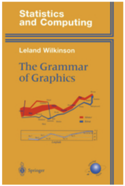
DSA2101

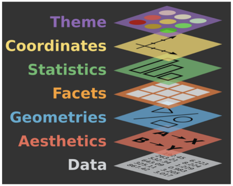
Essential Data Analytics Tools: Data Visualization

Yuting Huang

Weeks 10 Introduction to `ggplot2`

Introduction





- Just as the grammar of language that helps us construct meaningful sentences out of words, the **Grammar of Graphics** helps us construct graphs out of different visual elements.
- `ggplot2` implements the Grammar of Graphics.
  - The quality of graphs produced by this package is very high.
  - Bear in mind though, this is not the only method for making graphs in `R`.

We start by loading the required package. `ggplot2` is included in the `tidyverse` package.

```
library(tidyverse)
```

Artwork by Allison Horst

The mpg data set

Let's make a first plot using this package before we go any further.

- The `mpg` data frame in base R contains observations on 38 models of cars.
- For now, let's work with just two variables:
  - `displ`, car's engine size in litres.
  - `hwy`, fuel efficiency of the car on a highway.

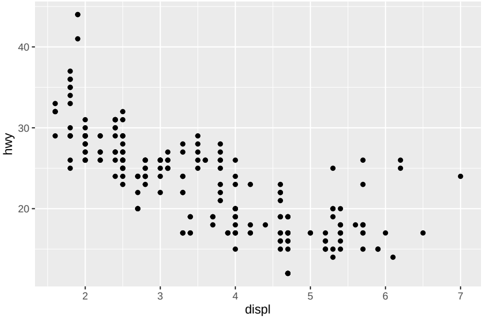
```
data(mpg)
mpg %>% head(4)
```

```
## # A tibble: 4 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    cl
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <c
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     co
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     co
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     co
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     co
```

The `mpg` data set

A scatterplot, with `displ` on the x-axis and `hwy` on the y-axis.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Breaking down the syntax

The function `ggplot()` creates a coordinate system that we can add layers to.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

- The first line is the data set to use in the plot
- `geom_point()` adds a layer of points to the plot, thus creating a scatterplot.
  - `displ` is mapped to the x-axis and `hwy` to the y-axis.

`ggplot()` template

Every `ggplot2` plot has three key components:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

1. **data**
2. A set of **aesthetic** mappings between variables in the data and visual properties.
3. At least one layer which describes how to render each observation. Layers are usually created with a **geom function**.

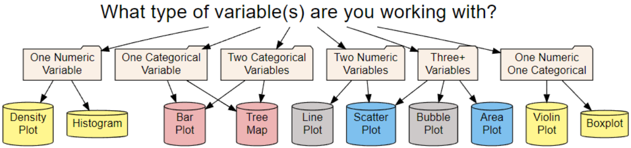
In `ggplot2`, we create graphs by adding *layers*.

Choosing the right plot

There are many `geoms` available in the `ggplot2` package.

The choice of which one to use largely depends on two questions:

- What are you trying to communicate?
- What type of variable(s) do you want to show?



Source: Alfredo Hernandez Sanchez

Outline

1. Aesthetics and geometrical objects
   - Scatterplot
   - Histogram
   - Line and text
   - Bar
   - Smoothers
   - Rug
   - Boxplot
   - Tiles and hexagons
2. Miscellaneous tasks
   - Arranging several plots
   - Themes
   - Colors

Aesthetics mappings

An **aesthetic** describes how properties of the data connects to visual properties of the graph, such as

- the position of a point.
- the size, shape, or color of the points plotted.
- the type of line (solid, dashed, etc), color and thickness of the line.

By adding aesthetics, we can extend a graph on a 2D medium to include several variables.

Geometrical objects

A **geom** refers to the geometrical object used to represent data.

In natural language, we typically use the geom to refer to a particular type of graph:

- Scatter plot uses the point geom.
- Bar chart uses the bar geom.
- Line chart uses the line geom.
- ...

Aesthetics and geoms

Each geom has a set of aesthetics associated with it.

- Some aesthetics are common to many `geoms`, but there are some aesthetics that only exist for a particular geom.
  - For instance, color, size, and coordiantes are associated with the point geom. They can also be associated with the line geom.
  - Line type is associated with the line geom, but not with the point geom.

Scatterplot: `geom_point()`

`geom_point()` is used to create scatter plots.

The aesthetics associated with it are

- `x`
- `y`
- `alpha`
- `color`
- `fill`
- `group`
- `shape`
- `size`

The defining characteristic of a point is its position, hence the `x` and `y` aesthetics are **required**. Others are optional.

How to map an aesthetic

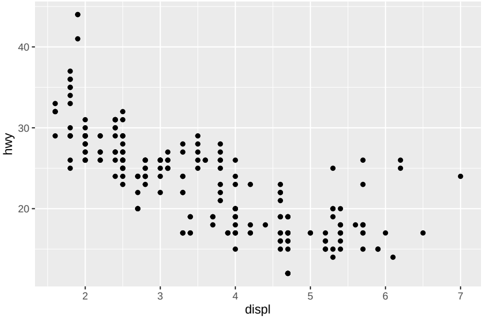
To map an aesthetic to a variable, associate the name of the aesthetic to the name of the variable inside `aes()`

- `ggplot2` will automatically assign a unique value of the aesthetic to each unique value of the variable.
- It will also add a legend that explains what levels of the aesthetic correspond to which values of the data.

A basic scatterplot

Recall the scatterplot we made under `ggplot2`:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



A basic scatterplot

Another way to write the code is:

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point()
```

Pay attention to the structure of this function call:

- Data and aesthetic mappings are supplied in `ggplot()`
- Layer(s) are added on with `+`

We can also drop the `x =` and `y =` if wanted to, since these are the first and second expected arguments of the function.

A basic scatterplot

This is an important pattern.

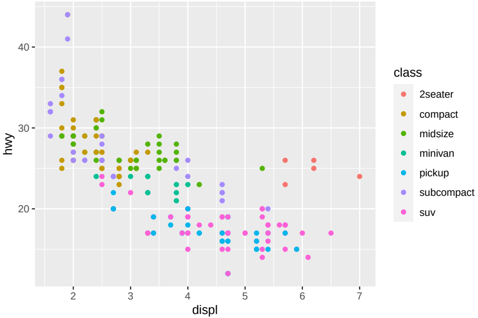
- As we learn more about `ggplot2`, we will construct increasingly sophisticated plots with multiple layers.
- Almost every layer maps a variable to `x` and `y`, so naming these aesthetics is tedious.
- We can avoid the tediousness by a **global aesthetic mapping** – supply the aesthetics in `ggplot()`, instead of individual geom functions.

In this way, all geom functions that are added as layers will default to these aesthetic mappings.

Mapping color

Suppose we want to map the colors of the points to the `class` variable.

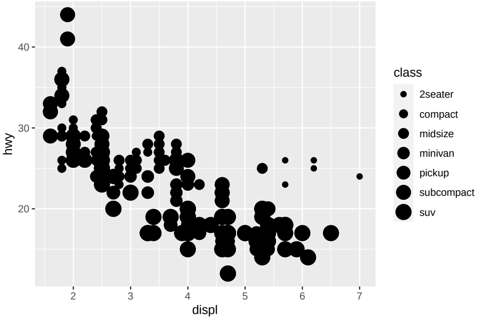
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(color = class))
```



Mapping size

If we map the size of the points, instead of their color, then we get what is sometimes referred to as a bubble chart.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(size = class))
```



Mapping size, warning message

Notice that `ggplot2` gives a warning message:

```
## Warning: Using size for a discrete variable is not advised.
```

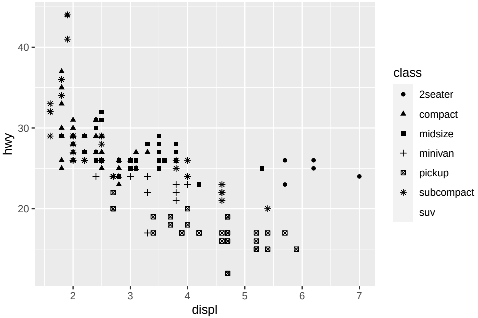
- This is because `class` is an unordered categorical variable.
- It is unclear how to map it to the size aesthetic, which is on a ratio scale. Hence the warning from `ggplot()`

Also notice that some circles are overlapping, making it difficult to see the actual size of the circles.

Maping shape

Instead of mapping size to `class`, we can use the shape aesthetic, which does not require ordering.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(shape = class))
```



Maping shape, warning message

The earlier warning no longer appears, but we get a different warning, suggesting that we have too many categories for the class variable:

```
## Warning: The shape palette can deal with a maximum of 6 discrete values beca
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 62 rows containing missing values (geom_point).
```

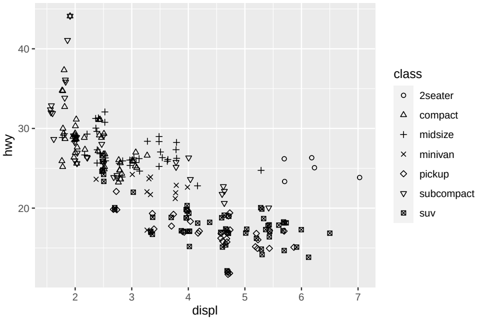
Furthermore, we still have **overplotting** – several points are plotted on top of each other.

Mapping shape (revised code)

1. To get rid of the warning, we can edit the **scale** that maps variables to shapes
   - The default scale in `ggplot2` uses only six entries. We need to manually add one.
2. To solve the overplotting problem, we need to **jitter** the points.
   - Jittering is referred to as a position adjustment for this geom.
   - The position adjustment should be specified **outside** the mapping argument of the geom function.

Maping shape (revised code)

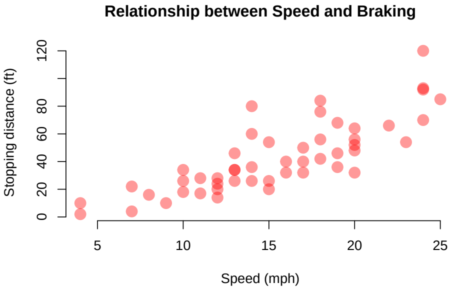
```r
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(shape = class), position = "jitter") +
  scale_shape_manual(values = 1:7)
```



Braking distance and speed

In Week 2, we created a scatterplot on the relationship between braking distance and speed using the base R plotting function.

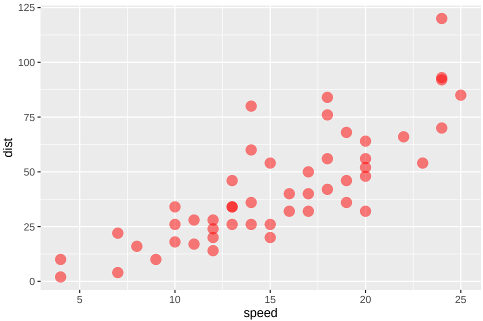
```
new_red = rgb(1, 0, 0, alpha = 0.4)
plot(cars, col = new_red, pch = 19, cex = 1.8, bty = 'n',
     xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     main = "Relationship between Speed and Braking")
```



We can recreate this in `ggplot2`:

- Set the point colors to be red using the `color` aesthetic.
- Add transparency to the points with `alpha`; change the font size with `size`.

```
ggplot(data = cars, aes(x = speed, y = dist)) +
  geom_point(color = "red", alpha = 0.5,  size = 4)
```



Braking distance and speed

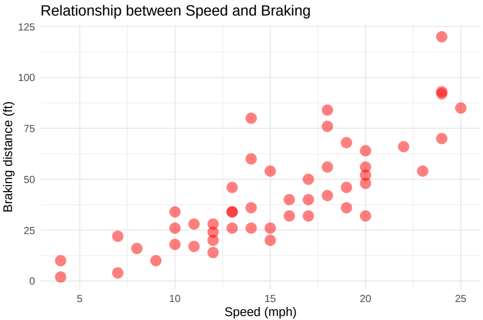
The other differences with the original plot was that we added labels and a title to the plot.

- To do so in `ggplot2`, we need to specify the `labs()` layer:
  - title
  - subtitle
  - x
  - y

We can also remove the grid lines and the background (more on this next week).

Braking distance and speed

```
ggplot(data = cars, aes(x = speed, y = dist)) +
  geom_point(color = "red", alpha = 0.5, size = 4) +
  labs(title = "Relationship between Speed and Braking",
       x = "Speed (mph)", y = "Braking distance (ft)") +
  theme_minimal()
```



Histogram: `geom_histogram()`

A histogram allows us to visualize the distribution of a single **continuous** variable.

- The x-axis will first be divided into bins. Then the number of observations in each bin will be counted.
- Three related `geoms`:
  - `geom_histogram()` displays the counts in each bin with bars.
  - `geom_density()` computes and draws kernel density estimate. It is a smoothed version of the histogram.
    - Allows comparison between distribution of a variable conditioned on a categorical one, e.g., income distribution for male and female.
  - `stat_ecdf()` displays the empirical distribution function.

Aesthetics

Some of the aesthetics for this geom are:

- `x`
- `alpha`
- `color`
- `fill`

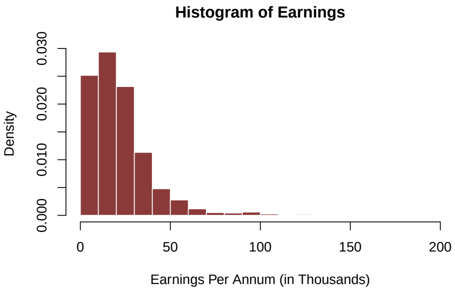
Apart from the aesthetics, we also need to consider the following issues:

- The width of the bins used
- The number of bins
- The location of the bins

Distribution of earnings

Let us revisit the histogram we encountered in Week 3.

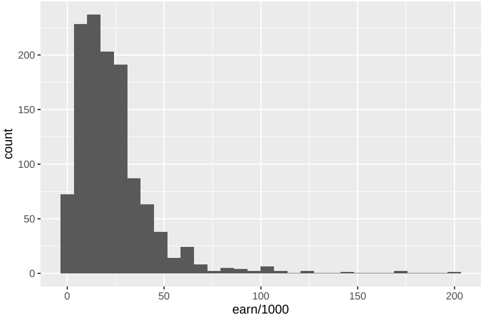
```
heights = read.csv("../data/heights.csv",
                   header = TRUE, stringsAsFactors = TRUE)
hist(heights$earn/1000, freq = FALSE,
     main = "Histogram of Earnings",
     xlab = "Earnings Per Annum (in Thousands)",
     col = "indianred4", border = "white", breaks = seq(0, 200, by = 10))
```



Distribution of earnings

Here is the first attempt to recreate it in `ggplot2`.

```
ggplot(data = heights) +
  geom_histogram(aes(x = earn/1000))
```

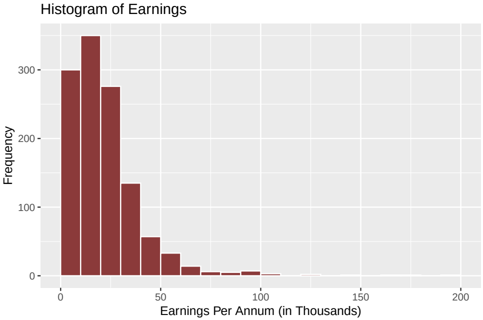


Distribution of earnings

- Let us work with bin widths of 10,000 as in Lecture 3.
- Notice that the left-most rectangle is centered at 0. This is not what we want as there are no negative incomes. We want the lower limit of the left-most bin to be 0.
- Also add color that we used last time.
- As well as labels and titles.

Distribution of earnings (revised code)

```
ggplot(data = heights) +
  geom_histogram(aes(x = earn/1000),
                 binwidth = 10, boundary = 0,
                 color = "white", fill = "indianred4") +
  labs(title = "Histogram of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Frequency")
```

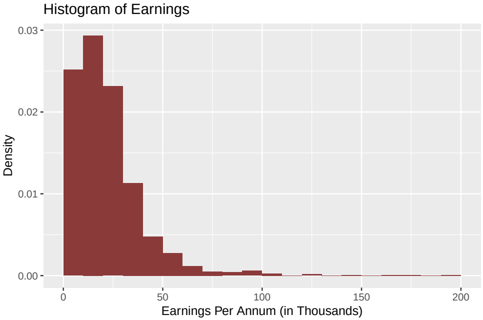


Distribution of earnings (revised code)

- The white outlines are interfering with the grid lines. We should drop them henceforth.
- In Lecture 3, we use the density for each bin, instead of counts. This makes the histogram closer in spirit to a pdf, since the area would sum to one.
  - The `geom_histogram()` computes certain summaries of the data. Among them are `count` and `density`
  - We can to tell `ggplot2` to use density instead of count on the `y` aesthetic.

Distribution of earnings (revised)

```
ggplot(data = heights) +
  geom_histogram(aes(x = earn/1000, y = ..density..),
                 binwidth = 10, boundary = 0, fill = "indianred4") +
  labs(title = "Histogram of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Density")
```



Earnings by gender

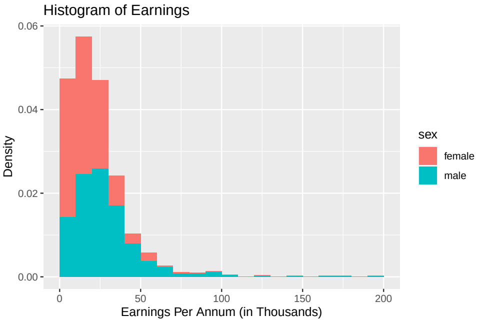
In Lecture 3, we realized that there was a stark difference between males and females in terms of income earned.

- How do we present this information?

Earnings by gender

Mapping a categorical variable to the `fill` aesthetics:

```
ggplot(data = heights) +
  geom_histogram(aes(x = earn/1000, y = ..density.., fill = sex),
                 binwidth = 10, boundary = 0) +
  labs(title = "Histogram of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Density")
```



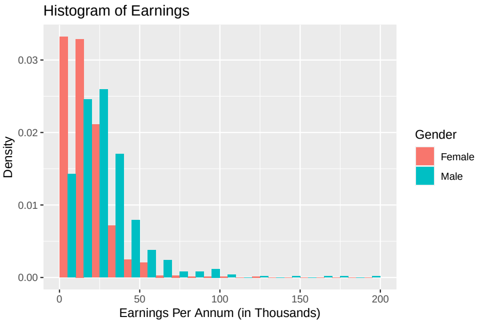
Earnings by gender

Notice that the bars for female have been stacked on top of the bars for males.

- We need a position adjustment of the bars in order to compare them side by side. To do this, we use the `position = "dodge"` adjustment.
- Also, use the `scale_fill_discrete()` function to control the `fill` scale and labels .

Earnings by gender

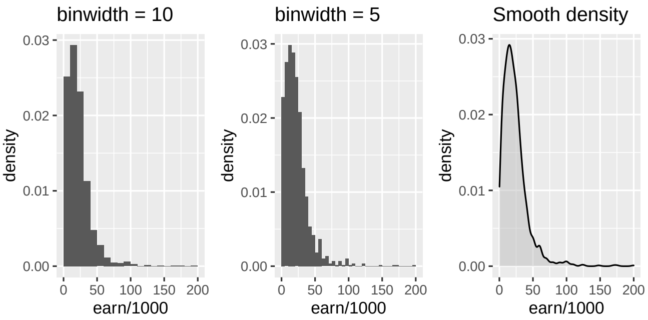
```
ggplot(data = heights, aes(x = earn/1000, y = ..density.., fill = sex)) +
  geom_histogram(binwidth = 10, boundary = 0, position = "dodge") +
  labs(title = "Histogram of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Density") +
  scale_fill_discrete(name = "Gender", labels = c("Female", "Male"))
```



Earnings, smooth density

If we want to compare distribution conditional on a categorical variable, we would be better off using smooth density plots.

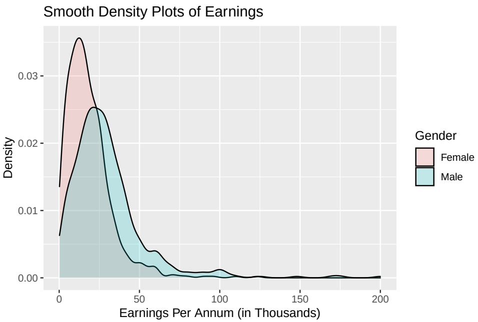
- The smooth density is a curve that gets through the top of the histogram bars when the bins are very, very small.



Earnings, smooth density

Compare densities using the `geom_density` function:

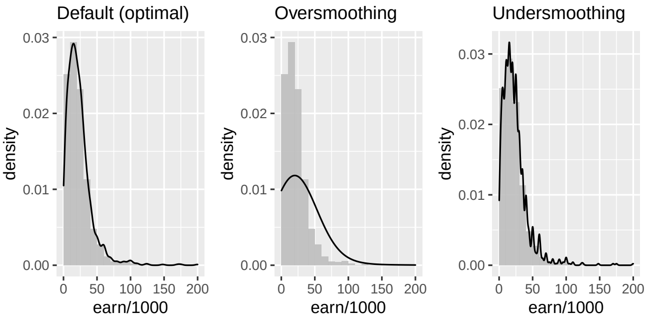
```
ggplot(heights, aes(x = earn/1000, fill = sex)) +
  geom_density(alpha = 0.2) +
  labs(title = "Smooth Density Plots of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Density") +
  scale_fill_discrete(name = "Gender", labels = c("Female", "Male"))
```



Earnings, smooth density

Note that **smoothness** is a relative term. We can actually control it through an option in the `geom_density()` function.

- The option that controls the smoothing bandwidth is `bw`.
- We should select a degree of smoothness that we can defend as being representative of the underlying data.



Earnings, eCDF

We can also use an **empirical cumulative distribution function (eCDF)** to examine a distribution.

- In `ggplot2`, we use `stat_ecdf()` to draw such graph.
- An eCDF for 20 randomly selected females.

```
set.seed(2101)
heights %>%
  filter(sex == "female") %>%
  sample_n(20) %>%
  ggplot(aes(x = earn/1000)) +
  stat_ecdf() +
  labs(title = "eCDF for Female earnings",
       x = "Earnings (in Thousands)", y = "")
```

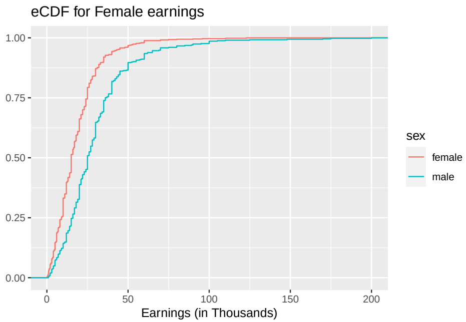
Earnings, eCDF

How to read the plot?

- About 75% of the sample earned less than 25,000 in a year.
- Only one female from this sample made more than 50,000.

Earnings, eCDF for multiple groups

```
heights %>%
  ggplot(aes(x = earn/1000, col = sex)) +
  stat_ecdf() +
  labs(title = "eCDF for Female earnings",
       x = "Earnings (in Thousands)", y = "")
```

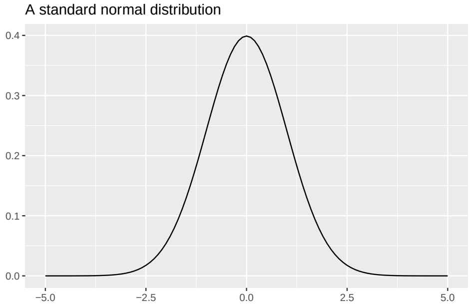


The normal distribution

Histogram and density plots provides good summaries of a distribution. What can we do further?

- We often see the average and standard deviation used as summary statistics.

```
ggplot(data = data.frame(x = seq(-5, 5)), aes(x)) +
  stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 1)) +
  labs(x = "", y = "", title = "A standard normal distribution")
```



The normal distribution

The normal distribution is one of the most famous mathematical concepts in history. Many distributions in real life can be approximated with normal distribution.

- Blood pressure, heights, weights, standardized test scores, etc.

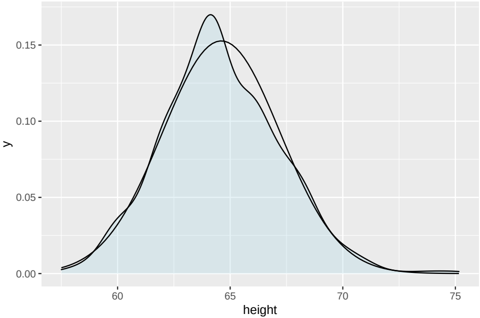
Normal distribution can be adapted to different data sets by adjusting two numbers: mean and standard deviation (SD).

- Once we are convinced that a variable has a distribution that is approximately normal,
- we can find the specific one that matches our data by matching the mean and SD of the data to the mean and SD of the normal distribution, respectively.

Distribution of `height`

Mapping the mean and SD of female heights to the arguments passed on to the normal distribution:

```
heights_female = heights %>% filter(sex == "female")
ggplot(data = heights_female, aes(x = height)) +
  geom_density(fill = "lightblue", alpha = 0.3) +
  stat_function(fun = dnorm,
                args = list(mean = mean(heights_female$height),
                            sd = sd(heights_female$height)))
```



Line: `geom_line()`

The line geom connects observations in the order of the variable on the x-axis (usually date and time).

- Suitable for plotting time-series data
- The aesthetics that the line geom uses are
  - `x`
  - `y`
  - `alpha`
  - `color`

UNESCAP population data

Let us return to the UN data set from Week 3.

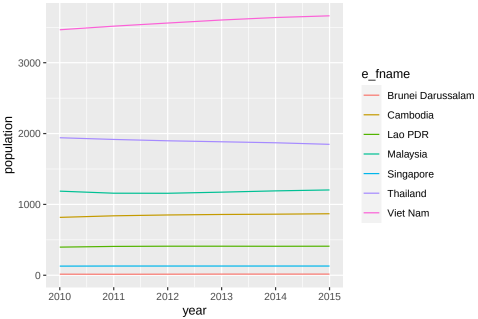
```
UN_data = readxl::read_excel("../data/UNESCAP_population_2010_2015.xlsx",
                             sheet = 3)
head(UN_data, n = 3)
```

```
## # A tibble: 3 x 7
##   e_fname     Y2010 Y2011 Y2012 Y2013 Y2014 Y2015
##   <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan  2447  2459  2454  2438  2422  2412
## 2 Armenia        92    94    97    99   101   101
## 3 Australia     710   731   740   743   745   752
```

```
pop1 = UN_data %>%
  gather(Y2010:Y2015, key = years, value = population) %>%
  mutate(year = as.integer(substr(years, 2, 5))) %>%
  filter(e_fname %in% c("Singapore", "Malaysia", "Cambodia",
                        "Thailand", "Viet Nam", "Lao PDR",
                        "Brunei Darussalam"))
```

UNESCAP population data

```
ggplot(data = pop1) +
  geom_line(aes (x = year, y = population, color = e_fname))
```



UNESCAP population data (revised code)

Several problems:

- The colors are not very helpful. We have to look very closely to distinguish them and match the lines to colors.
  - Instead, we shall use the same color for each line, and label it with texts, corresponding to the name of the country.
  - If we want to add text near the last point of each line, we will have to make space for it by extending the limits of the graph using the `xlim()` layer.
- The name "Brunei Darussalam" is too long for our purpose. We shall shorten it to "Brunei" using the `recode()` function from the `tidyverse`.

Text geom

`geom_text()` directly adds text to the plot.

- The aesthetics that it uses are
  - `x`
  - `y`
  - `label()`
- There are also additional arguments that allow us to control the position, alignment, and size of the labels.

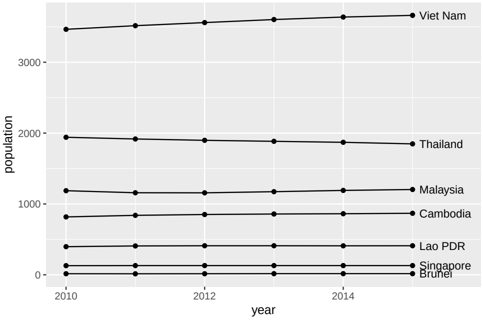
UNESCAP population data (revised code)

- Prepare the data for the text `geom`.
- Move the aesthetics mappings to the global level, inside the `ggplot()` function call.
- When necessary, we can override the global mapping by defining a new mapping within each layer.
- Also notice how we add multiple `geom`s on one graph.

```
pop2 = filter(pop1, year == 2015) %>%
  mutate(fname = recode(e_fname, "Brunei Darussalam" = "Brunei"))
```

UNESCAP population data (revised)

```
ggplot(data = pop1, aes(x = year, y = population, group = e_fname)) +
  geom_line() +
  geom_point() +
  geom_text(data = pop2, aes(label = fname),
            hjust = "left", nudge_x = 0.1, size = 3.5) +
  xlim(2010, 2015.7)
```



Reference lines

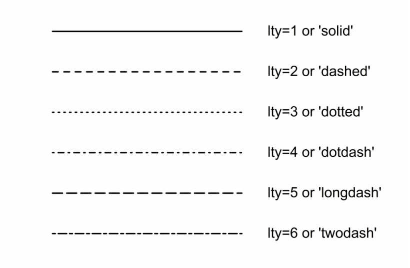
To add a reference line, we can use one of the followings:

- `geom_vline` for virtical lines
- `geom_hline` for horizontal lines
- `geom_abline` for straight lines defined by a slope or an intercept
    - `ggplot2` uses `ab` in the name to remind us that we are supplying the intercept (`a`) and slope (`b`).

Line types

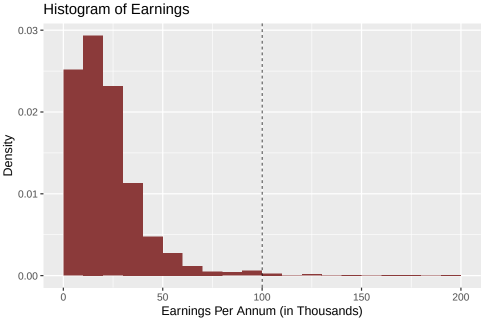
Suppose that in the income distribution histogram, we want to indicate high-income earners (more than 100K per annum) with a dashed vertical line.

- The argument `lty` or `linetype` specifies the type of the line.
- The argument `lwd` or `size` controls the thickness of the line.



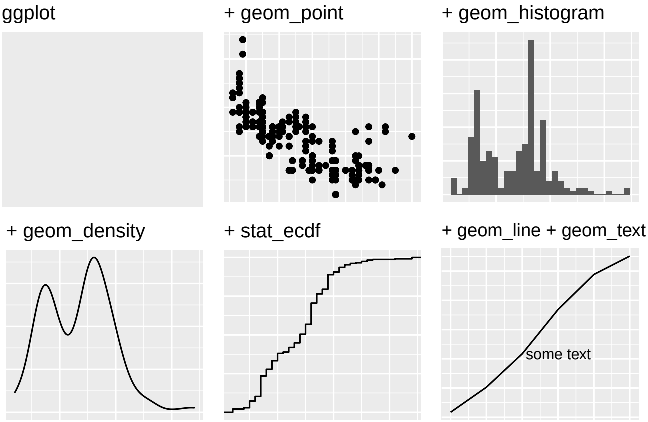
Reference lines

```
ggplot(data = heights, aes(x = earn/1000, y = ..density..)) +
  geom_histogram(binwidth = 10, boundary = 0, fill = "indianred4") +
  geom_vline(aes(xintercept = 100), linetype = 2, size = 0.3) +
  labs(title = "Histogram of Earnings",
       x = "Earnings Per Annum (in Thousands)", y = "Density")
```



First summary on `ggplot2`

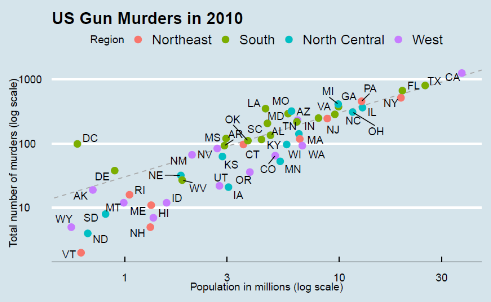
Summary on some `geoms` and `stats` we have learned so far.



Case study: US gun murders

Last week, we examined the components of a graph on US gun murders.

- We now construct this plot layer-by-layer in `ggplot2`.



US gun murders

We start by loading the data set:

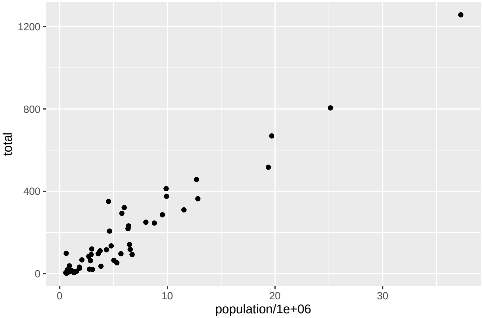
```r
# install.packages("dslabs")
library(dslabs)
df = murders
head(df)
```

```
##        state abb region population total
## 1    Alabama  AL  South    4779736   135
## 2     Alaska  AK   West     710231    19
## 3    Arizona  AZ   West    6392017   232
## 4   Arkansas  AR  South    2915918    93
## 5 California  CA   West   37253956  1257
## 6   Colorado  CO   West    5029196    65
```

US gun murders

1. Aesthetic mappings in a point `geom`:

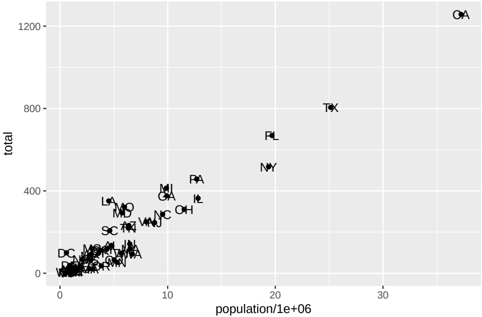
```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total))
```



US gun murders

2. A second layer of the plot: Labels to each point to identify the state.

```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total)) +
  geom_text(aes(population/1e6, total, label = abb))
```



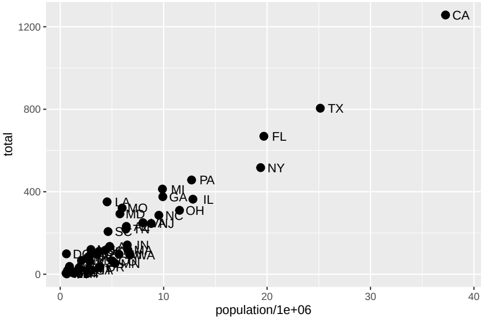
US gun murders

3. Tinkering the arguments:
    - In the original plot, the points are larger than the default size. We can change the point size using the `size` argument in `geom_point`.
    - Now because the points are larger, it is hard to see the labels.
    - We can move the text slightly to the right or to the left using the `nudge_x` arguemnt in `geom_text()`.

```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total), size = 3) +
  geom_text(aes(population/1e6, total, label = abb), nudge_x = 1.5)
```

US gun murders

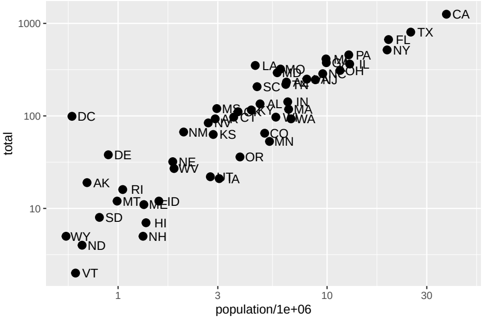
3. Tinkering the arguments makes the plot easier to read.



4. Scales

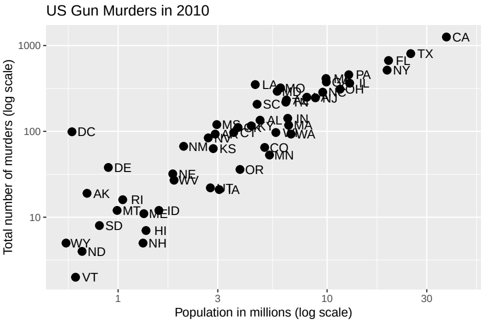
- The original plot is in log-scale. This is not the default in `ggplot2`. We can use the `scale_x_log10()` function to control the behavior of scales of the x axis.
- Because we are in log-scale now, the *nudge* must be made smaller.

```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total), size = 3) +
  geom_text(aes(population/1e6, total, label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10()
```



5. Next, change the labels and add a title.

```r
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total), size = 3) +
  geom_text(aes(population/1e6, total, label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)")
```



US gun murders

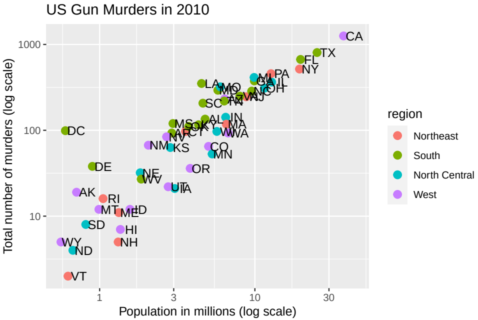
6. Categories as colors.
    - Change the color of the points using the `col` argument in the `geom_point()` function.
    - Since the choice of color is determined by a feature of each observation, this is an aesthetic mapping – we need to use it inside `aes`.

```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total, col = region), size = 3) +
  geom_text(aes(population/1e6, total, label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)")
```

US gun murders

6. Categories as colors.

   - `ggplot2` automatically adds a legend that maps color to region. To disable it, we can further set the `geom_point()` argument `show.legend = FALSE`.



US gun murders

7. Annotation, shapes, and adjustments:
   - Next, we want to add a dashed line that represents the average murder rate for the entire country.
   - The line is defined by the formula: $y = rx$.
   - In log-scale, this line turns into $\log(y) = \log(r) + \log(x)$.
   - So in our plot, it is a line with slope 1 and intercept $\log(r)$.
   - To compute this value, we use our `tidyverse` skills:

```
r = df %>%
  summarize(rate = sum(total) / (sum(population) / 1e6)) %>%
  pull(rate)          # extract a single column
```

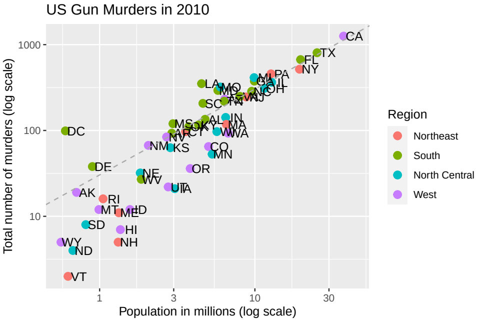
US gun murders

7. Annotation, shapes, and adjustments:
    - To add the line, we use the `geom_abline()` function.
    - We can change the line type and line color using arguments.

```
ggplot(data = df) +
  geom_point(aes(x = population/1e6, y = total, col = region), size = 3) +
  geom_text(aes(population/1e6, total, label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)") +
  geom_abline(slope = 1, intercept = log10(r),
              linetype = 2, color = "darkgrey") +
  scale_color_discrete(name = "Region")
```

US gun murders

7. Annotation, shapes, and adjustments.



US gun murders

Simplify our code using global aesthetic mappings:

```r
ggplot(data = df, aes(x = population/1e6, y = total, col = region)) +
  geom_point(size = 3) +
  geom_text(aes(label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)", col = "Region") +
  geom_abline(slope = 1, intercept = log10(r),
              linetype = 2, color = "darkgrey")
```

US gun murders

8. Add-on packages:

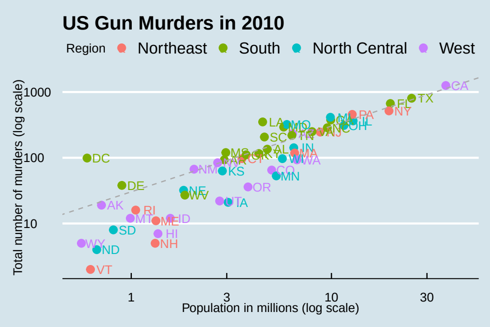
The power of `ggplot2` is augmented further due to the availability of add-on packages. In the remaining steps, we improve our plot using the `ggthemes` and `ggrepel` packages.

- `ggthemes` contains many popular themes such as `theme_economist` and `theme_fivethirtyeight`.
- `ggrepel` stands for repulsive textual annotations. It includes a geometry that adds labels while ensuring that they do not fall on top of each other.

```r
# install.packages(c("ggthemes", "ggrepel"))
library(ggthemes); library(ggrepel)
```

Gallery of themes: https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

```r
ggplot(data = df, aes(x = population/1e6, y = total, col = region)) +
  geom_point(size = 3) +
  geom_text(aes(label = abb), nudge_x = 0.07) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)", col = "Region") +
  geom_abline(slope = 1, intercept = log10(r),
              linetype = 2, color = "darkgrey") +
  theme_economist()
```



US gun murders

Final touch:

- Draw the line first so it doesn't go over the points.
- Replace `geom_text()` with `geom_text_repel()`.

```
ggplot(data = df, aes(x = population/1e6, y = total, col = region)) +
  geom_abline(slope = 1, intercept = log10(r),
              linetype = 2, color = "darkgrey") +
  geom_point(size = 3) +
  geom_text_repel(aes(label = abb)) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "US Gun Murders in 2010",
       x = "Population in millions (log scale)",
       y = "Total number of murders (log scale)", col = "Region") +
  theme_economist()
```

US gun murders