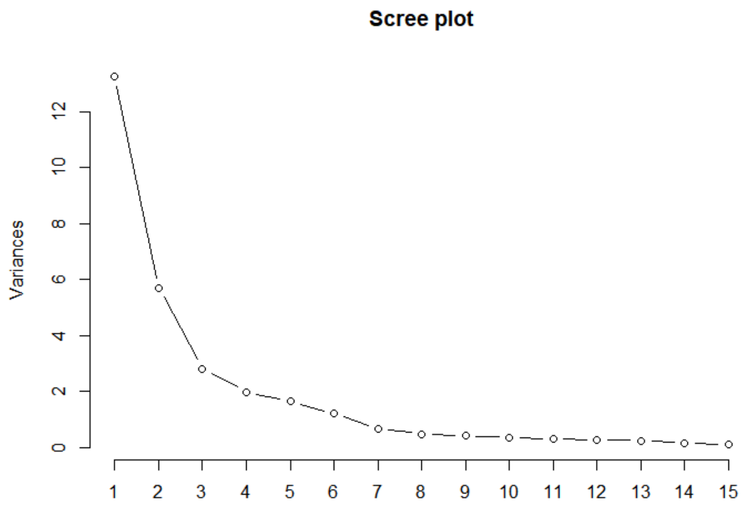
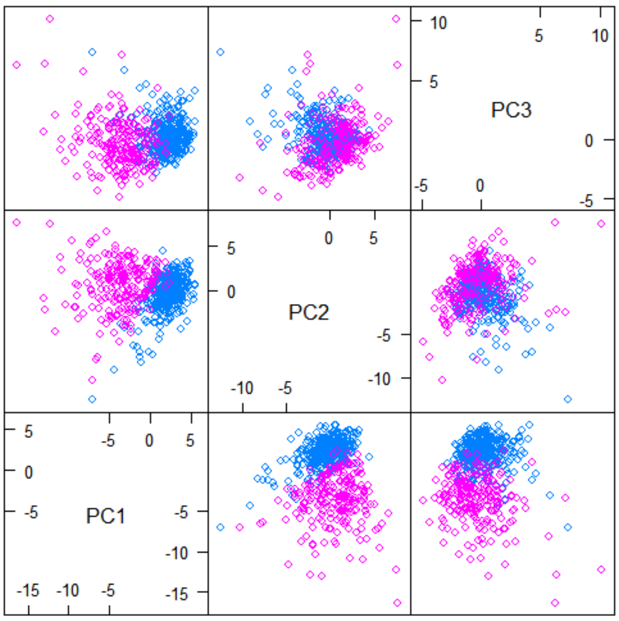
```
> ############################################
> # Statistical Modelling & Machine Learning #
> #            R Example3                    #
> ############################################
>
> options(warn = -1)  # Turn off warning message
>
> # Data: Breast Cancer Wisconsin Data
>
> dat = read.csv("wdbc.csv", header = F)
>
> x.name = c("radius", "texture", "perimeter", "area", "smoothness",
+           "compactness", "concavity", "concave_points", "symmetry",
+           "fractal_dimension")
>
> names(dat) = c("id", "diagnosis", paste0(x.name,"_mean"),
+              paste0(x.name,"_se"), paste0(x.name,"_worst"))
>
> dat = dat[,-1]
>
> head(dat)
  diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
1         M       17.99        10.38         122.80    1001.0         0.11840
2         M       20.57        17.77         132.90    1326.0         0.08474
3         M       19.69        21.25         130.00    1203.0         0.10960
4         M       11.42        20.38          77.58     386.1         0.14250
5         M       20.29        14.34         135.10    1297.0         0.10030
6         M       12.45        15.70          82.57     477.1         0.12780
  compactness_mean concavity_mean concave_points_mean symmetry_mean
fractal_dimension_mean
1          0.27760         0.3001             0.14710        0.2419                0.07871
2          0.07864         0.0869             0.07017        0.1812                0.05667
3          0.15990         0.1974             0.12790        0.2069                0.05999
4          0.28390         0.2414             0.10520        0.2597                0.09744
5          0.13280         0.1980             0.10430        0.1809                0.05883
6          0.17000         0.1578             0.08089        0.2087                0.07613
  radius_se texture_se perimeter_se area_se smoothness_se compactness_se concavity_se
1    1.0950     0.9053        8.589  153.40      0.006399        0.04904      0.05373
2    0.5435     0.7339        3.398   74.08      0.005225        0.01308      0.01860
3    0.7456     0.7869        4.585   94.03      0.006150        0.04006      0.03832
4    0.4956     1.1560        3.445   27.23      0.009110        0.07458      0.05661
5    0.7572     0.7813        5.438   94.44      0.011490        0.02461      0.05688
6    0.3345     0.8902        2.217   27.19      0.007510        0.03345      0.03672
  concave_points_se symmetry_se fractal_dimension_se radius_worst texture_worst
1           0.01587     0.03003             0.006193        25.38         17.33
2           0.01340     0.01389             0.003532        24.99         23.41
3           0.02058     0.02250             0.004571        23.57         25.53
4           0.01867     0.05963             0.009208        14.91         26.50
5           0.01885     0.01756             0.005115        22.54         16.67
6           0.01137     0.02165             0.005082        15.47         23.75
  perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
1          184.60     2019.0           0.1622            0.6656          0.7119
2          158.80     1956.0           0.1238            0.1866          0.2416
3          152.50     1709.0           0.1444            0.4245          0.4504
4           98.87      567.7           0.2098            0.8663          0.6869
5          152.20     1575.0           0.1374            0.2050          0.4000
6          103.40      741.6           0.1791            0.5249          0.5355
  concave_points_worst symmetry_worst fractal_dimension_worst
1               0.2654         0.4601                 0.11890
2               0.1860         0.2750                 0.08902
3               0.2430         0.3613                 0.08758
4               0.2575         0.6638                 0.17300
5               0.1625         0.2364                 0.07678
6               0.1741         0.3985                 0.12440
>
> # Principal Component Analysis (PCA) ----------------------------
>
> pr = prcomp(dat[,2:31], center = TRUE, scale = TRUE)
> summary(pr)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8    PC9
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172 0.69037
0.6457
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251 0.01589
0.0139
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010 0.92598
0.9399
                          PC10   PC11    PC12    PC13    PC14    PC15    PC16    PC17
Standard deviation     0.59219 0.5421 0.51104 0.49128 0.39624 0.30681 0.28260 0.24372
Proportion of Variance 0.01169 0.0098 0.00871 0.00805 0.00523 0.00314 0.00266 0.00198
Cumulative Proportion  0.95157 0.9614 0.97007 0.97812 0.98335 0.98649 0.98915 0.99113
                          PC18    PC19    PC20   PC21    PC22    PC23   PC24    PC25    PC26
Standard deviation     0.22939 0.22244 0.17652 0.1731 0.16565 0.15602 0.1344 0.12442
0.09043
Proportion of Variance 0.00175 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006 0.00052
0.00027
Cumulative Proportion  0.99288 0.99453 0.99557 0.9966 0.99749 0.99830 0.9989 0.99942
0.99969
                          PC27    PC28    PC29    PC30
Standard deviation     0.08307 0.03987 0.02736 0.01153
Proportion of Variance 0.00023 0.00005 0.00002 0.00000
Cumulative Proportion  0.99992 0.99997 1.00000 1.00000
>
> # Scree plot
> screeplot(pr, type = 'l', npcs = 15, main = 'Scree plot')
```

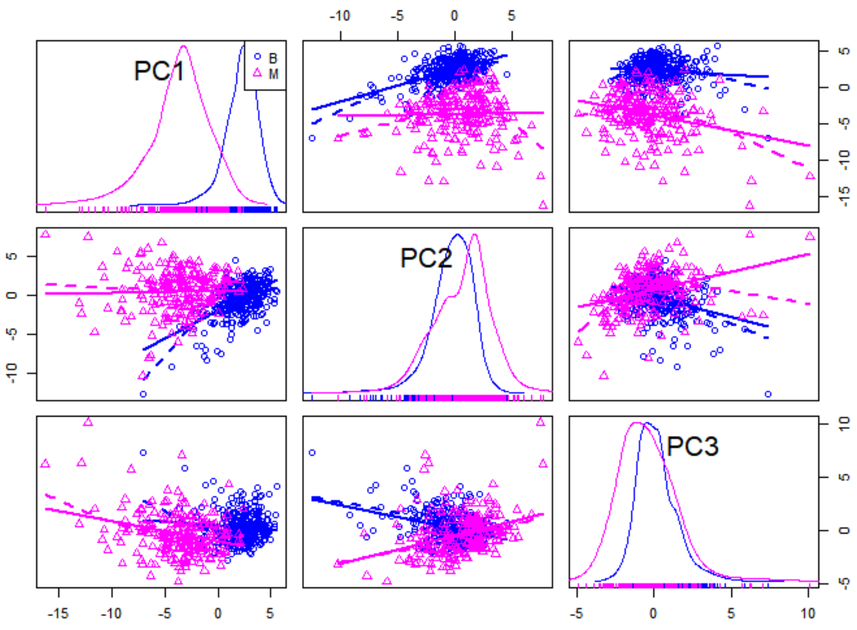


```
> # Visulalization: Scatter plot matrix
>
> library(lattice)
>
> pc.dat = data.frame(type = dat$diagnosis,pr$x[,1:3])
>
> pc.dat$type = as.factor(pc.dat$type)
>
> splom(~pc.dat[,2:4], groups=type, data=pc.dat, panel=panel.superpose)
```



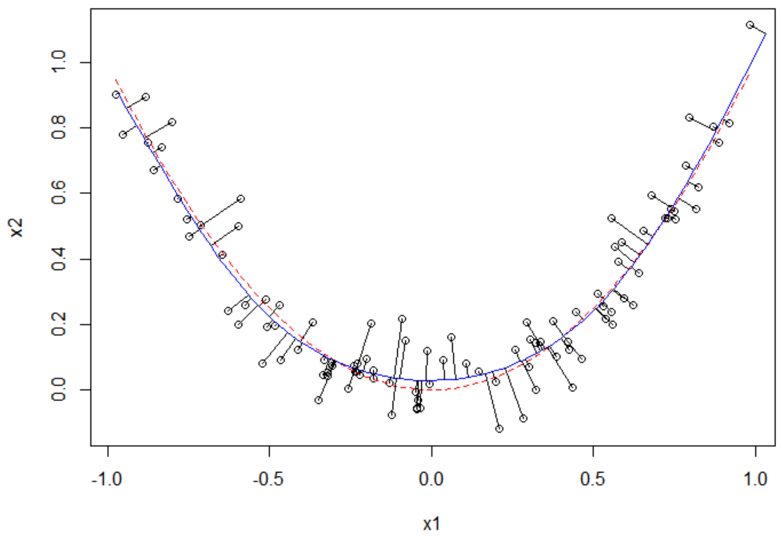
Scatter Plot Matrix

```
> install.packages('car')
> library(car)
> scatterplotMatrix(~PC1+PC2+PC3|type, data=pc.dat)
```

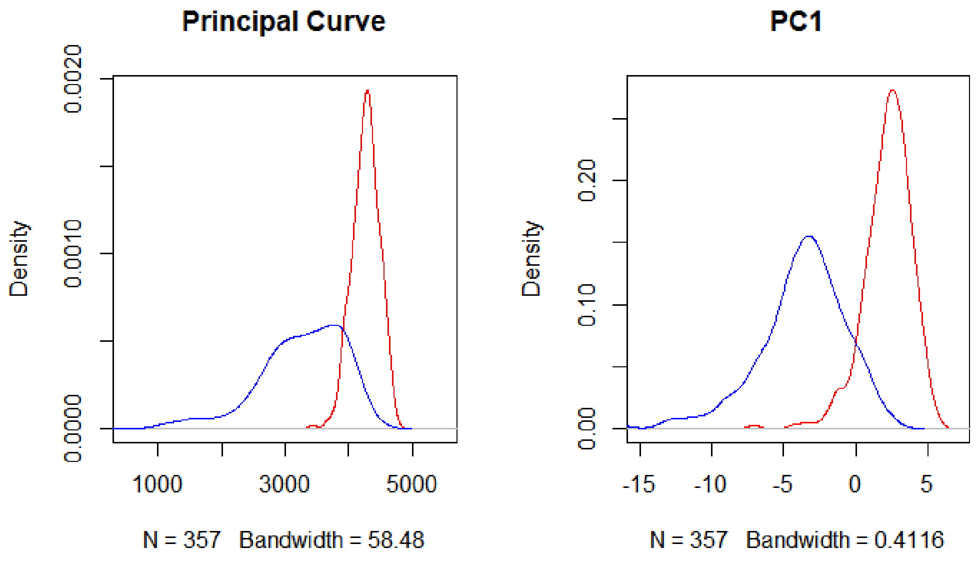


```
> # Application to logistic regression:
> install.packages('boot')
> library(boot)
>
>
> dat$diagnosis = as.factor(dat$diagnosis)
>
> # Comparison of CV statistics:
> set.seed(1)
> fit = glm(diagnosis~., data=dat, family=binomial)
> cv.glm(dat, fit, K=5)$delta
[1] 0.05982303 0.82816919
>
```

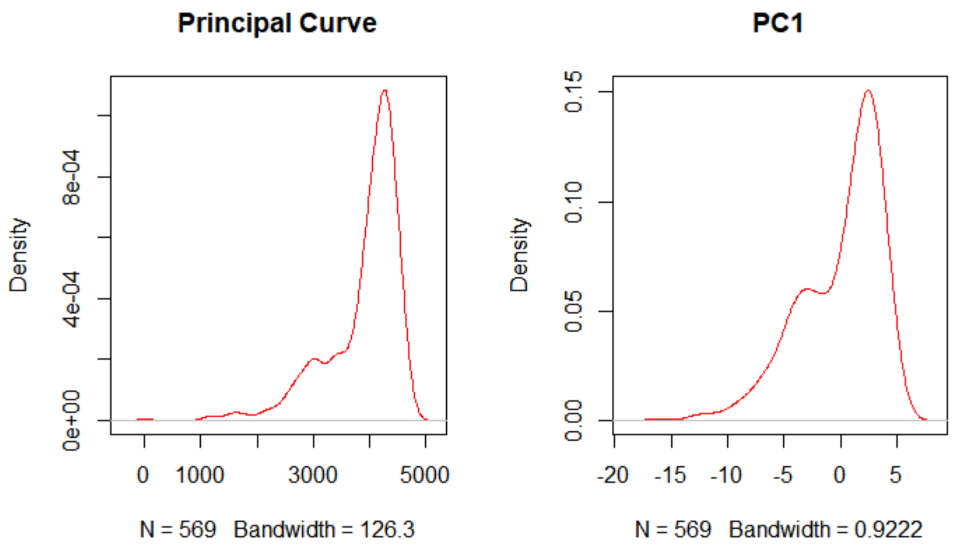
```
> fit1 = glm(type~., data=pc.dat, family=binomial)
> cv.glm(pc.dat, fit1, K=5)$delta
[1] 0.03728052 0.03671780
>
> # Principal Curve --------------------------------------------------
> install.packages('princurve')
> library(princurve)
>
> # Simple example
> set.seed(1)
> n=100
> x1 = runif(n,-1,1)
> x2 = x1^2 + rnorm(n, sd = 0.1)
> z = cbind(x1,x2)
>
> cor(x1,x2)
[1] 0.08086294
>
> fit = principal_curve(z)
>
> plot(x1,x2)
> lines(sort(x1),(sort(x1))^2,col='red',lty=2)
> lines(fit,col='blue')
> whiskers(z, fit$s)
```



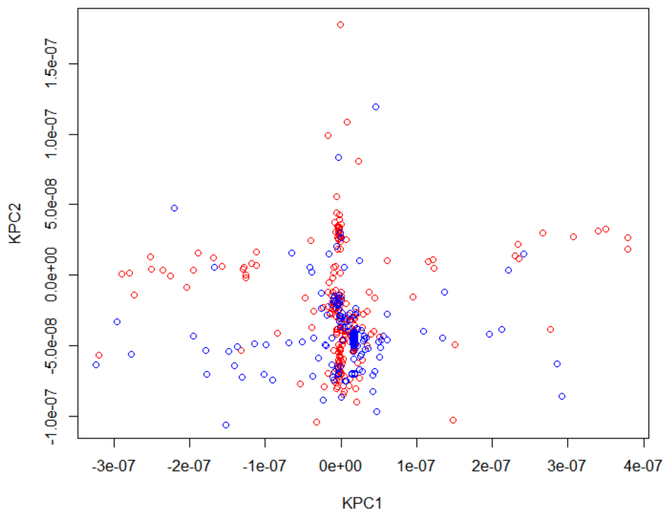
```
> # WDBC data application:
>
> fit = principal_curve(as.matrix(dat[,2:31]))
>
> # Density of two groups in principal curve and PC1
> par(mfrow=c(1,2))
> plot(density(fit$lambda[dat$diagnosis == 'B']),col='red',
+     xlim=c(500,5500),main='Principal Curve')
> lines(density(fit$lambda[dat$diagnosis == 'M']),col='blue')
>
> plot(density(pc.dat[dat$diagnosis == 'B',2]),col='red',
+     xlim=c(-15,7),main='PC1')
> lines(density(pc.dat[dat$diagnosis == 'M',2]),col='blue')
```



```
> # Density for principal curve and PC1
> par(mfrow=c(1,2))
> plot(density(fit$lambda),col='red',main='Principal Curve')
> plot(density(pc.dat[,2]),col='red',main='PC1')
```



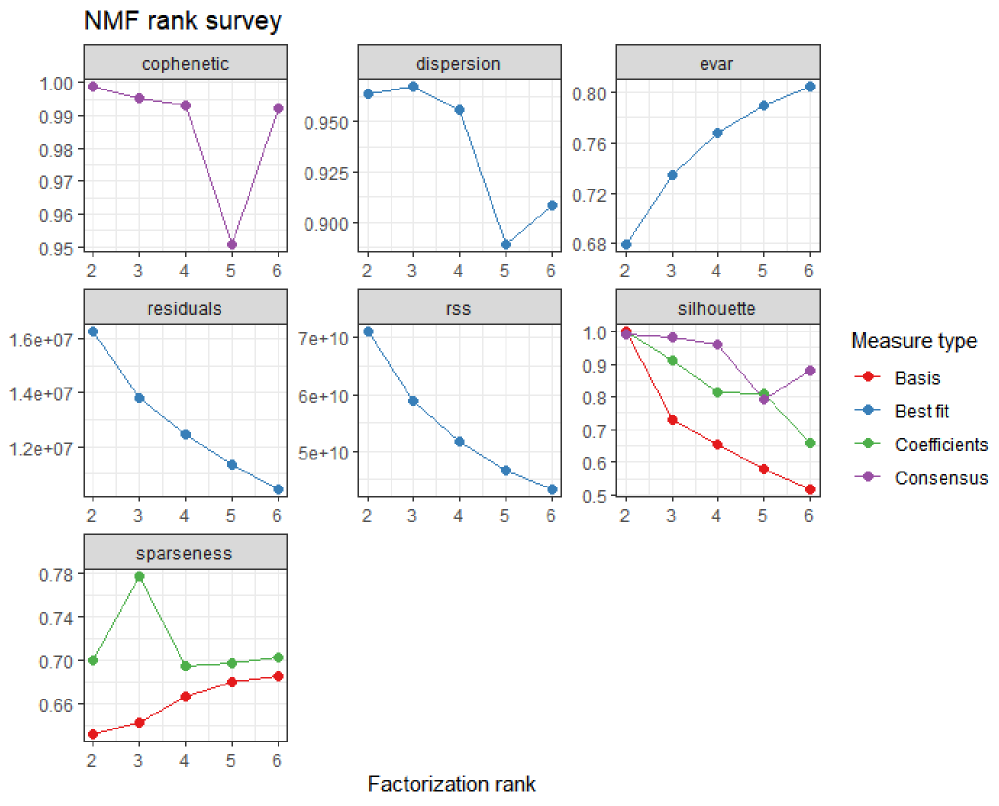
```
>
> dat1 = cbind(dat, pcurve=fit$lambda)
> dat1$diagnosis = as.factor(dat1$diagnosis)
>
> fit2 = glm(diagnosis~pcurve, data=dat1, family=binomial)
> cv.glm(dat1, fit2, K=5)$delta
[1] 0.06816360 0.06812241
>
> # Projection onto Principal curve
> new.obs = as.matrix(dat[31:40,2:31])
> project_to_curve(new.obs,fit$s)$lambda
       31        32        33        34        35        36        37        38 
4000.6148 2994.0477 1240.4169 1777.8542 2435.6018 2478.3771    0.0000  271.0701 
       39        40 
 496.7614  544.8027 
> # arc-length along the curve.
> # Kernel PCA ---------------------------------------------------------
> install.packages('kernlab')
> library(kernlab)
>
> x = dat[,2:31]
> fit = kpca(~., data=x, kernel='rbfdot', kpar=list(sigma=3), features=2)
> # feature: # of PC's
>
> # Kernel PC's
> pc = pcv(fit)
>
> B = pc[dat$diagnosis=='B',]
> M = pc[dat$diagnosis=='M',]
>
> par(mfrow=c(1,1))
> plot(B, col='red', xlab='KPC1',ylab='KPC2')
> points(M, col='blue')
```



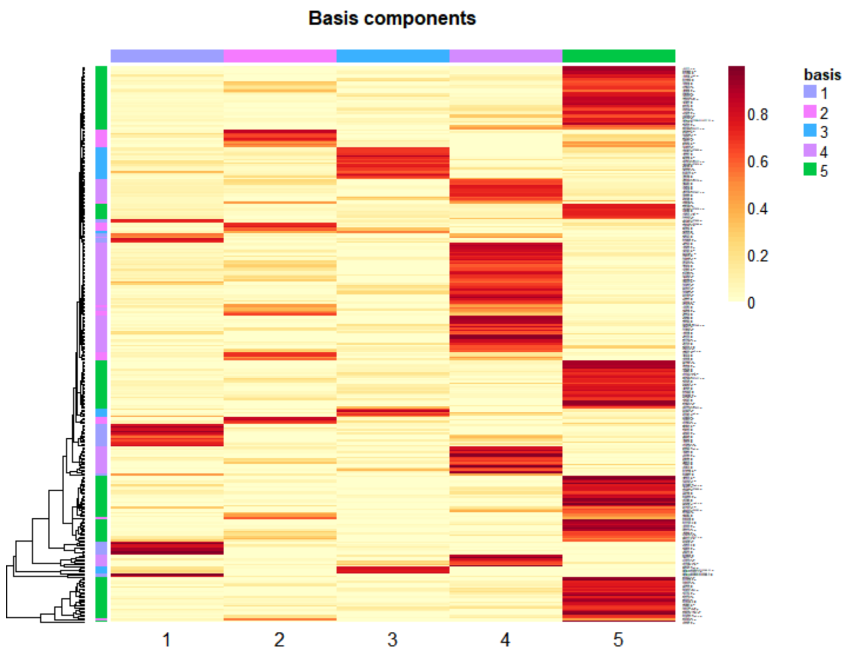
```
>
> # New observations
> predict(fit, new.obs)
             [,1]          [,2]
31  1.672321e-08 -5.958310e-08
32 -2.204971e-07  4.754435e-08
33 -3.603358e-08 -4.478933e-08
34 -3.231095e-07 -6.363222e-08
35  5.273944e-08 -5.169283e-08
36  3.538922e-08 -5.199263e-08
37  4.219275e-08 -7.082468e-08
38  1.504703e-07 -4.966250e-08
39 -9.474597e-07 -5.998169e-08
40 -3.543559e-06 -8.392185e-08


> # Non-negative matrix factorization ------------------------------------
>
> if (!requireNamespace("BiocManager", quietly = TRUE))
+   install.packages("BiocManager")
> BiocManager::install("Biobase")
>
> install.packages('NMF')
> library(NMF)
>
> ?esGolub
>
> data(esGolub)
>
```

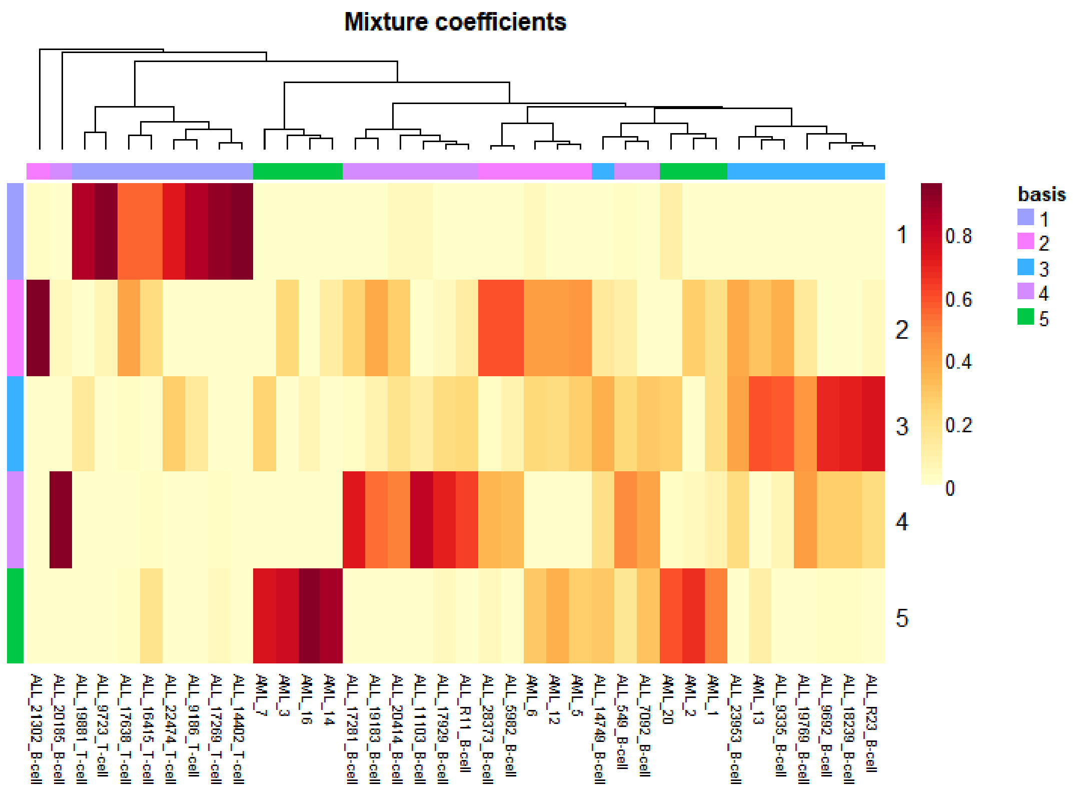
```
> dim(esGolub)
Features  Samples 
    5000       38 
> 
> res = nmf(esGolub, rank = 3, seed=123456)
> 
> # W matrix
> W = basis(res)
> dim(W)
[1] 5000    3
> 
> # H matrix
> H = coef(res)
> dim(H)
[1]  3 38
> 
> 
> if(requireNamespace("Biobase", quietly=TRUE))
+ {
+   estim.r = nmf(esGolub, 2:6, nrun=10, seed=123456)
+   plot(estim.r)
+ }
```



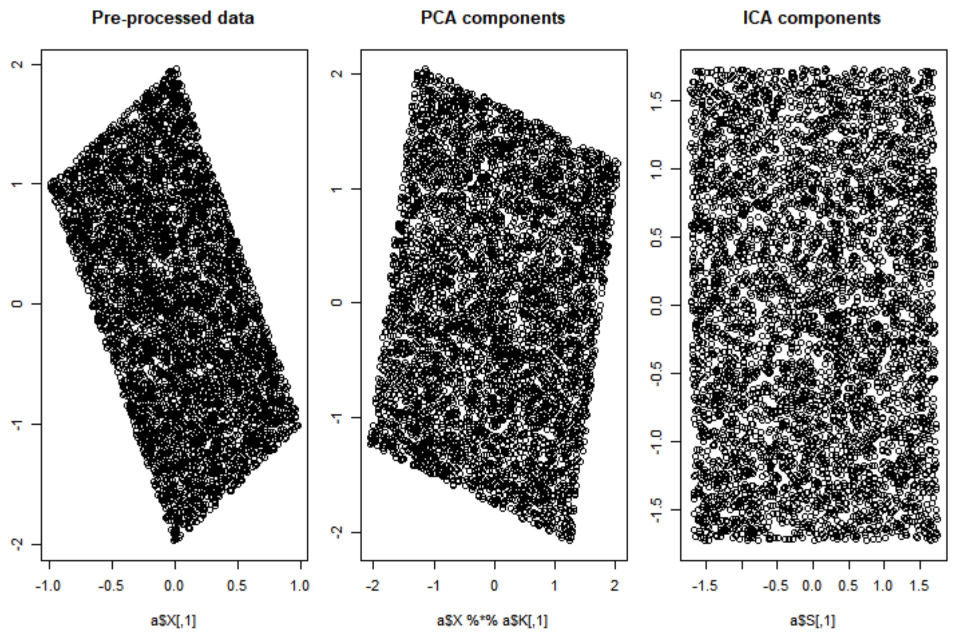
```
> 
> res = nmf(esGolub, rank = 5, seed=123456)
> 
> # Visualization
> basismap(res, subsetRow=TRUE)
```



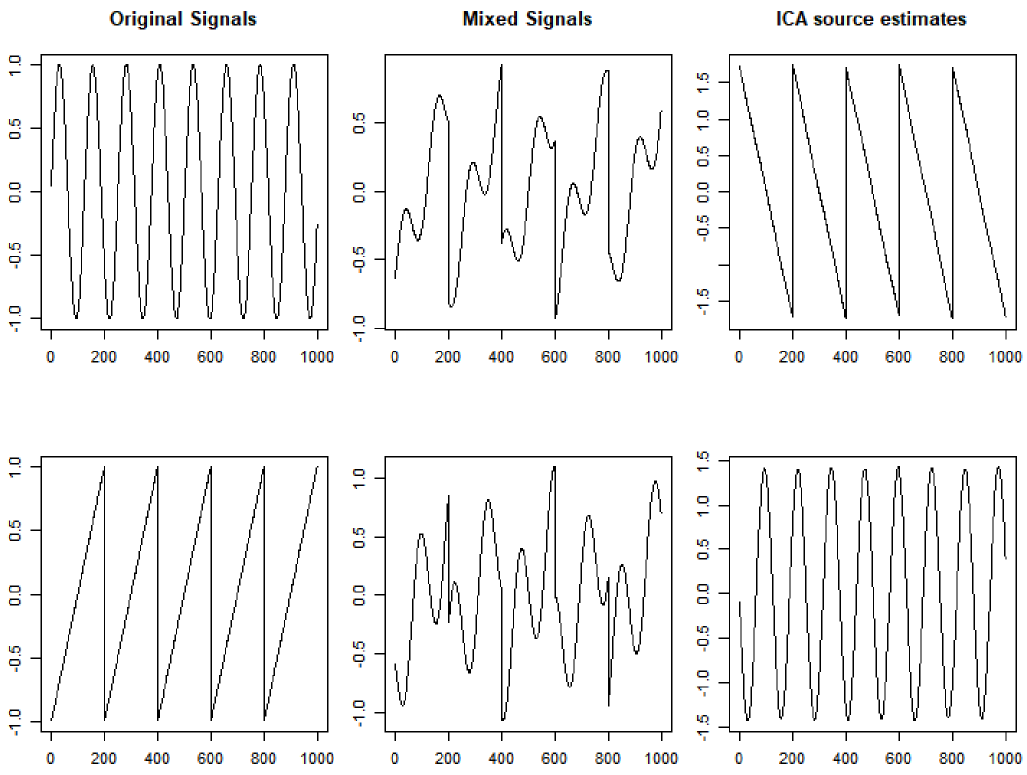
```
>
> coefmap(res)
```



```
>
> # Independent component analysis -----------------------------------
>
> install.packages('fastICA')
> library(fastICA)
>
>
> # Ex1:
>
> S = matrix(runif(10000), 5000, 2)
> A = matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
> X = S %*% A
> a = fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
+             method = "C", row.norm = FALSE, maxit = 200,
+             tol = 0.0001, verbose = TRUE)
Centering
Whitening
Symmetric FastICA using logcosh approx. to neg-entropy function
Iteration 1 tol=0.104526
Iteration 2 tol=0.003510
Iteration 3 tol=0.000001
> par(mfrow = c(1, 3))
> plot(a$X, main = "Pre-processed data")
> plot(a$X %*% a$K, main = "PCA components")
> plot(a$S, main = "ICA components")
```



```
> # Ex2:
>
> S = cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5))
> A = matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
> X = S %*% A
> a = fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
+             method = "R", row.norm = FALSE, maxit = 200,
+             tol = 0.0001, verbose = TRUE)
Centering
Whitening
Symmetric FastICA using logcosh approx. to neg-entropy function
Iteration 1 tol = 0.004695925
Iteration 2 tol = 4.449089e-06
>
> par(mfcol = c(2, 3))
> plot(1:1000, S[,1], type = "l", main = "Original Signals",
+      xlab = "", ylab = "")
> plot(1:1000, S[,2], type = "l", xlab = "", ylab = "")
> plot(1:1000, X[,1], type = "l", main = "Mixed Signals",
+      xlab = "", ylab = "")
> plot(1:1000, X[,2], type = "l", xlab = "", ylab = "")
> plot(1:1000, a$S[,1], type = "l", main = "ICA source estimates",
+      xlab = "", ylab = "")
> plot(1:1000, a$S[, 2], type = "l", xlab = "", ylab = "")
```


Original Signals
Mixed Signals
ICA source estimates