Data Management Plan

Washington Soil Health Initiative: State of the Soils Assessment

Jadey Ryan
Dani Gelardi

2024-01-26

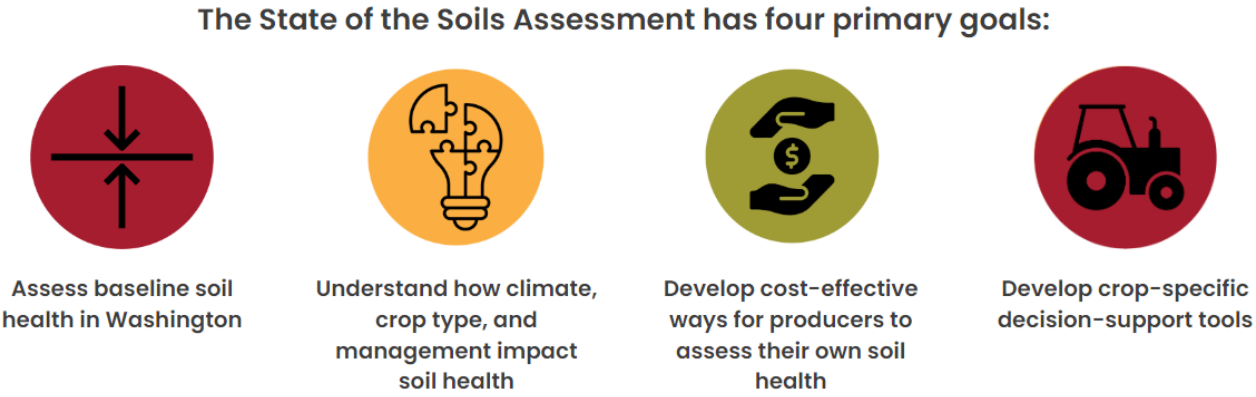
Table of contents

Overview

View this data management plan as an online book at https://wa-department-of-agriculture.github.io/washi-dmp/.

The Washington Soil Health Initiative (WaSHI) is a partnership between the Washington State Department of Agriculture (WSDA), Washington State University (WSU), and the State Conservation Commission. WaSHI establishes a coordinated approach to healthy soil in Washington.

To date, nearly 1,000 soil samples and management surveys across 50 different cropping systems have been collected as a part of the State of the Soils Assessment (SOS). WSDA and WSU lead this project with support from staff, students, conservation districts, and agricultural professionals throughout Washington.



Chapter outline

This Data Management Plan (DMP) is a living document to be continually reviewed and improved based on lessons learned, new information, and collaborator feedback.

Chapter 1 describes what data management is, how it is crucial to achieve our data-driven goals, and how our data move through the data life cycle.

Chapter 2 describes the various data formats we collect and manage. ISO standards are also described for date and geospatial data.

Chapter 3 describes naming conventions, best practices, and examples for how we name folders and files.

Chapter 4 describes how we organize our folders into a hierarchical structure.

Chapter 5 describes where we store data, what our backup policies are, how we protect our raw data, and how we use version control.

Chapter 6 describes how we record each element of the data life cycle with project-level, dataset-level, and variable-level documents such as standard operating procedures, readme files, data dictionaries, etc.

Chapter 7 describes how data are generated, processed, and moved from start to finish. Processes and tasks are grouped by pre, during, and post field season.

Chapter 8 describes how we protect producer privacy; how our data fits into WaTech data categories; requirements and processes for maintaining confidentiality; and our data share agreement, public access policies, and our preferred acknowledgements.

Chapter 9 describes our recommended project structures, code-specific naming conventions, script structures, and code style.

Roles and responsibilities

All WaSHI personnel who will be interacting with SOS data must familiarize themselves with the contents of this document. If all collaborators are not consistently implementing this DMP, then the benefits of effective data management are lost.

The WSDA Data Scientist, supported by the Co- Principal Investigators (CoPIs), is responsible for providing guidance to WaSHI staff working with SOS data and ensuring the implementation of this DMP. The Data Scientist is also responsible for reviewing and updating this document annually, and as needed. Upon updates, the Data Scientist will distribute this document to WaSHI staff and commit the source code to the GitHub repository.

Current roles as of January 2024

Role	Affiliation	Name	Title
CoPI	WSDA	Dani Gelardi	Senior Soil Scientist
CoPI	WSU	Deirdre Griffin LaHue	Assistant Professor
Data Manager	WSDA	Jadey Ryan	Data Scientist
Data Stewards	WaSHI personnel		

Staff turnover

When staff leave our group, they take their skills, institutional knowledge, and personal understanding of their file management with them. Proper offboarding is essential to ensure knowledge isn't lost, time isn't wasted trying to recreate workflows, and projects keep moving.

Before the employee leaves, the Senior Soil Scientist and Data Scientist ensure that:

- Folders and files are moved from the employee's personal drive to the shared drive. They are named and organized according to Chapter 3.
- Workflows and specific processes the employee was responsible for are well documented.
- Permissions and ownerships are transferred to the appropriate remaining staff.
  - GitHub WSDA organization
  - ArcGIS data products and online groups
  - Database credentials
  - Box.com folders

More resources and offboarding checklists from Harvard Research Data Management can be found in our data-management shared drive.

Acknowledgements

This DMP was adapted from the R.J. Cook Agronomy Farm Long-Term Agroecological Research Site DMP (Carlson 2021), U.S. Fish and Wildlife Service data management life cycle (U.S. Fish & Wildlife Service 2023), Harvard Medical School Longwood Research Data Management DMP guidelines (Harvard Medical School 2023), and the Data Management in Large-Scale Education Research book (Lewis 2023).

1. What is data management?

Effective data management involves properly documenting, storing, and sharing our data and the information we derive from the data. If the data aren't usable by researchers, policymakers, or growers, then all the time, energy, and effort spent collecting and analyzing the samples may be wasted.

The guidelines detailed in this DMP help us achieve our data-driven goals, while also optimizing the value of the data by supporting information sharing and innovation. Our data management policies attempt to follow **FAIR** (**F**indable, **A**ccessible, **I**nteroperable, **R**eusable) principles while also maintaining data privacy (Wilkinson et al. 2016).

1.1 Data life cycle

The U.S. Fish and Wildlife Service developed a great graphic to explain the elements of the data life cycle and emphasize the importance of data quality at every step (U.S. Fish & Wildlife Service 2023). Each step within the data life cycle requires careful intention to ensure transparency, quality, and integrity.

Our adaptation of this data life cycle is outlined below.

Plan

Each sampling year presents an opportunity to consider what worked and what could be improved from the previous year. Planning involves making decisions about data acquisition, management, and quality control. For example, each year we provide a spreadsheet template with our requested column headers to Soiltest lab to ensure the measurements are reported with correct units and in the format we use. Special projects that deviate from our standard operating procedures require additional planning.

Acquire

We acquire data by collecting and analyzing new samples, deriving new insights from existing data, or accepting datasets from collaborators.

Maintain

Maintenance involves processing data for aggregation, analyses, and reporting. We create metadata that facilitates interpretation of the data. We

also store at least one copy of our data in a non-proprietary format that is accessible to our collaborators and future selves.

Access

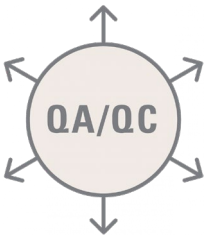
Access refers to data storage, publication, and security. Raw and processed data with accompanying metadata should be stored, backed up, and available for information sharing with our partners. With PI approval, anonymized and aggregated data that does not compromise growers' personally identifiable information (PII) should be made publicly available in a data repository or data product/decision-support tool.

Evaluate

We evaluate data while processing and analyzing it to maximize accuracy and productivity, while minimizing costs associated with errors or tedious data cleaning labor. Evaluation workflows should be efficient, well-documented, and reproducible. Our evaluated data help us better understand how environmental factors and management decisions impact soil health.

Archive

Properly archiving our results supports the long-term storage and usefulness of our data. While similar to the Access element of the life cycle, archiving focuses on preserving data for long-term/historical retention that aren't needed for immediate access. For example, we archive each year's raw data for long-term storage and set those files to `Read-Only`.



Quality Assurance / Quality Control (QA/QC)

Data quality management prevents data defects that hinder our ability to apply data towards our science-based conservation efforts. Defects include incorrectly entered data, invalid data, and missing or lost data. QA/QC processes should be incorporated in every element of the data life cycle.

The following chapters describe our internal processes and standards to follow throughout each step in the data life cycle.

2. Formats & standards

2.1 Data formats

Data generated from or integrated into WaSHI can be non-digital or digital.

Non-digital data

Non-digital data, such as field forms, management surveys, and chain of custody forms, are manually recorded on paper forms. Paper forms must be transcribed or converted to digital file formats and then stored in the WaSHI filing cabinet in the Natural Resources Building in Olympia.

Digital data

Digital data include tabular, spatial, and binary data, such as lab results, sample locations, and field photos. Non-conventional data also include code, algorithms, tools, and workflows.

Tabular data include comma separated values (csv), tab separated values (tsv), Microsoft Excel open XML spreadsheet (xlsx), and portable document format (pdf).

Spatial data include file geodatabases (gdb), vector shapefiles (zipped folder containing multiple file extensions), keyhole markup language (kml or kmz). Tabular data may also contain spatial data such as longitude and latitude.

Binary data include photos (jpeg, png, gif, tiff), videos (mp4), code (R, py, js), and object-oriented data files (RDS, Rdata, parquet, arrow).

Proprietary data formats include Microsoft Excel, Word, and Powerpoint files (xlsx, docx, pptx). RDS and RData files are examples of application-specific data formats that can only be opened using the R programming language or RStudio IDE. These types of files should be saved in conjunction with a copy of the data in a non-proprietary and open-standard format, such as csv, to maintain accessibility for those who do not have Microsoft Office or do not use R.

Written documents and presentations are in formats including Microsoft Word and Powerpoint (docx and pptx), hypertext markup language (HTML), and pdf.

Notebooks combine text with executable code to generate written documents and presentations in docx, pptx, html, or pdf formats. These notebooks are stored in formats depending on the programming language: a few examples include R markdown (rmd), Quarto (qmd), and Jupyter notebook (ipynb).

The list below is not exhaustive and will continue to grow as additional useful data sources are discovered.

Type	Source	Formats
Lab results	Provided by the lab analyzing the soil sample, principal investigator of a study, or grower	csv, xlsx, pdf, xml, json, RDS, RData
Management surveys	Collected through interviews with grower	csv, xlsx, RDS, RData, paper form (to be digitized)
Field forms	Completed in the field during/immediately after sampling	pdf, paper form (to be digitized), csv, xlsx
Sample locations	Identified prior to sampling and may be edited during sampling using ArcGIS Online, Collector, Field Maps or Google Maps	ArcGIS feature layer, shp, kmz, csv, xlsx
Chain of custody forms	Completed prior to shipping or dropping off samples	pdf, paper form (to be digitized)
Climate data	OSU PRISM, NOAA, Esri Living Atlas	csv, shp, netCDF, tiff, gdb
Soil data	NRCS Web Soil Survey, NRCS WA gSSURGO	gdb, accdb
Strata classification	Provided by Soil Health Institute in 2021 as a lyr file	lyr
Images	Logos, icons, photos taken in the field	jpeg, png, gif, tiff, svg
Videos	Recordings of meetings, training videos	mp4
Documents	Reports, manuscripts, SOP, QAPP, factsheets, brochures	docx, txt, html, pdf
Presentations	Powerpoints, slide decks	pptx, html, pdf
Code	Scripts for wrangling, processing, analyzing data; markdown for producing documents and presentations; style sheets for html outputs	R, py, ipynb, js, yml, rmd, qmd, css, scss

2.2 Data standards

Date will be expressed as `YYYY-MM-DD` according to the ISO 8601 standard.

Date with time will be expressed as YYYY-MM-DD**T**HH:MM:SS**Z**.

- **T** separates date from time. The **Z** indicates the date-time is using the Universal Time Coordinated (UTC) with no offset.
- Pacific Standard Time (PST) has a UTC-8:00 offset and Pacific Daylight Time (PDT) has a UTC-7:00 offset and would be expressed as YYYY-MM-DD**T**HH:MM:SS-8:00. The **Z** has been replaced with the offset.

- Example: `2023-11-28T14:55:56-08:00`.

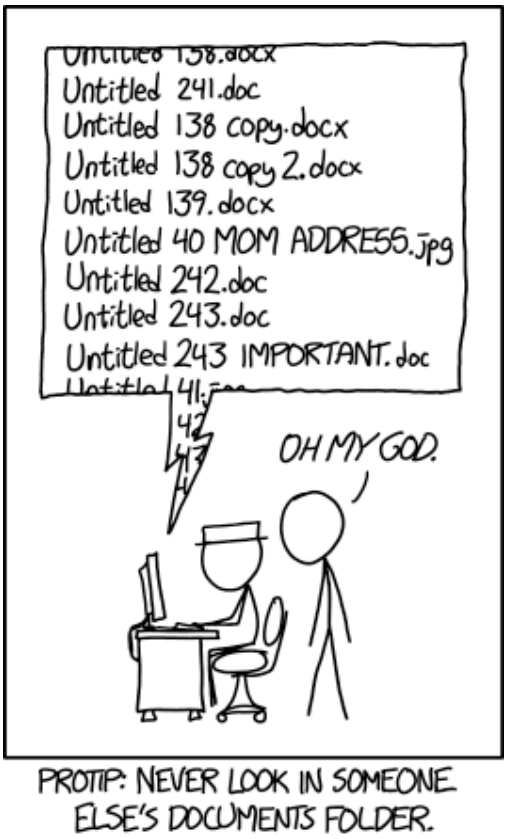
"ISO 8601" from Randall Munroe's xkcd

Geospatial data will be accompanied by metadata that abides by the ISO 19115 standard by following Esri's documentation when working in ArcGIS Pro. Metadata contains information about the identification, the extent, the quality, the spatial and temporal schema, spatial reference, and distribution of digital geographic data.

Code will follow the style guide in Chapter 9.

3. Naming conventions

When naming folders and files, we want consistent and clear names that are findable and understandable by both humans and computers. From only a file name, we should immediately know what the file contains, and which file is the most recent version.



"Documents" from Randall Munroe's xkcd

3.1 Why are conventions important?

- Improves consistency and predictability, making it easier to browse folders and know what the file/folder contains.
- Easily sort and retrieve files by date, conservation district, or another theme.
- Facilitates collaboration so all team members can find the information they need.
- Standardizes file paths and URLs for efficient programming and website hosting.
  - URLs and programming languages are case-sensitive. `WaSHI-data.csv` and `washi-data.csv` are completely different files.

- URLs cannot have spaces in them. They must be escaped with this character entity `%20`. For example, `wasoilhealth.org/producer spotlights` would need to be `wasoilhealth.org/producer%20spotlights`.

For more web-specific naming conventions, see this Learn the Web webpage.

3.2 Best practices

Some files and folders in our shared drive do not follow these best practices or naming conventions. We are learning and improving as we go.

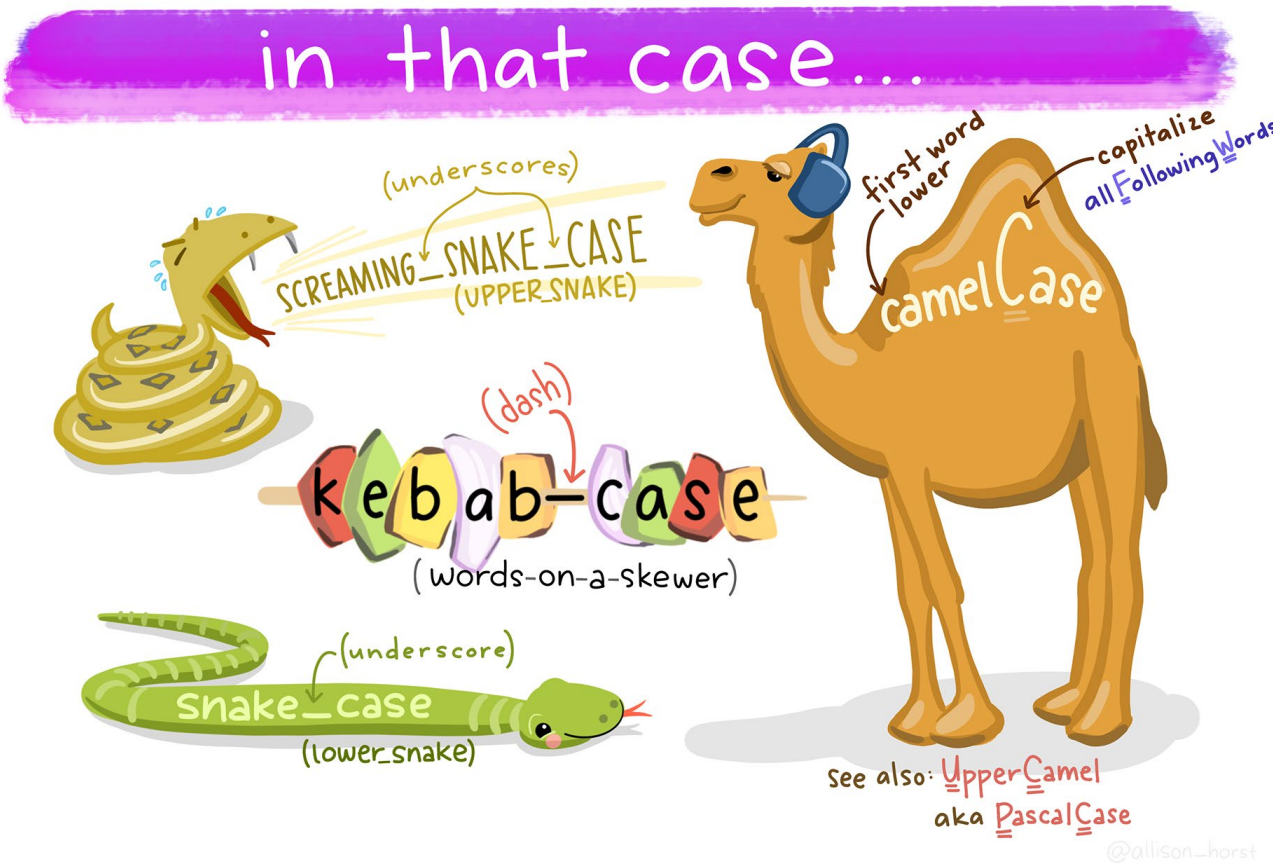
These are guidelines and naming things is hard, so try your best. If you're not sure how to name some files or you're adding a bunch of files that came from somewhere else, Jadey can help you organize and/or bulk rename them.

See Section 3.3 for a table of examples of folder and file names following these best practices.

Meaningful name casing

Different conventions work better for different purposes (folders and files versus programming objects).

- **kebab-case**: all lowercase with hyphens separating words. Use for all **folders** and **files**.
- **snake_case**: all lowercase with underscores separating words. Only use for **column names** in spreadsheets and **code**, such as functions and variables in R. See Section 9.2.3.1 for example R errors when including hyphens in object names.



Cartoon representations of common cases in coding. A snake screams "SCREAMING_SNAKE_CASE" into the face of a camel (wearing ear muffs) with "camelCase" written along its back. Vegetables on a skewer spell out "kebab-case" (words on a skewer). A mellow, happy looking snake has text "snake_case" along it. Artwork by @allison_horst.

Delimiters convey meaning

Deliberately use underscores and hyphens so we can easily understand the contents and programmatically parse file and folder names.

- Use underscores to delineate metadata elements (i.e. date from name from version `date_name_version`).
- Use hyphens to separate parts of one metadata element (i.e. date `YYYY-MM-DD` or name `wsda-washi-presentation`).

No spaces or special characters

Avoid spaces and special characters (only use underscores and hyphens). Characters like `/` `()` `!` `?` `%` `+` `"` `'` have special meaning to computers and can break file paths and URLs.

Character length matters

Computers are unable to read file paths and file names that surpass a certain character length. Be concise AND descriptive. Omit prepositions and articles when possible. Abbreviate long words. The path limit on Windows is 260 characters.

'Back to front' date

Remember to express date 'back to front' like `YYYY-MM-DD` according to the ISO 8601 standard. Left pad single-digit months and days with a zero. This maintains the chronological order of records when they are sorted alphanumerically.

✅ **Do this**	❌ **Don't do this**
2020-05-28_agenda.pdf	2-14-2023_Agenda.pdf
2023-01-01_agenda.pdf	2023-Jan-1_Agenda.pdf
2023-02-14_agenda.pdf	Dec052020_Agenda.pdf
2023-12-05_agenda.pdf	May_28_2020_Agenda.pdf

Group & sort files by name

Consider how folders and files should be grouped and sorted. Put that piece of metadata in the beginning of the file name. By putting the conservation district name in the front, we can sort by district. Or, we can sort by date by putting the date before the district name.

Sort by district	**Sort by date**
cowlitz_coc_2023-05-01.pdf	2023-05-01_cowlitz_coc.pdf
cowlitz_coc_2023-05-23.pdf	2023-05-01_cowlitz_tracking.pdf
cowlitz_tracking_2023-05-01.pdf	2023-05-09_ferry-cd_tracking.pdf
ferry-cd_coc_2023-05-10.pdf	2023-05-10_ferry-cd_coc.pdf
ferry-cd_coc_2023-05-17.pdf	2023-05-17_ferry-cd_coc.pdf
ferry-cd_coc_2023-06-06.pdf	2023-05-23_cowlitz_coc.pdf
ferry-cd_tracking_2023-05-09.pdf	2023-06-06_ferry-cd_coc.pdf

Version numbers

Including the date in the file name is one way to version a document. Alternatively, or in addition to, we can add a version number to the end of the file name. Consider how many possible versions there could be. If there may be more than 10, use leading zeros to left-pad single digit numbers so the file name always has the same length. v1 through v15 will not sort the same way as v01 through v15.

✅ Do this	❌ Don't do this
sop_v01.pdf	SOP_v1.pdf
sop_v02.pdf	SOP_v10.pdf
… [v03 - v09]	SOP_v11.pdf
sop_v10.pdf	SOP_v2.pdf
sop_v11.pdf	… [v3 - v9]

Collaboration

Add your initials to the end of the file name when "saving as" a file that multiple people are working on (i.e., `2023_sop-soil-health-monitoring_lm-jr.docx`). This ensures a version is kept as a backup. Alternatively, use Track Changes if working in a MS Word document.

Literature and references

When saving journal articles, user guides, and other reference materials, use the convention `author_year_abbreviated-title` where each authors' last name is separated with a hyphen, the title is abbreviated, and hyphens separate each word of the title. Remember to use underscores to separate different metadata.

✅ Do this	❌ Don't do this
lal_2004_soil-c-to-mitigate-cc.pdf	lal-2004-soil-c-to-mitigate-cc.pdf
clark-et-al_2020_pmn-sampling	clark-et-al_2020_pmn-sampling

Column names and code

Naming conventions for column names in spreadsheets and objects in code differ from folders and files. The hyphens in kebab-case cause errors in R and SQL code. Additionally, hyphens, spaces, and other special characters are invalid for ArcGIS table and field names.

Use **snake_case** for column names in spreadsheets and all **code objects** (R vectors, lists, dataframes, and functions).

In variable or parameter names, we include the measurement with the unit for clarity and self-documentation when sharing our data. This prevents unit confusion and reduces the risk of misinterpreting or inappropriately using the data.

Remember to not use special characters. Instead of `toc_%`, use `toc_percent`.

> **Note**
>
> The code naming convention here applies primarily to R. Python and JavaScript have different naming standards.

See Chapter 9 for more details in the code style guide.

3.3 Naming examples

	Naming convention	**Examples**
Folders	kebab-case	2024_sampling data-management
Files	kebab-case	2023-11-15_survey-perennial.xlsx records-management_v01.docx pend-oreille-cd_coc_2023-06-05.pdf geisseler-et-al_2019_ace-protein.pdf washi-logo-color.png washi-dmp.Rproj 01_load-metadata.R 2024_producer-report.qmd
Column Names & Code	snake_case	sample_id clay_percent pmn_lb_ac primary_color crop_summary assign_quality_codes()

4. Organization

We organize our folders into a hierarchical structure to clearly delineate segments of our projects, improve searchability, and ensure reproducibility across years.

4.1 Folder structure

We strive for a balance between a deep and shallow structure. If too shallow, there are too many files in one folder and they are hard to sort through. If too deep, we have to click too many times to get to a file and specific files can be difficult to find.

`Y:/NRAS/soil-health-initiative` is the parent folder for all WaSHI content.

Within the `state-of-the-soils` subfolder, we use a combination of **date-** (each year has its own subfolder) and **categorical-** based (dataset and documentation that span across years) folder structures.

```
Y:/NRAS/soil-health-initiative/state-of-the-soils/
├── _complete-dataset
├── 2019_scbg
├── 2021_sampling
├── 2022_partnerships-in-soil-health
├── 2023_sampling
├── 2024_sampling
├── data-management
├── data-sharing
├── data-sources
├── maps
├── projects
├── qapp
├── sop
├── training-videos
├── equipment-inventory.xlsx
├── archived-sample-inventory.xlsx
└── sos-impacts.xlsx
```

Within each year subfolder, we have sub-subfolders for planning, forms, data, and processes. This structure helps maintain a reproducible workflow year after year. See the `2023_sampling` folder tree for an example:

```
Y:/NRAS/soil-health-initiative/state-of-the-soils/2023_sampling/
├── applications
├── coc
├── equipment
├── field-forms
├── forms
├── gis
├── lab-data
├── labels
├── management-surveys
├── public-docs
```

```
├── purchases
├── reports
├── sample-id-assignments
├── scripts
├── 2023-data-tracking.xlsx
└── 2023_post-season-wrap-up.docx
```

It's good practice to maintain the raw data, see how to set files to `Read-Only` in Section 5.2. We use additional subfolders for the `lab-data` folder. Everything in `raw` has been set as `Read-Only`.

```
Y:/NRAS/soil-health-initiative/state-of-the-soils/2023_sampling/lab-data
├── 2023_data-template-soiltest.xlsx
├── clean
├── qc
├── raw
└── working
```

`soil-health-initative` > `state-of-the-soils` > `2023_sampling` > `lab-data` > `clean` already has five levels of nesting. We wouldn't want to add many more levels, or the hierarchy becomes difficult to manage.

4.2 Archive folders

When too many drafts or versions begin to clutter a subfolder, create a new folder with the naming convention of `archive-folder-description`. Place the old drafts there. Leave the most current, accurate file in the main folder.

For example, the most recent sample labels for each conservation district are listed in the top level `completed-labels` folder, and previous working drafts were moved to the `archive-labels` folder.

```
Y:/NRAS/soil-health-initiative/state-of-the-soils/2023_sampling/labels/completed-labels
├── archive-labels
│   ├── cowlitz-county_labels.docx
│   ├── ferry-cd_labels.docx
│   ├── lewis-cd_labels.docx
│   └── stevens-cd_labels.docx
├── cowlitz-county_labels_v2.docx
├── ferry-cd_labels_v2.docx
├── ...
├── south-yakima-cd_labels.docx
├── stevens-cd_labels_v2.docx
└── walla-walla-cd_labels.docx
```

4.3 Code-based project organization

Code-based projects should be organized according to Section 9.1.1 in the code style guide.

5. Storage

Non-digital data, such as paper forms, must be transcribed or converted to digital file formats and then stored in the WaSHI filing cabinet in the Natural Resources Building in Olympia.

All digital data are stored in the WSDA shared drives, and other locations listed below.

WSDA shared drives:

- Agency files: Y:/NRAS/soil-health-initiative
- GIS: K:/NRAS/Arc_Data/soil-health (access requires permissions from IT)

Esri products and services:

- ArcGIS Online Soil Health - WSDA Internal Group
- WSDA GIS on-premise ArcGIS REST Services Directory (only Jadey, Perry, and Joel can publish to this server; Ed Thompson is the contact for getting access)

Database for lab results and management data:

- WISKI, but very likely will migrate to SQL Server or a less water-focused database

GitHub organizations for code-based projects:

- WSDA
- WaSHI

Microsoft Teams for data sharing between WSDA and WSU:

- WSDA and WSU Teams WaSHI channels

Box.com for external file sharing:

- WSDA has a box.com account. Only Perry has a license; Jadey and Dani are currently editors.

Individual devices (laptop, tablet, phone):

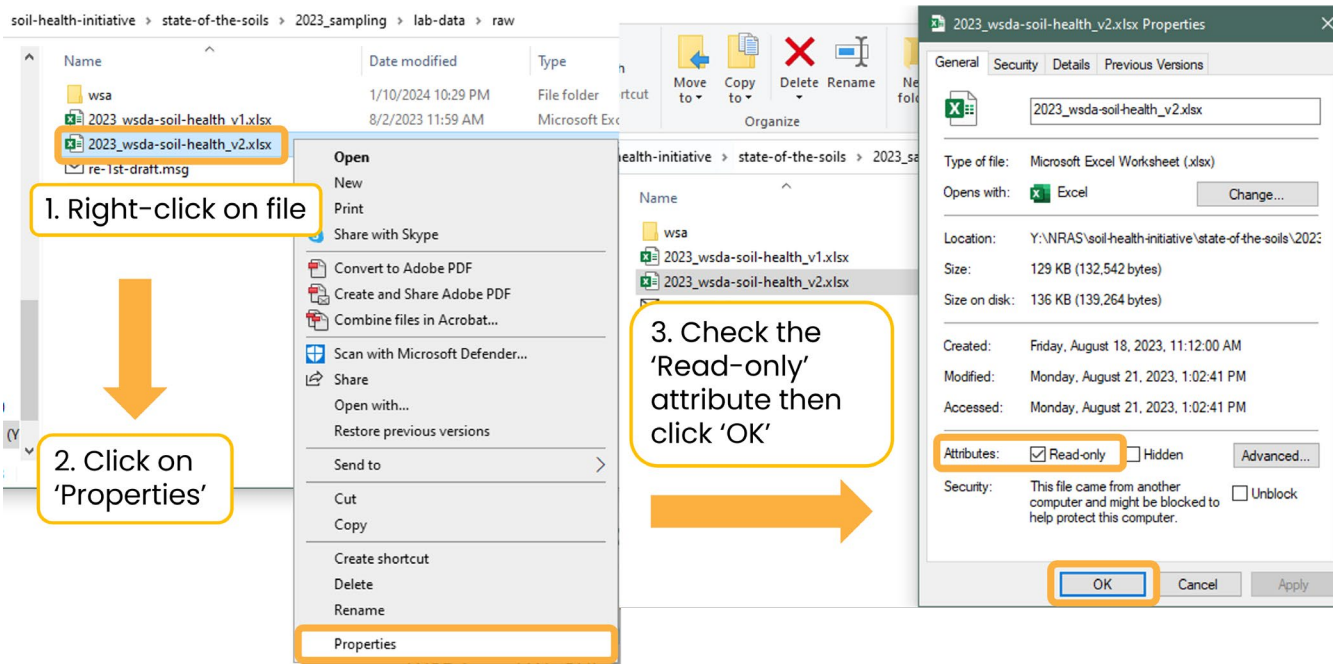
- Must NOT be the only place data are stored!

5.1 Backup

Data must be stored in multiple locations. At a bare minimum, data on an individual computer must also be saved on the WSDA shared drive. Backing up data using version control (GitHub) or a cloud service (Microsoft OneDrive or Box.com) is strongly recommended.

5.2 Read-only raw data

On our shared drives, raw data such as lab results from Soiltest or exports from ArcGIS Online, should immediately be set to `Read-only`. Right click the file > click on Properties > check the `Read-only` attribute box.



The file should then be copied over to a `working` folder for any processing or analyses. The final dataset should be saved in a separate `clean` folder, with a descriptive title. Keeping a `readme.txt` to document your processing steps is good practice, as discussed in Section 6.2.1.

5.3 Version control with Git and GitHub

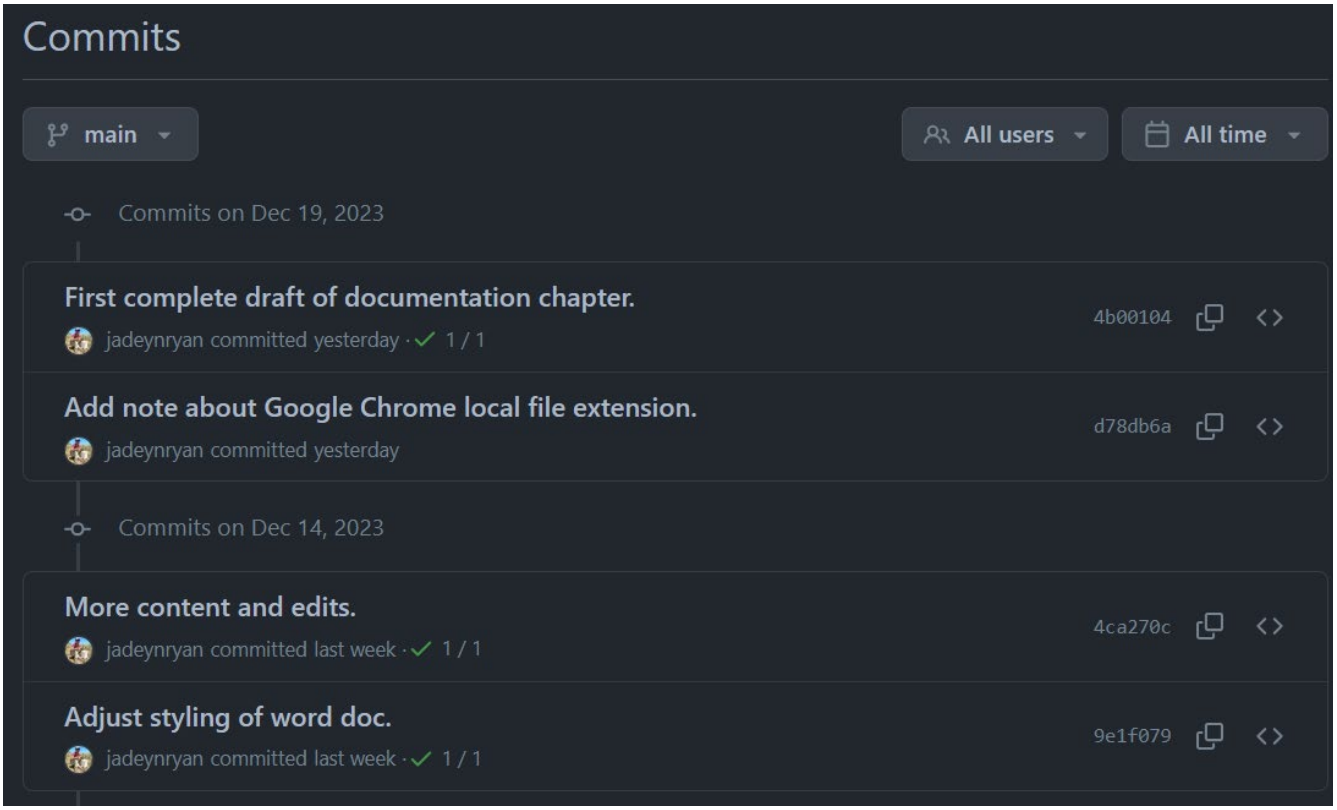
A version control system records changes to a file or set of files over time. Git is a free and open-source distributed version control system and GitHub is the hosting site WSDA and WaSHI use to interface with this system. Git and GitHub are an important foundation of reproducible statistical and data scientific workflows (Bryan 2018).

A major benefit of using version control is ensuring changes are well documented and previous versions are accessible if any changes must be recalled. Additionally, version control makes collaboration across projects much more robust.

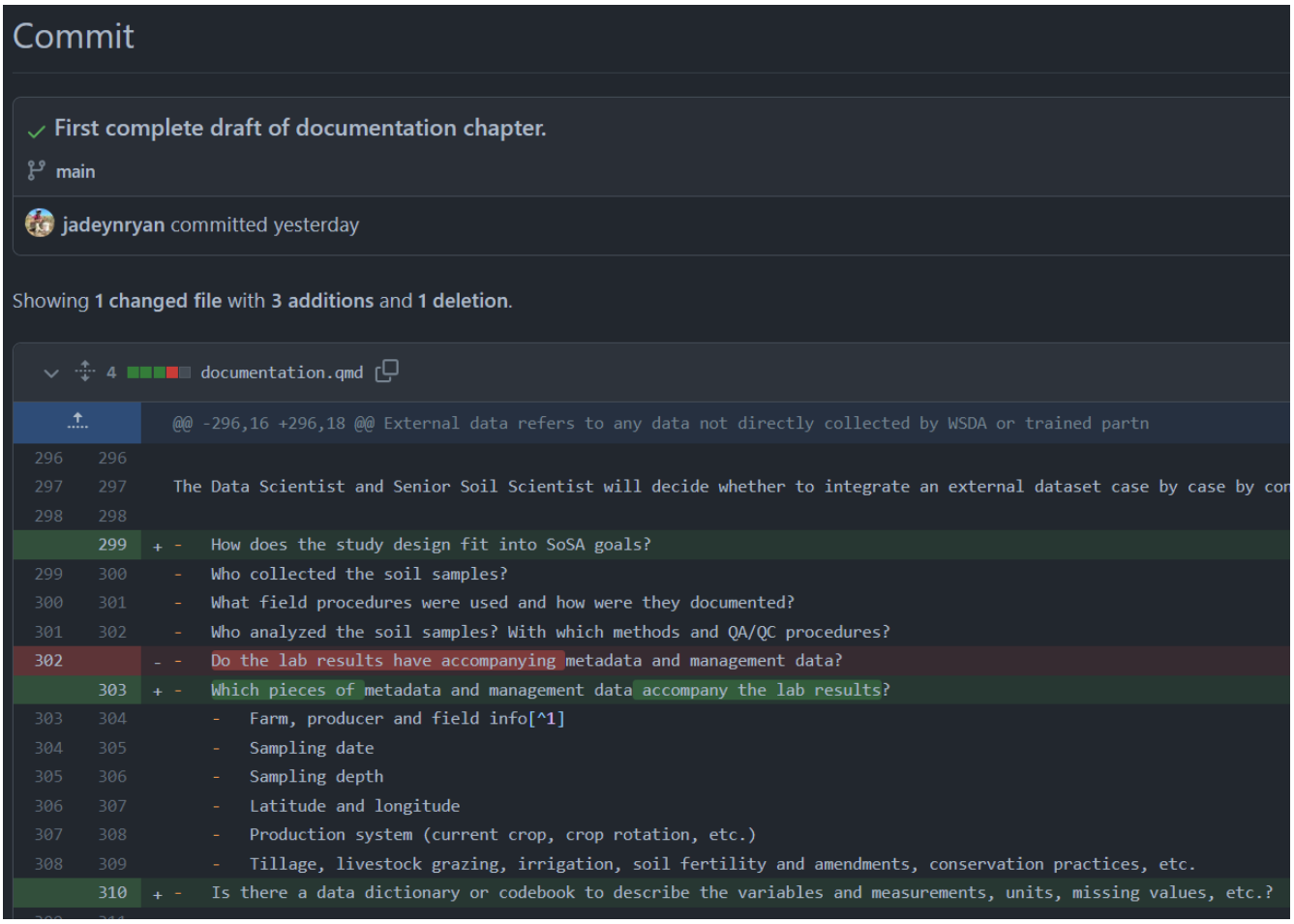
Version control is not just for code either! It's useful for scripts, documents, presentations, and books (like this DMP!). Instead of saving each version of a file with a different name (i.e., `report_v01.docx`

and `report_v02.docx`; for a reminder on version naming, see Section 3.2.7), there's only one file `report.docx` which automatically has its history and editors saved with Git and GitHub.

In the screenshot below, you can see who made a commit (which is basically a named version of changes), when that commit was made, and you can click on the commit message to view all of the files that were changed.



After clicking on the first commit message, we see the `documentation.qmd` file was changed with additions highlighted in green and deletions highlighted in red.



Privacy considerations

Review Chapter 8 to categorize the data included in the repository. If the data are not anonymized and aggregated, either 1) the repository must be set to private or 2) data files and any scripts containing Category 3 data as described in Section 8.1.0.2 must be added to the .gitignore file.

Git and GitHub resources

Read Jenny Bryan's article *Excuse Me, Do You Have a Moment to Talk About Version Control* for a great background on Git and GitHub, why we should be using it, and a brief how to get started. For detailed instructions , please follow along with Jenny Bryan's free online book *Happy Git and GitHub for the useR*. Another helpful book resource is *GitHub: A Beginner's Guide*, which was created by Birds Canada (avian conservation NGO) for people without a lot of programming background.

If you prefer to look through slides, check out Byron C. Jaeger's presentation *Happier version control with Git and GitHub (and RStudio)*.

6. Documentation

Documentation is the process of recording all aspects of project design; sampling; lab analyses; data cleaning; data analyses; data quality control and assurance procedures; and development of decision-support tools. Seem familiar? All of these aspects are also part of our data life cycle. We document our data throughout the life cycle to:

- standardize our procedures
- enable reproducibility
- establish credibility
- ensure others (including our future selves) use and interpret data correctly
- provide searchability

All documentation (including this document) should be updated (and versioned) as procedures change and lessons are learned.

Samples collected by WSDA for the SOS should follow the procedures and standards described in the below documentation.

External data must have at least the documentation outlined in Section 6.4 to be integrated into the SOS dataset.

6.1 Project-level

Project-level documentation includes all descriptive information about the SOS dataset, as well as planning decisions and process documentation. Documentation includes quality assurance project plans, standard operating procedures, and other high-level documents (i.e., request for proposals, applications, meeting agendas/notes, etc.).

Quality assurance project plan (QAPP)

The QAPP is the highest level of project documentation and covers everything from the project description; personnel roles and responsibilities; project timelines; data and measurement quality objectives; study design; and overviews of field, laboratory, and quality control.

Ours can be found in Y:/NRAS/soil-health-initiative/state-of-the-soils/qapp, though it needs to be updated.

Standard operating procedures (SOP)

SOPs provide detailed instructions for field, lab, or data processing procedures and decision making processes.

Ours can be found in Y:/NRAS/soil-health-initiative/state-of-the-soils/sop.

SOS sampling

The purpose of this SOP is to detail the procedures for a typical site visit in which soil samples are collected for physical, chemical, and biological soil health indicator analyses. Procedures include equipment preparation prior to sampling; best practices for filling out field forms; the selection of sampling locations; sampling protocols; sample handling and storage; and submitting samples to the lab. Following this SOP ensures data quality by creating audit trails and reminders to check that data are present, complete, and accurate. Additionally, this SOP will be used to maintain consistent sample collection procedures throughout the state for WSDA employees and partners.

Quality control / quality assurance (QA/QC)

This SOP outlines the process for screening sample metadata and lab results for completeness, consistency, and quality. Procedures involve subject matter expertise, investigation, communication with sampling teams and labs, algorithmic quality control, and tagging sample results with quality codes (listed in the below table). Data are then integrated into the statewide database according to a SOP not yet authored.

Code	Tag	Description	Inclusion in analyses
0	Excellent	Met lab's and WSDA's QC criteria	Yes
100	Estimate	Interpolated missing value	Yes
110	Derived	Derived from an estimated value	Yes
120	Suspect	Z-score is ≥ |3|	Yes
130	Calculated ND	Calculated value using at least one ND	Yes
140	Non-detect	Below the method detection limit	No
160	Poor	Did not meet lab's QC criteria	No
180	Outlier	Outlier, designated by soil scientist	No
200	Unknown	External dataset	Yes

SOPs we don't have (yet)

- assigning producer, field, and sample IDs
- data cleaning
- data import into the database
- data storage
- external data integration

6.2 Dataset-level

Dataset-level documentation applies to lab results, sample locations, grower information, and management data. We use readmes and changelogs to document what each dataset contains, how they are related, potential issues to be aware of, and any alterations made to the data.

Readme

`readme` files are plain text documents that contain information about the files in a folder, explanation of versioning, and instructions/metadata for data packages. These files are saved as `.txt`, instead of MS Word documents that take longer to open and can only be opened on computers with Microsoft installed.

Describe contents of folder

For example, the `_complete-dataset` folder contains a readme.txt that describes each files' structure, contents, and other pertinent information, such as the data source.

Explain versions

Another example is the readme.txt in the `2023_sampling` > `lab-data` > `raw` folder, which explains why there are two different versions of the lab results and where to find additional information.

Provide instructions

An example of a readme.txt that provides instructions on how to use the folder contents can be found in the ArcGIS soil sample points box.com folder. When this folder is shared with our partners, the readme helps them orient to the contents of the folder and modify the files as needed for their own adaptation.

Changelog

Changelogs are also simple and concise plain text documents saved in a folder alongside data files to document any changes to the dataset.

At the bare minimum, the `changelog.txt` should contain:

- date of modification
- initials of who made the changes
- description of the changes

6.3 Variable-level

Variable-level documentation includes data dictionaries and codebooks, which are often talked about interchangeably. However, we'll refer to the **data dictionary** as a tabular collection of names, definitions, and attributes about the variables in a dataset. Data dictionaries are ideally created in

the planning phase of the project before data are collected. In contrast, **codebooks** provide descriptive, variable-level information and univariate summary statistics to allow users to understand the contents of a dataset without opening it. The codebook is created or updated after data are collected, cleaned, and validated.

Data dictionary

In a data dictionary, each row is a different variable, while each column is a different attribute of that variable. With a data dictionary, any user should be able to properly interpret each variable in our data.

Our data-dictionary.xlsx in the _complete-dataset folder contains two tabs (`lab-results` and `sample-locations`) that describe the attributes of each variable.

TODO: update variable names in R scripts, data dictionary, and Soiltest template.

Codebook

Codebooks provide more information (i.e., existing values/ranges and summary statistics) than the data dictionary and can be used to understand a very high-level summary of the processed data. There are many R packages that generate codebooks; however, we have not implemented this type of documentation for our project yet.

Crystal Lewis gave the lightning talk *A Comparison of Packages to Generate Codebooks*. Once, our data live in a database, I'd like to generate codebooks.

6.4 External data

External data refers to any data not directly collected by WSDA or trained partners (e.g., WSU or conservation districts) that follow our SOPs. These can include other studies pre-dating WaSHI, special soil health surveys, or publicly available datasets.

The Data Scientist and Senior Soil Scientist will decide whether to integrate an external dataset case by case by considering the following questions:

- How does the study design fit into SOS goals?
- Who collected the soil samples?
- What field procedures were used and how were they documented?
- Who analyzed the soil samples? With which methods and QA/QC procedures?
- Which pieces of metadata and management data accompany the lab results?

  - Farm, producer and field info[1]
  - Sampling date
  - Sampling depth
  - Latitude and longitude
  - Production system (current crop, crop rotation, etc.)
  - Tillage, livestock grazing, irrigation, soil fertility and amendments, conservation practices, etc.
- Is there a data dictionary or codebook to describe the variables and measurements, units, missing values, etc.?

Generally, the external data should **1)** be well documented, **2)** be collected and analyzed by well-trained scientists and labs; and **3)** have adequate accompanying metadata and management data to facilitate interpretation of the results.

Some publicly available datasets to consider are listed in Y:/NRAS/soil-health-initiative/state-of-the-soils/data-sources.

[1] Enough farm, producer and field info to distinguish unique farmers and fields for assigning unique IDs. This info doesn't need to be personally identifiable information.

7. Data flow

Data flow refers to how information moves through a system and how it's processed along the way. This chapter outlines how our data are generated, processed, and moved from start to finish.

This chapter is subject to lots of changes as 2024 is the final year of our soil sampling program through funding conservation districts (CDs). However, since the CDs are now trained according to the WSDA SOP, we will accept their data provided they also submit management surveys. See Section 6.4 for the requirements to accept external data. Additionally, we do not yet have a database so the importing to the database section will be completed in the future.

2020 - 2023 data and scripts are housed in the `soils-internal` monolithic repository. Moving forward, each year (or special project) should have its own project and repository.

> ⓘ Accessible font
>
> For printed sample ID assignments, labels, chain of custodies, etc., use Atkinson Hyperlegible because it is very accessible and easy to differentiate 1) numeric zero 0 from uppercase O and 2) numeric one 1, lowercase l, and lowercase i.
>
> Download Atkinson Hyperlegible from Google Fonts.
>
> 
>
> "0O" and "1li" in Atkinson Hyperlegible.

7.1 Pre field season

Assign unique identifiers

Before sample IDs can be assigned, collect the following information for each proposed sample:

- County
- Organization of sampling team
- Farm name (optional)
- Producer name
- Producer contact information (optional)
- Field name
- Crop
- General management practice (i.e., conventional, cover crop, reduced tillage)

View examples of the 2023 Sample Request Form sent to CDs and the Berries Sample Request Form for the WSDA/WSU special project.

Once producers and fields have been identified, we assign a unique ID for the producer, field, and sample with the following convention:

- **Producer ID**: first three letters of county + three-digit landowner number
  - `WHA001`
- **Field ID**: two-digit field number
  - `01` and `02`
- **Pair ID** (optional): letter extension added to paired fields
  - `A`
- **Sample ID**: last two digits of year + Producer ID + Field ID + Pair ID
  - `24-WHA001-01-A` and `24-WHA001-02-A`

The following counties have different abbreviations than their first three letters:

- Clallam → CLL
- Grays Harbor → GRY
- Kitsap → KIS
- Skamania → SKM

Producer and field IDs should first be matched to previous participants. New producers and fields should continue the previous sequence. There should be no duplicate producer IDs or sample IDs.

For an example R script to automate this process, see assign-sample-ids.R.

Create sample labels

We automate sample label creation using R and Microsoft Word's mail merge tool. labels.R generates a spreadsheet with all column names to be printed on the labels. Then labels-template-mail-merge.docx ingests this spreadsheet and the data scientist runs the mail merge to generate a word document with all of the labels to be printed, as shown in the completed-labels folder.

Create a data tracking sheet

We keep a spreadsheet to track which pieces of data we have for each sample ID, including:

- GPS points submitted through the ArcGIS Field Maps field form
- Scanned paper field forms (for those without ArcGIS Field Maps)
- Management surveys through ArcGIS Survey123
- Scanned chain of custodies with shipping tracking numbers
- Location of archival falcon tubes

- Notes for if a sample will no longer be sampled, a sample ID was changed, etc.

See the 2023 spreadsheet for an example.

Develop ArcGIS web tools

We use ArcGIS to build tools for managing spatial data and collecting management survey data. A sample selection feature layer is hosted and a web map with offline capabilities is created using ArcGIS Pro. Domains are used for point numbers, bulk density, and crop types. Then, a field form is created on ArcGIS Online using Field Maps. Management surveys are created and hosted with Survey123 and Experience Builder. We also use an ArcGIS Notebook with python to back up our feature layer and survey data.

This template ArcGIS Pro project includes a readme.txt that describes this process.

The ArcGIS Notebook is set to run as a task Monday, Wednesday, and Friday during the field season.

View code from the ArcGIS Notebook

```
import arcgis
from arcgis.gis import GIS
import datetime as dt
from datetime import timezone, timedelta
gis = GIS("home")

folder_path = '/arcgis/home/backups/2023/points'
title = "2023*"
owner = "jryan_NRAS"
items = gis.content.search(query = "title:" + title + " AND owner:" + owner,
                           item_type='Feature Layer')
print(str(len(items)) + " items will be backed up to " + folder_path +". See the li
st below:")
items

def download_as_fgdb(item_list, backup_location):
    for item in item_list:
        try:
            if 'View Service' in item.typeKeywords:
                print(item.title + " is view, not downloading")
            else:
                print("Downloading " + item.title)
                version = dt.datetime.now(timezone(timedelta(hours=-8))).strftime("
%Y-%m-%d")
                result = item.export(item.title + "_" + version, "File Geodatabase"
)
                result.download(backup_location)
                result.delete()
                print("Successfully downloaded " + item.title)
        except:
```

```
            print("An error occurred downloading " + item.title)
    print("The function has completed")

download_as_fgdb(items, folder_path)

folder_path = '/arcgis/home/backups/2023/surveys'
title = "2023 * Survey* Production"
owner = "dgelardi_NRAS"
items = gis.content.search(query = "title:" + title + " AND owner:" + owner,
                           item_type='Feature Layer')
print(str(len(items)) + " items will be backed up to " + folder_path +". See the li
st below:")
items

def download_as_fgdb(item_list, backup_location):
    for item in item_list:
        try:
            if 'View Service' in item.typeKeywords:
                print(item.title + " is view, not downloading")
            else:
                print("Downloading " + item.title)
                version = dt.datetime.now(timezone(timedelta(hours=-8))).strftime("
%Y-%m-%d")
                result = item.export(item.title + "_" + version, "CSV")
                result.download(backup_location)
                result.delete()
                print("Successfully downloaded " + item.title)
        except:
            print("An error occurred downloading " + item.title)
    print("The function has completed")

download_as_fgdb(items, folder_path)
```

7.2 During field season

Data collection in the field is detailed in the monitoring SOP. We'll focus on the behind-the scenes tasks for managing data.

Update data tracking spreadsheet

Throughout the season, the data tracking spreadsheet must be updated as various forms and surveys, as described in create a data tracking sheet, are received.

Modify IDs when samples change

Sometimes a producer can no longer participate, or they want to change which field is sampled. The sample request form should be updated, versioned, and archived (`sample-request-form-ferry.xlsx` → `sample-request-form-ferry_v2.xlsx`).

The `assign-sample-ids.R` script should be run again to update the sample IDs. Lines 362 - 386 should be commented out as shown in the highlighted lines of the script on GitHub. Note: This link will take you to a 404 page if you are not logged into a GitHub account that is part of the WSDA organization.

Take a look through the 01_returned-sample-requests and 02_completed-sample-ids folders for an example of this flow.

A concise, explanatory note should be added to the data tracking spreadsheet.

7.3 Post field season

Organize multiple sources of data

We have many sources of data such as the initial sample request forms, ArcGIS Field Maps field forms, and management surveys. To organize these multiple sources into a single source of truth, we cross-reference each source and reach out to the sampling teams to resolve conflicting information. This is especially important for verifying the crop that was planted at the time of sampling.

See how to *mostly* automate this process in these 01_load-metadata.R and 02_check-crops.R scripts.

Process lab data

Follow the QA/QC SOP for processing the lab data.

See the 2023 processing scripts and QA/QC report:

- 03_process-spatial-data.R
- 04_load-lab-data.R
- 05_calculate-z-scores.R
- 2020-2023_qc-results-summary.qmd

Generate reports

Use the {soils} package to create a new project for each year. To avoid email attachment size limitations, reports may be saved to Box.com for distribution to the sampling partners who will send the reports to the participants. Note: this folder is private unless invited by Jadey, Dani, or Perry.

Import data to database

TBD. Currently data are only in spreadsheets.

Save data to shared drive and WSU Teams channel

The output data files and reports from process lab data and generate reports are copied to the state-of-the-soils folder in its respective `year_sampling` folder. Review Chapter 4 and look at previous years in the shared drive to follow the same folder structure and organization.

The final datasets in wide and long formats should also be saved to the WSU SCBG Soil Health Assessment Teams channel. If you don't have access to the *Data for stats* folder, email it to Deirdre Griffin-LaHue.

Archive jars and falcon tubes

Archival subsamples in glass jars are stored in the Yakima WSDA storage room and the cryogenic archive subsamples in falcon tubes are stored in the -80 °C freezer at the WSU Mount Vernon Northwestern Washington Research & Extension Center.

Labels on the falcon tubes must be taped with a generous amount of packing tape to avoid falling off as they stiffen up and flatten out when they freeze.

The archive spreadsheet must be updated.

8. Data sharing

Our data sharing policies promote FAIR principles such that our data are "**as open as possible, as closed as necessary**" (European Commission 2016). Data should be **open** enough to facilitate efficient re-use; avoid duplicating data collection efforts; enhance scholarly rigor; promote engagement across the research and public communities; and realize many other benefits (Whyte and Pryor 2011). However, data must also be as **closed** as necessary to protect grower privacy and to honor prior agreements with growers or other researchers.

SOS relies on growers' willingness to volunteer their fields for sampling and participate in the required management survey. Their willingness depends on their trust in WaSHI to protect their privacy. Only aggregated and anonymized results will be publicly available or shared. The below data privacy statement may be shared with potential participants.

Data privacy statement

> Data will be aggregated and reported in a way which mitigates personal identification of growers. Information will be used to understand broad impacts and characterize trends in soil health and production practices across regions. Results will not be reported in a way that makes individuals identifiable. Information collected in this survey may be subject to release in accordance with RCW 42.56 (Public Records Act).

Procedures for anonymizing data are detailed in Section 8.2.

8.1 WaTech data categorization

Under Washington State Policy 141.10 (Securing Information Technology Assets), state agencies must classify data into categories based on the sensitivity of the data. WaTech provides guidance on the four categories of data.

Category 4: "Confidential information requiring special handling"

WaTech: Data requires strict handling requirements applied by statues (e.g. HIPAA) or regulations (e.g. rules on employee files).

SOS: We don't manage any data under this category.

Category 3: "Confidential information"

WaTech: Data includes "personal information" as defined in RCW 42.56.590 (Security Breaches) and RCW 19.255.010 (Personal Information Disclosure). An individual's first name or first initial and last name *in combination* with at least one of the following elements: social security number, driver's license or Washington identification card number, or any account numbers that permit access to their financial account.

SOS: While we do not collect any of the above elements in combination with grower names, we still protect individual and farm names, and latitude and longitude coordinates as confidential information.

Category 2: "Sensitive information"

WaTech: Data are intended for official use only and withheld unless specifically requested.

SOS: Lab results and management surveys fall under this category. Access to this data requires a data share agreement.

Category 1: "Public information"

WaTech: Data is not covered in any of the above categories or is already released to the public.

SOS: De-identified and aggregated data such as the number of soil samples and from which counties and crops they were collected fall under this category. For example, the SOS dashboard is publicly available and the map zoom is disabled at the 1:1,600,000 scale (counties level).

8.2 Maintain confidentiality

Only under special circumstances and with proper justification in the data share agreement should the following Category 3 data be released to external collaborators. Under no circumstances should these data be made publicly available.

- farm name
- grower first and last name
- field names that contain street names or other identifying information
- latitude and longitude coordinates or other geospatial identifiers
- any information that identifies the individual farm or grower

Category 2 data should be anonymized and aggregated to honor our data privacy statement by either removing or replacing Category 3 confidential information with dummy data. The {randomNames} R package can be used to replace real names with fake names. Latitude and longitude should be rounded to a precision that does not identify the farm or fields sampled.

See the example R script below, copied from the private soils-internal GitHub repo (if you aren't part of the WSDA GitHub organization, this link will give you a 404 error). Note: this script does not follow the code style guide in Chapter 9 as it was written prior to this DMP.

R code to anonymize data

```
library(dplyr)
library(readxl)
library(janitor)
library(randomNames)
```

```r
load("./data/_completeDataset/2020-2023_labResults_wide.Rdata")
load("./data/_completeDataset/2020-2023_labResults_long.Rdata")
crops <- read_excel("./data/reference/crops.xlsx")
points <- read.csv("./data/_completeDataset/2020-2023_sampleLocations.csv") |>
  select(sampleId, longitude, latitude) |>
  mutate(across(where(is.numeric), \(x) round(x, 0)))

# Join lab data with gis data
data <- left_join(allResults_wide, points) |>
  # Remove WSU SCBG samples and 0-6/6-12 in WSDA samples
  subset(!year %in% c(2020, 2021) &
           !grepl("[A-Z]", fieldId)) |>
  janitor::remove_empty("cols")

# Join data with crop dictionary to get crop groups
data <- left_join(data, crops, by = c("crop" = "Crop Type")) |>
  select(-crop) |>
  relocate("crop" = "Crop Group", .after = county)

# Anonymize
anonData <- data |>
  # Farm name, producer name, sample ID
  mutate(
    farmName = paste0("Farm ", sprintf("%03d", cur_group_id())),
    producerName = randomNames(which.names = "first",
                               sample.with.replacement = FALSE),
    producerId = paste0(
      paste0(sample(LETTERS, 3, replace = TRUE), collapse = ""),
      "0",
      paste0(sample(1:9, 1, replace = TRUE), collapse = "")
    ),
    fieldName = paste0("Field ", fieldId),
    sampleId = paste0(
      substr(year, 3, 4), "-", producerId, "-", fieldId
    ),
    .by = c(county, producerName, producerId)
  ) |>
  # county
  mutate(
    county = paste0("County ", cur_group_id()),
    .by = c(county)
  )

# Grab 100 random samples by county
anon_subset <- anonData |>
  slice_sample(n = 100)

write.csv(anon_subset, file = "./data/_completeDataset/exampleData.csv",
```

```
          na = "",
          row.names = FALSE)
```

8.3 Data share agreement

SOS CoPIs created a Data Sharing and Scope of Work Agreement that details the type of data to be shared, the scope of work in which the data may be used, and terms for using SOS data.

Once the agreement has been signed by both CoPIs and the "Partnering Scientists", the agreement should be saved in its own folder within Y:/NRAS/soil-health-initiative/state-of-the-soils/data-sharing. If this agreement is also part of a grant proposal, it should also be saved in its corresponding grant folder Y:/NRAS/soil-health-initiative/contracts-grants/grants.

The request correspondence, code to subset the requested data, and the final dataset sent should all be saved in this folder. Internally documenting this information allows us to track publications and attributions resulting from this data sharing. This documentation also helps us track the broader impact of how our collaborators' use the SOS dataset.

8.4 Public access

Currently the only SOS data publicly available are the counts of samples across the project, counties, and crop types displayed in the ArcGIS Online Dashboard.

A small, anonymized subset is included as example data in the {washi} and {soils} R packages for demonstration purposes.

TODO: discussion with team on the below ideas

In the future when the data are more mature and hosted in a proper database, we may publish an anonymized subset in a public repository such as:

- GitHub via an R package or Shiny app
- Zenodo: integrates with GitHub and is citable with a DOI
- Data.WA.gov: open data portal for the State of Washington

More inspiration and ideas for enhancing data discoverability and sharing across the agricultural and soil health communities include:

- USDA LTAR data dashboards
- CAF LTAR metadata tool

Chapter 16 of *Data Management in Large-Scale Education Research* discusses more considerations for data sharing and choosing public repositories (Lewis 2023).

8.5 Acknowledgments

All research and data partially or completely funded by WaSHI must include acknowledgements to the State of Washington. The following text should be included in all publications resulting from this funding:

> Data were in part provided by the Washington Soil Initiative, which is supported by the State of Washington and administered by the Washington State Department of Agriculture, Washington State Conservation Commission, and Washington State University.

If WaSHI staff make substantial scientific contributions to the manuscript, discuss the possibility of co-authorship credit.

9. Code style guide

This style guide is an adaptation of the *tidyverse style guide* and includes best practices from *R for Data Science (2e)* (*R4DS*), *Data Management in Large-Scale Education Research*, and other resources.

All WaSHI staff who code in R should thoroughly read and consistently implement this style guide.

Using consistent project structures, naming conventions, script structures, and code style will improve code readability, analysis reproducibility, and ease of collaboration.

> Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread…
>
> All style guides are fundamentally opinionated. Some decisions genuinely do make code easier to use (especially matching indenting to programming structure), but many decisions are arbitrary. The most important thing about a style guide is that it provides consistency, making code easier to write because you need to make fewer decisions.
>
> - Hadley Wickham in the tidyverse style guide

9.1 Projects

All files associated with a given project (input data, R scripts, analytical results, figures, reports) should be kept together in one directory. RStudio has built-in support for this through **projects**. R projects bundle all the work in a portable, self-contained folder that can be moved around on your computer or on to other collaborators' computers and still work.

Learn more about projects in the Workflow: scripts and projects chapter of *R4DS*, in Jenny Bryan's article *Project-oriented workflow*, and Shannon Pileggi's workshop slides.

To create a project in RStudio, click `File` > `New Project`, then follow the steps in the below figure.

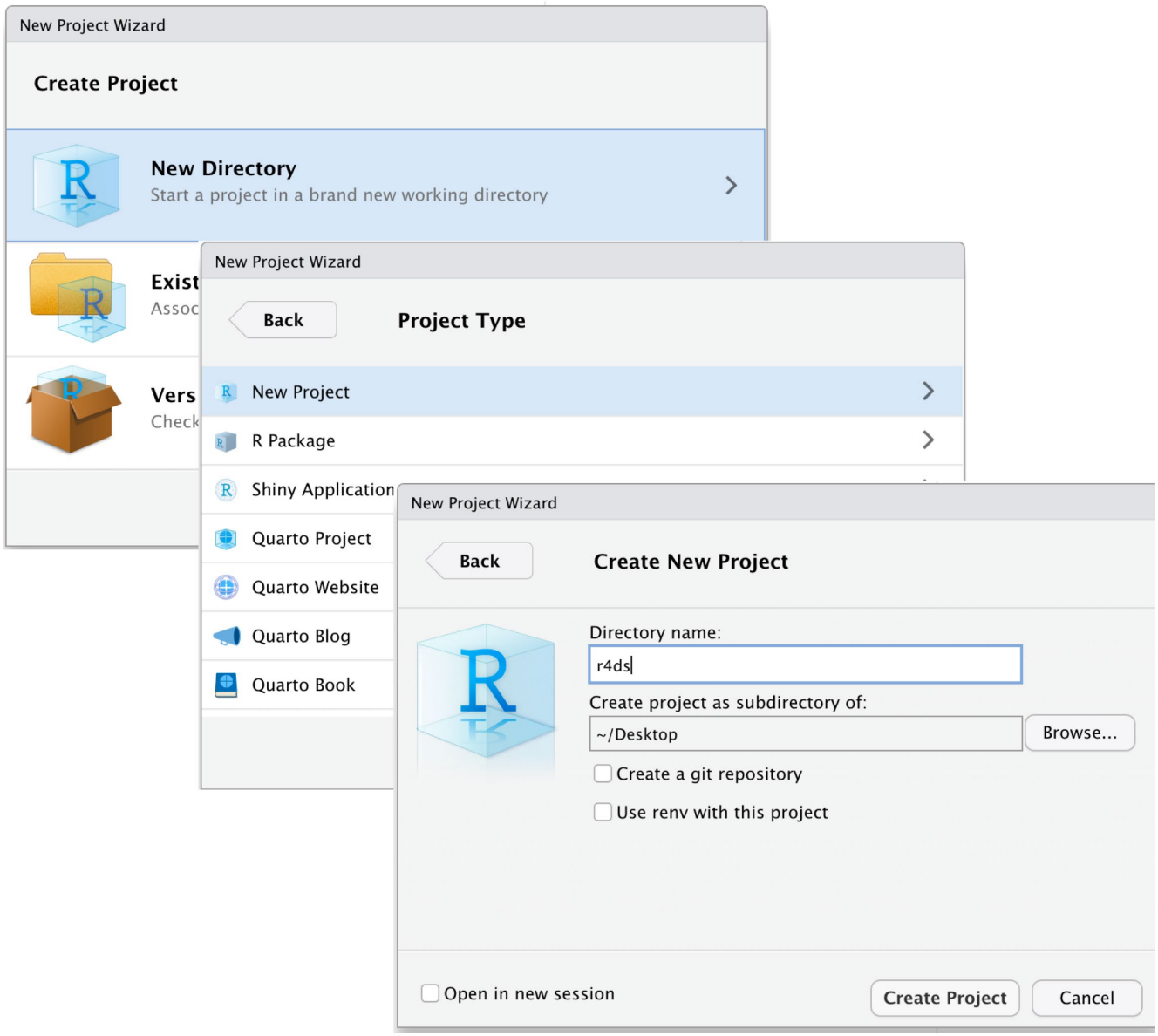


Figure from Chapter 6 of *R4DS*

The project folder should be checked into GitHub for version control as discussed in Section 5.3.

If not checked into GitHub, at bare minimum, the folder should be copied onto the shared drive.

Project folder structure

Having a consistent and logical folder structure will make it easier for you (especially future you) and collaborators to make sense of the files and work you've done. Well documented projects also make it easier to resume a project after some time away with minimal frustration of having to remember where everything is, what you did, and why you did it.

The below structure works most of the time and should be used as a starting point. However, different projects have different needs, so add and remove subfolders as needed.

- **root**: top-level project folder containing the `.Rproj` file.
- **data**: contains raw and processed data files in subfolders. Raw data should be made *read-only* and not changed in any way. See Section 5.2 for a reminder on how to make a file read-only.
- **output**: outputs from R scripts such as figures or tables.
- **R**: all R scripts containing data processing or function definitions.
- **reports**: Quarto or RMarkdown files are saved here, as well as the resulting reports.
- **README**: a markdown file (can be generated from Quarto or RMarkdown) to explain the project.

See an example project folder structure

```
├── project-demo.Rproj
├── data
│   ├── processed
│   │   └── data-clean.csv
│   └── raw
│       └── data-raw.xlsx
├── output
│   ├── fig-01.png
│   ├── fig-02.png
│   ├── tbl-01.png
│   └── tbl-02.png
├── R
│   ├── 01_import.R
│   ├── 02_tidy.R
│   ├── 03_transform.R
│   ├── 04_visualize.R
│   └── custom-functions.R
└── reports
    ├── soil-health-report.pdf
    ├── soil-health-report.qmd
    └── images
        └── logo.png
├── README.md
└── README.qmd
```

R packages may contain these additional subfolders and files:

- **inst**: any arbitrary additional files to be included with package installation such as `CITATION`, fonts, and Quarto templates.
- **man**: all `.Rd` ("R documentation") files for each function that are generated from {roxygen2}.

- **vignettes**: long-form guides that go beyond function documentation and can be used as tutorials to demonstrate a workflow using the package to solve a particular problem.
- **tests**: all test files, usually using {testthat}.
- **pkgdown** and **docs**: if using {pkgdown} to build a website for the package, you may have a pkgdown folder containing the favicon and any additional css and a docs folder containing the website source files.
- **DESCRIPTION**: file containing metadata about the package (authors, current version, dependencies).
- **LICENSE**: file describing the package usage agreement.
- **NAMESPACE**: file generated by {roxygen2} listing functions imported from other packages.
- **NEWS.md**: file documenting user-facing changes to a package.

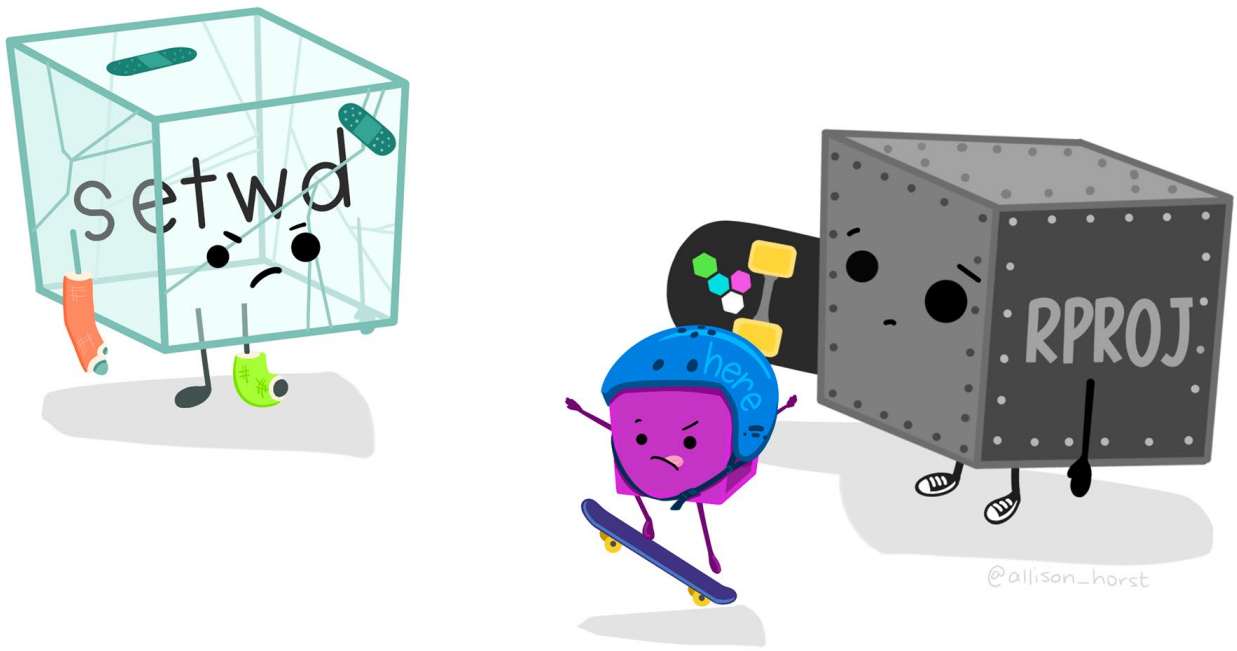
Learn more about other R package components in *R Packages (2e)*.

Absolute vs relative paths

> ⓘ Note
>
> I use directories and folders interchangeable here. But if you're interested in the technical differences: **directories** contain folders and files to organize data at *different levels* while **folders** hold subfolders and files in a *single level.*

❌ Absolute paths start with the root directory and provide the full path to a specific file or folder like `C:\\Users\\jryan\\Documents\\R\\projects\\project-demo\\data\\processed`.[2] You can run `getwd()` to find out where the current working directory is and `setwd()` to set a specific folder as your working directory. However, **please don't do use `setwd()`** because this absolute file path is going to break your code if you reorganize your directory and is not going to work on any collaborators' computers as their directory configuration will be different.

[2] Note the two backslashes. Windows paths use backslashes, which mean something specific in R. to get a single backslash in the path, we need to type two backslashes (or use forward slashes).



A cartoon of a cracked glass cube looking frustrated with casts on its arm and leg, with bandaids on it, containing "setwd", looks on at a metal riveted cube labeled "R Proj" holding a skateboard looking sympathetic, and a smaller cube with a helmet on labeled "here" doing a trick on a skateboard. Artwork by @allison_horst.

✅ Instead, always use relative paths in your scripts. Relative paths are *relative* to the working directory (i.e. the project's home) like `data/processed/data-clean.csv`. When working in a RStudio project, the default working directory is always the **root** project directory (where the `.Rproj` file is).

For example, say `01_import.R` contains the code `read.csv("data/processed/data-clean.csv")`. This will read the file from `C:/Users/jryan/Documents/R/projects/project-demo/data/processed/data-clean.csv`". The magic of relative paths means that if Dani were have this project on her desktop and run this code, it would read the file from `C:/Users/dgelardi/Desktop/project-demo/data/processed/data-clean.csv`. This is why relative paths are so important – they work no matter where the project folder is!

{here} package

In combination with R projects, use the {here} package to build *relative* file paths. This is especially important in Quarto files because when the `.qmd` file renders, its default current working directory is wherever the `.qmd` file lives. If we are using the above example project structure and wanted to read in our clean data to our `soil-health-report.qmd` file, we would get an error running

`read.csv("data/processed/data-clean.csv")` because it would be looking for a data subfolder in the reports folder. Instead, we can use the {here} package to build a relative path from our root with `read.csv(here::here("data", "processed", "data-clean.csv"))`. {here} takes care of the backslashes or forward slashes so the relative path will work no matter the operating system.

A cartoon showing two paths side-by-side. On the left is a scary spooky forest, with spiderwebs and gnarled trees, with file paths written on the branches like "~/mmm/nope.csv" and "setwd("/haha/good/luck/"), with a scared looking cute fuzzy monster running out of it. On the right is a bright, colorful path with flowers, rainbow and sunshine, with signs saying"here!" and "it's all right here!" A monster facing away from us in a backpack and walking stick is looking toward the right path. Stylized text reads "here: find your path." Learn more about here. Artwork by @allison_horst.

9.2 Naming conventions

> "There are only two hard things in Computer Science: cache invalidation and naming things."
>
> — Phil Karlton

Based on this quote, Indrajeet Patil developed slides with a lot of detailed language-agnostic advice on naming things in computer science.

R code specific naming conventions are listed below.

> **Note**
>
> The code naming convention here applies primarily to R. Python and JavaScript have different naming standards.
>
> **TODO**: add Python naming conventions? They differ from R as static variables (aka constants) are supposed to be SCREAMING_SNAKE_CASE.

Project folder, .RProj and GitHub repository

The project folder, .RProj file, and GitHub repository name should be the same. Be concise and descriptive. Use kebab-case.

Example: `washi-dmp` and `washi-dmp.RProj`.

Files

Be concise and descriptive. Avoid using special characters. Use kebab-case with underscores to separate different metadata groups (date_good-name).

Examples: `2024_producer-report.qmd`, `tables.R`, `create-soils.R`.

If files should be run in a particular order, prefix them with numbers. Left pad with zero if there may be more than 10 files.

Example:

```
01_import.R
02_tidy.R
03_transform.R
04_visualize.R
```

Variables, objects, and functions

Let's first define these terms so we're on the same page. In this style guide, **variables** are column names in spreadsheets (that become column names in R dataframes), **objects** are all data

structures in R and ArcGIS (vectors, lists, dataframes, fields, tables), and **functions** are self-contained modules of code that accomplish a specific task.

Variable examples:

```
# Good
clay_percent
min_c_96hr_mg_c_kg_day
pmn_mg_kg

# Bad

# Uses special character
clay_%

# Less human readable, inconsistent with style guide,
# starts with number and will error in R
96hrminc_mgckgday

# Hyphen will need to be escaped in R code to avoid error
pmn-mgkg
```

Objects and functions

Objects names should be nouns, while function names should be verbs (Wickham 2022). Again, use lowercase letters, numbers, and underscores. Do not put a number as the first character of the name. Do not use hyphens. Do not use names of common functions or variables.

Object examples:

```
# Good
primary_color
data_2023

# Bad

# Less human readable, inconsistent with style guide
primarycolor

# Using a hyphen in an object name causes error
data-2023 <- read.csv("2023_data-clean.csv")
Error in data - 2023 <- read.csv("2023_data-clean.csv") : could not find function "
-<-"

# Starting an object name with a number also causes error
2023_data <- read.csv("2023_data-clean.csv")
Error: unexpected input in "2023_"

# Overwrites R shortcut for TRUE
T <- FALSE
```

```
# Overwrites R function
c <- 10
```

Function examples:

```
# Good
add_row()
assign_quality_codes()

# Bad

# Uses noun instead of verb
row_adder()

# Inconsistent with style guide
assignQualityCodes()

# Overwrites common base R function
mean()
```

9.3 R scripts

Header template

Including a header in every R script helps standardize the metadata elements provided at the beginning of your code and documents its purpose. Dr. Timothy S Farewell wrote a great blog post for creating a template for the header of every R script. The following template and instructions are adapted from his post (Farewell 2018).

1. **Script name**: meaningful and concise.
2. **Purpose**: brief description of what the script aims to accomplish.
3. **Author(s) and email**: it's good to know where the script originated from if there are any questions, comments, or improvements.
4. **Date created**: this is automatically filled in from the template.
5. **Notes**: free-text space for any thoughts or todos.

```
## Header ==========================================================================
## Script name: check-crops.R
##
## Purpose: Cross reference sample requests, Field Maps forms, and management
## surveys to get the correct crop planted at the time of sampling.
##
## Author: Jadey Ryan
##
## Email: jryan@agr.wa.gov
##
## Date created: 2024-01-02
```

```
##
## Notes:
##

# Attach packages ============================================================

library(readxl)
library(writexl)
library(janitor)
library(dplyr)
library(tidyr)

# Load data ==================================================================
```

Add this template to RStudio using snippets:

1. Modify the below code with your name and preferred packages.
2. In RStudio, go to `Tools` > `Edit Code Snippets`.
3. Scroll to the bottom of the R code snippets, and paste your modified code (the indent and tabs are important!).
4. Click `Save` and close the window.
5. Try it out by opening a new blank `.R` script, typing "header", and then pressing `Shift` + `Tab`.

```
snippet header
    ## Header ====================================================================
    ##
    ## Script name:
    ##
    ## Purpose:
    ##
    ## Author: Jadey Ryan
    ##
    ## Email: jryan@agr.wa.gov
    ##
    ## Date created: `r paste(Sys.Date())`
    ##
    ## Notes:
    ##

    # Attach packages ============================================================

    library(readxl)
    library(writexl)
    library(janitor)
    library(dplyr)
    library(tidyr)

    # Load data ==================================================================
```

Section template

The above header template also uses section breaks (commented lines with `=` that break up the script into easily readable chunks). Section breaks are fantastic tools in RStudio because they allow you to easily show or hide blocks of code, see an outline of your script, and navigate through the source file. Read more about code folding and sections in this Posit article.

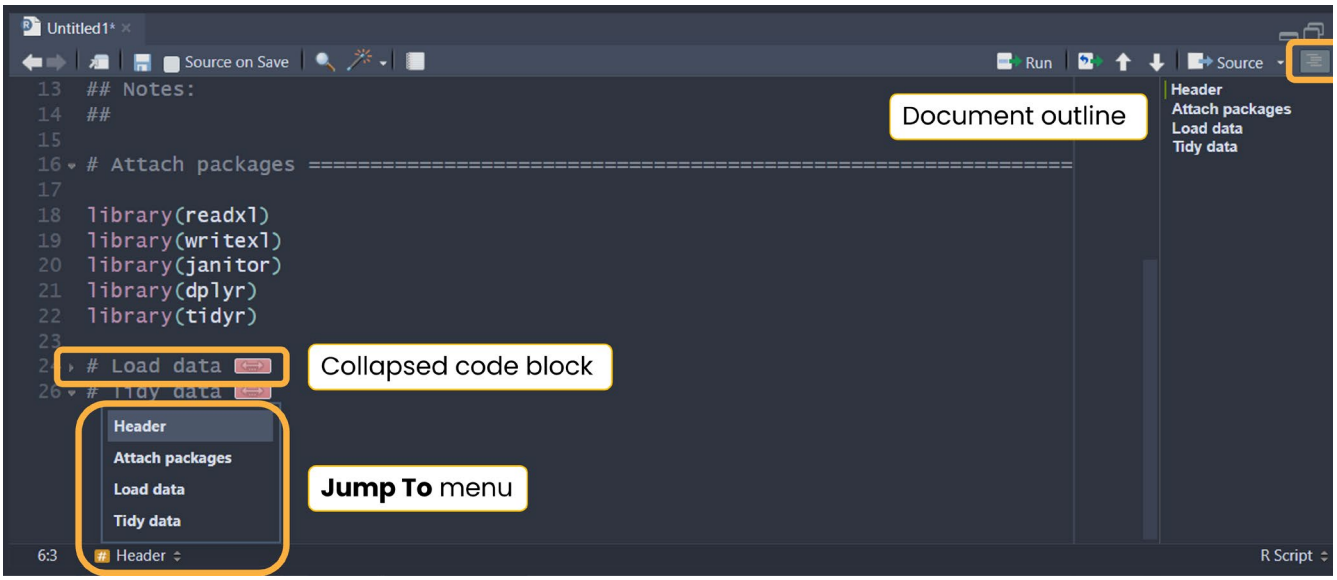
The snippet to create this section template that fills in the rest of the line with `=` was adapted from this stack overflow answer.

```
snippet end
    `r strrep("=", 84 - rstudioapi::primary_selection(rstudioapi::getActiveDocument
Context())$range$start[2])`
```

After adding the above code to your snippets, try creating a new section by typing "# Tidy data end" then pressing `Shift` + `Tab`.

`# Tidy data end<Shift+Tab>` results in:

```
# Tidy data ==========================================================================
```



9.4 Code styling

Review the Syntax chapter of the *tidyverse style guide* for a lengthy section that covers spacing, function calls, long lines, semicolons, assignments, comments, etc. Also skim through Chapter 4 Workflow: code style in *R4DS*, which highlights the opinionated "most important parts of the tidyverse style guide". Instead of rewriting all of these details and conventions into this style guide and making

us all memorize the content, we should all just use the {styler} R Package (as advised in *R4DS* Chapter 4).

{styler} is a package and RStudio Addin that formats code for you, so we can keep our coding style consistent across projects and better facilitate collaboration. We'll deviate slightly from the tidyverse style and instead use {grkstyle}. {grkstyle} is an extension package for {styler} that Garrick Aden-Buie developed based on the tidyverse style guide. I prefer {grkstyle} over the tidyverse style that {styler} defaults to mainly because of how it handles line breaks in function calls.

The below example and installation instructions are pretty much copied directly from Garrick's {grkstyle} README.

Examples

grkstyle

```
do_something_very_complicated(
    something = "that",
    requires = many,
    arguments = "some of which may be long"
)
```

styler::tidyverse_style

```
do_something_very_complicated(
  something = "that", requires = many,
  arguments = "some of which may be long"
)
```

Installation

Install {styler} and {grkstyle} with:

```
install.packages("styler")

options(repos = c(
    gadenbuie = "https://gadenbuie.r-universe.dev",
    getOption("repos")
))

# Download and install grkstyle in R
install.packages("grkstyle")
```

Set `grkstyle` as the default in {styler} functions and addins by running

```
# Set default code style for {styler} functions
grkstyle::use_grk_style()
```

or adding the following to your `~/.Rprofile`

```
options(styler.addins_style_transformer = "grkstyle::grk_style_transformer()")
```

To edit your `.Rprofile`, you can use `usethis::edit_r_profile()` to open the file in your RStudio. You may need to install the {usethis} package.

Usage

Once {styler} and {grkstyle} are installed, you can apply the style to your `.R`, `.qmd`, and `.Rmd` files using the command palette, keyboard shortcut, or addins menu.

Command palette

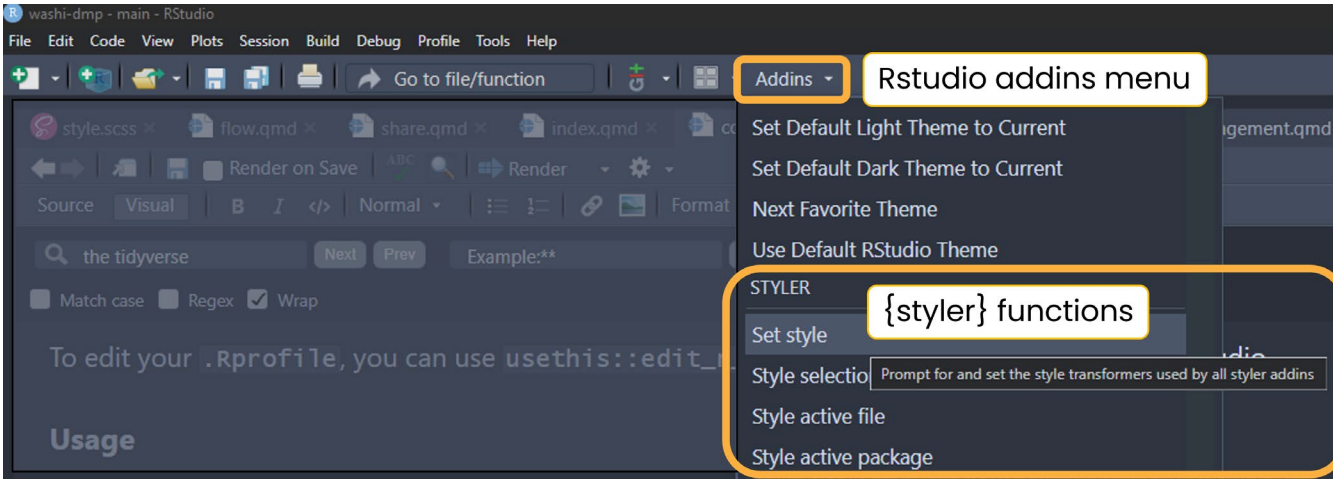
Use RStudio's **command palette** to quickly and easily access any RStudio command and see keyboard shortcuts. Open the command palette with `Cmd/Ctrl + Shift + P`, then type "styler" to see the shortcuts offered by {styler}.

Keyboard shortcuts

I often use `Cmd/Ctrl + Shift + A` to style the entire active file every time I finish a code block or section. To style just a selection, use `Cmd/Ctrl + Alt + Shift + A`.

Addins menu

You can also use the Addins menu in RStudio to style your files by clicking on a button to run the command.



References

Bryan, Jennifer. 2018. "Excuse Me, Do You Have a Moment to Talk about Version Control?" *The American Statistician* 72 (1): 20–27. https://doi.org/10.1080/00031305.2017.1399928.

Carlson, Bryan. 2021. "Data Management Plan for the R.J. Cook Agronomy Farm Long-Term Agroecological Research Site."

European Commission. 2016. "H2020 Programme Guidelines on FAIR Data Management in Horizon 2020." https://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/hi/oa_pilot/h2020-hi-oa-data-mgt_en.pdf.

Farewell, Dr Timothy S. 2018. "My Easy r Script Header Template Tim Farewell." https://timfarewell.co.uk/my-r-script-header-template/.

Harvard Medical School. 2023. "Data Management Plans." https://datamanagement.hms.harvard.edu/plan-design/data-management-plans.

Lewis, Crystal. 2023. *Data Management in Large-Scale Education Research [in Preparation]*. https://datamgmtinedresearch.com/.

U.S. Fish & Wildlife Service. 2023. "Data Management Life Cycle." https://www.fws.gov/data/life-cycle.

Whyte, Angus, and Graham Pryor. 2011. "Open Science in Practice: Researcher Perspectives and Participation." *International Journal of Digital Curation* 6 (March): 199–213. https://doi.org/10.2218/ijdc.v6i1.182.

Wickham, Hadley. 2022. *The Tidyverse Style Guide*. https://style.tidyverse.org/index.html.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 160018. https://doi.org/10.1038/sdata.2016.18.