Week 3

Data Analysis

prompt - in the example

assignment - exampleAnalysis.pdw

↑

writeUp (note references!)

one figure + figure caption

Rmd - R markdown

Rda - r data format

code/

final code/

raw code/

cleaned

data/

raw

rda

writing/

docx + pdf

Explonatory Analysis

Why graphs?

understand,
discover patterns
modelling strategies
• "debug" analysis
communicate results

Characteristics: quickly (raw)

many drafts

goal: for personal understanding
(not always esthetic)

How to display differences (how to compare?)

- position (common scale)
- different scales
- areas, volumes, .

etc

communicate accurately!

pie charts vs bar charts

↗ high error rate

↑ low error rate

Summary:

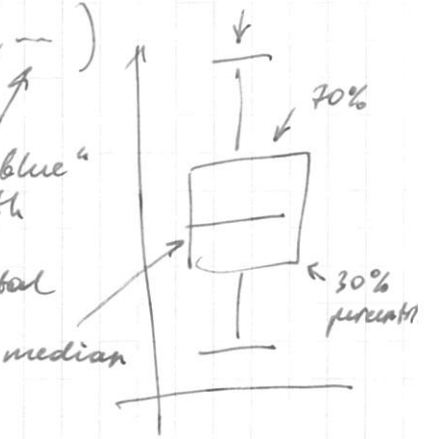
- use common scales when possible
- and position comparisons
- pie chart aren't good
- no 3D bar charts

- Boxplot

boxplot(pdata$AGEP, ...)

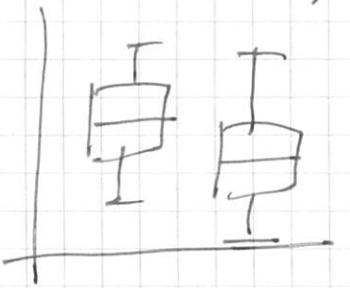
col = "blue"
col.width
names
horizontal

you can use 2 vars

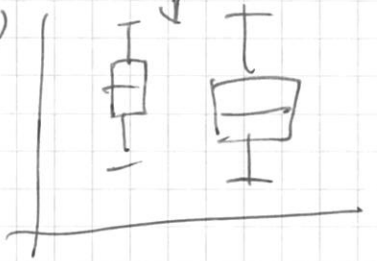


what is distribution?
is it compact?
symetric?

boxplot(p$a ~ as.factor(p$f),
col = "blue")
col = c("blue", "orange")



var with
(with is proportional
to volume)



- Barplot

```
barplot(table(p$c), col="blue")
```

(comparison ~~for~~ position)

- Histograms

```
hist(p$a, col="blue")
```

(values are grouped by ranges)

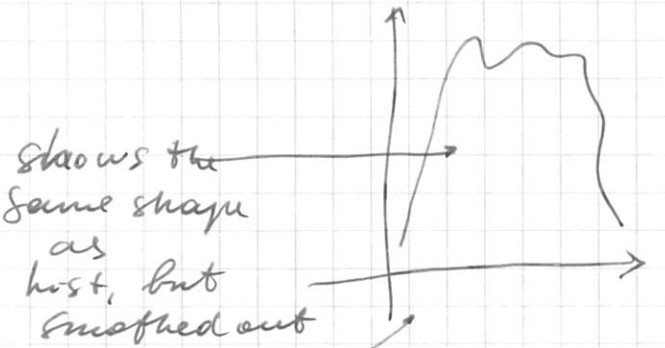
frequency distribution (you can see the shape)

param: breaks = 100 (how many bars)

- Density plot
  like a hist, but smoothed out

```
dens = density(p$a)
plot(dens, lwd=3, col="blue")
```

↑ thickness of line




```
lines(dens$x, dens$y, lwd=3, color="orange")
```

adds a new line to the plot (draws near)

- Scatter plot

visualize relationships between variables

plot(px, py, pch=19, col="blue")

- px, py: at the same lenght
- ?par - for parameters (most important time)
- pch=19: for dots (filled circles)
- col="blue"
- cex=0.5: for making circles smaller

col = p$sex ← for coloring different variables
↑ vector with 2 vars.

all these params can be of the same len

clpz percentage = 0.5
↑ for visualization relationships

lines
points ↗ for adding extra lines, etc

```
library(Hmisc)
ageGroups = cut2(p$age, g=5)
```

split onto 5 group

then use it for coloring your plot

if a number of dots is too large and they overlay each other,

- sample (random)
- smoothScatter
  creates smooth density plot

```
x = rnorm(1e5)
y = rnorm(1e5)
smoothScatter(x, y)
```

The darker the more points

- hexbin package similar

```
# library(hexbin)
x = rnorm(1e5)
y = rnorm(1e5)
hbo = hexbin(x, y)
plot(hbo)
```

- QQ plot

  plots quantiles

  are 2 distributions similar?

```
x = rnorm(20)
y = rnorm(20)
qqplot(x, y)
abline(c(0, 1))
```

  ↑ 45° line

- matplot | spaghetti

```
X <- matrix(rnorm(20*5), nrow=20)
matplot(X, type="b")
```

- heatmaps

  fuzzy histogram

```
image(1:10,
      161:236,
  as.matrix
    (p.data[1:10,
            161:236])
```

  ↓ same image

  the brighter the color, the larger the value

  (little bit confusing)

  t ↑ transpose function

- Maps

  geographic maps

```
library(maps)
map("world")
lat = runif(40, -180, 180)
...
lon = ricinf(40, -90, 90)
points(lat, lon, col = "blue", pch = 19)
```

↓ draws dots on the map

- R doesn't draw NAs.

```
boxplot(x ~ is.na(x))
```

how many NAs? ↗

Expository Graphs

to show them others
(to communicate information!)

main goal

color and size matter

understandable labels, axis, etc

```
plot(p$a, p$b, pch=19, col="blue",
     cex=0.5,
     xlab="Travel time (min)")
```

cex.lab
cex.axis
size of labels, axis

units are important!

`legend` command adds a legend

```
legend(100, 200000, legend="title",
       col, pch=19)
```

main="title"

```
legend(... legend=c("men", "women"),
       pch=c(19,19),
       cex=c(0.5,0.5))
```

several panel plots:

par(mfrow = c(1, 2))

hist(...)

plot(...)

mtext — label to be added to margin

(text = "a")

Figure captions

Figure 1... Title (a)...
(b)...

↑ key components

Save the file

? Devices – all devices

pdf device

inches

pdf(file = "file.pdf", height = 4, width = 8)

par(mfrow...)

code is unreal:

dev.off()

png ← the same

```
png(file = "1.png", height = (pixels)
                    merge )
```

tip: prepare graph

```
dev.copy2pdf(file = "file.pdf")
```

Hierarchical Clustering

how close things are?

- what is „close"?
- how do we group close things?

- An agglomerative approach
  - start by finding closest 2 things,
  - put them together
  - find next closest

  until you merge it all

how do we define distance?

how do we merge?

Result: a tree showing how close things are to each other

↑

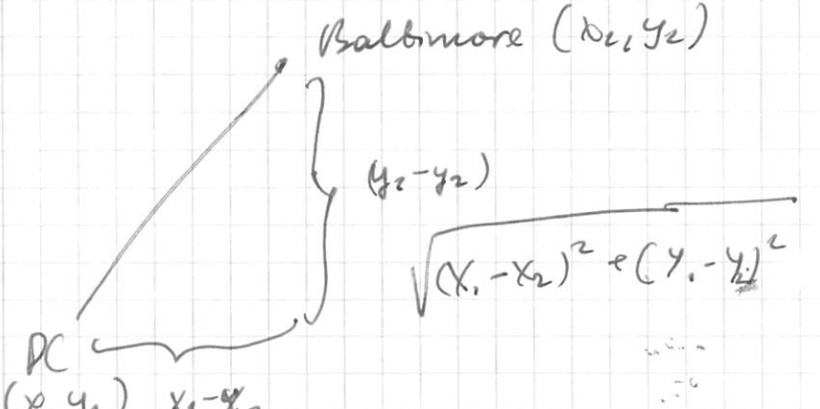
dendogram

How do we define "close"?

distance or similarity:

- cont - euclidean dist
- cont - correlation similarity
- binary - manhattan distance

• Euclidean

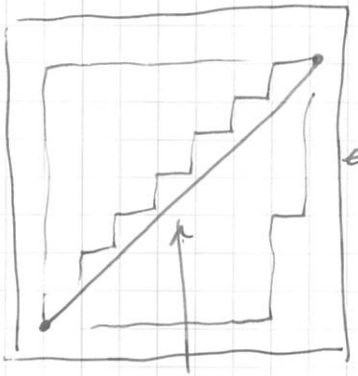


$$d = \sqrt{(A_1 - A_2)^2 + (B_1 - B_2)^2 + \dots (Z_1 - Z_2)^2}$$

General case

lecture 5 - clustering.pdf

Manhattan distance



← binary (cells)

$$d = |A_1 - A_2| + |B_1 - B_2| + \dots + |Z_1 - Z_2|$$

Euclidean
(you can't take it, there are buildings)

```
set.seed(1234); par(mar=c(0,0,0,0))
x = rnorm(12, mean=rep(1:3, each=4), sd=0.2)
y = rnorm(12, mean=rep(c(1,2,3), each=4), sd=0.2)
plot(x, y, col="blue", pch=19, cex=2)
text(x+0.05, y+0.05, labels=as.character(1:12))
```

labels
dots

```
dataFr = data.frame(x = x, y = y)
dist(dataFr)
        method = ...
```

↓

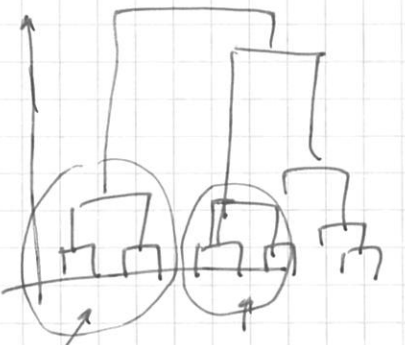
matrix of distances for

Algo:

take the closest.
merge them ($x_n = \frac{x_1 + x_2}{2}$, $y_n = \frac{y_1 + y_2}{2}$)

↑ middle

```
hClustering = hclust(dost.oy)
plot(hClustering)
```



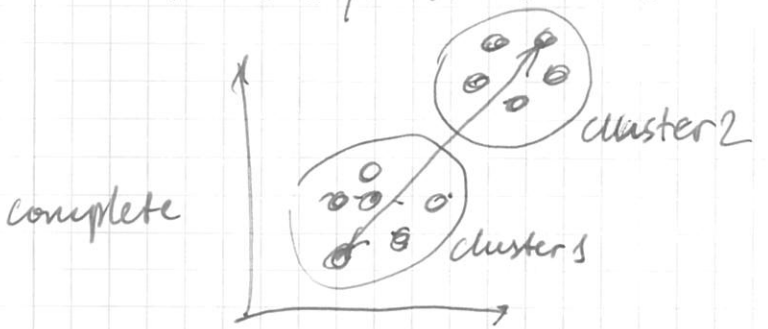
Cluster dendrogram →

you have to cut the tree at some point to get clusters

another way of merging:

complete linkage:

instead of calculating avg, we take the farest elements



heatmap() ← for 2-dim data sets

Clustering:

- gives an idea of the relationships between variables
- picture may be unstable
- choosing where to cut isn't very obvious

Books: Elements of Statistical learning
(slide 5-5, p21)

K-Means Clustering

- A partitioning approach
  (not agglomerative, as dendrograms)

  - fix a number of clusters
  - get "centroids" of each cluster
  - assign things to closest centroid
  - recalculate centroids

  (iterate)

need: defined distance
number of clusters
initial guess to centroids

gives final estimate of cluster centroids

an assignment of each point to cluster

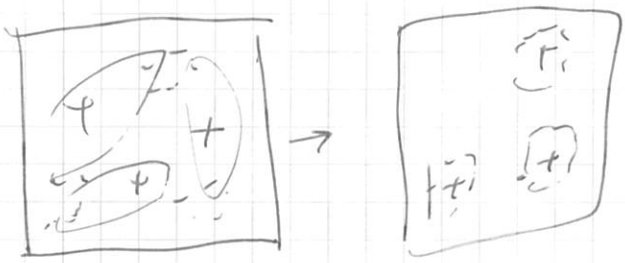
(the same code as prev)

Assume 3 clusters,

assign to the closest centroids
recalculate distance,
reassign

iterate until converges

eventually centroids will move close to actual clusters



kmeans()

kmeansObj = kmeans(dataFrame, centers=3)

↑ number of centroids

other params:

iter.max - if not converges, how many iterations to do until stop

not deterministic
results may vary

kmeansObj$cluster ← factor.

kmeansObj$centers

determine the number of clusters!

(intuition, algorithms, etc)

Dimention Reduction Techniques

- Principal Components Analysis PCA
- Singular Value Decomposition SVD

Take large set of variables ⇒

compress to a smaller one which is easier to interpret

SVD:

svd command in R

if X is a matrix with each variable in a column, and each observation is a row, then

SVD is a "matrix decomposition"

$$X = UDV^T$$

U - orthogonal columns (left singular vector)

V - orthogonal columns (right sing. vector)

D - diagonal matrix (singular values)

svd1 = svd(scale(matr))

svd1$u - left sing

svd1$v - right sing

svd1$d ← how many patterns are there?

PCA

equal to the right singular values if you first scale the variables

scale: to substract the mean and divide by standart deviation

prcomp ← exactly the same as svd$v

How SVD works

we are able to see if there are any patterns

(not clear! read about it, if needed)

---

impute – приписывать, обозначать

impute - replace NAs with calculated data
(avg of neighbours, etc)

So, SVD can be used for creating approximations and reducing the size by considering only found patterns

```
svd1 = svd(scale(face))
```

1) `approx1 = svd1$u[,1] %*% t(svd$v[,1] * svd$d[1])`

recall the formula

using only first variable pattern in U vector

5) `approx5 = svd1$u[,1:5] %*% diag(svd$d[1:5]) %*% t(svd$v[,1:5])`

need to make diagonal matrix out of d

Again! Elements of Statistical Learning!
TOREAD!!!