Plotting with R

Descriptive analysis and basic statistics in biomedical studies using R and Markdown

Juan R Gonzalez
juanr.gonzalez@isglobal.org

BRGE - Bioinformatics Research Group in Epidemiology
ISGlobal - Barcelona Institute for Global Health
http://brge.isglobal.org

Plotting

Plotting

- Simple plotting:
    - `plot`, `hist`, `pairs`, `boxplot`, ...
- Adding to existing plots:
    - `points`, `lines`, `abline`, `legend`, `title`, `mtext`, ...
- Interacting with graphics:
    - `locator`, `identify`
- Three dimensional data:
    - `contour`, `image`, `persp`, ...
- To see the many possibilities that R offers:

```
demo(graphics)
```

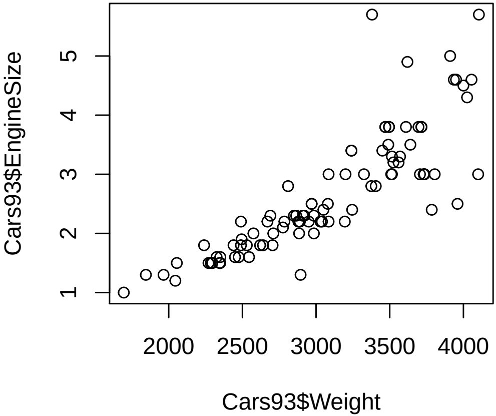
Basic plotting function is `plot()`. Possible arguments to `plot()` include:

- x, y (y may be omitted)
- xlim = c(lo, hi), ylim = c(lo, hi)
- `xlab = "x"`, `ylab = "y"` – labels for x- and y-axes respectively
- `type = "c"` – type of plot ("p", "l", "b", "h", "S", . . .)
- "lty = n" – line type (if lines used)
- "lwd = n" – line width
- "pch = v" – plotting character(s)
- "col = v" – colour to be used for everything.

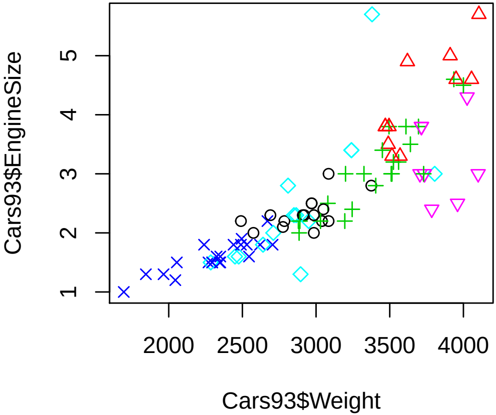
```r
library(MASS)
head(Cars93)
```

```
  Manufacturer   Model    Type Min.Price Price Max.Price MPG.city MPG.highway
1        Acura Integra   Small      12.9  15.9      18.8       25          31
2        Acura  Legend Midsize      29.2  33.9      38.7       18          25
3         Audi      90 Compact      25.9  29.1      32.3       20          26
4         Audi     100 Midsize      30.8  37.7      44.6       19          26
5          BMW    535i Midsize      23.7  30.0      36.2       22          30
6        Buick Century Midsize      14.2  15.7      17.3       22          31
             AirBags DriveTrain Cylinders EngineSize Horsepower  RPM
1               None      Front         4        1.8        140 6300
2 Driver & Passenger      Front         6        3.2        200 5500
3        Driver only      Front         6        2.8        172 5500
4 Driver & Passenger      Front         6        2.8        172 5500
5        Driver only       Rear         4        3.5        208 5700
6        Driver only      Front         4        2.2        110 5200
  Rev.per.mile Man.trans.avail Fuel.tank.capacity Passengers Length Wheelbase
1         2890             Yes               13.2          5    177       102
2         2335             Yes               18.0          5    195       115
3         2280             Yes               16.9          5    180       102
4         2535             Yes               21.1          6    193       106
5         2545             Yes               21.1          4    186       109
6         2565              No               16.4          6    189       105
  Width Turn.circle Rear.seat.room Luggage.room Weight  Origin          Make
1    68          37           26.5           11   2705 non-USA Acura Integra
2    71          38           30.0           15   3560 non-USA  Acura Legend
3    67          37           28.0           14   3375 non-USA       Audi 90
4    70          37           31.0           17   3405 non-USA      Audi 100
5    69          39           27.0           13   3640 non-USA      BMW 535i
6    69          41           28.0           16   2880     USA Buick Century
```

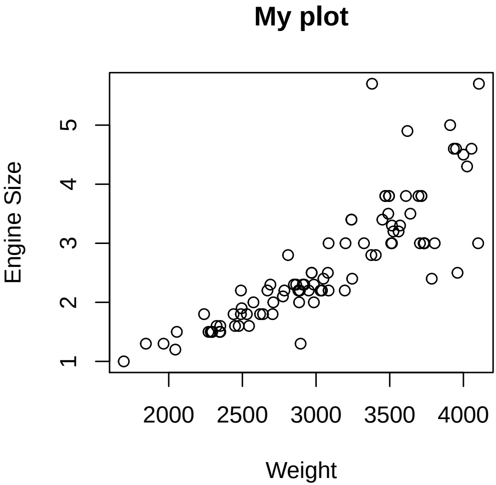
```r
plot(Cars93$Weight, Cars93$EngineSize)
```



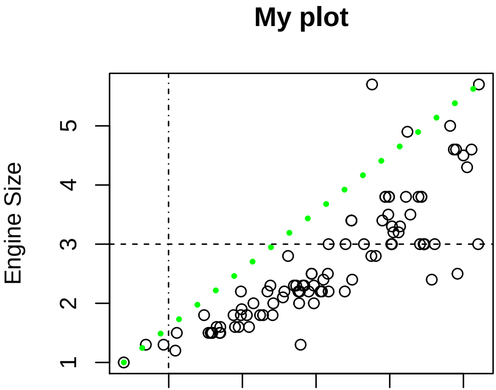
```
# Colour points and choose plotting symbols according to
# a levels of a factor
plot(Cars93$Weight, Cars93$EngineSize,
     col=as.numeric(Cars93$Type),
     pch=as.numeric(Cars93$Type))
```



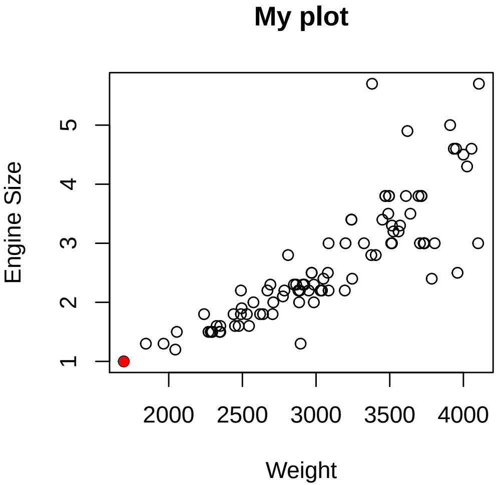
```
# Add x and y axes labels and a title.
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")
```



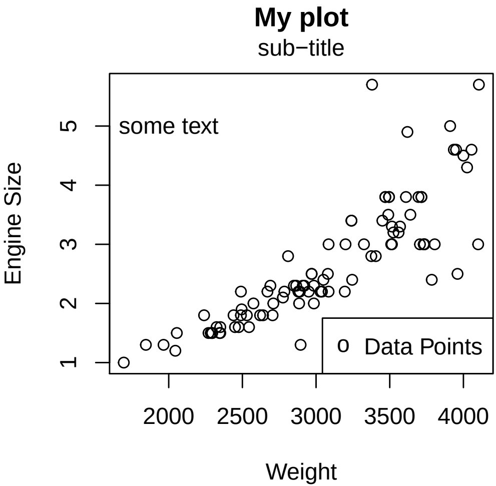
```r
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")

# Add lines to the plot.
lines(x=c(min(Cars93$Weight), max(Cars93$Weight)),
      y=c(min(Cars93$EngineSize),
      max(Cars93$EngineSize)), lwd=4, lty=3, col="green")
abline(h=3, lty=2)
abline(v=1999, lty=4)
```



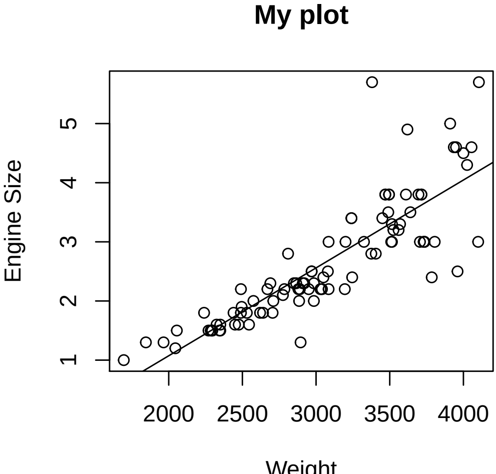
```r
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")

# Add points to the plot
points(x=min(Cars93$Weight), y=min(Cars93$EngineSize), pch=16, c
```



```r
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")

# Add text to the plot.
text(x=2000, y=5, "some text")

# Add text under main title.
mtext(side=3, "sub-title", line=0.45)

# Add a legend
legend("bottomright", legend=c("Data Points"), pch="o")
```


My plot
sub−title
some text
o Data Points
Engine Size
1
2
3
4
5
2000
2500
3000
3500
4000
Weight

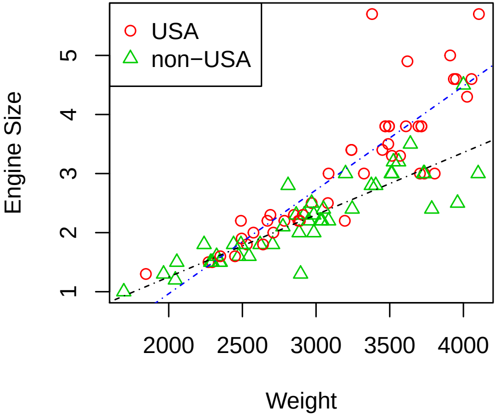
```r
plot(Cars93$Weight, Cars93$EngineSize, ylab="Engine Size",
xlab="Weight", main="My plot")

# Add regression line
mod <- lm(EngineSize ~ Weight, data=Cars93)
abline(mod)
```



```r
levels(Cars93$Origin)
```

```
[1] "USA"     "non-USA"
```

```r
plot(Cars93$Weight, Cars93$EngineSize,
     pch = (1:2)[Cars93$Origin],
     col = (2:3)[Cars93$Origin],
     xlab="Weight", ylab="Engine Size")
legend("topleft", legend=levels(Cars93$Origin),
       pch=1:2, col=2:3)

fm1 <- lm(EngineSize ~ Weight, Cars93, subset = Origin == "USA")
abline(coef(fm1), lty=4, col="blue")

fm2 <- lm(EngineSize ~ Weight, Cars93, subset = Origin == "non-U
abline(coef(fm2), lty=4, col="black")
```

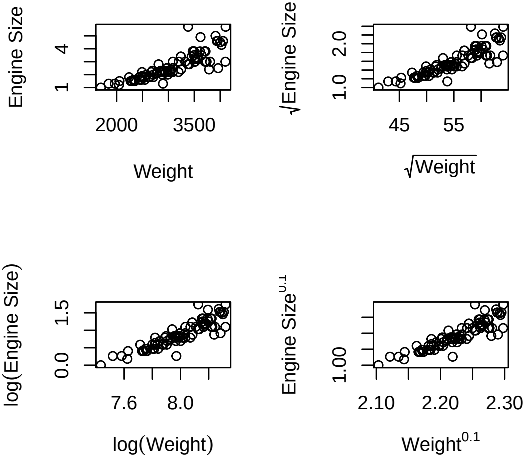

USA
non−USA
Engine Size
1
2
3
4
5
2000
2500
3000
3500
4000
Weight

Multiple figures

```r
# Will create 4 plots on the same page.
# Two in each row and two in each column.
par(mfrow=c(2,2))

plot(Cars93$Weight, Cars93$EngineSize,
     xlab="Weight", ylab="Engine Size")

plot(sqrt(Cars93$Weight), sqrt(Cars93$EngineSize),
     xlab=expression(sqrt(Weight)),
     ylab=expression(sqrt("Engine Size")))

plot(log(Cars93$Weight), log(Cars93$EngineSize),
     xlab=expression(log(Weight)),
     ylab=expression(log("Engine Size")))

plot(Cars93$Weight^0.1, Cars93$EngineSize^0.1,
     xlab=expression(Weight^0.1),
     ylab=expression("Engine Size"^0.1) )
```

```r
par(mfrow=c(1,1))     # Resets to create a single plot per page.
```

[Advanced:] The expression command plots mathematical symbols on the x and y axes. For more information 'help(expression)

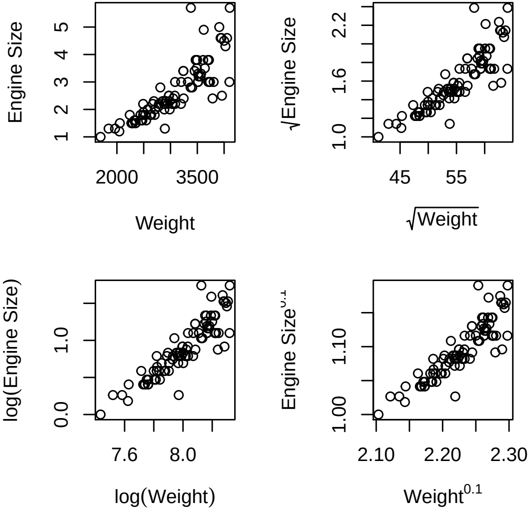

Engine Size
1
4
2000
3500
Weight
√Engine Size
1.0
2.0
45
55
√Weight
log(Engine Size)
0.0
1.5
7.6
8.0
log(Weight)
Engine Size$^{0.1}$
1.00
2.10
2.20
2.30
Weight$^{0.1}$

Improve the figure by

```
par("mar")
```

```
[1] 5.1 4.1 4.1 2.1
```

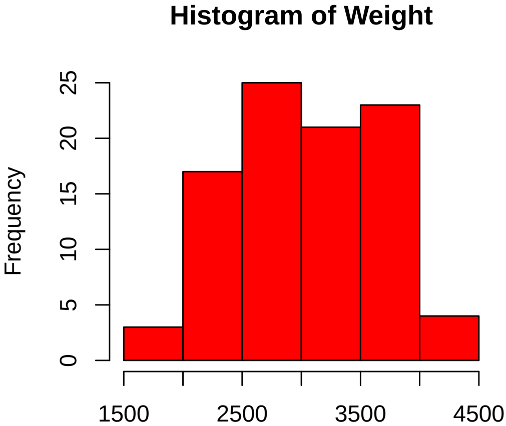
```
par(mar=c(5,4,1,2))
par(mfrow=c(2,2))
plot(Cars93$Weight, Cars93$EngineSize,
     xlab="Weight", ylab="Engine Size")
plot(sqrt(Cars93$Weight), sqrt(Cars93$EngineSize),
     xlab=expression(sqrt(Weight)),
     ylab=expression(sqrt("Engine Size")))
plot(log(Cars93$Weight), log(Cars93$EngineSize),
     xlab=expression(log(Weight)),
     ylab=expression(log("Engine Size")))
plot(Cars93$Weight^0.1, Cars93$EngineSize^0.1,
     xlab=expression(Weight^0.1),
     ylab=expression("Engine Size"^0.1) )
```


Engine Size
1
2
3
4
5
2000
3500
Weight
√Engine Size
1.0
1.6
2.2
45
55
√Weight
log(Engine Size)
0.0
1.0
7.6
8.0
log(Weight)
Engine Size$^{0.1}$
1.00
1.10
2.10
2.20
2.30
Weight$^{0.1}$

Histograms

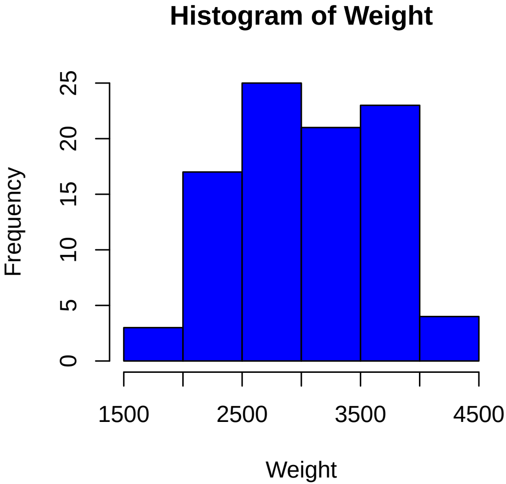
Histograms can be created using the `hist` command. Let us create a histogram of the car weights from the Cars93 data set:

```r
hist(Cars93$Weight, xlab="Weight",
     main="Histogram of Weight", col="red")
```



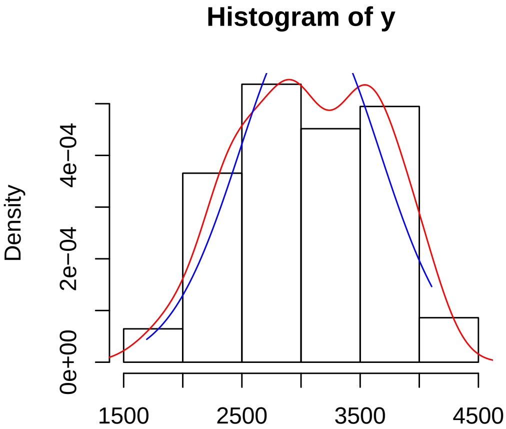
R automatically chooses the number and width of the bars. Can change this by specifying the number of break points.

```
hist(Cars93$Weight, breaks=8, xlab="Weight",
     main="Histogram of Weight", col="blue")
```



Normal curve can be added to the histogram by:

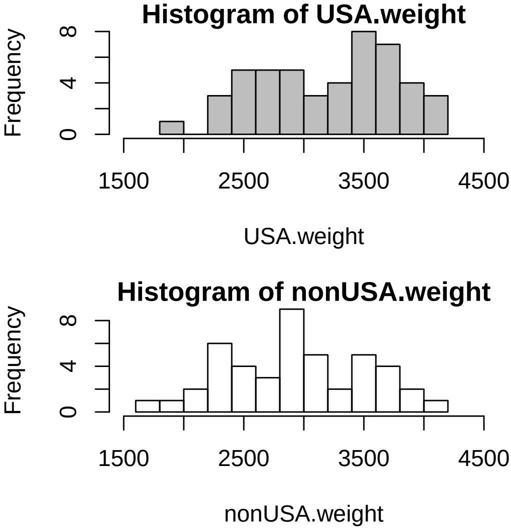
```
y <- Cars93$Weight # put here your variable of interest
hist(y, freq=FALSE)
lines(density(y), col="red")
x <- seq(min(y), max(y), length=100)
lines(x, dnorm(x, mean(y), sd(y)), col="blue")
```



Histograms for multiple groups

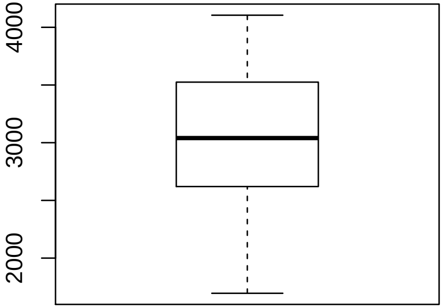
```r
USA.weight <- Cars93$Weight[Cars93$Origin == "USA"]
nonUSA.weight <- Cars93$Weight[Cars93$Origin == "non-USA"]

par(mfrow=c(2,1))
par(mar=c(5,4,1,2))
hist(USA.weight, breaks=10, xlim=c(1500,4500), col="grey")
hist(nonUSA.weight, breaks=10, xlim=c(1500,4500))
```

```r
par(mfrow=c(1,1))
```

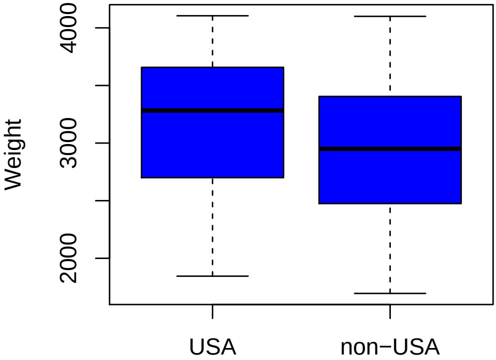

Histogram of USA.weight
Frequency
0
4
8
1500
2500
3500
4500
USA.weight
Histogram of nonUSA.weight
Frequency
0
4
8
1500
2500
3500
4500
nonUSA.weight

Boxplots

```
boxplot(Cars93$Weight)
```

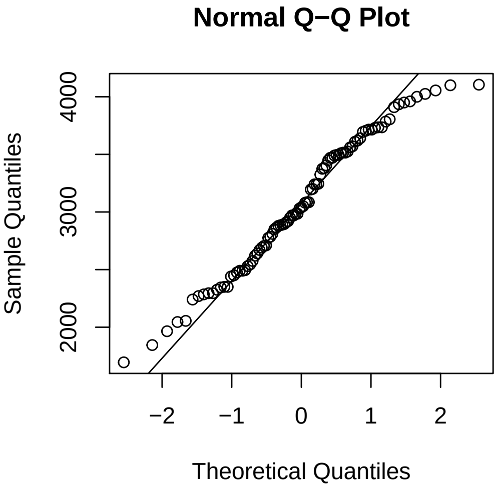


```
boxplot(Cars93$Weight ~ Cars93$Origin,
        col="blue", ylab="Weight")
```



Normal probability (Q-Q) plots

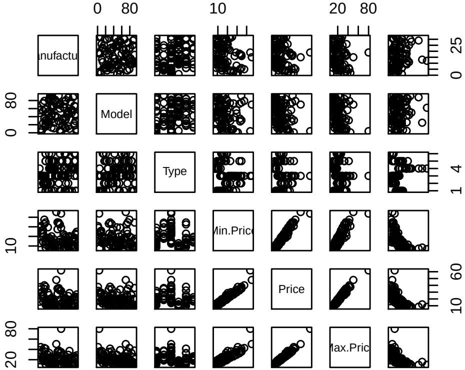
```
qqnorm(Cars93$Weight)
qqline(Cars93$Weight)
```



Plots for multivariate data

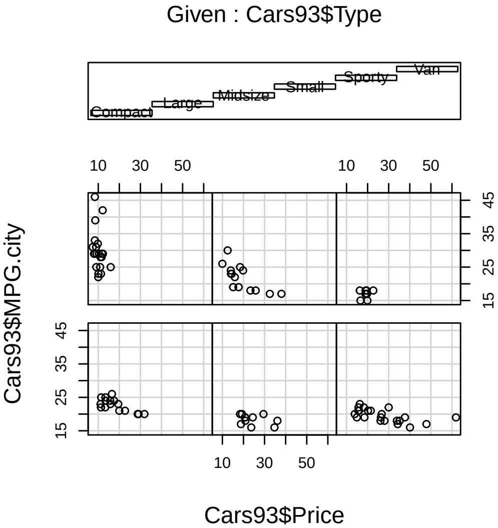
If your data are stored in a data frame with several columns, the `pairs` command produces pairwise plots of the data in each column, i.e. the data in column 1 vs the data in column 2, column 1 vs column 3, and so on.

```r
pairs(Cars93[,1:7])
```



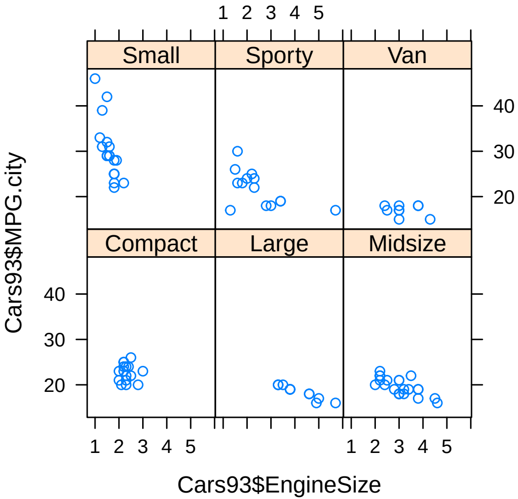
This is limited to 2 grouping variables. More flexible is `xyplot` in the `lattice` library.

```
coplot(Cars93$MPG.city~Cars93$Price|Cars93$Type)
```

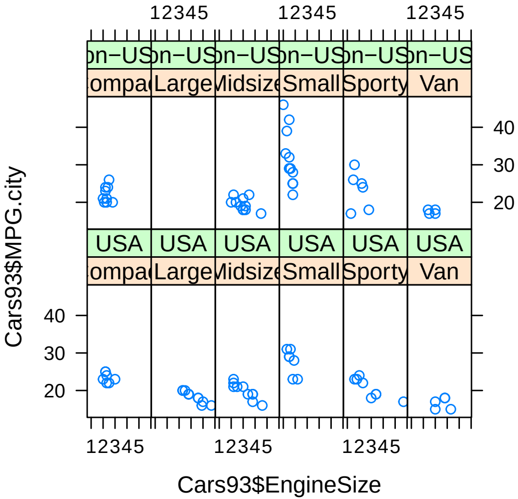


Lattice graphs

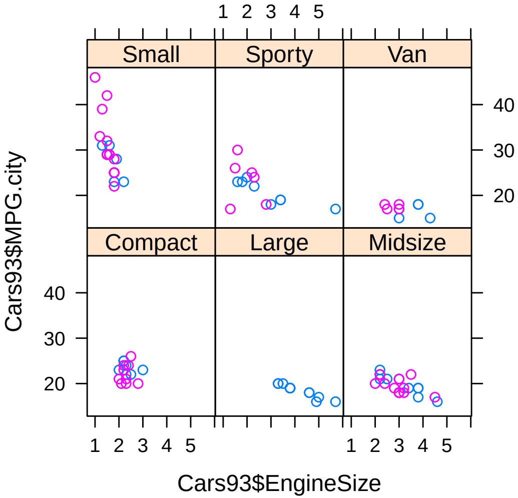
```
library(lattice)
xyplot(Cars93$MPG.city~Cars93$EngineSize|Cars93$Type)
```



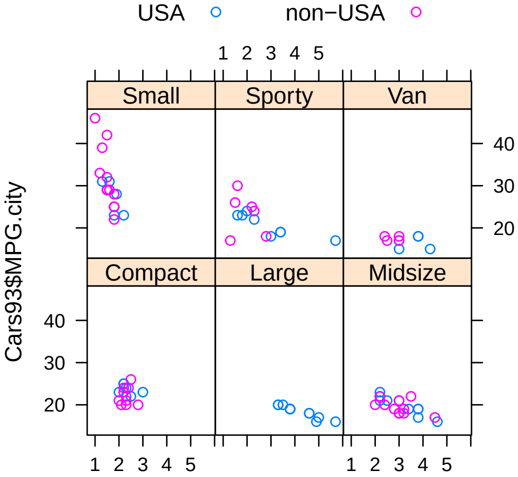
```
xyplot(Cars93$MPG.city~Cars93$EngineSize|Cars93$Type*Cars93$Orig
```



```
xyplot(Cars93$MPG.city~Cars93$EngineSize|Cars93$Type,
       panel=panel.superpose, groups=Cars93$Origin)
```



```r
# Adds a key
pars <- trellis.par.get("superpose.symbol")
xyplot(Cars93$MPG.city~Cars93$EngineSize|Cars93$Type,
       panel=panel.superpose, groups=Cars93$Origin,
       key = list(columns = 2, text = list(levels(Cars93$Origin)
                                   points = Rows(pars,1:2)))
```



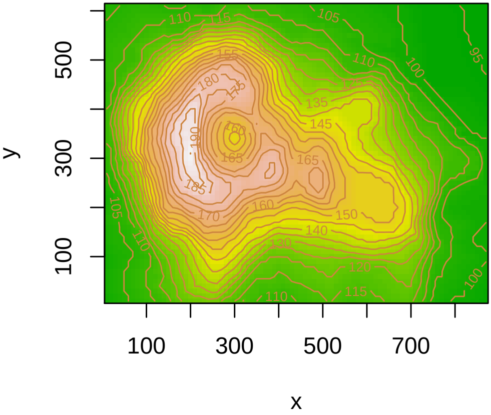
Other lattice plots

- splom(~ data.frame) # Scatterplot matrix
- bwplot(factor ~ numeric, . . .) # Boxplot
- qqmath(factor ~ numeric, . . .) # Q-Q plot
- dotplot(factor ~ numeric, . . .) # 1-D display
- stripplot(factor ~ numeric, . . .)
- barchar(character ~ numeric, . . .)
- histogram(~ numeric, . . .)
- densityplot(~ numeric, . . .) # Smoothed version of histogram

2-D and 3-D plots

```r
data(volcano)
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))

# Creates a 2-D image of x and y co-ordinates.
image(x, y, volcano, col = terrain.colors(100),
      axes = FALSE)

# Adds contour lines to the current plot.
contour(x, y, volcano, levels = seq(90, 200, by=5),
        add = TRUE, col = "peru")

# Adds x and y axes to the plot.
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))

# Draws a box around the plot.
box()

# Adds a title.
title(main = "Maunga Whau Volcano", font.main = 4)
```

Maunga Whau Volcano



Session info

```r
sessionInfo()
```

```
R version 3.4.1 (2017-06-30)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 16299)

Matrix products: default

locale:
[1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
[3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
[5] LC_TIME=Spanish_Spain.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] lattice_0.20-35 MASS_7.3-47     knitr_1.20

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.12     codetools_0.2-15 digest_0.6.12    rprojroot_1.3-2
 [5] grid_3.4.1       backports_1.1.0  magrittr_1.5     evaluate_0.10.1
 [9] stringi_1.1.6    rmarkdown_1.8    tools_3.4.1      stringr_1.3.0
[13] yaml_2.1.16      compiler_3.4.1   htmltools_0.3.6
```