Reproducibility Workflow Notes

Overview

This document summarizes the process of setting up reproducible analysis pipelines in both R and Python, following the SODA 501 course materials. The workflow involves Git version control, dependency management, logging, and structured project organization.

Project Structure

Both R and Python scripts create the same directory structure:

```
project/
├── data/
│   ├── raw/                # Original, untouched data
│   └── processed/          # Cleaned data
├── outputs/
│   ├── figures/            # Plots and visualizations
│   └── tables/             # Model summaries, coefficients
├── analysis_log.txt        # Audit trail of pipeline steps
├── renv.lock               # R dependency manifest
└── requirements.txt        # Python dependency manifest
```

Git Workflow

Initial Setup

Cloning the instructor repository required the repository root URL, not a file-specific URL:

```
# Correct
git clone https://github.com/jfedgerton/soda_501.git

# Incorrect (points to specific file)
git clone
https://github.com/jfedgerton/soda_501/blob/main/02_reproducibility/problem_set/ed
ucation_income.csv
```

Issue: Spaces in Folder Names

Problem: Navigating to folders with spaces caused errors.

```
cd Penn State
# bash: cd: too many arguments
```

Fix: Use quotes or escape the space.

```
cd "Penn State"
# or
cd Penn\ State
```

Issue: Author Identity Unknown

Problem: First commit failed because Git didn't know who was committing.

```
Author identity unknown
*** Please tell me who you are.
```

Fix: Configure Git with identity (one-time setup).

```
git config --global user.email "your.email@example.com"
git config --global user.name "Your Name"
```

Issue: Pasting in Git Bash

Problem: Ctrl+V doesn't work in Git Bash.

Fix: Use right-click and select Paste, or Shift+Insert.

Pushing to Own Repository

Since the instructor repo's origin points to their account, I had to replace it:

```
git remote remove origin
git remote add origin https://github.com/YourUsername/your-repo-name.git
git push -u origin main
```

R Workflow

Dependency Management with renv

The `renv` package creates project-isolated libraries with a lockfile recording exact package versions.

Key commands:

- `renv::init()` — Initialize (run once per project)
- `renv::snapshot()` — Update lockfile after installing packages
- `renv::restore()` — Recreate environment from lockfile

Issue: renv Cache on Wrong Drive

Problem: Default cache location was on C drive with limited space.

```
renv::paths$cache()
# "C:\\Users\\...\\AppData\\Local/R/cache/R/renv/cache/..."
```

Fix: Set a custom cache path permanently by editing `~/.Renviron`:

```
file.edit("~/.Renviron")
```

Add this line:

```
RENV_PATHS_ROOT=D:/r_workspace/renv
```

Restart R. The setting persists across sessions.

Issue: DEBUG Object Not Found

Problem: Running `logger::log_threshold(DEBUG)` before loading the package.

```
logger::log_threshold(DEBUG)
# Error: object 'DEBUG' not found
```

Fix: Load the library first.

```
library(logger)
logger::log_threshold(DEBUG)
```

Issue: Data Object Not Found

Problem: The script logs that it's loading data, but the actual `read_csv()` call was missing.

Fix: Add the read statement explicitly.

```
education_income_raw <- readr::read_csv("data/raw/education_income.csv")
```

Completed R Model Code

```
# Fit models
model_1 <- lm(income ~ education, data = education_income_clean)
model_2 <- lm(income ~ education + I(education^2), data = education_income_clean)
model_3 <- lm(log_income ~ education, data = education_income_log)

# Save summaries
writeLines(capture.output(summary(model_1)), "outputs/tables/model_1_summary.txt")
writeLines(capture.output(summary(model_2)), "outputs/tables/model_2_summary.txt")
writeLines(capture.output(summary(model_3)), "outputs/tables/model_3_summary.txt")

# Create coefficients table
coefficients_table <- bind_rows(
  tidy(model_1) |> mutate(model = "Model 1: Linear"),
  tidy(model_2) |> mutate(model = "Model 2: Quadratic"),
  tidy(model_3) |> mutate(model = "Model 3: Log Income")
)
write_csv(coefficients_table, "outputs/tables/regression_coefficients.csv")

# Save session info
writeLines(capture.output(sessionInfo()), "outputs/session_info.txt")
```

Python Workflow

Dependency Management

Python uses `requirements.txt` for dependency tracking, typically combined with virtual environments for isolation.

```
# Generate requirements file
pip freeze > requirements.txt

# Restore on another machine
pip install -r requirements.txt
```

Issue: Typo in Script

Problem: The instructor script had a typo.

```
import platformls  # Wrong
```

Fix:

```
import platform  # Correct
```

Issue: Typo in REPL

Problem: Typed `p.random.seed(123)` instead of `np.random.seed(123)`.

Fix: Use the correct alias.

```
np.random.seed(123)
```

Issue: File Not Found

Problem: Python couldn't find the CSV because the working directory wasn't set.

```
FileNotFoundError: [Errno 2] No such file or directory:
'data/raw/education_income.csv'
```

Fix: Set the working directory.

```
import os
os.chdir("D:/r_workspace/soda501/soda_501/02_reproducibility/demo")
```

Issue: Installing Packages from REPL

Problem: Needed statsmodels but was inside the Python REPL.

Fix: Either exit and run `pip install statsmodels` from terminal, or install from within Python:

```
import subprocess
subprocess.run(["pip", "install", "statsmodels"])
```

Issue: Statsmodels Version Attribute

Problem: Attempted to get version from wrong module.

```
f.write(f"Statsmodels version: {smf.__module__}\n")
# AttributeError: module 'statsmodels.formula.api' has no attribute '__module__'
```

Fix: Import the main module and use `__version__`.

```
import statsmodels
f.write(f"Statsmodels version: {statsmodels.__version__}\n")
```

Issue: pip freeze Path Syntax

Problem: Git Bash path syntax doesn't work in Windows Command Prompt.

```
cd /d/r_workspace/...  # Git Bash only
```

Fix: Use Windows path syntax in CMD/PowerShell.

```
cd D:\r_workspace\soda501\soda_501\02_reproducibility\demo
pip freeze > requirements.txt
```

Completed Python Model Code

```
import statsmodels.formula.api as smf
import statsmodels

# Fit models
model_1 = smf.ols("income ~ education", data=education_income_clean).fit()
model_2 = smf.ols("income ~ education + I(education**2)",
data=education_income_clean).fit()
model_3 = smf.ols("log_income ~ education", data=education_income_log).fit()

# Save summaries
with open("outputs/tables/model_1_summary.txt", "w") as f:
    f.write(model_1.summary().as_text())

with open("outputs/tables/model_2_summary.txt", "w") as f:
    f.write(model_2.summary().as_text())

with open("outputs/tables/model_3_summary.txt", "w") as f:
    f.write(model_3.summary().as_text())

# Create coefficients table
coefficients_table = pd.DataFrame({
    "model": ["Model 1: Linear"] * len(model_1.params) +
             ["Model 2: Quadratic"] * len(model_2.params) +
             ["Model 3: Log Income"] * len(model_3.params),
    "term": list(model_1.params.index) + list(model_2.params.index) +
list(model_3.params.index),
    "estimate": list(model_1.params) + list(model_2.params) +
list(model_3.params),
    "std_error": list(model_1.bse) + list(model_2.bse) + list(model_3.bse),
    "statistic": list(model_1.tvalues) + list(model_2.tvalues) +
list(model_3.tvalues),
    "p_value": list(model_1.pvalues) + list(model_2.pvalues) +
list(model_3.pvalues)
})
coefficients_table.to_csv("outputs/tables/regression_coefficients.csv",
    index=False)

# Save session info
with open("outputs/session_info.txt", "w") as f:
    f.write(f"Python version: {sys.version}\n")
    f.write(f"Platform: {platform.platform()}\n")
    f.write(f"NumPy version: {np.__version__}\n")
    f.write(f"Pandas version: {pd.__version__}\n")
    f.write(f"Statsmodels version: {statsmodels.__version__}\n")
    f.write(f"Timestamp: {datetime.now().isoformat()}\n")
```

Key Comparisons: R vs Python

Concept	R	Python
Package manager	renv	pip + venv
Lockfile	renv.lock	requirements.txt
Restore command	`renv::restore()`	`pip install -r requirements.txt`
Isolation	Built into renv	Requires separate venv
Data manipulation	tidyverse	pandas
Regression	`lm()`	statsmodels
Tidy model output	broom::tidy()	Manual extraction from fit object
Logging	logger package	logging module (built-in)
Random seed	`set.seed(123)`	`np.random.seed(123)`
Session info	`sessionInfo()`	Manual construction

Lessons Learned

1. **Read scripts before running** — Identify TODOs and missing pieces first.
2. **Set working directory early** — Both R and Python need to know where the project lives.
3. **Environment variables need persistence** — Session-level settings like `Sys.setenv()` reset when R restarts; use `.Renviron` for permanent changes.
4. **Load libraries before using their objects** — Constants like `DEBUG` aren't available until the package is loaded.
5. **Path syntax varies by terminal** — Git Bash uses `/d/path`, Windows uses `D:\path`.
6. **Git identity is a one-time setup** — Configure name and email before the first commit.
7. **Raw data goes in raw folder** — Manually move or copy data files to match script expectations.
8. **Commit regularly** — After completing major sections, commit work.