Week 1: Getting Started

August 26, 2026

THE POINT OF THIS COURSE

Engineers routinely design things nobody wants. The fix is not better intuition about customers — it's data. This course is about measuring what people actually value, in units you can put into a design decision: how much demand falls when price rises, how much more someone will pay for longer battery life, how a new product would fare against what's already on the shelf.

Every one of those answers depends on knowing *what people want*. Asking people directly doesn't work — ask what features they want and they'll say **all of them**, at no extra cost. So instead we watch them *choose*, and infer preferences from the trade-offs they make.

The conjoint approach, in one page

Show people a series of choices between products described by their attributes (price, brand, battery life, signal quality…). Each choice forces a trade-off, and the pattern of trade-offs across many respondents identifies how much each attribute is worth.

1. **Model utility** — the value person *n* gets from option *j*: $u_j = \beta_1 \text{price}_j + \beta_2 \text{brand}_j + \beta_3 \text{battery}_j + \varepsilon_j$
2. **Assume the random part** ε_j is iid extreme value.
3. **That gives a choice probability** (the logit model): $P_j = e^{\beta' x_j} / \sum_k e^{\beta' x_k}$
4. **Estimate the β's** by maximum likelihood — find the coefficients that make the choices people actually made the most likely ones to have observed.

> The β coefficients are the whole point. They convert "what people picked" into a number per attribute — and once you have them, everything else in this course is arithmetic on top of them.

What you do with the estimates

- **Willingness to pay (WTP).** With $u_j = \beta' x_j + \alpha p_j + \varepsilon_j$, the WTP for an attribute is $\omega = \beta/(-\alpha)$ — dividing an attribute's value by the (negative) value of price puts it in dollars. *"Respondents are willing to pay $XX for a 20% longer battery."*
- **Market simulation.** Plug in $\hat{\beta}$ and a set of competing products to predict market shares, price sensitivity curves, and how share moves as you change a design.

That's the arc of the semester: design a survey → field it → estimate a model → simulate the market → make a recommendation.

HOW THE COURSE WORKS

Item	Weight	Notes
Participation	5%	Attendance is taken
Homework	30%	13 assignments, lowest dropped
Quizzes	10%	6 total, lowest dropped, 10 min
Final Exam	10%	In class, hand-written, Dec 2
Semester Project	45%	Teams of 3–4

Homework is readings, recorded lectures, practice problems, and a short reflection. Due **11:59pm Monday** before class. Graded for completion — I'm looking for engagement, not perfection.

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.

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.

Project deliverables (45%)

Deliverable	Weight	Due
Project Proposal	5%	Sep 21
Survey Plan	5%	Sep 30
Pilot Survey	5%	Oct 14
Pilot Analysis	5%	Nov 2
Final Survey	5%	Nov 16
Final Analysis Report	15%	Dec 7
Final Presentation	5%	Dec 9

Teams of 3–4. The goal is to assess the market viability of a new technology or design and recommend the best design choices for a target market. **It starts now** — start thinking about what you want to study.

Prerequisites

This course assumes prior exposure to probability, multivariable calculus, linear algebra, and regression. Not sure? Take the self assessment.

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.
- Use the approach I teach: **agentic workflows**, starting next week.

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 (notifications **on**) · R · Positron · Quarto · GitHub Desktop · Claude Code. All linked from the course software page.

WHY R (AND NOT PYTHON)?

Marketing analytics grew out of **statistics**, not machine learning, and the choice-modeling tools we need (`logitr`, `cbcTools`, `surveydown`) live in R. Python is a general-purpose language that wasn't built for data analysis. Both are useful and you should probably learn Python too — but this semester is R.

R BASICS

Objects and assignment

```
x <- 42         # "<-" is preferred over "="
y <- 42
x + y
```

- Shortcut for `<-`: **Option + -** (Mac) / **Alt + -** (Windows).
- **Always put spaces around the assignment arrow** — `x<-2` could just as easily read as `x < -2`.
- R is **case sensitive**: `number`, `Number`, and `numbeR` are three objects.
- Use meaningful names in `snake_case` or `camelCase`: `car_speed_mph`, not `x`.
- `objects()` lists what's in your environment; `rm(x)` removes one, `rm(list = objects())` clears everything.

Data types

Type	What it holds
character	Strings, in quotes
integer	Whole numbers (`42L`)
numeric	Numbers with decimals
factor	A categorical variable
logical	`TRUE` or `FALSE`

Convert with `as.numeric(x)`, `as.character(x)`, `as.integer(x)`.
Check with `is.numeric(x)`, `is.character(x)`, `is.integer(x)`.
Comparisons return logicals: `==`, `!=`, `<=`, `>=`.

Vectors

A vector is a series of elements **of the same type**, built with `c()` ("concatenate"):

```
x <- c(1, 1, 2, 3, 5, 8, 13)
x[2]        # second element
x[2:3]      # second and third
x[c(1, 4)]  # first and fourth
```

Useful companions: `length()`, `sort()`, `order()`, `min()`, `max()`, `mean()`, `range()`, `summary()`. Build sequences with `seq(1, 10, by = 2)` and repeat values with `rep(x, times = 2)` or `rep(x, each = 2)`. Arithmetic on two vectors happens **element by element**.

Data frames

Like a spreadsheet: columns of vectors, and **we will use these constantly.**

Function	What it tells you
`head()` / `tail()`	First / last 6 rows
`str()`	The type of every column
`dim()`, `nrow()`, `ncol()`	How big it is
`names()`, `colnames()`	Column names

Pull out a column with `df$colname` or `df[, 1]`. Unlike a vector, different columns can hold different types. Try these on the built-in `mtcars` and `iris`.

Functions, loops, and if statements

```
simpleSum <- function(x, y) {
  value <- x + y
  return(value)
}

for (i in seq(1, 10)) { print('yay') }

if (value > 10) {
  print('yay')
} else {
  print('nope')
}
```

PACKAGES

There are **>20,000 packages on CRAN**. Two separate steps, and mixing them up is the most common beginner error:

Step	Code	How often
Install	`install.packages("name")`	**Once**, ever
Load	`library(name)`	**Every** session

Note the quotes: `install.packages()` needs them, `library()` doesn't. Installing is buying the lightbulb; loading is switching it on. Restart R and the lightbulb goes off — `wiki_randomfact()` errors again until you re-run `library()`.

You don't always have to load the whole library. `packagename::functionname()` reaches one function directly — `wikifacts::wiki_randomfact()` works with no `library()` call at all.

STAYING ORGANIZED

1. **Save your code in `.R` files.** Anything typed only into the console is gone when you close R.
2. **Keep all related work in a folder** — code, data, and outputs together.
3. **Always open the *folder* in Positron** (**File › Open Folder…**), never double-click a `.R` file. Open the wrong thing and every file path breaks.

We use **Positron** this semester, not RStudio.

ERRORS YOU WILL HIT THIS WEEK

Message	Usual cause
`could not find function`	The package isn't loaded — `library(name)`
`object 'x' not found`	Typo, or wrong case (`Number` ≠ `number`)
`install.packages()` errors	The package name needs quotes
`unexpected symbol`	Missing comma, quote, or paren — often the line above
`non-numeric argument to binary operator`	Doing math on a character column
`cannot open file`	You opened a file instead of the folder in Positron

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 — ask in the public channels, since someone else has the same question.
- **Schedule a meeting** with Prof. Helveston (Mon / Tue / Fri).
- **GW Coders** for general R help.

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

BEFORE NEXT CLASS

- Install everything on the software page, then run the course package install script.
- Add your **GitHub username** (not your GW netID) to the class sheet — you need this to get your course repo.
- Sign up for a team on the team-formation sheet.
- Skim the course schedule and the project overview.
- Do HW 1.

You should be able to:

- Open a folder in Positron and save your work in a `.R` script.
- Create objects, vectors, and inspect a data frame's size, names, and types.
- Explain the difference between installing and loading a package.
- Say why we infer preferences from *choices* rather than from asking directly.

If any of those are shaky, that's what Slack / office hours are for.