

STAT3622 Data Visualization (Lecture 2)

Data Manipulation

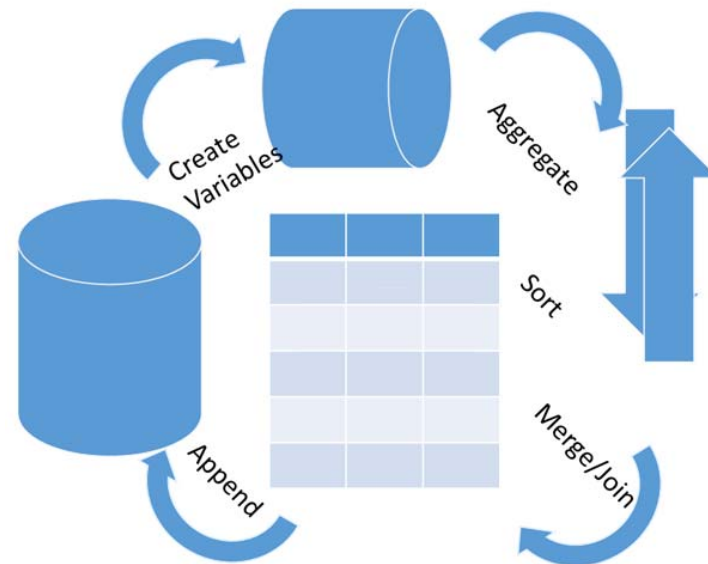
Dr. Yiwei Fan

The University of Hong Kong

25 January 2021

What's covered in this lecture?

- Data Manipulation
 - Data Wrangling
 - Feature Engineering
- R:dplyr Package
- Pipes %>%

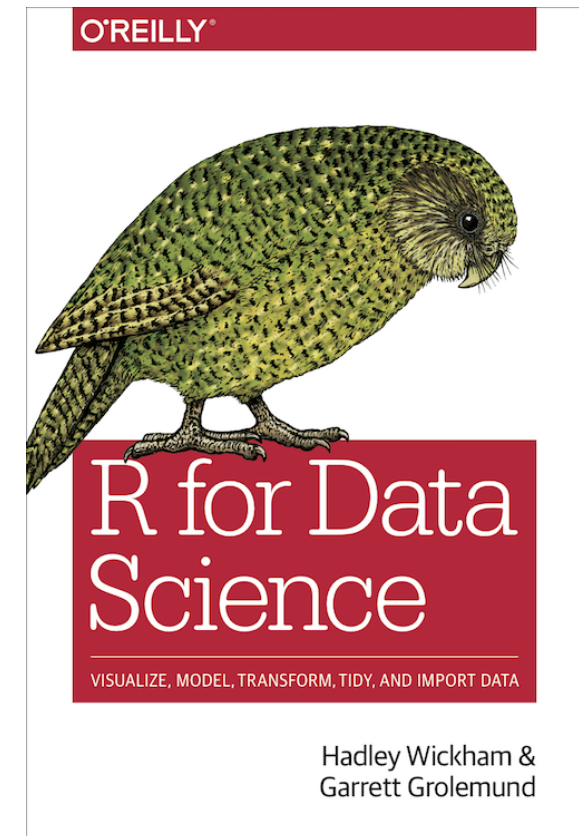


dplyr cheatsheet, Sparklyr, data.table

1. Data Manipulation

Data Exploration

- R for Data Science (O'Reilly 2017) by Hadley Wickham
- Free Online: <http://r4ds.had.co.nz/>
- Major coverage:
 - Data manipulation (R:dplyr)
 - Data visualization (R:ggplot2)
- Data Exploration = Data manipulation + Data visualization.



Data Wrangling

- Data wrangling (a term used in data science) is the process of transforming/mapping data from raw format into ready-to-analyze format.
- Besides ggplot2() for data visualization, Hadley Wickham has created a series of R packages for data wrangling, including
 - tidyr for tidy data: observations in rows, variables in columns
 - tibble for better ways to create, print and subset data frames
 - dplyr for data manipulation -> today
- Refer to the ggplot2 book Chapter 9 about Tidy Data; and Chapter 10 about dplyr.

Feature Engineering

- A term often used in machine learning

Andrew Ng (Stanford): "Coming up with features is difficult, time-consuming, requires expert knowledge. *Applied machine learning* is basically feature engineering."

- In statistics: variable creation and transformation
- Dictionary learning with overcomplete features ...
- Nowadays, deep learning algorithms aim at automatic feature learning instead of manual feature engineering ...

2. R:dplyr Package

R::dplyr verbs

- `filter()` to select observations
- `arrange()` to order observations
- `mutate()` to add new variables
- `group_by()` to group variables for summarise
- `R::base:merge()` to combine two data.frames (or `R::dplyr xxx_joins`)

Filter

```
library(dplyr)
(tmp = filter(iris, Species == 'versicolor' & Sepal.Length > 6.6))
```

| ## | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|------|--------------|-------------|--------------|-------------|------------|
| ## 1 | 7.0 | 3.2 | 4.7 | 1.4 | versicolor |
| ## 2 | 6.9 | 3.1 | 4.9 | 1.5 | versicolor |
| ## 3 | 6.7 | 3.1 | 4.4 | 1.4 | versicolor |
| ## 4 | 6.8 | 2.8 | 4.8 | 1.4 | versicolor |
| ## 5 | 6.7 | 3.0 | 5.0 | 1.7 | versicolor |
| ## 6 | 6.7 | 3.1 | 4.7 | 1.5 | versicolor |

- Rowwise selection of samples/observations
- Similar to base:which or subsetting

Arrange

```
arrange(tmp, Sepal.Length, Sepal.Width, desc(Petal.Length))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1         6.7         3.0         5.0         1.7 versicolor
## 2         6.7         3.1         4.7         1.5 versicolor
## 3         6.7         3.1         4.4         1.4 versicolor
## 4         6.8         2.8         4.8         1.4 versicolor
## 5         6.9         3.1         4.9         1.5 versicolor
## 6         7.0         3.2         4.7         1.4 versicolor
```

- Similar to base:sort and order functions

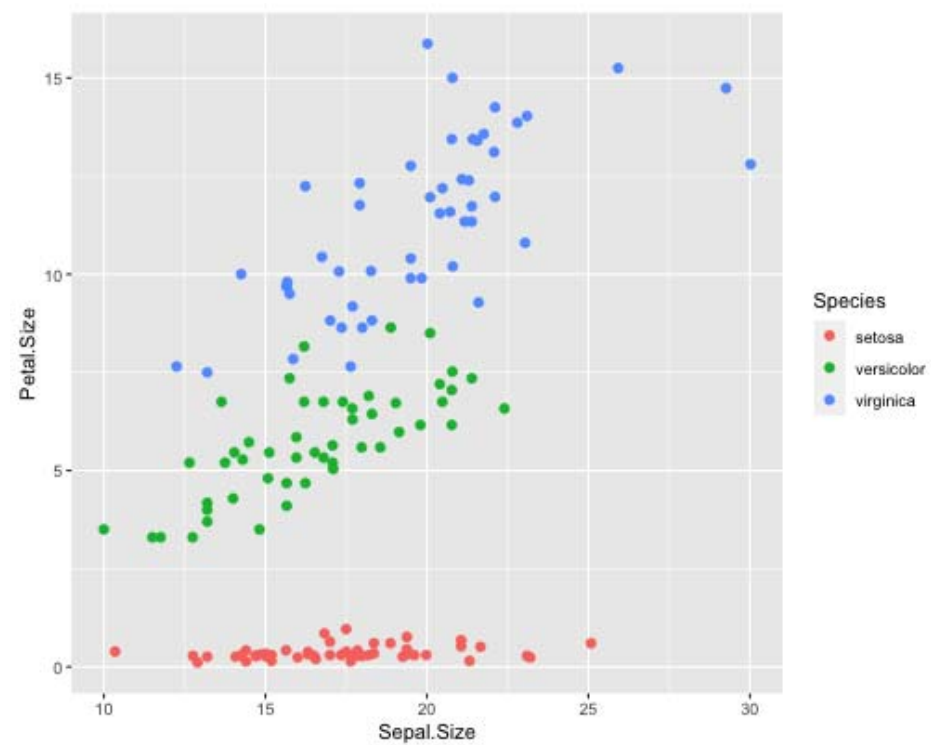
Mutate

```
tmp = mutate(iris, Sepal.Size = Sepal.Length*Sepal.Width,  
              Petal.Size = Petal.Length*Petal.Width)  
head(tmp)
```

| ## | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species | Sepal.Size | Petal.Size |
|------|--------------|-------------|--------------|-------------|---------|------------|------------|
| ## 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa | 17.85 | 0.28 |
| ## 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa | 14.70 | 0.28 |
| ## 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa | 15.04 | 0.26 |
| ## 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa | 14.26 | 0.30 |
| ## 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa | 18.00 | 0.28 |
| ## 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa | 21.06 | 0.68 |

- New variable/feature creation
- Base commands: tmp\$Sepal.Size = ...

```
library(ggplot2)
ggplot(tmp, aes(x=Sepal.Size, y=Petal.Size, colour=Species)) +
  geom_point(size=2)
```



Summarise

```
summarise(group_by(tmp, Species), mean(Sepal.Size), mean(Petal.Size))
```

```
## # A tibble: 3 x 3
##   Species    `mean(Sepal.Size)` `mean(Petal.Size)`
##   <fct>          <dbl>          <dbl>
## 1 setosa          17.3            0.366
## 2 versicolor      16.5            5.72
## 3 virginica       19.7           11.3
```

- Together with the `group_by` variable

Merge

```
(tmp1 = data.frame(Species=levels(iris$Species), x1 = c("A", "B", "C"), x2 = round(runif(3), 3)))
```

```
##      Species x1    x2
## 1    setosa  A 0.801
## 2 versicolor B 0.541
## 3  virginica C 0.830
```

```
head(merge(iris, tmp1, by = "Species"))
```

```
##   Species Sepal.Length Sepal.Width Petal.Length Petal.Width x1    x2
## 1  setosa         5.1         3.5         1.4         0.2  A 0.801
## 2  setosa         4.9         3.0         1.4         0.2  A 0.801
## 3  setosa         4.7         3.2         1.3         0.2  A 0.801
## 4  setosa         4.6         3.1         1.5         0.2  A 0.801
## 5  setosa         5.0         3.6         1.4         0.2  A 0.801
## 6  setosa         5.4         3.9         1.7         0.4  A 0.801
```

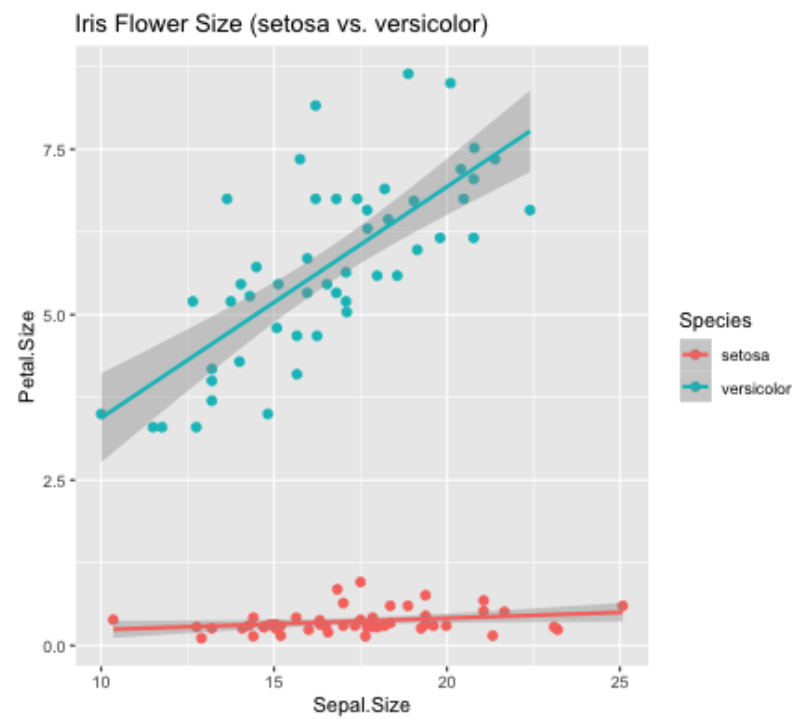
- I personally prefer base:merge() than dplyr:left/right/inner/full/etc_join functions

3. Pipes %>%

Lastly, Pipes %>%

- The pipe %>% requires R package dplyr or magrittr
- Powerful trick for coding a sequence of operations
- Output of old operation as the first argument of new operation
- Especially useful in combined with ggplot2
- Let's demonstrate it with Iris dataset ...


```
iris %>%
  mutate(Sepal.Size = Sepal.Length*Sepal.Width,
         Petal.Size = Petal.Length*Petal.Width) %>%
  filter(Species != 'virginica') %>%
  ggplot(aes(x=Sepal.Size, y=Petal.Size, colour=Species)) +
  geom_point(size=2) +
  geom_smooth(method = "lm") +
  labs(title = 'Iris Flower Size (setosa vs. versicolor)')
```



Thank you!