots using syntax that is intuitive and comparatively easy to remember.

Advantages of ggplot2:

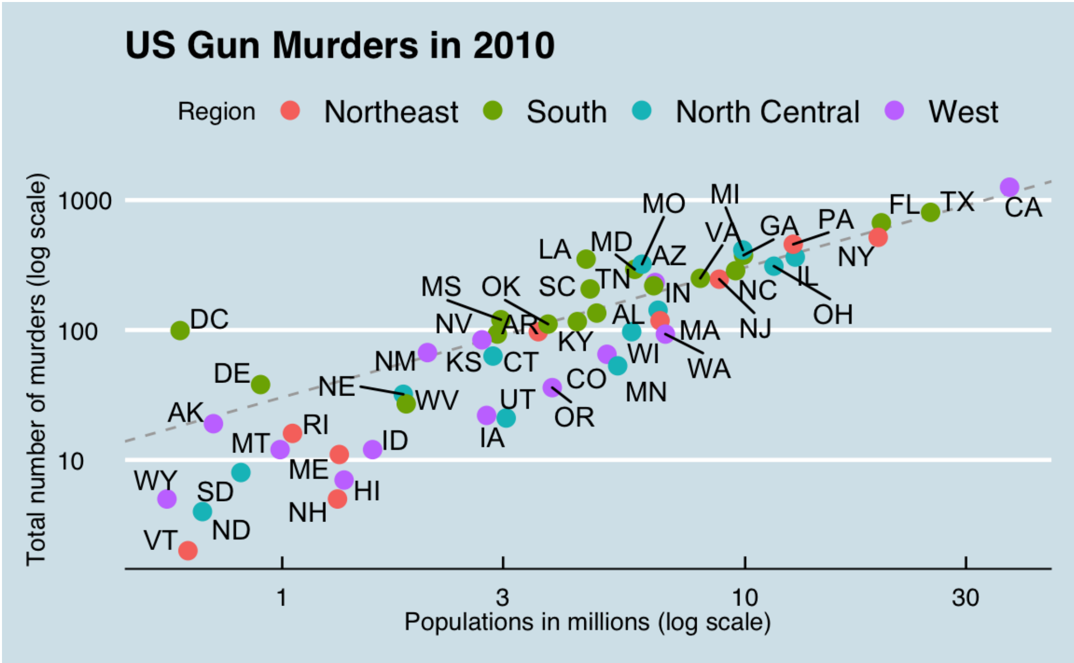
- ✓ Grammar of graphics
- ✓ Default behaviour
- ✓ ggplot2 sheet cheat

Disadvantages of ggplot2:

One limitation is that **ggplot2** is designed to work exclusively with data tables in tidy format (where rows are observations and columns are variables).

The components of a graph

We will construct a graph that summarizes the US murders dataset that looks like this:



The main three components to note are:
•**Data**: The US murders data table is being summarized.
•**Geometry**: The plot above is a scatterplot. This is referred to as the **geometry** component.
•**Aesthetic mapping**: The plot uses several visual cues to represent the information provided by the dataset.

ggplot objects

```
> ggplot(data = murders)
```
or
```
> murders %>% ggplot()
```
or
```
> p <- ggplot(data = murders)
```


no geometry has been defined!

Geometries

In ggplot2 we create graphs by adding *layers*. Layers can define geometries, compute summary statistics, define what scales to use, or even change styles. To add layers, we use the the symbol +. In general, a line of code will look like this:

DATA %>% ggplot() + LAYER 1 + LAYER 2 + … + LAYER N

Geometry function names follow the pattern: geom_X where X is the name of the geometry. Some examples include geom_point, geom_bar and geom_histogram.

```
> Aesthetics
>
> geom_point understands the following aesthetics (required aesthetics are in bold):
> x
> y
> alpha
> colour
```

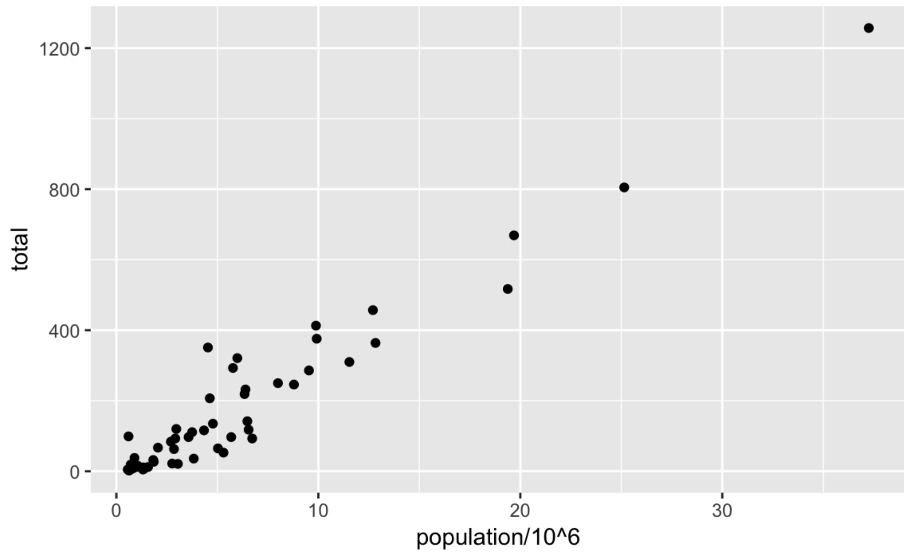
Aesthetic mappings

Aesthetic mappings describe how properties of the data connect with features of the graph, such as distance along an axis, size or color.

```
murders %>% ggplot() + geom_point(aes(x = population/10^6, y = total))
```

Or

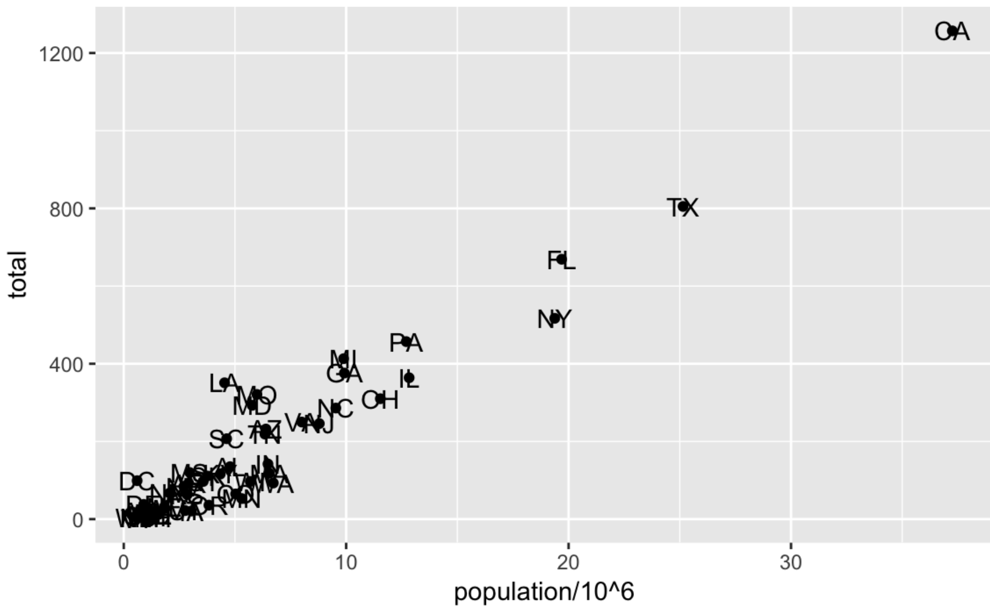
```
p + geom_point(aes(population/10^6, total))
```



Layers

A second layer in the plot we wish to make involves adding a label to each point to identify the state. The geom_label and geom_text functions permit us to add text to the plot with and without a rectangle behind the text respectively.

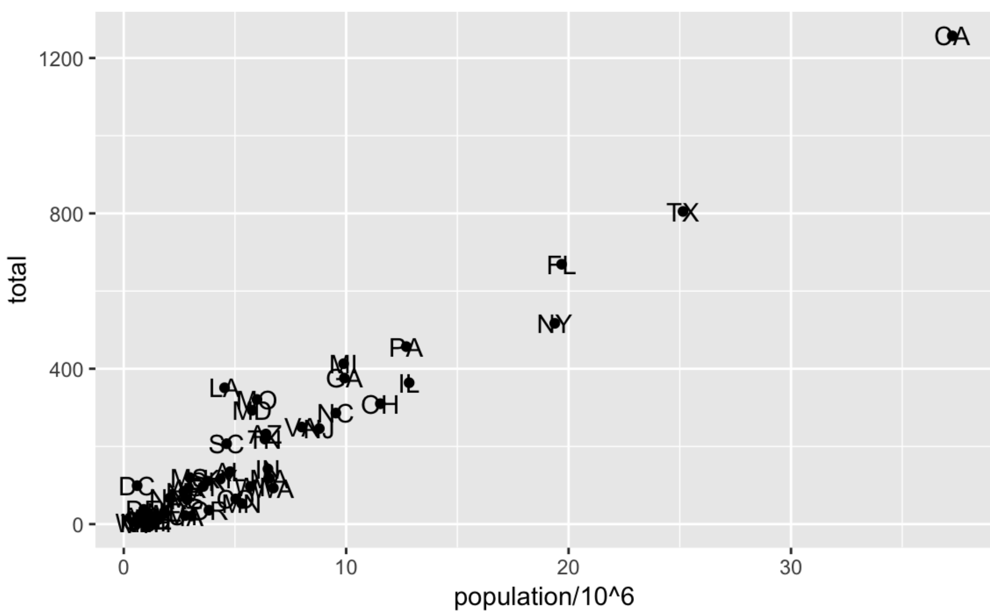
```
p + geom_point(aes(population/10^6, total)) + geom_text(aes(population/10^6, total, label = abb))
```

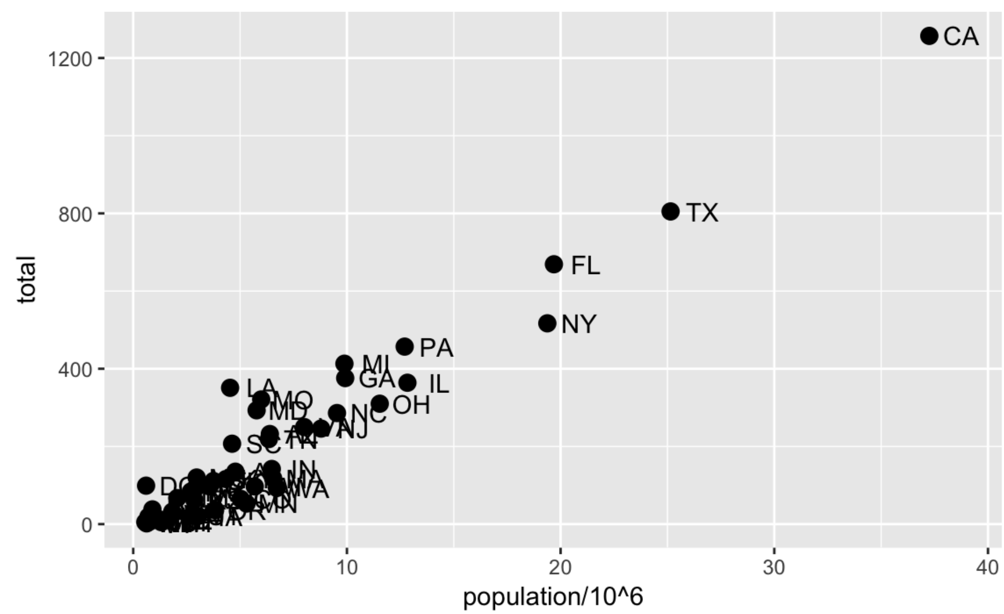


Layers

A second layer in the plot we wish to make involves adding a label to each point to identify the state. The **geom_label** and **geom_text** functions permit us to add text to the plot with and without a rectangle behind the text respectively.

```
p + geom_point(aes(population/10^6, total)) + geom_text(aes(population/10^6,
total, label = abb))

p + geom_point(aes(population/10^6, total), size = 3) +
```





Global versus local aesthetic mappings

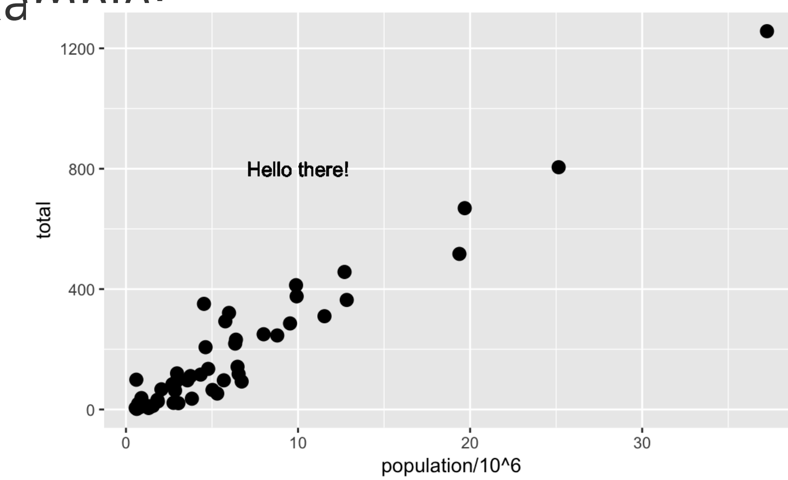
```
p + geom_point(aes(population/10^6, total), size = 3) +
geom_text(aes(population/10^6, total, label = abb), nudge_x = 1.(-5)
```

Or

```
p <- murders %>% ggplot(aes(population/10^6, total, label = abb))

p + geom_point(size = 3) + geom_text(nudge_x = 1.5)
```

If necessary, we can override the global mapping by defining a new mapping within each layer. These *local* definitions override the *global*. Here is an example:

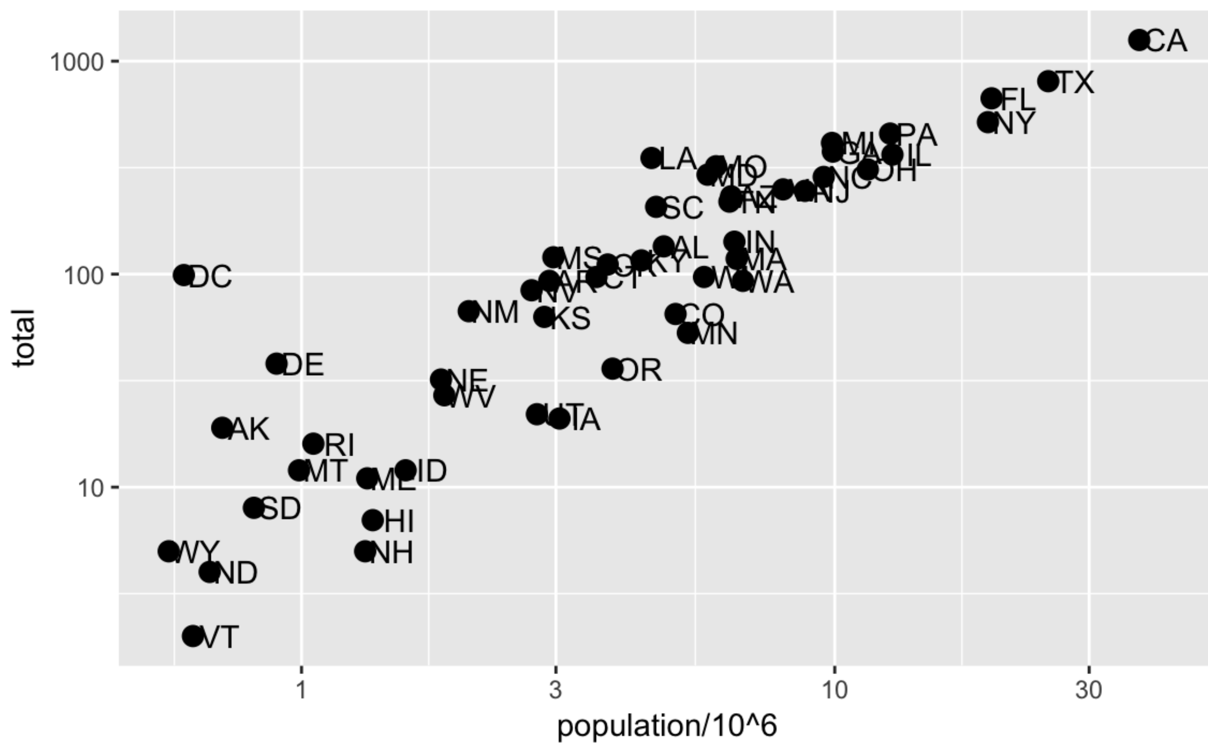
```
p + geom_point(size = 3) +
geom_text(aes(x = 10, y = 800,
label = "Hello there!"))
```



Scales

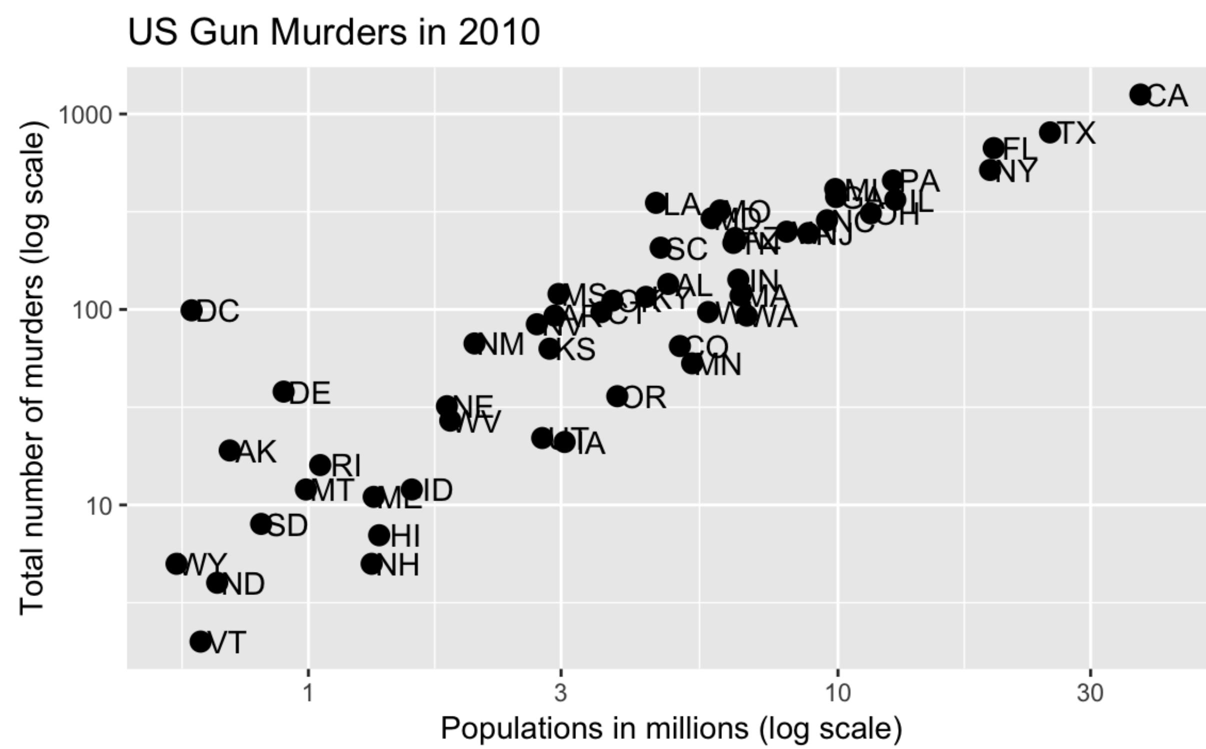
p + **geom_point**(size = 3) +
geom_text(nudge_x = 0.05) +
scale_x_continuous(trans =
"log10") +
scale_y_continuous(trans =
"log10")

p + **geom_point**(size = 3) +
geom_text(nudge_x = 0.05) +
scale_x_log10() +
scale_y_log10()



Labels and titles

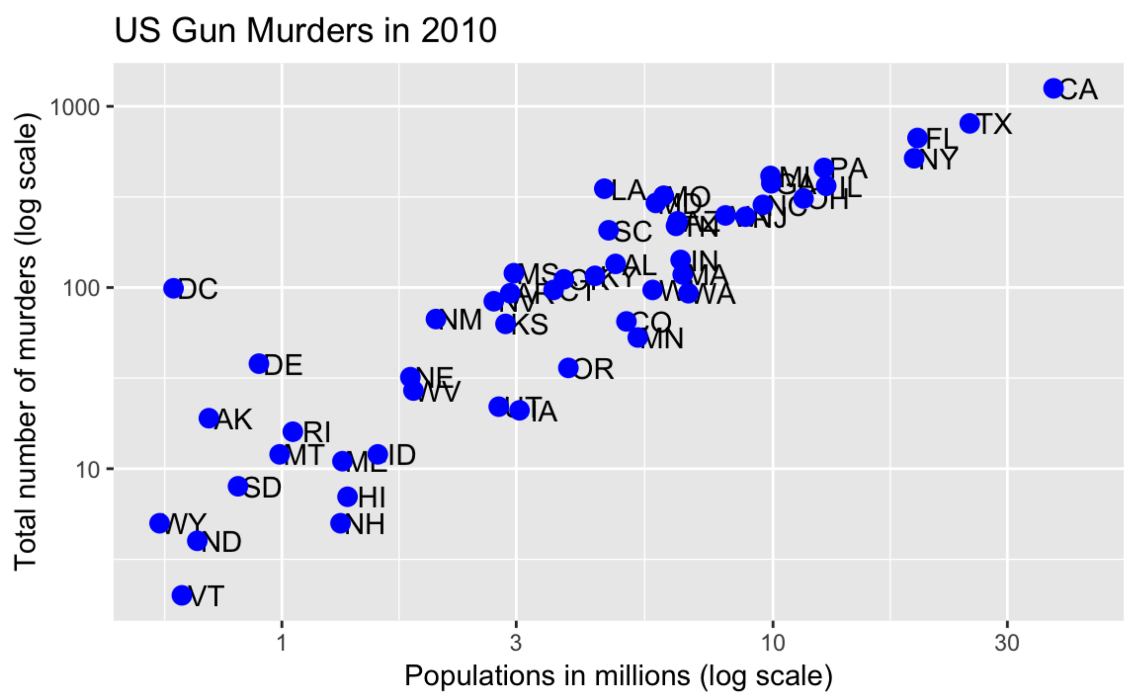
```
p + geom_point(size = 3) +
geom_text(nudge_x = 0.05) +
scale_x_log10() +
scale_y_log10() +
xlab("Populations in millions (log
scale)") + ylab("Total number of
murders (log scale)") +
ggtitle("US Gun Murders in 2010")
```



We are almost there! All we have left to do is add color, a legend and optional changes to the style.

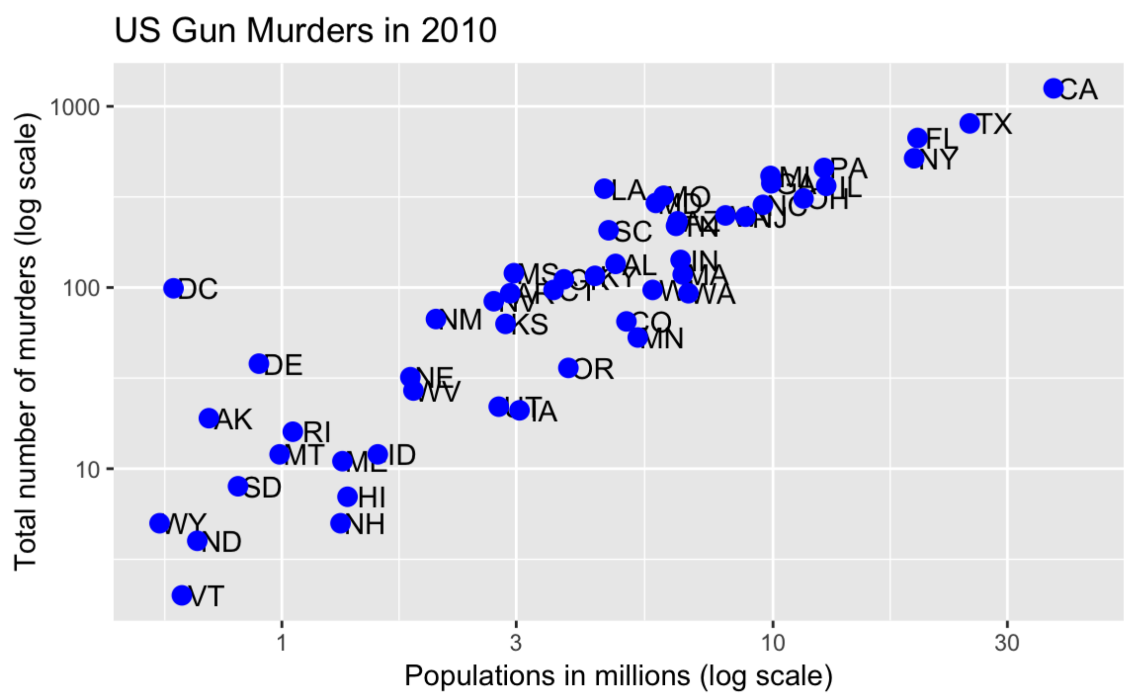
Categories as colors

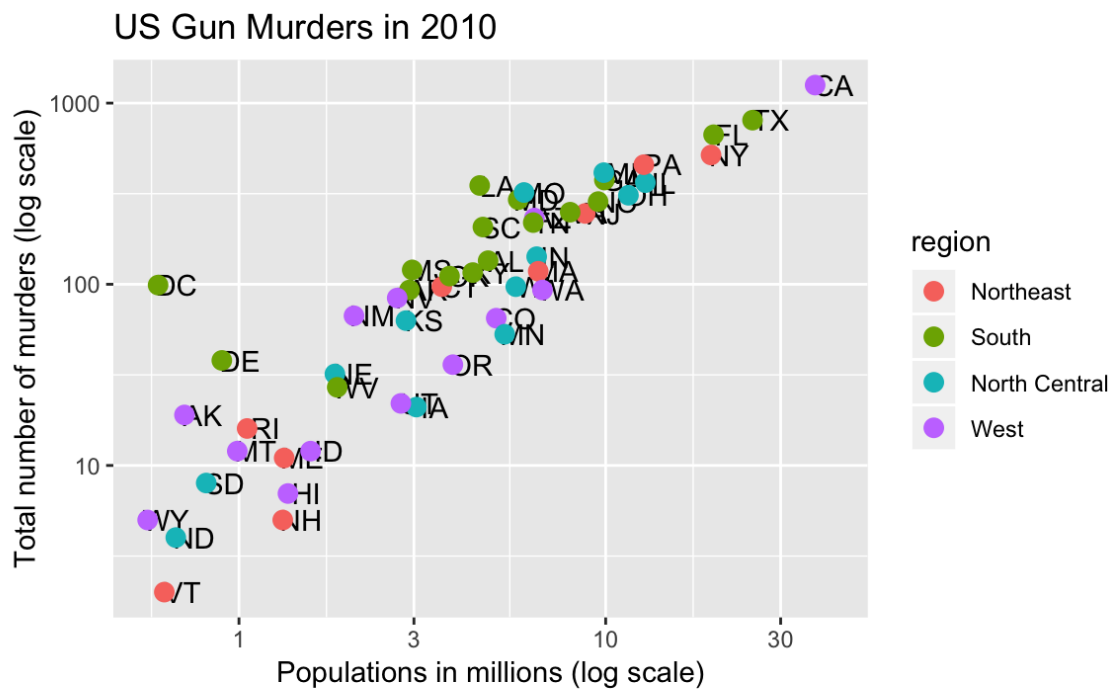
```r
p <- murders %>% ggplot(aes(population/10^6, total, label = abb)) + geom_text(nudge_x
= 0.05) + scale_x_log10() + scale_y_log10() + xlab("Populations in millions (log scale)") +
ylab("Total number of murders (log scale)") + ggtitle("US Gun Murders in 2010")

p + geom_point(color ="blue ", size = 3 )
```



Categories as colors

```
p <- murders %>% ggplot(aes(population/10^6, total, label = abb)) + geom_text(nudge_x = 0.05) + scale_x_log10() + scale_y_log10() + xlab("Populations in millions (log scale)") + ylab("Total number of murders (log scale)") + ggtitle("US Gun Murders in 2010")

p + geom_point(aes(col=region), size = 3 )
```





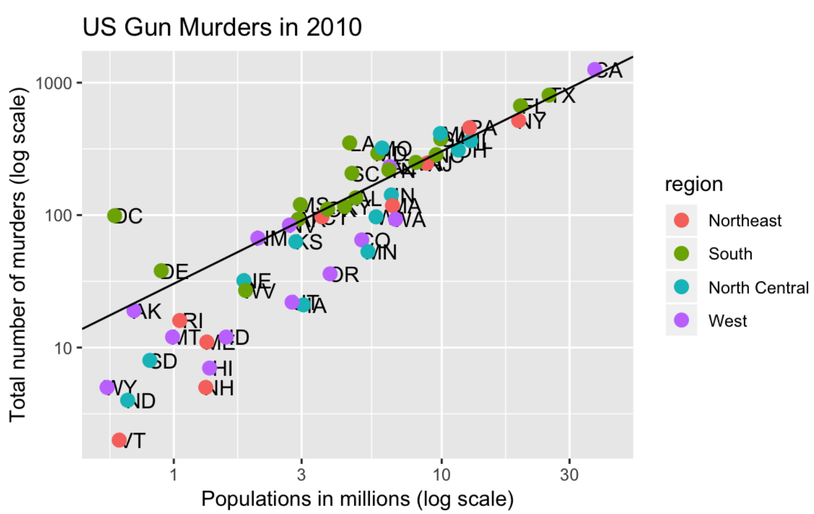
Annotation, shapes, and adjustments

Here we want to add a line that represents the average murder rate for the entire country

```
r <- murders %>% summarize(rate = sum(total) / sum(population) * 10^6) %>% pull(rate)
```

To add a line we use the geom_abline function. **ggplot2** uses ab in the name to remind us we are supplying the intercept (a) and slope (b). The default line has slope 1 and intercept 0 so we only have to define the intercept:

```
p + geom_point(aes(col=region), size = 3) + geom_abline(intercept = log10(r))
```



Add-on packages

The power of **ggplot2** is augmented further due to the availability of add-on packages. The remaining changes needed to put the finishing touches on our plot require the **ggthemes** and **ggrepel** packages.

```
library(ggthemes)
p + theme_economist()
```

The add-on package **ggrepel** includes a geometry that adds labels while ensuring that they don’t fall on top of each other. We simply change `geom_text` with `geom_text_repel`.

Putting it all together

Now that we are done testing, we can write one piece of code that produces our desired plot from scratch.

Putting it all together

```
library(ggthemes)
library(ggrepel)

r <- murders %>% summarize(rate = sum(total) / sum(population) * 10^6) %>%
pull(rate)

murders %>% ggplot(aes(population/10^6, total, label = abb)) +
geom_abline(intercept = log10(r), lty = 2, color = "darkgrey") +
geom_point(aes(col=region), size = 3) +
geom_text_repel() +
scale_x_log10() +
scale_y_log10() +
xlab("Populations in millions (log scale)") +
ylab("Total number of murders (log scale)") +
ggtitle("US Gun Murders in 2010") +
scale_color_discrete(name = "Region") +
theme_economist()
```

Quick plots with qplot

If we have values in two vectors, say:

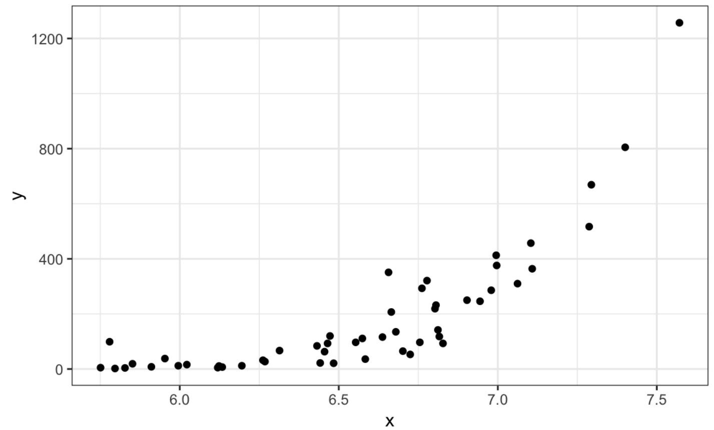
```
data(murders)
x <- log10(murders$population)
y <- murders$total
```

and we want to make a scatterplot with ggplot, we would have to type something like:

```
data.frame(x = x, y = y) %>% ggplot(aes(x, y)) + geom_point()
```

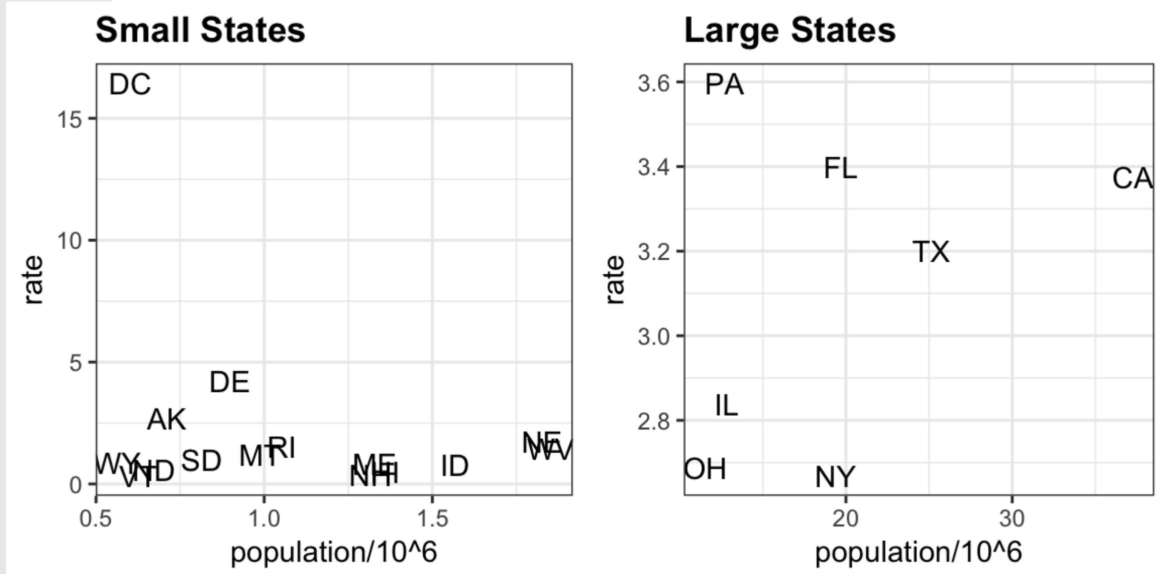
This seems like too much code for such a simple plot. The qplot function sacrifices the flexibility provided by the ggplot approach, but allows us to genera

```
qplot(x, y)
```



Grids of plots

```
library(gridExtra)
#> Attaching package: 'gridExtra’
#> The following object is masked from
'package:dplyr’:
 #> combine

p1 <- murders %>% mutate(rate =
total/population*10^5) %>%
filter(population < 2*10^6) %>%
ggplot(aes(population/10^6, rate, label = abb))
+ geom_text()
+ ggtitle("Small States")

p2 <- murders %>% mutate(rate =
total/population*10^5) %>%
filter(population > 10*10^6) %>%
ggplot(aes(population/10^6, rate, label = abb)) +
geom_text() +
ggtitle("Large States")
```



Exercise

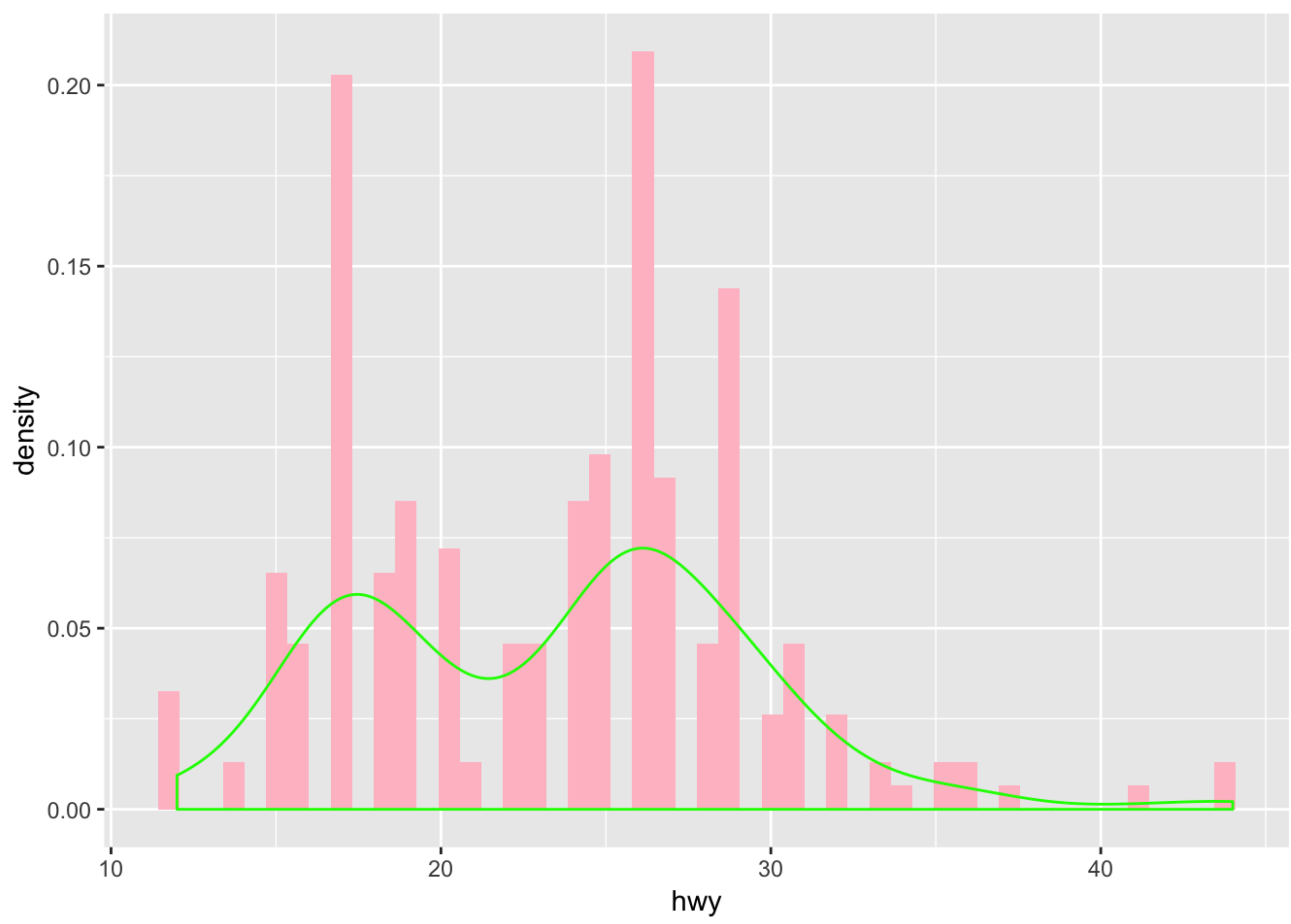
1. Create a grid of plots with (including title, axis labels, colors….):

A. state and abb.
B. total_murders and population_size.

2. Repeat the previous exercise but now change both axes to be in the log scale.

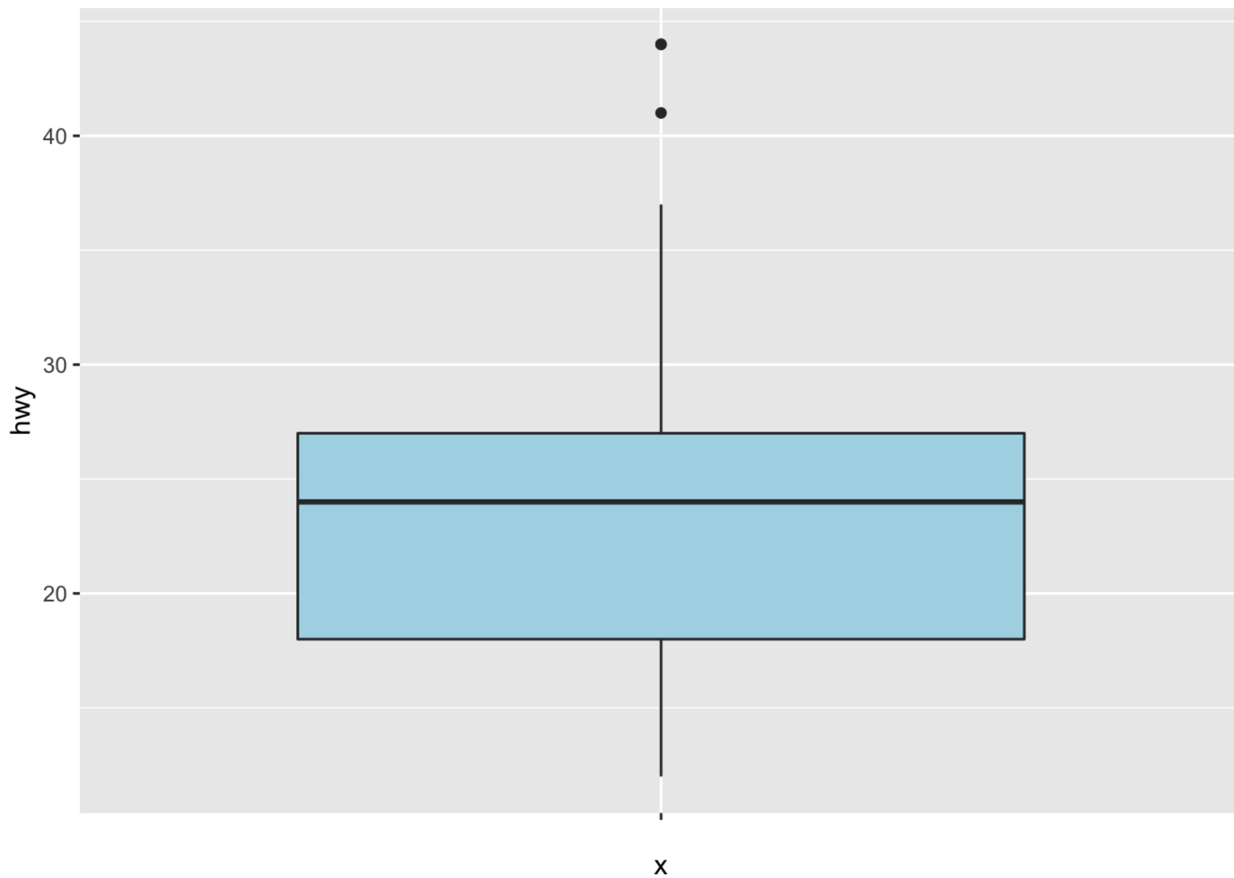
Histograms

```
ggplot(data = mpg) + geom_histogram(aes(x = hwy, y = ..density..), bins=50, fill = 'pink')
+ geom_density(aes(x = hwy),col = 'green')
```



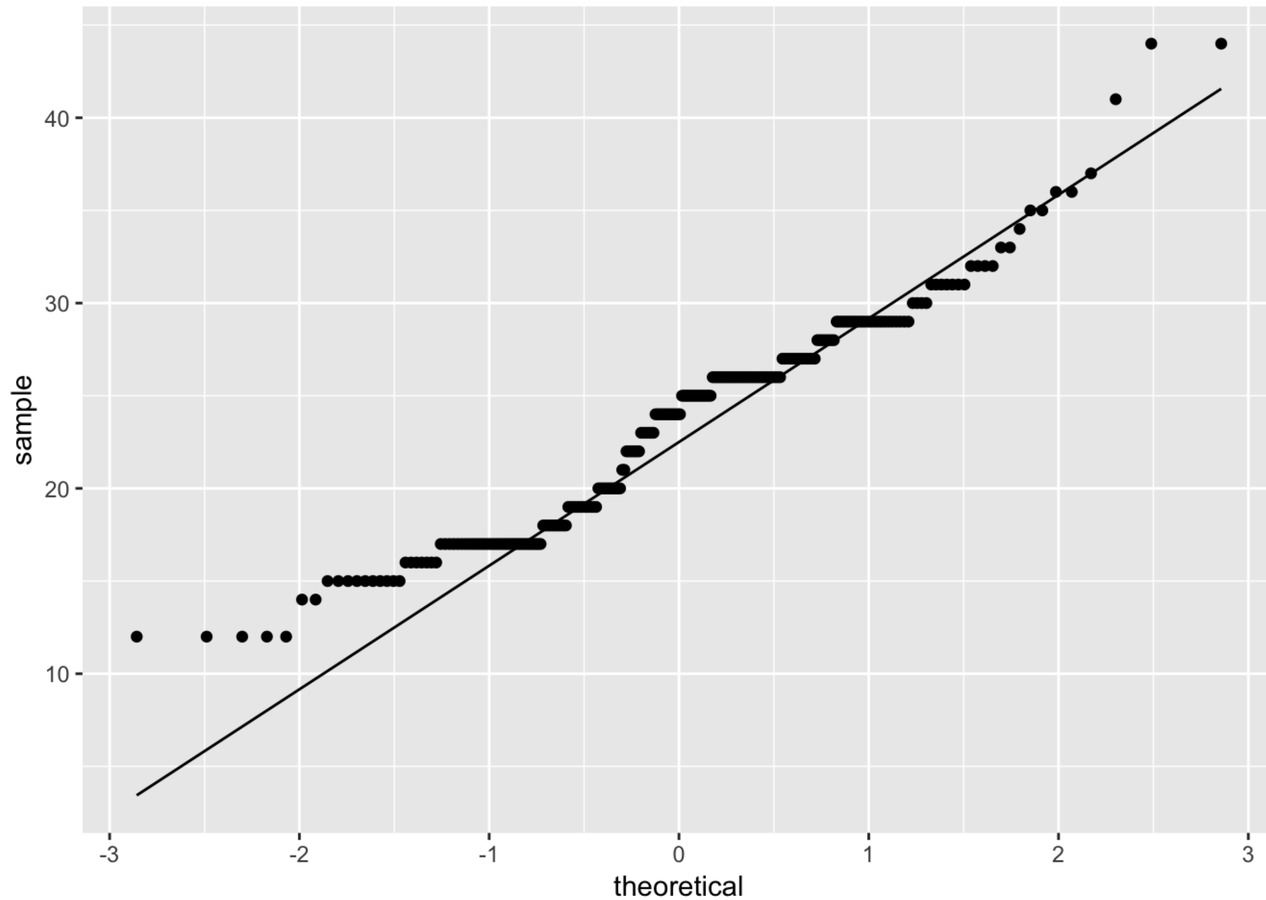
Boxplots

```
ggplot(data = mpg) + geom_boxplot(aes(x = '',y = hwy), fill = 'lightblue')
```



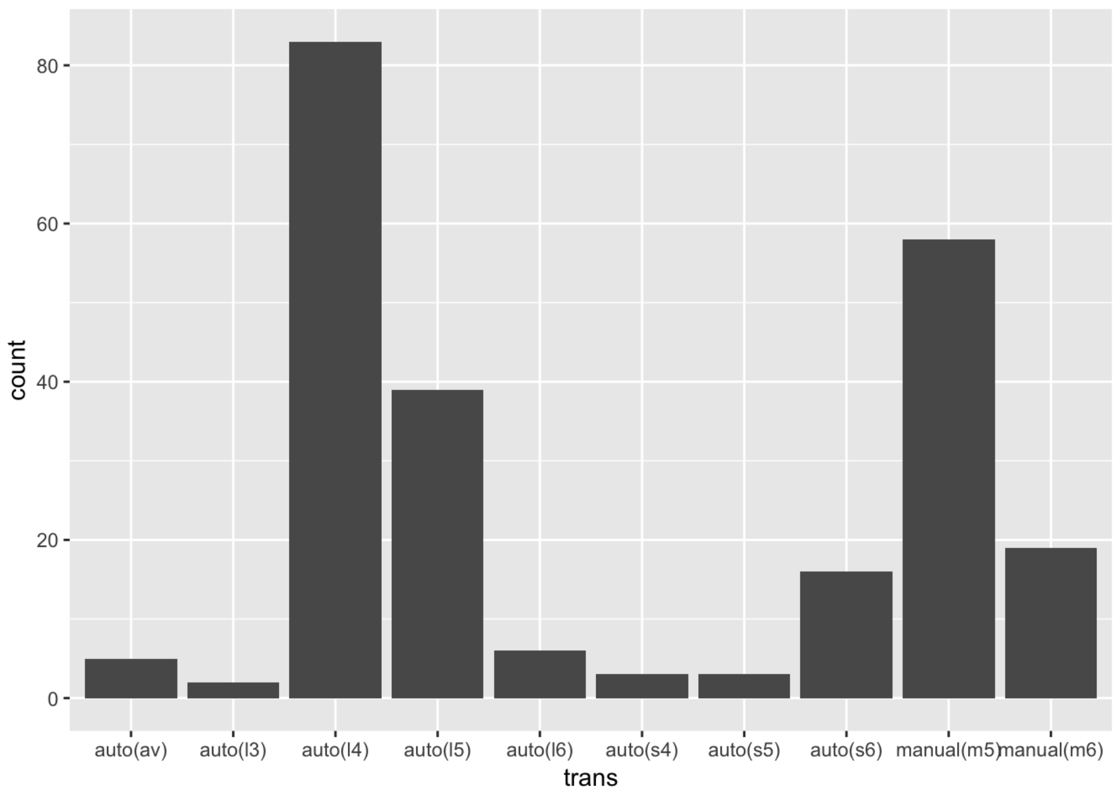
Compare distributions

```
ggplot(data = mpg) + geom_qq(aes(sample = hwy)) + geom_qq_line(aes(sample = hwy))
```

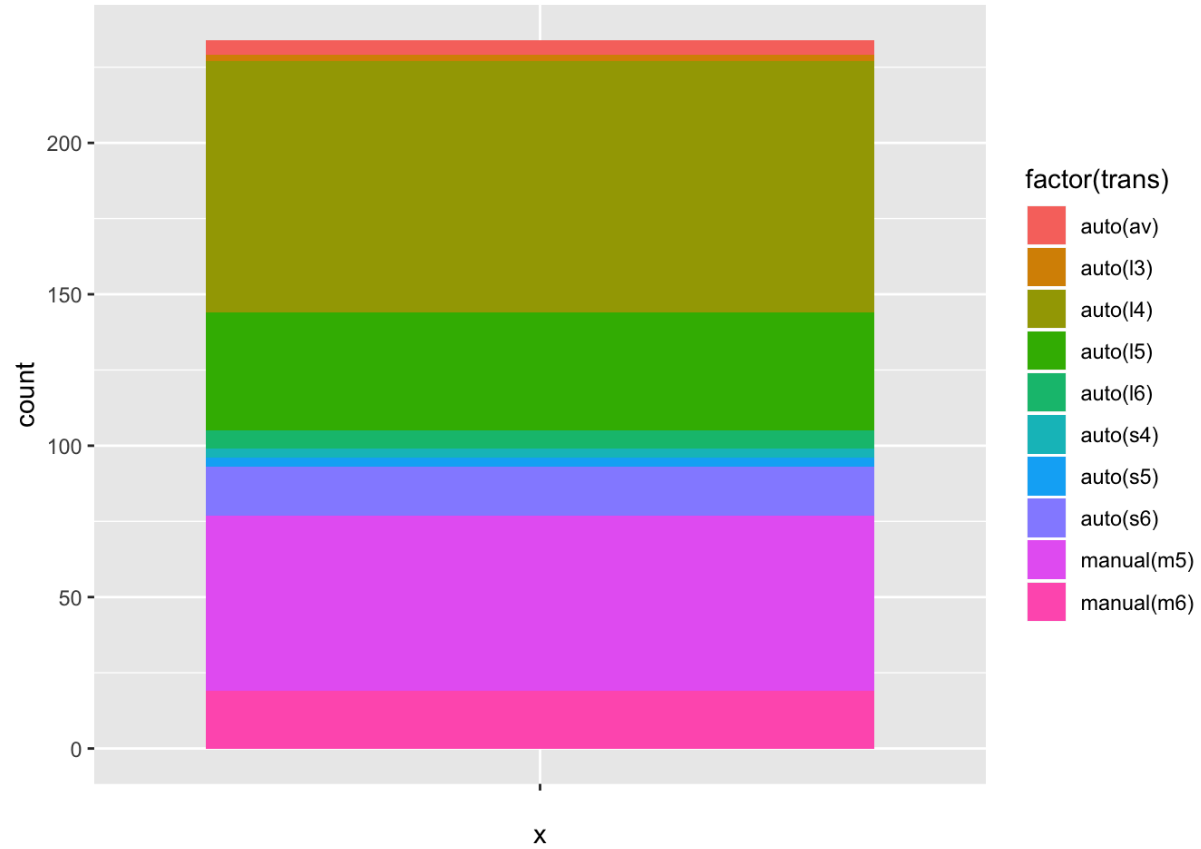


Barplots

```
ggplot(data = mpg) + geom_bar(aes(x = trans))
```

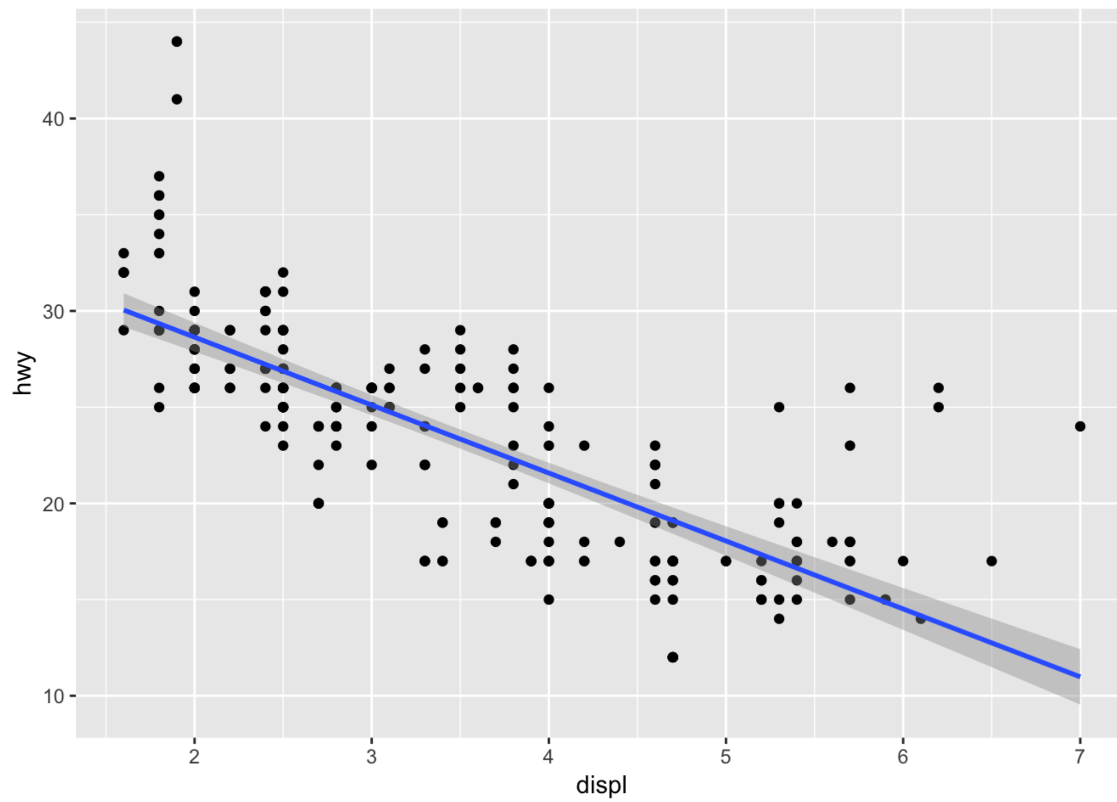



```
ggplot(data = mpg) + geom_bar(aes(x = "", fill = factor(trans)))
```



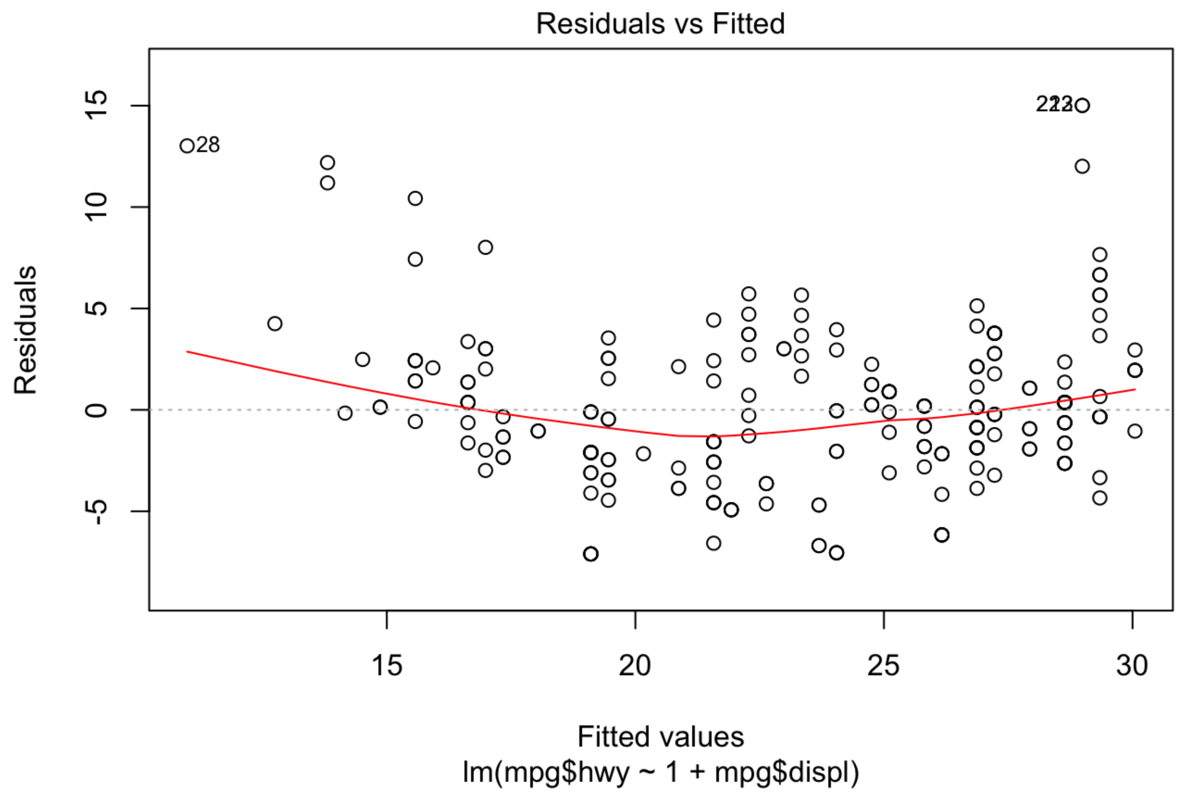
Linear model

```
gplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy)) +
geom_smooth(mapping = aes(x = displ, y = hwy),method = 'lm')
```



Linear model

```
out = lm(mpg$hwy ~ 1 + mpg$displ)
plot(out)
```



Exercise

Use the starwars data set in the dplyr package to:

- list the different human characters,
- list the different worlds,
- compute the average weight and height of the different character types,
- display on a plot the number of characters of each type in a deacresing order,

- visualize the relationship between the height and weight of the different characters.

Exercise

Compare two simulated datasets with a plot and a hypothesis test.
Use the functions below:
 visualises the two distributions with a histogram,
 uses a t-test to compares means and summarises the results with a string