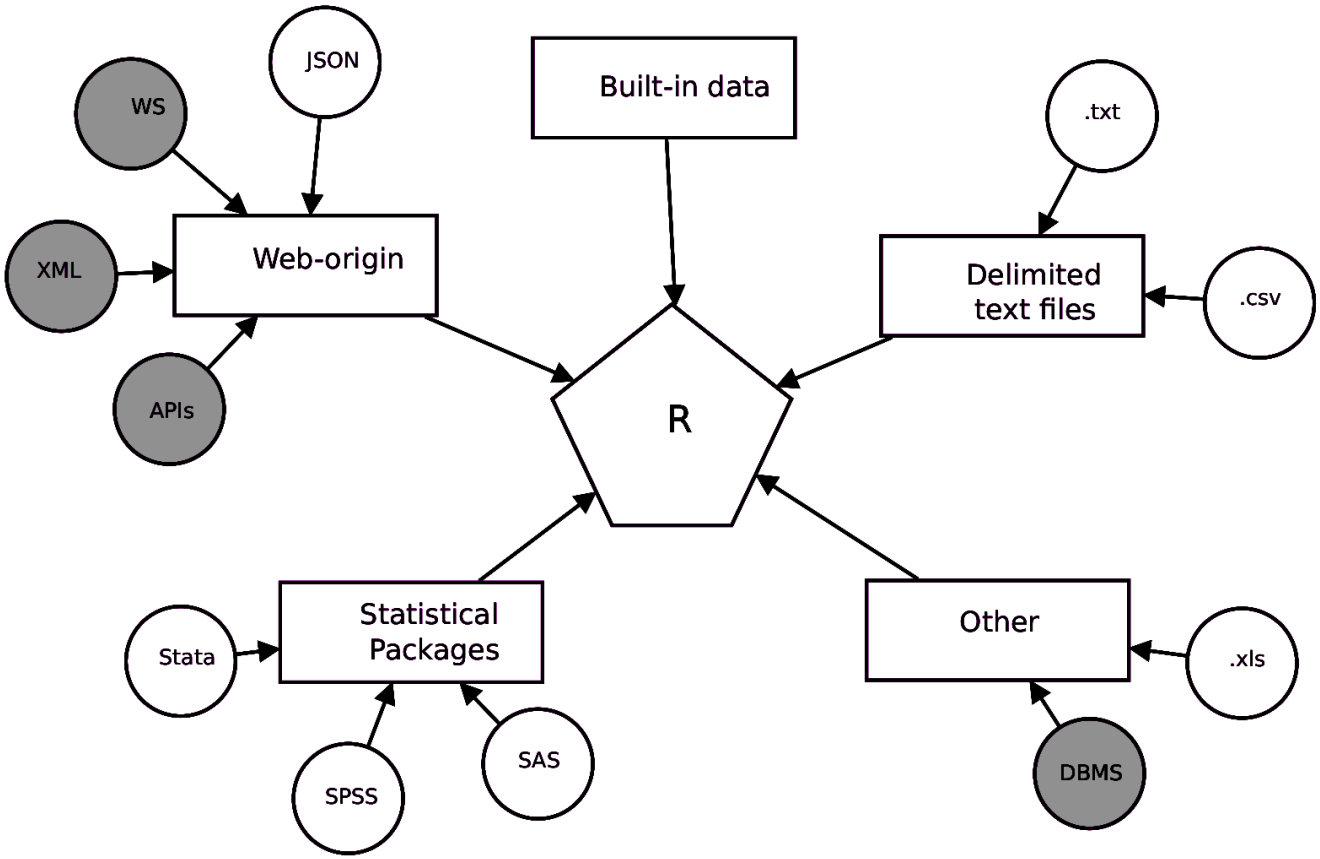
Statystyczna Analiza Danych – laboratorium

Wczytywanie zbiorów danych do R

Dorota Celińska-Kopczyńska

Uniwersytet Warszawski

Nad czym pracujemy



Konwencja

- W kodzie na niebiesko zaznaczone są miejsca, w które można wprowadzać własną zawartość
- Systemy UNIX korzystają z "/" jako separatora ścieżki a DOS korzysta z "\". W tej prezentacji, ścieżki zawierają "/", o ile nie zaznaczono inaczej

Praca z danymi wbudowanymi

- Pakiet `dataset` dane wbudowane – dostępne zawsze w R
- `data()` zwraca listę tych zbiorów danych
- Przykład wczytania: `data(iris)`
- Jeśli nie zostanie podana nowa nazwa dla zbioru, to zbiór otrzyma domyślną nazwę (np. iris)

Przykład pliku .txt lub .csv

- Dane zawarte w tabeli, pierwszy wiersz może zawierać nazwy zmiennych (**header** – nagłówek)
- Ważne: **separator** (znak oddzielający kolumny) i **quote char** (znak cytowania, jeśli napis zawiera separator)

```
id,name,edu,age,color
1,Adam,3,23,"red"
2,Beth,1,16,"green, yellow"
3,Celine,2,18,"blue"
4,David,4,30,"violet"
```

utils::read.table()

- `read.table()` wczytuje dane w **formacie tabeli** i zapisuje je jako **data frame**.
- Główny sposób wczytywania danych tabelarycznych do R

```
objectname <- read.table(‘‘plik’’, header=wartosc-logical, sep=’’character’’,
                         dec=’’character’’, quote=’’character’’, as.is=wartosc-logical,
                         na.strings=’’character’’, comment.char=’’character’’,
                         stringsAsFactors=wartosc-logical)
```

`read.table()`: argumenty

Argument	Opis	Domyślnie
file	nazwa pliku	brak – konieczne podanie
header	czy plik zawiera nazwy kolumn w pierwszym wierszu	FALSE
sep	znak separatora kolumn	sep = "" (whitespace)
dec	znak przecinka dziesiętnego	"."
quote	znaki używane do cytowania (kontekstu)	"\"'"
as.is	czy napisy mają pozostać napisami?	!stringsAsFactors
na.strings	wektor znaków, które mają być interpretowane jako brak danych (wartości NA)	"NA"
comment.char	znak, który oznacza początek komentarza	"#"
stringsAsFactors	czy napisy mają być konwertowane na factor?	TRUE

`read.table()`: trivia

- Nie trzeba podawać wartości dla wszystkich argumentów, jeśli domyślne są wystarczające
- Domyślnie kolumny zawierające napisy będą konwertowane na factor! Aby to zmienić albo należy użyć jednego z argumentów, albo później samodzielnie zmienić interpretację wybranych kolumn (`as.character()`).
- `tidyverse::tibble` i `data.table::data.table` nie konwertują domyślnie napisów na factor
- Można podać url do pliku, niekoniecznie ścieżkę na dysku
- Więcej specjalistycznych argumentów w manualu: `https://www.rdocumentation.org/packages/utils/versions/3.5.1/topics/read.table`

Inne funkcje do wczytania danych tabelarycznych

- `utils` zawiera dodatkowe funkcje o tej samej funkcjonalności co `read.table()` **ale z innymi domyślnymi wartościami**

Function	header	sep	dec	comment.char
`read.csv()`	TRUE	","	"."	""
`read.csv2()`	TRUE	";"	","	""
`read.delim()`	TRUE	"\t"	"."	""
`read.delim2()`	TRUE	"\t"	","	""

pliki .xlsx: prosty sposób dla większości zastosowań

- Zapisz plik .xlsx w formacie .csv i wczytaj przez `read.table()` lub podobne
- Jeśli plik zawiera więcej zakładek, to ta metoda nie będzie efektywna – zakładki zapisujemy pojedynczo

Inne metody dla plików .xlsx

- `openxlsx::read.xlsx`
- `gdata::read.xls`
- `readxl::read_excel`
- `RODBC::odbcConnectExcel()` + `sqlQuery()` (praca jak z bazą danych!)

```
# Skladnia openxlsx::read.xlsx
library(openxlsx)
xlsfile <- read.xlsx("plik", sheet = 1, colNames = T)

# Skladnia gdata::read.xls()
# Uwaga! Czasem wymaga podania sciezki do pliku wykonywalnego perla
library(gdata)
xlsfle <- read.xls("file", sheet=number)

# Skladnia readxl::read_excel
library(readxl)
xlsfile <- read_excel("file", range = anchored("cell-num", dim = vector), col_names = FALSE)

# Skladnia RODBC
library("RODBC")
xlsfile <- odbcConnectExcel("file")
sqlQuery(xlsfile, "select * from \" Sheet1 \"")
```

Przydatne funkcje dla danych z pakietów statystycznych

- Większość tych zbiorów danych jest przechowywana jako binarki – nie jest to format łatwy do odczytania

Źródło	funkcja	pakiet
SPSS	`read.spss()`	foreign
SPSS	`spss.get()`	Hmisc
SPSS	`read_sav()`	haven
SAS	`read.ssd()`	foreign
SAS	`sas.get()`	Hmisc
SAS	`read_sas()`	haven
Stata	`read.dta()`	foreign
Stata	`read_dta()`	haven

Pozyskiwanie danych z Internetu

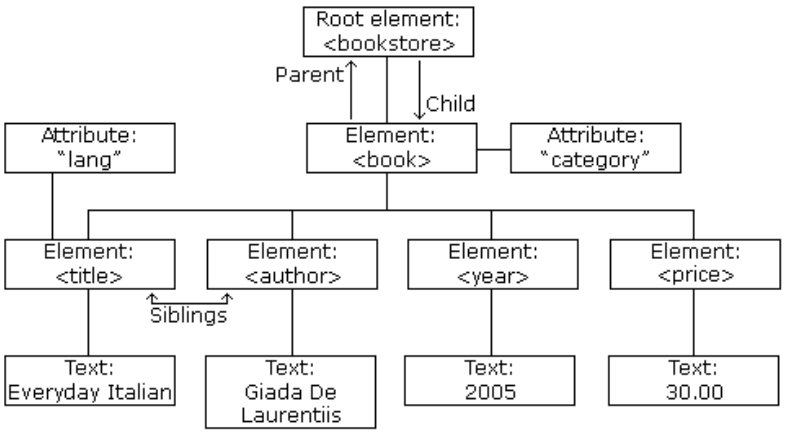
- Webscraping to proces pozyskiwania informacji ze stron internetowych
- Do R można wczytać dane bezpośrednio ze strony internetowej (pakiet `rvest`
- Tutaj przyjrzymy się ogólnym formatom plików, które mogą pojawić się przy webscrapingu

API

- API zapewniają narzędzia, które pozwalają programistom połączyć swoje oprogramowanie z "czymś jeszcze"
- Popularne API mają swoje pakiety, np. `twitteR`
- Brak ogólnych zasad – przeczytaj manual dla konkretnego API
- API czasem zwracają dane w formacie XML lub JSON

Przykład pliku XML

- XML to język znaczników przedstawiający dane w postaci struktury drzewiastej



Source: http://www.w3schools.com/xml/xml_tree.asp

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="fantasy">
    <title lang="en">Silmarillion</title>
    <author>J. R. R. Tolkien</author>
    <year>2005</year>
    <price>23.99</price>
  </book>
  <book category="science">
    <title lang="pl">Ekonometria</title>
    <author>Jerzy Mycielski</author>
    <year>2010</year>
    <price>20.00</price>
  </book>
</bookstore>
```

Przykład pliku JSON

- JSON reprezentuje dane jako krotki par atrybut-wartość zgromadzone w tablicach

```
[
  {
    "login": "octocat",
    "url": "https://api.github.com/users/octocat",
    "html_url": "https://github.com/octocat",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "eryxyre",
    "url": "https://api.github.com/users/eryxyre",
    "html_url": "https://github.com/eryxyre",
    "type": "User",
    "site_admin": false
  },
]
```

Wczytanie JSON lub XML do R

```
library(jsonlite)
mydf <- fromJSON("json-file.json")

# konwersja z powrotem do JSON
toJSON(mydf, pretty=TRUE)

# praca z plikami XML
library(XML)
doc <- xmlParse("xmlfile.xml")
mydf <- xmlToDataFrame(nodes=getNodeSet(doc,"root"))[c(vector)])

# lub z wykorzystaniem XPaths
```

Zapis do pliku

- `base::save()`
- `utils::write.table()`

```
write.table(obiekt, file = "plik", append = FALSE, quote = TRUE, sep = " ",
        eol = "", na = "NA", dec = ".", row.names = TRUE,
        col.names = TRUE, fileEncoding = "")

write.csv(...)  # wartosci domyslne jak w read.csv()
write.csv2(...) # wartosci domyslne jak w read.csv2()
```