Cran download logs aggregation time summary

A.Birek, M.Kosinski, N.Ryciak, W.Ryciuk

May 12, 2014

CONTENTS

Figure 1: Powered by ! https://github.com/MarcinKosinski/AlmostBigData

CHAPTER
ONE

DOWNLOADING DATA

Syntax used for downloading, unzipping and merging data is available in section 5.1. More or less downloading looked like this and took about: 1783.13 s

```
start <- as.Date("2012-10-01")
today <- as.Date("2014-05-10")
all_days <- seq(start, today, by = "day")
year <- as.POSIXlt(all_days)$year + 1900
urls <- paste0("http://cran-logs.rstudio.com/", year, "/", all_days, ".csv.gz")

destdir <- "D:/bd1/AlmostBigData/cran-logs/"
n <- length(urls)
i = 1
for (i in 1:n) {
    destfile <- stri_paste(destdir, as.character(all_days[i]))
    download.file(urls[i], destfile)
}
```

Unzipping files syntax looked like this and took:

```
lok <- "D:/bd1/AlmostBigData"
gzpath <- character(n)
i <- 1
for (i in 1:n) {
    gzpath[i] <- paste(lok, "/cran-logs", all_days[i], sep = "")
}
install.packages("R.utils")
library(R.utils)
for (i in 1:n) {
    gunzip(gzpath[i], destname = paste(gzpath[i], ".csv"), remove = TRUE)
}
```

Converting CSV files with proper delimiter syntax looked like this and time spent was:

```
for (i in 1:n) {
    write.csv2(read.csv2(paste(gzpath[i], ".csv"), sep = ","), paste(gzpath[i], "_new.csv"))
}
```

CHAPTER
TWO

SAS PATH

Syntax used for importing, merging and summarizing data is available in chapter 6.

2.1 Importing data

Importing csv files into **SAS** syntax looked like this and took: average 0.2 s for each file, that gives **2 min 40 S** (Stoper method.)

```
proc import datafile='D:/bd1/AlmostBigData/cran-logs2012-10-01 _new.csv'
out=CR.cran1 dbms=csv replace;
      delimiter = ';';
      getnames=yes;
      run;

...

proc import datafile='D:/bd1/AlmostBigData/cran-logs2014-05-09 _new.csv'
out=CR.cran586 dbms=csv replace;
      delimiter = ';';
      getnames=yes;
      run;
```

2.2 Merging files

Merging all those files syntax looked like this and time expired was:

```
data Cr.DANE;
set
CR.cran1,
CR.cran2,
....
CR.cran586;
run;
```

Time expired:

```
NOTE: The data set CR.DANE has 41611796 observations and 11 variables.
NOTE: DATA statement used (Total process time):
      real time           1:22.02
      cpu time            11.94 seconds
```

2.3 Summary for each variable

Summaries of each variable syntax looked like this and time expired was:

```
12   proc summary data=Cr.DANE2 print;
13   class package;
14   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           29.82 seconds
      cpu time            20.06 seconds


15
16   proc summary data=Cr.DANE2 print;
17   class version;
18   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           31.23 seconds
      cpu time            20.18 seconds


19
20   proc summary data=Cr.DANE2 print;
21   class r_arch;
22   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           30.65 seconds
      cpu time            10.12 seconds


23
24   proc summary data=Cr.DANE2 print;
25   class r_os;
26   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           30.59 seconds
      cpu time            12.58 seconds


27
28   proc summary data=Cr.DANE2 print;
29   class r_version;
30   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           30.56 seconds
      cpu time            10.95 seconds


31
32   proc summary data=Cr.DANE2 print;
33   class country;
34   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           30.07 seconds
```

```
      cpu time            12.48 seconds
```

2.4 Sorting

Sort procedure is required that frequency table can be computed. Unfortunately it takes over 3 minutes...

```
1    proc sort data=Cr.Dane out=CR.Dane2;
2    by r_os;
3    run;

NOTE: There were 41611796 observations read from the data set CR.DANE.
NOTE: The data set CR.DANE2 has 41611796 observations and 11 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           3:49.71
      cpu time            35.39 seconds
```

2.5 Frequency tables

2.5.1 r os

Frequency tables syntax and the time expired for `r os`:

```
    proc freq data=Cr.dane2;
    tables r_os;
    run;

NOTE: Writing HTML Body file: sashtml.htm
NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           32.60 seconds
      cpu time            5.46 seconds
```

2.5.2 Packages

Frequency tables syntax and the time expired for `packages`, grouped by `r os`:

```
   proc freq data=Cr.dane2 page;
   by r_os;
   tables package;
   run;

NOTE: There were 41611796 observations read from the data set CR.DANE2.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           53.25 seconds
      cpu time            32.46 seconds
```

CHAPTER
THREE

TRADITIONAL $\mathcal{R}$ PATH

3.1 Unmerged $\mathcal{R}$ files Path

3.2 Merged $\mathcal{R}$ files Path

Merging all filles with R looked like this:

```
start <- as.Date("2012-10-01")
today <- as.Date("2014-05-09")

all_days <- seq(start, today, by = "day")

temp_dir <- "~//BigData"
filenames <- paste0(temp_dir, "/", all_days, "_newf.csv")
for (i in 1:length(filenames)) {
    dane <- read.csv2(filenames[i])
    dane <- dane[c(-1)]

    write.table(dane, file = paste(temp_dir, "/dane.csv", sep = ""), append = TRUE, dec = ",",
        sep = ";", qmethod = "double", col.names = FALSE, row.names = FALSE)
}
```

It took around 25 mins.

3.2.1 Names

Getting all packages names, architectures kinds and operating system names was done in around 15 min using the following code:

```
options(stringsAsFactors = FALSE)
temp_dir <- destdir <- "~//BigData"
fname <- paste(temp_dir, "/dane.csv", sep = "")

nazwy <- vector("character")

skipy <- 0
nrowsy <- 2e+06
nazwy <- character(0)
arch <- character(0)
r_os <- character(0)

ileLinijek <- 0

while (class(try({
    d <- read.csv2(fname, skip = skipy, nrows = nrowsy)
}, silent = TRUE)) != "try-error") {
    nazwy <- union(d[, 7], nazwy)
    arch <- union(d[, 5], arch)
```

```
    r_os <- union(d[, 6], r_os)
    ileLinijek <- ileLinijek + length(d[, 1])
    skipy <- skipy + nrowsy
}


nazwy <- na.omit(nazwy)
arch <- na.omit(arch)
r_os <- na.omit(r_os)
n <- length(nazwy)
a <- length(arch)
r <- length(r_os)
a_r <- data.frame(matrix(0, nrow = a, ncol = r))
rownames(a_r) <- arch
names(a_r) <- r_os


pakiety <- data.frame(rep(0, length.out = n), row.names = nazwy)
names(pakiety) <- "Krotnosci"
```

3.2.2 Getting Data

To get required data we used the following code:

```
skipy <- 0
nrowsy <- 2e+06
while (class(try({
    d <- read.csv2(fname, skip = skipy, nrows = nrowsy)
}, silent = TRUE)) != "try-error") {
    for (j in 1:length(d[, 1])) {
        if (!is.na(d[j, 7]))
            pakiety[(d[j, 7]), ] <- pakiety[(d[j, 7]), ] + 1
    }


    skipy <- skipy + nrowsy
}
```

```
skipy <- 0
nrowsy <- 2e+06
while (class(try({
    d <- read.csv2(fname, skip = skipy, nrows = nrowsy)
}, silent = TRUE)) != "try-error") {
    un <- unique(d[, 10])
    un <- na.omit(un)
    for (i in 1:length(un)) {
        s <- (d[d[, 10] == un[i], ][1, ])
        aa <- s[1, 6]
        rr <- s[1, 5]
        if (!is.na(aa) & !is.na(rr))
            a_r[rr, aa] <- a_r[rr, aa] + 1
    }

    skipy <- skipy + nrowsy
}
```

First took around 2 hours, and second 3 hours.

CHAPTER
FOUR

RCPP PATH

4.1 Importing data

Finding `csv` files names (paths) using **R** syntax looked like this and took approximately 0.2 s.

```
system.time({
    names <- list.files("D:/Artur/AlmostBigData/", full.names = T, recursive = TRUE)
})
```

4.2 Merging files

Files were not merged into one file. They were being opened one by one in program.

4.3 Frequency tables

Syntax used for counting is available in section 7

4.3.1 `r os`

Frequency tables syntax and the time expired for `r os`:

```
system.time({
wynik <- CountDownloads(names, "r_os")
})

user     system    time
167.08   38.00     207.65
```

which is 2 min. 48 s.

4.3.2 Packages

Frequency tables syntax and the time expired for `packages`:

```
system.time({
wynik <- CountDownloads(names, "package")
})

user     system    time
212.10   40.25     255.82
```

which is 3 min. 32 s.

CHAPTER
FIVE

PREPARING REPORT

5.1 Data download, unzipp, conversion syntax

```
start <- as.Date("2012-10-01")
today <- as.Date("2014-05-10")
all_days <- seq(start, today, by = "day")
year <- as.POSIXlt(all_days)$year + 1900
urls <- paste0("http://cran-logs.rstudio.com/", year, "/", all_days, ".csv.gz")

destdir <- "D:/bd1/AlmostBigData/cran-logs/"
n <- length(urls)
i = 1
for (i in 1:n) {
    destfile <- stri_paste(destdir, as.character(all_days[i]))
    download.file(urls[i], destfile)
}

lok <- "D:/bd1/AlmostBigData"
gzpath <- character(n)
i <- 1
for (i in 1:n) {
    gzpath[i] <- paste(lok, "/cran-logs", all_days[i], sep = "")
}
install.packages("R.utils")
library(R.utils)
for (i in 1:n) {
    gunzip(gzpath[i], destname = paste(gzpath[i], ".csv"), remove = TRUE)
}

for (i in 1:n) {
    write.csv2(read.csv2(paste(gzpath[i], ".csv"), sep = ","), paste(gzpath[i], "_new.csv"))
}
```

CHAPTER

SIX

SAS SYNTAX

```
proc import datafile='D:/bd1/AlmostBigData/cran-logs2012-10-01 _new.csv'
out=CR.cran1 dbms=csv replace;
      delimiter = ';';
      getnames=yes;
      run;

...

proc import datafile='D:/bd1/AlmostBigData/cran-logs2014-05-09 _new.csv'
out=CR.cran586 dbms=csv replace;
      delimiter = ';';
      getnames=yes;
      run;

data Cr.DANE;
set
CR.cran1,
CR.cran2,
....
CR.cran586;
run;

proc summary data=Cr.DANE print;
class package;
run;

proc summary data=Cr.DANE print;
class version;
run;

proc summary data=Cr.DANE print;
class r_arch;
run;

proc summary data=Cr.DANE print;
class r_os;
run;

proc summary data=Cr.DANE print;
class r_version;
run;

proc summary data=Cr.DANE print;
class country;
run;
```

```
proc sort data=Cr.Dane out=CR.Dane2;
by r_os;
run;

proc freq data=Cr.dane2 page;
tables r_os;
run;


proc freq data=Cr.dane2 page;
by r_os;
tables package;
run;
```

CHAPTER

SEVEN

RCPP SYNTAX

```
#include <Rcpp.h>
#include <iostream>
#include <string>
#include <fstream>
#include <map>


using namespace Rcpp;
using namespace std;



// [[Rcpp::export]]
CharacterVector ExtractString(string str, int num)
{
   //string tmpstr = as<string>(str[0]);
   string sub_str;
   unsigned pos_start = 0;
   unsigned pos_end = 0;

   if (pos_start!=string::npos) {
      for (int j = 0; j < num; ++j) {
         pos_start = str.find(";", pos_start+1);
      }
      sub_str = str.substr(pos_start+1);
      pos_end = sub_str.find(";");
   }
   return sub_str.substr(0, pos_end);
}



// [[Rcpp::export]]
List CountDownloads(CharacterVector paths, CharacterVector colname)
{
   int colnum;
   string str;
   string val;
   CharacterVector strr;
   int n = paths.size();
   int nrows;

   map<string, int> column;
   map<string, int>::iterator iter;
```

```
   if (colname[0] == "r_version")    colnum=4;
   else if (colname[0] == "r_arch")  colnum=5;
   else if (colname[0] == "r_os")    colnum=6;
   else if (colname[0] == "package") colnum=7;
   else if (colname[0] == "version") colnum=8;
   else if (colname[0] == "country") colnum=9;
   else {cout << "error: Wrong column name." << endl; return 0;}


   for (int i = 0; i < n; i++) {
      char* filepath = (char*)(paths[i]);
      ifstream file (filepath);

      if(file)
      {
         getline(file, str);
         while(getline(file, str)) {
            strr = ExtractString(str, colnum);
            val = as<string>(strr[0]);
            iter = column.find(val);
            if (iter == column.end())
               column[val] = 1;
            else
               iter->second++;
         }
      }
      file.close();
   }

   nrows = column.size();
   CharacterVector col_name(nrows);
   IntegerVector col_count(nrows);
   iter = column.begin();
   for (int i = 0; i < nrows; i++) {
      col_name[i] = iter->first;
      col_count[i] = iter->second;
      iter++;
   }

   DataFrame dframe = DataFrame::create(Named("name") = col_name,
                                        Named("downloads") = col_count);
   List results = List::create(Named("downloads")  = dframe);

   return(results);

}
```