Loss modelling, reserving and fraud analytics in R - Prework

A hands-on workshop

Katrien Antonio & Jonas Crevecoeur

IA|BE workshop | June 3, 10 & 17, 2021

Prologue

Introduction

Course

https://github.com/katrienantonio/workshop-loss-reserv-fraud-2020

The course repo on GitHub, where you can find the data sets, lecture sheets, R scripts and R markdown files.

Us

https://katrienantonio.github.io/ and https://jonascrevecoeur.github.io/

katrien.antonio@kuleuven.be & jonas.crevecoeur@kuleuven.be

(Katrien, PhD) Professor in insurance data science at KU Leuven and University of Amsterdam

(Jonas, PhD) Post-doctoral researcher in biostatistics at KU Leuven

Checklist

☑ Do you have a fairly recent version of R?

```
version$version.string
## [1] "R version 4.0.3 (2020-10-10)"
```

☑ Do you have a fairly recent version of RStudio?

```
RStudio.Version()$version
## Requires an interactive session but should return something like "[1] ‘1.3.1093’"
```

☑ Have you installed the R packages listed in the software requirements?

or

☑ Have you created an account on RStudio Cloud (to avoid any local installation issues)?

What's out there - the R universe

What is R?

> The R environment is an integrated suite of software facilities for data manipulation, calculation and graphical display.

A brief history:

- R is a dialect of the S language.
- R was written by **R**obert Gentleman and **R**oss Ihaka in 1992.
- The R source code was first released in 1995.
- In 1998, the Comprehensive R Archive Network CRAN was established.
- The first official release, R version 1.0.0, dates to 2000-02-29. Currently R 4.0.3 (October, 2020).
- R is open source via the GNU General Public License.

Explore the R architecture

- R is like a car's engine

- RStudio is like a car's dashboard, an integrated development environment (IDE) for R.

R: Engine	RStudio: Dashboard

How do I code in R?

Keep in mind:

- unlike other software like Excel, STATA, or SAS, R is an interpreted language
- no point and click in R!
- you have to program in R!

R packages extend the functionality of R by providing additional functions, and can be downloaded for free from the internet.

R: A new phone	**R Packages: Apps you can download**

How to install and load an R package?

Install the {ggplot2} package for data visualisation

```
install.packages("ggplot2")
```

Load the installed package

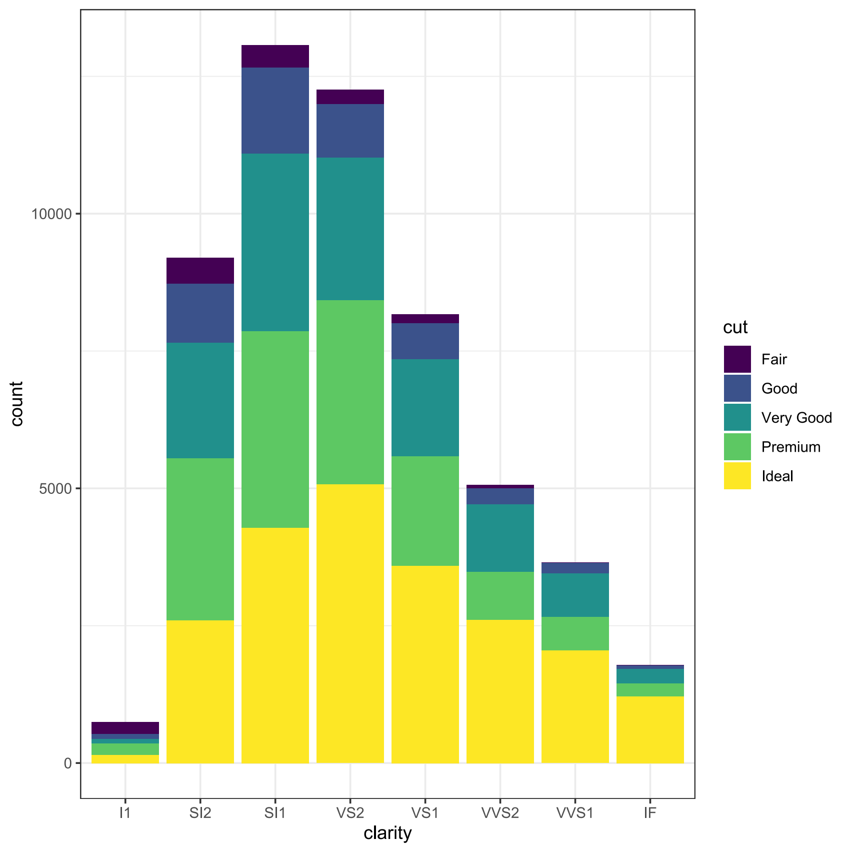
```
library(ggplot2)
```

And give it a try

```
head(diamonds)
ggplot(diamonds, aes(clarity, fill = cut)) +
  geom_bar() + theme_bw()
```

Packages are developed and maintained by R users worldwide.

They are shared with the R community through CRAN: now 16,460 packages online (on November 2, 2020)!



Why R and RStudio?

Data science positivism

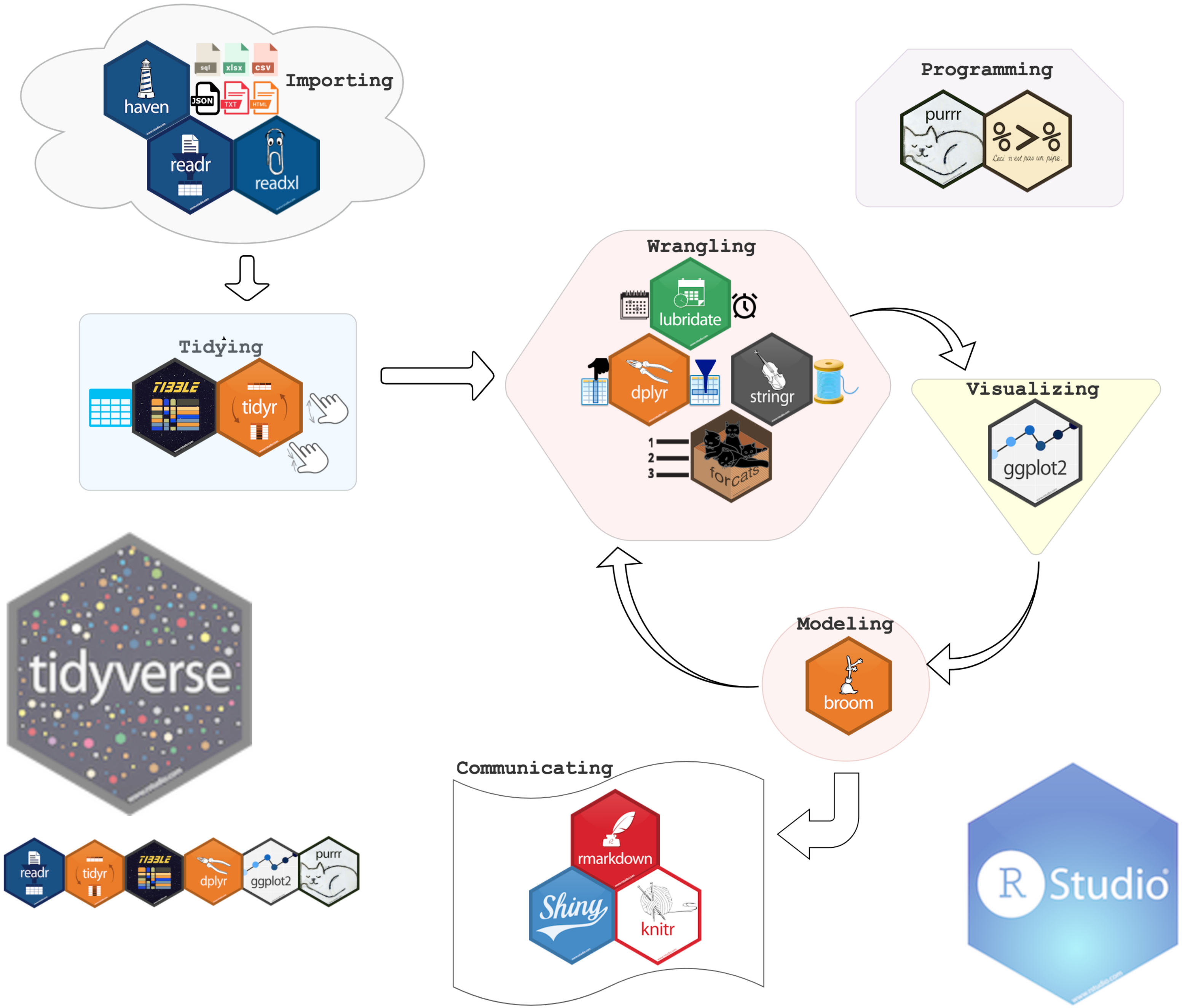
- Next to Python, R has become the *de facto* language for data science, with a cutting edge *machine learning toolbox*.
- See: The Popularity of Data Science Software
- R is open-source with a very active community of users spanning academia and industry.

Bridge to actuarial science, econometrics and other tools

- R has all of the statistics and econometrics support, and is amazingly adaptable as a “glue” language to other programming languages and APIs.
- R does not try to be everything to everyone. The RStudio IDE and ecosystem allow for further, seemless integration (with e.g. python, keras, tensorflow or C).
- Widely used in actuarial undergraduate programs

Disclaimer + Read more

- It's also the language that we know best.
- If you want to read more: R-vs-Python, when to use Python or R or Hadley Wickham on the future of R



Importing

Tidying

Wrangling

Programming

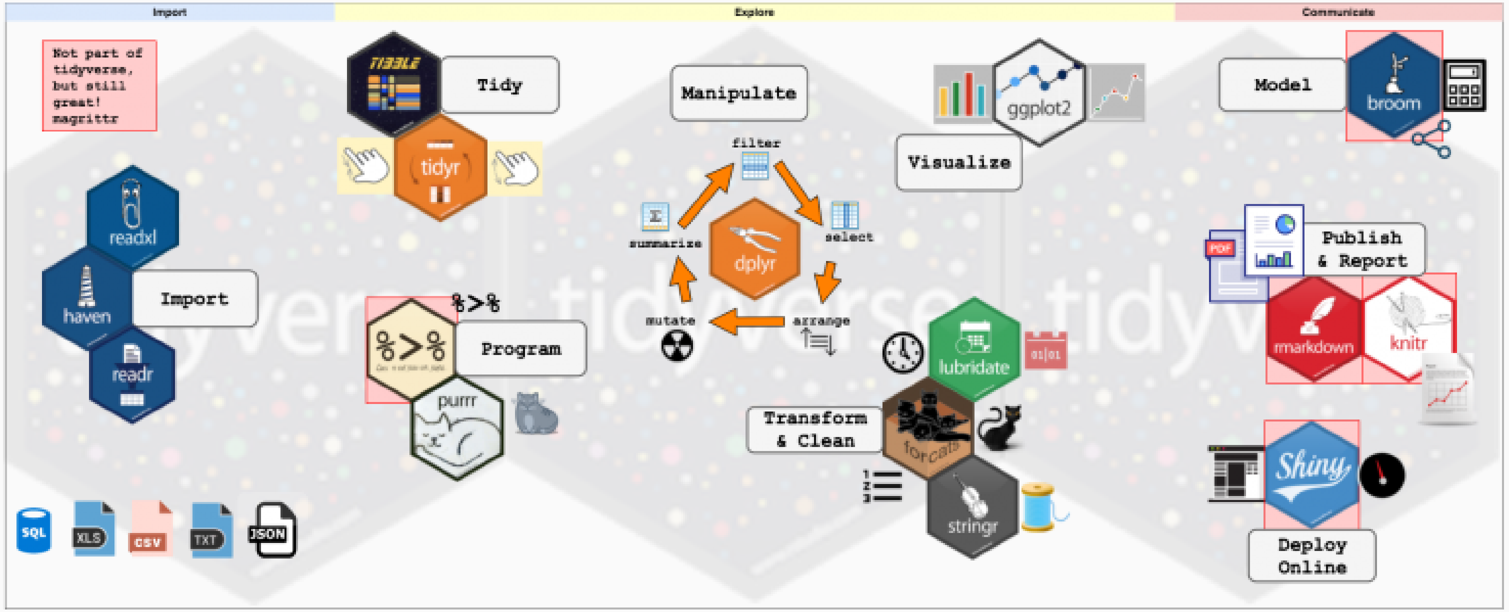
Visualizing

Modeling

Communicating

Welcome to the tidyverse!

> The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.



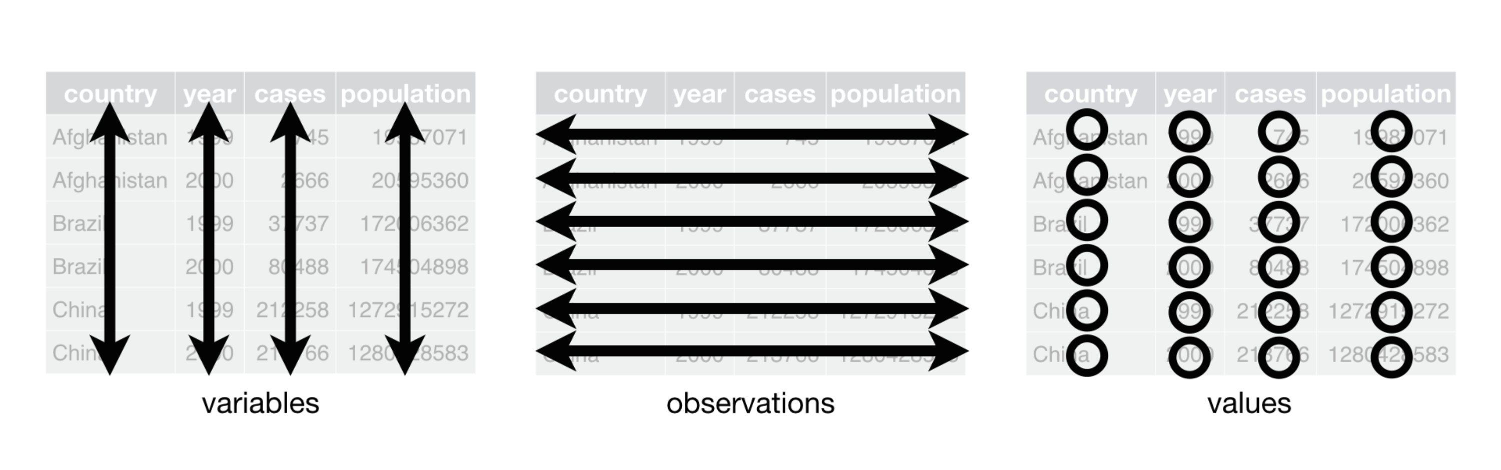
More on: tidyverse.

Install the packages with `install.packages("tidyverse")`. Then run `library(tidyverse)` to load the core tidyverse.

Principles of tidy data

Three interrelated rules from the R for data science book by Garrett Grolemund and Hadley Wickham:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.



This figure is taken from Chapter 12 on Tidy data in R for data science.

Workflow of a data scientist

Here is a model of the **tools needed in a typical data science project**:

> Together, tidying and transforming are called **wrangling**, because getting your data in a form that's natural to work with often feels like a fight!
>
> Models are complementary tools to visualisation. Once you have made your questions sufficiently precise, you can use a model to answer them. Models are a fundamentally **mathematical or computational tool**, so they generally scale well. But every model makes **assumptions**, and by its very nature a model cannot question its own assumptions. That means **a model cannot fundamentally surprise you**.

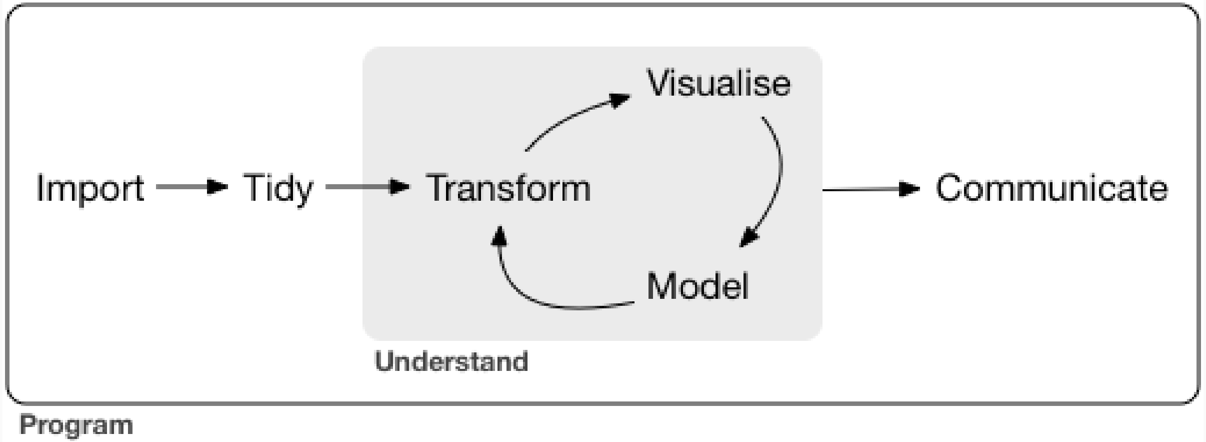


Figure and quote taken from Chapter 1 in R for data science.

Data wrangling and visualisation

A tibble instead of a data.frame

Within the tidyverse `tibble` is a modern take on a `data.frame`:

- keep the features that have stood the test of time
- drop the features that used to be convenient but are now frustrating.

You can use:

- `tibble()` to create a new tibble
- `as_tibble()` transforms an object (e.g. a data frame) into a tibble.

Quick example: explore the differences!

```
mtcars
# install.packages("tidyverse")
library(tidyverse)
as_tibble(mtcars)
```

Chains with the pipe operator

In R, the pipe operator is `%>%`.

It takes the output of one statement and makes it the input of the next statement.

When describing it, you can think of it as a “THEN”; with this operator it becomes easy to chain a sequence of calculations.

For example, when you have an input data and want to call functions `foo` and `bar` in sequence, you can write `data %>% foo %>% bar`.

A first example:

- take the `diamonds` data (from the {ggplot2} package)
- then subset

```
diamonds %>% filter(cut == "Ideal")
```

Some excellent blog posts about this operator: Pipes in R tutorial for beginners and how to write this in base R.

Data manipulation verbs

The {dplyr} package holds many useful data manipulation verbs:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names
- `filter()` picks cases based on their values
- `summarise()` reduces multiple values down to a single summary
- `arrange()` changes the ordering of the rows.

These all combine naturally with `group_by()` which allows you to perform any operation "by group".

A first example:

```r
diamonds %>% mutate(price_per_carat = price/carat) %>% filter(price_per_carat > 1500)
```

or

```r
diamonds %>% group_by(cut) %>% summarize(price = mean(price), carat = mean(carat))
```

Your turn

To get warmed up, let's do some basic explorations of the {tidyverse} instructions. The idea is to get some feel for these functions.

Q: you will work through the following exploratory steps.

1.1. Create a data frame (with `data.frame(.)`) or tibble (with `tibble(.)`) `df` with two variables `x` and `y`. Enter some values for these variables.

1.2. Create a new variable `z` that is the sum of `x` and `y`. Use `base` R instructions and then use the pipe operator and `mutate(.)`.

2.1. Create a new data vector `v` with some entries, use `c(.)`.

2.2. Try the following instructions:

```
round(mean(x), 2)
mean(x) %>% round(2)
x %>% mean %>% round(2)
```

First, you put together the `data.frame`

```
df <- data.frame(x = c(0, 1), y = c(0, 1))
df
```

or the `tibble`

```
df <- tibble(x = c(0, 1), y = c(0, 1))
df
```

Next, you create a new variable

```
df$z <- df$x + df$y
df
```

or with `mutate(.)`

```
df %>% mutate(z = x+y)
df
```

You create a vector `x` with some entries

```
x <- c(0.157, 0.135, 0.359)
```

and then you evaluate

```
round(mean(x), 2)
mean(x) %>% round(2)
x %>% mean %>% round(2)
```

These implementations all lead to the same result:

```
## [1] 0.22
```

Which one do you find most intuitive?

Plots with ggplot2

The aim of the {ggplot2} package is to create elegant data visualisations using the **grammar of graphics**.

Here are the basic steps:

- begin a plot with the function `ggplot()` creating a coordinate system that you can add layers to
- the first argument of `ggplot()` is the dataset to use in the graph

A first example

```r
library(ggplot2)
ggplot(data = mpg)
ggplot(mpg)
```

creates an empty graph.

You will now add layers to this graph!

Plots with ggplot2

You complete your graph by adding one or more **layers** to `ggplot()`.

For example:

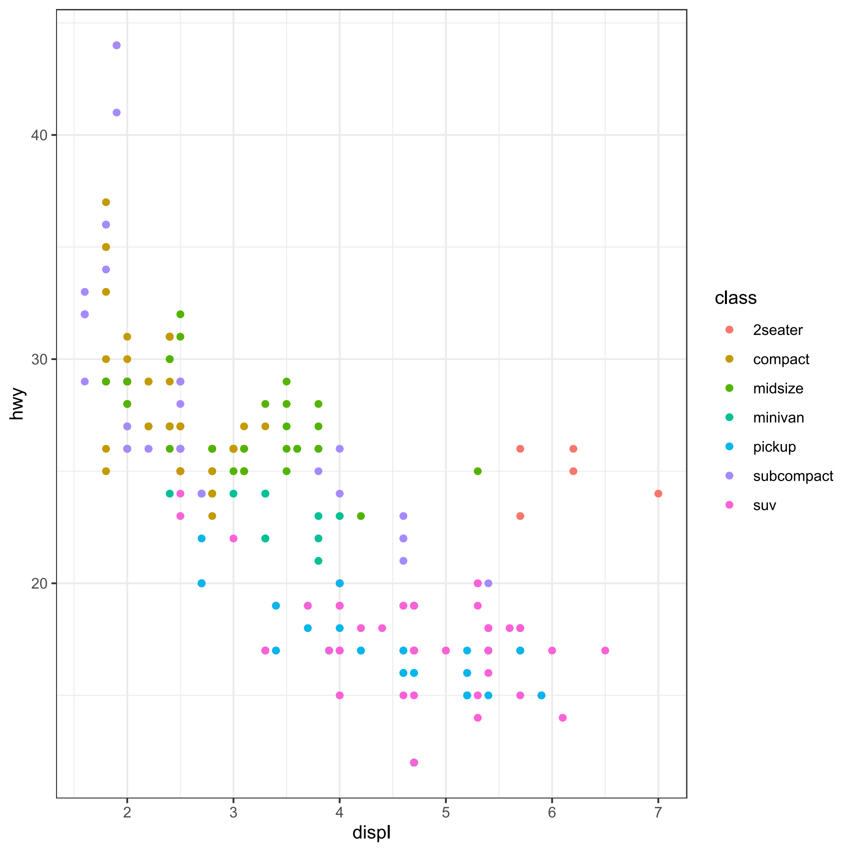
- `geom_point()` adds a layer of points to your plot, which creates a scatterplot
- `geom_smooth()` adds a smooth line
- `geom_bar` a bar plot

and many more, see ggplot2 documentation.

Each `geom` function in `ggplot2` takes an aesthetic mapping argument:

- maps variables in your dataset to visual properties
- always paired with `aes()` and the x and y arguments of `aes()` specify which variables to map to the x and y axes.

```r
library(ggplot2)
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point() + theme_bw()
```



Extend the empty graph now with (here: global) aesthetic mapping argument `aes(displ, hwy, colour = class)`.

This implies: `displ` on the x-axis, `hwy` on the y-axis and `class` to differentiate the color of the plotting symbol.

With `geom_point` you add a layer of points to the empty graph.

`theme_bw()` changes the `ggtheme` to a simple black-and-white theme.

What else is there?

Recall

> The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

There are (multiple) alternative ways to do what the packages and functions in the tidyverse do.

For instance:

- base R
- the {data.table} package

You can read more about comparisons on e.g. how to write this in base R or Base R, the tidyverse, and data.table: a comparison of R dialects to wrangle your data.

Thanks!

Slides created with the R package xaringan.

Course material available via

https://github.com/katrienantonio/workshop-loss-reserv-fraud-2020