

# Reproducibility in government

## The RAP initiative

Sonia Mazzi - ONS

10 October, 2019



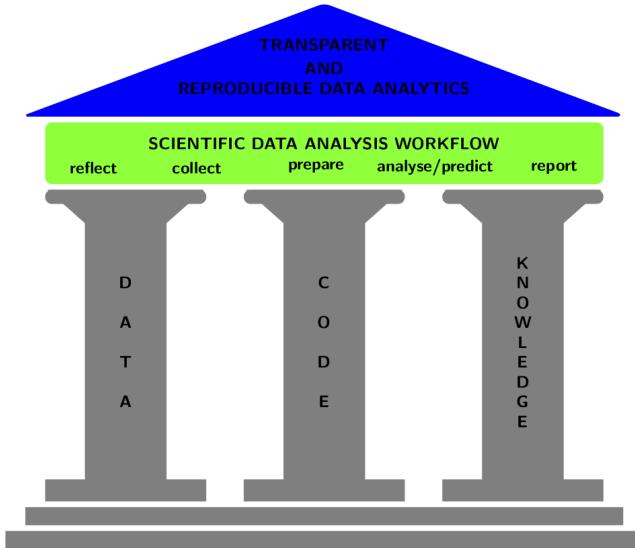
## What is reproducibility?

- ▶ **Reproducibility** is a feature of a data-driven project by which if it is repeated using the same data and methods identical/consistent results are obtained.
- ▶ Ideally a reproducible project should also be **replicable**: if the study is repeated with new data and alternative computations consistent results are obtained.

# Point and click analytics do not contribute to reproducibility



# How to achieve reproducibility



# Code is a key tool to reproduce analysis

```
16  $("#limit_val");
17  var b = k();
18  h();
19  var c = l(), a = " ", d = parseInt($("#limit_val").a()), f = parseInt($("#limit_val").f());
20  function("LIMIT_total:" + d);
21  function("rand:" + f);
22  d < f && (f = d, function("check rand\u00f3\u00f3rand: " + f + "tops: " + 0));
23  var n = [], d = d - f, e;
24  if (0 < c.length) {
25    for (var g = 0; g < c.length; g++) {
26      e = m(b, c[g]), -1 < e && b.splice(e, 1);
27    }
28    for (g = 0; g < c.length; g++) {
29      b.unshift({use_wystepuje:"parameter", word:c[g]});
```

# Reproducible Analytical Pipelines (RAP)

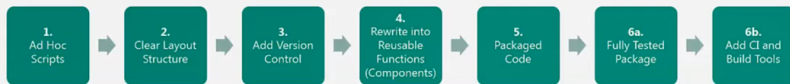
- ▶ RAP focuses on “code” and the deployment of software engineering tools for report production
- ▶ The process for official statistics production in Government:



# RAP in action

- ▶ In a RAP, a report is produced entirely by writing code. A report is the output of bespoke software.
- ▶ Tools from software engineering will create reproducibility features.

# Levels of RAP (by DAP)



Exploration



Construction

Level	Description	Output Produced using R/Python	Coding Standards	Version Control	Peer Review	Documentation of Functions	Automated Data QA	Dependency Packages Managed	Unit Testing of Functions	Unit Testing Guaranteed
1	Ad hoc Scripts	✓	✗	✗	✗	✗	✗	✗	✗	✗
2	Clear Layout Structure	✓	✓	✗	✗	✗	✗	✗	✗	✗
3	Add Version Control	✓	✓	✓	✓	✗	✗	✗	✗	✗
4	Rewrite into Reusable Functions (Components)	✓	✓	✓	✓	✓	✓	✗	✗	✗
5	Packaged Code	✓	✓	✓	✓	✓	✓	✓	✗	✗
6a	Fully Tested Package	✓	✓	✓	✓	✓	✓	✓	✓	✗
6b	CI and Build	✓	✓	✓	✓	✓	✓	✓	✓ (+75%)	✓



# Using version control for reproducibility



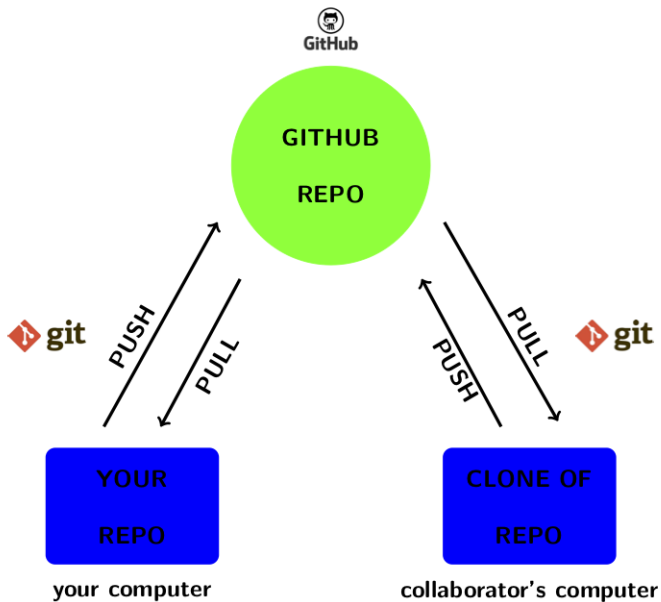
- ▶ One important tool for code quality and code sharing is a version control system (VCS).
- ▶ A version control system is a tool that manages changes made to the files and directories in a project.
- ▶ Git, together with GitHub, is one such VCS tool.
- ▶ Git was built to manage development of the Linux kernel (2005), which is probably very different from what the present day users do.

# How does VC work?

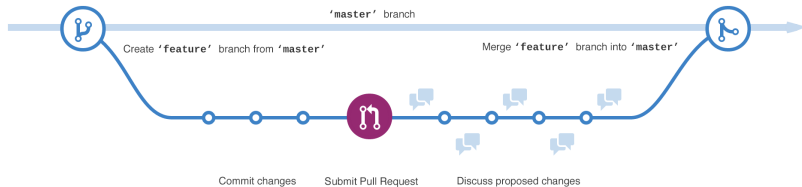
## LOCALLY IN YOUR COMPUTER



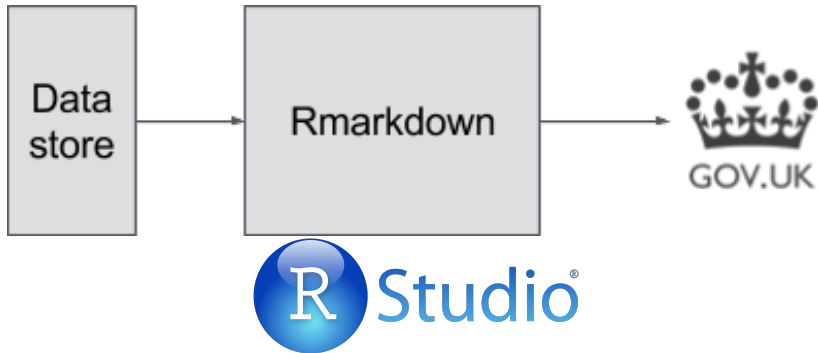
# SHARING YOUR REPO



# WORKING SIMULTANEOUSLY



## Worked example using R and RStudio



Produce reports on HIV prevalence by given continent and year

Files in [https://github.com/sonjam111/EFGS\\_2019](https://github.com/sonjam111/EFGS_2019)

# Produce reports on HIV prevalence by given continent and year

- ▶ `hiv_report.Rmd` with text and code that generates a report for a continent and a year. Key is YAML header

```
---
params:
  current_continent: Americas
  year: 2011
  data_file: hiv_wb_new.xlsx
title: "**`r paste('Report on HIV prevalence in',
                  params$current_continent, ', ', params$year)`**"
author: Sonia Mazzi - Data Science Campus
description: "An example of a reproducible report, from data importing,
             tidying, and cleaning, analysing to publishing with rmarkdown"
date: "`r format(Sys.Date(), '%d %B %Y')`"
output:
  html_document:
    theme: yeti
    highlight: haddock
    number_sections: TRUE
---
```

- ▶ When happy that the code in `hiv_report.Rmd` work, write a function to run it overriding the parameters in the YAML.

```
year <- 2010
render_report <- function(continent) {
  rmarkdown::render("hiv_report.Rmd",
    output_file = paste0(
      continent, "_", year, "_report_", ".html"),
  params = list(current_continent = continent),
  output_options = list(
    self_contained = FALSE, lib_dir = "libs"))
}
```



- Now write code that generates all 4 reports in one execution

```
continents <- c("Asia", "Europe", "Africa", "Americas")  
year <- 2011  
purrr::walk(continents, render_report)
```

