Slide 1: Hi, I'm Tobin Magle, the Data Management Specialist at the Morgan Library at Colorado State University. This session will cover to make your research more reproducible.

Slide 2: Specifically, we're going to

- Define reproducible research
- Discuss its advantages
- And show you how to do it with R studio

Slide 3: Normally, we think of research in the following cycle, from hypothesis generation, experimental design, data collection and analysis, and publishing results. However, new complications have arisen due to technological advances and thus increased human error.

1. New technology produces huge, complex datasets that require automation to analyze. It also gives us the ability to share more than just a manuscript in a print journal.
2. There's also a healthy dose of human error, both in the form of insufficient reporting and mistakes in analysis.

Slide 4: Because of these complications, the research cycle looks more like this, where there's multiple complex steps in data processing and analysis and we can share data and code in addition to research articles.

Slide 5: There are many ways to define and subdivide reproducible research. But for our purposes, reproducible research is the practice of distributing ALL data, software, source code and tools required to reproduce the results discussed in a research publication.

Slide 6: Put simply, reproducible research = data + code

Slide 7: And even more simply, reproducible research is about making the process of research more transparent and reliable.

Slide 8: This session will cover tools that help integrate reproducible practice into the data processing and analysis and make getting these research outputs shared easier.

Slide 9: Before we get too far into the details, I want to discuss the difference between reproducibility and replication, because these words are used differently in different fields. For our purposes today,

- **Replication** is answering a research question with a new experiment and coming to the same conclusion, which is the scientific gold standard.
- However, replication is not always feasible, such as costly clinical trials, or measuring real world events in real time like climate data.
- **Reproducibility** is the minimum standard to assure scientific validity: providing enough information to get the same results from the same data and the same source code.

- Reproducibility is especially important in these cases that can't be independently verified.

Slide 10: I find it useful to think of reproducibility as a spectrum from providing results and a description of the methods in a publication to full replication. There's no one "right" way to do things, but it's good to strive to be as close to the right end of the spectrum.

Slide 11: Coding in general and R Studio specifically have tools that make it easier to make your analysis reproducible. We're going to discuss using R scripts, git and R markdown.

Slide 12: Let's start with scripts. To accurately report how you went from the raw data that you collected to our research results in the form of visualizations and statistical tests. Optimally, these instructions should be in the form of executable code, but at a minimum, a written description of the exact steps you took to clean and analyze your data is necessary.

Slide 13: Research is repetitive by nature. We have to run independent replicates, we do the same assays on different samples, and we run longitudinal experiments.

Slide 14: When you're repeating the same thing over and over, doing things by hand is really inefficient. It's slow from the outset, it's hard to repeat and document.

Slide 15:
Exercise 1:
Let's use making graphs in excel as an example. Take 5 minutes, download the data and write a verbal description of how you would make that graph.

- Question: was describing your steps easy?
- Do you think someone could make the exact same steps based solely on your instructions?

Slide 16: You could avoid all that by automating this process. Learning how code takes time up front, but makes documentation fast and easy to repeat.

Slide 17: Back to the making graphs example, I wrote a script to do this. So if my collaborator wants to see how it was done, I can send her the script, and she can run it on her own in about a second.

Demo 1:

- Double click the R script file
- It opens in R studio
- Highlight all the text
- run the code (ctrl-enter)
- See the graph

Slide 18: Unfortunately, it's not that simple. The R language and its packages change over time, which can cause your code to break. To achieve reproducibility, you have to record additional information about your analysis.

- What software did you use?
- What version and settings were used?

Slide 19: Luckily, these questions can be answered using the sessionInfo() command in R studio.

Demo 2:

- Go to the Console window (lower left)
- run the sessionInfo() command
- See output in the terminal

as you can see it tells me the version of R and the OS it was running in, as well as the packages that were loaded when I ran the script.

Slide 20: Data analysis typically evolves over the course of a project. Usually people keep a backlog of older versions by saving multiple, similar versions of the same file. If you're really good, and really consistent this can work fine, but we're all human and our file names can devolve into chaos when the project gets tough.

Slide 21: This is where formal version control systems, like git, come in handy. These systems save all the changes made over time and allows you to recall specific versions. It records who made the changes, and if you're collaborating, identifies conflicting changes between versions. It's widely used for coding, but can be used with basically anything.

Slide 22: Today we're going to use git within R studio connected to a public git repository online. First, go to git hub and make a new repository

Demo 3:

- Go to github.com
- Log into your github account
- Click the plus sign on the upper right
- Select New repository
- Name it data-donuts
- Click Create repository

Slide 23: The rest of this demo requires that you have git and R Studio installed. then, tell R studio where git is installed.

Demo 4:

- Go to <preferences> under tha RStudio menu
- Select the <git SVN> tab
- Select the location of the git executable

Slide 24: Then make a new version controlled project

Demo 5:

- File > New Project
- Version Control
- Select git

Slide 25: Now link your remote GitHub repository with the project you just made

Demo 6:

- Go back to your online repository
- Click the green clone or download button on the right
- Copy the URL
- Paste it in the Repository URL field of the Git repository

Slide 26: What just happened here is that R studio just "cloned" your project for you. It took the repository from github (the remote), and brought it down to your computer (local) where you can make changes and add them to the repo. Our repo is empty, but note that it would pull any files down from a non-empty repository.

Slide 27: Now let's add some files. The repo won't track files unless you tell it to, so now we have to add the files to the "staging area" by clicking the check boxes next to the files.

Demo 7:

- Drag .csv file, the R script and the .Rmd file into the version controlled directory
- Look in the git tab on the upper right: see that they appear. This list is only files with untracked changes
- Click the check boxes to add them to the staging area
- Now these files are ready to be committed to the repository

Slide 28: What we just did is add the changes to the staging area

Slide 29: But to add these to the repo, we need to commit the changes in the staging area to the repo.

Demo 8:

- Clicking commit
- A window appears
- Look at the changes are going to be committed
- Type in a commit message
- hit commit.

Slide 30: Now the changes have been moved from the staging area to the repository. We can repeat this process by making changes to the script file and seeing that they show up in the changed file again

Slide 31:
Exercise 2

- Change the script file
- The file appears in the git tab
- Add the file by clicking the checkbox
- Commit by pressing the commit button, and adding a commit message
- Click commit

Slide 32: Note that at this point, all of our changes are stored in the repository, but only locally. If you want to back them up to the remote repository, you have to "push" them to the remote. To do this…

Demo 9:

- Go to the commit window
- Hit the push button on the upper right
- Look at your repository to see the added files

Slide 33: What just happened is that the newly updated repository (but not the staging areas) was pushed up to the git hub server.

Slide 34: Finally, we're going to talk about literate programming, which basically means alternating human readable text and machine readable code. For this, we're going to use R markdown.

Slide 35: we're not going to go over every possible thing you can do in R markdown today, but the r markdown cheat sheet is a good reference to look up syntax for things I don't explicitly cover. The general workflow for using R markdown goes as follows:

- Open
- Write
- Embed
- Render

Slide 36: to use markdown in R, you need to install a few packages: knitr and markdown, which I have preinstalled. You also need to install a version of TeX on your computer if you want to be able to produce PDF reports.

Slide 37: First create a markdown document

Demo 10:

- File>New>R Markdown
- Fill in Title and Author
- Click Ok

Slide 38: Now you can write. This is a good time to outline your analysis before you start coding. You can use rich text features like italic, bolding, headers and lists.

Demo 11:

- Headers - #
- Bulleted lists - *
- Numbered links – 1.
- Links [text](link)
- Pictures ![](link)

Slide 39: Next, embed your code. You can do this inline, or in code chunks

Demo 12:

- Section to download data
- Section to load data
- Section to make the graph

Additionally, you can alter the display options (evaluate the code? Display the code and results? Display warnings?)

Slide 40: Now that you have things the way you want them, you can render: the document will only render if the code has no errors.

Demo 13:

- Click knit
- See the HTML document open up

Now you have code that you know runs in a portable document that others can read.

Slide 41:

Exercise 3:

- Open a new markdown document
- Use the formatting commands from the R Markdown cheat sheet to create a bulleted list that looks like the content of this slide.
- https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

Slide 41: To put this all into the context of the research cycle, you can use scripts and version control as you're conducting your analysis. Additionally, you can use RMarkdown with your git hub scripts and session info to report your findings in a reproducible manner.

Slide 42: Here's a handy checklist that you can use to determine how reproducible your research workflows are currently.

Slide 43: Finally, if you need help, you can email me, visit our data management services website, or use the online content linked on this slide to learn more. Thanks for coming. I hope this session was useful.