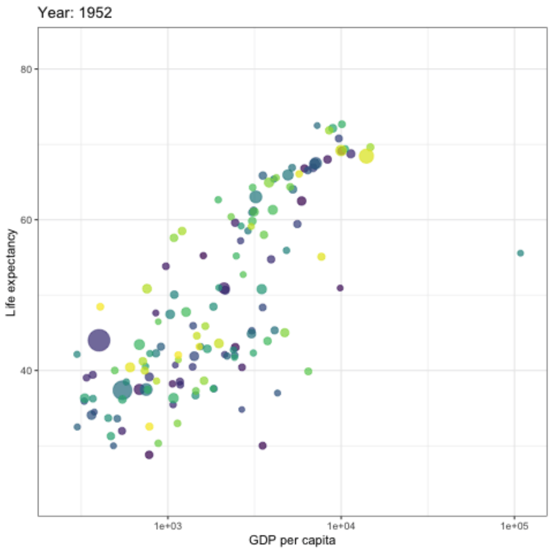
Introduction to Tidyverse

Dani Chu and Lucas Wu

Why R?

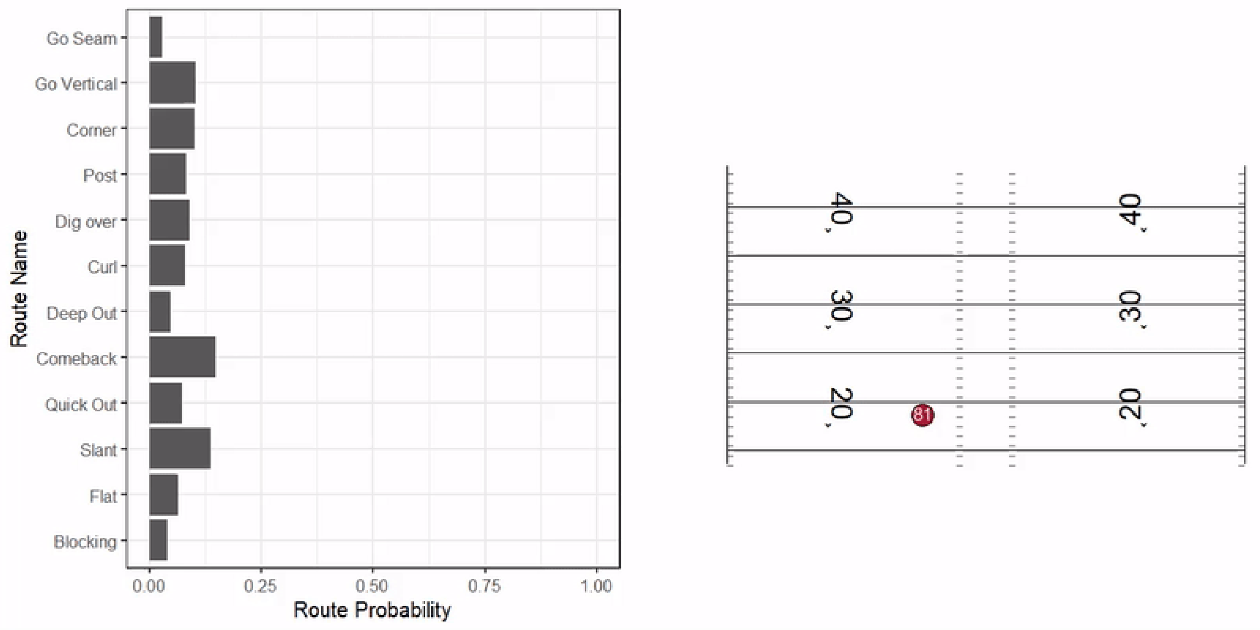
GDP vs life expectancy over time



Source: kassambara

Player Movement in NBA

Route identification in NFL

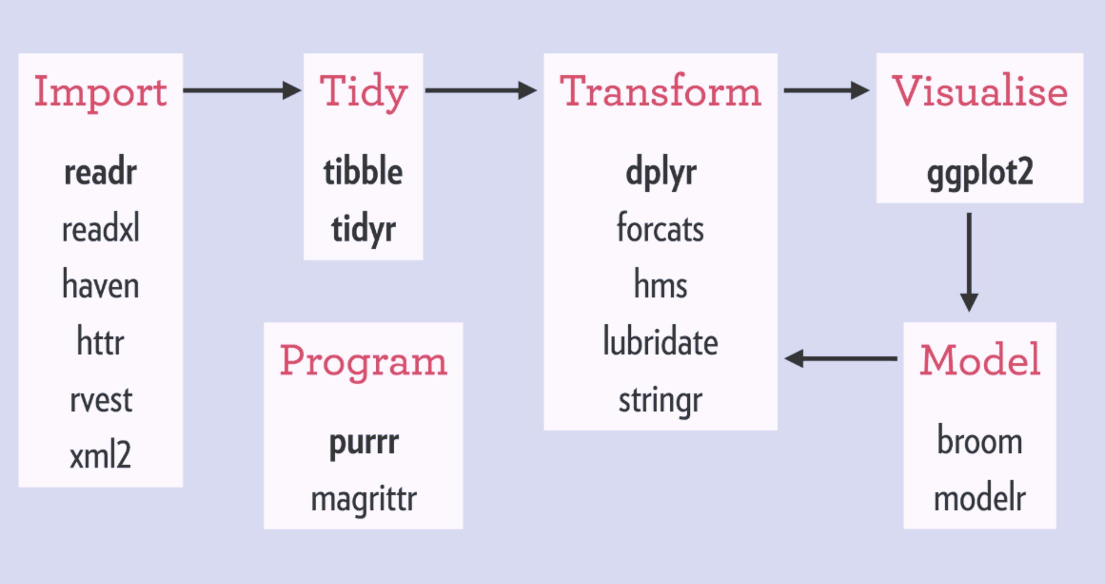


Source: [Dani Chu, Matthew Reyers, James Thomson and Lucas Wu]

What is tidyverse?

- A coherent collection of packages for data science, including ggplot2, dplyr, tidyr, readr and stringr etc.
- Extremely powerful for data manipulation, exploration and visualization
- Write codes like reading a sequence of tasks via the pipe operator
- In Statistics, we typically assume data is "tidy"
  - data is in a tabular form
  - 1 row == 1 observation
  - 1 column == 1 variable

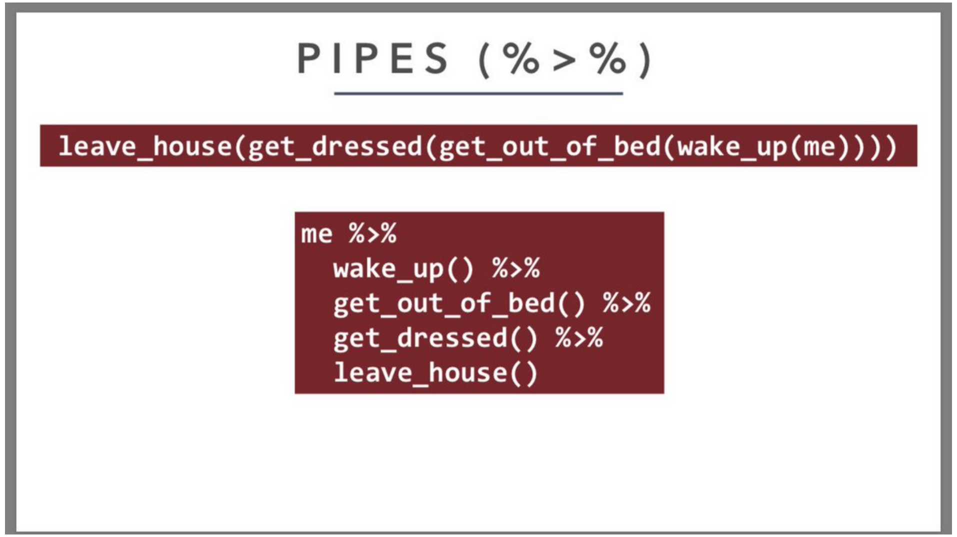
Tidyverse process



Source: Joseph Rickert

What is a pipe operator in Tidyverse?

- %>%
- Used to perform a sequence of tasks



PIPES (%>%)

```
leave_house(get_dressed(get_out_of_bed(wake_up(me))))
```

```
me %>%
  wake_up() %>%
  get_out_of_bed() %>%
  get_dressed() %>%
  leave_house()
```

Source: Andrew Heiss, @andrewheiss

What is a pipe operator in Tidyverse?

- Used to perform sequential tasks
- `f(x)` can be rewritten as `x %>% f`
- `f(x, y)` can be rewritten as `x %>% f(y)`
- Tip: use ctrl + shift + m as RStudio shortcut

```r
library(tidyverse)
x <- c(1.2, 3.3, 5.4)
# have to nest a lot of parentheses together
round(mean(x),0)
```

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

```r
# pipe operator makes it cleaner!
x %>%
  mean() %>%
  round(0)
```

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

What is a pipe operator in Tidyverse?

- `f(x, y)` can be rewritten as `x %>% f(y)`
- x %>% f(y) is equivalent to

```
x %>% f(., y)
```

- dot notation allows you to pass in a variable as the second or third argument to a function

```
x %>% f(y, .)
```

- which is equivalent to f(y,x)

Source: Most of following slides are adapted from Ryan Tibshirani

Basic functions of dplyr

- Some basic functions:
  - `select()`: select columns
  - `slice()`: subset rows based on index
  - `filter()`: subset rows based on conditions
  - `mutate()`: create new columns
  - `arrange()`: order rows
  - `rename()`: rename columns
  - `group_by()`: allows for group operations
  - `summarise()`: summarise values, often used together with `group_by()`

Data set

- Major League Baseball Data from the 1986 and 1987 seasons available from ISLR package
- We only select the first 10 players and 6 variables as a case study

	Hits	CHits	HmRun	Years	Salary	Division
-Alan Ashby	81	835	7	14	475.000	W
-Alvin Davis	130	457	18	3	480.000	W
-Andre Dawson	141	1575	20	11	500.000	E
-Andres Galarraga	87	101	10	2	91.500	E
-Alfredo Griffin	169	1133	4	11	750.000	W
-Al Newman	37	42	1	2	70.000	E
-Argenis Salazar	73	108	0	3	100.000	W
-Andres Thomas	81	86	6	2	75.000	W
-Andre Thornton	92	1332	17	13	1100.000	E
-Alan Trammell	159	1300	21	10	517.143	E

Source: Hitters data description

select()

- Use select() when you want to pick out certain columns:

```r
library(ISLR)
df = Hitters %>%
  na.omit() %>%
  head(10) %>%
  select( Hits, CHits, HmRun,
          Years, Salary, Division)

df %>%
  select(., CHits, Years, Salary) %>%
  head(., 2)
```

```
##                CHits Years Salary
## -Alan Ashby      835    14    475
## -Alvin Davis     457     3    480
```

slice()

- Use slice() when you want to indicate certain row numbers need to be kept:

```
df %>%
  select(., CHits, Years, Salary) %>%
  slice(c(1:3,7,8))
```

```
##   CHits Years Salary
## 1   835    14    475
## 2   457     3    480
## 3  1575    11    500
## 4   108     3    100
## 5    86     2     75
```

filter()

- Use filter() when you want to subset rows based on logical conditions:

```
df %>%
  filter(., (Years >= 10 &
               Hits >= 100))
```

```
##   Hits CHits HmRun Years  Salary Division
## 1  141  1575    20    11 500.000        E
## 2  169  1133     4    11 750.000        W
## 3  159  1300    21    10 517.143        E
```

mutate()

- Use mutate() when you want to create one or several columns:

```
df %>%
  select(CHits, Years) %>%
  mutate(.,
    hits_per_year = CHits/Years,
    player_name = rownames(df))
```

```
##    CHits Years hits_per_year       player_name
## 1    835    14      59.64286        -Alan Ashby
## 2    457     3     152.33333       -Alvin Davis
## 3   1575    11     143.18182      -Andre Dawson
## 4    101     2      50.50000 -Andres Galarraga
## 5   1133    11     103.00000  -Alfredo Griffin
## 6     42     2      21.00000         -Al Newman
## 7    108     3      36.00000  -Argenis Salazar
## 8     86     2      43.00000    -Andres Thomas
## 9   1332    13     102.46154   -Andre Thornton
## 10  1300    10     130.00000    -Alan Trammell
```

arrange()

- Use arrange() to order rows by values of a column:

```
df %>%
  arrange(., desc(Salary))
```

```
##    Hits CHits HmRun Years   Salary Division
## 1    92  1332    17    13 1100.000        E
## 2   169  1133     4    11  750.000        W
## 3   159  1300    21    10  517.143        E
## 4   141  1575    20    11  500.000        E
## 5   130   457    18     3  480.000        W
## 6    81   835     7    14  475.000        W
## 7    73   108     0     3  100.000        W
## 8    87   101    10     2   91.500        E
## 9    81    86     6     2   75.000        W
## 10   37    42     1     2   70.000        E
```

rename()

- Use rename() to easily rename columns:

```
df %>%
  select(.,CHits, Salary) %>%
  rename(career_hits = CHits,
          salary = Salary) %>%
  head()
```

```
##                    career_hits salary
## -Alan Ashby                835  475.0
## -Alvin Davis               457  480.0
## -Andre Dawson             1575  500.0
## -Andres Galarraga          101   91.5
## -Alfredo Griffin          1133  750.0
## -Al Newman                  42   70.0
```

group_by()

- Use group_by() to define a grouping of rows based on a column:
- Note that this does not change anything about the dataframe
- Only difference is that when it prints, we’re told about the groups

```
df %>%
    group_by(., Division)
```

```
## # A tibble: 10 x 6
## # Groups:   Division [2]
##      Hits CHits HmRun Years Salary Division
##   * <int> <int> <int> <int>  <dbl> <fct>
##  1    81   835     7    14  475   W
##  2   130   457    18     3  480   W
##  3   141  1575    20    11  500   E
##  4    87   101    10     2   91.5 E
##  5   169  1133     4    11  750   W
##  6    37    42     1     2   70   E
##  7    73   108     0     3  100   W
##  8    81    86     6     2   75   W
##  9    92  1332    17    13 1100   E
## 10   159  1300    21    10  517.  E
```

summarise()

- use summarise() to apply functions to rows—ungrouped or grouped—of a data frame:

```
# Ungrouped
df %>%
  summarise(.,
      avg_career_hits=mean(CHits),
      avg_salary=mean(Salary))

# # Grouped by Division
df %>%
  group_by(., Division) %>%
  summarise(.,
      avg_career_hits=mean(CHits),
      avg_salary=mean(Salary))
```

```
##   avg_career_hits avg_salary
## 1           696.9   415.8643

## # A tibble: 2 x 3
##   Division avg_career_hits avg_salary
##   <fct>              <dbl>      <dbl>
## 1 E                    870       456.
## 2 W                    524.      376
```

Your time!

Check out the following links if you would like to learn more:

- https://www.tidyverse.org/learn/
- https://www.datanovia.com/en/lessons/select-data-frame-columns-in-r/