Incorporating experimental designs

Mike Blazanin

Contents

Where are we so far?

1. Introduction: `vignette("gc01_gcplyr")`
2. Importing and reshaping data: `vignette("gc02_import_reshape")`
3. **Incorporating experimental designs:** `vignette("gc03_incorporate_designs")`
4. Pre-processing and plotting your data: `vignette("gc04_preprocess_plot")`
5. Processing your data: `vignette("gc05_process")`
6. Analyzing your data: `vignette("gc06_analyze")`
7. Dealing with noise: `vignette("gc07_noise")`
8. Best practices and other tips: `vignette("gc08_conclusion")`
9. Working with multiple plates: `vignette("gc09_multiple_plates")`
10. Using make_design to generate experimental designs: `vignette("gc10_using_make_design")`

So far, we've imported and transformed our measures data into `R`. Now we're going to address how to incorporate our experimental design.

If you haven't already, load the necessary packages.

```
library(gcplyr)
```

Including design elements

We often want to combine our data with information about the experimental design. `gcplyr` enables incorporation of design elements in two ways:

1. Designs can be imported from files
2. Designs can be generated in `R` using `make_design`

If you're interested in generating your designs in `R`, see `vignette("gc10_using_make_design")`

When reading design elements from files, `gcplyr` can read block-shaped or tidy-shaped design files:

- If design files are block-shaped, they can be read with `import_blockdesigns`
- If design files are tidy-shaped, they can simply be read with `read_tidys`

Importing block-shaped design files

To import block-shaped design files, use `import_blockdesigns`, which will return a tidy-shaped designs data frame (or list of data frames).

`import_blockdesigns` only requires a list of filenames (or relative file paths) and will return a data.frame (or list of data frames) in a **tidy format** that you can save in R.

A basic example

Let's look at an example. First, we need to create an example file for the sake of this tutorial (normally you'd create this file in Excel)

```
make_example(vignette = 3, example = 1, dir = ".")
#> Files have been written
#> [1] "./mydesign.csv"
```

Now let's take a look at what the file looks like:

```
print_df(read.csv("mydesign.csv", header = FALSE, colClasses = "character"))
#>     1   2   3   4   5   6   7   8   9  10  11  12
#> A Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> B Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> C Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> D Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> E Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> F Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> G Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> H Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
```

Here we can see that our design has Treatment 1 on the left-hand side of the plate (wells in columns 1 through 6), and Treatment 2 on the right-hand side of the plate (wells in columns 7 through 12). Let's import this design using `import_blockdesigns`, saving it with the column name `Treatment_numbers`.

```
my_design <- import_blockdesigns(files = "mydesign.csv",
                                 block_names = "Treatment_numbers")
head(my_design, 20)
#>    Well Treatment_numbers
#> 1    A1               Tr1
#> 2    A2               Tr1
#> 3    A3               Tr1
#> 4    A4               Tr1
#> 5    A5               Tr1
#> 6    A6               Tr1
#> 7    A7               Tr2
#> 8    A8               Tr2
#> 9    A9               Tr2
#> 10  A10               Tr2
#> 11  A11               Tr2
#> 12  A12               Tr2
#> 13   B1               Tr1
#> 14   B2               Tr1
#> 15   B3               Tr1
#> 16   B4               Tr1
#> 17   B5               Tr1
#> 18   B6               Tr1
#> 19   B7               Tr2
#> 20   B8               Tr2
```

Importing multiple block-shaped design elements

What do you do if you have multiple designs? For instance, what if you have several strains each in several treatments? In that case, you have two options:

1. Save each design component as a separate file, or in separate blocks within a file
2. Save the design components pasted together in a single file

Regardless of which option you use, you can then import them all in one go with `import_blockdesigns`.

Importing multiple separated block-shaped designs

First, let's create both our example designs files. Again, just imagine that you've created these files in Excel.

```
make_example(vignette = 3, example = 1, dir = ".")
#> Files have been written
#> [1] "./mydesign.csv"
make_example(vignette = 3, example = 2, dir = ".")
#> Files have been written
#> [1] "./mydesign2.csv"
```

Now let's take a look at what these files looks like:

```
print_df(read.csv("mydesign.csv", header = FALSE, colClasses = "character"))
#>     1   2   3   4   5   6   7   8   9  10  11  12
#> A Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
```

```
#> B Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> C Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> D Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> E Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> F Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> G Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
#> H Tr1 Tr1 Tr1 Tr1 Tr1 Tr1 Tr2 Tr2 Tr2 Tr2 Tr2 Tr2
print_df(read.csv("mydesign2.csv", header = FALSE, colClasses = "character"))
#>      1    2    3    4    5    6    7    8    9   10   11   12
#> A StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA
#> B StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA StrA
#> C StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB
#> D StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB StrB
#> E StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC
#> F StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC StrC
#> G StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD
#> H StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD StrD
```

As before, we have Treatment 1 on the left-hand side, and Treatment 2 on the right-hand side. In addition, we now also have Strain A in the first two rows, Strain B in the next two rows, and so on.

Let's now import both designs using `import_blockdesigns`, saving them to columns named `Treatment_numbers` and `Strain_letters`.

```
my_design <-
  import_blockdesigns(files = c("mydesign.csv", "mydesign2.csv"),
                      block_names = c("Treatment_numbers", "Strain_letters"))
head(my_design, 20)
#>    Well Treatment_numbers Strain_letters
#> 1    A1               Tr1           StrA
#> 2    A2               Tr1           StrA
#> 3    A3               Tr1           StrA
#> 4    A4               Tr1           StrA
#> 5    A5               Tr1           StrA
#> 6    A6               Tr1           StrA
#> 7    A7               Tr2           StrA
#> 8    A8               Tr2           StrA
#> 9    A9               Tr2           StrA
#> 10  A10               Tr2           StrA
#> 11  A11               Tr2           StrA
#> 12  A12               Tr2           StrA
#> 13   B1               Tr1           StrA
#> 14   B2               Tr1           StrA
#> 15   B3               Tr1           StrA
#> 16   B4               Tr1           StrA
#> 17   B5               Tr1           StrA
#> 18   B6               Tr1           StrA
#> 19   B7               Tr2           StrA
#> 20   B8               Tr2           StrA
```

Importing multiple pasted block-shaped designs

Alternative to saving your designs separated, often it may be easiest to save all the design information into a single block, separating the distinct components of the design with some character.

To demonstrate this, first let's create our example designs file. Again, just imagine that you've created this file in Excel.

```
make_example(vignette = 3, example = 3, dir = ".")
#> Files have been written
#> [1] "./mydesign_pasted.csv"
```

Now let's take a look at what the file looks like:

```
print_df(read.csv("mydesign_pasted.csv", header = FALSE, colClasses = "character"))
#>          1        2        3        4        5        6        7        8        9       10       11
#> A Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr2_StrA Tr2_StrA Tr2_StrA Tr2_StrA Tr2_StrA
#> B Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr1_StrA Tr2_StrA Tr2_StrA Tr2_StrA Tr2_StrA Tr2_StrA
#> C Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr2_StrB Tr2_StrB Tr2_StrB Tr2_StrB Tr2_StrB
#> D Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr1_StrB Tr2_StrB Tr2_StrB Tr2_StrB Tr2_StrB Tr2_StrB
#> E Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr2_StrC Tr2_StrC Tr2_StrC Tr2_StrC Tr2_StrC
#> F Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr1_StrC Tr2_StrC Tr2_StrC Tr2_StrC Tr2_StrC Tr2_StrC
#> G Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr2_StrD Tr2_StrD Tr2_StrD Tr2_StrD Tr2_StrD
#> H Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr1_StrD Tr2_StrD Tr2_StrD Tr2_StrD Tr2_StrD Tr2_StrD
```

As before, we have Treatment 1 on the left-hand side, and Treatment 2 on the right-hand side, with Strain A in the first two rows, Strain B in the next two rows, and so on. However, this information is now pasted together, with "_" as the separating string (you can use any string as a separator).

To import this design with `import_blockdesigns`, we simply need to specify the `sep` string, as well as the output column names. Since the designs have been pasted together, the column names will result from splitting the designs apart. The easiest way to specify these split column names is to use the `into` argument passed to `separate_tidy`. If `into` is not specified, `import_blockdesigns` will attempt to split the `block_names` (either specified or inferred) with `sep` to generate the output column names.

```
my_design <-
  import_blockdesigns(files = "mydesign_pasted.csv",
                      into = c("Treatment_numbers", "Strain_letters"),
                      sep = "_")
head(my_design, 20)
#>    Well Treatment_numbers Strain_letters
#> 1    A1               Tr1           StrA
#> 2    A2               Tr1           StrA
#> 3    A3               Tr1           StrA
#> 4    A4               Tr1           StrA
#> 5    A5               Tr1           StrA
#> 6    A6               Tr1           StrA
#> 7    A7               Tr2           StrA
#> 8    A8               Tr2           StrA
#> 9    A9               Tr2           StrA
#> 10  A10               Tr2           StrA
#> 11  A11               Tr2           StrA
#> 12  A12               Tr2           StrA
#> 13   B1               Tr1           StrA
#> 14   B2               Tr1           StrA
#> 15   B3               Tr1           StrA
#> 16   B4               Tr1           StrA
#> 17   B5               Tr1           StrA
#> 18   B6               Tr1           StrA
```

```
#> 19   B7              Tr2          StrA
#> 20   B8              Tr2          StrA
```

Importing tidy-shaped design files

You can import tidy-shaped designs with `read_tidys`.

`read_tidys` only requires a filename (or vector of filenames, or relative file paths) and will return a `data.frame` (or list of data.frames) that you can save in R.

Once these design elements have been read into the `R` environment, they are ready to merge.

Merging growth curve data with designs

Once we have both our design and data in `R` and tidy-shaped, we can merge them using `merge_dfs`.

To demonstrate this, we'll use the data in the `example_widedata_noiseless` dataset that is included with `gcplyr`, and which was the source for our previous examples with `import_blockmeasures` and `read_wides`.

In the `example_widedata_noiseless` dataset, we have 48 different bacterial strains. The left side of the plate has all 48 strains in a single well each, and the right side of the plate also has all 48 strains in a single well each:

Row names	Column 1	...	Column 6	Column 7	...	Column 12
Row A	Strain #1	...	Strain #6	Strain #1	...	Strain #6
Row B	Strain #7	...	Strain #12	Strain #7	...	Strain #12
...	...	...	...	...	...	...
Row G	Strain #37	...	Strain #42	Strain #37	...	Strain #42
Row H	Strain #43	...	Strain #48	Strain #43	...	Strain #48

Then, on the right hand side of the plate a phage was also inoculated (while the left hand side remained bacteria-only):

Row names	Column 1	...	Column 6	Column 7	...	Column 12
Row A	No Phage	...	No Phage	Phage Added	...	Phage Added
Row B	No Phage	...	No Phage	Phage Added	...	Phage Added
...	...	...	...	...	...	...
Row G	No Phage	...	No Phage	Phage Added	...	Phage Added
Row H	No Phage	...	No Phage	Phage Added	...	Phage Added

Let's transform the `example_widedata_noiseless` to tidy-shaped.

```
example_tidydata <- trans_wide_to_tidy(example_widedata_noiseless,
                                       id_cols = "Time")
```

`gcplyr` also includes the design for this data for easy use:

```
example_design <- example_design_tidy
head(example_design_tidy)
#>   Well Bacteria_strain    Phage
#> 1   A1        Strain 1 No Phage
#> 2   A2        Strain 2 No Phage
#> 3   A3        Strain 3 No Phage
#> 4   A4        Strain 4 No Phage
#> 5   A5        Strain 5 No Phage
#> 6   A6        Strain 6 No Phage
```

Now that we have our data and designs tidy-shaped, we merge the two using `merge_dfs`, saving the result to `ex_dat_mrg`, short for example_data_merged. `merge_dfs` merges using columns with the same name between the two data.frames.

```
ex_dat_mrg <- merge_dfs(example_tidydata, example_design)
#> Joining with `by = join_by(Well)`

head(ex_dat_mrg)
#>   Time Well Measurements Bacteria_strain    Phage
#> 1    0   A1        0.002        Strain 1 No Phage
#> 2    0   B1        0.002        Strain 7 No Phage
#> 3    0   C1        0.002       Strain 13 No Phage
#> 4    0   D1        0.002       Strain 19 No Phage
#> 5    0   E1        0.002       Strain 25 No Phage
#> 6    0   F1        0.002       Strain 31 No Phage
```

What's next?

Now that you've merged your data and designs, you can pre-process and plot your data

1. Introduction: `vignette("gc01_gcplyr")`
2. Importing and reshaping data: `vignette("gc02_import_reshape")`
3. Incorporating experimental designs: `vignette("gc03_incorporate_designs")`
4. **Pre-processing and plotting your data: `vignette("gc04_preprocess_plot")`**
5. Processing your data: `vignette("gc05_process")`
6. Analyzing your data: `vignette("gc06_analyze")`
7. Dealing with noise: `vignette("gc07_noise")`
8. Best practices and other tips: `vignette("gc08_conclusion")`
9. Working with multiple plates: `vignette("gc09_multiple_plates")`
10. Using make_design to generate experimental designs: `vignette("gc10_using_make_design")`