* Model development workflow
  + Workflow vs products (idea taken from openscience slides)
  + Workflow includes:
    - Editor you use to write code
    - Code you ran earlier
    - Name of home directory
  + Product includes:
    - Raw data
    - Packages used
    - Code someone needs to run to get your results
    - Figures and code used to make those figures
    - Functions
  + Main point: *Don’t want to hard code anything about workflow into your products*
* Openscience – many federal agencies have announced this year is the year of open science
  + “Open Science is the principle and practice of making research products and processes available to all, while respecting diverse cultures, maintaining security and privacy, and fostering collaborations, reproducibility, and equity”
  + Think of “open-ness” as a spectrum
    - Open with yourself
    - Open with co-workers (people on that project) – makes everyone feel included in the project and good for peer-review
      * We should get in the practice of peer-reviewing each other’s code
      * Allows people to jump from one aspect to another easier without having to spend a lot of time getting caught up or waiting for someone to send them all the files they need
    - Open with other SAP members (not directly working on project)
    - Open publically – eventually products will be this
  + Ok to share messy or incomplete code
    - Don’t wait to share until its perfect (push to github)
* Tools to use for openness
  + Google drive
  + Github repositories
    - Private
    - Public
  + Groups and trainings – openscapes and NMFS R User Group
* Why we should use an automated workflow
  + saves time
  + saves energy – not having to re-write code every time, reuse when you can
    - need to make code flexible and generic
  + makes onboarding easier – for the next person doing the assessment
  + for future you – helps you remember exactly what you did when you go back to the code a few months or even a couple of years later
  + makes things uniform
  + can put your energy towards developing the model and not towards the little details
  + Main point: *automate as much as possible!!!!*
    - Don’t waste your time doing the tedious tasks bc you have more important stuff to do than resizing figures
* What mistakes it can help avoid
  + Reduces the chance of making small mistakes that can have big impacts
  + If there are systemic mistakes, this can be an issue but once you find it and fix it, it’s easy to fix and re-run everything
* why SAP should go in this direction
  + anyone can download our repo and run the models themselves
  + made it a lot easier during the review to run alternate scenarios
  + can share the responsibility (this also means there isn’t one failpoint)
  + share knowledge within the group (and with others outside of the group as well)
* How we implemented it in American Samoa:
* Data – kept raw data in folder on google drive
  + Used code to determine if we had latest version on our computer and if not download and unzip newest version to replace old
  + Run data processing scripts to get model input (cleaned and processed data)
* Model inputs – kept spreadsheet with parameter values on google drive
  + Really leverage the googlesheets4 r package to connect directly to google drive through r
* Code – separate functions to do each component
  + Write SS files
  + Run SS
  + Run diagnostic tests
  + Run bootstrap
  + Run forecasting
  + Creating figures and tables
  + Use parallel processing to run multiple models
  + This method lets us quickly make a change, and re-run everything we need to generate the final output
* Quarto docs
  + Summary diagnostics report
  + Formatted figures and tables
* Github – keep all key files and summary reports so model files are available to anyone with access to repo
  + Tracking issues and tasks in github issues
  + Included some instructions in the readme on how to use the repo so when anyone goes back to it, they know where things are and what they need to run
* lessons we learned in our development
  + make sure you use packages that work with all platforms of R
  + want to make functions flexible enough to give you options but not worry about coding every single possibility
  + develop as you go, our structure changed from our initial ideas but as we worked we streamlined it
  + make code modular
* small things to implement in your code now to reduce friction later
  + more comments
  + read other people’s code (you can learn from their code, new functions, style, etc.)
  + make sure you are using relative paths and including all of the packages you used in that script
  + when naming scripts, give them informative names and try sequencing them
    - or make a note at the top if a different script needs to be run first
  + get in the habit of syncing code with repository regularly