Introduction to R/RStudio & reforking or recloning (CWRU, Pitt, UCF, UTRGV)

Profs: R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

TAs: W. Oltjen, K. Hernandez, M. Li, M. Li, D. Colvin

08 September, 2022

Contents

2.2.2.1 First we setup some global Markdown options

2.2.2.2 Class Readings, Assignments, Syllabus Topics

2.2.2.2.1 Reading, Lab Exercises, SemProjects

- Readings:
  - For today: OIS 1,2
  - For next class: PRP94-119
- Laboratory Exercises:
  - LE1 : **Due Tuesday 9/13/2022**
  - LE2 : Handed out Tuesday 9/13/2022
- Office Hours: (Class Canvas Calendar for Zoom Link)
  - Wednesday @ 4:00 PM to 5:00 PM, Will Oltjen
  - Saturday @ 3:00 PM to 4:00 PM, Kristen Hernandez
  - **Office Hours are on Zoom, and recorded**
- Semester Projects

  - DSCI 451 Students **Biweekly Update 1 Due Tomorrow, Friday 9/9/22**
  - DSCI 451 Students
    * Next **Report Out #1 is Due Friday September30th**
  - All DSCI 351/351M/451 Students:
    * **Peer Grading of Report Out # is Due October 11th, 2022**
  - Exams
    * MidTerm: Tuesday October 18th, in class or remote, 11:30 - 12:45 PM
    * Final: Monday December 19, 2022, 12:00PM - 3:00PM, Nord 356 or remote

2.2.2.2.2 Textbooks Introduction to R and Data Science

- For R, Coding, Inferential Statistics
  - Peng: R Programming for Data Science
  - Peng: Exploratory Data Analysis with R

Textbooks for this class

- OIS = Diez, Barr, Çetinkaya-Runde: Open Intro Stat v4
- R4DS = Wickham, Grolemund: R for Data Science

Textbooks for DSCI353/353M/453, And in your Repo now

- ISLR = James, Witten, Hastie, Tibshirani: Intro to Statistical Learning with R 2nd Ed.
- ESL = Trevor Hastie, Tibshirani, Friedman: Elements of Statistical Learning
- DLwR = Chollet, Allaire: Deep Learning with R

Magazine Articles about Deep Learning

- DL1 to DL13 are "Deep Learning" articles in 3-readings/2-articles/

2.2.2.2.3 Syllabus

2.2.2.3 As we are onboarding ~ 106 students

- Here is a picture of what we are setting up as our Git Environment for Class

And here is what we have all been doing

- Git Fork the prof repo
  - And change "-prof" to "-caseid"
- Git Clone your class repo, to your computers
  - Markov Data Science Cluster
  - ODS Desktop
  - Your personal notebook computer
    * For convenient reading of pdf docs in the repo

There is a .txt file in the root directory of our class repo

- Filename: HowToReforkTheProfRepo&RecloneYourPersonalClassRepo.txt

These are the instructions to reFork or reClone

- Other .txt files in the root directory of your course repo
- Are various solutions to problems you may have.

==========

2.2.2.4 Now lets start our Intro to R, from OISv4

Day:Date	Foundation	Practicum	Reading	Due
w01a:Tu:8/30/22	ODS Tool Chain	R, Rstudio, Git		
w01b:Th:9/1/22	Setup ODS Tool Chain	Bash, Git, Slack, Agile	PRP4-33	LE1
w02a:Tu:9/6/22	Bash-Git-Knuth-Lit.Prog.	RIntroR	PRP35-64	
w02b:Th:9/8/22	What is Data Science	OIS:Intro2R	OIS1,2	
w02Pr:Fr:9/9/22			PRP65-93	**451 Update1**
w03a:Tu:9/13/22	Data Intro	Data Analytic Style	PRP94-116	LE2 **LE1 Due**
w03b:Th:9/15/22	Rand. Var. Normal Dist.	Git, Rmds, Loops	OIS4	
w04a:Tu:9/20/22	Tidy Check Explore	Tidy GapMinder	EDA1-31	
w04b:Th:9/22/22	Inference, DSCI Process	Other Distrib. 7 ways	R4DS1-3	LE3 **LE2 Due**
w04Pr:Fr:9/23/22			EDA32-58	**451 Update2**
w05a:Tu:9/27/22	OIS4 Rand. Var.	EDA of PET Degr.	OIS5	
w05b:Th:9/29/22	OIS5 Found. of Infer.	Multivar Corr. Plot	R4DS4-6	
w05Pr:Fr:9/30/22				**451 RepOut1**
w06a:Tu:10/4/22	Pred., Algorithm, Model	Anscombe's Quartets	R4DS7-8	
w06b:Th:10/6/22	EDA stats, vis	Summ. Stats & Vis.	R4DS9-16	LE4 **LE3 Due**
w06Pr:Fr:10/7/22	Corr. Coeff. Pairs Plots			**451 Update3**
w07a:Tu:10/11/22	Confidence Intervals	Penguins	OIS6.1-2	**PeerRv1 Due**
w07b:Th:10/13/22	Midterm Rev.	Hypo.Test, Sampl. Dist.		
w08a:Tu:10/18/22	**MIDTERM**	**EXAM**		
w08b:Th:10/20/22	Programming & Coding	Coding Expect.		**LE4 Due**
w08Pr:Fr:10/21/22				**451 Update4**
Tu:10/24,25	**CWRU**	**FALL BREAK**	R4DS17-21	
w09b:Th:10/27/22	Cat. Inf. 1 & 2 propor.	Indep. Test,2-way tables	OIS6.3-4	LE5
w09Pr:Fr:10/28/22				**451 RepOut2**
w10a:Tu:11/1/22	Goodness of Fit, χ^2 test	t-tests 1&2 means	OIS7.1-4	
w10b:Th:11/3/22	Num. Infer, Cont. Tables	Stat. Power		
w10Pr:Fr:11/4/22				**451 Update5**
w11a:Tu:11/8/22	Sample & Effect Size	Stat. Power GGmap	OIS8	**PeerRv2 Due**
w11b:Th:11/10/22	Inf. 4 Regr, Test & Train	Curse of Dimen.	ISLR1,2.1,2	LE6 **LE5 Due**
w12a:Tu:11/15/22	Lin. Regr. Part 1	Residuals	OIS9	
w12b:Th:11/17/22	Lin. Regr. Part 2	Regr. Diagnostics		
w12Pr:Fr:11/18/22				**451 Update6**
w13a:Tu:11/22/22	Mult. Lin. Regr.	Var. & Mod. Selec.,	ISLR3.1	LE7 **LE6 due**
w13b:Th:11/24/22	Log. Regr.	GIS Trends	ISLR3.2	
w13Pr:Fr:11/25/22				**451 RepOut3**
w14a:Tu:11/23/22	Classificat., Sup. Lrning	Caret, Broom 4 modeling	ISLR4.1-3	
Th,Fr:11/24,25	**THANKSGIVIING**	**Vacation**		
w15a:Tu:11/29/22		Clustering		**PeerRv3 Due**
w15b:Th:12/1/22	Big Data Analytics	Dist. Comp., Hadoop		
w15SPr:Fr:12/2/22		Read Article by	Mirletz,2015	
w16a:Tu:12/6/22	Final Exam Review			
w15b:Th:12/8/22				**LE7 due**
Friday 12/12	**SemProj**	**Final Report**		**SemProj4 due**
Monday 12/19	**FINAL EXAM**	**12:00-3:00pm**	Nord 356	or remote

Figure 1: DSCI351-351M-451 Syllabus

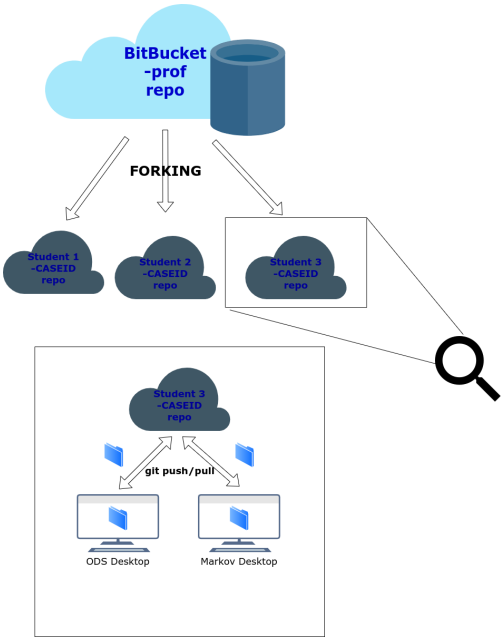


Figure 2: Our compute and repo architecture

2.2.2.4.1 First lets confirm our `.libPaths()` is correct

- A good first thing to check is your `libPaths()`
  - You need to check that you have the FIRST directory in the list
    - `/home/rxf131/ondemand/ubuntu2004/r4`
  - If its not, go to the file in your root directory
    - `FixRstudioServer-R-libPaths.txt`
  - And you will then run this command
    - In your R Console
    - `source('/home/rxf131/ondemand/share/config/r-lib-path-fix.R')`
  - And then recheck your `.libPaths()`

```
.libPaths()
```

```
## [1] "/home/frenchrh/R/x86_64-pc-linux-gnu-library/4.2"
## [2] "/usr/local/lib/R/site-library"
## [3] "/usr/lib/R/site-library"
## [4] "/usr/lib/R/library"
```

2.2.2.4.2 Lets load some R Packages

Lets "library in", or make available

- Two R packages
  - One is a metapackage of tidyverse capabilities
  - The other is a package of data and functions from our OIS book

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.7      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(openintro)
```

```
## Loading required package: airports
```

```
## Loading required package: cherryblossom
```

```
## Loading required package: usdata
```

2.2.2.5 The RStudio Interface

- The goal of this lab is to introduce you to R and RStudio,
  - which we'll be using throughout the course both
    - to learn the statistical concepts discussed in the course
    - and to analyze real data and come to informed conclusions.
  - To clarify which is which:
    - R is the name of the programming language itself
    - and RStudio is an integrated development environment (IDE).

As the labs progress,

- you are encouraged to explore beyond what the labs dictate;
  - a willingness to experiment will make you a much better programmer.

Before we get to that stage, however,

- you need to build some basic fluency in R.

Today we begin with the fundamental building blocks of R and RStudio:

- the interface,
- reading in data,
- and basic commands.

Go ahead and launch RStudio.

You should see a window that looks like the image shown below.

The panel on the lower left is where the action happens.

- It's called the *console*.
- Everytime you launch RStudio,
  - it will have the same text at the top of the console
  - telling you the version of R that you're running.
- Below that information is the *prompt*.
  - As its name suggests, this prompt is really a request:
  - a request for a command.
- Initially, interacting with R
  - is all about typing commands and interpreting the output.
- These commands and their syntax have evolved over decades (literally)
  - and now provide what many users feel
  - is a fairly natural way to access data
  - and organize, describe,
  - and invoke statistical computations.

The panel in the upper right

- contains your *environment*

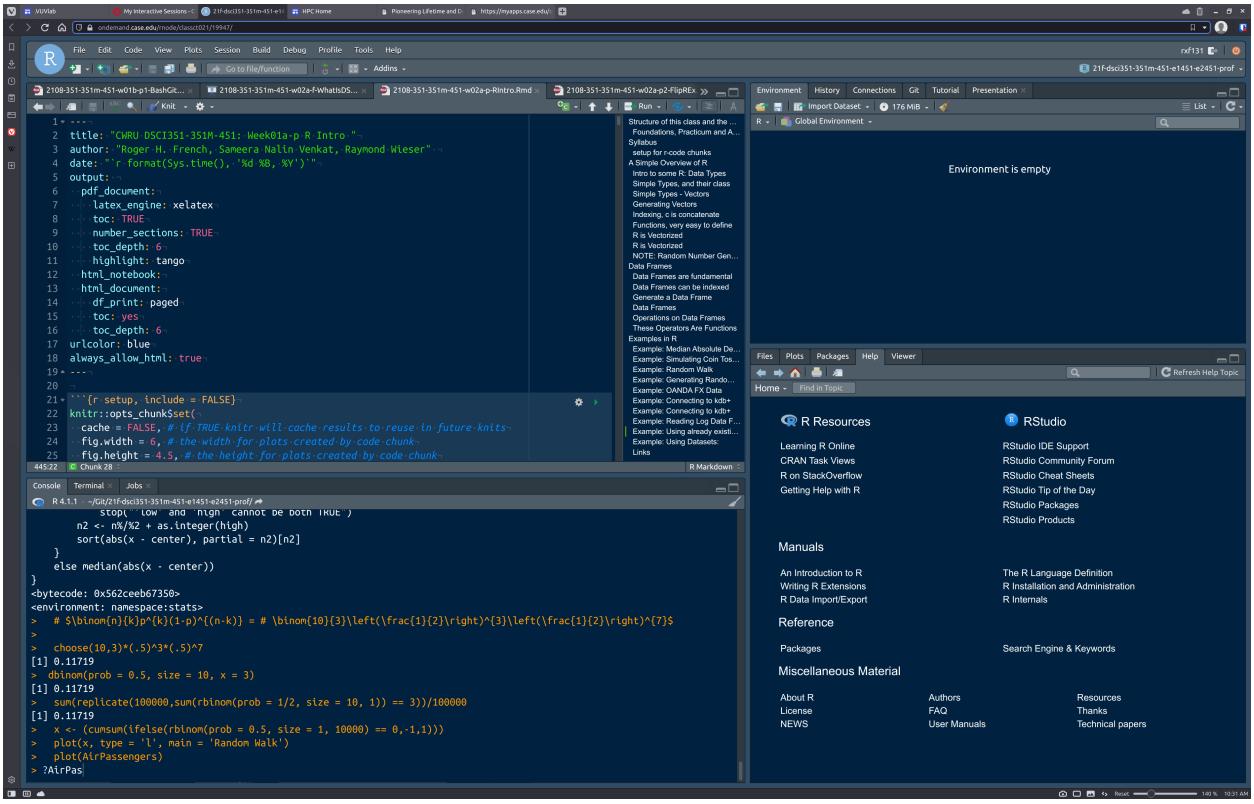


Figure 3: Rstudio Interface

- as well as a history of the commands
  - that you've previously entered.

Any plots that you generate will show up

- in the panel in the lower right corner.
- This is also where you can
  - browse your files,
  - access help,
  - manage packages, etc.

2.2.2.5.1 R Packages

- R is an open-source programming language,
  - meaning that users can contribute packages that make our lives easier,
    - and we can use them for free.

For this lab, and many others in the future,

- we will use the following R packages:
- The suite of **tidyverse** packages:
  - for data wrangling and data visualization
  - **openintro**: for data and custom functions with the OpenIntro resources

If these packages were not already available in your R environment,

- then you would install them by typing the following three lines of code
  - into the console of your RStudio session,

    - pressing the enter/return key after each one.
- Note that you can check to see
    - which packages (and which versions) are installed
    - by inspecting the *Packages* tab
    - in the lower right panel of RStudio.

```
# install.packages("tidyverse")
# install.packages("openintro")
```

You may be asked to select a server from which to download;

- any of them will work.

Next, you need to load these packages

- in your working "R environment".
- We do this with the `library` function.

Run the following three lines in your console.

```
library(tidyverse)
library(openintro)
```

You only need to *install* packages once,

- but you need to *load* them each time you relaunch RStudio.

The Tidyverse packages

- share common philosophies
- and are designed to work together.

You can find more about the packages in the tidyverse

- at tidyverse.org.

2.2.2.5.2 Creating a reproducible lab report

- We will be using R Markdown to create reproducible lab reports.
    - See the following videos describing why and how:
        * **Why use R Markdown for Lab Reports?**
        * **Using R Markdown for Lab Reports in RStudio**

In a nuthshell, in RStudio, go to New File -> R Markdown…

- Then, choose From Template and
- then choose `Lab Report for OpenIntro Statistics Lab 1`
    - from the list of templates.

Going forward you should refrain

- from typing your code directly in the console,
    - and instead type any code
    - (final correct answer, or anything you're just trying out)
    - in the R Markdown file
- and run the chunk using either
    - the Run button on the chunk (green sideways triangle)
    - or by highlighting the code and clicking Run
        * on the top right corner of the R Markdown editor.

If at any point you need to start over,

- you can Run All Chunks above the chunk you're working in

- by clicking on the down arrow in the code chunk.

2.2.2.6 Dr. Arbuthnot's Baptism Records

- To get started, let's take a peek at the data.

```
arbuthnot
```

```
## # A tibble: 82 x 3
##     year  boys girls
##    <int> <int> <int>
##  1  1629  5218  4683
##  2  1630  4858  4457
##  3  1631  4422  4102
##  4  1632  4994  4590
##  5  1633  5158  4839
##  6  1634  5035  4820
##  7  1635  5106  4928
##  8  1636  4917  4605
##  9  1637  4703  4457
## 10  1638  5359  4952
## # ... with 72 more rows
```

You can run the command by

- clicking on the green arrow at the top right of the code chunk in the R Markdown (Rmd) file, or
- putting your cursor on this line,
  - and clicking the **Run** button on the upper right corner of the pane, or
- holding `Ctrl-Shift-Enter`, or
- typing the code in the console.

This command instructs R to load some data:

- the Arbuthnot baptism counts for boys and girls.

You should see that the in the R environment area

- in the upper righthand corner of the RStudio window
- now lists a data set called `arbuthnot`
  - that has 82 observations on 3 variables.

As you interact with R, you will create a series of objects.

Sometimes you load them as we have done here,

- and sometimes you create them yourself
  - as the byproduct of a computation
  - or some analysis you have performed.

The Arbuthnot data set refers to the work of Dr. John Arbuthnot,

- an 18th century physician, writer, and mathematician.

He was interested in

- the ratio of newborn boys to newborn girls,
- so he gathered the baptism records for children born in London
  - for every year from 1629 to 1710.
- Once again, we can view the data by typing its name into the console.

```
arbuthnot
```

```
## # A tibble: 82 x 3
##     year  boys girls
##    <int> <int> <int>
##  1  1629  5218  4683
##  2  1630  4858  4457
##  3  1631  4422  4102
##  4  1632  4994  4590
##  5  1633  5158  4839
##  6  1634  5035  4820
##  7  1635  5106  4928
##  8  1636  4917  4605
##  9  1637  4703  4457
## 10  1638  5359  4952
## # ... with 72 more rows
```

However, printing the whole dataset in the console is not that useful.

- One advantage of RStudio is that it comes with a built-in data viewer.

Click on the name `arbuthnot` in the *Environment* pane

- (upper right window) that lists the objects in your environment.
- This will bring up an alternative display of the data set
  - in the *Data Viewer* (upper left window).
- You can close the data viewer by clicking on
  - the `x` in the upper lefthand corner.

What you should see are four columns of numbers,

- each row representing a different year:
- the first entry in each row is simply the row number
  - (an index we can use to access the data from individual years if we want),
- the second is the year,
- and the third and fourth are the numbers of boys and girls
  - baptized that year, respectively.

Use the scrollbar on the right side of the console window

- to examine the complete data set.

Note that the row numbers in the first column

- are not part of Arbuthnot's data.
- R adds them as part of its printout
  - to help you make visual comparisons.
- You can think of them as the index that you see
  - on the left side of a spreadsheet.
- In fact, the comparison to a spreadsheet will generally be helpful.

R has stored Arbuthnot's data

- in a kind of spreadsheet or table
- called a *data frame*.

You can see the dimensions of this data frame

- as well as the names of the variables
- and the first few observations by typing:

```
glimpse(arbuthnot)
```

```
## Rows: 82
## Columns: 3
## $ year  <int> 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639~
## $ boys  <int> 5218, 4858, 4422, 4994, 5158, 5035, 5106, 4917, 4703, 5359, 5366~
## $ girls <int> 4683, 4457, 4102, 4590, 4839, 4820, 4928, 4605, 4457, 4952, 4784~
```

`glimpse` is a tidyverse function to look at your data

It is better practice to type this command into your console,

- since it is not necessary code to include in your solution file.

This command should output the following

Rows: 82 Columns: 3 $ year 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639~ $ boys 5218, 4858, 4422, 4994, 5158, 5035, 5106, 4917, 4703, 5359, 5366~ $ girls 4683, 4457, 4102, 4590, 4839, 4820, 4928, 4605, 4457, 4952, 4784~

We can see that there are

- 82 observations
- and 3 variables in this dataset.

The variable names are

- `year`,
- `boys`,
- and `girls`.

At this point, you might notice

- that many of the commands in R
  - look a lot like functions from math class;
- that is, invoking R commands
  - means supplying a function with some number of arguments.
- The `glimpse` command, for example,
  - took a single argument,
  - the name of a data frame.

2.2.2.7 Some Exploration, or Exploratory Data Analysis (EDA)!

- Let's start to examine the data a little more closely.
  - We can access the data in
  - a single column of a data frame separately
    - using a command like

```
arbuthnot$boys
```

```
##  [1] 5218 4858 4422 4994 5158 5035 5106 4917 4703 5359 5366 5518 5470 5460 4793
## [16] 4107 4047 3768 3796 3363 3079 2890 3231 3220 3196 3441 3655 3668 3396 3157
## [31] 3209 3724 4748 5216 5411 6041 5114 4678 5616 6073 6506 6278 6449 6443 6073
## [46] 6113 6058 6552 6423 6568 6247 6548 6822 6909 7577 7575 7484 7575 7737 7487
## [61] 7604 7909 7662 7602 7676 6985 7263 7632 8062 8426 7911 7578 8102 8031 7765
## [76] 6113 8366 7952 8379 8239 7840 7640
```

This command will only show

- the number of boys baptized each year.

The dollar sign basically says

- "go to the data frame that comes before me,

  - and find the variable that comes after me".

1. What command would you use to extract just the counts of girls baptized?

- Try it!

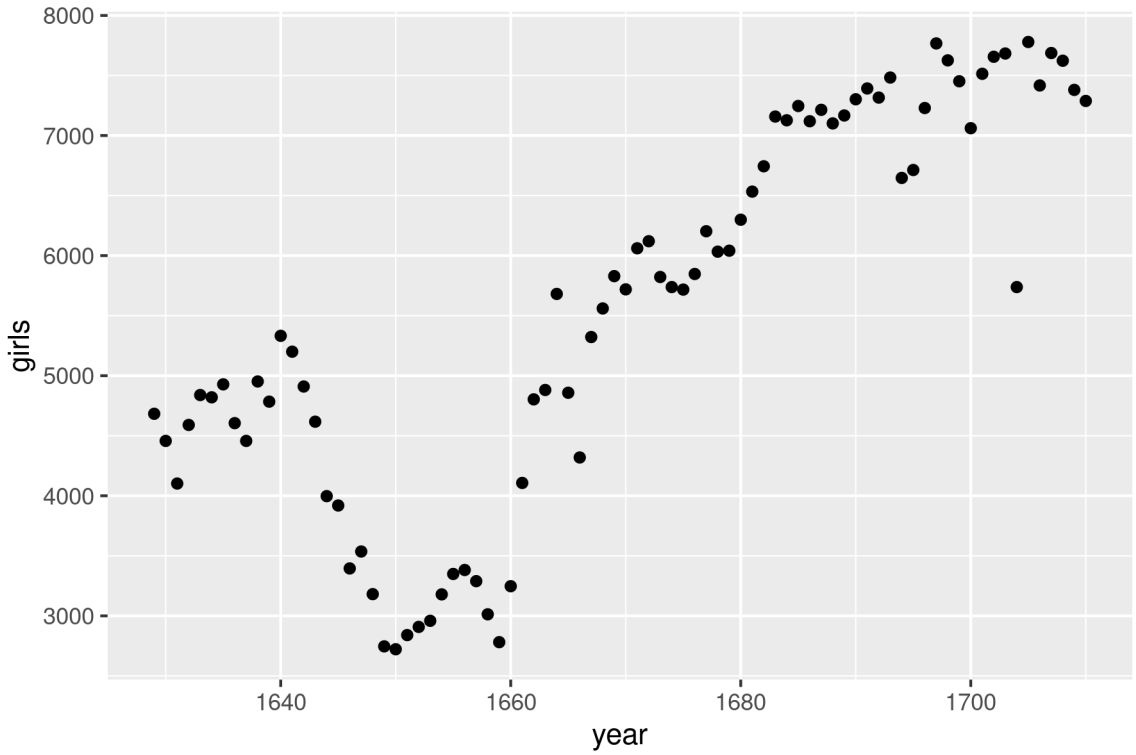
Notice that the way R has printed these data is different.

- When we looked at the complete data frame,
  - we saw 82 rows, one on each line of the display.
- These data are no longer structured in a table with other variables,
  - so they are displayed one right after another.
- Objects that print out in this way are called *vectors*;
  - they represent a set of numbers.
- R has added numbers in [brackets] along the left side of the printout
  - to indicate locations within the vector.
- For example, 5218 follows `[1]`,
  - indicating that `5218` is the first entry in the vector.
- And if `[43]` starts a line,
  - then that would mean the first number on that line
  - would represent the 43rd entry in the vector.

2.2.2.7.1 Data visualization

- R has some powerful functions for making graphics.
  - We can create a simple plot
    * of the number of girls baptized per year with the command

```
ggplot(data = arbuthnot, aes(x = year, y = girls)) +
  geom_point()
```



We use the `ggplot()` function to build plots.

- If you run the plotting code in your console,
  - you should see the plot appear under the *Plots* tab
  - of the lower right panel of RStudio.
- Notice that the command above again looks like a function,
  - this time with arguments separated by commas.

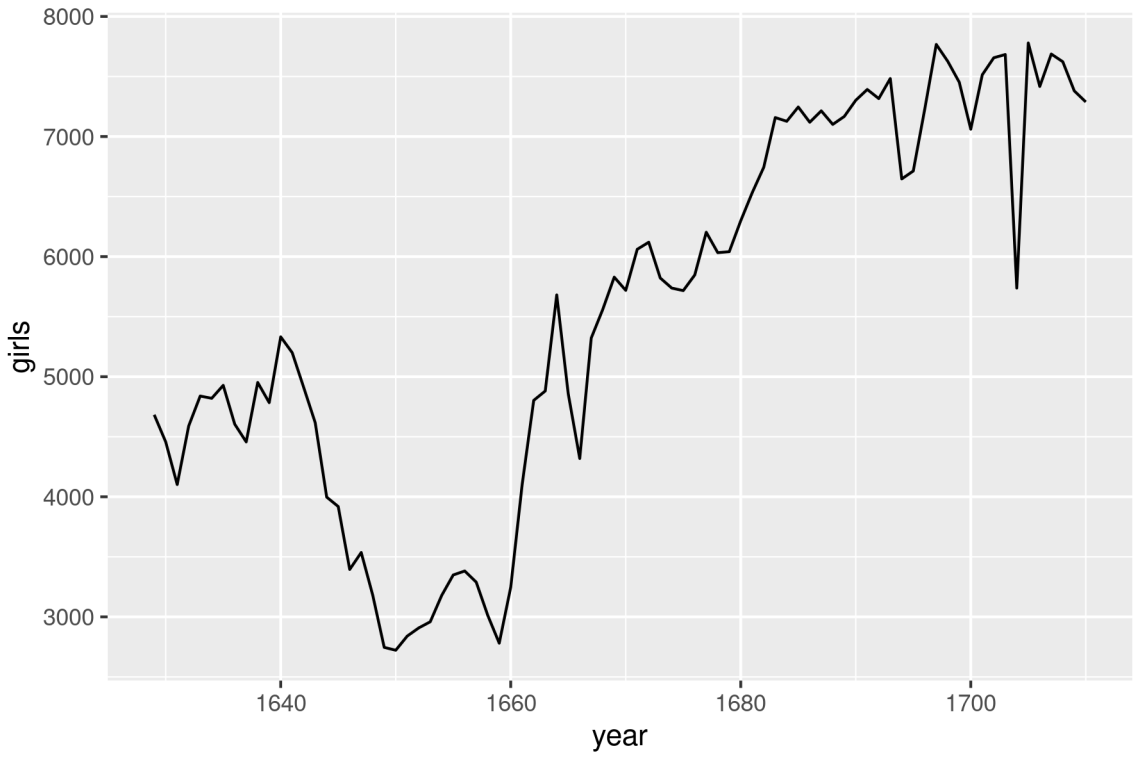
With `ggplot()`:

- The first argument is always the dataset.
- Next, you provide the variables from the dataset
  - to be assigned to `aes`thetic elements of the plot,
  - e.g. the x and the y axes.
- Finally, you use another "layer",
  - separated by a `+` to specify the `geom`etric object for the plot.
- Since we want to scatterplot,
  - we use `geom_point()`.

For instance, if you wanted to visualize the above plot

- using a line graph,
- you would
  - replace `geom_point()`
  - with `geom_line()`.

```
ggplot(data = arbuthnot, aes(x = year, y = girls)) +
  geom_line()
```



You might wonder how you are supposed to know

- the syntax for the `ggplot` function.
- Thankfully, R documents all of its functions extensively.
- To learn what a function does

  - and its arguments that are available to you,
  - just type in a question mark followed by the name of the function
  - that you're interested in.

Try the following in your console:

```
?ggplot
```

Notice that the help file

- replaces the plot in the lower right panel.
- You can toggle between plots and help files
  - using the tabs at the top of that panel.

1. Is there an apparent trend

- in the number of girls baptized over the years?
- How would you describe it?
  - (To ensure that your lab report is comprehensive,
  - be sure to include the code needed to make the plot
  - as well as your written interpretation.)

2.2.2.7.2 R as a big calculator

- Now, suppose we want to plot the total number of baptisms.

To compute this,

- we could use the fact that R is really just a big calculator.
- We can type in mathematical expressions like

```
5218 + 4683
```

```
## [1] 9901
```

to see the total number of baptisms in 1629.

We could repeat this once for each year,

- but there is a faster way.
- If we add the vector for baptisms for boys
  - to that of girls,
  - R will compute all sums simultaneously.

```
arbuthnot$boys + arbuthnot$girls
```

```
##  [1]  9901  9315  8524  9584  9997  9855 10034  9522  9160 10311 10150 10850
## [13] 10670 10370  9410  8104  7966  7163  7332  6544  5825  5612  6071  6128
## [25]  6155  6620  7004  7050  6685  6170  5990  6971  8855 10019 10292 11722
## [37]  9972  8997 10938 11633 12335 11997 12510 12563 11895 11851 11775 12399
## [49] 12626 12601 12288 12847 13355 13653 14735 14702 14730 14694 14951 14588
## [61] 14771 15211 15054 14918 15159 13632 13976 14861 15829 16052 15363 14639
## [73] 15616 15687 15448 11851 16145 15369 16066 15862 15220 14928
```

What you will see are 82 numbers

- (in that packed display, because we aren't looking at a data frame here),
- each one representing the sum we're after.
- Take a look at a few of them
  - and verify that they are right.

2.2.2.7.3 Adding a new variable to the data frame

- We'll be using this new vector to generate some plots,
  - so we'll want to save it as a permanent column
  - in our data frame.

```
arbuthnot <- arbuthnot %>%
  mutate(total = boys + girls)
```

The `%>%` operator is called the **pipe** operator.

- It takes the output of the previous expression
  - and pipes it into the first argument of the function in the following one.
- To continue our analogy with mathematical functions,
  - `x %>% f(y)` is equivalent to `f(x, y)`.

[This section of our script, from line 514 to 526, is html code]

A note on piping: Note that we can read these two lines of code as the following:

"Take the ***`arbuthnot`*** *dataset and* ***`pipe`*** *it into the* ***`mutate`*** *function. Mutate the* ***`arbuthnot`*** *data set by creating a new variable called* ***`total`*** *that is the sum of the variables called* ***`boys`*** *and* ***`girls`****. Then assign the resulting dataset to the object called* ***`arbuthnot`****, i.e. overwrite the old* ***`arbuthnot`*** *dataset with the new one containing the new variable."*

This is equivalent to going through each row and adding up the `boys` and `girls` counts for that year and recording that value in a new column called `total`.

Where is the new variable? When you make changes to variables in your dataset, click on the name of the dataset again to update it in the data viewer.

You'll see that there is now a new column called `total`

- that has been tacked onto the data frame.

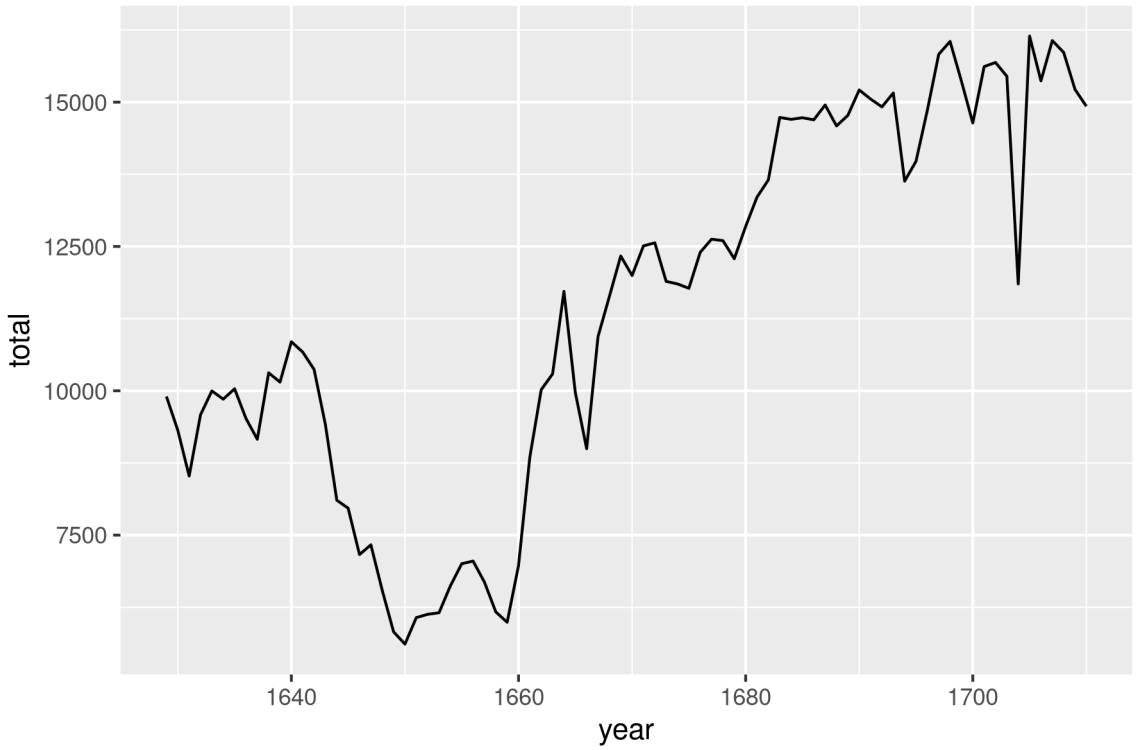
The special symbol `<-` performs an *assignment*,

- taking the output of one line of code
  - and saving it into an object in your environment.
- In this case, you already have an object called `arbuthnot`,
  - so this command updates that data set
  - with the new mutated column.

You can make a line plot

- of the total number of baptisms per year with the command

```
ggplot(data = arbuthnot, aes(x = year, y = total)) +
  geom_line()
```



Similarly to you we computed the total number of births,

- you can compute the ratio of
- the number of boys to the number of girls baptized in 1629 with

```
5218 / 4683
```

```
## [1] 1.114243
```

- or you can act on the complete columns with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(boy_to_girl_ratio = boys / girls)
```

You can also compute the proportion of newborns

- that are boys in 1629

```
5218 / (5218 + 4683)
```

```
## [1] 0.5270175
```

or you can compute this for all years simultaneously

- and append it to the dataset

```
arbuthnot <- arbuthnot %>%
  mutate(boy_ratio = boys / total)
```

Note that we are using the new `total` variable

- we created earlier in our calculations.

3. Now, generate a plot of the proportion of boys born over time. What do you see?

Tip: If you use the up and down arrow keys, you can scroll through your previous commands, your so-called command history. You can also access it by clicking on the history tab in the upper right panel. This will save you a lot of typing in the future.

Finally, in addition to simple mathematical operators

- like subtraction and division,
- you can ask R to make comparisons like
  - greater than, `>`,
  - less than, `<`,
  - and equality, `==`.

For example, we can ask

- if the number of births of boys
  - outnumber that of girls
- in each year with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(more_boys = boys > girls)
```

This command adds a new variable to the `arbuthnot` dataframe

- containing the values of either
  - `TRUE` if that year had more boys than girls,
  - or `FALSE` if that year did not
  - (the answer may surprise you).
- This variable contains a different kind of data
  - than we have encountered so far.

All other columns in the `arbuthnot` data frame

- have values that are numerical
  - (the year, the number of boys and girls).
- Here, we've asked R to create *logical* data,
  - data where the values are either `TRUE` or `FALSE`.

In general, data analysis will involve many different kinds of data types,

- and one reason for using R
- is that it is able to represent and compute with many of them.

2.2.2.8 More Practice

- In the previous few pages,
  - you recreated some of the displays
    - and preliminary analysis of Arbuthnot's baptism data.
  - Your assignment involves repeating these steps,
    - but for present day birth records in the United States.
  - The data are stored in a data frame called `present`.

To find the minimum and maximum values of columns,

- you can use the functions `min` and `max`
  - within a `summarize()` call,
  - which you will learn more about in the following lab.

Here's an example of how to find

- the minimum and maximum amount of boy births in a year:

```
arbuthnot %>%
  summarize(min = min(boys), max = max(boys))
```

```
## # A tibble: 1 x 2
##     min   max
##   <int> <int>
## 1  2890  8426
```

- What years are included in this data set?
  - What are the dimensions of the data frame?
  - What are the variable (column) names?
- How do these counts compare to Arbuthnot's?
  - Are they of a similar magnitude?
- Make a plot that displays the proportion of boys born over time.
  - What do you see?
    - Does Arbuthnot's observation
    - about boys being born in greater proportion than girls hold up in the U.S.?
  - Include the plot in your response.
    - *Hint:* You should be able to reuse your code from Exercise 3 above,
    - just replace the dataframe name.
- In what year did we see the most total number of births in the U.S.?
  - *Hint:* First calculate the totals and save it as a new variable.
  - Then, sort your dataset in descending order based on the total column. You can do this interactively in the data viewer
    - by clicking on the arrows next to the variable names.
  - To include the sorted result in your report
  - you will need to use two new functions:
    - `arrange` (for sorting).
    - We can arrange the data in a descending order with
    - another function: `desc` (for descending order).

Write your code below.

```
# present %>% arrange(desc(total))
```

These data come from reports by the Centers for Disease Control.

- You can learn more about them
  - by bringing up the help file using the command `?present`.

2.2.2.9 Resources for learning R and working in RStudio

- That was a short introduction to R and RStudio,
  - but we will provide you with more functions
  - and a more complete sense of the language as the course progresses.

In this course we will be using the suite of R packages from the **tidyverse**.

The book R For Data Science by Grolemund and Wickham

- is a fantastic resource for data analysis in R with the tidyverse.

If you are googling for R code,

- make sure to also include these package names in your search query.
- For example, instead of googling "scatterplot in R",
  - google "scatterplot in R with the tidyverse".

These cheatsheets may come in handy throughout the semester:

- RMarkdown cheatsheet
- Data transformation cheatsheet
- Data visualization cheatsheet

Note that some of the code on these cheatsheets

- may be too advanced for this course.
- However the majority of it will become useful throughout the semester.
