

大數據分析方法

Big Data Analytical Methods

R Programming

曾意儒 Yi-Ju Tseng

長庚大學資管系

2016/3/14

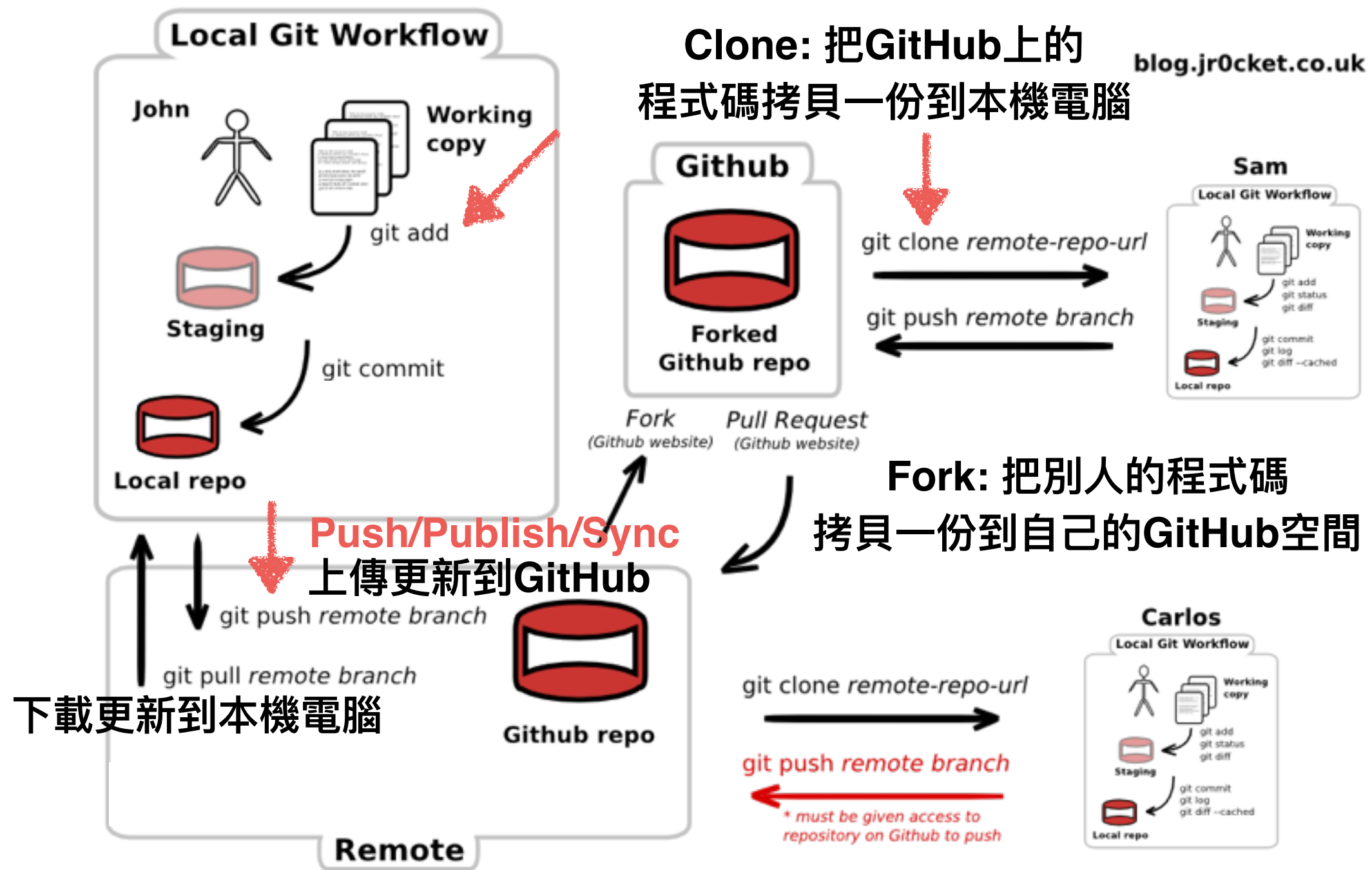
Outline

Date	Outline
2016-02-22	Data analysis introduction
2016-02-29	Holiday
2016-03-07	R programming/ GitHub overview
2016-03-14	R-Data types
2016-03-21	R-Control structures-1
2016-03-28	R-Control structures-2
2016-04-04	Holiday
2016-04-11	R-Functions/ Debugging tools
2016-04-18	Midterm
2016-04-25	Open data, how to use API
2016-05-02	Reading data-1 (HTML, text)
2016-05-09	Reading data-2 (JSON, XML)
2016-05-16	Data manipulation
2016-05-23	Exploratory data analysis & Analytic graphics-1
2016-05-30	Exploratory data analysis & Analytic graphics-2
2016-06-06	From data analysis to big data analysis
2016-06-13	Final Projects-1
2016-06-20	Final Projects-2

Final Project

- 2-4人一組 —> $44/3=15$ 組
- 兩次上課，300分鐘 —> 一組可用20分鐘
- 報告15分鐘，問問題5分鐘 —> 講重點，嚴格計時
- 題目與分析資料期中考後公佈
- 整組交一份書面報告
 - 組員與工作分配（會影響成績）、資料分析報告、資料討論/遇到的困難






Git & GitHub



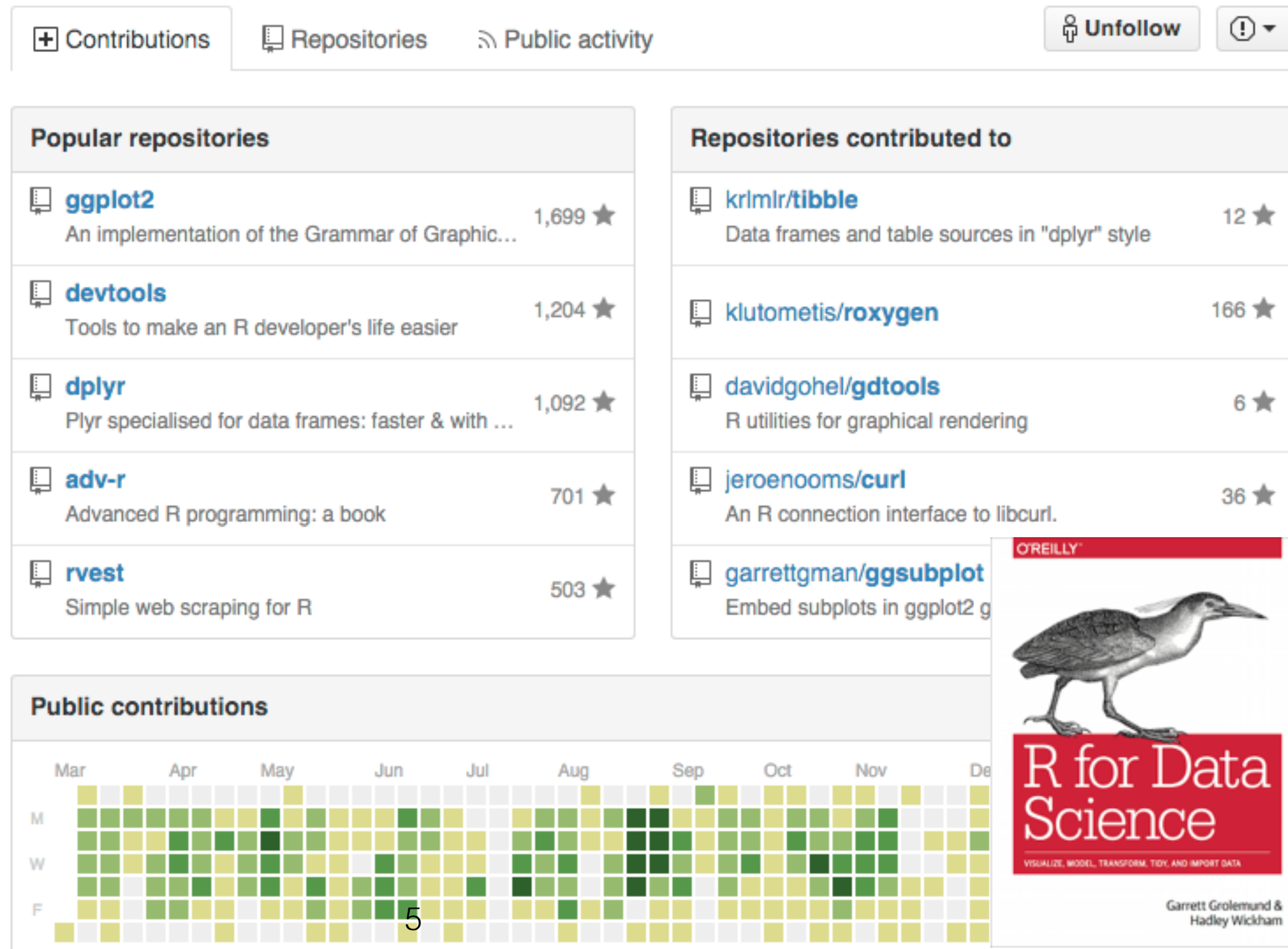
An Example of GitHub Profile



Hadley Wickham
hadley

 RStudio
 Houston, TX
 h.wickham@gmail.com
 <http://had.co.nz>
 Joined on Apr 1, 2008

5.6k Followers **111** Starred **7** Following



An Example of A README.md in A Repo

ggplot2

build passing CRAN 2.1.0

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and avoid bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

Find out more at <http://ggplot2.org>, and check out the nearly 500 examples of ggplot in use. If you're interested, you can also sign up to the [ggplot2 mailing list](#) at.

Installation

Get the released version from CRAN:

```
install.packages("ggplot2")
```

Or the development version from github:

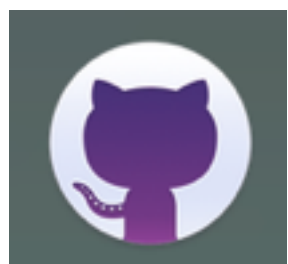
```
# install.packages("devtools")  
devtools::install_github("hadley/ggplot2")
```

複習

- **Character**: “aaa”, “2” / **Numeric**: 1,2,3
Logical: TRUE, FALSE
- 說明文件: ?functionName (函數名稱)
- +, -, *, /, ^ (次方) , %% (餘數) , round(6.555,digits=2) 四捨五入, sqrt(25) 平方根
- c() 序列, seq(1,9,2) 1到9的序列 , 間隔2, 1:20 (1到20的序列)
- mean(), max(), min(), sum(), summary(), sort(), rank()
- matrix(1:6, nrow = 2, ncol = 3)

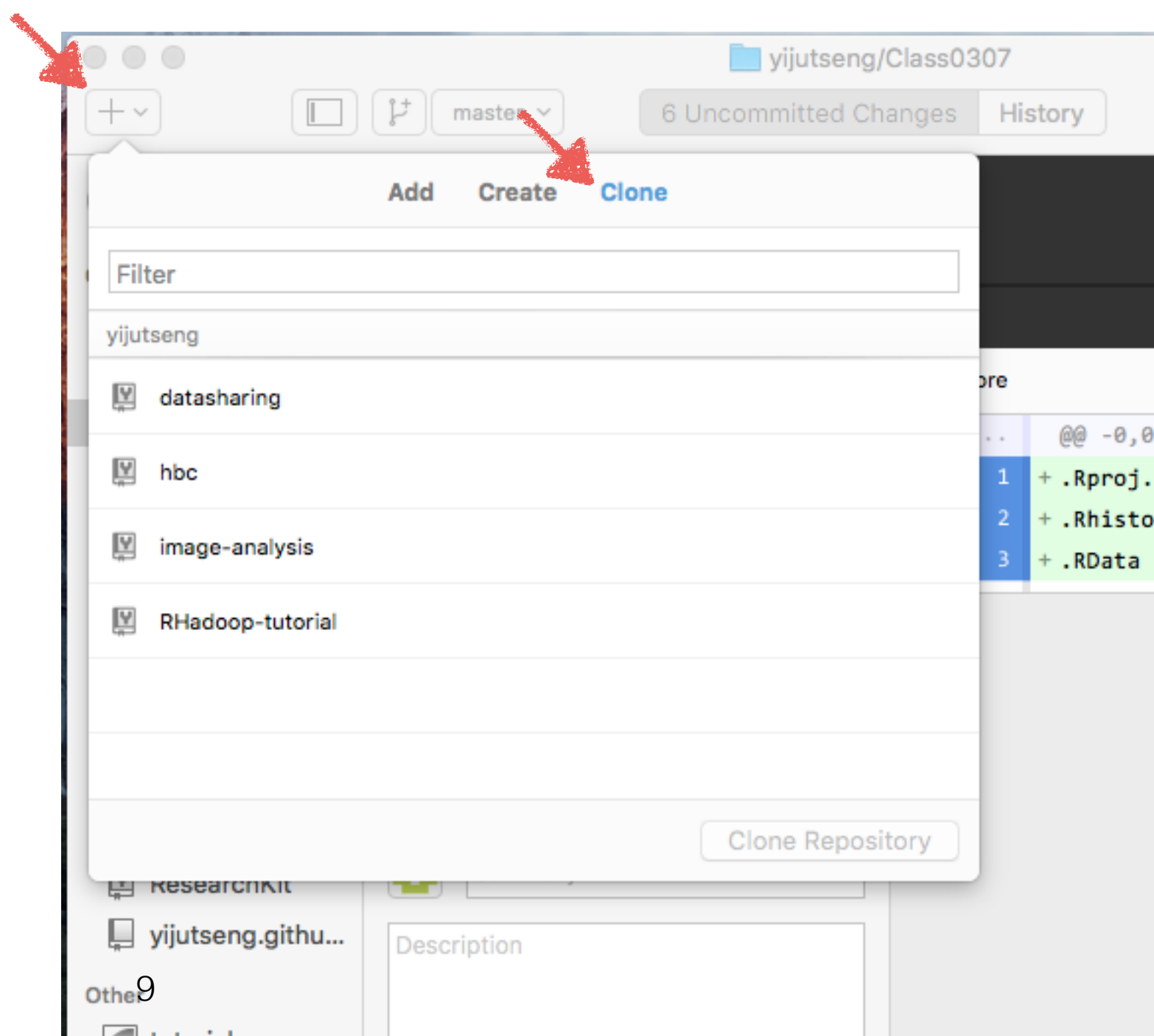
每次開始前.....

- 搜尋cmd —> 打開 **cmd Git**
- `git config --global user.name "Your Name"`
- `git config --global user.email "your_email@example.com"`



每次開始前.....

- 打開GitHub桌面版
- 打GitHub帳密
- **Clone**上次交的作業回本機端
- 用RStudio開啟上次作業做的專案



R Programming

列/行 結合

- `x <- 1:3`
- `y <- 10:12`
- `cbind(x, y)` : **column** , 把x和y各當成一列組合起來
- `rbind(x, y)` : **row** , 把x和y各當成一行組合起來

List

- 可以是不同類物件
- `x <- list(1, "a", TRUE, 1 + 4i)`
- x是包含numeric data, characters, boolean, complex各種不同物件的『序列』

Factors

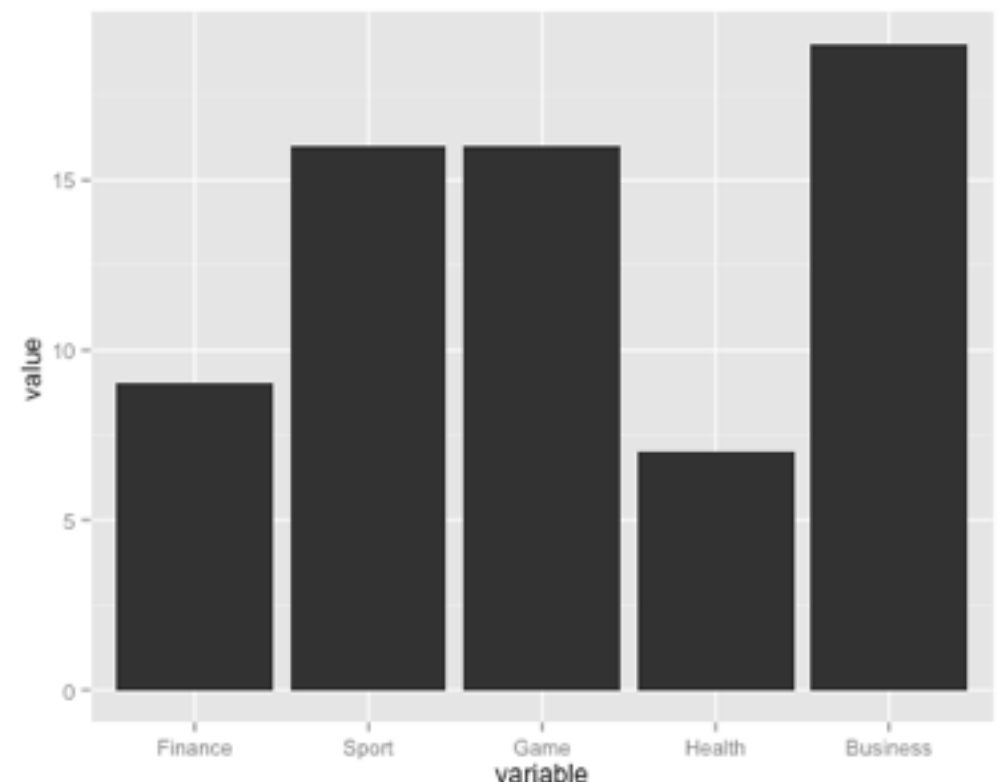
- Represent categorical data
- For modeling
- `x<-factor(c("yes", "yes", "no", "yes", "no"))`
 - yes->2, no->1
- `factor(c("yes", "yes", "no", "yes", "no"), levels = c("yes", "no"))`
 - yes->1, no->2

Data Frames

- like Excel table
- 同列要是相同類別，不同列可以是不同類別
- `x <- data.frame(foo = 1:4, bar = c(T, T, F, F))`
 - column names: foo, bar
- `nrow(x)`
- `ncol(x)`
- `names(x) <- c('fooNew', 'barNew')`

Let's Play with Real Data

- `install.packages("SportsAnalytics")` 安裝SportsAnalytics套件
- `library(SportsAnalytics)` 將SportsAnalytics讀入
- `NBA1415<-fetch_NBAPlayerStatistics("14-15")`
- `names(NBA1415)`
- `head(NBA1415)`
- `nrow(NBA1415)`
- `ncol(NBA1415)`



Dates and Times

- Dates —> Date class
 - Stored as number of **days** since 1970/01/01
- Times —> POSIXct or POSIXlt
 - Stored as number of **seconds** since 1970/01/01

Dates

- `x<-as.Date("1970-01-01")`
- `unclass(x)`
- `unclass(as.Date("1971-01-01"))`
- `weekdays()`, `months()`
- `seq(Sys.Date(), by='1 months', length.out=12)`

Times

`x<-Sys.time()` 系統時間

- POSIXct
 - Integer
- POSIXlt
 - List

`LisDate<-as.POSIXlt(x)` 轉為POSIXlt

`IntDate<-as.POSIXct(x)` 轉為POSIXct

`unclass(LisDate)`

`unclass(IntDate)`

Operations on Dates and Times

- 如果是 “2012/03/01”呢？ `as.Date("2012/03/01")`—>也可以！
- 如果是"2012 MAR 01"呢？
 - `as.Date("2012 MAR 01", "%Y %b %d")` 要告訴R你的日期格式
- `?strptime` 不可能背得起來的日期格式編碼表
- `+`, `-`, `>=`, `>` （可以運算，但要是同一類別）
 - `x <- as.Date("2012-03-01")`
 - `y <- as.Date("2012-02-28")`
 - `x-y`



Subset

- letters (R內建的資料集)
- letters [1]
- letters [1:10]
- letters[c(1,3,5)]
- letters [-1:-10]
- head(letters,5) 前五個物件
- tail(letters,5) 後五個物件
- islands (R內建的資料集)
- sort(islands)
- head(sort(islands))
- tail(sort(islands))

Data Frame Subsetting

- `dataFrame[row index, column index]` **篩選條件為index**
 - `iris[1,2]` (Row 1, Column 2)
- `dataFrame[, column name]`
 - `iris[, "Species"]` (取column name等於Species的那個column)
- `dataFrame$columnName`
 - `iris$Species` (取column name等於Species的那個column)
- `subset(dataFrame, logic)` **篩選條件為T/F**
 - `subset(iris, Species=="virginica")` 取Species的column值為virginica的所有rows
- `dataFrame[logic,]` **篩選條件為T/F**
 - `iris[iris$Species=="virginica",]` 取Species的column值為virginica的所有rows

Let's Start Playing with Real Data

- `install.packages("SportsAnalytics")`
- `library(SportsAnalytics)`
- `NBA1415<-fetch_NBAPlayerStatistics("14-15")`
- `San<-subset(NBA1415,Team=='SAN')`
- 誰打最多場？
 - `San[order(San$GamesPlayed,decreasing = T),"Name"]`
- 各隊哪個球員打最多場？ —>Google “select max group R”

str()

- 物件的詳細資料
- str(iris)
- str(NBA1415)

str()

492個Observations (行)，25個變數 (列)

列名

物件類別

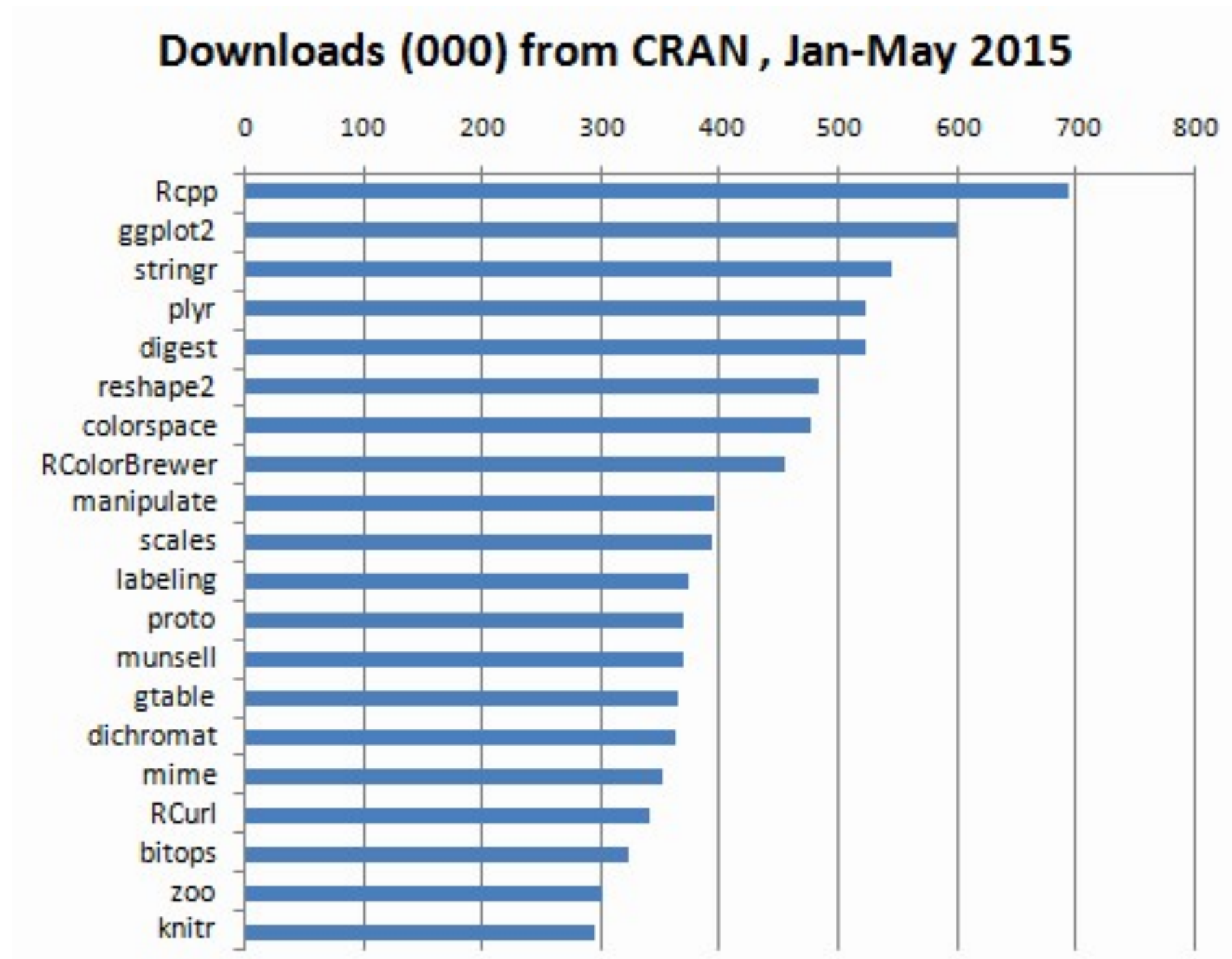
每行的資料

```
'data.frame': 492 obs. 25 variables:
 $ League      : Factor w/ 1 level "NBA": 1 1 1 1 1 1 1 1 1 1 ...
 $ Name        : chr  "Quincy Acy" "Jordan Adams" "Steven Adams" "Jeff Adrien" ...
 $ Team        : Factor w/ 30 levels "ATL","BOS","BRO",...: 20 15 21 18 25 19 23 20 25 12 ...
 $ Position     : Factor w/ 5 levels "C","PF","PG",...: 4 5 1 2 5 1 2 1 2 2 ...
 $ GamesPlayed  : int   68 30 70 17 78 68 41 61 71 63 ...
 $ TotalMinutesPlayed : int  1288 249 1776 215 2502 956 541 979 2514 1066 ...
 $ FieldGoalsMade   : int   152 35 217 19 375 181 40 144 659 142 ...
 $ FieldGoalsAttempted: int  331 86 399 44 884 328 78 301 1415 300 ...
 $ ThreesMade       : int    18 10 0 0 118 0 0 0 37 0 ...
 $ ThreesAttempted  : int    60 25 2 0 333 0 5 0 105 0 ...
 $ FreeThrowsMade    : int    76 14 103 22 167 81 13 50 306 33 ...
 $ FreeThrowsAttempted: int   97 23 205 38 198 99 27 64 362 48 ...
 $ OffensiveRebounds : int    79 9 199 23 27 104 78 101 177 124 ...
 $ TotalRebounds     : int   301 28 522 77 247 315 176 338 726 324 ...
 $ Assists           : int    68 16 65 15 129 47 28 75 124 72 ...
 $ Steals            : int    27 16 38 4 41 21 17 38 48 15 ...
 $ Turnovers         : int    60 14 99 9 116 69 17 59 122 40 ...
 $ Blocks            : int    22 7 85 9 7 51 16 65 68 42 ...
 $ PersonalFouls     : int   147 24 222 30 167 151 96 122 125 102 ...
```

Packages

- `available.packages()`
- `head(available.packages())`
- `install.packages("ggplot2")`
- In R studio
- `library(ggplot2)` —> Without “”

Top 20 R Packages



字串處理-1

- 分離
 - strsplit (想切的字,用什麼字切)
 - strsplit ("Hello World"," ") : Hello World, 用空白鍵切割字串
- 大小寫轉換
 - toupper(), tolower()
- 移除前後空白
 - str_trim(" Hello World ") , 需安裝stringr package

字串處理-2

- 連接字串
 - `paste(string1, string2, sep="")` `string1`和`string2`連接，中間不插任何字元
 - `paste0(string1, string2, string3, string4)` 功能同上
 - `paste(c(string1, string2), sep=",", collapse="")` 當須連結的字串是向量時，記得加`collapse=""`參數
- 切割字串
 - `substr(string, start=2, stop=4)` 只取第二個字元到第四個字元，生成新單字
- 字串取代
 - `gsub(欲取代的字, 新字, 整個字串)`
 - `gsub("o", "0", "Hello World")`

搜尋字串

- `grep(模式,字)`
 - 回傳index
 - `grep("Tim",NBA1415$Name)`
 - 回想到subset....
 - `NBA1415[grep("Tim",NBA1415$Name),]`
- `grep_(模式,字)`
 - 回傳T/F
 - 又回想到subset....
 - `subset(NBA1415,grepl('Tim',Name))`

Control Structures

運算式

- ==
- !=
- >, >=
- <, <=
- T & F
- T | F
- letters[13] (m)
- which(letters>"m") 回傳 index

if else

```
a<-3  
if(a>10){  
  b<-10  
}else if(a>5){  
  b<-5  
}else{  
  b<-a  
}
```

b會是...?

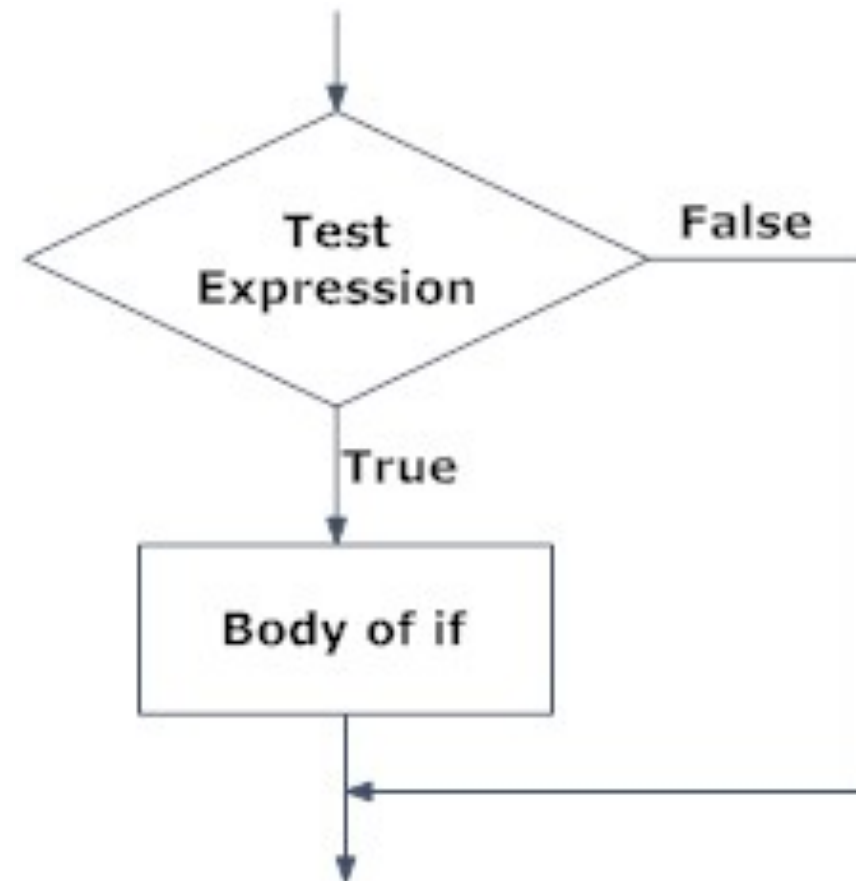


Fig: Operation of if statement

- ifelse(logic, TRUE要做的事, FALSE要做的事)
- ifelse(a>10, b<-10, b<-a)
 T F

ifelse 範例

- `ifelse(NBA1415$GamesPlayed>30,"Hardwork", "Lazy")`
- 若這個參數會重複使用
 - 替NBA1415新增一個參數Personality
 - `NBA1415$Personality<-
ifelse(NBA1415$GamesPlayed>30,"Hardwork","Lazy")`

for -1

想輸出1到10.....

```
for(index in 1:10){  
  print(index)  
}
```

```
index<-1  
repeat{  
  if(index>10){  
    break  
  }  
  print(index)  
  index<-index+1  
}
```

for -2

想跳過4這個不吉利的數字....

方法一

```
for(index in 1:10){  
  if(index!=4){  
    print(index)  
  }  
}
```

方法二

```
for(index in 1:10){  
  if(index==4){  
    next  
  }  
  print(index)  
}
```

for範例

- 印出2014-2015球季所有球員的名字

```
for(i in 1:nrow(NBA1415)){  
  print(NBA1415[i,"Name"])  
}
```

- 印出2014-2015球季，球員姓名內沒有"a"或"A"字母的球員的名字

```
for(i in 1:nrow(NBA1415)){  
  if(!grepl('a|A',NBA1415[i,"Name"])){  
    print(NBA1415[i,"Name"])  
  }  
}
```

for + if else範例

- 想找2014-2015球季，NBA各隊中，列出符合以下全部條件
 - 出場 (GamesPlayed) 超過70次
 - `NBA1415[i,"GamesPlayed"]>70`
 - 球季總得分 (TotalPoints) 超過1500分
 - `NBA1415[i,"TotalPoints"]>1500`
- 的球員姓名 (Name) /所屬球隊 (Team) /守備位置 (Position)
 - `NBA1415[i,c("Name","Team","Position")]`

都說了for不好用....

- 另解1
 - `subset(NBA1415, GamesPlayed > 70 & TotalPoints > 1500) [, c("Name", "Team", "Position")]`
- 另解2
 - `NBA1415[
NBA1415$GamesPlayed > 70 &
NBA1415$TotalPoints > 1500,
c("Name", "Team", "Position")
]`



apply()

- 有類似for迴圈的功能
- `apply(Data, MARGIN, FUN,....)`
 - Data：矩陣（Matrix），Data Frame
 - MARGIN：1=row, 2=column
 - FUN：函數
 -：函數要用的參數
- `apply(NBA1415,2,max)`：NBA1415的各列中，最大值是多少

Shortcut—>faster

- `rowSums = apply(x, 1, sum)`
- `rowMeans = apply(x, 1, mean)`
- `colSums = apply(x, 2, sum)`
- `colMeans = apply(x, 2, mean)`

sapply()

- Simplify->Vector
 - 如果回傳的list裡面，每個欄位長度都是1，回傳向量
 - 如果回傳的list裡面，每個欄位長度都是一樣，回傳矩陣
- sapply(Data, FUN...)
 - Data：矩陣（Matrix），Data Frame，**List**，**向量**
 - FUN：函數
- sapply(iris, mean)

lapply()

- List
- C code
- lapply(Data, FUN...)
 - Data : 矩陣 (Matrix) , Data Frame , List
 - FUN : 函數
- lapply(iris, mean)

tapply()

- `tapply(Data, INDEX, FUN,...)`
 - Data : 向量
 - INDEX : 分類因子
- `tapply(NBA1415$Name,NBA1415$Team,length)`
- `tapply(NBA1415$TotalPoints,NBA1415$Team,max)`

split()

- `tapply()`的部分功能：替向量做分群
- `split()`
- `split(1:30,gl(3, 10))` 1到30的向量，用第二個參數做分群
- `lapply(split(1:30,gl(3, 10)),mean)` 分群後算mean
- `tapply(1:30,gl(3, 10),mean)` 分群後算mean

split() -2

- 對data frame做分群
- split(Data, 分群依據)
 - NBA1415Team<-split(NBA1415,NBA1415\$Team)
- 搭配apply家族使用
 - lapply(NBA1415Team, colMeans) 回傳List
 - sapply(NBA1415Team, colMeans) 回傳Data frame

split() -3

- 使用多個分群依據
 - 隊伍、守備位置
- 用list將分群依據包起來
 - split(資料, list(分群依據1,分群依據2))
 - split(NBA1415[,c("TotalPoints","GamesPlayed")],
list(NBA1415\$Team,NBA1415\$Position))
 - 一樣可以用apply家族計算平均值

apply家族的使用時機

- apply：想要apply一個函數功能到陣列（Matrix）的行或列。不太建議在data frame使用
- lapply：想要apply一個函數功能到List中，回傳List
- sapply：同上，但想要回傳的東西是向量（Vector）
- tapply：想要對一個向量分群，分群後apply一個函數功能

類似apply家族的函數： aggregate()

- 像tapply(NBA1415\$TotalPoints,NBA1415\$Team,mean), split(), lapply()
- aggregate(資料, by=分組依據, FUN=函數功能)
 - aggregate(NBA1415\$TotalPoints,
by=list(NBA1415\$Team,NBA1415\$Position), FUN=mean)
- aggregate(formula, data=資料, FUN=函數功能)
 - aggregate(TotalPoints ~ Team+Position, data = NBA1415, mean)
- Formula: if you need all, put [.] into the formula
 - .~Team+Position

Missing Value

- `x<-c(1,2,3,4,5)`
- `mean(x)`
- `x<-c(x,NA)` 替向量x加上一個NA (Not a Value)
- `mean(x)` !?! 出現錯誤訊息！
- `mean(x, na.rm=T)` 設為忽略Missing value
- `sum(x)`
- `sum(x, na.rm=T)`

好慢.....

- Profiling
 - Measure, don't guess
- Using system.time(程式)
 - User time : CPU time by R session
 - System time : CPU time by OS
 - Elapsed time : "Real" time
- Rprof() : 每隔一段時間看一下程式碼做到哪裡，回傳
- summaryRprof()

List Subsetting

- List
 - [] 單括號取得list
 - [[]] 雙括號取得“值”
 - \$Name 取得“值”

Removing NA

- Subset
 - `x <- c(1, 2, NA, 4, NA, 5)`
 - `x[! is.na(x)]`
- `complete.cases()` 可使用在data frame，取出所有欄位都不是NA的row



R Markdown

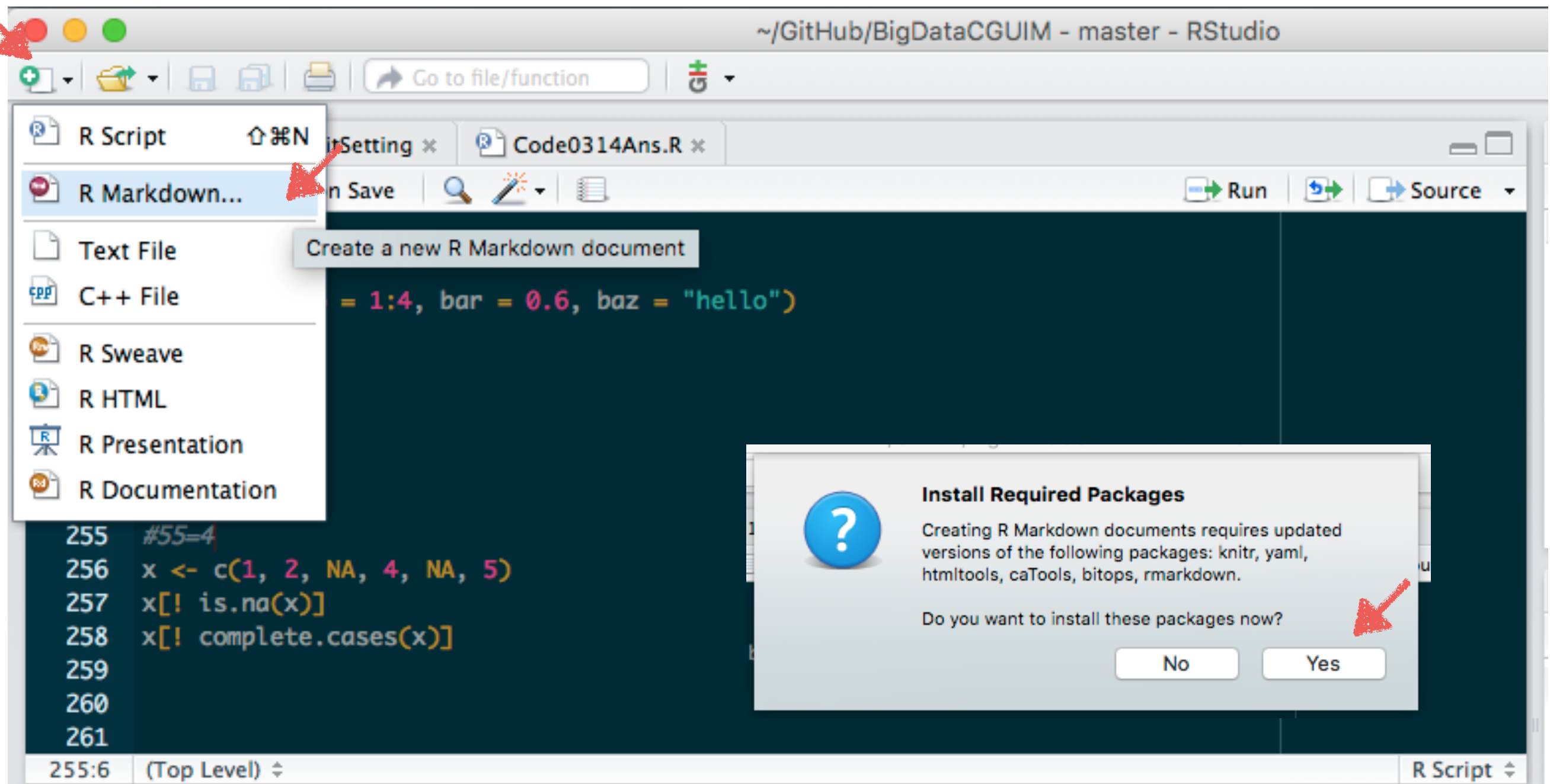
- <http://rmarkdown.rstudio.com/>
- R + Markdown
 - R Markdown—knitr—>Markdown
 - Markdown—markdown—>HTML
- 用來撰寫可以重複執行/動態的 R語言執行報告
- 報告格式：Slides, PDF, HTML. Word,...
- 又要交作業了....

R Markdown 元件

- 基於Markdown文件
- 所有寫在R Code Chunks的R程式碼都會執行，並將結果輸出
- 放上GitHub的文件，會輸出成.md檔

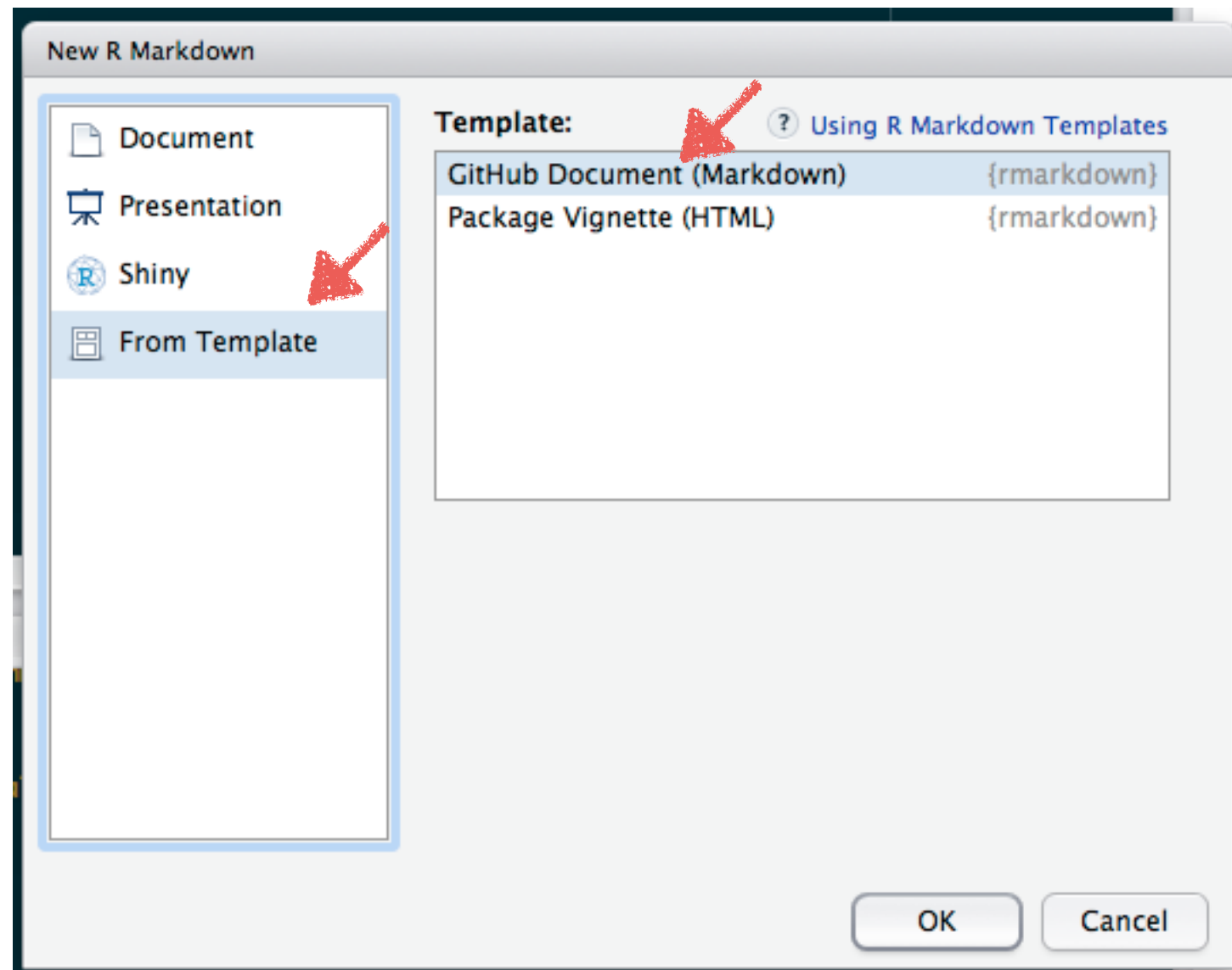
```
```${r}  
summary(cars)
```
```

Step 1 開啟新的R Markdown 檔案



Step 2 設定輸出格式

- 最後要放上GitHub，所以選From Template裡的GitHub Document



Step 3 設定文件基本資訊



```
1 ---  
2 title: "Untitled"  
3 output: github_document  
4 ---  
5  
6 ```{r setup, include=FALSE}  
7 knitr::opts_chunk$set(echo = TRUE)  
8 ```  
9  
10 ## GitHub Documents  
11  
12 This is an R Markdown format used for publishing markdown documents to GitHub. When you click the  
13 Knit button all R code chunks are run and a markdown file (.md) suitable for publishing to  
14 GitHub is generated.  
15  
16 ## Including Code  
17  
18 You can include R code in the document as follows:  
19  
20 ```{r cars}  
21 summary(cars)  
22 ```  
23  
24 ## Including Plots
```

12:212 (Top Level) R Markdown

其他都是內建的參考範例，先按“Knit”試試

自動生成的.md檔案在GitHub上的樣子

Untitled

GitHub Documents

This is an R Markdown format used for publishing markdown documents to GitHub. When you click the **Knit** button all R code chunks are run and a markdown file (.md) suitable for publishing to GitHub is generated.

Including Code

You can include R code in the document as follows:

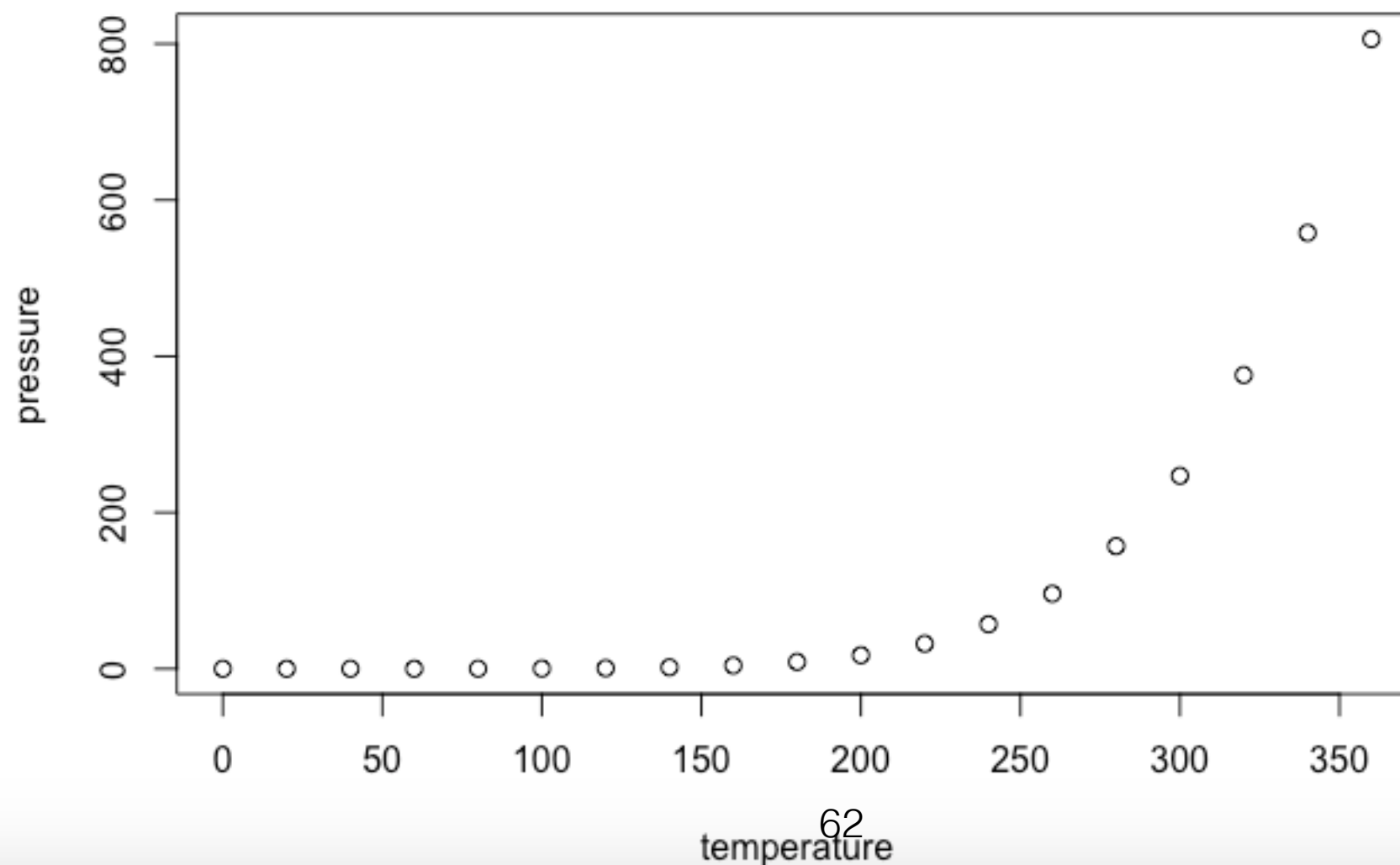
```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0    Min.   :  2.00
##  1st Qu.:12.0    1st Qu.: 26.00
##  Median :15.0    Median : 36.00
##  Mean   :15.4    Mean   : 42.98
##  3rd Qu.:19.0    3rd Qu.: 56.00
##  Max.   :25.0    Max.   :120.00
```

自動生成的.md檔案在GitHub上的樣子

Including Plots

You can also embed plots, for example:



R Code Chunk的參數

- eval：要不要執行並顯示在輸出的Markdown檔案內
- echo：要不要一起顯示程式碼以及執行結果
- warning, error, message：要不要顯示錯誤訊息
- cache：是否要暫存結果

Homework 3

- 分析NBA 2014到2015球季
 - 各隊最辛苦的球員（出戰分鐘數TotalMinutesPlayed最多）
 - 各隊得分王（TotalPoints最多）
 - 各隊最有效率的球員（總得分/出戰分鐘數 最高）
 - 各隊三分球出手最準的球員（ThreesMade/ThreesAttempted 最高）

Homework 3

- Title: NBA 2014-2015球季 各隊分析 (5 pt)
- 把資料讀進來的程式碼 (5 pt)
- 次標題1：各隊最辛苦的球員 (20 pt)
 - 標題 (2)，說明 (3)，程式碼 (5) 和結果 (10)
 - 結果要包括隊名 (2) 與球員姓名 (3) 與總出場數 (3)，按照總出場數排序(大到小) (2)
- 次標題2：各隊得分王 (20 pt)
 - 程式碼和結果，輸出要包括隊名與球員姓名與總分，按照總分排序
- 次標題3：各隊最有效率的球員 (25 pt)
 - 程式碼和結果，輸出要包括隊名與球員姓名與效率計算結果 (8)，按照效率計算結果排序
- 次標題4：各隊三分球出手最準的球員 (25 pt)
 - 程式碼和結果，輸出要包括隊名與球員姓名與三分球命中率 (8)，按照三分球命中率排序

作業範例 — R Markdown

```
10 ▾ ```{r echo=T}
11   #install.packages("SportsAnalytics")
12   library(SportsAnalytics)
13   NBA1415<-fetch_NBAPlayerStatistics("14-15")
14 ▲ ```
15
16 ▾ ## 各隊最辛苦的球員
17
18   計算依據為出戰分鐘數最多的球員
19
20 ▾ ```{r echo=T}
21   MaxPoint<-aggregate(TotalPoints~Team,NBA1415,max)
22   NBA1415MaxPoint<-merge(NBA1415,MaxPoint)
23   output<-NBA1415MaxPoint[order(NBA1415MaxPoint$TotalPoints,decreasing =
24     T),c("Team","Name","TotalPoints")]
25   library(knitr)
26   kable(output, digits=2)
26 ▲ ```
```

作業範例 — Markdown@GitHub

```
#install.packages("SportsAnalytics")
library(SportsAnalytics)
NBA1415<-fetch_NBAPlayerStatistics("14-15")
```

各隊得分王

計算依據為全季總得分最多的球員

```
MaxPoint<-aggregate(TotalPoints~Team,NBA1415,max)
#tapply(NBA1415$TotalPoints,NBA1415$Team,max)
NBA1415MaxPoint<-merge(NBA1415,MaxPoint)
output<-NBA1415MaxPoint[order(NBA1415MaxPoint$TotalPoints,decreasing = T),c("Team","Name","TotalPoints")]
library(knitr)
kable(output, digits=2)
```

| | Team | Name | TotalPoints |
|----|------|------------------|-------------|
| 11 | HOU | James Harden | 2217 |
| 10 | GSW | Stephen Curry | 1900 |
| 21 | OKL | Russel Westbrook | 1886 |
| 6 | CLE | Lebron James | 1740 |

作業繳交

Homework 3

大數據分析方法，作業三，請在3/20 11:59pm 前繳交

*必填

姓名 *

您的回答

Markdown檔案的Url *

您的回答 <https://github.com/yijutseng/BigDataCGUIM/blob/master/HW3.md>

Markdown檔案所屬Repo的SHA版本碼 *

您的回答 BigDataCGUIM的SHA碼

課程建議

您的回答

所有作業除了有特別寫，
遲交一天扣5分，最多扣40分（8天）



Debugging

- R的錯誤訊息
 - Message：有可能的錯誤通知，程式會繼續執行
 - Warning：有錯誤，但是不會影響太多，程式還是會繼續執行
 - `log(-1)`
 - Error：有錯，而且無法繼續執行程式
 - `mena(NA)`
 - Condition：可能會發生的情況

R Function

函數 Function

- 將程式碼依功能要求，寫成可重複使用的『函數』
 - `mean()`
 - `sum()`
- 好處：
 - 程式碼變短，閱讀和偵錯容易
 - 功能可以重複使用，節省開發時間

定義一個新的函數

- 函數名稱<-function(參數1, 參數2,)
{程式碼們}

- 四捨五入到小數點第二位

```
round2<-function(vector){  
  round(vector,digits = 2)  
}
```

```
round2(3.886)
```

Lazy Evaluation

- 沒用到Nothing參數，程式也不會出錯

```
round2Lazy<-function(vector,nothing){  
  round(vector,digits = 2)  
}  
round2(3.886)
```

Lazy Evaluation

- 使用f參數時，沒有輸入b，但R也不會檢查
- 在執行print(b)時才會跳出錯誤訊息

```
f <- function(a, b) {  
  print(a)  
  print(b)  
}  
f(45)
```

... 參數

- 意指引用其他函數的參數
- `apply(Data, MARGIN, FUN,...)`
- 例：平均再取四捨五入

```
roundmean<-function(vector, ...){  
  round(mean(vector, ...),digits=2)  
}  
roundmean(c(1.1,2,3,4,5))  
roundmean(c(1.1,2,3,4,5,NA))  
roundmean(c(1.1,2,3,4,5,NA),na.rm=T)
```

參數預設值

- 使用者如果沒有指定參數值，直接帶入預設

```
roundDe<-function(v=1.111:10.111){  
  round(v,digits = 2)  
}  
roundDe(1.66:6.66)  
roundDe()
```

retrun()

- 停止函數執行，並回傳值

```
round2<-function(v){  
  if(!is.numeric(v)){  
    print("輸入數字")  
    return()  
  }  
  round(v,digits = 2)  
}  
round2("a")
```

函數也可以當作參數來用

- `apply(Data, MARGIN, FUN,...)`
- `apply(iris,2,max)`

```
RoundNumber2<-function(v,XFun) {  
  round(XFun(v), digits = 2)  
}  
RoundNumber2(1.1:10.1,mean)
```

data.table 簡介

- <https://github.com/Rdatatable/data.table/wiki>
- Faster than data frame
- ~100GB
- Much easier
- 可由data frame 轉換而來

data.table 基本程式碼

- DT[where (i), select|update|do (j), by (b)]
- Take **DT**, subset rows using **(i)**, calculate **(j)** grouped by **(b)**
- Compare with data frame:
DataFrame[rows,columns]

data.table 範例—where (i)

- `DT[where (i), select|update|do (j), by (b)]`
- `library(data.table)`
- `NBA1415DT<-data.table(NBA1415)`
- `NBA1415DT[Team=='SAN']` 選取 Team欄位值等於'SAN'的所有rows
- 和Data frame比較
 - `NBA1415[NBA1415$Team=="SAN",]`
- 看起來好像差不多啊.....

data.table 範例—where (i)

- DT[where (i), select|update|do (j), by (b)]
- NBA1415DT[3:5] 選取第三行到第五行
- 和Data frame比較
 - NBA1415[3:5,]
- 還是差不多啊.....

data.table 範例— select| update|do (j)

- DT[where (i), select|update|do (j), by (b)]
- NBA1415DT[,list(Team,Name,TotalPoints)] 取隊伍名稱、姓名、總得分三個欄位
 - select
 - Data frame : NBA1415[,**c("Team","Name","TotalPoints")**]
 -就說了差不多啊！
- NBA1415DT[,list(mean(TotalPoints),sum(TotalPoints))] 算總得分平均與總和
 - do
 - Data frame : mean(NBA1415\$TotalPoints), sum(NBA1415\$TotalPoints)
 - 好像可以少幾行了....

data.table 範例— by (b)

- `DT[where (i), select|update|do (j), by (b)]`
- `NBA1415DT[,list(mean(TotalPoints),sum(TotalPoints)),by=Team]` 算各隊總得分平均與總和
- `do + by`
- Data frame :
`aggregate(TotalPoints ~ Team, data = NBA1415, mean)`
`aggregate(TotalPoints ~ Team, data = NBA1415, sum)`
- 好像可以少更多行了....

data.table 範例— by (b)

- `DT[where (i), select|update|do (j), by (b)]`
- `NBA1415DT[,list(mean(TotalPoints),sum(TotalPoints),sd(TotalPoints)),by=list(Team,Position)]` 算各隊、各守備位置總得分平均與總和
 - `do + by`
 - Data frame :
`aggregate(TotalPoints ~ Team+Position, data = NBA1415, mean)`
`aggregate(TotalPoints ~ Team+Position, data = NBA1415, sum)`
`aggregate(TotalPoints ~ Team+Position, data = NBA1415, sd)`
 - 好像可以少更多行了....

data.table 範例— do(j)+by (b)

- DT[where (i), select|update|do (j), by (b)]
- 也可以在do (j) 欄位做任何計算：像是三分球命中率
 - **ThreePerc=round(sum(ThreesMade)/sum(ThreesAttempted)**
- 當然也可以用where (i)欄位排序
 - [order(ThreePerc,decreasing = T)]

```
NBA1415DT[,list(ThreePerc=round(sum(ThreesMade)/  
sum(ThreesAttempted),digits = 2)),  
           by=list(Team,Position)]  
[order(ThreePerc,decreasing = T)]
```

data.table 範例— do(j)+by (b)

- DT[where (i), select|update|do (j), by (b)]
- 中鋒三分球命中率50% ! ? ! ? ! ? ! 有詐 !

```
NBA1415DT[,list(ThreePerc=round(sum(ThreesMade)/  
sum(ThreesAttempted),digits = 2),  
ThreeMadeTotal=sum(ThreesMade),ThreeAttTotal=sum(  
ThreesAttempted),  
by=list(Team,Position)]  
[order(ThreePerc,decreasing = T)]
```


data.table 常用代號

- DT[where (i), select|update|do (j), by (b)]
- .N : 總數
- .I : Row number

```
NBA1415DT[,list(TotalPointsAvg=mean(TotalPoints),  
                TotalPointsSum=sum(TotalPoints),  
                TotalPointsSd=sd(TotalPoints),  
                Max=max(TotalPoints),  
                Count=.N,.I),by=list(Team,Position)]
```

data.table 常用代號 2

- DT[where (i), select|update|do (j), by (b)]
- :=
- update (j)
- NBA1415DT[, **ThreePerc:=round(sum(ThreesMade)/sum(ThreesAttempted),digits = 2)**]
 - 新增一個三分球命中率**ThreePerc**的欄位

data.table 常用代號 4

- DT[where (i), select|update|do (j), by (b)]
- .SD
- select (j) all columns
- .SDcols

