REVOLUTION ANALYTICS

Importing data into R

Revolution Analytics

1 Importing data from other sources

2 Exporting data in ASCII format

Overview

In this session you get started with importing and exporting data.

- Familiarize with common R data formats
- Explore use of low- and high-level R functions for importing and exporting data
- Introduce remote database access and use of SQL syntax from within R environment.

Directory Setup

```
dataPath <- "../data"
outdir <- "../output"
if (!file.exists(outdir)) dir.create(outdir)
```

Exercise

- Set your course materials `R` directory as your working directory.
- Write the modified cars data set to file as a `.Rdata object` and re-load that object in your workspace.
- Save your entire workspace to file, and re-load that workspace in the RPE.

Outline

1. Importing data from other sources
2. Exporting data in ASCII format

Importing data from other sources

The function `scan()` is a low-level function that reads data into the workspace from either the keyboard or file.

This function may be more efficient for large data sets than higher-level import functions.

See

```
help(scan)
```

for information on default settings.

`read.table()` and `read.csv()`

The function `read.csv()` allows you to easily read data in comma delimited (csv) format, including directly from a web URL:

```
url <- file.path(dataPath, "google_stock_data.csv")
GOOG.data <- read.csv(url)
str(GOOG.data)
```

```
## 'data.frame':    923 obs. of  7 variables:
## $ Date    : Factor w/ 923 levels "2007-07-09","2007-07-10",..: 923 922 921 920 919 918 917 916 915
##  $ Open      : num  608 606 600 618 610 ...
##  $ High      : num  609 611 606 619 616 ...
##  $ Low       : num  600 605 595 599 608 ...
##  $ Close     : num  601 610 601 601 613 ...
## $ Volume  : int  3011000 1945300 2026700 3323200 2281500 1932400 2711700 2889600 3639900 3217900 .
##  $ Adj.Close: num  601 610 601 601 613 ...
```

Other Helpful Import Functions

read.table() Read a file in table format (most flexible)

read.csv2() Read alternate version of csv (, treated as decimal place; : as separator)

read.fwf() Read fixed-width fields

read.delim() Read delimited fields (default: tab-delimited)

Taking low level control with `scan()`

The function `scan()` allows you to take line-by-line control of the read process. This is somewhat more complicated to use but may be faster:

```
GOOG.data <- scan(url, skip = 1, sep = ",", what = list(Date = "",
  Open = 0, High = 0, Low = 0, Close = 0, Volume = 0, Adj.Close = 0))
GOOG.data <- as.data.frame(GOOG.data)
```

Importing data from other sources

Scanning data from compressed files with `scan` is the most efficient way to import data (though less user friendly).

Compressed files take less space on your hard drive and sometimes read 3x more quickly than uncompressed files (b/c hard drive access is slow)

Outline

1. Importing data from other sources
2. Exporting data in ASCII format

The function `write.table()`

Use `write.table()` or `write.csv()` to export matrices and dataframes:

```
write.table(GOOG.data, file = file.path(outdir, "google_data.txt"))
```

REVOLUTION ANALYTICS

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR