# INTRO TO DATA VISUALIZATIONS

# Goals for

- Understand what makes a visualization effective through the study of core principles
- Critically evaluate a visual representation of data by looking at various examples in media (newspapers, television and so on)
- Gain hands-on experience with visualization tools /libraries
- Incorporate visualization principles to build an interactive visualization of your own data

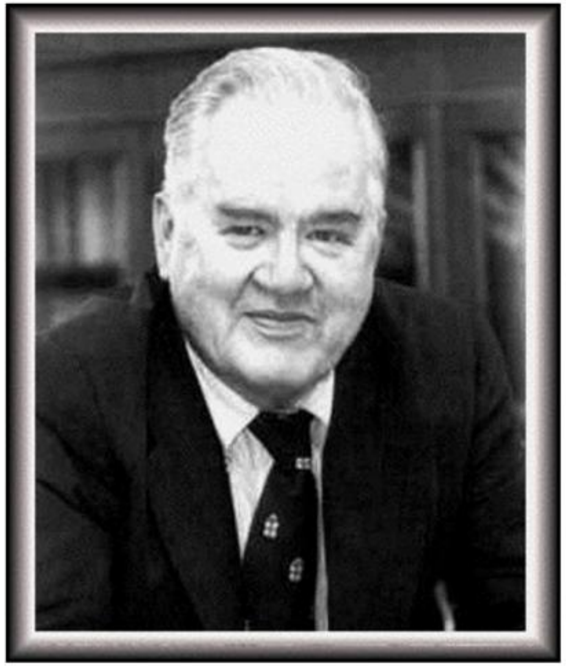
# What is Data Visualization?

- Visual representation of data

- For the purpose of exploration, discovery, and insight

# Why is Data Visualization important?

- Data Visualization is an important part of EDA.



JOHN W. TUKEY*

We often forget how science and engineering function. Ideas come from previous exploration more often than from lightning strokes. Important questions can demand the most careful planning for confirmatory analysis. Broad general inquiries are also important. Finding the question is often more important than finding the answer. Exploratory data analysis is an attitude, a flexibility, and a reliance on display, NOT a bundle of techniques, and should be so taught. Confirmatory data analysis, by contrast, is easier to teach and easier to computerize. We need to teach both; to think about science and engineering more broadly; to be prepared to randomize and avoid multiplicity.

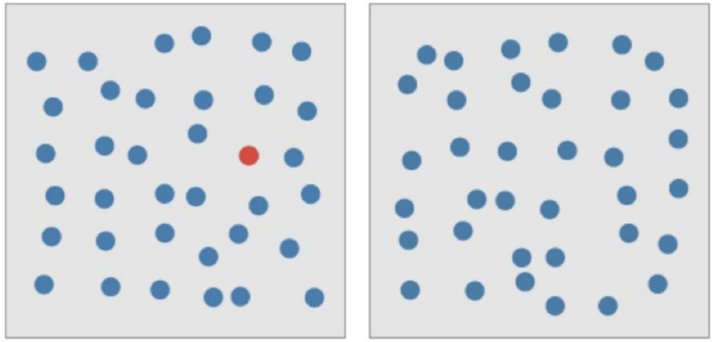
# Why is Data Visualization important?

- Baby Name Wizard
  - http://www.babynamewizard.com/voyager
- Origin of Species – Edits
  - http://benfry.com/traces/
- Netflix Queues
  - http://www.nytimes.com/interactive/2010/01/10/nyregion/20100110-netflix-map.html?ref=nyregion
- Unemployment Visualization (NYTimes)
  - http://www.nytimes.com/interactive/2009/11/06/business/economy/unemployment-lines.html
- r/dataisbeautiful subreddit

# Why is visualization important?

- Vision is the most powerful communication channel humans posses.
    - For instance, We can detect information faster than our eye can move ...

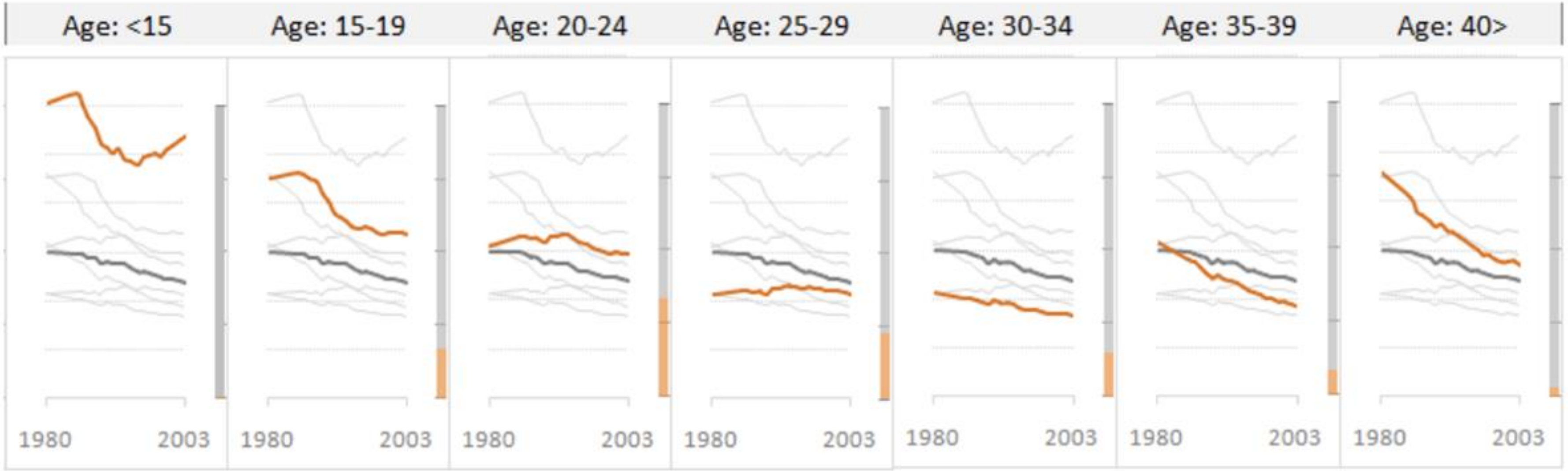
1) Preattentive features can be detected faster than eye movement (200 msec).



2) Humans are not very good at detecting patterns from numbers.

| | 1980 | 1985 | 1986 | 1987 | 1988 | 1989 | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Less than 15 years old | 607 | 624 | 605 | 578 | 553 | 523 | 515 | 502 | 511 | 492 | 488 | 479 | 493 | 498 | 504 | 497 | 512 | 519 | 529 | 537 |
| 15 to 19 years old | 451 | 462 | 457 | 449 | 444 | 418 | 403 | 379 | 370 | 364 | 353 | 347 | 350 | 346 | 341 | 337 | 339 | 341 | 339 | 337 |
| 20 to 24 years old | 310 | 328 | 328 | 327 | 327 | 318 | 328 | 330 | 333 | 334 | 326 | 317 | 314 | 307 | 301 | 297 | 296 | 298 | 296 | 293 |
| 25 to 29 years old | 213 | 219 | 219 | 216 | 218 | 213 | 224 | 224 | 228 | 230 | 227 | 224 | 228 | 226 | 224 | 221 | 220 | 219 | 215 | 211 |
| 30 to 34 years old | 213 | 203 | 201 | 197 | 194 | 189 | 196 | 192 | 192 | 189 | 183 | 179 | 178 | 176 | 174 | 171 | 169 | 171 | 169 | 167 |
| 35 to 39 years old | 317 | 280 | 277 | 265 | 254 | 244 | 249 | 241 | 239 | 234 | 226 | 219 | 215 | 208 | 203 | 200 | 195 | 195 | 190 | 186 |
| 40 years old and over | 461 | 409 | 381 | 374 | 361 | 350 | 354 | 339 | 338 | 329 | 320 | 309 | 301 | 291 | 290 | 283 | 276 | 276 | 278 | 268 |

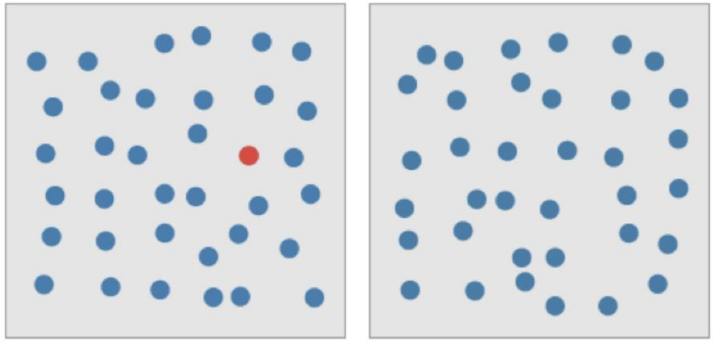
Which group has the highest/lowest rates? When?
Which group has an increasing/decreasing temporal trend?
Which group has a faster/slower rate of change?



# Why is visualization important?

- Vision is the most powerful communication channel humans posses.
  - For instance, We can detect information faster than our eye can move ...

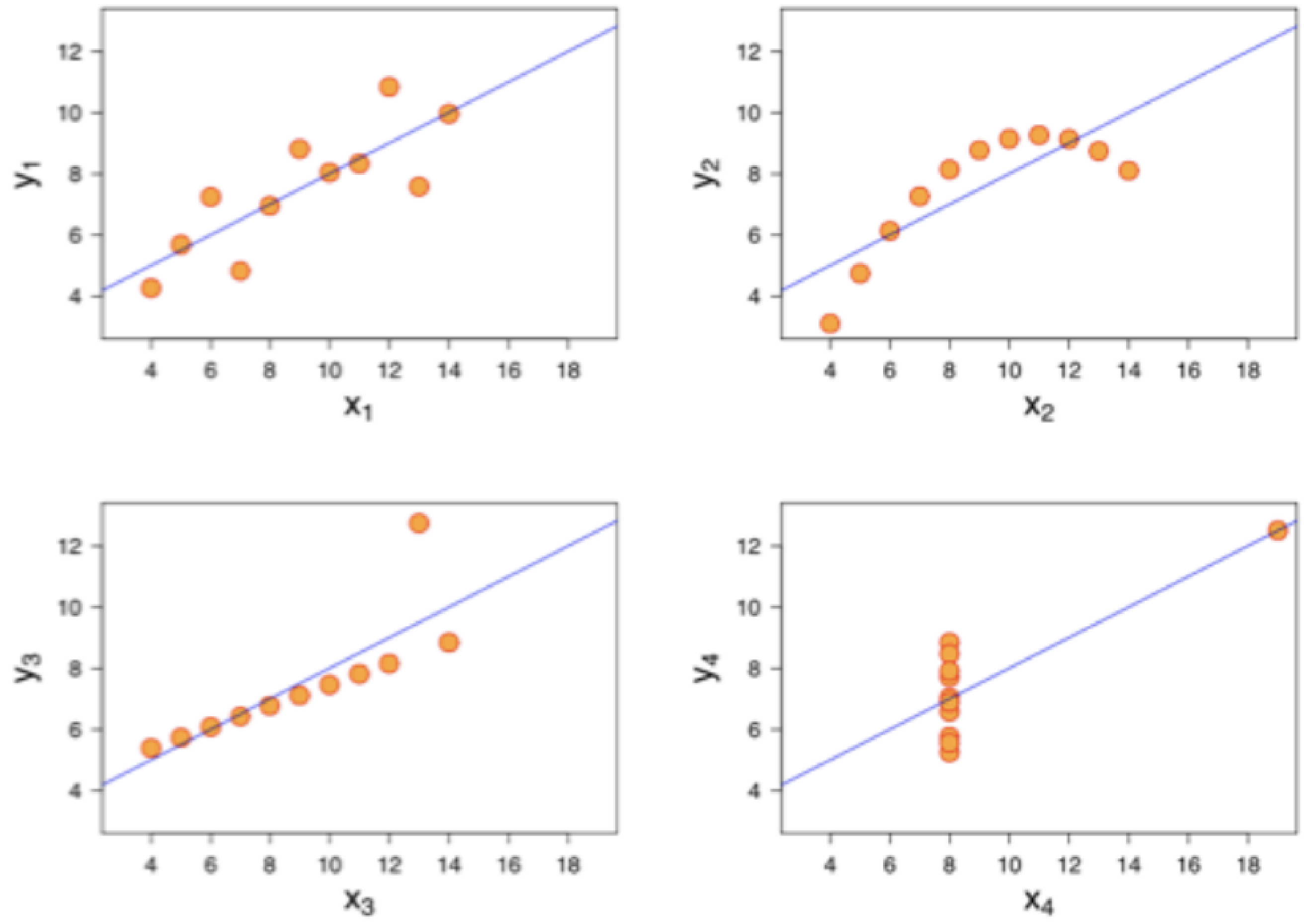
1) Preattentive features can be detected faster than eye movement (200 msec).



2) Humans are not very good at detecting patterns from numbers.

3) Summary statistics can hide important information

# Anscombe's Quartet




| Property | Value |
|---|---|
| Mean of $x$ in each case | 9 (exact) |
| Variance of $x$ in each case | 11 (exact) |
| Mean of $y$ in each case | 7.50 (to 2 decimal places) |
| Variance of $y$ in each case | 4.122 or 4.127 (to 3 decimal places) |
| Correlation between $x$ and $y$ in each case | 0.816 (to 3 decimal places) |
| Linear regression line in each case | $y = 3.00 + 0.500x$ (to 2 and 3 decimal places, respectively) |

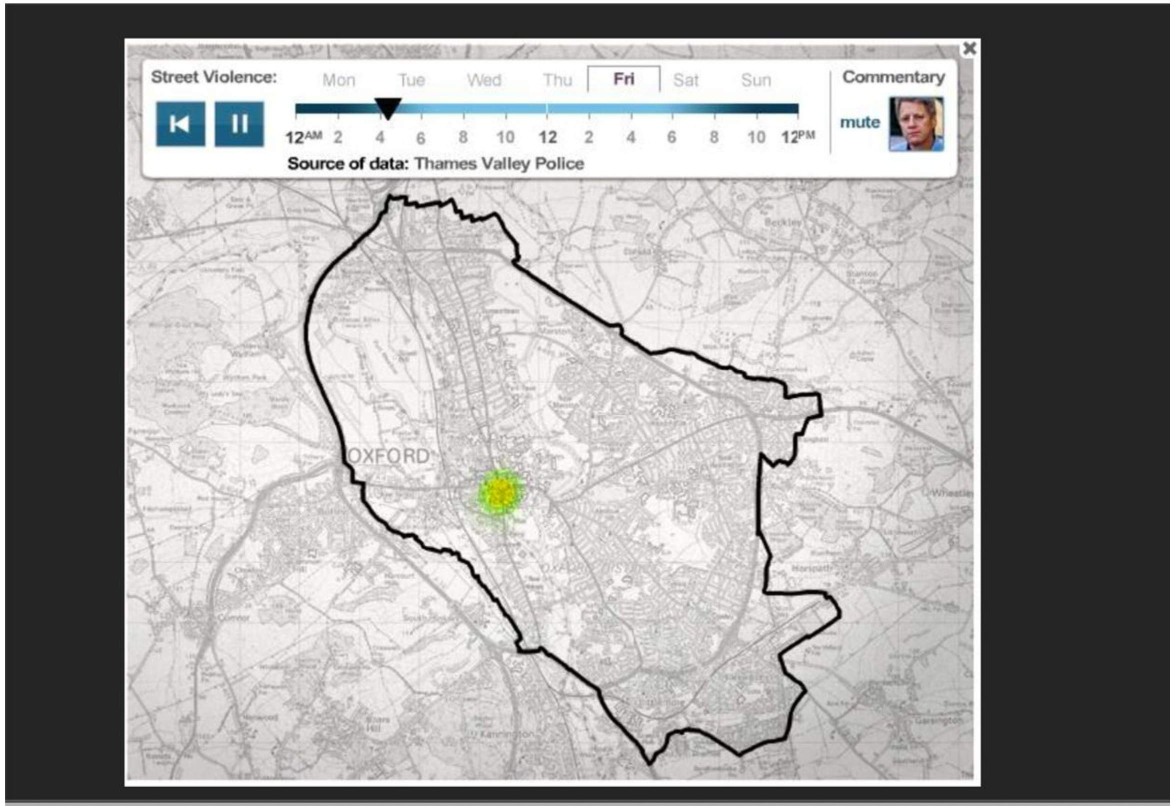
# Principles for good data visualizations

- 1. **Balance the design**
  - A balanced design is one with the visual elements like shape, color, negative space and texture equally distributed across the plot.

- There are three different types of balances in design:
- Symmetrical – Each side of the visual is the same as the other.
- Asymmetrical – Both sides are different but still have a similar visual weight.
- Radial – Elements are placed around a central object which acts as an anchor.

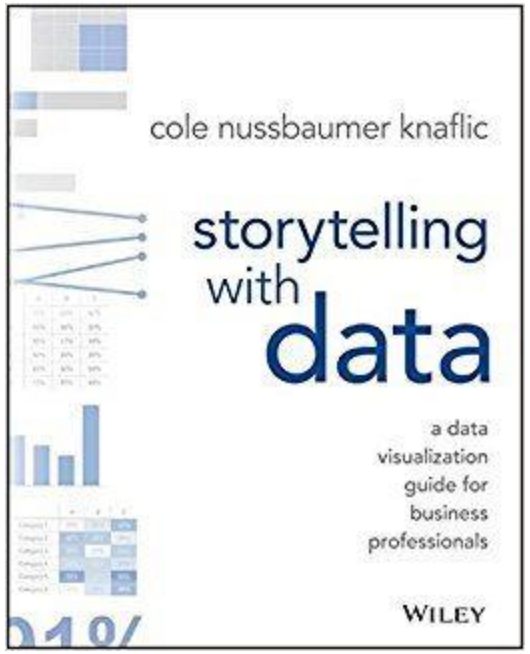
# Principles for good data visualizations

- **Emphasize the key areas**
  - The user's attention should be drawn to the right data points by carefully choosing the size, colors, contrast and negative space.



# Principles for good data visualizations

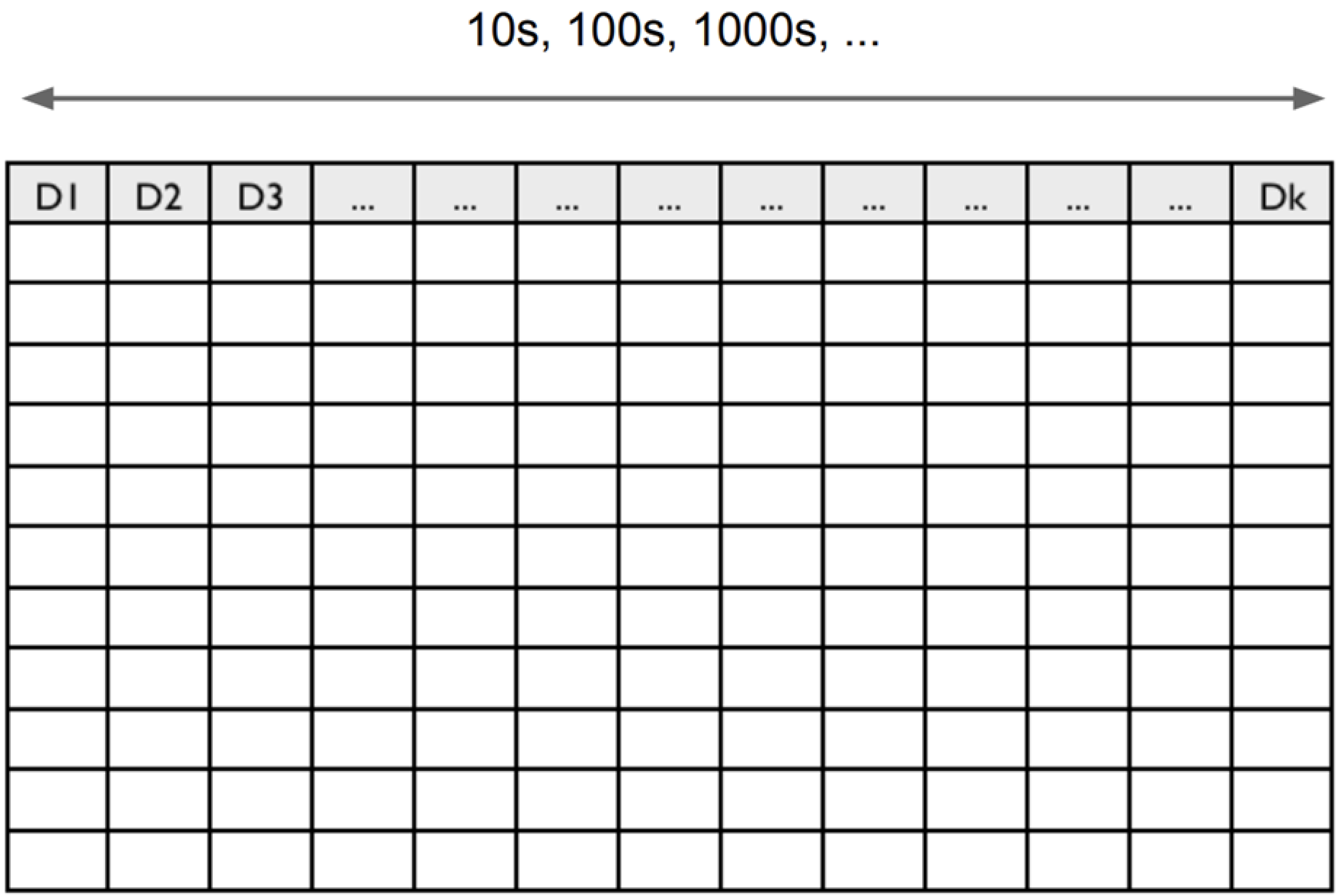
Check out this book on the UCR library (available online)



# Problems with building visualizations

- There’s simply too much data.
- Too many dimensions/features.
- Too many options and possible views.

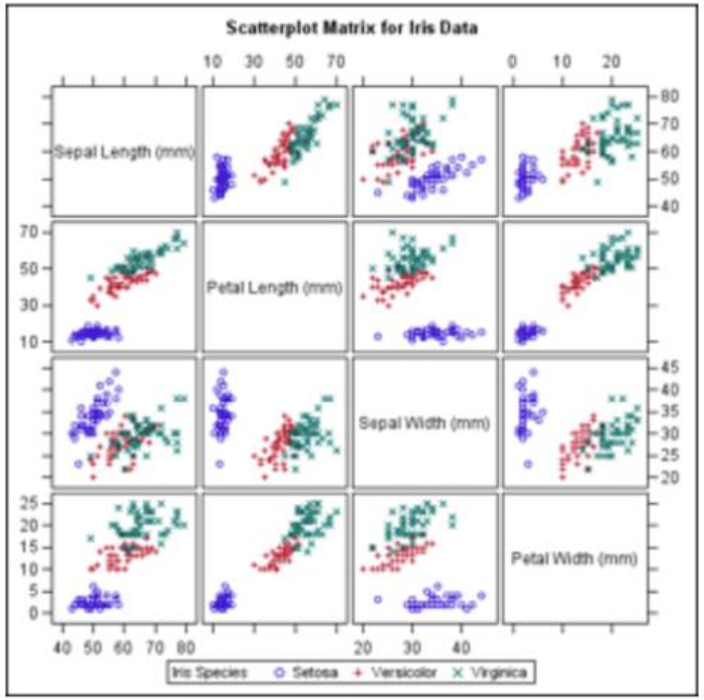
# High-Dimensional Data Visualization



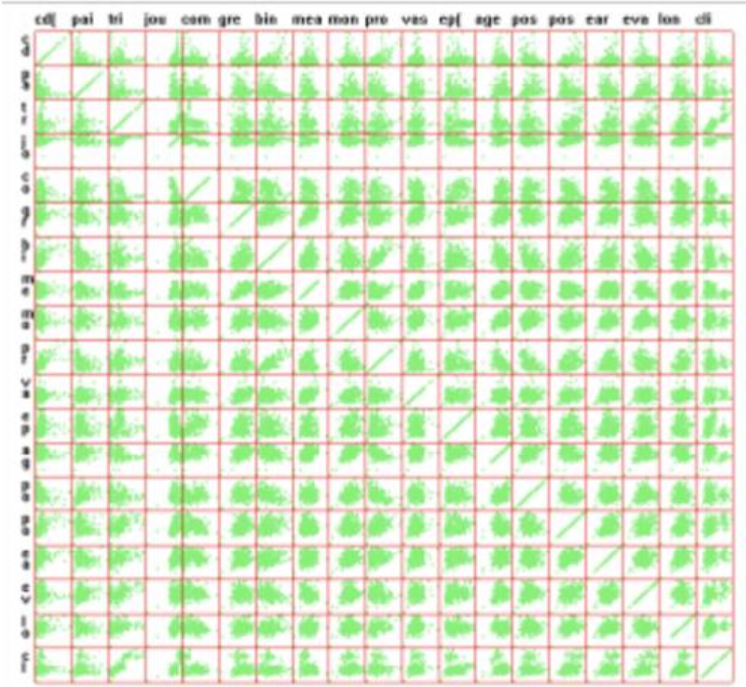

- Set of visual features very limited
- Resolution very limited
- Ability to make sense of it very limited!

# Example: Scatter Plot Matrix

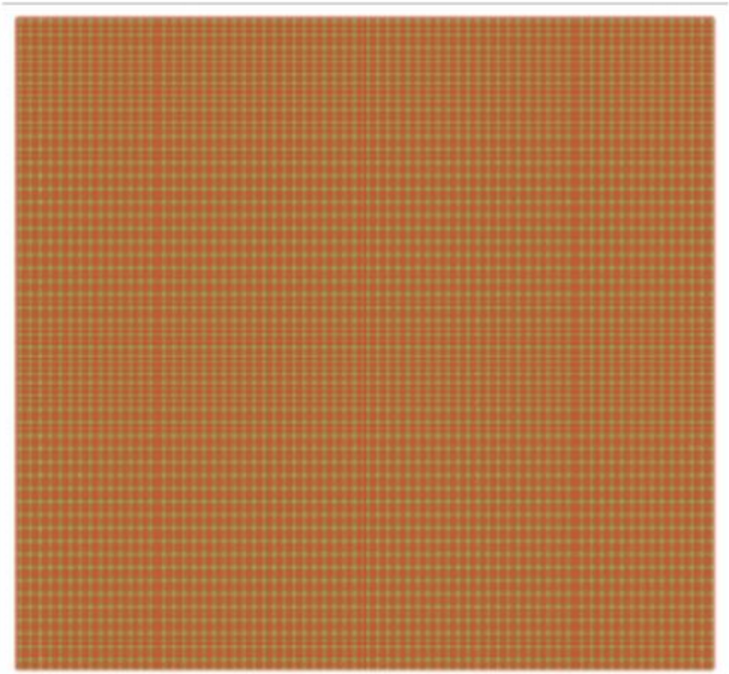
4 dimensions




20 dimensions



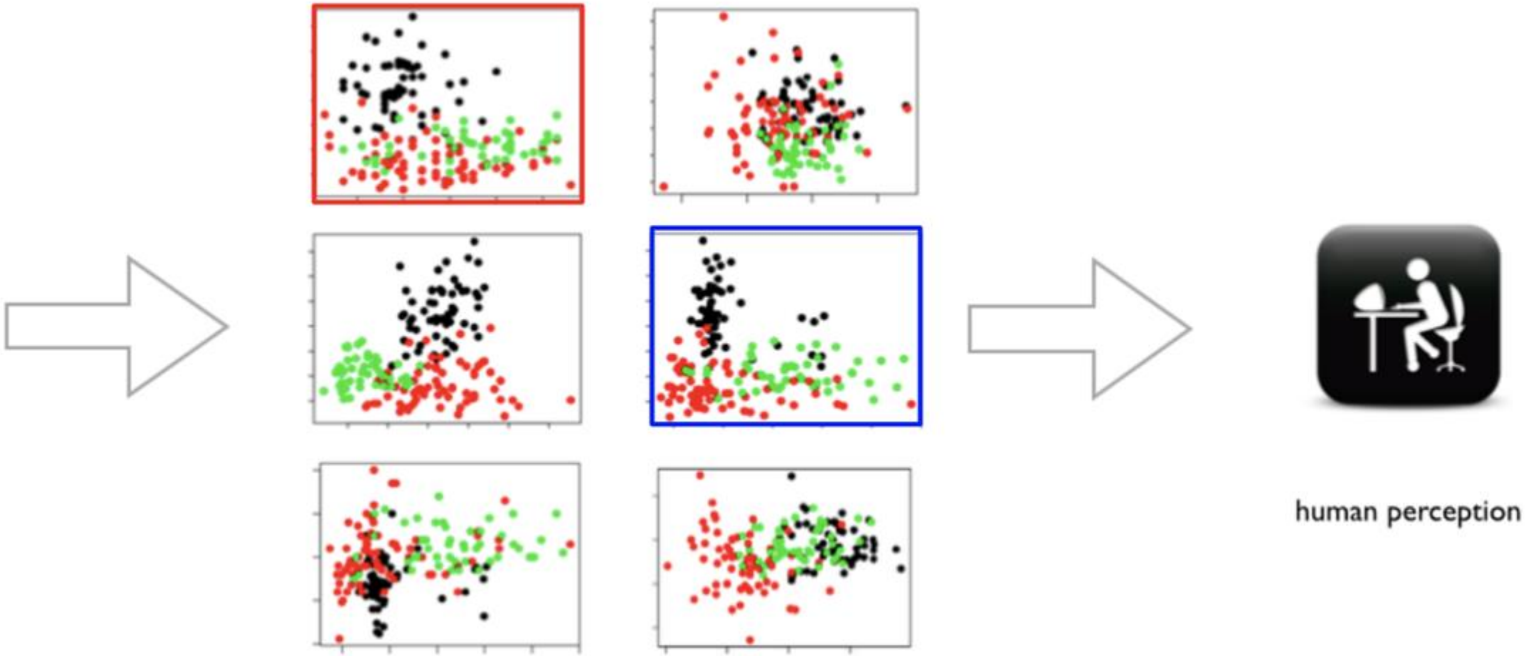
100 dimensions



Taken from: Jing Yang’s Interactive Hierarchical Dimension Ordering, Spacing and Filtering for Exploration of High Dimensional Datasets.

# Quality Metrics-Driven Visualization

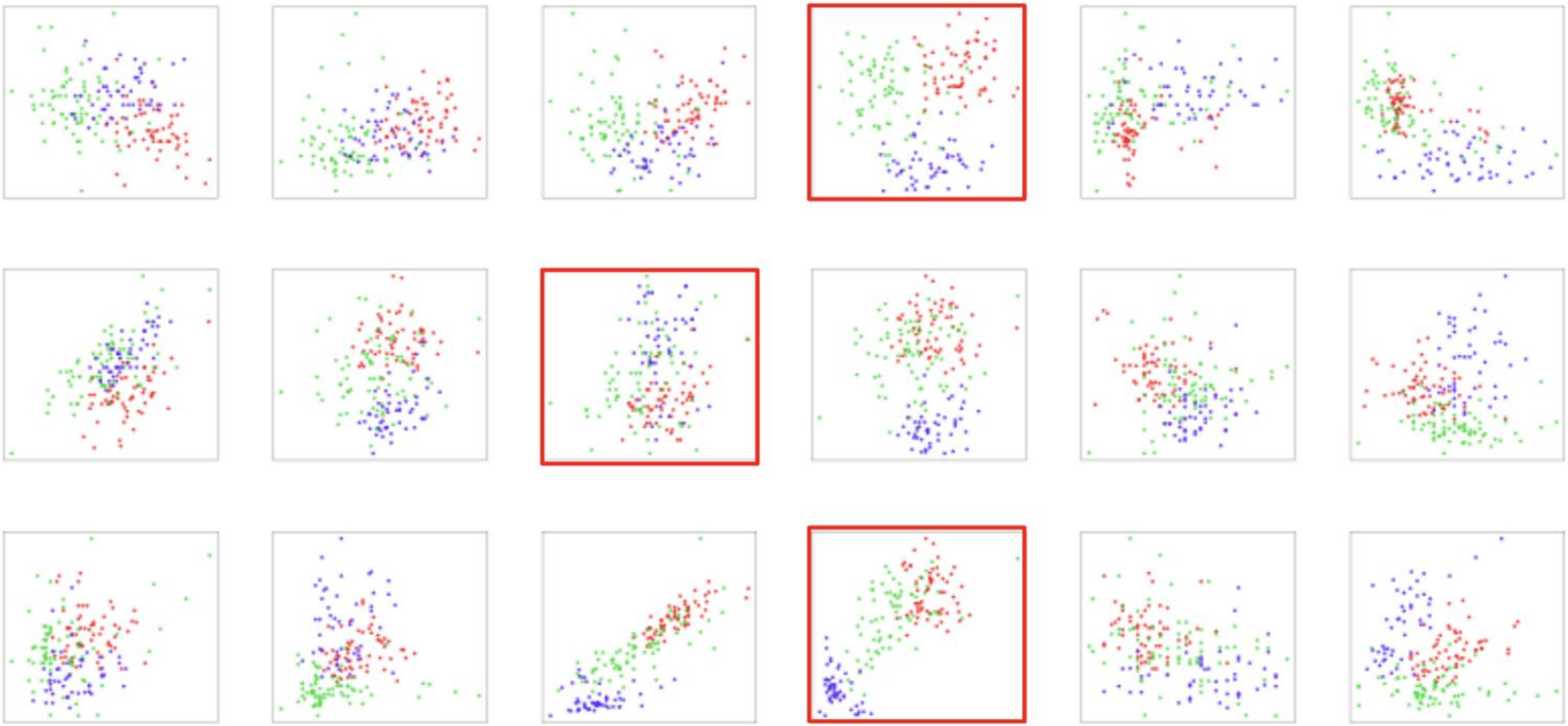
| KA | f2 | f3 | f4 | f5 | f6 | f7 | ... | fn |
|---|---|---|---|---|---|---|---|---|
| 2 | 60 | 733 | 476 | 72 | 73 | 46 | ... | 63 |
| 2 | 422 | 37 | 319 | 4635 | 422 | 37 | ... | 463 |
| 1 | 62 | 201 | 519 | 33 | 612 | 282 | ... | 39 |
| 3 | 154 | 22 | 405 | 59 | 15 | 224 | ... | 45 |
| 3 | 611 | 320 | 461 | 545 | 61 | 32 | ... | 54 |
| 1 | 431 | 67 | 22 | 173 | 431 | 67 | ... | 173 |
| 2 | 165 | 0 | 254 | 112 | 244 | 535 | ... | 12 |
| 1 | 40 | 16 | 307 | 352 | 43 | 61 | ... | 52 |
| 3 | 877 | 327 | 596 | 54 | 77 | 27 | ... | 55 |
| 2 | 92 | 12 | 244 | 535 | 192 | 48 | ... | 35 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |




How can we measure the interestingness of a projection?

Does it correlate with human perception?

# Automatic Ranking of Class Separation



# Many features

- When we have more than three to four quantitative variables, all-against-all scatter plot matrices quickly become unwieldy.

- Dimension reduction relies on the key insight that most high-dimensional datasets consist of multiple correlated variables that convey overlapping information.

- Such datasets can be reduced to a smaller number of key dimensions without loss of much critical information.

# DIMENSIONALITY REDUCTION

# Curse of Dimensionality

- The **curse of dimensionality** refers to various phenomena that arise when analyzing and visualizing data in high-dimensional spaces.

- Dimensionality Reduction
  - It reduces the time and storage space required for our Machine Learning algorithms
  - Removal of multi-collinear features improves the performance of the machine learning model.
  - It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.

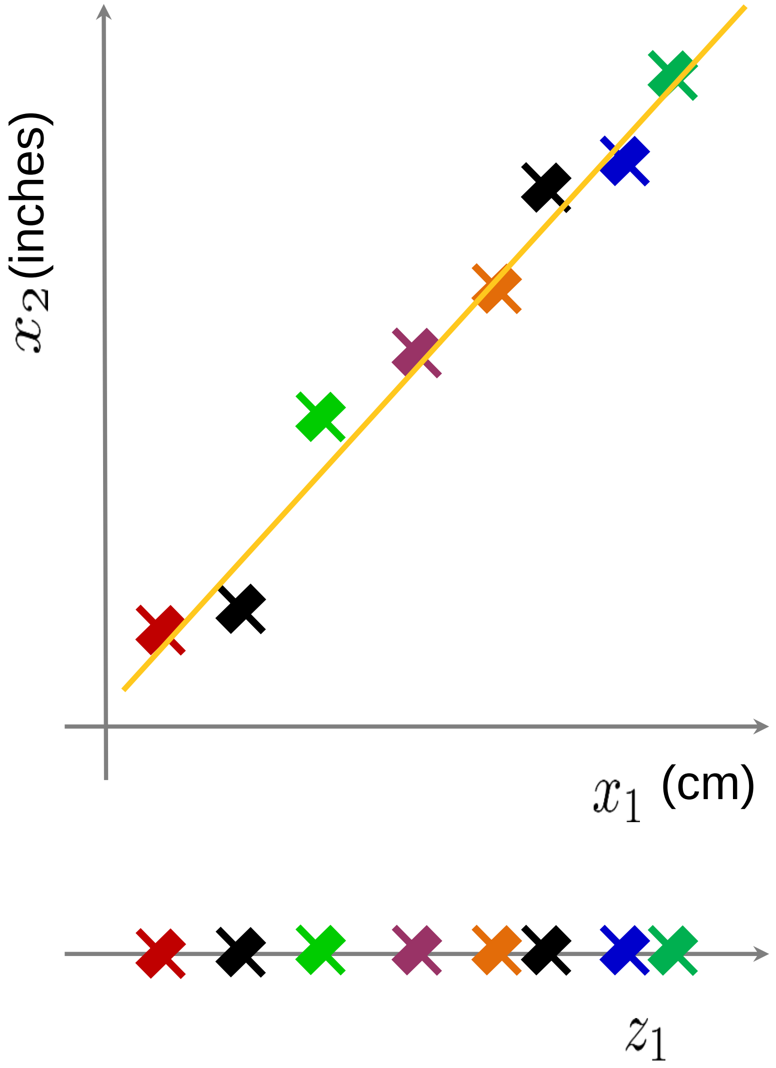
# Dimensionality Reduction

- **The main idea:** reduce the dimensionality of the space.

- Project the d-dimensional points in a k-dimensional space so that:
  - k << d
  - distances are preserved as well as possible

- Solve the problem in low dimensions

# Data Compression




Reduce data from
2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

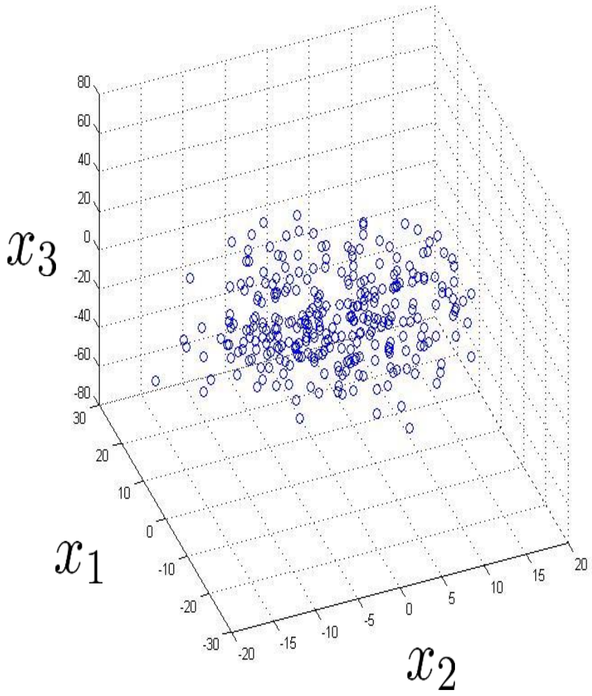
$$x^{(2)} \rightarrow z^{(2)}$$

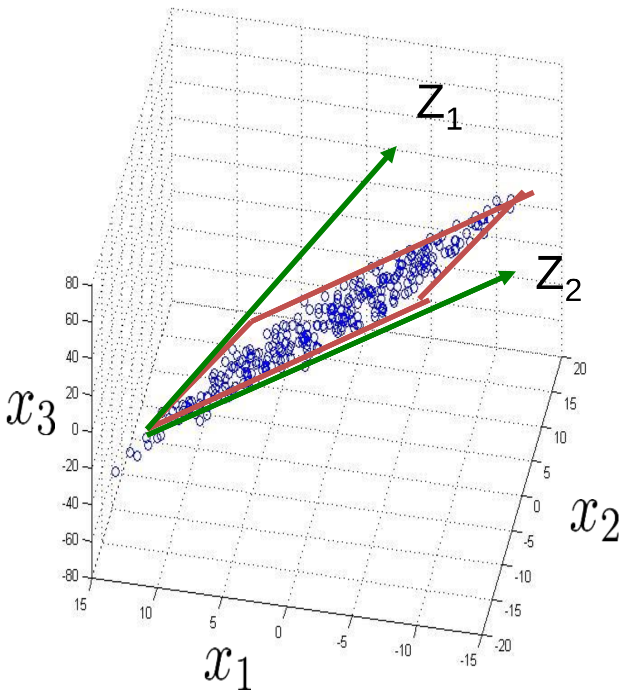
$$\vdots$$

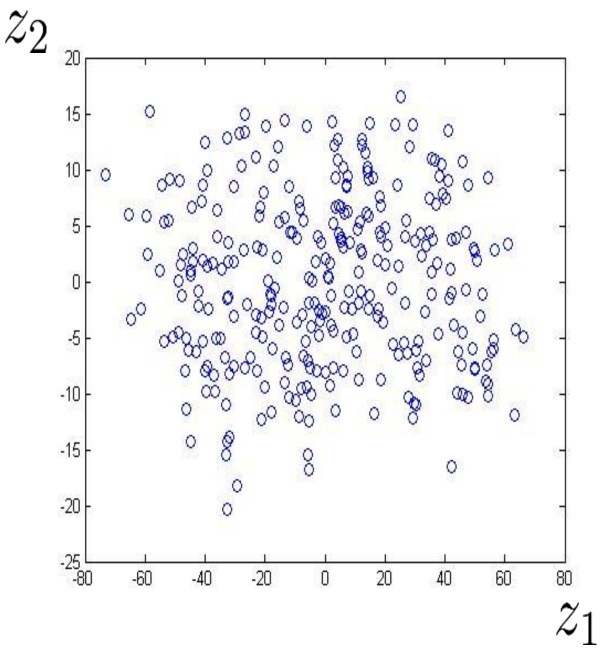
$$x^{(m)} \rightarrow z^{(m)}$$

# Data Compression

Reduce data from 3D to 2D



# Data Visualization

$x \in \mathbb{R}^{50}$

| Country | $X_1$ GDP (trillions of US$) | $X_2$ Per capita GDP (thousands of intl. $) | $X_3$ Human Development Index | $X_4$ Life expectancy | $X_5$ Poverty Index (Gini as percentage) | $X_5$ Mean household income (thousands of US$) | … |
|---|---|---|---|---|---|---|---|
| Canada | 1.577 | 39.17 | 0.908 | 80.7 | 32.6 | 67.293 | … |
| China | 5.878 | 7.54 | 0.687 | 73 | 46.9 | 10.22 | … |
| India | 1.632 | 3.41 | 0.547 | 64.7 | 36.8 | 0.735 | … |
| Russia | 1.48 | 19.84 | 0.755 | 65.5 | 39.9 | 0.72 | … |
| Singapore | 0.223 | 56.69 | 0.866 | 80 | 42.5 | 67.1 | … |
| USA | 14.527 | 46.86 | 0.91 | 78.3 | 40.8 | 84.3 | … |
| … | … | … | … | … | … | … | |

[resources from en.wikipedia.org]

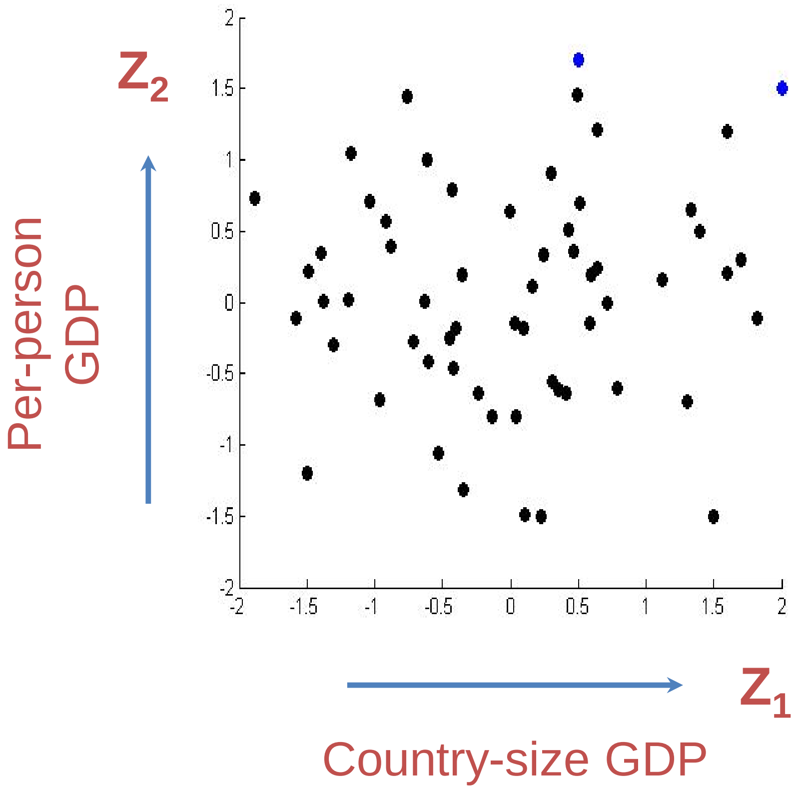
# Data Visualization

Reduce features from 50 to 2.

The vectors $Z_1$ and $Z_2$ summarize the features in some way

$$z^{(i)} \in \mathbb{R}^2$$

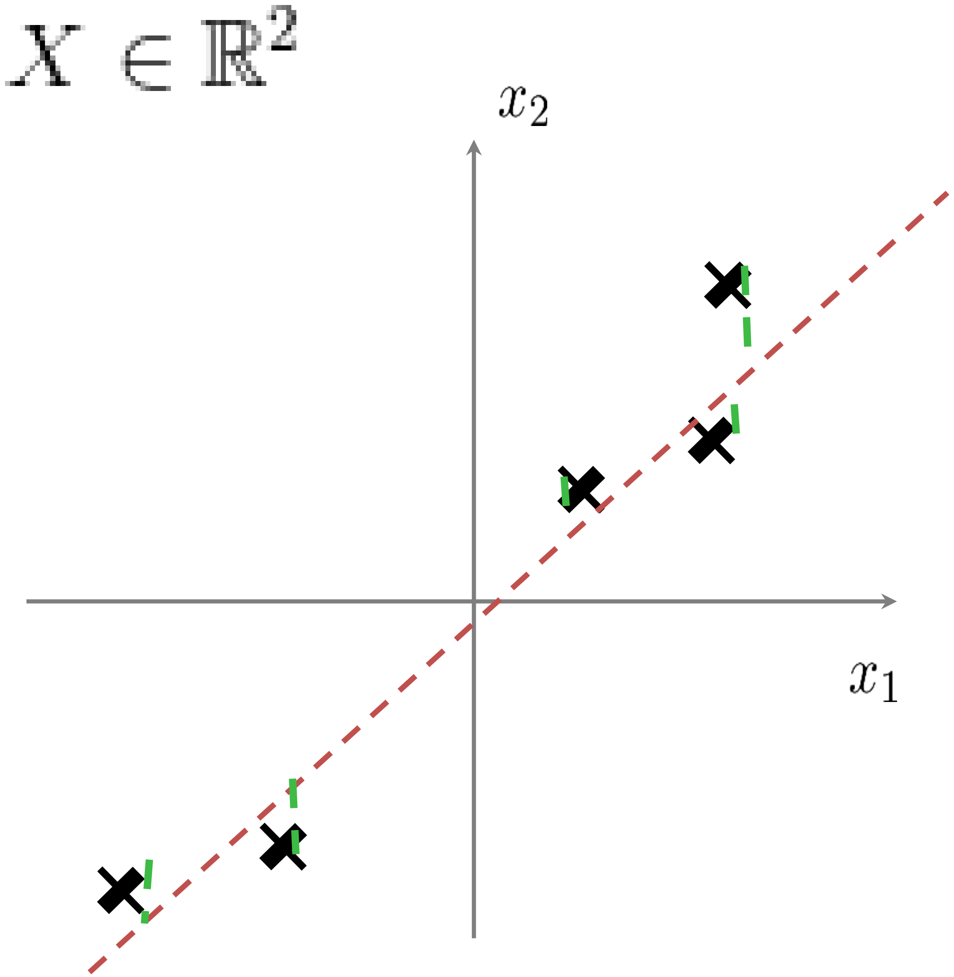
| Country | $Z_1$ | $Z_2$ |
|---|---|---|
| Canada | 1.6 | 1.2 |
| China | 1.7 | 0.3 |
| India | 1.6 | 0.2 |
| Russia | 1.4 | 0.5 |
| Singapore | 0.5 | 1.7 |
| USA | 2 | 1.5 |
| … | … | … |



# Method for Dimensionality Reduction

- Principle Component Analysis (PCA)
  - One of the most popular methods

# Principal Component Analysis (PCA) problem formulation

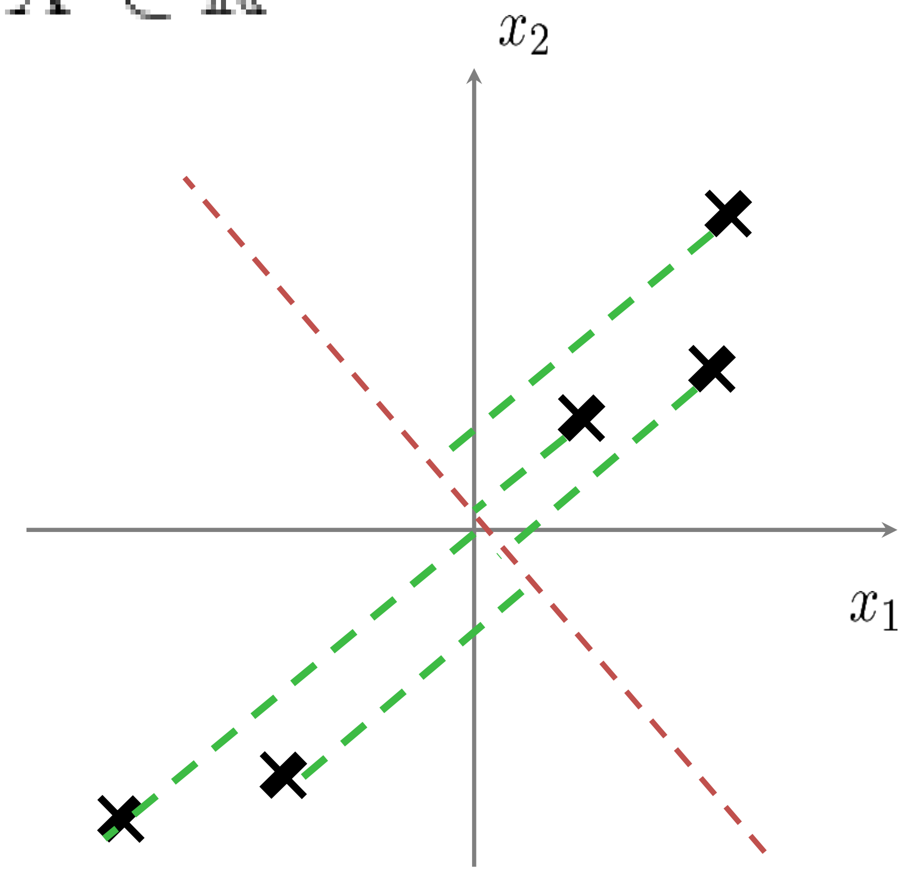



- What are we trying to do? We want to reduce the dimensions from 2D to 1D

- We want to find a line on to which to project the data
  - Distance between point $x_i$ and the projected line is minimized… sum of squares is minimized.
  - This is called projection error

# Principal Component Analysis (PCA) problem formulation

$X \in \mathbb{R}^2$

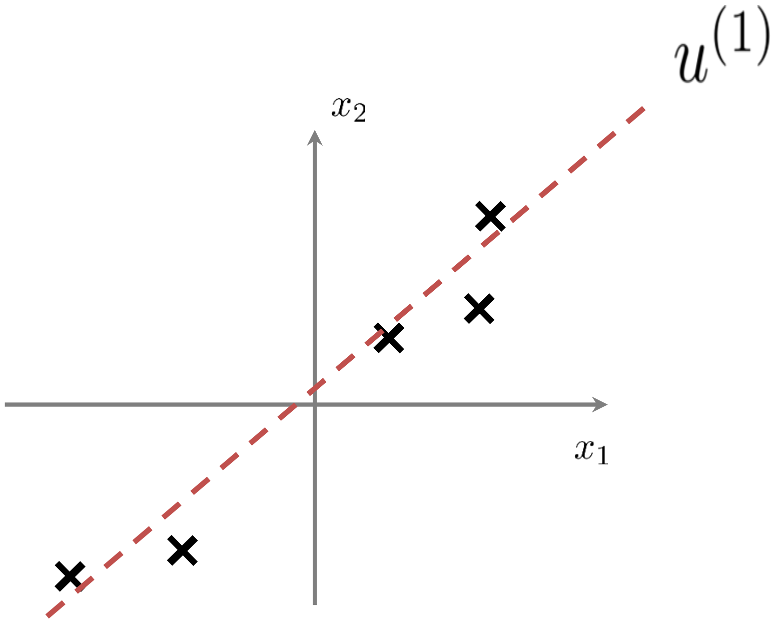


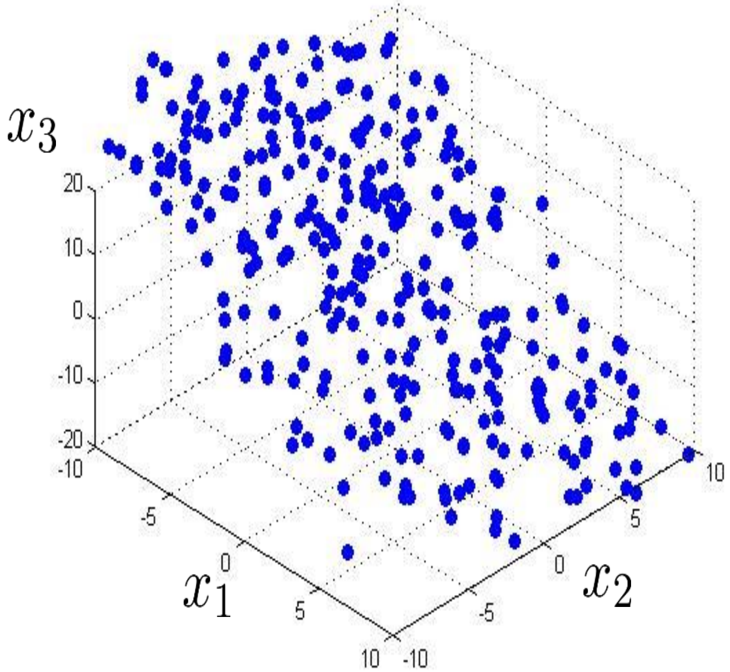

- What are we trying to do? We want to reduce the dimensions from 2D to 1D

- Consider this new projection, the points have to move a huge distance to get projected onto this line.

- Hence, the previous projection is better because it minimizes the error.

# Principal Component Analysis (PCA) problem formulation



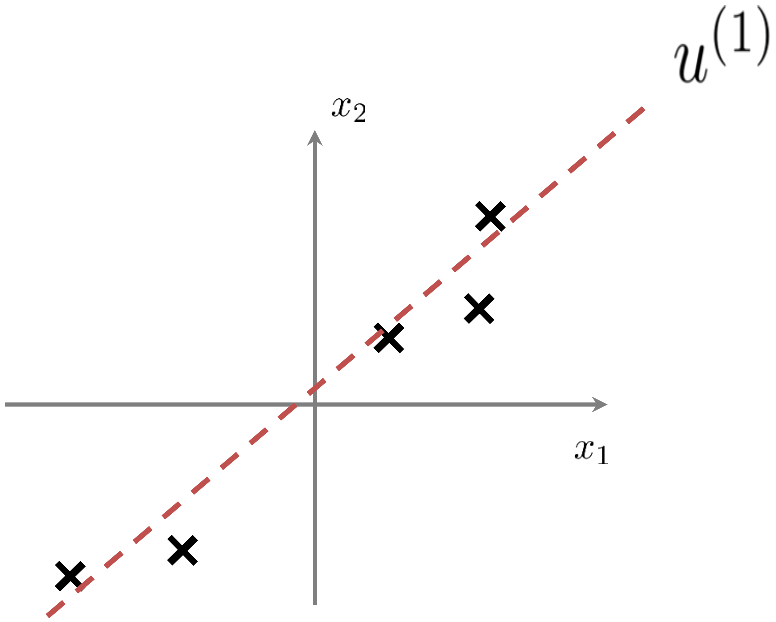




$$3D \rightarrow 2D$$
$$K = 2$$

**Reduce from 2-dimension to 1-dimension:**
Find a direction a vector onto which to project the data so as to minimize the projection error. $u^{(1)} \in \mathbb{R}^n$

Reduce from n-dimension to k-dimension: Find vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

# Principal Component Analysis (PCA) problem formulation




$$x^{(i)} \in \mathbb{R}^2 \longrightarrow z^{(i)} \in \mathbb{R}$$

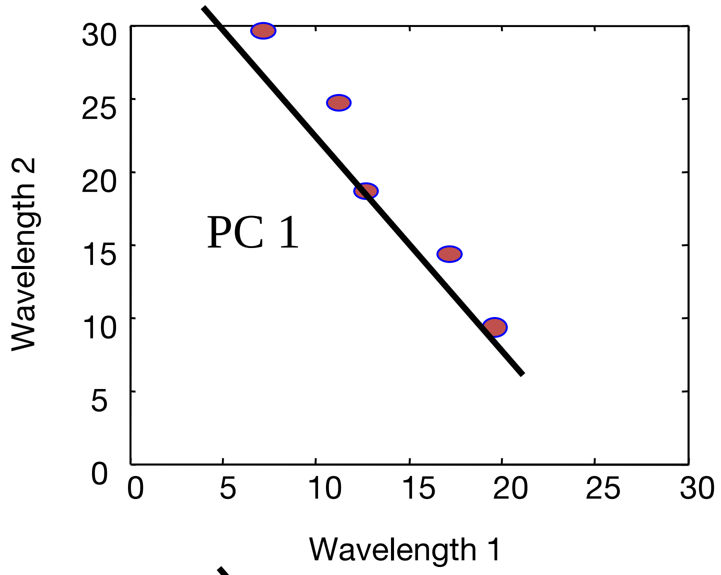
# Principal Component Analysis

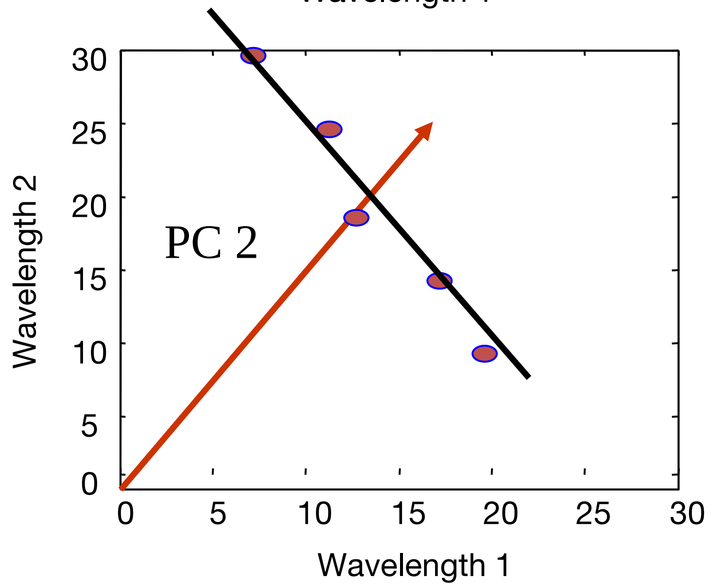
- So what are principal components then?
  - Underlying structure in the data.
  - They are the directions where there is the most variance, the directions where the data is most spread out.

- All principal components (PCs) start at the origin of the ordinate axes.

- First PC is direction of maximum variance from origin

- Subsequent PCs are orthogonal to 1st PC and describe maximum residual variance



# Step 1 - Data Preprocessing

**Training set:** $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$

Then divide each term ($x_j - \mu_j$) by the standard deviation σ ( or you can also divide by $X_{max}$-$X_{min}$)

# Step 2 – Apply PCA

Reduce data from **n**-dimensions to **k**-dimensions

**Compute "covariance matrix":**

nxn

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$$

measure of how much two random variables change together

nx1 1xn

**Compute "eigenvectors" of matrix :**

- Use SVD – Single Value Decomposition
- Eigenvector is a direction
- Every eigenvector has a eigenvalue, which is the variance in the data in that direction.
- The eigenvector with the highest eigenvalue is therefore the principal component.

# Step 2 – Apply PCA (Cont.)

How to compute the eigenvector?

Apply SVD (Single Value Decomposition) on covariance matrix to get [U,S,V] matrices.

$$U = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix}}_{\text{K vectors}} \in \mathbb{R}^{n \times n}$$

If we want to reduce the dimensions from n to k

Then we simply take the k vectors $u^1, u^2, \dots, u^k$

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{u_{Reduced}} X$$

# Choose ‘k’ Number of principle components

Average squared projection error:

Total variation in the data:

Typically, choose $k$ to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2} \leq 0.01 \qquad (1\%)$$

“99% of variance is retained”

# PCA Example (Spark / python)

- See pca_example.py under <SPARKPATH>/examples/src/main/python/ml

```
data =   [(Vectors.sparse(5, [(1, 1.0), (3, 7.0)]),),  \
          (Vectors.dense([2.0, 0.0, 3.0, 4.0, 5.0]),), \
          (Vectors.dense([4.0, 0.0, 0.0, 6.0, 7.0]),)]

df = spark.createDataFrame(data, [ "features” ])

pca = PCA(k=3, inputCol= "features", outputCol= "pcaFeatures")

model = pca.fit(df)

result = model.transform(df).select( "pcaFeatures” )

result.show(truncate=False)
```

A dataframe is a dataset (think about a matrix) that is organized into “named” columns.

WAIT!!! What about preprocessing the data by feature scaling and mean normalization!!!

# PCA Example (Spark / python)

- See standard_scaler_example.py under <SPARKPATH>/examples/src/main/python/mlib/standard_scaler_example.py

```
data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
dataFrame = sqlContext.createDataFrame(data)

scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures",
                        withStd=True, withMean=False)

# Compute summary statistics by fitting the StandardScaler
scalerModel = scaler.fit(dataFrame)

# Normalize each feature to have unit standard deviation.
scaledData = scalerModel.transform(dataFrame)
```

You can specify which to apply, or apply both

# Choose ‘k’ Number of principle components

Algorithm:

Try PCA with $k = 1$

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Check if

Avg squared projection error

Total variation in the data

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01?$$

```
[U,S,V] = svd(Sigma)
```

$$S = \begin{bmatrix} s_{11} & 0 & 0 & 0 \\ 0 & s_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & s_{nn} \end{bmatrix}$$

K =2

For a given k,

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

# Choose k' Number of principle components

```
[U,S,V] = svd(Sigma)
```

Pick smallest value of $k$ for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$$

(99% of variance retained)

# Supervised Learning Speedup

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$$\downarrow PCA$$

$$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$$

New training set:

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

# Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

- Visualization

# When to use PCA?

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train classifier $\{(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$ . Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$

# A 2D Numerical Example

# PCA Example – STEP 1

- Subtract the mean from each of the data dimensions.
- All the x values have $\bar{x}$ subtracted and y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations.
- The variance and co-variance values are not affected by the mean value.

# PCA Example –STEP 1

DATA:

| x | y |
|---|---|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

ZERO MEAN DATA:

| x | y |
|---|---|
| .69 | .49 |
| -1.31 | -1.21 |
| .39 | .99 |
| .09 | .29 |
| 1.29 | 1.09 |
| .49 | .79 |
| .19 | -.31 |
| -.81 | -.81 |
| -.31 | -.31 |
| -.71 | -1.01 |

# PCA Example –STEP 1

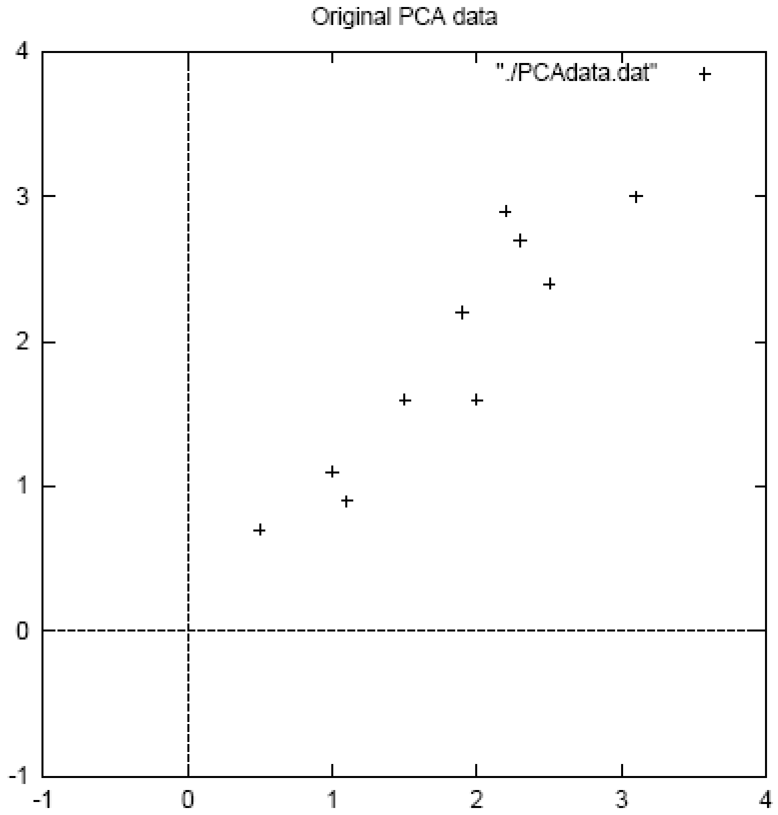



Figure 3.1: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data

# PCA Example –STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

# PCA Example –STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix
- $$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$
- $$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

# PCA Example –STEP 3

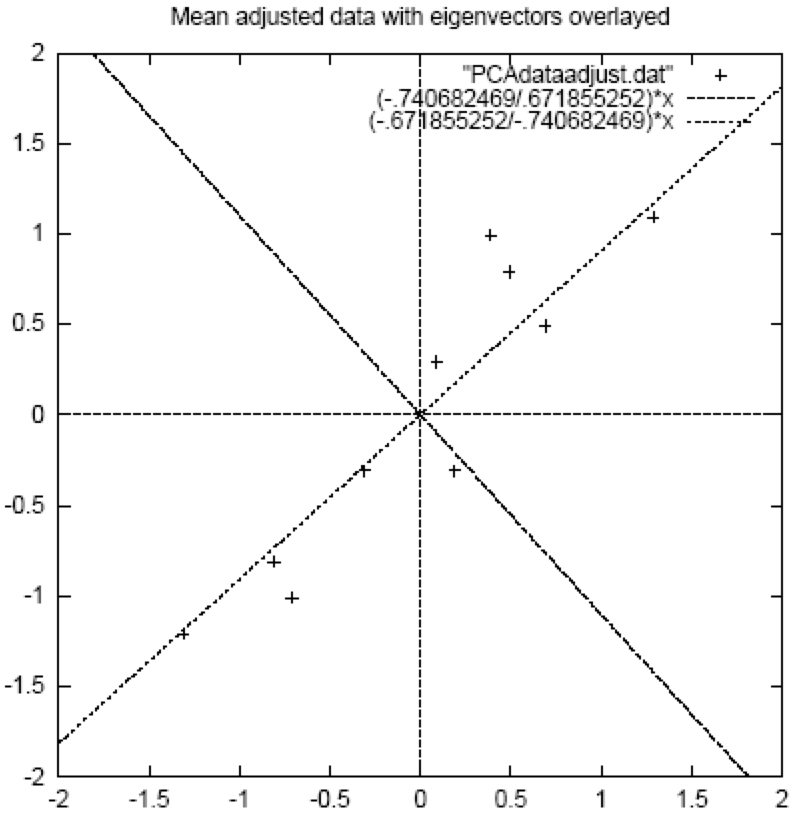



Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.

- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

# PCA Example –STEP 4

- Reduce dimensionality and form feature vector
  - the eigenvector with the highest eigenvalue is the principle component of the data set.

- In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data.

- Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance.

# PCA Example –STEP 4

- Now, if you like, you can decide to ignore the components of lesser significance.

- You do lose some information, but if the eigenvalues are small, you don't lose much
  - n dimensions in your data
  - calculate n eigenvectors and eigenvalues
  - choose only the first p eigenvectors
  - final data set has only p dimensions.

# PCA Example –STEP 4

- Feature Vector
  - FeatureVector = (eig1 eig2 eig3 … eign)
  - We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

  - or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} - \quad .677873399 \\ - \quad .735178656 \end{pmatrix}$$

# PCA Example – STEP 5

- Deriving the new data
  - FinalData = RowFeatureVector x RowZeroMeanData
  - RowFeatureVector is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top
  - RowZeroMeanData is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension.

# PCA Example –STEP 5

FinalData transpose: dimensions along columns

| x | y |
|---|---|
| -.827970186 | -.175115307 |
| 1.77758033 | .142857227 |
| -.992197494 | .384374989 |
| -.274210416 | .130417207 |
| -1.67580142 | -.209498461 |
| -.912949103 | .175282444 |
| .0991094375 | -.349824698 |
| 1.14457216 | .0464172582 |
| .438046137 | .0177646297 |
| 1.22382056 | -.162675287 |

# PCA Example –STEP 5

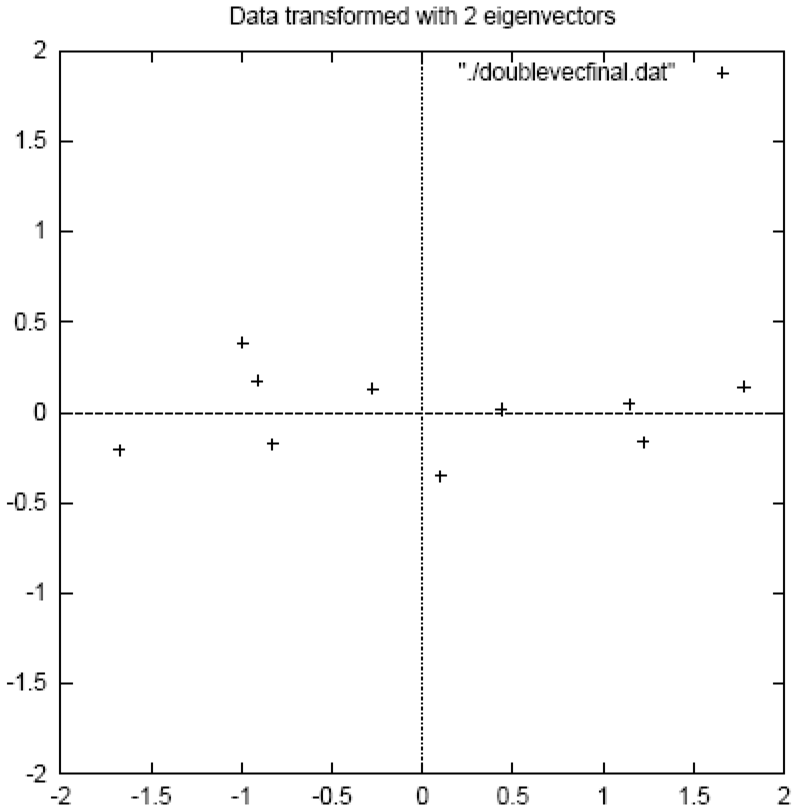



Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.