BIOINFORMATIC WORKFLOW SOP

Guidelines for transparent, reproducible and accessible bioinformatic reporting for scientific research

Serrano Lab

Reproducible Research

Reproducibility from bench to code: B2C

- **All research should be reproducible from a technical standpoint:**
  - If two runs of the same code produce different results, these are not valid results.
- **Never edit files using a GUI:**
  - GUI edits (like in Excel) can introduce human error due to the lack of traceability.
  - If provided with Excel or csv files which require editing or transforming, write code to do this.
    1. You will know exactly what you have done, and so will other researchers
    2. The chances of you having to repeat the editing with revised data are high, so write the code once
- **Reproducible doesn't mean correct**
  - Reproducible code with errors will simply reproduce the errors. Make sure your code is analytically sound.

Note: Some of the guidelines here are specific for RStudio, please feel free to add the appropiate guideline if you work on a different IDE

Project Management

Reproducing the Data Analysis with `renv`, a tool for reproducible enviroments

All our published project use `renv` to promote accessible, reproducible, and easier-to-share scientific analysis.

Why Using `renv` is Important

`renv` is crucial for several reasons:

- **Reproducibility**: Ensures that the analysis will work as expected, regardless of future changes to R packages. The `renv.lock` file captures the exact package versions used, allowing anyone to reproduce the environment precisely.
- **Isolation**: Each project can have its own isolated R environment. This means you can use different versions of packages in different projects without conflicts.
- **Collaboration**: You can use `renv` to recreate your exact environment, avoiding the common "it works on my machine" problem.

Setting Up the R Environment in RStudio Using `renv`

This will help you reproduce the R environment we used in this project.

1. **Open the Project in RStudio**: Open RStudio and navigate to the R project directory where you saved this project. You can also do this by:
   - Opening the `.Rproj` file within the project directory, or
   - Using the `Session -> Set Working Directory -> Choose Directory...` option in RStudio to set your working directory to the project folder.
2. **Install `renv`** (if not already installed): Open R or RStudio and run the following command to install `renv`:

   ```
   install.packages("renv")
   ```

3. **Restore the R Environment**:
   - The project directory should contain an `renv.lock` file, which `renv` uses to recreate the package environment. To restore the environment, open the R console in RStudio and run:

     ```
     renv::restore()
     ```

   - This command will:
     - Install all the R packages required for the project (say yes).
     - Install the exact versions of the packages as specified in the `renv.lock` file.
   - **Note**: This might take a few minutes, especially if the project uses many packages.

Work in self-contained projects

- All the files required to run your analysis must be contained within the same folder.
- Avoid reading a file from a different location in your computer or on the web.
- I strongly recommend the following folder structure

```
project/
- README.md    # Project description written in Markdown
- R/           # Where the R scripts live
- input/       # Data files
- output/      # Results: plots, tables..
```

Avoid the `setw()` nightmare

- `setwd()` sets the working directory of your R session, meaning all file paths are based on that location. Hardcoding file paths can create issues when the project is moved.
- This means that you'll have to update `setw()` *every* time (if) you move your folder
- This is unproductive and a bad working experience in collaborative projects
- Use R Studio Project to improve portability and reproducibility
- I recommend Project-oriented workflow by Jenny Bryan

Coding Practices

Write code for people, not computers

- **Variable names must be easy to interpret:**
  - e.g. a list of genomes should be called `genomeList` not `g`
  - Don't call objects `results` and `results2`. Include information in the name.
  - Typing extra characters is easy using auto-complete
- **Comment your code extensively:**
  - You will have to revisit most analyses in the future. Write comments to remind yourself of what you've done.
  - Collaborators with less coding experience than you will often need to read your code. Communication is much easier if they can understand what you've actually done.
- **Write intelligent comments**
  - Comments such as `# Load all data` could be better written as `# The data is spread across 183 csv files in the directory /myData. Load all into a single object`
  - Too much information in a comment never broke a program
- **Be stylistically consistent**
  - This just makes your code easier to follow for everyone

Always Use Version Control

- **Version control is not just for package or software development.**
  - Use version control for *every analysis*!
  - Only use private repositories if required
- **Be consistent across an organisation**
  - *Git* is the version control methodology used by the Bioinformatics Hub
- **Make incremental changes**
  - Recompile or rerun regularly to minimize coding errors.
  - *Stage after a successful run*: After you've confirmed that a specific change works save that change in version control (such as Git).
- **Write informative commit messages**
  - This makes it easier to find old code
- **Use a remote code repository**
  - Push your code daily. This provides an up-to-date copy in the case of disk failure
  - This also enables collaborators to contribute

Error Handling

- **Include Checks:**
  - These can include simple things like the presence of a file with the correct name, to verification of the output of a function. Another useful (and necessary) check is data structure and summary checks to ensure you are working with the intended data.
- **Error Handling:**
  - You will make mistakes! Often.
  - Write code to fail quickly and clearly on errors
  - Write clear error messages

Data Management

Data Integrity

- **Keep a copy of all source data**
  - Utilize storage which is backed up. Please review our Data BackUp guidelines in the On Boarding documentation
- **Make the source data read-only**
  - This prevent accidental deletion or editing as a result of poorly written, or copied scripts can easily do this
  - This setting can easily be temporarily changed as required

Metadata and Study Design are important

- **Always begin an analysis by describing the data, motivation and the intended path to the results.**
- Linking the publication that explains the experimental design is acceptable
- **Where possible, make raw data available**
  - At the very least, key collaborators should know where to find it.

Research Collaborations

- **Always confirm with the lead researcher (Lead Student or PI) before collaborating** (or adding new collaborators)
- **Agree on the key positions of authors in advance**
  - Any significant contribution should be rewarded with co-authorship

Record Keeping

- A lab book is a **requirement** for both in-lab and in-silico experiments to ensure completeness in research documentation.
- **Never use a GUI**
  - Ever
  - But, if for some kind *Meteor hits the Earth* situation you had to edit your table in Excel or similar (guilty of charge), please always keep the raw data as exported/generated in a dedicated folder. And please, for the love of code, comment appropriately in your markdowm/quarto so your future self (and the rest of human kind) can understand what's going on. Something as simple as `fsom_01` to `fsom01` can cost you hours of debugging (Learned this the hard way).
- **Analytic code must be version controlled**
  - Version control is the practice of tracking and managing changes to code over time. GitHub is the preferred platform for version control.
  - This provides digitally verified copies of the analysis with dates recorded
  - Old approaches which need to be reinstated can be accessed
- **Keep a lab book**
  - Despite working mainly *in silico*, records of new approaches should be written and signed off in a lab book
  - Keep full details of thought processes and ideas for new analytic approaches
  - These are essential to legal proceedings (e.g. prosecuting a patent)

Lab Preferred Approaches

Statistical And General Data Analysis

- This is to be perfomed using scripting languages such as `R` or `Python`
- Exporting data and loading in Prism is acceptable (not preferred). Also, let's be honest, if you've read so far down this document is because you are a so motivated that you can definitely perform all the statistical analysis within your code. You've got this!
- Excel is not acceptable. Please review our Training Folder: `Statistical Analysis Resources` and `Area Analysis` where you'll find templates to perform your analysis in R, Jamovi and Prism

Version Control

- Git is the only acceptable version control system
- https://github.com is the preferred repository
- Analysis stored in a private repository must have a **minimum of two users** with read access or higher

References and Further Reading

- A how-to guide for code-sharing in biology
- Productive R Worflow
- Best Practices for Bioinformatics Research
- Coding Best Practices