Getting your data into R

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

June 2015

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

In a database (SQL)

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

data.frame

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.
- Never turn characters into factors!
- Return a `tbl_df`.
  (better printing if dplyr loaded)

On disk

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

```r
# First argument is the path
haven::read_sas()
haven::read_spss()
haven::read_stata()

readxl::read_excel() # xls & xlsx

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

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
```

In a database

```r
# 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) Polite to disconnect from db when done
dbDisconnect(db)
```

Three families of database packages

$$\begin{tikzcd}
\text{RPostgres} \arrow[rrr] & & & \text{Postgres} \\
\text{(DBI)} & & & \\
\text{RODBC} \arrow[rr] & & \begin{array}{c}\text{ODBC}\\ \text{Postgres driver}\end{array} \arrow[r] & \text{Postgres} \\
\text{RJDBC} \arrow[r] & \text{Java} \arrow[r] & \begin{array}{c}\text{JDBC}\\ \text{Postgres driver}\end{array} \arrow[r] & \text{Postgres}
\end{tikzcd}$$

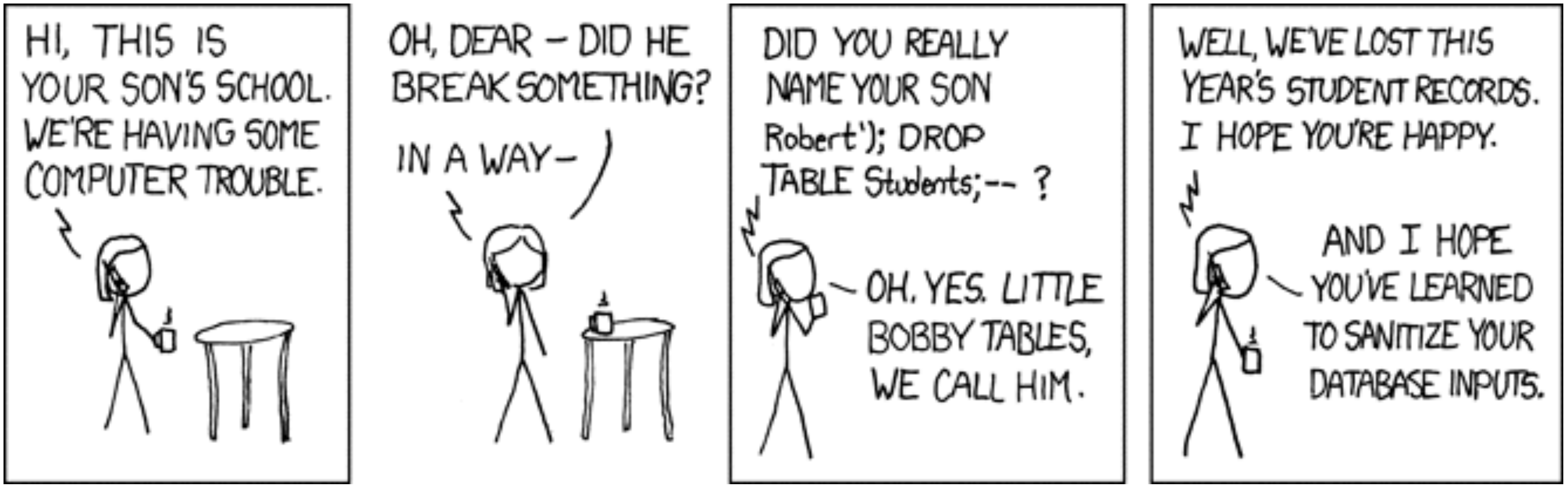
More layers make code slower and installation more painful (can’t just install R package, need Java, more drivers etc)

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**

```
# 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”)
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

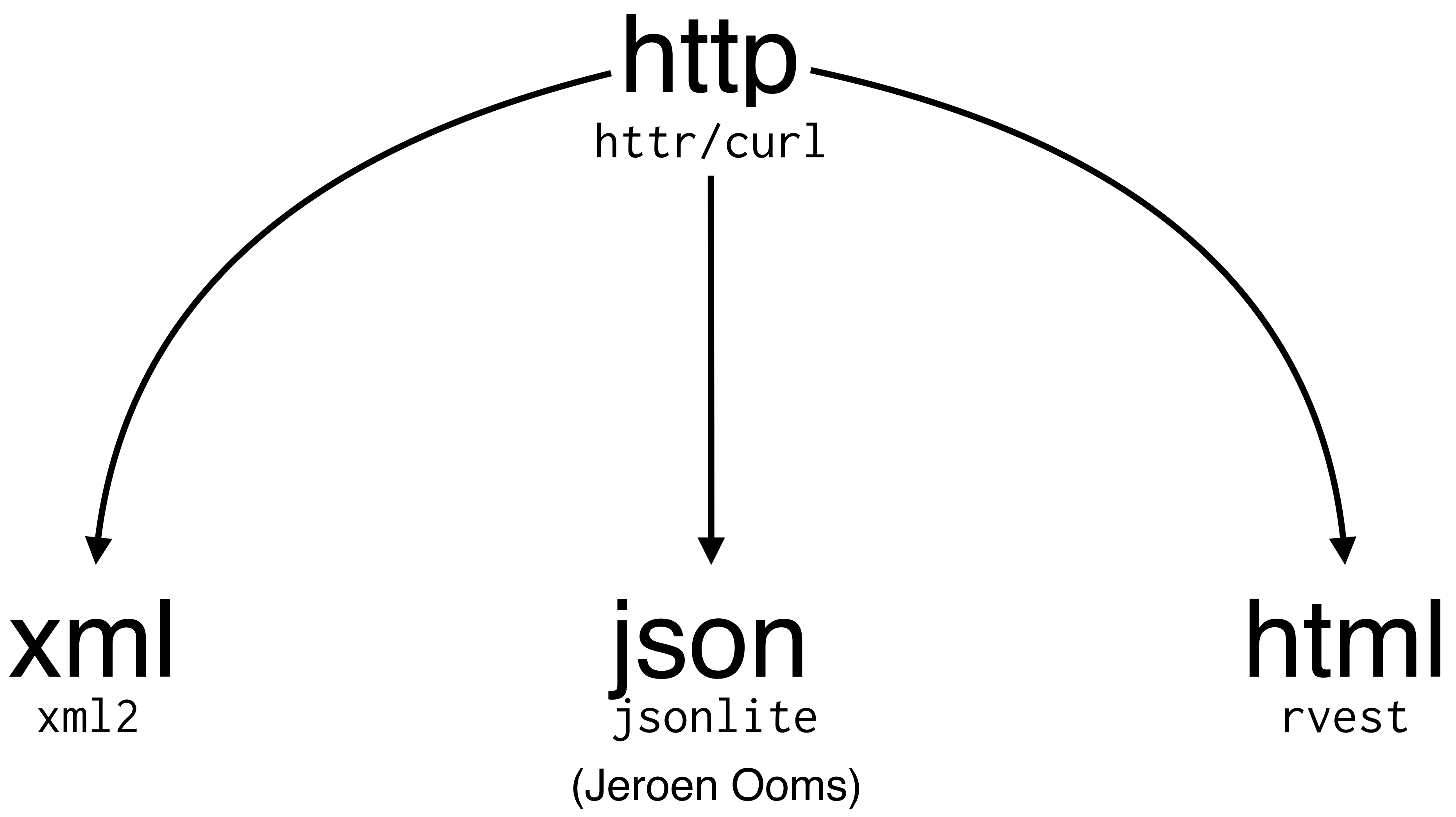
http://xkcd.com/327/

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; --');
```

```
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

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?