BUSN 5000 Project Guide

Chris Cornwell and Abbi Cormier

August 10, 2026

Table of contents

Introduction

The Project is a *summative* assignment in which you draw on key course concepts to learn about an empirical relationship and document what you learn. You will use R and R Markdown to conduct the analysis and report your findings, "knitting" the two together in a slide deck.

What the project is about

The project involves a nontrivial examination of an important persistent feature of the US labor market, namely the difference in average pay between women and men. You often hear this expressed in terms like: "Women earn 78 cents for every dollar a man earns." Your project will use recent Current Population Survey (CPS) data to document, quantify, and attempt to explain the pay gap. In Part I of the course, we will be pointing you to background material to anchor your analysis, but for now you might start with the work of Claudia Goldin.

Why we assign it

We assign it for two reasons. First, there are workflow principles to establish. How should you organize project files and think about the data pipeline so that your analysis is *reproducible*? This question cannot be adequately addressed through standard homework assignments. You need the experience of carrying out a meaningful empirical analysis from data acquisition to a beautiful deliverable that communicates your findings. Second, there are facts about the labor market you should know. How and why does pay typically vary over a career for women and men? The project walks you through the basic steps to answer such questions using the official monthly source for labor market information.

How to use this guide

This guide is your one-stop shop for how to carry out the project. It covers:

1. How to set up your computer for the project work
2. How to complete the Pre-project Exercise and why we require it
3. How to complete the Main Project
4. How to complete the Progress Check and why we offer it
5. How to submit your work to Gradescope
6. How to recognize and correct common errors

When you view the guide on a computer, you will see a left pane that toggles the major sections and a right pane that helps you navigate within a section.

We recommend reading through the entire guide once before you start the project.

Key dates

The deadline for submitting each project-related assignment is *11:59 PM* on the due date. *Late submissions will not be accepted.*

- **Pre-project Exercise due:** Wed, Aug 26
- **Optional Progress Check:** Fri, Oct 2
- **Main Project due:** Fri, Nov 13

If you find a section that's wrong, unclear, or out of date, reach out to the teaching team — see Getting help in the Common Errors chapter for how to contact us.

What about AI?

The truth is we can one-shot this project with Claude Code or ChatGPT's Codex and produce a fully format-compliant deliverable that collects all the points. We know this because we understand the workflow principles the project teaches you (and the agents follow) and we have the domain knowledge to *verify* the analysis output.

AI tools are permitted as learning and coding partners. You remain responsible for directing their work, reviewing every change, checking every result, and understanding the analysis you submit. A one-shot result that you cannot verify or explain does not satisfy that responsibility.

The AI resources chapter provides an overview of the state of play in AI as of August 2026.

Exam accountability

However you produce your slide deck, there will be accountability for your project work and findings on Exam 2.

1Setup

Your project work starts here. Setup involves:

- Installing R and RStudio
- Installing the required R packages
- Creating the required directory structure
- Downloading the file packs and sorting their contents into the right places
- Creating an "R Project" (`.Rproj` file) to govern your project work
- Successfully running the setup-check script

Setup activities typically take 30 minutes to an hour.

> **!** "My project won't knit"
>
> Don't rush this process. Almost every "my project won't knit" message we get traces back to a setup problem. Make sure you pass the setup check before you move on.

1.1 Install R and RStudio

If you haven't already, Install R and then install RStudio. Below are step-by-step guides for Windows and macOS.

1.1.1 Windows

Step 1: Install R

1. Go to the CRAN R Project website.
2. Click on "Download R for Windows".
3. Click on "base" and then select the link labeled "Download R x.x.x for Windows" (where x.x.x is the latest version).
4. Open the downloaded `.exe` file and follow the installation prompts, accepting the defaults.

Step 2: Install RStudio

1. Visit the RStudio download page.
2. Click the link to download the Windows installer.
3. Open the downloaded `.exe` file and complete the installation using default settings.

Step 3: Verify the installation

1. Launch RStudio from the Start menu or desktop shortcut.
2. In the Console pane, type `print("Hello, world!")` and press Enter.
3. If you see `[1] "Hello, world!"`, your installation is successful.

1.1.2 macOS

First, identify your Mac's chip. Newer Macs use Apple Silicon (M1, M2, M3, M4, etc.); older ones use Intel processors. You need to know which you have to pick the right R installer.

1. Click the Apple menu (top-left corner) → **About This Mac**.
2. Look at the **Chip** (Apple Silicon) or **Processor** (Intel) line:
    - If it says "Apple M1", "M2", "M3", "M4", or any Apple-labeled chip, you have **Apple Silicon**. You want the `arm64` installer.
    - If it says "Intel Core …", you have an **Intel Mac**. You want the `x86_64` installer.

Step 1: Install R

1. Visit the CRAN R Project website.
2. Click on "Download R for macOS".
3. Select the latest version matching your chip — either `R-x.x.x-arm64.pkg` (Apple Silicon) or `R-x.x.x-x86_64.pkg` (Intel).
4. Open the downloaded `.pkg` file and follow the installation prompts.

Step 2: Install RStudio

1. Go to the RStudio download page.
2. Click the link to download the macOS version of RStudio.
3. Open the downloaded `.dmg` file and drag the RStudio icon to your `Applications` folder.

Step 3: Verify the installation

1. Open RStudio from your `Applications` folder.
2. In the Console pane, type `print("Hello, world!")` and press Return.
3. If you see `[1] "Hello, world!"`, your installation is successful.

1.2 Install the required R packages

Now, install the seven packages required by the project.

1. Select the **Packages** tab in the bottom-right pane.
2. Click **Install** at the top of that pane.
3. In the "Install Packages" box, enter the package names separated by spaces or commas:

    ```
    here tidyverse modelsummary ggpmisc car ggrepel kableExtra
    ```

4. Make sure **Install dependencies** is checked.
5. Click **Install**.

> Tip
>
> You install packages **only once per machine**. Once installed, you "attach" them to a script (`.R`) or an R Markdown (`.Rmd`) using the `library()` function. The requisite `library()` calls are already included in the setup chunk of the `project.Rmd` template. Do **not** edit the setup

chunk.

1.3 Create the required directory or folder structure

1.3.1 Make a BUSN 5000 folder first

Before building the project directory structure, you need to set up a top-level **`BUSN 5000`** folder first. If you are new at this sort of thing, put it on your desktop. Otherwise, put it somewhere inside the `Documents` folder that makes sense to you.

Inside `BUSN 5000`, create two subfolders:

```
BUSN 5000/
  Homework/
  Project/
```

- `Homework/` is where you'll save completed homework assignments.
- `Project/` is the self-contained working directory for all project-related files.

The next step is creating the required structure inside of `Project/`. We won't have anything more to say about the `Homework/` directory in this guide.

1.3.2 Create the required subfolders inside `Project/`

The required directory structure has **two levels** – a main project folder, which you just created, and these five subfolders:

- `data/` contains input data sets
- `data_documentation/` contains the data documentation file
- `out/` contains outputs generated by the script
- `r/` contains the R script that produces outputs
- `css/` contains the CSS (cascading style sheet) file that styles the slide deck

After creating the five subfolders, your project directory should look like this:

```
BUSN 5000/
  Homework/
  Project/
      data/
      data_documentation/
      out/
      r/
      css/
```

You'll put files into these subfolders in the next step. The `.Rmd` files themselves will live in the `Project/` root, **not** in any of the subfolders.

1.4 Download the Pre-project file pack and sort its contents

Your project work starts with the Pre-project Exercise, so you'll download the Pre-project file pack from eLC first. It is a `.zip` file containing the seven files required to complete the Pre-project Exercise:

- `pre_project.Rmd` — R Markdown template for the Pre-project Exercise
- `LF.R` — R script that produces `LF.csv`

- `project_slidedeck.css` — the CSS file that styles the slide deck
- `pppub25.csv` — March 2025 CPS person-level data file
- `hhpub25.csv` — March 2025 CPS household-level data file
- `cpsmar25_documentation.pdf` — March 2025 CPS documentation file
- `check_setup_preproject.R` — R script that checks your setup for the Pre-project Exercise

Download and move the `.zip` file into your `Project/` folder and unzip it. Sort the files from the unzipped folder into the right subfolders in your `Project/` directory.

The Main Project requires a few additional files that are bundled in another file pack. We will cover those in the Main Project section of this guide.

Here is where each file goes:

File	Goes in
`pre_project.Rmd`	`Project/` (root)
`LF.R`	`Project/r/`
`project_slidedeck.css`	`Project/css/`
`pppub25.csv`	`Project/data/`
`hhpub25.csv`	`Project/data/`
`cpsmar25_documentation.pdf`	`Project/data_documentation/`
`check_setup_preproject.R`	`Project/r/`

After sorting each file into its proper location, your Project folder should look like this:

```
Project/
  pre_project.Rmd
  data/
      pppub25.csv
      hhpub25.csv
  data_documentation/
      cpsmar25_documentation.pdf
  out/
   (empty for now - LF.R will write LF.csv here)
  r/
      LF.R
      check_setup_preproject.R
  css/
       project_slidedeck.css
```

1.5 Create an R Project in the `Project` directory

Creating an R Project turns your Project/ folder into a single, self-contained workspace: every time you open it by double-clicking the `.Rproj` file, RStudio automatically makes that folder your working directory. That is what lets the `here()` paths in the scripts and the R Markdown template find your data and output files reliably – without it, RStudio has no fixed sense of where your project lives, which is the source of the cryptic "file not found" errors that come from running things out of your `Downloads` folder.

To create your R Project, follow these steps:

1. Open RStudio.
2. Go to **File → New Project**.
3. Choose **Existing Directory**.
4. Browse to your `Project/` folder and click **Create Project**.
5. RStudio re-opens in your `Project` folder loaded as the active project.

Creating the R Project writes the file `Project.Rproj` in the `Project/` root. After creating the R Project, **always, always, always open RStudio by double-clicking the `Project.Rproj` file**.

1.6 Run the setup check

After completing the above steps, run the setup-check script to confirm everything is in place. It checks for the presence of all required files and confirms that they are in the right places. If you have any setup problems, it will tell you what they are and how to fix them.

To run the setup check script:

1. Go to the **Files** tab in the bottom-right pane of RStudio.
2. Click the `r/` subfolder.
3. Click the file `check_setup_preproject.R` to open it in the editor.
4. In the editor, select all lines using Ctrl+A on Windows or Cmd+A on a Mac.
5. Click the **Run** button in the top-right of the editor pane.

The script's output appears under the Console tab in the pane below the editor, indicating whether each required file is in place — one line per file, marked either (good) or with a specific instruction telling you what to fix. Do not proceed until every line is a .

1.7 Common errors

In our experience, these are three common setup errors, each of which is easily preventable.

1. `project_slidedeck.css` in the wrong folder

   The CSS file styles your slide deck. If it's anywhere but `Project/css/project_slidedeck.css`, your knitted slides will not be format compliant and your submission will be rejected. The setup-check script will catch this, so there should be no excuse for not correcting the problem.

2. Working from your `Downloads` folder

If you launch a script or R Markdown file from your `Downloads` folder, RStudio's working directory will be `Downloads`, not your `Project/` folder and you will get cryptic "file not found" errors when the script or R Markdown tries to read files using `here()`. This occurs when you download the file pack into `Downloads` and then open the files directly from there.

3. Knitting the R Markdown before running the R script

   Knitting `pre_project.Rmd` before running `LF.R` will generate a fatal error because the Rmd reads `out/LF.csv`, which doesn't exist until you run `LF.R`. The setup-check script confirms that `LF.R` exists, but it can't confirm that it actually produced `LF.csv`. Only running the script can confirm that.

1.8 What happens next

When you are set up, continue to the Pre-project Exercise. After completing it, download and sort the Main Project file pack as described in the Main Project chapter, then run its setup check before beginning the analysis.

2Pre-project Exercise

If you have completed Setup, you are ready to start the Pre-project Exercise. If you haven't, you're not. Go back and complete it before you proceed.

The Pre-project Exercise has one real purpose and that is to get you and your computer ready for the Main Project. Successful completion entails:

- Installing the required R packages
- Setting up the required directory structure on your computer
- Learning how to "knit" an R Markdown (`.Rmd`) file to an HTML slide deck
- Understanding how (and why we need you) to save the HTML deck into the required PDF format

Do **not** treat this exercise as a mere nuisance. It counts for **5% of your Project score**.

> **!** Scoring the exercise
>
> The Pre-project Exercise is scored **all-or-nothing**; there is no partial credit. You will receive full credit **if and only if** your content is correct and the slide deck is 100% format compliant, as shown in the pre-project benchmark. Take the formatting condition seriously – any deviation will result in no credit.

2.1 The exercise deliverable

The Pre-project Exercise deliverable is a 3-slide deck, submitted as a PDF, that reads the March 2025 CPS and replicates the labor force participation rate (LFPR) and its components for that month. The deck includes:

1. A title slide
2. A slide that reads a CSV file and prints its contents
3. A slide that describes the contents of the CSV file in a paragraph using the correct values from the CSV.

That's it. Three slides.

2.2 The workflow that produces the deliverable

Follow these steps in order to produce the slide deck:

- **Step 1**: Open the Project in RStudio

  Double-click `Project.Rproj` in your `BUSN 5000/Project/` folder. RStudio opens with the project loaded. You should see "Project" in the top-right corner of the RStudio window.

- **Step 2**: Open and run the Pre-project script `LF.R`

  `LF.R` reads the raw CPS data file (`pppub25.csv`), computes an **unweighted sample LFPR**

for March 2025 and its components, and writes the results to `out/LF.csv`. The official BLS figure uses CPS survey weights and is 62.5%; the exercise intentionally uses the unweighted calculation. The script is complete; do **not** make any edits. To run the script:

1. Go to the **Files** tab in the bottom-right pane of RStudio.
2. Click the `r/` folder.
3. Click `LF.R` to open it in the editor.
4. In the editor, select all lines using Ctrl+A on Windows or Cmd+A on a Mac.
5. Click the **Run** button in the top-right of the editor pane.

If the script successfully runs, you should see the message `March 2025 LFPR = 62.6...` and the file `LF.csv` should appear in the `out/` subfolder.

- **Step 3**: Open and edit the Pre-project R Markdown template

  Go to the **Files** tab in the bottom-right pane of RStudio, move up to the main `Project/` folder and click `pre_project.Rmd` to open it in the editor.

  The **first (title) slide** is generated by the YAML (which stands for YAML Ain't a Markup Language) block, set off by the `---` markers. Slides two and three are delineated by the `##` headers, the second slide starting with "Read `LF.csv` and print its contents" and the third with "Document the contents of `LF.csv`."

  The **only** edit to make in the **YAML block** is to the `author` line. Replace `firstname lastname` with your **first and last name** like this:

  ```
  author: "Jane Doe"
  ```

  Do **not** change anything else in the YAML. If your YAML becomes corrupted, here is the definitive code block to replace your corrupted version.

  ```
  ---
  title: "BUSN 5000 :: Pre-Project Exercise"
  subtitle: Measuring Labor Force Participation
  author: "First Name Last Name"
  date: |
      | Fall 2026
      | (updated `r format(Sys.time(), '%d %b %y')`)
  output:
    ioslides_presentation:
      css: css/project_slidedeck.css
      widescreen: false
  urlcolor: "#004E60"  # Use hex code for Olympic blue from visual UGA guide
  ---
  ```

  Just below the YAML is the **setup chunk**, which is complete and requires no edits. It loads the required R packages and sets global options for the rest of the document. Do **not** touch this.

  The **second slide** of the template contains a small R code chunk marked by three backticks on the first and last lines:

```
lf <- read_csv(here("out", "LF.csv"))
print(__)
```

The first line reads the file `LF.csv` (generated by `LF.R`) and stores it in a data frame called `lf`. The second line contains a `print` command that you must complete by replacing the blank `__` with the name of the object you want to print, like this:

```
lf <- read_csv(here("out", "LF.csv"))
print(lf)
```

Run the chunk by clicking the small green play button at the top-right of the chunk. If there are no errors, the `lf` data frame contents will be printed in the console — six columns named `E`, `U`, `LF`, `NILF`, `Pop`, and `LFPR`, with one row of numbers.

When the chunk runs successfully, change the chunk option `eval = FALSE` to `eval = TRUE` at the top of the chunk. This tells R Markdown to run the chunk when you knit. We set it to `FALSE` by default so the template knits before you complete it.

The **third slide** is a descriptive paragraph with six blanks serving as placeholders for the LFPR and its components as defined in `LF.R`. Replace each blank with the appropriate value from the second slide's output.

- **Step 4**: Knit `pre_project.Rmd` to an HTML slide deck and save it as a PDF

  Click the **Knit** button at the top of the editor pane. If there are no errors, `pre_project.html` will appear in your `Project/` folder and RStudio will open it automatically in its viewer or your browser. The Submission chapter shows exactly what the pre-project slide deck should look like when rendered correctly. Compare yours with the benchmark before saving and submitting.

  Unfortunately, Gradescope does not accept HTML files, so you must save the HTML as a PDF. Do **not** knit directly to PDF; that will not produce the correct format. Instead, use your browser's Save As / Print to PDF function to save the HTML output as a PDF. See Submission for the exact procedure and browser-specific tips.

2.3 Avoid these mistakes

The most common errors we see in attempts of the Pre-project Exercise are:

1. **Leaving `eval = FALSE` after completing the code chunk.** If `eval = FALSE`, the chunk doesn't run when you knit, so the output doesn't appear on the slide. Set it to `TRUE` once your code is correct.
2. **Values in the third slide paragraph don't match those in `LF.csv`.** This usually means a student made a transcription error or used a source different from the output of `LF.R`.
3. **YAML edits beyond the author line.** Changing anything in the YAML besides the author name will break the knit format. Don't touch the `title`, `subtitle`, `date`, `output`, `css`, or any other field.
4. **CSS file not correctly placed.** If `project_slidedeck.css` is anywhere other than `Project/css/`, the knit produces an unstyled output that won't match the template.

When in doubt, see Common Errors, which provides a comprehensive listing of errors we've seen students make.

2.4 What happens next

After successfully completing the Pre-project Exercise, you are ready to begin work on the Main Project. The Main Project template will be downloaded to `Project/` folder and you will follow the same workflow outlined here.

3Main Project

If you have successfully completed the Pre-project Exercise, you're ready to start the Main Project.

The deliverable is a 35-slide deck examining pay differences between women and men using the March 2025 CPS. We **strongly recommend** adopting a cadence for your project activity that is in sync with our treatment of related course topics. Here is the mapping:

Course module and focus	Project point	Project section or milestone
Data Fundamentals - R scripts, loading data, and R Markdown basics	Slides 1-7	Introduction and Data (Part 1)
Beginning to Learn - Data-generating process and learning from data	Slides 8-12	Data (Part 2) and Baseline Earnings
Models for Exploration - Variance, covariance, and correlation	Slides 13-20	The Career Gender Gap
Making Inferences - Statistical uncertainty and significance	Slides 21-26	Explaining the Gap (Part 1)
Sources of Bias - Measurement error, selection, and confounding	Through Slide 20	Optional check: verify tables and figures
Regression Fundamentals - Regression for description, causation, and prediction	Slides 27-35	Explaining the Gap (Part 2) and Conclusion

One cohesive story: each module's learning enables the next piece of the project.

> **Note**
>
> If you fall behind on the course content, the project gets harder fast. Stay on schedule. We won't have any sympathy for those who start late and come up short in the end.
> To help you stay on track, we offer an optional **Project Progress Check**, which covers the first 20 slides. As a nudge to take us up on the option, we will award up to 5 bonus points depending on the correctness of your submission. Even if you whiff on the progress check, you will be in a much better position to succeed on the final project than if you skip the check entirely.

3.1 First, download the Main Project file pack and sort its contents

You should already have downloaded and sorted the contents of the Pre-project Exercise file pack. Now do the same with the Main Project file pack, which contains

- `project.Rmd` — R Markdown template you will complete and knit into a slide deck
- `cpsmar_e.R` — R script that creates the CPS data extract for your analysis from the raw CPS files
- `check_setup_project.R` — R script that checks your setup for the Main Project

Here's where each file goes:

File	Goes in
`project.Rmd`	`Project/` (root)
`cpsmar_e.R`	`Project/r/`
`check_setup_project.R`	`Project/r/`

Run `check_setup_project.R` to confirm everything's in the right place.

3.2 General instructions

Before you jump into your project work, take note of the following general instructions that apply to the entire project. Overlooking any of them will prove costly.

- **Slide format**

  An acceptable submission has **35 slides on 35 pages of PDF output**, rendered in landscape mode, precisely matching the format of the Final Project reference deck. If your deck doesn't comply exactly with the reference deck, it will receive a score of 0.

- **Write-up line limits**

  Each slide requiring text content has a specific line limit. **We will not read beyond the line limit.**

  > **!** What counts as a "line"
  >
  > "Lines of text" means **rendered, countable lines on your final PDF** — not sentences, and not lines as they appear in RStudio's editor or output preview. A long sentence that wraps into three visual lines on the rendered slide counts as three lines. Submissions that exceed the line limit are **heavily penalized**. Always check your knitted PDF and count the actual visible lines on the slide before you submit.

- **HTML code for table formatting**

  Code chunks associated with table construction are wrapped in the `<div class="table-...">` and `</div>` HTML tags, which govern table formatting. Do **not** edit or remove them.

- **The YAML block**

  As we explained in the Pre-project Exercise, the YAML block at the top of the template contains important formatting information. You must edit the `author:` field to insert your name, but **do not change anything else** in the YAML.

  If your YAML becomes corrupted, here is the definitive code block to replace your corrupted version:

```yaml
---
title: "BUSN 5000 Project"
subtitle: "Exploring Pay Differences between Women and Men"
author: "First Name Last Name"
date: |
    | Fall 2026
    | (updated `r format(Sys.time(), '%d %b %y')`)
output:
  ioslides_presentation:
    css: css/project_slidedeck.css
    widescreen: true
---
```

- **The setup chunk**

  Like with the Pre-project Exercise template, below the YAML you will find the **setup chunk** and it is complete as is. Do **not** touch it. As you should know from the Pre-project Exercise, the setup chunk loads the required R packages and sets global options for the rest of the document.

- **Echo settings**

  The setup chunk sets `echo = TRUE`, which will display code chunks by default. Note that we have overridden this global setting on certain slides by setting `echo = FALSE` in those chunks, where displaying the code would be redundant or otherwise not valuable. Do **not** change these individual `echo` settings.

- **Get the units right**

  In your write-ups, make sure you use **percent** and **percentage point** correctly. They are not interchangeable. For example, a change in the gender wage gap from 25% to 20% is a **5 percentage point** decrease — not a 5% decrease. A 5% decrease in the gender wage gap from a 25% baseline would amount to a change of 1.25 percentage points to 23.75%. We will definitely penalize you for this sort of error, so don't make it.

> Workflow tip
>
> You can run individual chunks without knitting the entire document, so work the project code chunks one at a time. After you complete a chunk, confirm it runs cleanly by clicking the green play button at the top-right of the chunk. If it does, set `eval = TRUE` on that chunk so its output appears when you knit the whole document.

3.3 Slide-by-slide instructions

There are 35 slides in a successfully rendered deck. Some are just section dividers, with no point values and requiring no contribution from you. For those that require your input – either code completions or text responses – we provide specific instructions for how to enter it correctly, along with the corresponding point values in parentheses. So, here it goes, slides 1-35:

1. **BUSN 5000 Project**.

Title slide. It's auto-generated by the YAML with your name as entered on the `author:` line.

2. **Academic Honesty Statement** *(required)*

   Type your first and last name on the "Signature:" line. You may consult Terry Analytics Lab staff, the TA, or the instructor and may use AI tools as assistants. You must direct, review, understand, and verify any assistance you receive; the deliverable must represent your work and be completed and submitted by you.

3. **Introduction**

   Section divider.

4. **Overview** *(2 points)*

   Provide a brief overview of the project. Include:

   - What you are trying to learn about
   - The data you are using to learn
   - A brief summary of your findings

   Limit your overview to **6 lines of text**.

   > Pro tip
   >
   > To quote Yoda: "Do or do not. There is no try." When you explain what your project is about or summarize what you learned, don't say "I try / seek / look / aim / attempt (or am trying / seeking / …)". Instead, say "I show / document / demonstrate / report / find …". Never say "I hope…"
   > You might write this slide *last*. It's easier to summarize your findings after you've completed the analysis.

5. **Data**

   Section divider.

6. **March 2025 CPS** *(4 points)*

   Using the ASEC documentation (`cpsmar25_documentation.pdf`), provide a brief overview of the March 2025 CPS. Include:

   - A description of the standard monthly CPS
   - The additional information collected in the ASEC
   - Approximately the number of households surveyed in the March 2025 ASEC

   Limit your summary to **4 lines of text**.

   > Pro tip
   >
   > You can ask AI, but you should verify the response by consulting the CPS documentation and formulate the overview in your own words.

7. **March 2025 CPS Extract** *(4 points)*

   Complete the `read_data` code chunk to read the extract you created with `cpsmar_e.R`.

Then, in the write-up section, explain the actions you applied to the March 2025 survey in `cpsmar_e.R` to create the data extract `cpsmar_e.csv`. Include:

- The variables you selected from the person file
- The variables you selected from the household file
- The restriction(s) you applied to the data extract
- The number of observations and variables in the data extract

Limit your explanation to **6 lines of text**.

> Pro tips
>
> Refer to each variable by its plain English meaning, not the name used in the script. Refer to the key tidyverse "verbs" in the script, like `mutate` and `rename`.

8. **Analysis sample** *(2 points)*

   Complete the `btl1` code chunk to create your analysis sample (`cpsmar_a`) with the following restrictions:

   - Restrict to individuals who are 23 to 62 years old (inclusive)
   - Restrict to individuals who have positive earnings

   Underneath the code chunk, document the number of observations in your analysis sample.

   Limit your documentation to **2 lines**.

   > Pro tips
   >
   > Refer to your notes or the slides for examples of `filter` and `mutate`. Consult the Environment tab in the Northeast pane of RStudio for information about the `cpsmar_a` data frame.

9. **Baseline earnings distributions**

   Section divider.

10. **Plotting earnings distributions**

    Complete the `btl2` code chunk to:

    - Create the `figure1` ggplot object (the earnings distribution by gender plot)
    - Calculate average earnings for each gender using the `earnings_fvm` object
    - Pull the average earnings for women and men into single values (`avg_earnings_f`, `avg_earnings_m`) using `filter` and `pull`

11. **Distribution of earnings by gender** *(8 points)*

    Complete the `btl2.5` code chunk to display Figure 1 by writing the name of the Figure 1 object.

12. **Baseline comparisons** *(4 points)*

    Summarize the main empirical facts associated with Figure 1. Include:

    - A description of the most important fact communicated by the figure

- The average earnings of men and women
- The dollar difference and percentage difference in average earnings between men and women in the sample

Limit your summary to **5 lines of text** on the slide.

> Pro tips
>
> Use inline R syntax with the `avg_earnings_f` and `avg_earnings_m` objects to insert the respective averages in your write-up rather than typing the numbers manually.

13. **The career gender gap**

    Section divider.

14. **Wages and hours differences** *(8 points)*

    Complete the `mi1` code chunk to create and display Table 1 (wages and hours by gender).

15. **Documenting the differences** *(2 points)*

    Summarize the wage and hours differences presented in Table 1.

    Limit your documentation to **3 lines of text** on the slide.

16. **Plotting career log wage profiles**

    Complete the `mi2` code chunk to estimate log wage profiles for women and men and create Figure 2.

17. **Career log wage profiles** *(8 points)*

    Complete the `mi2.5` chunk to display Figure 2.

18. **Estimating wage differences over a career**

    Complete the `mi3` code chunk to create Table 2. This involves:

    - Creating `males` and `females` objects using `filter` and `rename`
    - Merging the two into the `diff_fvm` object using `inner_join`
    - Calculating the difference between average log wages using `mutate`
    - Grouping by `age_group`
    - Using `kable()` to organize the results into the `table2` object

    Table 2 covers career years 0 through 30. Its first group, labeled 0-10, includes the starting year (year 0) through year 10.

19. **Evolution of the gender wage gap** *(8 points)*

    Complete the `mi3.5` code chunk to display Table 2.

20. **Discussing the gender wage gap evolution** *(2 points)*

    Summarize the results presented in Figure 2 and Table 2.

    Limit your summary to **3 lines of text** on the slide.

21. **Explaining the gender wage gap**

Section divider.

22. **Fitting the log wage profiles**

    Complete the `reg1` code chunk to create Figure 3 by fitting the career profiles with a quadratic in age.

23. **Log wage profiles with quadratic fits** *(8 points)*

    Complete the `reg1.5` chunk to display Figure 3.

24. **Gender differences in education** *(8 points)*

    Complete the `ed_vars` code chunk to create and display Table 3 (educational attainment by gender).

25. **Gender differences in demographics** *(8 points)*

    Complete the `demo_vars` code chunk to create and display Table 4 (demographic characteristics by gender).

26. **Documenting differences in characteristics** *(4 points)*

    - Summarize the educational attainment differences presented in Table 3
    - Summarize the differences in demographic characteristics presented in Table 4

    Limit your documentation to **6 lines of text** on the slide.

27. **Controlling for education and demographic characteristics**

    Complete the `reg2a` code chunk to create:

    - The `singles` subset of the analysis data
    - The `models` object containing five regression models that incrementally add controls for education and demographic characteristics, with the fifth restricting to unmarried workers without children under 6 ("Only Singles")

    In the regression analysis, you'll distinguish between personal and household characteristics among the demographic variables.

    > Pro tips
    >
    > - Moving from top to bottom in the `models` object, the list of controls grows as indicated by the name of the model — with the exception of the final model "Only Singles", which estimates the same relationship as the "Add Person" model but on the new subset.
    > - To decide whether a variable belongs in the "Add Person" or "Add Household" model, ask yourself: *"Can I define this variable for a particular individual irrespective of other individuals who may or may not exist in their life?"* If yes → Person. If no (it depends on someone else, like a spouse or a child) → Household.

28. **Reporting the results**

    Complete the `reg2b` code chunk to create Table 5. This involves:

    - Constructing the coefficient map object to display only the intercept, gender, and age coefficient estimates

- Constructing the goodness-of-fit object to display sample size and R^2 values
- Constructing a `rows` object to distinguish the regression specification associated with each column

> Pro tip
>
> Make sure you report robust standard errors and indicate that you do in a table note.

29. **Explaining the gender wage gap** *(8 points)*

    Complete the `reg2.5` chunk to display Table 5.

30. **Documenting the findings** *(4 points)*

    Summarize how the estimated average gender wage gap changes as you add education, personal, and household characteristics to the regression — and then how it changes when the sample is restricted to singles.

    Limit your documentation to **6 lines of text** on the slide.

> Pro tips
>
> Start your write-up with the baseline model and describe subsequent results relative to the baseline. Focus on the Female coefficient estimate and its standard error. Remember how the coefficient estimate is correctly interpreted.

31. **Conclusion**

    Section divider.

32. **Summary** *(4 points)*

    Briefly summarize the objective of the project and its main findings. Note:

    - The sample on which your analysis is based
    - The overall gender wage gap
    - How it evolves over a career
    - How it varies when controlling for education and demographic characteristics

    Limit your documentation to **7 lines of text** on the slide.

33. **Appendix**

    Section divider.

34. **Data documentation**

    Complete the `var_doc` code chunk to create a table of the main variables used in this project with their definitions.

35. **List of main variables with definitions** *(4 points)*

    Once you complete the `var_doc` chunk on the previous slide, set `eval = TRUE` on this slide to render the table of main variables with their definitions.

3.4 Common pitfalls

If you completed the Pre-project Exercise successfully, you have already bypassed the most common mistakes. If you nevertheless are having trouble completing the Main Project, Common Errors provides a comprehensive listing of problems we have seen students encounter, along with explanations and solutions. Some typical issues with the Main Project are:

- **Variable name confusion** — defining `cpsmar_a` but referencing `cpsmar_e` (or vice versa) in a later chunk. See Common Errors.
- **Forgetting to run `cpsmar_e.R` before knitting.** See Common Errors.
- **Touching the setup chunk's `include = FALSE`** — same scaffolding error as the pre-project. See Common Errors.
- **Editing the YAML beyond the author line.** See Common Errors.

3.5 What happens next

Whether you are at the Progress Check stage or at the end, the next step is to prepare to submit your work to Gradescope. Submission walks you through the entire process.

4Progress Check

As we said in the Main Project section, the Project Progress Check is an opportunity to get feedback on your project work well before the Main Project submission deadline. It covers the first 20 slides and is entirely **optional**.

While optional, we think it is important enough to award up to 5 bonus points. If you failed on the Pre-project check, think of these bonus points as a way to get those 5 points back. The bonus points are just a nudge – the primary value is in the feedback you will get to ensure the successful completion of the entire project. Suffice it to say, students who participate in the progress check do way better on the final project than those who don't.

4.1 What we check

The progress check covers the first 20 slides, reviewing your overall deck format and this specific content:

Slide	Title	What we check
1	Title slide	Your name filled in (and overall formatting is correct)
2	Academic Honesty Statement	Your name on the Signature line
11	Distribution of earnings by gender	Figure 1 is visible, completed, and correct
14	Wages and hours differences	Table 1 is visible, completed, and correct
17	Career log wage profiles	Figure 2 is visible, completed, and correct
19	Evolution of the gender wage gap	Table 2 is visible, completed, and correct

> **Note**
>
> The slide numbers above match the canonical 35-slide layout documented in the Final Project reference in the Submission chapter. Your deck must be 35 slides total even if many of the content slides are still placeholders.

We do **not** check your text responses or any content beyond slide 20.

4.2 How we score it

There are three possible outcomes for the progress check:

Result	Score	Criteria
Full credit	5 bonus points	Fully format compliant and zero errors in the content check.
Partial credit	3 bonus points	Fully format compliant but non-zero errors in the content check.
No credit	0 bonus points	Not fully format compliant.

> **!** Important
>
> A **necessary** condition for your progress check submission to earn any credit is to be fully format compliant. Any formatting error will result in a score of 0.

4.3 What happens next

When you are ready, submit your progress check deliverable to Gradescope following the same save-as-PDF workflow as spelled out in Submission for the Pre-project Exercise and Main Project.

Then get back to work on the Main Project.

5Submission

If you have successfully completed the Pre-project Exercise or Project Progress Check you've presumably mastered this process. If you have not, pay close attention because failure to follow the submission instructions to a "T" could land you in the **no credit** or **zero score** category.

First, this important reminder…

> **!** Late submission policy
>
> Per the syllabus: **late submissions will not be accepted** for the Pre-project, the Project Progress Check, or the Final Project. The relevant deadlines are on the course schedule. There are no extensions and no late-credit options.

5.1 Filename conventions

Each of the three project artifacts has a required filename. Use **lowercase** for first and last name, **underscores** between words, and the `.pdf` extension.

Artifact	Filename pattern	Example
Pre-project	`firstname_lastname_pre.pdf`	`jane_doe_pre.pdf`
Project Progress Check	`firstname_lastname_progress.pdf`	`jane_doe_progress.pdf`
Final Project	`firstname_lastname_project.pdf`	`jane_doe_project.pdf`

5.2 The submission workflow at a glance

Each submission goes through the same three steps:

1. **Knit** your `.Rmd` to an HTML document by clicking the **Knit** button in the editor pane. "Knit to HTML" is the default output format, so you don't need to select it from the drop-down arrow — just click the button itself.
2. **Save** the HTML document as a PDF using your browser's print-to-PDF function following the instructions below.
3. **Upload** the PDF to the corresponding Gradescope assignment under the correct filename.

5.3 Saving the HTML document as a PDF

After knitting, the HTML output opens in your browser or in RStudio's Viewer. If it opens in the Viewer, click the small icon at the top of the Viewer to open it in your browser.

Using your browser's print dialog you can save as PDF following these general steps:

- **Destination:** "Save as PDF"

- **Layout / Orientation:** Landscape
- **Color:** Color (not Black and White)
- **Margins:** Default or None
- **Pages per sheet:** 1
- **Background graphics:** Enabled (this is the easy-to-miss one — without it, the colored UGA styling disappears)

Browser print dialogs change over time, so use the settings checklist above rather than trying to match an old screenshot. Before saving, confirm that the preview shows exactly **3 pages** for the Pre-project or **35 pages** for the Progress Check and Main Project.

5.3.1 Google Chrome (and Microsoft Edge)

Edge uses the same print dialog as Chrome — these steps work for both.

1. Click the three vertical dots in the top-right of the browser.
2. Click **Print**.
3. Match the settings above — most importantly, **Destination = Save as PDF**, **Layout = Landscape**, and **More settings → Background graphics = checked**.
4. Click **Save**. Use the correct filename when prompted (see Filename conventions).

5.3.2 Firefox

1. Click the three horizontal lines (the "hamburger" menu) in the top-right of the browser.
2. Click **Print**.
3. Match the settings above. Key settings: **Destination = Save to PDF**, **Orientation = Landscape**, and **Options → Print backgrounds = checked**.
4. Click **Save** and use the correct filename.

5.3.3 Safari

1. Click **File** in the top-left menu bar.
2. Click **Print**.
3. Match the settings above. Key settings: **Orientation = Landscape**, **Print backgrounds = checked**, and the **PDF dropdown in the bottom-left** must be set to **Save as PDF** (not Print).
4. Click **Save** and use the correct filename.

5.4 Links to benchmark reference decks

Before you submit anything to Gradescope, check your deck against the relevant benchmarks for format compliance, and in the case of the progress check, content compliance. Here are the links:

- Pre-project benchmark: what your three slides should look like
- Main Project knitted template: what your 35 slides should look like before artifacts are produced and your explanations added
  - See Progress Check for artifacts covered in the Progress Check
  - See Main Project for slide-by-slide instructions for the entire project

6Common Errors

Here we catalog the most common errors we've seen students make. If you are having problems, you will likely find it documented here, along with the cause and fix. If your problem persists, reach out to the TAL or a member of the teaching team.

6.1 Getting help

Start with the setup-check script and the relevant section of this chapter. If the problem persists, bring the exact error message and your project files to the Terry Analytics Lab, your TA, or the instructor through the course's usual contact channel.

6.2 Setup errors

6.2.1 CSS not loading

Symptom: Your knitted slides have no UGA styling — plain text, no colors, default fonts.

Cause: Either (a) `project_slidedeck.css` is anywhere other than `Project/css/`, or (b) the `.Rmd` file you're knitting isn't in your `Project/` root (e.g., you left it in your Downloads folder), so the relative path `css/project_slidedeck.css` in the Rmd's YAML can't find the CSS.

Fix: Confirm both files are in the right places: the CSS in `Project/css/` AND the `.Rmd` in your `Project/` root. Open RStudio by double-clicking `Project.Rproj`, then re-knit. Visual example of what the bad output looks like: Wrong CSS.

6.2.2 Working from your Downloads folder

Symptom: "File not found" errors when running scripts or knitting. R can't see your data files even though they exist.

Cause: RStudio's working directory is your Downloads folder, not your `Project/` folder. The `here()` paths in scripts and the Rmd resolve to the wrong place.

Fix: Move all files into `Project/` and always open RStudio by double-clicking `Project.Rproj`. Never work from Downloads.

6.2.3 Required files missing or in the wrong subfolder

Symptom: "Cannot open the connection" or "object not found" errors during a script run or knit.

Cause: A required file isn't where the script or Rmd expects it (e.g., `LF.R` is in your project root instead of `r/`, or `pppub25.csv` is missing from `data/`).

Fix: Run `check_setup_preproject.R` (pre-project) or `check_setup_project.R` (main project). The script tells you exactly which file is misplaced and where it should go.

6.3 Workflow errors

6.3.1 Knitting before running the R script

Symptom: Knit errors out with a "file not found" message pointing at `out/LF.csv` or `out/cpsmar_e.csv`.

Cause: You haven't run `LF.R` (pre-project) or `cpsmar_e.R` (main project) yet, so the file the Rmd is trying to read doesn't exist.

Fix: Run the relevant R script first (open it in the editor, Select All, click Run), then knit the Rmd.

6.3.2 Variable name confusion

Symptom: Knit errors out mid-document with "object not found" — a chunk references `cpsmar_a` but you defined `cpsmar_e` (or similar).

Cause: Typo or misalignment between chunks. You defined the analysis sample with one name, then referenced a different name in a downstream chunk.

Fix: Read each chunk carefully. Match data-frame names exactly across chunks. The variable structure in the template is: `cpsmar_e` (the extract) → `cpsmar_a` (the analysis sample) → `singles` (the singles subset).

6.3.3 Touching the setup chunk

Symptom: Output looks broken or knit fails in unexpected ways (missing libraries, scientific notation issues, formatting wrong).

Cause: You changed `include = FALSE` to `include = TRUE` on the setup chunk, or otherwise modified the setup chunk's contents.

Fix: Restore the setup chunk to its original state. The `eval = FALSE → TRUE` toggle only applies to content code chunks (`btl1`, `btl2`, `mi1`, etc.), never the setup chunk.

6.4 Rendering errors (the knit step)

6.4.1 Leaving `eval = FALSE` on a completed chunk

Symptom: Your code is correct but the output doesn't appear in the knitted slide. The chunk shows the code but no result below it.

Cause: The chunk option `eval = FALSE` is still set. R Markdown displays the code but skips running it during knit.

Fix: Change `eval = FALSE` to `eval = TRUE` on every content chunk you've completed. Re-knit.

6.4.2 Editing the YAML beyond the author line

Symptom: The knit produces unexpected output — different layout, missing CSS, wrong document class, no slide breaks.

Cause: You changed something in the YAML besides the `author:` line (e.g., the `output:` value, the `css:` reference, the `widescreen:` setting).

Fix:

1. Identify which Rmd is broken: `pre_project.Rmd` or `project.Rmd`.
2. Copy the canonical YAML block from the relevant chapter:
    - **Pre-project:** the YAML block is in Pre-project Step 3.
    - **Main project:** the YAML block is in Main Project — The YAML block.
3. Open your working Rmd. Replace your current YAML with the YAML you just copied.
4. Re-add your name on the `author:` line.
5. Re-knit.

6.4.3 Using the Knit dropdown instead of the Knit button

Symptom: Output is a Word document, a LaTeX-styled PDF, or some format that doesn't match the slide deck reference.

Cause: You clicked the small dropdown arrow next to the Knit button and selected "Knit to PDF / Word / etc." That overrides the output format specified in the Rmd's YAML.

Fix: Click the **Knit** button itself, not the dropdown arrow. The YAML already tells R Markdown to knit to HTML; just trust the button.

6.5 Output format errors

These are the canonical "submission rejected" failure modes, each shown with a real example from a prior submission. If your output matches any of these images, your submission will receive a 0.

6.5.1 Wrong CSS

Symptom: Output is in landscape but has no UGA styling — plain text, gray gradients, default browser fonts.

Cause: `project_slidedeck.css` was not found at knit time (usually because it's not in `Project/css/`).

Fix: Verify the CSS file is in `Project/css/`. Verify the Rmd you are working on is in `Project/`, not somewhere else (like `Downloads/`). Re-knit.

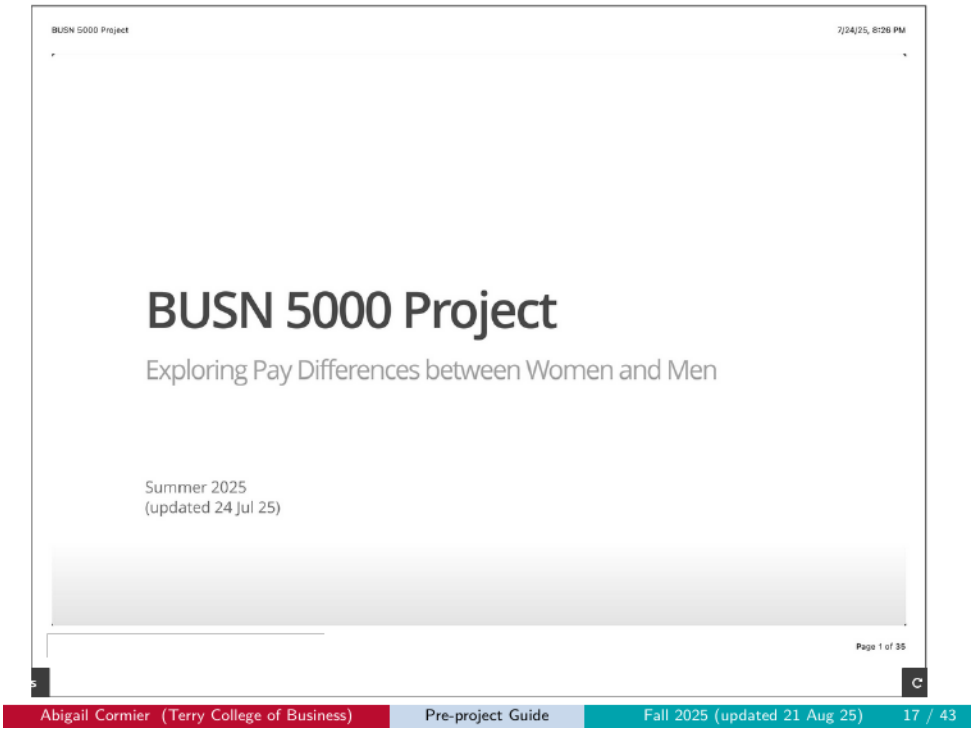

Figure 6.1: Wrong CSS example

6.5.2 Wrong CSS + wrong orientation

Symptom: Output is portrait AND has no UGA styling.

Cause: Both CSS is missing and the print dialog saved as portrait.

Fix: Move CSS to `Project/css/`. Verify the Rmd you are working on is in `Project/`, not somewhere else (like `Downloads/`). Re-knit, then save with **Landscape** orientation.

Figure 6.2: Wrong CSS, wrong orientation example

6.5.3 Wrong orientation

Symptom: Correct UGA styling, but the PDF is portrait.

Cause: Print dialog defaulted to portrait and you didn't change it.

Fix: In the browser's print dialog, set **Layout / Orientation to Landscape** before saving.

Figure 6.3: Wrong orientation example

6.5.4 Long landscape

Symptom: Landscape PDF, but the slide content is stretched horizontally across a single very wide page instead of broken into separate slide pages.

Cause: A print dialog scaling setting fit multiple slides onto one wide page.

Fix: Reset print dialog scaling to default. Make sure **Pages per sheet = 1**.

Figure 6.4: Long landscape example

6.5.5 Portrait

Symptom: Output is portrait. (Same as "Wrong orientation" above, isolated here as an example.)

Cause: Print dialog defaulted to portrait.

Fix: Set Layout to **Landscape**.

Figure 6.5: Portrait example

6.5.6 Portrait + wrong CSS

Symptom: Output is portrait AND has no UGA styling. Same underlying causes as the "Wrong CSS + wrong orientation" case, isolated here as a distinct example pattern.

Fix: Move CSS to `Project/css/`. Verify the Rmd you are working on is in `Project/`, not somewhere else (like `Downloads/`). Re-knit, then save with **Landscape** orientation.

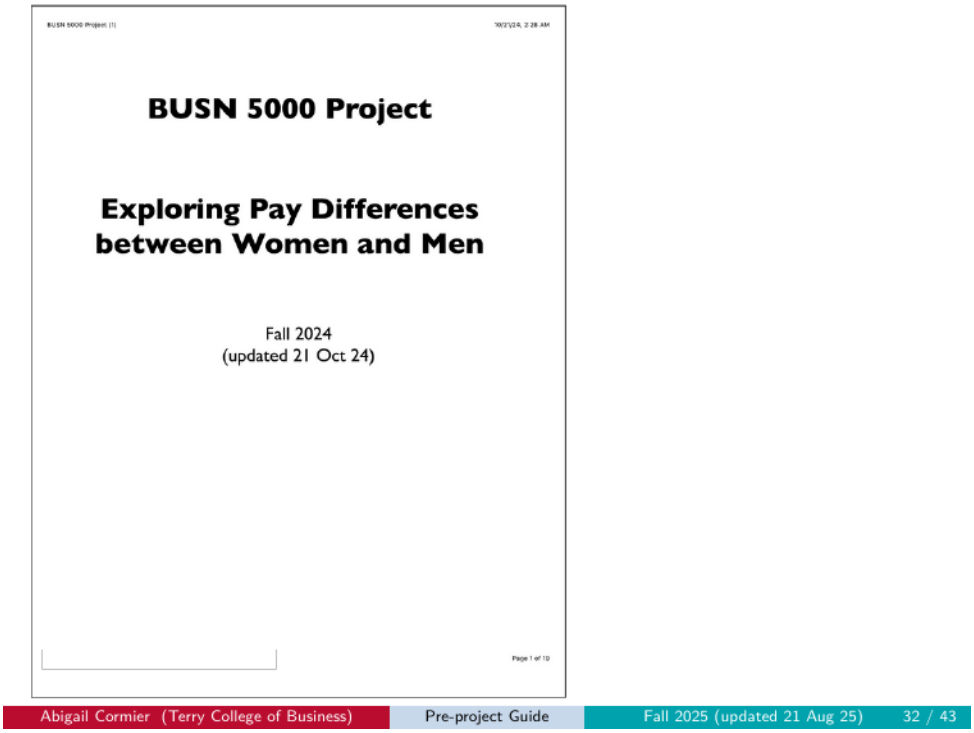

Figure 6.6: Portrait + wrong CSS example

6.5.7 LaTeX slides (knit-to-PDF, slide format)

Symptom: Output is a slide deck rendered with LaTeX/Beamer styling — different fonts, different layout, no UGA brand colors.

Cause: You used the Knit dropdown to select "Knit to PDF" with slide output, overriding the Rmd's HTML-based ioslides format.

Fix: Use the Knit button (not the dropdown). The Rmd's YAML specifies `ioslides_presentation` for a reason. If your YAML has also been modified (which often goes hand-in-hand with this error), restore it by following the 5-step process under Editing the YAML beyond the author line. Then re-knit.

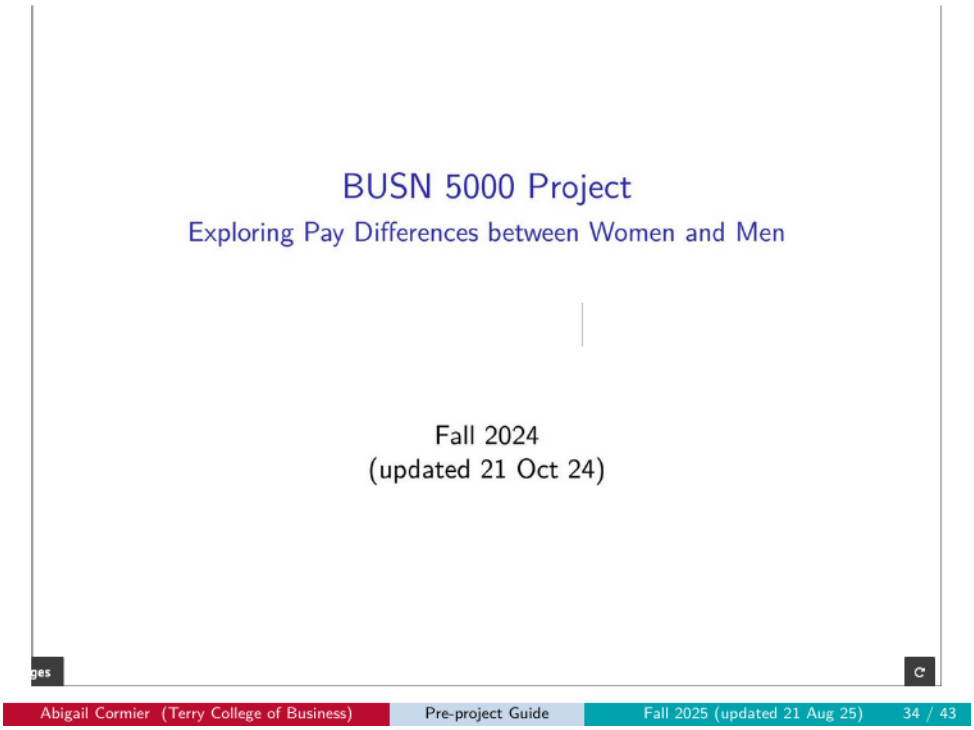

Figure 6.7: LaTeX slides example

6.5.8 LaTeX document (knit-to-PDF, document format)

Symptom: Output is a regular PDF document, not a slide deck at all. Looks like a paper or article.

Cause: You used the Knit dropdown to select "Knit to PDF" with document output, AND likely modified the YAML to change the output format.

Fix: Restore the YAML. Knit with the button. Save as PDF from the HTML output.

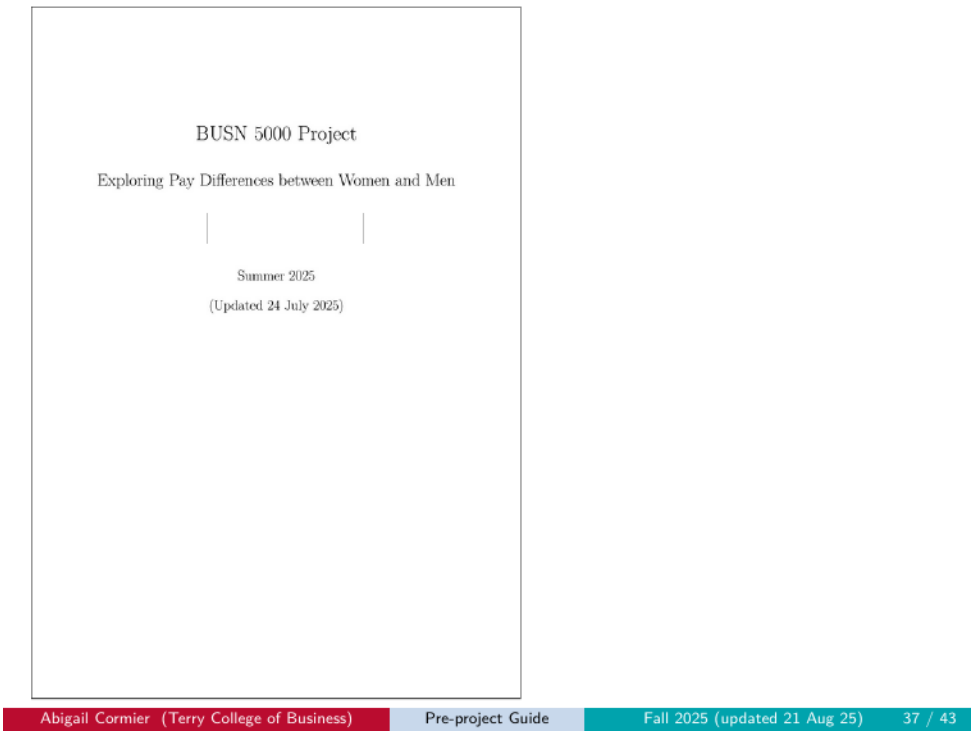

Figure 6.8: LaTeX document example

6.5.9 Not knit

Symptom: Submission is the raw `.Rmd` file, or a file that was never actually knit — no rendering at all.

Cause: You uploaded the wrong file (the `.Rmd` source instead of the saved-as-PDF output).

Fix: Knit the Rmd to HTML, save the HTML as PDF using your browser's print dialog, then upload that PDF.

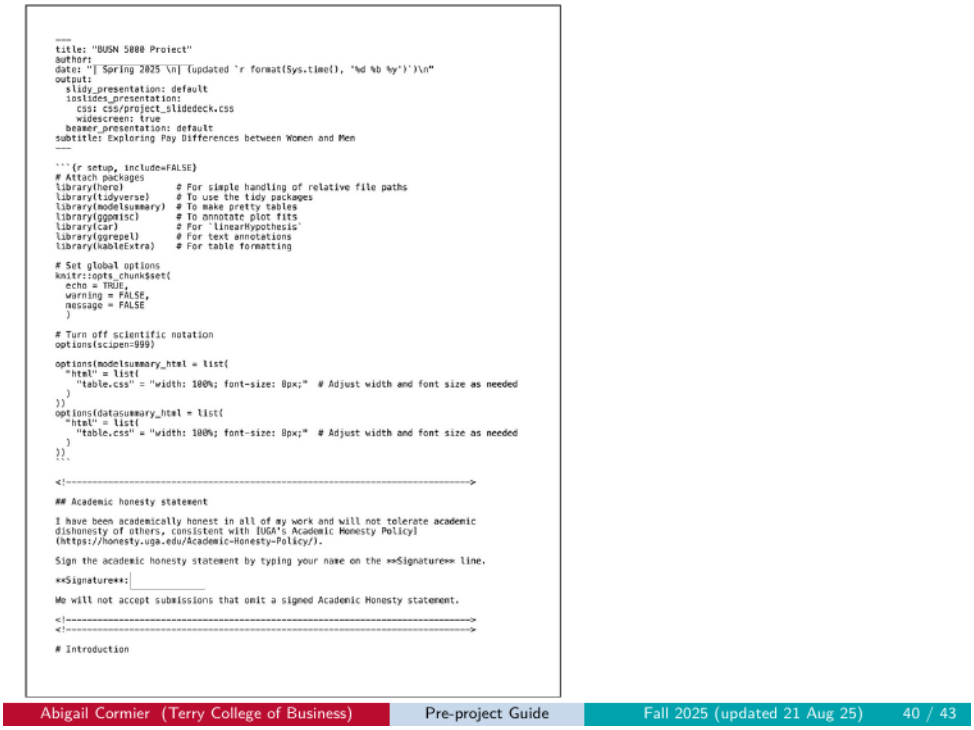

Figure 6.9: Not knit example

6.5.10 Wrong file

Symptom: Submission is something other than the intended project artifact — an old version, a template, an unrelated document.

Cause: You uploaded the wrong file.

Fix: Locate the correct PDF in your `Project/` folder and re-upload. Double-check the filename matches the convention.

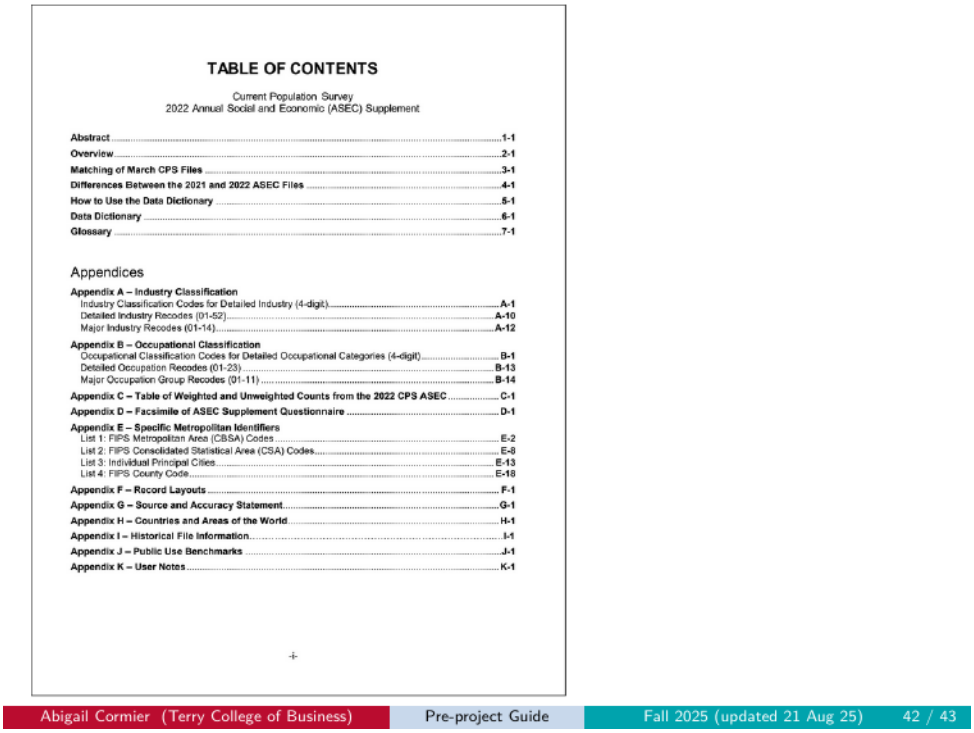
TABLE OF CONTENTS

Current Population Survey
2022 Annual Social and Economic (ASEC) Supplement

Figure 6.10: Wrong file example

6.6 Content errors

6.6.1 Numbers in your paragraph don't match `LF.csv` (pre-project)

Symptom: Your slide 3 paragraph cites numbers that don't reflect your own data — they came from a classmate or from a previous semester.

Cause: You copied numbers from a peer instead of running `LF.R` yourself, or `LF.R` didn't actually run and the values you wrote down were stale.

Fix: Run `LF.R` yourself. Open `out/LF.csv` (or print `lf` in the console). Fill in your slide 3 paragraph with those values, rounded to one decimal place.

6.6.2 Wrong filename

Symptom: Your file was uploaded but appears under an unexpected name in Gradescope, or you can't tell which submission is which in your `Project/` folder later.

Cause: The filename didn't match the required pattern.

Fix: Rename your file to the correct convention (`firstname_lastname_pre.pdf` / `firstname_lastname_progress` / `firstname_lastname_project.pdf`) and re-upload.

7R and AI Resources

A curated list of resources for learning and using R, and for working with AI as part of your data-science practice. Nothing here is required reading — but each item below has earned its place. Skim it now, bookmark it, and return as you go.

7.1 Resources for learning and using R

7.1.1 IDEs for R

RStudio remains the most widely used IDE for R, and, like R, it is open-source. Its "Help" tab is a gateway to a wide range of R and RStudio resources. However, Posit (the company behind RStudio) has released Positron, a next-generation IDE built for data science with both R and Python. Positron combines the familiar feel of RStudio with the extensibility of VS Code, and includes built-in AI assistance via Posit Assistant. If you're comfortable with RStudio, stick with it; if you're curious about a more modern interface or plan to work across R and Python, Positron is worth exploring.

7.1.2 Learning R

The place to start is the widely used R for Data Science by Wickham and Grolemund. They will guide you through the Tidyverse. Tidy Data Tutor helps you visualize your data analysis pipelines. James Scott has a great introduction to R that maps onto BUSN 5000.

Kieran Healy's Data Visualization book (also linked on the course schedule) is a great introduction to the wonders of ggplot, which is not R, but indispensable to analysis done in R.

Learning R Markdown or Quarto should go hand-in-hand with learning R. For R Markdown, get started here and keep a cheatsheet handy. You can get started with Quarto here.

7.1.3 Workflow

At least as important as learning R is understanding basic workflow principles. R-bloggers has a beginner's guide using RStudio Projects. You can find some of the same principles described in chapter 2 of Healy's book. Matt Gentzkow and Jesse Shapiro provide a fuller take on workflow principles from the perspective of academic social scientists (also posted on eLC). They even provide a manual for doing as they do.

For a perspective specifically on coding practices in data science education, Pruim, Gîrjău, and Horton (2023) make the case in "Fostering Better Coding Practices for Data Scientists" (*Harvard Data Science Review*) that good coding habits matter even for beginners. Their argument: it's far easier to learn good practices alongside R than to unlearn bad habits later. They organize their guidance into a practical top-10 list. Worth reading early in your data-science career.

7.2 AI in August 2026

7.2.1 The main AI systems and keeping up with them

In August 2026, widely used general-purpose AI products include ChatGPT from OpenAI, Claude from Anthropic, and Gemini from Google. These products are powered by underlying models, and both the products and models change frequently.

If your experience has been exclusively through a chat interface, you should know that AI tools can now carry out longer, multi-step tasks. This shift is reflected in the rapid growth of coding agents such as Codex and Claude Code. Availability and usage limits vary by product, surface, and plan; Codex currently has an entry point on the ChatGPT Free plan as well as higher-usage paid plans.

The AI landscape is changing fast. To keep up, I recommend following Ethan Mollick's Substack. If you want to swim in the deeper end of the pool, check out Zvi Moshowitz's Substack, Don't Worry About the Vase.

To learn more about AI and how to use it, check out the courses offered by OpenAI Academy and Anthropic Academy.

7.2.2 Coding with AI

One place AI really shines is coding. Chat assistants can answer a question or explain an error. An *agentic* coding tool can work across an entire project: it reads the relevant files, edits and runs code, checks the result, and reports what it changed. You describe the goal in plain English and remain responsible for reviewing the work. Whether the tool asks before each edit or proceeds automatically depends on the permission mode you select.

The landscape moves fast; as of mid-2026, the main options fall into two groups.

Built into your IDE. The assistant works right where you write your code:

- Posit Assistant: Built into Positron as its default AI experience. It can use context from your live data-science session, including loaded data, plots, and console history. You configure one of its supported model providers.
- GitHub Copilot: The most widely available assistant. It works in VS Code and Positron and, as an opt-in, in RStudio Desktop. It suggests code as you type and can also work as an agent. Verified students can often use it free through GitHub Education.

Standalone coding experiences. These work with a whole project rather than living inside the editor where you type R code:

- Claude Code: Anthropic's agentic coding tool, available in desktop and web experiences. Open your project and describe the task. If you want to review a proposed approach before any edits, select Plan mode.
- OpenAI Codex: OpenAI's agentic coding experience in the ChatGPT desktop app. It can work directly with a local project, and its permission settings determine what it may do without stopping for approval.

Finally, Cursor is an entire code editor rebuilt around AI. Software developers have embraced it, but for R work you would give up the data-science conveniences of RStudio and Positron.