End to End Case Assignment

As usual, I've provided some starter code in the `end_to_end_case.R` file. You can write all of your code in that file from within RStudio, then save your changes, commit, and push those changes to GitHub before the deadline to submit.

Assignment Instructions

Well, folks. We've made it. You now have everything you need to work with data in R. And that's pretty cool.

This assignment will bring all of those skills together in a real-world data preparation scenario. I've taken a different approach in putting this assignment together. Unlike the previous case assignments, I'm not going to ask you to do a series of separate problems. Instead, I'm giving you real data and asking you to get it to a format and state of cleanliness that would support a machine learning model.

In this assignment, you'll be acting as the data analyst assigned to get the data ready for analysis. My intention is for you to experience what it's like in the real world to be given a messy, raw dataset and get it ready for use. The data you'll be using is actual data that I've scraped from a website, for which I've subsequently built a cleaning script. Your assignment is to build a similar cleaning script that produces a dataset with requirements that I'll describe in detail.

Recall that the last couple of case assignments have not included messiness or "gotchas" in the data. As promised, I haven't been quite as kind in this assignment. You won't see any issues that you haven't seen in previous case assignments, but you will definitely want to **be defensive** as you work through your cleaning operations. A few of the things that you'll be handling have been manufactured by me, but most of them were a part of the data I scraped and are just good examples of what happens with real data. I'll include specific hints or requirements in my description below that will make it clear how the various "issues" should be handled, but my most important advice is this: be defensive at each stage, double-check that your cleaning operations are doing what you think they are doing, and, above all, just demonstrate careful attention to detail.

Description of Raw Data Files

You've been given two raw data files, one in wide format and one in long format. Together, they describe a set of mobile home parks that were listed for sale on a real estate investment website. The wide data contains the main lists of the properties, with property name and location information. As is common, however, the listings were very inconsistent in how much detail was included by the listing agent. Some properties have extensive detail, and others consist mostly of the listing location without much else. Thus, I used a long data storage format to capture those additional details. The `id` column in each dataset is a GUID that I created, which means it can be trusted to link the data between the two tables.

Your Mission

Our business manager would like to use this dataset to decide which mobile home parks are good investments. She needs the data cleaned up, and she has asked for a spreadsheet with one row per property and a column for each of the attributes of that property. Thus, your assignment is to do whatever needs to be done in order to get the data into a single, clean, wide dataset that is ready for her analysis.

Because this is a real-world data prep scenario, I'm purposefully not giving you step-by-step instructions that will be graded separately along the way. Nearly all of your grade for this assignment will be based on

three final, cleaned datasets (one each for the wide and long data, and one representing the final, joined data), just like it would be in the real world. **However**, please don't get nervous about this being an "all-or-nothing" grade. Below, I will clearly describe both the requirements as well as the final datasets in as much detail as possible without giving away too much. You will be given every opportunity to earn partial credit on every possible gradable aspect of the final datasets, and we will write the grading logic to be as generic as possible. Thus, as long as you meet the requirements below and as long as your final datasets fit the descriptions below, you'll get *most of* the available credit.

That said, for someone to get *full* credit on this assignment, the cleaning will have to be done as defensively and carefully as possible. You'll need to use common sense and clean the data as if you were being paid by our business manager to do so. There are a small handful of situations in the data where something is *clearly* wrong, and you'll be expected to make a reasonable judgement to fix it. (In most situations, of course, it will be difficult to know whether something is correct or not, and those sorts of things can be left as they are, and our grading will ignore them.)

Requirements

As you go about cleaning this data, keep the following requirements in mind:

1. The wide dataset contains a certain number of properties, and the final dataset should have exactly one row per property. Note again that the `id` column in both datasets was manufactured by me to *uniquely identify each property*.
2. Each of the columns present in the wide dataset should have no missing data. To be clear: by the end of your script, there should be zero `NA` s in any of the 8 columns present in the original wide data that you read in.
3. By contrast, the long data that will eventually be pivoted wider will end up being very sparse (i.e., with lots of `NA` s), and that is **completely expected**. You don't need to worry about any missing data in the long data, with one important exception, described in the next item.
4. There are exactly five columns in the long data whose *presence* in the long dataset indicates that they apply to the property in question. They are binary indicators, and should be denoted with a `1` in the final dataset, indicating that the property has one of those attributes. For example, one of these attributes is an indicator of whether the property has an on-site laundromat. The corresponding `has_laundromat` column in the final dataset will have a value of `1` for any property that has an on-site laundromat. There are four other attributes that should be treated in the same way, and each should be labeled as `has_` and then the attribute. (I'm specifically *not* specifying whether the other values in these five columns need to be `0` vs `NA`. Either way is fine.)
5. You'll notice in the descriptions of the datasets below that all of the final columns are the result of running each through the `janitor::clean_names()` function (i.e., they are all in beautiful, consistent, snake case format). You should run the datasets through the `clean_names()` function at an appropriate time to accomplish that.
6. The columns clearly contain data with different data types, and you will need to convert each column to the appropriate datatype as needed. More specifically, anything that is clearly a number should be converted to a numeric column, and anything that is clearly a date column should be converted to a date column. (Zip codes, by the way, are generally not considered true numbers, but you can convert those if you'd like to. The grader won't care either way.)
7. In addition to converting the numbers as described in the previous point, any numeric column that has a unit associated with it (e.g., currency, percent, acreage) should have the units embedded in the column name to avoid any confusion. You'll notice in the columns I'm sharing from the final

dataset, for example, the columns `price_usd`, `total_occupancy_rate` and `size_acres`. It's a good idea to always label a column with the unit if there is opportunity for confusion.
8. Specific to the percent/rate columns, they should be converted to math-friendly decimals (with values between 0 and 1).
9. The long dataset contains information about `purchase_method`, which describes the payment options that the seller is willing to accept. This column should be in the final dataset, but because some sellers are willing to accept multiple payment options, you should also derive exactly 4 additional "dummy-coded" variables from that column, representing binary indicators as to whether the seller of a property is willing to accept each payment option. These 4 columns will all have exactly 3 possible values: `1`, `0`, and `NA`. (And all of them will have at least some `NA`s.)
10. After you join the data together, you'll also need to add two additional columns that the business manager has requested: (a) `age_years`, calculated as the difference in years between today's date and the year the property was built, and (b) `price_per_lot_usd`, calculated as the price of the property divided by the number of mobile home lots (specifically, the `number_of_mh_lots` column).
11. After doing all of the above, you'll create a small batch of visualization plots that will help you QA the data and make any adjustments before you finalize your submission. This final QA pass will potentially reveal some things that you missed, and you may need to make any cleaning adjustments to your dataset. Just make sure that those additional operations are applied to the final joined dataset, as described in the next point.
12. You should clean the wide data separately from the long data, saving each using the object names specified below (`wide_clean` and `long_clean_pivoted`, respectively) before joining them together and saving the joined dataset using the object name specified below (`together_clean`). However, any final cleaning that you do as a part of your QA checks of the data at the end (including in building the visualizations) can be done on the final, joined dataset (i.e., `together_clean`). In other words, we'll only be checking the two separate datasets (`wide_clean` and `long_clean_pivoted`) in terms of their row count and the expected columns being present. All of the main grading of things like datatypes, extent of missingness, and other cleaning things will be evaluated against the final, joined dataset (`together_clean`). (This is to ensure that you don't need to worry about retroactively applying any "touch ups" to the other two datasets.)

> *Note: As you build your cleaning script, you are welcome to save the data out in as many objects as you like. (In fact, I recommend doing so at each main stage of cleaning so that you can iteratively improve your cleaning as you learn more about the dataset and make adjustments to account for different things.) They will all be ignored by the grader. The only objects that will be graded are the three final datasets as well as the five simple visualizations requested below.*

The Final Datasets

Your grade on this assignment will be determined by the extent to which your final datasets meet the requirements above *as well as* the extent to which you have carefully, methodically cleaned and prepared this dataset for analysis. Again, what this means is that, in addition to following the instructions above, you'll be expected to do some light checking of the dataset for some "gut checks" to see whether the columns make sense in terms of their datatypes, range, amount of missingness, etc.

Here are a few things to keep in mind while doing your QA / gut checks:

- Don't go crazy with these gut checks and stress about finding every single problem with the data. It's real-world data and it's still going to be a bit messy after you're done with it. But there are a handful of super obvious issues that you should identify and fix. Those issues should be easy to

identify by doing a light pass and asking "does the data in this column make sense, given what I can see?".
- If something is clearly off (as in, it's a data entry error), then you should either (1) fix the error with an obvious/reasonable replacement, or (2) if the error is clearly an error and there is no obvious replacement, you should convert the value to be a missing value.
- In addition, you'll see below that I'm asking you to create a handful of visualizations that will help you identify a few things that you may have missed. Since I'm giving you the plots themselves (in the `/plots` directory), you can ensure that they look the same as the plots that you create. If they are way off (and some of them likely will be at first!), then you can go back to the data and find the cause of those issues and take care of them.
- Note that the `property_name` column contains raw property name information, and there is not really a way to clean that one up. You can essentially ignore it in your QA because it won't be graded other than to make sure that it's there and named appropriately, etc.

The Wide Data

You should QA and adjust any issues you find in the wide data, following the requirements above. The result of your cleaning and fixing should then be saved to an object called `wide_clean` before joining with the other data.

> *Note: You might need to use other resources like, for example, Google to fill in some of the holes in the wide data. Did you know that Google Maps allows you to search by GPS coordinates?*

The `wide_clean` should have one row per property, and contain columns that pass the code test in the `Naming Checks` section at the bottom of your script.

> *Hint: There is a section at the bottom of the starter script that includes a series of tests designed to help you catch any naming issues before you submit your assignment. If you fail any of those tests, you should be given some hints as to which ones are missing.*

1. Ensure that you have a tibble called `wide_clean` in your environmnent

---

The Long Data

The long data will take considerably more work to QA, format, and fix. You should pay careful attention to the datatypes and naming requested in the requirements section above. You will eventually be required to pivot the data to wide format in preparation for joining to the `wide_clean` data.

The `long_clean_pivoted` should have one row per property (and should have the same row count as the `wide_clean` data). It should also contain columns that pass the code test in the `Naming Checks` section at the bottom of your script.

2. Ensure that you have a tibble called `long_clean_pivoted` in your environmnent

---

The Joined Data

After cleaning and preparing the two datasets above, you should then join the two using the `id` column to produce the final dataset. After doing any additional QA or cleaning, this dataset should be saved as `together_clean`.

3. Ensure that you have a tibble called `together_clean` in your environmnent

---

Visualizations

You will need to produce a total of 5 visualizations, partly to demonstrate that you know how, but also so that you can learn how visualizations can help identify formatting or cleaning issues. I've included example plots in the `/plots` directory in the repository, to which you can compare your plots based on your data.

Plot 1

Have a look at the distribution of the `price_usd` column using a density curve. (I used a "blue" fill and 0.6 alpha in my plot.) As with most pricing data, you'll see that the pricing is very skewed. We'll give it another look in the next plot.

4. For now, add the `theme_bw()` theme as well as axis labels and a title, and save the result to `plot_1`

Plot 2

It sometimes helps to look at heavily skewed columns using a log scale. Doing so can reveal anything that looks "out of the ordinary", given the skew of the data. Adding a `log()` calculation on the fly (meaning that you don't need to save a new column), add a new plot using the same setup as that of `plot_1`, but this time showing the density curve of the log-scaled price. You should see the data in a much more normal format, which you can compare to my `plot_2.png` in the `/plots` directory. (If your log-sclaed plot is similar but clearly different, you might want to have another look at the `price_usd` column to see if its contents really make sense.)

5. Add the `theme_bw()` theme as well as axis labels and a title, and save the result to `plot_2`

Plot 3

Now let's look at the distribution of the listings over time. Most of the data in this dataset comes from 2022, so filter the data to anything after Jan. 1st, 2022, then create a histogram of the properties over time. (You can just use the default of 30 bins for the histogram rather than going through the complicated process of figuring out the time-based binning.) I used the "red" color as my fill and an alpha of 0.6.

6. Add the `theme_bw()` theme as well as axis labels and a title, and save the result to `plot_3`

Plot 4

Now let's examine the property age information (derived from the `year_built` column). Without any filtering, create a histogram that shows the distribution of the `age_years` column. I used the "green" fill and an alpha of 0.6, and you can again just use the default of 30 bins. (Again, if your histogram looks different from mine, you might want to investigate why and do some additional cleaning.)

7. Add the `theme_bw()` theme as well as axis labels and a title, and save the result to `plot_4`

Plot 5

One final plot! Let's look at the various purchase methods available in our dataset. Because there are so many listings that don't have purchase method information, it's helpful to split those out separately. So we'll use a facet based on whether the purchase method is there or not. Add an on-the-fly column (meaning there is no need to save it to the tibble) called `has_purchase_method` that is "Yes" when `purchase_method` has a value, and "No" when it's `NA`. Then pipe that to create a plot (no filtering of the data needed) that shows a bar chart representing the count of properties in each state, and then split out the purchase methods in a stacked fashion. Finally, facet the data based on the `has_purchase_method`

column you added. Because the faceting here is best enjoyed with labels which we haven't talked about, I'll just give you the code for the facet command: `facet_wrap(~has_purchase_method, ncol = 1, labeller = label_both)`.

8. Add the `theme_bw()` theme as well as axis labels and a title, and save the result to `plot_5`

Final Cleanup and Submission

Final Run

Please do the following to make sure your code runs without errors and passes the naming checks by doing the following:

1. Restart your R session (Session >> Restart R).
2. Run your entire script from the beginning, watching for any errors along the way.
3. After your script is finished, make sure that you pass all tests in the code in the "Naming Checks" section at the bottom of your script.

Housekeeping

Once you have completed the steps in the prior section, please check all of the following and make adjustments as needed. (Failure to do the housekeeping steps below will likely cause an error in our grading process, and that will make it hard to give you the right credit for your hard work.)

1. If you used the `setwd()` command near the beginning of the script, please COMMENT OUT that line before committing and pushing your code.
2. Please also comment out any code where you are either using the `View()` or `glimpse()` functions. These cause issues with our grading procedures.
3. If you installed any new packages as you completed the assignment (using `install.packages()`), please comment those installation commands out as well.
4. Lastly, please ensure that you have **NO** absolute references in your code that are not commented out. (An absolute reference is something like: `/Users/YourName/Documents/GitHub/...` or `C:/GitHubProjects/...`.) These also throw errors and make it hard for us to grade your work.

Save, Commit, Push

You're now ready to do all three of the following:

1. Save your script.
2. Do a final commit with your git tool to stage your work for submission.
3. Push your changes up to your repository. And you're done!