


**Hacettepe University
Department of Industrial Engineering
Undergraduate Program
2023-2024 Fall**

**EMU 430 – Data Analytics
Week4
October 27, 2023**

Instructor: Erdi Dasdemir

edasdemir@hacettepe.edu.tr
www.erdidasdemir.com


**Previously on
EMU430**




Productivity Tools



Git and GitHub



Quarto



I drew inspiration primarily from [Dr. Rafael Irizarry's "Introduction to Data Science" Book](#) and ["Data Science" course by HarvardX on edX](#) for the slides this week.

Previously on

EMU430

Conditional Execution

- Logical (boolean) expressions
- Comparison operators (`==`, `x>y` etc.)
- Logical operators (`&`, `|`, `!`)
- `if`: condition execution
- `if-else`: alternative execution
- `if- else if – else` : chained conditionals
- `if – else (if – else)`: nested conditionals
- `ifelse` statement with vectors

```
a <- c(0, 1, 2, -4, 5)
```

```
ifelse(a > 0, 1/a, NA)
```

```
Output: [1] NA 1.0 0.5 NA 0.2
```

- Any and all conditional functions

Functions:

```
my_function <- function(x, y, z) {  
  operations that operate on x, y, z, which are defined by the user when they  
  call this function.  
}
```

Example:

```
avg <- function(x, arithmetic = TRUE){  
  n <- length(x)  
  ifelse(arithmetic, sum(x)/n, prod(x)^(1/n))  
}
```

For loops

```
for (i in range of values) {  
    operations that use i, which is changing across the range of values  
}
```

While loops

```
while (condition is satisfied) {  
    operations  
}
```

We normally rarely use for loops in R

R functions:

apply

sapply

lapply

tapply

mapply

```
compute_s_n <- function(n){  
  x <- 1:n  
  sum(x)  
}
```

```
m <- 25  
# create an empty vector  
s_n <- vector(length = m)  
for (i in 1:m){  
  s_n[i] <- compute_s_n(i)  
}
```



```
sapply(1:m, compute_s_n)
```



```
sapply(1:4, sqrt)
```

```
# 1.000000 1.414214 1.732051 2.000000
```

```
# Equivalent to:
```

```
sapply(1:4, function(i) sqrt(i))
```

Generate sample data

```
# Data frame
```

```
df <- data.frame(x = 1:4, y = 5:8, z = 10:13)
```

Sample data

```
apply(x = df, margin = 1, fun = sum)
```

```
apply(df, 1, sum)
```

Productivity Tools

- Organizing files and document preparation

- **Principles:**

- 1. **Be systematic when organizing your filesystem:** Minimize time spent looking for something

- 2. **Automize when possible:** If you repeat the same task repeatedly, there is probably a way to automize it.

- 3. **Minimize the use of the mouse:** When your fingers leave the keyboard, you lose productivity

- Unix shell
 - use as a tool for managing files
 - permit us use keyboard rather than mouse when organizing folders and files.
- The data analysis process is iterative and adaptive → We are constantly editing our scripts and reports.
 - Version Control System
 - **Git:** a version control system; a powerful tool for keeping track of these changes.
 - **GitHub:** a service that permits you to host and share your code, facilitate collaborations
 - **Side effect:** Showcase your work to potential employers.
 - Write reports in Quarto (or Rmarkdown)
 - Incorporate text and code into a single document
 - Write reproducible and aesthetically pleasing reports by running the analysis and generating the report simultaneously.
 - Produce webpages, blogs, books etc.

1. Install R
 - the programming language
2. Install RStudio
 - the integrated desktop environment
3. Install Git:
 - The version control system to keep track of changes made to our code and to sync local copies of code with copies hosted on GitHub.
 - For Windows users only, installing Git installs Git Bash, which emulates Unix on Windows Machines.
 - Install Git → Install Git Bash, emulates Unix on Windows Machines
4. Open a GitHub account and sync it with RStudio
5. Download Quarto
 - An open-source scientific and technical publishing system for analyzing, sharing and reproducing

1. Installing packages with RStudio

- `install.packages("tidyverse")` or RStudio Tools → Install packages
- `library(tidyverse)`

2. Keeping organized with RStudio Projects

- a typical project often involves several scripts
- RStudio projects help to keep these files organized.
- **Demonstration:**
 - RStudio → File → New Project → New Directory → New Project
 - Find the best location for yourself,
 - Choose a meaningful folder name (use lower case letters, no spaces, hyphens to separate words)
 - Example: `my-first-project.Rproj`
 - Organize your file system following a hierarchical approach (Projects → `xxx.Rproj`, `yyy.Rproj`, ...)
 - RStudio right-top shows the name of the project you are working on

Git and GitHub

1. RStudio Project → Share, collaborate, and version control using GitHub
2. GitHub: hosting system with code
3. GitHub needs Git.
 - i. A version control system
 - ii. Can be used with
 - a. Unix
 - b. Rstudio
 - c. GitHub Desktop (a GUI application)

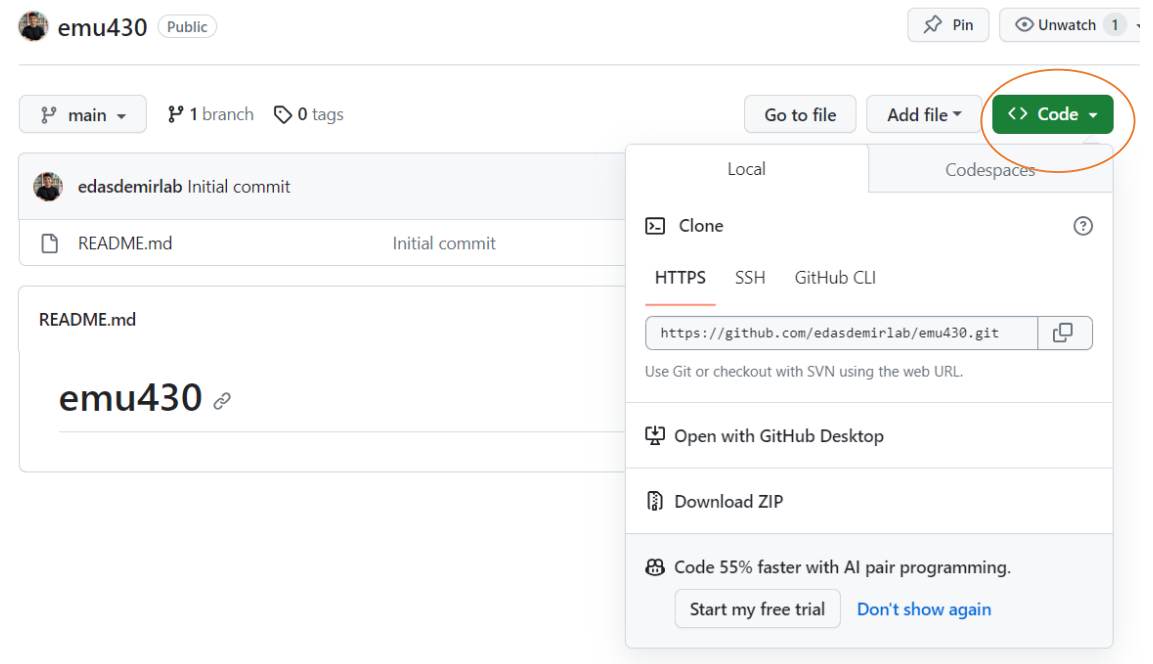
1. Go to git-scm.com → Downloads
2. Install the git.
 - i. Agree the licence
 - ii. Pick default editor for Git → select the one you are familiar with. If you do not recognize an editor you know among the options given, select Nano as your default editor.
 - iii. **Important!** Recommend selecting “Use Git and optional Unix tools from the Windows”
 - Permits learning Unix from within Rstudio
 - However, some commands that run on Windows Command Line will stop working (their Unix versions now will be effective). If you do not use your Windows command line, this will not be a problem.
 - iv. Select default installation options for the remaining.
3. Change your preference in RStudio to make Git Bash default Unix Shell (only for windows users)
 1. Rstudio → Tools → Terminal → Terminal Options
4. Open a new terminal in Rstudio (Rstudio → Tools → Terminal) to check if Git Bash is working.

1. Go to www.github.com to open an account (basic accounts are free)
2. Pick a username carefully → you will be sharing your account link with your collaborators, and potential employers
 - ✓ Short
 - ✓ Easy to remember and spell
 - ✓ Somewhat related to your name
 - ✓ Professional
 - ✓ Example: edasdemirlab
3. Connect GitHub Account to RStudio.
 1. RStudio → Global Options → Git / SVN
 2. Enter the path for the Git executable we just installed in the previous slide
 3. We will create SSH key → Create RSA Key



1. The General Idea: You will have two copies of your code:
 - i. One on your computer
 - ii. One on GitHub
2. If you add collaborators to this project, each will have a copy on their own computer.
3. GitHub copy is the “master copy” that each collaborator syncs to.
4. Gi will help you keep all the different copies synced.
5. Many data science companies use version control systems like Git → they might be impressed that you already know at least the basics.

1. Initialize repo on GitHub
 - Login to your account on www.github.com
2. Repositories → New
 1. Repository name (you may have dozens of repos, keep a clever name)
 2. Public repos are free but private ones require a monthly charge.
 3. Create a repo.
 4. GitHub → Code → Clone → Copy the link



You can use GitHub either with

1. [GitHub Desktop](#)
2. Terminal and Unix commands
3. RStudio

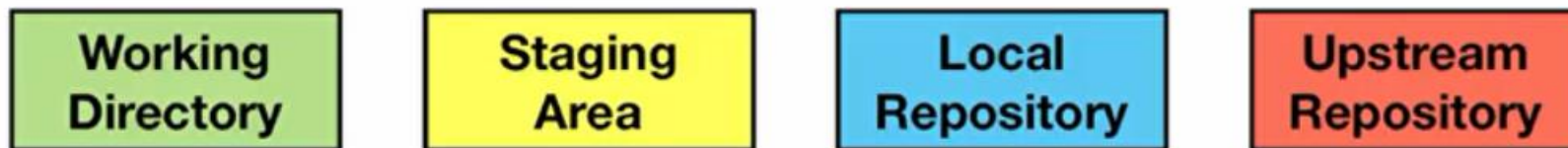
Three reasons to use them:

- 1. Version control:
 - keep track of changes we make to our code
 - Revert back to previous versions of our files.
 - Git also permits us to create branches in which we can test out ideas and then decide if we can merge the new branch with the original.

- 2. Collaborating
 - a central repo, you can have multiple people make changes to the code and keep the version synced.
 - **Pull request:** anybody to suggest changes to your code. You can easily either accept or deny these request.

- 3. The third reason is to **share**, and this is the main one we use here. Even if we do not take advantage of the advanced and powerful version control functionality, we can still use Git and GitHub to share our code.

To effectively permit version control and collaboration, in Git, files moves across for different areas



How to start?

1. Clone an existing repo
2. Initialize new one

Approach 1 to create a repository

- Clone <https://github.com/rairizarry/murders>: green clone button that you can click to copy the URL

- We have now cloned a GitHub repo and have a working git directory.

GitHub Desktop

- Fetch Original
- Pull origin
- Commit to main
- Push origin

- Create a file new-file.txt and commit to the repository.

Approach 2 to create a repository: **Initializing a GitHub Repository**

Initialize on our computer rather than cloning.

Create a local project independent of Git.

What is “reproducible report”?

Why it is important?

- Many scientific publications can be thought of as a final report of a data analysis.
- The reports are often textual descriptions of the findings along with some figures and tables resulting from the analysis.
- You have finished the analysis and the report, you are told that you were given the wrong dataset, and they sent you a new one and you are asked to run the same analysis but with this new dataset.
- Or, you realized a mistake was made, you fixed the code, and ran the same analysis.
- Or you used a code for your report and someone want so reproduce the results to learn more about the approach.

Quarto

- **Quarto:** code and text can be combined in the same document, and figures and tables are automatically added.
- Another tool is **R Markdown** (still in use, still R community supports it, but they stopped adding new features to it. The new features are developed for Quarto).
- What is the differences between Quarto and RMarkdown?
 - [FAQ for R Markdown Users](#)
 - Quarto is Posit's attempt to bring R Markdown to everyone! Unlike R Markdown, Quarto doesn't have a dependency or requirement for R. Quarto was developed to be multilingual, beginning with R, Python, Javascript, and Julia, with the idea that it will work even for languages that don't yet exist.

Quarto

- It is based on Markdown, a markup language that is widely used to generate HTML pages.
- www.markdowntutorial.com
- Unlike a word processor, such as Microsoft Word, where what you see is what you get, with Quarto, you need to compile the document into the final report.
- A Quarto document looks different than the final product. This seems like a disadvantage at first, but it is not at all. For example, instead of producing plots and inserting them one by one into the word processing document, the plots are automatically added.
- Rstudio → New File → Quarto Document
 - Title
 - Author

- To convert a Quarto file to a document we use `knitr` package
- .qmd extension
- In the template we have,
 - Header section between ---
 - output attribute defines the outputted file format.
 - R chunks
 - To write code
 - Compile document, this code will be evaluated

- R chunks
 - To write code
 - Compile document, this code will be evaluated
- Default settings will let R codes appear in the document
 - To avoid from this: adding argument `echo=FALSE`
- Recommendation: Get into the habit of adding a label to the R code chunks.

- We will prepare a report on Gun Murders:
 - Title: Report on Gun Murders
 - Output format (we can change this later)

- Example
 - `githubrepos/murders`
 - Open `report.qmd`
 - Change output type

- We will prepare a report on Gun Murders:
 - Title: Report on Gun Murders
 - Output format (we can change this later)

- Example
 - `githubrepos/murders`
 - Open `report.qmd`
 - Change output type

