Stat 133 HW06: Package versions in CRAN Archive

Gaston Sanchez

Introduction

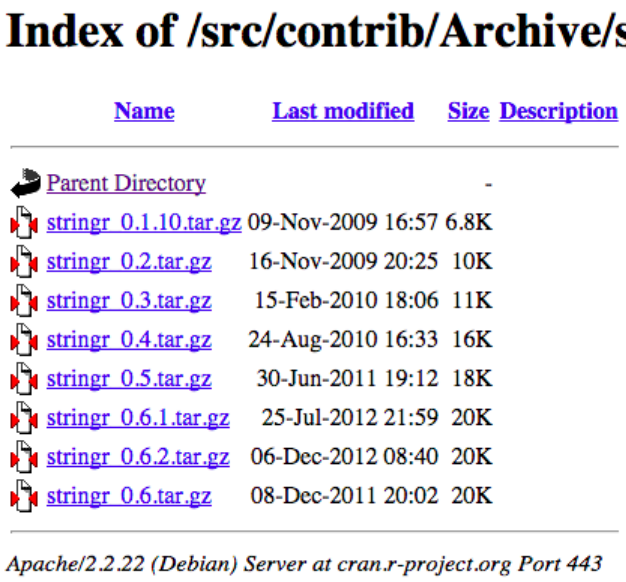
The main purpose of this assignment is to practice writing functions and to make you think in functional-modular form. You will have the chance to do some regex work, a bit of data visualization, and a bit of reporting. Also, you'll get to read HTML tables from the Web.

R packages in CRAN Archive

One of the strongest points of R is the vast number of contributed packages. So far you've worked with some of them such as `"readr"`, `"ggplot2"`, `"stringr"`, and `"knitr"`.

Most R packages live in the **Comprehensive R Archive Network** better known as CRAN. For any package in CRAN, you can check its associated archive. This archive contains the different versions of a package that have been submitted to CRAN. For example, the archive of `"stringr"` (shown in the screen capture below) is in the following url:

http://cran.r-project.org/src/contrib/Archive/stringr



Index of /src/contrib/Archive/stringr

Name	Last modified	Size	Description
Parent Directory		-	
stringr_0.1.10.tar.gz	09-Nov-2009 16:57	6.8K	
stringr_0.2.tar.gz	16-Nov-2009 20:25	10K	
stringr_0.3.tar.gz	15-Feb-2010 18:06	11K	
stringr_0.4.tar.gz	24-Aug-2010 16:33	16K	
stringr_0.5.tar.gz	30-Jun-2011 19:12	18K	
stringr_0.6.1.tar.gz	25-Jul-2012 21:59	20K	
stringr_0.6.2.tar.gz	06-Dec-2012 08:40	20K	
stringr_0.6.tar.gz	08-Dec-2011 20:02	20K	

Apache/2.2.22 (Debian) Server at cran.r-project.org Port 443

Reading HTML Tables

The R package `"XML"` (by Duncan Temple Lang) provides the function `readHTMLTable()` that allows us to read HTML tables from an HTML document. One example of an HTML table is the table with the CRAN archive of `"stringr"`. To read this table in R (as a data.frame) pass the url address to `readHTMLTable()`

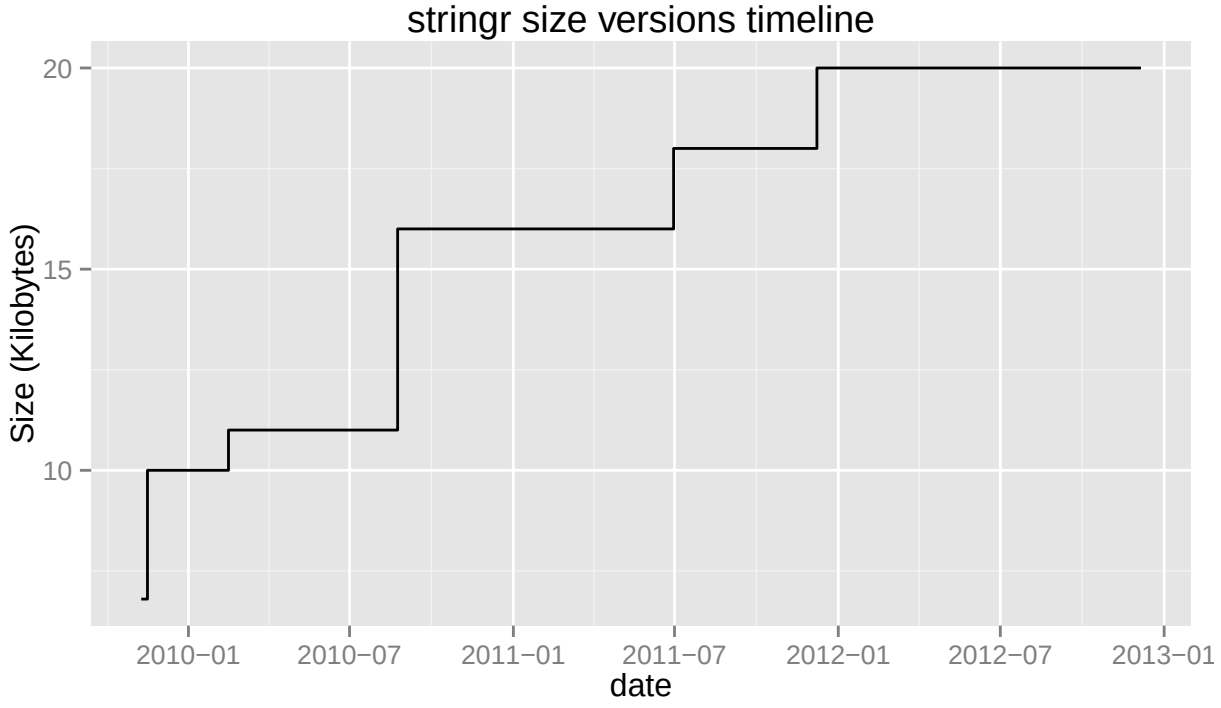
```
readHTMLTable('http://cran.r-project.org/src/contrib/Archive/stringr')
```

```
## $`NULL`
##                       Name     Last modified Size Description
## 1                     <NA>               <NA> <NA>        <NA>
## 2        Parent Directory                          -
## 3   stringr_0.1.10.tar.gz 09-Nov-2009 16:57 6.8K
## 4      stringr_0.2.tar.gz 16-Nov-2009 20:25  10K
## 5      stringr_0.3.tar.gz 15-Feb-2010 18:06  11K
## 6      stringr_0.4.tar.gz 24-Aug-2010 16:33  16K
## 7      stringr_0.5.tar.gz 30-Jun-2011 19:12  18K
## 8    stringr_0.6.1.tar.gz 25-Jul-2012 21:59  20K
## 9    stringr_0.6.2.tar.gz 06-Dec-2012 08:40  20K
## 10     stringr_0.6.tar.gz 08-Dec-2011 20:02  20K
## 11                    <NA>               <NA> <NA>        <NA>
```

The output of `readHTMLTable()` is a list with as many elements as tables are in the HTML document associated to the provided url. In the case of a CRAN archive, there is only one HTML table.

In this assignment, you are going to use the archive data of an R package in order to achieve two goals:

1. Get a cleaned data frame with the archive data
2. Make a step-line plot to visualize the timeline of the version sizes for an R package, like the following figure:



FUNCTIONS

You could write an R script to download the archive data, process it, and generate the step-line plot. If you were to do that, you would have to follow the steps listed below:

- Generate the url of a package CRAN archive
- Check if such url exists (this lets you determine if the provided name is a valid R package name)
- If the package exists (i.e. the name is valid) then download the HTML table
- Remove unwanted rows and keep desired columns from the downloaded data frame
- Transform the raw data to extract cleaned variables:
  - Extract the `name` from column `Name`
  - Extract the version `number` from column `Name`
  - Extract the `date` from column `Last modified` (`date` must be of class `Date`)
  - Get the `size` (in kilobytes) from column `Size`
- Generate a cleaned data.frame with the extracted columns `name`, `number`, `date`, and `size`
- Having a clean data rfame, then plot the step-line with the dates and version-sizes

Instead of writing a script with the previous steps, your mission is to write a set of functions to perform each of the listed steps. This involves writing the following functions:

1. `cran_archive()` creates the CRAN archive URL for a given package name. For instance:

```
# cran archive url for package 'stringr'
cran_archive('stringr')
```

```
## [1] "http://cran.r-project.org/src/contrib/Archive/stringr"
```

2. `is_package()` tests if a given string is a valid R package name. This is done by checking whether the CRAN archive URL exists. To create `is_package()` you'll have to use the function `url.exists()` from the package `"RCurl"`. If the provided string is a valid R package name, `is_package()` returns `TRUE`, otherwise it returns `FALSE`:

```
# TRUE
is_package('XML')

# FALSE
is_package('hola')
```

3. `download_archive()` downloads the archive table as a data.frame. It takes a string (the name of a package) and returns a raw data frame. To do this, you'll have to use the function `readHTMLTable()` from the package `"XML"`.

```
raw_table <- download_archive('stringr')
raw_table
```

```
##                     Name     Last modified Size Description
## 1                   <NA>              <NA> <NA>        <NA>
## 2        Parent Directory                    -
## 3   stringr_0.1.10.tar.gz 09-Nov-2009 16:57 6.8K
## 4      stringr_0.2.tar.gz 16-Nov-2009 20:25  10K
## 5      stringr_0.3.tar.gz 15-Feb-2010 18:06  11K
## 6      stringr_0.4.tar.gz 24-Aug-2010 16:33  16K
## 7      stringr_0.5.tar.gz 30-Jun-2011 19:12  18K
## 8    stringr_0.6.1.tar.gz 25-Jul-2012 21:59  20K
## 9    stringr_0.6.2.tar.gz 06-Dec-2012 08:40  20K
## 10     stringr_0.6.tar.gz 08-Dec-2011 20:02  20K
## 11                  <NA>              <NA> <NA>        <NA>
```

4. **`clean_archive_table()`** removes the unwanted rows, and keeps the desired raw columns `Name`, `Last modified`, and `Size`

5. **`version_names()`** extracts the package names from the raw column `Name` (actually all names are the same one)

6. **`version_nums()`** extracts the version numbers from the raw column `Name` (e.g. 0.1.10)

7. **`version_dates()`** extracts the dates (day-Month-year) from the raw column `Last modified`. The extracted dates should be of class `Date` with format `"%d-%b-%Y"`

8. **`version_sizes()`** extracts the sizes from the raw column `Size`. The extracted sizes must be numeric. Notice that most sizes are in kilobytes (e.g. `6.8K`), but there are packages with sizes in megabytes (e.g. `1.8M`). You should convert megabytes into kilobytes.

9. **`build_table()`** assembles the new data frame with the extracted columns: **`name`**, **`number`**, **`date`**, **`size`**.

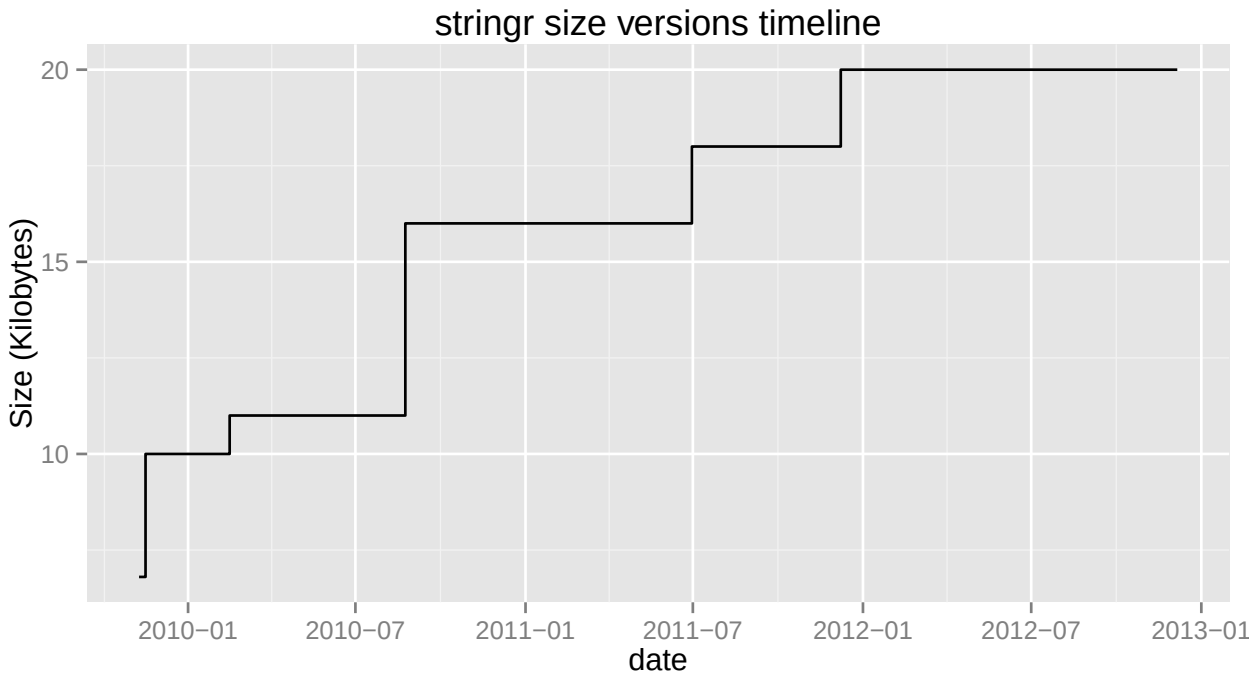
10. **`get_archive_table()`** is the workhorse function. This is actually the only function that you have to call to produce the cleaned and processed data frame. You pass it a string (the name of a package), and it returns the clean and transformed data frame:

```
# data frame of 'stringr' archive
stringr_df <- get_archive_table('stringr')
stringr_df
```

```
##      name number       date size
## 1 stringr 0.1.10 2009-11-09  6.8
## 2 stringr    0.2 2009-11-16 10.0
## 3 stringr    0.3 2010-02-15 11.0
## 4 stringr    0.4 2010-08-24 16.0
## 5 stringr    0.5 2011-06-30 18.0
## 6 stringr  0.6.1 2012-07-25 20.0
## 7 stringr  0.6.2 2012-12-06 20.0
## 8 stringr    0.6 2011-12-08 20.0
```

11. **`ggstep()`** takes the clean data frame and generates a step-line plot with the dates (x-axis) and the sizes (y-axis)

```
# step-line
ggstep(stringr_df)
```



REQUIREMENTS

You have to create a file `functions.R`. This file will be exclusively dedicated to write the functions (just the functions and nothing but the functions!) you'll need for this assignment .

The required packages are: **"RCurl"**, **"XML"**, and **"ggplot2"** (of course you can also use other packages as well)

List of functions (you can add more functions as well):

- `cran_archive()`
- `is_package()`
- `download_archive()`
- `clean_archive_table()`
- `version_names()`
- `version_nums()`
- `version_dates()`
- `version_sizes()`
- `build_table()`
- `get_archive_table()`
- `ggstep()`

Additional considerations:

- Write short functions: no more than 10 lines—excluding comments—. Preferably no more than 5 lines.
- All the listed functions must take just one argument.
- In addition to the listed functions, you can write more functions if you need to.
- Functions should not modify global variables.
- If you write more functions than those listed above, give names that reflect what a function is doing.

Ultimately, you just have to call `get_archive_table()` to get the job done: you pass it a string, and it should return a clean data frame.

REPORT

You'll have to write an `.Rmd` file to perform the analysis and report the results.

To load all your functions—contained in the file `functions.R`—use `source()` inside a code chunk:

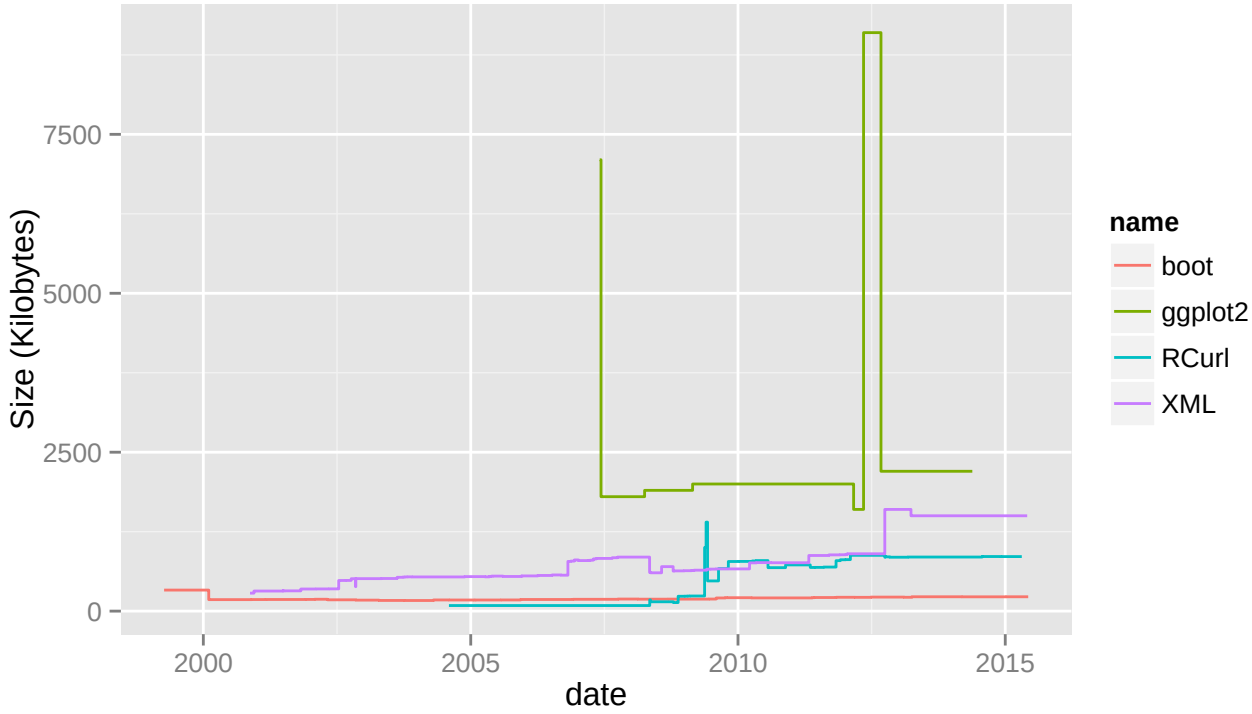
```
# source your functions
# (assuming the file is in your working directory)
source('functions.R')
```

Test your code with the package `"stringr"`; call `get_archive_table()` and `ggstep()`

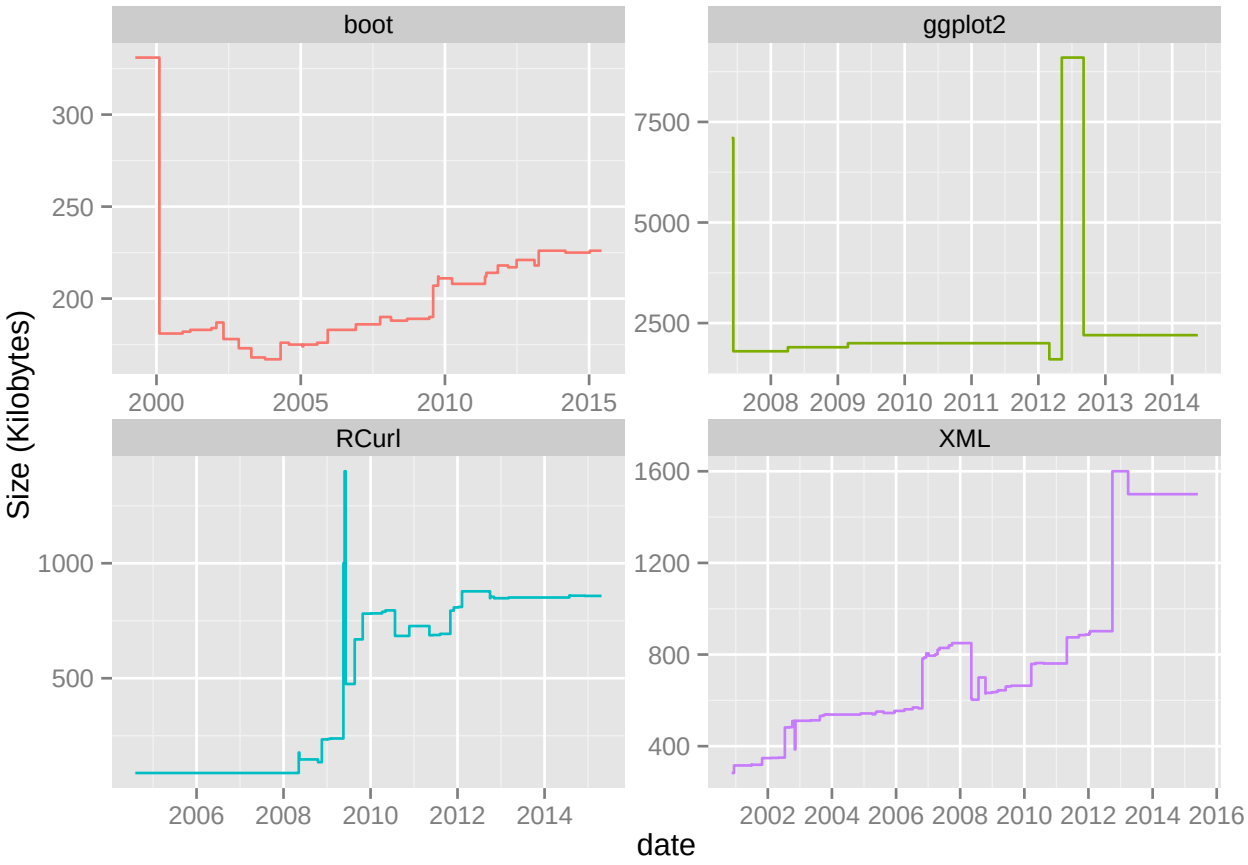
```
# get archive table
stringr_df <- get_archive_table('stringr')

# plot
ggstep(stringr_df)
```

Looking at various packages: Besides testing your code with `"stringr"`, you will have to get the CRAN archive tables for the packages `"XML"`, `"RCurl"`, `"ggplot2"` and `"boot"`. Combine (or merge) all the data tables in one single data.frame. Then, use `"ggplot2"` to generate the following plots:

Plot A: all packages in one single frame



Plot B: one package per facet



Write a brief description for the patterns you see in the plots.