Connect to a database

INTERMEDIATE IMPORTING DATA IN R

Filip Schouwenaars
Instructor, DataCamp

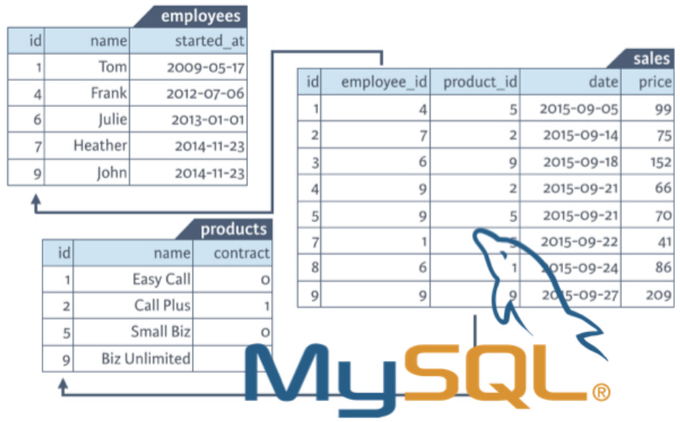
datacamp

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

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

- con is DBIConnection object

List and import tables

```
dbListTables(con)
```

```
"employees" "products"  "sales"
```

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

```
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)
```

```
TRUE
```

```
con
```

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

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 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
```

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, n = 2)
dbFetch(res) (second time)
```

```
  id          name contract
1  2     Call Plus        1
2  9 Biz Unlimited        1
```

In the first call, import only two records of the query result by setting the n argument to 2. In the second call, import all remaining queries (don't specify n). In both calls, simply print the resulting data frames.

```
dbClearResult(res)
```

```
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
id       name     contract
<0 rows> (or 0-length row.names)
```

```
dbClearResult(res)
```

```
TRUE
```

Disconnect

```
dbDisconnect(con)
```

```
TRUE
```

HTTP

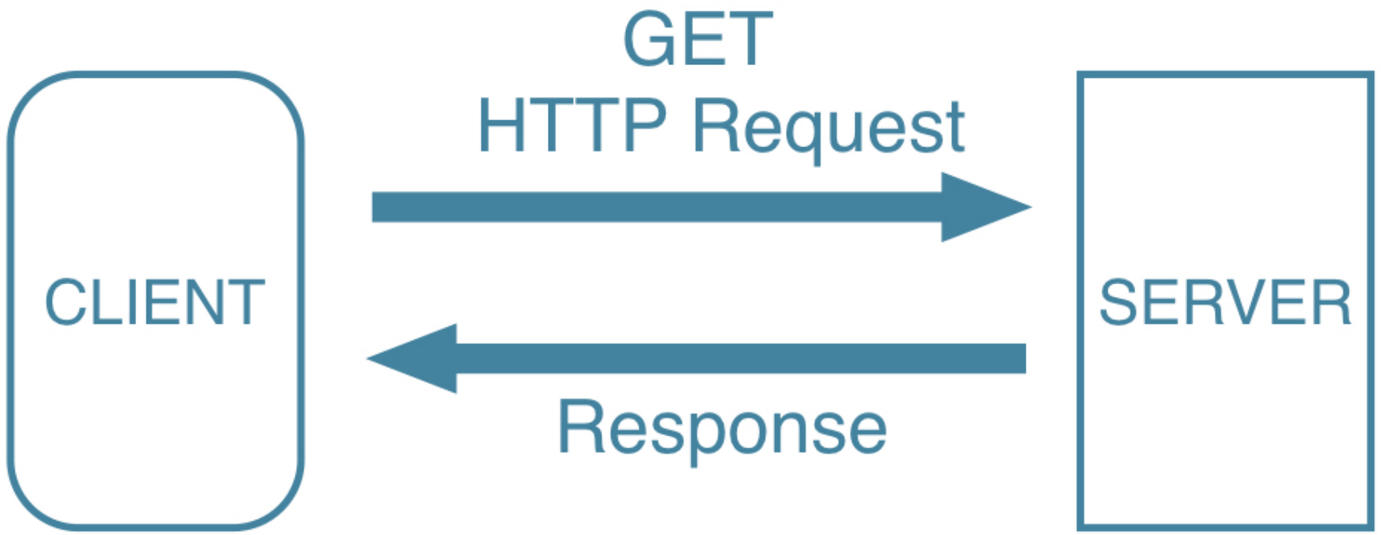
INTERMEDIATE IMPORTING DATA IN R

Filip Schouwenaars
Instructor, DataCamp

datacamp

HTTP

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



HTTP? httr! (1)

Downloading a file from the Internet means sending a GET request and receiving the file you asked for. Internally, all the previously discussed functions use a GET request to download files.

`httr` provides a convenient function, `GET()` to execute this GET request. The result is a `response` object, that provides easy access to the status code, content-type and, of course, the actual content.

You can extract the content from the request using the `content()` function. At the time of writing, there are three ways to retrieve this content: as a raw object, as a character vector, or an R object, such as a list. If you don't tell `content()` how to retrieve the content through the `as` argument, it'll try its best to figure out which type is most appropriate based on the content-type.

```
1  # Load the httr package
2  library(httr)
3
4  # Get the url, save response to resp
5  url <- "http://www.example.com/"
6  resp <- GET(url)
7
8  # Print resp
9  resp
10
11 # Get the raw content of resp: raw_content
12 raw_content <- content(resp, as = "raw")
13
14 # Print the head of raw_content
15 head(raw_content)
```

Example: CSV

http://s3.amazonaws.com/ ... /states.csv

```
# Manually download file through web browser

read.csv(url("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

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
```

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 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

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

name	value
string	string
	number
	boolean
	null
	JSON object
	JSON array

JSON object

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

```
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

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

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

```
4  7  4  6  4  5 10  6  6  8
```

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

```
fromJSON('[4, "a", 4, 6, 4, "b", 10, 6, false, null]')
```

```
"4"  "a"  "4"  "6"  "4"  "b"  "10"  "6"  "FALSE"  NA
```

JSON Nesting

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

JSON Nesting

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

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

```
RfromJSON('[{"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()

haven

INTERMEDIATE IMPORTING DATA IN R

Filip Schouwenaars
Instructor, DataCamp

datacamp

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"
```

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
```

STATA data

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

```
# R version of common data structure
class(ontime$Airline)
```

```
"labelled"
```

```
ontime$Airline
```

```
<Labelled>
8  7  6  2  5  4  9 10  1  3
attr(,"label")
"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)
```

```
TWA    Southwest  Northwest   American ... American West
Levels: Alaska American American West ... US Airways
```

```
as.character(as_factor(ontime$Airline))
```

```
"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()

foreign

INTERMEDIATE IMPORTING DATA IN R

Filip Schouwenaars
Instructor, DataCamp

datacamp

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,
         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)
```

- `convert.factors` TRUE by default

```
'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" ...
```

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`

Let's practice!

INTERMEDIATE IMPORTING DATA IN R