

# Data Wrangling with R: Day 1

## Base R and tidyverse

Presented by Emi Tanaka

Department of Econometrics and Business Statistics



MONASH University

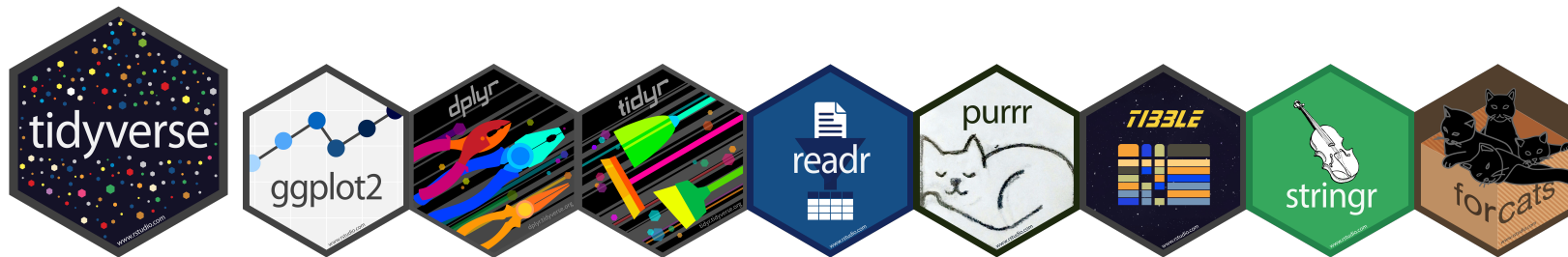
1st December 2020 @ Statistical Society of Australia | Zoom

# Base R

- **Base R** refers to the state of R when just launched in a clean interactive development environment (IDE, e.g. RStudio).
- In this state, R attached the following packages:
  - `base`,
  - `methods`, for complex reasons, you may need to explicitly call this when using Rscript and package development!
  - `stats`,
  - `graphics`,
  - `grDevices`,
  - `utils`, and
  - `datasets`.





# Tidyverse

- What packages does `library(tidyverse)` load?
- **Tidyverse** refers to a collection of R-packages: `ggplot2`, `dplyr`, `tidyr`, `readr`, `purrr`, `tibble`, `stringr`, `forcats`, DBI, haven, httr, readxl, rvest, jsonlite, xml2, lubridate, hms, blob, magrittr, glue and more recently, tidymodels.
- Eight of these packages form the **core tidyverse**.



- `library(tidyverse)` is a short hand for `library(ggplot2)`, `library(dplyr)`, ..., `library(forcats)`

# Base R and Tidyverse

- Tidyverse is not a substitute for Base R 
- Knowing Base R is essential to use Tidyverse effectively 
- Data wrangling in Tidyverse just gives you a different flavour of how you can do things in Base R 
- All data wrangling can be achieved using Base R, so why load extra package(s) to deal with the data wrangling? 
- Tidyverse packages share a common design philosophy, grammar and data structure
- This trains your mental model to do data science tasks in a certain manner which may make it easier, faster, and/or fun for you to do these tasks
- It's an opinionated approach so you should make a decision for yourself of what works for you *and* others that you share your work with

# data.frame Base R

- In R, `data.frame` is a special class of `list`
- Every column in the `data.frame` is a vector of the same length
- Each entry in a vector have the same type, e.g. logical, integer, double (or numeric), character or factor.
- It has an attribute `row.names` which could be a sequence of integers indexing the row number or some unique identifier

```
cars
##      speed dist
## 1         4    2
## 2         4   10
## 3         7    4
## 4         7   22
```

```
mtcars
##      mpg cyl  disp  hp drat   wt  qsec vs
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46 0
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02 0
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61 1
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44 1
```

# Tidy data

🎯 The typical aim of data wrangling is to make data into a **tidy data**

i

## Definition of a tidy data

- Each variable must have its own column
- Each observation must have its own row
- Each value must have its own cell

country	date	cases	deaths
USA	20/01/24	14,065	57
India	20/01/24	37,29	48
Brazil	20/01/24	16,03	34
France	20/01/24	4,52	50
Russia	20/01/24	25,73	1

variables

country	date	cases	deaths
←→	←→	←→	←→
←→	←→	←→	←→
←→	←→	←→	←→
←→	←→	←→	←→
←→	←→	←→	←→

observations

country	date	cases	deaths
○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

values

# Data wrangling with Base R

# Subsetting by column Base R Part 1

## By column names

```
mtcars[, c("mpg", "cyl")]
```

```
##           mpg  cyl
## Mazda RX4    21.0   6
## Mazda RX4 Wag 21.0   6
## Datsun 710    22.8   4
## Hornet 4 Drive 21.4   6
## Hornet Sportabout 18.7   8
## Valiant      18.1   6
## Duster 360   14.3   8
## Merc 240D    24.4   4
## Merc 230     22.8   4
## Merc 280     19.2   6
## Merc 280C    17.8   6
## Merc 450SE   16.4   8
```

## By index

```
mtcars[, 1:2]
```

```
##           mpg  cyl
## Mazda RX4    21.0   6
## Mazda RX4 Wag 21.0   6
## Datsun 710    22.8   4
## Hornet 4 Drive 21.4   6
## Hornet Sportabout 18.7   8
## Valiant      18.1   6
## Duster 360   14.3   8
## Merc 240D    24.4   4
## Merc 230     22.8   4
## Merc 280     19.2   6
## Merc 280C    17.8   6
## Merc 450SE   16.4   8
```



# Subsetting by column Base R Part 2

## By column names

```
mtcars[c("mpg", "cyl")]
```

```
##           mpg  cyl
## Mazda RX4    21.0   6
## Mazda RX4 Wag 21.0   6
## Datsun 710    22.8   4
## Hornet 4 Drive 21.4   6
## Hornet Sportabout 18.7   8
## Valiant      18.1   6
## Duster 360    14.3   8
## Merc 240D     24.4   4
## Merc 230      22.8   4
## Merc 280      19.2   6
## Merc 280C     17.8   6
## Merc 450SE    16.4   8
```

## By index

```
mtcars[1:2]
```

```
##           mpg  cyl
## Mazda RX4    21.0   6
## Mazda RX4 Wag 21.0   6
## Datsun 710    22.8   4
## Hornet 4 Drive 21.4   6
## Hornet Sportabout 18.7   8
## Valiant      18.1   6
## Duster 360    14.3   8
## Merc 240D     24.4   4
## Merc 230      22.8   4
## Merc 280      19.2   6
## Merc 280C     17.8   6
## Merc 450SE    16.4   8
```

# Lists Base R

- Remember, `data.frame` is just a special type of `list` and inherit methods applied for `list`.

```
x <- list( int = 1:3,  
          shop = c("apple", "pear"),  
          c(2020, 12, 1, 1.3))
```

x

```
## $int  
## [1] 1 2 3  
##  
## $shop  
## [1] "apple" "pear"  
##  
## [[3]]  
## [1] 2020.0 12.0 1.0 1.3
```

```
x[c("int", "shop")]
```

```
## $int  
## [1] 1 2 3  
##  
## $shop  
## [1] "apple" "pear"
```

Double square bracket to access inside the list

```
x[["int"]] # or x$int
```

```
## [1] 1 2 3
```

# list and data.frame Base R

```
x[, "int"] # this is a special operation for data frames
```

```
## Error in x[, "int"]: incorrect number of dimensions
```

```
mtcars[["cyl"]]
```

```
## [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

```
mtcars$cyl
```

```
## [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

```
mtcars["cyl"]
```

```
##           cyl
## Mazda RX4      6
## Mazda RX4 Wag  6
## Datsun 710     4
```



data.frame inherits methods for list, but list does not inherit methods for data.frame

# Subsetting by column Base R

- If you subset *a single column* using `[, ]`, then by default the output is a vector and not a `data.frame`.

```
mtcars[, "mpg"]  
  
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10  
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19  
## [31] 15.0 21.4
```

- If you want to preserve the output as a `data.frame` then:

```
mtcars[, "mpg", drop = FALSE]  
  
##           mpg  
## Mazda RX4  21.0  
## Mazda RX4 Wag 21.0  
## Datsun 710  22.8
```

# Subsetting by row Base R Part 1

By index:

```
mtcars[3:1, ]
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Datsun 710 22.8   4  108   93 3.85 2.320 18.61  1  1    4    1
## Mazda RX4 Wag 21.0   6  160  110 3.90 2.875 17.02  0  1    4    4
## Mazda RX4   21.0   6  160  110 3.90 2.620 16.46  0  1    4    4
```

If it has row names:

```
mtcars[c("Datsun 710", "Mazda RX4"), ]
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Datsun 710 22.8   4  108   93 3.85 2.32 18.61  1  1    4    1
## Mazda RX4  21.0   6  160  110 3.90 2.62 16.46  0  1    4    4
```

# Subsetting by row Base R Part 2

Using a logical vector,

```
mtcars[mtcars$mpg > 31, ]
```

```
##           mpg  cyl  disp  hp  drat    wt   qsec  vs  am  gear  carb
## Fiat 128    32.4   4  78.7  66  4.08  2.200 19.47  1  1    4    1
## Toyota Corolla 33.9   4  71.1  65  4.22  1.835 19.90  1  1    4    1
```

Or using **non-standard evaluation**:

```
subset(mtcars, mpg > 31)
```

```
##           mpg  cyl  disp  hp  drat    wt   qsec  vs  am  gear  carb
## Fiat 128    32.4   4  78.7  66  4.08  2.200 19.47  1  1    4    1
## Toyota Corolla 33.9   4  71.1  65  4.22  1.835 19.90  1  1    4    1
```

Non-standard evaluation? We'll come back to this later.

# Adding or modifying a column Base R

Append the new column:

```
df1 <- cbind(mtcars, gpm = 1 / mtcars$mpg)
```

Create a new column as you would create a new object in a list:

```
df1$gpm <- 1 / df1$mpg  
df1[["gpm"]] <- 1 / df1$mpg
```

Overwrite column if modifying an existing column:

```
df1$wt <- df1$gpm^2  
df1[["wt"]] <- df1$gpm^2
```

Modify only a part of it:

```
df1$wt[df1$cyl==6] <- 10
```

# Adding a row Base R

What do you notice with the order of the new entry?

```
df2 <- rbind(cars, data.frame(dist = 10, speed = 3))  
tail(df2, 3)
```

```
##      speed dist  
## 49      24  120  
## 50      25   85  
## 51       3   10
```

```
df2 <- rbind(cars, c(10, 3))  
tail(df2, 3)
```

```
##      speed dist  
## 49      24  120  
## 50      25   85  
## 51      10    3
```



# Sorting columns Base R

```
mtcars[, sort(names(mtcars))]
```

```
##           am carb  cyl  disp drat gear  hp  mpg  qsec vs   wt
## Mazda RX4      1   4   6 160.0 3.90   4 110 21.0 16.46  0 2.620
## Mazda RX4 Wag  1   4   6 160.0 3.90   4 110 21.0 17.02  0 2.875
## Datsun 710      1   1   4 108.0 3.85   4  93 22.8 18.61  1 2.320
## Hornet 4 Drive  0   1   6 258.0 3.08   3 110 21.4 19.44  1 3.215
## Hornet Sportabout 0   2   8 360.0 3.15   3 175 18.7 17.02  0 3.440
## Valiant        0   1   6 225.0 2.76   3 105 18.1 20.22  1 3.460
## Duster 360     0   4   8 360.0 3.21   3 245 14.3 15.84  0 3.570
## Merc 240D      0   2   4 146.7 3.69   4  62 24.4 20.00  1 3.190
## Merc 230       0   2   4 140.8 3.92   4  95 22.8 22.90  1 3.150
## Merc 280       0   4   6 167.6 3.92   4 123 19.2 18.30  1 3.440
## Merc 280C      0   4   6 167.6 3.92   4 123 17.8 18.90  1 3.440
## Merc 450SE     0   3   8 275.8 3.07   3 180 16.4 17.40  0 4.070
## Merc 450SL     0   3   8 275.8 3.07   3 180 17.3 17.60  0 3.730
```

# Sorting rows Base R

```
order(mtcars$mpg)
```

```
## [1] 15 16 24 7 17 31 14 23 22 29 12 13 11 6 5 10 25 30 1 2 4 32 21 3  
## [26] 8 27 26 19 28 18 20
```

```
mtcars[order(mtcars$mpg),]
```

```
##           mpg  cyl  disp  hp  drat    wt    qsec vs  am  gear  carb  
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98 0  0    3    4  
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82 0  0    3    4  
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41 0  0    3    4  
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84 0  0    3    4  
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42 0  0    3    4  
## Maserati Bora        15.0   8 301.0 335 3.54 3.570 14.60 0  1    5    8  
## Merc 450SLC          15.2   8 275.8 180 3.07 3.780 18.00 0  0    3    3  
## AMC Javelin          15.2   8 304.0 150 3.15 3.435 17.30 0  0    3    2  
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87 0  0    3    2
```

# Calculating statistical summaries by group Base R

🎯 Calculate the *average* weight (wt) of a car for each gear type in (gear) mtcars

```
tapply(mtcars$wt, mtcars$gear, mean)
```

```
##           3           4           5  
## 3.892600 2.616667 2.632600
```

🎯 Calculate the *median* weight (wt) of a car for each gear (gear) and engine (vs) type in mtcars

```
tapply(mtcars$wt, list(mtcars$gear, mtcars$vs), median)
```

```
##           0           1  
## 3 3.8100 3.215  
## 4 2.7475 2.550  
## 5 2.9700 1.513
```

**</> If you installed the `dwexercise` package,  
run below in your R console**

```
learnr::run_tutorial("day1-exercise-01", package = "dwexercise")
```

**🔗 If the above doesn't work for you, go [here](#).**  
**? Questions or issues, let us know!**

**15:00**

# Session Information

```
devtools::session_info()
```

```
## - Session info -----  
## setting value  
## version R version 4.0.1 (2020-06-06)  
## os      macOS Catalina 10.15.7  
## system x86_64, darwin17.0  
## ui      X11  
## language (EN)  
## collate en_AU.UTF-8  
## ctype   en_AU.UTF-8  
## tz      Australia/Melbourne  
## date    2020-11-30  
##  
## - Packages -----  
## package * version date      lib source  
## anicon   0.1.0   2020-06-21 [1] Github (emitanaka/anicon@0b756df)  
## assertthat 0.2.1  2019-03-21 [2] CRAN (R 4.0.0)
```

These slides are licensed under

