Modern Data Analysis in S-PLUS

Brian Ripley

Workshop on October 23, 1999,
International S-PLUS User Conference, New Orleans

ripley@stats.ox.ac.uk
http://www.stats.ox.ac.uk/∼ripley

Themes

- Illustrate new features of S-PLUS 2000 and `MASS3` library.
- Modest departures from linearity via smooth terms.
- Be robust!
- ‘Eye-balling’ is not good enough for large-scale datasets.
- Build complex analyses from smaller ideas and tools.

Programme

- looking at data
  - density estimation
  - visualization of data matrices
- regression & beyond
  - diagnostics and robust fits
  - bootstrapping
  - smooth terms
- random effects and mixed models
  - methods: (N)LME, GLMM, GEE
  - examples using the `nlme3.x` library

— lunch —

— lunch —

- survival analysis
  - smooth models for hazards
  - adding smooth terms in `coxph` and `survreg`
- multivariate analysis
  - correspondence analysis
  - discriminant analysis
- case study: MRI brain imaging

Looking at Data

Density estimation

Close look at data in one or perhaps two dimensions. Not many uses of density estimation other than visualization.

Kernel methods

are very well known:

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

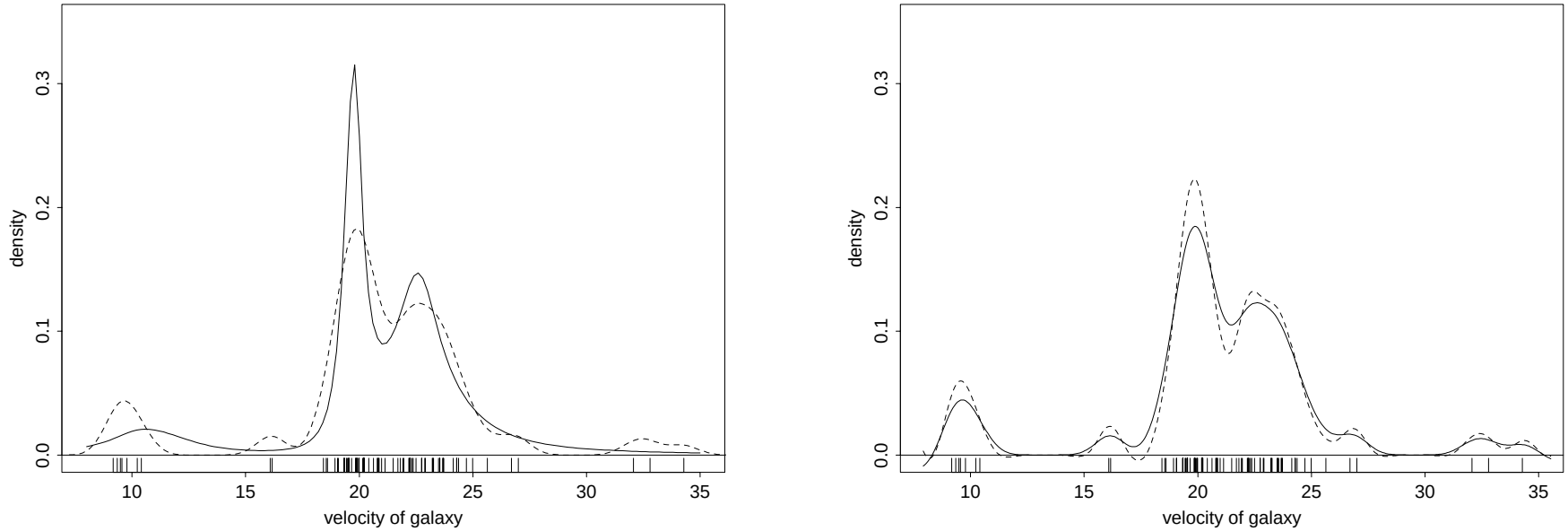
for a sample $x_1, \ldots, x_n$, a fixed kernel $K()$ and a bandwidth b.

Main issue is the choice of bandwidth b. State-of-the-art methods in libraries `MASS` and `KernSmooth` (Matt Wand).

Boundary effects can be severe: no sophisticated S implementations (?)

Local polynomial methods

Do density estimation via smoothing: effectively take a fine grid and count the number of points in each interval, then smooth that.



Left: Logspline (solid line) and kernel density (dashed) estimates for the `galaxies` data. Right: Local polynomial estimates by `locpoly` with linear (solid) and quadratic (dashed) local fits.

Smooth log-densities

There are several closely-related proposals to use a univariate density estimator of the form

$$f(y) = \exp g(y;\theta)$$

for a parametric family $g(\cdot;\theta)$ of smooth functions, most often splines. The fit criterion is maximum likelihood, possibly with a smoothness penalty.

It is necessary to ensure that the estimated density has unit mass, most conveniently by

$$f(y) = \exp g(y;\theta) / \int \exp g(y;\theta)\,\mathrm{d}y$$

The library `logspline` by Charles Kooperberg implements one variant on this theme. This uses a cubic regression spline for g, with smoothness by backwards selection on the knots. There is an AIC-like penalty; the number of the knots is chosen to maximize

$$\sum_{i=1}^{n} g(y_i;\widehat{\theta}) - n \log \int \exp g(y;\widehat{\theta})\,\mathrm{d}y - a \times \text{\#params}, \qquad a = \log n \text{ ?}$$

Splines

Interpolation splines A smooth curve through the data points (x_i, y_i) that is as smooth as possible in the sense

$$\int (f''(x))^2 \, \mathrm{d}x$$

is minimized. Turns out to be a piecewise cubic with breaks at the x_i.

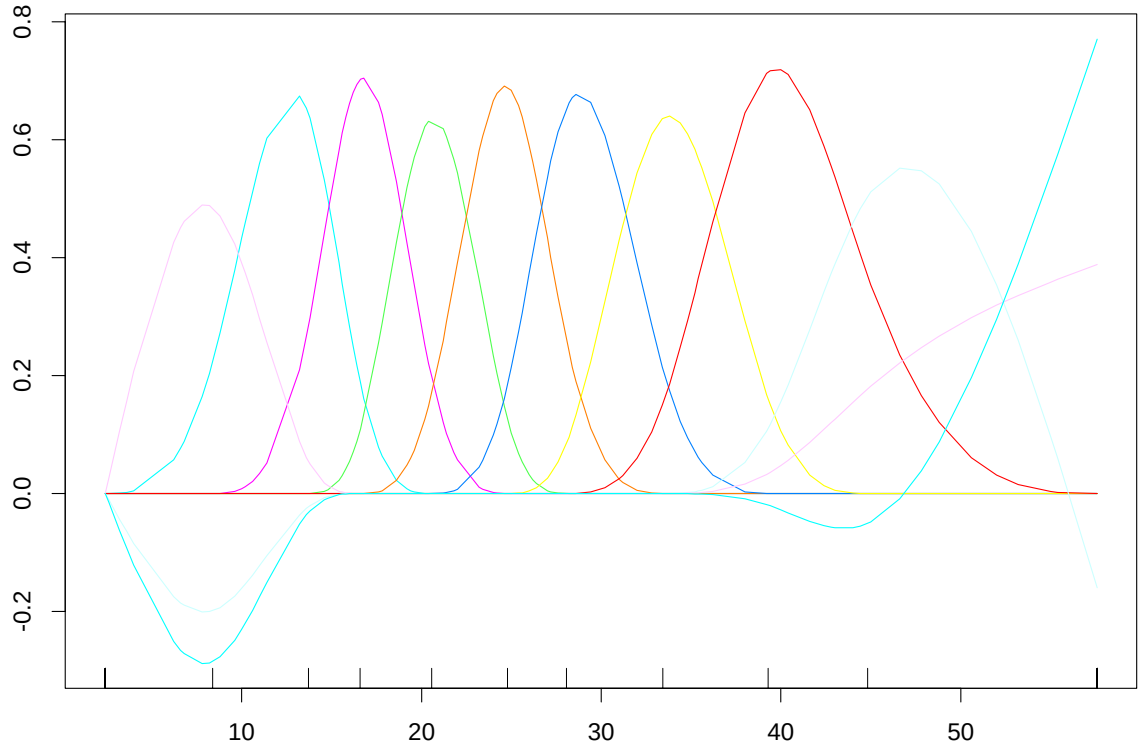
Smoothing splines A *smoothing spline* minimizes a compromise between the fit and the degree of smoothness of the form

$$\sum w_i [y_i - f(x_i)]^2 + \lambda \int (f''(x))^2 \, \mathrm{d}x$$

over all (measurably twice-differentiable) functions f. Again a piecewise cubic with breaks at the x_i, but does not interpolate the data points for $\lambda > 0$ and the degree of fit is controlled by λ.

How smooth? Various forms of cross-validation.

Regression splines Take the idea of cubic splines, but choose the knots to control the smoothness. Get cubic splines by `bs` and `ns`.



Splines created by `ns(times, df=10)`.

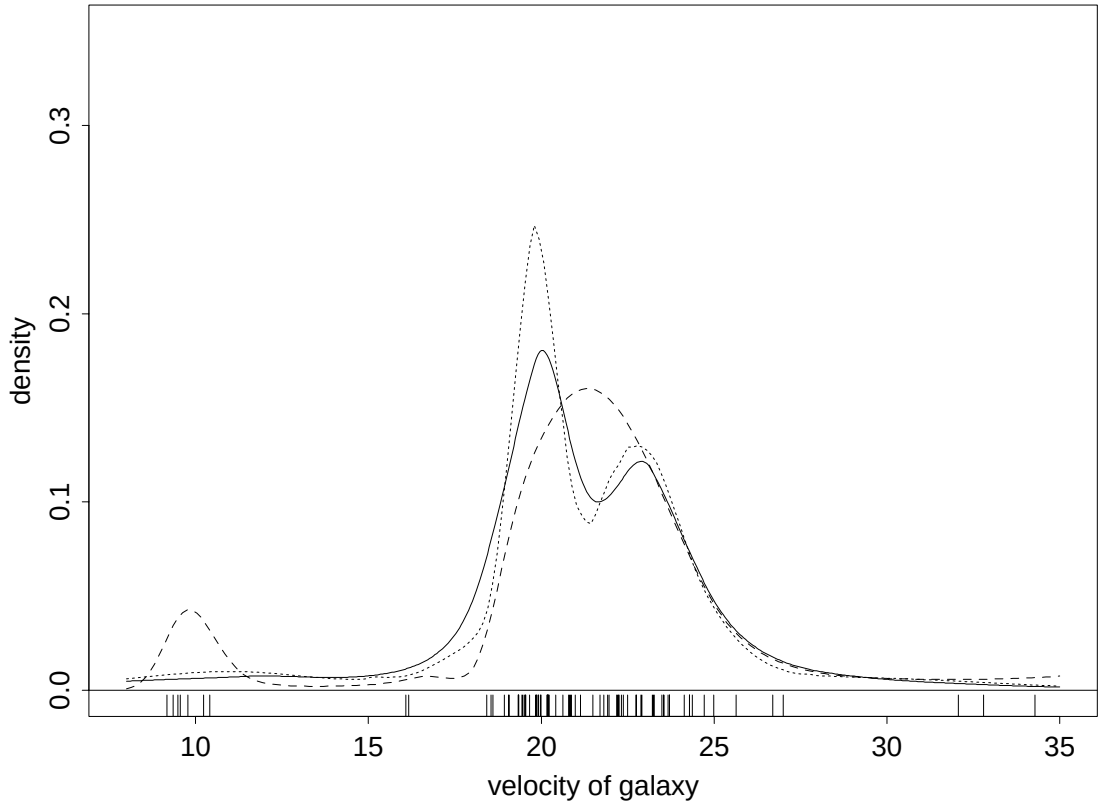
Local likelihood methods

Use a parametric model for $g(y) = \log f(y)$ (e.g. quadratic) fitted locally at x:

$$\sum_{i=1}^{n} K\left(\tfrac{y_i - x}{b}\right) g(y_i; \theta(x)) - n \log \int K\left(\tfrac{y-x}{b}\right) \exp g(y; \theta(x)) \, \mathrm{d}y$$

Once again have to choose the bandwidth b, possibly depending on x.

`locfit` by Clive Loader (+ recent Springer book).



`locfit` density estimates for the `galaxies` dataset. The solid line is the default, the dotted line has a variable bandwidth chosen by AIC, and the dashed line uses a surrogate Poisson model.

Series Estimation

$$\hat{f}(x) = \sum_{j=0}^{\hat{J}} \hat{w}_j \hat{\theta}_j \phi_j(x) + \sum_{j=\hat{J}+1}^{c_{J_M}} I_{\{\hat{\theta}_j^2 > C_T \bar{d} \ln(n)/n\}} \hat{\theta}_j \phi_j(x)$$

then take the positive part, remove small bumps, rescale to unit integral.

Here $(\phi_j(x))$ is an orthogonal basis, $\hat{\theta}_j$ are the 'Fourier' coefficients.

'universal orthogonal series estimator' of
S. Efromovich (1999) *Nonparametric Curve Estimation*, Springer.

Multidimensional analogues

All (except series estimation) generalize easily to 2 or more dimensions, but

- 'bandwidth' now involves shape as well as size
- 'curse of dimensionality' means 'local' is a rather large neighbourhood.
- computation can be very slow.
- what are you going to do with this?

Survival data

Typically has partially observed data, e.g. $X > t$ is all that is known for some cases. Straightforward for likelihood-based methods, but localization is less clear-cut.

Examples this afternoon.

Visualization

Challenge is to explore data in more than two or three dimensions.

via projections

Principal components is the most obvious technique: kD projection of data with largest variance matrix (in several senses). Usually ‘shear’ the view to give uncorrelated axes.

Lots of other projections looking for ‘interesting’ views, for example groupings, outliers, clumping.

Implementation via XGobi for Unix, and for Windows using an X server.

multidimensional scaling

Aim is to represent distances between points well.

Suppose we have distances (d_{ij}) between all pairs of n points, or a *dissimilarity* matrix. Classical MDS plots the first k principal components, and minimizes

$$\sum_{i \neq j} d_{ij}^2 - \widetilde{d}_{ij}^2$$

where $(\widetilde{d}_{ij})$ are the Euclidean distances in the kD space.

More interested in getting small distances right. Sammon (1969) proposed

$$\min E(d, \widetilde{d}) = \frac{1}{\sum_{i \neq j} d_{ij}} \sum_{i \neq j} \frac{(d_{ij} - \widetilde{d}_{ij})^2}{d_{ij}}$$

Implemented in function `sammon` in library `MASS`.

Kruskal and Shepard proposed only to preserve the ordering of distances, minimizing

$$STRESS^2 = \frac{\sum_{i \neq j} \left[\theta(d_{ij}) - \widetilde{d}_{ij}\right]^2}{\sum_{i \neq j} \widetilde{d}_{ij}^2}$$

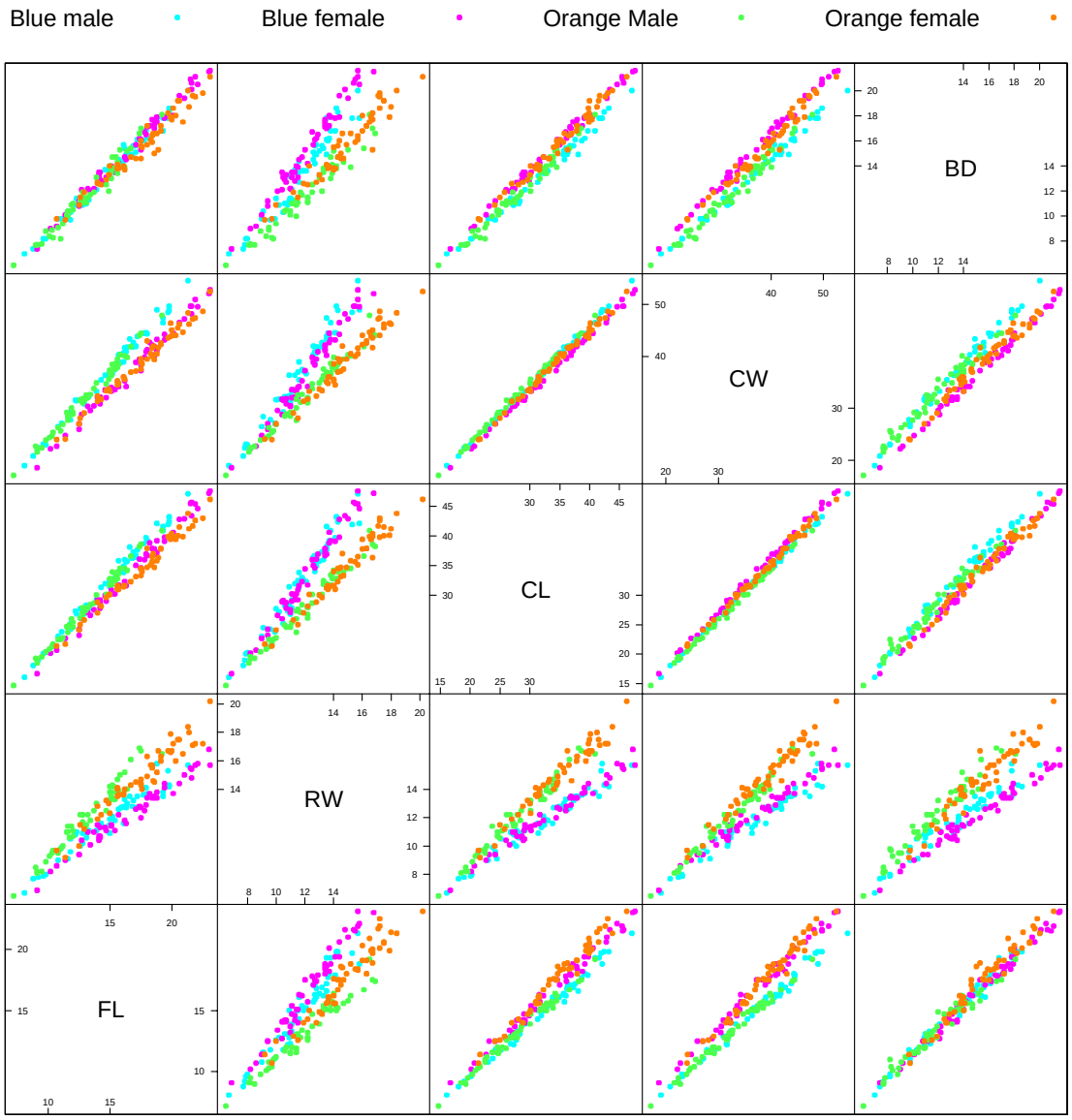
over both the configuration of points and an increasing function θ. Implemented in function `isoMDS` in library `MASS`. (This method is *isotonic*, hence the function name.)

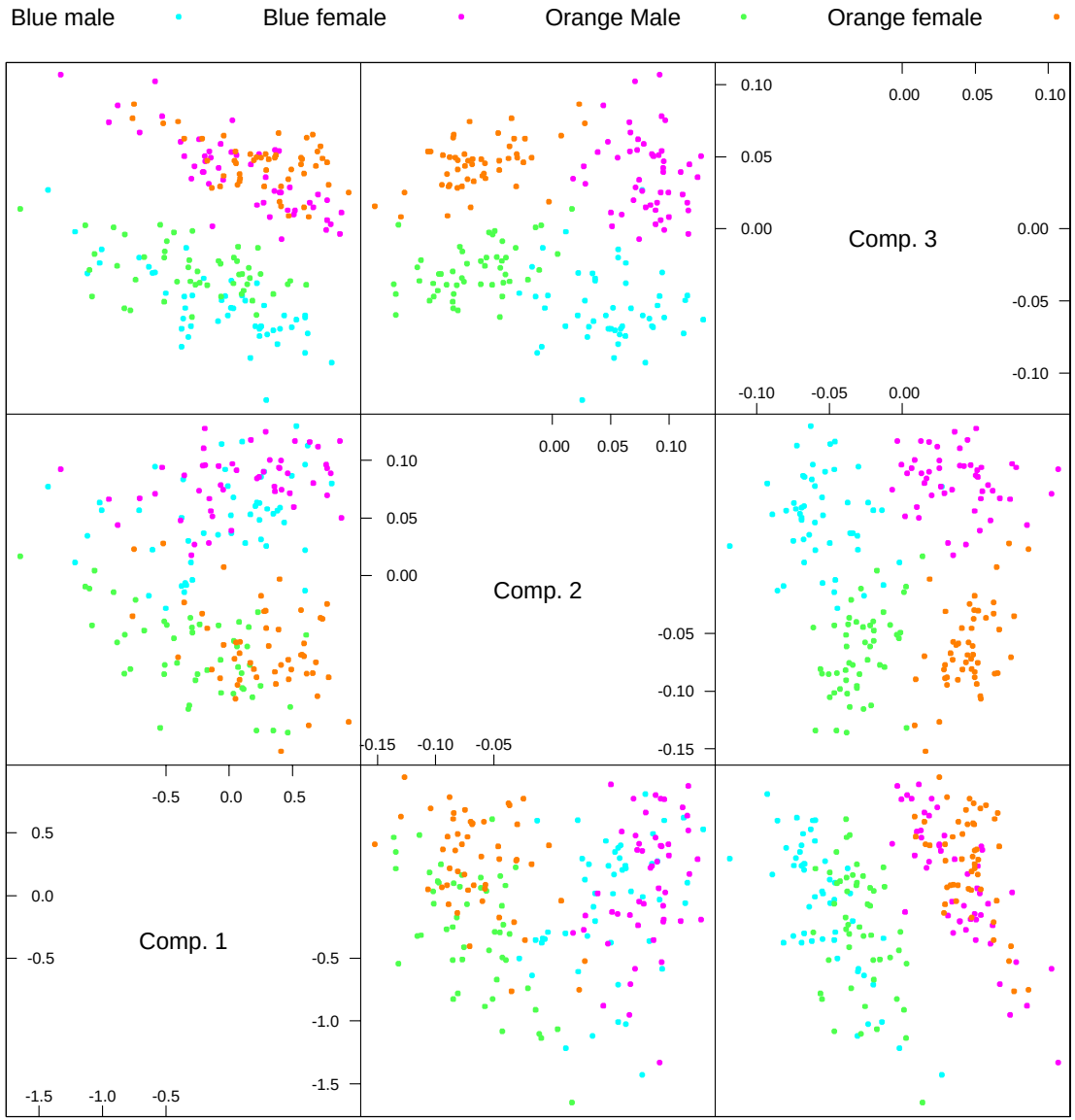
The optimization task is quite difficult and this can be slow.

Leptograpsus variegatus Crabs

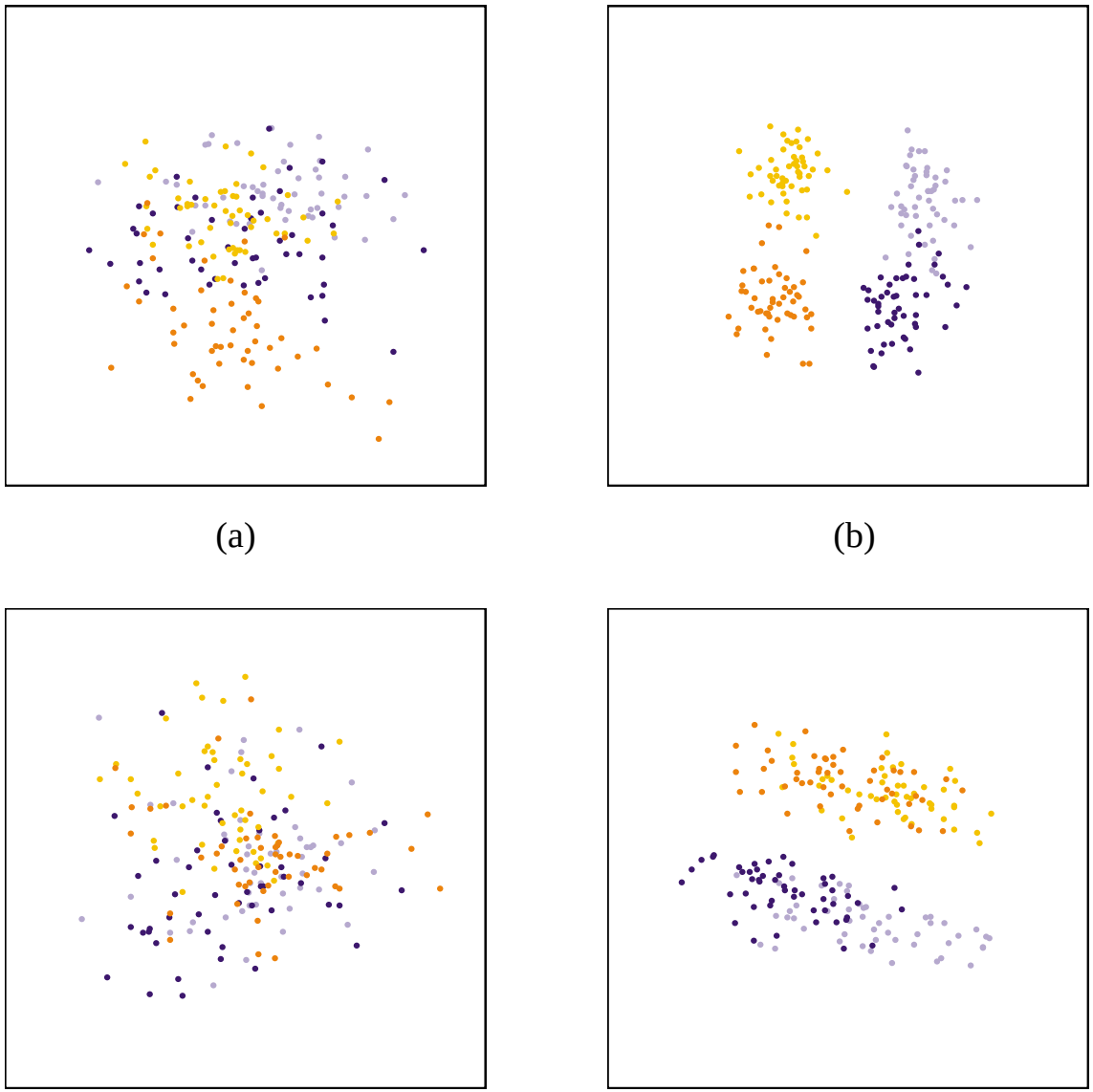
200 crabs from Western Australia. Two colour forms, blue and orange; collected 50 of each form of each sex.

Measurements of carapace (shell) length `CL` and width `CW`, the size of the frontal lobe `FL` and rear width `RW`.


Blue male
Blue female
Orange Male
Orange female
BD
CW
CL
RW
FL



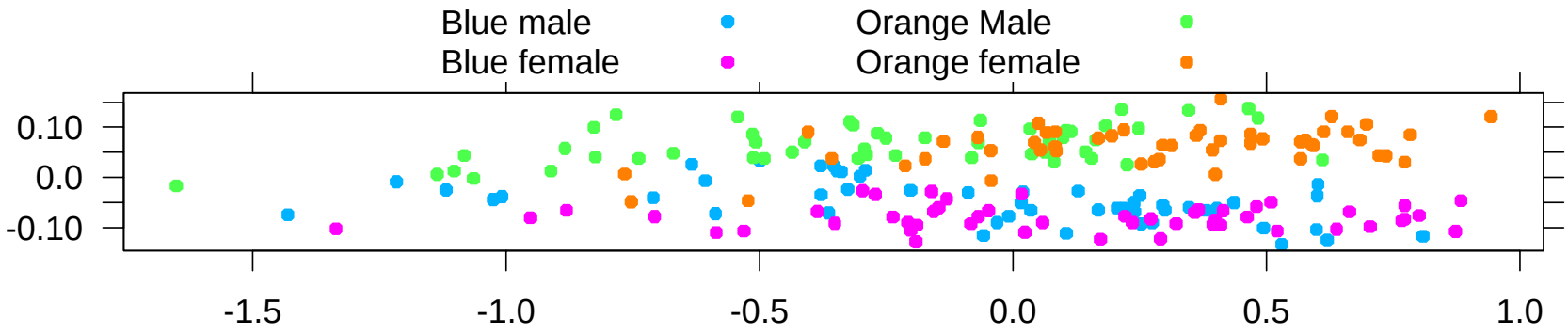
First three principal components on log scale.



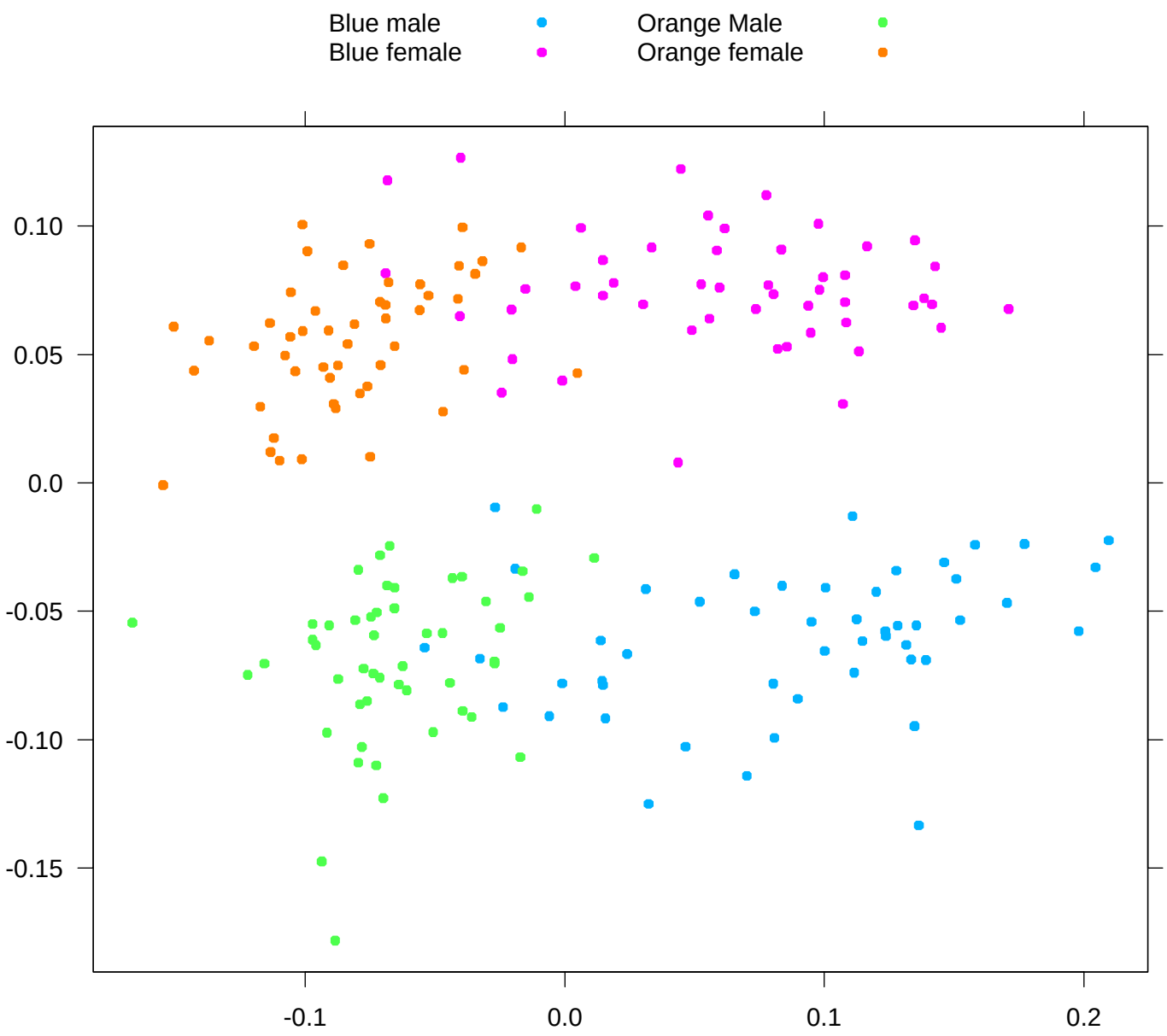
(a) (b) (c) (d)

Projections of the *Leptograpsus* crabs data found by projection pursuit. View (a) is a random projection. View (b) was found using the natural Hermite index, view (c) by the Friedman–Tukey index and view (d) by Friedman's (1987) index.

Multidimensional scaling



`isoMDS` plot of raw data.

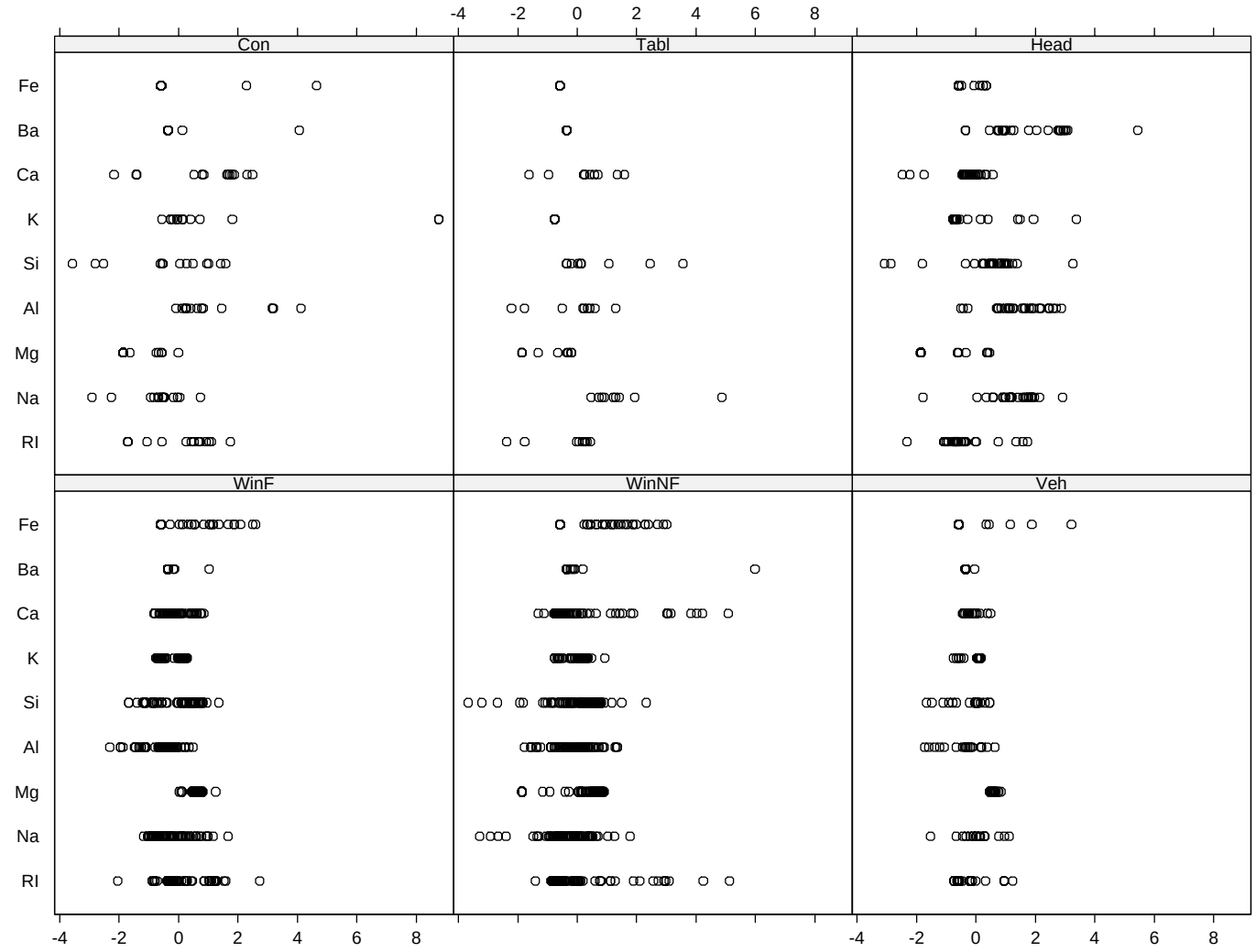


After re-scaling to (approximately) constant carapace area.

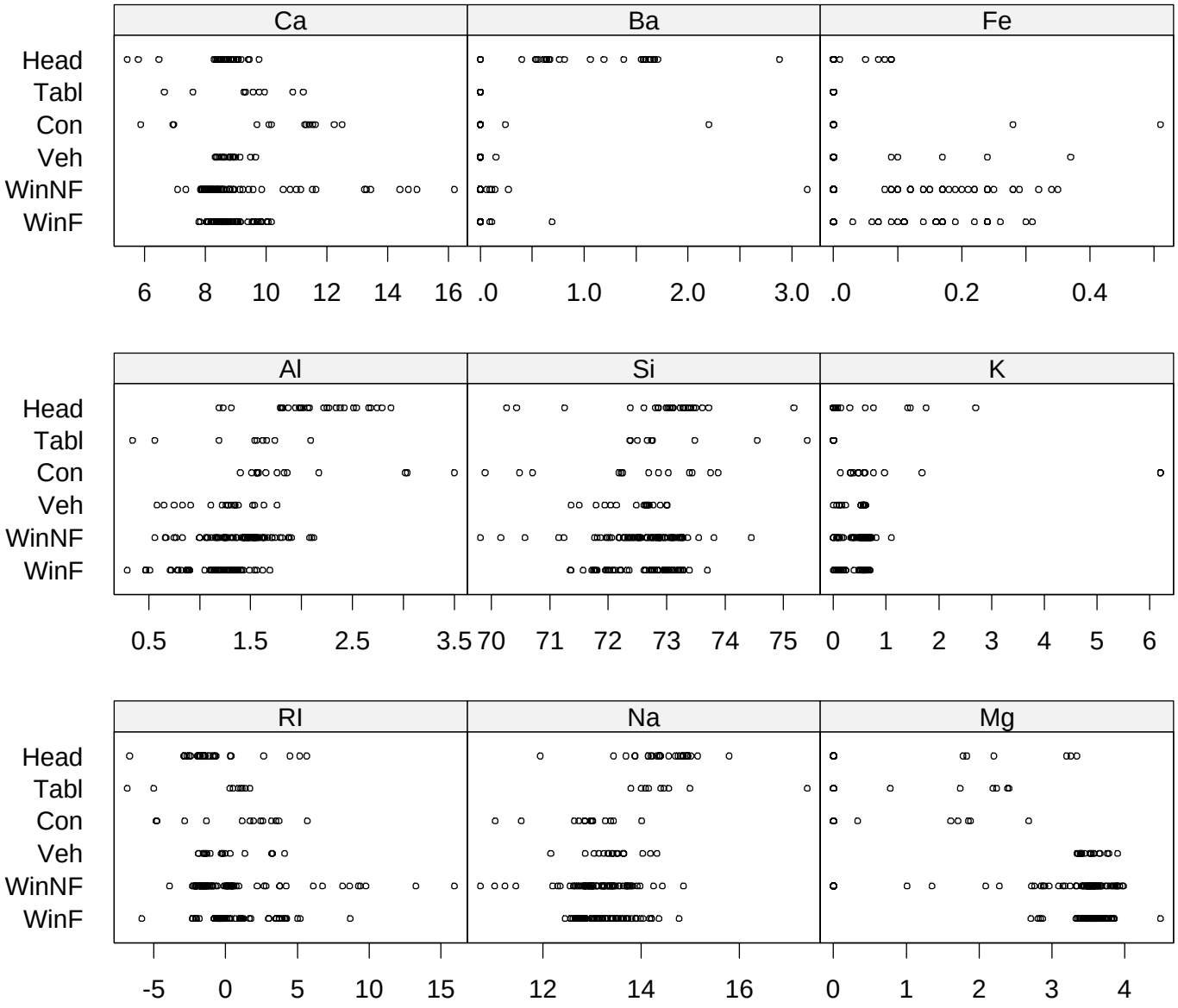
A Forensic Example

Data on 214 fragments of glass. Each has a measured refractive index and composition (weight percent of oxides of Na, Mg, Al, Si, K, Ca, Ba and Fe).

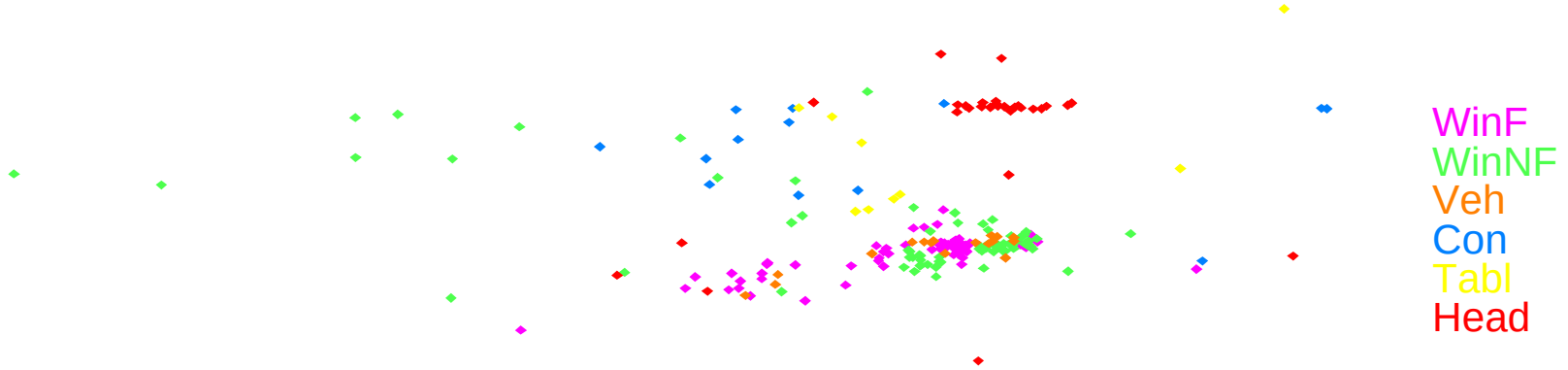
Grouped as window float glass (70), window non-float glass (76), vehicle window glass (17) and other (containers, tableware, headlamps) (22).



Strip plot by type of glass.



Strip plot by type of analyte.



Isotonic multidimensional scaling representation.

Regression
and
Beyond

Regression

Usually means least-squares fitting of a linear model. Assumptions:

- Only one sort of ‘error’, measurement error.
- Distribution of measurement errors is roughly normal.
- Distribution of measurement errors does not depend on the regressors.
- Regressors have been transformed suitably.
- Response has been transformed suitably.
- Linear relationship is adequate.
- All the regressors are relevant, and none have been over-looked.

Some related ideas are fitting functional relationships (errors in regressors), and the linear mixed models of next session.

Ideas here of diagnostics, non-least-squares fitting, and smooth transformations.

Scottish Hill Races

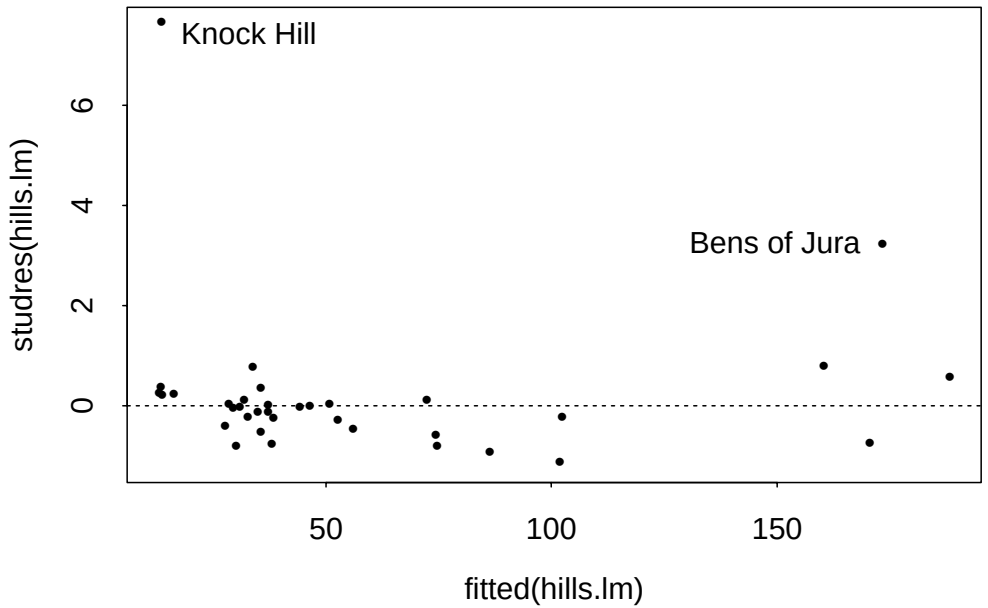
Data on record times in 35 Scottish hill races. Have overall race distance (miles), the total height climbed (feet) and the record time (minutes).

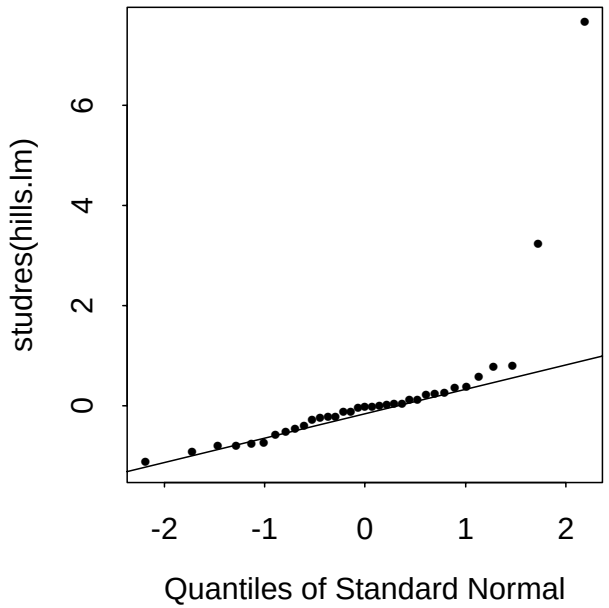
```
> hills.lm <- lm(time ~ dist + climb, hills)
> hills.lm
    ....
Coefficients:
 (Intercept)  dist    climb
      -8.992 6.218 0.011048

Degrees of freedom: 35 total; 32 residual
Residual standard error: 14.676
```

Do some standard diagnostics

'Studentized residuals' refit the model (in theory) dropping the current observation.

Look for 'high leverage'






```
> hills.hat <- lm.influence(hills.lm)$hat
> cbind(hills, lev=hills.hat)[hills.hat > 3/35, ]
              dist climb    time     lev
 Bens of Jura   16  7500 204.617 0.42043
  Lairig Ghru   28  2100 192.667 0.68982
    Ben Nevis   10  4400  85.583 0.12158
Two Breweries   18  5200 170.250 0.17158
 Moffat Chase   20  5000 159.833 0.19099
```

that is points that influence the fit unduly.

If we look at `Knock Hill` we see that the prediction is over an hour less than the reported record:

```
> cbind(hills,
        pred=predict(hills.lm))["Knock Hill", ]
           dist climb  time   pred
Knock Hill    3   350 78.65 13.529
```

and Atkinson (1988) suggests (guesses?) that the record is one hour out. We drop this observation to be safe.

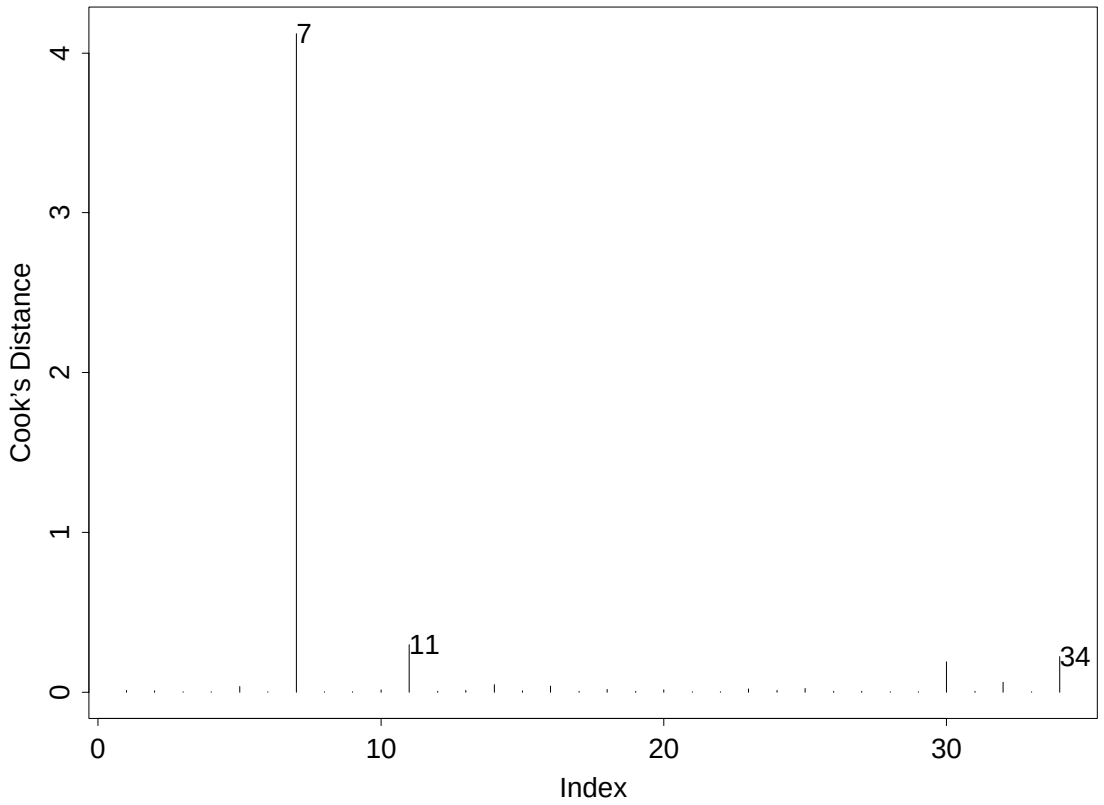
```
> hills1.lm <- lm(time ~ dist + climb, hills[-18, ])
> hills1.lm
    ....
Coefficients:
 (Intercept)   dist    climb
      -13.53 6.3646 0.011855

Degrees of freedom: 34 total; 31 residual
Residual standard error: 8.8035
```

Since `Knock Hill` did not have a high leverage, deleting it did not change the fitted model greatly.

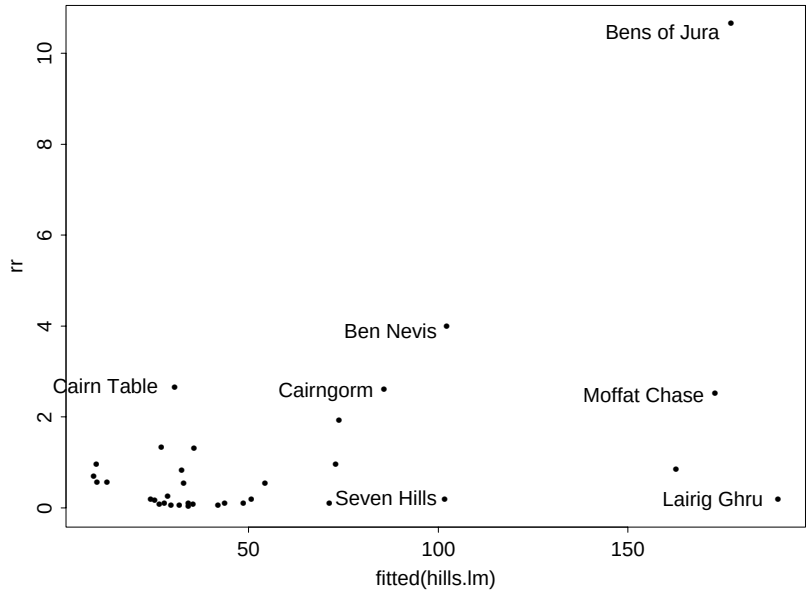
On the other hand, `Bens of Jura` had both a high leverage and a large residual and so does affect the fit:

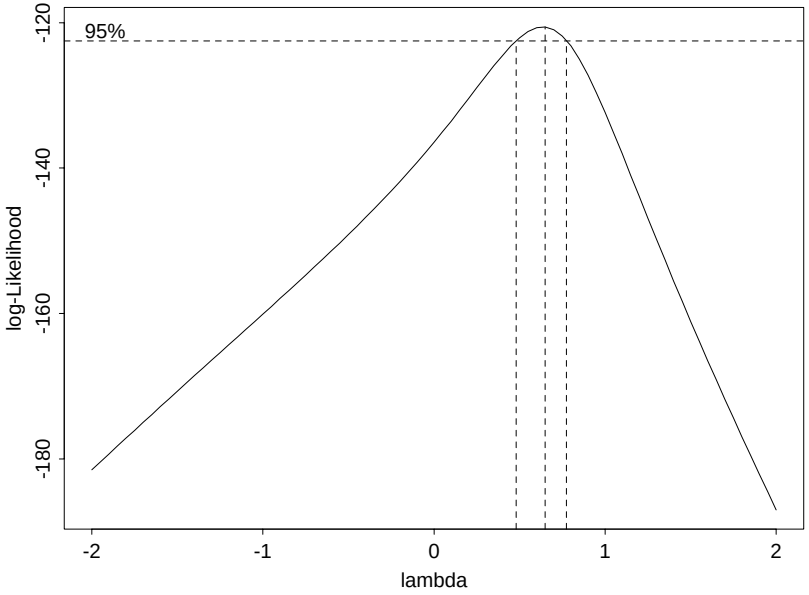
```
> lm(time ~ dist + climb, hills[-c(7,18), ])
    ....
Coefficients:
 (Intercept)   dist     climb
     -10.362 6.6921 0.0080468

Degrees of freedom: 33 total; 30 residual
Residual standard error: 6.0538
```

Cook’s statistic is often used to examine this graphically:



We can also look for evidence of heteroscedasticity by the Cook-Weisberg (1983) test:





(lots of evidence from `Bens of Jura`).

and for a transformation, via the Box–Cox family.

Is this adequate?

A number of unsatisfactory features:

- the predictions are negative for short races

```
> summary(hills1.lm)
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept) -13.530   2.649     -5.108   0.000
       dist   6.365   0.361     17.624   0.000
      climb   0.012   0.001      9.600   0.000
    ....
```

Notice the significantly negative intercept.

- we would not expect the predictions of times that range from 15 minutes to over 3 hours to be equally accurate. Try weighting

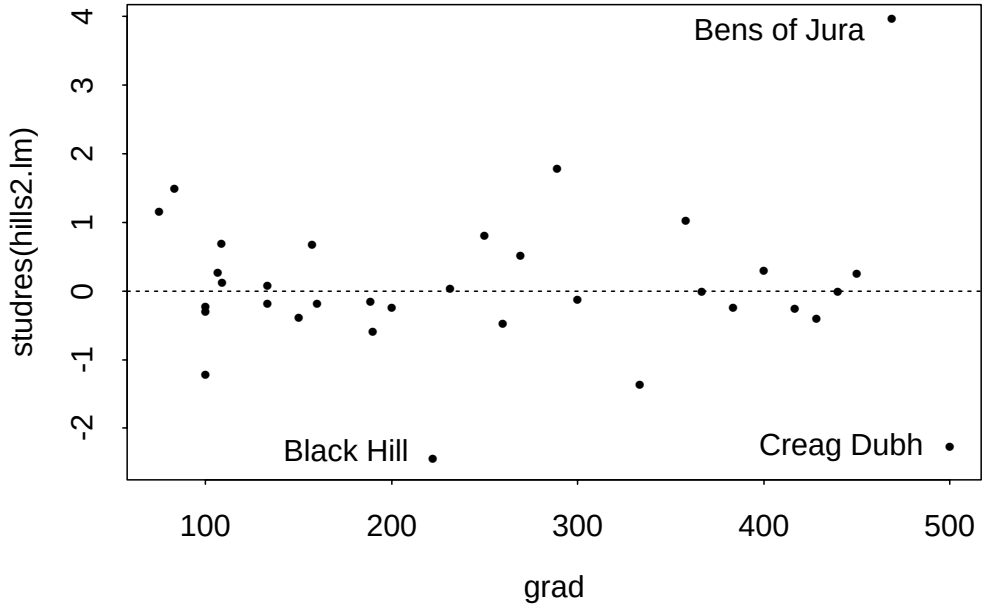
```
> summary(lm(time ~ dist + climb, hills[-18, ],
      weight=1/dist^2))
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  -5.809   2.034     -2.855   0.008
       dist   5.821   0.536     10.858   0.000
      climb   0.009   0.002      5.873   0.000

Residual standard error: 1.16 on 31 df
```

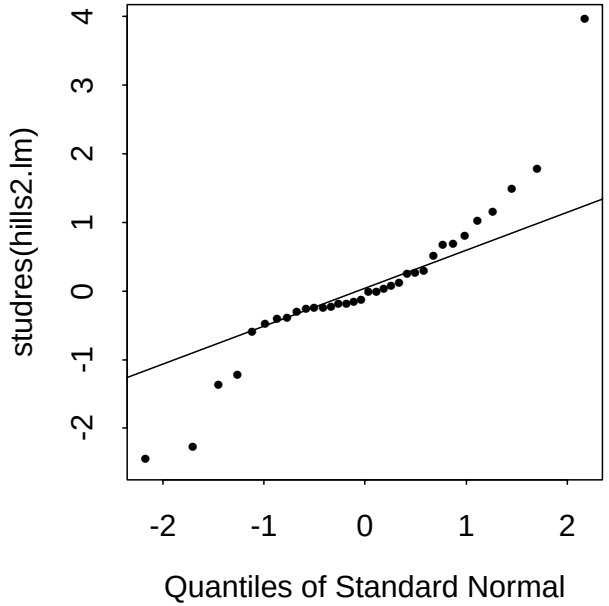
The intercept is still significantly non-zero.

If we are prepared to set it to zero on physical grounds, we can achieve the same effect by dividing the prediction equation by distance, and regressing inverse speed (time/distance) on gradient (climb/distance).

```
> lm(time ~ -1 + dist + climb, hills[-18, ],
     weight=1/dist^2)
Coefficients:
 dist     climb
  4.9 0.0084718

Degrees of freedom: 34 total; 32 residual
Residual standard error (on weighted scale): 1.2786
> hills$ispeed <- hills$time/hills$dist
> hills$grad <- hills$climb/hills$dist
> hills2.lm <- lm(ispeed ~ grad, hills[-18, ])
> hills2.lm
Coefficients:
 (Intercept)      grad
         4.9 0.0084718

Degrees of freedom: 34 total; 32 residual
Residual standard error: 1.2786
```

- there are some possible outliers.





and points of high(ish) leverage

```
> hills2.hat <- lm.influence(hills2.lm)$hat
> cbind(hills[-18,], lev=hills2.hat)
   [hills2.hat > 1.8*2/34, ]
               ispeed   grad     lev
Bens of Jura 12.7886 468.75 0.11354
  Creag Dubh  6.5542 500.00 0.13915
```

Robust Regression

Regression diagnostics can only tinker with leaving one or two points out.

Robust regression assumes a long-tailed distribution of errors.

Resistant regression tries to make a good fit to the majority of the data, discarding as much as it needs to.

Both are relatively computationally intensive.

Note there are conceptual issues here: can there be outlying x points, and if so, should they be discardable?

`lmsreg` and `ltsreg` are resistant methods, but unfortunately differ markedly between versions (and `ltsreg` is no longer resistant in S-PLUS 2000!) and are based on 15-year old methods and slow algorithms. `rreg` is also an old design implementing an old method. Will use `lqs` and `rlm` in library `MASS`.

Hill races again

```
> hills.lm
Coefficients:
 (Intercept)  dist    climb
      -8.992 6.218 0.011048
Residual standard error: 14.676

> hills1.lm # omitting Knock Hill
Coefficients:
 (Intercept)   dist    climb
      -13.53 6.3646 0.011855
Residual standard error: 8.8035

> rlm(time ~ dist + climb, hills)
Coefficients:
 (Intercept)   dist     climb
     -9.6067 6.5507 0.0082959
Scale estimate: 5.21
```

```
> summary(rlm(time ~ dist + climb, hills,
          weights=1/dist^2, method="MM"), cor=F)
Coefficients:
             Value Std. Error t value
(Intercept)  -1.804   1.665     -1.084
       dist   5.244   0.233     22.546
      climb   0.007   0.001      9.389
Residual standard error: 4.85 on 32 df
```

Method `"MM"` is in some sense the best of both worlds. Also in `lmRobMM`.

```
> lqs(time ~ dist + climb, data=hills,
      nsamp="exact")
Coefficients:
 (Intercept)     dist    climb
 -1.26        4.86     0.00851
Scale estimates 2.94 3.01
```

Notice that the intercept is no longer significant in the robust weighted fits.

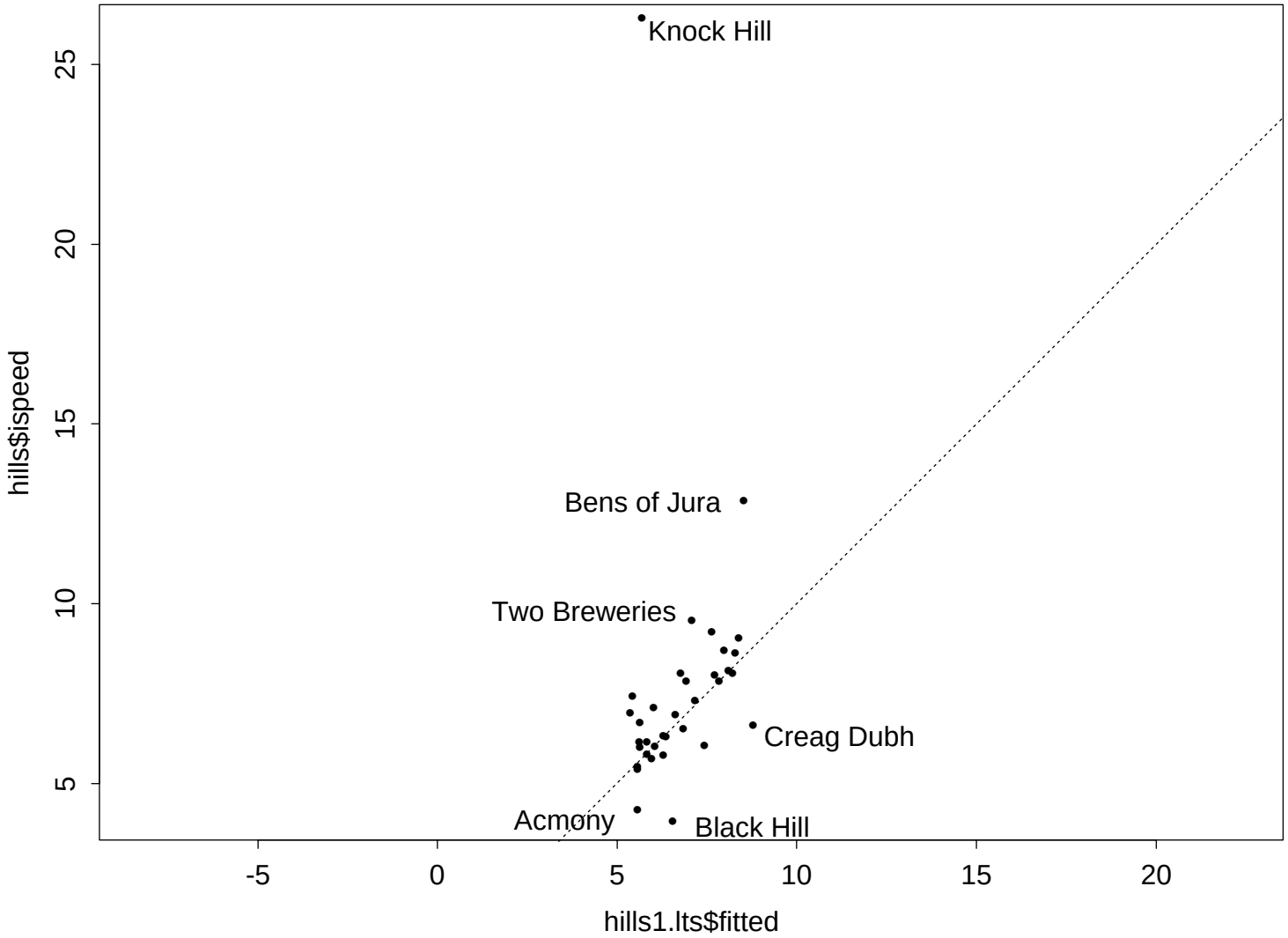
If we move to the model for inverse speed:

```
> summary(hills2.lm) # omitting Knock Hill
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  4.900  0.474     10.344   0.000
       grad  0.008  0.002      5.022   0.000

Residual standard error: 1.28 on 32 df

> summary(rlm(ispeed ~ grad, hills), cor=F)
Coefficients:
             Value Std. Error t value
(Intercept)  5.176  0.381     13.593
       grad  0.007  0.001      5.431

Residual standard error: 0.869 on 33 df
# method="MM" results are very similar.
```

```
> summary(lmRobMM(ispeed ~ grad, data=hills))
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  5.0754  0.4210    12.0563  0.0000
      grad  0.0077  0.0016     4.8817  0.0000

Residual scale estimate: 0.8189 on 33 df

> lqs(ispeed ~ grad, data=hills)
Coefficients:
 (Intercept)    grad
 4.75        0.00805
Scale estimates 0.608 0.643
```

Let us take a closer look at this last fit.

```
hills.lts <- lqs(ispeed ~ grad, data=hills,
                 nsamp="exact")
eqscplot(hills.lts$fitted, hills$ispeed)
abline(0, 1, lty=2)
identify(hills.lts$fitted, hills$ispeed,
         row.names(hills))
```

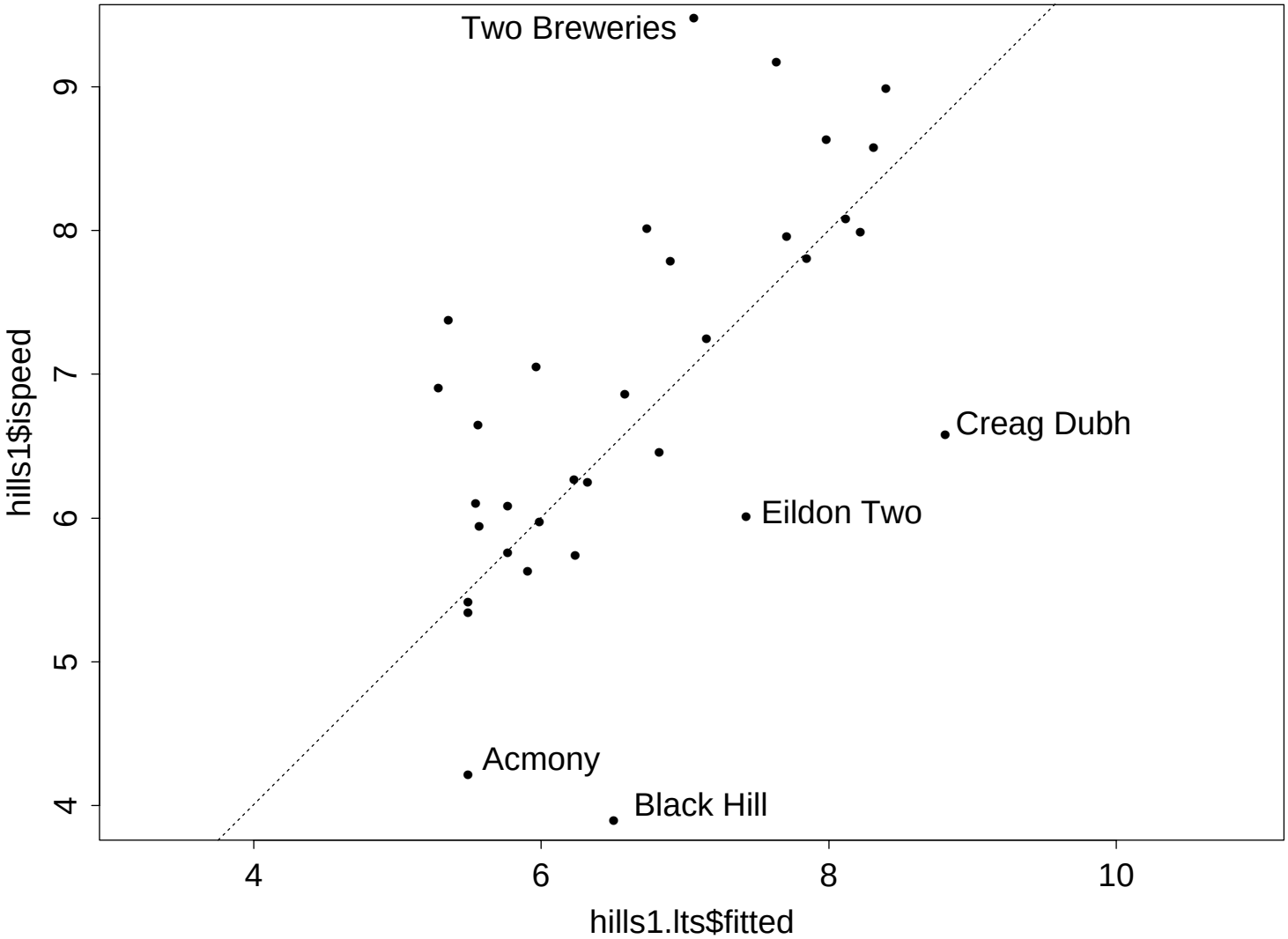

Knock Hill
Bens of Jura
Two Breweries
Creag Dubh
Acmony
Black Hill
hills$ispeed
hills1.lts$fitted
25
20
15
10
5
-5
0
5
10
15
20

`Knock Hill` is clearly way off and is rejected. We commented on the high influence of `Bens of Jura`.

Let us zoom in:

```
hills1 <- hills[-c(7,18),]
hills1.lts <- lqs(ispeed ~ grad, data=hills1,
                  nsamp="exact")
eqscplot(hills1.lts$fitted, hills1$ispeed)
abline(0, 1, lty=2)
identify(hills1.lts$fitted, hills1$ispeed,
         row.names(hills1))
```

We see that `Black Hill` is run at less than 4 minutes for each mile, for a 4.5 mile hill race gaining 1000 feet! That is none too plausible.


Two Breweries
Creag Dubh
Eildon Two
Acmony
Black Hill
hills1$ispeed
hills1.lts$fitted
4
5
6
7
8
9
4
6
8
10

Lessons

This example has been used as an example of regression diagnostics and in a textbook on robust methods. None spotted the rampant heteroscedasticity, nor the anomalous `Black Hill`. Even small and simple datasets are worth examining carefully.

Subject-matter knowledge is important.

Bootstrapping Regressions

Distribution theory for more advanced methods is approximate or unknown. Does the bootstrap help? Less than commonly supposed!

In frequentist inference we have to consider what might have happened but did not. Linear models can arise exactly or approximately in a number of ways. The most commonly considered form is

$$Y = X\beta + \boldsymbol{\epsilon}$$

in which only $\boldsymbol{\epsilon}$ is considered to be random. This supposes that in all (hypothetical) repetitions the same x points would have been chosen, but the responses would vary.

Another form of regression is sometimes referred to as the *random regressor* case in which the pairs (x_i, y_i) are thought of as a random sample from a population and we are interested in the regression function $f(x) = E\{Y \mid X = x\}$ which is assumed to be linear.

However, it is common to perform conditional inference in this case and condition on the observed xs, converting this to a fixed-design problem.

For example, in the hill races the inferences drawn depend on whether certain races, notably `Bens of Jura`, were included in the sample. As they were included, conclusions conditional on the set of races seems most pertinent.

These considerations are particularly relevant when we consider bootstrap resampling. The most obvious form of bootstrapping is to randomly sample pairs (x_i, y_i) with replacement (case-based resampling) which corresponds to randomly weighted regressions. However, this may not be appropriate in not mimicking the assumed random variation and in some examples of producing singular fits with high probability. The main alternative, *model-based resampling*, is to resample the residuals. After fitting the linear model we have

$$y_i = x_i\widehat{\beta} + e_i$$

and we create a new dataset by $y_i = x_i\widehat{\beta} + e_i^*$ where the (e_i^*) are resampled with replacement from the residuals (e_i).

There are a number of possible objections to this procedure. *First*, the residuals need not have mean zero if there is no intercept in the model, and it is usual to subtract their mean. *Second*, they do not have the correct variance or even the same variance. Thus we can adjust their variance by resampling the *modified residuals* $r_i = e_1/\sqrt{1-h_{ii}}$ which have variance σ^2.

No real need for bootstrapping with least-squares fitting of linear regression, but a useful test of the code. We use Canty's library `boot`.

```
library(boot)
fit <- lm(time ~ dist + climb, hills, weights=1/dist^2)
tmp <- data.frame(hills, res=resid(fit), fitted=fitted(fit))
tmp.fun <- function(data, i) {
  d <- data
  d$time <- d$fitted + d$res[i]
  coef(update(fit, data=d))
}
lm.boot <- boot(tmp, tmp.fun, R=499)
```

```
lm.boot
Bootstrap Statistics :
     original      bias    std. error
t1* 3.6271497  2.3858e+00  10.7492437
t2* 5.9395994  1.4032e-02   2.3299815
t3* 0.0038374 -5.1356e-05   0.0053167

boot.ci(lm.boot, index=2, type="norm")
Level      Normal
95%   (  1.359,  10.492 )
boot.ci(lm.boot, index=2, type=c("perc", "bca"))
Level     Percentile            BCa
95%   ( -1.031,   9.362 )   ( -2.971,   8.525 )
```

```
fit <- rlm(time ~ dist + climb, hills, weights=1/dist^2, method="MM")
tmp <- data.frame(hills, res=resid(fit), fitted=fitted(fit))
rlm.boot <- boot(tmp, tmp.fun, R=499)
rlm.boot
Bootstrap Statistics :
      original      bias    std. error
t1* -1.8038475  8.6192e-01   4.6642743
t2*  5.2441154 -3.2647e-02   1.0971856
t3*  0.0074503  5.4427e-05   0.0020185

boot.ci(rlm.boot, index=2, type="norm")
Level      Normal
95%   ( 3.126,  7.427 )
boot.ci(rlm.boot, index=2, type=c("perc", "bca"))
Level     Percentile            BCa
95%   ( 2.865,  7.175 )   ( 2.800,  7.123 )
```

Predicting Computer Performance

Ein-Dor & Feldmesser (1987) studied data on the performance on a benchmark of a mix of minicomputers and mainframes. The measure was normalized relative to an IBM 370/158–3.

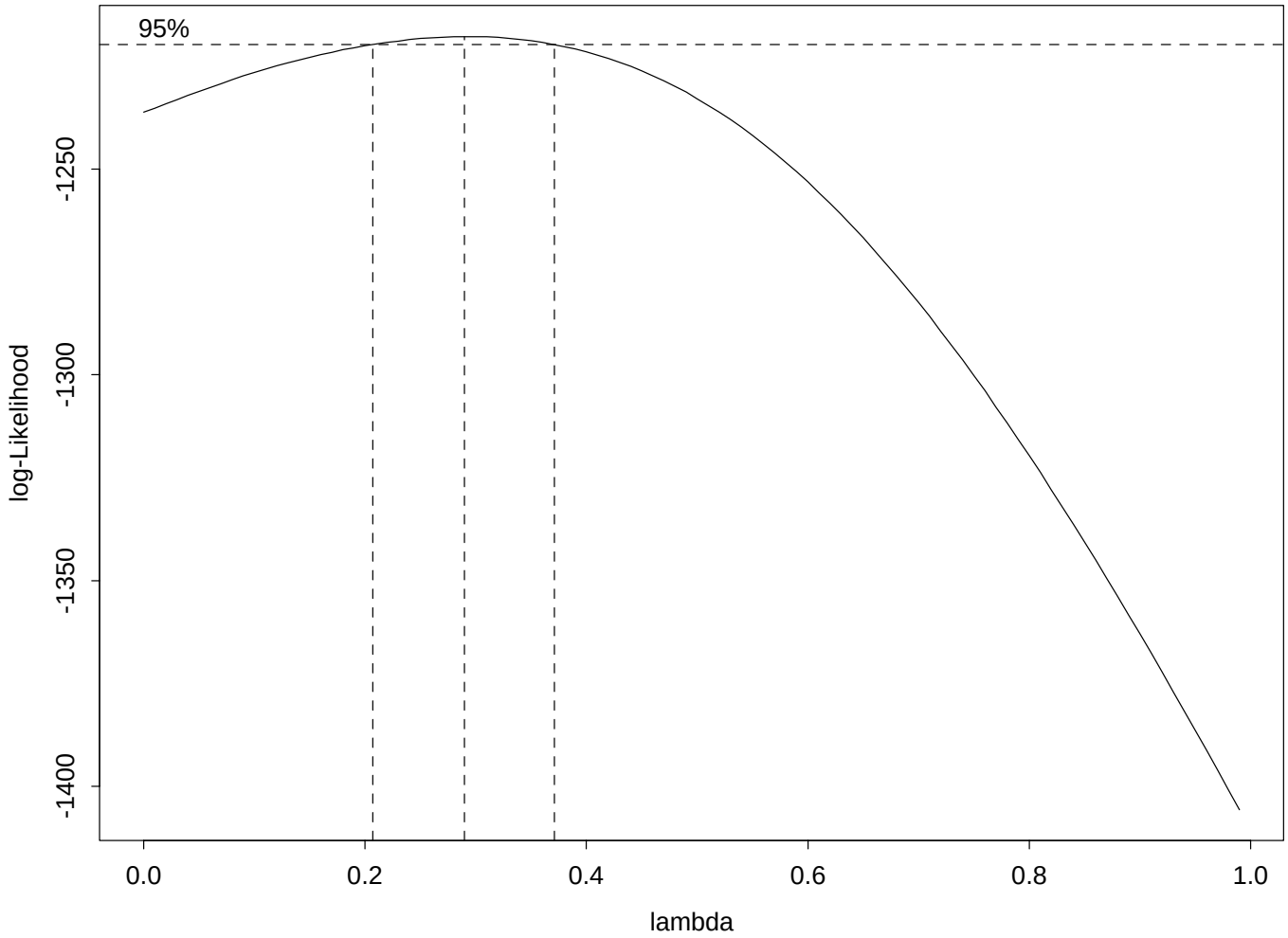
There were six machine characteristics, the cycle time (nanoseconds), the cache size (Kb), the main memory size (Kb) and number of channels. (For the latter two there are minimum and maximum possible values; what the actual machine tested had is unspecified.)

The original paper gave a linear regression for the square root of performance, but log scale looks more intuitive.

We can consider the Box–Cox family of transformations.

```
boxcox(perf ~ syct+mmin+mmax+cach+chmin+chmax,
       data=cpus, lambda=seq(0, 1, 0.1))
```

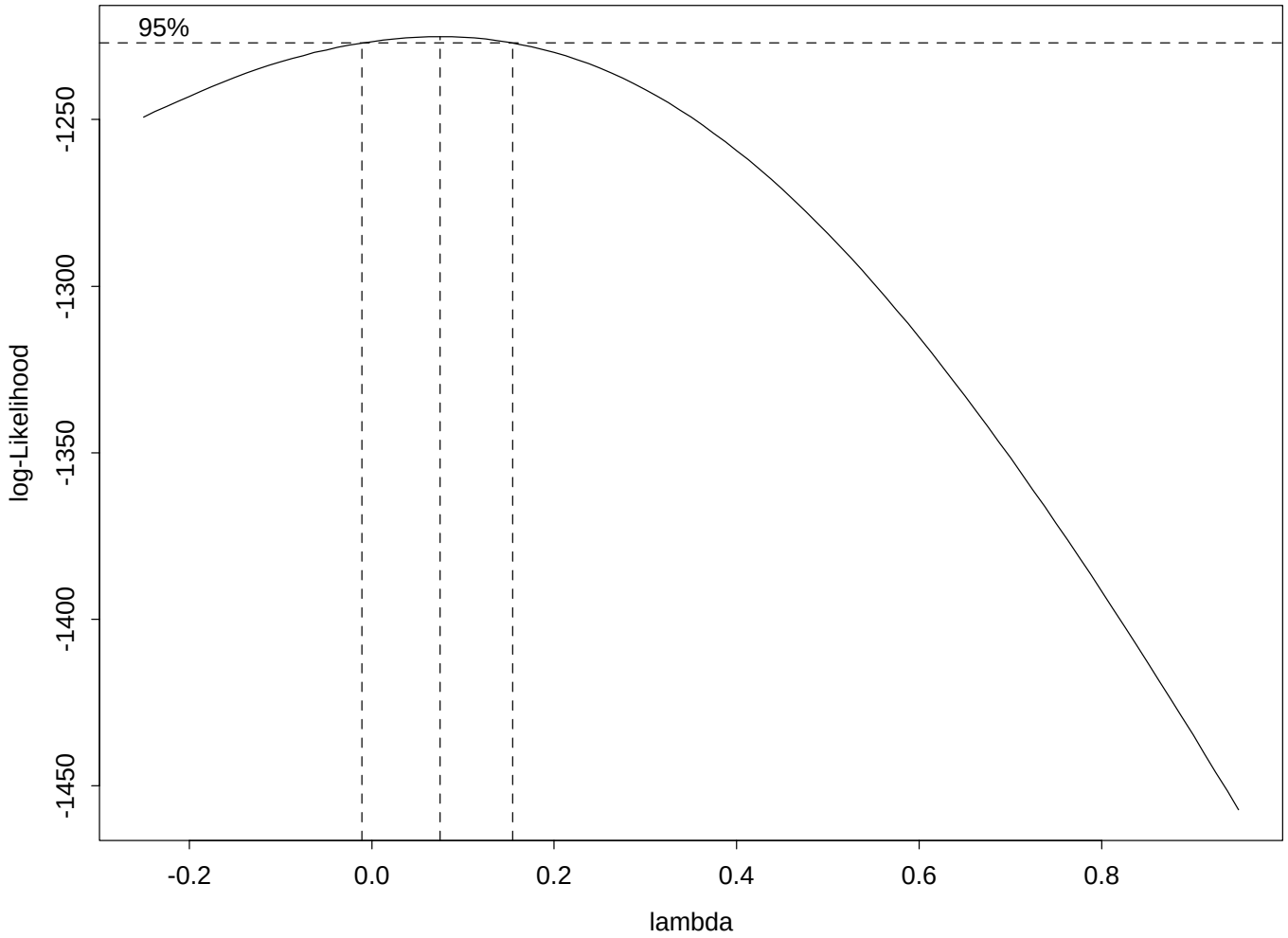
which tends to suggest a power of around 0.3 (and excludes both 0 and 0.5 from its 95% confidence interval).



However, this does not allow for the regressors to be transformed, and many of them would be most naturally expressed on log scale.

One way to allow the variables to be transformed is to discretize them and use them as factors.

```
cpus1 <- cpus
attach(cpus)
for(v in names(cpus)[2:6])
  cpus1[[v]] <-
    cut(cpus[[v]], unique(quantile(cpus[[v]])),
        include.lowest = T)
detach()
boxcox(perf ~ syct+mmin+mmax+cach+chmin+chmax,
       data = cpus1, lambda = seq(-0.25, 1, 0.1))
```



which does give a confidence interval including zero.

The purpose of this study is to predict computer performance. We randomly select 100 examples for fitting the models and test the performance on the remaining 109 examples.

First linear models on discretized variables with subset selection:

```
> set.seed(123)
> cpus2 <- cpus1[, 2:8]
# excludes names, authors' predictions
> cpus.samp <- sample(1:209, 100)
> cpus.lm <- lm(log10(perf) ~ ., data=cpus2[cpus.samp,2:8])
> test.cpus <- function(fit)
    sqrt(sum((log10(cpus2[-cpus.samp, "perf"]) -
         predict(fit, cpus2[-cpus.samp,]))^2)/109)
> test.cpus(cpus.lm)
[1] 0.20329
> cpus.lm2 <- stepAIC(cpus.lm, trace=F)
> cpus.lm2$anova
    Step Df Deviance Resid. Df Resid. Dev     AIC
1                           83     3.2300 -309.27
2 - syct  3 0.033248        86     3.2632 -314.25
test.cpus(cpus.lm2)
[1] 0.1955531
```

So selecting a smaller model does improve the prediction a little. Is the difference significant?

```
> res1 <- log10(cpus1[-cpus.samp, "perf"]) -
     predict(cpus.lm, cpus0[-cpus.samp,])
> res2 <- log10(cpus1[-cpus.samp, "perf"]) -
     predict(cpus.lm2, cpus2[-cpus.samp,])
> wilcox.test(res1^2, res2^2, paired=T, alternative="greater")
signed-rank normal statistic Z = 3.3135,
  p-value = 5e-04
```

Automated Transformations

For linear regression we have a dependent variable Y and a set of predictor variables $X_1, \ldots, X_p$, and model

$$Y = \alpha + \sum_{j=1}^{p} \beta_j X_j + \epsilon$$

So-called '*additive' models* replace the linear function $\beta_j X_j$ by a non-linear function to get

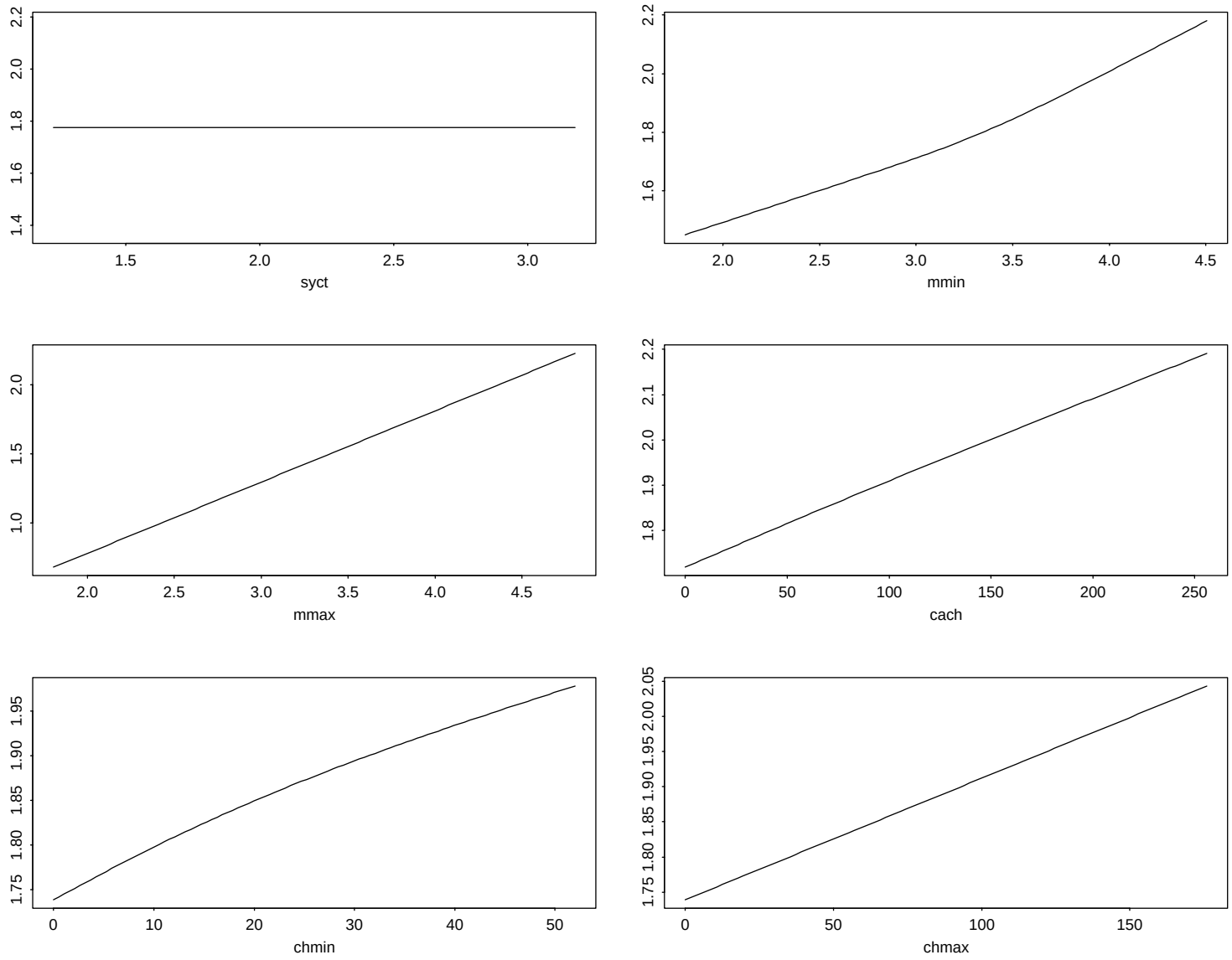
$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon \tag{1}$$

We could use `gam`, but that does not choose the degree of smoothness of the f_j.

Library `mda` of Hastie and Tibshirani provides functions `bruto` and `mars`. BRUTO fits additive models with smooth functions selected by smoothing splines and will choose between a smooth function, a linear term or omitting the variable altogether.

```
library(mda)
cpus0 <- cpus[, 2:8]
for(i in 1:3) cpus0[,i] <- log10(cpus0[,i])
cpus.bruto <- bruto(Xin, log10(cpus[cpus.samp,8]))

test2(cpus.bruto)
[1] 0.21336

cpus.bruto$type
[1] excluded smooth linear smooth smooth linear
cpus.bruto$df
 syct   mmin mmax   cach  chmin chmax
    0 1.5191    1 1.0578 1.1698     1
```

The result indicates that the non-linear terms have a very slight curvature, as might be expected from the equivalent degrees of freedom that are reported.



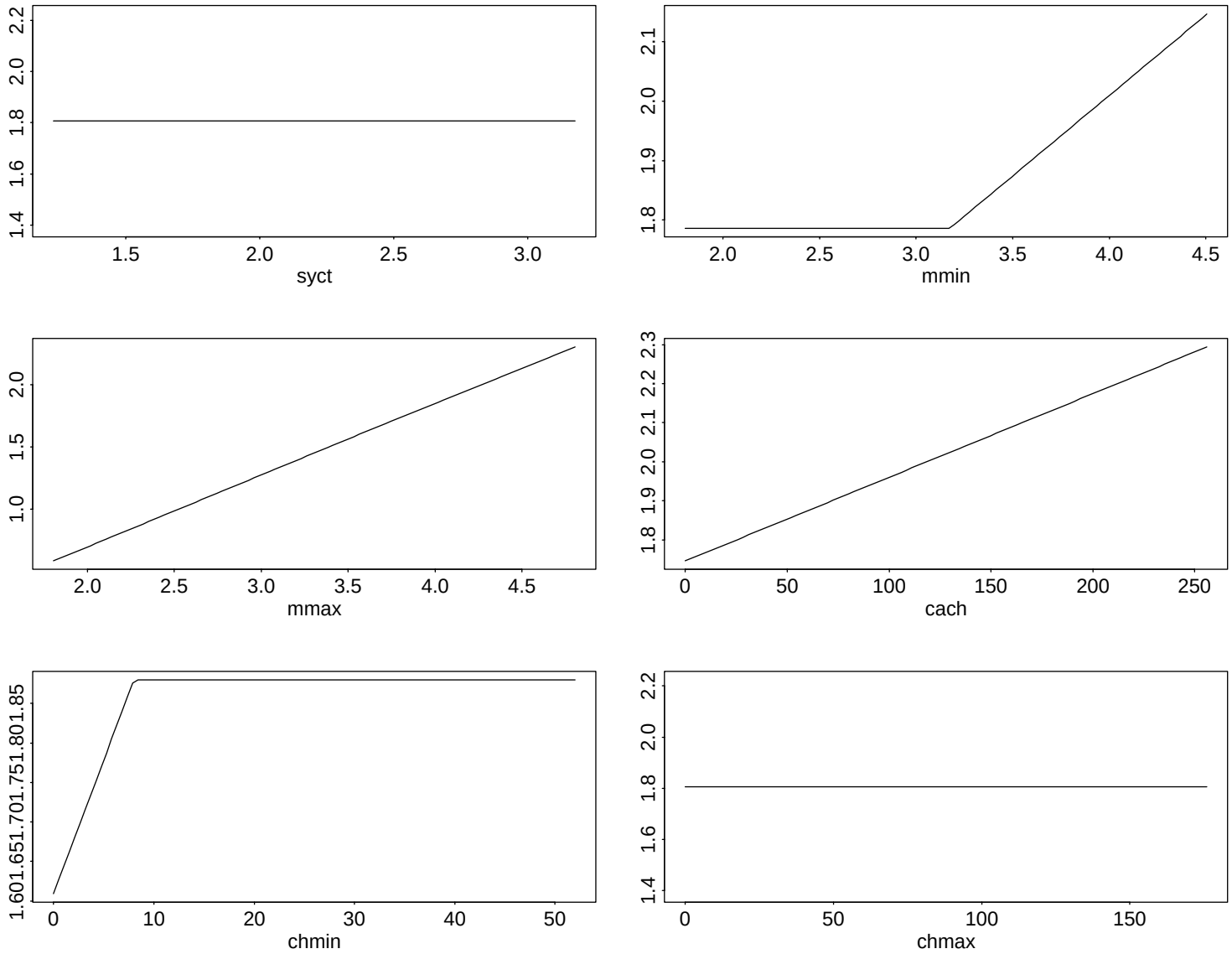
Plots of the additive functions used by `cpus.bruto`.

MARS

`mars` implements the MARS method of Friedman (1991). By default this is an additive method, fitting splines of order 1 (piecewise linear functions) to each variable; again the number of pieces is selected by the program so that variables can be entered linearly, non-linearly or not at all.

```
cpus.mars <- mars(Xin, log10(cpus[cpus.samp,8]))
> test2(cpus.mars)
[1] 0.21366

> cpus.mars2 <- mars(Xin, log10(cpus[cpus.samp,8]), degree=2)
> test2(cpus.mars2)
[1] 0.21495
> cpus.mars6 <- mars(Xin, log10(cpus[cpus.samp,8]), degree=6)
> test2(cpus.mars6)
[1] 0.20604
```

Allowing pairwise interaction terms (by `degree=2`) or allowing arbitrary interactions makes little difference to the effectiveness of the predictions.



Plots of the additive functions used by `cpus.mars`.

Response Transformation Models

If we want to predict Y, it may be better to transform Y as well, so we have

$$\theta(Y) = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon \tag{2}$$

for an invertible smooth function $\theta()$, for example the log function.

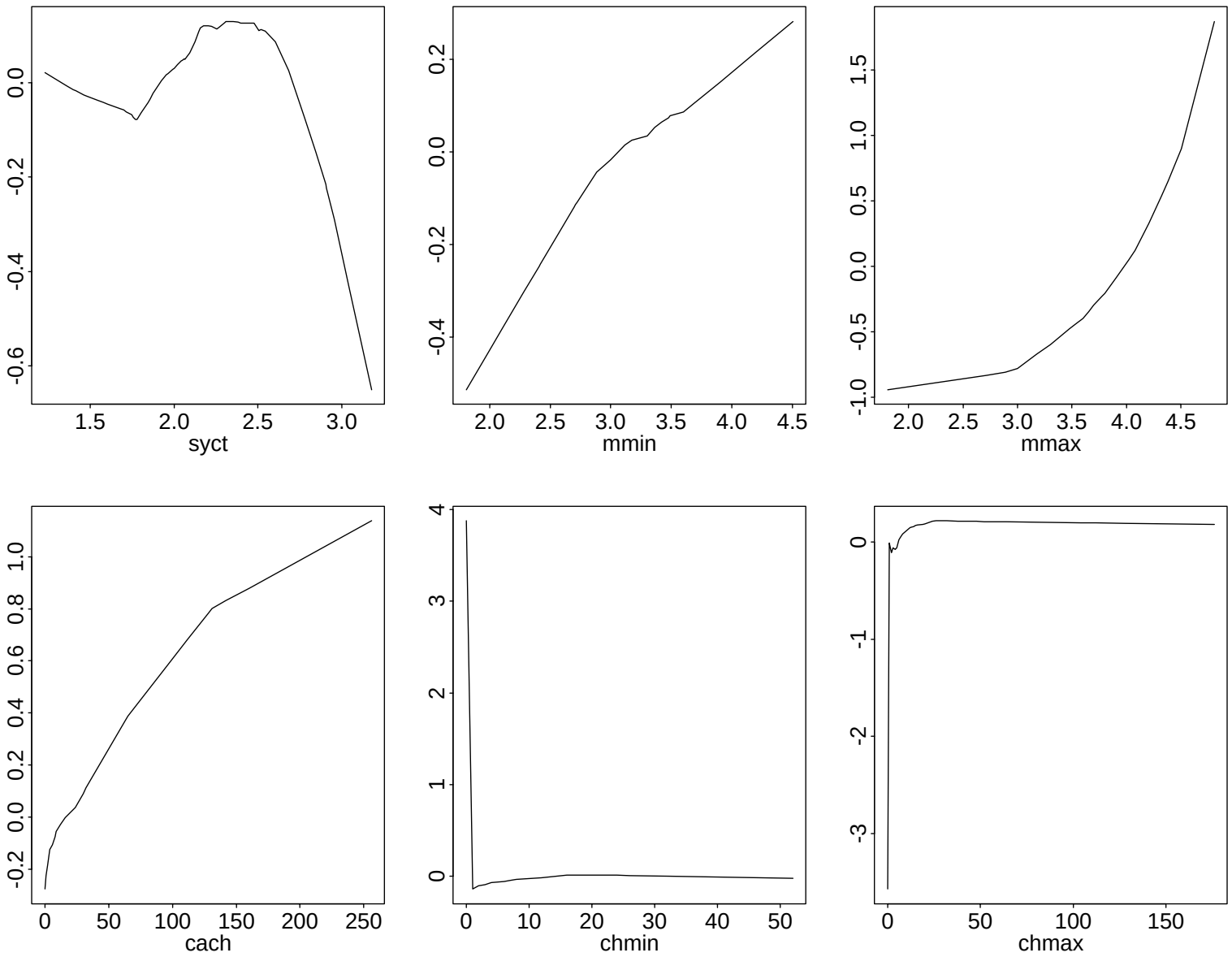
The ACE (alternating conditional expectation) algorithm of Breiman & Friedman (1985) chooses the functions θ and $f_1, \ldots, f_j$ to maximize the correlation between the predictor $\alpha + \sum_{j=1}^{p} f_j(X_j)$ and $\theta(Y)$.

Tibshirani's (1988) procedure AVAS (additivity and variance stabilising transformation) aims to achieve constant variance of the residuals for monotone θ.

CPUs data

We can consider the `cpus` data: we have already log-transformed some of the variables. AVAS accepts the log-scale for the response.

The strange shape of the transformations for `chmin` and `chmax` is probably due to local collinearity as there are five machines without any channels.



AVAS transformations of the regressors in the `cpus` dataset.

Random Effects
and
Mixed Models

Linear Mixed Models

$$Y = X\beta + Zb + \epsilon$$

where X and Z are specified design matrices, β is a vector of fixed effect coefficients, b and ϵ are random, mean zero, Gaussian if needed.

Usually think of b being constant over subjects, the ϵ as independent between subjects, possibly correlated within subjects. Let ω denote free parameters in the variance specification.

Likelihood

We observe n r.v.'s Y. Once the error structure is fully specified, and $\text{cov}(b, \epsilon) = 0$,

$$\begin{aligned} Y &\sim N\left(X\beta, V(\omega)\right) \\ V(\omega) &= \text{var}(\epsilon) + Z\text{var}(b)Z^T \end{aligned}$$

so minus twice the log-likelihood is

$$(Y - X\beta)^T V^{-1}(\omega)(Y - X\beta) + \log|V(\omega)|$$

Thus, given ω we find the MLE of β by generalized least squares.

One-way layout

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}, \qquad i = 1, ..., n_i$$

If we treat the $\tau_i \sim N\left(0, \sigma_b^2\right)$, we have a special case. The log-likelihood depends on β through the group means $m_i = \overline{y_{i\cdot}}$. Now

$$m_i \sim N\left(\mu, \sigma_b^2 + \sigma^2/n_i\right)$$

which suggests that we take a weighted mean of m_i with weights inversely proportional to var(m_i). This is MVUE and is in fact the MLE of μ (using the special structure of V).

What if the variances are unknown? For a balanced layout the estimator does not depend on them. In general it depends on σ_b^2/σ^2.

We can find the MLEs of σ_b^2 and σ^2, but even in the balanced case they are not the traditional ones: they have no adjustment for fitting means.

REML

Restricted / residual / reduced maximum likelihood: a method of estimation in LMEs.

Suppose that we can find some linear combinations AY whose distribution does not depend on β. In fact we can find up to $n - p$ linearly independent such. One choice is any $n-p$ of the least-squares residuals of the regression of Y on X.

In REML we treat AY as the data and use maximum-likelihood estimation of ω (the parameters in V).

The REML estimates do not depend on the choice of A, so this procedure is not as arbitrary as it sounds. Indeed, the REML estimates minimize

$$(Y - X\beta)^T V^{-1}(\omega)(Y - X\beta) + \log|V(\omega)| + \log|X^T V^{-1}(\omega)X|$$

Clearly the REML estimator of β is still GLS, plugging in the REML estimate of ω: *slightly* simpler to compute than MLEs.

Another perspective

The REML fit criterion is the marginal likelihood, integrating β out with a vague prior.

Relationship to classical ideas

In balanced designs REML gives the classical moment estimates of variance components (constrained to be non-negative).

Consider a paired comparison: REML will give the paired t–test analysis, ML will get the variance consistently low (by a factor of a half).

Drawbacks

No equivalents of likelihood-ratio tests (REMLs on models with different fixed effects are not comparable).

May be able to use Wald-like tests of extra parameters, but relevant asymptotic theory is hard to find.

Usual to quote GLS-based variances $X^T V^{-1}(\hat{\omega})X$ for $\hat{\beta}$ in both ML and REML procedures.

BLUPs

Best linear unbiased predictions. In an LME it is not clear what fitted values and hence residuals are. Our best prediction for subject i is *not* given by the mean relationship. We need to specify just what is common with an example we have already seen.

BLUPs replace the random effects b by their conditional means $\hat{b}$ given the data, and then make predictions using those values,

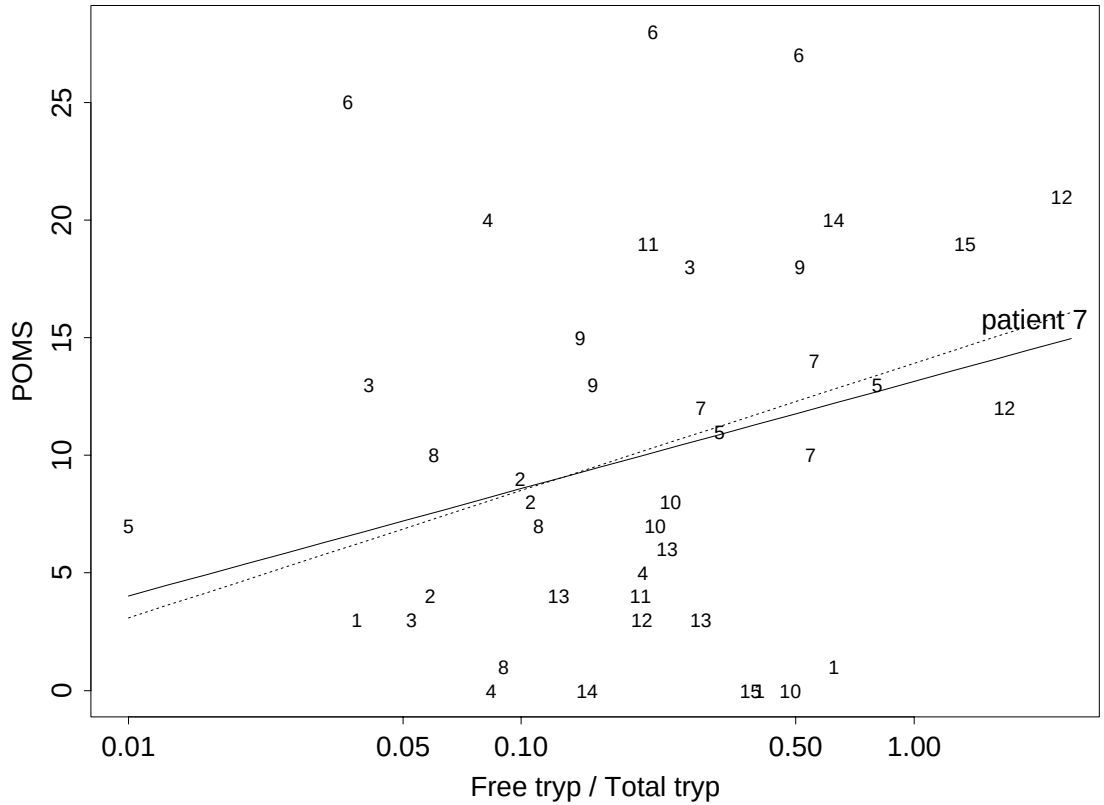
$$\hat{Y} = X\hat{\beta} + Z\hat{b}$$

Since everything is Gaussian, these are linear functions of the data, and as everything is linear, they are unbiased. They have minimum variance amongst such estimators.

Obviously if we have a new subject, $\hat{b} = 0$, and similarly in multilevel models. Therefore find several (in general) fitted values and several residuals.

Effects of Free Trytophan

James McGuire measured mood (POMS score) and abundance of free trytophan in the blood for 15 post-operative patients.



Classical model is

$$y_{ij} = \mu + \alpha_i + \beta x_{ij} + \epsilon_{ij}, \qquad \epsilon \sim N(0, \sigma^2)$$

a parallel line for each patient.

LME is

$$y_{ij} = \mu + \eta_i + \beta x_{ij} + \epsilon_{ij}, \qquad \eta \sim N(0, \sigma_\eta^2)$$

We could also consider a random effect for slope. This is hopeless in the classical case. The LME becomes

$$y_{ij} = \mu + \eta_i + (\beta + \zeta) x_{ij} + \epsilon_{ij}, \qquad \zeta \sim N(0, \sigma_\zeta^2)$$

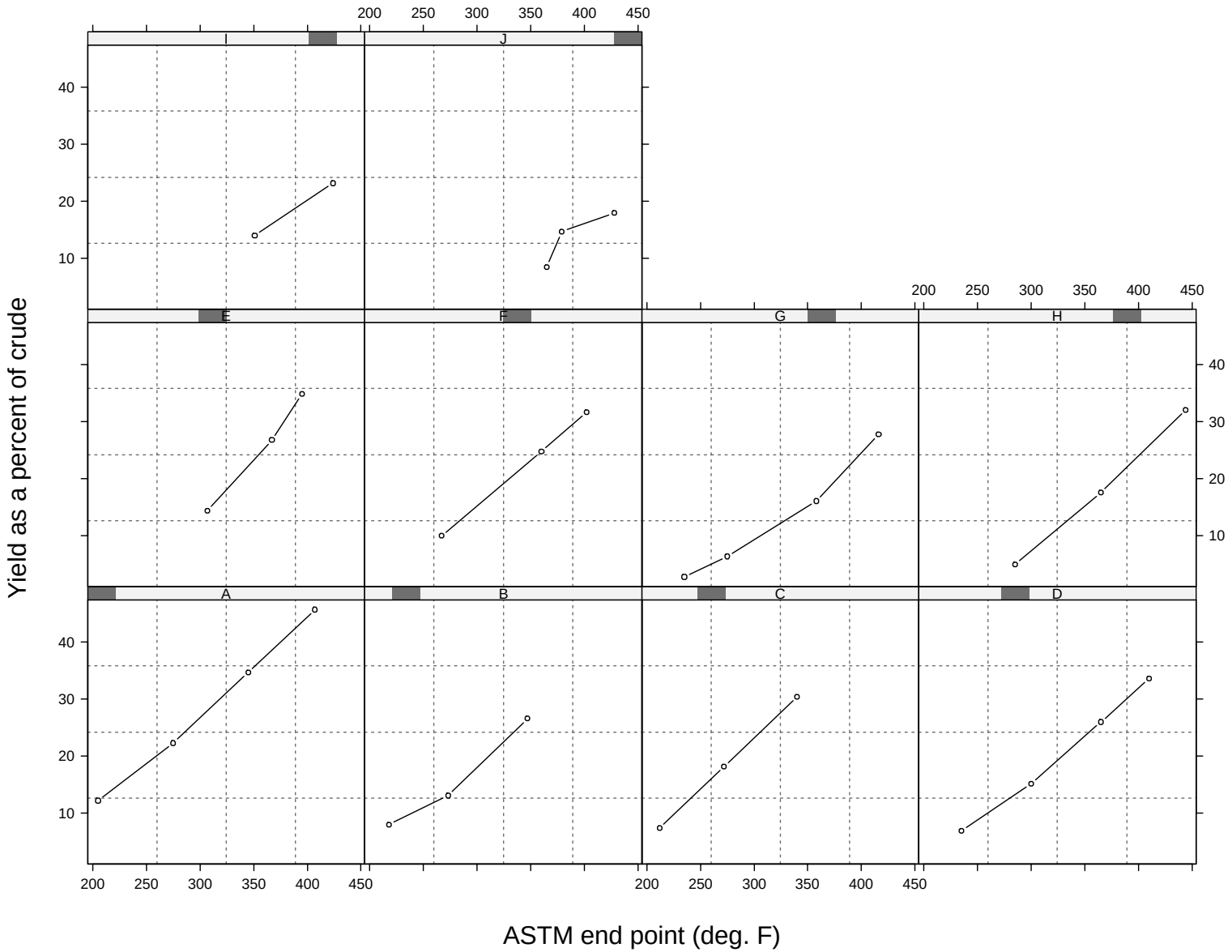
where either we allow η and ζ to be correlated or we centre x carefully. The estimate of β and its estimated s.e. are almost unchanged. The BLUPs for each patient are very different from the classical fits.

Linear Mixed-Effects Example

Gasoline data (Prater, 1956). Apparently 10 crude oil samples, 2–4 measurements on each.

Classical analysis

```
> pet1.lm <- lm(Y ~ No/EP - 1, Petrol)
> pet2.lm <- lm(Y ~ No - 1 + EP, Petrol)

> anova(pet2.lm, pet1.lm)
      Terms RDf    RSS  Df Sum of Sq F Value   Pr(F)
No - 1 + EP  21 74.132
  No/EP - 1  12 30.329   9    43.803  1.9257  0.1439

> pet3.lm <- lm(Y ~ SG + VP + V10 + EP, Petrol)

> anova(pet3.lm, pet2.lm)
             Terms RDf    RSS  Df Sum of Sq F Value    Pr(F)
SG + VP + V10 + EP  27 134.80
       No - 1 + EP  21  74.13   6    60.672  2.8645 0.033681
```



Yield versus ASTM end point within samples.

(Notice that SG, VP and V10 are constant within the levels of No so these two models are genuinely nested.) The result suggests that differences between intercepts are *not* adequately explained by such a regression.

Mixed model

A promising way of generalizing the model is to assume that the 10 crude oil samples form a random sample from a population where the intercepts after regression on the determining variables depend on the sample:

$$y_{ij} = \mu + e_i + \beta_1 \mathtt{SG}_i + \beta_2 \mathtt{VP}_i + \beta_3 \mathtt{V10}_i + \beta_4 \mathtt{EP}_{ij} + \epsilon_{ij}$$

where i denotes the sample and j the observation on that sample, and $e_i \sim N(0, \sigma_1^2)$ and $\epsilon_{ij} \sim N(0, \sigma^2)$, independently.

```
pet3.lme <- lme(Y ~ SG + VP + V10 + EP, random = ~ 1 | No, data = Petrol)
summary(pet3.lme)
Linear mixed-effects model fit by REML
 Data: Petrol
     AIC    BIC  logLik
  166.38 175.45 -76.191

Random effects:
 Formula: ~1 | No
        (Intercept) Residual
StdDev:      1.4447   1.8722


Fixed effects: Y ~ SG + VP + V10 + EP
              Value Std.Error DF t-value p-value
(Intercept)  19.707   0.56827 21  34.679  <.0001
         SG   0.219   0.14694  6   1.493  0.1860
         VP   0.546   0.52052  6   1.049  0.3347
        V10  -0.154   0.03996  6  -3.860  0.0084
         EP   0.157   0.00559 21  28.128  <.0001
    ....
```

using REML, and

```
> pet3.lme <- update(pet3.lme, method="ML")
> summary(pet3.lme)
Linear mixed-effects model fit by maximum likelihood
 Data: Petrol
     AIC    BIC  logLik
  149.38 159.64 -67.692

Random effects:
 Formula:  ~ 1 | No
        (Intercept) Residual
StdDev:     0.92889   1.8273
Fixed effects: Y ~ SG + VP + V10 + EP
              Value Std.Error DF t-value p-value
(Intercept)  19.694     0.478 21  41.188   0.000
         SG   0.221     0.123  6   1.802   0.122
         VP   0.549     0.441  6   1.246   0.259
        V10  -0.153     0.034  6  -4.469   0.004
         EP   0.156     0.006 21  26.620   0.000
    ....
```

by ML.

We can drop SG and VP.

```
> pet4.lme <- update(pet3.lme, fixed = Y ~ V10 + EP)
> anova(pet4.lme, pet3.lme)
         Model df    AIC    BIC  logLik    Test Lik.Ratio p-value
pet4.lme     1  5 149.61 156.94 -69.806
pet3.lme     2  7 149.38 159.64 -67.692 1 vs. 2    4.2285  0.1207
```

Finally we check if we need both random regression intercepts and slopes on EP, so we fit the model

$$y_{ij} = \mu + e_i + \beta_3 \mathtt{V10}_i + (\beta_4 + \eta_i)\mathtt{EP}_{ij} + \epsilon_{ij}$$

where (e_i, η_i) and ϵ_{ij} are independent, but e_i and η_i can be correlated.

```
> pet5.lme <- update(pet4.lme, random = ~ 1 + EP | No)
> anova(pet4.lme, pet5.lme)
         Model df    AIC    BIC  logLik    Test  Lik.Ratio p-value
pet4.lme     1  5 149.61 156.94 -69.806
pet5.lme     2  7 153.61 163.87 -69.805 1 vs. 2  0.0025194  0.9987
```

The simpler model is good enough.

Non-linear Mixed-Effects Models

$$Y_{ij} = f(x_{ij}; \beta, \eta_i) + \epsilon_{ij}$$

will be general enough for our discussion. What we usually assume is that

$$Y_{ij} = f(x_{ij}; \beta + \eta_i) + \epsilon_{ij}$$

where some components of η_i may always be zero. (Only α and θ have random effects in the next example.)

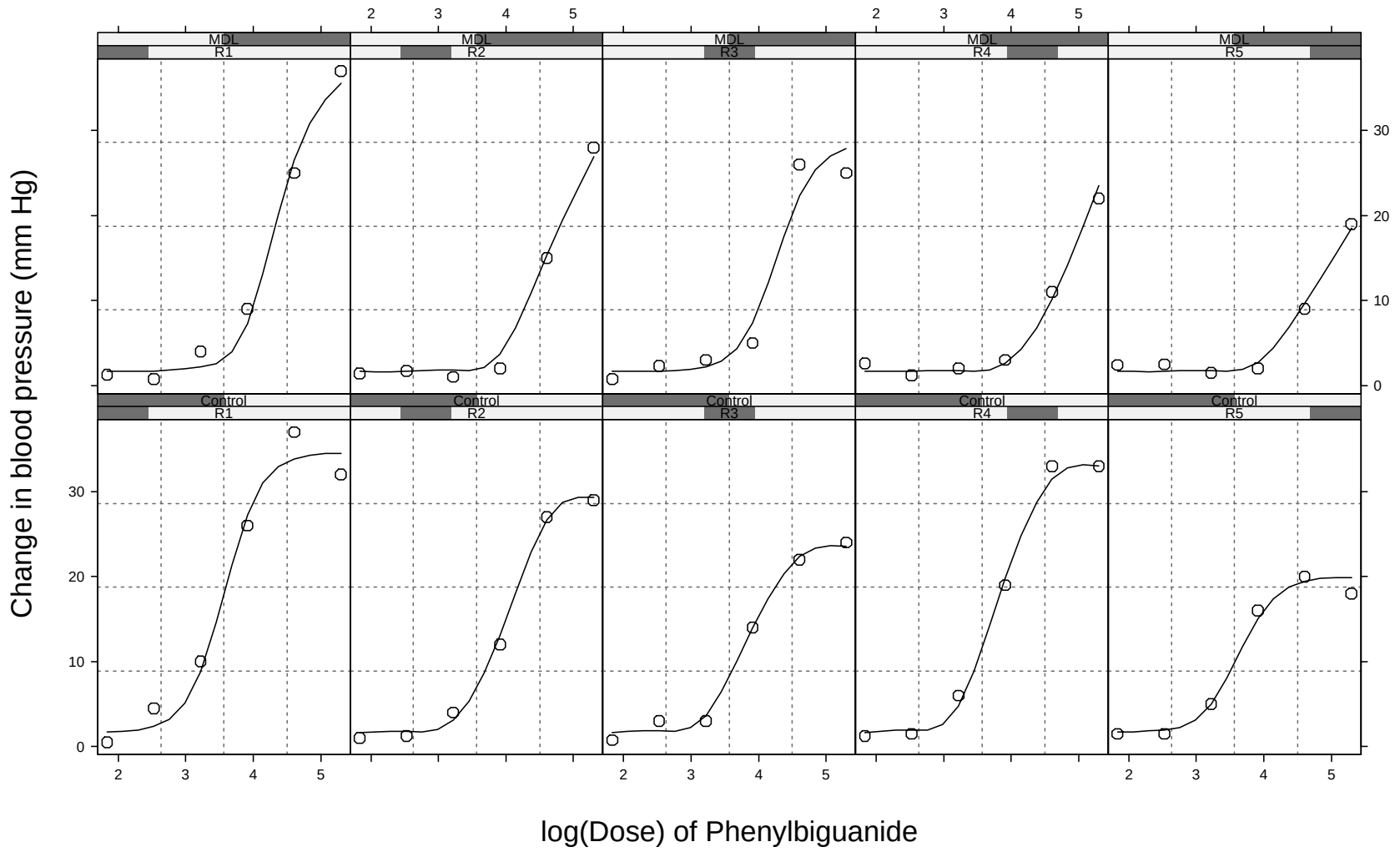
What is the likelihood? Only rarely can we integrate over (η_i). So 'MLEs' of NLMEs are based on approximations.

Blood Pressure in Rabbits

Five rabbits were studied on two occasions, after treatment with saline (control) and after treatment with the 5-HT_3 antagonist MDL 72222. After each treatment ascending doses of phenylbiguanide (PBG) were injected intravenously at 10 minute intervals and the responses of mean blood pressure measured. The goal was to test whether the cardiogenic chemoreflex elicited by PBG depends on the activation of 5-HT_3 receptors.

The response is the *change* in blood pressure relative to the start of the experiment.

$$f(x; \alpha, \beta, \lambda, \theta) = \alpha + \frac{\beta - \alpha}{1 + \exp[(x - \lambda)/\theta]}$$



Data points and BLUP curves for final fitted NLME model.

Fitting NLMEs

1. Fit a non-linear regression to each subject, and treat the parameter values as the data at subject level. If there is within-subject correlation, pool estimates of correlation parameters across subjects.

2. Use a Taylor-series expansion about the mean effects. This gives an LME which we can fit. Repeatedly expand about the fixed effects, that is write

$$Y_{ij} = f(x_{ij}; \hat{\beta}^0, 0) + X(\beta - \hat{\beta}^0) + Z\eta_i + \epsilon_{ij}$$

3. Use a Taylor-series expansion about estimates of (η_i):

$$Y_{ij} = f(x_{ij}; \hat{\beta}^0, \hat{\eta}^0) + X(\beta - \hat{\beta}^0) + Z(\eta_i - \hat{\eta}^0) + \epsilon_{ij}$$

   Lindstrom–Bates fit by simultaneously minimizing over (β, η_i); this effectively uses the BLUPs in the local linearization.

Inference in NLMEs

A problem! We have no likelihood to compare, and the `nlme` software appears to quote the likelihood of the final linearization.

We can use the estimated variance of the parameters and Wald–like tests.

Rabbits

Note that there are three strata of variation:

1. Animals
2. Occasions within animals
3. Measurements on the animal/occasion combination.

and the effect of interest, the treatment, varies in the second stratum.

We start by fitting separate models for each treatment:

```
Control:
  Log-likelihood: -66.502
  Fixed: list(A ~ 1, B ~ 1, ld50 ~ 1, th ~ 1)
      A      B   ld50      th
 28.332 1.5134 3.7744 0.28957

Random effects:
 Formula: list(A ~ 1, ld50 ~ 1)
 Structure: General positive-definite
          StdDev  Corr
       A 5.76889 A
    ld50 0.17953 0.112
Residual 1.36735
```

```
Treatment:
  Log-likelihood: -65.422
  Fixed: list(A ~ 1, B ~ 1, ld50 ~ 1, th ~ 1)
      A      B   ld50      th
 27.521 1.7839 4.5257 0.24236

Random effects:
 Formula: list(A ~ 1, ld50 ~ 1)
 Structure: General positive-definite
         StdDev   Corr
       A 5.36549 A
    ld50 0.18999 -0.594
Residual 1.44172
```

Now a combined model

```
R.nlme1 <-
  nlme(BPchange ~ Fpl(Dose, A, B, ld50, th),
    fixed = list(A ~ Treatment, B ~ Treatment, ld50 ~ Treatment, th ~ Treatment),
    random =  A + ld50 ~ 1 | Animal/Run, data = Rabbit, ...)

Random effects:
 Formula: list(A ~ 1, ld50 ~ 1)
 Level: Animal
 Structure: General positive-definite

                StdDev   Corr
   A.(Intercept) 4.6063 A.(Int
ld50.(Intercept) 0.0626 -0.166

 Formula: list(A ~ 1, ld50 ~ 1)
 Level: Run %in% Animal
 Structure: General positive-definite
                StdDev   Corr
   A.(Intercept) 3.2489 A.(Int
ld50.(Intercept) 0.1707 -0.348
        Residual 1.4113
```

```
Fixed effects:
                   Value Std.Error t-value p-value
   A.(Intercept)  28.326    2.7802  10.188  <.0001
     A.Treatment  -0.727    2.5184  -0.288  0.7744
   B.(Intercept)   1.525    0.5155   2.958  0.0050
     B.Treatment   0.261    0.6460   0.405  0.6877
ld50.(Intercept)   3.778    0.0955  39.579  <.0001
  ld50.Treatment   0.747    0.1286   5.809  <.0001
  th.(Intercept)   0.290    0.0323   8.957  <.0001
    th.Treatment  -0.047    0.0459  -1.020  0.3135
```

This suggests that the only difference by treatment is to shift the mean curve along (λ varies by treatment). If we fit that we find

```
                   Value Std.Error t-value p-value
               A 28.170    2.4909  11.309  <.0001
               B  1.667    0.3069   5.433  <.0001
ld50.(Intercept)  3.779    0.0921  41.036  <.0001
  ld50.Treatment  0.759    0.1217   6.233  <.0001
              th  0.271    0.0226  11.964  <.0001
```

Generalized Linear Mixed Models

Suppose we have a binomial or Poisson response. We can apply the same ideas, with linear predictor

$$\eta = X\beta + Zb$$

and distribution of Y_i depending on η_i through the link function.

Note that unless we have a Gaussian GLM with identity link, the marginal distribution of Y_i is not binomial, Poisson etc; the (Y_i) are always dependent (and usually positively correlated in clusters).

This is known as a *subject-specific* model. The alternative is a *marginal* or *population-averaged* model where the marginal distribution of the Y_i is binomial, Poisson, etc, but they are correlated in clusters.

Logistic GLMM

Simplest case, a random-intercept model:

$$Y_{ij} \sim \text{bin}(n_{ij}, p_{ij}), \quad \text{logit}\, p_{ij} = b_i + (X\beta)_{ij}$$

Here i labels the cluster.

Methods:

- Conditional analysis, conditional on $\sum_j y_{ij}$, which eliminates the random intercept.
- Approximate MLEs based on Laplace expansion.
- Approximate MLEs based on numerical integration (and need to estimate the variance of b_i).
- Bayesian analysis by Gibbs sampler.

Marginal Models

Suppose we have several observations Y_{ij} on each cluster i. We allow the mean μ_{ij} of Y_{ij} to depend on η_{ij} for a linear predictor $\eta = X\beta$, the variance of Y_{ij} to depend on its mean (and possibly a dispersion parameter ϕ). Observations on different clusters are independent, but $(Y_{i\cdot})$ are dependent, with a correlation matrix depending on parameters ω.

Apart from the dependence, this is how we model a GLM.

Identity link

Suppose $\mu_{ij} = \eta_{ij}$, and we fit β by GLS with weight matrix W,

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

Then asymptotically $\hat{\beta}_W$ is unbiased and normal with variance matrix

$$\Sigma_W = \left[(X^T W X)^{-1} X^T W\right] \text{var}(Y) \left[W X (X^T W X)^{-1}\right]$$

- We may be able to estimate var(Y) some other way (REML from a saturated model?)
- All we lose by not having the correct weights W is efficiency.

General link

Still use GLS, ignore the dependence of var$(Y_{i\cdot})$ on β:

$$\sum_{\text{clusters } i} \frac{\partial \mu_{ij}}{\partial \beta} \text{var}(Y_{i\cdot})^{-1} \left[Y_{i\cdot} - \mu_{i\cdot}\right] = 0$$

These are the GLM score equations, except for the correlations, which need to estimate simultaneously.

This approach (including equations for α) is known as GEE, *Generalized Estimating Equations*. It has asymptotic theory that shows consistency, asymptotic normality with estimable variance matrix.

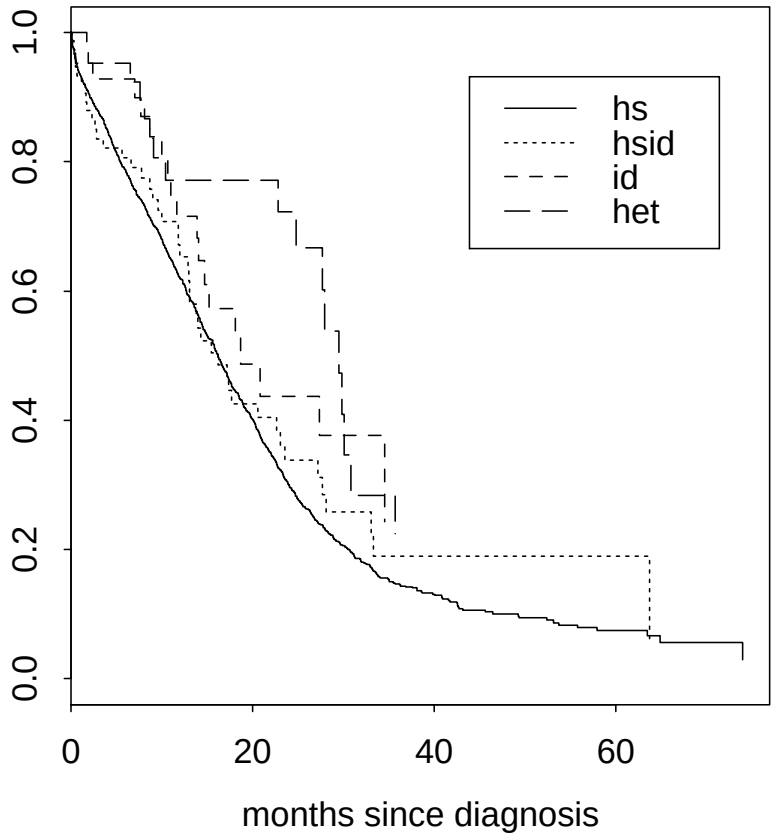
Survival Analysis

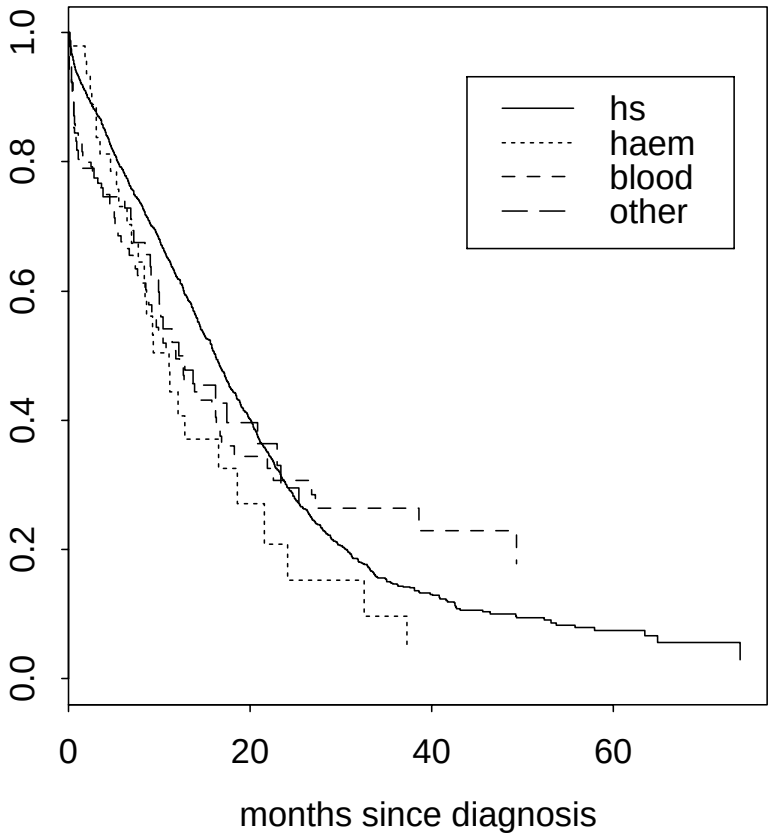
Smooth Survival Curves

Classical approaches are parametric (e.g. Weibull) or rough (piecewise constant) as in Kaplan-Meier.

There are analogues of density estimation for survival data in which we seek smooth estimates of the survival function S, the density f or (especially) the hazard function h.

Our main example is on 2 843 AIDS patients in Australia, of whom 1 770 had died by the end of the study. The main covariates were gender, age, transmission category and state.





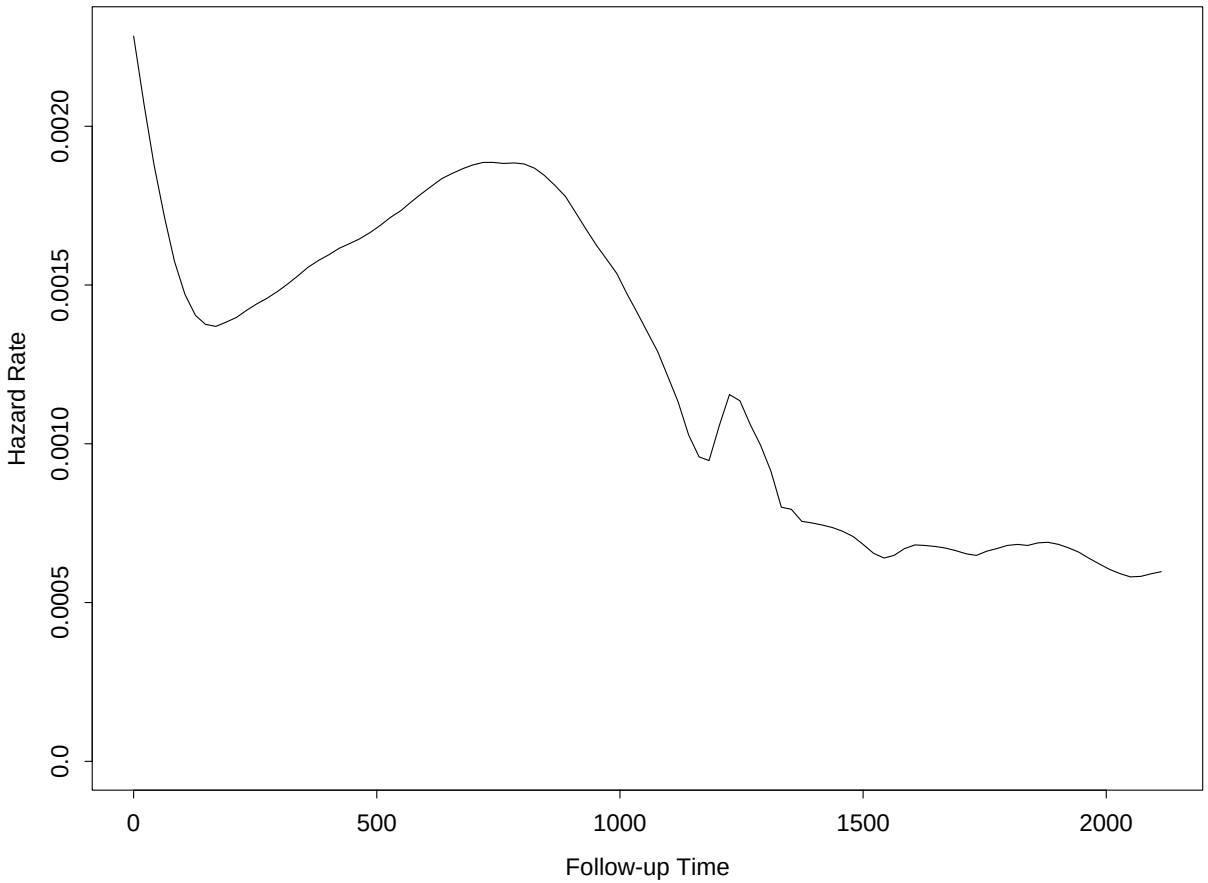
Survival of AIDS patients in Australia by transmission category.

Kernel-based approaches

Approach of Mueller & Wang (1994), in library `muhaz`.

```
attach(Aids2)
plot(muhaz(death-diag+0.9, status=="D"), n.est.grid=250)
```

This is slow (takes 30 seconds) and we had to refine the output grid to produce a fairly smooth result.



Hazard function fitted to the `Aids` dataset by `muhaz`.

Not very plausible, but some people prefer this method!

Likelihood approaches

The full log-likelihood is

$$\sum_{t_i:\delta_i=1} \log h(t_i) - \sum_i \int_0^{t_i} h(u)\,\mathrm{d}u$$

HEFT (Kooperberg *et al.*, 1995) uses cubic spline model is used for the log hazard, but with two additional terms $\theta_1 \log t/(t+c)$ and $\theta_2 \log(t+c)$ where c is the upper quartile for the uncensored data. Then the space of fitted hazards includes the functions

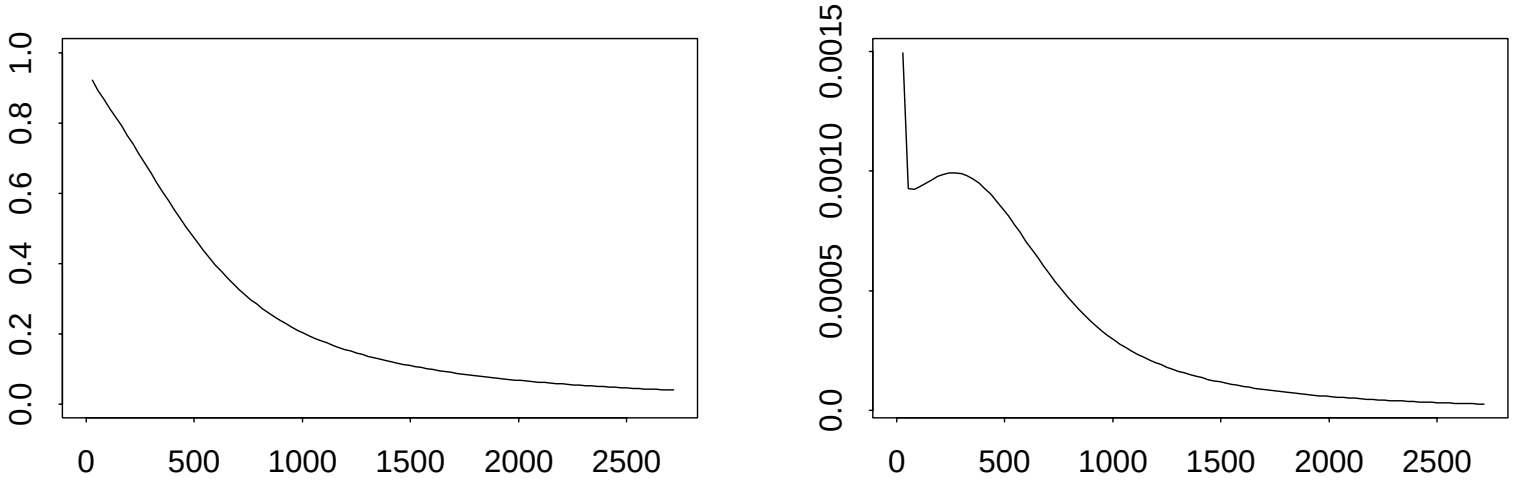
$$h(t) = e^{\theta_0} t^{\theta_1} (t+c)^{\theta_2 - \theta_1}$$

which include the Weibull family and the Pareto density

$$f(t) = \frac{bc^b}{(t+c)^{b+1}}$$

for given c. Thus there is some hope that the tail behaviour can be captured within this parametric family.

```
library(heft); attach(Aids2)
aids.heft <- heft.fit(death-diag+0.9, status=="D")
heft.summary(aids.heft)
par(mfrow=c(2,2))
heft.plot(aids.heft, what="s",  ylim=c(0,1)); heft.plot(aids.heft)
```



Survivor curve and hazard fitted to `Aids` by `heft.fit`.

This is rather slow (20 seconds). The spike at 0 of the hazard reflects the small number of patients diagnosed at death. Note that this is the *marginal* hazard and its shape need not be at all similar to the hazard fitted in a (parametric or Cox) proportional hazards model.

Local likelihood approach

We can localize the likelihood by adding weighting terms and uses locally polynomial (e.g. quadratic) form of the hazard.

In theory this can be done by Loader's `locfit`, but that fails after several minutes on this example.

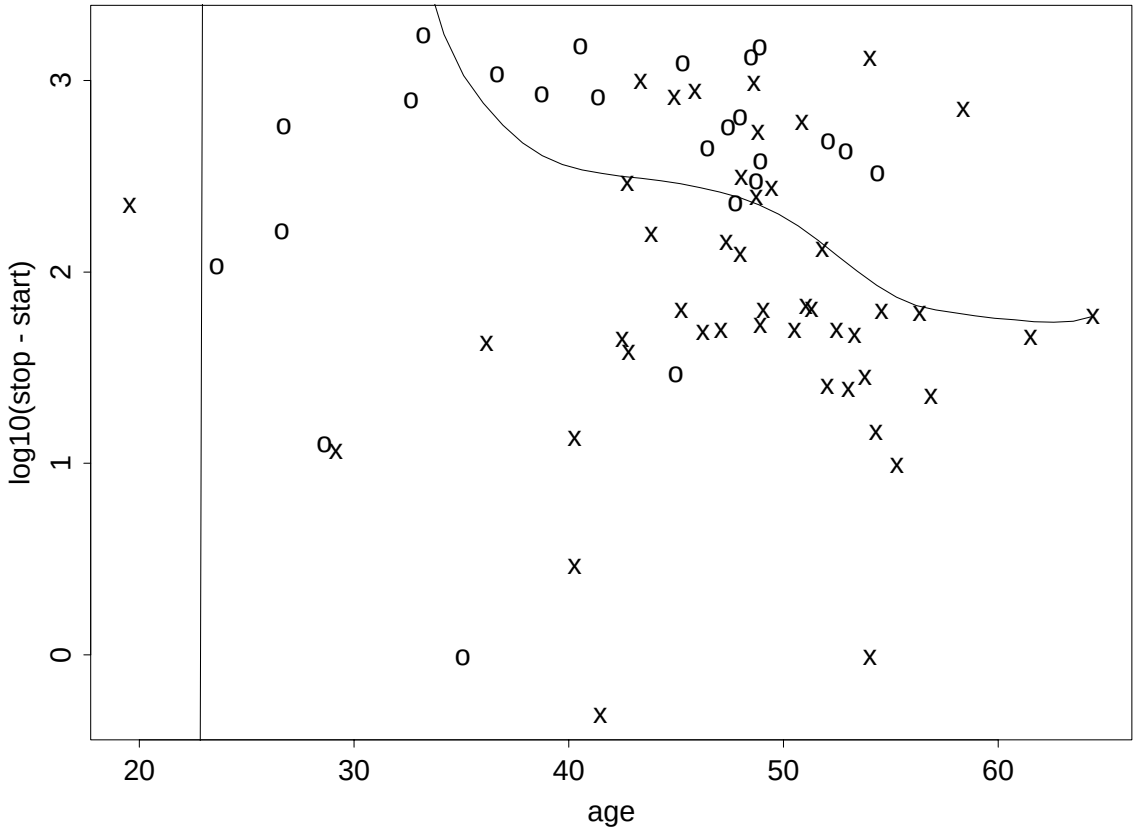
Adding Covariates

There have been a number of approaches to model the effect of covariates on survival without a parametric model. Perhaps the simplest is a localized version of the Kaplan-Meier estimator

$$\hat{S}(t \mid x) = \prod_{t_i \leqslant t, \delta_i = 1} \left[1 - \frac{w(x_i - x)}{\sum_{j \in R(t_i)} w(x_j - x)}\right]$$

which includes observations with weights depending on the proximity of their covariates to x. This does not smooth the survivor function, but the function `sm.survival` in library `sm` (Bowman & Azzalini, 1997) plots quantiles as a function of x by smoothing the inverse of the survival curve and computing quartiles of the smoothed fit. Following them, we can plot the median survival time after transplantation in the Stanford heart transplant data `heart` by

```
library(sm); attach(heart[heart$transplant==1,])
sm.survival(age+48, log10(stop - start), event, h=5, p=0.50)
```

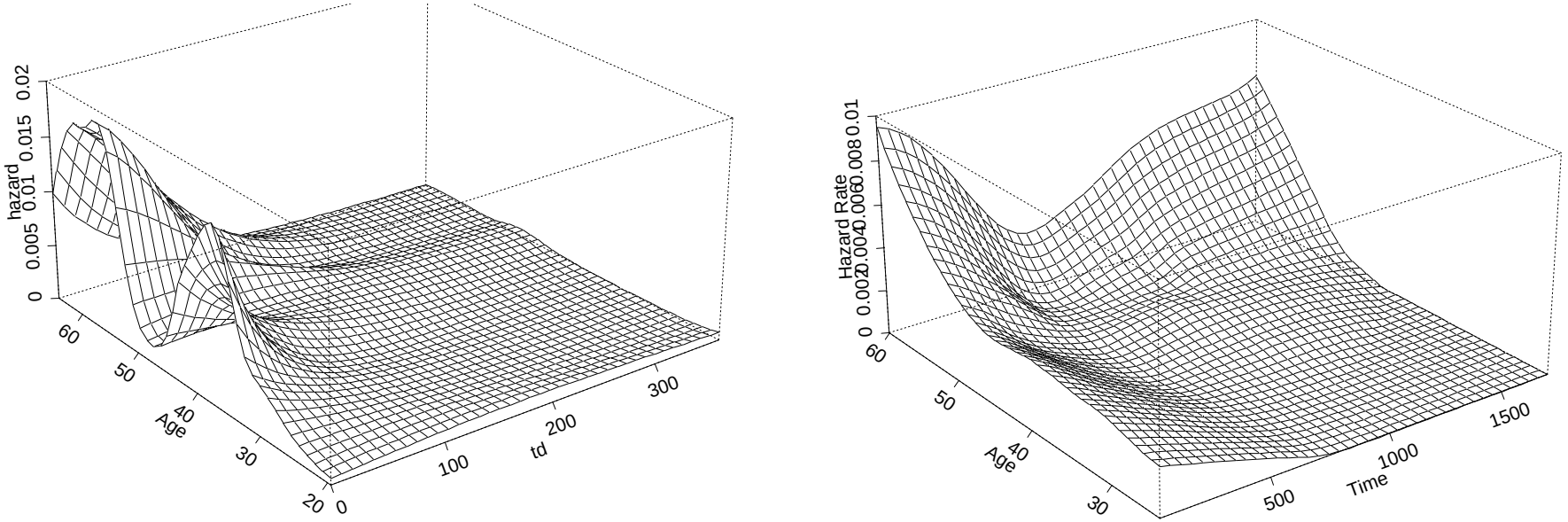


Median survival time for the Stanford heart transplant data by `sm.survival`. Deaths are marked by `x`, censorings by a circle.

This shows some evidence of a decline with age.

Likelihood approach

The local likelihood approach easily generalizes to localizing in covariate space too.



Smooth hazard functions (in days) as a function of age in the Stanford heart-transplant study by **left** `locfit` and **right** by `hazcov`.

Smooth functions of covariates

A Weibull model fits the AIDS dataset well, and the fitted baseline survival is almost exponential.

```
aids.wei <-
  survreg(Surv(survtime + 0.9, status) ~  state + T.categ + sex + age,
          data=Aidsp)
summary(aids.wei, correlation=F)
    ....
```

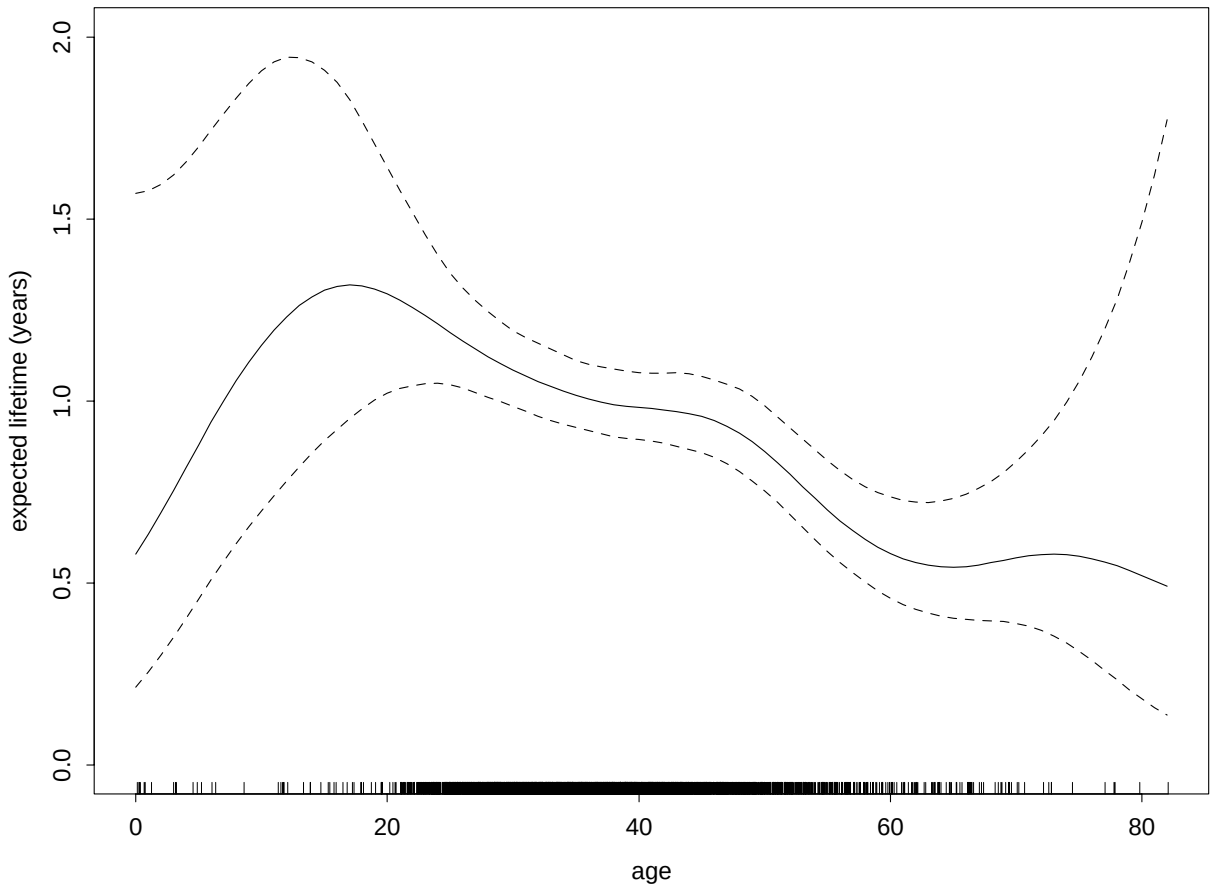
Coefficients:

	Value	Std. Error	z	p
(Intercept)	6.41825	0.2098	30.5970	1.34e-205
stateOther	0.09387	0.0931	1.0079	3.13e-01
stateQLD	-0.18213	0.0913	-1.9956	4.60e-02
stateVIC	-0.00750	0.0637	-0.1177	9.06e-01
T.categhsid	0.09363	0.1582	0.5918	5.54e-01
T.categid	0.40132	0.2552	1.5727	1.16e-01
T.categhet	0.67689	0.2744	2.4667	1.36e-02
T.categhaem	-0.34090	0.1956	-1.7429	8.14e-02
T.categblood	-0.17336	0.1429	-1.2131	2.25e-01
T.categmother	-0.40186	0.6123	-0.6563	5.12e-01
T.categother	-0.11279	0.1696	-0.6649	5.06e-01
sex	-0.00426	0.1827	-0.0233	9.81e-01
age	-0.01374	0.0026	-5.2862	1.25e-07
Log(scale)	0.03969	0.0193	2.0572	3.97e-02

Scale= 1.04

We also considered parametric non-linear functions of age by using a spline function. We use the P-splines of Eilers & Marx (1996) ('poor man's smoothing splines') as this is implemented in both survreg and coxph in survival5.

```
> aids.ps <-
  survreg(Surv(survtime+0.9,status) ~ state + T.categ + pspline(age,df=6),
          data=Aidsp)
```



Predicted survival versus age of a `NSW hs` patient (solid line), with pointwise 95% confidence intervals (dashed lines) and a rug of all observed ages.

Multivariate Analysis

Correspondence Analysis

A graphical technique, from the French ‘Data Analysis’ school of discrete multivariate analysis.

Original form applies to two-way tables of counts.

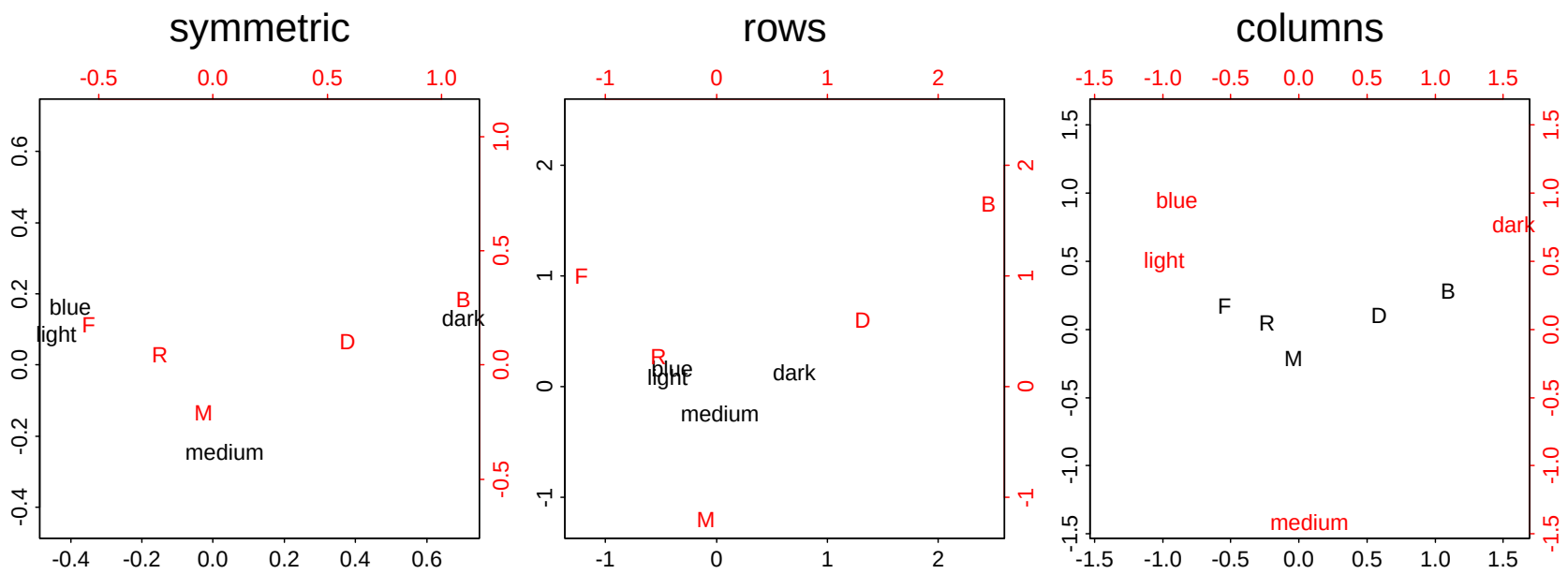
For example, consider Fisher’s (1940) example on colours of eyes and hair of people in Caithness, Scotland:

	fair	red	medium	dark	black
blue	326	38	241	110	3
light	688	116	584	188	4
medium	343	84	909	412	26
dark	98	48	403	681	85

Correspondence analysis seeks ‘scores’ f and g for the rows and columns which are maximally correlated (but not constant).

```
> corresp(caith)
First canonical correlation: 0.44637

 Row scores:
    blue    light   medium   dark
-0.89679 -0.98732 0.075306 1.5743

 Column scores:
   fair      red    medium   dark  black
-1.2187 -0.52258 -0.094147 1.3189 2.4518
```

There are various ways to plot these scores graphically, depending if we want the rows to explain the columns or the columns to explain the rows or to treat them symmetrically, as here.

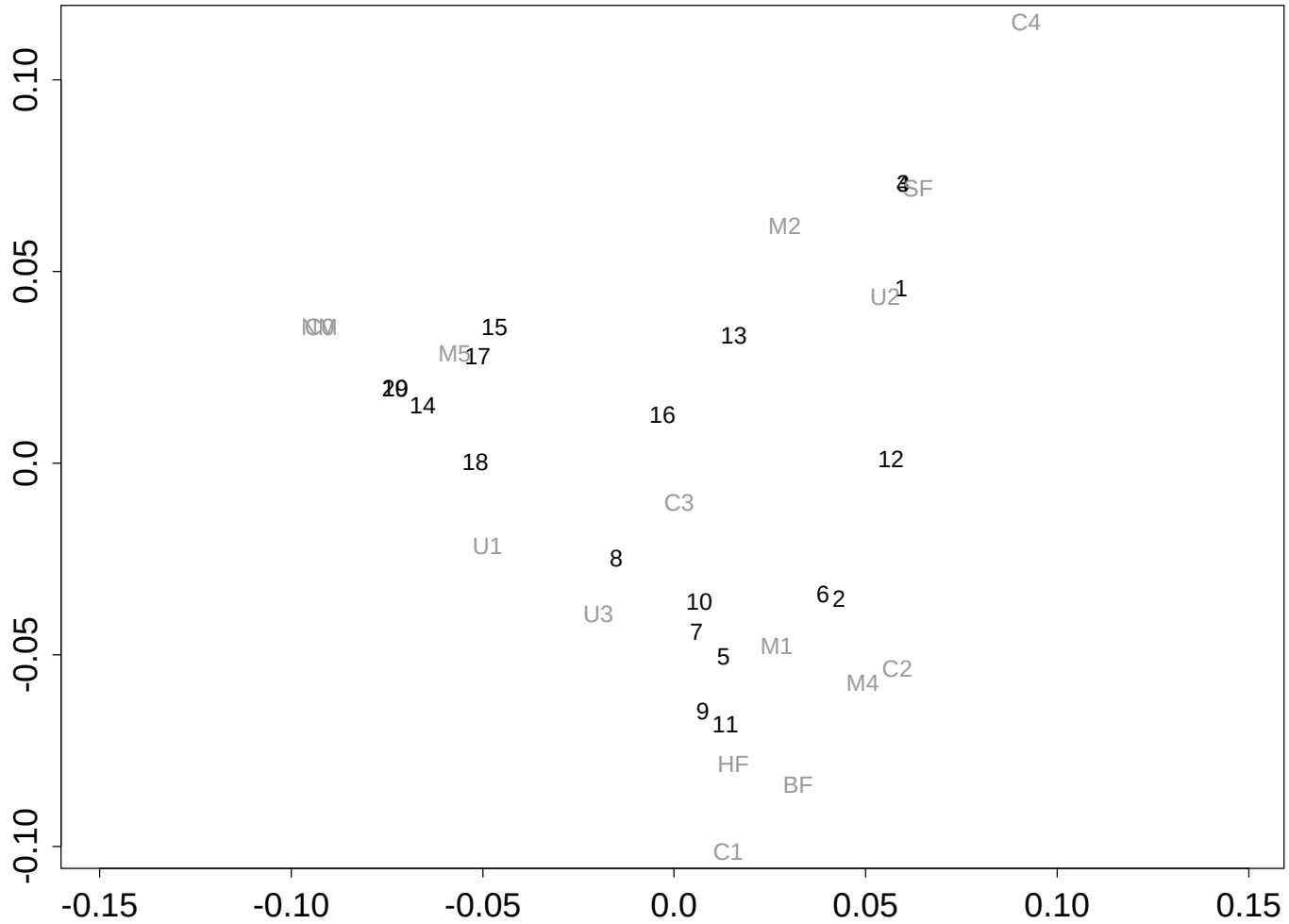


Three variants of correspondence analysis plots from Fisher's data on people in Caithness: (left) 'symmetric", (middle) 'row asymmetric' and (right) 'column asymmetric'.

Note that the symmetric plot (left) has the row points from the asymmetric row plot (middle) and the column points from the asymmetric column plot (right) superimposed on the same plot (but with different scales).

Multiple correspondence analysis

Multiple correspondence analysis (MCA) is (confusingly!) a method for visualizing the joint properties of $p \geqslant 2$ categorical variables that does *not* reduce to correspondence analysis (CA) for $p = 2$, although the methods are closely related.



Multiple correspondence analysis plot of data on 20 farms on the Dutch island of Terschelling. Numbers represent the farms and labels levels of moisture, grassland usage, manure usage and type of grassland management.

Discriminant Analysis

Discriminant analysis means several things:

- Fisher's (1936) LDF showing the difference between two groups, by maximizing the ratio of the between-group to the within group variance.
- The extensions to more than two groups by Rao (1948), Bryan (1951) and others, which maximizes a ratio of (co)variance matrices (in some senses).
- 'Allocation' procedures based on normal distributions and the posterior probabilities of allocating observations to the different groups.
  - Linear discriminant analysis, where the populations are assumed to have different means but the same variance matrix.
  - Quadratic discrminant analysis, where the populations are assumed to have different (but normal) distributions.
  - Variations in which the variance matrices are restricted.

The `MASS` library has had `lda` and `qda` for many years (indeed, prior to MASS1), and S-PLUS 2000 now has the closely related `discrim`.

All have menu interfaces in S-PLUS 2000.

Some theory

(But not much!)

We have a set of g classes, and for each of n cases we know the class (assumed correctly). We have p measurements on each case.

W is the within-class variance matrix, that is the covariance matrix of the variables centred on the class means.

B is the between-classes variance matrix, that is, of the predictions by the class means. Has rank at most $r = \min(p, g-1)$.

Fisher's LDF is a linear combination $\boldsymbol{x}^T\boldsymbol{a}$ of the variables maximizing $\boldsymbol{a}^T B\boldsymbol{a}/\boldsymbol{a}^T W\boldsymbol{a}$, originally for just two groups so the numerator was the squared difference in group means.

Easiest to think of this in two steps:

(i) *Sphere* the data, that is take a linear transform so that W is the identity.

(ii) Rotate the space to the r dimensions spanned by B.

Then LDF plots the data on up to the r dimensions given in the second step.

Sphering is done by taking the principal components of the data matrix centred on the group means, and rescaling each PC to unit variance.

(Lots of traps: suppose a linear combination is constant within a group. Is it constant between groups? Rounding errors?)

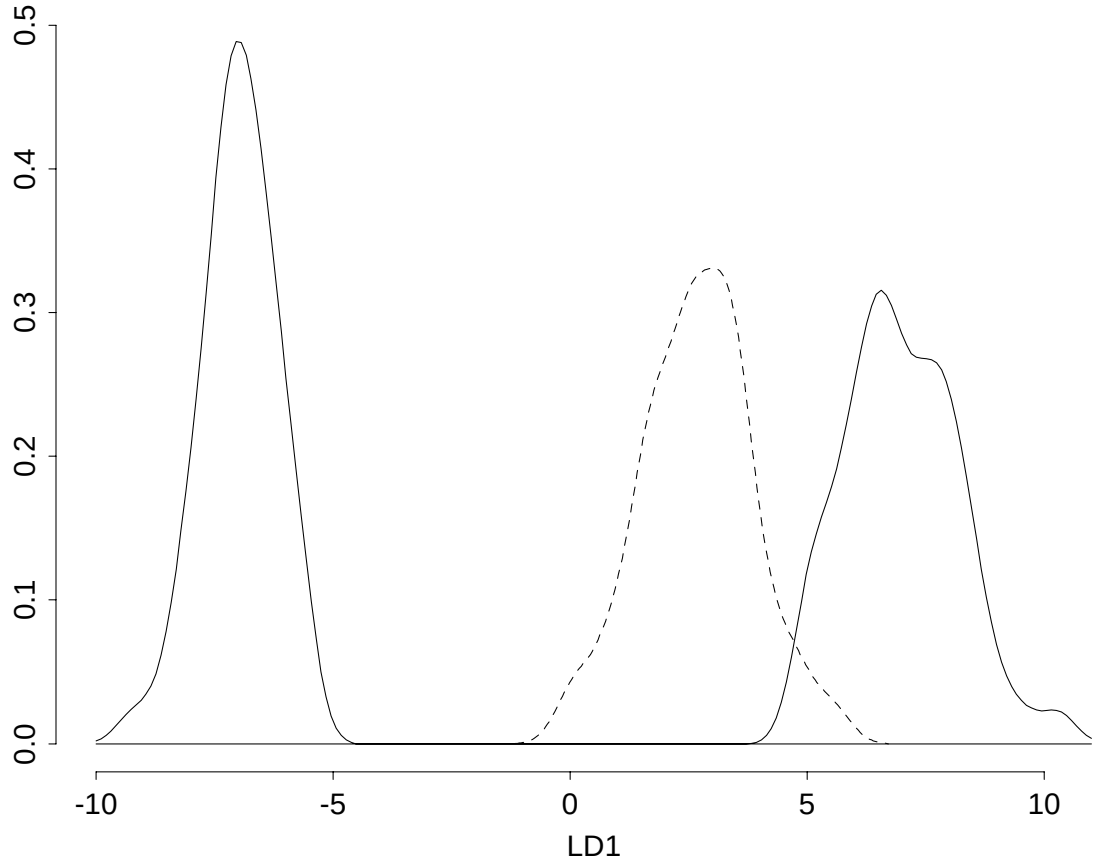
The rotation of the space is done by finding principal components of the sphered data.

There are different definitions of B. Most people weight by the observed group size, Rao did not weight at all, and `lda` weights by the population prevalences if these are given.

Examples

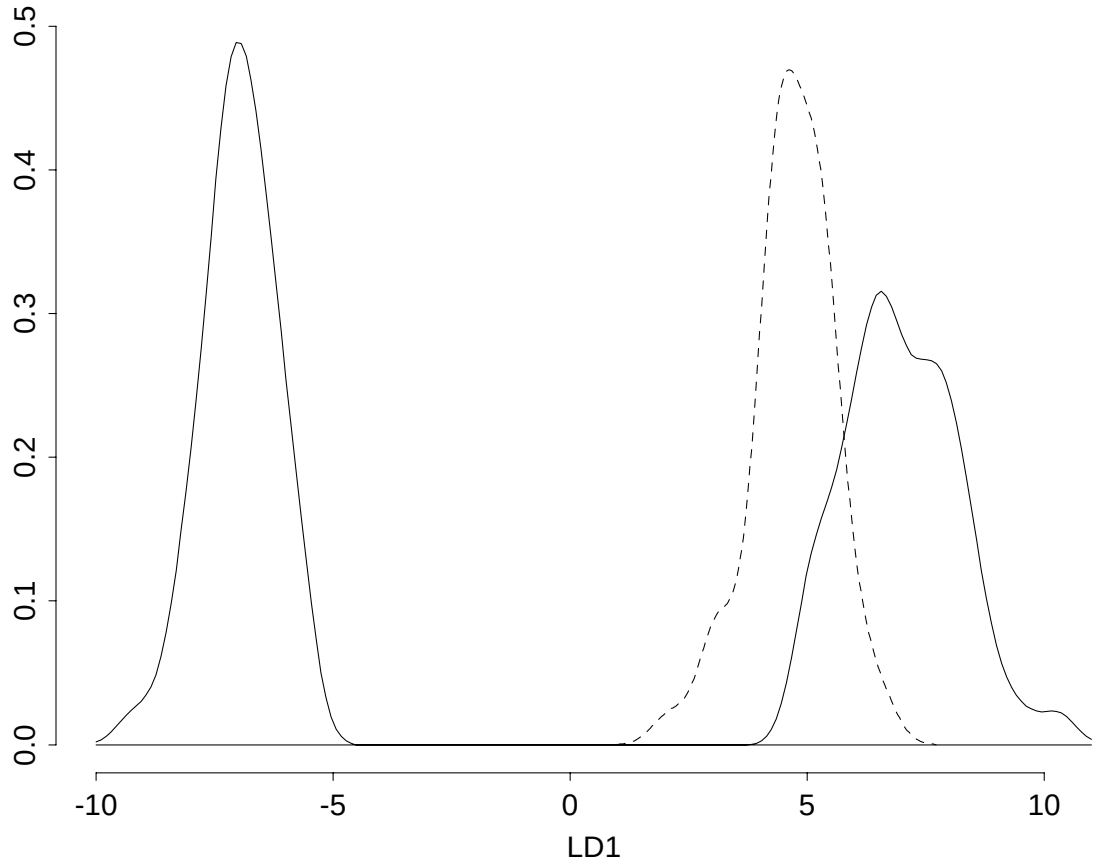
First, Edgar Anderson's data on Irises of the Gaspe Peninsula as used by Fisher (1936). Three species, 50 specimens from each, 4 physical measurements (`(petal, sepal)` $\times$ `(length, width)`).

Fisher was interested in the genetic makeup of the hexaploid *I. versicolor* as a product of the merging the other two. We can reproduce this by finding the LDF on the 'outer' two groups and plotting density plots of all three (*versicolor* is dashed).



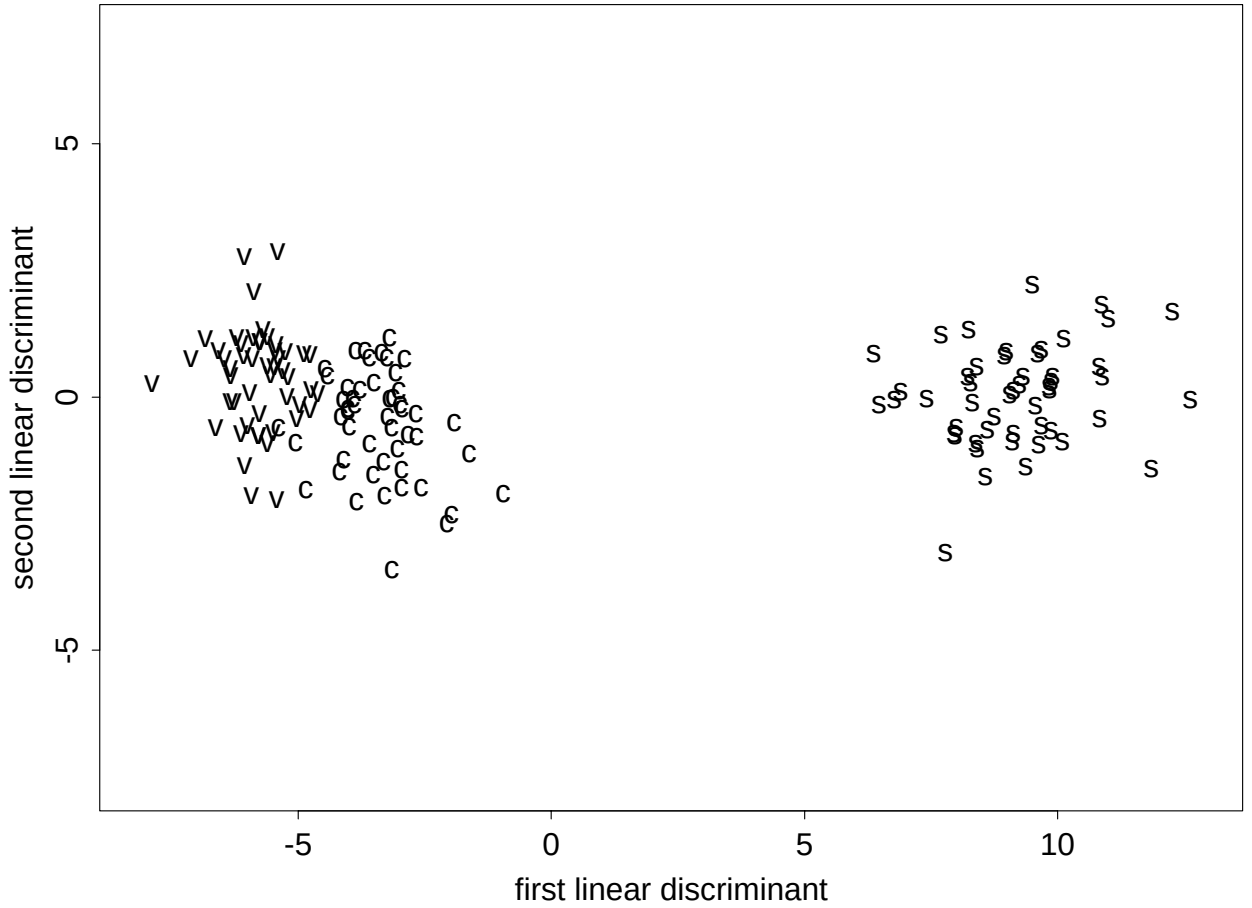
Fisher’s observation of a 1:2 mixture is confirmed.

Wait a minute! Are the variances equal? Do we not recommend transformations of size measurements to log scale?



Not so clearly 1:2!

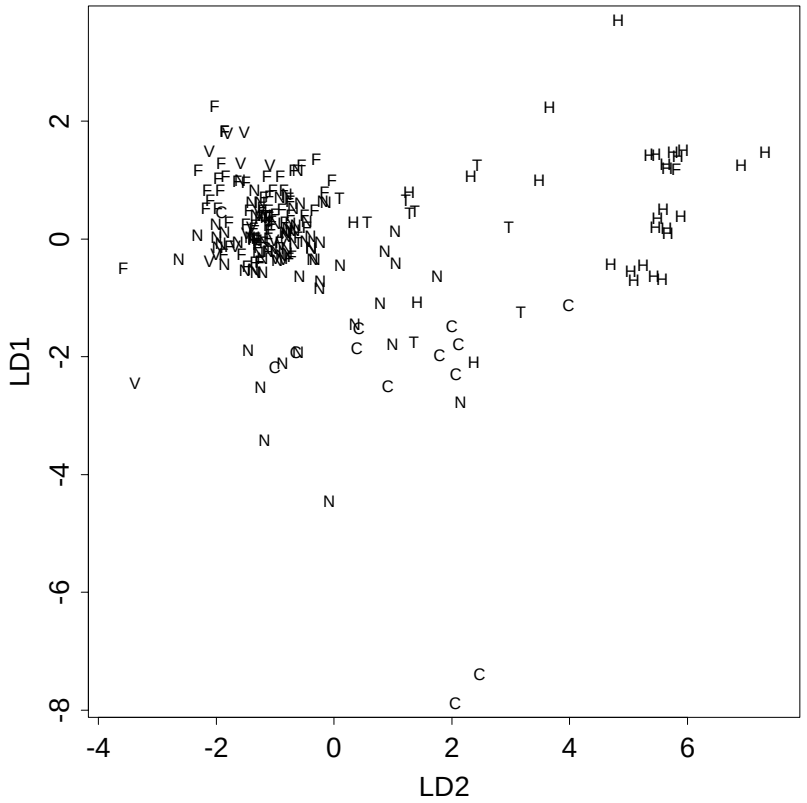
It is helpful to look at LDA for all three groups, using log-scale:



The result shows that the means are nearly collinear.

Forensic glass data

Six groups, widely different scatters.



Dominated by a few outliers.

‘Allocation’ approach

Let π_c denote the prior probabilities of the classes.

$p(\boldsymbol{x} \mid c)$ the densities for the observations $\boldsymbol{x}$ from class c.

The posterior distribution after observing a future $\boldsymbol{x}$ is

$$p(c \mid \boldsymbol{x}) = \frac{\pi_c \, p(\boldsymbol{x} \mid c)}{p(\boldsymbol{x})} \propto \pi_c \, p(\boldsymbol{x} \mid c)$$

Bayes rule: choose the class which maximizes $p(c \mid \boldsymbol{x})$: has the smallest expected number of errors.

QDA

$p(\boldsymbol{x} \mid c)$ is Normal, mean $\boldsymbol{\mu}_c$, variance Σ_c. Bayes rule minimizes

$$\begin{aligned} Q_c &= -2 \log p(\boldsymbol{x} \mid c) - 2 \log \pi_c \\ &= (\boldsymbol{x} - \boldsymbol{\mu}_c)\Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)^T + \log |\Sigma_c| - 2 \log \pi_c \end{aligned} \tag{3}$$

The first term of (3) is the squared *Mahalanobis distance* to the class centre, and can be calculated by the S function `mahalanobis`.

The difference between the Q_c for two classes is a quadratic function of $\boldsymbol{x}$, o the method is known as *quadratic discriminant analysis* and the boundaries of the decision regions are quadratic surfaces in $\boldsymbol{x}$ space.

LDA

Now suppose that the classes have a common variance matrix Σ. Differences in the Q_c are now *linear* functions of $\boldsymbol{x}$, and we can maximize $-Q_c/2$ or

$$L_c = \boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_c^T - \boldsymbol{\mu}_c\Sigma^{-1}\boldsymbol{\mu}_c^T/2 + \log \pi_c \tag{4}$$

'Plug-in' rules

To use (3) or (4) we have to estimate $\boldsymbol{\mu}_c$ and Σ_c or Σ. The obvious estimates are used, the sample mean and covariance matrix within each class, and W for Σ.

Then LDA coincides with the Bryan version of LDF, but that does not tell one how to allocate. Choosing the nearest group centre in an LDF plot is equivalent to assuming equal (π_c).

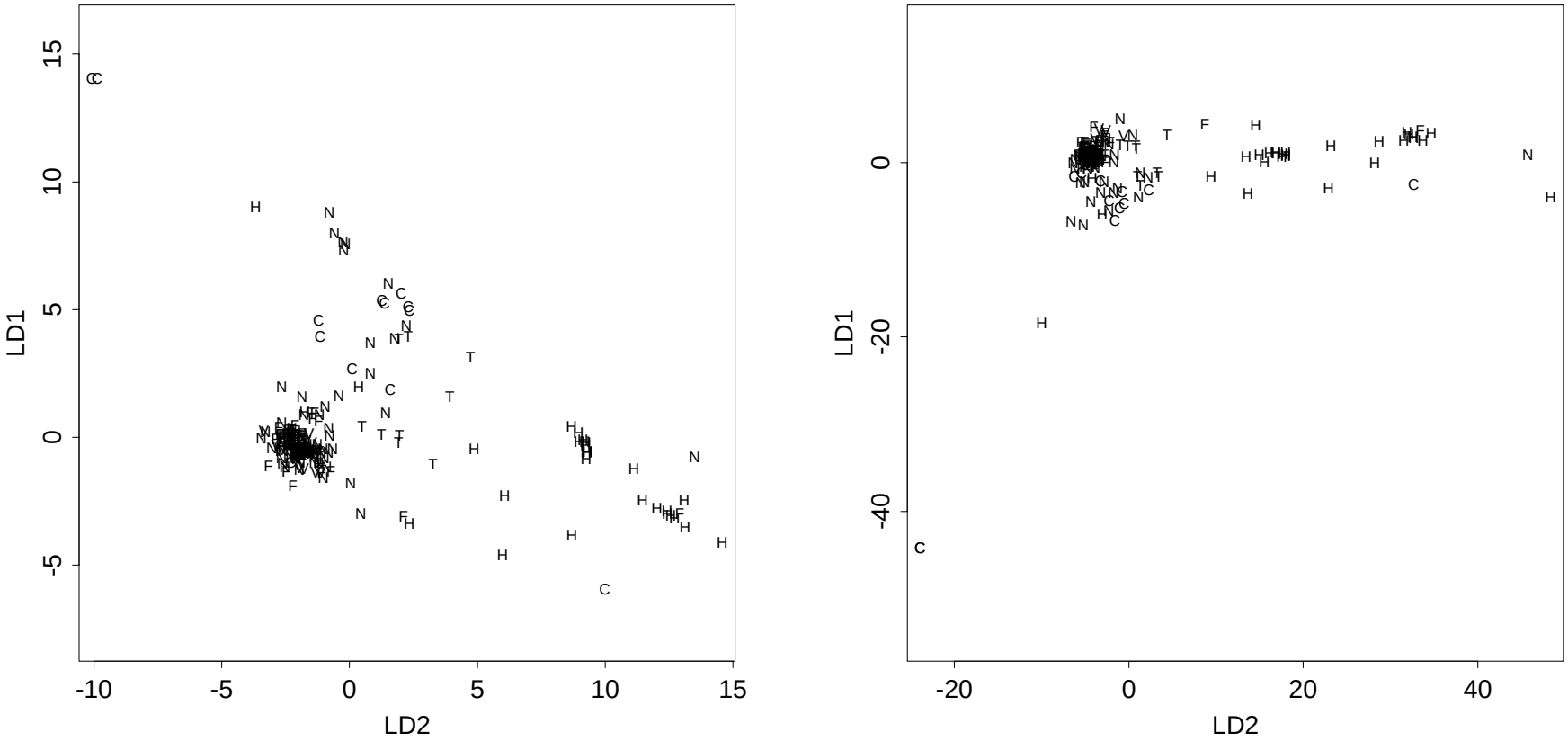
Snags in LDA / QDA

We have made a lot of dubious assumptions.

- Those ‘obvious estimates’ are none too obvious on closer inspection. We have unbiased estimates of $\boldsymbol{\mu}_c$ and Σ, but we use them in complex expressions. Might do better to have unbiased estimates of $p(\boldsymbol{x} \mid c)$ or $\log p(\boldsymbol{x} \mid c)$ or $\log p(c \mid \boldsymbol{x})$. Last two are option `"debiased'` in `lda`.
- We have assumed that the estimates are the true values, and ignored the variability in the estimates (even if we correct the bias). Can matter if the groups are of rather different sizes. *Predictive* methods average over the uncertainty in the estimates.
- LDA / QDA are very far from robust to non-normality.
  - real distributions might be normal, but might have errors in measurements.
  - real distributions might be non-normal.

All of these can make large differences, and ways around them are provided for `lda` and `qda`.

However, they probably indicate that discriminant analysis is not really a competitive technique these days, compared to logistic discrimination and non-linear extensions such as neural networks.



For the forensic glass data we can assume longer-tailed (t) distributions (left) or resistant estimates of the variance matrix (right).

Cushing's syndrome

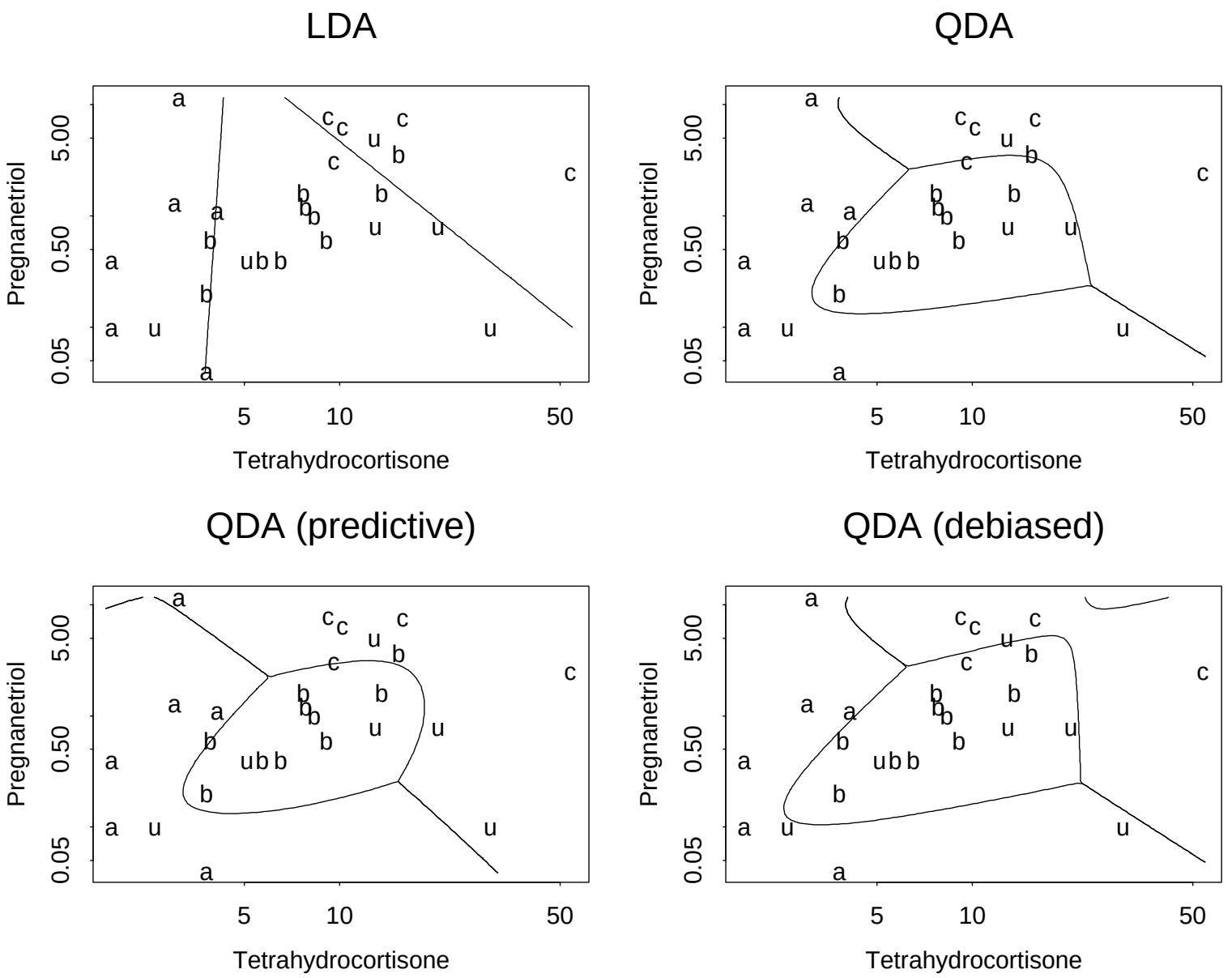
The data are on diagnostic tests on patients with *Cushing's syndrome*, a hypersensitive disorder associated with over-secretion of cortisol by the adrenal gland.

This dataset has three recognized types of the syndrome represented as a, b, c. (These encode 'adenoma', 'bilateral hyperplasia' and 'carcinoma', and represent the underlying cause of over-secretion. This can only be determined histopathologically.)

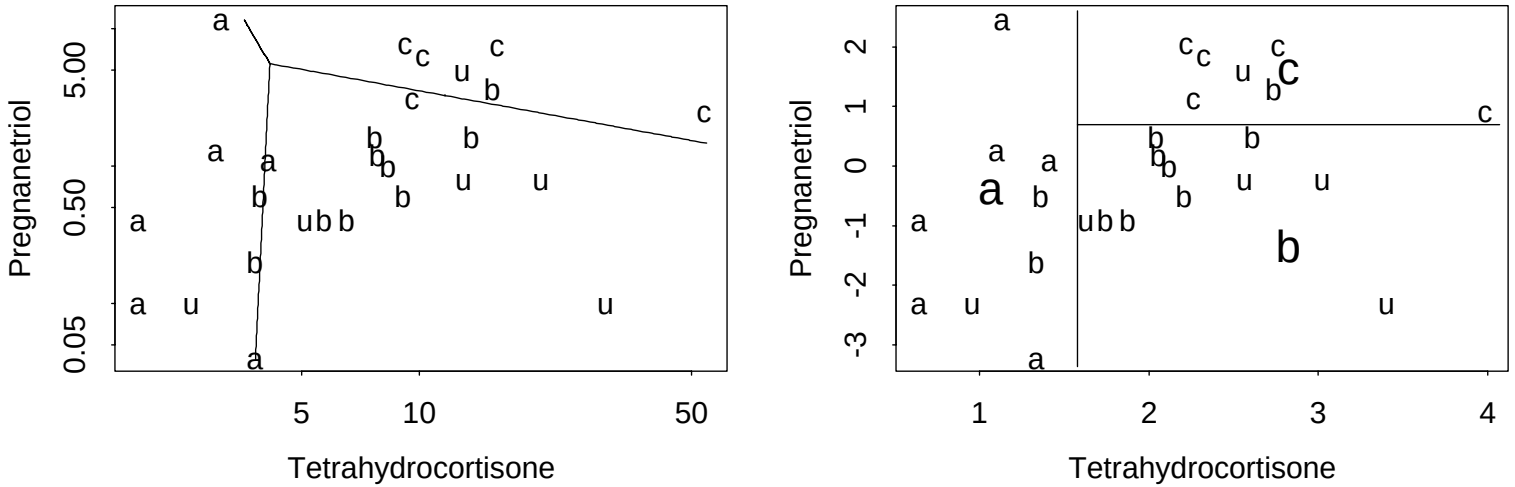
The observations are urinary excretion rates (mg/24h) of the steroid metabolites tetrahydrocortisone and pregnanetriol, and are considered on log scale.

There are six patients of unknown type (marked u).

Linear discriminant analysis is clearly inadequate: the various types of QDA differ quite a bit.



Linear and quadratic discriminant analysis applied to the Cushing's syndrome data.



Logistic regression and classification trees applied to the Cushing's syndrome data.

Case Study:
Magnetic Resonance Imaging of Brain Function

with Jonathan Marchini