Introduction to R

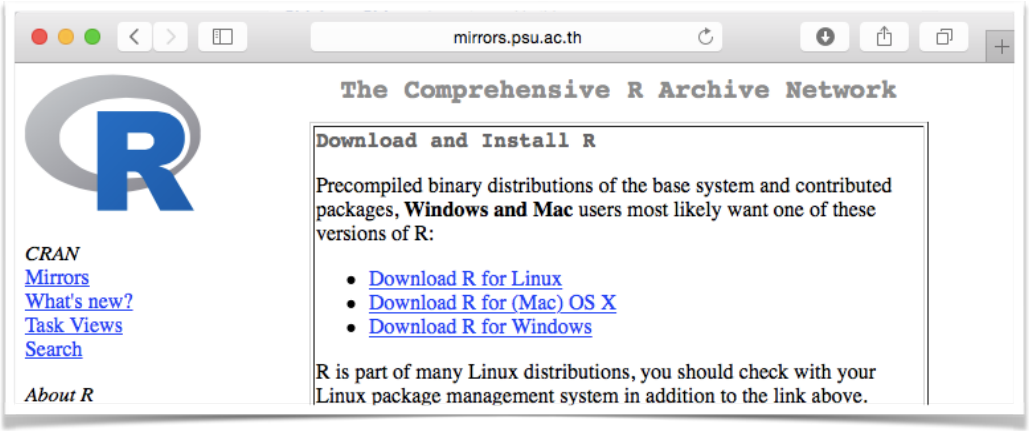
datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Introduction to R

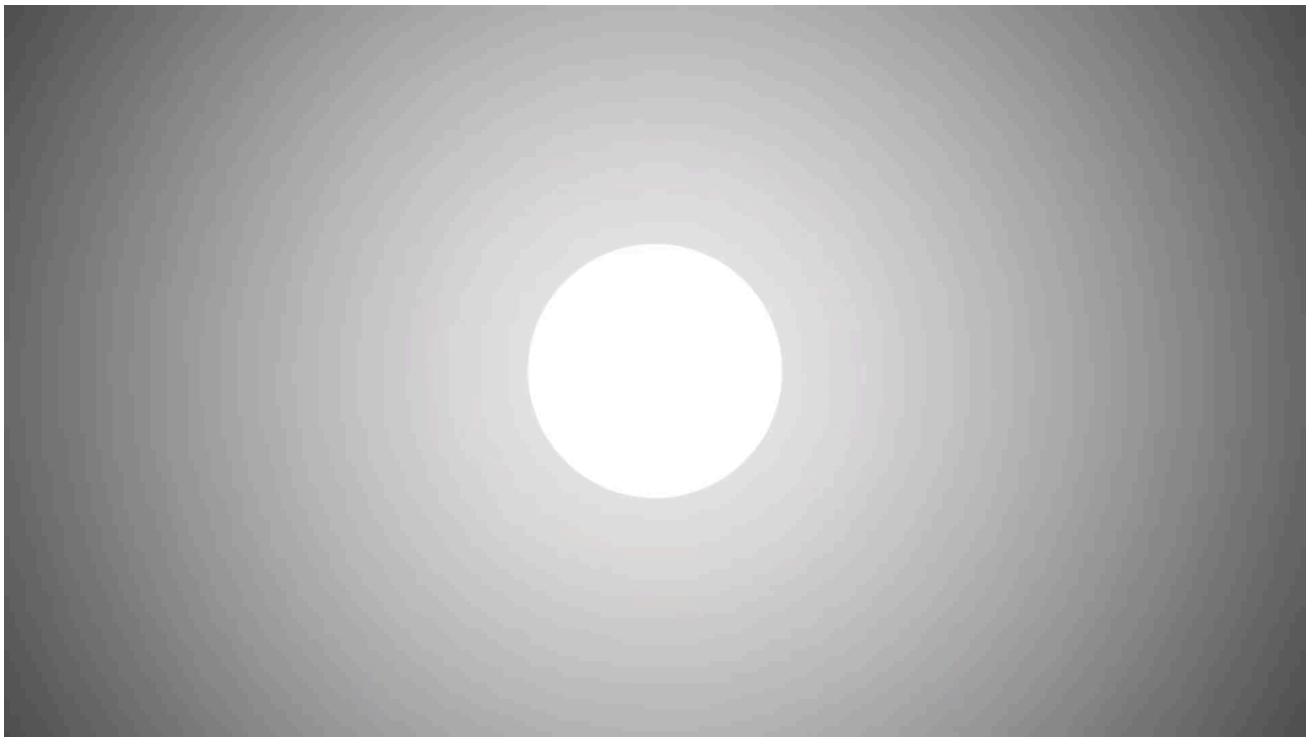
datacube

- R คือ ซอฟต์แวร์ open source ที่ใช้ทางด้านการคำนวณเชิงสถิติ
- ดาวน์โหลดได้จาก https://www.r-project.org
  - ติดตั้งได้ทั้ง Windows, OS X และ Linux



Introduction to R

datacube

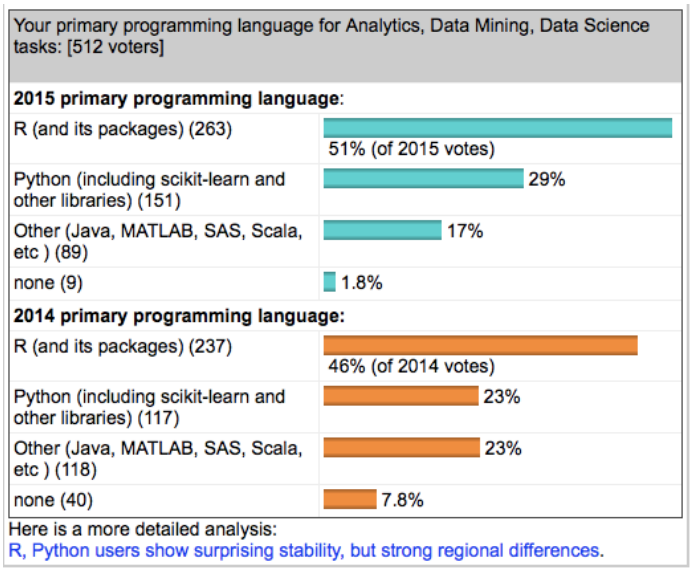


source: https://www.youtube.com/watch?v=TR2bHSJ_eck

Introduction to R

datacube

- จากผลการสำรวจของเว็บไซต์ kdnuggets.com พบว่าเป็นภาษาที่มีผู้ใช้ในการวิเคราะห์ข้อมูลมากที่สุดในปี 2014 และ 2015



Your primary programming language for Analytics, Data Mining, Data Science tasks: [512 voters]	
2015 primary programming language:	
R (and its packages) (263)	51% (of 2015 votes)
Python (including scikit-learn and other libraries) (151)	29%
Other (Java, MATLAB, SAS, Scala, etc) (89)	17%
none (9)	1.8%
2014 primary programming language:	
R (and its packages) (237)	46% (of 2014 votes)
Python (including scikit-learn and other libraries) (117)	23%
Other (Java, MATLAB, SAS, Scala, etc) (118)	23%
none (40)	7.8%

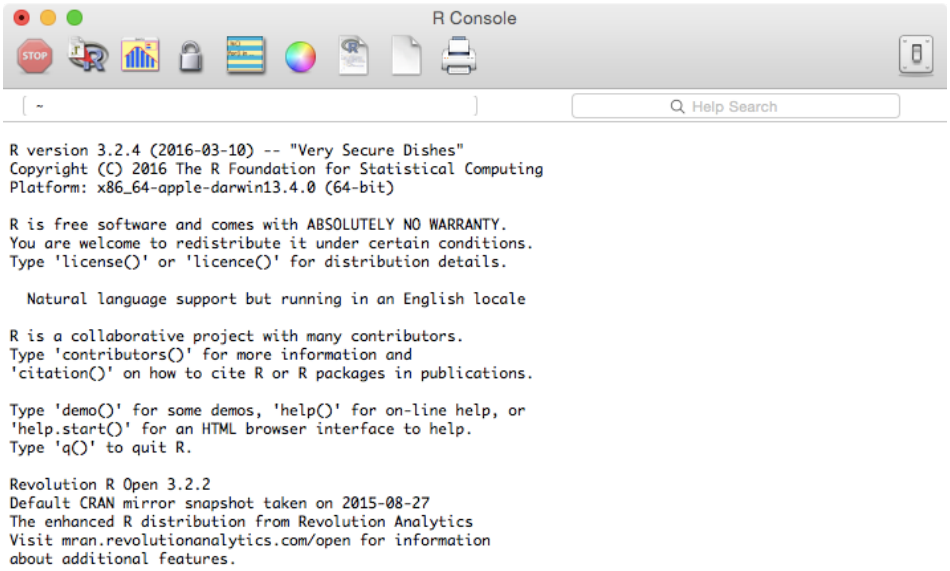
Here is a more detailed analysis:
R, Python users show surprising stability, but strong regional differences.

source: http://www.kdnuggets.com/polls/2015/r-vs-python.html

Introduction to R

datacube

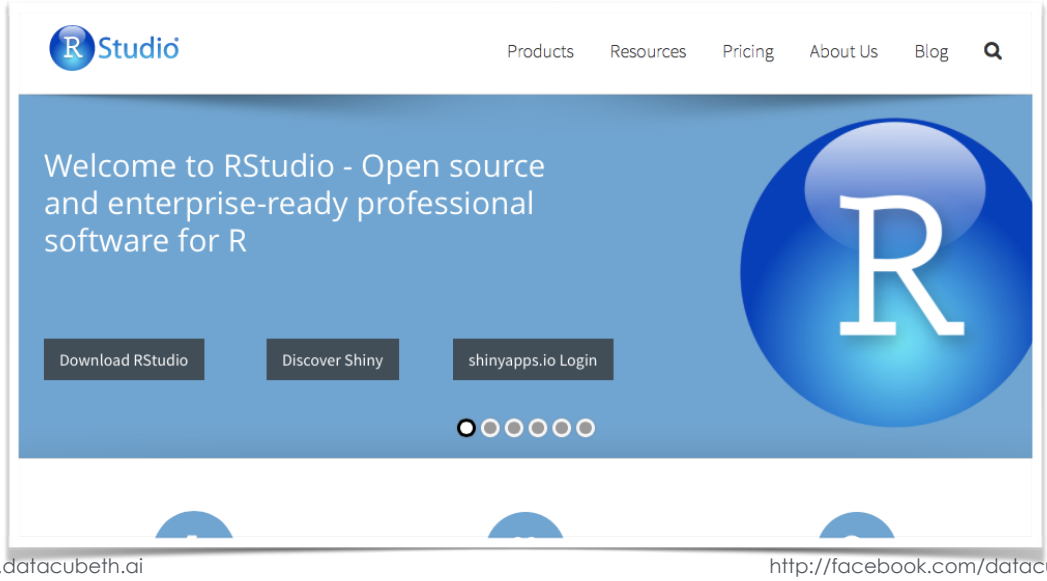
- R เป็นภาษาโปรแกรมอย่างหนึ่งที่ใช้งานง่ายกว่า C/C++ หรือ Java
- พิมพ์คำสั่งผ่านทาง R Console



Introduction to R

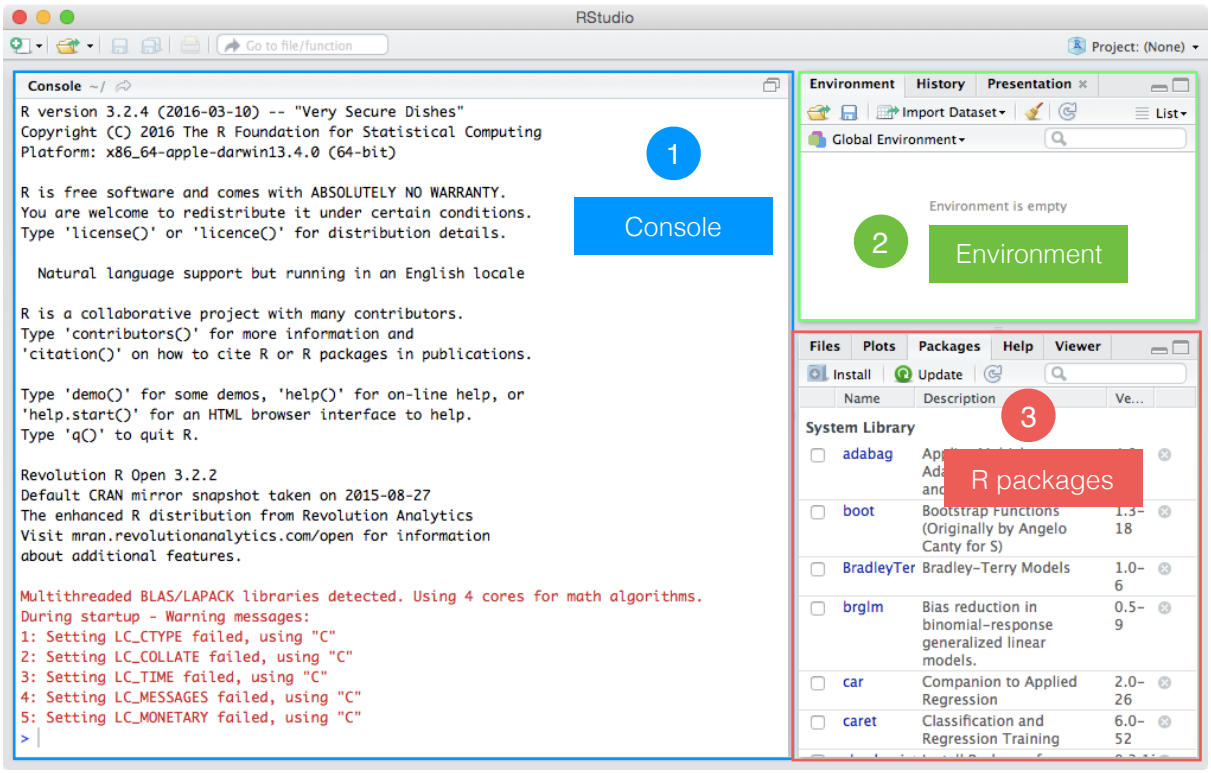
datacube

- เพื่อให้การใช้งาน R ง่ายขึ้นมีคนพัฒนาซอฟต์แวร์ RStudio เพื่อเป็น editor ซึ่งสามารถสั่งให้ทำงานได้เลย
- ดาวน์โหลด RStudio ได้จาก https://www.rstudio.com



Introduction to RStudio

datacube



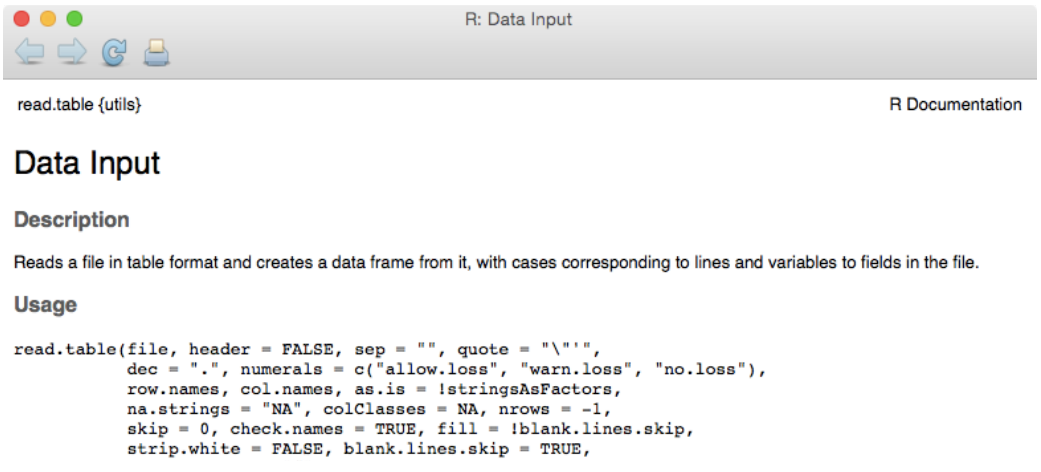
Introduction to RStudio

datacube

- คำสั่ง ? ใช้เพื่อเรียกดู help ของ R

เรียกดู help ของ R

```
> ?read.csv
```



Introduction to R

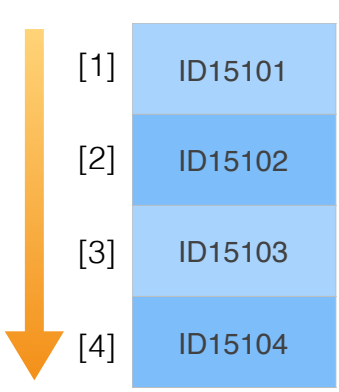
datacube

- แนะนำ R และ RStudio
- **โครงสร้างข้อมูลพื้นฐานใน R**
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

R Data Structures: Vectors

datacube

- เวกตอร์ (Vector)
  - เก็บข้อมูลเรียงตามลำดับ และข้อมูลจะต้องเป็นประเภทเดียวกันทั้งหมด เช่น ตัวเลข หรือ ข้อความ



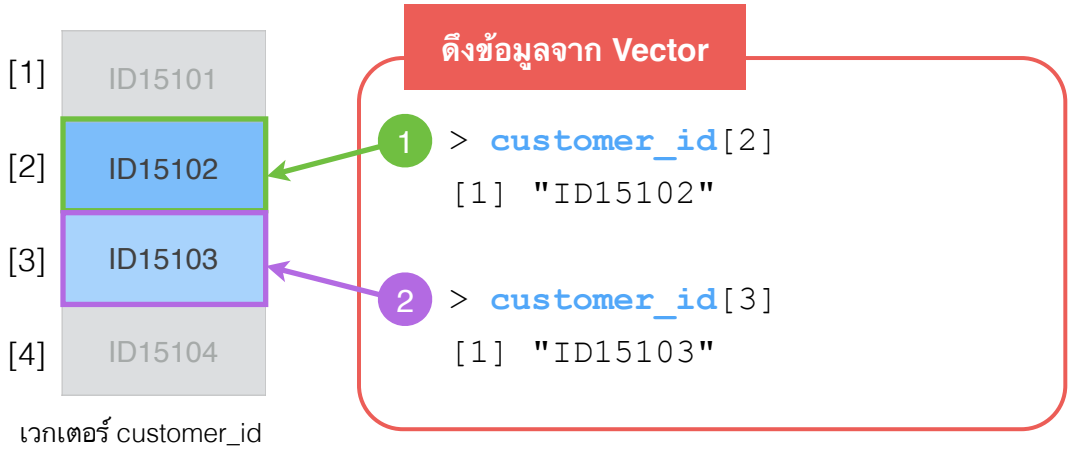
เวกเตอร์ customer_id

การสร้าง Vector

```
> customer_id <- c("ID15101",
                   "ID15102",
                   "ID15103",
                   "ID15104")
```

R Data Structures: Vectors

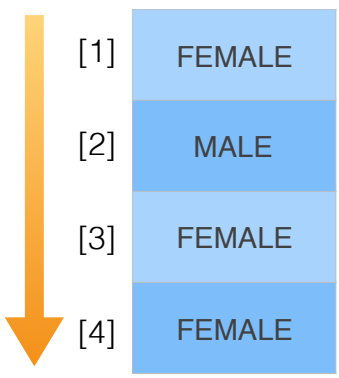
- เวกตอร์ (Vector)
  - เก็บข้อมูลเรียงตามลำดับ และข้อมูลจะต้องเป็นประเภทเดียวกันทั้งหมด เช่น ตัวเลข หรือ ข้อความ



เวกเตอร์ customer_id

R Data Structures: Factors

- แฟคเตอร์ (Factor)
  - มีลักษณะเหมือนกับเวกเตอร์ แต่เก็บข้อมูลประเภทที่เป็นนอมินอล (nominal) หรือ categorical เท่านั้น



แฟคเตอร์ gender

การสร้าง Factor

```
> gender <- factor(c("FEMALE",
                     "MALE",
                     "FEMALE",
                     "FEMALE"))
```

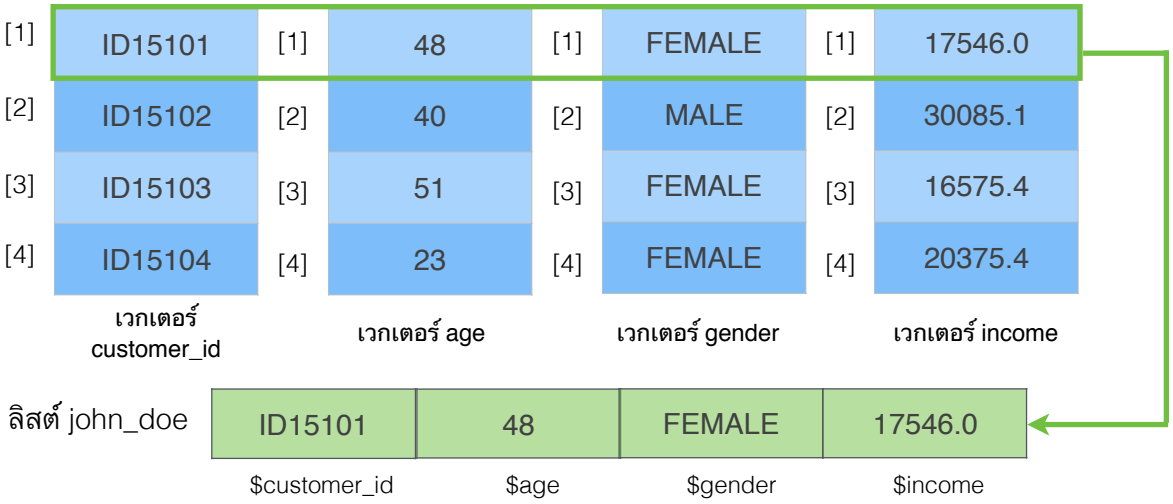
การเรียกใช้งาน Factor

```
> gender
  [1] FEMALE MALE FEMALE FEMALE
  Levels: FEMALE MALE
```

R Data Structures: List

datacube

- ลิสต์ (List)
  - เก็บข้อมูลเรียงตามลำดับ และเก็บข้อมูลต่างประเภท (type) กันได้ (ซึ่งไม่เหมือน Vector ที่เก็บประเภทเดียวกันเท่านั้น)



R Data Structures: List

datacube

การสร้าง List

```
> age <- c(48,40,51,23);
> gender <- c("FEMALE","MALE","FEMALE","FEMALE");
> income <- c(17546.0,30085.1,16575.4,20375.4);
> john_doe <- list(customer_id = customer_id[1],
                   age = age[1],
                   gender = gender[1],
                   income = income[1])
```

การดึงข้อมูลจาก List

```
> john_doe$customer_id
[1] "ID15101"
```

R Data Structures: List

datacube

การดึงข้อมูลจาก List

```
> john_doe$customer_id
[1] "ID15101"

> john_doe$age
[1] 48

> john_doe$gender
[1] "FEMALE"

> john_doe$income
[1] 17546
```

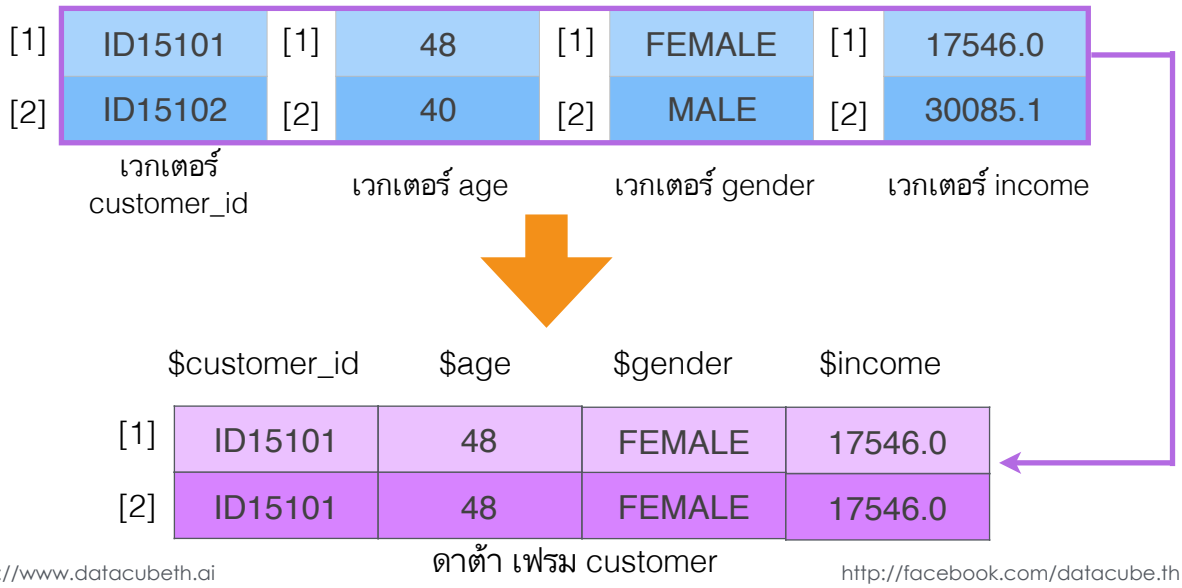
การดึงข้อมูลจาก List

```
> john_doe
$customer_id
[1] "ID15101"

$age
[1] 48

$gender
[1] "FEMALE"

$income
[1] 17546
```


R Data Structures: Data Frame

datacube

- ดาต้า เฟรม (Data frame)
  - เก็บข้อมูลในลักษณะของตาราง ประกอบด้วยแถว (row) และ คอลัมน์ (column) ซึ่งเก็บค่าประเภทต่างกันได้



R Data Structures: Data Frame

datacube

การสร้าง Data frame

```
> customer <- data.frame(customer_id, age,
                         gender, income)
```

การดึงข้อมูลจาก Data frame

```
> customer
      customer_id age gender income
1      ID15101       48 FEMALE 17546.0
2      ID15102       40   MALE 30085.1
3      ID15103       51 FEMALE 16575.4
4      ID15104       23 FEMALE 20375.4
```


R Data Structures: Data Frame

datacube

การดึงข้อมูลจาก Data frame

```
> customer[1,1] # [i,j], i = row, j = column
[1] ID15101
Levels: ID15101 ID15102 ID15103 ID15104

> customer[1,] # show only 1st row
  customer_id age gender income
1     ID15101  48 FEMALE  17546

> customer[,1] # show only 1st column
[1] ID15101 ID15102 ID15103 ID15104
Levels: ID15101 ID15102 ID15103 ID15104
```

R Data Structures: Data Frame

datacube

การดึงข้อมูลจาก Data frame

```
> customer[1,1:2]
  customer_id age
1     ID15101  48

> customer[1,c(1:2,4)]
  customer_id age income
1     ID15101  48  17546

> customer[1,-3]
  customer_id age income
1     ID15101  48  17546
```


R Data Structures: Matrix

datacube

- เมทริกซ์ (Matrix)
  - เก็บข้อมูลในรูปแบบแถวและคอลัมน์เหมือนกับ data frame แต่ต้องเป็นข้อมูลประเภทเดียวกันส่วนใหญ่จะเป็นข้อมูลประเภทตัวเลข

	[1]	[2]	[3]	[4]
[1]	100	500	900	1300
[2]	200	600	1000	1400
[3]	300	700	1100	1500
[4]	400	800	1200	1600

R Data Structures: Matrix

datacube

การสร้าง Matrix

```
> data <- matrix(c(100,200,300,400,500,600,
                   700,800,900,1000,1100,1200,
                   1300,1400,1500,1600),
                   nrow=4, ncol=4)
```

ดึงข้อมูลจาก Matrix

```
> data
     [,1] [,2] [,3] [,4]
[1,]  100  500  900 1300
[2,]  200  600 1000 1400
[3,]  300  700 1100 1500
[4,]  400  800 1200 1600
```


Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- **การอ่านไฟล์ข้อมูล**
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Importing Data

datacube

- คำสั่งสำหรับการอ่านข้อมูลประเภท CSV

การอ่านไฟล์ CSV

```
> data <- read.csv("Rcustomers.csv",header=TRUE)
```

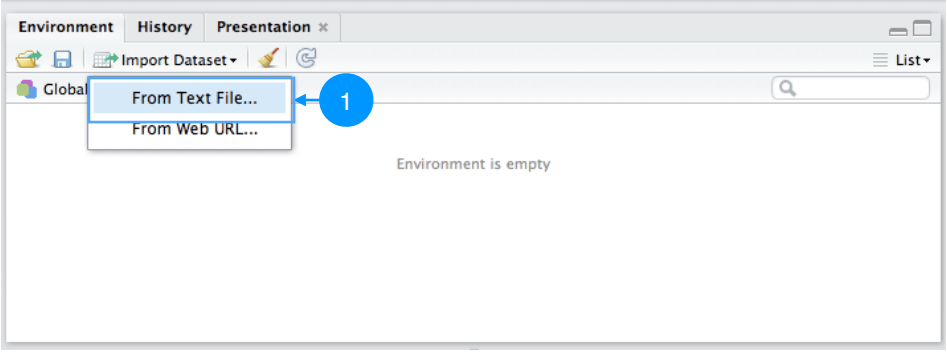
1 ชื่อไฟล์ประเภท CSV

2 กำหนดให้แถวแรกเป็นชื่อแอตทริบิวต์

Importing Data

datacube

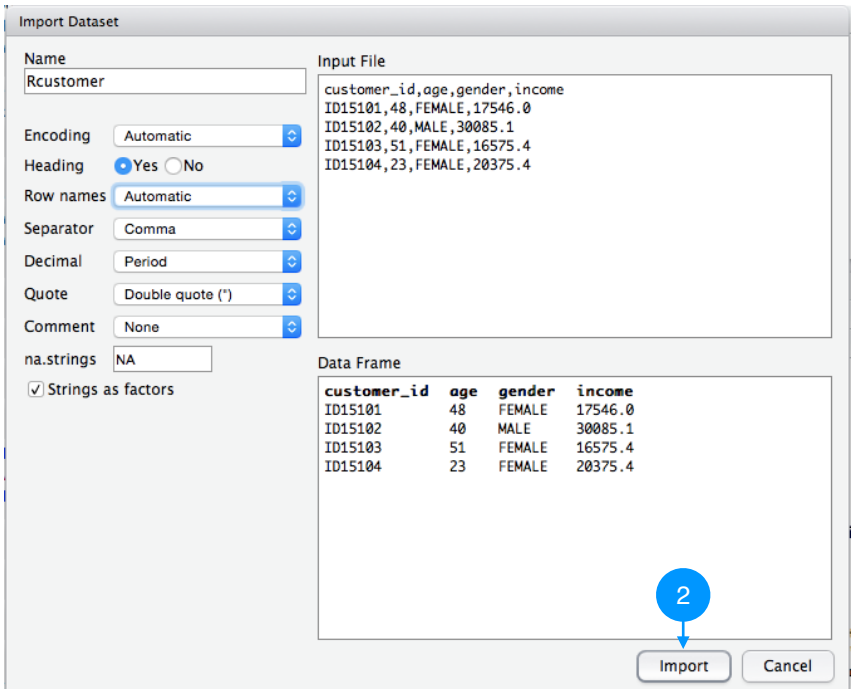
- การอ่านไฟล์ CSV ด้วย RStudio
- ในส่วนของ Environment เลือก Import Dataset
- เลือกเมนู From Text File



Importing Data

datacube

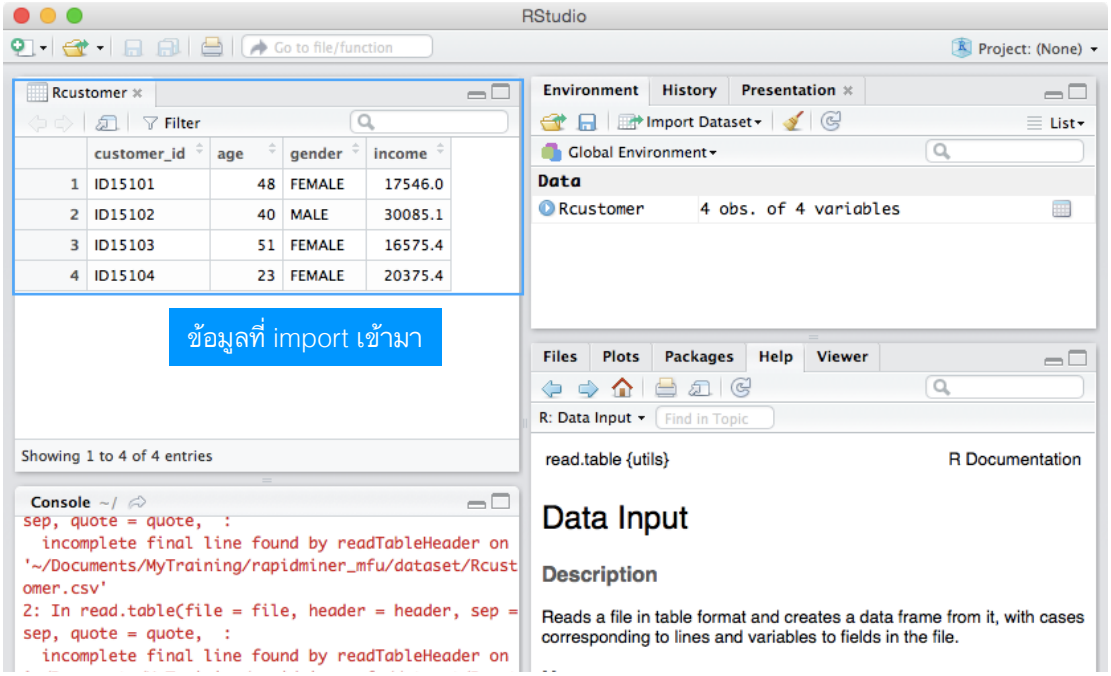
- เปลี่ยนชื่อ dataset ได้ในช่อง **Name**



Importing Data

datacube

- ข้อมูลที่ import เข้ามาจะแสดงในส่วนด้านซ้ายบน



Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- **การเขียนโปรแกรมภาษา R เบื้องต้น**
  - **การเลือกเงื่อนไข (IF)**
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Conditional execution

datacube

- คำสั่งในการตรวจสอบเงื่อนไข ใช้ IF

คำสั่ง IF

```
> if (condition) {
      # code executed when condition is TRUE
  } else {
      # code executed when condition is FALSE
  }
```

- ถ้าเงื่อนไขหลัง if เป็นจริง (true) จะทำตามคำสั่งที่อยู่หลัง if
  - อาจจะมีมากกว่า 1 คำสั่งได้แต่ต้องอยู่ในวงเล็บ {}
- ถ้าเงื่อนไขเป็นเท็จ (false) จะทำตามคำสั่งหลัง else

Conditional execution

datacube

- ตัวอย่างการใช้งาน

ตัวอย่างคำสั่ง IF

```
> accuracy.DT <- 70.5
> accuracy.KNN <- 93.0
> if (accuracy.DT > accuracy.KNN) {
     best.technique <- "DT"
  } else {
     best.technique <- "KNN"
  }
> best.technique
  # KNN
```


Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Repetitive execution

datacube

- คำสั่งในการวนรอบ (loop)

คำสั่ง FOR

```
> for (name in expression1) {
     # code executed
  }
```

  - คำสั่ง for จะวนรอบโดยที่
    - name คือ ชื่อตัวแปร
    - expression1 คือ เงื่อนไขในการวนรอบ เช่น 1:20

Repetitive execution

datacube

- ตัวอย่างคำสั่งในการวนรอบ (loop)

ตัวอย่างคำสั่ง FOR

```
> fold <- seq(1:10)
> for (i in 1:length(fold)) {
      cat("Fold no. ",i,"\n")
  }
# Fold no. 1
# Fold no. 2
# Fold no. 3
# Fold no. 4
# Fold no. 5
# Fold no. 6
# …
```

ตัวอย่างคำสั่ง FOR

```
> fold <- seq(1:10)
> for (i in fold) {
      cat("Fold no. ",i,"\n")
  }
# Fold no. 1
# Fold no. 2
# Fold no. 3
# Fold no. 4
# Fold no. 5
# Fold no. 6
# …
```

Repetitive execution

datacube

- ตัวอย่างคำสั่งในการวนรอบ (loop)

ตัวอย่างคำสั่ง FOR

```
> accuracy <- c(70.5, 93.0, 85.6)
> best.accuracy <- 0
> for (i in accuracy) {
     if (i > best.accuracy)
         best.accuracy <- i
  }
> best.accuracy
# 93.0
```


Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Function in R

datacube

- รูปแบบของคำสั่งการสร้างฟังก์ชัน (function)

คำสั่ง FUNCTION

```
> name <- function(arg1, arg2) {
    expression
  }
```

  - name คือ ชื่อฟังก์ชันที่สร้างขึ้นมา
  - function เป็น keyword สำหรับสร้างฟังก์ชัน
  - arg1, arg2 คือ argument ที่ส่งให้ฟังก์ชัน

Function in R

datacube

- ตัวอย่างการสร้างฟังก์ชัน rescale01

ตัวอย่างการสร้าง FUNCTION

```
> rescale01 <- function(x) {
    rng <- range(x, na.rm = TRUE)
    (x - rng[1]) / (rng[2] - rng[1])
  }
> data <- c(0,5,10)
> result <- rescale01(data)
> result
# 0.0 0.5 1.0
```

Function in R

datacube

- ตัวอย่างการสร้างฟังก์ชัน performance

ตัวอย่างการสร้าง FUNCTION

```
> performance <- function(TP,FP,TN,FN) {
    precision <- 100*TP/(TP+FP)
    recall <- 100*TP/(TP+FN)
    F.measure <- (2*precision*recall)/(precision+recall)
    list(precision=precision, recall=recall,
         F.measure=F.measure)
  }
> TP <- 100
> FP <- 55
> TN <- 35
> FN <- 40
```


Function in R

datacube

- ตัวอย่างการสร้างฟังก์ชัน performance (ต่อ)

ตัวอย่างการสร้าง FUNCTION

```
> result <- performance(TP,FP,TN,FN)
# $precision
# [1] 64.51613
# $recall
# [1] 71.42857
# $F.measure
# [1] 67.79661
> result$precision
# [1] 64.51613
> result$F.measure
# [1] 71.42857
```

Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- **การสร้างกราฟด้วย R เบื้องต้น**
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Visualization with R

datacube

- การสร้างกราฟแบบต่างๆ ด้วยภาษา R สามารถทำได้ง่ายโดยใช้ package ggplot2
- ติดตั้ง ggplot2 ด้วยคำสั่ง **`install.packages()`**

ติดตั้ง ggplot2

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

- ทดสอบด้วยการเรียกใช้งานด้วยคำสั่ง **`library()`**

เรียกใช้งาน ggplot2

```
> library("ggplot2")
```

Visualization with R: Bar & Line graph

datacube

- ข้อมูลที่ใช้งาน ggplot2 ได้ต้องเป็นประเภท data frame เท่านั้น

สร้างข้อมูล

```
> dat <- data.frame(
                model = factor(c("DT","KNN","NN"),
                levels=c("DT","KNN","NN")),
                accuracy = c(70.5, 93.0, 85.6))
> dat
 #>      model    accuracy
 #> 1   DT         70.5
 #> 2   KNN        93.0
 #> 3   NN         85.6

 # Load the ggplot2 package
 > library(ggplot2)
```


Visualization with R: Bar & Line graph

datacube

- กำหนดให้
  - แกน X คือ ตัวแปร **model**
  - แกน Y คือ ตัวแปร **accuracy** และแสดงความสูงเป็นค่าตัวเลข (โดยการกำหนด parameter **stat="identity"**

สร้างกราฟแท่งแบบพื้นฐาน

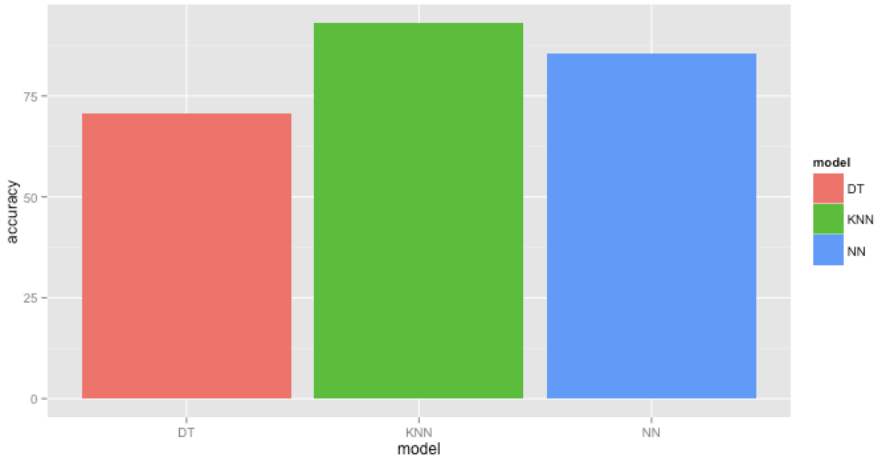
```
> ggplot(data=dat, aes(x=model, y=accuracy))+
          geom_bar(stat="identity")
```

Visualization with R: Bar & Line graph

datacube

สร้างกราฟแท่งแบบมีสี

```
> ggplot(data=dat, aes(x=model, y=accuracy, fill=model))
        + geom_bar(stat="identity")
```

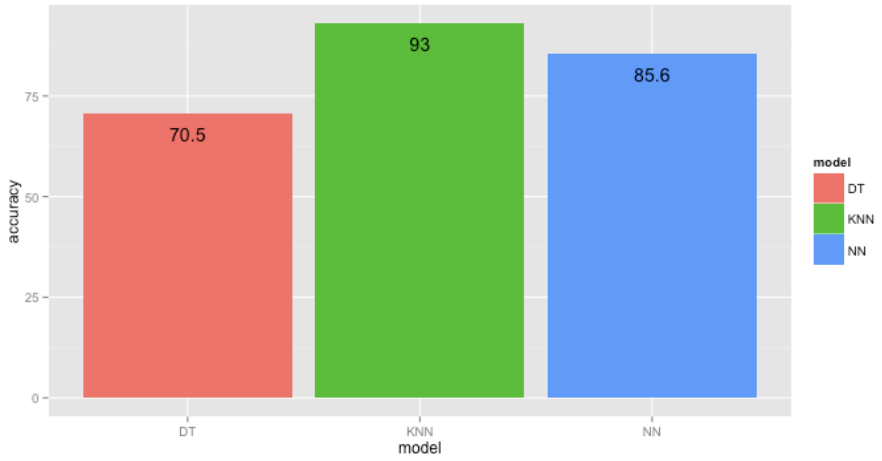


Visualization with R: Bar & Line graph

datacube

สร้างกราฟแท่งแบบมีสี

```
> accuracy = c(70.5, 93.0, 85.6)
> ggplot(data=dat, aes(x=model, y=accuracy, fill=model)) +
        geom_bar(stat="identity") +
        geom_text(aes(label = accuracy), size = 5,
          hjust = 0.5, vjust = 2)
```

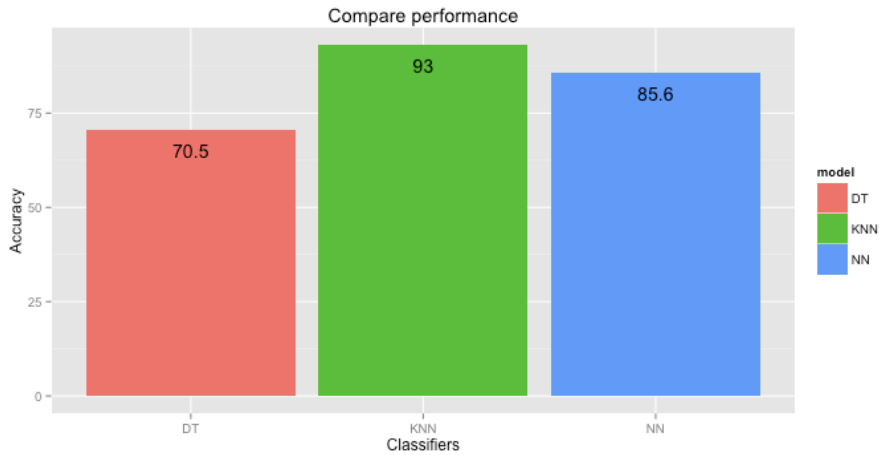


Visualization with R: Bar & Line graph

datacube

สร้างกราฟแท่งแบบมีสี

```
> ggplot(data=dat, aes(x=model, y=accuracy, fill=model)) +
          geom_bar(stat="identity") +
          geom_text(aes(label = accuracy), size = 5,
          hjust = 0.5, vjust = 2) + xlab("Classifiers") +
          ylab("Accuracy") + ggtitle("Compare performance")
```

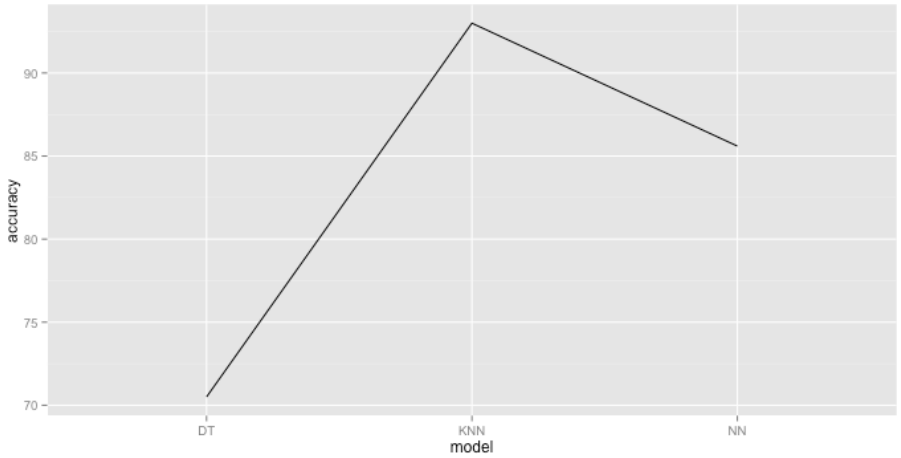


Visualization with R: Bar & Line graph

datacube

สร้างกราฟเส้นแบบพื้นฐาน

```
> ggplot(data=dat, aes(x=model, y=accuracy, group=1))+
          geom_line()
```

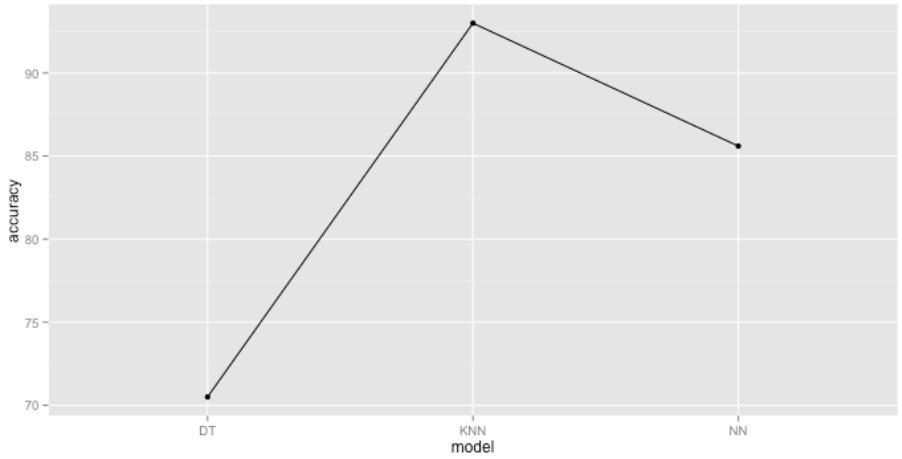


Visualization with R: Bar & Line graph

datacube

สร้างกราฟเส้นและจุด

```
> ggplot(data=dat, aes(x=model, y=accuracy, group=1))+
        geom_line()+
        geom_point()
```

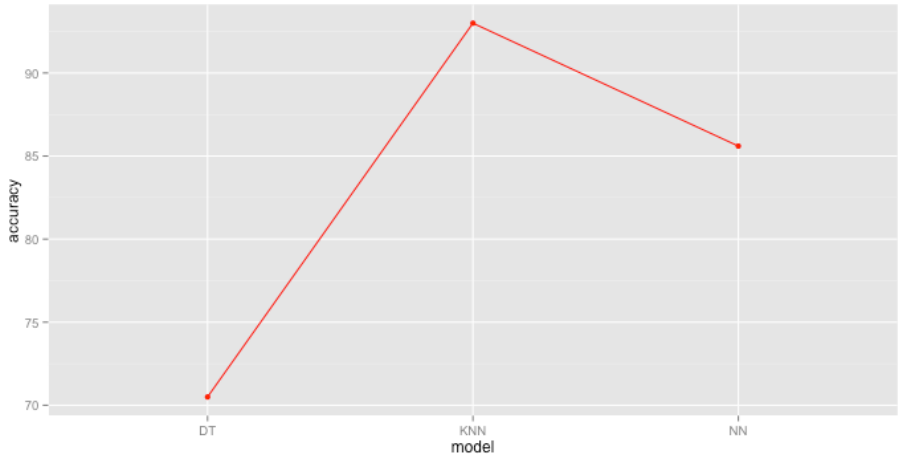


Visualization with R: Bar & Line graph

datacube

สร้างกราฟเส้นและจุดมีสี

```
> ggplot(data=dat, aes(x=model, y=accuracy, group=1))
        + geom_line(colour="red")
        + geom_point(colour="red")
```



Visualization with R: Bar & Line graph

datacube

- ข้อมูลมีมากกว่า 1 ชุด

สร้างข้อมูล

```
> dat2 <- data.frame(dataset =
          factor(c("fold1", "fold2", "fold3", "fold4", "fold5")),
           accuracyDT = c(70.5, 76.2, 77.0, 78.1, 74.5),
           accuracyKNN = c(93.0, 90.2, 91.0, 85.5, 86.6),
           accuracyNN = c(85.6, 80.1, 82.0, 81.0, 80.0))
> dat2
#    dataset accuracyDT accuracyKNN accuracyNN
# 1   fold1       70.5        93.0       85.6
# 2   fold2       76.2        90.2       80.1
# 3   fold3       77.0        91.0       82.0
# 4   fold4       78.1        85.5       81.0
# 5   fold5       74.5        86.6       80.0
```


Visualization with R: Bar & Line graph

datacube

- เปลี่ยนรูปแบบของข้อมูลตามแบบที่ ggplot2 รองรับ

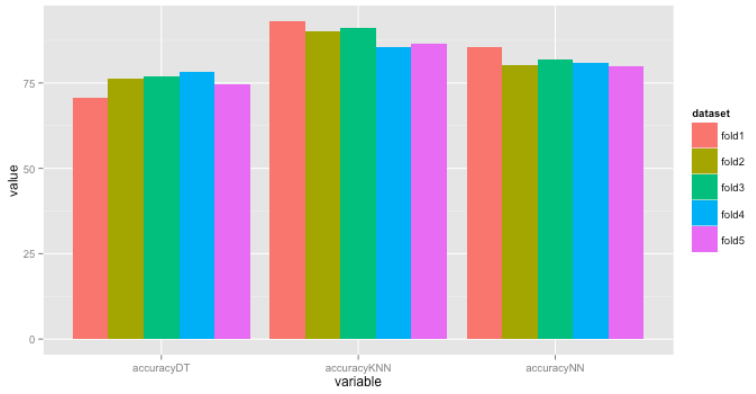
เปลี่ยนรูปแบบข้อมูล

```
> library(reshape2)
> dat3 <- melt(dat2, id.vars="dataset")
> dat3
#    dataset    variable value
1    fold1  accuracyDT  70.5
2    fold2  accuracyDT  76.2
3    fold3  accuracyDT  77.0
4    fold4  accuracyDT  78.1
5    fold5  accuracyDT  74.5
6    fold1 accuracyKNN  93.0
7    fold2 accuracyKNN  90.2
…
```

Visualization with R: Bar & Line graph

datacube

```
> bar <- ggplot(data=dat3, aes(x=variable, y=value,
                group=dataset, fill=dataset)) +
                geom_bar(stat="identity",
                position=position_dodge())
> bar
```

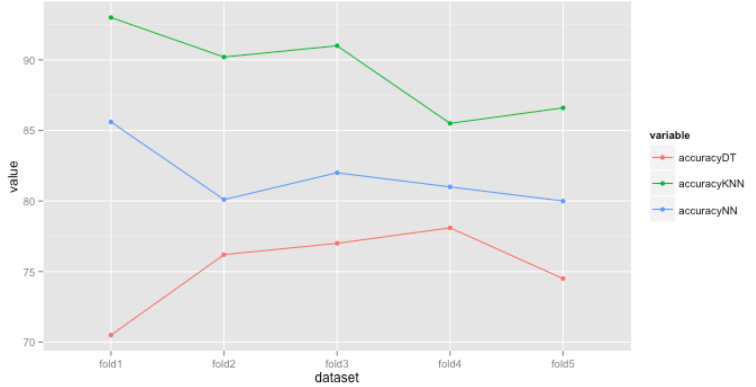


Visualization with R: Bar & Line graph

datacube

สร้างกราฟเส้นแบบหลายชุดข้อมูล

```
> line <- ggplot(data=dat3, aes(x=dataset, y=value,
                group=variable, colour=variable)) +
                geom_line()+ geom_point()
> line
```

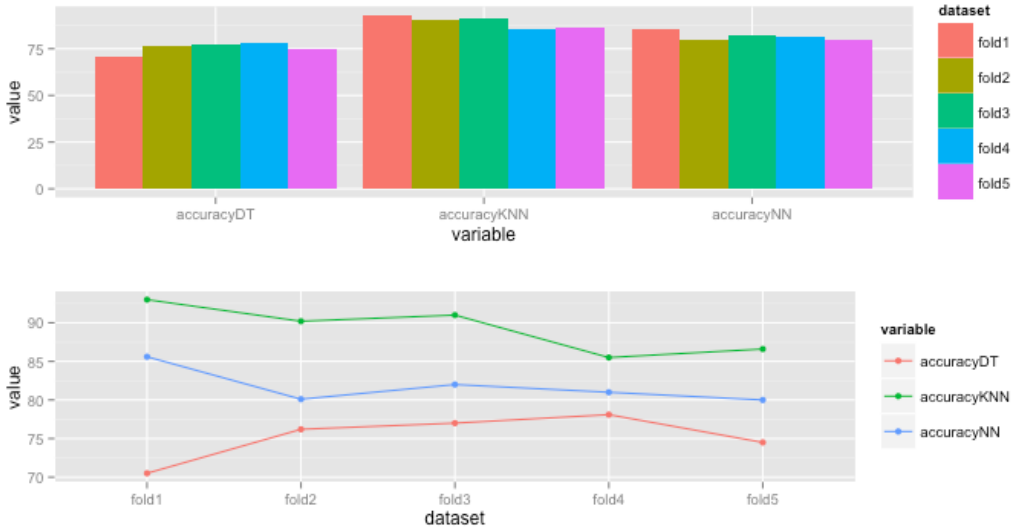


Visualization with R: Bar & Line graph

datacube

สร้างกราฟแบบหลายชุดข้อมูล

```
> source("D:/datacube/software/R/code/function.R")
> multiplot(line,bar)
```



Introduction to R

datacube

- แนะนำ R และ RStudio
- โครงสร้างข้อมูลพื้นฐานใน R
- การอ่านไฟล์ข้อมูล
- การเขียนโปรแกรมภาษา R เบื้องต้น
  - การเลือกเงื่อนไข (IF)
  - การวนรอบ (loop)
  - การเขียนฟังก์ชัน
- การสร้างกราฟด้วย R เบื้องต้น
- การสร้างโมเดล classification ด้วย R เบื้องต้น

Linear Regression model using R

datacube

- ในการสร้างโมเดล linear regression ใช้ฟังก์ชัน **lm()**

การสร้างโมเดล

```
> m <- lm(dv ~ iv, data=mydata)
```

  - โดยที่
    - **dv** คือ dependent variable หรือ แอตทริบิวต์ที่เป็นลาเบลคำตอบ
    - **iv** คือ independent variable หรือ แอตทริบิวต์ที่เป็นแอตทริบิวต์ทั่วไป
    - ทั้ง **dv** และ **iv** จะเป็นแอตทริบิวต์ที่อยู่ในข้อมูล **mydata**
    - ข้อมูล **mydata** คือ training data ที่จะนำมาสร้างโมเดล

ตัวอย่างการสร้างโมเดล

```
> m <- lm(GPA_grad ~ GPA_undergrad, data=grade)
```


Linear Regression model using R

datacube

- ในการนำโมเดล linear regression มา predict ใช้ฟังก์ชัน **predict()**

การใช้งานโมเดล

```
> p <- predict(m, test)
```

  - โดยที่
    - **m** คือ โมเดล linear regression ที่สร้างได้จากฟังก์ชัน **lm()**
    - **test** คือ ข้อมูลที่ต้องการหาคำตอบ (unlabeled data) หรือ อาจจะเป็นข้อมูลทดสอบก็ได้ (test data)

ตัวอย่างการใช้งานโมเดล

```
> p <- predict(m, test)
```

Example 3.1: Linear Regression

- import ไฟล์ insurance_train.csv จากโฟลเดอร์ dataset

import ไฟล์ CSV

```
> insurance_train <- read.csv("dataset/insurance_train.csv")
```

- สร้างโมเดล linear regression โดยการระบุแอตทริบิวต์ที่เกี่ยวข้องทั้งหมด

การสร้างโมเดล

```
> m <- lm(expenses ~ age + children + bmi + sex +
                     smoker + region, data=insurance_train)
```

  - **expenses** คือ ตัวแปรที่เป็นลาเบลคำตอบ
  - **age**, **children**, **bmi**, **sex**, **smoker** และ **region** คือ ตัวแปรที่เป็นแอตทริบิวต์ทั่วไป

Example 3.1: Linear Regression

- สร้างโมเดล linear regression โดยการระบุเครื่องหมาย .

การสร้างโมเดล

```
> m <- lm(expenses ~ ., data=insurance_train)
```

  - เครื่องหมายจุด (.) แทนทุกแอตทริบิวต์ที่ใช้ในการสร้างโมเดล

เรียกดูโมเดล

```
Call:
lm(formula = expenses ~ ., data = insurance_train)

Coefficients:
    (Intercept)              age          sexmale              bmi         children
      -12979.5             259.0             115.2             361.7             462.2
     smokeryes  regionnorthwest  regionsoutheast  regionsouthwest
       24109.9            -873.0            -996.4            -958.8
```

Example 3.1: Linear Regression

- import ไฟล์ insurance_test.csv จากโฟลเดอร์ dataset

import ไฟล์ CSV

```
> insurance_test <- read.csv("dataset/insurance_test.csv")
```

- การนำโมเดลที่ได้มาใช้งาน

ตัวอย่างการใช้งานโมเดล

```
> p <- predict(m, insurance_test)
> as.data.frame(p)
```