tidyplate: A Versatile R Package for Microplate Data Transformation

Shubham Dutta

2024-11-04

Abstract

Microtiter plates are critical tools in research and diagnostic settings, enabling efficient, high-throughput analysis in assays like ELISA, PCR, and immunoassays. However, working with plate-shaped data present several challenges for data manipulation and analysis. While it is easier to visualize plate-shaped data it is not ideal for downstream data analysis and vice-versa. The *tidyplate* package addresses these issues by transforming microplate-shaped data into tidy data back for easy visualization and data analysis. *tidyplate* supports a variety of standard plate formats and provides robust functions to validate, convert, and export plate data. Here, we introduce the primary functions of *tidyplate*, illustrate its applications, and discuss its benefits for researchers.

Introduction

Microtiter plates are standard tools in laboratories, offering advantages in high-throughput applications, including tissue culture, immunoassays, and diagnostic tests for infectious diseases such as HIV and COVID-19. By allowing the handling of numerous samples in a compact format, microplates streamline data collection and analysis, making them highly efficient and versatile. Yet, the challenges associated with transforming microplate data into analyzable formats remain significant. On one hand it is easier to visualize plate-shaped data but on the other hand it becomes difficult to perform data manipulation or data analysis.

Tidy dataframes are essential in data analysis because they offer a standardized structure that makes data easier to understand and manipulate [2]. In a tidy data, each variable forms a column, each observation forms a row, and each type of observational unit forms a table. This structure minimizes confusion and reduces errors in analysis, as each element is clearly defined and easily accessible. Tidy data facilitates interoperability between different data analysis tools and packages, such as those in the R (Tidyverse) and python (Pandas), which are designed to work seamlessly with tidy data [3,4]. By following a consistent format, tidy dataframes enable analysts to write more efficient and readable code, streamline workflows, and maintain reproducibility, making it easier to share and scale analyses across projects and collaborators.

The *tidyplate* R package aims to bridge this gap, helping users transforming various microplate layouts (6-well, 12-well, 24-well, 48-well, 96-well, 384-well, and 1536-well) from CSV or Excel files into tidy data and back to original plate-shaped format (Figure 1). This enables researchers to view their data and when necessary make them compatible with data manipulation and visualization workflows in R and Python [5,6]. In this article, we detail the key functions of *tidyplate*, demonstrate usage examples, and discuss the practical benefits for data-intensive applications.

Usage and examples

tidyplate has two core functions: `tidy_plate` and `generate_plate`. The `tidy_plate` function takes a specifically formatted input file (csv or excel) as input and transforms it into tidy data (also known as a dataframe or tibble in R). The data for an experiment must be stored such that the top left corner should hold the name of the plate, the column names for each plate should be named as 1, 2, 3 and so on and so forth, the row names should be A, B, C, and finally there must be one empty row between each plate. Figure 2 shows how a csv or excel file for a 12-well plate should be formatted and the same principle can be used on

other plate formats as well. In this experiment researchers looked at the effect of antibiotics on cell lines. The individual plates are named *drug*, *cell_line*, and *percent_survived*. The row and column names are A, B, C and 1, 2, 3, 4 respectively. Each plate is separated by an empty row. `tidy_plate` converts it into tidy data by organizing each plate into columns and adding a *well* column for the well identifiers. On the other hand `generate_plate` does the opposite. It takes tidy data and reformats it back into plate-shaped data. The fill colors are for illustrative purposes only.

Install *tidyplate*:

```
install.packages("tidyplate")
```

The users need to load the package using the `library` function and use the `tidy_plate` function to convert an example plate into tidy data.

```
library(tidyplate)
file <- system.file("extdata", "example_12_well.xlsx", package = "tidyplate")
data <- tidy_plate(file)
```

```
## Plate type: 12-well
```

```
head(data)
```

```
## # A tibble: 6 x 4
##   well  drug      cell_line percent_survived
##   <chr> <chr>     <chr>                <int>
## 1 A01   Neomycin  HEK293                  60
## 2 A02   Puromycin HEK293                  NA
## 3 A03   Neomycin  Hela                    52
## 4 A04   Puromycin Hela                    18
## 5 B01   Neomycin  HEK293                  62
## 6 B02   Puromycin HEK293                  23
```

Any tidy data can be transformed back into plate shaped format using the `generate_plate` function.

```
generate_plate(data, well_id = "well", plate_type = 12, file = "plate_shaped.csv")
```

In addition to the core functions, there are other helper functions to help the users. One of the most useful of these functions is `build_plate`. To assist with formatting, `build_plate` generates empty csv or xlsx templates for each plate type, ensuring compatibility with `tidy_plate`.

```
build_plate(plate_type = 96, n_plates = 2, file = "template_96.xlsx")
```

If the users have formatted their own input files they can use the `check_plate` function to ensure correct formatting before data processing. `check_plate` verifies if a file adheres to *tidyplate*'s input requirements. This function detects formatting issues, such as incorrect row and column identifiers, ensuring smoother workflows.

```
check_plate(file) # No error for valid file
```

```
## example_12_well.xlsx: OK; Plate type: 12-well
```

```
invalid_file <- system.file("extdata", "incorrect_format.csv", package = "tidyplate")
check_plate(invalid_file) # Error type displayed
```

```
## Error:
## ! Verify row and column ids in incorrect_format.csv.
## i Expected column ids: 1, 2, 3, and so on.
## i Expected row ids: A, B, C, and so on.
## i Use the `build_plate()` function to build an empty template.
```

Finally, an utility function called `view_plate_names` retrieves the names of individual plates within a file, providing a quick reference for users working with multi-plate files.

```
view_plate_names(file)
```

```
## [1] "drug"            "cell_line"       "percent_survived"
```

Conclusion

tidyplate is a powerful tool for researchers and analysts, simplifying the transformation of microplate data into tidy formats compatible with R and Python [5,6]. By addressing common challenges in plate data handling, *tidyplate* supports a wide range of research applications, from diagnostics to high-throughput screening. With its flexible functions, researchers can validate, format, and analyze microplate data efficiently, improving the reproducibility and accessibility of their workflows.

References

1. Wickham, H. Tidy Data. *Journal of Statistical Software* **2014**, *59*, 1–23, doi:10.18637/jss.v059.i10.
2. Müller, K.; Wickham, H. *Tibble: Simple Data Frames*; 2023;
3. Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L.D.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; et al. Welcome to the tidyverse. *Journal of Open Source Software* **2019**, *4*, 1686, doi:10.21105/joss.01686.
4. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the Proceedings of the 9th python in science conference; Walt, S. van der, Millman, J., Eds.; 2010; pp. 51–56.
5. R Core Team *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2016;
6. Rossum, G. van *Python Tutorial*; Centrum voor Wiskunde en Informatica (CWI): Amsterdam, 1995;

Figures

Figure 1

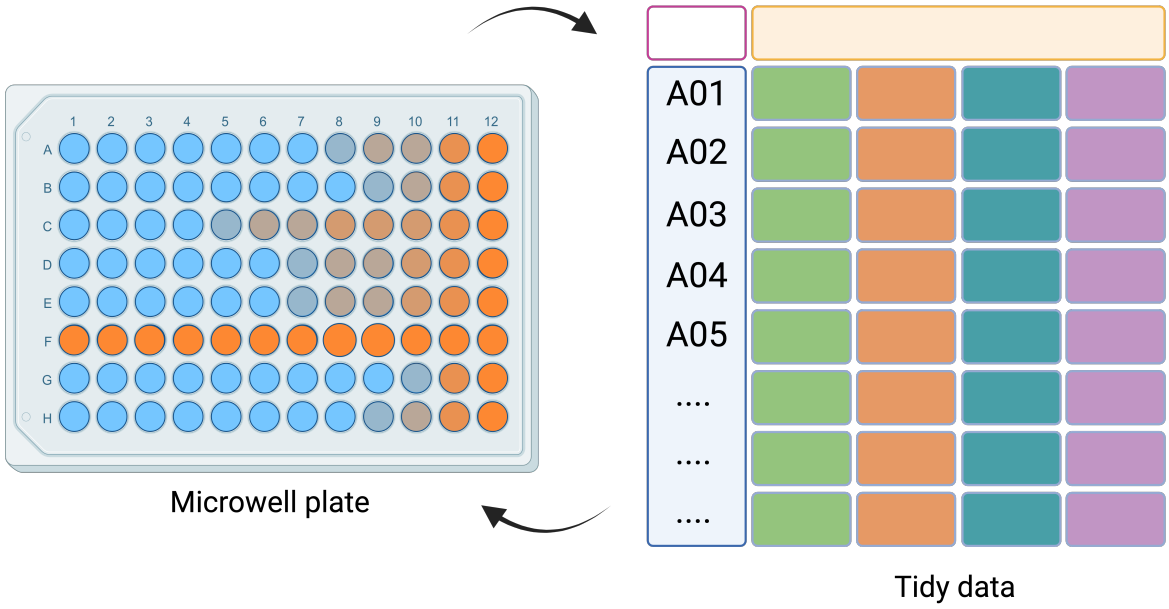
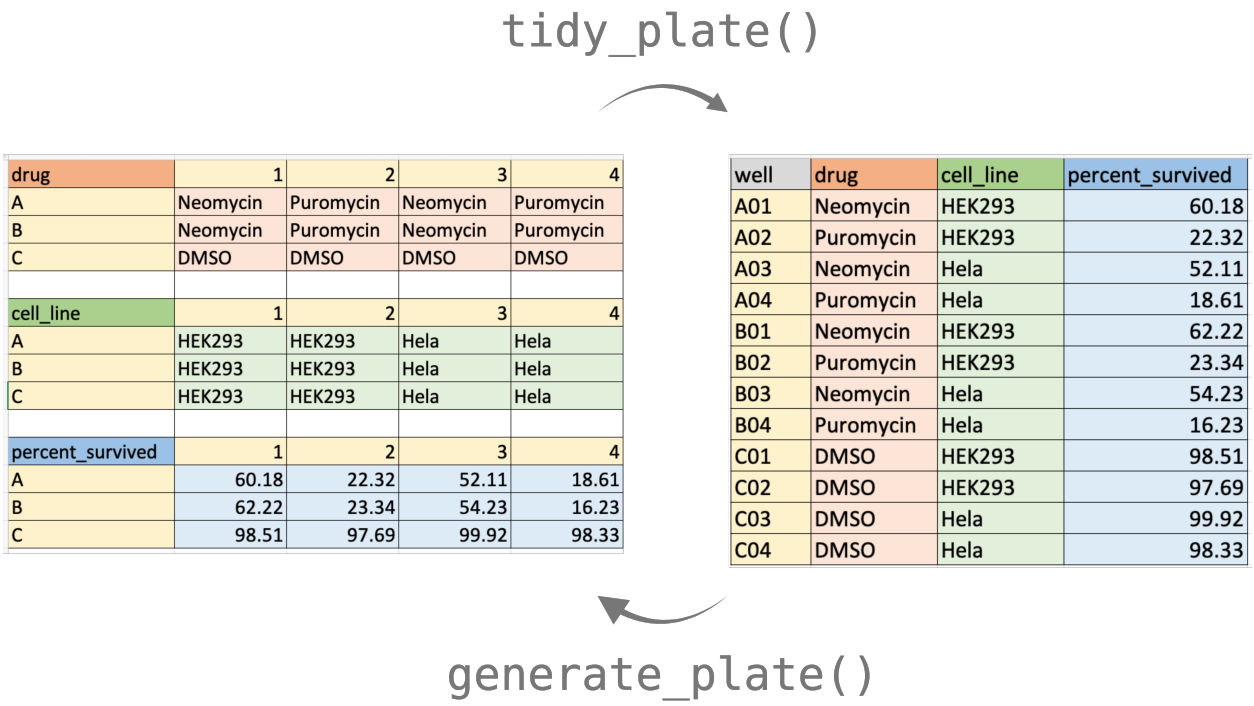


Figure 1: tidyplate transforms microwell plate-format data to tidy data and back

Figure 2



drug	1	2	3	4
A	Neomycin	Puromycin	Neomycin	Puromycin
B	Neomycin	Puromycin	Neomycin	Puromycin
C	DMSO	DMSO	DMSO	DMSO

cell_line	1	2	3	4
A	HEK293	HEK293	Hela	Hela
B	HEK293	HEK293	Hela	Hela
C	HEK293	HEK293	Hela	Hela

percent_survived	1	2	3	4
A	60.18	22.32	52.11	18.61
B	62.22	23.34	54.23	16.23
C	98.51	97.69	99.92	98.33

well	drug	cell_line	percent_survived
A01	Neomycin	HEK293	60.18
A02	Puromycin	HEK293	22.32
A03	Neomycin	Hela	52.11
A04	Puromycin	Hela	18.61
B01	Neomycin	HEK293	62.22
B02	Puromycin	HEK293	23.34
B03	Neomycin	Hela	54.23
B04	Puromycin	Hela	16.23
C01	DMSO	HEK293	98.51
C02	DMSO	HEK293	97.69
C03	DMSO	Hela	99.92
C04	DMSO	Hela	98.33

Figure 2: An illustration of how the core *tidyplate* functions `tidy_plate` and `generate_plate` work on a 12-well plate

Figure details

Figure 1: tidyplate transforms microwell plate-format data to tidy data and back for visualization, data manipulation, and analysis.

Figure 2: An illustration of how the core *tidyplate* functions `tidy_plate` and `generate_plate` work on a 12-well plate.