CHAPTER
ONE

RCPP PATH

1.1 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)
})
```

1.2 Merging files

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

1.3 Frequency tables

Syntax used for counting is available in section 2

1.3.1 `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.

1.3.2 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

TWO

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);

}
```