Helpful tools for accessible and reproducible research

Hansen Johnson

PhD Student
Oceanography Department, Dalhousie University
hansen.johnson@dal.ca

MEOPAR Annual Training Meeting
Victoria, BC
June 11, 2019

Presentation online at:

`https://hansenjohnson.org/talk/2019_meopar_atm/`

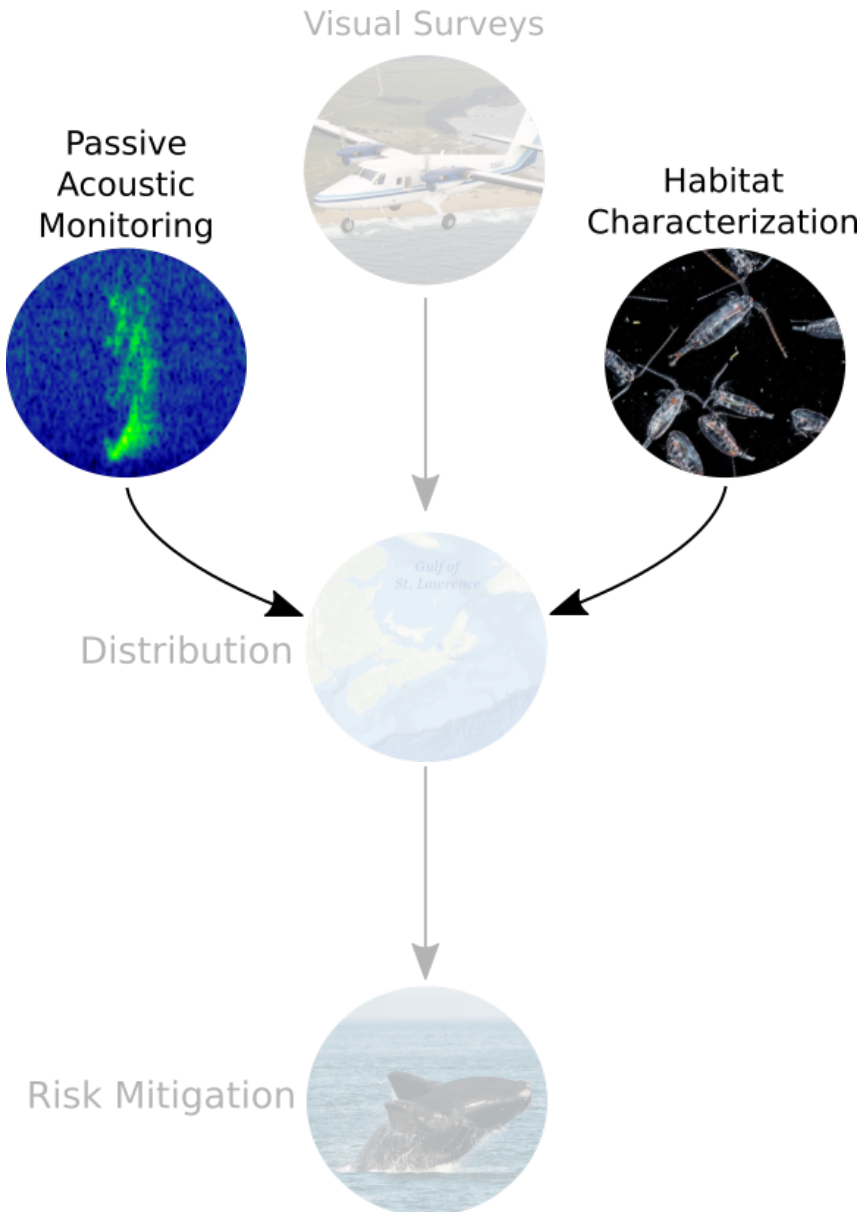
Tools of the trade

Many academic programs teach research concepts, but expect technical skills

- Most projects rely heavily on technology
- Little time or resources are allocated to developing technical skills and best practices
- These can make a HUGE impact for accessibility, efficiency and reproducibility
- Students must spend their limited time learning for themselves

My background

- Biology major in undergrad
- No training in computer programming or technical aspects of research before starting grad school (2015)
- Given a project that is **impossible** without technical chops



My background

- Luckily I have interest and supportive advisers
- Developed many helpful skills with help from my peers (especially Christoph Renkl) and the internet
- Hope to help others acquire these skills more efficiently

Some examples:

Methods in Ten Minutes:
`https://christophrenkl.github.io/mtm/`

R/Python Programming Tutorials:
`https://christophrenkl.github.io/programming_tutorials/`

Today's Goal

Goal: Provide some tools and concepts that I find essential for research

- Imagine we've been given some data on sea ice coverage and asked to characterize how it has changed over time
- Approach this simple project in 3 steps:
  1. Analyse the data
  2. Write a report
  3. Document the workflow
- We'll pause briefly after each section for questions and/or discussion

Disclaimer: these are the subjective opinions of a non-expert

Analysis

Goal: Process and plot some data

1. Structure the project
2. Read, process, and save data
3. Make and save plots

You will need

R (`www.r-project.org`)
Rstudio (`www.rstudio.com`)

A good project structure

A well-structured project allows you or someone else to easily understand and even reproduce the workflow

Organizing a project helps you:

- Expand, revisit and update efficiently
- Have confidence in the results
- Collaborate easily

Project structure

Projects vary and organizing them is hard. Some tips:

- Keep an untouchable 'sacred data directory' for raw data
- Dedicated directories for outputs (processed data and plots)
- Use simple file/folder names (ideally without spaces)
- Try to be consistent among projects
- Document prolifically (more later)

Check out `CookieCutterDataScience` for more details

Example [simple] project structure

- `example` — Project directory
  - `data` — All data
    - `processed` — Processed data by code in src
    - `raw` — Raw data - **never touch!**
  - `figures` — Plots produced by code in src
  - `reports` — Any reports or presentations
  - `src` — All source code
  - `wrk` — Development sandbox
  - `readme.md` — Project description
  - `master.R` — Master script

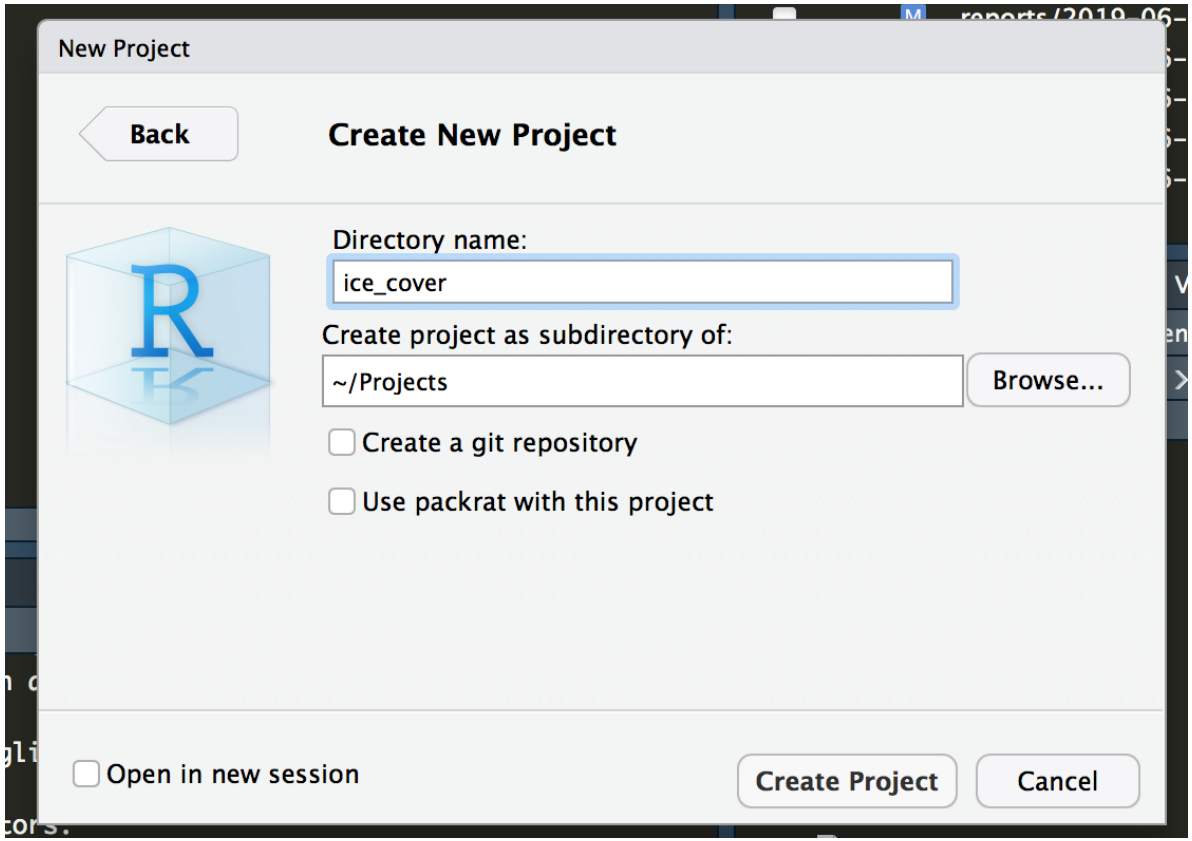
R and Rstudio

The basics of R and Rstudio are outside the scope of this session. See the tutorial here for more information:

`https://christophrenkl.github.io/programming_tutorials/`

R and Rstudio

1. Open Rstudio
2. Create new project in a logical place with a short, descriptive name (e.g., `~/Projects/ice_cover`)



Get the data

Download data from:
`https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/sea-ice.html`

Save the file in `data/raw/`

Process the data

Create a script called `src/process_data.R` to:

1. Read in data from `data/raw/`
2. Clean and format
3. Save output in `data/processed/`

src/process_data.R

```
## process_data ##
# Read, process, and save ice cover timeseries data

# input ------------------------------------------------------------------

# choose data file
infile = "data/raw/1.SeaIce-NCW-EN.csv"

# choose output file
outfile = "data/processed/ice_cover.rda"

# process ----------------------------------------------------------------

# read in data and rename columns
df = read.csv(infile, skip = 2, col.names = c("year", "ice_cover"))

# remove missing values
df = df[complete.cases(df),]

# format year
df$year = as.numeric(as.character(df$year))

# save
save(df, file = outfile)
```

Plot the data

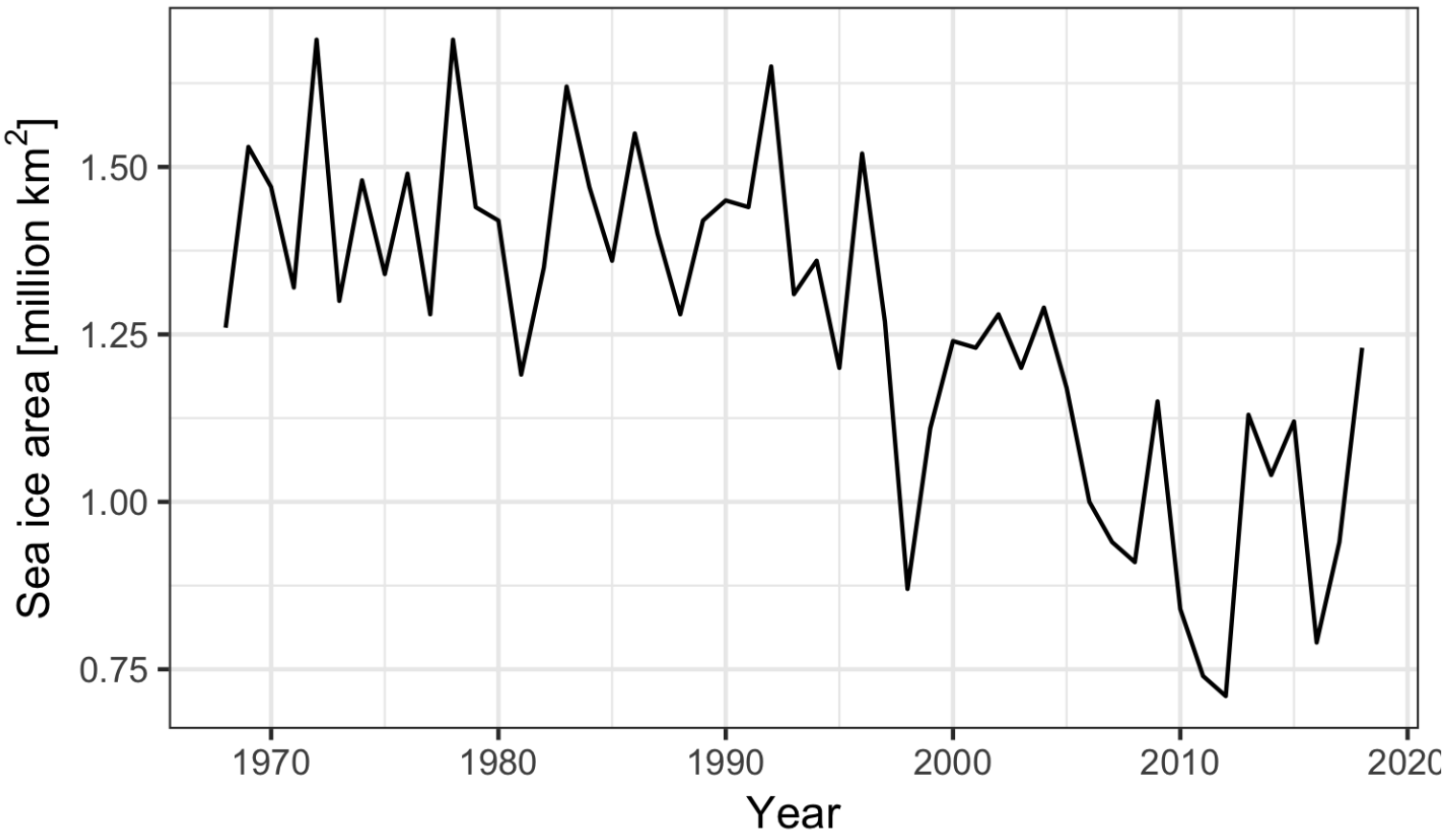
Create a script called `src/plot_timeseries.R` to:

1. Read in data from `data/processed/`
2. Make plot
3. Save output in `figures/timeseries.png`

src/plot_timeseries.R

```
## plot_timeseries ##
# Make and save an ice cover timeseries plot

# input ------------------------------------------------------------------

# data file
infile = "data/processed/ice_cover.rda"

# plot file
outfile = "figures/timeseries.png"

# setup ------------------------------------------------------------------

# external libraries
library(ggplot2)

# process ----------------------------------------------------------------

# plot
plt = ggplot(df)+
  geom_path(aes(x=year, y=ice_cover))+
  labs(x="Year", y=expression(paste("Sea ice area [million"," km"^"2","]")))+
  theme_bw()

# save
ggsave(plt, filename = outfile, height = 3, width = 5, units = "in", dpi = 300)
```

figures/timeseries.png



Simple project orchestration with a master script

Create a master file to execute all the analysis steps in the correct order. This should:

1. Run `src/process_data.R`
2. Run `src/plot_timeseries.R`

master.R

```
## master ##
# Process and plot example ice cover timeseries

# process raw data
source("src/process_data.R")

# plot timeseries
source("src/plot_timeseries.R")
```

Play around

The project is totally reproducible from raw data! Now you can:

- Make changes to either the plotting or the processing script
- Delete anything in `data/processed` or `figures`

And simply run `master.R` to re-build the entire project!

Key concepts

- Never edit raw data!
- All processed data and figures should be reproducible from raw data
- Use a master script (or other means) to orchestrate data processing
- Take time to improve code readability (use comments, indent, consolidate inputs, etc.)

Possible next steps

- Use `Make` instead of a master script to orchestrate the project more efficiently
- Use `symlinks` to link to large datasets that are stored remotely
- Use functions for repeated tasks

BREAK

Questions?

How do you keep your projects organized?

Writing

Goal: Find and organize references and draft a research report

1. Find references
2. Organize and review references with `Zotero`
3. Write and cite document with `Word` / `LibreOffice`

You will need

Zotero (`www.zotero.org`)

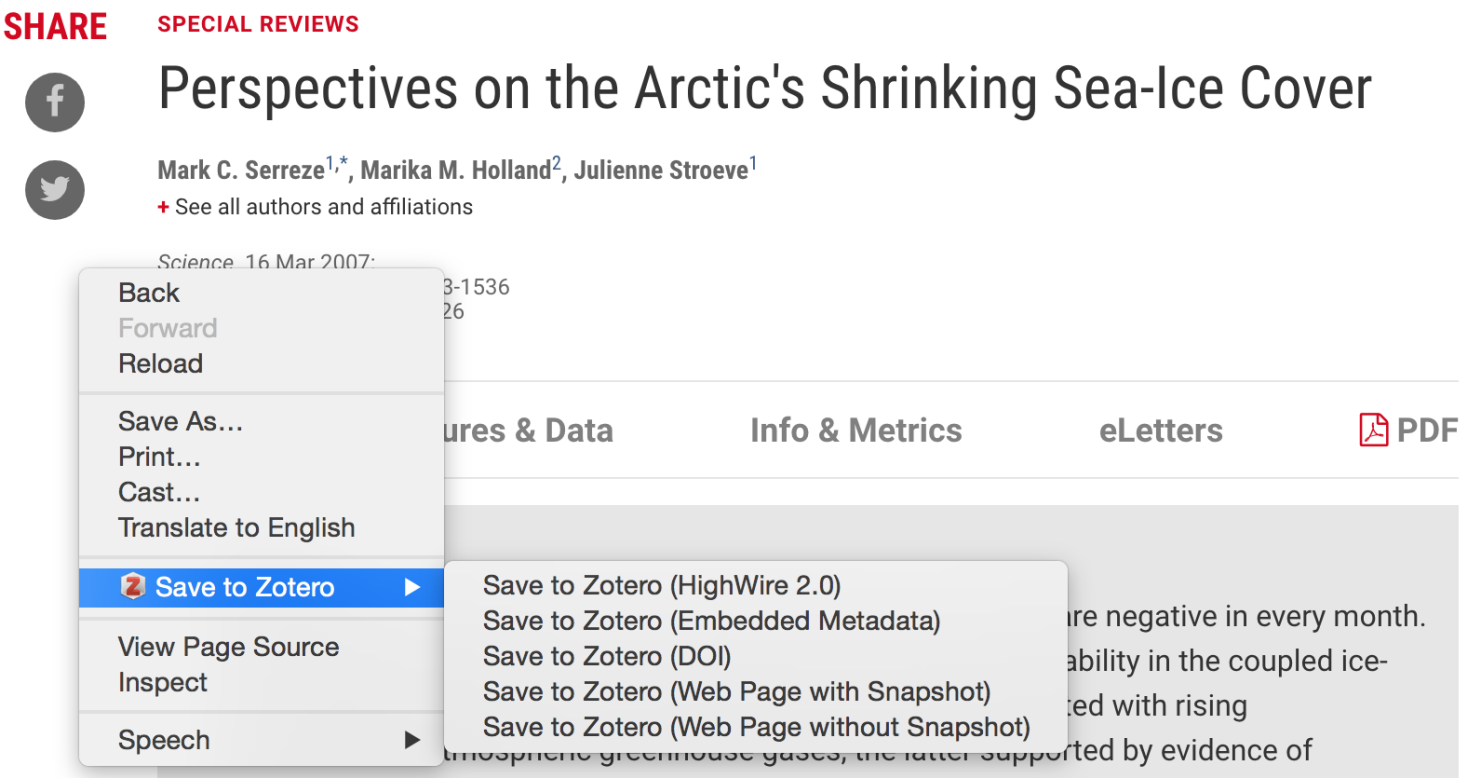
LibreOffice (`www.libreoffice.org`)
OR
Microsoft Office [paid] (`https://products.office.com/`)

Introducing Zotero

An open-source, one stop shop for acquiring, organizing, reviewing, and citing references

Acquiring

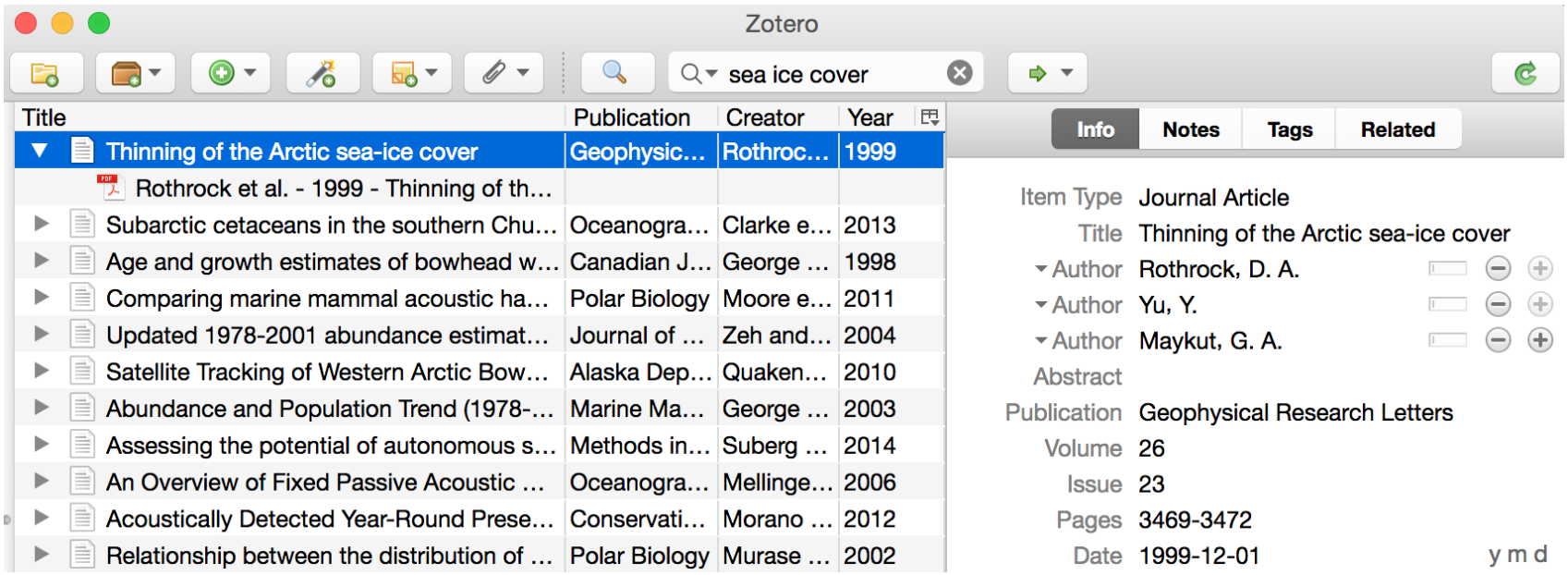
1. Install Zotero plugin for web browser
2. Find a reference (usually w/ Google Scholar)
3. Navigate to the journal page
4. Right click anywhere on the page and select `Save to Zotero (Embedded Metadata)`



Organizing

Open Zotero application and browse references. You can:

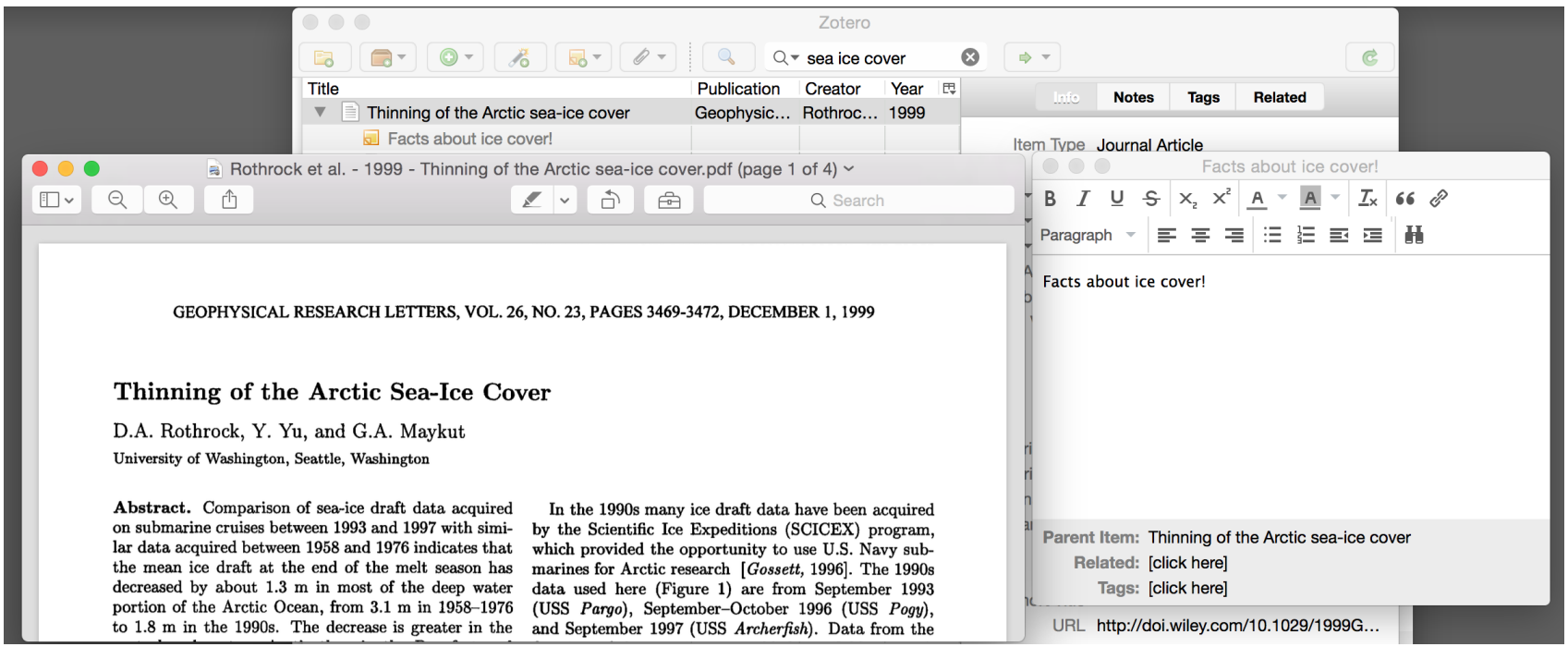
- Search / sort by author, year, journal, etc.
- Organize into project folders / collections / tags
- Add items from scratch



Reviewing

You can:

- View PDFs (with default viewer)
- Add notes / other files / etc
- Update / edit metadata
- Click and drag to share reference



Write and cite

In Word / Libre:

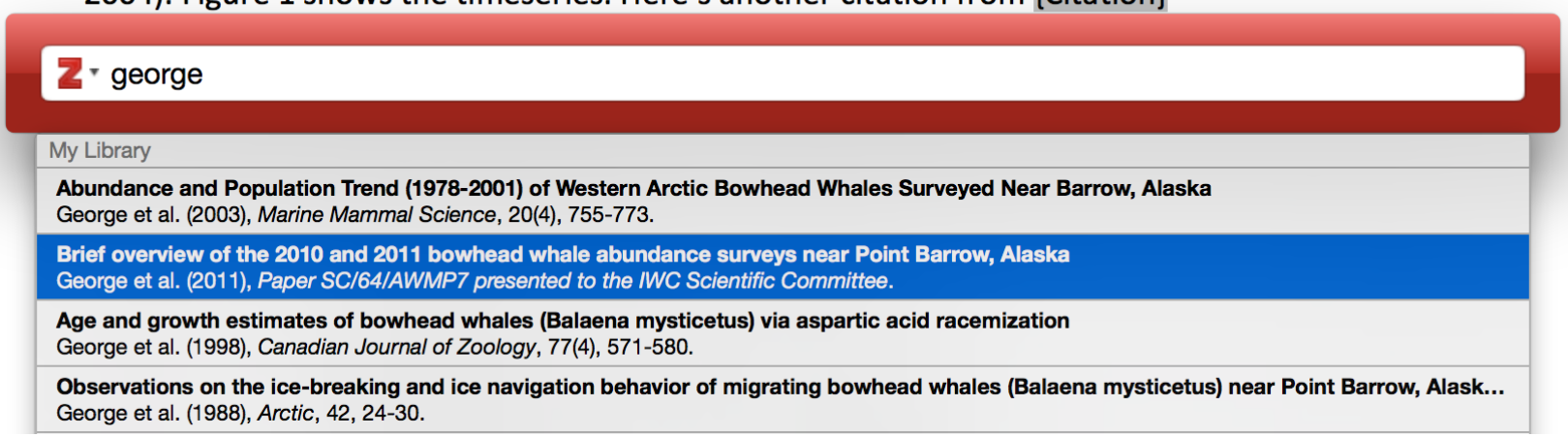
1. Install Zotero plugin
2. Click Zotero tab
3. Add references and bibliography with desired style

The past 30 years of sea ice cover in Canada

June 11, 2019

Hansen Johnson

Sea ice has been in decline for many years (Rothrock et al. 1999). Stroeve et al., (2008) suggest it declined sharply in 2007. This has been confirmed by modeling efforts (Saucier et al. 2003, 2004). Figure 1 shows the timeseries. Here's another citation from {Citation}



Key concepts

- Use Zotero to acquire, organize, review, and cite references

Possible next steps

- Use LaTeX for writing reports
- Use LaTeX `beamer` for making presentations
- Combine text, code and output into documents (html, pdf, word) and presentations (pdf, ppt, html) with `Rmarkdown`

BREAK

Questions?

What other tools do you rely on for writing?

Documentation

Goal: Document your work so that you can easily revisit, revert, and share

1. Add a `readme` file
2. Tracking changes with `git` and `Rstudio`
3. Remote backups and hosting with `GitHub`

You will need

git (`www.git-scm.com`)
GitHub account (`www.github.com`)

Readme files

What is a readme file?

- Usually simple text (`*.txt`) or markdown (`*.md`) file
- Includes any information required to implement or interpret the project workflow

Common things to include:

- Brief project background (goals, motivation etc.)
- Description of contents
- System requirements (code, software, etc.)
- Any caveats or known errors / bugs
- To do list
- Links for more information

readme.md

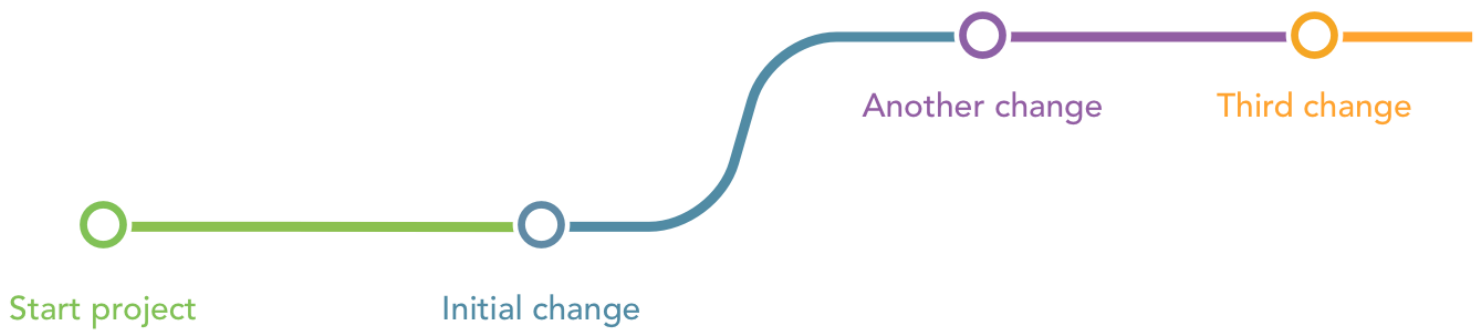
```
# README
Simple project to provide examples of helpful tools and
concepts for efficient and reproducible research

## Goal
Review recent trends in Canadian sea ice cover

## Dataset
Sea ice cover data were downloaded here:
https://www.canada.ca/en/environment-climate-change/services/environmental-indi

## Contents
‘data‘ - all data
  ‘processed‘ - cleaned and formatted data ready
  ‘raw‘ - only raw data *never touch*
‘src‘ - R code
‘wrk‘ - development sandbox
‘reports‘ - all presentations, reports, etc
‘figures‘ - all figures
‘master.R‘ - master script to reproduce full analysis
‘readme.md‘ - this file
```

What is `git`?

- `Git` is a hugely popular version control system (VCS)
- Open source software designed to help you track and document changes to projects
- Originally designed to be run on command line, but many more convenient interfaces now (e.g., `Rstudio`)

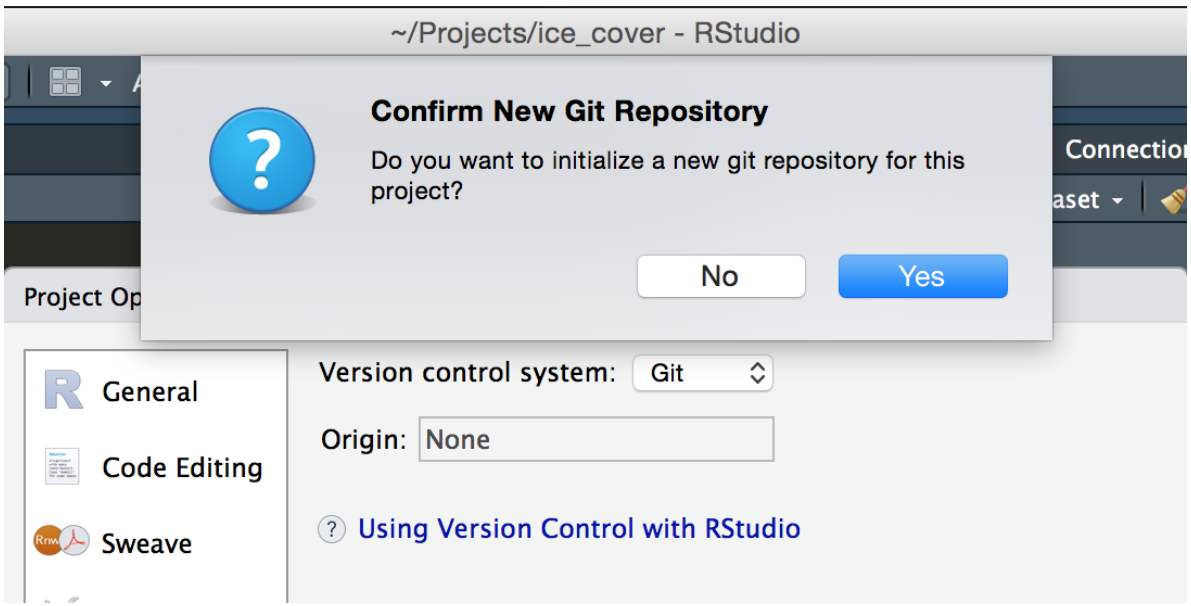
How does `git` work?

- `git` provides a convenient way to save a ‘snapshot’ of your project at a point in time
- Allows you to review project history and revert one or more files to a previous version
- You must add (‘commit’) changes to one or more files to the project timeline, and provide a description of your changes



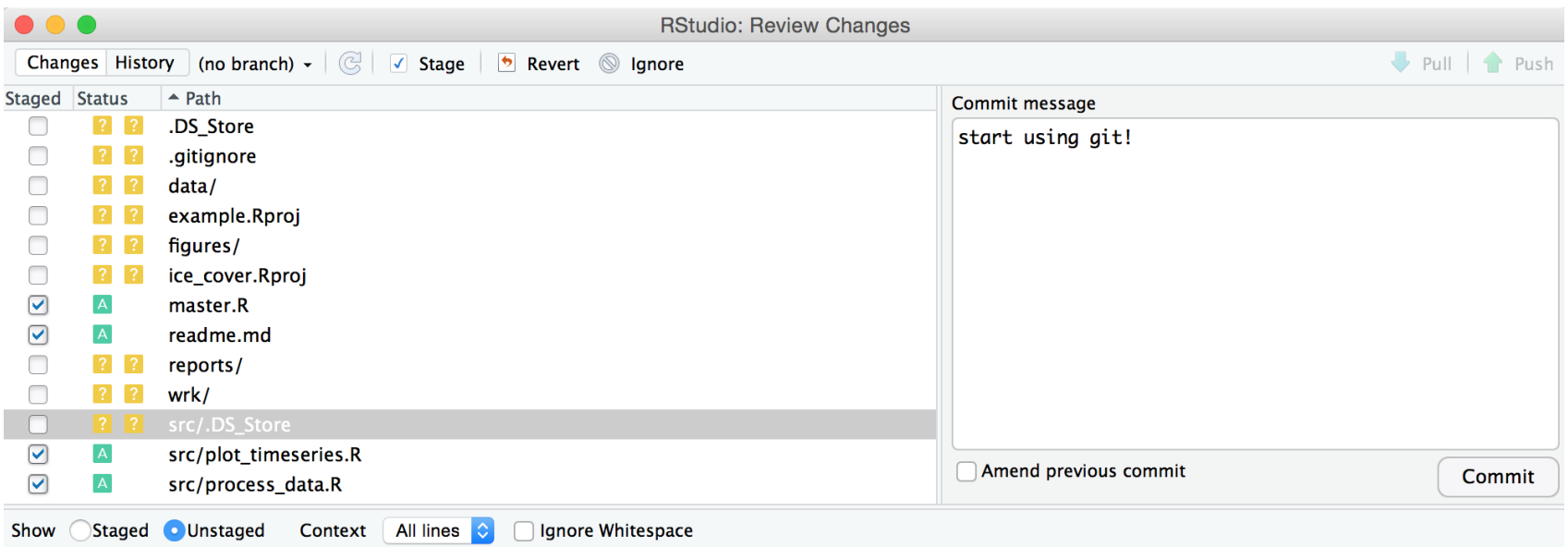
Using `git` in Rstudio

1. Navigate to `Tools -> Version Control -> Project Options -> Git/SVN` and switch `Version Control System` to `Git`
2. Restart Rstudio



Using `git` in Rstudio

1. Navigate to the `Git` tab and click `Commit`
2. Check the boxes next to all `*.R`, and `*.md` files
3. Write 'initial commit' in the box and click `Commit`

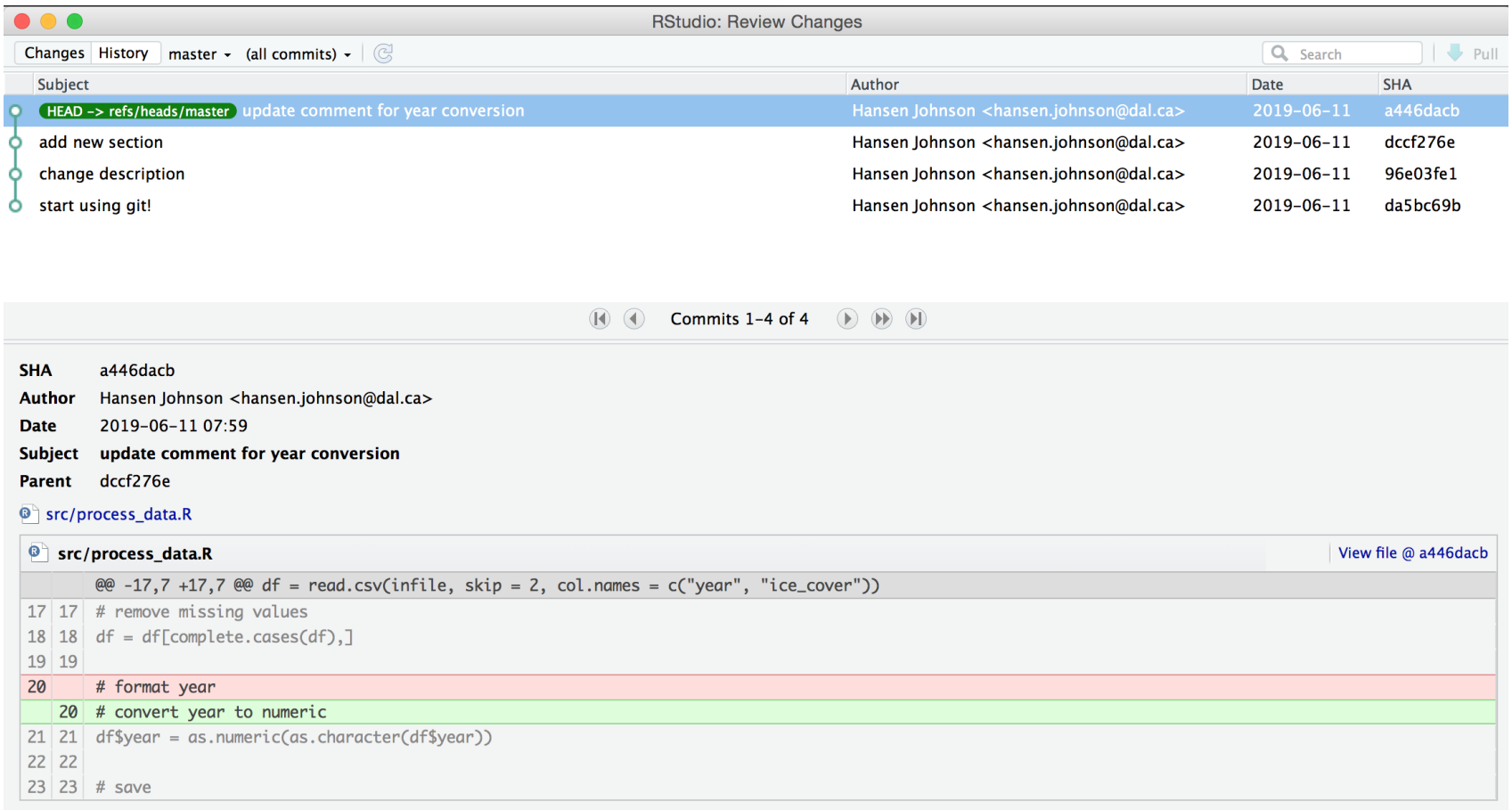


Tracking changes with `git`

- Edit various files and commit the changes
- Click on the `Git` tab, then on the clock icon to view your commit history (project timeline)
- You can view the full project history, or review changes to a particular file
- You can continue working in this self-contained way (i.e., not putting anything online) and track the entire history of your project

Avoid tracking any large datasets or private info. These can be ignored by listing them by name in a `.gitignore` file

Tracking changes with `git`



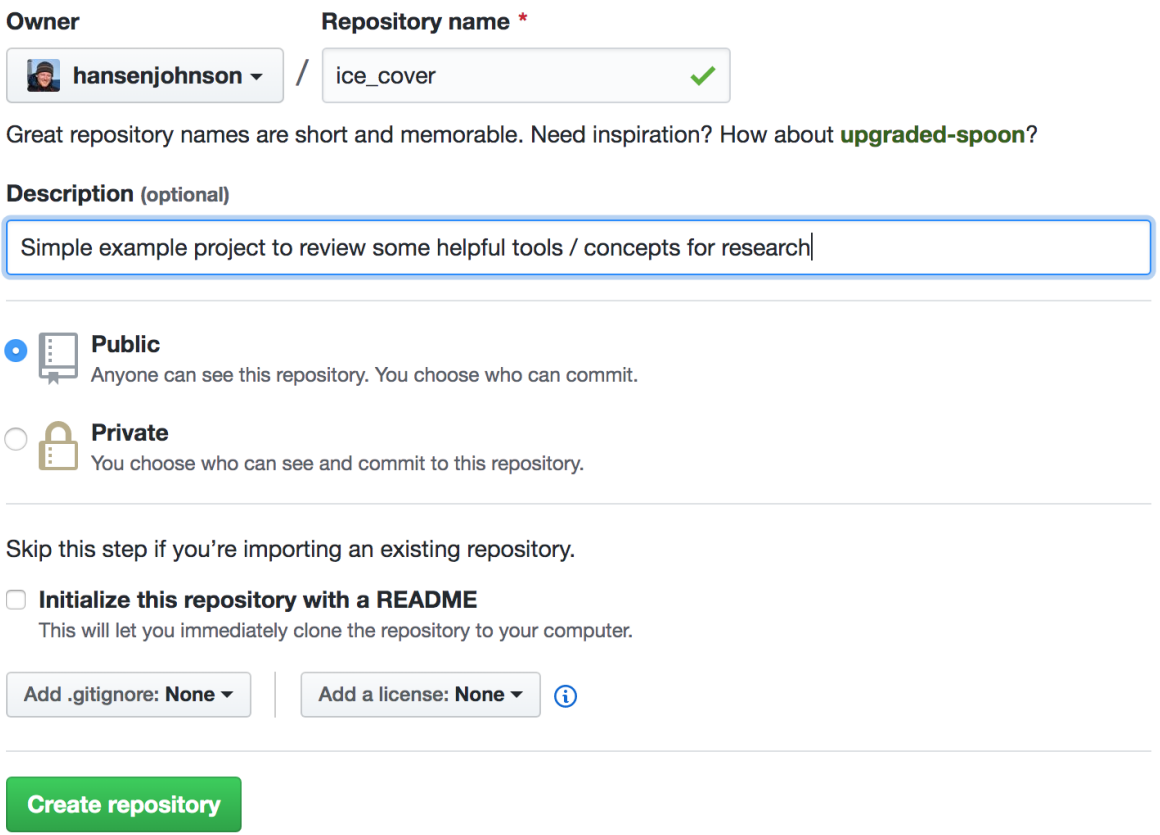
What is GitHub?

- `GitHub` is not `git`
- `GitHub` is a massive hosting service for `git` repositories
- Provides convenient tools for reviewing and collaborating on code (and free backups!)
- Unlimited free public and private* repositories

* Only with $\leq$ 3 collaborators (student accounts are unlimited)

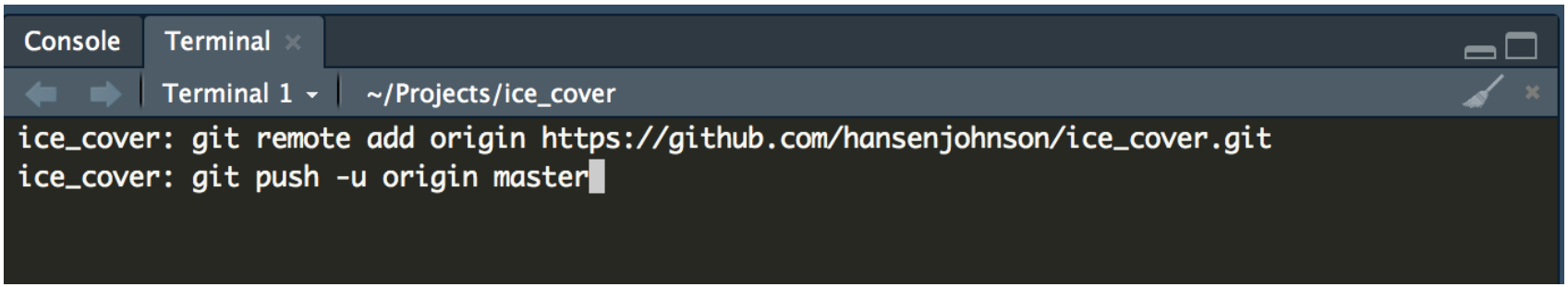
Creating and linking with GitHub repository

1. Go to GitHub user page
2. Create a new repository with the same name as our example project (e.g., `ice_cover`)
3. Choose to initialize without a readme



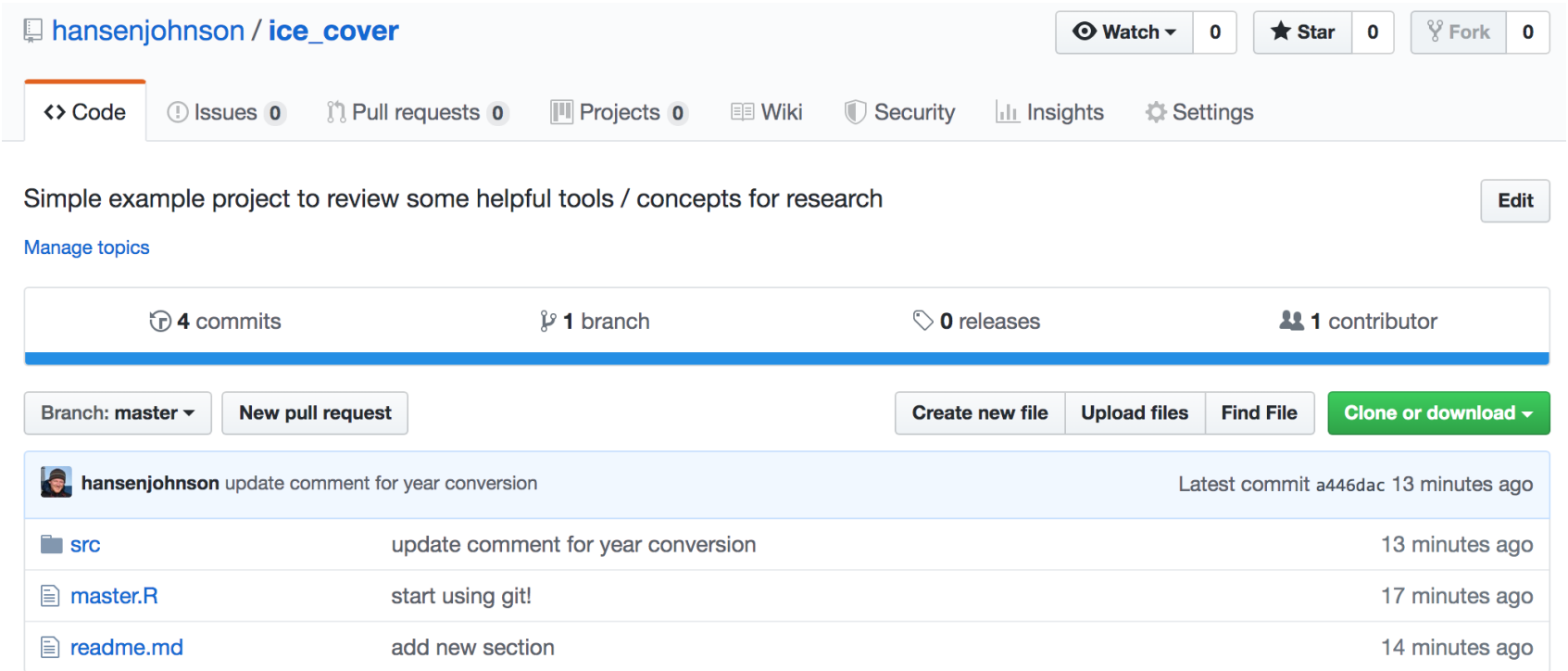
Creating and linking with GitHub repository

1. Copy code listed in "...or push an existing repository from the command line"
2. Move to Rstudio and open `Tools -> Terminal -> New Terminal`
3. Paste the lines into the terminal
4. Refresh your browser and check out your project online!



Using GitHub

1. Make commits on your computer
2. When ready, push commits to GitHub by clicking on `Push` arrow on the `git` tab in `Rstudio`
3. Check out new code online



Using GitHub

- Project contributors (collaborators, or you working on another computer) can `clone` the project onto their computer, commit changes, then `push` back to `GitHub`
- `git` and `GitHub` have many, many features for organization and collaboration including:
  - Branching
  - Merging / pull requests
  - Issue tracking
  - Website hosting

Check out fantastic `GitHub` documentation:

`https://guides.github.com`

Key concepts

1. Use readme files to describe your project, even if just to yourself
2. Use git in Rstudio to track changes
3. Use GitHub for backups, sharing, and collaboration

Possible next steps

- Dig deeper into `git` features (branching, pull requests, merging, etc)
- Use `git` and `GitHub` for collaboration
- Use Jekyll or Hugo to build project websites and host on `GitHub`

Questions?

Thanks to:
Christoph Renkl, Dalhousie Oceanography Student Association (DOSA), Methods in Ten Minutes (MTM), MEOPAR-WHaLE, and more!

Link to presentation:

`https://hansenjohnson.org/talk/2019_meopar_atm/`

Link to example project:

`https://github.com/hansenjohnson/ice_cover_example/`

Get in touch:

hansen.johnson@dal.ca