

# **Lecture 5: 3D and Interactive graphics**

October 17, 2017

## Packages:

- **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("ggplot2", "plotly", "heatmaply", "ggmap", "dplyr")  
lapply(.packages, install.packages, dependencies = TRUE)
```

**plotly**

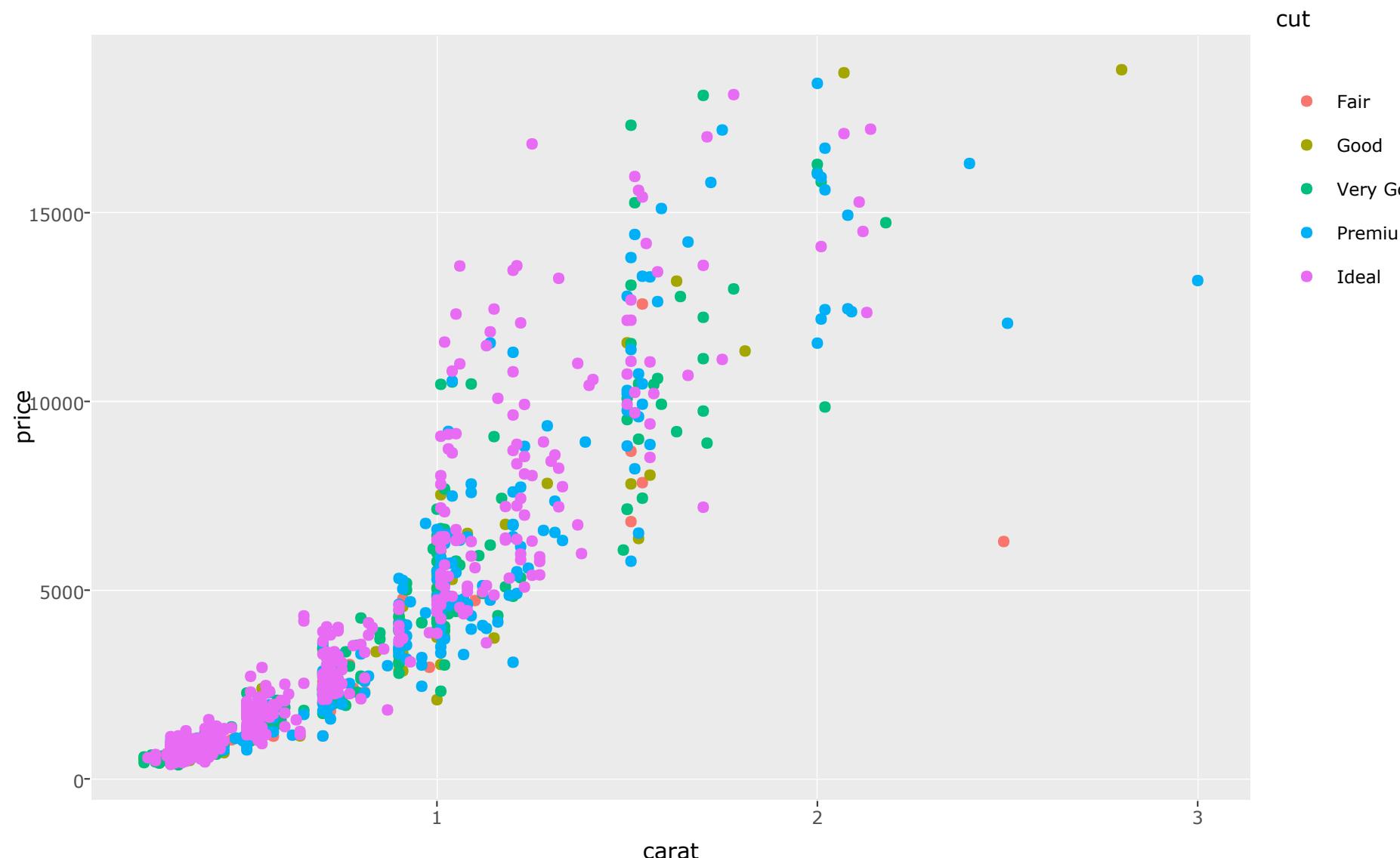
# plotly package

- **plotly is a package for visualization and a collaboration platform for 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](#)

# plotly integration with ggplot2

```
library(plotly); library(ggplot2); library(dplyr)
plt <- ggplot(diamonds %>% sample_n(1000), aes(x = carat, y = price)) +
  geom_point(aes(color = cut))
ggplotly(plt)
```

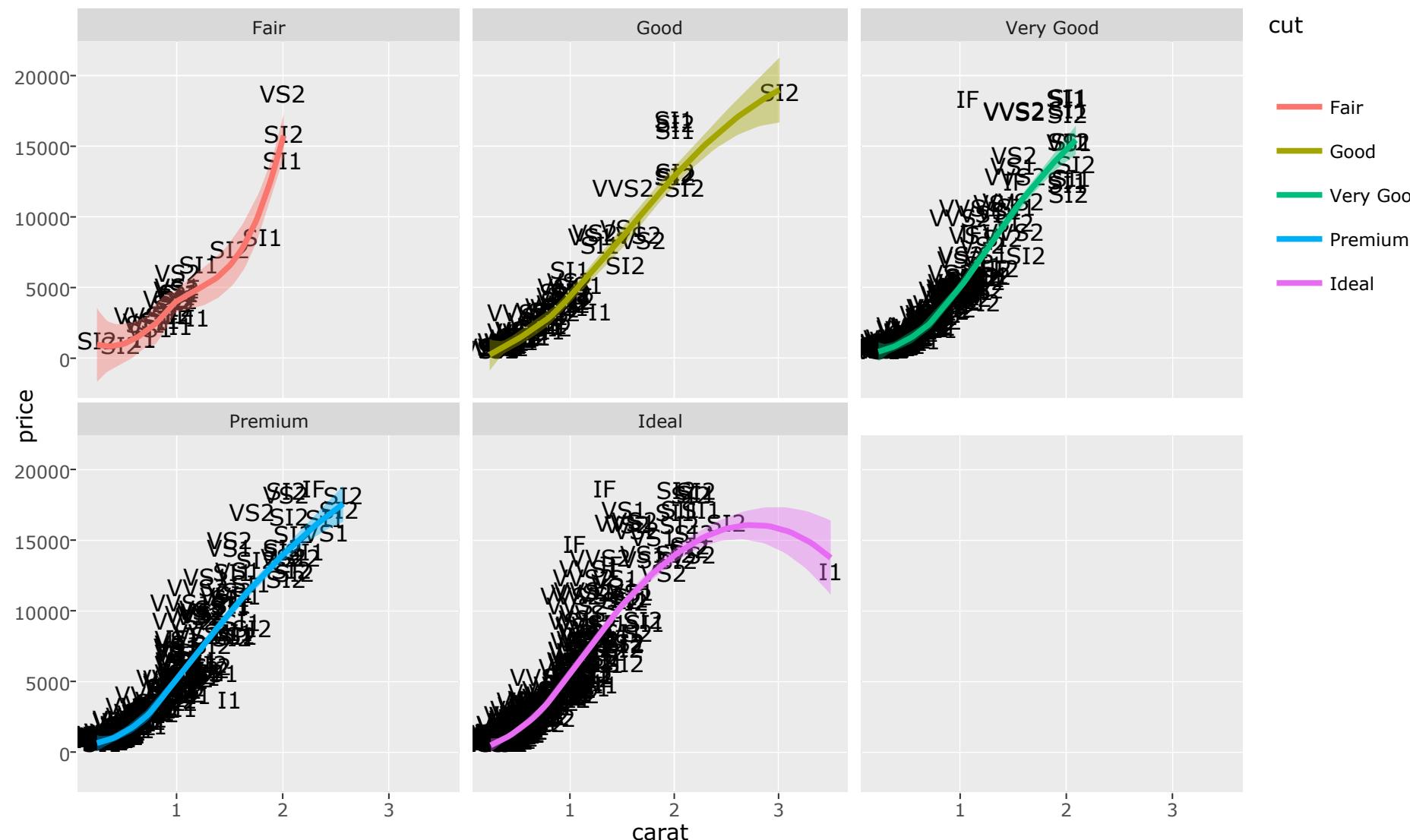
```
## Warning: We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```



```
plt <- ggplot(diamonds %>% sample_n(1000), aes(x = carat, y = price)) +  
  geom_text(aes(label = clarity), size = 4) +  
  geom_smooth(aes(color = cut, fill = cut)) +  
  facet_wrap(~cut)  
ggplotly(plt)
```

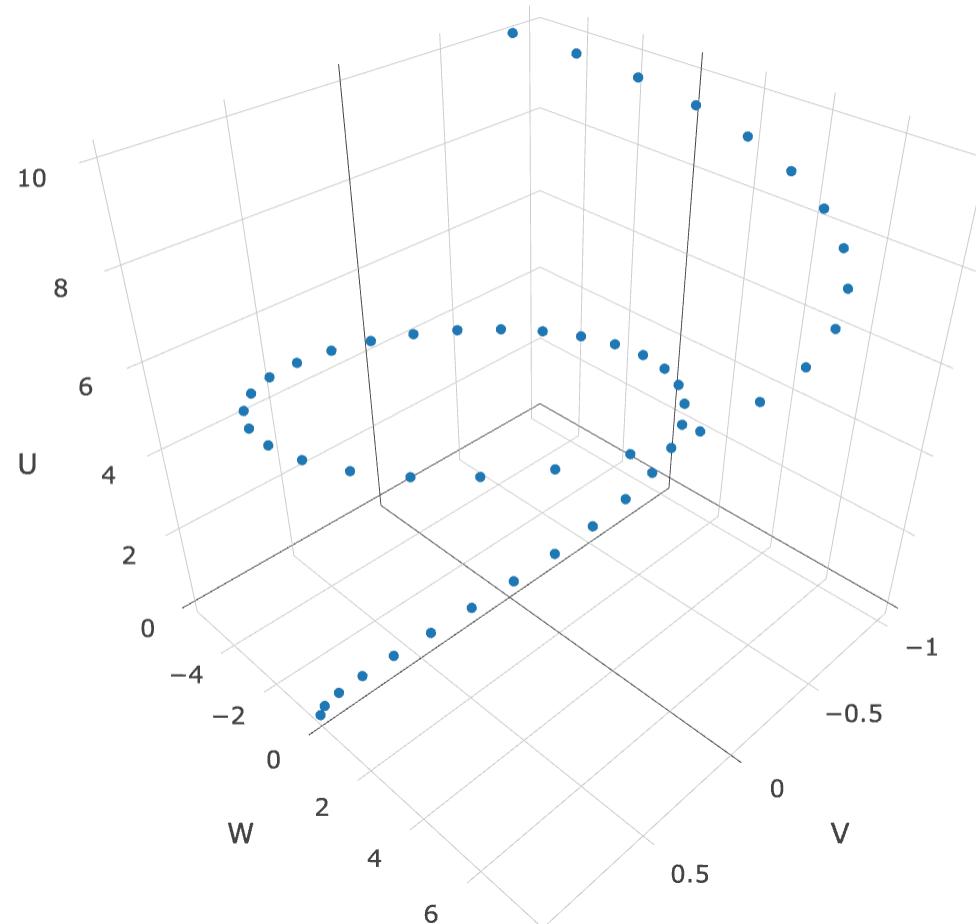
```
## Warning: We recommend that you use the dev version of ggplot2 with `ggplotly()  
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

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

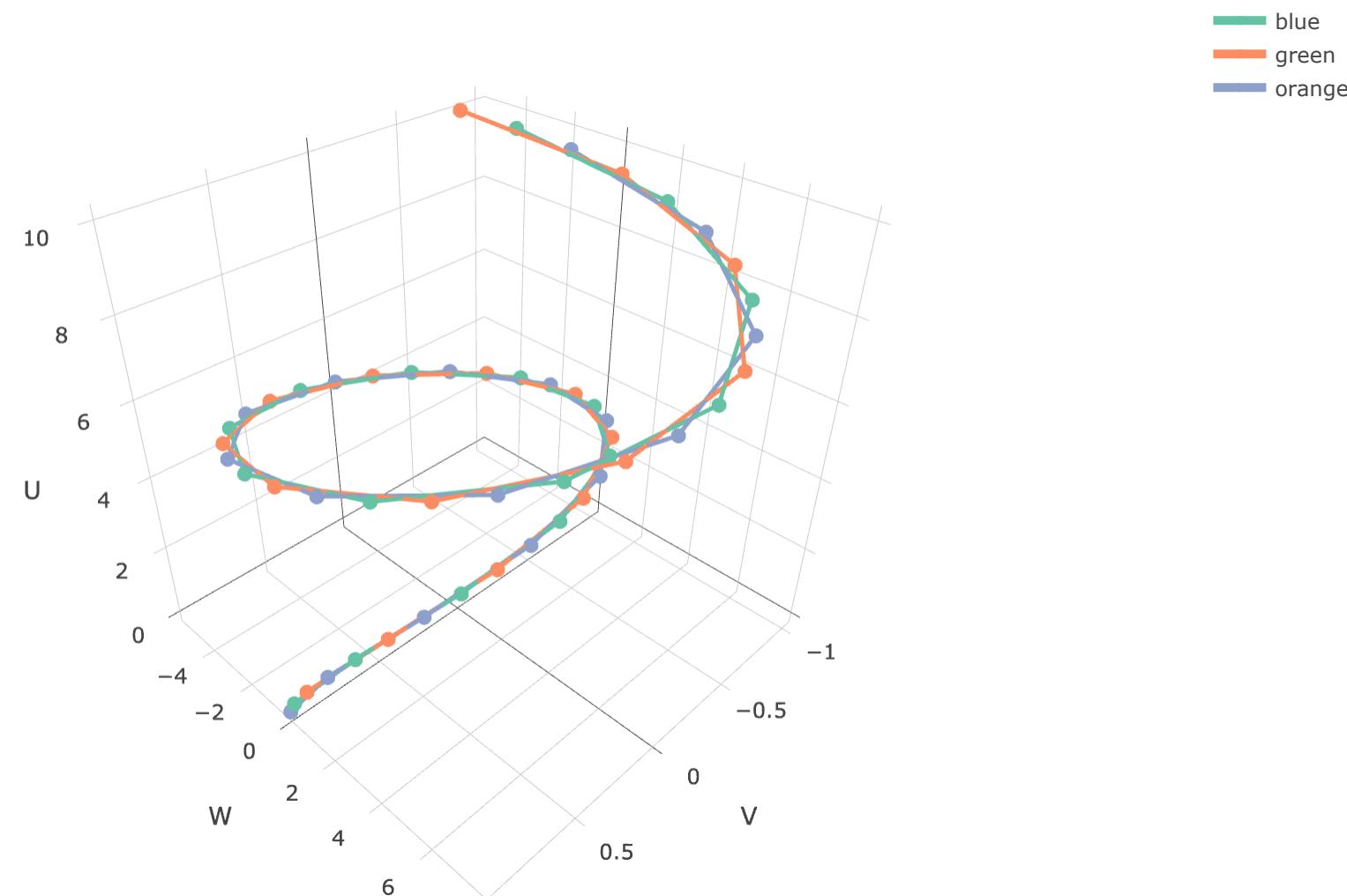


# 3D Scatter plots

```
theta <- seq(0, 10, 0.2);
df <- data.frame(U = theta, V = cos(theta), W = sin(theta)*theta)
plot_ly(df, x = ~V, y = ~W, z = ~U, type = "scatter3d", mode = "markers",
        marker = list(size = 3))
```

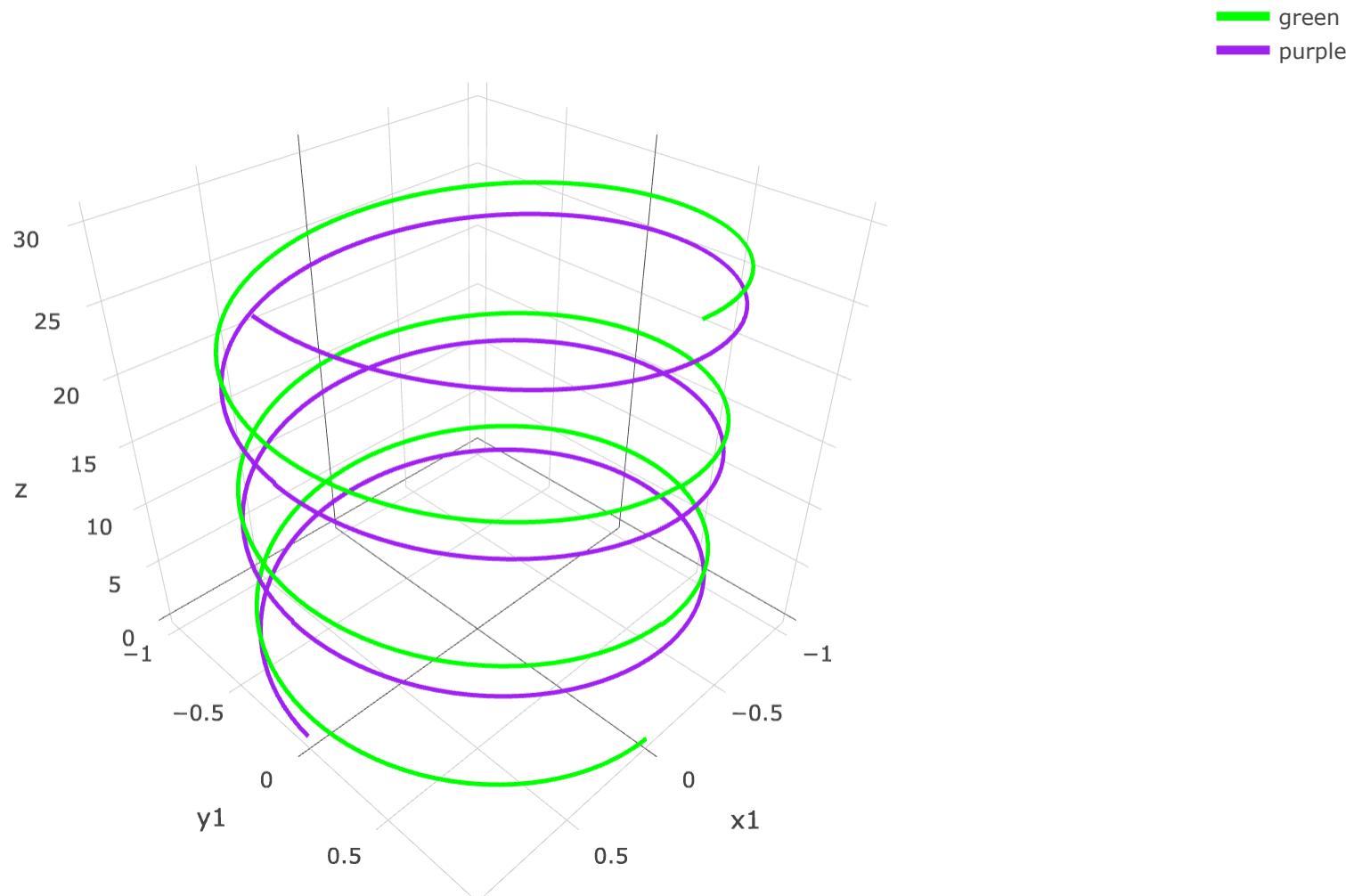


```
df$cols <- rep_len(c("orange", "blue", "green"), length.out = length(theta))
(plt <- plot_ly(df, x = ~V, y = ~W, z = ~U, color = ~cols,
type = "scatter3d", mode = "markers+lines",
marker = list(size = 5), line = list(width = 5)))
```



# Adding layers

```
dbl.helix <- data.frame(t = rep(seq(0, 2*pi, length.out = 100), 3)) %>%
  mutate(x1 = sin(t), y1 = cos(t), z = (1:length(t))/10,
        x2 = sin(t + pi/2), y2 = cos(t + pi/2))
plot_ly(dbl.helix, x = ~x1, y = ~y1, z = ~z, type = "scatter3d", mode = "lines",
        color = "green", colors = c("green", "purple"), line = list(width = 5)) %
  add_trace(x = ~x2, y = ~y2, z = ~z+0.2, color = "purple")
```



# Volcano dataset

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

```
dim(volcano)
```

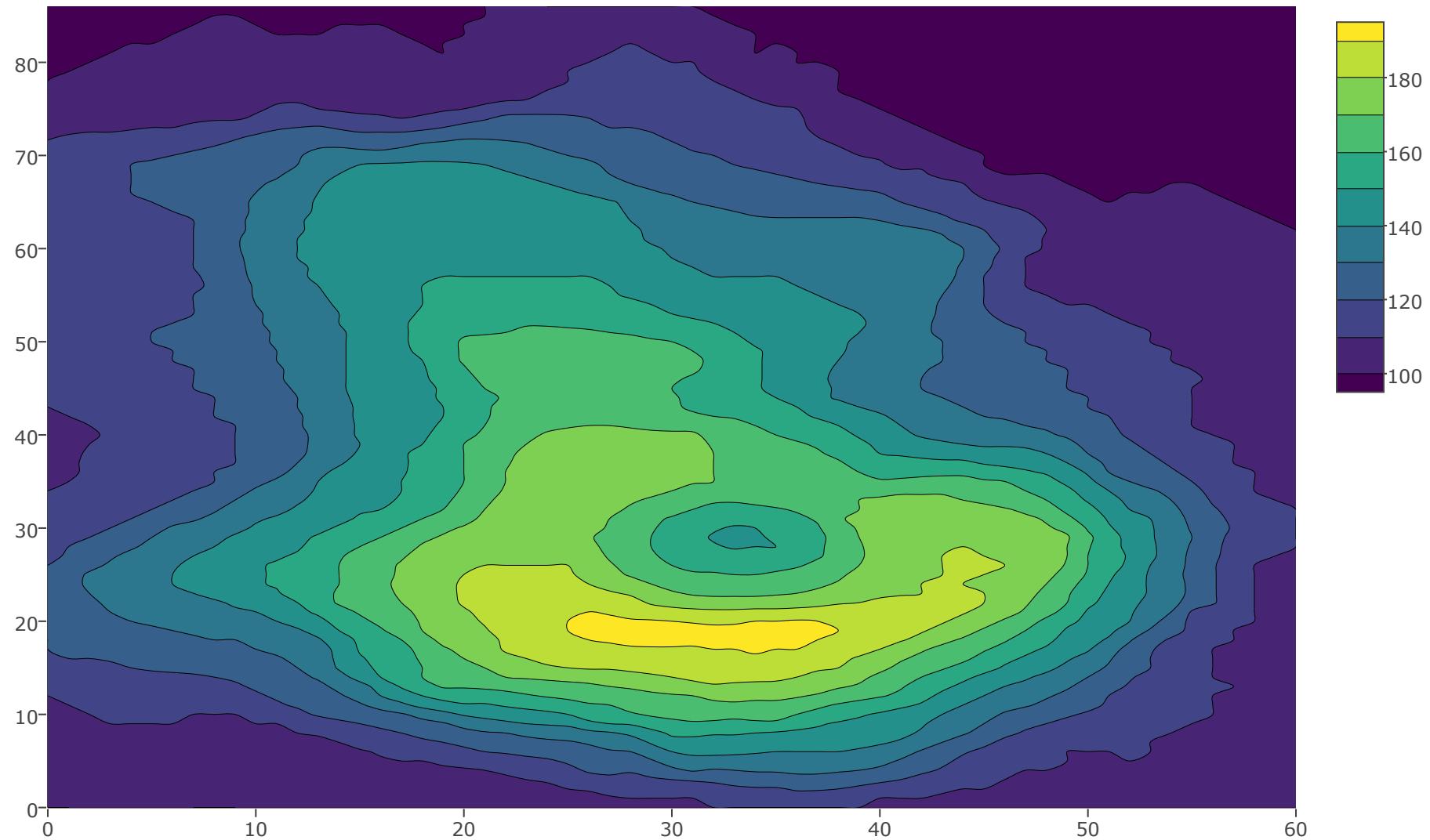
```
## [1] 87 61
```

```
volcano[1:5, 1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 100  100  101  101  101
## [2,] 101  101  102  102  102
## [3,] 102  102  103  103  103
## [4,] 103  103  104  104  104
## [5,] 104  104  105  105  105
```

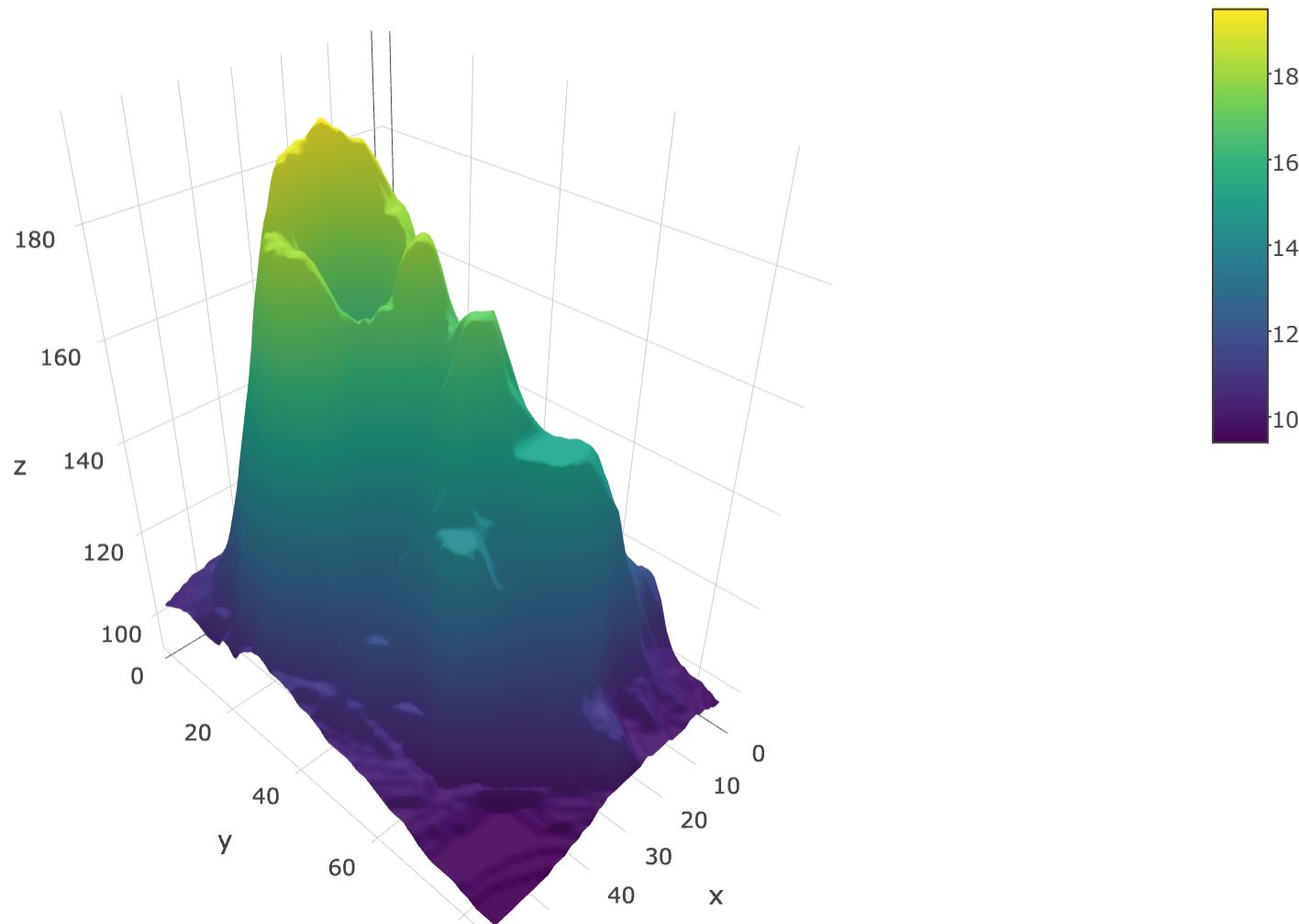
# 2D contour plots

```
plot_ly(z = volcano) %>% add_contour()
```



# 3D surface plots

```
plot_ly(z = volcano) %>% add_surface()
```



# Heatmaps

# Heatmap

- A *heatmap* is a popular graphical method for **visualizing high-dimensional data (table or matrix format)**
- Heatmaps display data (variables, correlations, sparsity/missing data pattern) as a grid of colored cells.
- They are commonly used in genomics papers to show **gene expression levels**,

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

# Scale mtcars data

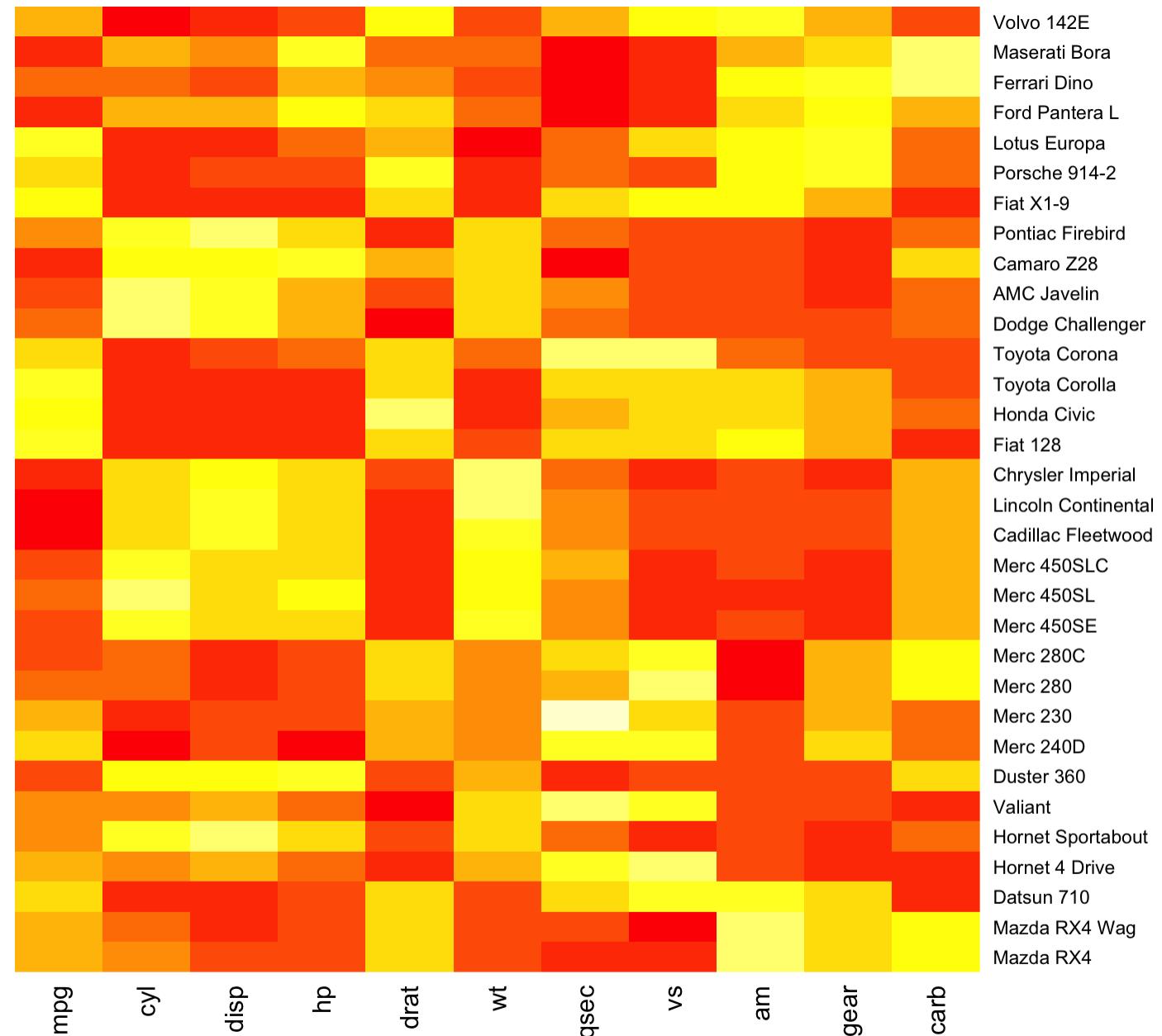
We convert each variable (column) in `mtcars` to Z-scores, i.e. we subtract the column mean and scale by its standard deviation.

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

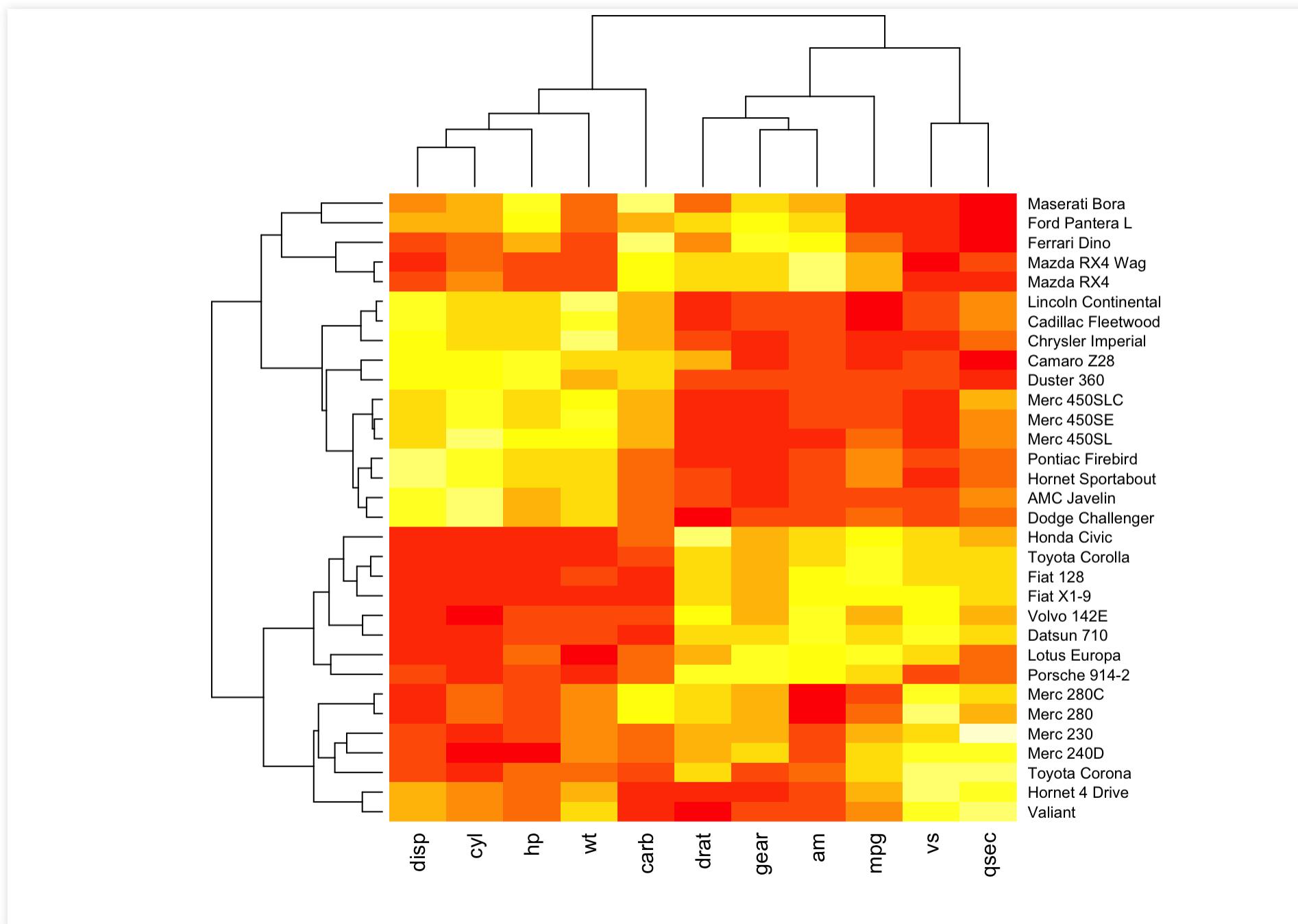
# Generate a basic heatmap

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



# Heatmap with dendograms

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

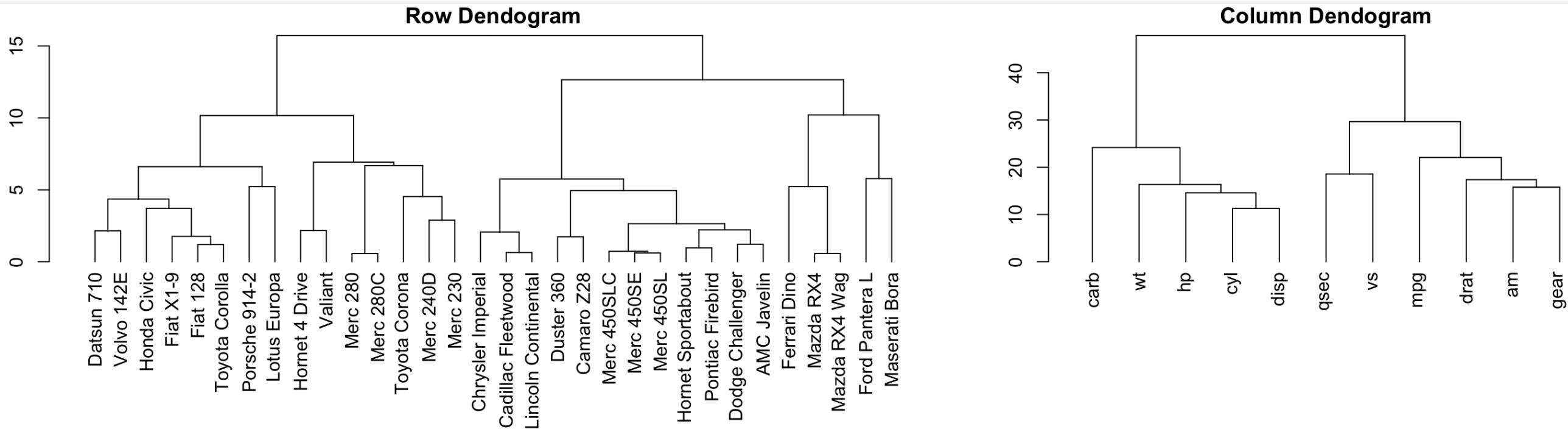


# Compute dendograms

```
# Cluster rows (by default healmaps uses method = "euclidean")
row.clus <- hclust(dist(mtscaled, method = "manhattan"), "aver")
row.dend <- as.dendrogram(row.clus)

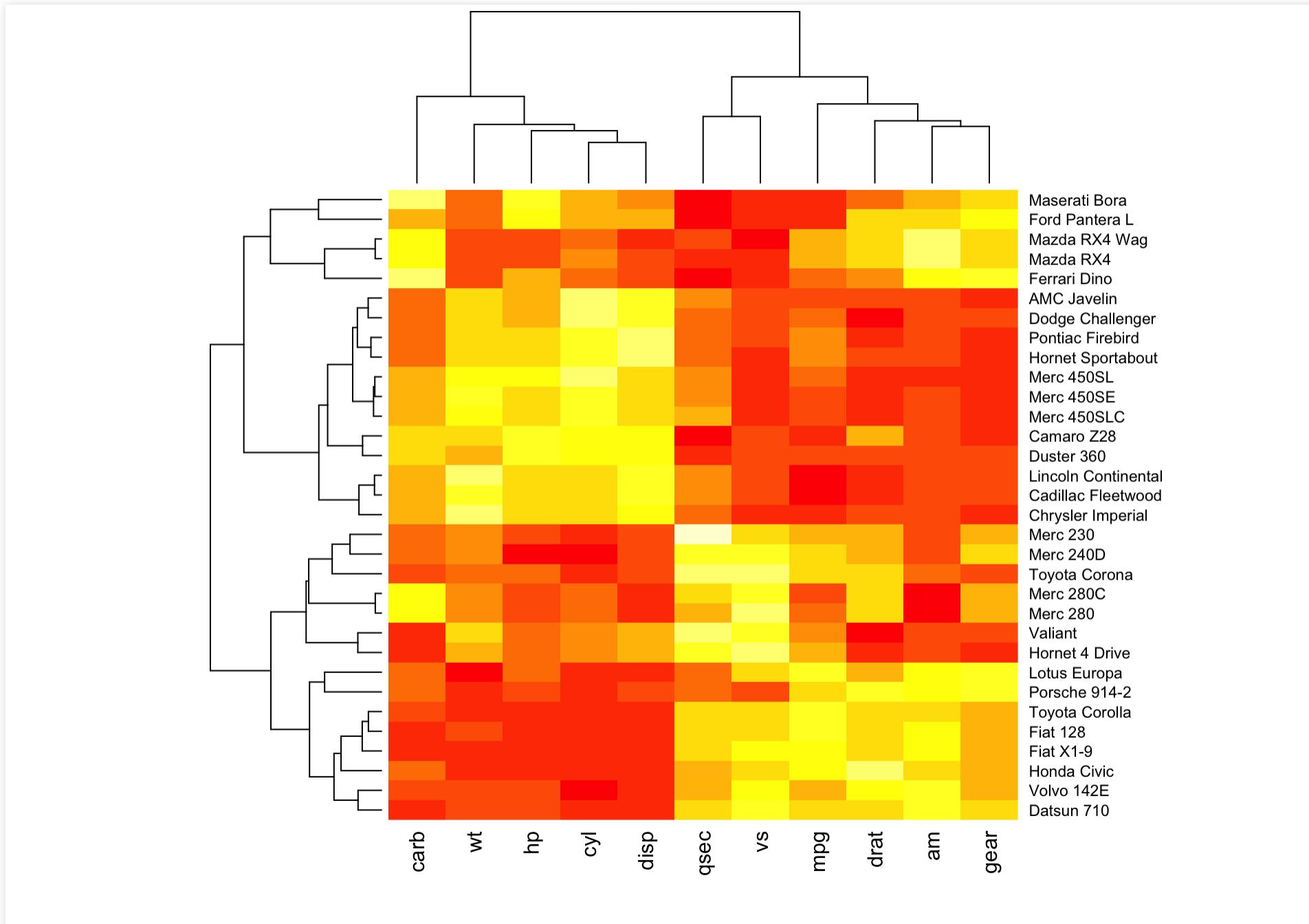
# Cluster columns
col.clus <- hclust(dist(t(mtscaled)), method = "manhattan"), "aver")
col.dend <- as.dendrogram(col.clus)

# Plot dendograms:
layout(matrix(c(1,2), nrow=1), widths=c(3,2)); par(mar = c(8,4,1,1))
plot(row.dend, xlab = "", main = "Row Dendogram", sub = "")
plot(col.dend, xlab = "", main = "Column Dendogram", sub = "")
```

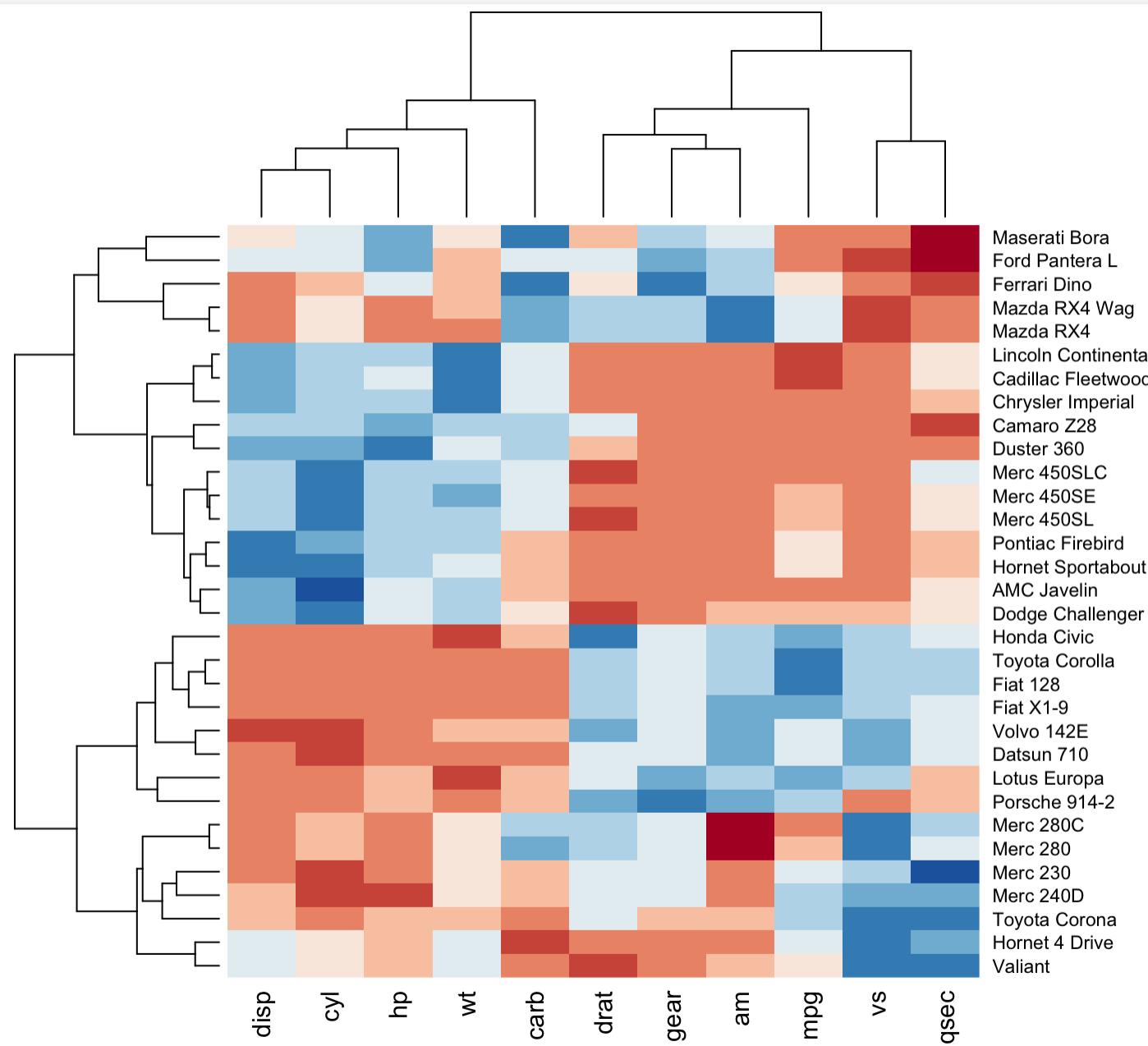


# Heatmaps with customized dendograms

```
# heatmaps with dendograms and reordered (by means) rows and columns  
heatmap(mtscaled, Rowv = row.dend, Colv = col.dend)
```

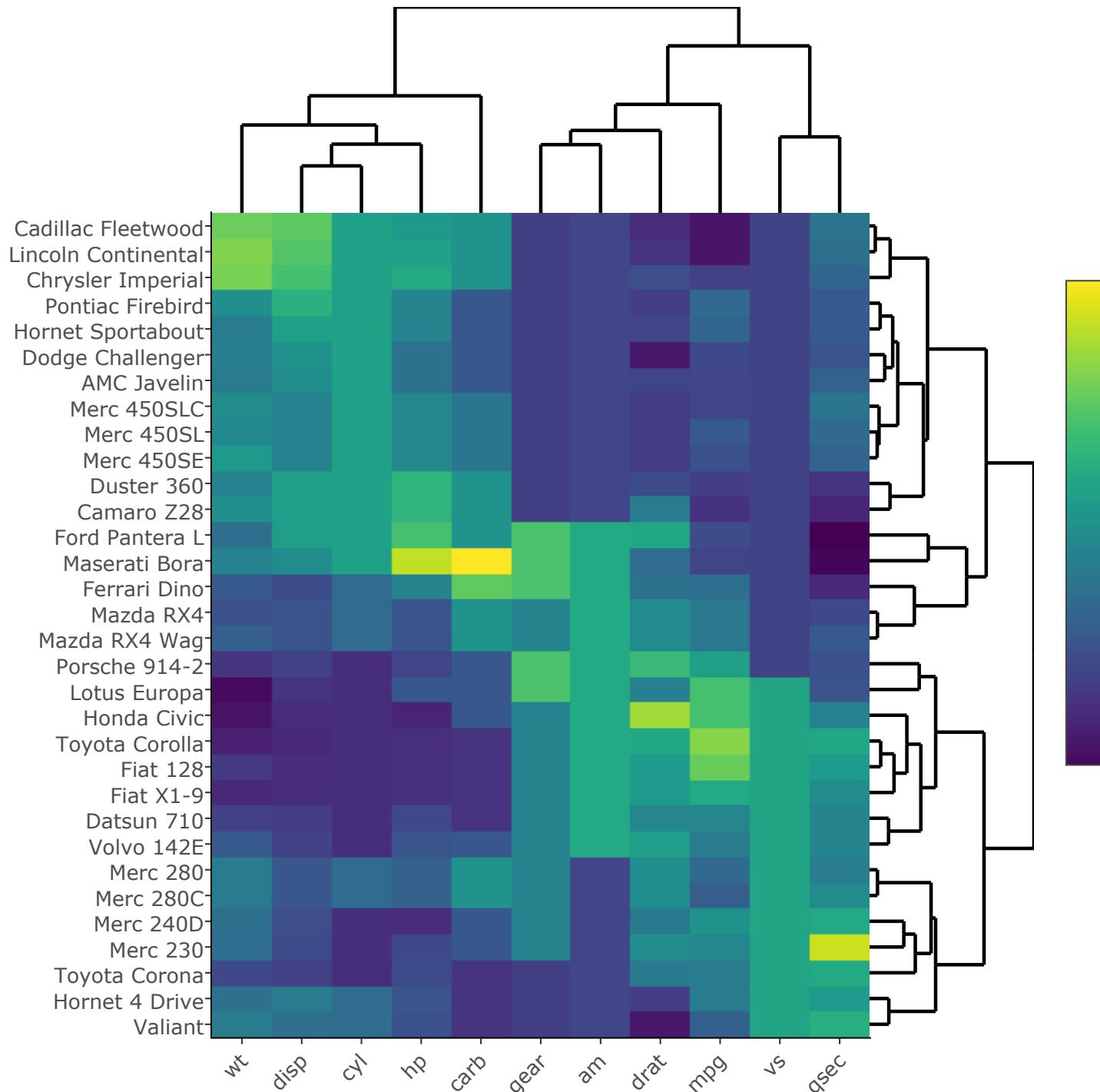


```
library(RColorBrewer)
# Define color scheme
scaleredblues <- colorRampPalette(colors = brewer.pal(9, "RdBu"))(10)
# "#B2182B" "#D25849" "#ED9576" "#FAC9B0" "#F9EAE1" "#E6EFF3" "#BCDAEA" "#80B9D8
heatmap(as.matrix(mtscaled), col = scaleredblues)
```



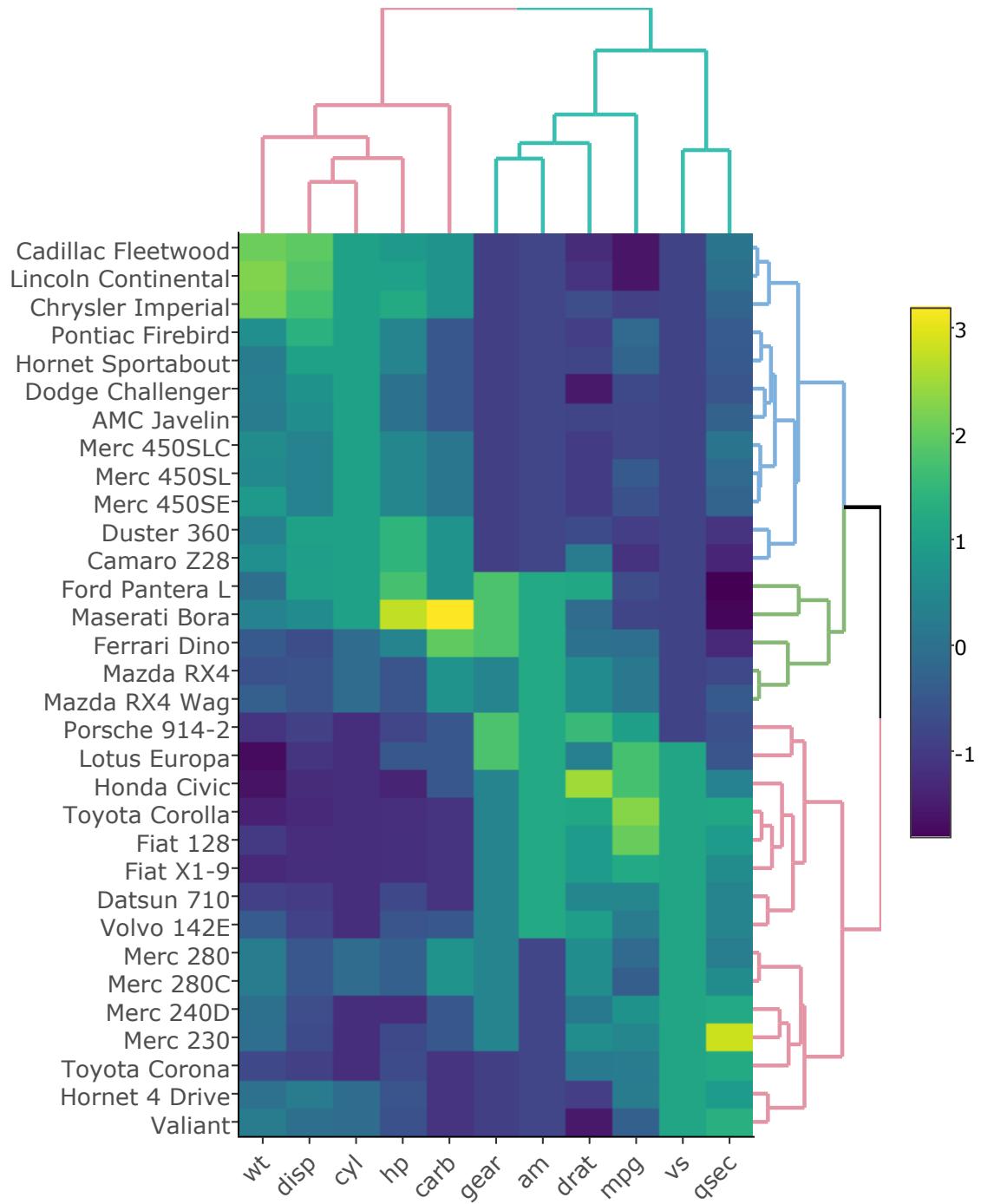
# heatmaply package for interactive heatmaps

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



# Color the dendrogram branches:

```
heatmaply(mtscaled, k_col = 2, k_row = 3) %>%
  layout(margin = list(l = 150, b = 50), autosize = F, width = 600, height = 700)
```



# **ggmap** for visualizing spatial data

# ggmap for plotting maps

## 1. Define location: 3 ways

- location/address
- lat/long
- bounding box

## 2. Define map source, type, and color with

`get_map( ):`

- Google Maps (“google”),
- OpenStreetMap (“osm”),
- Stamen Maps (“stamen”),
- CloudMade maps (“cloudmade”)

## 3. You can set the scale of the map using

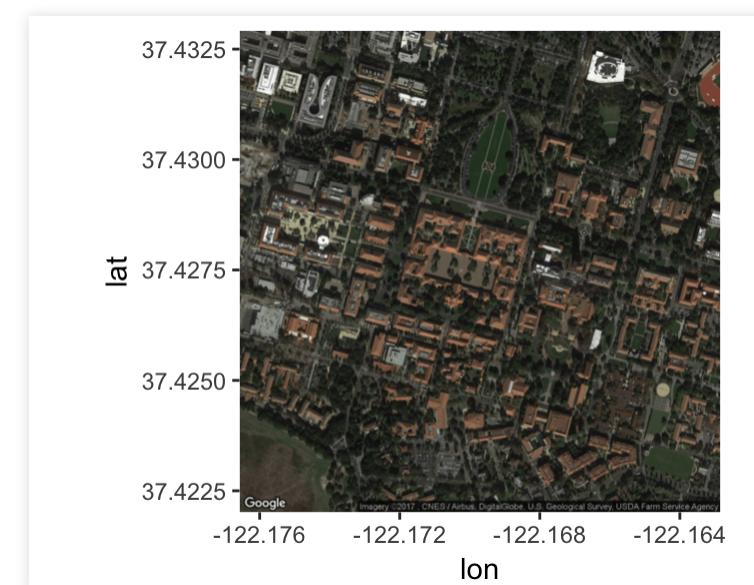
`zoom`. `get_map( )` takes a guess at the zoom level, but you can alter it.

- 3 (continent) to 21 (building), default value 10 (city).
- “openstreetmap” and “stamen” limit is 0-18.

```
library(ggmap)
geocode("Stanford University")
```

```
##          lon      lat
## 1 -122.1697 37.42747
```

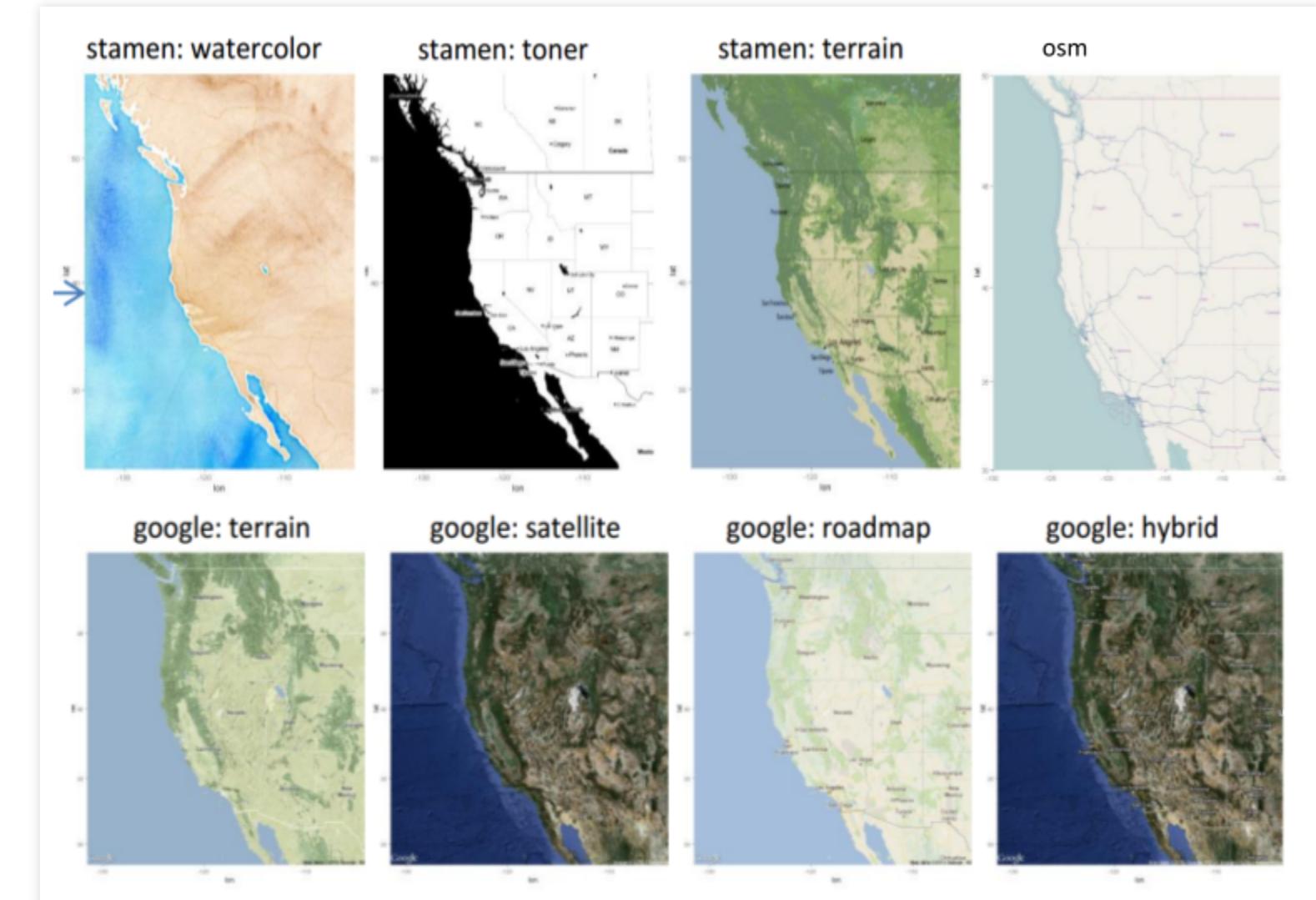
```
myLoc <- "Stanford University" # or myLoc <- c(-125,
# or myLoc <- c(lon = -122.1697, lat = 37.42747)
myMap <- get_map(location = myLoc, crop = TRUE, zoom
                  source = "google", maptype="satellite")
ggmap(myMap)
```



# Map sources

There are different map sources to obtain a map raster, and each of these sources has multiple “map types”. Here we show examples of maps from:

- **stamen**: `maptype = c("terrain", "toner", "watercolor")`
- **google**: `maptype = c("roadmap", "terrain", "satellite", "hybrid")`
- **osm**: [open street map](#)



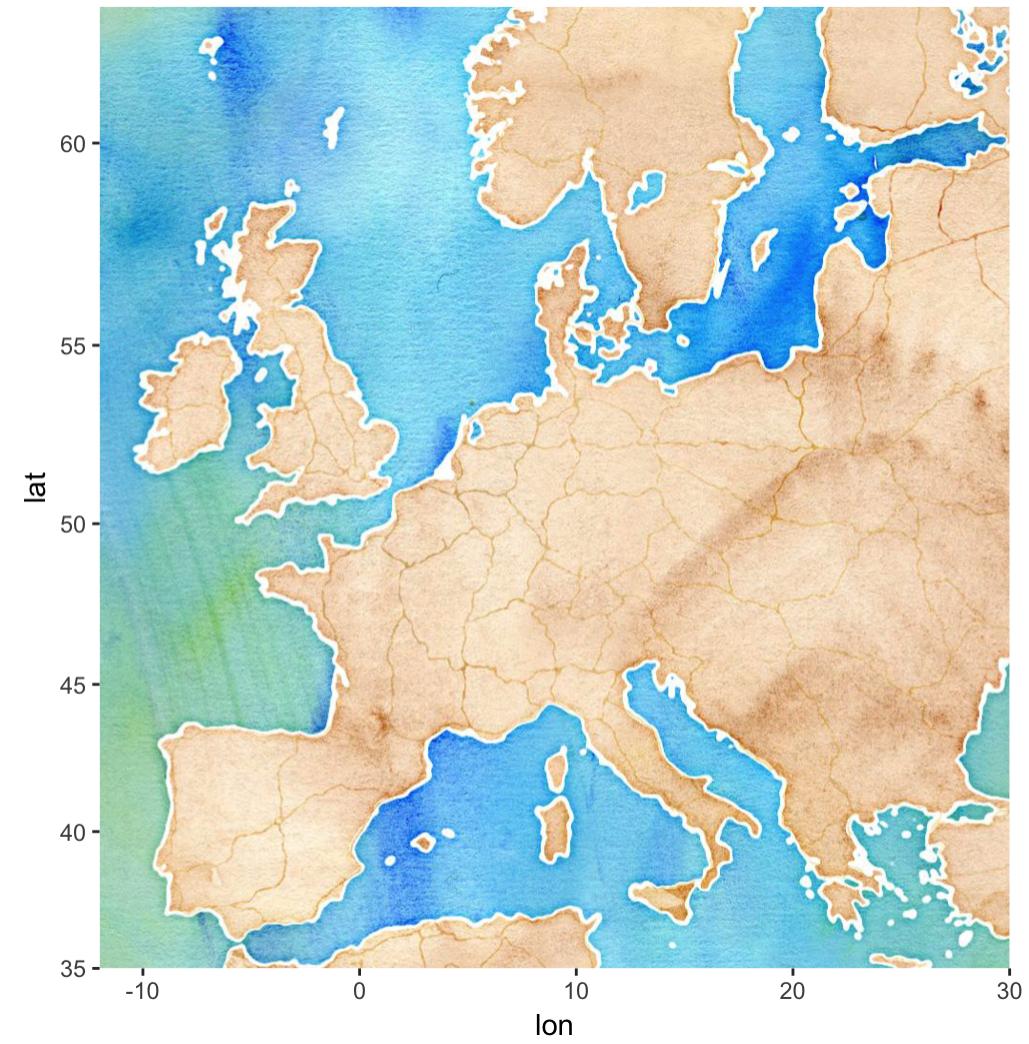
```
US <- c(left = -125, right = -67, bottom = 25.75, top = 49)
us_map <- get_map(location = US, source = "stamen", maptype = "toner-lite", zoom
class(us_map)
```

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

```
ggmap(us_map)
```



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



# Adding data

```
Lisbon <- geocode("Lisbon")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Lisbon
```

```
Warsaw <- geocode("Warsaw")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Warsaw
```

```
road <- data.frame(latitude = seq(Lisbon$lat, Warsaw$lat, length.out = 30),  
                    longitude = seq(Lisbon$lon, Warsaw$lon, length.out = 30))
```

```
ggmap(europe_map) +  
  geom_point(data = road, aes(x = longitude, y = latitude),  
             alpha = .5, color = "darkred")
```

