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

Zhijian Wen

University of Auckland

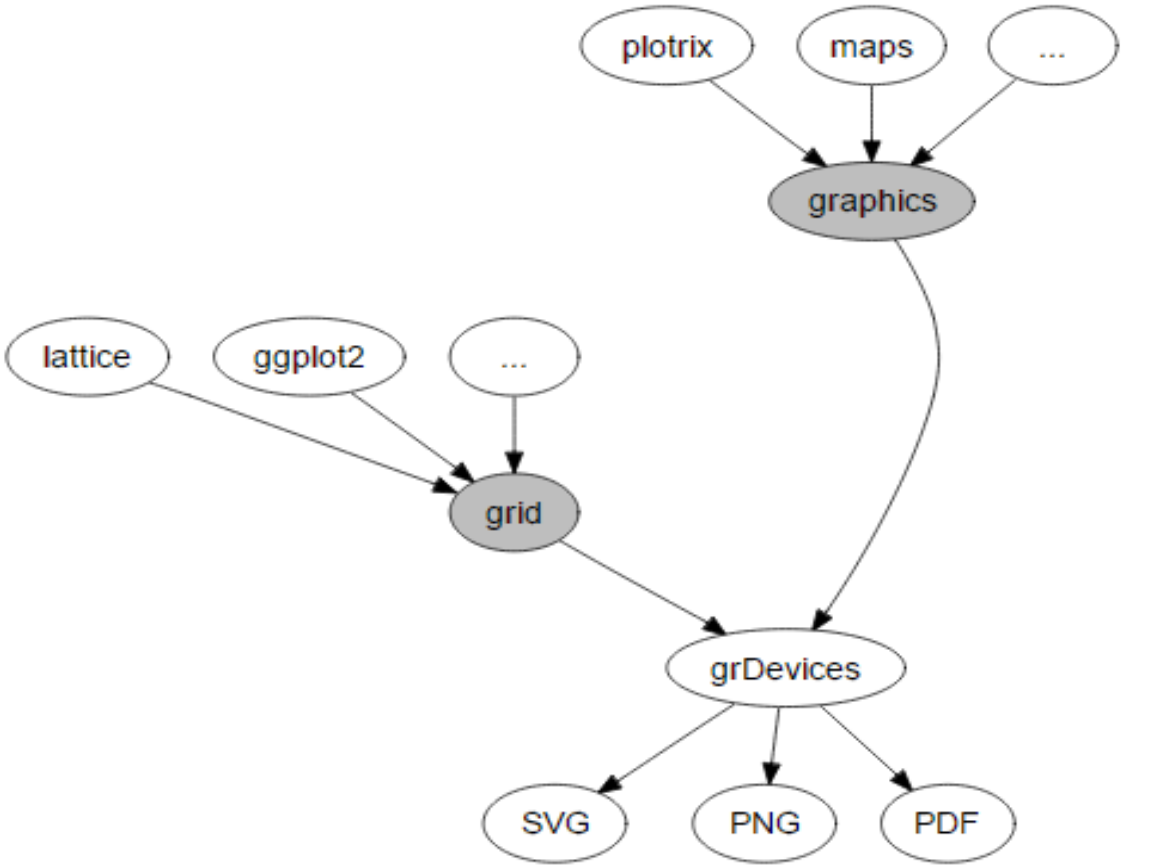
jwen246@aucklanduni.ac.nz

July 3, 2017

Overview

Introduction

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



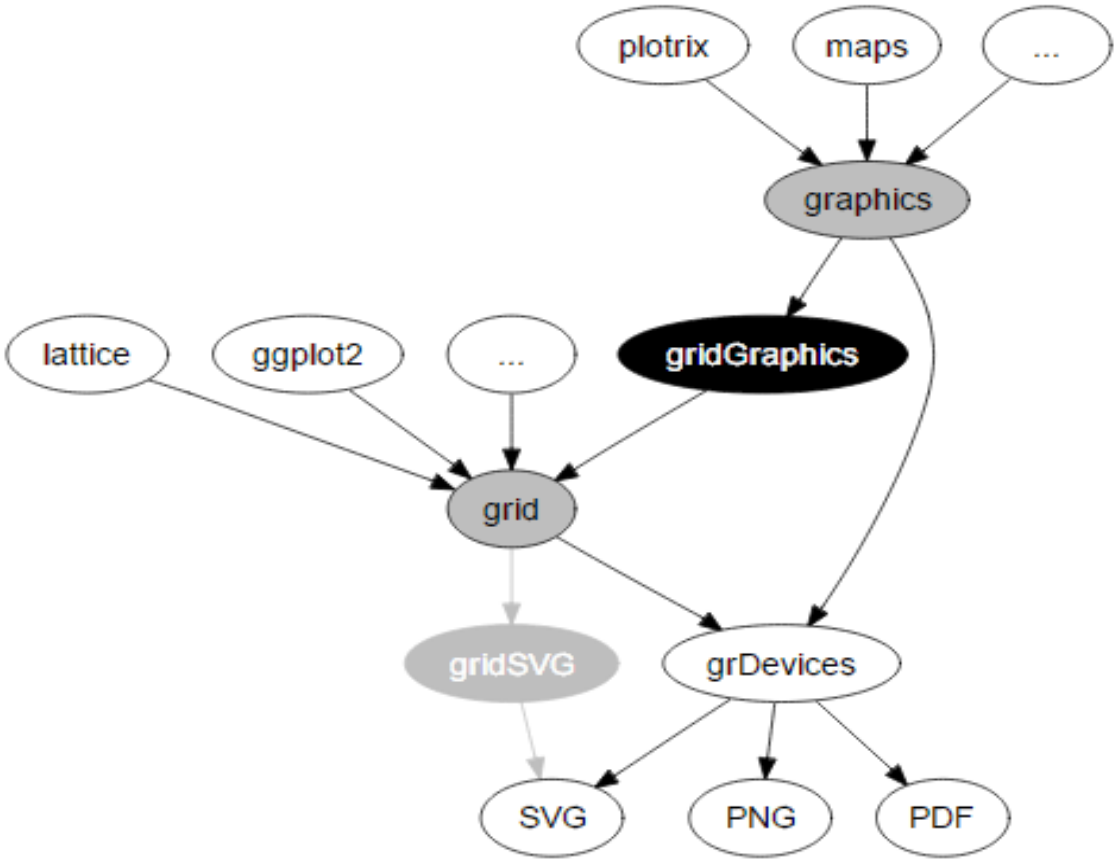
Introduction

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()`.

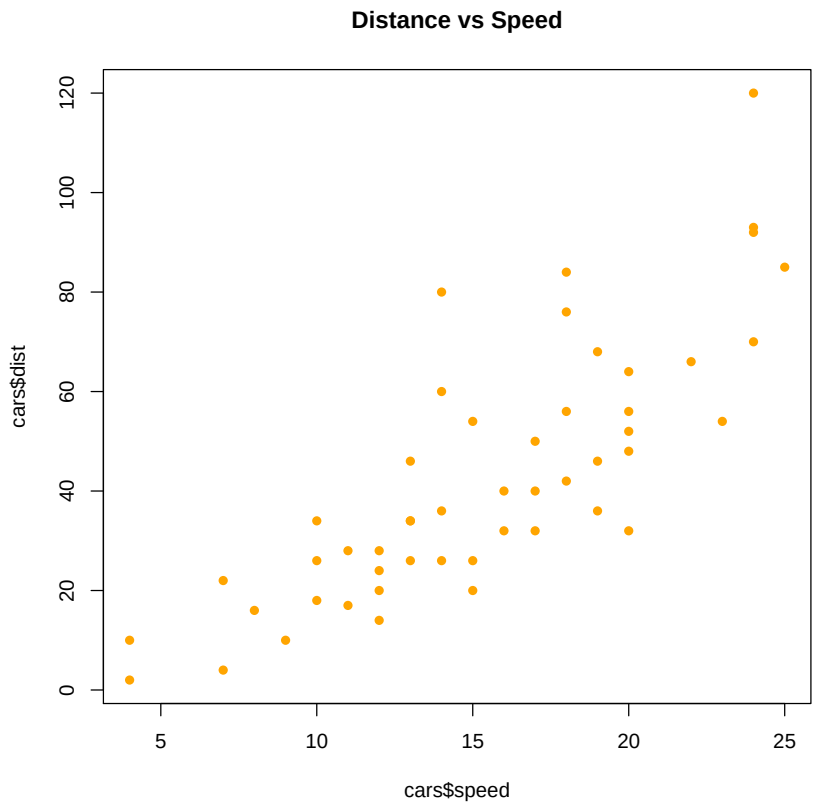
Introduction

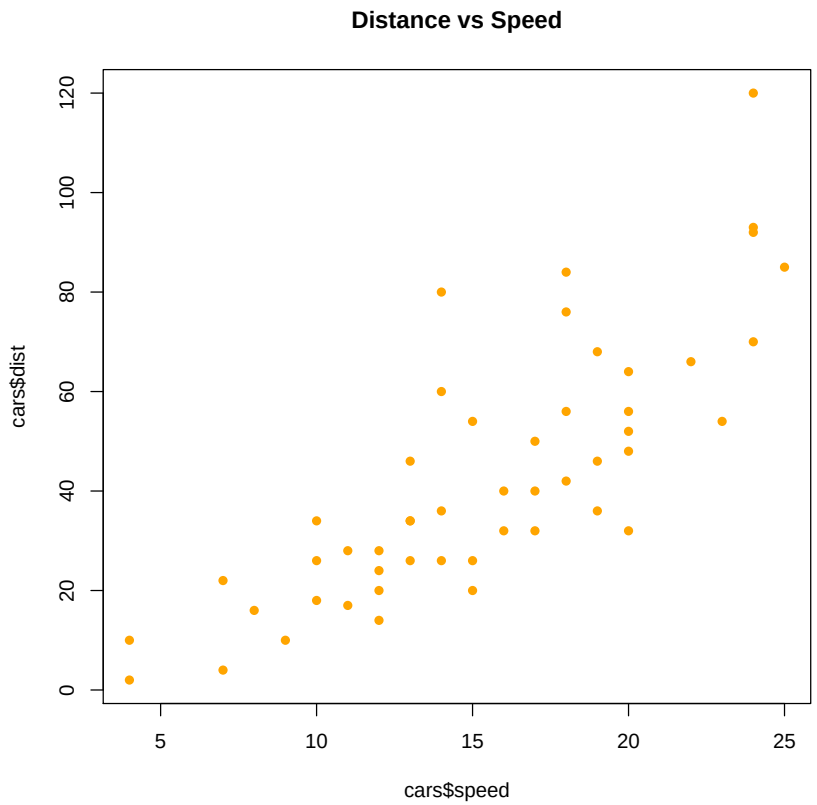
What is **gridGraphics**?



Example

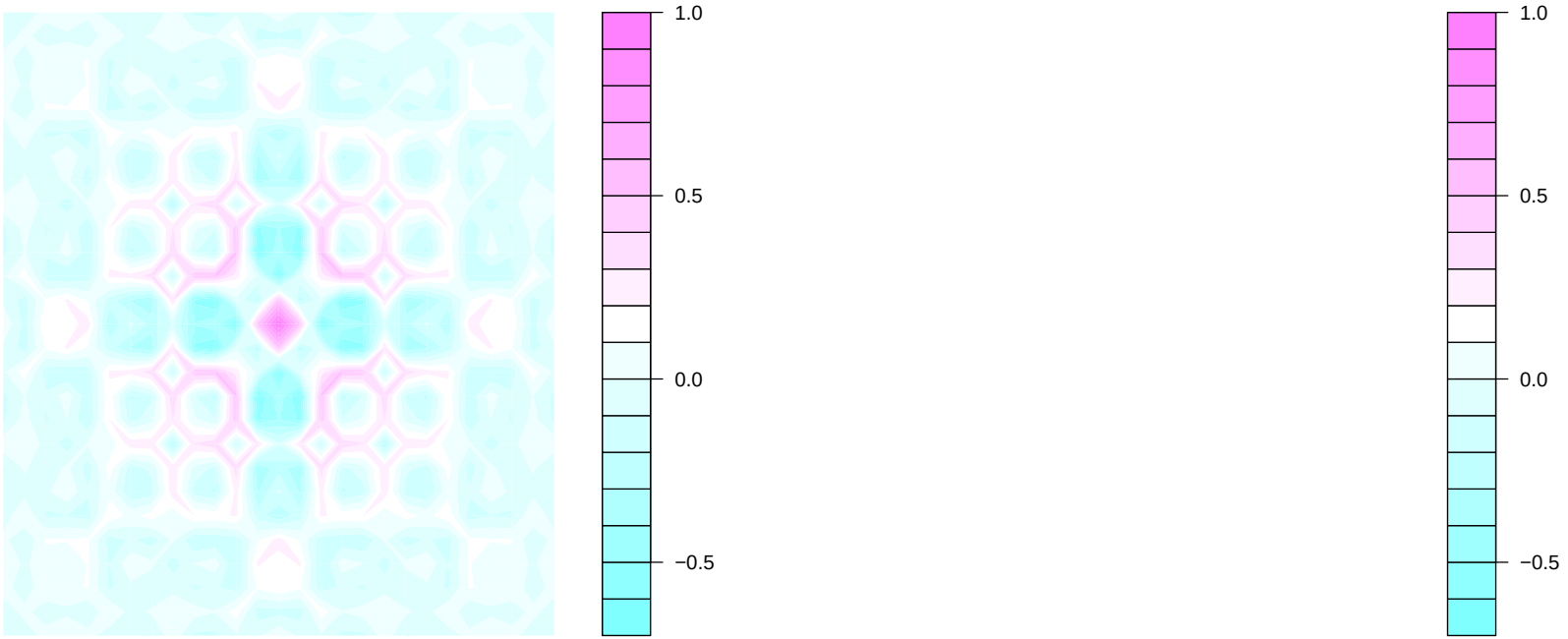
```
> plot(cars$dist ~ cars$speed, pch = 16,
+      col = 'orange', main = 'Distance vs Speed')
> library(gridGraphics)
> grid.echo()
```





The problem

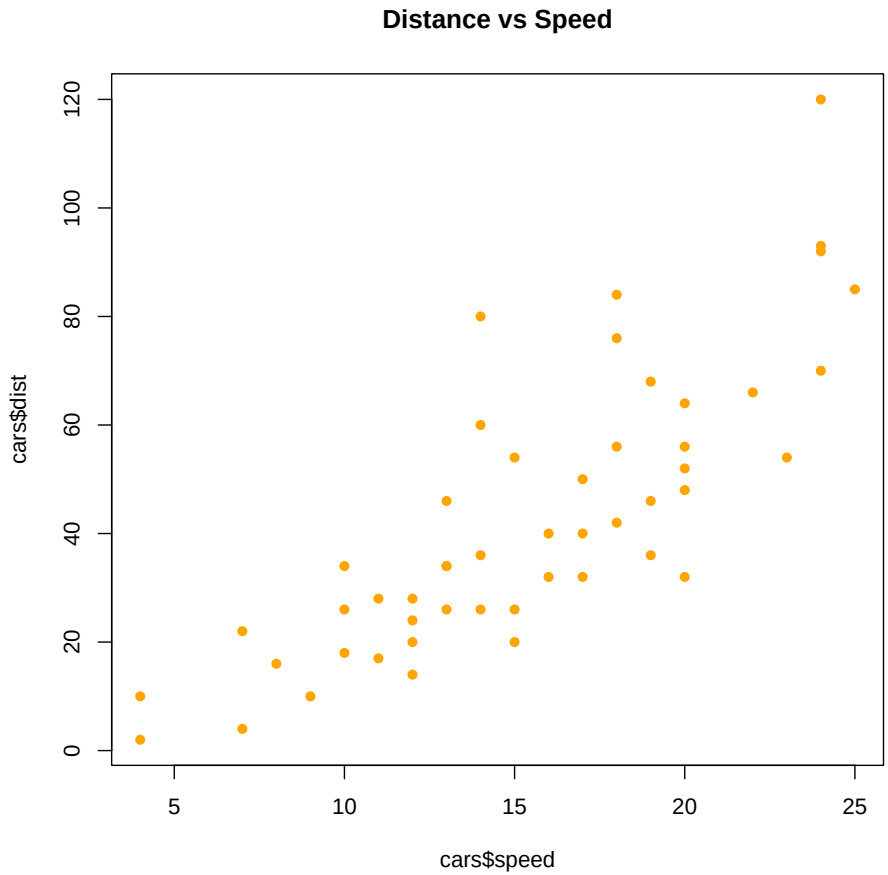
```
> x = y = seq(-4*pi, 4*pi, len = 27)
> r = sqrt(outer(x^2, y^2, "+"))
> filled.contour(cos(r^2)*exp(-r/(2*pi)))
> grid.echo()
```



How **gridGraphics** works?

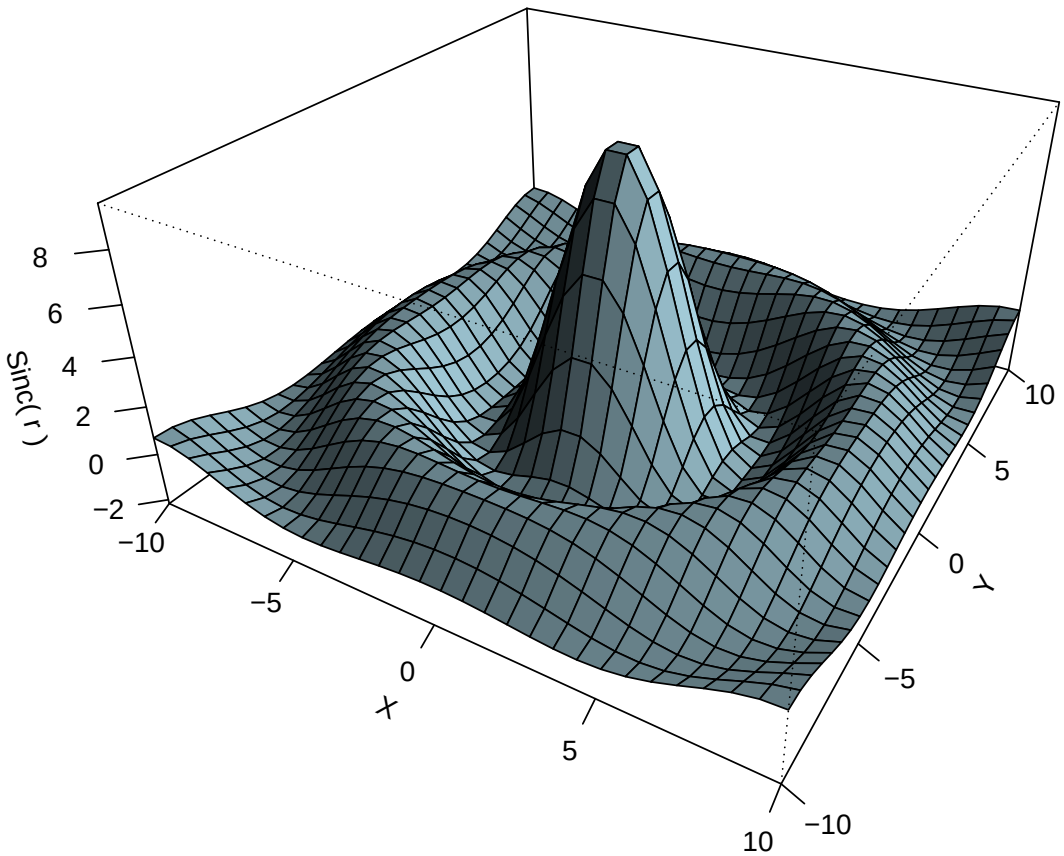
```
x <- recordPlot()
unlist(lapply(x[[1]], function(y) y[[2]][[1]]$name))
```

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



How **gridGraphics** works?

```
> Sinc_Curve()
```



How **gridGraphics** works?

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

[[1]]
[1] "C_plot_new"

[[2]]
[1] "palette2"

[[3]]
[1] "C_persp"
```

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?

Why just 'copy'?

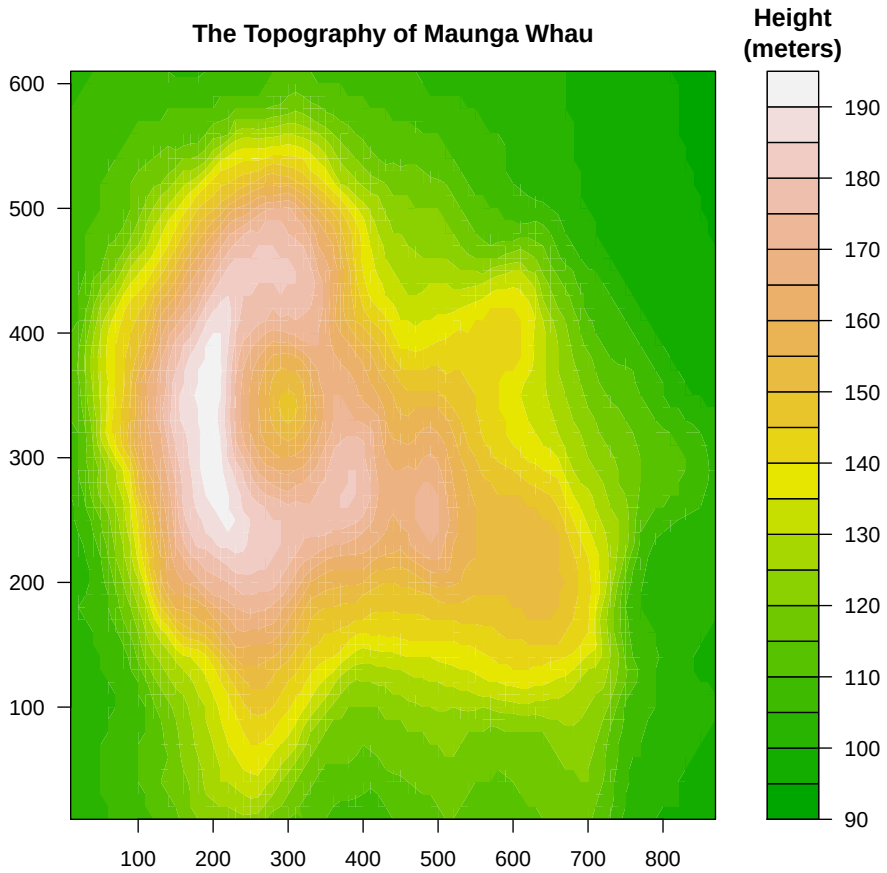
FILL IT!!

Why just not 'copy'?

FILL IT!!

Why just not 'copy'?

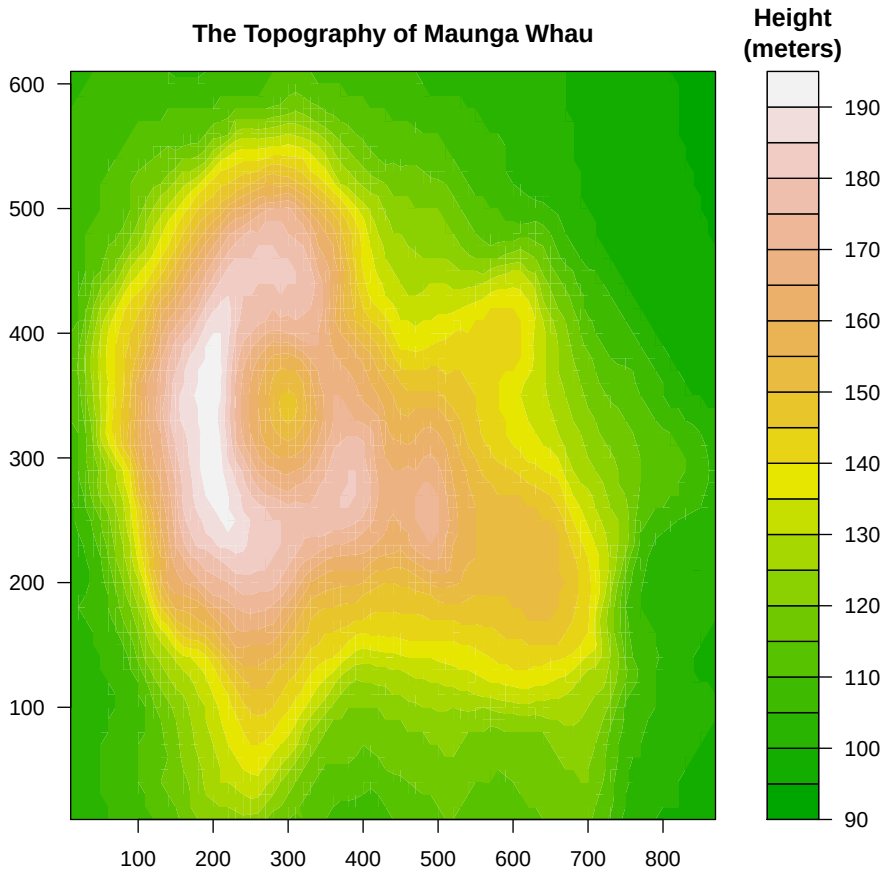
```
volcano_filled.contour()
xx = recordPlot()
info = xx[[1]][[12]][[2]]

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

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



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

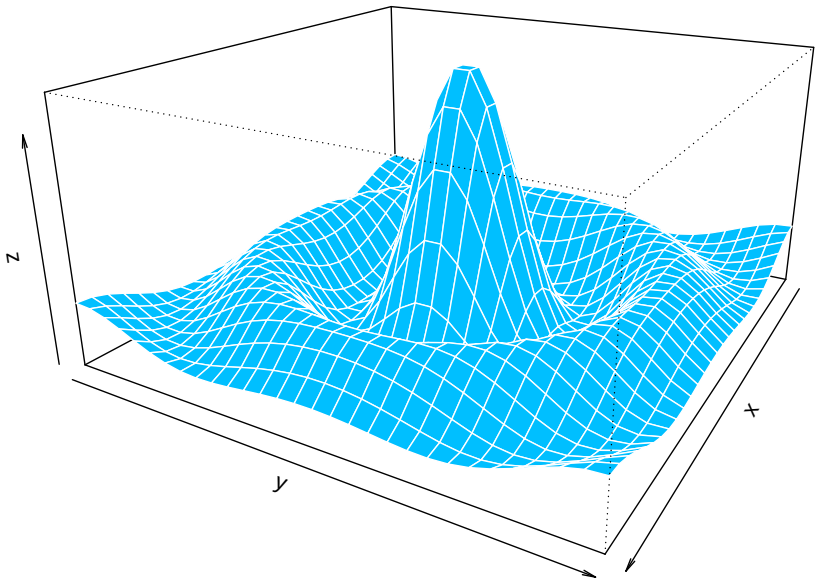
Why just not 'copy'?

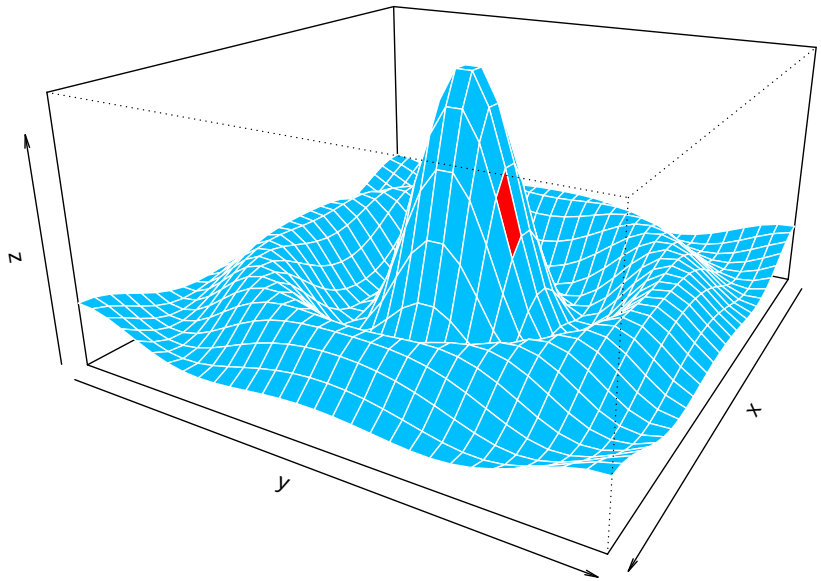
```
volcano_filled.contour()

## time comparison
## time for loop version
system.time(grid.echo())
# user system elapsed
# 10.03 0.23 10.32

## time for vectorizetion
system.time(grid.echo())
# user system elapsed
# 1.28 0.53 1.82
```



Any difference?

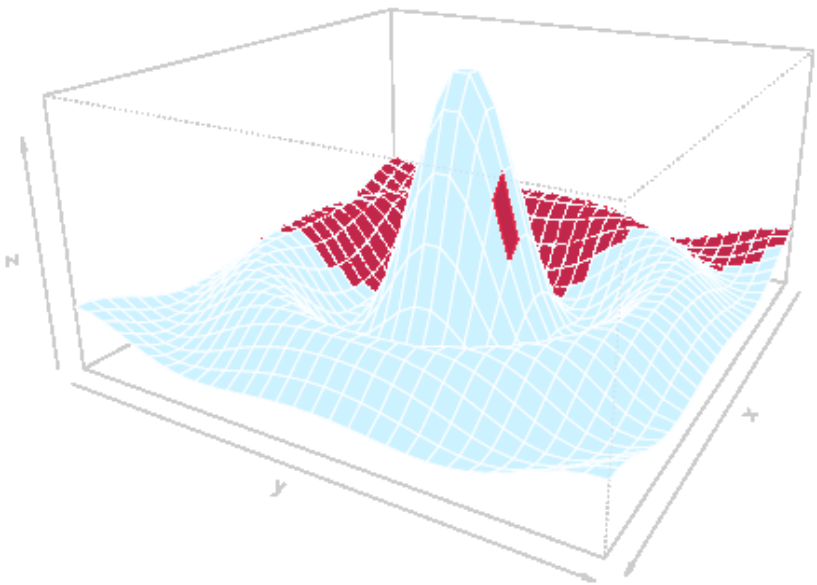
```
## left plot
Sinc_Curve(col = ??)
## right plot
Sinc_Curve(col = ??)
```





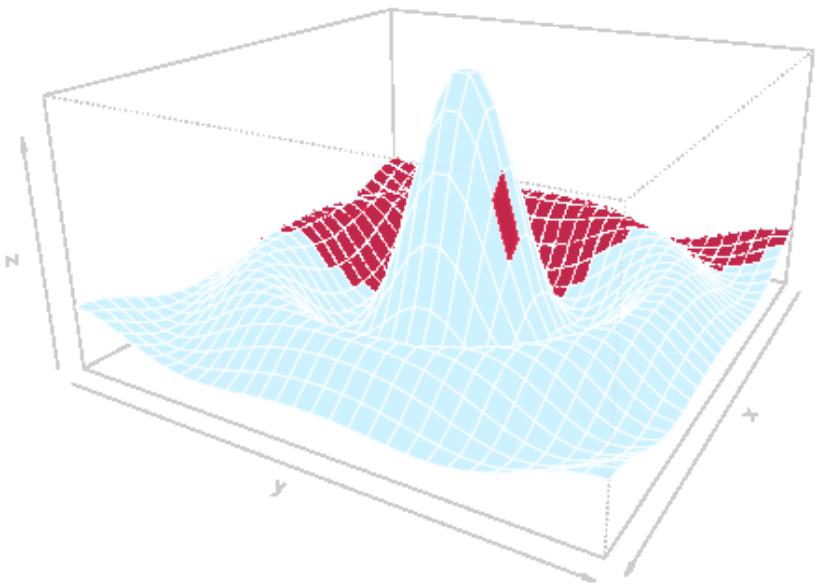
Answers

```
## color for left plot
col = rgb(0, 191, 255, maxColorValue = 255)
## extra diff color for right plot
col = rgb(0, 190, 255, maxColorValue = 255)
```



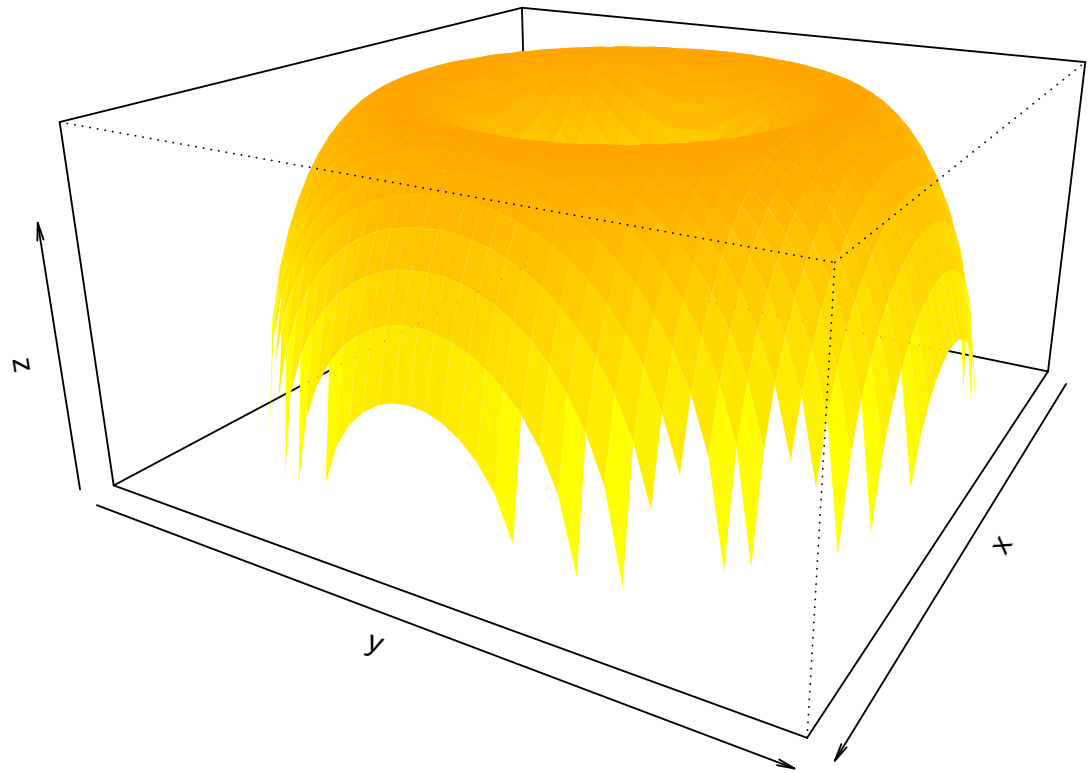
The solustions

```
## color for left plot
col = rgb(0, 191, 255, maxColorValue = 255)
## extra diff color for right plot
col = rgb(0, 190, 255, maxColorValue = 255)
```



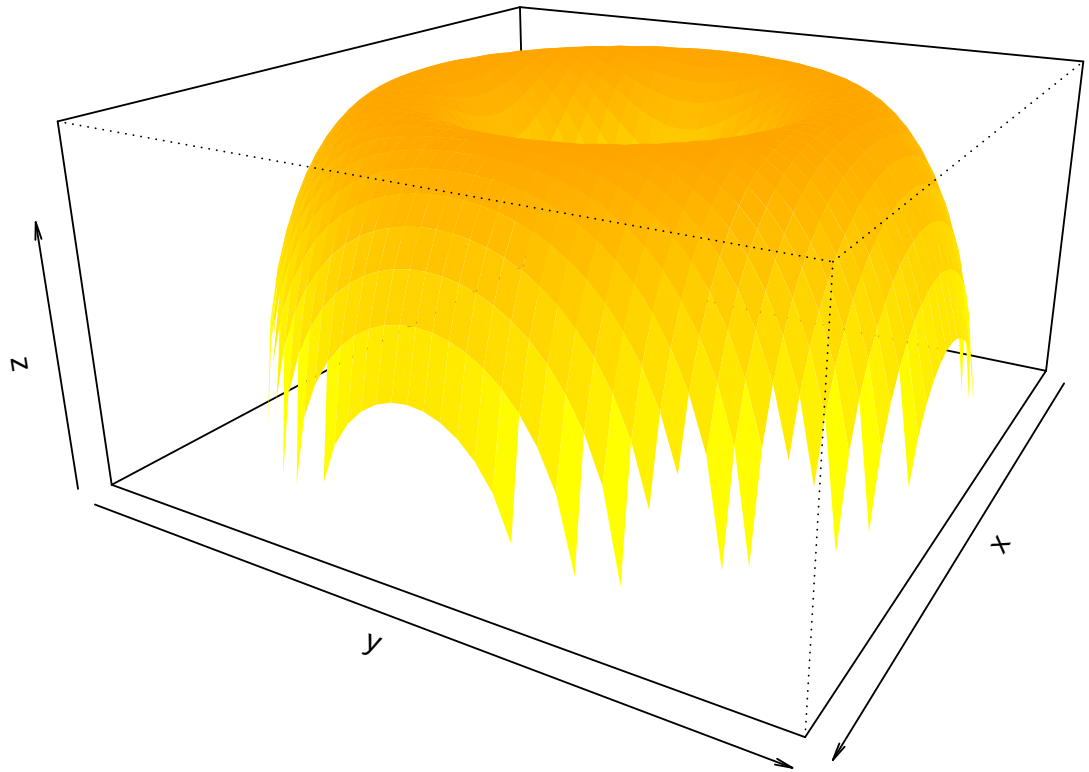
Final solution

```
> Torus()
```



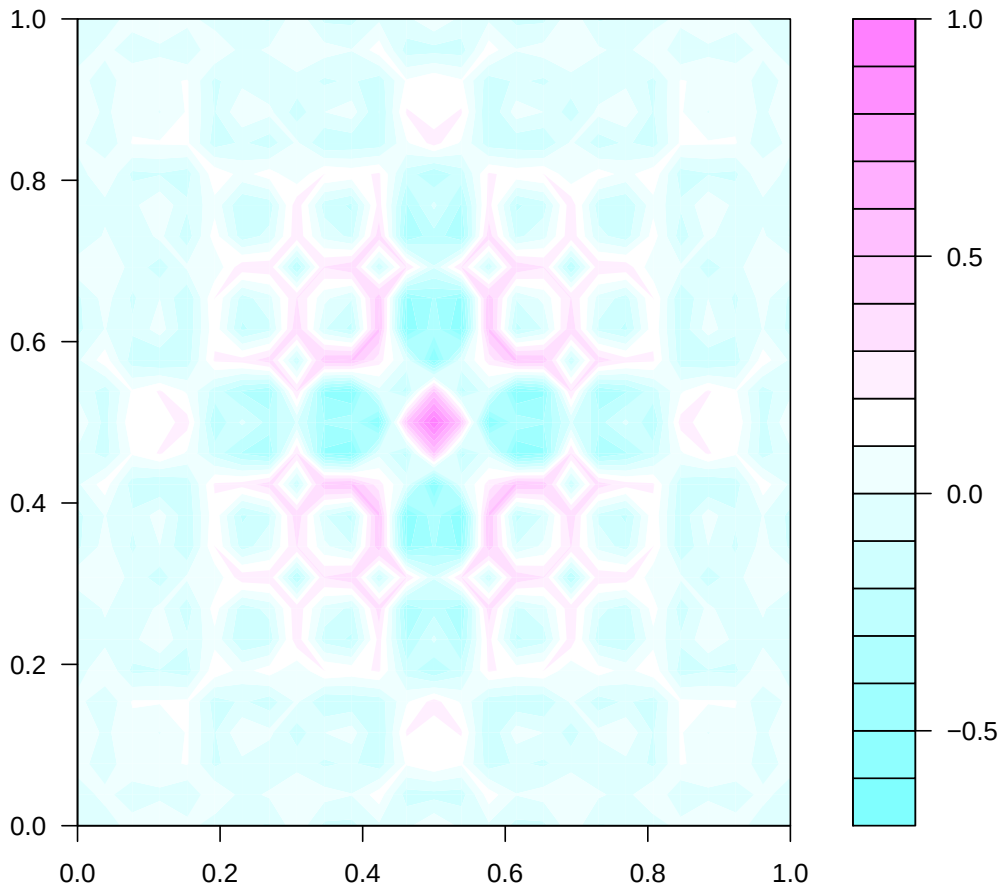
Final solution

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



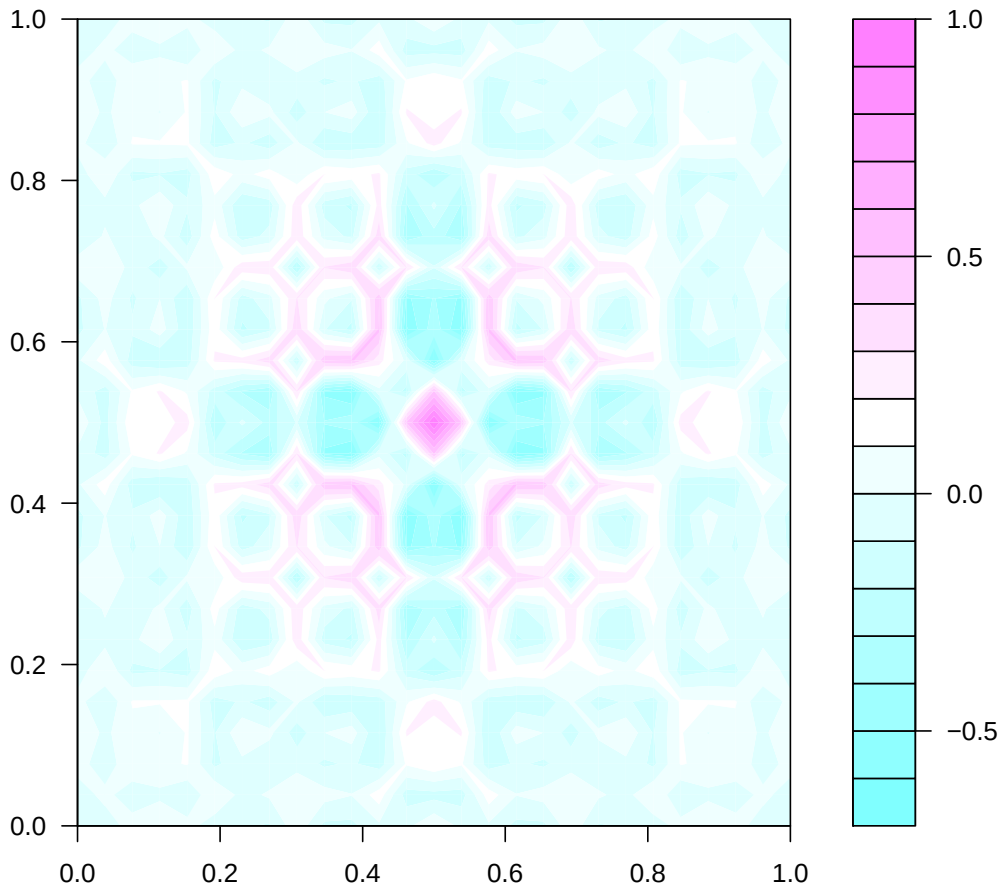
Final solution

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



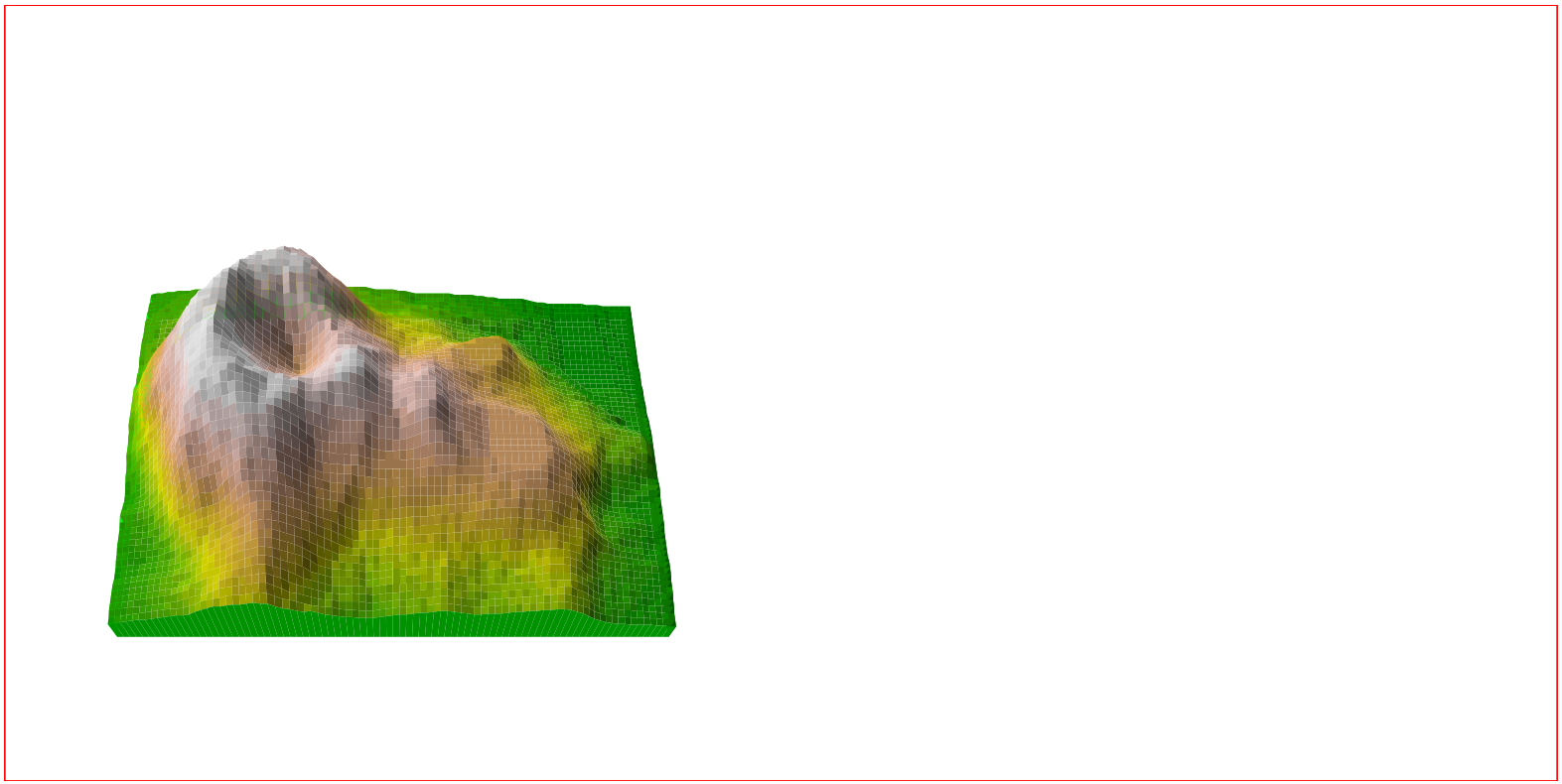
Final solution

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



Final solution

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



Final solution

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