MWSUG 2018 - Paper HS-143

PDF.PLOT.OUT: an R Function to Output Vector Graphics

Hillary Graham, Eli Lilly, Indianapolis, IN
Michelle Carlsen, Eli Lilly, Indianapolis, IN
Zeqing Lu, Eli Lilly, Indianapolis, IN

ABSTRACT

Typically pharmaceutical statistical analysts output graphics in RTF format. However, RTF files are difficult to alter for publication or submission because they are not an editable file type. On the other hand, vector-based files enable users to modify colors, sizes, labels, etc. in Adobe Illustrator before disclosure. Currently, R has the capability to create vector graphics in PDF format. However, the process of outputting graphics in PDF format from R requires several steps. This can be irritating for experienced users, and discouraging for new users. In order to make the process easier, we have generated R code to output graphics into a PDF. This code automates the creation of vector-based graphics in R, making it more efficient for both new and experienced R-users to prepare plots ready for disclosure.

INTRODUCTION

A large portion of a pharmaceutical statistical analyst's job is to generate graphics in order to address the many questions that arise when conducting clinical trials. Whatever software used to create the graphic (SAS®, R, etc.), it is common to save the results into a raster image file (like an RTF) to later be utilized for study team review and disclosures. However, raster files have some downsides that make them inconvenient to use for many publication outlets.

Therefore, vector based graphics may address many of the concerns that arise with using raster files. Vector images are high-resolution, easily editable, and convenient for downstream customers of the data. Vector based figures can be stored in PDF, EPS, or SVG format.

This paper will address the differences between vector-based files raster images, in addition to providing an efficient solution for saving R-generated figures in a vector based PDF.

VECTOR VS. RASTER IMAGES

A raster graphic is composed of pixels, small blocks of color, that make up a larger image. This contrasts with a vector graphic, which is made up of a series of either straight or curved lines.

The difference in how an image is saved leads to several advantages to using vector based images. First, when a viewer makes a raster image larger the picture will get blurry or "blocky" while a vector image will not lose quality no matter how far the viewer zooms in. This makes vector graphics high-resolution no matter how the resulting image is rescaled for publication or disclosure. Second, a vector image is easily editable with software designed to read them, such as Adobe Illustrator. Each component line of a vector image is re-sizeable, re-scalable, re-colorable, etc. Additionally, text may be changed in ways most users are familiar with: style, size, font, etc. Finally, and most importantly, using vector based images increases efficiency for both the analyst and the writer who is submitting the image for publication.

In the past, a writer tasked with preparing a graphic for publication might use software like SigmaPlot in order to recreate the appropriate image suited for the publication outlet. This process was time consuming and involved many steps: first, an analyst would create the image in a raster file and output a dataset containing the information used to generate the plot. Then a specialist, usually a writer, would need to use the dataset to recreate the plot so that it meets the publication requirements. Occasionally, a change could not be made and the writer would have to come back to the analyst and request further updates to the original plot and data. Having multiple functions regenerating the same figures is not efficient and leaves room for human error.

On the other hand, a vector image's components are inherently editable. Thus, the process is reduced to having the analyst generate the figure and then allowing the writer to edit the image's properties given that the integrity of the analysis is preserved. For example, a vector image saved as a PDF is editable

using the Adobe Acrobat or Adobe Illustrator software. Each component of the vector image can be selected and altered in the desired way as shown in Figure 1. An example of an analysis plot before and after editing are shown in Figure 2 and Figure 3.

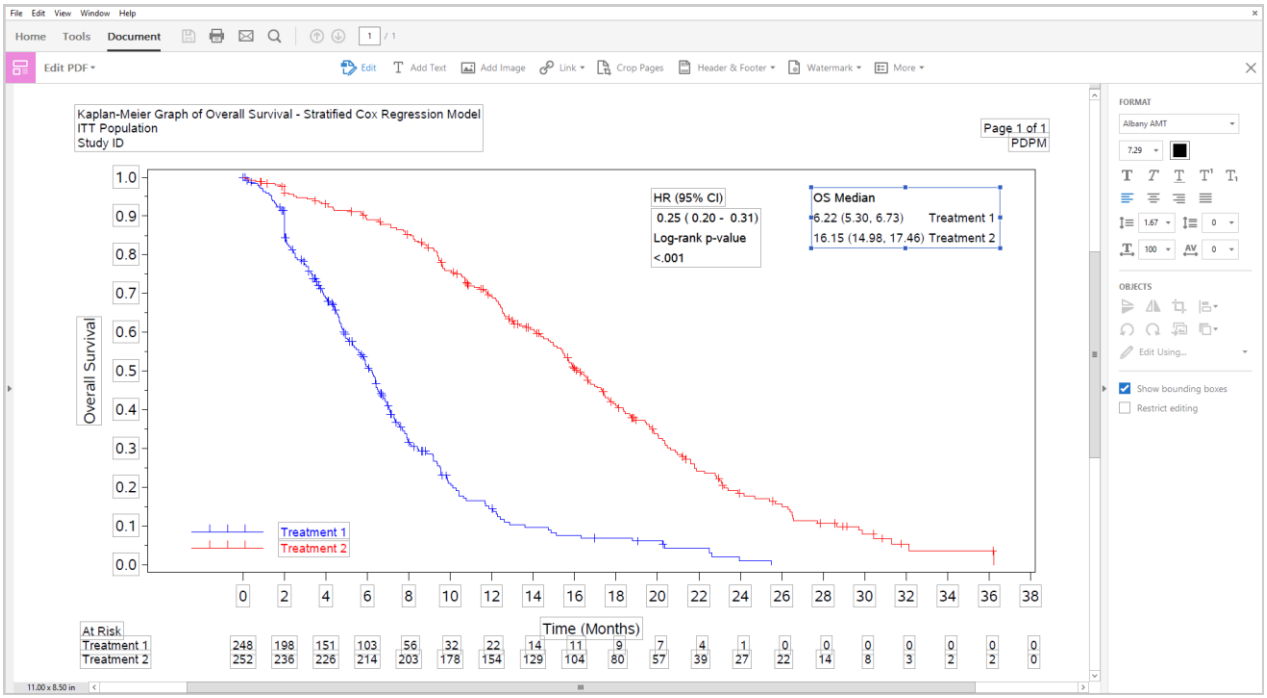

Figure 1. Editing a Vector Graphic

For the analyst, utilizing a vector-based image reduces the number of requests for small changes that different publications require to incorporate into the figure program, eliminates the need to generate a dataset containing the analysis, and reduces the risk for human error in recreating a figure in SigmaPlot.

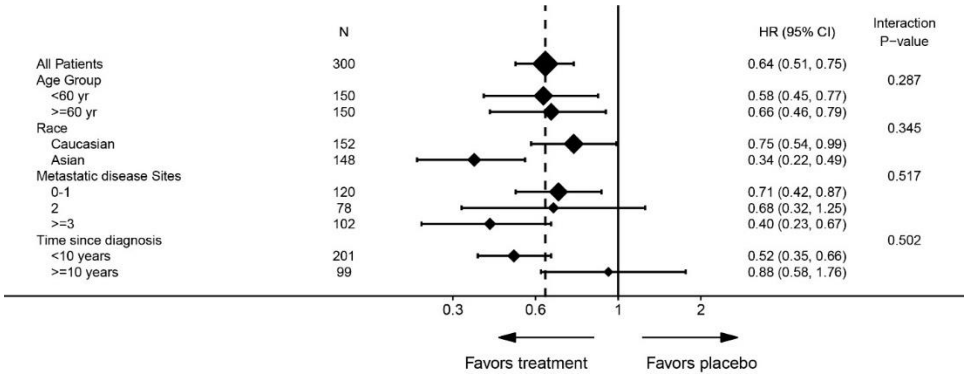

Figure 2. Vector Graphic before Editing

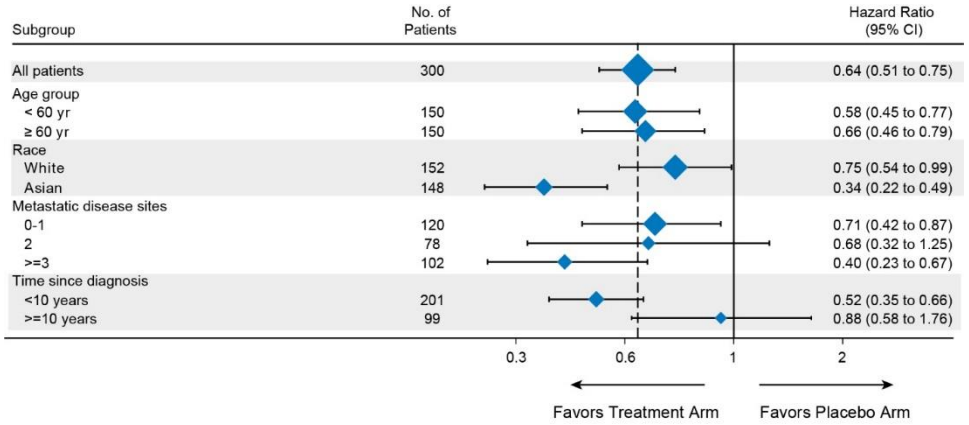

Figure 3. Vector Graphic after Editing

INTRODUCTION TO PDF.PLOT.OUT FUNCTION

When generating figures in R, outputting a PDF vector file with the customary annotations (such as titles and/or footnotes) can be daunting, particularly for a new user. This paper presents a solution for outputting a figure into a PDF with one R command.

Readers can find the pdf.plot.out function as well as component functions in Appendix 1 and Appendix 2. The proposed solution, pdf.plot.out, is a function that takes several arguments for customization of a PDF vector image file:

Required arguments:

- outpath – Path to save the resulting PDF output
- outname – A value to name the resulting PDF file; including the ".pdf" is optional
- plt – Object where plot is stored

Optional arguments:

- pgheight – Height PDF file in inches; Default is standard landscape size: 8.5 inches
- pgwidth – Width PDF file in inches; Default is standard landscape size: 11 inches
- pltheight – Height of analysis image in inches
- pltwidth – Width of analysis image in inches

  Note: When specifying pltheight and pltwidth, values must be smaller than the available page size after margins. Error will be given if plot is too large. Warning will be given if plot overlaps title and footnotes. If neither pltheight or pltwidth is given, the largest possible plot size will be calculated and used. If only one is specified, both will be treated as null.

- titles – Title of PDF; Defaults to null; Can be entered either as a character vector or a single string
- ftnts – Footnotes of PDF; Defaults to null; Can be entered either as a character vector or a single string
- bold.title – TRUE or FALSE value to indicate whether titles should be printed bold; Defaults to TRUE
- mar.size – Margin size in inches; Defaults to 0.75 inches

To further clarify the use of the pdf.plot.out function, two examples are given below. One illustrates using the output function with a figure generated with the ggplot2 package, while the second illustrates use

when outputting a base plot object. There are nuances to using the function in either case that will be made clear through these examples.

EXAMPLE 1: PDF.PLOT.OUT WITH GGPLOT2 FIGURE

To illustrate outputting a ggplot2 object into a vector PDF, we will use the built-in "mpg" dataset in R:

```
> head(mpg)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl   class
         <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p compact
5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p compact
6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p compact
```

A simple plot is coded below for this example:

```
> library(ggplot2)
> plot1 <- ggplot(mpg, aes(displ, hwy, colour = class)) + geom_point()
```

Figure 4. GGPLOT Example Figure

Now that the plot is saved as the "plot1" object, we will implement pdf.plot.out and check the resulting file:

```
> pdf.plot.out(outpath = "C:/Users/MyPlotPath1/MyPlotPath2",
               outname = "mpg_plot1.pdf", plt = plot1,
               pgsize = c(8.5, 11), pltsize = c(5.3, 9.5),
               titles = c('My Plot Title', 'Using the mpg Data'),
               ftnts = c("My Datapath: C:/Users/MyDataPath1/",
                         "My Output Path: C:/Users/MyPlotPath1/MyPlotPath2",
                         "My Program Path: C:/Users/MyProgPath1/"))
```

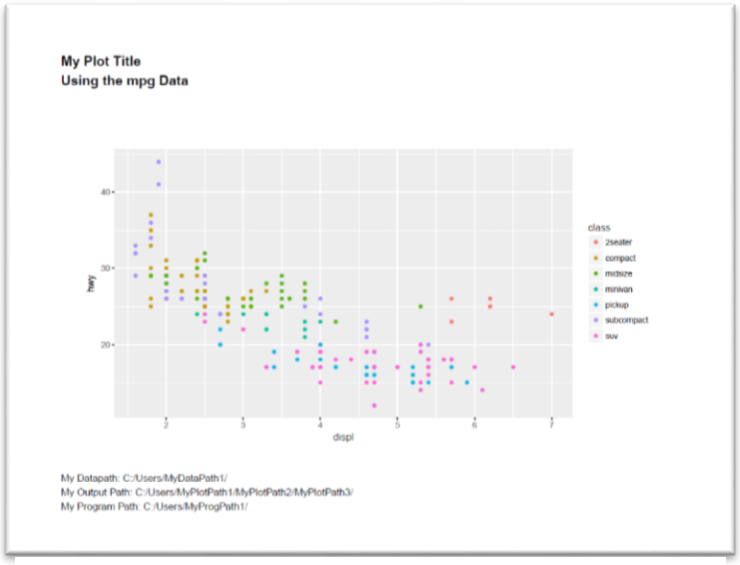

Figure 5. PDF Output with GGPLOT Example Figure

EXAMPLE 2: PDF.PLOT.OUT WITH BASE PLOT FIGURE

To illustrate outputting a base plot into a vector PDF, we will use the same built-in "mpg" dataset in R:

```
> head(mpg)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl   class
         <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p compact
5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p compact
6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p compact
```

A simple plot similar to the ggplot example is coded below:

```
> plot2 <- function(){plot(mpg$displ, mpg$hwy, col = as.factor(mpg$class), pch = 19)}
```

Note: The programmer will need to encase the base plot code in an empty function and save the resulting function object as shown. This is due to the fact that R runs base plot code immediately and will not save it to object name like a ggplot object.

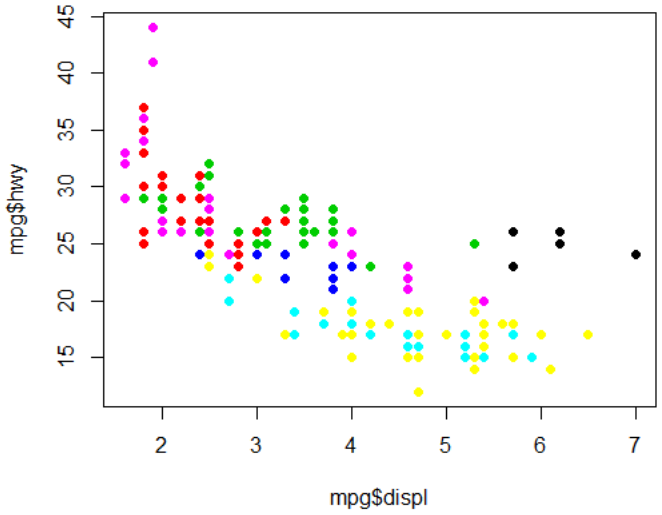

Figure 6. Base Plot Example Figure

Now that the plot is saved as the “plot2” function object, we will implement pdf.plot.out and check the resulting file:

```
> pdf.plot.out(outpath = "C:/Users/MyPlotPath1/MyPlotPath2",
               outname = "mpg_plot2.pdf", plt = “plot2”,
               pgsize = c(8.5, 11), pltsize = c(5.3, 9.5),
               titles = c('My Plot Title', 'Using the mpg Data'),
               ftnts = c("My Datapath: C:/Users/MyDataPath1/",
                         "My Output Path: C:/Users/MyPlotPath1/MyPlotPath2",
                         "My Program Path: C:/Users/MyProgPath1/"))
```

Note: To use pdf.plot.out with a base plot, supply the plt argument with the function name that the plot is stored in as a string (ie. “plot2” in this example).

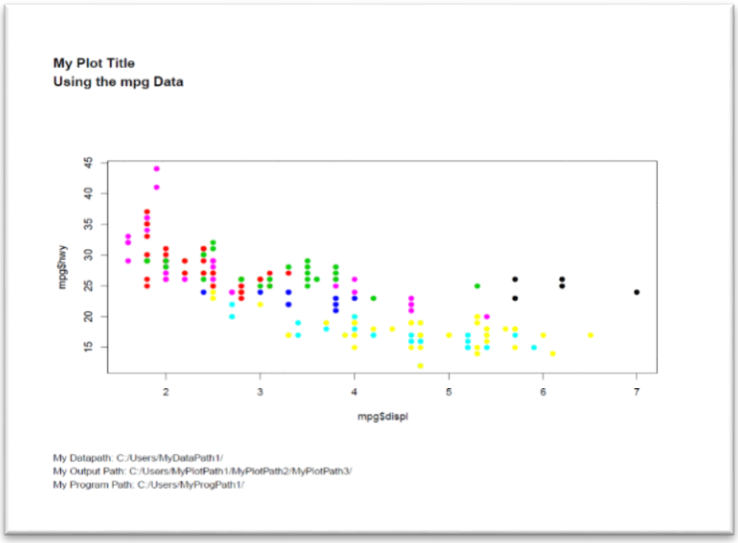

Figure 7. PDF Output of Base Plot Example Figure

CONCLUSION

Vector graphics can increase efficiency for analysts working in the health sciences. This paper presents a solution for exporting vector images into PDF files using R. The pdf.plot.out function automates the plot, title, and footnote placement for figures generated in R, reducing the amount of code an R user needs and making figure output simpler for new R users.

In the future, this project could be expanded to incorporate other R plots, such as ‘boxplot’, ‘barplot’, and ‘ggsurv’, which are specialized plots available in R. In addition, there is the possibility to expand into docx vector files, which may be more useful for documentation purposes.

CONTACT INFORMATION

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

Hillary Graham
Eli Lilly
317-276-4055
graham_hillary_t@lilly.com

Michelle Carlsen
Eli Lilly
317-276-9771
carlsen_michelle@lilly.com

Zeqing Lu
Eli Lilly
317-433-2136
lu_zeqing@lilly.com

APPENDIX 1 PDF.PLOT.OUT OUTPUT FUNCTION

```
pdf.plot.out <- function(outpath, outname, plt,
                         pgsize = c(8.5, 11), pltsize = NULL,
                         titles = NULL, ftnts = NULL,
                         bold.title = TRUE,
                         mar.size = .75) {

  require(RGraphics)
  require(gridGraphics)
  require(gridBase)
  require(ggplot2)

  title.size = 15
  ftnt.size = 11

  # Run Error Checks
  outpathchk(outpath)
  outnmchk(outname)
  if(!is.null(titles)) titchk(titles)
  if(!is.null(ftnts))  ftntschk(ftnts)
  #titfmtchk(title.size)
  #ftntfmtchk(ftnt.size)

  # Add .pdf to outname if missing
  outname <- check.name(outname)

  #Title/Footnote prep
  if (!is.null(titles)){
    titles2 <- paste0(titles, collapse = '\n')
    readytitles <- splitString(titles2, fontsize=title.size, availwidth = pgsize[2] - (2*mar.size), bold = bold.title)
    temptheight <- convertHeight(grobHeight(textGrob(readytitles, gp=gpar(fontsize=title.size, fontface =
ifelse(bold.title, 2, 1)))), "inches", valueOnly=TRUE)
  }
```

```
  if (!is.null(ftnts)){
    ftnts2 <- paste0(ftnts, collapse = '\n')
    readyfootnotes <- splitString(ftnts2, fontsize=ftnt.size, availwidth = pgsize[2] - (2*mar.size), bold =
FALSE)
    tempfheight <- convertHeight(grobHeight(textGrob(readyfootnotes, gp=gpar(fontsize=ftnt.size))),
"inches", valueOnly=TRUE)
  }

  # Set default pltsize if not supplied
  if(is.null(pltsize)){
    if(is.null(titles) & is.null(ftnts)) pltsize <- c(pgsize[1]-mar.size*2, pgsize[2]-mar.size*2)
    if(is.null(titles) & !is.null(ftnts))pltsize <- c(pgsize[1] - mar.size*2 - tempfheight, pgsize[2] - mar.size*2)
    if(!is.null(titles) & is.null(ftnts))pltsize <- c(pgsize[1] - mar.size*2 - temptheight, pgsize[2] - mar.size*2)
    if(!is.null(titles) & !is.null(ftnts))pltsize <- c(pgsize[1] - mar.size*2 - tempfheight - temptheight, pgsize[2] -
mar.size*2)
  }
  sizechk(pgsize, pltsize)
  pdf(file = file.path(outpath, outname), height = pgsize[1], width = pgsize[2], onefile=FALSE)

  grid.newpage()

  if(!is.character(plt)){
    #output plot
    vp = viewport(x = 0.5, y = 0.5,
                  width = unit(x = pltsize[2], units = "inches"),
                  height = unit(x = pltsize[1], units = "inches")
    )
    pushViewport(vp)
    print(plt + theme(plot.margin = unit(c(1.5,1,.5,1), "cm")), vp = vp)
  }

  # set viewport to full page
  pushViewport(viewport(x = 0.5, y = 0.5,
                        width = unit(x = pgsize[2], units = "inches"),
                        height = unit(x = pgsize[1], units = "inches")))

  if(is.character(plt)){
    # output base-plot
    par(omi = c((pgsize[1] - pltsize[1])/2, (pgsize[2] - pltsize[2])/2, (pgsize[1] - pltsize[1])/2, (pgsize[2] -
pltsize[2])/2), pty = 'm')
    grid.draw(eval(parse(text=paste0(plt, '()'))))
    upViewport(1)
  }

  # plot titles/footnotes if supplied
  if (!is.null(titles)) grid.text(readytitles, x = unit(mar.size, 'inches'), y = unit(pgsize[1] - mar.size, 'inches'),
gp = gpar(fontsize=title.size, fontface = ifelse(bold.title, 2, 1)), just = c('left', 'top'))
  if (!is.null(ftnts)) grid.text(readyfootnotes, x = unit(mar.size, 'inches'), y = unit(mar.size, 'inches'), gp =
gpar(fontsize=ftnt.size, fontface = 1), just = c('left', 'bottom'))

  # Generate warning if titles and/or footnotes are going to write over the plot
  if (!is.null(titles) & !is.null(ftnts)) warnpltsize(temptheight, tempfheight, pltsize, mar.size, pgsize) else
    if (!is.null(titles) & is.null(ftnts)) warnpltsize(temptheight, 0, pltsize, mar.size, pgsize) else
      if (is.null(titles) & !is.null(ftnts)) warnpltsize(0, tempfheight, pltsize, mar.size, pgsize)

  # close pdf
```

```
  try(dev.off(), silent = TRUE)
  try(dev.off(), silent = TRUE)
}
```

APPENDIX 2 PDF.PLOT.OUT COMPONENT FUNCTIONS

```
if (TRUE){
  splitString <- function(text, fontsize, availwidth, bold) {
    strings <- strsplit(text, "(?<=[/ ])", perl = TRUE)[[1]]
    newstring <- strings[1]
    linewidth <- grobWidth(textGrob(newstring, gp = gpar(fontsize = fontsize, fontface = ifelse(bold, 2, 1))))
    for (i in 2:length(strings)) {
      width <- grobWidth(textGrob(strings[i], gp = gpar(fontsize = fontsize, fontface = ifelse(bold, 2, 1))))
      if ((convertWidth(linewidth + width, "inches", valueOnly=TRUE) < availwidth) & !grepl('\n', strings[i])) {
        sep <- ""
        linewidth <- linewidth + width
      } else if((convertWidth(linewidth + width, "inches", valueOnly=TRUE) < availwidth) & grepl('\n',
strings[i])) {
        sep = ''
        linewidth <- grobWidth(textGrob(strsplit(strings[i], '\n', fixed = TRUE)[[1]][2], gp = gpar(fontsize =
fontsize, fontface = ifelse(bold, 2, 1))))
      } else {
        sep <- "\n"
        linewidth <- width
      }
      newstring <- paste(newstring, strings[i], sep=sep)
    }
    newstring
  }
  warnpltsize <- function(temptheight, tempfheight, pltsize, mar.size, pgsize) {
    suggestsize <- pgsize[1] - (temptheight + tempfheight + 2*mar.size)
    if(pltsize[1] > (pgsize[1] - (temptheight + tempfheight + 2*mar.size))) warning(paste0('Titles and
footnotes are writing over plot. Consider reducing plot height to ', round(suggestsize, 2) - .01, ' or reducing
the number of titles and/or footnotes.'), call. = FALSE)
  }

  # Error Checks
  sizechk <- function(pgsize, pltsize){
    if( pgsize[1]<pltsize[1] | pgsize[2]<pltsize[2] ) stop('Error: plot size exceeds page size')
  }
  outpathchk <- function(outpath){
    if( (is.vector(outpath) != TRUE | length(outpath) != 1 )) stop('Error: outpath format incorrect')
  }
  outnmchk <- function(outname){
    if(grepl('.pdf', outname) != TRUE) stop('Error: outname does not contain .pdf')
  }
  titchk <- function(titles){
    if(is.vector(titles) != TRUE | is.null(titles)==TRUE ) stop('Error: titles format incorrect')
  }
  ftntschk <- function(ftnts){
    if(is.vector(ftnts) != TRUE | is.null(ftnts)==TRUE ) stop('Error: footnotes format incorrect')
  }
  titfmtchk <- function(title.size){
    if(!is.numeric(title.size) ) stop('Error: titles format incorrect')
  }
  ftntfmtchk <- function(ftnt.size){
```

```
    if(!is.numeric(ftnt.size) ) stop('Error: footnotes format incorrect')
  }

  check.name <- function(outname) {
    substrRight <- function(outname, n){
      substr(outname, nchar(outname)-n+1, nchar(outname))
    }
    check4 <- substrRight(outname, 4)
    ifelse(check4==".pdf",outname,paste0(outname,".pdf"))
  }
}
```

APPENDIX 3 PDF.PLOT.OUT REQUIRED PACKAGES

```
install.packages("RGraphics", "gridGraphics", "gridBase", "ggplot2")

library(RGraphics, gridGraphics, gridBase, ggplot2)
```