Reading Excel Sheets using readxl

Augustus Pendleton

Goal for Today: Reading in Data from Excel Sheets

Even though I'm a die-hard R fan, I also LOVE Excel. I like to collect all my data in Excel sheets, and most collaborators will ask for data as Excel files. Excel is also handy for auto-filling repreated values or sequences. However, base R cannot naturally read in data from Excel. As such, we have two options:

1. Export our Excel sheets as .csv or .tsv files
2. Use the `readxl` package to read directly from Excel sheets.

For extremely large datasets and exports, it might make sense to save data as machine-readable delimited files (like csv, tsv, or txt), because they are much more compact and can be easily read across most coding languages.

However, for most lab work, I prefer to keep my data as excel sheets only. This way, when I add new data, I don't need to re-export the sheet every time. There have been many times where I have forgotten to export my files, and don't realize that I was reading in out-dated data into R. Finally, I can keep multiple sheets in an Excel file, potentially helping me stay more organized.

Our Data

In the data/ directory, I have an Excel file with three sheets. In this experiment, I measured grew wild-type (WT) and a mutant (Mut) for 10 hours in two types of media (LB and Minimal media). They each had a fluorescent reporter. At three time points, I measured their fluorescence and optical density (OD) in a well-plate.

What sheets are in our Excel file?

First, we need to load the readxl package

```
# If you haven't installed it, we need to run this line first
# install.packages("readxl")
library(readxl)
```

Now, we know that our file has multiple sheets within it. We can see which sheets are in the file using the `excel_sheets` function

```
excel_sheets("data/Fluor_assay.xlsx")
```

```
[1] "Fluor_Results" "OD_Results"    "Well Map"
```

This tells us that we have three sheets, named Fluor_Results, OD_Results, and Well Map.

Reading in data

We will read in data from our Excel sheets using the `read_excel` function. By default, it will read in the first sheet in the workbook.

```
read_excel("data/Fluor_Assay.xlsx")
```

```
# A tibble: 3 x 17
   Time    A1    A2    A3    A4    A5    A6    A7    A8    B1    B2    B3    B4
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     0   0.7   0.3     0   0.9   0.5   0.3   0.9   0.6   0.8   0.9   0.6     0
2     5  90    88      85  94    79    73    76    71    19    19    20      15
3    10 201   237     233 242   167   162   164   177    11    19    20      19
# i 4 more variables: B5 <dbl>, B6 <dbl>, B7 <dbl>, B8 <dbl>
```

If we want to read in a specific sheet, we can specify it by name

```
read_excel("data/Fluor_Assay.xlsx", sheet = "OD_Results")
```

```
# A tibble: 3 x 17
   Time    A1    A2    A3    A4    A5    A6    A7    A8    B1    B2    B3    B4
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     0  0.05  0.05  0.05  0.05  0.05  0.04  0.04  0.04  0.04  0.04  0.05  0.04
2     5  4     4.32  4.88  4.34  1.56  1.99  1.52  1.18  4.58  4.96  4.04  4.76
3    10 10.5   9.1   9.1   9.2   3.6   3.9   3     3.7   9.5  10.1  10.9   9.7
# i 4 more variables: B5 <dbl>, B6 <dbl>, B7 <dbl>, B8 <dbl>
```

Or number

```
read_excel("data/Fluor_Assay.xlsx", sheet = 2)
```

```
# A tibble: 3 x 17
   Time    A1    A2    A3    A4    A5    A6    A7    A8    B1    B2    B3    B4
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     0  0.05  0.05  0.05  0.05  0.05  0.04  0.04  0.04  0.04  0.04  0.05  0.04
2     5  4     4.32  4.88  4.34  1.56  1.99  1.52  1.18  4.58  4.96  4.04  4.76
3    10 10.5   9.1   9.1   9.2   3.6   3.9   3     3.7   9.5  10.1  10.9   9.7
# i 4 more variables: B5 <dbl>, B6 <dbl>, B7 <dbl>, B8 <dbl>
```

Why might reading in by name be a better idea than by number?

Reading and Saving as an Object

If you look at our "Environment" pane on the right, you'll notice that it's empty. This is because we haven't stored the data we're reading in as an object yet. Let's do that for our three sheets.

```r
fluor_df <- read_excel("data/Fluor_Assay.xlsx", sheet = "Fluor_Results")

od_df <- read_excel("data/Fluor_Assay.xlsx", sheet = "OD_Results")

well_map <- read_excel("data/Fluor_Assay.xlsx", sheet = "Well Map")
```

There's a little warning that popped up, saying R assigned new names to the columns of our well map. Maybe we should take a look at that.

```r
well_map
```

```
# A tibble: 17 x 7
   `Well Map Layout:` ...2  ...3    ...4  ...5  ...6  ...7
   <chr>              <chr> <chr>   <lgl> <lgl> <lgl> <chr>
 1 Well               Strain Media   NA    NA    NA    <NA>
 2 A1                 WT     LB      NA    NA    NA    <NA>
 3 A2                 WT     LB      NA    NA    NA    Here are some extraneous~
 4 A3                 WT     LB      NA    NA    NA    <NA>
 5 A4                 WT     LB      NA    NA    NA    <NA>
 6 A5                 WT     Minimal NA    NA    NA    <NA>
 7 A6                 WT     Minimal NA    NA    NA    <NA>
 8 A7                 WT     Minimal NA    NA    NA    <NA>
 9 A8                 WT     Minimal NA    NA    NA    <NA>
10 B1                 Mut    LB      NA    NA    NA    <NA>
11 B2                 Mut    LB      NA    NA    NA    <NA>
12 B3                 Mut    LB      NA    NA    NA    <NA>
13 B4                 Mut    LB      NA    NA    NA    <NA>
14 B5                 Mut    Minimal NA    NA    NA    <NA>
15 B6                 Mut    Minimal NA    NA    NA    <NA>
```

```
16 B7                 Mut    Minimal NA    NA    NA    <NA>
17 B8                 Mut    Minimal NA    NA    NA    <NA>
```

Hmmm. It looks like we have a column called "Well Map Layout:" and then funky named columns called …2 and …3, etc. In fact, it looks like our actual table headers are in the first row right now. This is super common, that people will put a descriptive header in the sheet, with the actual data further down. It also looks like I left myself a little note on the right side, which R thinks is a column of data. We can fix these these problems in two ways

Telling read_excel to skip the first row

```
read_excel("data/Fluor_Assay.xlsx", sheet = "Well Map", skip = 1)
```

```
# A tibble: 16 x 7
   Well  Strain Media   ...4  ...5  ...6  ...7
   <chr> <chr>  <chr>   <lgl> <lgl> <lgl> <chr>
 1 A1    WT     LB      NA    NA    NA    <NA>
 2 A2    WT     LB      NA    NA    NA    Here are some extraneous experimental~
 3 A3    WT     LB      NA    NA    NA    <NA>
 4 A4    WT     LB      NA    NA    NA    <NA>
 5 A5    WT     Minimal NA    NA    NA    <NA>
 6 A6    WT     Minimal NA    NA    NA    <NA>
 7 A7    WT     Minimal NA    NA    NA    <NA>
 8 A8    WT     Minimal NA    NA    NA    <NA>
 9 B1    Mut    LB      NA    NA    NA    <NA>
10 B2    Mut    LB      NA    NA    NA    <NA>
11 B3    Mut    LB      NA    NA    NA    <NA>
12 B4    Mut    LB      NA    NA    NA    <NA>
13 B5    Mut    Minimal NA    NA    NA    <NA>
14 B6    Mut    Minimal NA    NA    NA    <NA>
15 B7    Mut    Minimal NA    NA    NA    <NA>
```

```
16 B8    Mut    Minimal NA    NA    NA    <NA>
```

Now our column headers are right. We can also tell it which range to actually read in

```
read_excel("data/Fluor_Assay.xlsx", sheet = "Well Map", skip = 1, range = "A1:C18")
```

```
# A tibble: 17 x 3
   `Well Map Layout:` ...2   ...3
   <chr>              <chr>  <chr>
 1 Well               Strain Media
 2 A1                 WT     LB
 3 A2                 WT     LB
 4 A3                 WT     LB
 5 A4                 WT     LB
 6 A5                 WT     Minimal
 7 A6                 WT     Minimal
 8 A7                 WT     Minimal
 9 A8                 WT     Minimal
10 B1                 Mut    LB
11 B2                 Mut    LB
12 B3                 Mut    LB
13 B4                 Mut    LB
14 B5                 Mut    Minimal
15 B6                 Mut    Minimal
16 B7                 Mut    Minimal
17 B8                 Mut    Minimal
```

But be-warned: this will over-ride our skip argument. So we could instead do

```
read_excel("data/Fluor_Assay.xlsx", sheet = "Well Map", range = "A2:C18")
```

```
# A tibble: 16 x 3
   Well  Strain Media
   <chr> <chr>  <chr>
 1 A1    WT     LB
 2 A2    WT     LB
 3 A3    WT     LB
 4 A4    WT     LB
 5 A5    WT     Minimal
 6 A6    WT     Minimal
 7 A7    WT     Minimal
 8 A8    WT     Minimal
 9 B1    Mut    LB
10 B2    Mut    LB
11 B3    Mut    LB
12 B4    Mut    LB
13 B5    Mut    Minimal
14 B6    Mut    Minimal
15 B7    Mut    Minimal
16 B8    Mut    Minimal
```

In fact, if you're comfortable with Excel ranges, you can also use them to specify the sheet

```
read_excel("data/Fluor_Assay.xlsx", range = "Well Map!A2:C18")
```

```
# A tibble: 16 x 3
   Well  Strain Media
   <chr> <chr>  <chr>
 1 A1    WT     LB
 2 A2    WT     LB
 3 A3    WT     LB
```

```
 4 A4    WT     LB
 5 A5    WT     Minimal
 6 A6    WT     Minimal
 7 A7    WT     Minimal
 8 A8    WT     Minimal
 9 B1    Mut    LB
10 B2    Mut    LB
11 B3    Mut    LB
12 B4    Mut    LB
13 B5    Mut    Minimal
14 B6    Mut    Minimal
15 B7    Mut    Minimal
16 B8    Mut    Minimal
```

That looks good! Let's save it

```
well_map <- read_excel("data/Fluor_Assay.xlsx", sheet = "Well Map", range = "A2:C18")
```

Other Arguments

There are lots of other arguments we can pass to read_excel, that I won't really get into today. We can specify `col_names` and `col_types` to manually dictate what the header names and data types (like numerical or character) we want each column to be. `read_excel` automatically trims whitespace (like spaces and tabs) from the beginning and ends of data cells; you can turn this off with the `trim_ws` argument. If your file is absolutely huge, you can limit how many rows to read in using the `n_max` argument.

Cleaning and Analyzing Our Data

Now that we have our data in R, we can use our coding skills to rapidly (and reproducibly) clean, combine, and plot our data. Below, I'm going to convert my data frames to long format,

use joins to combine them with the strain and media information, summarized them, and plot them. I will use functions from the `tidyverse` packages.

```
library(tidyverse) # Load tidyverse packages

# Pivot our fluorescence data to long format
long_fluor <- fluor_df %>%
  pivot_longer(cols = !Time, names_to = "Well", values_to = "Fluorescence") # All columns

long_od <- od_df %>%
  pivot_longer(cols = !Time, names_to = "Well", values_to = "OD") # All columns that are N

joined <- well_map %>%
  inner_join(long_fluor, by = "Well") %>% # Join our fluorescence data based on Well
  inner_join(long_od, by = c("Well", "Time")) # Join our od data based on Well and Time



joined_with_norm_fluor <- joined %>%
  mutate(Normalized_Fluorescence = Fluorescence / OD) # Create a new column of Fluorescenc

summarized <- joined_with_norm_fluor %>%
  group_by(Time, Strain, Media) %>% # Group by Time, Strain, and Media
  summarize(Avg_Norm_Fluor = mean(Normalized_Fluorescence), # Calculate average normalized
            Sd_Norm_Fluor = sd(Normalized_Fluorescence)) # Calculate standard deviation
```

Writing Data to Excel

Perhaps your lab mate, PI, or collaborator just want to see the summarized data, or the long-formatted data for their own analyses. We can write data to excel files using the package `writexl`:

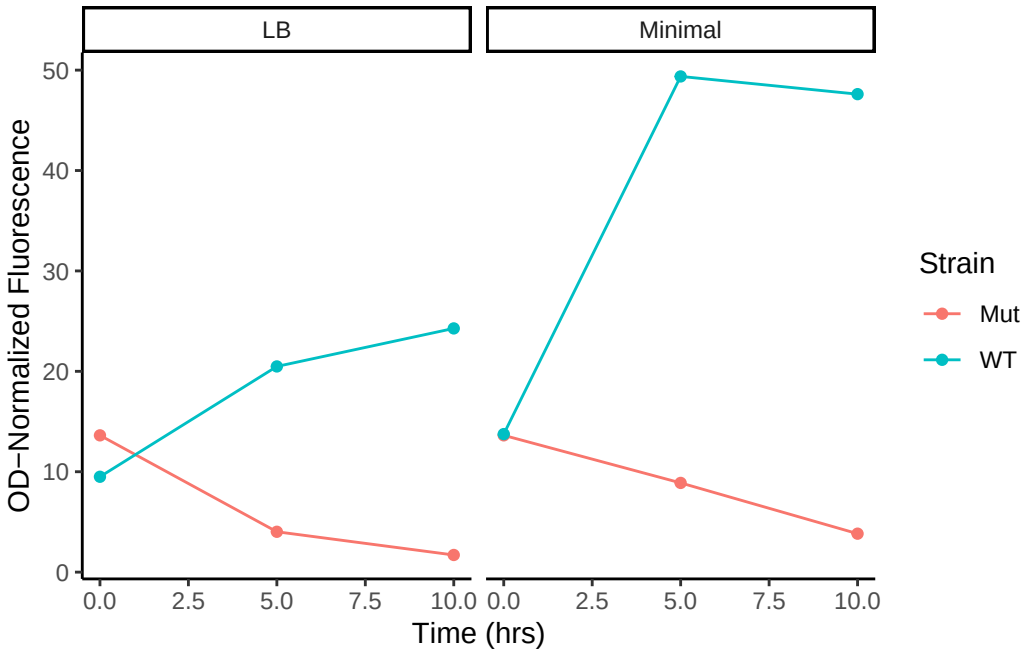
```r
# install.packages("writexl)

library(writexl)

write_xlsx(joined_with_norm_fluor, path = "data/Long_Formatted_Data.xlsx")

write_xlsx(summarized, path = "data/Summarized_Data.xlsx")
```

Apparently you can write multiple data frames to different sheets of the same Excel workbook using the `xlsx` package. However, this requires a separate Java installation and crashes my R session when I attempt to load it (these errors are documented here). Could be something useful in the future though!

Plotting Data

We did all that work to make some beautiful data! Let's plot it to finish the lesson out :)

```r
summarized %>%
  ggplot(aes(x = Time, y = Avg_Norm_Fluor, color = Strain)) +
  geom_line() +
  geom_point() +
  facet_wrap(~Media) +
  labs(x = "Time (hrs)", y = "OD-Normalized Fluorescence") +
  theme_classic()
```


LB
Minimal
OD−Normalized Fluorescence
50
40
30
20
10
0
0.0
2.5
5.0
7.5
10.0
Time (hrs)
Strain
Mut
WT