Exercises for 'Working in the Tidyverse'

Desi Quintans, Jeff Powell

30/04/2019

Exercises

01. Importing and exporting data

01-01 (Required)

Import the `light_trap` data and the write it to a file.

First, run this code block:

```
list_of_files <-
    list.files("_data/light_trap/", pattern = "csv", full.names = TRUE)

light_trap <-
    map_dfr(.x = set_names(list_of_files),
            .f = ~ read_csv(.x, col_types = cols(.default = "c")),
            .id = "source_file")
```

Then use `readr::write_rds()` to write `light_trap` to a file called `_output/01-01_light_trap.rds`.

01-02 (Required)

The main dataset we will be analysing throughout this workshop is inside the subfolders of `_data/wood_blocks/`.

1. Use `list.files()` to generate a list of **the four .csv files** in this folder structure. Make sure you are keeping their full names.
    - *Hint:* Look at the `recursive` argument.

01-03 (Required)

Work through the list of files you just generated.

1. Import `site_data.csv` by itself.
    - **Be sure to import it all as Character types, as in the examples.**
2. Write `site_data.csv` to a file called `_output/01_wood_site_data.rds`.
3. Import and row-bind *all other files in the list that are* ***not*** *`site_data.csv`*. Include an `id` column.
4. Write the resulting dataframe to a file called `_output/01_wood_blocks.rds`.

02. Reshaping and completing

02-01 (Required)

The `wood_blocks` dataset from the previous chapter needs a little bit of work. First, its column names need to be standardised.

1. Import `_output/01_wood_blocks.rds` from the previous exercise.
2. Use `janitor::clean_names()` to standardise the column names.
3. Save the resulting data frame as `_output/02_01_wood_blocks.rds`.

02-02 (Required)

Continuing with the `wood_blocks` dataset, the column `block_id` actually contains two distinct bits of information that can be separated into their own columns. A value like `C24` contains the Treatment (`"T"` or `"C"`, for Treatment versus Control) and the ID number of the wood block (`24`).

1. Import `_output/02_01_wood_blocks.rds` (or use the data frame you generated in Exercise 02-01).
2. Use `separate()` to split the `block_id` column into two new columns, called `treatment` and `wood_id`. Don't keep `block_id`.
3. Save the resulting data from as `_output/02_02_wood_blocks.rds`.

02-03

Experiment with spreading into a wide shape.

1. Import `_output/01_light_trap.rds` from the previous exercise using `readr::read_rds()`.
2. Randomly choose 50 rows from it and save them to a new variable name (hint: use `dplyr::sample_n()`).
3. Use `spread()` to practice spreading this long data frame. Notice how, when you spread by `order`, there is always an `NA` in one of the columns because a family cannot belong to two different orders.
4. Use the `fill` argument in `spread()` to change `NA` to different values.
5. Do not save the output to a file.

02-04

Experiment with gathering into a long shape.

1. There is a built-in dataset called `iris`, which you can access by simply typing `iris`. Inspect it.
2. Try using `gather()` to gather all of the measurements together.
3. Try to do Step #2 without manually naming every column that you want to gather.
    - Hint: Look at `?tidyselect::select_helpers`.
    - Hint: The `:` operator also works.

03. Joining data frames together

03-01 (Required)

Merge two tables together.

1. Import `_output/02_02_wood_blocks.rds`, which you prepared in the last chapter.
2. Import `_output/01_wood_site_data.rds`, which you prepared in Module 1.
3. Use `left_join()` to merge the site data into the main `wood_blocks` dataset.
    - Hint: If you get the error *"incompatible types (integer / character)"*, it is because you did not import the data frames as Character columns in the chapter 'Importing data'.
4. Inspect the result. Notice that `left_join()` has repeated the three rows of `01_wood_site_data.rds` wherever they matched the `site_id` in `02_wood_blocks.rds`.
5. Save the result as `_output_03_wood_blocks.rds`

04. Choosing and renaming columns

04-01 (Required)

Omit some of the columns from the `wood_blocks` dataset.

1. Import `_output/03_wood_blocks.rds`.
2. Omit these columns:
    - `source_file`
    - `date_deploy`, `date_collect`
    - `deploy_image`, `collect_image`
    - `wet_weight_pre_drill`, `wet_weight_post_drill`
3. Write the resulting data frame to `_output/04_wood_blocks.rds`.

05. Choosing rows

05-01 (Required)

We will continue cleaning our main dataset, `wood_blocks`.

1. Import `_output/04_wood_blocks.rds`.
2. Drop any rows that are duplicated by `treatment`, `wood_id`, `plot_id`, and `site_id`.
3. Drop rows that contain `NA` in any column **except for the `end_weight` column**, because we will compute that later.
4. Arrange the data frame by `site_id` and `init_weight`, with heaviest wood blocks in the site listed first.
5. Save the resulting data frame to `_output/05_wood_blocks.rds`.

05-02

1. In the built-in dataset `iris`, find all observations of the species *setosa* where the petal length was $>$ 1.5, but the sepal length was $<$ 5.0.

05-03

1. In the built-in dataset `starwars`, subset the data frame so that only the heaviest member of each species is kept.

06. Editing and creating columns

For the exercises in this module, we will continue to clean up our `wood_blocks` dataset.

06-01 (Required)

In the chapter 'Importing data', we imported every column in the `wood_blocks` dataset as a Character column so that all of the input spreadsheets could be row-binded without errors. The first step to doing that is to ensure that `NA` values as properly encoded.

1. Import `_output/05_wood_blocks.rds` and inspect it with `View()`.
2. Notice that although the `weight` columns (`init_weight:end_weight`) contain mostly numbers, one of the sites has used the string `"ND"` to indicate missing data.
3. Use `mutate_at()` to convert `"ND"` in these columns to `NA`.
    - Hint: You will need to write an anonymous function.
    - Hint: Remember that `NA` comes in several different data types. See `?NA` for more.
4. Write the resulting data frame to `_output/06_01_wood_blocks.rds`.

06-02 (Required)

Now that we have converted missing values to `NA`, we can begin converting the columns to the correct data types. The columns in this exercise can be coerced directly with `as.integer()` and `as.numeric()` and don't need to be recoded.

1. Import `_output/06_01_wood_blocks.rds` (or use the data frame that you created in Exercise 06-01).
2. Use `mutate()` to convert these columns into Integer type:
    - `wood_id`
    - `site_id`
3. Use `mutate_at()` to convert these columns into Numeric (AKA Double) type:
    - `init_weight`, `wet_weight_pre_drill`, `wet_weight_post_drill`, `end_weight_post_drill`, and `end_weight`
    - `lat`, `lon`
4. Use `mutate_at()` to convert these columns into ordered factors, with levels `0, 1, 2, 3, 4` and default labels:
    - `damage_fungal`, `damage_termite`
5. Save the result to `_output/06_02_wood_blocks.rds`.

06-03 (Required)

These next columns need to be recoded. from `"Y"` and `"N"` to `TRUE` and `FALSE`:

1. Import `_output/06_02_wood_blocks.rds` (or continue with the data frame from Exercise 06-02).
2. Recode these columns so that `"Y"` is recorded as `TRUE` and `"N"` is recorded as `FALSE`:
    - `termites`, `insects`, `fungi`
3. Recode the `treatment` column so that `"T"` is replaced with `"Treatment"` and `"C"` is replaced with `"Control"`.
4. Save the result to `_output/06_03_wood_blocks.rds`

06-04 (Required)

Now we will use our data frame to do some calculations.

1. Import `_output/06_03_wood_blocks.rds` (or continue with the data frame from Exercise 06-03).
2. Some of the rows in `end_weight` are `NA`. Replace those `NA`s with the value that is held in `end_weight_post_drill`.
3. Calculate how much wood the block lost while it was in the field (`init_weight - end_weight`) and store it in a new column called `weight_lost`.
4. Remove the `end_weight_post_drill` column.

5. Save the result to `_output/06_04_wood_blocks.rds`.

07. Grouping and summarising

07-01 (Required)

We will reduce our `wood_blocks` dataset to prepare it for plotting.

1. Import `_output/06_04_wood_blocks.rds`.
2. Group the data frame by `site`, `wood_id`, and `treatment`.
3. Summarise the `mean`, `median`, `min`, and `max` of these columns:
   - `init_weight`, `end_weight`, `weight_lost`
   - It is alright to lose the other columns.
4. Save the result to `_output/07_01_wood_blocks.rds`.

07-02

In our `wood_blocks` dataset, how many wood blocks experienced damage from termites, insects, and fungi?

1. Import `_output/06_04_wood_blocks.rds` (this is the same dataset that you imported at the start of Exercise 07-01, **not** the final result of 07-01).
2. Make a table of counts that shows each damage type and how many wood blocks were affected by it.
   - You may find that not all combinations of `termites Ã- insects Ã- fungi` are represented in your new data frame. How can you fix this so that all combinations are there?
3. Do not save the output.

07-03

Let's test our knowledge of editing columns and reshaping data. Instead of making a table of counts, we will create a contingency table (i.e. a matrix of counts).

1. Import `_output/06_04_wood_blocks.rds` (this is the same dataset that you imported at the start of Exercise 07-01, **not** the final result of 07-01).
2. Termites are insects, so make sure that the `insect` column is `TRUE` if termites were present in a wood block.
3. Create this result:

```
## # A tibble: 2 x 3
##   insects fungi_FALSE fungi_TRUE
##   <fct>         <dbl>      <dbl>
## 1 FALSE           104         10
## 2 TRUE              4          1
```

08. Graphing with ggplot2

08-01 (Required)

Make some graphs with the `wood_blocks` dataset that you have spent this workshop preparing.

1. Import `_output/07_01_wood_blocks.rds`.
2. Make a ggplot of `weight_lost_mean Ã- site`, with the `colour` aesthetic mapped to the `treatment` variable.
3. Add a boxplot geom.
4. Add jittered points (`geom_jitter()`). Jittered points are randomly offset from their true location to reduce overplotting. Change the `width` and `height` of this jittering so that there is no vertical jitter and minimal horizontal jitter.
5. Facet the plot so that each treatment level appears in a separate column.
6. Change the X and Y labels to `"Site"` and `"Weight lost (mean)"`.

08-02 (Required)

1. Import `_output/07_01_wood_blocks.rds`, or use the data you imported in Exercise 08-01.
2. Make a ggplot of `weight_lost_mean Ã- init_weight_mean`, with the `colour` aesthetic mapped to `treatment` and the `shape` aesthetic mapped to `site`.
3. Add points to create a scatterplot.
4. Add a linear model trend line.
5. Facet the plot so that each site appears in a separate column.
6. Change the X and Y labels to `"Starting weight (mean)"` and `"Weight lost (mean)"`.