Multiple batches boxplot

Hsin-Yi Chang

2026-10-08

Examples for generating Boxplots in R

This R Markdown document demonstrates how to make simple boxplots in R. This tutorial focuses on generating boxplots for data from a multiple batches.

The following OS, R version, and R packages were used to develop and test this document:

- Ubuntu 26.04.1 LTS
- R v4.6.1
- `readxl` v1.5.0.1
- `ggplot2` v4.0.3

1. Example Data

Data description: Multiple batches with replicates.

- **`sample2.xlsx`**
  - **Sheet1:** Contains raw data across three columns: `Strain`, `Batch`, `Value`. Each strain has three batches.
  - **Sheet2:** Contains the statistical test result (Tukey's HSD).

2. Preparation

This script requires two R packages: `ggplot2` and `readxl`. The code below checks whether these packages are already installed and automatically installs any missing packages.

```
# Required packages
packages <- c("ggplot2", "readxl")

# Check and install required packages
for (pkg in packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    install.packages(pkg, repos = "https://cloud.r-project.org")
  }
}
```

3. Configuration

Set paths, input/output filenames, variable names, and figure parameters.

3.1 File and Path Settings

Set up a dedicated working directory to store your raw files and output plots.

```
# working directory
working_path = "/scratch/hychang/R_wd/Rana001/Rana001.01/multiple-batches/"

# input data located directory
input_path = "/scratch/hychang/R_wd/Rana001/Rana001.01/multiple-batches/"

# output folder for storing plots
output_path = "/scratch/hychang/R_wd/Rana001/Rana001.01/multiple-batches/plots/"
```

3.2 File and Directory Settings

Specify the names of the input files, worksheet names, and output files.

- **Input file name**, including the file extension
- **Input worksheet name**
- **Output file names**, including the file extension
  - In this example, two boxplots will be generated and exported in both `PDF` and `PNG` formats.

```
# File name of the raw data table, including the file extension
input_file_raw = "sample2.xlsx"

# Worksheet name of the raw data table
input_sheet_raw = "Sheet1"

# File name of the statistical test result table, including the file extension
input_file_stat = "sample2.xlsx"

# Worksheet name of the statistical test result table
input_sheet_stat = "Sheet2"

# Column names in the input data
x_column = "Strain"
y_column = "Value"
batch_column = "Batch"

# Column name of statistical significance group letters,
# skip if the statistical test not available
stat_column = "groups"
```

```
# Output file names of the figures, including the file extension
output_file_1 = "mult-batches_v01.pdf"
output_file_2 = "mult-batches_v02.pdf"
output_file_3 = "mult-batches_v01.png"
output_file_4 = "mult-batches_v02.png"
```

Next, set up a dedicated working directory to store the input files and output plots.

After creating the directories, copy your input files into the `input_path` folder.

```
dir.create(working_path, recursive = TRUE, showWarnings = FALSE)
dir.create(input_path, recursive = TRUE, showWarnings = FALSE)
dir.create(output_path, recursive = TRUE, showWarnings = FALSE)
```

3.3 Figure Settings

- **Figure parameters**
  - Names and font sizes of x- and y- axis.
  - Sample ordering on the x axis
  - Y-axis range and breaks
  - Size of data points and boxes
  - Custom colors for data points

```
# the names of x and y axis
x_lab = "Strain"
y_lab = "Value"

# y axis range and breaks
y_range = c(0, 6)
y_breaks = c(0, 2, 4, 6)

# font size of x and y axis title (axis label)
axis_title_size = 16

# font size of x and y axis text (tick label)
axis_text_size = 14

# sample ordering
x_order = c("A", "B", "C")

# size of data points
point_size = 2

# box size
box_size = 0.4

# space between batches (points)
space_batch = 0.3
```

```r
# use multiple colors for the data points (by batch);
# the number of colors should match the number of batches
color_multiple = c(c("#1B9E77", "#D95F02", "#7570B3"))
```

3.4 Export setting

- For exporting **PDF** files, you can specify the output dimensions using the `width` and `height` parameters, in **inches**.
- For exporting **PNG** files, you can specify the output dimensions using the `width` and `height` parameters, in **pixels**.

```r
## Export setting
pdf_width = 6
pdf_height = 4
png_width = 600
png_height = 400
```

4. Data Preparation

4.1 Setup and Initialization

Set the working directory and load the required packages.

```r
# Set working directory
setwd(working_path)

# Load packages
library(readxl)
library(ggplot2)
```

4.2 Import Raw Data

Before importing the data, make sure the file is located in the `input_path` folder and that the file name (`input_file_raw`) and worksheet name (`input_sheet_raw`) are correct.

```r
# Import raw data
df1 = read_xlsx(file.path(input_path, input_file_raw),
                sheet = input_sheet_raw)
```

4.3 Import Statistical Results (Optional)

Import the results of your statistical test, such as Tukey's HSD or ANOVA. The `groups` column indicates which strains are significantly different from one another.

Note: Skip this step if your dataset does not include statistical test results.

```r
# Import statistical test result
df2 = read_xlsx(file.path(input_path, input_file_stat),
                sheet = input_sheet_stat)
```

4.4 Inspect the Data

Display the data frame and check its dimensions (number of rows and columns).

```r
df1
```

```
## # A tibble: 45 x 3
##    Strain Batch Value
##    <chr>  <chr> <dbl>
##  1 A      1      2.41
##  2 A      1      2.00
##  3 A      1      3.14
##  4 A      1      2.99
##  5 A      1      1.21
##  6 A      2      3.03
##  7 A      2      3.19
##  8 A      2      3.17
##  9 A      2      1.94
## 10 A      2      3.48
## # i 35 more rows
```

```r
dim(df1)
```

```
## [1] 45  3
```

```r
df2
```

```
## # A tibble: 3 x 3
##   Strain Value groups
##   <chr>  <dbl> <chr>
## 1 B       3.63 a
## 2 C       3.34 ab
## 3 A       2.81 b
```

```r
dim(df2)
```

```
## [1] 3 3
```

Optional checks:

- Check the number of observations for each strain, including `NA` values.
- Check the range of the data values.

```r
# Show sample counts, including NA values
table(df1[[x_column]], useNA = "always")
```

```
##
##    A    B    C <NA>
##   15   15   15    0
```

```r
# Show the range of values
range(df1[[y_column]])
```

```
## [1] 1.209719 5.804681
```

```r
# check batch counts, including NA values
table(df1[[batch_column]], useNA = "always")
```

```
##
##    1    2    3 <NA>
##   15   15   15    0
```

5. Generate Plots

Two example boxplots are provided below:

- **Version 1:** Displays data with batches shown in distinct colors.
- **Version 2:** Displays batches in distinct colors and adds statistical significance group labels at the top.

5.1 Version 1: Displays data with batches shown in distinct colors.

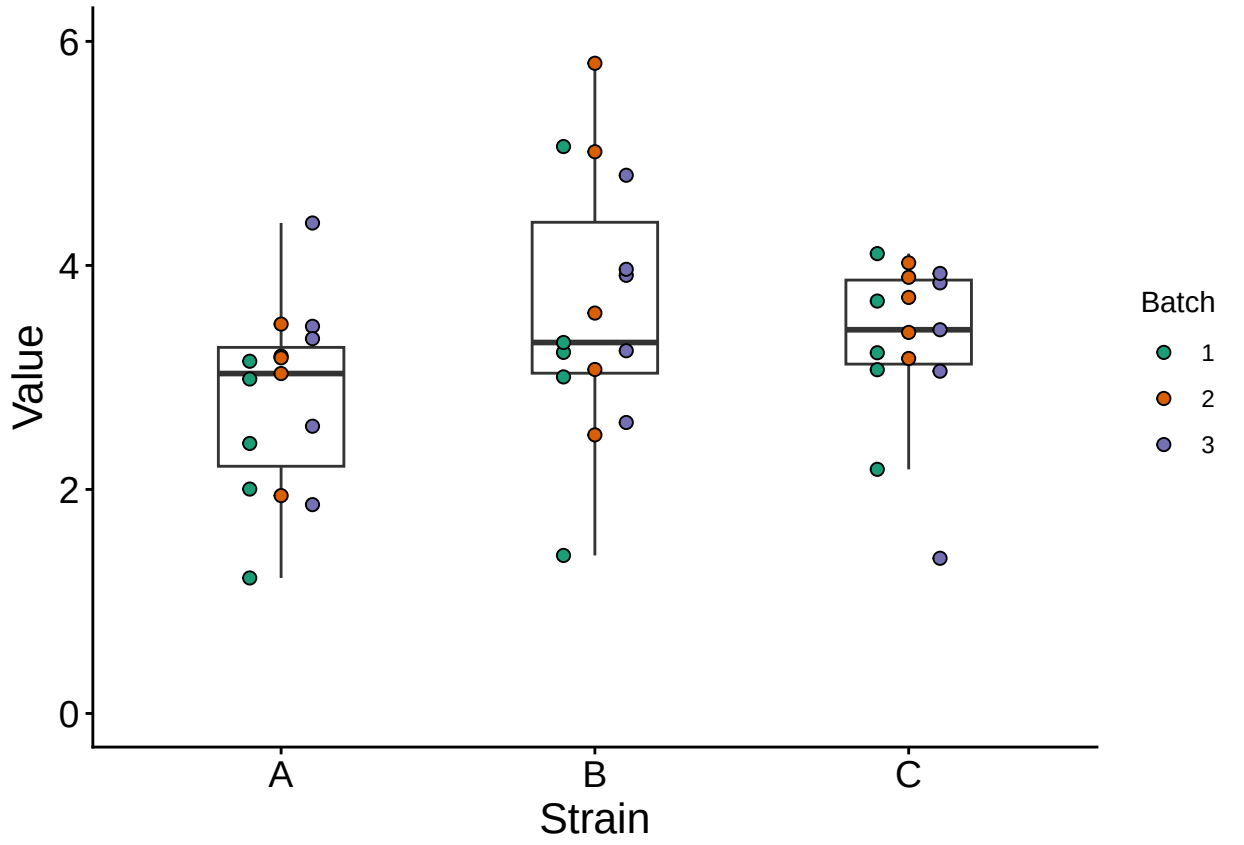
Customization options:

- **Column names:** Make sure `x = Strain` and `y = Value` in `ggplot()` match the corresponding column names in your data table.
- **Point appearance:** Adjust the point color (`color_single`) and point size (`point_size`) in `geom_point()`.
- **Font size:** Adjust the axis title and axis text sizes (`axis_title_size` and `axis_text_size`) in `theme()`.
- **Axis labels:** Update the x- and y-axis labels (`x_lab` and `y_lab`) in `labs()`.
- **Box width:** Adjust the box width (`box_size`) in `geom_boxplot()`.
- **Space between batches:** Adjust the spacing between batches using the `width` (`space_batch`) parameter in `position_dodge()` inside `geom_point()`.

```r
# Version 1
p1= ggplot(data= df1, aes(x= .data[[x_column]], y= .data[[y_column]]))+
  geom_boxplot(width = box_size,
               outlier.shape = NA)+
  geom_point(data= df1,
```

```
            aes(fill= .data[[batch_column]]),
            shape = 21,
            size = point_size,
            position= position_dodge(width = space_batch))+
  scale_fill_manual(values = color_multiple)+
  scale_y_continuous(limits = y_range,
                     breaks = y_breaks)+
  labs(x= x_lab, y= y_lab)+
  theme_classic()+
  theme(axis.title = element_text(size = axis_title_size),
        axis.text = element_text(size = axis_text_size))

# Display plot
p1
```



5.2 Version 2: Displays batches in distinct colors and adds statistical significance group labels at the top.

Confirm which column contains the group letters (e.g., `groups`) and update `label =` inside `geom_text()`. Ensure the x-axis variable name (`Strain`) is identical in both data frames (`df1` and `df2`).

Customization options:

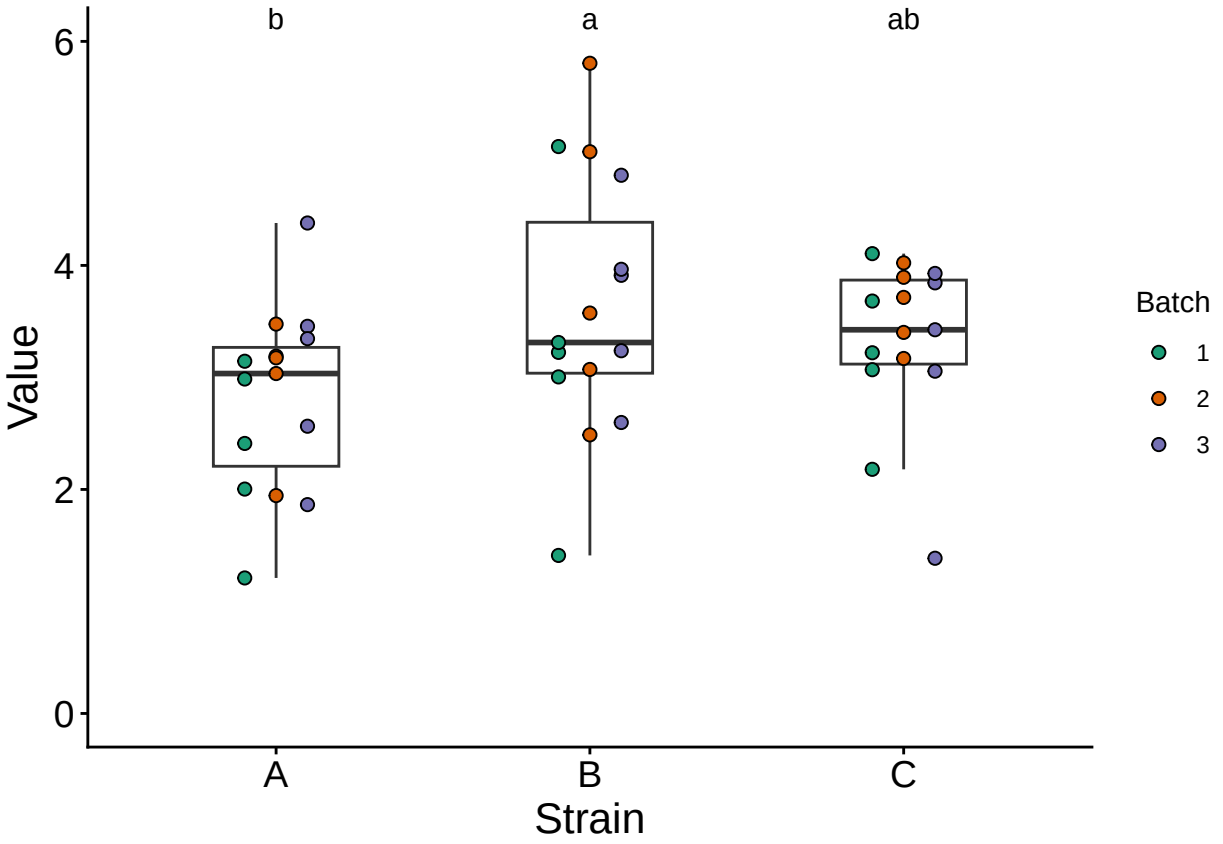
- **Custom colors:** Update the colors in `color_multiple` inside `scale_color_manual()`. Make sure

the number of colors provided matches the number of batches.

- The other customization options are the same as those described for Version 1.

```
# Version 2
p2= ggplot(data = df1, aes(x= .data[[x_column]], y= .data[[y_column]]))+
      geom_boxplot(width = box_size, outlier.shape = NA)+
      geom_point(aes(fill= .data[[batch_column]]),
                 shape = 21,
                 size = point_size,
                 position= position_dodge(width = space_batch))+
      geom_text(data = df2,
                aes(x= .data[[x_column]], y= Inf, label = .data[[stat_column]]),
                vjust= "inward")+
      scale_fill_manual(values = color_multiple)+
      scale_y_continuous(limits = y_range,
                         breaks = y_breaks)+
      labs(x= x_lab, y= y_lab)+
      theme_classic()+
      theme(axis.title = element_text(size = axis_title_size),
            axis.text = element_text(size = axis_text_size))


# Display plot
p2
```



6. Export Plots

Plots can be exported as **PDF** or **PNG** files. Other file formats are also available if needed.

By default, files are saved to the current working directory. To save the files to a specific folder, provide the directory path in the `file.path()` function. Make sure that the destination folder already exists.

6.1 Exporting as PDF

The `pdf()` function exports the plot as a PDF file. You can specify the output dimensions using the `width` and `height` parameters, in inches. Here, we use the width and height defined earlier.

```
# Version 1: PDF export
pdf(file.path(output_path, output_file_1), width = pdf_width, height = pdf_height)
p1
dev.off()

# Version 2: PDF export
pdf(file.path(output_path, output_file_2), width = pdf_width, height = pdf_height)
p2
dev.off()
```

Optional: To customize the figure dimensions, specify the `width` and `height` parameters.

- **Note:** In the examples below, the output file name is specified directly. Replace the file name inside the quotation marks (`""`) with your desired file name.

```
pdf(file.path(output_path, "mult_baches_v1_w8h6.pdf"), width = 8, height = 6)
p1
dev.off()

pdf(file.path(output_path, "mult_baches_v2_w8h6.pdf"), width = 8, height = 6)
p2
dev.off()
```

6.2 Exporting as PNG

The `png()` function exports the plot as a PNG image. By default, the output size is **480 × 480 pixels** with a resolution of **72 dpi**. Here, we use the width and height defined earlier.

```
# Version 1: PNG export
png(file.path(output_path, output_file_3), width = png_width, height = png_height)
p1
dev.off()

# Version 2: PNG export
png(file.path(output_path, output_file_4), width = png_width, height = png_height)
p2
dev.off()
```

Optional: You can customize the image dimensions using `width` and `height`, or adjust the image resolution using `res`.

- **Note:** In the examples below, the output file name is specified directly. Replace the file name inside the quotation marks (`""`) with your desired file name.

```
png(file.path(output_path, "mult_baches_v1_w800h600.png"), width = 800, height = 600)
p1
dev.off()

png(file.path(output_path, "mult_baches_v2_w800h600.png"), width = 800, height = 600)
p2
dev.off()
```