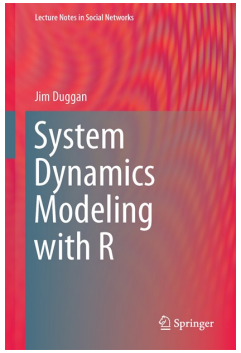
The first principle I propose is that our *Mission*, as users and creators of software for data analysis, is to enable the best and most thorough exploration of data possible. That means that users of the software must be able to ask meaningful questions about their applications, quickly and flexibly.

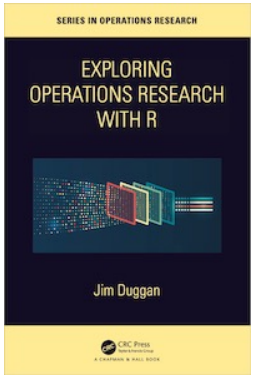
— John Chambers (Chambers, 2008)

Exploring Operations Research with R:

A Workshop

01 - Introduction

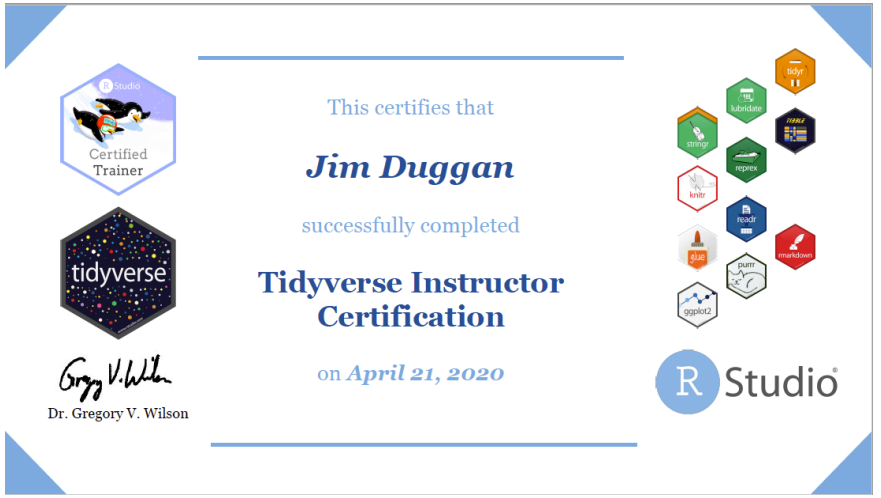




https://github.com/JimDuggan/Exploring-OR-with-R-Workshop

Instructor –Jim Duggan

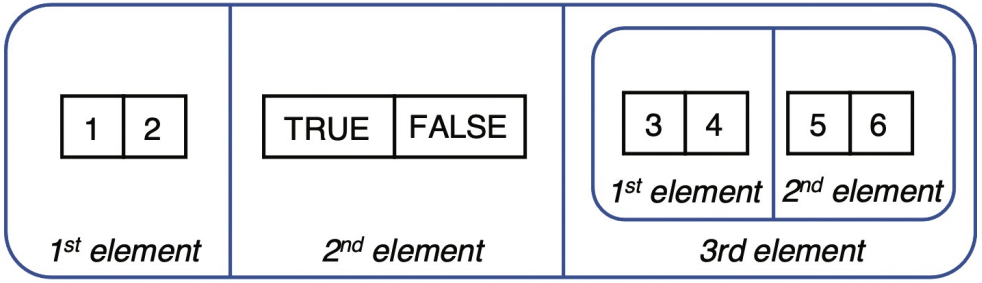
- Lectures in
  - Programming (R, MATLAB, Python),
  - Modelling & Simulation
- Research interests:
  - System Dynamics
  - Public Health
  - Data Science

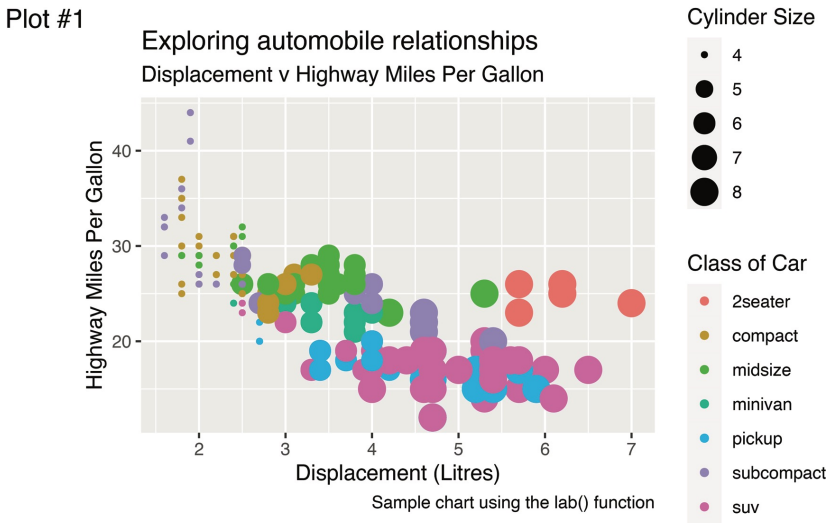


https://education.rstudio.com/trainers/people/duggan+jim/

Goal

- Show how the R programming language can be a valuable tool – and way of thinking – which can be successfully applied to the field of operations research (OR).
- Core R tools (tidyverse)
- Application areas to OR:
  - Exploratory Data Analysis
  - Linear Programming
  - Agent-Based Simulation
  - System Dynamics



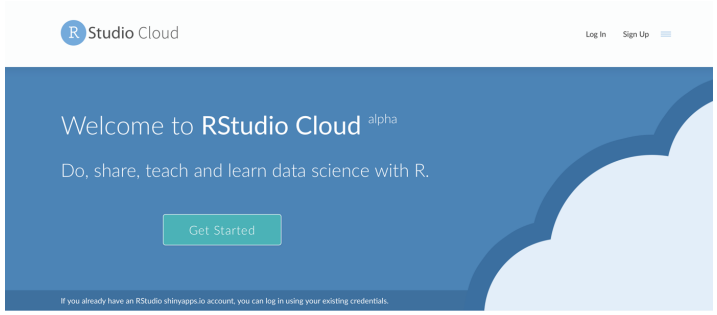


Topics Overview

Topic	Description	~Timing
1	Introduction to R and Posit Cloud	10 mins
2	Storing data using data frames / tibbles	10 mins
3	Visualisation with ggplot2	20 mins
4	Data transformation with dplyr	20 mins
5	Using R with OR – Four Examples	30 mins

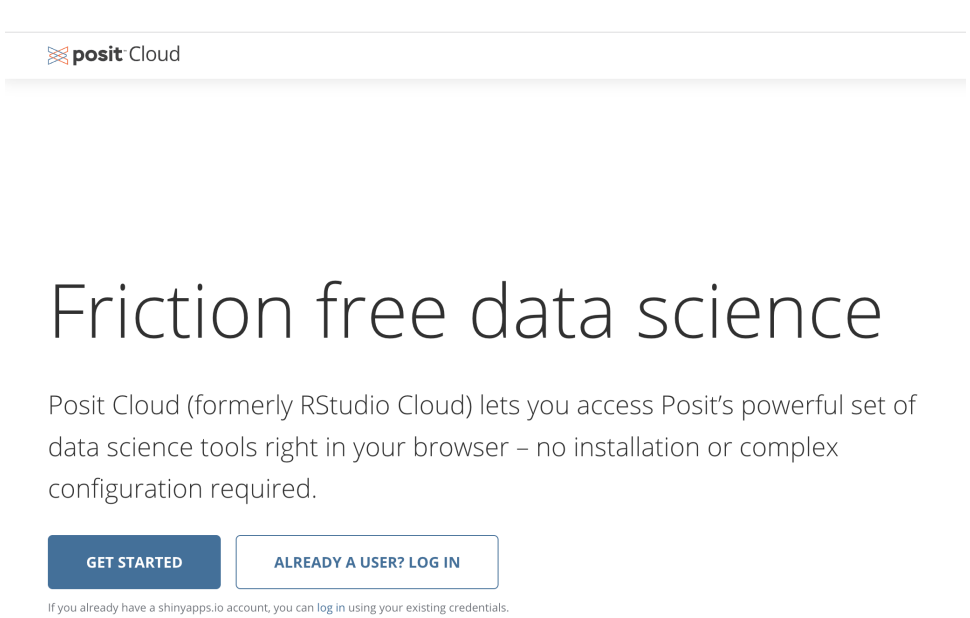
The R Project for Statistical Computing

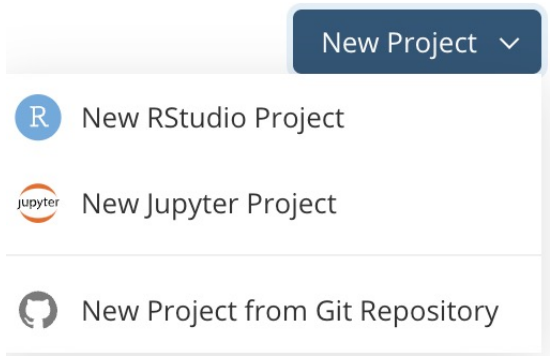
- R's *mission* is to enable the best and most thorough exploration of data possible (Chambers 2008).
- It is a dialect of the S language, developed at Bell Laboratories
- ACM noted that *S "will forever alter the way people analyze, visualize, and manipulate data"*

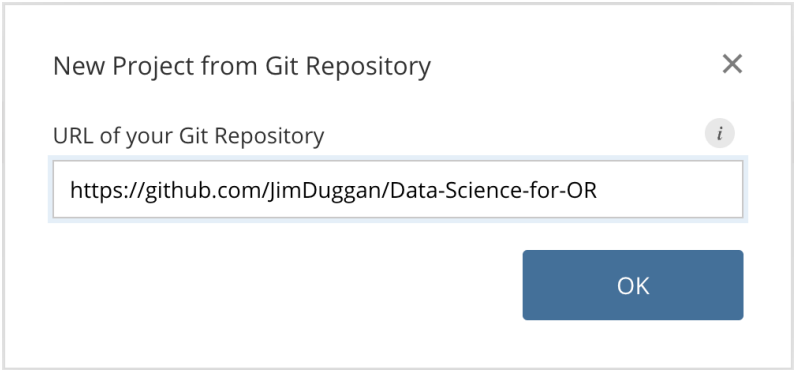



```
# We use this for processing the answer
# In programming, we "stand on the shoulders of giants"
library(stringi)

# This gets the input from the user.
# The result is stored in a variable
# Variables are important in programming!
name <- readline(prompt="Enter a name: ")

# We call a specially designed function to get the answer
# In R, we call functions all the time
# A function is a "mini-program"
ans <- stri_reverse(name)

# After all this work, we output the result
cat("The reverse of ", name, "is ===>", ans)
```

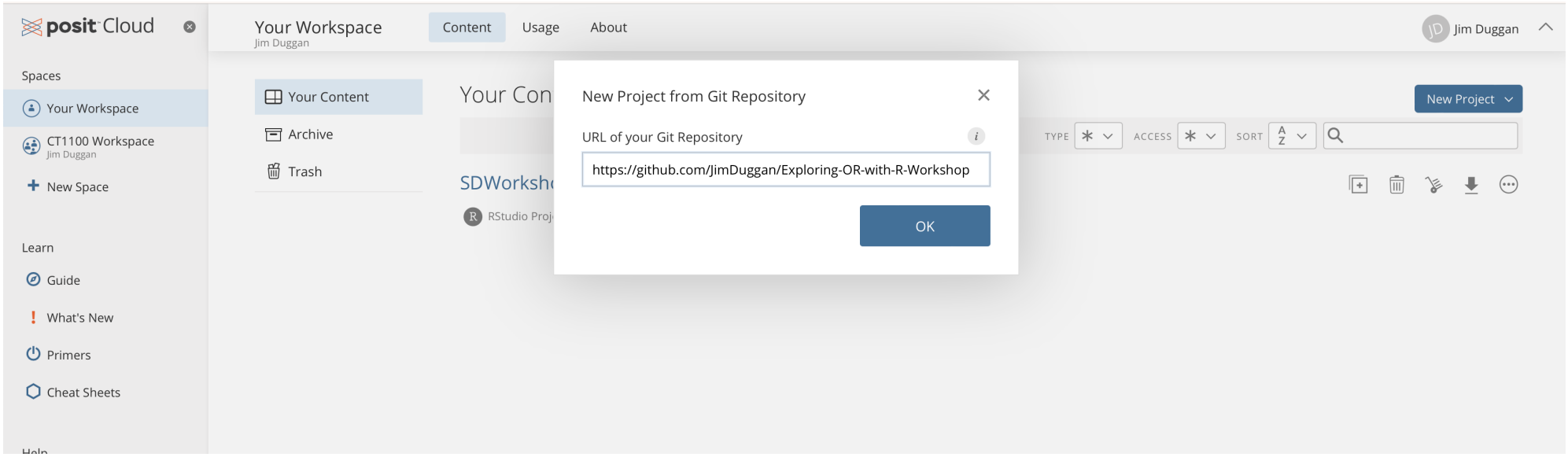
First Steps: Posit Cloud – Create Your Account



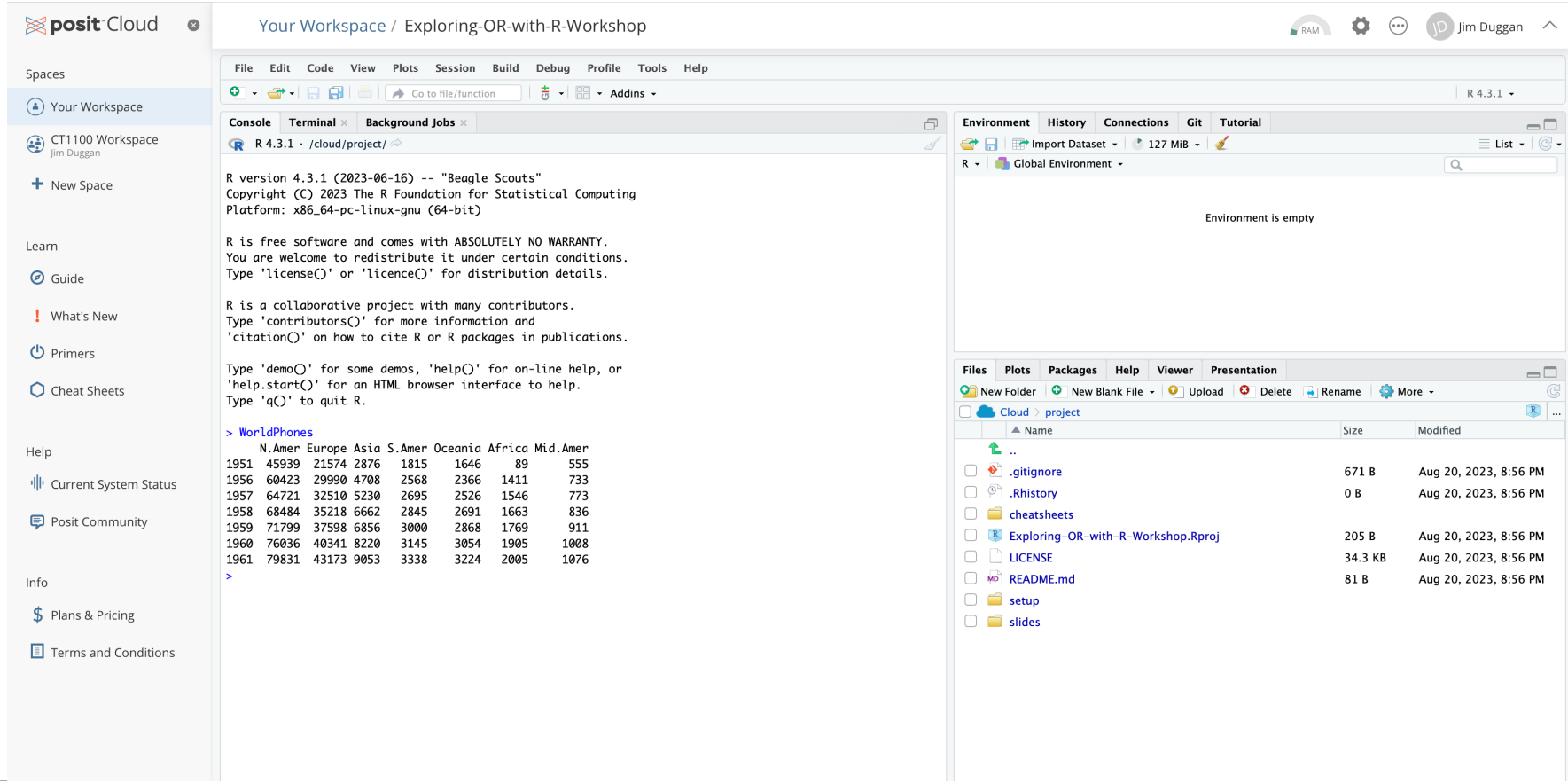




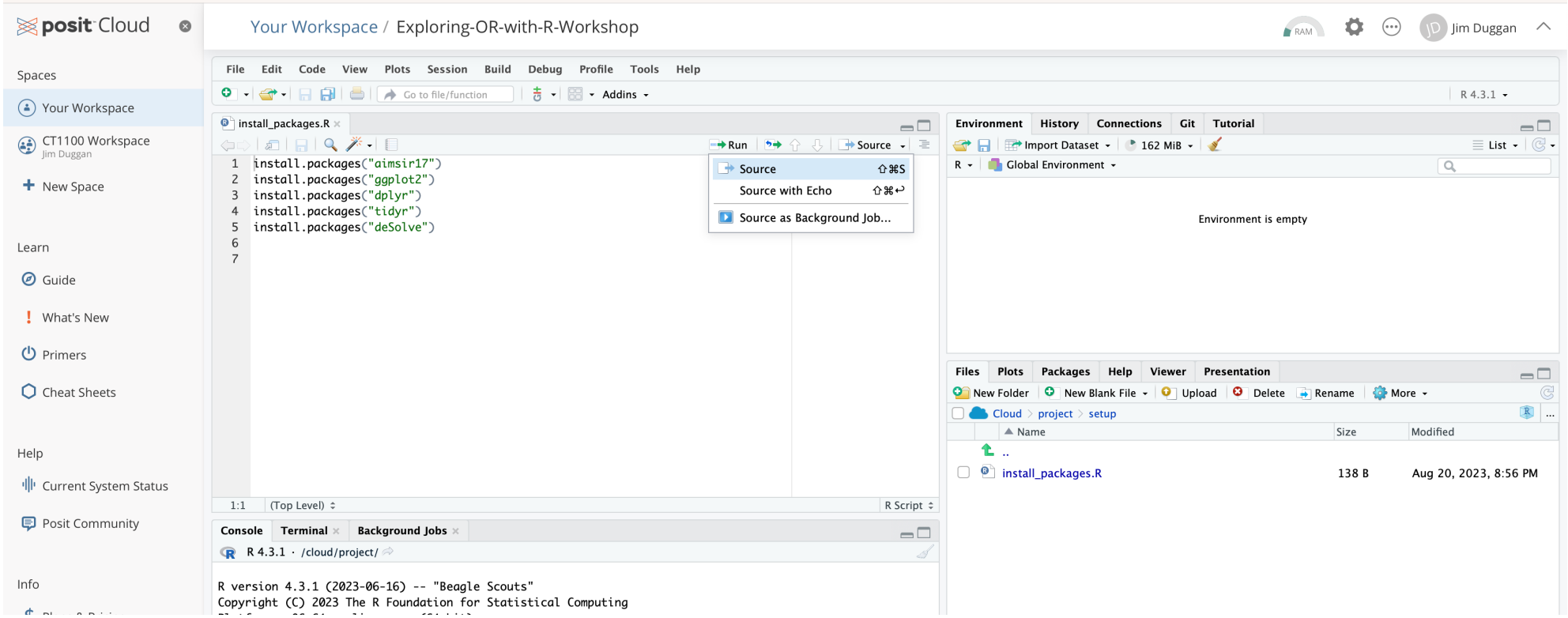
Install the codebase from github…



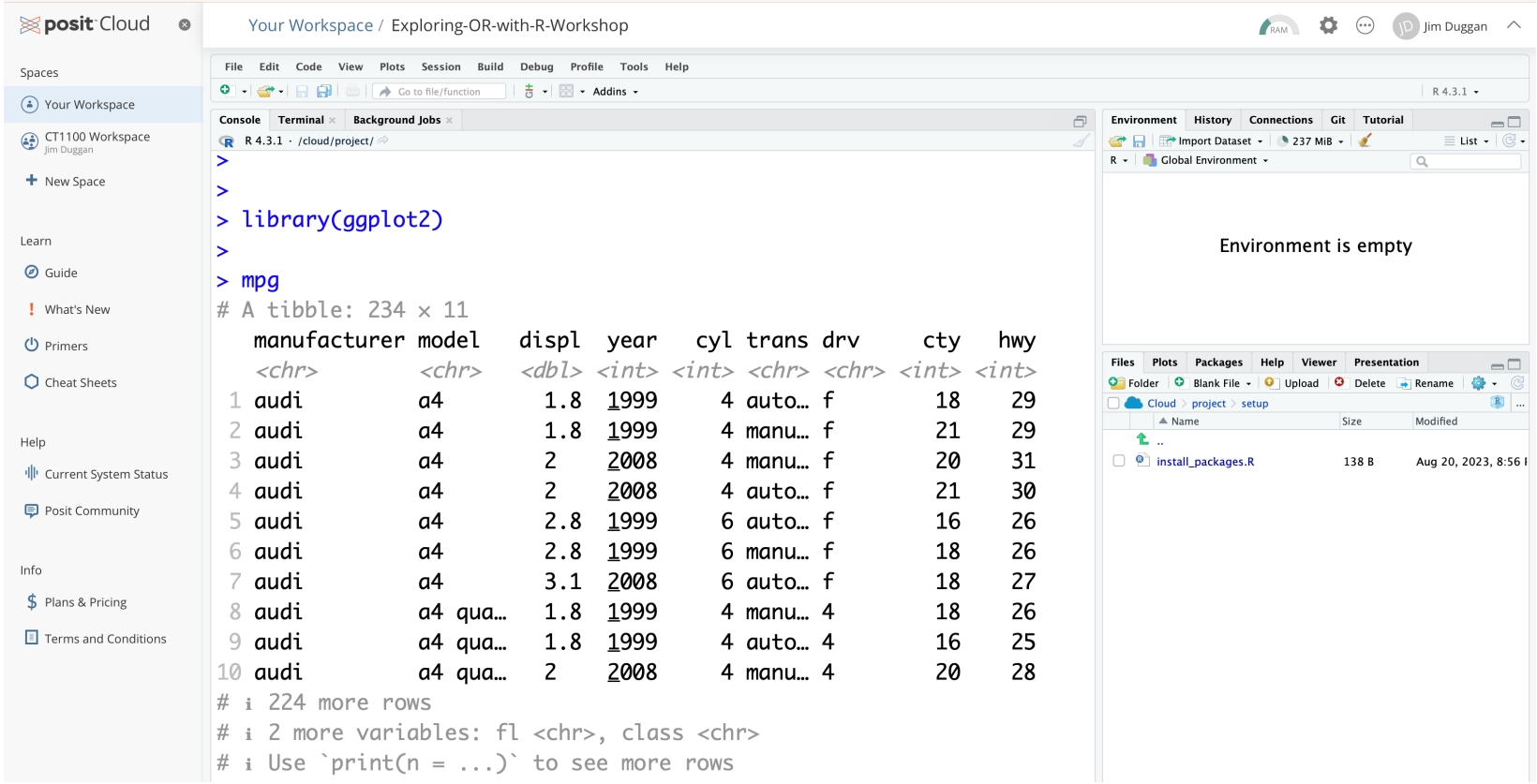
IDE Available for running scripts



Run setup file…



A quick check… the ggplot2 **mpg** dataset



> Sometimes data require more complex storage than simple vectors, and thankfully R provides a host of data structures. The most common are the data.frame, matrix and list, followed by the array.
>
> — Jared P. Lander (Lander, 2017)

Exploring Operations Research with R: *A Workshop*

02 - Tibbles

https://github.com/JimDuggan/Exploring-OR-with-R-Workshop

Topics Overview

Topic	Description	~Timing
1	Introduction to R and Posit Cloud	10 mins
2	Storing data using data frames / tibbles	10 mins
3	Visualisation with ggplot2	20 mins
4	Data transformation with dplyr	20 mins
5	Using R with OR – Four Examples	30 mins

Data Frames/Tibbles

- The most common way of storing data in R
- A two-dimensional structure, with rows (observations) and columns (variables)

```
> mpg
# A tibble: 234 × 11
   manufacturer model   displ  year   cyl trans drv     cty   hwy
   <chr>        <chr>   <dbl> <int> <int> <chr> <chr> <int> <int>
 1 audi         a4        1.8  1999     4 auto… f        18    29
 2 audi         a4        1.8  1999     4 manu… f        21    29
 3 audi         a4        2    2008     4 manu… f        20    31
 4 audi         a4        2    2008     4 auto… f        21    30
 5 audi         a4        2.8  1999     6 auto… f        16    26
 6 audi         a4        2.8  1999     6 manu… f        18    26
 7 audi         a4        3.1  2008     6 auto… f        18    27
 8 audi         a4 qua…   1.8  1999     4 manu… 4        18    26
 9 audi         a4 qua…   1.8  1999     4 auto… 4        16    25
10 audi         a4 qua…   2    2008     4 manu… 4        20    28
# ℹ 224 more rows
# ℹ 2 more variables: fl <chr>, class <chr>
# ℹ Use `print(n = ...)` to see more rows
```

tibble abbreviations (variable types)

Abbreviation	Data Type
int	integers
dbl	doubles (real numbers)
chr	character vectors (strings
dttm	date-times
lgl	logical
fctr	factor (categorical variables with fixed possible values)
date	dates

In the computer's memory, just like a spreadsheet

Your Workspace / Exploring-OR-with-R-Workshop

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

mpg

Filter

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
11	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
14	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact

Showing 1 to 14 of 234 entries, 11 total columns

Console | Terminal | Background Jobs

R 4.3.1 · /cloud/project/

```
> View(mpg)
```

The summary() function

```
> summary(mpg)
 manufacturer          model               displ            year          cyl           trans
 Length:234         Length:234         Min.   :1.600   Min.   :1999   Min.   :4.000   Length:234
 Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999   1st Qu.:4.000   Class :character
 Mode  :character   Mode  :character   Median :3.300   Median :2004   Median :6.000   Mode  :character
                                       Mean   :3.472   Mean   :2004   Mean   :5.889
                                       3rd Qu.:4.600   3rd Qu.:2008   3rd Qu.:8.000
                                       Max.   :7.000   Max.   :2008   Max.   :8.000
     drv                 cty              hwy              fl                class
 Length:234         Min.   : 9.00   Min.   :12.00   Length:234         Length:234
 Class :character   1st Qu.:14.00   1st Qu.:18.00   Class :character   Class :character
 Mode  :character   Median :17.00   Median :24.00   Mode  :character   Mode  :character
                    Mean   :16.86   Mean   :23.44
                    3rd Qu.:19.00   3rd Qu.:27.00
                    Max.   :35.00   Max.   :44.00
```

Challenge 2.1

- Explore some of the following tibbles in R
  - mpg (ggplot2)
  - diamonds (ggplot2)
  - observations (aimsir17)
  - eirgrid17 (aimsir17)
  - stations (aimsir17)

Practically, `ggplot2` provides beautiful, hassle-free plots that take care of fiddly details like drawing legends. The plots can be built up iteratively and edited later.

— Hadley Wickham (Wickham, 2016)

Exploring Operations Research with R: *A Workshop*

03 – ggplot2

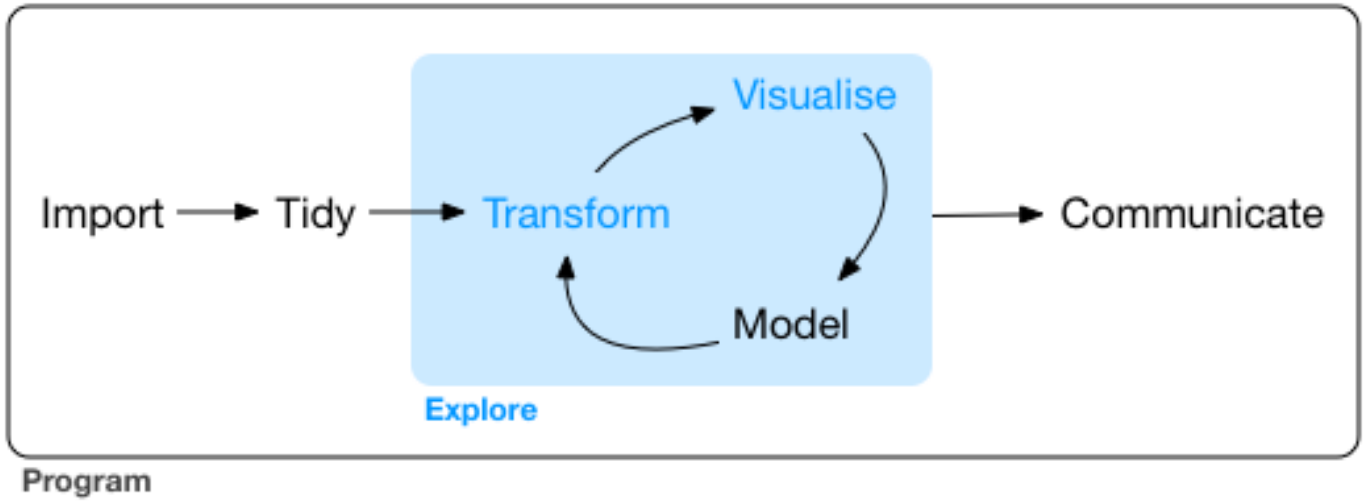
https://github.com/JimDuggan/Exploring-OR-with-R-Workshop

Topics Overview

Topic	Description	~Timing
1	Introduction to R and Posit Cloud	10 mins
2	Storing data using data frames / tibbles	10 mins
3	Visualisation with ggplot2	20 mins
4	Data transformation with dplyr	20 mins
5	Using R with OR – Four Examples	30 mins

Data Exploration

"Data exploration is the art of looking at your data, rapidly generating hypotheses, quickly testing them, then repeating again and again and again." (Wickham and Grolemund 2017).



ggplot2

- Layered grammar of graphics, enables us to concisely describe the components of a graphic.
- Benefits:
  - plots can be designed in a layered manner (+ operator)
  - a wide range of plots can be generated to support decision analysis, including scatterplots, histograms and time series charts
  - charts can be developed rapidly, and this support an iterative process of decision support

ggplot2::mpg, N = 234

Variable	Description
manufacturer	Manufacturer name
model	Model name
displ	Engine displacement (liters)
year	Year of manufacture
cyl	Number of cylinders
trans	Type of transmission
drv	Type of drive train (e.g. front wheel)
cty	City miles per gallon
hwy	Highway miles per gallon
fl	Fuel type
class	"type" of car (e.g. "compact")

Data Visualisation with **ggplot2**

"The simple graph has brought more information to the data analyst's mind that any other device." – John Tukey

```
> library(ggplot2)
>
> mpg
# A tibble: 234 × 11
   manufacturer model      displ  year   cyl trans      drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18    29 p     compact
 2 audi         a4           1.8  1999     4 manual(m5) f        21    29 p     compact
 3 audi         a4           2    2008     4 manual(m6) f        20    31 p     compact
 4 audi         a4           2    2008     4 auto(av)   f        21    30 p     compact
 5 audi         a4           2.8  1999     6 auto(l5)   f        16    26 p     compact
 6 audi         a4           2.8  1999     6 manual(m5) f        18    26 p     compact
 7 audi         a4           3.1  2008     6 auto(av)   f        18    27 p     compact
 8 audi         a4 quattro   1.8  1999     4 manual(m5) 4        18    26 p     compact
 9 audi         a4 quattro   1.8  1999     4 auto(l5)   4        16    25 p     compact
10 audi         a4 quattro   2    2008     4 manual(m6) 4        20    28 p     compact
# … with 224 more rows
# ℹ Use `print(n = ...)` to see more rows
```

First Steps

- Generate a first graph to help answer the following question:
  - *Do cars with big engines use more fuel than cars with small engines*
- What might the relationship between engine size and fuel efficiency look like?
- Need to draw a scatter plot

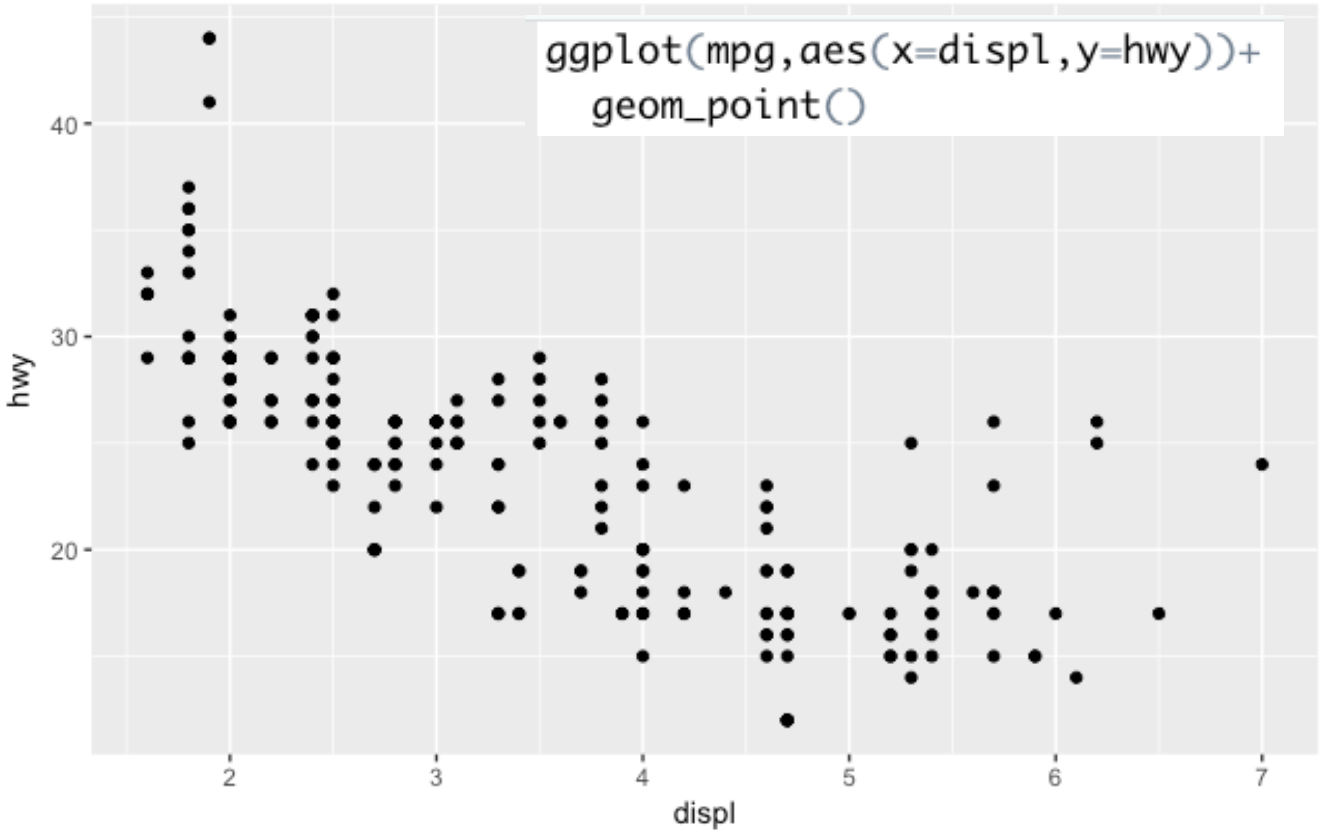
```
> library(ggplot2)
>
> mpg
# A tibble: 234 × 11
   manufacturer model      displ  year   cyl trans      drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18    29 p     compact
 2 audi         a4           1.8  1999     4 manual(m5) f        21    29 p     compact
 3 audi         a4           2    2008     4 manual(m6) f        20    31 p     compact
 4 audi         a4           2    2008     4 auto(av)   f        21    30 p     compact
 5 audi         a4           2.8  1999     6 auto(l5)   f        16    26 p     compact
 6 audi         a4           2.8  1999     6 manual(m5) f        18    26 p     compact
 7 audi         a4           3.1  2008     6 auto(av)   f        18    27 p     compact
 8 audi         a4 quattro   1.8  1999     4 manual(m5) 4        18    26 p     compact
 9 audi         a4 quattro   1.8  1999     4 auto(l5)   4        16    25 p     compact
10 audi         a4 quattro   2    2008     4 manual(m6) 4        20    28 p     compact
# … with 224 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Selecting data (from columns)

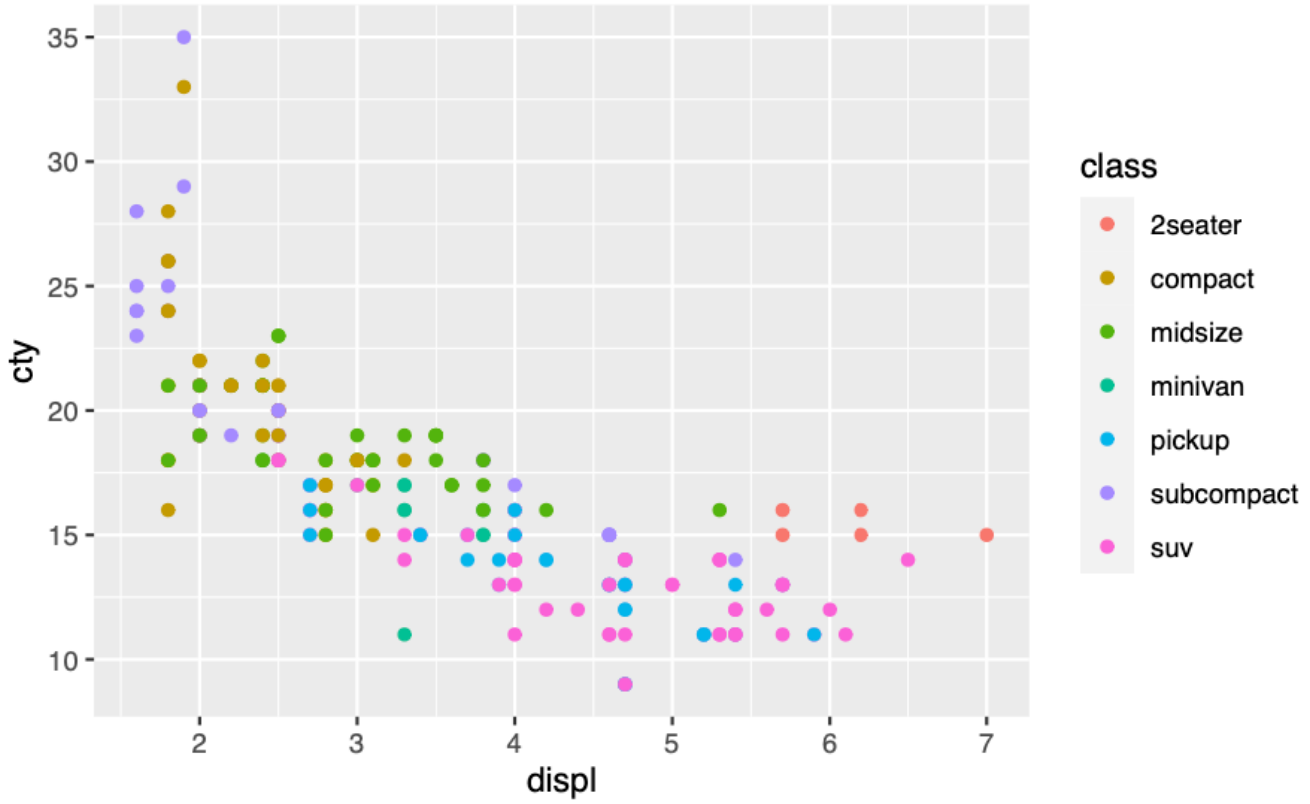
```
> mpg
# A tibble: 234 × 11
   manufacturer model      displ  year   cyl trans      drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18    29 p     compact
 2 audi         a4           1.8  1999     4 manual(m5) f        21    29 p     compact
 3 audi         a4           2    2008     4 manual(m6) f        20    31 p     compact
 4 audi         a4           2    2008     4 auto(av)   f        21    30 p     compact
 5 audi         a4           2.8  1999     6 auto(l5)   f        16    26 p     compact
 6 audi         a4           2.8  1999     6 manual(m5) f        18    26 p     compact
 7 audi         a4           3.1  2008     6 auto(av)   f        18    27 p     compact
 8 audi         a4 quattro   1.8  1999     4 manual(m5) 4        18    26 p     compact
 9 audi         a4 quattro   1.8  1999     4 auto(l5)   4        16    25 p     compact
10 audi         a4 quattro   2    2008     4 manual(m6) 4        20    28 p     compact
# … with 224 more rows
# ℹ Use `print(n = ...)` to see more rows
```

- Among the variables are:
  - **displ**, a car's engine size in litres
  - **hwy**, a car's fuel efficiency on the highway in miles per gallon

Creating a ggplot



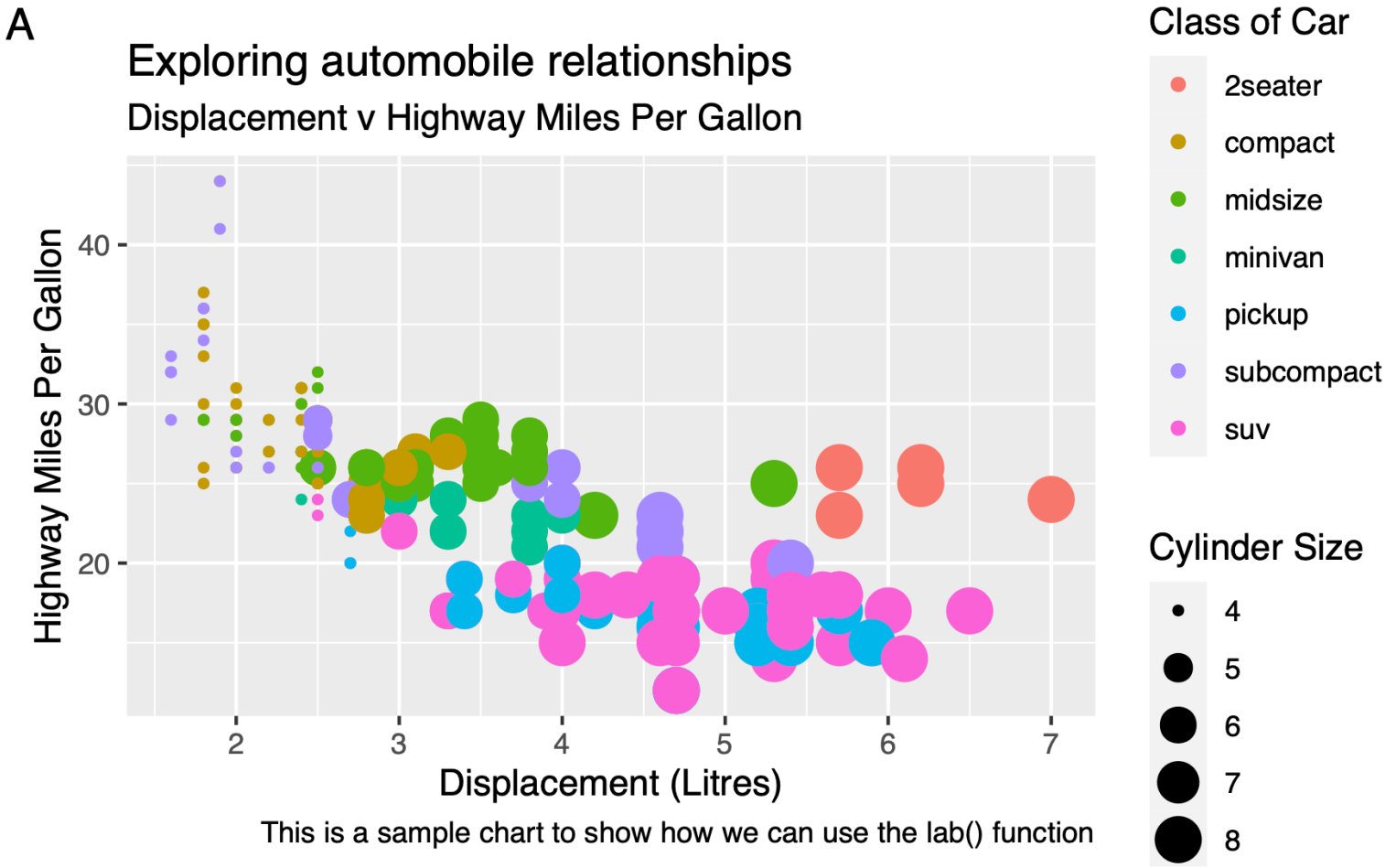
```
ggplot(data=mpg,mapping=aes(x=displ,y=cty,colour=class))+
  geom_point()
```



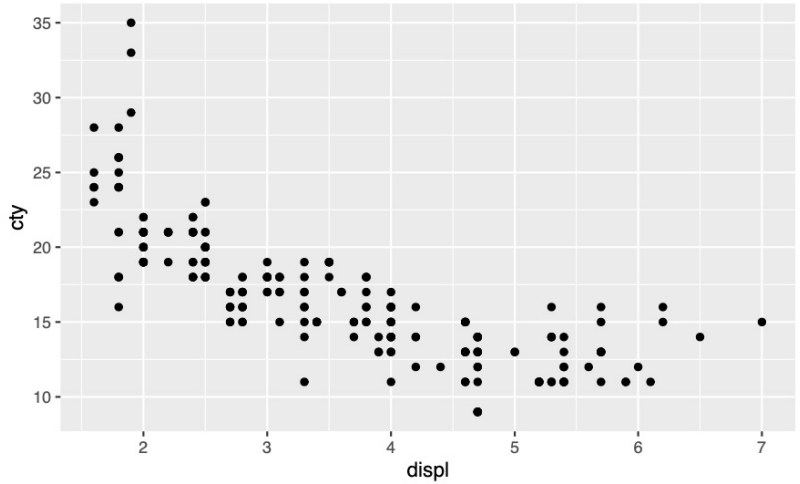
Customising Plot Appearance with lab()

Argument	Role
title	provides an overall title text for the plot
subtitle	adds a subtitle text
colour	allows you to specify the legend name for the colour attribute
caption	inserts text on the lower right hand side of your plot
size	allows you to name the size attribute
x	name the x-axis
y	name the y-axis
tag	text for tag label for top-left of plot

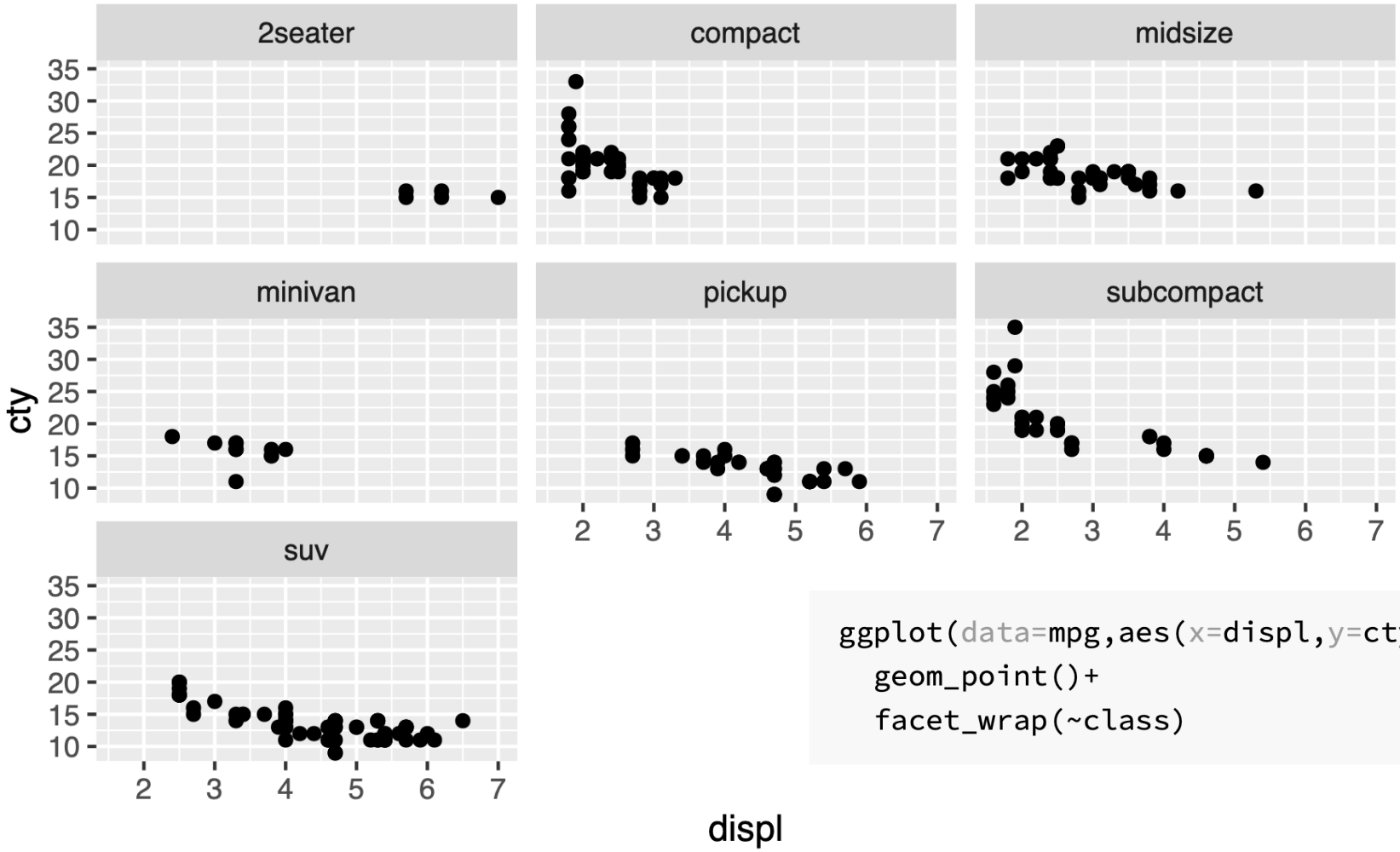
```
p1 <- ggplot(data=mpg,aes(x=displ,y=hwy,size=cyl,colour=class))+
        geom_point()

p1 <- p1 +
       labs(
         title = "Exploring automobile relationships",
         subtitle = "Displacement v Highway Miles Per Gallon",
         colour = "Class of Car",
         size = "Cylinder Size",
         caption = "This is a sample chart to show how we can use the lab() function",
         tag = "A",
         x = "Displacement (Litres)",
         y = "Highway Miles Per Gallon"
       )

p1
```


A
Exploring automobile relationships
Displacement v Highway Miles Per Gallon
Highway Miles Per Gallon
40
30
20
2
3
4
5
6
7
Displacement (Litres)
This is a sample chart to show how we can use the lab() function
Class of Car
2seater
compact
midsize
minivan
pickup
subcompact
suv
Cylinder Size
4
5
6
7
8

Subplots with Facets



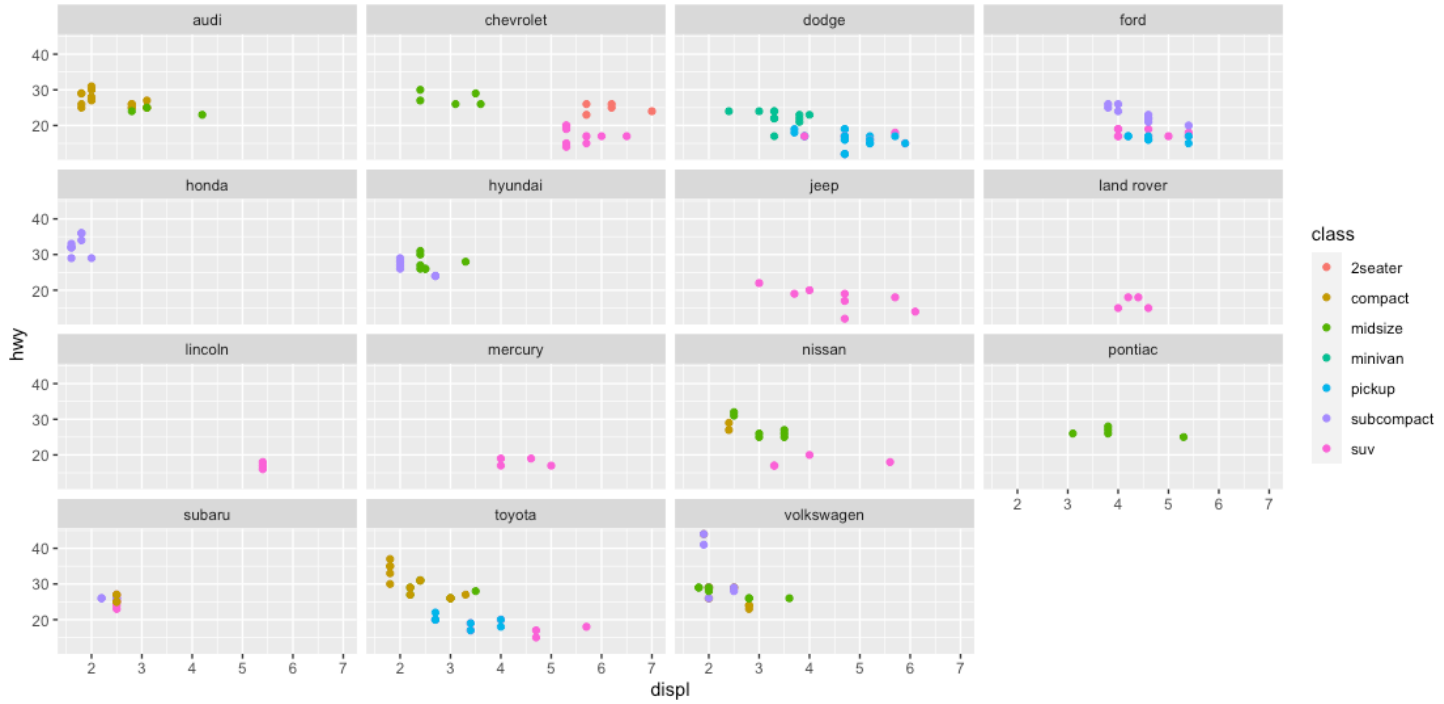

```
ggplot(data=mpg,aes(x=displ,y=cty))+
  geom_point()+
  facet_wrap(~class)
```




```
ggplot(data=mpg,aes(x=displ,y=cty))+
  geom_point()+
  facet_wrap(~class)
```

Challenge 3.1

- Generate the following scatter plot (displ v hwy) from the tibble mpg (ggplot2 library)



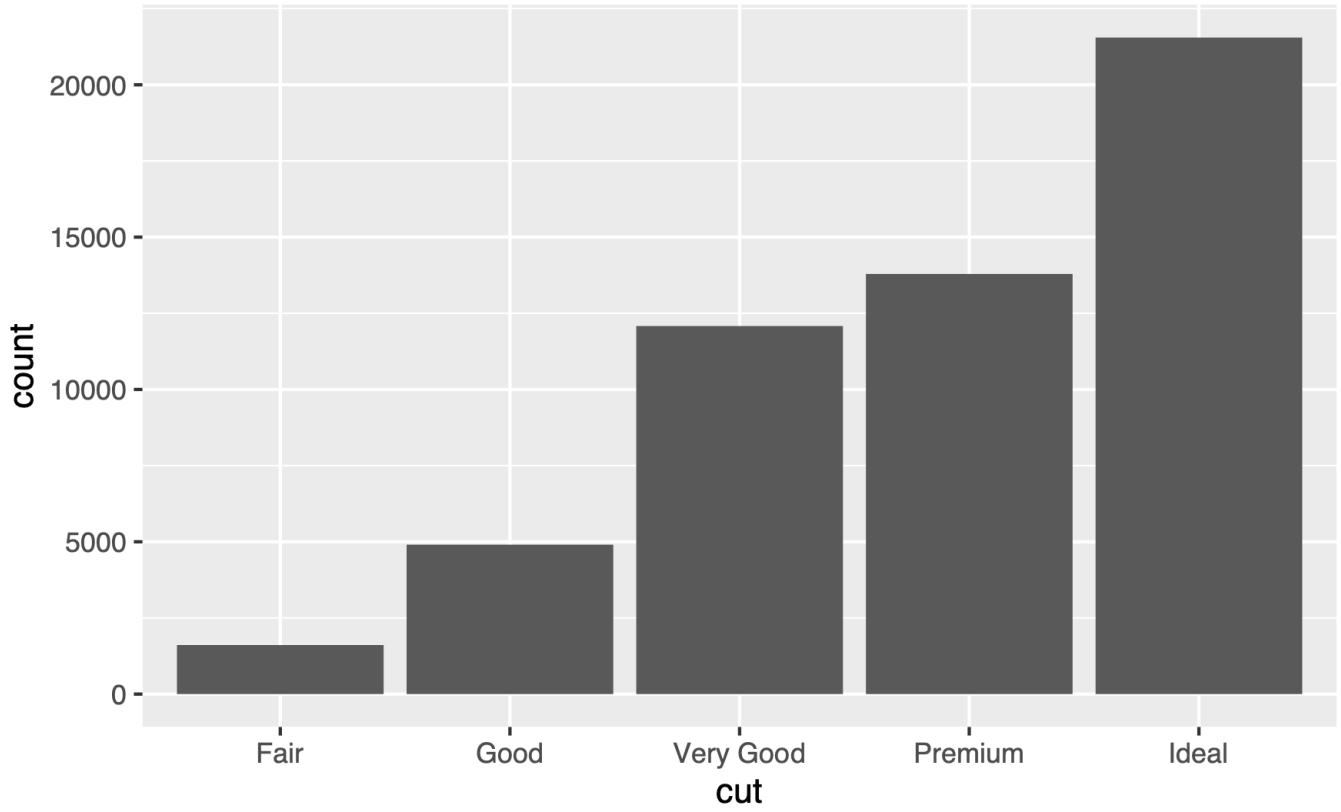
Statistical Transformations

- Many graphs, like scatterplots, plot the raw values of the dataset
- However, other graphs (e.g. bar charts) *calculate new values to plot*
  - **Bar charts, histograms and frequency polygons** bin your data and plot bin counts, the number of points that fall in each bin
  - **Smoothers** fit a model to your data and the plot predictions from the model
  - **Boxplots** compute a robust summary of the distribution and display a specially formatted box

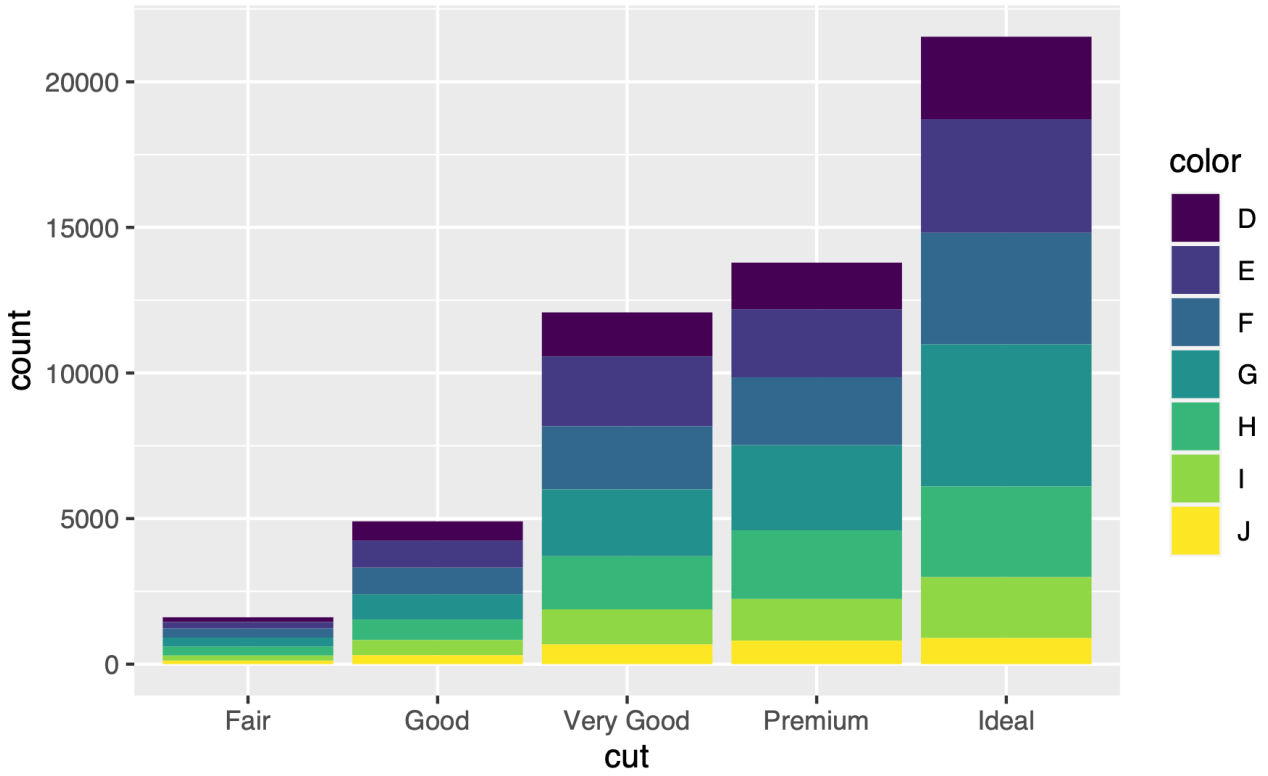
ggplot2::diamonds, N = 53,940

Variable	Description
carat	Weight of the diamond
cut	Quality of the cut (categorical)
color	Diamond colour (categorical)
clarity	Diamond clarity (categorical)
depth	Total depth percentage
table	Measure related to width of diamond top
price	Price in dollars
x	Length in mm
y	Width in mm
z	Depth in mm

```
ggplot(data=diamonds,mapping=aes(x=cut))+
  geom_bar()
```

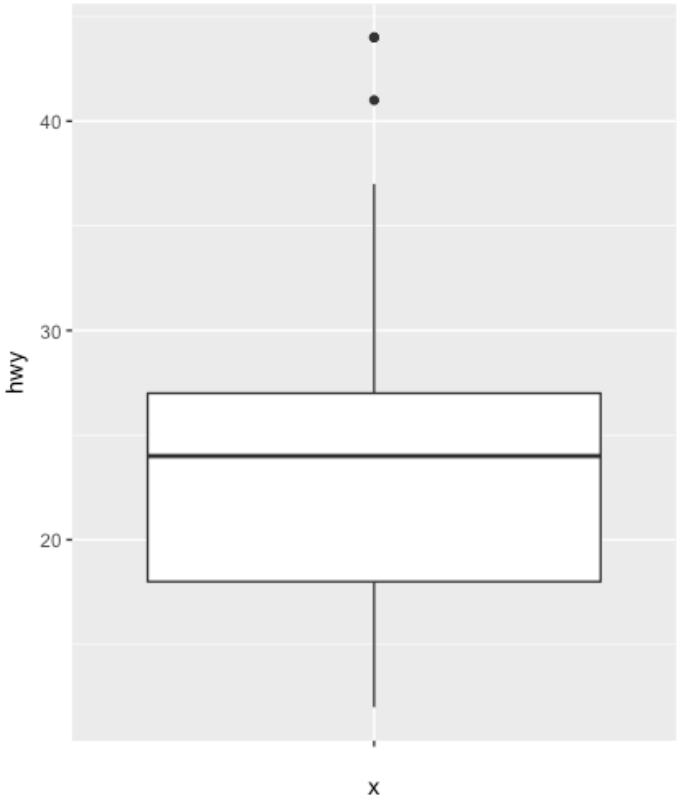


```
ggplot(data=diamonds,mapping=aes(x=cut,fill=color))+
  geom_bar()
```

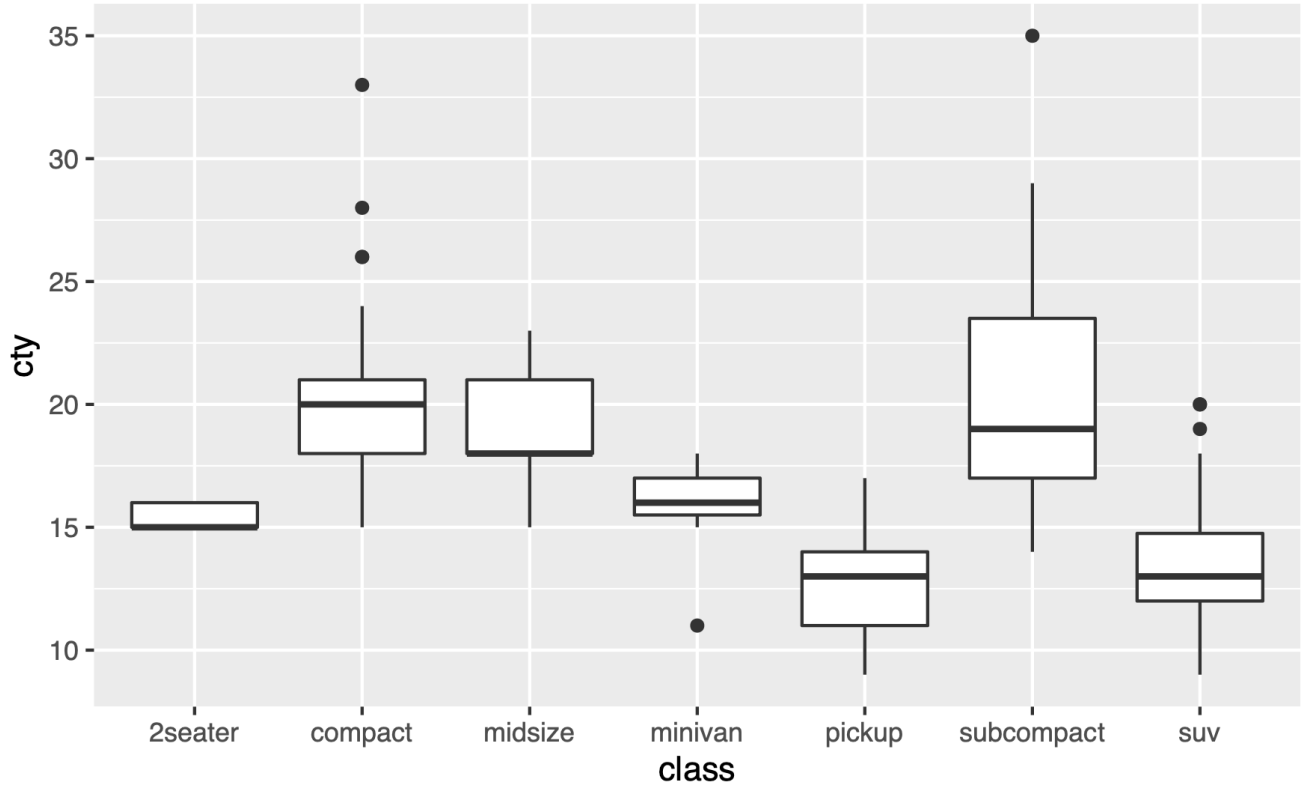


Boxplot

- Display the distribution of a continuous variable broken down by a categorical variable
- Box that stretches from the 25th to 75th percentile a distance known as the interquartile range (IRQ)
- Median in the middle of box
- Points outside more that 1.5 times the IQR from either edge of the box are displayed (outliers)
- Whisker extends to the farthest non-outlier point in the distribution

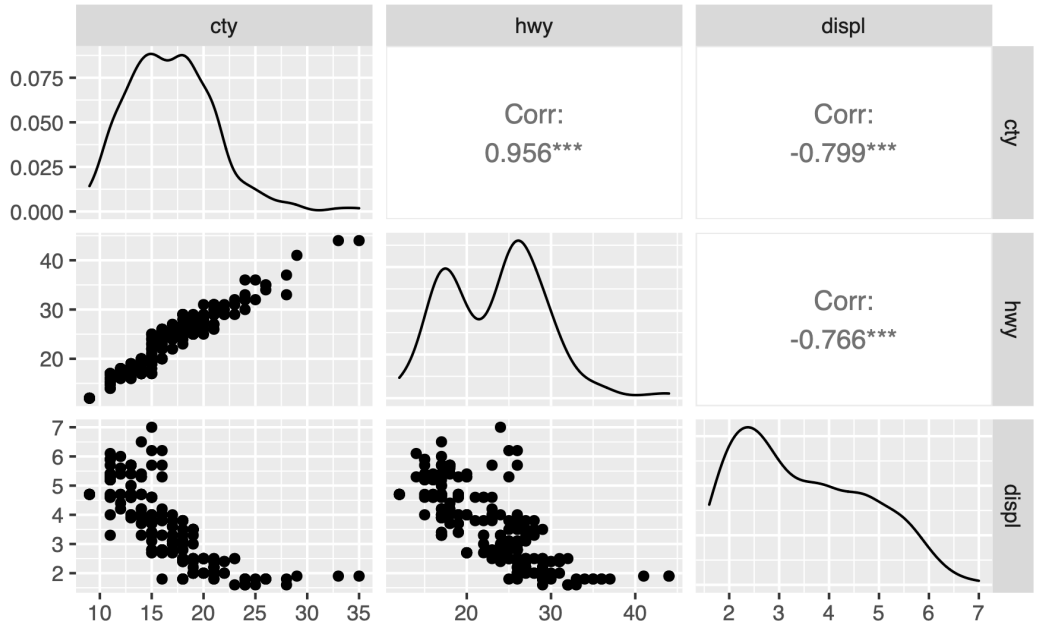


```
ggplot(data=mpg,mapping=aes(y=cty,x=class))+
  geom_boxplot()
```



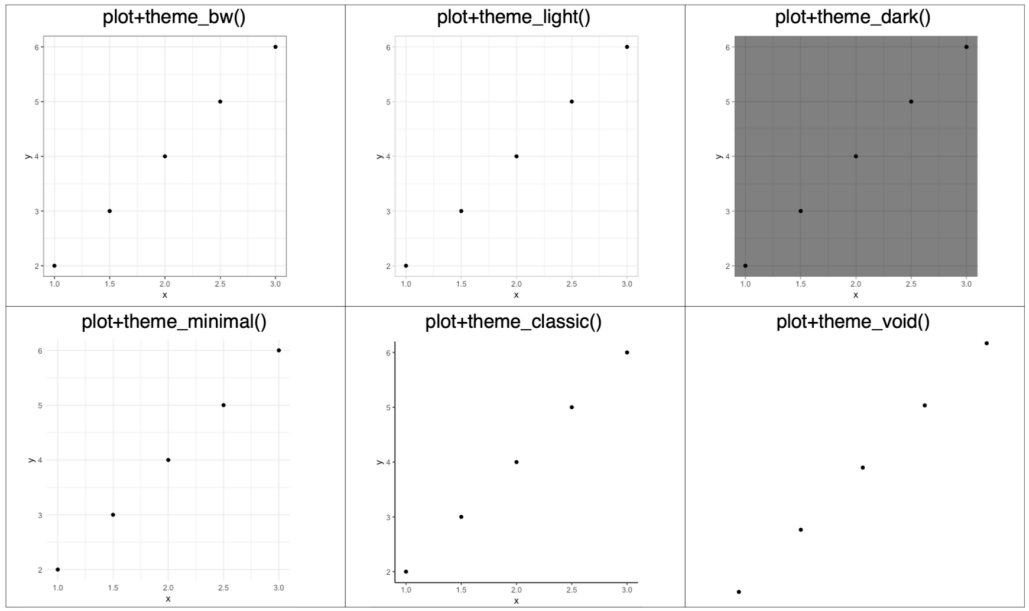
Covariation with ggpairs()

```
library(GGally)
my_vars <- subset(mpg,select=c(cty,hwy,displ))
ggpairs(my_vars)
```



Ready to use themes

```
d <- tibble(x=seq(1,3,by=0.5),y=2*x)
plot <- ggplot(data=d,mapping=aes(x=x,y=y))+
          geom_point()
```



https://bbc.github.io/rcookbook/

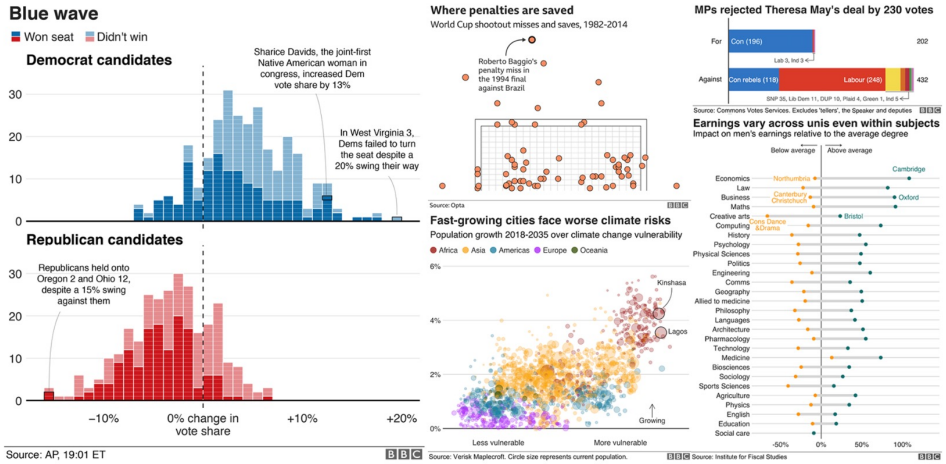
BBC Visual and Data Journalism cookbook for R graphics

Last updated: 2019-01-24

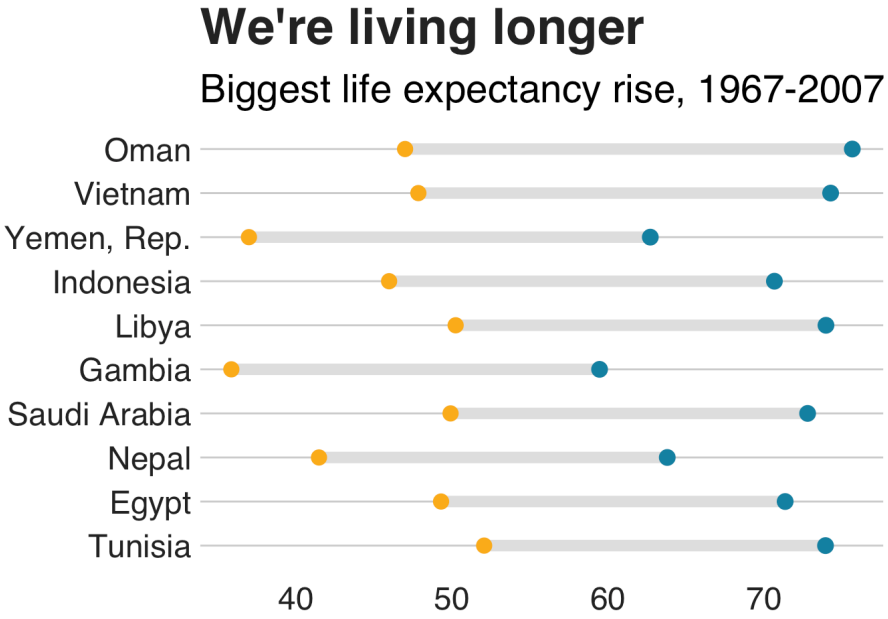
How to create BBC style graphics

At the BBC data team, we have developed an R package and an R cookbook to make the process of creating publication-ready graphics in our in-house style using R's ggplot2 library a more reproducible process, as well as making it easier for people new to R to create graphics.

The cookbook below should hopefully help anyone who wants to make graphics like these:



We'll get to how you can put together the various elements of these graphics, but **let's get the admin out of the way first…**

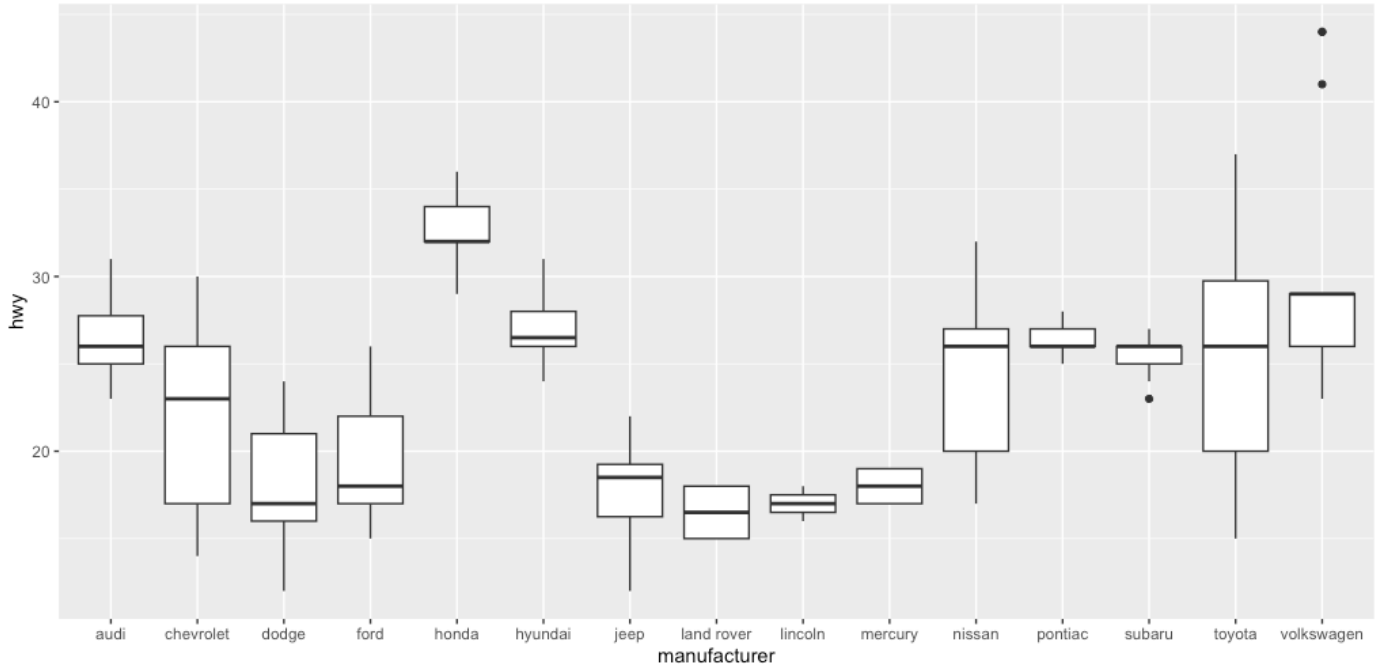


Useful geoms

Geom	Purpose
geom_smooth()	Fits a smoother to data and displays the smooth and its standard error
geom_boxplot()	Produces a box-and-whisker plot to summarise the distribution of a set of points
geom_histogram() geom_freqpoly()	Shows the distribution of continuous variables
geom_bar()	Shows the distribution of categorical variables
geom_path() geom_line()	Draws lines between data points
geom_area()	Draws an area plot, which is a line plot filled to the y-axis. Multiple groups will be stacked upon each other
geom_rect() geom_tile() geom_raster()	Draw rectangles
geom_polygon()	Draws polygons, which are filled paths.

Challenge 3.2

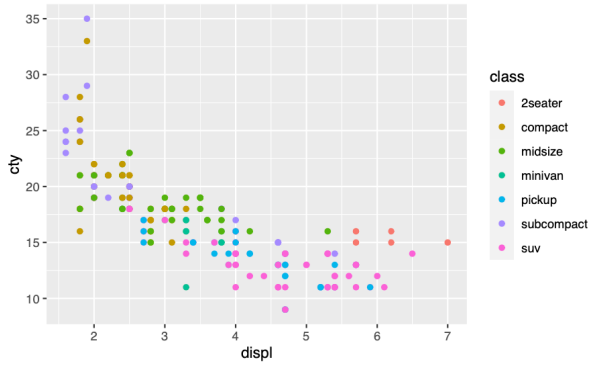
- Generate the following box plot (hwy by manufacturer) from mpg.

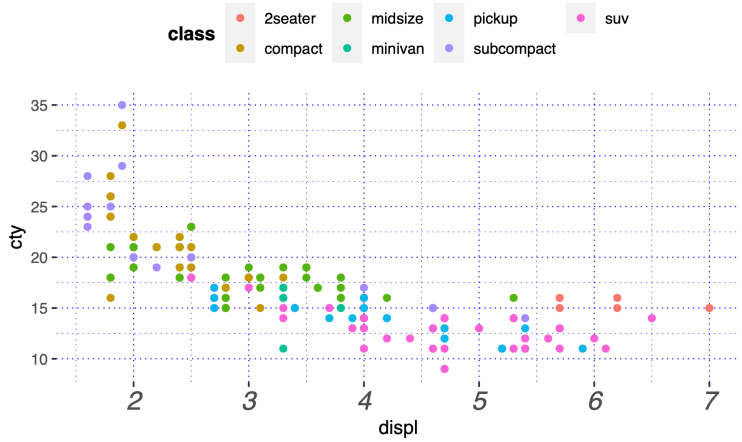


Summary

- Layered grammar of graphics, enables us to concisely describe the components of a graphic.
- Benefits:
  - plots can be designed in a layered manner (+ operator)
  - a wide range of plots can be generated to support decision analysis, including scatterplots, histograms and time series charts
  - charts can be developed rapidly, and this support an iterative process of decision support

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





> Writing code with `dplyr` imvolves using the "grammar of data" to peform data munging. Each step is done by a single function that represents a verb.
>
> — Jared P. Lander (Lander, 2017)

Exploring Operations Research with R: *A Workshop*

04 – dplyr

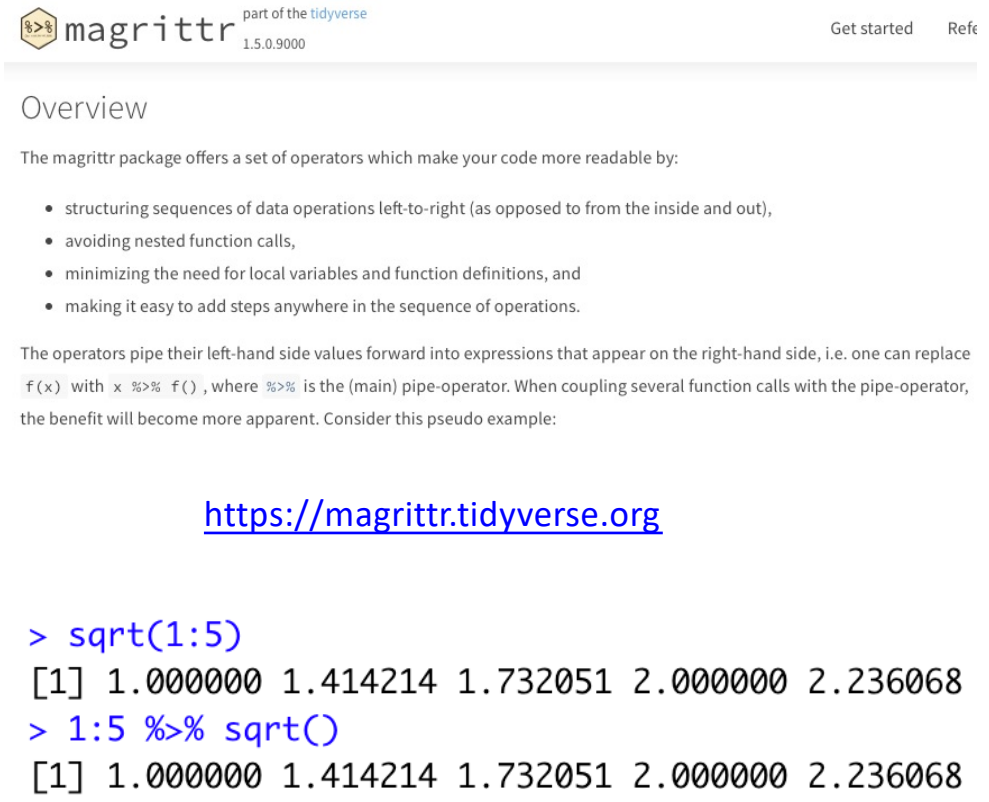
https://github.com/JimDuggan/Exploring-OR-with-R-Workshop

Topics Overview

Topic	Description	~Timing
1	Introduction to R and Posit Cloud	10 mins
2	Storing data using data frames / tibbles	10 mins
3	Visualisation with ggplot2	20 mins
4	Data transformation with dplyr	20 mins
5	Using R with OR – Four Examples	30 mins

Combining operations with the Pipe

- The pipe %>% comes from the magrittr package (Stefan Milton Bache)
- Helps to write code that is easier to read and understand
- x %>% f(y) turns into f(x, y)
- x %>% f(y) %>% g(z) turns into g(f(x, y),z)




```
> sqrt(1:5)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> 1:5 %>% sqrt()
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```r
set.seed(100)
arr   <- rpois(12,50)
arr
#>  [1] 46 38 56 50 52 45 53 47 50 50 48 55
# The function sort takes a parameter to sort in descending order
s_arr <- sort(arr,decreasing = TRUE)
top_6 <- head(s_arr)
top_6
#> [1] 56 55 53 52 50 50
```

```r
set.seed(100)
top_6  <- rpois(12,50) %>%
            sort(decreasing = TRUE) %>%
            head()
top_6
#> [1] 56 55 53 52 50 50
```

Utility of dplyr

- Visualisation is an important tool for insight generation, but it's rare that you get the data in exactly the right form you need (Wickham and Grolemund 2017)
  - Create new variables
  - Create summaries
  - Order data
- **dplyr** package is designed for data transformation

dplyr Basics: 5 key functions

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	*Create new variables with functions of existing variables*
summarise()	*Collapse many values down to a single summary*

- ”A grammar of data manipulation”
- All verbs (functions) work similarly
  - The first argument is a data frame/tibble
  - The subsequent arguments decide what to do with the data frame/tibble
  - The result (data frame/tibble) supports chaining of steps – NOTE the “pipe operator” which we will cover later.

https://dplyr.tidyverse.org

1. filter() aimsir17 tibble observations

- First argument the name of the data frame
- Subsequent arguments are expressions that filter the data frame
- Subsequent arguments can be viewed as a succession of “and” statements
- Number of columns does not change
- Number of rows reduced (filtered)

```
> bel <- filter(observations,station=="BELMULLET")
> bel
# A tibble: 8,760 x 12
   station  year month   day  hour date                rain
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl>
 1 BELMUL… 2017     1     1     0 2017-01-01 00:00:00   0
 2 BELMUL… 2017     1     1     1 2017-01-01 01:00:00   0.5
 3 BELMUL… 2017     1     1     2 2017-01-01 02:00:00   0
 4 BELMUL… 2017     1     1     3 2017-01-01 03:00:00   0.4
 5 BELMUL… 2017     1     1     4 2017-01-01 04:00:00   0.6
 6 BELMUL… 2017     1     1     5 2017-01-01 05:00:00   0.1
 7 BELMUL… 2017     1     1     6 2017-01-01 06:00:00   0
 8 BELMUL… 2017     1     1     7 2017-01-01 07:00:00   0
 9 BELMUL… 2017     1     1     8 2017-01-01 08:00:00   0
10 BELMUL… 2017     1     1     9 2017-01-01 09:00:00   0
# … with 8,750 more rows, and 5 more variables: temp <dbl>,
#   rhum <dbl>, msl <dbl>, wdsp <dbl>, wddir <dbl>
```

Relational operators in R

Operators	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	not x
x | y	x OR y
x & y	x AND y

```
> bel <- filter(observations,station=="BELMULLET")
> bel
# A tibble: 8,760 x 12
   station  year month   day  hour date                rain
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl>
 1 BELMUL… 2017     1     1     0 2017-01-01 00:00:00   0
 2 BELMUL… 2017     1     1     1 2017-01-01 01:00:00   0.5
 3 BELMUL… 2017     1     1     2 2017-01-01 02:00:00   0
 4 BELMUL… 2017     1     1     3 2017-01-01 03:00:00   0.4
 5 BELMUL… 2017     1     1     4 2017-01-01 04:00:00   0.6
 6 BELMUL… 2017     1     1     5 2017-01-01 05:00:00   0.1
 7 BELMUL… 2017     1     1     6 2017-01-01 06:00:00   0
 8 BELMUL… 2017     1     1     7 2017-01-01 07:00:00   0
 9 BELMUL… 2017     1     1     8 2017-01-01 08:00:00   0
10 BELMUL… 2017     1     1     9 2017-01-01 09:00:00   0
# … with 8,750 more rows, and 5 more variables: temp <dbl>,
#   rhum <dbl>, msl <dbl>, wdsp <dbl>, wddir <dbl>
```

Show rows for “MACE HEAD” in January

```
> mhj <- filter(observations,station=="MACE HEAD",month==1)
>
> mhj
# A tibble: 744 x 12
   station    year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>     <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 MACE HEAD  2017     1     1     0 2017-01-01 00:00:00   0.5   5.6    88 1023.    17   340
 2 MACE HEAD  2017     1     1     1 2017-01-01 01:00:00   0     5.4    84 1023.    17   340
 3 MACE HEAD  2017     1     1     2 2017-01-01 02:00:00   0.1   4.7    87 1023.    14   340
 4 MACE HEAD  2017     1     1     3 2017-01-01 03:00:00   0     4.7    81 1023.    15   350
 5 MACE HEAD  2017     1     1     4 2017-01-01 04:00:00   0     4.5    80 1024.    12   350
 6 MACE HEAD  2017     1     1     5 2017-01-01 05:00:00   0     5      71 1024     13    20
 7 MACE HEAD  2017     1     1     6 2017-01-01 06:00:00   0     5.1    66 1024.    13    30
 8 MACE HEAD  2017     1     1     7 2017-01-01 07:00:00   0     4.8    76 1026.    19    10
 9 MACE HEAD  2017     1     1     8 2017-01-01 08:00:00   0.1   4.8    78 1026.    16   360
10 MACE HEAD  2017     1     1     9 2017-01-01 09:00:00   0.1   4.4    82 1027.    15    10
# … with 734 more rows
```

Useful approaches for filtering more than one value

- %in% operator in R

```
> filter(observations,station %in% c("ATHENRY","MACE HEAD"),month==1,day==1,hour==12)
# A tibble: 2 x 12
  station    year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
  <chr>     <dbl> <dbl> <int> <int> <dttm>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ATHENRY    2017     1     1    12 2017-01-01 12:00:00     0   5.1    75 1027.    11   360
2 MACE HEAD  2017     1     1    12 2017-01-01 12:00:00     0   6.7    67 1028.    16    20
>
> filter(observations,station == "ATHENRY" | station == "MACE HEAD",month==1,day==1,hour==12)
# A tibble: 2 x 12
  station    year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
  <chr>     <dbl> <dbl> <int> <int> <dttm>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ATHENRY    2017     1     1    12 2017-01-01 12:00:00     0   5.1    75 1027.    11   360
2 MACE HEAD  2017     1     1    12 2017-01-01 12:00:00     0   6.7    67 1028.    16    20
```

Challenge 4.1

- Show the weather for “ROCHES POINT” on October 16th at 12 midday. This should return just one observation.

2. arrange()

- Changes the order of rows.
- Used for sorting values
- Takes a tibble and a set of column names to order by

```
> arrange(observations,temp)
# A tibble: 219,000 x 12
   station     year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>      <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 CASEMENT    2017    12    11     4 2017-12-11 04:00:00     0  -6.2    91  989.     5   250
 2 GURTEEN     2017    12    11     3 2017-12-11 03:00:00     0  -6      94  989.     2   240
 3 GURTEEN     2017    12    11     4 2017-12-11 04:00:00     0  -6      95  990.     1   240
 4 GURTEEN     2017    12    11     1 2017-12-11 01:00:00     0  -5.9    92  988.     3   230
 5 GURTEEN     2017    12    11     5 2017-12-11 05:00:00     0  -5.8    95  990.     1   260
 6 GURTEEN     2017    12    11     0 2017-12-11 00:00:00     0  -5.7    94  988      2   280
 7 CASEMENT    2017    12    11     2 2017-12-11 02:00:00     0  -5.6    92  988.     4   230
 8 GURTEEN     2017    12    11     2 2017-12-11 02:00:00     0  -5.6    94  989.     3   230
 9 MOORE PARK  2017     1     3     9 2017-01-03 09:00:00     0  -5.6    91 1033.     1   330
10 CASEMENT    2017    12    11     3 2017-12-11 03:00:00     0  -5.4    92  988.     4   250
# … with 218,990 more rows
```

Mean Sea Level Pressure

```
> arrange(observations,msl)
# A tibble: 219,000 x 12
   station              year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>               <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 VALENTIA OBSERVATORY 2017    10    16    11 2017-10-16 11:00:00   9.8  14.6    95  962.    24   100
 2 BELMULLET            2017     2     2    20 2017-02-02 20:00:00   2.5   9.4    94  964.    25   140
 3 BELMULLET            2017     2     2    19 2017-02-02 19:00:00   0     9.3    89  964.    15   140
 4 BELMULLET            2017     2     2    18 2017-02-02 18:00:00   0.1   9.4    87  965.    17   140
 5 MACE HEAD            2017     2     2    15 2017-02-02 15:00:00   0.2  10.1    86  965.    23   120
 6 BELMULLET            2017     2     2    17 2017-02-02 17:00:00   0.3   9.6    88  965     18   140
 7 MACE HEAD            2017     2     2    16 2017-02-02 16:00:00   0.4   9.7    90  965.    19   140
 8 MACE HEAD            2017     2     2    17 2017-02-02 17:00:00   0.2   9.5    90  965.    17   140
 9 BELMULLET            2017     2     2    16 2017-02-02 16:00:00   0    10.6    79  965.    18   140
10 MACE HEAD            2017     2     2    14 2017-02-02 14:00:00   0    10.8    82  966.    22   120
# … with 218,990 more rows
```

More than one value

```
> arrange(observations,month,temp)
# A tibble: 219,000 x 12
   station     year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>      <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 MOORE PARK  2017     1     3     9 2017-01-03 09:00:00     0  -5.6    91 1033.     1   330
 2 MOORE PARK  2017     1     3     8 2017-01-03 08:00:00     0  -5.4    91 1033.     1   160
 3 MARKREE     2017     1    23     4 2017-01-23 04:00:00     0  -5.1    96 1024.    NA    NA
 4 MOORE PARK  2017     1     3     7 2017-01-03 07:00:00     0  -5.1    92 1033.     1   250
 5 MARKREE     2017     1    23     5 2017-01-23 05:00:00     0  -5      98 1024.    NA    NA
 6 MARKREE     2017     1    23     2 2017-01-23 02:00:00     0  -4.8    97 1025.    NA    NA
 7 MARKREE     2017     1    23     3 2017-01-23 03:00:00     0  -4.8    98 1025.    NA    NA
 8 MOORE PARK  2017     1     3     6 2017-01-03 06:00:00     0  -4.8    92 1033.     1   270
 9 MT DILLON   2017     1    21     8 2017-01-21 08:00:00     0  -4.6    96 1027.     2   350
10 MARKREE     2017     1    23     1 2017-01-23 01:00:00     0  -4.4    96 1026.    NA    NA
# … with 218,990 more rows
```

In descending order - desc()

```
> arrange(observations,desc(temp))
# A tibble: 219,000 x 12
   station       year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>        <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 PHOENIX PARK  2017     6    21    13 2017-06-21 13:00:00   0.1  28.3    51 1010     NA    NA
 2 PHOENIX PARK  2017     6    21    12 2017-06-21 12:00:00   0    27.5    54 1011.    NA    NA
 3 PHOENIX PARK  2017     6    21    14 2017-06-21 14:00:00   0    27.5    49 1010.    NA    NA
 4 PHOENIX PARK  2017     6    21    16 2017-06-21 16:00:00   0    26.8    61 1009.    NA    NA
 5 CASEMENT      2017     6    21    12 2017-06-21 12:00:00   0    26.6    54 1011.    11   150
 6 MOORE PARK    2017     6    19    16 2017-06-19 16:00:00   0    26.6    50 1018.     3   200
 7 DUNSANY       2017     6    21    12 2017-06-21 12:00:00   0    26.5    55 1010.     8   150
 8 PHOENIX PARK  2017     6    21    11 2017-06-21 11:00:00   0    26.5    56 1011.    NA    NA
 9 PHOENIX PARK  2017     6    17    16 2017-06-17 16:00:00   0    26.4    42 1024.    NA    NA
10 PHOENIX PARK  2017     6    21    15 2017-06-21 15:00:00   0    26.4    61 1009.    NA    NA
# … with 218,990 more rows
```

Challenge 4.2

- Arrange the observations by month and by highest temperature

3. select()

- It is not uncommon to get datasets with hundreds, or even thousands, of variables
- A challenge is to narrow down on the variables of you're interested in
- select() allows you to rapidly zoom in on a useful subset using operations based on the variable names
- Number of rows does not change

```
> new_obs <- select(observations,station,year,month,day,hour,temp)
> new_obs
# A tibble: 219,000 x 6
   station  year month   day  hour  temp
   <chr>   <dbl> <dbl> <int> <int> <dbl>
 1 ATHENRY  2017     1     1     0   5.2
 2 ATHENRY  2017     1     1     1   4.7
 3 ATHENRY  2017     1     1     2   4.2
 4 ATHENRY  2017     1     1     3   3.5
 5 ATHENRY  2017     1     1     4   3.2
 6 ATHENRY  2017     1     1     5   2.1
 7 ATHENRY  2017     1     1     6   2
 8 ATHENRY  2017     1     1     7   1.7
 9 ATHENRY  2017     1     1     8   1
10 ATHENRY  2017     1     1     9   1.1
# … with 218,990 more rows
```

Useful options with select()

```
> select(observations,station:rain)
# A tibble: 219,000 x 7
   station  year month   day  hour date                rain
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl>
 1 ATHENRY  2017     1     1     0 2017-01-01 00:00:00   0
 2 ATHENRY  2017     1     1     1 2017-01-01 01:00:00   0
 3 ATHENRY  2017     1     1     2 2017-01-01 02:00:00   0
 4 ATHENRY  2017     1     1     3 2017-01-01 03:00:00   0.1
 5 ATHENRY  2017     1     1     4 2017-01-01 04:00:00   0.1
 6 ATHENRY  2017     1     1     5 2017-01-01 05:00:00   0
 7 ATHENRY  2017     1     1     6 2017-01-01 06:00:00   0
 8 ATHENRY  2017     1     1     7 2017-01-01 07:00:00   0
 9 ATHENRY  2017     1     1     8 2017-01-01 08:00:00   0
10 ATHENRY  2017     1     1     9 2017-01-01 09:00:00   0
# … with 218,990 more rows
```

```
> select(observations,-(station:rain))
# A tibble: 219,000 x 5
    temp  rhum   msl  wdsp wddir
   <dbl> <dbl> <dbl> <dbl> <dbl>
 1   5.2    89 1022.     8   320
 2   4.7    89 1022      9   320
 3   4.2    90 1022.     8   320
 4   3.5    87 1022.     9   330
 5   3.2    89 1023.     8   330
 6   2.1    91 1023.     8   330
 7   2      89 1024.     7   330
 8   1.7    89 1024.     7   340
 9   1      91 1025      7   330
10   1.1    91 1026.     8   330
# … with 218,990 more rows
```

Special functions with select()

Special functions

As well as using existing functions like `:` and `c`, there are a number of special functions that only work inside `select`

- `starts_with(x, ignore.case = TRUE)`: names starts with `x`
- `ends_with(x, ignore.case = TRUE)`: names ends in `x`
- `contains(x, ignore.case = TRUE)`: selects all variables whose name contains `x`
- `matches(x, ignore.case = TRUE)`: selects all variables whose name matches the regular expression `x`
- `num_range("x", 1:5, width = 2)`: selects all variables (numerically) from x01 to x05.
- `one_of("x", "y", "z")`: selects variables provided in a character vector.
- `everything()`: selects all variables.

Examples

```
> select(observations,starts_with("w"))
# A tibble: 219,000 x 2
    wdsp wddir
   <dbl> <dbl>
 1     8   320
 2     9   320
 3     8   320
 4     9   330
 5     8   330
 6     8   330
 7     7   330
 8     7   340
 9     7   330
10     8   330
# … with 218,990 more rows
```

```
> select(observations,ends_with("p"))
# A tibble: 219,000 x 2
    temp  wdsp
   <dbl> <dbl>
 1   5.2     8
 2   4.7     9
 3   4.2     8
 4   3.5     9
 5   3.2     8
 6   2.1     8
 7   2       7
 8   1.7     7
 9   1       7
10   1.1     8
# … with 218,990 more rows
```

everything()

```
> select(observations,ends_with("p"),everything())
# A tibble: 219,000 x 12
    temp  wdsp station  year month   day  hour date                rain  rhum   msl wddir
   <dbl> <dbl> <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl>
 1   5.2     8 ATHENRY  2017     1     1     0 2017-01-01 00:00:00   0      89 1022.   320
 2   4.7     9 ATHENRY  2017     1     1     1 2017-01-01 01:00:00   0      89 1022    320
 3   4.2     8 ATHENRY  2017     1     1     2 2017-01-01 02:00:00   0      90 1022.   320
 4   3.5     9 ATHENRY  2017     1     1     3 2017-01-01 03:00:00   0.1    87 1022.   330
 5   3.2     8 ATHENRY  2017     1     1     4 2017-01-01 04:00:00   0.1    89 1023.   330
 6   2.1     8 ATHENRY  2017     1     1     5 2017-01-01 05:00:00   0      91 1023.   330
 7   2       7 ATHENRY  2017     1     1     6 2017-01-01 06:00:00   0      89 1024.   330
 8   1.7     7 ATHENRY  2017     1     1     7 2017-01-01 07:00:00   0      89 1024.   340
 9   1       7 ATHENRY  2017     1     1     8 2017-01-01 08:00:00   0      91 1025    330
10   1.1     8 ATHENRY  2017     1     1     9 2017-01-01 09:00:00   0      91 1026.   330
# … with 218,990 more rows
```

4. mutate()

- It is often useful to add new columns that are functions of existing columns
- mutate() always adds new columns at the end of your data set.
- For example, convert the mph wind speed in observations to a new column, kph.
- Use a simplified observations tibble with day, month, station, wdsp as columns, and for “ROCHES POINT” on October 16th
- Assume 1 mi = 1.609344 km

Example of mutate

```
# dplyr::mutate()
CK2Kmh <- 1.852

obs1 <- observations %>%
          select(day, month, station, wdsp) %>%
          filter(station=="ROCHES POINT",
                 day==16,
                 month==10)

obs1 <- mutate(obs1,wdsp_kph=wdsp*CK2Kmh)
```

```
> obs1
# A tibble: 24 × 5
     day month station       wdsp wdsp_kph
   <int> <dbl> <chr>        <dbl>    <dbl>
 1    16    10 ROCHES POINT    11     20.4
 2    16    10 ROCHES POINT    11     20.4
 3    16    10 ROCHES POINT    14     25.9
 4    16    10 ROCHES POINT    15     27.8
 5    16    10 ROCHES POINT    22     40.7
```

Useful Creation Functions

- There are many functions for creating new variables that can be used with mutate()
- The key property is that the function **must be vectorised:**
  - It must take a vector of values as input, and,
  - Return a vector with the same number of values as output

Grouping	Examples
Arithmetic Operators	+, -, *, /, ^
Modular Arithmetic	%/% - Integer division && - Remainder
Logical comparisons	<, <=, >, >=, !=
If-else Functions	ifelse(), case_when()

Challenge 4.3

- Add a new column rain_in, which measures the hourly rainfall in inches
- Assume a conversion constant of 1 mm = 0.0393701 inches

5. summarise()

- The last key verb is summarise()
- It collapses a data frame into a single row
- Not very useful unless paired with group_by() (it will just perform a summary of the whole data set)
- Very useful to combine with the pipe operator

group_by()

- Most data operations are useful done on groups defined by variables in the the dataset.
- The group_by function takes an existing tbl and converts it into a grouped tbl where operations are performed "by group".

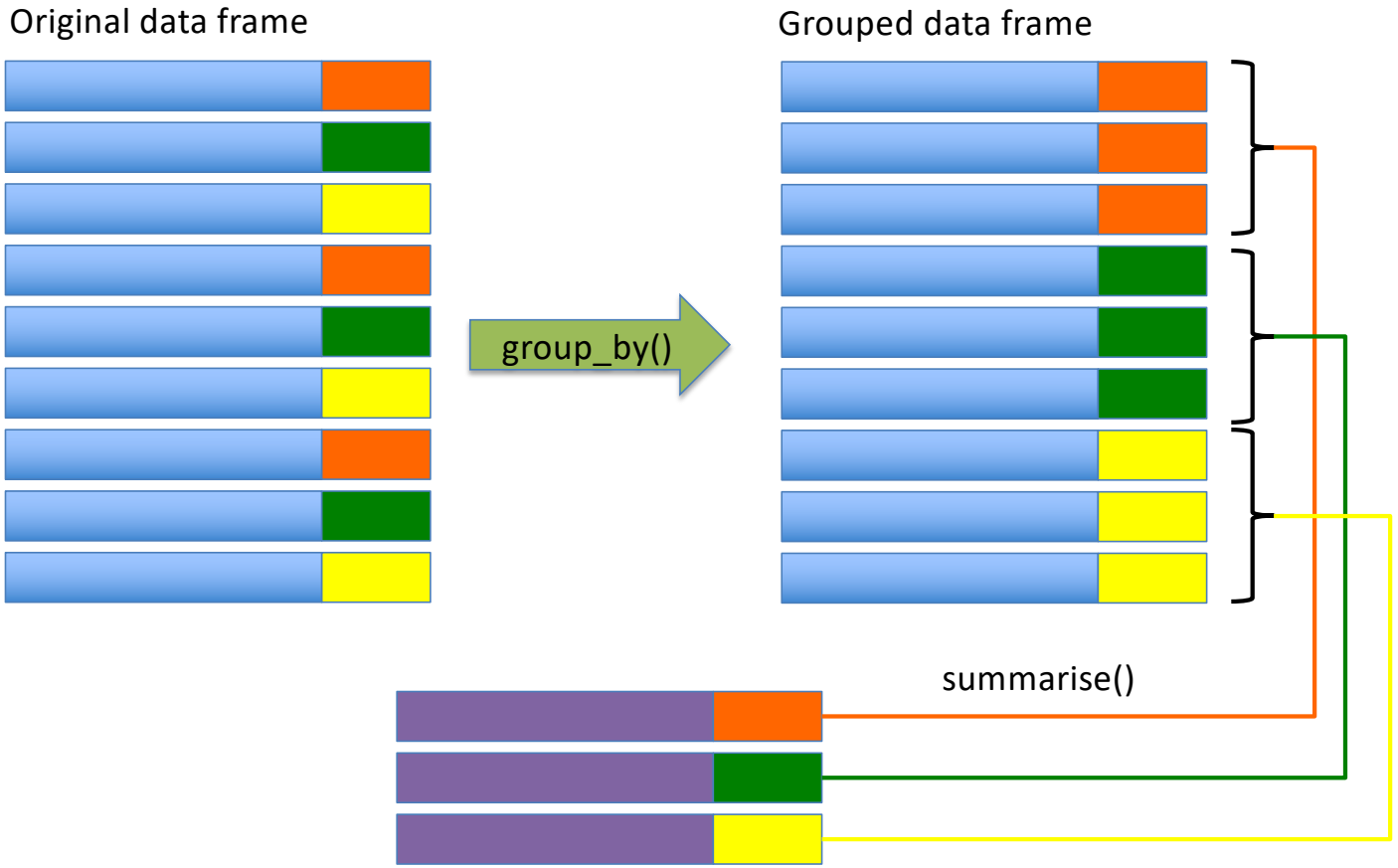
```
> test_g <- group_by(test,day)
> test_g
# A tibble: 744 x 12
# Groups:   day [31]
   station  year month   day  hour date                rain  temp
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl>
 1 MACE H… 2017    10     1     0 2017-10-01 00:00:00   1.2  11.7
 2 MACE H… 2017    10     1     1 2017-10-01 01:00:00   1.9  11.3
```

Key idea

- This grouping can then be exploited by summarise

```
> summarise(test_g,TotalRainfall=sum(rain,na.rm=T))
# A tibble: 31 x 2
     day TotalRainfall
   <int>         <dbl>
 1     1           4.7
 2     2           1.3
 3     3           0
 4     4           4.2
 5     5           0.1
 6     6           3.1
 7     7           0.4
 8     8           0.3
 9     9           1.7
10    10           1.6
# … with 21 more rows
```

Overall idea…



Useful Summary Functions

Grouping	Examples
Measures of location	mean(), median()
Measures of spread	sd(), IQR(),mad()
Measures of rank	min((), quantile(), max()
Measures of position	first(), nth(), last()
Counts	n(), n_distinct()
Counts and proportions of logical values	sum(x>0) when used with numeric functions, (T,F) converted to (1,0)

Challenge 4.4

- For “BELMULLET” calculate the total daily rainfall for October, and show using ggplot.

dplyr Summary: 5 key functions

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- ”A grammar of data manipulation”
- All verbs (functions) work similarly
  - The first argument is a data frame/tibble
  - The subsequent arguments decide what to do with the data frame/tibble
  - The result (data frame/tibble) supports chaining of steps – NOTE the “pipe operator” which we will cover later.

https://dplyr.tidyverse.org

Practically, `ggplot2` provides beautiful, hassle-free plots that take care of fiddly details like drawing legends. The plots can be built up iteratively and edited later.

— Hadley Wickham (Wickham, 2016)

Exploring Operations Research with R: *A Workshop*

05 – Operations Research Examples

https://github.com/JimDuggan/Exploring-OR-with-R-Workshop

Topics Overview

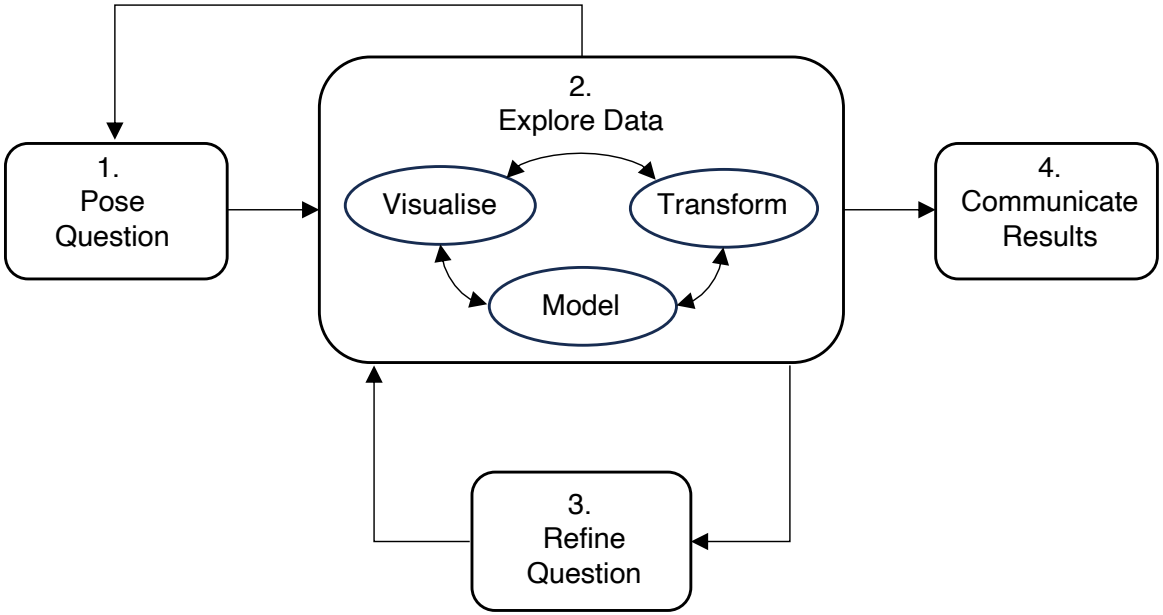
Topic	Description	~Timing
1	Introduction to R and Posit Cloud	10 mins
2	Storing data using data frames / tibbles	10 mins
3	Visualisation with ggplot2	20 mins
4	Data transformation with dplyr	20 mins
5	Using R with OR – Four Examples	30 mins

Example	Description
1	Exploratory Data Analysis
2	Linear Programming
3	Agent Based Simulation
4	Data transformation with dplyr

(1) Exploratory Data Analysis

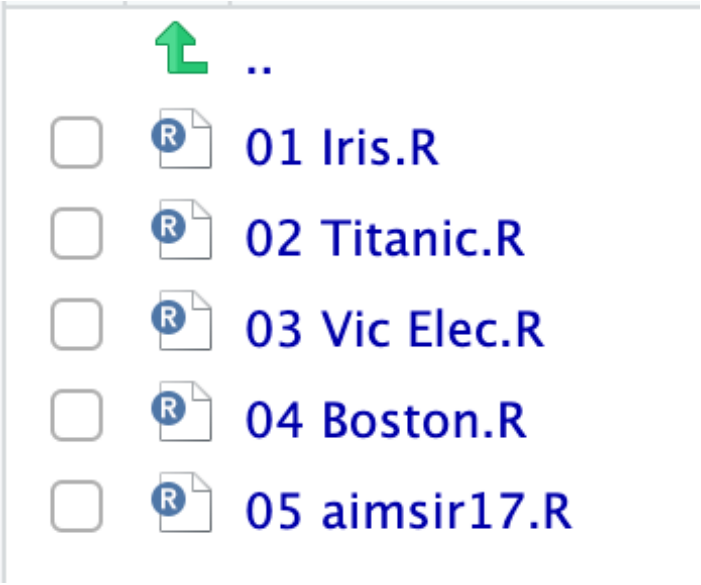
"Exploratory analysis is what you do to understand the data and figure out what might be noteworthy or interesting to highlight to others. When we do exploratory analysis, it's like hunting for pearls in oysters."

Cole Nussbaumer Knaflic (Knaflic, 2015)

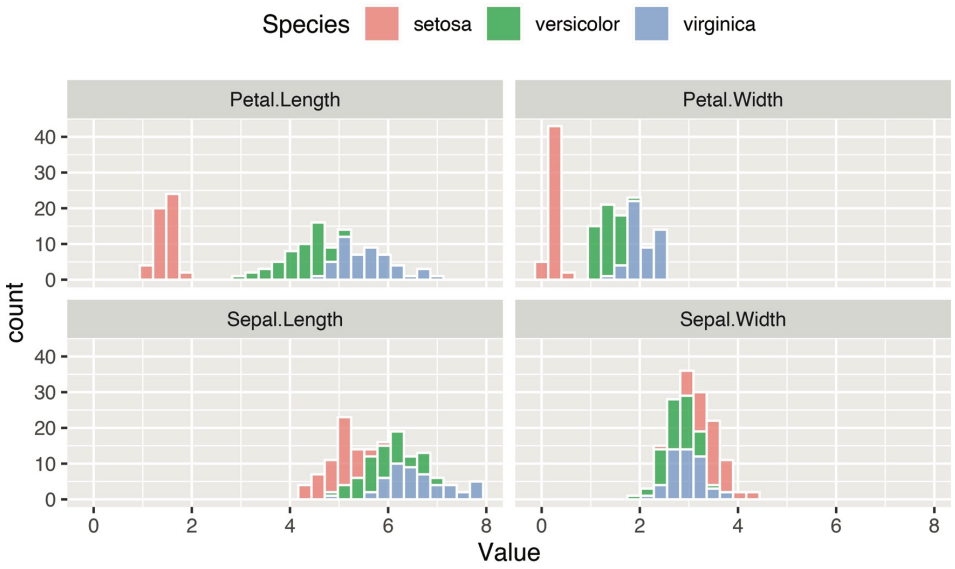


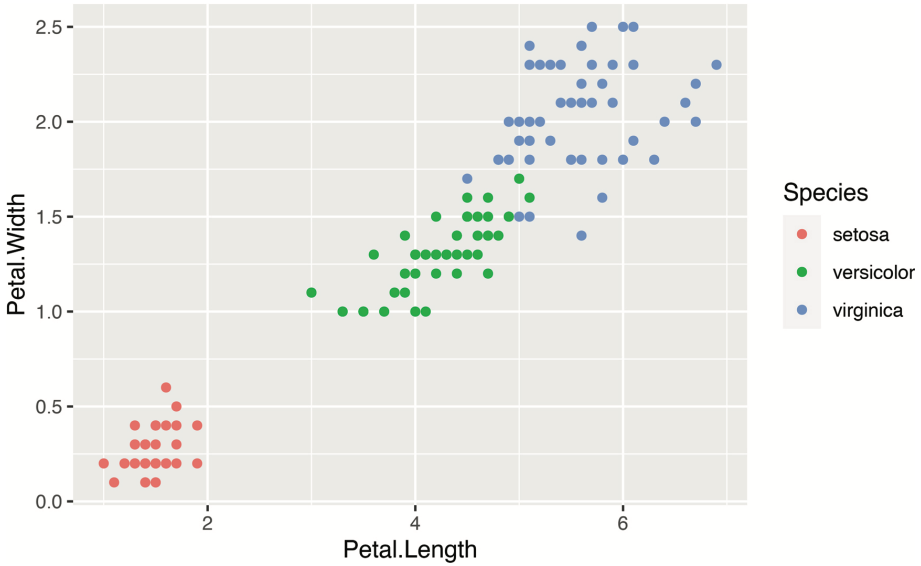
EDA Code Examples

- Iris Flowers
- Titanic
- Victoria Electricity
- Boston Housing
- Irish Weather

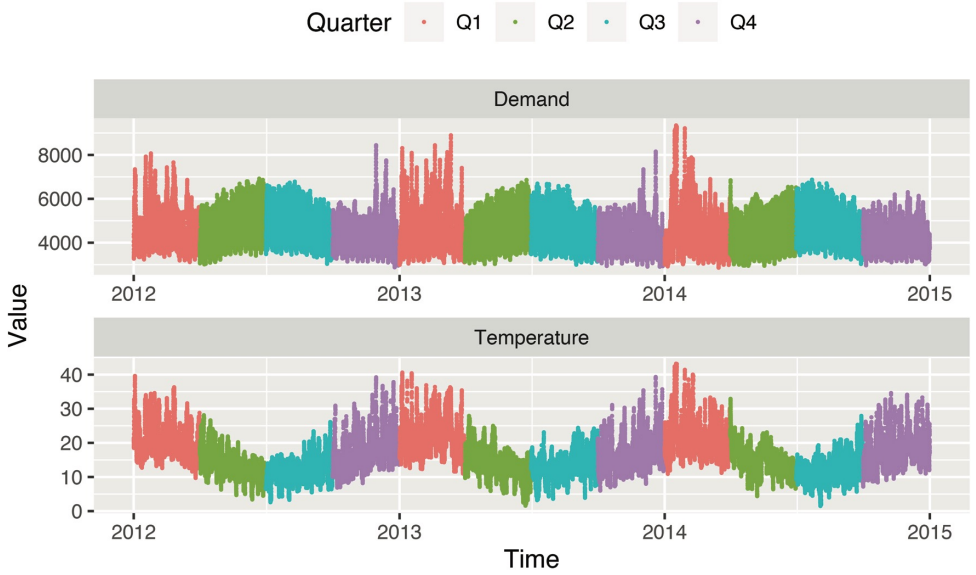


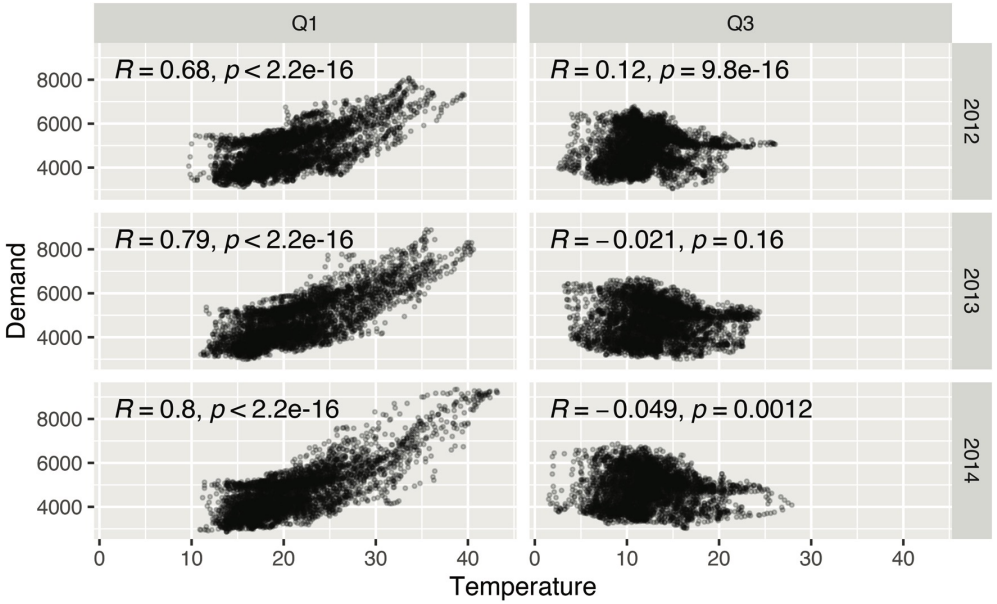
(1) Iris Data Set



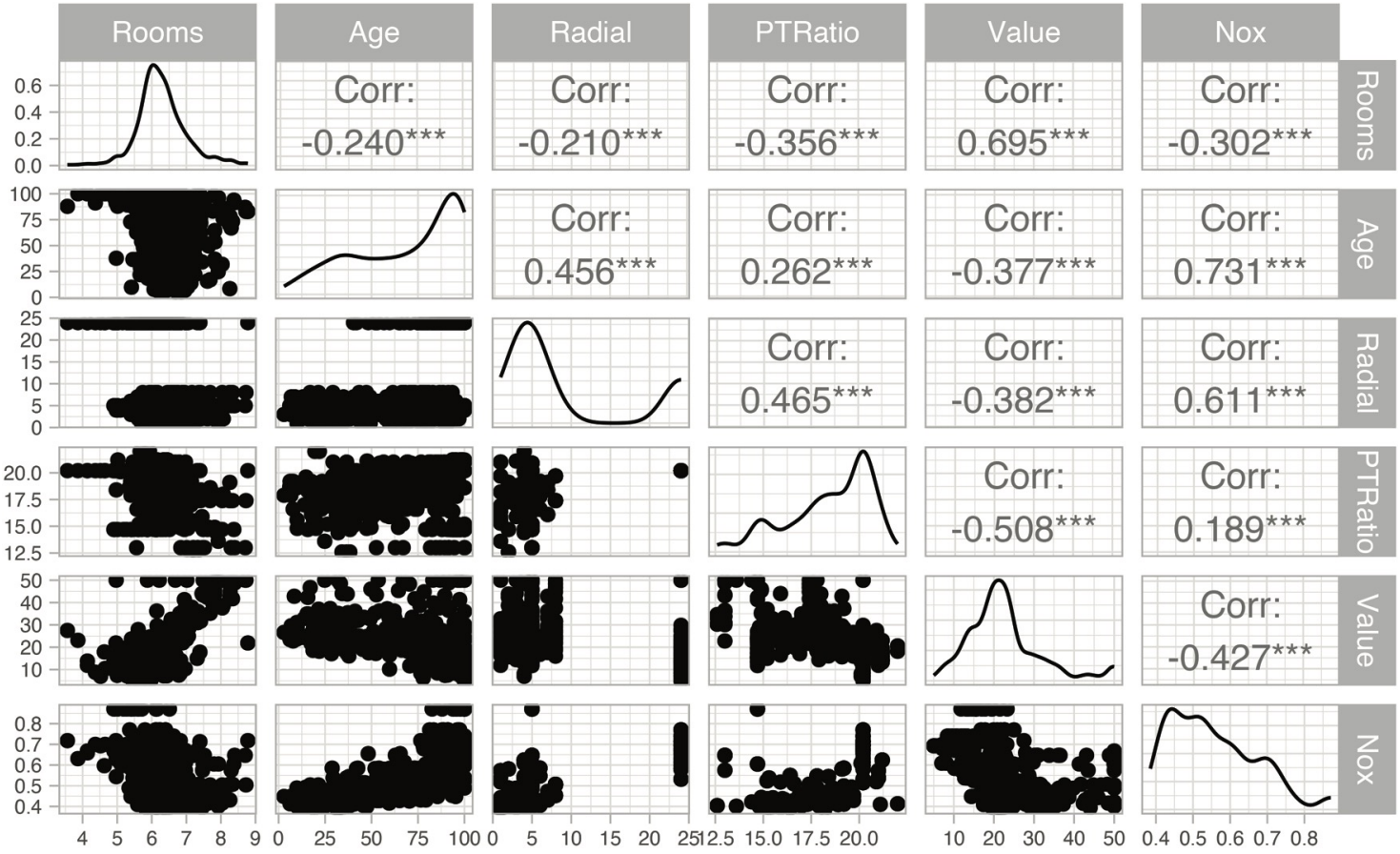


(2) Electricity Demand (Victoria)

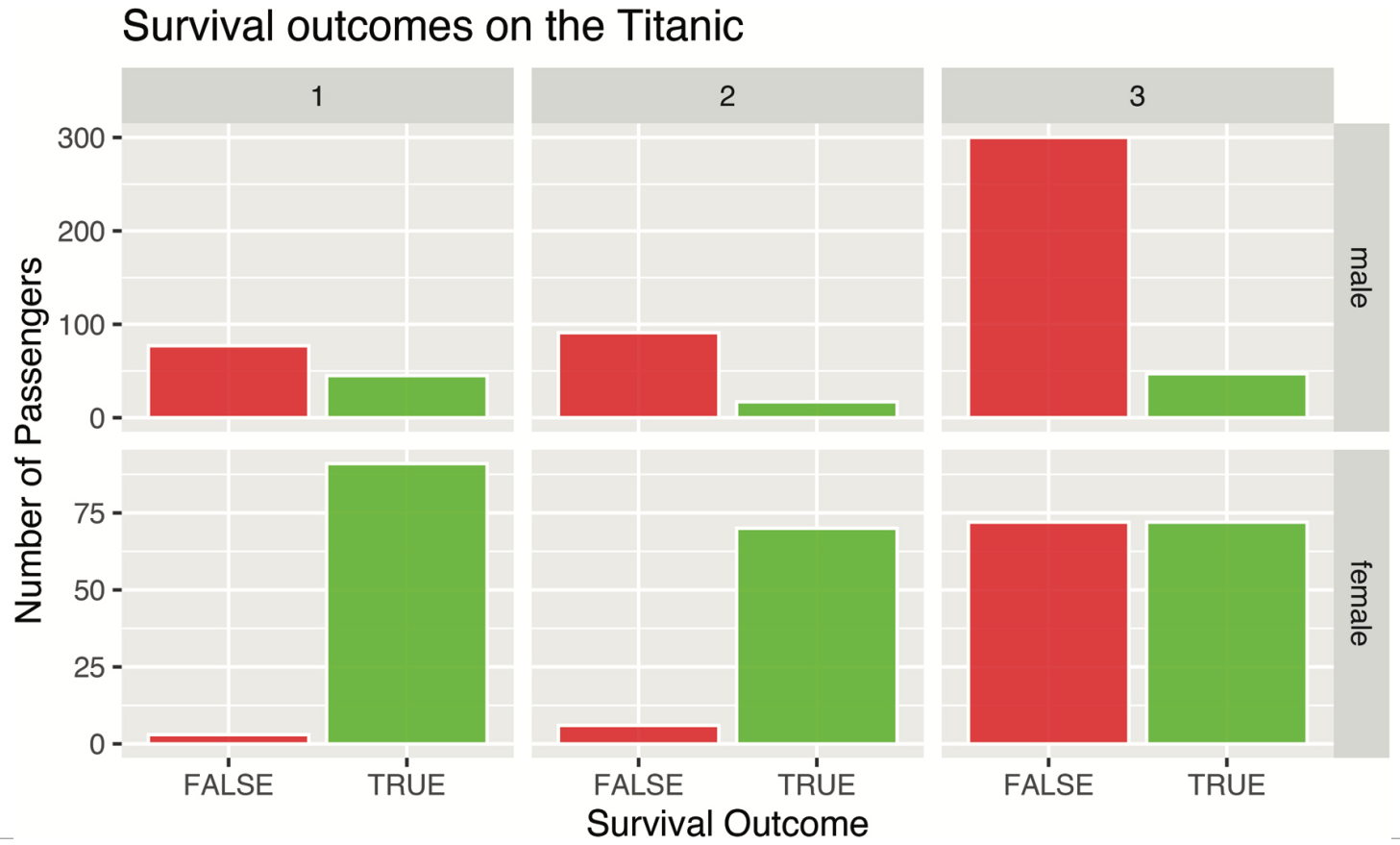




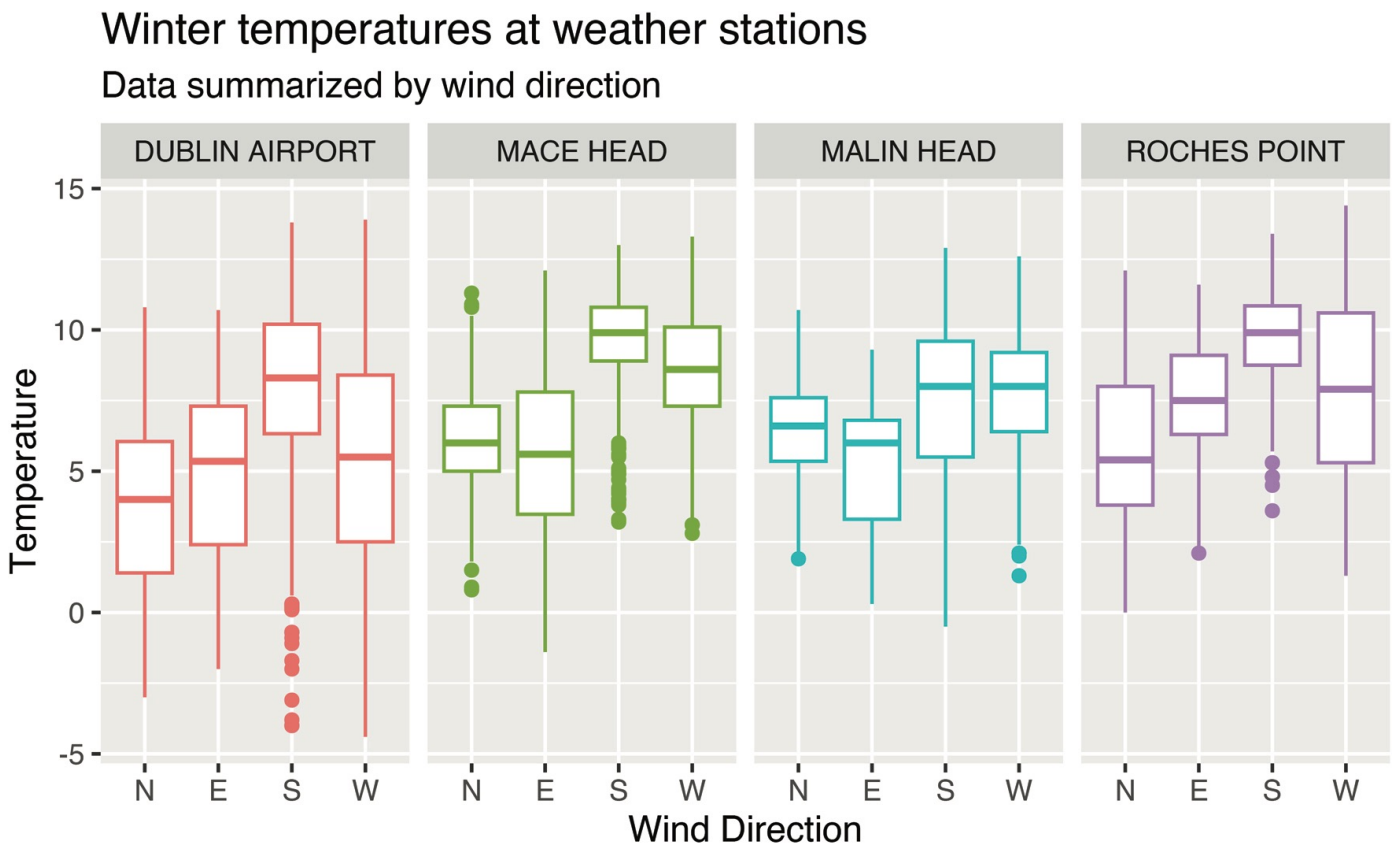
(3) Boston Data



(4) Titanic Survival Data



(5) Irish Weather Analysis (Winter)



(2) Linear Programming

"The success of an OR technique is ultimately measured by the spread of its use as a decision making tool. Ever since in introduction in the late 1940s, linear programming (LP) has proven to be one of the most effective operations research tools."

Hamdy A. Taha (Taha, 1992)

$$Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

subject to the constraints

$$a_{11}x_1 + a_{11}x_2 + \cdots + a_{1n}x_n \leq b_1$$
$$a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$$
$$\vdots$$
$$a_{m1}x_2 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$$

and where

$$x_1 \geq 0, x_2 \geq 0, \cdots + x_n \geq 0$$

The Reddy Mikks example (Taha 1995)

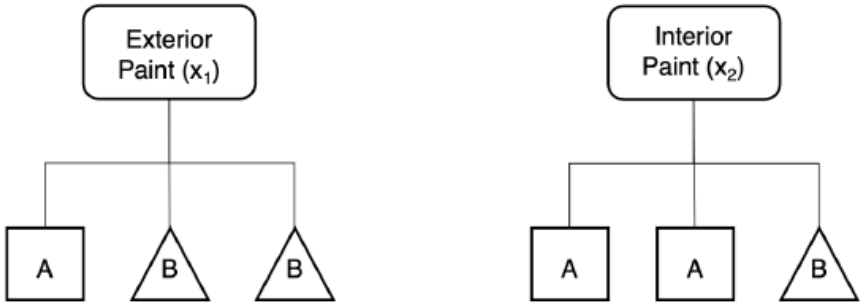


FIGURE 13.1 Raw materials needed (ton) for one ton of interior and exterior paints

Further to these manufacturing details, additional information is available through a market survey:

- The daily demand for interior paint cannot be greater than demand for exterior paint by more than on ton
- The maximum demand for interior paint is limited to two tons per day
- The wholesale price for exterior paint is \$3000 per ton, while the price for interior paint is \$2000 per ton.

In summary, the full mathematical model for this optimisation problem can be defined as follows:
To determine the tons of exterior and interior paint to be produced, expressed as the objective function

$$z = 3x_1 + 2x_2$$

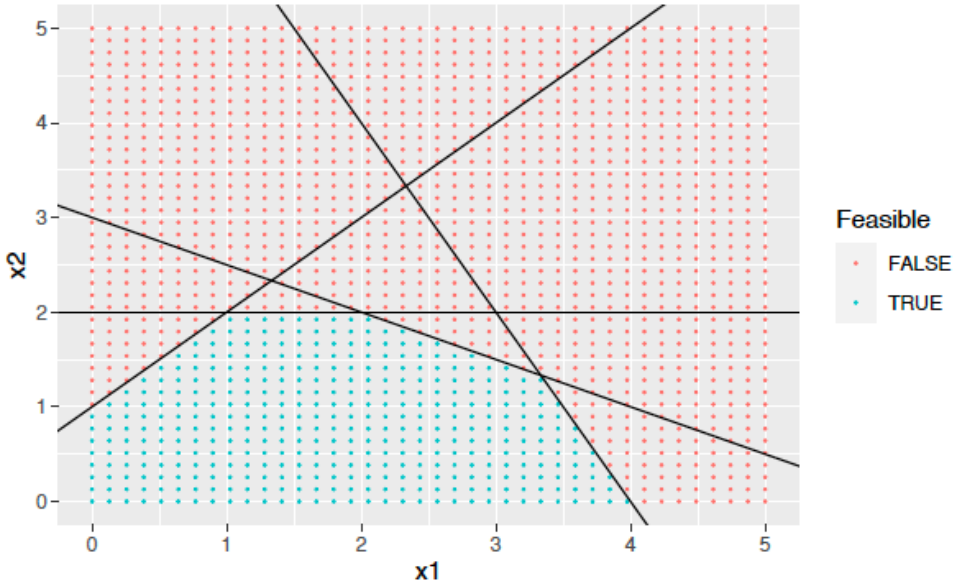
Subject to the following constraints:

$$x_1 + 2x_2 \leq 6$$
$$2x_1 + x_2 \leq 8$$
$$x_2 + x_1 \leq 1$$
$$x_2 \leq 2$$

$$x_1 \geq 0$$
$$x_2 \geq 0$$

We will now focus on solving this problem, and the first step is to explore a range offeasible solutions using R.

13.4 Exploring a two-variable decision space using R



```
library(lpSolve)
```

In order to optimise a linear programming problem, the function `lp()` is called, and the required arguments are:

- `direction`, which is either "min" for minimisation, or "max" for maximisation , where the default value is "min".
- `objective.in`, which contains the numeric vector of coefficients for the objective functions (essentially the c_i values).
- `const.mat`, which is a matrix of constraints, with one row per constraint, and one column for each variable.
- `const.dir`, a vector of character strings that provides the constraint directions, for example "<=", "<", ">=" or ">".
- `const.rhs`, which contains the number values for the right hand side of the constraints.

Every decision variable is assumed to be ≥ 0.

The `lp()` function returns an object of type "lp", which contains the following elements.

lp Element	Explanation
`direction`	Optimisation direction, as specified in the call
`x.count`	The number of decision variables in the objective function
`objective`	The vector of objective function coefficients
`const.count`	The total number of constraints provided
`constraints`	The constraint matrix, as provided
`int.count`	The number of integer variables (not used here)
`int.vec`	The vector of integer variable's indices (not used here)
`objval`	The objective value function at the optimium
`solution`	The vector of optimal coefficents for the decision variables
`num.bin.solns`	The number of solutions returned
`status`	Return status, 0 = success, 2= no feasible solution

$$z = 3x_1 + 2x_2$$

Subject to the following constraints:

$$x_1 + 2x_2 \leq 6$$
$$2x_1 + x_2 \leq 8$$
$$x_2 + x_1 \leq 1$$
$$x_2 \leq 2$$

We now construct the linear programming formulation for the Reddy Mikks problem. First, we define the objective function in a vector. These are coefficient values for c_1 and c_2.

```
z <- c(3,2)
z
#> [1] 3 2
```

Next, we define the left hand side of the four constraints. This is a $4X2$ matrix, where each row represents a the multipliers for each constraint, and each column represents a decision variable.

```
cons <- matrix(c(1,2,
                 2,1,
                 -1,1,
                 0,1),byrow = T,nrow=4)
cons
#>      [,1] [,2]
#> [1,]    1    2
#> [2,]    2    1
#> [3,]   -1    1
#> [4,]    0    1
```

$$z = 3x_1 + 2x_2$$

Subject to the following constraints:

$$x_1 + 2x_2 \leq 6$$
$$2x_1 + x_2 \leq 8$$
$$x_2 + x_1 \leq 1$$
$$x_2 \leq 2$$

```
eql <- c("<=", "<=", "<=","<=")
eql
#> [1] "<=" "<=" "<=" "<="
```

Finally, the right hand side values for the constaints are specifed, with four values provided, one for each constraint.

```
rhs <- c(6,8,1,2)
rhs
#> [1] 6 8 1 2
```

With these values specified, we can now call the function `lp()` to generate an optimal solution.

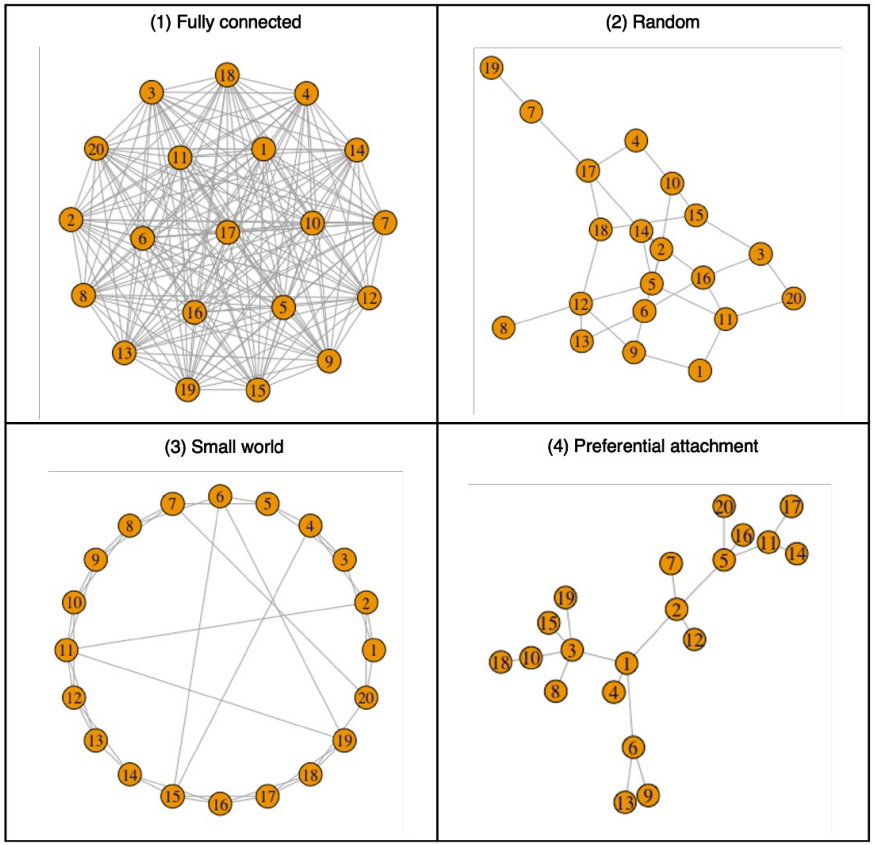
```
opt <- lp("max",z,cons,eql,rhs)
# Show the status
opt$status
#> [1] 0
# Show the objective function value
opt$objval
#> [1] 12.67
# Show the solution points
```

```
opt$solution
#> [1] 3.333 1.333
```

(3) Agent Based Simulation

Situate an initial population of autonomous heterogeneous agents in a relevant spatial environment; allow them to interact according to simple local rules, and thereby generate – or "grow" - the macroscopic regularity from the bottom up.

- Joshua M. Epstein (Epstein, 2012)



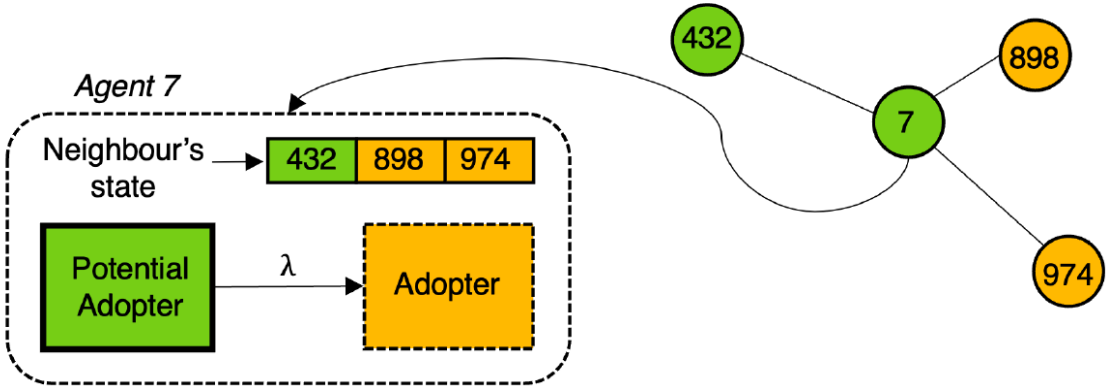
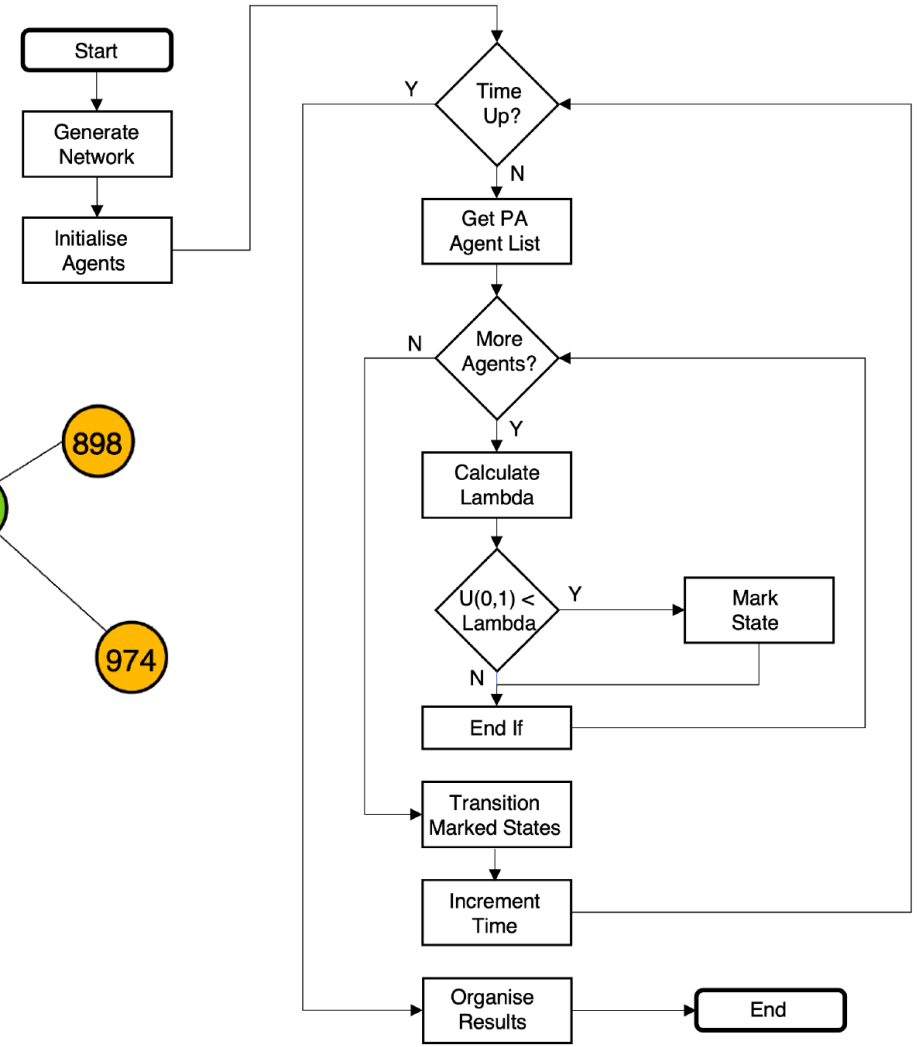
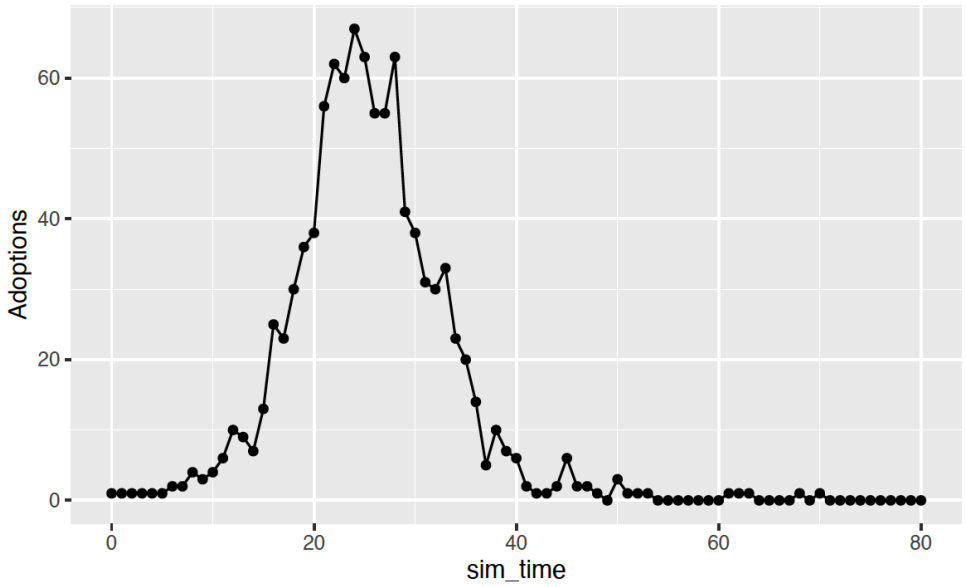
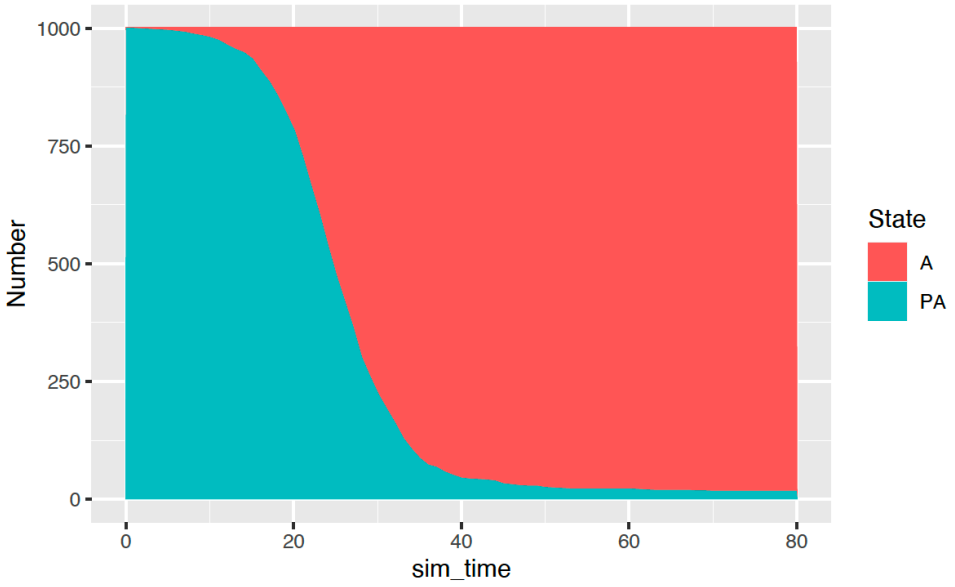


FIGURE 14.2 The state transition mechanism for Agent 7

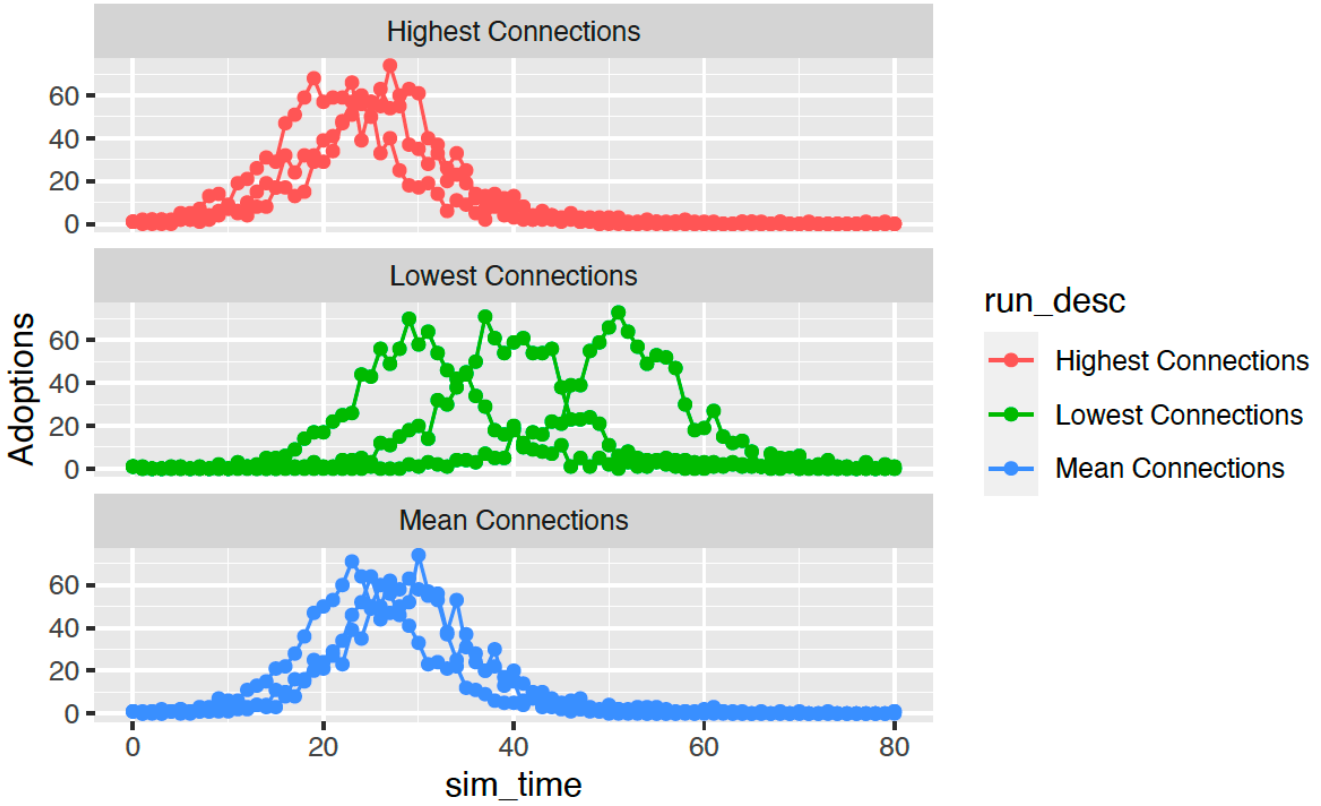


A single run…

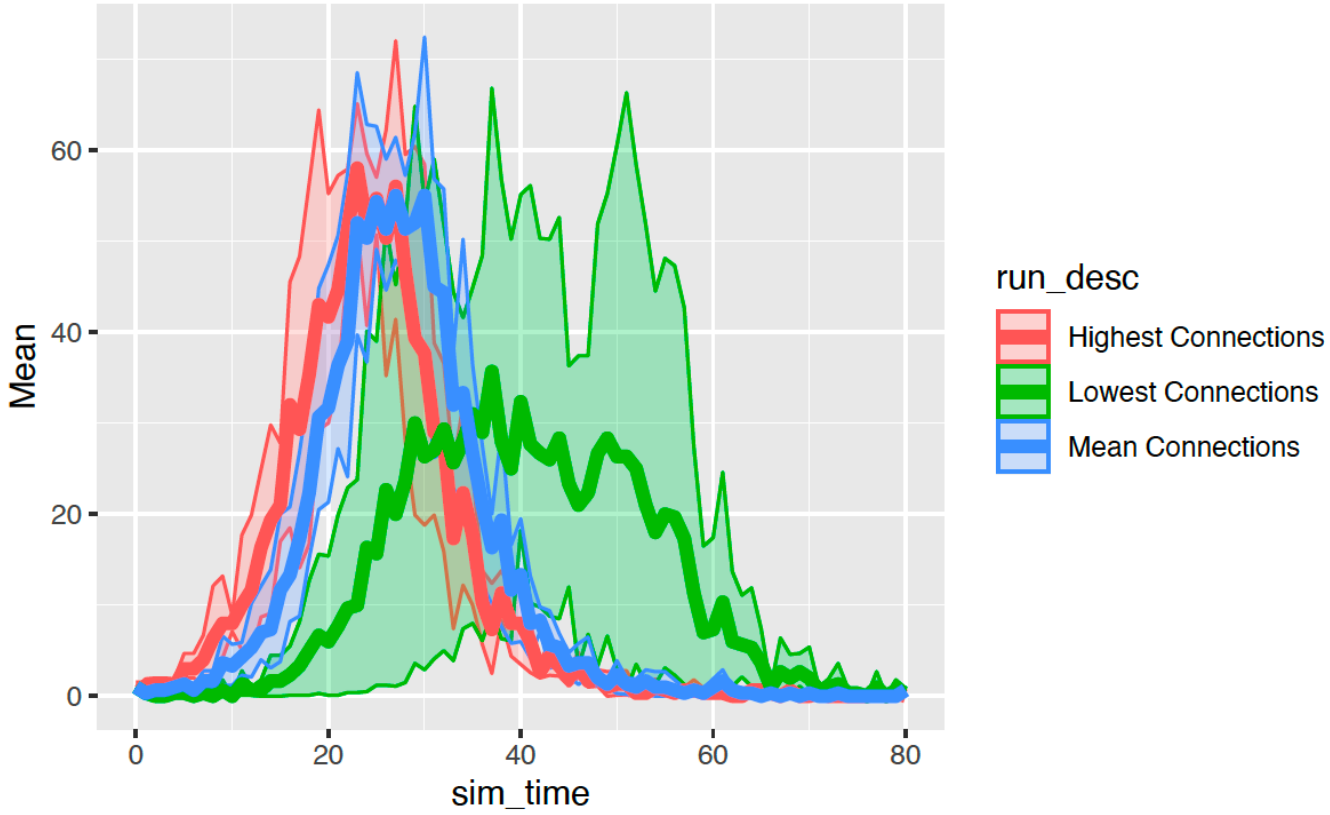

Adoptions
60
40
20
0
0
20
40
60
80
sim_time


Number
1000
750
500
250
0
0
20
40
60
80
sim_time
State
A
PA

Three Scenarios (Different seeded individual)



Quantile analysis



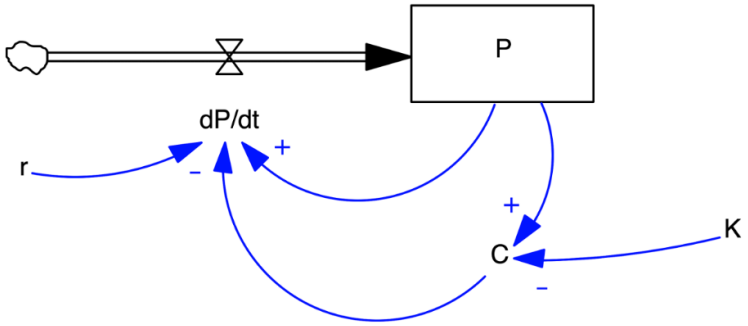
(4) System Dynamics

"Everything we do as individuals, as an industry, or as a society is done in the context of an information-feedback system."

- Jay W. Forrester (Forrester, 1961)

Limits to Growth Model

(a) Limits to growth stock and flow model



(b) Limits to growth differential equation model

$$\frac{dP}{dt} = r\,P\,(1 - C) \quad (1) \qquad C = {}^{P}/_{K} \quad (2) \qquad r = 0.15 \quad (3)$$

$$K = 100{,}000 \quad (4) \qquad P_{INIT} = 100 \quad (5)$$

deSolve

R's `deSolve` package solves initial value problems written as ordinary differential equations (ODE), differential algebraic equations (DAE) and partial differential equations (PDE) (Soetaert et al., 2010). This package provides wrapper functions around various ODE solvers. Within the package `deSolve`, we will make use of the function `ode()`, which solves a system of ordinary differential equations (i.e. a system of stock and flow equations). It takes the following arguments.

```
ode(y, times, func, parms,
method = c("lsoda", "lsode", "lsodes", "lsodar", "vode", "daspk",
           "euler", "rk4", "ode23", "ode45", "radau",
           "bdf", "bdf_d", "adams", "impAdams", "impAdams_d", "iteration"),...)
```

Arguments	Description
`y`	The initial (state) values for the ODE system, a vector. If `y` has a name attribute, the names will be used to label the output matrix.
`times`	The time sequence for which output is wanted; the first value of times must be the initial time.
`func`	The R-function that computes the values of the derivatives in the ODE system (the model definition) at time `t`. It must be defined as: `func <- function(t, y, parms,...)`, where `t` is the current time point of the integration, and `y` is the current estimate of the ODE system variables. The return value of `func` should be a list whose first element is a vector containing the derivatives of `y` with respect to time, and whose next elements are (optional) global values to be recorded. The derivatives must be specified in the same order as the state variables `y`
`times`	The time sequence for which output is wanted; the first value of times must be the initial time.
`params`	Parameters passed to `func`
`method`	Normally a string to indicate the numerical integration method, for example, "euler", "rk4", "ode23", "ode45".
`returns`	A matrix of S3 class `deSolve` with up to as many rows as elements in times and as many columns as elements in y plus the number of "global" values returned in the second element of the return from `func`, plus an additional column (the first) for the time value. This can be easily converted to a data frame object using the function `data.frame()`

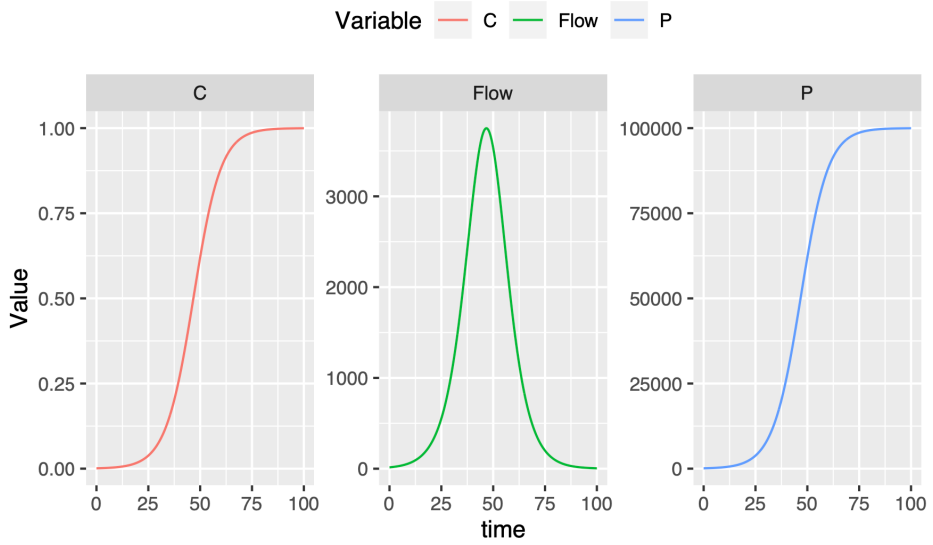
```r
ltg <- function(time, stocks, auxs){
  with(as.list(c(stocks, auxs)),{
    C <- P/K              # Eq (2)
    dP_dt <- r*P*(1-C)    # Eq (1)
    return (list(c(dP_dt), r=r, K=K,C=C,Flow=dP_dt))
  })
}
```

```r
simtime <- seq(0,100,by=0.25)
stocks  <- c(P=100)           # Eq (5)
auxs    <- c(r=0.15,K=100000)  # Eq (3) and Eq (4)
```

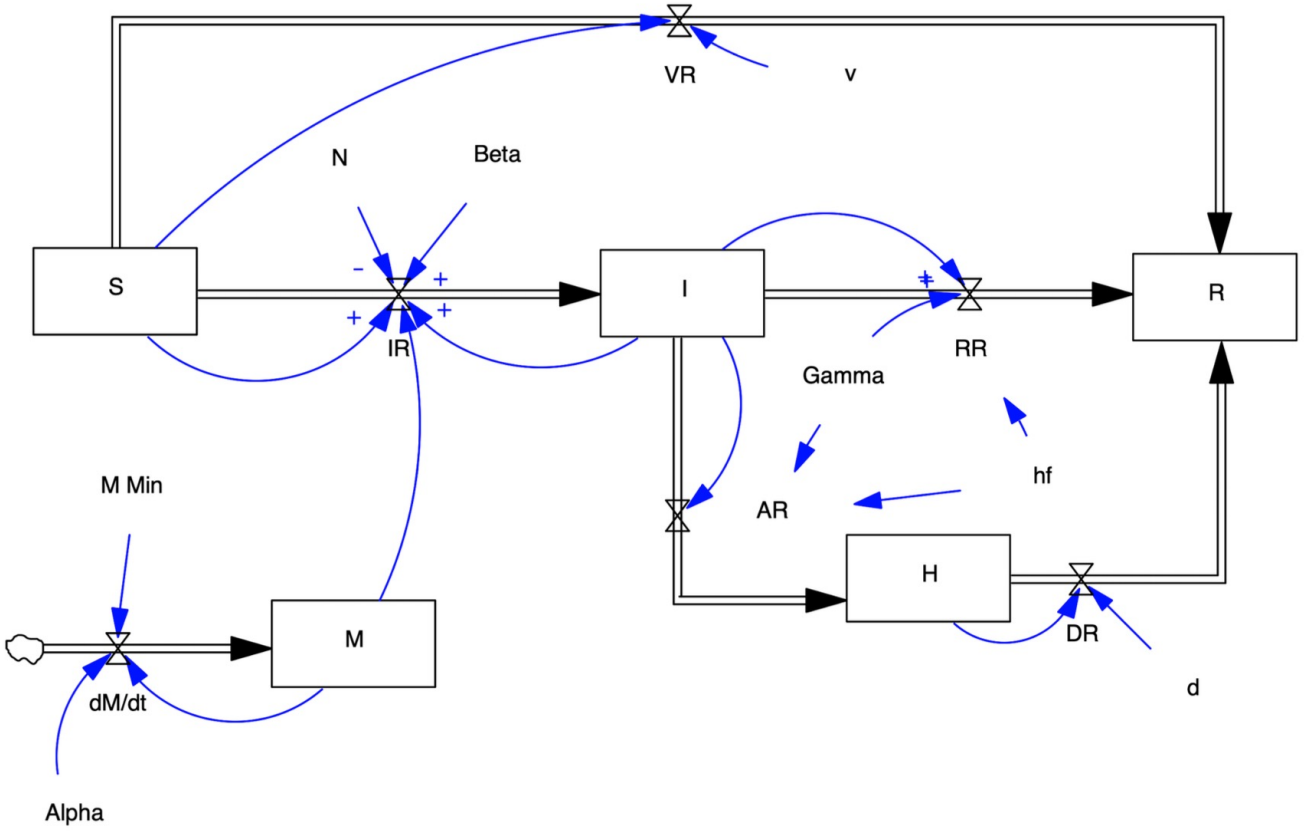
```r
res <- ode(y=stocks,
           times=simtime,
           func = ltg,
           parms=auxs,
           method="euler") %>%
        data.frame() %>%
        as_tibble()
```

```
res
#> # A tibble: 401 x 6
#>     time     P     r      K        C  Flow
#>    <dbl> <dbl> <dbl>  <dbl>    <dbl> <dbl>
#>  1  0     100   0.15 100000 0.001     15.0
#>  2  0.25  104.  0.15 100000 0.00104   15.5
#>  3  0.5   108.  0.15 100000 0.00108   16.1
#>  4  0.75  112.  0.15 100000 0.00112   16.7
#>  5  1     116.  0.15 100000 0.00116   17.4
#>  6  1.25  120.  0.15 100000 0.00120   18.0
#>  7  1.5   125.  0.15 100000 0.00125   18.7
#>  8  1.75  129.  0.15 100000 0.00129   19.4
#>  9  2     134.  0.15 100000 0.00134   20.1
#> 10  2.25  139.  0.15 100000 0.00139   20.9
#> # i 391 more rows
```

```
res_long <- res %>%
              dplyr::select(time,C,P,Flow) %>%
              pivot_longer(names_to = "Variable", values_to = "Value",-time)

ggplot(res_long,aes(x=time,y=Value,colour=Variable)) +
  geom_line() + facet_wrap(~Variable,scales = "free")+
  theme(legend.position = "top")
```



SIRH Model



$$\frac{dS}{dt} = -IR - VR \quad (17) \qquad \frac{dI}{dt} = IR - RR - AR \quad (18) \qquad \frac{dH}{dt} = AR - DR \quad (19)$$

$$\frac{dR}{dt} = RR + DR + VR \quad (20) \qquad \frac{dM}{dt} = \alpha\,(M_{min} - M) \quad (21) \qquad IR = \beta\, M\, I\, \frac{S}{N} \quad (22)$$

$$VR = v\, S \quad (23) \qquad AR = h_f I\, \gamma \quad (24) \qquad RR = (1 - h_f) I\, \gamma \quad (25)$$

$$DR = H\, d \quad (26) \qquad \beta = 1.0 \quad (27) \qquad \gamma = 0.25 \quad (28)$$

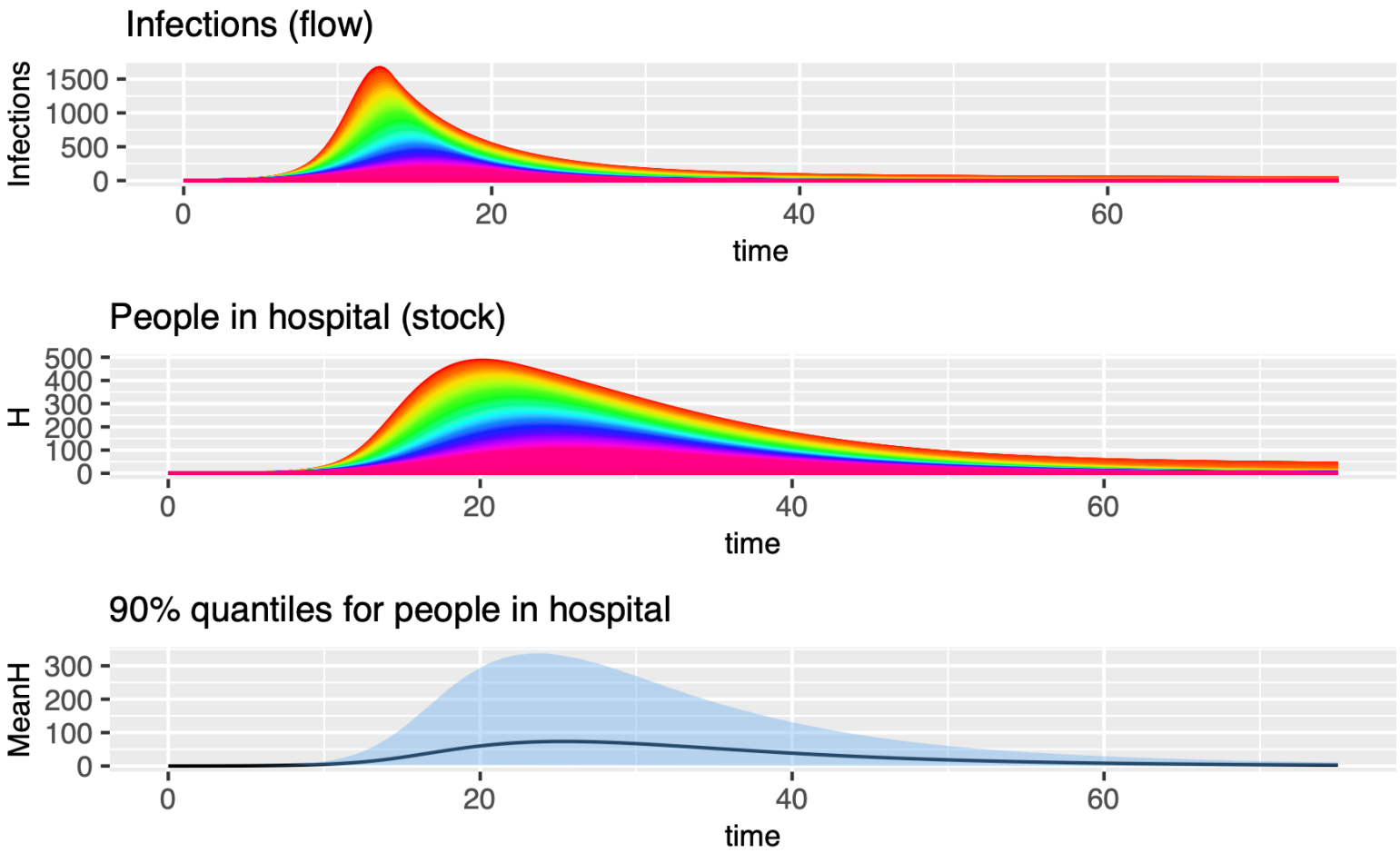
$$v = 0.1 \quad (29) \qquad h_f = 0.1 \quad (30) \qquad N = 10{,}000 \quad (31)$$

$$d = 0.10 \quad (32) \qquad \alpha = 0.5 \quad (33) \qquad S_{INIT} = 9999 \quad (34)$$

$$I_{INIT} = 1 \quad (35) \qquad H_{INIT} = 0 \quad (36) \qquad R_{INIT} = 0 \quad (37)$$

$$M_{INIT} = 1 \quad (38) \qquad M_{MIN} = 0.3 \quad (39)$$

```r
sirh <- function(time, stocks, auxs){
  with(as.list(c(stocks, auxs)),{
    N <- S + I + R + H          # Eq (31)
    IR <- beta*I*S/N*M          # Eq (22)
    VR <- v*S                   # Eq (23)
    AR <- hf*gamma*I            # Eq (24)
    RR <- (1-hf)*gamma*I        # Eq (25)
    DR <- d*H                   # Eq (26)

    dS_dt  <- -IR -VR           # Eq (17)
    dI_dt  <-  IR - AR - RR     # Eq (18)
    dH_dt  <-  AR - DR          # Eq (19)
    dR_dt  <-  RR + DR + VR     # Eq (20)
    dM_dt  <- (M_min - M) *
                alpha           # Eq (21)
    return (list(c(dS_dt,dI_dt,dH_dt,dR_dt,dM_dt),
                Beta=beta,
                Gamma=gamma,
                HF=hf,
                V=v,
                Alpha=alpha,
                M_Min=M_min,
                Infections=IR,
                Recovering=RR,
                Vaccinated=VR,
                Hospitalised=AR,
                Discharged=DR,
                CheckSum=S + I + R + H))
  })
}
```

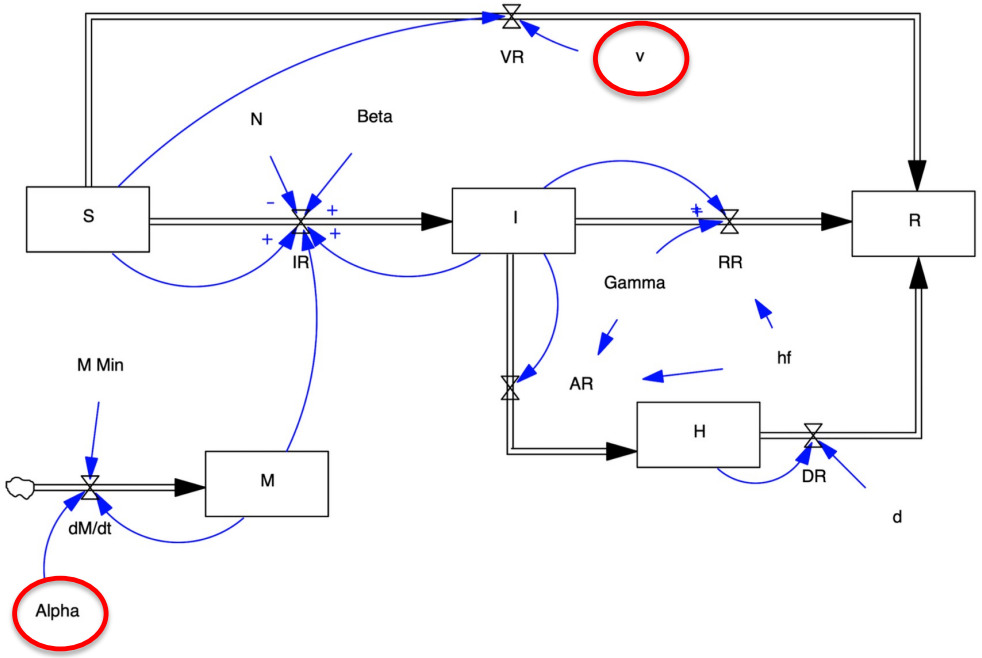

Infections (flow)
Infections
1500
1000
500
0
0
20
40
60
time
People in hospital (stock)
H
500
400
300
200
100
0
0
20
40
60
time
90% quantiles for people in hospital
MeanH
300
200
100
0
0
20
40
60
time

Exploring scenarios: *2 policy levers*

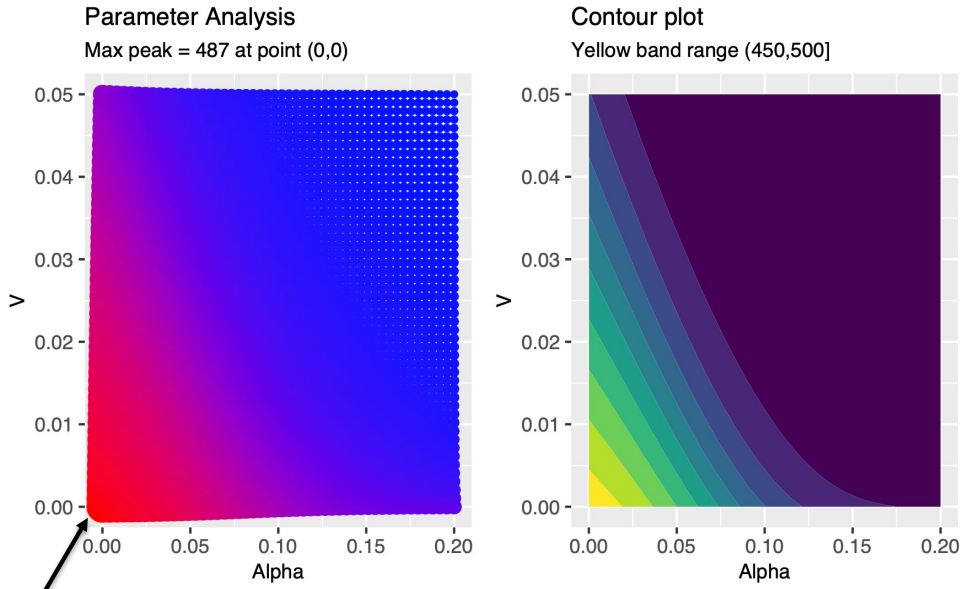
The advantage of creating the function `run_scenario()` is that we can now call this for a range of parameter values. Here, we are going to sample two policy variables:

- α, which models the speed of mobility restriction implementations. For example, a higher value of α would mean that the population responds quickly to the request for social mobility reductions. In our simulations $0 \leq \alpha \leq 0.20$.
- v, which models the speed of vaccination. A higher value of v means that people transfer more quickly from S to R, and therefore the burden on the hospital sector should be reduced. In our simulations $0 \leq v \leq 0.05$, which indicates that the minimum vaccination duration is 20 days $(1/0.05)$.

Speed of vaccination



Speed of mobility reduction



No vaccination,
no change in mobility = Max point

Recap.

- R programming language can be a valuable tool – and way of thinking – which can be successfully applied to the field of operations research (OR).
- Core R tools (tidyverse)
- Application areas to OR:
  - Exploratory Data Analysis
  - Linear Programming
  - Agent-Based Simulation
  - System Dynamics