Lab 2 Instructions

PUBH 523/623

Nicky Wakim

> ❗ Important
>
> This lab is **ready**! (Nicky 8/2/26)

Directions

Please turn in your `.qmd` and `.html` file on Brightspace. Please use the same naming convention as practice assignments: `lastname_firstinit_lab_02.qmd` and `lastname_firstinit_lab_02.html`.

You can download the `.qmd` file for this lab here.

The above link will take you to your **editing** file. **Please do not remove anything from this editing file!!** You will only add your code and work to this file.

Purpose

The purpose of this lab is to prepare our data for further investigation using several data transformations. We will filter to our target subpopulation, select only the variables relevant to our project, and create new variables that combine information from multiple source columns. **This is meant to take your practice on data transformation one step further: translating written human language to the needed action, function, and arguments.**

Lab activities

The following table will help you identify the appropriate population, variables, and composite score based on your selected project question from Lab 1.

#	Project Question	Population	Population variables	Main variables	Suggested additional variables	Composite score
1	For individuals under the federal poverty line, how do food support systems help individuals' food security?	People under 100% federal poverty line (FPL) and received free groceries or meals in the past 12 months	FPL_LT50, FPL50_100	FOOD_INSEC, Q26D_1, Q26D_2, Q26D_3, Q26D_4	Q23, PPETHM, Q37, NUM_CHILD_HH, Q25, Q25A, Q26, Q26A	Composite score of ease of access to free groceries or meals (combined score from Q26D_)

#	Project Question	Population	Population variables	Main variables	Suggested additional variables	Composite score
2	For individuals who identify as LGBTQ, how does trust and/or fair treatment in healthcare effect self-reported health?	People who identify as LGBTQ and have a healthcare provider. (You can change this population if you want. For example, you may want to focus on how trust and fair treatment may improve health for people with larger bodies. Check out Q70F for options!)	`LGBTQ`, `Q108`	`Q108B_2`, `Q108B_1`, `Q27`	`Q108`, `Q70F_4`, `Q70F_5`, `PPETHM`, `Q28O`	Level of trust, courtesy, and respect around treatment (combined score from `Q108B_1` and `Q108B_2`)

#	Project Question	Population	Population variables	Main variables	Suggested additional variables	Composite score
3	For individuals with school-aged children and below the federal poverty line, how do benefits received effect their mental health?	People who have at least one child (aged 5-18 years old) in the household and are under the 100% FPL	`ANY_CHILD_5_18`, `FPL_LT50`, `FPL50_100`	`Q30_1`, `Q30_2`, `Q30_3`, `Q30_4`, `Q30_5`, `Q30_6`, `Q53_A`, `Q53_B`, `Q53_C`, `Q53_D`, `Q53_E`, `Q53_G`, `Q53_H`, `Q53_I`, `Q53_J`, `Q11`, `Q31_4`	`NUM_CHILD_HH`, `Q1`, `Q37`, `PPETHM`, `Q27`	Total mental health (combined scores from `Q30_`), Total benefits (combined scores from `Q53_`, `Q11`, and `Q31_4`)

Import your saved data from Lab 1

> **!** Task
>
> From the exported data file you created in Lab 1, import your data into R using `import()` from the `rio` package. Save this as a new object called `wbns_00`. Print out the number of rows and columns to verify the import.

Filtering to Your Target Subpopulation

To implement an asset-focused analysis, we must first isolate the subpopulation relevant to your project using `filter()`.

- **For Project 1:** Filter to respondents under 100% of the federal poverty line by keeping rows where `FPL_LT50` **or** `FPL50_100` equal `"Yes"`.

- **For Project 2:** Filter to respondents who identify as LGBTQ **and** who have a healthcare provider (filter out `Q108` answered as "No")
- **For Project 3:** Filter to respondents who have at least one child aged 5-18 in the household (`ANY_CHILDHH_5_18 == "Yes"`) **and** who are under 100% of the federal poverty line (`FPL_LT50` **or** `FPL50_100` equal `"Yes"`).

> **!** Task
>
> Write a pipeline that filters the dataset down to your specific population of interest. Save this as a new object called `wbns_01_pop`. Print out the number of rows before and after filtering to verify the operation.

> Caution
>
> Please make sure you are referencing the PDF with the questionnaire when filtering out your population. There are important descriptions and caveats in the questionnaire!

Selecting Project Variables

Your dataset contains hundreds of variables. For project clarity, reduce your data frame to only the variables in the informational table above. This means keeping ID variables, the population variables, the **main** variables, and suggested additional variables. If you are interested in other variables, feel free to include 0-3 additional variables not listed.

> **!** Task
>
> Use `select()` to keep the variables relevant to your project. Save this as a new object called `wbns_02_pop_vars`. Print out the number of columns before and after filtering to verify the operation.

Quick browsing of data

Using `skim()` or another summary function, take a quick look at your data to see what you are working with. This will help you identify any missing values or refusals that may need to be handled in your new variable construction.

> **!** Task
>
> Using `skim()` or another summary function, take a quick look at your data to see what you are working with.

Take note of any potential issues with the data. Are there many missing values? Do categories align with the codebook? Are there any values that are out of range? Are there any refusals that need to be handled? Write a brief paragraph summarizing your findings.

Rename variables for clarity

Most variables in the WBNS dataset are named after their survey question code (e.g. `Q108B_2`), which isn't very informative on its own and is easy to mix up as your pipeline grows. Renaming your selected variables to short, descriptive names now will make the rest of your code easier to read and less error-prone.

! Task

Use `rename()` to give your **main** and **suggested additional** variables clear, descriptive names (for example, renaming `Q108B_2` to `trust_treatment`). Save this as a new object called `wbns_03_renamed`. Use `glimpse()` to confirm your new variable names.

Remove NA's

During your "Quick browsing of data" step, you likely noticed some missing values and/or "Refused" or "Don't know" responses. These need to be handled *before* you calculate your composite score, since `NA` (and non-numeric strings like `"Refused"`) will disrupt arithmetic and summary functions — for example, `5 + NA` returns `NA`, not `5`.

! Task

Recode any "Refused" or "Don't know" responses to `NA` (if they aren't already), then use `drop_na()` (or `filter()`) to remove rows with missing values in your key variables. Save this as a new object called `wbns_04_no_na`. Print the number of rows before and after this step to verify the operation, and write 1-2 sentences describing how many rows you lost and whether that seems reasonable.

Constructing the Composite Score

> **Caution**
>
> Please make sure you are referencing the PDF with the questionnaire when creating your composite score. There are important descriptions and caveats in the questionnaire!

Your project must create a brand-new variable that combines information from at least two source columns using `mutate()`. Please see the table above for the variables to combine, and the project-specific guidance below.

Regardless of your project, the general recipe is the same:

1. Confirm the response scale for each item in the questionnaire (is it Yes/No? A 4- or 5-point scale? What order are the levels in?).
2. Convert each item to a usable numeric or binary form with `as.numeric()` or `if_else()`/`case_when()`. Do this *after* you've handled `NA`/refusals, so they don't get silently converted into a fake numeric value.
3. Combine (e.g., add) the converted items into one new column with `mutate()`.
4. Sanity-check the result: print a `summary()` and confirm the range and direction make sense (e.g., does a higher score mean *more* of what you think it means?).

> **Task**
>
> Create your new composite scores using `mutate()`, following the guidance above for your project. Save this as a new object called `wbns_05_composite`. Display a summary of each combined variable, and briefly explain (1-2 sentences) whether the range and direction of your composite score make sense.

Below are project-specific instructions for your composite score:

Project 1: Ease of accessing free food

Your composite combines the four `Q26D_` items, which each ask how difficult or easy a different aspect of accessing free groceries/meals was (finding a place, finding transportation, variety of food, and getting there in a reasonable time). Per the questionnaire, the response scale is **Very difficult (1), Difficult (2), Neither difficult nor easy (3), Easy (4), Very easy (5), Don't know (6)**.

- Recode any value of 6 ("Don't know") to `NA` before converting to numeric.
- Convert each item to numeric with `as.numeric()`. Since 5 = "Very easy" and 1 = "Very difficult," higher values already mean *easier*.
- Add the four converted items together into one column (e.g., `ease_free_food`)

  - With 4 items on a 1-5 scale, the possible range is 4 (very difficult across the board) to 20 (very easy across the board). Check that your summary falls in this range.

Project 2: Trust, courtesy, and respect

Your composite combines `Q108B_1` ("I am treated with courtesy and respect by my personal health care provider") and `Q108B_2` ("I trust my personal health care provider"). Per the questionnaire, both share the same response scale: **Never true (1), Rarely true (2), Sometimes true (3), Often true (4), Always true (5)**

- Convert both items to numeric with `as.numeric()`. Higher values already mean *more* trust/courtesy/respect — no reverse-coding needed.
- Add the two converted items together into one column (e.g., `trust_courtesy_respect`), which will range from 2 (never true on both) to 10 (always true on both).
- *Optional:* if you pull in `Q70F_4` and `Q70F_5`, you could build a second, related composite capturing whether the respondent felt judged unfairly based on gender identity or sexual orientation, and compare it against your trust/courtesy/respect score.

Project 3: Mental health and total benefits — read this one carefully!

This project asks for *two* composite scores, and both have a wrinkle worth slowing down for. Don't try to do it all in one `mutate()` — build it up piece by piece.

Mental health composite (from `Q30_1`–`Q30_6`):

These six items ask how often in the past 30 days the respondent felt nervous, hopeless, restless/fidgety, so sad that nothing could cheer them up, that everything was an effort, or worthless. Per the questionnaire, it runs **All of the time (1), Most of the time (2), Some of the time (3), A little of the time (4), None of the time (5)**. That means a *lower* numeric code = more frequent distress (worse mental health), and a *higher* numeric code = less frequent distress (better mental health).

- Decide up front which direction you want your composite to run, and say so explicitly in your write-up:
  - Sum the raw numeric codes as-is, and your total will run low = worse, high = better.
  - Or reverse-code each item first (e.g., `6 - as.numeric(Q30_1)`) so your total instead runs low = better, high = worse, which many people find more intuitive.
- Either direction is fine for this lab: just be consistent across all 6 items and clearly state which way you went.

Total benefits composite (from `Q53_A`–`Q53_J`, `Q11`, and `Q31_4`) — this is the tricky part:

The `Q53_` series asks about several *different* benefit programs, each a separate Yes/No/Don't know question — not a scale. Two of them have **built-in skip logic** in the actual survey that will create a lot of misleading missing data if you don't account for it:

- `Q53_B` (Medicaid/CHIP) is *only asked* of people who did **not** already report Medicaid/CHIP coverage back in `Q31` (item d, "covered"). If someone already said "Covered" on `Q31_4`, they were skipped past `Q53_B` entirely — so `Q53_B` will show up missing for them, even though they clearly do have this benefit.
- `Q53_C` (housing assistance) has the same issue with `Q11` ("is your household paying lower rent because a government program is paying part of the cost?"): anyone who already said "Yes" on `Q11` was skipped past `Q53_C`.
- If you just `drop_na()` on `Q53_B` or `Q53_C` without combining them with `Q31_4`/`Q11` first, you will incorrectly throw out (or miscode) people who actually have these benefits.

Steps:

1. **Combine the skipped pairs.** For CHIP and housing, combine the two questions into a single 0/1 indicator, counting the person as receiving the benefit if *either* question says yes: e.g., `chip = if_else(Q53_B == "Yes" | Q31_4 == "Covered", 1, 0)`. Think carefully about what should happen if *both* are missing.
2. **Convert the remaining `Q53_` items** (SNAP, SSI, SSDI, cash assistance/TANF, unemployment insurance, free/reduced school meals, and WIC) to simple 0/1 indicators (1 = "Yes", 0 = "No").
    - Free/reduced school meals is only asked of households with a child aged 5-18 — since your Project 3 population already requires `ANY_CHILDHH_5_18 == "Yes"`, this item should be answered for everyone in your filtered sample.
    - WIC, on the other hand, is only asked of households with a child aged 0-5 — which your population filter does **not** guarantee. WIC may be legitimately missing for many people in your sample (because they weren't asked, not because they refused). Decide whether to include WIC in your total at all, and if you do, decide how to treat those missing values (e.g., recode missing WIC responses to 0 rather than dropping the row).
3. **Add every 0/1 benefit indicator together** into `total_benefits` — a count of how many *different* types of benefits the household receives (not how many `Q53_` questions they answered "Yes" to).
4. *Optional:* bucket `total_benefits` into a categorical variable (e.g., "0", "1", "2", "3", "4+") for easier visualization later.

Export datasets

It's important to save your work! We want to export the datasets we created along the way so we can use them in future labs and in our final project. Use `export()` in `rio` to save the following datasets in a `.rda` file in your `data_project` folder:

- `wbns_01_pop`
- `wbns_02_pop_vars`
- `wbns_03_renamed`
- `wbns_04_no_na`
- `wbns_05_composite`

> **!** Task
>
> Save your datasets in a `.rda` file in your `data_project` folder. Name the file something that will help you identify it later, like `wbns_data_lab_02_work.rda` or `wbns_data_02_pop_comp.rda`