



# Introduction to R

Your new data analysis software

# Reminder

1. Email your TA, Yana (yanalo@hku.hk), your group name, your HKU ID and your own names on or before **TOMORROW**.

Up to 6 persons per group.

Let the TA know if you cannot find a group.

2. Bring your laptop with you next week.



# In this class...

You will gain a *very preliminary* understanding of how to use R.

You will learn some simple functions such as linear regression.

You will *not* become an expert in using the programming language.





Which software do you use for data analysis?



# Which software do you use for data analysis?





# SO, WHY CHOOSE R?

R is an open-source software --- basically, it's FREE.

R is easy to use --- no much learning needed.

It is popular --- you can find free resources on R everywhere.

It supports machine learning.





# Let's Download and install R.

Your installation path must not contain any non-English characters. Otherwise, you will have troubles using it.

安裝路徑必須為純英文，否則運行可能出錯





**Next, let's download R-Studio.**  
**It is also free.**

Your installation path must not contain any non-English characters. Otherwise, you will have troubles using it.

安裝路徑必須為純英文，否則運行可能出錯





# Data Types in R

R has several data types:

**Number:** 1, 2, 10.5, 100. These values can be used for calculation (e.g., addition, multiplication).

**String/Character:** "123", "hello", "MKT1000". These values are like English words and cannot be used for calculation (here "123" is not a number).



# Data Types in R

R has several data types:

**Logical:** It only has two values, TRUE and FALSE. You can make branch based on logical value (if TRUE, do something, if FALSE, do something else).

**Integer:** It means the value is an integer. In our class we don't really use it.

**You can use NA to represent missing data.**



# Numerical Operations

```
a = 1
b = 5
result1 = a + b^2
result2 = sqrt(b)
print(result1)
print(result2)
```

```
> a = 1
> b = 5
> result1 = a + b^2
> result2 = sqrt(b)
> print(result1)
[1] 26
> print(result2)
[1] 2.236068
```

# Assigning Values

```
a = 100  
b <- 100 # This is same to b = 100  
print(a)  
print(b)
```

# Comparison

```
a = 100
b <- 100
c1 <- (a == b) # compare if a is equal to b
print(c1)
c2 <- (a < 90) # compare if a is smaller than 90
print(c2)
c3 <- (a >= 100) # compare if a is greater than
or equal to 100
print(c3)
```

# String Operations

```
a = "Big"
b = "Data"
c = 100
d1 = paste(a, b)      #concatenation
d2 = paste0(a, b)    #concatenation
d3 = toString(c)
d4 = paste0(b, c)    #transform a value to a string
print(d1)
print(d2)
print(d3)
print(d4)
```



# String Operations

Note that when referring to a string, you can either put the string between `'` and `'`, or between `"` and `"`.

In other words, in R, `'123'` is equivalent to `"123"`. Both refer to the same string.



# String Operations

```
str = "Marketing and Big Data"  
print(nchar(str))    #number of characters in the  
string  
y = strsplit(str, split=' ')  
#split the string when meeting a space  
print(y)  
z = substr(str, 2, 5)  
#substring from the 2nd to the 5th character  
print(z)
```



# List

```
vec <- c(1, 3, 5)
char_vec <- c("Spring", "Summer", "Autumn",
"Winter")
logic_vec <- c(TRUE, FALSE, TRUE, FALSE)
print(vec)
print(char_vec)
print(logic_vec)
```

# Data Plot

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
plot(x, y)
plot(x, y, type="b")
```

# Data Plot

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
plot(x, y, pch = 17)
```

# Data Plot

The “pch” code defines the appearance of your points

```
plot(x, y, pch = 17)
```

0	1	2	3	4	
□	○	△	+	×	
5	6	7	8	9	
◇	▽	⊠	✱	⬠	
10	11	12	13	14	
⊕	⊗	⊞	⊠	⊞	
15	16	17	18	19	
■	●	▲	◆	●	
20	21	22	23	24	25
●	●	■	◆	▲	▼

# Data Plot

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y1 <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
y2 <- c(1, 2, 4.1, 7, 5, 3, 8, 5, 6.9, 5.0, 6.3)
plot(x, y1, pch = 2, lty = 2, type="b")
lines(x, y2, pch = 1, lty = 1, type="b")
```

# Data Plot

The “lty” code defines the appearance of your line

```
plot(x, y1, pch = 2,  
lty = 2, type="b")
```

0. 'blank'

1. 'solid'

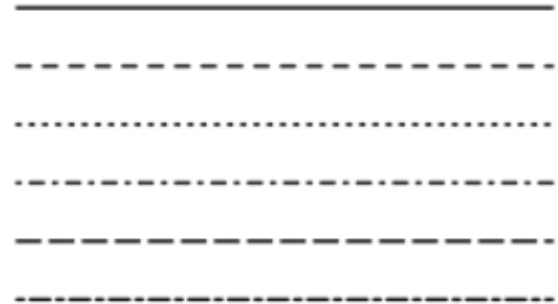
2. 'dashed'

3. 'dotted'

4. 'dotdash'

5. 'longdash'

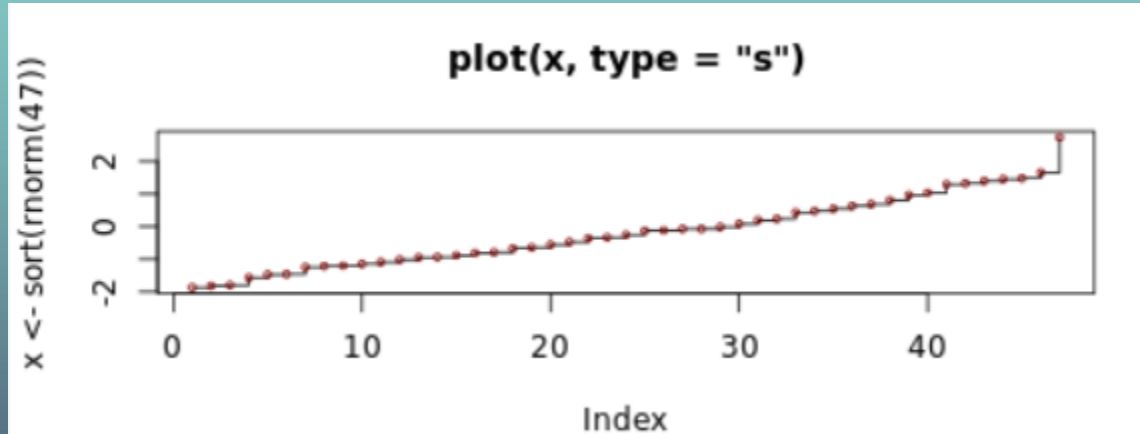
6. 'twodash'



# Data Plot

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y1 <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
y2 <- c(1, 2, 4.1, 7, 5, 3, 8, 5, 6.9, 5.0, 6.3)
plot(x, y1, pch = 2, col = rgb(1, 0, 0), lty = 2,
type="b")
lines(x, y2, pch = 1, col = rgb(0, 1, 1), lty = 1,
type="b")
```

# Data Plot




Indeed, R allows more features than we have described here. Please click [here](#) to find them out and try yourself!





# Data Plot

Now let's consider some advanced plotting functions. In R, we can install and use the “**ggplot2**” package to plot nice figures.



# INSTALL PACKAGES

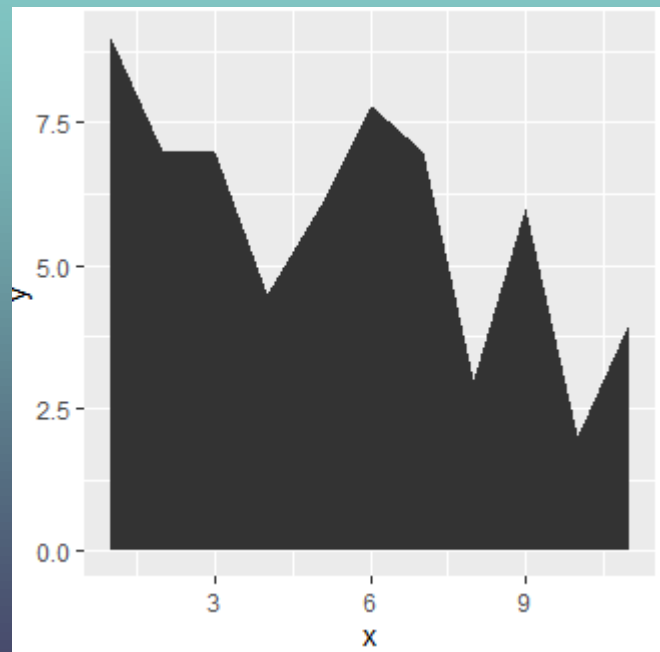
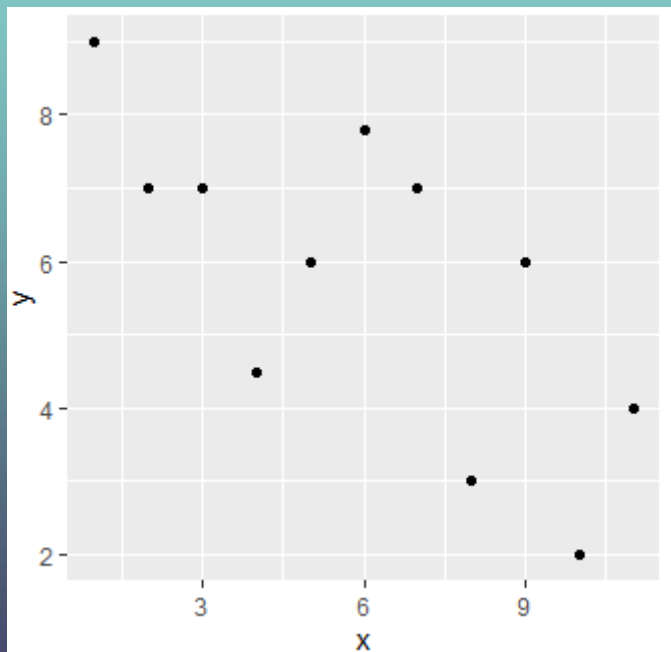
```
> install.packages("car")
Warning in install.packages("car") :
  'lib = "C:/Program Files/R/R-4.0.4/library"' is not writable
--- Please select a CRAN mirror for use in this session ---
also installing the dependencies 'assertthat', 'cpp11', 'digest', 'mime', 'cli'$

There are binary versions available but the source versions are later:
      binary source needs_compilation
tidyr      1.1.2  1.1.3                TRUE
pillar     1.5.0  1.5.1                FALSE
dplyr      1.0.4  1.0.5                TRUE
MatrixModels 0.4-1  0.5-0                FALSE
```

# Data Plot

```
library(ggplot2)
# create data
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
data <- data.frame(x, y)
# Plot
ggplot(data, aes(x, y)) + geom_point()
ggplot(data, aes(x, y)) + geom_area()
```

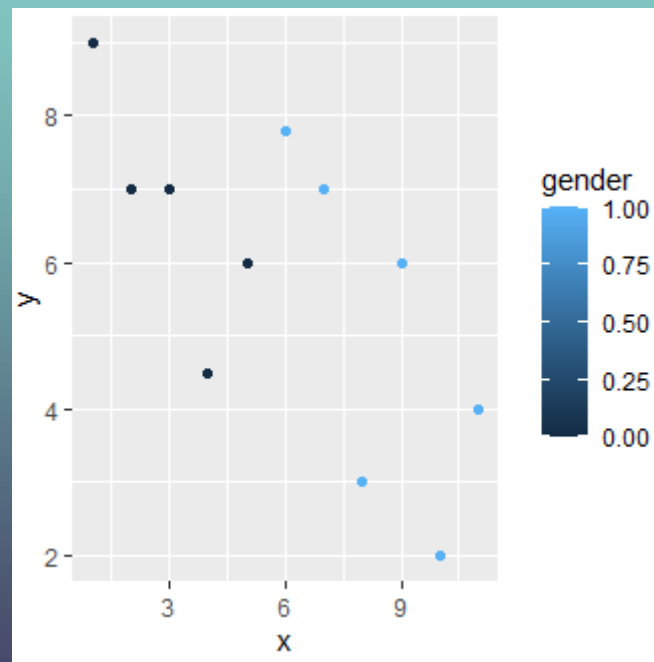
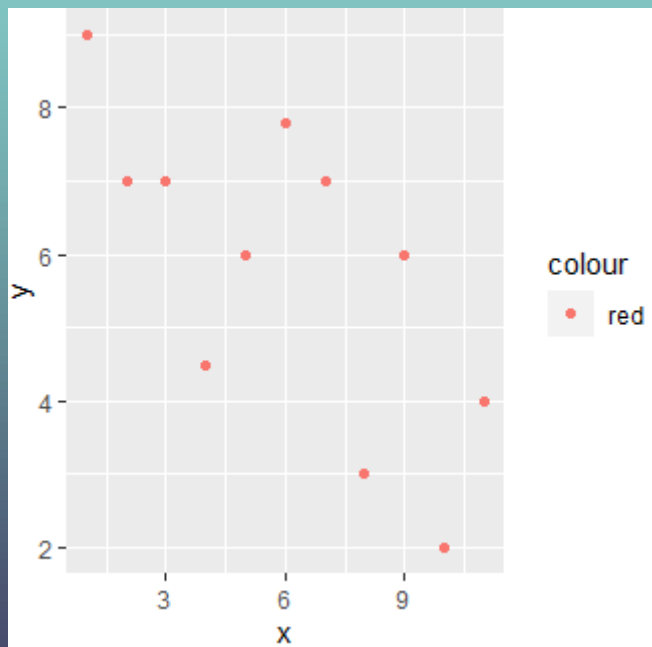
# Data Plot



# Data Plot

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
y <- c(9, 7, 7, 4.5, 6, 7.8, 7, 3, 6, 2, 4)
gender <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
data <- data.frame(x, y, gender)
ggplot(data, aes(x, y, color = 'red')) + geom_point()
ggplot(data, aes(x, y, color = gender)) + geom_point()
```

# Data Plot



# Data Frame

A data frame is a list of variables of the same number of rows with unique row names, given class "data.frame".

```
employees <- data.frame(  
  name = c('Alice', 'Bob', 'Carol', 'Denis'),  
  salary = c(20000, 19000, 23000, 22000),  
  job = c('IT', 'Sales', 'Finance', 'IT' ))
```

# Data Frame

You may have missing values in your data frame. In this case you can enter “NA” to represent the missing value.

```
employees <- data.frame(  
  name = c('Alice', 'Bob', 'Carol', 'Denis'),  
  salary = c(20000, NA, 23000, 22000),  
  job = c('IT', 'Sales', NA, 'IT'))
```



# Data Frame

You can use the dollar sign "\$" to select a specific variable:

```
print (employees)
print (summary (employees))
print (employees$name)
```

# Statistics

```
vector <- c(0, 8, 4, 6, 7, 9, 5)
print(mean(vector))
print(median(vector))
print(var(vector))      #variance
print(sd(vector))      #standard deviation
print(max(vector))     #maximum
print(min(vector))     #mimimum
print(sort(vector))    #sort the data in increasing order
```

# If ... Else Operations

```
x <- 0
if (x < 0) {
  print("Negative number")
} else if (x > 0) {
  print("Positive number")
} else {
  print("Zero")
}
```

# While LOOP

```
count = 0
while (count <= 5)
{
    count = count + 1
    print(count)
}
```

# For LOOP

```
vector = c(1, 3, 5, 7)
print(vector)
for (item in vector)
  print(item)
```

# Scan the QR code to join a survey

Code:



# For LOOP (continued)

```
for (year in 2000: 2020)
    print (year)
```

```
for (year in 2000: 2020)
{
    if(year == 2008)
        next           #skip to the next iteration
    print (year)
}
```

# Functions

```
f1 <- function(a) {  
  print(a + 1)  
}
```

```
f1(0.5)
```

```
f1(2)
```



# Functions (Continued)

```
f2 <- function(a) {  
  return(a+2)  
}
```

```
print(f2(0.5))  
print(f2(2))
```

# Functions (Continued)

```
f3 <- function(a) {  
  a <- toString(a)  
  a <- paste(a, "data")  
  return (a)  
}
```

```
print(f3(100))  
print(f3("big"))
```

# Functions (Continued)

```
f4 <- function(a) {  
  r1 <- a + 1  
  r2 <- a + 2  
  mylist <- list("r1" = r1, "r2" = r2)  
  return(mylist)  
}
```

```
mylist <- f4(15)  
print(mylist$r1)  
print(mylist$r2)
```

# Generating Random Numbers

```
a = runif(1)    #generate a random number between 0 and 1
print(a)
vec = runif(5) #generate a list of 5 random numbers
print(vec)
vec = runif(3, min=0, max=100)
#generate 3 random numbers between 0 and 100
print(vec)
```

# Generating Random Numbers

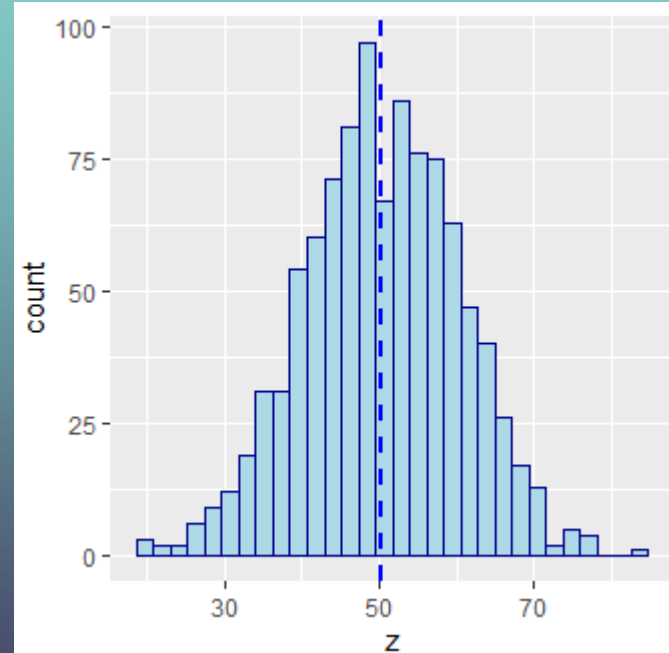
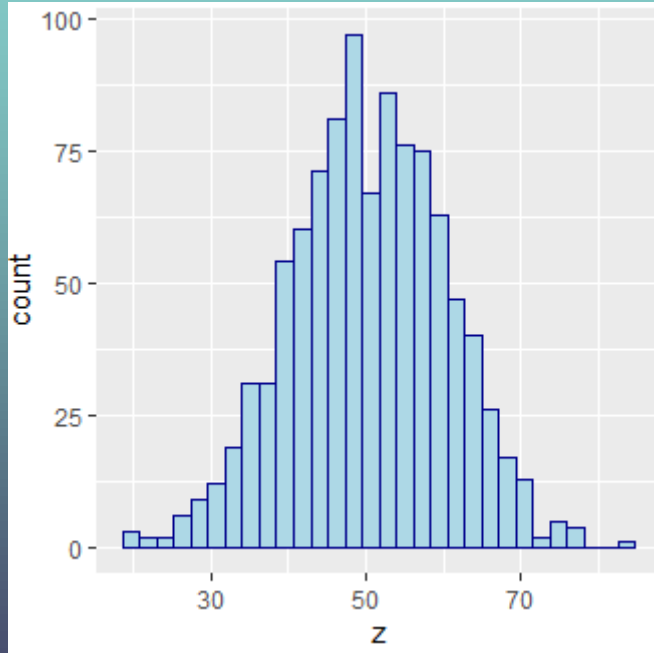
```
x = rnorm(1)
#generate a random number using the standard normal
distribution
print (x)
y = rnorm(4, mean=50, sd=10)
#generate 4 random numbers following the specified normal
distribution
print (y)
z <- rnorm(1000, mean=50, sd=10)
hist(z)
#generate the histogram of z
```

# Plotting Histograms

```
library(ggplot2)
z <- rnorm(1000, mean=50, sd=10)
data <- data.frame(z)
```

```
figure <- ggplot(data, aes(z)) +
  geom_histogram(color="darkblue", fill="lightblue")
figure
figure + geom_vline(aes(xintercept=mean(z)),
                    color="blue", linetype="dashed",
                    size=1)
```

# Plotting Histograms



# Directory

```
getwd()  
#get working directory
```

```
setwd('C:/Users/Xi/Dropbox/Marketing  
Classes/Algorithm')  
#set working directory  
getwd()
```



# Write to text files

```
file1<-file("output.txt")  
writeLines(c("Big", "Data"), file1)  
close(file1)
```

```
file2<-file("C:/Users/Xi/Dropbox/Marketing  
Classes/output.txt")  
writeLines(c("Big", "Data"), file2)  
close(file2)
```

# Directory

Here, “C:/Users/Xi/Dropbox/Marketing Classes/output.txt” is the path to your txt file. You can think of it as the address of your txt file.

You can also write “C:\Users\Xi\Dropbox\Marketing Classes\output.txt”

However, you cannot write

“C:\Users\Xi\Dropbox\Marketing Classes\output.txt”.

# Write to text files (Continued)

Here is another way to do this:

```
sink("output.txt")  
cat("Big")  
cat("\n")           #set up a new line  
cat("Data")  
sink()
```

# Write to text files (Continued)

Now let's write a dataframe:

```
employees <- data.frame(  
  name = c('Alice', 'Bob', 'Carol', 'Denis'),  
  salary = c(20000, NA, 23000, 22000),  
  job = c('IT', 'Sales', NA, 'IT'))
```

```
setwd('C:/Users/Xi/Dropbox/Marketing  
Classes/Algorithm')
```

```
write.table(employees, file = "output.txt", sep =  
"\t", row.names = FALSE)
```

# Reading Data Files

R allows you to read data from various files. If you want to read a spreadsheet, you are recommended to save the file as a csv file (Comma-Separated Values), and open it with the following codes:

 Untitled - R Editor

```
mydata <- read.csv("C:/Users/Xi/Dropbox/r-exercise.csv",  
                  fileEncoding="UTF-8-BOM")
```

# Reading Data Files

You can print the first five rows of the data to see if it works well:

```
> head(mydata)
  Rating Expertise Votes Purpose
1      4         6     0  couple
2      5         5     0  friend
3      5         5     0  family
4      4         4     0  family
5      4         4     0 business
6      5         5     0  family
```

# Summary Statistics of the Data

To see the summary statistics of the data

```
> summary(mydata)
```

Rating	Expertise	Votes	Purpose
Min. :1.000	Min. :0.000	Min. : 0.0000	Length:180635
1st Qu.:4.000	1st Qu.:1.000	1st Qu.: 0.0000	Class :character
Median :5.000	Median :3.000	Median : 0.0000	Mode :character
Mean :4.286	Mean :2.892	Mean : 0.8217	
3rd Qu.:5.000	3rd Qu.:5.000	3rd Qu.: 1.0000	
Max. :5.000	Max. :6.000	Max. :75.0000	

# Choose a Subset of Data

Suppose that we only want to use reviews with rating  $\leq 4$ .

```
> subdata=subset(mydata, Rating <= 4)
> head(subdata)
```

	Rating	Expertise	Votes	Purpose
1	4	6	0	couple
4	4	4	0	family
5	4	4	0	business
7	4	3	0	family
11	4	4	0	couple
12	4	5	0	couple

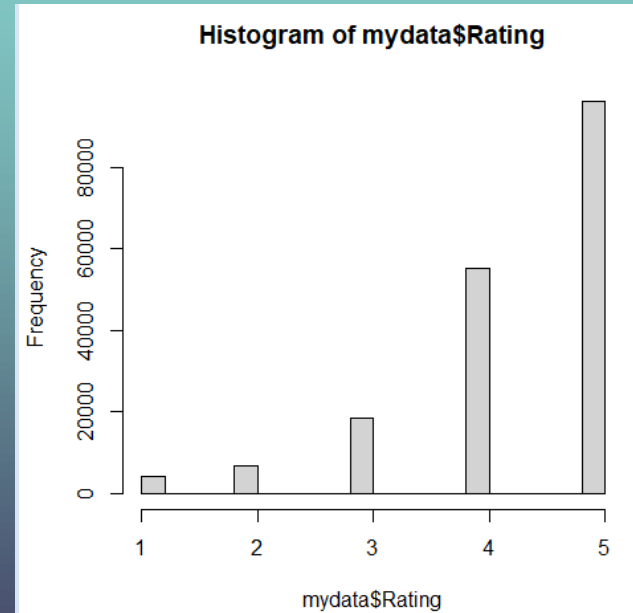


# Number of rows and columns

```
> nrow(mydata)
[1] 180635
> ncol(mydata)
[1] 4
```

# Histogram

```
hist(mydata$Rating)
```



# Linear Regression

Suppose that you want to do the following regression analysis:

$$\text{Rating} = a + b_1 \text{Experience}$$

```
result = lm(Rating ~ Expertise, data = mydata)
summary(result)
```

- Here “lm” stands for “linear model”.

# Linear Regression

```
> summary(result)

Call:
lm(formula = Rating ~ Expertise, data = mydata)

Residuals:
    Min       1Q   Median       3Q      Max
-3.3326 -0.3003  0.6674  0.7158  0.7642

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.332610   0.003872 1118.98  <2e-16 ***
Expertise    -0.016138   0.001091  -14.79  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9531 on 180633 degrees of freedom
Multiple R-squared:  0.001209, Adjusted R-squared:  0.001203
F-statistic: 218.6 on 1 and 180633 DF,  p-value: < 2.2e-16
```

# Linear Regression

This means you get the following result:

$$\text{Rating} = 4.332 - 0.016 \text{ Experience}$$

In addition, we get the significance value of experience (p-value) is smaller than  $2 \times 10^{-16} \ll 1\%$ , meaning that the coefficient is significantly different from 0. This implies that experienced reviewers give significant high ratings (to hotels).

# Linear Regression

We can make predictions based on the regression output. For example, suppose we have another review with expertise 4, then you can do the followings:

```
> prediction <- predict(result, data.frame(Expertise = 4))
> print(prediction)
      1
4.26806
```

# Linear Regression

Likewise, we can also run multiple regression:

```
> result = lm(Votes ~ Expertise + Rating, data = mydata)
> summary(result)

Call:
lm(formula = Votes ~ Expertise + Rating, data = mydata)

Residuals:
    Min       1Q   Median       3Q      Max
-1.421 -0.860 -0.686  0.301  74.301

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.573719   0.019761  79.637 <2e-16 ***
Expertise    0.004350   0.001979   2.198  0.028 *
Rating      -0.178399   0.004264 -41.840 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Linear Regression

Similarly, we can also make predictions based on the regression result:

```
> prediction <- predict(result, data.frame(Expertise = 4, Rating = 2))  
> print(prediction)  
      1  
1.23432
```



# Linear Regression

Moreover, we can also run linear regression with fixed effects: Here, we take purpose as a fixed effect which takes the following values: business, couple, family, friend, solo, and unknown.

```
result = lm(Votes ~ Expertise + Rating + factor(Purpose), data = mydata)
summary(result)
```

# Linear Regression

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	1.405531	0.020426	68.812	< 2e-16	***
Expertise	0.004507	0.001960	2.299	0.0215	*
Rating	-0.180778	0.004238	-42.660	< 2e-16	***
factor(Purpose) couple	0.176869	0.011199	15.794	< 2e-16	***
factor(Purpose) family	0.091479	0.012095	7.564	3.94e-14	***
factor(Purpose) friend	0.065447	0.016380	3.996	6.46e-05	***
factor(Purpose) solo	0.081461	0.018771	4.340	1.43e-05	***
factor(Purpose) Unknown	1.130019	0.016878	66.954	< 2e-16	***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Linear Regression

And making predictions accordingly...

```
> prediction <- predict(result,  
+ data.frame(Expertise = 4, Rating = 2, Purpose = "family"))  
> print(prediction)  
      1  
1.153482
```

# Organizing Regression Output

Suppose that we want to save the regression result in an organized matter, then we can use the “stargazer” package which can be installed on R.

```
library(stargazer)
stargazer(result, title="Regression Results",
align=TRUE, out="result.html", type = "html")
```

### Regression Results

	<i>Dependent variable:</i>
	Votes
Expertise	0.005** (0.002)
Rating	-0.181*** (0.004)
factor(Purpose)couple	0.177*** (0.011)
factor(Purpose)family	0.091*** (0.012)
factor(Purpose)friend	0.065*** (0.016)
factor(Purpose)solo	0.081*** (0.019)
factor(Purpose)Unknown	1.130*** (0.017)
Constant	1.406*** (0.020)
Observations	180,635
R <sup>2</sup>	0.036
Adjusted R <sup>2</sup>	0.036
Residual Std. Error	1.704 (df = 180627)
F Statistic	968.891*** (df = 7; 180627)
<i>Note:</i>	* p<0.1; ** p<0.05; *** p<0.01

# Organizing Regression Output

We can also contrast the regression output from different model specifications:

```
result0 = lm(Votes ~ Expertise, data = mydata)
result1 = lm(Votes ~ Expertise + Rating, data = mydata)
result2 = lm(Votes ~ Expertise + Rating + factor(Purpose), data = mydata)
stargazer(result0, result1, result2, title="Regression Results",
align=TRUE, out="result.html", type = "html")
```

### Regression Results

	<i>Dependent variable:</i>		
	(1)	Votes (2)	(3)
Expertise	0.007 <sup>***</sup> (0.002)	0.004 <sup>**</sup> (0.002)	0.005 <sup>**</sup> (0.002)
Rating		-0.178 <sup>***</sup> (0.004)	-0.181 <sup>***</sup> (0.004)
factor(Purpose)couple			0.177 <sup>***</sup> (0.011)
factor(Purpose)family			0.091 <sup>***</sup> (0.012)
factor(Purpose)friend			0.065 <sup>***</sup> (0.016)
factor(Purpose)solo			0.081 <sup>***</sup> (0.019)
factor(Purpose)Unknown			1.130 <sup>***</sup> (0.017)
Constant	0.801 <sup>***</sup> (0.007)	1.574 <sup>***</sup> (0.020)	1.406 <sup>***</sup> (0.020)
Observations	180,635	180,635	180,635
R <sup>2</sup>	0.0001	0.010	0.036
Adjusted R <sup>2</sup>	0.0001	0.010	0.036
Residual Std. Error	1.736 (df = 180633)	1.727 (df = 180632)	1.704 (df = 180627)
F Statistic	13.230 <sup>***</sup> (df = 1; 180633)	881.976 <sup>***</sup> (df = 2; 180632)	968.891 <sup>***</sup> (df = 7; 180627)

*Note:*

\* p<0.1; \*\* p<0.05; \*\*\* p<0.01

# Organizing Summary Statistics

We can easily generate the summary statistics of our dataset:

```
stargazer(mydata, title="Summary Statistics",  
align=TRUE, out="summary.html", type = "html")
```



# Organizing Summary Statistics


Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
Rating	180,635	4.286	0.954	1	4	5	5
Expertise	180,635	2.892	2.055	0	1	5	6
Votes	180,635	0.822	1.736	0	0	1	75



# Another Example of Linear Regression

In this regression, our dataset comes from Los Angeles Neighborhoods Data. The data source is [here](#).

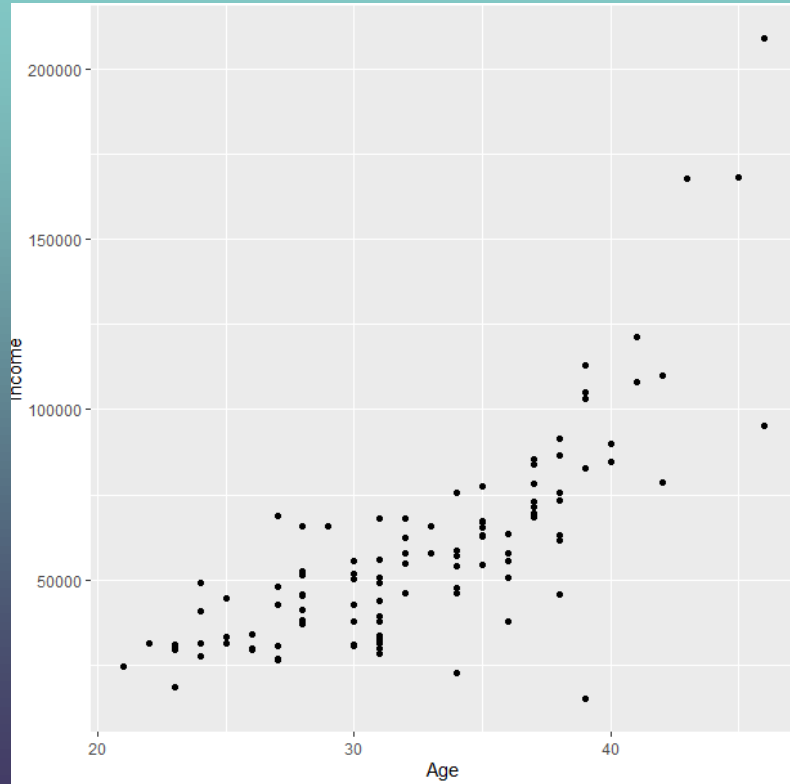
It covers some basic information of several neighborhoods in Los Angeles (e.g., income, age, ethnic group, ...)



# Another Example of Linear Regression

```
require(ggplot2)
file = "C:/Users/Xi/Dropbox/Marketing
Classes/Algorithm/r-exercise.txt"
mydata <- read.table(file, header = TRUE)
ggplot(mydata, aes(y=Income, x=Age)) + geom_point()
```

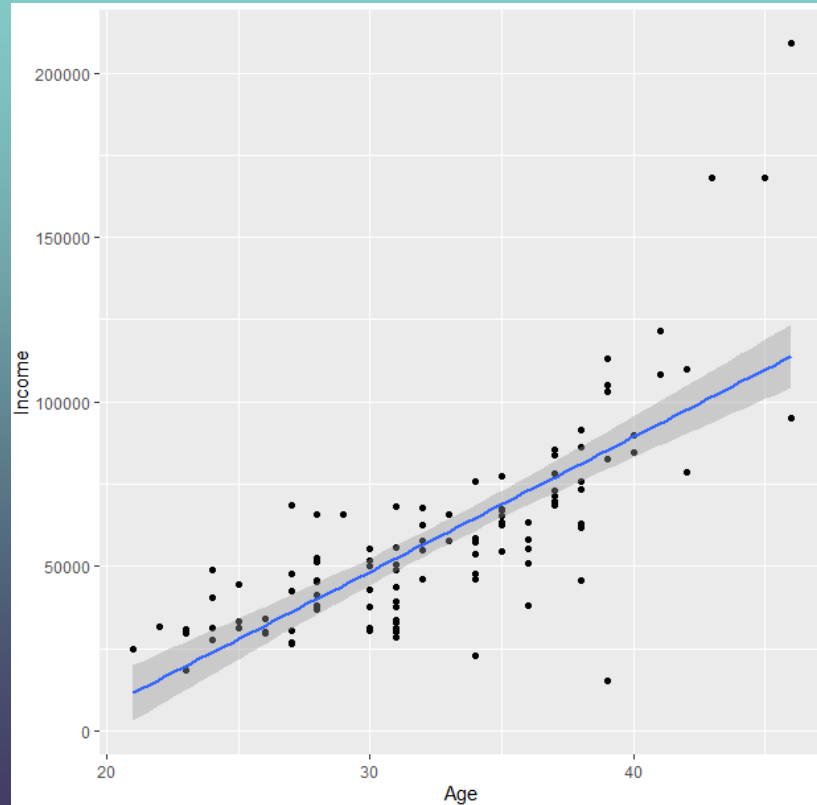
# Another Example of Linear Regression



# Another Example of Linear Regression

```
result <- lm(Income ~ Age, data = mydata)
summary(result)
ggplot(mydata, aes(y=Income, x=Age)) + geom_point() + geom_smooth(method="lm")
```

# Another Example of Linear Regression



# T-test

It is very convenient to run t-tests on R:

```
x = c(1, 3, 3, 5, 3, 2, 4, 3, 5, 7)
y = c(2, 6, 3, 4, 5, 2, 5, 8, 1, 6)
t.test(x, y)
```

# T-test

```
Welch Two Sample t-test
```

```
data:  x and y
```

```
t = -0.68034, df = 16.975, p-value = 0.5055
```

```
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
```

```
-2.46089  1.26089
```

```
sample estimates:
```

```
mean of x mean of y
```

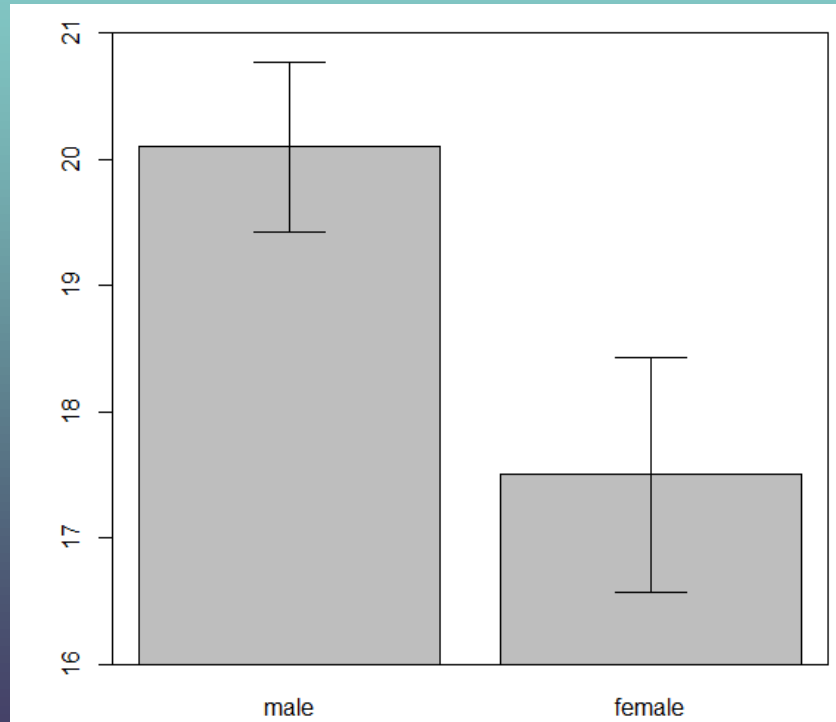
```
3.6      4.2
```



# Visualize t-test

```
male <- c(18,22,21,17,20,17,23,20,22,21)
female <- c(16,20,14,21,20,18,13,15,17,21)
data      = c(mean(male), mean(female))
names(data) = c("male", "female")
se        = c(sd(male)/sqrt(length(male)),
              sd(female)/sqrt(length(female)))
windows()
bp = barplot(data, ylim=c(16, 21), xpd=FALSE)
box()
arrows(x0=bp, y0=data-se, y1=data+se, code=3, angle=90)
```

# Visualize t-test



# Reminder

1. Email your TA, Yana (yanalo@hku.hk), your group name, your HKU ID and your own names on or before **TOMORROW**.

Up to 6 persons per group.

Let the TA know if you cannot find a group.

2. Bring your laptop with you next week.

# Scan the QR code to join a test

Code:

