Week 1: Getting Started

August 26, 2026

THE POINT OF THIS COURSE

Data is not information. A table of numbers is data. A chart that makes someone say "oh — *that's* what's going on" is information. This course is about making that translation, over and over, and defending the choices you made along the way.

Your prior course (P4A) taught *computational literacy* — how to make R do things. This course teaches *data literacy* — how to decide what's worth doing, and how to tell whether the result is trustworthy.

Exploration is a loop, not a pipeline

1. **Encode** the data (import it, get it into R).
2. **Reshape** it into the form the question needs.
3. **Plot it fast and ugly** to see what's there.
4. **Look again** — highlight the comparison that actually matters.
5. **Polish**: direct labels, drop chart junk, fix the title.
6. Discover a new question → go back to step 2.

Nobody gets to step 5 on the first pass. The rough plot at step 3 is not a failure — it's the tool that tells you what to do next.

> **The title of a chart is an argument.** "Engagement scores by school type" is a label. "County general-ed classes have surprisingly low engagement" is a finding. Aim for the second one.

HOW THE COURSE WORKS

Item	Weight	Notes
Participation	7%	Attendance is taken
Weekly HW	11%	Weekly, lowest dropped
Quizzes	10%	5 total, lowest dropped, 10 min
Mini Projects (3)	27%	9% each, individual
Final Project	35%	Proposal 6, progress 6, report 17, presentation 6
Final Interview	10%	Individual

Quizzes happen at the *start* of class, roughly every other week. Make-ups only for excused absences — don't be late. They exist because of the *retrieval effect*: you have to practice remembering things or you won't remember them.

Final project teams: undergrads in teams of 3–4, grads in teams of 2. Proposal Sep 20 · Initial report Nov 1 · Final report Dec 6 · Presentations Dec 8.

Late work: 3 late days for the semester, no questions asked, but no more than 2 on any single assignment. Special cases — talk to me.

Using AI in this class

We will use AI tools heavily this semester. The policy is short:

- LLMs are good. Sometimes they are terrible.
- **I grade what you submit. It should not suck.**
- Never submit code you haven't run.
- Never submit something you don't understand. If the AI wrote it, ask it to explain *why* it works — then check that the explanation is true.

Copying is fine; stealing is not. Reverse-engineering someone's approach is how you learn. Passing off their work as yours is not.

Software to install

Slack (turn notifications **on**) · R · Positron · Posit Cloud (free account) · Quarto · GitHub Desktop. All linked from the course software page.

QUARTO: ANATOMY OF A QMD FILE

Three parts, in this order:

1. **The header** (YAML between `---` fences) — the document's settings.
2. **Markdown text** — the narrative.
3. **R code chunks** — the analysis.

Rendering runs the code fresh, top to bottom, in a clean session, and stitches the results into the output. That's what makes the document *reproducible*: if it renders on your machine, it renders on mine.

```
---
title: Your title
author: Author name
toc: true
format:
  html:
    theme: united
---
```

Swap `format:` to change the output: `html`, `pdf` (needs LaTeX — run `tinytex::install_tinytex()` once), or `docx`.

Chunk options

Options go at the top of a chunk with the `#|` comment:

Option	Effect
(default)	Show code **and** output
`echo: false`	Output only — hide the code
`eval: false`	Code only — don't run it
`include: false`	Run it, show nothing

A `setup` chunk at the top with `include: false` is where you load libraries and set global defaults with `knitr::opts_chunk$set()`.

Markdown, in ten seconds

Type this	To get
`_italic_`	*italic*
`**bold**`	**bold**
`~~strikethrough~~`	~~strikethrough~~
`` `code` ``	`code`
`#` through `######`	headings, level 1 to 6
`- item`	a bullet list
`1. item`	a numbered list
`[text](https://url)`	a link

Bookmark https://commonmark.org/help/ — and when you have ten spare minutes, do the tutorial there.

GETTING DATA INTO R

1. **Open the project *folder* in Positron.** Do not double-click a `.R` or `.qmd` file — that starts R in the wrong place and every path breaks.
2. **Build paths with `file.path()`**, never with `"C:/Users/..."`. Paths are relative to the folder you opened, and `file.path()` joins them correctly on both Mac and Windows:

```
path <- file.path("data", "file.csv")
```

3. **Import with the function that matches the file type:**

File type	Function	Library
`.csv`	`read_csv()`	**readr**
`.txt`	`read.table()`	**utils**
`.xlsx`	`read_excel()`	**readxl**

Messy files need arguments: `skip = 5` to jump over preamble rows, `header = FALSE` plus `names(df) <- c(...)` when there's no header row, `sheet = "..."` to pick an Excel sheet. **Always `glimpse()` right after importing** — that's where you catch a numeric column that came in as text.

What a data frame is

- **Columns are vectors** — every value in one column is the same type. Pull one out with `df$colname` or `df[, 2]`.
- **Rows are observations.** `df[1, ]` is the first one, `df[1:2, ]` the first two.
- First questions for any new dataset: how many rows and columns? What type is each column? What does *one row* represent?

WRANGLING WITH DPLYR

Chain steps with the pipe `%>%`, which reads as "…and then…":

```
data %>%
  filter(race == "Elf") %>%
  select(-race)
```

Verb	What it does
`select()`	Pick columns by name (`-col` drops one)
`filter()`	Keep rows matching a condition
`arrange()`	Sort rows (`desc()` for descending)
`mutate()`	Create or modify columns
`summarize()`	Collapse to summary values

Conditions for `filter()`

Operator	Keeps rows where the value is…
`> >= < <=`	greater / less than
`== !=`	equal to / not equal to
`%in% c(1, 4)`	one of a set of values
`!is.na(x)`	not missing

Chain conditions together with `&` (both must be true) or `|` (either one). Use `==` for comparison, never `=`. Inside `mutate()`, `ifelse(condition, value_if_true, value_if_false)` handles two-way logic.

VISUALIZING WITH GGPLOT2

A plot is built in **layers**, joined with `+`:

1. **Data** — `ggplot()` sets the dataset.
2. **Aesthetic mapping** — `aes()` maps variables to visual properties (`x`, `y`, `color`, `fill`, `size`).
3. **Geometries** — `geom_point()`, `geom_col()`, `geom_line()`, `geom_smooth()`: what the data becomes on the page.
4. **Labels** — `labs()` for axes, legend, and title.
5. **Theme** — `theme_bw()`, `theme_minimal()`, `theme_classic()`, `theme_void()`.

```
mpg %>%
  ggplot(aes(x = displ, y = hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE) +
  labs(
    x = "Engine displacement (liters)",
    y = "Highway fuel economy (mpg)",
    title = "Bigger engines, worse mileage"
  ) +
  theme_bw()
```

Watch the two `+` vs `%>%` rules: **`%>%` pipes data between functions, `+` adds layers to a plot.** Mixing them up is the single most common ggplot error.

Aesthetics set *inside* `aes()` map to a variable; set *outside* `aes()` they're a fixed value — `geom_point(aes(color = class))` colors by class, while `geom_point(color = "blue")` makes every point blue.

ERRORS YOU WILL HIT THIS WEEK

Message	Usual cause
`could not find function`	The library isn't loaded — `library(tidyverse)`
`object 'x' not found`	Typo, or you ran chunks out of order
`cannot open file`	Wrong path — use `file.path()` and open the *folder*
`unexpected symbol`	Missing comma, quote, or paren on the line above
Plot never appears	You used `%>%` where a `+` belongs
Numbers came in as text	Preamble rows in the file — `skip = n`

When you get stuck, read the error out loud. It usually names the thing that's wrong. If it doesn't, paste the *whole* message — not a paraphrase — into Slack or into your AI tool.

GETTING HELP, AND HOW TO SUCCEED

- **Slack** is the front door for questions. Turn notifications on, and ask in the public channels — someone else has the same question.
- **Meet with the tutor.** Lola Nurullaeva is the course GTA.
- **Schedule a meeting** with Prof. Helveston (Mon / Tue / Fri).
- **GW Coders** for general R help.

The students who do well in this course all do the same five things: show up and participate, start assignments early and read them *carefully*, do the reading *before* class, sleep and take breaks, and ask for help long before they're desperate.

BEFORE NEXT CLASS

- Install everything on the software page — R, Positron, Quarto, GitHub Desktop, Slack.
- Add your **GitHub username** (not your GW netID) to the class sheet.
- Sign up for a team using the team-formation sheet.
- Skim the course schedule; it's the best starting point for everything in this course.
- Do the Week 1 HW.

You should be able to:

- Open a project folder in Positron, render a `.qmd`
- Import a csv / txt / xlsx file, describe what one row of a data frame represents
- Use the five dplyr verbs (`select`, `filter`, `arrange`, `mutate`, `summarize`) with data in R.
- Build a ggplot from data → aes → geom → labs → theme.
- If any of those are shaky, that's what Slack / office hours are for.