Assignment 5 - Part 2: Code Quality and Git (Take-Home)

AQM2 - UC3M

Spring 2026

1. Code improvement

The original script has the following problems: single-letter variable names (`x`, `m`), no comments, no constants, repeated plotting code, no assertions, poor formatting (no spaces around operators, `$` notation in `lm()`), and no diagnostics after data transformations.

Below is the improved version saved as `improved_script.R`. Each of the seven required improvements is marked with a comment.

```r
# setwd("~/path/to/assignment5")
options(stringsAsFactors = FALSE)

# ============================================================
# Improved analysis script
# ============================================================

# ============================================================
# Load packages
# ============================================================

library(modelsummary)

# ============================================================
# Constants (Improvement 2: constants at top)
# ============================================================

start_year = 2000
end_year = 2020

# ============================================================
# Load and clean data
# ============================================================

# (Improvement 1: meaningful variable name instead of 'x')
panel = read.csv("mydata.csv")

# (Improvement 5: assertion after loading)
if(nrow(panel) == 0) stop("Dataset is empty")
```

```r
expected_cols = c("year", "outcome", "gdp", "pop", "education", "health")
missing_cols = setdiff(expected_cols, names(panel))
if(length(missing_cols) > 0) stop("Missing columns: ", paste(missing_cols, collapse = ", "))

# Filter to analysis period using constants
panel = panel[panel$year >= start_year & panel$year <= end_year, ]

# (Improvement 7: diagnostic print after filtering)
cat("Rows after filtering:", nrow(panel), "\n")

# ============================================================
# Estimate models
# (Improvement 6: spaces around operators, data = argument)
# ============================================================

# (Improvement 1: meaningful model names)
m_base = lm(outcome ~ gdp + pop, data = panel)
m_educ = lm(outcome ~ gdp + pop + education, data = panel)
m_full = lm(outcome ~ gdp + pop + education + health, data = panel)

# Print summary table
modelsummary(
  list("Base" = m_base, "Education" = m_educ, "Full" = m_full),
  stars = TRUE,
  gof_map = c("r.squared", "nobs"))

# ============================================================
# Scatter plots
# (Improvement 4: function to avoid repeating pdf/plot/dev.off)
# ============================================================

save_scatter = function(data, xvar, yvar, filename) {
  pdf(filename, width = 7, height = 5)
  plot(data[[xvar]], data[[yvar]],
    xlab = xvar, ylab = yvar,
    pch = 16, col = rgb(0, 0, 0, 0.5))
  dev.off()
  cat("Saved:", filename, "\n")
}

save_scatter(panel, "gdp", "outcome", "fig_gdp.pdf")
save_scatter(panel, "education", "outcome", "fig_education.pdf")
```

Summary of all seven improvements:

1. **Meaningful variable names**: `panel` instead of `x`; `m_base`, `m_educ`, `m_full` instead of `m`, `m2`, `m3`.
2. **Constants at top**: `start_year` and `end_year` defined once, used in the filter.
3. **Comments with section dividers**: Each section has a header using **# ============** dividers explaining what

the code does.

4. **Function for repeated pattern**: `save_scatter()` replaces the duplicated `pdf()/plot()/dev.off()` blocks.
5. **Assertion after loading**: Checks that the dataset is non-empty and that expected columns exist.
6. **Proper formatting**: Spaces around =, ~, +; consistent indentation; `data = panel` argument in `lm()` instead of `x$outcome ~ x$gdp`.
7. **Diagnostic print**: `cat("Rows after filtering:", nrow(panel), "\n")` after subsetting.

2. Git practices

a–b) The `assignment5` folder should be part of the course repository. Make at least 3 commits, each corresponding to a logical unit of work.

c) Example commit messages (each completes "This commit will…"):

- `Add folder structure and README for assignment 5`
- `Create analysis script with regression models`
- `Add plots script with scatter plot`
- `Add Makefile to automate pipeline`
- `Add improved script and .gitignore`

d) Example `.gitignore`:

```
# Generated output files
*.pdf
*.aux
*.log
analysis/output/
plots/output/

# System files
.DS_Store
```

This `.gitignore` prevents generated outputs (which can be reproduced by running `make`) and system files from being tracked. The key principle is that anything that can be regenerated from the source code should not be committed.

e) Push to GitHub with `git push origin main`.

f) List the commit messages used at the end of the `README.md`.

3. Reflection

Example answers (students should write their own):

a) The most useful practice was organizing a project into separate folders for data, analysis, and plots, with a Makefile tying them together. This makes it clear what each script does and ensures that outputs can be reproduced with a single command, rather than manually running scripts in the right order.

b) For the final project, I would set up the same folder structure from the beginning: a `data/` folder for raw data, separate scripts for cleaning, analysis, and visualization, and a Makefile to run the full pipeline. I would also define constants at the top of each script and use assertions to catch data problems early.

c) I recognized the habit of using single-letter variable names like `x` and `m` when working quickly, and not adding comments because the code "makes sense to me right now." The exercise showed that even a short script becomes hard to read without meaningful names and section dividers.