R programming
clustering, PCA, regression

Slides 2-15 are modified from https://wiki.illinois.edu/wiki/display/HPCBio/2020+Workshops

Slides 16-24 are modified from https://course.ccs.neu.edu › Clustering_Overview

Slides 42-50 are modified from https://web.stanford.edu › hrp259 › lecture13

Slides 52 to 58 are modified from *https://harvard-iacs.github.io › presentation*

How R Works

- R has **strict formats** for entering commands and referring to objects; commands that are not properly formatted will not be evaluated.

- () and “” must come in pairs; if not done correctly, R will indicate command is not finished, or will return error

- R is **case-sensitive**, except for file names on Windows/Mac

  Plot != plot but “myfile.txt” == “MyFile.txt”

- Spaces generally do not matter, except within quotes

  temp<-c(1,2) == temp <- c (1 , 2)

- To use \ must use \\ , else use /

Getting Help and Codes

- Help: ?function or help(function)

  for example: ?read.table or ??read.table

- Html help:

  1) type help.start()

  2) Menu: help→html help

Help for functions

- type in ?rownames
- Anatomy of a help page:
  - very top: main.function (package)
  - Title
  - Sections:
    - **Description**
    - **Usage**: names arguments in order with (usually) default values
    - **Arguments**: description and possible input
    - **Details**: further information
    - **Value**: the output of the function
    - ... possibly other sections
    - **Note**: any other useful information
    - **References**: see for more information, what to cite
    - **See Also**: related functions
    - **Examples**: how can be used

Command-line

- R has a **command-line** driven interface; entered commands are evaluated and the proper output is returned

```
2+2
3*3
3+8*2
log10(1000)
log2(8)
abs(-10)
sqrt(81)
```

2 key concepts in R

1. **Object**
    1. Holds information
    2. “class” of the object depends on type/s and arrangement of information

2. **Function**
    1. Pre-written code to perform specific commands
    2. Typically used on objects

“types” of information or objects

Most common:

Numeric – 1, 2, 426, 4500, etc.

Character – “a”, “B”, “data”, “cell line”, etc.

Factor – reads as character, but treated as categorical

Logical – TRUE or FALSE

T	F
1	0

Missing - NA to indicate missing values

Common object "classes"

vector – a series of data, all of the same type

matrix – multiple columns of same length, all must have the same type of data

data.frame – multiple columns of same length, can be mix of data types

list – a collection of other objects; each item in the list can be a separate type of object

function – a command in R

Naming Objects

- In R, use “<-” to create objects; what’s on the left side is the object name and can be *almost* anything.

```
x <- 4
```

- Object names can consist of letters, numbers, periods* and underscores.
  - Cannot start with a number; best to start with letter.
  - e.g., x, mydata, mydata_normalized, TrtRep2
- Check to make sure desired object name is not already a function

```
?objectname
```

*best practice is to not use . because it means something very different in R

How to use functions

- Functions are indicated by parentheses – ()

```
sqrt(81)
```

- “Arguments” are the input to functions within () and are separated by **commas**

```
ls()                    0 arguments
rm(myobject)            1 argument
cbind(x1, x1 + 1) 2 arguments
```

- Most functions have > 1 argument; input can either be listed in order, or associated by name.

```
write.table(object, “outputname.txt”, FALSE)
write.table(object, append = FALSE, file = “outputname.txt”)
```

Functions for Exploring Objects

str() – overall structure of the object

class() – gives the "class" of the object

length() – gives the number of entries in 1D objects (vector, list)

dim(), nrow(), ncol() – gives number of rows/columns for 2D objects (matrix, data.frame)

names() – gives/sets names of list items or data.frame columns

rownames(), colnames() – gives/sets row & column names of a matrix or data.frame

How to use functions II

- R add-on packages - sets of functions that do particular things.

- ONCE only per R version: packages need to be **installed** (see the R Installation guide).

- EVERY time you open R: packages need to be **loaded**

  library(edgeR)

  - “Error in library(edgeR) : there is no package called ‘edgeR’ – package has not been installed yet

- "Error: could not find function "xxxx" " – package has probably not been loaded from library.

back to demo...

Subsetting objects

[and $ are the main *base R* ways to subset:

- use [ ] to subset 1D objects (vector, list)
- use [,] to subset 2D objects (matrix, data.frame)
    - rows first, then columns
- inside [ ] or [,] can be positions, names in quotes or TRUE/FALSE values.
    - Can also be used to re-order objects

- $ can pull out a named column from a data.frame or a named item in a list
- a $ must be followed by the name

Subsetting Lists

If list x is a train carrying objects, then:

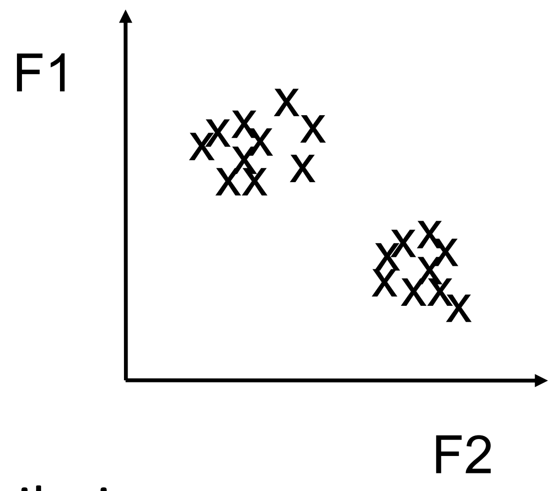
- x[4:6] is a train of cars 4-6
- x[5] is a train of just car 5

How to quit R

- Command line: q()
  - or in RStudio/Rgui, top right corner X of window
- Default prompt asking whether you want to save the workspace image
  - If pick "**Yes**", will save objects in workspace as unnamed .RData file and commands as unnamed .Rhistory in current working directory; DO NOT GET IN THE HABIT OF USING THIS!
  - If pick "**No**", will lose objects and codes unless you have saved them elsewhere; *despite risk, this is best for reproducible research*
  - If pick "**Cancel**", return to R

Goal of Clustering

- Given a set of data points, each described by a set of attributes, find clusters such that:
  - Inter-cluster similarity is maximized
  - Intra-cluster similarity is minimized

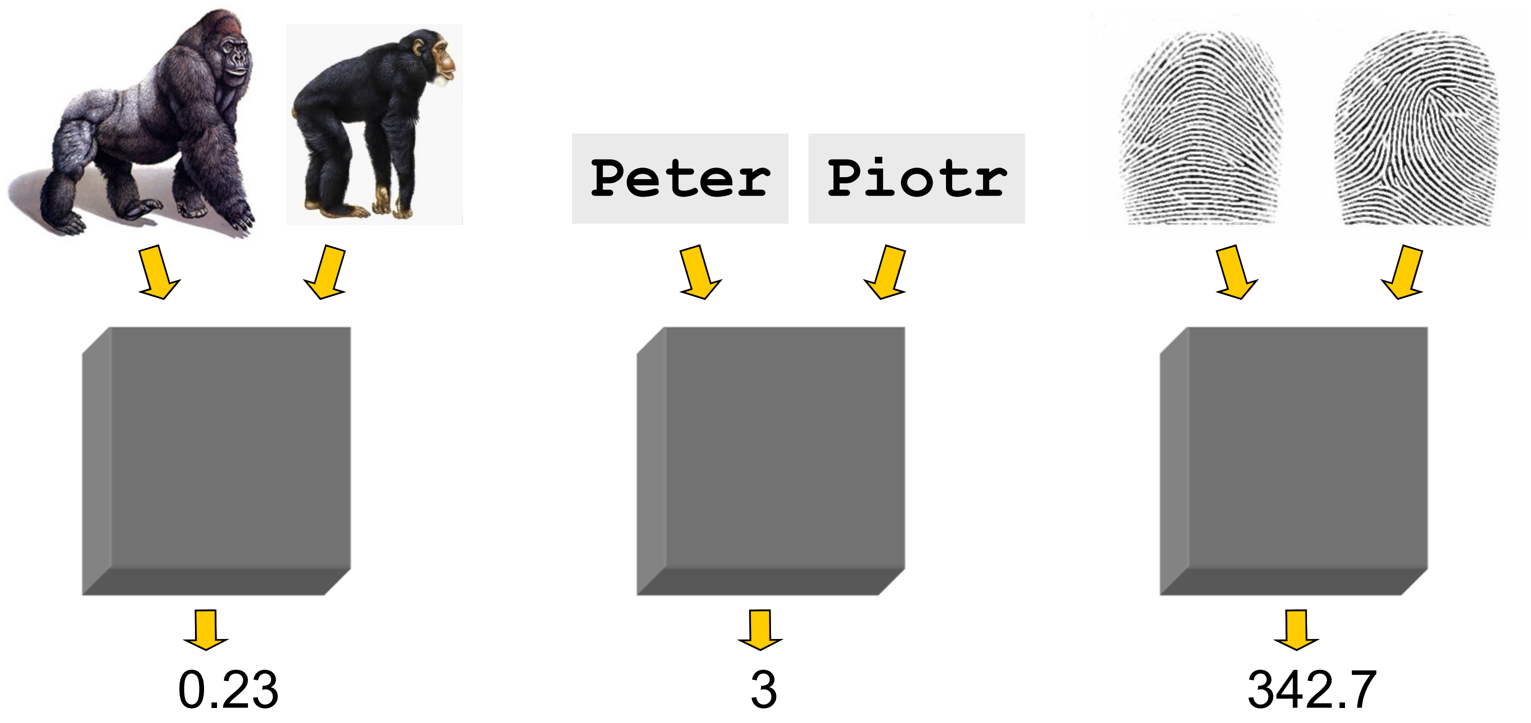


- Requires the definition of a similarity measure

Defining Distance Measures

Slide from Eamonn Keogh

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



Slide based on one by Eamonn Keogh

What properties should a distance measure have?

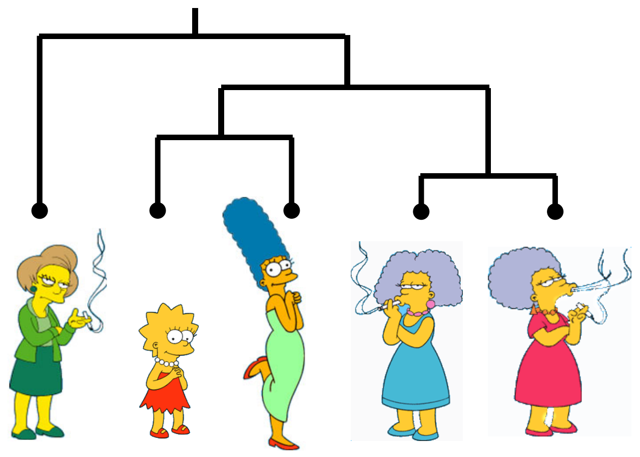
• $D(A,B) = D(B,A)$	*Symmetry*
• $D(A,A) = 0$	*Constancy of Self-Similarity*
• $D(A,B) = 0$ iif A= B	*Positivity (Separation)*
• $D(A,B) \leq D(A,C) + D(B,C)$	*Triangular Inequality*

Slide based on one by Eamonn Keogh

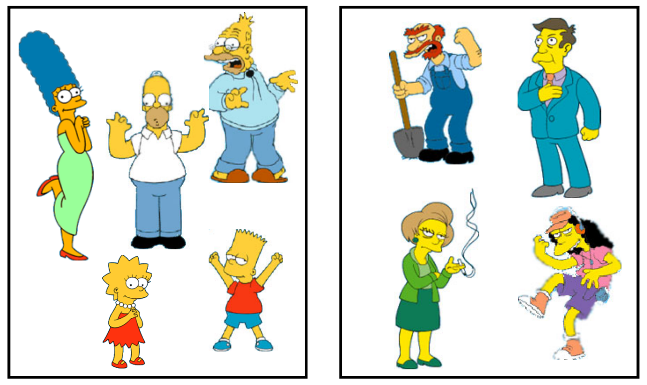
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

Hierarchical



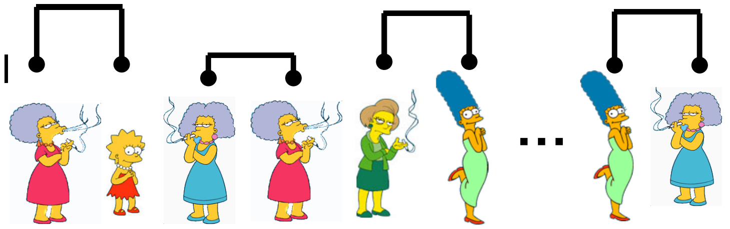
Partitional



Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

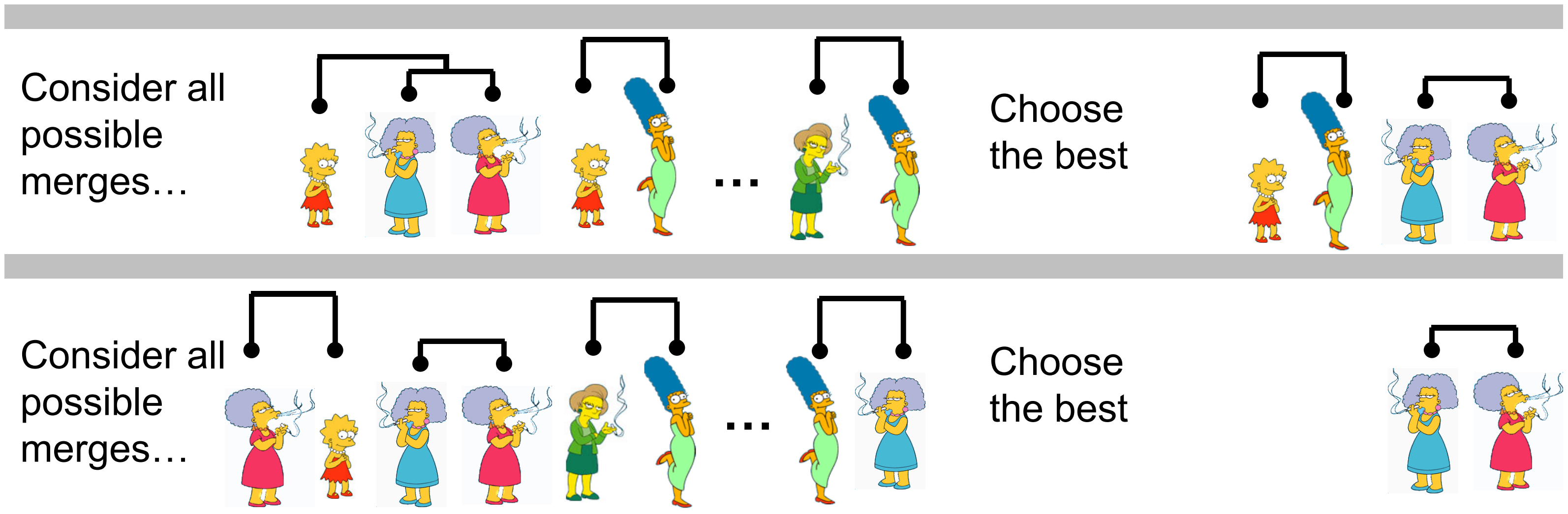
This slide and next 4 based on slides by Eamonn Keogh

Consider all possible merges…

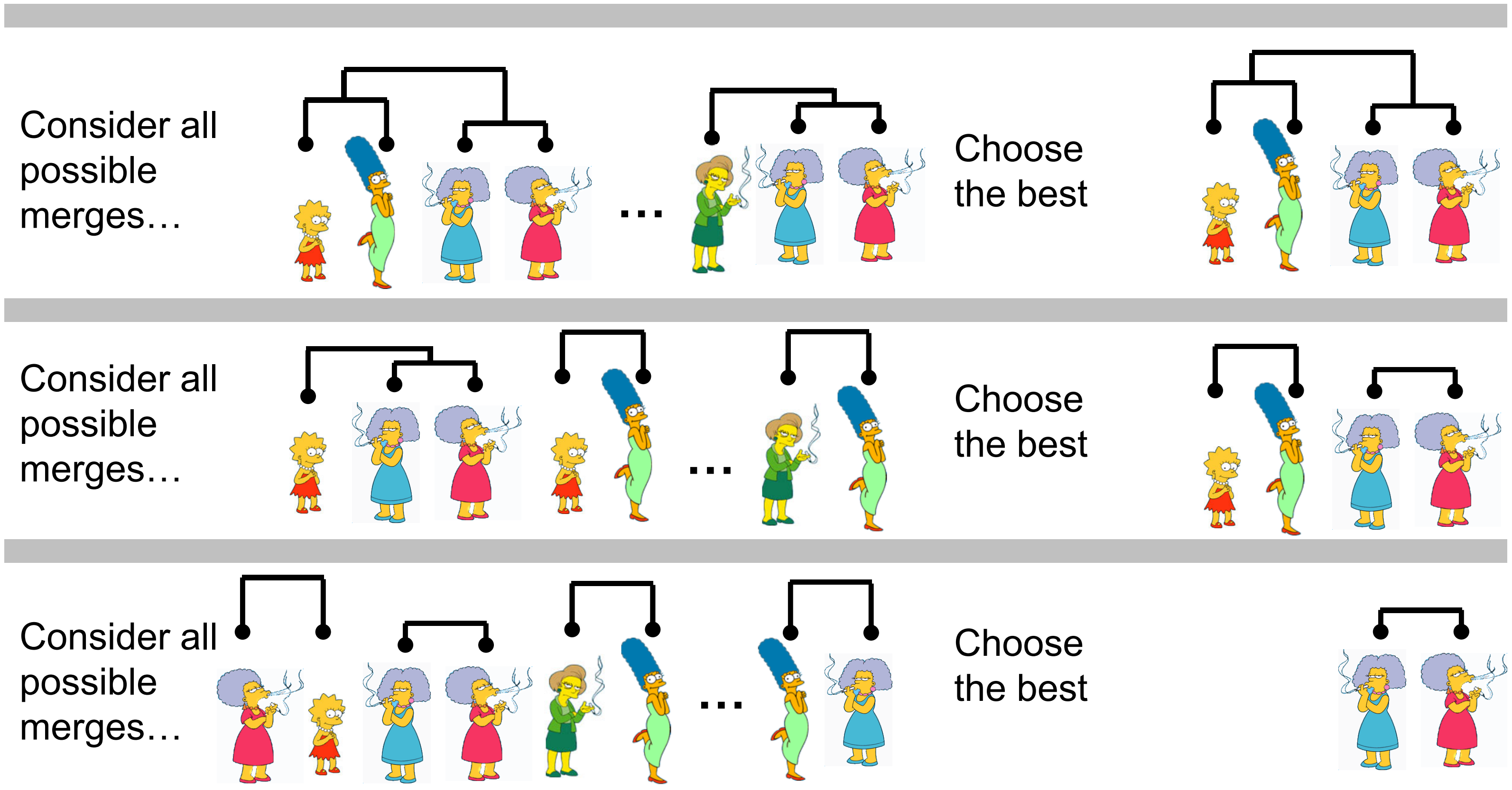


Choose the best

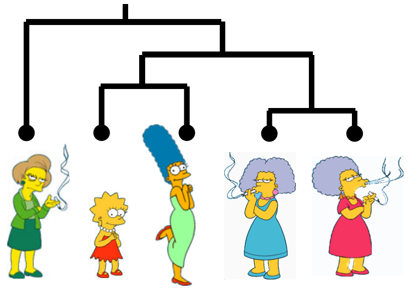
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



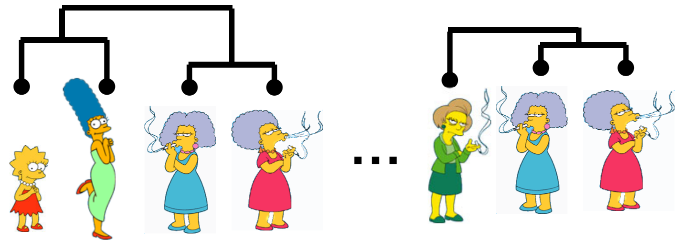
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.


Consider all possible merges…
…
Choose the best
Consider all possible merges…
…
Choose the best
Consider all possible merges…
…
Choose the best

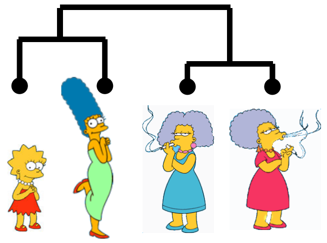
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



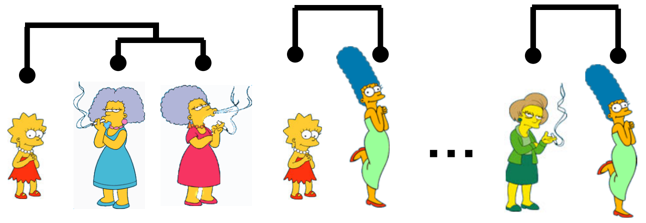
Consider all possible merges…



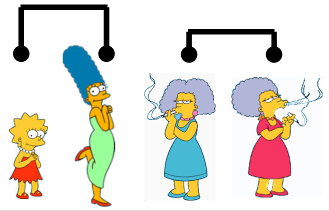
Choose the best



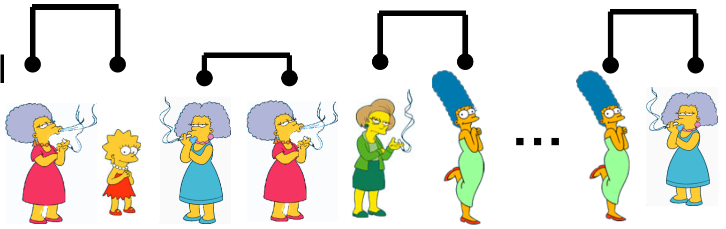
Consider all possible merges…



Choose the best



Consider all possible merges…



Choose the best

Slide based on one by Eamonn Keogh

We know how to measure the distance between two objects, but defining the distance between an object and a cluster, or defining the distance between two clusters is non obvious.

- **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").
- **Group average linkage:** In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.

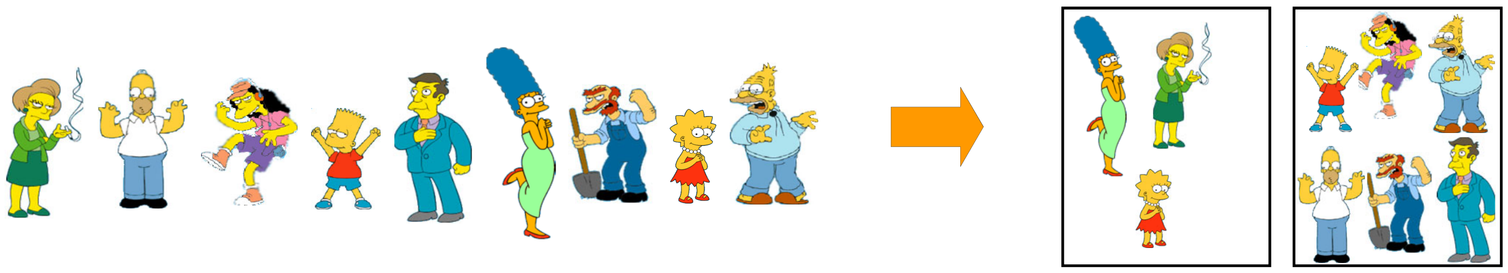
Hierarchical clustering

```
library(datasets)
head(mtcars)
cars <- mtcars[, c(1:4, 6:7, 9:11)]  # Select variables
?hclust
hc <- hclust(dist(cars), "ave")
?rect.hclust
rect.hclust(hc, k=2, border="blue")
rect.hclust(hc, k=4, border=“pink")
```

Slide based on one by Eamonn Keogh

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.



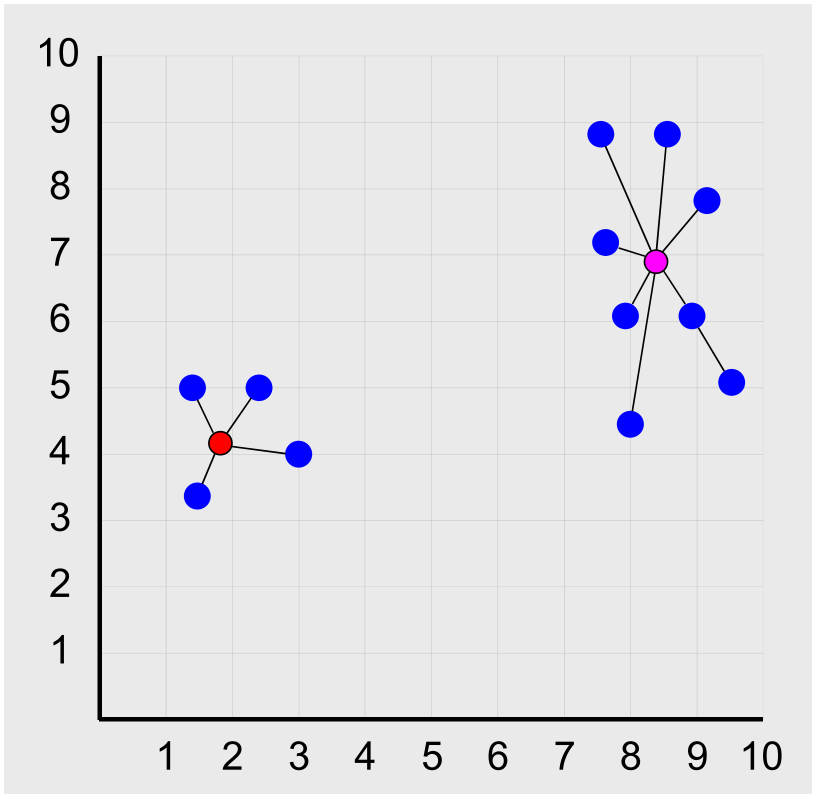
Slide based on one by Eamonn Keogh

Squared Error

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



Slide based on one by Eamonn Keogh

Partition Algorithm 1: k-means

1. Decide on a value for *k*.

2. Initialize the *k* cluster centers (randomly, if necessary).

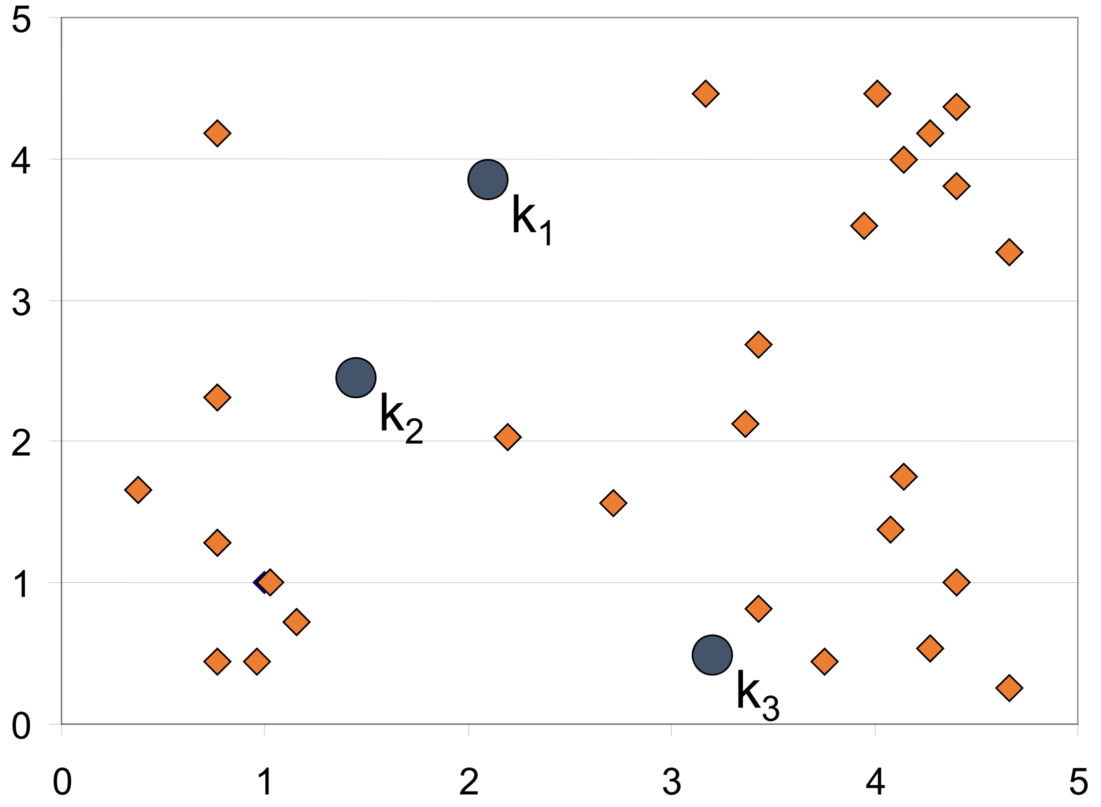
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

K-means Clustering: Step 1

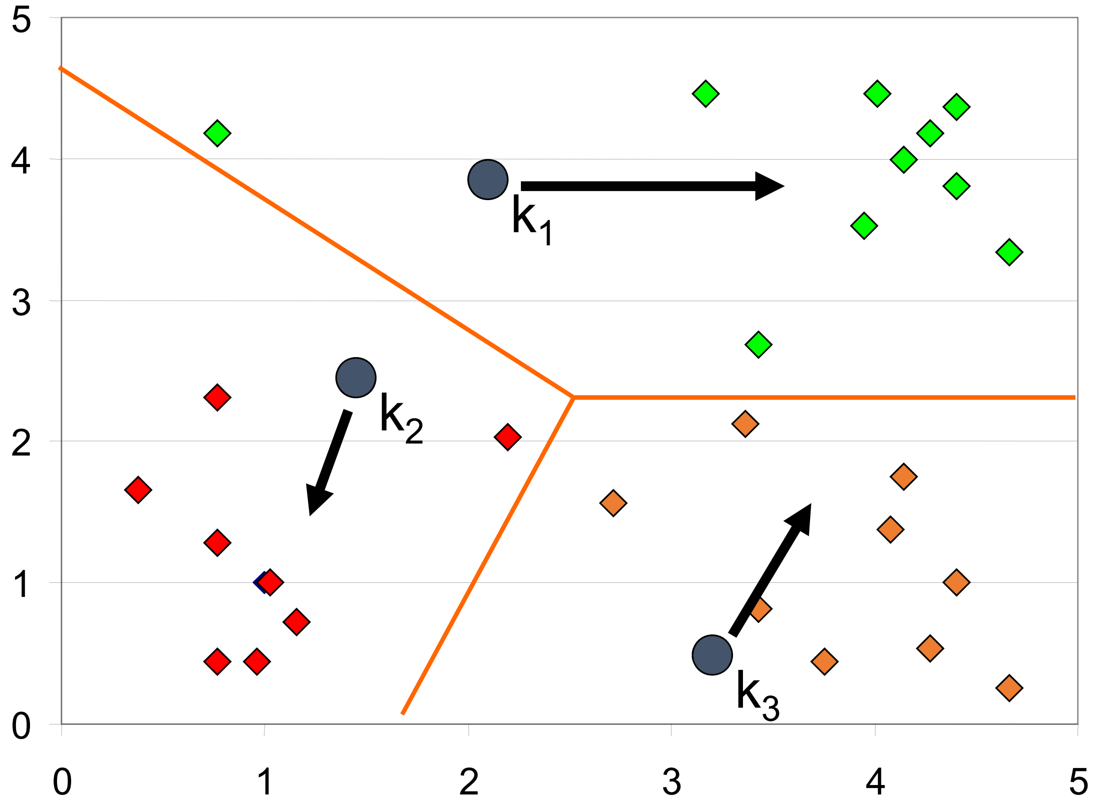
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 2

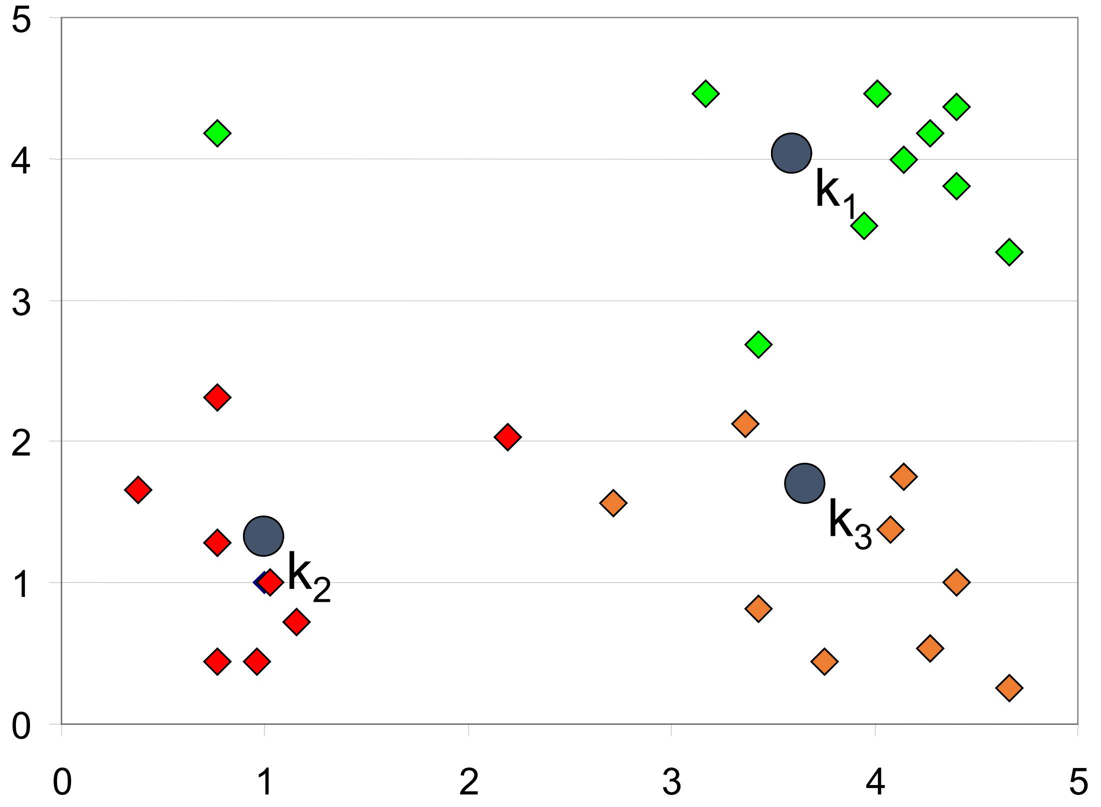
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 3

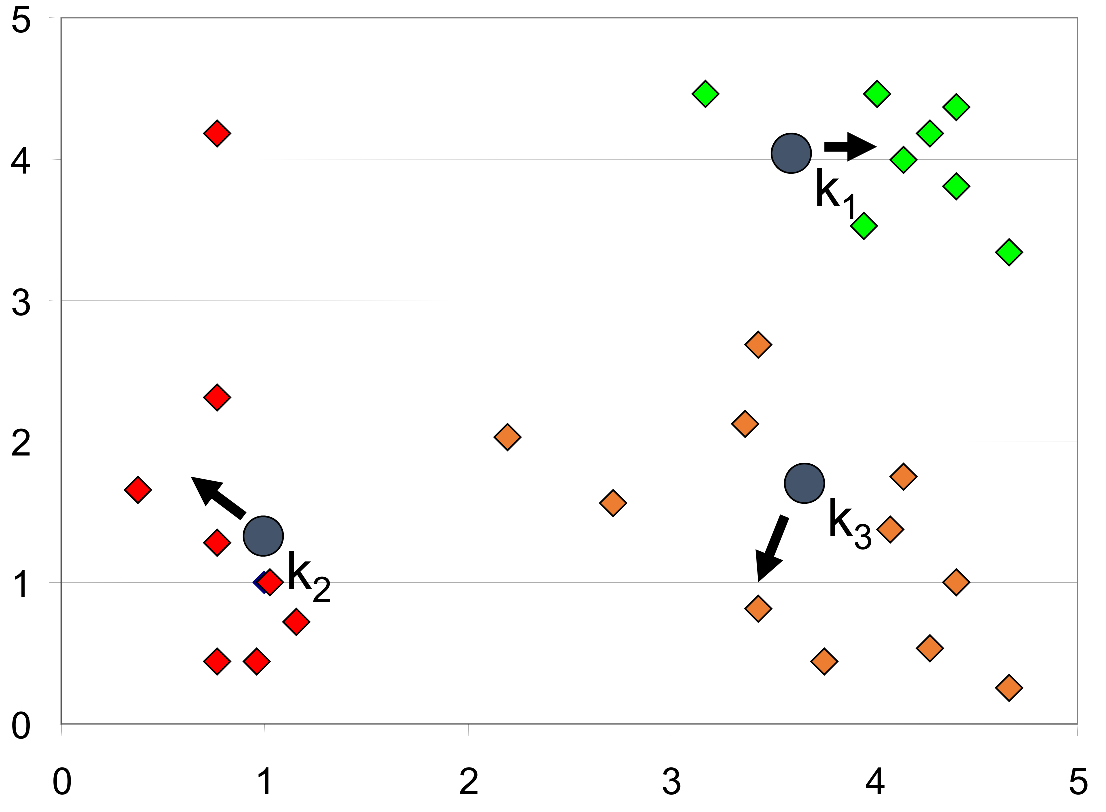
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 4

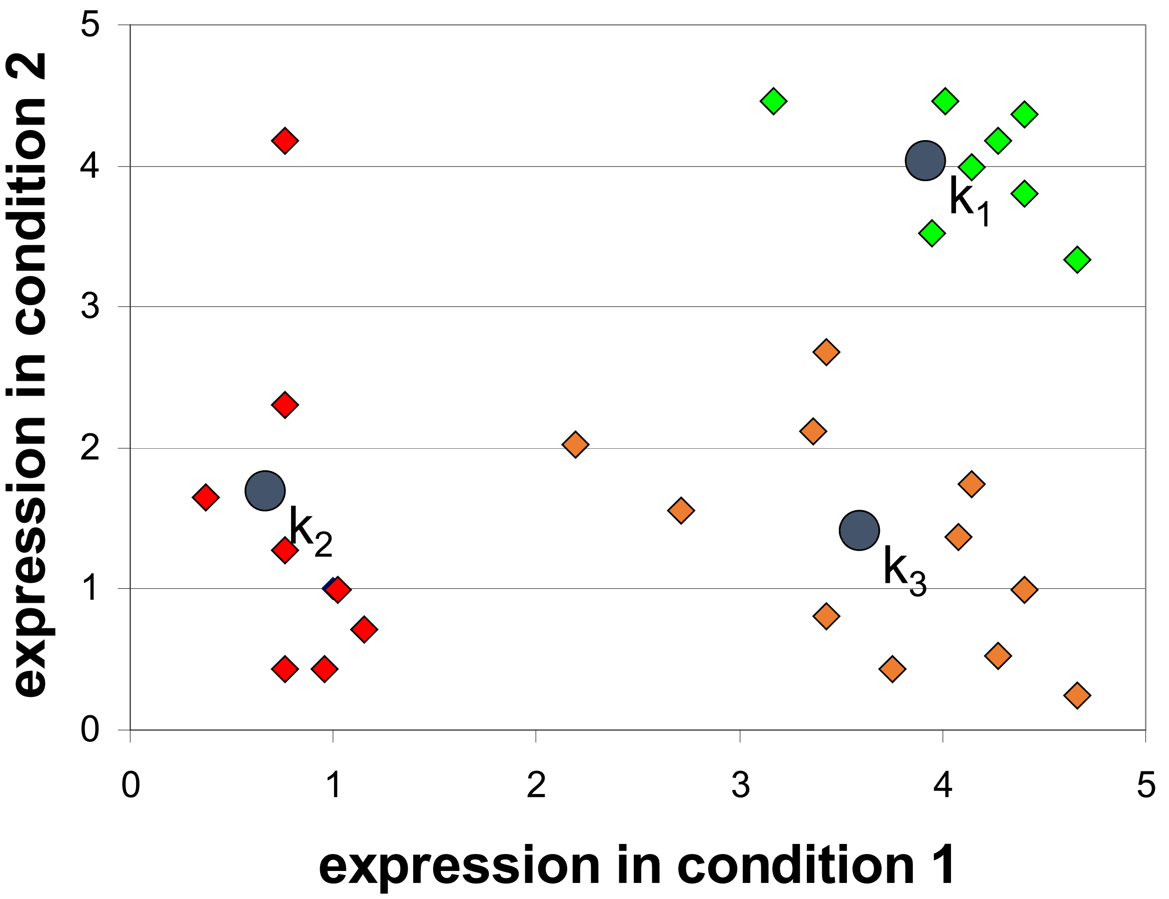
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

Comments on k-Means

- <u>Strengths</u>
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, t << n.
  - Often terminates at a local optimum.
- <u>Weakness</u>
  - Applicable only when mean is defined, then what about categorical data?
  - Need to specify *k,* the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

Slide based on one by Eamonn Keogh

K-means clustering

```
library(datasets)
head(mtcars)
cars <- mtcars[, c(1:4, 6:7, 9:11)]  # Select variables
?kmeans
par(mfrow=c(2,1))
hc <- kmeans(dist(cars), 3)
plot(dist(cars),col=hc$cluster)
hc <- kmeans(dist(cars), 5)
plot(dist(cars),col=hc$cluster)
```

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

   $$y = A(x - \mu)$$

   where y is the new point, x is the old one, and the rows of A are the eigenvectors

Why $y = A(x - \mu)$ (1)

Assume you have the gene expression data

(n genes and m experiments).

$$X = (x_{ij}), i = 1\ to\ m, \qquad j = 1\ to\ n$$

Normalize the gene expression so that the expression of each gene has mean 0 variance 1. Assume the normalized expression data is still X.

$$X'X$$

This product is actually calculate the covariance of the gene expression.

Why $y = A(x - \mu)$ (2)

$X'X$ is symmetric and have eigenvalues and eigenvectors.

$X'Xt = \lambda t$, where t is a vector of m dimensions, t is the eigenvector.

Rayleigh quotient

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

Why $y = A(x - \mu)$ (3)

Define $A = (t_1, t_2, \dots, t_n)$, where each t_i is the normalized eigenvector for the matrix $X'X$. We have $X'XA = \Lambda A$, where Λ is the diagonal matrix of the eigenvalues.

$$A' X'XA = A'\Lambda A = \Lambda A'A = \Lambda$$

Rayleigh quotient

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

A simple video on PCA concept

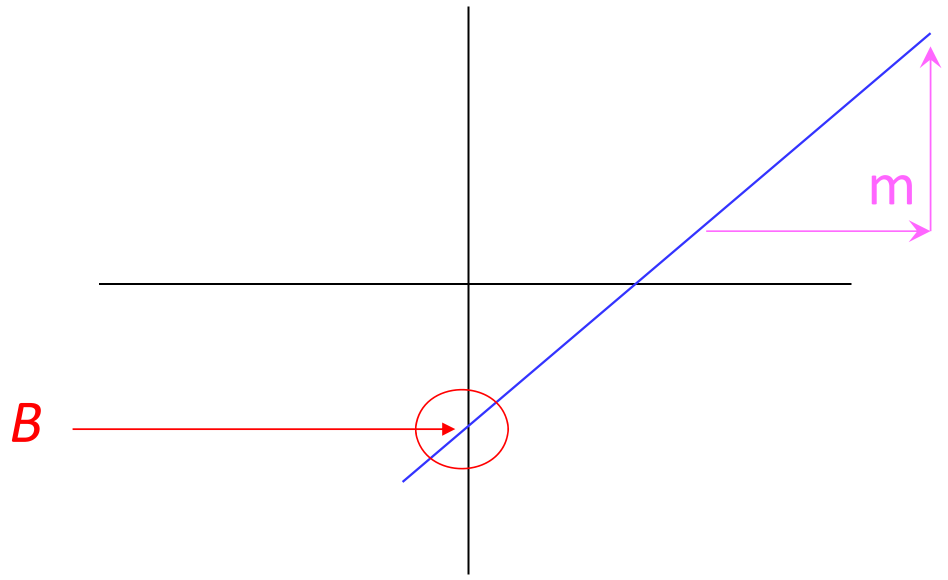
https://www.youtube.com/watch?v=HMOI_lkzW08

Principle Component Analysis

```
library(datasets)
head(mtcars)
cars <- mtcars[, c(1:4, 6:7, 9:11)]  # Select variables
pc<-prcomp(cars, center=T, scale=T)
summary(pc)
names(pc)
plot(pc)
biplot(pc)
```

What is “Linear”?

- Remember this:
- *Y=mX+B?*



What’s Slope?

A slope of 2 means that every 1-unit change in X yields a 2-unit change in Y.

Prediction

If you know something about X, this knowledge helps you predict something about Y. (Sound familiar?…sound like conditional probabilities?)

Regression equation…

Expected value of y at a given level of *x*=

$$\boxed{E(y_i / x_i) = \alpha + \beta x_i}$$

Predicted value for an individual…

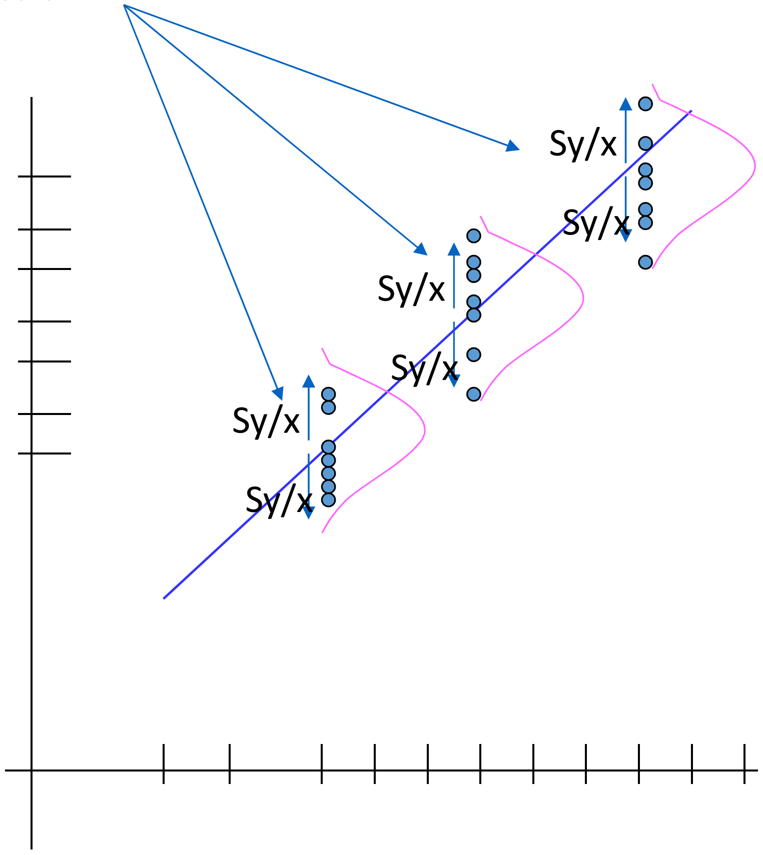
$$y_i = \underbrace{\alpha + \beta * x_i}_{\text{Fixed – exactly on the line}} + \boxed{\text{random error}_i}$$

Follows a normal distribution

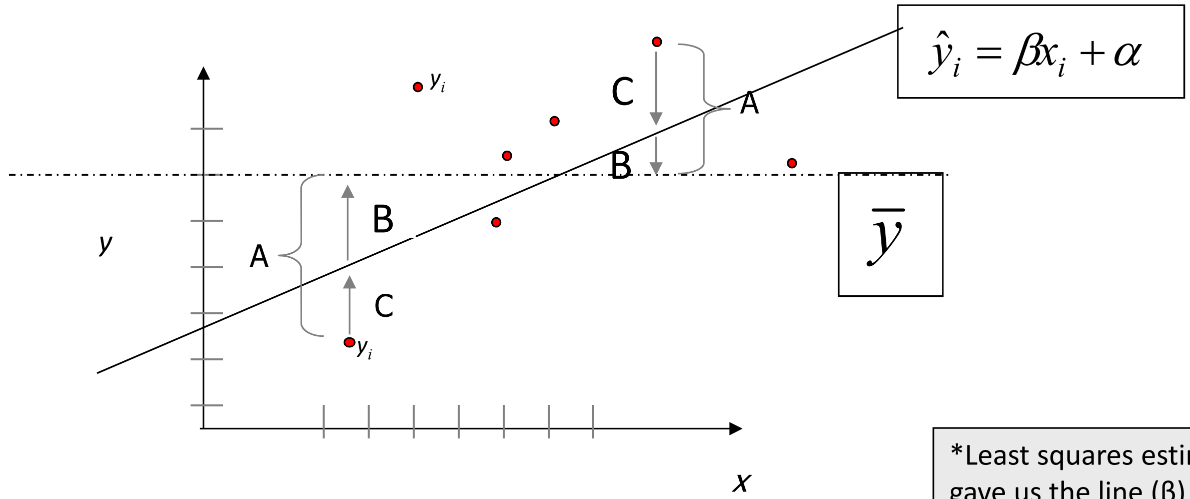
Assumptions (or the fine print)

- Linear regression assumes that…
  - 1. The relationship between X and Y is linear
  - 2. Y is distributed normally at each value of X
  - 3. The variance of Y at every value of X is the same (homogeneity of variances)
  - 4. The observations are independent

The standard error of Y given X is the average variability around the regression line at any given value of X. It is assumed to be equal at all values of X.



Regression Picture



*Least squares estimation gave us the line (β) that minimized C^2

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2 + \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

R^2=SSreg/SStotal

A^2

SS_{total}
Total squared distance of observations from naïve mean of y
Total variation

B^2

SS_{reg}
Distance from regression line to naïve mean of y
Variability due to x (regression)

C^2

$SS_{residual}$
Variance around the regression line
Additional variability not explained by x—what least squares method aims to minimize

Continuous outcome (means)

Outcome Variable	Are the observations independent or correlated?		Alternatives if the normality assumption is violated (and small sample size):
	independent	correlated	
Continuous (e.g. pain scale, cognitive function)	**Ttest:** compares means between two independent groups **ANOVA:** compares means between more than two independent groups **Pearson's correlation coefficient** (linear correlation): shows linear correlation between two continuous variables **Linear regression:** multivariate regression technique used when the outcome is continuous; gives slopes	**Paired ttest:** compares means between two related groups (e.g., the same subjects before and after) **Repeated-measures ANOVA:** compares changes over time in the means of two or more groups (repeated measurements) **Mixed models/GEE modeling**: multivariate regression techniques to compare changes over time between two or more groups; gives rate of change over time	Non-parametric statistics **Wilcoxon sign-rank test**: non-parametric alternative to the paired ttest **Wilcoxon sum-rank test** (=Mann-Whitney U test): non-parametric alternative to the ttest **Kruskal-Wallis test:** non-parametric alternative to ANOVA **Spearman rank correlation coefficient:** non-parametric alternative to Pearson's correlation coefficient

regression

```
library(datasets)
?USJudgeRatings
head(USJudgeRatings)
data<-USJudgeRatings
x<-as.matrix(data[-12])
y<-data[,12]
reg1<-lm(y~x)
reg1<-lm(RTEN~CONT+INTG+DMNR+DILG+CFMG+DECI+PREP+FAMI+ORAL+WRIT+PHYS,
data=USJudgeRatings)
summary(reg1)
names(reg1)
anova(reg1)
coef(reg1)
resid(reg1)
hist(residuals(reg1))
```

LASSO Regression

- Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.
- Together our regularized loss function is:
- Note that $\sum_{j=1}^{J} |\beta_j|$ is the ***l*₁** norm of the vector ***β***

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} |\beta_j|.$$

$$\sum_{j=1}^{J} |\beta_j| = \|\boldsymbol{\beta}\|_1$$

Choosing λ

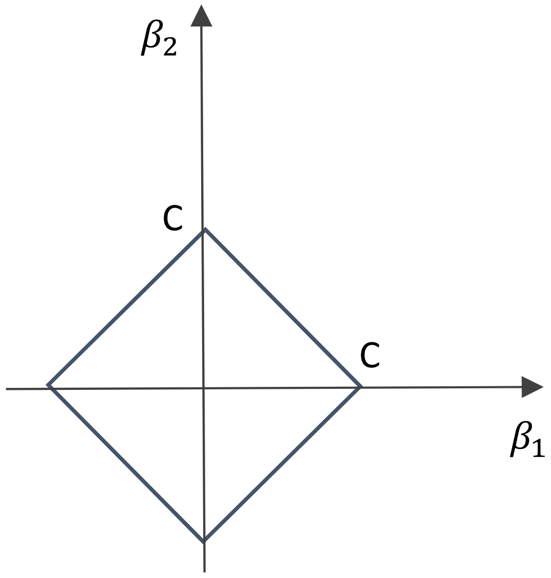
- In LASSO regression, we see that the larger our choice of the **regularization parameter** λ, the more heavily we penalize large values in $\boldsymbol{\beta}$,
- If λ is close to zero, we recover the MSE, i.e. LASSO regression is just ordinary regression.
- If λ is sufficiently large, the MSE term in the regularized loss function will be insignificant and the regularization term will force $\boldsymbol{\beta}_{\text{LASSO}}$ to be close to zero.
- To avoid ad-hoc choices, we should select λ using cross-validation.

The Geometry of Regularization (LASSO)

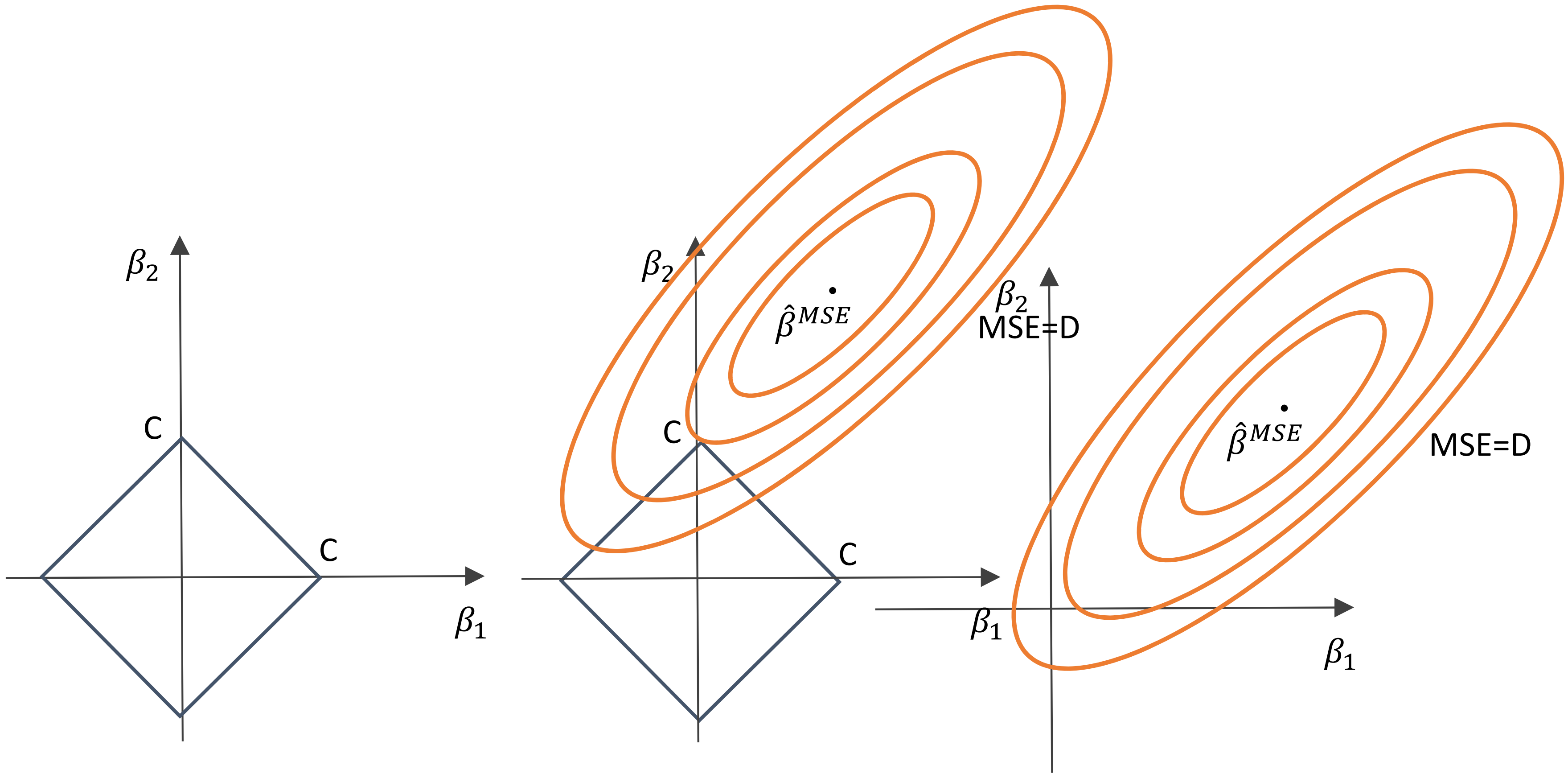
- $L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \text{argmin}\, L_{LASSO}(\boldsymbol{\beta})$$

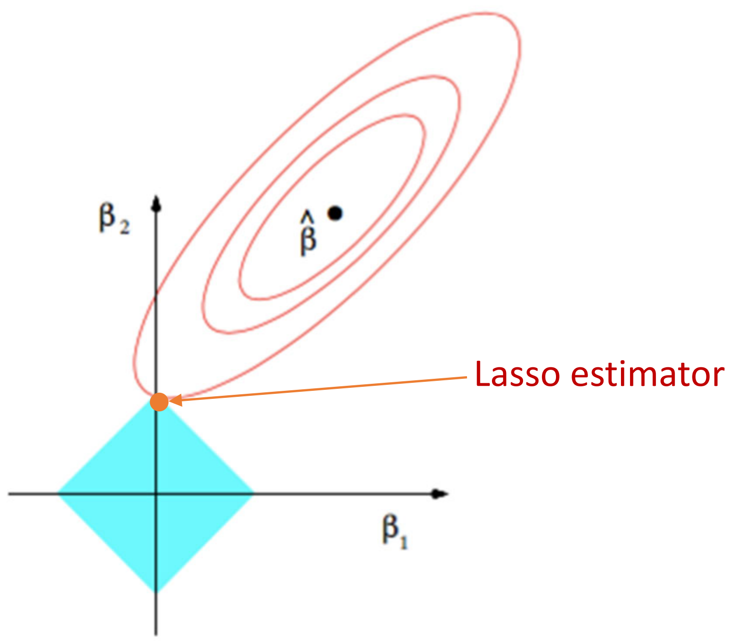
$$\lambda \sum_{j=1}^{J} | \hat{\beta}_j^{LASSO} | = C$$



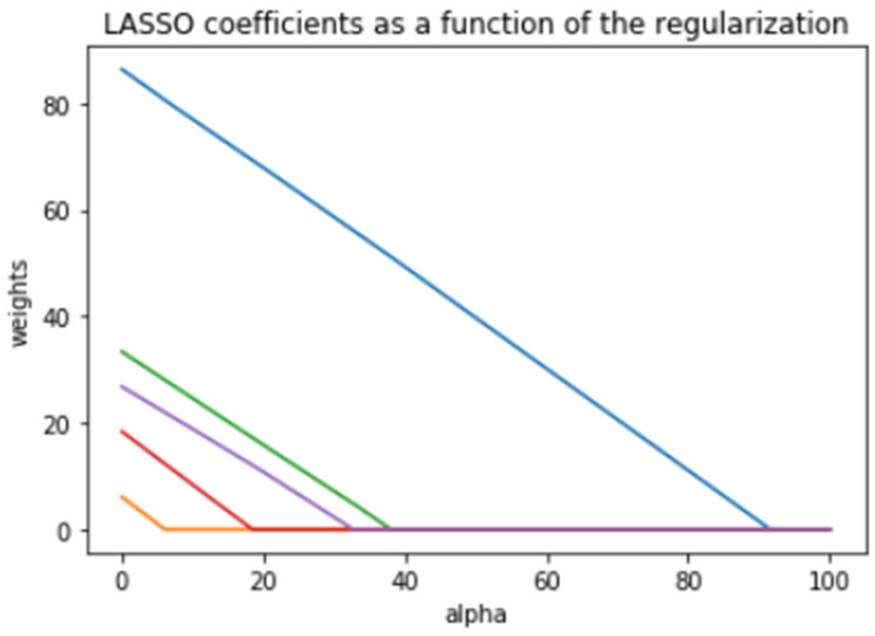
The Geometry of Regularization (LASSO)



LASSO visualized



The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axis.



The values of the coefficients decrease as lambda increases, and are nullified fast.

LASSO - Computational complexity

LASSO has no conventional analytical solution, as the L1 norm has no derivative at 0. We can, however, use the concept of subdifferential or subgradient to find a manageable expression. See a–sec2 for details.

Lasso regularization with **validation** only: step by step

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots \lambda_{max}\}$:
    A. determine the β that minimizes the L_{lasso}, $\hat{\beta}_{lasso}(\lambda)$, using the train data. **This is done using a solver.**
    B. record $L_{MSE}(\lambda)$ using validation data
3. select the λ that minimizes the loss on the validation data, $\lambda_{lasso} = \operatorname{argmin}_{\lambda} L_{MSE}(\lambda)$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{lasso}(\lambda_{lasso})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{lasso}(\lambda_{lasso})$

regression

```
install.packages("pacman")
library(pacman)
p_load(lars,caret)
data<-USJudgeRatings
x<-as.matrix(data[-12])
y<-data[,12]
stepwise<-lars(x,y, type="stepwise")
forward<-lars(x,y, type=forward.stagewise")
lar<-lars(x,y,type="lar")
lasso<-lars(x,y,type="lasso")
Lasso$beta[,1]
rm(list=ls())
```

Lar: least Angel Regression;
lasso: least absolute shrinkage and selection operator

Other references

- https://www.youtube.com/watch?v=_V8eKsto3Ug
- https://www.youtube.com/watch?v=fAPCaue8UKQ