Journal of Statistical Software

April 2018, Volume 84, Code Snippet 3. doi: 10.18637/jss.v084.c03

multipanelfigure: Simple Assembly of Multiple Plots and Images into a Compound Figure

Johannes Graumann
Weill Cornell Medicine – Qatar

Richard Cotton
Weill Cornell Medicine – Qatar

Abstract

In scholarly publications, multiple graphical elements such as plots or raster images are traditionally combined into a single compound figure using panels arranged in a grid. Package **multipanelfigure** is a GPL-3 licensed R package that eases assembly of such compound figures, allowing for straightforward integration of R specific plots using base graphics, **lattice** and **ggplot2** plots, as well as **grid** grobs in general. Also provided are facilities to incorporate R external graphical output stored as SVG, PNG, JPEG, or TIFF formatted files.

Keywords: plotting, reproducible research, R.

1. Introduction

In scholarly writing of documents such as reports, journal articles or books, multiple figures are traditionally combined into multi-panel figures. Such figures represent graphical elements arranged in a grid, with individual panels often identified by labels for ease of cross-referencing. Component figures may be plots or raster images such as work flow diagrams and photographs. In the case of R (R Core Team 2018), plots may be created using base graphics, **lattice** (Sarkar 2008), **ggplot2** (Wickham 2009), etc. or may even be raw **grid** grobs.

Organizing the layout of multi-panel figures may be performed interactively using dedicated desktop publishing, image editing, or office software. The manual nature of such assembly renders this prone to human error. Common mistakes are including elements of different resolutions, misalignment and heterogeneous font sizes. Being able to script the layout process is thus a feature useful in the context of aiming at high quality output, reproducible research and speedy generation of a press-ready final product.

1.1. Existing solutions

R provides several solutions for combining multiple plots and images. The `layout` function in package **graphics** allows users to arrange a matrix of base plots. `ggmatrix` in package **GGally** (Schloerke *et al.* 2017) provides similar functionality for plots from package **ggplot2**. The **gtable** (Wickham 2016) package and `grid.arrange` in package **gridExtra** (Auguie 2017) offer more flexible solutions with the same aim and allow for inclusion of arbitrary **grid** grobs. Providing a higher level convenience layer, package **multipanelfigure** (Graumann 2018) is based upon the functionality of package **gtable** and extends it to include base graphics, **lattice**-based output and images in PNG, JPEG, SVG and TIFF format with the addition of convenience functionality for panel labeling, handling of grob dimensions and capture of base graphics. The package falls back on functionality from package **gridGraphics** (Murrell 2018), the successor to package **gridBase**, for capturing base graphics as grobs and **grid**'s `rasterGrob` to include raster images. Package **multipanelfigure** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=multipanelfigure`.

2. Creating a multi-panel figure

2.1. Specifying the layout

The `multi_panel_figure` function is used to create the layout to hold individual panels. Two ways of specifying the dimensions of each panel within the structure have been implemented. Firstly, one may specify the width and height of the entire assembled object, along with the number of rows and columns of panels included. The following creates a multi-panel figure layout setup to arrange panels in two rows and three columns. Measurement units default to millimeters, but all units listed on the `?grid::unit` help page are supported via the `unit` argument. Aside from this **ggplot2**-inspired interface, dimensions may also be provided as **grid** unit objects directly. The `unit` parameter is used to define an internal unique unit if different units are mixed.

```
R> library("multipanelfigure")
R> figure1 <- multi_panel_figure(width = 90, height = 30, columns = 3,
+    rows = 2)
```

Printing a figure object without any panels shows its layout, as demonstrated by Figure 1.

```
R> figure1
```

Notice that in this case, all the rows and columns are the same size. Also notice that by default a 5 mm gap has been created before each row and column (used for panel labels). This may be changed or turned off by adjusting the `row_spacing` and `column_spacing` values.

The arguments `width` and `height` default to `"auto"`, which will derive the figure's dimensions from the graphics device currently in use and works well for development.

A second way of specifying the dimensions of an assembled figure is to pass a vector of row heights and a vector of column widths. This method allows individual rows/columns to have different dimensions (see Figure 2 for an example).

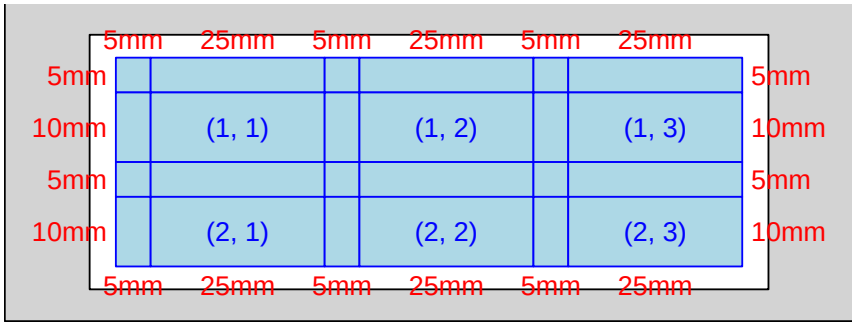

Figure 1: A multi-panel figure layout with equal row heights and column widths.

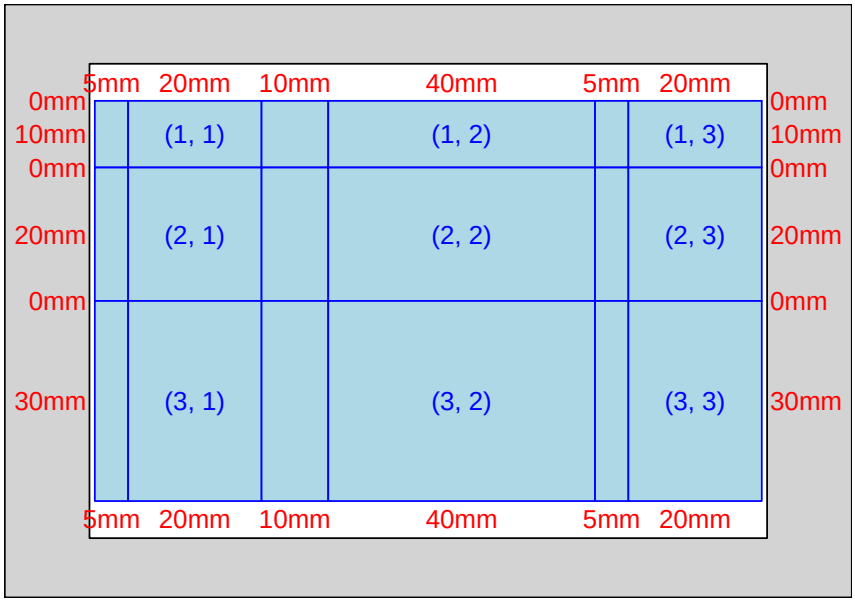

Figure 2: A multi-panel figure layout with varying row heights and column widths.

```
R> figure2 <- multi_panel_figure(width = c(20, 40, 20),
+    height = c(10, 20, 30), row_spacing = 0, column_spacing = c(5, 10))
R> figure2
```

When specifying figure dimensions in this way, minor arithmetic is involved in calculating the size of the total figure, especially when interpanel spacing is used. For convenience, the size of multi-panel figures (or other grobs) may be easily determined using `figure_width` and `figure_height`.

```
R> figure_width(figure2)

[1] 100

R> figure_height(figure2)

[1] 60
```

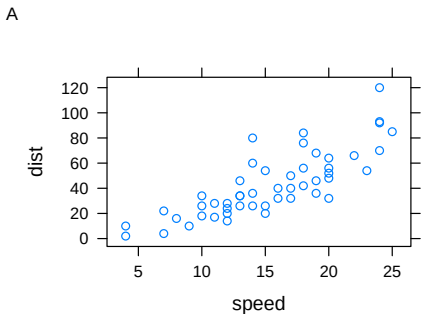

Figure 3: A multi-panel figure with the left-hand panel filled with a scatter-plot of the cars dataset.

2.2. Filling panels

Panels are inserted into a pre-created figure layout using the function `fill_panel`. The first argument to this function is a '`multipanelfigure`' object, and the second argument specifies the plot/grob/image to be included. The function defaults to utilizing the first row with a free panel, followed by the first column in that row with a free panel. Using the `row` and `column` parameters, this behavior may be overridden and specified panels filled instead. In the following example, a **lattice** plot is added by passing the object directly to `fill_panel`. Figure 3 uses the `cars` data set from the **datasets** package, representing stopping distances of cars by speed in the 1920s. Further down a JPEG will be inserted into this figure. As JPEG images are currently stretched to fill the specified panel, height in the following example is chosen to preserve the aspect ratio of the image used.

```
R> figure3 <- multi_panel_figure(width = 200, height = 71.8, rows = 1,
+    columns = 2)
R> library("lattice")
R> p <- lattice::xyplot(dist ~ speed, cars)
R> figure3 <- fill_panel(figure3, p)
```

Once panels have been inserted, the figure may be plotted as usual by printing it – either by typing the variable name or explicitly calling `print`.

```
R> figure3
```

The signature of `fill_panel` is amenable for use with piping syntax. To facilitate use of that add-on functionality, package **multipanelfigure** reexports **magrittr**'s (Bache and Wickham 2014) `%>%` forward pipe operator and `%<>%` compound assignment pipe operators to make them accessible without explicitly loading the package of origin.

The following example uses a pipe to update the '`multipanelfigure`' object with an additional panel, in this case a photo of a car from the 1920s. To include an image file in the figure, simply pass a path to that image (either a path to a local file or a URL). The file type (in this case JPG) is determined from the file extension. The updated multi-panel figure is shown in Figure 4.

```
R> library("magrittr")
R> figure3 %<>% fill_panel(
```

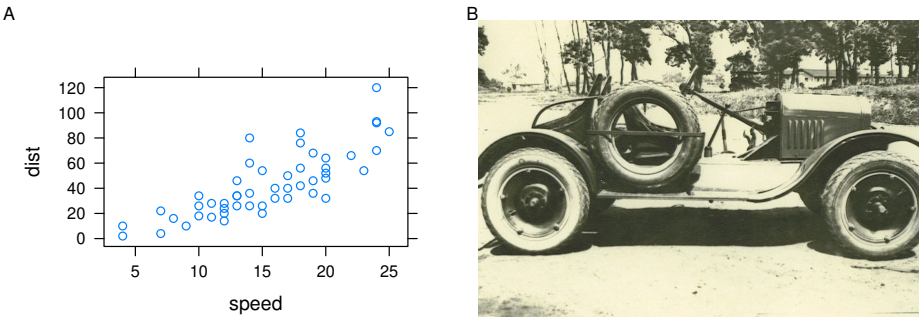

Figure 4: The multi-panel figure from Figure 3, with an additional JPEG image in the right-hand panel.

```
+    paste0("https://upload.wikimedia.org/wikipedia/commons/4/4d/",
+      "Ford-reconnaissance-car-haugh.jpg"), column = 2)
R> figure3
```

By default, panels are labeled with uppercase letters. This may be changed by using the `panel_label_type` argument to `multi_panel_figure`. Alternatively, labels may be set for individual panels using the `label` argument to `fill_panel`.

2.3. Saving a figure

Wrapping functionality provided by **ggplot2** (`ggplot2::ggsave`), `save_multi_panel_figure` provides a convenient means of exporting '`multipanelfigure`' (or any **grid**) objects. The export maintains the dimensions defined for the figure and the syntax is piping-compatible. The graphics format used is derived from the extension of the supplied file name. Currently supported are `"eps"`, `"ps"`, `"tex"` (pictex), `"pdf"`, `"jpeg"`, `"tiff"`, `"png"`, `"bmp"`, `"svg"` and `"wmf"` (Windows only). Figure 4 assembled above may thus be written to a portable network graphics (PNG) file using the following:

```
R> figure3 %>% save_multi_panel_figure(filename = "Cars.png")
```

2.4. A more complex example

Extending the examples to more panels is simple: the creation of a '`multipanelfigure`' object with more rows and columns is followed by further calls to `fill_panel`. The following example recreates Supplementary Figure 4 from Billing *et al.* (2016). This combines images, a ggplot, a grob, and a base plot. First we create the plot and grob objects.

Package **ggplot2** plots from are included in the same way as **lattice** plots, by directly passing the '`ggplot`' object.

```
R> library("dplyr")
R> library("ggplot2")
R> panel_e_barplots <- billing2016_suppfig4e %>%
+    ggplot(aes(Experiment, Intensity)) + geom_bar(stat = "identity") +
+    geom_vline(xintercept = seq(3.5, 24.5, 3), linetype = "dotted") +
```

```
+    facet_wrap(~ GeneName) + xlab(NULL) +
+    theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 3))
```

The **VennDiagram** package (Chen 2018) creates Venn diagrams as 'gList' objects, which are essentially lists of grobs. These may also be added directly to multi-panel figures.

```
R> library("VennDiagram")
R> panel_f_venn <- draw.triple.venn(7129, 7023, 7136, 6536, 6566, 6655, 6312,
+    paste0("EXP", 1:3), scaled = FALSE, col = NA, fill = "grey50",
+    alpha = 1/3, ind = FALSE)
```

To include base graphics, they must first be converted to a **grid** equivalent, which is facilitated by the `capture_base_plot` function provided by package **multipanelfigure**. In the following example, a heatmap is created using an HCL color scale from the **colorspace** package (Zeileis, Hornik, and Murrell 2009; Ihaka, Murrell, Hornik, Fisher, and Zeileis 2016). Base graphics support in package **multipanelfigure** is work in progress and currently placement optimization is required for each individual plot. This is exemplified by the adaptation of `margin` and `cexRow`/`cexCol` in the example.

```
R> library("colorspace")
R> color_scale <- diverge_hcl(25, h = c(150, 0), c = 100)
R> panel_g_heatmap <- capture_base_plot(heatmap(billing2016_suppfig4g,
+    margins = c(12, 5), col = color_scale, cexRow = 0.7, cexCol = 0.7))
```

Constructing the complete figure is simply a matter of calling `multi_panel_figure`, then chaining calls to `fill_panel` for each of the individual plots and images. In the following example, note the use of the `column` argument to specify the column positioning of the panels to be used. By setting it to a range column indices, a plot may span multiple columns. Likewise, by modifying `row`, a panel may take up multiple rows.

For example, in the following code for Figure 5:

- Panel A sets `column = 1:2`. The panel accordingly covers elements (1, 1) and (1, 2).
- Panel E sets `row = 2` and `column = 2:3`. Consequently this panel covers elements (2, 2) and (2, 3).

```
R> figure5 <- multi_panel_figure(width = c(75, 60, 75),
+      height = c(75, 75, 50, 75)) %>%
+    fill_panel("content/billing_2016_supp_fig4_a_design.png",
+      column = 1:2) %>%
+    fill_panel("content/billing_2016_supp_fig4_b_cytosol.png",
+      column = 3) %>%
+    fill_panel("content/billing_2016_supp_fig4_c_gel.png", row = 2) %>%
+    fill_panel("content/billing_2016_supp_fig4_d_western_blot.png",
+      row = 3, column = 1) %>%
+    fill_panel(panel_e_barplots, row = 2, column = 2:3) %>%
+    fill_panel(panel_f_venn, row = 4) %>%
+    fill_panel(panel_g_heatmap, row = 4, column = 2:3)
R> figure5
```

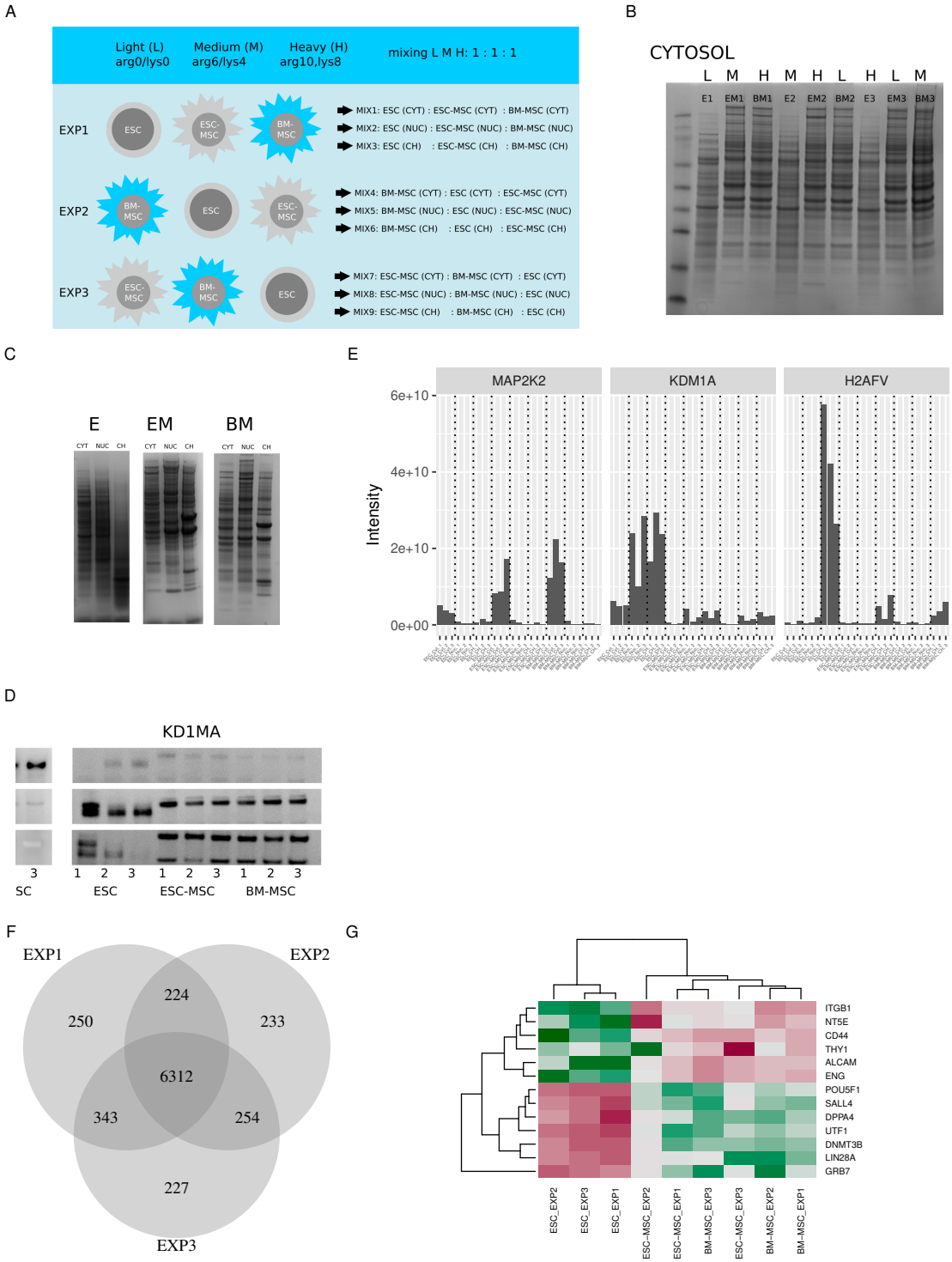

Figure 5: A multi-panel figure reproducing Billing 2016 Supplementary Figure 4. The figure's panels contain several PNG files, a ggplot, a grob, and a base plot.

3. Summary

Package **multipanelfigure** is a GPL-3 licensed R package that enables the scriptable generation of compound figures. Such figures are commonly used in scholarly publications and the package provides an integrated, approachable high-level convenience interface to **grid**-based functionality drawn from packages **gtable** (Wickham 2016), **gridGraphics** (Murrell 2018) and **grid** itself. Inclusion of the corresponding raster graphics formats uses packages **png** (Urbanek 2013a), **jpeg** (Urbanek 2014) and **tiff** (Urbanek 2013b), respectively, and SVG images are imported via package **rsvg** (Ooms 2017).

The resulting tool set makes it possible to script the entire assembly of compound figures traditionally common in scientific literature in a straightforward manner. It thus removes sources of human error inherent to assembly using interactive program options, speeds up the process and renders it completely reproducible and documentable. In particular manuscript preparation with R-based statistical analysis and plotting using tools like packages **rmarkdown** (Allaire *et al.* 2018) and/or **knitr** (Xie 2015), may benefit from the capabilities of a high-level tool for the assembly of compound figures without the need to leave the chosen authoring/development environment.

Acknowledgments

R.J.C, J.G. and the Proteomics Core at WCM-Q are supported by "Biomedical Research Program" funds at Weill Cornell Medicine – Qatar, a program funded by Qatar Foundation.

References

Allaire JJ, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham H, Cheng J, Chang W (2018). ***rmarkdown**: Dynamic Documents for R*. R package version 1.9, URL https://CRAN.R-project.org/package=rmarkdown.

Auguie B (2017). ***gridExtra**: Miscellaneous Functions for **grid** Graphics*. R package version 2.3, URL https://CRAN.R-project.org/package=gridExtra.

Bache SM, Wickham H (2014). ***magrittr**: A Forward-Pipe Operator for R*. R package version 1.5, URL https://CRAN.R-project.org/package=magrittr.

Billing AM, Ben Hamidane H, Dib SS, Cotton RJ, Bhagwat AM, Kumar P, Hayat S, Yousri NA, Goswami N, Suhre K, Rafii A, Graumann J (2016). "Comprehensive Transcriptomic and Proteomic Characterization of Human Mesenchymal Stem Cells Reveals Source Specific Cellular Markers." *Scientific Reports*, **6**(21507), 1–15. doi:10.1038/srep21507.

Chen H (2018). ***VennDiagram**: Generate High-Resolution Venn and Euler Plots*. R package version 1.6.20, URL https://CRAN.R-project.org/package=VennDiagram.

Graumann J (2018). ***multipanelfigure**: Infrastructure to Assemble Multi-Panel Figures (from 'grob's)*. R package version 1.0.0, URL https://CRAN.R-project.org/package=multipanelfigure.

Ihaka R, Murrell P, Hornik K, Fisher JC, Zeileis A (2016). ***colorspace**: Color Space Manipulation*. R package version 1.3-2, URL https://CRAN.R-project.org/package=colorspace.

Murrell P (2018). ***gridGraphics**: Redraw Base Graphics Using **grid** Graphics*. R package version 0.2-1, URL https://CRAN.R-project.org/package=gridGraphics.

Ooms J (2017). ***rsvg**: Render SVG Images into PDF, PNG, PostScript, or Bitmap Arrays*. R package version 1.1, URL https://CRAN.R-project.org/package=rsvg.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Schloerke B, Crowley J, Cook D, Briatte F, Marbach M, Thoen E, Elberg A (2017). ***GGally**: Extension to **ggplot2***. R package version 1.3.2, URL https://CRAN.R-project.org/package=GGally.

Urbanek S (2013a). ***png**: Read and Write PNG Images*. R package version 0.1-7, URL https://CRAN.R-project.org/package=png.

Urbanek S (2013b). ***tiff**: Read and Write TIFF Images*. R package version 0.1-5, URL https://CRAN.R-project.org/package=tiff.

Urbanek S (2014). ***jpeg**: Read and Write JPEG Images*. R package version 0.1-8, URL https://CRAN.R-project.org/package=jpeg.

Wickham H (2009). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag.

Wickham H (2016). ***gtable**: Arrange 'grob's in Tables*. R package version 0.2.0, URL https://CRAN.R-project.org/package=gtable.

Xie Y (2015). *Dynamic Documents with R and **knitr***. 2nd edition. Chapman & Hall/CRC. doi:10.1201/9781315382487.

Zeileis A, Hornik K, Murrell P (2009). "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis*, **53**(9), 3259–3270. doi:10.1016/j.csda.2008.11.033.

Affiliation:

Johannes Graumann
Weill Cornell Medicine – Qatar
Qatar Foundation, Education City, P.O.Box 24144, Doha, State of Qatar

Current address:
Scientific Service Group Biomolecular Mass Spectrometry
Max Planck Institute for Heart and Lung Research
Ludwigstr. 43, D-61231 Bad Nauheim, Germany
E-mail: johannes.graumann@mpi-bn.mpg.de

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2018, Volume 84, Code Snippet 3 *Submitted:* 2016-05-03
doi:10.18637/jss.v084.c03 *Accepted:* 2017-06-30