Introduction to R for Auditors

2022 ALGA Annual Conference Workshop

Lisa Callas

City of Edmonton

2022/05/04

Objectives

1. Be able to import and conduct basic analytics using your own (simple!) data in RStudio

2. Be able to set up a model working paper in RStudio to align to auditing standards

3. Be able to create print-worthy visualizations using ggplot2

Introductions

Lisa Callas

Former Internal Auditor with the City of Edmonton

Sr. Human Resources Business Partner

Instructor with NorQuest College

Tools:

- R & RStudio
- Excel
- Google Sheets
- Google Data Studio
- Database design

Vince Callas

IT Analyst with Alberta Health Services

Tools:

- Tableau
- SQL
- General Hardware and Software Support

Agenda

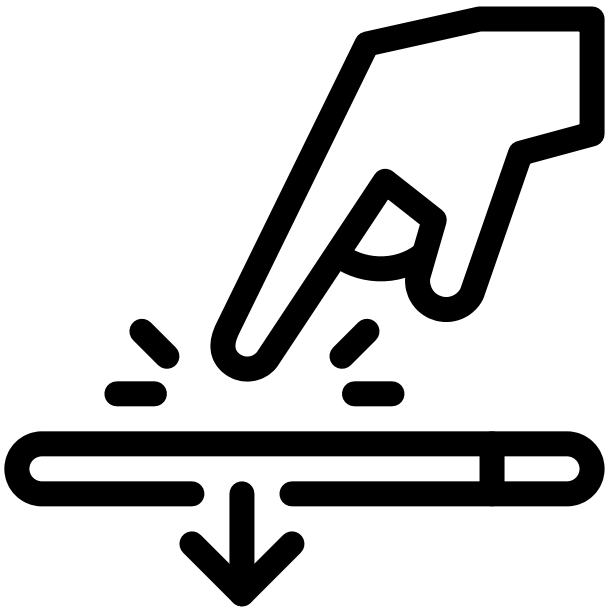
1. Overview of programming and RStudio
2. Planning your analysis and creating an integrated working paper
3. Basic data manipulation and visualization using a built-in data set
4. Reshaping data sets, working with dates and strings
5. Individual project development and support

Programming & RStudio

Programming vs. Excel

Telling

Doing

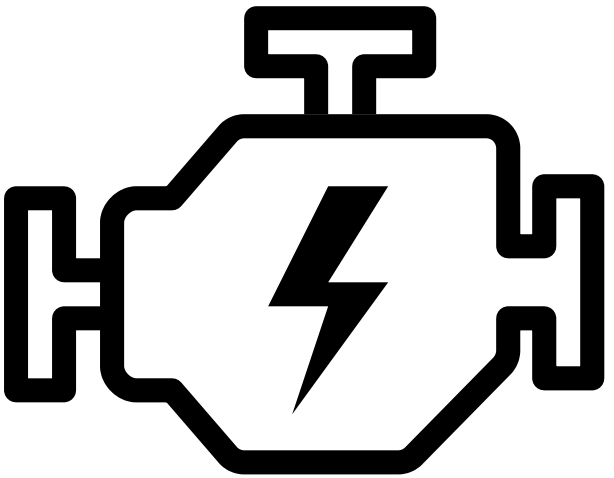


Benefits

- Free
- Open source
- Supportive user community
- Robust learning resources

R & RStudio

R is the Engine



RStudio is the Dashboard

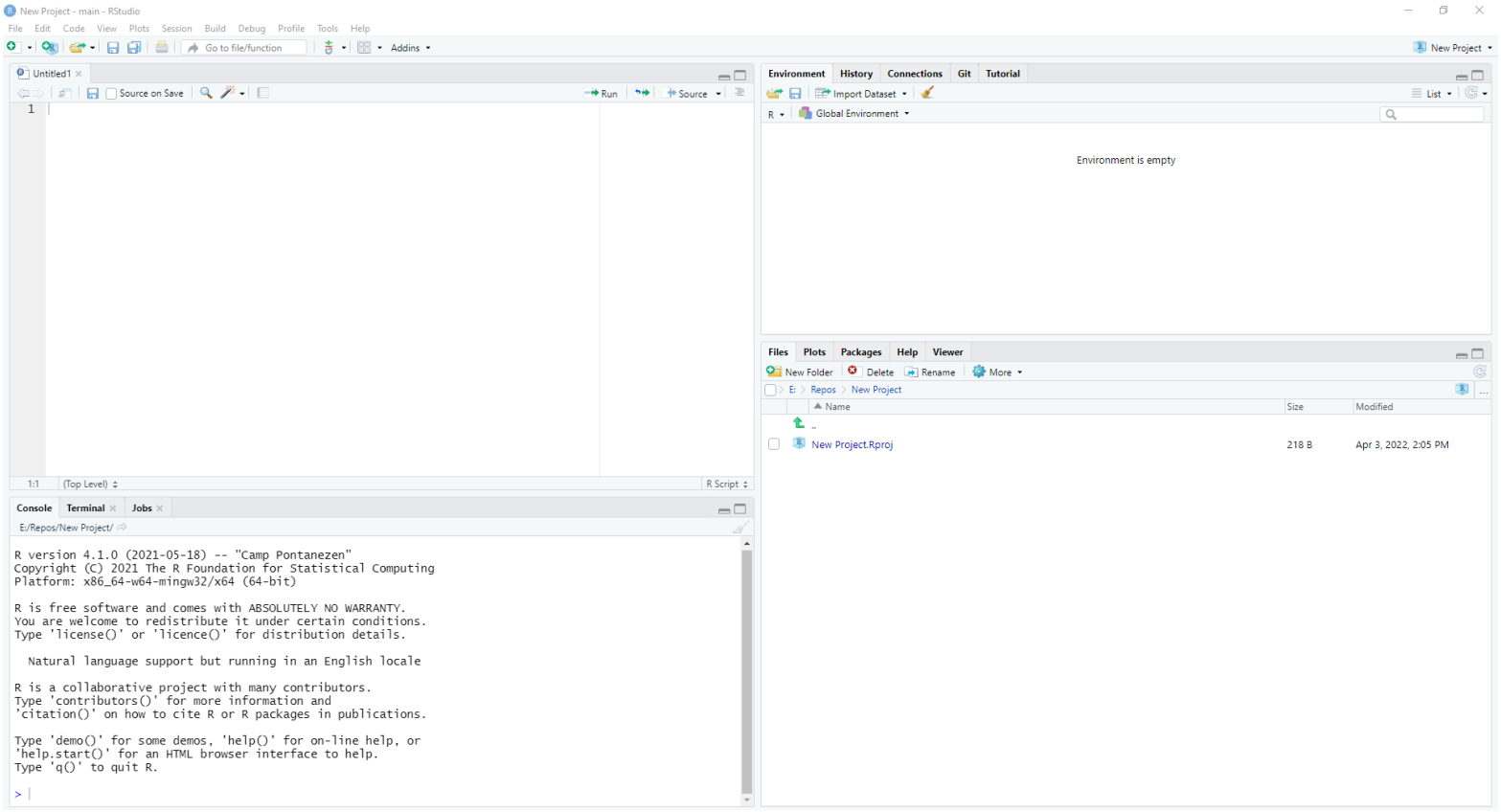
Installation

R: https://www.r-project.org/

RStudio: https://www.rstudio.com/products/rstudio/download/

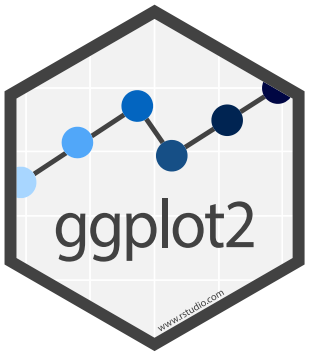
RStudio Cloud: https://rstudio.cloud/

Navigating RStudio



Packages

Packages are programming extensions that contain code, data and documentation in a consistent format.



...like a book

Think about packages like a book.

To use the information in a book, first you must buy the book (install.package("package_name")), then you have to open it up every time you want to reference anything in it (library(package_name)).

```
library(tidyverse)
```

Planning Your Analysis and Working Paper

Directories & Projects

A directory in RStudio is the default location where R will look for your files and save your scripts.

You can define a working directory to keep all of your files and scripts together in a folder on your computer (in the same way you'd set up a folder to keep related documents together).

Projects are associated with working directories. When you start a new project in RStudio and define it's path, that becomes your working directory for the project where RStudio will look for and save your files.

Activity: Set up a new project

R Script

Benefits

- Miminal set up
- Can be 'called' and reused modularly
- Simple

Challenges

- All integrated documentation is through #commenting
- Requires re-running each time to see output

R Markdown

Benefits

- Commentary, code, and results are integrated into a single document
- Output can be in .pdf, .html, or .doc
- Can create a fully comprehensive working paper

Challenges

- Requires some additional planning and set up
- Additional learning curve working with code chunks to ensure the desired output
- Requires some additional troubleshooting for errors and technical problems 'knitting' documents

Activity: Create a a new R Markdown document, and a new script in your project

Helpful hint: When closing your scripts, save your scripts but DO NOT SAVE the workspace.

Work Structure & Sequencing

- Sequential processing: Your code will run in the sequence it is written.
- Comments: Add comments to your code by using #
- Script headers: Best practice is to include a script header on your R scripts. Rmarkdown documents may not need an explanatory header depending on the design of the working paper.
- Snippets: You can custom design snippets (little pieces of code or comments) to support consistency in your scripts. These are great shortcuts for script headers and script sections.
- Libraries: Best practice is to include your libraries at the beginning of your code - but this is not required.

Activity: Format your R Script

- Set up a header snippet
- Insert a header
- Install and call libraries

Tidyverse

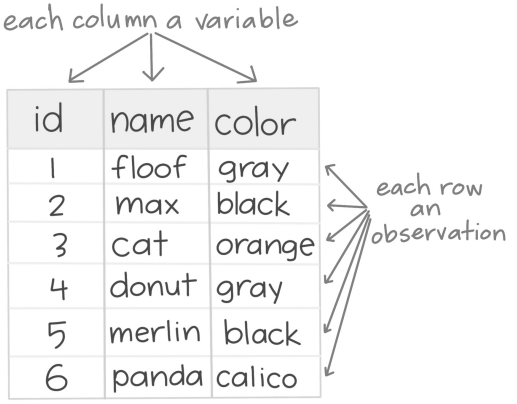
Data Manipulation

Tidy Data

> "TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."
>
> –HADLEY WICKHAM

In tidy data:

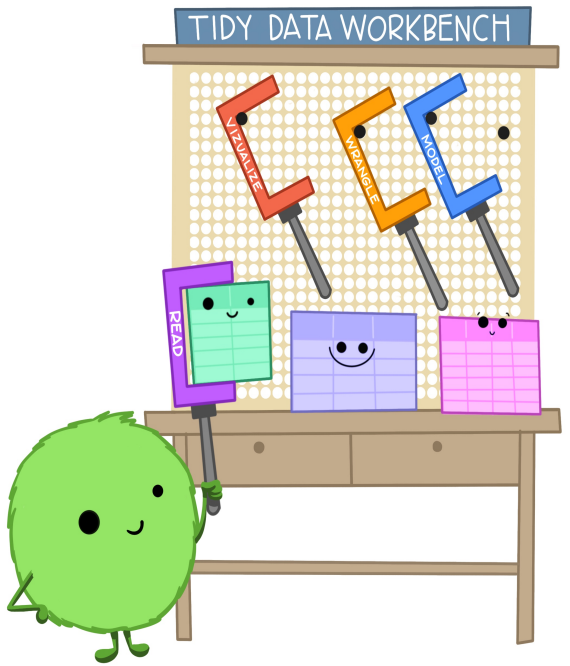
- each variable forms a column
- each observation forms a row
- each cell is a single measurement



Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

Artwork by @alisonhorst

When working with tidy data, we can use the same tools in similar ways for different datasets...



...but working with untidy data often means reinventing the wheel with one-time approaches that are hard to iterate or reuse.

Artwork by @alisonhorst

Built-in Datasets

- Base R has a number of built in datasets.
- Additionally, almost all packages come with built in datasets.
- Often the online help for using functions is demonstrated on these built in datasets so that you can reproduce the same demo in your own script and get identical results.
- Built-in data has typically be cleansed and is tidy.

Activity: List the Built-in Datasets & Call a Few to Explore

```
data()

Titanic
iris
mtcars
```

Data Structures and Types

Structures:

- vector: a sequence of data elements of the same basic type
- data frame: a table or 2 dimensional array-like structure
- factor: a variable used to categorize data with a limited number of values
- list: objects that contain elements of different types
- matrix: 2 dimensional, homogenous data structure. Single data type

Types:

- character
- numeric
- integer
- logical
- complex

Objects may have attributes such as name, dimension, and class

Activity: Read in and explore a built-in dataset

```
iris <- iris
```

Vectors and Variables

The assignment function: <- or -> (shortcut is Alt + _)

Using the assignment function creates named objects in your environment that you can use in your code.

Activity: Create variables and vectors

```
today <- Sys.Date()

months <- 12

total <- c(6,45,21,34,643,11,865)
```

Operators

Arithmetic	+	-	*	/	%%	%/%	^
Relational	<	>	==	<=	>=	!=	
Logical	&	|	!	&&	||		
Assignment	=	<-	->	<<-	->>		
Misc.	:	%in%	%*%				

Activity: Complete a calculation to create a variable

```
amount <- 12+4

x <-  months * total
```

Functions

Functions automate common tasks and calculations.

hint: Cheatsheet as a help resource (see RStudio Help menu for a drop down option)

Common Functions:

```
sum() #add
mean() #average
min() #minimum value
max() #maximum value
n() #count of values


select() #chooses which columns
filter() #chooses which rows
mutate() #creates and modifies columns
group_by() #groups together for summarizing
summarize() #summarizes based on groups
```

Piping

The pipe (%>%) operator allows you to string a series of functions together into a code block.

%>% is the equivalent of saying "...and then do..." between instructions.

Shortcut is Cntl+Shift+m

```
iris %>%
  select(Species, Petal.Width) %>%
  filter(Species == "setosa") %>%
  slice_sample(n=10)
```

```
##    Species Petal.Width
## 1   setosa         0.2
## 2   setosa         0.2
## 3   setosa         0.2
## 4   setosa         0.2
## 5   setosa         0.2
## 6   setosa         0.2
## 7   setosa         0.2
## 8   setosa         0.1
## 9   setosa         0.2
## 10  setosa         0.1
```

Grouping and Summarizing

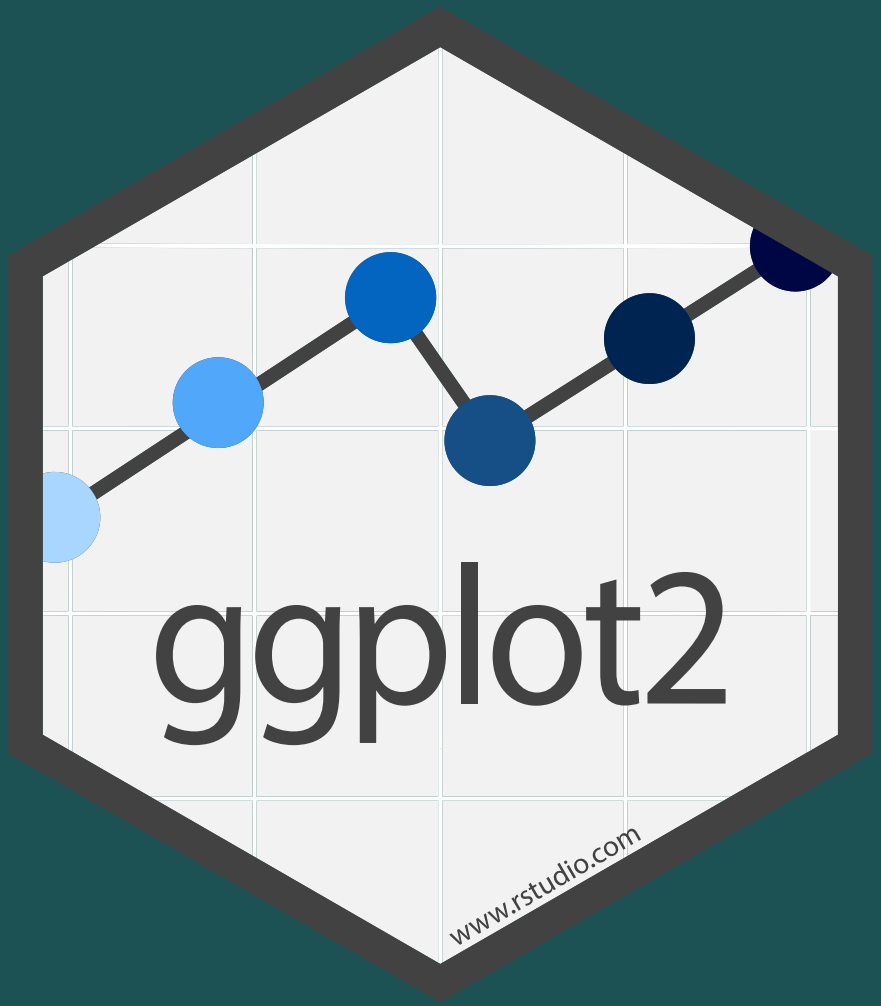
Grouping and summarizing data in categories is often required.

If the intent is to visualize the data, save the output as an object.

```
(iris %>%
  group_by(Species) %>%
  summarize(Total=n(),
            Mean=mean(Petal.Width)) ->species_summary)
```

```
## # A tibble: 3 x 3
##   Species    Total  Mean
##   <fct>      <int> <dbl>
## 1 setosa        50 0.246
## 2 versicolor    50 1.33
## 3 virginica     50 2.03
```

Visualization



Key Components

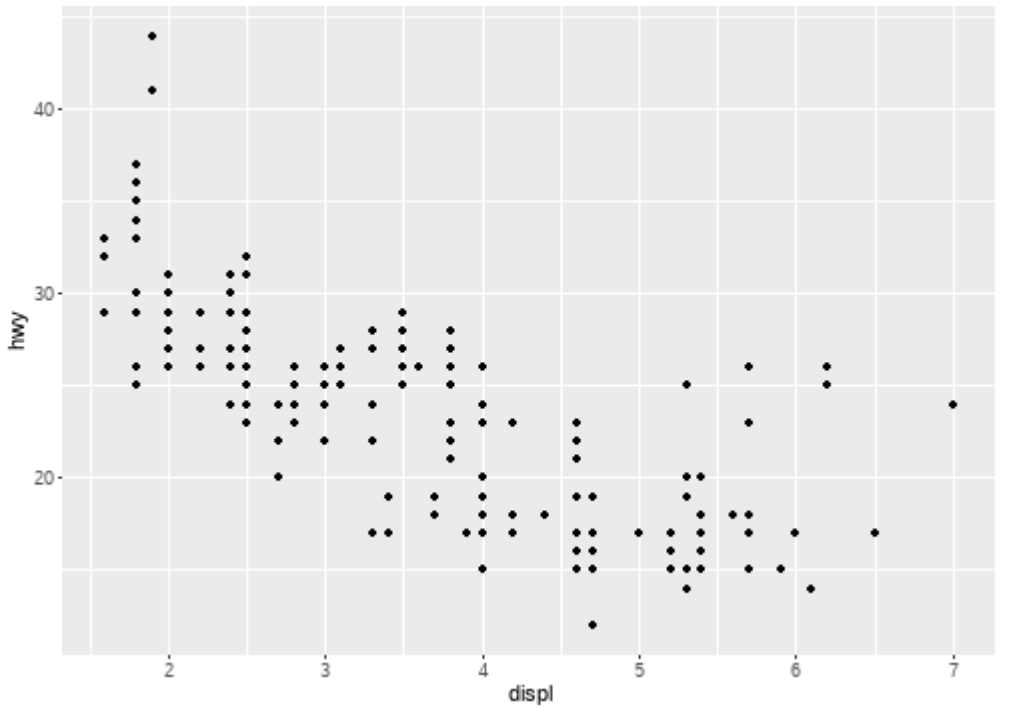
Every ggplot2 plot has 3 key components

1. Data
2. Aesthetic mappings between variables inthe data and visual properties
3. At least one layer describing how to render each observation (usually with a geom function)

Activity: Create graphs

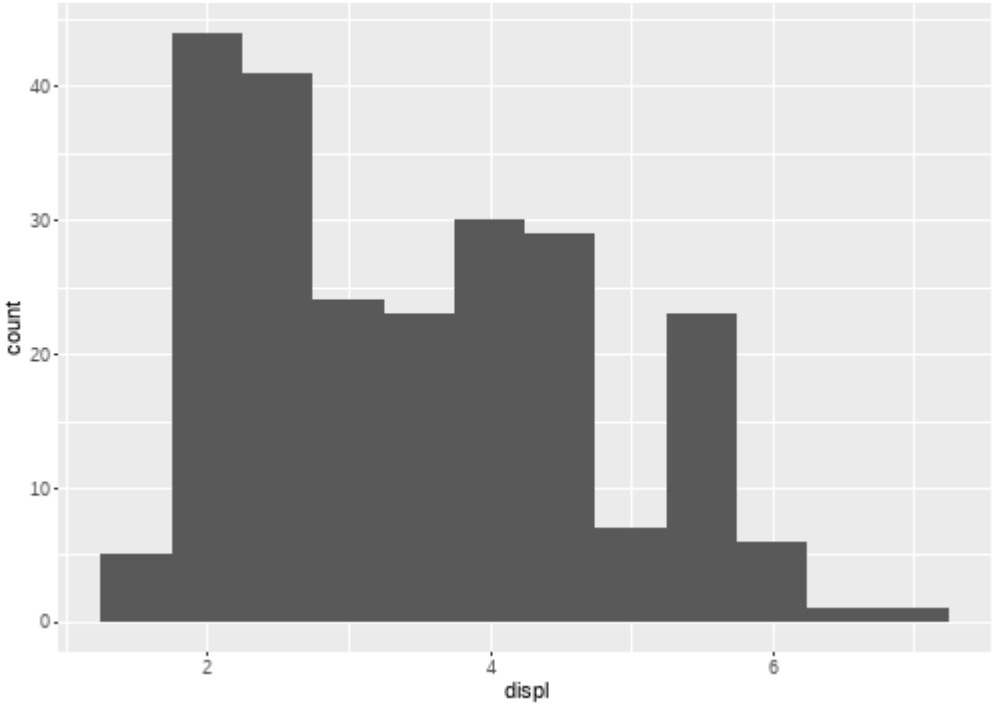
Create a point graph using the mpg dataset

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



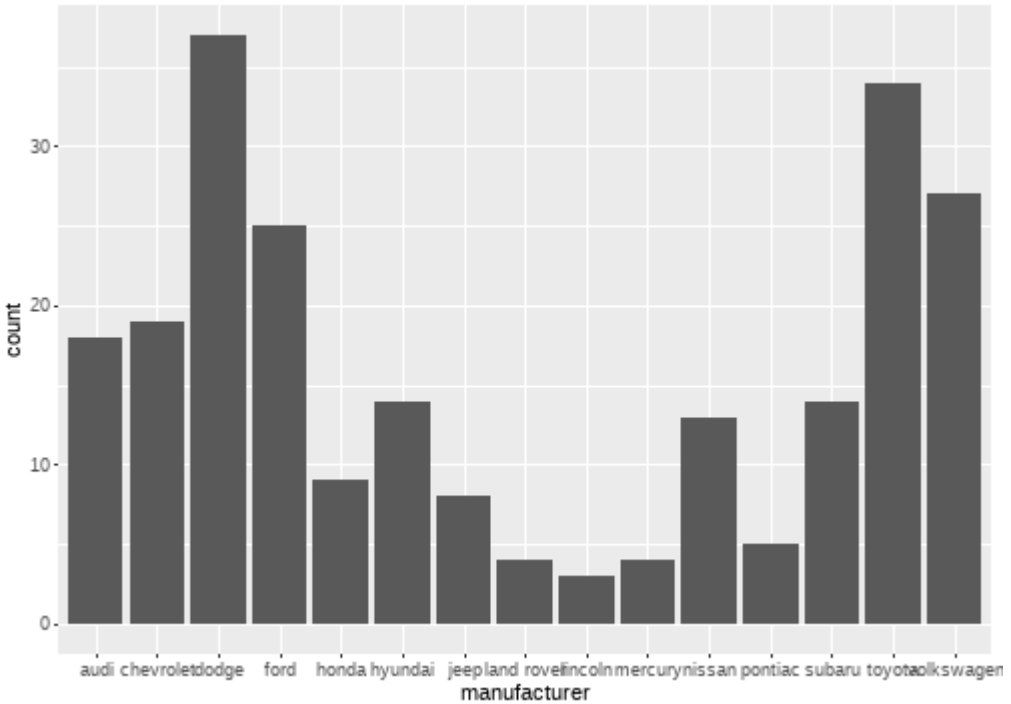
Create a histogram

```
ggplot(mpg, aes(displ)) +
  geom_histogram(binwidth = 0.5)
```



Create a barplot

```
ggplot(mpg, aes(manufacturer)) +
  geom_bar()
```



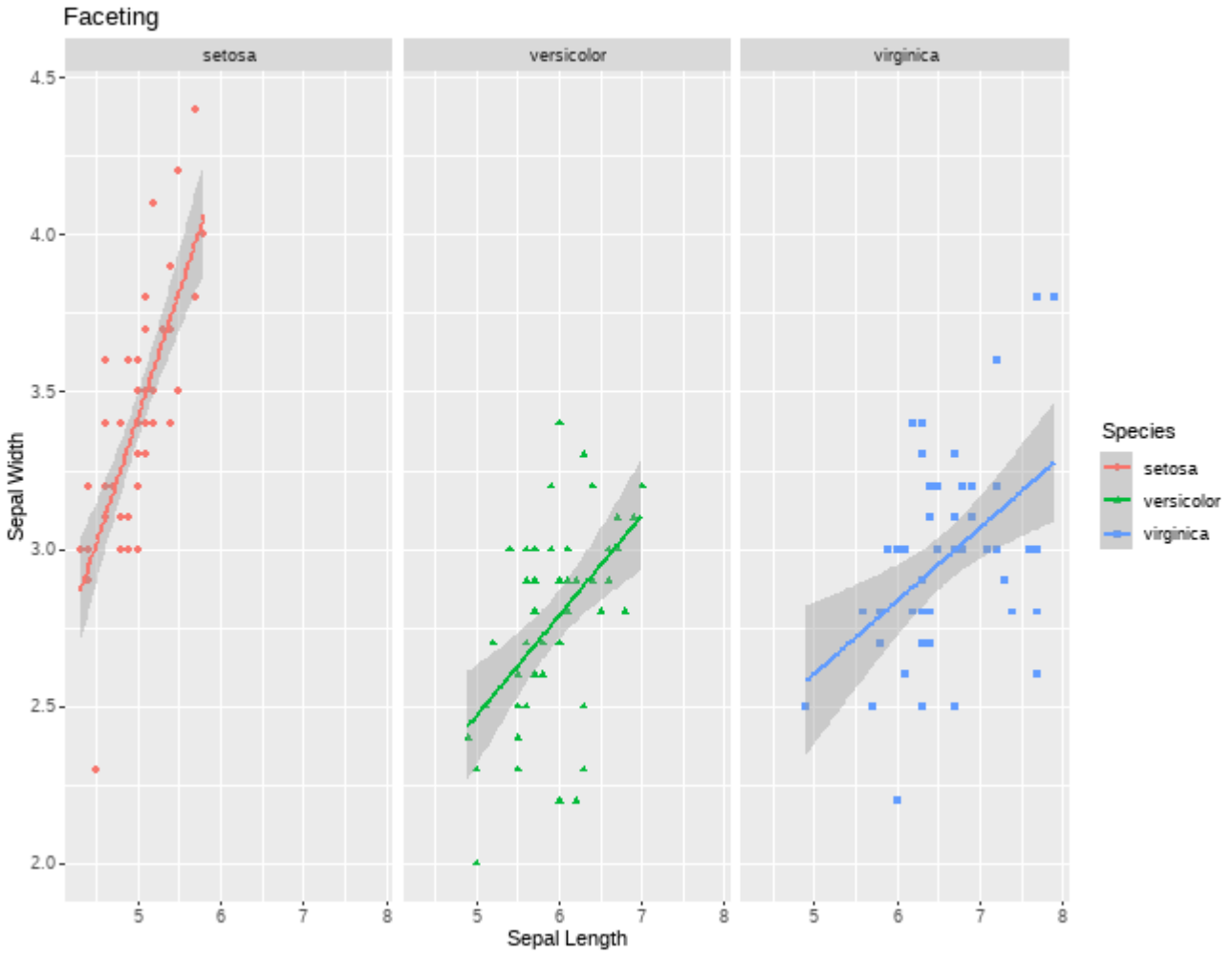
Customizing visualizations

- colour / fill
- facet
- labels
- size
- shapes
- themes

Layered demo

```
ggplot(data=iris, aes(Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(aes(shape=Species), size=1.5) +
  geom_smooth(method="lm") +
  xlab("Sepal Length") +
  ylab("Sepal Width") +
  ggtitle("Faceting")+
  facet_grid(. ~ Species)->faceted
```

```
faceted
```


Faceting
setosa
versicolor
virginica
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
8
Species
setosa
versicolor
virginica

Reshaping Datasets, Dates, & Strings

Pivoting

Often operational data is not in a tidy format and needs to be reshaped before it can be more easily used.

Example:

```
(data <- read_csv("untidy_data.csv"))
```

```
## # A tibble: 7 x 9
##   ...1                  `2015` `2016` `2017` `2018` `2019` `2020` `2021` `
##   <chr>                  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  
## 1 Human Resources        14791  10835  14746   5650   6584  18874   1120  
## 2 Financial Services      5919  14645   1074  15281   9322  17813   8716  
## 3 City Operations         8013  13441   5863  12980    297   2113  10835  
## 4 Community Services      4093   7544   3135   5964  14101  15858  12259  
## 5 Office of the City Ma~  7088  14643  19856   9497  17651  11994   3803  
## 6 Legal and Risk Manage~  5267   2336  16924  19485   6907  19239   6082  
## 7 Boards and Commissions  8337   4945  11687  11579   8875  16856  10400
```

Pivot Longer

```
data %>%
  pivot_longer(cols=c(-...1),
               names_to = "Year",
               values_to = "Values")
```

```
## # A tibble: 56 x 3
##    ...1                Year  Values
##    <chr>               <chr>  <dbl>
##  1 Human Resources     2015   14791
##  2 Human Resources     2016   10835
##  3 Human Resources     2017   14746
##  4 Human Resources     2018    5650
##  5 Human Resources     2019    6584
##  6 Human Resources     2020   18874
##  7 Human Resources     2021    1120
##  8 Human Resources     2022   10685
##  9 Financial Services  2015    5919
## 10 Financial Services  2016   14645
## # ... with 46 more rows
```

Dates

If a date column isn't automatically recognized, a character or numeric column will need to be converted to a date format.

```
library(lubridate)

(sample_sick_data <- read_csv("sample_sick_data.csv"))
```

```
## # A tibble: 94,822 x 3
##    `Event Date` `Hours Absent` `Function Category`
##    <date>                <dbl> <chr>
##  1 2021-11-23              8    SI - STD Pay @ 100%
##  2 2021-11-24              8    SIE - STD Pay @ 100% Incident End
##  3 2021-02-26              6.75 SIE - STD Pay @ 100% Incident End
##  4 2021-06-24              6.75 SIE - STD Pay @ 100% Incident End
##  5 2021-12-15              6.75 SI - STD Pay @ 100%
##  6 2021-12-16              6.75 SI - STD Pay @ 100%
##  7 2021-12-17              6.75 SIE - STD Pay @ 100% Incident End
##  8 2021-08-27              7.5  SIE - STD Pay @ 100% Incident End
##  9 2021-07-20             10    SIE - STD Pay @ 100% Incident End
## 10 2021-01-15             10    SI - STD Pay @ 100%
## # ... with 94,812 more rows
```

Replace the full event date with just the year

```
sample_sick_data %>%
  mutate(`Event Date`= year(`Event Date`))
```

```
## # A tibble: 94,822 x 3
##    `Event Date` `Hours Absent` `Function Category`
##           <dbl>          <dbl> <chr>
##  1         2021           8    SI - STD Pay @ 100%
##  2         2021           8    SIE - STD Pay @ 100% Incident End
##  3         2021           6.75 SIE - STD Pay @ 100% Incident End
##  4         2021           6.75 SIE - STD Pay @ 100% Incident End
##  5         2021           6.75 SI - STD Pay @ 100%
##  6         2021           6.75 SI - STD Pay @ 100%
##  7         2021           6.75 SIE - STD Pay @ 100% Incident End
##  8         2021           7.5  SIE - STD Pay @ 100% Incident End
##  9         2021          10    SIE - STD Pay @ 100% Incident End
## 10         2021          10    SI - STD Pay @ 100%
## # ... with 94,812 more rows
```

Strings

Filter for values that contain a specific string

```
sample_sick_data %>%
  filter(str_detect(`Function Category`, "SIE"))
```

```
## # A tibble: 11,041 x 3
##    `Event Date` `Hours Absent` `Function Category`
##    <date>                <dbl> <chr>
##  1 2021-11-24              8    SIE - STD Pay @ 100% Incident End
##  2 2021-02-26              6.75 SIE - STD Pay @ 100% Incident End
##  3 2021-06-24              6.75 SIE - STD Pay @ 100% Incident End
##  4 2021-12-17              6.75 SIE - STD Pay @ 100% Incident End
##  5 2021-08-27              7.5  SIE - STD Pay @ 100% Incident End
##  6 2021-07-20             10    SIE - STD Pay @ 100% Incident End
##  7 2021-01-18             14    SIE - STD Pay @ 100% Incident End
##  8 2021-07-05             10    SIE - STD Pay @ 100% Incident End
##  9 2021-11-28             14    SIE - STD Pay @ 100% Incident End
## 10 2021-02-12              8    SIE - STD Pay @ 100% Incident End
## # ... with 11,031 more rows
```

Individual Projects

GitHub

Demo of how to access and download resources from GitHub

Additional Resources

- R for Data Science
  (https://r4ds.had.co.nz/index.html)

- R Graphics Cookbook
  (https://r-graphics.org/)

- ggplot2: Elegant Graphics for Data Analysis
  (https://ggplot2-book.org/index.html)

- RStudio Cheat Sheets

- Twitter
  (@rstudiotips, @rweekly_org, @R-LadiesGlobal, @WeAreRLadies,
    @drob, @Rbloggers, @RLangTip, #rstats, #30DayChartChallenge)

- YouTube

- Stack Overflow