Introduction to R (1)

Seminar of the Public Health Division, May 13, 2004, by Minato NAKAZAWA

1 How to install R

Windows Downloading the file to install R-1.8.1 from CRAN mirror site[1], and double-click the icon of rw1081.exe, which will lead to completion of install by responding some dialogues.

Macintosh If your OS version is earlier than MacOS9, the latest version available for you is R-1.7.1. You can use R-1.8.1 or newer version only on MacOS X. Old OS user should download rm171.sit from CRAN mirror[2], and OS X user should download R.dmg[3]. Double-clicking rm171.sit directly leads to installation, but double-clicking R.dmg leads to produce some files including R.pkg. Double-clicking R.pkg again leads to actual installation.

Linux/FreeBSD I cannot explain here. But it may not be so difficult.

2 Basic manipulation

The explanation below is based on Windows version.

Starting Rgui (R with graphical users interface) is done by double-clicking the icon of R on your Desktop[4]. It will open the window where the prompt "`>`" is shown. Entering the R commands (functions) interactively should be done for this prompt. All commands (functions) you have entered can be saved as an single text log file via "Save History ..." of the "File" menu. The log can re-run later by entering `source("/path/to/log-file")` for the prompt (it is equivalent to the "Source" item of "File" menu). On the Windows environment, the delimiter of path should be `/` or `\\`. The command line histories can be revived by entering upward arrow key.

2.1 The most elementary

Exit from R	`q()`
Assign value to variable	`<-`
Defining a function* (eg. return the list of mean and standard deviation)	`meansd <- function(X) { list(mean(X),sd(X)) }`
Installing additional libraries (eg. install vcd from CRAN)	`install.packages("vcd")`

* Defining a function can be done for more complicated purpose using several lines, in which the last line is returned as the value of the function. Variables used within the function have local scopes, which make the original values of the variables remain unchanged if those are assigned new values within the function using `<-`. If you would like to change the original values of variables within a function, you should use `<<-` instead of `<-`. Therefore, for example, defining a function will be done as the next screen, where `>` means the prompt, `+` means a prompt for continued lines, others are the outputs of R system.

[1] `ftp://ftp.u-aizu.ac.jp/pub/lang/R/CRAN/bin/windows/base/rw1081.exe`

[2] `ftp://ftp.u-aizu.ac.jp/pub/lang/R/CRAN/bin/macos/rm171.sit`

[3] `ftp://ftp.u-aizu.ac.jp/pub/lang/R/CRAN/bin/macosx/R.dmg`

[4] The work directory in the property of this icon is to be the current directory of R environment.

```
> x <- 2
> y <- function(a) {
+      x <- x+a
+      x }
> y(5)
[1] 7
> x
[1] 2
> z <- function(a) {
+      x <<- x+a
+      x }
> z(5)
[1] 7
> x
[1] 7
```

2.2 Reading data from files

Reading data to a data frame (`dat`) from the text file of which fields are delimited with "tab" code and the first line is composed of the names of variables	`dat <- read.delim("/path/to/filename")`
Reading data to a data frame (`dat`) from the text file of which fields are delimited with comma and the first line is composed of the names of variables	`dat <- read.csv("/path/to/filename")`

If the data file exists in a current work directory, `/path/to` is unnecessary.

2.3 Configuration of the types of variables

Defining a variable's type as categorical (if the data is characters, automatically done so)	`dat$C <- as.factor(dat$C)`
Defining a variable's type as ordinal	`dat$I <- as.ordered(dat$I)`
Defining a variable's type as numeric	`dat$X <- as.single(dat$X)`

2.4 Confirmation

Showing the structure of data	`str(dat)`
Browsing the names of variables within a data frame	`names(dat)`

3 How to enter the data

3.1 For small data

If the sample size is relatively small, the variable can be assigned values within R command line using `c()`, which means a vector. For instance, calculating mean and standard deviation of three individual's statures, 155 cm, 160 cm, and 170 cm, at first you should define a variable `dat` as follows.

```
dat <- c(155,160,170)
```

Here `dat` is a numeric vector containing 155, 160, and 170. Then, `mean(dat)` generates the mean of these three values, and `sd(dat)` generates the unbiased standard deviation of these three values. Using R, obtaining mean and sd can be summarized as follows.

```
cat("mean=",mean(dat),"sd=",sd(dat),"\n")
```

Here, `cat()` is the function to print the concatenated elements delimited with comma. The elements quoted by `""` mean character strings, others numeric. `"\n"` means carrige-return and newline.

R can easily handle matrix, which is convenient to use cross tabulation. Consider the data as following table.

Exposure	Disease	Healthy
Exposed	20	10
Unexposed	12	18

To assign the data of a variable `dat`,

```
dat <-  matrix(c(20,12,10,18),nc=2)
rownames(dat) <- c('Exposed','Unexposed')
colnames(dat) <- c('Disease','Healthy')
print(dat)
```

`matrix()` is the function to define a matrix, aligning the first element by the second element. `nc=2` means the number of columns. Here the vector of the first element is read by the order that from upper left, lower left, upper right, lower right. After definition, conducting chi-square test of independence between exposure and disease can be done by `chisq.test(dat)`.

3.2 For middle-sized data

When you handle middle-sized data, you should make the data file, separately from program (functions). MS-Excel or OpenOffice.org's calc is useful to enter such data. As a simple example, consider stature and weight data for 10 individuals as a following table.

Subject ID	Stature (cm)	Weight (kg)
1	170	70
2	172	80
3	166	72
4	170	75
5	174	55
6	199	92
7	168	80
8	183	78
9	177	87
10	185	100

The first row of the data file should be the names of variables, which should not contain spaces nor special characters, but alphabets and period (R is case-sensitive. PID and pID are treated as different variables). Here, the names of variables may be PID, HT, and WT. After you complete to enter the whole data, you should save the data file in the original format of the software you used (In MS-Excel, *.xls file). After saving, you can select "save as" menu to export the data in the text format delimited with "tab" code. If the saved text data file is `D:\DESAMPLE.TXT` (see the next screen, where [TAB] means "tab" code), R can read it to assign the data frame `dat` by `dat <- read.delim("d:/desample.txt")`. To confirm correctness of the data reading, you can simply type `dat`, or type `str(dat)` to see the structure of the data frame.

```
PID [TAB] HT [TAB] WT
1 [TAB] 170 [TAB] 70
2 [TAB] 172 [TAB] 80
3 [TAB] 166 [TAB] 72
4 [TAB] 170 [TAB] 75
5 [TAB] 174 [TAB] 55
6 [TAB] 199 [TAB] 92
7 [TAB] 168 [TAB] 80
8 [TAB] 183 [TAB] 78
9 [TAB] 177 [TAB] 87
10 [TAB] 185 [TAB] 100
```

To analyze the variables included in the data frame, you can specify the variable using `$`. In this case, you can obtain the mean and unbiased standard deviation of stature as follows.

```
cat("mean=",mean(dat$HT),"sd=",sd(dat$HT),"\n")
```

If you omit to type `dat$` every time, once you should type `attach(dat)`. Then typing `HT` is equivalent to `dat$HT` during the session until typing `detach(dat)`. An example follows.

```
attach(dat)
cor.test(HT,WT)
detach(dat)
```

3.3 For large-scale data

Usually DBMS (Microsoft Access, Oracle, postgreSQL, and so on) should be used as backbone. The form to enter the data is specially designed using DBMS itself or PHP4 and apache httpd. R can directly handle DBMS using additional libraries. However, in such cases, you should consult with some specialits of data management.

4 Stratification

4.1 Example

Consider the statures and weights of 10 subjects as follows.

Subject ID	Stature (cm)	Weight (kg)	Gender	Age
1	170	70	M	54
2	162	50	F	34
3	166	72	M	62
4	170	75	M	41
5	164	55	F	37
6	159	62	F	55
7	168	80	F	67
8	183	78	M	47
9	157	47	F	49
10	185	100	M	45

Let us name the variables as PID, HT, WT, SEX, AGE in the first row and save data as tab-delimited text file `d:\stsample.txt`.

```
dat <- read.delim("d:/stsample.txt")
```

Typing as above screen make the data assigned to the data frame `dat`.

4.2 Analyzing subdata

R can specify the subgroup to analyze by `[conditional term]` accompanied to the variable name. For example, to calculate the mean and unbiased standard deviation only for males, you can do as follows.

```
cat("mean=",mean(dat$HT[dat$SEX=='M']),"sd=",sd(dat$HT[dat$SEX=='M']),"\n")
```

In such cases, defining a function is convenient.

```
cmeansd <- function(X,C) { cat("mean=",mean(X[C]),"sd=",sd(X[C]),"\n") }
```

The definition of function `cmeansd()` above enables the shorter expression as shown below.

```
cmeansd(dat$HT,dat$SEX=='M')
cmeansd(dat$HT,dat$SEX=='F')
```

If the definition of function is as follows, the result can be reused in other functions.

```
cmeansd.noprint <- function(X,C) { list(mean=mean(X[C]),sd=sd(X[C])) }
```

The conditional term can not only be equality (==), but also unequality and other logical expressions like `is.na()`. For example, mean and unbiased standard deviation only for 40 years old or older subjects can be calculated as below.

```
cmeansd <- function(X,C) {
    cat("N=",length(X[C]),"\t mean=",mean(X[C]),"\t sd=",sd(X[C]),"\n")
}
cmeansd(dat$HT,dat$AGE>=40)
```

Combination of conditions using `&` (and) or `|` (or) is also possible.

```
cmeansd(dat$HT,(((dat$AGE>=40)&(dat$SEX=='M'))|((dat$AGE<30)&(dat$SEX=='F'))))
```

Conditions can be assigned to a logical variable as follows.

```
overforty <- (dat$AGE>=40)
cmeansd(dat$HT,overforty)
cmeansd(dat$HT,!overforty)
```

4.3 Stratified analysis

The function `tapply()` is to calculate separately for each subgroup. An example follows.

```
meansd <- function(X) { list(mean=mean(X),sd=sd(X)) }
tapply(dat$HT,dat$SEX,meansd)
```

5 A list of functions

5.1 Graphic function for a single variable

barplot	`barplot(table(C))`
normal QQ plot	`qqnorm(X)`
histogram	`hist(X)`
box and whisker plot	`boxplot(X)`

5.2 Summary of a single variable

frequency distribution	`table(C)`
minimum, Q1, median, Q3, maximum	`fivenum(X)`
sample size	`length(X)`
summation	`sum(X)`
mean	`mean(X)` (same as `sum(X)/length(X)`
unbiased variance	`var(X)` (same as `sum((X-mean(X))^2)/(length(X)-1)`)
unbiased standard deviation	`sd(X)` (same as `sqrt(var(X))`)

5.3 hypothesis testing for a single variable

Normality of distribution (Shapiro-Wilk test)	`shapiro.test(X)`
Population mean	`t.test(X,mu=population.mean)`
Population proportion	`binom.test(table(B)[2],length(B),p=population.proportion)`

5.4 Graphic functions for two variables

Mosaic plot	`mosaicplot(table(C1,C2))`
Stratified box and whisker plot	`boxplot(X~C)`

Strip chart	`stripchart(X~C)`
	`points(IX<-c(1.15,2.15),MX<-tapply(X,C,mean),pch=18)`
	`SX<-tapply(X,C,sd)`
	`arrows(IX,MX-SX,IX,MX+SX,code=3,angle=90,length=0.1)`

Scatterplot	`plot(Y~X)` (`plot(X,Y)` と同値)
Scatterplot with regression line	`plot(Y~X); abline(lm(Y~X))`

5.5 Basic calculation for two variables

Cross tabulation	`table(C1,C2)`
Pearson's product moment correlation coefficient	`cor(X,Y)`
Spearman's ordinal correlation coefficient	`cor(X,Y,method="spearman")`

5.6 Hypothesis testing for two variables

Chi-square test for independence between two categorical variables	`chisq.test(table(C1,C2))`
Fisher's exact test	`fisher.test(table(C1,C2))`
Odds ratio and its confidence intervals	`library(vcd); summary(oddsratio(table(B1,B2),log=F))`
phi coefficient and Cramer's V	`library(vcd); assoc.stats(table(C1,C2))`
Kappa coefficients	`library(vcd); Kappa(table(C1,C2))`
Cochran-Armitage test	`prop.trend.test(table(B,I)[2,],table(I))`
Test for equal variance	`var.test(X~B)`
t-test	`t.test(X~B,var.equal=T)`
Welch's test	`t.test(X~B)`
Wilcoxon's rank sum test	`wilcox.test(X~B)`
Paired t-test for the pairwise comparison of two quantitative variables	`t.test(X,Y,paired=T)`
Bartlett's test to compare variances among subgroups	`bartlett.test(X~C)`
Comparison of means among three or more subgroups: One-way ANOVA and multiple comparisons	If variances are equal, `aov(X~C)` can do one-way ANOVA, when the main effect of C is significant, `TukeyHSD(aov(X~C))` or `pairwise.t.test(X,C)`. The former execute multiple comparisons by Tukey's method and the latter by Holm's method. `library(multcomp)` enables `simtest(X~C,type="Dunnett")` (comparing with controll), and `simtest(X~C,type="Williams")` (Williams' method).
Comparison of skewed or with-outlier distribution by factor: Kruskal-Wallis test and multiple Wilcoxson's rank sum test adjusted by Holm's method	`kruskal.test(X~C)` and `pairwise.wilcox.test(X,C)`
Testing non-zero correlation for normally distributing data	`cor.test(X,Y)`
Testing non-zero correlation for skewed or with-outlier data	`cor.test(X,Y,method="spearman")`
Testing non-zero regression coefficient by linear regression	`summary(lm(Y~X))`

ANACOVA can be done by `summary(glm(Y~B+X+B:X))` and `summary(glm(Y~B+X))`. `:` makes interaction terms. Cochran-Mantel-Haenthel test can be done by `mantelhaen.test(C1,C2,C3)` or `TMP <- table(C1,C2,C3)` and `mantelhaen.test(TMP)`.