# Unsupervised Machine Learning:

## Big Data:

1) too large n (long data)

- ) Computers are happy and easy to deal with.

But:

- ) Large number of observations makes it easy to capture the slightest correlations. (very low P-value because n is high)
- ) So sometimes some variables are statistically significant without being substatively significant.

2) Large number of variables (wide data)
(P is Large)

- ) Modeling is more complex (Computationally and Conceptually)

3) n and P are not large
but the structure of data is very complex.

- ) Long Computational process.

## Unsupervised Learning:

we take a large set of variables and observations and try to detect a smaller set of underlying group. (Factors & clusters) (we do not have y, all variables are $x_1, x_2, \dots x_p$)

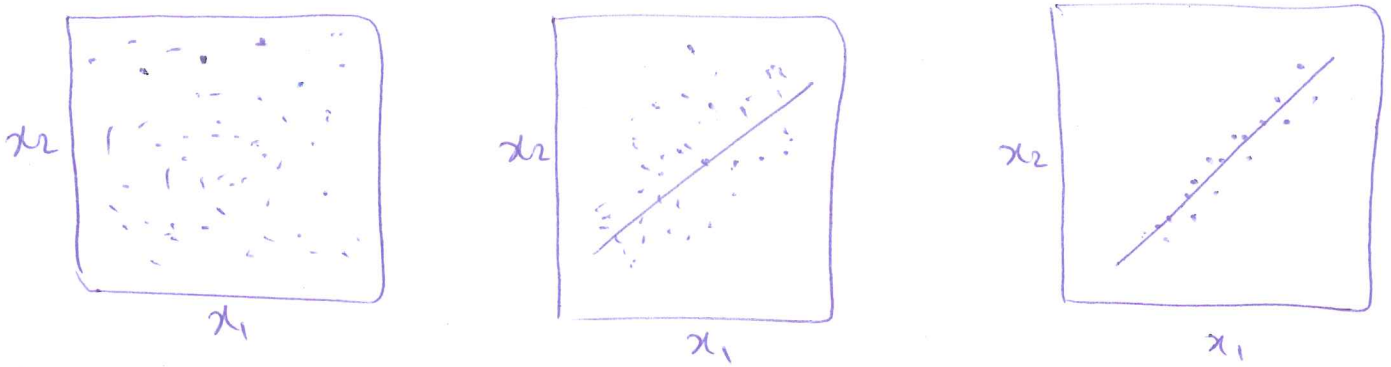
## Supervised Learning:

Any technique with the goal to use a large number of variables and observations to model and predict Y.

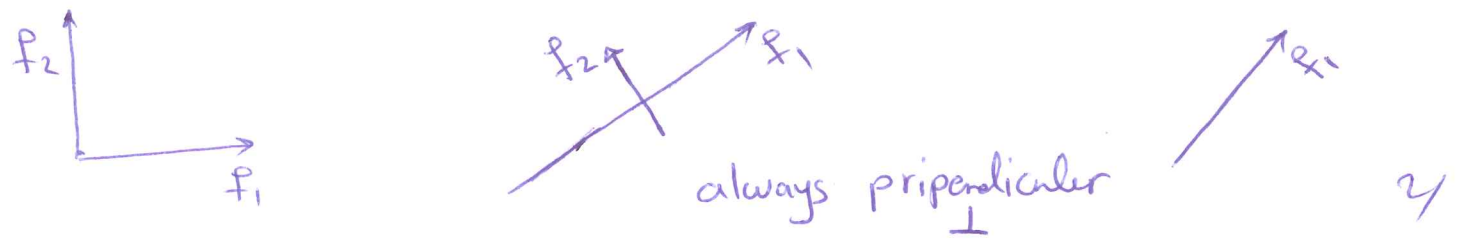
---

## Factor Analysis:

is the search for those underlying dimensions, or factors underneath a large set of variables.




Factors:



Factors equation: $p^{th}$ Principal Component = $C_p$

$$C_p = \phi_{P1} X_1 + \phi_{P2} X_2 + \phi_{P3} X_3 + \cdots \phi_{PP} X_P \quad (P\text{-dimension})$$

$X_1$: a vector containing $n$ value for variable $X_1$

* There are always as many components as there are variables ($P$).
* $\{\phi_{P1}, \phi_{P2}, \cdots\}$ is the projection of the original variables onto $C_p$. So the larger $\phi_{12}$ is, for instance, the larger the projection of $X_2$ on $C_1$. So the larger $\phi_{12}$ is the more similar $X_2$ & $C_1$ are.
* Before any analysis, data in all variables must be scaled using normal scaling, or linear, ... — (mean=0, sd=1)
* Two study is being done but they tend to produce very similar results:
  a) Factor Analysis (FA)
  b) Principal Component Analysis (PCA)

# Eigen Vectors and Eigen Values:

1. a PxP matrix has exactly p eigenvectors where each eigen vector is a vector of length p.

2. An eigen vector is defined as that vector such that, when multiplied by its associated matrix gives as a result the exact same vector, except stretched or shrunk by some value know as the eignvalue.

$$M v_p = \lambda_p v_p$$

$\lambda_p$: real number; $v_p$: vector Px1

3. Eigen vectors turns up all over mathematics & science.

4. Eigen vectors, like Principal Components, give you a reduced version of the data that contains much of the original data.

5. For temporal Process, they often describe where, temporal structures with lots of feedback end up in complex equilibrium over time.

(6) Every Matrix $M_{n \times p}$, has P eigenvectors and p eigen values associated with it:

$\{v_1, \lambda_1\}, \{v_2, \lambda_2\}, \{v_3, \lambda_3\}, \ldots$ (ordered by size of $\lambda_i$)

(7) The eigen vector with the Largest $\lambda_p$ is the first eigen vector (Thus the first principal Component) and so on for all the rest.

(8) For our data Analysis: Matrix M is:

M = Covariance matrix of all variables.

Procedure for Factor Analysis:

1. Standardize the variables: (ex. mean=0, Sd=1)
2. Create the Covariance Matrix "M"
3. Find the eigenvectors & eigen values "M"
4. Choose How many of them we want to keep and analyze.

example) * get data from "psych" packge

* It also has some machine learnig function

```
Library(psych)                    Like: fa ⟶ factor analysis.
data(msq)
msq7 <- msq[,2:72]  ⟶  all indep. variable  x1, x2, ... x72
msq7[msq7=="9"] = NA
msq7 <- data.frame(msq7)
msq2 <- na.omit(msq7)
names(msq2)   # "afraid", "alert", "angry", "anxious"
dim(msq2)  # 1747 71        ... "unhappy", "upset", "guilty", "happy", ...
fact <- fa(msq2, nfactors=2)
fact1 <- fact$loadings[,1]   #extract the projection of the variables onto the factors.
fact1 <- [order(fact1)]
```

$\phi_{11}, \phi_{12}, \phi_{13}$ ...
$\phi_{21}, \phi_{22}, \phi_{23}$, ...

```
# So the highest φ_ij's
  are "energetic", "lively" (+)
  and "sluggish", "tiered" (-)
fact2 <- fact$Loadings[,2]
fact2 <- [order(fact2)]   # "relaxed" "Calm" (+)
                          "distressed" "Frustrated" (+)
```

② Other Methods: { prcomp / PrinComp — buil-in R

```
pcaA <- prcomp(msq2)
pcaA1 <- pcaA$rotation[,1]          } method 1

pcaB <- princomp(msq2)
pcaB1 <- pcaB$loadings[,1]          } method 2
```

③ Manual:

```
covm <- cov(msq2)
eigenm <- eigen(covm)
eigen1 <- eigenm$vectors[,1]
eigen_value1 <- eigenm$values[,1]
```

* All these results: "{fact1, PcaA1, PcaB1, eigen1}

are very similar.

④ Estimation Boundary

```
tm.msq2 <- t(as.matrix(msq2))
m.msq2 <- as.matrix(msq2)
Va <- rnorm(ncol(msq2))
for i in 1:10{
 Vb <- m.msq2 %*% scale(Va)
 Va <- tm.msq2 %*% scale(Vb)
}
```

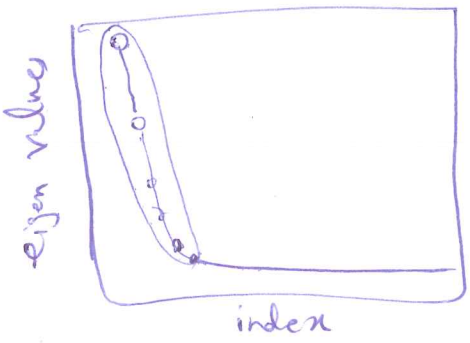
⇒ first eigvector estimated. ☺

# How many of $\underline{p}$ factors to keep?

① 

```
plot( eigenm$values, type="b")
```

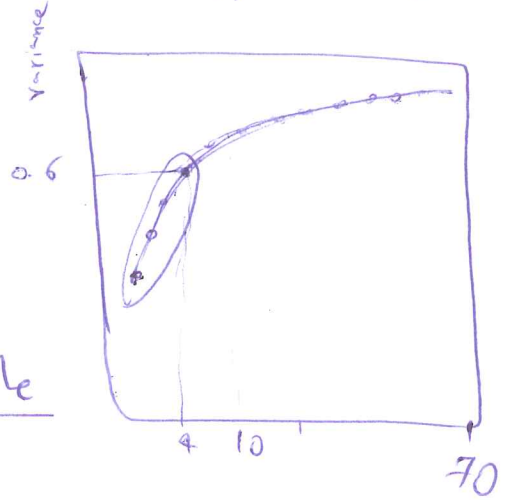
look for the "elbow" →




* The flat area is all the factors that are just noise. So we count from left until the elbow.

② by plotting the proportion of totall variance in data explained by the factors as you add them together.

```
plot(cumsum(eigenm$values)/sum(eigenm$values), ylim=
                                              =c(0,1))
```



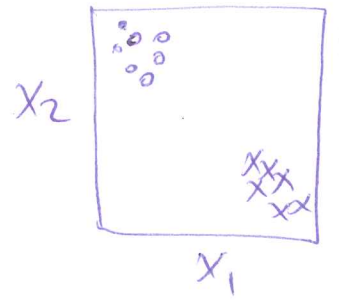

⇒ more than 50% of the total variance in data is explained by just those 4 variable

# Clustering:

① unlike factor analysis, cluster analysis is targeted as observation level rather than the variables.

② we want to know what natural group the observation fall into:

goal: ← which group $(x_1, x_2, x_3, \cdots x_p)$ a sample observation




Clustering Methods:
- ① k-mean
- ② Hierarchial

① ✗ k-means Algorithm:

① Knowing how many clusters we divide data into
② Assign a category at random to every data point.
3a for each category calculate the mean of all points in that group. (centroid of that group)
3b after calculating all centroids, reassign each point to the group of the nearest centroid. go back 3a until convergance.

Codes:

```
kout <- kmeans(msq2, center=2, nstart = 25)   # 25 times starts randomly
centroids <- kout$centers
topVars_centroid1 <- centroids[1, order(centroids[1,])]
topVars_centroid2 <- centroids[2, order(centroids[2,])]
```

The tutorial Code is in the file Computer.

manual

how many clusters?
start with low increase one by one, as total variance is lowering significantly.

(2) Hierarchial Clustering:

goal: → ① Finding how many k (clusters)

"Dissimilarity":

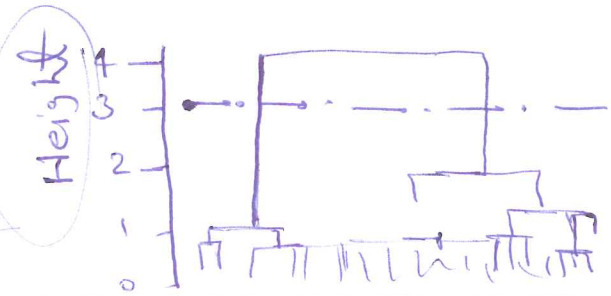
Complete: the dissimilatry between two cluster A and B is equal to the largest distance between a member of group A and a member of group B

Single: like complete, but sets the dissimilarity between A & B to the smallest distance between a member of A & a member of B

Average: The average dissimilarity between every member of A and every member of B

Centroid: The distance between the centroid of A and the centroid of B

```
hout <- hclust(dist(simdat1[,1:2]), method = "average")
plot(hout)
abline(a=3, b=0, col="red")
```




- every observation is a cluster centroid.
  Then we find dissimilarities using "average" method
  The two clusters centroids having smallest dissimilarity join together. This continues until they are all united.

- The "height" where any two clusters join is equal to the dissimilarity between those two clusters.

- draws a line, a = intercept (where it touches y-axis), b = inclination

```
as.vector(cutree(hout, 2))
  # This gives a vector, 2 clusters
      of clusters associated with each observation
                  # 1 1 1 2 1 1 1 2 2 2 1 2 1 ...
as.vector(cutree(hout, h=3))
                  height   # The same output
```