1 Introduction

1.1 Background

The core graphics system in R can been divided in to two main packages. The first package is the graphics package. It is older and it provides the original GRZ graphics system from S, sometimes referred to as "traditional" graphics. It is relatively fast and many other R packages build on top of it. The newer package is the grid package. It is actually slower but is has more flexibility and additional features compared to the graphics package.

A graph that is drawn using grid can been edited in many more ways than a graph that has been drawn using the basic graphics package. However, there is a new package, called gridGrahics, which allows us to convert a plot that has been drawn by the graphics package to an equivalent plot drawn by grid graphics. This means that the additional flexibility and features of grid become available for any plot drawn using the graphics package.

1.2 The `gridGraphic` package

`gridGraphic` is like a 'translator' that translate the plot which been drawn by using the basic graphics package to the plot which been drawn by using grid package. The `gridGraphic` package has a main function called `grid.echo()`, which takes a recorded plot (or NULL for the current plot of the current graphic dervice) as an argument. Then it replicate the plot by using grid so that the user may edites the plot in more way than the plot drawn by bacis graphic package. The following code provides a quick example. We generate 25 random numbers for x and y. First, we draw a scatter plot using the function plot() from the basic graphic package, then we redraw it using grid.echo() from the gridGraphic package with grid.

```
setwd(110)
x = runif(25)
y = runif(25)
plot(x,y, pch = 16)
grid.echo()
```

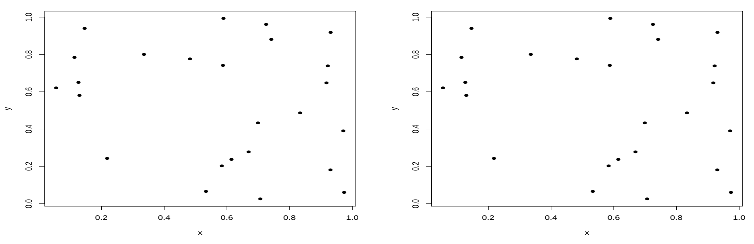


Figure 1: The left plot drawn by using plot(), the Right plot is redraw it by using grid.echo() on grid graphic system, overall, they are identical to each other

Alternatively, one example that shows the advantage of drawing the plot using grid rather than basic is that there is an object, called grid grobs, which recored a list of the details of each components of the plot that been drawn. The list of grobs can been seen by calling the function `grid.ls()`.

```
graphics-background
graphics-plot-1-points-1
graphics-plot-1-bottom-axis-line-1
graphics-plot-1-bottom-axis-ticks-1
graphics-plot-1-bottom-axis-labels-1
graphics-plot-1-left-axis-line-1
graphics-plot-1-left-axis-ticks-1
graphics-plot-1-left-axis-labels-1
graphics-plot-1-box-1
graphics-plot-1-xlab-1
graphics-plot-1-ylab-1
```

As we see, the `grid.ls()` returns a list of grid grobs of the pervious plot that been redrawn by grid. there is one element called ”graphics-plot-1-bottom-axis-labels-1” which is the element of the label of the bottom axis. There are several function on the `grid` package that used for mainpulate this grob. For example, if the user wants to rotate the labels of the bottom axis by 30 degrees and changes the color from default to orange, then the following code mainpulate this changes.

```
grid.edit("graphics-plot-1-bottom-axis-labels-1",
          rot=30, gp=gpar(col="orange"))
```

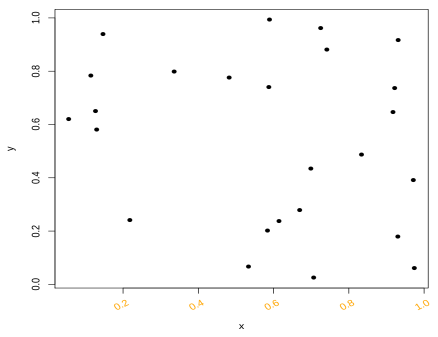


Figure 2: The angel and the color of the bottom axis of the previous plot been change by 30 degree and orange

1.3 The problem

The grid.echo() function can replicate most plots that are drawn by the graphics package. However, there are a few functions in the graphics package that grid.echo() cannot replicate. One such function is persp() which draws 3-dimemtional surfaces, the other one is the filled.contour(). This leads to the aim of this project. If we can draw a plot with persp() or filled.countour(), the result from calling grid.echo() is a blank screen

```
x <- y <- seq(-4*pi, 4*pi, len = 27)
r <- sqrt(outer(x^2, y^2, "+"))
filled.contour(cos(r^2)*exp(-r/(2*pi)), frame.plot = FALSE, plot.axes = {})
```

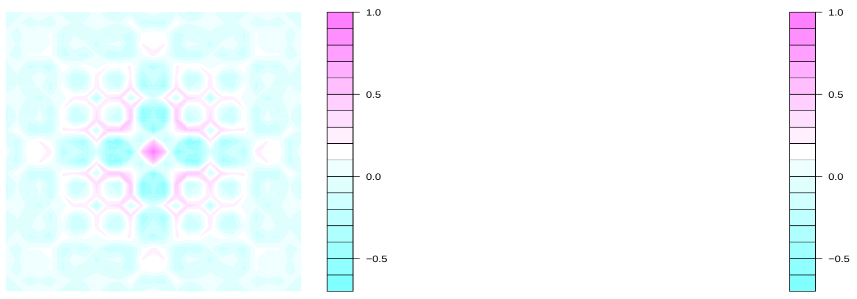


Figure 3: The left plot been drawn by using `filled.contour` and the right plot been redrawn by calling grid.echo(). There is a "blank" page on the right plot because the grid.echo cannot emulate filled.contour()

1.4 Aim of this project

The functions persp() and filled.contour() are wrote by C. However, it is very hard to debug and track the C code. One possible solution will be: 1. read and understand the C code, do the direct translation from C code to R code. 2. ....

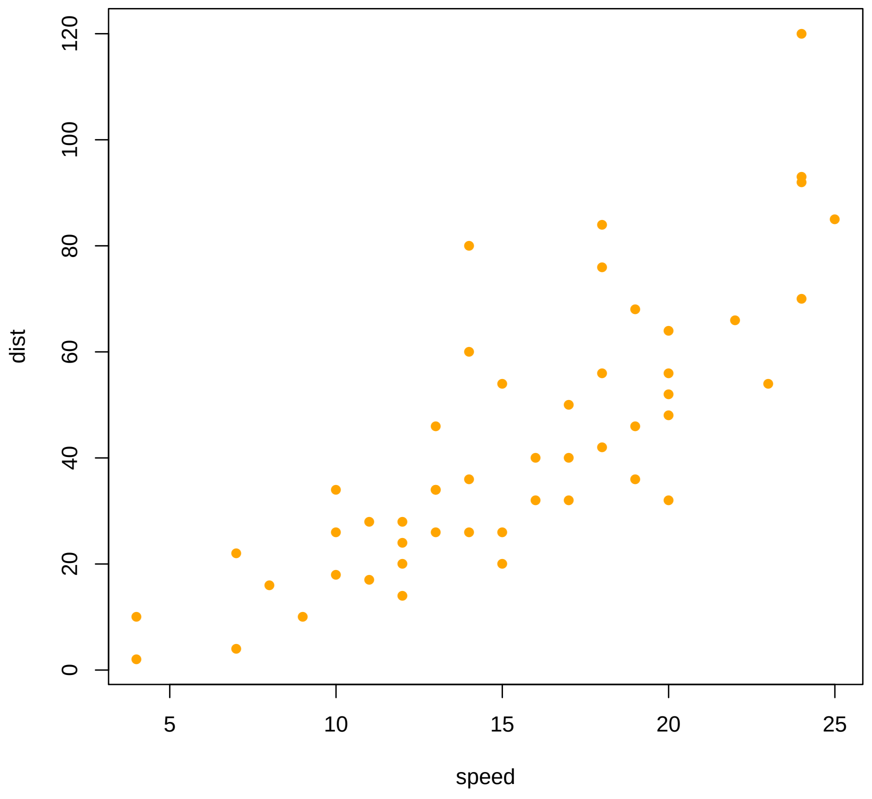
NOTE to Jason: explain how gridGraphics works first: graphics display list; gridGraphics implements an R version of each low-level C function on the display list (e.g., for C_plot_xy there is an R function called C_plot_xy in the gridGraphics package). THEN maybe write about 3D to 2D transformations, but only maybe.

2 The graphics engine display list

The information of evey plot drawn by R can be recorded. For example, In the simple `plot()` function, it is possible to obtain the parameters for x and y, even the label of the x-axis and y-axis. This information is called the graphics engine display list. This is the list that contain the informations so that user can access and use for further work. In this paper, we use this graphics engine display list for replicate the persp() plot and contour() plot by using grid.

To access the graphics engine display list, the `recordPlot()` function been used. This is the function for saving the plot in an R object.

```r
plot(cars$speed, cars$dist, col = 'orange',
      pch = 16, xlab = 'speed', ylab = 'dist')
```



```r
reco = recordPlot()
## Displays the inputs
reco[[1]][[4]][[2]][[2]]

## $x
##  [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14
## [24] 15 15 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24
## [47] 24 24 24 25
##
## $y
##  [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34
## [18]  34  46  26  36  60  80  20  26  54  32  40  32  40  50  42  56  76
## [35]  84  36  46  68  32  48  52  56  64  66  54  70  92  93 120  85
##
## $xlab
## [1] "cars$speed"
##
```

```
## $ylab
## [1] "cars$dist"
```

This example shows that: suppose we have a data set called `cars`, which contain two columns, the speed of the cars and the distance of travel. We have a plot which plotted the speed againist to the distance of travel. the `recordPlot()` will save this plot as an R object. As result, we can access the information of this plot. For example, the x-coordinate and the y-coordinate, or the x-label and the y-label.

There are many way for solving this problem, one possible solution will be translate the C code to R code such that as simliar as possible. The reason for doing this direct translation because:

1. It is hard to debug and track the C code. 2. It is very simple to debug the R code. If the R code is almost identical to the C code, then we can debug the R code to ensure that the R code can also provide the same result.

2.1 standalone

The functions that provides the persp() are huge, hence this section will only described some key step for building the functions.