Lecture 5: 3D and Interactive graphics

October 17, 2016

Packages:

- **`plot3D`** - package provides functions for static plotting 2-D and 3-D data,
- **`plotly`** - package translates ‘ggplot2’ graphs to an interactive web-based version and/or create custom interactive web-based visualizations directly from R.
- **`heatmaply`** - package for generating interactive heatmaps
- **`ggmap`** - package that let you retrieve maps from popular online mapping services like Google Maps, OpenStreetMap, Stamen Maps, and plot them using the ggplot2 framework

Install the packages

```
# Install packages:
.packages <- c("plot3D", "plotly", "heatmaply", "ggmap")
lapply(.packages, install.packages, dependencies = TRUE)
```

plot3D

Volcano dataset

- **`volcano`** - a (built-in) dataset on topographic information for Maunga Whau (Mt Eden), one of 50 volcanos in Auckland, New Zealand.
- It consist of a 87 x 61 matrix storing the mountain's atlitutes [m] on a 10m by 10m grid.
- rows run east to west, and columns south to north

```r
dim(volcano)
```

```
## [1] 87 61
```

```r
volcano[1:10, 1:10]
```

```
## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,] 100 100 101 101 101 101 101 100 100 100
## [2,] 101 101 102 102 102 102 102 101 101 101
## [3,] 102 102 103 103 103 103 103 102 102 102
## [4,] 103 103 104 104 104 104 104 103 103 103
## [5,] 104 104 105 105 105 105 105 104 104 103
## [6,] 105 105 105 106 106 106 106 105 105 104
## [7,] 105 106 106 107 107 107 107 106 106 105
## [8,] 106 107 107 108 108 108 108 107 107 106
## [9,] 107 108 108 109 109 109 109 108 108 107
## [10,] 108 109 109 110 110 110 110 109 109 108
```

We reduce the resolution by a factor of 3 to make everything run faster:

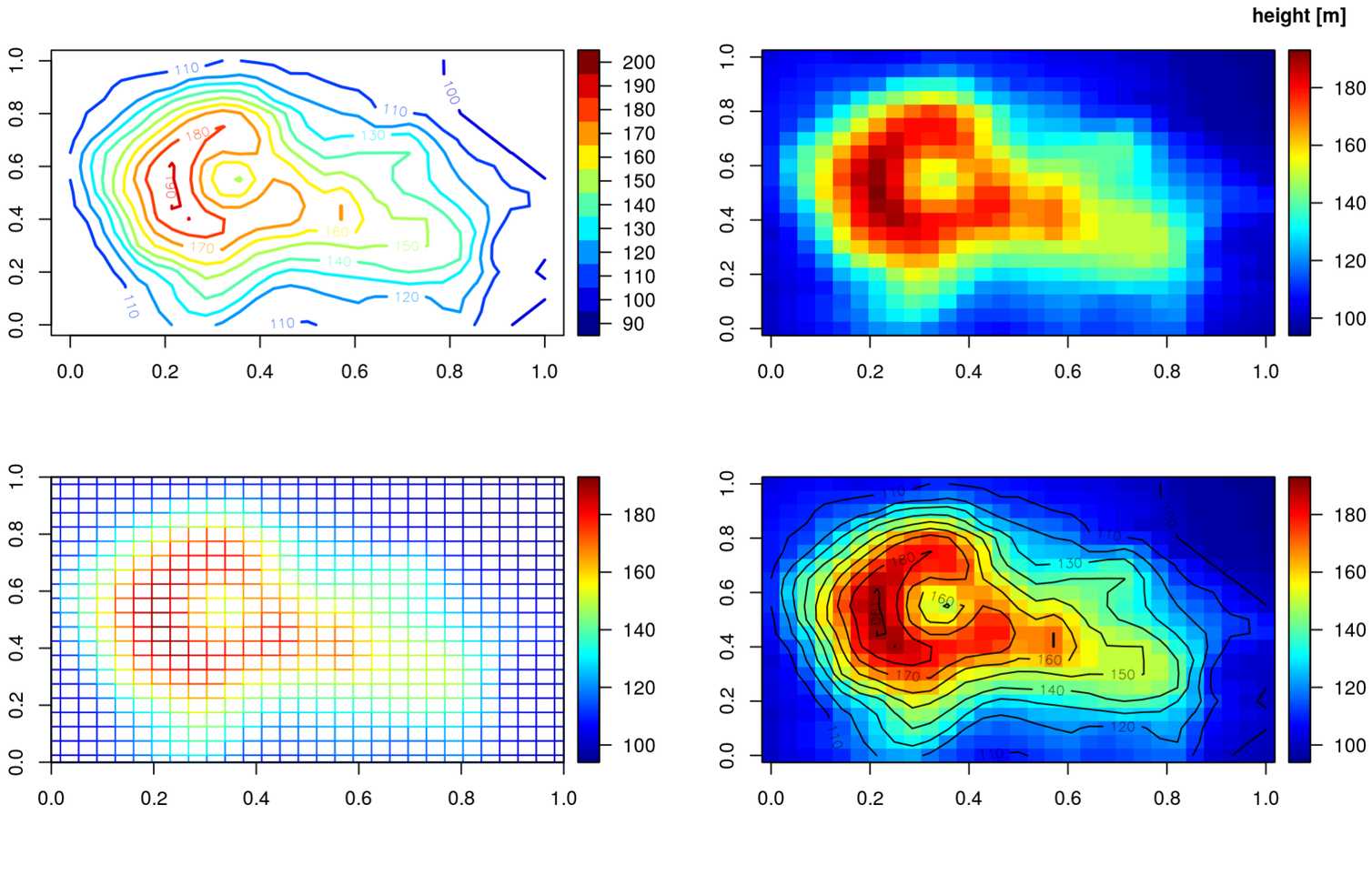
```r
# Reduce the resolution
Volcano <- volcano[seq(1, nrow(volcano), by = 3),
```

```
seq(1, ncol(volcano), by = 3)]
```

2D images of volcano

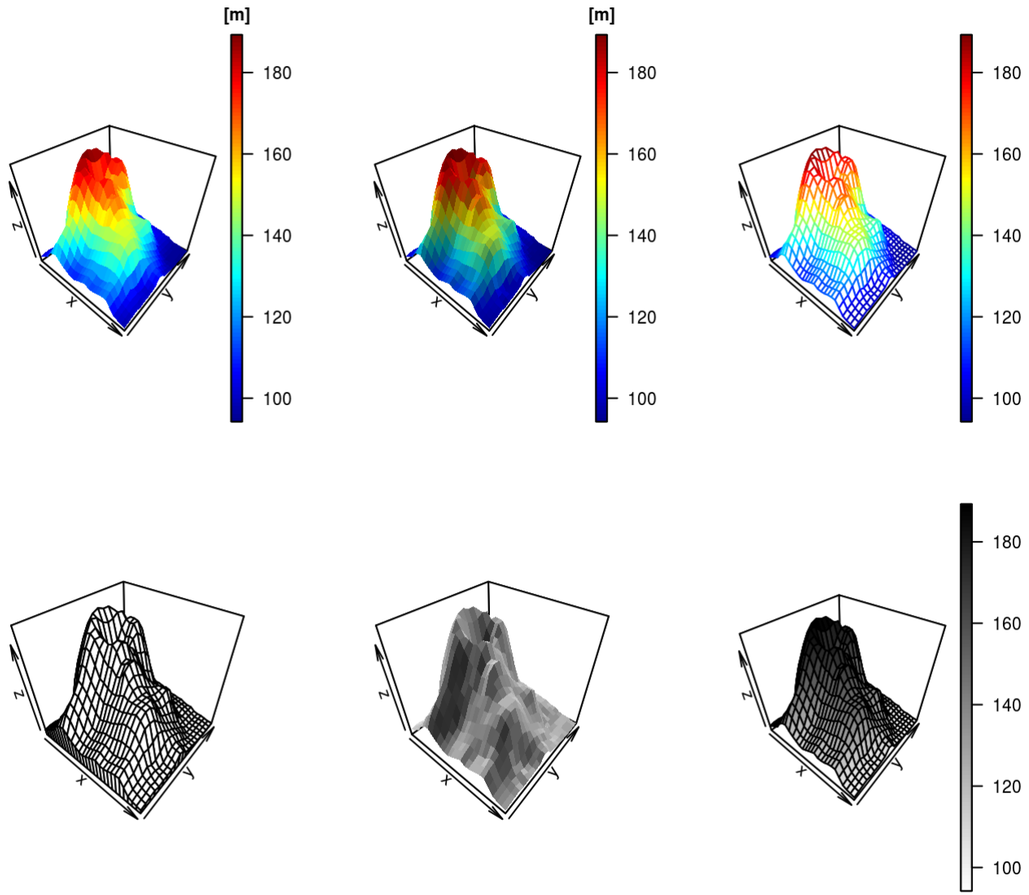
- **`plot3D`** includes useful `contour2D()` and `image2D()` functions which extend `contour()` and `image()` from R basic graphics.
- They let you produce 2D topological maps.

```r
library(plot3D)
par(mfrow = c(2, 2), mar = c(3, 3, 3, 3)) # mar = c(bottom, left, top, right)
contour2D(Volcano, lwd = 2)
image2D(Volcano, clab = "height [m]") # set the color label
image2D(Volcano, facets = FALSE) # don't fill in the tiles
image2D(Volcano, contour = TRUE) # include contours
```

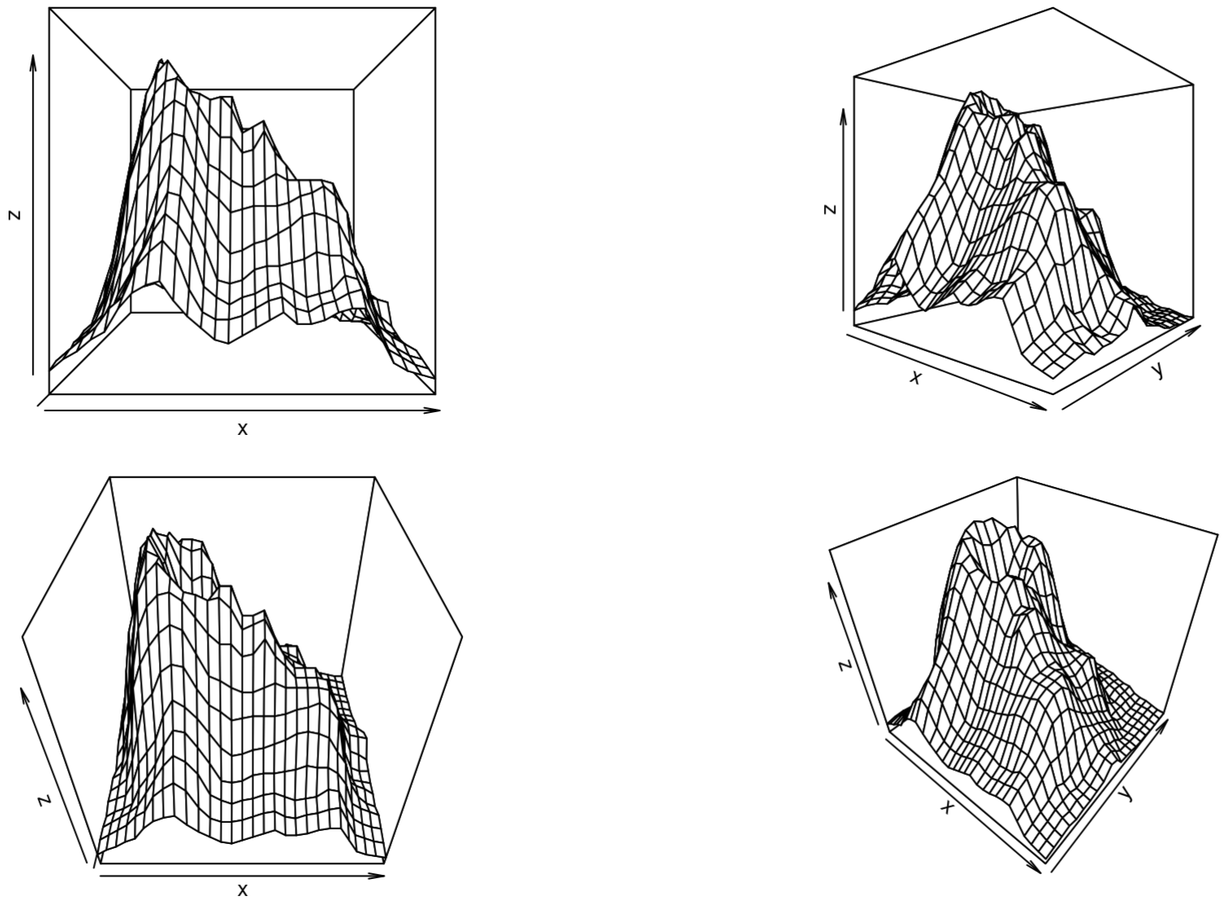


3D Volcano perspective plots

```
par(mfrow = c(2, 3), mar = c(2, 2, 2, 3.5))
persp3D(z = Volcano, clab = "[m]")
persp3D(z = Volcano, clab = "[m]", shade = 0.2)
persp3D(z = Volcano, facets = FALSE)
persp3D(z = Volcano, facets = FALSE, col = "black", curtain = TRUE)
persp3D(z = Volcano, col = "white", shade = 0.5)
persp3D(z = Volcano, col = ramp.col(c("white", "black")), border = "black")
```

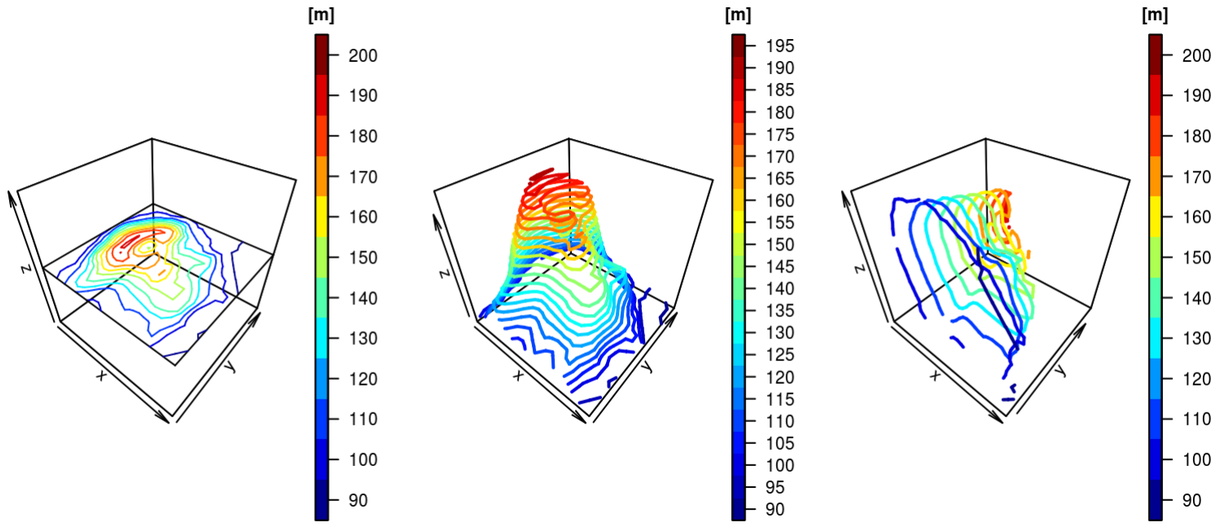

[m]
180
160
140
120
100
z
x
y

```r
par(mfrow = c(2, 2), mar = c(1, 1, 1, 1))
persp3D(z = Volcano, facets = FALSE, col = "black", theta = 0, phi = 0)
persp3D(z = Volcano, facets = FALSE, col = "black", theta = 40, phi = 0)
persp3D(z = Volcano, facets = FALSE, col = "black", theta = 0, phi = 40)
persp3D(z = Volcano, facets = FALSE, col = "black")
```

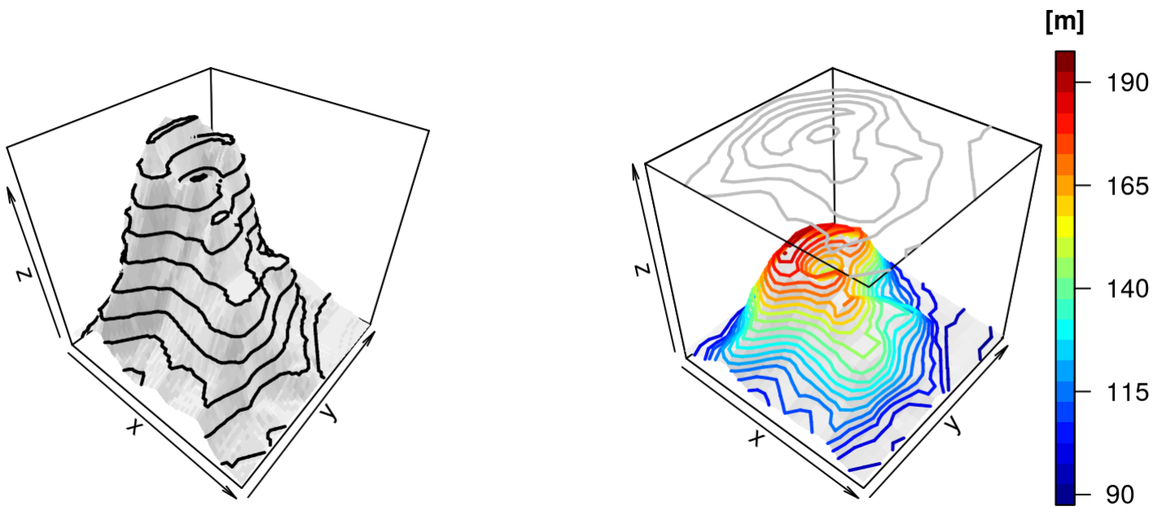


3D Contour plots

```
par(mfrow = c(1, 3), mar = c(1, 1, 2, 3))
contour3D(z = 100, colvar = Volcano, clab = c("[m]"))
contour3D(z = Volcano, colvar = Volcano, lwd = 2,
nlevels = 20, clab = "[m]")
contour3D(y = Volcano, colvar = Volcano, lwd = 2,
nlevels = 10, clab = "[m]")
```

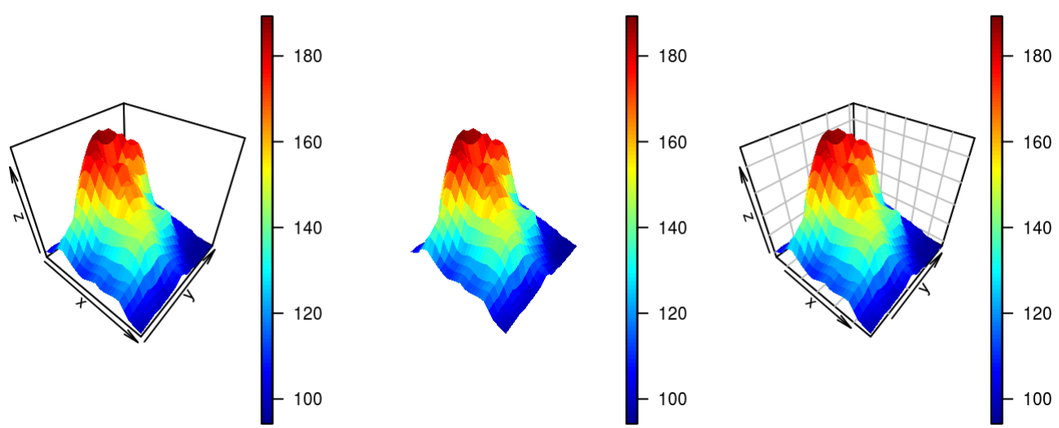


```r
par(mfrow = c(1, 2), mar = c(1, 1, 2, 3))
# Generating perspective and contour plots together
persp3D(z = volcano, col = "white", shade = 0.1, plot = FALSE)
contour3D(z = volcano, colvar = volcano, lwd = 2,
add = TRUE, col = "black")
persp3D(z = Volcano, zlim = c(90, 300), col = "white",
shade = 0.1, d = 2, plot = FALSE)
contour3D(z = Volcano, colvar = Volcano, lwd = 2, add = TRUE,
nlevels = 20, clab = "[m]", plot = FALSE,
colkey = list(at = seq(90, 190, length.out = 5)))
contour3D(z = 300, colvar = Volcano, lwd = 2, col = "grey",
add = TRUE, nlevels = 5)
```

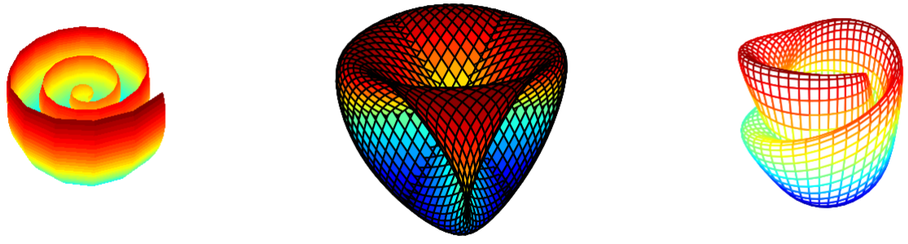


3D Surface plots

```r
par(mfrow = c(1, 3), mar = c(1, 1, 2, 3))
# Recall that with persp3D we only need a matrix of altitudes (z)
persp3D(z = Volcano)
# With surf3D we need to specify (x,y,z) at all mesh points
# First we create a mesh:
r <- 1:nrow(Volcano); c <- 1:ncol(Volcano)
M <- mesh(r, c)
# x, y, z are all matrices here
# dim(M$x) = dim(M$y) = dim(Volcano) = (29 x 21)
surf3D(x = M$x, y = M$y, z = Volcano)
surf3D(x = M$x, y = M$y, z = Volcano, bty = "b2")
```



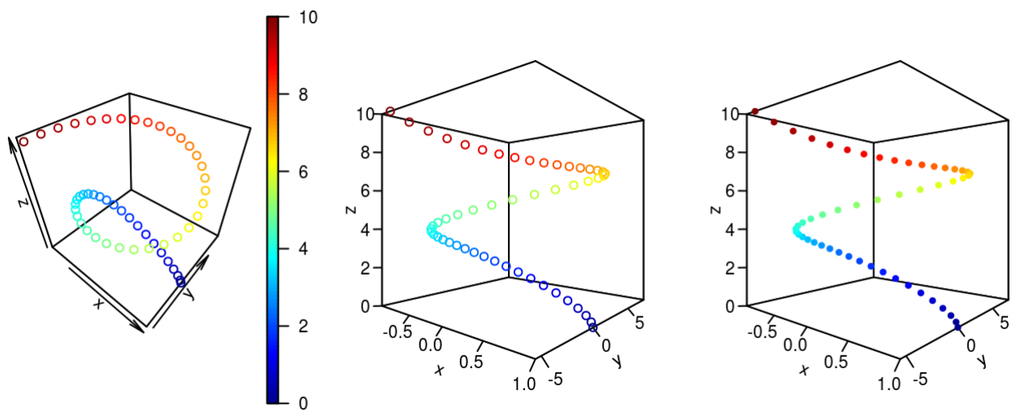
```r
par(mfrow = c(1, 3), mar = c(1, 1, 2, 3))
# Shape 1
M <- mesh(seq(0, 6*pi, length.out = 50),
seq(pi/3, pi, length.out = 50))
u <- M$x ; v <- M$y
x <- u/2 * sin(v) * cos(u)
y <- u/2 * sin(v) * sin(u)
z <- u/2 * cos(v)
surf3D(x, y, z, colvar = z, colkey = FALSE, phi = 50)
# Shape 2: add border
M <- mesh(seq(0, 2*pi, length.out = 50),
seq(0, 2*pi, length.out = 50))
u <- M$x ; v <- M$y
x <- sin(u); y <- sin(v); z <- sin(u + v)
surf3D(x, y, z, colvar = z, border = "black", colkey = FALSE)
# shape 3: uses same mesh, other perspective (d >1)
x <- (3 + cos(v/2)*sin(u) - sin(v/2)*sin(2*u))*cos(v)
y <- (3 + cos(v/2)*sin(u) - sin(v/2)*sin(2*u))*sin(v)
z <- sin(v/2)*sin(u) + cos(v/2)*sin(2*u)
surf3D(x, y, z, colvar = z, colkey = FALSE, facets = FALSE)
```



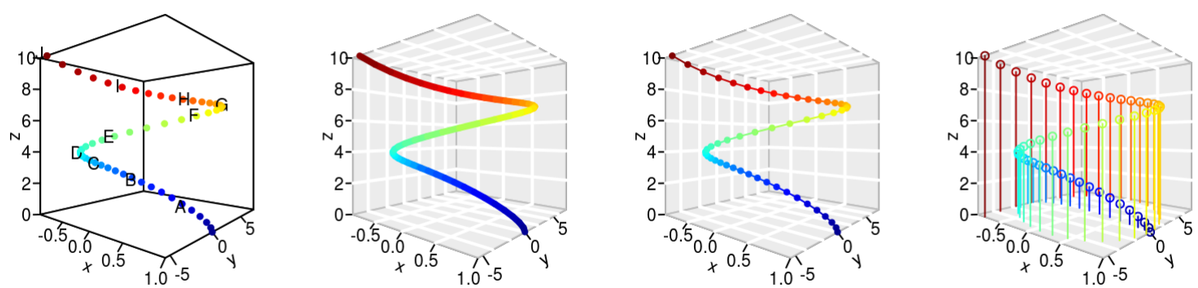
3D Scatter plots

```
par(mfrow = c(1, 3), mar = c(2, 2, 2, 2))
z <- seq(0, 10, 0.2)
x <- cos(z); y <- sin(z)*z

scatter3D(x, y, z)
scatter3D(x, y, z, colkey = FALSE, phi = 0, ticktype = "detailed")
scatter3D(x, y, z, colkey = FALSE, phi = 0, pch = 20, ticktype = "detailed")
```



```r
par(mfrow = c(1, 4), mar = c(1, 2, 1, 2))
scatter3D(x, y, z, colkey = FALSE, phi = 0, pch = 20, ticktype = "detailed")
text3D(x = cos(1:10), y = (sin(1:10)*(1:10) - 1), # add text
z = 1:10, colkey = FALSE, add = TRUE,
labels = LETTERS[1:10], col = "black")
# line plot (bty = "g" greyish background)
scatter3D(x, y, z, colkey = FALSE, phi = 0, bty = "g",
type = "l", ticktype = "detailed", lwd = 4)
# points and lines
scatter3D(x, y, z, colkey = FALSE, phi = 0, bty = "g",
type = "b", ticktype = "detailed", pch = 20)
# vertical lines
scatter3D(x, y, z, colkey = FALSE, phi = 0, bty = "g",
type = "h", ticktype = "detailed")
```



ggplot2 + plotly

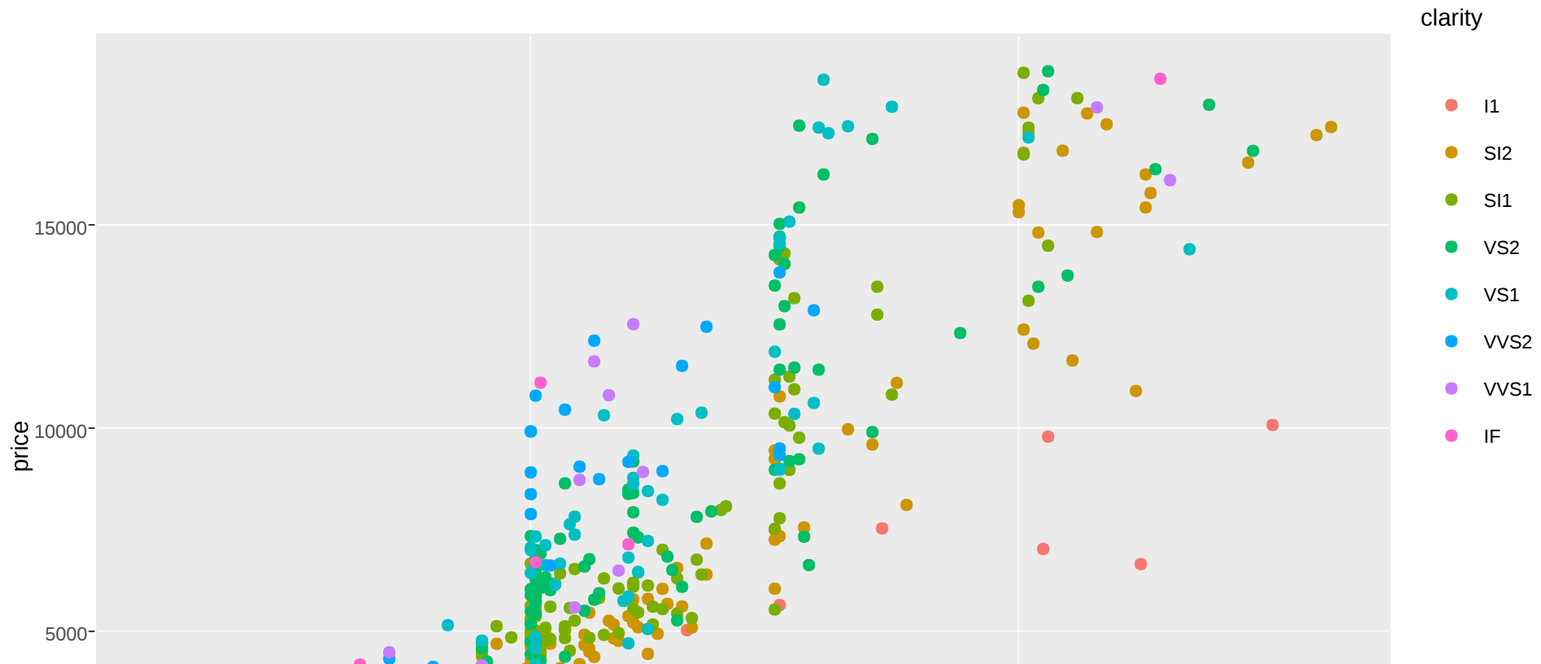
plotly

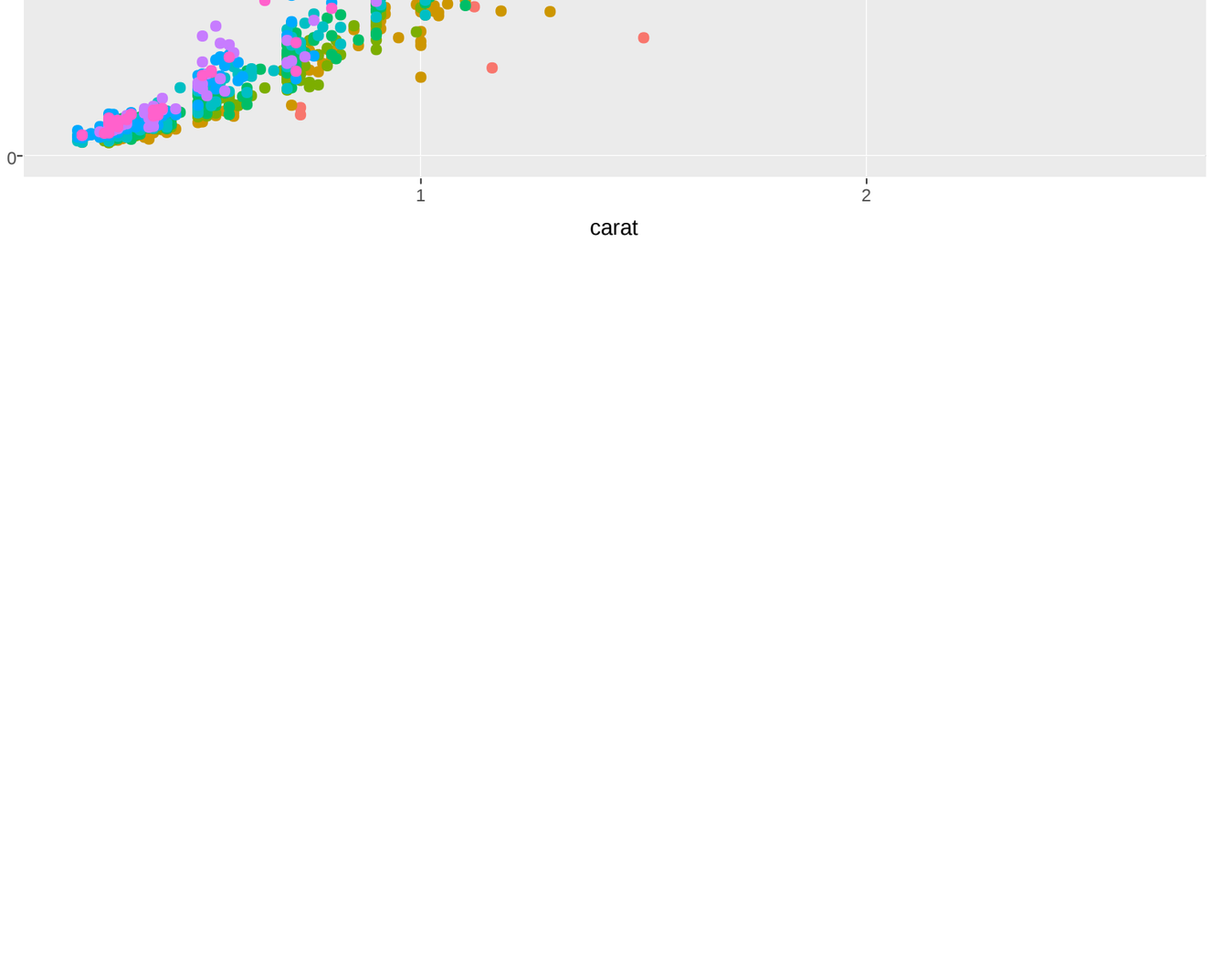
- `plotly` is *the collaboration platform for modern data science*
- Available in **R, python, MATLAB, scala**.
- You can produce **interactive graphics including 3D plots** (with zooming and rotating).
- You can open a **'plotly' account** to upload 'plotly' graphs and view or modify them in a web browser.
- Resources: cheatsheet, book

Integration with ggplot2

```
library(ggplot2); library(plotly)
set.seed(1234)
dsmall <- diamonds[sample(nrow(diamonds), 1000), ]
plt <- ggplot(dsmall, aes(x = carat, y = price)) +
geom_point(aes(colour = clarity))
ggplotly(plt)
```

```
## Warning: We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```




0
1
2
carat

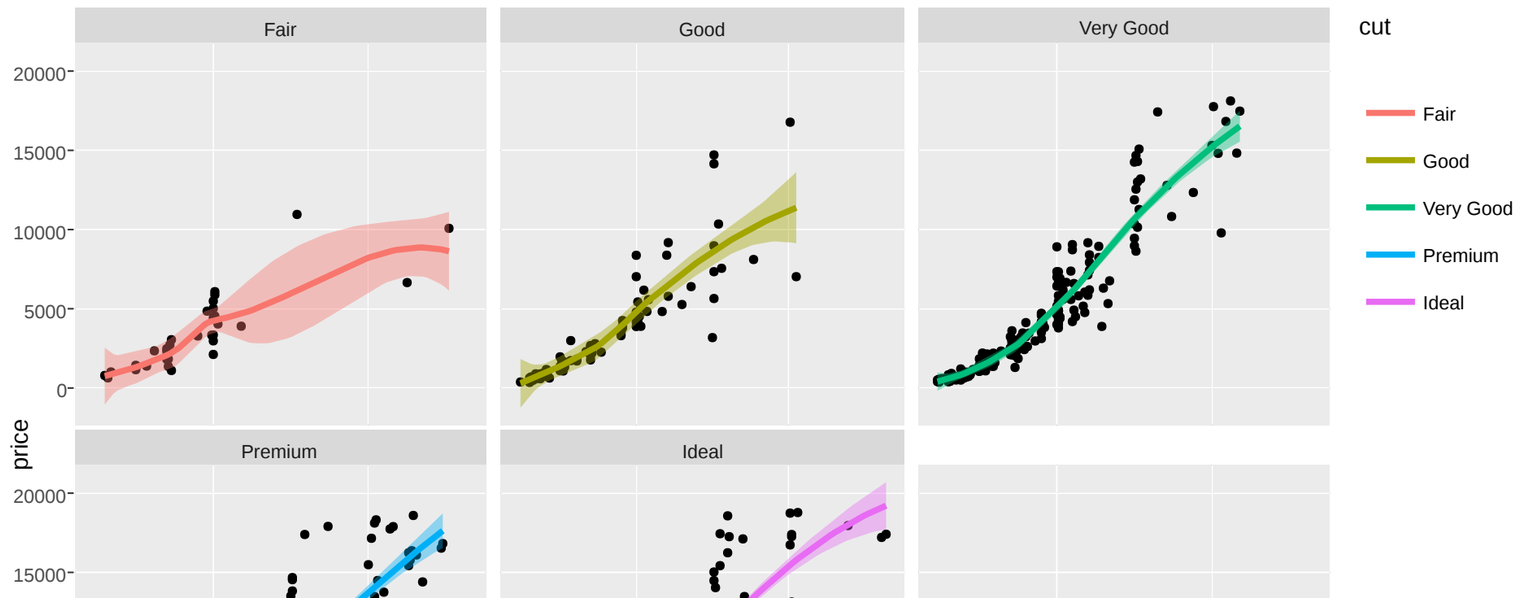
```
plt <- ggplot(data = dsmall, aes(x = carat, y = price)) +
geom_point(aes(text = paste("Clarity:", clarity)), size = 1) +
geom_smooth(aes(colour = cut, fill = cut)) + facet_wrap(~ cut)
```

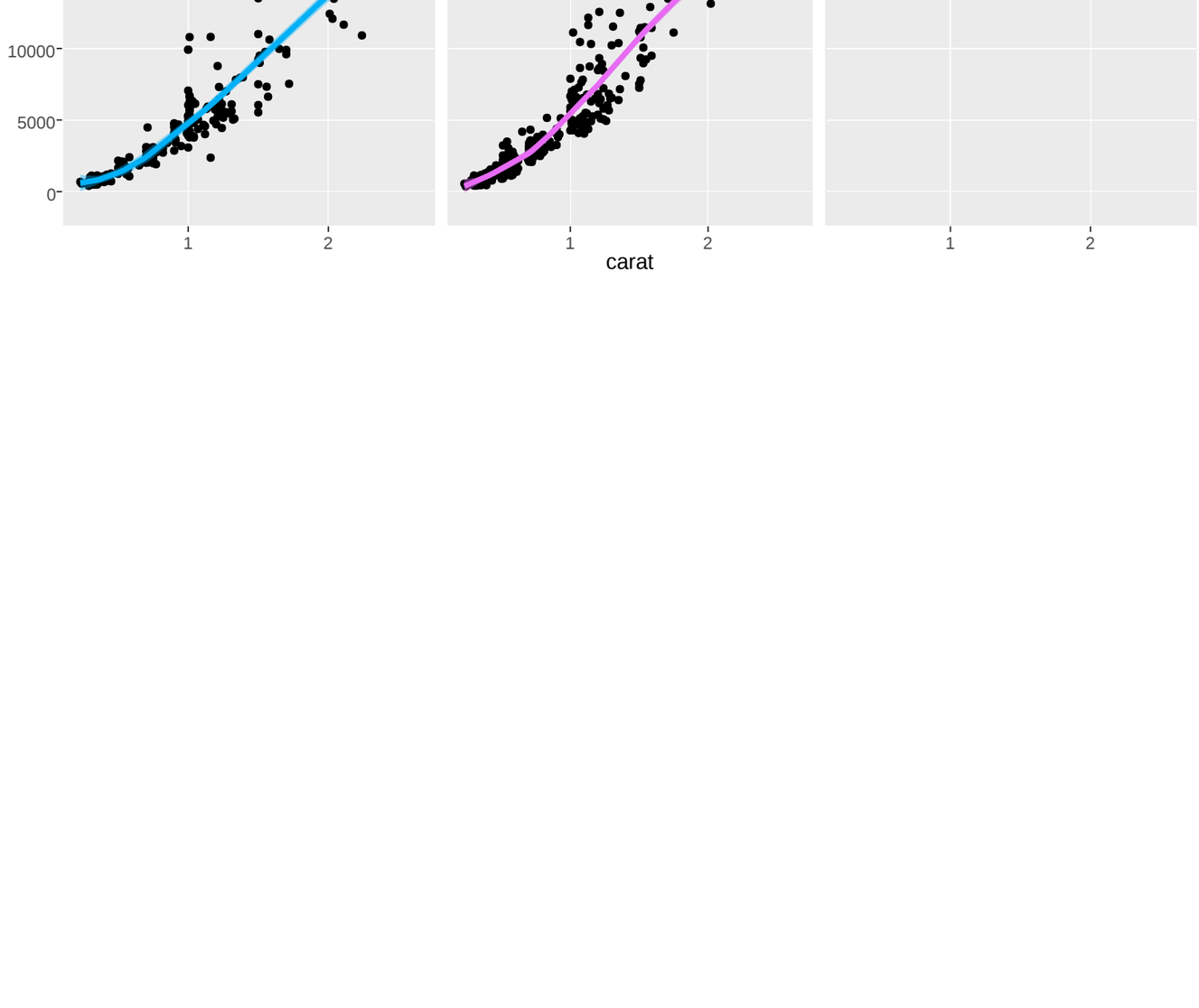
```
## Warning: Ignoring unknown aesthetics: text
```

```
ggplotly(plt)
```

```
## Warning: We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

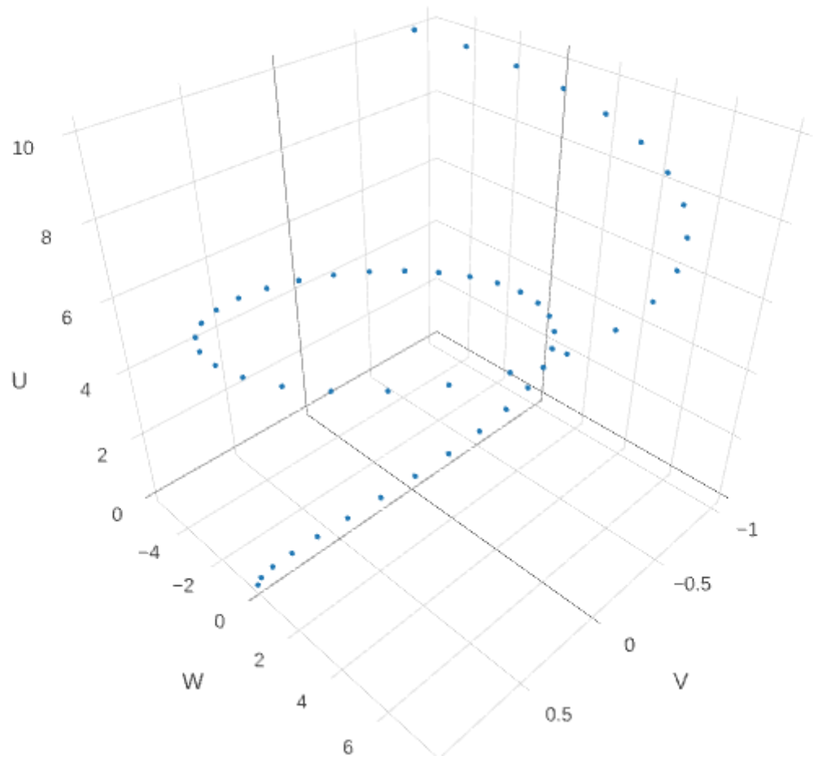
```
## `geom_smooth()` using method = 'loess'
```



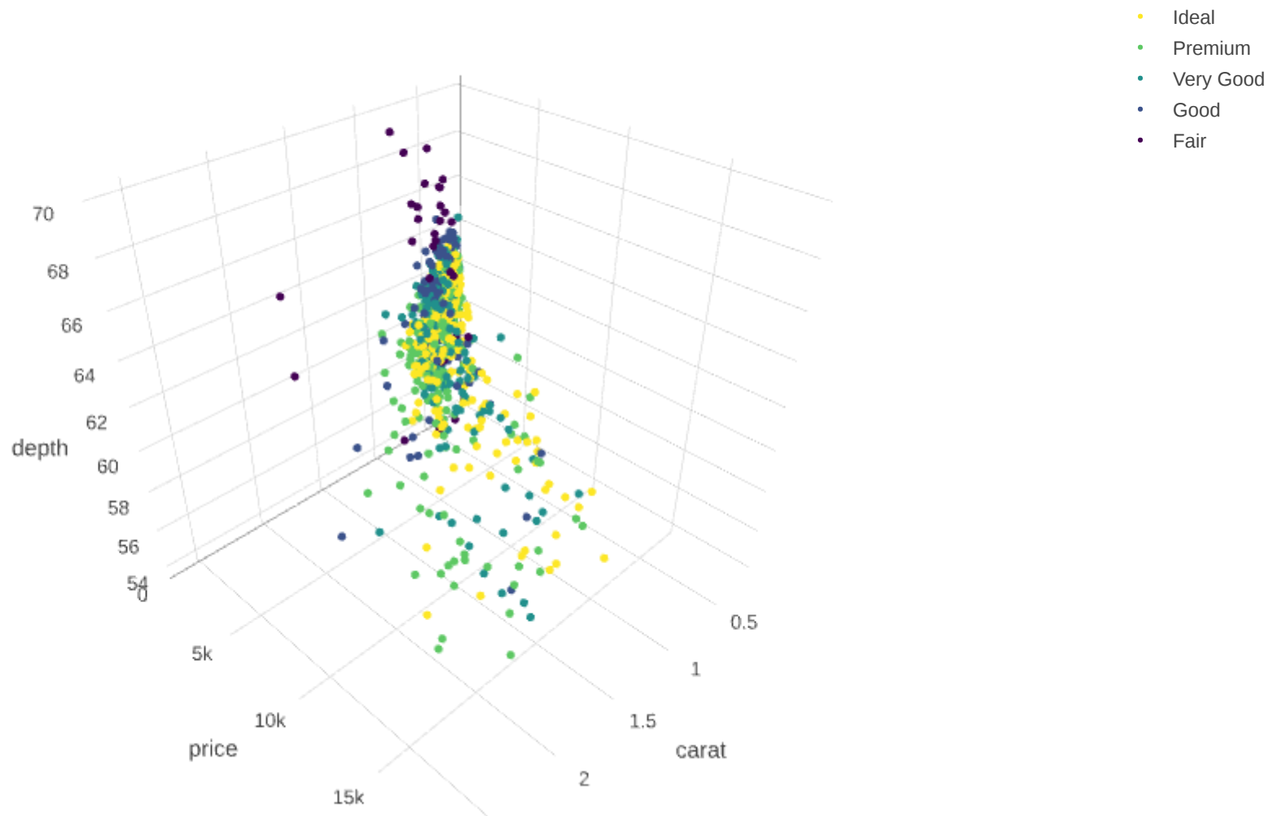

10000
5000
0
1
2
1
2
1
2
carat

3D Scatter plots

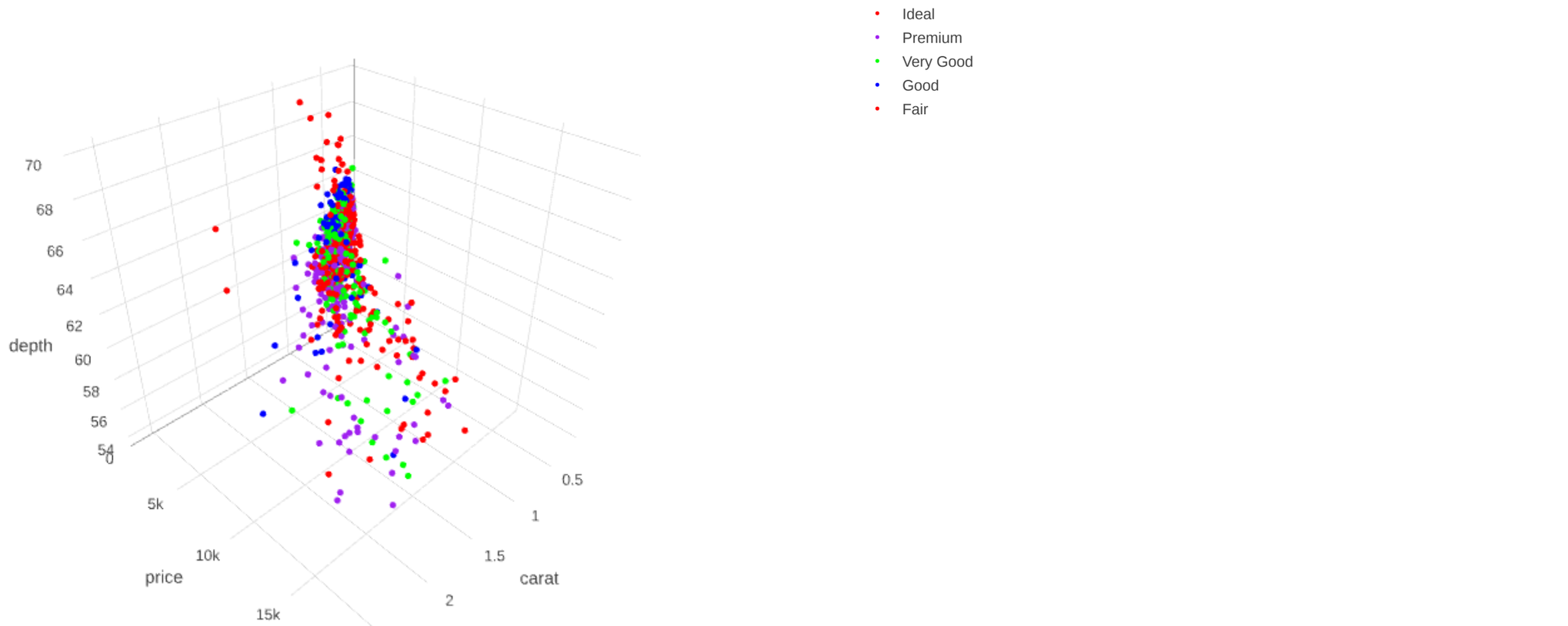
```
z <- seq(0, 10, 0.2)
df <- data.frame(U = seq(0, 10, 0.2), V = cos(z), W = sin(z)*z)
plot_ly(df, x = ~V, y = ~W, z = ~U, type = "scatter3d", mode = "markers",
marker = list(size = 2))
```



```
plot_ly(dsmall, x = ~carat, y = ~price, z = ~depth) %>%
add_markers(color = ~cut, text = ~paste("Clarity: ", clarity),
marker = list(size = 3))
```

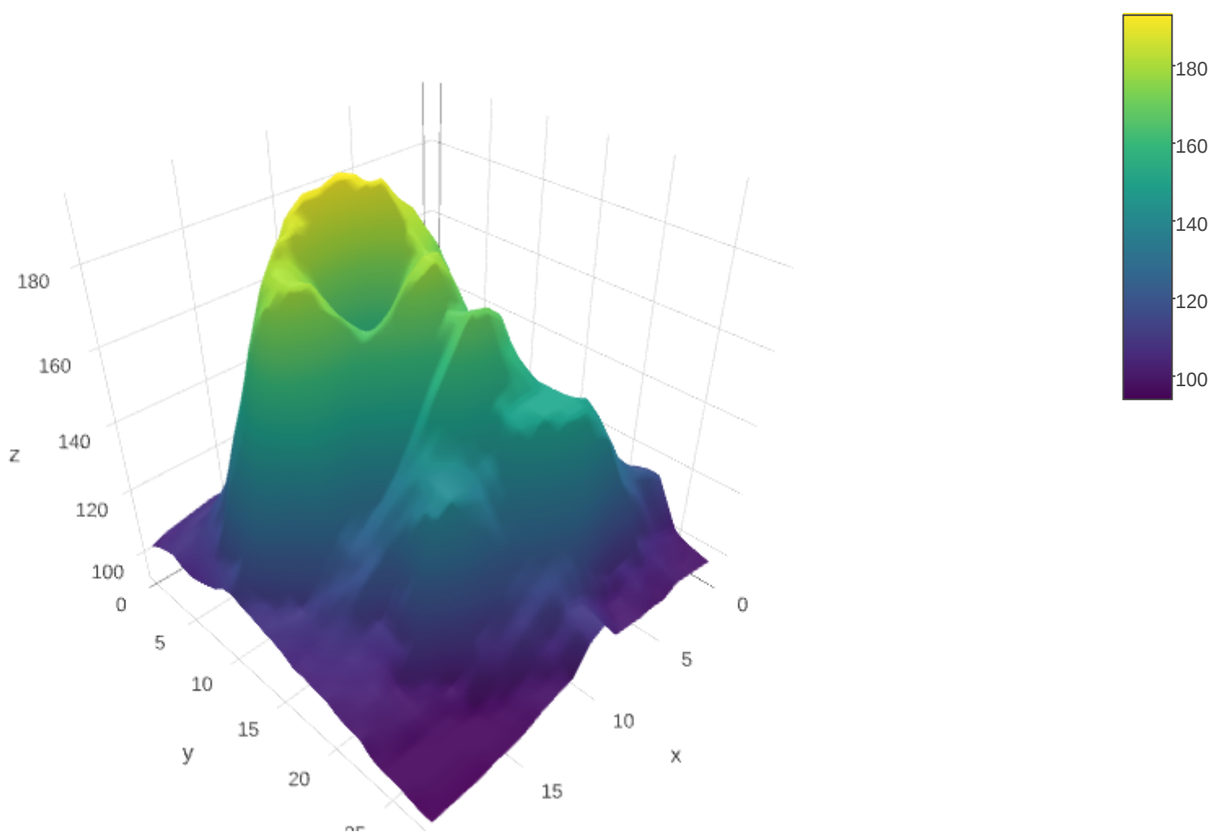


```
pal <- c("red", "blue", "green", "purple", "red")
plot_ly(dsmall, x = ~carat, y = ~price, z = ~depth) %>%
add_markers(color = ~cut, text = ~paste("Clarity: ", clarity),
marker = list(size = 3), colors = pal)
```



3D Surface plots

```r
# volcano is a numeric matrix that ships with R
p <- plot_ly(z = Volcano) %>% add_surface()
p
```



plotly account

- When you knit an R markdown document to an html, the **interactive plots will be included** in your html file.
- Additionally, you can **publish your charts to the web** with plotly's web service.

To open a plotly account:

1. Create a free plotly account
2. Save your authentication credentials

```
# Find your authentication API keys in your online settings. Set them in your R
session with:
Sys.setenv("plotly_username"="your_plotly_username")
Sys.setenv("plotly_api_key"="your_api_key")
# Save these commands in your .Rprofile file to be run everytime you start R.
```

3. Publish your graphs to plotly with `plotly_POST`

```
# filename sets the name of the file inside your online plotly account.
plotly_POST(p, filename = "surf3D-volcano")
```

Heatmaps

Heatmaps

- are a popular graphical method for **visualizing high-dimensional data** in a form of a table or a matrix,
- encode the information (variables, correlations, sparsity/missing data pattern) as a grid of colored cells.
- are used in many fields e.g. for gene expression,

> *The rows and the columns of the heatmaps are usually ordered to highlight the patterns in the data, and are usually accompanied by dendograms*[1].

Motor Trend Car Road Tests Dataset

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

```
?mtcars
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Scale the data

```
mtscaled <- as.matrix(scale(mtcars))
head(mtscaled)
```

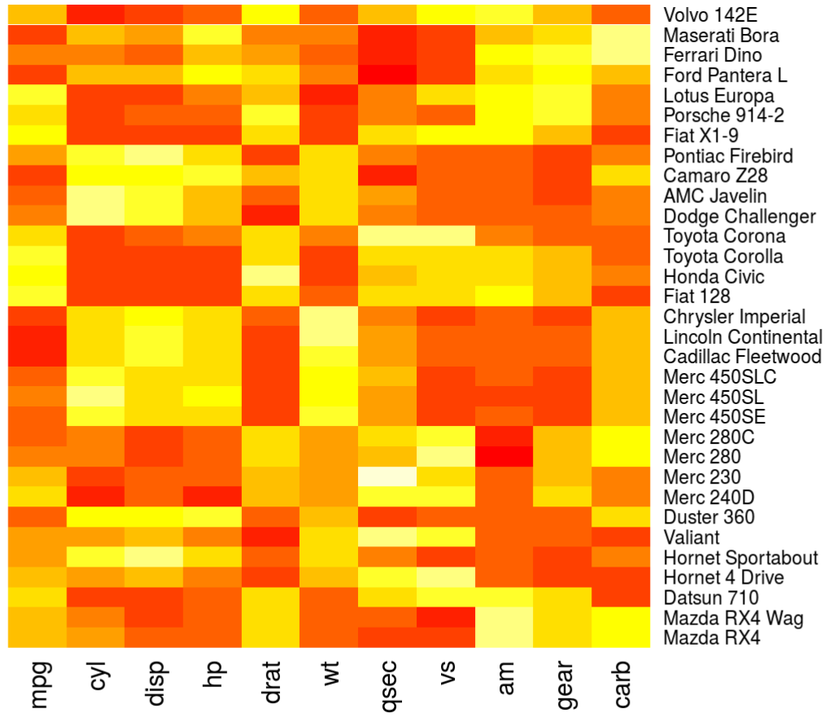
```
## mpg cyl disp hp drat
## Mazda RX4 0.1508848 -0.1049878 -0.57061982 -0.5350928 0.5675137
## Mazda RX4 Wag 0.1508848 -0.1049878 -0.57061982 -0.5350928 0.5675137
## Datsun 710 0.4495434 -1.2248578 -0.99018209 -0.7830405 0.4739996
## Hornet 4 Drive 0.2172534 -0.1049878 0.22009369 -0.5350928 -0.9661175
## Hornet Sportabout -0.2307345 1.0148821 1.04308123 0.4129422 -0.8351978
## Valiant -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078
## wt qsec vs am gear
## Mazda RX4 -0.610399567 -0.7771651 -0.8680278 1.1899014 0.4235542
## Mazda RX4 Wag -0.349785269 -0.4637808 -0.8680278 1.1899014 0.4235542
## Datsun 710 -0.917004624 0.4260068 1.1160357 1.1899014 0.4235542
## Hornet 4 Drive -0.002299538 0.8904872 1.1160357 -0.8141431 -0.9318192
## Hornet Sportabout 0.227654255 -0.4637808 -0.8680278 -0.8141431 -0.9318192
## Valiant 0.248094592 1.3269868 1.1160357 -0.8141431 -0.9318192
## carb
## Mazda RX4 0.7352031
## Mazda RX4 Wag 0.7352031
## Datsun 710 -1.1221521
```

```
## Datsun 710 -1.1221521

## Hornet 4 Drive -1.1221521
## Hornet Sportabout -0.5030337
## Valiant -1.1221521
```

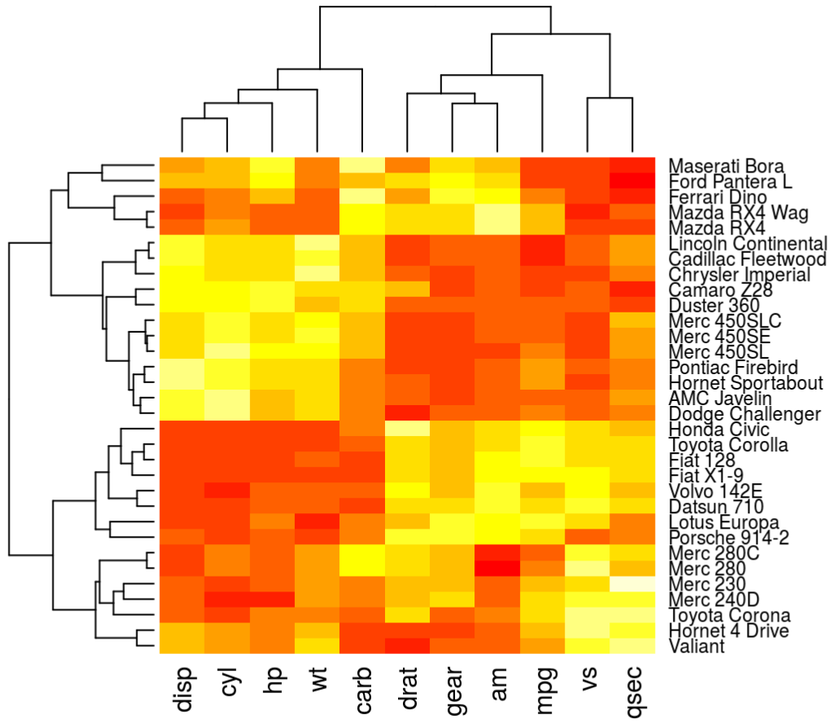
Generate a basic heatmap

```
# Using basic `stats::heatmap` function:
heatmap(mtscaled, Rowv = NA, Colv = NA)
```



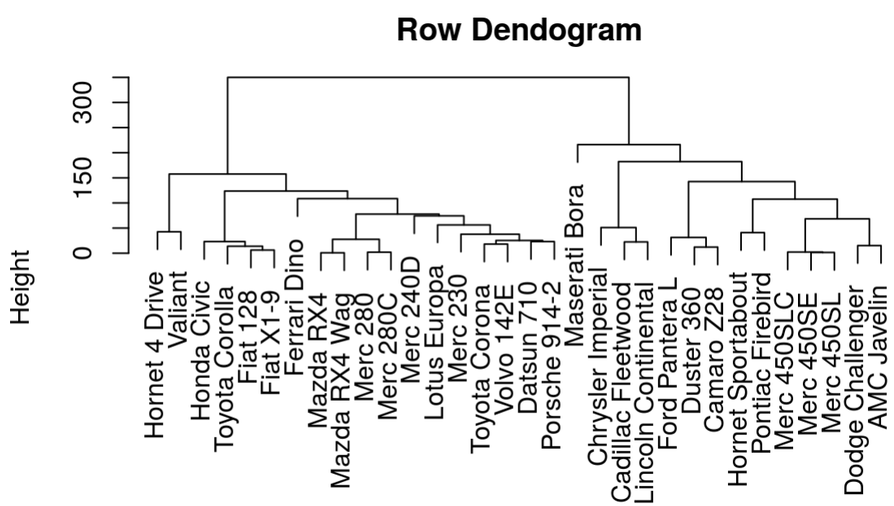
Heatmap with dendograms

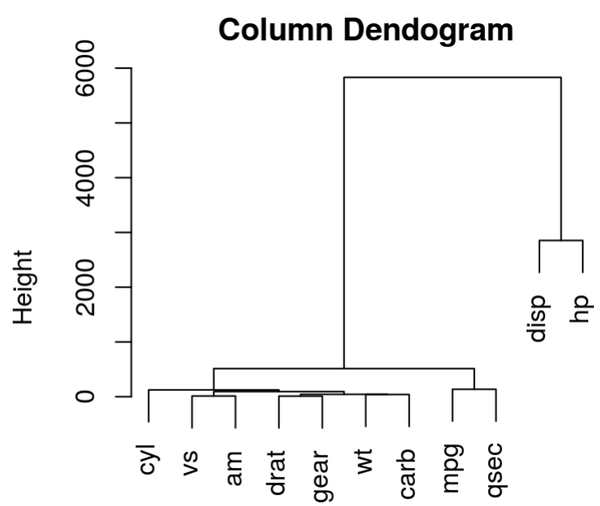
```
# heatmaps with dendograms and reordered (by means) rows and columns
heatmap(as.matrix(mtscaled))
```



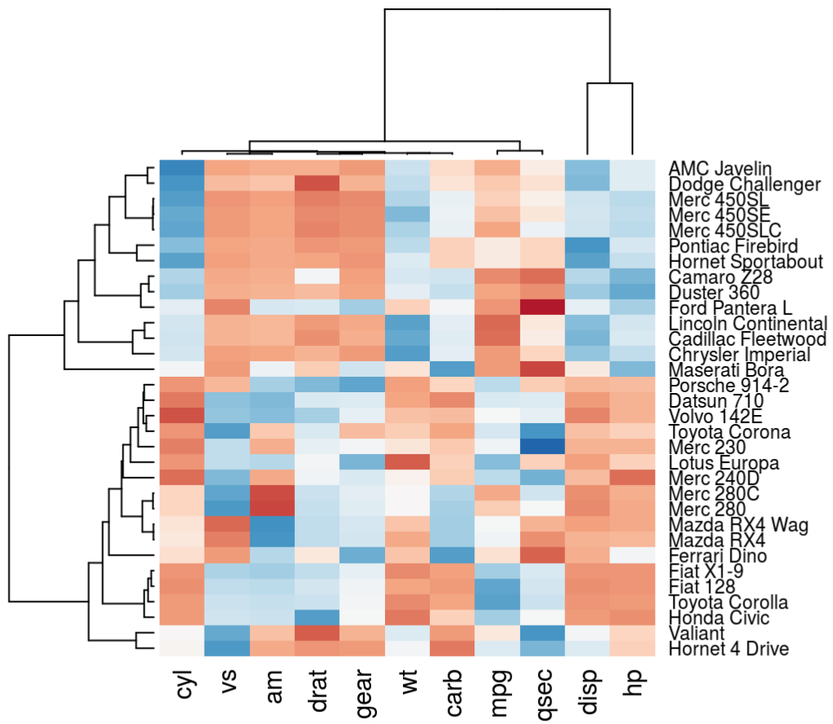
Compute dendograms

```r
# Cluster rows (by default healmaps uses method = "euclidean")
data.dist.r <- dist(mtcars, method = "manhattan")
row.clus <- hclust(data.dist.r, "aver")
# Cluster columns
data.dist.c <- dist(t(mtcars), method = "manhattan")
col.clus <- hclust(data.dist.c, "aver")
# Plot dendograms:
layout(matrix(c(1,2),nrow=1), widths=c(3,2)); par(mar = c(1,4,2,1))
plot(row.clus, xlab = "", main = "Row Dendogram", sub = "")
plot(col.clus, xlab = "", main = "Column Dendogram", sub = "")
```



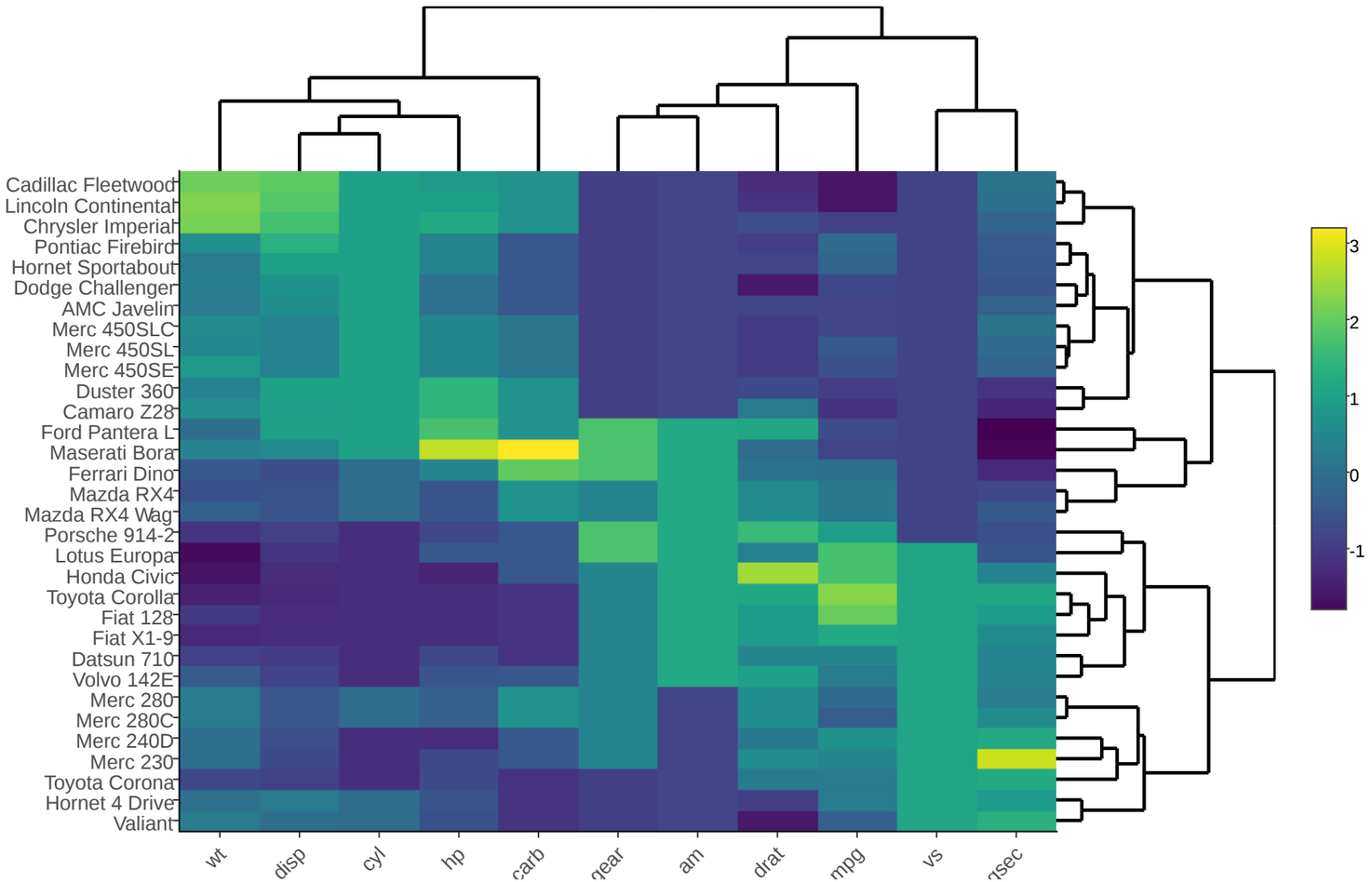


```r
library(RColorBrewer)
# Define color scheme
scaleredblues <- colorRampPalette(colors = brewer.pal(9, "RdBu"))(100)
heatmap(as.matrix(mtscaled), col = scaleredblues,
Rowv = as.dendrogram(row.clus),
Colv = as.dendrogram(col.clus))
```

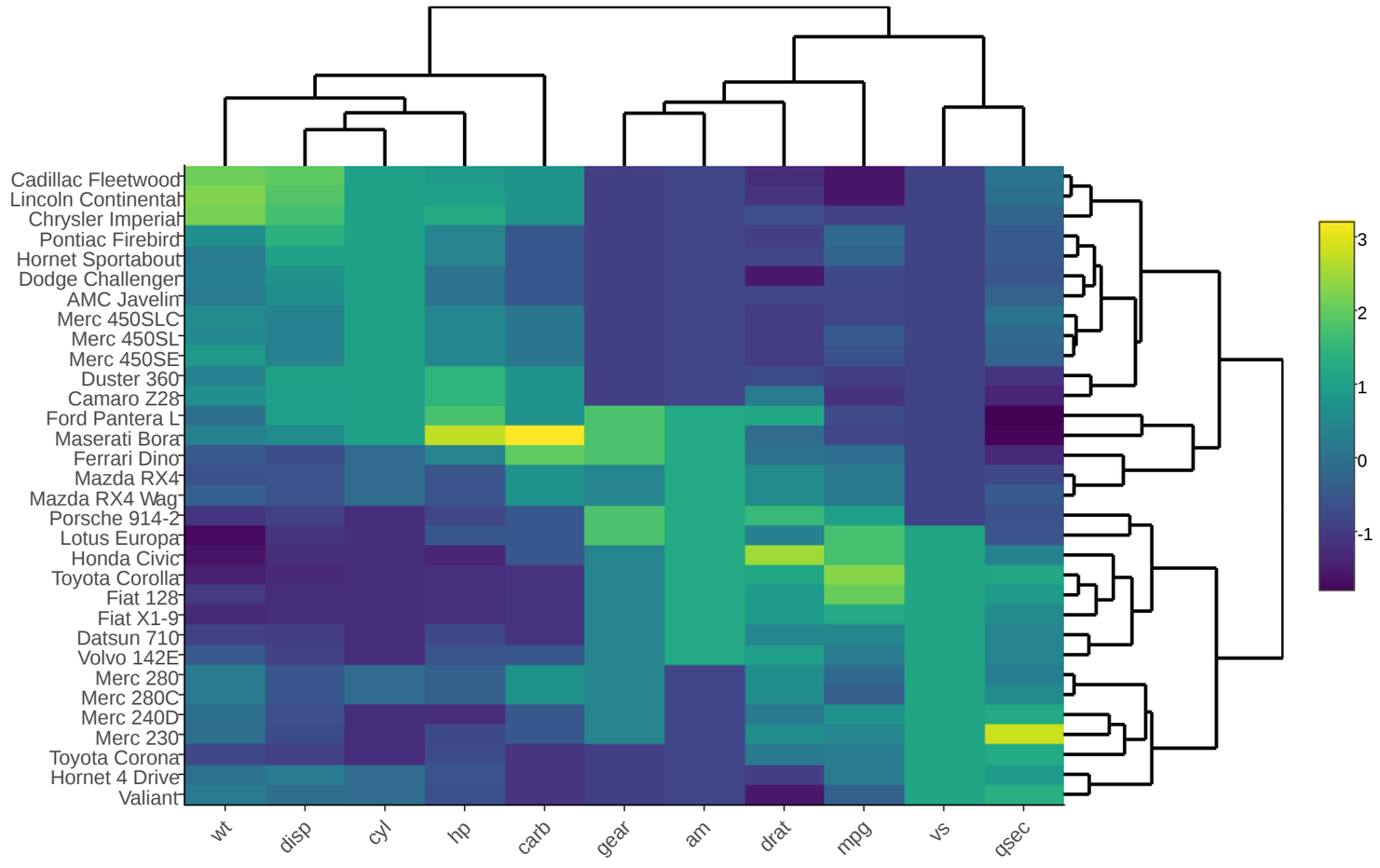


heatmaply package for interactive heatmap

```
library(heatmaply)
heatmaply(mtscaled, dendogram = "both")
```

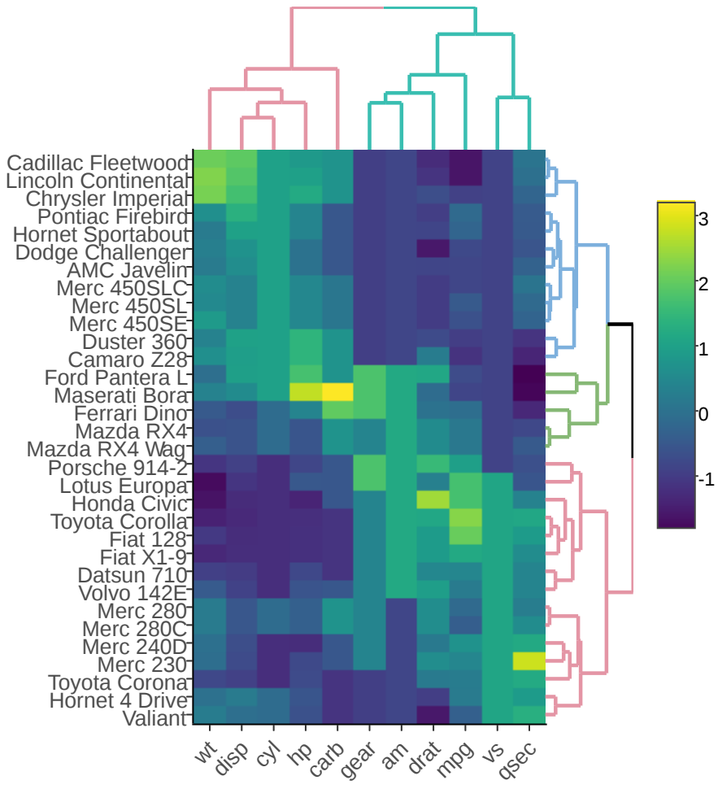


```
heatmaply(mtscaled) %>% layout(margin = list(l = 150, b = 50))
```



Color the dendogram branches:

```
heatmaply(mtscaled, k_col = 2, k_row = 3) %>%
layout(margin = list(l = 150, b = 160),
autosize = F, width = 500, height = 600)
```



Example: Gene expression

Heatmap of the Yeast cell cycle data by Spellman et al.

```
# read the data in from URL
bots <- read.table(url("http://genome-
www.stanford.edu/cellcycle/data/rawdata/combined.txt"), sep="\t", header=TRUE)

# get just the alpha data
abot <- bots[,c(8:25)]
rownames(abot) <- bots[,1]
abot[1:7, 1:7]
```

```
## alpha0 alpha7 alpha14 alpha21 alpha28 alpha35 alpha42
## YAL001C -0.15 -0.15 -0.21 0.17 -0.42 -0.44 -0.15
## YAL002W -0.11 0.10 0.01 0.06 0.04 -0.26 0.04
## YAL003W -0.14 -0.71 0.10 -0.32 -0.40 -0.58 0.11
## YAL004W -0.02 -0.48 -0.11 0.12 -0.03 0.19 0.13
## YAL005C -0.05 -0.53 -0.47 -0.06 0.11 -0.07 0.25
## YAL007C -0.60 -0.45 -0.13 0.35 -0.01 0.49 0.18
## YAL008W -0.28 -0.22 -0.06 0.22 0.25 0.13 0.34
```

Process the data

```
# get rid of NAs
abot[is.na(abot)] <- 0
dim(abot)
```

```
## [1] 6178 18
```

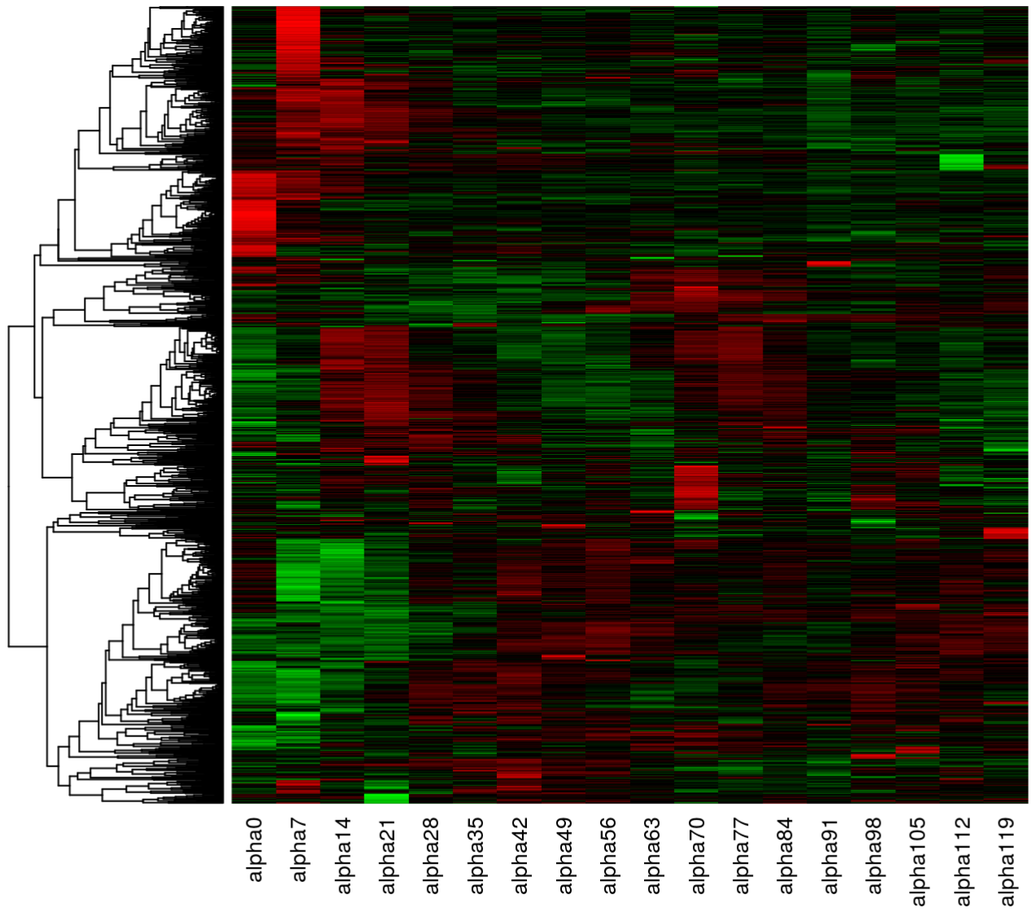
We need to reduce the data.

Sort on max difference and take first 1000.

```
min <-apply(abot, 1, min)
max <- apply(abot, 1, max)
sabot <- abot[order(max - min, decreasing=TRUE),][1:1000,]

# cluster on correlation
cdist <- as.dist(1 - cor(t(sabot)))
hc <- hclust(cdist, "average")
```

```
# install.packages("gplots")
heatmap(as.matrix(sabot),col=gplots::greenred(80),
Rowv=as.dendrogram(hc), Colv=NA, labRow = "")
```



GGMAP

Plotting Maps with ggmap

```
library(ggmap)
us <- c(left = -125, right = -67, bottom = 25.75, top = 49)
# Retrieve maps from http://maps.stamen.com/ with get_stamenmap()
us_map <- get_stamenmap(bbox = us, zoom = 5, maptype = "toner-lite")
class(us_map)
```

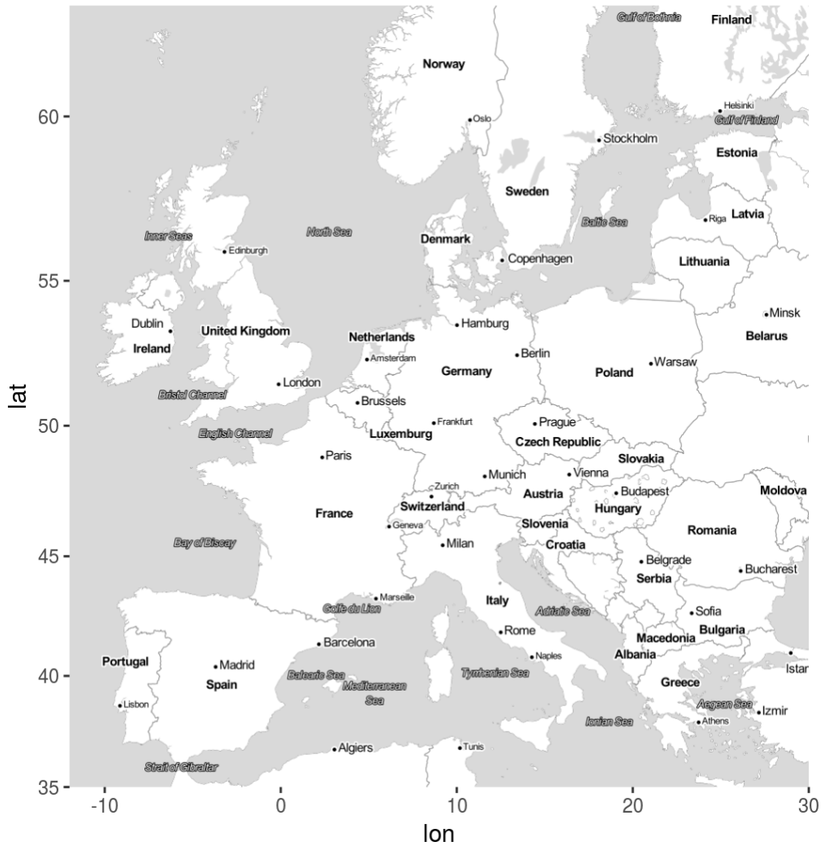
```
## [1] "ggmap" "raster"
```

```
# maptype = c("terrain", "terrain-background",
#"terrain-labels", "terrain-lines",
#"toner", "toner-2010", "toner-2011",
# "toner-background", "toner-hybrid",
# "toner-labels", "toner-lines",
# "toner-lite", "watercolor")
```

```
ggmap(us_map)
```

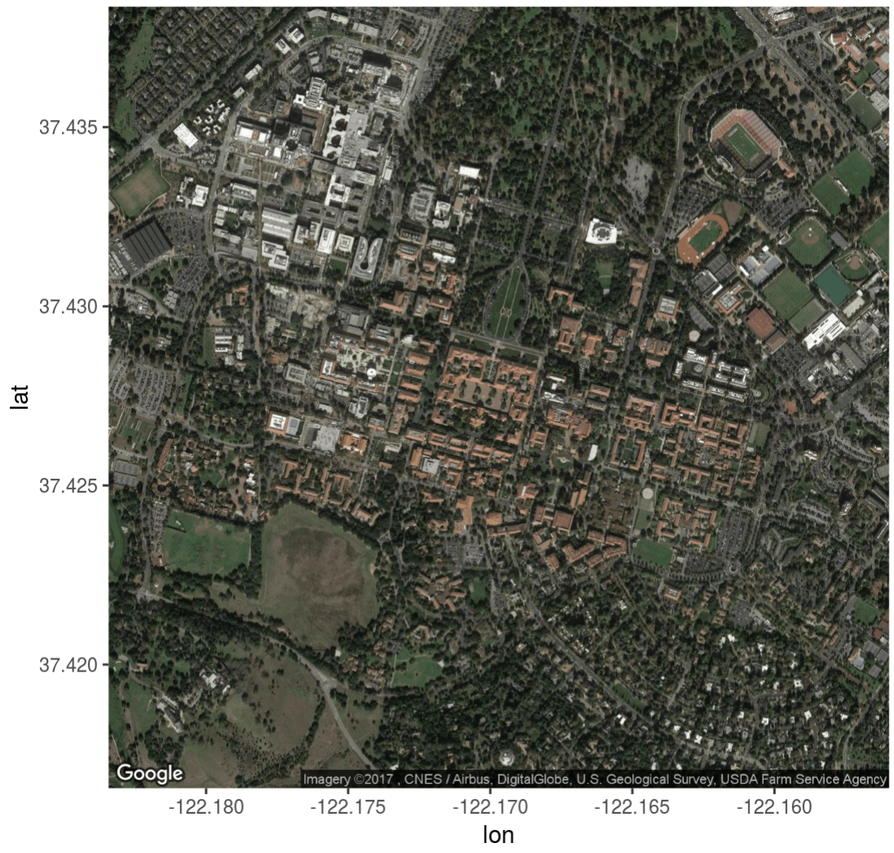

Seattle
WA
Portland
OR
MT
ND
MN
SD
ID
WY
NE
NV
UT
CO
KS
Salt Lake City
Denver
Kansas City
Minneapolis
WI
MI
IA
IL
IN
OH
Milwaukee
Detroit
Chicago
Cleveland
Pittsburgh
Indianapolis
Columbus
Cincinnati
St. Louis
MO
KY
WV
MD
Baltimore
Washington
Chesapeake Bay
VA
Québec
Montréal
Ottawa
Toronto
NY
PA
New York
Philadelphia
ME
VT
NH
CT
Boston
Francisco
San Jose
CA
Las Vegas
AZ
Albuquerque
NM
OK
TX
AR
LA
MS
AL
TN
Nashville
Memphis
NC
Charlotte
SC
Atlanta
GA
FL
Tampa
Miami
New Orleans
Houston
Dallas
San Antonio
Los Angeles
Phoenix
San Diego
Puebla
Tijuana
Juarez
Chihuahua
Gulf of California
45
40
35
30
lat
-120
-110
-100
-90
-80
-70
lon

```
europe <- c(left = -12, right = 30, bottom = 35.0, top = 63)
europe_map <- get_stamenmap(bbox = europe, zoom = 5, maptype = "toner-lite")
ggmap(europe_map)
```



```
stanford_map <- get_googlemap("Stanford", zoom = 15)
ggmap(stanford_map)
```

```
stanford_map <- get_googlemap("Stanford", zoom = 15, maptype = "satellite")
ggmap(stanford_map)
```



1. https://www.r-statistics.com/2016/05/heatmaply-interactive-heat-maps/#more-61399↩