ELEMENTS OF DATA SCIENCE AND STATISTICAL LEARNING

SPRING 2017

Week 8

OUTLINE

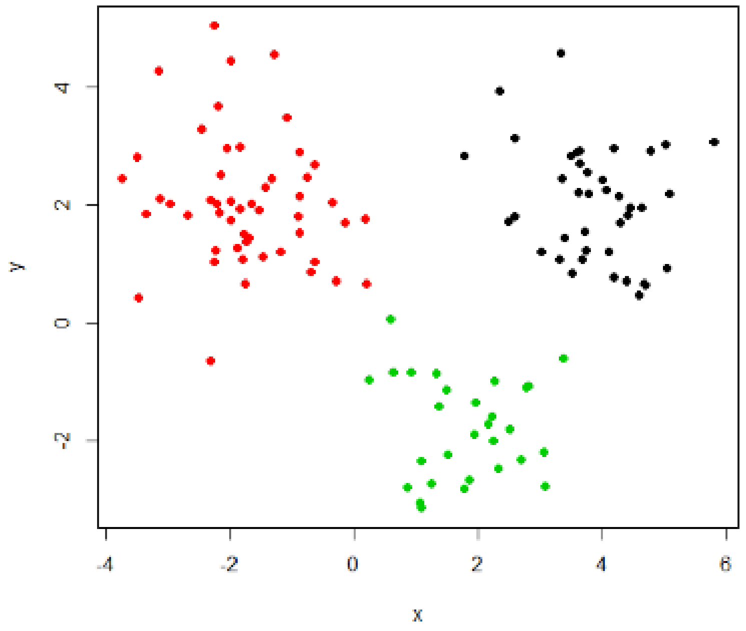
Unsupervised learning, continued

- How many clusters?
- Measures of "goodness of clustering"
- Examples

TEST DATASET

- Let us consider again our artificial example. We will use it to develop some intuition (and code)

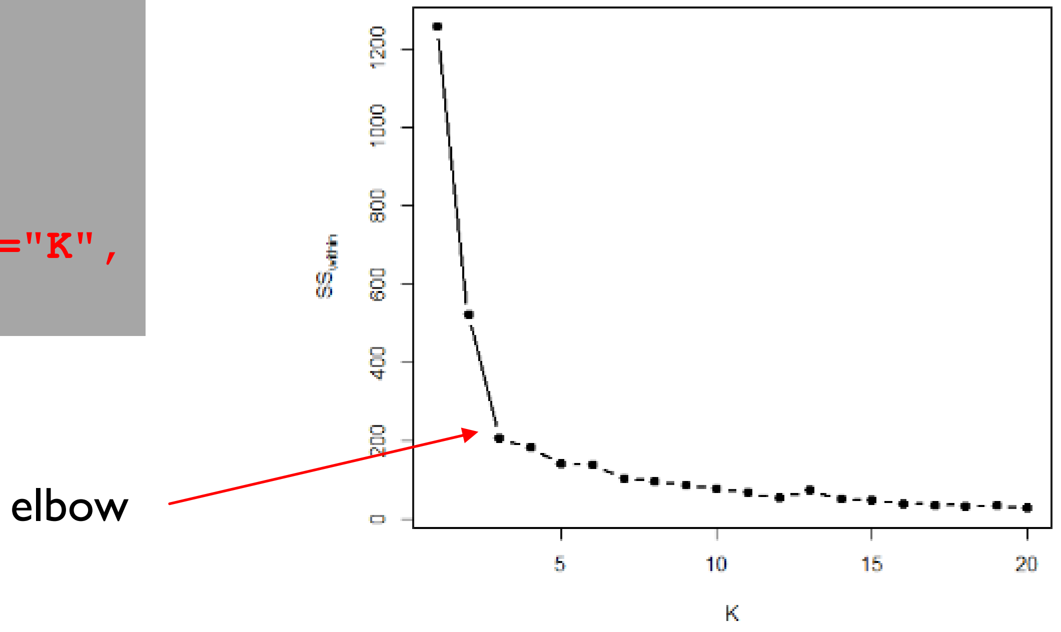
```
x=c(rnorm(30,mean=2),rnorm(50,mean=-2),rnorm(40,mean=4))
y=c(rnorm(30,mean=-2),rnorm(50,mean=2),rnorm(40,mean=2))
kf=kmeans(cbind(x,y),3)
plot(x,y,pch=19,col=kf$cluster)
l.cl=c(rep("A",30),rep("B",50),rep("C",40))
```



WITHIN-CLUSTER VARIATION

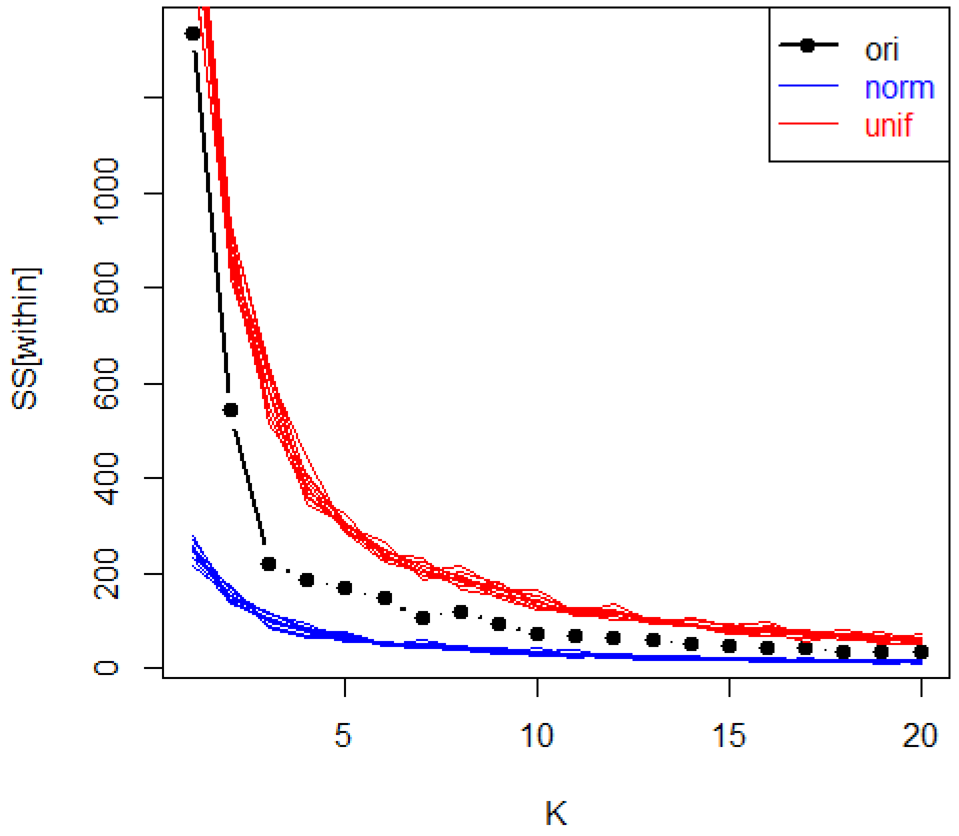
- Within-cluster variation $W = \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - \bar{x}_k\|_2^2 \leftarrow$ (sum of squares): $\boldsymbol{v} = (v_1, \dots, v_p) \Rightarrow \|\boldsymbol{v}\|_2 = \sqrt{\sum_{j=1}^{p} v_p^2}$
  - Where the cluster centers are defined as $\bar{x}_k = \frac{1}{n_k} \sum_{C(i)=k} x_i$
  - Data points x_i are considered here to be multi-dimensional , i.e. each point = a vector of observations for p variables

```
w=numeric(20)
for ( k in 1:20 ) {
  kf=kmeans(cbind(x,y),k)
  w[k] = kf$tot.withinss
}
plot(1:20,w,type="b",lwd=2,pch=19,xlab="K",
        ylab=expression("SS[within]"))
```

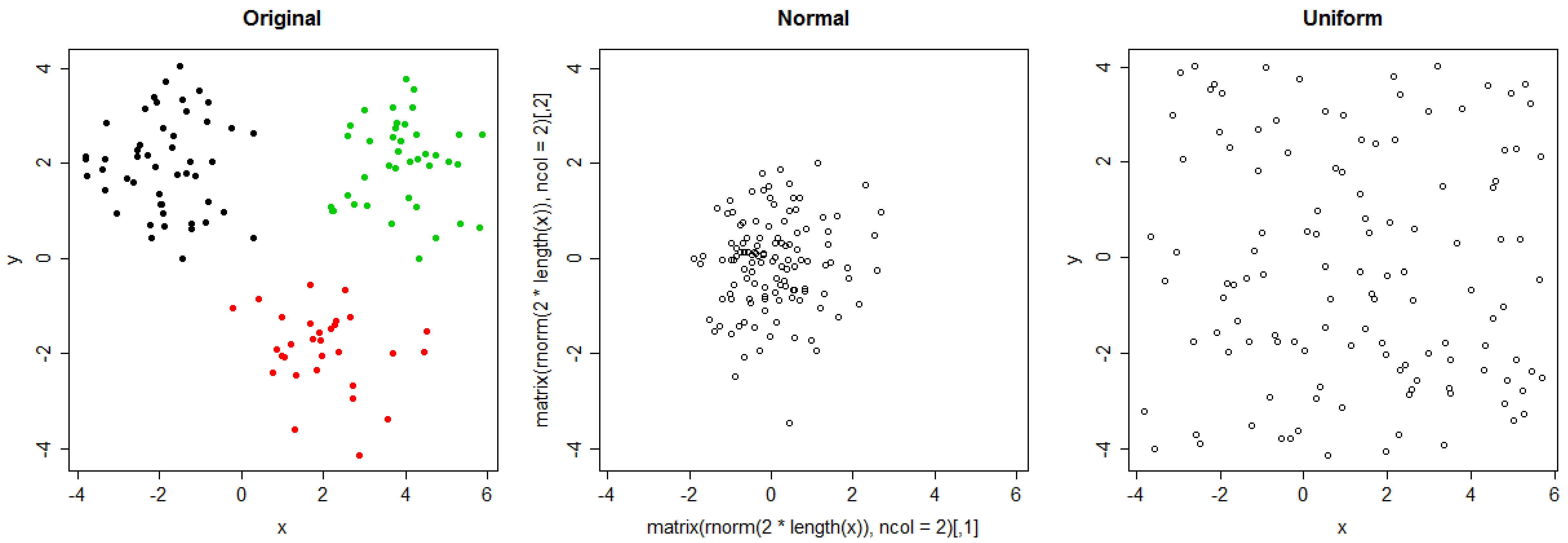


WITHIN CLUSTER VARIATION ON RANDOM DATA

- How striking is the elbow?
- Striking compared to what?
- For example, results of the same algorithm (K-means clustering) applied to data that is *known* to have *no clusters*
  - Random sample from normal or uniform distribution
- Of course, distribution we sample has to be roughly representative of the data we work with
  - E.g. standard normal has much lower variance than our simulated dataset
- Random sample of uniform distribution bounded by range of every attribute is commonly used alternative
- In this case total within cluster variance decreases faster for original data as compared to random sample



ORIGINAL, NORMAL AND UNIFORM EXAMPLES



- Sample from bivariate standard normal is obviously much tighter than that in the "original" data
- Uniform sample is more representative of "the same data without clusters"

CODE TO GENERATE PREVIOUS PLOTS

```
plot(1:20,w,type="b",lwd=2,pch=19,xlab="K",
        ylab=expression("SS[within]"))
for ( i in 1:10 ) {
    wrnd = numeric()
    for ( k in 1:20 ) {
        krnd =
kmeans(matrix(rnorm(2*length(x)),ncol=2),k)
        wrnd[k] = krnd$tot.withinss
    }
    points(wrnd,type="l",col="blue")
}
```

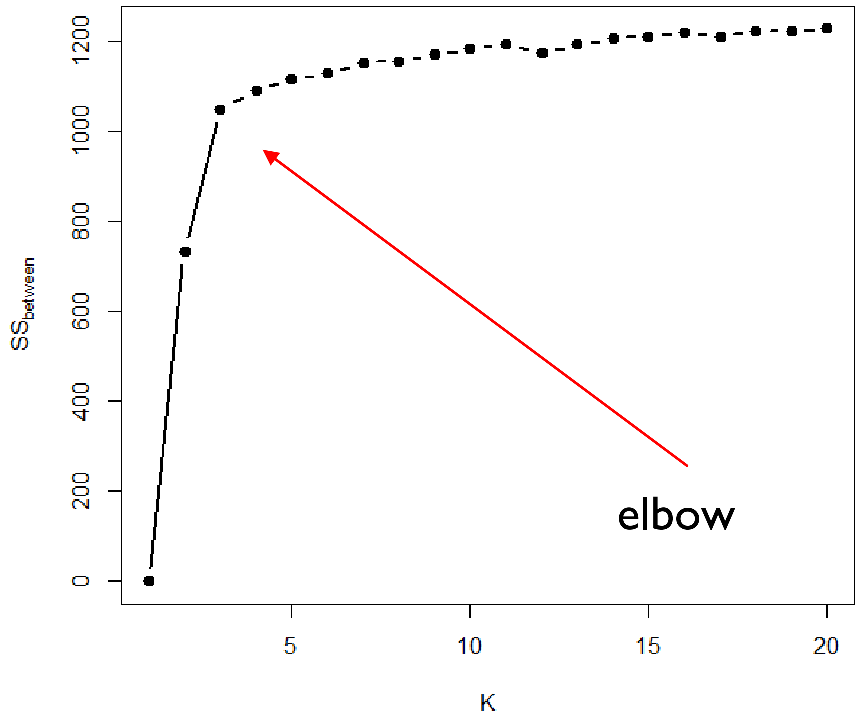
```
for ( i in 1:10 ) {
    wrnd = numeric()
    for ( k in 1:20 ) {
        krnd =
kmeans(apply(cbind(x,y),2,function(x)runif(leng
th(x),min(x),max(x))),k)
        wrnd[k] = krnd$tot.withinss
    }
    points(wrnd,type="l",col="red")
}

legend("topright",c("ori","norm","unif"),col=c(
"black","blue","red"),
            text.col=c("black","blue","red"),pc
h=c(19,NA,NA),lty=1,lwd=c(2,1,1))
```

```
old.par <- par(mfrow=c(1,3),ps=16)
plot(x,y,pch=19,col=kf$cluster,main="Original")
plot(matrix(rnorm(2*length(x)),ncol=2),xlim=range(x),ylim=range(y),main="Normal")
plot(apply(cbind(x,y),2,function(x)runif(length(x),min(x),max(x))),main="Uniform")
par(old.par)
```

BETWEEN-CLUSTER VARIATION

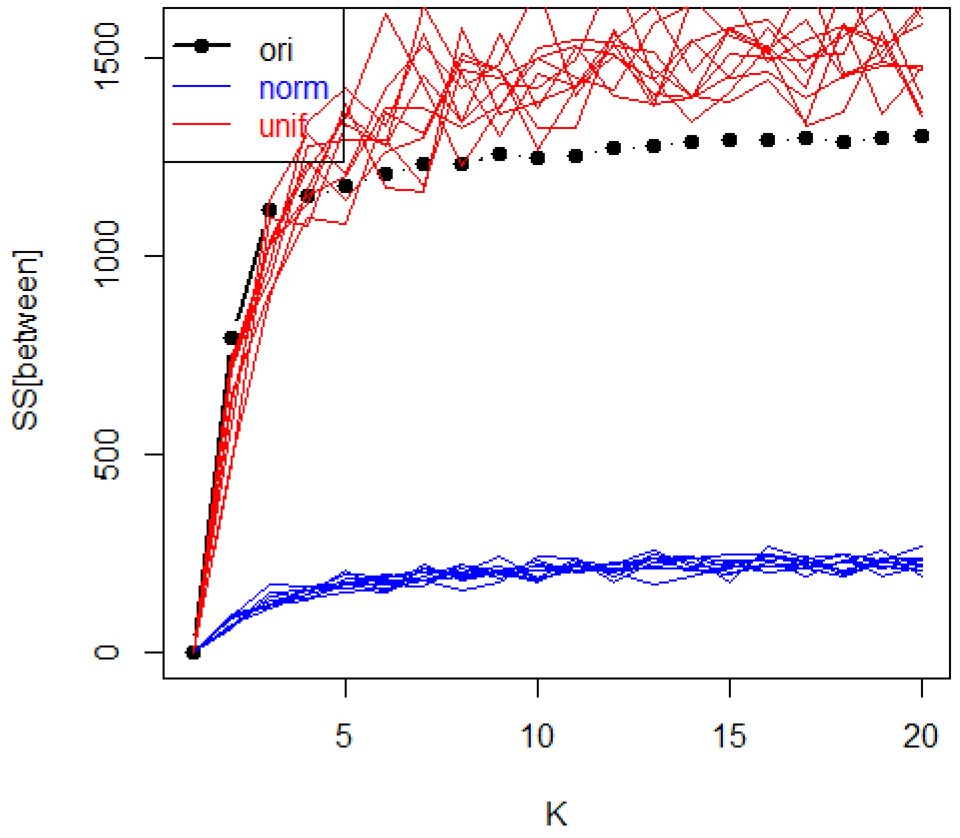
- Between-cluster variation, $B = \sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|_2^2$
- Note that the *total sum of squares* $T = \sum_{i=1}^{N} \|x_i - \bar{x}\|_2^2 = B + W$

```
w=numeric(20)
for ( k in 1:20 ) {
  kf=kmeans(cbind(x,y),k)
  w[k] = kf$betweenss
}
plot(1:20,w,type="b",lwd=2,pch=19,xlab="K",
    ylab=expression("SS[between]"))
```



BETWEEN-CLUSTER VARIATION ON RANDOM DATA

- We know now that standard normal is poor comparator for the data with three clusters
- For samples from uniform distribution between cluster sum of squares also increases rapidly for the first few clusters
- And even to higher levels on average than what is observed for higher values of K



CODE FOR THE PREVIOUS PLOT

```
plot(1:20,w,type="b",lwd=2,pch=19,xlab="K",
    ylab=expression("SS[between]"),ylim=rang
e(w)*c(1,1.2))


for ( i in 1:10 ) {
    btwrnd = numeric()
    for ( k in 1:20 ) {
        krnd =
kmeans(matrix(rnorm(2*length(x)),ncol=2),k)
        btwrnd[k] = krnd$betweenss
    }
    points(btwrnd,type="l",col="blue")
}
```

```
for ( i in 1:10 ) {
    btwrnd = numeric()
    for ( k in 1:20 ) {
        krnd =
kmeans(apply(cbind(x,y),2,function(x)runif(
length(x),min(x),max(x))),k)
        btwrnd[k] = krnd$betweenss
    }
    points(btwrnd,type="l",col="red")
}

legend("topleft",c("ori","norm","unif"),col
=c("black","blue","red"),
            text.col=c("black","blue","red"
),pch=c(19,NA,NA),lty=1,lwd=c(2,1,1))
```

CH INDEX

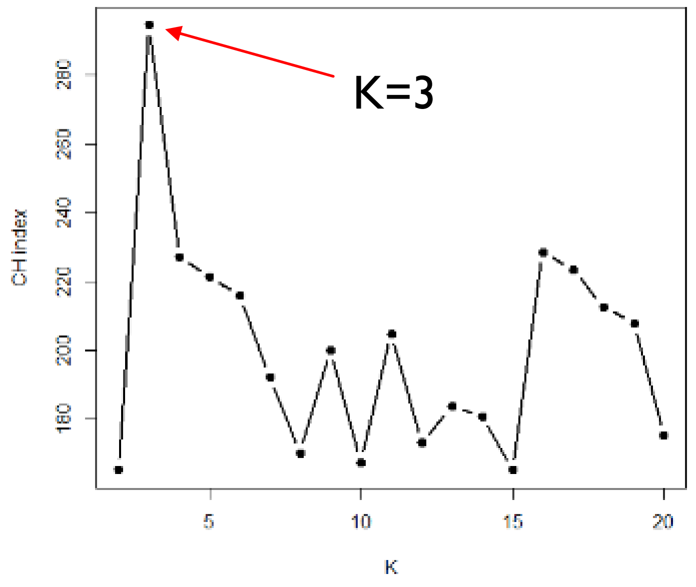
- Consider a score that attempts to strike a balance between within- and between-cluster sums of squares:

$$\mathrm{CH}(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

compare this to F-statistic (e.g. for linear regression it's Eq(3.23)in ISLR): $F = \frac{(TSS-RSS)/p}{RSS/(n-p-1)}$

- Called CH index [Calinski & Harabasz (1974), "A dendrite method for cluster analysis"]
- Calculate in a (reasonable) range of K values, pick K that maximizes the index:

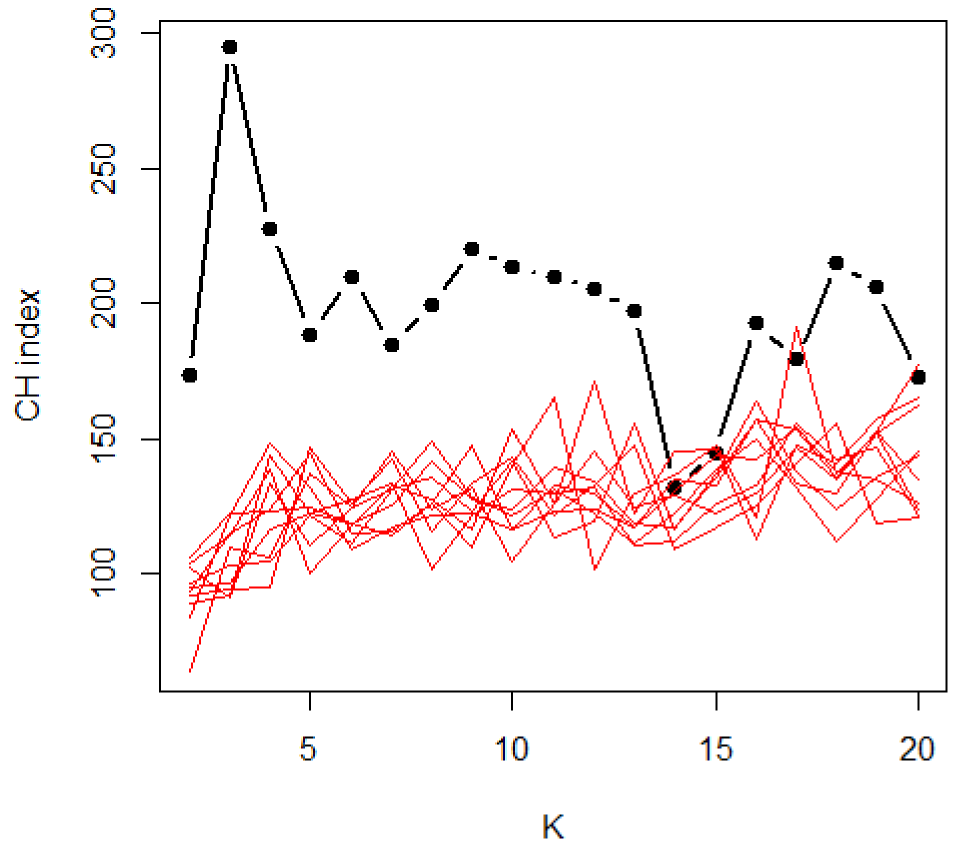
```
w=numeric(20)
for ( k in 2:20 ) {
  kf=kmeans(cbind(x,y),k)
  w[k] = (kf$betweenss/(k-1))/
         (kf$tot.withinss/(length(x)-k))
}
plot(2:20,w[-1],type="b", lwd=2,pch=19,xlab="K",
       ylab="CH index")
```



CH-INDEX ON RANDOM DATA

- CH-index achieves much higher values on original as compared to random data from uniform distribution with the same ranges
- Tends to gradually increase with the increase of K
- Value of K corresponding to the maximum in CH-index does not imply that this is the number of "real" clusters

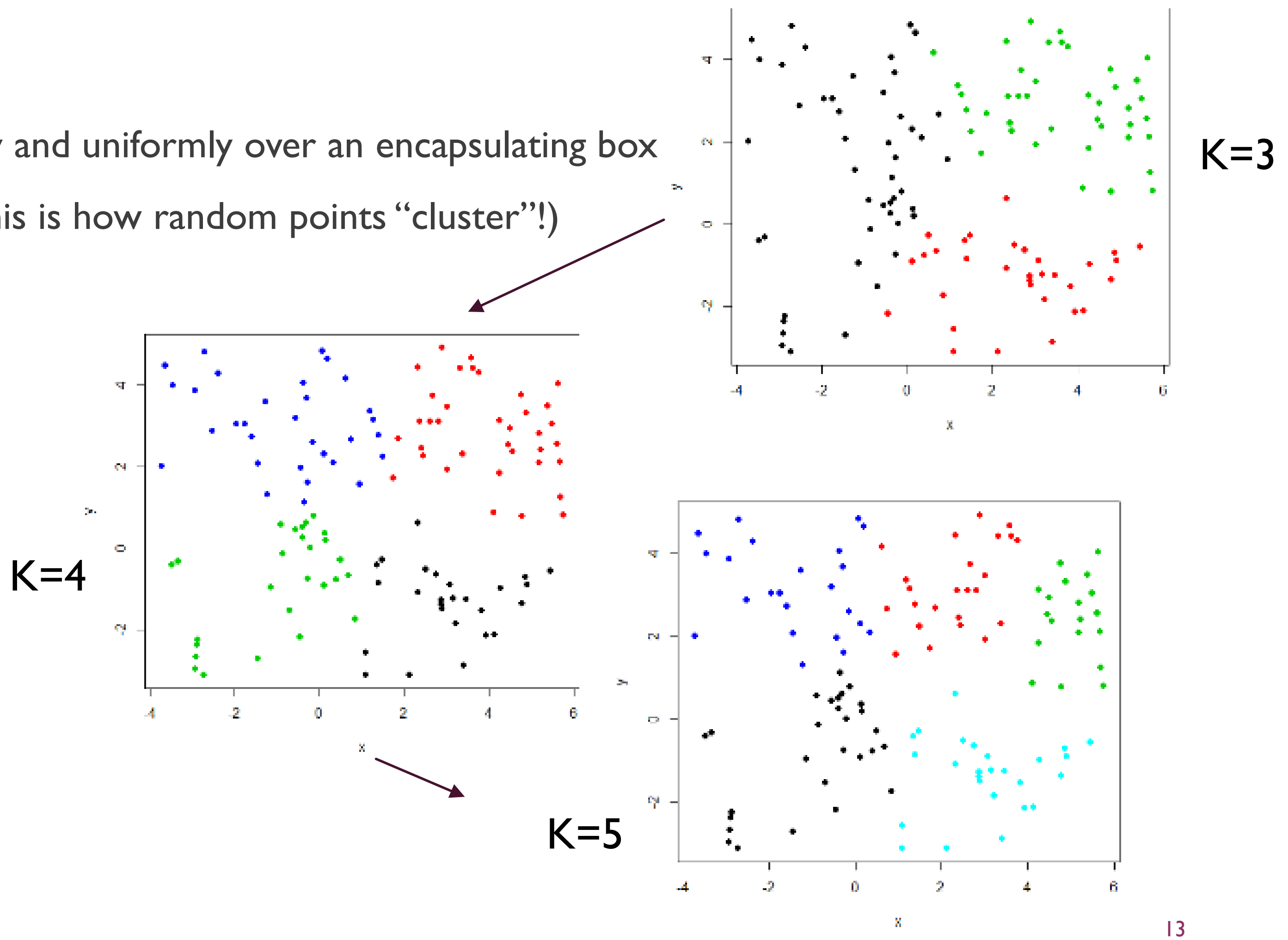
```
for ( i in 1:10 ) {
  chrnd = numeric()
  for ( k in 1:20 ) {
    krnd =
kmeans(apply(cbind(x,y),2,function(x)runif(len
gth(x),min(x),max(x))),k)
    chrnd[k] = (krnd$betweenss/(k-1))/
     (krnd$tot.withinss/(length(x)-k))
  }
  points(chrnd,type="l",col="red")
}
```



GAP STATISTICS

- Idea: compare against the "null distribution"
  - What is "null"?
  - No clusters = the points are distributed randomly and uniformly over an encapsulating box
  - Let's try it. Are those real clusters? (remember, this is how random points "cluster"!)

```
m.new=apply(cbind(x,y),2,function(x) {
     runif(length(x),min=min(x),max=max(x))
})
m.new[1:3,]
             x        y
[1,] 0.3242968 2.114528
[2,] 4.8677956 3.327992
[3,] 3.6180172 4.437447
kf=kmeans(m.new,3)
plot(m.new,pch=19,col=kf$cluster)
kf=kmeans(m.new,4)
plot(m.new,pch=19,col=kf$cluster)
kf=kmeans(m.new,5)
plot(m.new,pch=19,col=kf$cluster)
```



GAP STATISTICS: CONTINUED

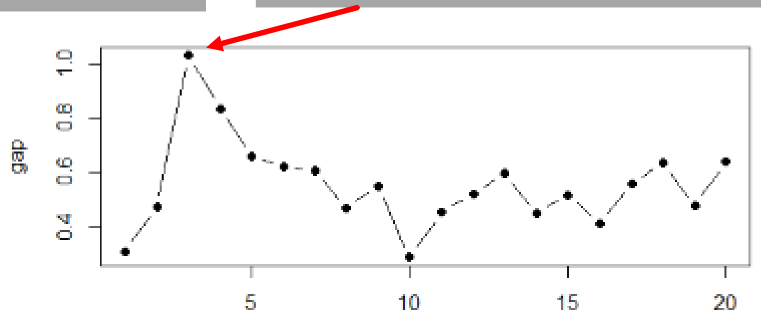
- Let us use the ratio of the "null" within-cluster sum of squares $W_{unif}(K)$ (computed from uniformly scattered points) to the actual within-cluster SS, W(K) (computed on real data) as the measure of "goodness of clustering"
  - With no cluster structure we expect the clusters to be "bad" and wide ➔ large W_{unif} (but still decreases with K!)
  - If there are true clusters in the data, we expect them to be "tighter" than those "seen" in uniform random data, so $W_{unif}(K)/W(K)$ is large.
  - Since we are computing a quantity (W_{unif}) from random data, it is a good idea to perform a few resamplings and take the average, so below we consider W_{unif} to be such average, with standard error s(K)
  - It is convenient to take log: $\text{Gap}(K) = \log W_{unif}(K) - \log W(K)$
  - We define the optimal K as the value where gap reaches its first maximum:
    - $K_{\text{opt}} = \min\{K \in \{1 \ldots K_{\max}\}: \text{Gap}(K) \geq \text{Gap}(K+1) - s(K+1)\}$

GAP STATISTICS: CODE

- Let's develop the code for gap statistics:

```
# takes matrix of observations of p variables (points in
# p-dimensional space: row=point), generates the
# same number of p-dimensional points randomly and
# uniformly scattered in the bounding box:
lw.unif=function(m,K,N=20) {
  w=numeric(N)
  for ( i in 1:N ) {
    m.new=apply(m,2,function(x) {
       runif(length(x),min=min(x),max=max(x))
     })
    kf=kmeans(m.new,K)
    w[i] = kf$tot.withinss
  }
  return( list(LW=mean(log(w)),SE=sd(log(w))/sqrt(N)) )
}
```

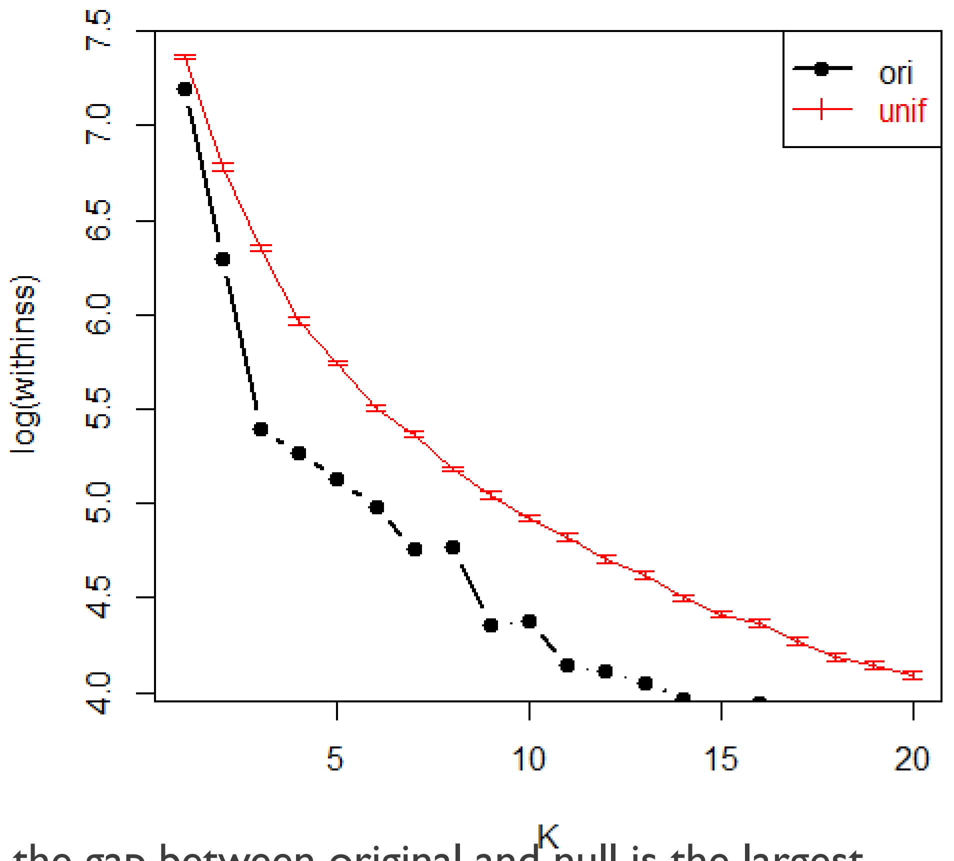
```
# computes the gap and the LogW_unif SE
# for different K:
gap=numeric(20)
se = numeric(20)
for ( k in 1:20 ) {
  kf=kmeans(cbind(x,y),k)
  sim = lw.unif(cbind(x,y),k)
  gap[k] = sim$LW - log(kf$tot.withinss)
  se[k] = sim$SE
}
plot(1:20,gap,pch=19,type="b")
# find optimal K:
min(which(gap[-length(gap)]>=
      (gap-se)[-1] ) )
[1] 3
```



Note: gap statistics is implemented in packages lga ang SAGx

ORIGINAL AND NULL WITHIN-CLUSTER SUM OF SQUARES

```
se = numeric(20)
rndlw = numeric(20)
orilw = numeric(20)
for ( k in 1:20 ) {
  kf=kmeans(cbind(x,y),k)
  sim = lw.unif(cbind(x,y),k)
  rndlw[k] = sim$LW
  orilw[k] = log(kf$tot.withinss)
  se[k] = sim$SE
}
plot(1:20,rndlw,type="l",lwd=1,col="red",xlab="K"
,ylab="log(withinss)")
arrows(1:20, rndlw-se, 1:20, rndlw+se,
length=0.05, angle=90, code=3, col=2)
points(orilw,type="b",pch=19,lwd=2)
legend("topright",c("ori","unif"),text.col=1:2,co
l=1:2,pch=c(19,3),lty=1,lwd=c(2,1))
```

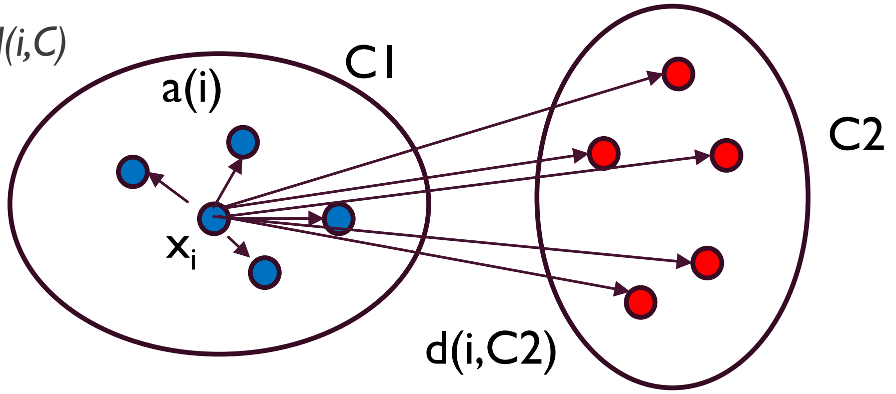


- According to gap statistics, the "best" number of clusters is where the gap between original and null is the largest

SILHOUETTE

- Consider a point x_i. *Silhouette* is defined for each individual point in the following way:
  - Let a(i) be the average dissimilarity (i.e. distance) between x_i and all other points of the cluster *C(i)* to which x_i belongs
  - If x_i is in its own cluster (with no other points in it), then the silhouette is *s(i)=0*; otherwise
  - For each other cluster *C* (i,e. excluding *C(i)*) let *d(i,C)* be the average distance from x_i to all observations in C
  - Define the "distance to the closest cluster" as $b(i)=\min_C d(i,C)$
  - Define silhouette as

$$\boxed{s(i) = \frac{b(i)-a(i)}{\max(\,a(i),b(i))}}$$

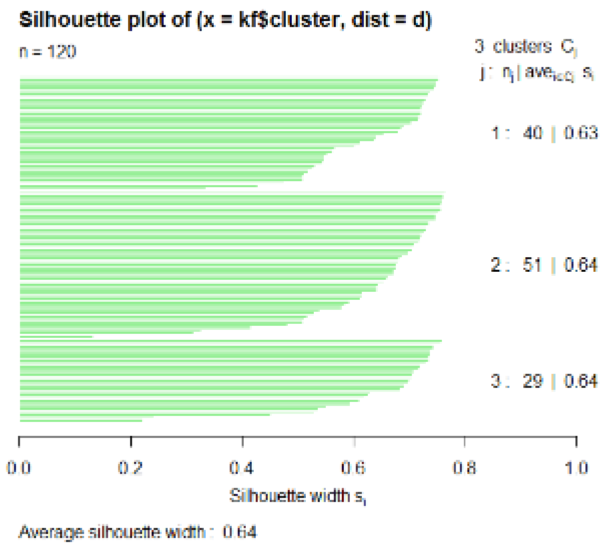


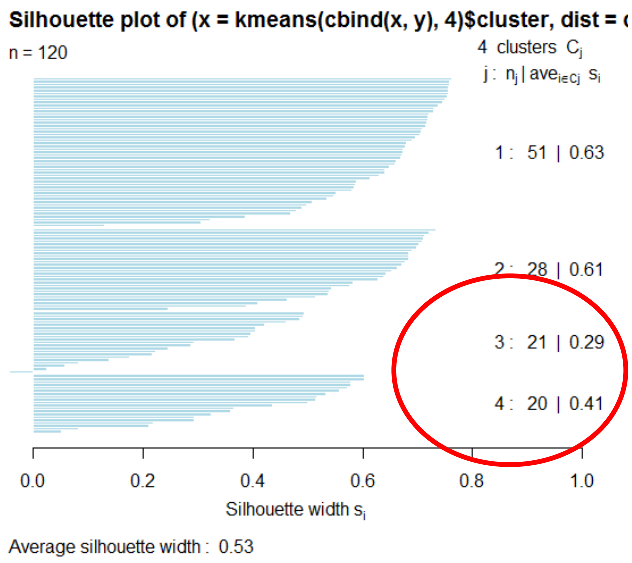
- Properties:
  - Normalized!
  - If "average width" a(i) is small and the distance to the closest cluster b(i) is large, then s(i) → 1
  - If s(i) is small, the point is "between clusters" (or in its own cluster): the distance to the other members of the same cluster is about the same as to other clusters
  - If s(i) is negative (a(i) > b(i)), the point is seriously misclassified: it's closer to another cluster than to other members of the cluster it was assigned to

SILHOUETTE: R CODE

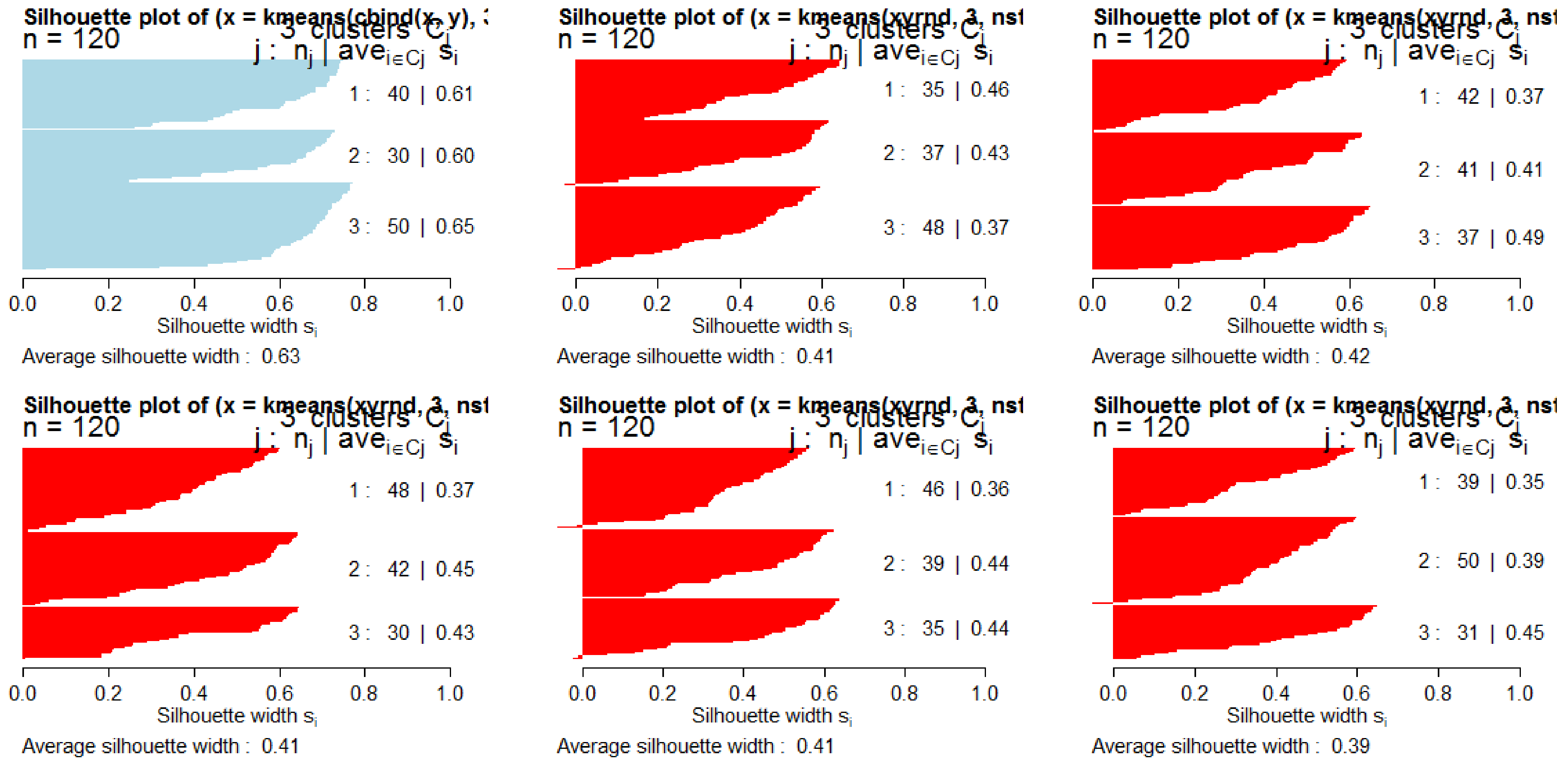
- R has a function for calculating the silhouette (in package `cluster`)
  - Watch for average silhouette (cf. all average silhouettes > 0.6 at K=3 (green) and two much weaker clusters at K=4 (blue))

```
kf=kmeans(cbind(x,y),3)
d=dist(cbind(x,y))
sl=silhouette(kf$cluster,d)
summary(sl)
Silhouette of 120 units in 3 clusters ...
 Cluster sizes and average silhouette widths:
      40        51        29
0.6316847 0.6408889 0.6410914
Individual silhouette widths:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1327  0.5657  0.6812  0.6379  0.7287  0.7635
plot(sl,col="lightgreen")
plot(silhouette(kmeans(cbind(x,y),4)$cluster,d),
      col='lightblue')
```





SILHOUETTES OF RANDOM DATA



- Wider silhouette for the original data as compared to random uniform

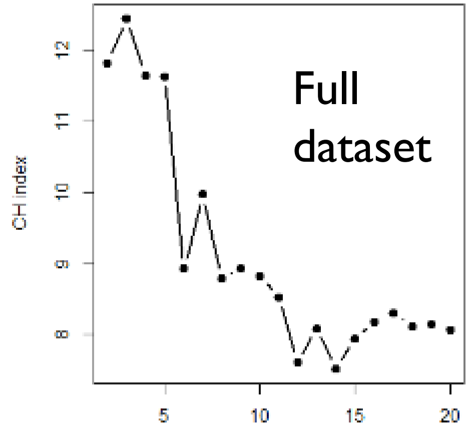
CODE FOR THE PREVIOUS PLOTS

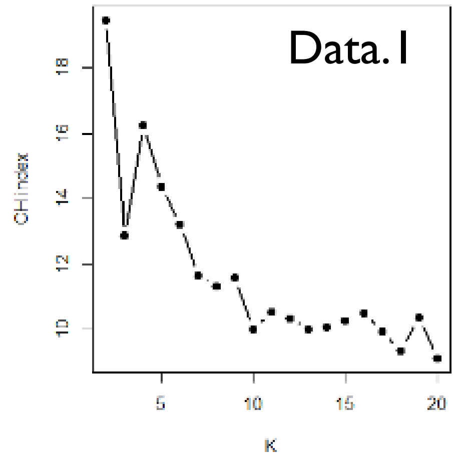
```
old.par = par(mfrow=c(2,3),ps=16)
d=dist(cbind(x,y))
plot(silhouette(kmeans(cbind(x,y),3,nstart=10)$cluster,d),col='lightblue')
for ( iRnd in 1:5 ) {
    xyrnd = apply(cbind(x,y),2,function(x)runif(length(x),min(x),max(x)))
    drnd = dist(xyrnd)
    plot(silhouette(kmeans(xyrnd,3,nstart=10)$cluster,drnd),col='red')
}
par(old.par)
```

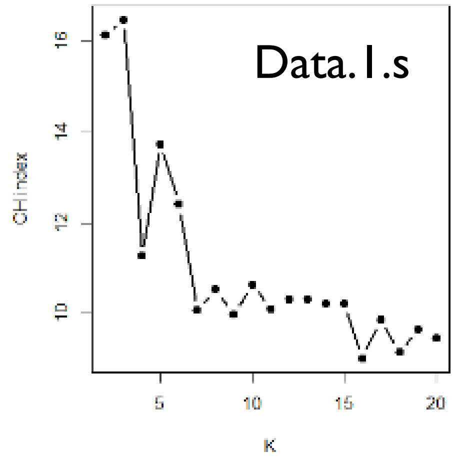
REAL DATA EXAMPLE

- Let us consider again the NCI60 dataset first introduced last week
  - 64 tumor samples, expression levels of 6380 genes are measured in each sample

```
> library(ISLR)
> data(NCI60)
> rownames(NCI60$data)=NCI60$labs
> data.1 = NCI60$data[,apply(NCI60$data,2,sd)>1]
> data.1.s = scale(data.1)
# try k-means first:
> w=numeric(20)
> for ( k in 2:20 ) {
  kf=kmeans(data.1,k)
  w[k] = (kf$betweenss/     # CH index
    (k-1))/(kf$tot.withinss/(length(kf$cluster)-k))
  }
> plot(2:20,w[-1],type="b",lwd=2,pch=19,xlab="K",
      ylab="CH index")
```







! Gap statistics similarly unconvincing!

CLUSTER STABILITY

- How stable are the clusters as we wiggle parameters/rescale the data etc?
- Build a "contingency table" of one clustering result against another
  - Caveat: cluster IDs will be different, need to rearrange the table
  - Problem: given matrix A, find permutation p: [1…n] →[1…n] such that the trace of the rearranged matrix, $\sum_i A[i,p(i)]$, reaches it maximum
  - "Hungarian algorithm" provides the solution, but it is not a simple one… But there is a library!

```
matrix.sort <- function(m) {
    require(clue)
    p = solve_LSAP(m, maximum=T)  # find the permutation…
    m[, p] # and apply it!
}
```

$$\begin{array}{ccc}7&1&2\\2&0&12\\0&9&3\end{array} \Rightarrow \begin{array}{ccc}7&2&1\\2&12&0\\0&3&9\end{array}$$

```
cmp.shortcut = function(K) {
   matrix.sort(table(
      FULL=kmeans(NCI60$data,K,10)$cluster,
      SCALED.SUBSET=
         kmeans(data.1.s,K,10)$cluster))
}
cmp.shortcut(4)
    SCALED.SUBSET
FULL   1  2  3  4
   1   8  0  0  0
   4   0 28  0  0
   3   0  0  9  0
   2   0  2  0 17
```

```
cmp.shortcut(5)
 SCALED.SUBSET
FULL  1  2  3  4  5
   2  8  0  0  0  1
   3  0  9  0  0  0
   1  0  0 15  0  0
   5  0  0  0  8  0
   4  0  0  0  0 23
```

IS HIERARCHICAL BETTER?

- Functions for within/between cluster variance:

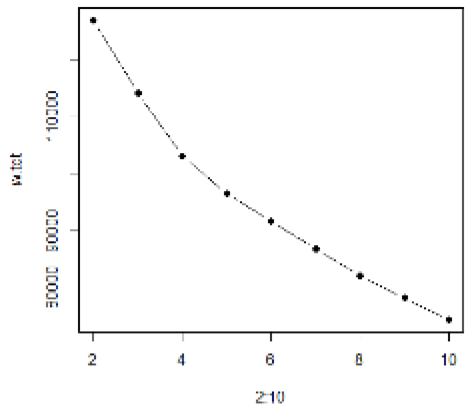
```
within=function(d,clust) {
  w=numeric(length(unique(clust)))
  for ( i in sort(unique(clust)) ) {
    members = d[clust==i,,drop=F]
    centroid = apply(members,2,mean)
    members.diff = sweep(members,2,centroid)
    w[i] = sum(members.diff^2)
  }
  return(w)
}
```

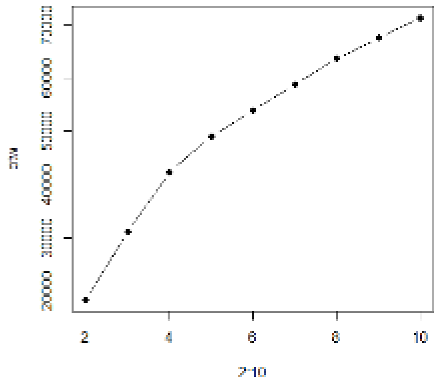
```
between=function(d,clust) {
  b=0
  total.mean = apply(d,2,mean)
  for ( i in sort(unique(clust)) ) {
    members = d[clust==i,,drop=F]
    centroid = apply(members,2,mean)
    b = b + nrow(members)*
        sum( (centroid-total.mean)^2 )
  }
  return(b)
}
```

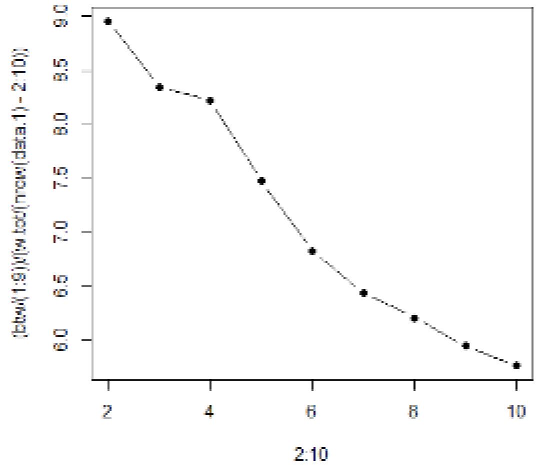
EXAMINING DIFFERENT NUMBERS OF HIERARCHICAL CLUSTERS

- The clear cluster structure is missing from the hierarchical clustering result as well…

```
data.1=NCI60$data[,apply(NCI60$data,2,sd)>1]
dd.1=dist(data.1)
hw.1=hclust(dd.1,method="ward.D2")
w.tot=numeric(9)
btw=numeric(9)
for ( k in 2:10 ) {
  clust = cutree(hw.1,k=k)
  w = within(data.1,clust)
  w.tot[k-1]=sum(w)
  btw[k-1] = between(data.1,clust)
}
plot(2:10,w.tot,pch=19,type="b")
plot(2:10,btw,pch=19,type="b")
plot(2:10,(btw/(1:9))/
    (w.tot/(nrow(data.1)-2:10)),pch=19,type="b")
```



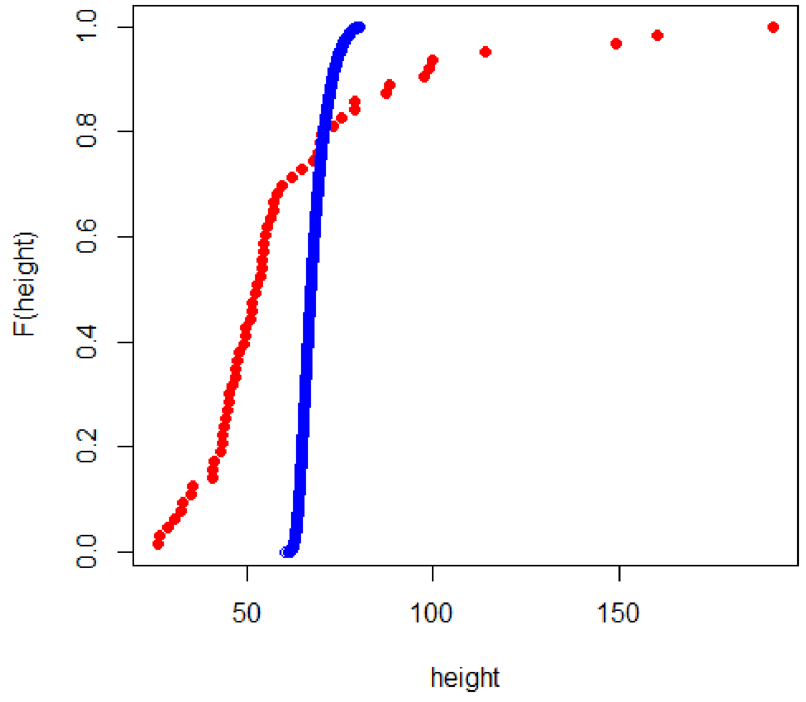




BRUTE FORCE RANDOMIZATION

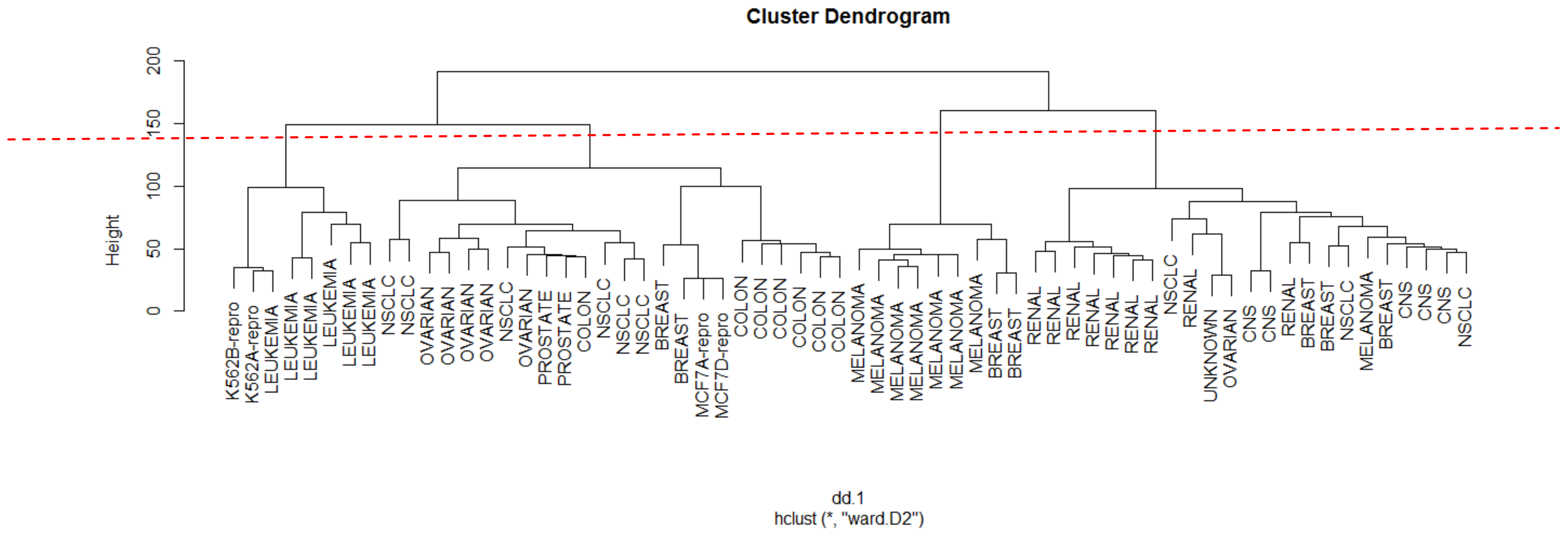
- Let us randomize expression levels across each sample and repeat clustering
- All the "similarities" between samples we are now observing are due to random chance
- What is the distribution of the distances between samples/clusters and how does it compare to the original data?

```
ori.heights = hw.1$height
rnd.heights = numeric()
for ( i.sim in 1:100 ) {
   data.rnd <-apply(data.1,2,sample)
   hw.rnd=hclust(dist(data.rnd),method="ward.D2")
   rnd.heights <- c(rnd.heights,hw.rnd$height)
}
plot(ori.heights,rank(ori.heights)/length(ori.heights),
     col="red",xlab="height",ylab="F(height)",pch=19)
points(rnd.heights,rank(rnd.heights)/length(rnd.heights),
     col="blue")
```

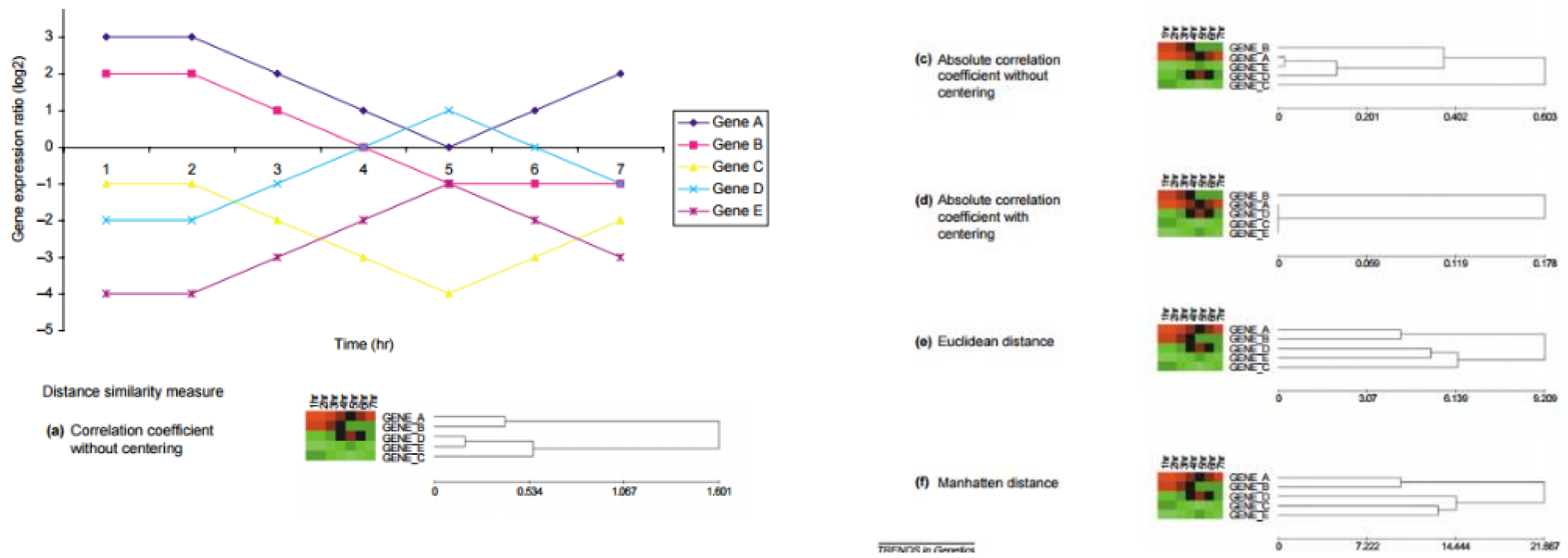


NCI60 CLUSTERS

- About 4 top level clusters *might* be real
  - Brute force resampling can be too optimistic!
- Many other methods exist, including bootstrap of variables (genes in our example) or multilevel bootstrap
  - See package `pvclust`



NOTE ON DISTANCES



“AIRWAY” DATASET

- Another gene expression experiment
  - Can be downloaded as library/dataset from Bioconductor (another large repository of R packages, biology-oriented)
  - 4 cell lines (lung airway, asthma), 2 conditions: untreated (control) and treated with a drug

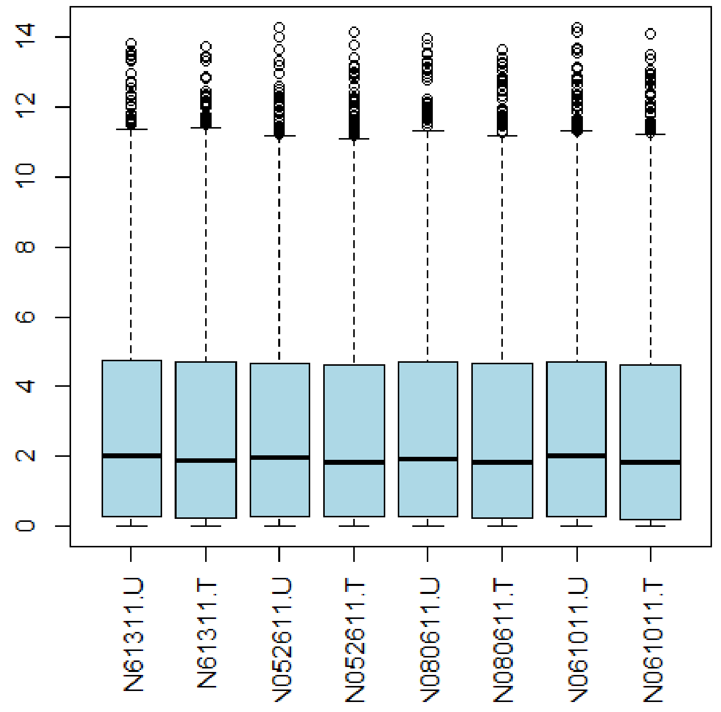
```
source("http://bioconductor.org/biocLite.R")
biocLite("airway")
library(airway)
data(airway)
airway
class: SummarizedExperiment
dim: 64102 8
exptData(1): ''
assays(1): counts
rownames(64102): ENSG00000000003 ENSG00000000005 ...
rowData metadata column names(0):
colnames(8): SRR1039508 SRR1039509 ...
colData names(9): SampleName cell ...
```

```
> colData(airway)[1:3,1:4]
DataFrame with 3 rows and 4 columns
           SampleName     cell      dex    albut
             <factor> <factor> <factor> <factor>
SRR1039508 GSM1275862   N61311    untrt    untrt
SRR1039509 GSM1275863   N61311      trt    untrt
SRR1039512 GSM1275866  N052611    untrt    untrt
> assay(airway)[1:3,1:3]
                SRR1039508 SRR1039509 SRR1039512
ENSG00000000003        679        448        873
ENSG00000000005          0          0          0
ENSG00000000419        467        515        621
```

PREPARE DATA

- Many genes are in fact not measured (count = 0 or extremely low) : remove unmeasured genes
- If we "count longer", we will count proportionally more events per gene. Need to normalize the counts by the total number of counts in each sample
  - Better methods do exist!
- Distribution of expression levels is log-normal. Take the log.

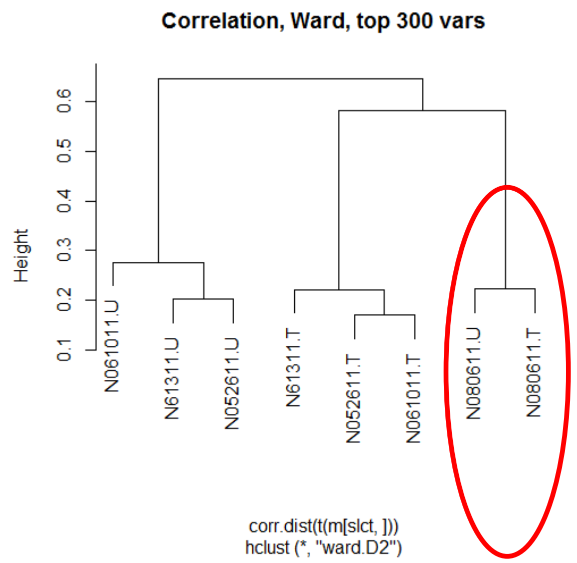
```
> m = assay(airway)
# use cell line/treatment to rename columns (samples):
> colnames(m)= paste(colData(airway)$cell,
         ifelse(colData(airway)$dex=="untrt","U","T"),sep=".")
> labels=colnames(m)
> m = m[ apply(m,1,sum) > 5, ]  # cuts m from 64K rows (genes) to 24K
> total.counts=apply(m,2,sum)
> m = sweep(m,2,total.counts/1000000,FUN="/") # normalize per MILLION cnts
> m=log2(m+1) # regularize log
> boxplot(m,col="lightblue‟,las=3)
```

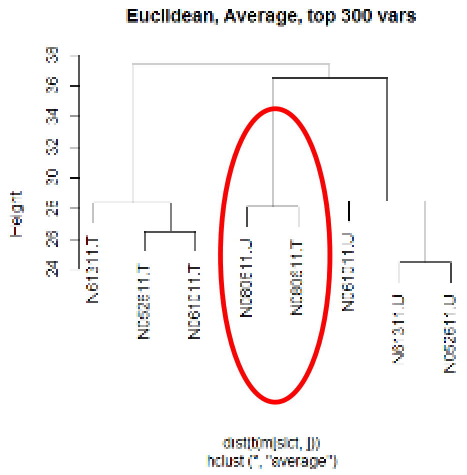


CLUSTER SAMPLES

- Let us consider the genes ("variables") with largest variance across the "experimental observations" (samples) and cluster samples on those genes
  - Not shown: we could cluster on full set of variables, the results would be less stable but still strange

```
# correlation distance (Note the t(): we are going to use the
# same contract as dist() which calculates pairwise
# distances between rows (observations)
corr.dist=function(x) { as.dist(1-cor(t(x))) }
vars=apply(m,1,var)
slct=order(vars,decreasing=T)[1:300]
plot(hclust(corr.dist(t(m[slct,])),method="ward.D2"),
   main="Correlation, Ward, top 300 vars")
plot(hclust(dist(t(m[slct,])),method="average"),
        main="Euclidean, Average, top 300 vars")
```

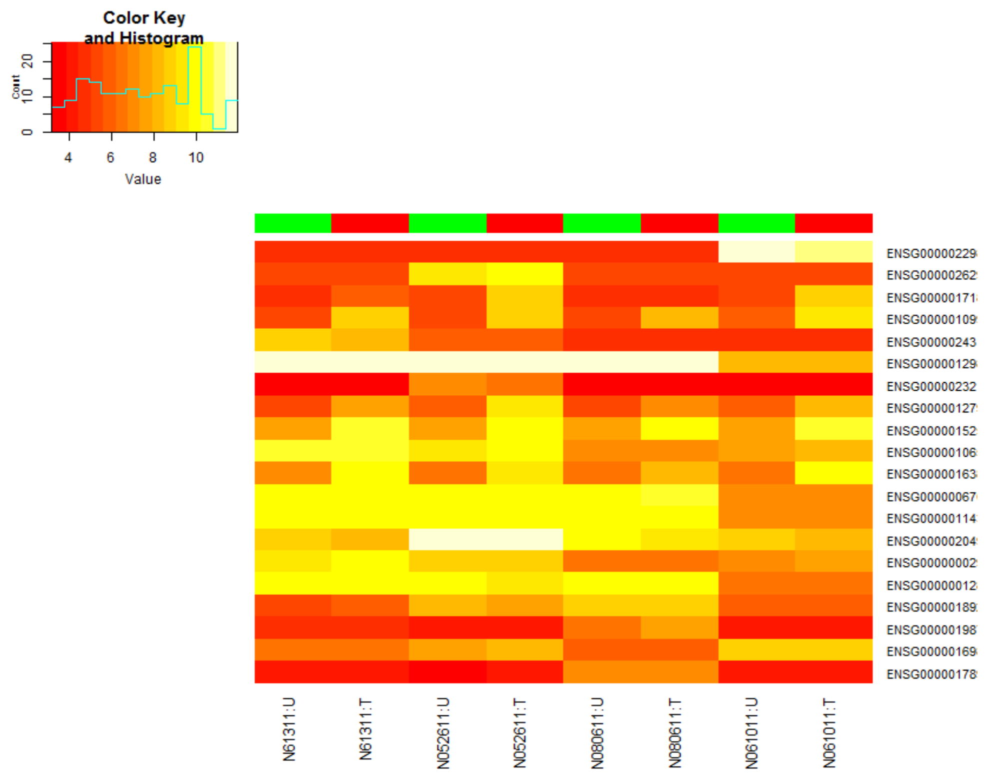




INTRODUCING HEATMAP

- Simplest heatmap
  - Order of rows/columns the same as in the data table (must use Rowv=F and Colv=F); values are represented with color gradient (see the legend)
  - Note how we set color code for untreated/treated (green/red) on top (ColSideColors)

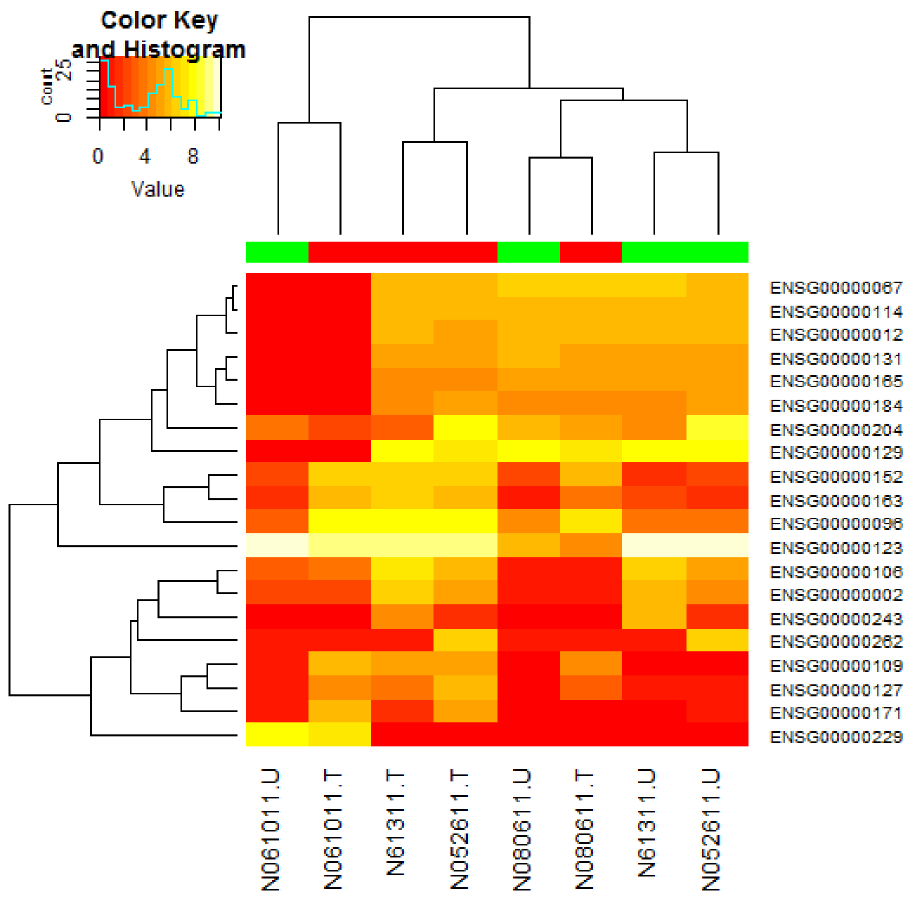
```
library(gplots)
slct=order(vars,decreasing=T)[1:20]
heatmap.2(m[slct,],trace="none",labCol=labels,
    ColSideColors=rep(c("green","red"),4),
    margins=c(7,7),Rowv=F,Colv=F)
```



CLUSTERED HEATMAP

- Heatmap can also automatically cluster both rows AND columns
- Two-way clustering: samples are clustered according to the similarity of expression patterns across (considered) genes in each sample
- Genes are clustered according to the similarity of their expression patterns across the samples

```
heatmap.2(m[slct,],trace="none",
  labCol=labels,
  ColSideColors=rep(c("green","red"),4),
  margins=c(7,7))
```



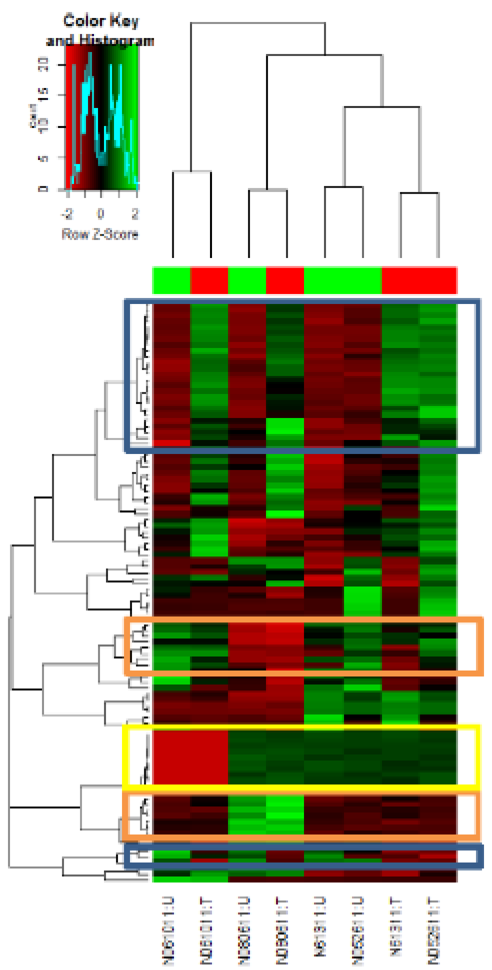
CONSIDERING LARGER SET

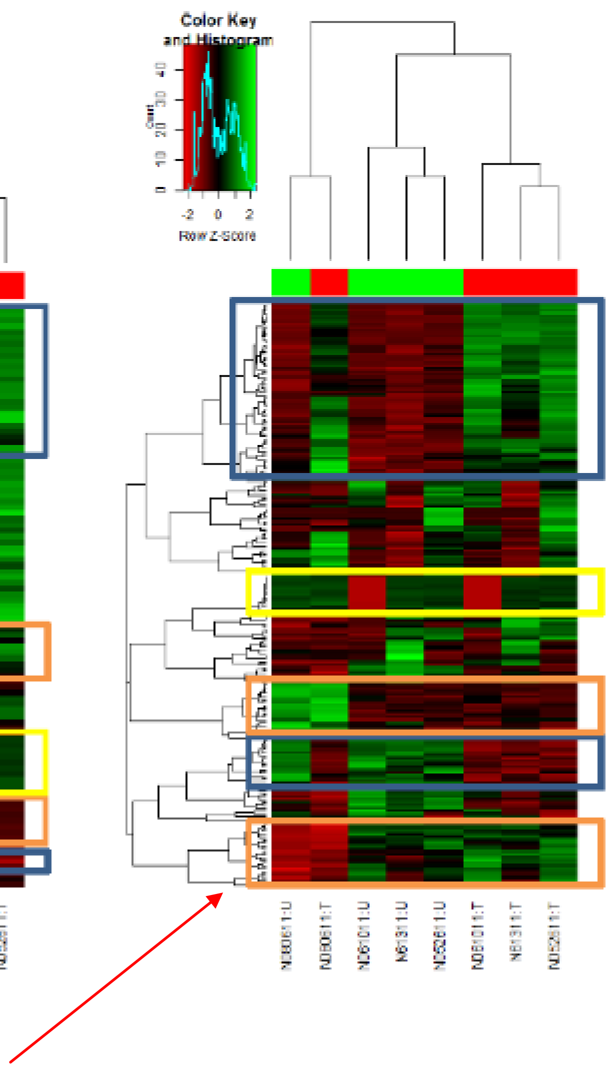
- Let us (a) consider the larger set of genes (comparable to what we used before), (b) use the same clustering distance (heatmap allows us to do that!) , and (c) enhance visualization:

```
# generates a vector of n colors transitioning from red, through black,
# to green
redgreen <- function(n) {
     c( hsv(h=0/6, v=seq(1,0,length=n/2) ),hsv(h=2/6, v=seq(0,1,length=n/2) ) )
}
# try 100 highest-variance genes first
slct=order(vars,decreasing=T)[1:100]
# use correlation distance, center and scale rows, use red-green palette,
# and turn off gene labels for now - there are too many to be readable anyway:
heatmap.2(m[slct,],trace="none",labCol=labels, labRow=F,
            col=redgreen(100),scale="row",
            ColSideColors=rep(c("green","red"),4),margins=c(7,7),distfun=corr.dist)
#now try 200 highest-variance genes
slct=order(vars,decreasing=T)[1:200]
heatmap.2(m[slct,],trace="none",labCol=labels,labRow=F,
            col=redgreen(100),scale="row",
            ColSideColors=rep(c("green","red"),4),margins=c(7,7),distfun=corr.dist)
```

DISTINCT PATTERS OF EXPRESSION

- We observe the same effect: one cell line starts clustering away when we consider 200 genes
- We clearly see a few different gene clusters with specific expression patterns.
  - Some distinct clusters are outlined with colored rectangles, note different expression patterns across samples
  - Note the cluster that differentiates the outlier sample from the rest
  - Upon close examination, we would discover that downregulated genes in this cluster are located… on Y-chromosome!
  - In *contrast to what's claimed in the publication*, one cell line comes from a female!





SUMMARY

- The question "how many clusters" is *very difficult*
  - Just like everything else we have seen so far: the question is not only "how many clusters are truly there" but also "how many clusters we can possibly distinguish with the data in hand" ! (cf. : "what's the true underlying dependence" vs "what we can possibly do with available data" – for instance should we even try higher order polynomial regression, or that would give us too many variables for the small dataset we have?
  - There is a lot of metrics and statistics that can (and should) be examined (again, similar to the large number of statistics we may want to look at when performing model selection
- The question of "how good the clusters are" is closely related (when the "goodness" metrics are clear and telling, we are also likely to know how many clusters are there!)
- Nonetheless, unsupervised learning (e.g. clustering) is very important and useful for understanding the structure of the data, sample QC, and for discovery/hypothesis generation
- Another potential use: a *recommender system* (e.g. find customers who are "similar" to customer X and send X offers that other members of their cluster were responding to)