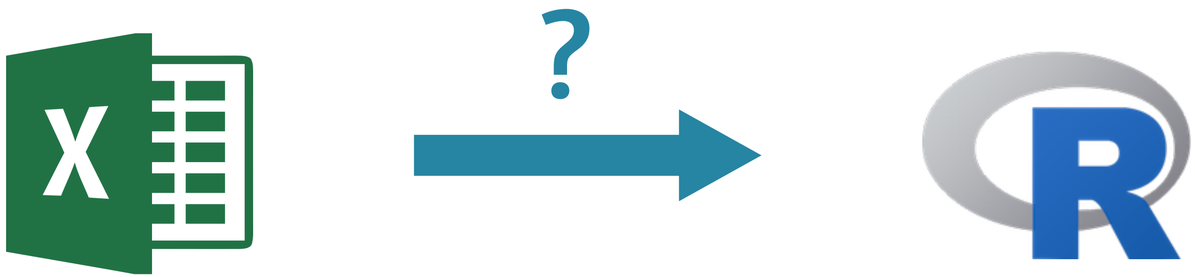
IMPORTING DATA IN R

Introduction read.csv

Importing data in R



5 types

- Flat files
- Data from Excel
- Databases
- Web
- Statistical software

Flat Files

states.csv

Comma Separated Values

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

Field names

?

```
> wanted_df
          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2      New York     Albany   19.746    54555
3        Oregon      Salem    3.970    98381
4       Vermont Montpelier    0.627     9616
5        Hawaii   Honolulu    1.420    10931
```

utils - read.csv

- Loaded by default when you start R

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

```
> read.csv("states.csv", stringsAsFactors = FALSE)
```

Import strings as categorical variables?

What if file in datasets folder of home directory?

```
> path <- file.path("~", "datasets", "states.csv")

> path
[1] "~/datasets/states.csv"

> read.csv(path, stringsAsFactors = FALSE)
```

read.csv()

```
> read.csv("states.csv", stringsAsFactors = FALSE)

          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931

> df <- read.csv("states.csv", stringsAsFactors = FALSE)
> str(df)
'data.frame':5 obs. of  4 variables:
 $ state   : chr  "South Dakota" "New York" "Oregon" "Vermont" ...
 $ capital : chr  "Pierre" "Albany" "Salem" "Montpelier" ...
 $ pop_mill: num  0.853 19.746 3.97 0.627 1.42
 $ area_sqm: int  77116 54555 98381 9616 10931
```

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

read.delim
read.table

Tab-delimited file

states.txt

```
state   capital pop_mill    area_sqm
South Dakota    Pierre  0.853   77116
New York    Albany  19.746  54555
Oregon  Salem  3.970   98381
Vermont Montpelier  0.627   9616
Hawaii  Honolulu    1.420   10931
```

```
> read.delim("states.txt", stringsAsFactors = FALSE)
         state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

Exotic file format

states2.txt

```
state/capital/pop_mill/area_sqm
South Dakota/Pierre/0.853/77116
New York/Albany/19.746/54555
Oregon/Salem/3.970/98381
Vermont/Montpelier/0.627/9616
Hawaii/Honolulu/1.420/10931
```

read.table()

- Read any tabular file as a data frame
- Number of arguments is huge

states2.txt

```
state/capital/pop_mill/area_sqm
South Dakota/Pierre/0.853/77116
New York/Albany/19.746/54555
Oregon/Salem/3.970/98381
Vermont/Montpelier/0.627/9616
Hawaii/Honolulu/1.420/10931
```

```
> read.table("states2.txt",
             header = TRUE,
             sep = "/",
             stringsAsFactors = FALSE)

          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

first row lists variable names (default FALSE)

field separator is a forward slash

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Final thoughts

Wrappers

- read.table() is the main function
- read.csv() = wrapper for CSV
- read.delim() = wrapper for tab-delimited files

read.csv

- Defaults
  - header = TRUE
  - sep = ","

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

```
> read.table("states.csv", header = TRUE, sep = ",",
             stringsAsFactors = FALSE)

> read.csv("states.csv", stringsAsFactors = FALSE)
```

read.delim

- Defaults
  - header = TRUE
  - sep = "\t"

states.txt

```
state   capital pop_mill    area_sqm
South Dakota    Pierre  0.853   77116
New York    Albany  19.746  54555
Oregon  Salem  3.970   98381
Vermont Montpelier  0.627   9616
Hawaii  Honolulu    1.420   10931
```

```
> read.table("states.txt", header = TRUE, sep = "\t",
             stringsAsFactors = FALSE)

> read.delim("states.txt", stringsAsFactors = FALSE)
```

Documentation

```
> ?read.table
```

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Locale differences

states_aye.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

states_nay.csv

```
state;capital;pop_mill;area_sqm
South Dakota;Pierre;0,853;77116
New York;Albany;19,746;54555
Oregon;Salem;3,97;98381
Vermont;Montpelier;0,627;9616
Hawaii;Honolulu;1,42;10931
```

Locale differences

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)
```

```
read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

states_nay.csv

```
> read.csv("states_nay.csv", stringsAsFactors = FALSE)
                      state.capital.pop_mill.area_sqm
South Dakota;Pierre;0                        853;77116
New York;Albany;19                           746;54555
Oregon;Salem;3                                97;98381
Vermont;Montpelier;0                          627;9616
Hawaii;Honolulu;1                              42;10931

> read.csv2("states_nay.csv", stringsAsFactors = FALSE)
         state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

states_nay.csv

```
state;capital;pop_mill;area_sqm
South Dakota;Pierre;0,853;77116
New York;Albany;19,746;54555
Oregon;Salem;3,97;98381
Vermont;Montpelier;0,627;9616
Hawaii;Honolulu;1,42;10931
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

readr
read_csv & read_tsv

Overview

- Before: utils package
- Specific R packages
  - readr
  - data.table

readr

- Hadley Wickham
- Fast, easy to use, consistent
- utils: verbose, slower

```
> install.packages("readr")
> library(readr)
```

CSV files

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

```
> read.csv("states.csv", stringsAsFactors = FALSE)
          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931

> read_csv("states.csv")
# A tibble: 5 × 4
          state    capital pop_mill area_sqm
          <chr>      <chr>    <dbl>    <int>
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

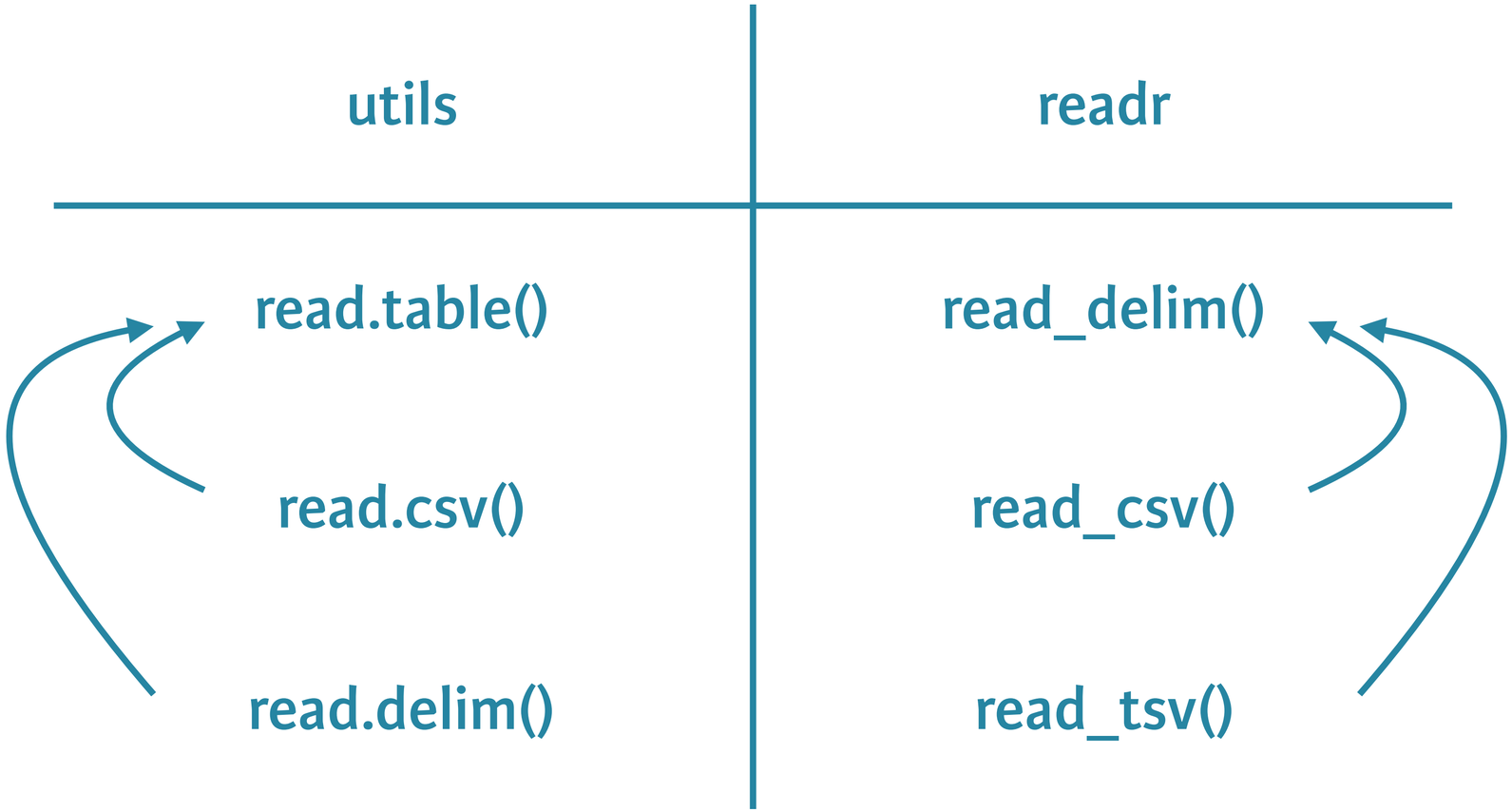
TSV files

states.txt

```
state   capital pop_mill    area_sqm
South Dakota    Pierre  0.853   77116
New York    Albany  19.746  54555
Oregon  Salem  3.970   98381
Vermont Montpelier  0.627   9616
Hawaii  Honolulu    1.420   10931
```

```
> read.delim("states.txt", stringsAsFactors = FALSE)
          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931

> read_tsv("states.txt")
# A tibble: 5 × 4
          state    capital pop_mill area_sqm
          <chr>      <chr>    <dbl>    <int>
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

Wrapping in utils and readr



IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

readr
read_delim

states2.txt

```
> read.table("states2.txt", header = TRUE, sep = "/",
             stringsAsFactors = FALSE)
          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931

> read_delim("states2.txt", delim = "/")
# A tibble: 5 x 4
          state    capital pop_mill area_sqm
          <chr>      <chr>    <dbl>    <int>
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

col_names

col_types

states2.txt

```
state/capital/pop_mill/area_sqm
South Dakota/Pierre/0.853/77116
New York/Albany/19.746/54555
Oregon/Salem/3.970/98381
Vermont/Montpelier/0.627/9616
Hawaii/Honolulu/1.420/10931
```

col_names

```
> read_delim("states3.txt", delim = "/", col_names = FALSE)
           X1          X2     X3    X4
        <chr>       <chr>  <dbl> <int>
1 South Dakota     Pierre  0.853 77116
2     New York     Albany 19.746 54555
3       Oregon      Salem  3.970 98381
4      Vermont Montpelier  0.627  9616
5       Hawaii   Honolulu  1.420 10931

> read_delim("states3.txt", delim = "/",
             col_names = c("state", "city", "pop", "area"))
       state       city    pop  area
        <chr>      <chr>  <dbl> <int>
1 South Dakota     Pierre  0.853 77116
2     New York     Albany 19.746 54555
3       Oregon      Salem  3.970 98381
4      Vermont Montpelier  0.627  9616
5       Hawaii   Honolulu  1.420 10931
```

states3.txt

```
South Dakota/Pierre/0.853/77116
New York/Albany/19.746/54555
Oregon/Salem/3.970/98381
Vermont/Montpelier/0.627/9616
Hawaii/Honolulu/1.420/10931
```

col_types

states2.txt

```
state/capital/pop_mill/area_sqm
South Dakota/Pierre/0.853/77116
New York/Albany/19.746/54555
Oregon/Salem/3.970/98381
Vermont/Montpelier/0.627/9616
Hawaii/Honolulu/1.420/10931
```

```
> read_delim("states2.txt", delim = "/")
         state    capital pop_mill area_sqm
         <chr>      <chr>    <dbl>    <int>
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931

> read_delim("states2.txt", delim = "/", col_types = "ccdd")
         state    capital pop_mill area_sqm
         <chr>      <chr>    <dbl>    <dbl>
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

c = character
d = double
i = integer
l = logical
_ = skip

skip and n_max

```
> read_delim("states2.txt", delim = "/",
             skip = 2, n_max = 3)
# A tibble: 3 x 4
  New York     Albany 19.746 54555
     <chr>      <chr>  <dbl> <int>
1   Oregon      Salem  3.970 98381
2  Vermont Montpelier  0.627  9616
3   Hawaii   Honolulu  1.420 10931

> read_delim("states2.txt", delim = "/",
             col_names = c("state", "city", "pop", "area"),
             skip = 2, n_max = 3)
# A tibble: 3 x 4
     state        city    pop  area
     <chr>      <chr>  <dbl> <int>
1 New York     Albany 19.746 54555
2   Oregon      Salem  3.970 98381
3  Vermont Montpelier  0.627  9616
```

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

data.table: fread

data.table

- Matt Dowle & Arun Srinivasan
- Key metric: speed
- Data manipulation in R
- Function to import data: fread()

```
> install.packages("data.table")
> library(data.table)
```

- Similar to read.table()

fread()

states.csv

```
state,capital,pop_mill,area_sqm
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

states2.csv

```
South Dakota,Pierre,0.853,77116
New York,Albany,19.746,54555
Oregon,Salem,3.970,98381
Vermont,Montpelier,0.627,9616
Hawaii,Honolulu,1.420,10931
```

```
> fread("states.csv")
          state     capital pop_mill area_sqm
1: South Dakota      Pierre    0.853    77116
2:     New York      Albany   19.746    54555
3:       Oregon       Salem    3.970    98381
4:      Vermont Montpelier    0.627     9616
5:       Hawaii    Honolulu    1.420    10931
```

```
> fread("states2.csv")
             V1          V2     V3    V4
1: South Dakota      Pierre  0.853 77116
2:     New York      Albany 19.746 54555
3:       Oregon       Salem  3.970 98381
4:      Vermont Montpelier  0.627  9616
5:       Hawaii    Honolulu  1.420 10931
```

fread()

- Infer column types and separators
- It simply works
- Extremely fast
- Possible to specify numerous parameters
- Improved read.table()
- Fast, convenient, customizable

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

readxl (1)

Microsoft Excel

- Common data analysis tool
- Many R packages to interact with Excel
- readxl - Hadley Wickham

Typical Structure Excel Data

Different sheets with tabular data

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

Source: Wikipedia

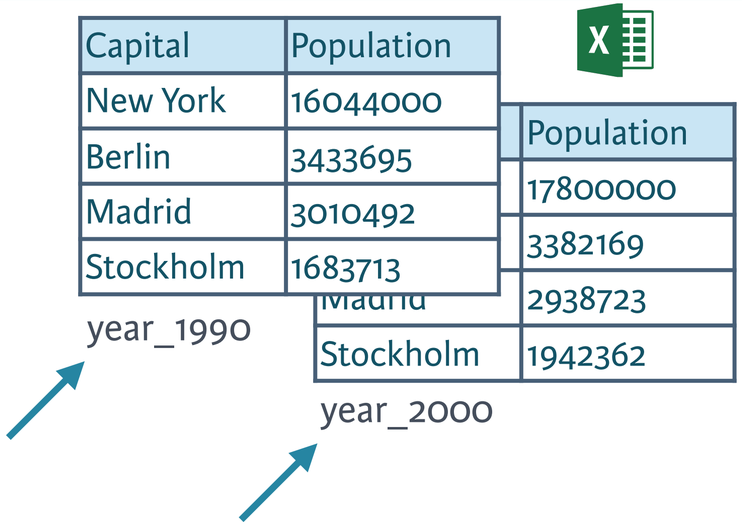
readxl

- excel_sheets() list different sheets
- read_excel() actually import data into R

```
> install.packages("readxl")
> library(readxl)
```

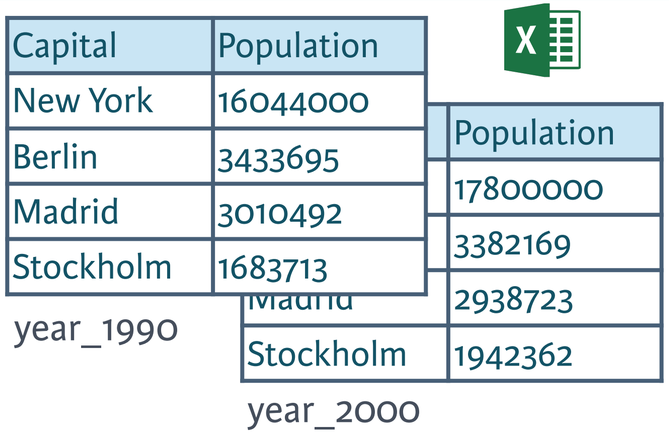
excel_sheets()

```
> dir()
[1] "cities.xlsx"  "the_rest_is_secret.txt"

> excel_sheets("cities.xlsx")
[1] "year_1990" "year_2000"
```



read_excel()

```
> read_excel("cities.xlsx")
# A tibble: 4 × 2
    Capital Population
      <chr>      <dbl>
1  New York   16044000
2    Berlin    3433695
3    Madrid    3010492
4 Stockholm    1683713

> read_excel("cities.xlsx", sheet = 2)
> read_excel("cities.xlsx", sheet = "year_2000")
# A tibble: 4 × 2
    Capital Population
      <chr>      <dbl>
1  New York   17800000
2    Berlin    3382169
3    Madrid    2938723
4 Stockholm    1942362
```



IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

readxl (2)

read_excel()

```
read_excel(path, sheet = 1,
           col_names = TRUE,
           col_types = NULL,
           skip = 0)
```

read_excel() - col_names

```
read_excel(path, sheet = 1,
           col_names = TRUE,
           col_types = NULL,
           skip = 0)
```

col_names = FALSE: R assigns names itself
col_names = character vector: manually specify

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

read_excel() - col_types

```
read_excel(path, sheet = 1,
           col_names = TRUE,
           col_types = NULL,
           skip = 0)
```

```
> read_excel("cities.xlsx", col_types = c("text", "text"))

# A tibble: 4 × 2
    Capital Population
      <chr>      <chr>
1  New York   16044000
2    Berlin    3433695
3    Madrid    3010492
4 Stockholm    1683713
```

numeric **date** **blank**

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

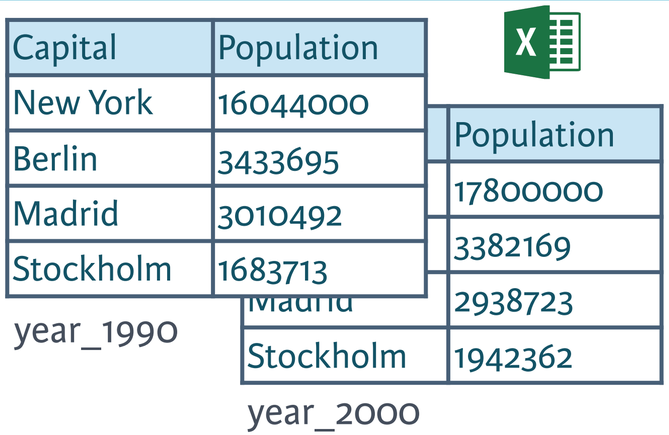
year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

read_excel() - col_types

```
read_excel(path, sheet = 1,
           col_names = TRUE,
           col_types = NULL,
           skip = 0)
```



Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

	Population
	17800000
	3382169
	2938723
Stockholm	1942362

year_2000

```
> read_excel("cities.xlsx",
             col_types = c("text", "blank"))

# A tibble: 4 × 1
    Capital
      <chr>
1  New York
2    Berlin
3    Madrid
4 Stockholm
```

read_excel() - skip

```
read_excel(path, sheet = 1,
           col_names = TRUE,
           col_types = NULL,
           skip = 0)
```

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

Population
17800000
3382169
2938723
1942362

year_2000

```
> read_excel("cities.xlsx",
             col_names = c("Capital", "Population"),
             skip = 2)

# A tibble: 3 × 2
    Capital Population
      <chr>      <dbl>
1    Berlin    3433695
2    Madrid    3010492
3 Stockholm    1683713
```

n_max not (yet) available

Wrap-up

- **excel_sheets()**
- **read_excel()**
- Everything you need!
- Fast
- Same arguments as in readr package
- Consistency

IMPORTING DATA IN R

Let’s practice!

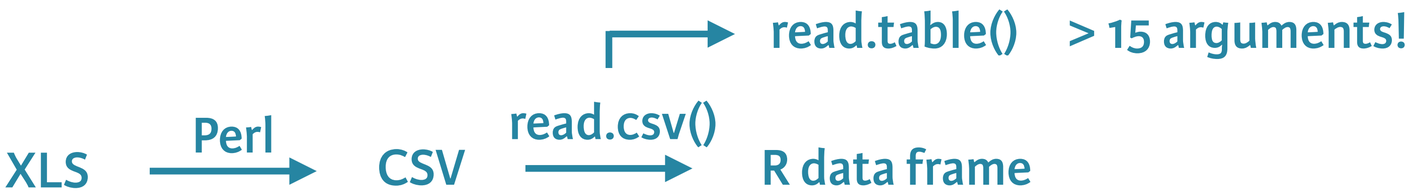
IMPORTING DATA IN R

gdata

gdata

- Gregory Warnes
- Entire suite of tools for data manipulation
- Supercharges basic R
- `read.xls()`
- Support for XLS
- Support for XLSX with additional driver
- No readxl::excel_sheets() equivalent

gdata



- Elegant extension of utils package
- Easy if familiar with utils
- Extremely inefficient
- readxl < v1.x

cities.xls

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

	Population
	17800000
Berlin	3382169
Madrid	2938723
Stockholm	1942362

year_2000

read.xls()

```
> install.packages("gdata")
> library(gdata)

> read.xls("cities.xls")

    Capital Population
1  New York   16044000
2    Berlin    3433695
3    Madrid    3010492
4 Stockholm    1683713

> read.xls("cities.xls", sheet = "year_2000")

    Capital Population
1  New York   17800000
2    Berlin    3382169
3    Madrid    2938723
4 Stockholm    1942362
```

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Reading sheets

XLConnect

- Martin Studer
- Work with Excel through R
- Bridge between Excel and R
- XLS and XLSX
- Easy-to-use functionality

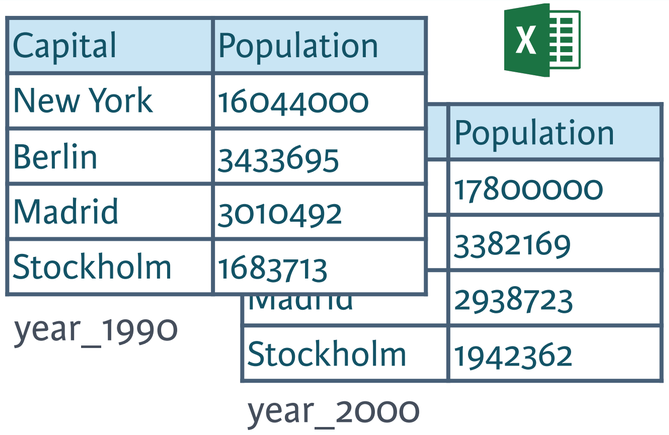
Installation

```
> install.packages("XLConnect")
also installing the dependencies 'XLConnectJars', 'rJava'
...
```

- Problems?
  - Install Oracle's Java Development Kit (JDK)
  - Google your error!

loadWorkbook()

```
> library("XLConnect")

> book <- loadWorkbook("cities.xlsx")

> str(book)
Formal class 'workbook' [package "XLConnect"]
with 2 slots
  ..@ filename: chr "cities.xlsx"
  ..@ jobj    : ...
```



Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

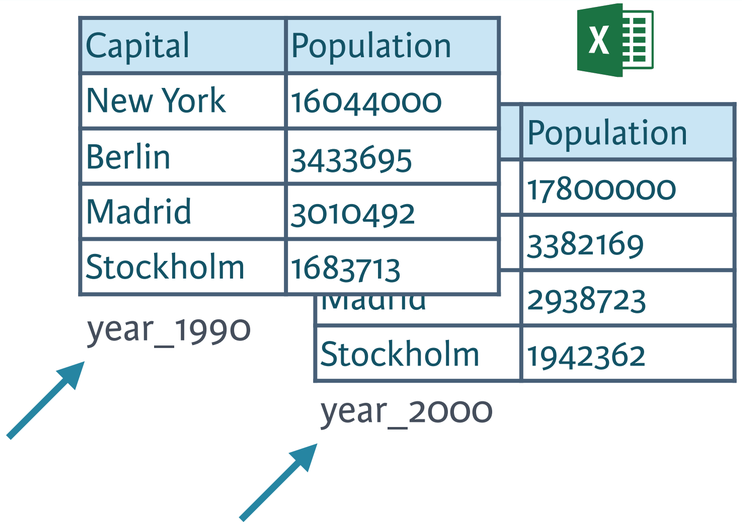
year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

getSheets()

```
> getSheets(book)
[1] "year_1990" "year_2000"

> library(readxl)
> excel_sheets("cities.xlsx")
[1] "year_1990" "year_2000"
```



readWorksheet()

Capital	Population
New York	16044000
Berlin	3433695
Madrid	3010492
Stockholm	1683713

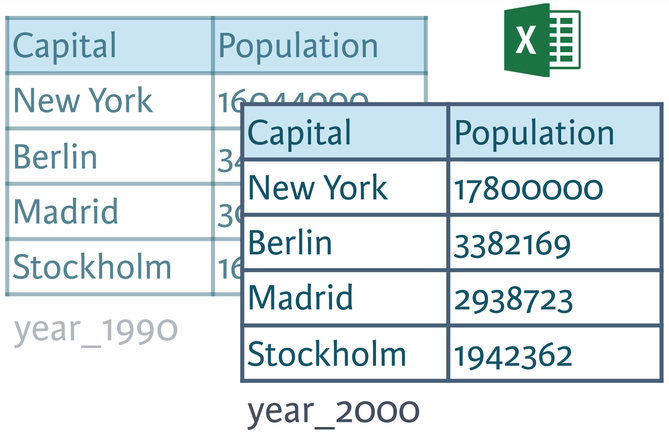
year_1990

	Population
	17800000
	3382169
Madrid	2938723
Stockholm	1942362

year_2000

readWorksheet()

```
> readWorksheet(book, sheet = "year_2000")

    Capital Population
1  New York   17800000
2    Berlin    3382169
3    Madrid    2938723
4 Stockholm    1942362
```



readWorksheet()

Capital	Population
New York	17800000
Berlin	**3382169**
Madrid	**2938723**
Stockholm	1942362

row 3

row 4

year_2000 **col 2**

```
> readWorksheet(book, sheet = "year_2000",
                startRow = 3, endRow = 4,
                startCol = 2, header = FALSE)

     Col1
1 3382169
2 2938723
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Adapting sheets

New data!

```
> pop_2010 <- data.frame(
    Capital = c("New York", "Berlin", "Madrid", "Stockholm"),
    Population = c(8191900, 3460725, 3273000, 1372565))

> pop_2010

    Capital Population
1  New York    8191900
2    Berlin    3460725
3    Madrid    3273000
4 Stockholm    1372565
```

createSheet()

```
> pop_2010 <- ... # truncated
> library(XLConnect)
> book <- loadWorkbook("cities.xlsx")
> createSheet(book, name = "year_2010")
```

Capital	Population
New York	16044000
Berlin	34
Madrid	30
Stockholm	16

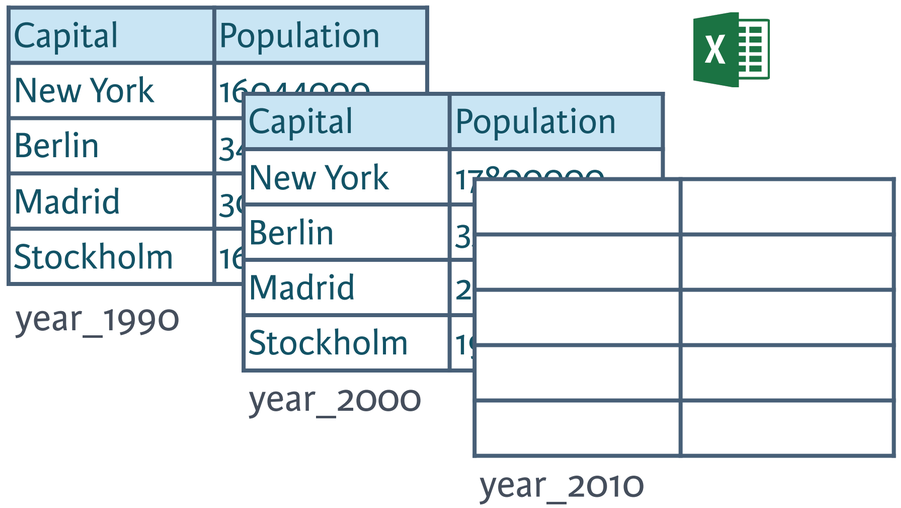
year_1990

Capital	Population
New York	17800000
Berlin	3382169
Madrid	2938723
Stockholm	1942362

year_2000

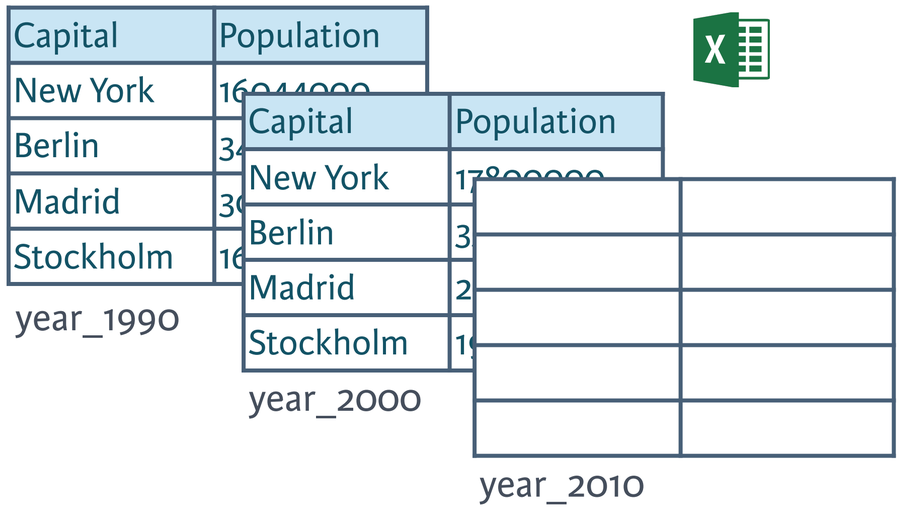
createSheet()

```
> pop_2010 <- ... # truncated
> library(XLConnect)
> book <- loadWorkbook("cities.xlsx")
> createSheet(book, name = "year_2010")
```



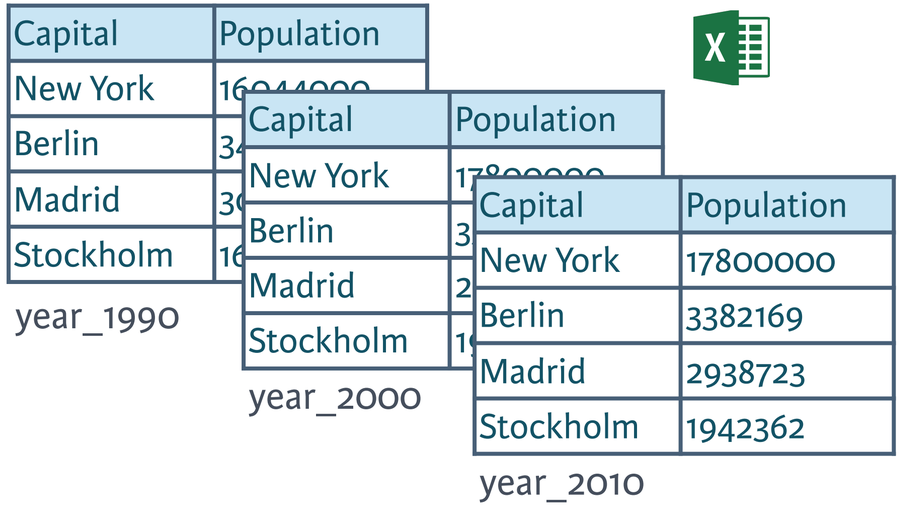
writeWorksheet()

```
> pop_2010 <- ... # truncated
> library(XLConnect)
> book <- loadWorkbook("cities.xlsx")
> createSheet(book, name = "year_2010")
> writeWorksheet(book, pop_2010, sheet = "year_2010")
```



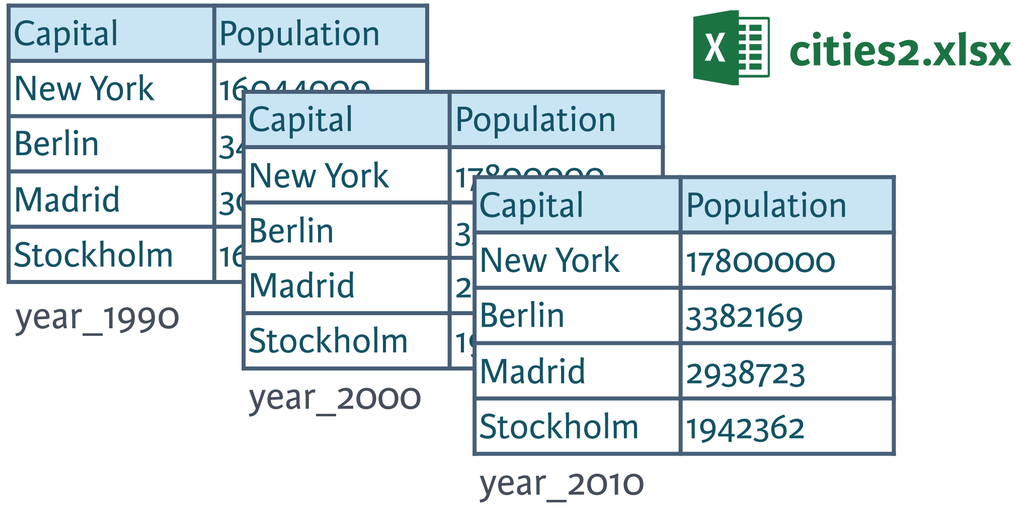
writeWorksheet()

```
> pop_2010 <- ... # truncated
> library(XLConnect)
> book <- loadWorkbook("cities.xlsx")
> createSheet(book, name = "year_2010")
> writeWorksheet(book, pop_2010, sheet = "year_2010")
```



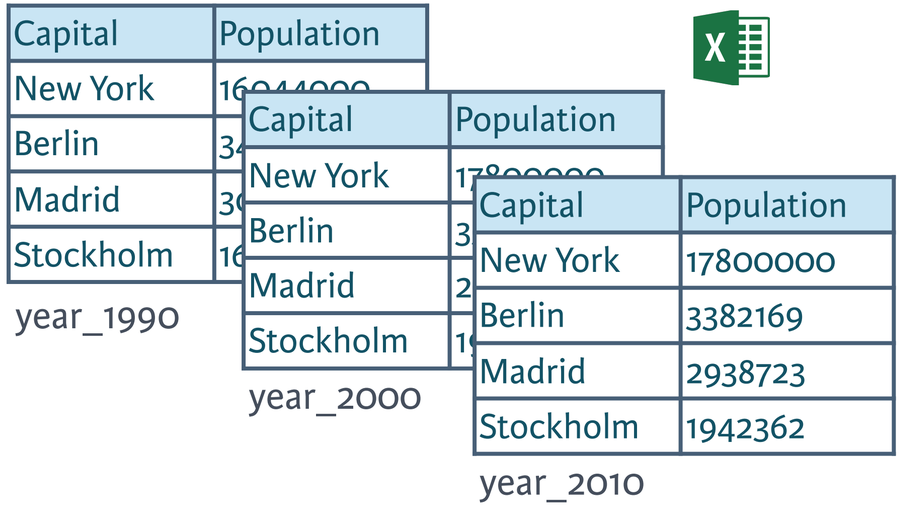
saveWorkbook()

```
> pop_2010 <- ... # truncated
> library(XLConnect)
> book <- loadWorkbook("cities.xlsx")
> createSheet(book, name = "year_2010")
> writeWorksheet(book, pop_2010, sheet = "year_2010")
> saveWorkbook(book, file = "cities2.xlsx")
```



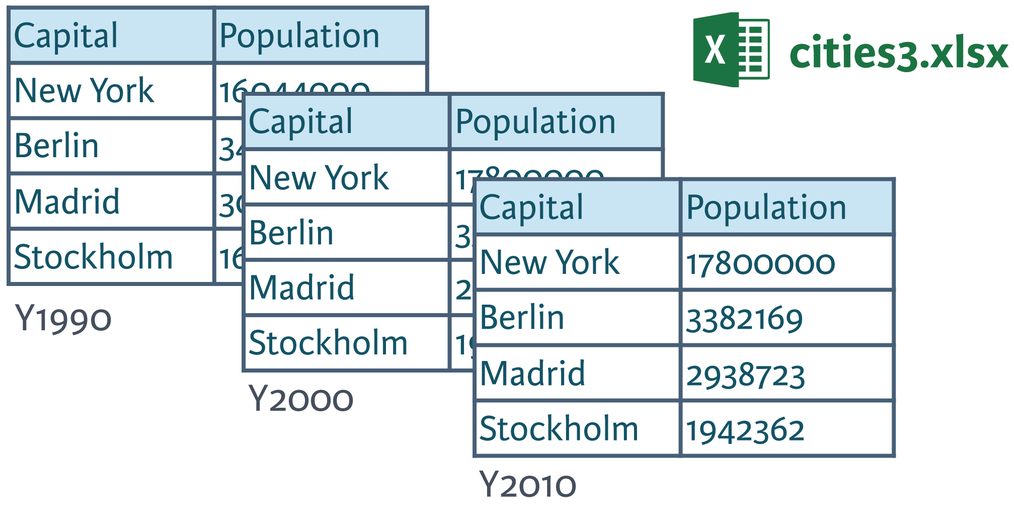
renameSheet()

```
> renameSheet(book, "year_1990", "Y1990")
> renameSheet(book, "year_2000", "Y2000")
> renameSheet(book, "year_2010", "Y2010")
> saveWorkbook(book, file = "cities3.xlsx")
```



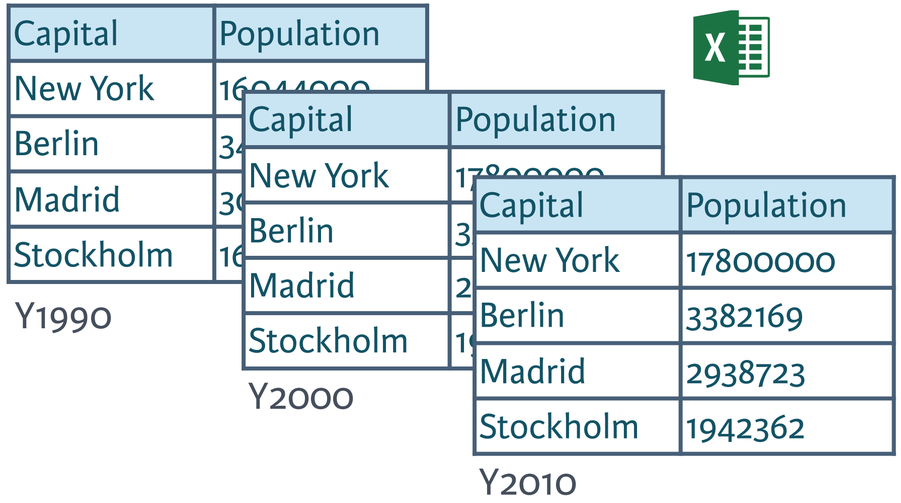
renameSheet()

```
> renameSheet(book, "year_1990", "Y1990")
> renameSheet(book, "year_2000", "Y2000")
> renameSheet(book, "year_2010", "Y2010")
> saveWorkbook(book, file = "cities3.xlsx")
```



removeSheet()

```
> removeSheet(book, sheet = "Y2010")
> saveWorkbook(book, file = "cities4.xlsx")
```



removeSheet()

```
> removeSheet(book, sheet = "Y2010")
> saveWorkbook(book, file = "cities4.xlsx")
```

Capital	Population
New York	16044000
Berlin	34
Madrid	30
Stockholm	16

Y1990

Capital	Population
New York	17800000
Berlin	3382169
Madrid	2938723
Stockholm	1942362

Y2000

cities4.xlsx

Wrap-up

- Basic operations
- Reproducibility is the key!
- More functionality
  - Styling cells
  - Working with formulas
  - Arranging cells
  - ...

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Connect to a database

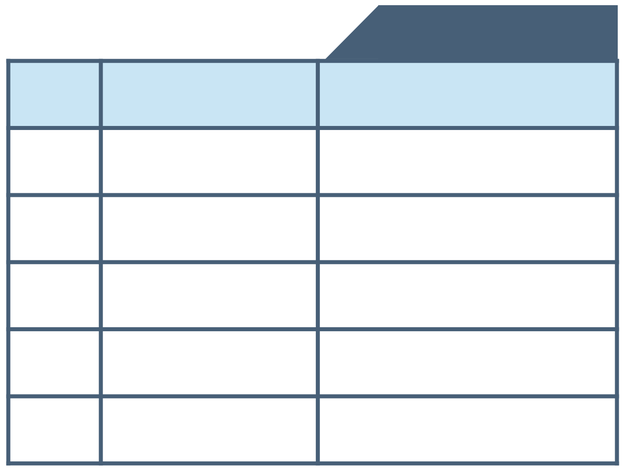
Up to now

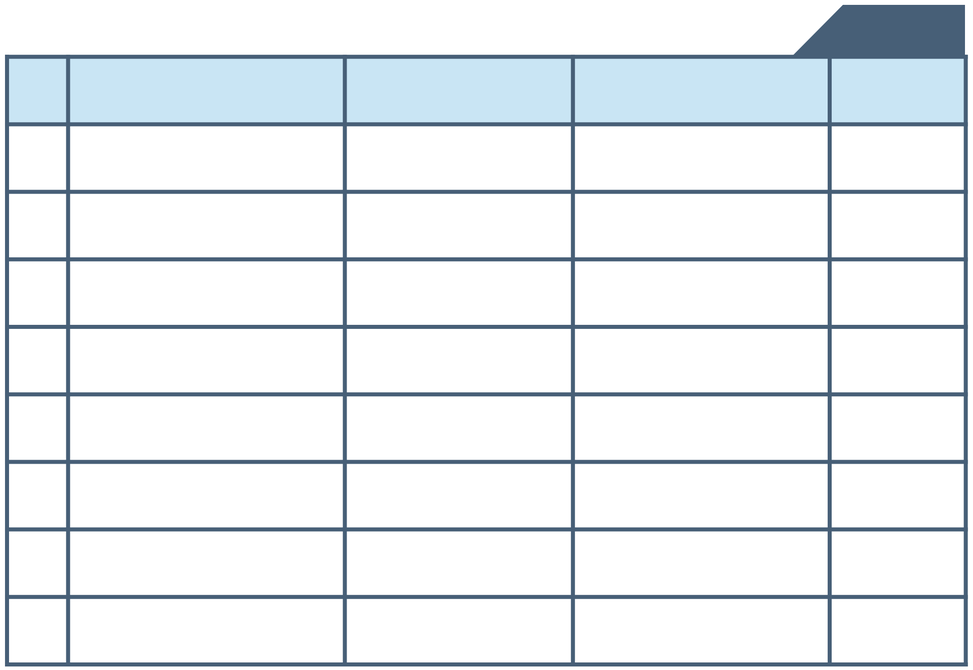
- Flat files
- Excel files

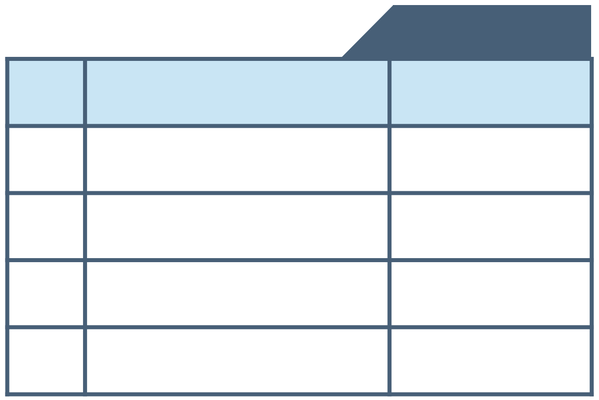
Relational Databases

- What is a relational database?
- How to connect?
- How to read table?

company



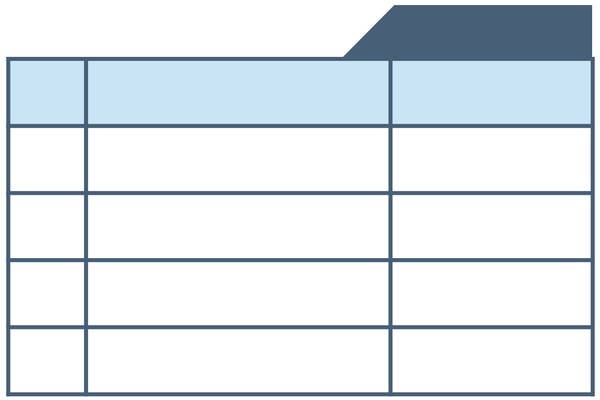


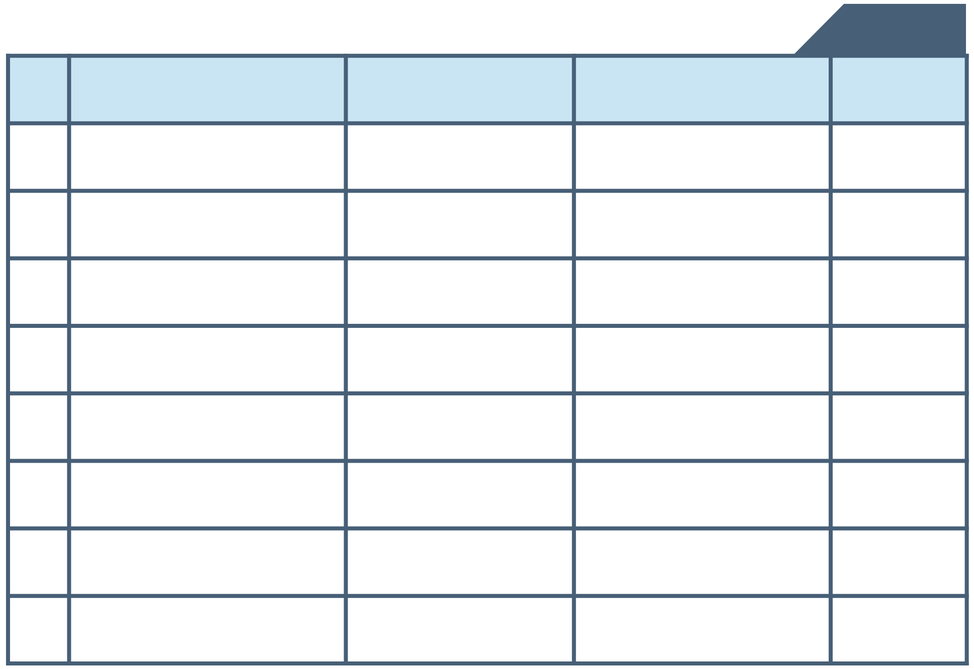


company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23





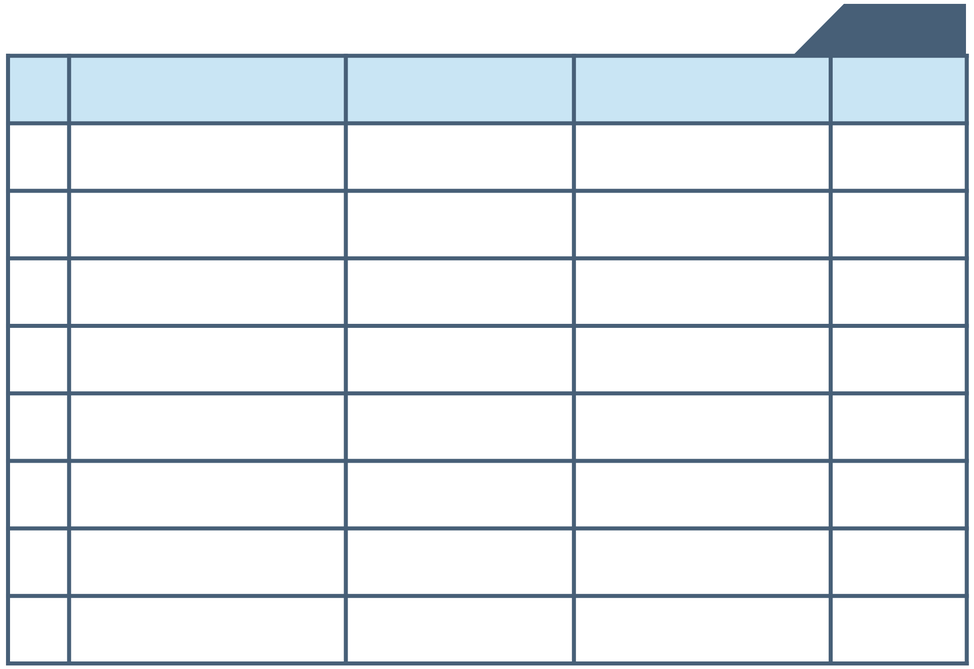
company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1



company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	6	9	2015-09-18	152
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	**Julie**	**2013-01-01**
7	Heather	2014-11-23
9	John	2014-11-23

relation

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	**6**	**9**	**2015-09-18**	**152**
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

products

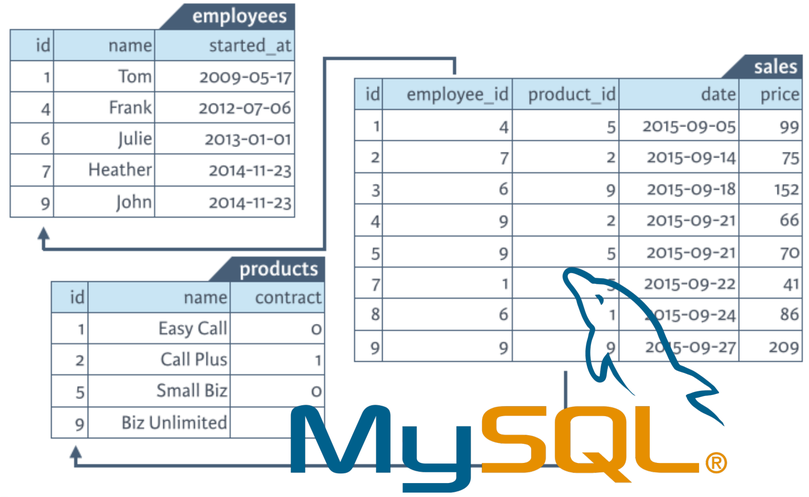
id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	**Biz Unlimited**	**1**

relation

Database Management System

- DBMS
- Open source
  - MySQL, PostgreSQL, SQLite
- Proprietary
  - Oracle Database, Microsoft SQL Server
- SQL = Structured Query Language

Databases in R



- Different R packages
- MySQL — RMySQL
- PostgresSQL — RPostgresSQL
- Oracle Database — ROracle
- Conventions specified in DBI

```
> install.packages("RMySQL")
> library(DBI)    # library(RMySQL) not required
```

Connect to database

Construct SQL driver

```
> con <- dbConnect(RMySQL::MySQL(),
           dbname = "company",
           host = "courses.csrrinzqubik.us-
                              east-1.rds.amazonaws.com",
           port = 3306,
           user = "student",
           password = "datacamp")
```

con is DBIConnection object

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Import table data

con

```
> con <- dbConnect(RMySQL::MySQL(),
           dbname = "company",
           host = "courses.csrrinzqubik.us-
                              east-1.rds.amazonaws.com",
           port = 3306,
           user = "student",
           password = "datacamp")
```

List and import tables

```
> dbListTables(con)
[1] "employees" "products"  "sales"

> dbReadTable(con, "employees")

  id    name  started_at
1  1     Tom  2009-05-17
2  4   Frank  2012-07-06
3  6   Julie  2013-01-01
4  7 Heather  2014-11-23
5  9    John  2015-05-12

> dbDisconnect(con)
[1] TRUE

> con
Error in .local(dbObj, ...) :
  internal error in RS_DBI_getConnection: ...
```

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

SQL Queries from inside R

Entire table

dbReadTable()

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

Fraction of data

?

employees

name	started_at
Julie	2013-01-01
John	2014-11-23

Selection

Selective importing

- SQL Queries
- DBI -> RMySQL, RPostgreSQL, ...
- Just the basics of SQL

company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	6	9	2015-09-18	152
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

company

Names of employees that started after 2012-09-01?

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	6	9	2015-09-18	152
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1

Load package and connect

```
> library(DBI)
> con <- dbConnect(RMySQL::MySQL(),
           dbname = "company",
           host = "courses.csrrinzqubik.us-
                              east-1.rds.amazonaws.com",
           port = 3306,
           user = "student",
           password = "datacamp")
```

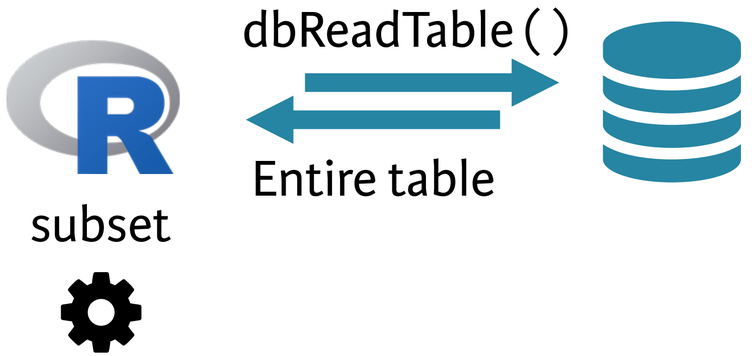
Example 1

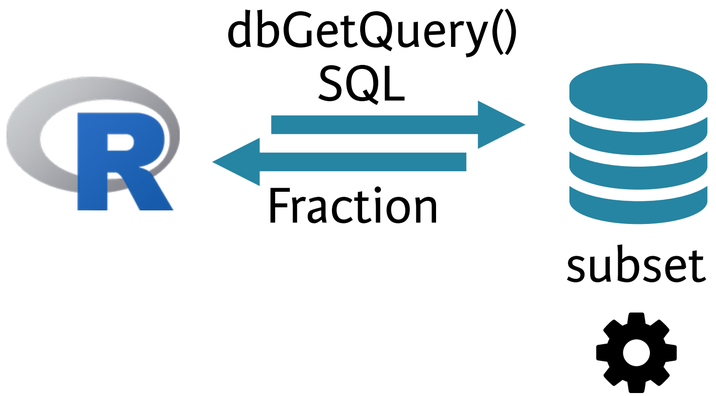
```
> employees <- dbReadTable(con, "employees")

> subset(employees,
         subset = started_at > "2012-09-01",
         select = name)
     name
3   Julie
4 Heather
5    John

> dbGetQuery(con, "SELECT name FROM employees
                   WHERE started_at > \"2012-09-01\"")
     name
1   Julie
2 Heather
3    John
```

Example 1

```
> employees <- dbReadTable(con, "employees")

> subset(employees,
         subset = started_at > "2012-09-01",
         select = name)
     name
3   Julie
4 Heather
5    John

> dbGetQuery(con, "SELECT name FROM employees
                      WHERE started_at > \"2012-09-01\"")
     name
1   Julie
2 Heather
3    John
```

Way more efficient for big databases!





company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	6	9	2015-09-18	152
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

company

employees

id	name	started_at
1	Tom	2009-05-17
4	Frank	2012-07-06
6	Julie	2013-01-01
7	Heather	2014-11-23
9	John	2014-11-23

products

id	name	contract
1	Easy Call	0
2	Call Plus	1
5	Small Biz	0
9	Biz Unlimited	1

All variables of products with contract

sales

id	employee_id	product_id	date	price
1	4	5	2015-09-05	99
2	7	2	2015-09-14	75
3	6	9	2015-09-18	152
4	9	2	2015-09-21	66
5	9	5	2015-09-21	70
7	1	5	2015-09-22	41
8	6	1	2015-09-24	86
9	9	9	2015-09-27	209

Example 2

```
> products <- dbReadTable(con, "products")


> subset(products, subset = contract == 1)
  id           name contract
2  2     Call Plus        1
4  9 Biz Unlimited        1


> dbGetQuery(con, "SELECT * FROM products
                     WHERE contract = 1")
  id           name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1
```

keep all columns

single equals sign

Example 2

```
> dbGetQuery(con, "SELECT * FROM products
                                    WHERE contract = 1")
  id           name contract
1  2      Call Plus        1
2  9 Biz Unlimited        1
```

dbGetQuery()

```
> dbGetQuery(con, "SELECT * FROM products
                                  WHERE contract = 1")
  id           name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1


> res <- dbSendQuery(con, "SELECT * FROM products
                                  WHERE contract = 1")

> dbFetch(res)
  id           name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1

> dbClearResult(res)
[1] TRUE
```

dbFetch() one by one

```
> res <- dbSendQuery(con, "SELECT * FROM products
                           WHERE contract = 1")

> while(!dbHasCompleted(res)) {
+   chunk <- dbFetch(res, n = 1)
+   print(chunk)
+ }

  id      name contract
1  2 Call Plus        1
  id          name contract
1  9 Biz Unlimited        1
[1] id       name     contract
<0 rows> (or 0-length row.names)

> dbClearResult(res)
[1] TRUE
```

Disconnect

```
> dbDisconnect(con)
[1] TRUE
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

DBI internals

dbGetQuery()

```
> dbGetQuery(con, "SELECT * FROM products
                           WHERE contract = 1")
  id           name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1


> res <- dbSendQuery(con, "SELECT * FROM products
                           WHERE contract = 1")

> dbFetch(res)
  id           name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1

> dbClearResult(res)
[1] TRUE
```

dbFetch() one by one

```
> res <- dbSendQuery(con, "SELECT * FROM products
                           WHERE contract = 1")

> while(!dbHasCompleted(res)) {
+   chunk <- dbFetch(res, n = 1)
+   print(chunk)
+ }

  id      name contract
1  2 Call Plus        1
  id          name contract
1  9 Biz Unlimited        1
[1] id       name     contract
<0 rows> (or 0-length row.names)

> dbClearResult(res)
[1] TRUE
```

Disconnect

```
> dbDisconnect(con)
[1] TRUE
```

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

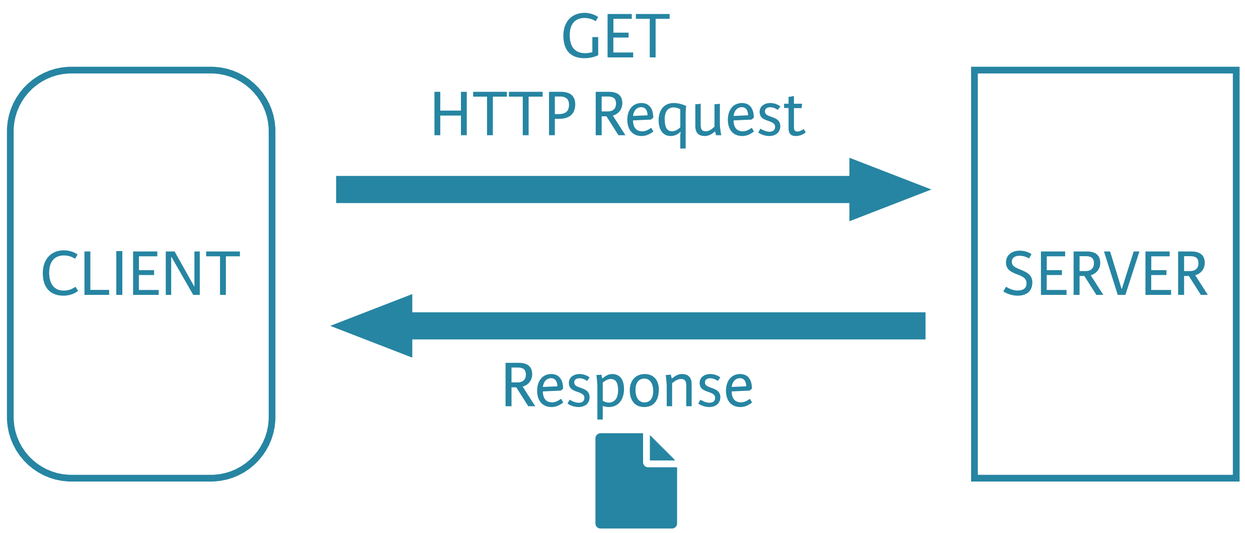
HTTP

Data on the web

- Already worked with it!
- Many packages handle it for you
- File formats useful for web technology
- JSON

HTTP

- HyperText Transfer Protocol
- Rules about data exchange between computers
- Language of the web



Example: CSV

http://s3.amazonaws.com/assets.datacamp.com/course/importing_data_into_r/states.csv

```
# Manually download file through web browser

> read.csv("path/to/states.csv")

          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

Example: CSV

```
> read.csv("http://s3.amazonaws.com/
                assets.datacamp.com/course/
                importing_data_into_r/states.csv")

          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

R sees it's a URL, does GET request, and reads in the CSV file

Example: CSV

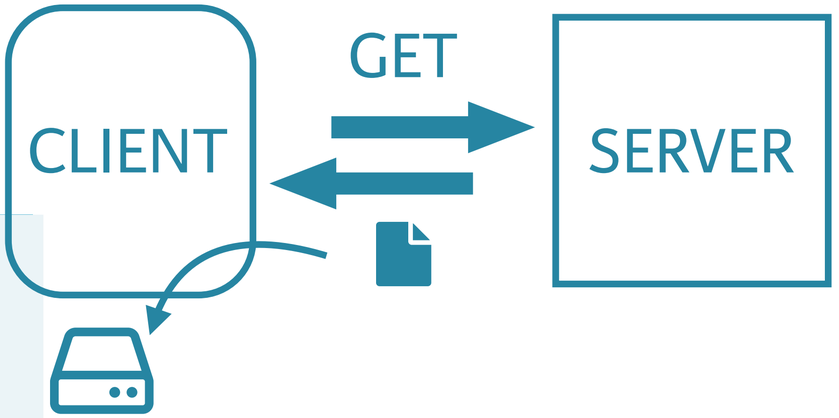
```
> read.csv("https://s3.amazonaws.com/
                assets.datacamp.com/course/
                importing_data_into_r/states.csv")

          state    capital pop_mill area_sqm
1 South Dakota     Pierre    0.853    77116
2     New York     Albany   19.746    54555
3       Oregon      Salem    3.970    98381
4      Vermont Montpelier    0.627     9616
5       Hawaii   Honolulu    1.420    10931
```

HTTPS support since R version 3.2.2

Example: Excel

```
> library(readxl)

> read_excel("http://s3.amazonaws.com/
                 assets.datacamp.com/course/
                 importing_data_into_r/cities.xlsx")
Error: 'http://s3.amazonaws.com/assets.datacamp.com/course/
importing_data_into_r/cities.xlsx' does not exist in current
working directory.
```

download.file()




```
> url <- "http://s3.amazonaws.com/assets.datacamp.com/
            course/importing_data_into_r/cities.xlsx"

> dest_path <- file.path("~", "local_cities.xlsx")

> download.file(url, dest_path)
// Messages showing download progress omitted

> read_excel(dest_path)
    Capital Population
1  New York   16044000
2    Berlin    3433695
3    Madrid    3010492
4 Stockholm    1683713
```

Why download.file()?

- Reproducibility
- ~~Point and click~~
- HTTP from inside R
  - Authentication
  - Additional parameters
  - httr - Hadley Wickham

IMPORTING DATA IN R

Let’s practice!

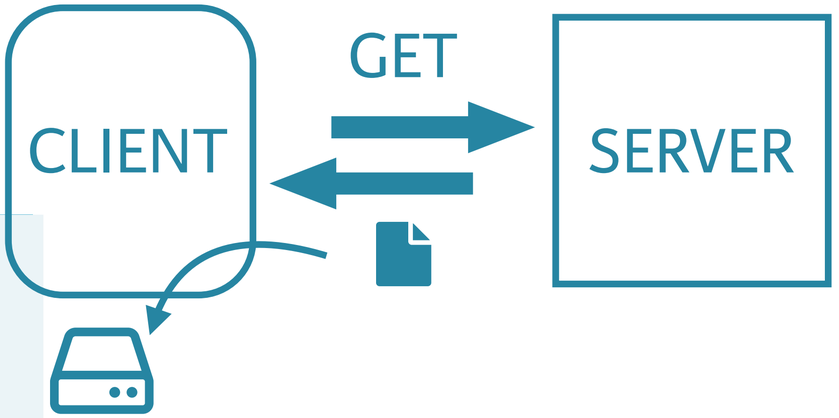
IMPORTING DATA IN R

Downloading files

Example: Excel

```
> library(readxl)

> read_excel("http://s3.amazonaws.com/
                 assets.datacamp.com/course/
                 importing_data_into_r/cities.xlsx")
Error: 'http://s3.amazonaws.com/assets.datacamp.com/course/
importing_data_into_r/cities.xlsx' does not exist in current
working directory.
```

download.file()




```
> url <- "http://s3.amazonaws.com/assets.datacamp.com/
            course/importing_data_into_r/cities.xlsx"

> dest_path <- file.path("~", "local_cities.xlsx")

> download.file(url, dest_path)
// Messages showing download progress omitted

> read_excel(dest_path)
    Capital Population
1  New York   16044000
2    Berlin    3433695
3    Madrid    3010492
4 Stockholm    1683713
```

Why download.file()?

- Reproducibility
- ~~Point and click~~
- HTTP from inside R
  - Authentication
  - Additional parameters
  - httr - Hadley Wickham

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

APIs & JSON

Other data formats

- Before: pages and files from the web
- JSON
- Simple, concise, well-structured
- Human-readable
- Easy to parse and generate for computers
- For communication with Web APIs

API

- Application Programming Interface
- Set of routines and protocols for building software
- How different components interact
- Web API
  - interface to get or add data to server
  - HTTP verbs (GET and others)

Twitter

- https://dev.twitter.com/rest/public
- Get tweets
- Place comments on tweets
- Many applications
    - Research effect of tweets

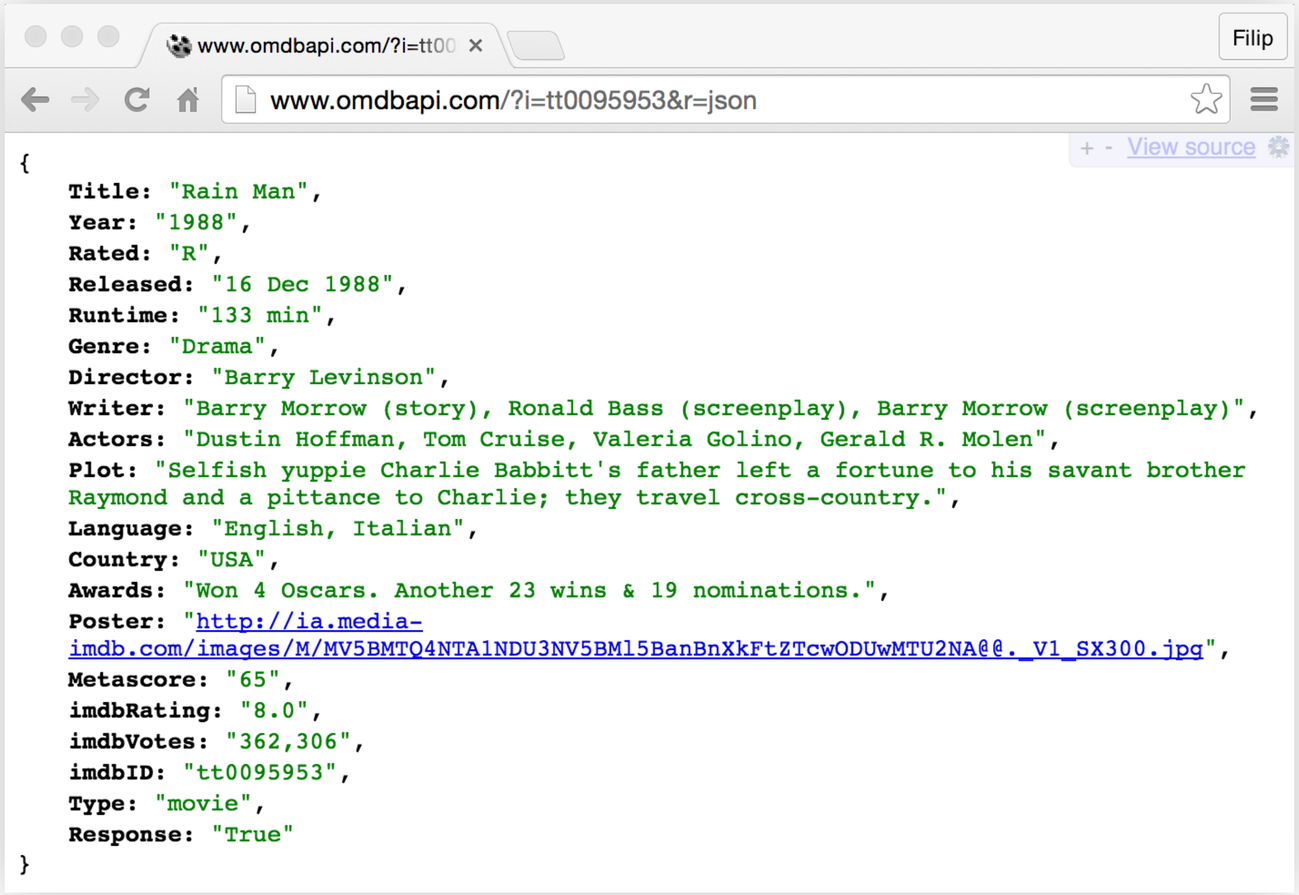
Info on Rain Man (1988)

```
> url <- "http://www.imdb.com/title/tt0095953/"
> download.file(url, "local_imdb.html")
```

```
        <div class="pro-title-link text-center">
<a href="http://pro.imdb.com/title/tt0095953?rf=cons_tt_contact&ref_=cons_tt_conta
>Contact the Filmmakers on IMDbPro &raquo;</a>
        </div> </td>
                    <td id="overview-top">
        <div id="prometer_container">
            <div id="prometer" class="meter-collapsed up">
                <div id="meterHeaderBox">
                    <div id="meterTitle" class="meterToggleOnHover">Popularity</di
                    <span id="meterRank">1,303</span>
                </div>
                <div id="meterChangeRow" class="meterToggleOnHover">
                        <span>Up</span>
                    <span id="meterChange">163</span>
                    <span>this week</span>
                </div>
            </div>
        </div>
<h1 class="header"> <span class="itemprop" itemprop="name">Rain Man</span>
            <span class="nobr">(<a href="/year/1988/?ref_=tt_ov_inf"
>1988</a>)</span>
```

Rain Man JSON (OMDb API)

http://www.omdbapi.com/?i=tt0095953&r=json



jsonlite

- Jeroen Ooms
- Improvement of earlier packages
- Consistent, robust
- Support all use-cases

Rain Man list in R

```
> install.packages("jsonlite")
> library(jsonlite)

> fromJSON("http://www.omdbapi.com/?i=tt0095953&r=json")
List of 20
 $ Title      : chr "Rain Man"
 $ Year       : chr "1988"
 $ Rated      : chr "R"
 $ Released   : chr "16 Dec 1988"
 $ Runtime    : chr "133 min"
 ...
 $ imdbVotes  : chr "359,903"
 $ imdbID     : chr "tt0095953"
 $ Type       : chr "movie"
 $ Response   : chr "True"
```

Way more structure!

JSON object

```JSON
{"id":1,"name":"Frank","age":23,"married":false}
```

name : value

string string
number
boolean
null
JSON object
JSON array

JSON object

```JSON
{"id":1,"name":"Frank","age":23,"married":false}
```

```R
> x <- '{"id":1,"name":"Frank","age":23,"married":false}'

> r <- fromJSON(x)

> str(r)
List of 4
 $ id     : int 1
 $ name   : chr "Frank"
 $ age    : int 23
 $ married: logi FALSE
```

JSON array

```
[4, 7, 4, 6, 4, 5, 10, 6, 6, 8]
```
JSON

```
> fromJSON('[4, 7, 4, 6, 4, 5, 10, 6, 6, 8]')
[1]  4  7  4  6  4  5 10  6  6  8
```
R

```
[4, "a", 4, 6, 4, "b", 10, 6, false, null]
```
JSON

```
> fromJSON('[4, "a", 4, 6, 4, "b", 10, 6, false, null]')
[1] "4"  "a"  "4"  "6"  "4"  "b"  "10"  "6"  "FALSE"  NA
```
R

JSON Nesting

```JSON
{
    "id": 1,
    "name": "Frank",
    "age": 23,
    "married": false
}
```

JSON Nesting

```JSON
{
    "id": 1,
    "name": "Frank",
    "age": 23,
    "married": false,
    "partner": {
      "id": 4,
      "name": "Julie"
    }
}
```

JSON Nesting

R

```
> r <- fromJSON('{"id":1,"name":"Frank","age":23,
      "married":false,"partner":{"id":4,"name":"Julie"}}')

> str(r)
List of 5
 $ id     : int 1
 $ name   : chr "Frank"
 $ age    : int 23
 $ married: logi FALSE
 $ partner:List of 2
  ..$ id  : int 4
  ..$ name: chr "Julie"
```

JSON Array of JSON Objects

```JSON
[
   {"id":1, "name":"Frank"},
   {"id":4, "name":"Julie"},
   {"id":12, "name":"Zach"}
]
```

```R
> fromJSON('[{"id":1, "name":"Frank"},
             {"id":4, "name":"Julie"},
             {"id":12, "name":"Zach"}]')
  id  name
1  1 Frank
2  4 Julie
3 12  Zach
```

Other jsonlite functions

- toJSON()
- prettify()
- minify()

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

APIs & JSON

Other data formats

- Before: pages and files from the web
- JSON
- Simple, concise, well-structured
- Human-readable
- Easy to parse and generate for computers
- For communication with Web APIs

API

- Application Programming Interface
- Set of routines and protocols for building software
- How different components interact
- Web API
  - interface to get or add data to server
  - HTTP verbs (GET and others)

Twitter

- https://dev.twitter.com/rest/public
- Get tweets
- Place comments on tweets
- Many applications
    - Research effect of tweets

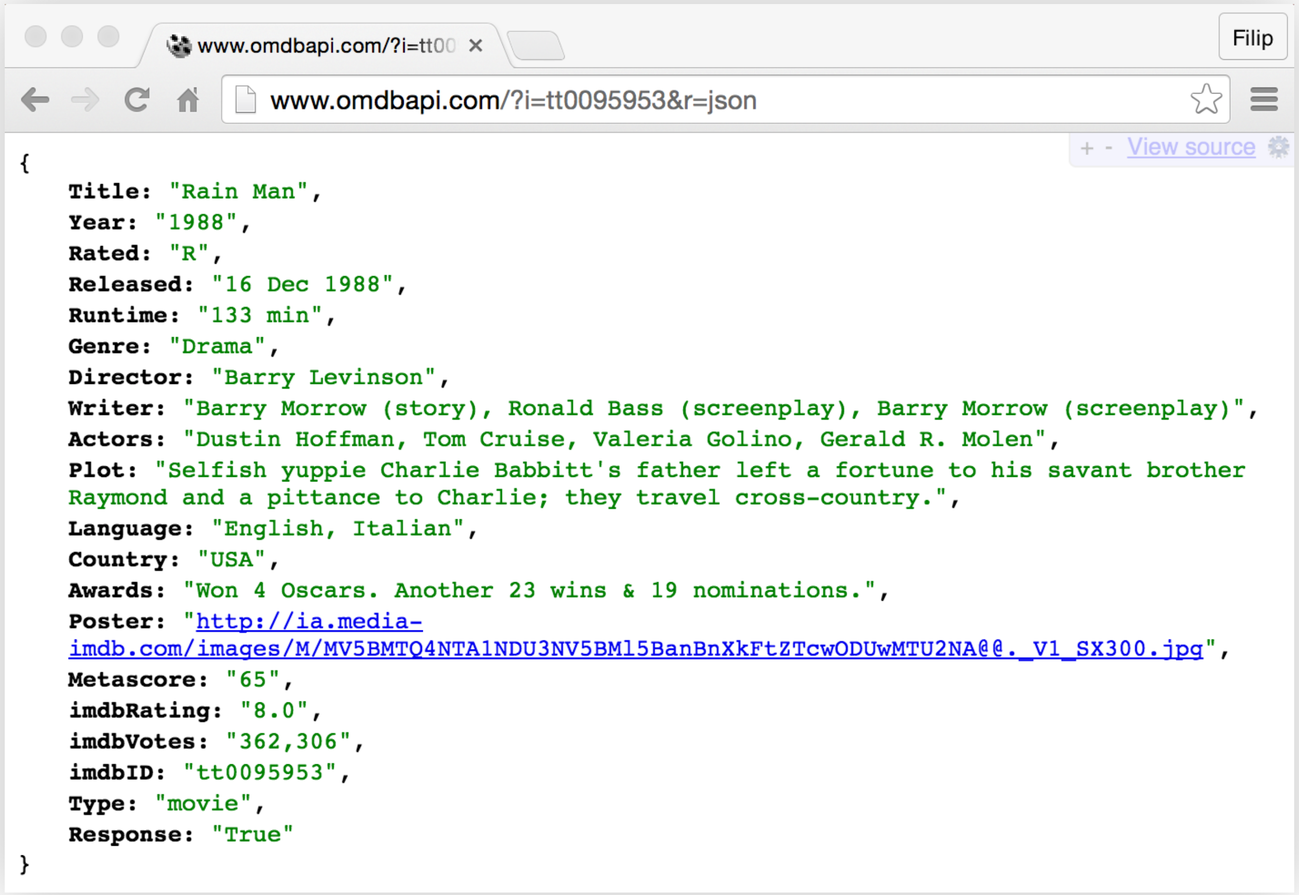
Info on Rain Man (1988)

```
> url <- "http://www.imdb.com/title/tt0095953/"
> download.file(url, "local_imdb.html")
```

```
        <div class="pro-title-link text-center">
<a href="http://pro.imdb.com/title/tt0095953?rf=cons_tt_contact&ref_=cons_tt_conta
>Contact the Filmmakers on IMDbPro &raquo;</a>
        </div> </td>
                    <td id="overview-top">
        <div id="prometer_container">
            <div id="prometer" class="meter-collapsed up">
                <div id="meterHeaderBox">
                    <div id="meterTitle" class="meterToggleOnHover">Popularity</di
                    <span id="meterRank">1,303</span>
                </div>
                <div id="meterChangeRow" class="meterToggleOnHover">
                        <span>Up</span>
                    <span id="meterChange">163</span>
                    <span>this week</span>
                </div>
            </div>
        </div>
<h1 class="header"> <span class="itemprop" itemprop="name">Rain Man</span>
            <span class="nobr">(<a href="/year/1988/?ref_=tt_ov_inf"
>1988</a>)</span>
```

Rain Man JSON (OMDb API)

http://www.omdbapi.com/?i=tt0095953&r=json



jsonlite

- Jeroen Ooms
- Improvement of earlier packages
- Consistent, robust
- Support all use-cases

Rain Man list in R

```
> install.packages("jsonlite")
> library(jsonlite)

> fromJSON("http://www.omdbapi.com/?i=tt0095953&r=json")
List of 20
 $ Title      : chr "Rain Man"
 $ Year       : chr "1988"
 $ Rated      : chr "R"
 $ Released   : chr "16 Dec 1988"
 $ Runtime    : chr "133 min"
 ...
 $ imdbVotes  : chr "359,903"
 $ imdbID     : chr "tt0095953"
 $ Type       : chr "movie"
 $ Response   : chr "True"
```

Way more structure!

JSON object

```JSON
{"id":1,"name":"Frank","age":23,"married":false}
```

name : value

string string
number
boolean
null
JSON object
JSON array

JSON object

```JSON
{"id":1,"name":"Frank","age":23,"married":false}
```

```R
> x <- '{"id":1,"name":"Frank","age":23,"married":false}'

> r <- fromJSON(x)

> str(r)
List of 4
 $ id     : int 1
 $ name   : chr "Frank"
 $ age    : int 23
 $ married: logi FALSE
```

JSON array

```
[4, 7, 4, 6, 4, 5, 10, 6, 6, 8]
```
JSON

```
> fromJSON('[4, 7, 4, 6, 4, 5, 10, 6, 6, 8]')
[1]  4  7  4  6  4  5 10  6  6  8
```
R

```
[4, "a", 4, 6, 4, "b", 10, 6, false, null]
```
JSON

```
> fromJSON('[4, "a", 4, 6, 4, "b", 10, 6, false, null]')
[1] "4"  "a"  "4"  "6"  "4"  "b"  "10"  "6"  "FALSE"  NA
```
R

JSON Nesting

```JSON
{
    "id": 1,
    "name": "Frank",
    "age": 23,
    "married": false
}
```

JSON Nesting

```JSON
{
    "id": 1,
    "name": "Frank",
    "age": 23,
    "married": false,
    "partner": {
      "id": 4,
      "name": "Julie"
    }
}
```

JSON Nesting

R

```
> r <- fromJSON('{"id":1,"name":"Frank","age":23,
      "married":false,"partner":{"id":4,"name":"Julie"}}')

> str(r)
List of 5
 $ id     : int 1
 $ name   : chr "Frank"
 $ age    : int 23
 $ married: logi FALSE
 $ partner:List of 2
  ..$ id  : int 4
  ..$ name: chr "Julie"
```

JSON Array of JSON Objects

```JSON
[
   {"id":1, "name":"Frank"},
   {"id":4, "name":"Julie"},
   {"id":12, "name":"Zach"}
]
```

```R
> fromJSON('[{"id":1, "name":"Frank"},
             {"id":4, "name":"Julie"},
             {"id":12, "name":"Zach"}]')
  id  name
1  1 Frank
2  4 Julie
3 12  Zach
```

Other jsonlite functions

- toJSON()
- prettify()
- minify()

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Importing Data from Statistical Software haven

Statistical Software Packages

Package	Expanded Name	Application	Data File Extensions
SAS	Statistical Analysis Software	Business Analytics Biostatistics Medical Sciences	.sas7bdat .sas7bcat
STATA	STAtistics and daTA	Economists	.dta
SPSS	Statistical Package for Social Sciences	Social Sciences	.sav .por

R packages to import data

- haven
  - Hadley Wickham
  - Goal: consistent, easy, fast
- foreign
  - R Core Team
  - Support for many data formats

haven

- SAS, STATA and SPSS
- ReadStat: C library by Evan Miller
- Extremely simple to use
- Single argument: path to file
- Result: R data frame

```
> install.packages("haven")
> library(haven)
```

SAS data

- ontime.sas7bdat
- Delay statistics for airlines in US
- read_sas()

```
> ontime <- read_sas("ontime.sas7bdat")
```

SAS data

```
> ontime <- read_sas("ontime.sas7bdat")

> str(ontime)

Classes ‘tbl_df’, ‘tbl’ and 'data.frame':10 obs. of  4
variables:
 $ Airline    : atomic  TWA Southwest Northwest ...
  ..- attr(*, "label")= chr "Airline"
 $ March_1999 : atomic  84.4 80.3 80.8 72.7 78.7 ...
  ..- attr(*, "label")= chr "March 1999"
 $ June_1999  : atomic  69.4 77 75.1 65.1 72.2 ...
  ..- attr(*, "label")= chr "June 1999"
 $ August_1999: atomic  85 80.4 81 78.3 77.7 75.1 ...
  ..- attr(*, "label")= chr "August 1999"
```

Labels assigned inside SAS

SAS data

```
> ontime <- read_sas("ontime.sas7bdat")

> ontime
          Airline March_1999 June_1999 August_1999
1             TWA       84.4      69.4         85.0
2       Southwest       80.3      77.0         80.4
3       Northwest       80.8      75.1         81.0
4        American       72.7      65.1         78.3
5           Delta       78.7      72.2         77.7
6     Continental       79.3      68.4         75.1
7          United       78.6      69.2         71.6
8      US Airways       73.6      68.9         70.1
9          Alaska       71.9      75.4         64.4
10 American West       76.5      70.3         62.5
```

SAS data

```
> ontime <- read_sas("ontime.sas7bdat")
```

	Airline Airline	March_1999 March 1999	June_1999 June 1999	August_1999 August 1999
1	TWA	84.4	69.4	85.0
2	Southwest	80.3	77.0	80.4
3	Northwest	80.8	75.1	81.0
4	American	72.7	65.1	78.3
5	Delta	78.7	72.2	77.7
6	Continental	79.3	68.4	75.1
7	United	78.6	69.2	71.6
8	US Airways	73.6	68.9	70.1
9	Alaska	71.9	75.4	64.4
10	American West	76.5	70.3	62.5

STATA data

- STATA 13 & STATA 14
- read_stata(), read_dta()

STATA data

```
> ontime <- read_stata("ontime.dta")
> ontime <- read_dta("ontime.dta")

> ontime
   Airline March_1999 June_1999 August_1999
1        8       84.4      69.4         85.0
2        7       80.3      77.0         80.4
3        6       80.8      75.1         81.0
4        2       72.7      65.1         78.3
5        5       78.7      72.2         77.7
6        4       79.3      68.4         75.1
7        9       78.6      69.2         71.6
8       10       73.6      68.9         70.1
9        1       71.9      75.4         64.4
10       3       76.5      70.3         62.5
```

Numbers, not character strings?!

STATA data

```
> ontime <- read_stata("ontime.dta")
> ontime <- read_dta("ontime.dta")

> class(ontime$Airline)
[1] "labelled"
```

R version of common data structure

```
> ontime$Airline
<Labelled>
 [1]  8  7  6  2  5  4  9 10  1  3
attr(,"label")
[1] "Airline"
Labels:
       Alaska   American  American West  ...  US Airways
            1          2              3  ...          10
```

as_factor()

```
> ontime <- read_stata("ontime.dta")
> ontime <- read_dta("ontime.dta")


> as_factor(ontime$Airline)
 [1] TWA    Southwest  Northwest   American ... American West
Levels: Alaska American American West ... US Airways


> as.character(as_factor(ontime$Airline))
 [1] "TWA" "Southwest" "Northwest" ... "American West"
```

as_factor()

```
> ontime$Airline <- as.character(as_factor(ontime$Airline))

> ontime
          Airline March_1999 June_1999 August_1999
1             TWA       84.4      69.4         85.0
2       Southwest       80.3      77.0         80.4
3       Northwest       80.8      75.1         81.0
4        American       72.7      65.1         78.3
5           Delta       78.7      72.2         77.7
6     Continental       79.3      68.4         75.1
7          United       78.6      69.2         71.6
8      US Airways       73.6      68.9         70.1
9          Alaska       71.9      75.4         64.4
10 American West       76.5      70.3         62.5
```

SPSS data

- read_spss()
- .por -> read_por()
- .sav -> read_sav()

```
> read_sav(file.path("~","datasets","ontime.sav"))

   Airline Mar.99 Jun.99 Aug.99
1        8   84.4   69.4   85.0
2        7   80.3   77.0   80.4
3        6   80.8   75.1   81.0
4        2   72.7   65.1   78.3
5        5   78.7   72.2   77.7
...
10       3   76.5   70.3   62.5
```

Statistical Software Packages

Package	Expanded Name	Application	Data File Extensions	haven function
SAS	Statistical Analysis Software	Business Analytics Biostatistics Medical Sciences	.sas7bdat .sas7bcat	read_sas()
STATA	STAtistics and daTA	Economists	.dta	read_dta() read_stata()
SPSS	Statistical Package for Social Sciences	Social Sciences	.sav .por	read_spss() read_por() read_sav()

IMPORTING DATA IN R

Let’s practice!

IMPORTING DATA IN R

Importing Data from Statistical Software foreign

foreign

- R Core Team
- Less consistent
- Very comprehensive
- All kinds of foreign data formats
- SAS, STATA, SPSS, Systat, Weka ...

```
> install.packages("foreign")
> library(foreign)
```

SAS

- Cannot import .sas7bdat
- Only SAS libraries: .xport
- sas7bdat package

STATA

- STATA 5 to 12
- read.dta() — read_dta()

```
read.dta(file,          <- path to local file or URL
         convert.factors = TRUE,
         convert.dates = TRUE,
         missing.type = FALSE)
```

read.dta()

```
> ontime <- read.dta("ontime.dta")

> ontime
          Airline March_1999 June_1999 August_1999
1             TWA       84.4      69.4         85.0
2       Southwest       80.3      77.0         80.4
3       Northwest       80.8      75.1         81.0
4        American       72.7      65.1         78.3
5           Delta       78.7      72.2         77.7
6     Continental       79.3      68.4         75.1
7          United       78.6      69.2         71.6
8      US Airways       73.6      68.9         70.1
9          Alaska       71.9      75.4         64.4
10 American West       76.5      70.3         62.5
```

read.dta()

```
> ontime <- read.dta("ontime.dta")

> str(ontime)
'data.frame':10 obs. of  4 variables:
 $ Airline    : Factor w/ 10 levels "Alaska",..: 8 7 6 2 5 4 ...
 $ March_1999 : num  84.4 80.3 80.8 72.7 78.7 79.3 78.6 ...
 $ June_1999  : num  69.4 77 75.1 65.1 72.2 68.4 69.2 68.9 ...
 $ August_1999: num  85 80.4 81 78.3 77.7 75.1 71.6 70.1 ...
 - attr(*, "datalabel")= chr "Written by R.                "
 - attr(*, "time.stamp")= chr ""
 - attr(*, "formats")= chr  "%9.0g" "%9.0g" "%9.0g" "%9.0g"
 - attr(*, "types")= int  108 100 100 100
 - attr(*, "val.labels")= chr  "Airline" "" "" ""
 - attr(*, "var.labels")= chr  "Airline" "March_1999" ...
 - attr(*, "version")= int 7
 - attr(*, "label.table")=List of 1
  ..$ Airline: Named int  1 2 3 4 5 6 7 8 9 10
  .. ..- attr(*, "names")= chr  "Alaska" "American" ...
```

convert.factors TRUE by default

read.dta() - convert.factors

```
> ontime <- read.dta("ontime.dta", convert.factors = FALSE)

> str(ontime)
'data.frame':10 obs. of  4 variables:
 $ Airline    : int  8 7 6 2 5 4 9 10 1 3
 $ March_1999 : num  84.4 80.3 80.8 72.7 78.7 79.3 78.6 ...
 $ June_1999  : num  69.4 77 75.1 65.1 72.2 68.4 69.2 68.9 ...
 $ August_1999: num  85 80.4 81 78.3 77.7 75.1 71.6 70.1 ...
 - attr(*, "datalabel")= chr "Written by R.                "
 - attr(*, "time.stamp")= chr ""
 - attr(*, "formats")= chr  "%9.0g" "%9.0g" "%9.0g" "%9.0g"
 - attr(*, "types")= int  108 100 100 100
 - attr(*, "val.labels")= chr  "Airline" "" "" ""
 - attr(*, "var.labels")= chr  "Airline" "March_1999" ...
 - attr(*, "version")= int 7
 - attr(*, "label.table")=List of 1
  ..$ Airline: Named int  1 2 3 4 5 6 7 8 9 10
  .. ..- attr(*, "names")= chr  "Alaska" "American" ...
```

read.dta() - more arguments

```
read.dta(file,
         convert.factors = TRUE,
         convert.dates = TRUE,
         missing.type = FALSE)
```

convert.factors: convert labelled STATA values to R factors
convert.dates: convert STATA dates and times to Date and POSIXct
missing.type: if FALSE, convert all types of missing values to NA
if TRUE, store how values are missing in attributes

SPSS

read.spss()

```
read.spss(file,
          use.value.labels = TRUE,
          to.data.frame = FALSE)
```

use.value.labels: convert labelled SPSS values to R factors
to.data.frame: return data frame instead of a list
trim.factor.names
trim_values
use.missings
...

IMPORTING DATA IN R

Let’s practice!