Using make_design to generate 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")`**

In `vignette("gc03_incorporate_designs")`, we focused on importing designs from files, since that's the most common way of creating designs. Here, we're going to show how designs can alternatively be generated within `R` using the `gcplyr` function `make_design`.

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

```
library(gcplyr)
```

Including design elements

As a reminder, `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`

For generating designs in `R`, `make_design` can create:

- block-shaped data.frames with your design information (for saving to files)
- tidy-shaped data.frames with your design information (for saving to files and merging with tidy-shaped data)

An example with a single design

Let's start with a simple design.

Imagine you have a 96 well plate (12 columns and 8 rows) with a different bacterial strain in each row, leaving the first and last rows and columns empty.

Row names	Column 1	Column 2	Column 3	...	Column 11	Column 12
Row A	Blank	Blank	Blank	...	Blank	Blank
Row B	Blank	Strain #1	Strain #1	...	Strain #1	Blank
Row B	Blank	Strain #2	Strain #2	...	Strain #2	Blank
...	...	...	...	...	...	...
Row G	Blank	Strain #5	Strain #5	...	Strain #5	Blank
Row G	Blank	Strain #6	Strain #6	...	Strain #6	Blank
Row H	Blank	Blank	Blank	...	Blank	Blank

Typing a design like this manually into a spreadsheet can be tedious. But generating it with `make_design` is easier.

`make_design` first needs some general information, like the `nrows` and `ncols` in the plate, and the `output_format` you'd like (typically `blocks` or `tidy`).

Then, for each different design component, `make_design` needs five different pieces of information:

- a vector containing the possible values
- a vector specifying which rows these values should be applied to
- a vector specifying which columns these values should be applied to
- a string or vector of the pattern of these values
- a Boolean for whether this pattern should be filled byrow (defaults to TRUE)

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12,
  Bacteria = list(c("Str1", "Str2", "Str3", "Str4", "Str5", "Str6"),
                  2:7,
                  2:11,
                  "123456",
                  FALSE)
)
```

So for our example above, we can see:

- the possible values are `c("Strain 1", "Strain 2", "Strain 3", "Strain 4", "Strain 5", "Strain 6")`
- the rows these values should be applied to are `2:7`
- the columns these values should be applied to are `2:11`
- the pattern these values should be filled in by is `"123456"`
- and these values should *not* be filled by row (they should be filled by column)

```
my_design_blk
#> [[1]]
#> [[1]]$data
#>   1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" NA
#> C NA "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" NA
#> D NA "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" NA
#> E NA "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" NA
#> F NA "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" NA
#> G NA "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[1]]$metadata
#> block_name
#> "Bacteria"
```

This produces a `data.frame` with `Bacteria` as the `block_name` in the metadata. If we save this design to a file or transform it to tidy-shaped, this `block_name` metadata will come in handy.

A few notes on the pattern

The pattern in `make_design` is flexible to make it easy to input designs.

The "0" character is reserved for `NA` values, and can be put into your pattern anywhere you'd like to have the value be `NA`

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12,
  Bacteria = list(c("Str1", "Str2", "Str3",
                    "Str4", "Str5", "Str6"),
                  2:7,
                  2:11,
                  "123056",
                  FALSE)
)
my_design_blk
#> [[1]]
#> [[1]]$data
#>   1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" NA
#> C NA "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" NA
```

```
#> D NA "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" NA
#> E NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> F NA "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" NA
#> G NA "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[1]]$metadata
#> block_name
#> "Bacteria"
```

In the previous examples, I used the numbers 1 through 6 to correspond to our values. If you have more than 9 values, you can use letters too. By default, the order is numbers first, then uppercase letters, then lowercase letters (so "A" is the 10th index). However, if you'd like to only use letters, you can simply specify a different `lookup_tbl_start` so that `make_design` knows what letter you're using as the `1` index.

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12, lookup_tbl_start = "A",
  Bacteria = list(
    c("Str1", "Str2", "Str3", "Str4", "Str5", "Str6"),
    2:7,
    2:11,
    "ABCDEF",
    FALSE)
)
```

You can also specify the pattern as a vector rather than a string.

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12,
  Bacteria = list(
    c("Str1", "Str2", "Str3", "Str4", "Str5", "Str6"),
    2:7,
    2:11,
    c(1,2,3,4,5,6),
    FALSE)
)
```

Continuing with the example: multiple designs

Now let's return to our example growth curve experiment. *In addition* to having a different bacterial strain in each row, we now also have a different media in each column of the plate.

Row names	Column 1	Column 2	Column 3	...	Column 11	Column 12
Row A	Blank	Blank	Blank	...	Blank	Blank
Row B	Blank	Media #1	Media #2	...	Media #10	Blank
...	...	...	...	...	...	...
Row G	Blank	Media #1	Media #2	...	Media #10	Blank
Row H	Blank	Blank	Blank	...	Blank	Blank

We can generate both designs with `make_design`:

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12, lookup_tbl_start = "a",
  Bacteria = list(c("Str1", "Str2", "Str3",
                    "Str4", "Str5", "Str6"),
                  2:7,
                  2:11,
                  "abcdef",
                  FALSE),
  Media = list(c("Med1", "Med2", "Med3",
                 "Med4", "Med5", "Med6",
                 "Med7", "Med8", "Med9",
                 "Med10", "Med11", "Med12"),
               2:7,
               2:11,
               "abcdefghij")
  )

my_design_blk
#> [[1]]
#> [[1]]$data
#>   1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" NA
#> C NA "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" NA
#> D NA "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" NA
#> E NA "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" "Str4" NA
#> F NA "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" NA
#> G NA "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[1]]$metadata
#> block_name
#> "Bacteria"
#>
#>
#> [[2]]
#> [[2]]$data
#>   1  2      3      4      5      6      7      8      9      10     11      12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA      NA
#> B NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> C NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> D NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> E NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> F NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> G NA "Med1" "Med2" "Med3" "Med4" "Med5" "Med6" "Med7" "Med8" "Med9" "Med10" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA      NA
#>
#> [[2]]$metadata
#> block_name
#>    "Media"
```

However, the real strength of `make_design` is that it is not limited to simple alternating patterns. `make_design` can use irregular patterns too, replicating them as needed to fill all the wells.

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12, lookup_tbl_start = "a",
  Bacteria = list(c("Str1", "Str2"),
                  2:7,
                  2:11,
                  "abaaabbbab",
                  FALSE),
  Media = list(c("Med1", "Med2", "Med3"),
               2:7,
               2:11,
               "aabbbc000abc"))

my_design_blk
#> [[1]]
#> [[1]]$data
#>   1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Str1" "Str2" "Str1" "Str1" "Str1" "Str1" "Str2" "Str1" "Str1" "Str1" NA
#> C NA "Str2" "Str2" "Str1" "Str2" "Str2" "Str2" "Str2" "Str1" "Str2" "Str2" NA
#> D NA "Str1" "Str1" "Str1" "Str1" "Str2" "Str1" "Str1" "Str1" "Str1" "Str2" NA
#> E NA "Str1" "Str2" "Str2" "Str2" "Str2" "Str1" "Str2" "Str2" "Str2" "Str2" NA
#> F NA "Str1" "Str1" "Str2" "Str1" "Str1" "Str1" "Str1" "Str2" "Str1" "Str1" NA
#> G NA "Str2" "Str2" "Str2" "Str1" "Str2" "Str2" "Str2" "Str2" "Str1" "Str2" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[1]]$metadata
#> block_name
#> "Bacteria"
#>
#>
#> [[2]]
#> [[2]]$data
#>   1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Med1" "Med1" "Med2" "Med2" "Med2" "Med3" NA     NA     NA     "Med1" NA
#> C NA "Med2" "Med3" "Med1" "Med1" "Med2" "Med2" "Med2" "Med3" NA     NA     NA
#> D NA NA     "Med1" "Med2" "Med3" "Med1" "Med1" "Med2" "Med2" "Med2" "Med3" NA
#> E NA NA     NA     NA     "Med1" "Med2" "Med3" "Med1" "Med1" "Med2" "Med2" NA
#> F NA "Med2" "Med3" NA     NA     NA     "Med1" "Med2" "Med3" "Med1" "Med1" NA
#> G NA "Med2" "Med2" "Med2" "Med3" NA     NA     NA     "Med1" "Med2" "Med3" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[2]]$metadata
#> block_name
#>    "Media"
```

There is also an optional helper function called `make_designpattern`, or `mdp` for short. `make_designpattern` just reminds us what arguments are necessary for each design. For example:

```
my_design_blk <- make_design(
  output_format = "blocks",
  nrows = 8, ncols = 12, lookup_tbl_start = "a",
```

```
  Bacteria = mdp(
    values = c("Str1", "Str2", "Str3",
               "Str4", "Str5", "Str6"),
    rows = 2:7, cols = 2:11, pattern = "abc0ef",
    byrow = FALSE),
  Media = mdp(
    values = c("Med1", "Med2", "Med3",
               "Med4", "Med5", "Med6",
               "Med7", "Med8", "Med9",
               "Med10", "Med11", "Med12"),
    rows = 2:7, cols = 2:11, pattern = "abcde0ghij"))
```

```
my_design_blk
#> [[1]]
#> [[1]]$data
#>    1  2      3      4      5      6      7      8      9      10     11     12
#> A NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> B NA "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" "Str1" NA
#> C NA "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" "Str2" NA
#> D NA "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" "Str3" NA
#> E NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#> F NA "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" "Str5" NA
#> G NA "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" "Str6" NA
#> H NA NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>
#> [[1]]$metadata
#> block_name
#> "Bacteria"
#>
#>
#> [[2]]
#> [[2]]$data
#>    1  2      3      4      5      6      7  8      9      10     11      12
#> A NA NA     NA     NA     NA     NA     NA NA     NA     NA     NA      NA
#> B NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> C NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> D NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> E NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> F NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> G NA "Med1" "Med2" "Med3" "Med4" "Med5" NA "Med7" "Med8" "Med9" "Med10" NA
#> H NA NA     NA     NA     NA     NA     NA NA     NA     NA     NA      NA
#>
#> [[2]]$metadata
#> block_name
#>    "Media"
```

For merging our designs with plate reader data, we need it tidy-shaped, so we just need to change the `output_format` to `tidy`.

```
my_design_tdy <- make_design(
  output_format = "tidy",
  nrows = 8, ncols = 12, lookup_tbl_start = "a",
  Bacteria = mdp(
```

```
    values = c("Str1", "Str2", "Str3",
               "Str4", "Str5", "Str6"),
    rows = 2:7, cols = 2:11, pattern = "abc0ef",
    byrow = FALSE),
  Media = mdp(
    values = c("Med1", "Med2", "Med3",
               "Med4", "Med5", "Med6",
               "Med7", "Med8", "Med9",
               "Med10", "Med11", "Med12"),
    rows = 2:7, cols = 2:11, pattern = "abcde0ghij"))
#> Inferred 'into' column names as: Bacteria, Media

head(my_design_tdy, 20)
#>    Well Bacteria Media
#> 1    A1     <NA>  <NA>
#> 2    A2     <NA>  <NA>
#> 3    A3     <NA>  <NA>
#> 4    A4     <NA>  <NA>
#> 5    A5     <NA>  <NA>
#> 6    A6     <NA>  <NA>
#> 7    A7     <NA>  <NA>
#> 8    A8     <NA>  <NA>
#> 9    A9     <NA>  <NA>
#> 10  A10     <NA>  <NA>
#> 11  A11     <NA>  <NA>
#> 12  A12     <NA>  <NA>
#> 13   B1     <NA>  <NA>
#> 14   B2     Str1  Med1
#> 15   B3     Str1  Med2
#> 16   B4     Str1  Med3
#> 17   B5     Str1  Med4
#> 18   B6     Str1  Med5
#> 19   B7     Str1  <NA>
#> 20   B8     Str1  Med7
```

Saving designs to files

If you'd like to save the designs you've created with `make_design` to files, you just need to decide if you'd like them tidy-shaped or block-shaped. Both formats can easily be read back into `R` by `gcplyr`.

Saving tidy-shaped designs

These design files will be less human-readable, but easier to import and merge. Additionally, tidy-shaped files are often better for data repositories, like Dryad. To save tidy-shaped designs, simply use the built-in `write.csv` function.

```
#See the previous section where we created my_design_tdy
write.csv(x = my_design_tdy, file = "tidy_design.csv",
          row.names = FALSE)
```

Saving block-shaped designs

These design files will be more human-readable but slightly more computationally involved to import and merge. For these, use the `gcplyr` function `write_blocks`. Typically, you'll use `write_blocks` to save files in one of two formats:

- `multiple` - each block will be saved to its own `.csv` file
- `single` - all the blocks will be saved to a single `.csv` file, with an empty row in between them

Saving block-shaped designs to multiple files The default setting for `write_blocks` is `output_format = 'multiple'`. This creates one `csv` file for each block. If we set `file = NULL`, the default is to name the files according to the `block_names` in the metadata.

```
# See the previous section where we created my_design_blk
write_blocks(my_design_blk, file = NULL)

# Let's see what the files look like
print_df(read.csv("Bacteria.csv", header = FALSE, colClasses = "character"))
#>   1    2    3    4    5    6    7    8    9   10   11 12
#> A
#> B   Str1 Str1 Str1 Str1 Str1 Str1 Str1 Str1 Str1 Str1
#> C   Str2 Str2 Str2 Str2 Str2 Str2 Str2 Str2 Str2 Str2
#> D   Str3 Str3 Str3 Str3 Str3 Str3 Str3 Str3 Str3 Str3
#> E
#> F   Str5 Str5 Str5 Str5 Str5 Str5 Str5 Str5 Str5 Str5
#> G   Str6 Str6 Str6 Str6 Str6 Str6 Str6 Str6 Str6 Str6
#> H

print_df(read.csv("Media.csv", header = FALSE, colClasses = "character"))
#>   1    2    3    4    5    6 7    8    9   10    11 12
#> A
#> B   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> C   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> D   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> E   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> F   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> G   Med1 Med2 Med3 Med4 Med5   Med7 Med8 Med9 Med10
#> H
```

Saving block-shaped designs to a single file The other setting for `write_blocks` is `output_format = 'single'`. This creates a single `csv` file that contains all the blocks, putting metadata like `block_names` in rows that precede each block.

Let's take a look what the `single` output format looks like:

```
# See the previous section where we created my_design_blk
write_blocks(my_design_blk, file = "Design.csv", output_format = "single")

# Let's see what the file looks like
print_df(read.csv("Design.csv", header = FALSE, colClasses = "character"))
#> block_name Bacteria
#>                  1    2    3    4    5    6    7    8    9   10    11 12
#>          A
```

```
#>          B          Str1 Str1 Str1 Str1 Str1 Str1 Str1 Str1 Str1  Str1
#>          C          Str2 Str2 Str2 Str2 Str2 Str2 Str2 Str2 Str2  Str2
#>          D          Str3 Str3 Str3 Str3 Str3 Str3 Str3 Str3 Str3  Str3
#>          E
#>          F          Str5 Str5 Str5 Str5 Str5 Str5 Str5 Str5 Str5  Str5
#>          G          Str6 Str6 Str6 Str6 Str6 Str6 Str6 Str6 Str6  Str6
#>          H
#>
#> block_name    Media
#>                   1    2    3    4    5    6    7    8    9   10    11 12
#>          A
#>          B          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          C          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          D          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          E          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          F          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          G          Med1 Med2 Med3 Med4 Med5      Med7 Med8 Med9 Med10
#>          H
```

Here we can see all our design information has been saved to a single file, and the metadata has been added in rows before each block.

Merging growth curve data with designs

Once we have both our design and data in `R` and tidy-shaped, we can merge them just the same way as described in `vignette("gc03_incorporate_designs")`