Commerce 693 Keith Head's Session

Rajesh Vijayaraghavan

2018-09-13

Introduction

This markdown document shows basic data analysis in R. The PDF file is generated using the R markdown code.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Loading Packages

These are the packages needed for executing the R code below. Depending on the machine, some of these packages may not have been installed. First step is to install the packages below. The next chunk runs code to check and install packages needed.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(DBI)
library(dbplyr)
```

```
##
## Attaching package: 'dbplyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     ident, sql
```

```
library(RSQLite)
```

Install Packages Relevant to Run this Program

```
#specify the packages of interest
packages = c("readr", "RSQLite","dbplyr","dplyr","DBI")
# https://gist.github.com/smithdanielle/9913897
# http://www.vikram-baliga.com/s/packagecheck.R
#use this function to check if each package is on the local machine
#if a package is installed, it will be loaded
#if any are not, the missing package(s) will be installed and loaded
package.check <- lapply(packages, FUN = function(x) {
  if (!require(x, character.only = TRUE)) {
    install.packages(x, dependencies = TRUE)
    library(x, character.only = TRUE)
  }
})

#verify they are loaded
search()
```

```
##  [1] ".GlobalEnv"        "package:RSQLite"   "package:dbplyr"
##  [4] "package:DBI"       "package:readr"     "package:dplyr"
##  [7] "package:stats"     "package:graphics"  "package:grDevices"
## [10] "package:utils"     "package:datasets"  "package:methods"
## [13] "Autoloads"         "package:base"
```

Reading the Input Files

The csv files are downloaded from Compustat from WRDS.

```
DIR_NAME <- "/Users/rajeshvijayaraghavan/Dropbox/UBCCourseWork/DoctoralSeminars/WinterTerm2018_KeithHea
FILE_1 <- "wrds_compustat_file1.csv"
FILE_2 <- "wrds_compustat_file2.csv"
# Change the DIR_NAME above to your own local directory, where the two files are stored.

# Read FILE_1 into data frame df.file1
INPUT_FILE <- paste0(DIR_NAME,FILE_1)
df.file1 <- read_csv(file=INPUT_FILE,col_names = TRUE)
```

```
## Parsed with column specification:
## cols(
##   gvkey = col_character(),
##   datadate = col_integer(),
##   fyear = col_integer(),
##   indfmt = col_character(),
##   consol = col_character(),
##   popsrc = col_character(),
##   datafmt = col_character(),
##   tic = col_character(),
##   conm = col_character(),
##   curcd = col_character(),
##   fyr = col_integer(),
##   at = col_double(),
##   cik = col_character(),
```

```
##   costat = col_character()
## )
```

```
# Read FILE_2 into data frame df.file2
INPUT_FILE <- paste0(DIR_NAME,FILE_2)
df.file2 <- read_csv(file=INPUT_FILE,col_names = TRUE)
```

```
## Parsed with column specification:
## cols(
##   gvkey = col_character(),
##   datadate = col_integer(),
##   fyear = col_integer(),
##   indfmt = col_character(),
##   consol = col_character(),
##   popsrc = col_character(),
##   datafmt = col_character(),
##   tic = col_character(),
##   conm = col_character(),
##   curcd = col_character(),
##   fyr = col_integer(),
##   act = col_double(),
##   lct = col_double(),
##   cik = col_character(),
##   costat = col_character()
## )
```

To check the First Five Rows

Returns the first five rows of the data frames df.file1 and df.file2.

```
head(df.file1)
```

```
## # A tibble: 6 x 14
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 4 more variables: fyr <int>, at <dbl>, cik <chr>, costat <chr>
```

```
head(df.file2)
```

```
## # A tibble: 6 x 15
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 5 more variables: fyr <int>, act <dbl>, lct <dbl>, cik <chr>,
```

```
## #   costat <chr>
```

To get the number of rows and columns in the dataframe

```
dim(df.file1)
```

```
## [1] 373635     14
```

```
dim(df.file2)
```

```
## [1] 373635     15
```

Rename the Variable Names

Suppose we want to rename variable names at in df.file1, and act, lct in df.file2. They refer to assets, current assets, and current liabilities.

```
df.file1 <- dplyr::rename(df.file1, total_assets = at) # renames at as total_assets
df.file2 <- dplyr::rename(df.file2, current_assets = act, current_liabilities = lct)
```

Confirm if the Variable Rename works

```
head(df.file1)
```

```
## # A tibble: 6 x 14
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 4 more variables: fyr <int>, total_assets <dbl>, cik <chr>,
## #   costat <chr>
```

```
head(df.file2)
```

```
## # A tibble: 6 x 15
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 5 more variables: fyr <int>, current_assets <dbl>,
## #   current_liabilities <dbl>, cik <chr>, costat <chr>
```

Create a new variable

Create a new variable log_total_assets in df.file1, and total_current_assets in df.file2

```
df.file1 <- mutate(df.file1, log_total_assets = log(total_assets))
df.file2 <- mutate(df.file2, total_current_assets = current_assets + current_liabilities)
```

SQL

Let us use SQLite, which is inbuilt with RStudio. SQLite is powerful relational database management system (database). Very popular – and even the iPhone runs this database. No specific installation required to integrate with R/RStudio.

Other powerful databases: https://www.postgresql.org/

Data are stored in tables, which are placed in databases. Two steps involved when storing data in databases. 1) Creating a database. 2) Each databases have multiple tables where the data reside.

In the example below, a database "comm693-db.sqlite" is created. This database houses two tables table1 and table2.

```
mydb <- dbConnect(RSQLite::SQLite(), "comm693-db.sqlite") # Creates the database.
mydb ## returns TRUE stating the DB is created, if it does not exist already.
```

```
## <SQLiteConnection>
##   Path: /Users/rajeshvijayaraghavan/Dropbox/UBCCourseWork/DoctoralSeminars/WinterTerm2018_KeithHeadC
##   Extensions: TRUE
```

```
# mydb is the connection which can be used to refer to the database to work with tables. In other words


prtime <- proc.time() # Push df.file1 in to the SQL Database
rs <- dbWriteTable(mydb, "table1", as.data.frame(df.file1), row.names=FALSE)
proc.time() - prtime
```

```
##    user  system elapsed
##   0.698   0.018   0.723
```

```
dbGetQuery(mydb,"CREATE INDEX index_gvkey_db1 ON table1 (gvkey,fyear)")
```

```
## Warning in result_fetch(res@ptr, n = n): Don't need to call dbFetch() for
## statements, only for queries
```

```
## data frame with 0 columns and 0 rows
```

```
bank_db <- tbl(mydb, "table1") ## You are calling the version of the data from the SQL table
dim(bank_db)
```

```
## [1] NA 15
```

```
colnames(bank_db)
```

```
##  [1] "gvkey"            "datadate"         "fyear"
##  [4] "indfmt"           "consol"           "popsrc"
##  [7] "datafmt"          "tic"              "conm"
## [10] "curcd"            "fyr"              "total_assets"
## [13] "cik"              "costat"           "log_total_assets"
```

```
head(bank_db)
```

```
## # Source:   lazy query [?? x 15]
## # Database: sqlite 3.22.0
## #   [/Users/rajeshvijayaraghavan/Dropbox/UBCCourseWork/DoctoralSeminars/WinterTerm2018_KeithHeadClass
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 5 more variables: fyr <int>, total_assets <dbl>, cik <chr>,
## #   costat <chr>, log_total_assets <dbl>
```

```
# Push df.file2 in to the SQL table
rs <- dbWriteTable(mydb, "table2", as.data.frame(df.file2), row.names=FALSE)
proc.time() - prtime
```

```
##    user  system elapsed
##   1.623   0.063   1.706
```

```
dbGetQuery(mydb,"CREATE INDEX index_gvkey_db2 ON table2 (gvkey,fyear)")
```

```
## Warning in result_fetch(res@ptr, n = n): Don't need to call dbFetch() for
## statements, only for queries
```

```
## data frame with 0 columns and 0 rows
```

```
bank_db <- tbl(mydb, "table2") ## You are calling the version of the data from the SQL table
dim(bank_db)
```

```
## [1] NA 16
```

```
colnames(bank_db)
```

```
##  [1] "gvkey"               "datadate"            "fyear"
##  [4] "indfmt"              "consol"              "popsrc"
##  [7] "datafmt"             "tic"                 "conm"
## [10] "curcd"               "fyr"                 "current_assets"
## [13] "current_liabilities" "cik"                 "costat"
## [16] "total_current_assets"
```

```
head(bank_db)
```

```
## # Source:   lazy query [?? x 16]
## # Database: sqlite 3.22.0
## #   [/Users/rajeshvijayaraghavan/Dropbox/UBCCourseWork/DoctoralSeminars/WinterTerm2018_KeithHeadClass
##   gvkey  datadate fyear indfmt consol popsrc datafmt tic   conm      curcd
##   <chr>     <int> <int> <chr>  <chr>  <chr>  <chr>   <chr> <chr>     <chr>
## 1 001003 19880131  1987 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 2 001003 19890131  1988 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 3 001003 19900131  1989 INDL   C      D      STD     ANTQ  A.A. IMP~ USD
## 4 001004 19880531  1987 INDL   C      D      STD     AIR   AAR CORP  USD
## 5 001004 19890531  1988 INDL   C      D      STD     AIR   AAR CORP  USD
## 6 001004 19900531  1989 INDL   C      D      STD     AIR   AAR CORP  USD
## # ... with 6 more variables: fyr <int>, current_assets <dbl>,
```

```
## #   current_liabilities <dbl>, cik <chr>, costat <chr>,
## #   total_current_assets <dbl>
```

Finding the unique number of rows in the files

This process can be done either from R or the SQL Database.

```
# The number of distinct rows from R.
## For file1
prtime <- proc.time()
distinct.df1 <- distinct(df.file1)
proc.time() - prtime
```

```
##    user  system elapsed
##   0.296   0.015   0.315
```

```
# The number of distinct rows from R.
## For file2
prtime <- proc.time()
distinct.df2 <- distinct(df.file2)
proc.time() - prtime
```

```
##    user  system elapsed
##   0.366   0.018   0.386
```

```
# The number of distinct rows from SQLite
## For table1
prtime <- proc.time()  # starts the clock
bank_db <- tbl(mydb, "table1") # call the table table1 into R
distinct.mydb.df1 <- distinct(bank_db) # run the number of distinct rows from the SQL
# distinct.gvkey.df1 <- collect(distinct.mydb.df1) # Run this to get the data from the SQL to R, if nee
proc.time() - prtime # for the time that is elapsed
```

```
##    user  system elapsed
##   0.004   0.000   0.004
```

```
## For table2
# Repeat running this for file 2.
prtime <- proc.time()
bank_db <- tbl(mydb, "table2")
distinct.mydb.df2 <- distinct(bank_db)  # Run this to get the data from the SQL to R, if needed
# distinct.gvkey.df2 <- collect(distinct.mydb.df2)
proc.time() - prtime
```

```
##    user  system elapsed
##   0.003   0.000   0.003
```

Removing the tables from the database.

This code is for your reference. Exercise caution when running in real program.

```
rs <- dbRemoveTable(mydb, "table1")
rs <- dbRemoveTable(mydb, "table2")
```

Joining the data frames

Suppose the final data analysis involve combining the two dataframes, it is required to join the two data frames. The steps that are needed are the following.

We will perform the analysis in R, although similar joins can be run in SQL.

- First let us get the reliable data records and remove duplicates (for various reasons the source data has duplicates)
- Refer to https://wrds-www.wharton.upenn.edu/pages/support/research-wrds/sample-programs/annual-data-extract-compustat-north-america/ A screen of if indfmt='INDL' and datafmt='STD' and popsrc='D' and consol='C'; is applied to extract the most reliable, standardized data records and to remove potential duplicate records.

```
df.file1.subset <- filter(df.file1, indfmt == "INDL" & datafmt == "STD" & popsrc=="D" & consol=="C")
df.file2.subset <- filter(df.file2, indfmt == "INDL" & datafmt == "STD" & popsrc=="D" & consol=="C")


# Select a subset of columns from df.file1
df.file1.subset <- select(df.file1.subset, gvkey, fyear, total_assets, log_total_assets)
df.file1.subset <- distinct(df.file1.subset)

# Select a subset of columns from df.file2
df.file2.subset <- select(df.file2.subset, gvkey, fyear, current_assets, current_liabilities)
df.file2.subset <- distinct(df.file2.subset)

df.joined.file1.file2 <- left_join(df.file1.subset, df.file2.subset, by = c("gvkey" = "gvkey", "fyear" =
```