Elements of Statistical Learning: Lecture 1. Refresh of Linear Regression Models

Guy Nason[1]

Department of Mathematics
Imperial College

Reminder: Properties of Estimators

An estimator $\hat{\theta}$ of θ is unbiased if and only if $\mathbb{E}(\hat{\theta}) = \theta$ and $\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$. The variance of an estimator is $\mathbb{E}[\{\hat{\theta} - \mathbb{E}(\hat{\theta})\}^2]$.

The mean-squared error (MSE) of an estimator is $\mathbb{E}\{(\hat{\theta} - \theta)^2\}$.

$$
\begin{aligned}
\text{MSE}(\hat{\theta}) &= \mathbb{E}[\{\hat{\theta} - \mathbb{E}(\hat{\theta}) + \mathbb{E}(\hat{\theta}) - \theta\}^2] \\
&= \mathbb{E}[\{(\hat{\theta} - \mathbb{E}(\hat{\theta})\}^2] + 2\mathbb{E}[\{\hat{\theta} - \mathbb{E}(\hat{\theta})\}\{\mathbb{E}(\hat{\theta}) - \theta\}] \\
&\quad + \mathbb{E}[\{\mathbb{E}(\hat{\theta}) - \theta\}^2] \\
&= \text{var}(\hat{\theta}) + 2\{E(\hat{\theta}) - \theta\}\underbrace{[\mathbb{E}\{\hat{\theta} - \mathbb{E}(\hat{\theta})\}]}_{\to 0} + \text{bias}(\hat{\theta})^2 \\
&= \text{var}(\hat{\theta}) + \text{bias}(\hat{\theta})^2,
\end{aligned}
$$

and $\text{MSE}(\hat{\theta}) = \text{var}(\hat{\theta})$ for an unbiased estimator.

Univariate (Simple) Linear Regression

Example.
Suppose you are interested in the relationship between heights (in cm) and weights (in kg) of a set of children in a particular school class.

You might be interested in generic questions such as 'what is the typical weight of a child *given its height is X cm*?

Or, what is the 'typical range'?

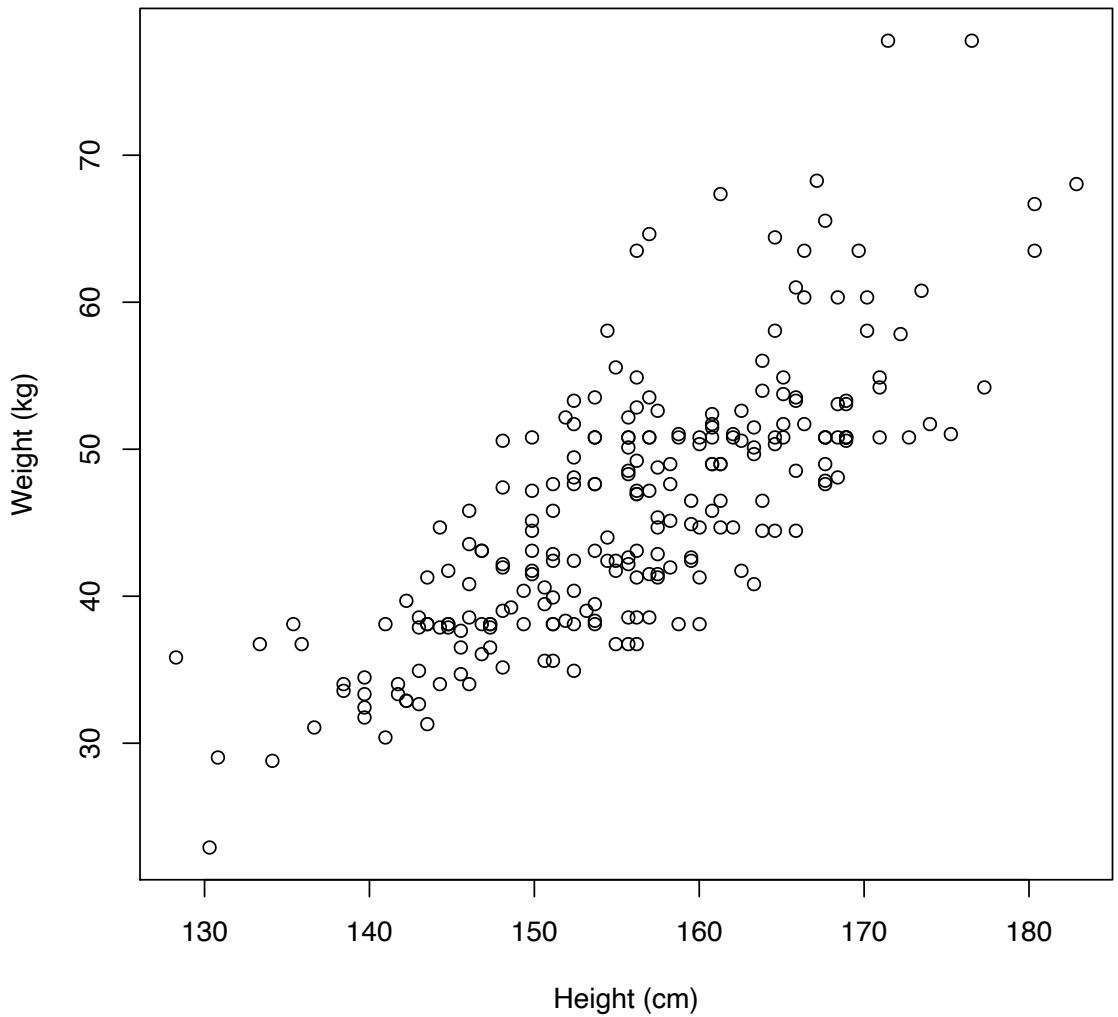
How are we to answer this question?

The first thing to do is carry out some EDA: exploratory data analysis or IDA (initial data analysis), in other words, a plot.

This is to discover whether a linear relationship is tenable.

Lewis and Taylor (1967) Child Height-Weight Data

On 237 children/teenagers ranging in age from 11.5 to 21 years.



Assumptions behind linear model

A very simple model is

$$W_i = a + bH_i + \epsilon_i, \tag{1}$$

where H_i, W_i are the height and weight of the ith child with $i = 1, \ldots, n$, $n = 237$ is the number of children.

The assumptions are vital. They are:

- the model is linear in the parameters, (a, b)
- the errors $\{\epsilon_i\}_{i=1}^n$ are independent random variables
- the errors are identically distributed.

Usually $\mathbb{E}(\epsilon_i)$ is assumed $= 0$.

The last assumption means that $\text{var}(\epsilon_i)$ is constant over i, finite, and can write $\text{var}(\epsilon_i) = \sigma^2$.

IID = independent and identically distributed.

Model Fitting

We NEVER observe the values of ϵ_i, but we believe they exist.

We do see actual data $(h_i, w_i)_{i=1}^n$. One can form a set of residuals

$$e_i(a, b) = w_i - a - bh_i, \quad i = 1, \ldots, n.$$

Each residual is a function of (a, b).

A good fit is when all the residuals are small.

A measure of 'overall smallness' can be

$$R(a, b) = n^{-1} \sum_{i=1}^{n} e_i^2(a, b), \tag{2}$$

the residual sum of squares.

Least Squares Fitting

The least-squares estimators $(\tilde{a}, \tilde{b})$ of (a, b) are obtained by minimising $R(a, b)$ over $(a, b) \in \mathbb{R}^2$.

The estimates can be obtained using simple calculus (differentiate $R(a, b)$ wrt (a, b) and set derivative equal to zero, etc.) and are given by

$$\tilde{b} = \frac{n \sum h_i w_i - (\sum h_i)(\sum w_i)}{n \sum h_i^2 - (\sum h_i)^2} \tag{3}$$

and

$$\tilde{a} = n^{-1} \left(\sum_{i=1}^{n} w_i - \tilde{b} \sum h_i \right). \tag{4}$$

Exercise: Homework Sheet 1 for derivation.

Implementation in R

Suppose the vectors `height` and `weight` contain the 237 heights and weight observations.

We can then put them in a data frame and fit a linear regression by

```
hwData <- data.frame(height=height, weight=weight)
hwModel <- lm(weight ~ height, data=hwData)
hwModel
Call:
lm(formula = weight ~ height, data = hwData)

Coefficients:
(Intercept)       height
   -60.3237       0.6818
```

Interpretation of Output: Anova

Analysis of variance table

```
> anova(hwModel)
Analysis of Variance Table

Response: weight
           Df  Sum Sq Mean Sq F value    Pr(>F)
height      1 11018.7 11018.7  353.14 < 2.2e-16 ***
Residuals 235  7332.6    31.2
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ?
```

The F-statistic p-value is extremely small (smaller than 5%, 1% or even 0.1%). Hence, have strong evidence that supports the hypothesis that height has linear exploratory power for weight (but, of course, that is evident from the plot).

Interpretation of Output: summary

Coefficients and their standard errors: very small p-vals

```
> summary(hwModel)
Call:
lm(formula = weight ~ height, data = hwData)

Residuals:
     Min       1Q   Median       3Q      Max
-10.6832  -4.0278  -0.6629   3.0895  21.2127

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -60.32371    5.66705  -10.64   <2e-16 ***
height        0.68184    0.03628   18.79   <2e-16 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 5.586 on 235 degrees of freedom
Multiple R-squared:  0.6004,Adjusted R-squared:  0.5987
F-statistic: 353.1 on 1 and 235 DF,  p-value: < 2.2e-16
```

Model Checking

Fitting the model on its own is NOT enough.

We need to check, as far as we can that the model assumptions are satisfied.

We can do this via residuals.

Define the ith fitted value

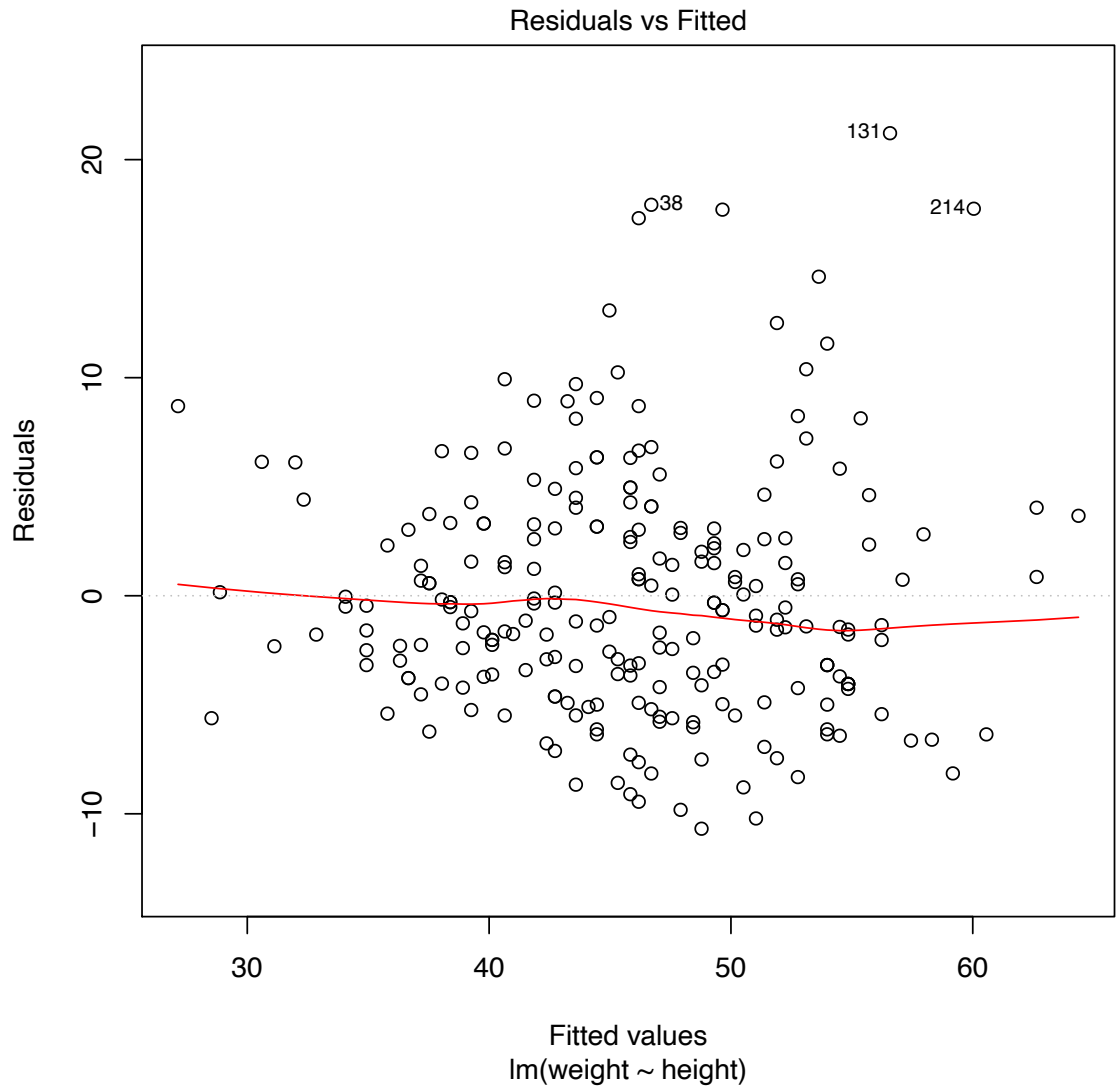
$$\hat{w}_i = \tilde{a} + \tilde{b}h_i,$$

for $i = 1, \ldots, n$. Then the ith residual is given by

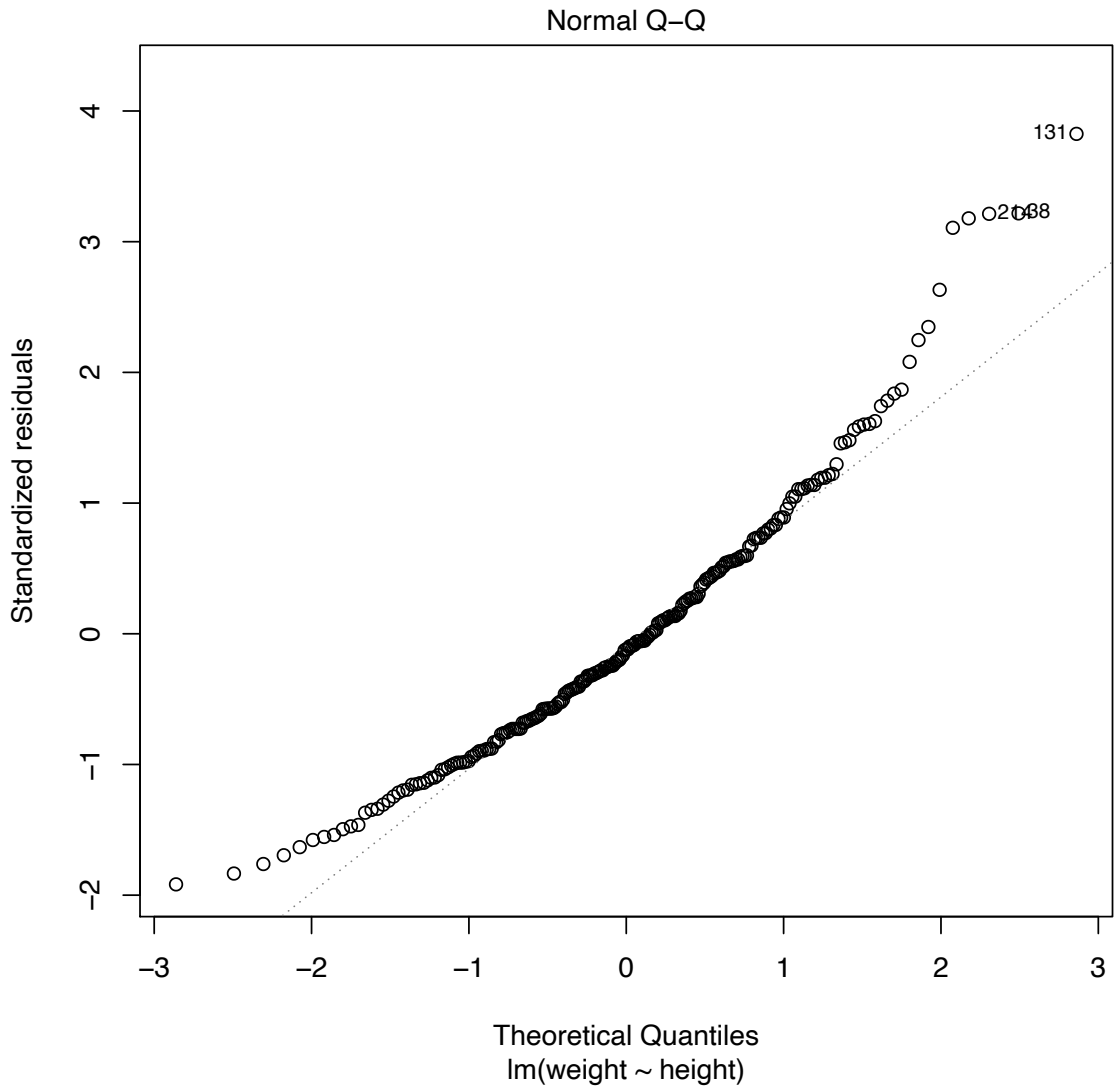
$$\hat{e}_i = \hat{w}_i - w_i.$$

Model checking using `plot.lm()`, i.e. `plot(hwModel)`

Plot of Residuals

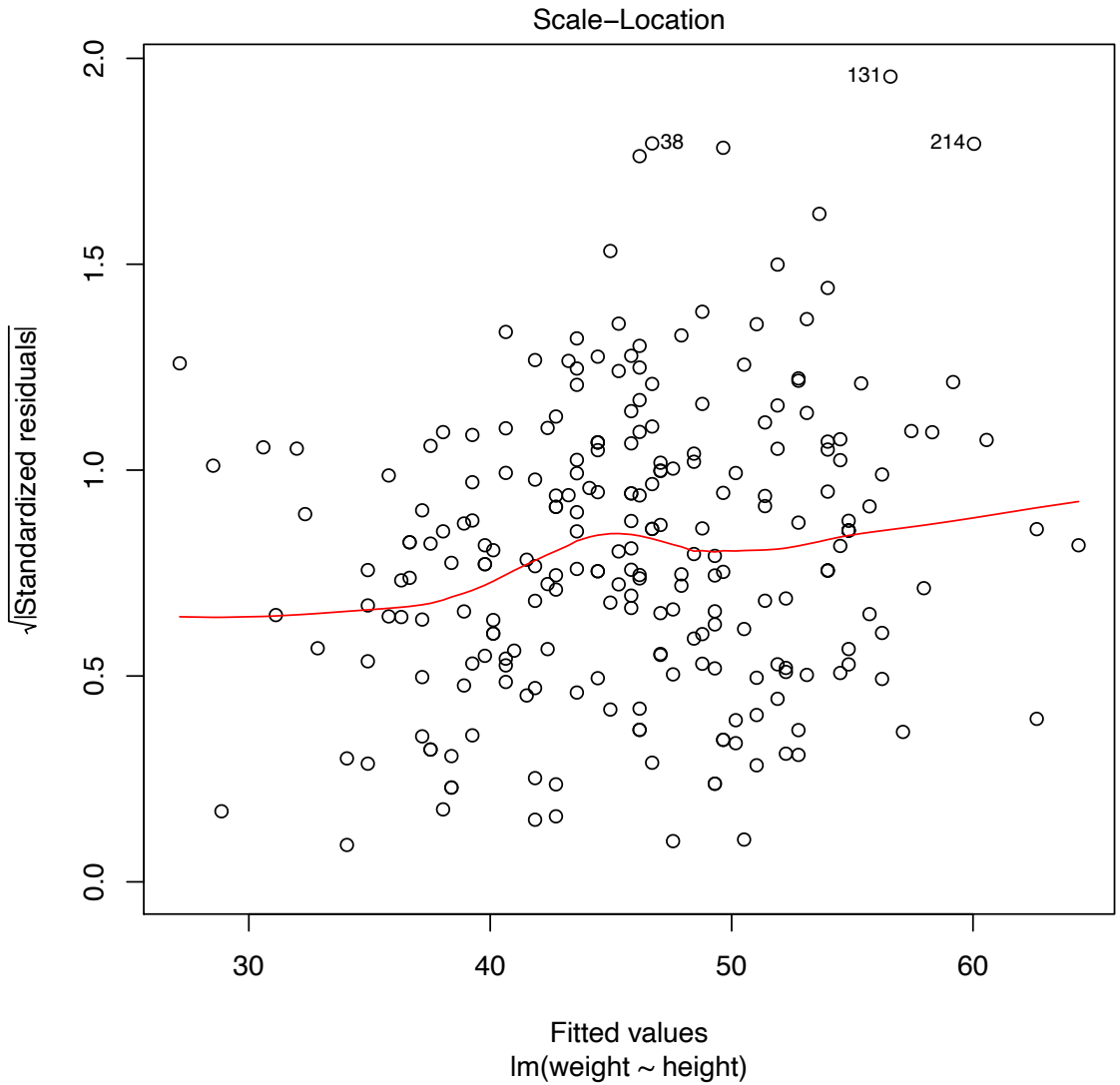


QQ Plot of Residuals



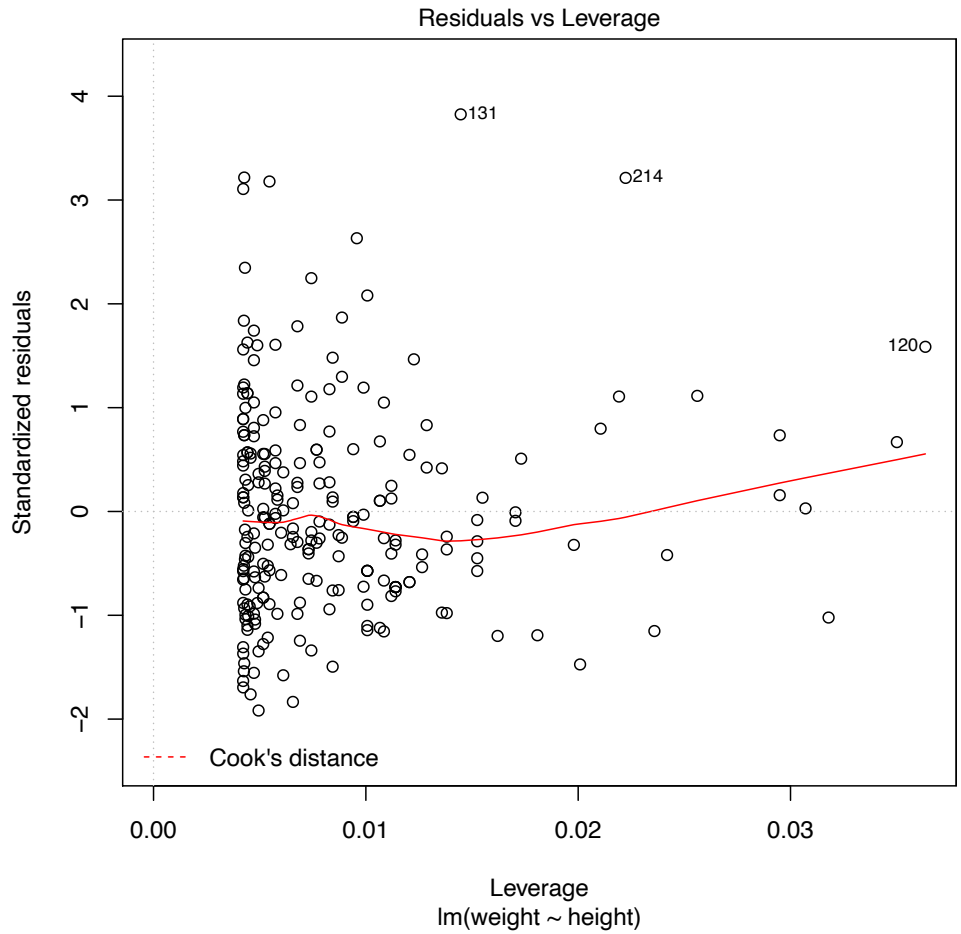
Standardised Residuals against fitted values

$\hat{e}_i / \text{s.d.}(\hat{e})$



Standardised Residuals against Cook's distance

Cook's distance = how much fit changes by omitting point



See also `http://strata.uga.edu/8370/rtips/regressionPlots.html`

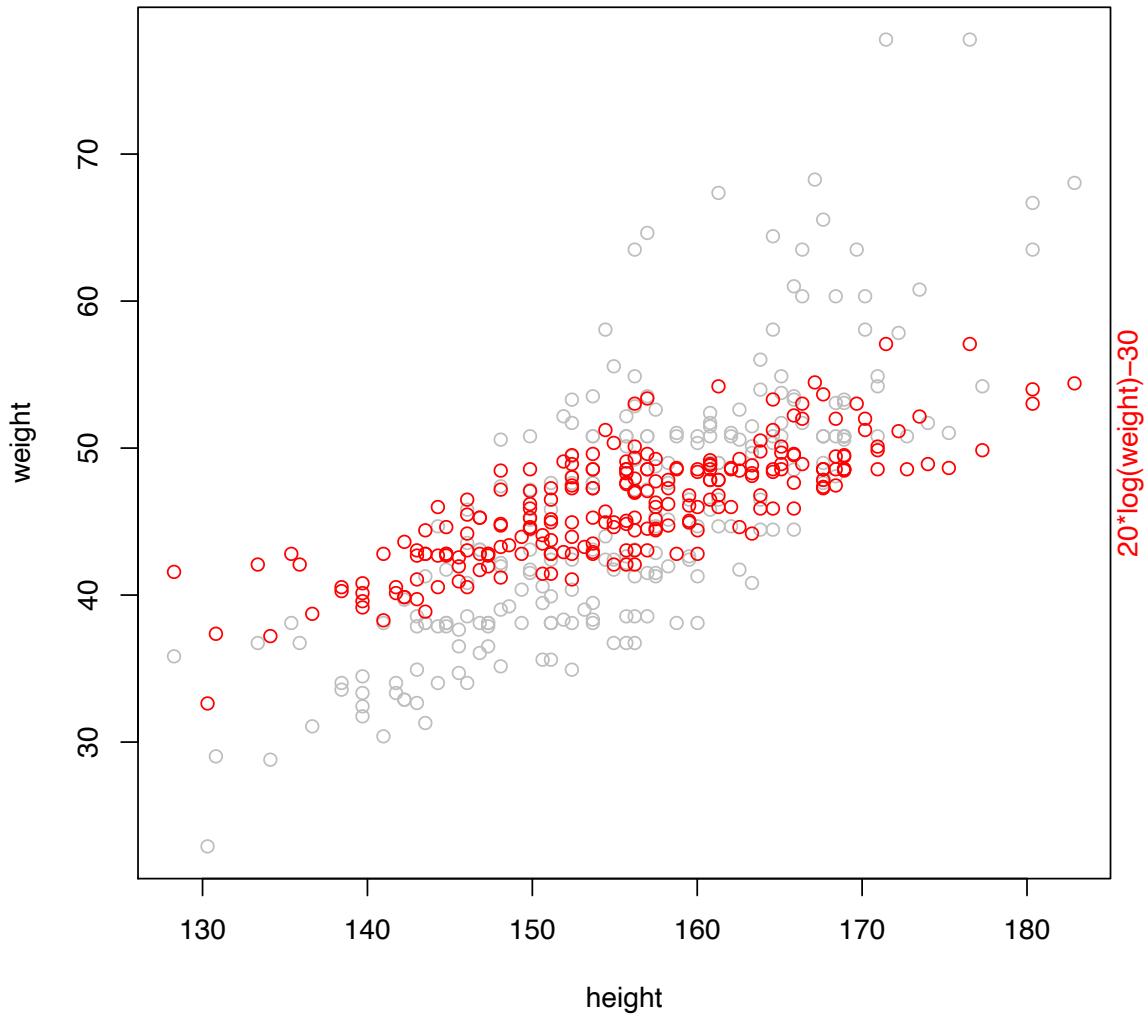
Modelling Cycle

1. fit model;
2. check residuals;
3. If lack of fit: address (transformations, remove outliers, different model) then return to 1., otherwise exit.

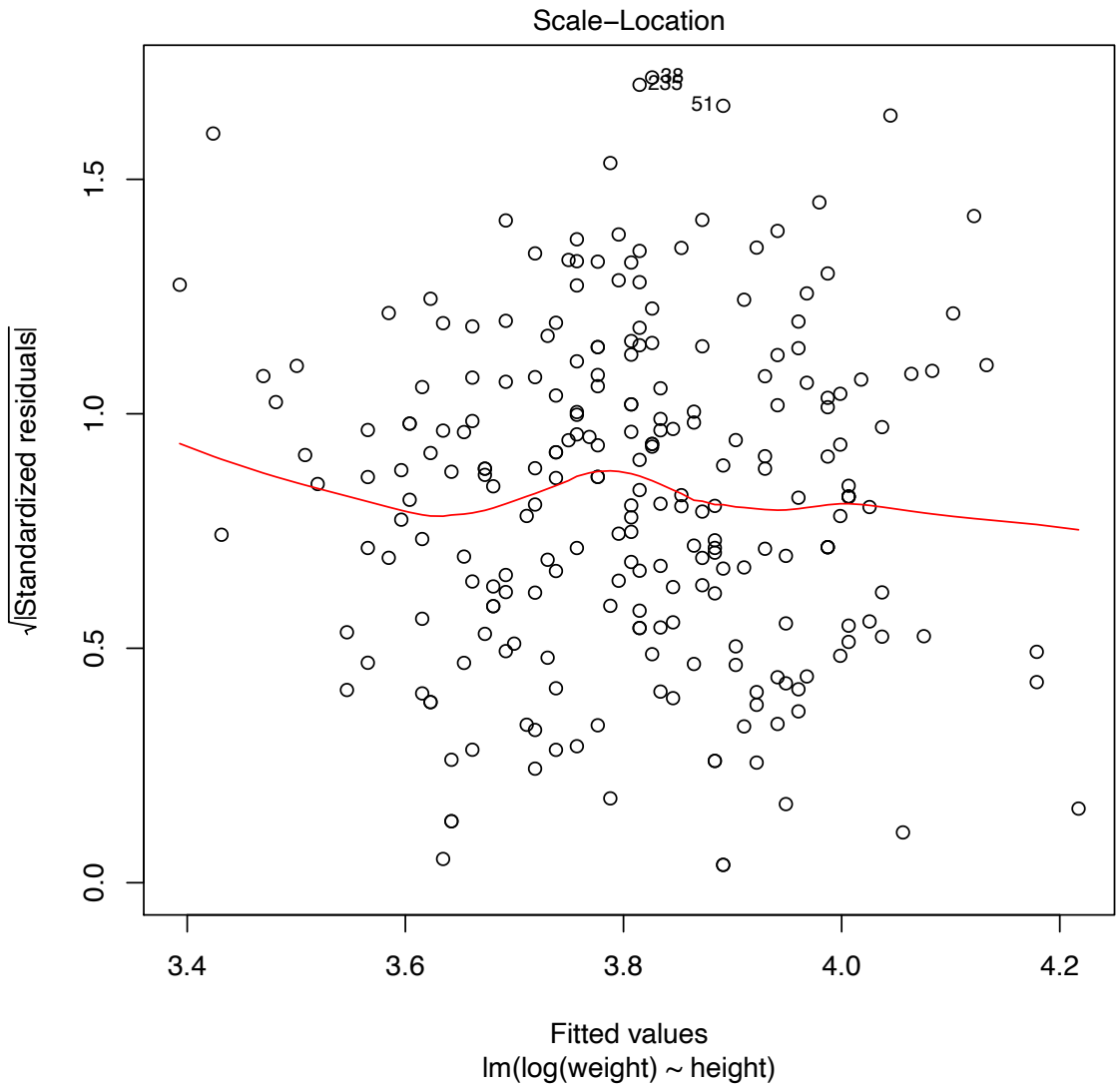
For our data it looks like variance is higher on the right hand side.

So, experiment with log transform, i.e. `weight -> log(weight)`

Log weight against height (variance stabilisation)



Standardised Residuals against fitted values (for log)

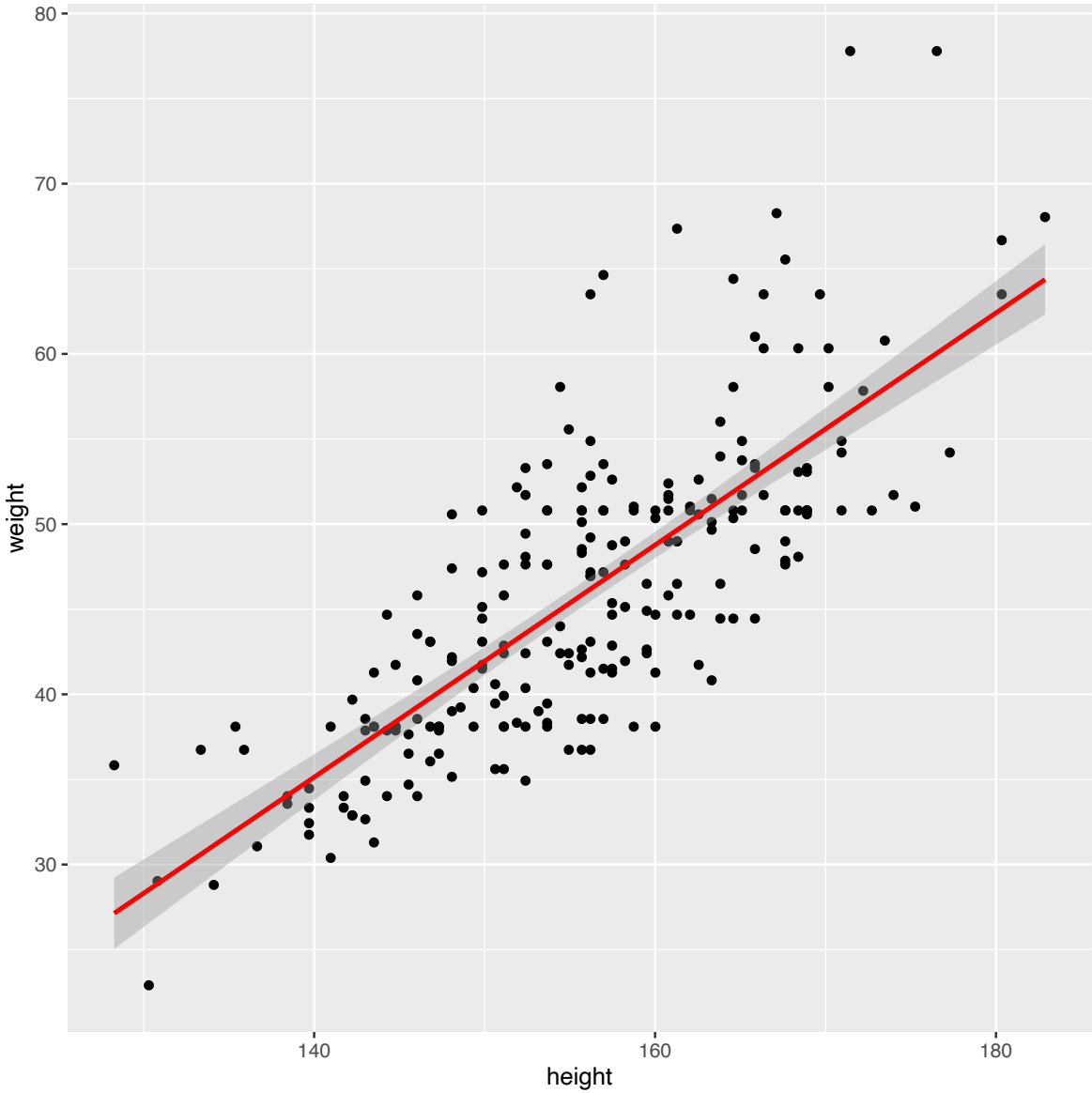


Nice plot using ggplot2 library

```
> library("ggplot2")
> ggplot(hwData, aes(x=height, y=weight)) +
    geom_point() +
    stat_smooth(method="lm", col="red")
```

The grey belt around the red line is the 95% confidence band for the regression line.

ggplot version of the regression



Variants: Maximum Likelihood (ML)

We can additionally attach a probability distribution $F(x)$, or density, $f(x)$ to the errors.

So ϵ_i is an independent draw from F.

This means that W_i has the distribution $F(w - a - bh_i)$.

The likelihood of (a, b) given data $\mathcal{D} = (h_i, w_i)_{i=1}^n$ is

$$L\{(a,b)|\mathcal{D}\} = f(\mathcal{D}|(a,b)) \quad (5)$$

$$= \prod_{i=1}^{n} f\{(h_i, w_i)|(a,b)\} \quad (6)$$

Then the maximum likelihood estimators are given by $(\hat{a}, \hat{b}) = \text{argmax}_{(a,b)} L\{(a,b)|\mathcal{D}\}$.

Variant: ML Gaussian

Let's assume data suggest that the errors in (1) are Gaussian.

[Not a bad starting assumption if there are no obvious outliers, or skew, or data is not low-valued count data.]

Then $\epsilon_i \sim N(0, \sigma^2)$, $i = 1, \ldots, n$, IID and
$f(x) = (2\pi\sigma^2)^{-1/2} e^{-x^2/2\sigma^2}$.

Hence,

$$L\{(a, b)|\mathcal{D}\} = (2\pi\sigma^2)^{-n/2} \exp\left\{(2\sigma^2)^{-1} \sum_{i=1}^{n} (w_i - a - bh_i)^2\right\}$$

ML estimators can be obtained by maximising the log-likelihood, equivalent to (2), and then they are equivalent to least squares.

Variant: Heavy tails, skew

If the plot reveals outliers, this is an indicator that the Gaussian distribution is *possibly* not appropriate. Similarly if skew.

In which case, swap the error distribution from Gaussian to something like Student's t on a low number of degrees of freedom (e.g. 3, which still means ϵ has a mean and finite variance.)

Often skew and outliers become more obvious on examination of the residuals from a Gaussian fit.

The Q-Q plot is effective at picking up departures from a Gaussian model.

Correlated Errors

The simple model (1) is also invalidated when the errors are correlated, e.g. $\text{cor}(\epsilon_i, \epsilon_j) \neq 0$. Often this can be detected by lagged scatter plots or autocorrelation plots (using `acf` in R).

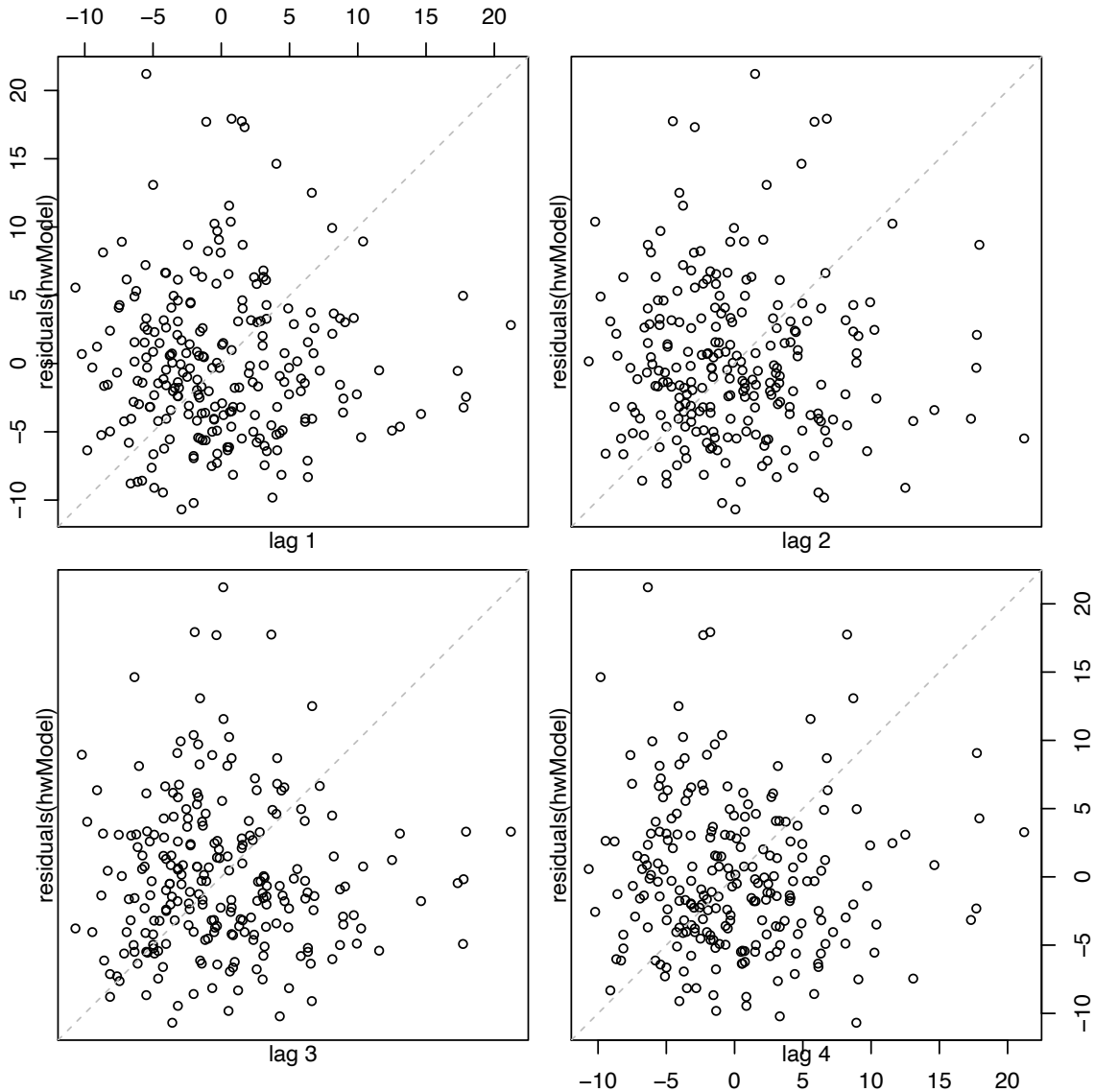
Lagged scatterplot plot of vector v_t at lag τ plots $(v_t, v_{t+\tau})$ and indicates how much association is between v_t and $v_{t+\tau}$.

When (horizontal) explanatory variable is time, or has meaningful ordering, then can also use the *autocorrelation function* to assess correlation between residuals.

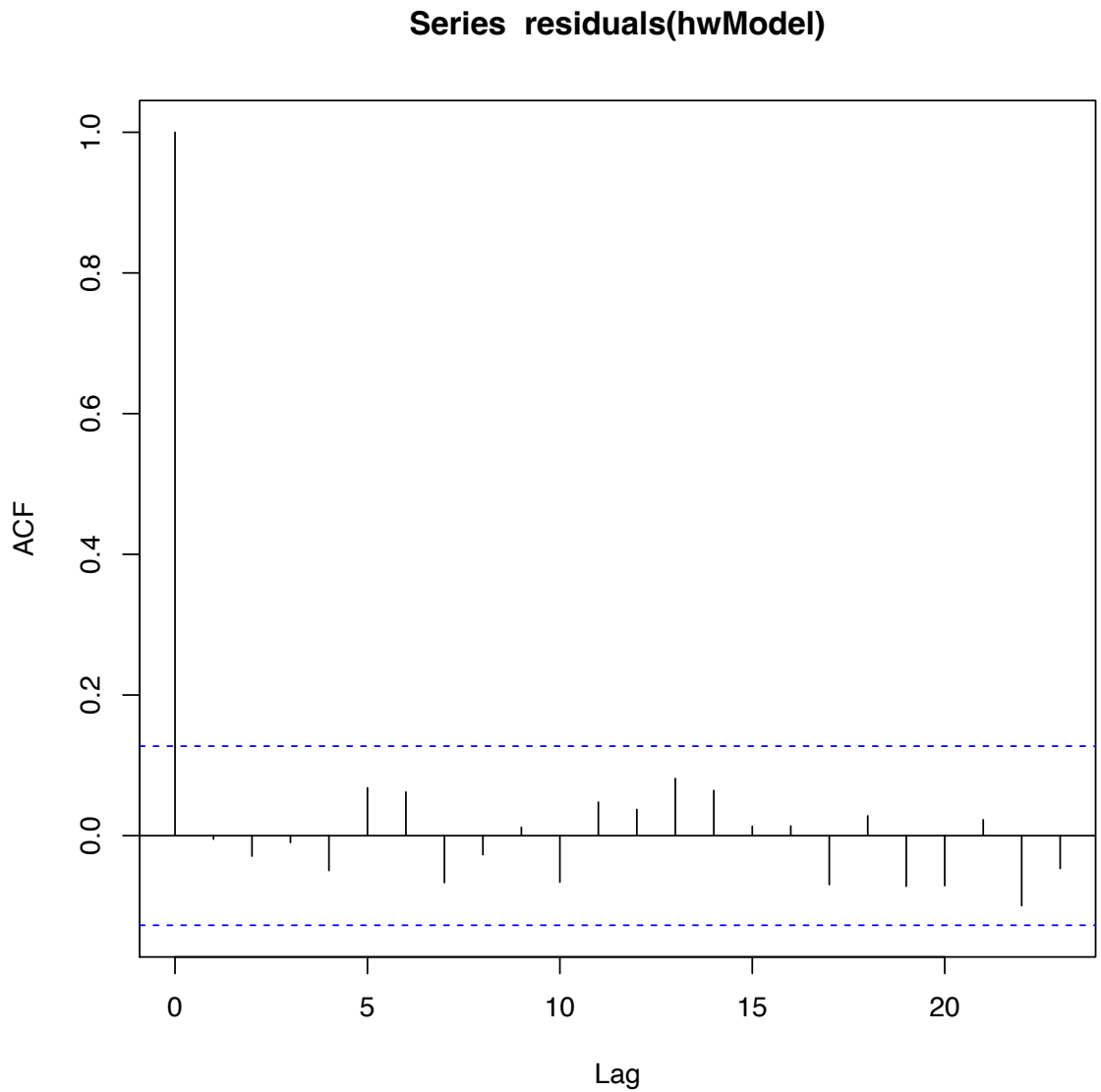
Here, $r_\tau = \text{cor}(v_t, v_{t+\tau})$. The Durbin-Watson statistic can be used to statistically test for the presence of autocorrelation at lag 1.

Correlation can be important, +ve correlation can lead to parameter estimate standard error variance underestimation, causing parm estimates to appear more significant than they really are, and suggest an effect when there is none.

Lagged Scatterplot



Autocorrelation function estimate on residuals



Linearity Assumption

The linear model might not be appropriate, and this should show up in the residual plot. For example, might be quadratic structure, or more complex, which we'll talk about in later blocks.

Measurement Error

Often, we don't observe the exact variables H_i, W_i, but we observe the variable we want contaminated by measurement error.

E.g. in air pollution monitoring the exact traffic flow might be τ_i but you actually measure T_i and we assume $T_i = \tau_i + \nu_i$. Similarly, we measure pollutant level, P_i is a contaminated version of the true level, ρ_i with $P_i = \rho_i + \eta_i$, where ν_i, η_i are mutually iid random variances with zero mean and some variances.

We'd like to investigate the relationship between τ_i and ρ_i, but we can't observe them directly.

The regression of P_i on T_i is different to that of ρ_i on τ_i, but there are methods to estimate the one between the true values from the observed values.

Bayesian view

Assume that the parameters have a distribution themselves.

Before we see any data: elicit prior distributions for (a, b).

After data observation, form likelihood (above) and update our prior distributions for (a, b) into posteriors using Bayes' theorem.

Elicit is an interesting word. Do you really have strong knowledge about the parameters (a, b)?

Sometimes you do. E.g. strong domain knowledge. A doctor might have very strong opinions on the weight distribution of children, and know this very well. Personal/subjective probability.

Sometimes, maths supplies incredibly strong knowledge to enable the Bayesian approach (e.g. Bayesian wavelet shrinkage, later).

Different types of variables

Wish to do regression/modelling with different kinds of variables:

nominal separate categories, not ordered (e.g. color, taste, political preference)

ordered categories categories, that can be ordered (e.g. level of education [post-school, GCSE, A Level, BSc, MSc, PhD], tax status: no tax, basic rate, higher rate)

interval numerical scale, difference is meaningful, zero arbitrary (e.g. temperature, difference in temperates mean something, zero arbitrary — e.g. Fahrenheit zero is different to Celsius) Can't say 30° is twice the temperature of 15°.

ratio as interval, but absolute zero makes sense. E.g. weight, height, amount of money in your bank account, you'd prefer to have £200 rather than £100.

Methods to deal with all of these exist

Reasons for Modelling

In general, you might wish to gain some scientific understanding of the relationship between two variables.

Given a new child & new height: want to predict their weight.

Then, you can compare their actual weight with the predicted weight and "take a view".

Comparisons almost certainly would want to take account of the variability around the prediction. MOSTLY VARIATION IS UNDERAPPRECIATED.

You might wish to *extrapolate*, that is predict a weight from a height which is outside of the current range of the height data.But, extrapolation can often be dangerous, as you don't have the data to support the prediction.

More than one explanatory variable

In the example above, height was the only explanatory variable.

Often (usually), you have more than one candidate variable.

E.g. age of child, gender of child, socio-economic status, height/weight of parents, etc.

In some situations there may be large numbers of potential explanatory variables and it is not clear, *a priori*, which of them have good explaining power for the response (weight).

Variable selection is the process that attempts to find the variables that have good explaining power, and contribute to effective regression. More later ...

Summary: Important things

Plots — exploratory analyses

Model assumptions: know what they are and bear them in mind.

Least squares, Maximum Likelihood, Bayesian, others?

Model checking

Variants

All models are wrong, but some are useful, (George Box (1976), but existed earlier.)

"You can't use that model for financial data, because everyone knows that such data are generated using GARCH processes"!

Elements of Statistical Learning: Lecture 2. Multiple Regression and Variable Selection

Guy Nason[1]

Department of Mathematics
Imperial College

Multivariate Linear Model

Now suppose we have p explanatory variables.

Call them $\mathbf{X} = (X_1, \ldots, X_p)$.

We have the values of $\mathbf{X}$ on n subjects.

The outcome (response) for the ith subject is Y_i (univariate).

And exploratory data analysis (such as plots of X_j against Y) indicate that a linear model is appropriate.

So, could write

$$Y_i = X_{i,1}\beta_1 + X_{i,2}\beta_2 + \cdots X_{i,p}\beta_p + \epsilon_i,$$

for $i = 1, \ldots, n$. I.e. n equations, so clearer to write in matrix form.

Matrix formulation of multivariate linear model

Let $Y = (Y_1, \ldots, Y_n)^T$ and $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$.

Let $\beta = (\beta_1, \ldots, \beta_p)$ and $X = (X_{i,j})$ the $n \times p$ matrix containing elements $X_{i,j}$ values of $\mathbf{X}$ on the ith individual and jth variable.

Then, we can write the multivariate linear model as

$$Y = X\beta + \epsilon. \tag{1}$$

Here ϵ is a n-vector of IID random variables with mean zero and variance σ^2 as before. We assume Gaussianity.

Residuals, sum of squares, least squares

The residual, as a function of β, can be defined as a vector e

$$e = Y - X\beta.$$

The residual sum of squares as

$$\mathrm{RSS}(\beta) = e^T e.$$

The least squares estimator is

$$\hat{\beta} = \mathrm{argmin}_\beta \, RSS(\beta).$$

And the (empirical) residual is usually written as

$$e = Y - X\hat{\beta}. \tag{2}$$

Example: simple linear regression

Here, $Y = (Y_1, \ldots, Y_n)$, $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ and

$\beta = (a, b)^T$ and

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

For model $Y_i = a + bx_i + cx_i^2 + \epsilon_i$ then Y, ϵ are the same, but $\beta = (a, b, c)^T$ and

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}$$

.

Lemma: vector and matrix differentiation

Lemma 1. Let $u = \beta^T v = \sum_{i=1}^{p} \beta_i v_i$, where v is a p-vector.

Then

$$\frac{\partial u}{\partial \beta_j} = \sum_{i=1}^{p} v_i \frac{\partial \beta_i}{\partial \beta_j} = v_j,$$

so $\frac{\partial u}{\partial \beta} = v$.

Lemma 2. Let the quadratic form $w = \beta^T A \beta$ for some symmetric matrix A. We can write

$$w = \sum_{i=1}^{p} \beta_i \sum_{j=1}^{p} A_{i,j} \beta_j,$$

then $\frac{\partial w}{\partial \beta} = 2A\beta$ (see exercises).

Least squares estimation

Write RSS as

$$\begin{aligned} \text{RSS}(\beta) &= (Y - X\beta)^T (Y - X\beta) \\ &= Y^T Y - (X\beta)^T Y - Y^T X\beta + (X\beta)^T X\beta \\ &= Y^T Y - 2\beta^T X^T Y + \beta^T X^T X\beta. \end{aligned}$$

Differentiating with respect to β

$$\frac{\partial \text{RSS}(\beta)}{\partial \beta} = -2X^T Y + 2X^T X\beta \tag{3}$$

Setting equal to zero and solving for β gives

$$X^T Y = X^T X\hat{\beta} \implies \hat{\beta} = (X^T X)^{-1} X^T Y$$

$X^T X$ is invertible if X is of full rank.

Orthogonality of residual vector to columns of X

First, note that

$$\frac{\partial^2 \text{RSS}(\beta)}{\partial \beta \, \partial \beta^T} = 2X^T X,$$

which is positive definite (since $X^T X$ is invertible) and hence, we have a minimum at this point.

Second, note that, in solving the equations $\frac{\partial \text{RSS}(\beta)}{\partial \beta} = 0$ on the previous slide we can write

$$X^T (Y - X\hat{\beta}) = 0,$$

i.e. we've substituted the solving value of $\beta = \hat{\beta}$. The $(Y - X\hat{\beta})$ is the residual of the least squares fit. So, the residual vector of the fit is orthogonal to the columns of X.

Hat matrix

Define $H = X(X^T X)^{-1} X^T$ to be the hat matrix.

Then the fitted values of the model are

$$\hat{Y} = X\hat{\beta} = HY,$$

so that hat matrix puts the hat on Y.

Note that:

1. Parameter estimates $\hat{\beta}$ are linear combinations of the Y, and
2. fitted values, $\hat{Y}$, are also linear combinations of the Y.

Inference about β: Expectation, variance of Y

First, remember $\mathbb{E}(Y) = X\beta$, from (1).

The $p \times p$ variance matrix of random p-vector U is defined as

$$\Sigma = \text{var}(U) = \mathbb{E}\left[\{U - \mathbb{E}(U)\}\{U - \mathbb{E}(U)\}^T\right] \tag{4}$$

so $\text{cov}(U_i, U_j) = \Sigma_{i,j}$

Hence,

$$\begin{aligned} \text{var}(Y) &= \mathbb{E}\left[\{Y - \mathbb{E}(Y)\}\{Y - \mathbb{E}(Y)\}^T\right] \\ &= \mathbb{E}(\epsilon\epsilon^T) = \Sigma_\epsilon = \sigma^2 I_n, \end{aligned}$$

where I_n is the n-dimensional identity matrix.Also:

$$\sigma^2 I_n = \mathbb{E}(YY^T) - \mathbb{E}(Y)\mathbb{E}(Y^T) \implies \mathbb{E}(YY^T) = \sigma^2 I_n + X\beta\beta^T X^T.$$

Inference about β: Expectation, variance of β

Expectation

$$\begin{aligned}\mathbb{E}(\hat{\beta}) &= \mathbb{E}\left\{(X^T X)^{-1} X^T Y\right\} \\ &= (X^T X)^{-1} X^T \mathbb{E}(Y) = (X^T X)^{-1} X^T X \beta = \beta.\end{aligned}$$

So, least-squares estimator is unbiased.What about variance?

$$\begin{aligned}\text{var}(\hat{\beta}) &= \mathbb{E}\left[\{\hat{\beta} - \mathbb{E}(\hat{\beta})\}\{\hat{\beta} - \mathbb{E}(\hat{\beta})\}^T\right] \\ &= \mathbb{E}(\hat{\beta}\hat{\beta}^T) - \mathbb{E}(\hat{\beta})\mathbb{E}(\hat{\beta}^T) \\ &= \mathbb{E}\left[(X^T X)^{-1} X^T Y Y^T X (X^T X)^{-1}\right] - \beta\beta^T \\ &= (X^T X)^{-1} X^T \mathbb{E}(YY^T) X (X^T X)^{-1} - \beta\beta^T \\ &= (X^T X)^{-1} X^T (\sigma^2 I_n + X\beta\beta^T X^T) X (X^T X)^{-1} - \beta\beta^T \\ &= \sigma^2 (X^T X)^{-1}\end{aligned}$$

Explanation about var($\hat{\beta}$)

Here var($\hat{\beta}$) is a $p \times p$ covariance matrix.

The covariance between $\hat{\beta}_i$ and $\hat{\beta}_j$ is $\left\{\sigma^2(X^T X)^{-1}\right\}_{i,j}$.

The parameter variance depends on X, i.e. the design, which can sometimes be controlled with proper planning.

For example, if X is an orthogonal matrix then $X^T X = I_p$ and var $\hat{\beta}$ is a diagonal matrix, so $\hat{\beta}_i$ and $\hat{\beta}_j$ are uncorrelated for $i \neq j$.

This is practically useful because it means that different variables can be considered individually, without reference to the others.

Inference about β: Distribution of β

If the $\{\epsilon_i\}$ are normally distributed $N(0, \sigma^2)$ then, since $\hat{\beta}$ is a linear combination of normals, then it must be normal itself and we must have

$$\hat{\beta} \sim N_p\{\beta, \sigma^2(X^T X)^{-1}\}.$$

(And if X is orthogonal, then $\hat{\beta}_i$ and $\hat{\beta}_j$ are independent, for $i \neq j$.)

[Because uncorrelated and Gaussian $\implies$ independent]

Estimating σ^2

Let v_j be the jth diagonal element of $(X^T X)^{-1}$. Then $\text{var}(\hat{\beta}_j) = \sigma^2 v_j$. Then the standardised coefficient or Z-score is

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}, \tag{5}$$

where we typically estimate σ^2 by

$$\hat{\sigma}^2 = (n-p)^{-1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

and $\mathbb{E}\hat{\sigma}^2 = \sigma^2$.

Under the model assumptions and the null hypothesis $H_0 : \beta_j = 0$, the quantity z_j is distributed as a Student's t-distribution on $n-p$ degrees of freedom (and usually approximated by $N(0,1)$ for n large, and larger than p).

p or $p+1$?

When we add a mean or intercept parameter into the model, e.g. β_0, as we will do later, this can be included in the design matrix X as a column of 1s as on Slide 5.

This increases the number of parameters from p to $p+1$, and then the $n-p$ quantities on the previous slide become $n-p-1$.

Of course, for even moderately large n the difference between $1/(n-p)$ and $1/(n-p-1)$ is not large, unless, of course, p is large too.

Inference about β: terminology

We are interested in whether β_j is zero or not.We calculate z_j and compare this to $N(0,1)$ (or appropriate t).

The p-value is the probability of getting a more extreme value than z_j, if assuming it comes from a $N(0,1)$ distribution.

We are indifferent to positive or negative values so, we compute

$$p_{z_j} = \mathbb{P}(|Z| > z_j) = \mathbb{P}(Z < -z_j) + \mathbb{P}(Z > z_j) = 2\mathbb{P}(Z < -z_j) = 2\Phi(-z_j),$$

where $Z \sim N(0,1)$ and its distribution function is Φ.

Usually say if $p_{z_j} < 0.01$ then strong evidence to reject H_0, else $p_{z_j} < 0.05$ some evidence to reject H_0, otherwise have little or no evidence to reject H_0 (but this does NOT mean accept H_0).

Bayesian methods give distribution for z_j, might be more useful.

Dropping/Including Groups of Variables

Consider two models: model M0 and model M1.

Model M1 is the larger model with $p_1 + 1$ variables.

Model M0 is smaller, with $p_0 + 1$, all contained in M1 (nested).

If we go from the larger to the smaller, we're constraining $p_1 - p_0$ parameters to be zero.

Calculate

$$F = \frac{(\text{RSS}_0 - \text{RSS}_1)/(p_1 - p_0)}{\text{RSS}_1/(n - p_1 - 1)},$$

Note, $\text{RSS}_1 \leq \text{RSS}_0$ as M1 has more parameters (better fit).

Inference about models

Under the model assumptions and that M0 (smaller) is true, it can be shown that $F \sim F_{p_1-p_0,n-p_1-1}$ distribution, which gives us a way of assessing which model to prefer as our working assumption going forward.

Child Data Example

The child data has two other variables: `age` and `sex`.

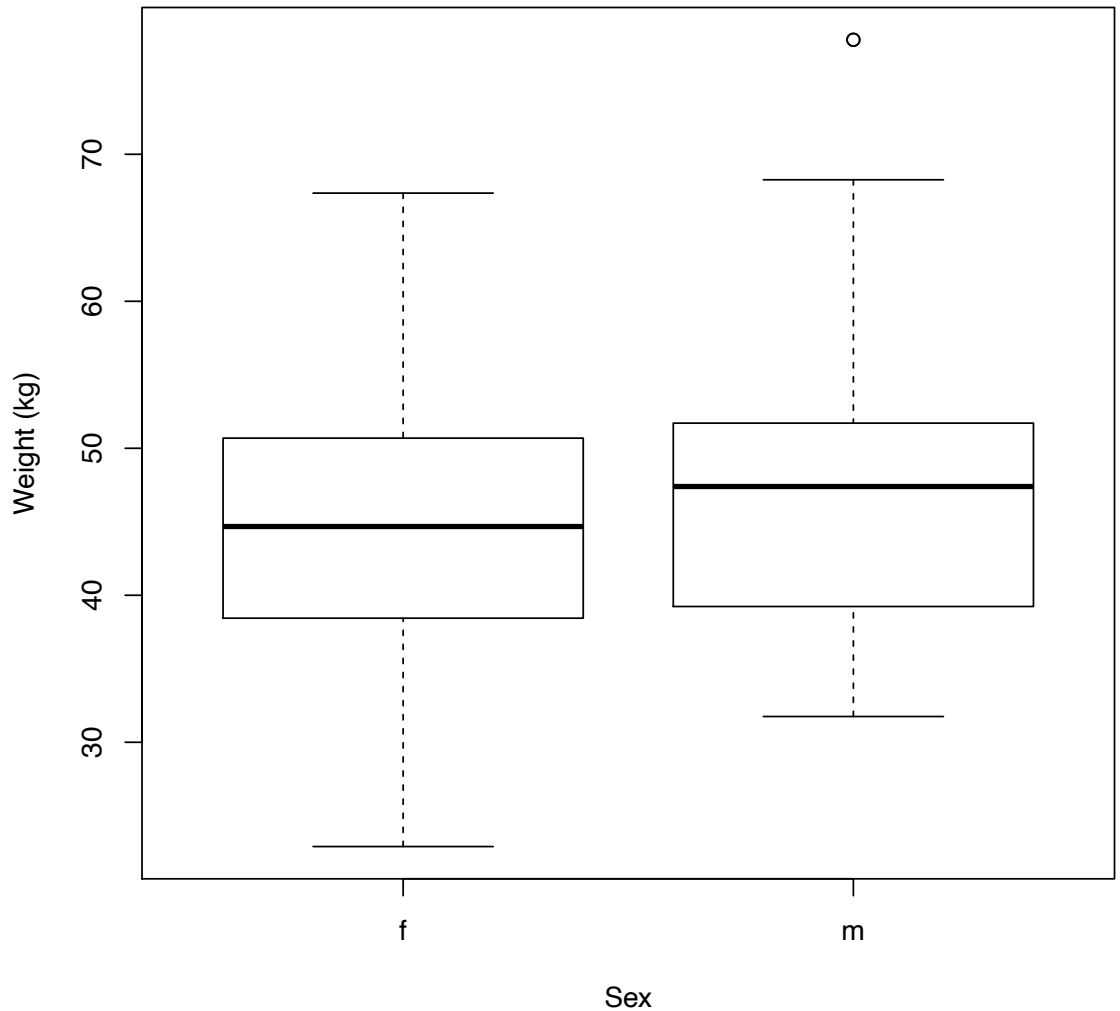
All the variables are contained in the `childhwDF` data frame.

Let's look at some exploratory plots.

The `R` function `split`, splits a variable into (two) groups depending on a (dichotomous) factor.

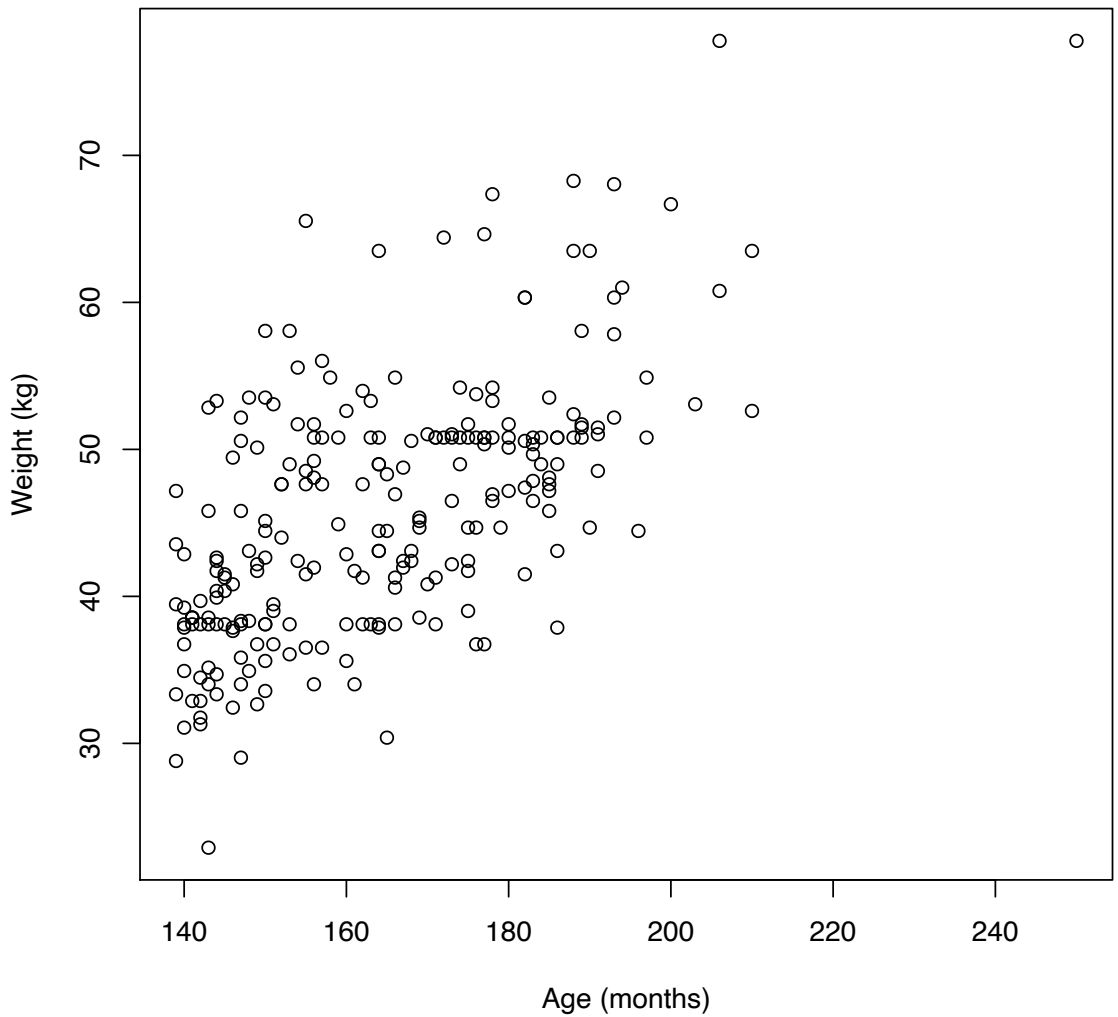
Weight split by sex

```
attach(childhwDF)
boxplot(split(x=weight, f=sex))
```



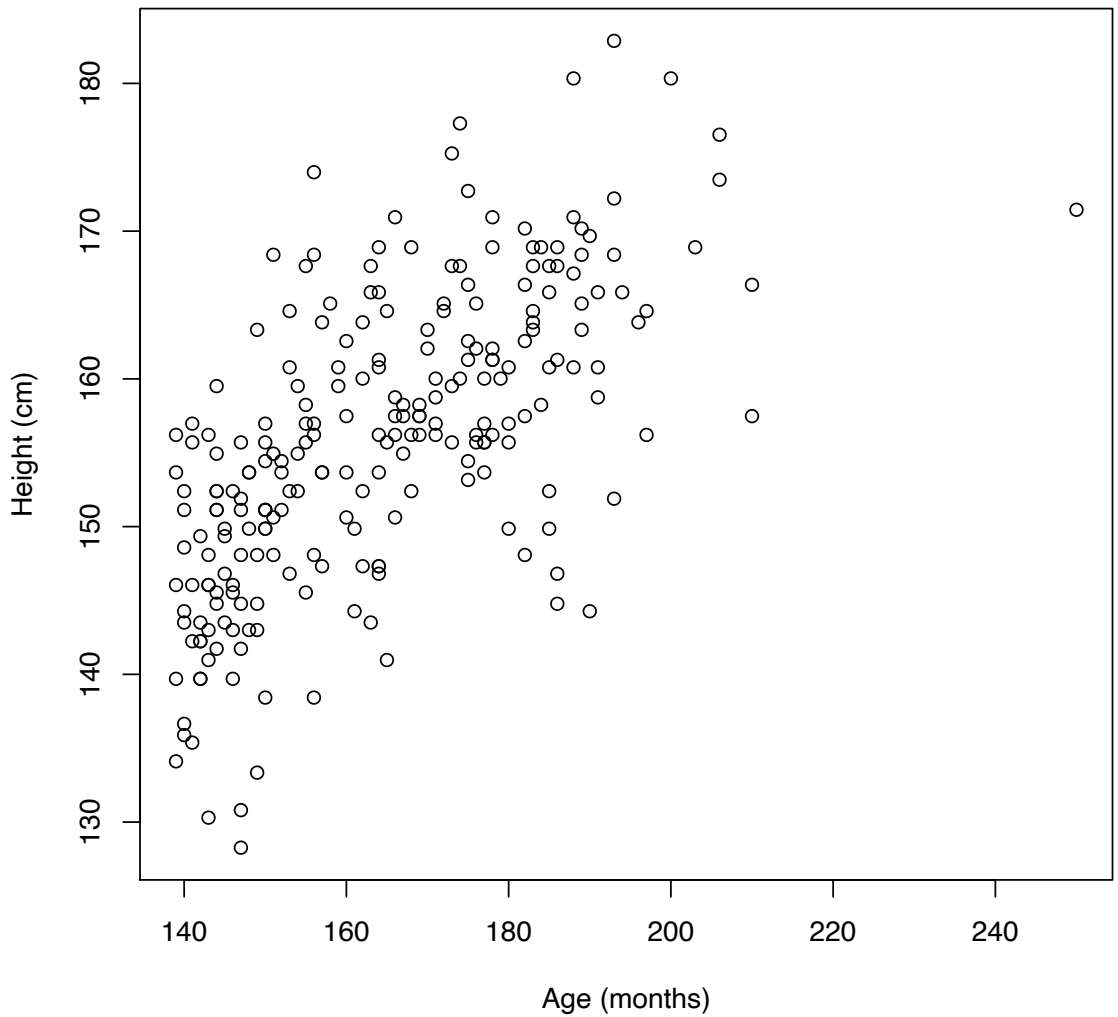
Weight against age

```
plot(age, weight, xlab="Age (months)", ylab="Weight (kg)")
```



Height against age

```
plot(age, height, xlab="Age (months)", ylab="Height (cm)")
```



F-test on the three variables

```
#
# Fit full model with all three variables and intercept
#
fit1 <- lm( weight~ height+sex+age, data=childhwDF)
#
# Now just fit intercept
#
fit0 <- lm( weight~ 1, data=childhwDF)
#
# Calculate RSS for both models
#
rss1 <- sum(resid(fit1)^2)
rss0 <- sum(resid(fit0)^2)
```

F-test on the three variables (continued)

```
#
# Calculate F statistic
#
((rss0 - rss1)/(4-1))/(rss1/(237-4-1))
[1] 131.9756
#
# 99% quantile of appropriate F
#
qf(0.99, df1=3, df2=237-4-1)
[1] 3.867119
```

The null hypothesis would be $\beta_{\text{sex}} = \beta_{\text{age}} = \beta_{\text{height}} = 0$

Now $132.0 \gg 3.87$, so have strong evidence to reject the null hypothesis.

Stepwise regression: backward deletion

`summary(fit1)`

Start with full model and look at coefficients and s.e.'s

	Estimate	Std. Error	t value	$\mathbb{P}(>\|t\|)$
(Intercept)	-58.15	5.56	-10.45	$< 2 \times 10^{-16}$
`height`	0.55	0.048	11.62	$< 2 \times 10^{-16}$
`sex-m`	-0.15	0.73	-0.21	0.834
`age`	0.11	0.025	4.25	3.1×10^{-5}

The sex variable is not significant, so we drop it from the regression:

Stepwise regression: backward deletion

```
fit2 <- update(fit1, . ~ . -sex)
summary(fit2)
```

	Estimate	Std. Error	t value	$\mathbb{P}(>\|t\|)$
(Intercept)	-57.98	5.49	-10.56	$< 2 \times 10^{-16}$
`height`	0.55	0.046	12.01	$< 2 \times 10^{-16}$
`age`	0.11	0.025	4.36	1.95×10^{-5}

Backward-stepwise selection: starts with the full model and sequentially delete the predictor that has least input on the fit.

I stop when all variables are statistically significant at the 5% level.

Forward-stepwise selection

Start with intercept in model
`fitInt <- lm(weight ~ 1, data=childhwDF)`

Sequentially add the term that most improves the fit.

If you have p variables to add, add them one by one and see which one improves the fit the most.

Stop when the fit stops improving, or when the introduced variable is is not statistically significant (i.e. the coefficient in the model is not statistically significant).

Summary

This lecture introduced the following concepts:

The Multivariate Linear Model

Vector and matrix differentiation

Computing Least Square Estimates for the multivariate linear model

Orthogonality between the residual vector and the columns of X

Hat matrix

Expectation and Variance matrix of $\hat{\beta}$

Distribution of $\hat{\beta}$ and estimation of σ^2.

p-values

Variable Selection groups, backward deletion and forward stepwise

Child Data example

Elements of Statistical Learning: Lecture 3. Which variables are important?

Guy Nason[1]

Department of Mathematics
Imperial College

Gauss Markov theorem

Let $\hat{\beta}$ be the least-squares estimator.

Let $\check{\beta} = CY$ be another unbiased linear estimator of β with $C = (X^T X)^{-1} X^T + D$, where D is a $p \times n$ non-zero matrix.Then

$$\begin{aligned}
\mathbb{E}(\check{\beta}) &= \mathbb{E}(CY) \\
&= \mathbb{E}\left[\{(X^T X)^{-1} X^T + D\}(X\beta + \epsilon)\right] \\
&= \{(X^T X)^{-1} X^T + D\}X\beta + \{(X^T X)^{-1} X^T + D\}\mathbb{E}(\epsilon) \\
&= (X^T X)^{-1} X^T X\beta + DX\beta \\
&= (I_p + DX)\beta,
\end{aligned}$$

and $\check{\beta}$ is unbiased if and only if $DX = 0$.

Gauss Markov theorem continued 2

Also:

$$
\begin{aligned}
\text{var}(\check{\beta}) &= \text{var}(CY) \\
&= C\,\text{var}(Y)C^T \\
&= \sigma^2 CC^T \\
&= \sigma^2\{(X^TX)^{-1}X^T + D\}\{(X^TX)^{-1}X^T + D\}^T \\
&= \sigma^2\{(X^TX)^{-1}X^T + D\}\{X(X^TX)^{-1} + D^T\} \\
&= \sigma^2\{(X^TX)^{-1}X^TX(X^TX)^{-1} + DX(X^TX)^{-1} \\
&\quad +(X^TX)^{-1}X^TD^T + DD^T\} \\
&= \sigma^2\{(X^TX)^{-1} + DD^T\} \\
&= \text{var}(\hat{\beta}) + \sigma^2 DD^T.
\end{aligned}
$$

since $\check{\beta}$ is unbiased and $DX = 0$.

Gauss Markov theorem continued 3

Now examine an arbitrary linear combination of parameters: $\theta = \alpha^T \beta$ for some p-vector α.

Let $\hat{\theta} = \alpha^T \hat{\beta}$ and $\check{\theta} = \alpha^T \check{\beta}$.Clearly, both $\hat{\theta}$, $\check{\theta}$ are unbiased for θ.

Now

$$\begin{aligned} \text{var}(\alpha^T \check{\beta}) &= \alpha^T \text{var}(\check{\beta}) \alpha \\ &= \alpha^T \{\text{var}(\hat{\beta}) + \sigma^2 D D^T\} \alpha \\ &= \text{var}(\alpha^T \hat{\beta}) + \sigma^2 \alpha^T D D^T \alpha. \end{aligned}$$

Now $\alpha^T D D^T \alpha = v^T v = \sum_{i=1}^{n} v_i^2 \geq 0$, or recognise DD^T is positive semi-definite, where v is some vector $v = D^T \alpha$

Implication of Gauss Markov: BLUE

Gauss Markov means $\text{var}(\alpha^T \hat{\beta}) \leq \text{var}(\alpha^T \check{\beta})$.

And recall $\check{\beta}$ was arbitrary linear unbiased estimator.

So, least squares $\hat{\beta}$ is best linear unbiased estimator or BLUE.

However, it's not necessarily the best estimator in terms of mean squared error (MSE).

There might be estimators that are a little biased, but have better variance and a smaller MSE overall.

With subset selection we either include or omit variables, there is nothing in between — which can lead to high variability.

So, we try modifying coefficients instead — shrinkage.

Centring and standardising

Often, the methods below are not invariant to changes in origin and variance of the Xs.

Fix a particular variable: $j \in \{1, \ldots, p\}$ then $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$ will have a mean $\bar{x}_j$ and sample variance s_j^2.

Centering and standardising means we operate on the new variable $x^\star$ defined by

$$x^\star_{i,j} = (x_{i,j} - \bar{x}_j)/s_j,$$

for $i = 1, \ldots, n$.

This often makes the variables more comparable, and sometimes makes $(X^T X)$ less singular.

Ridge regression: What is it?

Ridge regression solves the optimisation problem

$$\hat{\beta}^{\text{ridge}}(\lambda) = \text{argmin}_{\beta} \left\{ \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

This is a penalized least squares optimisation: $\lambda > 0$, to be chosen.

We want to get a good fit and keep control on the overall size of the elements of β.

The λ controls the balance between fidelity of the fit and size of the coefficients.

Note $\hat{\beta}^{\text{ridge}}(0) = \hat{\beta}$ and $\hat{\beta}_0^{\text{ridge}}(\lambda) \to \bar{Y}$ for λ getting large (and $\beta_j \to 0$ for $j \neq 0$).

Alternative formulation

Ridge regression solution can be written as

$$\hat{\beta}^{\text{ridge}} = \text{argmin}_{\beta} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j} \beta_j \right)^2,$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq t$$

Instability in ordinary least squares: ridge helps

When variables in least squares are highly correlated it can cause instability in the least squares estimators.

E.g. suppose $X^T X = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. Then $\det(X^T X) = 1 - \rho^2$ and

$$(X^T X)^{-1} = \frac{1}{1-\rho^2} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}$$

So, if ρ gets close to 1, then the entries if $(X^T X)^{-1}$ become very large, i.e. the variance of the least squares parameter estimates gets large, and so some entries of $\hat{\beta}$ can become unreasonably large.

The constraint $\sum_{j=1}^{p} \beta_j^2$ ameliorates this problem.

Ridge Regression RSS in matrix form

Ridge objective using centred variables (and Y also centred by $\bar{Y}$, see course book top of page 64)

$$\text{RSS}(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta.$$

Differentiating with respect to β (using Lemma 2)

$$\frac{\partial \text{RSS}(\beta)}{\partial \beta} = -2X^TY + 2X^TX\beta + 2\lambda\beta.$$

Setting equal to zero and solving for β gives

$$(X^TX + \lambda I_p)\hat{\beta}^{\text{ridge}} = X^TY$$

Hence,

$$\hat{\beta}^{\text{ridge}} = (X^TX + \lambda I_p)^{-1}X^TY.$$

Condition number

The condition number of a matrix measures how "hard" it is to invert a matrix.

A matrix with a high condition number is "closer" to being singular.

The condition number of a normal matrix, A, is given by

$$\kappa(A) = \xi_{\text{max}}(A)/\xi_{\text{min}}(A),$$

where $\xi_{\text{max}}(A)$ is the largest eigenvalue of A and similarly for the minimum.

Condition number

Think of $A = X^T X$.

The eigenvalues of $A = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ are $(1-\rho, 1+\rho)$, so if $\rho > 0$ then

$$\kappa(A) = (1+\rho)/(1-\rho).$$

Note, $\kappa(A) \to \infty$ as $\rho \to 1$.

Whereas, the eigenvalues of $A + \lambda I = \begin{pmatrix} 1+\lambda & \rho \\ \rho & 1+\lambda \end{pmatrix}$ are $(1+\lambda-\rho, 1+\lambda+\rho)$, so

$$\kappa(A) = (1+\lambda+\rho)/(1+\lambda-\rho).$$

Since $\lambda > 0$ then denominator does not tend to zero.

Ridge Regression Interpretation

Adding the λ to the diagonal makes the $X^T X$ matrix "more" invertible.

Reduces the variability in estimated coefficients.

Called ridge, as it adds to the 'ridge' of the matrix. (original motivation)

A way of dealing with multi-collinearity between the X design columns.

Sometimes known as Tikhonov regularisation.

Bayesian interpretation

If prior $\beta_j \sim N(0, \tau^2)$ independently, for $j = 1, \ldots, p$.

And, $Y_i \sim N(\beta_0 + x_i^T \beta, \sigma^2)$, independently.

Then $\beta | Y \sim N$ with posterior mean $\hat{\beta}^{\text{ridge}}$, with $\lambda = \sigma^2/\tau^2$.

$\hat{\beta}^{\text{ridge}}$ is also the posterior mode.

Choosing the prior is equivalent to choosing the regularisation parameter λ (don't have to choose λ separately, but do have to choose τ^2, so moves problem to another position 'under the carpet')

Singular value decomposition (Shrinkage)

The singular value decomposition (SVD) of a $n \times p$ (centred) matrix X is

$$X = UDV^T,$$

where U and V are $n \times p$ and $p \times p$ orthogonal matrices, respectively, with the columns of U spanning the column space of X and V spanning the row space.

D is a $p \times p$ diagonal matrix with values $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ called the singular values.

If one or more values of $d_j = 0$, then the X is singular.

We have described the *economy sized* SVD where U is not square and D is.

SVD applied to $X^T X$.

If $X = UDV^T$ then

$$X^T X = VD^T U^T UDV^T = VD^T DV^T = VD^2 V^T,$$

as D is diagonal. So

$$(X^T X)^{-1} = (VD^2 V^T)^{-1} = (V^T)^{-1} D^{-2} V^{-1},$$

here D^{-2} is the diagonal matrix with entries d_j^{-2}.

Shrinkage: LS (SVD)

Using the SVD, we can write the least squares fitted vector as

$$\begin{aligned} X\hat{\beta} &= X(X^T X)^{-1} X^T Y \\ &= UDV^T (V^T)^{-1} D^{-2} V^{-1} V D^T U^T Y \\ &= UDI_p D^{-2} I_p D^T U^T Y \\ &= UU^T Y. \end{aligned}$$

Note that $U^T Y$ are the coordinates of Y with respect to the orthonormal basis implied by U.

Shrinkage: Ridge (SVD)

Same again, but for ridge

$$X\hat{\beta}^{\text{ridge}} = (UDV^T)(VD^2V^T + \lambda I_p)^{-1}(VD^TU^T)Y \qquad (1)$$

What is this?

Well, let M, L be two invertible $p \times p$ matrices. Then properties of the matrix inverse mean

$$ML^{-1} = (LM^{-1})^{-1}.$$

Let's apply this to (1) with $M = V^T$ and $L = VD^2V^T + \lambda I_p$ to get

$$\begin{aligned} V^T(VD^2V^T + \lambda I_p)^{-1} &= \{(VD^2V^T + \lambda I_p)(V^T)^{-1}\}^{-1} \\ &= \{VD^2 + \lambda(V^T)^{-1}\}^{-1} \end{aligned}$$

Shrinkage: Ridge (SVD) — 2

Now apply again, but here $M = \{VD^2 + \lambda(V^T)^{-1}\}^{-1}$ and $L^{-1} = V$, gives

$$\begin{aligned}
\{VD^2 + \lambda(V^T)^{-1}\}^{-1}V &= [V^{-1}\{VD^2 + \lambda(V^T)^{-1}\}]^{-1} \\
&= \{D^2 + \lambda V^{-1}(V^T)^{-1}\}^{-1} \\
&= \{D^2 + \lambda(V^T V)^{-1}\}^{-1} \qquad (2) \\
&= (D^2 + \lambda I_p)^{-1}, \qquad (3)
\end{aligned}$$

where we use the $(AB)^{-1} = B^{-1}A^{-1}$ property of matrix inverses in (2) and then the orthogonality of V on the last line.

Shrinkage: Ridge (SVD) — 3

Putting together (1) with (3) yields

$$\begin{aligned} X\hat{\beta}^{\text{ridge}} &= UD(D^2 + \lambda I_p)^{-1} DU^T Y \\ &= \sum_{j=1}^{p} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}_j^T Y, \end{aligned}$$

where $\mathbf{u}_j$ are the columns of U.

So, ridge regression, like least squares, transforms the basis to U, then *shrinks* the *coordinates* according to $d_j^2/(d_j^2 + \lambda)$.

If d_j^2 is large, then $d_j^2/(d_j^2 + \lambda) \approx 1$.
If d_j^2 is small, then the ratio is less than one.

So, ridge regression shrinks coordinates that correspond to small d_j^2, i.e. directions where the singular value is small.

Link between SVD and Principal Components

The *sample* covariance matrix of the *centred* data matrix X $(n \times p)$is $S = n^{-1} X^T X$ (c.f. the variance of a random vector, from Lecture 2, equation (4)).

Then $X^T X = (UDV^T)^T UDV^T = VD(U^T U)DV^T = VD^2 V^T$, which is eigendecomposition of $X^T X$ (and S, up to factor of n).

The eigenvectors v_j (columns of V) are called the *principal components* or Karhunen-Loeve directions of X.

Let $z_{1,(n\times 1)} = X_{(n\times p)} v_{1,(p\times 1)}$ be the projection of the data matrix X onto the first principal component. Then (assume X has mean zero, hence z = centred)

$$S(z_1) = n^{-1}(z_1^T z_1) = v_1^T X^T X v_1/n = v_1^T VD^2 V^T v_1/n = d_1^2/n,$$

Link between SVD and Principal Components — 2

So, the variance of X in direction v_1 is d_1^2/n.

And, the variance of X in direction v_j is d_j^2/n and we know $d_j > d_i$ if $i > j$.

So, we can partition the variance of X into orthogonal directions each with variance d_j^2/n.

Plot of principal components of age/height matrix

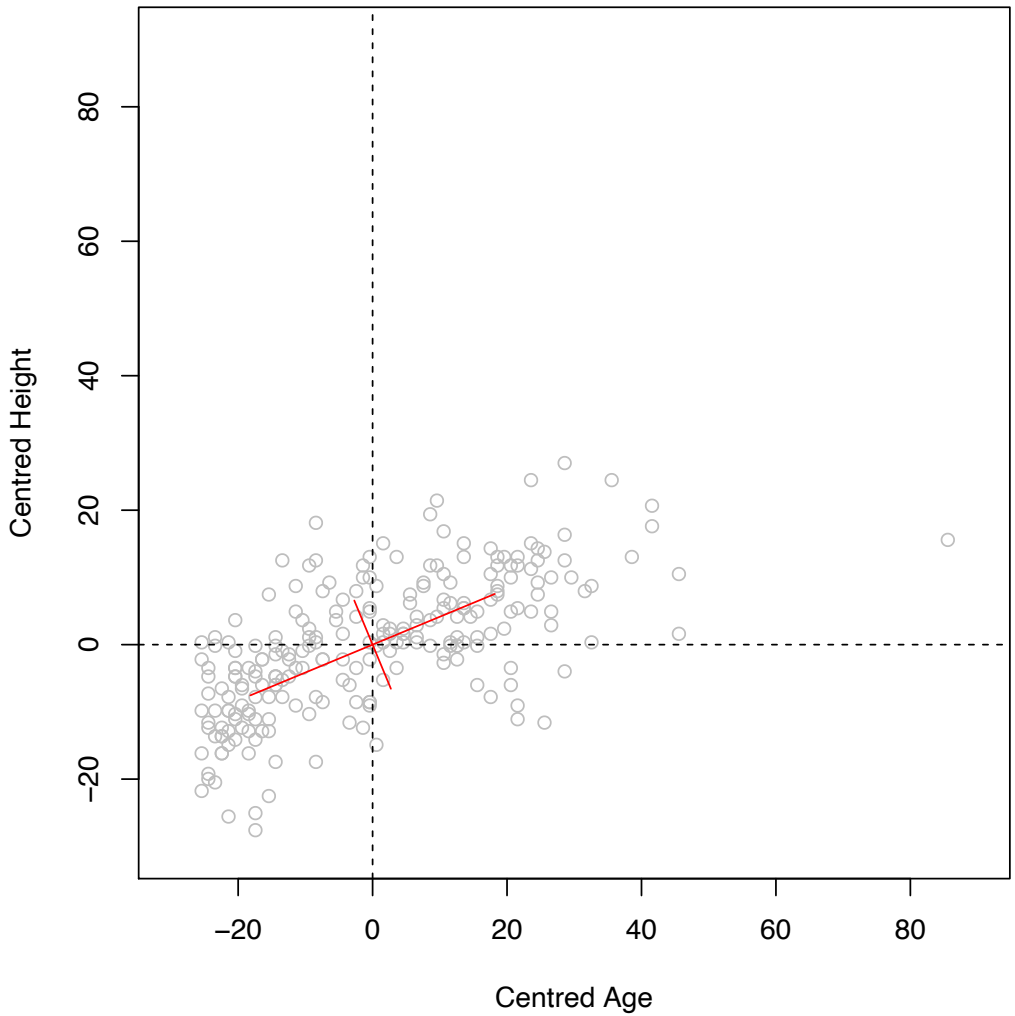
```
# Centre age and height, plot them
age2 <- age-mean(age)
height2 <- height-mean(height)

oldpar <- par(pty="s") # Make square plot
plot(age2, height2, xlim=c(-30, 90), ylim=c(-30, 90),
    xlab="Centred Age", ylab="Centred Height", col="grey")

abline(h=0, lty=2) # cross hair horizontal
abline(v=0, lty=2) # cross hair vertical

# Put centred variables in new 2D data matrix
xdf <- cbind(age2, height2)
ed <- eigen( t(xdf)%*% xdf/length(age2)) # Eigendecomposition

# Extract eigenvectors
evec <- ed$vectors
```

PC plot (continued)

```
# Calculate standard deviations of principal components
pc1sd <- sqrt(ed$values[1])
lines(c(-1, 1)*pc1sd*evec[1,1], c(-1,1)* pc1sd*evec[2,1],
    col=2) # Plot first PC

pc2sd <- sqrt(ed$values[2])
lines(c(-1, 1)*pc2sd*evec[1,2], c(-1,1)* pc2sd*evec[2,2],
    col=2) # Plot second PC

par(oldpar) # Restore graphics parameters
```

Note, we plot the standard deviation - and, as the data looks fairly normal, most of the data lies within ± 2 standard deviations.

Plot of Principal Components of Height and Age (centred)



Projection of Data Matrix onto Projection Vectors

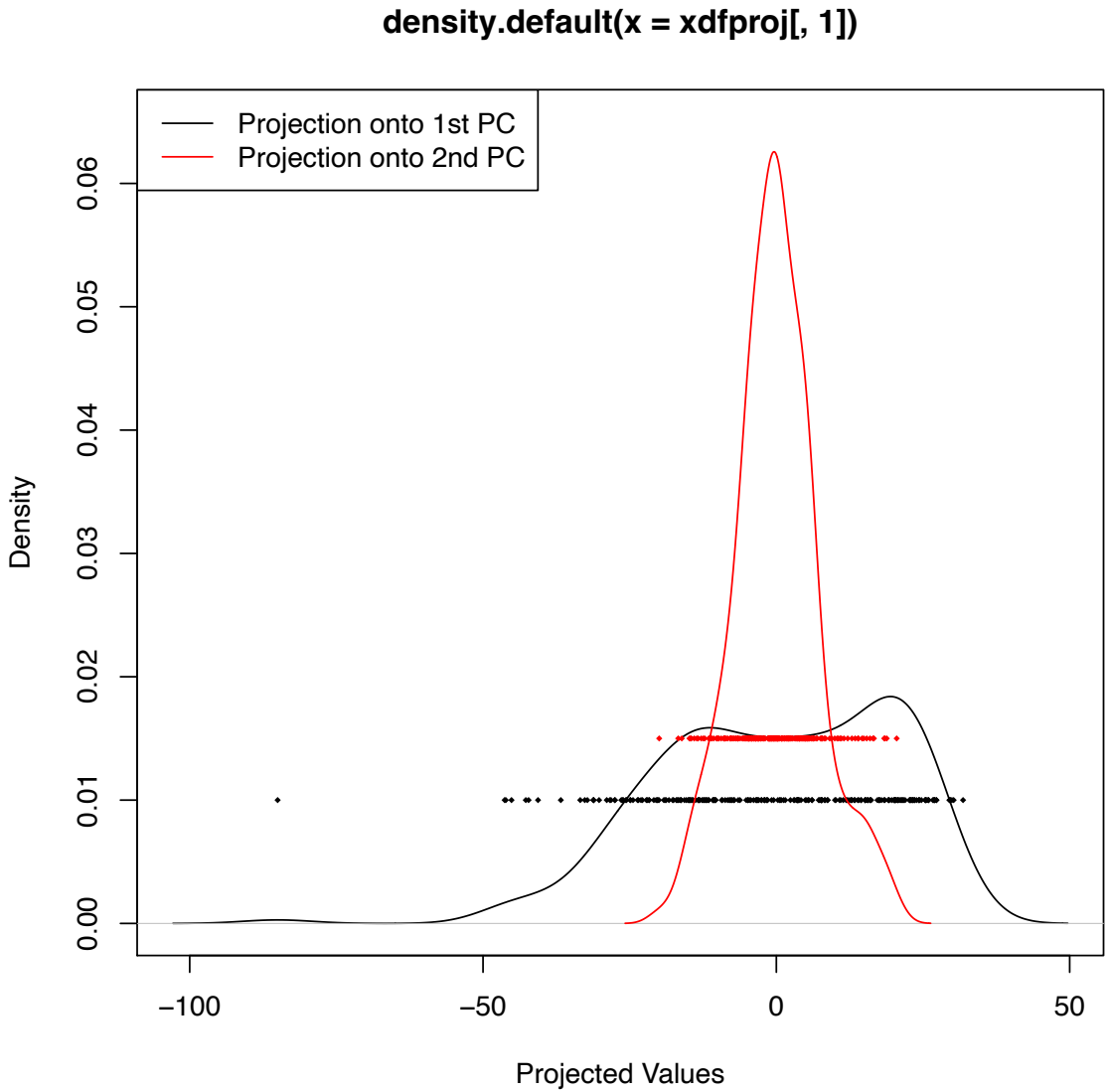
We can project the p-dimensional data matrix onto the eigenvectors with

$$Z = XV$$

and then do a density estimate of the first, second, etc. column.

```
xdfproj <- xdf %*% ed$vectors

plot(density(xdfproj[,1]), ylim=c(0,0.065),
    xlab="Projected Values", sub="") # Density est. of z_1

points(xdfproj[,1], rep(0.01,237), pch=18, cex=0.5)
lines(density(xdfproj[,2]), col=2) # Density est. of 2
points(xdfproj[,2], rep(0.015,237), pch=18, col=2, cex=0.5)
legend(x="topleft", col=1:2, lwd=1,
    legend=c("Projection onto 1st PC",
    "Projection onto 2nd PC"))
```

Density ests. of data projected onto principal components.



Ridge regression interpretation

Ridge regression shrinks coordinates in directions with smaller variance.

Geometrically, imagine a plane being fitted on top of the data — you get more stability from the longer directions, i.e. those with greater variance.

This stability improvement was the original reason for developing ridge regression.

(Effective) Degrees of Freedom

Define the "effective" hat matrix as $H_\lambda = X(X^T X + \lambda I_p)^{-1} X^T$.

Then, the *effective degrees of freedom* is defined to be

$$\begin{aligned} \mathrm{df}(\lambda) &= \mathrm{tr}\{H_\lambda\} \\ &= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}, \end{aligned}$$

as the trace of a matrix is the sum of its eigenvalues.

Note: $\mathrm{df}(0) = p$, which is the actual degrees of freedom from the least squares estimate.

Bias and variance

We've mentioned that the variance of $\hat{\beta}^{\text{ridge}}$ is less than $\hat{\beta}$ due to the constraint — this translates to the fitted values as $\hat{Y} = X\hat{\beta}^{\text{ridge}}$.

What about the bias of ridge regression estimates for β?

$$\begin{aligned} \mathbb{E}(\hat{\beta}^{\text{ridge}}) &= (X^T X + \lambda I)^{-1} X^T \mathbb{E}(Y) \\ &= (X^T X + \lambda I)^{-1} X^T X \beta \end{aligned}$$

This can be worked out for simple cases, or approximated.

Summary

This lecture has covered the following concepts.

Gauss-Markov theorem and best linear unbiased estimators (BLUE)

Centring and standardising variables

Ridge regression for co-linear variables, L_2 penalty, interpretations (Bayesian)

Ridge regression in matrix form

Condition number

Ridge as shrinkage

Principal components

Effective Degrees of Freedom

Elements of Statistical Learning: Lecture 4. The Lasso

Guy Nason[1]

Department of Mathematics
Imperial College

Lasso regression

In contrast to ridge regression the lasso solves:

$$\hat{\beta}^{\text{lasso}}(\lambda) = \text{argmin}_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

Again, a penalized least squares optimisation: $\lambda > 0$, to be chosen.

We want to get a good fit and keep control on the overall size of the elements of β.

The λ controls the balance between fidelity of the fit and size of the coefficients.

Note $\hat{\beta}^{\text{lasso}}(0) = \hat{\beta}$ and $\hat{\beta}_0^{\text{lasso}}(\lambda) \to \bar{Y}$ for λ getting large (and $\beta_j \to 0$ for $j \neq 0$).

Alternative formulation

The lasso solution can be written as

$$\hat{\beta}^{\text{lasso}} = \text{argmin}_{\beta} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j \right)^2,$$
$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq t \tag{1}$$

Lasso is also known as basis pursuit.

Unfortunately, there is no closed form solution, but efficient computational methods exist (quadratic programming), same computational effort order as for ridge regression.

What's the difference between lasso and ridge?

The penalties: $\sum \beta_j^2$ and $\sum |\beta_j|$.

E.g. if β_j is small, then β_j^2 is tiny, and β_j^2 does not contribute much to the penalty, and this means β_j is not forced to become smaller to relieve the penalty.

However, in lasso, $|\beta_j|$ could be considerably bigger than β_j^2, and it contributes more to the penalty. Hence, it is worth shrinking it more severely to relieve the penalty.

Indeed, some β_j get shrunk so much, they get shrunk to exactly zero, and contribute nothing to the penalty.

So, lasso has the ability to select (remove) variables (set coefficients equal to zero) as well as shrink the coefficients.

Values of t: the penalty bound

Let $t_0 = \sum_{j=1}^{p} |\hat{\beta}|$, where $\hat{\beta}$ are the least squares estimators.

If t is larger than t_0 in the lasso constraint (1), then the $\hat{\beta}^{\text{lasso}} = \hat{\beta}^{\text{ls}}$. This is because the fidelity condition can be perfectly satisfied within the realm of the constraint.

Roughly speaking, if $t = t_0/2$, then the lasso estimates shrink the least squares ones by 50%.

However, the precise shrinkage is not always obvious.

The goal (in variable selection for least squares, λ in ridge regression or λ in lasso) is to choose the selection/λ to minimise the overall prediction error.

Comparison when X is orthogonal

It's possible to find closed form solutions, when X is orthogonal, i.e. $X^T X = I_p$.

For least squares

$$\hat{\beta}^{\text{ls}} = (X^T X)^{-1} X^T Y = X^T Y.$$

For ridge

$$\hat{\beta}^{\text{ridge}} = (I + \lambda I)^{-1} X^T Y = (1+\lambda)^{-1} X^T Y = \hat{\beta}^{\text{ls}}/(1+\lambda)$$

For the lasso, we want (remember $Y^T X\beta = \beta^T X^T Y$)

$$\min_{\beta} \frac{1}{2}(Y - X\beta)^T (Y - X\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$= \min_{\beta} \frac{1}{2} Y^T Y - Y^T X\beta + \frac{1}{2}\beta^T X^T X\beta + \lambda \sum_{j=1}^{p} |\beta_j|.$$

Lasso for orthogonal

We can ignore the $Y^T Y$ term as it does not depend on β and recall $X^T X = I$, so the problem becomes

$$\min_{\beta} -Y^T X\beta + \frac{1}{2}\beta^T\beta + \lambda\sum_{j=1}^{p}|\beta_j| \tag{2}$$

$$= \min_{\beta} -(\hat{\beta}^{\text{ls}})^T\beta + \frac{1}{2}\beta^T\beta + \lambda\sum_{j=1}^{p}|\beta_j| \tag{3}$$

$$= \min_{\beta} \sum_{j=1}^{p}\left(-\hat{\beta}_j^{\text{ls}}\beta_j + \frac{1}{2}\beta_j^2 + \lambda|\beta_j|\right). \tag{4}$$

The last line is a sum of terms, with each term involving one, and only one, β_j, so we can do the minimisation term by term, each over a separate β_j.

Minimising Lasso objective term by term for orthogonal X

We have to minimise

$$M_j = -\hat{\beta}_j^{\text{ls}} \beta_j + \frac{1}{2}\beta_j^2 + \lambda|\beta_j|. \tag{5}$$

Case a: Let $\hat{\beta}_j^{\text{ls}} > 0$.

Suppose $\beta_j < 0$. However, we can achieve a smaller M_j by swapping β_j with $-\beta_j$. This swap leaves β_j^2 and $|\beta_j|$ unchanged. Hence, $\beta_j \geq 0$.

Take derivative of M_j and set equal to zero (with $\beta_j \geq 0$)

$$\begin{aligned} \frac{\partial M_j}{\partial \beta_j} &= -\hat{\beta}_j^{\text{ls}} + \hat{\beta}_j^{\text{lasso}} + \lambda = 0 \\ \implies \quad & \hat{\beta}_j^{\text{lasso}} = \hat{\beta}_j^{\text{ls}} - \lambda. \end{aligned}$$

Minimising Lasso for orthogonal X — 2

$$\hat{\beta}_j^{\text{lasso}} = \hat{\beta}_j^{\text{ls}} - \lambda.$$

This quantity only feasible if ≥ 0, so solution is

$$\begin{aligned} \hat{\beta}_j^{\text{lasso}} &= (\hat{\beta}_j^{\text{ls}} - \lambda)^+ \\ &= \text{sgn}(\hat{\beta}^{\text{ls}})(|\hat{\beta}_j^{\text{ls}}| - \lambda)^+, \end{aligned} \tag{6}$$

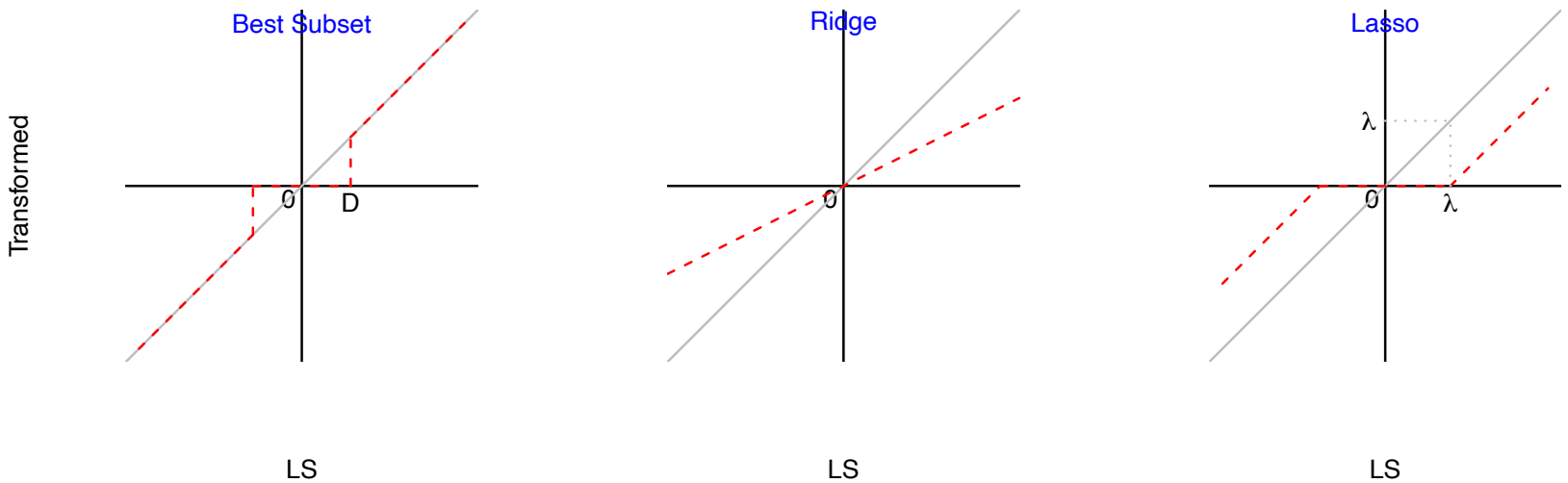
where $x^+ = x\mathbb{I}(x > 0)$ and $\text{sgn}(x) = 2\mathbb{I}(x > 0) - 1$ or

$$\text{sgn}(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0. \end{cases}$$

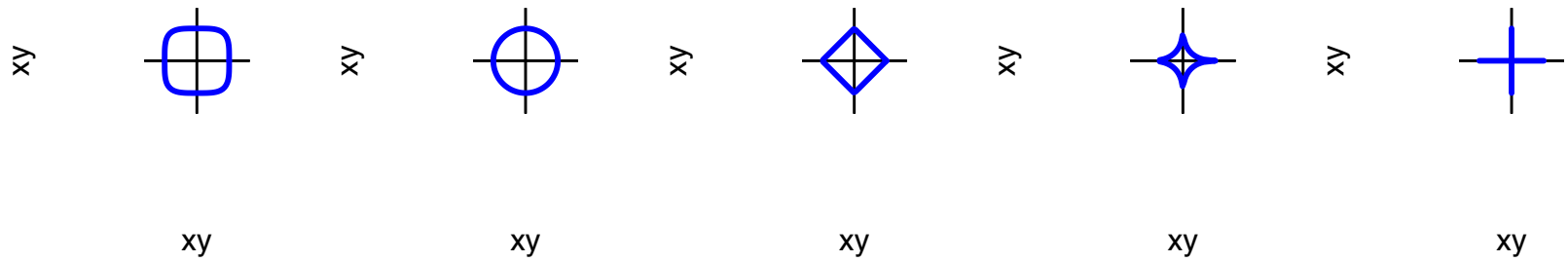
Operation (6) is called *soft thresholding*: we'll see it again later.

Case b: Let $\hat{\beta}^{\text{ls}} < 0$ (see homework).

How different methods modify coefficients



Bayesian formulation: Contours of prior distribution on β



Figures indicate how the β's start off in the prior.

Axes are β_1, β_2. Plots are equal contour of $|\beta_1|^q + |\beta_2|^q = 1$.
Values of q are (left to right) $q = 4, 2, 1, 0.5, 0.1$.

See Fig 3.11 in ELS — contours of error distribution are more likely to touch sharp corners of $q = 0$ then the flat faces (the corners 'stick' out), which correspond to a coefficient being exactly zero.

NOT EXAMINABLE 2023. Least Angle Regression

Has algorithmic similarity to forward stepwise regression.

Recall, this starts with nothing, and then adds variables to the active set one at a time.

Actually turns out to be a close cousin of the lasso, and very useful for understanding lasso AND computing it efficiently.

NOT EXAMINABLE 2023: Least Angle Regression: Summary

Enters variables one at a time, but only partially enters them, depending on how good they are!

That is, imagine the starting coefficient vector β to be all zeroes, and then, for the entered variable, gradually increase its contribution from zero up to its (usual) least squares coefficient.

Which one do we enter first? The variable that is most correlated with the response.

As the coefficient increases, more of the variable is entered. Progressively, the correlation of the variable with the residual will decrease (as more of the variable is in the model, not in the residual).

As soon as the first variable's residual correlation dips below another variable, we start entering a second variable, and continue.

NOT EXAMINABLE 2023: LAR Algorithm (ELS Alg 3.2)

1. Standardise the predictors to have zero mean and unit norm. Start with the residual $r = Y - \bar{Y}, \beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor X_j most correlated with r
3. Move β_j from 0 to its least-squares coefficient $\hat{\beta}_j =< X_j, r >$, until some other competitor X_k has as much correlation with the current residual as does X_j.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on (X_j, X_k), until some other competitor X_ℓ has as much correlation with the current residual.
5. Continue in this way until all p predictors have been entered. After $\min(n-1, p)$ steps, we get to the least squares solution.

NOT EXAMINABLE 2023: LAR: Details

Suppose $\mathcal{A}_k$ is the active set at the beginning of the kth step.

Let $\beta_{\mathcal{A}_k}$ be the coefficient vector for these variables at this step. There will be $k-1$ non-zero values, the one just entered will be 0.

The current residual is $r_k = Y - X_{\mathcal{A}_k}\beta_{\mathcal{A}_k}$

The direction to travel in for this step is

$$\delta_k = (X_{\mathcal{A}_k}^T X_{\mathcal{A}_k})^{-1} X_{\mathcal{A}_k}^T r_k.$$

Then evolve the coefficients as

$$\beta_{\mathcal{A}_k}(\alpha) = \beta_{\mathcal{A}_k} + \alpha \cdot \delta_k.$$

NOT EXAMINABLE 2023: LAR: Details 2

Thus the fit at step k, $\hat{f}_k$ evolves as

$$\hat{f}_k(\alpha) = \hat{f}_k + \alpha \cdot u_k,$$

where $u_k = X_{\mathcal{A}_k} \delta_k$.

The name "least angle" stems from the fact that u_k makes the smallest and equal angle with each of the predictors in $\mathcal{A}_k$

NOT EXAMINABLE 2023: LAR: Algorithm info

The LAR coefficients evolve in a piecewise linear fashion.

That's because coefficient(s) evolve linearly moving towards a value and then jump to a new 'state' when a new competitor is about to beat the active set.

It turns out that you don't need to move in small steps. You can work out what the next competitor is and what direction it needs to go in, and the amount it needs to move.

This means that LAR is extremely computationally efficient, as efficient as standard least squares.

NOT EXAMINABLE 2023: LAR Correlation Progression

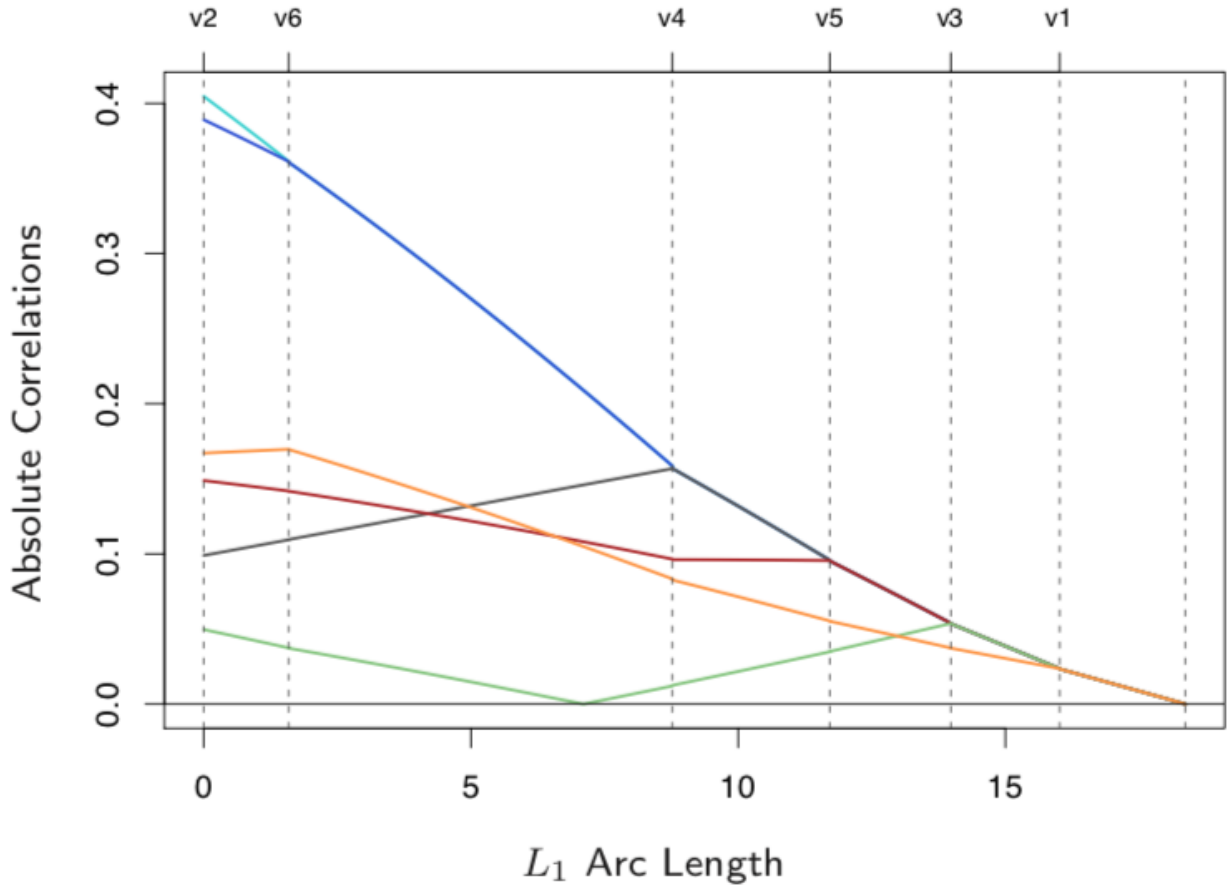


FIGURE 3.14. *Progression of the absolute correlations during each step of the LAR procedure, using a simulated data set with six predictors. The labels at the top of the plot indicate which variables enter the active set at each step. The step length are measured in units of L_1 arc length.*

LAR PART NOT EXAMINABLE 2023: LAR/Lasso

Coefficient Progression

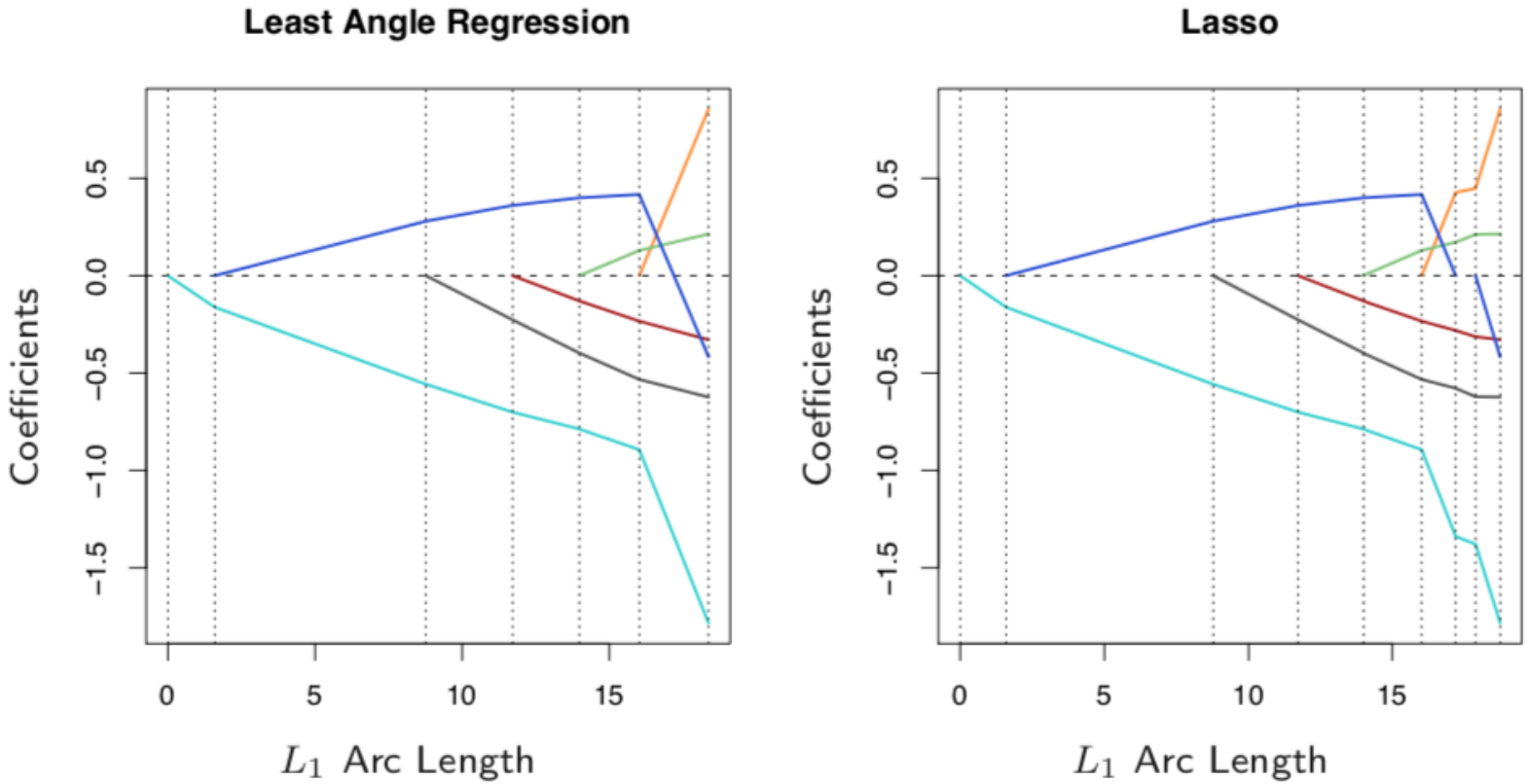


FIGURE 3.15. *Left panel shows the LAR coefficient profiles on the simulated data, as a function of the L_1 arc length. The right panel shows the Lasso profile. They are identical until the dark-blue coefficient crosses zero at an arc length of about* 18.

Principal Components Regression

From Lecture 3 we projected the data matrix onto the columns of V from the SVD — the eigenvectors of the sample covariance matrix.

Let $z_m = Xv_m$, $m = 1, \ldots, p$.

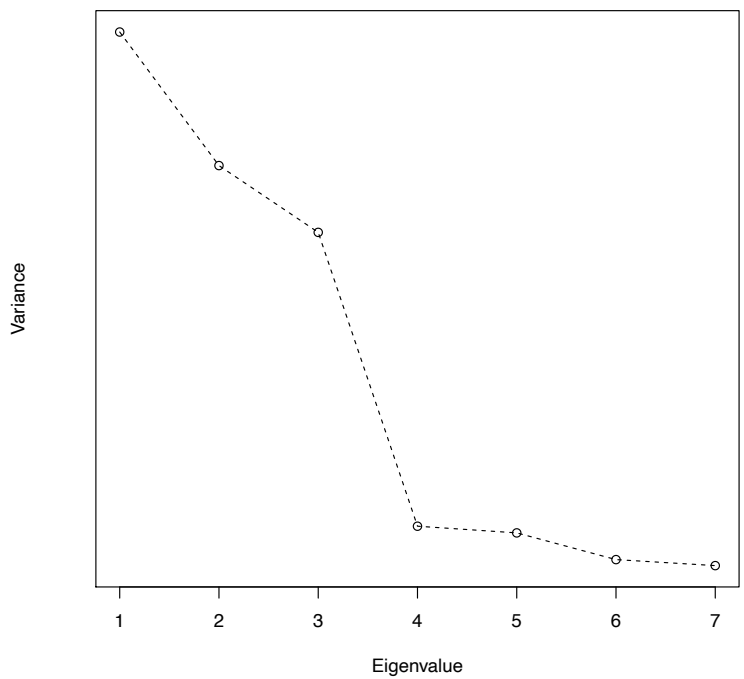
Recall that principal components is just a change of basis (actually a rotation since $VV^T = I_p$) and it presents the data according to a new set of variables, $z_1, \ldots, z_p$, where the variance of the data according to z_1 is greatest, and progressively smaller variances.

An interpretation of this is that the *most* information(=variance)in the data set is put into z_1; the next largest variance, in a direction orthogonal to v_1, gets put into z_2, the next largest, in a direction orthogonal to v_1 and v_2 is put into z_3, etc.

Principal Components Regression: scree plot

Often, the variance of the last principal components can be very small.

The variances are just the eigenvalues and can just be plotted against m. We usually select a M so that variances, $m < M$ are 'big' and those $m \geq M$ are 'small' .



Here, we might choose $M = 4$.

Principal Components Regression

PC regression can be written as the sum of univariate regressions (since the regression variables are orthogonal) as

$$\hat{y}^{\text{pcr}}_{(M)} = \bar{Y}1_n + \sum_{m=1}^{M} \hat{\theta}_m z_m, \tag{7}$$

where $\hat{\theta}_m =< z_m, y > / < z_m, z_m >$, the usual regression coefficient, but using $\{z_m\}_{m=1}^{M}$ variables not the Xs. Since $z_m = Xv_m$ and $\hat{\theta}_m$ is a scalar we can write:

$$\hat{y}^{\text{pcr}}_{(M)} = \bar{Y}1_n + \sum_{m=1}^{M} \hat{\theta}_m X v_m = \bar{Y}1_n + X \sum_{m=1}^{M} \hat{\theta}_m v_m, \tag{8}$$

so we can think of $\hat{\beta}^{\text{pcr}} = \sum_{m=1}^{M} \hat{\theta}_m v_m$.

Comments on Principal Components Regression

Usually operate principal components regression on scaled inputs.

If $M = p$ we get back to usual least-squares (since column space of both is same, just a rotation).

Principal components regression has similarities to ridge regression. The latter shrinks the coefficients of the principal components (the d_j) especially the small ones. Whereas principal components discards the $p - M$ smallest ones.

Read §3.5.2 of ESL, page 80 on *Partial Least Squares*.

Summary

This lecture discussed:

- The Lasso
- Least angle regression
- Principal Components Regression

Elements of Statistical Learning: Lecture 5. Regression Real Examples

Guy Nason[1]

Department of Mathematics
Imperial College

Regression with the Swiss Data Set

Some of these examples for this lecture originate from the R-bloggers article "Ridge Regression and the Lasso" by `realdataweb` at

`https://www.r-bloggers.com/ridge-regression-and-the-lasso/`

The `swiss` data set contains a response: a standardized fertility measure and five other explanatory variables (listed on next but one page) obtained from each of 47 French-speaking provinces of Switzerland in (around) 1888.

We will analyse this data set using linear modelling, ridge regression, the lasso and least angle regression.

Language distribution of Switzerland

Variables of `swiss` data set

You can find out for yourself using the `?swiss` help command in R. However, they are:

Fertility	common standardized fertility measure
Agriculture	% of males involved in agriculture
Examination	% draftees receiving highest mark on army exam
Education	% education beyond primary school for draftees
Catholic	% catholic (as opposed to protestant)
Infant.Mortality	live births who live less than one year

All variables, apart from Fertility are as percentage of population and thus already somewhat standardized.

In 1888 Switzerland was entering a period known as the 'democratic transition', where fertility was beginning to fall from the high level typical of underdeveloped countries.

Source and Bibliographic Details

It is always important to acknowledge the source of the data. The data come from Project "16P5" from pages 549–551 in Mosteller, F. and Tukey, J. W. (1977) *Data Analysis & Regression: A Second Course in Statistics*. Addison-Wesley, Reading Mass.

Data inclusion statement: "Data used by permission of Franice van de Walle. Office of Population Research, Princeton University, 1976. Unpublished data assembled under NICHD contract number No 1-HD-O-2077."

WARNING: This is a HISTORICAL data set, collected in times that were different to today. One might argue that there is nothing ostensibly wrong with this data set or collection. There may be sensitivity to certain variables, models, assumptions and so this lecture is not intended to be prescriptive or suggest that the data are good or bad in a moral sense. However, most national statistical offices, e.g. UK Office for National Statistics still collect 'religion' in the Census. E.g. 400k people registered as Jedi in the 2001 Census

Modern Data

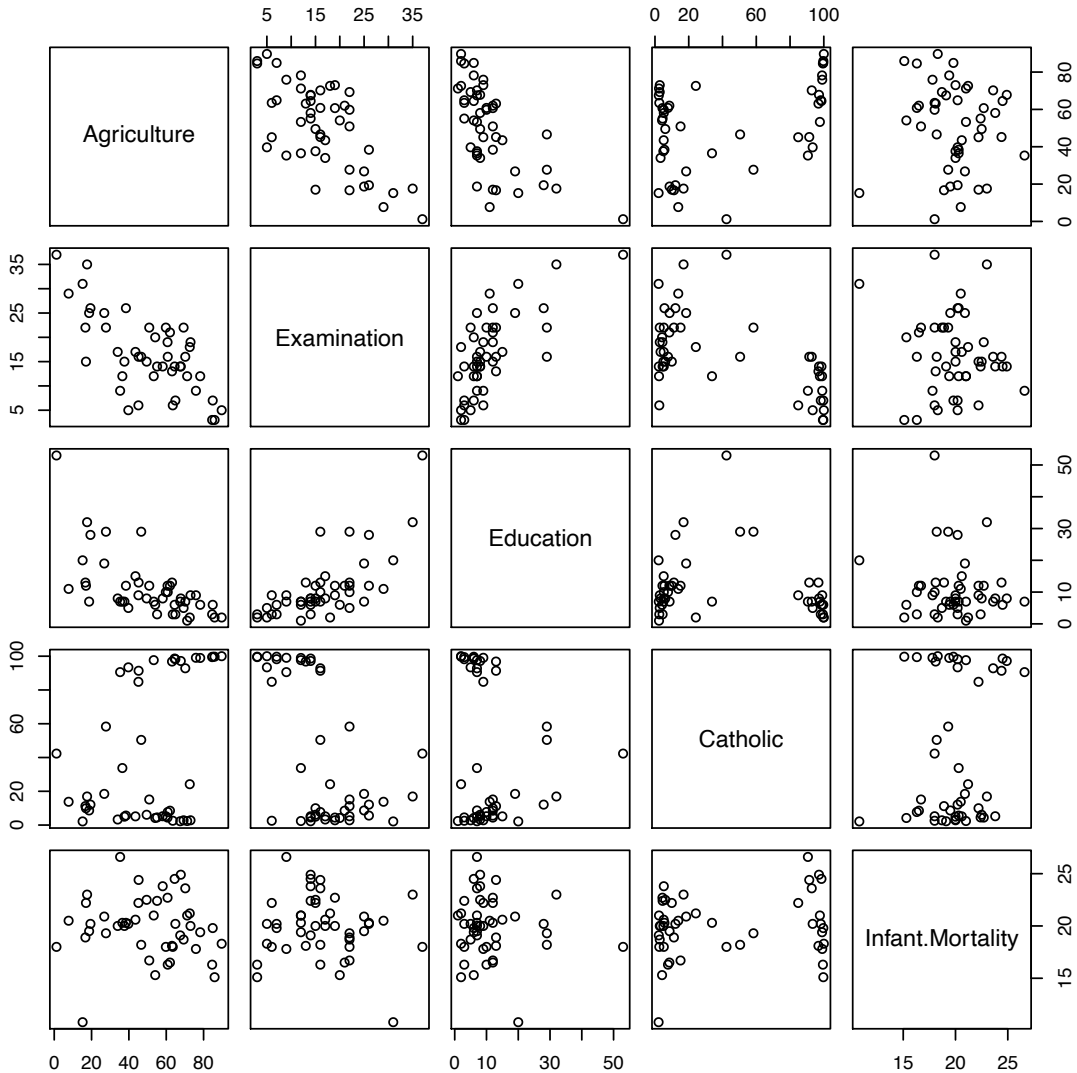
Before we start see:

`https://ourworldindata.org/grapher/fertility-vs-child-mortality`

Now back to our data ...

Exploratory plots

```
attach(swiss)
pairs(~Agriculture+Examination+Education+Catholic+Infant.Mortality)
```



Explanatory variables

Some reasonable correlations may appear, e.g.

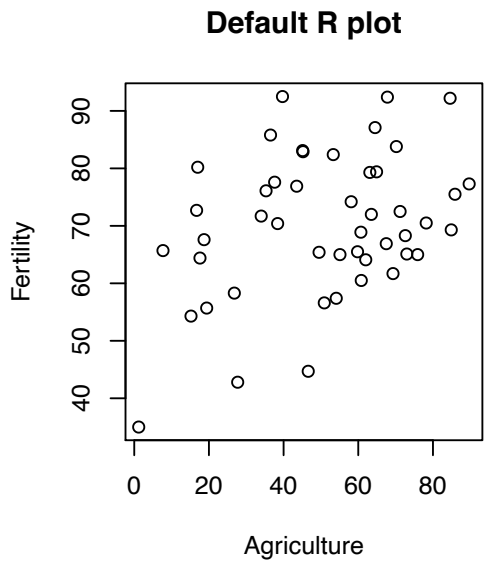
```
cor(Agriculture, Examination)
[1] -0.6865422 cor(Education, Examination)
[1] 0.6984153
```

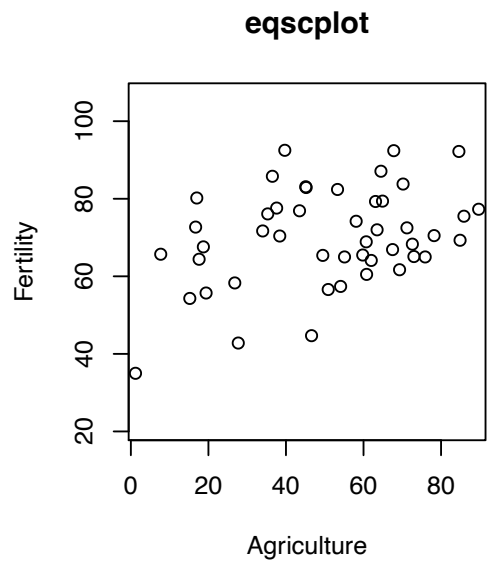
So, clearly some non-trivial correlations between explanatory variables.

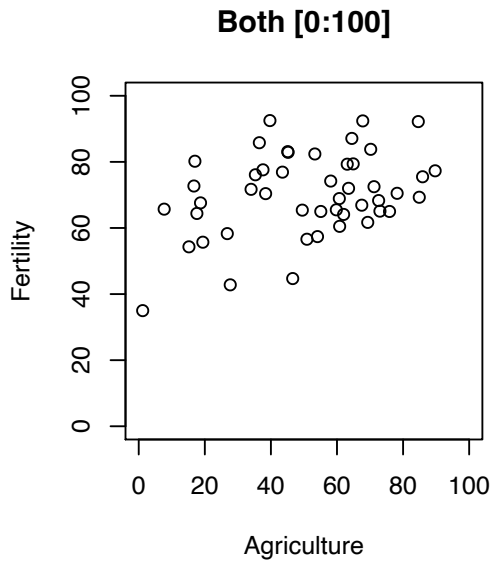
Potentially Deceptive Plots: Watch the scale

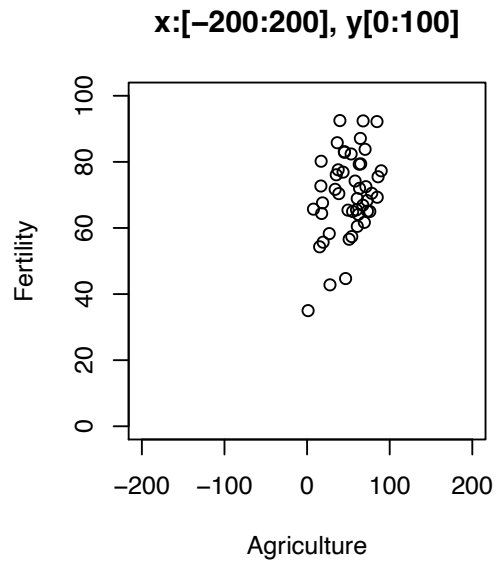
```
oldpar <- par(mfrow=c(2,2), pty="s")

plot(Agriculture, Fertility, main="Default R plot")
eqscplot(Agriculture, Fertility, main="eqscplot")
plot(Agriculture, Fertility,
    xlim=c(0, 100), ylim=c(0,100), main="Both [0:100]")
plot(Agriculture, Fertility,
    xlim=c(-200, 200), ylim=c(0,100),
    main="x:[-200:200], y[0:100]")

par(oldpar)
```

Potentially Deceptive Plots: THE SAME DATA



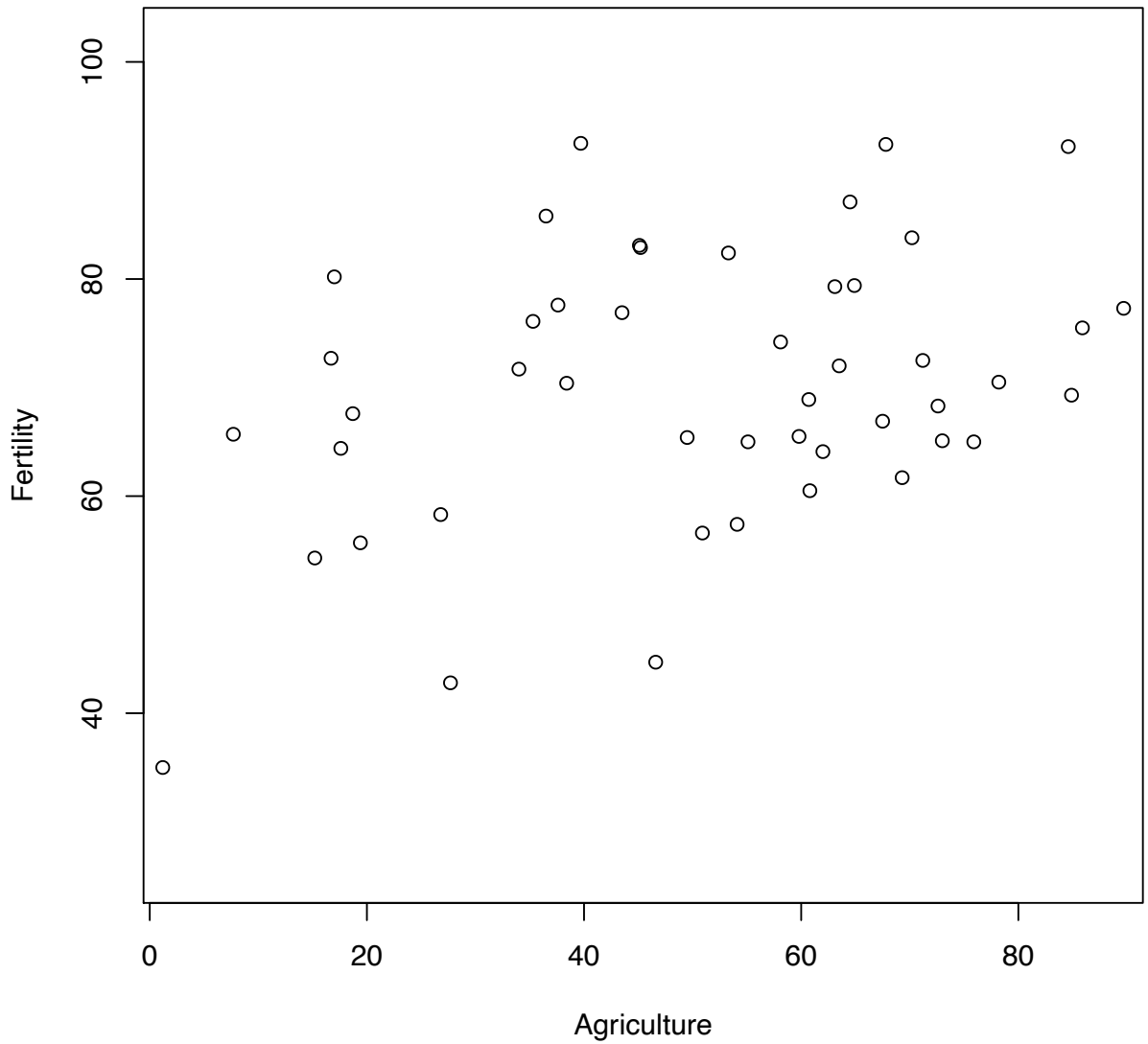






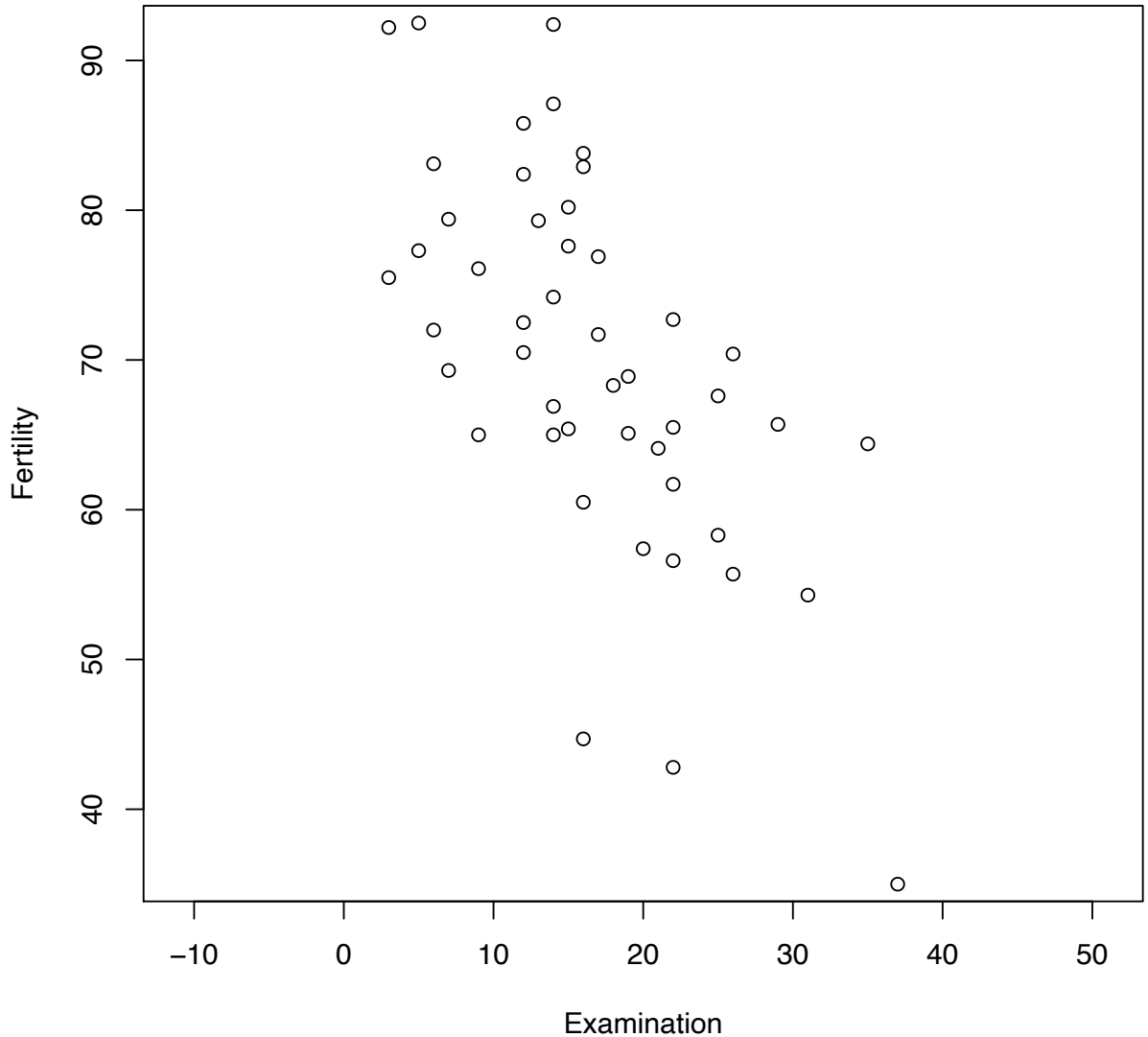
Response versus Explanatory: Agriculture

```
library("MASS")
eqscplot(Agriculture, Fertility)
```



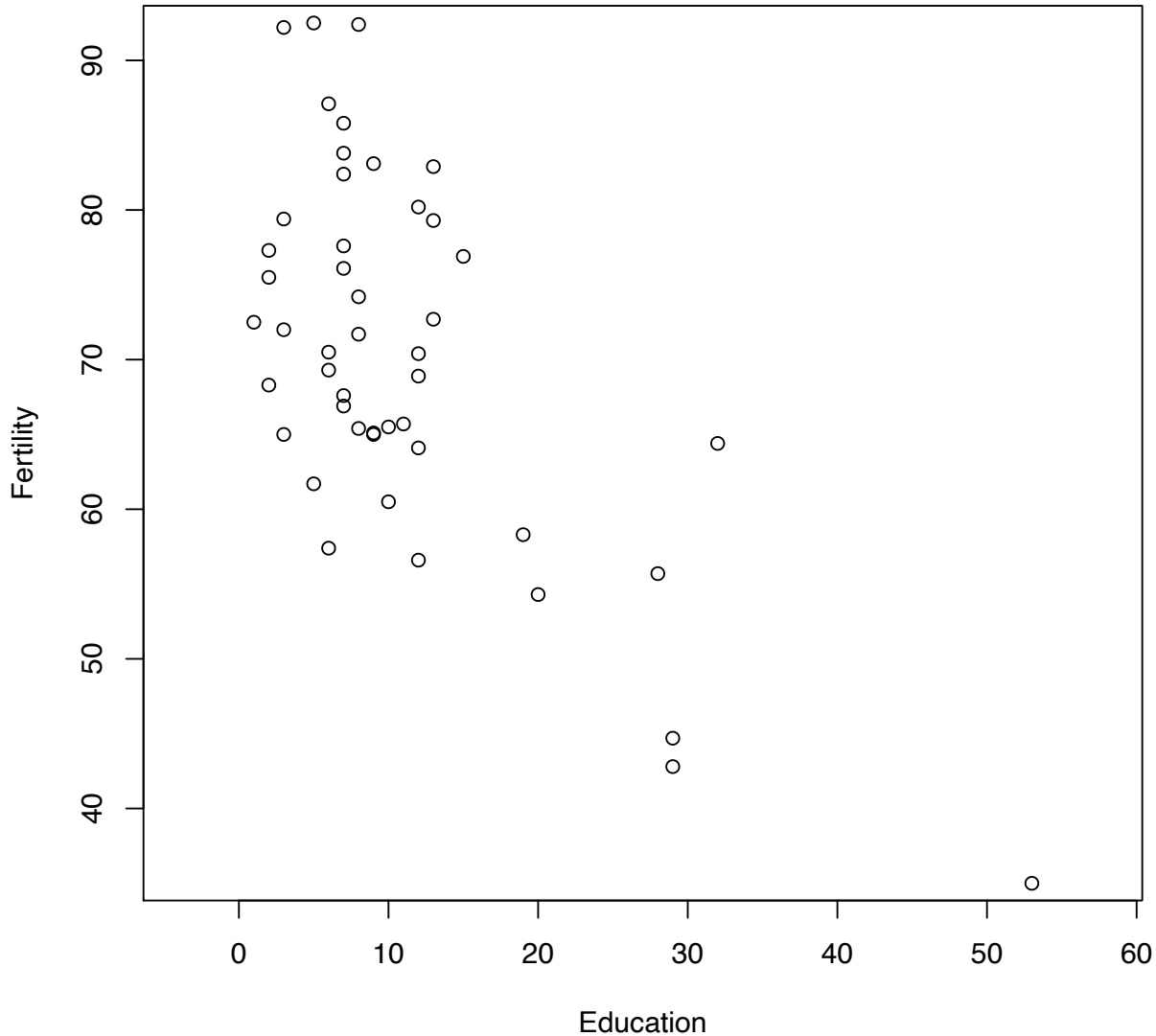
Response versus Explanatory: Examination

```
eqscplot(Examination, Fertility)
```



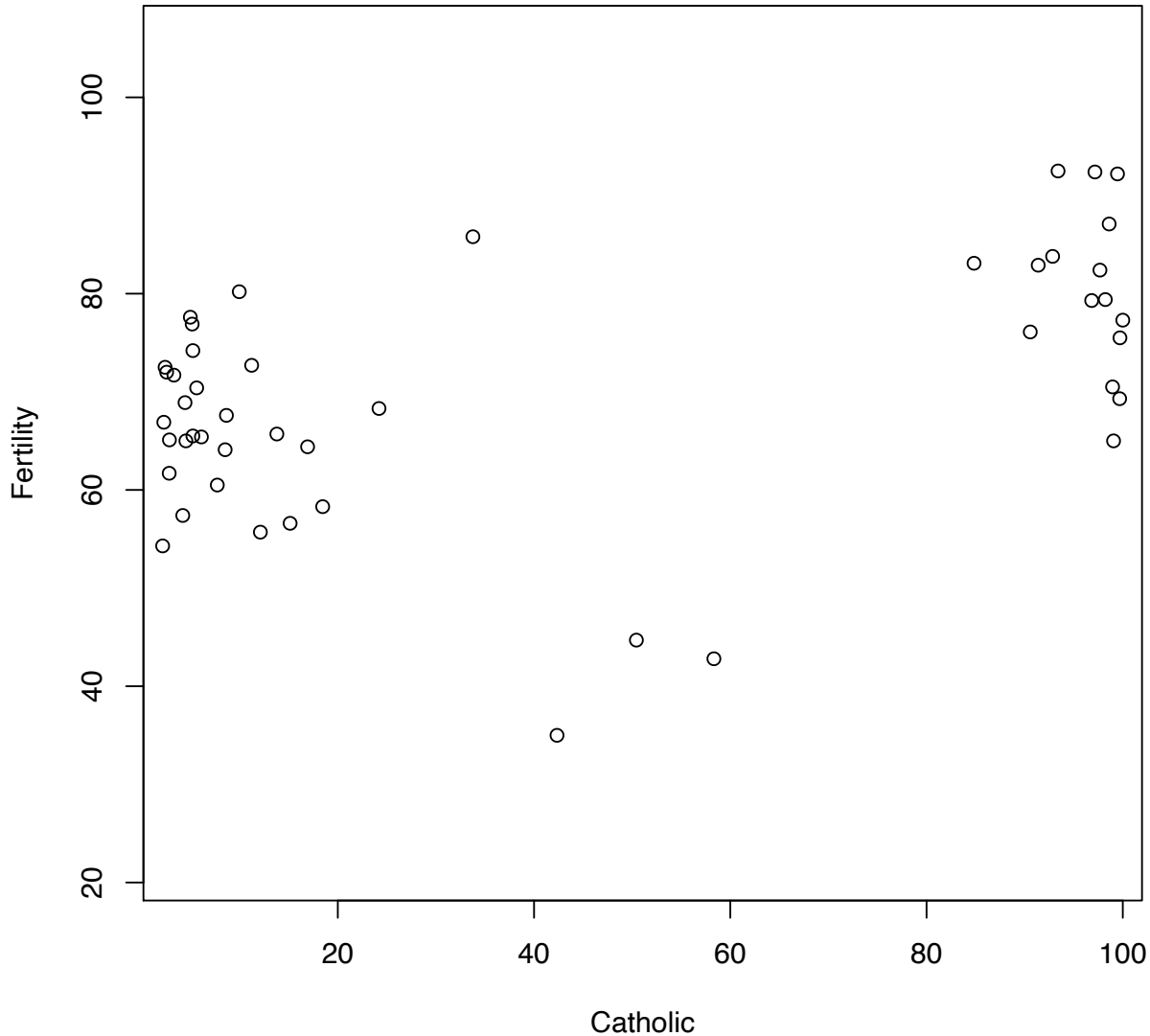
Response versus Explanatory: Education

```
eqscplot(Education, Fertility)
```



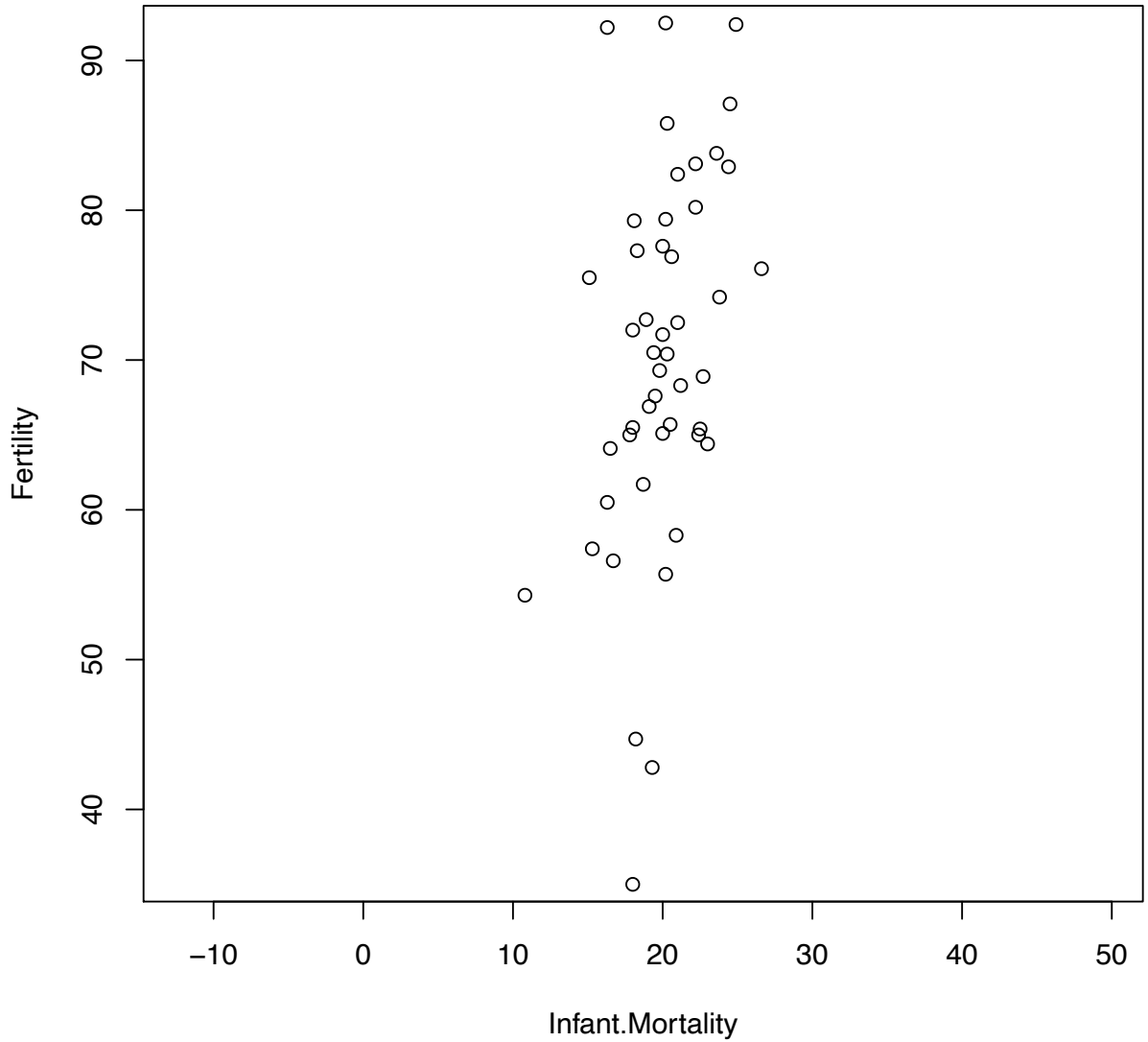
Response versus Explanatory: Catholic

```
eqscplot(Catholic, Fertility)
```



Response versus Explanatory: Infant Mortality

```
eqscplot(Infant.Mortality, Fertility)
```



Comments on Graphs for Modelling

The `eqscplot` is useful as it plots the variables on equal scales, which makes it easier to assess relationships.

There are some reasonable looking linear relationships (Fertility on Examination, Fertility on Education, and, possible, Fertility on Agriculture).

The Fertility on Catholic — less so.

Seems like Infant.Mortality & Fertility are not linked.

However, the variables themselves are reasonably highly correlated — so the relationships are not necessarily separate and independent.

However, we feel confident in fitting a simple linear model.

Fit least squares linear model

```
swiss.lm <- lm(Fertility ~ Agriculture+Examination+
    Education+Catholic+Infant.Mortality)

summary(swiss.lm)

Call:
lm(formula = Fertility ~ Agriculture + Examination +
    Education + Catholic + Infant.Mortality)

Residuals:
     Min       1Q   Median       3Q      Max
-15.2743  -5.2617   0.5032   4.1198  15.3213
```

Comment on residuals

R linear model output

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       66.91518   10.70604   6.250 1.91e-07 ***
Agriculture       -0.17211    0.07030  -2.448  0.01873 *
Examination       -0.25801    0.25388  -1.016  0.31546
Education         -0.87094    0.18303  -4.758 2.43e-05 ***
Catholic           0.10412    0.03526   2.953  0.00519 **
Infant.Mortality   1.07705    0.38172   2.822  0.00734 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.165 on 41 degrees of freedom
Multiple R-squared:  0.7067,Adjusted R-squared:  0.671
F-statistic: 19.76 on 5 and 41 DF,  p-value: 5.594e-10
```

F-stat indicates 'meaningful' explanation for group. Drop Examination, probably too highly correlated with other variables to provide independent explanation

Remove Examination variable from model and re-fit

```
swiss.lm2 <- update(swiss.lm, . ~ . -Examination)
summary(swiss.lm2)

Call:
lm(formula = Fertility ~ Agriculture + Education + Catholic +
    Infant.Mortality)

Residuals:
     Min       1Q   Median       3Q      Max
-14.6765  -6.0522   0.7514   3.1664  16.1422
```

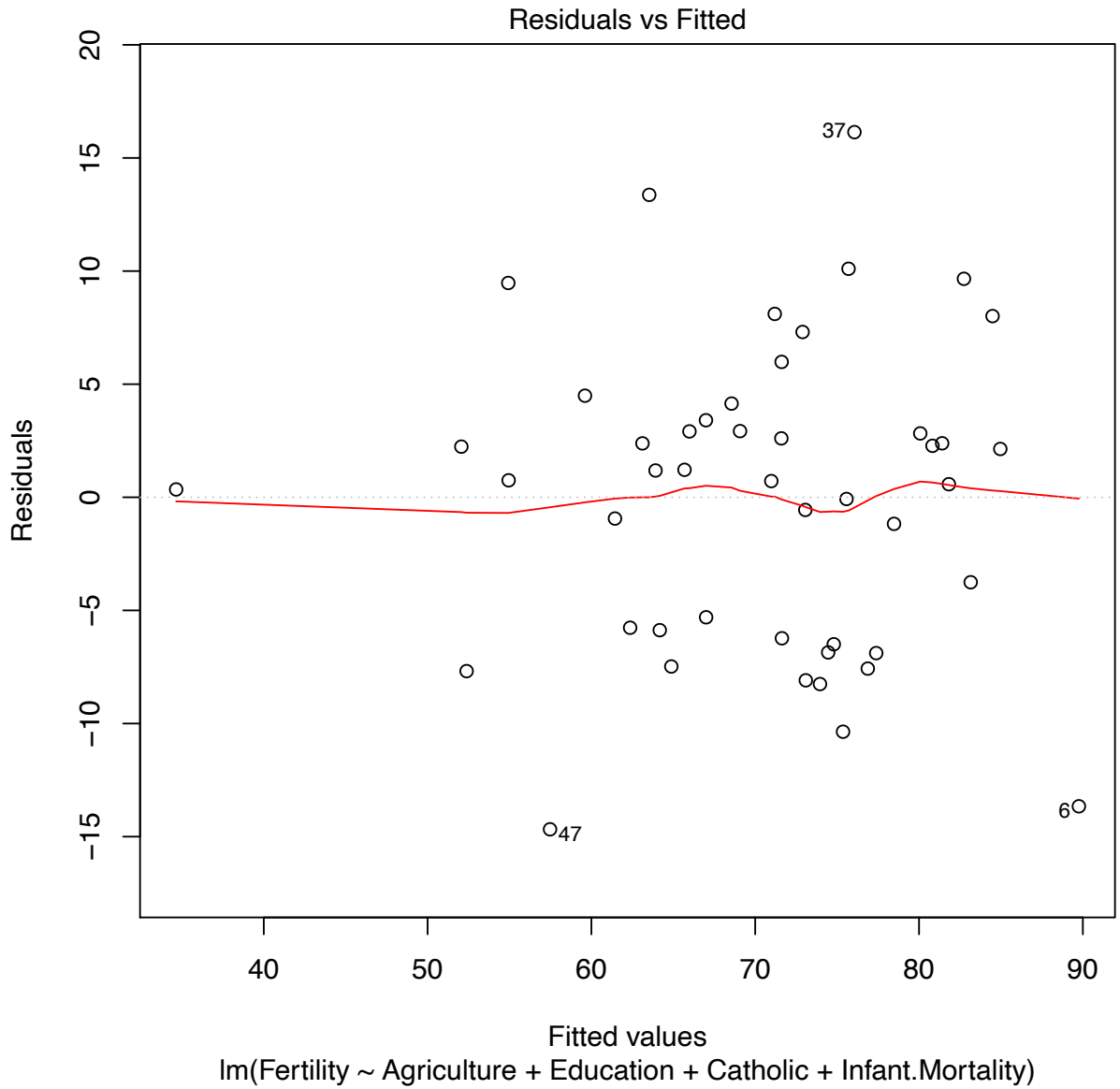
Remove Examination and re-fit: contd

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       62.10131    9.60489   6.466 8.49e-08 ***
Agriculture       -0.15462    0.06819  -2.267  0.02857 *
Education         -0.98026    0.14814  -6.617 5.14e-08 ***
Catholic           0.12467    0.02889   4.315 9.50e-05 ***
Infant.Mortality   1.07844    0.38187   2.824  0.00722 **
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 7.168 on 42 degrees of freedom
Multiple R-squared:  0.6993,Adjusted R-squared:  0.6707
F-statistic: 24.42 on 4 and 42 DF,  p-value: 1.717e-10
```

All variables now statistically significant at least at the 3% level. The null hypothesis for each of these tests: $H_0 : \beta_{\text{Agriculture}} = 0$ versus $\neq$. Omnibus F-test also significant.

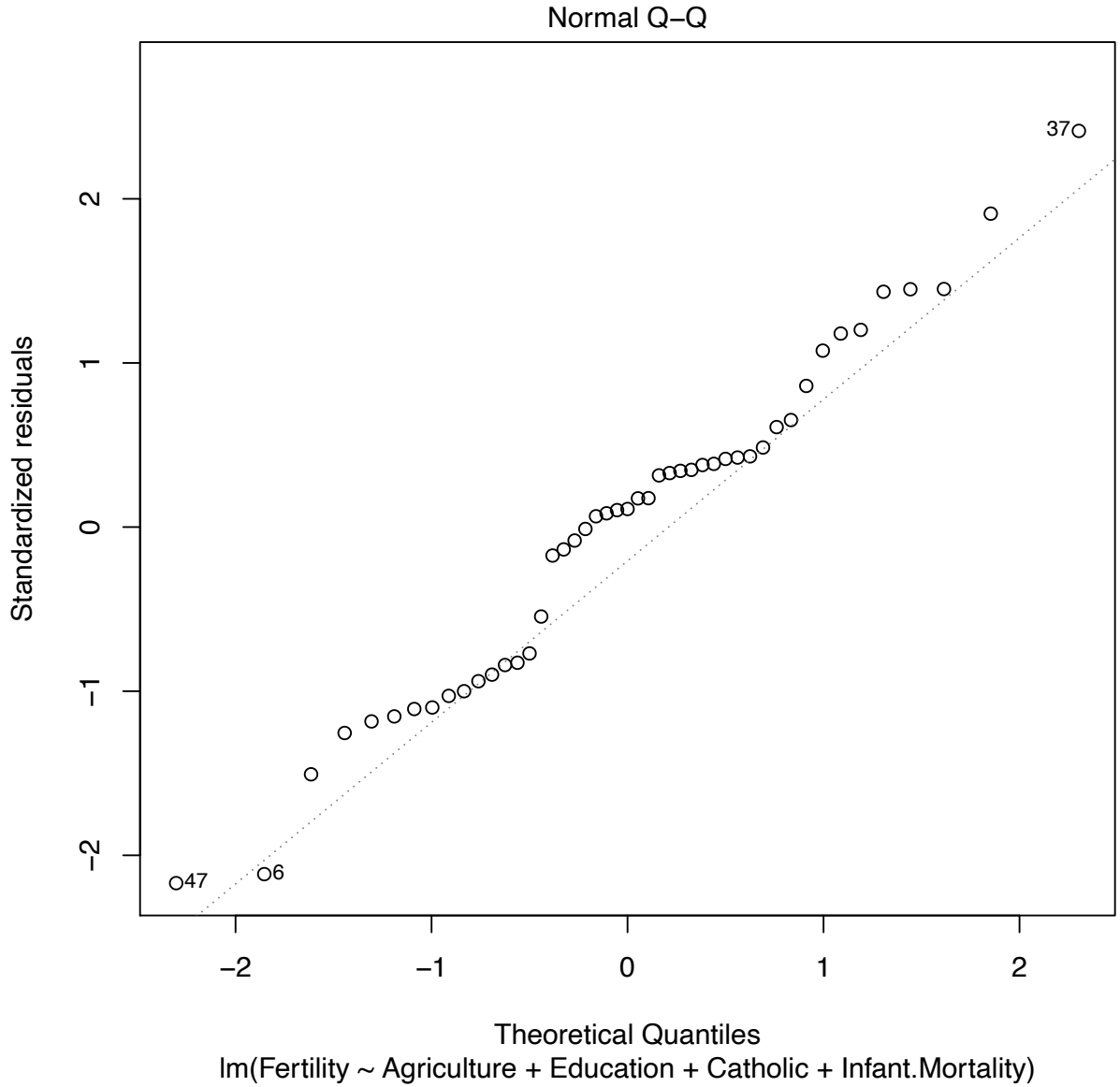
Diagnostic Plot: Residuals versus fitted

```
plot(swiss.lm2,  which=1)
```



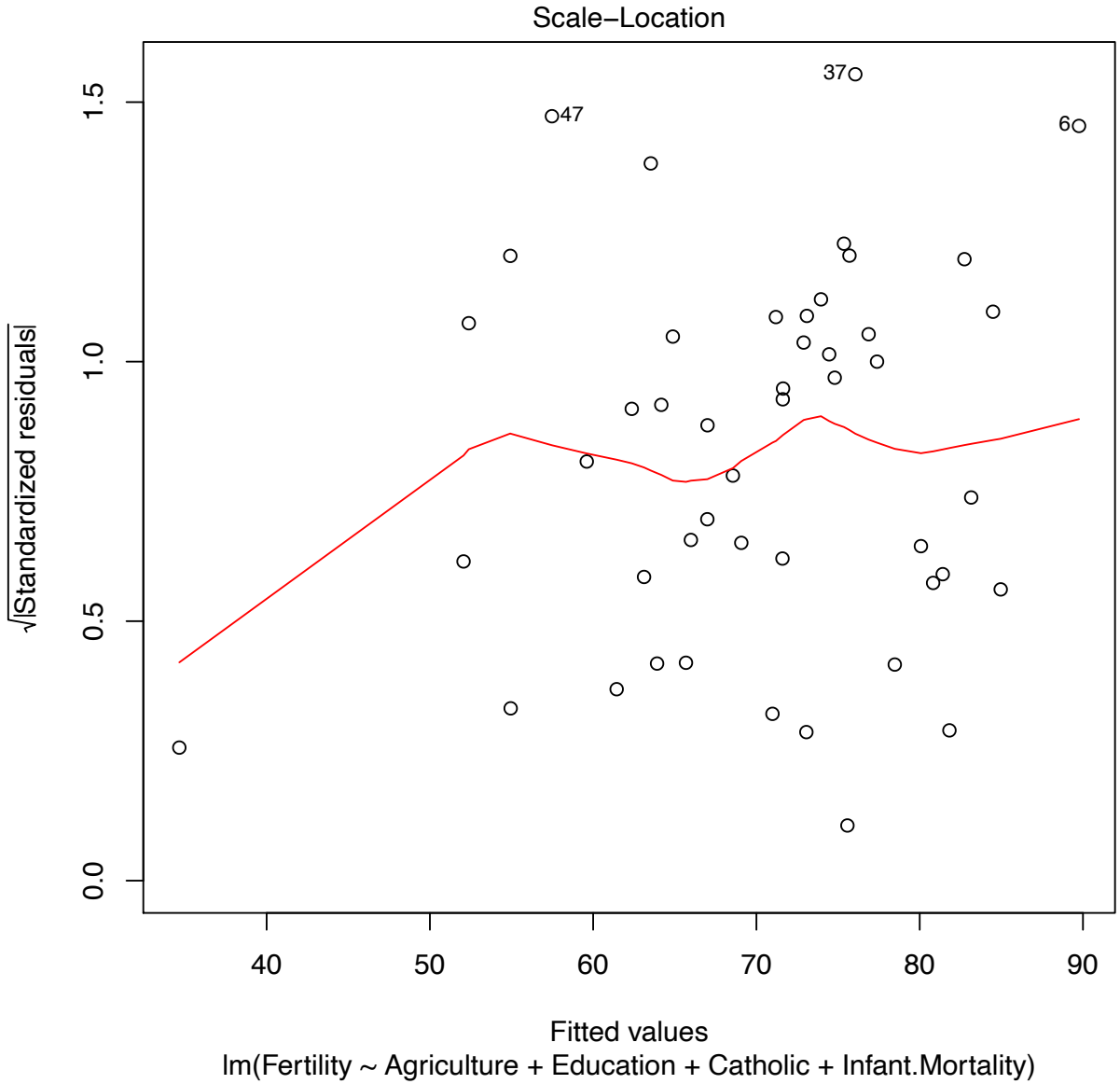
Diagnostic Plot: Residual Q-Q plot

```
plot(swiss.lm2,  which=2)
```



Diagnostic Plot: Standardised residuals versus fitted

```
plot(swiss.lm2,  which=3)
```



Comments on Linear Model

Actually, fit does not seem too bad.

Now, let's split the set into a *training* set and a *test* set.

Why? If you evaluate the results of a fit on the *same* set of data, then you are fooling yourself. As the model will be tuned to *that* set of data. To make it fair you should use completely separate training and test sets.

We will use the `glmnet` library to compute ridge and the Lasso. This uses matrices and vectors rather than data frames in R.

Construct the model matrix, response variable & a λ *sequence*.

```
x <- model.matrix(Fertility~., swiss)[,-1]
y <- swiss$Fertility
lambda <- 10^seq(from=10, to=-2, length=100)
```

Generating a Training and Test Sample

```
set.seed(183)
train <- sample( 1:nrow(x), nrow(x)/2)
test <- -train
ytest <- y[test]
```

`train` is a list of 23 indices from $1, \ldots, 47$. They correspond to 23 provinces on which we'll fit the models.

`test` is the complement of `train`, and gives us a way of excluding the training observations, leaving just the test provinces.

So `ytest` are the responses of those observations in the test set.

Fit Models to Training Set

First, fit least squares model to *just* the training data.

```
swiss.train.lm <- lm(Fertility~., data=swiss, subset=train)
```

We use all of the variables here. Now do the same for ridge

```
swiss.train.ridge <- glmnet(x[train,], y[train], alpha=0,
    lambda=lambda)
```

The `alpha` variable chooses ridge: `alpha=0` or lasso: `alpha=1`.

We can use cross-validation to find a 'good' λ:

```
cv.swiss.ridge <- cv.glmnet( x[train,], y[train], alpha=0)
bestlam <- cv.swiss.ridge$lambda.min
bestlam
[1] 0.6642065
```

Predicting

We will predict the response associated with (new) test set observations, based on models built from *training* set observations.

```
swiss.predict.lm <- predict(swiss.train.lm,
    newdata=swiss[test,])
swiss.ridge.pred <- predict(swiss.train.ridge,
    s=bestlam, newx=x[test,])
```

Here's a function to compute the (empirical) mean squared error

```
mse <- function(y1, y2) { sum((y1-y2)^2) }
```

Let's now compare the ridge estimate with the least squares

```
mse(ytest, swiss.predict.lm)
[1] 1980.309
mse(ytest, swiss.ridge.pred)
[1] 1893.207
```

So, ridge is doing better here.

The Coefficients

The coefficients from the linear model are easy to get

```
coef(swiss.train.lm)
```

but for the ridge, we use the `predict` function and have to tell it which λ we're interested in (here `bestlam`):

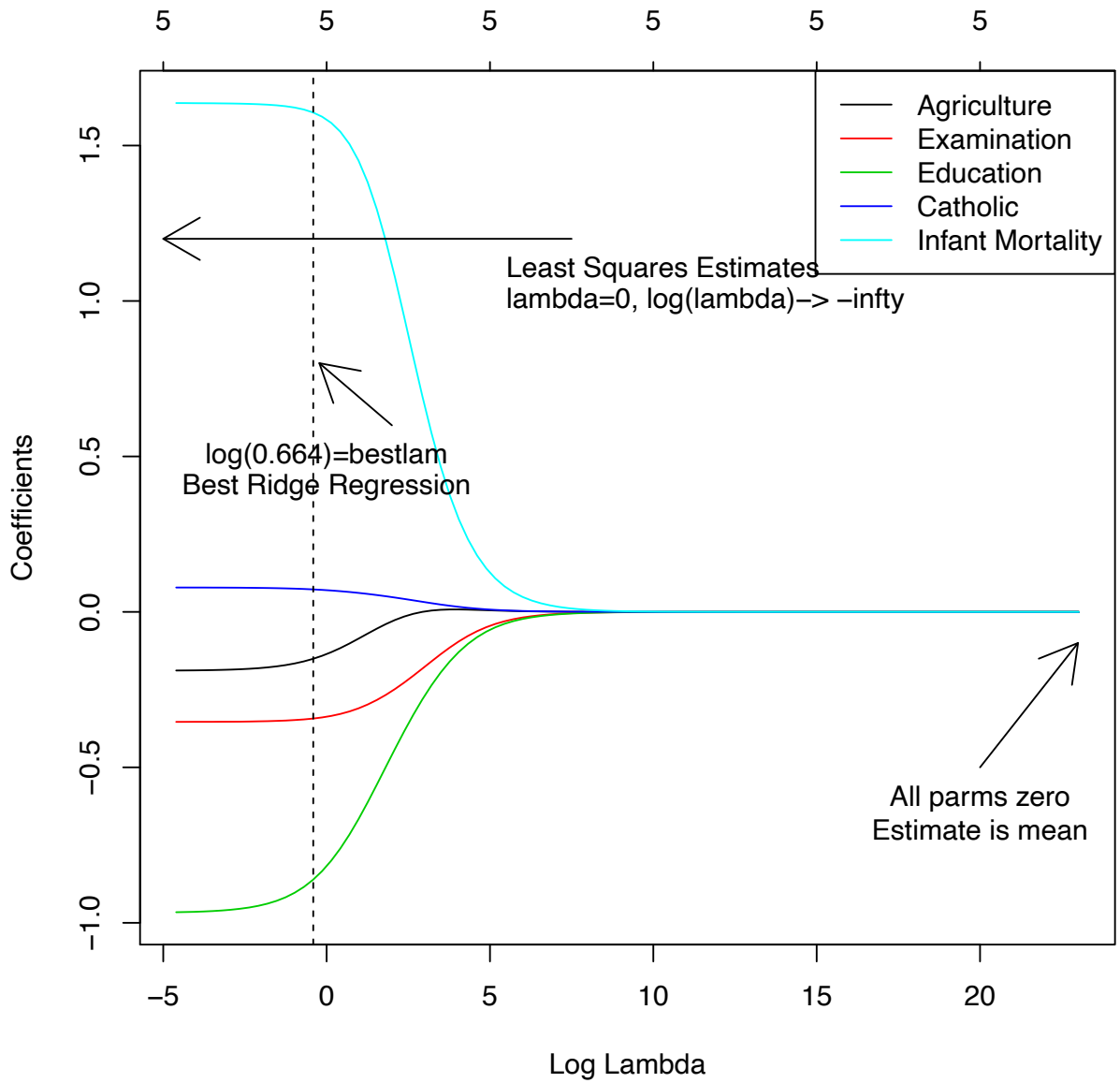
```
predict(swiss.train.ridge, type="coefficients", s=bestlam)
```

And put them in a nice table:

	Agri	Exam	Educ	Cath	InfMort
LS	-0.189	-0.354	-0.968	0.079	1.64
Ridge	-0.151	-0.343	-0.861	0.072	1.61

So, nicely shrunk (but not that much). Agriculture is shrunk the most, about 20%.

plot(swiss.train.ridge, xvar="lambda")



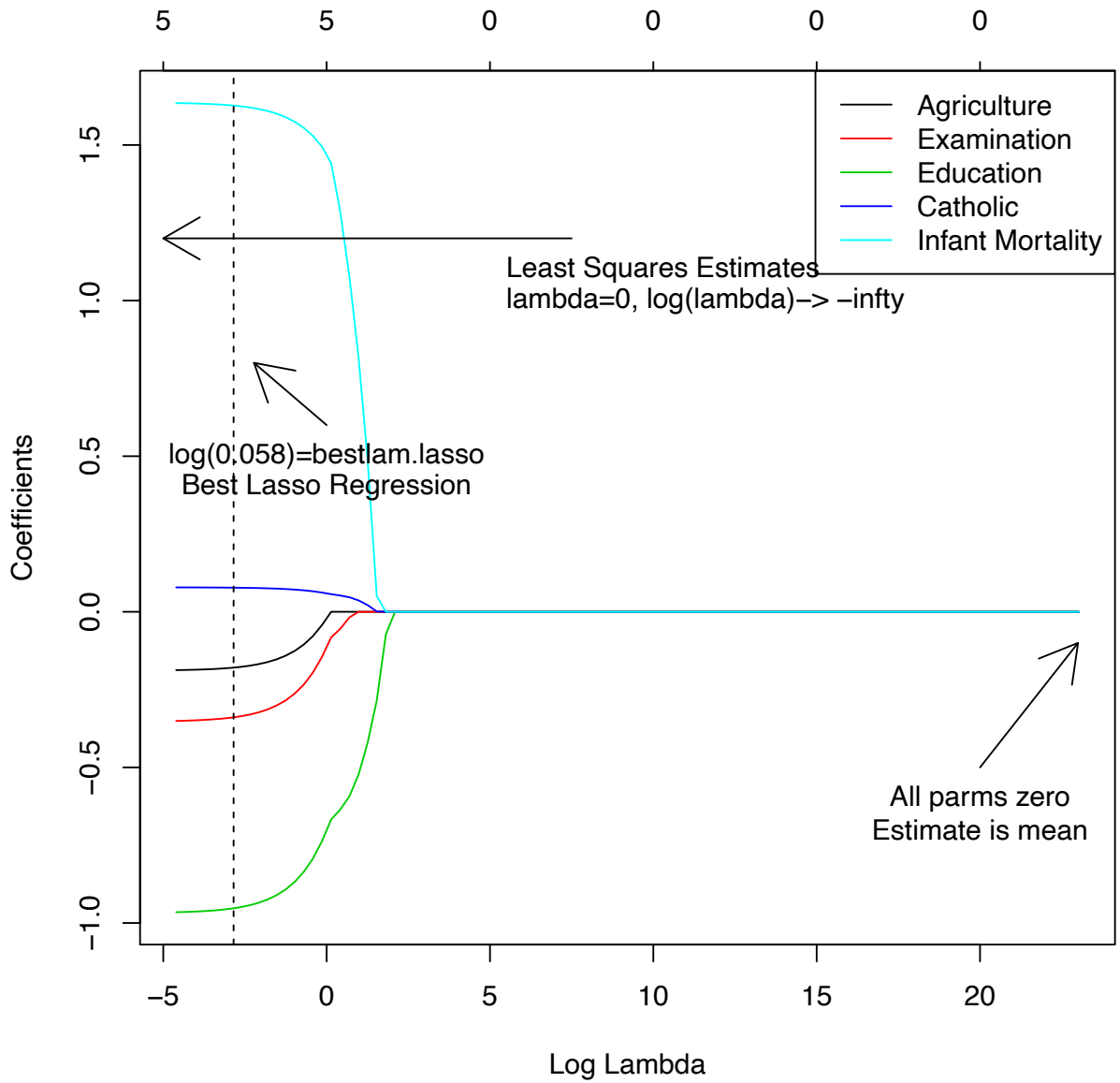
Lasso

Change `alpha=0` to `alpha=1`.

```
swiss.train.lasso <- glmnet(x[train,], y[train], alpha=1,
    lambda=lambda)
cv.lasso <- cv.glmnet(x[train,], y[train], alpha=1)
bestlam.lasso <- cv.lasso$lambda.min
bestlam.lasso
[1] 0.05776929

swiss.lasso.pred <- predict(swiss.train.lasso,
    s=bestlam.lasso, newx=x[test,])
mse(ytest, swiss.lasso.pred)
[1]  1972.31
```

Lasso is higher than for ridge, but less than least squares.

plot(swiss.train.lasso, xvar="lambda")



Principal Components of X

```
options(digits=3) # More sensible numbers of digits printed!

prcomp(x)
Standard deviations (1, .., p=5):
[1] 43.36 21.43  7.67  3.73  2.75

Rotation (n x k) = (5 x 5):
                     PC1     PC2      PC3     PC4     PC5
Agriculture       0.2815 -0.8838 -0.36962  0.0265  0.0486
Examination      -0.1207  0.1739 -0.44998  0.8669 -0.0332
Education        -0.0584  0.3108 -0.80696 -0.4849  0.1172
Catholic          0.9501  0.3029  0.00166  0.0715 -0.0223
Infant.Mortality  0.0105  0.0193  0.09853  0.0867  0.9911

cumsum(prcomp(x)$sdev^2)/sum(prcomp(x)$sdev^2)
[1] 0.777 0.967 0.991 0.997 1.000
```

So, about 97% variation is accounted for by first two PCs.

Principal Components Regression

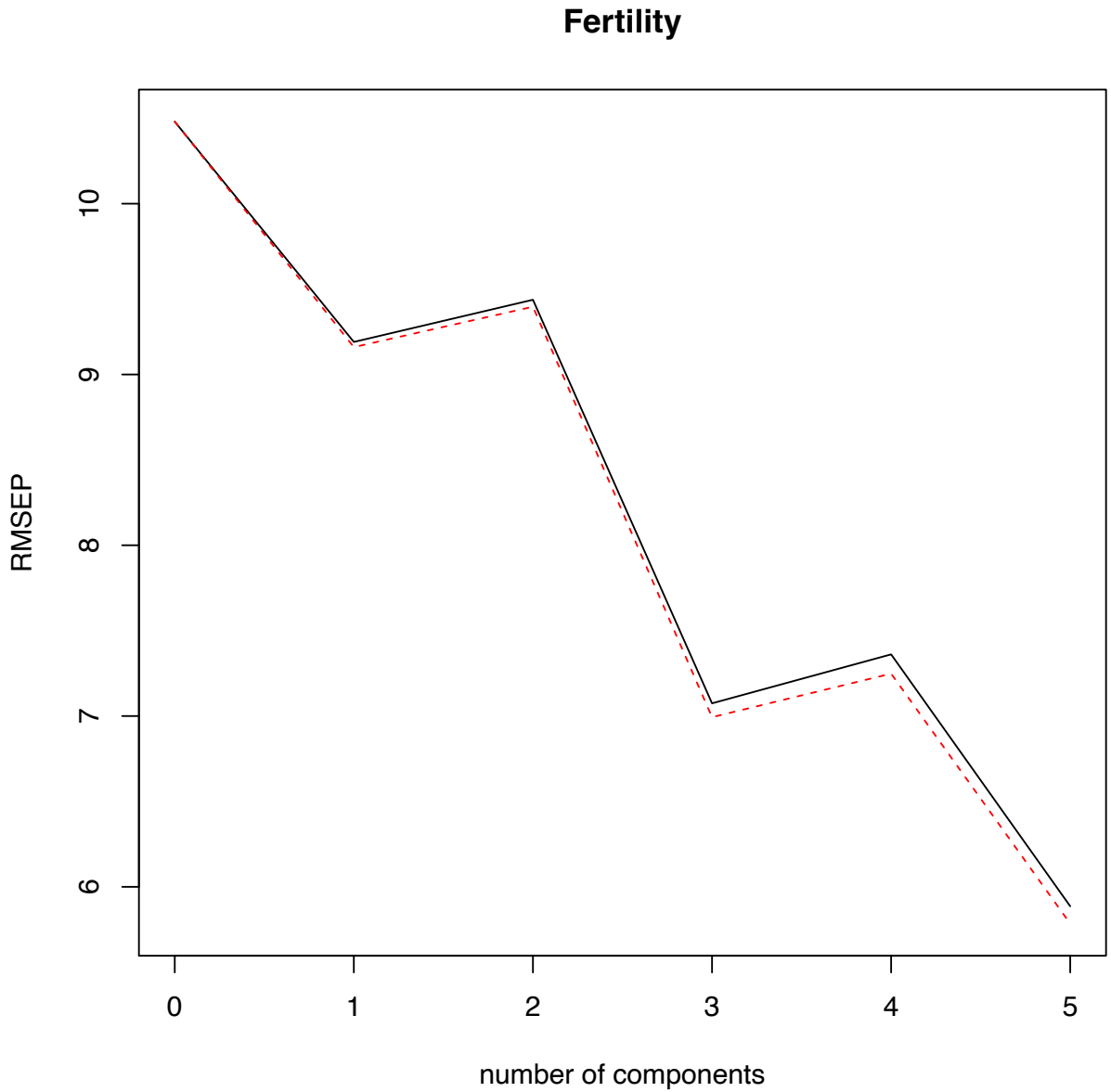
However, variance explained does not necessarily map to 'response explained' — just how big the variation is in different directions of X space.

`pls` is the partial least squares & principal components regression package.

```
library("pls")
swiss.train.pcr <- pcr(Fertility~., data=swiss[train,],
    validation="CV")
validationplot(swiss.train.pcr)
```

This plot says we should choose all the PCs (i.e. least squares). However, maybe 3 is quite good?

PCR validation plot



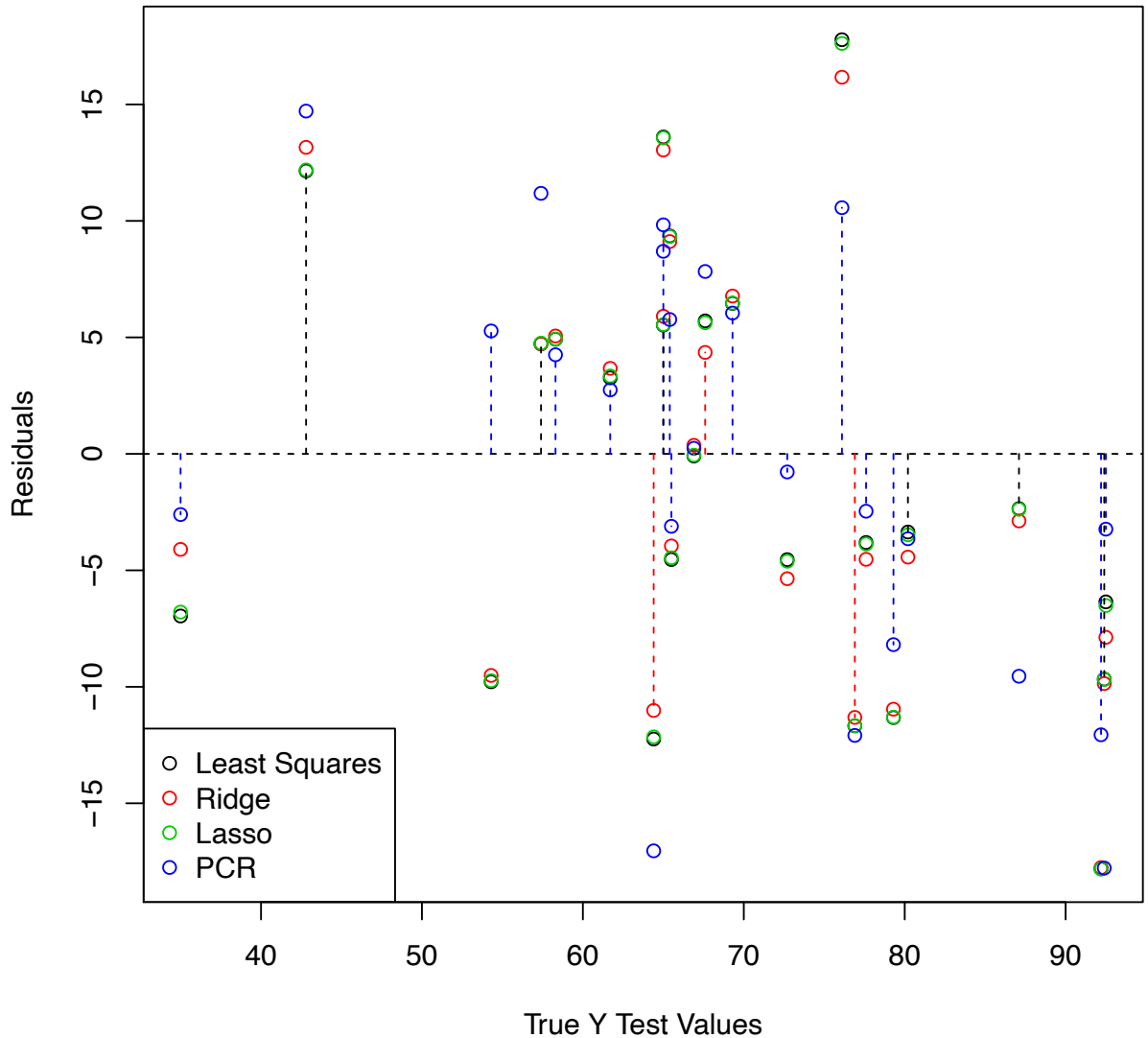
Prediction on the test set with PCR model

```
swiss.pcr.pred <- predict(swiss.train.pcr, swiss[test,],
    ncomp=3)
mse(y[test],swiss.pcr.pred)
[1] 1913
```

Better than lasso and least squares, not quite as good as ridge.

Residuals: Least squares, ridge, lasso, PCR

Blue pic?



Code behind residuals plot

```
lm.res <- swiss.predict.lm - y[test]
ri.res <- swiss.ridge.pred - y[test]
la.res <- swiss.lasso.pred - y[test]
pc.res <- swiss.pcr.pred - y[test]
ylim <- range(c(lm.res, ri.res, la.res, pc.res))
plot(y[test], lm.res, ylim=ylim, ylab="Residuals",
    xlab="True Y Test Values")
points(y[test], ri.res, col=2)
points(y[test], la.res, col=3)
points(y[test], pc.res, col=4)
abline(h=0, lty=2)
n <- length(pc.res)
ytest <- y[test]
for(i in 1:n)   {
        rvec <- c(lm.res[i], ri.res[i], la.res[i], pc.res[i])
        ix <- which(abs(rvec)==min(abs(rvec)))
        lines(c(ytest[i], ytest[i]), c(0, rvec[ix]),
            col=ix, lty=2)
        }
legend(x="bottomleft", col=1:4, legend=c("Least Squares", "Ridge",
    "Lasso", "PCR"), pch=1)
```

Warnings

After all of these model fits, we should scrupulously examine the residuals plots and other diagnostic plots.

We have not done this here.

Take all p-values with skepticism — only use them as informal information to help guide you to a better model and/or collect more data.

There is a temptation to use statistical models as a "black box". The temptation is much greater when you are analysing many data sets simultaneously — and you need to be quick.

Summary

We have subjected the `swiss` data set to a

- multiple linear regression fitted by least squares
- ridge regression, with manual and cross-validated choice of parameters
- lasso (with the same)
- principal components

Different methods will work differently on different data sets. There is not one single best method. However, ridge, lasso and PCR have strengths, especially when considering explanatory variables that are correlated.

Elements of Statistical Learning: Lecture 6. Distance and Dissimilarity Methods

Guy Nason[1]

Department of Mathematics
Imperial College

The basic problem

The methods in this block are examples of unsupervised learning.

In essence, we are conducting an exploration of the data, to see what kind of patterns and relationships exist in the data, which can't be tensioned or referenced against other variables, such as response variables or training sets.

In this block, we *do not* have access to a data matrix, X, which was observations on n individuals taken on p variables.

With distance (or dissimilarity) based methods, the assumption is that we have information on the distances (or dissimilarities) between individuals and we wish to recover what X might look like.

We call a proposed X a configuration: a set of n points in some space of dimension q, which might not be p.

French Road Distances Example

Obtained PDF of road distance table, scan and use optical character recognition to produce file

```
TOURS TOULOUSE TOULON STRASBOURG ST-ETIENNE ROUEN RENNES REIMS
PERPIGNAN PARIS NIMES NICE NANTES NANCY MONTPELLIER MARSEILLE
LYON LIMOGES LILLE LEMANS LEHAVRE GRENOBLE DIJON CLERMONTFD
CHERBOURG CALAIS BREST BORDEAUX BIARRITZ BESANCON ANGERS AMIENS
AMIENS 345 835 970 495 595 115 415 155 1045 130 840 1060 ...
ANGERS 105 560 915 735 530 285 125 435 760 295 735 1005 90 ...
BESANCON   470 740 590 230 265 530 680 315 665 405 460 640 ...
BIARRITZ   530 310 775 1100 695 805 620 910 445 765 595 865 ...
...
```

Then used a simple text editor to remove the name of the city at the first column on each row, apart from the first.

Wrangled Road Distances File

After wrangling

```
TOURS TOULOUSE TOULON STRASBOURG ST-ETIENNE ROUEN RENNES REIMS
PERPIGNAN PARIS NIMES NICE NANTES NANCY MONTPELLIER MARSEILLE
LYON LIMOGES LILLE LEMANS LEHAVRE GRENOBLE DIJON CLERMONTFD
CHERBOURG CALAIS BREST BORDEAUX BIARRITZ BESANCON ANGERS AMIENS
345 835 970 495 595 115 415 155 1045 130 840 1060 480 360 ...
105 560 915 735 530 285 125 435 760 295 735 1005 90 595 ...
470 740 590 230 265 530 680 315 665 405 460 640 660 205 ...
530 310 775 1100 695 805 620 910 445 765 595 865 515 990 ...
```

So, distance between Tours and Amiens is 345km, etc.

The file `frdist2.txt` can be read into R using

```
frdist2 <- read.table("frdist2.txt", header=TRUE)
```

Sicily Archaeological Data

Data from the archeological site *Capo Milazzese, Sicily*.

Objects found at 13 sites of possible huts.

374 objects found, classified into 20 different categories.

Are the huts related to each other? Time dimension?

Can we see construct a picture of the relationship?

Can we get anything from a visual inspection of the table?

Beh, E.J., Lombardo, R. and Alberti, G. (2018) Correspondance analysis and the Freeman-Tukey statistic: A study of archaeological data. *Computational Statistics and Data Analysis*, **128**, 73–86.

Sicily Archaeological Data

Code	*Object*	*Huts - Cxx*												
		1	2	3	4	5	6	8	9	10	11	16	18	20
R1	A-dinner (drink/eat)	0	1	2	1	1	0	1	1	4	6	3	2	2
R2	A-dinner (pour)	1	0	0	0	0	0	0	1	1	0	0	0	1
R3	Ae-cook	0	0	0	0	0	0	0	0	1	0	0	0	0
R4	Ae-dinner (drink/eat)	0	0	0	0	0	0	0	0	1	2	1	1	0
R5	Ae-storage	0	1	1	0	0	0	0	1	1	1	1	1	0
R6	A-storage	0	1	2	0	1	0	0	0	1	0	0	0	0
R7	Cooking	2	6	3	0	1	0	3	5	2	3	2	1	5
R8	Dinner (drink/eat-cware)	1	3	0	0	0	0	0	0	0	0	0	0	1
R9	Dinner (eat-cware)	0	0	0	0	0	0	0	2	1	0	0	1	2
R10	Dinner (drink/eat-fware)	1	3	1	1	1	2	2	5	4	2	2	1	3
R11	Dinner (pour-cware)	0	0	0	0	0	0	1	0	1	0	0	0	1
R12	Dinner (pour-fware)	1	3	4	3	2	2	2	4	4	7	2	5	5
R13	Dinner (supporting)	0	1	4	2	2	1	1	0	0	2	3	2	1
R14	Dinner (processing)	1	0	0	2	1	0	0	2	0	0	0	0	0
R15	Processing	0	6	3	2	0	0	2	15	4	2	1	0	8
R16	Spinning	0	0	0	0	0	0	0	2	5	0	0	0	0
R17	Storage (long-term)	0	2	2	1	0	1	2	2	2	1	4	0	1
R18	Storage (short-term)	2	2	1	2	1	1	3	4	2	1	1	2	3
R19	Storage (other)	5	5	2	2	1	1	2	3	4	0	1	1	1
R20	Working	0	6	8	0	2	6	3	10	12	0	5	5	6

See some clustering?

Distances to configurations

Of course, we can just get a map of France to see how towns are geographically related to each other.

However, suppose we did not have a map and we wanted to construct one from the distance table.

Also, the distance table we have is *road distances*. This is different to the distance of "how the crow flies" or the direct distance.

E.g. a city might only be reachable across a slow mountain road, so its effective road distance might me substantially greater than the "direct" distance.

Also, we could form distances between huts for the Sicily data. That is easy to do.

Distances to configurations

We could just use the Euclidean distance:
$e^2_{m,\ell} = \sum_{v=1}^{p}(X_{m,v} - X_{\ell,v})^2$, even on the Sicily data.

However, does the object absence/presence really lead to a configuration that makes sense in a Euclidean space?

E.g. if one hut has 3 objects and another hut has 6 is that the same contribution to distance as if it were 10 and 13??

We just don't know.

Thinking of an appropriate distance measure is an important step of the process of moving from distances, or more properly, *dissimilarities* to configurations.

Configurations to Distances

Before we study methods that obtain configurations from distances, let us remind ourselves of how we can calculate a Euclidean distance from a configuration, X.

Or rather, how to get distances from coordinates.

We already wrote

$$e_{m,\ell} = \sum_{v=1}^{p} (X_{m,v} - X_{\ell,v})^2, \tag{1}$$

the squared Euclidean distance. Let E be the $n \times n$ matrix of Euclidean distances $(e_{m,\ell})$.

Configurations to distances 2

Let's rearrange this

$$
\begin{aligned}
e_{m,\ell} &= \sum_{v=1}^{p}(X_{m,v} - X_{\ell,v})^2 && (2)\\
&= \sum_{v=1}^{p} X_{m,v}X_{m,v} + \sum_{v=1}^{p} X_{\ell,v}X_{\ell,v} - 2\sum_{v=1}^{p} X_{m,v}X_{\ell,v} && (3)\\
&= X_{(m)}^T X_{(m)} + X_{(\ell)}^T X_{(\ell)} - 2X_{(m)}^T X_{(\ell)}, && (4)
\end{aligned}
$$

where $X_{(m)}^T$ is the mth row of X, a p-dimensional vector, $m, \ell = 1, \ldots, n$.

Hence, distances are expressible in terms of inner products.

Inner Product Representation of Distances

First form the *inner product matrix*

$$B = XX^T, \tag{5}$$

which is an $n \times n$ matrix, where $b_{m,\ell} = X_{(m)}^T X_{(\ell)}$.

So, we can write the Euclidean distance as

$$e_{m,\ell} = b_{m,m} + b_{\ell,\ell} - 2b_{m,\ell}, \tag{6}$$

$m, \ell = 1, \ldots, n$.

Loss of Information (Rotation)

Let P be an orthogonal $p \times p$ matrix - geometrically, this corresponds to a rotation in the p-dimensional space.

Then the inner product matrix formed from $Y = XP$ is

$$B_Y = YY^T = XP(XP)^T = XPP^TX^T = XI_pX^T = XX^T = B. \tag{7}$$

So, X and a rotated version, Y, have exactly the same inner product matrix.

Hence, on going from X to B we *lose* orientation information.

And, in fact, we often operate on the CENTRED X matrix.

Loss of Information (Position)

Suppose $W_{(m)} = X_{(m)} - \mu$, where μ is an arbitrary p-vector.

Then

$$W_{(m)}^T W_{(\ell)} = X_{(m)}^T X_{(\ell)} - X_{(m)}^T \mu - \mu^T X_{(\ell)} + \mu^T \mu. \tag{8}$$

Now form distances using W

$$\begin{aligned} e_{m,\ell}^{(W)} &= W_{(m)}^T W_{(m)} + W_{(\ell)}^T W_{(\ell)} - 2W_{(m)}^T W_{(\ell)} && (9)\\ &= X_{(m)}^T X_{(m)} - X_{(m)}^T \mu - \mu^T X_{(m)} + \mu^T \mu && (10)\\ &\quad + X_{(\ell)}^T X_{(\ell)} - X_{(\ell)}^T \mu - \mu^T X_{(\ell)} + \mu^T \mu && (11)\\ &\quad - 2(X_{(m)}^T X_{(\ell)} - X_{(m)}^T \mu - \mu^T X_{(\ell)} + \mu^T \mu) && (12)\\ &= e_{m,\ell}. && (13) \end{aligned}$$

Hence, on going from B to E we *lose* position information.

Recovery of X from B from E: Choosing origin

On going from E to B we have to choose an origin (position): might as well put the centroid of the data at the origin.

The p-dimensional mean vector, $\bar{X}$, associated with data matrix X

$$\bar{X} = n^{-1}\mathbf{1}_n^T X, \tag{14}$$

where $\mathbf{1}_n$ is the n-vector consisting of ones.

Hence, we want the condition $\bar{X} = 0$ or $\mathbf{1}_n^T X = 0$.

If $\mathbf{1}_n^T X = 0$, then $0 = \mathbf{1}_n^T XX^T = \mathbf{1}_n^T B = 0$, and
If $\mathbf{1}_n^T B = 0$, then $0 = \mathbf{1}_n^T XX^T \implies 0 = \mathbf{1}_n^T XX^T\mathbf{1}_n \implies$
$\mathbf{1}_n^T X(\mathbf{1}_n^T X) = 0 \implies \mathbf{1}_n^T X = 0$.

Hence $\mathbf{1}_n$ is an eigenvector of B^T with eigenvalue of 0.

Recovering B from E

Recall from (6)

$$e_{m,\ell} = b_{m,m} + b_{\ell,\ell} - 2b_{m,\ell}. \tag{15}$$

Summing over m gives us ($e_{\bullet,\ell} = \sum_{m=1}^{n} e_{m,\ell}$)

$$e_{\bullet,\ell} = b_{\bullet,\bullet} + nb_{\ell,\ell} - 2b_{\bullet,\ell}. \tag{16}$$

Since $\mathbf{1}$ is an eigenvector of B^T, it is also of B (due to construction of $B = XX^T$), hence row and column sums of B are zero. Hence, $b_{\bullet,\ell} = b_{m,\bullet} = 0$.

Hence,

$$e_{\bullet,\ell} = b_{\bullet,\bullet} + nb_{\ell,\ell}. \tag{17}$$

Recovering B from E — 2

Similarly,

$$e_{m,\bullet} = b_{\bullet,\bullet} + nb_{m,m}. \tag{18}$$

Summing over m and n gives

$$e_{\bullet,\bullet} = nb_{\bullet,\bullet} + nb_{\bullet,\bullet} = 2nb_{\bullet,\bullet}. \tag{19}$$

Now rearrange (6) to give

$$b_{m,\ell} = -\tfrac{1}{2}(e_{m,\ell} - b_{m,m} - b_{\ell,\ell}), \tag{20}$$

and using (17) and (18) we have

$$\begin{aligned} b_{m,\ell} &= -\tfrac{1}{2}\left\{e_{m,\ell} - (e_{m,\bullet} - b_{\bullet,\bullet})/n - (e_{\bullet,\ell} - b_{\bullet,\bullet})/n\right\} && (21)\\ &= -\tfrac{1}{2}(e_{m,\ell} - \tfrac{e_{m,\bullet}}{n} - \tfrac{e_{\bullet,\ell}}{n} + 2\tfrac{b_{\bullet,\bullet}}{n}) && (22)\\ &= -\tfrac{1}{2}(e_{m,\ell} - \tfrac{e_{m,\bullet}}{n} - \tfrac{e_{\bullet,\ell}}{n} + \tfrac{e_{\bullet,\bullet}}{n^2}). && (23) \end{aligned}$$

Recovering B from E — 3

$$
\begin{aligned}
b_{m,\ell} &= -\tfrac{1}{2}(e_{m,\ell} - \tfrac{e_{m,\bullet}}{n} - \tfrac{e_{\bullet,\ell}}{n} + \tfrac{e_{\bullet,\bullet}}{n^2}) && (24)\\
&= -\tfrac{1}{2}(\text{entry} - \text{row av.} - \text{col av.} + \text{grand av.}), && (25)
\end{aligned}
$$

or in matrix terms

$$
B = -\frac{1}{2}(I_n - \mathbf{1}\mathbf{1}^T/n)\, E\, (I_n - \mathbf{1}\mathbf{1}^T/n), \qquad (26)
$$

it's an exercise to prove this and that $\mathbf{1}_n$ is an eigenvector of B with eigenvalue 0.

Recovering X (or something like it) from B

Recall B is an $n \times n$ matrix.

Also $B = XX^T$ so $a^T Ba = (a^T X)(X^T a) = (a^T X)(a^T X)^T \geq 0$, so B is positive semi-definite and symmetric so we can form the following eigendecomposition:

$$B = \sum_{i=1}^{n} \lambda_i e^{(i)} e^{(i)T}, \tag{27}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n'} > 0$ and $\lambda_{n'+1}, \ldots, \lambda_n = 0$ and $\{e^{(i)}\}$ are the eigenvectors of B.

We know at least one of the eigenvalues is zero, with eigenvector $\mathbf{1}$.

This eigendecomposition is different to the one for $X^T X$ that we had in the previous block.

Recovering X (or something like it) from B — 2

We define a new set of vectors

$$f^{(i)} = \sqrt{\lambda_i} e^{(i)}, \tag{28}$$

for $i = 1, \ldots, n'$. These are all n-vectors. And write Y as the stacked matrix of these.

$$Y_{n \times n'} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ f^{(1)} & f^{(2)} & \cdots & f^{(n')} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \tag{29}$$

Recovering X (or something like it) from B — 3

Then

$$
\begin{aligned}
YY^T &= \sum_{i=1}^{n'} f^{(i)} f^{(i)T} && (30)\\
&= \sum_{i=1}^{n'} (\sqrt{\lambda_i} e^{(i)})(\sqrt{\lambda_i} e^{(i)T}) && (31)\\
&= \sum_{i=1}^{n'} \lambda_i e^{(i)} e^{(i)T} && (32)\\
&= \sum_{i=1}^{n} \lambda_i e^{(i)} e^{(i)T} && (33)\\
&= XX^T = B, && (34)
\end{aligned}
$$

since $\lambda_{n'+1}, \ldots, \lambda_n = 0$.

Recovering X (or something like it) from B — 4

Note that $n' = \text{rank}(B) \leq p$.
Also, note that

$$(Y^T Y)_{n' \times n'} = (f^{(j)^T} f^{(i)}) = \sqrt{\lambda_j}\sqrt{\lambda_i} e^{(j)^T} e^{(i)} \quad (35)$$

$$= \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n'}), \quad (36)$$

as the eigenvectors $e^{(i)}$ and $e^{(j)}$ are orthonormal.

So, Y is already in principal axis form (i.e. the principal components of Y are Y).

Worked Numerical Example

Suppose we have four observations in three dimensions.

$$X = \begin{pmatrix} 3 & 1 & 4 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \\ 4 & 3 & 2 \end{pmatrix}. \tag{37}$$

The inner product matrix, B is:

$$XX^T = \begin{pmatrix} 3 & 1 & 4 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \\ 4 & 3 & 2 \end{pmatrix} \begin{pmatrix} 3 & 1 & 2 & 4 \\ 1 & 2 & 3 & 3 \\ 4 & 1 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 26 & 9 & 17 & 23 \\ 9 & 6 & 10 & 12 \\ 17 & 10 & 17 & 21 \\ 23 & 12 & 21 & 29 \end{pmatrix}. \tag{38}$$

Clearly symmetric.

Getting E from B

Use (6): $e_{m,\ell} = b_{m,m} + b_{\ell,\ell} - 2b_{m,\ell}$.

Diagonal: $e_{m,m} = b_{m,m} + b_{m,m} - 2b_{m,m} = 0$.

E.g. $e_{1,2} = b_{1,1} + b_{2,2} - 2b_{1,2} = 26 + 6 - 18 = 14$.

So

$$E = (e_{m,\ell}) = \begin{pmatrix} 0 & 14 & 9 & 9 \\ 14 & 0 & 3 & 11 \\ 9 & 3 & 0 & 4 \\ 9 & 11 & 4 & 0 \end{pmatrix}. \tag{39}$$

Now imagine, we start with E and want to recover configuration.

Getting B' from E

Use (26)

$$B' = -\frac{1}{2}(I_n - \mathbf{1}\mathbf{1}^T/n)\, E\, (I_n - \mathbf{1}\mathbf{1}^T/n), \tag{40}$$

Note, B and B' might be different as B' assumes origin of data is at 0, which might not be the case for X. So

$$\begin{aligned} B' &= -\frac{1}{2}\left\{\frac{1}{4}\begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}\right\} E \left\{\frac{1}{4}\begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}\right\} \\ &= \tfrac{1}{8}\begin{pmatrix} 39 & -21 & -13 & -5 \\ -21 & 31 & 7 & -17 \\ -13 & 7 & 7 & -1 \\ -5 & -17 & -1 & 23 \end{pmatrix}. \end{aligned} \tag{41}$$

This matrix B' is different to B

Computing B' from centred X

The mean vector of X is $\bar{X} = (2.5, 2.25, 2.25)^T$.
If we subtract the mean off every observation in X we get the centred X_C matrix:

$$X_C = \frac{1}{4}\begin{pmatrix} 2 & -5 & 7 \\ -6 & -1 & -5 \\ -2 & 3 & -1 \\ 6 & 3 & -1 \end{pmatrix}, \tag{42}$$

and

$$X_C X_C^T = \frac{1}{8}\begin{pmatrix} 39 & -21 & -13 & -5 \\ -21 & 31 & 7 & -17 \\ -13 & 7 & 7 & -1 \\ -5 & -17 & -1 & 23 \end{pmatrix} = B'. \tag{43}$$

Obtaining Y from B'

Using R the eigendecomposition of B' is:

```
> eigen(Bprime)
eigen() decomposition
$values
[1]  7.711920e+00  4.306294e+00  4.817855e-01 -2.253617e-16

$vectors
           [,1]       [,2]        [,3] [,4]
[1,]  0.6952353  0.5161676  0.01479546 -0.5
[2,] -0.6432941  0.3693135  0.44696789 -0.5
[3,] -0.2512168 -0.1224748 -0.81968899 -0.5
[4,]  0.1992756 -0.7630062  0.35792563 -0.5
```

Let $\lambda_1 = 7.71, \lambda_2 = 4.31, \lambda_3 = 0.482, \lambda_4 = 0$ (neglecting small rounding error).

The last eigenvector is (multiple of) $\mathbf{1}$ with eigenvalue of $\lambda_4 = 0$.

Building f vectors

Clearly, $n' = 3$.
So,

$$f^{(1)^T} = \sqrt{7.71}(0.695, -0.643, -0.251, 0.199) \quad (44)$$
$$= (1.93, -1.78, -0.698, 0.553). \quad (45)$$

Similarly

$$f^{(2)^T} = (1.07, 0.766, -0.254, -1.58), \quad (46)$$
$$f^{(3)^T} = (0.01, 0.310, -0.569, 0.248). \quad (47)$$

Recovered Data Matrix: Y

$$Y = \begin{pmatrix} 1.93 & 1.07 & 0.01 \\ -1.78 & 0.766 & 0.310 \\ -0.698 & -0.254 & -0.569 \\ 0.553 & -1.58 & 0.248 \end{pmatrix} \tag{48}$$

You can calculate that $YY^T = B'$ indeed.

And you get E from B'.

So, Y is a configuration that is consistent with E (but it is not unique, clearly any translated or rotated configuration will also do).

Elements of Statistical Learning: Lecture 7. Classical Multidimensional Scaling

Guy Nason[1]

Department of Mathematics
Imperial College

Classical Multidimensional Scaling

The method of obtaining from Y from E is variously called *classical multidimensional scaling*, or *metric multidimensional scaling* or *principal coordinates analysis*.

The earliest work was due to Torgerson (1958) and notable later contributions have been from Kruskal, Gower and others.

It can be carried out in R using the `cmdscale` function.

Assume that `frdist` contains the French road distances data: a symmetric matrix with zeroes on the diagonal.

French road distances example

```
# Convert distance matrix into distance object
frdistobj <- as.dist(frdist)

# Apply scaling. Initially choose n' = 10
frscalesoln <- cmdscale(frdistobj, k=10, eig=TRUE)

# Here are the components of the compound object
names(frscalesoln)
[1] "points" "eig"    "x"      "ac"     "GOF"
```

`points` is a 32×10 proposed configuration solution Y. Here we chose $n' = 10$, but really we need to look properly at the eigenvalues in `eig`

The Eigenvalues

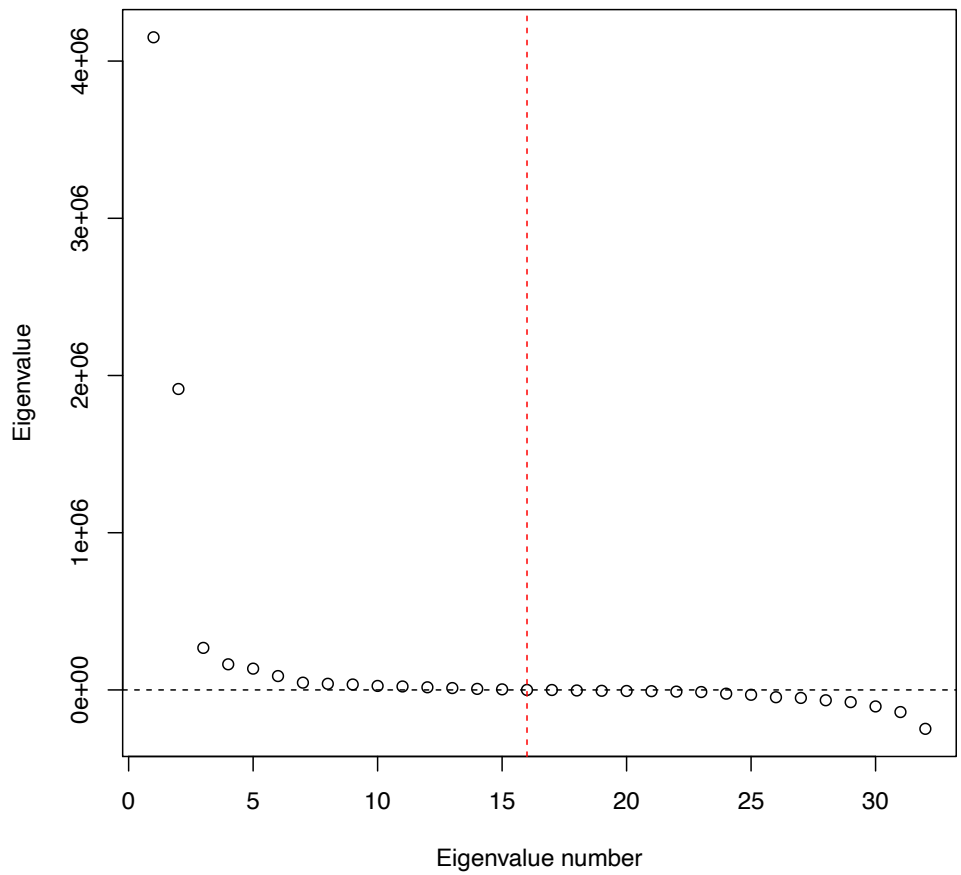
The first and last eigenvalues from `frscalesoln$eig` $\times 100000$:

$$41.5,\ 19.1,\ 2.7,\ 1.6,\ 1.4,\ 0.9\ldots$$
$$\ldots \textit{ (20 values less than 0.5 magnitude)} \ldots$$
$$\ldots -0.5,\ -0.7,\ -0.8,\ -1.05,\ -1.41,\ -2.47$$

Really only TWO big ones.

French road distances example: Eigenvalues

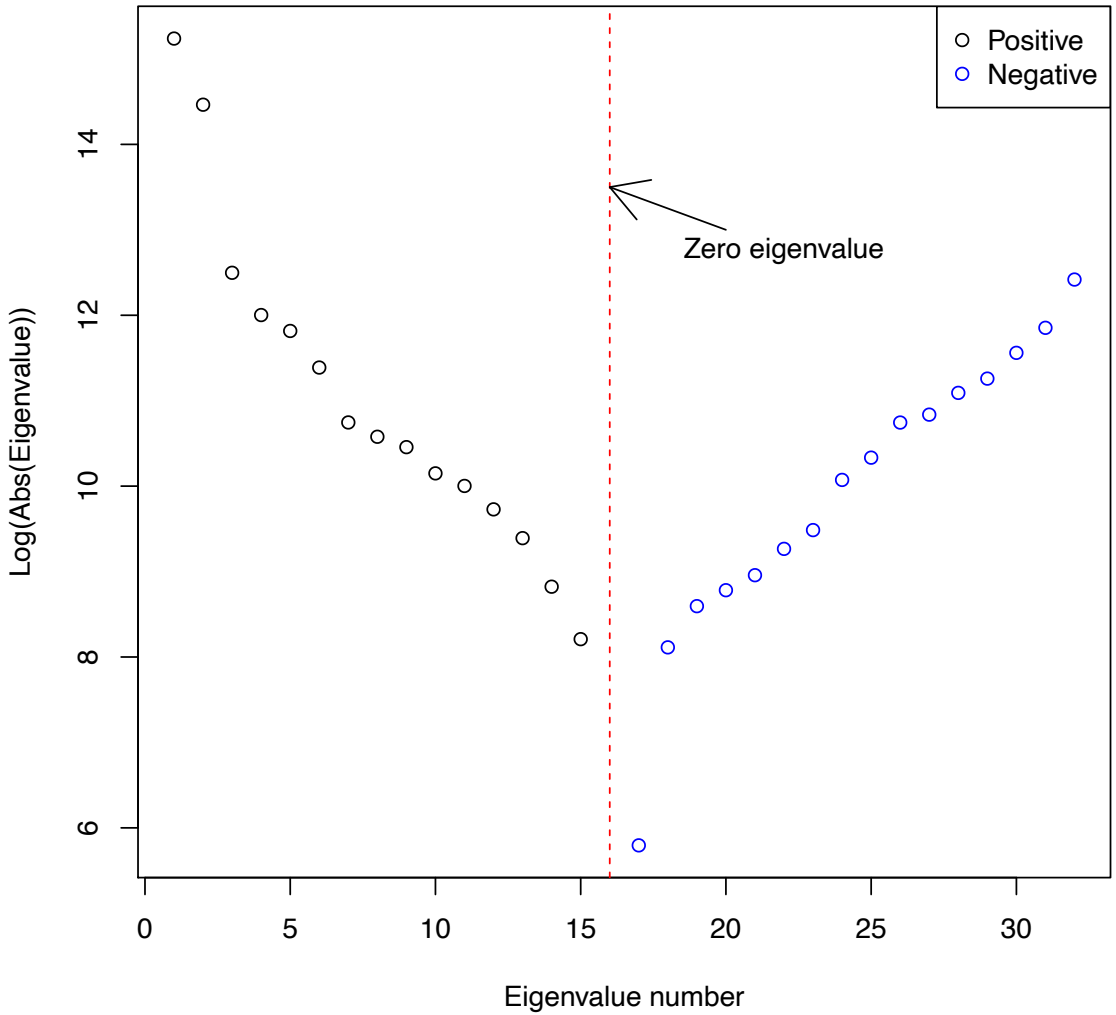
```
plot(1:32, frscalesoln$eig, xlab="Eigenvalue number",
    ylab="Eigenvalue")
abline(h=0, lty=2)
abline(v=16, lty=2, col=2)
```



French road distances example: Log Abs Eigen

```
the.eigs <- frscalesoln$eig
the.eigs[16] <- NA

plot(1:32, log(abs(the.eigs)), xlab="Eigenvalue number",
    ylab="Log(Abs(Eigenvalue))", type="n")
points(1:15, log(abs(the.eigs[1:15])), col=1)
points(17:32, log(abs(the.eigs[17:32])), col=4)
abline(h=0, lty=2)
abline(v=16, lty=2, col=2)
arrows(x0=20, y0=13, x1=16, y1=13.5)
text(x=22, y=12.75, label="Zero eigenvalue")

legend(x="topright", col=c(1,4), pch=1,
    legend=c("Positive", "Negative"))
```

Log Abs Eigen



Apart from the first two, there are as many negative eigenvalues that are roughly of the same magnitude as the positive ones.

Conclusions from Eigenvalue Plots

Two eigenvalues are significantly bigger than the rest.

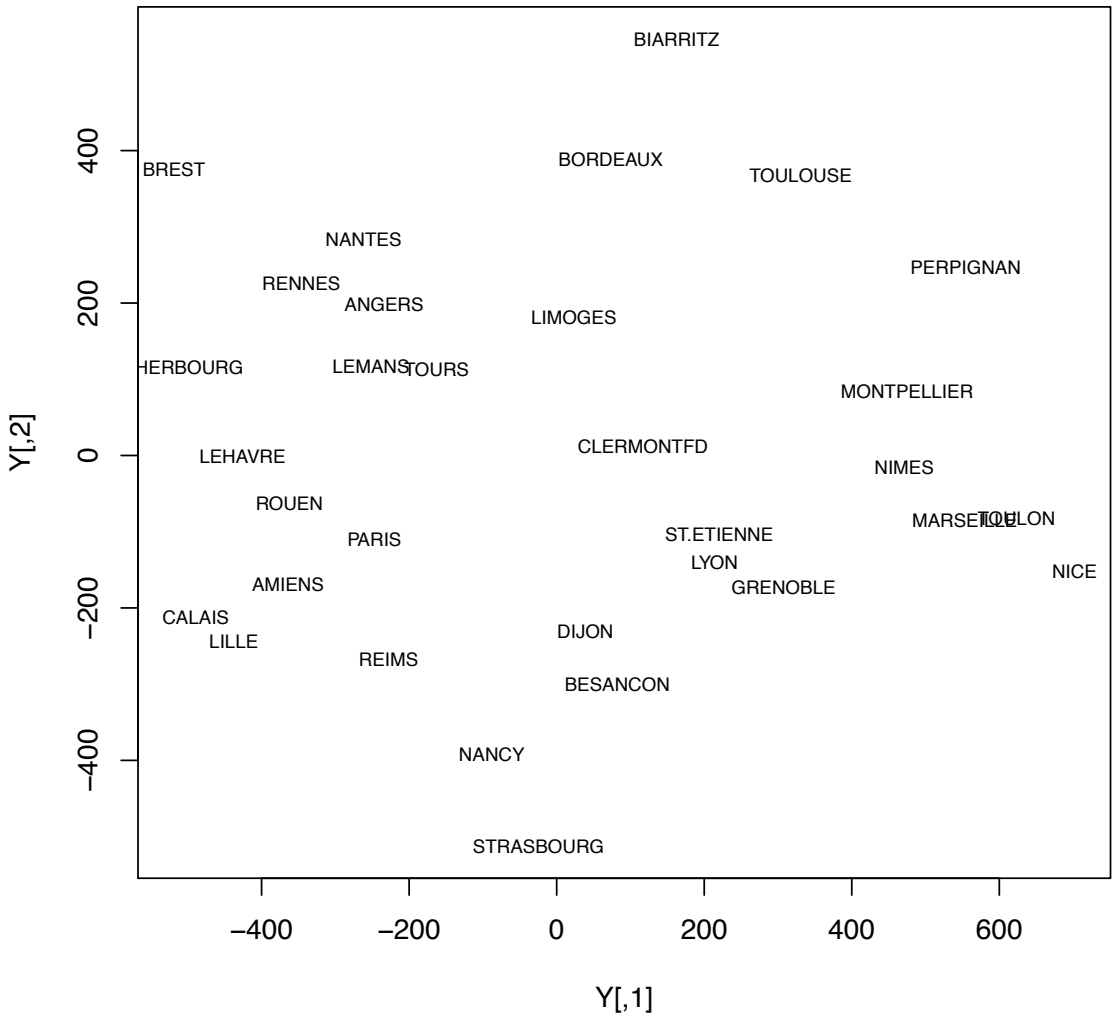
There are a lot of negative eigenvalues — these correspond to the non-Euclidean nature of the road distances.

Why non-Euclidean? Roads are strange — they do not always go from A to B in the most direct way. There are often hills and valleys or other natural features to avoid.

But, it looks like two-dimensions are roughly right to project this data into. So, plot observations according to the first two columns of Y, which are stored in the `points` component of `frscalesoln`.

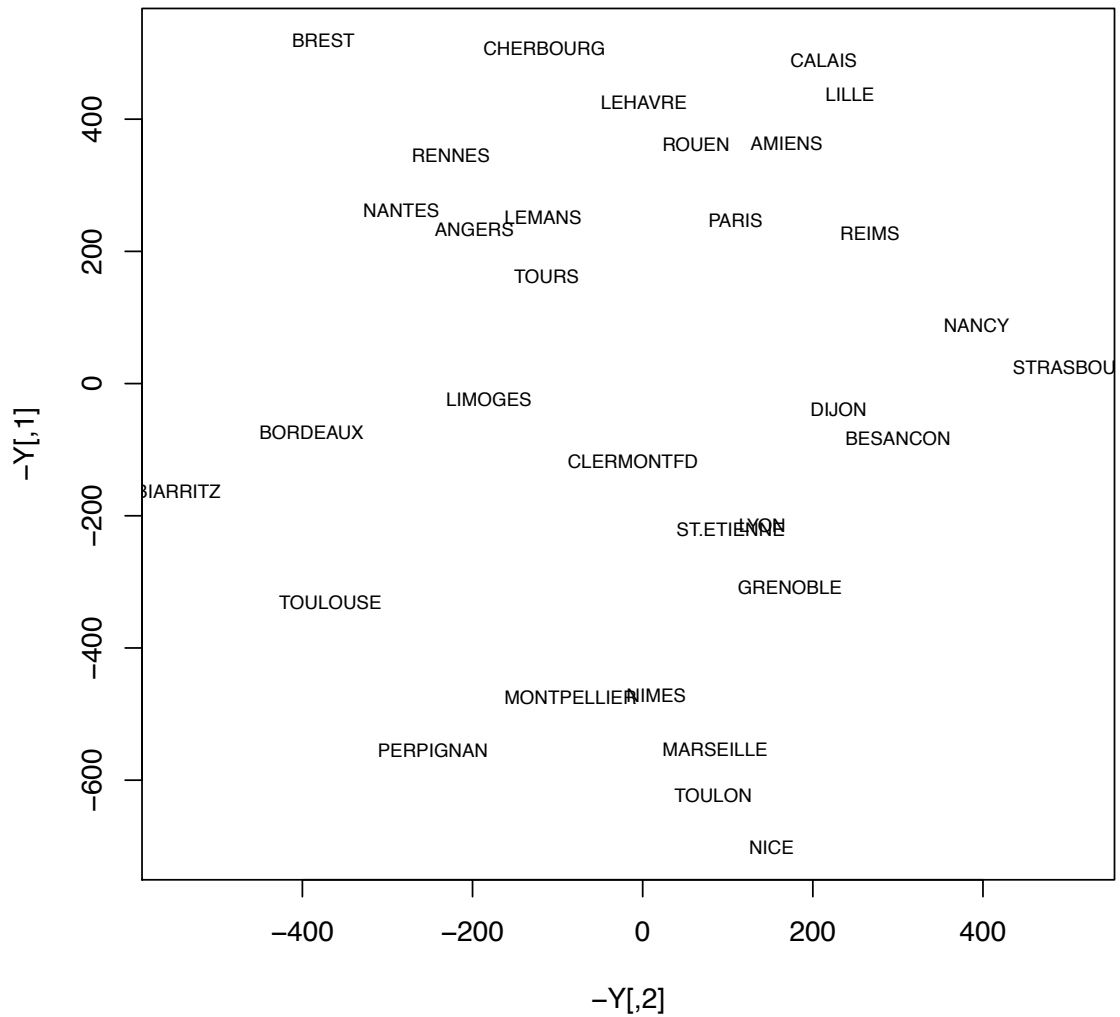
Also examine 3rd dimension. Which towns have high values on this dimension?

Plot of towns against first two scaling coordinates



Doesn't look like map of France? Orientation? Where is origin?

Better oriented plot



Map of France

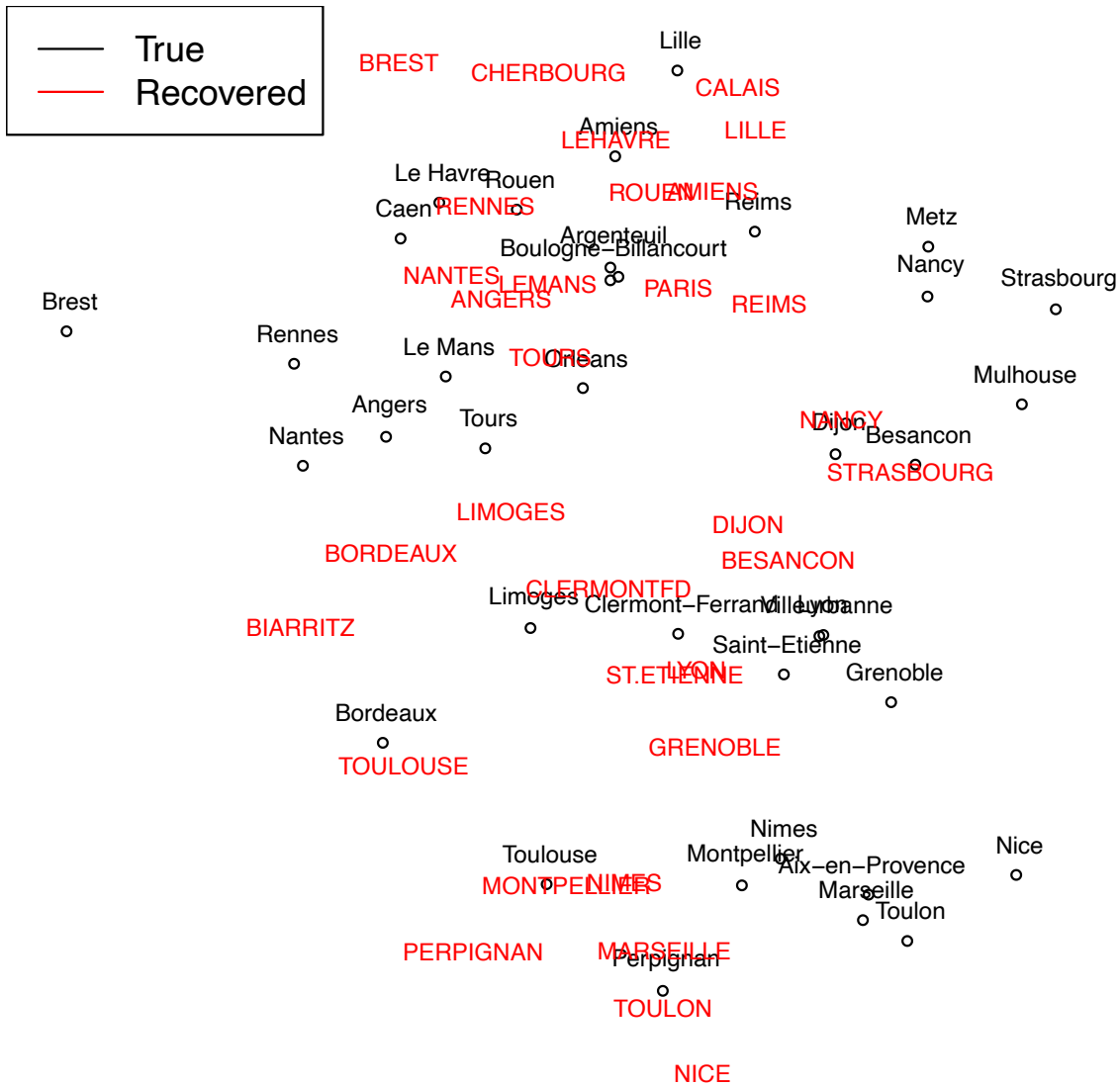
Superimposed true cities and recovered

Plot map of true locations of cities, then superimpose our configuration.

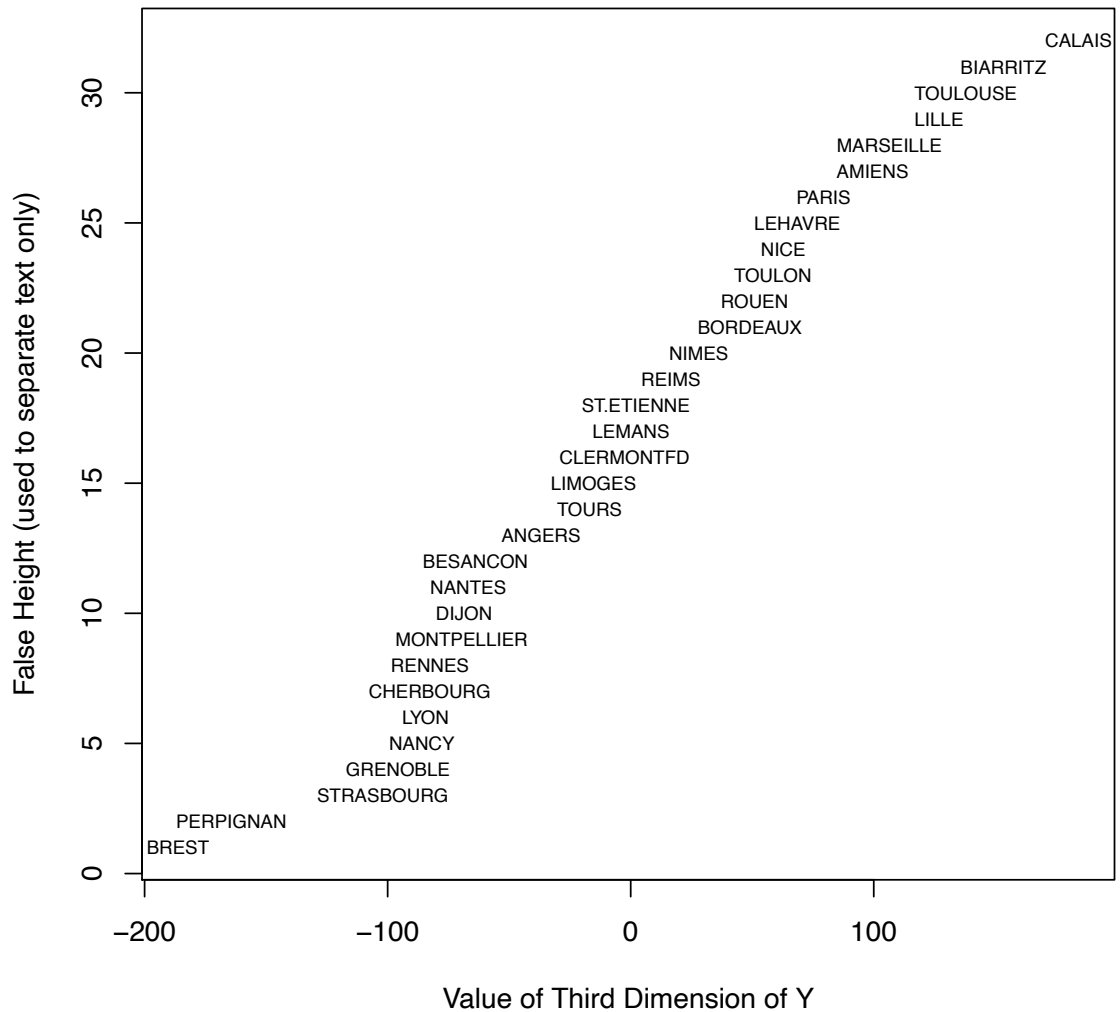
```
library("maps")
map('france', type="n")
map.cities(x=world.cities, country="France", minpop=100000,
    label=TRUE)
text(-frscalesoln$points[,2]/140+2.3,
    -frscalesoln$points[,1]/140+47,
    labels=dimnames(frdist)[[1]], cex=0.7, col=2)
legend(x="topleft", legend=c("True", "Recovered"),
    col=1:2, lty=c(1,1))
```

Note, the addition of 2.3 and 47 are roughly the longitude and latitude of France and that, with the scaling of /140 changes the position and scale of the cities onto the France map. However, this is not perfect, and we'll see a better way of doing this later.

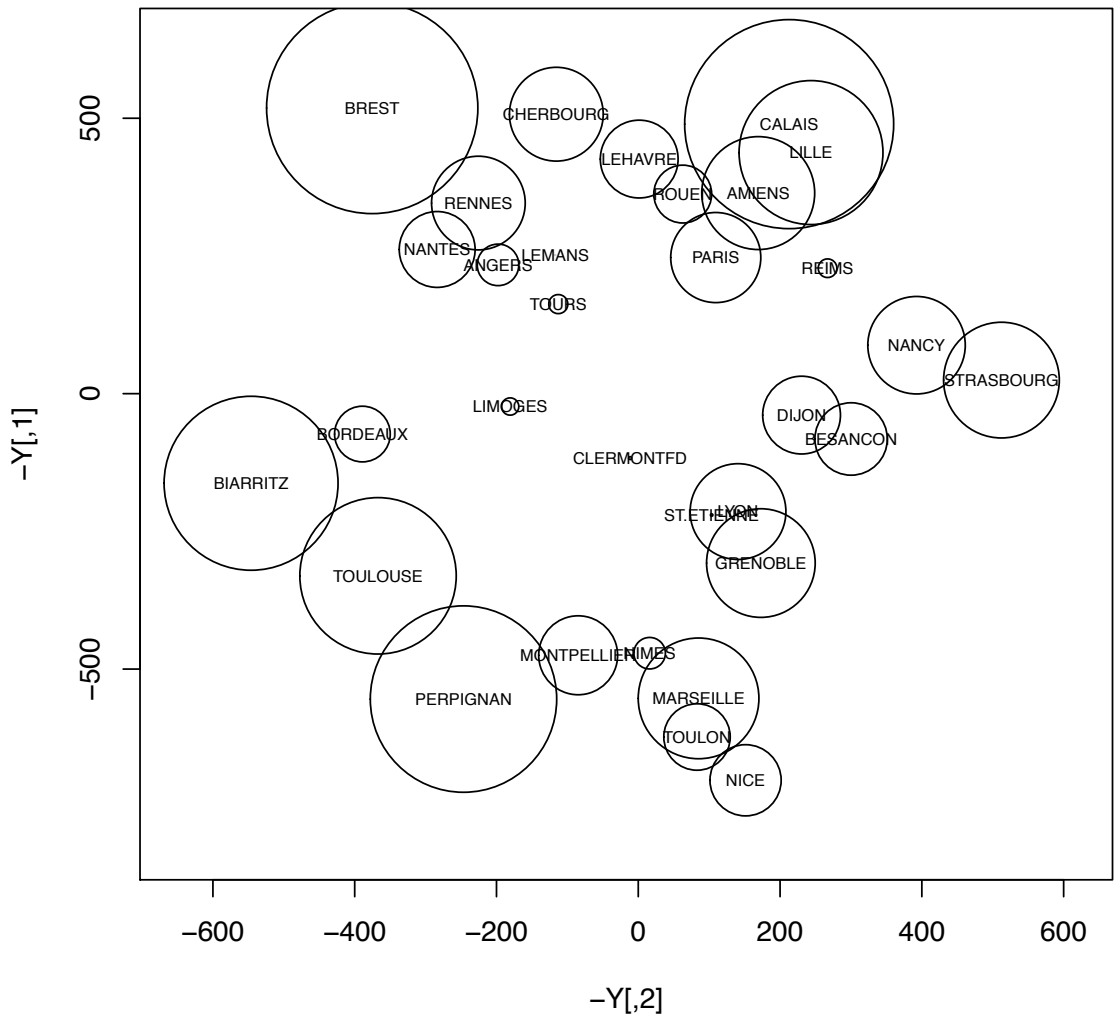
Superimposed true cities and recovered



Values of third dimension for each town



Third dimension against location



Symmetry

To form the spectral decomposition of B it needs to be symmetric.

If E is symmetric, then so is B.

However, sometimes E is not symmetric. For example, in the French road distances example there were cases of the distance from A to B, was not the same as B to A!

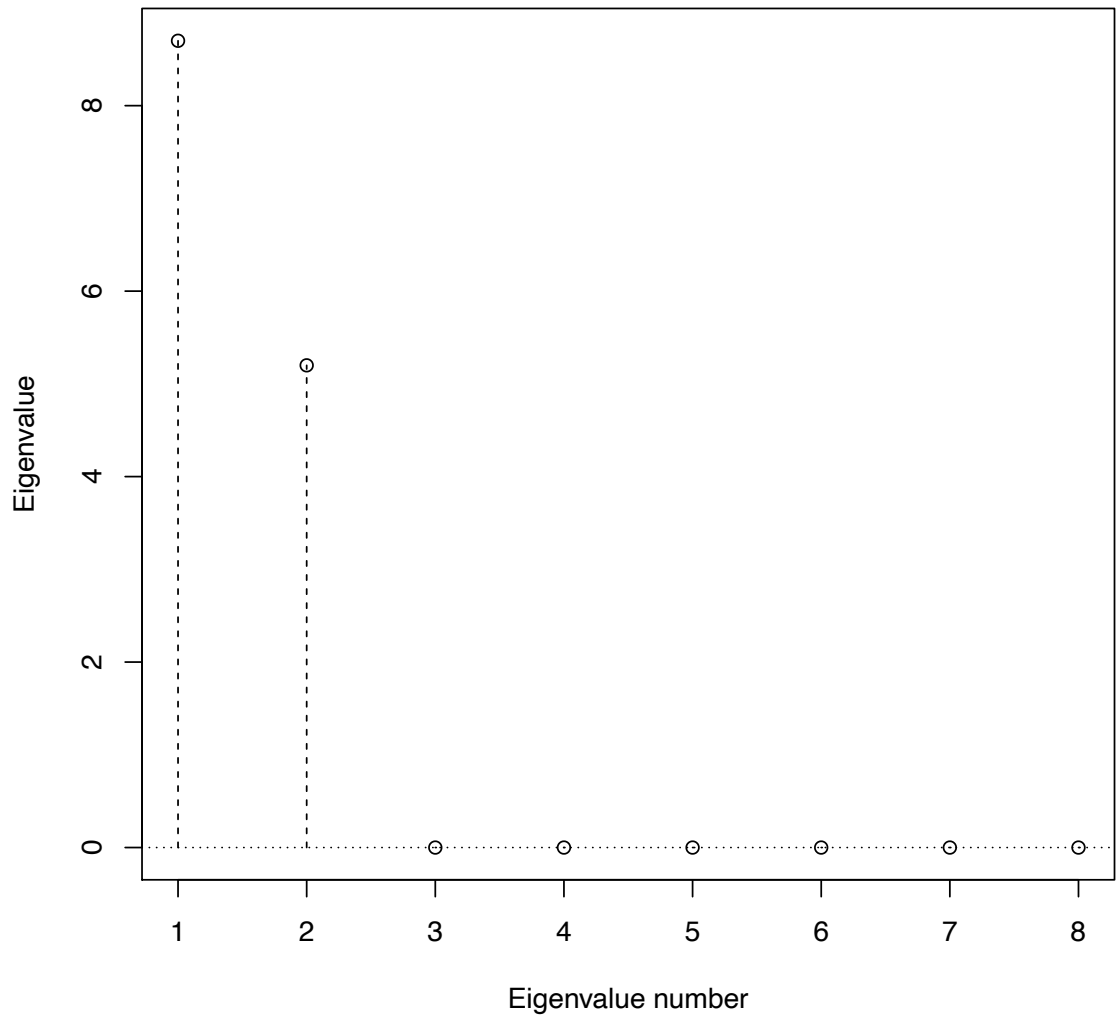
This might have been due to

- an error, e.g. printed, or in the data collection.
- a genuine difference.

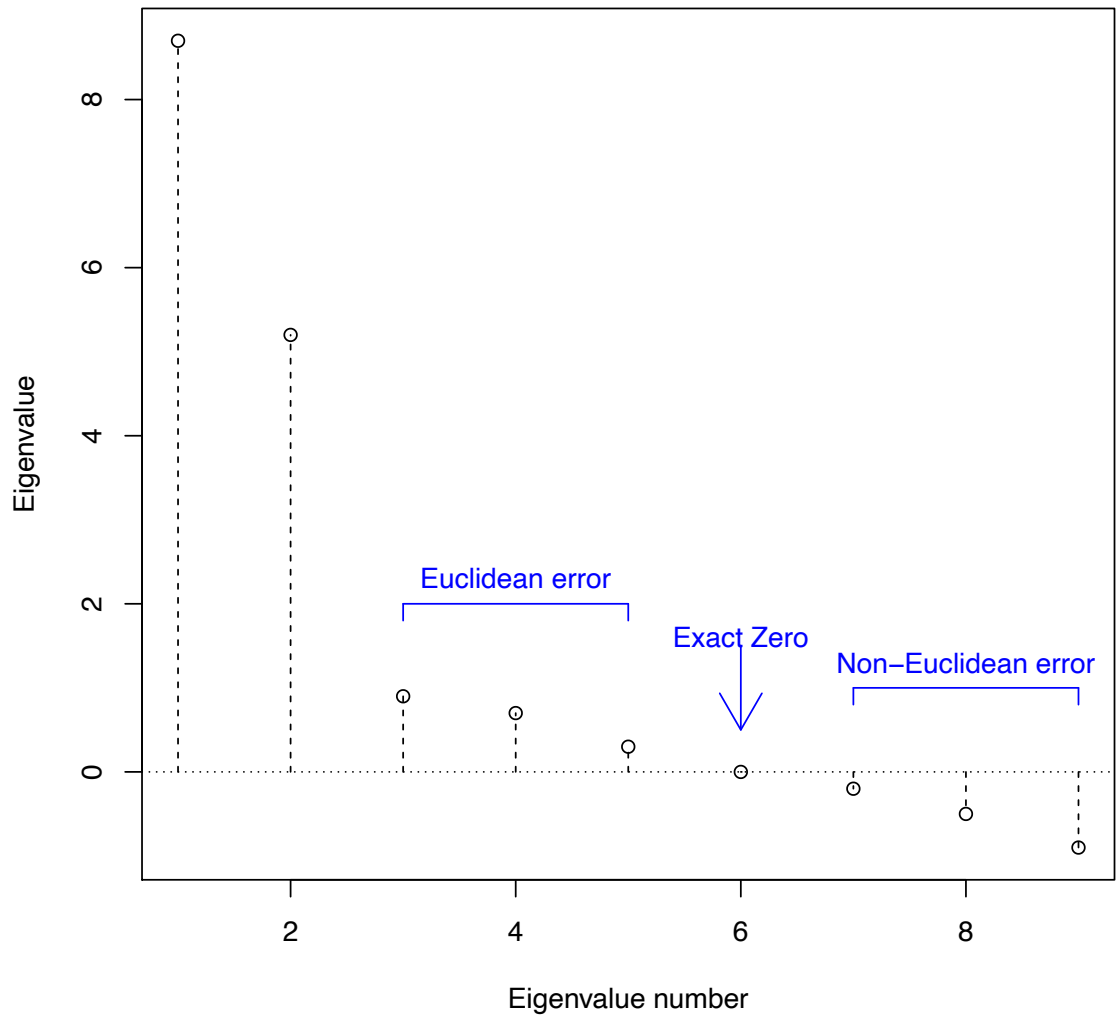
Non-symmetric distances (dissimilarities) often occur. E.g. a one-way system in a city; temporary speed restrictions in one direction of a two-way railway line, etc.

To use classical scaling, E must be symmetrised.

Classical Scaling in Practice: Ideal



Classical Scaling in Practice: Usual



What does classical scaling give us?

1. Method to recover Y configuration from E.
2. A test for Euclidean-ness.
3. A method for estimating dimensionality.

Summary of Classical Scaling Procedure

1. Form/obtain E, matrix of squared distances.

2. Form B from E (ensure symmetric).

3. Write $B = \sum_i \lambda^{(i)} e^{(i)} e^{(i)T}$, where

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n'} > 0 = \lambda_{n'+1} \geq \lambda_{n'+2} \geq \cdots \geq \lambda_n.$$

Choose breakpoint $n'' \leq n'$ and form $f^{(1)}, \ldots, f^{(n'')}$ and Y associated matrix of these stacked vectors.

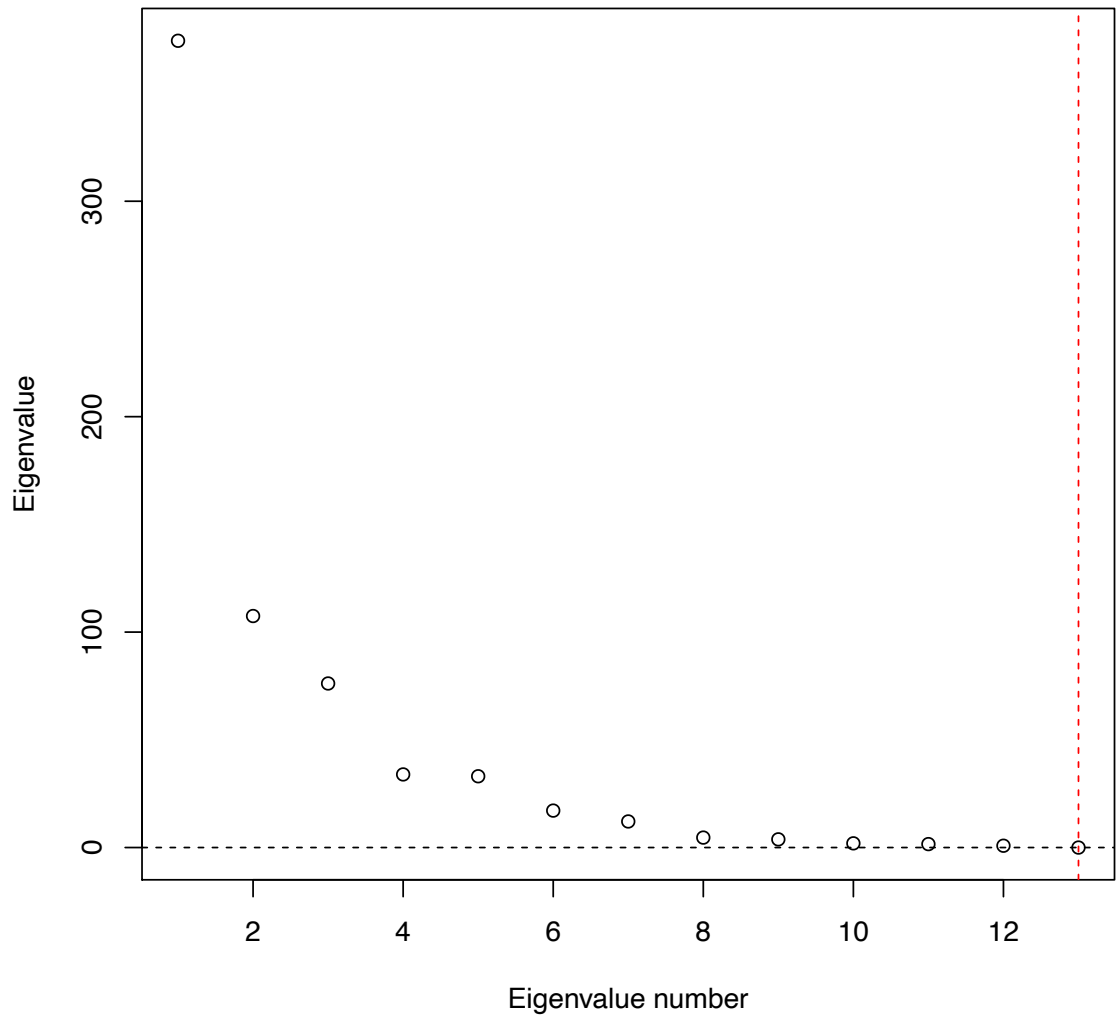
Choice of Breakpoint

1. $n'' = 2$. Often used in practice, as it is then easy to draw and visualise.
2. Look for a large drop in the values of the eigenvalues (and choose n'' to be the number of the eigenvalue immediately before the drop).
3. Trace method. Look at the cumulative sum of the λs. Choose n'' so that
$$\sum_{i=1}^{n''} \lambda_i \approx \mathrm{tr}(B).$$
This is based on the idea that the non-Euclidean error is junk.
4. Magnitude criterion: reject any λ_i which is not $> |\lambda_n|$.

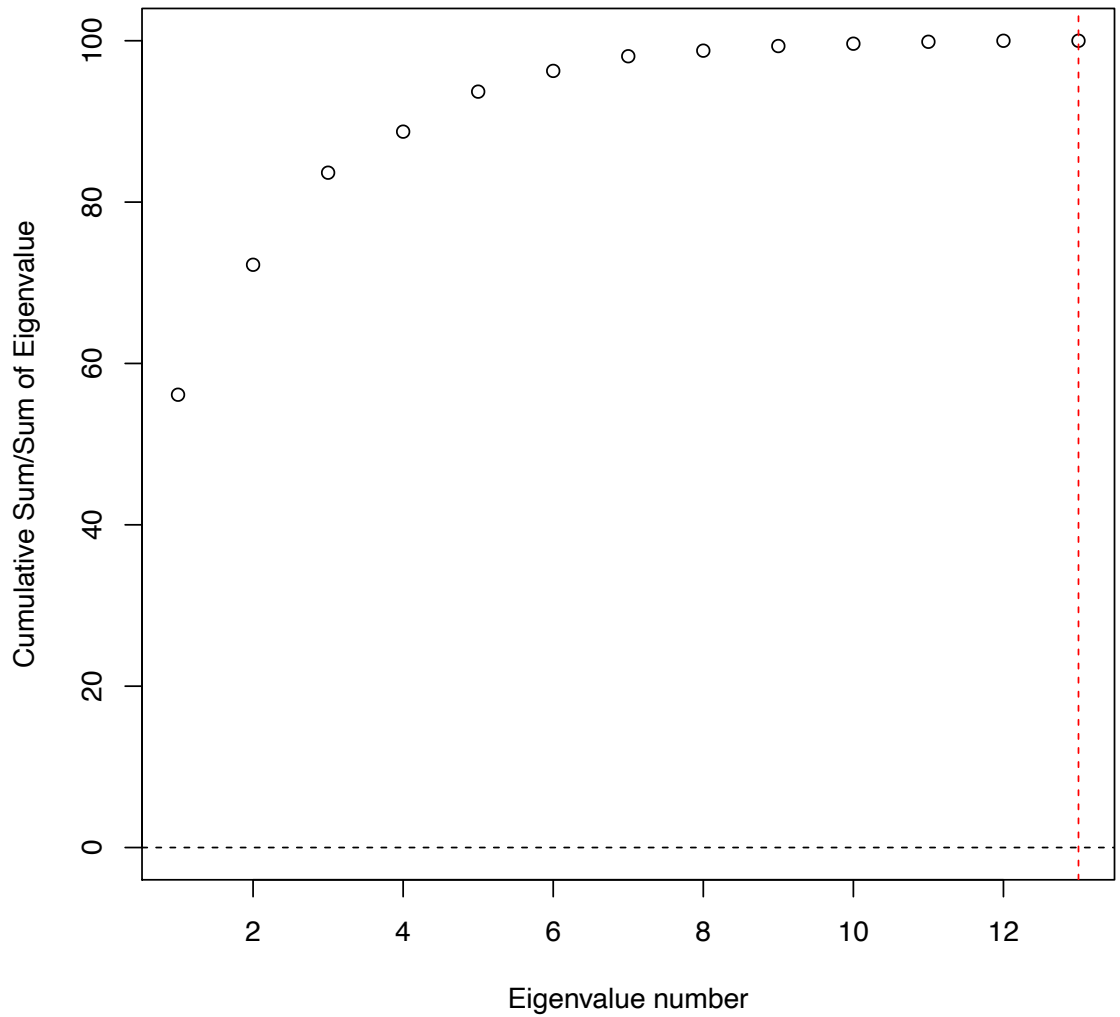
Sicily Classical Scaling

```
sicily.eucdist <- dist(t(sicily))
sicily.euccs <- cmdscale(sicily.eucdist, k=5, eig=TRUE)

# Plot eigenvalues
plot(1:13, sicily.euccs$eig,
, xlab="Eigenvalue number", ylab="Eigenvalue")

# Plot cumulative sum, divided by sumx100
plot(1:13, pvarexp(sicily.euccs$eig), ylim=c(0, 100),
    xlab="Eigenvalue number",
    ylab="Cumulative Sum/Sum of Eigenvalue")

# Where
pvarexp <- function(x)
{
100*cumsum(x)/sum(x)
}
```

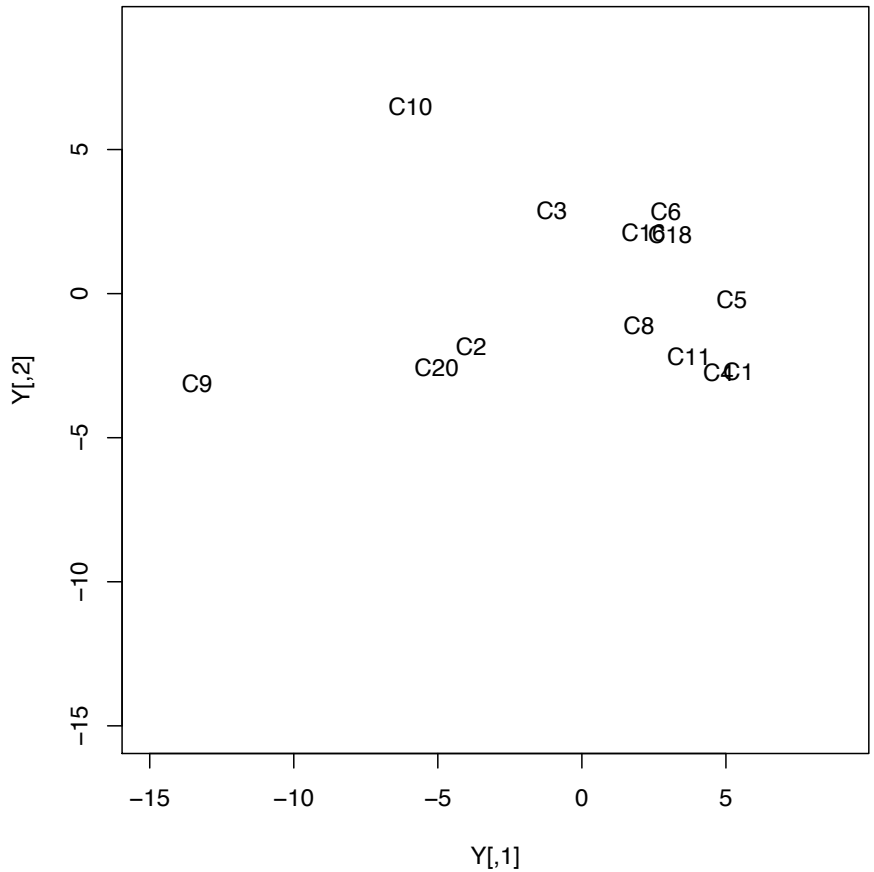
Sicily Eigenvalues



Sicily Cumulative/Sum Eigenvalues



Sicily Two-dimensional Configuration



Configuration is 'almost' one dimensional. In archeology this sometimes happens and the major dimension can align with time (from older huts to newer ones) — although the scaling solution does not tell you the direction of time, other information might help with that.

Elements of Statistical Learning: Lecture 8. Distances + Dissimilaries + Non-metric Scaling

Guy Nason[1]

Department of Mathematics
Imperial College

Being More Flexible

Often, we clearly don't have Euclidean configurations in $\mathbb{R}^n$.

Instead of numbers, we might have *attributes*.

E.g. in a biological application we might have

flower colour $\in \{\text{red, yellow, blue}\}$.

petal number $\in \{0, 1, 2, \ldots\}$.

chromosome number

plant size $\in \{\textit{small}, \textit{medium}, \textit{large}\}$.

Rather than distances, we work with *dissimilarities*.

We now outline some possibilities.

Hamming distance

Given two objects, their Hamming distance is number of times that they disagree on those attributes.

E.g. suppose objects are described by binary strings, each bit corresponding to an attribute,

$$x : 01001011010$$
$$y : 10101101110$$

Hamming distance is often using in information and communication theory.

Hamming Distance by Table

We can write it in tabular form as

		Bits Sent	
		1	0
Bits	1	a	b
Received	0	c	d

The number of mismatches is $h(x, y) = b + c$.

Properties of a Good Dissimilarity

We usually want a dissimilarity d to satisfy the following definition of a metric:

1. $d(x, y) \geq 0$ and $d(x, y) = 0$ if $x = y$ (and sometimes 'only if', but not always. E.g. imagine two plants, of different species, but have the same attribute set).
2. $d(x, y) = d(y, x)$ — but not always true, e.g. one way traffic systems.
3. $d(x, y) + d(y, z) \geq d(x, z)$ — triangle inequality: maybe, maybe not.

Lemma: If $\{d_\alpha\}_{\alpha \in \mathcal{A}}$ is a family of metrics, then $\sum_{\alpha \in \mathcal{A}} d_\alpha$ is a metric.
Proof: Exercise.

Hamming distance is a metric

Suppose we deal with strings of length n. Let

$$d_i(x,y) = \begin{cases} 0 & \text{if } x_i = y_i, \\ 1 & \text{otherwise,} \end{cases}$$

where x_i, y_i is the ith digit of the strings x and y, respectively.

Then, $d_i(x,y)$ is a metric because (i) $d_i(x,y) \geq 0$, clearly; (ii) if $x = y$, then $x_i = y_i$ and $d_i(x,y) = 0$ and for the triangle inequality, we can construct a table with 8 rows:

x_i	y_i	z_i	$d_i(x,y)$	$d_i(y,z)$	Sum	$d_i(x,z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	2	0
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	0	1	1	1	2	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

Hamming Distance is a Metric

So, the quantity $d_i(\cdot, \cdot)$ is a metric.

The Hamming distance $d(x, y) = \sum_{i=1}^{n} d_i(x, y)$ is therefore also a metric by the lemma.

Let's slightly reformulate the table we had earlier. Here, a is the number of times object x and y both have an attribute, similarly b, c, d:

		Attribute x	
		Yes	No
Attribute y	Yes	a	b
	No	c	d

and $a, b, c, d \geq 0$.

Other Similarity/Dissimilarity measures

The *simple matching coefficient* between objects x and y is

$$\frac{a+d}{a+b+c+d}, \tag{1}$$

This is a similarity and due to Sneath.

The *Jaccard distance* is given by

$$d_J(x,y) = \frac{b+c}{a+b+c}. \tag{2}$$

This is subtly different on the denominator and useful for, e.g. archaeological studies, or other studies where you don't care about the absence of something in both objects/locations.

E.g. don't care if two ancient sites are *both* missing Apple iPhones!

Is Jaccard distance a metric?

Clearly, $d_J(x, y) \geq 0$.

Also, $d_J(x, y) = 0$ is attributes match exactly.

Also, $d_J(x, y) = d_J(y, x)$ (exchange b, c in formula makes no difference).

What about the triangle inequality?

Jaccard Triangle Assistance Lemma

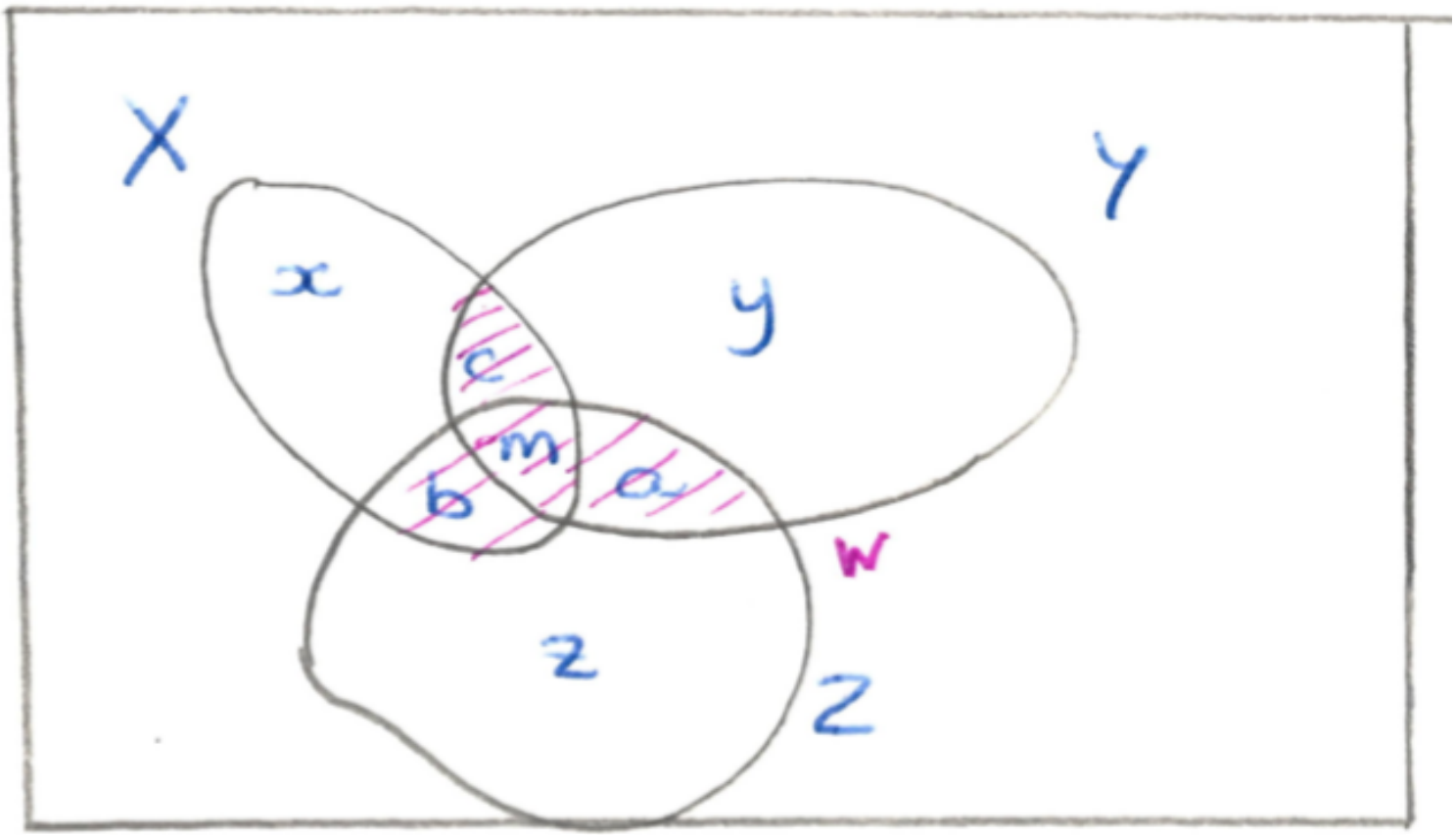
Let $r, s, t \geq 0$ and $r \leq s$.

Lemma: $r/s \leq (r+t)/(s+t)$.

Proof: Since $r \leq s$, we have $0 \leq w = r/s \leq 1$. Hence:

$$
\begin{aligned}
w\frac{t}{s} &\leq \frac{t}{s} && (3)\\
\implies w + w\frac{t}{s} &\leq w + \frac{t}{s} && (4)\\
\implies w(1 + t/s) &\leq w + \frac{t}{s} && (5)\\
\implies w &\leq \frac{w + t/s}{1 + t/s} && (6)\\
\implies r/s &\leq (r+t)/(s+t). && (7)
\end{aligned}
$$

Triangle Inequality for Jaccard distance



Nb: letters a, b, c are different from those above!

Proof of Jaccard distance is metric

Let $W = (X \cap Y) \cup (Y \cap Z) \cup (X \cap Z)$.

Now $d_J(X, W) = (x + a)/(x + a + b + c + m)$, and

$d_J(Z, W) = (z + c)/(z + a + b + c + m)$.

Then

$$\begin{aligned} d_J(X, W) + d_J(Z, W) &\geq (x + a + z + c)/(x + z + a + b + c + m) \\ &= d_J(X, Z) \end{aligned}$$

see by splitting sum in two and that denominator gets addition of z and x, respectively.

Proof of Jaccard distance is metric — 2

Now

$$\begin{aligned} d_J(X,W) &= (x+a)/(x+a+b+c+m) \\ &\leq (x+a+b)/(x+a+b+c+m) \\ &\leq (x+a+b+y)/(x+y+a+b+c+m) \\ &= d_J(X,Y), \end{aligned}$$

the last step uses the Assistance Lemma.

Similarly, $d_J(Z,W) \leq d_J(Z,Y)$.

Hence $d_J(X,Y) + d_J(Y,Z) \geq d_J(X,W) + d_J(Z,W) \geq d_J(X,Z)$, which shows Jaccard distance satisfies the triangle inequality (proof due to K. Moody).

Some Other Dissimilarities

Maximum this is the L_∞ norm.

Manhattan Distance or 'city-block' as it is the distance you cover when walking in a regular block city-scape from point A to point B. If you have two points, x, u in p dimensions it can be written $|x_1 - u_1| + |x_2 - u_2| + \cdots + |x_p - u_p|$. It is also the L_1 metric.

Canberra distance similar to Manhattan, but weighted, e.g. $\sum_{q=1}^{p} \frac{|x_q - u_q|}{|x_q| + |u_q|}$. (The authors Lance and Williams worked in Canberra, Australia: I think that is the origin of the name.)

Gower's similarity coefficient

Gower's similarity coefficient between objects i, j is $S_{i,j} = \frac{\sum_{q=1}^{p} w_{i,j,q} S_{i,j,q}}{\sum_{q=1}^{p} w_{i,j,q}}$, where $w_{i,j,q}$ quantifies your confidence in variable q.

$S_{i,j,q}$ is the similarity between i, j on variable q.

If variable q is ordinal or continuous then $S_{i,j,q} = 1 - |x_{i,q} - x_{j,q}|/r_q$, where r_q is the range of values on the qth variable.

If variable q is nominal then $S_{i,j,q} = 1$ if $x_{i,q} = x_{j,q}$ or 0 otherwise.

Gower's similarity coefficient — 2

For binary q we have this table:

	Case i	Case j	$S_{i,j,q}$	$w_{i,j,q}$
Value of attribute q	+	+	1	1
	+	−	0	1
	−	+	0	1
	−	−	0	0

Reminiscent of the Jaccard coefficient, and, in fact, if all variables are binary, then Gower's coefficient (1-similarity) is Jaccard.

Gower's similarity coefficient is extremely useful for mixed variables.

Ordinal or Non-metric or Shepard-Kruskal Scaling

Essential for when dissimilarities are not Euclidean. In this case we will

observe a set of dissimilarites $\delta_{m,\ell}$, which are not necessarily Euclidean distances.

postulate a set of distances, $d_{m,\ell}$, which result in some configuration that we've "invented"

Non-metric scaling: Basic procedure

Essentially, create a 'reasonable' configuration from the dissimilarities $\{\delta_{m,\ell}\}_{m,\ell}$: the created configuration is X which has a set of *distances*: $\{d_{m,\ell}\}_{m,\ell}$.

We then compute the *stress* function (or penalty function) that measures the degree of agreement of $\{\delta_{m,\ell}\}_{m,\ell}$ with $\{d_{m,\ell}\}_{m,\ell}$.

We then minimise the stress over all possible configurations.

Difficulty: One cannot do anything (e.g. arithmetic) on the δs, but we would like a differentiable stress function.

Measuring the closeness of dissimilarities to distances

We start by creating a configuration of n points X in K dimensions (and have to choose K, more later).

The (Euclidean) distances between each point are $d_{m,\ell}$.

We create a set of fitted distances $\hat{d}_{m,\ell}$ from the $d_{m,\ell}$ by computing a monotone linear regression fit, where the $\hat{d}_{m,\ell}$ are in the *same order* as the ordered dissimilarities $\delta_{m,\ell}$ of the points.

Both d and $\hat{d}$ depend on the coordinates in the configuration X, but the $\hat{d}$ are a kind of *proxy* for the δs.

Temporarily, to ease notation. We rename $\{d_{m,\ell}\}$ to be y_i for $i = 1, \ldots, I$ and rename $\{\hat{d}_{m,\ell}\}$ to be z_i, $i = 1, \ldots, I$.

Monotone Linear Regression

Given observations $\{y_i\}_{i=1}^I$ the task is to minimise $\sum_{i=1}^I (y_i - z_i)^2$, the residual sum of squares of $\{y_i\}$ about $\{z_i\}$ where we minimize over $z_1 \leq z_2 \leq \cdots \leq z_I$.

There are various strategies to find the best $\{z_i\}_{i=1}^I$ set.

The *Miles Algorithm* carries out the following steps:

1. Write down the y_is in singleton blocks.
2. Scanning from left to right, unite any two blocks where there is not a strict increase from left to right (uniting means forming the mean of all elements in both blocks, merging the blocks, keeping the same number of entries, but each entry replaced by the mean).
3. Keep checking until all boundaries are satisfied.

Miles Algorithm Example

Suppose the y_i are $2, 1, 17, 62, 7, 13, 4, 25, 9$, which correspond one-to-one to a set of z_i *in increasing order*.

Write them down in singleton blocks

$$| \ 2 \ | \ 1 \ | \ 17 \ | \ 62 \ | \ 7 \ | \ 13 \ | \ 4 \ | \ 25 \ | \ 9$$

Now search for first occurrence of non-strict-increasing

$$| \ 2 \ | \ 1 \ | \ 17 \ | \ 62 \ | \ 7 \ | \ 13 \ | \ 4 \ | \ 25 \ | \ 9$$

Unite these two $(2 + 1)/2 = 3/2 = 1.5$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 62 \ | \ 7 \ | \ 13 \ | \ 4 \ | \ 25 \ | \ 9$$

Now look for others

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 62 \ | \ 7 \ | \ 13 \ | \ 4 \ | \ 25 \ | \ 9$$

Miles Algorithm Example — 2

Unite these blocks. Mean of 62 and 7 is $(62 + 7)/2 = 34.5$, other two means are 8.5 and 17

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 34.5 \ 34.5 \ | \ 8.5 \ 8.5 \ | \ 17 \ 17$$

Next scan, gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 34.5 \ 34.5 \ | \ 8.5 \ 8.5 \ | \ 17 \ 17$$

Unite: $(34.5 + 34.5 + 8.5 + 8.5)/4 = 86/4 = 21.5$ and next scan

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 21.5 \ 21.5 \ 21.5 \ 21.5 \ | \ 17 \ 17$$

Mean of two blocks is $(21.5 \times 4 + 17 \times 2)/6 = 20$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 20 \ 20 \ 20 \ 20 \ 20 \ 20$$

Notes on Miles Algorithm

The algorithm must terminate because:

1. The number of blocks decreases at each step, or algorithm stops.
2. If algorithm continues, it must eventually stop when we are left with one single block with all elements in.

At the end, we obtain a monotone sequence which consists of a set of blocks containing constant values, with a strict increase from block to block.

The function we obtain is piecewise constant.

This algorithm gives us the optimum (but we won't prove it).

There are variants, which are more efficient

Young's boundary search algorithm (optional - for GN)

Recursive from left.

Given monotone regression on $1, \ldots, i$, append value at $i + 1$ and check back to left until we need unite no more.

Starting blocks:

$$| \; 2 \; | \; 1 \; | \; 17 \; | \; 62 \; | \; 7 \; | \; 13 \; | \; 4 \; | \; 25 \; | \; 9$$

Then

$$| \; 1.5 \; 1.5 \; | \; 17$$

O.k., don't need to unite:

$$| \; 1.5 \; 1.5 \; | \; 17 \; | \; 62$$

O.k., don't need to unite:

Young's boundary search algorithm — 2

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 62 \ | \ 7$$

We do need to unite $(62 + 7)/2 = 69/2 = 34.5$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 34.5 \ 34.5 \ | \ 13$$

We do need to unite $(69 + 13)/3 = 82/3 = 27\frac{1}{3}$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 27\tfrac{1}{3} \ 27\tfrac{1}{3} \ 27\tfrac{1}{3} \ | \ 4$$

We do need to unite $(82 + 4)/4 = 86/4 = 21.5$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 21.5 \ 21.5 \ 21.5 \ 21.5 \ | \ 25$$

O.k., don't need to unite:

Young's boundary search algorithm — 3

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 21.5 \ 21.5 \ 21.5 \ 21.5 \ | \ 25 \ | \ 9$$

We do need to unite $(25 + 9)/2 = 34/2 = 17$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 21.5 \ 21.5 \ 21.5 \ 21.5 \ | \ 17 \ 17$$

Unite last boundary $(21.5 \times 4 + 17 \times 2)/6 = 120/6 = 20$ gives

$$| \ 1.5 \ 1.5 \ | \ 17 \ | \ 20 \ 20 \ 20 \ 20 \ 20 \ 20 \ ,$$

which gives same answer as before.

There are other variants (due to Kruskal and another by Sibson) where you grow the sequence alternately right and left, starting at the left most, or that plus you keep going once you're winning!

The Stress function

Given a configuration and a set of dissimilarities $\{\delta_{m,\ell}\}$ we can form the distances $\{d_{m,\ell}\}$ and the least squares monotone regression fit, $\{\hat{d}_{m,\ell}\}$ to the d using δs.

Let $S^* = \sum_{m<\ell}(d_{m,\ell} - \hat{d}_{m,\ell})^2$ be the residual sum of squares from the least squares monotone regression.

Let $T^* = \sum_{m<\ell} d_{m,\ell}^2$.

Then the stress function S is given by

$$S(X) = \sqrt{S^*/T^*}. \tag{8}$$

Note that S depends on the configuration X, as both S^*, T^* do, because both $d_{m,\ell}$ and $\hat{d}_{m,\ell}$ do also.

"Created" Configuration and Optimisation

We choose a configuration X, i.e. coordinates for n points in K dimensions (and we have to choose K).

Hence, S is a function of NK variables (although a few degrees of freedom are lost as creation of d loses orientation and position information AND S is scale invariant (division by T^*).

Initial Configuration: Could start with a random configuration. E.g. all entries of X have random independent uniform distribution on $[0, 1]$ (or anything really).

Optimisation: huge subject, all by itself. We can evaluate $S, \nabla S$ with respect to all $x_{i,k}$ and use an iterative hill-descending algorithm to search for the optimum. However, S is usually highly non-quadratic and so easily gets stuck in local minima.

Initial Configurations

(i) Kruskal suggested an 'L'-shaped configuration.

(ii) Random starts (as above), but often repeat the procedure each time on a set of random start configurations to see if they all end up at the same place.

(iii) 'high to low' strategy: e.g. $6 \to 4 \to 2$. Start with $K = 6$ dimensions; Iterate to Optimality. Then project this solution to 4 principal directions (e.g. using principal components). Take this and iterate to optimality. Then project the 4-dimensional solution to 2 principal directions. Then iterate to optimality.

(iv) Start from the classical scaling solution — useful for avoiding local optima. However, this does involve giving dissimilarities numerical values somehow.

Choice of (Final Dimensionality)

1. $K = 1$ does not work! S ALWAYS gets stuck in local optima — this is because points "can't push through one another on the line"
2. Often a good idea to have K larger than 1, than you eventually want
3. traditionally look for 'elbows' on an optimal stress versus dimension graph (e.g. 3D 'true' configuration gives high stress in $K = 1$ or $K = 2$, but low stress in $K = 3$.
4. interpretability: humans can easily visualise $K = 3$, $K = 2$ useful for paper graphics. 1D might be useful for seriation problems, where you are interested in the one dimension of time.

Elements of Statistical Learning: Lecture 9. Stress and Stretching

Guy Nason[1]

Department of Mathematics
Imperial College

Differentiation of the Stress

Recall configuration $X_{n\times K}$ and stress $S = \sqrt{S^*/T^*}$, where

$S^* = \sum_{m<\ell}(d_{m,\ell} - \hat{d}_{m,\ell})^2$ and $T^* = \sum_{m<\ell} d^2_{m,\ell}$.

So

$$\frac{\partial S}{\partial x_{i,k}} = \frac{1}{2S}\left\{\frac{T^*\frac{\partial S^*}{\partial x_{i,k}} - S^*\frac{\partial T^*}{\partial x_{i,k}}}{(T^*)^2}\right\} \quad (1)$$

$$= \frac{S}{2}\left\{\frac{1}{S^*}\frac{\partial S^*}{\partial x_{i,k}} - \frac{1}{T^*}\frac{\partial T^*}{\partial x_{i,k}}\right\}. \quad (2)$$

Here: m, ℓ correspond to objects/individuals: range $1, \ldots, n$ and

k corresponds to a dimension of the proposed config: $k = 1, \ldots, K$ and i corresponds to an individual, $i = 1, \ldots, n$.

Derivative of T^*

Now

$$\frac{\partial T^*}{\partial x_{i,k}} = \sum_{m<\ell} \frac{\partial d^2_{m,\ell}}{\partial x_{i,k}}, \tag{3}$$

where

$$d^2_{m,\ell} = \sum_{p=1}^{K} (x_{m,p} - x_{\ell,p})^2. \tag{4}$$

So

$$\frac{\partial d^2_{m,\ell}}{\partial x_{i,k}} = 2(x_{m,k} - x_{\ell,k}) \left\{ \frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{\ell,k}}{\partial x_{i,k}} \right\}. \tag{5}$$

Note: $\frac{\partial x_{m,k}}{\partial x_{i,k}} = 0$ unless $m = i$ and $\frac{\partial x_{\ell,k}}{\partial x_{i,k}} = 0$ unless $\ell = i$. So

$$\frac{\partial T^*}{\partial x_{i,k}} = 2 \sum_{m<\ell} (x_{m,k} - x_{\ell,k}) \left\{ \frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{\ell,k}}{\partial x_{i,k}} \right\}. \tag{6}$$

Derivative of S^*.

Examine

$$\frac{\partial (d_{m,\ell} - \hat{d}_{m,\ell})^2}{\partial x_{i,k}} = 2(d_{m,\ell} - \hat{d}_{m,\ell}) \left(\frac{\partial d_{m,\ell}}{\partial x_{i,k}} - \frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}} \right). \quad (7)$$

Note that

$$\frac{\partial d_{m,\ell}^2}{\partial x_{i,k}} = 2 d_{m,\ell} \frac{\partial d_{m,\ell}}{\partial x_{i,k}}, \quad (8)$$

so

$$\frac{\partial d_{m,\ell}}{\partial x_{i,k}} = \frac{1}{2 d_{m,\ell}} \frac{\partial d_{m,\ell}^2}{\partial x_{i,k}} \quad (9)$$

$$= \frac{1}{2 d_{m,\ell}} \cdot 2(x_{m,k} - x_{\ell,k}) \left\{ \frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{\ell,k}}{\partial x_{i,k}} \right\}, \quad (10)$$

from (5).

Derivative of S^* continued

What about $\frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}}$? Fortunately, we don't have to work it out. Why? Well, we can write the derivative of S^* as

$$\frac{\partial S^*}{\partial x_{i,k}} = \sum_{m<\ell} 2(d_{m,\ell} - \hat{d}_{m,\ell}) \frac{\partial d_{m,\ell}}{\partial x_{i,k}} - \sum_{m<\ell} 2(d_{m,\ell} - \hat{d}_{m,\ell}) \frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}} \quad (11)$$

Let's examine the second part of this and write the sum over m, ℓ as over 'blocks' and 'sum within blocks' in the least squares monotone regression, i.e.

$$\sum_{\text{blocks}} \sum_{\text{within blocks}} 2(d_{m,\ell} - \hat{d}_{m,\ell}) \frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}} \quad (12)$$

Within a block $\hat{d}_{m,\ell}$ is constant, so we can take $\frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}}$ out of the inner sum, leaving

Derivative of S^* continued — 2, and then S

$$\sum_{\text{within blocks}} 2(d_{m,\ell} - \hat{d}_{m,\ell}), \tag{13}$$

but $\hat{d}_{m,\ell}$ is the average in a block of the 'within block values' so this sum is zero. Hence, all of the second term is zero, so we don't need to work out $\frac{\partial \hat{d}_{m,\ell}}{\partial x_{i,k}}$.

Hence, putting it all together from (2), (6), (11)

$$\frac{\partial S}{\partial x_{i,k}} = \frac{S}{2}\left\{\frac{1}{S^*}\frac{\partial S^*}{\partial x_{i,k}} - \frac{1}{T^*}\frac{\partial T^*}{\partial x_{i,k}}\right\} \tag{14}$$

$$= \frac{S}{2}\left[\frac{1}{S^*}\left\{\sum_{m<\ell} 2(d_{m,\ell} - \hat{d}_{m,\ell})\frac{\partial d_{m,\ell}}{\partial x_{i,k}}\right\}\right. \tag{15}$$

$$\left. -\frac{2}{T^*}\left\{\sum_{m<\ell}(x_{m,k} - x_{\ell,k})\left(\frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{\ell,k}}{\partial x_{i,k}}\right)\right\}\right] \tag{16}$$

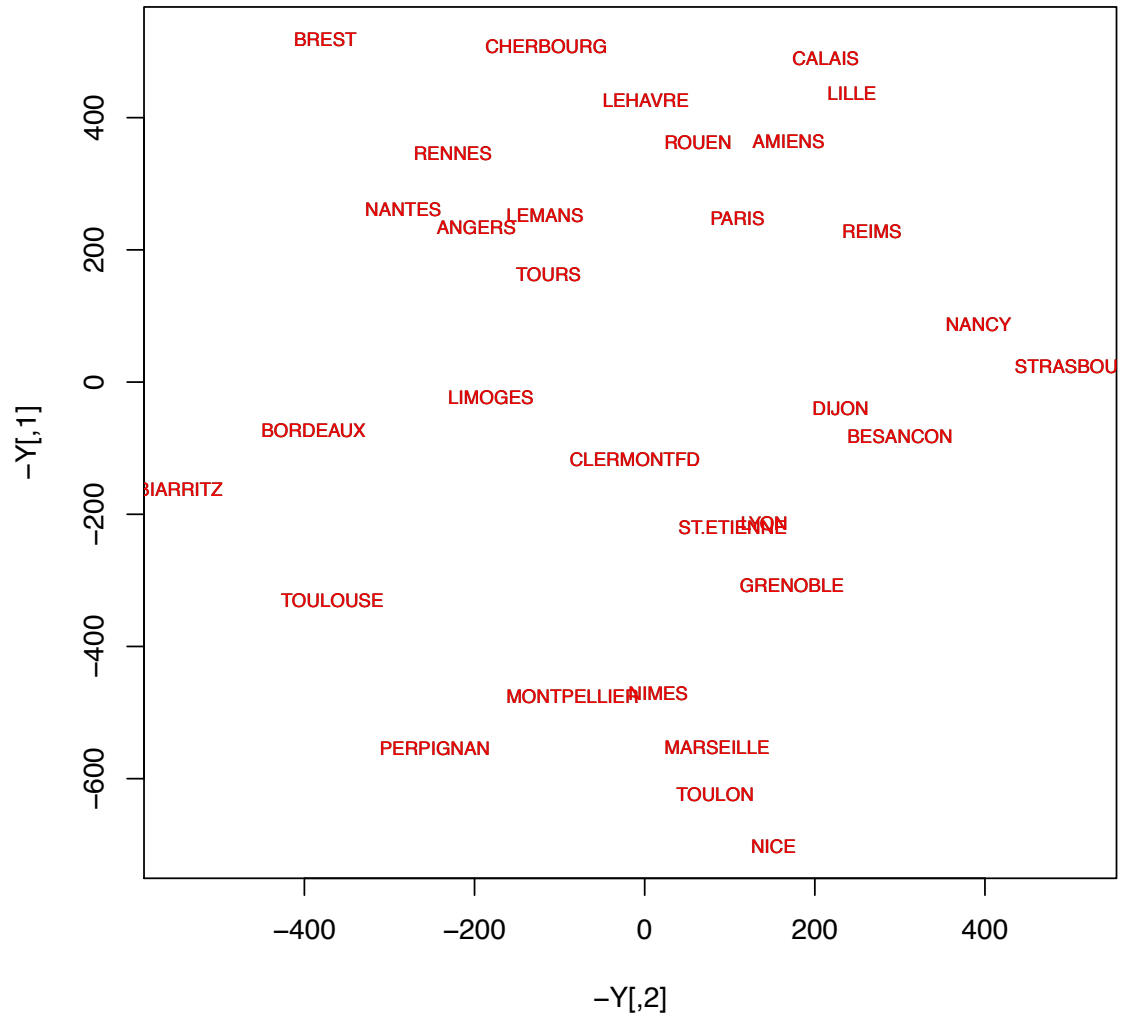
Notes on Stress Derivative

S is exactly once continuously differentiable (e.g. so perhaps not useful to look + use second derivative in optimisation algorithm).

Given the complexity of S, and its computation via least squares monotone regression, it's kind of remarkable that it is once continuously differentiable, and hence provides very useful information to optimisers.

France road distances using ordinal scaling

```
frisosoln <- isoMDS(d=frdistobj)
# Plot plots BOTH solutions superimposed!
```

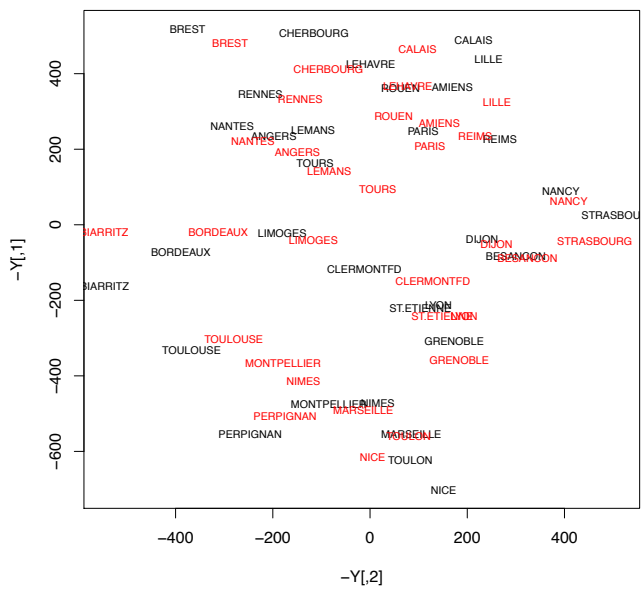


France road distances using ordinal scaling — random start

```
frisosoln.random1 <- isoMDS(d=frdistobj,
    y=matrix(runif(64), nrow=32, ncol=2))
```

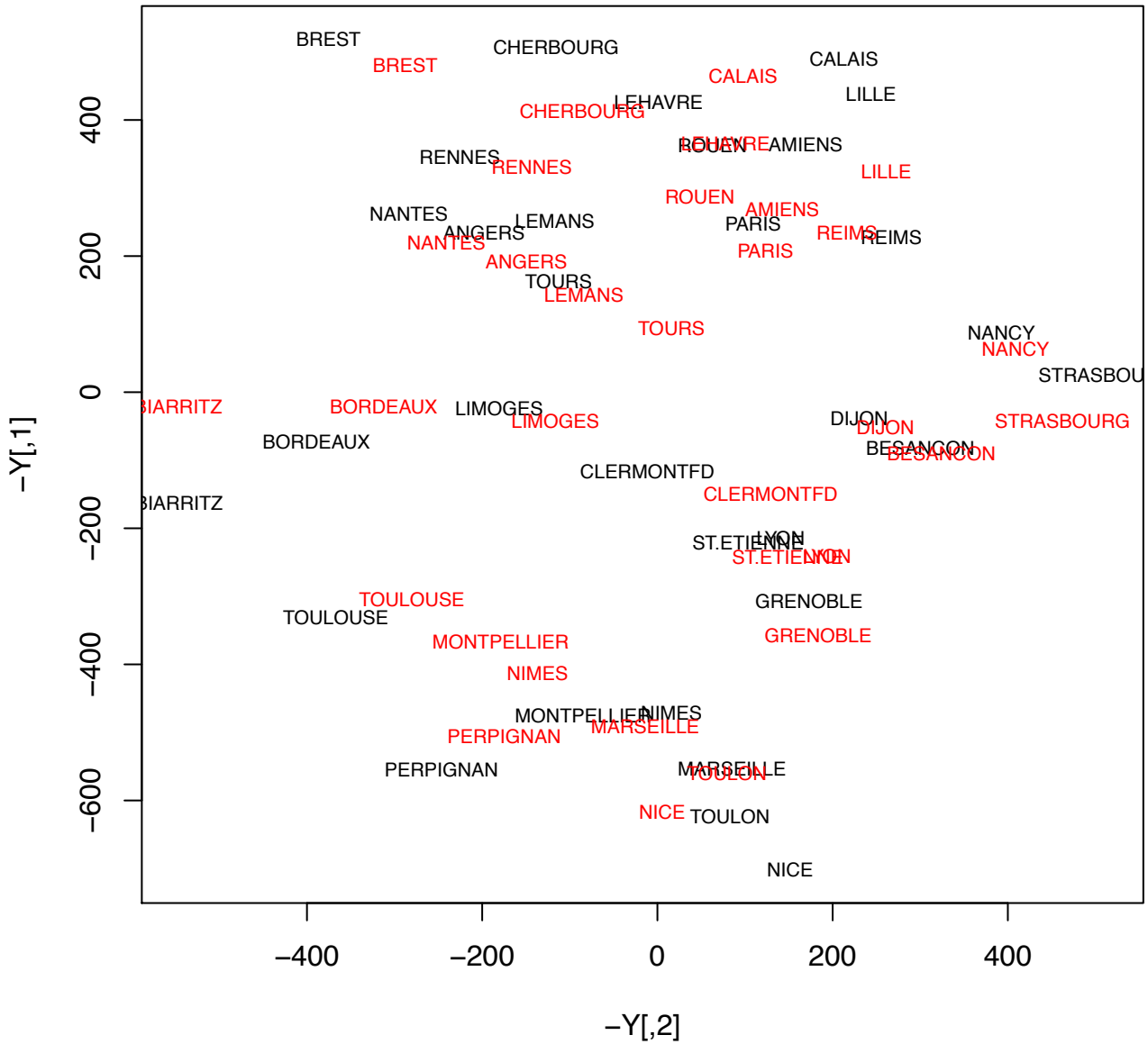
The next plot plots the original scaling solution and superimposes the random start obtained configuration in red.

However, the red solution has been 'blown up' by a factor of 25



See next page for bigger version.

France road distances using ordinal scaling — random start



Sicily: ordinal scaling on 'Euclidean' distances

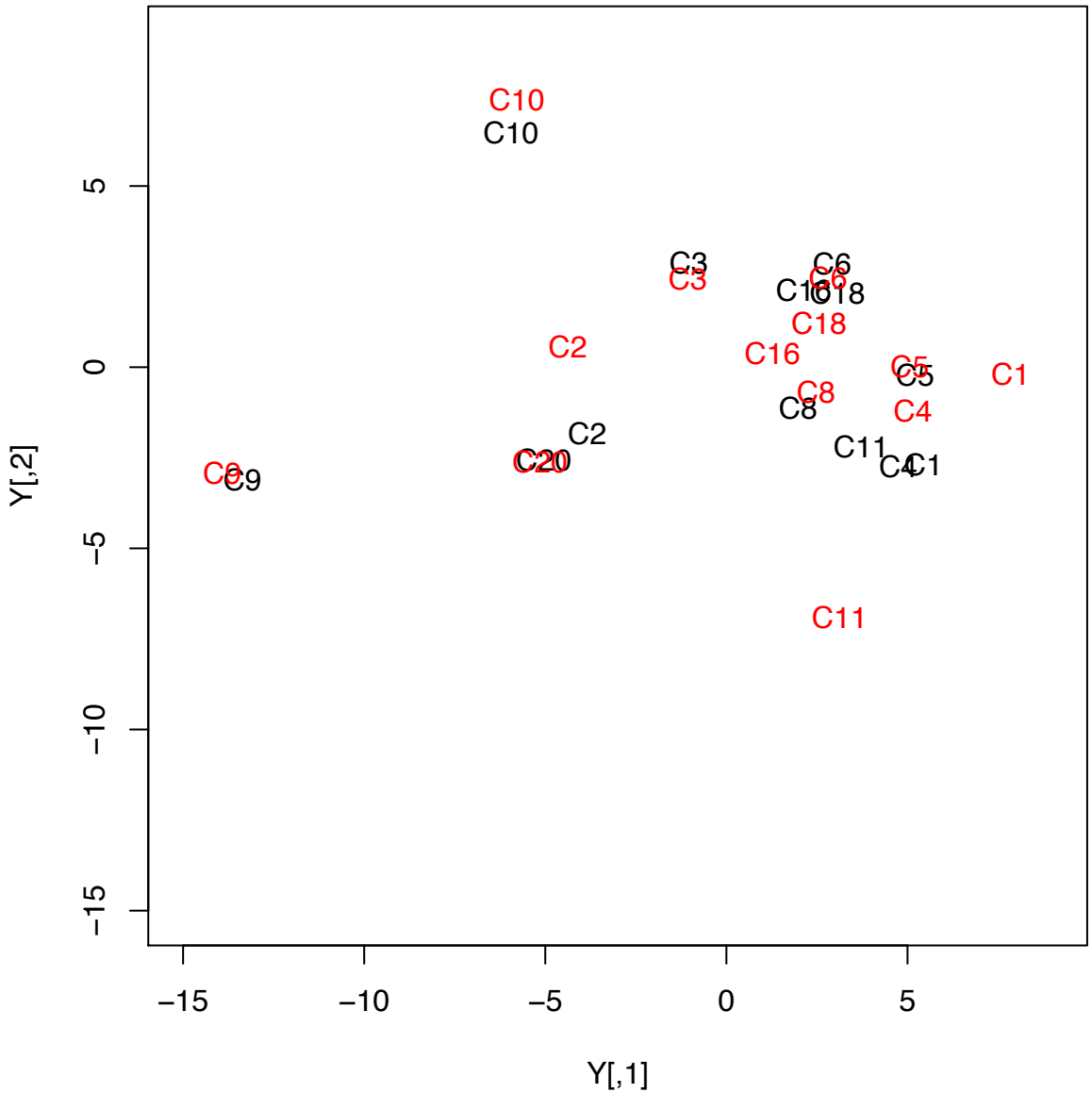
```
> sicily.eusiso <- isoMDS(sicily.eucdist)
initial  value 14.635084
iter   5 value 8.217162
iter  10 value 6.790808
final  value 6.758704
converged
```

The `isoMDS` function prints out the stress value as it changes every five iterations.

There is a significant reduction in stress, so we'd expect the scaling solution to be different.

```
alim <- c(-15, 9)
oldpar <- par(pty="s")
plot(sicily.euccs$points[,1], sicily.euccs$points[,2], type="n",
    xlim=alim, ylim=alim, xlab="Y[,1]", ylab="Y[,2]")
text(sicily.euccs$points[,1], sicily.euccs$points[,2],
    lab=dimnames(sicily)[[2]])
text(sicily.eusiso$points[,1], sicily.eusiso$points[,2],
    lab=dimnames(sicily)[[2]], col=2)
par(oldpar)
```

Sicily: classical scaling solution and ordinal from classical



Sicily: using a different distance measure

Let's use the `binary` method for computing distances.

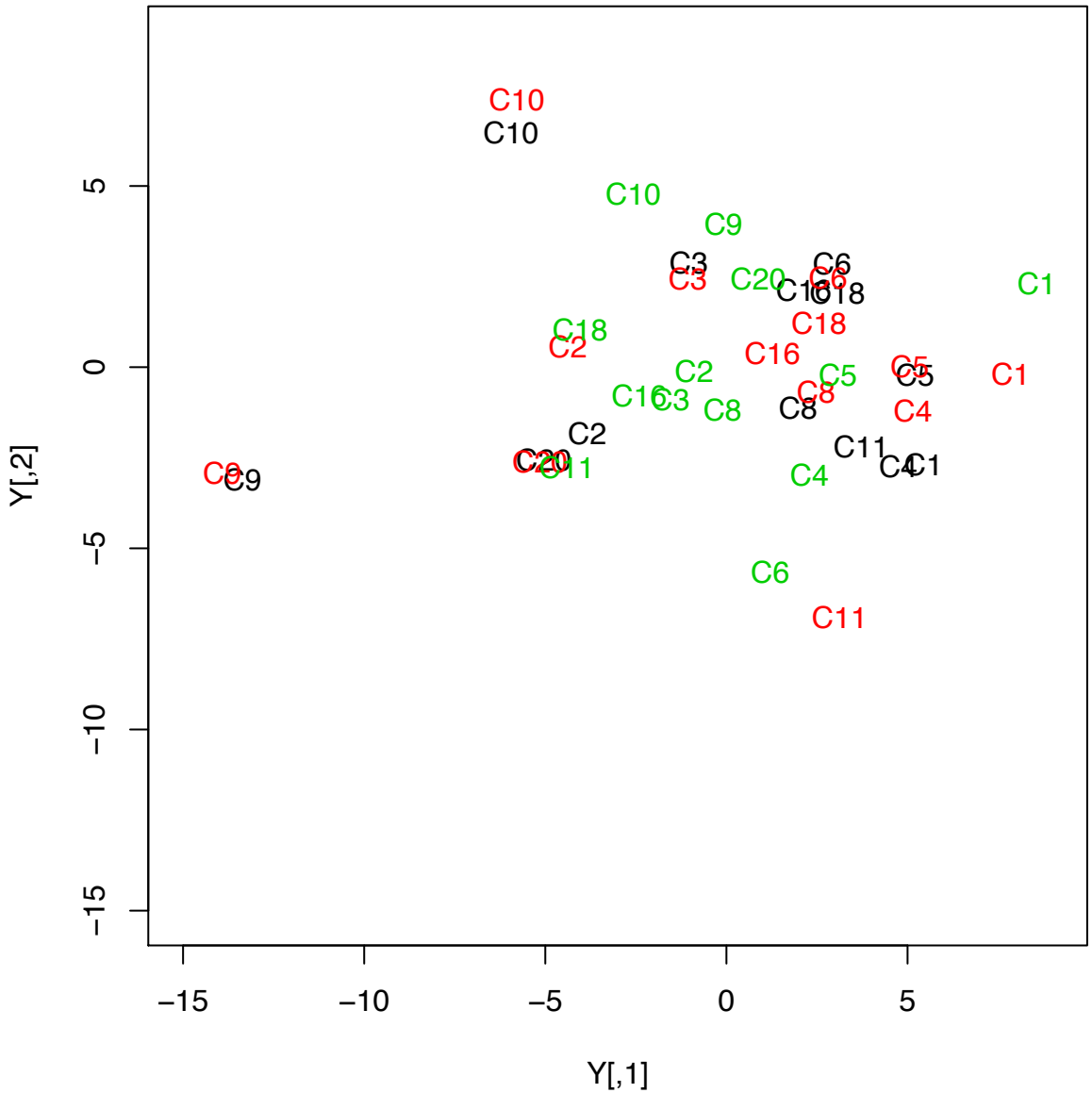
Although the Sicily entries are not zero/ones, the `binary` method computes Jaccard.

```
> sicily.bindist <- dist(t(sicily),method="binary")
> sicily.biniso <- isoMDS(sicily.bindist)
initial  value 18.890930
iter   5 value 13.354798
final  value 13.099448
converged
```

Again, stress is considerably reduced, so ordinal scaling is changing the configuration.

In fact, configuration is quite different. Probably need some way of matching configurations as origin & orientation are different.

Sicily: classical+ordinal solutions and ordinal from `binary`



Missing Distances: great practical importance

The `isoMDS()` and ordinal scaling can cope with missing distances.

That is, if you don't know, or can't work out, the dissimilarity between two objects then you can put `NA` into the distance matrix.

It won't work with a high proportion of missing observations (e.g. 90% is doubtful).

It also won't work if the missingness is highly concentrated on one object (e.g. if all the distances to an object are `NA`, then that object cannot be placed).

You can also put in ∞ dissimilarities (but we won't use this here).

Missing distances are just omitted from the stress calculation.

French Road Distances: code for missing example

```
fig.frmissing <- function (no=1)
{
# Establish basic parameters
nrnc <- nrow(frdist)
nvals <- nrnc^2

# Generate initial config and oMDS soln using
# full distances
set.seed(101)
init.config <- matrix(runif(nrnc*2), nrow=nrnc, ncol=2)
friso0soln <- isoMDS(as.matrix(frdist), y=init.config)

# Plot "full" answer
plot(friso0soln$points[,1], -friso0soln$points[,2],
    xlab="-Y[,2]", ylab="-Y[,1]", type="n")
text(friso0soln$points[,1], -friso0soln$points[,2],
    labels=dimnames(frdist)[[1]], cex=0.7, col=1)
   ...
```

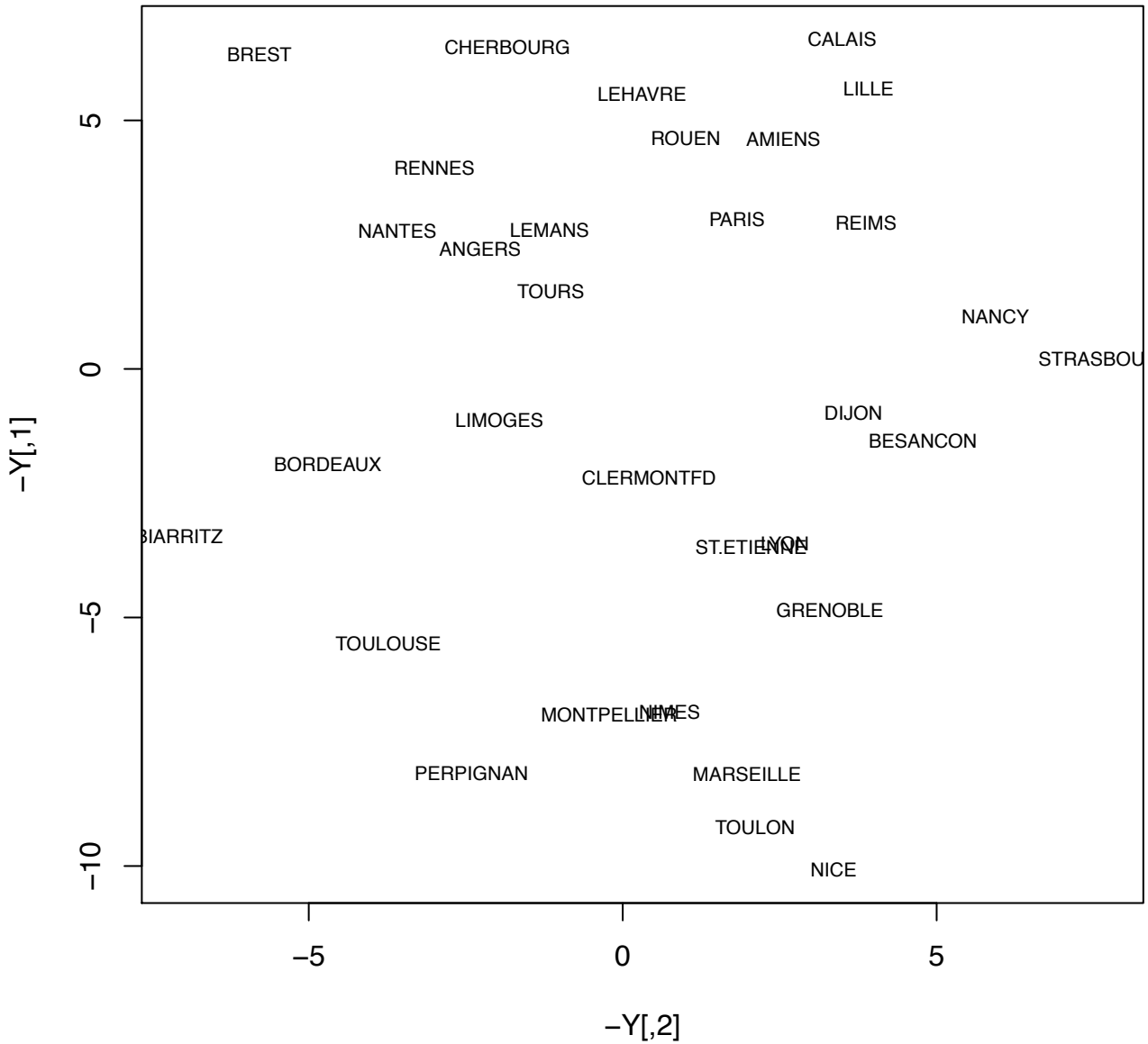
French code example — 2

```
...
if (no==0)
    return()

# Turn distance matrix into vector
frdistvec <- as.vector(as.matrix(frdist))

# Introduce 5% missing vals, and turn back to matrix
na.ix <- sample(1:nvals, size=0.05*nvals)
frdistvec5 <- frdistvec
frdistvec5[na.ix] <- NA
frdist5 <- matrix(frdistvec5, nrow=nrnc, ncol=nrnc)

friso5soln <- isoMDS(as.matrix(frdist5), y=init.config)

text(friso5soln$points[,1], -friso5soln$points[,2],
    labels=dimnames(frdist)[[1]], cex=0.7, col=2)
    ...
```

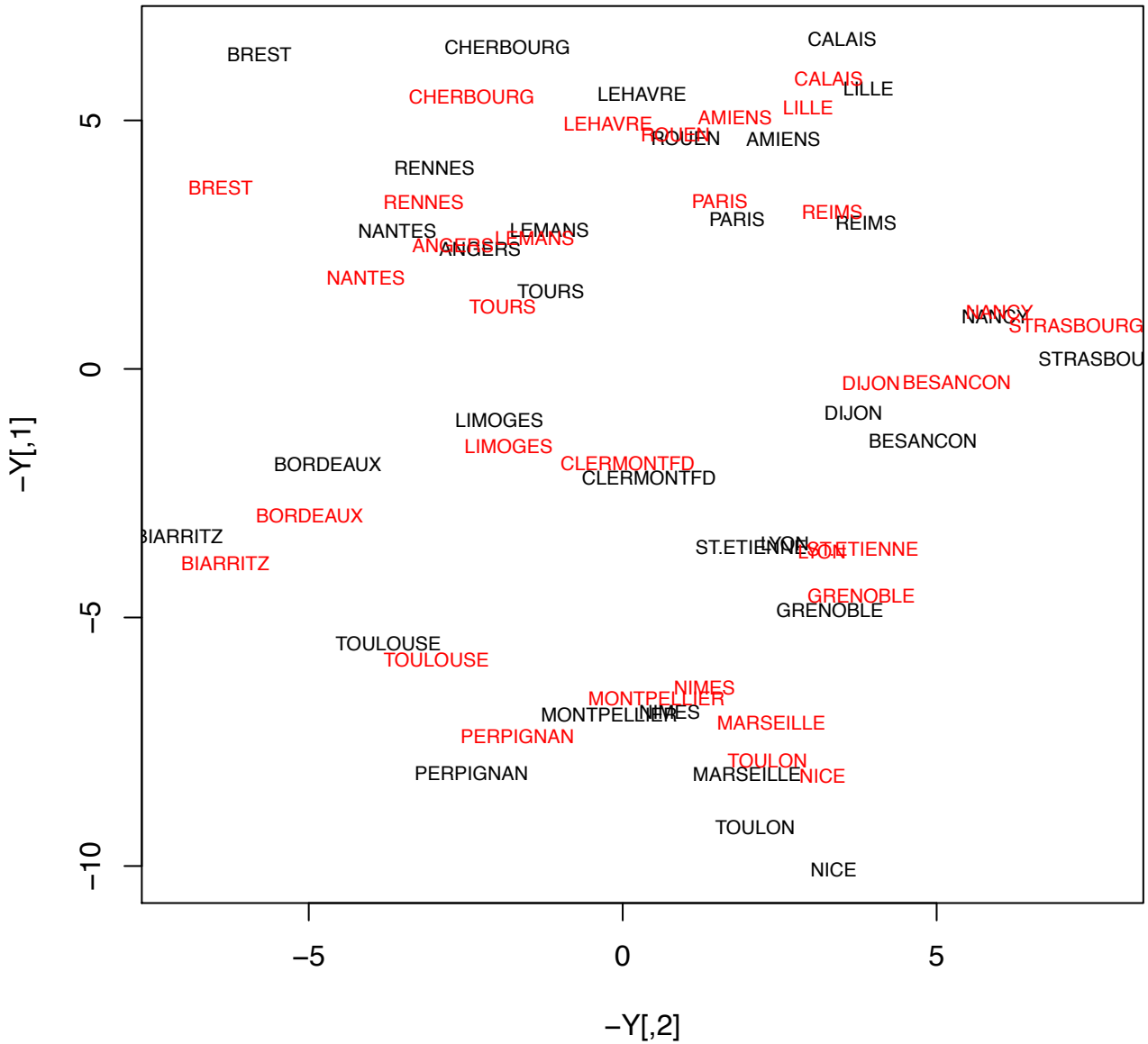
French code example — 3

```
...
if (no==1)
    return()

# Introduce 20% missing vals, and turn back to matrix
na.ix <- sample(1:nvals, size=0.2*nvals)
frdistvec20 <- frdistvec
frdistvec20[na.ix] <- NA
frdist20 <- matrix(frdistvec20, nrow=nrnc, ncol=nrnc)

friso20soln <- isoMDS(as.matrix(frdist20), y=init.config)

text(friso20soln$points[,1], -friso20soln$points[,2],
    labels=dimnames(frdist)[[1]], cex=0.7, col=3)
}
```

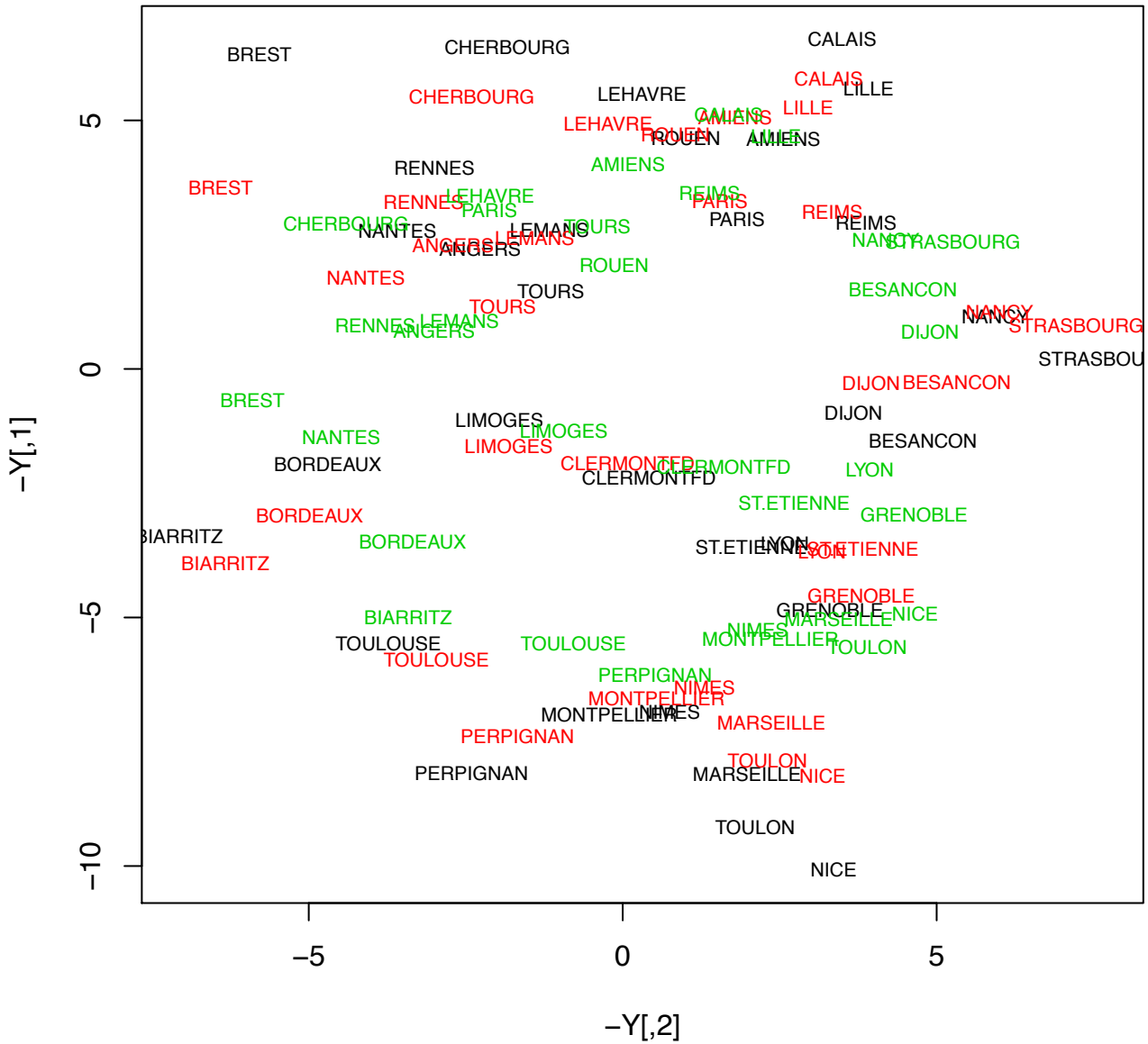
French Road Distances: ordinal scaling random start



Continued: ordinal scaling random start + 5% missing



Continued: ordinal scaling random start + 5, 20% missing



k-means clustering

Here, suppose we have a $n \times p$ configuration X.

Suppose we want to divide the n observations into k clusters.

A k-means clustering solution is one where the k cluster means are defined as being the mean of all points in the cluster *and* a point is in cluster $j = 1, \ldots, k$ if and only if it is closer to the jth cluster mean than any other cluster mean.

Note: choice of k is crucial to the success of the algorithm, and its often difficult to know what it should be.

k-means algorithm

Start with an initial set of k (p-dimensional) mean vectors $m_1^{(1)}, \ldots, m_k^{(1)}$. Each step s consists of:

Assignment: assign each observation $\mathbf{X_i}$ to the cluster whose mean has the least squared Euclidean distance. This means forming k clusters, $C_j^{(s)}$, at step s, where

$$C_j^{(s)} = \{\mathbf{X}_i : ||\mathbf{X}_i - m_j^{(s)}||^2 \leq ||\mathbf{X}_i - m_r^{(s)}||^2, \text{for all } r = 1, \ldots, k\}. \tag{17}$$

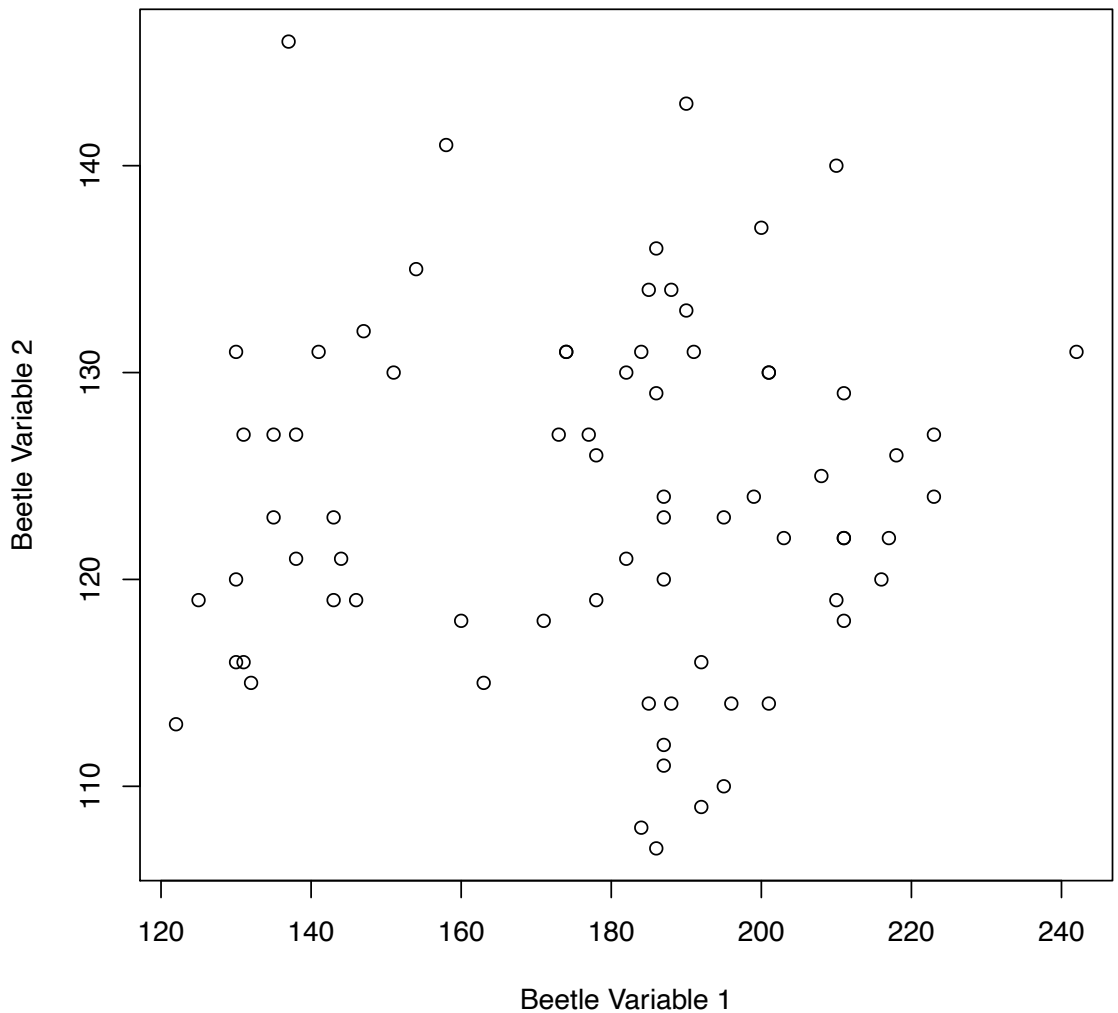
Update: calculate new means from observations in the new clusters:

$$m_j^{(s+1)} = \left|C_j^{(s)}\right|^{-1} \sum_{\mathbf{X}_i \in C_j^{(s)}} \mathbf{X}_i. \tag{18}$$

Then, iterate until clusters remain the same.

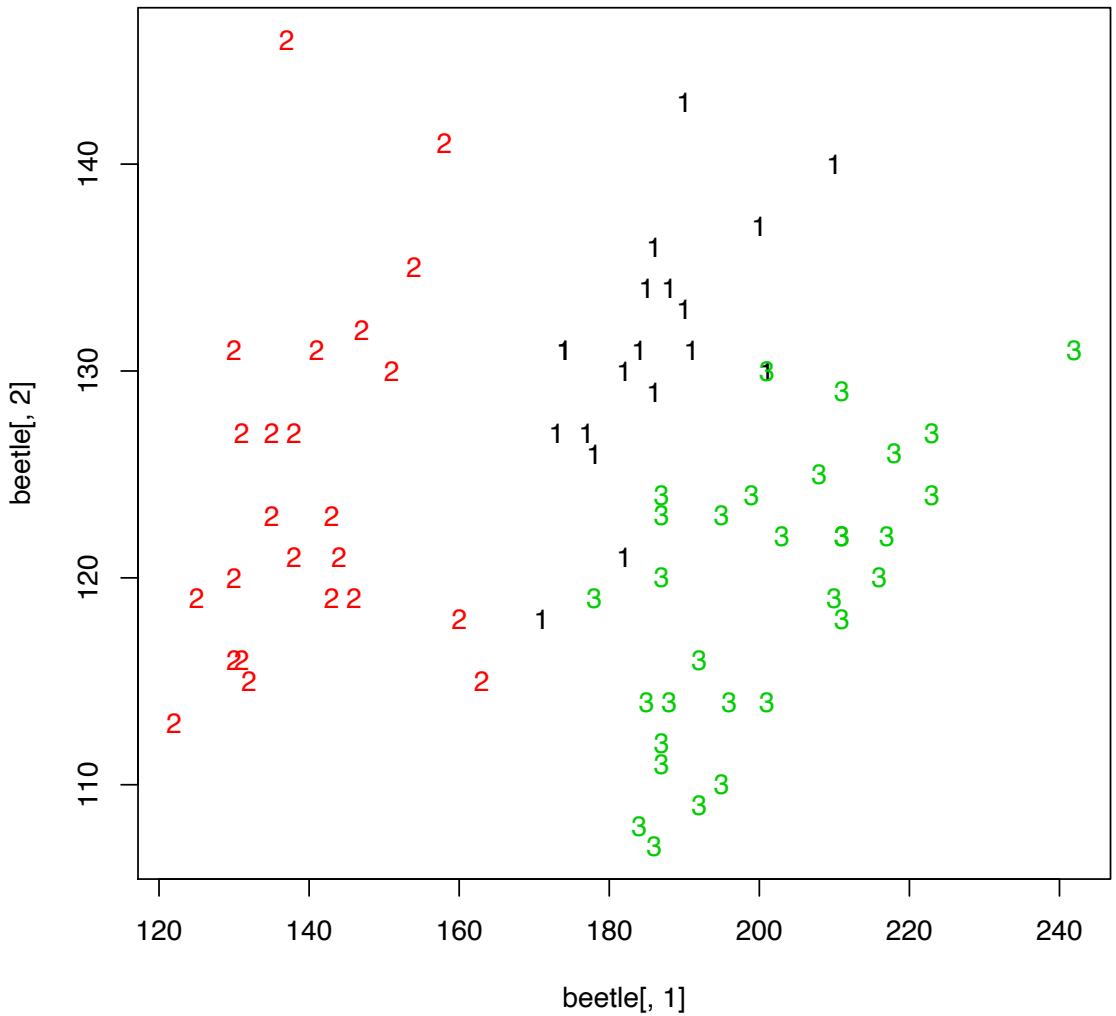
Six-dimensional beetle data ($n = 74$)

```
library("PP3")
data(beetle)
plot(beetle[,1], beetle[,2], xlab="Beetle Variable 1",
    ylab="Beetle Variable 2")
```

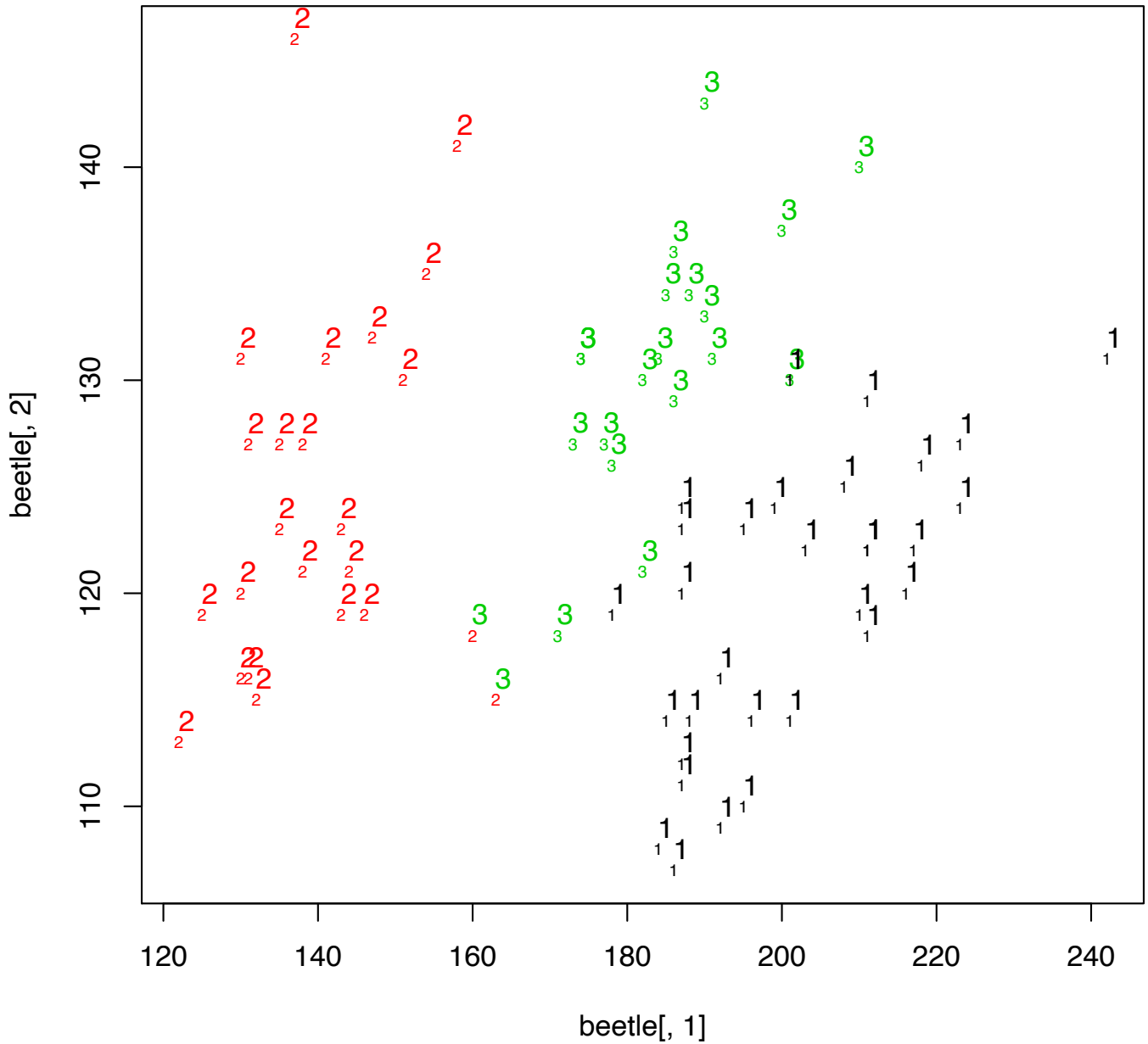


k-means example on beetle data

```
beet3 <- kmeans(x=beetle, centers=3, nstart=100)
plot(beetle[,1], beetle[,2], type="n")
text(beetle[,1], beetle[,2], col=beet3$cluster, lab=beet3$cluster)
```



Beetle k-means + superimposing true clustering



Self-organising maps (SOM, simple version)

Think of as constrained version of k-means clustering or type of artificial neural network. Apparently, method is modelled on how sensory information is handled in parts of the brain.

SOM with a two-dimensional rectangular grid of k prototypes.

Prototypes are akin to cluster means as being *representative* of the cluster values.

Often, prototype are parametrized according to an integer coordinate pair $\ell_j \in \mathcal{Q}_1 \times \mathcal{Q}_2$.

Where $\mathcal{Q}_1 = \{1, \dots q_1\}$ and similarly $\mathcal{Q}_2$ and $k = q_1 q_2$.

Here X, the configuration, is thought of as the *training* data.

Algorithm (basic)

The observations are processed sequentially $\mathbf{X}_i$.

We find the closest prototype to $\mathbf{X}_i$, call it m_j.

Find the closest neighbours, m_ℓ of m_j, move neighbours m_k, including the closest m_j toward $\mathbf{X}_i$ via the update:

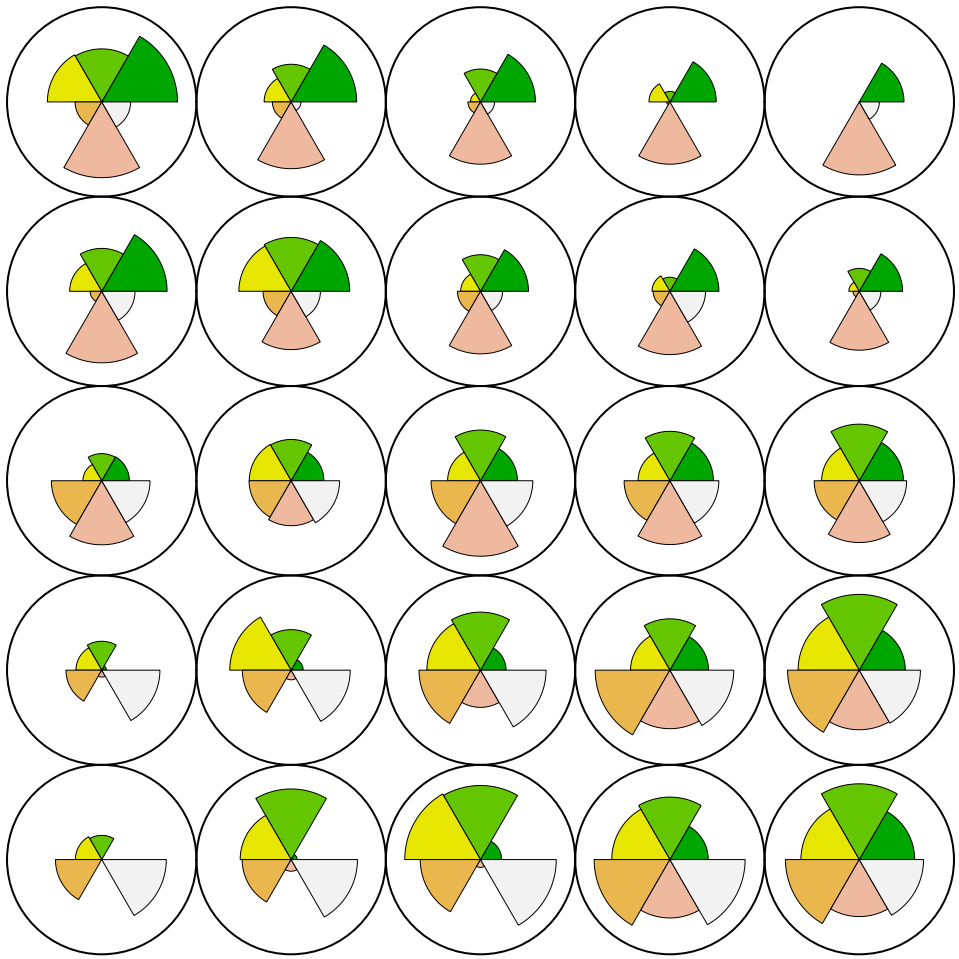
$$m_k \leftarrow m_k + \alpha(\mathbf{X}_i - m_k), \tag{19}$$

where α is called the *learning rate* and neighbours are determined by any points that are closer than a distance r — these parameters control the evolution of the algorithm.

The map then evolves to match the configuration, and then the data get projected down into the map into the "bins" defined by the prototypes.

Simple SOM example

```
library("kohonen")
plot(som(beetle, somgrid(5,5, "rectangular")), type="codes")
```



Summary

Scaling is a method for turning distances or dissimilarities into 'recovered' configurations.

It comes in two types: classical and ordinal.

k-means clustering and self-organising maps work on configurations, but use distance information to form groups and clusters.

The common link is distance/dissimilarity and the performance of any algorithm will depend on what dissimilarity you use and/or things like choice of K and, for SOM, α and r.

Elements of Statistical Learning: Lecture 10. Procrustes Analysis

Guy Nason[1]

Department of Mathematics
Imperial College

Procrustes

Procrustes comes from Greek myth and he is the son of Poseidon (god of the sea, storms, earthquakes and horses).

The myth tells of how Procrustes would invite travellers to spend the night on his special bed.

If the traveller was too tall, Procrustes would cut some of the traveller off, so that they would fit the bed.

If the traveller was too small, Procrustes would stretch the traveller on the bed until they fitted.

The traveller usually died. Ewww.

Procrustes was eventually killed by the Greek hero Theseus, who killed him on his own bed.

Procrustes Analysis

How do we compare scaling methods?

Or different configurations obtained from different data sets, or dissimilarities (e.g. Sicily on Eucidean or binary)?

Suppose we have two configurations X and Y, both described by $n \times K$ matrices.

Recall that our scaling reconstructions are unique up to position and orientation (and also scale, e.g. if we're using dissimilarities and there is no obvious natural scale provided).

We want to move Y to be as close as possible to X and then see how close the resemblance is.

Measures of Closeness and How to Make Them Closer

Given X, Y we measure their closeness using

$$G(X, Y) = \sum_{k=1}^{K} \sum_{i=1}^{n} (X_{i,k} - Y_{i,k})^2. \tag{1}$$

We want to minimise G under the following group actions: translation group, Euclidean group (rotation, reflections, fixed scale) and similarity group (similar shape).

It turns out that the best way to do this is sequentially

1. match first under translation
2. then under rotation/reflection
3. then under scale change.

Translation Step

Let $\bar{X}_k = n^{-1} \sum_{i=1}^{n} X_{i,k}$, $\bar{Y}_k = n^{-1} \sum_{i=1}^{n} Y_{i,k}$.
Write

$$
\begin{aligned}
G(X, Y) &= \sum_{i,k} (X_{i,k} - \bar{X}_k + \bar{X}_k - \bar{Y}_k + \bar{Y}_k - Y_{i,k})^2 && (2) \\
&= \sum_{i,k} (X_{i,k} - \bar{X}_k)^2 + \sum_{i,k} (Y_{i,k} - \bar{Y}_k)^2 && (3) \\
&\quad + \sum_{i,k} (\bar{X}_k - \bar{Y}_k)^2 + \text{two X terms} && (4) \\
&\quad + 2 \sum_{i,k} (X_{i,k} - \bar{X}_k)(Y_{i,k} - \bar{Y}_k). && (5)
\end{aligned}
$$

Term $\sum_{i,k} (X_{i,k} - \bar{X}_k)^2$ does not depend on Y.

Translation Step — 2

Term $\sum_{i,k}(Y_{i,k} - \bar{Y}_k)^2$ does not change if we shift Y by a fixed vector. The same is true for term (5).
The other two 'X terms' are of the form

$$\sum_{i,k}(X_{i,k} - \bar{X}_k)(\bar{X}_k - \bar{Y}_k) = \sum_{k}(\bar{X}_k - \bar{Y}_k)\sum_{i=1}^{n}(X_{i,k} - \bar{X}_k) \quad (6)$$

And $\sum_{i=1}^{n} X_{i,k} = n\bar{X}_k$, which is also the value of the second term. So (6) is zero and so is the other cross term.

So, to minimize $G(X, Y)$ we have to choose $\bar{Y}_k = \bar{X}_k$, i.e. match the centroids of the two configurations.

Rotation Step: notation reminder

Remember some basic things about inner products and norms.

Given two matrices A, B the *Frobenius inner product* is

$$< A, B >_F = \sum_{i,j} \bar{A}_{i,j} B_{i,j} = \mathrm{tr}(\bar{A}^T B), \tag{7}$$

however here we will only considered real-valued matrices so drop the complex conjugate (bar). The Frobenius norm is

$$||A||_F^2 = < A, A >_F \ . \tag{8}$$

Putting our problem into norm form

Suppose we have two configurations X and $A = YP$, where P is a rotation matrix.

Then

$$
\begin{aligned}
G(X, A) = \sum_{i,k} (X_{i,k} - A_{i,k})^2 &= \sum_{i,k} (X_{i,k} - A_{i,k})(X_{i,k} - A_{i,k}) && (9) \\
&= \operatorname{tr}\left\{ (X - A)^T (X - A) \right\} && (10) \\
&= ||X - A||_F^2. && (11)
\end{aligned}
$$

We thus want to minimise $||X - YP||_F^2$ over all orthogonal rotation matrices P to find the P^* the minimises it.

Solving the problem

We have

$$
\begin{aligned}
P^* &= \arg\min_P ||YP - X||_F^2 && (12)\\
&= \arg\min_P < YP - X, YP - X >_F && (13)\\
&= \arg\min_P < YP, YP >_F + < X, X >_F - 2 < YP, X >_F,
\end{aligned}
$$

and $< YP, YP >_F = \mathrm{tr}(P^T Y^T YP) = \mathrm{tr}(PP^T Y^T Y) = \mathrm{tr}(IY^T Y) = \mathrm{tr}(Y^T Y)$ and doesn't depend on P, by properties of the trace operator.

Hence, $P^* = \arg\max_P < P, Y^T X >_F$ (again, by properties of trace, if you like).

Solving the problem — 2

Let $U\Sigma V^T$ be the singular value decomposition of $Y^T X$. Then

$$\begin{aligned} P^* &= \arg\max_P < P, U\Sigma V^T >_F && (14) \\ &= \arg\max_P < U^T PV, \Sigma >_F && (15) \\ &= \arg\max_P < S, \Sigma >_F, && (16) \end{aligned}$$

where $S = U^T PV$ is an orthogonal matrix as U, P and V are.

Now $< S, \Sigma >_F = \sum_{i,j} S_{i,j}\Sigma_{i,j}$, but Σ is diagonal with non-negative entries,so $\Sigma_{i,j} = 0$ for $i \neq j$, so

$$< S, \Sigma >_F = \sum_{i,i} S_{i,i}\Sigma_{i,i}. \quad (17)$$

Solving the problem — 3

Since S is orthogonal all its entries are bounded in magnitude by 1. So, the biggest any $S_{i,i}$ can be is 1 and when all the diagonals are 1, all the other entries must be zero. So, to maximize (16) S must be the identity.

So, the maximising $S = I$ and this means $I = U^T PV$ and hence the maximising $P = UV^T$.

Procrustes Distance

The Procrustes distance is the minimised distance, when $P = P^*$.

I.e.

$$
\begin{aligned}
||YP^* - X||_F^2 &= < YP^*, YP^* >_F + < X, X >_F - 2 < YP^*, X >_F \\
&= < Y, Y >_F + < X, X >_F - 2 < P^*, Y^T X >_F \quad (18) \\
&= ||Y||_F^2 + ||X||_F^2 - 2 < P^*, U\Sigma V^T >_F \quad (19) \\
&= ||Y||_F^2 + ||X||_F^2 - 2 < UV^T, U\Sigma V^T >_F \quad (20) \\
&= ||Y||_F^2 + ||X||_F^2 - 2 < V^T, U^T U\Sigma V^T >_F \quad (21) \\
&= ||Y||_F^2 + ||X||_F^2 - 2 < V^T V, \Sigma >_F \quad (22) \\
&= ||Y||_F^2 + ||X||_F^2 - 2 < I, \Sigma >_F, \quad (23)
\end{aligned}
$$

and $< I, \Sigma >= \text{tr}(I^T \Sigma) = \text{tr}(\Sigma) = \sum_i \Sigma_{i,i}$.

Explore $P = (Y^T X)(X^T Y Y^T X)^{-1/2}$.

Procrustes Scale Change

We minimize

$$\begin{aligned} G(X, \alpha Y) &= \sum_{i,k} (X_{i,k} - \alpha Y_{i,k})^2 && (24) \\ &= \sum_{i,k} X_{i,k}^2 - 2\alpha \sum_{i,k} X_{i,k} Y_{i,k} + \alpha^2 \sum_{i,k} Y_{i,k}^2 && (25) \\ &= a\alpha^2 + b\alpha + c, && (26) \end{aligned}$$

where $a = \sum_{i,k} Y_{i,k}^2$, $b = -2 \sum_{i,k} X_{i,k} Y_{i,k}$, $c = \sum_{i,k} X_{i,k}^2$.

Minimise with respect to α, differentiate and set to zero

$$\frac{\partial G}{\partial \alpha} G(X, \alpha Y) = 2a\alpha + b = 0, \quad (27)$$

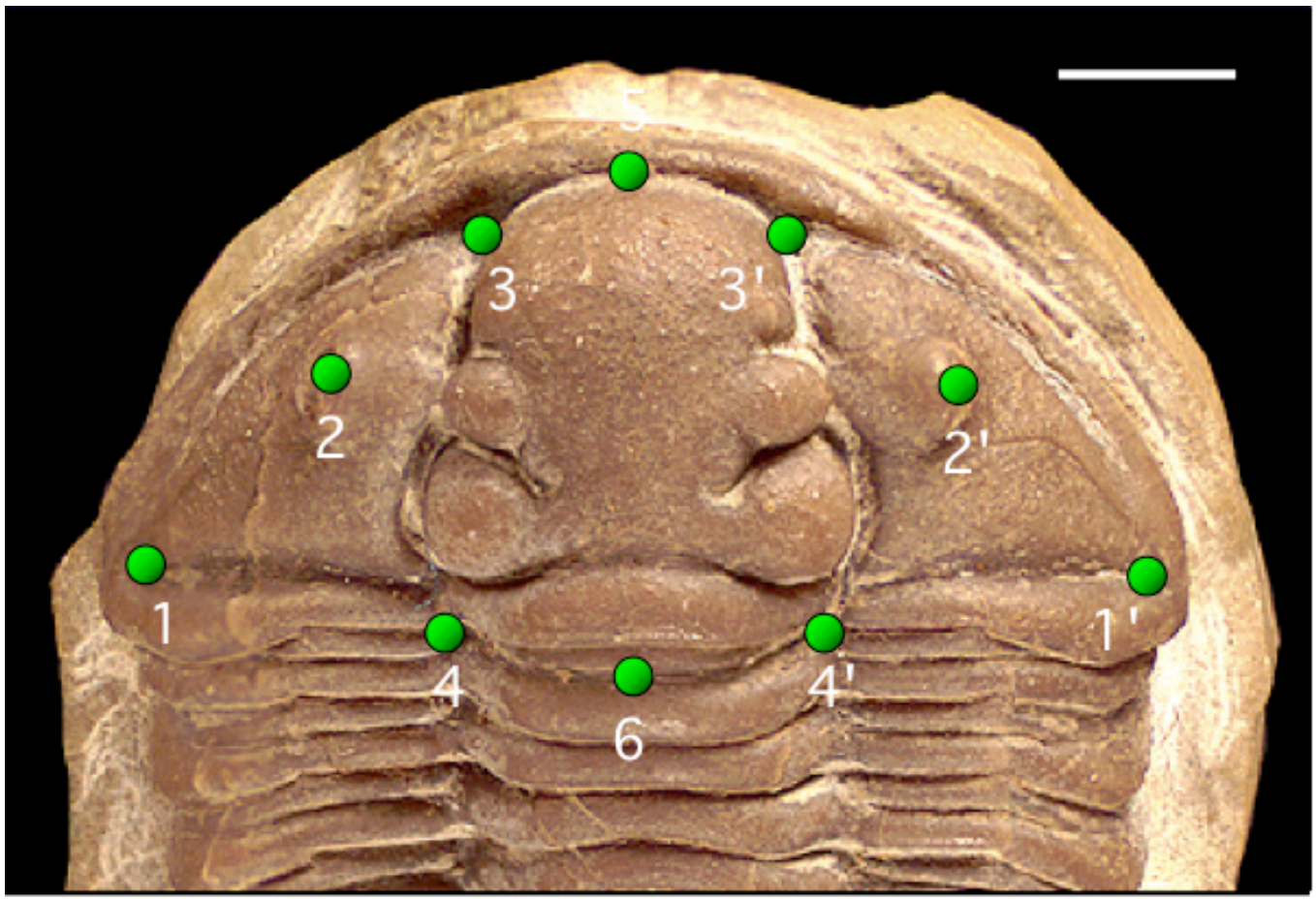
and $a > 0$, so we have minimum at $\alpha = -b/2a$, so set $\alpha^* = \sum_{i,k} X_{i,k} Y_{i,k} / \sum_{i,k} Y_{i,k}^2$.

Purpose of Procrustes analysis

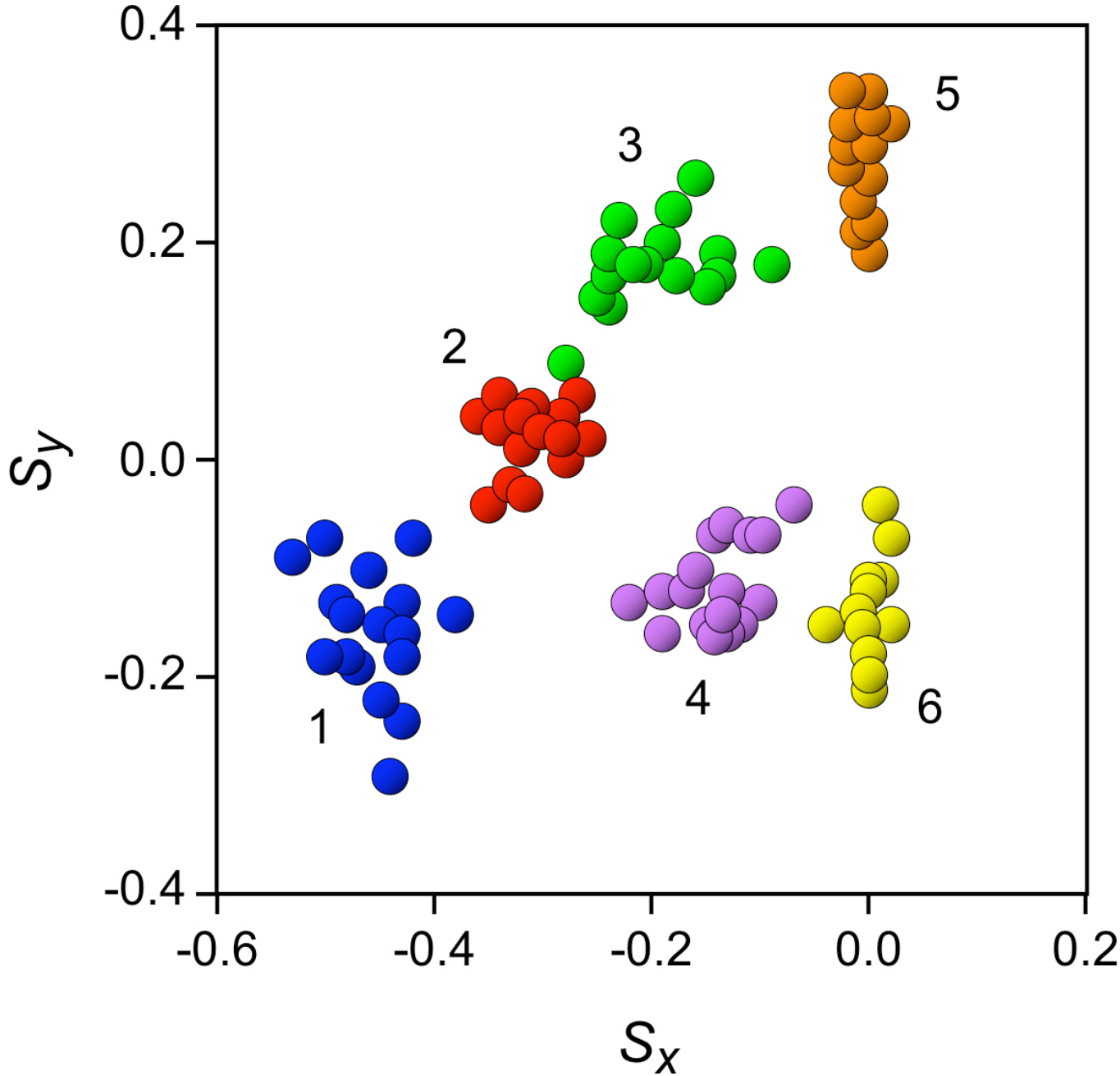
1. assessing the performance of scaling methods (e.g. finding the distribution of G_{min} when the distances are perturbed by normal distribution. So, that we can assess the uncertainty around scaling solutions, or rather the Procrustes distance)
2. comparing different configurations produced by scaling (e.g. to compare scaling methods)
3. compare the results of different testers. E.g. if a number of people taste wine, each person's taste scores can be turned into their individual distance matrix (one per person). Each one of these is turned into a scaling configuration and Procrustes can be used to match them. See, e.g. Filipe-Ribeiro, L. *et al.* (2017) Data on changes in red wine phenolic compounds . . . , *Data in Brief*, **12**,188–202. (see Generalised version, below).
4. matching natural specimens to identify and classify species.

Procrustes in Paleo-biology

https://www.palass.org/publications/newsletter/palaeomath-101/palaeomath-part-16-who-procrustes-and-what-has-he-done-my-data



Procrustes in Paleo-biology



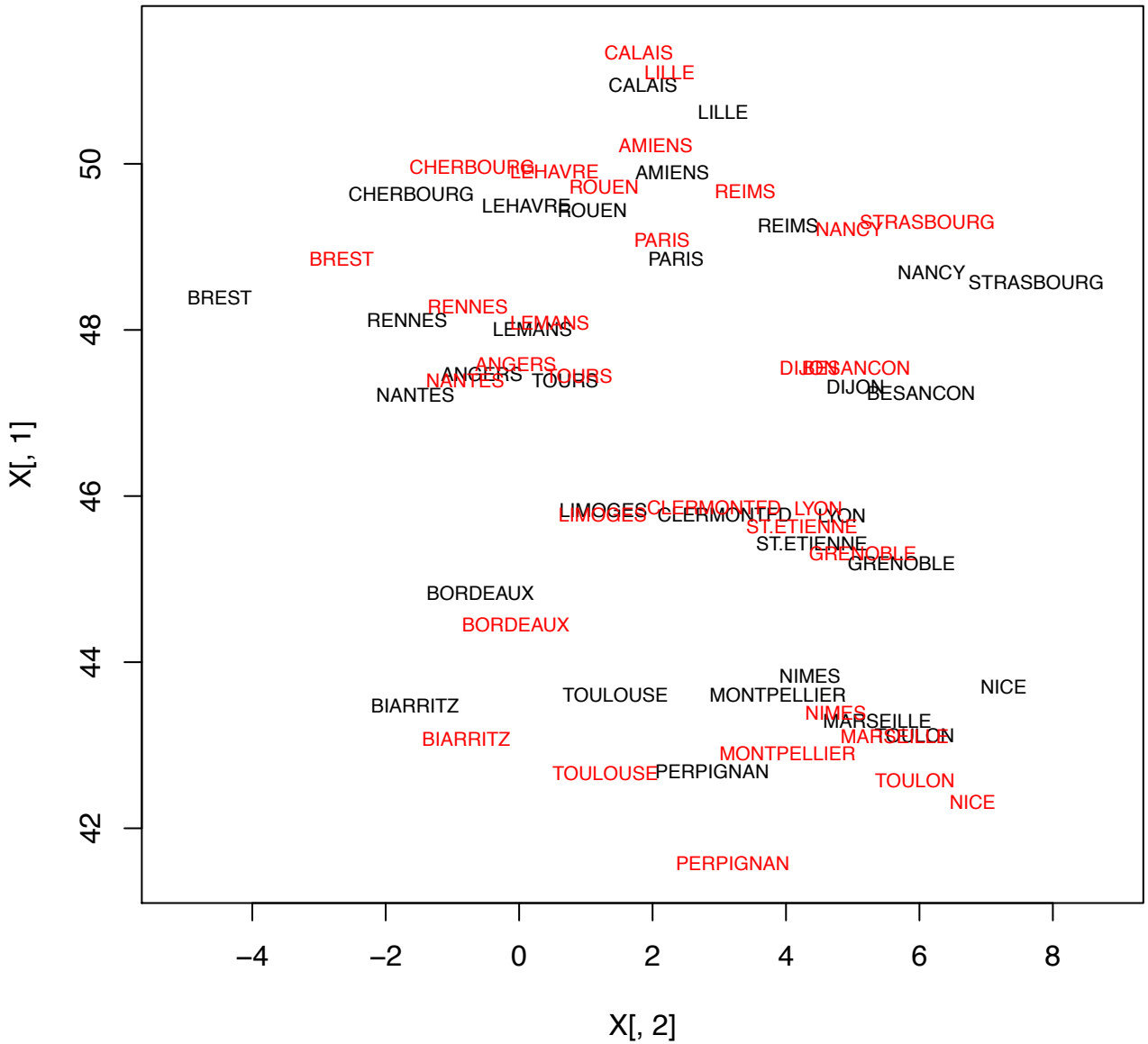
Procrustes in R

We use the `Procrustes()` function in the `smacof` package.

In the following, object `frcity` has the latitude and longitude of the French cities in columns two and three.

```
library("smacof")
X <- cbind(as.numeric(frcity[,2]), as.numeric(frcity[,3]))
Y <- cbind(frscalesoln$points[,1], frscalesoln$points[,2])
fr.procr <- Procrustes(X=X, Y=Y)

frcitynames <- frcity[,1]

plot(X[,2], X[,1], type="n", xlim=c(-5.1, 8.8),
    ylim=c(41.5,51.5))
text(X[,2], X[,1], lab=frcitynames, cex=0.7)
text(fr.procr$Yhat[,2], fr.procr$Yhat[,1], lab=frcitynames,
    cex=0.7, col=2)
```

Procustes of scaling solution to true locations



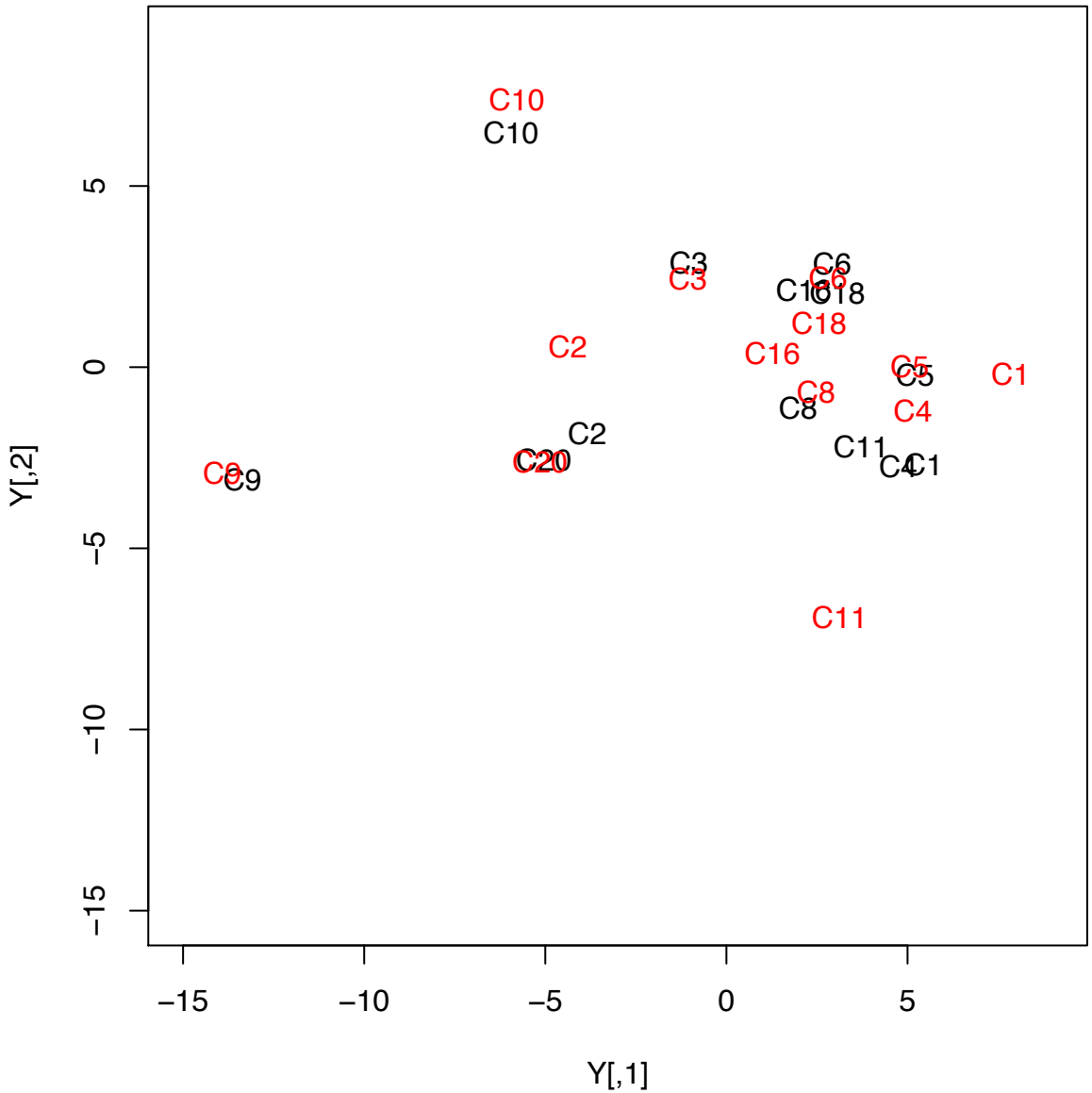
Notes: Procrustes match of `cmdscale` solution to truth

Some towns are far away from their true positions, e.g. *Brest, Perpignan, Strasbourg, Toulouse*.
These are further from the centre and in, perhaps, slightly harder to get to locations with roads that have to cope with trickier natural features (hills, mountains, lakes).

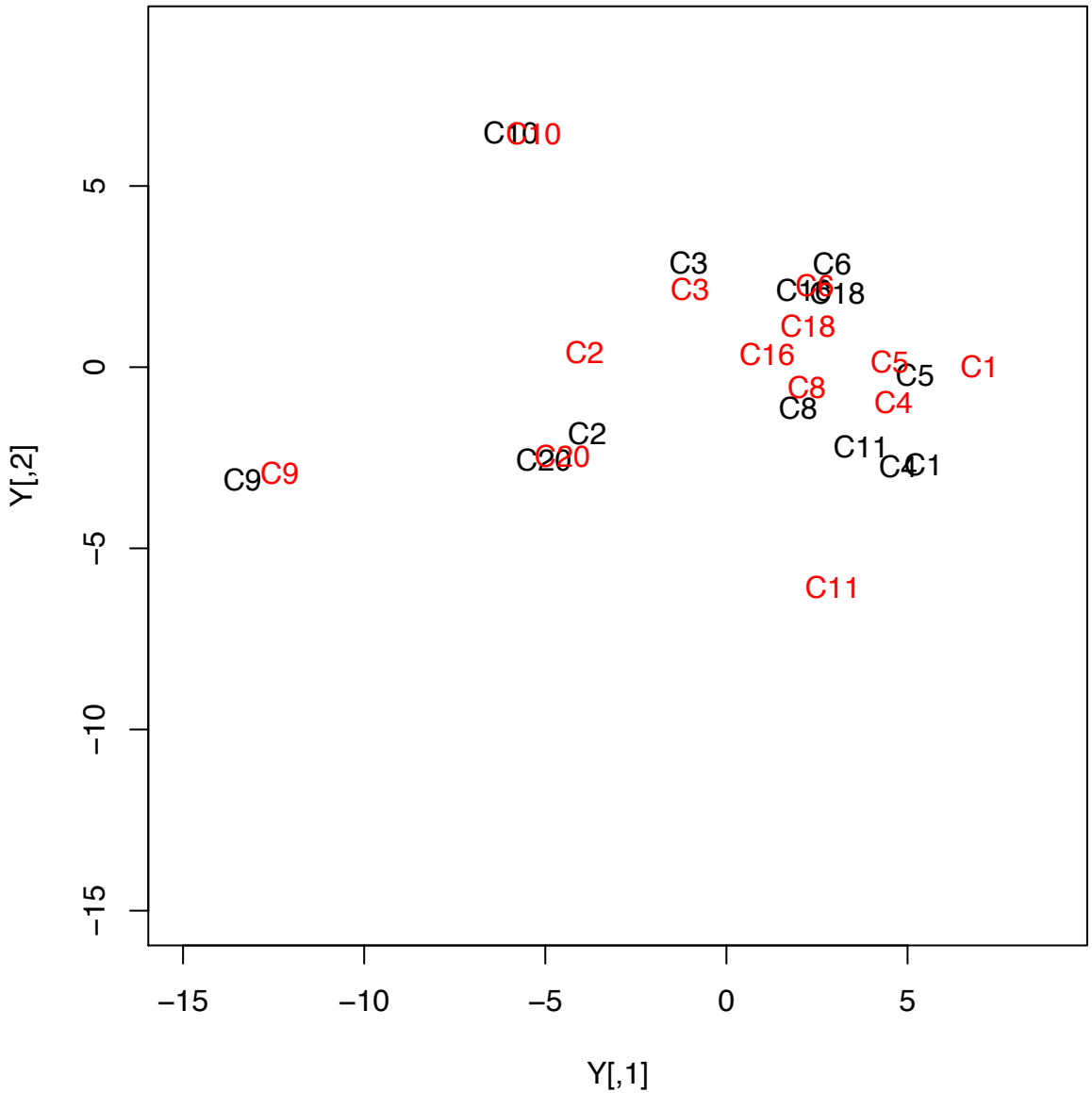
Cities nearer the middle of the country seem to be fairly well-placed, e.g. *Limoges, Tours, Le Mans, Angers*.

The value of the return from the `Procrustes()` function gives the following extra useful information. The optimum scale factor $\alpha = 0.009$. The optimum rotation matrix $\begin{pmatrix} -0.848 & -0.529 \\ -0.529 & -0.848 \end{pmatrix}$ and the optimum translation vector $(46.774, 2.472)$ — the latter is the centroid of the latitudes and longitudes of the 32 cities.

Sicily: classical & ordinal scaling (p 12 from Lecture 9)



Sicily: same configs but aligned using Procrustes



Generalized Procrustes Analysis

Sometimes we obtain several configurations and we wish to find an average of them.

E.g. we may have many reference points on several trilobites and we wish to work out the average of them.

Generalised Procrustes Analysis takes a set of configurations $X_1, \ldots, X_L$, all of dimension $n \times p$ and returns a configuration average of the same order.

Define $S_{\{X_\ell\}_{\ell=1}^L}(\{R_\ell\}_{\ell=1}^L, M) = \sum_{\ell=1}^L ||X_\ell R_\ell - M||_F^2$,we wish to solve

$$\min_{\{R_\ell\}_{\ell=1}^L, M} S_{\{X_\ell\}_{\ell=1}^L}(\{R_\ell\}_{\ell=1}^L, M), \tag{28}$$

i.e. find the M that is closest in average squared Procrustes distance to all the shapes.

Generalised Procrustes Analysis algorithm

The generalised Procrustes average can be obtained by

1. choose an arbitrary configuration, X_i, say, and set $M = X_i$;
2. use Procrustes analysis to match all configurations (apart from X_i if this is the first loop) to M — call the matched configurations $Y_1, \ldots, Y_L$ (and $Y_i = X_i$ on the first loop) and the matching matrices $\{R_\ell\}_{\ell=1}^L$.
3. compute the mean shape of all the matched configurations: $M \leftarrow L^{-1} \sum_{i=1}^L Y_i$.
4. compute $S_{\{X_\ell\}_{\ell=1}^L}(\{R_\ell\}_{\ell=1}^L, M)$ and see if it has converged, if not, then go to 2.

Generalised Procrustes Analysis: Suresh signature

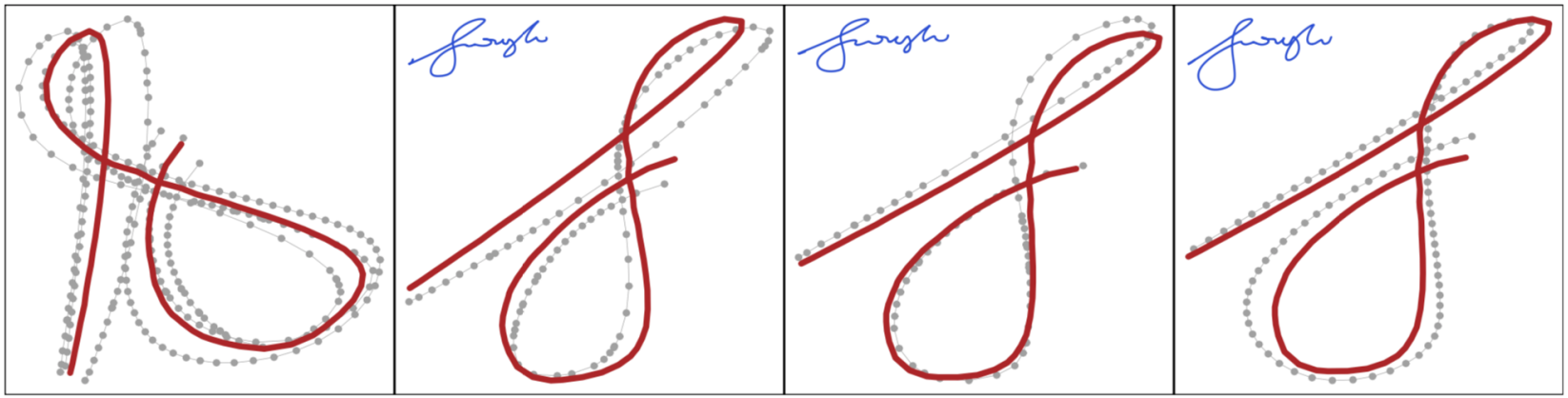
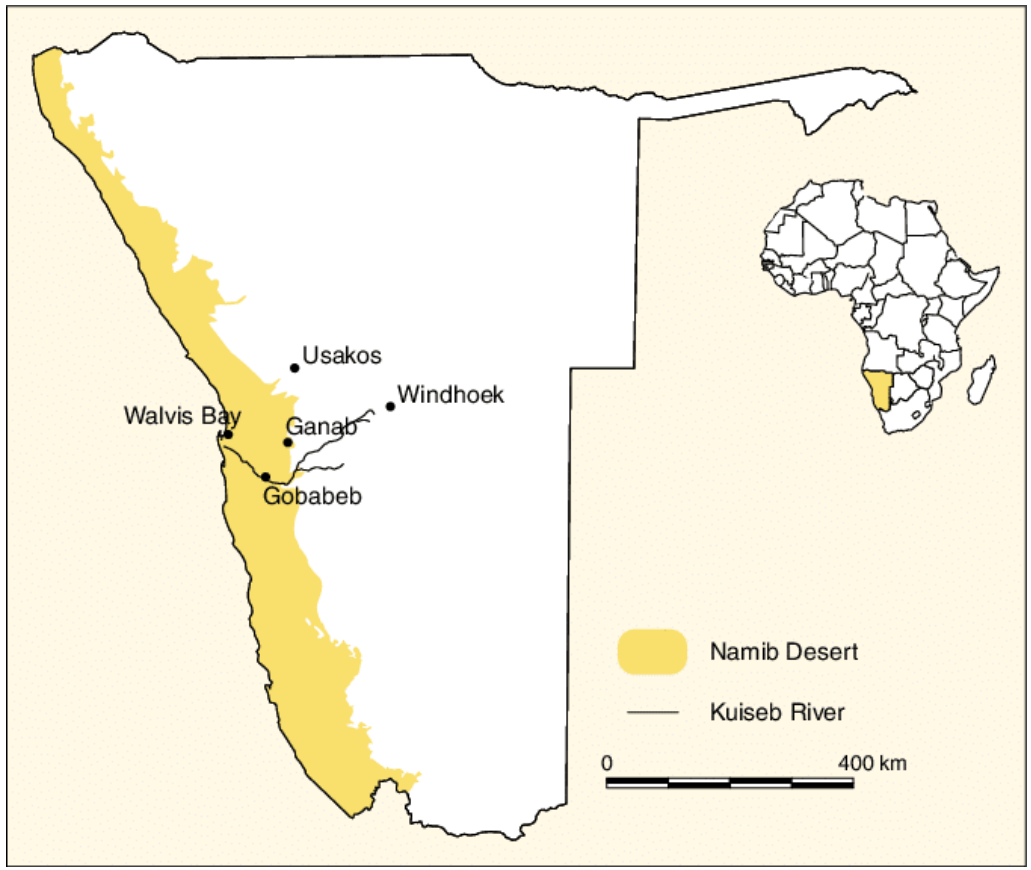


FIGURE 14.26. *The Procrustes average of three versions of the leading S in Suresh's signatures. The left panel shows the* preshape *average, with each of the shapes* $\mathbf{X}'_\ell$ *in preshape space superimposed. The right three panels map the preshape* $\mathbf{M}$ *separately to match each of the original S's.*

from Hastie, Tibshirani and Friedman book, page 541.

Using Generalised Procrustes Analysis in R

Data from the Namib desert:



(image from `conservationinstitute.org`).

Using Generalised Procrustes Analysis in R — 2

Using data from the `provenance` package in `R`: 14 sand samples from Namib Sand Sea and two from Orange River (southern Africa).

```
library("provenance")
data(Namib)
Namib$DZ[[3]]$T1    # One location T1
  [1]  268.0 1051.2 1741.0  522.2 1152.5 1288.8  558.4 1072.4  530.7  475.5
 [11]  641.3 1084.0  531.4  848.1  768.4  925.5  535.0  765.7 1196.4 1326.6
 [21] 1077.0  518.4 1077.0  566.8 1016.3  532.2  513.9 1631.4 1021.2 1222.9
 [31]  560.1  590.4  549.8 1966.0 1056.4 1066.4 1052.1  891.7  521.0 1268.8
 [41]  960.2 3148.4  558.4 1061.2  510.1 1097.6  673.8  509.2 1650.3 1038.6
 [51]  502.3 1231.0 1049.6 1048.4 1891.0  521.4 1157.4  921.9  986.5 1106.3
 [61]  850.9 1097.1  283.2 2731.0 1094.1  978.5 1785.6 1098.3 1069.8  581.9
 [71]  506.9 1066.8  526.1  537.8  477.9 1145.1 2128.8  512.4 1192.5  590.5
 [81] 1099.5 1836.4  662.9  997.0 1881.5  528.4  898.8  637.5  260.0 1267.9
 [91]  553.2 2759.3  276.2  525.7 1166.5 1046.0 1305.5 1097.6  930.5  542.6
[101]  409.4  703.6  521.0 1906.7 1002.5 1215.5
```

Using Generalised Procrustes Analysis in R — 3

Heavy minerals at locations, N14, N13, etc, minerals zirconium, tourmaline, rutile, etc.

```
> Namib$HM[[2]]
    zr tm rt TiOx sph ap ep gt st and ky sil amp cpx opx
N14  0  0  1    1   0  1  5  8  0   0  0   0  18 162   6
N13  2  1  2    4   3  4 47 13  0   0  0   0  20 121   4
N12  3  0  0    1   0  1  5  3  0   0  0   0   8 184   5
N11  1  0  2    0   0  0  7 27  1   0  0   0  11 158   1
N10  2  0  0    0   4  0 16 28  2   0  1   0  34 126   0
N9  10  2  0    0   9  0 26 80  2   0  0   0  18  56   0
N8   5  0  0    0   2  0 23 46  1   0  0   1  11 113   0
N7   4  0  1    0   0  0 12 19  0   1  0   0  10 155   1
N6   3  1  2    1   2  0 11 12  1   0  0   0  31 139   3
N5   1  2  2    5  12  2 91 43  5   2  0   0  32  10   1
N4   0  2  4    1   4  0 59 54  0   1  0   0  19  54   2
N3   1  0  0    1   1  0  9 21  1   0  0   0  10 157   1
N2   0  0  1    0   3  1 11 18  1   0  0   1   3 170   1
N1   1  1  0    0   0  1 10 24  2   0  0   0   1 163   1
T8   0  1  0    0   2  0  7  1  0   0  0   0  25 162   2
T13  2  0  1    0   0  2 11  1  0   0  0   0  24 154   4
```

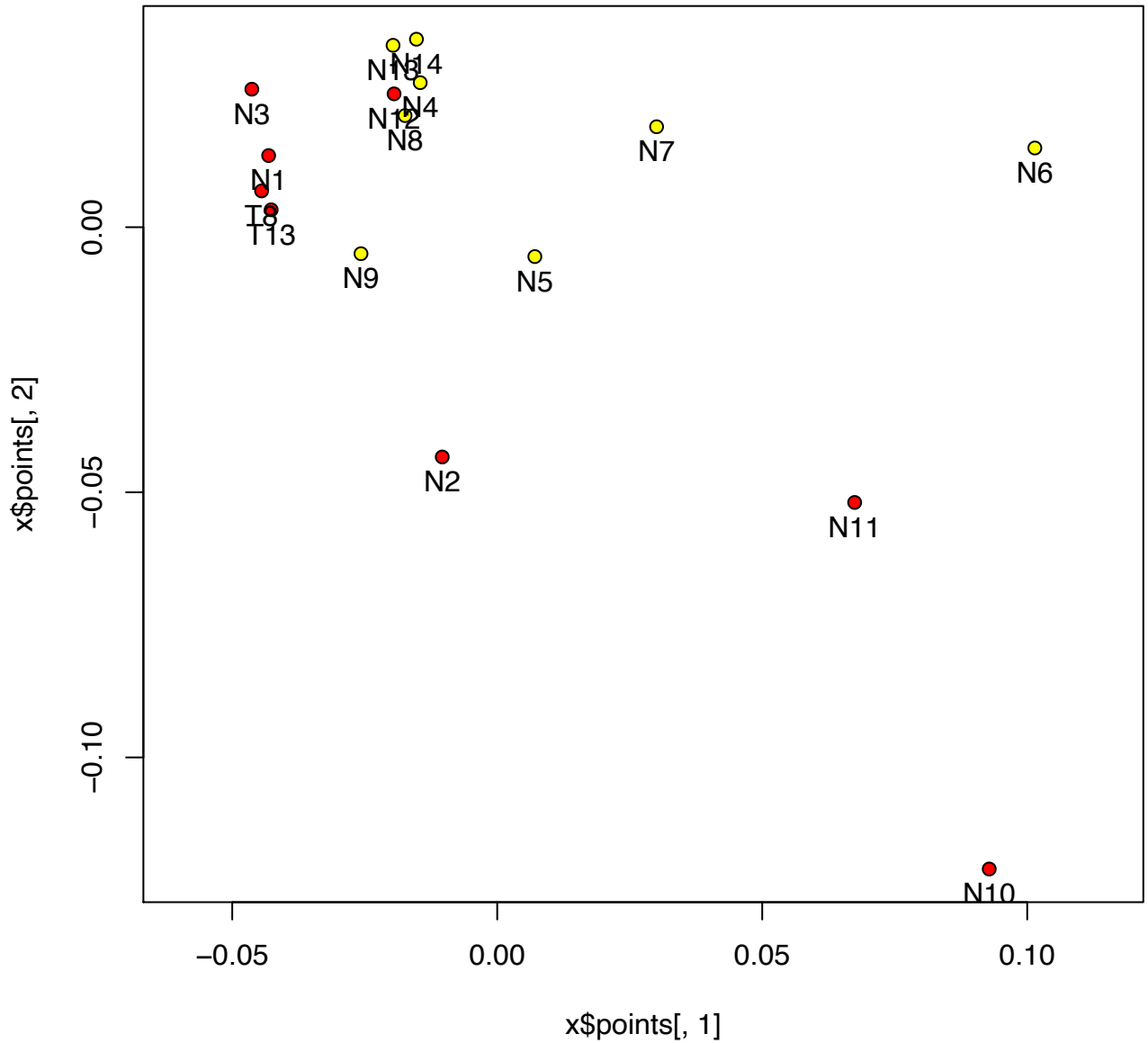
Using Generalised Procrustes Analysis in R — 4

Configurations formed from these quite different data sets via distances.

Then matched using Procrustes.

```
library("provenance")
data(Namib)
  GPA <- procrustes(Namib$DZ,Namib$HM)
coast <- c('N1','N2','N3','N10','N11','N12','T8','T13')
snames <- names(Namib$DZ)
bgcol <- rep('yellow',length(snames))
bgcol[which(snames %in% coast)] <- 'red'
plot(GPA,pch=21,bg=bgcol)
```

Using Generalised Procrustes Analysis in R — 5



Summary

In this lecture we covered:

Procrustes formulation

Three steps: translation, rotation and scaling and how to do it

Procrustes in R with `smacof` package on French road distances and Sicily data

Generalised Procrustes Analysis problem specification

Examples of Generalised Procrustes Analysis Suresh's signature and the Namib desert data.

Elements of Statistical Learning: Lecture 11. Basis Expansions (Splines)

Guy Nason[1]

Department of Mathematics
Imperial College

Diverging from linear functions

So far, in regression, we've looked at linear representations.

I.e. the function $f(X) = \mathbb{E}(Y|X)$ is a linear function of X.

In the simple linear regression, it was

$$f(X) = \mathbb{E}(Y|X) = a + bX. \tag{1}$$

Reasons why linear can be good:

- the data, and diagnostics, indicate that linear is good fit;
- you don't have much data, e.g. n small and you can't do much more complicated;
- you need something simple, either to communicate or automate, and the fit is tolerable for the purposes you need.

Otherwise, if the data suggest the function is bendy or has discontinuities, we will need something else, more flexible.

New idea for flexibility + better fit

Idea is we replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models involving these new inputs.

As ESL denote by $h_m(X) : \mathbb{R}^p \to \mathbb{R}$ the mth transformation of X, $m = 1, \ldots, M$.Then model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{2}$$

which is a *linear basis expansion* in X.

The advantage of this approach is that, once the $\{h_m\}$ have been fixed, the models are still linear in $\{\beta_m\}$, so we can use all the fitting tools from before.

Some examples of useful and popular h_ms

$h_m(X) = X_m$, $m = 1, \ldots, p$ is the original linear model.

$h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ creates additional inputs that model polynomial terms, which can be used to approximate to higher orders. However, note that the number of variables grows exponentially in the degree of the polynomial. A full quadratic model of p variables requires $\mathcal{O}(p^2)$ square and cross-product terms, or, more generally, $\mathcal{O}(p^d)$ for a degree d polynomial.

$h_m(X) = \log(X_j), \sqrt{X_j}, X_j^{-1}, \sin(X_j), \ldots$. Functions can involve more than one input, e.g. $h_m(X) = ||X||$.We already used this in the previous lecture.

$h_m(X) = \mathbb{I}(L_m \leq X_k \leq U_m)$, an indicator for the region $[L_m, U_m]$, results in a piecewise constant model.

Why are *global* polynomials not very useful

Getting global polynomials to fit parts of the data can result in unwieldy behaviour in other parts. It's hard to get it all right.

Experience has shown that it is better to deal with 'chunks' of data 'reasonably disconnected' to other chunks.

We'll look at families of *piecewise polynomials*, *splines* and *wavelets*.

However, these families tend to have *MANY* flexible members, far too many to fit all of them.

The member sets are often called *dictionaries*. Or dictionaries can be sets of bases.

Complexity Control

We need some way of controlling the number/style of our model.

This is called *controlling the complexity*. Three main types

Restriction methods: where we limit the class of functions *a priori*. For example, we could insist on additive models of the form

$$f(X) = \sum_{j=1}^{p} f_j(X_j) \tag{3}$$

$$= \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{j,m} h_{j,m}(X_j). \tag{4}$$

The size of the model is limited by the M_j for component $j = 1, \ldots, p$.

Complexity Control — 2

Selection methods: which continually look at the dictionary and put in members (basis functions) that improve the fit or remove those which are not contributing. For example, variable selection methods such as forward stepwise.

Regularization methods: where the whole dictionary is included, but the coefficients are restricted - e.g. ridge regression or the lasso.

We'll assume one dimension from now on, but more dimensions/variables can be handled.

Piecewise polynomial

A piecewise constant polynomial might be

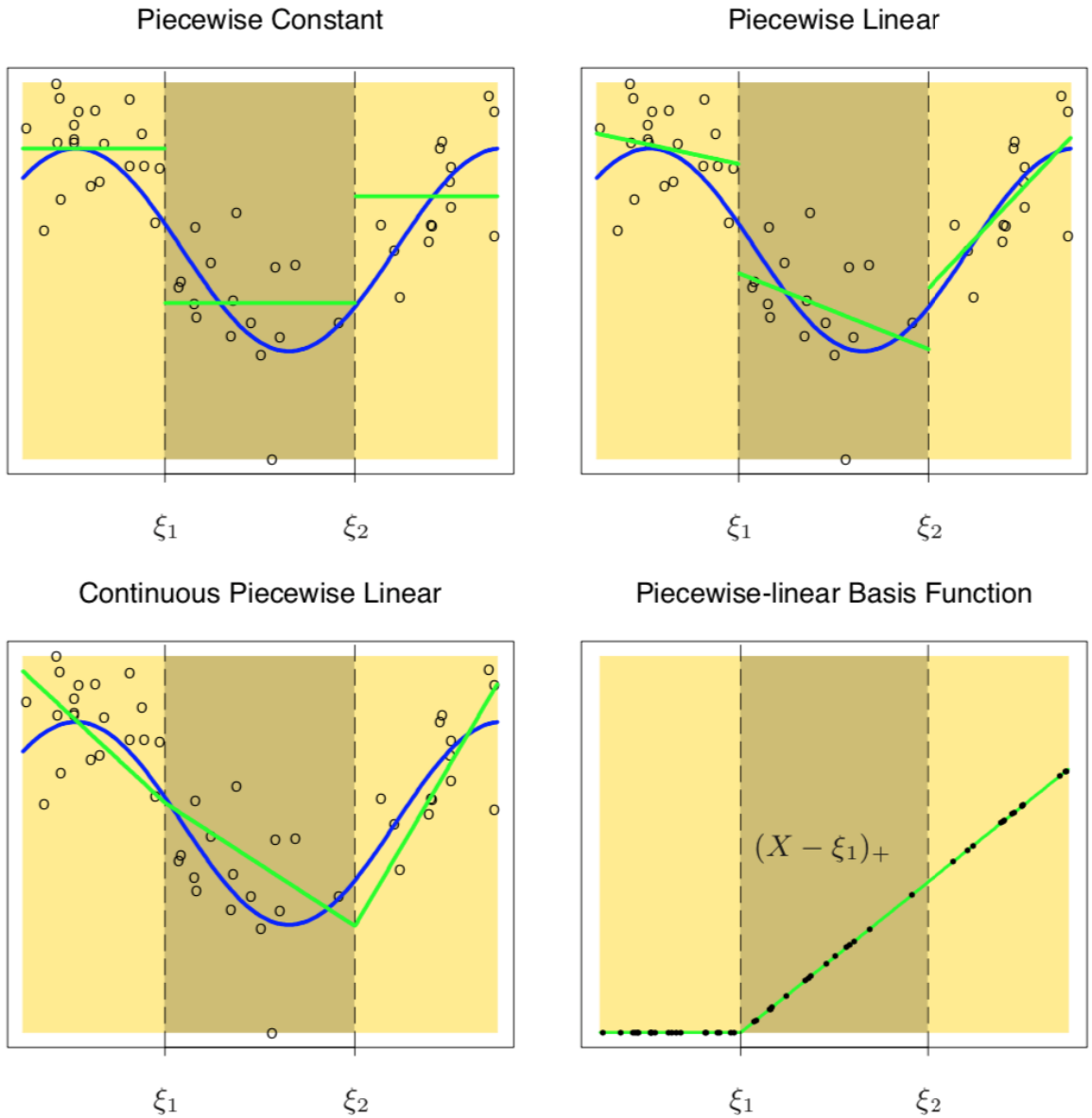
$$h_1(X) = \mathbb{I}(X < \xi_1), \quad h_2(X) = \mathbb{I}(\xi_1 \leq X < \xi_2), \quad h_3(X) = \mathbb{I}(\xi_2 \leq X), \tag{5}$$

since they're constant over disjoint regions, the best least squares fit assigns the mean over each region to the coefficient.

E.g. if $f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$ then $\hat{\beta} = \bar{Y}_m$, where $\bar{Y}_m$ is the mean of the Y values in the mth region.

This fit can be seen in the top-left panel of the figure on the next page

Piecewise polynomial



Piecewise linear

The top-right panel displays a piecewise linear fit. We need three additional basis functions $h_{m+3}(X) = h_m(X)X$, $m = 1, 2, 3$.

So, the polynomial on the first part will be $\beta_1 + x\beta_4$,and on the second one $\beta_2 + x\beta_5$.

Often, we prefer the bottom right panel, which is also piecewise linear, *but* it is forced to be continuous at the two *knots* ξ_1, ξ_2.

The constraint forces a kind of smoothness or, at least, regularity onto the fitted curve.

This means that $f(\xi_1^-) = f(\xi_1^+) \implies \beta_1 + \xi_1\beta_4 = \beta_2 + \xi_1\beta_5$

And have to have similar constraint at ξ_2.

Continuous piecewise linear

For the continuous piecewise linear we start with six free parameters $\beta_1, \ldots, \beta_6$, or six degrees of freedom.

The two constraints mean we lose two degrees of freedom. So, the piecewise continuous linear system has $6 - 2 = 4$ degrees of freedom.

Alternatively, we can build the constraints directly into the basis functions, e.g.

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = (X - \xi_1)_+, \quad h_4(X) = (X - \xi_2)_+, \tag{6}$$

where t_+ is the positive part of t.

E.g., so having h_4 means that it only effects the linear bit on the last region, but does not effect the earlier regions.

Piecewise cubic polynomials

Similar to piecewise linear, but we have a cubic on each piece. Where we have continuity and continuity of first and second derivatives the total function is called a *cubic spline*.

A basis represents any of these (on these regions):

$$h_1(X) = 1, \quad h_3(X) = X^2, \qquad h_5(X) = (X - \xi_1)^3_+, \tag{7}$$

$$h_2(X) = X, \quad h_4(X) = X^3, \qquad h_6(X) = (X - \xi_2)^3_+. \tag{8}$$

Why six?

Each cubic has four parameters $a + bx + cx^2 + dx^3$ and there are three regions: so $4 \times 3 = 12$ in total.

At each knot: three constraints: for continuity, 1st and 2nd deriv.

There are two knots, so total constraints takes away $2 \times 3 = 6$ degrees of freedom. So, total number of degrees of freedom is $12 - 6 = 6$.

Remarks on piecewise splines

Cubic splines are (meant to be) the lowest order spline for which the knot-discontinuity is not visible to the human eye

With fixed knots, these are known as *regression splines*.

You need to select the order of the spline (e.g. cubic), the number of knots and their placement.

There are many alternative bases doing the same thing.

A *natural cubic spline* is a cubic spline where the pieces at the two end regions are linear, not cubic. This releases four degrees of freedom (for the quad. and cubic bits, at each end) to put more knots in the middle.

Natural Cubic Spline basis

A natural cubic spline with K knots is represented by K basis functions, given by (see ELS-(5.4))

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{K-1}(X), \quad (9)$$

where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}. \quad (10)$$

for $k = 1, \ldots, K - 2$.

Read §5.1, §5.2, §5.3 of ELS.

Smoothing Splines

Widely used. Easy to use Nice functions in R. Don't have to select knots — uses maximal set. Complexity is controlled by regularization.

Given a set of data $(x_i, y_i)_{i=1}^n$.

Consider the following. Let $\mathcal{F}$ be the set of all functions with two continuous derivatives.

Find $f \in \mathcal{F}$ that minimises the following penalised residual sum of squares:

$$\text{RSS}(f, \lambda) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt, \qquad (11)$$

where λ is a fixed *smoothing parameter*.

Smoothing splines (explanation)

The first term $\sum_{i=1}^{n}\{y_i - f(x_i)\}^2$ measures the *fidelity* or goodness of fit of the model, $f(x_i)$ to the data y_i.

The second term, $\int\{f''(t)\}^2 dt$ penalizes curvature.

E.g. if $f(x)$ is constant or linear then the penalty is zero.

The λ value controls the tradeoff between the two aspects.

Two special cases are:

$\lambda = 0$: $f \in \mathcal{F}$ can be any function that interpolates the data

$\lambda = \infty$: the simple least squares line fit, as no second derivative/curvature is allowed.

We can models f from very rough to very smooth by changing λ.

Motorcycle Data

Data set called `mcycle` in the `MASS` library.

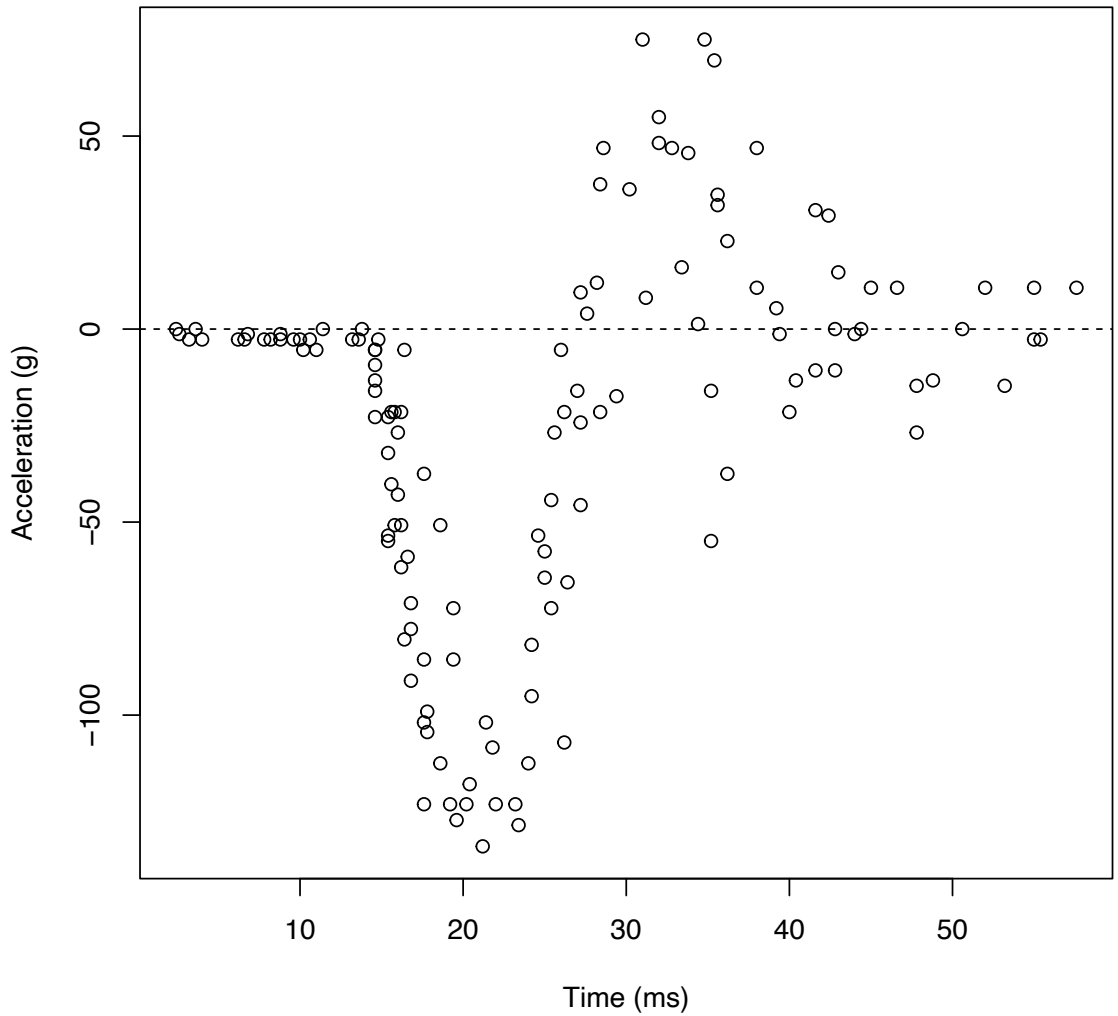
Gives a series of 133 measurements of head acceleration in a simulated motorcycle accident, used to test crash helmets.

Is a data frame with two variables:

- `times` time in milliseconds after impact
- `accel`, in g, of the head

From: Silverman, B.W. (1985) Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society*, Series B, **47**, 1–52.

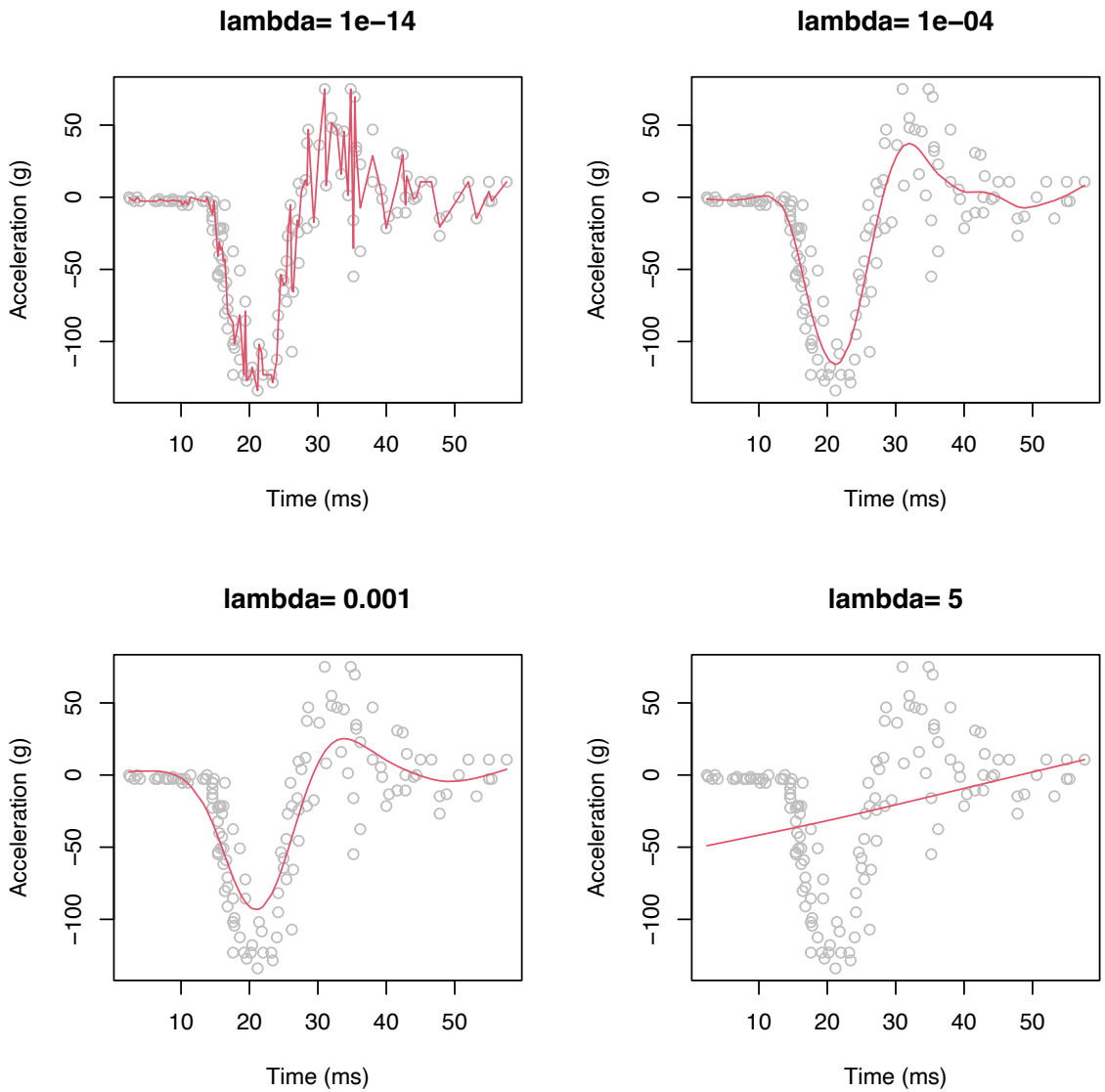
Motorcycle Data



Code for 4 smooth.spline fits to motorcycle data

```
library("MASS")
data(mcycle)

oldpar <- par(mfrow=c(2,2))  # Set up 2x2 plots

lvec <- c(1e-14, 0.0001, 0.001, 5) # Set up lambda

# Do four plots
for(i in 1:4) {
    plot(mcycle$times, mcycle$accel, xlab="Time (ms)",
        ylab="Acceleration (g)", col="grey",
        main=paste("lambda=", lvec[i]))
    ss <- smooth.spline(x=mcycle$times, y=mcycle$accel,
        lambda=lvec[i], all.knots=TRUE)
    lines(ss$x, ss$y, col=2)
}
par(oldpar) # Restore plot settings
```

Four `smooth.spline` fits to the motorcycle data



Interpolating for $\lambda = 0$?

In the top-left plot of the previous page, the fit does not seem to interpolate.

Why not?

This is because there is sometimes more than one y_i for each x_i.

For example, both $(55.0, -2.7)$ and $(55.0, 10.7)$ exist

The spline then interpolates the mean $(-2.7 + 10.7)/2 = 8/2 = 4$.

The argument *all.knots* = *TRUE* uses all of the data points as knots.

Results about the solution

The space $\mathcal{F}$ is infinite dimensional (actually a Sobolev space).

It can be shown that (11) has an explicit finite-dimensional unique minimizer, which is a natural cubic spline, with knots at the unique values of x_i

See, e.g.

`https://www.cs.utexas.edu/users/cline/CS383D/ReinschSmoothing.pdf`

or the paper by Reinsch therein, which uses calculus of variations.

It might seem that the problem is over-parametrised, since we have n knots $\implies$ n parameters, but the penalty shrinks them.

Understanding the smoothing spline

As the solution is a natural spline, we can write it as

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j, \tag{12}$$

where $N_j(x)$ are an n-dimensional set of basis functions for representing this family of natural splines.

Hence, the criterion to optimise reduces to

$$\text{RSS}(\theta, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega_n\theta, \tag{13}$$

where the matrix $\{N\}_{i,j} = N_j(x_i)$, and

Understanding the smoothing spline — 2

$$\int \{f''(t)\}^2 dt = \int \left\{ \sum_{j=1}^{n} N_j''(t)\theta_j \right\}^2 dt \tag{14}$$

$$= \int \sum_{j=1}^{n} N_j''(t)\theta_j \sum_{i=1}^{n} N_i''(t)\theta_i \, dt \tag{15}$$

$$= \sum_{j=1}^{n}\sum_{i=1}^{n} \theta_i\theta_j \int N_i''(t)N_j''(t)\, dt \tag{16}$$

$$= \sum_{j=1}^{n}\sum_{i=1}^{n} \theta_i\theta_j(\Omega_n)_{i,j} \tag{17}$$

$$= \sum_{j=1}^{n} \theta_j \sum_{i=1}^{n} \theta_i(\Omega_n)_{i,j} = \theta^T\Omega_n\theta. \tag{18}$$

Understanding the smoothing spline — 3

And

$$(\Omega_n)_{i,j} = \int N_i''(t) N_j''(t)\, dt. \tag{19}$$

The criterion

$$\text{RSS}(\theta, \lambda) = (y - N\theta)^T (y - N\theta) + \lambda \theta^T \Omega_n \theta, \tag{20}$$

is similar that of reparametrised ridge regression with solution

$$\hat{\theta} = (N^T N + \lambda \Omega_n)^{-1} N^T y. \tag{21}$$

(see exercises). The fitted smoothing spline is then

$$\hat{f}(x) = \sum_{j=1}^{n} N_j(x) \hat{\theta}_j, \tag{22}$$

and $\hat{f}$ is the vector of fitted values.

Choosing λ

Clearly, λ is crucial to the performance.

If λ is prechosen and fixed, then the smoothing spline is a *linear smoother* because the estimated parameters, $\hat{\theta}$ are then linear combinations of the y_i.

Then:

$$\hat{f} = N(N^T N + \lambda \Omega_n)^{-1} N^T y = S_\lambda y, \tag{23}$$

where S_λ is called the *smoother matrix*.

Note: S_λ does not depend on y, just x and λ.

S_λ has similarities to the hat matrix we defined in Lecture 2 and worked out an effective degrees of freedom in Lecture 3 as $\text{df} = \text{tr}(H_\lambda)$.

Effective degrees of freedom

We can do the same here with S_λ and call the *effective degrees of freedom* of a smoothing spline to be

$$\mathrm{df}_\lambda = \mathrm{tr}(S_\lambda). \tag{24}$$

For the motorcycle data:

λ	10^{-14}	0.0001	0.001	5
df_λ	94.00	12.54	7.55	2.05

So, for $\lambda = 5$ penalty is 'essentially infinite' so have straight line with slope parameter and intercept parameter, so *two* degrees of freedom. Why 94.00 and not 133? Well, `length(unique(mcycle$times))`is 94 (actually software reports 93.99996.).

See Green, P.J. and Silverman, B.W. (1994) *Nonparametric regression and Generalized Linear Models: a roughness penalty approach*. CRC Press, section 3.3.4 on page 37. Chapter 2 and 3 describe in great detail the construction of splines and the smoothing spline (but the detail is beyond the scope of this course).

Understanding the smoothing spline — 4

We can write the smoother matrix in *Reinsch* form:

$$S_\lambda = (I + \lambda K)^{-1}, \tag{25}$$

where K does not depend on λ.

Since $\hat{f} = S_\lambda y$ solves

$$\min_f (y - f)^T (y - f) + \lambda f^T K f, \tag{26}$$

K is known as the penalty matrix.

S_λ is symmetric and positive semi-definite (see exercises).

Understanding the smoothing spline — 5

The eigendecomposition of S_λ is

$$S_\lambda = \sum_{k=1}^{n} \rho_k(\lambda) u_k u_k^T, \tag{27}$$

where u_k are the eigenvectors and $\rho_k(\lambda)$ are its eigenvalues and it can be shown that

$$\rho_k(\lambda) = \frac{1}{1 + \lambda d_k}, \tag{28}$$

where d_k is the corresponding eigenvalue from K.

Looks like a shrinkage operation.

Facts about smoothing spline

1. The eigenvectors are not affected by λ, the whole family of smoothing splines (for a particular x) indexed by λ have the same eigenvectors.

2. $S_\lambda y = \sum_{k=1}^{n} u_k \rho_k(\lambda) < u_k, y >$. So, the smoothing spline operation decomposes y into the basis u and then shrinks the components by $\rho_k(\lambda)$ (whereas selection does 0-1).

3. See other facts at the end of §5.5 of ELS.

Automatic choosing of λ

One method is to use 'leave one out cross-validation'.

Goal is to make $\hat{f}_\lambda$ as close as possible to $f(x)$.

E.g. minimise $\mathbb{E}\left[\int \left\{\hat{f}_\lambda(x) - f(x)\right\}^2 dx\right] = \text{MISE}$.

But we don't know $f(x)$, as that's what we're trying to obtain.

Define $\hat{f}_\lambda^{(-i)}$ to be the smoothing spline estimator for f constructed without the ith point.

So, estimate MISE by $\text{CV}(\hat{f}_\lambda) = \sum_{i=1}^n \left\{y_i - \hat{f}_\lambda^{(-i)}(x_i)\right\}^2$.

I.e. y_i proxy for $f(x)$ and $\hat{f}_\lambda^{(-i)}(x_i)$ proxy for $\hat{f}_\lambda(x)$.

Automatic choosing of λ — 2

It turns out that

$$\mathrm{CV}(\hat{f}_\lambda) = n^{-1} \sum_{i=1}^{n} \left\{ \frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)} \right\}^2, \tag{29}$$

which can be computed for each value of λ from the original fitted values and the diagonal elements $S_\lambda(i,i)$, don't have to compute each $\hat{f}_\lambda^{(-i)}$.

See Christensen, R. (2010) *Plane Answers to Complex Questions.* Springer, New York — Proposition 13.5.5 to get general idea.

Equivalent Kernels

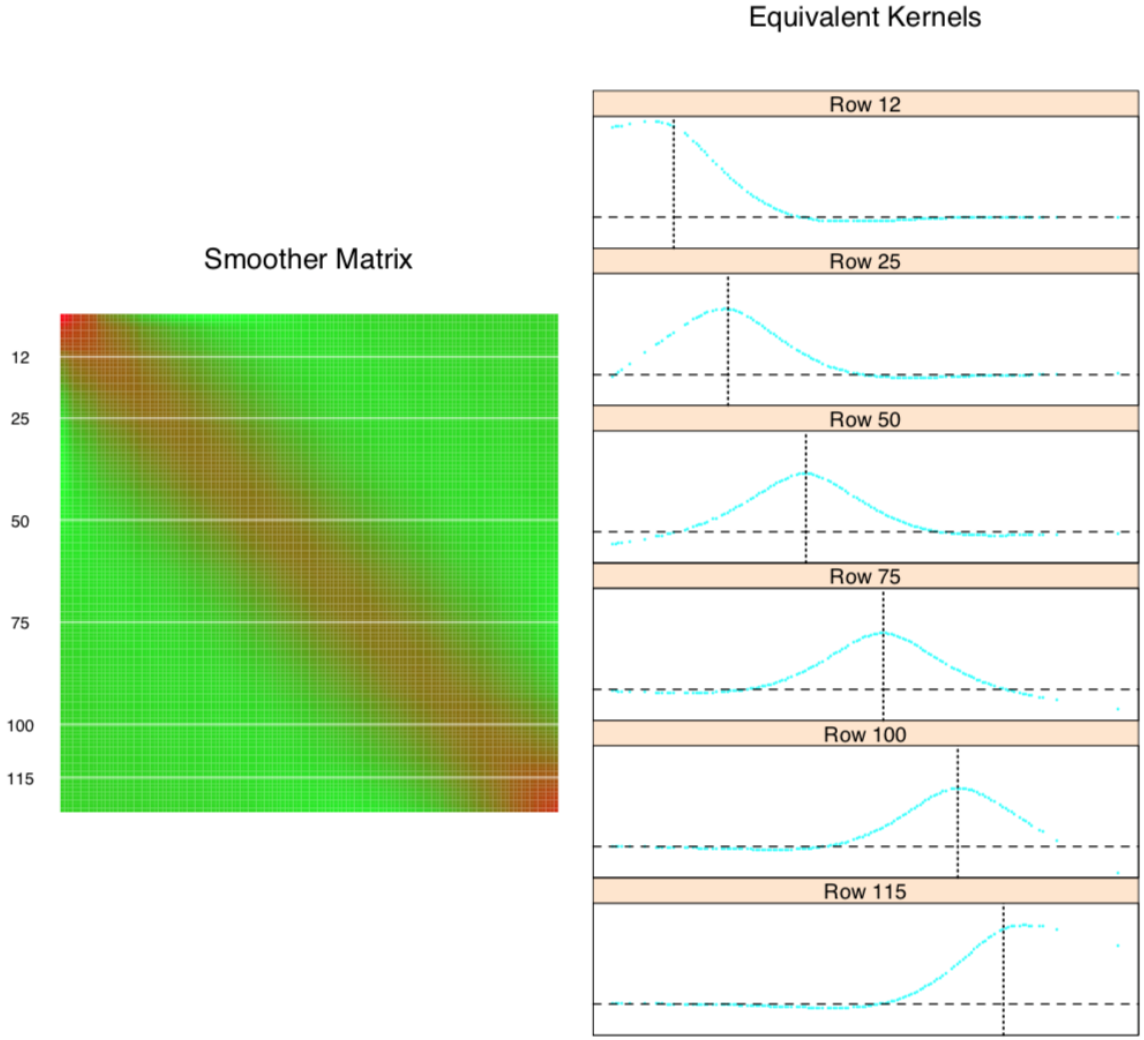


FIGURE 5.8. *The smoother matrix for a smoothing spline is nearly banded, indicating an equivalent kernel with local support. The left panel represents the elements of* $\mathbf{S}$ *as an image. The right panel shows the equivalent kernel or weighting function in detail for the indicated rows.*

Summary

This lecture covered the following topics

Alternatives to linearity

Piecewise polynomial: linear + cubic

Constraints, knots and splines

Smoothing splines

Smoothing spline is like ridge regression, shrinkage

Choosing smoothing parameter and cross-validation

Equivalent Kernels

Elements of Statistical Learning: Lecture 12. Kernel Smoothing

Guy Nason[1]

Department of Mathematics
Imperial College

Kernel Smoothing

Kernel smoothing refers to a wide range of techniques for doing density estimation and nonparametric regression.

The estimators from kernel smoothing are then often used in further analyses, such as classification, discrimination or just plain visualising.

Some good books on kernel smoothing (in case you're interested):

Wand, M.P. and Jones, M.C. (1994) *Kernel Smoothing*, Chapman and Hall, Boca Raton.

Silverman, B.W. (1986) *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, Boca Raton.

Note: the kernel methods here are different to the ones used in machine learning, such as those used by support vector machines.

Density Estimation — the problem

Suppose we have a set of data $X_1, \ldots, X_n$ which is an IID sample from probability density $f(x)$.

Here, we suppose $X_i \in \mathbb{R}$, but sometimes we have higher dimensional data.

We don't know $f(x)$.

Problem: we observe $X_1, \ldots, X_n$ and we wish to estimate $f(x)$.

The simplest method — the histogram — is something that we're surely all seen and been taught from a reasonably young age.

The histogram

However, the histogram suffers from several problems:

- the heights of bars depend quite drastically on the width of the bins. You can sometimes get a quite different looking histogram, just by changing the bar width.
- bars can be of different widths, which makes interpretation sometimes hard
- the histogram generally produces an estimator with discontinuities — this is not satisfactory if the true density $f(x)$ is smooth.

Kernels for kernel density estimation

We define the *kernel function* $K : \mathbb{R} \to \mathbb{R}$ which satisfies:

$K(x) \geq 0$, for all x

$\int_{\mathbb{R}} K(x) = 1$.

Usually: $K(-x) = K(x)$, i.e. K is symmetric and often we choose K to be smooth.

A key point is that *WE* choose the kernel.

Kernel Density Estimator

The kernel density estimator for $X_1, \ldots, X_n$, with kernel function K and *bandwidth* h is given by

$$\hat{f}_{n,h,K}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right). \tag{1}$$

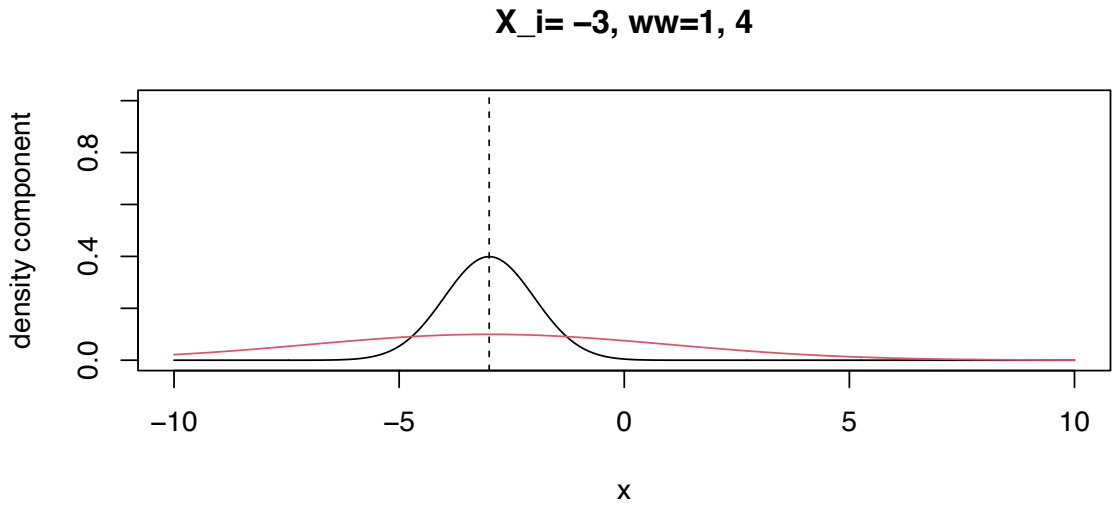
The bandwidth is sometimes called the *window width*.

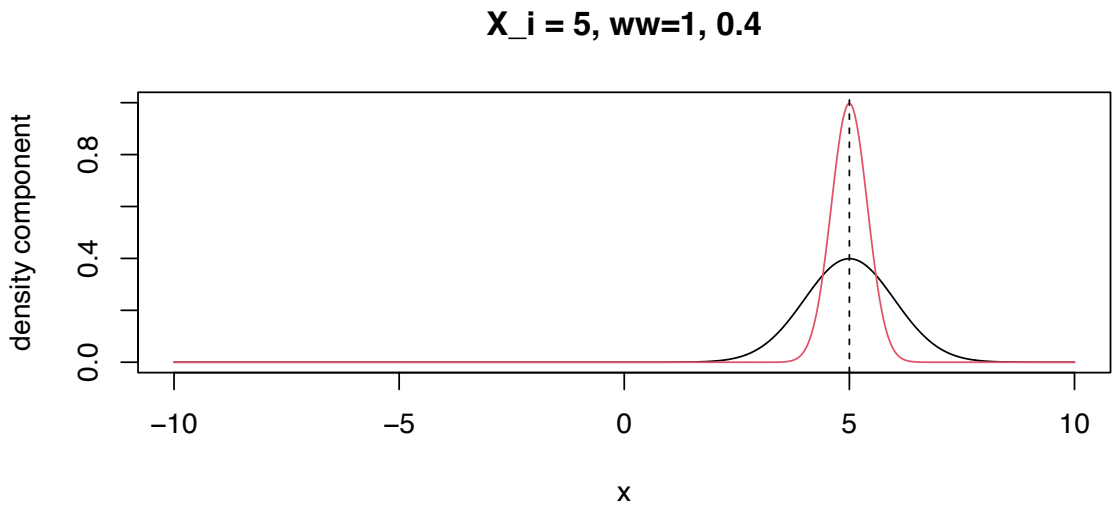
Sometimes we drop some of the subscripts and write $\hat{f}_h(x)$, etc.

The key is the component $K\{(X_i - x)/h\}$.

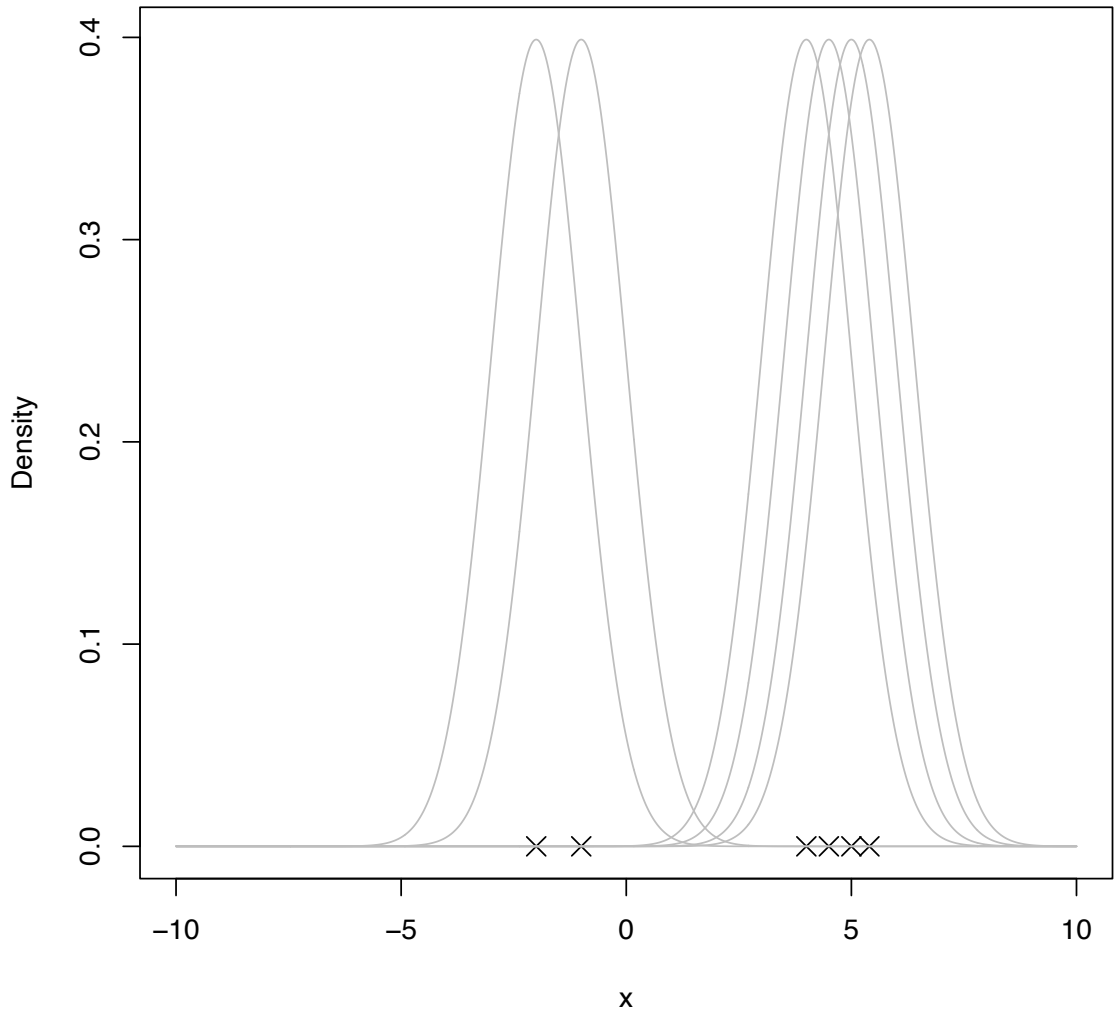
This term places a kernel function, centred at X_i, with width controlled by h.

Some kernel components

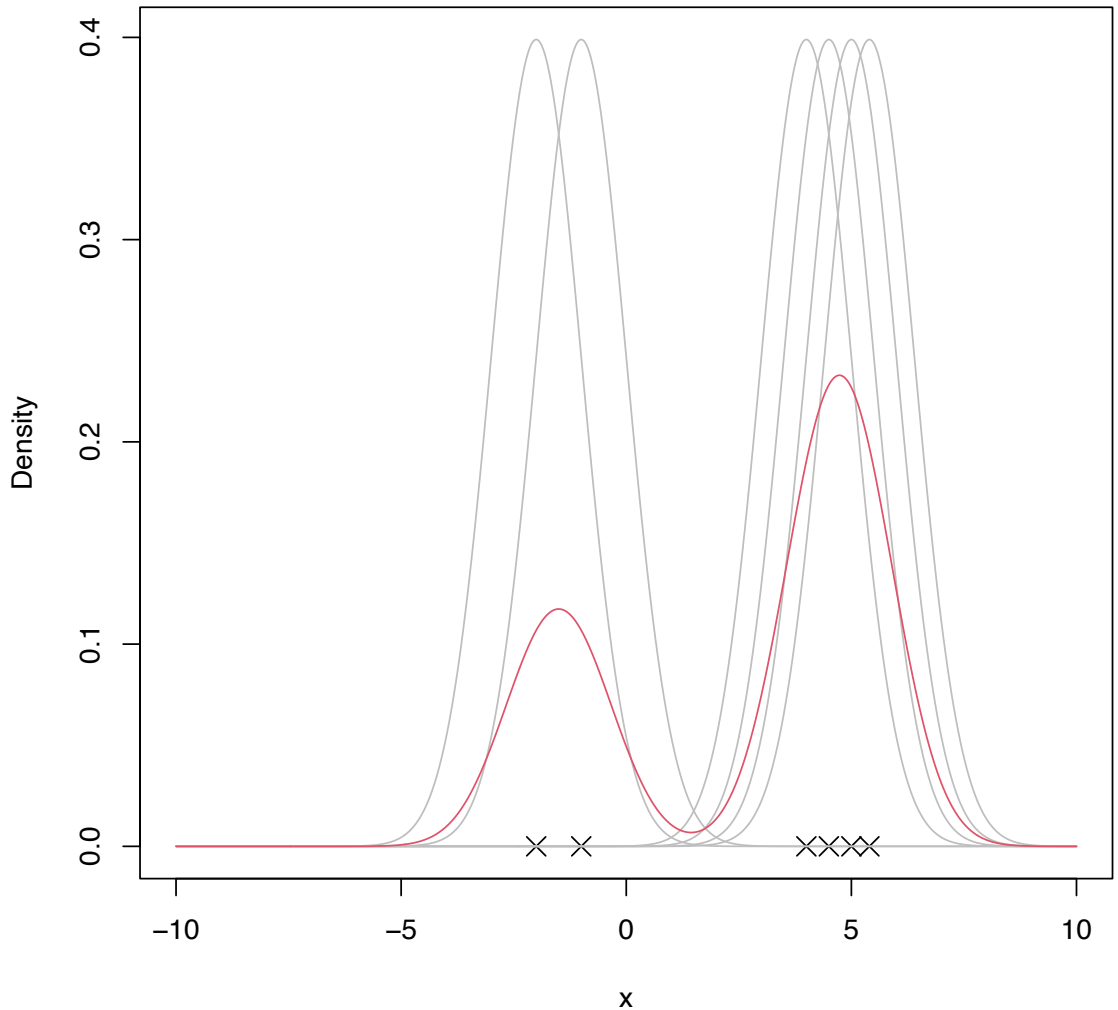




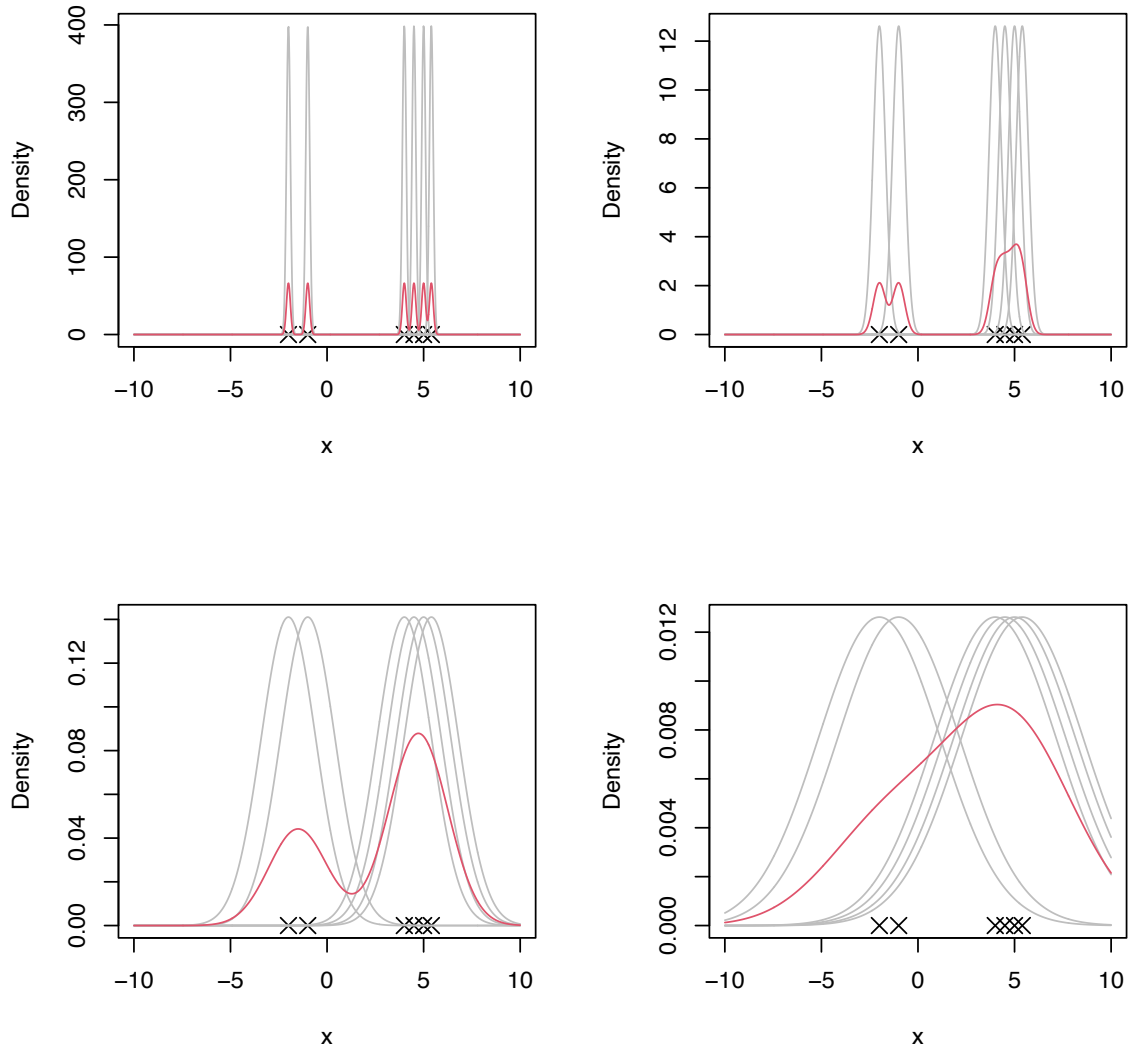
Kernel placement $h = 1$ (sample data set of 6 points)



As before, but averaging kernels



Four different bandwidths ($h = 0.1, 0.316, 1.41, 3.16$)



Choice of bandwith is very important

If h is small, then the density estimate is highly variable.

If h is too large, then the density estimate is very biased.

h smaller $\implies$ higher variance, lower bias.

h larger $\implies$ higher bias, lower variance.

Note: $\hat{f}_{n,h,L}(x)$ is a *random variable*, depends on $X_1, \ldots, X_n$.

Working out the bias and variance

Given a single observation, X, what is the probability of it landing in the interval $(x - h/2, x + h/2)$?

This is

$$p = \mathbb{P}(x - h/2 < X < x + h/2) = F(x + h/2) - F(x - h/2), \quad (2)$$

where $F(x)$ is the distribution associated with $f(x)$.

Now suppose we use a *rectangular* kernel function

$$K(x) = \begin{cases} 1 & \text{for } x \in (-1/2, 1/2), \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Hence, we only get a contribution with respect to X_i when

$$-1/2 \leq (X_i - x)/h \leq 1/2 \text{ iff } x - h/2 \leq X_i \leq x + h/2. \quad (4)$$

Bias and variance — rectangular kernel

So

$$\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \tag{5}$$

counts the number of $X_1, \ldots, X_n$ that fall into the interval $(x - \frac{h}{2}, x + \frac{h}{2})$.

Hence $n^{-1} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$ is the average number of points in the interval and $\hat{f}_{n,h}(x) =$ average number of points$/h$.

What is the number of points that fall into the interval?

This is a random variable, call it N^*. Since the chance of one point falling in is p (above, distributed according to a Bernoulli r.v.), by summing over all the points, the distribution of N^* is

$$nh\hat{f}_{n,h}(x) \sim \text{Bin}(n, p). \tag{6}$$

Bias and variance — rectangular kernel — 2

Recall that the mean and variance of a Bin(n, p) binomial r.v. is

$$\mathbb{E}(N^*) = np, \quad \text{var}(N^*) = np(1-p). \tag{7}$$

Hence,

$$\begin{aligned} \mathbb{E}\{\hat{f}_{n,h}(x)\} &= (nh)^{-1} n\{F(x+h/2) - F(x-h/2)\} && (8) \\ &= \frac{F(x+h/2) - F(x-h/2)}{h}. && (9) \end{aligned}$$

We'd like $\mathbb{E}\{\hat{f}_{n,h}(x)\} = f(x)$, but it is not. So, the density estimator is, in general, biased.
However, if $F(x)$ is differentiable then

$$\frac{F(x+h/2) - F(x-h/2)}{h} \to f(x) \quad \text{as } h \to 0, \tag{10}$$

so the bias of the estimate should decrease as h gets small.This fits in with our experience of the plots.

Comments on h

Clearly, h cannot be too small because

- few points will end up under most of the kernels
- the estimator will be very variable.

However, if we get more data, and n increases there will be many more points and we can allow h to shrink.

This suggests that a good h should depend on n, i.e. $h = h_n$ and

$$h_n \to 0 \quad \text{as } n \to \infty. \tag{11}$$

Variance of (rectangular) kernel estimator

Invent notation

$$F(x - h/2, x + h/2) = F(x + h/2) - F(x - h/2). \qquad (12)$$

Due to properties of binomial random variables

$$\text{var}\{nh\hat{f}(x)\} = nF(x-h/2, x+h/2)\{1-F(x-h/2, F+h/2)\} \quad (13)$$

Since $\text{var}(cW) = c^2 \text{var}(W)$ we can write (13) as

$$\begin{aligned} \text{var}\{\hat{f}(x)\} &= (nh)^{-2} nF(x - h/2, x + h/2)\{1 - F(x - h/2, F + h/2)\} \\ &= \frac{1}{nh} \frac{F(x - h/2, x + h/2)}{h} \{1 - F(x - h/2, x + h/2)\}. \quad (14) \end{aligned}$$

If h is small, then $F(x + h/2) - F(x - h/2)$ is small as this is the probability that X lies in the interval $(x - h/2, x + h/2)$.

Variance of (rectangular) kernel estimator — 2

Hence,

$$1 - F(x - h/2, x + h/2) \to 1 \text{ as } h \to 0. \tag{15}$$

For the second term of (14), this is the area under the $f(x)$ curve, which, since h is small we can approximate by a rectangle with width h and height $f(x)$. So,

$$F(x - h/2, x + h/2) \approx hf(x). \tag{16}$$

So, the second term is this divided by h, so the second term of (14) tends to $f(x)$.

Hence, putting it altogether

$$\text{var}\{\hat{f}(x)\} \approx \frac{1}{nh} f(x). \tag{17}$$

Consistency

Consistency means that the mean squared error tends to zero as $n \to \infty$.

This means that we want the bias and the variance to tend to zero as $n \to \infty$. (Recall we showed that MSE $=$ bias^2 $+$ variance).

For the bias, we need $h_n \to 0$ from above.

For the variance, we need

$$nh_n \to \infty \text{ as } n \to \infty. \tag{18}$$

Tradeoff: if h is small, then bias is good, but variance large and the opposite happens if h is large.

Best choice of h?

Suppose we want to minimise the MSE as a function of h?

We have a good approximation for the variance, but now we need one for the bias too (which we'd previously only worked out a limit for).

We use Taylor approximation to do this.

However, we need an extra assumption that f is twice differentiable.

We will expand $F(x + h/2)$ around $F(x)$ assuming h small.

Taylor Series for F

First, the generic series expansion:

$$F(x+\delta) = F(x) + \delta F'(x) + \delta^2 F''(x)/2! + \delta^3 F^{(3)}(x)/3! + \mathcal{O}(\delta^4). \tag{19}$$

Also, recall that $f(x) = F'(x)$, density is derivative of distribution.

Hence,

$$F(x+h/2) = F(x) + \frac{h}{2} f(x) + \frac{1}{2} f'(x) \left(\frac{h}{2}\right)^2 + \frac{1}{6} f''(x) \left(\frac{h}{2}\right)^3 + \mathcal{O}(h^4). \tag{20}$$

Similarly,

$$F(x-h/2) = F(x) - \frac{h}{2} f(x) + \frac{1}{2} f'(x) \left(\frac{h}{2}\right)^2 - \frac{1}{6} f''(x) \left(\frac{h}{2}\right)^3 + \mathcal{O}(h^4). \tag{21}$$

Taylor Series for F — 2

Then the difference is

$$F(x - h/2, x + h/2) = hf(x) + \frac{1}{24}h^3 f''(x) + \mathcal{O}(h^4). \qquad (22)$$

Hence, the bias from (10) is

$$\text{bias}\{\hat{f}(x)\} = \frac{F(x - h/2, x + h/2)}{h} - f(x) \approx \frac{1}{24}h^2 f''(x), \qquad (23)$$

ignoring remainder terms. Now use MSE = variance + bias^2

$$\text{MSE}\{\hat{f}_{n,h,K}(x)\} \approx C_1 \frac{1}{nh} f(x) + C_2 \{f''(x)\}^2 h^4, \qquad (24)$$

where $C_1 = 1$ and $C_2 = (1/24)^2$.

Minimising MSE over h

The derivative

$$\frac{\partial \,\mathrm{MSE}}{\partial h} \approx -C_1 n^{-1} h^{-2} f(x) + 4C_2 \{f''(x)\}^2 h^3. \tag{25}$$

Setting $\frac{\partial \,\mathrm{MSE}}{\partial h}|_{h^*} = 0$ gives

$$C_1 n^{-1} (h^*)^{-2} f(x) = 4C_2 \{f''(x)\}^2 (h^*)^3 \tag{26}$$

which implies

$$(h^*)^5 = \frac{C_1}{4C_2} \frac{f(x)}{\{f''(x)\}^2} n^{-1}. \tag{27}$$

or

$$h^* = C^* n^{-1/5}, \tag{28}$$

where $C^* = \left[\frac{C_1}{4C_2} \frac{f(x)}{\{f''(x)\}^2} \right]^{1/5}$.

Properties of optimal bandwidth

Clearly, $h_n \to 0$ as $n \to \infty$.

Clearly $nh_n \propto n^{4/5} \to \infty$ as $n \to \infty$.

But h^* depends on C^* and we don't know $f(x)$, let alone $f''(x)$, that is what we are trying to estimate. These calculations for optimal bandwidth often fall into a circular argument.

The bandwidth h^* depends on x. So, the best bandwidth *depends* on where in the domain of f you are trying to estimate.

For more general kernels, $C_2 = \frac{1}{4}\{\int x^2 K(x)\,dx\}^2 = \frac{1}{4}C_3^2$ and $C_1 = \int K^2(x)\,dx$.

Expectation with general kernel

Take expectation of the definition (1) directly.

$$\mathbb{E}\{\hat{f}(x)\} = (nh)^{-1} \sum_{i=1}^{n} \mathbb{E}\left\{ K\left(\frac{X_i - x}{h} \right) \right\}. \tag{29}$$

Let

$$\begin{aligned} (*) &= \mathbb{E}\left\{ K\left(\frac{X_i - x}{h} \right) \right\} && (30) \\ &= \int K\{(y-x)/h\} f(y)\, dy. && (31) \end{aligned}$$

Substitute $v = (y - x)/h$. Hence $dv = dy/h$. Also $h > 0$.

Expectation with general kernel — 2

Now we use a Taylor expansion of f around x, so

$$f(x+\delta) = f(x) + \delta f'(x) + \delta^2 f''(x)/2! + \mathcal{O}(\delta^3). \tag{32}$$

So with $\delta = hv$ we have

$$(*) = h \int K(v) f(hv + x)\, dv \tag{33}$$

$$= hf(x) \underbrace{\int K(v)\, dv}_{1} + h^2 f'(x) \underbrace{\int vK(v)\, dv}_{0} \tag{34}$$

$$+ \frac{1}{2} h^3 f''(x) \underbrace{\int v^2 K(v)\, dv}_{C_3} + \mathcal{O}(h^4). \tag{35}$$

The middle zero because $K(x)$ is symmetric.

Expectation with general kernel — 3

So,

$$(*) = hf(x) + \tfrac{1}{2} C_3 h^3 f''(x) + \mathcal{O}(h^4), \tag{36}$$

so that

$$\text{bias}\{\hat{f}_{n,h,K}(x)\} \approx \tfrac{1}{2} C_3 h^2 f''(x). \tag{37}$$

The bias of our estimator depends on the second derivative of $f(x)$

So, the bias in our estimator will be worse near to areas where $f''(x)$ is large.

The areas where $f''(x)$ are large are area of high curvature.

So, we expect our estimator to be biased near to where $f(x)$ is very wiggly.

Bounding variance with general kernel

First, recall that

$$\text{var}(W) = \mathbb{E}(W^2) - E(W)^2 \leq \mathbb{E}(W^2). \tag{38}$$

Then

$$\begin{aligned}
\text{var}\{\hat{f}(x)\} &= \frac{1}{nh^2}\,\text{var}\left\{K\left(\frac{X_i - x}{h}\right)\right\} && (39)\\
&\leq \frac{1}{nh^2}\mathbb{E}\left\{K^2\left(\frac{X_i - x}{h}\right)\right\} && (40)\\
&= \frac{1}{nh^2}\int K^2\left(\frac{y - x}{h}\right) f(y)\,dy && (41)\\
&= \frac{1}{nh}\int K^2(v) f(hv + x)\,dv, && (42)
\end{aligned}$$

now using the same Taylor series expansion for $f(x + hv)$ as before, gives ...

Bounding variance with general kernel

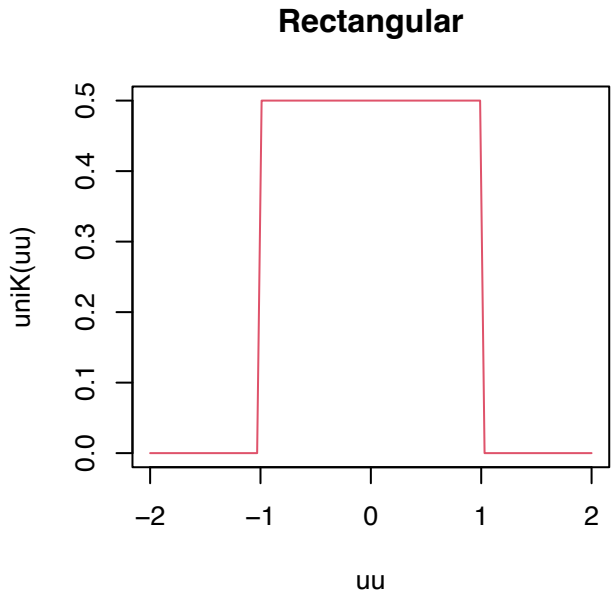
$$
\begin{aligned}
\operatorname{var}\{\hat{f}(x)\} &\leq \frac{1}{nh}\left\{ f(x)\int K^2(v)\,dv + hf'(x)\int vK^2(v)\,dv \right. && (43)\\
&\quad \left. +\mathcal{O}(h^2)\right\} && (44)\\
&= \frac{1}{nh} f(x) C_1 + \frac{1}{n} C_4 + \mathcal{O}(h/n) && (45)\\
&\quad \to 0 \text{ as } n \to \infty, && (46)
\end{aligned}
$$

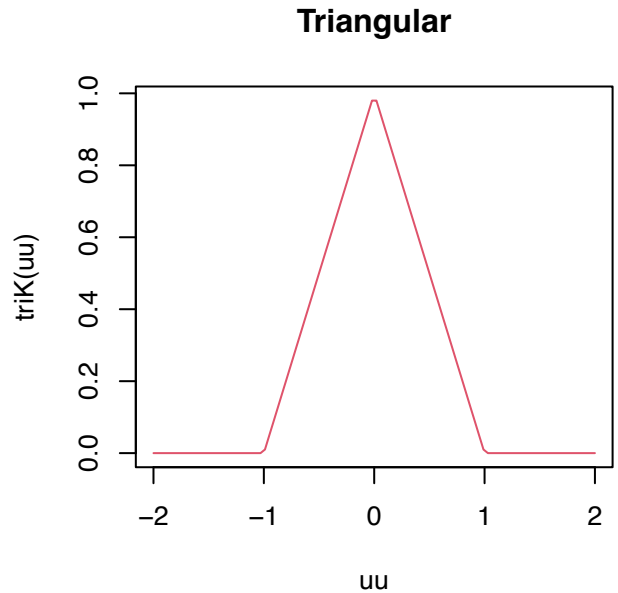
where $C_4 = f'(x)\int vK^2(v)\,dv$.

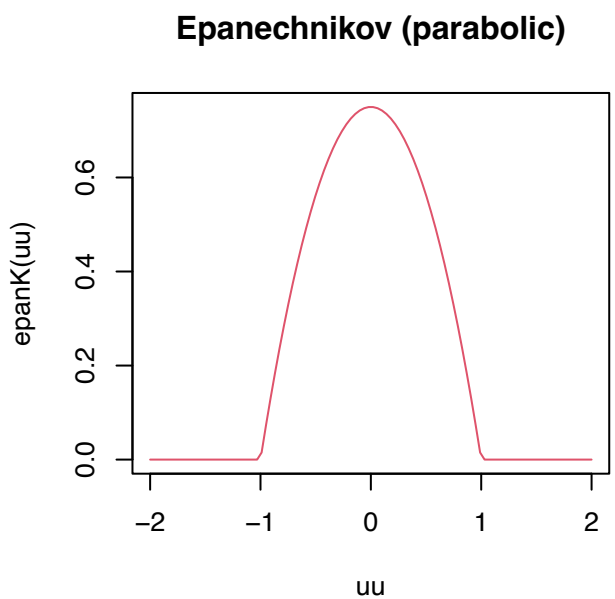
Variance is high when the density is high.

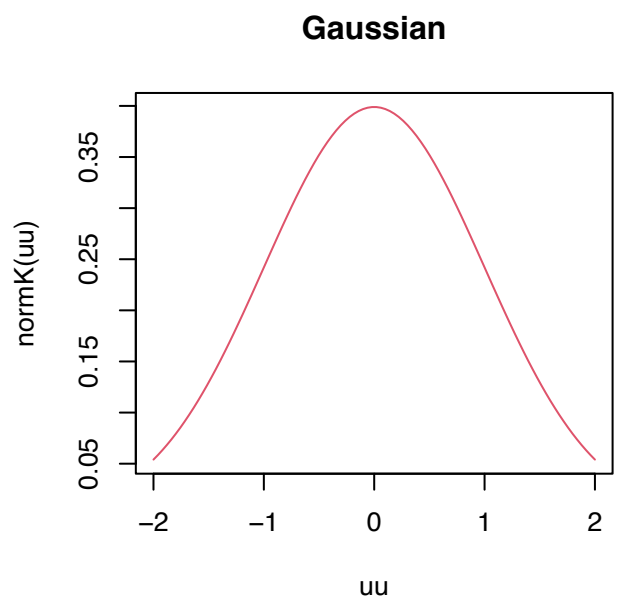
Unsurprisingly, the bias and variance are similar to previously, with the rectangular kernel, so same kind of optimal bandwidth results.

Some Example Kernels

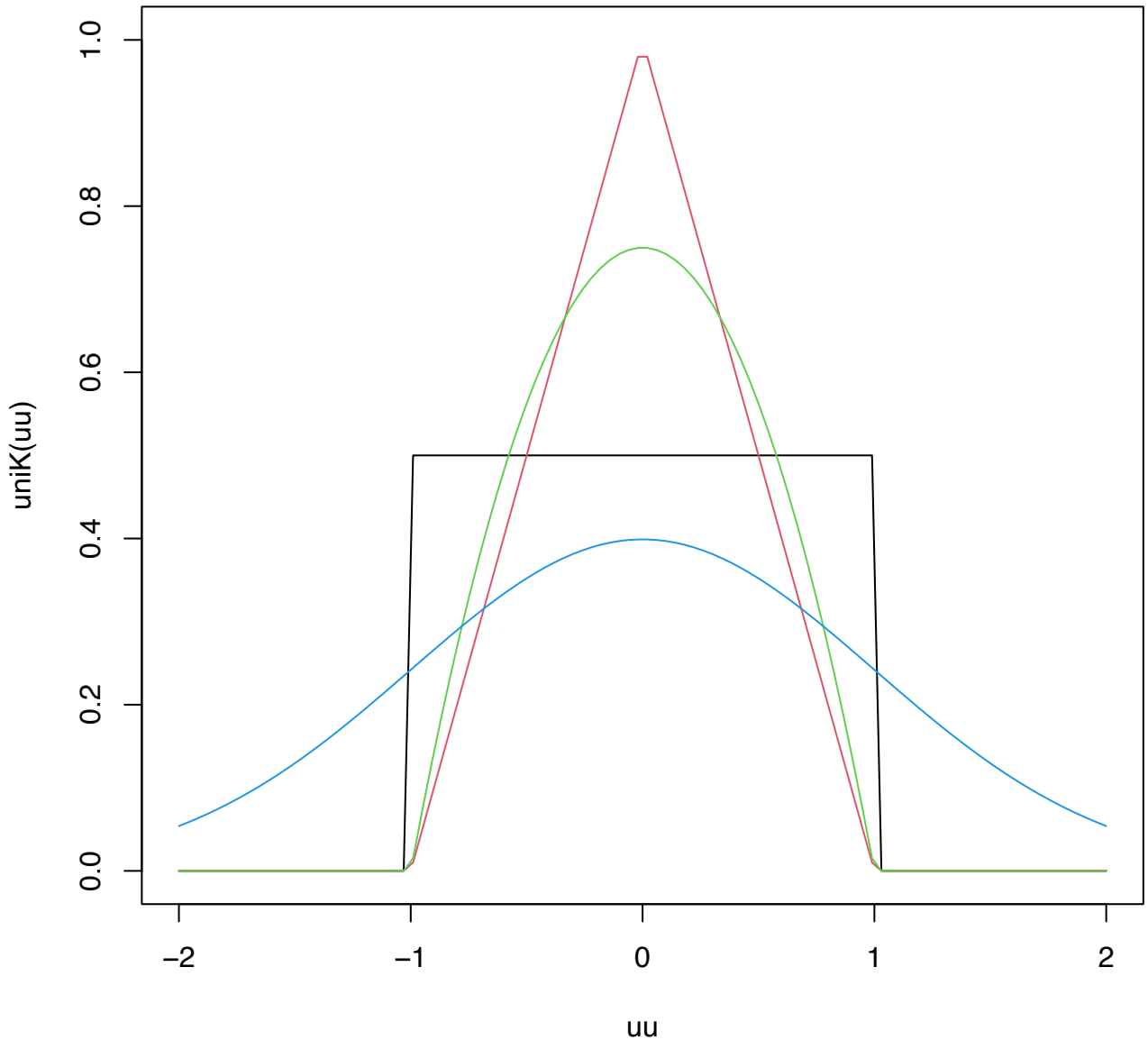








Some Example Kernels Compared



Example Kernel Formulae

Rectangular

$$K(x) = \begin{cases} 1/2 & x \in (-1, 1), \\ 0 & \text{otherwise.} \end{cases}$$

Triangular

$$K(x) = \begin{cases} 1 - |x| & x \in (-1, 1), \\ 0 & \text{otherwise.} \end{cases}$$

Epanechnikov

$$K(x) = \begin{cases} \frac{3}{4}(1 - x^2) & x \in (-1, 1), \\ 0 & \text{otherwise.} \end{cases}$$

Normal/Gaussian

$$K(x) = (2\pi)^{-1/2} \exp(-x^2/2), \ x \in \mathbb{R}.$$

Summary

This lecture has covered:

Kernel Smoothing

Kernel Density Estimation

Bias and Variance and bandwidth selection for Kernel DE

Types of Kernel

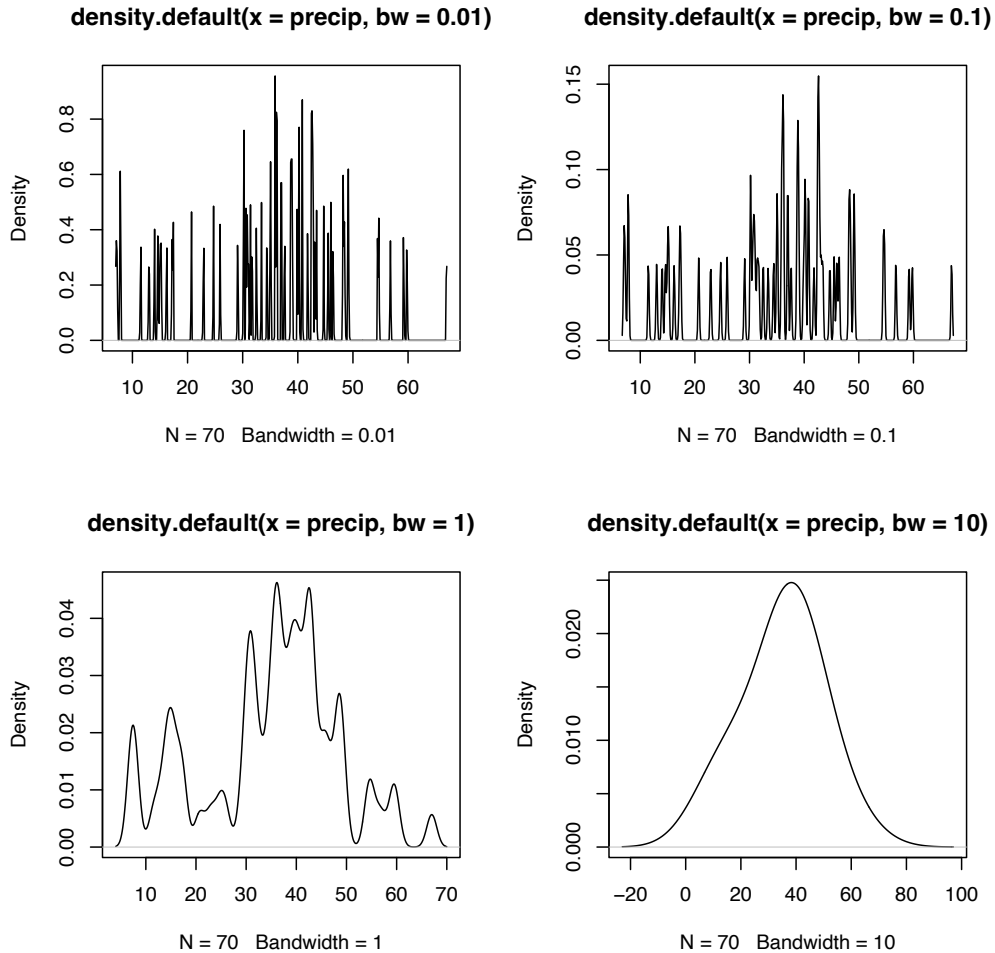
Elements of Statistical Learning: Lecture 13. Kernel DE and Regression

Guy Nason[1]

Department of Mathematics
Imperial College

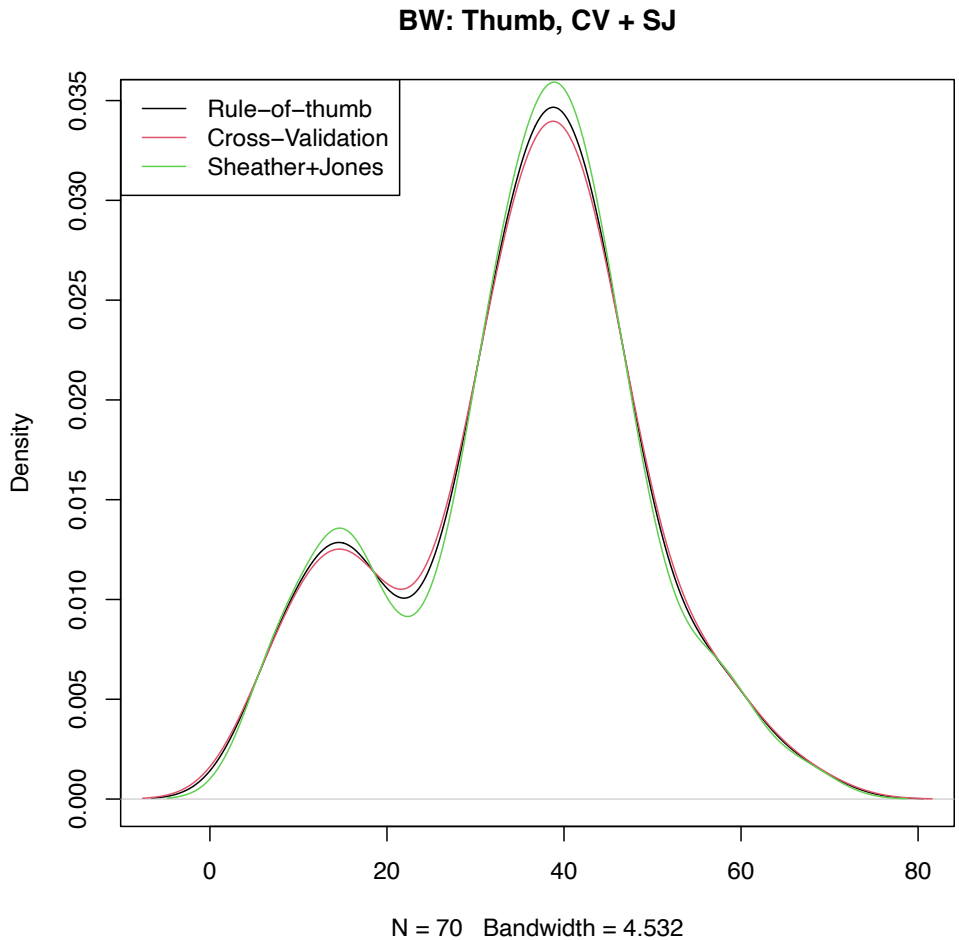
precip kernel density estimates

```
plot(density(precip, bw=0.01))
```



precip with automatic estimate of bandwidth

```
plot(density(precip, bw=bw.nrd(x=precip)), main="BW: Thumb
```



Kernel Regression

Suppose now we have variables X, Y and we want to find $\mathbb{E}(Y|X)$.

We go via the joint density $f_{X,Y}(x, y)$.

$$\begin{aligned} \mathbb{E}(Y|X) &= \int y f(y|x)\, dy && (1) \\ &= \int y \frac{f(x,y)}{f(x)}\, dy && (2) \\ &= \frac{\int y f(x,y)\, dy}{f(x)} && (3) \end{aligned}$$

Now we make use of our kernel density estimate for X

$$\hat{f}(x) = n^{-1} \sum_{i=1}^{n} K_h(x - X_i), \qquad (4)$$

where $K_h(x) = h^{-1} K(x/h)$.

Kernel Regression — 2

We introduce the 2D kernel density estimator

$$\hat{f}(x,y) = n^{-1} \sum_{i=1}^{n} K_h(x - X_i) K_h(y - Y_i) \tag{5}$$

So,

$$\begin{aligned} \hat{\mathbb{E}}(Y|X=x) &= \frac{\int y \hat{f}(x,y)\, dy}{\hat{f}(x)} && (6) \\ &= \frac{\sum_{i=1}^{n} K_h(x - X_i) \int y K_h(y - Y_i)\, dy}{\sum_{i=1}^{n} K_h(x - X_i)}, && (7) \end{aligned}$$

and the n^{-1} cancel.

Kernel Regression — 3

Now let $v = y - Y_i$, then

$$\int yK_h(y - Y_i)\,dy = \int (v + Y_i)K_h(v)\,dv \tag{8}$$

$$= \int_{}^{0} vK_h(v)\,dv + Y_i \int_{}^{1} K_h(v)\,dv, \tag{9}$$

$$= Y_i \tag{10}$$

since the kernel is symmetric and integrates to 1.

Hence.

$$\hat{\mathbb{E}}(Y|X = x) = \frac{\sum_{i=1}^{n} Y_i K_h(x - X_i)}{\sum_{i=1}^{n} K_h(x - X_i)} \tag{11}$$

This estimator is called the *Nadaraya-Watson* kernel regression.

Local Polynomial Regression

This generalizes the Nadaraya-Watson estimator.

First, we define the local polynomial estimator:

$$m_{x_0}(x) = \sum_{j=0}^{p} \beta_j(x_0)(x - x_0)^j, \tag{12}$$

centred on x_0 or *local* to x_0.

E.g. $\{\beta_j(x_0)\}$ are the regression coefficients near to x_0 and

$\{\beta_j(x_1)\}$ are the regression coefficients near to x_1.

I.e. the local regression coefficients change over location.

Local Polynomial Regression — 2

Here, we have a weighted residual sum of squares:

$$\text{RSS}(x_0) = \sum_{i=1}^{n} \{Y_i - m_{x_0}(X_i)\}^2 K_h(X_i - x_0). \tag{13}$$

The h bandwidth ensures that only (X_i, Y_i) close to x_0 get included in the local regression.

The nearer they are, the higher their contribution.

Writing the design matrix as

$$X = \begin{pmatrix} 1 & X_1 - x_0 & \cdots & (X_1 - x_0)^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n - x_0 & \cdots & (X_n - x_0)^p \end{pmatrix} \tag{14}$$

Local Polynomial Regression — 3

Write the $n \times n$ weight matrix as

$$W_{x_0} = \text{diag}\{K_h(X_1 - x_0), \cdots, K_h(X_n - x_0)\} \tag{15}$$

Then, the weighted least-squares problem (13) can be written

$$\text{RSS}(x_0) = \{Y - X\beta(x_0)\}^T W_{x_0} \{Y - X\beta(x_0)\}. \tag{16}$$

We can minimize this using previous methods (easy to see, $W_{x_0}^{1/2}$ exists, as W_{x_0} is diagonal and all entries ≥ 0. Set $X_R = W_{x_0}^{1/2} X$ and $Y_R = W_{x_0}^{1/2} Y$ and use least squares method on X_R, Y_R.)

The minimiser of $\text{RSS}(x_0)$ is

$$\hat{\beta}(x_0) = (X^T W_{x_0} X)^{-1} X^T W_{x_0} Y. \tag{17}$$

Bias of NW [Local Constant] and Local Linear regression

Long calculation.

Let $f(x)$ be density of the $\{X_i\}$ as above, and $g(x) = \mathbb{E}(Y|X = x)$ be the unknown regression function.

Variance of NW and local linear regressions is the same, up to order $o\{(nh)^{-1}\}$.

Bias of NW is

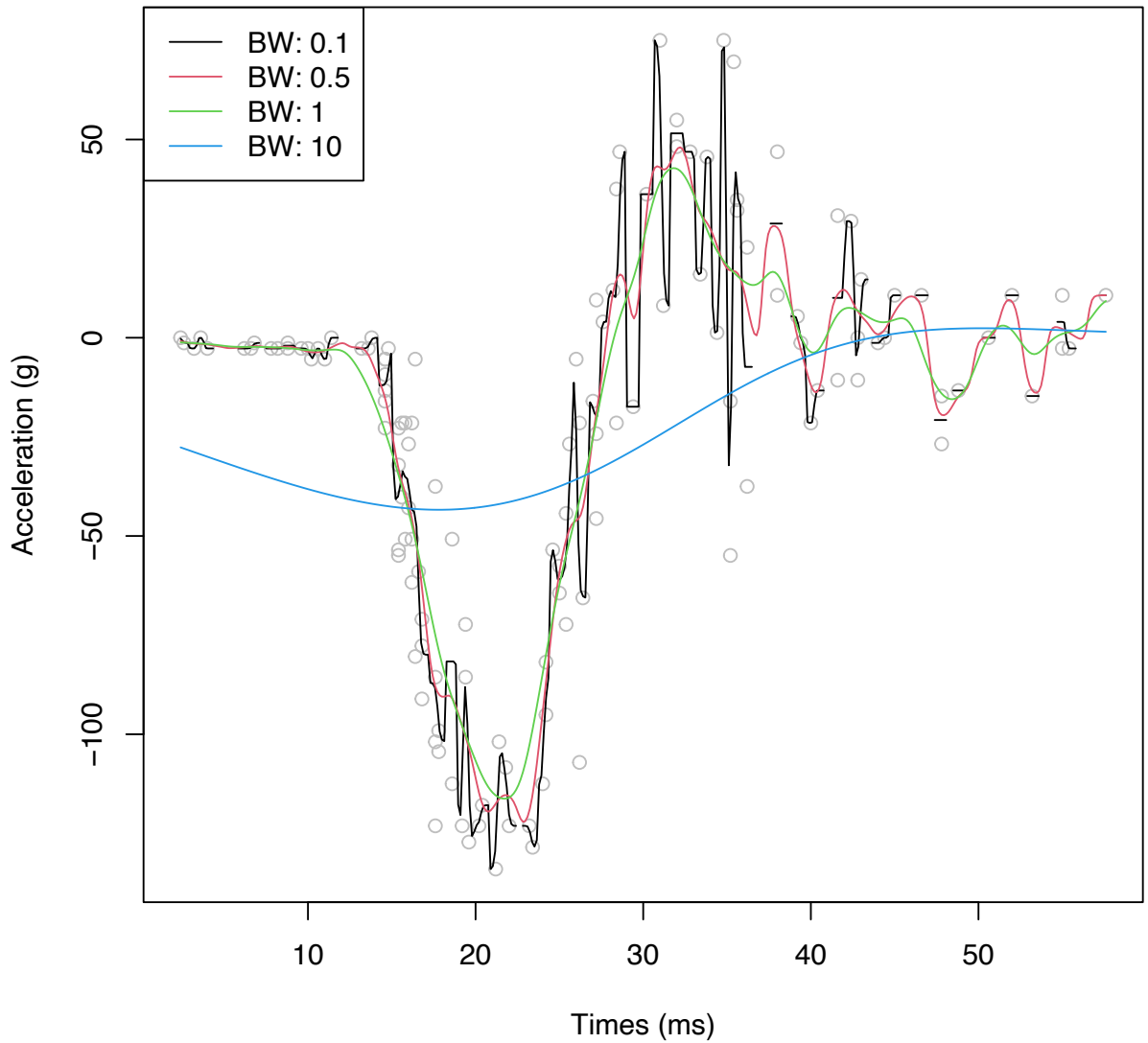
$$h^2 \left\{ \tfrac{1}{2} g''(x) + \frac{g'(x) f'(x)}{f(x)} \right\} \int K^2(u)\, du + o(h^2), \qquad (18)$$

Bias of local linear is

$$h^2 \tfrac{1}{2} g''(x) \int K^2(u)\, du + o(h^2). \qquad (19)$$

Generally, choose *odd* polynomial order.

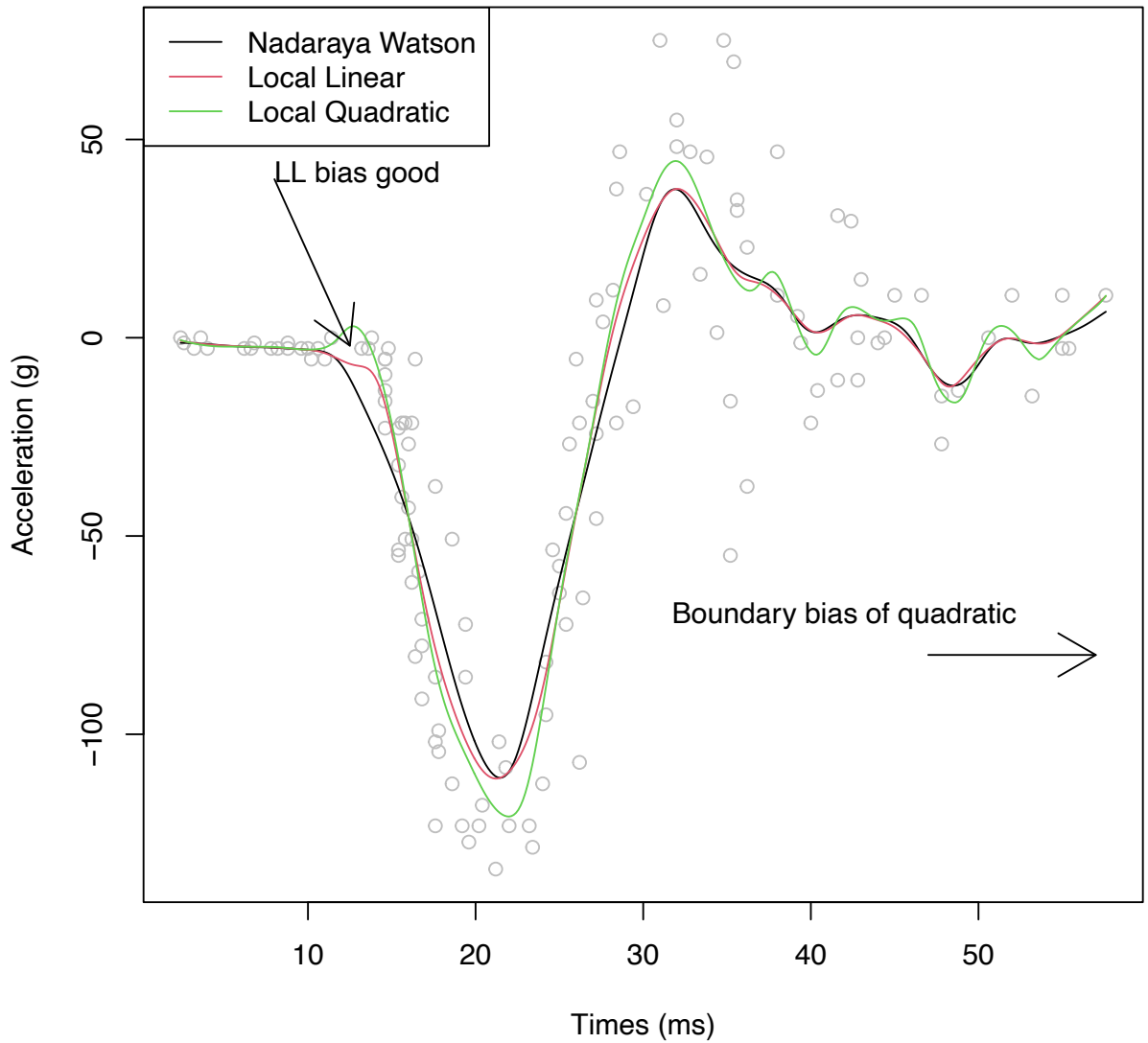
Motorcycle: Nadaraya-Watson with different bandwidths



Motorcycle: NW, local linear and quadratic. BW=1.44

```
xx <- mcycle[,1]
yy <- mcycle[,2]

good.bw <- dpill(x=xx, y=yy)

cat("Good bw: ", good.bw, "\n")

lp.d0 <- locpoly(x=xx, y=yy, degree=0, bandwidth=good.bw)
lp.d1 <- locpoly(x=xx, y=yy, degree=1, bandwidth=good.bw)
lp.d2 <- locpoly(x=xx, y=yy, degree=2, bandwidth=good.bw)

plot(xx, yy, col="gray", xlab="Times (ms)",
    ylab="Acceleration (g)")
lines(lp.d0$x, lp.d0$y)
lines(lp.d1$x, lp.d1$y, col=2)
lines(lp.d2$x, lp.d2$y, col=3)
```

Motorcycle: NW, local linear and quadratic. BW=1.44



Orthogonal series expansions

Here, let $\{\rho_\nu(x)\}$ be an orthogonal series basis for some function space (assume univariate and real-valued).

This means, for functions, $f(x)$, in the space we have

$$f(x) = \sum_\nu f_\nu \rho_\nu(x), \tag{20}$$

where the coefficients $\{f_\nu\}$ are found by

$$f_\nu =< f, \rho_\nu >= \int f(x)\rho_\nu(x)\, dx. \tag{21}$$

The orthogonality between the $\{\rho_\nu(x)\}$ means

$$\int \rho_\nu(x)\rho_\mu(x)\, dx = \delta_{\nu,\mu}, \tag{22}$$

where the Kronecker delta $\delta_{\nu,\mu} = 1$ if $\nu = \mu$ and 0 otherwise.

Two-dimensional expansion

A 2D separable expansion might be

$$f(x,y) = \sum_{\nu} \sum_{\mu} f_{\nu,\mu} \rho_\nu(x) \rho_\mu(y), \tag{23}$$

where

$$f_{\nu,\mu} = \int \int f(x,y) \rho_\nu(x) \rho_\mu(y)\, dxdy. \tag{24}$$

We also have (insist) that the set of basis functions is *complete* for the space of functions we are interested in, which means

$$\lim_{m\to\infty} \int \left\{ f(x) - \sum_{r=-m}^{m} f_r \rho_r(x) \right\}^2 dx = 0, \tag{25}$$

for all f in the space.

Orthogonal Series Regression

Now (3) says we can make use of $f(x)$. Can we estimate it? From (21) we have

$$f_\nu = \int f(x)\rho_\nu(x)\,dx = \mathbb{E}\{\rho_\nu(X)\} \approx n^{-1}\sum_{i=1}^{n}\rho_\nu(X_i) = \hat{f}_\nu. \quad (26)$$

as $f(x)$ is the density of X.Similarly,

$$f_{\nu,\mu} = \mathbb{E}\{\rho_\nu(X)\rho_\mu(Y)\} \approx n^{-1}\sum_{i=1}^{n}\rho_\nu(X_i)\rho_\mu(Y_i) = \hat{f}_{\nu,\mu}. \quad (27)$$

We can use $\{\hat{f}_\nu\}$ and $\{\hat{f}_{\nu,\mu}\}$ to estimate $f(x), f(x,y)$ by plugging the coefficients into (20) and (23), respectively.

Orthogonal Series Regression — 2

If we can calculate $\hat{f}_\nu$ *well* for all ν then

$$\begin{aligned}
\mathbb{E}\{\hat{f}(x)\} &= \mathbb{E}\left[\sum_\nu \left\{ n^{-1} \sum_{i=1}^n \rho_\nu(X_i) \right\} \rho_\nu(x)\right] && (28)\\
&= n^{-1} \sum_{i=1}^n \sum_\nu \mathbb{E}\{\rho_\nu(X_i)\} \rho_\nu(x) \quad (\mathbb{E}\{\rho_\nu(X_i)\} \to f_\nu) && (29)\\
&= n^{-1} \sum_{i=1}^n f(x) = f(x), && (30)
\end{aligned}$$

i.e., it looks like $\hat{f}(x)$ is unbiased.

But, the range of ν is often infinite, e.g. $\nu = -\infty, \ldots, \infty$.

And we can't use a finite set of things $X_1, \ldots, X_n$ to estimate an infinite number of things sensibly.

Linear Orthogonal Series Estimators

So, usually we only estimate for $\nu = -m, \ldots, m$, with $m < n$ and

$$\hat{f}(x) = \sum_{\nu=-m}^{m} \hat{f}_\nu \rho_\nu(x), \tag{31}$$

which is called the *linear orthogonal series estimator with truncation point of m*.This new estimator is biased, e.g.

$$\begin{aligned}
\text{bias}\{\hat{f}(x)\} &= \mathbb{E}\{\hat{f}(x)\} - f(x) && (32)\\
&= \sum_{\nu=-m}^{m} \mathbb{E}(\hat{f}_\nu)\rho_\nu(x) - \sum_{\nu=-\infty}^{\infty} f_\nu \rho_\nu(x) && (33)\\
&= -\sum_{|\nu|>m} f_\nu \rho_\nu(x), && (34)
\end{aligned}$$

which is not zero, in general. As for kernel density estimation, approximation to the bias and variance can be worked out, and good behaviours for m worked out (as we did for h^* for the kernel density estimator).

Fourier Basis Example

Let's consider density functions on the real line. For PDFs we know that $\int_{\mathbb{R}} f(x)\,dx = 1$ and $f(x) \geq 0$. Hence, PDFs f are absolutely integrable, $f \in L_1(\mathbb{R})$ and the Fourier transform exists.

What are the Fourier transform coefficients of a kernel DE?

$$R_h(\omega) = \int_{\mathbb{R}} \hat{f}_h(x)\exp(-2\pi i\omega x)\,dx \tag{35}$$

$$= (nh)^{-1}\sum_{j=1}^{n}\int_{\mathbb{R}} K\left(\frac{X_j - x}{h}\right)\exp(-2\pi i\omega x)\,dx. \tag{36}$$

Substituting $y = (X_j - x)$, $dy = -dx$ gives

Fourier Basis Example

$$R_h(\omega) = n^{-1} \sum_{j=1}^{n} \int_{\mathbb{R}} K_h(y) \exp(-2\pi i \omega [X_j - y])\, dy \qquad (37)$$

And $\exp\{-2\pi i\omega(X_j - y)\} = \exp(-2\pi i\omega X_j)\exp(2\pi i\omega y)$, so

$$\begin{aligned} R_h(\omega) &= n^{-1} \sum_{j=1}^{n} \exp(-2\pi i\omega X_j) \int_{\mathbb{R}} K_h(y) \exp(2\pi i\omega y)\, dy & (38) \\ &= \hat{X}(\omega)\tilde{K}_h(\omega), & (39) \end{aligned}$$

where $\hat{X}(\omega) = n^{-1}\sum_{j=1}^{n} e^{-2\pi i\omega X_j}$ & $\tilde{K}_h(\omega) = \int_{\mathbb{R}} K_h(y) e^{2\pi i\omega y}\, dy$.

K has been decoupled from the X_j. So, e.g. the $\hat{X}$ term, the discrete Fourier transform of the data, only need be computed once. The $\tilde{K}$ can often be computed in closed form. So, it's quick and easy to compute $R_h(\omega)$ for different h, e.g.

Orthogonal series estimator on $[-T, T]$

Recall the Fourier series estimator on the interval $[-T, T]$ for some $T > 0$.

We will think of our data, all of its density and estimators to live entirely within this interval.

With densities that are supported entirely on the line, e.g. $\phi(x)$ itself, we will ignore the teeny bit of density that would be outside of the interval for T large enough. T can be chosen by us and be as large as we like to make the 'excess' density negligible.

Orthogonal series on $[-T, T]$

The orthogonal series we will use on $[-T, T]$ is

$$f(x) = \sum_{\nu=-\infty}^{\infty} f_\nu e^{\frac{2\pi i\nu x}{T}}, \text{ and } f_\nu = (2T)^{-1} \int_{-T}^{T} f(x) e^{-\frac{2\pi i\nu x}{T}}, \quad (40)$$

which are just (21) and (22) for a specific example and

$$\hat{f}_\nu = \frac{1}{2nT} \sum_{j=1}^{n} \exp\left\{-\frac{2\pi i\nu X_j}{T}\right\} = T^{-1}\hat{X}(\nu/T), \quad (41)$$

which is (26) and using (39) for the definition of $\hat{X}$.

This orthogonal series estimator truncated at m is then

$$\hat{f}_m^{\mathrm{OS}}(x) = T^{-1} \sum_{\nu=-m}^{m} \hat{X}(\nu/T) e^{\frac{2\pi i\nu x}{T}}. \quad (42)$$

Kernel is modified Fourier

Fourier trans. coefs of KDE from (39) are $R_h(\omega) = \hat{X}(\omega)\tilde{K}_h(\omega)$.

So, using the inverse Fourier transform, the KDE is

$$\hat{f}_h^{\mathrm{KDE}}(x) = \int_{\mathbb{R}} R_h(\omega)\, e^{2\pi i x\omega}\, d\omega \tag{43}$$

$$= \int_{\mathbb{R}} \tilde{K}_h(\omega)\, \hat{X}(\omega)\, e^{2\pi i x\omega}\, d\omega. \tag{44}$$

In practice, the integral (44) is computed numerically on a discretized set, $\{\omega_q\}_{q=-M}^{M}$, as

$$\hat{f}_h^{\mathrm{KDE}}(x) = \sum_{q=-M}^{M} \tilde{K}_h(\omega_q)\hat{X}(\omega_q)e^{2\pi i\omega_q x}, \tag{45}$$

where typically $M > m$. Compare (42) with (45). Interesting, what is $\tilde{K}_h$?

Fourier transform of Gaussian kernel

Let, e.g., $K(x) = (2\pi)^{-1/2} \exp(-x^2/2) =$ standard normal.

Since the Fourier transform is equivalent to taking the characteristic function ($\mathbb{E}[e^{itX}]$) it is the case that

$$\begin{aligned} \tilde{K}_h(\omega) &= \int K_h(y) \exp(2\pi i \omega)\, dy && (46) \\ &= \exp(-2\pi^2 h^2 \omega^2). && (47) \end{aligned}$$

Clearly, $\tilde{K}_h(\omega)$ gets rapidly smaller as $|\omega|$ moves away from zero.

So, instead of choosing m, a cutoff to form a linear series estimator, kernel density estimation is effectively equivalent to a Fourier series estimator, but the coefficients of the estimator are caused to be rapidly downweighted.

Summary

In this lecture we learnt about

Some examples of kernel density estimates

Kernel regression and the Nadaraya Watson estimator

Local polynomial regression

Orthogonal Series Expansions

Elements of Statistical Learning: Lecture 14. Wavelets

Guy Nason[1]

Department of Mathematics
Imperial College

Wavelet Methods

Let's consider L_2 functions, to be definite. However, the methods in this section apply to much more interesting spaces, such as *Besov* spaces, which contain functions with discontinuities, discontinuities in derivatives (e.g. $|x|$) and other inhomogeneities.

(2D) Functions that are images have discontinuities, and hence wavelets perform well for tasks such as image compression, e.g. contained within the JPEG2000 standard. In fact, wavelets' good compression abilities is what makes them useful for statistics, and form the background for the popularity of sparsity-based methods.

Hence, wavelet methods are extremely general and, hence, powerful and *do not assume much*. In contrast to Fourier and kernel-type methods that assume smoothness.

Those methods typically do not (and can not) test for smoothness and it tends to, sadly, get ignored, stupidly.

Multiresolution Analysis (MRA) — informal

MRA is a framework for examining functions at different scales.

With MRA, one can "zoom-in" to examine fine scale details, or 'zoom-out' to get the big picture.

We will index scale by j.

Examination of function f at resolution j is by linear projection of f onto an *approximation space* V_j, some subspace of L_2.

Informally, define V_j to be the functions that are of resolution $\leq j$.

So, functions in V_j, for j large, contain functions that possess a high level of detail.Similarly, those in V_j for j small, contain 'coarse' looking functions.

Nason, G.P. (2008) *Wavelet Methods in Statistics with R*, Springer: Berlin, for further info on L14

MRA definition

The spaces $\{V_j\}_{j\in\mathbb{Z}}$ form a ladder as low res. spaces automatically belong to higher res. spaces.

$$\text{MRA1} \quad \cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \tag{1}$$

And

$$\text{MRA2} \quad \overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R}) \tag{2}$$

which means that the collection of $\{V_j\}$ spaces contains all of the functions we are interested in.

The functions get progressively less detailed as $j \to -\infty$ so

$$\text{MRA3} \quad \bigcap_{j\in\mathbb{Z}} V_j = \{0\}, \tag{3}$$

the zero function.

MRA definition 2

The next property links the spaces across scales.

$$\text{MRA4} \qquad f(x) \in V_j \implies f(2x) \in V_{j+1}, \forall j \in \mathbb{Z}, \tag{4}$$

and we also need

$$\text{MRA5} \qquad f(x) \in V_0 \implies f(x-k) \in V_0, \forall k \in \mathbb{Z}, \tag{5}$$

integer translates of a function in V_0 are also in V_0.

Finally, there exists $\phi(x) \in V_0$ such that $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis for V_0. This is MRA6.

If $\{V_j\}_{j\in\mathbb{Z}}$ and ϕ satisfy MRA1 to MRA6 they are said to form a MRA of L_2.

The ϕ is called the *father wavelet*. They often look like kernels, but are used differently.

Haar MRA

The Haar father wavelet is

$$\phi_H(x) = \begin{cases} 1 & x \in (0,1) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Recall inner product $< f, g >= \int f(x)g(x)\,dx$, then

$$||\phi_H||^2 =< \phi_H, \phi_H > \; = \int \phi_H^2(x)\,dx = \int_0^1 1\,dx = 1. \tag{7}$$

And

$$< \phi_H(x-\ell), \phi_H(x-m) >= \delta_{\ell,m}, \tag{8}$$

as different integer translates of $\phi_H(x)$ do not overlap, $\ell, m \in \mathbb{Z}$.

So $\{\phi_H(x-\ell)\}_{\ell\in\mathbb{Z}}$ forms an orthonormal set and we can set $V_0 = \text{span}_{\ell\in\mathbb{Z}}\{\phi_H(x-\ell)\}$. We drop the H subscript for now.

Scaling and Dilation

Dyadically scale & translate the wavelet (or any function) by

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k), \quad \forall j, k \in \mathbb{Z}. \tag{9}$$

Each set $\{\phi_{j,k}(x)\}_{k\in\mathbb{Z}}$ forms an orthonormal basis for $V_j, j \in \mathbb{Z}$.

The next plot shows $\phi(x) = \phi_{0,0}(x)$; $a\phi_{1,0}(x)$ and $b\phi_{1,1}(x)$.

Then we show for an original function $f(x)$

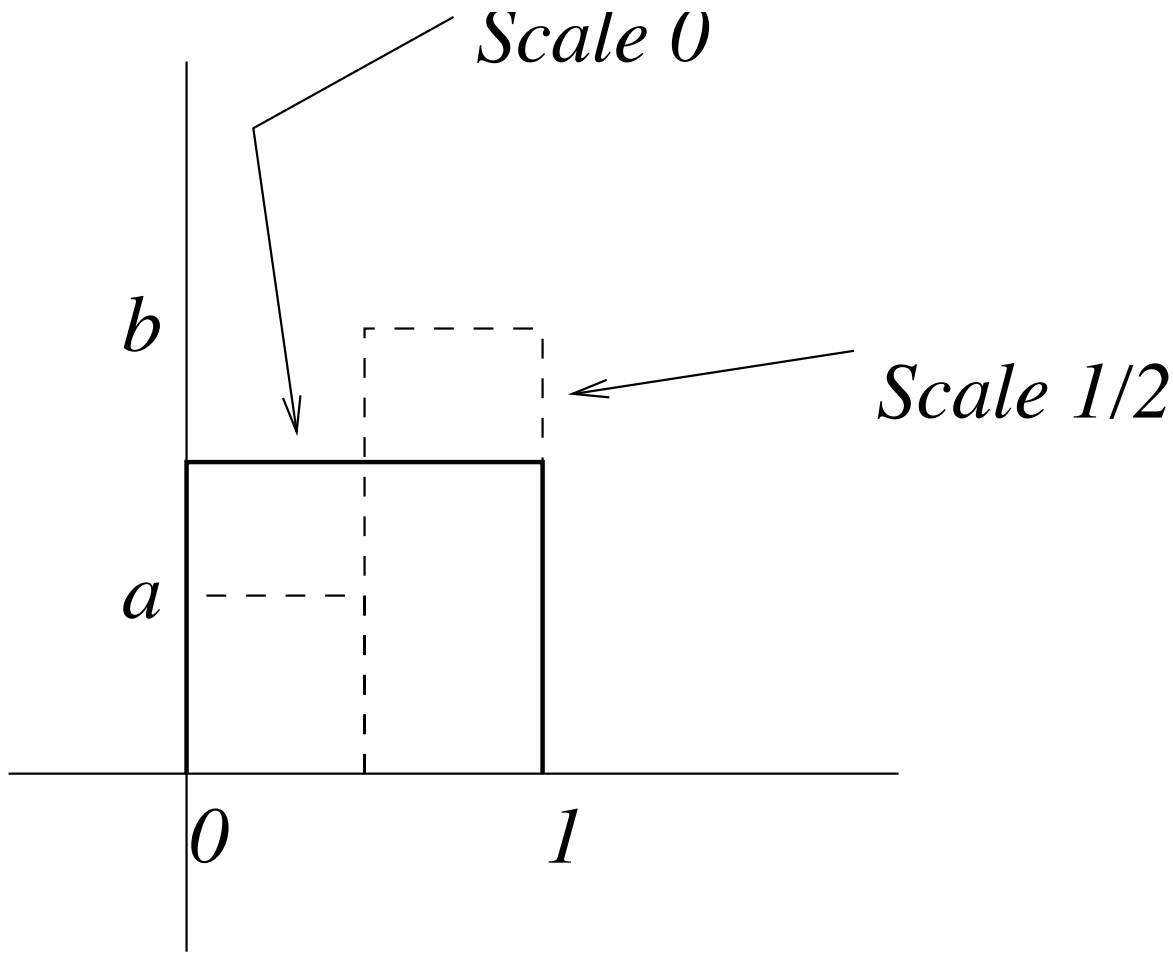
$$P_j f = \sum_{k\in\mathbb{Z}} c_{j,k}\phi_{j,k}(x), \tag{10}$$

for $j = 5, 3, 1$: the projection of $f(x)$ onto V_j, where

$$c_{j,k} = \int f(x)\phi_{j,k}(x)\, dx, \tag{11}$$

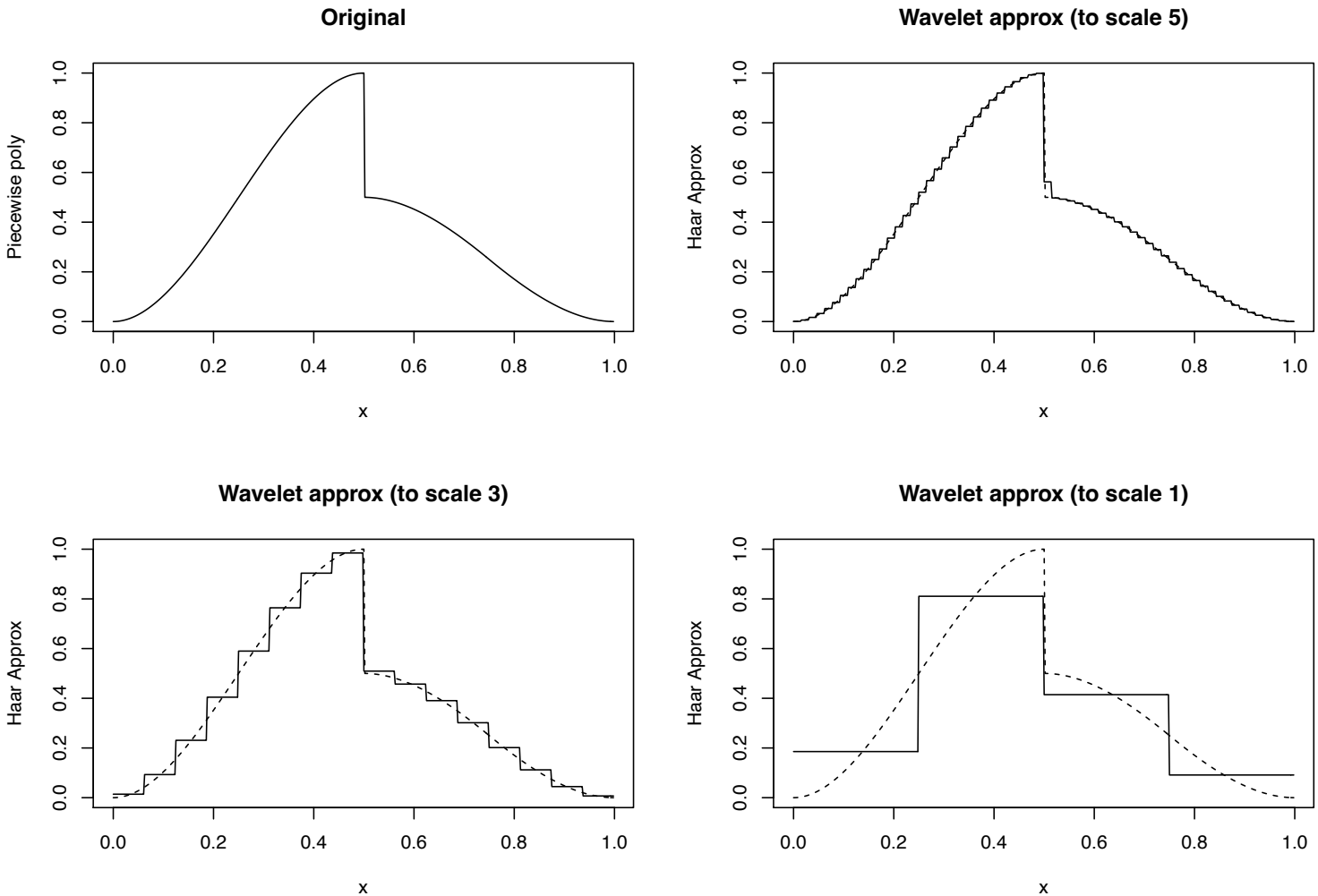
because of orthogonality.

Father wavelets at different scales



Note: $\phi(x) = \phi(2x) + \phi(2x-1)$.

Father wavelets at different scales



What disappears when going from finer scale to coarser?

Suppose our function $f(x)$ is

$$f(x) = \begin{cases} a & x \in (0, 1/2), \\ b & x \in (1/2, 1). \end{cases} \tag{12}$$

for $a, b \in \mathbb{R}$. Now on $(0, 1/2)$ we have $\phi_{1,0}(x) = \sqrt{2}\phi(2x)$, so the height of this basis function on $(0, 1/2)$ is $\sqrt{2}$.

So, $f(x) = \frac{a}{\sqrt{2}}\phi_{1,0}(x) + \frac{b}{\sqrt{2}}\phi_{1,1}(x) = f_1(x)$: scale $j = 1$ representation.And $c_{1,0} = a/\sqrt{2}, c_{1,1} = b/\sqrt{2}$.

What is $f_0(x)$, i.e. the projection of $f(x)$ onto V_0? We need to compute

$$c_{0,0} = \int_0^1 f(x)\phi(x)\,dx = \int_0^{1/2} a\,dx + \int_{1/2}^1 b\,dx = (a+b)/2. \tag{13}$$

So $f_0(x) = (a+b)\phi_{0,0}(x)/2$.

$c_{0,0}$ and $c_{1,0}, c_{1,1}$ relation & difference between V_0, V_1.

We have
$c_{0,0} = (a+b)/2 = \frac{1}{2}(\sqrt{2}c_{1,0} + \sqrt{2}c_{1,1}) = (c_{1,0} + c_{1,1})/\sqrt{2}$. So

$$
\begin{aligned}
f_0(x) - f_1(x) &= c_{0,0}\phi_{0,0}(x) - c_{1,0}\phi_{1,0}(x) - c_{1,1}\phi_{1,1}(x) && (14)\\
&= c_{0,0}\phi_{0,0}(x) - c_{1,0}\sqrt{2}\phi(2x) - c_{1,1}\sqrt{2}\phi(2x-1)\\
&= c_{0,0}\{\phi(2x) + \phi(2x-1)\}\\
&\quad -c_{1,0}\sqrt{2}\phi(2x) - c_{1,1}\sqrt{2}\phi(2x-1) && (15)\\
&= (c_{0,0} - \sqrt{2}c_{1,0})\phi(2x) + (c_{0,0} - \sqrt{2}c_{1,1})\phi(2x-1)\\
&= \{(c_{1,1} - c_{1,0})\phi(2x) + (c_{1,0} - c_{1,1})\phi(2x-1)\}/\sqrt{2},
\end{aligned}
$$

as $c_{0,0} - \sqrt{2}c_{1,0} = c_{1,0}/\sqrt{2} + c_{1,1}/\sqrt{2} - \sqrt{2}c_{1,0}$ and
$\frac{1}{\sqrt{2}} - \sqrt{2} = (1-2)/\sqrt{2} = -1/\sqrt{2}$.

Difference between V_0 and V_1: wavelet

Now define $d_{0,0} = (c_{1,1} - c_{1,0})/\sqrt{2}$, then

$$f_0(x) - f_1(x) = d_{0,0}\{\phi(2x) - \phi(2x-1)\}. \tag{16}$$

At this point, we define the *Haar mother wavelet*

$$\psi_H(x) = \begin{cases} 1 & x \in (0, \frac{1}{2}) \\ -1 & x \in (\frac{1}{2}, 1) \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Then

$$f_0(x) - f_1(x) = d_{0,0}\psi(x). \implies f_1(x) = c_{0,0}\phi(x) - d_{0,0}\psi(x). \tag{18}$$

So, the higher resolution f_1 can be obtained from the lower resolution f_0 plus some *detail* encapsulated in the wavelet and its coefficient.

Multiscale

We can call the space spanned by $\psi(x-k)$ W_0. Note that $\psi(x)$ is orthogonal to $\phi(x)$ overlay the diagrams: $V_1 = V_0 \oplus W_0$. The $\oplus$ means that the functions in the two spaces are also orthogonal. What we've seen so far is

- On $[0, 1]$
- Only from one scale to the next

We can extend the idea to multiple scales and locations to

$$f(x) = \sum_{k\in\mathbb{Z}} c_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{19}$$

for a function f, which is called a wavelet representation starting at primary resolution j_0. Note infinite scales and we can get

$$L_2 = V_{j_0} \oplus \bigoplus_{j=j_0}^{\infty} W_j \tag{20}$$

The set $\{d_{j,k}\}$ are called the wavelet coefficients.

Wavelets are orthogonal series: discrete wavelet transform

The wavelets are all mutually orthogonal, hence (19) is an orthogonal series representation.

Recall that if we had $c_{1,0}, c_{1,1}$ we can obtain the coarser father wavelet coefficient and the associated wavelet coefficient by

$$c_{0,0} = (c_{1,1} + c_{1,0})/\sqrt{2} \tag{21}$$

$$d_{0,0} = (c_{1,1} - c_{1,0})/\sqrt{2}. \tag{22}$$

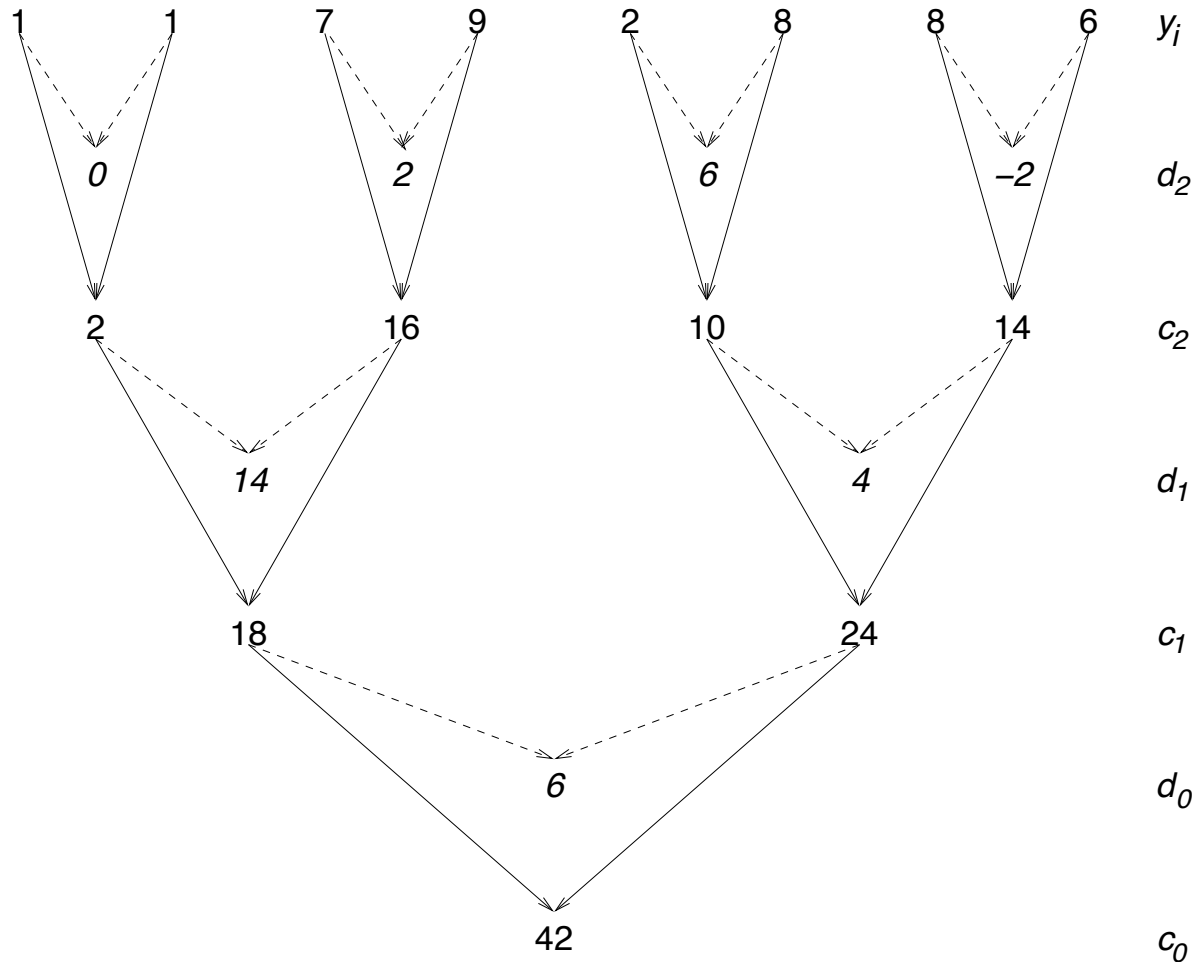
This can be generalised to obtain:

$$c_{j-1,k} = (c_{j,2k+1} + c_{j,2k})/\sqrt{2} \tag{23}$$

$$d_{j-1,k} = (c_{j,2k+1} - c_{j,2k})/\sqrt{2} \tag{24}$$

Suppose we start with sequence $1, 1, 7, 9, 2, 8, 8, 6$ as $c_{3,i} = y_i$ then the coarser coefficients (without the $\sqrt{2}$ scaling) are:

Example of Haar wavelet transform: Pyramid algorithm



There are 7 d and 7 c operations for 8 data. Algorithm is $\mathcal{O}(n)$ FAST!

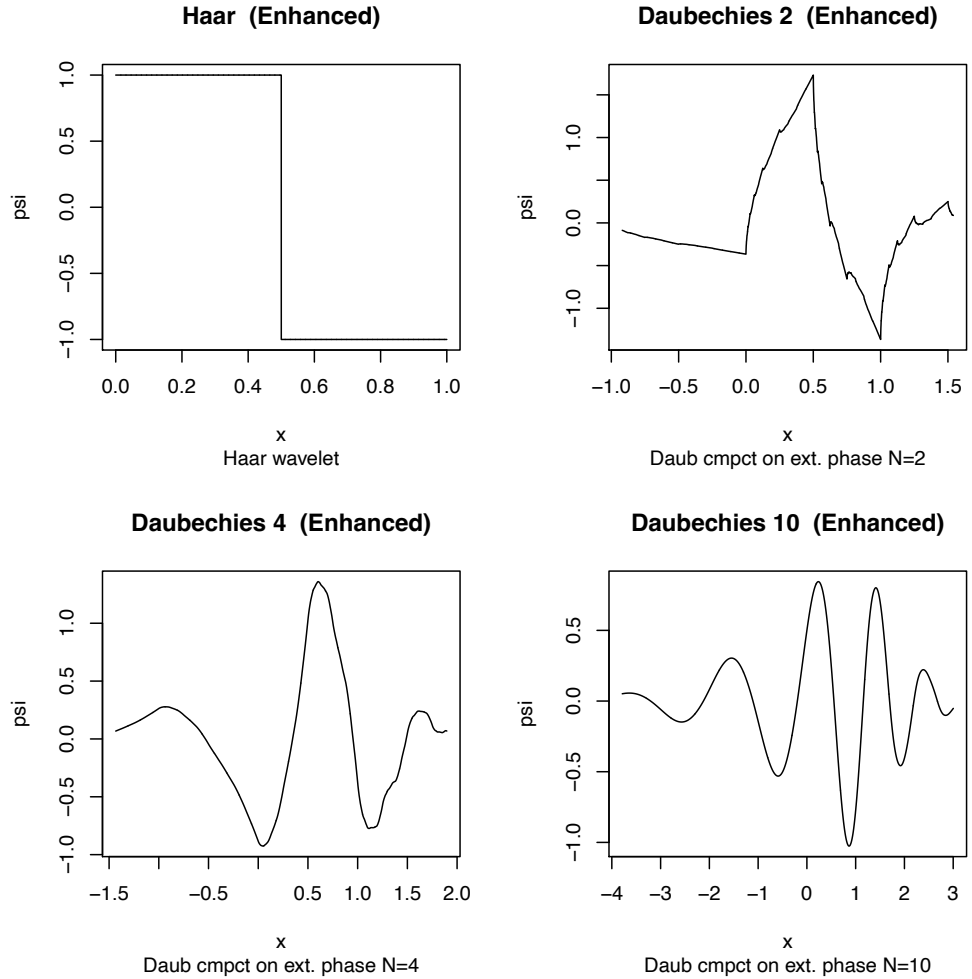
Matrix Representation

$$W = \begin{bmatrix} \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 1/2 & 1/2 & -1/2 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & -1/2 & -1/2 \\ \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & \sqrt{2}/4 & -\sqrt{2}/4 & -\sqrt{2}/4 & -\sqrt{2}/4 & -\sqrt{2}/4 \end{bmatrix}, \tag{25}$$

If the input data is $y = c(1, 1, 7, 9, 2, 8, 8, 6)^T$, then the wavelet coefficients can be written as $d = Wy$.

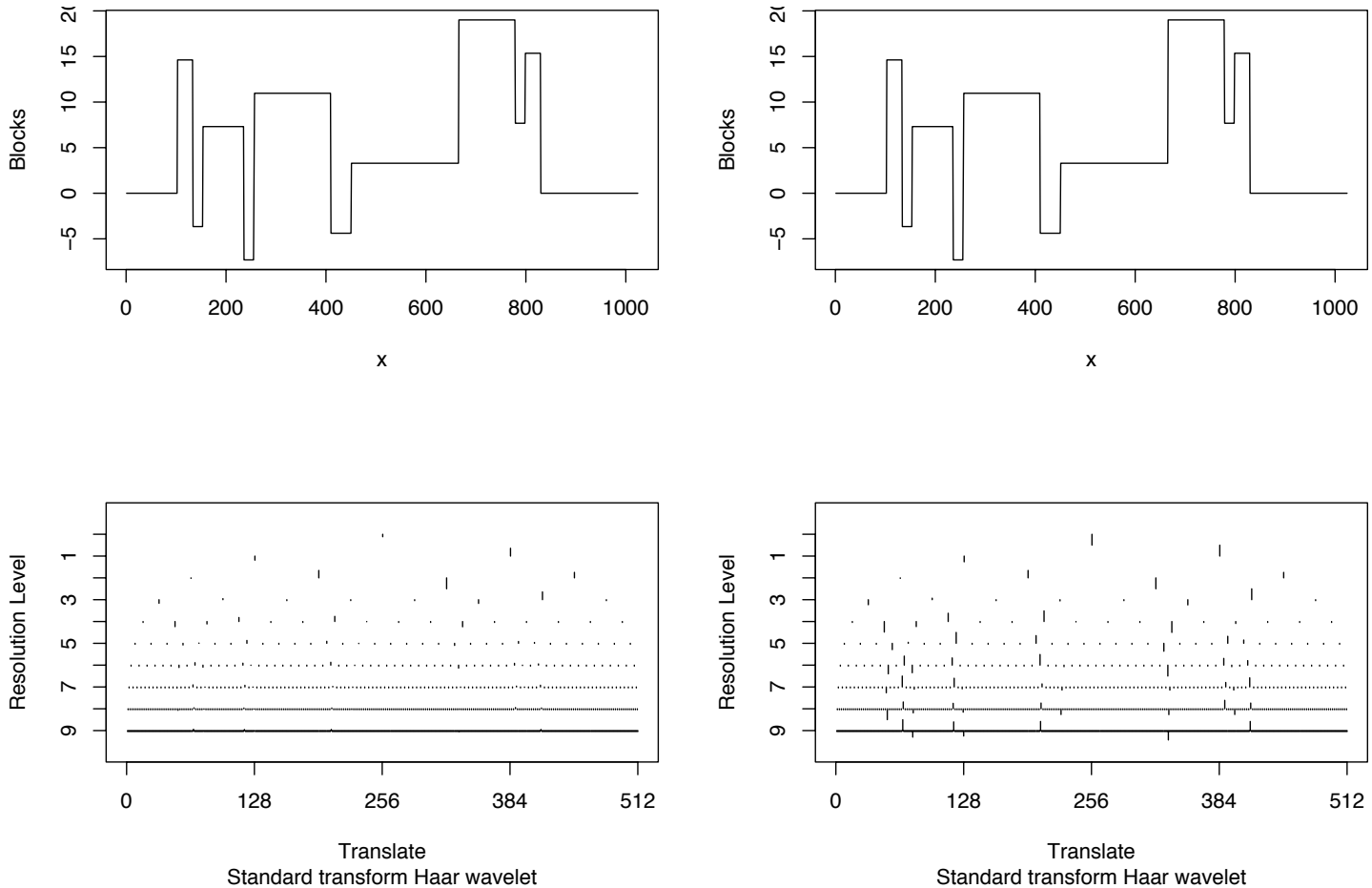
Although, the pyramid algorithm is used in practice.

Other mother wavelets



Formula (23) is generalised to $c_{j-1,k} = \sum_\ell h_{\ell-2k} c_{j,\ell}$, and similar one involving $g_{\ell-2k}$ for ds.

Haar wavelet transform of Blocks signal



Left: coefficients on same scale; *Right:* scaled according to level. Note horizontal alignment between coefficients and original

Vanishing moments

You may have noticed that the coefficients near to smooth parts of the function were zero.

This is no accident.

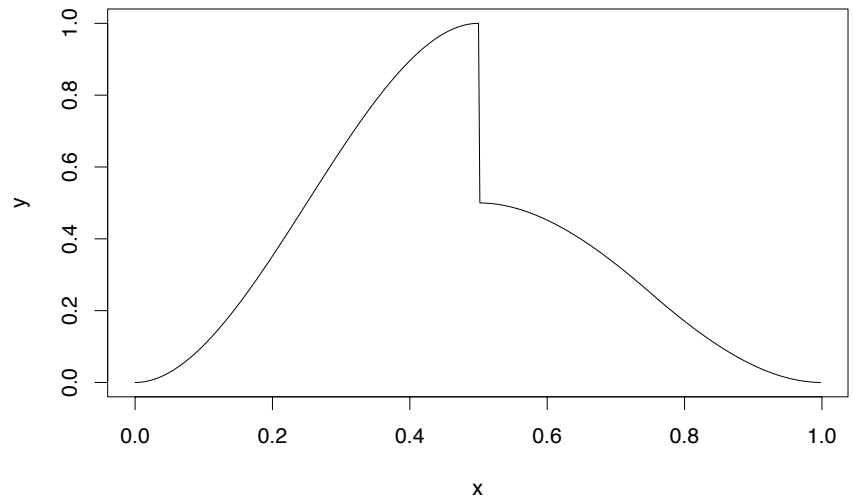
A wavelet has *m vanishing moments* if it satisfies

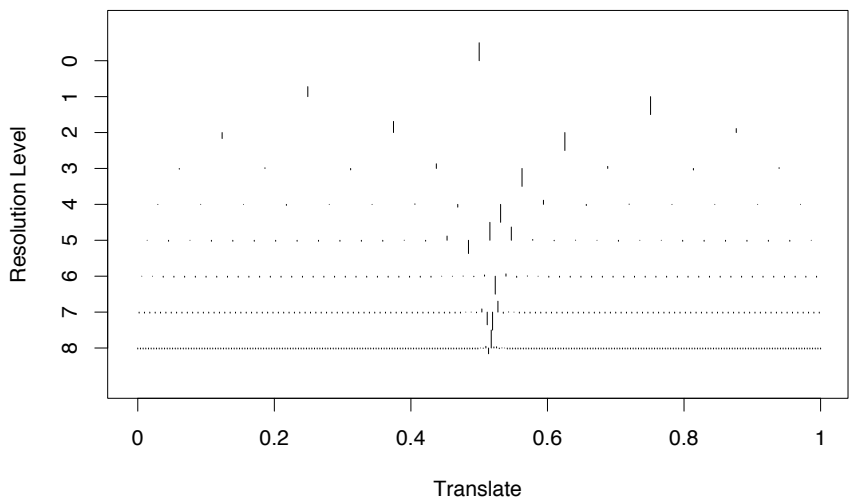
$$\int x^{\ell} \psi(x)\, dx = 0, \tag{26}$$

for $\ell = 0, \ldots, m-1$ (with some other conditions).

This is part of the secret of wavelets' success. It essentially means that the wavelet coefficients of the smooth parts of functions is zero, and mostly the non-zero coefficients relate to inhomogeneities.

Piecewise polynomial and its wavelet coefficients





Sparsity

Wavelet transforms of a wide variety of functions are very sparse.

```
library("wavethresh") # An R wavelet library
length(example.1()$y)
[1] 512
sum(abs(wd(example.1()$y, filter.number=4,
    family="DaubExPhase")$D) < 1e-10)
[1] 418
```

The `example.1()$y` function generates the piecewise polynomial. It has 512 entries.

`wd` computes the wavelet transform (here with 4 vanishing moments, as piecewise polynomial has cubic terms). The `$D` extracts the wavelet coefficients.

Effective number of zeroes is 418 — compressed 512 to 94.

Sparsity — 2

Wavelet transforms have the ability to sparsify signals.

This is why they are used in compression applications.

There are ‘variants’ especially for images — e.g. curvelets, ridgelets, etc.

However, it makes them REALLY useful for statistical learning.

Wavelet Shrinkage

Suppose our model is:

$$y_i = f_i + \epsilon_i \quad i = 1, \ldots, n, \tag{27}$$

where we observe $\{y_i\}_{i=1}^n$. We assume that ϵ_i are a set of IID random variables with mean zero and variance of σ^2 (so the ϵ vector has covariance matrix of $\sigma^2 I_n$) and the $\{f_i\}_{i=1}^n$ is an *unknown* function that we want to estimate from the y data.

The wavelet transform of the noise $e = W\epsilon$ has the following interesting properties:

$$\mathbb{E}(e) = W\mathbb{E}(\epsilon) = 0 \tag{28}$$

and $\text{var}(e) = \mathbb{E}(ee^T) =$

$$W\mathbb{E}(\epsilon\epsilon^T)W^T = W\,\text{var}(\epsilon)W^T = \sigma^2 W I_n W^T = \sigma^2 WW^T = \sigma^2 I_n. \tag{29}$$

Note: W orthogonality, other way round than previously.

Wavelet Shrinkage — 2: Useful properties

Applying W to the (27), $w = Wy, d = Wf$ gives

$$w = d + e, \tag{30}$$

The noise, e, is uncorrelated (and independent if ϵ is Gaussian) and hence the noise gets spread 'evenly' across all coefficients.

The d is sparse, as it's a wavelet transform of a (sampled) function.

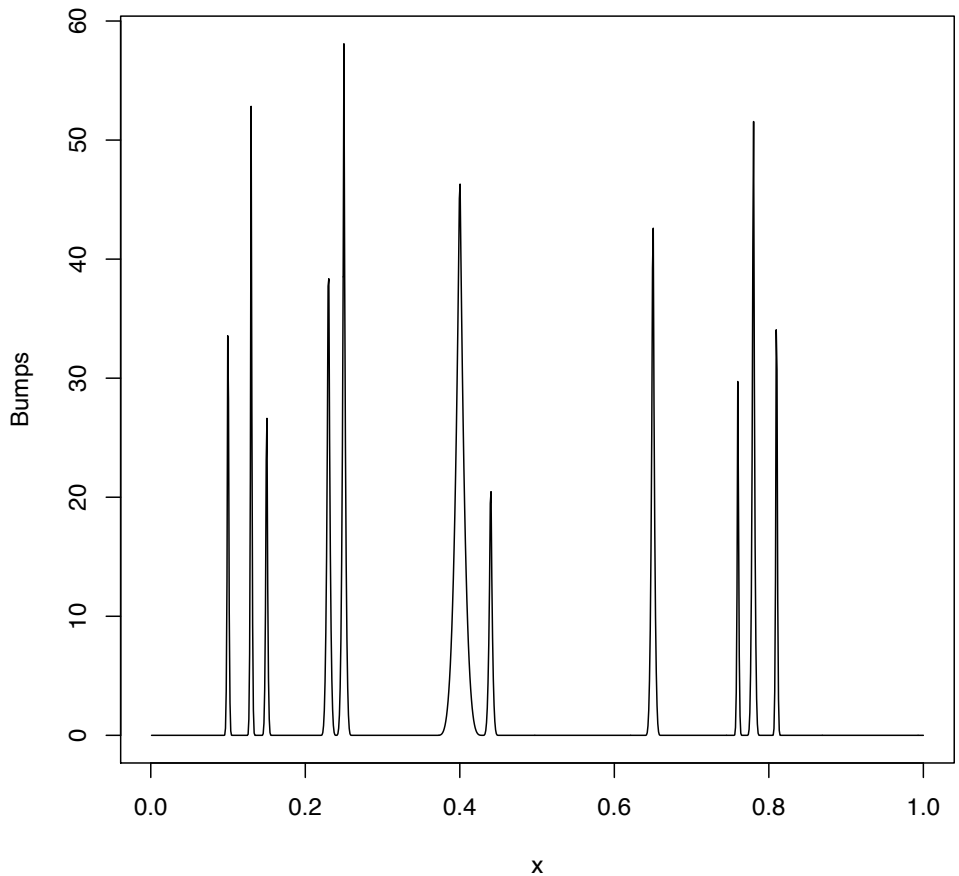
Due to Parseval's theorem, and that wavelets are orthogonal series we have $||d|| = ||f||$.

So, f gets squeezed into much fewer locations, but has the same energy.

Hence, the signal to noise ratio for signal coefficients is *MUCH IMPROVED*. Strategy for denoising: get rid of small coefficients.

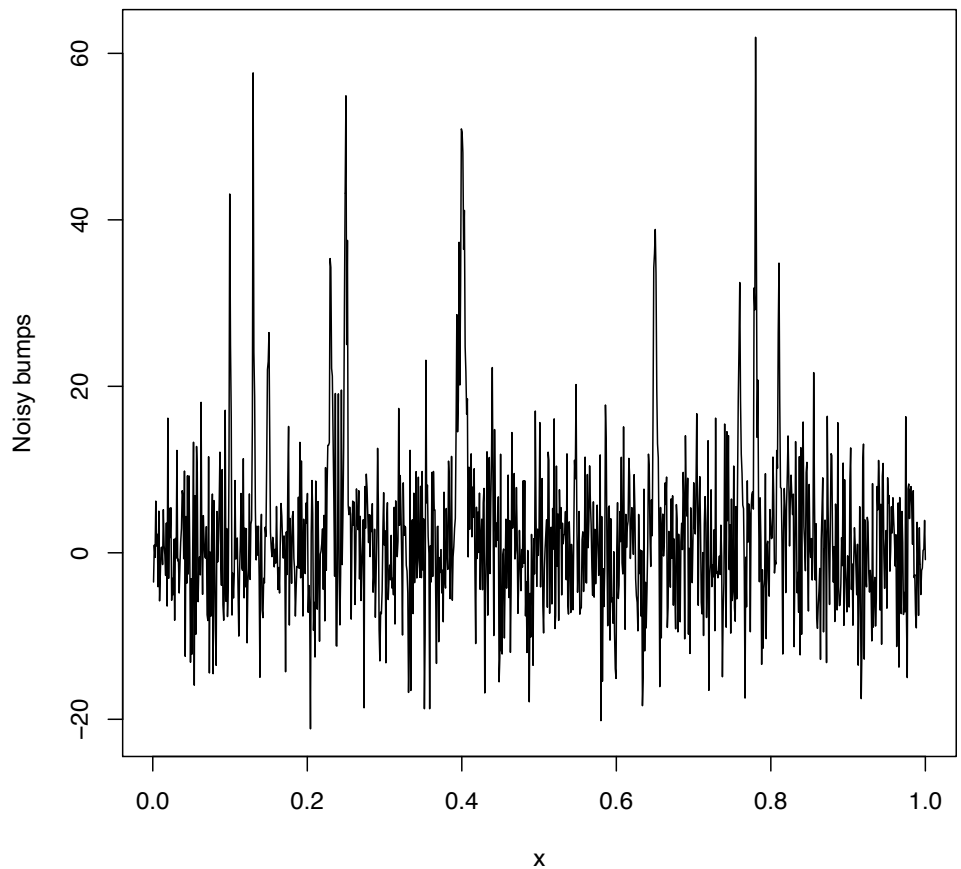
Bumps signal (simulacrum of NMR signal)

```
v <- DJ.EX()
x <- (1:1024)/1024  # X coordinates
plot(x, v$bumps, type="l", ylab="Bumps")
```



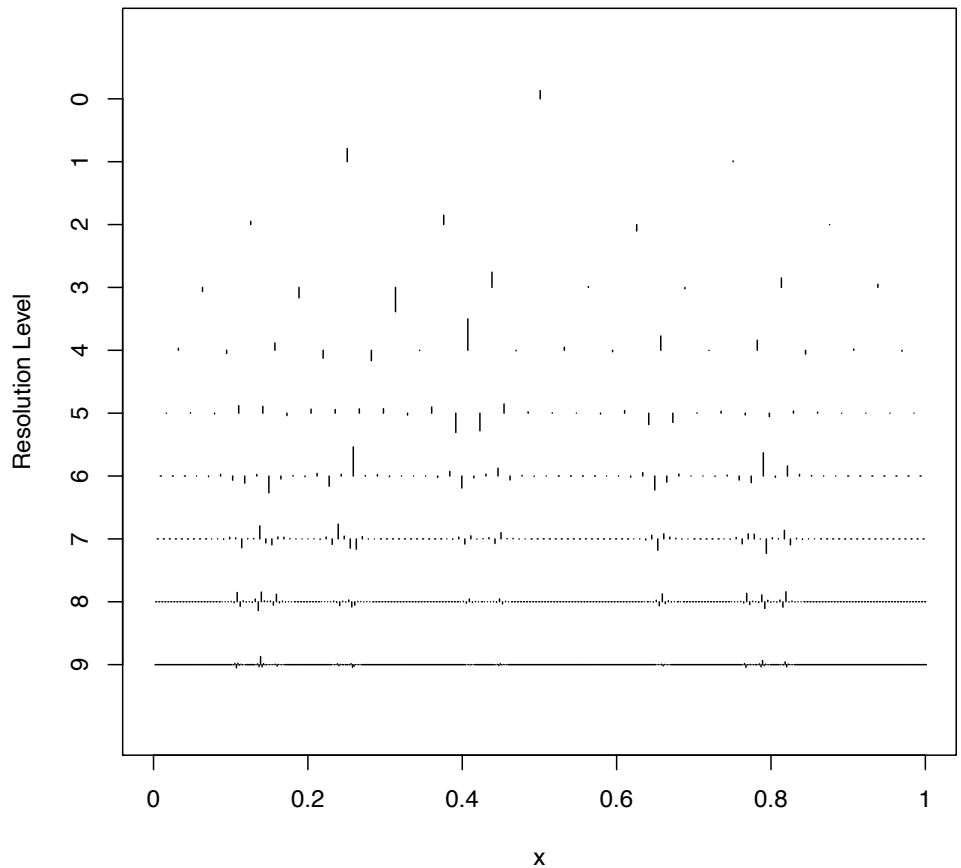
Noisy bumps

```
set.seed(100)
ssig <- sd(v$bumps)
e <- rnorm(1024, mean=0, sd=ssig) # SNR=1
y <- v$bumps + e
plot(x, y, type="l", ylab="Noisy bumps")
```



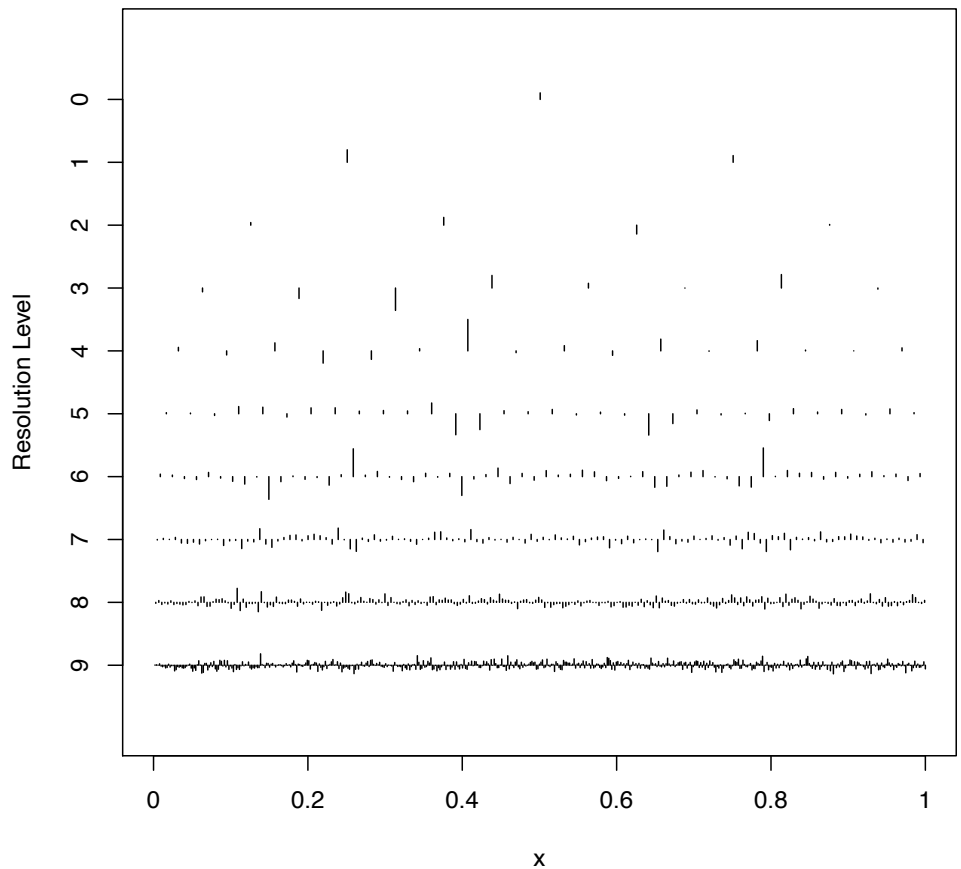
Wavelet transform of Bumps

```
xlv <- seq(from=0, to=1.0, by=0.2)
bumpswd <- wd(v$bumps)
plot(bumpswd, main="", sub="", xlabvals=xlv*512,
    xlabchars=as.character(xlv), xlab="x")
```

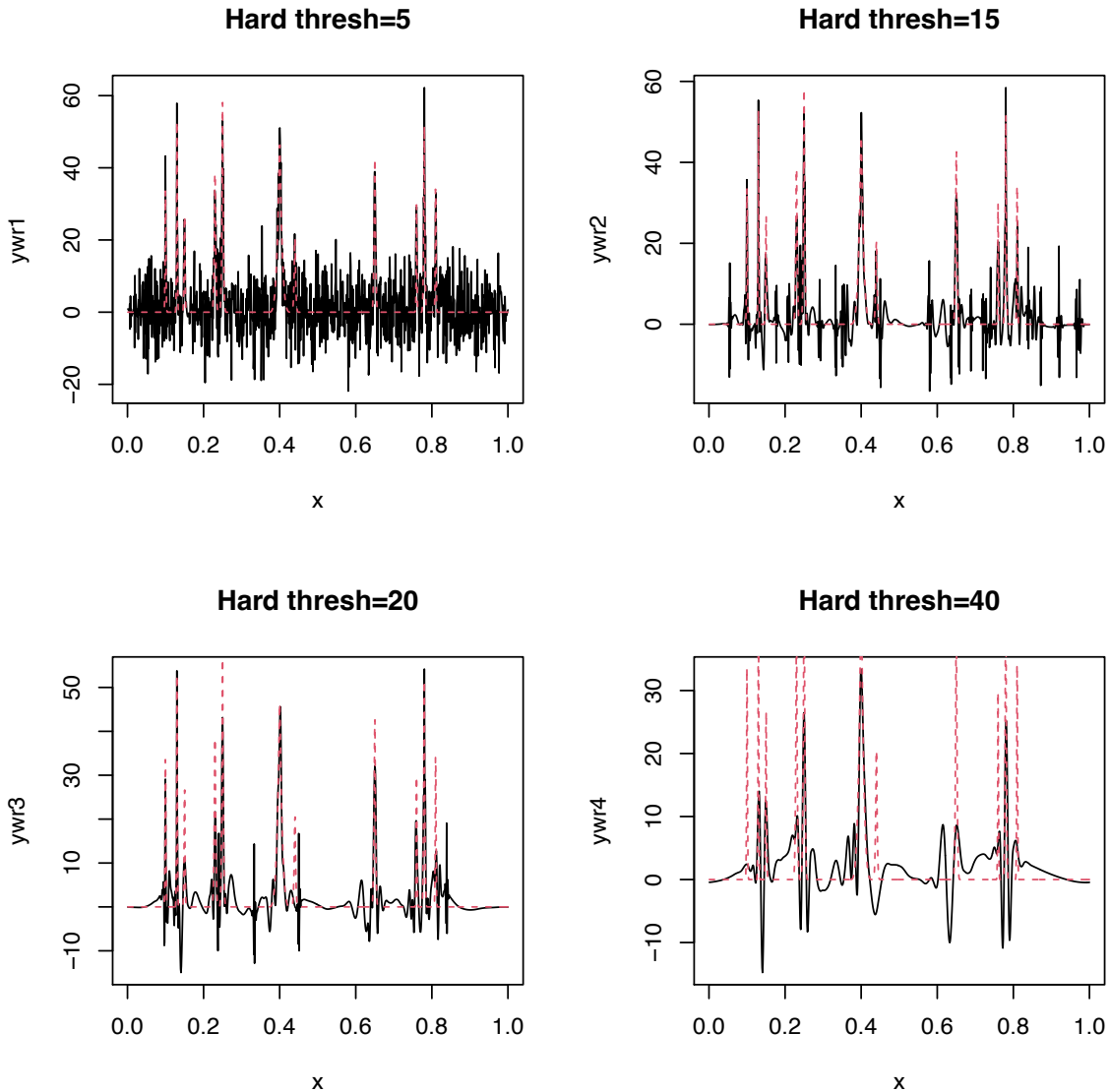


Wavelet transform of Noisy Bumps

```
ywd <- wd(y)
plot(ywd, main="", sub="", xlabvals=xlv*512,
    xlabchars=as.character(xlv), xlab="x")
```

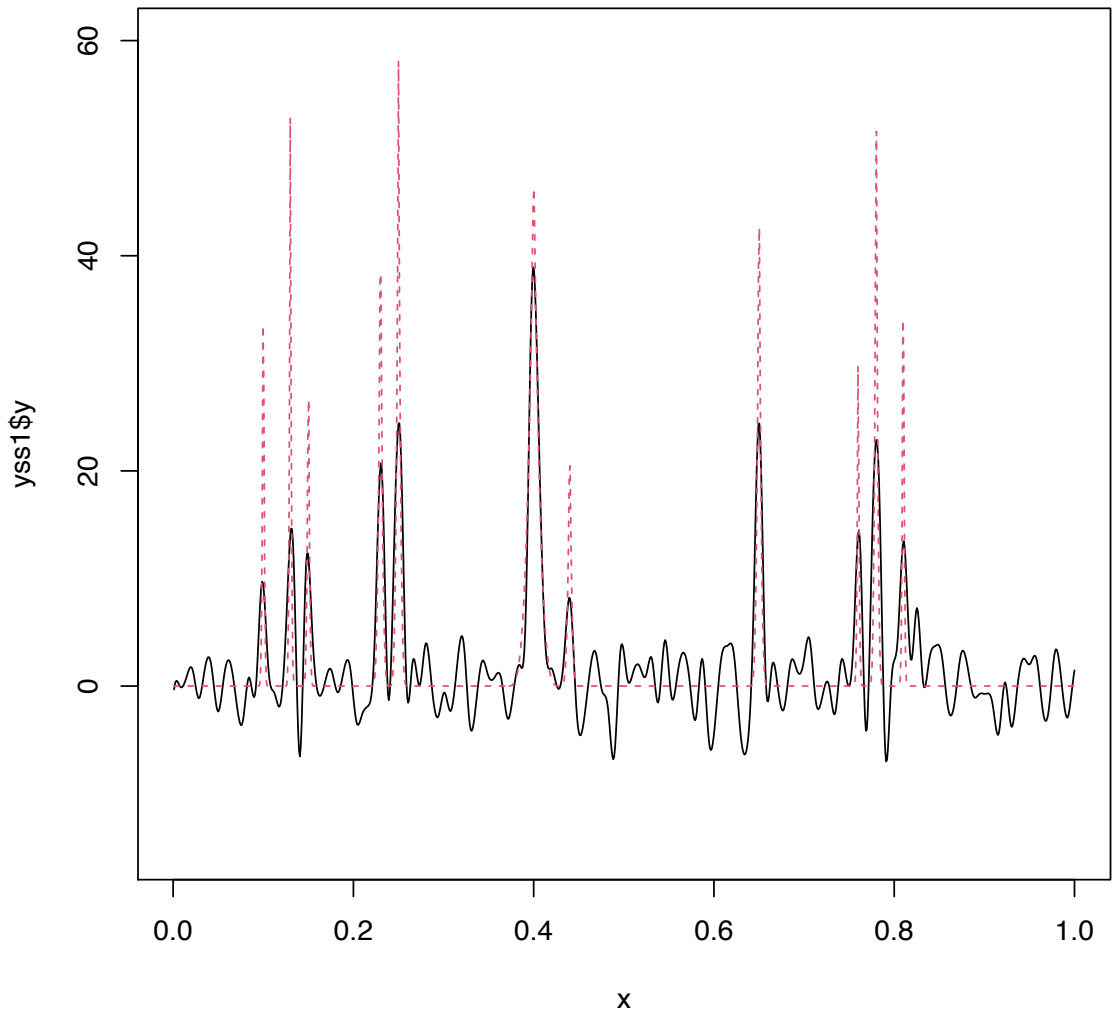


Inverted Thresholded Estimates of Bumps



Cross-validated smoothing spline estimate of Bumps

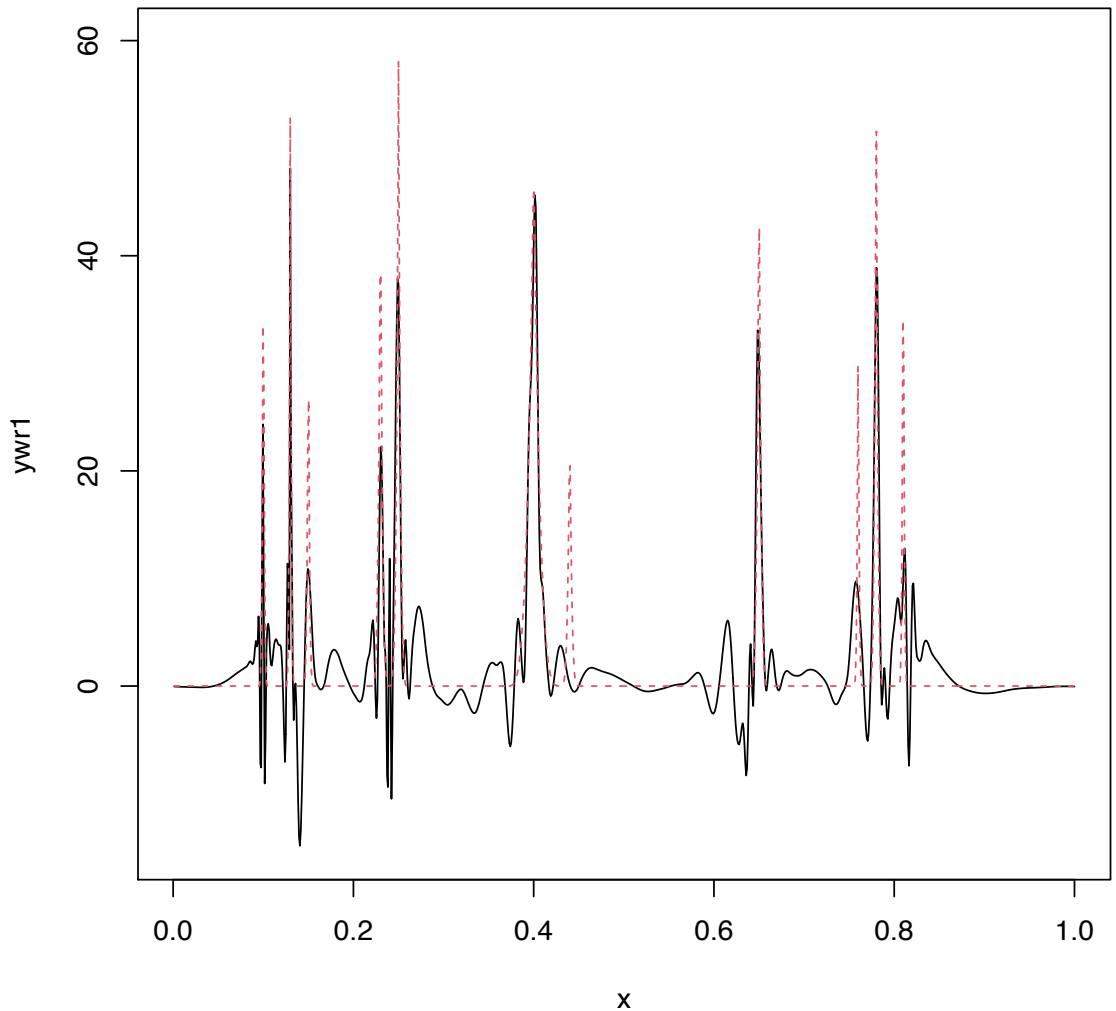
```
yss1 <- smooth.spline(x=x, y=y)
plot(x, yss1$y, type="l", ylim=c(-15,60))
lines(x, v$bumps, col=2, lty=2)
```



Mean-squared error is 17.39.

Cross-validated wavelet shrinkage estimate of Bumps

```
ywr1 <- wr(threshold(ywd, policy="cv", type="hard"))
plot(x, ywr1, type="l", ylim=c(-15,60))
lines(x, v$bumps, col=2, lty=2)
```



Mean-squared error is 15.27

Thresholding Types

To "get rid" of small coefficients, we can use hard thresholding:

$$T_{\text{hard}}(w, \lambda) = w\mathbb{I}(|w| > \lambda). \tag{31}$$

Or soft thresholding

$$T_{\text{soft}}(w, \lambda) = \text{sgn}(w)(|w| - \lambda)\mathbb{I}(|w| > \lambda), \tag{32}$$

where w is the noisy wavelet coefficient.

Bayesian Wavelet Shrinkage

For many functions, their wavelet transforms are sparse.

So their coefficients have a 'size' or are precisely zero.

Actually "many" in this context really means 'all functions we'll probably want to think about'.

So, it is natural to think of the following as a prior distribution for the wavelet coefficients

$$d_{j,\cdot} = \gamma_j N(0, \tau_j^2) + (1 - \gamma_j)\delta_0(x), \tag{33}$$

where $\delta_0(x)$ is a point mass (Dirac delta) at zero and γ_j is a Bernoulli random variable with parameter p_j. [δ_0 can be thought of as a 'density' with distribution function $H_\delta(u) = \mathbb{I}(u > 0)$. H is known as the Heaviside step function.]

Bayesian Wavelet Shrinkage — 2

The likelihood comes from $w = d + e$ and, if e assumed Gaussian we have $w|d \sim N(d, \sigma^2)$.

It can be shown that

$$F(d|w) = r\Phi\left\{\frac{d - w\nu^2}{\sigma\nu}\right\} + (1 - r)\mathbb{I}(d > 0), \qquad (34)$$

where Φ is the Gaussian CDF, $\nu^2 = \tau^2(\sigma^2 + \tau^2)^{-1}$ and $r \in (0, 1)$.

The form of $F(d|w)$ is exactly that of the prior (33), (the Berger-Müller prior). Distribution on coefficients $\implies$ distribution on functions $\implies$ Bayesian nonparametrics.

So, estimated coefficients are either exactly zero, or something not zero. Later versions of wavelet shrinkage replace the Gaussian in (33) by a heavy-tailed distribution, such as the Student's t.

Summary

In this lecture, we covered:

Multiresolution Analysis MRA

- Wavelets mother, father, Haar, Daubechies

Discrete Wavelet Transform the pyramid algorithm.

Vanishing moments sparsity

Bumps and Blocks function

Wavelet shrinkage

Bayesian wavelet shrinkage

Elements of Statistical Learning: Lecture 15. Exploratory Projection Pursuit

Guy Nason[1]

Department of Mathematics
Imperial College

Background to Principal Components

In lecture 3 we came across the SVD of a $n \times p$ data matrix X

$$X = UDV^T, \tag{1}$$

where U and V are $n \times p$ and $p \times p$ orthogonal matrices, respectively and D a $p \times p$ diagonal matrix with entries $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$.

The $p \times p$ sample covariance matrix $S = n^{-1}X^TX$ of the centred data matrix could be written

$$S = n^{-1}X^TX = n^{-1}VD^2V^T, \tag{2}$$

an eigendecomposition with eigenvectors v_j (columns of V), called *principal components*, with eigenvalues d_j^2.

Principal Components Review

We can project the data matrix X onto the principal components forming

$$z_{n\times 1} = X_{n\times p} v_{p\times 1}. \tag{3}$$

We also showed that the variance of the new projected one-dimensional data set z_1 was d_1^2/n, and the variance of the data set projected onto v_2, i.e. z_2 was d_2^2/n in the direction v_2, which is orthogonal to v_1, as they are eigenvectors. This continues by projecting onto the third principal component, v_3 to get one-dimensional set z_3, ..., until we get to the pth one.

Suppose we project onto an arbitrary p-vector a, i.e. $y_{n\times 1} = X_{n\times p} a$? What is the variance of this new data set y?

Projection onto arbitrary vector

First, we can show that y has zero mean because

$$n^{-1} y^T \mathbf{1} = n^{-1} a^T X^T \mathbf{1} = 0, \tag{4}$$

as X is centred.

The variance of y is thus

$$S_y(a) = n^{-1} y^T y = n^{-1} a^T X^T X a = a^T S a, \tag{5}$$

where S is the sample variance matrix of X.

We can make S_y as big as we like, just by choosing entries of a big. This is not a well-posed problem.

So, the question is what a makes the variance biggest, for when a is a unit vector. I.e. $a^T a = 1$? The answer is $a = v_1$ and, since v_1 is the first column of an orthogonal matrix it is of unit length!

Optimisation formulation of PCA

First PC is solution to optimisation

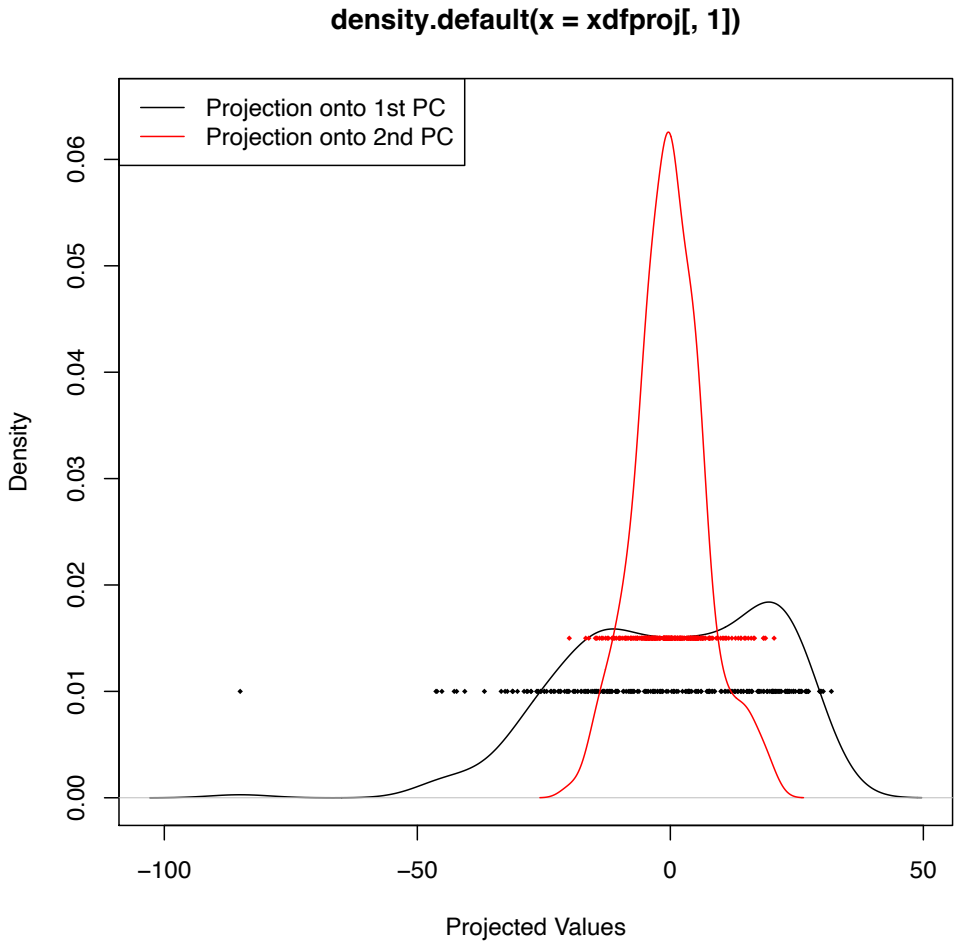
$$\max_{a} S_y(a) \text{ subject to } a^T a = 1. \tag{6}$$

The ith PC, $i = 2, \ldots, p$ is solution to

$$\max_{a} S_y(a) \text{ subject to } a^T a = 1 \text{ and } a^T v_j = 0, \tag{7}$$

for all $j = 1, \ldots, i-1$, where v_j is the jth PC.

Picture of Projected PC data



Is large variance alway interesting?

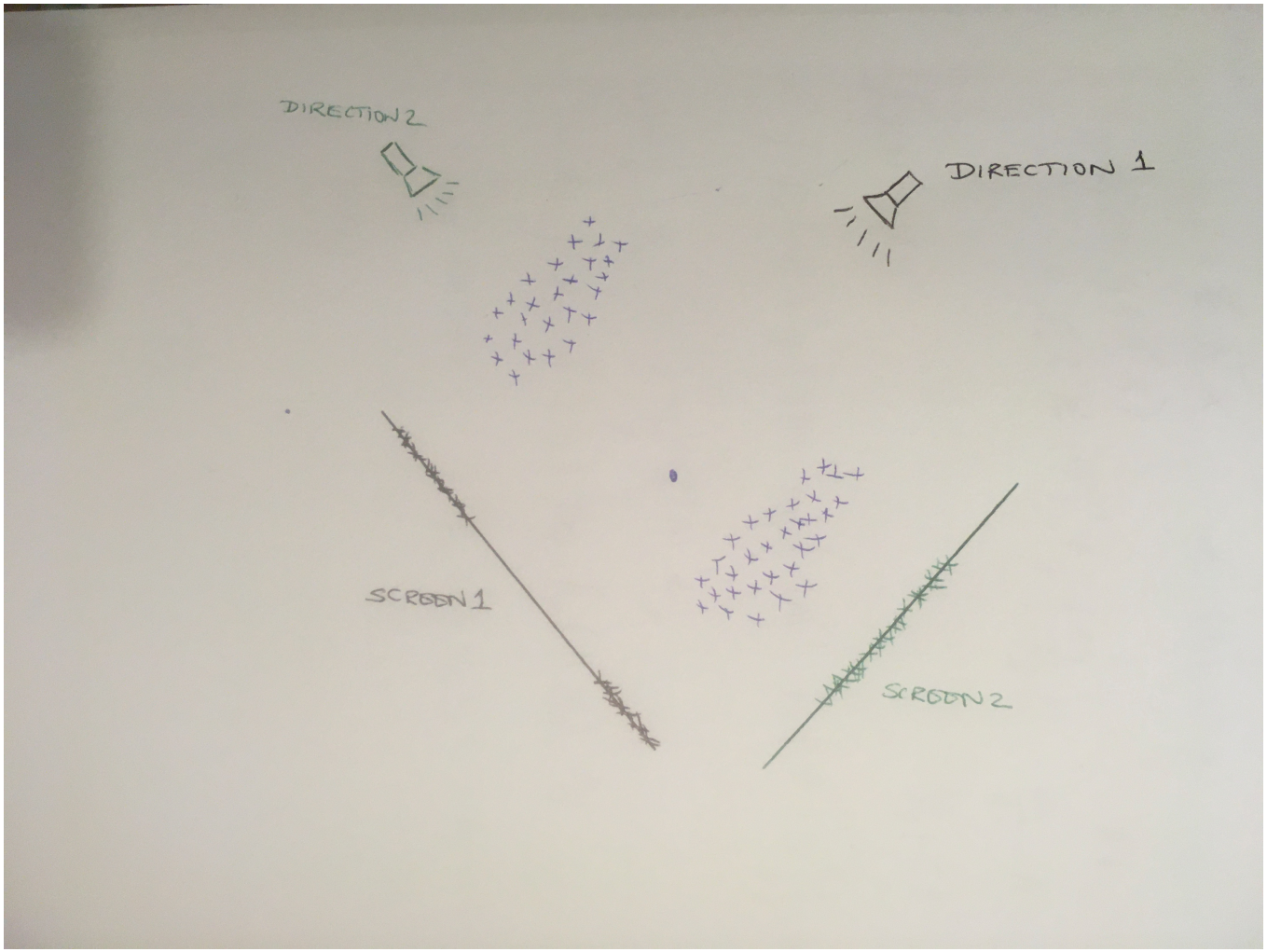
Variance?

Is large variance always interesting?

In previous picture, black has two humps, but not clearly separated.

So, big doesn't mean interesting. If the projected data was split into two or more clusters, that would be more interesting.

Exploratory Projection Pursuit



What is interesting?

A Gaussian distribution is boring.

It only has one mode.

We'd like to get away from a boring Gaussian, e.g. find something that is multimodal.

Kullback-Leibler Divergence and Entropy

Let $f(x), g(x)$ be two densities on $\mathbb{R}$.

The *Kullback-Leibler Divergence* of g from f is

$$D_{\mathrm{KL}}(g||f) = \int_{\mathbb{R}} g(x) \log \left\{ \frac{g(x)}{f(x)} \right\} dx. \tag{8}$$

Easy to see that $D_{\mathrm{KL}}(g||f) \geq 0$, as f, g are densities (which are ≥ 0) and $D_{\mathrm{KL}}(g||f) = 0$ if and only if $f = g$ almost everywhere.

The *entropy* of density g is defined to be

$$H(g) = -\int_{\mathbb{R}} g(x) \log\{g(x)\} dx \tag{9}$$

Entropy measures 'interestingness'

We aren't interested in large variance, PCA can do that. So, we only consider densities with the same variance, e.g. 1.

Also, our data matrix is usually centred, so we consider only densities with zero mean.

We find the Gaussian $N(0, \sigma^2)$ to be the most boring. So, let's find a criterion that is maximised by a Gaussian and then find densities that minimise that criterion.

Gaussian maximises entropy

Let $f(x)$ be the density of a $N(0, \sigma^2)$ variable.

Let $g(x)$ be any other density with mean zero and variance of σ^2.

Then

$$\begin{aligned} 0 &\leq D_{\mathsf{KL}}(g||f) && (10)\\ &= \int_{\mathbb{R}} g(x) \log\left\{\frac{g(x)}{f(x)}\right\} dx && (11)\\ &= \int_{\mathbb{R}} g(x) \log\{g(x)\}\, dx - \int_{\mathbb{R}} g(x) \log\{f(x)\}\, dx && (12)\\ &= -H(g) - \int_{\mathbb{R}} g(x) \log\{f(x)\}\, dx. && (13) \end{aligned}$$

Recall $f(x) = (2\pi\sigma^2)^{-1/2} \exp\{-x^2/(2\sigma^2)\}$,

Gaussian maximises entropy — 2

$$\log\{f(x)\} = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{x^2}{2\sigma^2}. \tag{14}$$

So the second integral in (13) is

$$-\frac{1}{2}\log(2\pi\sigma^2)\int_{\mathbb{R}} g(x)\,dx - \frac{1}{2\sigma^2}\int_{\mathbb{R}} x^2 g(x)\,dx = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}. \tag{15}$$

The entropy of our Gaussian is

$$\begin{aligned} -H(f) &= \int_{\mathbb{R}} f(x)\log\{f(x)\}\,dx && (16)\\ &= -\frac{1}{2}\log(2\pi\sigma^2)\int_{\mathbb{R}} f(x)\,dx - \frac{1}{2\sigma^2}\int_{\mathbb{R}} x^2 f(x)\,dx && (17)\\ &= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}. && (18) \end{aligned}$$

Gaussian maximises entropy — 3

So,

$$0 \leq -H(g) + H(f) \implies H(f) \geq H(g), \tag{19}$$

for arbitrary density $g(x)$ with variance of σ^2.

So, $N(0, \sigma^2)$ maximises the entropy.

(Centring and) Sphering

We have already mention centring a data matrix (translating so its centroid is the origin) — assume X centred.

Sphering, transforms matrix so that its variance is the identity.

Recall $S = n^{-1}X^TX$. Calculate $R = S^{-1/2}$ (e.g. using method we used for classical scaling) and then form sphered data matrix $W = XR$.

The variance matrix of W is then

$$S_W = n^{-1}W^TW = n^{-1}R^TX^TXR = R^TSR = I_p. \qquad (20)$$

Principal components then has *nothing to use* as

$$\text{var}(Wa) = a^TS_Wa = a^Ta = 1, \qquad (21)$$

i.e. the variance is the same in all directions.

Summary of process and optimisation

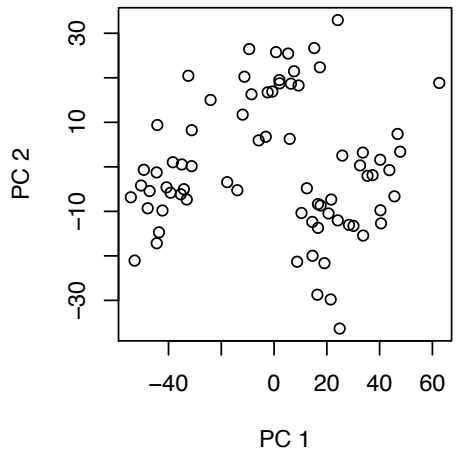
1. Start with centred and sphered data matrix, W.
2. Choose initial unit projection vector, a.
3. Form projected data $u_a = Wa$.
4. Form density estimate, $\hat{f}_{U,a}(u)$, from $u_1, \ldots, u_n$.
5. Compute entropy $H\{\hat{f}_{U,a}(u)\}$.
6. Solve $\text{argmin}_{a:a^T a=1} H\{\hat{f}_{U,a}(u)\}$.

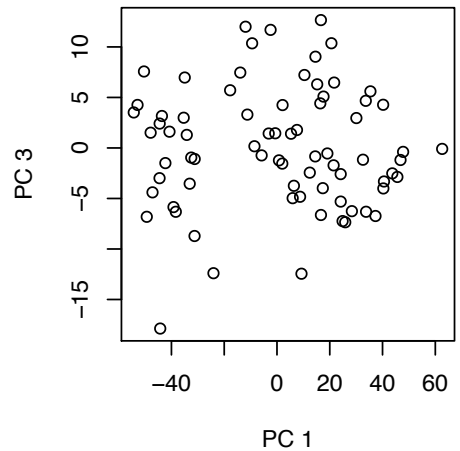
It is often possible to derive $\frac{\partial H\{\hat{f}_{U,a}(u)\}}{\partial a_j}$ analytically.

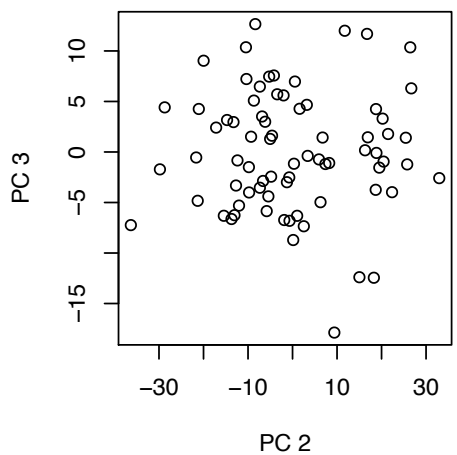
All other steps are usually carried out numerically.

Can build up multidimensional solutions, by then optimising over unit b that is orthogonal to a, etc.

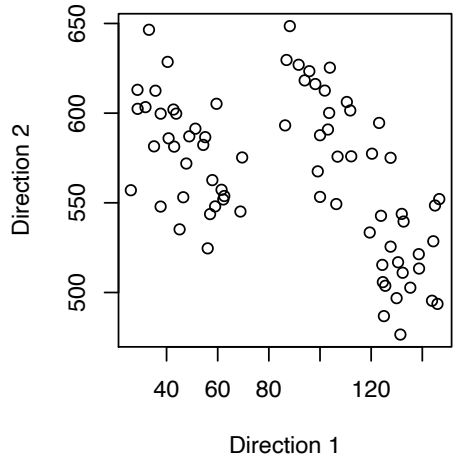
Principal Components of Beetle

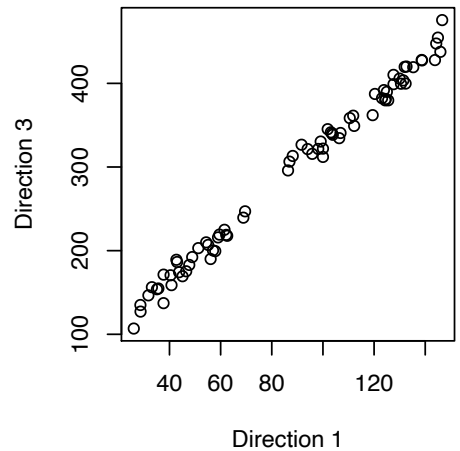


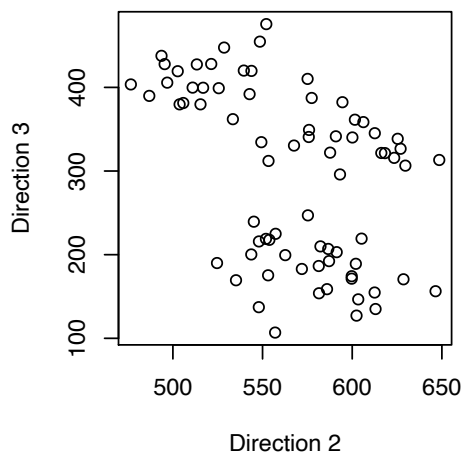




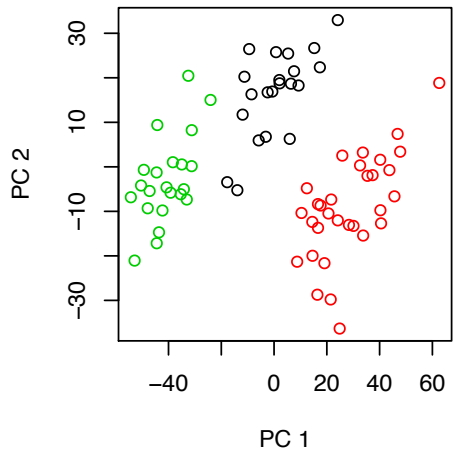
ThreeD Projection Pursuit Solution

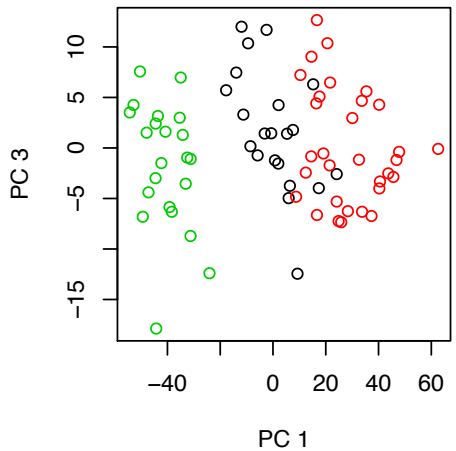


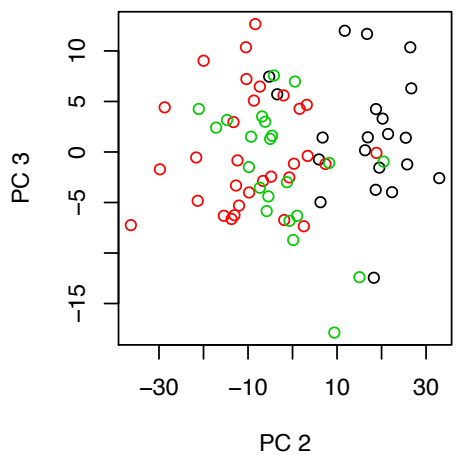




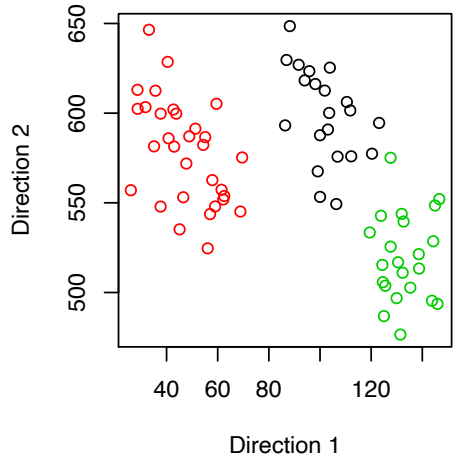
PCA Beetle with labels

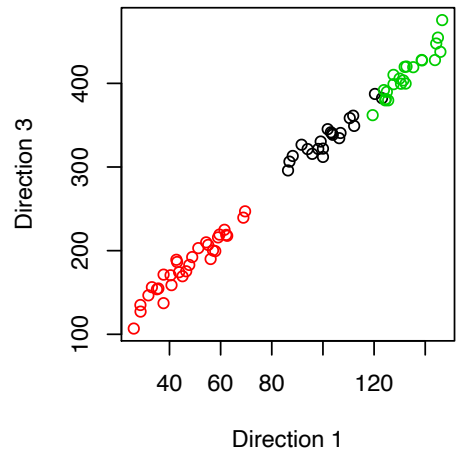


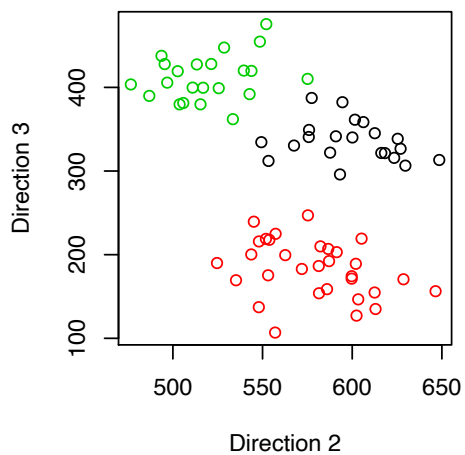




3D PP solution with labels







Remarks

Have to choose starting value of a.

This value then gets changed by the optimiser as it tries to move to a minimum.

Often, choose initial value of a using random values.

However, final solution then depends on initial random starting position.

If initial value is in very uninteresting position, then final view can be uninteresting also.

Optimiser often gets stuck in local minima.

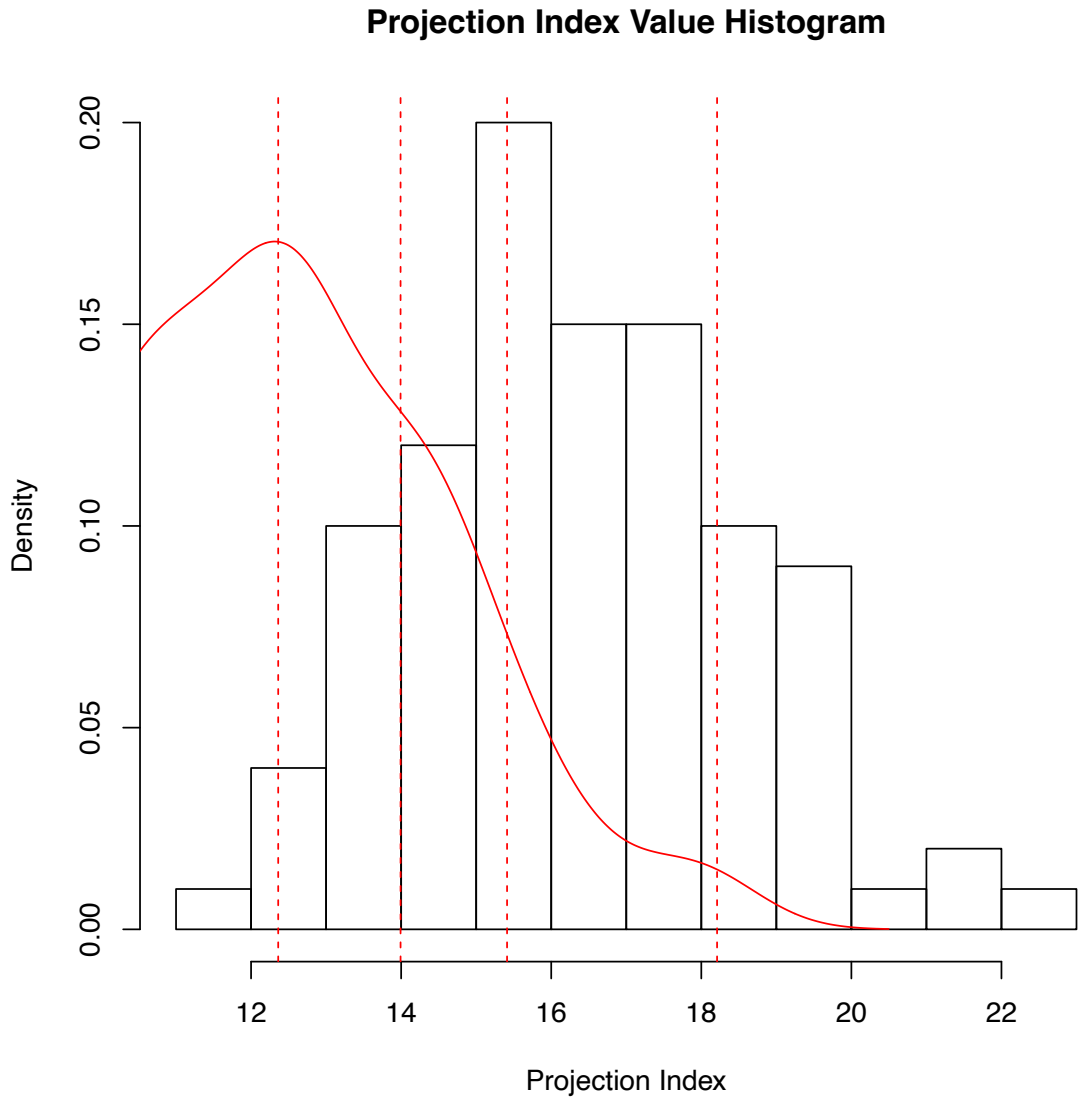
Command for PP3

So, usually, we repeat the projection pursuit procedure from multiple random starts and pick the best one.

We can get pseudo-p-values by performing projection pursuit on an identical n, p configuration, but coordinates chosen at random.

```
beetle.PP3 <- PP3many(t(beetle), nrandstarts=100)
plot(beetle.PP3)
```

Plot showing PP3 indices



Remarks

PP obtains different information than PCA.

Sometimes it 'works' better, sometimes not.

PCA uses centred and standardised variables (makes marginal variance the same for all variables, i.e. on diagonal of variance matrix).

PP uses centred and sphered — transforms variance matrix to identity — much harsher.

Both methods can additionally use the Varimax method afterwards.

Rotate solution in plane of projection solution to try and put maximum weight on fewest original variables. This is a penalised optimisation like lasso. Can be added to PP optimisation as penalty.

Summary

PCA	equate interesting with large variance, which does not always equal clustering.
Exploratory PP	looks for divergence from Gaussian distribution + ignores variance.
	Gaussian maximises entropy
Sphering	variance matrix to identity
Exploratory PP	is numerically optimised, PCA computed 'immediately' from eigendecomposition.

Elements of Statistical Learning: Lecture 16. Independent Components Analysis

Guy Nason[1]

Department of Mathematics
Imperial College

Background

Often data comes as a set of results and we think there are common underlying reasons.

- Educational tests. Tests in a variety of subjects (maths, english, languages, science, humanities) and results might be seen as a reflection of pupils' general intelligence, drive, social background.
- a large portfolio of financial funds — prices driven by common factors (economy, politics, energy costs, global pandemic).

Let's use first scenario. Suppose the test result for pupil $i = 1, \ldots, n$ on test $j = 1, \ldots, p$ is $X_{i,j}$

Factor Model

Then, can imagine

$$X_{i,1} = a_{1,1}S_{i,1} + a_{1,2}S_{i,2} + \cdots + a_{1,p}S_{i,p} \quad (1)$$
$$X_{i,2} = a_{2,1}S_{i,1} + a_{2,2}S_{i,2} + \cdots + a_{2,p}S_{i,p} \quad (2)$$
$$\vdots \qquad\qquad\qquad\qquad \vdots$$
$$X_{i,p} = a_{p,1}S_{i,1} + a_{p,2}S_{i,2} + \cdots + a_{p,p}S_{i,p} \quad (3)$$

The X will be correlated, but we want the S to be uncorrelated (to make interpretation easier).

Using SVD we have $X = UDV^T$. Let $S = \sqrt{n}U$, $(n \times p)$ and $A^T = DV^T/\sqrt{n}$, we can write $X = SA^T$.

To make the maths a bit simpler we assume that the X is centred (and hence so is S).

Remarks on Factor Model

What is (empirical) covariance of S?

$$n^{-1}S^T S = n^{-1}\sqrt{n}U^T U\sqrt{n} = I_p, \tag{4}$$

so S variables are uncorrelated.

Let R be a $p \times p$ orthogonal matrix and write

$$X = SA^T = SRR^T A^T = S^*(A^*)^T, \tag{5}$$

and

$$\text{cov}(S^*) = R^T \text{cov}(S)R = R^T R = I_p. \tag{6}$$

So, there is no unique decomposition into uncorrelated factors $S_1, \ldots, S_p$. In (5) either S or S^* will do the job, even though these are potentially different factors.

Factor Analysis Model

$$X_{i,1} = a_{1,1}S_{i,1} + a_{1,2}S_{i,2} + \cdots + a_{1,q}S_{i,q} + \epsilon_{i,1} \quad (7)$$
$$X_{i,2} = a_{2,1}S_{i,1} + a_{2,2}S_{i,2} + \cdots + a_{2,q}S_{i,q} + \epsilon_{i,2} \quad (8)$$
$$\vdots \qquad\qquad \vdots$$
$$X_{i,p} = a_{p,1}S_{i,1} + a_{p,2}S_{i,2} + \cdots + a_{p,q}S_{i,q} + \epsilon_{i,p} \quad (9)$$

Here, there are $q < p$ factors, ϵ are zero mean and uncorrelated and S often assumed to be Gaussian.

Can fit using maximum likelihood, but identifiability problem mentioned in (5) still exists.

So, some view this method with suspicion.

Independent Component Analysis

ICA model has same form as (1)–(3) except the S_ℓ are now assumed to be statistically independent, not just uncorrelated.

This specifies relationships with moments of all orders concerning S, not just second order.

Gaussian assumptions on S put us in the same situation as before (as independence and uncorrelated are the same thing for Gaussian).

So, independence assumptions do not help for Gaussian model.

Hence, for something unique, we have to assume that the S are independent AND non-Gaussian.

ICA

As with Exploratory PP we sphere X — this is also known as *whitening*.

Since both S, X have identity covariance matrix, this means we are searching for *orthogonal* A in the problem $X = SA^T$.

Problem is: find orthogonal A such that the components of vector random variable $S = AX$ are independent and non-Gaussian (n.b. vectors here, data matrices come a bit later).

Cocktail party problem

A set of microphones X_j pick up a mixture of different independent sources=, S_ℓ (people having different conversations).

ICA can perform *blind source separation* — disentangle the signals from data X_j to estimate the original conversations S_ℓ.

ICA often uses entropy H.

Blind source separation of ICA

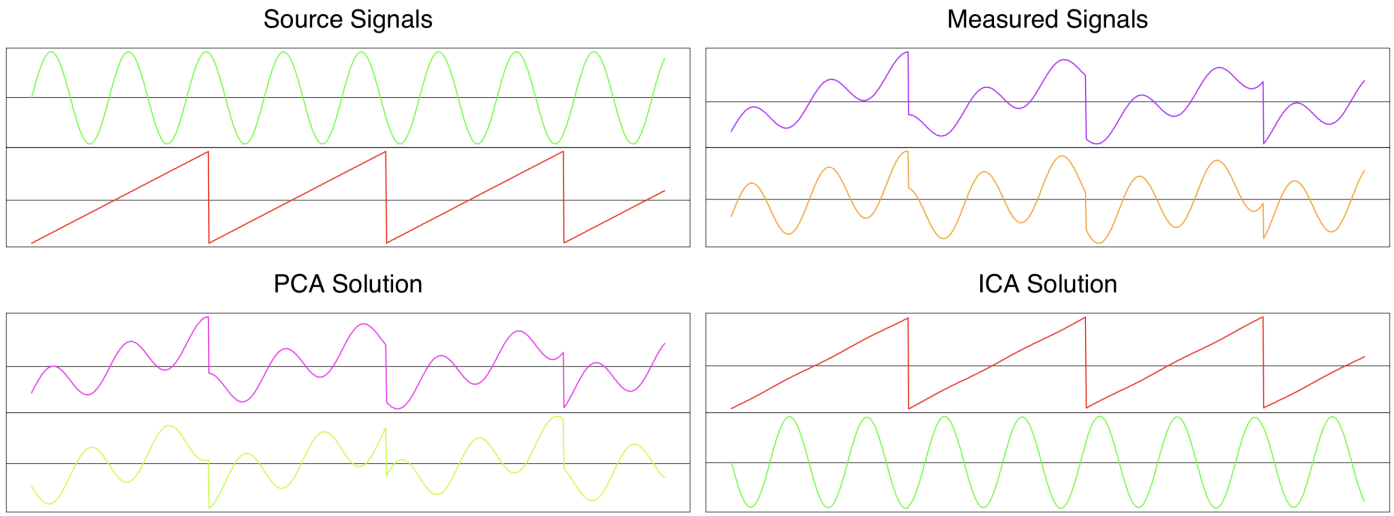


FIGURE 14.37. *Illustration of ICA vs. PCA on artificial time-series data. The upper left panel shows the two source signals, measured at* 1000 *uniformly spaced time points. The upper right panel shows the observed mixed signals. The lower two panels show the principal components and independent component solutions.*

ICA: Measure of dependence

If $Y \sim g$ (i.e. g is pdf of Y) can write $H(g) = H(Y)$.

Define the *mutual information* between components of a p-dimensional random vector Y to be

$$I(Y) = \sum_{j=1}^{p} H(Y_j) - H(Y), \tag{10}$$

where $H(Y_j)$ is the entropy of the component Y_j and $H(Y)$ of the entire vector.

Not difficult to see that $\sum_{j=1}^{p} H(Y_j)$ is the entropy of the 'independence version' of the density of Y, i.e. $\prod_{j=1}^{p} g_j(y_j)$ and $H(Y)$ the entropy of the full p-dimensional density $g(y)$. (Exercise Homework 4)

ICA: Measure of dependence — 2

Now, if X has covariance I, and $Y = A^T X$, with A orthogonal then, it can be shown that (Exercise Homework 4):

$$I(Y) = \sum_{j=1}^{p} H(Y_j) - H(X) - \log|\det A| \quad (11)$$

$$= \sum_{j=1}^{p} H(Y_j) - H(X). \quad (12)$$

We are minimising this over A. Since X is fixed, the only thing that can change are the Y_js, we need to minimise $\sum_{j=1}^{p} H(Y_j)$.

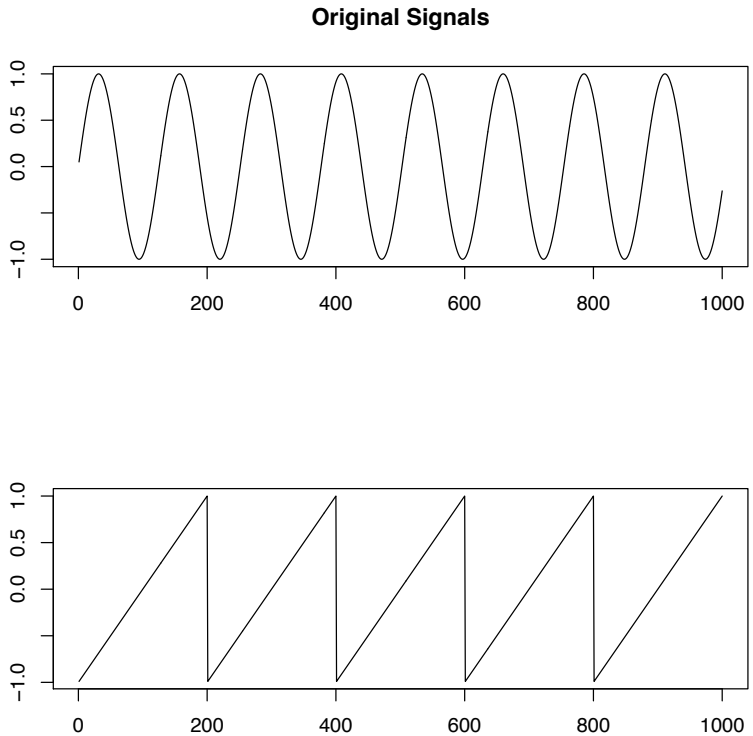
This amounts to making them all as non-Gaussian as possible.

In practice, a version of $H(Y_j)$ re-centred by $H(\phi)$ is used, or [$D_{\mathrm{KL}}(g||f)$ from Lecture 15, called negentropy] which is ≥ 0, then minimising the sum amounts to trying to force all of the individual negentropies to be as small as possible.

Example (from fastICA)

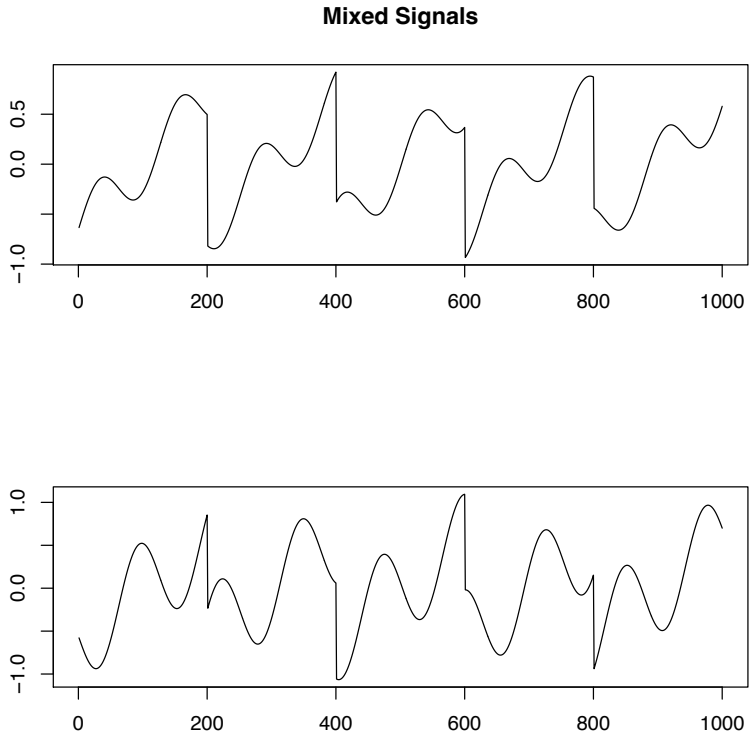
Let's build a sine wave and a sawtooth function.

```
library("fastICA")
S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5))
oldpar <- par(mfrow=c(2, 1))
plot(1:1000, S[,1 ], type = "l", main = "Original Signals",
    xlab = "", ylab = "")
plot(1:1000, S[,2 ], type = "l", xlab = "", ylab = "")
```



Mix the signals

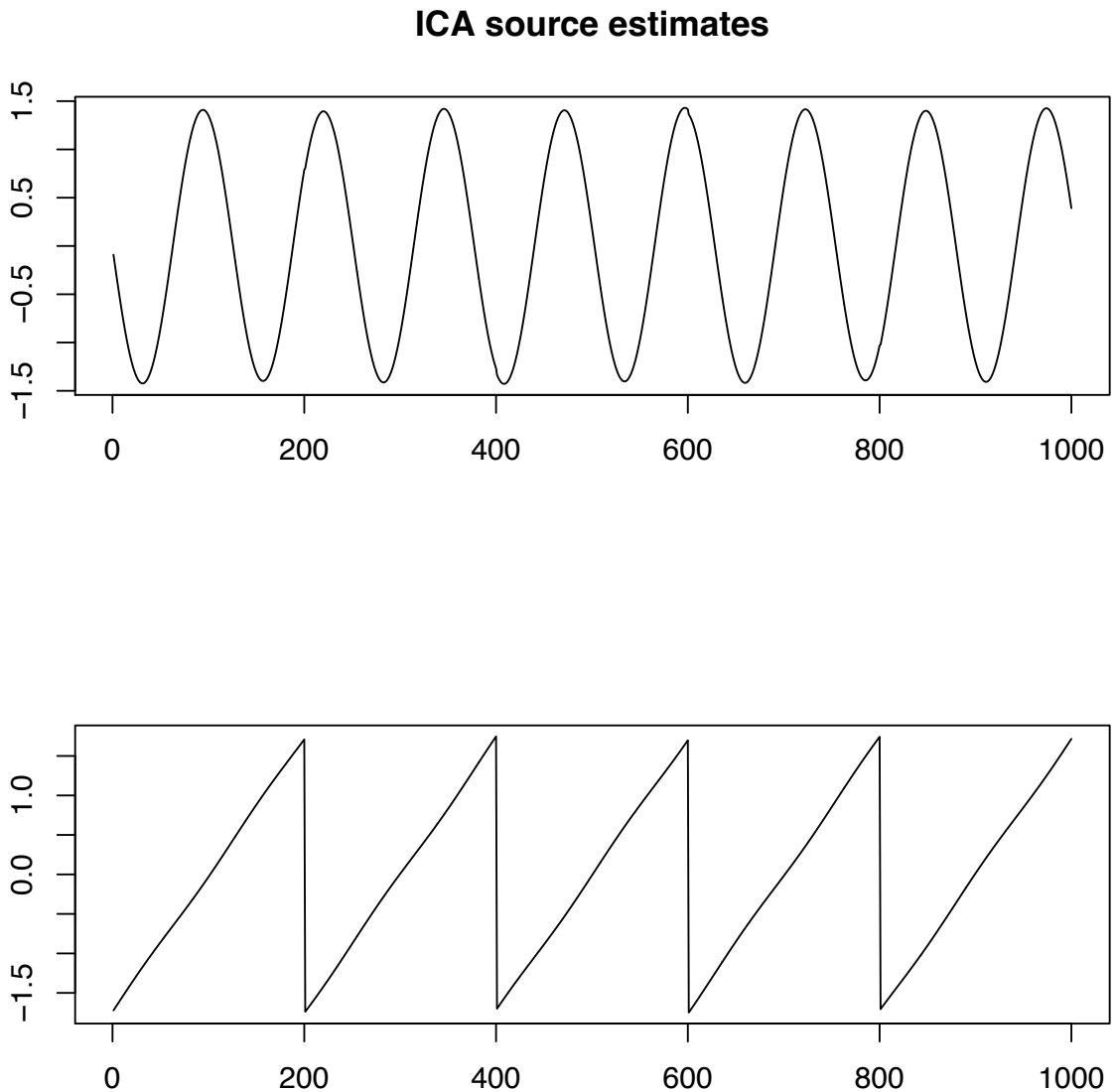
```
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
X <- S %*% A
plot(1:1000, X[,1 ], type = "l", main = "Mixed Signals",
    xlab = "", ylab = "")
plot(1:1000, X[,2 ], type = "l", xlab = "", ylab = "")
```



Apply ICA

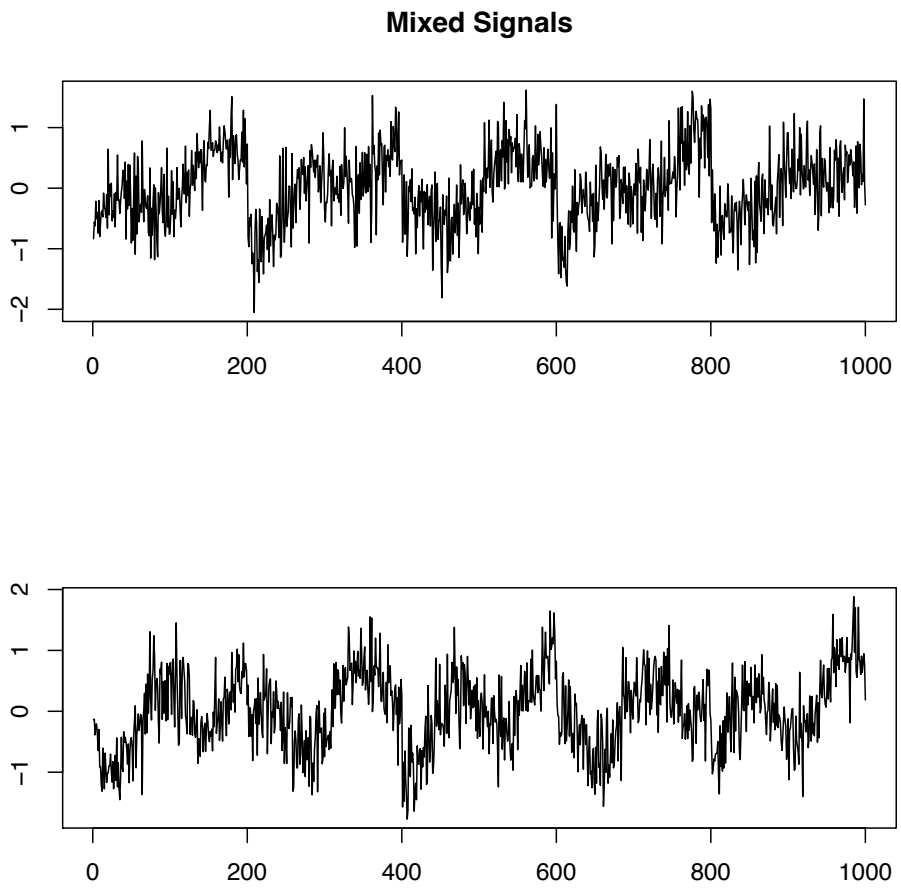
```
> a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh",
    alpha = 1, method = "R", row.norm = FALSE, maxit = 200,
    tol = 0.0001, verbose = TRUE)
Centering
Whitening
Symmetric FastICA using logcosh approx. to neg-entropy function
Iteration 1 tol = 6.259096e-05

> plot(1:1000, a$S[,1 ], type = "l",
    main = "ICA source estimates", xlab = "", ylab = "")
> plot(1:1000, a$S[, 2], type = "l", xlab = "", ylab = "")
```

Apply ICA Plots

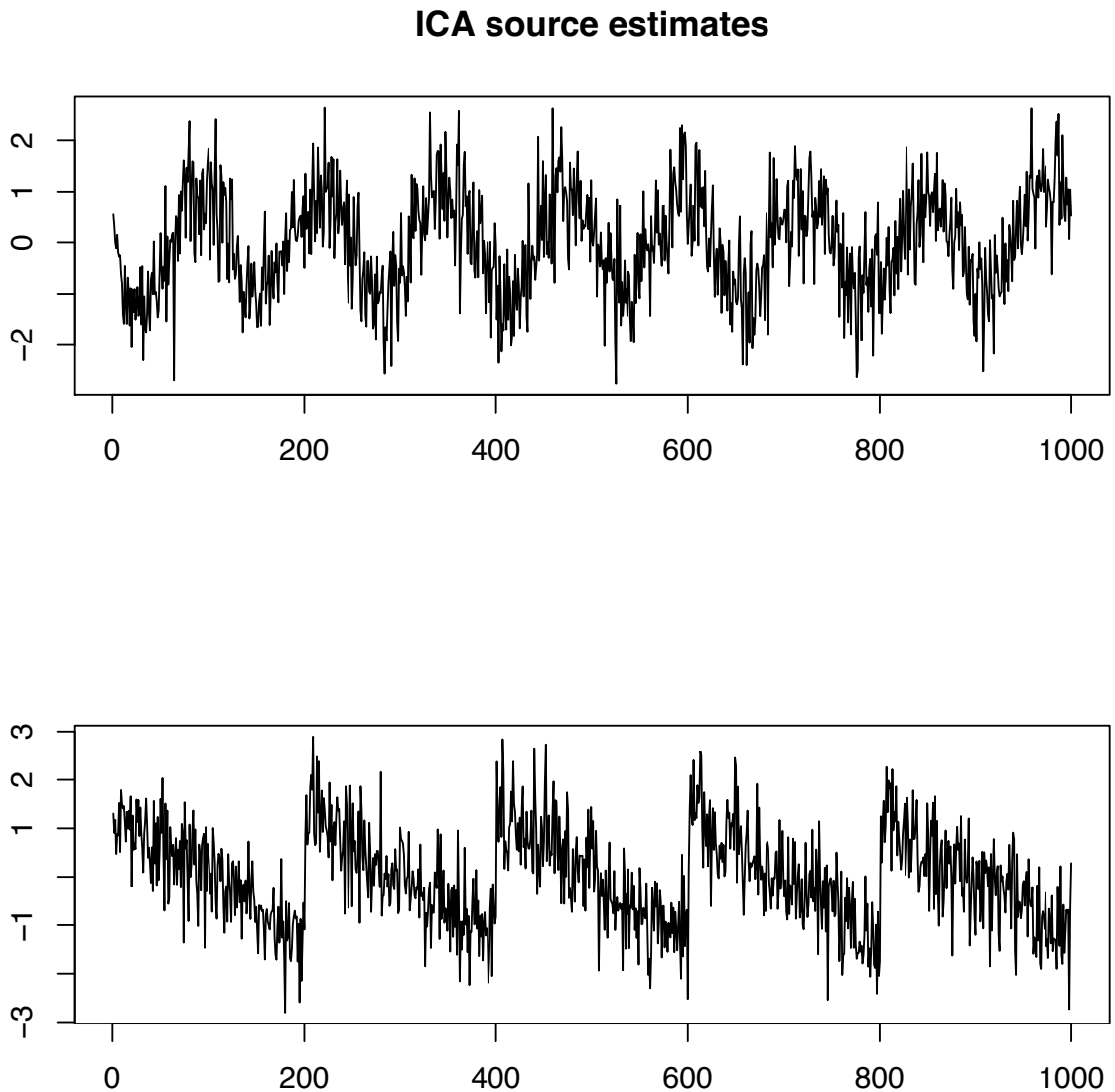


Mix the signals

```
set.seed(100)
Xnoise <- X + matrix(rnorm(2000, mean=0, sd=0.4), ncol=2)
```



Apply ICA Plots on the noisy version



Summary

This lecture covered:

Common factors the idea

Factor model

Identifiability issues

Independent Components Analysis

Cocktail Party Problem

- Examples using the `fastICA` package

Elements of Statistical Learning: Lecture 17. Neural Networks

Guy Nason[1]

Department of Mathematics
Imperial College

Background

Here, we are doing regression again.

In vector terms, we have

X a p-dimensional explanatory variable vector

Y the response, also known as a target

Note: when we collected data, then the vector X becomes the data matrix X.

Projection Pursuit Regression

Here the model is

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X), \tag{1}$$

this is an additive model on the derived directions $V_m = \omega_m^T X$, and not the X directly; ω_m are unit vectors

the functions g_m are not specified and estimated as part of the regression, along with the ω_m using a flexible smoothing method.

The function $g_m(\omega_m^T X)$ is called a *ridge function* in $\mathbb{R}^p$ and only varies in the direction ω_m.

The V_m is the projection of X onto ω_m, and we want to find ω_m so the model fits well: like projection pursuit.

PPR: Issues

The PPR model is extremely general and flexible.

E.g. $X_1 \cdot X_2$ can be obtained from $\{(X_1 + X_2)^2 - (X_1 - X_2)^2\}/4$.

For large M, for appropriate $\{g_m\}$, the PPR model can approximate any continuous function on $\mathbb{R}^p$ arbitrarily well. Called a *universal approximator*.

However, it is easy to *overfit* and so we need to keep things simple, restrictions on M and, perhaps, the $\{g_m\}$.

Interpretation can be very difficult — map from inputs X to f can be complex involving two layers of linear combination and the very flexible $\{g_m\}$ functions.

The model with $M = 1$ is called the *single index model*.

How to fit PPR models

Suppose we have training data (x_i, y_i), $i = 1, \ldots, n$; $x_i \in \mathbb{R}^p$

We wish to find approximate minimisers of the error

$$E = \sum_{i=1}^{n} \left\{ y_i - \sum_{m=1}^{M} g_m(\omega_m^T x_i) \right\}^2, \tag{2}$$

over $\{g_m\}$ and direction vectors $\{\omega_m\}$.

Let's consider $M = 1$ model and drop the m subscript.

Given a direction, ω, we can form $v_i = \omega^T x_i$.

Now have a 1D smoothing problem of y_i on v_i. Can use, e.g. smoothing spline.

Finding good directions

Assume that we've found a suitable g, via smoothing spline.

Now we want to minimise the error of ω.

We can use Gauss-Newton search (i.e. numerical calculus).

Let ω_{old} be the existing estimate of ω.

Then

$$g(\omega^T x_i) \approx g(\omega_{\text{old}}^T x_i) + g'(\omega_{\text{old}}^T x_i)(\omega - \omega_{\text{old}})^T x_i, \qquad (3)$$

Finding good directions — 2

So that

$$
\begin{aligned}
E &= \sum_{i=1}^{n} \{y_i - g(\omega^T x_i)\}^2 && (4)\\
&\approx \sum_{i=1}^{n} \{y_i - g(\omega_{\text{old}}^T x_i) - g'(\omega_{\text{old}}^T x_i)(\omega - \omega_{\text{old}})^T x_i\}^2 && (5)\\
&= \sum_{i=1}^{n} g'(\omega_{\text{old}}^T x_i)^2 \left\{ \frac{y_i - g(\omega_{\text{old}}^T x_i)}{g'(\omega_{\text{old}}^T x_i)} - (\omega - \omega_{\text{old}})^T x_i \right\}^2 \\
&= \sum_{i=1}^{n} g'(\omega_{\text{old}}^T x_i)^2 \left[\left\{ \omega_{\text{old}}^T x_i + \frac{y_i - g(\omega_{\text{old}}^T x_i)}{g'(\omega_{\text{old}}^T x_i)} \right\} - \omega^T x_i \right]^2 && (6)
\end{aligned}
$$

This is just a weighted least squares regression with target $\left[\omega_{\text{old}}^T x_i + \{y_i - g(\omega_{\text{old}}^T x_i)\}/g'(\omega_{\text{old}}^T x_i)\right]$ on the input x_i with weights $g'(\omega_{\text{old}}^T x_i)^2$ and no intercept. Gives ω_{new}.

Alternating/Iterating algorithm

The PPR algorithm proceeds by iterating:

- finding good g using current ω (e.g. smoothing spline)
- using g to update current ω to ω

Iterate until convergence — e.g. only need to examine changes in ω.

For general M — add a (ω_m, g_m) pair after convergence at each stage.

Can decide M by examining drop in error as M increases — and stop when error drop isn't worth it — and/or cross-validation.

Neural Networks

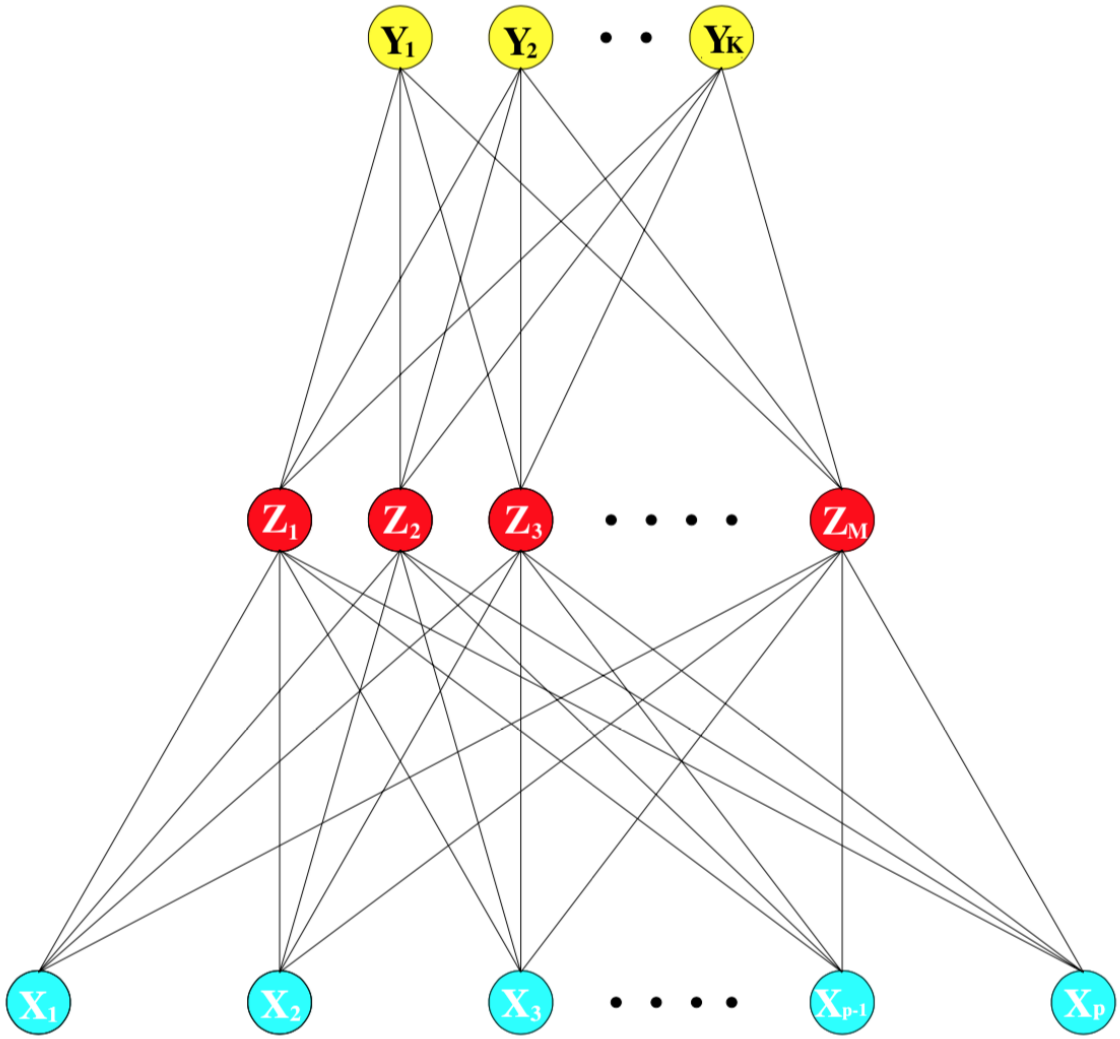
Large class of models — variety of descriptions

We will discuss the single hidden layer back-propagation network or *single layer perceptron*.

Possibly over-hyped in the sense that they are just nonlinear statistical models.

However, to be fair, enormous computational resources have been thrown at these problems and that it is this that has really helped them become popular.

Neural Network Diagram (Fig. 11.2 in book)



Neural Network Explanation

The inputs (explanatory variables) $X_1, \ldots, X_p$ are at the bottom.

There are K units at the top. For regression usually $K = 1$.

For classification, the K units relate to K classes, with zero/one in depending on whether the inputs have stimulated membership of any of the classes.

Derived features Z_m are created from linear combinations of the inputs, and then Y_k is modeled as a function of linear combinations of the Z_m.

Neural Network Model

The model is

$$Z_m = \sigma(\alpha_{0,m} + \alpha_m^T X), \quad m = 1, \ldots, M, \tag{7}$$

$$T_k = \beta_{0,k} + \beta_k^T Z, \quad k = 1, \ldots, K, \tag{8}$$

$$f_k(X) = g_k(T), \quad k = 1, \ldots, K, \tag{9}$$

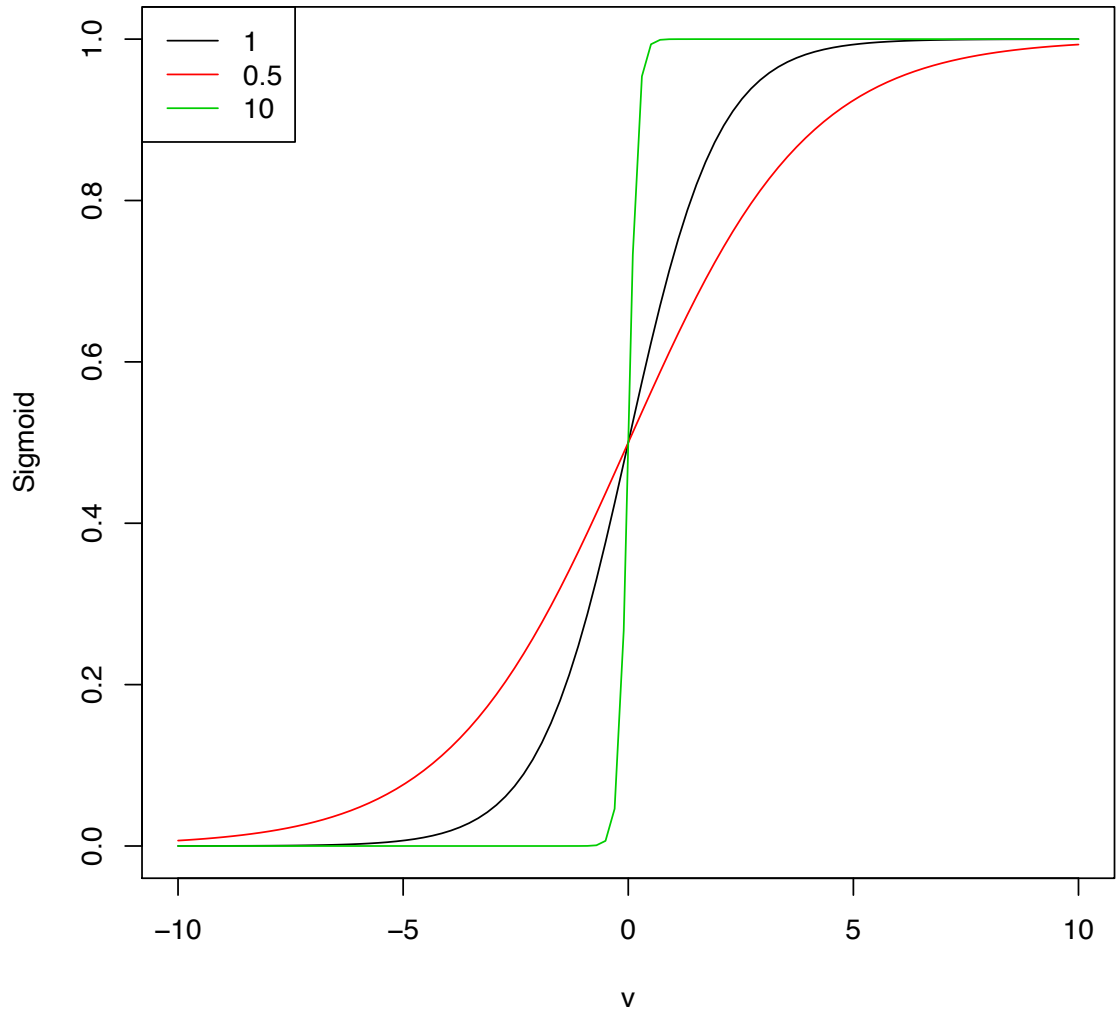
where $Z = (Z_1, \ldots, Z_M)$ and $T = (T_1, \ldots, T_K)$.

The function $\sigma(v)$ is called the *activation function* and often chosen as

$$\sigma(v) = 1/\{1 + \exp(-v)\}, \tag{10}$$

or something similar.

Example activation functions



Large s implies hard activation — bit like a threshold

Remarks

Intercepts can be added to add bias, where needed.

The output function $g_k(T)$ permits a final transformation of the vector of outputs.

For regression, typically $g_k(T_k) = T_k$ the identity.

For classification, the identity was used, but now the *softmax* function

$$g_k(T) = \frac{\exp(T_k)}{\sum_{\ell=1}^{K} \exp(T_\ell)}, \tag{11}$$

is used and occurs elsewhere in statistics.

Remarks — 2

The derived features Z_m are more properly known as *hidden units* as the Z_m are not directly observed.

There is often more than one hidden layer.

If σ is the identity, then we're back to a linear model — so a neural network is a nonlinear generalisation, greatly enlarged.

Indeed, if the values of the αs in (7) are small, then the $X \to Z$ relationship will be operating in the linear range of σ.

Brain Model

The name, neural network, comes from the fact that they were originally developed as models for the human brain. Each unit a neuron and the connections are the synapses.

The sigmoid activation is a reflection of the fact that neurons need enough stimulus to fire, and then, when they get it, it fires.

However, the brain is much more complex: neurons process stuff inside also themselves (each neuron may store different charges in different places, hence can run parallel internal calculations and calculations running through time); different types of chemical internals and interactions between neurons; network architecture is much more complex (not just layers); vastly more neurons; vastly more power efficient; comes with 'built-in' laws about the world.

Connection with PPR

(Artificial) Neural Network (ANN) with one hidden layer has the same form as a the PPR model.

The main difference is that the PPR model uses nonparametric functions $g_m(v)$, whereas the ANN uses the simpler function based on $\sigma(v)$.

Indeed, we can write

$$\begin{aligned} g_m(\omega_m^T X) &= \beta_m \sigma(\alpha_{0,m} + \alpha_m^T X) && (12) \\ &= \beta_m \sigma(\alpha_{0,m} + ||\alpha_m||(\omega^T X)), && (13) \end{aligned}$$

where $\omega_m = \alpha_m / ||\alpha_m||$ is the mth unit vector.

Since $\sigma(v)$ is less complex than g_m, you tend to need many more nodes in ANN than functions in PPR to get the same approximative power.

Fitting Neural Networks

The parameters are called *weights*.

The weights, $\{\theta\}$, are:

$$\{\alpha_{0,m}, \alpha_m : m = 1, \ldots, M\} \qquad M(p+1)\text{weights} \tag{14}$$

$$\{\beta_{0,k}, \beta_k; k = 1, \ldots, K\} \qquad K(M+1)\text{weights.} \tag{15}$$

For regression we use the usual sum of squares criterion for goodness of fit:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{n} \{y_{i,k} - f_k(x_i)\}^2 \tag{16}$$

Typically, we need some constraints as the global minimiser of $R(\theta)$ is likely to severely overfit the data.

Minimisation happens by gradient descent, which is called *back propagation* here.

Back propagation

We need to the gradient, which is not tricky to do, due to the component nature.

Let $z_{m,i} = \sigma(\alpha_{0,m} + \alpha_m^T x_i)$ from (7) and let $z_i = (z_{1,i}, \ldots, z_{M,i})$.

Then

$$R(\theta) = \sum_{i=1}^{n} R_i(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \{y_{i,k} - f_k(x_i)\}^2, \tag{17}$$

with derivatives via chain rule:

$$\frac{\partial R_i(\theta)}{\partial \beta_{k,m}} = -2\{y_{i,k} - f_k(x_i)\} g_k'(\beta_k^T z_i + \beta_{0,k}) z_{m,i}, \tag{18}$$

$$\begin{aligned} \frac{\partial R_i(\theta)}{\partial \alpha_{m,\ell}} = &-2 \sum_{k=1}^{K} \{y_{i,k} - f_k(x_i)\} g_k'(\beta_k^T z_i + \beta_{0,k}) \beta_{k,m} \\ &\times \sigma'(\alpha_m^T x_i + \alpha_{0,m}) x_{i,\ell}. \end{aligned} \tag{19}$$

Back propagation — 2

From these derivatives, we can form a back-propagation/gradient descent algorithm by

$$\beta_{k,m}^{(r+1)} = \beta_{k,m}^{(r)} - \gamma_r \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \beta_{k,m}^{(r)}}, \tag{20}$$

and

$$\alpha_{m,\ell}^{(r+1)} = \alpha_{m,\ell}^{(r)} - \gamma_r \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \alpha_{m,\ell}^{(r)}}, \tag{21}$$

where γ_r is called the *learning rate*.

Now write (18) and (19) as $\delta_{k,i} z_{m,i}$ and $s_{m,i} x_{i,\ell}$ respectively, these can be seen as the "errors" of the current model at the output and hidden layers, respectively.

Back-propagation equations

Putting it all together we can get

$$s_{m,i} = \sigma'(\alpha_m^T x_i + \alpha_{0,m}) \sum_{k=1}^{K} \delta_{k,i} \beta_{k,m} \tag{22}$$

These are called the *back-propagation equations*.

The algorithm proceeds in two stages:

Forward pass: the current weights are fixed, and the predicted values $\hat{f}_k(x_i)$ are computed from (7)—(9). Then

Backward pass: the errors $\delta_{k,i}$ are computed from (18) and then back-propagated via (22) to give the errors $s_{m,i}$ and these form the gradients for the updates in (20) and (21).

Issues around fitting ANNs

Starting values: usually chosen to be random, but small, near zero. Then the ANN is similar to a linear model (as noted earlier) and the weights get increased over time.

Overfitting: so many parameters so that optimising too hard results in overfitted solution. Early techniques just stopped. More modern versions optimise more, but shrink the weights — this is like ridge regression.

Scaling of inputs: as with other methods but perhaps worse - solution is sensitive to input scales. So, as with earlier methods often centre and standardise inputs.

Number of hidden units and layers: usually use too many hidden units, can always optimise their weights to zero. More tricky to know the best number of hidden layers — experimentation with a problem.

Neural networks on Boston data

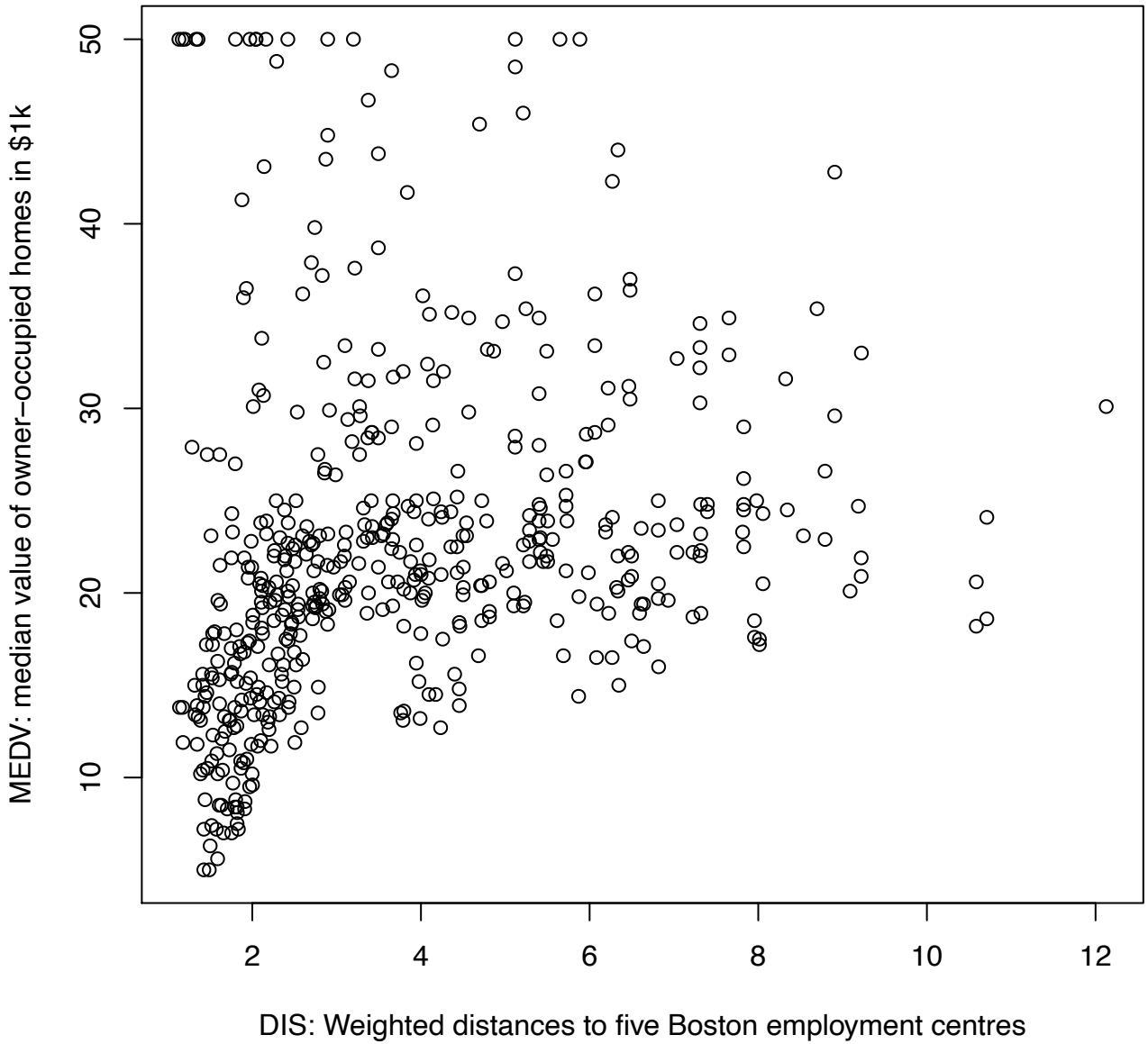
See:

`https://www.r-bloggers.com/`
`fitting-a-neural-network-in-r-neuralnet-package/`

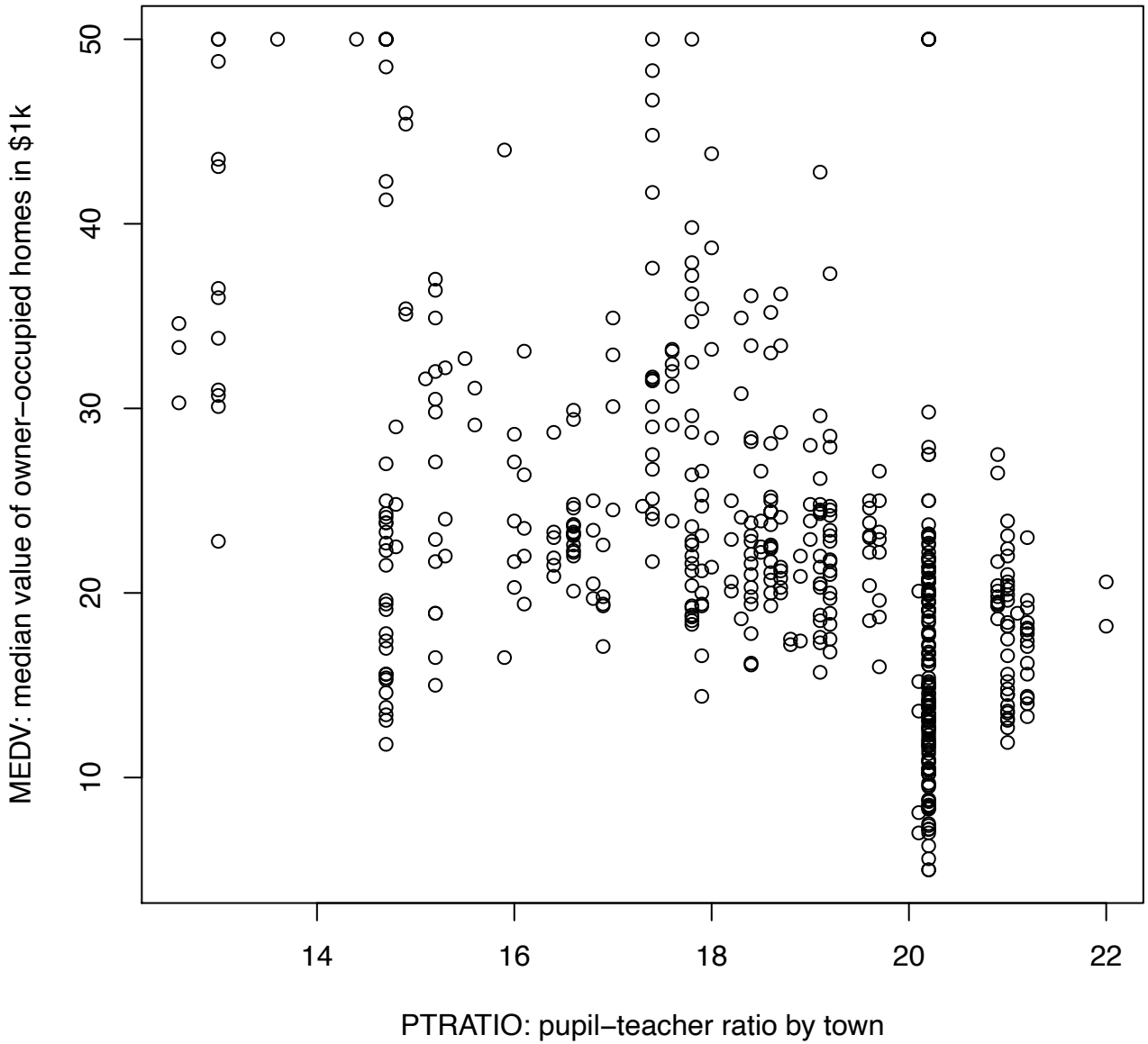
Boston: data on housing in the Boston area, 506 and 14 variables. Two possible response variables: MEDV - median value of owner-occupied homes and NOX, nitrogen oxide levels

```
library("MASS")
librart("glmnet")
library("neuralnet")
data(Boston)
```

Boston data plot of two variables



Boston data plot of two other variables



Neural networks on Boston data

```
set.seed(500)
# Generate training and test set
train.index <- sample(1:nrow(Boston), round(0.75*nrow(Boston)))
train <- Boston[train.index,]
test <- Boston[-train.index,]

# Fit linear model to training set
Boston.train.lm <- lm(medv ~ ., data=train)

# Predict new Boston data on the linear model
pr.Boston <- predict(Boston.train.lm, newdata=test)

# Plot the predictions against the truth
plot(test$medv, pr.Boston, xlab="Actual Data", ylab="Predicted")
abline(a=0, b=1, lty=2)

ssq <- sum((pr.Boston - test$medv)^2)
cat("SSQ: ", ssq, "\n")
```

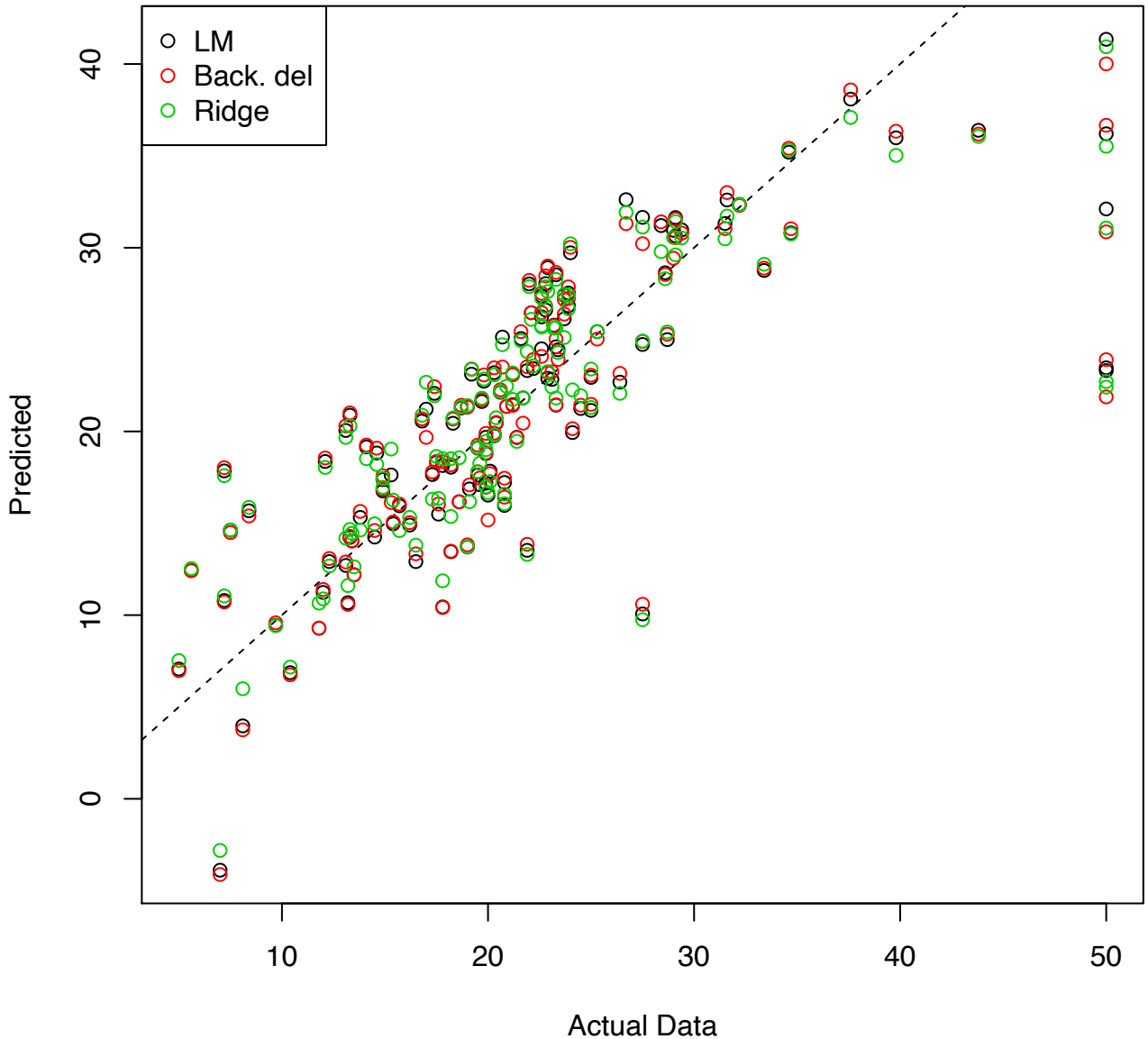
Continued — 2

```
# Fit updated model, by dropping insignficant variables
Boston.lm2 <- update(Boston.train.lm, . ~ . -indus-age-chas,
    data=train)
pr.Boston2 <- predict(Boston.lm2, newdata=test)

# Plot the prediction of the updated model
points(test$medv, pr.Boston2, col=2)

ssq <- sum((pr.Boston2 - test$medv)^2)
cat("(Back. deletion:) SSQ: ", ssq, "\n")

# Create model matrices for ridge and lasso software
xB <- model.matrix(medv ~., train)[,-1]
xBtest <- model.matrix(medv ~., test)[,-1]
yB <- train$medv
# Compute Ridge CV value
cv.Boston.train <- cv.glmnet(xB, yB, alpha=0)
bestlam <- cv.Boston.train$lambda.min
cat("bestlam: ", bestlam, "\n")
```

Continued — 3

```
# Compute ridge model and prediction
Boston.train.ridge <- glmnet(xB, yB, alpha=0, lambda=0.5)
Boston.train.predict.ridge <- predict(Boston.train.ridge,
    s=bestlam, newx=xBtest)

# Plot the predictions
points(test$medv, Boston.train.predict.ridge, col=3)

ssq <- sum((Boston.train.predict.ridge - test$medv)^2)
cat("(CV Ridge:) SSQ: ", ssq, "\n")
```

Predictions against truth



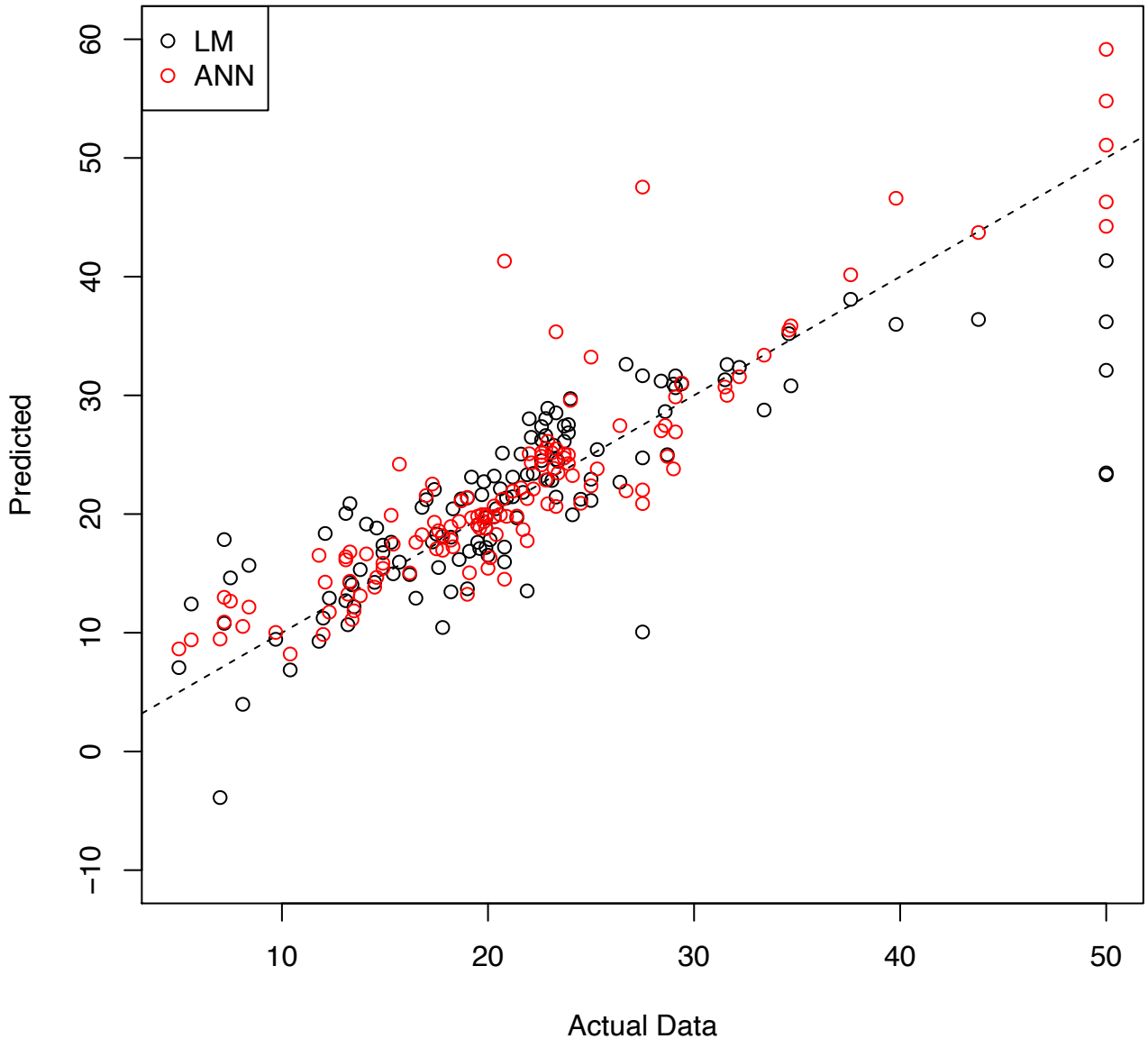
Neural Network code

```
# Get max and min of each variable in Boston
maxs <- apply(Boston, 2, max)
mins <- apply(Boston, 2, min)

# Rescale data to be on similar scales
scaled <- as.data.frame(scale(Boston, center=mins,
    scale=maxs-mins))

# Get training and test set from scaled data
trainS <- scaled[train.index,]
testS <- scaled[-train.index,]

# Do neuralnet fit with two hidden layers with 5 and 3 nodes
nm <- names(trainS)
form <- as.formula(paste("medv ~",
    paste(nm[!nm %in% "medv"], collapse=" + ")))
Boston.nn <- neuralnet(f=form, data=trainS, hidden=c(5,3),
    linear.output=TRUE)
```

Neural Network code — 2

```
# Compute predictions from ANN model, omit medv
predict.Boston.nn <- compute(Boston.nn, testS[, 1:13])

# Undo the earlier scaling
predict.Boston.unsc <- predict.Boston.nn$net.result*
    (max(Boston$medv)-min(Boston$medv)) + min(Boston$medv)

points(test$medv, predict.Boston.unsc, col=2)
legend(x="topleft", legend=c("LM", "ANN"), col=1:2, pch=1)
```

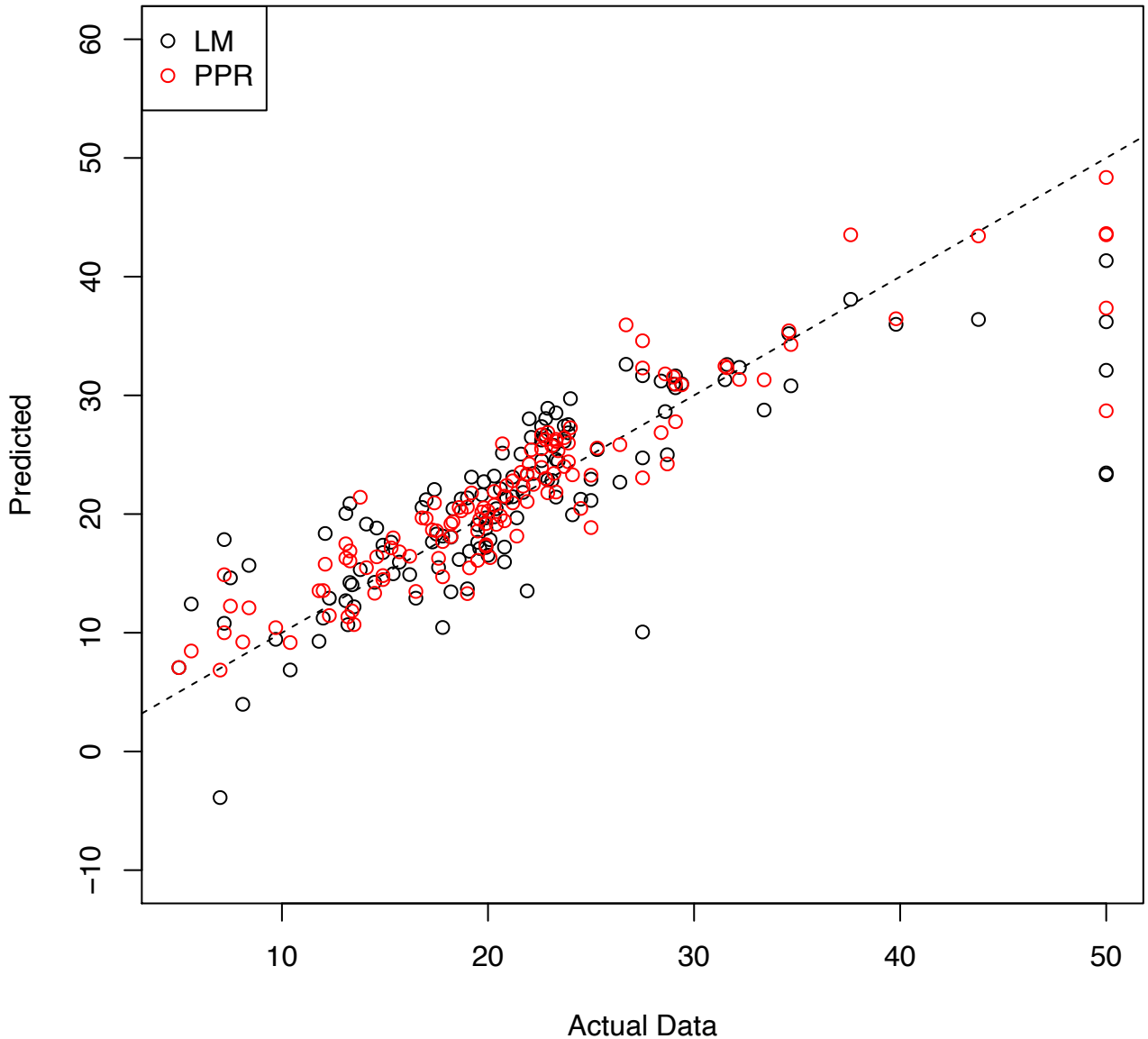
Predictions against truth



Projection pursuit regression

```
Boston.ppr <- ppr( medv ~ ., data=trainS, nterms=2,
    max.terms=5)

predict.Boston.ppr <- predict(Boston.ppr, newdata=testS)

predict.Boston.ppr.unsc <- predict.Boston.ppr*
    (max(Boston$medv)-min(Boston$medv)) + min(Boston$medv)

points(test$medv, predict.Boston.ppr.unsc, col=2)
legend(x="topleft", legend=c("LM", "PPR"), col=1:2, pch=1)
```

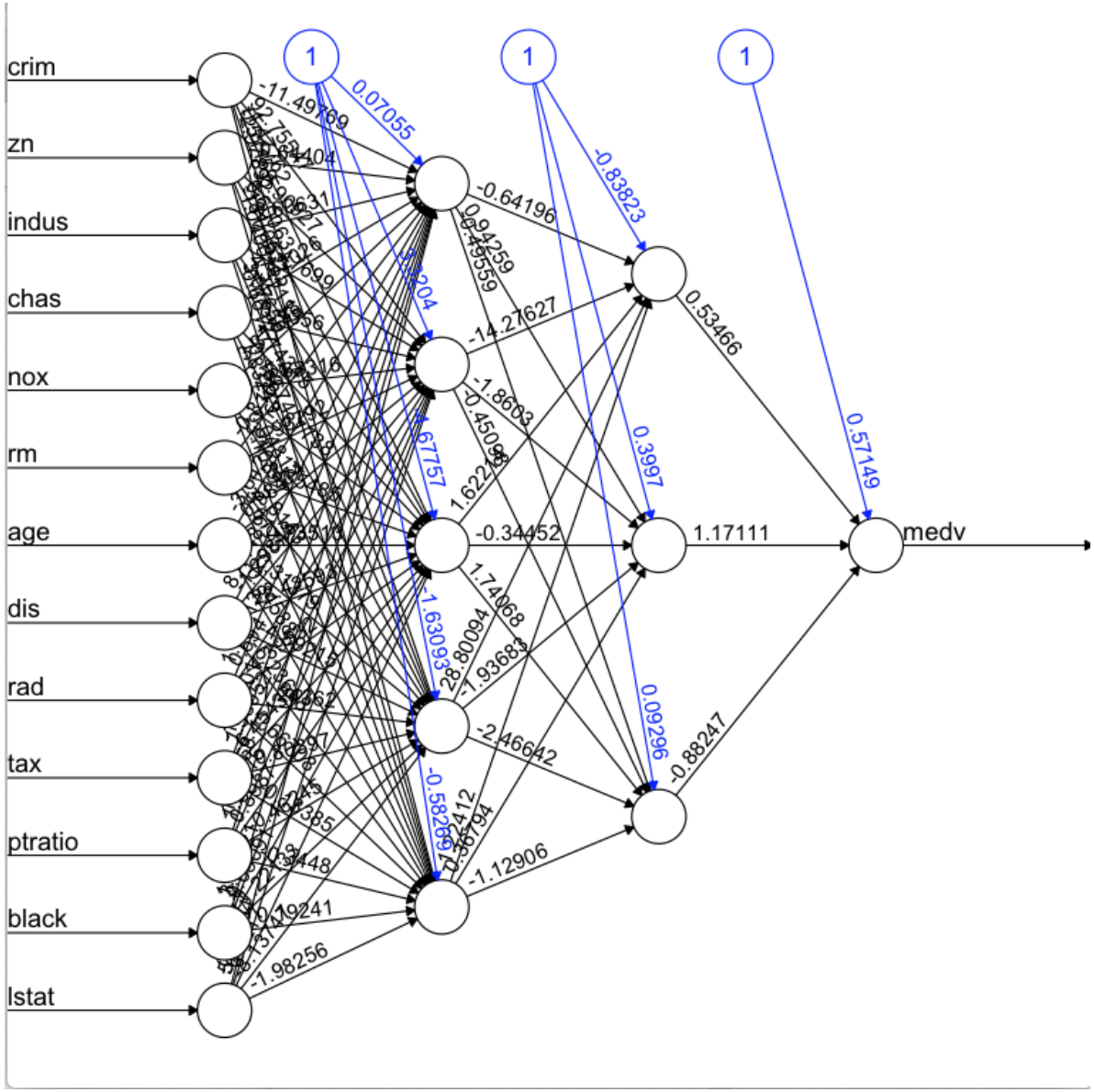
Predictions against truth



Sum of squared error

Method	SSQ
Least Squares	3939
Back deletion	4020
Ridge	3981
Lasso	3938
ANN	2073
PPR	1651

Fitted neural network



PPR solution

Smoothing functions g_m are supersmoother.
Directions

Variable	ω_1	ω_2
crim	-0.507	-0.174
zn	-0.002	-0.010
indus	0.058	-0.275
chas	-0.010	0.136
nox	-0.168	-0.408
rm	0.541	-0.509
age	-0.052	0.158
dis	-0.258	-0.544
rad	0.091	0.226
tax	-0.156	0.042
ptratio	-0.163	0.056
black	0.120	0.031
lstat	-0.525	-0.269

Summary

This lecture covered

Projection Pursuit Regression (PPR)

Fitting PPR models

Neural Networks Model connection to PPR + the brain model

Fitting Neural Networks Back-propagation

Issues around fitting ANNs

Elements of Statistical Learning: Lecture 18. Tree-Based Methods

Guy Nason[1]

Department of Mathematics
Imperial College

Introduction

Suppose we have a list of variables X_1, X_2, X_3.

These variables come together to make a 3D *feature space*.

We assume that, for individual i, there is a response Y_i that is associated with $X_{i,1}, X_{i,2}, X_{i,3}$.

E.g. in the child height-weight data, child 5 is $X_{5,1} =$ female, of age $X_{5,2} = 191$ months (nearly 16), of height $X_{5,3} = 158.75$cm and weight $Y_5 = 51.02$kg.

Tree-based methods partition feature space into a set of rectangles, and then assign a constant to each of those rectangles.

The constant is the estimate for the 'truth' in that region of the explanatory variables.

Intro: Classification and Regression Trees (CART)

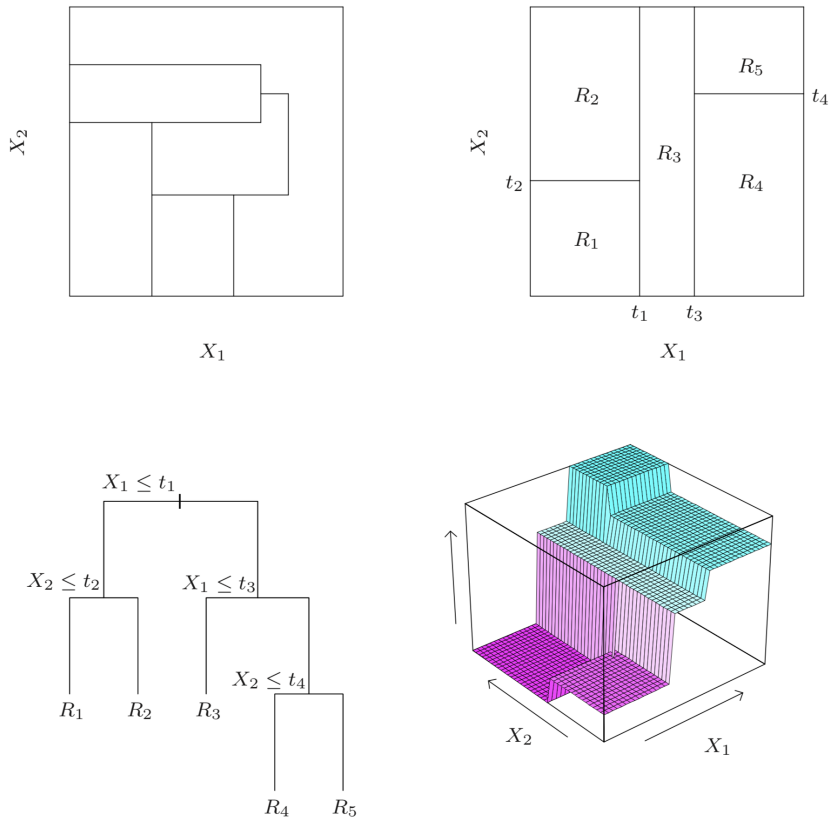


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

Nomenclature

The top of the tree is called the *root*.

The bottom nodes $R_1, \ldots, R_5$ are called *leaves* or *terminal nodes*.

Decisions split the tree, there are no decisions at the leaves.

The outcome for a new observation can be obtained by looking at its explanatory variables, then answering the 'decision questions' at each node, following the Yes/No questions, and traversing the tree from the root to one of the leaves.

The leaves can be numerical values (for regression) or categories (for classification).

Intro: Options

Let's now consider 2D explanatory variables (e.g. `age` & `height`).

Could partition space with separators at any angle or even shape.

Extremely hard to do computationally and also very hard to study.

Simpler: use horizontal and vertical splits only — corresponds to simple functions of the explanatory variables, but again, still hard to study/explain and a huge range of possible choices.

So, what has survived are recursive binary splits — like the right-hand plot of the previous picture.

Intro: Binary splits explained

The binary splits depicted in the top-right can be explained by:

The tree in the bottom-right.

First, decide if $X_1 \leq t_1$. If so, then proceed down that branch;

Then look in the region to the left of $X_1 \leq t_1$,

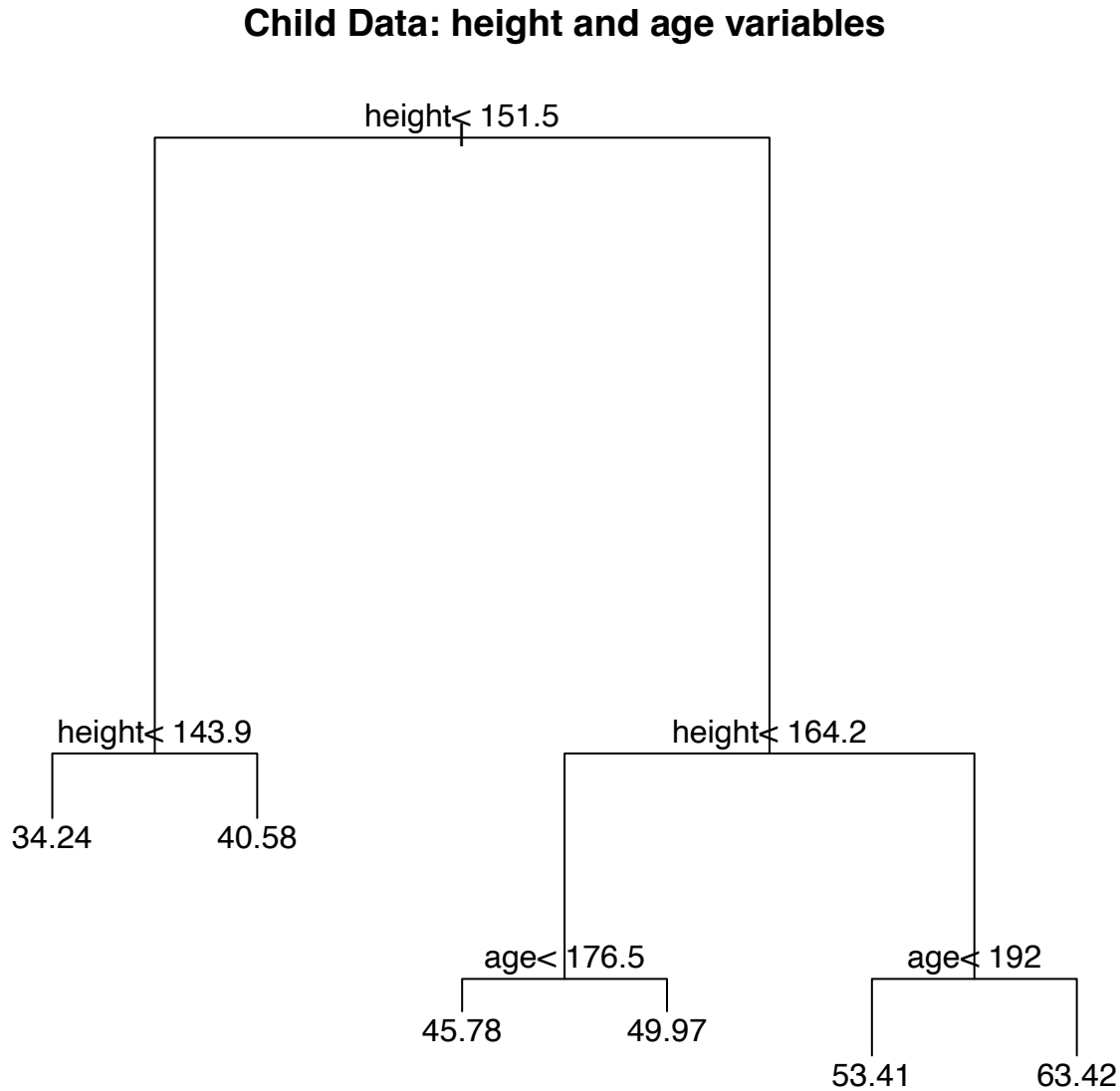
That is then split on $X_2 \leq t_2$ and depending on which will either give the value R_1 or R_2.

The estimated function is shown in the bottom-right.

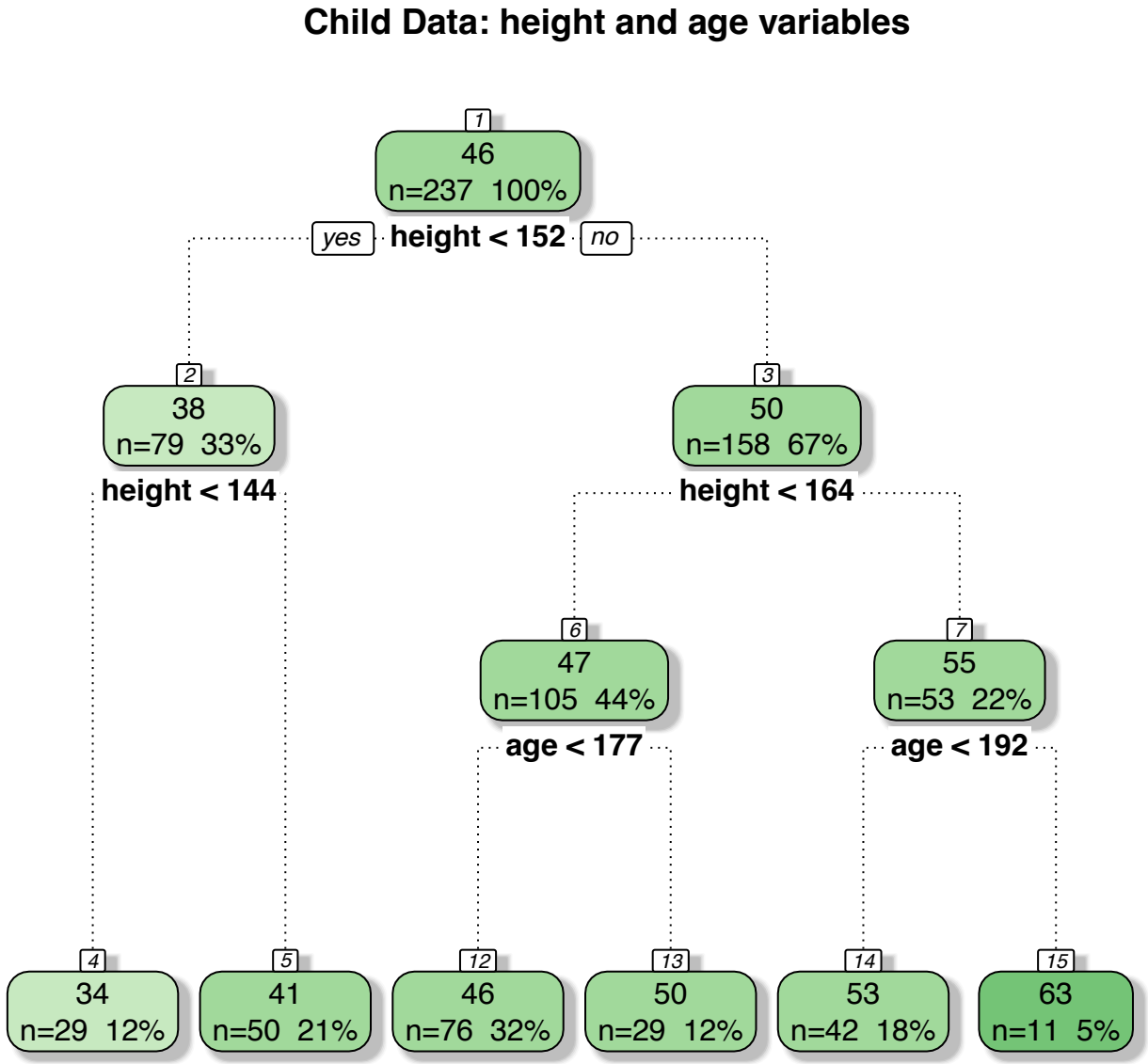
Example: on the child data

```
library("rpart") # Does trees, see also tree package
library("rattle")
library("rpart.plot") # Nice plotting functions
library("RColorBrewer") # Has color palettes

childtree2D <- rpart( weight ~ height + age, data=childhwDF)

# Regular plot
plot(childtree2D,
    main="Child Data: height and age variables")
text(childtree2D)

# Fancy plot
fancyRpartPlot(childtree2D, main=
    "Child Data: height and age variables")
```

Regular Plot



Fancy Plot


Child Data: height and age variables
1
46
n=237 100%
yes
height < 152
no
2
38
n=79 33%
height < 144
3
50
n=158 67%
height < 164
6
47
n=105 44%
age < 177
7
55
n=53 22%
age < 192
4
34
n=29 12%
5
41
n=50 21%
12
46
n=76 32%
13
50
n=29 12%
14
53
n=42 18%
15
63
n=11 5%

How good is it?

```
# Generate test and training set indices
set.seed(20)
nkids <- nrow(childhwDF)
nsample <- round(0.6*nkids)
train <- sample(1:nkids, size=nsample)
test <- -train

# Train tree
kids.train.tree <- rpart(weight ~ height + age,
    data=childhwDF, subset=train)

fancyRpartPlot(kids.train.tree) # Plot tree

# Get explanatory variables on test set
kids.newdata <- childhwDF[test,]

# Predict values on test set using model
kids.tree.predict <- rpart.predict(object=kids.train.tree,
    newdata=kids.newdata)
```

How good is it? — 2

```
# Get the truth too
kids.truth <- childhwDF[test, "weight"]

# Calculate least squares linear estimator on these vars
kids.lm <- lm(weight ~ height+age, data=childhwDF,
    subset=train)

# Predictions using this model
kids.lm.predict <- predict(kids.lm, newdata=kids.newdata)

# Plot tree predictions against truth
oldpar <- par(pty="s")   # Square plot
eqscplot(kids.truth, kids.tree.predict, xlab="True Weight",
    ylab="Predicted Weight",
    main="Predicted weight againts true weight")
points(kids.truth, kids.lm.predict, col=2)
abline(a=0, b=1, lty=2) # Plot y=x line for visual guidance
par(oldpar)
```

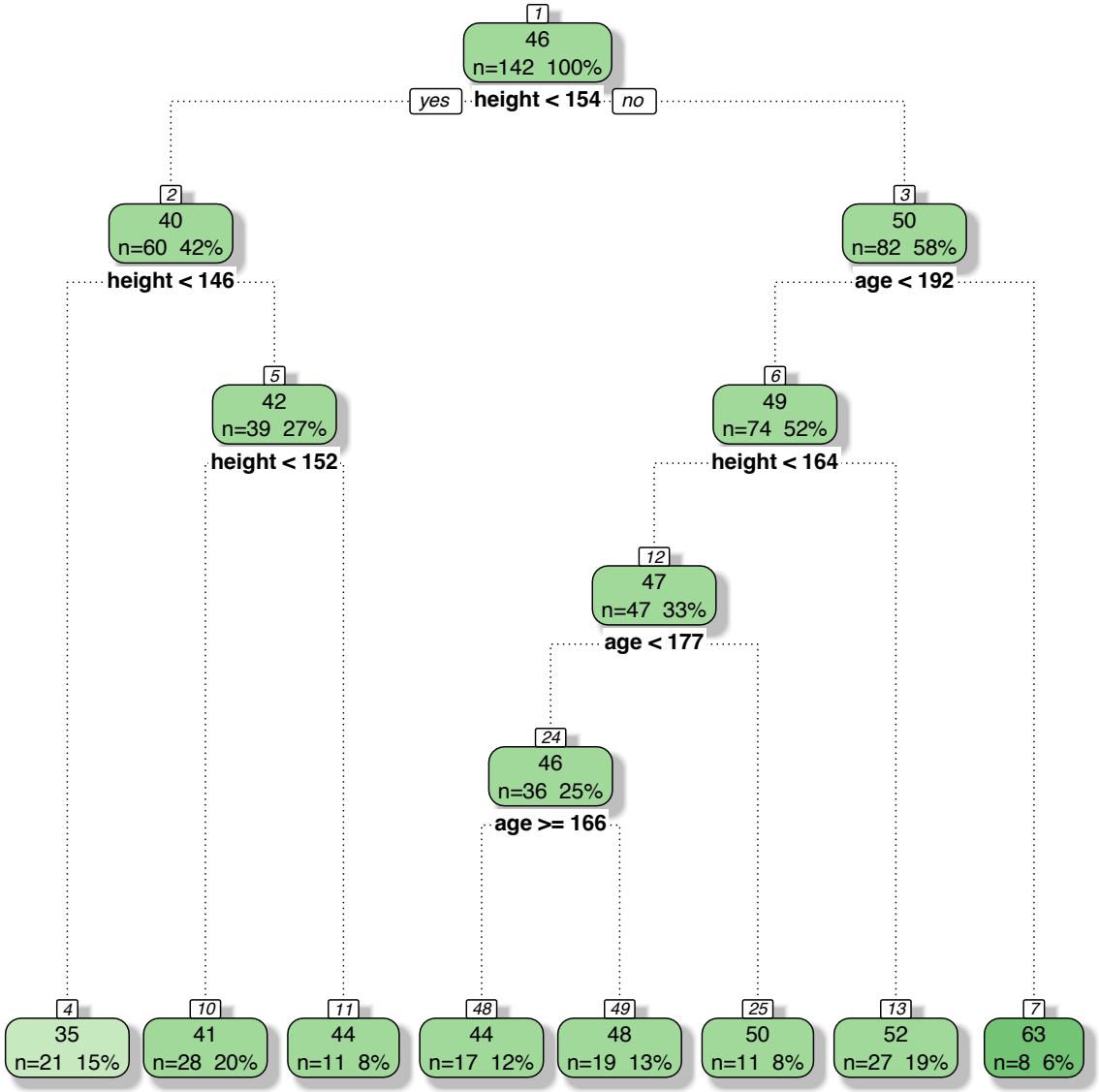
How good is it? — 3

```
# Compute SSQ error for both sets of predictions
ssq.tree <- sum( (kids.tree.predict-kids.truth)^2)
ssq.lm <- sum( (kids.lm.predict-kids.truth)^2)

cat("SSQ Tree: ", ssq.tree, "\n")
cat("SSQ LM: ", ssq.lm, "\n")
```

The printed SSQ values are:

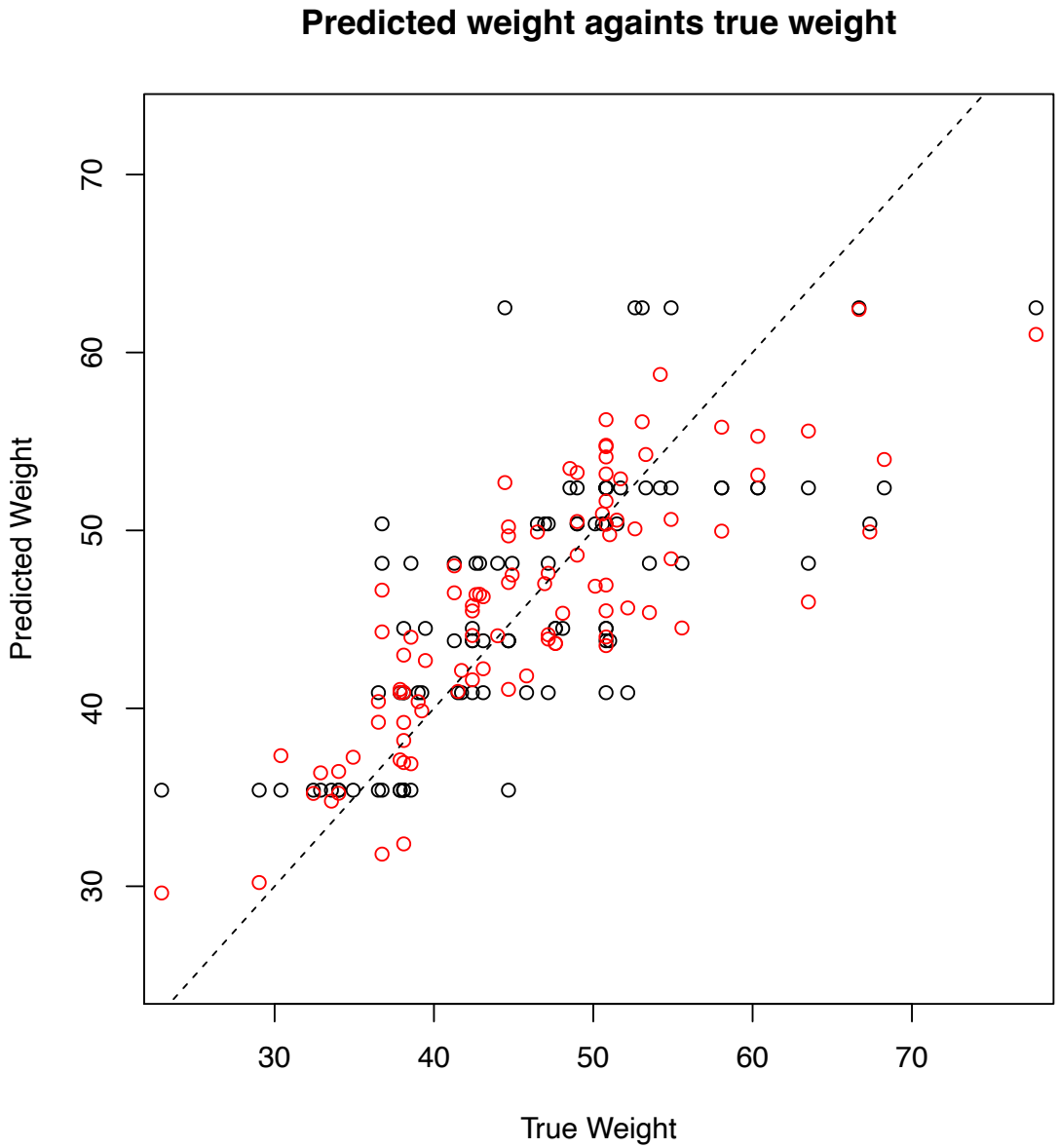
```
SSQ Tree:  3655.515
SSQ LM:  2767.588
```

Tree constructed from training set of child data



Rattle 2020–Feb–24 13:19:05 magpn

Tree predictions and those from linear modelling



How it works: growing a tree

The algorithm needs to decide:
which variables to split on;
whereabouts to split on a given variable;
and what shape the tree is (e.g. binary or ternary, etc)

Suppose that we have a partition into M regions: $R_1, \ldots, R_M$,

and model the response as constant c_m in each region so that

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}(x \in R_m). \tag{1}$$

We want to minimise the criterion:

$$\text{SSQ} = \sum_{i=1}^{n} \{Y_i - f(X_i)\}^2 . \tag{2}$$

How it works: growing a tree — 2

Since we're fitting a constant to each region, the estimate must be the mean of all the Y_i in that region, i.e.

$$\hat{c}_m = n_m^{-1} \sum_{i:X_i \in R_m} Y_i = \text{ave}(Y_i | X_i \in R_m). \tag{3}$$

where n_m are the number of $X_i \in R_m$.

Finding the best binary partition in terms of SSQ is computationally infeasible.

So, we use a greedy algorithm. Start with all the data.

Consider a splitting variable $j = 1, \ldots, p$ and split point s (somewhere on X_j), and define the pair of half-planes:

$$R_1(j, s) = \{x | X_j < s\} \text{ and } R_2(j, s) = \{x | X_j > s\}. \tag{4}$$

How it works: growing a tree — 3

Then, we seek the splitting variable j and split point s that solves

$$\min_{j,s}\left\{\min_{c_1}\sum_{X_i\in R_1(j,s)}(Y_i-c_1)^2+\min_{c_2}\sum_{X_i\in R_2(j,s)}(Y_i-c_2)^2\right\}. \quad (5)$$

The inner minimisations are easily solved by
$\hat{c}_1 = \text{ave}(Y_i|X_i \in R_1(j,s))$ and similarly for $\hat{c}_2$.

For a given j, finding the split s that minimises

$$\sum_{X_i\in R_1(j,s)}(Y_i-\hat{c}_1)^2+\sum_{X_i\in R_2(j,s)}(Y_i-\hat{c}_2)^2, \quad (6)$$

isn't too tricky. Why?

How it works: growing a tree — 4

The sum in (6) does not change, except when s crosses X_i for some i.

When s crosses X_i from left to right, then one point (X_i, Y_i) moves from the right-hand term, to the left hand-term.

So, there will only be $n + 1$ (quickly computable) different values for (6) and we choose the s that gives us the minimum.

We can repeat the inner minimisation over all variables j and this is worst case $\mathcal{O}\{p(n + 1)\}$ calculation.

We partition the data into the two resulting regions, and the repeat the splitting process on each of the regions.

This is then repeated. However, when do we stop?

Ideas for stopping

If the tree is grown too large, down to the bottom, then there will be one 'leaf' for each data point and the $\hat{c}_m$ will match the Y_i.

Such a model is not a summary of the data and overfits.

If the tree is too small, i.e. not many splits, then we might miss important structure in the data.

This is underfitting.

We could grow the tree and stop when the next split does not substantially improve the sum of squares criterion.

However, this can quickly be shown to be a poor strategy, as the split-after-next could suddenly be an effective one, with a large reduction of SSQ.

Better Idea: Cost-complexity pruning

The usual strategy is: grow a large tree T_0, only stopping when the leaves contain five (X_i, Y_i) or fewer.

Then we *prune* the tree, using *cost-complexity pruning*.

Define a subtree $T \subset T_0$, to be any tree that can be obtained by pruning T_0. That is, collapsing any number of its internal (non-terminal) nodes.

We index terminal nodes (leaves) by m, with node m representing region R_m.

Let $|T|$ be the number of terminal nodes (leaves) of tree T.

Better Idea: Cost-complexity pruning — 2

Let n_i as previously described (number of $X_i \in R_m$).

Then $\hat{c}_m = n_m^{-1} \sum_{X_i \in R_m} Y_i$, and

$Q_m(T) = n_m^{-1} \sum_{X_i \in R_m} (Y_i - \hat{c}_m)^2$.

Then, the cost-complexity criterion is

$$C_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha |T|. \tag{7}$$

The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ that minimizes $C_\alpha(T)$.

Cost-complexity pruning: how does it work?

If T is a small tree, then $|T|$ is small, but $\sum_{m=1}^{|T|} n_m Q_m(T)$ is large (not great fit).

If T is a large tree, then $|T|$ is large, but $\sum_{m=1}^{|T|} n_m Q_m(T)$ is small (overfit).

So, with minimizing $C_\alpha(T)$ we want to hit the sweet spot and the 'rate of interchange' between $|T|$ and $\sum_{m=1}^{|T|} n_m Q_m(T)$ is governed by α.

So, if $\alpha = 0$, then the full tree is fitted.

If α is very large, then a small tree gets fitted.

Weakest link pruning

We create a sequence of trees.

Start with the full tree, T_0.

Then, successively collapse the internal node with the smallest per-node increase in $\sum_{m=1}^{|T|} n_m Q_m(T)$.

Until we end up with a single node tree.

It can be shown that this sequence contains tree T_α, the tree that minimises $C_\alpha(T)$.

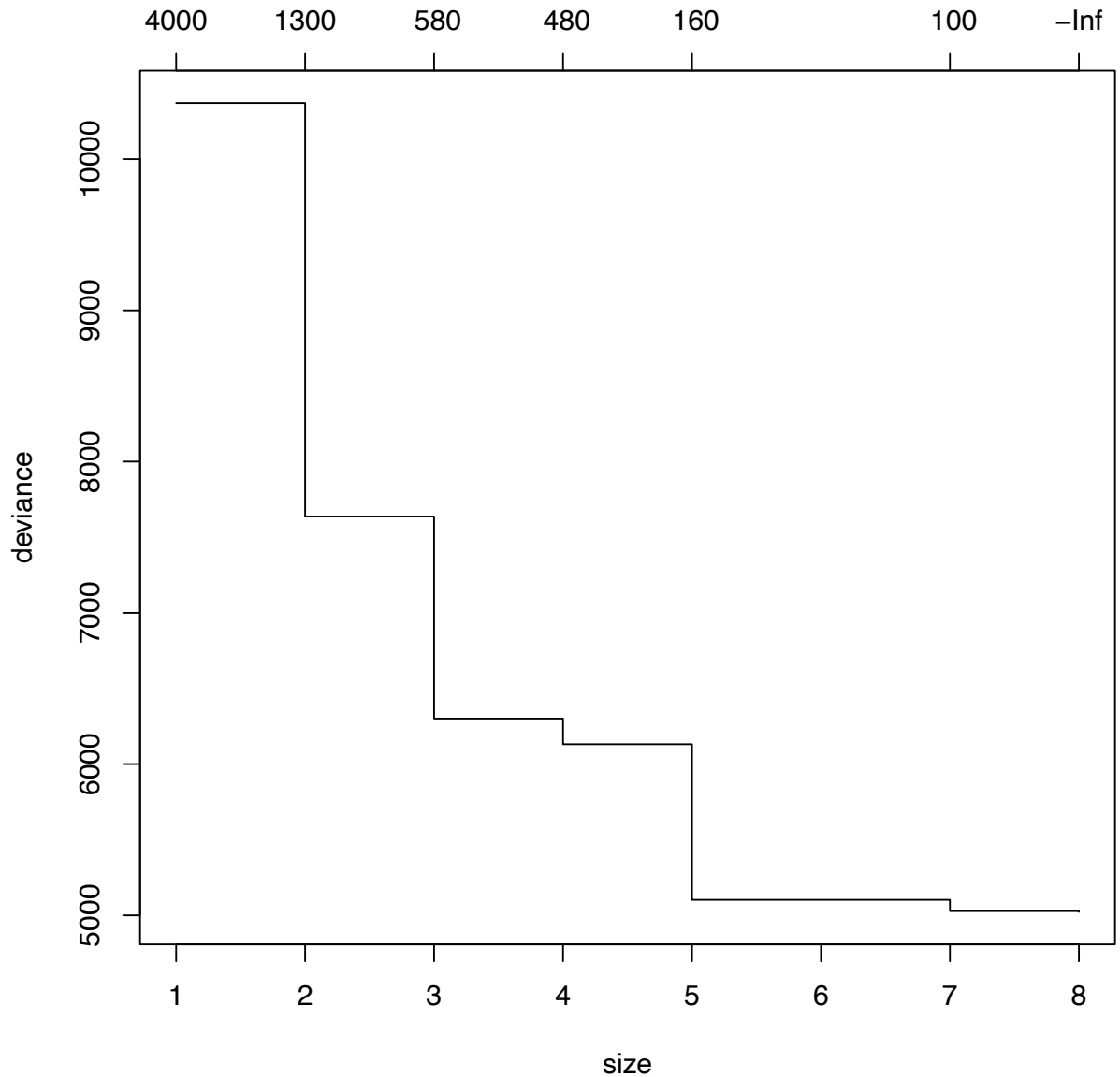
Estimation of α is found usually by five- or ten-fold cross-validation. The final tree is $T_{\hat{\alpha}}$.

Pruning and Cross-validation example

```
library("tree")  # Use the tree library, not rpart
kids.train.tree <- tree(weight ~ height + age,
    data=childhwDF, subset=train)

# This tree turns out to be the same as the one from rpart,
# but I think that this is not always the case

kids.train.tree.cv <- cv.tree(kids.train.tree)
plot(kids.train.tree.cv)
```

The plot suggests we choose a big tree.

Pruning and Cross-validation



Classification Trees

Here, the outcome is not a 'number', but a discrete target, e.g. category $1, 2, \ldots, K$.

E.g. you have a number of attributes about you and the categories are the movies you might like to watch (e.g. historical, comedy, romantic, documentary, thriller, horror, scifi, etc) then can build *recommender system* to recommend new movies to people.

Few changes from regression trees. For regression we used $Q_m(T)$, SSQ not appropriate for classification.

However, we can use

$$\hat{p}_{m,k} = n_m^{-1} \sum_{X_i \in R_m} \mathbb{I}(Y_i = k), \tag{8}$$

the proportion of class k observations in node m.

Classification Trees — 2

We classify the observations in node m to class $k(m) = \arg\max_k \hat{p}_{m,k}$, the majority class in node m.

Then we can use the following alternatives for $Q_m(T)$:

Misclassification error:
$n_m^{-1} \sum_{i \in R_m} \mathbb{I}\{Y_i \neq k(m)\} = 1 - \hat{p}_{m,k(m)}$.

Gini index:
$\sum_{k \neq k'} \hat{p}_{m,k} \hat{p}_{m,k'} = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k})$.

Cross-entropy:
$-\sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k}$.

Pros and Cons of CART

Method is simple — basic idea works well for regression and classification.

Easily explained to non-technical person (just follow tree)

Fast to compute.

Lack of continuity: small changes in input data can result in very different trees.

Inefficient in some cases (e.g. class divide lines on a line not parallel to axes

Oncology Example

> *Application of CART analysis to the prostate biopsy decision results in a significant reduction in unnecessary biopsies while retaining a high degree of sensitivity when compared with the standard of performing a biopsy of all patients with an abnormal PSA or DRE.*

Conclusion of 'Improved Detection of Prostate Cancer using Classification and Regression Tree Analysis', Garzotto, M. *et al.* (2005), *Journal of Clinical Oncology*, **23**, 4322–4329.

PSA: prostate-specific antigen; DRE: digital rectal examination.

CART for Prostate Cancer Data

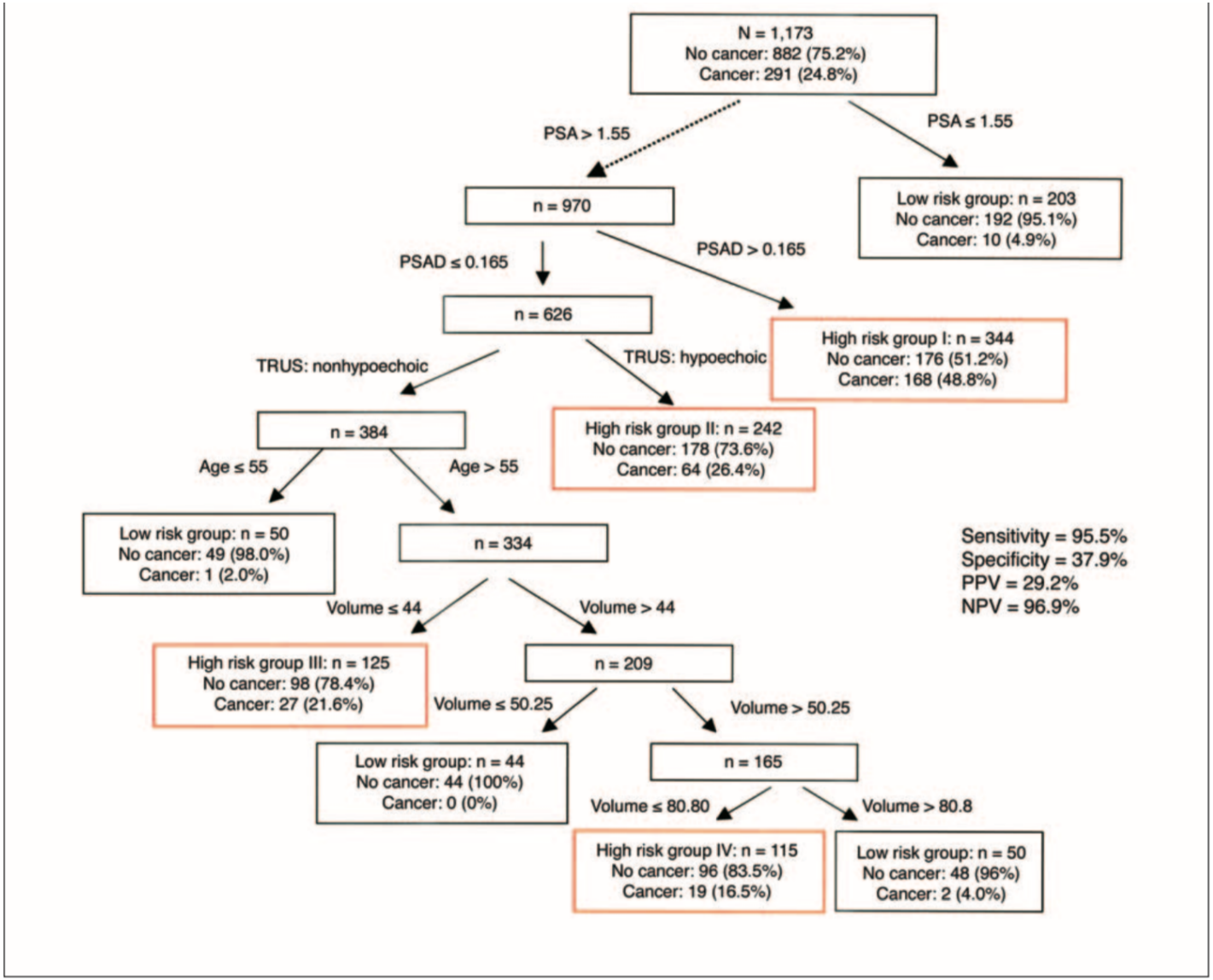


Fig 1. Classification and regression tree analysis in men with a serum PSA level of ≤ 10 ng/mL. High-risk groups are identified by red-colored boxes. The results of the model building dataset are presented with the corresponding results from the validation set in brackets.

Summary

This lecture covered

Basic idea of a tree

An example of a regression tree on the child data

How to grow a tree

How to stop growing

How to choose the 'right' tree

Classification trees

Elements of Statistical Learning: Lecture 19. Model Inference, Averaging, Bootstrap, Bagging and Random Forests

Guy Nason[1]

Department of Mathematics
Imperial College

Bootstrap

In the beginning is a distribution function F.

Suppose we acquire an IID sample $\{X_i\}_{i=1}^n$ from F.

A statistic, θ, is some functional of F. E.g. the mean of X_i

$$\theta(F) = \int x \, dF(x) = \int x f(x) \, dx. \tag{1}$$

We can estimate $\theta(F)$ by replacing F by the empirical distribution function:

$$\hat{F}_n(x) = n^{-1} \sum_{i=1}^{n} \mathbb{I}(X_i \leq x). \tag{2}$$

From earlier courses (or books) that $F_n(x) \to F(x)$ in several modes of convergence (e.g. almost surely, uniformly)

Bootstrap — 2

We can simulate from $\hat{F}_n(x)$ as it puts mass of $1/n$ on each of the $X_1, \ldots X_n$.

So, let $X_1^{(1)}, \ldots, X_n^{(1)}$ be a random sample from $X_1, \ldots, X_n$ with replacement.

So, $X_1^{(1)}$ is picked at random from $X_1, \ldots, X_n$, each with probability $1/n$, then replaced, and then this repeated n times, to get $X_2^{(1)}, X_3^{(1)}, \ldots$

Call $\{X_i^{(b)}\}_{i=1}^n$ the bth bootstrap sample and let their empirical distribution be $\hat{F}^{(b)}$ for $b = 1, \ldots, B$

Mean example

Let's return to our mean example.

We replace F by $\hat{F}$ in our mean functional (1).

$$\hat{\theta} = \theta(\hat{F}) = \int x d\hat{F}(x) = n^{-1} \sum_{i=1}^{n} X_i, \tag{3}$$

as $d\hat{F}(x)$ puts a point mass of n^{-1} at each X_i.

Or, if you refer a formula to 'magic' then the (distributional) derivative of $\hat{F}$ is

$$d\hat{F}(x) = n^{-1} \sum_{i=1}^{n} \delta(x - X_i) dx, \tag{4}$$

where $\delta(x)$ is the Dirac delta function.

Sampling Distribution

Usually most interested in the sampling distribution of $\hat{\theta}$ around θ.

Classically, we'd assume a distribution for $\{X_i\}$ (eg normal) and use this to work out the distributional properties of $\hat{\theta}$.

E.g. if $X_i \sim N(\mu, 1)$, then $\bar{X} \sim N(\mu, n^{-1})$.

However, $\{X_i^{(b)}\}$ gives us direct information on the sampling distribution, $\hat{F}(x)$, as it is a sample from that distribution and we can repeat the sampling operation as often as we like.

Loosely speaking, we can estimate the distribution of the distribution and get a handle on the sampling distribution of things like $\hat{\theta}$, *without* making strong distributional assumptions on X_i and/or F.
Bootstrap also works, when it is hard to work out the maths

Example: estimating σ from exponential

(Also, read §8.2 in ELS.)

Suppose $X \sim \text{Exp}(1/4)$, so mean and sd are 4 respectively.

```
bsexp <- function (Bsims=100, n=30, the.seed=500)
{
set.seed(the.seed)

x <- rexp(n, rate=1/4)  # Generate sample

bssd <- rep(0, Bsims) # Space for the bootstrap sd

for(b in 1:Bsims) {
    bs.sam <- sample(x, size=n, replace=TRUE)
    bssd[b] <- sd(bs.sam)
    }
```

Example: estimating σ — 2

```
nts <- 10000 # True samples (for comparison)
tsd <- rep(0, nts)
for(i in 1:nts) {
    x <- rexp(n, rate=1/4)
    tsd[i] <- sd(x)
    }
return(list(bssd=bssd, tsd=tsd))
}
```

```
> tmp <- bsexp(Bsims=10000, the.seed=102)
> quantile(tmp$bssd, probs=c(0.025, 0.975)) # uses SINGLE sample
    2.5%    97.5%
2.177554 4.347557
> quantile(tmp$tsd, probs=c(0.025, 0.975))
    2.5%    97.5%
2.329919 6.078311
```

Bagging

Bootstrap (above) was used to discover sampling distribution info.

We can use it to find better estimators also.

Suppose we have regression data (as usual)
$Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and then fit a regression model $\hat{f}(x)$ at input x.

We can produce B bootstrap samples $Z^{(b)}$ for $b = 1, \ldots, B$ and each one giving estimate $\hat{f}^{(b)}(x)$.

The *B*ootstrap *agg*regation, or Bagging, estimate averages these predictions over many bootstrap samples by

$$\hat{f}_{\text{bag}}(x) = B^{-1} \sum_{b=1}^{B} \hat{f}^{(b)}(x). \tag{5}$$

(Actual) bagging

Let $\hat{P}$ be the distribution that puts mass of n^{-1} onto each of the (x_i, y_i).

The "true" bagging estimate is defined by $\mathbb{E}_{\hat{P}}\{\hat{f}^*(x)\}$, where $\hat{f}^*$ is obtained from a set $Z^* = \{(x_i^*, y_i^*)\}_{i=1}^n$, where each $(x_i^*, y_i^*) \sim \hat{P}$.

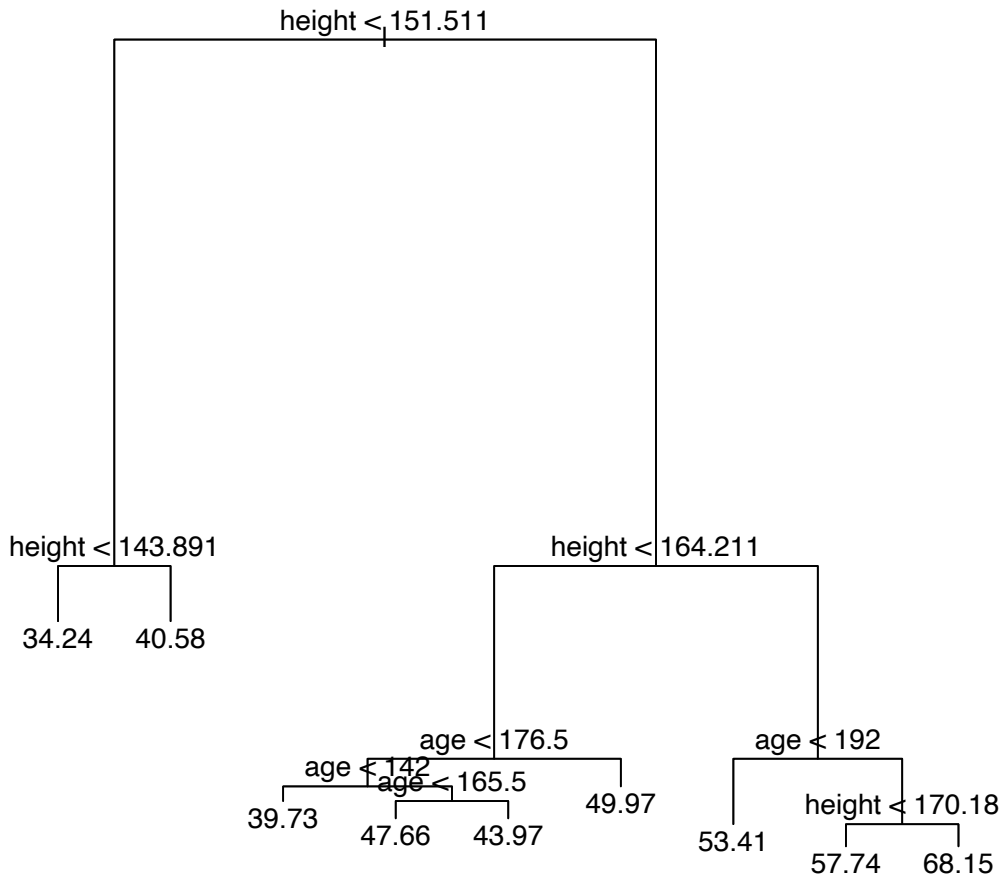
Then $\hat{f}_{\text{bag}}(x)$ is an estimate of the true bagging estimate, approaching as $B \to \infty$.

The bagged estimate will only differ from the original estimate, $\hat{f}(x)$, when the estimator is a nonlinear or adaptive estimator.

If the estimator is linear then $\hat{f}_{\text{bag}}(x) \to \hat{f}(x)$ as $B \to \infty$.

Bagging Regression Trees

```
tmp <- tree(weight ~height+age+sex, data=childhwDF)
plot(tmp) # Tree on actual data
text(tmp)
```



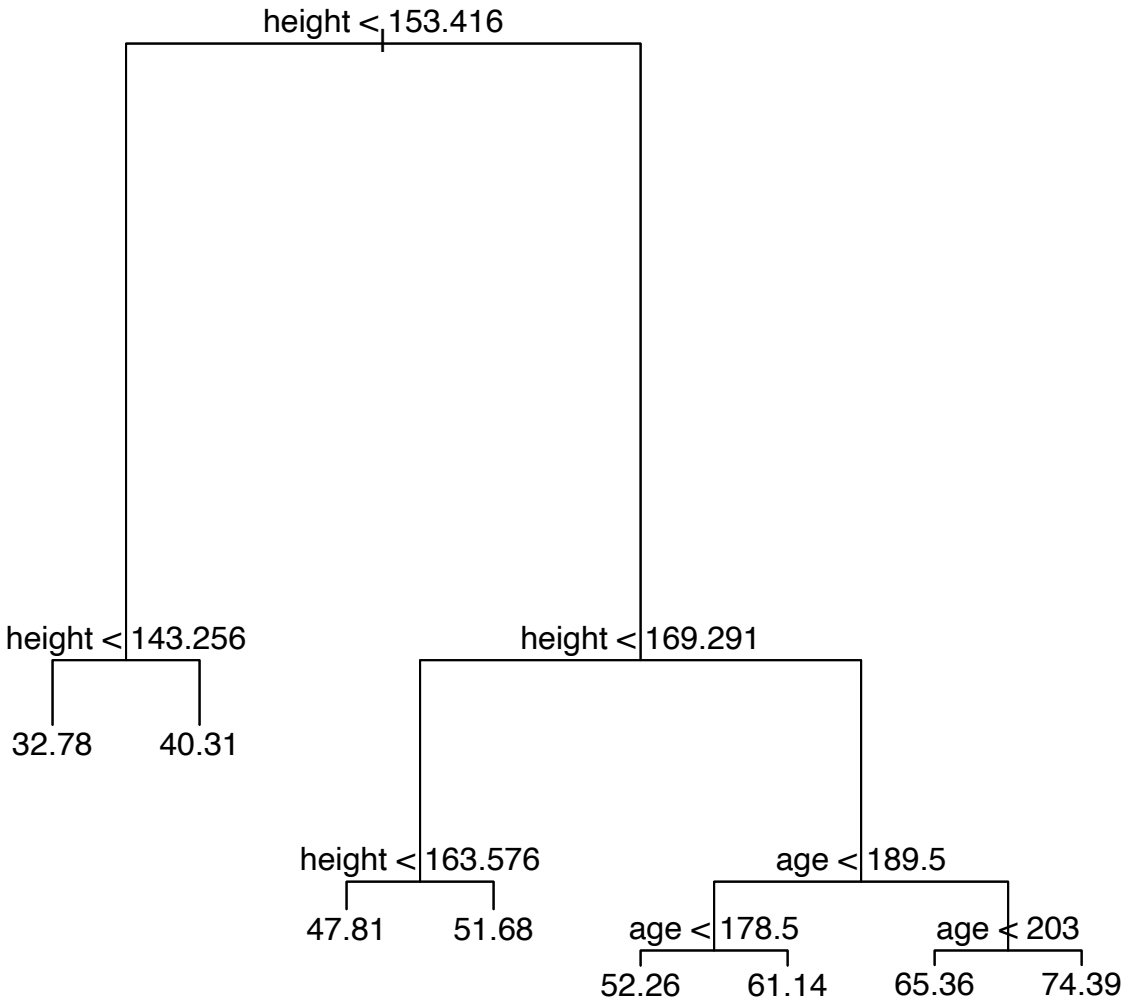
Bootstrap data frame

```
> bsdf <- function (df)
{
n <- nrow(df) # number of cases

# Bootstrap sampling indices
bs.sample <- sample(1:n, size=n, replace=TRUE)

return(df[bs.sample,]) # Return the bootstrapped data frame
}
```

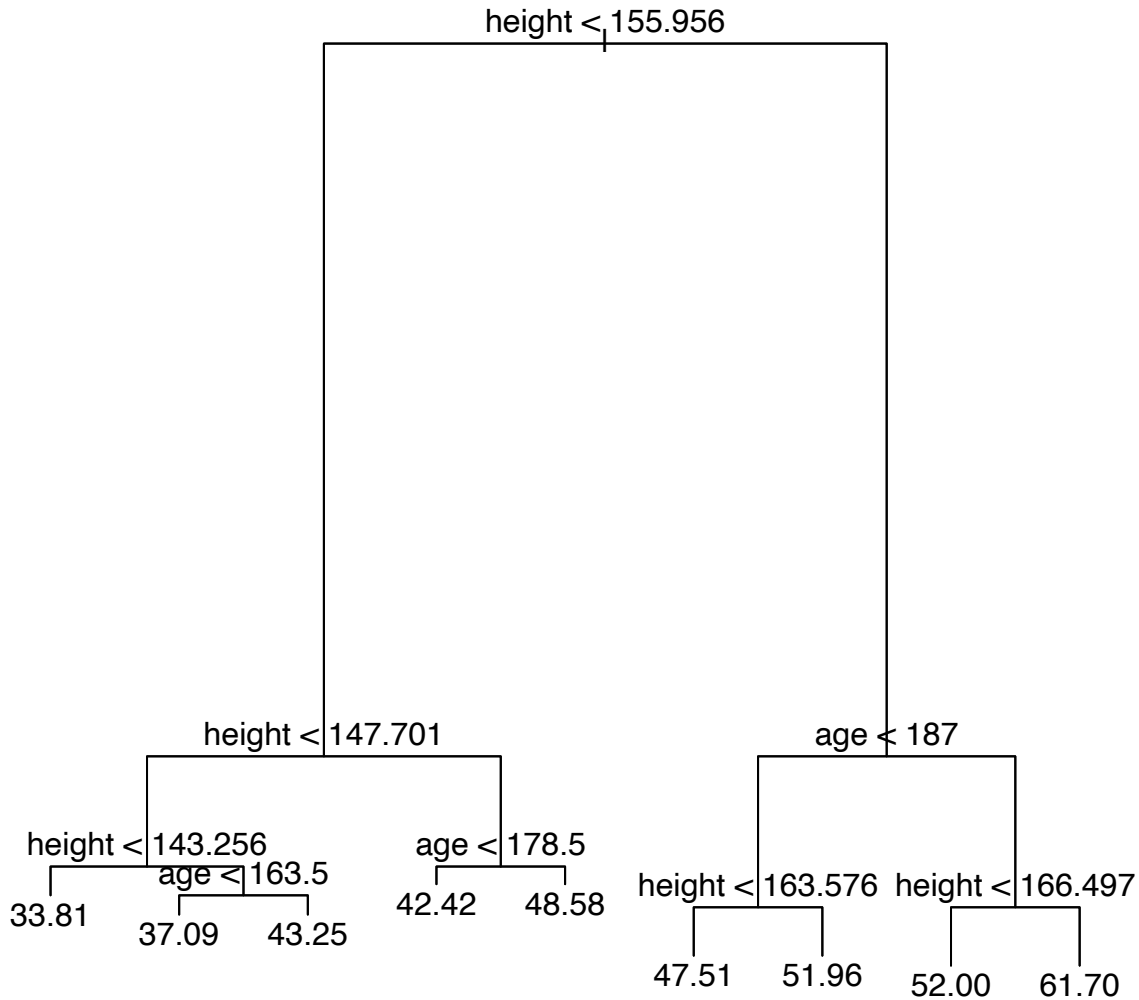
Now bootstrap tree

```
set.seed(100)
tmpdf <- bsdf(childhwDF)
tmp <- tree(weight ~height+age+sex, data=tmpdf)
plot(tmp)
text(tmp)
```

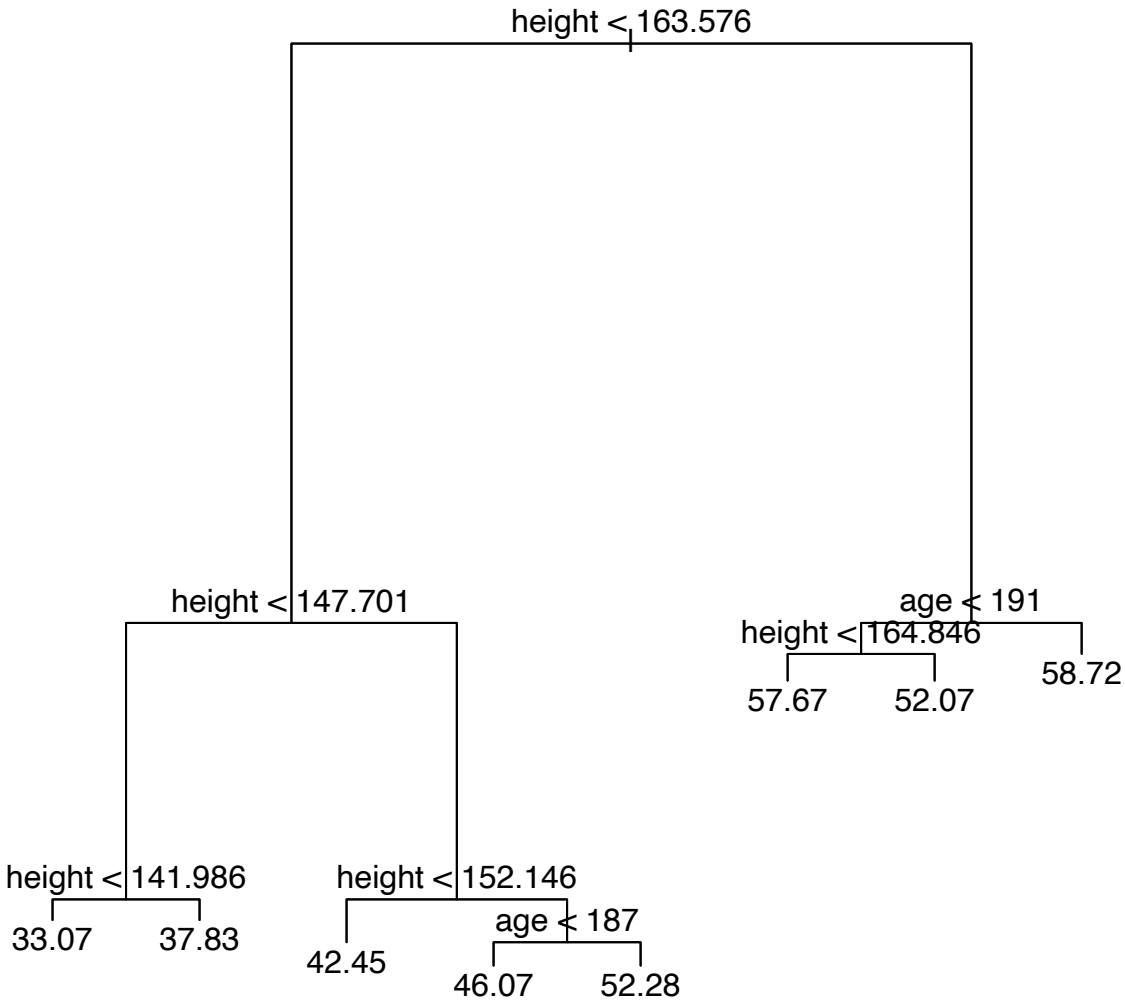
Bagging Regression Trees — 1



Bagging Regression Trees — 2



Bagging Regression Trees — 3



Bagging Regression Trees

So, we generate MANY bootstrapped trees.

Then, given a new case, run that case through every tree.

Then average the results from each tree.

E.g. suppose child: male, age=162 months & height 162.5 then:

Tree 1: right>left>left$\rightarrow$ 47.81
Tree 2: right>left>left$\rightarrow$ 47.51
Tree 3: left>right>right>left$\rightarrow$ 46.07

Bagged prediction over three trees =
$(47.81 + 47.51 + 57.67)/3 = 51.0$.

Usually, do many more trees.

Why does bagging work?

Assume that training observations $\{(x_i, y_i)\}_{i=1}^n$ are drawn independently from distribution P.

Think about the ideal bagging estimator $f_{\text{ag}}(x) = \mathbb{E}_P \hat{f}^*(x)$.

x is fixed and the bootstrap data (x_i^*, y_i^*) is drawn from P, *NOT* the data.

So, it's not a practical estimate, but useful to use to think about the problem.

Now consider ...

Why does bagging work? — 2

$$\begin{aligned}
\mathbb{E}_P\left[\{Y-\hat{f}^*(x)\}^2\right] &= \mathbb{E}_P\left[\{Y-f_{\text{ag}}(x)+f_{\text{ag}}(x)-\hat{f}^*(x)\}^2\right] && (6)\\
&= \mathbb{E}_P\left[\{Y-f_{\text{ag}}(x)\}^2\right]+\mathbb{E}_P\left[\{\hat{f}^*(x)-f_{\text{ag}}(x)\}^2\right]\\
&\geq \mathbb{E}_P\left[\{Y-f_{\text{ag}}(x)\}^2\right]. && (7)
\end{aligned}$$

The cross term

$$\mathbb{E}_P\left[\{Y-f_{\text{ag}}(x)\}\{f_{\text{ag}}(x)-\hat{f}^*(x)\}\right]=0, \qquad (8)$$

because the two terms are statistically independent as $\hat{f}^*(x)$ is generated by the resampling process and the first term involves P and $\mathbb{E}_P\{\hat{f}^*(x)\}=f_{\text{ag}}(x)$.

Hence, the true population aggregate never increases mean squared error, which suggests bagging will often decrease mean squared error.

Bagging Trees

The bootstrap trees look similar.

In fact, they tend to be highly correlated (especially, when there are 'dominant' variables).

Suppose $W_1, \ldots, W_B$ are a set of independent random variables, with variance σ^2.

Let $\bar{W} = B^{-1} \sum_{b=1}^{B} W_b$. Then

$$\text{var}(\bar{W}) = B^{-2} \text{var}(\sum_{b=1}^{B} W_b) = B^{-2} \sum_{b=1}^{B} \text{var}(W_b) = B^{-2} \sum_{b=1}^{B} \sigma^2 = B^{-1}\sigma^2, \tag{9}$$

and $\text{var}(\bar{W}) = B^{-1}\sigma^2 \to 0$ as $B \to \infty$.

Bagging Trees — 2

Now suppose $\text{cov}(W_b, W_d) = \sigma^2\rho > 0$ for $b \neq d$, then

$$
\begin{aligned}
\text{var}(\bar{W}) &= B^{-2}\,\text{var}\Big(\sum_{b=1}^{B} W_b\Big) = B^{-2}\,\text{cov}\Big(\sum_{b=1}^{B} W_b, \sum_{d=1}^{B} W_d\Big) && (10)\\
&= B^{-2}\sum_{b=1}^{B}\sum_{d=1}^{B}\text{cov}(W_b, W_d) && (11)\\
&= B^{-2}\left\{\sum_{b=1}^{B}\text{cov}(W_b, W_b) + \sum_{b=1}^{B}\sum_{d=1, d\neq b}^{B}\text{cov}(W_b, W_d)\right\}\\
&= B^{-2}\left\{B\sigma^2 + B(B-1)\rho\sigma^2\right\} && (12)\\
&= \sigma^2/B + \rho\sigma^2 - \rho\sigma^2/B = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2, && (13)
\end{aligned}
$$

which does not tend to 0 as $B \to \infty$. So, correlated trees limits the effectiveness of bagging.

Random Forests

How can we make the trees be random and look less similar?

Essentially, we add an extra step into the tree construction.

At each tree split decision, we randomly select a set of variables to split on.

If a couple of variables are dominant, then occasionally they do not get considered to split.

Even thought, in regular tree construction, they might be good choices.

See Algorithm 15.1 in ELS.

Random Forests Algorithm 15.1

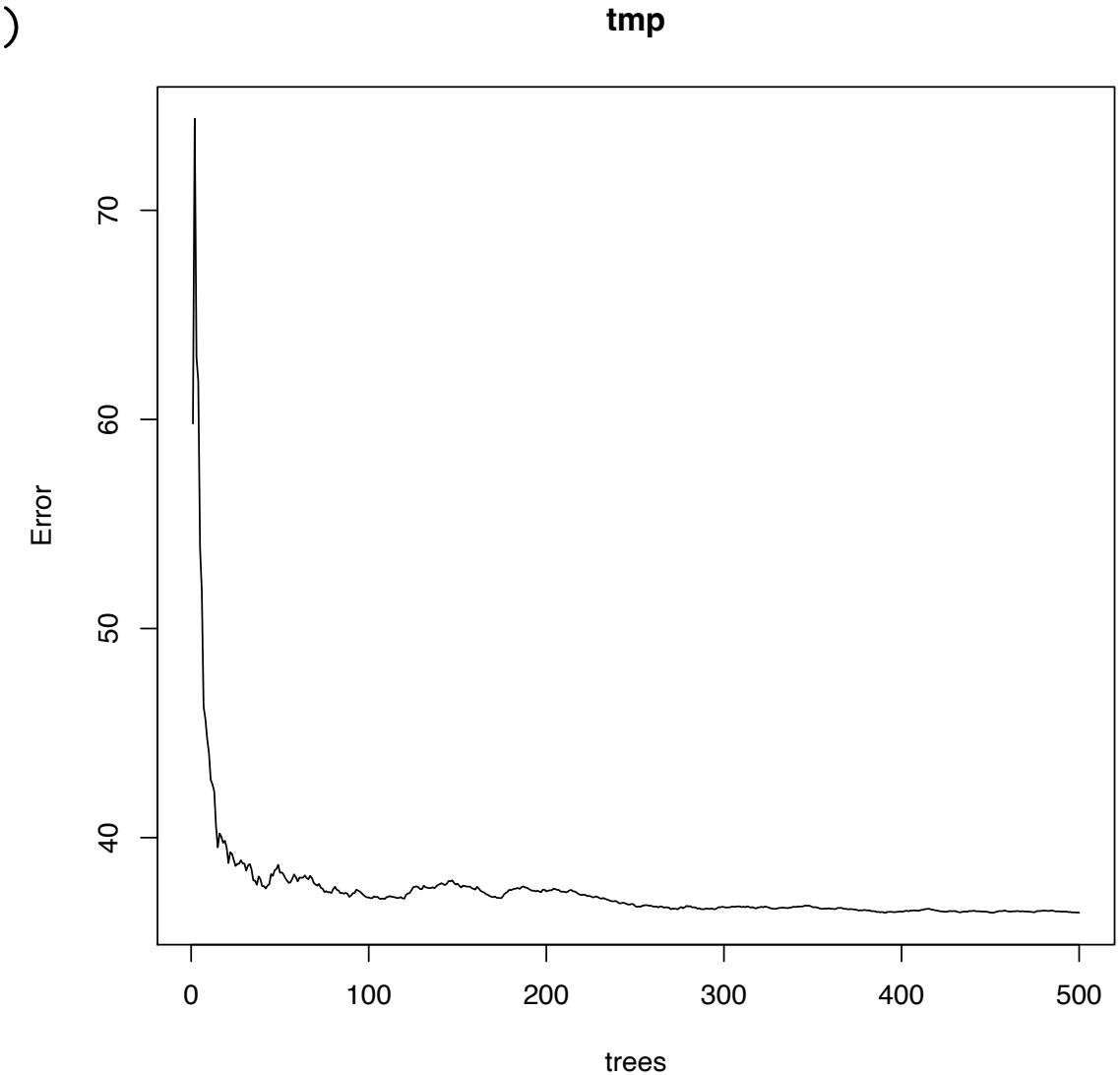
1. For $b = 1, \ldots, B$:
   - 1.1 Draw a bootstrap sample Z^* of size n from the training data.
   - 1.2 Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
     - 1.2.1 Select m variables at random from the p variables.
     - 1.2.2 Pick the best variable/split-point among the m.
     - 1.2.3 Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

For a prediction at new point x:
Regression: $\hat{f}_{\text{rf}}^{B}(x) = B^{-1} \sum_{b=1}^{B} T_b(x)$.

Classification: let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^{B}(x) = \text{majority vote}\{\hat{C}_b(x)\}_{b=1}^{B}$.

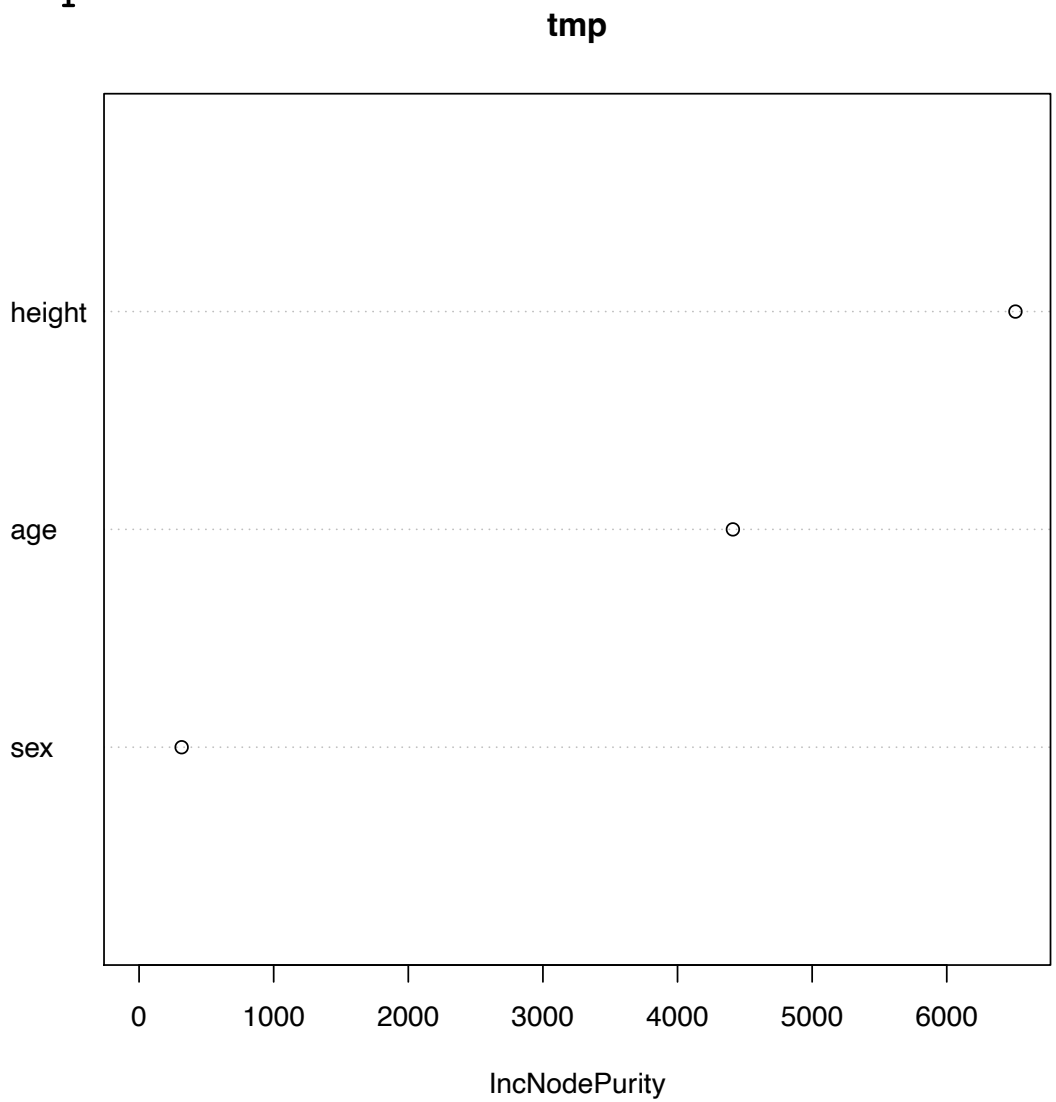
Child HW example (small number of vars)

```
library("randomForest")
tmp <- randomForest(weight~sex+age+height, data=childhwDF)
plot(tmp)
```



Child HW example: variable importance plot

```
varImpPlot(tmp)
```



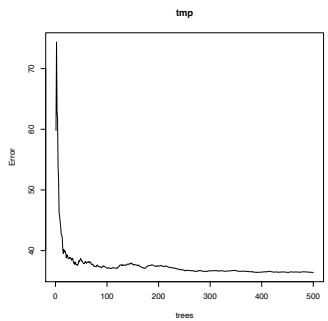
Random Forests — Out of bag samples

In cross-validation, we removed a point i, computed the estimator without that point, then compared the fit without the point $\hat{f}^{(-i)}$ to y_i.

With random forests, we can do that as we go along and as we add trees, get a continually updated plot of the prediction error with those trees.

How? Essentially compare (x_i, y_i) to trees produced by bootstrap samples that (randomly) did not contain that point.

This gives plots like the first one:

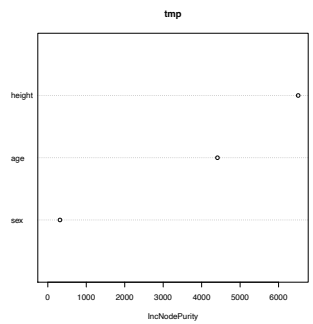


Random Forests — Variable Importance Plot

At each split in the tree, the improvement in the split-criterion is added to the importance measure for that variable, and then this accumulated over all trees in the forest.

Alternatives can involve prediction accuracy. E.g. by threading OOB samples down, you can measure the predictive performance of variables, and then averaged across the forest.

This gives plots like the 2nd one:



Proximity Maps

Start with $n \times n$ proximity matrix — all set to zero.

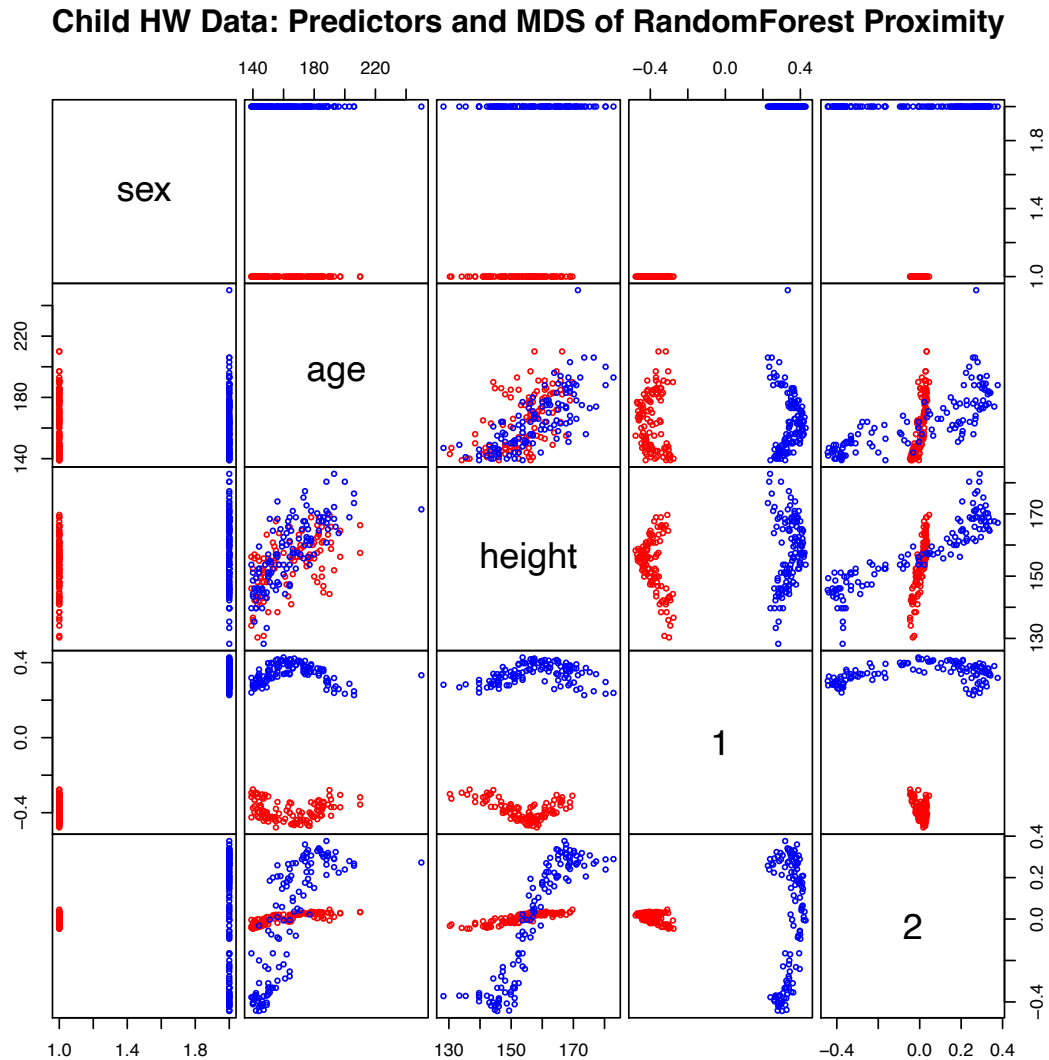
Indicates the similarity/proximity between individuals.

As the OOB samples are propagated down, if they end up in the same terminal node, then increment their proximity count by one.

Turn proximities into dissimilarities ($d = 1 - p/\text{maxp}$), then use scaling to plot how the individuals relate to each other *from the point of view of the random forest algorithm*).

```
## Do MDS on 1 - proximity:
childhw.mds <- cmdscale(1 - tmp$proximity, eig=TRUE)
op <- par(pty="s")
pairs(cbind(childhwDF[,1:3], childhw.mds$points), cex=0.6,
    gap=0, col=(c("red", "blue"))[childhwDF[,1]],
    main="Child HW Data: Predictors and MDS of RandomForest Prox
par(op)
```

Child HW example: proximity plot



Proximity axis 1 has strong association with gender.

Summary

This lecture covered

- **Bootstrap** the basics and an example
  - **Bagging** using the bootstrap samples to estimate
- Bagging Trees
- Why Bagging Works
- Random Forests working with 'uncorrelated' trees
- OOB, Variable importance, Proximity plots

Elements of Statistical Learning: Lecture 20. Boosting. Data Ethics

Guy Nason[1]

Department of Mathematics
Imperial College

Boosting: Setup

Technique to combine the outputs of many "weak" classifiers to make a strong one.

Suppose we have a two class problem with output code as $Y \in \{-1, 1\}$.

Given a vector of predictor variables, X, a classifier $G(X)$ produces a prediction taking one of the two values -1 or 1.

The error rate on the training sample is

$$\bar{err} = n^{-1} \sum_{i=1}^{n} \mathbb{I}\{y_i \neq G(x_i)\}, \tag{1}$$

If you classifier is just random guessing, then the error rate is 50%. A weak classifier's error rate might be something like 45%.

Boosting: Idea

We will build a sequence of classifiers, $G_m(x)$, $m = 1, \ldots, M$, where $G_1(x) = G(x)$.

We obtain $G_{m+1}(x)$ from $G_m(x)$ by focusing weight on those observations that did not get classified well by $G_m(x)$. So, it tries harder on the more difficult ones.

We combine the results from all the classifiers using

$$G^*(x) = \text{sgn}\left\{\sum_{m=1}^{M} \alpha_m G_m(x)\right\}, \tag{2}$$

where the $\{\alpha_m\}_{m=1}^{M}$ are computed by the boosting algorithm.

Boosting Weights

We apply weights $w_1, \ldots, w_n$ to each of the training observations (x_i, y_i), $i = 1, \ldots, n$.

Initially, all weights are set to n^{-1} — equal weights.

For each successive iteration, $m = 2, \ldots, M$ the weights are modified, and the classifier is reapplied to the weighted observations.

At step m, those observations that were previously incorrectly classified get their weight increased.

So, difficult to classify observations, get progressively more weight.

AdaBoost.M1 (Alg. 10.1 in ELS)

1. Initialize observation weights $w_i = n^{-1}$, $i = 1, \ldots, n$.
2. For $m = 1, \ldots, M$:
   - 2.1 Fit classifier $G_m(x)$ to training data using weights $\{w_i\}$.
   - 2.2 Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{n} w_i \mathbb{I}\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^{n} w_i}. \tag{3}$$
   - 2.3 Compute $\alpha_m = \log\{(1 - \text{err}_m)/\text{err}_m\}$.
   - 2.4 Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot \mathbb{I}\{y_i \neq G(x_i)\}]$, $i = 1, \ldots, n$.
3. Output $G^*(x) = \text{sgn}\left\{\sum_{m=1}^{M} \alpha_m G_m(x)\right\}$.

Boosting Example

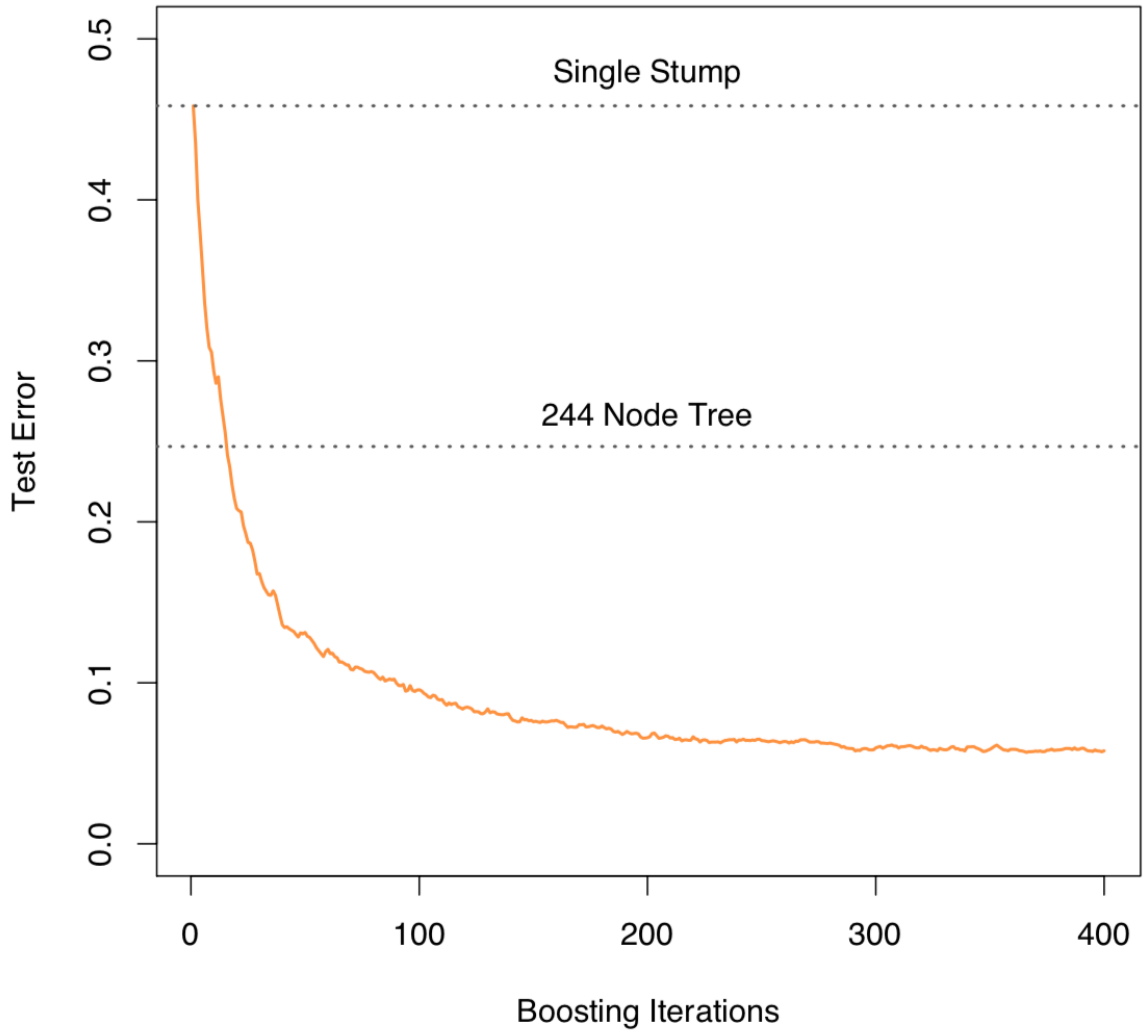
Suppose $X_1, \ldots, X_{10}$ are independent standard Gaussians $N(0, 1)$.

Now define

$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > 9.34, \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

They use a basic (silly) two branch tree, which has classification error rate of 45.8%. Not much better than random guessing.

Boosting Example — 2



More details in §10 in ELS.

Ethics

From Wikipedia:
"Ethics is a branch of philosophy that involves systematizing, defending and recommending concepts of right and wrong conduct".

Dictionary definition of *ethical behaviour*: "Acting in ways consistent with what society and individuals typically think are good values".

Collectively, we all benefit from ethical behaviour.

Horror Stories — Cambridge Analytica [from Wiki]

"Revealed — 50 million Facebook profiles harvested for Cambridge Analytica in major data breach"

"Facebook's week of shame" /pause

"Cambridge Analytica . . . used personal information taken without authorisation in early 2014 to build a system that could profile individual US voters, in order to target them with personalised political advertisements".

Used a Facebook app "This is your digital life" used by 270k Facebook users and then acquired information on a further 87 million users who were friends of the original 270k.

Other breaches: "Cruz Crew" mobile app that tracked physical movements and contacts.

Cambridge Analytica — Major investigation

By Channel 4, New York Times, the Observer, etc.

Interviewed the CEO — undercover recording that CA used honey traps, bribery stings amd prostitutes to discredit politicians, funded companies to entrap political opponents into bad behaviours.

UK-EU membership referendum involvement.

"No campaign contributions, in cash or in kind by CA were reported to UK electoral authorities." "In March 2018, Brittany Kaiser (CA's former director of business development, revealed that the company misled the public and MPs over its links with Leave.EU. She said she felt she had lied by supporting CA's company line." [Wikipedia]

Netflix + Prize competition [from Wikipedia]

Netflix ran a competition and released data on movie preferences.

Training + test sets + had to develop algorithm to beat Netflix's.

Grand Prize of $1M.

Data sets were constructed/anonymised to preserve privacy.

However, in 2007 two researchers from the University of Texas were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database.

In 2009, four Netflix users sued Netflix alleging that they had violated the Video Privacy Protection Act because they released the data sets.

Netflix cancelled the next competition and came to a financial settlement with the plaintiffs!

Reproducibility Crisis [Wiki]

"The field of Recommender systems has been impacted by the Replication crisis as well. A systematic analysis of publications applying deep learning or neural methods to the top-k recommendation problem, published in top conferences (SIGIR, KDD, WWW, RecSys), has shown that on average less than 40% of articles are reproducible, with as little as 14% in some conferences. Overall the study identifies 18 articles, only 7 of them could be reproduced and 6 of them could be outperformed by much older and simpler properly tuned baselines. The article also highlights a number of potential problems in today's research scholarship and calls for improved scientific practices in that area, (Ferrari Dacrema, M. *et al.* 2019)."

Ferrari Dacrema, M., Cremonesi, P. and Dietmar, J. (2019) Are we really making much progress? A worry analysis of recent neural recommendation approaches. *Proceedings of the 13th ACM Conference on Recommender Systems, ACM: 101–109.*

Horror Stories: Action Fraud (The Times, Sept 2019)

"The Home Office is manipulating crime figures by instructing Action Fraud to dismiss as many as tens of thousands of legitimate cases"

"police service is wrongly failing to record cases of identity theft"

"as many as 50000 reported frauds every year are not included in official crime statistics"

The UK has a unique body, the UK Statistics Authority (UKSA) and the Office for Statistical Regulation, which issues a code of practice for official statistics (more later).

The UKSA essentially said, approximately, don't believe police crime figures too much, but believe the national Crime Survey for England and Wales (independently run survey). Not very helpful. Why release police statistics then?

Horror Stories: Action Fraud (The Times, Sept 2019)

In fact, UKSA has said:

> *"Incidents of fraud referred to the NFIB [police] by Action Fraud, CIFAS and UK Finance will include reports from businesses and other organisations. They also tend to be focused on the more serious cases."*

Implication is that fraud cases of ordinary people are being ignored.

A Home Office report itself has a section entitled "Accuracy of Quarterly Data and Negative Figures".

How can there be negative numbers of fraud cases?

Number of cases is reported monthly. They score +1 for each case coming in and -1 for every one they pass to Action Fraud. In some months, more cases go out than come in, hence the negative figures. This seems wrong.

Horror Stories: Action Fraud (The Times, Sept 2019)

Especially, since these numbers might be aggregated.

I have personally written to OSR about this - anybody can do this.

Another issue: Government depts are bound by the Code of Practice.

External contractors (e.g. ActionFraud) are not, but Government Depts should satisfy themselves as to the quality of the statistics — not sure they are doing this here.

UKSA Code of Practice

https://www.statisticsauthority.gov.uk/publication/code-of-practice/

The Code of Practice for Official Statistics promotes the production and dissemination of official statistics that inform decision making, and supports the continuous improvement of those statistics. It is a concise and specific statement that requires sound judgement and interpretation.

The Code applies to all UK bodies that produce official statistics. It encourages and supports them to maintain their independence and to ensure adequate resourcing for statistical production. It helps producers and users of statistics by setting out the necessary principles and practices to produce statistics that are trustworthy, high quality and of public value.

Compliance with the Code is a statutory requirement on bodies that produce statistics that are designated as National Statistics through the Office for Statistics Regulation's Assessment process.

UKA Code of Practice contd.

While the formal scope of the Code is official statistics, many of the principles and practices are likely to be helpful to producers of other data such as management information and those outside of official statistics such as third sector and private sector organisations.

The Code is consistent with the United Nations Fundamental Principles of Official Statistics and the European Statistics Code of Practice.

Occasionally we recognise an exemption or exception to a specific practice within the Code. A breach occurs when an organisation producing official statistics fails to meet the standards outlined in the Code and where an exemption or exception does not apply.

Corporate Examples: Accenture

`https://www.accenture.com/us-en/insight-data-ethics`

See building digital trust slides

Open Data Institute and Ada Lovelace Institute

Open Data Institute's data ethics canvas

`https://theodi.org/article/data-ethics-canvas/`

The Ada Lovelace Institute

`https://www.adalovelaceinstitute.org`

Simple Guide to Data Ethics

1. Do the right thing.
2. Think things through (tech is complex).
3. Do no harm, don't let others harm.
4. Ask permission/authorisation.
5. Be transparent and open.
6. Do the right thing!