EKN-812 Lecture 0

GitHub and RMarkdown Setup

Jesse Naidoo

University of Pretoria

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

What is git?

- a version control system in very wide use across the public and private sectors, and academia
  - a program that keeps track of each version of each file you *commit* into a *repository*
  - think of this like a very powerful version of the "track changes" feature in Word
  - this is useful even when working by yourself, but especially when working with others
- git has many capabilities
  - easy to get overwhelmed by all the learning materials online
  - I am not an expert!
  - we will go through a simple workflow that will be enough for our class
  - then, I'll point you to some resources if you want to learn more

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

git: Mechanics

- a *commit* is a snapshot of the state of all the files in your *repository*
  - commits are related to their "parents"
  - this creates a *history*
- either on the command line or via a graphical client (SourceTree, GitKraken, etc), you will
  - *stage* a file to be committed
  - compare the state of your working directory to the staging area, or across commits with `git diff`
  - once you're satisfied with your work, *commit* with a descriptive message, e.g. "added a scatterplot of exchange rates vs temperature"
- GitHub provides you with a platform for your *remotes* (can have none, or several)
  - you *push* changes from your local repo to the remote
  - you *pull* down changes from the remote to your local
  - if there are *merge* conflicts, git will force you to resolve them and then commit manually

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

What is GitHub?

- a code-hosting site
  - useful for collaborative software development
  - can also be used for research and teaching
  - alternatives: BitBucket, GitLab, SourceForge, others
- basic idea behind git is *distributed version control*
  - each developer has their own copy of the project (a *repository*)
  - can work independently on different parts of project
  - can experiment with different features in *branches*, again independently of others
- GitHub (or other code hosting sites) provide a convenient platform and set of protocols for merging and synchronizing work
  - we will mostly exploit the ability to distribute starter code or documents
  - and, you will submit homework using the platform

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Worst-Case)

- sign up for an account on github.com
  - please use your real name and upload a picture of yourself
  - this helps me learn your names!
- sign in and navigate to one of your repositories
  - for now think of this as a folder or directory
- from the web interface, you can
  - create plain-text files
  - upload other files (data, PDFs, images)
  - commit changes
  - remove files
  - compare ("diff") commits
  - keep track of priorities in the issue tracker

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Mechanics: How To Use It (Better Case)

- this workflow assumes you have
  - installed git on your local machine
  - created either SSH or HTTPS keys so you can push and pull to/from GitHub
  - authorized RStudio to commit, push and pull on your behalf
  - follow the installation instructions in "Happy Git with R" to do this
- each assignment will be created as a new repository for you
  - clone it down to your local machine
  - create a new branch immediately (at the command line `checkout -b my-new-branch`)
  - work on it (even offline), committing to your local repo as you see fit
  - when you're ready to hand in, push to GitHub and start a pull request
  - I will be able to see it and add comments; when ready we can approve the pull request and merge in the changes

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Why Bother?

- At the very least, not worse than ClickUP, and better than email (or paper)
- Try to develop habits (reproducibility, documentation) useful for later work
- A nice interface for organizing larger projects with git
  - more advanced usage requires you to install git locally
  - allows you to experiment with new features before you *push* to a *remote*
  - follow the instructions in "Happy git with R" closely

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

Further Reading

- confused about what git, GitHub etc are and why you should care?
  - Wei Yang Tham's "version control for economists" workshop (link)
  - Matt Gentzkow and Jesse Shapiro's handbook on "code and data for the social sciences"
  - "Excuse me, do you have a moment to talk about version control?" (link)
- Jake Torcasso's slides (link)
  - more details about the structure of git repos and branching
  - might be more advanced than we need at the moment
- most useful resource: happygitwithr.com
  - to get git working on your local computer and talking to GitHub: Ch. 4 - 13
  - don't worry - they are short chapters!
- Open Science Framework (osf.io)
  - free platform for code and data hosting

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably have that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

RMarkdown Mechanics: How Can We Use It?

- install R and RStudio
- you will also need a LaTeXdistribution
  - MikTeX is a common choice for Windows
  - MacTex is a common choice for OS/X
  - on Linux, use your favorite package manager to install TeXLive
    - but you probably knew that already
- see Ch. 27 and 29 of "R for Data Science" (r4ds.had.co.nz) for a full introduction
- if this sounds like too much work, you can use the Econ lab

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^{\infty} \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^\infty \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^\infty \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^\infty \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^{\infty} \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^\infty \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^{\infty} \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Mathematical Typesetting

- you can write LaTeXcode directly in RMarkdown
- some common tasks:
  - Greek letters (α, β, γ)
  - superscripts x^2 or subscripts x_j
  - derivatives and integrals:

$$\frac{\partial f}{\partial K}(K, L) = \int_0^\infty \alpha(s) g(s) ds$$

  - limits: $\lim_{x \to 0} f(x) = 1$
  - equation referencing:

$$e^{i\pi} + 1 = 0. \tag{1}$$

    Equation (1) ties together five fundamental constants of mathematics, and is considered very beautiful by mathematicians.

- see Overleaf.com's guide "learn LaTeXin 30 minutes" (link) for more

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Benefits of Using RMarkdown

- reproducibility
  - data analysis integrated directly into documents: eliminates copy-paste errors
- transparent mapping from code $\rightarrow$ numerical results $\rightarrow$ visual information or summaries
  - reduces ambiguities: how did you define the price index in section 2?
- lightweight, simple language that "just works" for many output formats
  - PDF articles
  - beamer, Xarigan, etc presentations
  - websites
  - even Word, if you really want

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File > New File > R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File > New File > R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File > New File > R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File > New File > R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File $>$ New File $>$ R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File $>$ New File $>$ R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Installation

- in RStudio:
  - `install.packages(tidyverse)`
  - `install.packages(rmarkdown)`
- open the PDF template by clicking File $>$ New File $>$ R Markdown
  - click "Knit"; you should see a PDF pop up with a scatterplot
  - if not, read the error messages and look them up!
- again, at worst you can use the Econ lab

Table of Contents