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 = purrr::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`.

```
write_rds(light_trap, "_output/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.

```
my_list <- list.files("_data/wood_blocks", recursive = TRUE, full.names = TRUE)
```

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`.

```
# The obvious approach is to index the list: my_list[1:3] and my_list[4]. This
# would break if you added another spreadsheet to the folder, but is a little
# safer for stable datasets where the files will no longer change.

read_csv(my_list[4], col_types = cols(.default = "c")) %>%
    write_rds("_output/01_wood_site_data.rds")

map_dfr(.x = purrr::set_names(my_list[1:3]),
```

```
        .f = ~ read_csv(.x, col_types = cols(.default = "c")),
        .id = "source_file") %>%
    write_rds("_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`.

```
read_rds("_output/01_wood_blocks.rds") %>%
    clean_names() %>%
    write_rds("_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`.

```
read_rds("_output/02_01_wood_blocks.rds") %>%
    separate(block_id, into = c("treatment", "wood_id"), sep = 1) %>%
    glimpse() %>%
    write_rds("_output/02_02_wood_blocks.rds")
```

```
## Observations: 360
## Variables: 19
## $ source_file           <chr> "_data/wood_blocks/Australia_Calperum/CL...
## $ treatment             <chr> "C", "C", "T", "T", "C", "C", "C", "C", ...
## $ wood_id               <chr> "01", "05", "23", "30", "28", "02", "27"...
## $ plot_id               <chr> "BS-01", "BS-01", "BS-01", "BS-01", "BS-...
## $ site_id               <chr> "10", "10", "10", "10", "10", "10", "10"...
## $ date_deploy           <chr> "2017_08_21", "2017_08_21", "2017_08_21"...
## $ date_collect          <chr> "2018_08_23", NA, "2018_08_23", NA, "201...
## $ termites              <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ insects               <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ fungi                 <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ damage_fungal         <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ damage_termite        <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ init_weight           <chr> "189.65", "197.19", "172.57", "179.03", ...
## $ wet_weight_pre_drill  <chr> NA, NA, NA, NA, "184.91", NA, NA, NA, "2...
## $ wet_weight_post_drill <chr> NA, NA, NA, NA, "181.65", NA, NA, NA, "2...
## $ end_weight_post_drill <chr> NA, NA, NA, NA, "153.15", NA, NA, NA, "1...
## $ end_weight            <chr> "179.1742", NA, "160.0825", NA, NA, "202...
## $ deploy_image          <chr> "4061", "4061", "4061", "4061", "4062", ...
## $ collect_image         <chr> "168", NA, "168", NA, "169", "169", NA, ...
```

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.

```
light_trap <-
    read_rds("_output/01_light_trap.rds") %>%
    sample_n(50)

light_trap %>%
    spread(order, individuals) %>%
    head()
```

```
## # A tibble: 6 x 6
##   source_file             family       date1  date2  COLEOPTERA LEPIDOPTERA
##   <chr>                   <chr>        <chr>  <chr>  <chr>      <chr>
## 1 _data/light_trap/1992.~ GELECHIIDAE  7/6/92 7/8/92 <NA>       24
## 2 _data/light_trap/1992.~ GEOMETRIDAE  7/9/92 7/9/92 <NA>       7
## 3 _data/light_trap/1992.~ NOTODONTIDAE 7/21/~ 7/23/~ <NA>       3
## 4 _data/light_trap/1992.~ PLUTELLIDAE  7/24/~ 7/28/~ <NA>       3
## 5 _data/light_trap/1993.~ PYRALIDAE    6/7/93 6/10/~ <NA>       14
## 6 _data/light_trap/1994.~ COLEOPHORID~ 9/1/94 9/13/~ <NA>       4
```

```
light_trap %>%
    spread(family, individuals) %>%
    head()
```

```
## # A tibble: 6 x 35
##   source_file order date1 date2 ALUCITIDAE ANOBIIDAE ANTHICIDAE
##   <chr>       <chr> <chr> <chr> <chr>      <chr>     <chr>
## 1 _data/ligh~ LEPI~ 7/21~ 7/23~ <NA>       <NA>      <NA>
## 2 _data/ligh~ LEPI~ 7/24~ 7/28~ <NA>       <NA>      <NA>
## 3 _data/ligh~ LEPI~ 7/6/~ 7/8/~ <NA>       <NA>      <NA>
## 4 _data/ligh~ LEPI~ 7/9/~ 7/9/~ <NA>       <NA>      <NA>
## 5 _data/ligh~ LEPI~ 6/7/~ 6/10~ <NA>       <NA>      <NA>
## 6 _data/ligh~ COLE~ 7/20~ 7/21~ <NA>       <NA>      <NA>
## # ... with 28 more variables: BATRACHEDRIDAE <chr>, CARABIDAE <chr>,
## #   COLEOPHORIDAE <chr>, CORTICARIIDAE <chr>, CRYPTOPHAGIDAE <chr>,
## #   CURCULIONIDAE <chr>, ETHMIIDAE <chr>, GELECHIIDAE <chr>,
## #   GEOMETRIDAE <chr>, GYRINIDAE <chr>, HETEROCERIDAE <chr>,
## #   HYDROPHILIDAE <chr>, LYONETIDAE <chr>, MELYRIDAE <chr>,
## #   MOMPHIDAE <chr>, NOCTUIDAE <chr>, NOTODONTIDAE <chr>,
## #   OECOPHORIDAE <chr>, PLUTELLIDAE <chr>, PTEROPHORIDAE <chr>,
## #   PYRALIDAE <chr>, SCARABAEIDAE <chr>, SILPHIDAE <chr>,
## #   STAPHYLINIDAE <chr>, THROSCIDAE <chr>, TINEIDAE <chr>,
## #   TORTRICIDAE <chr>, YPONOMEUTIDAE <chr>
```

```
light_trap %>%
    spread(source_file, individuals) %>%
    head()
```

```
## # A tibble: 6 x 20
##   order family date1 date2 `_data/light_tr~ `_data/light_tr~
##   <chr> <chr>  <chr> <chr> <chr>            <chr>
## 1 COLE~ ANOBI~ 7/11~ 7/23~ <NA>             <NA>
## 2 COLE~ ANTHI~ 8/7/~ 8/10~ <NA>             <NA>
```

```
## 3 COLE~ CARAB~ 7/9/~ 7/18~ <NA>             <NA>
## 4 COLE~ CARAB~ 8/26~ 8/27~ <NA>             <NA>
## 5 COLE~ CORTI~ 8/14~ 8/16~ <NA>             <NA>
## 6 COLE~ CRYPT~ 8/4/~ 8/7/~ <NA>             <NA>
## # ... with 14 more variables: `_data/light_trap/1994.csv` <chr>,
## #   `_data/light_trap/1995.csv` <chr>, `_data/light_trap/1997.csv` <chr>,
## #   `_data/light_trap/1998.csv` <chr>, `_data/light_trap/1999.csv` <chr>,
## #   `_data/light_trap/2000.csv` <chr>, `_data/light_trap/2001.csv` <chr>,
## #   `_data/light_trap/2003.csv` <chr>, `_data/light_trap/2004.csv` <chr>,
## #   `_data/light_trap/2005.csv` <chr>, `_data/light_trap/2006.csv` <chr>,
## #   `_data/light_trap/2007.csv` <chr>, `_data/light_trap/2008.csv` <chr>,
## #   `_data/light_trap/2009.csv` <chr>
```

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.

```
# View(iris) OR glimpse(iris) OR head(iris)

iris %>%
    gather(measure, value, -Species) %>%
    head()
```

```
##   Species      measure value
## 1  setosa Sepal.Length   5.1
## 2  setosa Sepal.Length   4.9
## 3  setosa Sepal.Length   4.7
## 4  setosa Sepal.Length   4.6
## 5  setosa Sepal.Length   5.0
## 6  setosa Sepal.Length   5.4
```

```
iris %>%
    gather(measure, value, Sepal.Length:Petal.Width) %>%
    head()
```

```
##   Species      measure value
## 1  setosa Sepal.Length   5.1
## 2  setosa Sepal.Length   4.9
## 3  setosa Sepal.Length   4.7
## 4  setosa Sepal.Length   4.6
## 5  setosa Sepal.Length   5.0
## 6  setosa Sepal.Length   5.4
```

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`

```
left_join(x = read_rds("_output/02_02_wood_blocks.rds"),
          y = read_rds("_output/01_wood_site_data.rds")) %>%
    glimpse() %>%
    write_rds("_output/03_wood_blocks.rds")
```

```
## Joining, by = "site_id"
```

```
## Observations: 360
## Variables: 22
## $ source_file           <chr> "_data/wood_blocks/Australia_Calperum/CL...
## $ treatment             <chr> "C", "C", "T", "T", "C", "C", "C", "C", ...
## $ wood_id               <chr> "01", "05", "23", "30", "28", "02", "27"...
## $ plot_id               <chr> "BS-01", "BS-01", "BS-01", "BS-01", "BS-...
## $ site_id               <chr> "10", "10", "10", "10", "10", "10", "10"...
## $ date_deploy           <chr> "2017_08_21", "2017_08_21", "2017_08_21"...
## $ date_collect          <chr> "2018_08_23", NA, "2018_08_23", NA, "201...
## $ termites              <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ insects               <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ fungi                 <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ damage_fungal         <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ damage_termite        <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ init_weight           <chr> "189.65", "197.19", "172.57", "179.03", ...
## $ wet_weight_pre_drill  <chr> NA, NA, NA, NA, "184.91", NA, NA, NA, "2...
## $ wet_weight_post_drill <chr> NA, NA, NA, NA, "181.65", NA, NA, NA, "2...
## $ end_weight_post_drill <chr> NA, NA, NA, NA, "153.15", NA, NA, NA, "1...
## $ end_weight            <chr> "179.1742", NA, "160.0825", NA, NA, "202...
## $ deploy_image          <chr> "4061", "4061", "4061", "4061", "4062", ...
## $ collect_image         <chr> "168", NA, "168", NA, "169", "169", NA, ...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <chr> "-34.049289", "-34.049289", "-34.049289"...
## $ lon                   <chr> "140.82822", "140.82822", "140.82822", "...
```

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`.

```
read_rds("_output/03_wood_blocks.rds") %>%
    select(-source_file,
           -starts_with("date"),
           -ends_with("_image"),
           -starts_with("wet_weight")) %>%
    glimpse() %>%
    write_rds("_output/04_wood_blocks.rds")
```

```
## Observations: 360
## Variables: 15
## $ treatment             <chr> "C", "C", "T", "T", "C", "C", "C", "C", ...
## $ wood_id               <chr> "01", "05", "23", "30", "28", "02", "27"...
## $ plot_id               <chr> "BS-01", "BS-01", "BS-01", "BS-01", "BS-...
## $ site_id               <chr> "10", "10", "10", "10", "10", "10", "10"...
## $ termites              <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ insects               <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ fungi                 <chr> "N", NA, "N", NA, "N", "N", NA, NA, "N",...
## $ damage_fungal         <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ damage_termite        <chr> "0", NA, "0", NA, "0", "0", NA, NA, "0",...
## $ init_weight           <chr> "189.65", "197.19", "172.57", "179.03", ...
## $ end_weight_post_drill <chr> NA, NA, NA, NA, "153.15", NA, NA, NA, "1...
## $ end_weight            <chr> "179.1742", NA, "160.0825", NA, NA, "202...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <chr> "-34.049289", "-34.049289", "-34.049289"...
## $ lon                   <chr> "140.82822", "140.82822", "140.82822", "...
```

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`.

```
read_rds("_output/04_wood_blocks.rds") %>%
    distinct(treatment, wood_id, plot_id, site_id, .keep_all = TRUE) %>%
    drop_na(-end_weight) %>%
    arrange(site_id, desc(init_weight)) %>%
    glimpse() %>%
    write_rds("_output/05_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## $ treatment             <chr> "T", "C", "T", "T", "T", "C", "C", "T", ...
## $ wood_id               <chr> "18", "39", "19", "10", "34", "30", "12"...
## $ plot_id               <chr> "BS-02", "BS-16", "BS-06", "BS-20", "BS-...
## $ site_id               <chr> "10", "10", "10", "10", "10", "10", "10"...
## $ termites              <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ insects               <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ fungi                 <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ damage_fungal         <chr> "0", "0", "0", "0", "0", "0", "0", "0", ...
## $ damage_termite        <chr> "0", "0", "2", "0", "2", "0", "0", "0", ...
## $ init_weight           <chr> "209.35", "207.24", "205.73", "202.57", ...
## $ end_weight_post_drill <chr> "191.94", "197.01", "187.19", "185.03", ...
## $ end_weight            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <chr> "-34.049289", "-34.049289", "-34.049289"...
## $ lon                   <chr> "140.82822", "140.82822", "140.82822", "...
```

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.

```
iris %>%
    filter(Species == "setosa", Petal.Length > 1.5, Sepal.Length < 5) %>%
    glimpse()
```

```
## Observations: 4
## Variables: 5
## $ Sepal.Length <dbl> 4.8, 4.8, 4.7, 4.8
## $ Sepal.Width  <dbl> 3.4, 3.4, 3.2, 3.1
## $ Petal.Length <dbl> 1.6, 1.9, 1.6, 1.6
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2
## $ Species      <fct> setosa, setosa, setosa, setosa
```

05-03

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

```
# Using arrange() and then distinct() in this way only works when you want to
# keep a single row. To keep the top 'n' rows, use group_by() and then top_n().
# This is covered in the chapter about grouping and summarising.

starwars %>%
    arrange(desc(mass)) %>%
    distinct(species, .keep_all = TRUE) %>%
    glimpse()
```

```
## Observations: 38
## Variables: 13
## $ name       <chr> "Jabba Desilijic Tiure", "Grievous", "IG-88", "Dart...
## $ height     <int> 175, 216, 200, 202, 234, 190, 198, 191, 229, 196, 1...
## $ mass       <dbl> 1358.0, 159.0, 140.0, 136.0, 136.0, 113.0, 102.0, 9...
## $ hair_color <chr> NA, "none", "none", "none", "brown", "none", "none"...
## $ skin_color <chr> "green-tan, brown", "brown, white", "metal", "white...
## $ eye_color  <chr> "orange", "green, yellow", "red", "yellow", "blue",...
## $ birth_year <dbl> 600.0, NA, 15.0, 41.9, NA, 53.0, NA, NA, NA, NA, 41...
## $ gender     <chr> "hermaphrodite", "male", "none", "male", "male", "m...
## $ homeworld  <chr> "Nal Hutta", "Kalee", NA, "Tatooine", "Kashyyyk", "...
## $ species    <chr> "Hutt", "Kaleesh", "Droid", "Human", "Wookiee", "Tr...
## $ films      <list> [<"The Phantom Menace", "Return of the Jedi", "A N...
## $ vehicles   <list> [<>, "Tsmeu-6 personal wheel bike", <>, <>, <>, <>...
## $ starships  <list> [<>, "Belbullab-22 starfighter", <>, "TIE Advanced...
```

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`.

```
read_rds("_output/05_wood_blocks.rds") %>%
    mutate_at(vars(contains("weight")),
              ~ if_else(. == "ND", NA_character_, .)) %>%
    glimpse() %>%
    write_rds("_output/06_01_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## $ treatment             <chr> "T", "C", "T", "T", "T", "C", "C", "T", ...
## $ wood_id               <chr> "18", "39", "19", "10", "34", "30", "12"...
## $ plot_id               <chr> "BS-02", "BS-16", "BS-06", "BS-20", "BS-...
## $ site_id               <chr> "10", "10", "10", "10", "10", "10", "10"...
## $ termites              <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ insects               <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ fungi                 <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ damage_fungal         <chr> "0", "0", "0", "0", "0", "0", "0", "0", ...
## $ damage_termite        <chr> "0", "0", "2", "0", "2", "0", "0", "0", ...
## $ init_weight           <chr> "209.35", "207.24", "205.73", "202.57", ...
## $ end_weight_post_drill <chr> "191.94", "197.01", "187.19", "185.03", ...
## $ end_weight            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <chr> "-34.049289", "-34.049289", "-34.049289"...
## $ lon                   <chr> "140.82822", "140.82822", "140.82822", "...
```

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`.

```
read_rds("_output/06_01_wood_blocks.rds") %>%
    mutate(wood_id = as.integer(wood_id),
           site_id = as.integer(site_id)) %>%
    mutate_at(vars(contains("weight"), lat, lon), as.numeric) %>%
    mutate_at(vars(contains("damage")), ~ ordered(., levels = c(0, 1, 2, 3, 4))) %>%
    glimpse() %>%
    write_rds("_output/06_02_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## $ treatment             <chr> "T", "C", "T", "T", "T", "C", "C", "T", ...
## $ wood_id               <int> 18, 39, 19, 10, 34, 30, 12, 59, 41, 50, ...
## $ plot_id               <chr> "BS-02", "BS-16", "BS-06", "BS-20", "BS-...
## $ site_id               <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, ...
## $ termites              <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ insects               <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ fungi                 <chr> "N", "N", "N", "N", "N", "N", "N", "N", ...
## $ damage_fungal         <ord> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ damage_termite        <ord> 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ init_weight           <dbl> 209.35, 207.24, 205.73, 202.57, 200.26, ...
## $ end_weight_post_drill <dbl> 191.94, 197.01, 187.19, 185.03, 181.82, ...
## $ end_weight            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <dbl> -34.04929, -34.04929, -34.04929, -34.049...
## $ lon                   <dbl> 140.8282, 140.8282, 140.8282, 140.8282, ...
```

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`

```
read_rds("_output/06_02_wood_blocks.rds") %>%
    # You could use either if_else() or case_when() here, or even
    # dplyr::recode(), which I chose not to teach.
    mutate_at(vars(termites:fungi), ~ case_when(. == "Y" ~ TRUE,
                                                . == "N" ~ FALSE,
                                                   TRUE ~ NA)) %>%
    # I use mutate_at() for convenience, because then I can refer to the column as '.'
    # instead of repeatedly writing out 'treatment' all the time.
    mutate_at(vars(treatment), ~ case_when(. == "T" ~ "Treatment",
                                           . == "C" ~ "Control",
                                              TRUE ~ NA_character_)) %>%
    glimpse() %>%
    write_rds("_output/06_03_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## $ treatment             <chr> "Treatment", "Control", "Treatment", "Tr...
## $ wood_id               <int> 18, 39, 19, 10, 34, 30, 12, 59, 41, 50, ...
## $ plot_id               <chr> "BS-02", "BS-16", "BS-06", "BS-20", "BS-...
## $ site_id               <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, ...
## $ termites              <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ insects               <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ fungi                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ damage_fungal         <ord> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ damage_termite        <ord> 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ init_weight           <dbl> 209.35, 207.24, 205.73, 202.57, 200.26, ...
## $ end_weight_post_drill <dbl> 191.94, 197.01, 187.19, 185.03, 181.82, ...
## $ end_weight            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ site                  <chr> "Calperum", "Calperum", "Calperum", "Cal...
## $ lat                   <dbl> -34.04929, -34.04929, -34.04929, -34.049...
## $ lon                   <dbl> 140.8282, 140.8282, 140.8282, 140.8282, ...
```

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`.

```
read_rds("_output/06_03_wood_blocks.rds") %>%
    mutate(end_weight = if_else(is.na(end_weight),
                                end_weight_post_drill,
                                end_weight),
           weight_lost = init_weight - end_weight) %>%
    select(-end_weight_post_drill) %>%
    glimpse() %>%
    write_rds("_output/06_04_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## $ treatment      <chr> "Treatment", "Control", "Treatment", "Treatment...
## $ wood_id        <int> 18, 39, 19, 10, 34, 30, 12, 59, 41, 50, 6, 25, ...
## $ plot_id        <chr> "BS-02", "BS-16", "BS-06", "BS-20", "BS-12", "B...
## $ site_id        <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,...
## $ termites       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ insects        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ fungi          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE...
## $ damage_fungal  <ord> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ damage_termite <ord> 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ init_weight    <dbl> 209.35, 207.24, 205.73, 202.57, 200.26, 197.60,...
## $ end_weight     <dbl> 191.94, 197.01, 187.19, 185.03, 181.82, 181.09,...
## $ site           <chr> "Calperum", "Calperum", "Calperum", "Calperum",...
## $ lat            <dbl> -34.04929, -34.04929, -34.04929, -34.04929, -34...
## $ lon            <dbl> 140.8282, 140.8282, 140.8282, 140.8282, 140.828...
```

```
## $ weight_lost    <dbl> 17.41, 10.23, 18.54, 17.54, 18.44, 16.51, 11.48...
```

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`.

```
read_rds("_output/06_04_wood_blocks.rds") %>%
    group_by(site, wood_id, treatment) %>%
    summarise_at(vars(contains("weight")),
                 list(~mean, ~median, ~min, ~max)) %>%
    glimpse() %>%
    write_rds("_output/07_01_wood_blocks.rds")
```

```
## Observations: 119
## Variables: 15
## Groups: site, wood_id [87]
## $ site               <chr> "Calperum", "Calperum", "Calperum", "Calper...
## $ wood_id            <int> 6, 10, 12, 15, 18, 19, 19, 20, 24, 25, 28, ...
## $ treatment          <chr> "Treatment", "Treatment", "Control", "Treat...
## $ init_weight_mean   <dbl> 191.00, 202.57, 195.15, 179.05, 209.35, 187...
## $ end_weight_mean    <dbl> 172.63, 185.03, 183.67, 164.06, 191.94, 178...
## $ weight_lost_mean   <dbl> 18.37, 17.54, 11.48, 14.99, 17.41, 8.26, 18...
## $ init_weight_median <dbl> 191.00, 202.57, 195.15, 179.05, 209.35, 187...
## $ end_weight_median  <dbl> 172.63, 185.03, 183.67, 164.06, 191.94, 178...
## $ weight_lost_median <dbl> 18.37, 17.54, 11.48, 14.99, 17.41, 8.26, 18...
## $ init_weight_min    <dbl> 191.00, 202.57, 195.15, 179.05, 209.35, 187...
## $ end_weight_min     <dbl> 172.63, 185.03, 183.67, 164.06, 191.94, 178...
## $ weight_lost_min    <dbl> 18.37, 17.54, 11.48, 14.99, 17.41, 8.26, 18...
## $ init_weight_max    <dbl> 191.00, 202.57, 195.15, 179.05, 209.35, 187...
## $ end_weight_max     <dbl> 172.63, 185.03, 183.67, 164.06, 191.94, 178...
## $ weight_lost_max    <dbl> 18.37, 17.54, 11.48, 14.99, 17.41, 8.26, 18...
```

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.

```
# Not all levels are represented.
read_rds("_output/06_04_wood_blocks.rds") %>%
    count(termites, insects, fungi)
```

```
## # A tibble: 5 x 4
##   termites insects fungi     n
##   <lgl>    <lgl>   <lgl> <int>
## 1 FALSE    FALSE   FALSE   104
```

```
## 2 FALSE    FALSE   TRUE     10
## 3 FALSE    TRUE    FALSE     3
## 4 FALSE    TRUE    TRUE      1
## 5 TRUE     FALSE   FALSE     1
```

```
# Force all levels to be represented by mutating these columns to Factor type
# and setting .drop = FALSE in count(). The student would need to a) understand
# Factors, and b) look in the docs to know that .drop exists.
read_rds("_output/06_04_wood_blocks.rds") %>%
    mutate_at(vars(termites:fungi), as.factor) %>%
    count(termites, insects, fungi, .drop = FALSE)
```

```
## # A tibble: 8 x 4
##   termites insects fungi     n
##   <fct>    <fct>   <fct> <int>
## 1 FALSE    FALSE   FALSE   104
## 2 FALSE    FALSE   TRUE     10
## 3 FALSE    TRUE    FALSE     3
## 4 FALSE    TRUE    TRUE      1
## 5 TRUE     FALSE   FALSE     1
## 6 TRUE     FALSE   TRUE      0
## 7 TRUE     TRUE    FALSE     0
## 8 TRUE     TRUE    TRUE      0
```

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
```

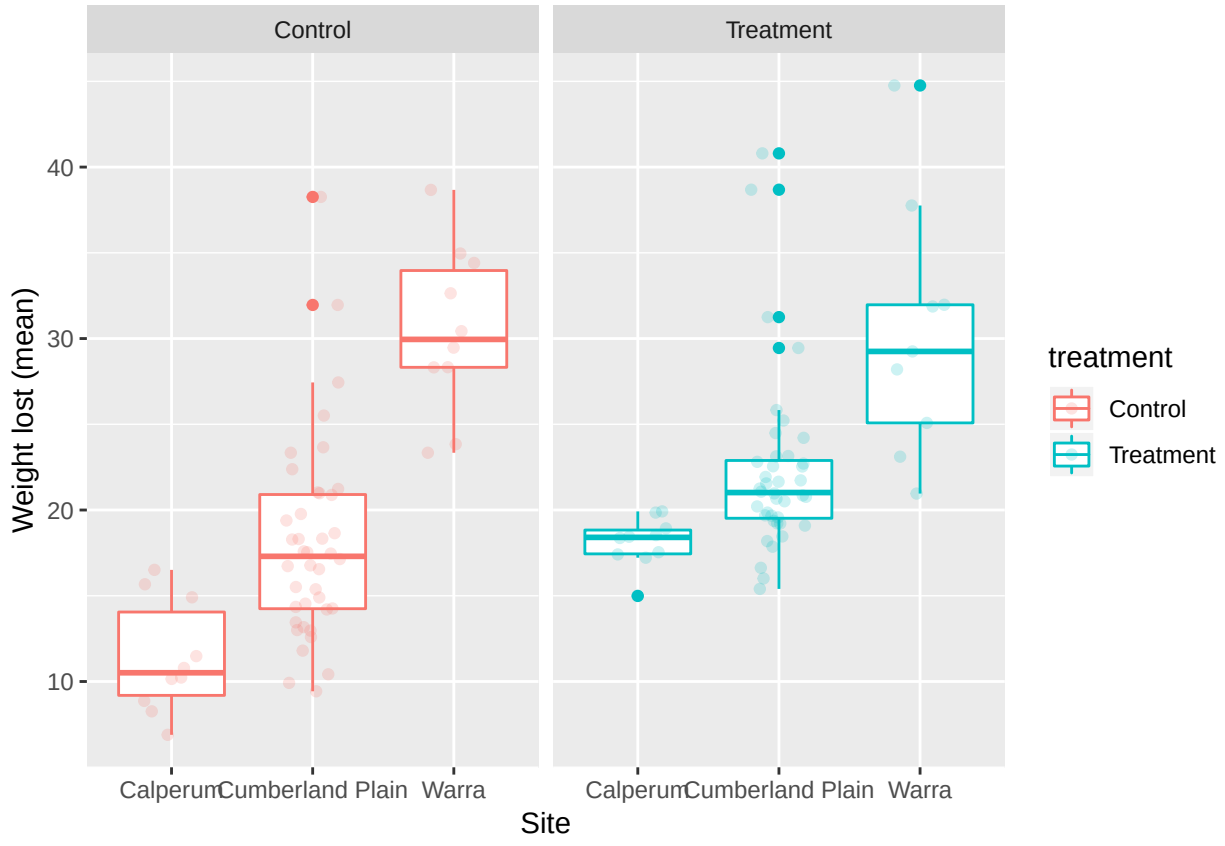
08. Graphing with ggplot2

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)"`.

```
read_rds("_output/07_01_wood_blocks.rds") %>%
    ggplot(mapping = aes(x = site, y = weight_lost_mean, colour = treatment)) +
    geom_boxplot() +
    geom_jitter(width = 0.2, height = 0, alpha = 0.2) +
    facet_grid(. ~ treatment) +
    labs(x = "Site", y = "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)"`.

```
read_rds("_output/07_01_wood_blocks.rds") %>%
    ggplot(mapping = aes(x = init_weight_mean, y = weight_lost_mean,
                         colour = treatment, shape = site)) +
    geom_point() +
    geom_smooth(method = "lm") +
    facet_wrap(site ~ .) +
    labs(x = "Starting weight (mean)", y = "Weight lost (mean)")
```