NC STATE UNIVERSITY

Importing Data: Excel Data, Databases, & More

Justin Post

What is this course about?

Basic use of R for reading, manipulating, and plotting data!

- read and write basic R programs
- **import well formatted data into R**
- do basic data manipulation in R
- produce common numerical and graphical summaries in R
- describe a use case of an analysis done in R

Recap/Next

- Read data from other sources

Type of file	Package	Function
Delimited	`readr`	`read_csv()`, `read_tsv()`, `read_table()`, `read_delim()`
Excel (.xls,.xlsx)	`readxl`	`read_excel()`
SAS (.sas7bdat)	`haven`	`read_sas()`
SPSS (.sav)	`haven`	`read_spss()`

- Resources for JSON, XML, databases, and APIs

Excel Data

- Read in censusEd.xlsx
- Use `read_excel()` from `readxl` package!

Excel Data

- Read in censusEd.xlsx
- Use `read_excel()` from `readxl` package!
    - Reads both xls and xlsx files
    - Detects format from extension given
    - Can't pull from web though!

Excel Data

```
#install package if necessary
library(readxl)
#reads first sheet by default
edData <- read_excel("../datasets/censusEd.xlsx")
edData
```

```
## # A tibble: 3,198 x 42
##   Area_name STCOU EDU010187F EDU010187D EDU010187N1 EDU010187N2 EDU010188F
##   <chr>     <chr>      <dbl>      <dbl> <chr>       <chr>            <dbl>
## 1 UNITED S… 00000          0   40024299 0000        0000                 0
## 2 ALABAMA   01000          0     733735 0000        0000                 0
## 3 Autauga,… 01001          0       6829 0000        0000                 0
## 4 Baldwin,… 01003          0      16417 0000        0000                 0
## 5 Barbour,… 01005          0       5071 0000        0000                 0
## # … with 3,193 more rows, and 35 more variables: EDU010188D <dbl>,
## #   EDU010188N1 <chr>, EDU010188N2 <chr>, EDU010189F <dbl>, EDU010189D <dbl>,
## #   EDU010189N1 <chr>, EDU010189N2 <chr>, EDU010190F <dbl>, EDU010190D <dbl>,
## #   EDU010190N1 <chr>, EDU010190N2 <chr>, EDU010191F <dbl>, EDU010191D <dbl>,
## #   EDU010191N1 <chr>, EDU010191N2 <chr>, EDU010192F <dbl>, EDU010192D <dbl>,
## #   EDU010192N1 <chr>, EDU010192N2 <chr>, EDU010193F <dbl>, EDU010193D <dbl>,
## #   EDU010193N1 <chr>, EDU010193N2 <chr>, EDU010194F <dbl>, EDU010194D <dbl>,
## #   EDU010194N1 <chr>, EDU010194N2 <chr>, EDU010195F <dbl>, EDU010195D <dbl>,
## #   EDU010195N1 <chr>, EDU010195N2 <chr>, EDU010196F <dbl>, EDU010196D <dbl>,
## #   EDU010196N1 <chr>, EDU010196N2 <chr>
```

Excel Data

- Read in censusEd.xlsx
- Use `read_excel()` from `readxl` package!
    - Specify sheet with name or integers (or `NULL` for 1st) using `sheet =`
    - Can look at sheets available

```
excel_sheets("../datasets/censusEd.xlsx")
```

```
##  [1] "EDU01A" "EDU01B" "EDU01C" "EDU01D" "EDU01E" "EDU01F" "EDU01G" "EDU01H"
##  [9] "EDU01I" "EDU01J"
```

```
read_excel("../datasets/censusEd.xlsx", sheet = "EDU01D")
```

Excel Data

- Use `read_excel()` from `readxl` package!
    - Specify cells with contiguous range with `range =`

```
edData <- read_excel("../datasets/censusEd.xlsx", sheet = "EDU01A",
                     range = cell_cols("A:D"))
edData
```

```
## # A tibble: 3,198 x 4
##   Area_name     STCOU EDU010187F EDU010187D
##   <chr>         <chr>      <dbl>      <dbl>
## 1 UNITED STATES 00000          0   40024299
## 2 ALABAMA       01000          0     733735
## 3 Autauga, AL   01001          0       6829
## 4 Baldwin, AL   01003          0      16417
## 5 Barbour, AL   01005          0       5071
## # … with 3,193 more rows
```

SAS Data

- SAS data has extension '.sas7bdat'
- Read in smoke2003.sas7bdat
- Use `read_sas()` from `haven` package
- Not many options!

SAS Data

- SAS data has extension '.sas7bdat'
- Read in smoke2003.sas7bdat
- Use `read_sas()` from `haven` package
- Not many options!

```
#install if necessary
library(haven)
smokeData <- read_sas("https://www4.stat.ncsu.edu/~online/datasets/smoke2003.sas7bdat")
smokeData
```

```
## # A tibble: 443 x 54
##    SEQN SDDSRVYR RIDSTATR RIDEXMON RIAGENDR RIDAGEYR RIDAGEMN RIDAGEEX RIDRETH1
##   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 21010        3        2        2        2       52      633      634        3
## 2 21012        3        2        2        1       63      765      766        4
## 3 21048        3        2        1        2       42      504      504        1
## 4 21084        3        2        1        2       57      692      693        3
## 5 21093        3        2        1        2       64      778      778        2
## # … with 438 more rows, and 45 more variables: RIDRETH2 <dbl>, DMQMILIT <dbl>,
## #   DMDBORN <dbl>, DMDCITZN <dbl>, DMDYRSUS <dbl>, DMDEDUC3 <dbl>,
## #   DMDEDUC2 <dbl>, DMDEDUC <dbl>, DMDSCHOL <dbl>, DMDMARTL <dbl>,
## #   DMDHHSIZ <dbl>, INDHHINC <dbl>, INDFMINC <dbl>, INDFMPIR <dbl>,
## #   RIDEXPRG <dbl>, DMDHRGND <dbl>, DMDHRAGE <dbl>, DMDHRBRN <dbl>,
## #   DMDHREDU <dbl>, DMDHRMAR <dbl>, DMDHSEDU <dbl>, SIALANG <dbl>,
## #   SIAPROXY <dbl>, SIAINTRP <dbl>, FIALANG <dbl>, FIAPROXY <dbl>,
## #   FIAINTRP <dbl>, MIALANG <dbl>, MIAPROXY <dbl>, MIAINTRP <dbl>,
## #   AIALANG <dbl>, WTINT2YR <dbl>, WTMEC2YR <dbl>, SDMVPSU <dbl>,
## #   SDMVSTRA <dbl>, Gender <dbl>, Age <dbl>, IncomeGroup <chr>,
## #   Ethnicity <chr>, Education <dbl>, SMD070 <dbl>, SMQ077 <dbl>, SMD650 <dbl>,
## #   PacksPerDay <dbl>, lbdvid <dbl>
```

SAS Data

- Note: Variables had SAS labels. Don't show on print!
    - Will show on `View(smokeData)` (or click on data from environment)

```
str(smokeData)
```

```
## tibble [443 × 54] (S3: tbl_df/tbl/data.frame)
##  $ SEQN       : num [1:443] 21010 21012 21048 21084 21093 ...
##   ..- attr(*, "label")= chr "Patient ID"
##  $ SDDSRVYR   : num [1:443] 3 3 3 3 3 3 3 3 3 3 ...
##   ..- attr(*, "label")= chr "Data Release Number"
##  $ RIDSTATR   : num [1:443] 2 2 2 2 2 2 2 2 2 2 ...
##   ..- attr(*, "label")= chr "Interview/Examination Status"
##  $ RIDEXMON   : num [1:443] 2 2 1 1 1 2 1 2 1 1 ...
##   ..- attr(*, "label")= chr "Six month time period"
##  $ RIAGENDR   : num [1:443] 2 1 2 2 2 2 1 2 1 2 ...
##   ..- attr(*, "label")= chr "Gender 1=M 2=F"
##  $ RIDAGEYR   : num [1:443] 52 63 42 57 64 63 66 60 65 47 ...
##   ..- attr(*, "label")= chr "Age in Years at Exam"
##  $ RIDAGEMN   : num [1:443] 633 765 504 692 778 763 801 731 786 573 ...
##   ..- attr(*, "label")= chr "Age in Months - Recode"
##  $ RIDAGEEX   : num [1:443] 634 766 504 693 778 763 801 732 787 573 ...
##   ..- attr(*, "label")= chr "Exam Age in Months - Recode"
##  $ RIDRETH1   : num [1:443] 3 4 1 3 2 3 1 3 3 3 ...
##   ..- attr(*, "label")= chr " Ethnicity 1=MexAm 2=OthHisp 3=OthCauc 4=OthBla 5=Oth"
##  $ RIDRETH2   : num [1:443] 1 2 3 1 5 1 3 1 1 1 ...
##   ..- attr(*, "label")= chr "Linked NH3 Race/Ethnicity - Recode"
##  $ DMQMILIT   : num [1:443] 2 2 2 2 2 2 2 2 1 2 ...
##   ..- attr(*, "label")= chr "Veteran/Military Status"
##  $ DMDBORN    : num [1:443] 1 1 1 1 3 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "Country of Birth - Recode"
##  $ DMDCITZN   : num [1:443] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "Citizenship Status"
##  $ DMDYRSUS   : num [1:443] NA NA NA NA 9 NA NA NA NA NA ...
##   ..- attr(*, "label")= chr "Length of time in US"
##  $ DMDEDUC3   : num [1:443] NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "label")= chr "Education Level - Children/Youth 6-19"
##  $ DMDEDUC2   : num [1:443] 4 3 3 4 1 3 1 4 4 4 ...
##   ..- attr(*, "label")= chr "Education Level for Over 20"
##  $ DMDEDUC    : num [1:443] 3 2 2 3 1 2 1 3 3 3 ...
##   ..- attr(*, "label")= chr "Education - Recode (old version)"
##  $ DMDSCHOL   : num [1:443] NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "label")= chr "Now attending school?"
##  $ DMDMARTL   : num [1:443] 6 6 3 1 2 1 6 3 1 1 ...
##   ..- attr(*, "label")= chr "Marital Status"
```

SAS Data

- Note: Variables had SAS labels. Don’t show on print!
    - Will show on `View(smokeData)` (or click on data from environment)
    - Can access via

```
attr(smokeData$SDDSRVYR, "label")
```

```
## [1] "Data Release Number"
```

SPSS Data

- SPSS data has extension ".sav"
- Read in bodyFat.sav
- Use `read_spss()` from `haven` package
- Not many options!

SPSS Data

- SPSS data has extension ".sav"
- Read in bodyFat.sav
- Use `read_spss()` from `haven` package
- Not many options!

```
bodyFatData <- read_spss("https://www4.stat.ncsu.edu/~online/datasets/bodyFat.sav")
bodyFatData
```

```
## # A tibble: 20 x 4
##        y    x1    x2    x3
##    <dbl> <dbl> <dbl> <dbl>
##  1  19.5  43.1  29.1  11.9
##  2  24.7  49.8  28.2  22.8
##  3  30.7  51.9  37    18.7
##  4  29.8  54.3  31.1  20.1
##  5  19.1  42.2  30.9  12.9
##  6  25.6  53.9  23.7  21.7
##  7  31.4  58.5  27.6  27.1
##  8  27.9  52.1  30.6  25.4
##  9  22.1  49.9  23.2  21.3
## 10  25.5  53.5  24.8  19.3
## 11  31.1  56.6  30    25.4
## 12  30.4  56.7  28.3  27.2
## 13  18.7  46.5  23    11.7
## 14  19.7  44.2  28.6  17.8
## 15  14.6  42.7  21.3  12.8
## 16  29.5  54.4  30.1  23.9
## 17  27.7  55.3  25.7  22.6
## 18  30.2  58.6  24.6  25.4
## 19  22.7  48.2  27.1  14.8
## 20  25.2  51    27.5  21.1
```

Recap

- Reading Data

Type of file	Package	Function
Delimited	`readr`	`read_csv()`, `read_tsv()`, `read_table()`, `read_delim()`
Excel (.xls,.xlsx)	`readxl`	`read_excel()`
SAS (.sas7bdat)	`haven`	`read_sas()`
SPSS (.sav)	`haven`	`read_spss()`

Resources for Other Data Sources

JSON - JavaScript Object Notation

- Used widely across the internet and databases
- Can represent usual 2D data or heirarchical data

Resources for Other Data Sources

JSON - JavaScript Object Notation

- Uses key-value pairs

```
{
  {
    "name": "Barry Sanders"
    "games" : 153
    "position": "RB"
  },
  {
    "name": "Joe Montana"
    "games": 192
    "position": "QB"
  }
}
```

Resources for Other Data Sources

JSON - JavaScript Object Notation

Three major R packages

1. `rjson`
2. `RJSONIO`
3. `jsonlite`
   - many nice features
   - a little slower implementation

Resources for Other Data Sources

JSON - JavaScript Object Notation

`jsonlite` basic functions:

Function	Description
`fromJSON`	Reads JSON data from file path or character string. Converts and simplfies to R object
`toJSON`	Writes R object to JSON object
`stream_in`	Accepts a *file connection* - can read streaming JSON data

---

Resources for Other Data Sources

APIs - Application Programming Interfaces

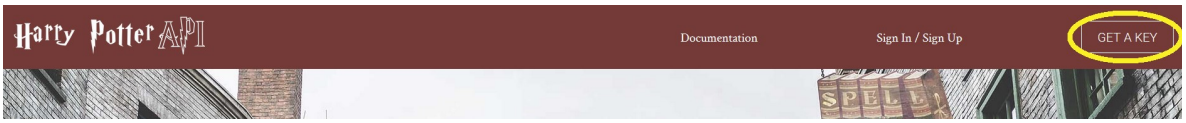
A defined method for asking for information from a computer

- Useful for getting data
- Useful for allowing others to run your model without a GUI (like Shiny)
- Many open APIs, just need key
- Often just need to construct proper URL

Resources for Other Data Sources

APIs - Quick Example

- Query Harry Potter database https://www.potterapi.com/
- Get key in top right (sign up for account)



Resources for Other Data Sources

APIs - Quick Example

- Query Harry Potter database https://www.potterapi.com/
- Documentation:
  - All routes need to be prefixed with https://www.potterapi.com/v1/
  - GET request: /spells returns all spells
  - Key goes on the end

```
baseURL <- "https://www.potterapi.com/v1/"
value <- "spells?"
key <- "key=$2a$10$UMvDCH.93fa2KOjKbJYkOOPMNzdzQpJ0gMnVEtcHzW5Ic04HUmcsa"
URL <- paste0(baseURL, value, key)
spellData <- RCurl::getURL(URL)
```

Resources for Other Data Sources

APIs - Quick Example

- Query Harry Potter database https://www.potterapi.com/
- Default response format is JSON

```
spellData
```

```
## [1] "[{\"_id\":\"5b74ebd5fb6fc0739646754c\",\"spell\":\"Aberto\",\"type\":\"Charm\",\"effect
```

Resources for Other Data Sources

APIs - Quick Example

- Query Harry Potter database https://www.potterapi.com/
- Default response format is JSON

```
spellDataDF <- jsonlite::fromJSON(spellData)
as_tibble(spellDataDF)
```

```
## # A tibble: 151 x 5
##   `_id`                spell          type      effect                      `__v`
##   <chr>                <chr>          <chr>     <chr>                       <int>
## 1 5b74ebd5fb6fc073964… Aberto         Charm     opens objects                  NA
## 2 5b74ecfa3228320021a… Accio          Charm     Summons an object               0
## 3 5b74ed2f3228320021a… Age Line       Enchantm… Hides things from younger…      0
## 4 5b74ed453228320021a… Aguamenti      Charm     shoots water from wand          0
## 5 5b74ed583228320021a… Alarte Ascend… Spell     shoots things high in the…      0
## # … with 146 more rows
```

Resources for Other Data Sources

APIs - Application Programming Interfaces

Access in R

- Article here discusses accessing APIs generically with R
- Same website gives a list of APIs

Resources for Other Data Sources

XML - eXtensible Markup Language

- Used widely across the internet and databases
- Can represent usual 2D data or heirarchical data

Resources for Other Data Sources

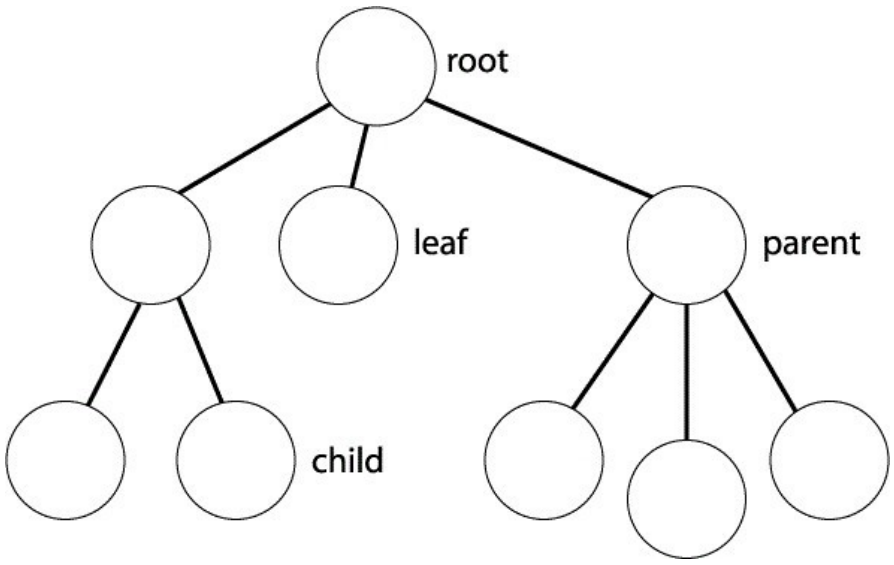
XML - eXtensible Markup Language

- Uses tags < > (similar to HTML)

```
<roster>
  <player>
    <name>Barry Sanders</name>
    <games>153</games>
    <position>RB</position>
  </player>
  <player>
    <name>Joe Montana</name>
    <games>192</games>
    <position>QB</position>
  </player>
</roster>
```

Resources for Other Data Sources

XML - eXtensible Markup Language



Source: mysamplecode.com

Resources for Other Data Sources

XML - eXtensible Markup Language

Two major R packages

1. `XML`
    - pretty well developed
2. `xml2`
    - basic functionality to get data into R

- Reading XML data generally tough since structure of tags varies by data source!

Resources for Other Data Sources

`XML` package basic functions:

Function	Description
`xmlParseDoc`	Parse an XML document
`xmlChildren`	Returns a list of child XMLNode objects
`xmlParent`	Returns a reference to a parent node
`xmlAttrs`	Returns a named character vector giving the name-value pairs of attributes of an XMLNodeobject
`xmlName`	Returns the name associated with an XMLNode object
`xmlToList`	Converts XML node/document to a more R-like list

Quick XML Example (Can be tricky!)

```
returned <- ZillowR::GetDeepSearchResults(address = "14707 W SUNNY DR",
                                citystatezip = "Los Angeles, CA", zws_id = "...")
```

```
returned
```

```
## $request
## $request$address
## [1] "14707 W SUNNY DR"
##
## $request$citystatezip
## [1] "Los Angeles, CA"
##
##
## $message
## $message$text
## [1] "Request successfully processed"
##
## $message$code
## [1] "0"
##
##
## $response
## <response>
##  <results>
##   <result>
##    <zpid>72106858</zpid>
##    <links>
##     <homedetails>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/72106858_z
##     <graphsanddata>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/7210685
##     <mapthishome>http://www.zillow.com/homes/72106858_zpid/</mapthishome>
##     <comparables>http://www.zillow.com/homes/comps/72106858_zpid/</comparables>
##    </links>
##    <address>
##     <street>14707 Sunny Dr</street>
##     <zipcode>91342</zipcode>
##     <city>Sylmar</city>
##     <state>CA</state>
##     <latitude>34.30136</latitude>
##     <longitude>-118.453601</longitude>
##    </address>
##    <FIPScounty>6037</FIPScounty>
##    <useCode>SingleFamily</useCode>
##    <taxAssessmentYear>2019</taxAssessmentYear>
##    <taxAssessment>476290.0</taxAssessment>
##    <yearBuilt>2006</yearBuilt>
```

Quick XML Example (Can be tricky!)

```
returned$response


## <response>
##  <results>
##   <result>
##    <zpid>72106858</zpid>
##    <links>
##     <homedetails>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/72106858_z
##     <graphsanddata>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/7210685
##     <mapthishome>http://www.zillow.com/homes/72106858_zpid/</mapthishome>
##     <comparables>http://www.zillow.com/homes/comps/72106858_zpid/</comparables>
##    </links>
##    <address>
##     <street>14707 Sunny Dr</street>
##     <zipcode>91342</zipcode>
##     <city>Sylmar</city>
##     <state>CA</state>
##     <latitude>34.30136</latitude>
##     <longitude>-118.453601</longitude>
##    </address>
##    <FIPScounty>6037</FIPScounty>
##    <useCode>SingleFamily</useCode>
##    <taxAssessmentYear>2019</taxAssessmentYear>
##    <taxAssessment>476290.0</taxAssessment>
##    <yearBuilt>2006</yearBuilt>
##    <lotSizeSqFt>43342</lotSizeSqFt>
##    <finishedSqFt>1984</finishedSqFt>
##    <bathrooms>3.0</bathrooms>
##    <bedrooms>4</bedrooms>
##    <lastSoldDate>06/12/2015</lastSoldDate>
##    <lastSoldPrice currency="USD">425000</lastSoldPrice>
##    <zestimate>
##     <amount currency="USD">563157</amount>
##     <last-updated>09/06/2020</last-updated>
##     <oneWeekChange deprecated="true"/>
##     <valueChange duration="30" currency="USD">8122</valueChange>
##     <valuationRange>
##      <low currency="USD">534999</low>
##      <high currency="USD">591315</high>
##     </valuationRange>
##     <percentile>0</percentile>
##    </zestimate>
##    <localRealEstate>
##     <region name="Sylmar" id="34213" type="neighborhood">
##      <zindexValue>473,900</zindexValue>
```

Quick XML Example (Can be tricky!)

```
#parse it via xml functions
rawDataValues <- xmlChildren(xmlChildren(xmlChildren(returned$response)$results)$result)

rawDataValues


## $zpid
## <zpid>72106858</zpid>
##
## $links
## <links>
##  <homedetails>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/72106858_zpid
##  <graphsanddata>http://www.zillow.com/homedetails/14707-Sunny-Dr-Sylmar-CA-91342/72106858_zp
##  <mapthishome>http://www.zillow.com/homes/72106858_zpid/</mapthishome>
##  <comparables>http://www.zillow.com/homes/comps/72106858_zpid/</comparables>
## </links>
##
## $address
## <address>
##  <street>14707 Sunny Dr</street>
##  <zipcode>91342</zipcode>
##  <city>Sylmar</city>
##  <state>CA</state>
##  <latitude>34.30136</latitude>
##  <longitude>-118.453601</longitude>
## </address>
##
## $FIPScounty
## <FIPScounty>6037</FIPScounty>
##
## $useCode
## <useCode>SingleFamily</useCode>
##
## $taxAssessmentYear
## <taxAssessmentYear>2019</taxAssessmentYear>
##
## $taxAssessment
## <taxAssessment>476290.0</taxAssessment>
##
## $yearBuilt
## <yearBuilt>2006</yearBuilt>
##
## $lotSizeSqFt
## <lotSizeSqFt>43342</lotSizeSqFt>
##
## $finishedSqFt
```

Quick XML Example (Can be tricky!)

```
#simplify the remaining xml structure to a vector
rawDataVector <- unlist(sapply(xmlToList, X = rawDataValues))

#grab data of interest if it is there by using names
dataVector <- rawDataVector[c("address.street", "address.zipcode", "address.city",
                              "useCode", "taxAssessmentYear")]
dataVector
```

```
##    address.street   address.zipcode      address.city           useCode
##  "14707 Sunny Dr"           "91342"          "Sylmar"    "SingleFamily"
## taxAssessmentYear
##            "2019"
```

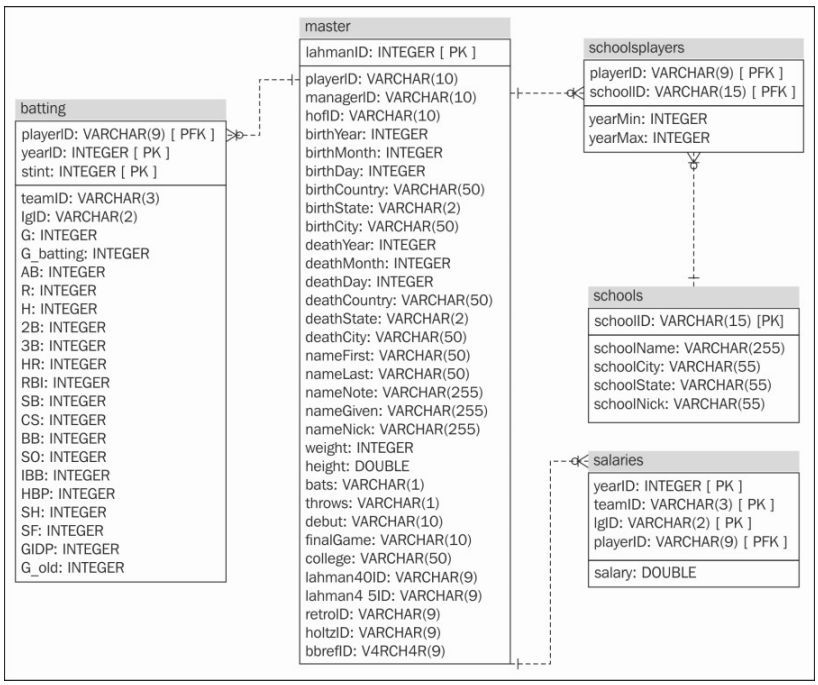
Resources for Other Data Sources

Databases

- Collection of data, usually a bunch of (related) 2D tables

Resources for Other Data Sources

Example database structure



Source: oreilly.com

Resources for Other Data Sources

Databases

- Collection of data, usually a bunch of (related) 2D tables
- Relational Database Management System (RDBMS) controls how users interact
- Structured Query Language (SQL) - language used by RDBMS

Resources for Other Data Sources

Databases

- Collection of data, usually a bunch of 2D tables
- Relational Database Management System (RDBMS) controls how users interact
- Structured Query Language (SQL) - language used by RDBMS
  - Used to obtain data from tables
  - Used to combine data from separate tables (‘keys’ relate tables)
  - Used to manipulate and create variables, structure and edit databases, etc.

Resources for Other Data Sources

Databases

Many popular RDBMS, some free some proprietary (often referred to as databases…)

- Oracle - most popular (cross platform)
- SQL Server - Microsoft product
- DB2 - IBM product
- MySQL (open source) - Not as many features but popular
- PostgreSQL (open source)

Basic SQL language constant across all - features differ

Resources for Other Data Sources

Databases - Common flow in R

1. Connect to the database with `DBI::dbConnect()`

- Need appropriate R package for database backend
  - `RSQLite::SQLite()` for RSQLite
  - `RMySQL::MySQL()` for RMySQL
  - `RPostgreSQL::PostgreSQL()` for RPostgreSQL
  - `odbc::odbc()` for Open Database Connectivity
  - `bigrquery::bigquery()` for google's bigQuery

```
con <- DBI::dbConnect(RMySQL::MySQL(),
  host = "hostname.website",
  user = "username",
  password = rstudioapi::askForPassword("DB password")
)
```

Resources for Other Data Sources

Databases - Common flow in R

1. Connect to the database with `DBI::dbConnect()`

- Need appropriate R package for database backend

2. Use `tbl()` to reference a table in the database

```
tbl(con, "name_of_table")
```

Resources for Other Data Sources

Databases - Common flow in R

1. Connect to the database with `DBI::dbConnect()`

- Need appropriate R package for database backend

2. Use `tbl()` to reference a table in the database
3. Query the database with `SQL` or `dplyr/dbplyr` (we'll learn `dplyr` soon!)

Resources for Other Data Sources

Databases - Common flow in R

1. Connect to the database with `DBI::dbConnect()`

- Need appropriate R package for database backend

2. Use `tbl()` to reference a table in the database
3. Query the database with `SQL` or `dplyr/dbplyr` (we'll learn `dplyr` soon!)
4. Disconnect from database with `dbDisconnect()`

Resources for Other Data Sources

Databases - Quick Example

- Connect to Google's BigQuery database

```
#devtools::install_github("r-dbi/bigrquery")
library(DBI)
con <- dbConnect(
  bigrquery::bigquery(),
  project = "publicdata",
  dataset = "samples",
  billing = "your-project-id-here"
  )
```

Resources for Other Data Sources

Databases - Quick Example

- Connect to Google's BigQuery database

```
dbListTables(con)
natality <- tbl(con, "natality")

natality %>%
  select(starts_with("mother"), year, cigarette_use, weight_pounds) %>%
  collect()

dbDisconnect(con)
```

- More about R Studio and Databases

Recap

- Read data from other sources

Type of file	Package	Function
Delimited	`readr`	`read_csv()`, `read_tsv()`, `read_table()`, `read_delim()`
Excel (.xls,.xlsx)	`readxl`	`read_excel()`
SAS (.sas7bdat)	`haven`	`read_sas()`
SPSS (.sav)	`haven`	`read_spss()`

- Resources for JSON, XML, databases, and APIs