1

Statistical Modelling and Computing

Nick Fieller

Department of Probability & Statistics
University of Sheffield, UK

visiting
University of Tampere
2004/05

2

Books

- **Venables, W. N. & Ripley, B. D.** (2002) *Modern Applied Statistics with S-PLUS,* (4th Edition). Springer
  - http://www.stats.ox.ac.uk/pub/MASS4
- **Verzani, J.** (2005) *Using R for Introductory Statistics*, Chapman & Hall.
  - This book provides many good examples of the more elementary techniques
- The **R** on-line help system **`help(…)`**

3

Objectives

- Introduce some techniques of modern statistical methodology
  - understanding data
  - discovering structure
  - & making inferences about the wider world
- Applied Statistics is not mathematics
  - but mathematics is useful in methodological development

it is closer to '*data engineering*'

- so need to consider computational side

4

Revolution in statistics

- development of the statistical computing language **S**
  - (at Bell Laboratories, late 70s)
- allowed extension of traditional methods and new ways of investigating practical problems
  - more intuitive ideas and less mathematical
  - implements known ideas but previously computationally impossible
  - e.g. simulation / randomization tests
  - attention to graphics (intuitive presentation)
- **S-Plus:–** commercially supported implementation of **S**
- **S** for **S**tatistics

5

R developed by Ross Ihaka & Rob Gentleman in mid 1990s

- since 1997 run by **R** Development Core Team
  - part of GNU project
- free open source software
  - download from web
  - http://www.r-project.org
- **R** because it is 'very close' to **S**

6

Outline

- Overview of **R** and **S+**
  - how does it work & what can it do
- Exploratory data analysis
  - robust summaries, alternatives to histograms
- Classical univariate statistics
  - 1 & 2 sample tests, bootstrap & permutation tests
- Linear statistical methods
  - robust and smooth regression, additive models
- Multivariate methods
  - EDA, PCA, Trees, Neural networks etc

7

- Overview of **R**
  - ♦ set of tools for statistics & data analysis
  - ♦ a language for expressing statistical models
  - ♦ graphical facilities for interactive analysis
  - ♦ object-orientated language
    - can easily be extended
  - ♦ expanding set of publicly available libraries
    - almost all new techniques & methods appear as a new library in **R**
      - if not then they are never used

8

- **R** and **S**-Plus compared
  - ♦ both are *open source* systems
    - i.e. possible to see exactly what algorithms are used
      - important for 'intuitive' or 'heuristic' methods
    - (unlike SPSS or SAS)
  - ♦ **S**-plus has menus and dialogue boxes **R** has only command line
  - ♦ **R** is small with extensions **S**-Plus is monolithic
  - ♦ **R** is faster and runs on smaller machines

9

- Key features of **R**
  - ♦ all commands are *functions* operating on *arguments*
    - `plot(x,y)` plots vector x against vector y
    - `help(plot)` provides help on the function plot
    - `quit()` stops the session – has a null argument
  - ♦ items within **R** are *objects* of various types
    - vectors, matrices, graphs, data arrays, complete results of analyses
      - display by typing its name or `summary(object)`
  - ♦ **R** is *case-sensitive*
    - `t.test()` and `T.test()` are different

10

- Example
  - ♦ use of `library()`
  - ♦ use of `help()` system
  - ♦ opening and listing datasets
  - ♦ `attach(dataset)`
  - ♦ simple plotting
  - ♦ t-tests
  - ♦ generic functions

11

- ♦ `library(MASS)`
  - opens Venables & Ripley's library of statistical functions & interesting data sets
- ♦ `library(help=MASS)`
  - gives a description of what is inside library
- ♦ `data(hills)`
  - opens dataset `hills` and can perform global operations (listing, all pairwise plots etc) but cannot refer to individual variables
- ♦ `attach(hills)`
  - opens dataset **and** makes names of variables available
- ♦ `t.test(A, B)` and `t.test(A-B)`
  - 2-sample and 1-sample t-tests, generic function

12

Exploratory Data Analysis

- ♦ Standard summary statistics
  - `mean()`, `median()` and `var()` etc
- ♦ some may be very sensitive to ***outliers***

```
> data(chem)
> chem
 [1]  2.90  3.10  3.40  3.40  3.70  3.70
      2.80  2.50  2.40  2.40  2.70  2.20
[13] 5.28  3.37  3.03  3.03 28.95  3.77
      3.40  2.20  3.50  3.60  3.70  3.70
> mean(chem)
[1] 4.280417
```

- ♦ summary statistic 4.28 larger than all but two of the observations – not satisfactory
- ♦ `median(chem)` `3.385` – a more typical value

- The median is ***robust*** to presence of outliers
  - or ***resistant*** to large errors in data
- may prefer a description of the general picture not influenced by one-off outliers
  - calculate mean after removing largest & smallest few observations i.e. a trimmed mean

```
> mean(chem, trim=0.05)
[1] 3.253636
```

  - mean after trimming away top & bottom 5% observations

- similar ideas for robust estimators of scale & in many other analyses (e.g. fitting lines)
- robust methods offer protection against (minor) failure of assumptions but at a price
- robust estimators may be less precise
  - i.e. have a higher variance
- pay a price (i.e. loose precision) but gain protection if data contaminated or model deviates from that assumed

Graphical Summaries

- stem & leaf plots

```
> data(hills)
> dist
  2.5 6.0 6.0 7.5 8.0 8.0 16.0 6.0 5.0 6.0 28.0
  5.0 9.5 6.0 4.5 10.0 14.0 3.0 4.5 5.5 3.0 3.5
  6.0 2.0 3.0 4.0 6.0 5.0 6.5 5.0 10.0 6.0 18.0
  4.5 20.0
> stem(dist)
The decimal point is 1 digit(s) to the right of
  the |
  0 | 2333344
  0 | 55555556666666667888
  1 | 0004
  1 | 68
  2 | 0
  2 | 8
```

Quick, easy, no data are lost — actual values are retained

Boxplots

```
> attach(geyser)
> boxplot(duration,sub="duration")
```

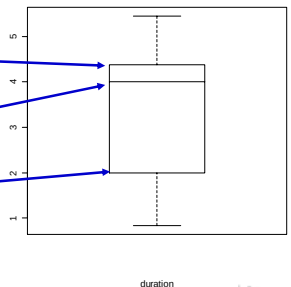
Box contains 'middle half' of data

upper quartile

median

lower quartile

whiskers give range of data values (extend to value no more than 1.5 time inter-quartile range, outliers beyond this marked individually)

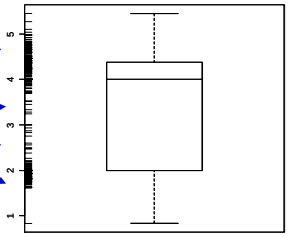


Boxplots

```
> attach(geyser)
> boxplot(duration,sub="duration")
> rug(duration, side=2)
```

actual values

a rug is a type of small carpet



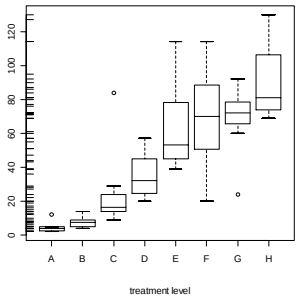

```
> boxplot(decrease~treatment)
> rug(decrease, side=2)
```

Effective for showing several data sets together

Counts increase with treatment level and variance also increases

`decrease~treatment`

"relate decrease to treatment"



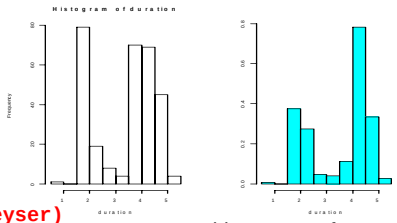
19

Histograms and Density Estimates

- Sample of observations
- Histogram gives an *estimate* of underlying density of the data
- Need to choose
  - Bin width
  - Starting value
- Various rules for choosing these so histogram is not too rough nor too smooth
- Can have very different looking histograms of same data on same scale and same bins

20

Histograms and Density Estimates

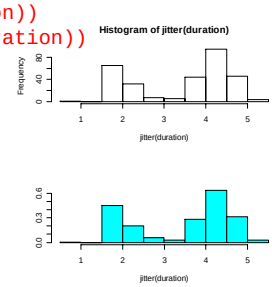



```
> attach(geyser)
> hist(duration)
> truehist(duration)
```

Two histograms of same data with different rules for values on class boundaries

21

```
> hist(jitter(duration))
> truehist(jitter(duration))
```



Add random amount to avoid values on class boundary

22

Kernel Density Estimates

- In a histogram each observation contributes one small rectangular brick to build up column in the bin it falls into

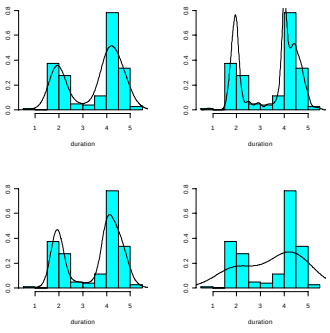
$$\tilde{f}(x) = \frac{\#(x; c_i \le x < c_{i+1})}{n(c_{i+1} - c_i)}$$

- In a kernel density estimate each observation adds a smooth curve K(.) instead of a solid brick
  - Still requires choice of bandwidth b

$$\hat{f}(x) = \frac{1}{nb}\sum_{j=1}^{n} K\left(\frac{x - x_j}{b}\right)$$

23

Kernel density estimates of Old Faithful data with default, 0.3×default, 0.7×default and 2.5×default bandwidths



Default is often a little too big and adjust by 0.7 or 0.8 is often a little better

24

Classical Univariate Statistics

(& Alternatives)

- Classical statistical tests and p-values
- Simple simulation methods
- The Bootstrap
- Permutation and Randomization tests

25

- Example: shoes data
  - Measures of wear of materials A and B
  - If no difference between A & B then expect average difference in wear to be about 0
  - Differences:
    - -0.8 -0.6 -0.3 0.1 -1.1 0.2 -0.3 -0.5 -0.5 -0.3
  - Average difference = – 0.41
  - Does this give evidence that A & B differ??

26

- Need to decide if such a difference could have happened by chance even if A & B are the same
- Classical approach:
  - Classical approach: calculate test statistic **t**

$$t = \frac{\text{mean of differences}}{\text{standard deviation of differences}}$$

then decide if this statistic is bigger than could happen by chance

How?

27

- **Answer:**

  use a series of theorems which says that if there is no difference between A & B then this value is an observation from, a Student's t-distribution on 9 degrees of freedom.

  And so we can calculate the probability of obtaining an observation as big as this (or even bigger)

28

- Theorems needed:-
  - If observations are Normal then sample mean is Normal
  - If observations are independent and Normal then sample variance is chi-squared with n – 1 degrees freedom
  - If observations are Normal then sample mean and variance are statistically independent
  - Ratio of Normal and square root of independent chi-squared is a Student t-distribution

29

- Theorems needed
  - If observations are Normal then sample mean is Normal
  - If observations are independent and Normal then sample variance is chi-squared with n – 1 degrees freedom
  - If observations are Normal then sample mean and variance are statistically independent
  - Ratio of Normal and square root of independent chi-squared is a Student t-distribution

Several different theorems

30

- If the conditions needed for the theorems to be true are OK
  - then we can calculate the *significance* of our observed statistic and come to a [statistical] decision on whether the observed difference could have arisen just by chance even if the materials A & B were the same.

- In this case we have t = –3.489 and calculations show if there is no difference between A & B then this could only happen with probability 0.008539 (see p44)
  - Less than one in one hundred
  - Conclude unlikely to happen just by chance and so decide that A & B are not the same

- In principle we can use a similar procedure for other situations:
  - Find a sensible test statistic
  - Use mathematics to find its theoretical probability distribution
  - Calculate probability of observing the value we have obtained
- OK in simple situations

- **One alternative:** Wilcoxon sign test
  - Differences:
    - -0.8 -0.6 -0.3 0.1 -1.1 0.2 -0.3 -0.5 -0.5 -0.3
- If no systematic difference between A & B then equal chance that A or B will be bigger so expect equal chance of a positive or negative difference
  - -0.8 -0.6 -0.3 **0.1** -1.1 **0.2** -0.3 -0.5 -0.5 -0.3
  - 2 positive, 8 negative, chance of getting this is 0.01431 (see p46)

- Fewer mathematical theorems needed
- Don't need to assume data are Normal
- Larger p-value (i.e. weaker evidence)
  - Because we have assumed less about the data (i.e. not assumed data are Normal)
  - less in $\Rightarrow$ less out
    more in $\Rightarrow$ more out
- Had to use different test statistic
  - Not easy to generalize to other situations

- What could we do if we had no theorems to help calculate the probability of our test statistic?
  - e.g. in a more complex situation
- Try the experiment again with same material on each shoe
- Experiment artificially by simulation

- Generate 10 Normally distributed values to simulate the A sample
- Generate 10 Normally distributed values to simulate the B sample
  but make sure they have the same mean as A
- Calculate the test statistic **t** and compare with our observed one
- One simulated sample is not enough so repeat many times and compare

37

- Generating Normal samples
  - **R** command `rnorm()`
    - `help(rnorm)`for details
  - Generate a sample uniformly distributed in (0,1) and convert by the Normal distribution function to obtain a sample from a Normal distribution.

38

Converting uniform sample to Normal sample

Random sample uniformly distributed on (0,1)

1.0

0.0

39

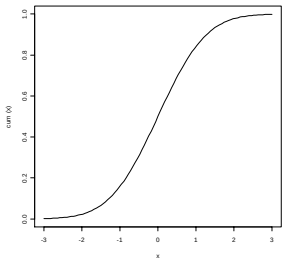
Converting uniform sample to Normal sample

Random sample uniformly distributed on (0,1)

1.0

0.0

Normal distribution function



40

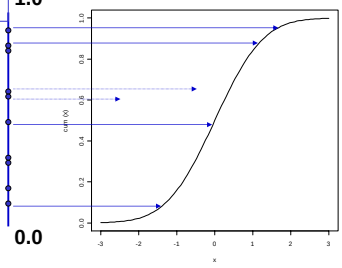
Converting uniform sample to Normal sample

Random sample uniformly distributed on (0,1)

1.0

0.0

Normal distribution function



41

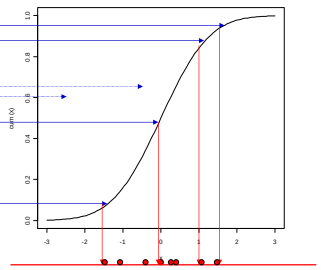
Converting uniform sample to Normal sample

Random sample uniformly distributed on (0,1)

1.0

0.0

Normal distribution function



42

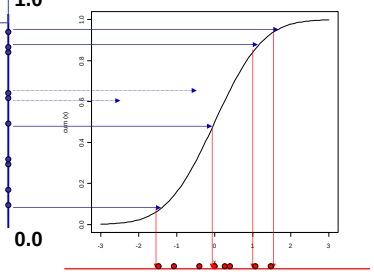
Converting uniform sample to Normal sample

Random sample uniformly distributed on (0,1)

1.0

0.0

Normal distribution function



Random sample Normally distributed

- Different example (P49)
  - Sample of size 20 from N(5,2.7^2)
    - Mean= 4.76, var=2.67^2 (=7.11)
  - Want a confidence interval for the population mean
  - Use mathematical results to say we need to use a value from Student's t_{19} distribution

OR

- Simulate 100 separate samples of size 20 from N(5,2.7^2)
  - Calculate the mean of each of them
  - Find what range of values 95% of these have
    - Easy to this by sorting in order
- Difficulty: will not really know that the true mean and variance are 5 and 2.7^2

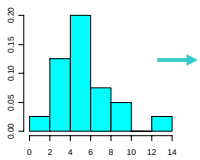
SO

- Use best guess of sample mean (4.76) and sample variance (2.67^2)
  - Simulate the samples from N(4.76, 2.67^2)
  - Calculate the mean of each of them
  - Find what range of values 95% of these have
- Difficulty: do not really know that data are Normal

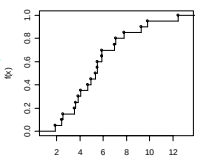
SO

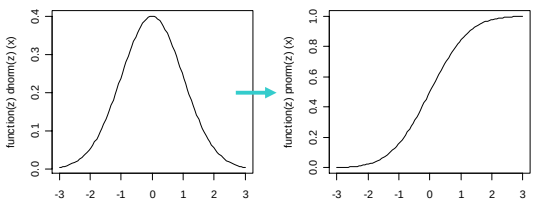
- Need to ***estimate*** the distribution
  - Histogram
  - Kernel density estimate

Histogram



empirical distribution function





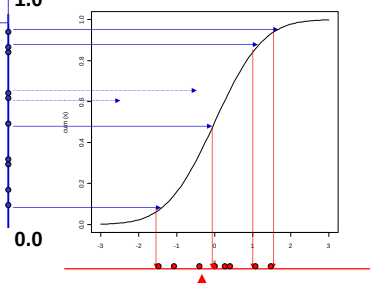
Normal density **Normal distribution function**

Converting uniform sample to Normal sample

Normal distribution function

1.0

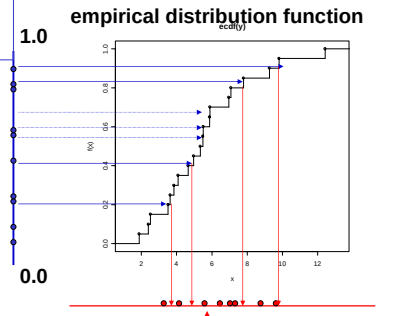
Random sample uniformly distributed on (0,1)

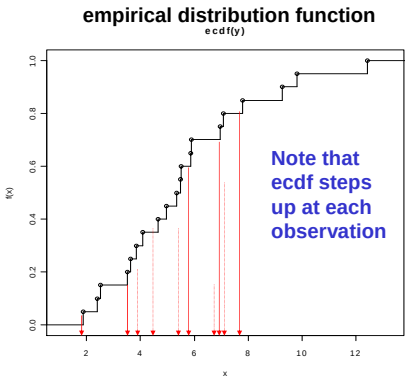


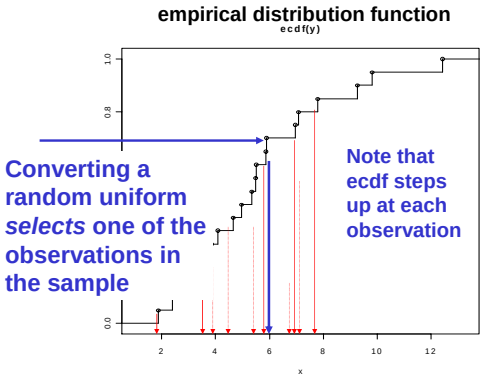
0.0

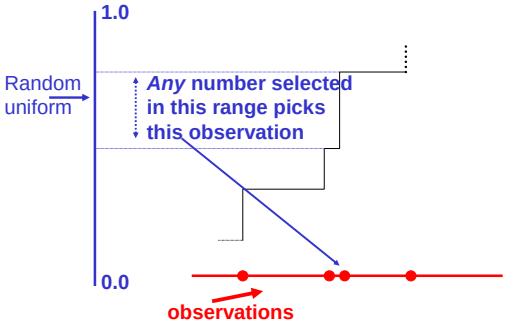
Random sample Normally distributed

Converting uniform sample to empirical sample









- So, if we want to sample from our best estimate of the unknown density we can sample ***with replacement*** from our original data
- In **R** this can be done by `sample(x, replace=T)`
  - takes a sample with replacement of the same size as x
- This is called a ***BOOTSTRAP*** sample

- **Digression**
  - ***BOOTSTRAP:-***
    term invented by Bradley Effron ~1978
    - After the fairy story about the Giant who could pick himself up by his bootstraps
  - He almost decided to call it The Shotgun
    - (You don't have to think to use it!)
  - Had been used many times before but not recognised as a powerful general technique
    - e.g. by D.G. Kendall

55

- Procedure here used a ***histogram*** estimate of the density
  - Could use a ***smoothed*** estimate
    - c.f. kernel density estimate
  - – add a small random amount to each observation when it is selected
    - (easier than integrating the kernel density estimate but not quite the same)

56

- Similar ideas with designed experiments
  - i.e. when we have more than just single sample
  - e.g. two samples A and B
- Permutation and randomization tests
  - If there is no difference between samples then the ***labels*** A and B attached to each observation are arbitrary and it would not matter if we permuted them
    - (i.e. randomly changed A and B)
      - keeping total numbers of As & Bs same

57

- Permutation test:
  - Consider **all** possible arrangements
- Randomization test:
  - Consider a random selection of all arrangements
- Compare value of observed test statistic with that from permutation or randomization distribution

58

- Summary
  - Calculation of p-values of observed test statistics needs mathematics
    - Ok in simple situations
  - Simulation can provide an empirical answer
    - Need to assume we know the distribution
  - Bootstrapping estimates the distribution
    - c.f. density estimation
  - Randomization & permutation tests similar
    - Useful in more structured situations

59

Linear Models & Smooth Regression

- Linear models
- Diagnostics
- Robust regression
- Bootstrapping linear models
- Scatterplot smoothing
- Spline regression
- Non-linear regression

60

- **Linear Models**
  - *Linear* in parameters
    - not necessarily fitting a straight line
  - General **R** statement is `lm(formula)`
    - **`lm`** for **`l`**inear **`m`**odel
  - `formula`:

Dependent variable ~ linear function of independent variables

  - ~ means here “is related to”

- e.g. `lm(time~dist)` fits model
  $time_i = \alpha + \beta\, dist_i + error_i$
  and produces estimates of α and β
- e.g. `lm(time~dist + climb)` fits model
  $time_i = \alpha + \beta_1\, dist_i + \beta_2\, climb_i + error_i$
- `lm()` produces an *object* which can by examined in usual way with `summary()` and other special commands

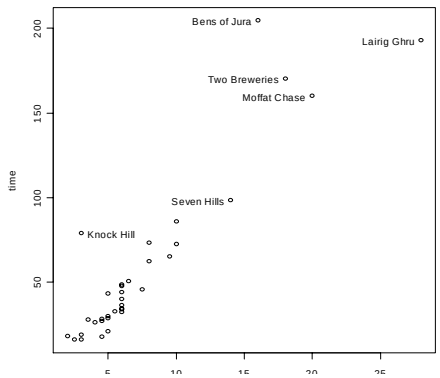
- e.g. `hillslm<-lm(time~dist)`
  - produces object `hillslm`
  - `hillslm$coefficients` gives vector of coefficients
  - `hillslm$fitted` gives fitted values $\hat{\alpha} + \hat{\beta}\, dist_i$
  - `hillslm$residuals` gives vector of residuals
    - i.e. estimates of the errors in model $time_i = \alpha + \beta\, dist_i + errror_i$
    - residual = $time_i$ – $fitted_i$

- Regression Diagnostics
  - analysis of the residuals can indicate whether model is satisfactory:
  - if model is appropriate for the data then
    - Residuals should look like a random sample from a Normal distribution
    - Residuals should be independent of fitted values
  - easy to check these graphically
    - `plot.lm(hillslm)` will give basic checks

- Example: Scottish Hill Races
  - in data set hills in the MASS library

```
> library(MASS)
> data(hills)
> attach(hills)
> plot(dist,time)
> identify(dist,time,row.names(hills))
 [1]  7 11 17 18 33 35
```

  - Note use of `identify()` which allows interactive identification of points
    - good to identify obvious outliers



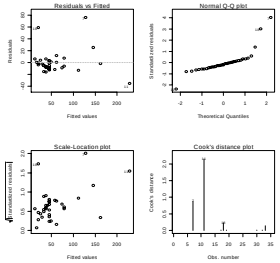
- Now fit model and look at results

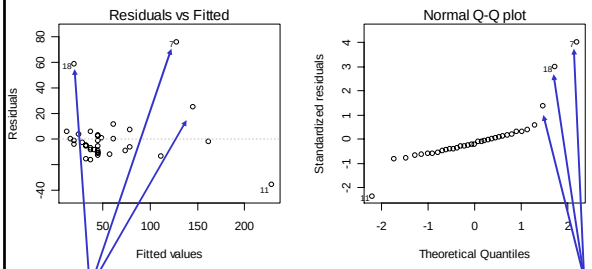
```
> hillslm<- lm(time~dist)
> summary(hillslm)
Call:
lm(formula = time ~ dist)
Residuals:
    Min      1Q  Median      3Q     Max
-35.745  -9.037  -4.201   2.849  76.170
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -4.8407     5.7562  -0.841    0.406
dist          8.3305     0.6196  13.446    6e-15 ***
```

  - Note five summary statistics of residuals and estimates of coefficients with st. errs.
    - Also a lot more detail — more than usually needed on any single occasion

- Look at basic diagnostics with

```
> plot.lm(hillslm)
```





Outliers and so plot does not look random

Points not on a straight line so deviation from Normality

(Other two plots of less interest here)

- Next Steps:
  - Remove outliers from data
    - `> lm(time~dist,data=hills[-7,])`
    - Removes 7th observation
    - `> lm(time~dist,data=hills[-c(7,18),])`
    - Removes both 7th and 18th
  - Use a ***robust method*** for estimating linear relationship which is not so greatly affect by outliers

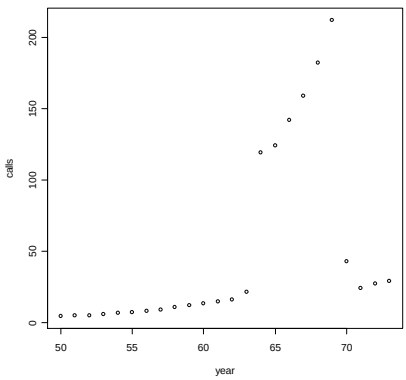
- Robust Regression
  - Available routines:
    - `rlm()` in MASS library and `lqs()` in `lqs` library
  - Example on hills data:

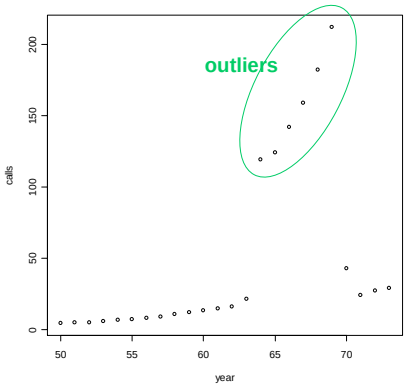
```
hillslm1<-lm(time~dist,
             data=hills[-c(7,18),])
```

    - Gives intercept and slope as –5.81 and 7.91

```
hillsrlm<-rlm(time~dist)
```

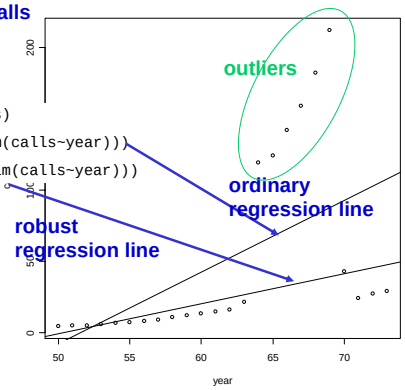
    - Gives intercept and slope as –6.36 and 8.051 (and with smaller standard errors of estimates)





Data: phone calls in Belgium

```
> plot(year,calls)
> lines(abline(lm(calls~year)))
> lines(abline(rlm(calls~year)))
```



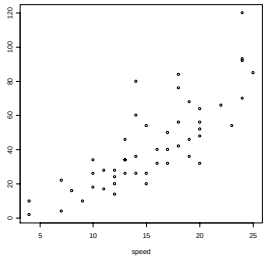
Bootstrapping linear models

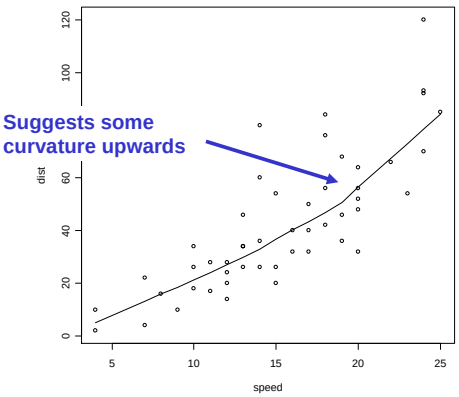
- May not want to assume a Normal error structure and if so then we may want some bootstrap estimate of (say) a confidence interval for the slope
- Not appropriate to sample points $\{(x_i,y_i);i=1,\ldots n\}$
  - Since usually the x-values are *fixed*
- Instead have to bootstrap residuals

- Steps:-
  - Fit regression (with `lm()` or `rlm()` or …)
  - Extract residuals with `obj$residuals`
  - Extract coefficients with `obj$coefficients`
  - Take bootstrap samples of residuals and construct new bootstrap data sets with
    - actual x-values +
    - estimated coefficients +
    - bootstrapped residuals
  - Calculate quantity of interest

Scatterplot smoothing

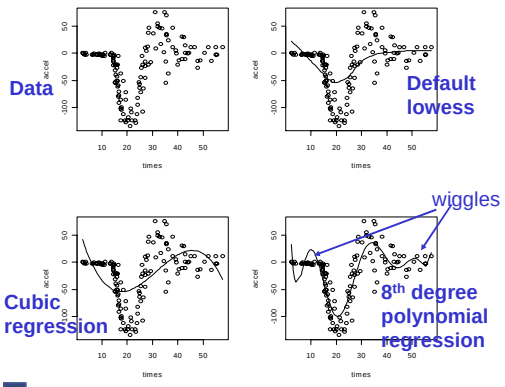
- tool for informal investigation of structure in scatterplot
- can see increase in distance with speed but is this constant or does it tail upwards?
  - use lowess

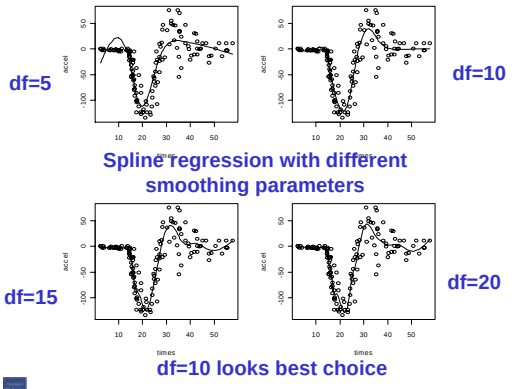




Spline regression

- Polynomial regression often not satisfactory since behaviour of a polynomial depends upon values over its entire range
- Instead, ***spline*** regression uses 'local piecewise polynomials' which adapt to local values better and do not influence distant ones





Non-linear regression

- May be external reasons for specifying a particular *non-linear* model
- e.g of form

$$y = \alpha + \beta x^{-2/\theta}$$

- Can be estimated using routine `nls()` in library `nls` (non-linear least squares)
  - Need to specify *starting values*

Summary

- Seen how regression diagnostics leads to dropping outliers or use of ***robust regression*** methods
- Ideas of bootstrapping linear models
- ***Scatterplot smoothing*** useful informal tool
  - Use of `lowess()`
- Smooth regression with **splines**
- Availability of non-linear regression

Multivariate Methods

- Multivariate data
- Data display
- Principal component analysis
  - *Unsupervised learning technique*
- Discriminant analysis
  - *Supervised learning technique*
- Cluster analysis
  - *Unsupervised learning technique*
    - (Read notes on this)

- Measurements of ***p*** variables on each of ***n*** objects
  - e.g. lengths & widths of petals & sepals of each of 150 iris flowers
- key feature is that variables are **correlated** & observations **independent**

- **Data Display**
  - ♦ Scatterplots of pairs of components
    - Need to choose which components
  - ♦ Matrix plots
  - ♦ Star plots
  - ♦ etc. etc. etc.
- None is very satisfactory when ***p*** is big
  - ♦ Need to ***select*** best components to plot
  - ♦ i.e. need to ***reduce dimensionality***

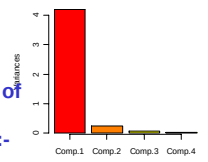
- Digression on **R** language details:
  - ♦ Many multivariate routines in library `mva`
  - ♦ So far only considered data in a *dataframe*
  - ♦ Multivariate methods in **R** often need data in a ***matrix***
  - ♦ Use commands such as
    - `as.matrix(.)`
    - `rbind(.)`
    - `cbind(.)`
  - ♦ Which create matrices (see `help`)

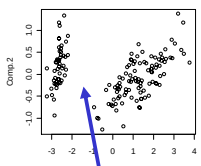
- **Principal Component Analysis (PCA)**
  - ♦ Technique for finding which linear combinations of variables contain most information.
  - ♦ Produces a new coordinate system
    - Plots on the first few components are like to show structure in data (i.e. information)
  - ♦ Example:
    - Iris data


```
> library(mva)
> library(MASS)
> par(mfrow=c(2,2))
> data(iris)
> attach(iris)
> ir<-cbind(Sepal.Length, Sepal.Width, Petal.Length,
+ Petal.Width)
> ir.pca<-princomp(ir)
> plot(ir.pca)
> ir.pc<-predict(ir.pca)
> plot(ir.pca$scores[,1:2])
> plot(ir.pca$scores[,2:3])
> plot(ir.pca$scores[,3:4])
```

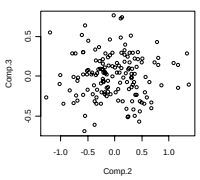
This creates a matrix `ir` of the iris data, performs pca, uses the generic `predict` function to calculate the coordinates of the data on the principal components and plots the first three pairs of components

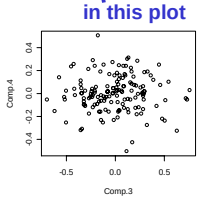
Shows importance of each component:- most information in first component



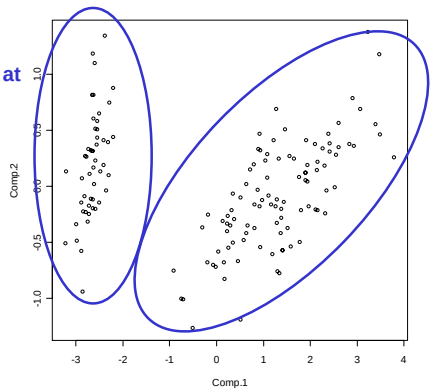


most information in this plot

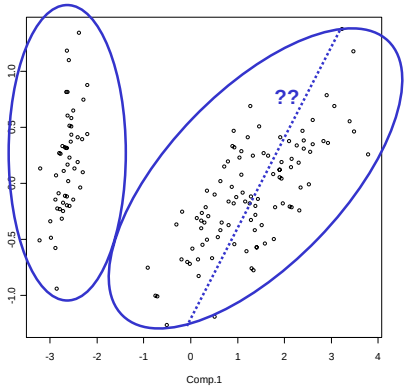




Can detect at least two separate groups in data and maybe……



Can detect at least two separate groups in data and maybe one group divides into two



- Can interpret principal components as reflecting features in the data by examining loadings
  - away from the main theme of course
  - see example in notes.

- Principal component analysis is a useful basic tool for investigating data structure and reducing dimensionality

Discriminant Analysis

- Key problem is to use multivariate data on different types of objects to classify future observations.
- e.g. the iris flowers are actually from 3 different species (50 of each)
- What combinations of sepal & petal length & width are most useful in distinguishing between the species and for classifying new cases

- e.g. consider a plot of petal length *vs* width
  - First set up a vector to label the three varieties as **s** or **c** or **v**

```
> ir.species<-factor(c(rep("s",50),
+ rep("c",50),rep("v",50)))
```

  - Then create a matrix with the petal measurements

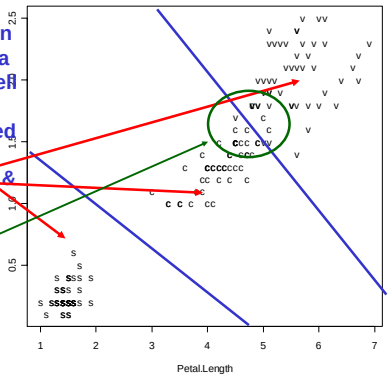
```
> petals<-cbind(Petal.Length,
+ Petal.Width)
```

  - Then plot the data with the labels

```
> plot(petals,type="n")
> text(petals,
+ labels=as.character(ir.species))
```


Informally we can see some separation between the species and if a new observation fell in this region it should be classified as type "s" & here as type "c" & here as type "v"

However, some misclassification here between "c" and "v"



- This method uses just petal length & width
  - makes some mistakes
- Could we do better with all measurements?
- ***linear discriminant analysis (LDA)*** will give the best method when boundaries between regions are straight lines
  - And ***quadratic discriminant analysis (QDA)*** when boundaries are quadratic curves

- How do we evaluate how well LDA performs?
  - Could use rule to classify data we actually have
    - Can use generic function **`predict(.)`**

```
> ir.lda<- lda(ir,ir.species)
> ir.ld<- predict(ir.lda,ir)
> ir.ld$class
  [1] s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s
 [38] s s s s s s s s s s s s s s s c c c c c c c c c c c c c c c c c c c c c c c
 [75] c c c c c c c c c v c c c c c c c c c c c c c c c c c c c c c c c c c c c c
[112] v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v v
[149] v v
Levels: c s v
```


- Or in a table:

```
> table(ir.species, ir.ld$class)
ir.species  c  s  v
         c 48  0  2
         s  0 50  0
         v  1  0 49
```

- Which shows that
  - Correct classification rate 147/150
  - 2 species C classified as V
  - 1 species V classified as a C

- However, this is optimistic ('cheating')
  - Using same data for calculating the `lda` and evaluating its performance
  - Better would be ***cross-validation***
    - Omit one observation
    - Calculate lda
    - Use lda to classify this observation
    - Repeat on each observation in turn
  - Also better would be a ***randomization test***
    - Randomly permute the species labels

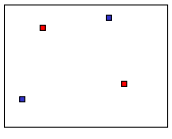
- Use `sample(ir.species)` to permute labels
  - Note:- sampling ***without replacement***

```
> randomspecies<-sample(ir.species)
> irrand.lda<-lda(ir,randomspecies)
> irrand.ld<-predict(irrand.lda,ir)
> table(randomspecies,irrand.ld$class)
randomspecies  c  s  v
            c 29 17  4
            s 17 28  5
            v 17 20 13
```

  - Which shows that only 70 out of 150 would be correctly classified
    - (compare 147 out of 150)

- This could be repeated many times and the ***permutation distribution*** of the correct classification rate obtained.
  - (or strictly the ***randomization distribution***)
- The observed rate of 147/150 would be in the extreme tail of the distribution
  - i.e. the observed rate of 147/150 is much higher than could occur by chance

- **General Comment**
  - If we have **high** number of dimensions & **small** number of points then *always* easy to get ***near perfect*** discrimination
  - A randomization test will show if a high classification rate is a result of a real difference between cases or just geometry.
    - e.g. 2 groups, 2 dimensions
      3 points → always perfect
      4 points → 75% chance of perfect discrimination
    - 3 dimensions always perfect with 2 groups 4 points



- To estimate the true classification rate we should apply the rule to new data
  - e.g. to construct the rule on a random sample and apply it to the other observations

```
> samp<- c(sample(1:50,25),
+ sample(51:100,25), sample(101:150,25))
```

  - `samp` will contain
    - 25 numbers from 1 to 50
    - 25 from 51 to 100
    - 25 from 101 to 150


```
> samp
 [1]  43   7  46  10  19  47   5  49  45  37  33   8  12  28
27  11   2  29   1
[20]  32   3  14   4  25   6  54  92  67  74  89  71  81  97
62  73  93  99  60
[39]  58  70  51  94  83  72  66  59  65  86  98  82 132 101
139 108 138 112 125
[58] 146 103 129 109 124 102 137 121 147 144 128 116 131 113
104 148 115 122
```

- So `ir[samp,]`will have just these cases
  - With 25 from each species
  - `ir[-samp,]`will have the others
- Use `ir[samp,]` to construct the lda and then predict on `ir[-samp,]`


```
> irsamp.lda<-
  lda(ir[samp,],ir.species[samp])
> irsamp.ld<-predict(irsamp.lda, ir[-
  samp,])
> table(ir.species[-samp], irsamp.ld$class)
     c  s  v
  c 22  0  3
  s  0 25  0
  v  1  0 24
```

  - So rule classifies correctly 71 out of 75

- Other examples in notes

- **Summary**
  - PCA was introduced
  - Ideas of discrimination & classification with lda and qda outlined
  - Ideas of using analyses to ***predict*** illustrated
  - Taking random permutations & random samples illustrated
- Predictions and random samples will be used in other methods for discrimination & classification using neural networks etc.

Tree-Based Methods

- Methods for analyzing problems of discrimination and regression
  - Classification & Decision Trees
    - For factor outcomes
  - Regression Trees
    - For continuous outcomes
  - Difference from other methods is in effective display and intuitive appeal

Classification Trees

- Aim is to find a rule for classifying cases
  - Use a step-by-step approach
    - (one variable at a time)
  - Aim is to produce a rule for classifying objects into categories
  - Similar problems of evaluation of performance
    - high dimensions and complicated rules give over-optimistic performance

Example: Iris data

1st: divide on petal length:-

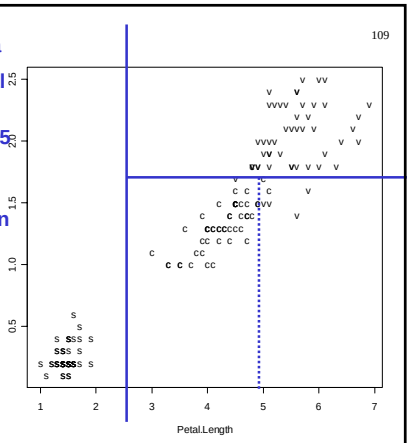
If petal length < 2.5 then type "S"

2nd: petal width

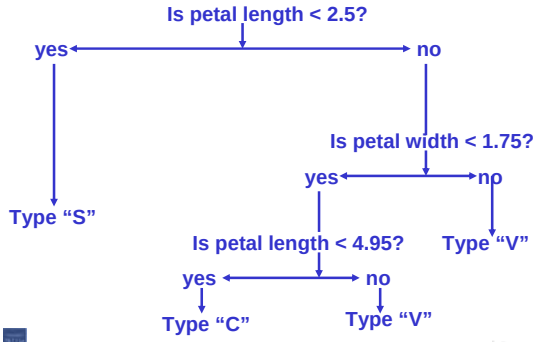
If width < 1.75 then most of type "C"

If length < 4.95 then "C" and if > 4.95 then "V"

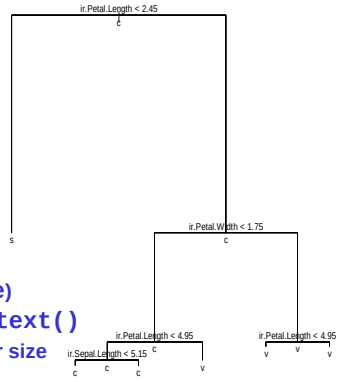
If width >1.75 then "V"



Can display this as a tree:




```
> library(tree)
> ir.tr<-
+ tree(ir.species~ir)
> plot(ir.tr)
> text(ir.tr,
+ all=T,cex=0.8)
```



Note

- **call to `library(tree)`**
- **addition of labels with `text()`**
- **`cex` controls character size**

- Note misclassification rate with this tree is 4/150 or correct rate is 146/150
  - Compare LDA of 147/150
  - Could look at cross-validation method
    - Special routine `tree.cv(.)`
  - Could permute labels
- Note we can grow tree on a random sample of data and then use it to classify new data (as with `lda`)


```
> irsamp.tr<-
+ tree(ir.species[samp]~ir[samp,])
> ir.pred<-predict(irsamp.tr,
+ ir[-samp,],type="class")
> table(ir.pred,ir.species[-samp])
irpred  c  s  v
     c 24  0  0
     s  0 25  0
     v  1  0 25
```

- So correct classification rate of 74/75

- Other facilities
  `snip.tree(.)`
  - Interactive chopping of tree to remove unwanted branches
  - Works in similar way to **`identify()`**
  - Try **`help(snip.tree)`**
  - **`library(help=tree)`** for list of all facilities in library **`tree`**
  - Also **`library(rpart)`**

Similar Methods

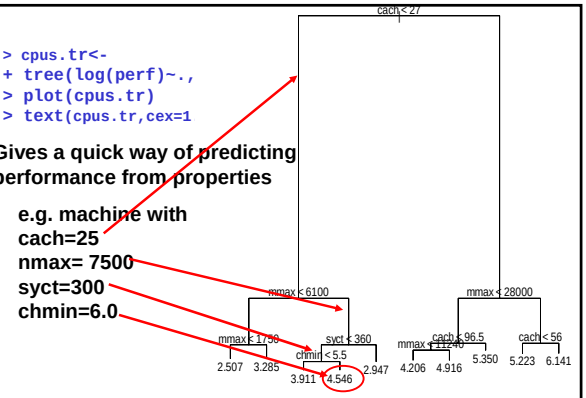
- Decision trees
  - Essentially the same as classification trees
  - See shuttle example
- Regression trees
  - Continuous outcome to be predicted from explanatory independent variables

  Can be
  - continuous
  - ordered factors
  - multiple unordered categories
- Continuous outcome is made 'discrete'
  - makes it similar to classification trees


```
> cpus.tr<-
+ tree(log(perf)~.,
> plot(cpus.tr)
> text(cpus.tr,cex=1
```

Gives a quick way of predicting performance from properties

e.g. machine with
cach=25
nmax= 7500
syct=300
chmin=6.0



Comments on mathematics

- PCA and lda have rigorous mathematical foundation
- Obtained from applications of general statistical theory
- Results similar to Neyman-Pearson Lemma etc., etc.

- Tree-Based Methods **WORK** in practice
  - algorithmic basis instead of mathematical
  - Give good results in some cases when classical methods are less satisfactory

Summary

- Classification & Regression Trees
  - Take one variable at a time
  - Facilities for cross-validation and randomization
  - Variables can be continuous or ordered or unordered factors
  - Facilities for interactive pruning
  - Can be problems with high dimensions and small numbers of cases
  - Theoretical foundation is algorithmic not mathematical
  - They can WORK in practice

Neural Networks

- Technique for discrimination & regression problems
- More mathematical theoretical foundation
- Works well on many practical problems
- Same problems when there are large number of variables and small numbers of observations

Mathematical description

- A '***neural network***' is a particular type of non-linear regression model
- Typically it has very many parameters
- Sometimes even more parameters than observations

Aside:

- Usually in applied statistics more parameters than observation ⇒ **TROUBLE**
- Can be trouble here as well
  - Important to check performance of neural networks with randomization, X-validation etc

121

- Note that the technique was developed by Computer Scientists and terminology may not appear to be standard statistical terms

  e.g.
  - ‘inputs’ ≡ data (independent variables)
  - ‘targets’ ≡ data (dependent variables)
  - ‘outputs’ ≡ predicted values
  - ‘weights’ ≡ unknown parameters
  - ‘training a network’ ≡ estimating unknown parameters

122

- Many of the methods developed for neural networks came from algorithmic ideas.
- Many also have general statistical justification
- Many are very successful
- Some problems remain
  - e.g. *over-parameterization*
    - (no free lunches even with neural nets)

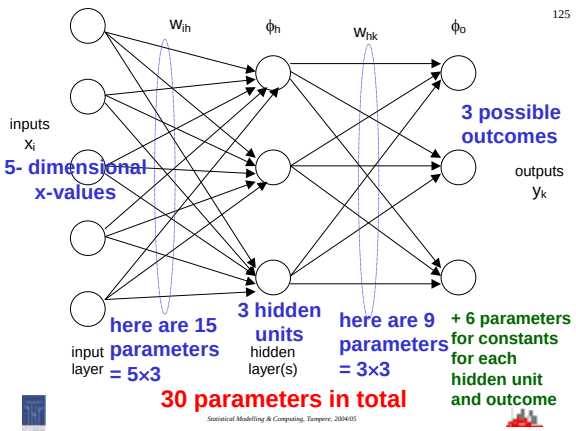
123

- Outline of steps
  - For each value of **x** there is a target **y**
    - e.g. x=(sepal.length, ……,petal width)
    - y = type ”S” – coded as (1,0,0)
      or type “C” – coded as (0,1,0)
      or type “V” – coded as (0,0,1) say
  - Calculate several different linear functions $a_i(x)$ of x using weights (parameters) $w_{i1},w_{i2},w_{i3},w_{i4}$ for a suitable number of functions or ‘*units in a hidden layer*’
  - Use an *activation function* $\phi(.)$ on each $a_i(x)$ to get $\phi(a_i(x))$

124

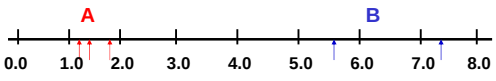
- Take another linear function of the $\phi_i(a_i(x))$ using weights (or parameters) & feed through another *activation function* to get each element of output **y**
- i.e. three separate (activated) linear functions to get a value (1,0,0) or …. or (0,0,1)
- Now estimate parameters w_{ij} to make outputs match targets as closely as possible e.g. *least squares minimization*
- Evaluate rule and perhaps *tune network* by changing number of hidden units

125



126

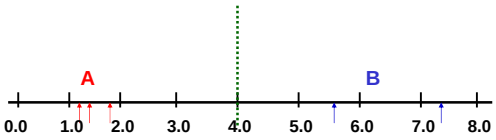
- **Simple example:** (see P142)
  - Univariate data; 2 categories A and B
  - 3 samples from A and 2 from B
  - A samples: 1.1, 1.7, 1.3
  - B samples: 5.6, 7.2



127

- Easy to discriminate between A & B with a simple rule such as

  If x < 4.0 then classify as A, else B



However, neural nets work differently…

128

- Aim is to find two formulas $f_1(x)$ and $f_2(x)$ such that

  $f_1(x) \approx 1$ when x is of type A and ≈ 0 when x is of type B

  $f_2(x) \approx 1$ when x is of type B and ≈ 0 when x is of type A

So want $f_1(x) \approx 1$ for x= 1.1, 1.3, 1.7

≈ 0 for x= 5.6, 7.2

Impossible with a linear function of x

129

- **Neural net solution** with one hidden layer of 2 units and logistic activation functions
  - Calculate, for i = 1 and 2 $g_i(x) = \exp\{b_i + w_i x\}/[1+\exp\{b_i + w_i x\}]$
  - Calculate, for i = 1 and 2 $f_i(x) = \exp\{c_i+w_{1i}g_1(x)+w_{2i}g_2(x)\} / [1 + \exp\{c_i+w_{1i}g_1(x)+w_{2i}g_2(x)\}]$
  - This has 10 unknown parameters
  - Estimate these by minimizing $(f_1(1.1) -1)^2+\ldots\ldots\ldots\ldots\ldots+(f_2(7.2) -1)^2$

130

- Note, 10 parameters but only 5 observations, though each observation used twice in minimization since we have terms $(f_1(1.1) -1)^2$ and $(f_2(1.1) -0)^2$
- Surprisingly this works
  - Also solution does not seem to be ***data dependent***
    - i.e. it does not seem to be very dependent on the particular data we use.

131

- Implementation in **R**:

```
> nick<-
+
  data.frame(x=c(1.1,1.7,1.3,5.6,7.2,8.1,1.8,3.0)
  ,
+ targets=c(rep("A",3),rep("B",2),rep("U",3)))
> attach(nick)
```

Sets up a data set with values for training samples for A and B and test samples for U (8.1, 1.8, 3.0)

```
> nick.net<-
  nnet(targets~.,data=nick[1:5,],size=2)
```

Calculates a neural net from first 5 data points (i.e. the *training data*)

132

```
> predict(nick.net,nick[1:5,])
              A            B
1 1.000000e+00 2.286416e-18
2 1.000000e+00 3.757951e-18
3 1.000000e+00 2.477393e-18
4 1.523690e-08 1.000000e+00
5 2.161339e-14 1.000000e+00
```

Uses the estimated functions and evaluates them for the training data:

Note that $f_1(x)=1$ & $f_2(x)=0$ for x= 1.1, 1.7 & 1.3 and $f_1(x)=0$ & $f_2(x)=1$ for x=5.6 and 7.2

```
> predict(nick.net,nick[1:5,],type="class")
[1] "A" "A" "A" "B" "B"
```

Checks that the categories are correct using the type="class" option

```
> predict(nick.net,nick[6:8,])
             A            B
6 1.364219e-15 1.000000e+00
7 1.000000e+00 4.659797e-18
8 1.000000e+00 1.769726e-08
```

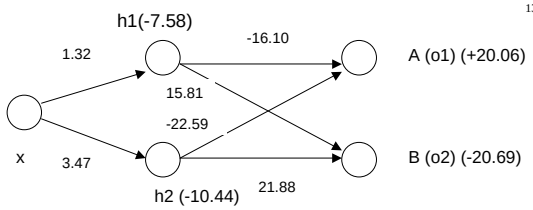
Looks at numerical predictions on test data U


```
> predict(nick.net,nick[6:8,],type="class")
[1] "B" "A" "A"
```

Gives classifications for new data.

```
> summary(nick.net)
a 1-2-2 network with 10 weights
options were - softmax modelling
 b->h1 i1->h1
 -7.58   1.32
 b->h2 i1->h2
-10.44   3.47
 b->o1 h1->o1 h2->o1
 20.06 -16.10 -22.59
 b->o2 h1->o2 h2->o2
h1(-7.58)-20.69  15.81  21.88
```

Estimates of parameters in the functions $g_i(.)$ and $f_i(.)$



Gives a pictorial representation of the two functions $f_1(.)$ and $f_2(.)$ with values of the estimates of the parameters

- **Notes**
  - Sophisticated numerical optimization in calculations in **R**
    - some parameters to `nnet(.)` control this
  - key ***statistical*** parameter in is **size**
    - controls number of units in hidden layer
    - i.e. number of parameters to be estimated
  - increasing **size** gives a better fit
    - does well on training data

  but may make the prediction less reliable on new data

- Notes (continued)
  - i.e. may be dangers of ***overfitting***
  - aim is to get a model with as few parameters as possible that still performs well
- **Strategy:**
  - fit model & investigate statistical properties
    - permutations
    - random sampling
    - applications to new data

- Lecture notes give some examples of this approach on iris data and data set *book*
  - NB 'book' is just a code name & the variable names are not available
    - (current research data set with a company)
- **NOTE:-**
  - **Output in notes has been edited**
  - **Not all of the R commands are given**
    - **should be sufficient to see the type of approach used**

139

- **Summary**
  - Very simple introduction to neural networks
    - Many other types of networks available
    - References by Ripley (1996) & Bishop (1995)
  - Technique for classification based on constructing numerical formula
    - Can also be applied to regression problems
      - (compare tree-based methods)
  - General statistical principles of high number of parameters and overfitting
    - Less serious than could be expected
    - But still a problem to consider

140

- Final Comments & Summary
  - Introduction to the language **R**
    - Provides facilities for learning new statistical techniques in all areas
    - All new statistical developments in the next few years will be available in **R**
  - Ideas of what to do if assumption of Normality is not sensible
    - Ideas of **robust methods**
    - Ideas of cross-validation
    - Ideas of random resampling and permutations
    - **Bootstrapping**

141

- Final Comments & Summary (ctd)
  - Regression methods
    - Interactive identification of points
    - Regression diagnostics
    - Robust regression
  - Multivariate methods
    - Based on traditional mathematical & statistical theory
    - Problems of discrimination
    - Methods for evaluating performance based on random resampling, permutation, etc

142

- Final Comments & Summary (ctd)
  - Other methods of classification etc
    - Based on algorithmic ideas
    - Tree-based methods
    - Neural networks
  - Evaluate statistical properties by statistical experiment
    - Random resampling, permutation etc etc.
- **Aristotle:** for the things we have to know before we can do them, we ***learn*** by doing them
- i.e. **TRY IT & SEE WHAT HAPPENS**

143

144