

# Day 02: More Complex Data Structures

Erin Rossiter

30 January, 2023

## Today's plan

# Today's plan

- Github Skills
- Review
  - » Vectors
  - » Object classes
- More complex data structures
  - » Matrix, list, dataframe
  - » Working with logicals
- Inclass activity
- Rmd help
- ChatGPT discussion

# Announcements

# Announcements

- Syllabus updated
  - » drop lowest **two** psets
  - » final project due dates
  - » chatGPT guidelines
- Still todo: suggested readings (more important for advanced topics)
- Next week: we'll have a discussion about projects

## Github Skills

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)



# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » (github.com/erossiter/ProgrammingSpring2023)
- Forked it to create *your* repository
  - » (github.com//ProgrammingSpring2023)
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (~/.documents/github/ProgrammingSpring2023)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » (github.com/erossiter/ProgrammingSpring2023)
- Forked it to create *your* repository
  - » (github.com//ProgrammingSpring2023)
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (~/.documents/github/ProgrammingSpring2023)

# Skillset #1: Initial set up – downloading code for local use (review)

Using the online interface, you:

- Navigated to *my* repository
  - » ([github.com/erossiter/ProgrammingSpring2023](https://github.com/erossiter/ProgrammingSpring2023))
- Forked it to create *your* repository
  - » ([github.com//ProgrammingSpring2023](https://github.com//ProgrammingSpring2023))
- Cloned your repository to your computer using Github Desktop
- Opened, edited, saved files *locally*
  - » (`~/documents/github/ProgrammingSpring2023`)

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?



## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?



## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Skillset #2: Personal workflow – committing and pushing changes online

### **(Don't do these steps yet)**

Today, we need to:

- “Commit” edits locally
  - » Review changes
    - Red is the old line; green is the new line
  - » Give a good Summary (title) and Description
- “Push origin”
  - » “You have 1 local commit waiting to be pushed to GitHub”
- Review online, noting:
  - » Your last commit shows up many places
    - Upper left of repo
    - Next to affected folders and files
    - History of all commits, click on “X commits” in upper right
  - » Why is this useful?

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit



## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

## Personal workflow cont.

- Let's focus on local version control
- It can help you with your own organization, transparency, and experimentation
- You can see exact changes as you work and keep them (or not!)
- Tips:
  - » You don't have to push every local commit online
    - Why might I commit and not push? What's the difference?
  - » You can try something and revert back easily
    - Discard changes to a file before committing or
    - Undo a commit

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online



# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online



# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

# Inclass Activity

1. Change the names of any files I provided but you edited (e.g., add initials on the end)
  - » in class activities
  - » problem set
  - » etc.
2. Examine “Changes” tab in Github Desktop
  - » Click through the files and skim through all changes
3. Write an informative Summary and Description
4. Commit (remember this is just a local commit)
5. Push online
6. Examine changes online (hint: refresh browser)
7. Go through process again
  - » make a simple edit to any file
  - » review, commit, push, view online

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)



## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

## Skillset #3: Collaborative workflow – merging upstream changes

I'll commit and push changes to *my* repo

- problem sets
- class content
- etc.

You'll want to merge my latest changes to your fork aka:

- update branch
- merge upstream changes
- sync your fork
- fetch my commits
- (sorry there's so much new language)

# How? Online version

Ex: instructor added new slides, I added my homework → no conflicts → “Update branch”

- Do not discard your work!
- Make sure all local changes are committed and pushed first

The screenshot shows a GitHub repository page for 'erossiter/WUSTL', which is a public fork of 'justingrimmer/WUSTL'. The repository is currently on the 'master' branch, which is 1 branch away from the upstream. The file list shows several files, including 'HW', 'Afternoon\_RCode.R', 'ConsumerNarratives.csv', 'Debate1.html', 'Mach.tar', and 'StudentDrinking.RData'. A warning message indicates that the branch is out-of-date because it is 25 commits behind the upstream repository. The warning suggests discarding the 25 commits or updating the branch to merge the latest changes from the upstream repository.

erossiter / WUSTL Public

forked from justingrimmer/WUSTL

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[master](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is **25 commits ahead, 6 commits behind** justingrimmer:master. [Contribute](#) [Sync fork](#)

erossiter Done with HW5	
HW	Done with HW5
Afternoon_RCode.R	LASSO Code
ConsumerNarratives.csv	Day 5 material
Debate1.html	added stuff
Mach.tar	Day 5 material
StudentDrinking.RData	Day 4 Material

**This branch is out-of-date**

Update branch to merge the latest changes from the upstream repository into this branch.

Discard 25 commits to make this branch match the upstream repository. 25 commits will be removed from this branch.

[Learn more](#)

[Discard 25 commits](#) [Update branch](#)

5 years ago

# How? Online version

Ex: instructor added new slides, I added my homework → no conflicts → “Update branch”

- **Do not discard your work!**
- Make sure all local changes are committed and pushed first

The screenshot shows a GitHub repository page for 'erossiter/WUSTL', which is a fork of 'justingrimmer/WUSTL'. The repository is public and has 1 branch (master) and 0 tags. The file list shows several files, including 'HW', 'Afternoon\_RCode.R', 'ConsumerNarratives.csv', 'Debate1.html', 'Mach.tar', and 'StudentDrinking.RData'. A warning box indicates that the branch is out-of-date, being 25 commits ahead and 6 commits behind the upstream repository. The warning suggests updating the branch or discarding the 25 commits.

erossiter / WUSTL Public  
forked from justingrimmer/WUSTL

<> Code Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file <> Code

This branch is 25 commits ahead, 6 commits behind justingrimmer:master. Contribute Sync fork

**This branch is out-of-date**  
Update branch to merge the latest changes from the upstream repository into this branch.  
Discard 25 commits to make this branch match the upstream repository. 25 commits will be removed from this branch.  
[Learn more](#)

erossiter Done with HW5	
HW	Done with HW5
Afternoon_RCode.R	LASSO Code
ConsumerNarratives.csv	Day 5 material
Debate1.html	added stuff
Mach.tar	Day 5 material
StudentDrinking.RData	Day 4 Material

Discard 25 commits Update branch

5 years ago

# How? Online version

Ex: instructor added new slides, I added my homework → no conflicts → “Update branch”

- **Do not discard your work!**
- Make sure all local changes are committed and pushed first

The screenshot shows a GitHub repository page for 'erossiter/WUSTL', which is a fork of 'justingrimmer/WUSTL'. The repository is public and has 1 branch (master) and 0 tags. The main branch is 'master'. The repository is 25 commits ahead and 6 commits behind the upstream repository 'justingrimmer:master'. A warning message states: 'This branch is out-of-date. Update branch to merge the latest changes from the upstream repository into this branch. Discard 25 commits to make this branch match the upstream repository. 25 commits will be removed from this branch. Learn more'. The file list includes: HW (Done with HW5), Afternoon\_RCode.R (LASSO Code), ConsumerNarratives.csv (Day 5 material), Debate1.html (added stuff), Mach.tar (Day 5 material), and StudentDrinking.RData (Day 4 Material). The repository was created 5 years ago.

erossiter / WUSTL Public

forked from justingrimmer/WUSTL

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[master](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is **25 commits ahead, 6 commits behind** justingrimmer:master. [Contribute](#) [Sync fork](#)

**erossiter Done with HW5**

HW	Done with HW5
Afternoon_RCode.R	LASSO Code
ConsumerNarratives.csv	Day 5 material
Debate1.html	added stuff
Mach.tar	Day 5 material
StudentDrinking.RData	Day 4 Material

**This branch is out-of-date**

Update branch to merge the latest changes from the upstream repository into this branch.

Discard 25 commits to make this branch match the upstream repository. 25 commits will be removed from this branch.

[Learn more](#)

[Discard 25 commits](#) [Update branch](#)

5 years ago



# How? Online version

Ex: instructor added new slides, I added my homework → no conflicts → “Update branch”

- **Do not discard your work!**
- Make sure all local changes are committed and pushed first

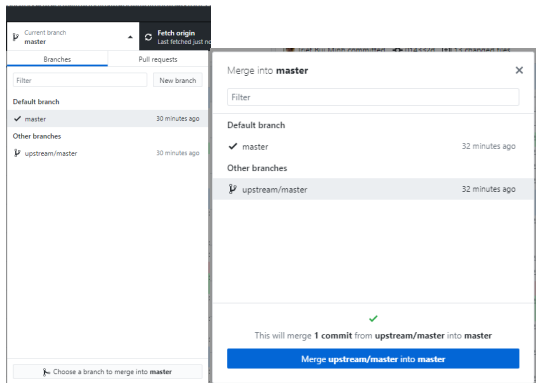
The screenshot shows a GitHub repository page for 'erossiter/WUSTL', which is a public fork of 'justingrimmer/WUSTL'. The repository has 1 branch (master) and 0 tags. The file list includes:

erossiter Done with HW5	
HW	Done with HW5
Afternoon_RCode.R	LASSO Code
ConsumerNarratives.csv	Day 5 material
Debate1.html	added stuff
Mach.tar	Day 5 material
StudentDrinking.RData	Day 4 Material

A warning box titled 'This branch is out-of-date' is displayed, indicating that the branch is 25 commits ahead and 6 commits behind the upstream repository. It provides instructions to update the branch or discard the 25 commits. The repository was last updated 5 years ago.

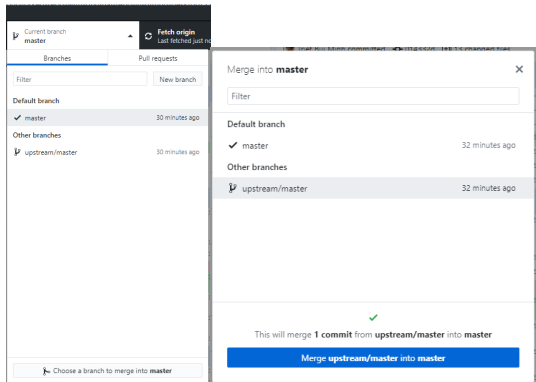
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



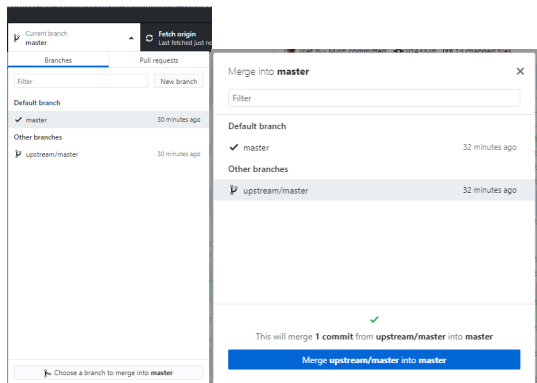
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



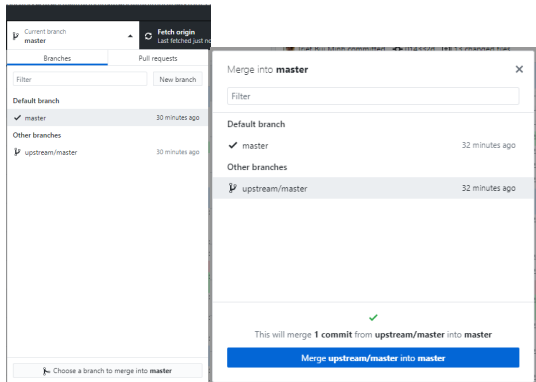
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



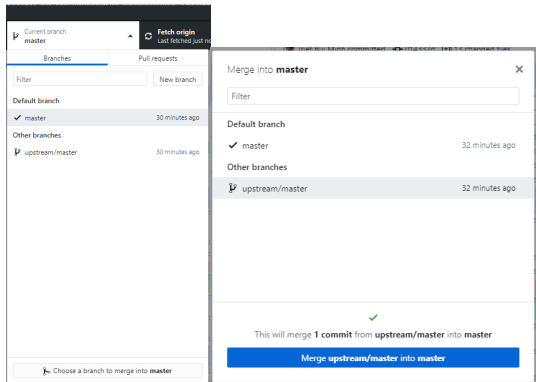
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



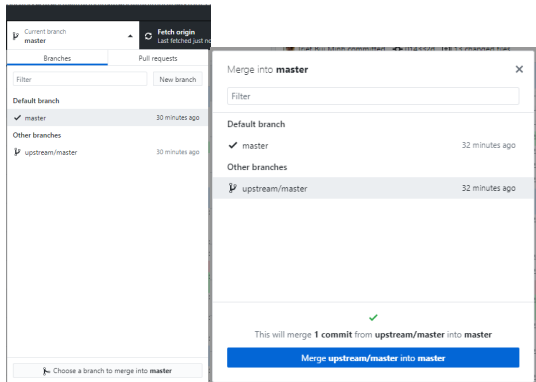
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



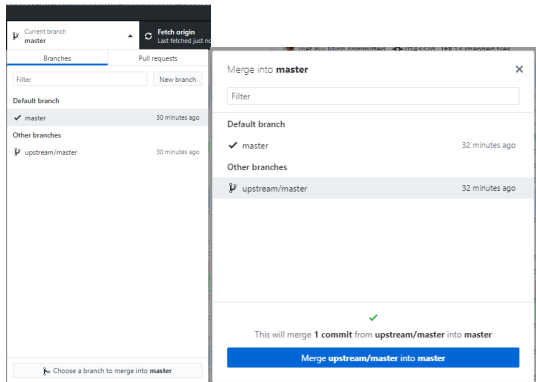
## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned



## How? Desktop Version

- Make sure all local changes are committed and pushed first
- Click on the “current branch” tab and click the “choose a branch to merge into master” button at the bottom
- Select “upstream/master” (that’s my repo)
  - » Pulls the changes down from my repo
  - » Bring your local clone up to date
- Then you can push changes online as we already learned





## Rmd Overview

## ChatGPT Discussion

# What's going on?

ChatGPT is a generative text model, trained on a wide variety of source texts, such that it interacts in a conversational way *even generating R code*.

- Discussion of pros, cons, successes, and limitations if you're interested
- Banned from Stackoverflow because wrong too often

# What's going on?

ChatGPT is a generative text model, trained on a wide variety of source texts, such that it interacts in a conversational way *even generating R code*.

- Discussion of pros, cons, successes, and limitations if you're interested
- Banned from Stackoverflow because wrong too often

# What's going on?

ChatGPT is a generative text model, trained on a wide variety of source texts, such that it interacts in a conversational way *even generating R code*.

- Discussion of pros, cons, successes, and limitations if you're interested
- Banned from Stackoverflow because wrong too often

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.



# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

# Pros/cons for this class

## Pros

- Since it synthesizes available resources into one succinct answer. For basic R, I've found it is correct and gives good explanations.

## Cons

- It will impede your learning to just copy/paste the answers. If you don't generate the code yourself, you might become reliant, which isn't ideal because:
  - » ChatGPT might not be available forever (e.g., might require payment, business could go under, etc.)
  - » You might not become fluent in the basics, which will slow you down
- Hasn't been trained with texts newer than 2021

## Neutral

- It might not help with advanced coding, which is our goal
- From a pedagogical standpoint, I'm not sure the best way to teach the language if I'm letting someone else speak for you.

## Guidelines:

Balancing these pros and cons, I have the following rules, now on the syllabus:

- Use it on any question unless I explicitly say not to.
- Indicate in writing exactly what part of the answer chatGPT gave, including if it was the entire thing.
- Each *literal* keystroke must be your own as it says on the syllabus. Seriously. Type it yourself. (If I find evidence of copy/paste, the PSet grade will be a 0.)
- Finally, I trust that if chatGPT gives an answer, you'll dig into it to make sure you see why/how the code is working.



## Guidelines:

Balancing these pros and cons, I have the following rules, now on the syllabus:

- Use it on any question unless I explicitly say not to.
- Indicate in writing exactly what part of the answer chatGPT gave, including if it was the entire thing.
- Each *literal* keystroke must be your own as it says on the syllabus. Seriously. Type it yourself. (If I find evidence of copy/paste, the PSet grade will be a 0.)
- Finally, I trust that if chatGPT gives an answer, you'll dig into it to make sure you see why/how the code is working.

## Guidelines:

Balancing these pros and cons, I have the following rules, now on the syllabus:

- Use it on any question unless I explicitly say not to.
- Indicate in writing exactly what part of the answer chatGPT gave, including if it was the entire thing.
- Each *literal* keystroke must be your own as it says on the syllabus. Seriously. Type it yourself. (If I find evidence of copy/paste, the PSet grade will be a 0.)
- Finally, I trust that if chatGPT gives an answer, you'll dig into it to make sure you see why/how the code is working.

## Guidelines:

Balancing these pros and cons, I have the following rules, now on the syllabus:

- Use it on any question unless I explicitly say not to.
- Indicate in writing exactly what part of the answer chatGPT gave, including if it was the entire thing.
- Each *literal* keystroke must be your own as it says on the syllabus. Seriously. Type it yourself. (If I find evidence of copy/paste, the PSet grade will be a 0.)
- Finally, I trust that if chatGPT gives an answer, you'll dig into it to make sure you see why/how the code is working.

## Guidelines:

Balancing these pros and cons, I have the following rules, now on the syllabus:

- Use it on any question unless I explicitly say not to.
- Indicate in writing exactly what part of the answer chatGPT gave, including if it was the entire thing.
- Each *literal* keystroke must be your own as it says on the syllabus. Seriously. Type it yourself. (If I find evidence of copy/paste, the PSet grade will be a 0.)
- Finally, I trust that if chatGPT gives an answer, you'll dig into it to make sure you see why/how the code is working.

Upcoming

# Upcoming

- PS01 due Tuesday (tomorrow) at 4pm EST via GitHub commit
  - » late assignments will not be accepted
  - » practice committing ASAP
- PS02 distributed tomorrow
- Next class – flow control (ifelse, loops) and advanced functions

# Upcoming

- PS01 due Tuesday (tomorrow) at 4pm EST via GitHub commit
  - » late assignments will not be accepted
  - » practice committing ASAP
- PS02 distributed tomorrow
- Next class – flow control (ifelse, loops) and advanced functions

# Upcoming

- PS01 due Tuesday (tomorrow) at 4pm EST via GitHub commit
  - » late assignments will not be accepted
  - » practice committing ASAP
- PS02 distributed tomorrow
- Next class – flow control (ifelse, loops) and advanced functions



# Upcoming

- PS01 due Tuesday (tomorrow) at 4pm EST via GitHub commit
  - » late assignments will not be accepted
  - » practice committing ASAP
- PS02 distributed tomorrow
- Next class – flow control (ifelse, loops) and advanced functions

# Upcoming

- PS01 due Tuesday (tomorrow) at 4pm EST via GitHub commit
  - » late assignments will not be accepted
  - » practice committing ASAP
- PS02 distributed tomorrow
- Next class – flow control (ifelse, loops) and advanced functions