Statistical Computing with R: Masters in Data Sciences 503 (S26) Fourth Batch, SMS, TU, 2025

Shital Bhandary
Associate Professor
Statistics/Bio-statistics, Demography and Public Health Informatics
Patan Academy of Health Sciences, Lalitpur, Nepal
Faculty, Masters in Medical Research, NHRC/Kathmandu University
Faculty, FAIMER Fellowship in Health Professions Education, India/USA

Review Preview: Unsupervised models

- Dimension reduction
  - Principal component analysis
  - Principal axis factoring (exploratory factor analysis)
  - Multi-dimensional scaling
- Clustering
  - K-means clustering
  - Hierarchical clustering
- Association rules
- Monte Carlo simulations

Unsupervised learning:

- Dependent variable/labelled data is not available for this type of learning
- We need to work with the independent variables **ONLY**
- We can form a **composite score** using all the independent variables (dimension reduction)
- Dimension = variables = features
- We can group a similar cases with similar characteristics (clustering)
- In both cases, correlation and distance measures plays very important role so this is known and multivariate exploratory analysis in statistics

You need to read Chapter 12: Unsupervised Learning (page 503 of PDF file) of the book “An Introduction to Statistical Learning” shared to you in the Google Classroom few days back. It contains the theory + lab works!

IBM: Unsupervised learning

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets.

- **These algorithms discover hidden patterns or data groupings without the need for human intervention.**

- Its ability to discover similarities and differences in information makes it the ideal solution for **exploratory data analysis**, cross-selling strategies, customer segmentation, and image recognition.

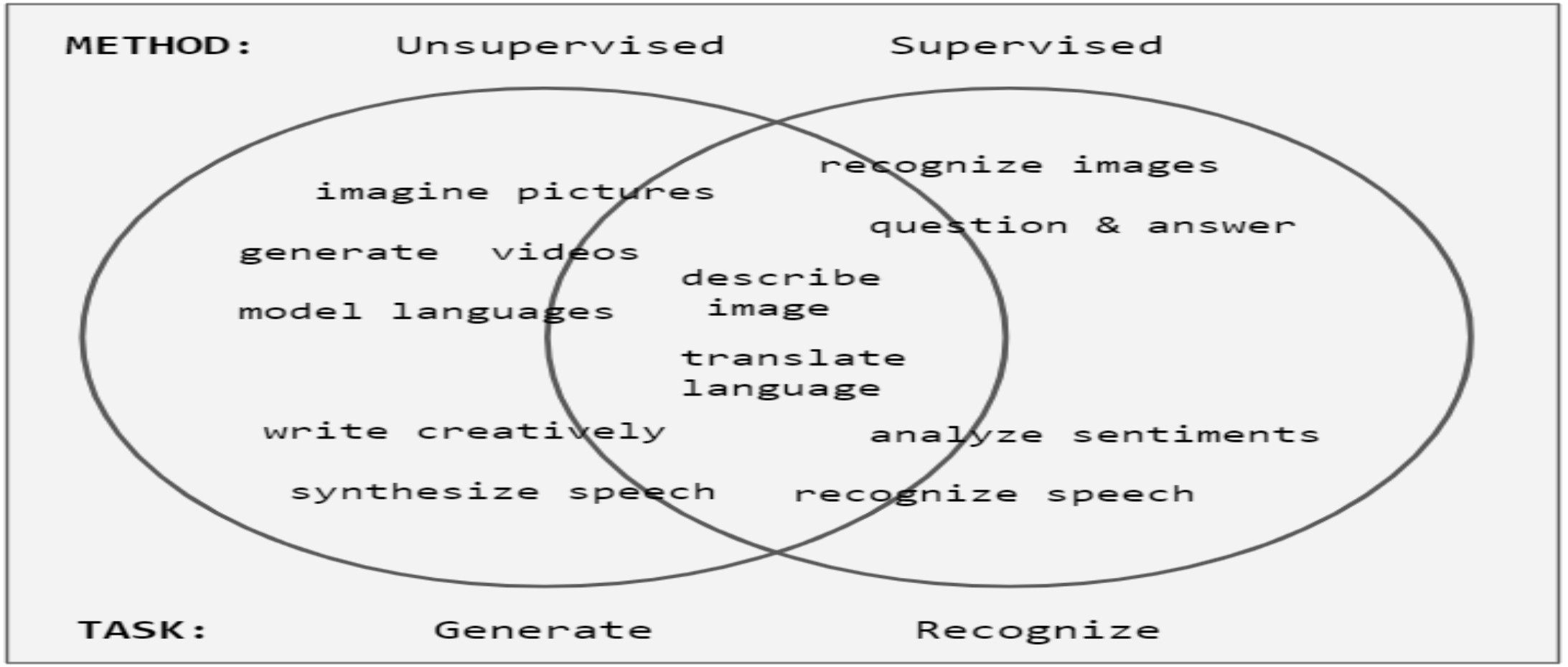
Alternatively: https://en.wikipedia.org/wiki/Unsupervised_learning

- **Unsupervised learning** is a type of machine learning in which the algorithm is not provided with any pre-assigned labels or scores for the training data.
- As a result, unsupervised learning algorithms must first self-discover any naturally occurring patterns in that training data set.
- Common examples include **clustering**, where the algorithm automatically groups its training examples into categories with similar features, and **principal component analysis**, where the algorithm finds ways to compress the training data set by identifying which features are most useful for discriminating between different training examples, and discarding the rest.

Application of unsupervised learning: https://www.ibm.com/cloud/learn/unsupervised-learning

- **Computer vision**
  - Unsupervised learning algorithms are used for visual perception tasks, such as object recognition.
- **Medical imaging**
  - Unsupervised machine learning provides essential features to medical imaging devices, such as image detection, classification and segmentation, used in radiology and pathology to diagnose patients quickly and accurately.
- **Anomaly detection**
  - Unsupervised learning models can comb through large amounts of data and discover atypical data points within a dataset. These anomalies can raise awareness around faulty equipment, human error, or breaches in security.
- **Customer personas**
  - Defining customer personas makes it easier to understand common traits and business clients' purchasing habits. Unsupervised learning allows businesses to build better buyer persona profiles, enabling organizations to align their product messaging more appropriately.
- **Recommended Engines**
  - Using past purchase behavior data, unsupervised learning can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.

Unsupervised vs Supervised learning and overlapping:



https://en.wikipedia.org/wiki/Unsupervised_learning

Dimension Reduction (using variables)

- Factor Analysis
  - **PCA**
  - PAF
  - CFA
- Multi-Dimensional Scaling
  - MDS

Unsupervised Learning-Dimension Reduction: https://www.ibm.com/cloud/learn/unsupervised-learning

- While more data generally yields more accurate results, it can also impact the performance of machine learning algorithms (e.g. overfitting) and it can also make it difficult to visualize datasets.
- **Dimensionality reduction is a technique used when the number of features, or dimensions, in a given dataset is too high.**
- It reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible.

- It is commonly used in the preprocessing data stage i.e. before fitting a model, and there are a few different dimensionality reduction methods that can be used, such as:
  - Principal Component Analysis
  - Singular Value Decomposition (**ISLR Chapter 12 has example of this method!**)
  - Autoencoder (neural networks)

- **PCA is used to classify households poverty level with "wealth quintiles" in surveys and find components of a questionnaire with large number of items based on their correlations**

Principal Component Analysis (PCA) for Dimension Reduction: Read ISLR Chapter 12!

- Dimensionality reduction is a technique used when the number of features, or variables, in a given dataset is too high. It reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible.

- **Principal component analysis (PCA) is a type of dimensionality reduction algorithm which is used to reduce redundancies and to compress datasets through feature extraction**. This method uses a linear transformation to create a new data representation, yielding a set of "principal components."

- The first principal component is the direction which maximizes the variance of the dataset. While the second principal component also finds the maximum variance in the data, **it is completely uncorrelated to the first principal component**, yielding a direction that is perpendicular, or orthogonal, to the first component.
- This process repeats based on the number of dimensions, where a next principal component is the direction orthogonal to the prior components with the most variance.
- **We can use Kaiser's rule or Scree plot or both to extract components after PCA is applied to the data**

Let's use PCA: The ISLR Chapter 12 explains PCA with USArrests data too!

#We will use the built-in USArrests data

- head(USArrests)

- We need to combine the murder, assault and rape variables and create a latent variable i.e. "criminality" score using these three variables

	Murder	Assault	UrbanPop	Rape
• Alabama	13.2	236	58	21.2
• Alaska	10.0	263	48	44.5
• Arizona	8.1	294	80	31.0
• Arkansas	8.8	190	50	19.5
• California	9.0	276	91	40.6
• Colorado	7.9	204	78	38.7

"An Introduction to Statistical Learning" book chapter 12 uses all these variables to fit the PCA so I strongly suggest you to read and compare the result with the one we will get now onwards.

Fit a PCA in the data without the 3rd feature:

#Getting **criminality scale/score** after removing 3rd column i.e. Urban Population and scaling them

- library(dplyr)
- USArrests.1 <- USArrests[,-3] %>% scale

#Fitting PCA in the new data:

- pca.1 <- prcomp(USArrests.1)
- summary(pca.1)

- Importance of components:
-

	PC1	PC2	PC3
SD:	**1.5358**	0.6768	0.42822
Prop. Var	0.7862	0.1527	0.06112
Cum. Prop.	0.7862	0.9389	1.00000

PC1 explained around 79% of variance whereas PC2 explained around 15% and PC3 explained around 6% of variance with the latent variable i.e. criminality score.

As per Kaiser's rule, PC with Eigenvalue >= 1 (SD-square) must be used/retained for the latent variable.

We can say that we need only one component to create the latent variable as it explains 79% variance and SD^2>=1.

#It can also be done directly:

```
pca.2 <- prcomp(USArrests[,-3], scale = TRUE)
summary(pca.2)
```

Alternative, we can use “psych” package:

#Alternative

- library(psych)
- fa.1 <- psych::principal(USArrests.1, nfactors = 3, rotate = "none")
- fa.1

- **Standardized loadings** (pattern matrix) based upon correlation matrix

	PC1	PC2	PC3	h2	u2	com
Murder	**0.89**	-0.36	0.26	1	-2.2e-16	1.5
Assault	**0.93**	-0.14	-0.33	1	-2.2e-16	1.3
Rape	**0.83**	0.55	0.09	1	4.4e-16	1.8

The correlation of murder, assault and rape is very high with PC1 so variance explained by PC1 is also high!

#Sum of square and variance explained:

SS loading = Eigenvalue = SD-square

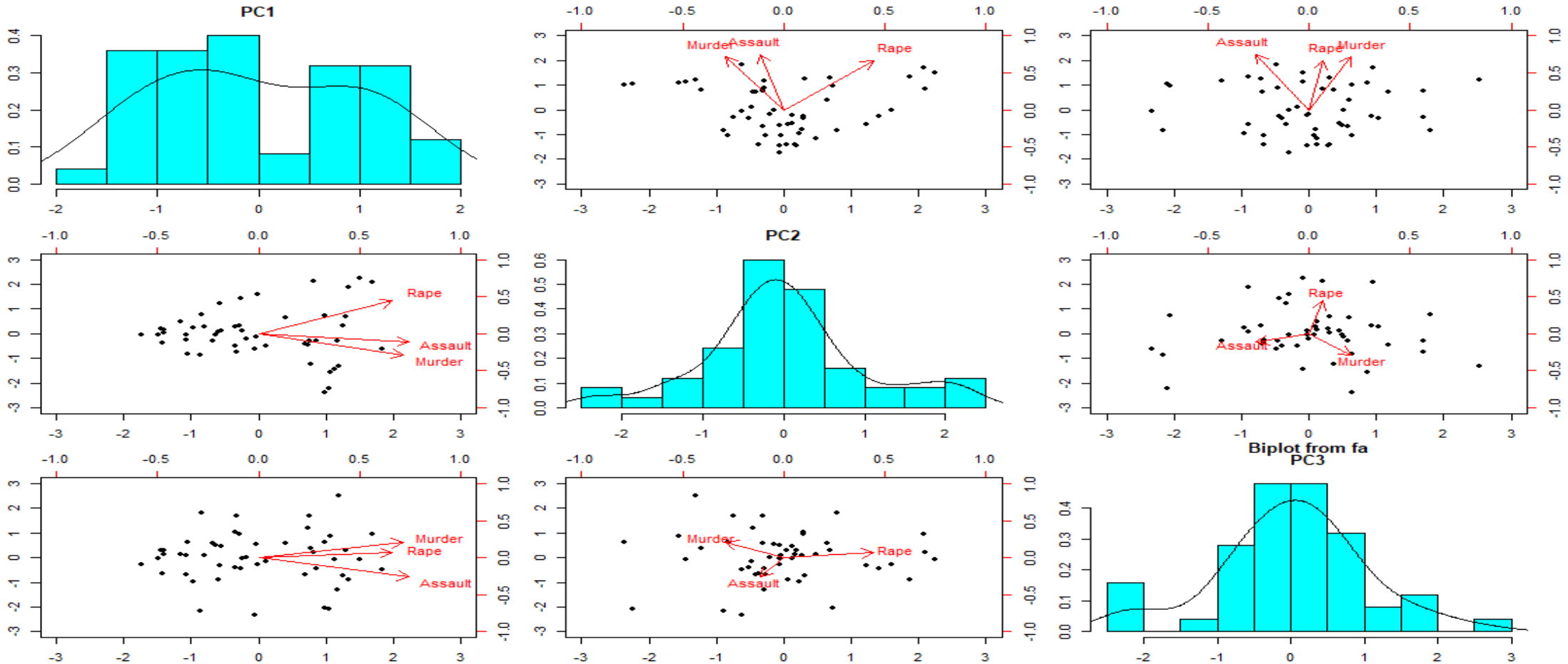
	PC1	PC2	PC3
SS loadings	2.36	0.46	0.18
Proportion Var	**0.79**	**0.15**	**0.06**
Cumulative Var	0.79	0.94	1.00

Prop. Explained	0.79	0.15	0.06
Cum. Proportion	0.79	0.94	1.00

Proportion variance explained by PC1 = 79%, PC2=15% and PC3 = 6%.
SS loadings = Eigenvalues (check with squared SD)

Bi-plot using “psych” package: Interpretation?

biplot(fa.1, labels = rownames(USArrests.1))



Getting eigen values from “FactoMineR”:

- library(factoextra)
- library(FactoMineR)

- str(USArrests)
- pca.data <- USArrests[,-3]
- head(pca.data)

- res.pca <- PCA(pca.data, graph=F)
- res.pca$eig

eigenvalue

- **comp 1 2.3585802**
- comp 2 0.4580513
- comp 3 0.1833685

percentage of variance

- comp 1 78.619340
- comp 2 15.268378
- comp 3 6.112282

Eigenvalue > 1 = Only 1 component

Principal Component Analysis in R Tutorial https://www.datacamp.com/tutorial/pca-analysis-r

- This tutorial show how the PCA are derived from the correlation matrix ~ covariance matrix
- The correlation matrix is used to get covariance matrix, which is then used to get eigenvectors and eigenvalues
- Eigenvalue is used to select the principal components

- Five steps for PCA:

1. Data Normalization
2. Covariance matrix
3. Eigenvectors and eigenvalues
4. Selection of principal components with criteria
5. Data transformation in new dimensional space

Can we improve the PCA?
We can try PCA with “VARIMAX” rotation!

- #Rotated PCA with variance maximization
- fa.2 <- psych::principal(USArrests.1, nfactors = 3, rotate = "varimax")
- summary(fa.2)

	RC2	RC3	RC1
• Murder	0.26	**0.89**	0.37
• Assault	0.36	0.45	**0.82**
• Rape	**0.93**	0.24	0.27

	RC2	RC3	RC1
• SS loadings	1.06	1.05	0.88
• Proportion Var	**0.35**	**0.35**	**0.29**
• Cumulative Var	0.35	0.71	1.00
• Proportion Explained	0.35	0.35	0.29
• Cumulative Proportion	0.35	0.71	1.00

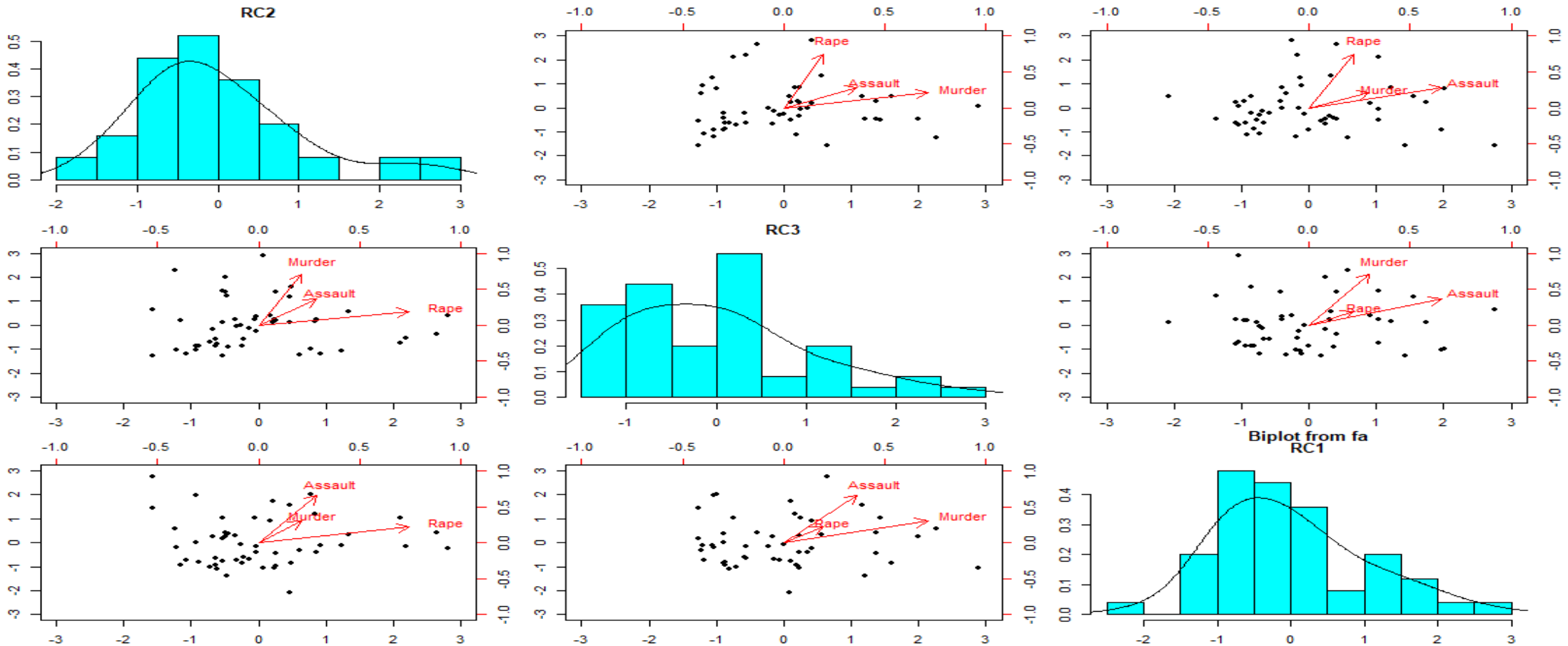
Did it improve the PCA?

What happened? **Is this a “data reduction”!**

Why?

Biplot: **Interpret these bi-plots now!**

biplot(fa.2, labels = rownames(USArrests.1))



Note:

- PCA must be used to produce "orthogonal" components
- PCA with "varimax" rotation also produced "orthogonal" components
- PCA with "varimax" rotation cannot be interpreted as a "true" PCA, why?
- If we need to get a **latent variable** with "correlated" components then we must use other **oblique** rotation methods and PCA not longer applies here
- Thus, we must use principal axis factoring (PFA) or factor analysis in such situations
- The common oblique rotation are:
  - Promax
  - Equimax etc.

I strongly suggest to use FactoMineR package to fit PCA and other factor analysis methods in R.
Methods: http://factominer.free.fr/factomethods/index.html
Example of PCA with this package here: http://www.sthda.com/english/wiki/wiki.php?id_contents=7851

PAC with varimax: Is it really a PCA? Discussion: https://stats.stackexchange.com/questions/612/is-pca-followed-by-a-rotation-such-as-varimax-still-pca

- Two main types of rotation are used: orthogonal when the new axes are also orthogonal to each other, and oblique when the new axes are not required to be orthogonal.

- Because the rotations are always performed in a subspace, the new axes will always explain less inertia than the original components (which are computed to be optimal).

- However, the part of the inertia explained by the total subspace after rotation is the same as it was before rotation (only the partition of the inertia has changed).
- It is also important to note that because rotation always takes place in a subspace (i.e., the space of the retained components), the choice of this subspace strongly influences the result of the rotation.
- From: Abdi & Williams, 2010, Principal component analysis

So, how many components to retain?

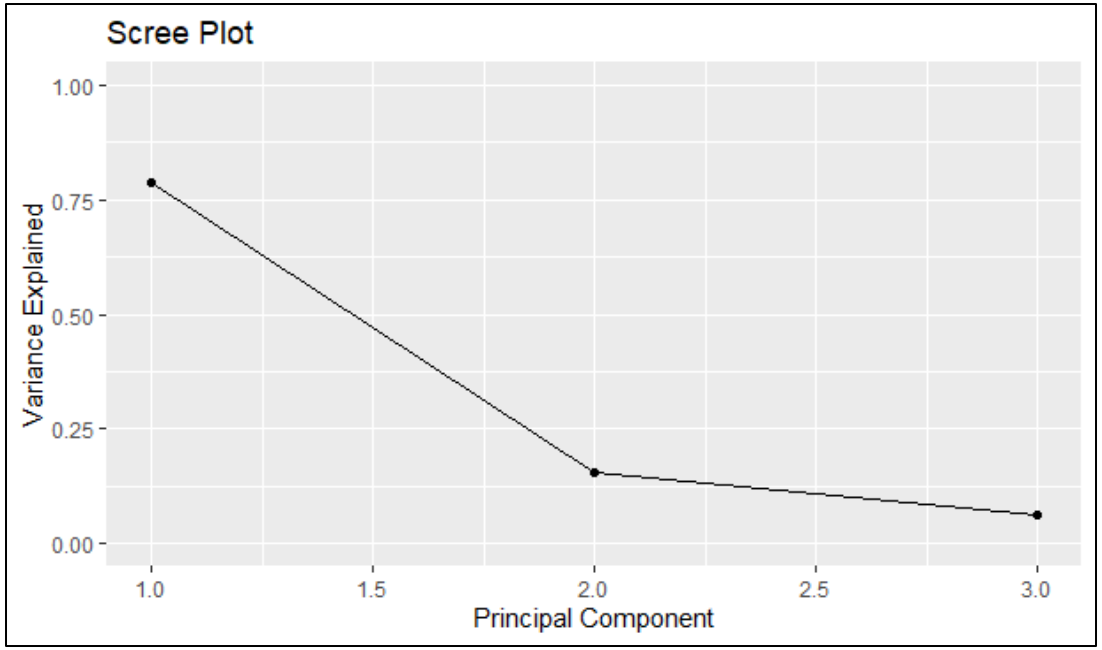
We need scree plot of pca.1 model!

#calculate total variance explained by each principal component

- var_explained = pca.1$sdev^2 / sum(pca.1$sdev^2)

#create scree plot

- library(ggplot2)
- qplot(c(**1:3**), var_explained) +
- geom_line() +
- xlab("Principal Component") +
- ylab("Variance Explained") +
- ggtitle("Scree Plot") +
- ylim(0, 1)



pca.1$sdev^2 = pca.1 (sdev-square) = pca.1 (eigenvalue)

It will be wise to use Kaiser's rule and Scree plot to decide how to many components to retain for the problem in hand! We can use scree plot's suggestion for now: 2

Screeplot model: PCA with 2-components?

If only 1-component is retained, it will be “unidimensional”!

#Final DRM: Unrotated PCA with 2 components

- fa.3 <- psych::principal(USArrests.1, nfactors = 2, rotate = "none")

- fa.3

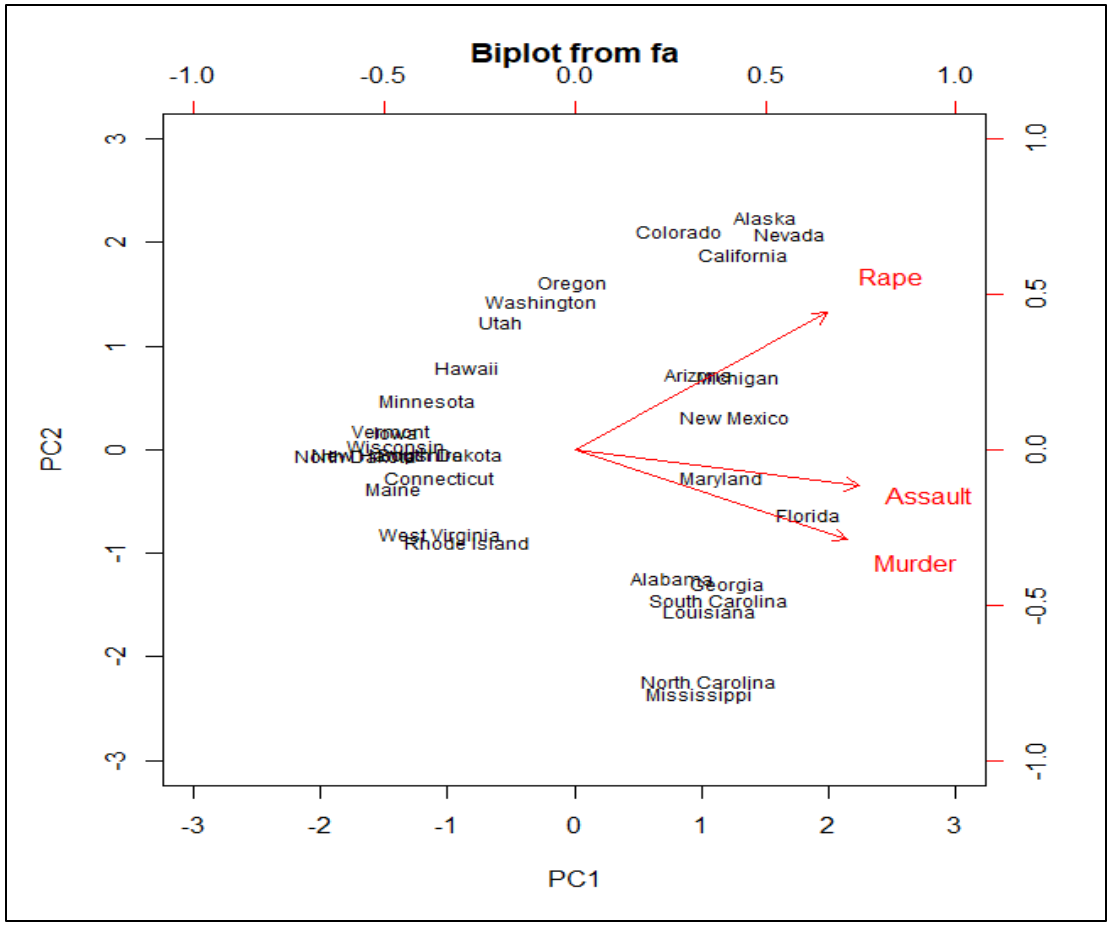
	PC1	PC2
• SS loadings	2.36	0.46
• Proportion **Var**	**0.79**	**0.15**
• Cumulative Var	0.79	0.94
• Prop. Explained	0.84	0.16
• Cum. Proportion	0.84	1.00

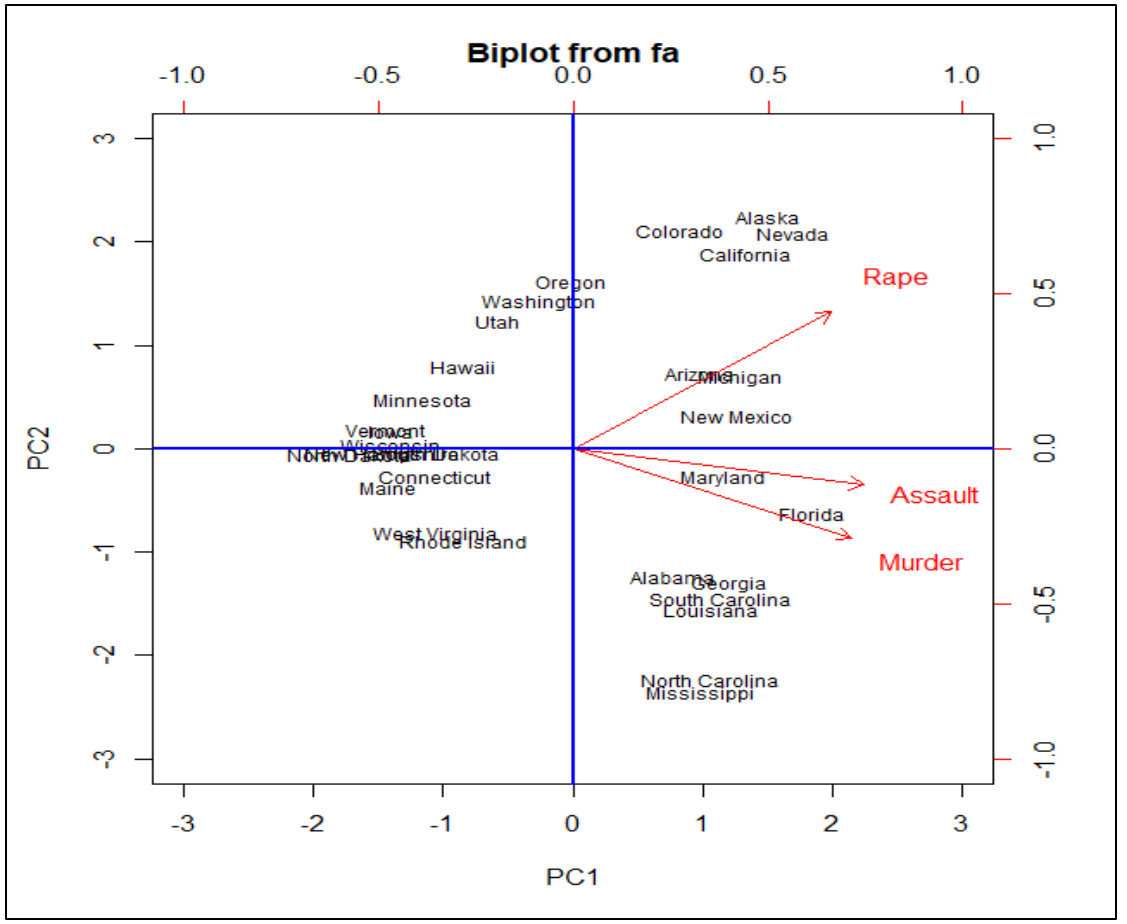
Latent variable = criminality score can be created using two variables.

These two variables explains around 94% of the variance in the latent variable!

Bi-plot: https://statisticsglobe.com/biplot-pca-explained

biplot(fa.3, labels = rownames(USArrests.1))





Final components:

- PC1 and PC2 components using Scree plot
- This is “suggestive”

- PC1 = Single component using Kaiser’s criteria of EV > 1
- This is “confirmative”
- So, it is better to use PC1 as the **“criminality score”** to represent 3 features/variables using PCA method in the USArrests data

Question/queries so far?

Multi-dimensional scaling (MDS)

https://www.statisticshowto.com/multidimensional-scaling/

- Multidimensional scaling is a **visual representation of distances or dissimilarities between sets of objects**.
- "Objects" can be colors, faces, map coordinates, political persuasion, or any kind of real or conceptual stimuli (Kruskal and Wish, 1978).
- Objects that are more **similar** (or have shorter distances) are closer together on the graph than objects that are less similar (or have longer distances).
- As well as interpreting dissimilarities as distances on a graph, MDS can also serve as a dimension reduction technique for high-dimensional data (Buja et. al, 2007).

Data Visualization with MDS paper – Buja et.al.

https://faculty.wharton.upenn.edu/wp-content/uploads/2012/04/Buja-Swayne-Littman-Dean-Hofman-Chen-JCGS-2008-06-Vol17-Data-Visualization-With-Multidimensional-Scaling.pdf

- There exist several types of MDS, and they differ mostly in the loss function they use.

- Kruskal-Shepard distance scaling

vs

- Classical Torgerson–Gowerinner-product scaling

Or,

- Metric scaling

vs

- Nonmetric scaling

Basic steps of classical MDS:

https://www.statisticshowto.com/multidimensional-scaling/

- **1. Assign a number of points to coordinates in n-dimensional space**.
- *N*-dimensional space could be **2-dimensional, 3-dimensional**, or higher spaces (at least, theoretically, because 4-dimensional spaces and above are difficult to model).
- The orientation of the coordinate axes is arbitrary and is mostly the researcher's choice.

- **2. Calculate Euclidean distances for all pairs of points**.
  The Euclidean distance is the "as the crow flies" straight-line distance between two points x and y in Euclidean space.
- It's calculated using the Pythagorean theorem ($c^2 = a^2 + b^2$), although it becomes somewhat more complicated for n-dimensional space (see "Euclidean Distance in n-dimensional space"). This results in the similarity matrix.

Remaining basic steps:

- **3. Compare the similarity matrix with the original input matrix** by evaluating the **stress** function.
- *Stress* is a goodness-of-fit measure, based on differences between predicted and actual distances.
- In his original 1964 MDS paper, **Kruskal wrote that fits close to zero are excellent**, while anything over 0.2 should be considered "poor".
- More recent authors suggest evaluating stress based on the **quality of the distance matrix and how many objects** are in that matrix.
- **4. Adjust coordinates, if necessary, to minimize stress**.

Let’s fit MDS in the USArrests data:

#Getting scaled data sans 3rd variable:

USArrests.1 <- scale(USArrests[,-3])

#Classical MDS using Kruskall’s stress

We first need to get the distance

#Distance calculation

- state.disimilarity <- dist(USArrests.1)

- #MDS fit
- mds.1 <- cmdscale(state.disimilarity)
- summary(mds.1)

- #Stress
- ks <- MASS::isoMDS(state.disimilarity, k=2)
- ks$stress = 5.467755 is a fair fit!

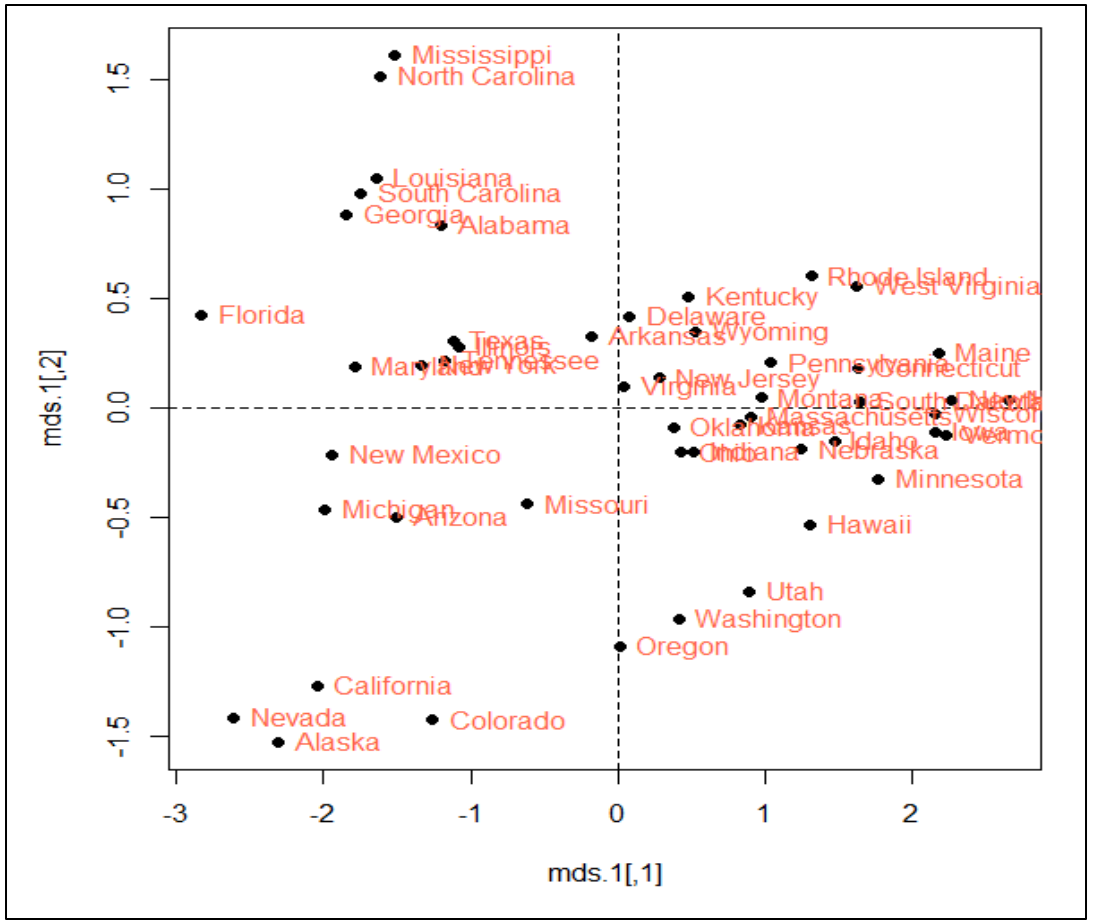
More here: https://rpubs.com/YaPi/393252

Let's plot the MDS:
This is a bi-plot so interpret as we did it!

#MDS plot

- plot(mds.1, pch = 19)
- abline(h=0, v=0, lty=2)
- text(mds.1, pos = 4, labels = rownames(USArrests.1), col = 'tomato')

#What is the interpretation of this plot?



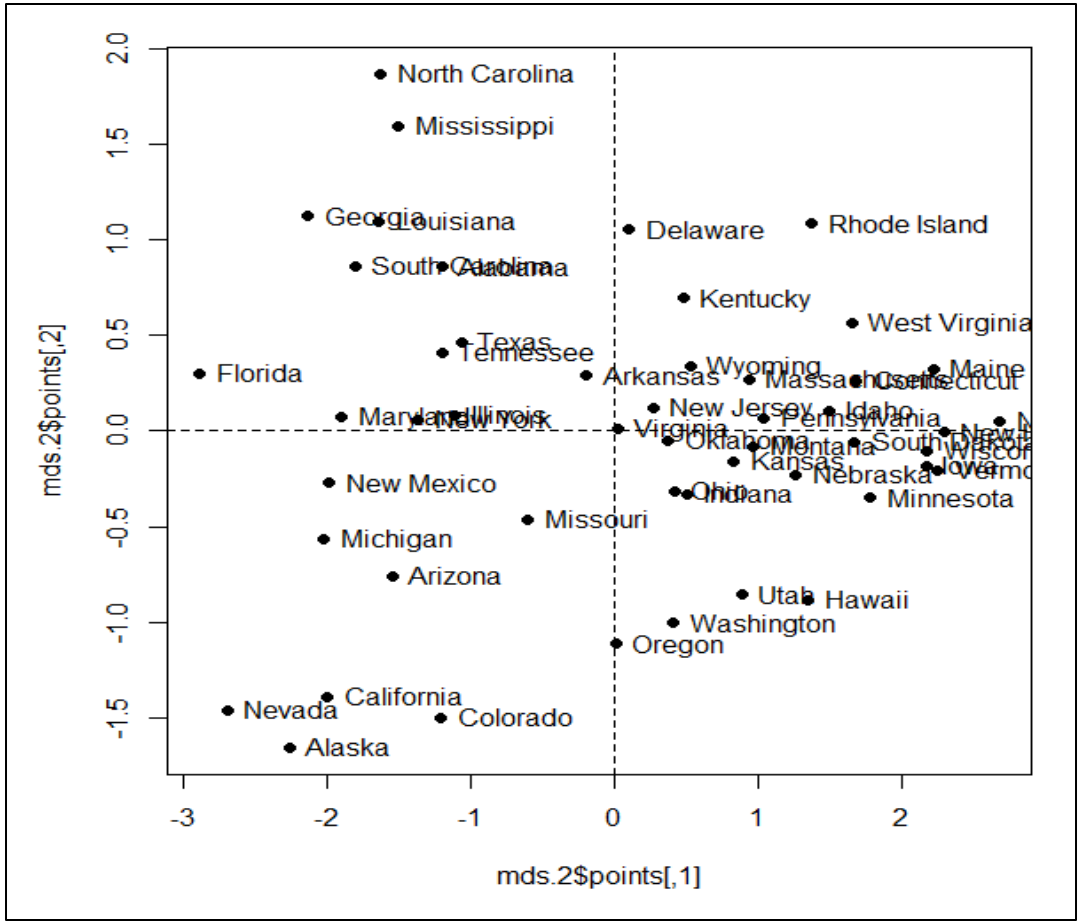
Can we improve it?

We can try using Sammon’t stress!

#Alternative approach with Sammon's stress

- mds.2 <- MASS::sammon(state.disimilarity, trace = FALSE)
- plot(mds.2$points, pch = 19)
- abline(h=0, v=0, lty=2)
- text(mds.2$points, pos = 4, labels = rownames(USArrests.1))

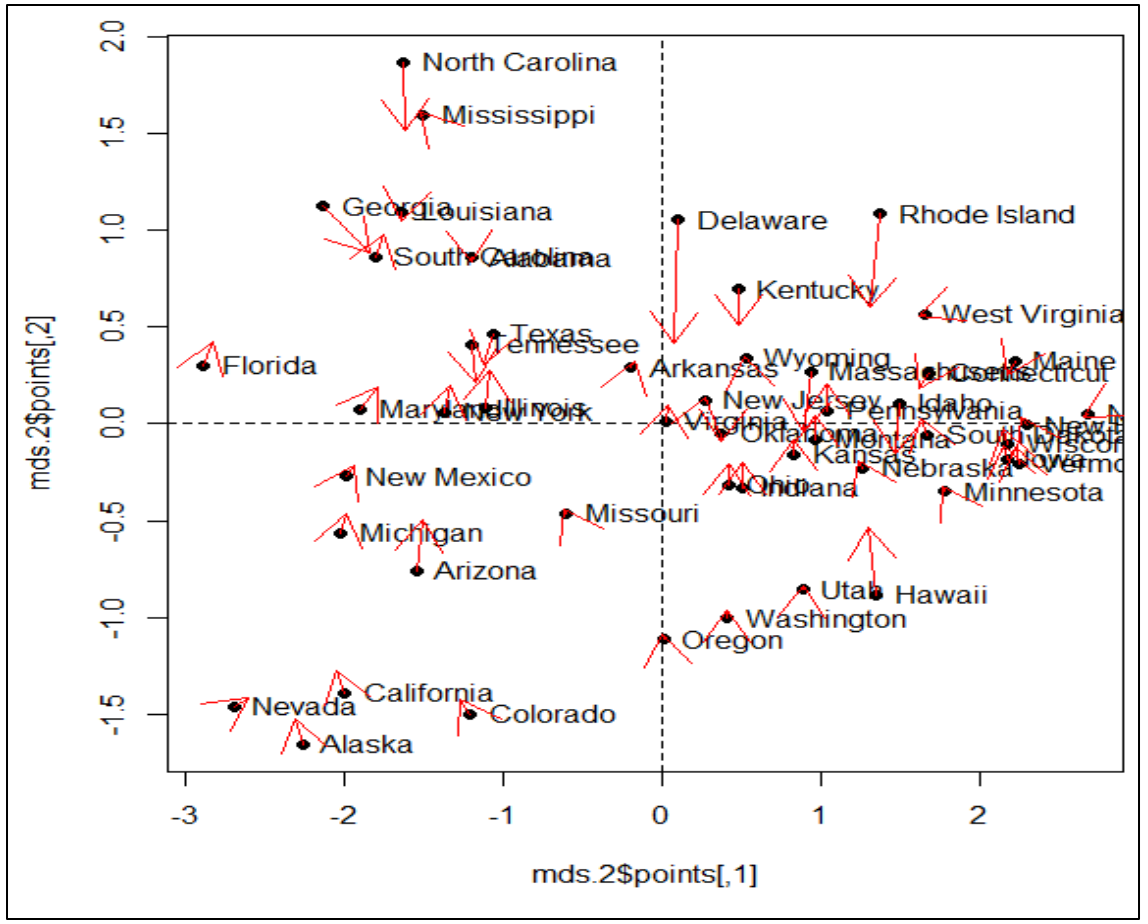
Get stress and compare with previous one, better?



Comparing pca.1 (last class) and mds.2 models with annotations:

 # Compare with PCA (first two PCs):

- arrows(
- x0 = mds.2$points[,1], y0 = mds.2$points[,2],
- x1 = pca.1$x[,1], y1 = pca.1$x[,2],
- col='red', pch=19, cex=0.5)



Question/queries?

- Next class

Unsupervised learning with:

- Clustering
  - K-means cluster analysis
  - HCA cluster analysis

Thank you!

@shitalbhandary