

# **Mergers, Appending, and Web Scrapping**

Jose M. Fernandez

September 11, 2018

## Combining Data Sets: Mergers

Merge - adds variables to a dataset. This document will use **-merge-** function. Merging two datasets require that both have *at least* one variable in common (either string or numeric). If string make sure the categories have the same spelling (i.e. country names, etc.).

Explore each dataset separately before merging. Make sure to use all possible common variables (for example, if merging two panel datasets you will need country and years).

```
# merge two data frames by ID
total <- merge(data.frameA,data frameB,by="ID")

# merge two data frames by ID and Country
total <- merge(data.frameA,data frameB,by=c("ID","Country"))
```

## Different type of merges

An **inner** join of df1 and df2: Return only the rows in which the left table have matching keys in the right table.

An **outer** join of df1 and df2: Returns all rows from both tables, join records from the left which have matching keys in the right table.

A **left** outer join (or simply left join) of df1 and df2 Return all rows from the left table, and any rows with matching keys from the right table.

A **right** outer join of df1 and df2 Return all rows from the right table, and any rows with matching keys from the left table.

## Example of merge

```
df1 = data.frame(CustomerId = c(1:6), Product = c(rep("Toaster", 3), rep("Radio", 3)))
df2 = data.frame(CustomerId = c(2, 4, 6), State = c(rep("Alabama", 2), rep("Ohio", 1)))
inner.df<-merge(df1, df2)
outer.df<-merge(x = df1, y = df2, by = "CustomerId", all = TRUE)

left.df<-merge(x = df1, y = df2, by = "CustomerId", all.x = TRUE)

right.df<-merge(x = df1, y = df2, by = "CustomerId", all.y = TRUE)

cross.df<-merge(x = df1, y = df2, by = NULL)
```

## Combining Data Sets: Append

Append - adds cases/observations to a dataset. This document will use the **-rbind-** function.

Appending two datasets require that both have variables with exactly the same name. If using categorical data make sure the categories on both datasets refer to exactly the same thing (i.e. 1 "Agree", 2 "Disagree", 3 "DK" on both).

# Stacking Dataset

Sometimes you get tables by year, but you want a panel dataset. So one way to merge the files is by stacking them.

```
# Stacking -----  
  
# Let's generate some fake data to illustrate combining data frames by stacking.  
  
first <- data.frame(x0=1:5,  
                   x1=rnorm(5),  
                   x2=c("M", "F", "M", "F", "F"))  
second <- data.frame(x0=10:14,  
                    x1=rnorm(5),  
                    x2=c("M", "F", "M", "F", "F"))  
third <- data.frame(x4=c(3,3,1,3,2),  
                   x5=c("e", "g", "v", "b", "z"))
```

# Appending Data with rbind

```
# We can use the rbind() function to stack data frames. Make sure the number of
# columns match. Also, the names and classes of values being joined must match.
# Here we stack the first, the second and then the first again:
rbind(first, second, first)
```

```
##      x0      x1 x2
## 1  1  0.06437021 M
## 2  2 -0.34096954 F
## 3  3  1.38914779 M
## 4  4  1.47539390 F
## 5  5 -0.81389668 F
## 6 10  0.06716851 M
## 7 11 -0.98140793 F
## 8 12 -2.04668773 M
## 9 13 -1.06328944 F
## 10 14  0.35823543 F
## 11  1  0.06437021 M
## 12  2 -0.34096954 F
## 13  3  1.38914779 M
## 14  4  1.47539390 F
## 15  5 -0.81389668 F
```

```
class(rbind(first, second, first)) # still a data frame
```

```
## [1] "data.frame"
```

```
## [1] "data.frame"
# works with vectors too:
rbind(1:3,4:6)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
class(rbind(1:3,4:6)) # matrix
```

```
## [1] "matrix"
```

## Column Bind

By adding columns: If the two sets of data have an equal set of rows, and the order of the rows is identical, then adding columns makes sense. Your options for doing this are **data.frame** or **cbind()**.

```
# Side-by-side -----
```

```
# Use the cbind function to combine data frames side-by-side:
```

```
cbind(first,third)
```

```
##      x0          x1 x2 x4 x5
## 1  1  0.06437021  M  3  e
## 2  2 -0.34096954  F  3  g
## 3  3  1.38914779  M  1  v
## 4  4  1.47539390  F  3  b
## 5  5 -0.81389668  F  2  z
```

```
class(cbind(first,third))
```

```
## [1] "data.frame"
```



## Warning with cbind

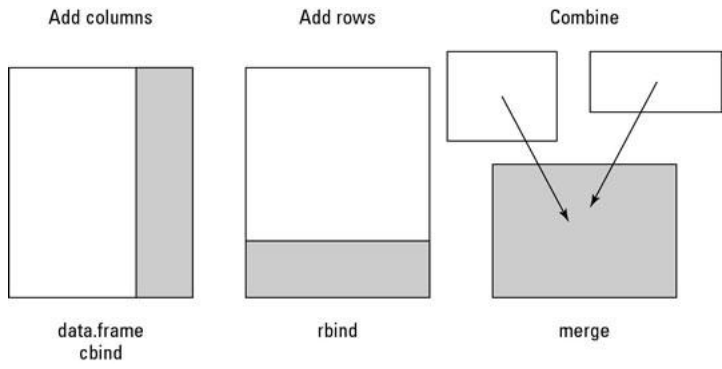
The function `cbind` does not require the vectors to be of the same size. The shorter vector will get recycled.

```
cbind(rbind(first,second),third)
```

```
##      x0      x1 x2 x4 x5
## 1  1  0.06437021 M 3 e
## 2  2 -0.34096954 F 3 g
## 3  3  1.38914779 M 1 v
## 4  4  1.47539390 F 3 b
## 5  5 -0.81389668 F 2 z
## 6 10  0.06716851 M 3 e
## 7 11 -0.98140793 F 3 g
## 8 12 -2.04668773 M 1 v
## 9 13 -1.06328944 F 3 b
## 10 14  0.35823543 F 2 z
```

```
# This is NOT true with rbind
```

# Summary of binding methods



## Merger, Append, and Join

# Webscrapping

We are going to scrape the top 100 movies from IMBD.

There are four steps to webscrapping 1. Extract - obtaining the data - Library to use rvest 2. Preprocess - Cleaning the data - Library to use stringr 3. Analyze - Using the data - Libraries to use dplyr, ggplot2

## What is Web Scrapping?

Web scraping is a technique for converting the data present in unstructured format (HTML tags) over the web to the structured format which can easily be accessed and used.

Almost all the main languages provide ways for performing web scrapping. In these slides, we'll use R for scrapping the data for the most popular feature films from the IMDb website.

We'll get a number of features for each of the 100 popular feature films. Also, we'll look at the most common problems that one might face while scrapping data from the internet because of lack of consistency in the website code and look at how to solve these problems.

## Warning about webscrapping

- Although the information is available on a public website, webscrapping is actively prevented by some companies.
- Using very complicated html code or unorthodox html tags
- embedding webpages
- limiting the number of calls your ip can make to the website
- Your webscrap program will need to website specific. There is no one size fits all approach.
- Your webscrap program will need to be modified if the website is updated.

## Why do we need Web Scrapping?

I am sure the first questions that must have popped in your head till now is “Why do we need web scrapping”? As I stated before, the possibilities with web scrapping are immense.

To provide you with hands-on knowledge, we are going to scrap data from IMDB. Some other possible applications that you can use web scrapping for are:

- Scrapping movie rating data to create movie recommendation engines.
- Scrapping text data from Wikipedia and other sources for making NLP-based systems or training deep learning models for tasks like topic recognition from the given text.
- Scrapping labeled image data from websites like Google, Flickr, etc to train image classification models.
- Scrapping data from social media sites like Facebook and Twitter for performing tasks Sentiment analysis, opinion mining, etc.
- Scrapping user reviews and feedbacks from e-commerce sites like Amazon, Flipkart, etc.

## Ways to scrap data

There are several ways of scraping data from the web. Some of the popular ways are:

- **Human Copy-Paste:** This is a slow and efficient way of scraping data from the web. This involves humans themselves analyzing and copying the data to local storage.
- **Text pattern matching:** Another simple yet powerful approach to extract information from the web is by using regular expression matching facilities of programming languages. You can learn more about regular expressions here.
- **API Interface:** Many websites like Facebook, Twitter, LinkedIn, etc. provides public and/ or private APIs which can be called using standard code for retrieving the data in the prescribed format.
- **DOM Parsing:** By using the web browsers, programs can retrieve the dynamic content generated by client-side scripts. It is also possible to parse web pages into a DOM tree, based on which programs can retrieve parts of these pages.

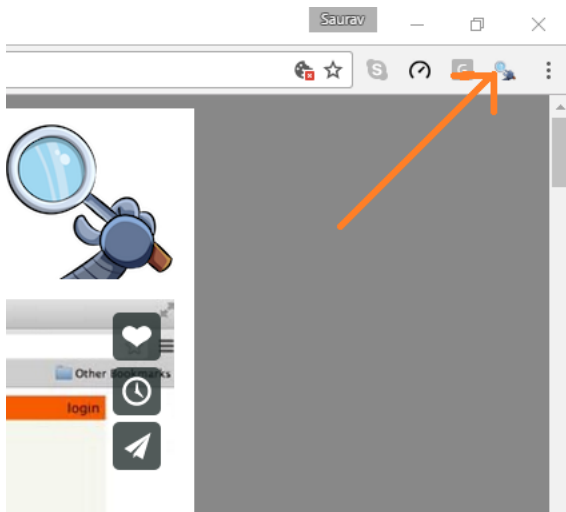
We'll use the DOM parsing approach during the course of this article. And rely on the CSS selectors of the webpage for finding the relevant fields which contain the desired information. But before we begin there are a few prerequisites that one need in order to proficiently scrap data from any website.

## Pre-requisites

- You will need some knowledge of R. Specifically, we need to learn one of the webscrapping packages. The most popular packages is

```
install.packages('rvest')
```

- Second, some knowledge of HTML and CSS will be a huge advantage. However, if you do not have this knowledge, myself included, the use of Select Gadget is very helpful.





## Scraping a webpage using R

Now, let's get started with scraping the IMDb website for the 100 most popular feature films released in 2016. You can access them here.

```
#Loading the rvest package  
library('rvest')
```

```
## Loading required package: xml2
```

```
library('stringr')  
  
#Specifying the url for desired website to be scrapped  
url <- 'https://www.imdb.com/list/ls055592025/'  
  
#Reading the HTML code from the website  
page <- read_html(url)
```

## Now, we'll be scraping the following data from this website.

- **Rank:** The rank of the film from 1 to 100 on the list of 100 most popular feature films released in 2016.
- **Title:** The title of the feature film.
- **Description:** The description of the feature film.
- **Runtime:** The duration of the feature film.
- **Genre:** The genre of the feature film,
- **Rating:** The IMDb rating of the feature film.
- **Votes:** Votes cast in favor of the feature film.
- **Gross\_Earning\_in\_Mil:** The gross earnings of the feature film in millions.
- **Director:** The main director of the feature film. Note, in case of multiple directors, I'll take only the first.
- **Actor:** The main actor of the feature film. Note, in case of multiple actors, I'll take only the first.

# Here's a screenshot that contains how all these fields are arranged.

Find Movies, TV shows, Celebrities and more... PRD | Help

Movies, TV & Showtimes | Celebs, Events & Photos | News & Community | Watchlist | Sign in with Facebook | Other Sign in options

### Top 100 Greatest Movies of All Time (The Ultimate List)


by [ChrisWalczyk55](#) | created - 21 Dec 2012 | updated - 28 Mar 2017 | Public

The movies on this list are ranked according to their success (awards & nominations), their popularity, and their cinematic greatness from a directing/writing perspective. To me, accuracy when making a Top 10/Top 100 all time list is extremely important. My lists are not based on my own personal favorites; they are based on the true greatness and/or success of the person, place or thing being ranked. In other words, a film's commercial success (Oscars & BAFTA Awards), and greatness in direction, screenwriting and production, is how I ranked the films on this list. If you guys would like to view my other Top 10/Top 100 lists, feel free to check out my YouTube page and/or my IMDb page at \*ChrisWalczyk55\*.

Thanks guys and don't forget to LIKE & comment! :)

[Refine](#) | See titles to watch instantly, titles you haven't rated, etc

100 titles Sort by: List Order View:




**1. The Godfather (1972)**  
 R | 175 min | Crime, Drama  
 ★ 9.2 ☆ Rate 100 Metascore  
 The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.  
 Director: [Francis Ford Coppola](#) | Stars: [Marlon Brando](#), [Al Pacino](#), [James Caan](#), [Diane Keaton](#)  
 Votes: 1,365,475 | Gross: \$134.97M

[Watch Now or On Disc](#)  
 From \$2.99 (SD) on [Prima Video](#)

Actors: 5 Stars  
 Direction: 5 Stars  
 Screenplay: 5 Stars


Oscars: 3  
 Oscar Nominations: 11  
 BAFTA Awards: 0  
 BAFTA Nominations: 4  
 Golden Globes: 6  
 Golden Globe Nominations: 8



**2. The Shawshank Redemption (1994)**  
 R | 142 min | Drama  
 ★ 9.3 ☆ Rate 88 Metascore  
 Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.  
 Director: [Frank Darabont](#) | Stars: [Tim Robbins](#), [Morgan Freeman](#), [Bob Gunton](#), [William Sadler](#)  
 Votes: 1,993,524 | Gross: \$28.34M

Actors: 4.8 Stars  
 Direction: 5 Stars  
 Screenplay: 4.9 Stars

Oscars: 0



FROM THE CREATORS OF THIS IS US  
**LIFE ITSELF**  
 ONLY IN THEATERS SEPTEMBER 21  
 < EXPAND >

CREATE A NEW LIST  
 List your movie, TV & celebrity picks.

**List Activity**  
 Views: 13,701,490 | in last week 61,030

**Tell Your Friends**  
 Share this list:

**Feedback?** Tell us what you think about this feature.

**Other Lists by ChrisWalczyk55**

- Top 20 Greatest Suspense/Thrillers of All Time (The Ultimate List)**  
 a list of 20 titles
- Top 100 Greatest Actors of All Time (The Ultimate List)**  
 a list of 100 people
- Top 25 Greatest War Movies of All Time (The Ultimate List)**  
 a list of 25 titles
- Top 50 Greatest Biopics of All Time (The Ultimate List)**  
 a list of 50 titles
- Top 50 Worst Movies of All Time (The Ultimate List)**  
 a list of 50 titles

[See all lists by ChrisWalczyk55](#)

No valid path found. Clear Toggle Position XPath ? X

# Here's a screenshot that contains how all these fields are arranged.

Step 1: Now, we will start with scraping the Rank field. For that, we'll use the selector gadget to get the specific CSS selectors that encloses the rankings. You can click on the extension in your browser and select the rankings field with cursor.

The screenshot displays a web browser interface with movie details for 'The Godfather' and 'The Shawshank Redemption'. A selector gadget is overlaid on the page, showing the CSS selector '.text-primary' selected for the movie titles. The gadget also includes buttons for 'Clear (100)', 'Toggle Position', 'XPath', '?', and 'X'.

**1. The Godfather (1972)**  
 R | 175 min | Crime, Drama  
 ★ 9.2 ☆ Rate 100 Metascore  
 The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.  
 Director: Francis Ford Coppola | Stars: Marlon Brando, Al Pacino, James Caan, Diane Keaton  
 Votes: 1,365,475 | Gross: \$134.97M  
 Watch Now or On Disc  
 From \$2.99 (SD) on Prime Video

Actors: 5 Stars  
 Direction: 5 Stars  
 Screenplay: 5 Stars

Oscars: 3  
 Oscar Nominations: 11  
 BAFTA Awards: 0  
 BAFTA Nominations: 4  
 Golden Globes: 6  
 Golden Globe Nominations: 8

**2. The Shawshank Redemption (1994)**  
 R | 142 min | Drama  
 ★ 9.3 ☆ Rate 80 Metascore  
 Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.  
 Director: Frank Darabont | Stars: Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler  
 Votes: 1,993,524 | Gross: \$28.34M

Actors: 4.8 Stars  
 Direction: 5 Stars  
 Screenplay: 4.9 Stars

Oscars: 0

Selector Gadget: .text-primary | Clear (100) | Toggle Position | XPath | ? | X

**Tell Your Friends**  
 Share this list:

Feedback? Tell us what you think about this feature.

**Other Lists by ChrisWalczyk55**

- Top 20 Greatest Suspense/Thrillers of All Time (The Ultimate List) a list of 20 titles
- Top 100 Greatest Actors of All Time (The Ultimate List) a list of 100 people
- Top 25 Greatest War Movies of All Time (The Ultimate List) a list of 25 titles
- Top 50 Greatest Biopics of All Time (The Ultimate List) a list of 50 titles
- Top 50 Worst Movies of All Time (The Ultimate List) a list of 50 titles

See all lists by ChrisWalczyk55 »

Make sure that all the rankings are selected. You can select some more ranking sections in case you are not able to get all of them and you can also de-select them by clicking on the selected section to make sure that you only have those sections highlighted that you want to scrap for that go.


## Step 2

Step 2: Once you are sure that you have made the right selections, you need to copy the corresponding CSS selector that you can view in the bottom center.

### Most Popular Feature Films Released 2016-01-01 to 2016-12-31

1 to 100 of 12,613 titles | [Next »](#) View Mode: [Compact](#) | [Detailed](#)

Sort by: [Popularity▲](#) | [Alphabetical](#) | [IMDb Rating](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#) | [Release Date](#)



**1. Sing** (2016) +


108 min | Animation, Comedy, Family

★ 7.2 [Rate this](#) 59 Metascore

In a city of humanoid animals, a hustling theater impresario's attempt to save his theater with a singing competition becomes grander than he anticipates even as its finalists' find that their lives will never be the same.

Directors: [Christophe Lourdelet](#), [Garth Jennings](#) | Stars: [Matthew McConaughey](#), [Reese Witherspoon](#), [Seth MacFarlane](#), [Scarlett Johansson](#)

Votes: 40,603 | Gross: \$269.36M



**2. Moana** (I) (2016) +


PG | 107 min | Animation, Adventure, Comedy

★ 7.7 [Rate this](#) 81 Metascore

In Ancient Polynesia, when a terrible curse incurred by the Demigod Maui reaches an impetuous Chieftain's daughter's island, she answers the Ocean's call to seek out the Demigod to set things right.

Directors: [Ron Clements](#), [Don Hall](#), [John Musker](#), [Chris Williams](#) | Stars: [Auli'i Cravalho](#), [Dwayne Johnson](#), [Rachel House](#), [Timuera Morrison](#)

Votes: 91,333 | Gross: \$248.04M



**3. Moonlight** (I) (2016) +

R | 111 min | Drama

★ 7.6 [Rate this](#)

`.text-primary` Clear (100) Toggle Position XPath ? X

## Step 3

Step 3: Once you know the CSS selector that contains the rankings, you can use this simple R code to get all the rankings:

```
#Using CSS selectors to scrap the rankings section  
movie.rank <- html_nodes(page, '.text-primary')  
  
#Converting the ranking data to text  
movie.rank <- html_text(movie.rank)  
  
#Let's have a Look at the rankings  
head(movie.rank)
```

```
## [1] "1." "2." "3." "4." "5." "6."
```

## Step 4

Step 4: Once you have the data, make sure that it looks in the desired format. I am preprocessing my data to convert it to numerical format.

```
#Data-Preprocessing: Converting rankings to numerical  
movie.rank<-as.numeric(movie.rank)
```

```
#Let's have another Look at the rankings  
head(movie.rank)
```

```
## [1] 1 2 3 4 5 6
```

## Step 5

Step 5: Now you can clear the selector section and select all the titles. You can visually inspect that all the titles are selected. Make any required additions and deletions with the help of your cursor. I have done the same here.

### Most Popular Feature Films Released 2016-01-01 to 2016-12-31

1 to 100 of 12,613 titles | [Next »](#) View Mode: [Compact](#) | [Detailed](#)

Sort by: [Popularity](#) | [Alphabetical](#) | [IMDb Rating](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#) | [Release Date](#)



**1. Sing** (2016) +


PG | 108 min | Animation, Comedy, Family

★ 7.2 ☆ [Rate this](#) 59 Metascore

In a city of humanoid animals, a hustling theater impresario's attempt to save his theater with a singing competition becomes grander than he anticipates even as its finalists' find that their lives will never be the same.

Directors: [Christophe Lurdelet](#), [Garth Jennings](#) | Stars: [Matthew McConaughey](#), [Reese Witherspoon](#), [Seth MacFarlane](#), [Scarlett Johansson](#)

Votes: 40,603 | Gross: \$269.36M



**2. Moana** (I) (2016) +

PG | 107 min | Animation, Adventure, Comedy

★ 7.7 ☆ [Rate this](#) 81 Metascore

In Ancient Polynesia, when a terrible curse incurred by the Demigod Maui reaches an impetuous Chieftain's daughter's island, she answers the Ocean's call to seek out the Demigod to set things right.

Directors: [Ron Clements](#), [Don Hall](#), [John Musker](#), [Chris Williams](#) | Stars: [Auli'i Cravalho](#), [Dwayne Johnson](#), [Rachel House](#), [Temuera Morrison](#)

Votes: 91,333 | Gross: \$248.04M



**3. Moonlight** (I) (2016) +

R | 111 min | Drama

.list-item-header a
Clear (100)
Toggle Position
XPath
?
X

We repeat this for each variable



# Extraction Code

```
# movie.rank <- page %>% html_nodes(".text-primary") %>% html_text()
titles <- page %>% html_nodes(".list-item-header a") %>% html_text()
ratings <- page %>% html_nodes(".ipl-rating-star.small .ipl-rating-star_rating") %>% html_text() %>% as.numeric()
release.date <- page %>% html_nodes(".list-item-header") %>% html_text()
movie.rating <- page %>% html_nodes(".certificate") %>% html_text()
starring.cast <- page %>% html_nodes(".text-small:nth-child(6)") %>% html_text()
movie.genre <- page %>% html_nodes(".genre") %>% html_text()
movie.runtime <- page %>% html_nodes(".runtime") %>% html_text()
movie.summary <- page %>% html_nodes(".ipl-rating-widget+ p , .ratings-metascore+ p") %>% html_text()
movie.votes <- page %>% html_nodes(".text-small+ .text-small") %>% html_text()
```

# Data Cleaning

Let's look at each one of our variables starting with titles, the star rating, and the movie rating

```
head(titles) # These one Looks good
```

```
## [1] "The Godfather"      "The Shawshank Redemption"  
## [3] "Schindler's List"   "Raging Bull"  
## [5] "Casablanca"        "Citizen Kane"
```

```
head(ratings)
```

```
## [1] 9.2 9.3 8.9 8.2 8.5 8.4
```

```
head(movie.rating)
```

```
## [1] "R" "R" "R" "R" "PG" "PG"
```

```
movie.rating<-factor(movie.rating)
```

# Data Cleaning

There are some that need to be cleaned up

```
head(release.date) #Not so pretty
```

```
## [1] "\n      1.\n      \n      The Godfather\n      (1972)\n      "\n## [2] "\n      2.\n      \n      The Shawshank Redemption\n      (1994)\n      "\n## [3] "\n      3.\n      \n      Schindler's List\n      (1993)\n      "\n## [4] "\n      4.\n      \n      Raging Bull\n      (1980)\n      "\n## [5] "\n      5.\n      \n      Casablanca\n      (1942)\n      "\n## [6] "\n      6.\n      \n      Citizen Kane\n      (1941)\n      "
```

## Getting rid of some characters

A common type of html code used is new line or return We will use the gsub function to get rid of those terms

```
release.date<-gsub("\n","",release.date)
head(release.date)
```

```
## [1] " 1. The Godfather (1972)"
## [2] " 2. The Shawshank Redemption (1994)"
## [3] " 3. Schindler's List (1993)"
## [4] " 4. Raging Bull (1980)"
## [5] " 5. Casablanca (1942)"
## [6] " 6. Citizen Kane (1941)"
```

## Getting rid of extra spacing

We will use the `gsub` function to get rid of extra spaces as well

```
release.date<-gsub(" ", "", release.date)
head(release.date)
```

```
## [1] "1.The Godfather(1972)"      "2.The Shawshank Redemption(1994)"
## [3] "3.Schindler's List(1993)"    "4.Raging Bull(1980)"
## [5] "5.Casablanca(1942)"        "6.Citizen Kane(1941)"
```

Typically these commands are all you need to clean up a string. However, we want to extract the year between the `()`, which is difficult in R because a lot of functions use `()` and ignore them as a character.

## Getting items between parantheses using stringr

```
# First, we tell R to grab any thing within ()
release.date<-str_extract_all(release.date, "\\([^()]+\\)")
# This code says to grab the items between parantheses
release.date <- substring(release.date, 2, nchar(release.date)-1)
# Lastly, we change the values from character to numeric
release.date <- release.date %>% as.numeric()
head(release.date)
```

```
## [1] 1972 1994 1993 1980 1942 1941
```

## Grabbing only number from a character string

Sometimes we only want to grab the numerical values from a character string. For example in run-time we do not need the label for minutes just the number

```
head(movie.runtime)
```

```
## [1] "175 min" "142 min" "195 min" "129 min" "102 min" "119 min"
```

```
#Grab the first set of numbers  
movie.runtime<-as.numeric(gsub("[0-9]+.*$", "\\1", movie.runtime))  
head(movie.runtime)
```

```
## [1] 175 142 195 129 102 119
```

## Cleaning Votes and Earning

In this slide, we deal with the most complicated variable `movie.vote`. This variable contains both the number of voters as well as Gross Earning for the movie. We need to not only clean up the variable, but also split the variable. The best method is to isolate the values you want for each variable. Let's start with vote totals first.

```
#First, we clean movie.votes
movie.votes <- gsub("\n", "", movie.votes) # removes \n
movie.votes <- gsub(" ", "", movie.votes) # removes extra white space
movie.votes <- gsub("Votes:", "", movie.votes) #removes the word Votes
movie.votes <- gsub("Gross:", "", movie.votes) #removes the word Gross
movie.votes <- gsub(",", "", movie.votes) #removes all the commas
movie.votes1 <- readr::parse_number(movie.votes) #extracts the first number
head(movie.votes1)
```

```
## [1] 1366010 1994300 1028929 275767 455118 349036
```



## Cleaning Revenue

Now that we have vote totals, we want to obtain revenue. We will need to split our movie.vote variable by the character “|”.

```
# This one extracts only Revenue
# It splits the variable at each instance of "|" and holds the second value
# We need to do this for each row of the vector.
# We use sapply to repeat the split at each row
movie.rev<-sapply(strsplit(movie.votes, split='|',fixed=TRUE),function(x) (x[2]))
head(movie.rev)
```

```
## [1] "$134.97M" "$28.34M" "$96.07M" "$23.38M" "$1.02M" "$1.59M"
```

```
# Lastly, we need to extract the numerical values
movie.rev <- readr::parse_number(movie.rev)
head(movie.rev)
```

```
## [1] 134.97 28.34 96.07 23.38 1.02 1.59
```

# Directors and Starring Cast

```
starring.cast<-gsub("\n","",starring.cast)
starring.cast<-gsub(" ","",starring.cast)
starring.cast<-gsub("Director:", "",starring.cast)
starring.cast<-gsub("Directors:", "",starring.cast)
movie.director<-sapply(strsplit(starring.cast, split="| Stars:", fixed=TRUE),function(x) (x[1]))
movie.cast<-sapply(strsplit(starring.cast, split="| Stars:", fixed=TRUE),function(x) (x[2]))
head(movie.director)
```

```
## [1] "Francis Ford Coppola " "Frank Darabont " "Steven Spielberg "
## [4] "Martin Scorsese " "Michael Curtiz " "Orson Welles "
```

```
head(movie.cast)
```

```
## [1] "Marlon Brando, Al Pacino, James Caan, Diane Keaton"
## [2] "Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler"
## [3] "Liam Neeson, Ralph Fiennes, Ben Kingsley, Caroline Goodall"
## [4] "Robert De Niro, Cathy Moriarty, Joe Pesci, Frank Vincent"
## [5] "Humphrey Bogart, Ingrid Bergman, Paul Henreid, Claude Rains"
## [6] "Orson Welles, Joseph Cotten, Dorothy Comingore, Agnes Moorehead"
```

# Clean-up Summary

```
movie.summary <- gsub(" ", "", gsub("\n", "", movie.summary))  
head(movie.summary)
```

```
## [1] "The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son."  
## [2] "Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency."  
## [3] "In German-occupied Poland during World War II, Oskar Schindler gradually becomes concerned for his Jewish workforce after wit  
## [4] "The life of boxer Jake LaMotta, as the violence and temper that leads him to the top in the ring destroys his life outside of  
## [5] "A cynical nightclub owner protects an old flame and her husband from Nazis in Morocco."  
## [6] "Following the death of a publishing tycoon, news reporters scramble to discover the meaning of his final utterance."
```

# Genre Clean-up

```
movie.genre <- gsub(" ", "", gsub("\n", "", movie.genre))  
movie.genre.primary <- factor(gsub(",.*", "", movie.genre))  
head(movie.genre.primary)
```

```
## [1] Crime      Drama      Biography Biography Drama      Drama  
## 11 Levels: Action Adventure Biography Comedy Crime Drama ... Western
```

# Finally Create Data Frame

```
movie.info<-data.frame(rank = movie.rank, titles, rating = movie.rating, score = ratings, year = release.date, votes = movie.votes1, gross = movie.rev, genres=movie
```

# Some Data Analysis

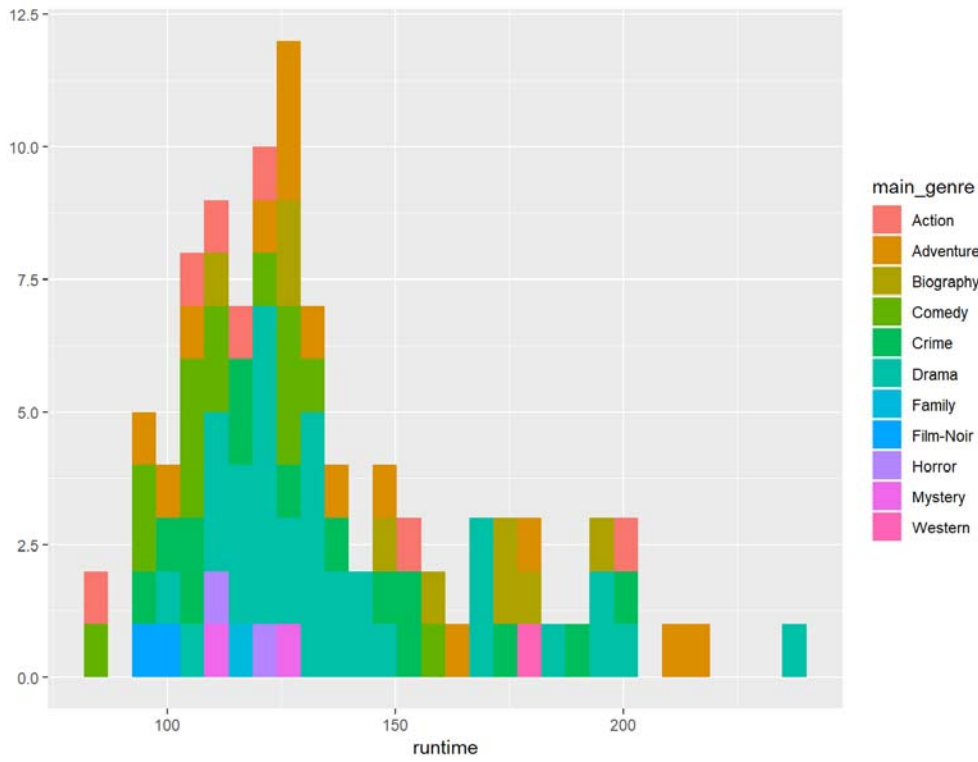
```
library(knitr)
table(movie.info$rating,movie.info$main_genre) %>% kable(caption="Top 100 Movies by Genre and Rating")
```

Top 100 Movies by Genre and Rating

|           | Action | Adventure | Biography | Comedy | Crime | Drama | Family | Film-Noir | Horror | Myster |
|-----------|--------|-----------|-----------|--------|-------|-------|--------|-----------|--------|--------|
| Approved  | 0      | 0         | 0         | 0      | 1     | 1     | 0      | 0         | 0      |        |
| G         | 0      | 2         | 1         | 2      | 0     | 2     | 0      | 0         | 0      |        |
| GP        | 0      | 0         | 1         | 0      | 0     | 0     | 0      | 0         | 0      |        |
| Not Rated | 0      | 3         | 1         | 5      | 3     | 6     | 0      | 2         | 0      |        |
| Passed    | 0      | 2         | 0         | 1      | 1     | 3     | 0      | 0         | 0      |        |
| PG        | 3      | 5         | 1         | 5      | 0     | 6     | 1      | 0         | 0      |        |
| PG-13     | 1      | 2         | 0         | 0      | 0     | 4     | 0      | 0         | 0      |        |
| R         | 3      | 0         | 5         | 1      | 9     | 12    | 0      | 0         | 2      |        |

# Plot 1

```
library('ggplot2')  
qplot(data = movie.info, runtime, fill = main_genre, bins = 30)
```



# Plot 2

```
ggplot(movie.info, aes(x=score,y=gross))+geom_smooth()
```

