Getting your data into R

Hadley Wickham

@hadleywickham

Chief Scientist, RStudio

June 2015

Visualise

Surprises, but doesn't scale

Tid

Transform

Model

Scales, but doesn't (fundamentally) surprise

On disk (csv, excel, SAS, ...)

In a database (SQL)

On the web (xml, json, ...)

Common features

Input

- Fast enough.
  (Want **fastest**? use data.table)
- No external dependencies.
  (just C and C++ bundled with the package)
- Consistent function names and arguments.
- Underscores, not dots.

Output

- No row names.
- Never change column names.
- Retain dates.
- Return a `tbl_df`.
  (better printing if dplyr loaded)

Never turn characters into factors!

On disk

```
readr::read_csv()
readr::read_csv2()
readr::read_tsv()
readr::read_log()
readr::read_delim()
readr::read_fwf()
readr::read_table()

readxl::read_excel()

haven::read_sas()
haven::read_spss()
haven::read_stata()
```

Column types

- Logical, integer, double, character
- Factor
- ISO8601 date times
- Dates with format string (`%Y-%m-%d`)
- Sloppy numeric parser

```
library(readr)

read_csv("my.csv",
  col_names = c("x", "y", "z")
  col_types = list(
    x = col_date("%m/%d/%Y"),
    y = col_datetime(),
    z = col_integer()
  )
)

# Heuristic currently looks at first 1000 rows
# Any problems recorded in a data frame
```

Data	Package	Alternatives
Statistics packages	haven	foreign, sas7bdat, readstata13
Excel	readxl	gdata, openxlsx, XLConnect, xlsx
Flat files	readr	base, data.table

In a database

```
# Best way to talk to a database is with the DBI
# package. It provides a common front-end to many
# backends

# 1) Load the DBI package
library(DBI)

# 2) Connect to a specific database
db <- dbConnect(RPostgres::Postgres(), user, pass, ...)
db <- dbConnect(RMySQL::MySQL(), user, pass, ...)
db <- dbConnect(RSQLite::SQLite(), path)

# 3) Execute a query
dbGetQuery(db, "SELECT * FROM mtcars")

# 4) Be polite and close connection
dbDisconnect(db)
```

Three families of database packages

RPostgres → Postgres

(DBI)

RODBC → ODBC Postgres driver → Postgres

RJDBC → Java → JDBC Postgres driver → Postgres

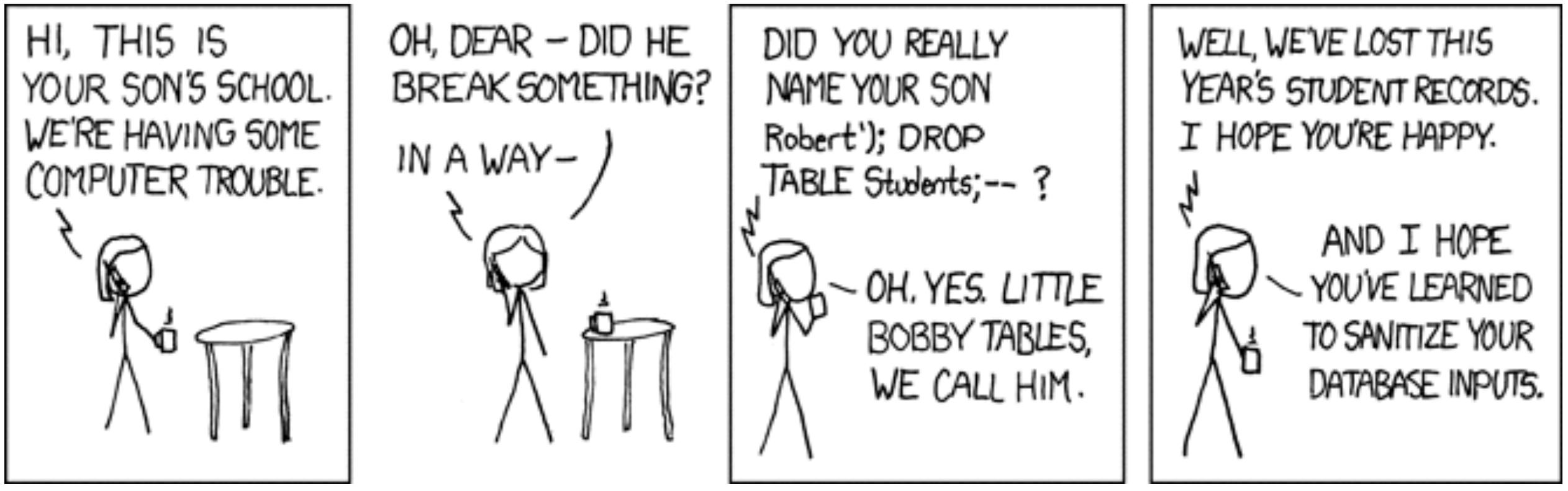
More layers make code slower and installation more painful (can't just install R package, need Java, more drivers etc)

Dev versions

(Somewhat aspirational goals)

- Never leak memory. Never leak connections. Never crash.
- Always send and receive UTF-8 text
- Always send and receive datetimes in UTC.
- A little faster than previous versions.
- **Provide parameterised query interface**

```
# For the rest of the talk I want to focus
# on the development versions. Probably a month
# or two away from CRAN, but contain some important
# new features

# http://github.com/rstats-db/
devtools::install_github("rstats-db/DBI")
devtools::install_github("rstats-db/RPostgres")
devtools::install_github("rstats-db/RMySQL")
devtools::install_github("rstats-db/RSQLite")
```



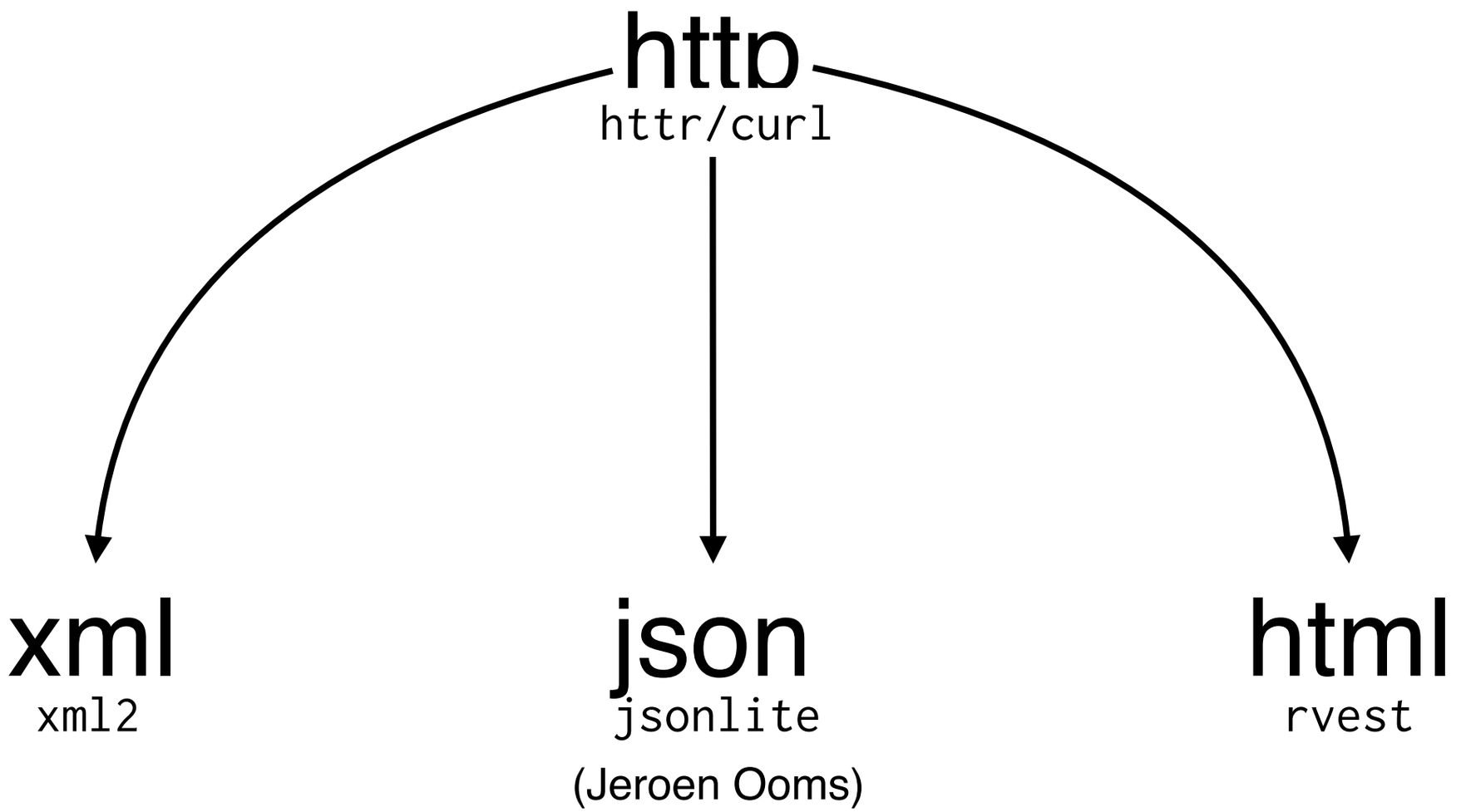
http://xkcd.com/327/

http://bobby-tables.com/

```
find_student <- function(db, name) {
  sql <- paste0("SELECT * FROM Students",
    "WHERE (name = '", name, "');")
  dbGetQuery(db, sql)
}

find_student("Hadley")
# SELECT * FROM Students
# WHERE (name = 'Hadley');

find_student("Robert'); DROP TABLE Students; --")
# SELECT * FROM Students
# WHERE (name = 'Robert');
# DROP TABLE Students; --');
```

```r
find_student <- function(db, name) {
  sql <- "SELECT * FROM Students WHERE (name = ?);
  dbGetQuery(db, sql, list(name))
}

find_student("Hadley")
# SELECT * FROM Students
# WHERE (name = 'Hadley');

find_student("Robert'); DROP TABLE Students; --")
# SELECT * FROM Students
# WHERE (name = 'Robert'' DROP TABLE Students; --')
```

On the web

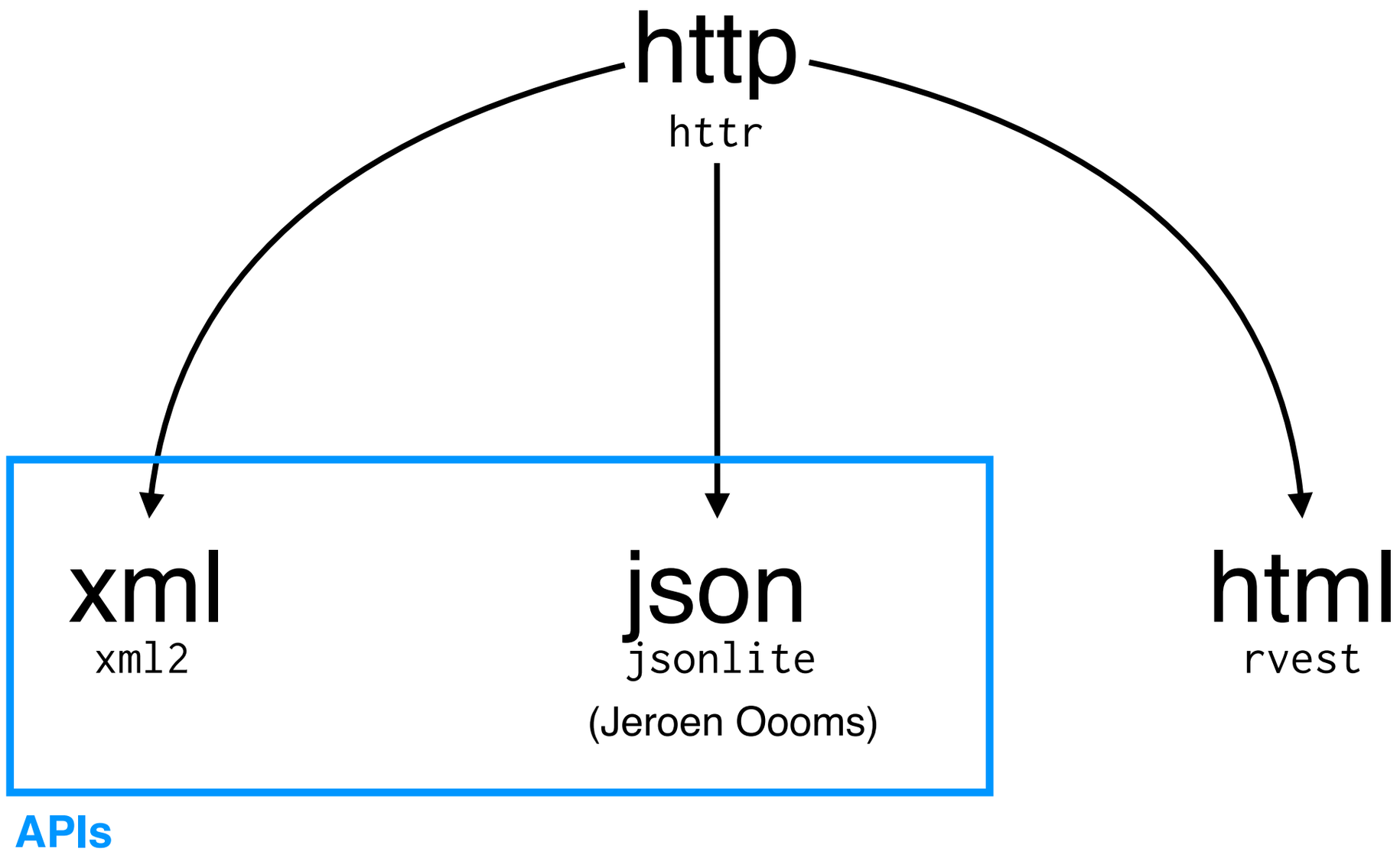

http
httr/curl
xml
xml2
json
jsonlite
(Jeroen Ooms)
html
rvest

```
Request ------------------------------
[VERB] [URL] [VERSION]
[HEADER-NAME]: [HEADER-VALUE]

[BODY?]
```

```
Response -----------------------------
[VERSION] [STATUS]
[HEADER-NAME]: [HEADER-VALUE]

[BODY?]
```

```
-> GET / HTTP/1.1
-> User-Agent: curl/7.37.1 Rcurl/1.95.4.5 httr/0.6.1.9000
-> Host: www.google.com
-> Accept-Encoding: gzip
-> Accept: application/json, text/xml, application/xml, */*
->
<- HTTP/1.1 200 OK
<- Date: Mon, 23 Mar 2015 17:31:11 GMT
<- Expires: -1
<- Cache-Control: private, max-age=0
<- Content-Type: text/html; charset=ISO-8859-1
<- Set-Cookie: PREF=...; expires=Wed, 22-Mar-2017 17:31:11 GMT; path=/;
<- Server: gws
<- X-XSS-Protection: 1; mode=block
<- X-Frame-Options: SAMEORIGIN
<- Vary: Accept-Encoding
<- Transfer-Encoding: chunked
<-
<<  <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
<<  lang="en"><head><meta content=<< "Search the world's information, including
<<  webpages, images, videos and more. Google has many special features to help you
<<  find exactly what you're looking for." name="description"><meta content="noodp"
<<  name="robots"><meta content="/logos/...
...
```

```
library(httr)

r <- GET("http://google.com", verbose(data_in = TRUE))
http_status(r)
content(r)

# Also POST, PUT, DELETE, HEAD, & VERB

# Lots of helper functions to generate headers
# * authenticate()
# * oauth1_token(); oauth2_token()
# * add_header()
# * content_type()

# Other helpers connect to underlying curl:
# * progress()
# * verbose()
```

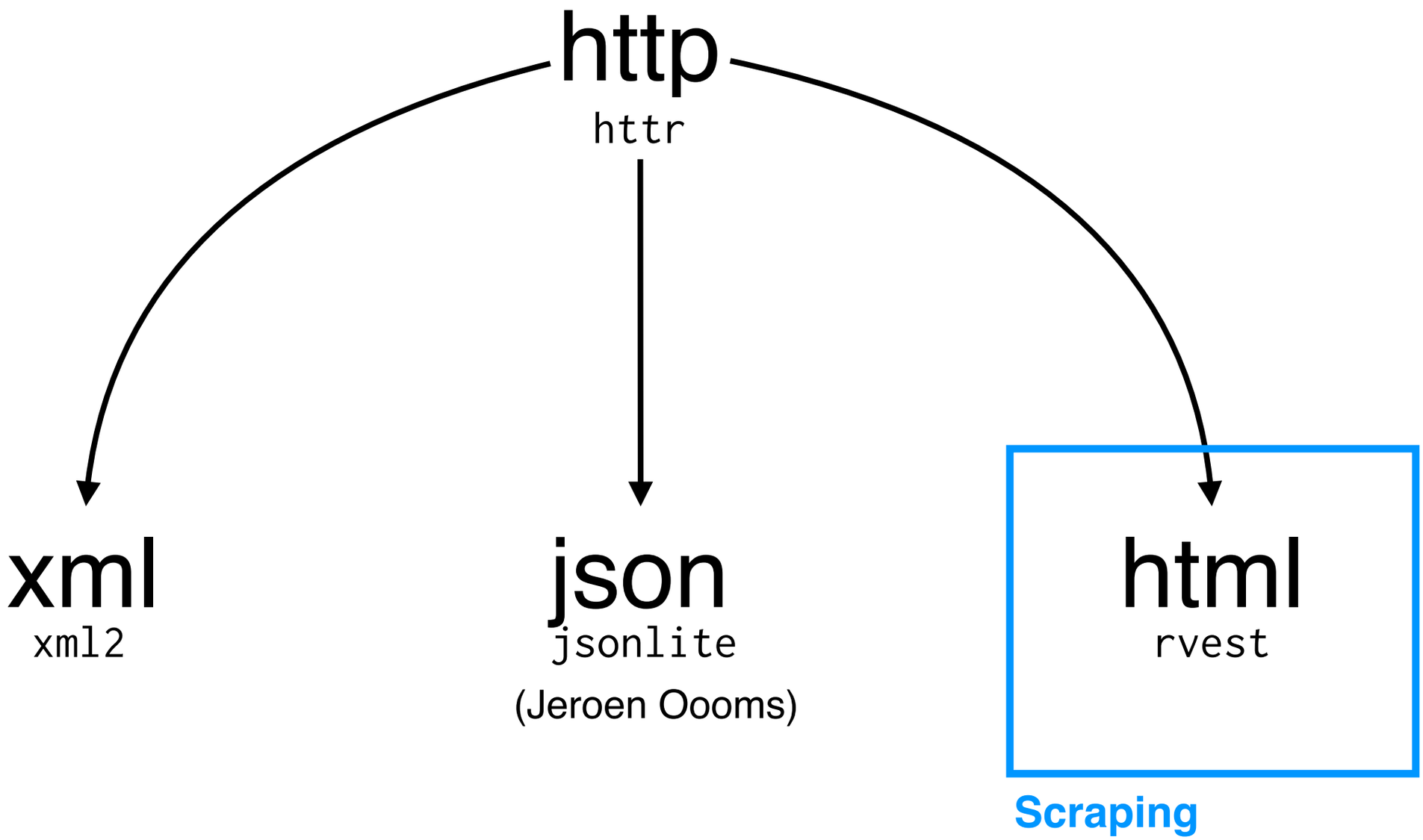

http
httr
xml
xml2
json
jsonlite
(Jeroen Oooms)
html
rvest
APIs

xml

- xml2 is a new binding to libxml2 (alternative to XML package).
- Works with xml and html.
- Simple, no memory leaks, no crashes. RCpp & C++.
- I hope you don’t have to work with xml, but if you do xml2 should make it less painful.

jsonlite

- Use jsonlite, not rjson, or RJSONIO
- Actively maintained
- Lots of unit tests
- C backend & performance improving


http
httr
xml
xml2
json
jsonlite
(Jeroen Oooms)
html
rvest
Scraping

rvest

- Not all data comes in via a machine readable format like **json** or **xml**.
- Sometimes you need to *scrape* (or *harvest*) data from human readable **html**.
- Goal of rvest is to provide pipeable API to make that as easy as possible.
- (rvest = beautiful soup for R)

Demo

Tidy data

On disk (csv, excel, SAS, ...)

In a database (SQL)

On the web (xml, json, ...)

Tidy data = data that makes data analysis easy

Storage	Meaning
Table / File	Data set
Rows	Observations
Columns	Variables

```
library(tidyr)
library(dplyr, warn = FALSE)
tb <- tbl_df(read.csv("tb.csv", stringsAsFactors = FALSE))
tb
#> Source: local data frame [5,769 x 22]
#>
#>    iso2 year m04 m514 m014 m1524 m2534 m3544 m4554 m5564 m65 mu f04 f514
#> 1    AD 1989  NA   NA   NA     NA     NA     NA     NA     NA  NA NA  NA   NA
#> 2    AD 1990  NA   NA   NA     NA     NA     NA     NA     NA  NA NA  NA   NA
#> 3    AD 1991  NA   NA   NA     NA     N
#> 4    AD 1992  NA   NA   NA     NA     N
#> 5    AD 1993  NA   NA   NA     NA     N
#> 6    AD 1994  NA   NA   NA     NA     N
#> 7    AD 1996  NA   NA    0      0
#> 8    AD 1997  NA   NA    0      0      1      2      2      1   6 NA  NA   NA
#> 9    AD 1998  NA   NA    0      0      0      1      0      0   0 NA  NA   NA
#> 10   AD 1999  NA   NA    0      0      0      1      1      0   0 NA  NA   NA
#>..   ...  ... ...  ...  ...  ...    ...    ...    ...    ...    ... ... .. ...  ...
#> Variables not shown: f014 (int), f1524 (int), f2534 (int), f3544 (int),
#>  f4554 (int), f5564 (int), f65 (int), fu (i
```

What are the variables in this dataset? (Hint: f = female, u = unknown, 1524 = 15-24)

```
# To convert this messy data into tidy data
# we need two verbs. First we need to gather
# together all the columns that aren't variables
```

```
tb2 <- tb %>%
  gather(demo, n, -iso2, -year, na.rm = TRUE)
tb2
```

```r
# Then separate the demographic variable into
# sex and age
tb3 <- tb2 %>%
  separate(demo, c("sex", "age"), 1)
tb3
```

```r
# tidyr provides a few other useful verbs:
# spread (opposite of gather)
# extract (like separate, but uses regexp groups)
# unite (opposite of extract/gather)
# nest & unnest, …
```

Conclusions

Future plans

- Bug fixing and testing (you can help!)
- Get on CRAN! (RPostgres, RMySQL, RSQLite)
- GUI for all these packages in RStudio
- Better tools for navigating complex hierarchical data

Acknowledgements

- JJ Allaire
- Jeroen Ooms
- Evan Miller (ReadStat)
- rapidxml, libxml2, libxls, Rcpp, MySQL, Postgres, SQLite

Questions?