Principal Component Analysis

Jayanth Rasamsetti
Founder www.sgmoid.com
Columbia University (MS)
IIT Madras (B.Tech & M.Tech)

Unsupervised Techniques

1. No class/target variable
2. Useful in pre-processing steps

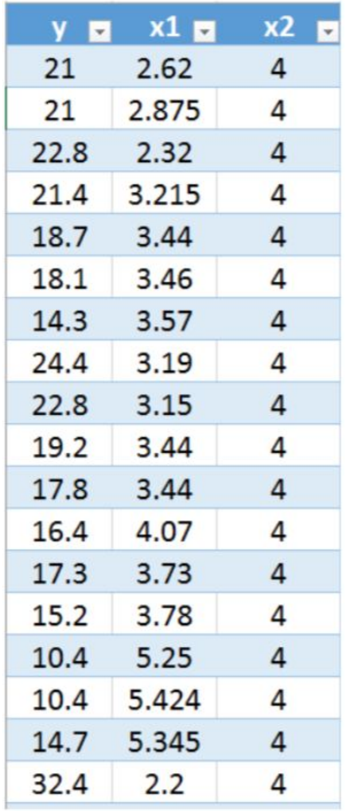
Algorithmic Trading

1. Analyze patterns in economic data & past stock prices to forecast future price movements
2. Target Variable: Stock Price Direction (Up or Down)
3. Predictor Variables: 352 different variables quantifying economic data and price movements
4. Approx 14 Hours to do the simulation

Algorithmic Trading

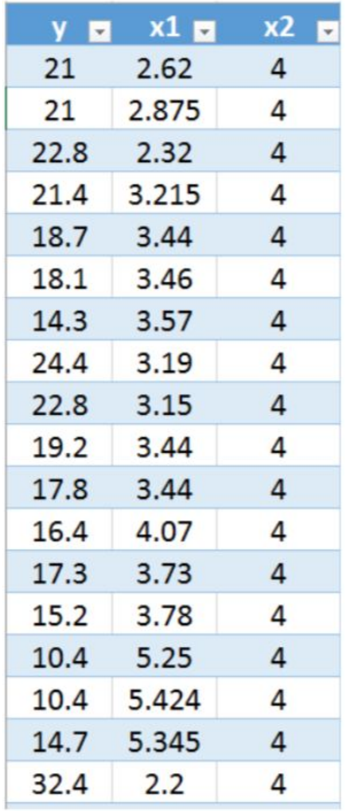
Is there a way to figure out which variables are important and which are not essential?

Which is the relevant variable?



y	x1	x2
21	2.62	4
21	2.875	4
22.8	2.32	4
21.4	3.215	4
18.7	3.44	4
18.1	3.46	4
14.3	3.57	4
24.4	3.19	4
22.8	3.15	4
19.2	3.44	4
17.8	3.44	4
16.4	4.07	4
17.3	3.73	4
15.2	3.78	4
10.4	5.25	4
10.4	5.424	4
14.7	5.345	4
32.4	2.2	4

Which is the relevant variable?

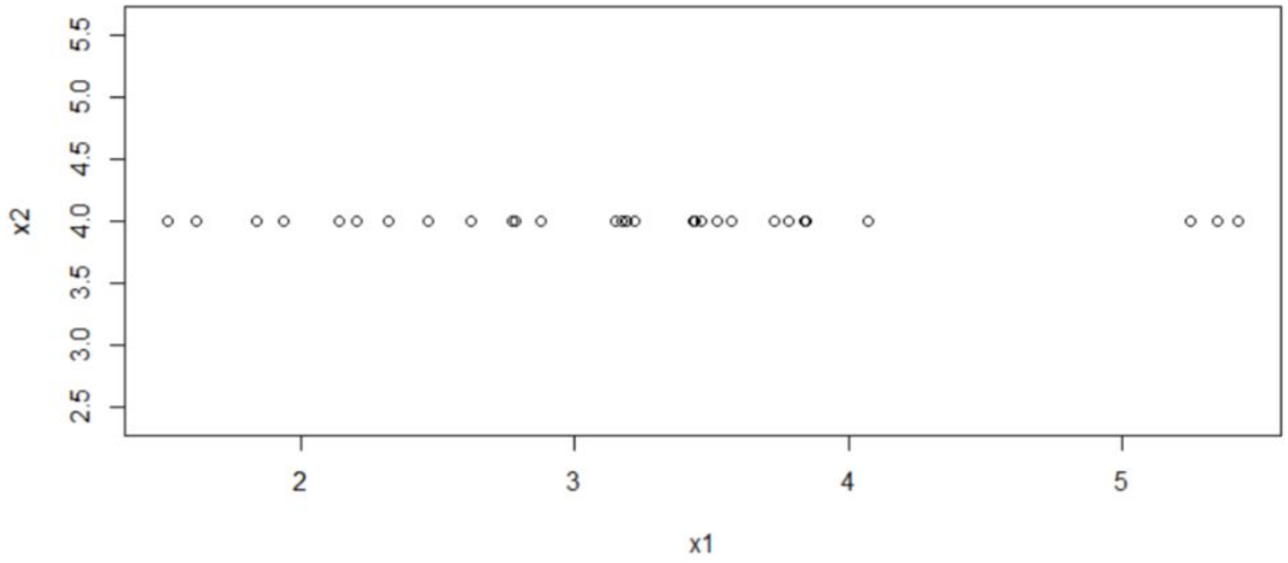


y	x1	x2
21	2.62	4
21	2.875	4
22.8	2.32	4
21.4	3.215	4
18.7	3.44	4
18.1	3.46	4
14.3	3.57	4
24.4	3.19	4
22.8	3.15	4
19.2	3.44	4
17.8	3.44	4
16.4	4.07	4
17.3	3.73	4
15.2	3.78	4
10.4	5.25	4
10.4	5.424	4
14.7	5.345	4
32.4	2.2	4

There is no variation in x2. All the variation is only along x1. So y can not be explained by x2.

$$sd(x_1) = 0.98$$
$$sd(x_2) = 0$$

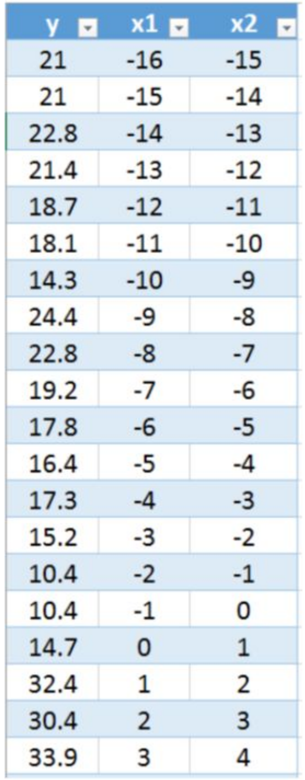
We didn't even need to look at y to make the conclusion!



All the variation in the explanatory variables is in x1 direction only.

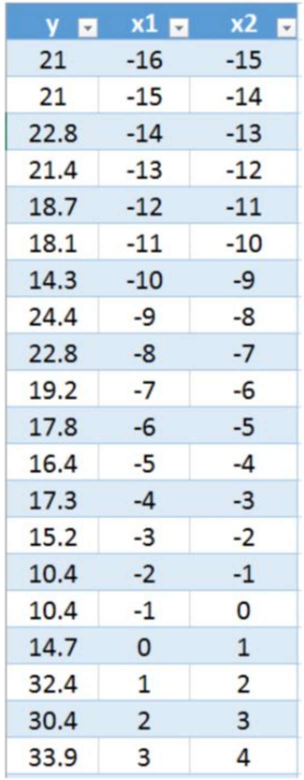
This direction is called the dominant Principal Component of the explanatory variables.
x2 direction is redundant, since there is no variation along that axis.

How many relevant features are here?

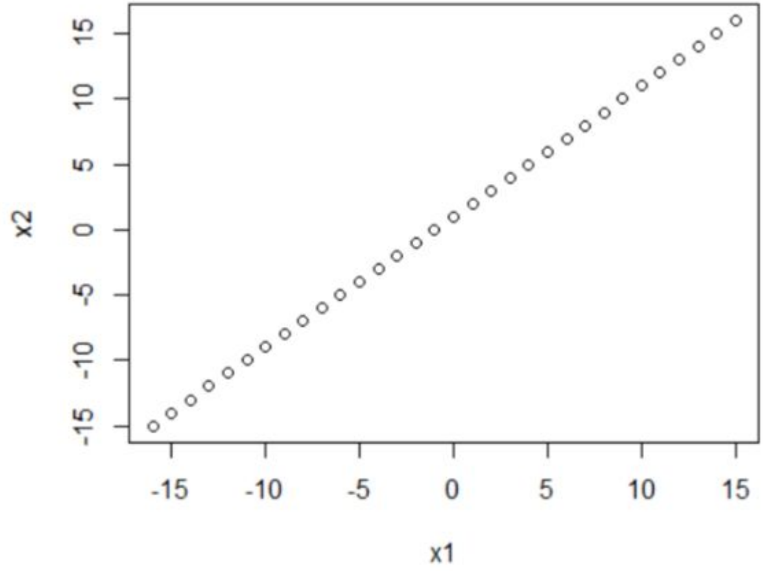


y	x1	x2
21	-16	-15
21	-15	-14
22.8	-14	-13
21.4	-13	-12
18.7	-12	-11
18.1	-11	-10
14.3	-10	-9
24.4	-9	-8
22.8	-8	-7
19.2	-7	-6
17.8	-6	-5
16.4	-5	-4
17.3	-4	-3
15.2	-3	-2
10.4	-2	-1
10.4	-1	0
14.7	0	1
32.4	1	2
30.4	2	3
33.9	3	4

How many relevant features are here?

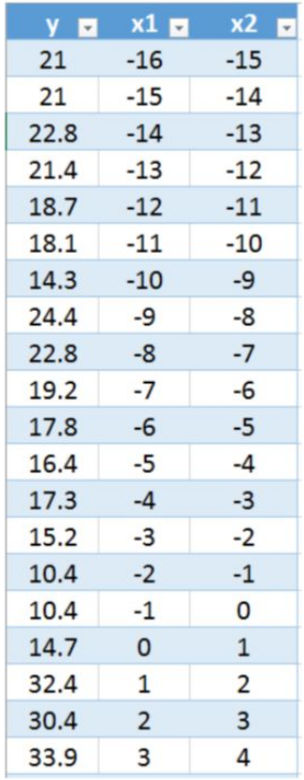


y	x1	x2
21	-16	-15
21	-15	-14
22.8	-14	-13
21.4	-13	-12
18.7	-12	-11
18.1	-11	-10
14.3	-10	-9
24.4	-9	-8
22.8	-8	-7
19.2	-7	-6
17.8	-6	-5
16.4	-5	-4
17.3	-4	-3
15.2	-3	-2
10.4	-2	-1
10.4	-1	0
14.7	0	1
32.4	1	2
30.4	2	3
33.9	3	4



This seems to have variation in both x1 & x2.
However, the data points in x1 & x2 seem correlated.

How many relevant features are here?



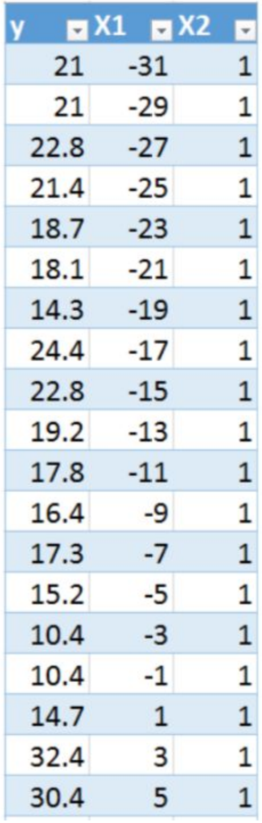
y	x1	x2
21	-16	-15
21	-15	-14
22.8	-14	-13
21.4	-13	-12
18.7	-12	-11
18.1	-11	-10
14.3	-10	-9
24.4	-9	-8
22.8	-8	-7
19.2	-7	-6
17.8	-6	-5
16.4	-5	-4
17.3	-4	-3
15.2	-3	-2
10.4	-2	-1
10.4	-1	0
14.7	0	1
32.4	1	2
30.4	2	3
33.9	3	4

If we create new variables

$$X_1 = x_2 + x_1$$

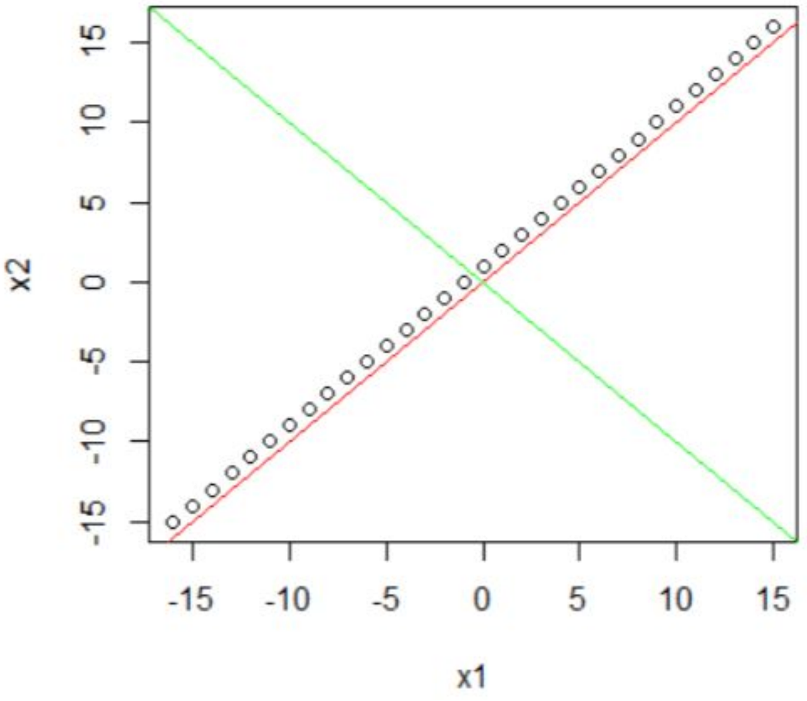
$$X_2 = x_2 - x_1$$

In terms of new variables X1 & X2 it is clear that there is only one true relevant feature.



y	X1	X2
21	-31	1
21	-29	1
22.8	-27	1
21.4	-25	1
18.7	-23	1
18.1	-21	1
14.3	-19	1
24.4	-17	1
22.8	-15	1
19.2	-13	1
17.8	-11	1
16.4	-9	1
17.3	-7	1
15.2	-5	1
10.4	-3	1
10.4	-1	1
14.7	1	1
32.4	3	1
30.4	5	1

Transformed variables: Interpretation



The transformation we did is equivalent to viewing from the data points from a rotated coordinate system.

Red=X1 axis
Green = X2 axis

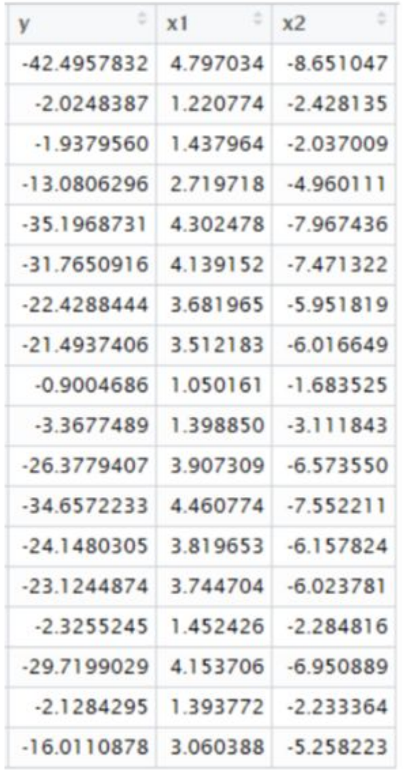
The dominant Principal component is X1 axis

Principal component Analysis

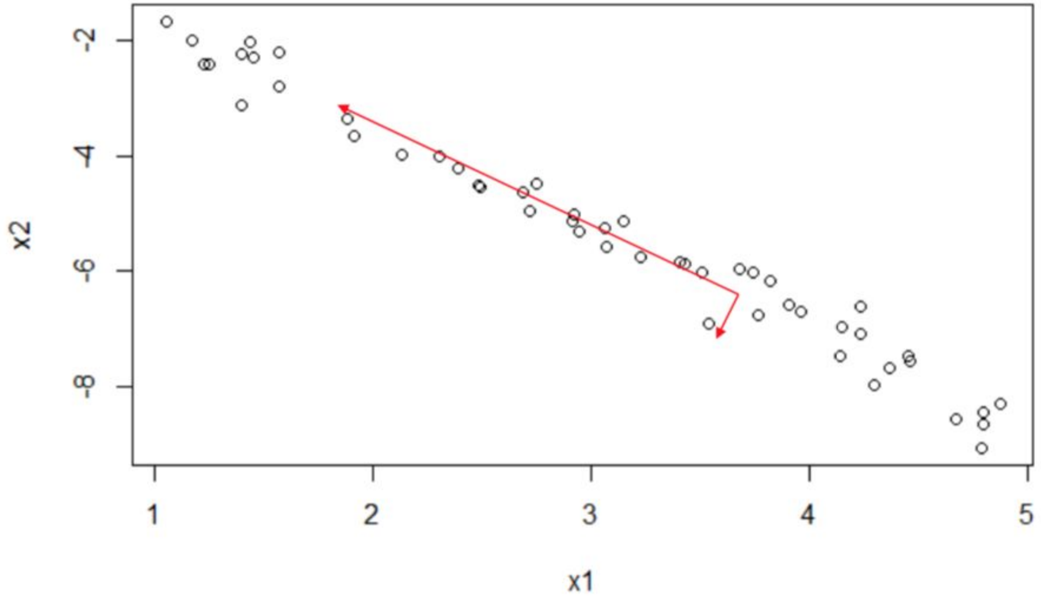
PCA is the method which allows you to identify the “directions” in which most of the variations in the data is present

Equivalently, it can be thought as method to identify the “directions” along which there is least variations (or least useful info). Identifying this would allow us to drop this irrelevant directions in our regressions/model building

Principal component directions



y	x1	x2
-42.4957832	4.797034	-8.651047
-2.0248387	1.220774	-2.428135
-1.9379560	1.437964	-2.037009
-13.0806296	2.719718	-4.960111
-35.1968731	4.302478	-7.967436
-31.7650916	4.139152	-7.471322
-22.4288444	3.681965	-5.951819
-21.4937406	3.512183	-6.016649
-0.9004686	1.050161	-1.683525
-3.3677489	1.398850	-3.111843
-26.3779407	3.907309	-6.573550
-34.6572233	4.460774	-7.552211
-24.1480305	3.819653	-6.157824
-23.1244874	3.744704	-6.023781
-2.3255245	1.452426	-2.284816
-29.7199029	4.153706	-6.950889
-2.1284295	1.393772	-2.233364
-16.0110878	3.060388	-5.258223



Can you identify the Principal Components?

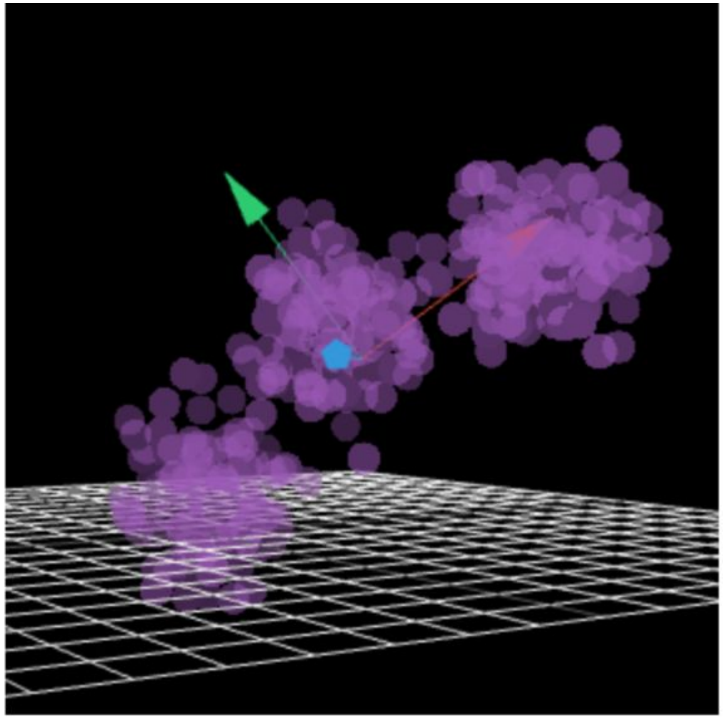
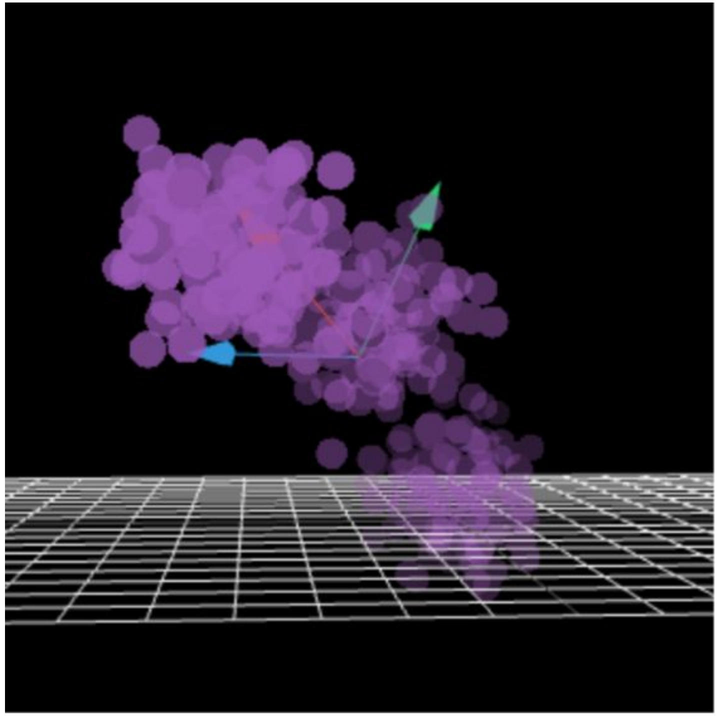
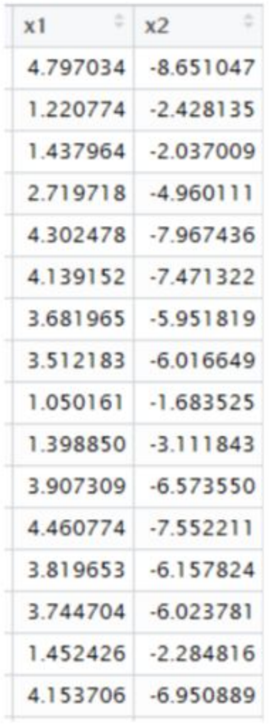



Image from: http://setosa.io/ev/principal-component-analysis/

Principal Component Analysis

1. PCA is a way for identifying the directions of largest to smallest variance
2. It also gives a transformation to transform the data into those coordinate system
3. Once we transform the data in to those coordinate system, we can choose to drop the variables which do not have much variance
4. This gives a way to reduce dimensions and focus on ones with largest variance

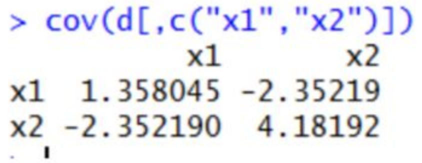
PCA Methodology: In Class exercise: Please rework in Python (5 min)



x1	x2
4.797034	-8.651047
1.220774	-2.428135
1.437964	-2.037009
2.719718	-4.960111
4.302478	-7.967436
4.139152	-7.471322
3.681965	-5.951819
3.512183	-6.016649
1.050161	-1.683525
1.398850	-3.111843
3.907309	-6.573550
4.460774	-7.552211
3.819653	-6.157824
3.744704	-6.023781
1.452426	-2.284816
4.153706	-6.950889

Start with analysis the covariance matrix of the features

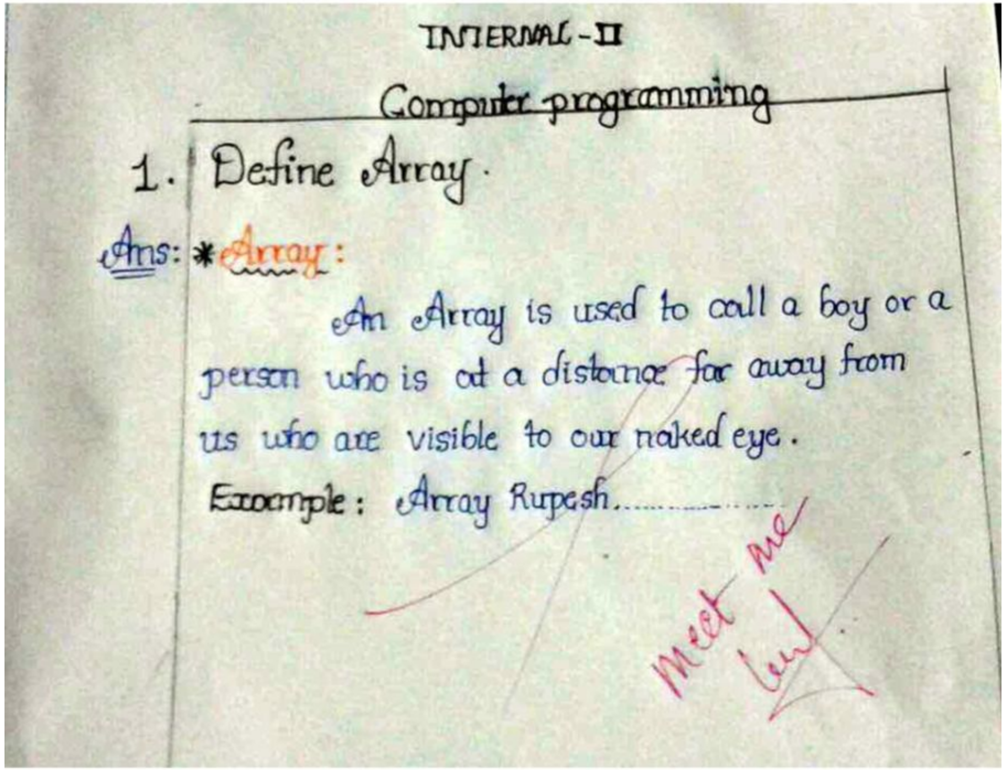
$$C = \begin{bmatrix} cov(x_1, x_1) & cov(x_1, x_2) \\ cov(x_2, x_1) & cov(x_2, x_2) \end{bmatrix}$$



```
> cov(d[,c("x1","x2")])
          x1        x2
x1  1.358045 -2.35219
x2 -2.352190  4.18192
```

The covariance matrix contains information about both correlations and the “special directions” of maximal variances

Array! Matrix! And other nightmares!



INTERNAL - II

Computer programming

1. Define Array.

Ans: * Array :

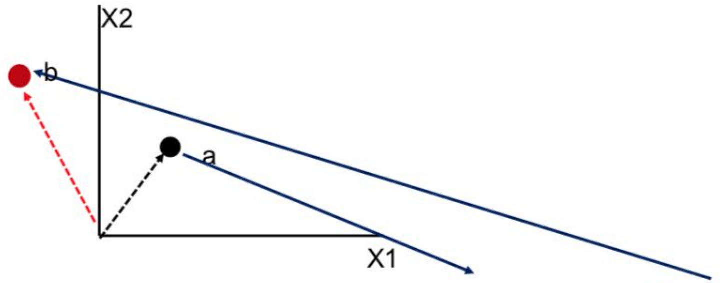
An Array is used to call a boy or a person who is at a distance far away from us who are visible to our naked eye.

Example : Array Rupesh..............

meet me

[illegible]

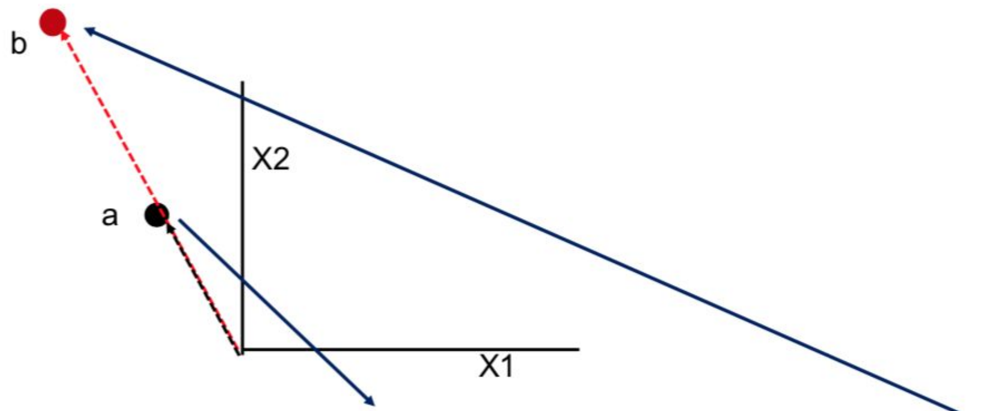
Matrix as a transformation on a vector



$$\begin{bmatrix} 1.36 & -2.35 \\ -2.35 & 4.18 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.99 \\ 1.83 \end{bmatrix}$$

Action of a matrix on a general vector can be thought of as a combination of stretch and rotation.

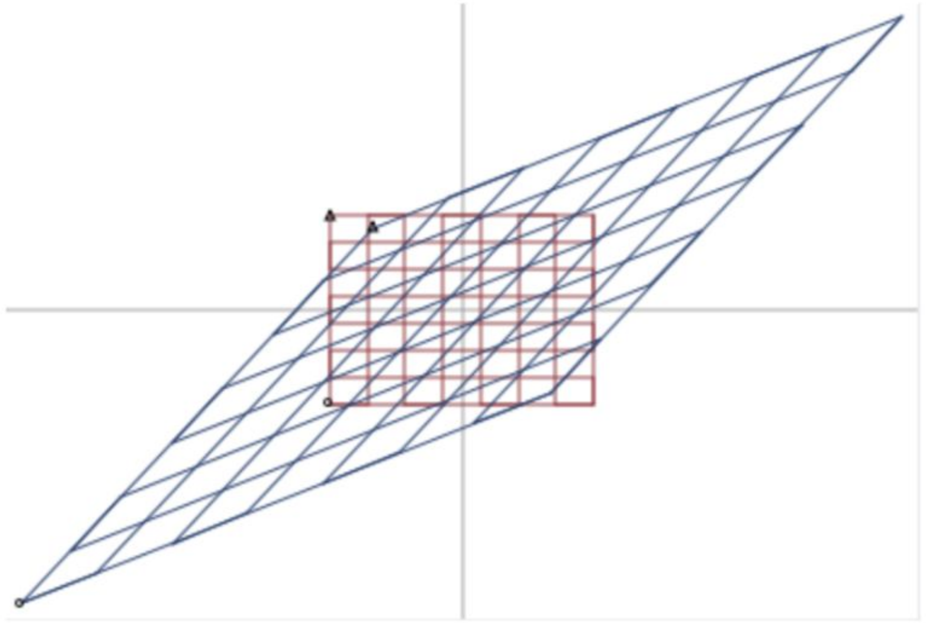
Matrix as a transformation on a vector



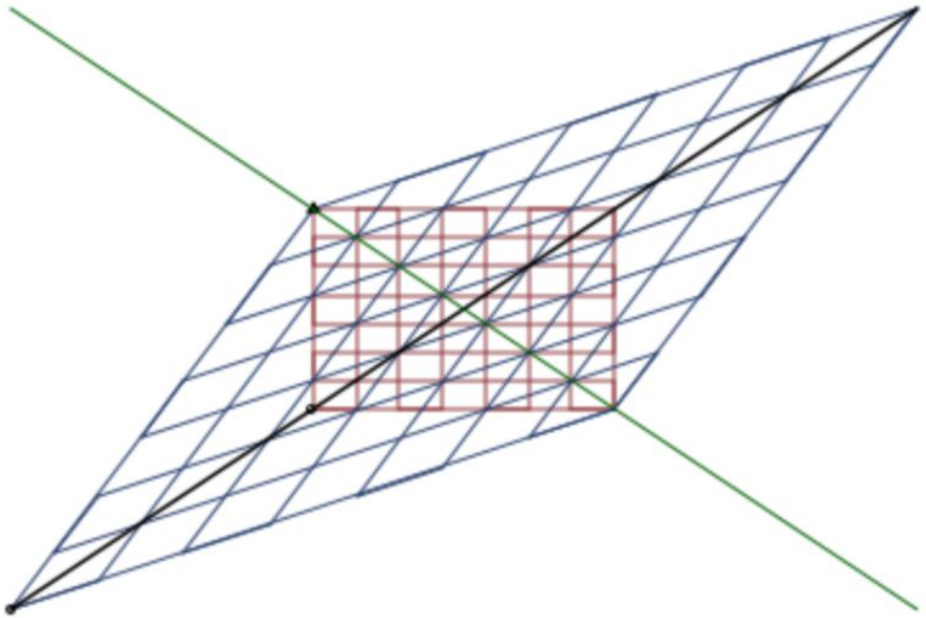
$$\begin{bmatrix} 1.36 & -2.35 \\ -2.35 & 4.18 \end{bmatrix} \begin{bmatrix} -0.49 \\ 0.87 \end{bmatrix} = \begin{bmatrix} -2.715 \\ 4.795 \end{bmatrix} = 5.51 \begin{bmatrix} -0.49 \\ 0.87 \end{bmatrix}$$

For a given matrix there are special directions, along which its effect is only to stretch (without rotation). These special directions are called Eigendirection or Eigenvectors

A transformation matrix that stretches, rotates and skews



Eigen Vectors



Eigen Vector Mathematics

The eigenvectors and eigenvalues of matrix A are defined to be the nonzero x and λ values that solve

$Ax = \lambda x$ (A is just stretching)

For a n-dim square matrix, there are atmost n eigen-vectors and n eigen-values.

Eigen Vector & PCA

Eigenvectors are the Principal component directions

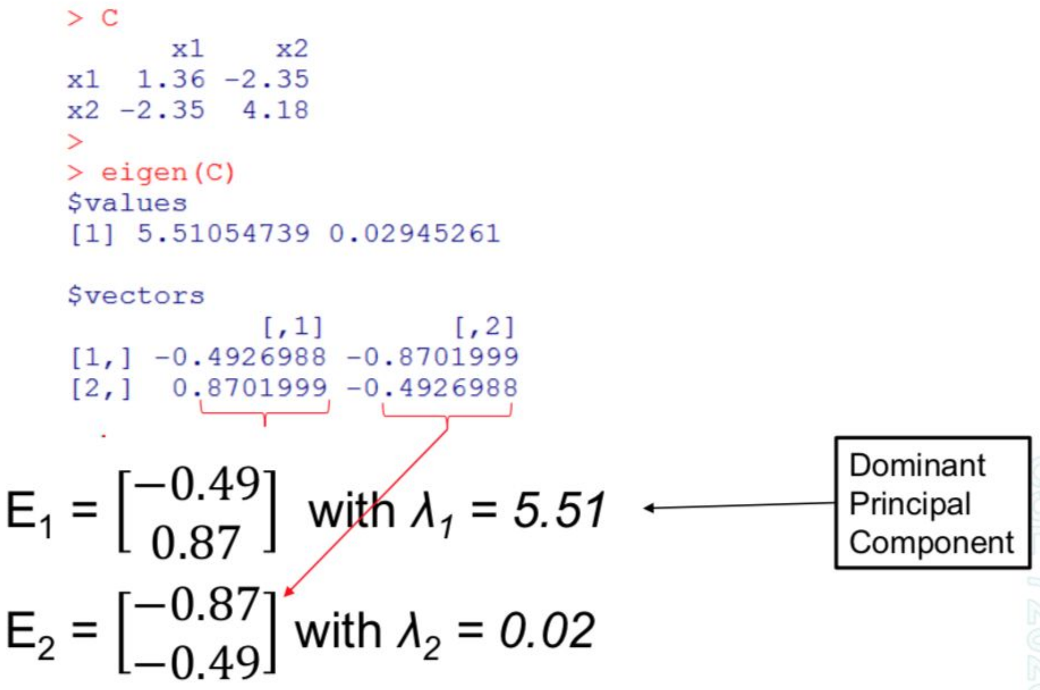
Eigenvalues are the magnitude of stretch

Eigenvalues represent the magnitude of variance along those directions

Eigen Vector & PCA

The relative size of eigenvalues says variance in 1 direction is much higher than in the other.

<u>In Class exercise: Please rework in Python (15 min)</u>



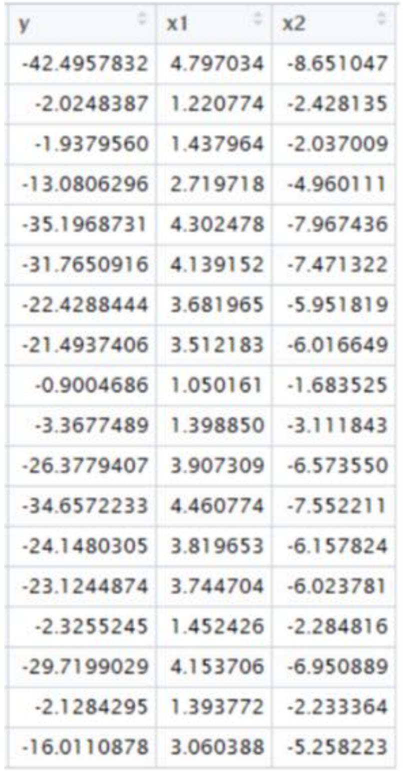
```
> C
        x1    x2
x1   1.36 -2.35
x2  -2.35  4.18
>
> eigen(C)
$values
[1] 5.51054739 0.02945261

$vectors
           [,1]       [,2]
[1,] -0.4926988 -0.8701999
[2,]  0.8701999 -0.4926988
```

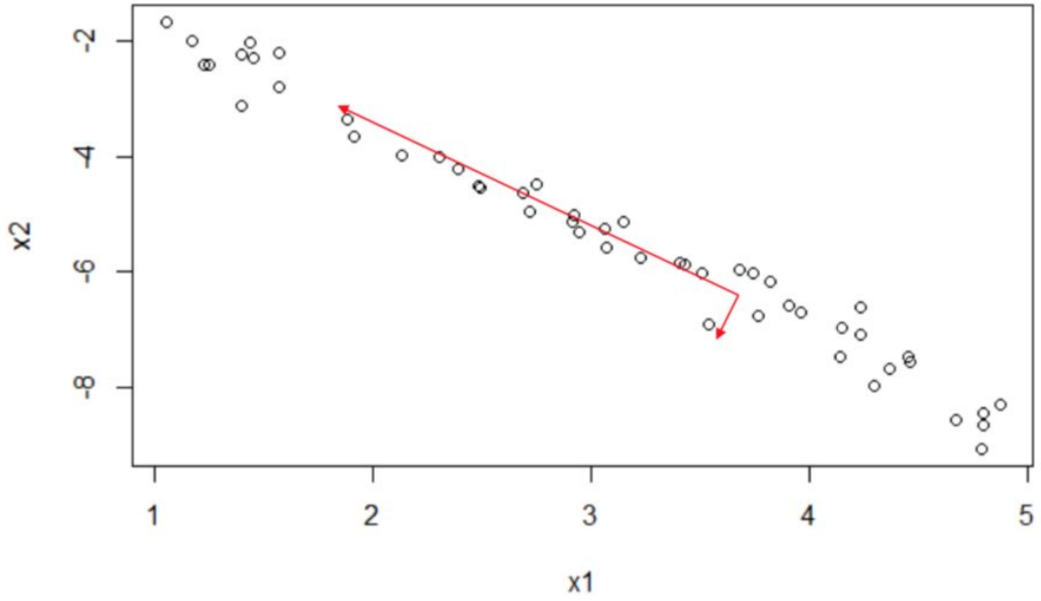
$$E_1 = \begin{bmatrix} -0.49 \\ 0.87 \end{bmatrix} \text{ with } \lambda_1 = 5.51 \leftarrow \boxed{\text{Dominant Principal Component}}$$

$$E_2 = \begin{bmatrix} -0.87 \\ -0.49 \end{bmatrix} \text{ with } \lambda_2 = 0.02$$

Dimension Reduction

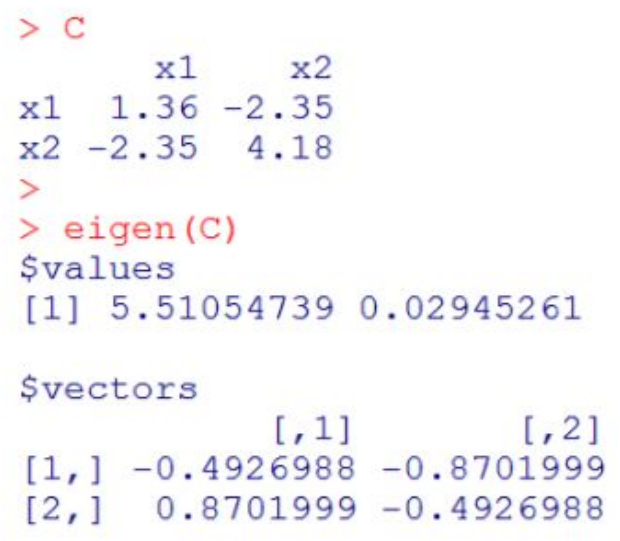


y	x1	x2
-42.4957832	4.797034	-8.651047
-2.0248387	1.220774	-2.428135
-1.9379560	1.437964	-2.037009
-13.0806296	2.719718	-4.960111
-35.1968731	4.302478	-7.967436
-31.7650916	4.139152	-7.471322
-22.4288444	3.681965	-5.951819
-21.4937406	3.512183	-6.016649
-0.9004686	1.050161	-1.683525
-3.3677489	1.398850	-3.111843
-26.3779407	3.907309	-6.573550
-34.6572233	4.460774	-7.552211
-24.1480305	3.819653	-6.157824
-23.1244874	3.744704	-6.023781
-2.3255245	1.452426	-2.284816
-29.7199029	4.153706	-6.950889
-2.1284295	1.393772	-2.233364
-16.0110878	3.060388	-5.258223



If we simplify and pick only one feature, it easier to do so, in the transformed variables. We can keep the dominant principal component and skip the non-dominant component.

Dimension Reduction



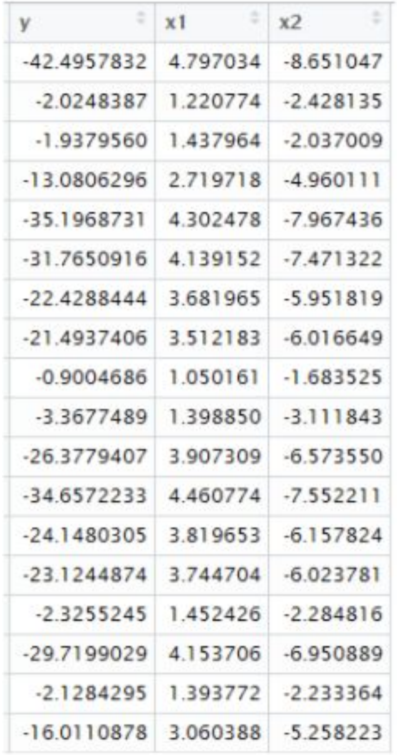
```
> C
       x1    x2
x1   1.36 -2.35
x2  -2.35  4.18
>
> eigen(C)
$values
[1] 5.51054739 0.02945261

$vectors
           [,1]       [,2]
[1,] -0.4926988 -0.8701999
[2,]  0.8701999 -0.4926988
```

Remember, in the other example, when we transformed the data points from original variable x1, x2 into new transformed variable, X1 & X2, we could reduce the dimensions?

This matrix of eigen-vectors as a whole also allows you to transform each one of our data-points into new variables X1 & X2

Transform into Principal Components

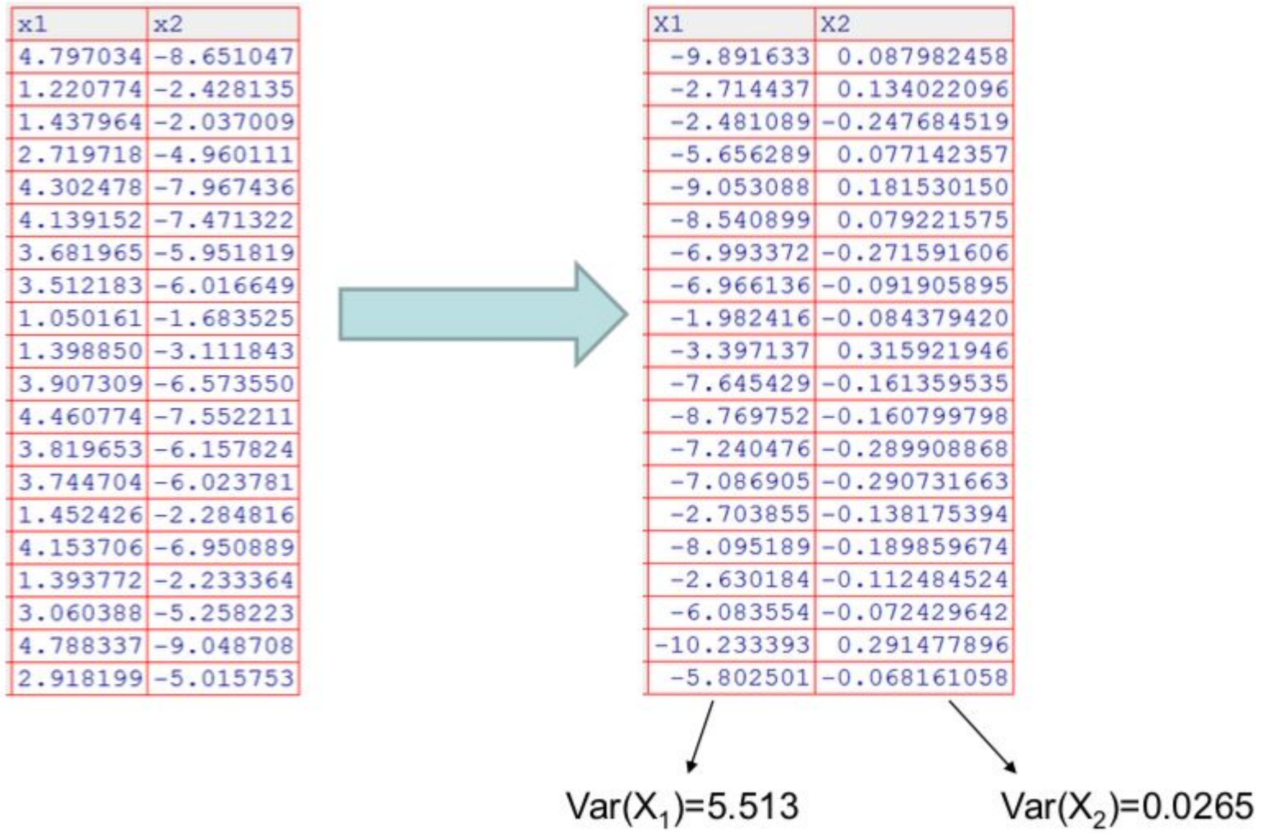


y	x1	x2
-42.4957832	4.797034	-8.651047
-2.0248387	1.220774	-2.428135
-1.9379560	1.437964	-2.037009
-13.0806296	2.719718	-4.960111
-35.1968731	4.302478	-7.967436
-31.7650916	4.139152	-7.471322
-22.4288444	3.681965	-5.951819
-21.4937406	3.512183	-6.016649
-0.9004686	1.050161	-1.683525
-3.3677489	1.398850	-3.111843
-26.3779407	3.907309	-6.573550
-34.6572233	4.460774	-7.552211
-24.1480305	3.819653	-6.157824
-23.1244874	3.744704	-6.023781
-2.3255245	1.452426	-2.284816
-29.7199029	4.153706	-6.950889
-2.1284295	1.393772	-2.233364
-16.0110878	3.060388	-5.258223

Any record in our data set (x1,x2) when multiplied by the matrix of eigenvectors, we get the new coordinates in the rotated principal component axis.

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.49 & -0.87 \\ 0.87 & -0.49 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.49\, x_1 - 0.87 x_2 \\ 0.87 x_1 + 0.49 x_2 \end{bmatrix}$$



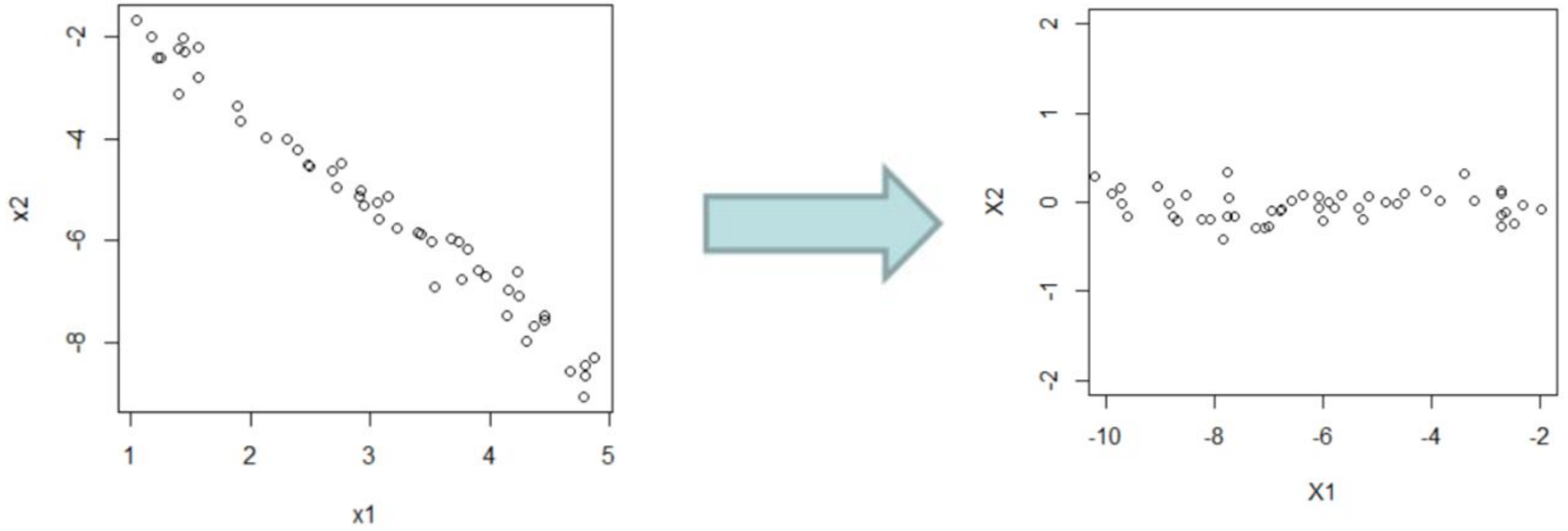
x1	x2
4.797034	-8.651047
1.220774	-2.428135
1.437964	-2.037009
2.719718	-4.960111
4.302478	-7.967436
4.139152	-7.471322
3.681965	-5.951819
3.512183	-6.016649
1.050161	-1.683525
1.398850	-3.111843
3.907309	-6.573550
4.460774	-7.552211
3.819653	-6.157824
3.744704	-6.023781
1.452426	-2.284816
4.153706	-6.950889
1.393772	-2.233364
3.060388	-5.258223
4.788337	-9.048708
2.918199	-5.015753

X1	X2
-9.891633	0.087982458
-2.714437	0.134022096
-2.481089	-0.247684519
-5.656289	0.077142357
-9.053088	0.181530150
-8.540899	0.079221575
-6.993372	-0.271591606
-6.966136	-0.091905895
-1.982416	-0.084379420
-3.397137	0.315921946
-7.645429	-0.161359535
-8.769752	-0.160799798
-7.240476	-0.289908868
-7.086905	-0.290731663
-2.703855	-0.138175394
-8.095189	-0.189859674
-2.630184	-0.112484524
-6.083554	-0.072429642
-10.233393	0.291477896
-5.802501	-0.068161058

$\mathrm{Var}(X_1)=5.513$ $\mathrm{Var}(X_2)=0.0265$

Remember: The eigenvalues were exactly the same values!

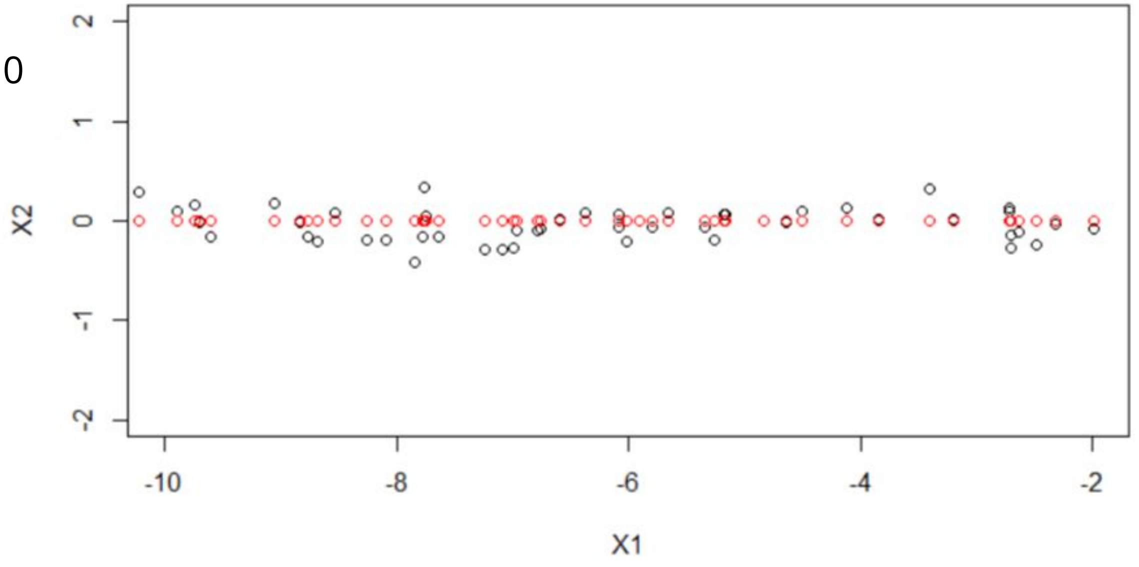
Data transformed into the basis of Principal components



Since X2 variable has small variance, we can now drop it by setting it to zero

Dimensionality Reduction

x2 has been set to 0



Red dots are the approximated dots

So now instead of doing regression of y vs (x1,x2), we are going to do regression of y vs X1. This is the point of doing PCA! It allows us to ignore variables with low variance.

Lets now apply the idea to an example with more features

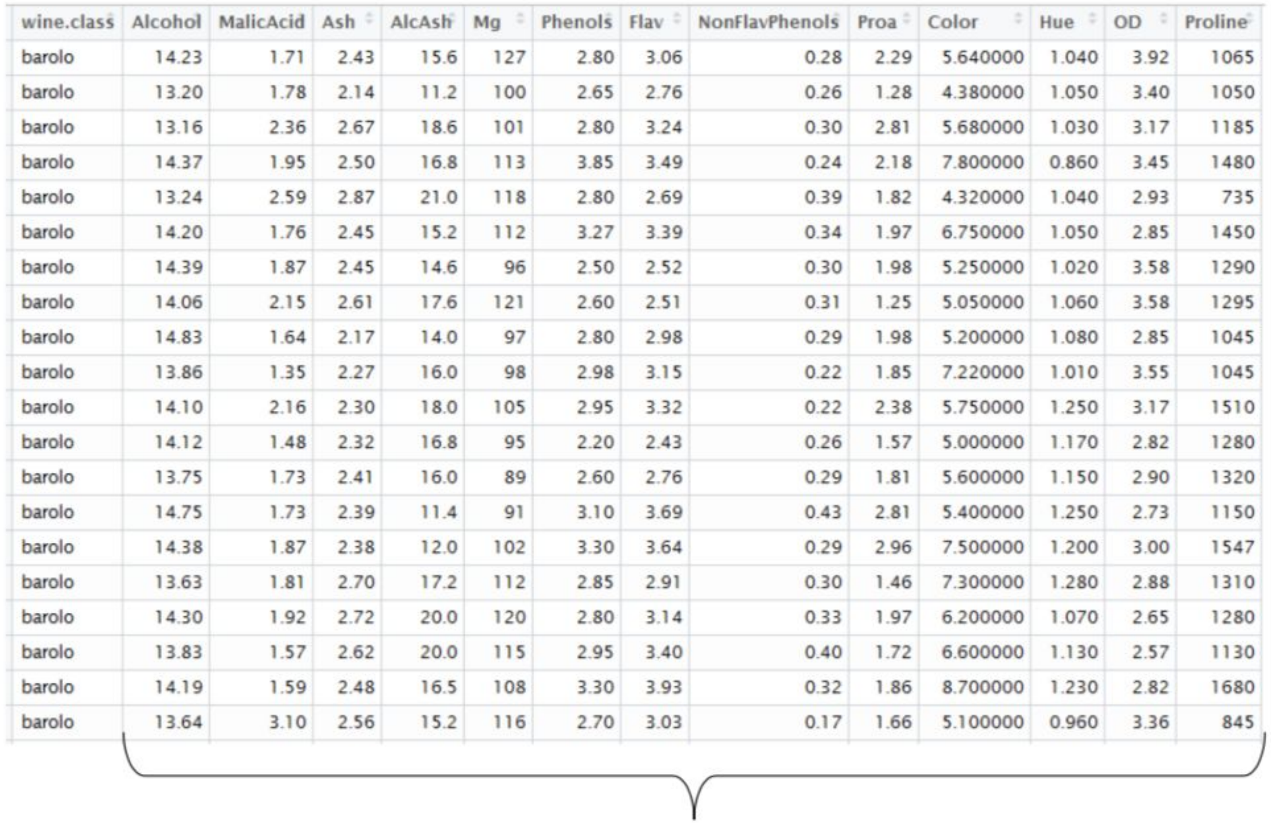
Chemical Analysis of Wine

We have chemical analysis of 178 different wines produced by vineyards from the same region of Italy.

The wines come from 3 different grapes – Barolo, Grignolino, Barbera.

We are trying to identify the type of grape from 13 features from chemical analysis

Dataset



wine.class	Alcohol	MalicAcid	Ash	AlcAsh	Mg	Phenols	Flav	NonFlavPhenols	Proa	Color	Hue	OD	Proline
barolo	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.640000	1.040	3.92	1065
barolo	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.380000	1.050	3.40	1050
barolo	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.680000	1.030	3.17	1185
barolo	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.800000	0.860	3.45	1480
barolo	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.320000	1.040	2.93	735
barolo	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.750000	1.050	2.85	1450
barolo	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.250000	1.020	3.58	1290
barolo	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.050000	1.060	3.58	1295
barolo	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	5.200000	1.080	2.85	1045
barolo	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	1.85	7.220000	1.010	3.55	1045
barolo	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22	2.38	5.750000	1.250	3.17	1510
barolo	14.12	1.48	2.32	16.8	95	2.20	2.43	0.26	1.57	5.000000	1.170	2.82	1280
barolo	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29	1.81	5.600000	1.150	2.90	1320
barolo	14.75	1.73	2.39	11.4	91	3.10	3.69	0.43	2.81	5.400000	1.250	2.73	1150
barolo	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29	2.96	7.500000	1.200	3.00	1547
barolo	13.63	1.81	2.70	17.2	112	2.85	2.91	0.30	1.46	7.300000	1.280	2.88	1310
barolo	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33	1.97	6.200000	1.070	2.65	1280
barolo	13.83	1.57	2.62	20.0	115	2.95	3.40	0.40	1.72	6.600000	1.130	2.57	1130
barolo	14.19	1.59	2.48	16.5	108	3.30	3.93	0.32	1.86	8.700000	1.230	2.82	1680
barolo	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17	1.66	5.100000	0.960	3.36	845

Dimension: 178 X 14.

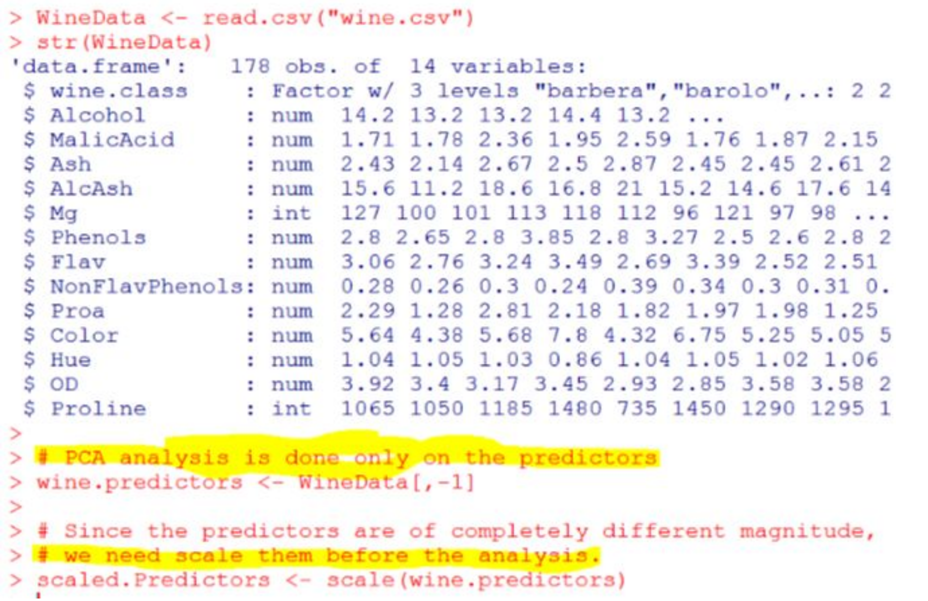
Number of Features = 13

PCA Steps

PCA is done only on predictor *numerical* variables

Convert categories to dummy numerical variables

If there are values of different order of magnitude, scale them

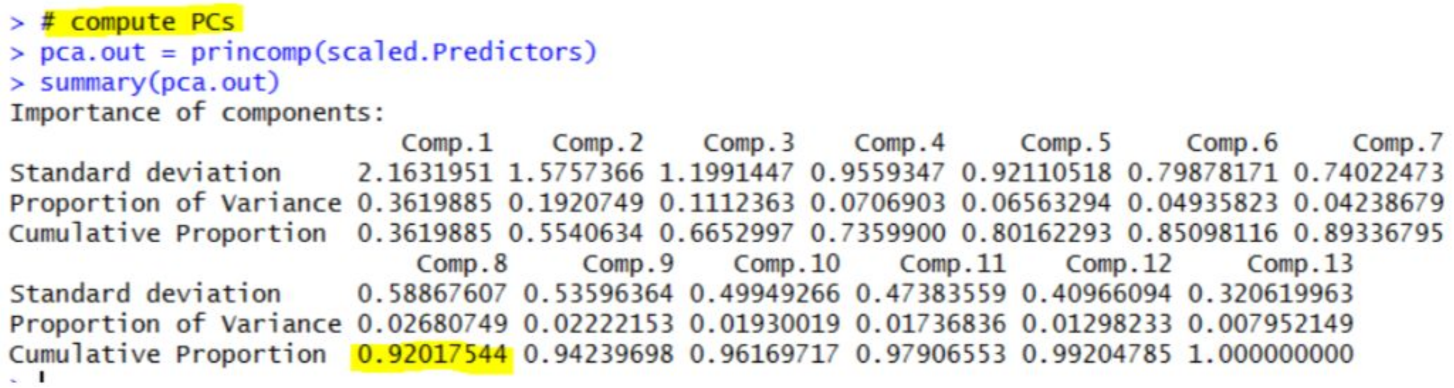


```
> WineData <- read.csv("wine.csv")
> str(WineData)
'data.frame':   178 obs. of  14 variables:
 $ wine.class    : Factor w/ 3 levels "barbera","barolo",..: 2 2
 $ Alcohol       : num  14.2 13.2 13.2 14.4 13.2 ...
 $ MalicAcid     : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15
 $ Ash           : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2
 $ AlcAsh        : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14
 $ Mg            : int  127 100 101 113 118 112 96 121 97 98 ...
 $ Phenols       : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2
 $ Flav          : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51
 $ NonFlavPhenols: num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.
 $ Proa          : num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25
 $ Color         : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5
 $ Hue           : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06
 $ OD            : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2
 $ Proline       : int  1065 1050 1185 1480 735 1450 1290 1295 1
>
> # PCA analysis is done only on the predictors
> wine.predictors <- WineData[,-1]
>
> # Since the predictors are of completely different magnitude,
> # we need scale them before the analysis.
> scaled.Predictors <- scale(wine.predictors)
```

Principal Components in R

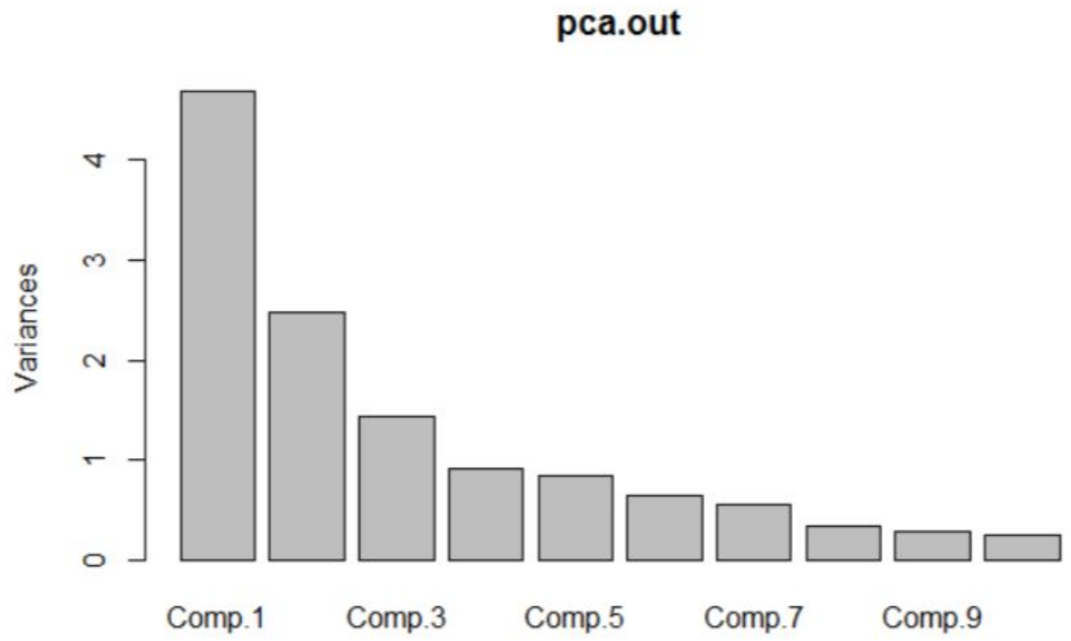
princomp performs a principal components analysis on a given numeric data matrix (In class exercise - find out the equivalent 3 Python commands ?) (5 min)

It outputs are:
– loadings: The coefficients for linear transformations into the new variables
– scores: The input features into the transformed bases
 – It also gives a summary of percentage of variance explained by each Principal Component



```
> # compute PCs
> pca.out = princomp(scaled.Predictors)
> summary(pca.out)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5     Comp.6     Comp.7
Standard deviation     2.1631951 1.5757366 1.1991447 0.9559347 0.92110518 0.79878171 0.74022473
Proportion of Variance 0.3619885 0.1920749 0.1112363 0.0706903 0.06563294 0.04935823 0.04238679
Cumulative Proportion  0.3619885 0.5540634 0.6652997 0.7359900 0.80162293 0.85098116 0.89336795
                           Comp.8     Comp.9    Comp.10    Comp.11    Comp.12     Comp.13
Standard deviation     0.58867607 0.53596364 0.49949266 0.47383559 0.40966094 0.320619963
Proportion of Variance 0.02680749 0.02222153 0.01930019 0.01736836 0.01298233 0.007952149
Cumulative Proportion  0.92017544 0.94239698 0.96169717 0.97906553 0.99204785 1.000000000
```

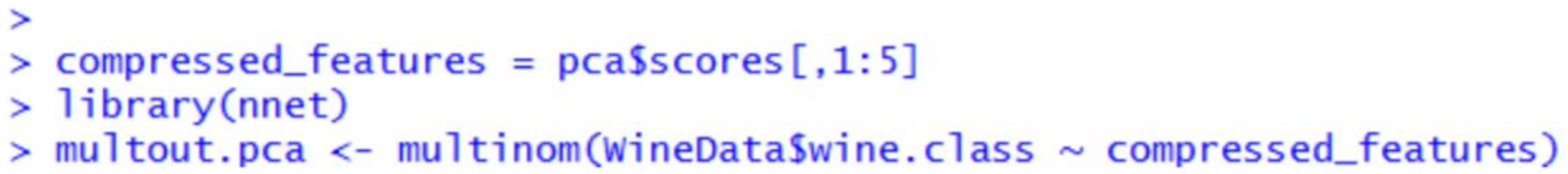
- There are 13 principal components (since there were 13 initial features)
- The output principal components is ordered according to the percentage of variance explained.
- The top row gives the eigenvalues, the 2nd row gives the percentage of explained variance.
- First 8 components explain over 90% of the variance!



```
> plot(pca.out)
```

Dimensionality Reduction

Let us say we set a threshold of 80% variance explanatory power
Keeping the first 5 components of PC, explains 80% of variance
So we can choose to ignore the remaining 8 components and build our model
component1: q1b1 + q2b2 + q3b3 + q4b4
0.078 b1 + 0.062 b2 − 0.182 b3 + 0.179 b4



```
>
> compressed_features = pca$scores[,1:5]
> library(nnet)
> multout.pca <- multinom(WineData$wine.class ~ compressed_features)
```

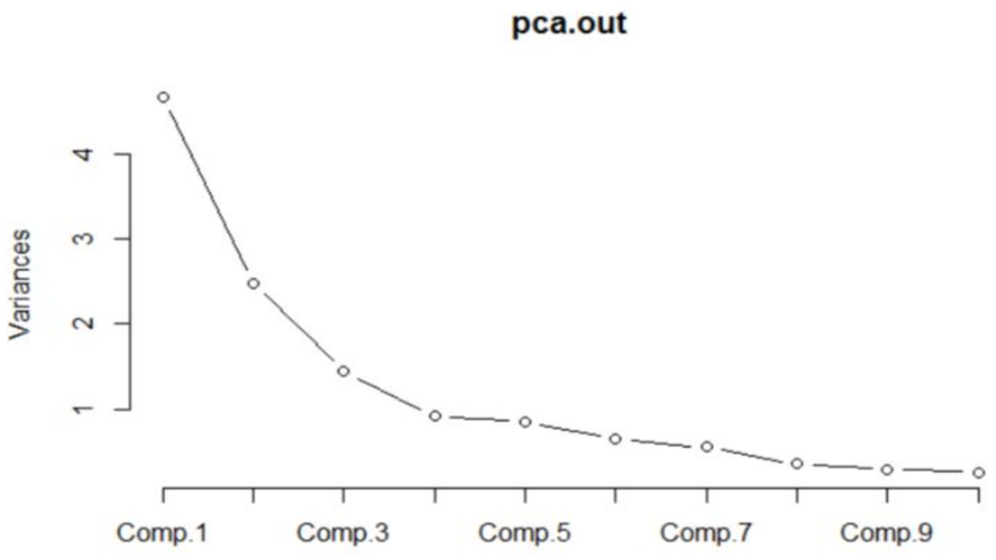
How many components to choose?

Rule of thumb:

Pick the point, which is at the "elbow" in the plot.

Over here, it happens at around 4th or 5th component.

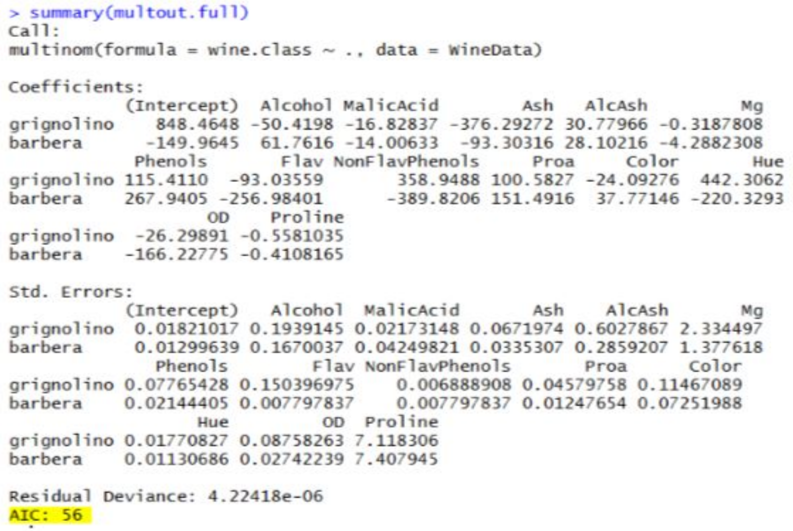
To know the components: pca.components_



```
>
> screeplot(pca.out, type="lines")
```

PCA performance: Multinomial Classification

Regular



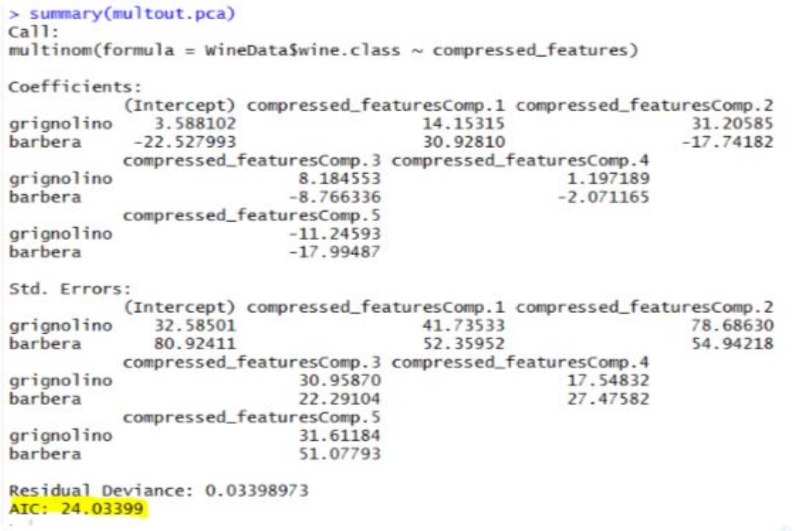
```
> summary(multout.full)
Call:
multinom(formula = wine.class ~ ., data = WineData)

Coefficients:
           (Intercept)  Alcohol MalicAcid        Ash   AlcAsh         Mg
grignolino    848.4648 -50.4198 -16.82837 -376.29272 30.77966 -0.3187808
barbera      -149.9645  61.7616 -14.00633  -93.30316 28.10216 -4.2882308
            Phenols       Flav NonFlavPhenols     Proa     Color      Hue
grignolino 115.4110  -93.03559       358.9488 100.5827 -24.09276 442.3062
barbera    267.9405 -256.98401      -389.8206 151.4916  37.77146 -220.3293
                  OD    Proline
grignolino  -26.29891 -0.5581035
barbera    -166.22775 -0.4108165

Std. Errors:
           (Intercept)   Alcohol  MalicAcid       Ash    AlcAsh       Mg
grignolino  0.01821017 0.1939145 0.02173148 0.0671974 0.6027867 2.334497
barbera     0.01299639 0.1670037 0.04249821 0.0335307 0.2859207 1.377618
              Phenols        Flav NonFlavPhenols       Proa      Color
grignolino 0.07765428 0.150396975    0.006888908 0.04579758 0.11467089
barbera    0.02144405 0.007797837    0.007797837 0.01247654 0.07251988
                  Hue         OD  Proline
grignolino 0.01770827 0.08758263 7.118306
barbera    0.01130686 0.02742239 7.407945

Residual Deviance: 4.22418e-06
AIC: 56
```

With PCA



```
> summary(multout.pca)
Call:
multinom(formula = WineData$wine.class ~ compressed_features)

Coefficients:
           (Intercept) compressed_featuresComp.1 compressed_featuresComp.2
grignolino    3.588102                  14.15315                  31.20585
barbera     -22.527993                  30.92810                 -17.74182
           compressed_featuresComp.3 compressed_featuresComp.4
grignolino                  8.184553                  1.197189
barbera                    -8.766336                 -2.071165
           compressed_featuresComp.5
grignolino                 -11.24593
barbera                    -17.99487

Std. Errors:
           (Intercept) compressed_featuresComp.1 compressed_featuresComp.2
grignolino    32.58501                  41.73533                  78.68630
barbera       80.92411                  52.35952                  54.94218
           compressed_featuresComp.3 compressed_featuresComp.4
grignolino                  30.95870                  17.54832
barbera                     22.29104                  27.47582
           compressed_featuresComp.5
grignolino                  31.61184
barbera                     51.07793

Residual Deviance: 0.03398973
AIC: 24.03399
```

PCA does not always guarantee better performance. The main use is dimensionality reduction

PCA: Visualization

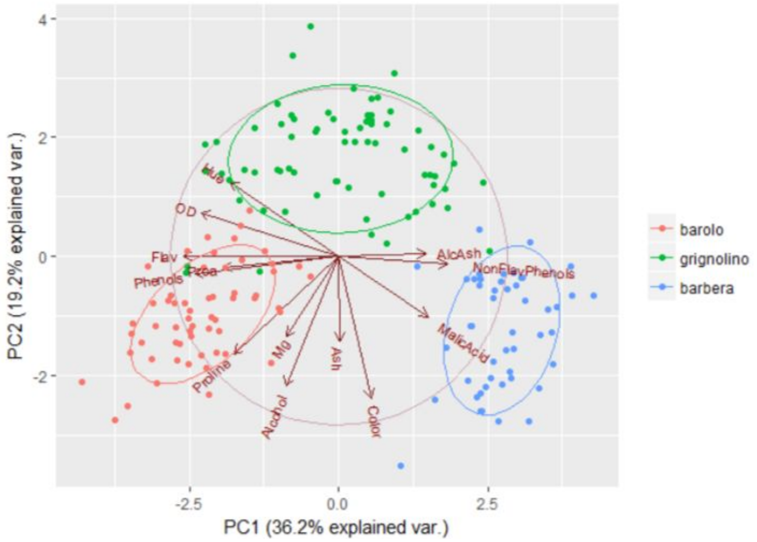
Many times visualizing the spread in the data-points using only the first 2 Principal Components can provide useful information about the spread of the data

The function biplot lets you do that

Also check ggbiplot in the library(ggbiplot)

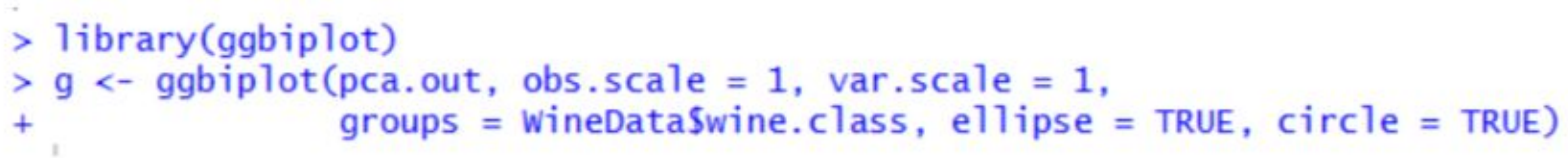
PCA is also referred to LSI, SVD, LDA, Eigen Decomposition, Factor Analysis, etc (domain dependent)

Projection using first 2 components



Each class, nicely gets segmented when viewed from the Principal Component basis

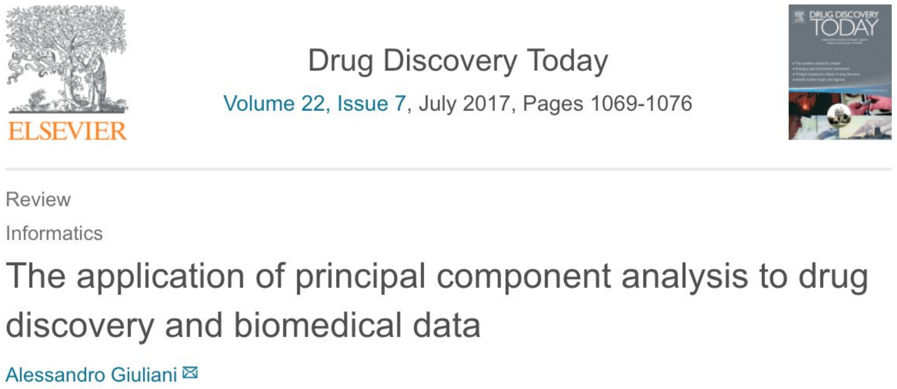
biplot is also occasionally used to identify outliers



```
> library(ggbiplot)
> g <- ggbiplot(pca.out, obs.scale = 1, var.scale = 1,
+               groups = WineData$wine.class, ellipse = TRUE, circle = TRUE)
```

PCA Applications

1. Useful in pre-processing steps
2. Data mining, bioinformatics
3. Image compression


Drug Discovery Today

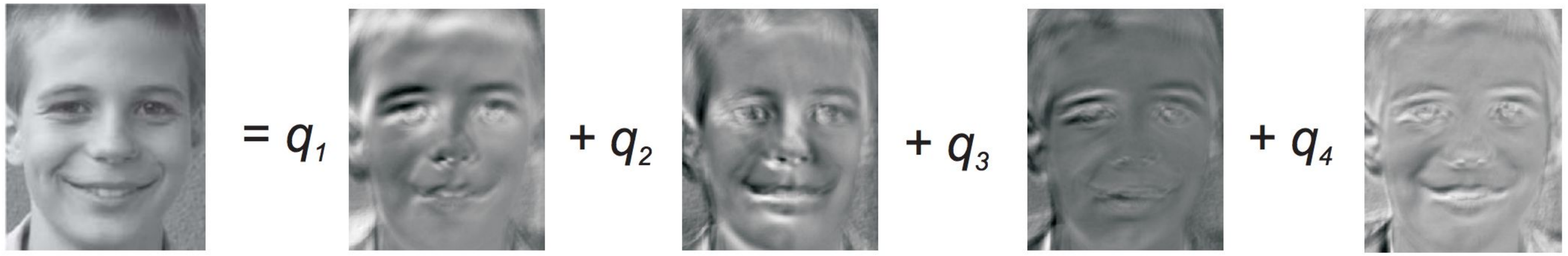
Volume 22, Issue 7, July 2017, Pages 1069-1076

Review

Informatics

The application of principal component analysis to drug discovery and biomedical data

Alessandro Giuliani



Thank you! Questions?