

The figuRes2 Package for Supporting Large-scale Figure Production with Applications to Clinical Trial Reporting Efforts

Greg Cicconetti, Takeda, Boston, MA, USA

ABSTRACT

The figuRes2 package (<https://CRAN.R-project.org/package=figuRes2>) takes the view that a figure is a collection of graphs/tables assembled on a page and optionally annotated with metadata (titles, headers and footers). The steps to figure building can then be chunked as follows:

1. Data importation
2. Data pre-processing
3. Graph/table building (with subsequent processing necessary)
4. Assembling graph/tables on a page
5. Optional annotation to complete the figure

This package and its vignettes introduce users to functions that had been successfully employed to support clinical trial reporting for a subset of studies at GlaxoSmithKline in 2013-2015 period.

INTRODUCTION

The figuRes2 package provides a suite of functions for producing harmonized figures using the ggplot2 packages. Additional ggplot themes are included. The package provides functions to assist with assembling multiple graphics on a page and annotating the page with headers and footnotes. Functions to facilitate data processing and mass figure production are included. Data sets are included to demonstrate how the functions work and this document contains a section that walks through the workflow for large scale figure production.

All graphing functions in this package presume a data.frame is supplied with a specific data structure. In practice these can be either imported (e.g., as a .csv file) or generated with R (e.g., output of simulation or call to a probability distribution function).

Data pre-processing of imported files may be required to ensure the data.frames are organized properly, factors are properly organized and labeled appropriately, etc. To handle this, the user may wish to author functions to assist with this pre-processing. The demog.data data set and related process.bslchar function provide an example.

The build.page function is designed to help visualize how graphics are organized on a page, as well as execute the task. The graphics passed to this function can be created with the functions in this package or by the user. With the former, keep in mind that these are merely functions that facilitate the construction of ggplot objects.

In the simplest case a figure will consist of a single graphic.

Some figures call for augmenting a graphic with a table (e.g., forest plots, Kaplan-Meier curves). In these cases, the tables are built using either table.plot or nsubj.plot (or again, the user coded ggplot text table). In the case of Kaplan-Meier curves, it is standard practice to arrange the KM curve on top of a table reporting the Number at Risk. Other figures call for juxtaposing two figures. In these cases, the task is either to arrange 2 graphics in a 1 (row) x 2 (col) or a 2 x 1 grid. More generally, the task is to arrange a dashboard of graphics/tables on an nrow x ncol grid and place them on page with predefined margins.

Once the individual graphs/tables have been created for a figure, pre-processing may be required. E.g., there may be a need to align the y-axes when stacking graphics: if Graph A has the longest y-axis tick label, Graph B will need to be adjusted so graphics are aligned when arranging them on a 2 x 1 grid.

When the collection of graphs/tables have been pre-processed, they can be passed to the `build.page` function. This function requires the user to specify how the row widths and column heights should be specified as well as the order in which to populate the cells of the grid of graphics.

The defaults presume figures are being displayed on an 8.5 inch x 11 inch page, with landscape orientation and margins of 1.5 inches at the top and bottom and 1 inch margins at the left and right. These dimensions provide sufficient room for 2 lines of headers, 4 lines of footnotes and a effective central region for graphs and tables of size (8.5 - 3) inch x (11 - 2) inch. Generalizing from the defaults is straightforward. Trial and error will be required to fine tune aesthetic aspects.

The function `annotate.page` has been coded to optionally populate with blank entries (helpful when building graphics that don't require annotation and where margins are minimized), dummy entries (helpful in development phases) or entries coming from a `data.frame` called `outputplan` (helpful for mass figure production).

The `build.page` and `annotate.page` functions

The `build.page` function is a wrapper for the `gridExtra::grid.arrange` function. It can be used to help visualize how a page will be partitioned. However, its primary purpose is to arrange a list of graphic objects on a page.

Visualizing page partitioning

Throughout this document we work under the assumption that figures are assembled on an 8.5 x 11 inch page with landscape orientation, though generalizing to other dimensions is straightforward. In the simplest case, a single graphic populates the page. The defaults of the `build.page` function allocate a predetermined value for left, right, top and bottom margins. These defaults provide sufficient space for the 3 lines of headers and the 5 lines for footnotes presumed by the `figures2::annotate.page` function. By adding a call to `annotate.page`, we can see how page division looks together with dummy headers and footers.

```
require(figuRes2)
require(grid)
require(gridExtra)
require(ggplot2)
require(scales)
#pdf("figure1 - testing dimensions.pdf", h=8.5, w=11)
build.page(interior.h=c(1),
interior.w=c(1),
ncol=1, nrow=1,
test.dim=TRUE)
annotate.page(override="")
#dev.off()
```

(1, 1)	Upper Left Header 1 Upper Left Header 2 Upper Left Header 3	(1, 2)	Upper Right Header 1 Upper Right Header 2 Upper Right Header 3	(1, 3)
(2, 1)	Figure 1.100 If ggplot populates title, annotate.page's title argument gets a list of whitespace text strings. If annotate.page is populating titles, use whitespaces and newline escape characters in ggplot titles to ensure ggplot object is shrunk titles do not stamp over your graphs			(2, 3)
(3, 1)	Footnote1: Up to five lines of footnotes can be annotated. Footnote2: Graphic region height can be flexed. Footnote3 Footnote4 Footnote5: In large-scale production, this may hold file name, time stamp, etc. 08FEB2015 23:00			(3, 3)

Figure 1: A figure built using a single graphic with top and bottom margins set default values

Note: Expect distortion when running this code. The ultimate destination is the 8.5 in x 11 in page. Uncomment the pdf and dev.off calls to see what final product looks like.

Tweaking the arguments of the build.page function

Suppose two graphics are to populate a figure. E.g., forest plots typically juxtapose a graphic reporting on a stacked collection of confidence intervals with a table to the right or left reporting summary statistics. In designing a forest plot, one needs to decide how to

- allocate the page's real estate for the graphic, table, and margins,
- ensure proper alignment of labels, line segments and text, and
- annotate the page with footnotes, headers, etc.

Alternatively, a Kaplan-Meier figure may stack a graphic of the survival curves and a table reporting Number of Subjects at Risk. In this case, it is important to ensure proper alignment of the x-axes.

In designing forest plot [Kaplan-Meier] figures, we may broadly think of partitioning the page to accommodate a 1x2 [2x1] 'matrix' of graphics components. In thinking about page layout, this matrix, gets padded on top, right, bottom and left by the page margins. The build.page function helps to focus attention on graphic components, while still providing access to margins. Motivating application will follow.

```
build.page(interior.h=c(1),
interior.w=c(.5, .5),
ncol=2, nrow=1,
test.dim=TRUE)
annotate.page(override="")
```

In this example, a 3x1 grid of graphics is planned.

```
build.page(interior.h=c(1/3,1/3,1/3),
interior.w=c(1),
ncol=1, nrow=3,
test.dim=TRUE)
annotate.page(override="")
```

An example with unequal allocation:

```
build.page(interior.h=c(2, 1, 3)/6,
interior.w=c(.6, .4),
ncol=2, nrow=3,
test.dim=TRUE)
annotate.page(override="")
```

Tweaking the page.heights and page.widths arguments to manipulate graph region

In some applications we might want to manipulate the margins or make them negligible. E.g., suppose a figure is needed exclusively for a PowerPoint presentation and header and footers are not needed.

```
build.page(interior.h=c(1/3,1/3,1/3),
interior.w=c(.5, .5),
ncol=2, nrow=3,
test.dim=TRUE,
top.margin=.1,
bottom.margin=.1,
right.margin=.1,
left.margin=.1)
```

(1, 1)	Upper Left Header 1 Upper Left Header 2 Upper Left Header 3	(1, 2)	Figure 1.100	(1, 3)	Upper Right Header 1 Upper Right Header 2 Upper Right Header 3	(1, 4)
(2, 1)		(2, 2)	If ggplot populates title, annotate.page's title argument gets a list of whitespace text strings. If annotate.page is populating titles, use whitespaces and newline escape characters in ggplot titles to ensure ggplot object is shrunk titles do not stamp over your graphs	(2, 3)		(2, 4)
(3, 1)	Footnote1: Up to five lines of footnotes can be annotated. Footnote2: Graphic region height can be flexed. Footnote3 Footnote4 Footnote5: In large-scale production, this may hold file name, time stamp, etc.	(3, 2)	08FEB2015 23:00	(3, 3)		(3, 4)

Figure 2: A figure built using a two graphics side-by-side.

(1, 1)	Upper Left Header 1 Upper Left Header 2 Upper Left Header 3	(1, 2) Figure 1.100	Upper Right Header 1 Upper Right Header 2 Upper Right Header 3	(1, 3)
(2, 1)	If ggplot populates title, annotate.page's title argument gets a list of whitespace text strings. If annotate.page is populating titles, use whitespaces and newline escape characters in ggplot titles to ensure ggplot object is shrunk titles do not stamp over your graphs			(2, 3)
(3, 1)	(3, 2)			(3, 3)
(4, 1)	(4, 2)			(4, 3)
(5, 1)	Footnote1: Up to five lines of footnotes can be annotated. Footnote2: Graphic region height can be flexed. Footnote3 Footnote4 Footnote5: In large-scale production, this may hold file name, time stamp, etc. 08FEB2015 23:00			(5, 3)

Figure 3: A figure built using 3 graphics stacked with annotation.

(1, 1)	Upper Left Header 1 Upper Left Header 2 Upper Left Header 3	(1, 2)	Figure 1.100	(1, 3)	Upper Right Header 1 Upper Right Header 2 Upper Right Header 3	(1, 4)
(2, 1)	If ggplot populates title, annotate.page's title argument gets a list of whitespace text strings. If annotate.page is populating titles, use whitespaces and newline escape characters in ggplot titles to ensure ggplot object is shrunken titles do not stamp over your graphs					(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)			
(4, 1)	(4, 2)	(4, 3)	(4, 4)			
(5, 1)	Footnote1: Up to five lines of footnotes can be annotated. Footnote2: Graphic region height can be flexed. Footnote3 Footnote4 Footnote5: In large-scale production, this may hold file name, time stamp, etc. 08FEB2015 23:00	(5, 2)	(5, 3)	(5, 4)		

Figure 4: Example 1d: A figure built using 3x2 grid of graphics.

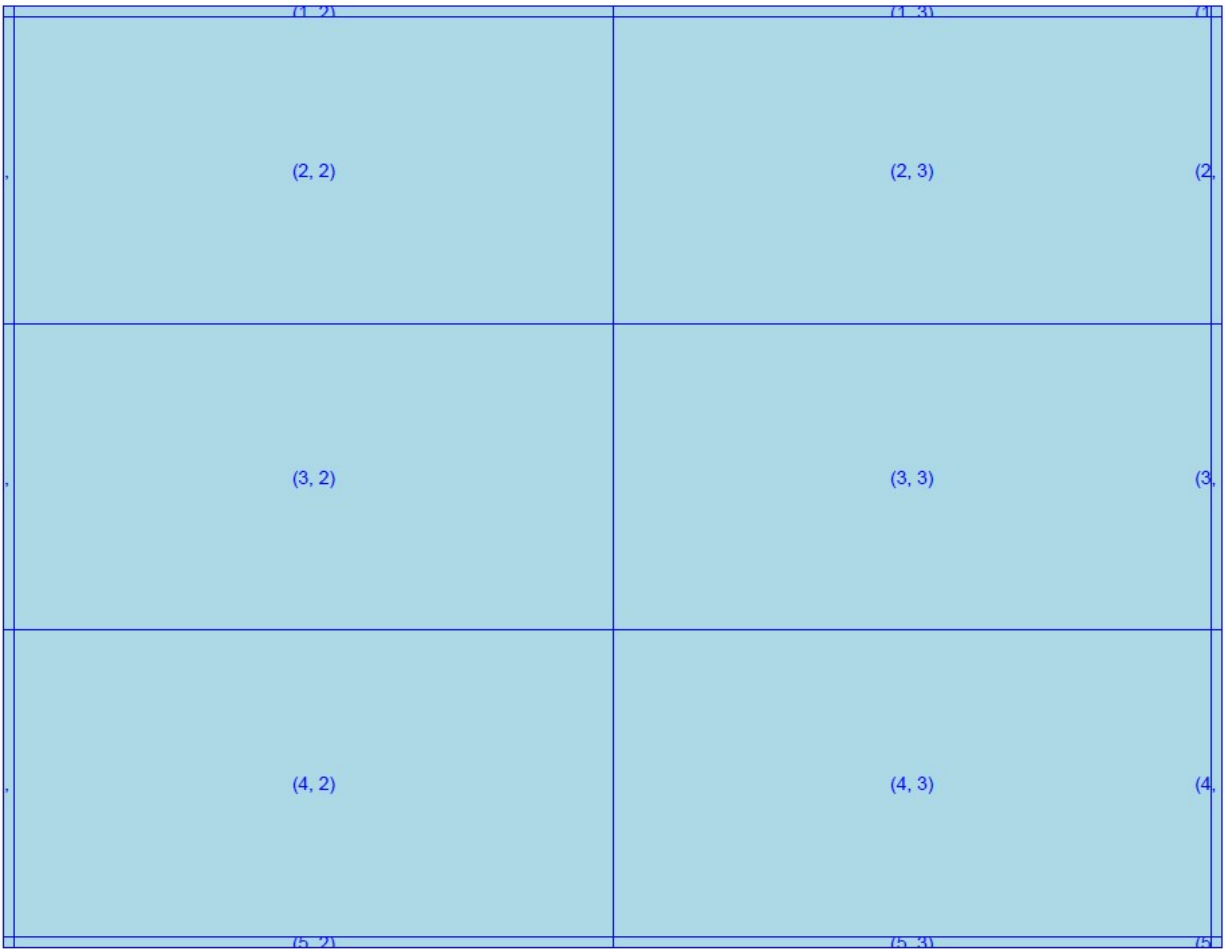


Figure 5: Shrinking the perimeter reserved for margins

Code along the following lines was used in an iterative fashion to land on values of .92 and .165. Trial and error can be used to fine tune dimensions. In the first series of figures produced in the pdf, top and bottom margins are set to be large, in the second series, top and bottom margins are shrunk by 0.5 in to increase the region of the page for graphics.

The value in such an exercise: In large scale figure production, it may be desirable to build figures that dynamically adjust the page layout to based on the number of lines required for titles and footers.


```
pdf("testdim.pdf", h=8.5,w=11)
for(bm in seq(.92,2,.165)){
  build.page(interior.h=c(1/3,1/3,1/3),
interior.w=c(.5, .5),
ncol=2, nrow=3,
test.dim=TRUE,
top.margin=1.6,
bottom.margin=bm)
  annotate.page(override="", title=list(bm,"title2","title3","title4"),
top.margin=1, bottom.margin=1)
}
for(bm in seq(.92-.5,2-.5,.165)){
  build.page(interior.h=c(1/3,1/3,1/3),
interior.w=c(.5, .5),
ncol=2, nrow=3,
test.dim=TRUE,
top.margin=1.6-.5,
bottom.margin=bm)
  annotate.page(override="", title=list(bm,"title2","title3","title4"))
}
dev.off()
```

By working along similar lines, one could determine the `build.page` and `annotate.page` margin parameters that would be suited using other page dimension and orientations, e.g., legal page with portrait orientation.

The `plot.margin` theme option and its relation to `build.page`

The `figures2::default.settings` functions includes a call to the following `ggplot2` function:

```
theme_set(theme_grey2_nomargins())
```

The `figures2` package includes a handful of themes which are slight variations on either the `ggplot2` default theme, `theme_grey`, or `theme_bw`. In particular, `theme_grey2_default_margins` and `theme_grey2_nomargins` are two variations. Relative to `theme_grey` these themes have:

- `axis.text` color changed from "grey50" to "black"
- `legend.position` changed from "right" to "bottom"
- `legend.direction` changed to "horizontal"
- `theme_grey2_default_margins` **retains**: `plot.margin = unit(c(1, 1, 0.5, 0.5), "lines")`
- `theme_grey2_nomargins` **changes**: `plot.margin = unit(c(0, 0, 0, 0), "in")`

The `default.settings` functions sets the theme to `theme_grey2_nomargins` since the `build.page` function controls global page margins. By setting `plot.margins` to zero the page real estate available for graphics is maximized relative to the page layout scheme and margins dictated to `build.page`. Should there be a need to alter the `plot.margins` of the individual graphics objects being passed to `build.page`, one can do so within the build of those graphics or by resetting the theme used.

An example where this might be used: Suppose we are juxtaposing two graphics and want to increase the padding between the left and right graphics. If `left.graphic` and `right.graphic` are `ggplot` graphics built under `theme_grey2_nomargins`, then the following code would give `left.graphic` and `right.graphic` padding of 0.1 inches on left and right sides, respectively. Negative values could also be used: replacing 0.1 with -0.1 would shrink the padding between the two graphics. Note that this can lead to the second graphic obscuring the first.

```
# The vector holds c(top, right, bottom, left) margin measurements.
left.graphic <- left.graphic + theme(plot.margin=unit(c(0,.1,0,0), "in"))
right.graphic <- right.graphic + theme(plot.margin=unit(c(0,0,0,.1), "in"))
grid.arrange(left.graphic, right.graphic, nrow=1)
```

Applications of build.page

In this section we'll populate the page with graphics. First we start a session:

```
remove(list=ls())
require(figures2)
default.settings()
```

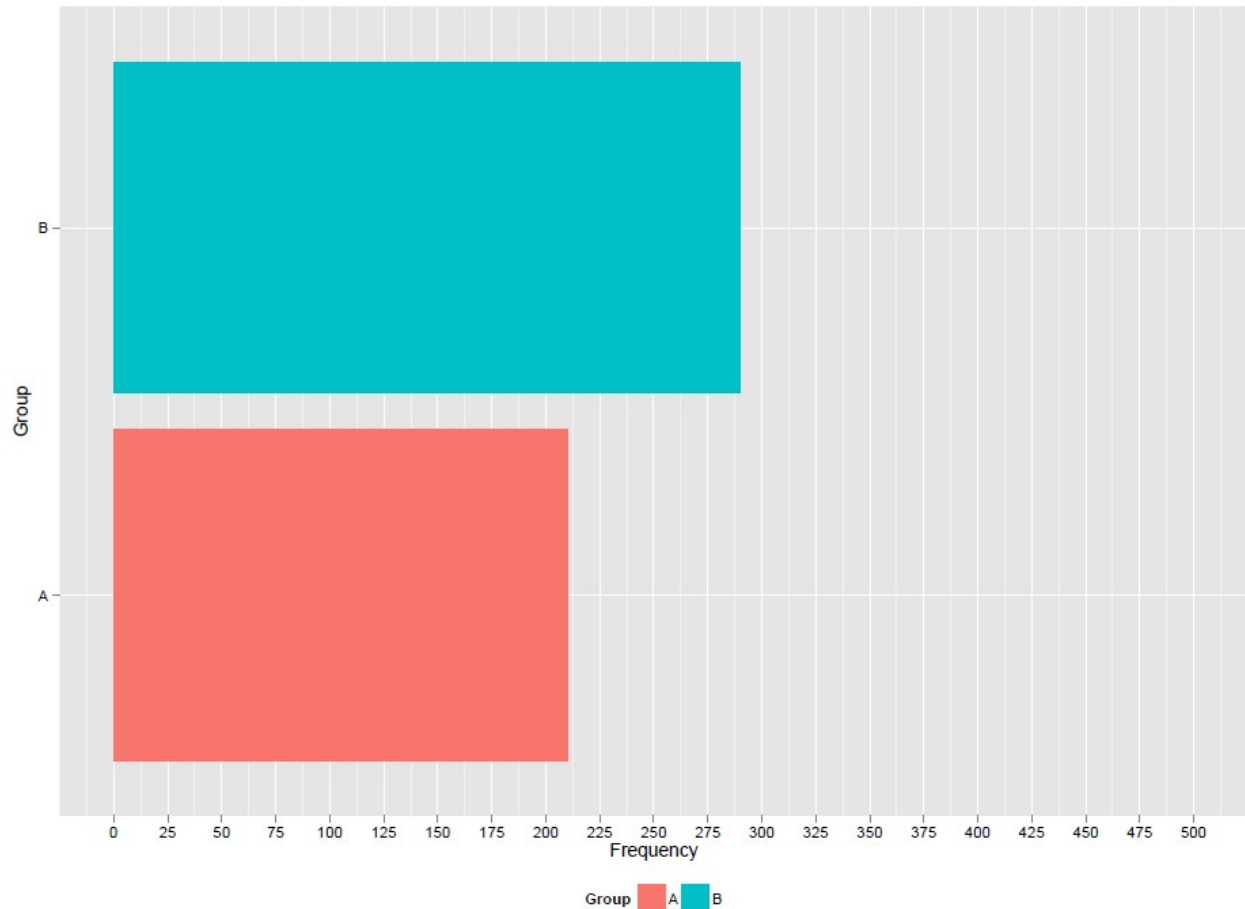
Example 1: A bar chart

Some data are manufactured. A data.frame is built with

- random standard normals in column 'x',
- random group categorical variable assignments in column 'group'
- response variable in column 'y'

```
set.seed(8675309) # To ensure reproducibility
working.df <- data.frame(x=rnorm(500, 0, 1))
working.df$group <- factor(sample(x=c("A", "B"), replace=TRUE,
size=nrow(working.df), prob=c(.4,.6)))
working.df$y <- working.df$x +
as.numeric(working.df$group)*working.df$x +
rnorm(n = nrow(working.df), 0, 3)
A simple bar chart is created.
ex.bar <- ggplot(data=working.df, aes(x=group, fill=group)) +
geom_bar() +
labs(x="Group", y="Frequency", title="", fill="Group") +
scale_y_continuous(limits=c(0,500), breaks=seq(0,500,25)) +
coord_flip()

print(ex.bar)
```



Adding annotation

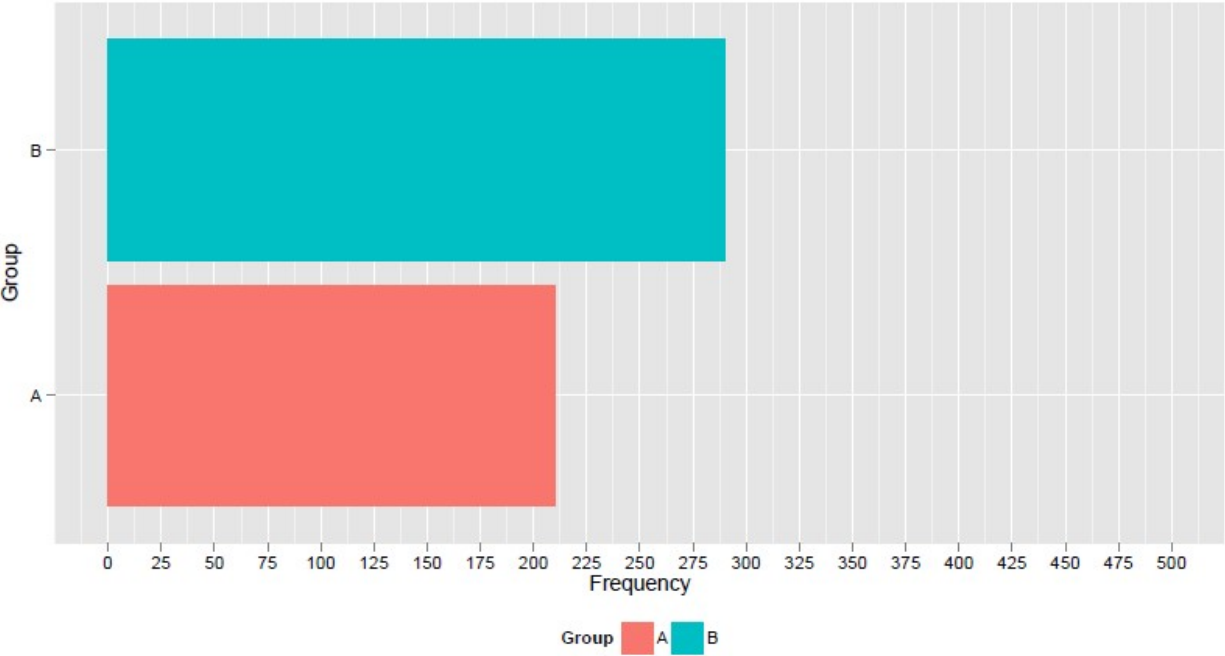
The following call to `annotate.page` gives a sense of the final product.

```
build.page(interior.h = c(1),
interior.w = c(1),
ncol=1,
nrow=1,
interior = list(ex.bar + ggtitle("\\n\\n\\n\\n\\n"))
annotate.page(override = "")
```

Sending the graphic to a pdf file and demonstrating how to accommodate multiple title lines

The following code will create a pdf file in your current working directory. Recall that `annotate.page` is adding the title to the figure. This series demonstrates that using the newline escape character provides the means by which the graphic is shrunken to accommodate multiple title lines.

Figure 1.100
If ggplot populates title, annotate.page's title argument gets a list of whitespace text strings. If annotate.page is populating titles, use whitespaces and newline escape characters in ggplot titles to ensure ggplot object is shrunken titles do not stamp over your graphs



Footnote1: Up to five lines of footnotes can be annotated.
Footnote2: Graphic region height can be flexed.
Footnote3
Footnote4
Footnote5: In large-scale production, this may hold file name, time stamp, etc. 08FEB2015 23:00

Figure 7: An assembled bar chart figure

```

getwd() # The file will be placed in this location
pdf("barchart.pdf", height=8.5, width=11)
# In the build of ex.bar title="" allows for room for a single title line
build.page(interior.h = c(1),
interior.w = c(1),
ncol=1,
nrow=1,
interior=list(ex.bar))
annotate.page(override = "")
# manipulating the title of the ggplot object allows for two lines
build.page(interior.h = c(1),
interior.w = c(1),
ncol=1,
nrow=1,
interior=list(ex.bar+ggtitle("\n")))
annotate.page(override = "")
# manipulating the title of the ggplot object allows for three lines
build.page(interior.h = c(1),
interior.w = c(1),
ncol=1,
nrow=1,
interior=list(ex.bar+ggtitle("\n\n")))
annotate.page(override = "")
# manipulating the title of the ggplot object allows for four lines
build.page(interior.h = c(1),
interior.w = c(1),
ncol=1,
nrow=1,
interior=list(ex.bar+ggtitle("\n\n\n")))
annotate.page(override = "")
dev.off() # Shuts the pdf device

```

In some applications using the `annotate.page` function is inconvenient. (See the example above where page margins were shrunk to be negligible.) In these cases, build the title into your ggplot object rather than use blank titles.

Example 2: Assembling a scatterplot with marginal densities

Initialize a session and generate data similar to the previous bar chart example.

```

remove(list=ls())
require(figures2)
default.settings()
working.df <- data.frame(x=rnorm(500, 0, 1))
working.df$group <- factor(
sample(x=c("A", "B"), replace=TRUE, size=nrow(working.df), prob=c(.4,.6)))
working.df$y <- working.df$x +
as.numeric(working.df$group)*working.df$x +
rnorm(n = nrow(working.df), 0, 3)
head(working.df)

```

	x	group	y
1	-0.4064429	A	3.100391
2	-0.3690869	B	3.311304
3	-1.5428036	B	-1.017126
4	-0.2147842	B	-6.528522
5	-0.1271307	B	2.003076
6	-0.8606796	A	-4.809386

Here's a useful exploratory data analysis figure. The build of main.plot is full of customization; step through the builds. First determine the limits of the data:

```
xmin <- min(working.df$x)
xmax <- max(working.df$x)
ymin <- min(working.df$y)
ymax <- max(working.df$y)
```

Building the graphical components

Scatterplot The following build:

- associates color with group and shape
- alters the size and transparency of the symbols
- adds OLS lines with confidence bands
- alters labels
- changes symbols and colors
- manipulates the limits and tick mark locations of the axes
- the expand argument removes padding (remove this argument and contrast result)
- repositions the legend

```
main.plot <- ggplot(data=working.df, aes(x=x,y=y, color=group, shape=group)) +
  geom_point(size=3, alpha=.3) +
  geom_smooth(method = "lm", size=.75) +
  labs(x="x values", y="y values", color="Group", shape="Group") +
  scale_shape_manual(values=c(16, 17)) +
  scale_color_manual(values=c("red", "blue"))+
  scale_x_continuous(limits=c(xmin+.5,xmax+.5), breaks=seq(-4,4,1), expand=c(0,0))+
  scale_y_continuous(limits=c(ymin+.5,ymax+.5), expand=c(0,0))+
  theme(legend.position=c(.15, .8))
print(main.plot)
```

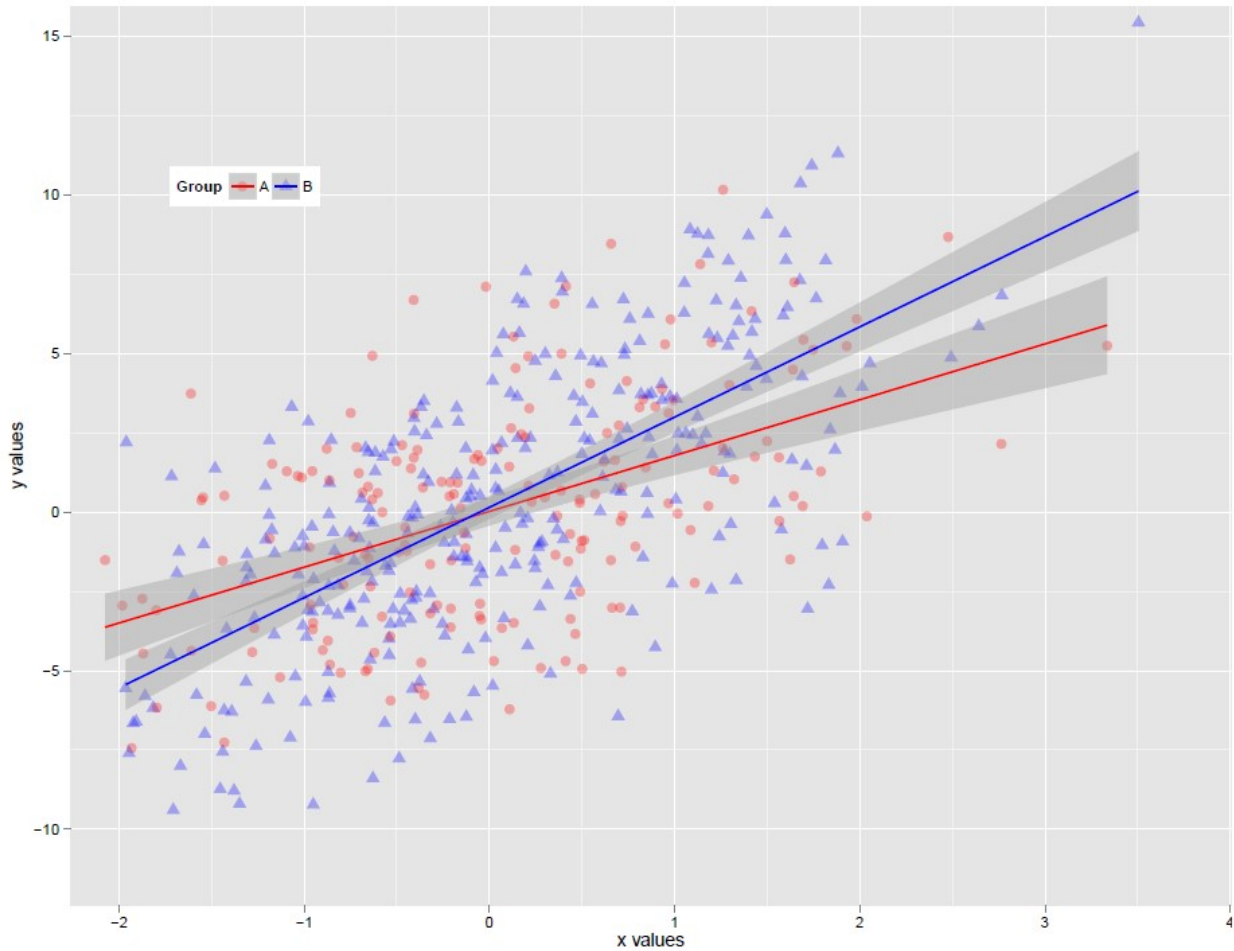


Figure 8: The main scatterplot

Density plot for x-values This chunk creates densities for the x-values, suppressing labels.

```
density.plot.x <- ggplot(data=working.df, aes(x=x, fill=group, shape=group)) +
  geom_density(alpha=.4) +
  scale_fill_manual(values=c("red", "blue"))+
  scale_x_continuous(limits=c(xmin+.5,xmax+.5), expand=c(0,0))+
  theme(axis.text=element_text(color="white"),
        axis.ticks=element_line(color="white")) +
  labs(x=NULL, y="", title="\n") +
  guides(fill=FALSE)
print(density.plot.x)
```

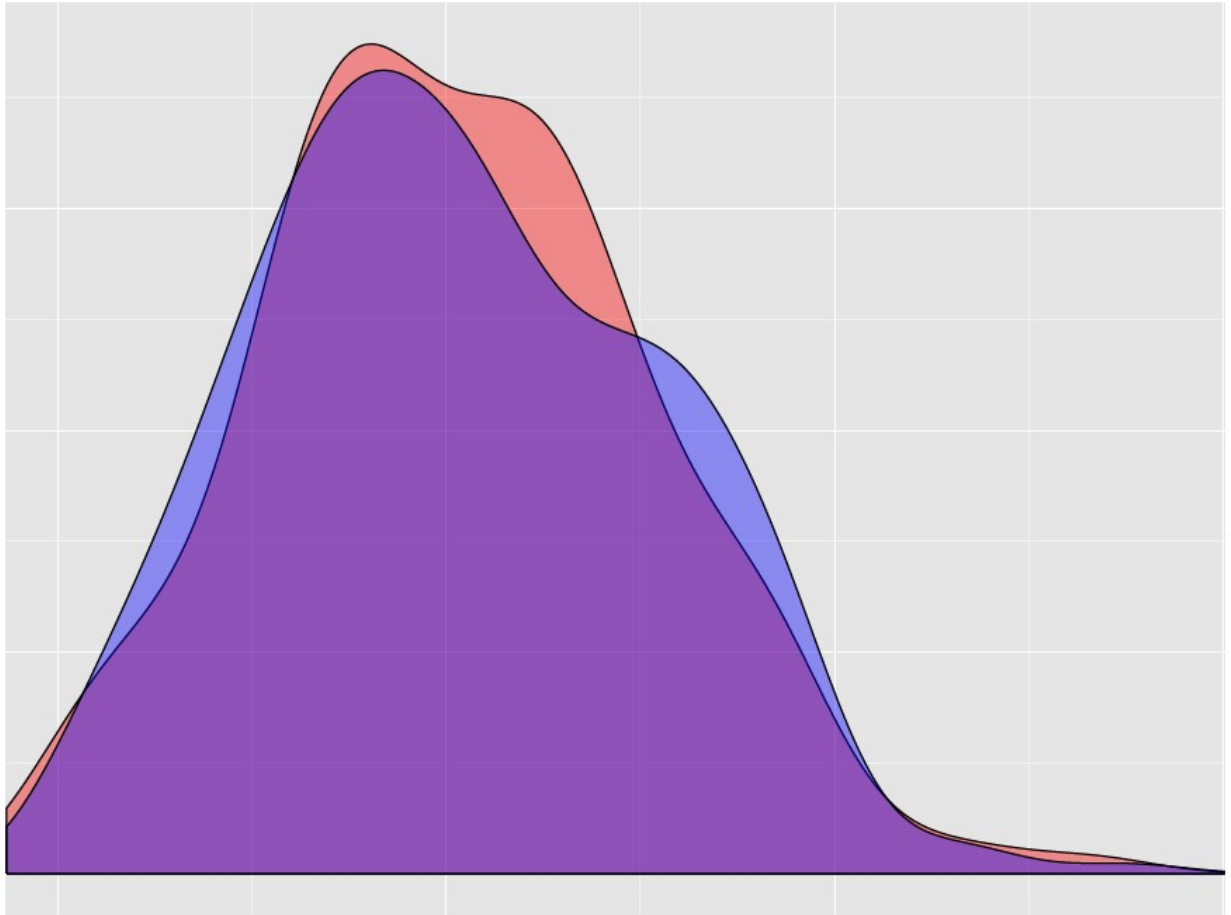


Figure 9: Density plot of x-values

Rotated density plot for y-values: This chunk creates densities for the y-values, suppressing labels and rotating the graphic.

```
density.plot.y <- ggplot(data=working.df, aes(x=y, fill=group, shape=group)) +  
  geom_density(alpha=.4) +  
  scale_fill_manual(values=c("red", "blue"))+  
  
  theme(axis.text=element_text(color="white"),  
        axis.ticks=element_line(color="white")) +  
  scale_x_continuous(limits=c(ymin+.5,ymax+.5), expand=c(0,0))+  
  labs(x=NULL, y="") +  
  guides(fill=FALSE) +  
  coord_flip()  
  
print(density.plot.y)
```

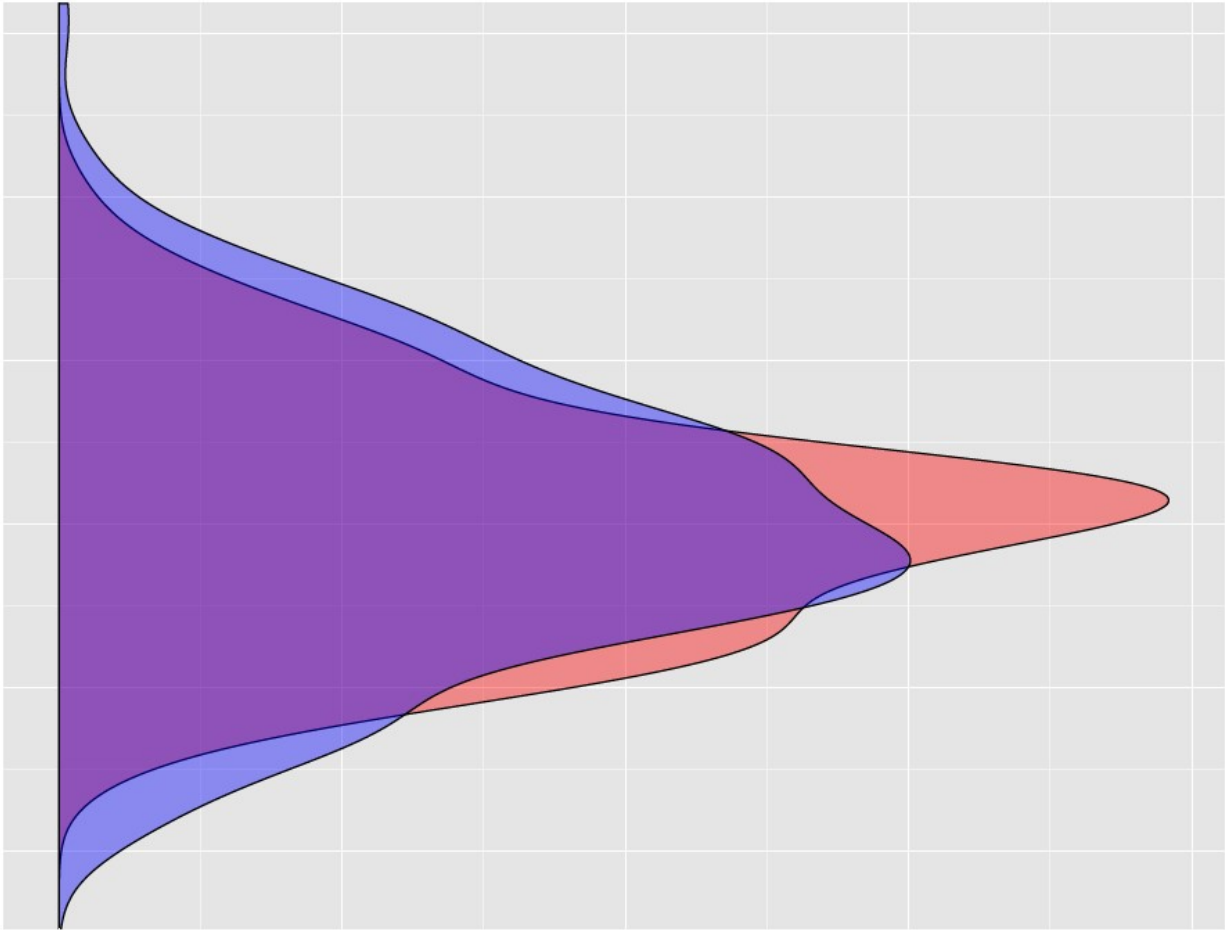



Figure 10: Density plot of y-values

Note that object blankPanel is an object placed in the global environment by figures2::default.settings.

```
# blankPanel <- grid.rect(gp=gpar(col="white"), draw=FALSE) # created by default.settings
build.page(interior.h = c(.35, .65),
interior.w = c(.75, .25),
ncol=2,
nrow=2,
interior=list(
density.plot.x, blankPanel,
main.plot, density.plot.y))

annotate.page(override = "", title=list("Title Line 1", "", "", "", ""))
```

Upper Left Header 1
Upper Left Header 2
Upper Left Header 3

Figure 1.100
Title Line 1

Upper Right Header 1
Upper Right Header 2
Upper Right Header 3

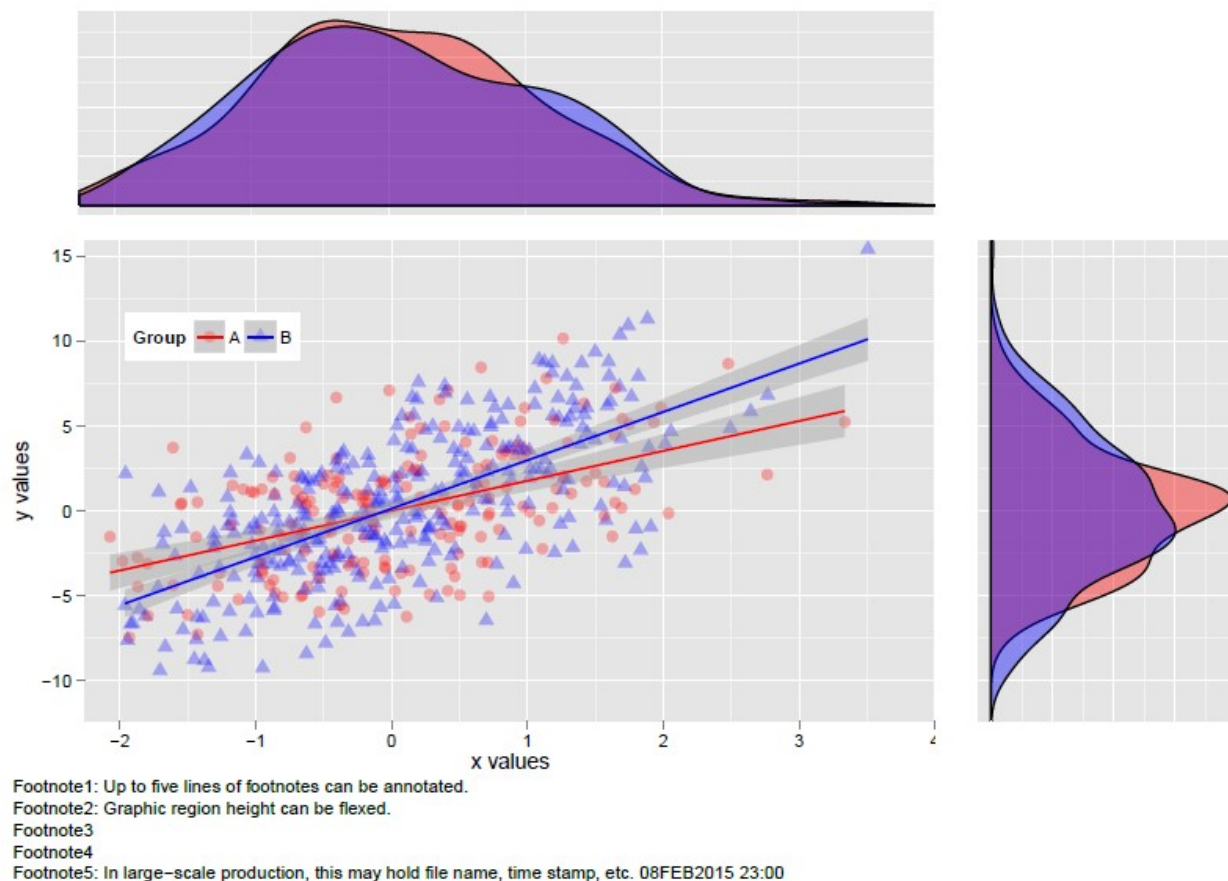
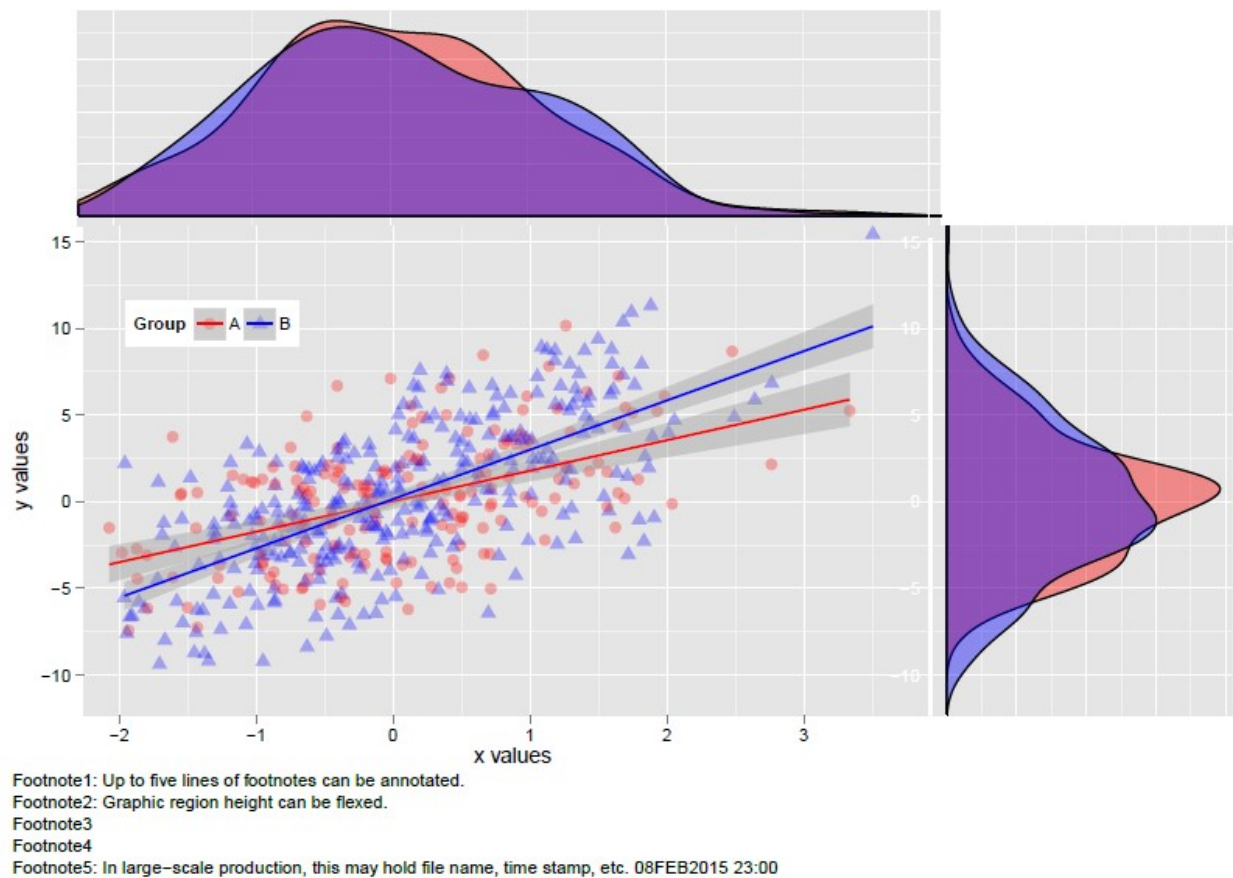


Figure 11: The final assembled figure

Note: There is still white space between the main plot and the density plots. This can be reduced by manipulating the plot.margins and using negative values. The values chosen were obtained via trial and error.

```
build.page(interior.h = c(.35, .65),
interior.w = c(.75, .25),
ncol=2,
nrow=2,
interior=list(
density.plot.x+
theme(plot.margin= unit(c(0, -.1, -.1, 0), unit="in")),
blankPanel,
main.plot+theme(plot.margin=unit(c(-.1, -.1, 0, 0), unit="in")),
density.plot.y + theme(plot.margin=unit(c(-.1, 0, 0, -.3),
unit="in"))))
```

Figure 1.100
Title Line 1



Concluding remarks

The interested reader may refer to `figuRes2` package (<https://CRAN.R-project.org/package=figuRes2>) for additional details and examples for employee the functions of this R Package.

CONTACT INFORMATION (HEADER 1)

(In case a reader wants to get in touch with you, please put your contact information at the end of the paper.)

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Greg Cicconetti

Company: Takeda

Email: greg.cicconetti@takeda.com

Brand and product names are trademarks of their respective companies.