sseep-manual

Gavin Fay

7/5/2022

Table of contents

SSEEP manual

This is the project manual for the SSEEP project.

Rationale: documenting workflow, etc. for project team members.

Intent: document, document, document. even if a little scattered for now.

1 Introduction: How we work

There are several shared resources for SSEEP.

- SSEEP website (public facing, project information)
- SSEEP GItHub (main github repo, where the website is served from)
- SSEEP GitHub Team (project team members, project boards, collection of SSEEP-related repositories)
- SSEEP GitHub Project Boards
- SSEEP Google Drive (shared folder(s), meeting notes, workshop materials, presentations)
- SSEEP manual (this book, largely for project team members, containing workflows, documentation, etc.)
- SSEEP folder on Fay lab storage server at SMAST (for data not on GitHub and for backups)
- SSEEP Gantt Chart documenting major deliverables, timeline, etc. (to come)

1.1 Communication

(how to communicate with project team members, and for what)

- GitHub issues: repo-specific, preferred method of communication in-team regarding project cards, tasks, etc.
- GitHub Team issues: As for repo-specific issues, but items related to overall project management
- Email: preferred method of communication with larger team and with steering committee.

2 SSEEP GitHub repositories

There are several repositories for SSEEP-related work. (add more as they come).

All should be associated with the SSEEP GitHub Team.

Preferred naming, use `sseep-[name]`

- sseep (main github repo, where the website is served from)
- sseep-manual (this repo)
- SimSurvey (fork of {SimSurvey})
- sseep-analysis (cleaning/tidying/prep of trawl survey data, statistical analyses of BTS data for SSEEP)

3 File organization

Recommended organization for files across SSEEP repositories.

(borrowing heavily from Wilson et al. 2017 "Good enough practices in scientific computing")

Use Projects (.Rproj). Keep all project files in their respective directories.

Do not duplicate data across repositories. If something is to be re-used across projects, consider adding to the main sseep folder. (perhaps consider a separate `sseep-data` repo?)

Be mindful of what is part of the github repository and what files are unique to your local machine. Use `.gitignore` files to prevent pushes of local files that do not need to be tracked.

Suggested directory structure: (don't need to rigidly adhere to this, but borrow the principle

[repo-name] - data (directory containing repo specific data files)

[repo-name] - docs (directory containing text files related to the project)

[repo-name] - R (directory containing R scripts)

[repo-name] - results (directory containing output files)

3.0.1 Local file locations

Recommend having all projects in the same root folder, so that it is easy to navigate among them via relative file paths if want to access something in a different repo.

e.g.

```
~/research/sseep/
~/research/sseep-data/~/research/sseep-manual/~/research/SimSurvey
```

then if say you are in `SimSurvey` and need to access something in `sseep-data`, might use something like:

```
datafile <- "../sseep-data/data/wind-map.csv"
```

4 File naming

4.1 R scripts

Recommended naming for R scripts across SSEEP repositories.

Many repos will have a `R` directory containing R scripts that perform analyses for the work in that repository.
It is helpful for viewers to be able to quickly understand how to interact with these.

File names should give an overview of the main purpose of the script.
Number scripts within folders according to the order of operations.
e.g.

```
01-clean-raw-data.R
02-tidy-cleaned-data.R
03-import-windarea-shapefile.R
```

At the start of each R script, write a section of comments that overviews the objectives of the particular code, what the dependencies are, and what the outcomes of running the code are (e.g. objects created, files written, etc.).

Consider creating documentation via a README or a wiki page that outlines the workflow for particular tasks.

Avoid performing too many steps within a specific script (give each script its own objective), to both aid future users of the code but also to facilitate modularity of workflow and script re-use.

References