

# Political Data Science

Lektion 5:

Web scraping & API

Undervist af Jesper Svejgaard, foråret 2018  
Institut for Statskundskab, Københavns Universitet  
[github.com/jespersvejgaard/PDS](https://github.com/jespersvejgaard/PDS)

# I dag

1. Opsamling fra sidst
2. Dagens pensum
3. Opgave-session
4. Opsamling og næste gang

# Overblik

1. Intro til kurset og R
2. R Workshop I: Explore
3. R Workshop II: Import, tidy, transform
4. R Workshop III: Programmering & Git
5. Web scraping & API
6. Tekst som data
7. Visualisering
8. GIS & spatiale data
9. Estimation & prædiktion
10. Superviseret læring I
11. Superviseret læring II
12. Usuperviseret læring
13. Refleksioner om data science
14. Opsamling og eksamen

# In other news

- Inspiration til replikationsstudier findes på  
`PDS/seminaropgaver/replikationsdata.txt`

Opsamling fra sidst

# Opsamling fra sidst

## Funktioner & conditionals

```
hemmelig_funktion <- function(df){  
  
  if (is.data.frame(df)) {  
    klasser <- map(df, class)  
    return(klasser)  
  } else  
    "Stik mig en DF!"  
}
```

# Opsamling fra sidst

## Funktioner & conditionals

```
# Eksekverer funktion på flights-datasættet fra pakken nycflights13  
hemmelig_funktion(nycflights13::flights)
```

```
## $year  
## [1] "integer"  
##  
## $month  
## [1] "integer"  
##  
## $day  
## [1] "integer"  
##  
## $dep_time  
## [1] "integer"  
##  
## $sched_dep_time  
## [1] "integer"  
##  
## $dep_delay
```

# Opsamling fra sidst

## Loops

```
df <- nycflights13::flights
gns <- list()

for (i in seq_along(df)){
  if (is.numeric(df[[i]])){
    gns[[i]] <- median(df[[i]], na.rm = T)
  } else
    gns[[i]] <- "Variablen er ikke numerisk"
}

names(gns) <- names(df)
```



# Opsamling fra sidst

## Loops

```
# Tjekker listen gns ud  
glimpse(gns)
```

```
## List of 19  
## $ year      : num 2013  
## $ month     : num 7  
## $ day       : num 16  
## $ dep_time  : int 1401  
## $ sched_dep_time: num 1359  
## $ dep_delay : num -2  
## $ arr_time  : int 1535  
## $ sched_arr_time: num 1556  
## $ arr_delay : num -5  
## $ carrier   : chr "Variablen er ikke numerisk"  
## $ flight    : num 1496  
## $ tailnum   : chr "Variablen er ikke numerisk"  
## $ origin    : chr "Variablen er ikke numerisk"  
## $ dest      : chr "Variablen er ikke numerisk"  
## $ air_time  : num 129
```

# Version control

Snak med sidepersonen:

- Hvad er version control
- Hvorfor version control
- Hvordan er et alm. workflow

Dagens pensum

# Web data og API'er

Hvad er en API?

- API = Application Programming Interface
- "... en softwaregrænseflade, der tillader et stykke software at interagere med andet software" jf. [Wikipedia](#)
- Svarer til en tjener på en restaurant: Modtager og leverer bestillinger
- Bruges bl.a. til at forbinde programmer og distribuere data
- Kan returnere alt fra tekst til billeder - vi vil ofte få XML- og JSON-filer retur

Hvordan?

- Interaktion via klienter, fx pakker som `rtweet` og `Rfacebook`
- Interaktion med HTTP-forespørgsler som GET og POST, fx via pakken `httr`
- Man må selv læse (eller tænke sig til) API'ens dokumentation

# Etik og respektfuld adfærd

## Authentication

- Mange API'er kræver, at man identificerer sig med en **key** og en **secret**

## User agents

- Det er god stil at sende et ID og evt. forklaring med sine requests
- `Fx GET("url", user_agent("jsj@ifs.ku.dk data til undervisning"))`

## Rate limiting

- Begrænsning af sine requests med tidsintervaller for at spare serverne.

## Åbne vs. tilgængelige data

- Tilgængelige data  $\neq$  åbne data

# Hente data via en API

## Eksempel

```
# Definerer sti til API
url <- "http://oda.ft.dk/api/Afstemning?$inlinecount=allpages&$skip=0"

# Eksekverer GET-request
response <- GET(url)

# Tjekker response-objekt ud
response # indeholder bl.a. status code, header, body

## Response [http://oda.ft.dk/api/Afstemning?$inlinecount=allpages&$skip=0]
##   Date: 2018-03-06 08:58
##   Status: 200
##   Content-Type: application/json; charset=utf-8
##   Size: 6.95 kB
## {
##   "odata.metadata":"http://oda.ft.dk/api/%24metadata#Afstemning","odata....
##   {
##     "id":1,"nummer":411,"konklusion":"Vedtaget\n\n108 stemmer for fors...
##   },{
```

# Hente data via en API

## Eksempel

```
# Ekstraherer response-objektets body med `content()``  
content(response, as = "text")
```

```
## [1] "{\r\n  \"odata.metadata\": \"http://oda.ft.dk/api/%24metadata#Afstemning\", \"odata.count
```

# JSON

Hvad er JSON?

- JSON = JavaScript Object Notation
- Et let tekstformat til opbevaring og udveksling af data
- Plain tekst filer med særlige konventioner til at beskrive data-strukturer:
  - objects: key-value-pairs
  - arrays: ordnede lister
  - værdier: strenge, tal, logicals, objekter, arrays

Hvorfor JSON?

- Nested struktur = kan indeholde mere kompliceret data end rektangulær tabel



# JSON eksempel

Her: én array med to objekter, der hver har to key-value-pairs: titel og år

```
[
  {
    "title" : "A New Hope",
    "year" : 1977
  },
  {
    "title" : "The Empire Strikes Back",
    "year" : 1980
  }
]
```

# Håndtering af JSON-filer

Fra URL til dataframe vha. pakkerne `httr`, `jsonlite` og `dplyr`:

```
# Definerer URL, laver GET-request og modtager et response-objekt  
url <- "http://oda.ft.dk/api/Afstemning?$inlinecount=allpages"  
response <- GET(url)
```

```
# Ekstraherer tekst/JSON fra response-objektet  
response_json <- content(response, as = "text")
```

```
# Tjekker JSON-objektet ud  
response_json
```

```
## [1] "{\r\n  \"odata.metadata\": \"http://oda.ft.dk/api/%24metadata#Afstemning\", \"odata.count\"
```

# Håndtering af JSON-filer

```
# Laver JSON-filen om til en liste og data frame
afstemning_list <- fromJSON(response_json)
afstemning_df <- bind_rows(afstemning_list$value)
```

```
# Tjekker dataframe
glimpse(afstemning_df)
```

```
## Observations: 20
## Variables: 9
## $ id          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,...
## $ nummer      <int> 411, 412, 1, 7, 412, 410, 408, 407, 404, 405, ...
## $ konklusion  <chr> "Vedtaget\n\n108 stemmer for forslaget (V, S, ...
## $ vedtaget    <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FAL...
## $ kommentar   <chr> NA, NA, "", "", "", "", "", "", "", "", "", ""...
## $ mødeid      <int> 17, 18, 41, 156, 18, 15, 962, 962, 962, 962, 9...
## $ typeid      <int> 2, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ sagstrinid  <int> NA, 4849, 17351, 18370, 4849, 16581, 3311, 322...
## $ opdateringsdato <chr> "2014-09-09T09:05:59.653", "2014-09-09T09:25:0...
```

# XML

Hvad er XML?

- XML = Extensible Markup Language
- Markup = anoteret tekst, fx med tags som `<id> ... </id>`
- XML-filer er plain tekst filer, der består af markup og content

Hvorfor XML?

- Annoteret tekst = godt til at opbevare data

# XML eksempel

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

# Håndtering af XML-filer

Fra URL til dataframe vha. pakken `xml2`:

```
# Definerer URL, laver GET-request og modtager et response-objekt
url <- "https://raw.githubusercontent.com/jespersvejgaard/PDS/master/data/afstemninger.xml"
response <- GET(url)

# Ekstraherer tekst/XML fra response-objektet
response_xml <- content(response, as = "text")

# Laver XML-filen om til en liste
afstemninger_list <- read_xml(response_xml)

# Henter de tre nodes "id", "nummer" og "konklusion" vha. XPATH
afstemning_id <- xml_find_all(afstemninger_list, "//d:id")
afstemning_nr <- xml_find_all(afstemninger_list, "//d:nummer")
afstemning_konklusion <- xml_find_all(afstemninger_list, "//d:konklusion")
```

# Håndtering af XML-filer

```
# Laver XML-elementer om til dataframe
afstemninger_df <- tibble(afstemning_id = xml_text(afstemning_id),
                           afstemning_nr = xml_text(afstemning_nr),
                           afstemning_konklusion = xml_text(afstemning_konklusion))
```

```
# Tjekker dataframe
glimpse(afstemninger_df)
```

```
## Observations: 20
## Variables: 3
## $ afstemning_id      <chr> "1", "2", "3", "4", "5", "6", "7", "8", ...
## $ afstemning_nr      <chr> "411", "412", "1", "7", "412", "410", "4...
## $ afstemning_konklusion <chr> "Vedtaget\n\n108 stemmer for forslaget (..."
```

# Web scraping

Hvad er web scraping?

- Ekstrahere information fra en hjemmeside.

Hvorfor web scraping?

- Web scraping bruges, når der ikke er en API til rådighed.

Hvordan fungerer det?

- Når du besøger en hjemmeside laver du via din browser en GET-request, og du får et HTML objekt tilbage som response. HTML er også et markup-sprog med tags, som vi kan bruge til at identificere den information, vi vil ekstrahere.



# Web scraping

Fremgangsmåde v. brug af `rvest`

1. Indlæs hjemmeside af interesse med `read_html()`
2. Find elementer af interesse, fx via:
  - "Inspect" eller tilsvarende i din browser
  - Med en selektor i din browser, fx `SelectorGadget` til Chrome
  - Sidens kildekode
3. Ekstraher elementer af interesse med `html_nodes()`
4. Konverter elementerne med `html_table()` og `html_text()`

# Eksempel

## Scraping af aktiekurser med pakken `rvest`

```
# Definerer hjemmeside vi vil scrape fra
url <- "https://npinvestor.dk/aktier-og-kurslister/aktier/danmark/alle-danske-aktier"

# Indlæser html-objekt og ekstraherer teksten fra elementer med klassen .float-columns
url %>%
  read_html() %>%
  html_nodes(css = ".float-columns") %>%
  html_text()
```

```
## [1] "Navn KursLavesteHøjeste+/-+/-(% )Tid"
## [2] "A.P. Møller - Mærsk A  9.260,009.225,009.300,0060,000,65%09:42:32"
## [3] "A.P. Møller - Mærsk B  9.636,009.602,009.684,0076,000,79%09:42:32"
## [4] "AaB  194,00190,00194,000,000,00%05/03/18"
## [5] "Aarhus Elite B A/S  0,000,000,000,000,00%23/01/18"
## [6] "Admiral Capital A/S B  1,671,671,700,000,00%02/03/18"
## [7] "ALK-Abello B  732,00729,00740,00-6,00-0,81%09:42:12"
## [8] "Alm. Brand  65,7065,3065,900,801,23%09:41:44"
## [9] "Ambu A/S  114,70114,30115,600,200,17%09:42:31"
## [10] "Andersen & Martini B  61,0061,0061,002,003,39%09:17:47"
```

Opgave-session

# Opgave-session

Vælg mellem:

1. Sidde selv og generere ideer til/arbejde på egen seminar-opgave
2. Sidde sammen og generere idéer

Opsamling og næste gang

# Vigtigste pointer fra i dag

- Nogle API'er kan benyttes med klienter
- Andre må vi selv lære at bruge - fx med `http`
- JSON-filer, arrays og objekter
- XML-filer og tags
- Scraping med `requests`

# Næste gang

- Indhold:
  - Tekst som data
- Pensum:
  - Grimmer & Stewart (2013) om tekst som data - læses
  - Wickham (2010) om `stringr` - læses
- DataCamp:
  - Sentiment Analysis in R: The Tidy Way

Tak for i dag!