|  |  |
| --- | --- |
| Weplot | (Version 1.30) |

Weplot is a function that can create a variety of figures with a single line of code. It uses the plotting framework of the *ggplot2* package and allows for a range of input data types and formats.

**Installation**

**download.file(url = "https://raw.githubusercontent.com/AldenGriffith/weplot/main/current-version/weplot.R",  
 destfile = "weplot.R")**

**source("weplot.R")**

**Using individual objects as input data**

Weplot can plot x and y variables based on the values of individual objects in a manner similar to the built-in **plot** function. However, the values of multiple objects can be overlaid by combining objects using the **list** function. The following examples highlight the multiple ways in which weplot can work with data stored in individual objects, in this case named **X.Obj**, **Y.Obj**, etc.

|  |
| --- |
| **weplot(Y.Obj)**   * If a single object is given with no formal argument name (i.e. no **x=** or **y=** ), it plots values on the y axis with index values on the x axis. This is the same behavior as the **plot** function. |
| **weplot(y = Y.Obj)**   * This is the same outcome as above; plots values on the y axis with index value on the x axis. |
| **weplot(x = X.Obj)**   * If the **x** argument is formally provided (**x=**) without a **y** input argument, the figure will default to a histogram (**type = "hist"**) unless the **type** argument is provided. |
| **weplot(x = X.Obj, y = Y.Obj)**   * This plots a y object (**Y.Obj**) against an x object (**X.Obj**). * Both objects must have the same length (i.e. same number of values). |
| **weplot(x = X.Obj, y = list(Y.Obj1, Y.Obj2))**   * This overlays two y objects (**Y.Obj1** and **Y.Obj2**) against a common x object (**X.Obj**). * All objects must have the same length. * More than two Y objects can be added to the list, e.g. **list(Y.Obj1, Y.Obj2, Y.Obj3, Y.Obj4, ...)** * The overlay order corresponds to the list order, with the first object listed placed in the background and the last object listed placed in the foreground. |
| **weplot(x = list(X.Obj1, X.Obj2), y = list(Y.Obj1, Y.Obj2))**   * This overlays two y objects (**Y.Obj1** and **Y.Obj2**) against two corresponding x objects (**X.Obj1** and **X.Obj2**). * Each corresponding pair must have the same length (e.g. **X.Obj1** and **Y.Obj1**), but different pairs may have different lengths. * More than two x,y pairs can be added to the list. |

**Using data frame objects as input data**

Weplot can also create figures in a manner similar to *ggplot* by using variables contained within a single data frame object. Overlaying variables in this case requires an existing grouping variable within the data frame. However, you can always overlay two variables in a data frame by passing them to weplot as individual objects as described above, e.g.  
**y = list(Data.Obj$Y.Var1, Data.Obj$Y.Var2**).

|  |
| --- |
| **weplot(x = X.var, y = Y.var, data = Data.Obj)**   * This plots a y variable (**Y.Var**) against an x variable (**X.Var**) contained with the data frame (**Data.Obj**). * This is equivalent to **weplot(x = Data.Obj$X.Var, y = Data.Obj$Y.Var)** |
| **weplot(x = X.var, y = Y.var, data = Data.Obj, group = Group.Var)**   * This groups the data based on the values in the variable **Group.Var**. * The grouping variable can be categorical or continuous |
| **weplot(x = `X variable`, y = "Y variable", data = Data.Obj)**   * Note that either backticks **` `** or double quotes **" "** can be used for variable names that contain spaces. |

If only a single variable is provided, weplot behaves the same whether working with individual objects or data frames. For example, if no formal arguments (**x=** or **y=**) are provided it will plot the values on the y axis with the corresponding index value on the x axis. If the **x** argument is formally provided (**x=**) without a **y** input argument, the figure will default to a histogram (**type = “hist”**) unless the **type** argument is provided.

**Using matrix objects as input data**

When the given y variable is a matrix object, weplot will consider each column to be a separate set of values to overlay. This is similar to the built-in **matplot** function. Currently

**Optional formatting arguments**

|  |  |
| --- | --- |
| **Argument** | **Use and examples** |
| **type** | This describes the type of plot(s) to draw.   * **type = "point"** (default) * **type = "line"** (connects data in order along the x axis) * **type = "point+line"** (point and line overlay) * **type = "path"** (connects data in the order found in the dataset) * **type = "point+path"** (point and path overlay) * **type = "area"** (fills are below y values) * **type = "hist"** (histogram for single variables) * **type = "box"** (boxplot for categorical x variables) * **type = "bar"** (barplot for categorical x variables) |
| **color** | Specifies the color(s) of points, lines, and filled areas. This can be a single color value or a vector that corresponds to the length of the number of categorical groups.   * **color = "red"** ([see all named colors here](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf)) * **color = c("blue", "orange", "darkviolet")** (e.g. for grouped data) * **color = rgb(0.2, 0.1, 0.7)** (red, green, blue - rgb - color mixing) * **color = "#79C470"** (hex color code) |
| **color**  (continued) | If the grouping variable is numeric/continuous then the argument provides colors for a gradient mix or a specified set of colors (e.g. from a palette).   * **color = c("blue", "darkviolet")** (2 color gradient from blue to darkviolet) * **color = c("blue", "orange", "darkviolet")** (3 color gradient) * **color = hcl.colors(100, "viridis")** (100 colors from the viridis palette) |
| **transparency** | Specifies the transparency of all points, lines, and filled areas.   * **transparency = 0** (0%; default) * **transparency = 0.5** (50%) |
| **edge.color** | Specifies the color(s) of the edges of filled areas for boxplots, histograms, area plots, and bar plots. Works just as the **color** argument, but only accepts a single color (not by groups). |
| **size** | Specifies the size of points and thickness of lines.   * **size = 1** (default) * **size = 2** (twice as large as the default size) |
| **xlab**  **ylab** | Specifies custom axis labels. For example:   * **ylab = "Electric power (kW)"** |
| **xlim**  **ylim** | Specifies the range/limit of axes. For example:   * **xlim = c(0, 100)** (x axis from 0 to 100) |
| **group.type** | Specifies how grouping is displayed.   * **group.type = "color"** (group by color; default) * **group.type = "panels"** (individual panel for each group) |
| **group.lab** | Specifies the label for the grouping variable. |
| **group.names** | Specifies the names of group categories. Must be a character vector that matches the number of groups. (Be careful that the order of names matches the grouping order!) For example:   * **group.names = c("Apples", "Oranges")** |
| **error** | Specifies the error bars for bar plots.   * **error = "sd"** (standard deviation; default) * **error = "se"** (standard error) * **error = "none"** (no error bars) |
| **error.width** | Specifies the width of the error bar caps (as a fraction of bar width).   * **error.width = 0.1** (10%; default) |
| **title** | Specifies figure title. For example:   * **Title = "My title"** |
| **log** | Allows you to show your x and/or y axes on a log scale:   * **log = "x"** * **log = "y"** * **log = "xy"** |
| **give.data** | If **TRUE** will return the data frame used to generate the figure. (Only includes x, y, and/or grouping variables). |