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

Zhijian Wen

University of Auckland

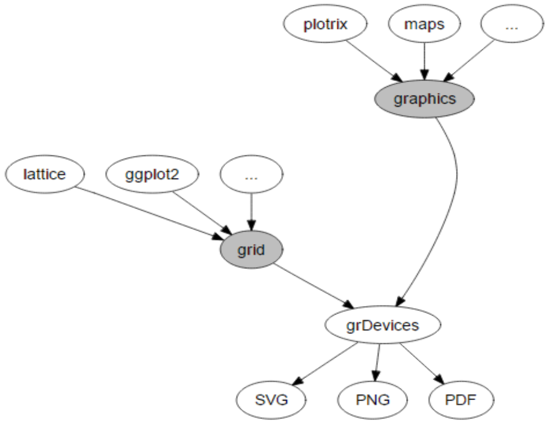
jwen246@aucklanduni.ac.nz

July 4, 2017

Introduction

Introduction

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



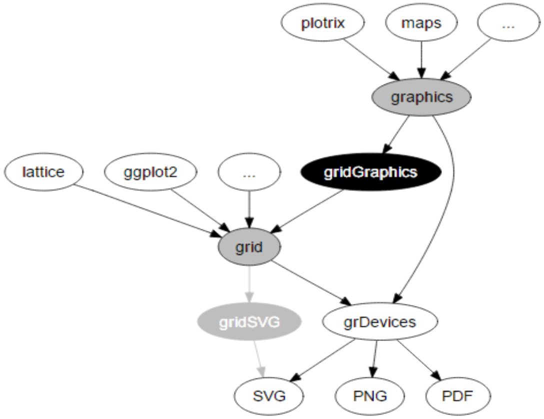
Introduction

Then, what is **gridGraphics**?

1. A **R** package
2. A "translator" that translates a **graphics**-plot to a **grid**-plot
3. With a main function `grid.echo()`.

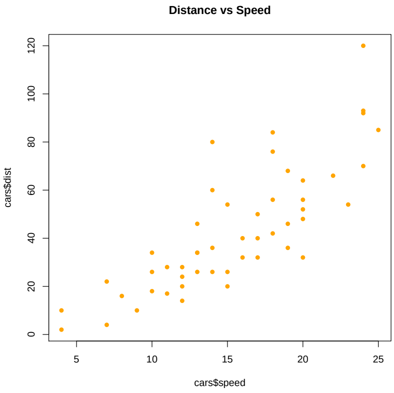
Introduction

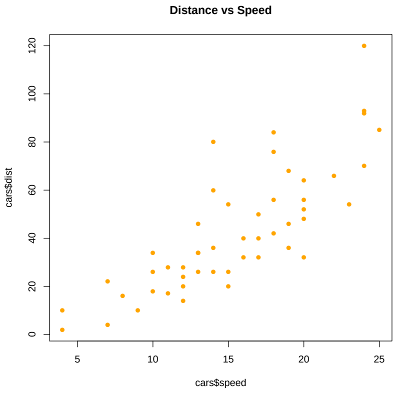
What is **gridGraphics**?



Example

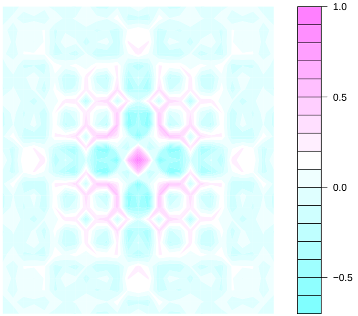
```
> plot(cars$dist ~ cars$speed, pch = 16,
+      col = 'orange', main = 'Distance vs Speed')
> library(gridGraphics)
> grid.echo()
```

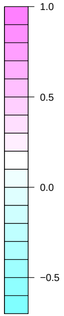




The problem

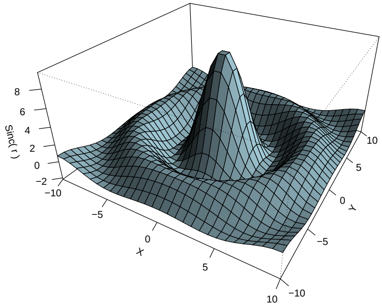
```
> Persian_Rug_Art() ##filled.contour()
> grid.echo()
```





The problem

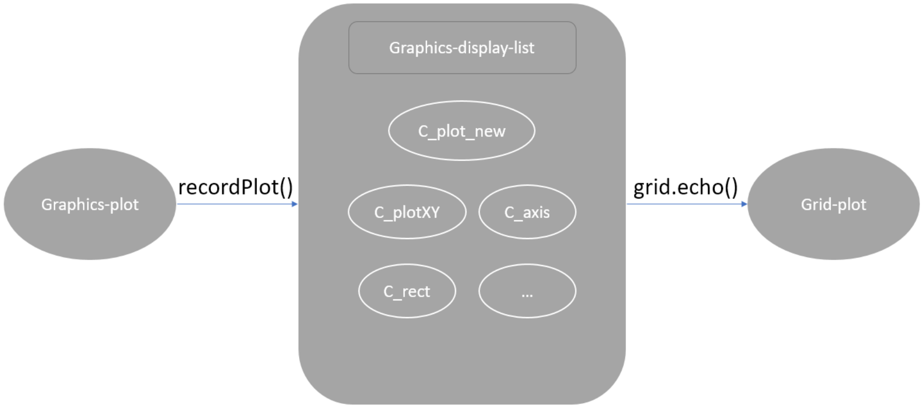
```
> Sinc_Curve() ##persp()
> grid.echo()
```



The graphics engine display list

The graphics engine display list

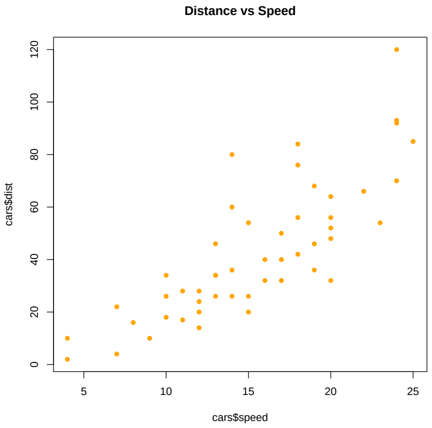
How does **gridGraphics** works?



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"
```



C_plot_new from **graphics**

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);
    ...
}
```

C_plot_new from **gridGraphics**

The **R** code

```
C_plot_new <- function(x) {
    ...
    if (page) {
        ...
        if (get("newpage", .gridGraphicsEnv))
            grid.newpage()
        ...
        pushViewport(viewport(name=vpname("root")))
        upViewport()
        setUpInner(par)
    } else {
        setUpFigure(par)
    }
}
```

Structure of the C code

Structure of the **C** code (pointers)

The problems

```
static int LimitCheck(double *lim, double *c, double *s)
{
    if(!R_FINITE(lim[0]) || !R_FINITE(lim[1]) ||
          lim[0] >= lim[1])
    return 0;
    *s = 0.5 * fabs(lim[1] - lim[0]) ;
    *c = 0.5 * (lim[1] + lim[0]) ;
    return 1;
}
...
if(!LimitCheck(REAL(xlim), &xc, &xs))
  error(_("invalid 'x' limits"));
```

Structure of the C code (pointers)

Solution

```
LimitCheck = function(lim){
    if(!is.finite(lim[1]) || !is.finite(lim[2])
            || lim[1] >= lim[2])
        stop("invalid limits");

    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?

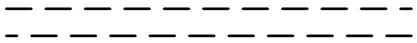
Copy or not copy?

Copy or not copy?

Why just 'copy'?

1. To make sure the **graphics**-plot is identical to the **grid**-plot

```
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)
```

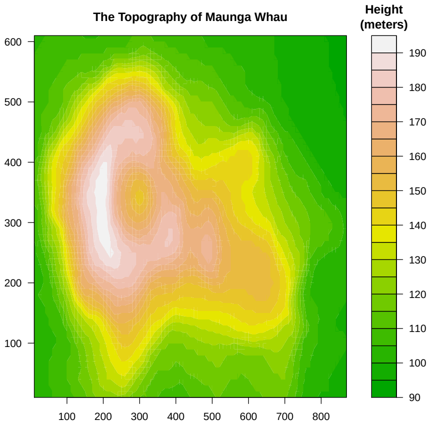


Why not just 'copy'?

1. Speed

Why not just 'copy'?

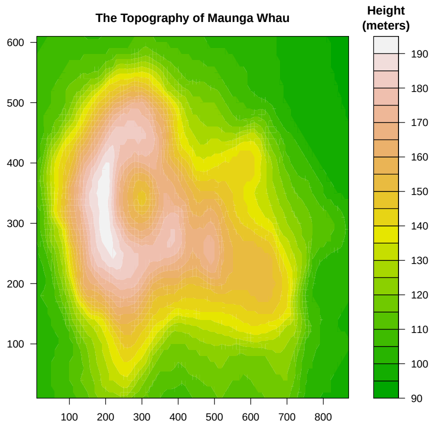
```
volcano_filled.contour()
xx = recordPlot()
info = xx[[1]][[12]][[2]]

dim(info[[4]])
[1] 87 61

length(info[[5]])
[1] 22
```



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

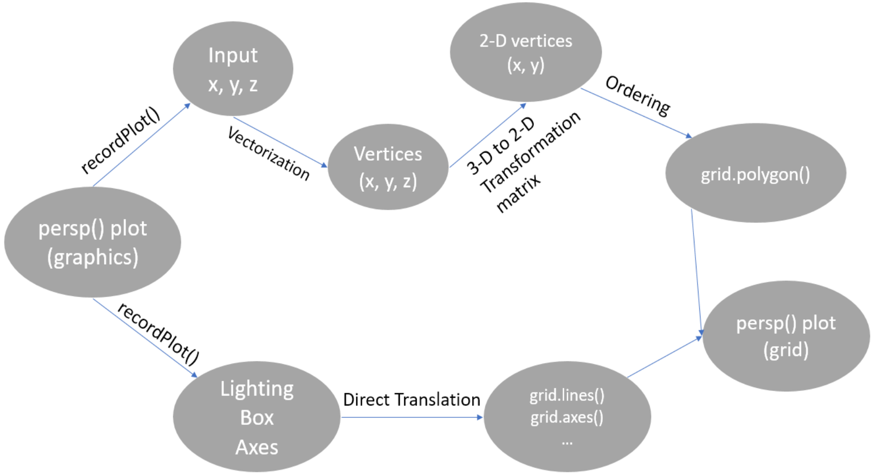
Why not just 'copy'?

```
volcano_filled.contour()


## For loop
system.time(grid.echo())

# user system elapsed
# 10.03 0.23 10.32


## vectorizetion
system.time(grid.echo())

# user system elapsed
# 1.28 0.53 1.82
```



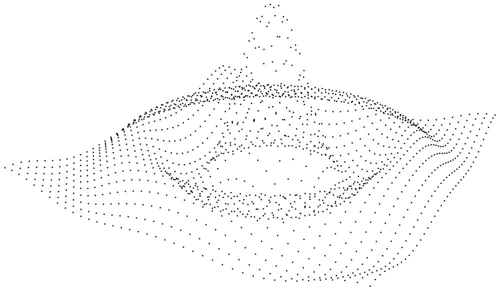
Solution to `persp()`

Solution to persp()



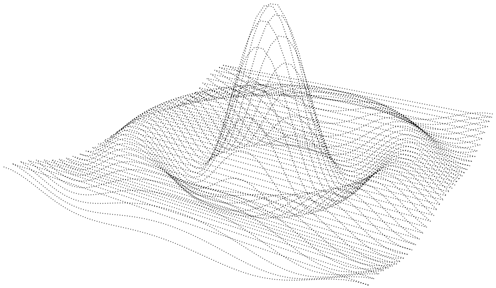
Solution to `persp()`

The points...



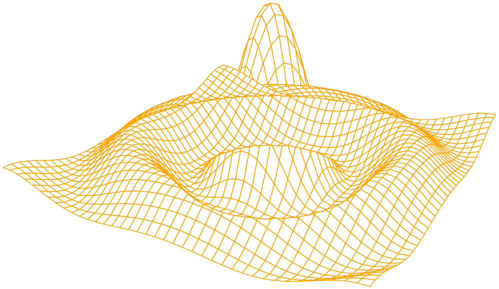
Solution to `persp()`

The lines...



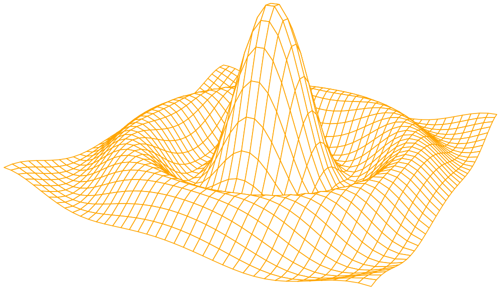
Solution to `persp()`

The polygons(unordered)...



Solution to `persp()`

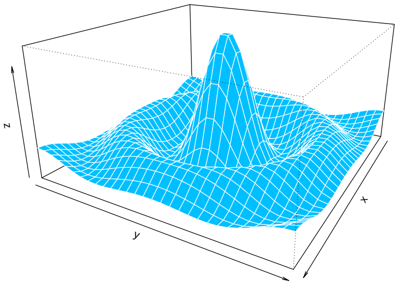
And the polygons(Solution)

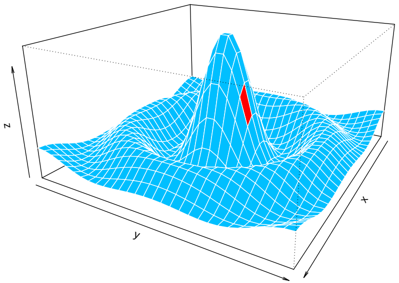


Testing

Any difference?

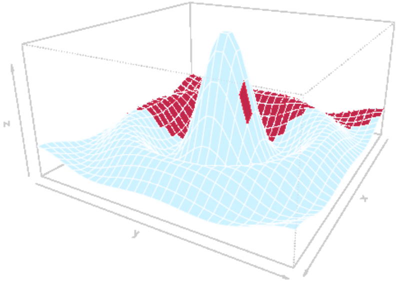
```
## left plot
Sinc_Curve(col = ??)
## right plot
Sinc_Curve(col = ??)
```





Answers

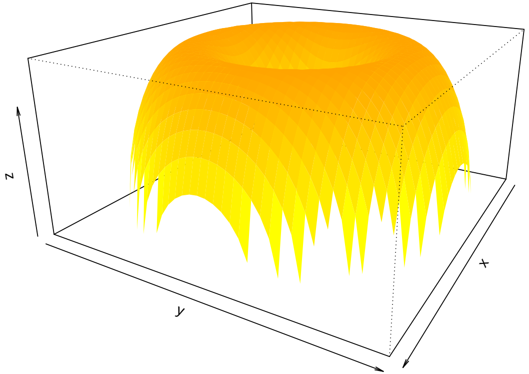
```
## color for left plot
col = rgb(0, 191, 255)

## extra diff color for right
plot
col = rgb(0, 190, 255)
```



Difference dected by using the sorftware* **ImageMagick

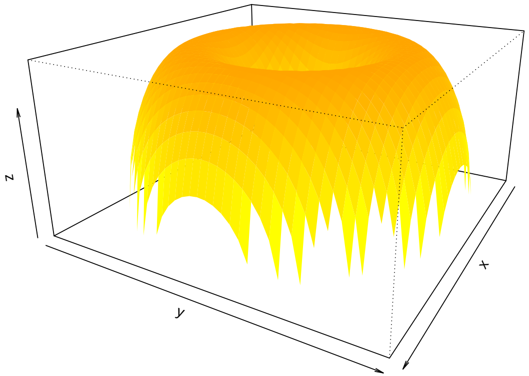
Final solution

```
> Torus()
```



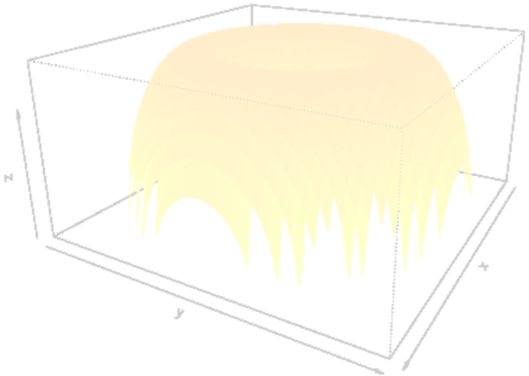
Final solution

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



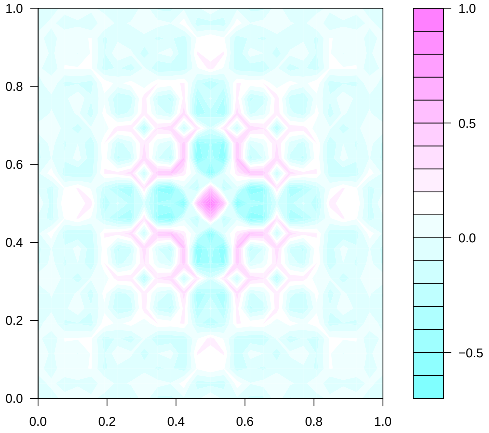
Final solution

Difference



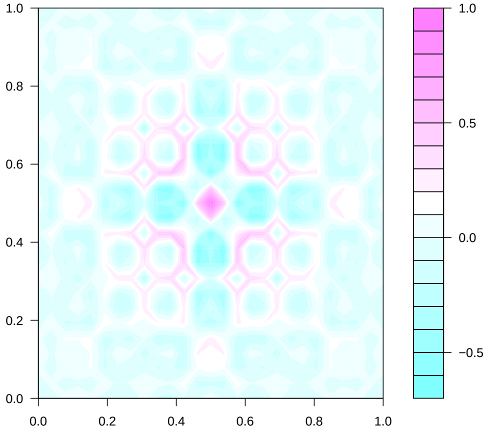
Final solution

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



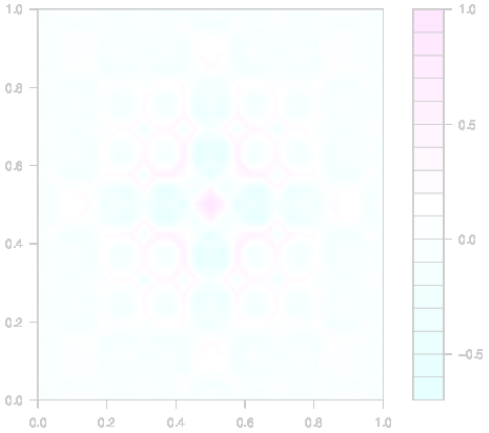
Final solution

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



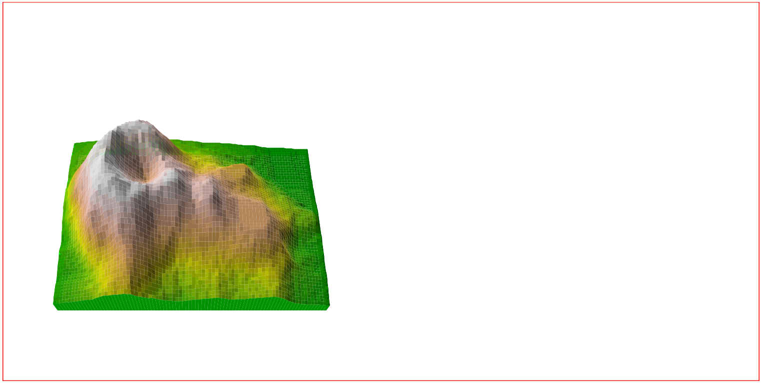
Final solution

Difference



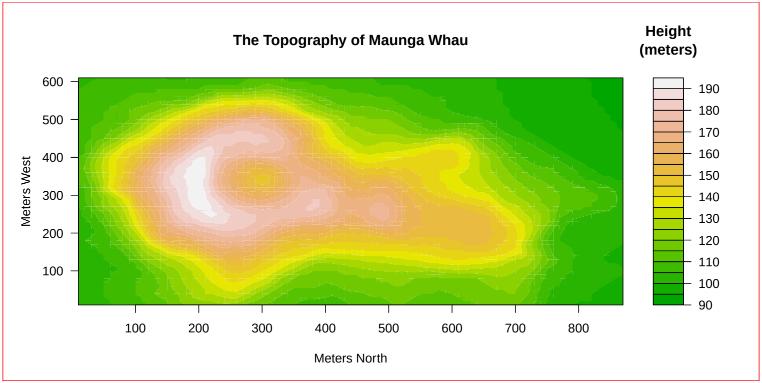
Why use **grid**?

```
> par(mfrow = c(1,2))
> Volcano.persp()
> box('outer', col = 'red')
## volcano_filled.contour()
```



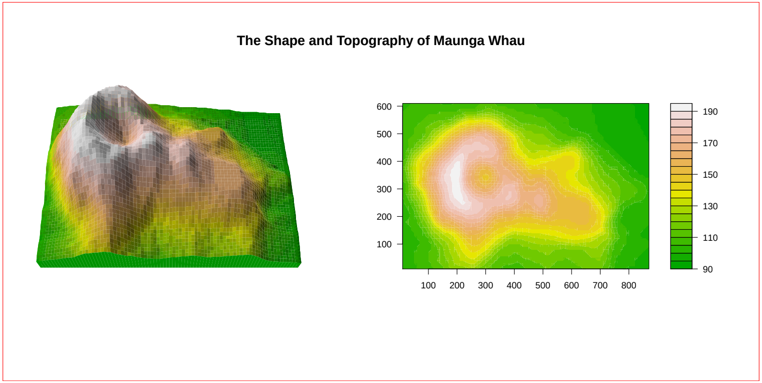
Why use **grid**?

```
> par(mfrow = c(1,2))
> Volcano.persp()
> box('outer', col = 'red')
> volcano_filled.contour()
```



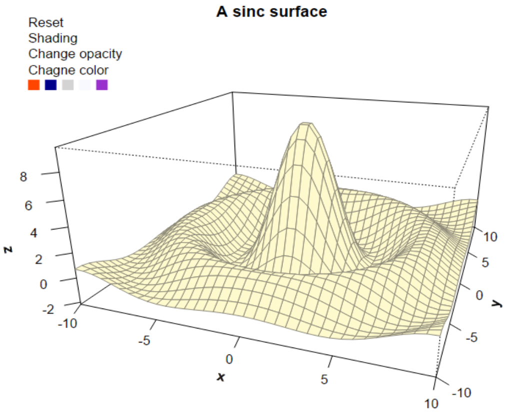
Why is **grid**?

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



Why use **grid** (Advance)?

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



Any Question(s)?