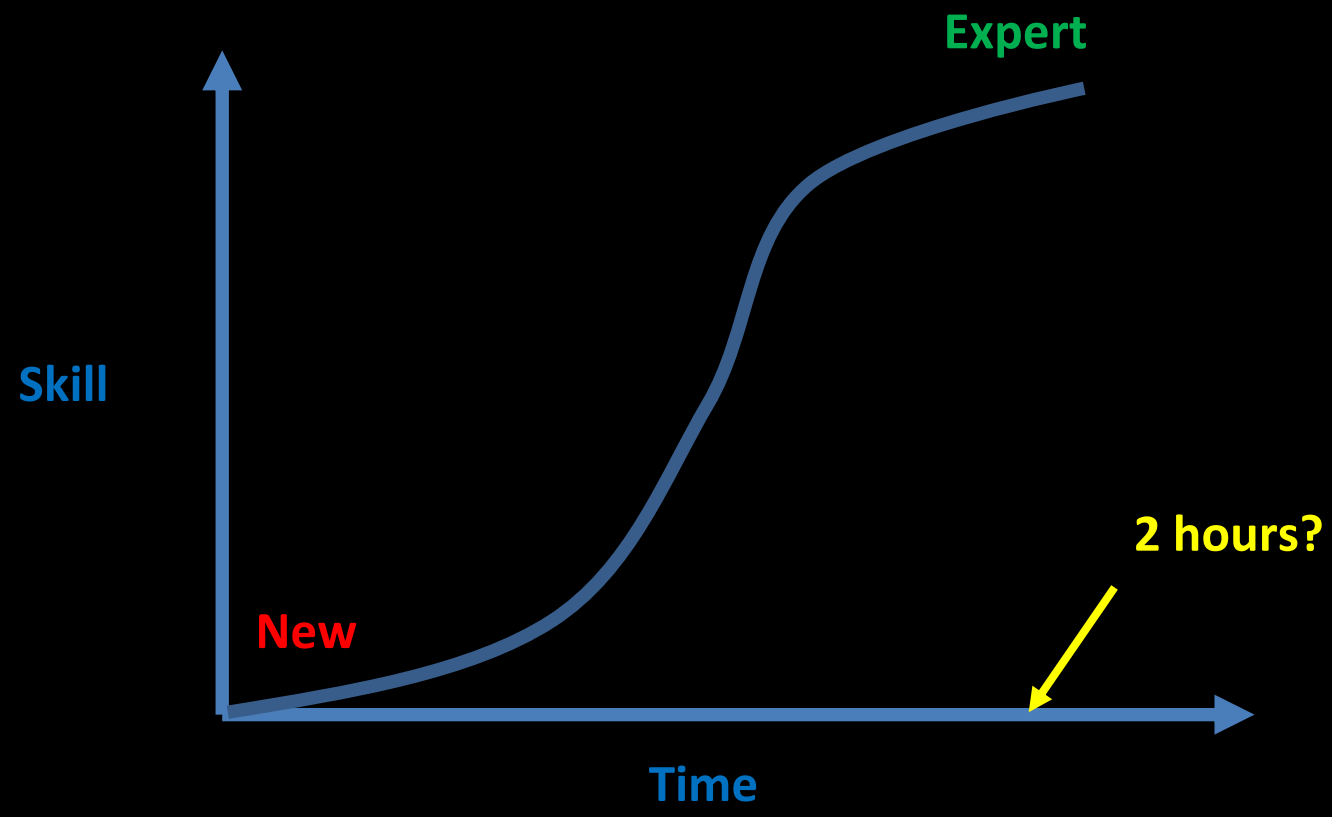
一起学

Learn SQL+R from beginning to expert



Programming: learn SQL + R/Python

/

- Language to query from databases
- SQL like thinking is critical in programming

- Very good statistics packages
- Very good visualization packages
- Dplyr!!

- Accepted by CS
- Closest language to C++/Java
- Decent amount of packages

Goal of a good programmer: 指哪打哪

- It is fine to Google
- You need to write short code
- Code as you think

Key areas planed to cover in R

- Basic syntax
- How to write all codes with data frame (dplyr)
- Best tool for visualization (ggplot2)
- How to avoid for loop
- How to validate

```
msleep %>%
  group_by(vore) %>%
  summarise_at(vars(contains("sleep")), mean, na.rm=TRUE) %>%
  rename_at(vars(contains("sleep")), ~paste0("avg_", .))

## # A tibble: 5 x 4
##   vore    avg_sleep_total avg_sleep_rem avg_sleep_cycle
##   <chr>             <dbl>         <dbl>           <dbl>
## 1 carni              10.4          2.29           0.373
## 2 herbi               9.51         1.37           0.418
## 3 insecti            14.9          3.52           0.161
## 4 omni               10.9          1.96           0.592
## 5 <NA>               10.2          1.88           0.183
```

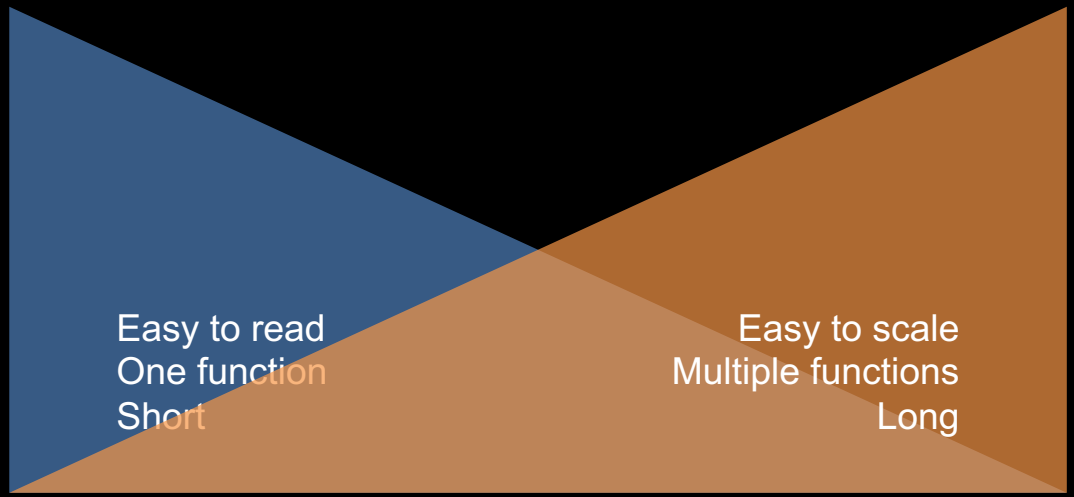
What to expect today

- R basics
- dplyr
- ggplot
- reshape
- QUIZ
- dplyr and SQL
- Coding speed
- Demo and practice

- 箴11:14 “无智谋，民就败落；谋士多；人便安居。”
- 箴12:15 “愚妄人所行的，在自己眼中看为正直；惟智慧人肯听人的劝教。”
- 箴13:10 “骄傲只启争竞；听劝言的，却有智慧。”
- 箴15:22 “不先商议，所谋无效；谋士众多，所谋乃成。”
- 箴19:20 “你要听劝教，受训诲，使你终久有智慧。”
- 箴20:18 “计谋都凭筹算立定；打仗要凭智谋。”
- 箴27:9 “膏油与香料使人心喜悦；朋友诚实的劝教，也是如此甘美。”

Coding in data science is different than software engineer

Coding in data science



Software engineer

R basics

Data types in R

- Logical: TRUE, FALSE (T, F)
- Numeric: v <- 23.5
- Character: v <- "TRUE"
- Vector: apple <- c('red','green',"yellow")
- Factors: factor(c('green','green','yellow','red','red','red','green'))
- Data frame: BMI <- data.frame(gender = c("Male", "Male","Female"), height = c(152, 171.5, 165), weight = c(81,93, 78), Age = c(42,38,26))
- List, matrices, arrays

Variables

- Naming convention: only special character dot (.) is allowed in R
- List all variables: print(ls())
- Remove variables: rm(df), rm(list=ls())

Assign

- **<-:** v <- 3.5
- **>-:** 3.5 >- v

Operations

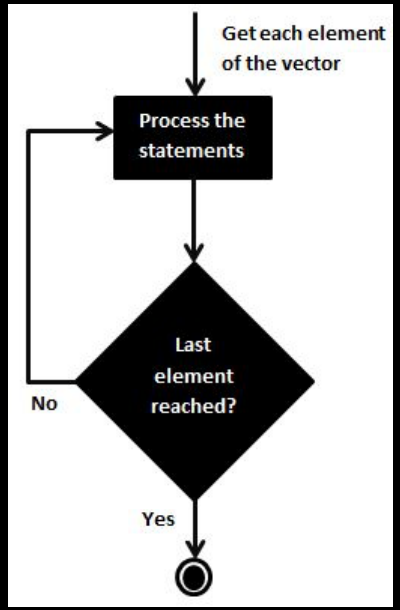
- + - * /
- %% remainder
- %/% quotient
- ^: power
- >, <, ==, <=, >=, !=
- &, |, !

Decision making in R

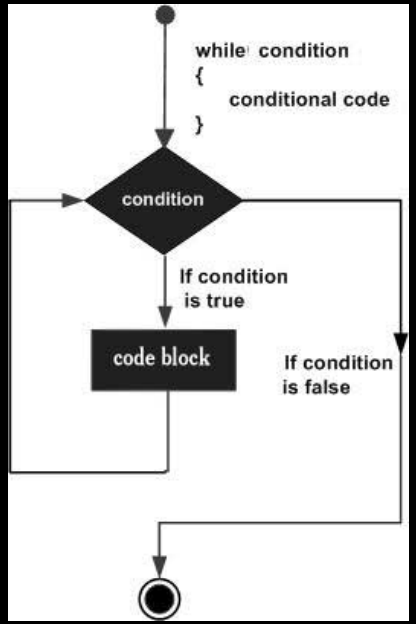
- **If else:**
- x <- 30L if(is.integer(x)) {
  print("X is an Integer") }

Loops

for (value in vector) {
statements }



while (test_expression) {
statement }



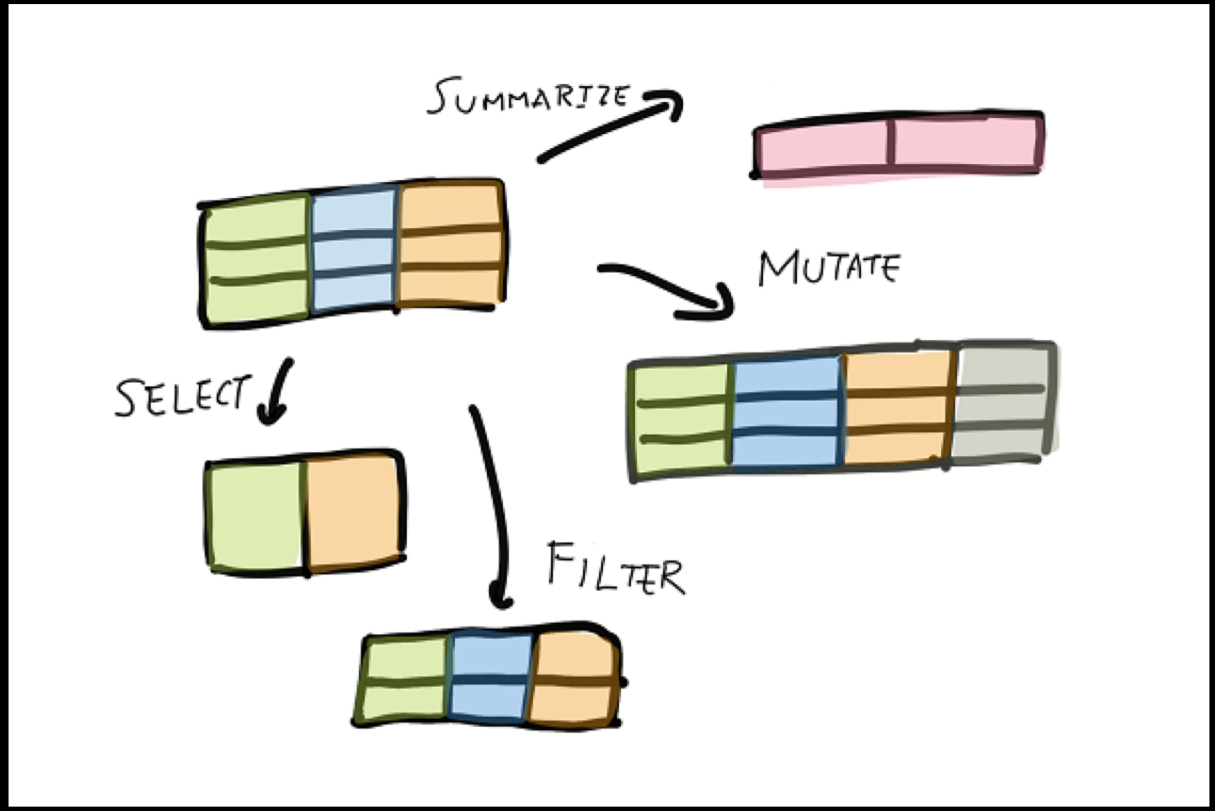
Functions

```
1  # Create a function to print squares of numbers in sequence.
2  new.function <- function(a) {
3      for(i in 1:a) {
4          b <- i^2
5          print(b)
6      }
7  }
8
9  # Call the function new.function supplying 6 as an argument.
10 new.function(6)
```

source(file_directory)

dplyr

Matrix (data frame) thinking is the most important



Data frame

ID	Gender	Height
1	Female	5
2	Male	6
3	Male	NA

Create a data frame

```
df <- data.frame(ID = 1:3,
                 Gender = c('Female', 'Male', 'Male'),
                 Hight = c(5, 6, NA))
```

Extract one column (vector)

```
df[, 'Gender']
df$Gender
```

Extract two columns (data frame)

```
df[, c('Gender', 'Height')]
```

Column names as a vector

```
names(df)
```

Extract first two rows as a data frame

```
df[1:2, ]
```

dplyr is the perfect tool to match the speed of your thinking process

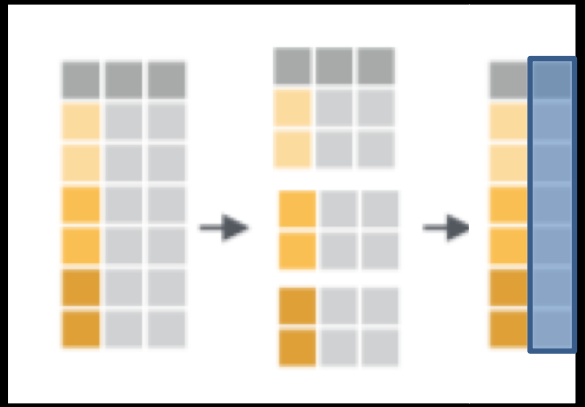
```
12  test %>% mutate(score = predictions) %>% filter(!is.na(score), !is.na(label)) %>%
13    mutate(prediction = ifelse(score > median(score), 1, 0)) %>%
14    mutate(accurate = ifelse(prediction == label, 1, 0)) %>%
15    select(label, accurate, prediction) %>%
16    summarise(accuracy = sum(accurate) /n())
```

- **Pipeline operator %>%: pass the outcome before it as the first input parameter of the function after it**

Dplyr group_by %>% summrise group_by %>% mutate

```
group_by(c1) %>%
summarise(avg=mean(c1),
max = max(c1))
```

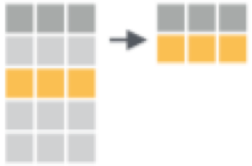
```
group_by(c1) %>%
mutate(avg=mean(c1))
```



```
df %>% group_by(Gender, Class) %>%
    summarise(grade = mean(grade)) %>% ungroup()
    group_by(Gender) %>%
    summarise(best = max(grade) %>% ungroup()
```

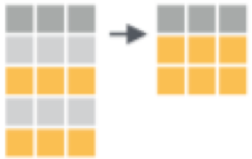
Multiple group_bys

dplyr (select by rows)



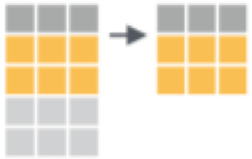
filter(.data, …**)** Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE**)** Remove rows with duplicate values.
distinct(iris, Species)



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*



slice(.data, …**)** Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt**)** Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

dplyr (select by columns)

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1**)** Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …**)**
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

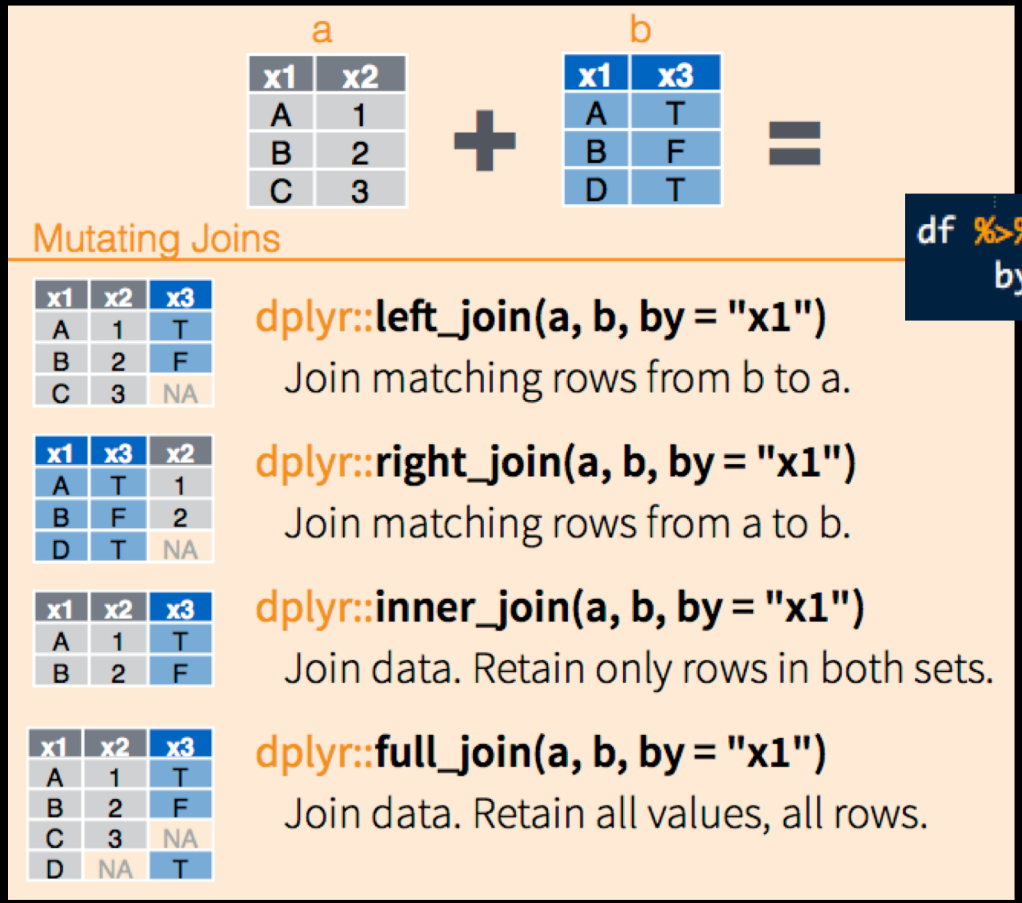
contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g. mpg:cyl
ends_with(match**)**	**one_of(**…**)**	**-**, e.g, -Species
matches(match**)**	**starts_with(**match**)**	

dplyr if_else case_when

```
if_else(x < 0, "negative", "positive")
```

```
x <- 1:50
case_when(
  x %% 35 == 0 ~ "fizz buzz",
  x %% 5 == 0 ~ "fizz",
  x %% 7 == 0 ~ "buzz",
  TRUE ~ as.character(x)
)
```

dplyr left_join inner_join




```
df %>% left_join(df2,
    by = c('Gender', 'Height'='height')
```

```
df %>% left_join(df2 %>% select(height, grade),
    by = c('Gender', 'Height'='height')
```

- Only use left_join to simplify your thinking process
- Think before you join
- Make sure check the uniqueness of the right table
- Validate the number of rows after each join

Questions

- What's the operation to collapse the matrix to a single number?
- In df <- data.frame(height = x), is x a vector or a data frame?
- What does this return df$height, a vector or a data frame?
- What does this return df %>% select(height), a vector or a data frame?
- How to return a vector in dplyr?
- If you want to add a column to a data frame without changing the number of rows, what operator(s) should you choose?
- If you want to add a column to a data frame that's based on groups of some variables, what operation(s) should you choose?
- If you want to create a new column which is based on a binary condition, what operation(s) should you use?
- If you want to select the 2 highest female and 2 highest male, what operations should you use?

Additional questions

- How to check if there are duplicates of two columns of a data frame? How to clean it before joining if there are?
- How to check the unique values of one column? Can you name two approaches?
- How to check the column names of a data frame? Name two approaches.
- Can you name three approaches to return the number of rows of a data frame?

library(reshape2)

- analysis Reshape2 conflicts with dplyr!! (run detach("package:reshape2", unload=TRUE) after using reshape2)
- always prefers long format, shift to wide format when needed

Name	Week	Distance
Coco	1	5
Coco	2	6
Coco	3	4
Coco	4	5
Yang	1	1
Yang	2	2
Yang	3	1
Yang	4	0

Long format

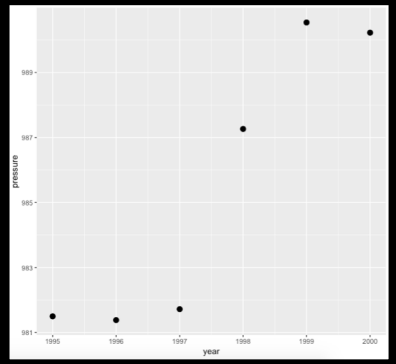
Name	Distance_ Week1	Distance_ Week2	Distance_ Week3	Distance_ Week4
Coco	5	6	4	5
Yang	1	2	1	0

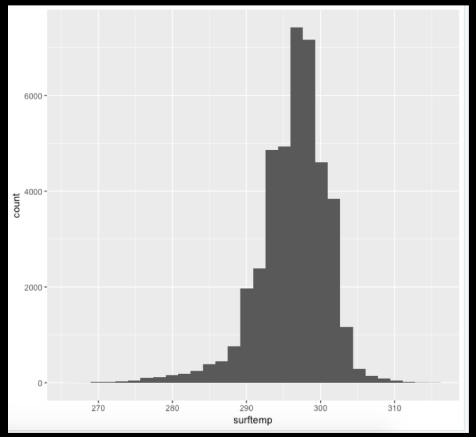
Wide format

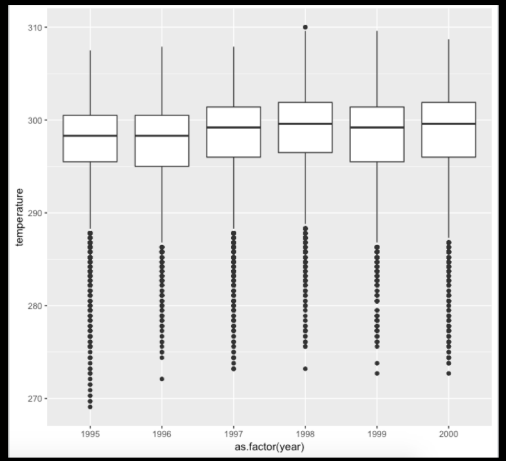
library(ggplot2)

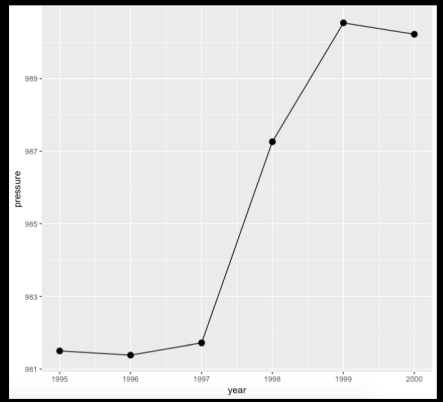
Four plotting functions

```
ggplot() + geom_point(...)
ggplot() + geom_line(...)
ggplot() + geom_histogram(...)
ggplot() + geom_boxplot(...)
```









Two/four labeling functions

```
+ scale_x_continuous()
+ scale_y_continouse()
+ scale_x_discrete()
+ scale_y_discrete()
```

Typical variables inside the functions

- **name = “Height”**
- **limits = c(5, 8)**
- **breaks = …**
- **labels = …**

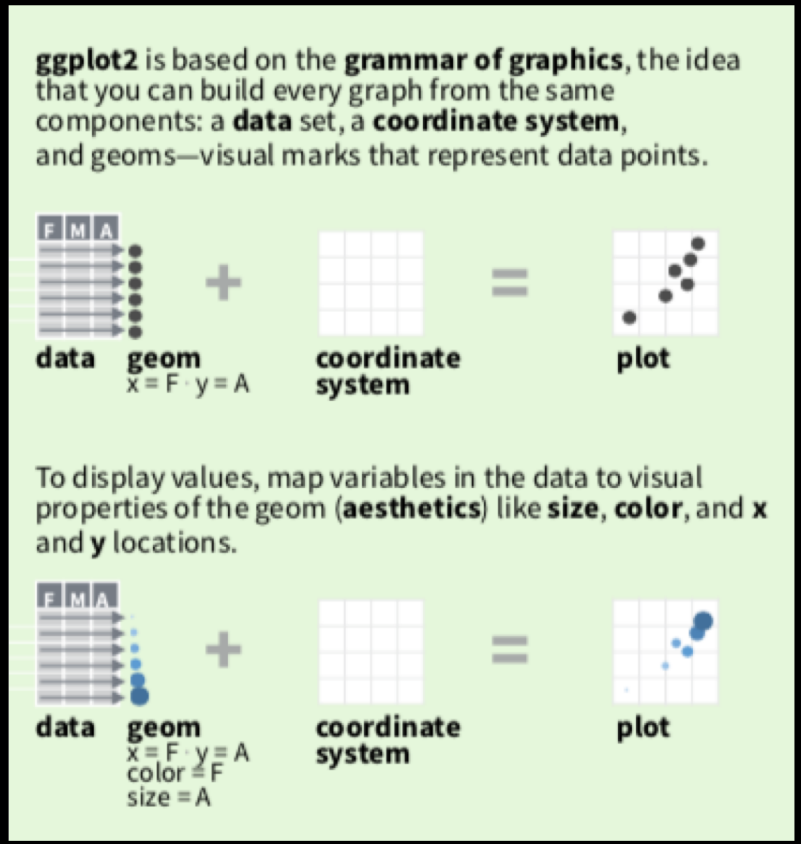
Examples

```
library(dplyr)
library(ggplot2)

data(nasa)
nasa %>% as.data.frame() -> df

ggplot(df %>% group_by(year) %>% summarise(pressure = mean(pressure))) +
  geom_point(aes(year, pressure))
ggplot(df %>% group_by(year) %>%
         summarise(pressure = mean(pressure)), aes(year, pressure)) +
  geom_line() + geom_point()
ggplot(df) + geom_histogram(aes(x=pressure))
ggplot(df) + geom_boxplot(aes(x=as.factor(year), y = temperature))
```

ggplot2



- To use data frame variables in the plot, wrap variables with aes(). Exmaple aes(x = height, y = weight)
- For any other static variable, put it outside aes. Example: ggplot(df) + geom_point(aes(x=height, y=weight), color =‘red’)
- If df, aes() are put in ggplot() then they are shared for all later on functions
- If df, aes() are put in specific plot (like geom_point()) then it only applies to that specific function

A few important rules

- Almost no for loop (if there is for loop, minimize it to be less than 100 total runs)
- No magic number
- Code should be run in seconds (if not, remove for loop; analyze where the code is slow; downsample the data)
- Validate number of rows after every dplyr operation
- Use left_join only
- Save data in the long format as much as possible

SQL

SQL is about selecting data from database tables

```
select *
from table_name;
```

One chart to understand SQL from dplyr

dplyr	SQL
select(c1, c2)	select c1, c2
filter(c1 == 3, c2 > 4)	where c1 = 3 and c2 > 4
arrange(c1)	order by c1
left_join(df2, by = c('c1' = 'c2', 'c3' = 'c4', ...))	left join df2 on df.c1 = df2.c2 and df.c3 = df2.c4
group_by(c1, c2, c3) + summarise(c4 = …)	select c1, c2, c3, ... as c4 group by c1, c2, c3
group_by(c1, c2, c3) + mutate(c4=...)	select c4=… over (partition by c1, c2, c3)
n()	count(*)

- The main difference is syntax
- SQL has very specific orders
- Many group_by summarise lef_join will result in a very long and nested code in SQL

Examples

dplyr

```
df %>% group_by(year) %>%
  summarise(temperature = mean(temperature))
```

```
df %>% group_by(year) %>%
  mutate(temperature = mean(temperature))
```

```
df %>% filter(year >= 1997, month >= 7) %>%
  group_by(year, month) %>%
  summarise(temperature = mean(temperature))
```

SQL

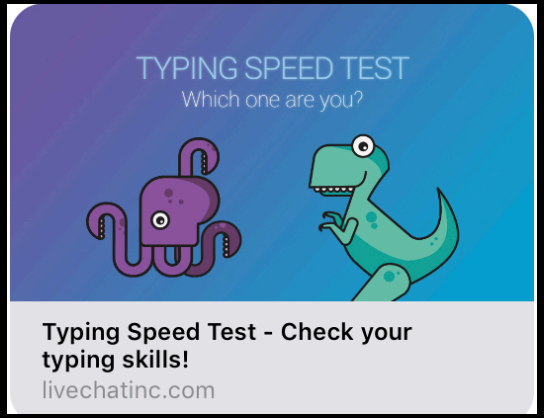
```
select year, mean(temperature) as temperature
from df
group by year
```

```
select year,
    mean(temperature) over (partition by year) as temperature
from df
```

```
select year, month, mean(temperature) as temperature
from df
where year >= 1997 and month >= 7
group by year, month
```

Coding speed

Are you an Octopus?



Important shortcuts to improve your coding speed

- Move cursor by one word: option + left/right arrow
- Move cursor to the beginning or end of a line: Command + left/right arrow
- Select one word: shift + option + left/right arrow
- Select one line: shift + command + left/right arrow
- Select multiple lines: shift + command + up/down arrow

Learn coding shortcuts

- TAB
- Parenthesis
- Quote

Browser and switching between browser and RStudio

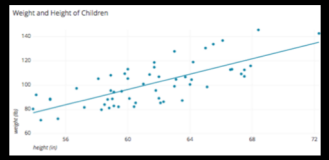
- Switch: command + tab
- Next tab: command + shift +]
- Previous tab: command + shift + [

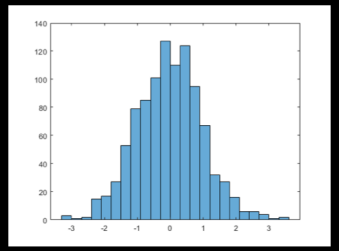
Demo

Checklist

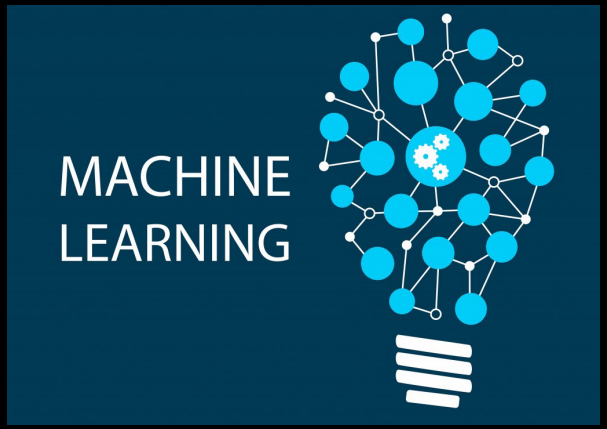
- Assign variables (logical, numeric, character, vector)
- Operations (+ - * /, ^)
- If () {} else {}
- Loops (while loop, for loop)
- Create a function
- Data frame – create a data frame, extract one to many columns, extract a column as a vector, extract first few rows and few rows by group
- Data frame – group by & summarise, group by & mutate
- Data frame – left join
- Data frame – create a new column from a vector, rename a column, create a column as a constant, create a column using if else, create a column using case_when
- Data frame – arrange
- Data frame – read.csv, write.csv
- Reshape (check the help)
- ggplot – aes
- ggplot – geom_point, geom_histogram, geom_boxplot, geom_line
- Ggplot – scale_x_continuous, scale_x_discrete, scale_y_continuous, scale_y_discrete

Next time - key modeling techniques



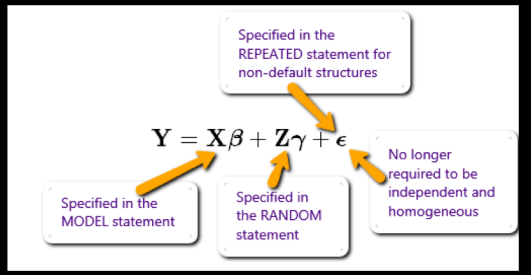


Basics



Machine learning (regression, tree based, neural network)

Operations Research



Variance decomposition

Planning for this course