Einführung in Web-Scraping mit R

In der täglichen Arbeit begegnen uns Daten nicht immer in Tabellenform. Wenn wir an die vorhergehenden Projekte denken, so konnten wir die benötigten (Zeitreihen) Daten auf der jeweiligen Webseite, z.B. des Ifo-Instituts oder der Bundesbank, mittels eines Download-Buttons herunterladen. Doch auf vielen Webseiten steht ein solcher Download-Button nicht zur Verfügung. Die Informationen sind in der HTML der Webseite eingebettet und nur von dort verfügbar. Um solche Informationen zu analysieren müssen wir diese zuerst in ein für uns geeignetes Format bringen. Hier hilft uns das sogenannte *web scraping* weiter, welches in R durch `rvest` implementiert werden kann.

Der Terminus *web scraping* wird oft verwendet um den Download von Daten eine Webseite zu beschreiben. Dies ist möglich, da der Computercode, welcher in hyper text markup language (HTML) geschrieben wurde, vom Browser als **Text** empfangen und entsprechend gerendert wird. Um sich den HTML-Code einer Website anzeigen zu lassen können Sie auf der jeweiligen Website die rechte Maustaste drücken und dann auf *Seitenquelltext anzeigen* (*View Source*) gehen.

Wenn wir uns HTML-Code anschauen, dann scheint es eher mühsam zu sein die geweiligen Informationen herunterzuladen und in Tabellenform zu bringen. Jedoch bietet R einige Pakete um den Prozess deutlich zu vereinfachen. Für einen ersten Einblick, wie HTML-Code aussehen könnte, zeigen wir ihnen hier einen Ausschnitt der Seite www.imdb.com mit den besten 250 Filmen. Im Laufe dieses Tutoriums lernen Sie, wie Sie diese Informtionen der HTML automatisiert entnehmen.

```
      <table class="chart full-width" data-caller-name="chart-top250movie">
        <colgroup>
          <col class="chartTableColumnPoster"/>
          <col class="chartTableColumnTitle"/>
          <col class="chartTableColumnIMDbRating"/>
          <col class="chartTableColumnYourRating"/>
          <col class="chartTableColumnWatchlistRibbon"/>
        </colgroup>
        <thead>
        <tr>
          <th></th>
          <th>Rank &amp;amp; Title</th>
          <th>IMDb Rating</th>
          <th>Your Rating</th>
          <th></th>
        </tr>
        </thead>
        <tbody class="lister-list">

  <tr>
    <td class="posterColumn">

    <span name="rk" data-value="1"></span>
    <span name="ir" data-value="9.216784978442458"></span>
    <span name="us" data-value="7.791552E11"></span>
    <span name="nv" data-value="1967223"></span>
    <span name="ur" data-value="-1.7832150215575417"></span>
<a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-327b42fe94b1&pf_rd_r=X25RJR4TAEAZG9P04HNC&pf_r
> <img src="https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU0
</a>    </td>
    <td class="titleColumn">
      1.
      <a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-327b42fe94b1&pf_rd_r=X25RJR4TAEAZG9P04HN
title="Frank Darabont (dir.), Tim Robbins, Morgan Freeman" >Die Verurteilten</a>
        <span class="secondaryInfo">(1994)</span>
    </td>
    <td class="ratingColumn imdbRating">
          <strong title="9,2 based on 1.967.223 user ratings">9,2</strong>
    </td>
    <td class="ratingColumn">
```

```
    <div class="seen-widget seen-widget-tt0111161 pending" data-titleid="tt0111161">
        <div class="boundary">
            <div class="popover">
<span class="delete"> </span><ol><li>1<li>2<li>3<li>4<li>5<li>6<li>7<li>8<li>9<li>10</ol>            </div>
        </div>
        <div class="inline">
            <div class="pending"></div>
            <div class="unseeable">NOT YET RELEASED</div>
            <div class="unseen"> </div>
            <div class="rating"></div>
            <div class="seen">Seen</div>
        </div>
    </div>
    </td>
    <td class="watchlistColumn">
        <div class="wlb_ribbon" data-tconst="tt0111161" data-recordmetrics="true"></div>
    </td>
  </tr>
```

Hier sehen Sie die Daten direkt! Der Ausschnitt zeigt, dass der Film "Die Verurteilten" mit 9,2 Punkten (basierend auf 1 967 223 Bewertungen) laut imdb.com der beste Film aller Zeiten ist. In der HTML finden Sie weiterhin die Hauptdarsteller und das Jahr der Veröffentlichung.

Mittels HTML Tags und CSS können Sie diese Informationen auch aus der HTML extrahieren. Hierzu im Laufe des Tutorials mehr. Wenn Sie sich überlegen zukünftig im Bereich der Datenanalyse zu arbeiten, so ist es immer vorteilhaft etwas über HTML und CSS zu wissen. Dies verbessert ihre *web scraping* Fähigkeiten und natürlich auch ihre Fähigkeit eigene Webseiten zu gestalten. Tutorials zu HTML und CSS finden sie beispielsweise auf code academy und WWW3 school

Das Paket `rvest`

Innerhalb von `tidyverse` gibt es das Paket `rvest`, mit welchem Sie *web scraping* in einer ihnen bekannte Syntax durchführen können. Zuerst importieren Sie die gewünschte Seite in R. Hierfür nutzen Sie den Befehl *read_html*:

```
library(tidyverse)
library(rvest)
url <- "https://www.imdb.com/chart/top"
imdb250 <- read_html(url)
```

Nun ist die komplette Website von imdb mit den 250 besten Filmen in `imdb250` gespeichert. Die Klasse von `imdb250` ist:

```
class(imdb250)
```

```
## [1] "xml_document" "xml_node"
```

Das `rvest` Paket ist sehr allgemein gehalten und kann XML Dokumente handhaben. Eine XML Datei ist in einer allgemeinen Markup Sprache geschrieben (daher ML) und kann zur Repräsentation jeglicher Daten dienen. HTML Dateien sind eine Sonderform von XML Dateien um Webseiten darzustellen.

Unser Ziel ist es nun, die Informationen aus der imdb Seite über Ranking, Rating, Anzahl der Stimmen etc. zu extrahieren. Wenn wir uns `imdb250` ausgeben lassen, so sind wir diesem Ziel noch nicht näher gekommen, außer das die HTML nun als R Objekt verfügbar ist:

```
imdb250
```

```
## {xml_document}
## <html xmlns:og="http://ogp.me/ns#" xmlns:fb="http://www.facebook.com/2008/fbml">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset= ...
## [2] <body id="styleguide-v2" class="fixed">\n            <img height="1" ...
```

Denken wir zurück an unseren HTML Ausschnitt von oben. In diesem wird deutlich, dass unsere Informationen sich in einer HTML Tabelle befinden. Zu sehen ist dies an der Code-Zeile `<table class="chart full-width" data-caller-name="chart-top250movie">`. Die einzelnen Teile einer HTML, welche oft mit einem sprechenden Text zwischen < und > definiert sind, werden als *Nodes* bezeichnet. Mit dem Paket `rvest` können wir die Nodes einer HTML extrahieren: `html_nodes` liest hierbei alle Nodes mit der entsprechenden Definition aus und `html_node` nur die erste Node. Beispielsweise können wir die Tabelle mit den 250 besten Filmen mit folgendem Befehl in der Variable `table` speichern:

```
table <- imdb250 %>% html_node("table")
table
```

```
## {xml_node}
## <table class="chart full-width" data-caller-name="chart-top250movie">
## [1] <colgroup>\n<col class="chartTableColumnPoster">\n<col class="chartT ...
## [2] <thead><tr>\n<th></th>\n          <th>Rank &amp;amp; Title</th>\n        ...
## [3] <tbody class="lister-list">\n<tr>\n<td class="posterColumn">\n\n     ...
```

Allerdings sieht diese Tabelle immer noch sehr stark nach HTML aus und weniger nach einem für uns zu bearbeitenden Data Frame. Mittels `rvest` können wir das jedoch schnell beheben und aus der HTML Tabelle einen Data Frame machen, hierzu nutzen wir die Funktion `html_table`:

```
table <- table %>% html_table
class(table)
```

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

```
nrow(table)
```

```
## [1] 250
```

Nun sind wir unserem Wunschergebnis schon sehr viel näher:

```
head(table)
```

```
##                                       Rank & Title IMDb Rating
## 1 NA       1.\n      Die Verurteilten\n       (1994)         9.2
## 2 NA              2.\n      Der Pate\n       (1972)         9.2
## 3 NA            3.\n      Der Pate 2\n       (1974)         9.0
## 4 NA        4.\n      The Dark Knight\n       (2008)         9.0
## 5 NA 5.\n      Die zwölf Geschworenen\n       (1957)         8.9
## 6 NA       6.\n      Schindlers Liste\n       (1993)         8.9
##
## 1 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
## 2 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
## 3 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
## 4 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
## 5 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
## 6 12345678910\n        \n        \n           \n            NOT YET RELEASED\n            \n
##
## 1 NA
## 2 NA
## 3 NA
## 4 NA
## 5 NA
## 6 NA
```

Hier müssten wir immer noch ein paar Dinge umgestalten, wie z.B. die *newline* entfernen, um zu einem wirklich zufriedenstellenden Ergebnis zu gelangen. Jedoch gibt es bessere und allgemeineren Vorgehensweisen um Informationen aus HTML Seiten zu extrahieren. Hierfür widmen wir uns im nächsten Abschnitt

CSS-Selektoren.

CSS-Selectoren

Webseiten, welche ausschließlich mit grundständigem HTML programmiert sind, wirken recht unattraktiv. Um eine ansprechende Webseite zu gestalten greifen die meisten Webmaster auf CSS zurück. Durch CSS können Sie beispielsweise alle Seiten eines Unternehmens einheitlich gestalten, indem sie auf eine einheitliche CSS Datei zurückgreifen. Grundsätzlich können Sie mit CSS das Aussehen jedes Elements einer Webseite bestimmen. Hierunter fallen unter anderem der Titel, Überschriften, Listen, Tabellen und Links, wobei jedes Element seine eigene Schrift, Farbe, Größe etc. erhalten kann. Um die einzelnen Anpassungen zu machen nutzt CSS sogenannte Selektoren. Beispielsweise wäre `table` ein solcher Selektor, welchen wir zuvor verwendet haben.

Wenn wir nun Daten aus einer Webseite herunterladen und wir kennen eben diese Selektoren, so können wir die Informationen direkt mit `html_nodes` herunterladen. Leider ist es nicht immer einfach herauszufinden, welcher Selektor für welches Element maßgeblich ist. Glücklicherweise gibt es SelectorGadget welches Sie als Lesezeichen zu ihrem Browser hinzufügen können. Bitte schauen Sie sich den Screencast auf www.selectorgadget.com an um einen Einblick in Selectorgadget zu erhalten. Weiterhin erhalten Sie eine Einführung in SelectorGadget indem Sie folgenden Befehl in RStudio ausführen:

```
vignette("selectorgadget")
```

```
## starting httpd help server ... done
```

Mit SelectorGadget können Sie interaktiv jeden CSS Selektor bestimmen welchen Sie ansteuern möchten. Somit können Sie spezifische Inhalte einer Webseite genau ansteuern und herunterladen. Wenn Sie mit SelectorGadget auf ein Element einer Webseite klicken werden alle mit diesem CSS Selektor zusammenhängenden Objekte entsprechend gelb markiert.

Um die Möglichkeiten von CSS Selektoren etwas besser zu veranschaulichen wollen wir uns wieder imdb.com zuwenden. Im weiteren Verlauf des Tutorials wollen wir Daten zu allen Filme, welche 2017 auf den Markt gekommen sind, herunterladen und analysisieren.

Hierzu suchen wir zuerst auf www.imdb.com nach den entsprechenden Filmen um die url herauszufinden Der Link in dem unteren Chunk enthält die url unserer Suche.

```
imdb2017 <- read_html("https://www.imdb.com/search/title?count=100&release_date=2017,2017&title_type=feature&page=1&ref_=adv_nxt")
```

Beschauen wir uns diesen Link etwas genauer:

- count: Anzahl der Filme die auf einer Seite gezeigt werden, hier 100
- release_date: Jahr in dem der Film gedreht wurde (wird als [von,bis] angegeben, d.h. im gesamten Jahr 2017)
- page: Auf welcher Seite wir uns aktuell befinden

Im Folgenden möchten wir uns nun folgende Informationen über den jeweiligen Film genauer anschauen: - Popularität - Titel - Rating - Dauer - Genre - Anzahl der Stimmen

```
#Sortierung nach Popularität
popul <- imdb2017 %>%
  html_nodes(".text-primary") %>%
  html_text() %>%
  as.numeric()

#Titel
titel <- imdb2017 %>%
  html_nodes(".lister-item-header a") %>%
  html_text()
```

```
#Rating
rating <- imdb2017 %>%
  html_nodes(".ratings-imdb-rating strong") %>%
  html_text() %>%
  as.numeric()

#Dauer
dauer <- imdb2017 %>%
  html_nodes(".text-muted .runtime") %>%
  html_text() %>%
  str_replace(" min", "") %>%
  as.numeric()

#Genre (immer das erste Genre als Klassifikation)
genre <- imdb2017 %>%
  html_nodes(".genre") %>%
  html_text() %>%
  str_replace("\n", "") %>% str_replace(" ", "") %>%
  str_replace(",.*", "") %>% str_trim() %>%
  as.factor()

#Anzahl an Stimmen
stimmen <- imdb2017 %>%
  html_nodes(".sort-num_votes-visible span:nth-child(2)") %>%
  html_text() %>%
  str_replace(",", "") %>%
  as.numeric()
```

Teilweise sind die Selektoren recht komplex und wir bekommen auch nicht immer 100%-ig saubere Werte zurückgeliefert, sondern müssen diese nachträglich bearbeiten. Dies geschieht mit dem Paket `stringr`, welches Teil des `tidyverse` ist, insbesondere der Funktion *str_replace* um bspw. Ersetzungen vorzunehmen. Doch es wird auch deutlich, dass wir durch CSS-Selektoren direkt auf einzelnen Elemente zugreifen können und dadurch deutlich mehr Flexibilität als mit reinen HTML Befehlen haben.

Auf dieser Grundlage können wir ein Data Frame erstellen, welcher alle geforderten Informationen beinhaltet:

```
Filme2017 <- tibble(popul,titel, rating, dauer, genre, stimmen)
Filme2017
```

```
## # A tibble: 100 x 6
##    popul titel                                rating dauer genre    stimmen
##    <dbl> <chr>                                 <dbl> <dbl> <fct>      <dbl>
##  1     1 Thor: Tag der Entscheidung              7.9   130 Action    353837
##  2     2 Kingsman: The Golden Circle             6.8   141 Action    186449
##  3     3 Greatest Showman                        7.7   105 Biograp~  136273
##  4     4 Blade Runner 2049                       8.1   164 Drama     308348
##  5     5 Es                                      7.4   135 Drama     284410
##  6     6 Star Wars - Episode VIII: Die letz~     7.3   152 Action    395173
##  7     7 Coco - Lebendiger als das Leben         8.5   105 Animati~  191951
##  8     8 Jumanji: Willkommen im Dschungel        7     119 Action    177807
##  9     9 The Endless                             6.6   111 Fantasy     3541
## 10    10 Dark River                              5.7    90 Drama        814
## # ... with 90 more rows
```

Dadurch, dass alle Seitenelemente dem gleichen Schema folgen, könnten wir für die obere Abfrage auch eine Funktion schreiben.

Diese Funktion können wir anschließend auf die 2. Seite anwenden (und auf alle weiteren):

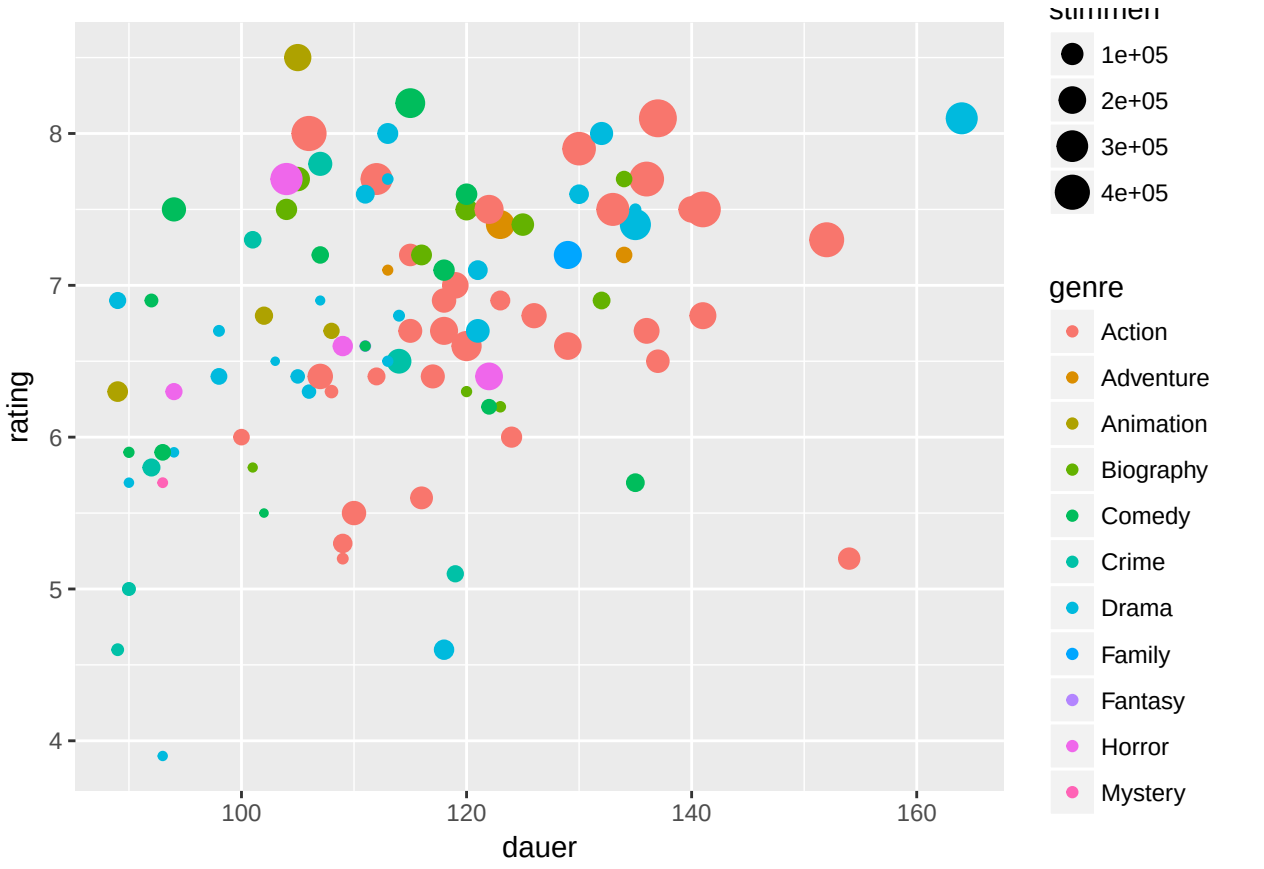
```
get_imdb2017 <- function(url){
  imdb <- read_html(url)
  popul <- imdb %>% html_nodes(".text-primary") %>% html_text() %>% as.numeric()
  titel <- imdb %>% html_nodes(".lister-item-header a") %>% html_text()
  rating <- imdb %>% html_nodes(".ratings-imdb-rating strong") %>% html_text() %>% as.numeric()
  dauer <- imdb %>% html_nodes(".text-muted .runtime") %>% html_text() %>% str_replace(" min", "") %>% as.numeric()
  genre <- imdb %>% html_nodes(".genre") %>% html_text() %>% str_replace("\n", "") %>% str_replace(" ", "") %>%
    str_replace(",.*", "") %>% as.factor()
  stimmen <- imdb %>% html_nodes(".sort-num_votes-visible span:nth-child(2)") %>% html_text() %>% str_replace(",", "") %>%
    as.numeric()
  tibble(popul,titel, rating, dauer, genre, stimmen)
}
```

```
Filme2 <- get_imdb2017("https://www.imdb.com/search/title?count=100&release_date=2017,2017&title_type=feature&page=2&ref_=adv_nxt")
```

Die Daten, welche wir uns heruntergeladen haben können wir nun auch analysieren. Beispielsweise wäre interessant zu wissen, welche Filme in 2017 die meisten Stimmen mit dem höchsten Rating erhalten haben. Dies getrennt nach Genre und Dauer des Films:

```
library(ggplot2)

#which genre has the highest votes
ggplot(Filme2017,aes(x=dauer,y=rating)) + geom_point(aes(size=stimmen,col=genre))
```



Wenn Sie sich vertieft mit *web scraping* beschäftigen wollen so können Sie sich insbesondere noch die Funktionen `html_form`, `set_values`, und `submit_form` anschauen. Jedoch gehen diese Funktionen für eine Einführung in das *web scraping* etwas zu weit.