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()
```

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"))
```

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 = {})
```

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.

Firstable, it is necessary to understand some important theory behind the 3-dimentional plot, such as the transformation from 3-D to 2-D, the drawing order of each ploygons (will be explained later). ect. Then we try to redraw it by using the grid package and we can replicate this function on grid by using the grid.echo().

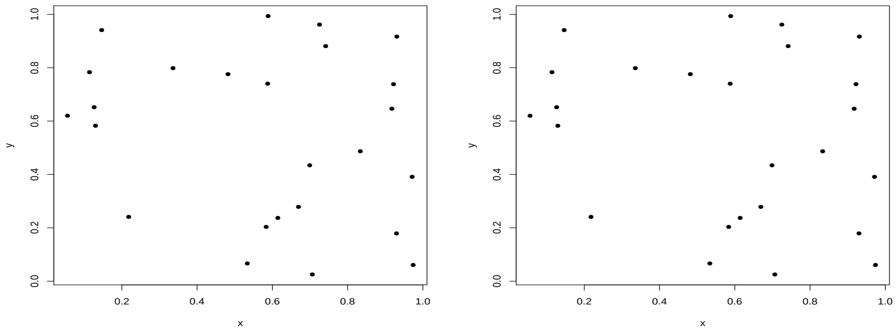


Figure 1: The left plot drawn by using plot(), the Right plot is redraw it by using grid.echo() on grid graphic system, overall, they are identical to each other

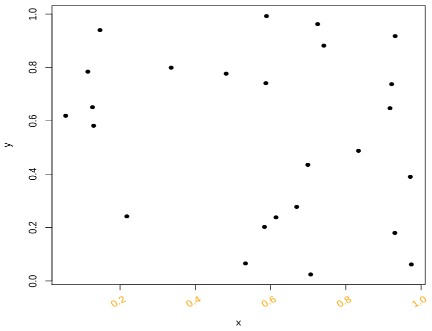


Figure 2: The angel and the color of the bottom axis of the previous plot been change by 30 degree and orange

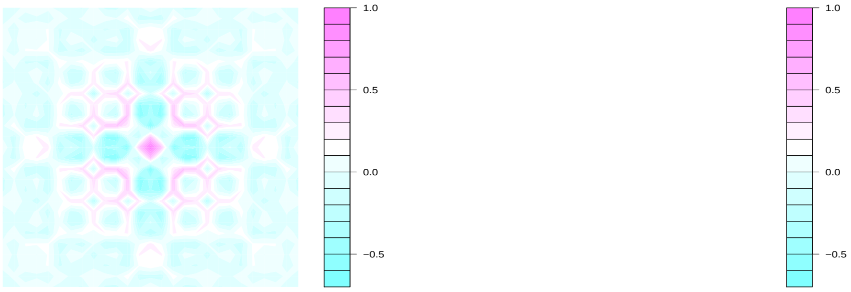
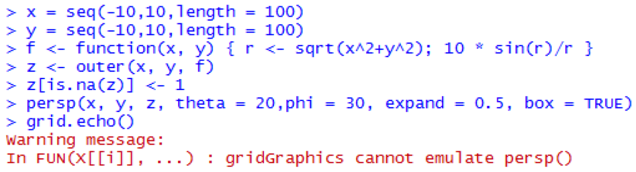


Figure 3: The angel and the color of the bottom axis of the previous plot been change by 30 degree and orange



```
> x = seq(-10,10,length = 100)
> y = seq(-10,10,length = 100)
> f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
> z <- outer(x, y, f)
> z[is.na(z)] <- 1
> persp(x, y, z, theta = 20,phi = 30, expand = 0.5, box = TRUE)
> grid.echo()
Warning message:
In FUN(X[[i]], ...) : gridGraphics cannot emulate persp()
```

Figure 4: an example shows that the grid.echo() cannot replicate the persp() on grid