Statistics and Data Analysis

R Programming

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Road map

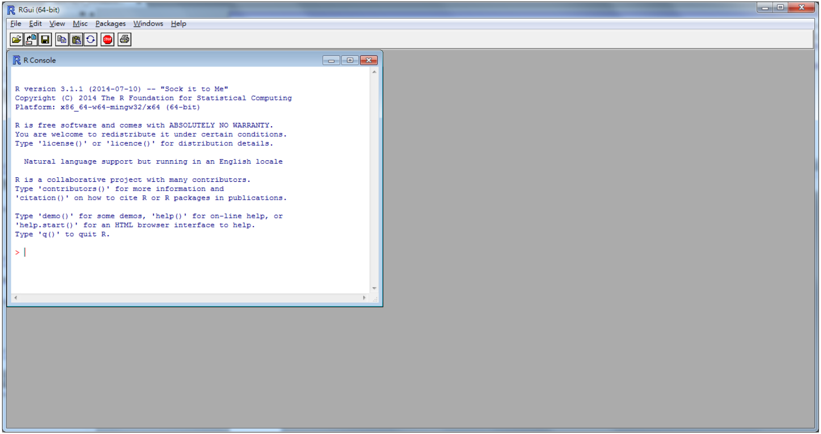
- **The R programming language**.
- More functions and techniques.
- Regression in R.

The R programming language

- **R** is a programming language for statistical computing and graphics.
- R is open source.
- R is powerful and flexible.
  - It is fast.
  - Most statistical methods have been implemented as packages.
  - One may write her own R programs to complete her own task.
- `http://www.r-project.org/`.
- To download, go to `http://cran.csie.ntu.edu.tw/`, choose your platform, then choose the suggested one (the current version is 3.2.3).

The programming environment

- When you run R, you should see this:



Try it!

- Type some mathematical expressions!

```
> 1 + 2
[1] 3
> 6 * 9
[1] 54
> 3 * (2 + 3) / 4
[1] 3.75
> log(2.718)
[1] 0.9998963
> 10 ^ 3
[1] 1000
> sqrt(25)
[1] 5
```

Let's do statistics

- A wholesaler has 440 customers in Portugal:
  - 298 are "horeca"s (hotel/restaurant/café).
  - 142 are retails.
- These customers locate at different regions:
  - Lisbon: 77.
  - Oporto: 47.
  - Others: 316.
- Data source: `http://archive.ics.uci.edu/ml/datasets/Wholesale+customers`.

Let's do statistics

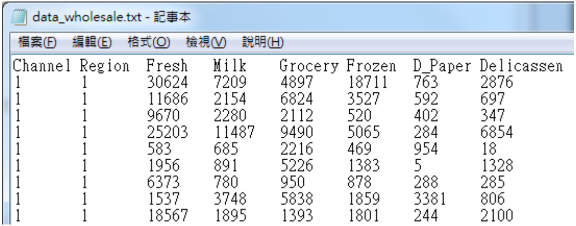
- The data:

Channel	Label	Fresh	Milk	Grocery	Frozen	D. & P.	Deli.
1	1	30624	7209	4897	18711	763	2876
1	1	11686	2154	6824	3527	592	697
				⋮			
2	3	14531	15488	30243	437	14841	1867

- The wholesaler records the annual amount each customer spends on six product categories:
  - Fresh, milk, grocery, frozen, detergents and paper, and delicatessen.
  - Amounts have been scaled to be based on "monetary unit."
- Channel: hotel/restaurant/café = 1, retailer = 2.
- Region: Lisbon = 1, Oporto = 2, others = 3.

Data in a TXT file

- The data are provided in an MS Excel worksheet "wholesale."
- Let's **copy and paste** the data to a TXT file "wholesale.txt."
- Copying data from Excel and pasting them to a TXT file will make data in columns **separated by tabs**.



Channel	Region	Fresh	Milk	Grocery	Frozen	D_Paper	Delicassen
1	1	30624	7209	4897	18711	763	2876
1	1	11686	2154	6824	3527	592	697
1	1	9670	2280	2112	520	402	347
1	1	25203	11487	9490	5065	284	6854
1	1	583	685	2216	469	954	18
1	1	1956	891	5226	1383	5	1328
1	1	6373	780	950	878	288	285
1	1	1537	3748	5838	1859	3381	806
1	1	18567	1895	1393	1801	244	2100

- DO NOT modify anything after pasting even if data are not aligned perfectly. Just copy and paste.

Reading data from a TXT file

- Let's put the TXT file to your **work directory**.
  - A file should be put in the work directory for R to read data from it.[1]
- To find the default work directory:[2]

```
> getwd()
[1] "C:/Users/user/Documents"
```

- To **read** the data into R, we execute:

```
> W <- read.table("wholesale.txt", header = TRUE)
```

  - `W` is a **data frame** that stores the data.
  - `<-` assigns the right-hand-side values to the variable at its left.

[1] Or one may use `setwd()` to choose an existing folder as the work directory.
[2] The work directory on your computer may be different from mine.

Browsing data

- To browse the data stored in a data frame:

```
> W
> head(W)
> tail(W)
```

- To extract a row or a column:

```
> W[1, ]
> W$Channel
> W[, 1]
```

- What is this?

```
> W[1, 2]
```

Extracting more rows or columns

- To extract multiple rows or columns:

```
> W[1:6, ]
> W[, 1:3]
> head(W[, 1:3])
```

- How about nonconsecutive rows or columns?

```
> W[c(1, 4:6), ]
> head(W[, c(2, 5:6)])
```

- In general, `c()` does all kinds of concatenations and `i:j` produces a sequence of integers from `i` to `j`.
- How about these?

```
> head(data.frame(W$Channel, W$Region))
> head(data.frame(Channel = W$Channel, Region = W$Region))
```

Road map

- The R programming language.
- **More functions and techniques**.
- Regression in R.

Basic statistics

- The **mean** (average) expenditure on milk:

```
> mean(W$Milk)
```

- The **sample standard deviation** of expenditure on milk:

```
> sd(W$Milk)
```

- What is the mean expenditure on milk for those who
  - live in Lisbon (`Region` is 1) and
  - consume at hotel/restaurant/café (`Channel` is 1)?

```
> mean(W$Milk[1:59])
```

- There must be a better way!

Extracting rows by conditions

- Let's find those records for consumption at hotel/restaurant/café:

```
> which(W$Channel == 1)
```

  - `which()` takes a vector and examine whether each element satisfies the given condition. If so, it returns that index.
  - `W$Channel[1]` is 1, `W$Channel[400]` is 2, etc.
- `=` is for **assignment** and `==` is for **comparison**!
  - To assign a value to a variable, use `=`.
  - To test whether two values are equal, use `==`.
- Now, we know what this is:

```
> mean(W$Milk[which(W$Channel == 1)])
```

- What is next?

Combining conditions

- To specify an "and" operation, use & (ampersand).

```
> mean(W$Milk[which(W$Channel == 1 & W$Region == 1)])
```

- To specify an "or" operation, use | (bar).

```
> mean(W$Milk[which(W$Channel == 1 | W$Region == 1)])
```

- To specify a "not" operation, use ! (exclamation).

```
> mean(W$Milk[which(W$Channel == 1 | !(W$Region == 1))])
```

- This also works:

```
> index <- which(m$Channel == 1 & m$Region == 1)
> mean(m$Milk[index])
```

Exercises

- Fill in this table:

Channel	Region		
	1	2	3
1	3870.20		
2			

Mean expenditures on milk

Some more basic statistics

- Counting:

```
> length(which(W$Channel == 1 & W$Region == 1))
```

- Median:

```
> median(W$Milk[which(W$Channel == 1 & W$Region == 1)])
```

- Maximum and minimum:

```
> max(W$Milk[which(W$Channel == 1 & W$Region == 1)])
> min(W$Milk[which(W$Channel == 1 & W$Region == 1)])
```

- Correlation coefficient:

```
> a <- W$Milk[which(W$Channel == 1 & W$Region == 1)]
> b <- W$Grocery[which(W$Channel == 1 & W$Region == 1)]
> cor(a, b)
[1] 0.654953
```

Basic statistics

- **Correlation coefficient**:

```
> cor(W$Milk, W$Grocery)
```

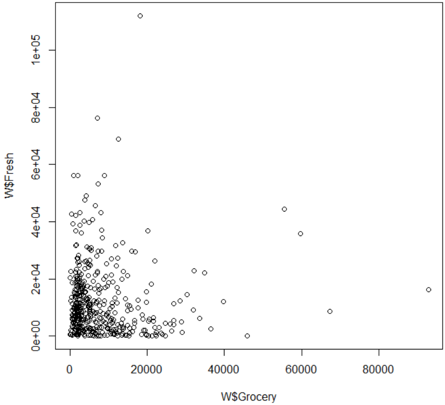
- In fact, you may simply do:

```
> W2 <- W[, 3:8]
> cor(W2)
```

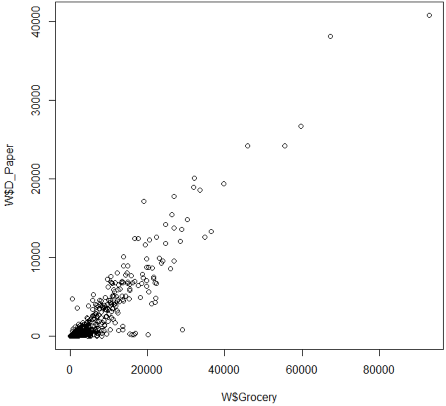
  - `3:8` is a vector $(3, 4, 5, 6, 7, 8)$.
  - `W[, 3:8]` is the third to the eighth columns of `W`.
  - `cor(W2)` is the **correlation matrix** for pairwise correlation coefficients among all columns of `W2`.

Basic graphs: Scatter plots

```
> plot(W$Grocery, W$Fresh)
```

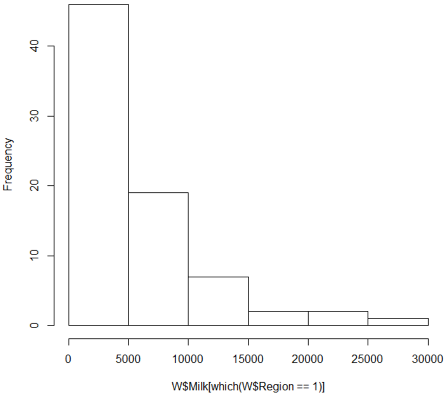



```
> plot(W$Grocery, W$D_Paper)
```



Basic graphs: histograms

```
> hist(W$Milk[which(W$Region == 1)])
```



Writing scripts in a file

- It is suggested to **write scripts** (codes) in a **file**.
  - This makes the codes easily modified and reusable.
  - Multiple statements may be executed at the same time.
  - These codes can be stored for future uses.
- To do so, open a new script file in R and then write codes line by line.
  - Execute a line of codes by pressing "**Ctrl + R**" in Windows or "**Command + return (enter)**" in Mac.
  - Select **multiple lines of codes** and then execute all of them together in the same way.
- In your file, put **comments** (personal notes of your program) after **#**. Characters after # will be ignored when executing a line of codes.
- The saved .R files can be edit by any **plain text editor**.
  - E.g., Notepad in Windows.

Storing data to a TXT file

- To store the results of our calculation permanently:

```
> C <- cor(W[, 3:8])
> write.table(C, "cor_wholesale.txt")
> write.table(C, "cor_wholesale.txt", col.names = NA,
              row.names = TRUE, quote = FALSE, sep = "\t")
```

- Before you close your R environment:
  - Save the current work **image** to store all the variables and their values.

Road map

- The R programming language.
- More functions and techniques.
- **Regression in R**.

Regression in R

- Let's do regression in R. First, let's load the data:
  - Copy all the data in the MS Excel worksheet "bike_day."
  - Paste them into a TXT file with "bike.txt" as the file name.
  - Put the file in the work directory.
  - Execute

    ```
    B <- read.table("bike_day.txt", header = TRUE)
    ```

- Take a look at B:

  ```
  head(B)
  mean(B$cnt)
  cor(B$cnt, B$temp)
  hist(B$cnt)
  ```

- Try them!

  ```
  pairs(B)
  pairs(B[, 10:16])
  ```

Simple regression

- Let's build a **simple regression** model by using the function `lm()`:

```
fit <- lm(B$cnt ~ B$instant)
summary(fit)
```

  - Put the dependent variable **before** the ~ operator.
  - Put the independent variable **after** the ~ operator.

- We will obtain the regression report:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2392.9613   111.6133   21.44   <2e-16 ***
B$instant      5.7688     0.2642   21.84   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1507 on 729 degrees of freedom
Multiple R-squared:  0.3954,    Adjusted R-squared:  0.3946
F-statistic: 476.8 on 1 and 729 DF,  p-value: < 2.2e-16
```

Multiple regression

- Let's **add more variables** using the + operator:

```
fit <- lm(B$cnt ~ B$instant + B$workingday + B$temp)
summary(fit)
```

- The regression report:

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -280.3863   138.8325   -2.02   0.0438 *
B$instant       5.0197     0.1925   26.07   <2e-16 ***
B$workingday  145.3731    86.5121    1.68   0.0933 .
B$temp        140.2238     5.4246   25.85   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1086 on 727 degrees of freedom
Multiple R-squared:  0.6871,    Adjusted R-squared:  0.6858
F-statistic: 532.1 on 3 and 727 DF,  p-value: < 2.2e-16
```

Interaction

- Let's consider **interaction** using the * operator:

```
fit <- lm(B$cnt ~ B$instant + B$workingday * B$temp)
summary(fit)
```

- The regression report:

```
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            -631.776    204.732  -3.086  0.00211 **
B$instant                 5.026      0.192  26.183  < 2e-16 ***
B$workingday            675.120    243.232   2.776  0.00565 **
B$temp                  157.912      9.323  16.938  < 2e-16 ***
B$workingday:B$temp     -26.471     11.364  -2.329  0.02012 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1083 on 726 degrees of freedom
Multiple R-squared:  0.6894,    Adjusted R-squared:  0.6877
F-statistic: 402.9 on 4 and 726 DF,  p-value: < 2.2e-16
```

Qualitative variables

- Let's add a non-binary **qualitative variable** (in a **wrong** way):

```
fit <- lm(B$cnt ~ B$instant + B$workingday * B$temp + B$season)
summary(fit)
```

- The regression report:

```
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          -628.7340   208.7156  -3.012  0.00268 **
B$instant               5.0324     0.2085  24.141  < 2e-16 ***
B$workingday          675.0576   243.3996   2.773  0.00569 **
B$temp                158.0409     9.4807  16.670  < 2e-16 ***
B$season               -3.1710    41.5623  -0.076  0.93921
B$workingday:B$temp   -26.4682    11.3722  -2.327  0.02022 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1083 on 725 degrees of freedom
Multiple R-squared:  0.6894,    Adjusted R-squared:  0.6873
F-statistic: 321.9 on 5 and 725 DF,  p-value: < 2.2e-16
```

Qualitative variables

- To correctly include a qualitative variable, use the function `factor()`:

```
fit <- lm(B$cnt ~ B$instant + B$workingday * B$temp + factor(B$season))
summary(fit)
```

  - `factor()` tells the R program to interpret those values as categories even if they are numbers.
  - If the values are already non-numeric, there is no need to use `factor()`.
- Let's read the regression report.

Qualitative variables

- The regression report:

```
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          -749.4834   209.3085  -3.581 0.000366 ***
B$instant               5.1296     0.2015  25.459  < 2e-16 ***
B$workingday          632.4411   233.8650   2.704 0.007006 **
B$temp                146.5942    11.7999  12.423  < 2e-16 ***
factor(B$season)2     827.2798   143.1463   5.779 1.12e-08 ***
factor(B$season)3     142.7658   188.6595   0.757 0.449454
factor(B$season)4     272.6144   126.7112   2.151 0.031770 *
B$workingday:B$temp   -24.5086    10.9264  -2.243 0.025195 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1041 on 723 degrees of freedom
Multiple R-squared:  0.7142,    Adjusted R-squared:  0.7115
F-statistic: 258.2 on 7 and 723 DF,  p-value: < 2.2e-16
```

Changing the reference level

- To change the reference level, use the function `relevel()`:

```
season.new <- relevel(factor(B$season), "2")
fit <- lm(B$cnt ~ B$instant + B$workingday * B$temp + season.new)
summary(fit)
```

  - `relevel()` sets a (factored) qualitative variable's reference level (to be the second argument).
  - It does not change the original variable. It returns a **new variable**!
- Let's read the regression report.

Changing the reference level

- The regression report:

```
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            77.7965   271.5195   0.287  0.77456
B$instant               5.1296     0.2015  25.459  < 2e-16 ***
B$workingday          632.4411   233.8650   2.704  0.00701 **
B$temp                146.5942    11.7999  12.423  < 2e-16 ***
season.new1          -827.2798   143.1463  -5.779 1.12e-08 ***
season.new3          -684.5141   124.6621  -5.491 5.54e-08 ***
season.new4          -554.6654   125.5916  -4.416 1.16e-05 ***
B$workingday:B$temp   -24.5086    10.9264  -2.243  0.02520 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1041 on 723 degrees of freedom
Multiple R-squared:  0.7142,    Adjusted R-squared:  0.7115
F-statistic: 258.2 on 7 and 723 DF,  p-value: < 2.2e-16
```

Transformation: method 1

- To add $temp^2$, there are two ways:

```
tempSq <- B$temp^2
fit <- lm(B$cnt ~ B$instant + B$workingday * (B$temp + tempSq))
summary(fit)
```

- The regression report:

```
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         -3313.2904   462.5027  -7.164 1.93e-12 ***
B$instant               4.7928     0.1874  25.576  < 2e-16 ***
B$workingday         1934.5264   578.2195   3.346 0.000863 ***
B$temp                482.5310    50.6541   9.526  < 2e-16 ***
tempSq                 -8.1197     1.2489  -6.501 1.48e-10 ***
B$workingday:B$temp  -180.0186    62.5810  -2.877 0.004138 **
B$workingday:tempSq     3.9116     1.5382   2.543 0.011200 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

Transformation: method 2

- Alternatively, we may create the new variable as a **new column** in the MS Excel worksheet.
- Then copy and paste to update the content in the TXT file.
- Execute `read.table()` again to update the data frame `B`.
- Finally, redo `lm()` and `summary()`.

Fitted values

- Once we execute

```
tempSq <- B$temp^2
fit <- lm(B$cnt ~ B$instant + B$workingday * (B$temp + tempSq))
```

  the object `fit` contains more than the regression report.

- It contains the **fitted values** $\hat{y}_i$:

```
predict(fit)
plot(B$cnt)
points(predict(fit), col =
"red")
```

  - `plot()` makes a scatter plot.
  - `points()` add points onto an existing scatter plot.
  - `col = "red"` makes red points.