Lecture 1 — Introduction & Tools

Data & Code Management: From Collection to Application

Samuel Orso

2025-09-25

Welcome!

- Course: **Data & Code Management: From Collection to Application**
- Time & place: **Thursdays 9:00–12:00**
- Communication: **Slack (class workspace)**

- Materials: GitHub repo & website
- Grading: participation, homeworks, and final project

> "Reproducibility is job #1 for modern data science."
> — everyone who has ever lost a script

Note

Today's goals 1. Understand course scope & expectations
2. See why **reproducibility** matters
3. Meet the core **toolchain** (R, Python, SQL, GitHub, Markdown/Jupyter/Quarto)
4. Try small, focused **exercises**

Agenda

1. Course overview & evaluation

2. Reproducibility in analytics

3. Core tools & workflows

4. Mini-exercises (static snippets)

Tip: Slides include short activities you can try during/after class.

Course Overview

- Orientation: **hands-on data & code practices** for analytics
- You will:
  - write clean **R/Python** code
  - query data via **SQL**
  - document with **Markdown/Jupyter/Quarto**
  - use **Git/GitHub** (branches, PRs, issues)
  - deliver a **reproducible project** (group)
- We value: clarity, collaboration, curiosity

Tip

Success checklist - Commit early, commit often - Make small, reversible changes - Automate what repeats - Document decisions, not only code - Prefer scripts & notebooks over manual clicks

---

Expectations & Evaluation

- **Participation:** engaged presence, Slack questions/answers
- **Homeworks:** short, targeted (programming, SQL, tooling)
- **Project:** real-world style, **reproducible** deliverable
- **Academic integrity:** cite sources, no copy-paste answers

Tools allowed: R, Python, SQL, Quarto, GitHub, AI copilots (with provenance & verification).

> **!** Important
>
> **AI policy (short)**
> Use AI to brainstorm, outline, or lint code. **Own** the result: verify outputs, write your tests, and **document** AI assistance (what, why, where).

Reproducibility: Why it matters

Symptoms of non-reproducible work (raise your hand if you've seen these):
- "It works on my machine."
- "I changed nothing and it broke."
- "Which file is the final_final_v3.R?"

Principles
- Deterministic environments
- Versioned code **and** data contracts
- Scripts, not clicks
- Single-source of truth (parameters, config)

- Literate programming (Markdown/Quarto)
- Automated checks (CI later in course)

Minimal Project Structure

```
project/
  data/              # raw/ and processed/ (never ove
  R/ or src/         # functions, modules
  notebooks/         # exploratory analysis
  reports/           # Quarto/Markdown outputs
  tests/             # unit tests
  renv/ or .venv/ # R or Python environment
  .gitignore
  README.md
```

> **Note**
>
> **Exercise (2 thinking):**
> What would you add for your domain (e.g., `sql/`, `fig/`, `configs/`)? Jot down 2 items.

Tooling Map

- **R** (tidyverse, data.table) & **Python** (pandas, polars)
- **SQL** for data retrieval/joins/aggregations
- **Git** + **GitHub** for versioning & collaboration
- **Markdown/Jupyter/Quarto** for literate workflows
- Optional helpers: **make**, **pre-commit**, **linters**

R snippet

```r
# Vectorized transform
library(dplyr)
set.seed(42)
df <- tibble(x = rnorm(5), y = rnorm(5))
df |>
  mutate(z = x + y, grp = if_else(z > 0, "pos", "n
```

Python snippet

```
import pandas as pd
import numpy as np
rng = np.random.default_rng(42)
df = pd.DataFrame({"x": rng.normal(size=5), "y": r
df.assign(z=lambda d: d.x + d.y,
          grp=lambda d: np.where(d.z > 0, "pos", '
```

SQL Refresher (we'll go deeper later)

```
-- Top customers by revenue
SELECT c.customer_id, SUM(o.amount) AS revenue
FROM customers c
JOIN orders o USING (customer_id)
WHERE o.order_date >= DATE '2025-01-01'
GROUP BY c.customer_id
ORDER BY revenue DESC
LIMIT 10;
```

Tip

Tip: Keep SQL in `.sql` files and load/parametrise from R/Python for reproducibility.

Git in 6 commands

```
git status
git add -A
git commit -m "Explain what/why, not how"
git pull --rebase
git push
git switch -c feature/your-topic    # create a feat
```

Note

Activity (think-pair-share):
What makes a good commit message? Write one for "fixed weird bug in script" that would help your future self.

Literate Programming with Quarto

- Write text + code together
- Render to HTML/PDF/slides/reports
- Parametrized reports & caching
- Works with **R** and **Python**

````
::: {.cell}

```{.r .cell-code}
plot(mtcars$wt, mtcars$mpg)
```

:::
````

````
Render:
```bash
quarto render report.qmd
````
````

> **!** Important
>
> **House rule:** every analysis step appears in a script/notebook—no manual spreadsheet edits.

Environments (determinism)

R — renv

```r
install.packages("renv")
renv::init()
renv::snapshot()    # lock versions
renv::restore()     # reproduce elsewhere
```

Python — venv + requirements

```bash
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
pip freeze > requirements.txt
```

> **Note**
>
> **Exercise (1):** List one package you rely on in R and in Python. Why lock its version?

Data Contracts & File Hygiene

- Never overwrite **raw/** data
- Validate schemas (columns, types, keys)
- Record data provenance (source, timestamp)
- Use `.gitignore` to avoid committing large/secret files

```
# data & local env
data/raw/*
!.gitkeep
.venv/
renv/library/
*.sqlite
*.parquet
```

Quick Wins You Can Adopt Today

- Create a project with folders from the template earlier
- Initialize **Git** and push to **GitHub**
- Set up **renv** or **.venv**
- Convert one analysis to **Quarto**

> Tip
>
> **Stretch goal:** Add a small test (R `testthat` or Python `pytest`) for a helper function you wrote.

Mini-Exercise 1 (R)

Objective: Write a function and test it quickly.

```r
# R: rolling mean for a numeric vector
roll_mean <- function(x, k = 3) {
  stopifnot(is.numeric(x), k >= 1, k == as.integer
  stats::filter(x, rep(1/k, k), sides = 2)
}

# quick check
x <- 1:7
roll_mean(x, k = 3)
```

Discussion: How would you handle NA edges?

Mini-Exercise 2 (Python)

Objective: Clean a small dataset and compute a grouped metric.

```python
import pandas as pd
df = pd.DataFrame({
    "team": ["A","A","B","B","B"],
    "score": [10, None, 9, 12, 13]
}).assign(score=lambda d: d.score.fillna(d.score.m
df.groupby("team", as_index=False)["score"].mean()
```

Discussion: Where would assertions / schema checks go?

Mini-Exercise 3 (SQL)

Objective: Translate a business question into SQL.

> *"Which products grew the most month-over-month in 2025?"*

```sql
WITH monthly AS (
  SELECT product_id,
         DATE_TRUNC('month', order_date) AS month,
         SUM(amount) AS revenue
  FROM orders
  WHERE order_date >= DATE '2025-01-01'
  GROUP BY product_id, DATE_TRUNC('month', order_d
),
growth AS (
  SELECT product_id, month,
         revenue,
         LAG(revenue) OVER (PARTITION BY product_i
  FROM monthly
)
SELECT product_id, month, revenue, prev_revenue,
       (revenue - prev_revenue) AS delta
FROM growth
WHERE prev_revenue IS NOT NULL
ORDER BY delta DESC
LIMIT 10;
```

Collaboration Rituals

- Branch → small PR → peer review → merge
- Use **Issues** with labels ("bug", "enhancement", "question")
- Templates: `PULL_REQUEST_TEMPLATE.md`, `ISSUE_TEMPLATE.md`
- Document **decisions** in `CHANGELOG.md`

> **Note**
>
> **Activity:** In pairs, outline a PR description for adding a new `utils/plot.R` with one function and one test.

Common Pitfalls & How to Avoid Them

- Undocumented notebooks → add titles, goals, outputs

- Hidden state (globals) → pass parameters explicitly
- One giant script → split into modules
- No seeds → set seeds where randomness matters
- Unpinned packages → lock versions

Literate Tools — Markdown, Jupyter, Quarto (curated)

What is RMarkdown?

- RMarkdown: `R` + `markdown`
- `markdown` contrasts `markup` languages (e.g. HTML) which require syntax that can be quite difficult to decipher for the uninitiated
- RMarkdown is a framework that provides a literate programming format for data science.

- **Literate programming**: programmers add narrative context with code to produce documentation for the program simultaneously.
- **Reproducible research**: the whole process (collecting data, performing analysis, producing output,...) can be reproduced the same way by someone else.

Is there a reproducibility crisis?

What is RMarkdown?

In a nutshell, R Markdown stands on the shoulders of knitr and Pandoc. The former executes the computer code embedded in Markdown, and converts R Markdown to Markdown. The latter renders Markdown

to the output format you want (such as PDF, HTML, Word, and so on). .right[– R Markdown: The Definitive Guide]

Git & GitHub — Workflows & Tips (curated)

GitHub

Motivation

- When working on a project, there are usually different people working on the same file/folder
- You want to avoid sending each modification by email

- You could use dropbox/google drive and the likes but it is good practice to keep track of modifications and have a platform to plan and discuss changes

---

Motivation

GitHub allows you: - record the entire history of a file; - revert to a specific version of the file; - collaborate on the same platform with other people; - make changes without modifying the main file and add them once you feel comfortable with them.

What's Next

- **Next lecture:** Programming foundations (R & Python)
- **Before next time:** ensure you can

  1. clone a GitHub repo,
  2. create a branch & commit,
  3. render a Quarto `.qmd` to HTML,
  4. set up `renv` **or** `.venv`.

> **Tip**
>
> If stuck: ask on **Slack**—show error, steps tried, and minimal example.

Q&A

Thanks!
Optional: After class, try converting one old analysis to **Quarto** and push it to GitHub with a short README.