Northeastern University

CS6020: Collecting, Storing, and Retrieving Information

Programming in R

Programming in R

BASIC DATA INPUT & OUTPUT

Lesson Objectives

- After completing this lesson, you are able to:
  - load data from text files
  - save data to text files

Reading and Writing Data

- The ability to read and write external text files is an essential part of data processing.
- Many data sets are stored in simple text files.
- Excel and other programs can export and import text files in certain formats.

Generating a Simple Data File

- Load the built-in data set *AirPassengers* containing monthly international airline passenger data between 1949 and 1960.
- After displaying the data set, copy the data into a simple text file.

```
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 …
```

Setting the Working Directory

- To find the current working directory, i.e., where R looks for external files, use `getwd()`.
- Navigate to the directory (folder) where the data file is located using the function `setwd()`:

```
> getwd()
[1] "C:/Users/Martin/Documents"
> setwd("C:/Users/Martin/Downloads")
```

Reading Files

- While R has several functions for reading files, the most commonly used function for reading text files is `read.table()`.

```
> setwd("C:/Users/Martin/Downloads")
> ap <- read.table("AirPassengers.txt",header=TRUE,sep="")
> ap
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 …
```

Skipping Input

- The `read.table()` function has a `skip=x` parameter which allows you to skip some number of lines.

```
> ap <- read.table("AirPassengers.txt",skip=4,header=TRUE,sep="")
> ap
  X1952 X171 X180 X193 X181 X183 X218 X230 X242 X209 X191 X172 X194
1  1953  196  196  236  235  229  243  264  272  237  211  180  201
2  1954  204  188 …
```

- Note that this also skips the header.

Reading a Select Number of Rows

- The `nrows=x` argument allows you to specify how many lines (rows) of input should be read from the file.
- This can be combined with `skip`.

```
> ap <- read.table("AirPassengers.txt",skip=4,nrows=3)
> ap
    V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13
1 1952 171 180 193 181 183 218 230 242 209 191 172 194
2 1953 196 196 236 235 229 243 264 272 237 211 180 201
3 1954 204 188 235 227 234 264 302 293 259 229 203 229
>
```

Adding New Columns

- New columns can be added to a data set using the `cbind()` function.

```
> ap$Total <- cbind(rowSums(ap))
> ap
    V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 Total
1 1952 171 180 193 181 183 218 230 242 209 191 172 194  4316
2 1953 196 196 236 235 229 243 264 272 237 211 180 201  4653
3 1954 204 188 235 227 234 264 302 293 259 229 203 229  4821
>
```

Writing Data

- A data object can be exported to a file using the `write.table()` function.

```
> getwd()
[1] "C:/Users/Martin/Downloads"
> write.table(ap,"AirPassNG.txt",col.names=NA,
              row.names=TRUE,quote=FALSE,sep=",")
```

- Note that R requires the use of a forward slash (‘/’) to separate directories (folders) not the back slash (‘\’) used by Windows.

File Compression

- Text files can often be very large and thus increase storage space and transmission time.
- The most common file compression standards are:
  - ZIP: fast, widely supported
  - RAR: better compression and encryption
  - 7Z: volume spanning, better encryption
- These standards also offer encryption.

Summary

- In this lesson, you learned that:
  - data input and output is an essential part of data processing
  - the `read.table()` function reads a columnar text file with a specified separator
  - the `write.table()` function exports a data set to a text file with a user specified separator
  - text files are often compressed to save storage and transmission time

Summary, Review, & Questions…