Introduction to ggplot2

A Layered Grammar of Graphics

Vik Gopal

What is to be sought in designs for the display of information is the clear portrayal of complexity.

About ggplot2

- ggplot2 implements the grammar of graphics, which is an abstract method of building up graphics from components.
- The quality of graphs produced by this package is very high.
- It is excellent for exploring data in many ways.
- Bear in mind, that it is not the only method for making graphs in R.
  - The lattice package is an excellent alternative, based on the paradigm of conditioned plots.

Load Libraries

- ggplot2 is part of the tidyverse collection of packages:

```
library(tidyverse)
```

- You should see that the ggplot2 package has been loaded.
- Let us make a first plot using this package before we go any further.

The mpg Data Frame

- The mpg data frame contains observations collected by the EPA on 38 models of cars.

```
mpg
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans      drv     cty
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18
 2 audi         a4           1.8  1999     4 manual(m5) f        21
 3 audi         a4           2    2008     4 manual(m6) f        20
...
```

- For now, we shall work with just two of the variables:
  1. `displ`, which corresponds to the car's engine size in litres.
  2. `hwy`, which corresponds to the fuel efficiency of the car on a highway.

The mpg Data Frame

ggplot2 code

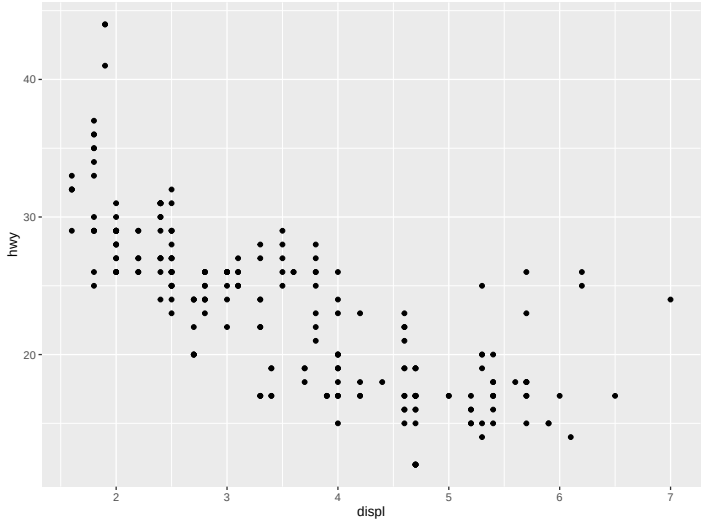
- The following code will create a plot, putting `displ` on the x-axis and `hwy` on the y-axis.

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

- We say that `displ` is mapped to values on the x-axis, and `hwy` is mapped to values on the y-axis.

The mpg Data Frame

plot



Breaking Down The Syntax

- The `ggplot2()` function creates a coordinate system that we can add layers to.
  - The first argument is the dataset to use in the graph.
- To complete the graph, we have to add one or more layers.
  - `geom_point()` adds a layer of points to the plot, thus creating a scatterplot.
- `ggplot2` comes with a huge variety of `geom` functions. Each adds a different type of layer to the plot.
- The input to each `geom` function is a mapping argument. This defines how variables in our dataset are mapped to visual properties.
  - In this case, `displ` is mapped to the x-axis and `hwy` to the y-axis.

A Graphing Template

- The basic template for making graphs with `ggplot2` is

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

- The rest of this chapter shows how different choices for the values in angled-brackets result in different types of graphs.

Getting Help on ggplot2

- The help pages within R (for each function).
- The vignette, which can be accessed with the following command, within R.

```
vignette('ggplot2-specs')
```

- The website `http://ggplot2.tidyverse.org/`
- Chapters 3 and 28 from the course textbook `http://r4ds.had.co.nz/`
- The stackoverflow website. You will see several questions answered by Hadley Wickham himself here.

1. Aesthetics and Geoms
   - Point Geom
   - Histogram Geom
   - Line and Text Geoms
   - Bar Geom
   - Smooth Geom
   - Rug Geom
   - Boxplot Geom
   - Tiles and Hexagons
2. Miscellaneous Tasks
   - Arranging Several Plots
   - Themes & Extensions
3. Summary

Mapping Variables to Aesthetics

- An **aesthetic** is a visual property of the objects in your plot.
- Aesthetics include things like the size, shape or colour of the points plotted.
- The position of a point, i.e. its x- and y- coordinates, is another aesthetic.
- The type of line (solid, dashed, etc.), colour of the line and thickness of the line are aesthetics too.
- For each aesthetic, the *level* of the aesthetic can be changed to reflect different *values* in the data.
- By doing so, we can extend a graphic on a 2-D medium to include several variables.

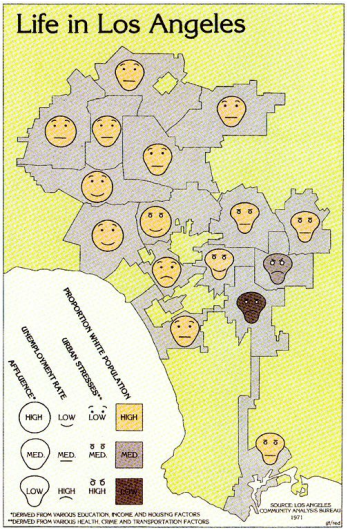
Geometrical Objects

- In `ggplot2`, 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.
  - Bar charts use the bar geom.
  - Line charts use the line geom.
  - Boxplot use the boxplot geom.
- Scatterplots use the point geom. We seldom refer to a scatterplot as a **point geom**, but this is the geometrical object that defines it.

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, colour, size and (x- and y-) coordinates are all aesthetics associated with the point geom.
- All of these aesthetics are also associated with the line geom. In addition, the line type and size are aesthetics associated with the line geom, but not with the point geom.
- The lower, middle and upper horizontal lines are aesthetics associated with the boxplot geom, but not with the point or line geom.
- Before we study some of these geoms and their associated aesthetics, let's take a look at an example.

Chernoff Faces



- The **fill** aesthetic is mapped to proportion of white people.
- The **smile** aesthetic is mapped to affluence.
- The **face shape** is mapped to unemployment.
- The **eyes** are mapped to stress.

Scatterplots

- The point geom is used to create scatter plots.
- The aesthetics associated with it are
  - x, y
  - alpha
  - colour
  - fill
  - group
  - shape
  - size
  - stroke

- The defining characteristic of a point is its position. Hence the x- and y- aesthetics are **required**. The rest are optional.
- Mapping variables to one or more of the other aesthetics allows us to present more than 2 variables in a scatterplot.

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 level of the aesthetic to each unique value of the variable.
  - This is a process known as **scaling**. We shall see how to edit this mapping ourselves later on.
- `ggplot2` will also add a legend that explains what levels of the aesthetic correspond to which values of the data.

Mapping Colour Aesthetic

`mpg` dataset

- Suppose that, in the `mpg` dataset, we wish to map the colours of the points to the `class` variable for each car.

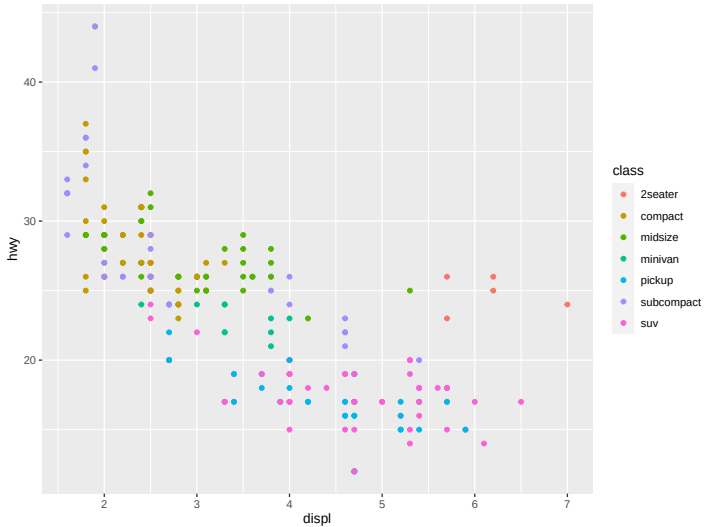
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y=hwy, color=class))
```

Remember:

- The mappings go *inside* the `aes( )` function, which goes inside the `mapping` argument.
- The mappings are of the form <aesthetic> = <dataset variable>.
- Since we are in the tidyverse, we do not need inverted commas to denote variable names, i.e. we do not need to put `x = "displ"`.

Mapping Colour Aesthetic

colour aesthetic to class



Mapping Size Instead of Colour

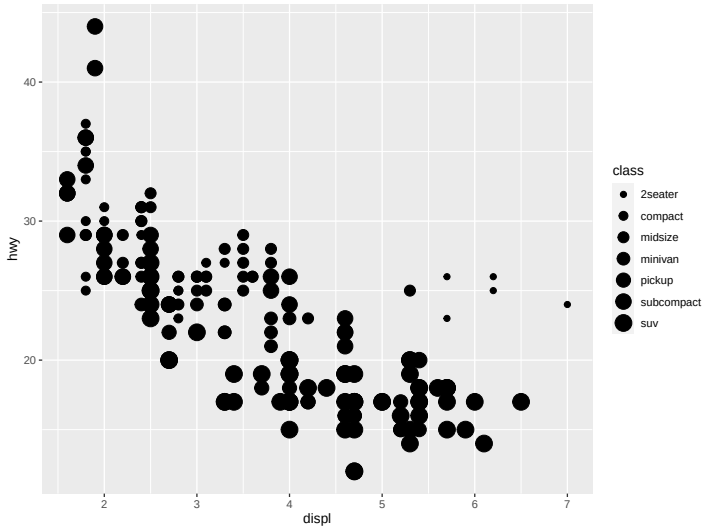
size aesthetic to class

- If we map the size of the points, instead of their colour, then we get what is sometimes referred to as a bubblechart.

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

Mapping Size Instead of Colour

cont'd



Problems

- `ggplot2` gives a warning in the previous call because the `class` variable is unordered.
- It is not clear how to map the size aesthetic, which is on a ratio scale, to class, and hence the warning from `ggplot2`.
- Another problem we see is that some of the circles are overlapping, and it is difficult to see the actual size of several circles.

Problems

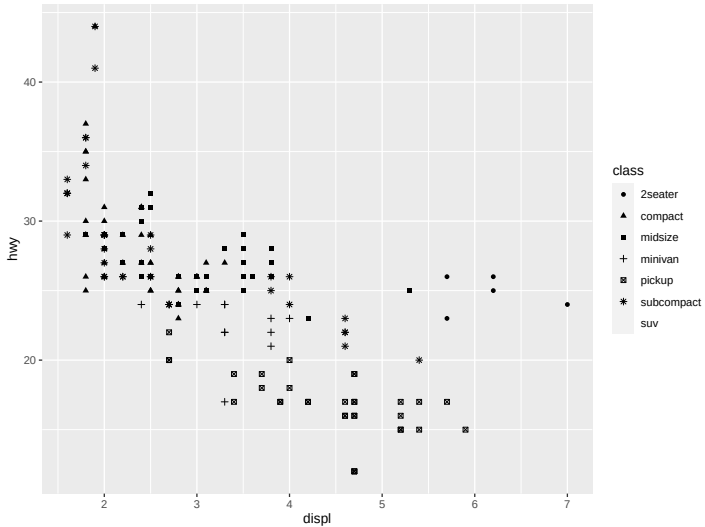
solution

- Instead of mapping size to class, we could use the shape aesthetic.
- Take a look at this command, and the corresponding output on the next page.

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

Mapping Shape Instead of Size

more problems



Mapping Shape Instead of Size

cont'd

- The earlier warning no longer appears, but we get a different warning, suggesting that we have too many categories for the `class` variable.
- We also observe that we have overplotting - several points are being plotted on top of each other.

Solutions to Problems

overplotting

- To solve the overplotting problem, we need to jitter the points, as we discussed in topic 01.
- In `ggplot2`, jittering the points is referred to as a position adjustment (for this geom). We shall see position adjustments for other geoms later.
- The position adjustment should be specified **outside** the `mapping` argument of the geom function.

Solutions to Problems

mapping to shapes

- To get rid of the warning, we must remember that the mapping from variable to aesthetic is done via a **scale**.
- We need to edit the scale that maps variables to shapes.
- In this case, the default scale that `ggplot2` uses only has 6 entries. We have to create one manually.
- The following code will solve the two problems we observed earlier.

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

Summary (So Far)

- Aesthetics expect to see a certain kind of variable. For instance, size (of symbol) does not expect to be mapped to a categorical variable.
- We may want to edit the mapping from variable to aesthetic. For instance, we may want the colour for a particular class of car to be dark red. Or we might want a particular shape for a particular class. To do so, we have to edit the scale for that aesthetic.
- Jittering is referred to as a position adjustment in `ggplot2`.

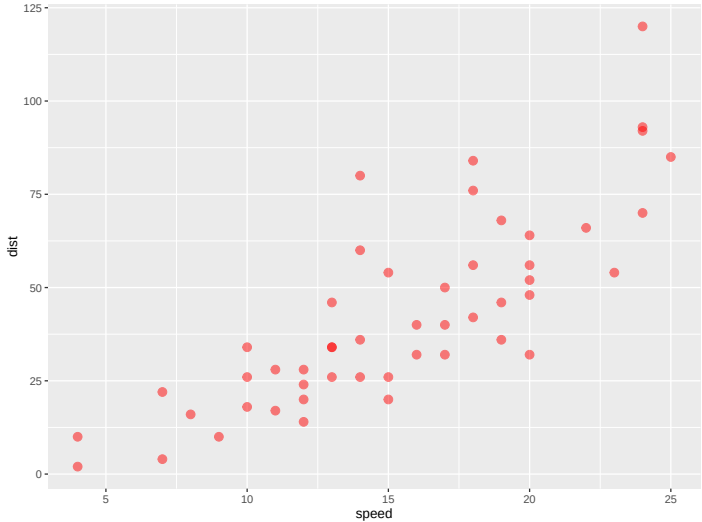
Braking Distance and Speed

- In slide 124 of topic 01, we created a scatterplot depicting the relationship between braking distance and speed.
- In order to re-create the picture, we need to add transparency to the points, and set their colour to be red.
- Both alpha and colour are aesthetics that could be mapped to variables, but here, we just want to fix their values to something other than the default.
- To do so, we specify these values **outside** the mapping argument. This fixes the level of an aesthetic.

```
ggplot(data= cars) +
  geom_point(mapping=aes(x=speed, y=dist), alpha=0.5,
             colour="red", size=3)
```

Braking Distance and Speed

plot



Fixing the Level of an Aesthetic

- When we fix the level, we must use a value that makes sense for that aesthetic.
- The following are valid specifications of common aesthetics:
  1. *Colour aesthetic:* The name of a colour as a character string, e.g "red" or "blue".
  2. *Size aesthetic:* The size of a point in mm.
  3. *Shape aesthetic:* The shape of a point as a number.
- For more details on what you can use to specify the level of an aesthetic, refer to the vignette on *Aesthetic specifications*.

Adding Labels

- The other difference with the original plot we made was that we added labels and a title to the plot.
- To do so with ggplot2, we need to specify the `labs()` layer.
- The basic arguments to `labs()` are:
  1. `title`
  2. `subtitle`
  3. `x`
  4. `y`

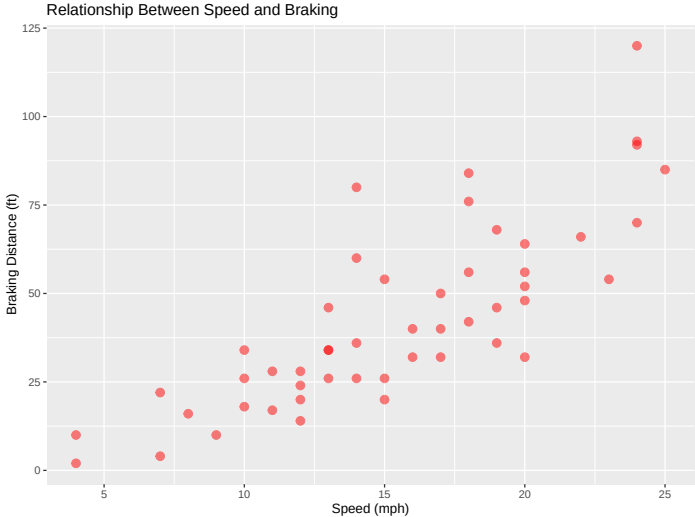
Braking Distance and Speed

adding labels

```
ggplot(data= cars) +
  geom_point(mapping=aes(x=speed, y=dist), alpha=0.5,
             colour="red", size=3) +
  labs(title="Relationship Between Speed and Braking",
       x = "Speed (mph)", y="Braking Distance (ft)")
```

Braking Distance and Speed

plot with labels



Histograms

- The histogram allows us to visualise 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.
- There are two related geoms here:
  1. `geom_histogram` will display the counts in each bin with bars.
  2. `geom_freqpoly` will display the counts with lines. This is more appropriate when we wish to compare the distribution of a variable conditioned on a categorical one. For instance, we may want to compare income distribution for males and females.

Aesthetics

Some of the aesthetics for this geom are:

- x- coordinate
- alpha
- colour
- fill

- They are similar to the aesthetics for a bar geom, since both use rectangles to represent the summaries of the data.

Geom Arguments

- Apart from the aesthetics, we also have to consider other issues when using this geom.
- These include:
  - The widths of the bins used,
  - The number of bins,
  - The location of the bins.

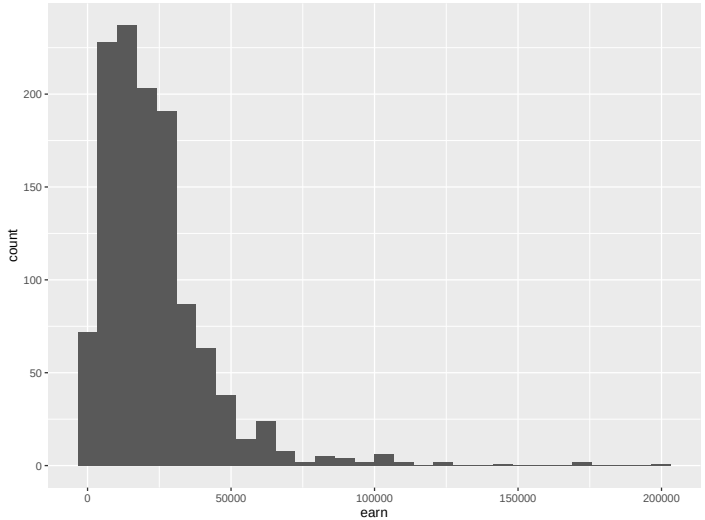
Income and Heights

- Let us revisit the dataset we encountered in topic 02, and the histogram we made on slide 18 there.
- Here is the code for a first attempt:

```
heights <- read.csv("../data/heights.csv")
ggplot(heights) +
  geom_histogram(aes(x=earn))
```

Income and Heights

a first histogram



Income and Heights

cont'd

- We get a warning, telling us that 30 bins have been used, and that we should re-consider.
- Let us work with bin widths of 10,000 as before.
- Notice that the left-most rectangle is centred at 0. That is not what we want as there are no negative incomes. We want the lower limit of the left-most bin to be 0.
- Let us also add the colours that we used last time. These are aesthetics, but we wish to fix their values.
- Finally, we shall add the informative labels.

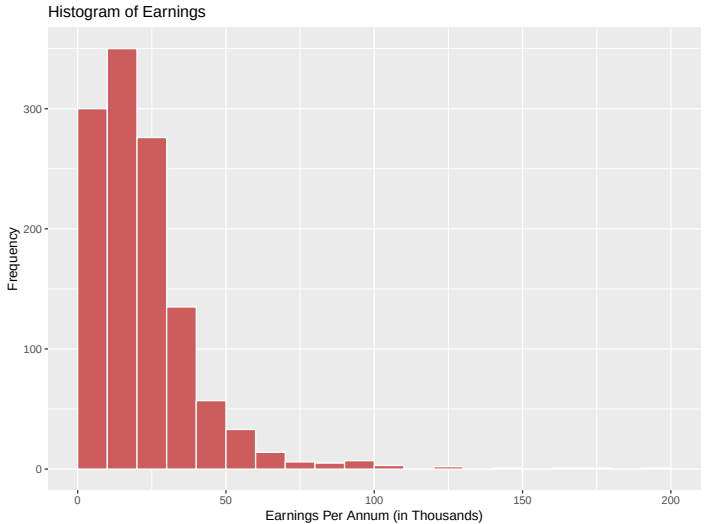
Income and Heights

code

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3), binwidth = 10, boundary=0,
                 colour="white", fill="indianred") +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Frequency")
```

Income and Heights

a second attempt



Income and Heights

density not count

- The white outlines are interfering with the grid lines. We shall drop them henceforth.
- In topic 02, we used the density for each bin instead of the counts.
- This made the histogram closer in spirit to a pdf, since the areas would sum to 1.
- In order to do so here, we must know that the `geom_hist()` computes certain summaries of the data. Among them are:
  - count,
  - density,
- We need to tell `ggplot2` to use density instead of count on the y-axis aesthetic.

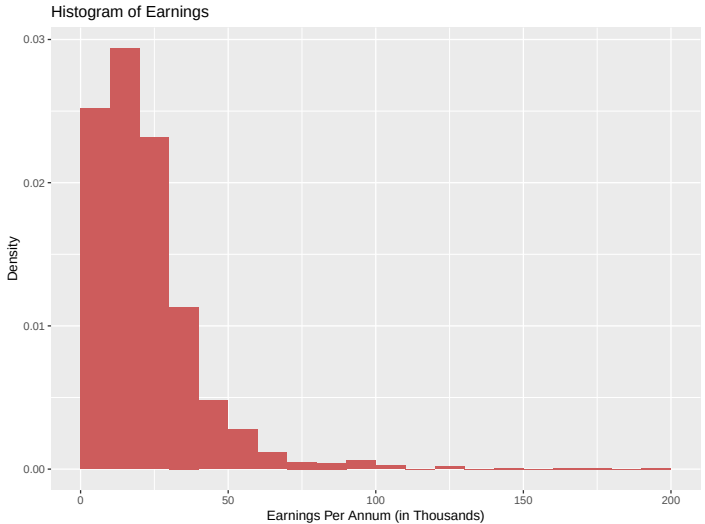
Income and Heights

code

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3, y=after_stat(density)),
                 binwidth = 10, boundary=0, fill="indianred") +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Density")
```

Income and Heights

a third attempt



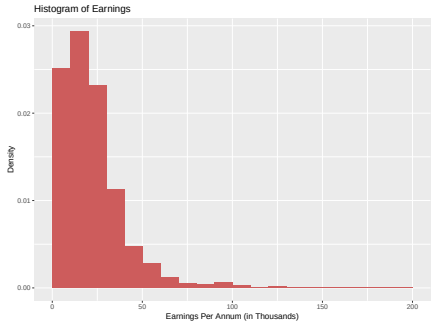
Zooming In

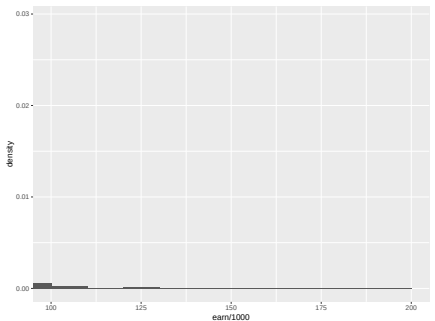
- Sometimes, we wish to zoom in on a particular section of the histogram.
- The layer to add is `coord_cartesian()`.
- Imagine using a magnifying glass to focus on a particular section of the graph; that is what this function does.
- It is easy to use `xlim()` layer instead. However, for the histogram geom, `ggplot` will display a conditional distribution instead.

Income Data

zooming in

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3, y=after_stat(density)),
                 binwidth=10, boundary=0,
                 show.legend = FALSE) +
  coord_cartesian(xlim=c(100, 200))
```



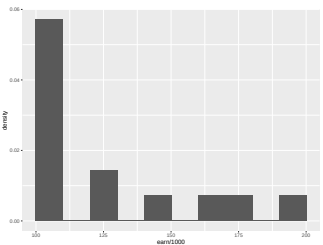


Income Data

conditional distribution

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3, y=after_stat(density)),
                 binwidth=10, boundary=0,
                 show.legend = FALSE) +
  xlim(c(100, 200))
```

```
Warning: Removed 1178 rows containing non-finite values (stat_bin).
```



- This is a new distribution.
- The areas sum to one.

Income and Heights

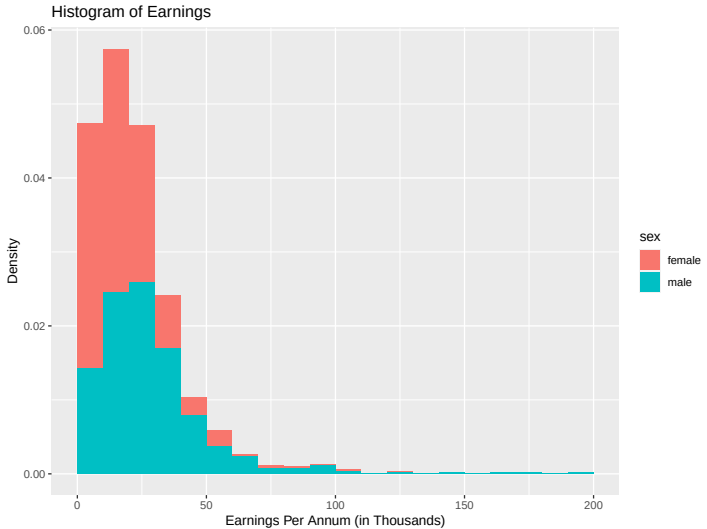
consider gender

- Earlier, we realised that there was a stark difference between males and females in terms of income earned.
- How can we present this information?
- We could consider mapping an aesthetic to the sex variable.

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3, y=after_stat(density), fill=sex),
                 binwidth = 10, boundary=0) +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Density")
```

Income and Heights

plot



Income and Heights

improvements

- The bars for females have been stacked *on top* (notice the y-axis values) of the bars for the males.
- We need a position adjustment of the bars in order to compare them better.
- There are only a few adjustments possible. In this case we should go for the `dodge` adjustment.
- While we are at it, we shall try to change the labels on the legend.

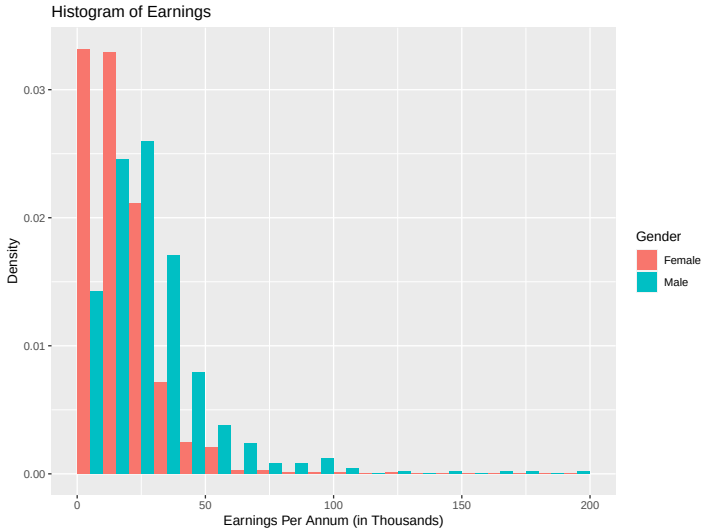
Income and Heights

code

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3, y=after_stat(density), fill=sex),
                 binwidth = 10,
                 boundary=0, position="dodge") +
  scale_fill_discrete(labels=c("Female", "Male"),
                      name="Gender") +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Density")
```

Income and Heights

plot



Income and Heights

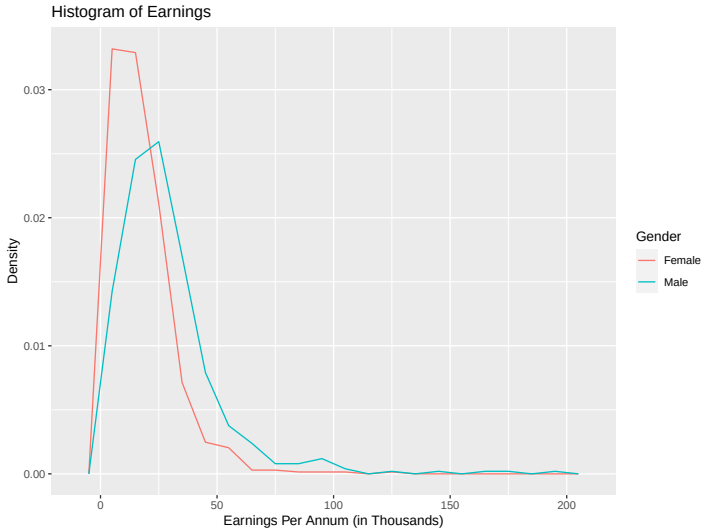
frequency polygons

- In fact, if we wish to compare two distributions, conditioned on a categorical variable, we would be better off using a frequency polygon.
- Since it does not use bars, we do not have a fill aesthetic any more.

```
ggplot(heights) +
  geom_freqpoly(aes(x=earn/1e3, y=after_stat(density), colour=sex),
                binwidth = 10,
                boundary=0) +
  scale_colour_discrete(labels=c("Female", "Male"),
                        name="Gender") +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Density")
```

Income and Heights

plot



Line Geom

- This geom connects observations in the order of the variable on the x-axis.
- It is suitable for plotting time series.
- It is **not** what you use to add a reference line to a scatterplot.
- The aesthetics that the line geom uses are
  - x- and y- coordinates
  - alpha,
  - colour

UNESCAP Population Data

- Let us return to the UN dataset from topic 02.
- Here, we shall try to plot the time series of population for females, aged between 0 and 4 years, for all South-East Asian countries.
- The first step is to process this data frame. We need
  - The years in one column, not six separate ones.
  - Only the rows for the SEA countries.
  - Only the rows for females aged 0 – 4 years of age.

UNESCAP Population Data

process data

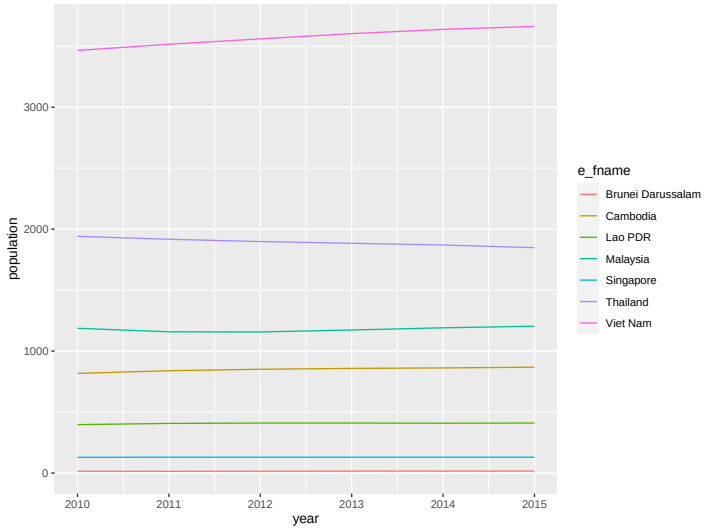
```
all_data <- readRDS("../data/un_data.rds")

pop1 <-  pivot_longer(all_data, Y2010:Y2015, names_to="year",
                      values_to="population", names_prefix = "Y",
                      names_transform = list(year = as.integer)) %>%
  filter(gender == "Female",
         age_group == "0-4 years",
         e_fname %in% c("Singapore", "Malaysia",
                        "Cambodia", "Thailand",
                        "Viet Nam", "Lao PDR",
                        "Brunei Darussalam"))

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

UNESCAP Population Data

plot



UNESCAP Population Data

cont'd

- The colours are not helpful. We have to peer very closely to distinguish them and match the lines to the colours.
- Instead, we shall label each line with a text label, that is the name of the corresponding country.
- For time series, we prefer a plot that is wider rather than narrow.
- Thus, let us try to do away with the legend.

Text Geom

- The text geom adds text directly to the plot.
- The aesthetics that it uses are
  - the x- and y- coordinates, and
  - a label.
- There are also additional arguments that allow us a very low-level control over the position, alignment, and size of the labels that are printed.

UNESCAP Population Data

considerations for adding text

- Notice that the plot on slide 61 has no space on the right or left of the lines.
- If we wish to add text at the end of each line, we have to make space for it by extending the limits of the graph.
  - This can be done with the `xlim()` layer.
- We need to add the text near the last point on each line, but the `pop1` dataset contains many more points than we need. How to proceed?
- The name "Brunei Darussalam" is too long for our purpose. We shall shorten it to "Brunei" using the `recode()` function from the tidyverse.

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

UNESCAP Population Data

more considerations for adding text

- We need to plot a separate line for each country, but with the same colour for all of them.
- How do we do this?
- We use the `group` argument to `aes()`. This informs R to create separate geoms for each group. As long we do not map colour to anything, they will all have the same line.
- *Try out the code on slide 66 without the group argument.*
- Notice how we have added three geoms onto one graph on the next slide.
- Also notice how we override the data used for the last geom with the data we need to make the plot.

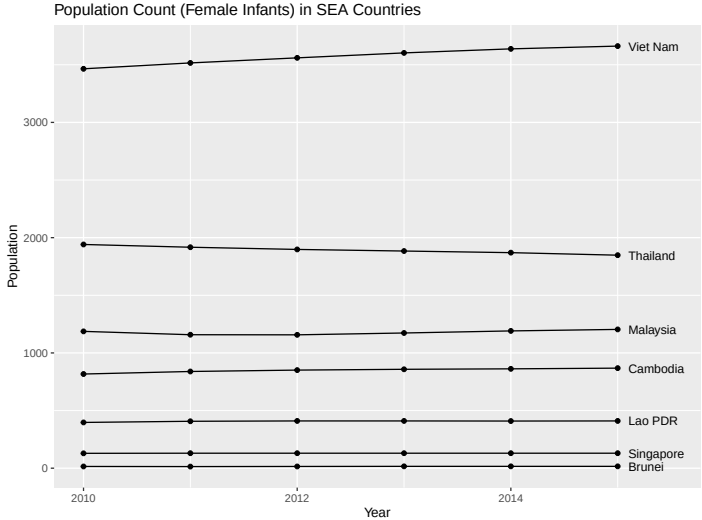
UNESCAP Population Data

plotting code

```
ggplot(pop1) +
  geom_line(aes(x=year, y=population, group=e_fname)) +
  geom_point(aes(x=year, y=population, group=e_fname)) +
  xlim(2010, 2015.5) +
  geom_text(data=pop2,
            mapping = aes(x=year, y=population,
                          label=fname),
            hjust="left", nudge_x=0.1, size=3.5)
```

UNESCAP Population Data

plot



Adding Reference Lines

- To add a reference line to the graph, we must use one of
  - `geom_vline` for vertical lines,
  - `geom_hline` for horizontal lines, or
  - `geom_abline` for straight lines defined by a slope and an intercept.

Adding Reference Lines

not ordinary geoms

- These three are not ordinary geoms in the sense that variables cannot be mapped to their aesthetics.
- Behind the scenes, `ggplot2` in fact generates a data frame and maps the x- and y- variables from this data frame onto the line aesthetic.
- Suppose that, in the income distribution histogram, we wish to indicate high-income earners (more than 100,000 per annum) with a dashed vertical line.

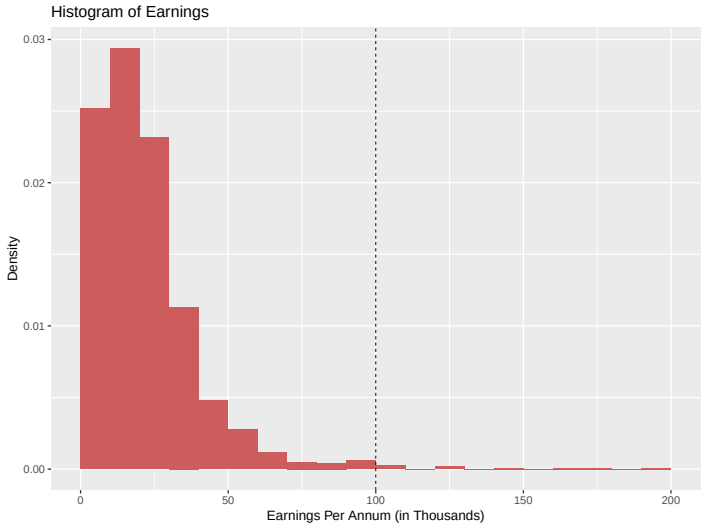
Income and Heights

adding a vertical reference line

```
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3), binwidth = 10,
                 boundary=0,
                 colour="white", fill="indianred") +
  geom_vline(aes(xintercept=100), lty=2, size=0.3) +
  labs(title="Histogram of Earnings",
       x="Earnings Per Annum (in Thousands)", y="Frequency")
```

Income and Heights

plot



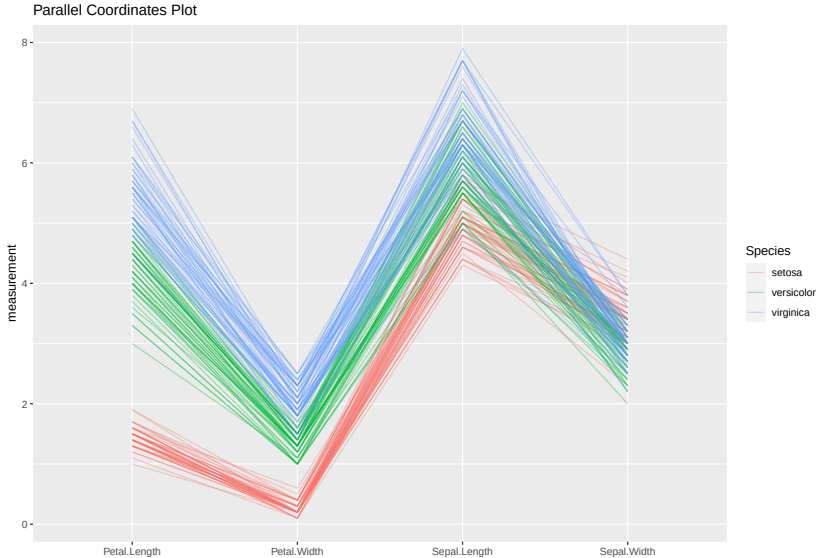
Parallel Coordinates Plot

- This particular type of plot uses the line geom.
- It allows you to compare the relationship between variables, and to compare that relationship between groups.

```
iris %>% mutate(id = 1:nrow(iris)) %>%
  group_by(id) %>%
  pivot_longer(Sepal.Length:Petal.Width, names_to="attribute",
               values_to="measurement") %>%
  ggplot() +
  geom_line(aes(x=attribute, y=measurement,
                group=id, colour=Species),
            alpha=0.35) +
  labs(title="Parallel Coordinates Plot")
```

Parallel Coordinates Plot

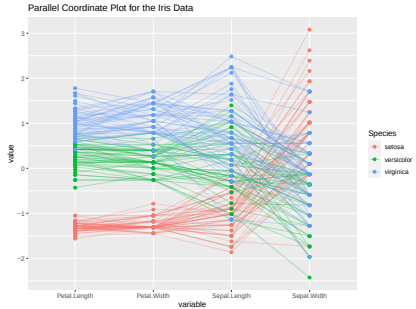
cont'd



From GGally Package

Here's a version of the same type of graph, produced by the GGally package:

```
library(GGally)
ggparcoord(data = iris, columns = 1:4, groupColumn = 5,
           order = "anyClass",
           showPoints = TRUE,
           title = "Parallel Coordinate Plot for the Iris Data",
           alphaLines = 0.3)
```



Aesthetics

- There are two types of bar charts:
  - `geom_col` makes the height of the bar represent values in the data.
  - `geom_bar` makes the height of the bar proportional to the number of cases in each group.
- Some of the aesthetics that these geoms understand are
  - x- and y- coordinates,
  - alpha,
  - colour,
  - fill,
  - linetype and
  - size

Budget Bar chart

- Recall the budget bar chart that we created on slide 128 of topic 01, to depict operating budget categories.
- This requires the `geom_col` object, not the `geom_bar` object.
- The data frame was created with the following commands:

```
exp_cat <- str_to_title(c("manpower",
                          "asset",
                          "other"))
amount <- c(519.4, 38, 141.4)
op_budget <- data.frame(amount, exp_cat)
op_budget
```

```
  amount  exp_cat
1  519.4 Manpower
2   38.0    Asset
3  141.4    Other
```

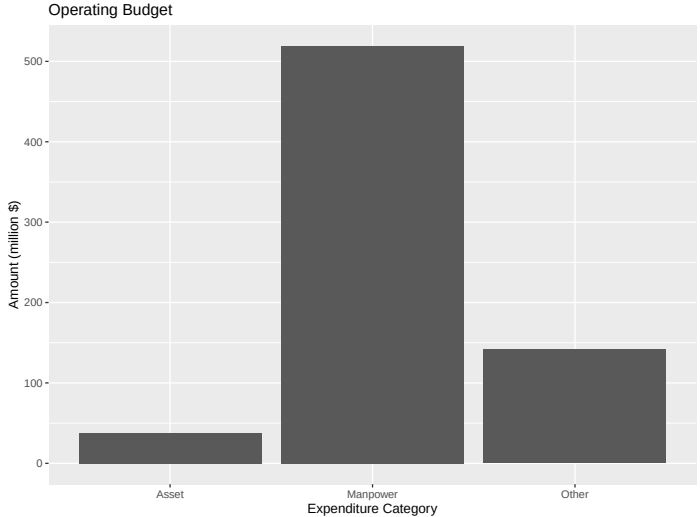
Budget Bar Chart

cont'd

```
ggplot(op_budget) +
  geom_col(mapping=aes(x=exp_cat, y=amount)) +
  labs(title="Operating Budget", y="Amount (million $)",
       x="Expenditure Category")
```

Budget Bar Chart

first plot



Budget Bar Chart

- Notice how the expenditure categories have been arranged.
- The category (`exp_cat`) was stored as a factor. When plotted, the categories were arranged according to alphabetical order.
- In bar charts, we typically arrange them in order of tallest to shortest.
- To do this, we need to `reorder()` the levels of the factor.

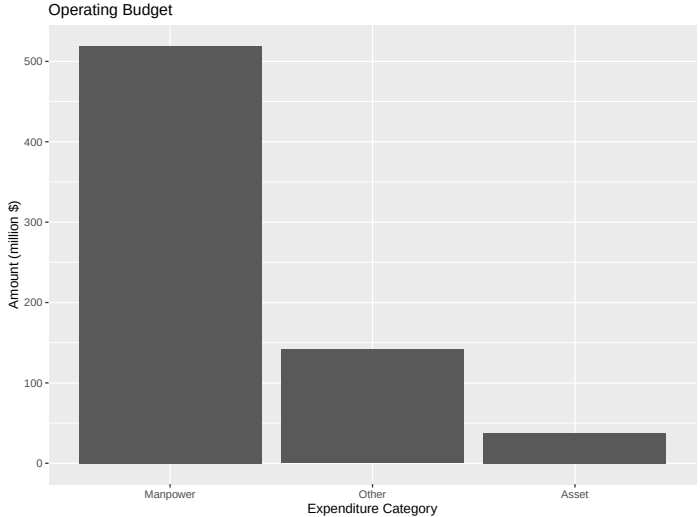
Budget Bar Chart

cont'd

```
op_budget <- mutate(op_budget,
                    exp_cat= reorder(exp_cat, -amount))
ggplot(op_budget) +
  geom_col(mapping=aes(x=exp_cat, y=amount)) +
  labs(title="Operating Budget", y="Amount (million $)",
       x="Expenditure Category")
```

Budget Bar Chart

second plot



IMDA Data

- The IMDA data contains several variables:
  - age group,
  - year,
  - activity,
  - percentage of group who have ever participated in that activity.
- With ggplot, we can easily plot different views of the data.
- Let us first recreate the plot from slide 81 of topic 02.

```
imda <- readRDS("../data/imda.rds")
young_adults <- filter(imda, age == "20-29",
                       year==2015) %>%
  mutate(pct = as.numeric(ever_used),
         media_activity= reorder(media_activity, pct))
```

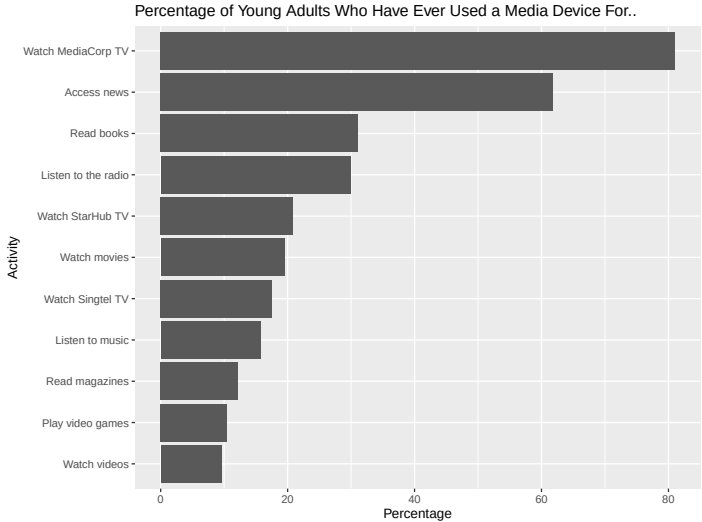
IMDA Bar Chart

cont'd

```
ggplot(young_adults) +
  geom_col(mapping=aes(x=media_activity, y=pct)) +
  coord_flip() +
  labs(title="Percentage of Young Adults ...", y="Percentage",
       x="Activity")
```

IMDA Bar Chart

plot



IMDA Bar Chart

cont'd

- Now suppose we focus only on the following three categories:
  - Watch Mediacorp TV
  - Watch StarHub TV
  - Watch Singtel TV
- We wish to study how the proportions of people who have ever used a media device for these purposes varies with
  - age group, and
  - year.
- How can we display these values?

IMDA Bar Chart

data preparation

```
tv <- filter(imda,
        media_activity %in% c("Watch MediaCorp TV",
                              "Watch StarHub TV",
                              "Watch Singtel TV")) %>%
  mutate(pct=as.numeric(ever_used),
         age = reorder(as.factor(age),
                       as.numeric(str_sub(age, 1, 2))))
```

Facets

- Instead of mapping the 4 variables to 4 different aesthetics, we can use facets for one of them.
- The faceting variable is typically a categorical one.
- Using a facet splits your data up into subsets according to levels of the factor, and one sub-plot is created for each level of the factor.

Facets

cont'd

- To facet by a single variable, we use the `facet_wrap()` layer call.
- The argument to it should be in the form of an R formula.
- These plots are what Tufte referred to as **small multiples**.
- They allow you to make comparisons across and within the sub-plots easily.

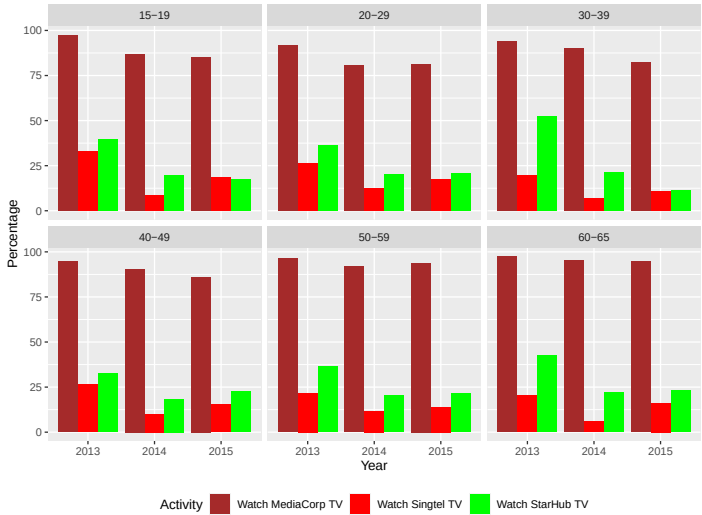
IMDA Bar Chart

with facets

```
ggplot(tv) +
  geom_col(mapping=aes(x=year, y=pct,
                       fill=media_activity),
           position = "dodge") +
  facet_wrap( ~ age) + labs(x="Year", y="Percentage") +
  scale_fill_manual(values=c("brown", "red", "green"),
                    name="Activity") +
  theme(legend.position = "bottom")
```

IMDA Bar Chart

facet plot



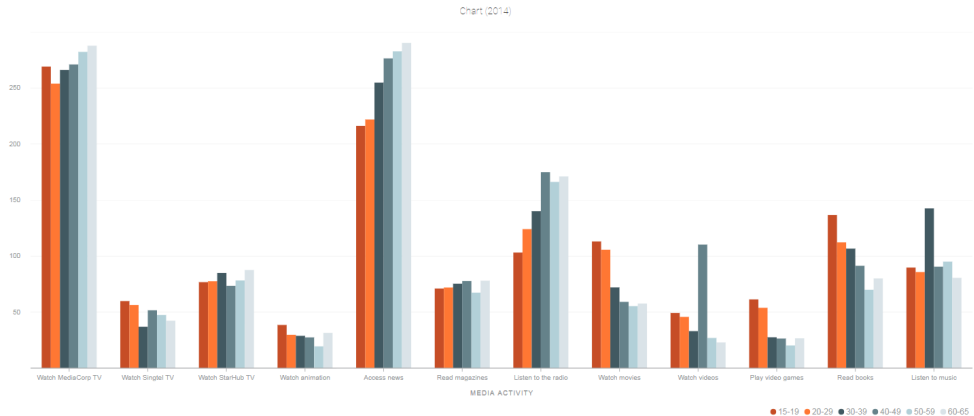
IMDA Bar Chart

thoughts

- What do we observe from the data?
- How else could we have chosen to represent the data?
- Which is the correct choice?
- What other geoms could we have used for this set of variables?
- Is it an improvement over this?

IMDA Bar Chart

from data.gov.sg



Income Data

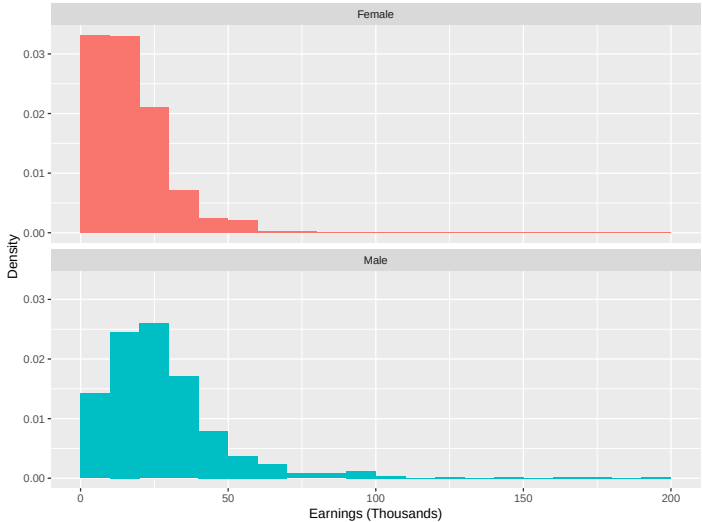
- Earlier, we created histograms for the income data, and considered a couple of options for visualising histograms of income conditioned on another variable by carrying out position adjustment.
- We could just as easily have faceted the display.

Income Data

```
heights <- read.csv("../data/heights.csv")
ggplot(heights) +
  geom_histogram(aes(x=earn/1e3,  y=after_stat(density),
                     fill=sex), binwidth=10,
                 boundary=0,
                 show.legend = FALSE) +
  facet_wrap(~ sex, nrow=2,
             labeller=as_labeller(c(`female`="Female",
                                    `male`="Male"))) +
  labs(x="Earnings (Thousands)", y="Density")
```

Income Data

facet plot



Geom Smooth

- The geometric object represented by a smoother allows the eye to see patterns in the data by drawing a **line** through the points. It allows us to depict
  - trends in time series data
  - (non-linear) relationships between variables.
- There are several different types of smoothers. Different smoothers use different criteria to fit different lines of best fit.
- We shall study just a couple of them in brief detail:
  - linear regression models
  - loess smoother

Aesthetics

- Some of the aesthetics that this geom understands are
  - x- and y- coordinates
  - alpha
  - colour
  - fill
  - linetype

mpg Smooth

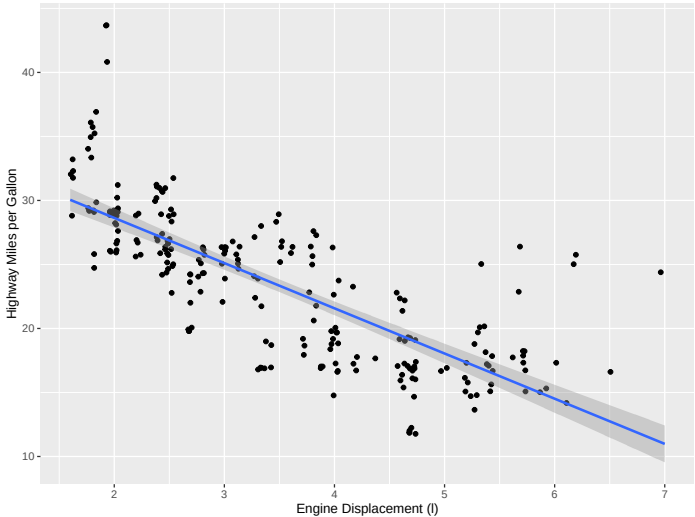
- Recall the scatterplot we made to understand the relationship between `displ` and `hwy`.
- Let us now add a smooth linear regression line to the data.
- In the code below, notice how the variables are mapped only once, within the `ggplot()` call.
- The subsequent geoms (there are two of them) inherited that mapping.

mpg Smooth

```
ggplot(data=mpg,
       mapping=aes(x=displ, y=hwy)) +
  geom_point(position="jitter") +
  geom_smooth(method="lm") +
  labs(x="Engine Displacement (l)", y="Highway Miles per Gallon")
```

mpg Smooth

plot



Linear Regression Smoother

- Using `lm` invokes a simple linear regression in this case.
- The blue line is the line of best fit.
- The gray regions represent 95% confidence intervals for the mean.
- The line does not appear to be suitable for this data, which has some non-linearity.
- To represent or allow for this non-linearity, we have a couple of options:
  1. Use a higher order polynomial term in the linear regression, or
  2. Use a loess smoother. A loess smoother is a locally weighted regression smoother.

Loess Fitting

- Suppose x_i and y_i are measurements of an independent and dependent variable respectively.
- The loess regression curve, $\hat{g}(x)$, is a smoothing of y given x that can be computed for any value of x.
- To compute $\hat{g}(x)$,
  - First choose a value q, that will serve as the span.
  - The q values of x_i that are closest to x will be given a weight, based on how far they are from x, typically through a kernel function.
  - x_i values that are closer to x will receive a larger weight.
  - Perform a weighted least squares regression using the above weights.
- The `loess` smoother is the default smoother used by `geom_smooth`.

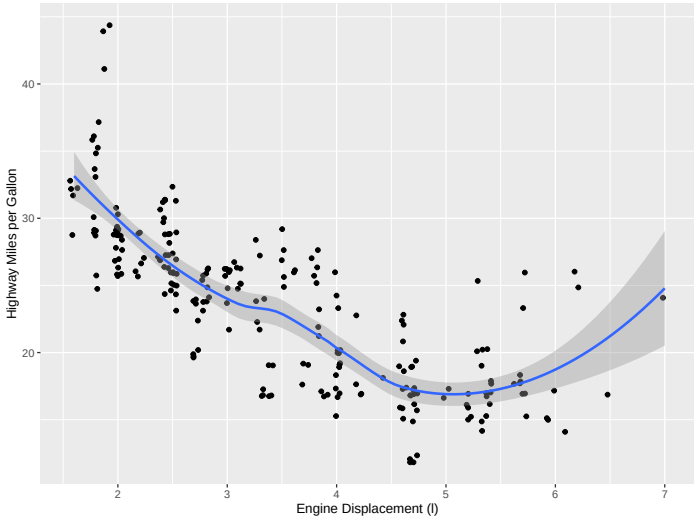
mpg Smooth

cont'd

```
ggplot(data=mpg,
       mapping=aes(x=displ, y=hwy)) +
  geom_point(position="jitter") + geom_smooth(span=0.6) +
  labs(x="Engine Displacement (l)", y="Highway Miles per Gallon")
```

mpg Smooth

loess smoother



mpg Smooth

cont'd

- The new curve reflects the presence of several cars that have large engines, but whose mileage is also good.
- Now suppose that we wish to study how this relationship varies with the drive type:
  - Front-wheel drive,
  - Rear-wheel drive, or
  - Four-wheel drive.
- We can do so by mapping this variable to the line-type aesthetic.

mpg Smooth

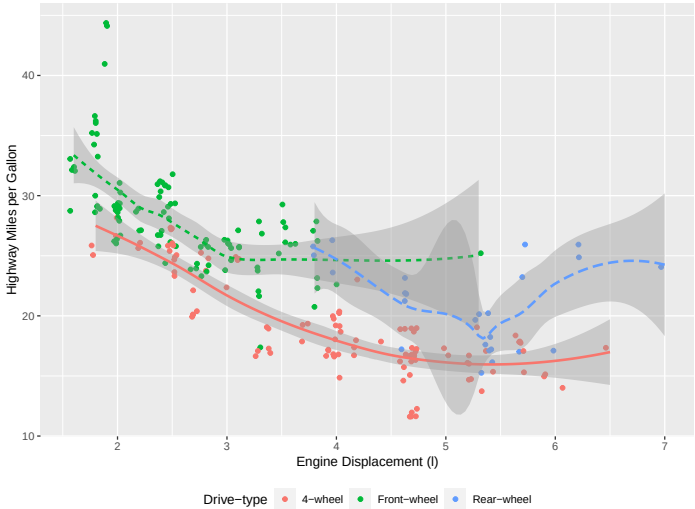
drive type to line type

```
ggplot(data=mpg, mapping=aes(x=displ, y=hwy, col=drv)) +
  geom_point(position="jitter") +
  geom_smooth(mapping=aes(linetype=drv),
              show.legend=FALSE) +
  labs(x="Engine Displacement (l)", y="Highway Miles per Gallon") +
  scale_colour_discrete(name="Drive-type",
    labels=c("4-wheel", "Front-wheel", "Rear-wheel"))+
  theme(legend.position = "bottom")
```

```
`geom_smooth()` using method = 'loess'
```

mpg Smooth

plot



Other Smoothers

- The loess smoother is computationally intensive, so when the number of datapoints is large (more than 1000), ggplot will use a generalised additive model, which uses a set of basis functions instead.
- Other smoothers can be used to model binary data (logistic regression) and in general, GLMs.
- We won't go further into these; once you take a class on them, you will be able to use them with confidence.

Rug Plots

- A rug plot is a compact visualisation designed to supplement a 2D display with marginal distributions.
- The aesthetics that it understands are
  - alpha
  - colour,
  - group,
  - linetype,
  - size.

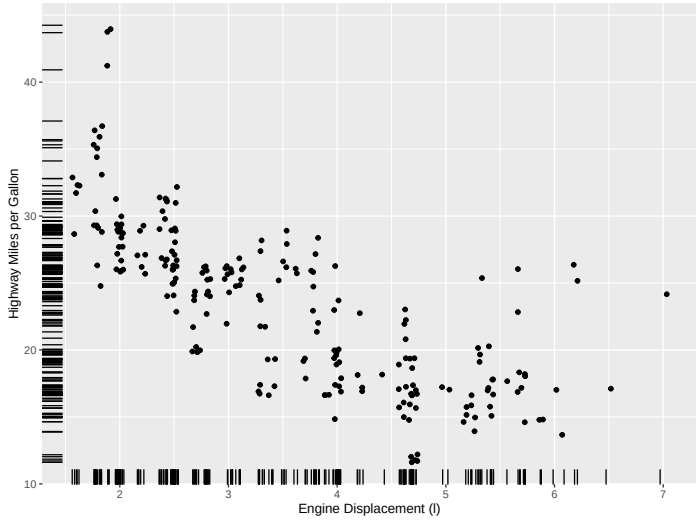
mpg Smooth

rug plot code

```
ggplot(mpg, mapping=aes(x=displ, y=hwy)) +
  geom_point(position="jitter") +
  geom_rug(position="jitter") +
  labs(x="Engine Displacement (l)", y="Highway Miles per Gallon")
```

mpg Smooth

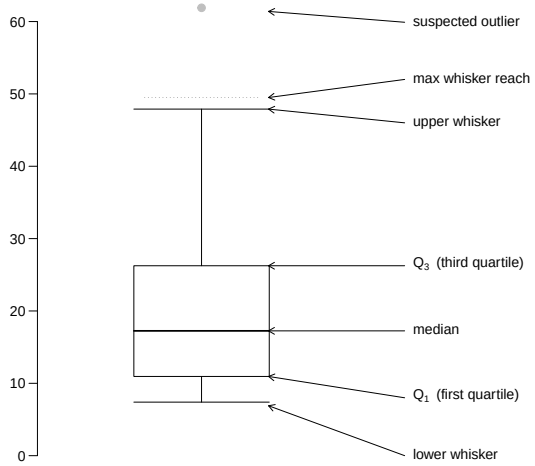
plot



Boxplots

- Boxplots are a visual representation of 3 of the five numbers in the five-number summary.
- They identify the median, lower and upper quartiles, and suggest which points could be *outliers*.
- An outlier is an observation that is very different from the majority of the data. An observation is defined to be an outlier if it falls more than $1.5 \times IQR$ below the lower quartile or more than $1.5 \times IQR$ above the upper quartile.

Boxplot Description



The distance from the "max whisker reach" to the third quartile is exactly $1.5 \times$ IQR.

Boxplots Versus Histograms

- A boxplot does not portray certain features of a distribution, such as distinct mounds and possible gaps in the data.
- If a distribution is indeed unimodal, then a boxplot does give an indication about the skew of a distribution.
- Boxplots are useful for identifying potential outliers, and for comparing groups with respect to their "center" and "spread".

Aesthetics

- The essential aesthetics that it understands are
  - x- value
  - lower,
  - upper,
  - middle,
  - ymin, and
  - ymax
- Other aesthetics include
  - colour,
  - fill,
  - linetype

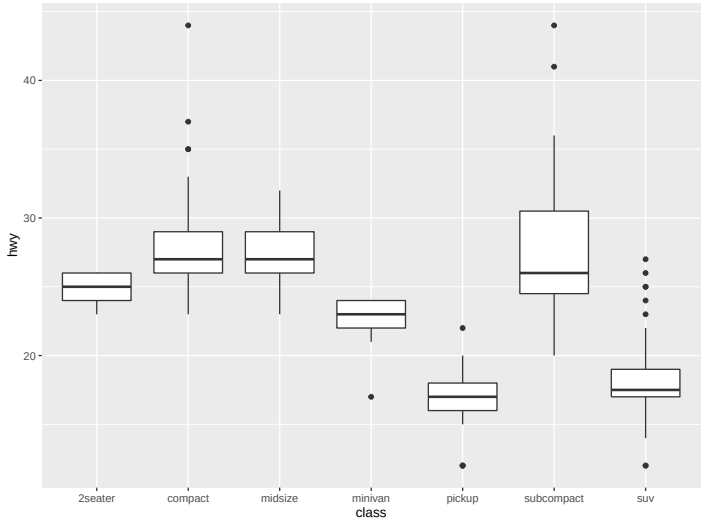
Series of Boxplots

- A boxplot summarises a variable well.
- Hence, when comparing the same variable between populations, it is useful to create a boxplot for each one, and put them side by side.
- It allows a comparison of the centres and spreads of the distributions.
- Let us return to the `mpg` dataset, and consider how `hwy` varies with `class` of the vehicle.

```
ggplot(data=mpg, mapping = aes(x = class, y = hwy))+
  geom_boxplot()
```

mpg Boxplot

plot



mpg Boxplot

- `class` is an unordered factor. Hence the levels are ordered alphabetically when plotting.
- We might want to order them according to the median in each individual boxplot.
- We can use the `reorder` function to do this.

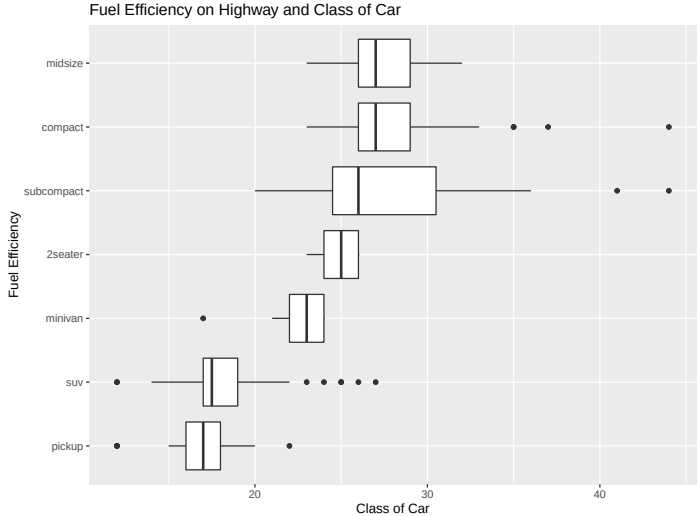
mpg Boxplot

reordered

```
mpg2 <- mutate(mpg, class2 =
         reorder(class, hwy, FUN=median))
ggplot(data=mpg2) +
  geom_boxplot(mapping=aes(x=class2, y=hwy)) +
  coord_flip() +
  labs(title="Fuel Efficiency on Highway and Class of Car",
       x="Fuel Efficiency", y="Class of Car")
```

mpg Boxplot

cont'd



Inter-quartile Range in 2-D

- Sometimes, wish to study the region in which 2 variables exists, sort of a generalisation of boxplots to 2D.
- A simple way is to bin the x-variable, and compute the upper and lower quartiles for the y-variable.
- Tukey later sharpened this idea with the bagplot, which allows one to identify outliers in 2-D space.

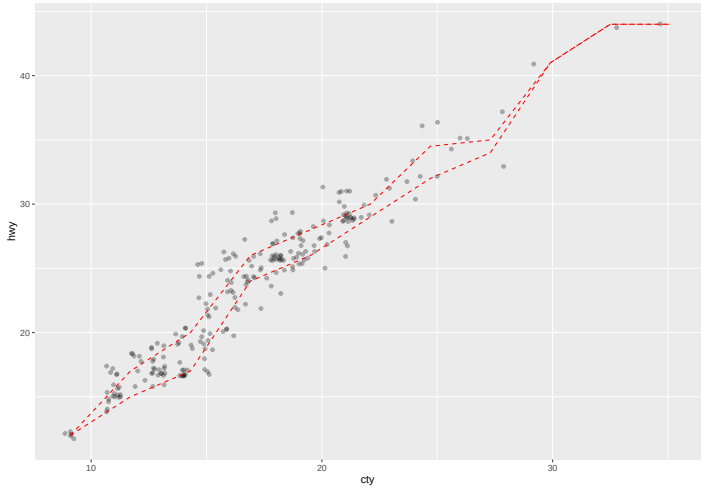
Inter-quartile Range in 2-D

cont'd

```
ggplot(mpg, mapping=aes(x=cty, y=hwy)) +
  geom_point(alpha=0.3, position="jitter") +
  stat_summary_bin(fun="quantile",
    fun.args=list(probs=0.25), colour="red",
    geom="line", bins=10, lty=2) +
  stat_summary_bin(fun="quantile",
    fun.args=list(probs=0.75), colour="red",
    geom="line", bins=10, lty=2)
```

Inter-quartile Range in 2-D

plot



Inter-quartile Range in 2-D

questions

With this plot, we can consider questions such as:

- How does spread of the y-variable change with the x-variable?
- Where (in $\mathbb{R}^2$) do most of the points exist?
- What is the general trend, robust to outliers?

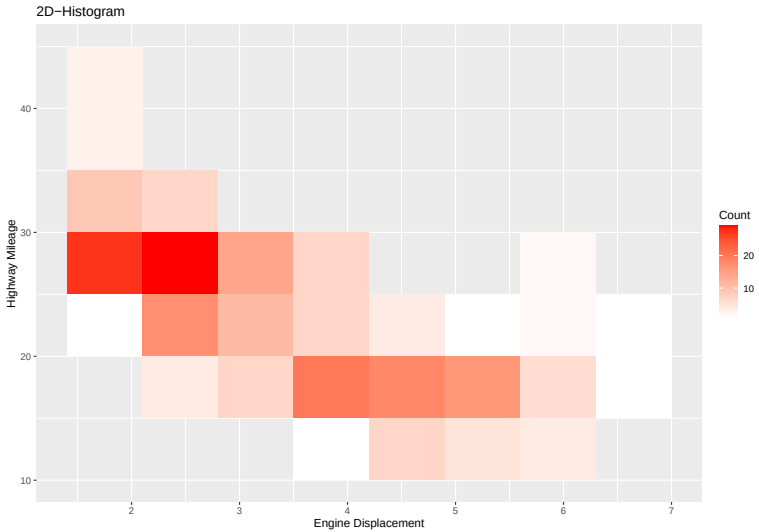
2D-Histograms

- Earlier, in slide 36, we introducted histograms.
- These are used to study the distribution of a **single** variable. It depicts the count of the variable within bins.
- The y-aesthetic was used to depict the counts.
- If we need to study the distribution of two variables at a time, we use the *fill* aesthetic to depict the counts.

```
ggplot(mpg) +
  geom_bin2d(aes(x=displ, y=hwy), binwidth=c(0.7, 5)) +
  scale_fill_gradient(name = "Count",
                      low="white", high="red") +
  labs(title="2D-Histogram",
       x="Engine Displacement", y="Highway Mileage")
```

2D-Histograms

cont'd



2D-Histograms

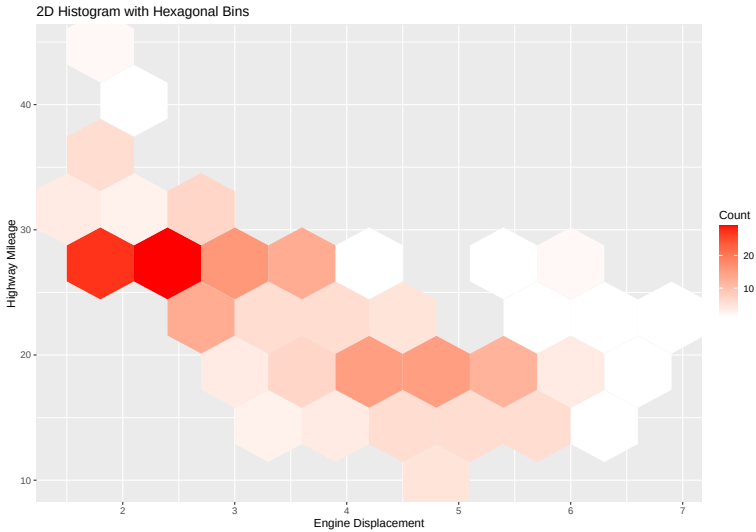
tiles to hexagonal bins

- Be defaults, a *tile* geom is used to represent each bin.
- Previously, in the 1-D case, the *column/bar* geom was used to represent each bin.
- It has been said that the tiles create a "block" effect, so it is often swapped out for hexagonal bins.

```
p1 <- ggplot(mpg) +
  geom_hex(aes(x=displ, y=hwy), binwidth=c(0.6, 5)) +
  scale_fill_gradient(name = "Count",
                      low="white", high="red") +
  labs(title="2D Histogram with Hexagonal Bins",
       x="Engine Displacement", y="Highway Mileage")
p1
```

2D-Histograms

cont'd



2D-Histograms

Points to Note

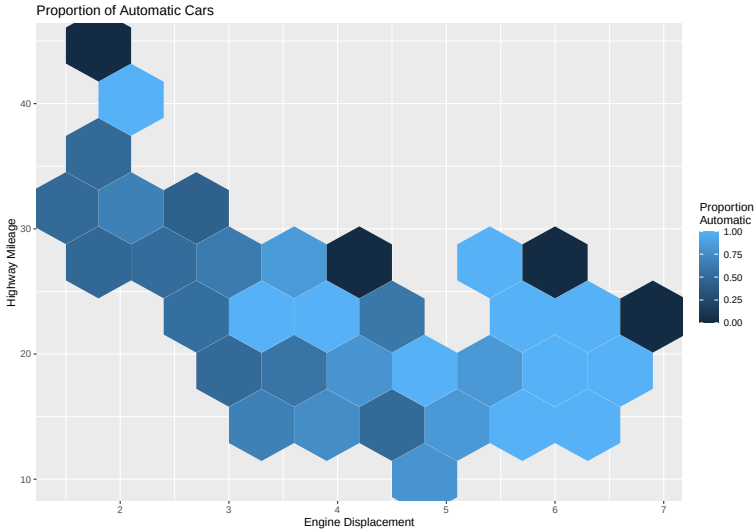
- Bear in mind that it is the fill aesthetic that represents counts, using a continous colour scale.
- Sometimes, we may wish to map a variable *other than count* to the fill aesthetic.
- In those cases, we have to use `stat_summary_2d` or `stat_summary_hex`.

Using stat_summary_hex()

```
mpg2 <- mutate(mpg,
        ma3=if_else(str_detect(trans, "auto"), 1, 0))
p2 <- ggplot(mpg2, mapping=aes(x=displ, y=hwy, z=ma3))+
 stat_summary_hex(fun= mean, binwidth=c(0.6, 5)) +
 scale_fill_gradient(name = "Proportion\nAutomatic") +
 labs(title="Proportion of Automatic Cars",
      x="Engine Displacement",
      y="Highway Mileage")
p2
```

Using `stat_summary_hex()`

cont'd



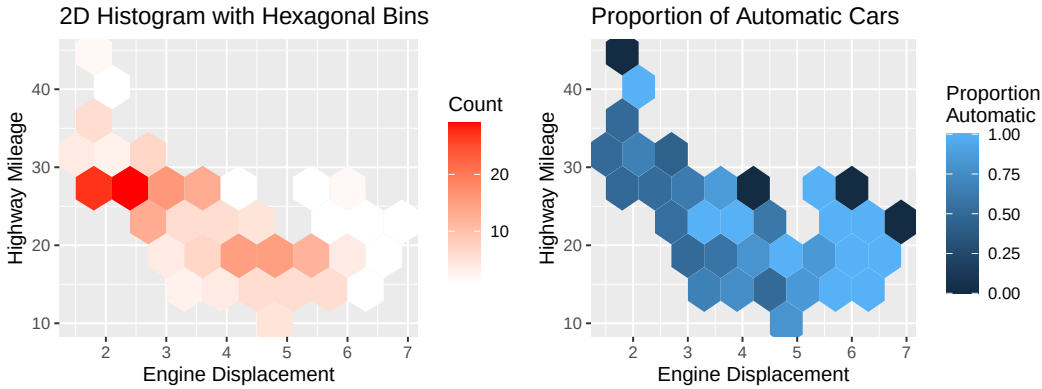
Grid of Plots

- When we made base-R plots, we used `mfrow` within `par()` to create sub-plots.
- To do similar things with ggplot, we can use the `gridExtra` package.
- Remember that a ggplot object is just another object in R; it is only when we *print* it that it is shown/created.

```
library(gridExtra)
grid.arrange(p1, p2, nrow=1)
```

Grid of Plots

two plots

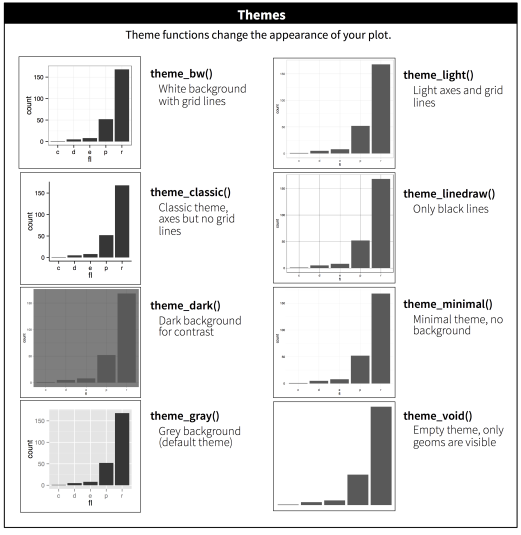


ggplot2 Themes

- You may be wondering why the background of a ggplot2 graph is always light gray.
- There are several reasons that the designers have for this:
  1. White grid lines are visible, yet easy to tune out, keeping the data prominent.
  2. The grey background gives a similar colour to typographic text, preventing it from jumping out.
  3. It creates a continuous field of colour which ensures that the plot is perceived as a single visual entity.
- You may agree with these or disagree with these points. If you would like to alter some of these elements of the plot, they can be done by selecting a different theme for your plot.
  - It is theoretically possible to customise every single element, right down to the font style for each axes.

List of Themes

- However, if you do not like the default look, there are other themes that you can turn to.

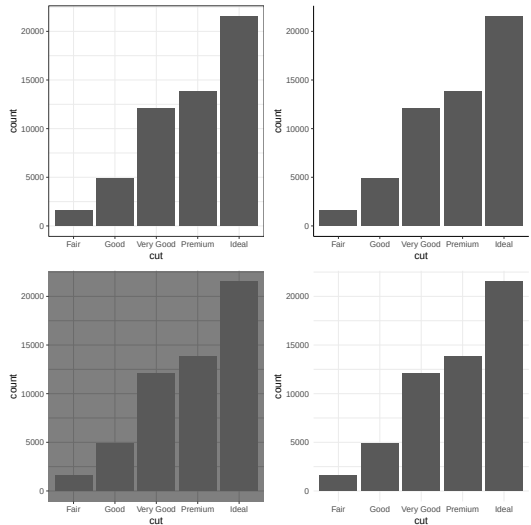


Grid Of Themes

```
p0 <-  ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
p1 <- p0 + theme_bw(); p2 <- p0 + theme_classic()
p3 <- p0 + theme_dark(); p4 <- p0 + theme_minimal()
grid.arrange(p1, p2, p3, p4, nrow=2)
```

Grid Of Themes

plot

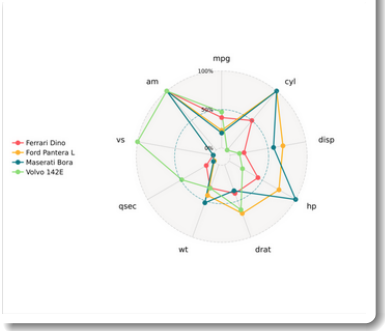


Extensions

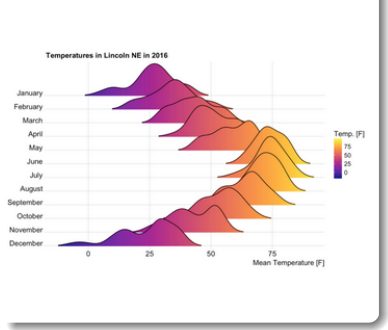
- The ggplot2 universe is maturing. There are many extensions that extend the available geoms.
- Go to `https://exts.ggplot2.tidyverse.org/gallery/` to see what's available.
- Here are three examples:

ggally

ggradar



ggridges



Layered Grammar of Graphics

- Our initial template (from slide 8) can be extend to:

```
ggplot(data = <DATA>) +
   <GEOM_FUNCTION>(
      mapping = aes(<MAPPINGS>),
      stat = <STAT>,
      position = <POSITION>
   ) +
   <COORDINATE_FUNCTION> +
   <FACET_FUNCTION>
```

Seven Parameters

- The 7 parameters in the template compose a formal grammar of graphics.
- *Any* plot can be described as a combination of the 7 parameters.
- Once again, the Rstudio cheatsheet for ggplot is invaluable:
  `https://www.rstudio.org/links/data_visualization_cheat_sheet`

Common Layers

- To modify a particular geom, work with the `geom_xxx()` layer.
- To modify the mapping for a particular aesthetic, look into one of the `scale_xxx_yyy()` layers. For instance, to modify the colours used in the *fill* aesthetic, look up `scale_fill_discrete()` aesthetic.
- To modify low-level elements of a graph, e.g. tick marks, tick positions, etc., look up the `theme()` layer.

Which Geom to Use???

The following thought process may help:

- Think first in terms the question you wish to answer. If there is no question, pick two variables.
- Once you have these two variables, inspect their types (numeric, factor, integer, or date?). This would narrow the number of geoms down.
- Then, recall the "ranking" that Cleveland came up with (from topic 04). What is the audience of your chart? How detailed a comparison do you think they will need to make?
- Create a basic plot and look closely at it. Think about whether the question can be answered. If it cannot, come up with some ideas why:
  - Do we need to add another variable?
  - Do we need to transform the data? Do we need to bin the data differently?
  - Do we need to add a smooth?
  - Why are there so many outliers? Do they follow some pattern that could be explained by another variable?