# Reading in Data Files

ACTIVITY 13

One of the first things you need to learn to do with R is bring data in for analysis. This is often the first of lines in our R scripts. R can work with many different data formats (CSV, TXT, XLS, google spreadsheet, SAS, SPSS) through the base R or packages that others have written. In this class, we will work with two different data files. In most of the class, we will interface directly with the class database through an ODBC connection (on Windows boxes). It is common, though, to have to read in data files from a format called CSV, or comma separated values. From the class database, we selected records with this query:

```
SELECT * FROM trimet.route19 WHERE service_day >= '03-04-2007'
AND service_day < '03-10-2007'
```

and exported it to a .csv file. We've placed this .csv file on the companion website for easy reading.

## Purpose

The purpose of this activity is to help you become familiar with reading in external data frames for analysis in R.

## Learning Objective

Read in data files from CSV, Excel, and direct connection to PostgreSQL.

## Required Resources

- R, R Studio
- Sample script file from class web site

## Time Allocated

15 minutes in class

## Tasks

*This activity begins assuming R Studio is open. If not, please start R Studio.*
*In R Studio, start a new script file.*

It is always good practice to use comments throughout your code. One good tip is to create a file header such as the following:

```
#######################################################
###     Activity 13  - Reading in Data Files        ###
###                                                 ###
###     Understanding and Communicating             ###
###     Multimodal Transportation Data              ###
###                                                 ###
###     Script: C. Monsere                          ###
###     Revision: 03.22.2012                        ###
#######################################################

#-------------------------------------------------------
# 1 - Read in the CSV file and explore
#-------------------------------------------------------
```

### A. Loading the CSV file

We are going to read this rather large CSV file into R. The function to do this is `read.csv`. But first, at the R prompt, type:

```
? read.csv or help(read.csv)
```

Take some time to browse the options. The following command will read in the CSV file, assuming that you set the working directory properly. Make sure you are familiar with the options.

```
trimet <- read.csv("http://www.transportation-data.com/resources/
route19march07.csv", header=TRUE, sep=",", na.strings="NA", dec=".",
strip.white=TRUE)
```

Use the `str()` function to see what R is interpreting the structure to be of the data frame *trimet*. This can be very valuable in the future. This tells you how R is reading each column. You can also view the data elements in R Studio by clicking on the name *trimet* in the data window. A spreadsheet-like display will appear in the source pane (Figure 14). Notice that within the workspace pane R Studio gives you a helpful summary: 29,928 obs. of 26 variables. This means there are 29,928 rows and 26 columns of data.

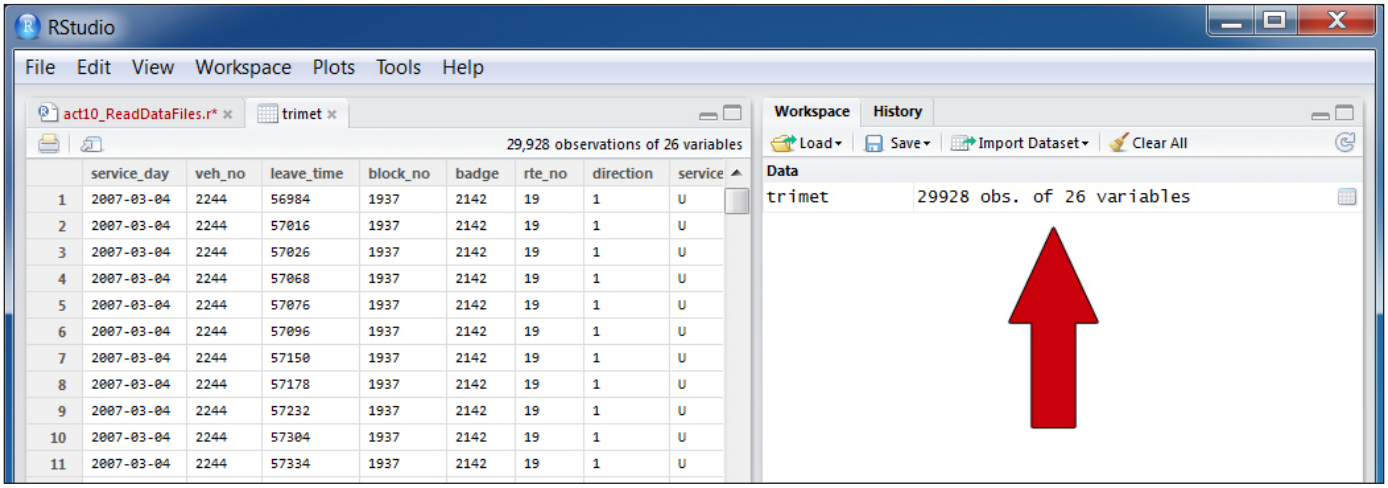


**Figure 14** Screen capture of table view of data set in RStudio

There a number of R functions that give you this same information:

How many variables (columns) in the data frame?

```
ncol(dataframe)
```

How many records in the data frame?

```
nrow(dataframe)
```

What data type does R seem to think the *service_day* column is?

```
str(trimet$service_day)
```

What are the levels of service day?

```
levels(trimet$service_day)
```

How many elements in the service day column?

```
length (trimet$service_day)
```

### B. Count Functions

These count functions can be helpful in programming the analysis of the data frame. In this data, the rows are structured such that each row is a stop event. Just knowing the number of rows tells you how

many stops were observed. Another useful function is one that returns unique values in a column. To answer the question, how many unique stops are represented in the data frame, you could do this with:

```
numuniqstops <- unique(trimet$stop_id)
length (numuniqstops)
```

1. Can you adapt the above code to answer how many types of schedule status flags appear in the data frame?

Now, can you create a new data frame containing only records that occurred at `stop_id = 2107`?

2. How many records in the subsetted data frame?
3. What is the total amount of dwell time in the subsetted data? Hint, use the function `sum`
4. How many different unique vehicles were used in this data set to provide service?

### C. Reading in a File from Your Local Directory

If you have not done so already, create a directory where you will be saving files from this class. Download the CSV file from the web to this directory:

***http://www.transportation-data.com/resources/route19march07.csv***

One helpful step that I take in R is to set the working directory. Setting the working directory is not required. If you set this, it becomes the default directory. The command is:

```
setwd ("N:/My Documents/CE510")
```

This will be the default path for the R session until you change it. Note that I have set this to be a folder in my N: drive. This is now the root path for all R operations (reading, saving, etc). It is critical that you understand that R requires that the normal convention for windows backslashes \ to be reversed / . If you don't set this, you will need to specify the full path name to the file. In this example, the working directory is set to **N:/My Documents/CE510**. The CSV is copied from the web

```
setwd ("N:/My Documents/CE510")
trimet_local <- read.csv("route19march07.csv", header=TRUE, sep=",",
na.strings="NA", dec=".", strip.white=TRUE)
```

You can also use the R Studio option to ***Import Data Set*** under the Workspace menu. This has a nice GUI but has the disadvantage of requiring you to do the point-click every time you want to load the file, rather than just running the script.

### D. Reading in Excel File Using RODBC

Open the CSV and save as XLS (**not** XLSX). Go ahead and change the name of the tab to something other than the file name. In my example, I changed the name of the tab to *data*.

Now install the package RODBC from the R Studio menu options. Once a package is installed, to load it in an active session of R you need to call it. Add the package to your R session using the command:

```
library(RODBC)
```

RODBC is a windows platform that allows you to interact with set up a connection to XLS. Since the file is in our working directory, only the file name is needed (not the entire path).

```
xls_connect <- odbcConnectExcel("route19march07.xls")
```

Now call the `odbcConnect` function that we established previously and tell R you want data from the tab named what you called it in the first step.

```
trimet_xls <- sqlFetch(xls_connect, "data")
```

Check this data using `str()`.

## Deliverable

An R script file showing the work in this activity. Be sure to add comments and headers to your file.

## Assessment

Annotated code – submit your R script to the Dropbox. To receive all 10 points, the code must contain comments and the answers to the 4 questions.

**Activity 12 Grading Rubric**

| | Excellent (10) | Good (8) | Poor (6) | 0 |
|---|---|---|---|---|
| **Answers** | 0 Incorrect | 2 or less Incorrect | >2 Incorrect | Code does not execute |
| **Script** | Organized, complete, accurate and executes. | Missing minor parts, but executes and is otherwise organized and accurate. | Missing significant portions of the activity, unorganized, inaccurate, but executes. | Code does not execute |
| **Annotation** | Annotations are complete and describe what the code is accomplishing. | Some annotations are incomplete or do not describe what the code is accomplishing. | No annotations were provided. | Code does not execute |