

Data explorations with R

Ken Mwai and Greg Fegan

June 1, 2015

Reading data files

Import dataset

```
> data <- read.csv("bwmal_subset.csv")
```

Reading data files

Import dataset

```
> data <- read.csv("bwmal_subset.csv")
```

- Once read in, assigning the loaded data to objects
- Datasets in R are typically stored as data frames, which have a matrix structure
- Observations are arranged as rows and variables, either numerical or categorical, are arranged as columns
- Dataset contains 8 demographic variables for 20 individuals

Reading data files

Import dataset

```
> data <- read.csv("bwmal_subset.csv")
```

- Once read in, assigning the loaded data to objects
- Datasets in R are typically stored as data frames, which have a matrix structure
- Observations are arranged as rows and variables, either numerical or categorical, are arranged as columns
- Dataset contains 8 demographic variables for 20 individuals

Get the dimension of the dataset

```
> dim(data)
```

```
[1] 21  8
```

Viewing data

Explore variable names of the dataset

```
> names(data)
```

```
[1] "X"           "matage"      "mheight"     "gestwks"     "sex"         "bweight"  
[8] "pfplacen"
```

R has ways to look at the dataset at a glance

```
> head(data) #Returns first six rows of dataset
```

	X	matage	mheight	gestwks	sex	bweight	smoke	pfplacen
1	1	26	1.575	40	0	3.11	0	0
2	2	23	1.529	40	0	2.65	0	0
3	3	18	1.540	40	1	3.41	0	0
4	4	25	1.581	40	1	2.99	0	0
5	5	25	1.555	40	1	3.16	0	0
6	6	21	1.561	40	1	2.82	0	0

Viewing data

We can access variables directly by using their names, using the object \$ variable notation

```
> data$gestwks
```

```
[1] 40 40 40 40 40 40 41 38 40 41 39 38 39 39 39 37 39 39 39 40 39
```

Check last six rows of the dataset

```
> tail(data , n=6) #Returns first six rows of dataset
```

	X	matage	mheight	gestwks	sex	bweight	smoke	pfplacen
16	195	19	1.583	37	0	2.42	0	0
17	196	20	1.534	39	1	2.93	0	0
18	197	30	1.543	39	0	2.59	1	1
19	198	38	1.602	39	0	2.48	1	0
20	199	20	1.540	40	1	3.02	1	1
21	200	24	1.503	39	0	2.79	0	1

Viewing data

To access a certain entry, we most commonly use `object[row,column]`

Single cell value

```
> data[2,3]
```

```
[1] 1.529
```

Omitting row value implies all rows; here all rows in column 3

```
> data[,3]
```

```
[1] 1.575 1.529 1.540 1.581 1.555 1.561 1.590 1.502 1.666 1.5
[13] 1.540 1.502 1.560 1.583 1.534 1.543 1.602 1.540 1.503
```

More data viewing

Omitting column values implies all columns; here all columns in row 2

```
> data[2,]
```

	X	matage	mheight	gestwks	sex	bweight	smoke	pfplacen
2	2	23	1.529	40	0	2.65	0	0

Can also use ranges - rows 2 and 3, columns 2 and 3

```
> data[2:3, 2:3]
```

	matage	mheight
2	23	1.529
3	18	1.540

Data summaries

- Enables us to see the main characteristics of data before any formal modeling or hypothesis testing
- Particular techniques depends on the type of variable: Continuous or categorical
- Continuous eg. age, height
- Categorical eg. smoking status, sex

Some data explorations: Continuous variables

```
> #some data explorations  
> mean(data$mheight)  
  
[1] 1.558571
```

Some data explorations: Continuous variables

```
> #some data explorations  
> mean(data$mheight)  
[1] 1.558571  
  
> var(data$mheight)  
[1] 0.001461357
```

Some data explorations: Continuous variables

```
> #some data explorations  
> mean(data$mheight)  
[1] 1.558571  
  
> var(data$mheight)  
[1] 0.001461357  
  
> sd(data$matage)  
[1] 5.527852  
  
> median(data$matage)  
[1] 22
```

More data explorations

Produce various summaries of continuous variable

```
> summary(data$matage) #sumarize continous variable
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
17.00	20.00	22.00	23.57	25.00	38.00

Explorations for categorical variables

Summarize single categorical variable

```
> table(data$sex)
```

```
 0  1  
11 10
```

Cross-tabulation of two categorical variables

```
> table(data$sex, data$smoke)
```

```
  0 1  
0 8 3  
1 9 1
```

Alternative cross tabulation

Using *'with'* command includes variable labels in the table

```
> with(data, table(sex,smoke))
```

```
      smoke
sex 0 1
  0 8 3
  1 9 1
```

Use R as calculator

```
> 1000-2*10^2/(8+2)  #expression to evaluate
```

```
[1] 980
```

```
> #Built-in functions:
```

```
> log(1.4)  #returns the natural logarithm of the number 1.4
```

```
[1] 0.3364722
```

```
> log10(1.4)  # returns the log to the base of 10
```

```
[1] 0.146128
```

```
> sqrt(16)  #returns the square root of 16
```

```
[1] 4
```


Calculations with assignment statements

We can store a value(s) in an R object using the assignment symbol `<-` ("less than" followed by a hyphen)

```
> x <- 2.5
```

To check what is in a variable type the variable name

```
> x
```

```
[1] 2.5
```

Can store a computation under a new R object or change the current value stored in an old object

```
> y <- 3*log(x)
```

Search and Replace

```
> x <- "Lorem Ipsum is simply dummy text of the
+ printing and typesetting industry. This is for the dummy tex
> grep("the", x)

[1] 1

> grep("the",x,value=TRUE)

[1] "Lorem Ipsum is simply dummy text of the \nprinting and ty
> #first replacement
> x2 <- sub( "the" , "THE",x)
> x2

[1] "Lorem Ipsum is simply dummy text of THE \nprinting and ty
> #All replacement
> x3 <- gsub( "the" , "THE",x)
> x3

[1] "Lorem Ipsum is simply dummy text of THE \nprinting and ty
```