Regular Expressions Lab

Ryan Chui

SCRAPING THE WEB WITH THE XML PACKAGE

We will use the XML R package and regular expressions to retrieve the 250 highest-ranked movies from IMDB and organize the movie name, ranking, and release year into a data frame.

In this section, we have provided all of the code to scrape the Web site. You simply need to execute the commands in sequence to optain the data frame top250.

Be sure to answer the questions in the text and fill in the blanks

```
# If you don't have the XML library already installed,
# do that now. You can run the following line:
# install.packages("XML", dependencies=TRUE)

# Load the XML library
library(XML)
```

Since the ranking information is organized in an HTML table, we will use the function readHTMLTable from the XML() package to read in the rankings. This function converts HTML tables into data frames.

```
tabs = readHTMLTable("http://www.imdb.com/chart/top")
class(tabs)
```

```
## [1] "list"
```

```
length(tabs)
```

```
## [1] 2
```

Use sapply to check the class of each element of tabs and to check the dimensions of each element of tabs.

```
sapply(tabs, class)
```

```
##              NULL amazon-affiliates
##      "data.frame"      "data.frame"
```

```
sapply(tabs, dim)
```

```
##      NULL amazon-affiliates
## [1,]  250                1
## [2,]    5                8
```

Based on the dimensions we found, which table has 250 rows?

ANS: _____

Let's look more closely at the first element in tabs, so we can see how the data is read in and make changes as we need to.

```
movieTab = tabs[[1]]
head(movieTab)
```

```
##                                      Rank & Title IMDb Rating
```

```
## 1  1.\n      The Shawshank Redemption\n        (1994)         9.2
## 2             2.\n      The Godfather\n        (1972)         9.2
## 3    3.\n       The Godfather: Part II\n        (1974)         9.0
## 4           4.\n      The Dark Knight\n        (2008)         8.9
## 5              5.\n      12 Angry Men\n        (1957)         8.9
## 6          6.\n      Schindler's List\n        (1993)         8.9
##
Your Rating
## 1 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
## 2 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
## 3 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
## 4 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
## 5 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
## 6 12345678910\n        \n        \n            \n            NOT YET RELEASED\n
\n            \n            Seen
##   structure(c("", "", "", "", "", ""), class = "AsIs")
## 1
## 2
## 3
## 4
## 5
## 6
```

```
names(movieTab)
```

```
## [1] ""              "Rank & Title" "IMDb Rating"  "Your Rating"
## [5] ""
```

```
sapply(movieTab, class)
```

```
##              Rank & Title  IMDb Rating  Your Rating
##     "factor"     "factor"     "factor"     "factor"     "factor"
```

Fill in the blank:

There are _____ columns and their types are _____

We redo the call to readHTMLTable, this time using some of the arguments to clean this up.

```
top250 = readHTMLTable("http://www.imdb.com/chart/top",
                       which = 1, header = TRUE,
                       stringsAsFactors = FALSE,
                       colClasses = c("character", "character",
                                      "numeric", "character",
                                      "character"))
```

There are two columns which are empty. Find them, as well as the column corresponding to “Your Rating.” Use subsetting to drop these two columns.

```
top250 = top250[ , -c(1,4,5)]
```

After doing this, your data frame should look like this:

```
top250$Yr = as.numeric(gsub("[^[:digit:]]", "", top250[[1]]))
head(top250)
```

```
#                                  Rank & Title IMDb Rating
# 1 1.\n    The Shawshank Redemption\n    (1994)         9.2
# 2            2.\n    The Godfather\n    (1972)         9.2
# 3   3.\n    The Godfather: Part II\n    (1974)         9.0
# 4          4.\n    The Dark Knight\n    (2008)         8.9
# 5             5.\n    Pulp Fiction\n    (1994)         8.9
# 6         6.\n    Schindler's List\n    (1993)         8.9
```

In case this is not what you have after reading in the data because of some version discrepancies in the XML package or R or otherwise, I have uploaded top250 as an RDA file on bCourses so you can read that in and start from there.

```
load("top250.rda")
```

WORKING WITH REGULAR EXPRESSIONS

The first column is a mix of unwanted characters, the title of each movie, and the year it was released. In this section, we will use regular expressions to clean this up so that we have separate columns for title and year, and remove all of the unwanted characters.

First, let's extract the year from the first column and create a new variable with this information. How about eliminating all but digits from the title?

The gsub() command replaces anything that matches the pattern in its first argument with the string in the second argument. We will use gsub to throw away all literals that are not digits.
To do this, we provide an empty string for the second argument.
The first argument needs to correspond to any literal except a digit. Replace the word "pattern"" in the first argument with a pattern that will match any literal except a digit. Recall that equivalence classes of characters are put between [ ] and that a ^ at the start of the equivalence class acts as a negative, i.e., [^azx] says that match aany literal except for lower case a, z, or x.

```
top250$Yr = as.numeric(gsub("[^[:digit:]]", "", top250[[1]]))

head(top250$Yr)
```

```
## [1]   11994   21972   31974   42008 5121957   61993
```

```
# Should give you the results:
# [1] 11994 21972 31974 42008 51994 61993
```

It looks like the year is just the last four characters of the strings in the vector we just created. Use substring() and nchar() to pull these out. Provide the values for start and stop arguments to substring.

```
top250$Yr = sapply(top250$Yr,
                   function(x) {
                     as.numeric(substr(x,
                                       start =  nchar(x)-3,
                                       stop = nchar(x) ))
                   })
```

Great! Now, year is a separate variable.

Let's create a variable now for just the title of the movie.

We'll demonstrate how to use strsplit(). You could actually pull out year this way too, but doing it the way above was some good practice with regular expressions.

The strsplit() function splits each element of its first argument on the character string provided in the second. Here, because each element in the first column looks like "1.The Shawshank Redemption(1994)", splitting on "" is a good way to isolate the title.

Fill in the "split" argument in call below so that this works. Remember that in R we must escape the backslash with an extra

backslash so that R passes the backslash along to the regular espression parser.

```
x = strsplit(top250[[1]], split="\n")

class(x)
```

```
## [1] "list"
```

```
sapply(x, length)
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [211] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [246] 3 3 3 3 3
```

```
x[[1]]
```

```
## [1] "1."                                "      The Shawshank Redemption"
## [3] "        (1994)"
```

Notice that the return value is a list of length 250. If it isn't then recheck your pattern in strsplit.

The second element has the title in it. We can extract it with

```
titleVec = sapply(x, function(el) el[2])
head(titleVec)
```

```
## [1] "      The Shawshank Redemption" "      The Godfather"
## [3] "      The Godfather: Part II"   "      The Dark Knight"
## [5] "      12 Angry Men"             "      Schindler's List"
```

Notice that the titles have some extra leading blanks. Use the gsub function to replace one or more blanks at the start of a string with exactly one blank. To do this supply the pattern below.
Recall that the metacharacter "^" can be used to anchor the pattern to the start of the string. Also recall that there are many different metacharacters to match multiple occurrences of literals. For example, "*" stands for 0 or more, "?" is for 0 or 1, +" is for 1 or more, and the curly braces can be used for exact ranges, e.g., "{2,5}" is for 2 to 5 inclusive. You might also want to use the metacharacter for a blank, i.e., [:blank:]. If you use this then it should be placed within the [ ] for a character class, i.e., [[:blank:]].

```
top250$Title = gsub("^[[:blank:]]+", "", titleVec)
head(top250$Title)
```

```
## [1] "The Shawshank Redemption" "The Godfather"
## [3] "The Godfather: Part II"   "The Dark Knight"
## [5] "12 Angry Men"             "Schindler's List"
```

Now, remove the first column of top250, since we extracted each variable we want from it. We will also add the ranking variable since ther movies are in order and we can easily create it.

```
top250 = top250[ ,-1]
top250$Rank = 1:250
```

We'll save the result as an RDA file.

```r
saveRDS(top250, file="lastnameFirstname_top250.rda")
```