R and Version Control

2020 DSS Bootcamp

Shawn Santo

08-03-20

Supplementary materials

Companion videos

1. Version control
2. R / RStudio and reproducible research

Preliminaries

GitHub account

To get everyone on the same page:

- If you don't have one, sign up for a GitHub account (it takes 1 min.)
  - Go to https://github.com/join
  - Enter your information
  - Pick the `Unlimited public repositories for free.` plan

A few suggestions on picking a GitHub username:

- Incorporate your actual name! People like to know who they're dealing with. It makes your username easier for people to guess or remember.
- Pick a username you will be comfortable revealing to your future boss.
- Shorter is better than longer.
- Be as unique as possible in as few characters as possible. In some settings GitHub auto-completes or suggests usernames.
- Make it timeless. Don't highlight your current university, employer, or place of residence.
- Avoid words laden with special meaning in programming.

Accessing RStudio Pro

To get started as quickly as possible, the preferred method is to use DSS RStudio servers. To access RStudio Pro:

1. If off campus, use the VPN to create a secure connection from your computer to Duke. If you are on campus, be sure you are connected to the Dukeblue network.

2. Navigate to one of

    - http://pawn.stat.duke.edu:8787
    - http://rook.stat.duke.edu:8787 (MS students)
    - http://knight.stat.duke.edu:8787 (PhD students)

3. Log-in with your Duke NetID and password.

If you are having trouble accessing RStudio Pro see the next slide.

DSS RStudio Pro alternatives

If you cannot access RStudio Pro via DSS servers:

- Use R and RStudio locally on your computer
    - Download R on your computer here
    - Download RStudio here
- Use a docker container from Duke OIT
    1. Go to https://vm-manage.oit.duke.edu/containers
    2. Log in with your Duke NetID and password
    3. Find `RStudio - statistics application with Rmarkdown and knitr support`
    4. Click the link to create your environment

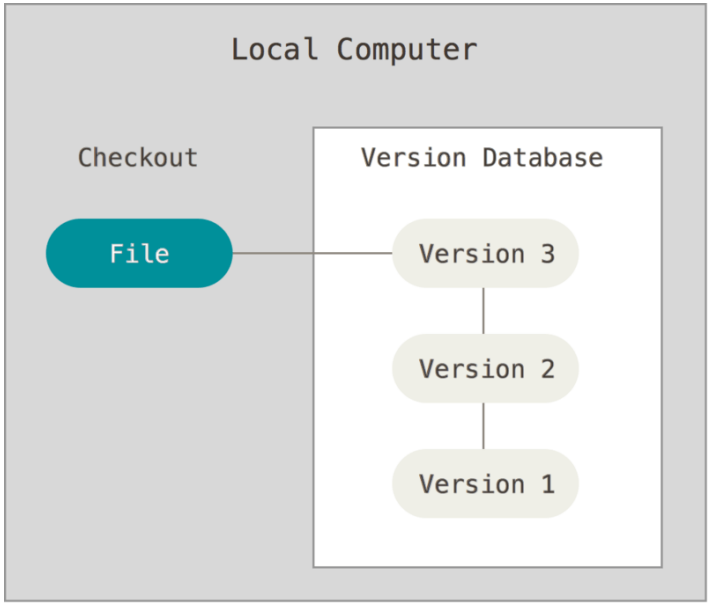
Bootcamp materials

- Everything from this computing bootcamp is available on my GitHub account at https://github.com/shawnsanto/computing_bootcamp_2020.
- You'll learn how to get this on your computer shortly.

git and GitHub

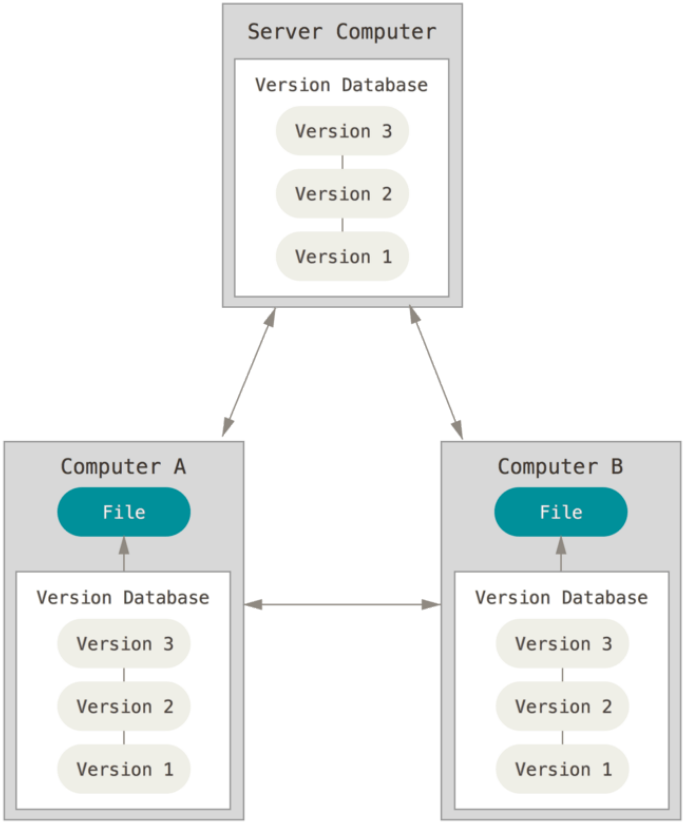
Why version control?

- Simple formal system for tracking all changes to a project
- Time machine for your projects
  - Track blame and/or praise
  - Remove the fear of breaking things
- Learning curve is steep, but when you need it you *REALLY* need it



Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups
- Popular and Successful
  - Active development
  - Shiny new tools and ecosystems
  - Fast
- Tracks any type of file
- Branching
  - Smarter merges



Verifying git exists

git is already set-up on the DSS servers. In the terminal tab you can verify this by

```
[sms185@numeric1 ~]$ git --version
git version 2.20.1
```

```
[sms185@numeric1 ~]$ which git
/usr/bin/git
```

To install git and connect git, GitHub and RStudio on your own computer follow the directions in Happy Git and GitHub for the useR.

git and GitHub live demo

Getting started today

In order to get started, you need to obtain today's files from GitHub. The steps below will give you access.

1. Log in to GitHub
2. Navigate to https://github.com/shawnsanto/computing_bootcamp_2020
3. Fork the repository
4. Copy the link under `Clone or Download` to clone with HTTPS
5. In RStudio go to `File > New Project > Version Control > Git`
6. Paste the URL, that you copied in step 4, in the box under `Repository URL:`

You now should have all the files in the repository in a directory on the server or your own computer.

Configure git

The following will tell git who you are, what editor you want to use, and to store your username and password.

```
$ git config --global user.name "Shawn Santo"
$ git config --global user.email "shawn.santo@duke.edu"
$ git config --global push.default simple
$ git config --global core.editor [editor-of-choice]
$ git config --global credential.helper 'cache --timeout=600000'
```

On DSS my editor of choice is vim; on my own computer my editor of choice is Sublime Text 3.

You will need to do this configuration on each machine in which you choose to use git.

For windows users, the last line should be

```
$ git config --global credential.helper wincred
```

Configure git verification

To verify you configured git correctly, run

```
[sms185@numeric1 ~]$ git config --global -l
user.name=Shawn Santo
user.email=shawn.santo@duke.edu
core.editor=vim
push.default=simple
credential.helper=cache --timeout=600000
```

You should see output similar to above.

Using command `git config --global -l` on Windows:

```
user.name=Shawn Santo
user.email=shawn.santo@duke.edu
core.editor='c:/program files/sublime text 3/sublime_text.exe' -w
credential.helper=wincred
push.default=simple
```

Using command `git config --global -l` on Mac:

```
credential.helper=osxkeychain
user.name=Shawn Santo
user.email=shawn.santo@duke.edu
core.editor=/Applications/Sublime\ Text.app/Contents/SharedSupport/bin/subl -n -w
push.default=simple
credential.helper=cache --timeout=600000
```

Version control best practices

- Commit early, often, and with complete code.
- Write clear and concise commit summary messages.
- Test code before you commit.
- Use branches.
- Communicate with your team.

git and GitHub resources

- git's Pro Git book, Chapters Getting Started and Git Basics will be most useful if you are new to git and GitHub
- Git cheatsheet by Atlassian
- GitHub's interactive tutorial
- Free online course from Udacity
- Happy Git with R by Jenny Bryan

Responsible research and reproducibility

Seizure study retracted after authors realize data got "terribly mixed"

- From the authors of **Low Dose Lidocaine for Refractory Seizures in Preterm Neonates**:
- *"The article has been retracted at the request of the authors. After carefully re-examining the data presented in the article, they identified that data of two different hospitals got terribly mixed. The published results cannot be reproduced in accordance with scientific and clinical correctness."*

Source: http://retractionwatch.com/2013/02/01/seizure-study-retracted-after-authors-realize-data-got-terribly-mixed/

Bad spreadsheet merge kills depression paper, quick fix resurrects it

- The authors informed the journal that the merge of lab results and other survey data used in the paper resulted in an error regarding the identification codes. Results of the analyses were based on the data set in which this error occurred. Further analyses established the results reported in this manuscript and interpretation of the data are not correct.
- **Original conclusion:** Lower levels of CSF IL-6 were associated with current depression and with future depression [...].
- **Revised conclusion:** Higher levels of CSF IL-6 and IL-8 were associated with current depression [...].

Source: http://retractionwatch.com/2014/07/01/bad-spreadsheet-merge-kills-depression-paper-quick-fix-resurrects-it/

Study of social media retracted when authors can't provide data

- "*A business journal has retracted a 2016 paper about how social media can encourage young consumers to become devoted to particular brands, after discovering flaws in the data and findings.*"
- Reasons for retraction:
  - Error in data
  - Error in results and/or conclusions
  - Results not reproducible

Source: http://retractionwatch.com/2017/07/31/study-social-media-retracted-authors-cant-provide-data/

Heart pulls sodium meta-analysis over duplicated, and now missing, data

- "*The journal Heart has retracted a 2012 meta-analysis after learning that two of the six studies included in the review contained duplicated data. Those studies, it so happens, were conducted by one of the co-authors.*"
- From the retraction notice, "*The Committee considered that without sight of the raw data on which the two papers containing the duplicate data were based, their reliability could not be substantiated. Following inquiries, it turns out that the raw data are no longer available having been lost as a result of computer failure.*"
- Reasons for retraction:
  - Duplication of data
  - Results not reproducible

Source: http://retractionwatch.com/2013/05/02/heart-pulls-sodium-meta-analysis-over-duplicated-and-now-missing-data/

Reproducibility: why should we care?

1. Convince researchers to adopt a reproducible research workflow.
2. Train new researchers who don't have any other workflow.

Donald Knuth "Literate Programming" (1983)

"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do."

"The practitioner of literate programming [...] strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other."

- These ideas have been around for years!
- Tools for putting them to practice have also been around.
- They have never been as accessible as the current tools.

Reproducibility checklist

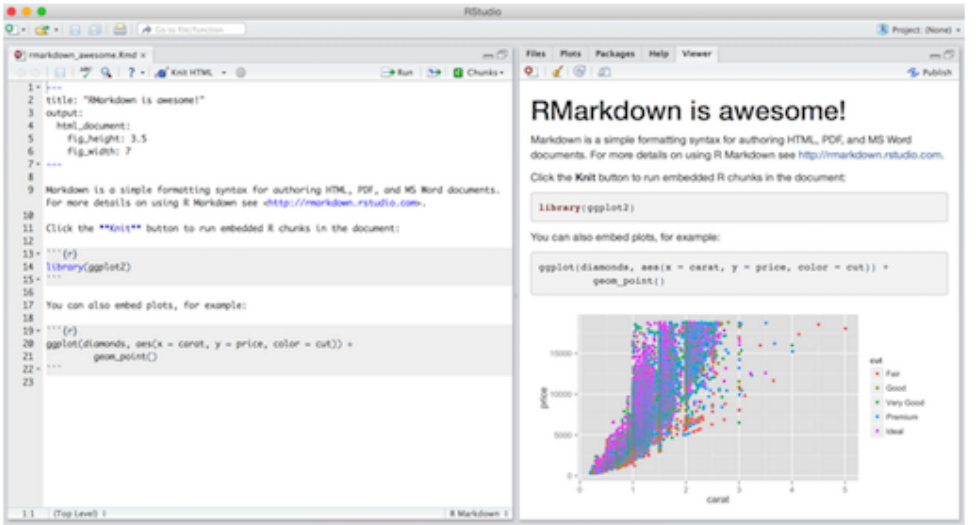
- Are the tables and figures reproducible from the code and data?
- Does the code actually do what you think it does?
- In addition to what was done, is it clear *why* it was done? (e.g., how were parameter settings chosen?)
- Can the code be used for other data, especially future updates to the current data?
- Can you extend the code to do other things?

Ambitious goal

We need an environment where

- data, analysis, and results are tightly connected, or better yet, inseparable,
- reproducibility is built in,
    - the original data remains untouched
    - all data manipulations and analyses are inherently documented
- documentation is human readable and syntax is minimal.

Toolkit


RStudio
RMarkdown is awesome!
Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.
Click the Knit button to run embedded R chunks in the document:
You can also embed plots, for example:
carat
price

Reproducible data analysis

- Scriptability $\rightarrow$ R
- Literate programming $\rightarrow$ R Markdown
- Version control $\rightarrow$ git / GitHub

Could these tools have prevented some of the aforementioned retractions?

What is markdown?

- Markdown is a lightweight markup language for creating HTML (or XHTML) documents.
- Markup languages are designed to produce documents from human readable text (and annotations).
- Some of you may be familiar with LaTeX. This is another (less human friendly) markup language for creating pdf documents.
- Why markdown is great:
    - Easy to learn and use.
    - Focus on **content**, rather than **coding** and debugging **errors**.
    - Once you have the basics down, you can get fancy and add HTML, JavaScript, and CSS.

R supports R Markdown - an authoring framework for data science and statistical analysis.

What is R Markdown?

What is R Markdown? from RStudio, Inc. on Vimeo.

R Markdown

Something simple

Simple

Shawn Santo

8/12/2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

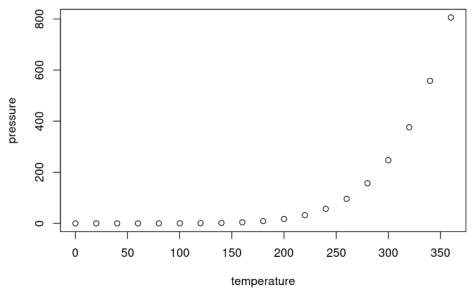
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

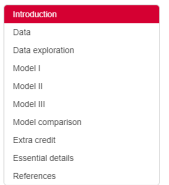
```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

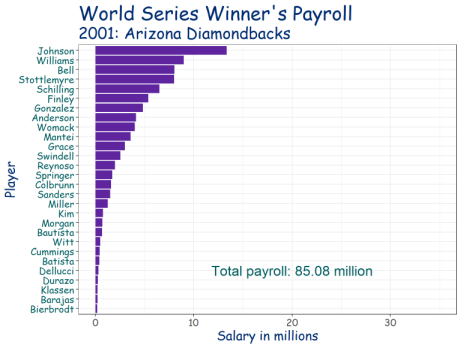
You can also embed plots, for example:



Something fancy



Exam 2



Introduction

You will work with yearly baseball statistics for each team from the years 2000 to 2015. You do not need any deep insight into the sport of baseball to successfully complete this exam. However, if you think you do not understand a specific variable and how it relates to baseball then ask questions.

A baseball team beats its opponent when it outscores its opponent. This happens when a team scores more runs than it allows in a game. The win percentage (percentage of games played that resulted in a win) of a team is related to the number of runs a team scores versus the number of runs a team allows over the course of a full season (usually 162 games). Runs scored and runs allowed are a function of hits and walks, and hits allowed and walks allowed, respectively. You will explore three models that investigate the relationship between win percentage, and runs scored and runs allowed. This is the first step if you ultimately want to quantify players' value to a team.

Data

Below is a preview of the data set. Consult the data dictionary for further details on the variables. Definitions of variables are available if you load package `Lahman` and type `?Teams` in your Console. The file `mlb.csv` is a subset of data frame `Teams`. You should not work with the `Teams` data frame.

To get started, load packages `tidyverse` and `broom`. Next, read in `mlb.csv` with function `read_csv()` and save it as an object named `mlb`.

R Markdown resources

- In RStudio, go to `Help > Cheatsheets` and select
  - R Markdown Cheat Sheet
  - R Markdown Reference Guide
- Check out the official R Markdown book: R Markdown: The Definitive Guide by Yihui Xie, J. J. Allaire, and Garrett Grolemund
- Check out bookdown: Authoring Books and Technical Documents with R Markdown by Yihui Xie.
- Take a look at RPubs web published R Markdown documents.

R / RStudio

R / RStudio

- Rather than use the R graphical user interface we will use RStudio.
- RStudio is a free and open-source integrated development environment (IDE) for R.

R packages

- Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data.
- In the following exercises we'll use the `tidyverse` package.
    - The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.
    - The core tidyverse packages consists of `ggplot2`, `tibble`, `tidyr`, `readr`, `purrr`, `dplyr`, `stringr`, and `forcats` packages.
- This package is already installed for you on the DSS servers. If needed, you can install it by running the following in the `Console`:

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

You only need to install a package once, but you must load it with function `library()` each R session.

A note on workspaces

- Your R Markdown document and your Console do not share workspaces.
- This is good for reproducibility, but can sometimes result in frustrating errors.
- This also means any packages needed for your analysis need to be loaded in your R Markdown document as well.

Gapminder data analysis

To get started,

- open `gapminder_analysis.Rmd`,
- run the code chunk to load `tidyverse` (install it, if needed)

```
library(tidyverse)
```

- run the code chunk to read in the data and store it as an object named `gap`.

```
gap <- read_csv("https://bit.ly/gap_data")
```

Function `read_csv()` is part of package `readr`, which is automatically loaded from package `tidyverse`. Function `read_csv()` not only reads in the data, but it ensures object `gap` is a neatly formatted data frame - known as a tibble.

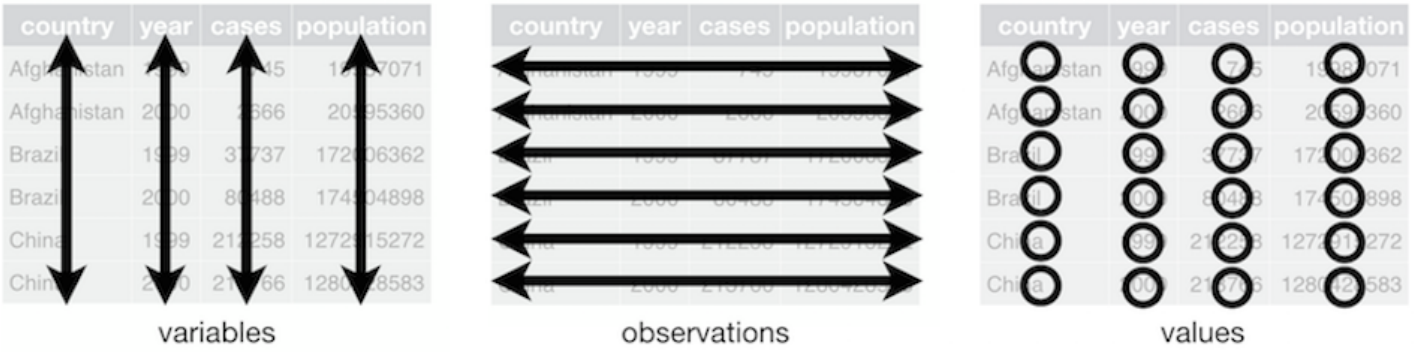
Examine `gap`

Let's look at the first 10 rows of `gap`.

```
gap %>%
  slice(1:10)
```

```
#> # A tibble: 10 x 6
#>    country             continent  year lifeExp       pop gdpPercap
#>    <chr>               <chr>     <dbl>   <dbl>     <dbl>     <dbl>
#>  1 United Kingdom      Europe     1967    71.4  54959000    14143.
#>  2 Eritrea             Africa     2007    58.0   4906585      641.
#>  3 Trinidad and Tobago Americas   1997    69.5   1138101     8793.
#>  4 Taiwan              Asia       2002    77.0  22454239    23235.
#>  5 Kuwait              Asia       2007    77.6   2505559    47307.
#>  6 Brazil              Americas   1987    65.2 142938076     7807.
#>  7 Cameroon            Africa     1957    40.4   5359923     1313.
#>  8 Hungary             Europe     1982    69.4  10705535    12546.
#>  9 Iceland             Europe     1952    72.5    147962     7268.
#> 10 Turkey              Europe     1952    43.6  22235677     1969.
```

Tidy data



Source: Grolemund and Wickham, R for Data Science, http://r4ds.had.co.nz/

Exercise 1

Take a glimpse at the data set with function `glimpse()`.

```
glimpse(gap)
```

```
#> Rows: 1,363
#> Columns: 6
#> $ country   <chr> "United Kingdom", "Eritrea", "Trinidad and Tobago", "Taiwan…
#> $ continent <chr> "Europe", "Africa", "Americas", "Asia", "Asia", "Americas",…
#> $ year      <dbl> 1967, 2007, 1997, 2002, 2007, 1987, 1957, 1982, 1952, 1952,…
#> $ lifeExp   <dbl> 71.360, 58.040, 69.465, 76.990, 77.588, 65.205, 40.428, 69.…
#> $ pop       <dbl> 54959000, 4906585, 1138101, 22454239, 2505559, 142938076, 5…
#> $ gdpPercap <dbl> 14142.8509, 641.3695, 8792.5731, 23235.4233, 47306.9898, 78…
```

How many variables and observations are in `gap`? What are the variable types for the variables in `gap`?

Variable	Type
`country`	character
`continent`	character
`year`	double
`lifeExp`	double
`pop`	double
`gdpPercap`	double

Exercise 2

Part 1

Create a scatter plot of life expectancy versus year for the United States.

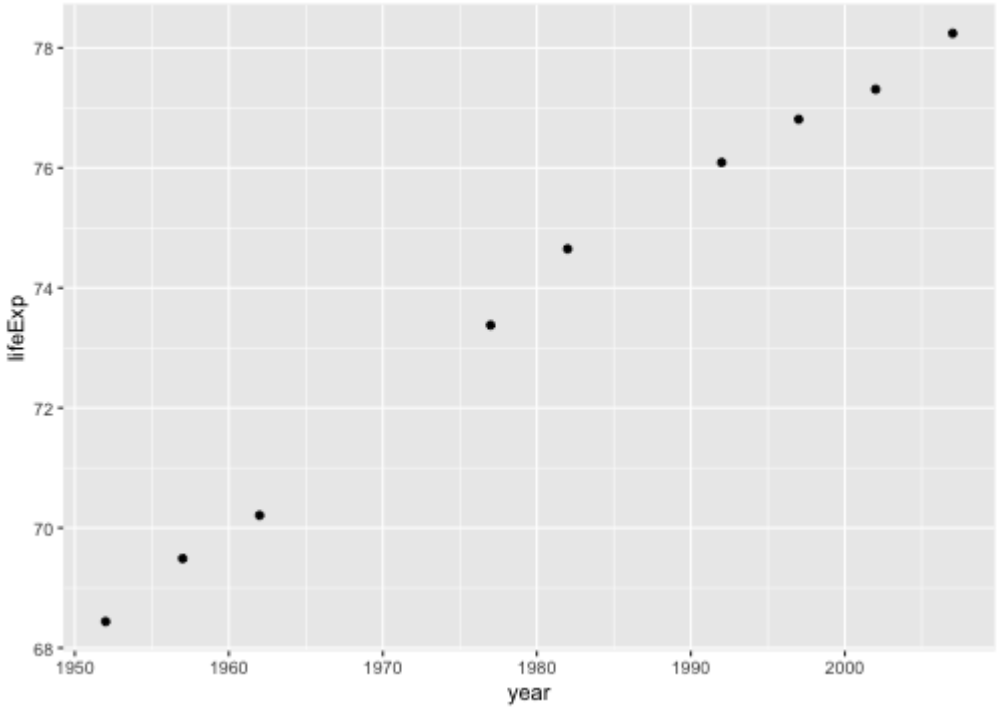
```
gap_usa <- gap %>%
  filter(country == "United States")

ggplot(data = gap_usa, mapping = aes(x = year, y = lifeExp)) +
  geom_point()
```

Exercise 2

Part 1

Create a scatter plot of life expectancy versus year for the United States.

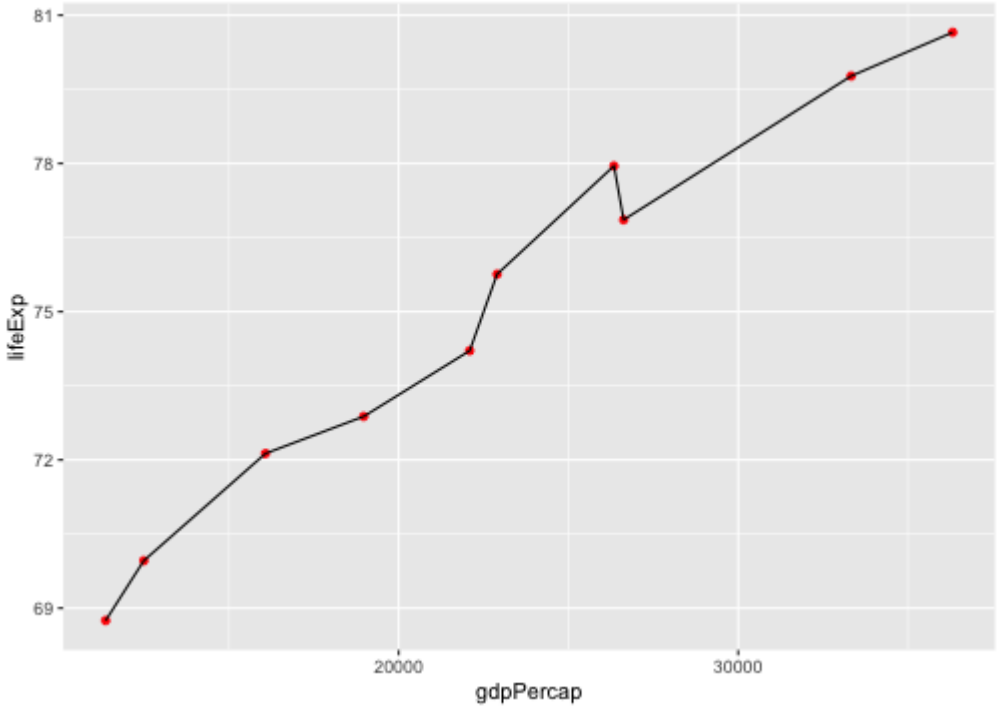


Think about the connection between the code and the plot.

Exercise 2

Part 2

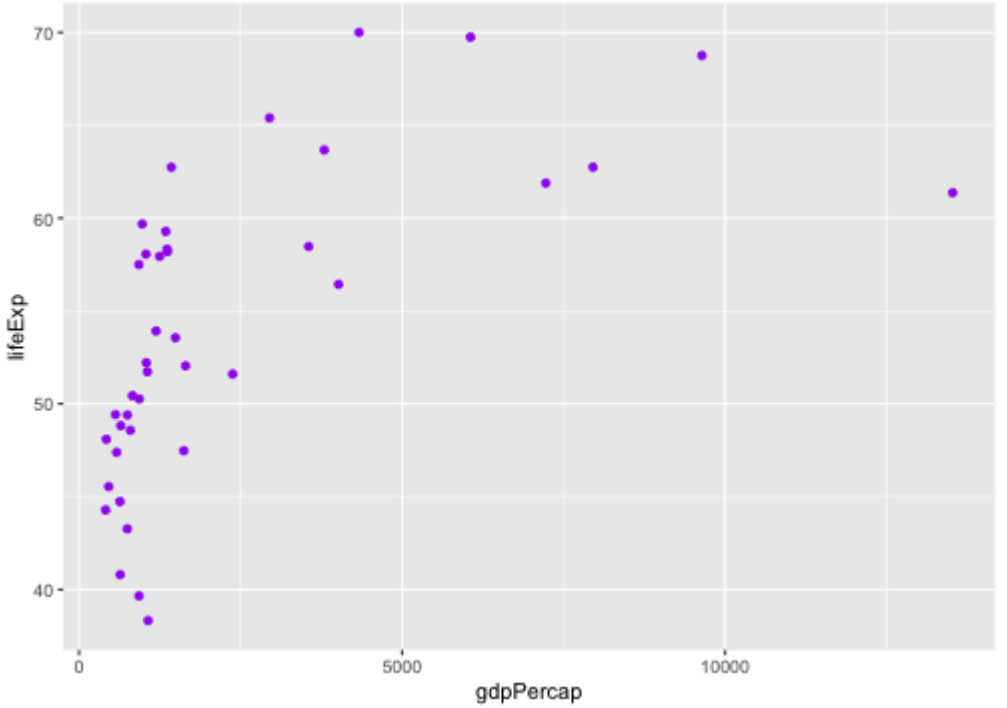
Create a plot of life expectancy verses GDP per capita for Canada. Set the color of the points to be red and connect them with a solid black line.



Exercise 2

Part 3

Create a scatter plot of life expectancy versus GDP per capita based on all countries in Africa in 1992. Set the color to be purple.



Exercise 3

Part 1

Create a plot of life expectancy versus GDP per capita based on all the countries in Africa and Europe in 1992. Set the point colors to code for the two continents.

Hints:

1. Filter `gap` so it only contains observations from Africa or Europe. Do this with `continent %in% c("Africa", "Europe")`. Also, filter so that the observations are only from 1992. Separate multiple conditions with commas.

2. Add another aesthetic to the plot. Instead of just `x` and `y`, include `color = continent` inside function `aes()`. The color of the points will be set for each level of the variable `continent`.

Exercise 3

Part 2

Create comparison boxplots by continent of GDP per capita for the year 2007. Fill the boxplots with color `darkgreen`.

Hints:

1. Filter `gap` so it only contains observations from 2007.
2. Inside `aes()` set `x = continent, y = gdpPercap`.
3. Use `geom_boxplot()` and set the fill to `darkgreen`.

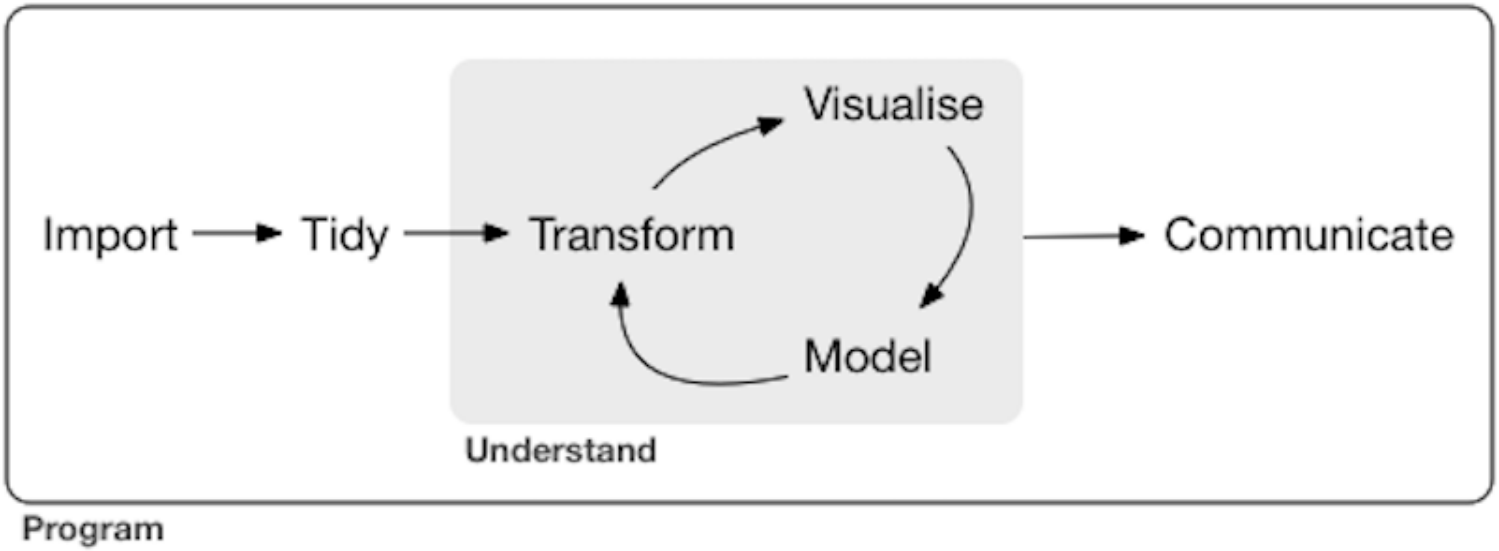
Reproducible?

Is your work reproducible? Are you able to easily redo your analysis with the newly updated data?

```
gap_upd <- read_csv("https://bit.ly/gap_data_upd")
```

Did you remember to commit and push your work to GitHub?

Workflow



Source: Grolemund and Wickham, R for Data Science, http://r4ds.had.co.nz/

Suggestions and tips

Organizing a more complex analysis

- Use folders: raw_data, processed_data, code, figures, etc.
- Number your script files:
  - `00_load_data.R`
  - `01_data_cleanup.R`
  - `02_eda.R`
  - ...
- Use informative names that indicate versioning
  - use dates
  - avoid things like `project_fin`, `project_finalfinal`, etc.

R Markdown suggestions

- Always name your code chunks
- Familiarize yourself with chunk options
    - Use global chunk options to reduce duplication
- Load packages at the start of a document, generally the chunk after your set-up chunk
- Familiarize yourself with various outputs: Make slides with `output:` `ioslides_presentation` or `xaringan`, make websites with `blogdown`, author a book with `bookdown`, etc.

These slides were made with R Markdown and `xaringan`.

Projects

Always use them. It will make your life easier.

Configuration

- Panes
- View in pane for Rmd
- Themes
- Fonts

Related resources

R programming resources

- Style
  - Tidyverse style guide
  - Google's R style guide
- Beginner
  - swirl: swirl teaches you R programming and data science interactively, at your own pace, and right in the R console
  - R manuals
  - R for Data Science by Hadley Wickham and Garret Grolemund
  - R Cookbook by Paul Teetor
- Advanced
  - Advanced R by Hadley Wickham
  - R Packages by Hadley Wickham
- Miscellaneous
  - All available CRAN packages, sorted by name

More R / RStudio resources

- Some useful resources from RStudio: https://www.rstudio.com/resources/cheatsheets/
    - RStudio IDE Cheat Sheet
    - R Markdown Cheat Sheet
    - R Markdown Reference Guide
    - Data Import Cheat Sheet
    - Data Transformation Cheat Sheet
    - Data Visualization Cheat Sheet

Some of the above cheat sheets are available in RStudio: `Help > Cheatsheets`

References

1. DukeStatSci/computing-bootcamp-2018. (2020). Retrieved from https://github.com/DukeStatSci/computing-bootcamp-2018

2. Git - Book. (2020). Retrieved from https://git-scm.com/book/en/v2

3. Grolemund, G., & Wickham, H. (2020). R for Data Science. Retrieved from https://r4ds.had.co.nz/

4. Retraction Watch. (2020). Retrieved from https://retractionwatch.com/

5. R Markdown. (2020). Retrieved from https://rmarkdown.rstudio.com/index.html