Multivariate Stats in R

Mike Herrmann

Thursday, October 29, 2015

Contents

Statistical tests that we will be covering

1. ANOVA and LM
2. Principal Component Analysis
3. Discriminant Analysis
4. Non-metric Multidimentional Scaling (NMDS)

```
# Install required packages
install.packages("MASS")
install.packages("ggplot2")
install.packages("vegan")
install.packages("Ecdat")
install.packages("ellipse")
```

```
# Load packages
library(MASS)
library(ggplot2)
library(vegan)
library(Ecdat)
```

ANOVA, ANCOVA, and LM

First example, basic ANOVA. We will use a diamond dataset in the package Ecdat. It compares the price of diamonds to several measurements.

```
attach(Diamond)
View(Diamond)

#test for effects of transmission type, starting with a simple one-way test
```

```r
diamond_test1 <- aov(price ~ clarity)
summary(diamond_test1)
```

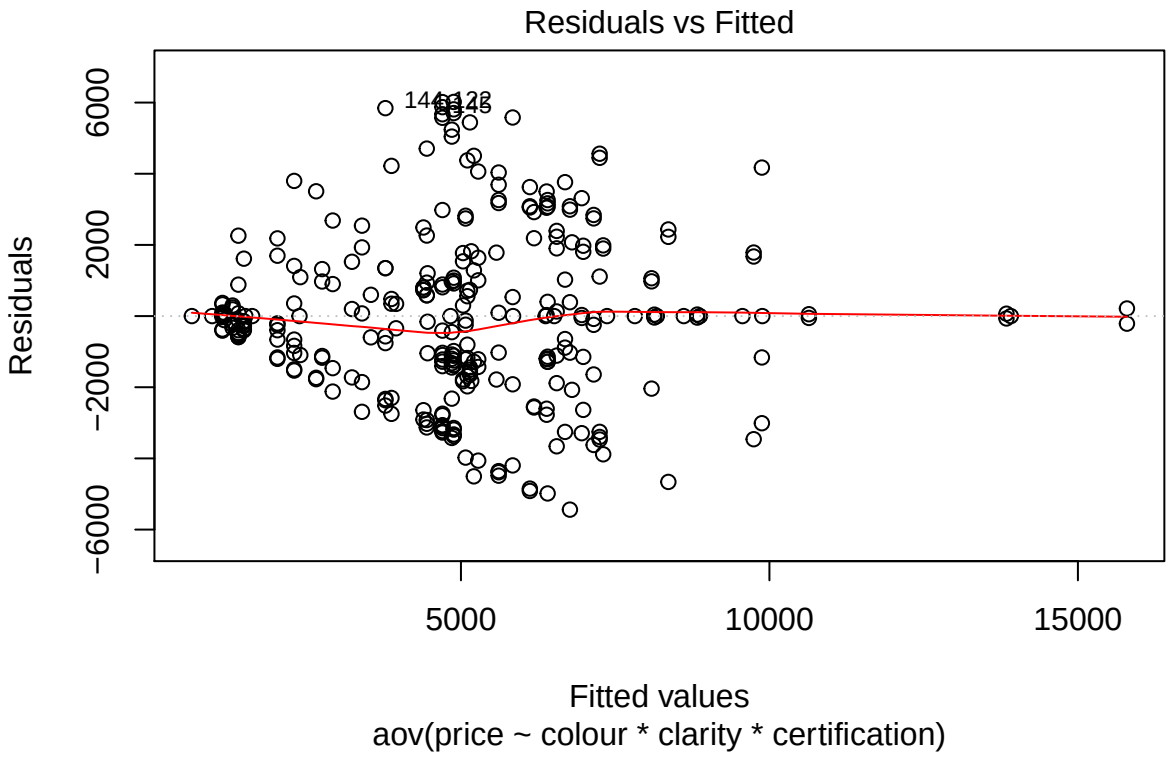
```
##              Df    Sum Sq  Mean Sq F value   Pr(>F)
## clarity       4 2.997e+08 74917212   6.972 2.22e-05 ***
## Residuals   303 3.256e+09 10745078
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

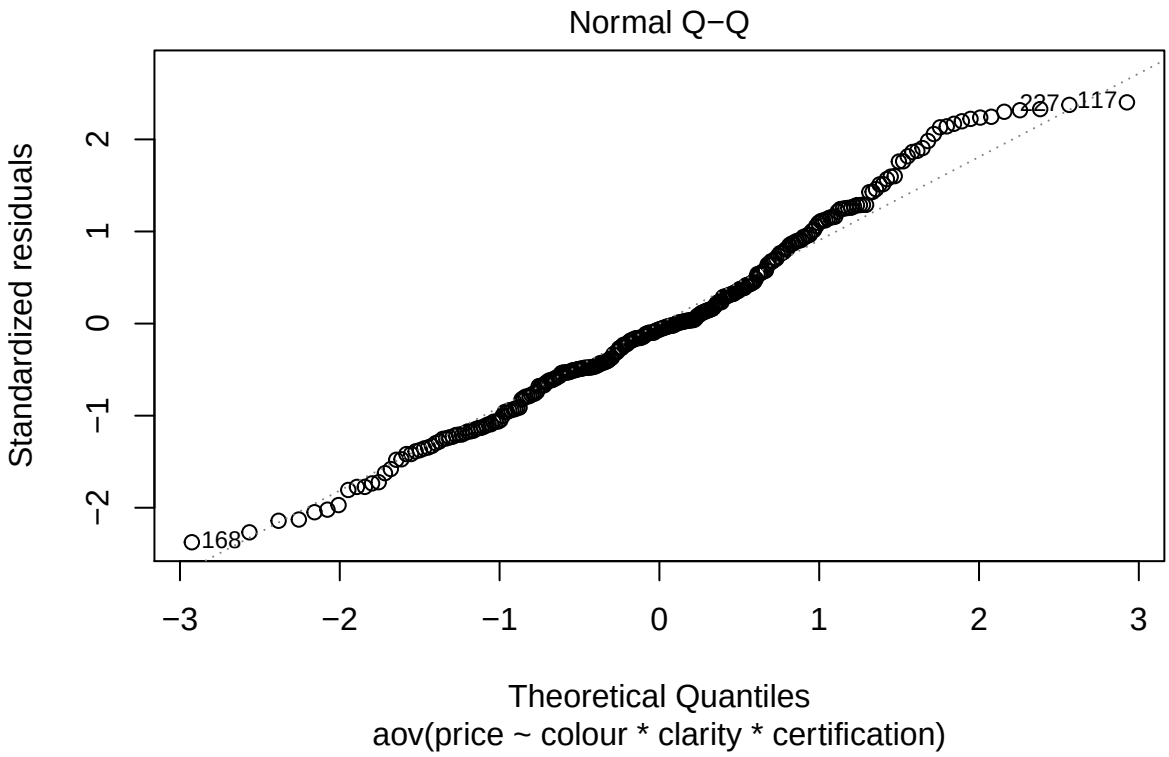
```r
#add in more factors, using * between factors will give you a full-factorial design
#A WARNING! R uses a type-I sequencial SS for their analysis. THIS ANALYSIS WILL GIVE DIFFERENT RESULTS
diamond_test3 <- aov(price ~ colour*clarity*certification)
summary(diamond_test3)
```

```
##                                Df    Sum Sq   Mean Sq F value   Pr(>F)
## colour                          5 1.082e+08  21645956   3.008 0.011864 *
## clarity                         4 2.961e+08  74015430  10.287 1.09e-07 ***
## certification                   2 7.245e+08 362244662  50.346  < 2e-16 ***
## colour:clarity                 20 2.829e+08  14142796   1.966 0.009612 **
## colour:certification           10 1.314e+08  13141607   1.826 0.056947 .
## clarity:certification           8 1.980e+08  24754541   3.440 0.000914 ***
## colour:clarity:certification   24 1.307e+08   5444933   0.757 0.788139
## Residuals                     234 1.684e+09   7195127
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

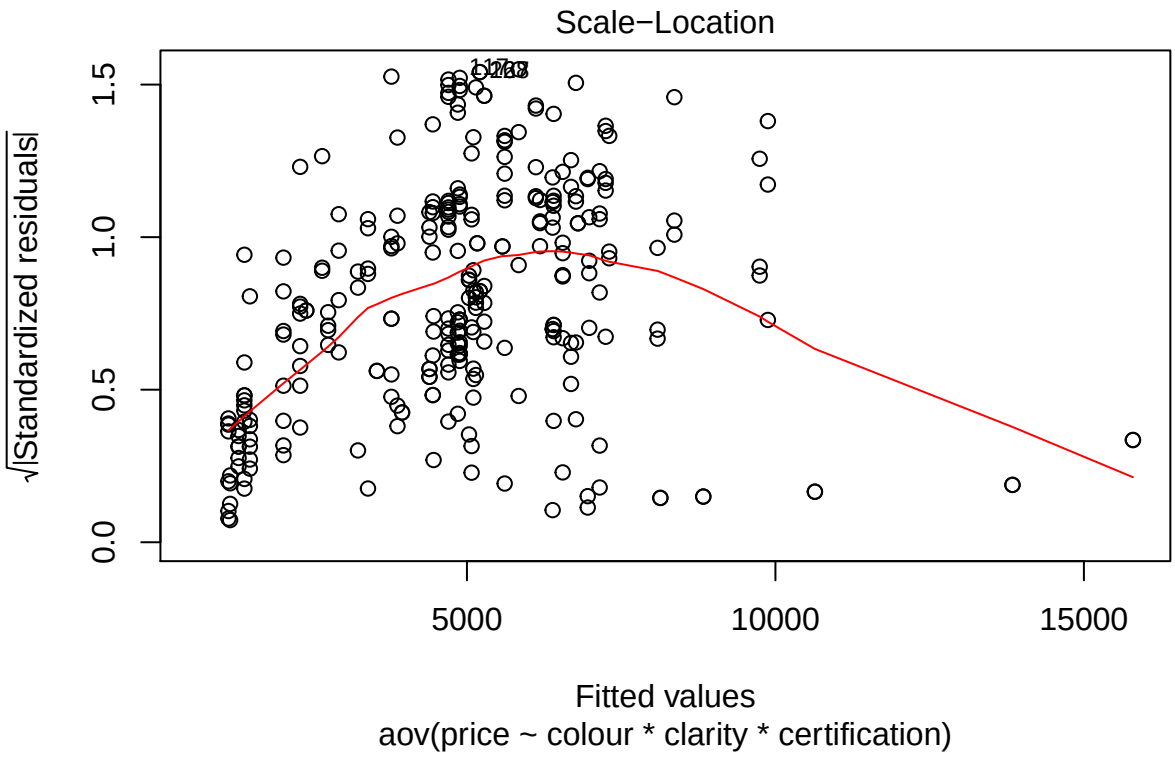
```r
#To look at diagnostic plots, we can use the plot function
plot(diamond_test3)
```

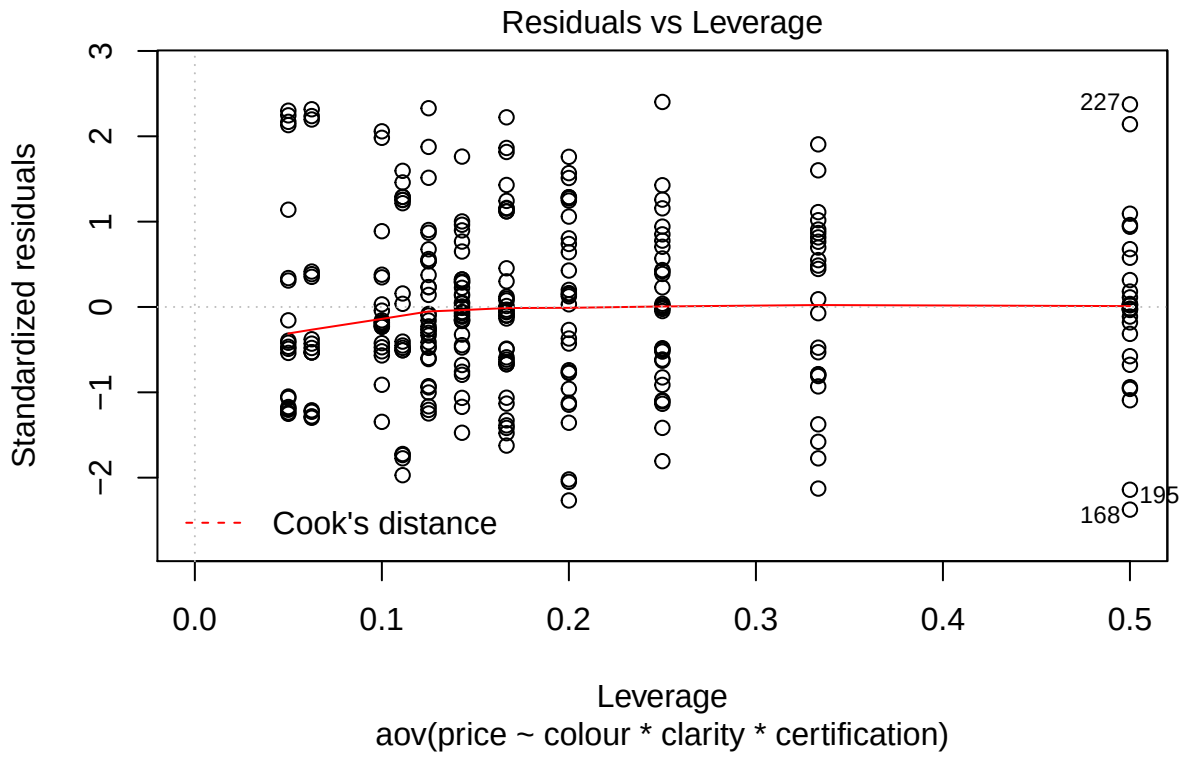
```
## Warning: not plotting observations with leverage one:
##   61, 88, 107, 110, 120, 146, 158, 169, 207, 208, 210, 218, 228, 229, 256, 262, 293
```


Residuals vs Fitted
Residuals
6000
2000
−2000
−6000
5000
10000
15000
Fitted values
aov(price ~ colour * clarity * certification)

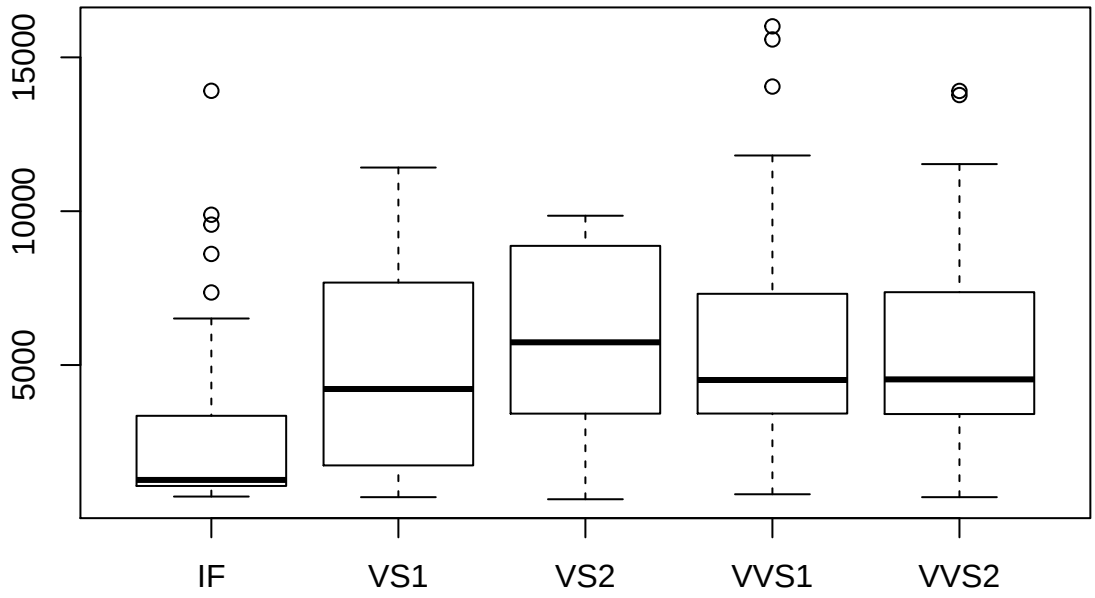



```
## Warning: not plotting observations with leverage one:
##   61, 88, 107, 110, 120, 146, 158, 169, 207, 208, 210, 218, 228, 229, 256, 262, 293
```

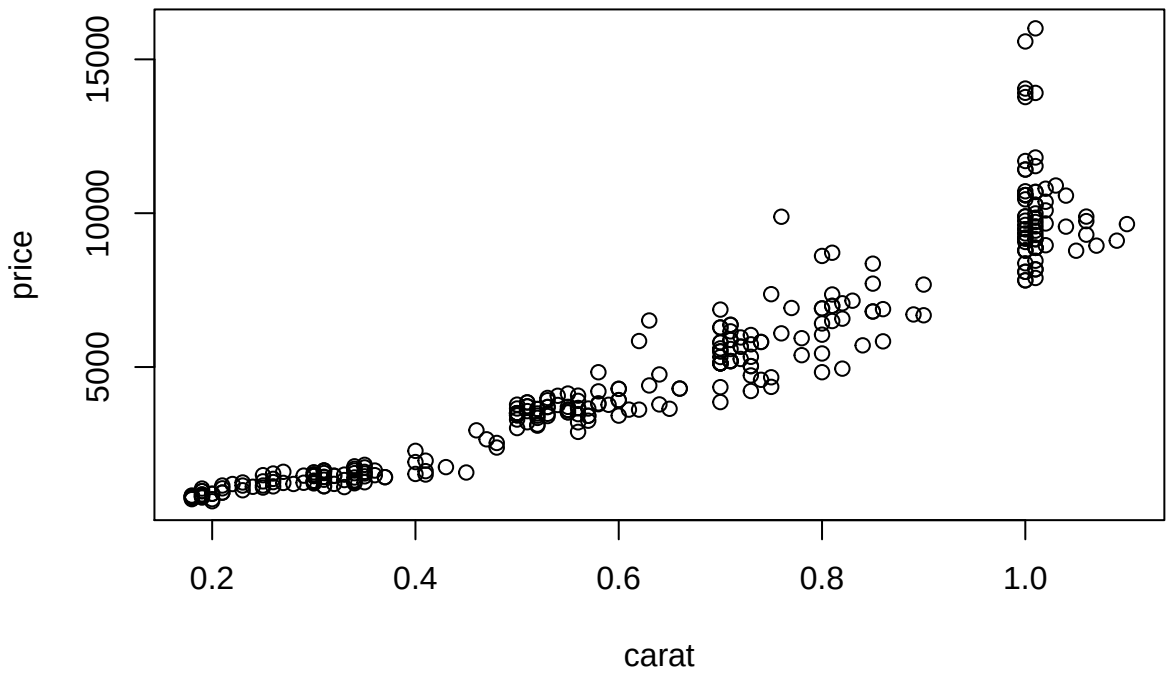

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
5000
10000
15000
Fitted values
aov(price ~ colour * clarity * certification)




```
#to see effect directions, boxplots work well
boxplot(price~clarity)
```




```
plot(price~carat)
```




```
#For an ANCOVA, replace * with +. This is also useful if you have a random variable, such as with a blo

diamond_ancova<-aov(price ~ clarity + colour)
summary(diamond_ancova)
```

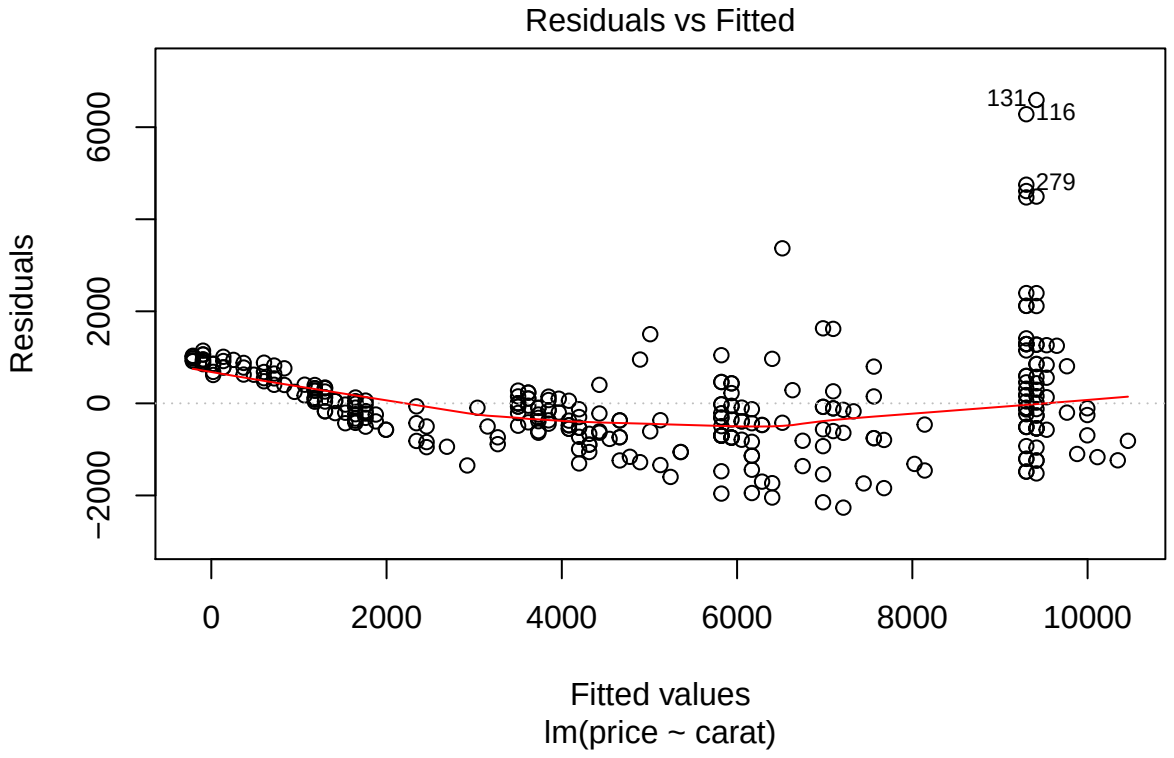
```
##              Df    Sum Sq  Mean Sq F value   Pr(>F)
## clarity       4 2.997e+08 74917212   7.085 1.84e-05 ***
## colour        5 1.046e+08 20924530   1.979   0.0816 .
## Residuals   298 3.151e+09 10574281
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

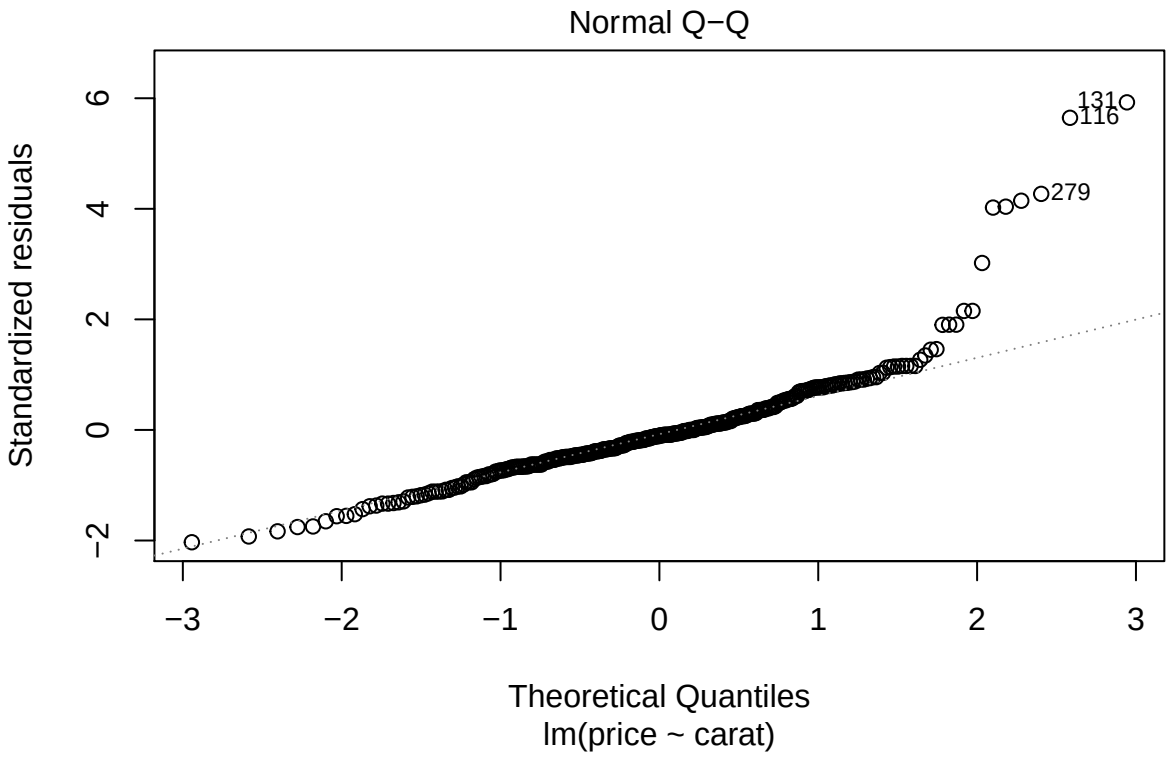
```
#linear models work essentially the same way, but instead of "aov", we use "lm"

diamond_lm <- lm(price~carat)
summary(diamond_lm)
```

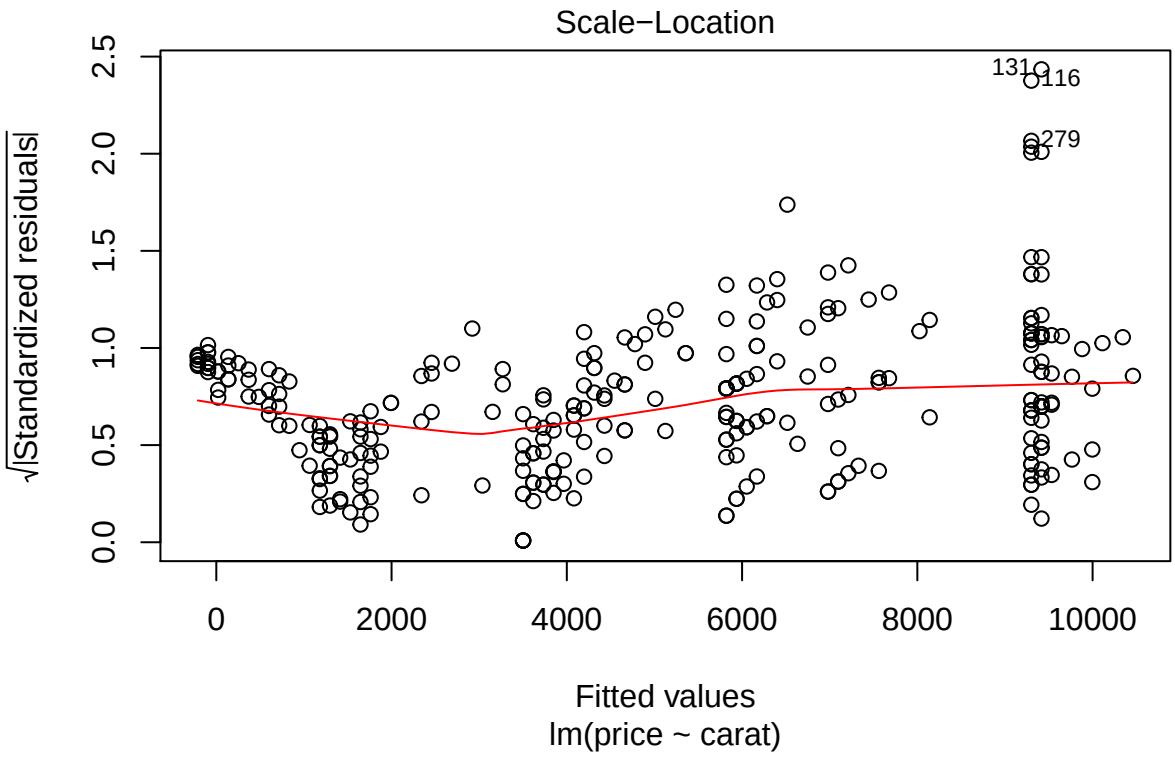
```
##
## Call:
## lm(formula = price ~ carat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2264.7  -604.3  -116.1   435.1  6591.5
##
```

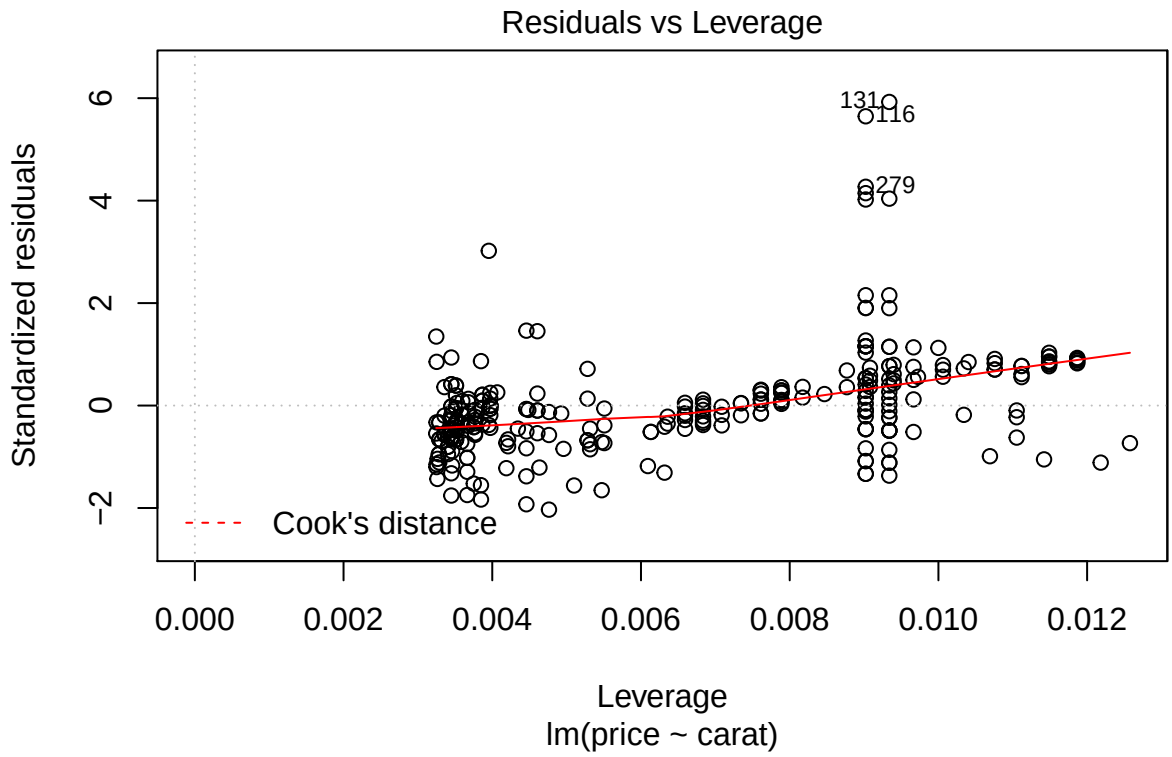
```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2298.4      158.5  -14.50   <2e-16 ***
## carat        11598.9      230.1   50.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1118 on 306 degrees of freedom
## Multiple R-squared:  0.8925, Adjusted R-squared:  0.8922
## F-statistic:  2541 on 1 and 306 DF,  p-value: < 2.2e-16
```

```
plot(diamond_lm)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(price ~ carat)
131
116
279
−3
−2
−1
0
1
2
3
−2
0
2
4
6


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
131
116
279
0
2000
4000
6000
8000
10000
Fitted values
lm(price ~ carat)




```
detach(Diamond)
```

Principal Component Analysis (PCA)

Principal component analysis- to look at this, we will use a different sample data set that comes with R, the Iris dataset

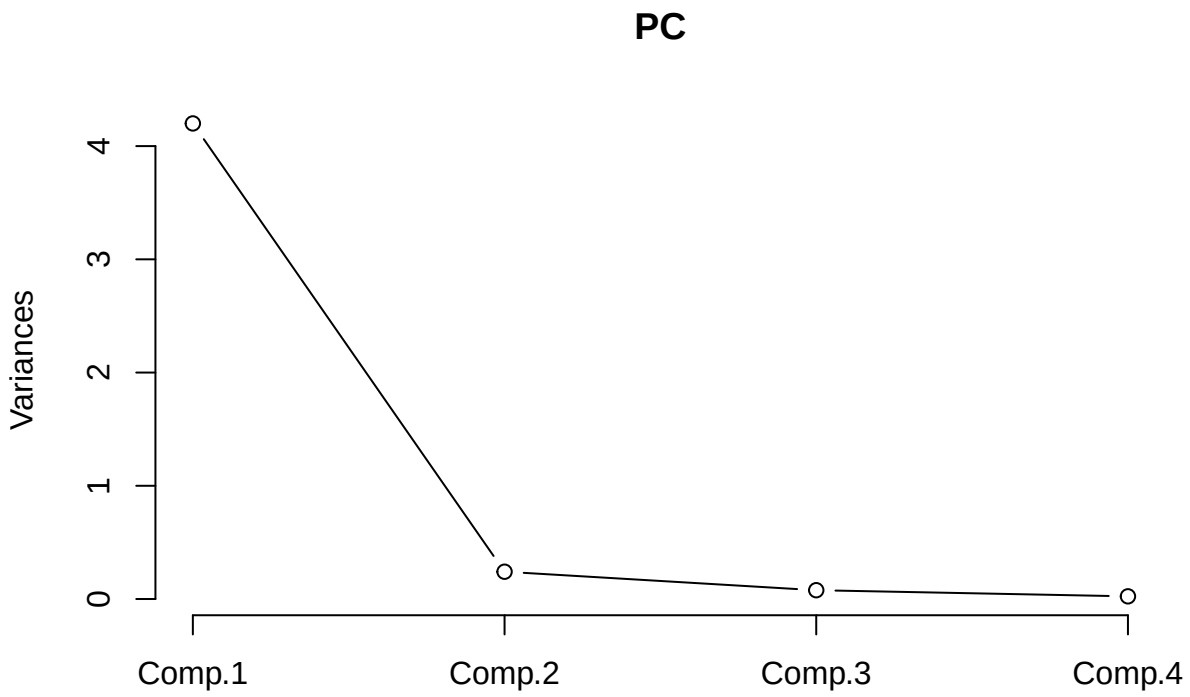
```
attach(iris)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

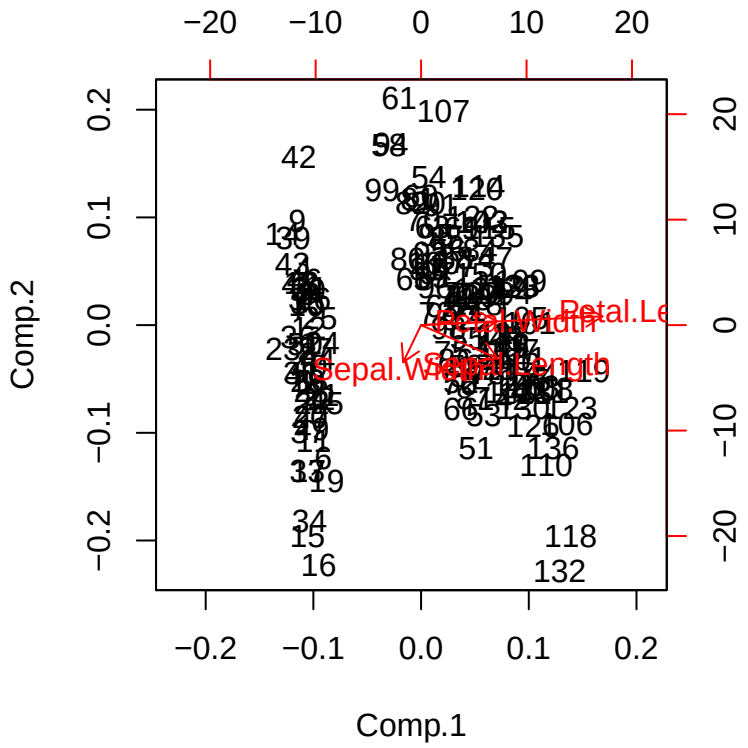
```r
par(mfrow=c(1,1))
PC <- princomp(iris[,1:4], scores=TRUE)
summary(PC)
```

```
## Importance of components:
##                           Comp.1     Comp.2     Comp.3      Comp.4
## Standard deviation     2.0494032 0.49097143 0.27872586 0.153870700
## Proportion of Variance 0.9246187 0.05306648 0.01710261 0.005212184
## Cumulative Proportion  0.9246187 0.97768521 0.99478782 1.000000000
```

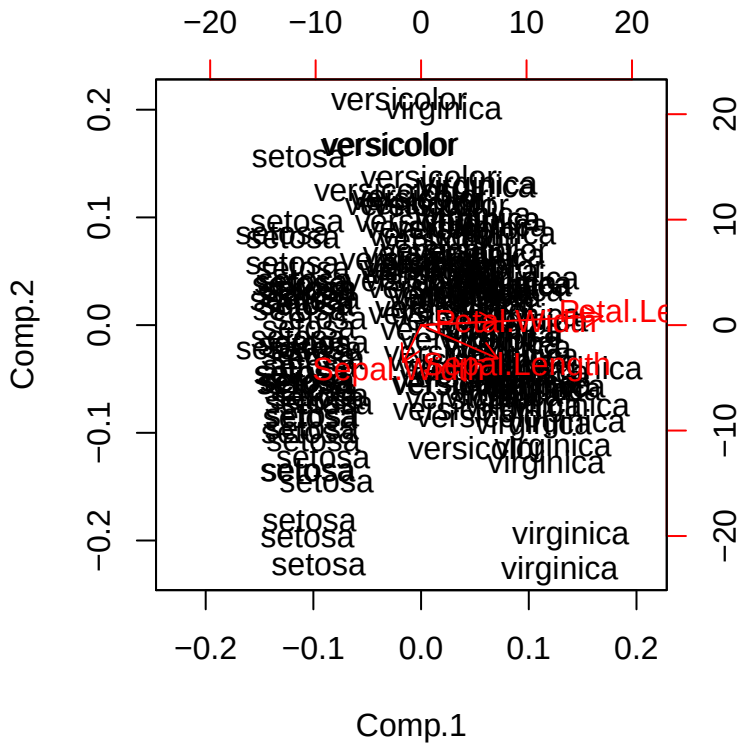
```r
plot(PC, type="lines")
```



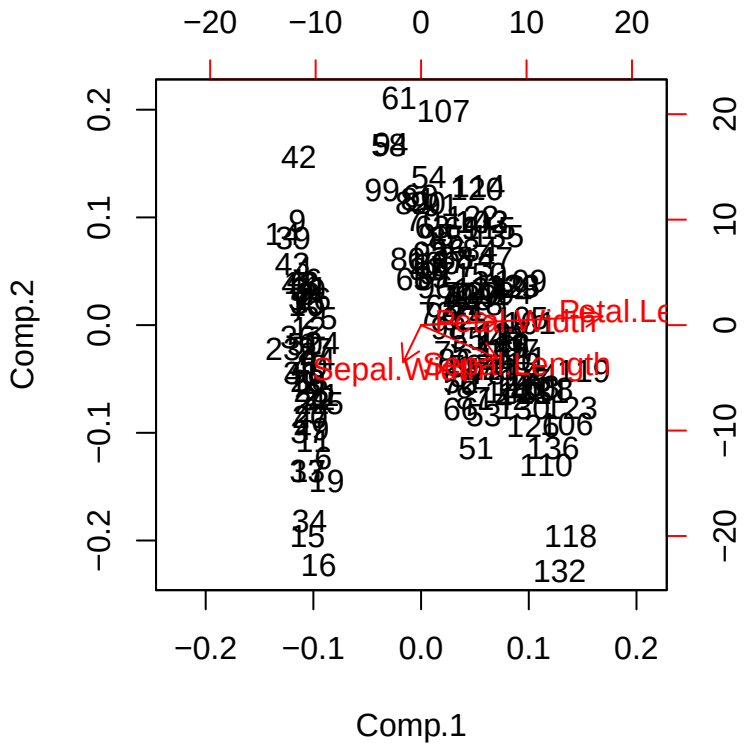

```r
biplot(PC)
```




```
biplot(PC, xlabs=Species)
```


−20
−10
0
10
20
0.2
0.1
0.0
−0.1
−0.2
20
10
0
−10
−20
Comp.2
Comp.1
−0.2
−0.1
0.0
0.1
0.2
versicolor
virginica
setosa
virginica

```
biplot(PC)
```




```
#set up a 3D plot
library(rgl)
plot3d(PC$scores[,1:3], col=as.integer(Species))

#We can perform unsupervised clustering using k-means
cl <- kmeans(iris[,1:4],3)
iris$cluster <- as.factor(cl$cluster)
#Now lets check how our clustering performed
table(iris$cluster, Species)
```

```
##    Species
##     setosa versicolor virginica
##   1     33          0         0
##   2      0         46        50
##   3     17          4         0
```

Linear Discriminant Analysis

```
# Check for collinear values
cor <- cor(iris[,1:4])
dissimilarity <- 1-abs(cor)

distance <- as.dist(dissimilarity)
```

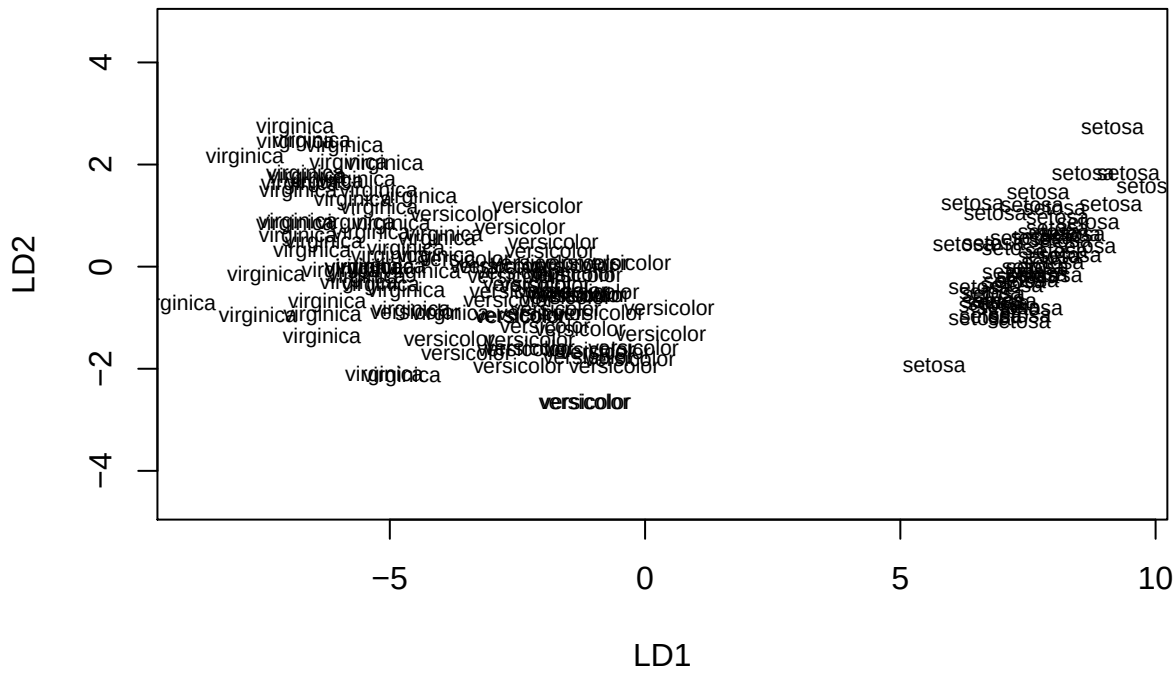
```
hc <- hclust(distance)

clusterV=cutree(hc, h=0.05)
clusterV
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##            1            2            3            3
```

```
cor(Petal.Length, Petal.Width)
```

```
## [1] 0.9628654
```

```
Petal.Width <- NULL
# The LDA
irisLDA <- lda(iris[,1:4], Species, CV=FALSE)
irisCV <- lda(iris[,1:4], Species, CV=TRUE)

plot(irisLDA)
```




```
irisLDA$counts
```

```
##     setosa versicolor  virginica
##         50         50         50
```

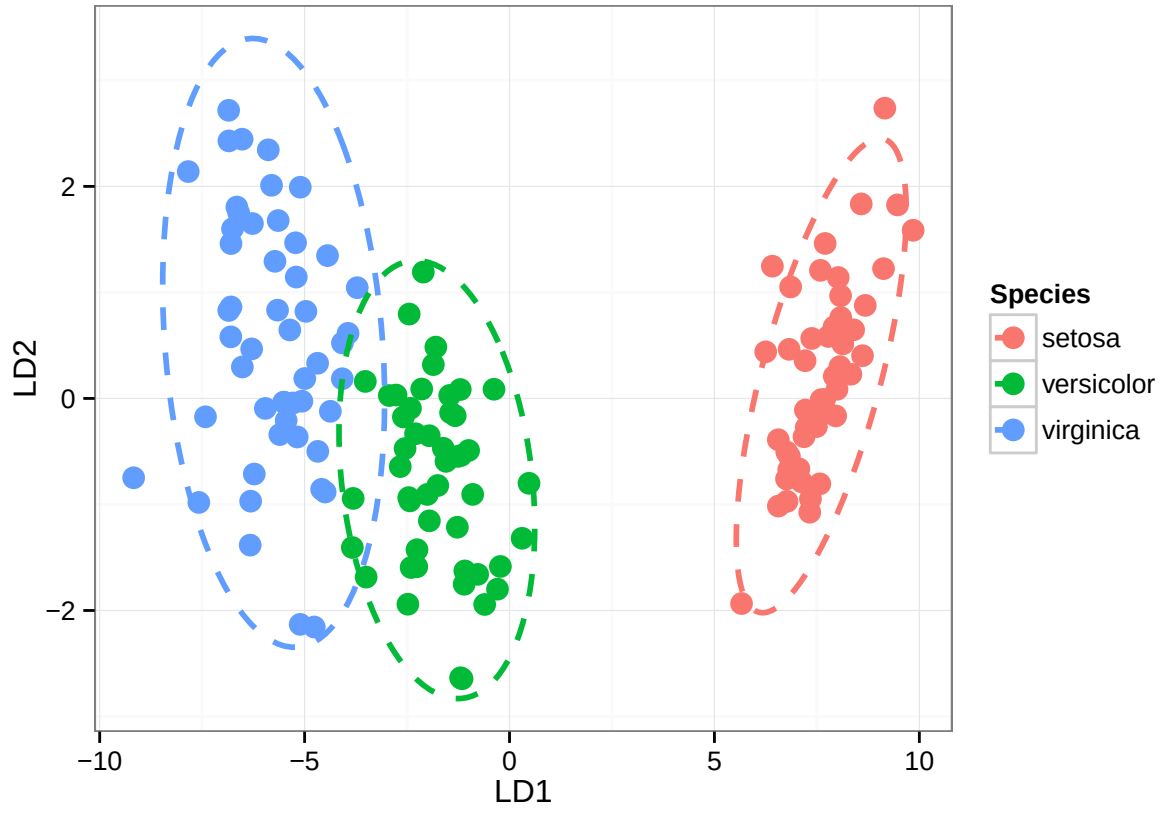
```r
#CV
table(Species, irisCV$class)
```

```
##
## Species      setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```r
pred <- predict(irisLDA)
#Percent explained by each DA
prop <- irisLDA$svd^2/sum(irisLDA$svd^2)
prop
```

```
## [1] 0.991212605 0.008787395
```

```r
#use ggplot to make a much prettier version of the LDA plot
pred<-data.frame(Species=predict(irisLDA)$class,predict(irisLDA)$x)
library(ellipse)
dat_ell <- data.frame()

for(g in levels(pred$Species)){
dat_ell <- rbind(dat_ell, cbind(as.data.frame(with(pred[pred$Species==g,], ellipse(cor(LD1, LD2), scale=

ggplot(pred, aes(x=LD1, y=LD2, col=Species) ) + geom_point( size = 4, aes(color = Species))+theme_bw()+g
```

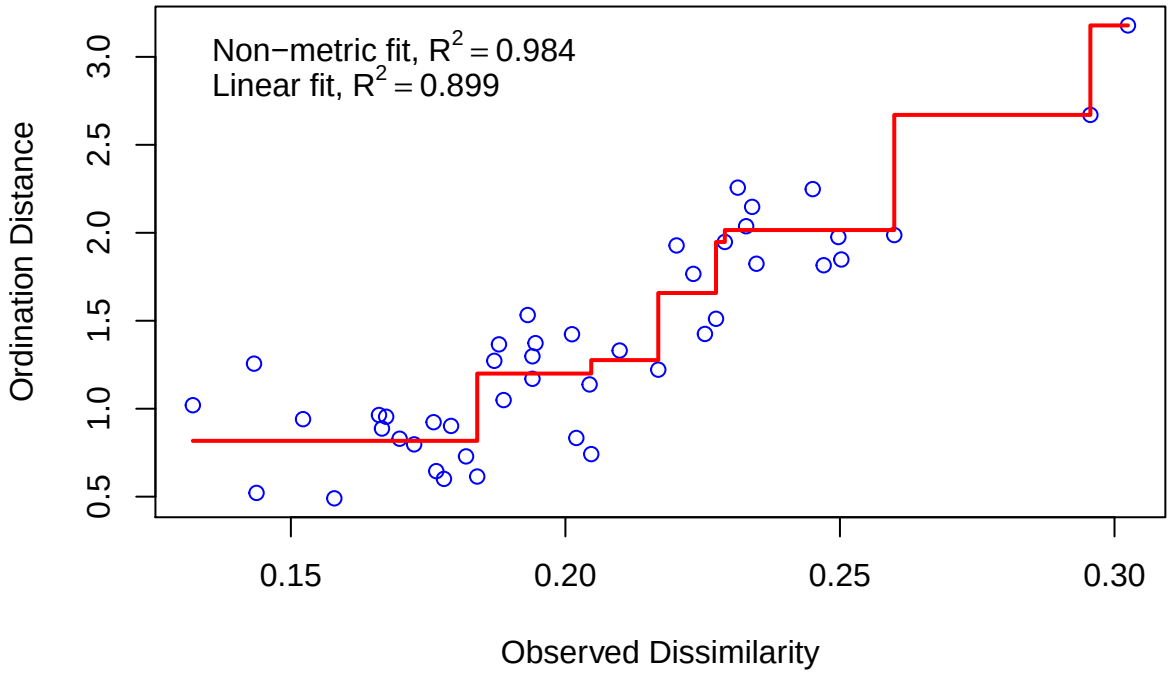


Non-metric Multidimentional Scaling (NMDS)

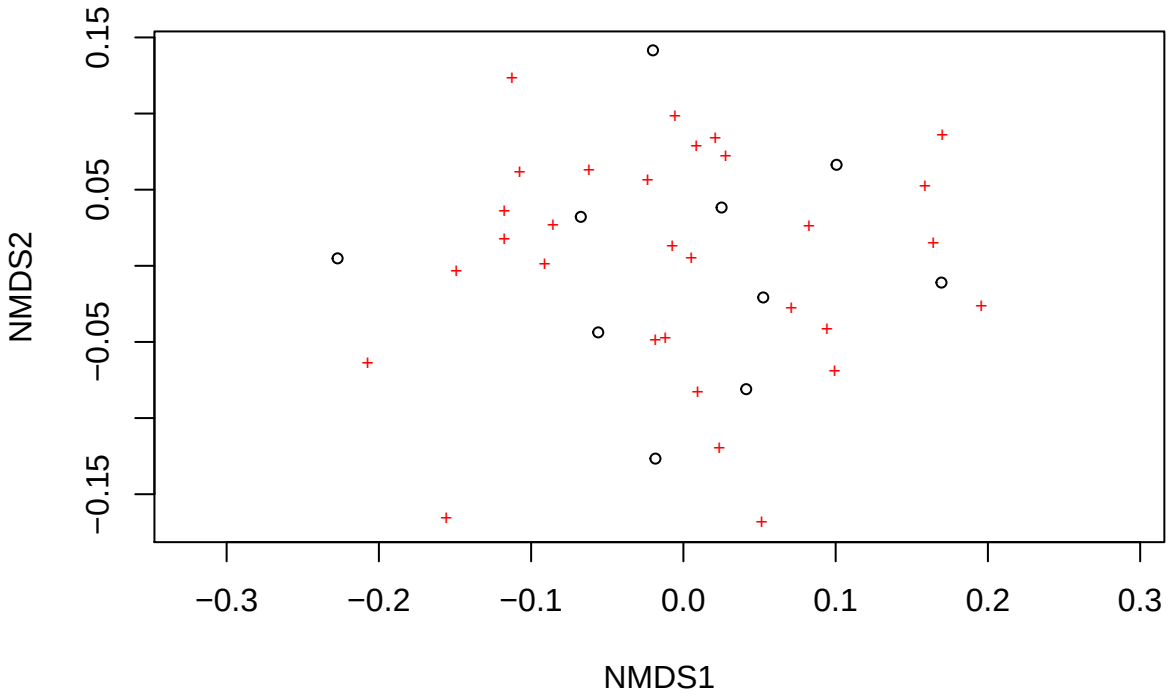
```
set.seed(2)
#set up random species data
community_matrix=matrix(
   sample(1:100,300,replace=T),nrow=10,
   dimnames=list(paste("community",1:10,sep=""),paste("sp",1:30,sep="")))

example_NMDS=metaMDS(community_matrix, # Our community-by-species matrix
                     k=2) # k = number of dimentions
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.1280709
## Run 1 stress 0.1280709
## ... procrustes: rmse 3.168062e-05  max resid 5.925658e-05
## *** Solution reached
```

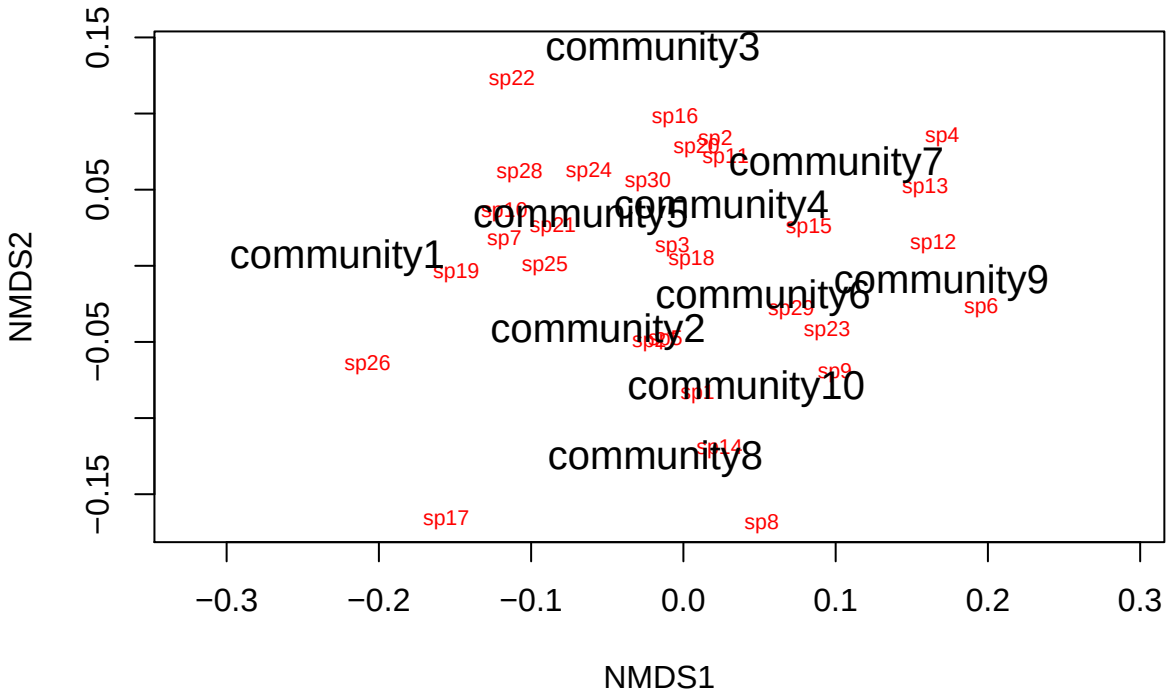
```
#stressplot - if points remain close to the line, our data retains its original differences despite the
stressplot(example_NMDS)
```



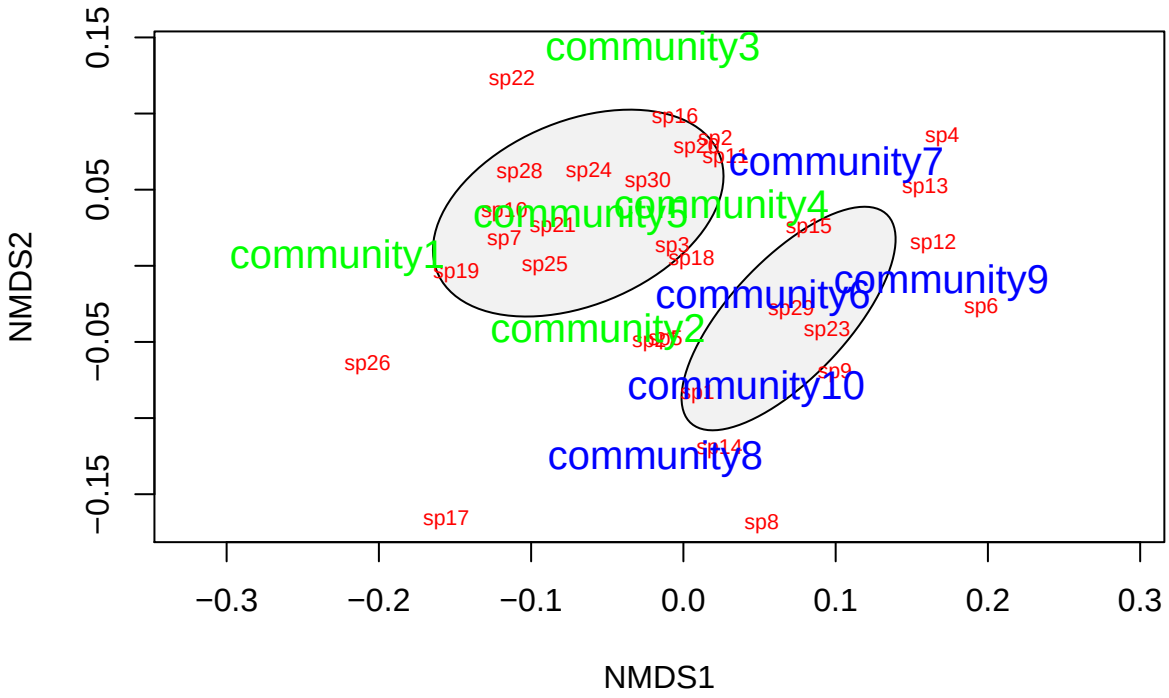

```
#plot nmds- communities ("sites") are open circles, species are red crosses
plot(example_NMDS)
```




```
#we can use ordiplot to create a plot with labels

ordiplot(example_NMDS,type="n")
orditorp(example_NMDS,display="species",col="red",air=0.01)
orditorp(example_NMDS,display="sites",cex=1.25,air=0.01)
```




```
treat=c(rep("Treatment1",5),rep("Treatment2",5))
ordiplot(example_NMDS,type="n")
ordiellipse(example_NMDS,groups=treat,draw="polygon",col="grey90",label=F)
orditorp(example_NMDS,display="species",col="red",air=0.01)
orditorp(example_NMDS,display="sites",col=c(rep("green",5),rep("blue",5)),
   air=0.01,cex=1.25)
```


community3
sp22
sp16
sp2
sp4
sp11
community7
sp13
sp28
sp24
sp30
community4
community5
sp21
sp15
sp12
community1
sp7
sp3
sp18
sp19
sp25
community9
community6
sp29
sp6
community2
sp23
sp26
sp9
community10
sp14
community8
sp17
sp8
NMDS1
NMDS2
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.15
0.05
−0.05
−0.15