Reproducible Workflow

by Lorraine Gaudio

Activity generated on October 05, 2025

BOISE STATE UNIVERSITY

Contents

1. Class Activity (Part One)

A reproducible workflow enables someone else to run your analysis on a different computer and obtain the same results. Today, you will package your data, containing minimal transforms, into a small, portable project using the appropriate file formats for the target audiences (R users vs. non-R users). This activity is offered as an optional opportunity to earn technical skill points.

Lesson Overview

By the end of this activity you will be able to:

1. □ Remember – common save formats and their trade-offs.
2. □ Understand – why Projects + relative paths support portability.
3. □ Apply – a minimal Lesson-6 transform and save results to data_clean/.
4. □ Analyze – object structure after saving and re-loading.
5. □ Evaluate – which format to share with R vs. non-R users.
6. □ Create – a runnable, documented mini-pipeline that another student can execute in Part 2.

Keep these goals in mind as you move through each section.

1.1 What you'll do

Part one (40 min):

1. Create a new RStudio Project and folders (optional)
2. Load a raw dataset (built-in or your own CSV)
3. Apply a minimal transform (e.g., from Lesson 6)
4. Save the cleaned dataset as .rds (for R users) and .csv (for non-R users)

Part two (preview):

We'll swap your .rds (and .csv) + 01_save_data.Rmd with a peer. Your peer will clone your structure, re-run your script, load your .rds, and verify that object names, dimensions. They'll document any issues and fixes and offer suggestion on other minimal transformations. (No action needed now.)

1. Load your peer's .rds to verify it works
2. Explore the object structure
3. Apply a minimal transform (e.g., from Lesson 6-7)

4. Save the cleaned dataset as .rds (for R users) and .csv (for non-R users)

1.2 Points Rubric

This optional activity could earn you up to 2 technical skill points!

Task	Points
Create a new RStudio Project and folders	0.0
Load a raw dataset (built-in or your own CSV)	0.25
Apply a minimal transform (e.g., from Lesson 6)	1.50
Save the cleaned dataset as .rds (for R users) and .csv (for non-R users)	0.5
Total	2

1.3 What is an .rds?

It's a single, compressed R object saved to disk (serialization). It preserves the object's classes, factors, dates, list-columns, attributes. This allows you to share data and workflows with collaborators. The purpose of this activity is to practice saving a data frame from your R session as different kinds of shareable files. This is an essential step in a reproducible R workflow.

See this quick format guide for when to use .rds, .RData, .csv, and .xlsx.

1) .rds

- Use when: you want a single object saved exactly as-is, for fast, reliable re-use inside R projects.
- Pros: preserves types/attributes; fast; compact; explicit (you assign the name on load).
- Cons: R-specific; not human-readable; one object per file.

2) .RData (created with save()/load())

- Use when: you want to snapshot many objects together.
- Pros: multiple objects; same fidelity as .rds.
- Cons: load() dumps objects into your workspace by their old names (less explicit/reproducible); R-only.

3) .csv

- Use when: you need a simple, universal text file to share with anyone (Python, Excel, Stata, etc.).
- Pros: human-readable; portable; tiny dependency surface.
- Cons: loses metadata (factors, classes); can mangle types (e.g., character vs numeric); can't store list-columns; larger than .rds for complex data.

4) .xlsx (Excel workbook)

- Use when: a stakeholder requires Excel (multi-sheet handoff, lightweight manual review).
- Pros: multi-sheet; familiar to non-coders.
- Cons: manual edits reduce reproducibility; type guessing can break things (dates, leading zeros); heavier dependencies; not ideal for scripted pipelines.

Rule of thumb: use .rds for your internal, scripted workflow; export .csv (or .xlsx only if asked) for sharing with *non-R users*; use .RData sparingly for multi-object checkpoints.

2. Mini reproducible workflow

Below is a starter pattern you can run on any machine with R + RStudio.

2.1 Setup (5 min)

Create a new RStudio Project and folders (optional), move your raw data to the correct folder, and create a new R Markdown file.

2.1.1 Make a Project & folders

Step one: Create a new RStudio Project (File > New Project > New Directory > New Project).

Step two: Name it something relevant (e.g., "save_rds_activity") within the folder of this course. This will create a new folder nested inside your course folder and a new .Rproj file.

Step three: Create folders: data, scripts, output.

```
#  Create folders
dirs <- c("data_raw", "data_clean")
invisible(lapply(dirs, dir.create, showWarnings = FALSE))
```

 This creates folders, (data_raw and data_clean) if they don't already exist, and suppresses warnings if they do. Check your file explorer or finder to verify the folders are there.

2.1.2 R Markdown

Step one: Create a new R Markdown file (File > New File > R Markdown).

Step two: Name it and save it as 01_save_data.Rmd. Save it in the Project root (same level as data_clean/).

2.2 Load Raw data (5 min)

Choose ONE path

You need to choose a "raw" dataset for this activity. You can use (A) a built-in/package dataset or (B) a CSV on your computer (place it in data_raw/).

Ideally, you will use the dataset you will be using for your signature assignment. If you do not have a dataset ready for your signature assignment, consider using a dataset from dslabs for this activity.

```
# Option A — use a package dataset
# dslabs package
library(dslabs)  # For more datasets
utils::data(package = "dslabs")$results[, "Item"]
```

```
##  [1] "admissions"
##  [2] "brca"
##  [3] "brexit_polls"
##  [4] "death_prob"
##  [5] "divorce_margarine"
##  [6] "gapminder"
##  [7] "greenhouse_gases"
##  [8] "heights"
##  [9] "historic_co2"
## [10] "mice_weights (mice_weigths)"
## [11] "mnist_127"
## [12] "mnist_27"
## [13] "movielens"
## [14] "murders"
## [15] "na_example"
## [16] "nyc_regents_scores"
## [17] "oecd (gapminder)"
## [18] "olive"
## [19] "opec (gapminder)"
## [20] "outlier_example"
## [21] "polls_2008"
## [22] "polls_us_election_2016"
## [23] "pr_death_counts (pr-death-counts)"
## [24] "raw_data_research_funding_rates"
## [25] "reported_heights"
## [26] "research_funding_rates"
## [27] "results_us_election_2012"
## [28] "results_us_election_2016 (polls_us_election_2016)"
## [29] "sentiment_counts (trump_tweets)"
## [30] "stars"
## [31] "temp_carbon"
## [32] "tissue_gene_expression"
## [33] "trump_tweets"
## [34] "us_contagious_diseases"
```

 If you choose a dataset from dslabs, you can load it with data("dataset_name").

```
#  load dataset from dslabs
data("movielens")
#   Verify
head(movielens)
```

movieId	title	year	genres	userId	rating	timestamp
31	Dangerous Minds	1995	Drama	1	2.5	1260759144
1029	Dumbo	1941	Animation|Children|Drama|Musical	1	3.0	1260759179
1061	Sleepers	1996	Thriller	1	3.0	1260759182
1129	Escape from New York	1981	Action|Adventure|Sci-Fi|Thriller	1	2.0	1260759185
1172	Cinema Paradiso (Nuovo cinema Paradiso)	1989	Drama	1	4.0	1260759205
1263	Deer Hunter, The	1978	Drama|War	1	2.0	1260759151

 I use mtcars for an example in this activity of how to use a built-in dataset.

```
#  Example "import" step (pretend mtcars is your raw data)
mtcars_df <- mtcars
#   Verify
mtcars_df[1:6 , c(1, 4)]
```

	mpg	hp
Mazda RX4	21.0	110
Mazda RX4 Wag	21.0	110
Datsun 710	22.8	93
Hornet 4 Drive	21.4	110
Hornet Sportabout	18.7	175
Valiant	18.1	105

If you choose a CSV dataset from your own computer, read it in with read.csv() or readr::read_csv(). Place the raw data file in the data_raw folder you created above. As an example of loading and external dataset, I'll load signature_assignment_example.csv from the data_raw folder.

```
# Option B — use your own CSV (put it in data_raw/)
#  Example load dataset from your computer
example_df <- read.csv(file.path("data_raw", "signature_assignment_example.csv"))
#
example_df[1:6 , c(1:2,19)]  # show first 6 rows and a few columns
```

id	study_area	humans_versus_AI_control
1871504[illegible]5	Natural Sciences (e.g., biology, chemistry, physics, astronomy, environmental science, evolutionary anthropology, neruopsychology)	3. Aquire
Su8274642	Natural Sciences (e.g., biology, chemistry, physics, astronomy, environmental science, evolutionary anthropology, neruopsychology)	2. Awareness
Su6073642	Social Sciences (e.g., sociology, cultural anthropology, social psychology)	2. Awareness
0466274[illegible]2	Social Sciences (e.g., sociology, cultural anthropology, social psychology)	2. Awareness
Su0435824	Natural Sciences (e.g., biology, chemistry, physics, astronomy, environmental science, evolutionary anthropology, neruopsychology)	2. Awareness
Su6718158	Health Sciences (e.g., medicine, nursing, pharmacy, dentistry, veterinary medicine, public health)	2. Awareness

□ The dataset loaded successfully.

2.3 Clean/transform in scripts (15 min)

Make a transformation or two to the dataset. Your example can do minimal transformations. Apply any steps you would like to practice from Lesson 6.

2.3.1 mtcars example transformations

```
# Step one: explore the data
mtcars_df[1:6 , c("mpg", "cyl")]
```

	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Hornet Sportabout	18.7	8
Valiant	18.1	6

□ I add a new column called "model" that contains the row names.

```
# □ keep model names as a column
mtcars_df[ , "model"] <- rownames(mtcars_df)
```

```
# □ Verify:
mtcars_df[1:6 , c("model", "mpg", "cyl")]
```

	model	mpg	cyl
Mazda RX4	Mazda RX4	21.0	6
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6
Datsun 710	Datsun 710	22.8	4
Hornet 4 Drive	Hornet 4 Drive	21.4	6
Hornet Sportabout	Hornet Sportabout	18.7	8
Valiant	Valiant	18.1	6

 The change was successful. rownames(mtcars_df) collects the rownames as a vertex then mtcars_df[, "model"] assigns the vertex to a new column called "model".

 I remove the row names.

```
#  remove row names
rownames(mtcars_df) <- NULL
```

```
#  Verify:
mtcars_df[1:6 , c("model", "mpg", "cyl")]
```

model	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Hornet Sportabout	18.7	8
Valiant	18.1	6

 The change was successful. The rownames(mtcars_df) <- NULL command successfully removed the row names.

2.3.2 Signature assignment example transformations

```
# Step one: explore the data
example_df[1:6, c("id", "humans_versus_AI_control")]
```

id	humans_versus_AI_control
187150405	3. Aquire
Su8274642	2. Awareness
Su6073642	2. Awareness
046627462	2. Awareness
Su0435824	2. Awareness
Su6718158	2. Awareness

id	humans_versus_AI_control

 I want to make two different types of columns for the AI Competency results section. I would like to have the numeric values in one column and the words in another column. This will make it easier to analyze.

```
# What are my options
unique(example_df$humans_versus_AI_control) # An example column
```

```
## [1] "3. Aquire"      "2. Awareness"   "4. Deepen"      "5. Create"
## [5] "1. No Awareness"
```

 Using ifelse() and overwriting values from lesson 6, I will create a new column called humans_versus_AI_control_num that contains numeric values for the humans_versus_AI_control column. I will then overwrite the values in the original column so that it only contains the words.

1) Create a numeric column with ifelse()

```
#  Create numeric column for humans_versus_AI_control
example_df[ , "humans_versus_AI_control_num"] <-
  ifelse(example_df[ , "humans_versus_AI_control"] ==
           "1. No Awareness", 1L,
  ifelse(example_df[ , "humans_versus_AI_control"] ==
           "2. Awareness",    2L,
  ifelse(example_df[ , "humans_versus_AI_control"] ==
           "3. Aquire",       3L,
  ifelse(example_df[ , "humans_versus_AI_control"] ==
           "4. Deepen",       4L,
  ifelse(example_df[ , "humans_versus_AI_control"] ==
           "5. Create",
         5L, NA_integer_)))))
```

```
#  Verify:
example_df[1:6 , c("id", "humans_versus_AI_control", "humans_versus_AI_control_num")]
```

id	humans_versus_AI_control	humans_versus_AI_control_num
187150405	3. Aquire	3
Su8274642	2. Awareness	2
Su6073642	2. Awareness	2
046627462	2. Awareness	2
Su0435824	2. Awareness	2
Su6718158	2. Awareness	2

 A new column called humans_versus_AI_control_num was successfully created.

2) □ Overwrite the original column to keep only the words.

```
# □ Overwrite original column to keep only the words
example_df[ example_df[ , "humans_versus_AI_control"] ==
            "1. No Awareness" , "humans_versus_AI_control"] <- "No Awareness"
example_df[ example_df[ , "humans_versus_AI_control"] ==
            "2. Awareness"    , "humans_versus_AI_control"] <- "Awareness"
example_df[ example_df[ , "humans_versus_AI_control"] ==
            "3. Aquire"       , "humans_versus_AI_control"] <- "Aquire"
example_df[ example_df[ , "humans_versus_AI_control"] ==
            "4. Deepen"       , "humans_versus_AI_control"] <- "Deepen"
example_df[ example_df[ , "humans_versus_AI_control"] ==
            "5. Create"       , "humans_versus_AI_control"] <- "Create"
```

```
#□ Verify:
example_df[1:6 , c("id", "humans_versus_AI_control", "humans_versus_AI_control_num")]
```

id	humans_versus_AI_control	humans_versus_AI_control_num
187150405	Aquire	3
Su8274642	Awareness	2
Su6073642	Awareness	2
046627462	Awareness	2
Su0435824	Awareness	2
Su6718158	Awareness	2

3) □ I select only the columns I want to keep for my final dataset to share.

```
# □ Select only the columns I want to keep
example_df_clean <- example_df[ , c("id", "humans_versus_AI_control", "humans_versus_AI_control_num")]
```

```
# □ Verify:
example_df_clean[1:6, ]
```

id	humans_versus_AI_control	humans_versus_AI_control_num
187150405	Aquire	3
Su8274642	Awareness	2
Su6073642	Awareness	2
046627462	Awareness	2
Su0435824	Awareness	2
Su6718158	Awareness	2

3. Save cleaned data (5 min)

We'll save the cleaned object twice: once as .rds (for R pipelines) and once as .csv (for sharing).

3.1 RDS

 file.path() builds correct paths on Windows/Mac/Linux and keeps your code portable inside an RStudio Project.

```
#  Create a file save path
rds_path <- file.path("data_clean", "dataset_clean.rds")
```

This creates a path to save the cleaned dataset as dataset_clean.rds in the data_clean folder. You can change dataset to something relevant to your dataset.

```
#  / 
saveRDS(mtcars_df, rds_path) # change mtcars_df to the name of your cleaned dataset
```

 Check your Files pane (RStudio) to verify the folders are there.

3.2 CSV

```
#  Create a file save path
csv_path <- file.path("data_clean", "dataset_clean.csv")
```

```
#  CSV for non-R users
write.csv(mtcars_df, csv_path)
```

3.3 Submission (2 min)

Upload the folowing files to Module 6 → Class Activity 6.

- 01_save_data.Rmd (the R Markdown you used for this activity)
- data_clean/dataset_clean.rds
- data_clean/dataset_clean.csv

3.4 Today you practiced:

- worked inside an RStudio Project with relative paths,
- loaded a raw dataset (package or CSV),
- performed a minimal, documented transform (Lesson 6 skill),
- saved a clean dataset for two audiences: .rds for R users and .csv for non-R users,
- verified your results by reloading/checking structure.

This gives you a portable, re-runnable mini-pipeline that your collaborator can execute in Part Two.

3.4.1 P.S. How to load rds

```
#
cars <- readRDS("mtcars_clean.rds")  # adjust file path as needed
```