Basic Data Management in R (R Instructional Worksheet)

November 12, 2017

Contents

R Packages and Libraries

The functions in R are organized in packages or libraries. The commonly used packages are preinstalled within R. However, if you need to use more specialized packages, you will need to install and then load the package before you can use functions available in that package.

Packages are installed and loaded in the Packages tab in the bottom right corner or through the dropdown menu at the top of the screen (Tools -> Install Packages).

If a package is already installed on your computer, it will be listed in this tab. To load this package so you can use it, hit the checkmark next to the name of the package.

If a package is not installed, click the 'Install' button at the top of the tab, type in the name of the package, and click 'Install'. Once it has been installed click the checkmark next to the name to load it.

Installed packages can also be loaded from the command line or within a script using the library function.

```
library(popbio)  #load previously installed package 'popbio'
```

A list of all available packages can be found here: `http://cran.r-project.org/web/packages/available_packages_by_name.html`

Importing CSV and TXT files

R can import data stored in a wide variety of file types (including Excel files), but the file type you'll probably encounter most frequently has a .csv or .txt extension. The .csv extension stands for "comma-separated values." A .txt file, on the other hand, is a simple text file created with a program like Notepad. Instead of commas, text files often use hidden characters, the tab and newline characters for example, to separate values.

A CSV file is recommended for use in R.

Using the Environment tab

1. In the Environment tab, in the top right corner click on 'Import Dataset', and then 'From Text File. . . '.
2. Choose the file that you want to import.
3. Choose a name for the data – enter into the 'Name' blank. (This name cannot contain any spaces)
4. Choose yes or no depending on whether or not the file has a Heading (column names)
5. Choose the separator, decimal, quote, and na.strings depending on your data
6. Click import

Read Commands

The following command can be typed into the console to import the data.

For a CSV file:

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

For a text file:

```
data <- read.table(“data.txt”, header = TRUE, sep=” ”)
```

The first argument is the file name (found within the working directory set earlier). The second argument, header, refers to whether or not the column names are listed in the first line of the file, TRUE is used for yes, FALSE for no. The third argument, sep, refers to what is separating the values. A CSV uses commas, but a text document could use semicolons, colons, etc. Import the chimpanzee dataset into R using either of the two methods explained above. Make sure to save it as a csv file in Excel before importing it into R. When you import it into R, name the variable containing the data 'chimp' and '. We will use it later! You can view the data after it is imported in R by using the command View, or by double clicking the file in the Environment tab. This dataset gives the population number of chimpanzees in Gombe National Park, Tanzania from 1964-1973.

Matrices and Data Frames

Matrices

Matrices are two-dimensional, for example a 4 x 5 matrix has 4 rows and 5 columns. All values within the matrix must be of the same type; i.e. all numeric or all characters

```
mymatrix <- matrix(1:4, nrow = 2, ncol = 2)
mymatrix
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

The values of a matrix are accessed using their position, like for vectors, but for matrices it is necessary to designate the row and column.

```
mymatrix[1, 2]  #the value in row 1 column 2
```

```
## [1] 3
```

```
mymatrix[2, 1]  #the value in row 2 column 1
```

```
## [1] 2
```

```
mymatrix[, 2]  #the values for all rows, column 2
```

```
## [1] 3 4
```

```
mymatrix[1, ]  # the values for row 1, and all columns
```

```
## [1] 1 3
```

```
mymatrix[1:2, ]  # the values from row 1 to 2, and all columns
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Change the value in row 1 column 2 to 0

```
mymatrix[1, 2] <- 0
mymatrix[1, 2]
```

```
## [1] 0
```

Data Frames

Data frames are the workhorses of R and are a general form of matrix. The values do not have to be of the same type, but instead one data frame can contain numbers, characters, and factors.

```
d <- c(1, 2, 3, 4)
e <- c("red", "white", "red", "blue")
f <- factor(c("high", "low", "high", "high"))
mydata <- data.frame(d, e, f)
names(mydata) <- c("ID", "Color", "Level")
mydata
```

```
##   ID Color Level
## 1  1   red  high
## 2  2 white   low
## 3  3   red  high
## 4  4  blue  high
```

The values of a data frame can be accessed in a variety of ways.

```
mydata[1]  #calles column 1
```

```
##   ID
## 1  1
## 2  2
## 3  3
## 4  4
```

```
mydata["ID"]  #another way to access column 1 - using the column name
```

```
##   ID
## 1  1
## 2  2
## 3  3
## 4  4
```

```
mydata$ID  #last way to access column 1 - also using the column name [1] 1 2 3 4
```

```
## [1] 1 2 3 4
```

```
mydata[1, ]  #access the first row
```

```
##   ID Color Level
## 1  1   red  high
```

Useful Functions for Data Sets

```
str(Loblolly)  # structure of the dataframe
```

```
## Classes ’nfnGroupedData’, ’nfGroupedData’, ’groupedData’ and ’data.frame’:   84 obs. of  3 variables:
##  $ height: num  4.51 10.89 28.72 41.74 52.7 ...
##  $ age   : num  3 5 10 15 20 25 3 5 10 15 ...
##  $ Seed  : Ord.factor w/ 14 levels "329"<"327"<"325"<..: 10 10 10 10 10 10 13 13 13 13 ...
##  - attr(*, "formula")=Class ’formula’  language height ~ age | Seed
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##  - attr(*, "labels")=List of 2
##   ..$ x: chr "Age of tree"
##   ..$ y: chr "Height of tree"
##  - attr(*, "units")=List of 2
##   ..$ x: chr "(yr)"
##   ..$ y: chr "(ft)"
```

```
Loblolly  # displays all the data
head(Loblolly)  # displays first 6 rows of data
tail(Loblolly)  # displays last 6 rows of data
```

```
nrow(Loblolly)  # number of rows
```

```
## [1] 84
```

```
ncol(iris)  # number of columns
```

```
## [1] 5
```

```
names(Loblolly)  # column names
```

```
## [1] "height" "age"    "Seed"
```

```
length(Loblolly$height)  # number of data points in that column
```

```
## [1] 84
```

R has many datasets built into the program that you can use for practicing. Type data() into the console to see a list of all of the available datasets. We will use the Loblolly dataset. If you type help(Loblolly) you can read all about the data included in this dataset.

#Subsetting Data
Sometimes if we have a large dataset for multiple categories, we might want to subset it into smaller datasets for analysis.

For example, lets look at the Loblolly dataset, which gives us information on the height and age of loblolly pine trees from different seeds.

We can create separate datasets for each of the seeds.

```
seed301 <- subset(Loblolly, Loblolly$Seed == "301")
# This creates a new dataframe called
# 'seed301' which contains all columns of data
# for just the seed numbered 301
seed303 <- subset(Loblolly, Loblolly$Seed == "303")
# This creates a new dataframe called
# 'seed303' which contains all columns of data
# for just the seed numbered 303
```

We can also subset based on criteria - in this case such as age or height.

```
Loblolly2 <- subset(Loblolly, Loblolly$age < 5)
# This dataframe only contains the data on
# Loblolly treess at less than 5 years of age
# (only the trees at 3 years of age)
```

Merging Data Frames

Sometimes you need to merge two dataframes, using a variable (column) that is shared between them. In this example, we first create two dataframes from a couple Gapminder .csv files.

As it is easier to load data into R when it is in .csv format, I download the Gapminder Excel files, open them, and then 'save as' to a .csv format. With these two files, I also renamed the first column to 'country' as Gapminder used a different name.

Set the working directory as the files are staged there.

```
setwd("c:/informatics")
```

Load the two Gapminder .csv files into dataframes. This is our raw data…

```
child_deaths <- read.csv(file = "childDeaths.csv",
    header = TRUE, stringsAsFactors = FALSE)
income_person <- read.csv(file = "incomePerson.csv",
    header = TRUE, stringsAsFactors = FALSE)
```

Create two new dataframes with just the country and 2007 (death & income) variables. (The original dataframes contain a lot of additional information we don't need.) Note: R appends an 'X' to numeric column names so the 2007 column in both files loads as 'X2007'. Naming restrictions apply to variable (column) names – hence, the reason for this behavior.

```
deaths_2007 <- data.frame(child_deaths$country,
    child_deaths$X2007, stringsAsFactors = FALSE)
income_2007 <- data.frame(income_person$country,
    income_person$X2007, stringsAsFactors = FALSE)
```

Assign names to our columns. We'll join the dataframes by country so these two variables must have the exact same name and case. (I always use lowercase for variable names.)

```
colnames(deaths_2007) <- c("country", "deaths_2007")
colnames(income_2007) <- c("country", "income_2007")
```

And finally, combine the two dataframes using the *merge()* function.

```
income_deaths <- merge(deaths_2007, income_2007,
    by = "country")
```

PROBLEMS

19. Import the 'lionCrater' file into R and save the dataframe as "Lion". This file gives the population numbers for lions in the Ngorongoro Crater in Tanzania.

20. Import the 'blackRhinoCrater' file into R and save the dataframe as "Rhino". This file gives the population numbers for black rhinos in the Ngorongoro Crater in Tanzania.

21. Display the structure of the Lion data frame. What is included? How many observations and how many variables? What are the variable names?

22. What are three ways to display the data in column 1 of the Lion data frame?

23. How do you access the 5*th* row of data in the Lion data frame?

24. Display the first 6 rows of the Rhino data frame.

25. Display the last 6 rows of the Rhino data frame.

26. How many rows are in the Rhino data frame?

27. How many columns are in the Rhino data frame?

28. Look at the Year column for both Lions and Rhinos. Which years overlap? Create two new data frames Lion2 and Rhino2 that only contain the overlapping years.

29. Create a new data frame called Lion3 that only contains the years when the population is over 50 individuals. How many rows are in this new data frame?

30. Create a new data frame called Rhino3 that only contains the years when the population of rhinos is less than 12 individuals. How many rows are in this new data frame?