Lab 3: Reading Data Tables

Data Science UTB

Enrique J. De La Hoz D.

Learning Objectives

- Importing Data Tables in R
- base reading tables functions
- R package `"readr"`
- R package `"foreign"`

Importing Data Tables

Data sets come in many different presentations. One common format is that of a data table—like a spreadsheet—although you can find data in other formats (especially when the data is in its raw version). However, once you get to the analysis stage, you will likely be dealing with some sort of tabular format.

Because data tables are so ubiquituous, it is fundamental that you learn how to import them in R. In the first part of today's lab, we are going to review various aspects that have to do with reading in tables in R.

We will talk about the following functions (and packages)

- `read.table()`
- `read.csv()`
- `read.delim()`
- `read.fwf()`
- R package "readr"
- `readHTMLTable()` from package "XML"
- Packages to read data sets from other software programs

Abalone Data Set

The first data set to consider is the **Abalone Data Set** that is part of the UCI Machine Learning Repository

The location of the data file is:

http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data

The location of the data dictionary (description of the data) is:

http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.names

Look at both the dataset file, and the file with its description, and answer the following questions:

- What's the character delimiter?
- Is there a row for column names?
- Are there any missing values?
- What are the data types of each column?

One basic way to read this file in R, is by passing the url location of the file directly to any of the `read.table()` functions:

```r
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
abalone <- read.table(url, sep = ",")
```

Getting a Local Copy of the Data

My suggestion when reading datasets from the Web, is to always try to get a local copy of the data file in your machine (as long as you have enough free space to save it in your computer). To do this, you can use the function `download.file()` and specify the url address, and the name of the file that will be created in your computer. For instance, to save the abalone data file in **your working directory**, type the following commands:

```r
# download copy
origin <- 'http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data'
destination <- 'abalone.data'
download.file(origin, destination)
```

Now that you have a local copy of the dataset, you can read it with `read.table()` like so:

```r
# reading data from your working directory
abalone <- read.table("abalone.data", sep = ",")
```

Once you read a data table, you may want to start looking at its contents, usually taking a peek at a few rows. This can be done with `head()` and/or with `tail()`:

```r
# take a peek of first rows
head(abalone)

# take a peek of last rows
tail(abalone)
```

Likewsie, you may also want to examine how R has decided to take care of the storage details (what data type is used for each column?). Use the function `str()` to check the structure of the data frame:

```r
# check data frame's structure
str(abalone, vec.len = 1)
```

Detailed information about the columns

So far we have been able to read the data file in R. But we are missing a few things. First, we don't have names for the columns. Second, it would be nice if we could specify the data types of each column instead of letting R guess how to handle each data type.

According to the description of the Abalone data set, the columns represent these variables:

Name	Data Type
Sex	nominal
Length	continuous
Diameter	continuous
Height	continuous
Whole weight	continuous
Shucked weight	continuous
Viscera weight	continuous
Shell weight	continuous
Rings	integer

Let’s create a vector of columns names, and another vector of data types:

```
col_names <- c(
    'sex',
    'length',
    'diameter',
    'height',
    'whole_weight',
    'shucked_weight',
    'viscera_weight',
    'shell_weight',
    'rings'
)

col_types <- c(
    'factor',
    'numeric',
    'numeric',
    'numeric',
    'numeric',
    'numeric',
    'numeric',
    'numeric',
    'integer'
)
```

Now we can re-read the table in a more complete (and usually more efficient) way:

```
abalone <- read.table(
    'abalone.data',
    col.names = col_names,
    colClasses = col_types,
    sep = ","
)

# check its structure again
str(abalone, vec.len = 1)
```

Your turn

- Read the Abalone data with the `read.csv()` function
- Look at the data description

http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.names and confirm the following statistics:

```
       Length Diam  Height  Whole  Shucked  Viscera    Shell    Rings
Min     0.075  0.055 0.000   0.002    0.001    0.001    0.002        1
Max     0.815  0.650 1.130   2.826    1.488    0.760    1.005       29
Mean    0.524  0.408 0.140   0.829    0.359    0.181    0.239    9.934
SD      0.120  0.099 0.042   0.490    0.222    0.110    0.139    3.224
```

Basic Plots

As you can tell, the Abalone data contains 9 variables. To start exploring the content, we begin by producing charts for each single variable, focused on looking at their distributions:

- Quantitative variables: histogram, boxplot
- Qualitative variables: barchart, piechart

The workhorse plotting function in R is `plot()`. This function is actually a *method*, meaning that it behaves differently depending on the type of input.

If the provided input is a factor, `plot()` will generate a barchart.

```r
# plot of a factor
plot(abalone$sex)
```

Alternatively, you can always create a frequency table first—via `table()`— and then plot a barchart with `barplot()`

```r
table_sex <- table(abalone$sex)
barplot(table_sex)
```

For a quantitative variable, the typical graphics to examine the distribution are histograms (`hist()`) and boxplots (`boxplot()`)

```r
hist(abalone$diameter)
boxplot(abalone$diameter, horizontal = TRUE)
```

Pittsburgh Bridges Data Set

The data set is part of the UCI Machine Learning Repository:

http://archive.ics.uci.edu/ml/datasets/Pittsburgh+Bridges

The data Description is here:

http://archive.ics.uci.edu/ml/machine-learning-databases/bridges/bridges.names

The Data file is here:

http://archive.ics.uci.edu/ml/machine-learning-databases/bridges/bridges.data.version1

Read the description, and take a look at the data set:

- Are there column names?
- What is the field separator?
- Are there any missing values?
- What is the character for missing values (if any)?
- What is the data type of each variable (i.e. column)?
- Download a copy of the data to your computer (use `download.file()`) and save it in a file named `bridges.data`

```r
# download a copy of the data file
```

Reading the Data

- Create a vector of column names
- Create a vector of column types

- Use the function `read.table()` to read the data. Name it `bridges`.

```
# vector of column names


# vector of column types


# reading the data with 'read.table()'


# reading the data with 'read.csv()'
```

Using `read.csv()`

Now use the function `read.csv()` to re-read the bridges data

```
# your code
```

Basic Inspection

Use functions to start examining the `bridges` data frame:

- `str()`
- `summary()`
- `head()` and `tail()`
- `dim()`
- `names()`
- `colnames()`
- `nrow()`
- `ncol()`

```
# your code
```

Research Questions

Write R code to find:

- Year of the oldest bridge
- Year of the most recent erected bridge
- Frequency of bridges by purpose
- Frequency of materials
- Average length of the bridges
- Plot a timeline: year -vs- length

```
# your code
```
