

# Emulate the `persp()` plot and `filled.contour()` plot on **gridGraphics**

Zhijian Wen

Supervisor: Associate Professor Paul Murrell

University of Auckland

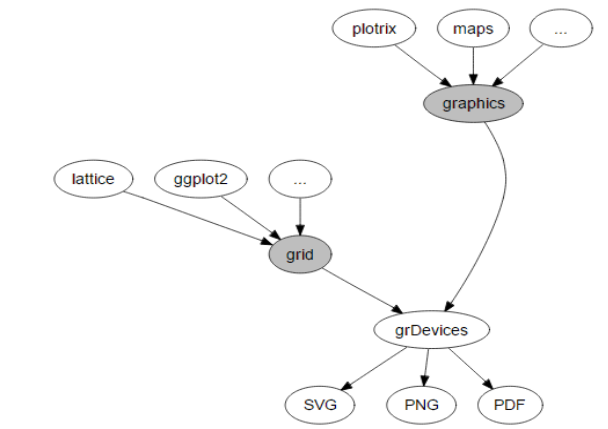
*jwen246@aucklanduni.ac.nz*

July 9, 2017

# Introduction

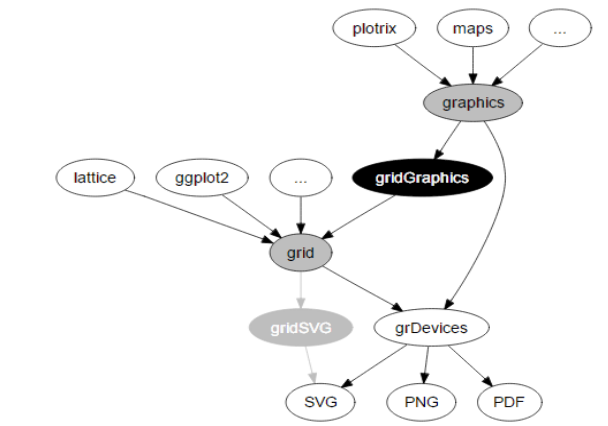
# Introduction

What is **graphics** and what is **grid** ?



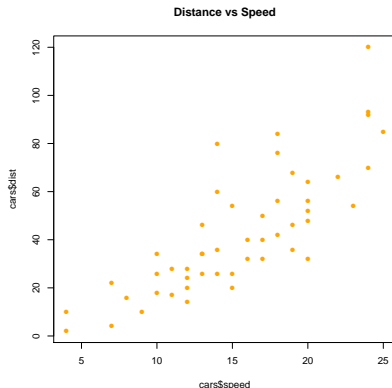
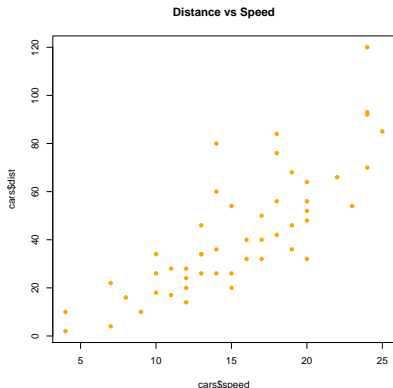
# Introduction

What is **gridGraphics**?



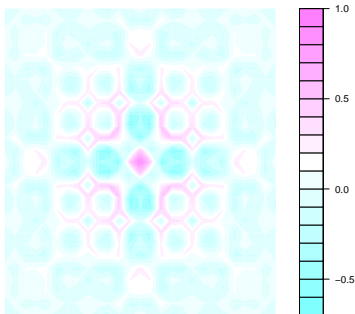
# Example

```
> plot(cars$dist ~ cars$speed, pch = 16,  
+       col = 'orange', main = 'Distance vs Speed')  
> library(gridGraphics)  
> grid.echo()
```



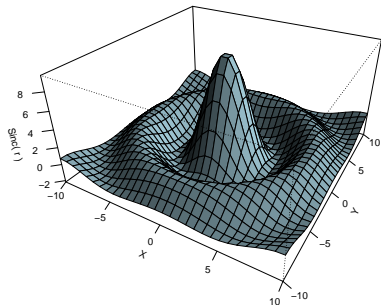
# The problem

```
> Persian_Rug_Art() ##filled.contour()  
> grid.echo()
```



# The problem

```
> Sinc_Curve() ##persp()  
> grid.echo()
```

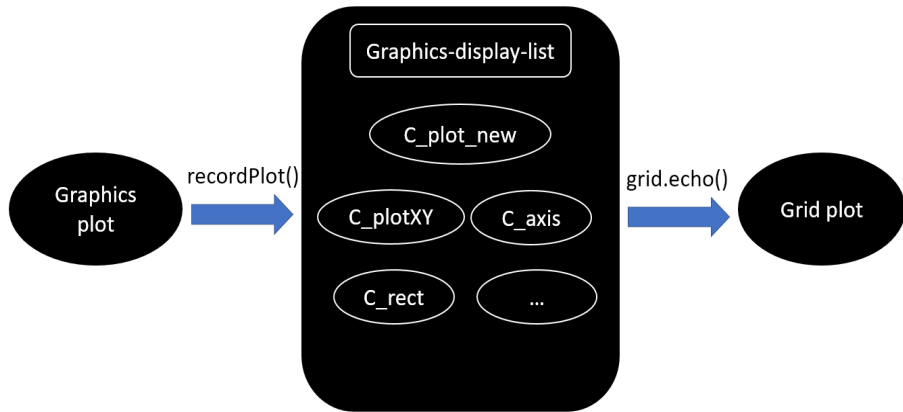


# The graphics engine display list



# The graphics engine display list

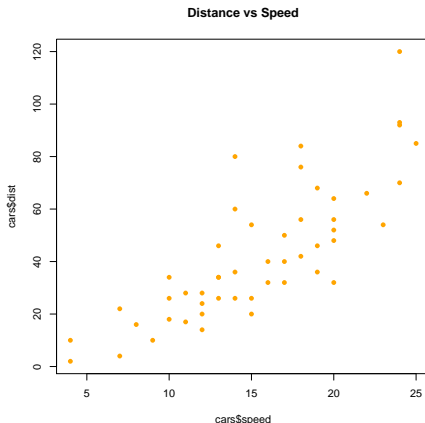
How does **gridGraphics** works?



# The graphics engine display list

```
x <- recordPlot()  
unlist(lapply(x[[1]], function(y) y[[2]][[1]]$name))
```

```
"C_plot_new"  
"palette2"  
"C_plot_window"  
"C_plotXY"  
"C_axis"  
"C_axis"  
"C_box"  
"C_title"
```



## The C code

```
SEXP C_plot_new(SEXP call, SEXP op, SEXP args, SEXP rho)
{
    ...
    dd = GNewPlot(GRecording(call, dd));
    ...
    GScale(0.0, 1.0, 1, dd);
    GScale(0.0, 1.0, 2, dd);
    GMapWin2Fig(dd);
    GSetState(1, dd);
    ...
}
```

# Structure of the C code

# Structure of the C code (pointers)

## The C code

```
static int LimitCheck(double *lim, double *c, double *s){
    ...
    *s = 0.5 * fabs(lim[1] - lim[0]);
    *c = 0.5 * (lim[1] + lim[0]);
    ...
}
LimitCheck(REAL(xlim), &xc, &xs)
```

## The R code

```
LimitCheck <- function(lim){
    ...
    s <- 0.5 * abs(lim[2] - lim[1])
    c <- 0.5 * (lim[2] + lim[1])
    c(s, c)
}
xs <- LimitCheck(xr)[1]; xc <- LimitCheck(xr)[2]
```

# How much C codes?

# Copy or not copy?

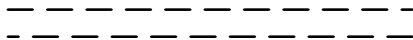
# “Copy” or not “copy”?

Why “copy”?

- 1 To make sure the **graphics**-plot is identical to the **grid**-plot (accuracy)

```
segments(x0 = 0, 0.5, x1 = 1, 0.5, lty = 1331, lwd = 5)
```

```
segments(x0 = 1, 0.5, x1 = 0, 0.5, lty = 1331, lwd = 5)
```



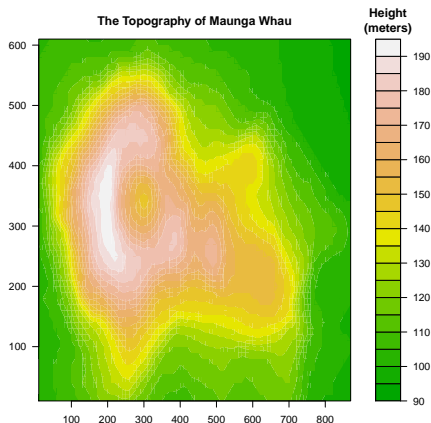
Why not just “copy”?

- 1 Speed (efficiency)



# Why not just “copy”?

```
volcano_filled.contour()  
xx <- recordPlot()  
info <- xx[[1]][[12]][[2]]  
  
dim(info[[4]])  
[1] 87 61  
  
length(info[[5]])  
[1] 22
```



There are at most  $(87 - 1) * (61 - 1) * (22 - 1) = 108360$  polygons.

# Why not just “copy”?

```
volcano_filled.contour()
```

```
## For loop
```

```
system.time(grid.echo())
```

```
# user system elapsed
```

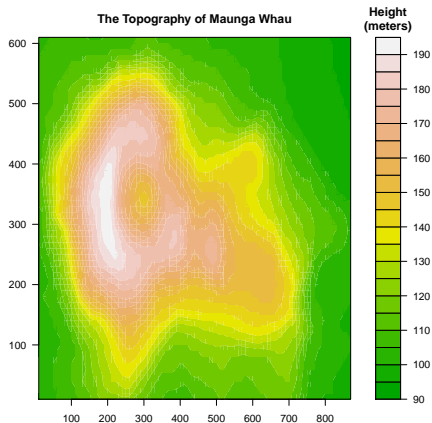
```
# 10.03 0.23 10.32
```

```
## vectorization
```

```
system.time(grid.echo())
```

```
# user system elapsed
```

```
# 1.28 0.53 1.82
```



# Testing

# Why doing the testing by using a software?

- To ensure the plot drawn by **graphics** is identical to the plot drawn by **grid**
- Using our eyes to check the identity will be wasting time and not reliable

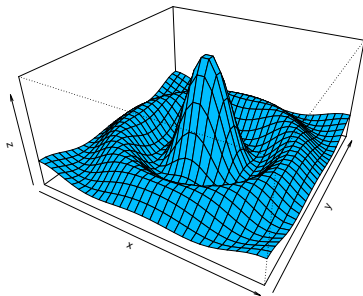
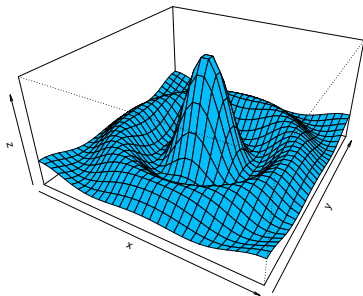
# Any difference?

```
## left plot
```

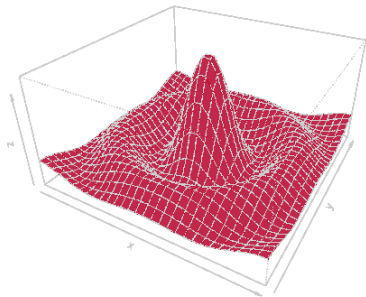
```
Sinc_Curve(col = rgb(red = 0, blue = 191, green = 255, ...))
```

```
## right plot
```

```
Sinc_Curve(col = rgb(red = 0, blue = 190, green = 255, ...))
```



```
cmd <- 'compare diff_1.pdf  
      diff_2.pdf  
      diff_out.pdf '  
system(cmd)
```

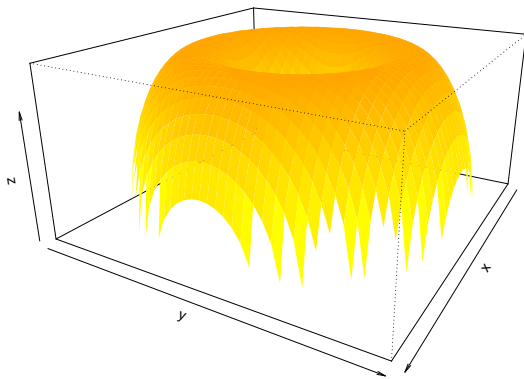


*\*Difference detected by using the software **ImageMagick***

# Final solution

# Final solution

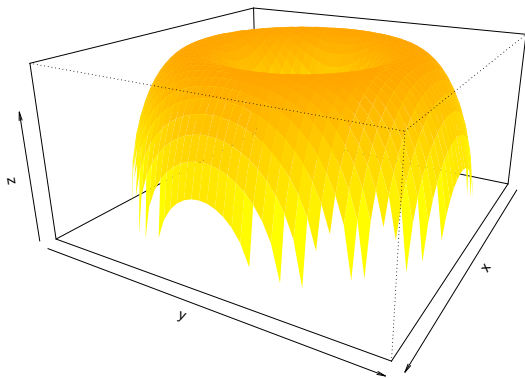
> Torus()





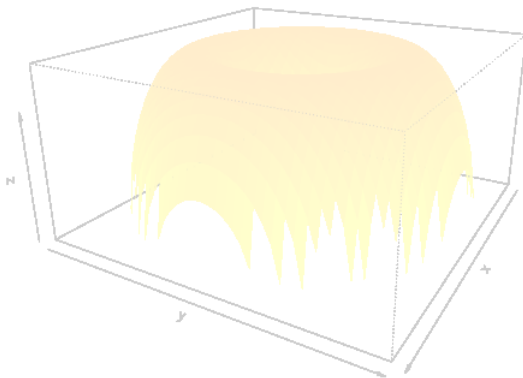
# Final solution

```
> grid.echo()
```



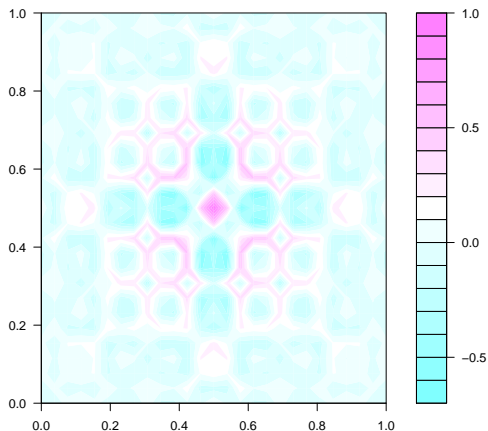
# Final solution

## Difference



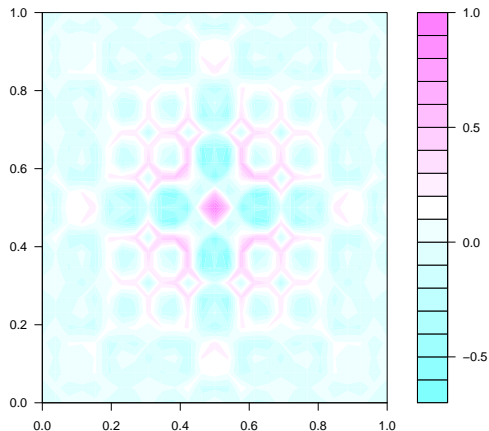
# Final solution

```
> filled.contour(cos(r^2) * exp(-r/(2 * pi)))
```



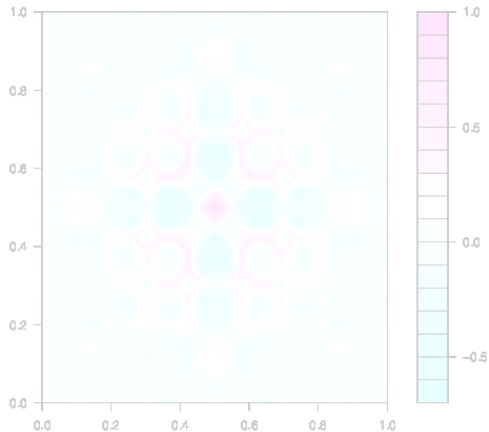
# Final solution

```
> grid.echo()
```



# Final solution

## Difference

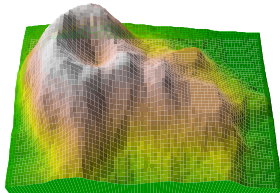


# Why use **grid**?

- **grid** is more flexible
- A complex plot cannot be produced by **graphics** but it might be produced by **grid**

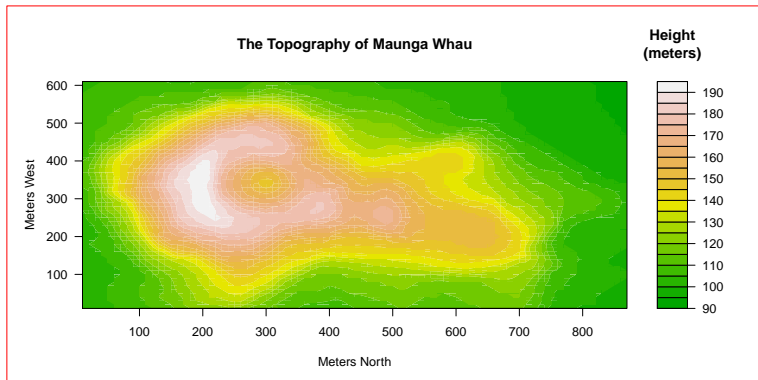
# Why use **grid**?

```
> par(mfrow = c(1,2))  
> Volcano.persp()  
## volcano_filled.contour()
```



# Why use **grid**?

```
> par(mfrow = c(1,2))  
> Volcano.persp()  
> volcano_filled.contour()
```

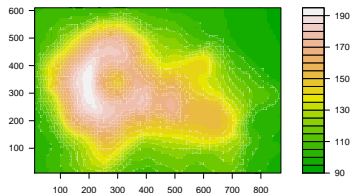
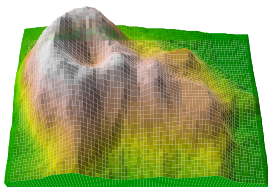




# Why use **grid**?

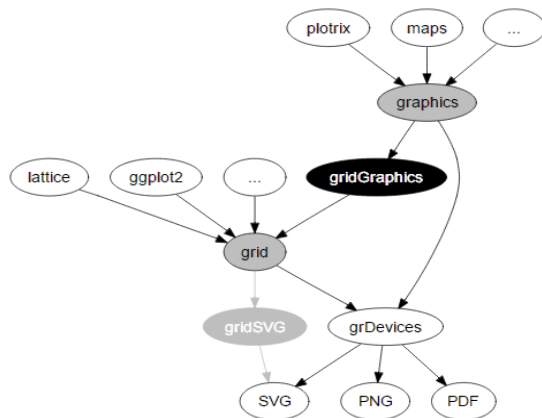
```
> vp <- viewport(...)  
> pushViewport(vp)  
> grid.echo(Volcano.persp, newpage=FALSE)  
> upViewport()
```

The Shape and Topography of Maunga Whau



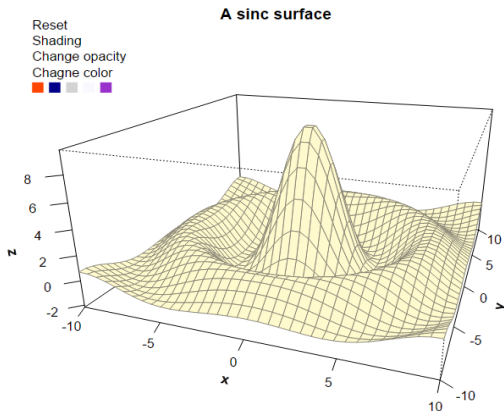
# Why use **grid**?

- A **grid**-plot can be export to SVG image by using the **gridSVG**
- The animation and interaction of this SVG image can produced easily.



# Why use **grid**?

```
> surface(); addFeatures()  
> library(gridSVG)  
> grid.script(file = "example.js")  
> grid.export("example.svg")
```



The **gridGraphics** now supports `persp()` and `filled.contour()`

*\*<http://cran.stat.auckland.ac.nz/web/packages/gridGraphics/index.html>*

Any Question(s)?