

Eltecon Data Science Course by Emarsys

Introduction

Gábor Kocsis

September 11, 2019

R and RStudio

R and RStudio

- R is a programming language
- RStudio is an IDE for using R
 - IDE: Integrated Development Environment
 - Code editor
 - Built-in automation tools
- R may be used without RStudio, but RStudio cannot be used without R

RStudio projects

- RStudio projects help to divide and organize your work
- Each project has its own working directory and workspace
- You can work in multiple projects at the same time

Relative paths

- Check your working directory with `getwd()`, it might return `"/Users/your_name"`
- If you need, set your working directory with `setwd()`
 - e.g. `setwd("~/eltecon-ds/first_class")`
- When you refer to a file in the working directory, don't need to specify the path
 - `foo <- fread("sales.csv")`
- but, when you refer to a file in another directory, you need to define the path relative to the working directory
 - `foo <- fread("data/sales.csv")`
 - `foo <- fread("../sales.csv")`

Annoying situations

Code calls files that are not available anymore

```
Error in fread("data/raw_data.csv"): File  
'data/raw_data.csv' does not exist or is non-readable.
```

Infinite number of versions saved



Don't understand what is happening in the code you wrote last year

```
dt <- fread("data/data.csv")
boo <- foo(dt)
out1 <- doCalculations(boo)
out2 <- doMoreCalculations(boo)
plotFigures(boo)
plotMoreDetailedFigures(boo)
```

Have to change one of your assumptions in your model
that was copy pasted throughout the whole project



Working on the same project with others at the same time



Code and data - Guidelines

A good directory structure

- project_directory
 - analysis.R
 - functions.R
 - data
 - figures
 - README.md
 - .gitignore

analysis.R

- Save your analysis script here
- Write comments to separate sections, but do not overuse them
- Use intention revealing names
 - e.g. instead of `ps` use `product_supply`
- Make meaningful distinctions
 - e.g. don't use `cust` and `customers` close to each other
- Use pronounceable names
 - e.g. instead of `purchymd` use `purchase_date`
- Use uppercase letters for constants
 - e.g. `PROPORTION_OF_INCOME_SAVED`, `LEVEL_OF.Utility`

functions.R

- Save your functions here
- Functions in long scopes should have short evocative names
 - e.g. `replaceNAWithZero()`
- Functions in small scopes should have long and precise names
 - e.g. `plotPriceElasticityOfGasolineDemand()`

Clean coding



*“You shouldn't have to read the body of a function to know what it does.
It's name should tell you.”* - Robert Cecil Martin aka Uncle Bob

data folder

- File names should declare their function
- Always keep a version of the original raw data
- Optionally, use sub-folders in the data folder like
 - raw_data
 - processed_data

Version control with Git

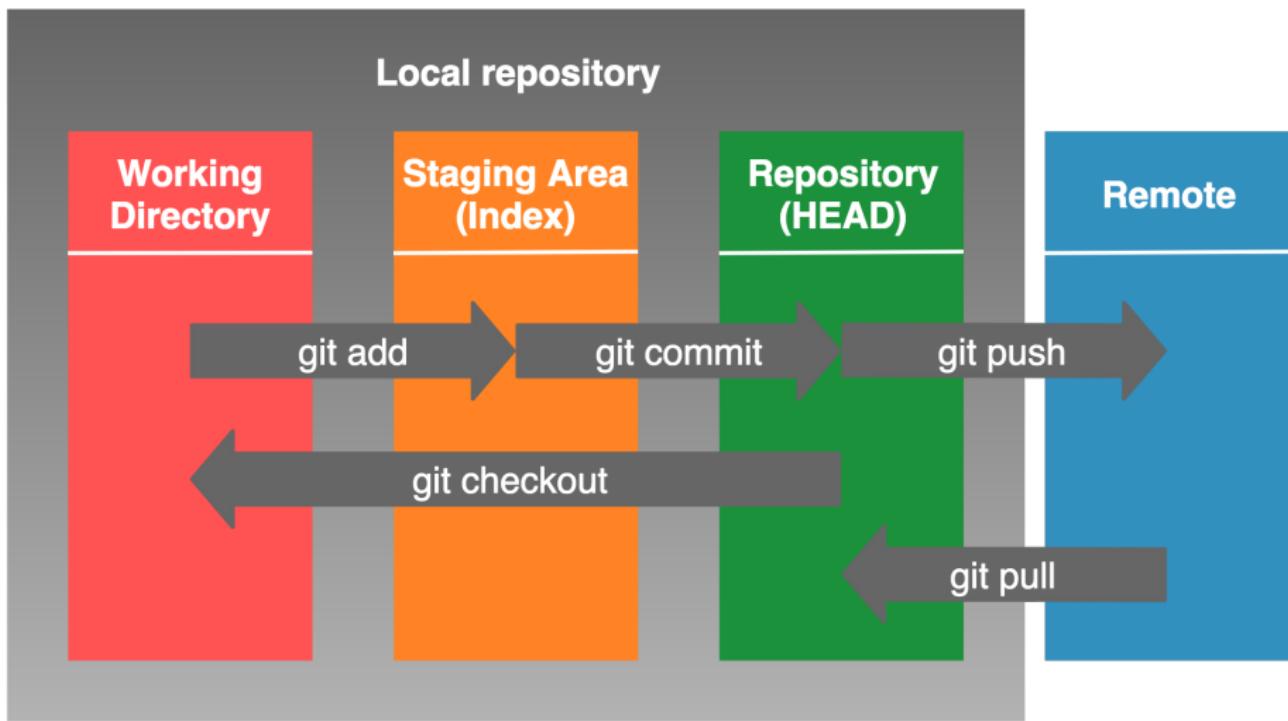
What does git stand for?

- The word “git” is slang for somebody who’s rotten or stupid
- *“I’m an egotistical bastard, so I name all my projects after myself. First Linux, now git.”* - by Linus Torvalds (he wrote the original git version)
- The git README file on github has a longer description

Git is a version control system

- Store code and data under version control
- Benefits
 - Track different versions of the same file
 - Keep only things you currently need
 - Collaborate easily
- Git in RStudio
- Git in command line

Git stages



Git status in Rstudio

The screenshot shows the RStudio interface with several panes:

- Top Bar:** Shows the RStudio logo, menu items (File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help), and system status (Jul 2, Tue 8:02:53 Andras Berczi).
- Header Bar:** Shows the current project (~project/elitecon-ds - master - RStudio) and a search bar.
- Console Tab:** Displays R code and its output. The output shows the command used to generate an HTML file from an RMD file, including pandoc options like --from markdown-autoLink_bare_uris+ascii_identifiers and --output git.html.
- Git Tab:** Shows the current branch (master) and a list of staged changes. The staged changes include files like figures/, prerequisite/git.Rmd, and prerequisite/git.html.
- Code Editor:** Shows an R script named git.Rmd with code related to Git basics and knitr setup.
- File Browser:** Shows the project structure under Home > project > elitecon-ds. It includes files like .gitignore, .Rhistory, elitecon-ds.Rproj, figures, and prerequisite.

Git add in Rstudio

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help.
- Header Bar:** Go to file/function, Addins, ~project/elitecon-ds - master - RStudio, eltecon-ds -
- Console Tab:** Shows the command `git add .` followed by the output of a pandoc conversion script.
- Git Tab:** Shows staged files: `figures/`, `prerequisite/git.Rmd`, and `prerequisite/git.html`.
- Code Editor:** Shows the R code for a `git.Rmd` file, which includes a setup chunk and a note about Git.
- File Browser:** Shows the project structure under `Home > project > elitecon-ds`:

Name	Size	Modified
..	50 B	Jul 1, 2019, 7:11 AM
.gitignore	0 B	Jul 1, 2019, 7:28 PM
.Rhistory	205 B	Jul 1, 2019, 11:01 PM
elitecon-ds.Rproj		
figures		
prerequisite		

Git commit in Rstudio

RStudio Environment:

- Console:** Shows R code execution and output.
- Code Editor:** Displays the file `git.Rmd` containing R Markdown code.
- File Explorer:** Shows the project structure with folders like `gitignore`, `.Rhistory`, `elitecon-ds.Rproj`, `figures`, and `prerequisite`.

Changes View:

Staged	Status	Path
<input checked="" type="checkbox"/>	<input type="checkbox"/>	prerequisite/git.Rmd
<input checked="" type="checkbox"/>	<input type="checkbox"/>	prerequisite/git.html

Commit Message:

```
Amend previous commit
Commit
```

Commit Content:

```
---  
title: "Git basics"  
output: html_document  
---  
# What is Git?  
Git is a version control system. Version control helps you to track the changes you have made in your project, so you can go back to an earlier stage if needed and it is a good, easily trackable way to collaborate with others.  
A project or repository is basically a folder where you have all your codes, files, subfolders, etc on your local computer. You can upload this repository to a remote server (GitHub, GitLab, etc), so you won't depend only on your computer.  
Also, this way others can collaborate to your project or you can download a project made by someone else and collaborate to it.  
What Git does is it tracks the files in your project. You can create a snapshot of it any time and upload it to a remote server.  
## How to use git?  
### Using Git with RStudio
```

Git commands

- git clone
- git status
- git add .
- git diff
- git commit -m "Commit message"
- git push
- git pull

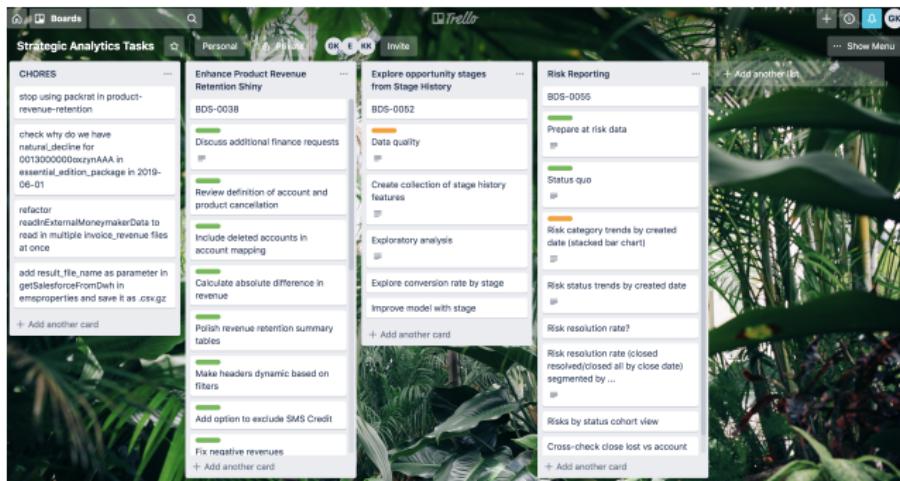
Git tips

- Commit early, commit often
- Use short but descriptive commit messages
- Keep your commits as small as possible, but keep your changes in each commit related
- Run the whole code you modified before pushing it to the repository
- Include files in .gitignore you don't want to track

Task management

Task management

- Manage your tasks with a task management system
- We use Trello and Google Spreadsheets



Summary

Summary

- R and RStudio
 - RStudio projects
 - Relative paths
- Annoying situations
- Code and data - Guidelines
 - A good directory structure
 - analysis.R
 - functions.R
- Version control with Git
- Task management

Sources

Sources

- <https://www.r-bloggers.com/structuring-r-projects/>
- <https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects>
- <https://web.stanford.edu/~gentzkow/research/CodeAndData.pdf>
- <http://www.informit.com/articles/article.aspx?p=1323426>
- <https://dzone.com/articles/naming-conventions-from-uncle-bobs-clean-code-phil>
- <https://www.quora.com/What-does-git-stand-for>