MSG500/MVE190
Linear Models - Lecture 12

Rebecka Jörnsten
Mathematical Statistics
University of Gothenburg/Chalmers University of Technology

November 29, 2012

1 Generalized linear models

This lecture we will discuss how you fit logistic regression models to data. Along the way, we will learn about weighted least squares (WLS) and nonlinear regression models (NLM).

- Linear model: assumptions and key results
  - $y = X\beta + \epsilon$ sufficient model
  - $E(\epsilon) = 0$, symmetric
  - $V(\epsilon) = \sigma^2$, constant error variance
  - Uncorrelated errors, or slightly stronger independent observations y_i
  - no outliers
  - May also assume $\epsilon \sim N(0, \sigma^2)$ -> t-test, F-test
  - Closed for LS solution $\hat{\beta} = (X'X)^{-1}X'y$, $\hat{\sigma^2} = RSS(\hat{\beta})/(n-p)$
- Nonlinear model:
  - $y = f(x;\beta) + \epsilon$, where form of f assumed known
  - $E(\epsilon) = 0$, symmetric
  - $V(\epsilon) = \sigma^2$, constant error variance
  - Uncorrelated errors, or slightly stronger independent observations y_i
  - no outliers
  - May also assume $\epsilon \sim N(0, \sigma^2)$ -> F-test, approx. t-test
  - Generally no closed form solution: solve for $\hat{\beta}$ using *iterative least squares*
  - Caution: starting values and convergence issues, exact confidence intervals for $\hat{\beta}$ may be difficult to construct
- Generalized linear model:
  - $y \sim f_Y(x;\beta)$, where the distribution f_Y assumed known, e.g. Gaussian, Poisson, Binomial.
  - The f_Y induces a known Expectation-Variance relationship; $E(y_i) = b'(\theta_i)$, $V = \phi b''(\theta_i)$, where $b()$ is a function specified by the distribution f_Y. θ is called the *canonical parameter* for which the likelihood takes on the most simple form (see below).
    $y_i \sim Poisson(\mu_i) \rightarrow E(y_i) = \mu_i,\ V(y_i) = \mu_i$
    $y_i \sim Binomial(m_i, \pi_i) \rightarrow E(y_i) = m_i\pi_i,\ V(y_i) = m_i\pi_i(1-\pi_i)$
  - y_i are assume drawn independently from $f_Y(x_i, \beta)$
  - We assume β and the mean parameter μ are related with the known *link function* such that: $E(y_i) = \mu_i$ and $g(\mu_i) = \eta_i = x_i\beta$). η_i is called the *linear predictor*.
  - Analogously to the model sufficiency of a linear model, we assume that the link function is correct.

– and no outliers...

The theory for generalized linear models (GLM) is based on maximum likelihood estimation for *exponential family distributions*. Such distributions can be written as:

$$f(y|\theta,\phi) = exp\big[\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\big]$$

The parameter θ is the mean parameter, which we will assume is observation specific in the GLM setting ($\theta_i = \theta(x_i)$) and a function of the regression parameters β. ϕ is a scale parameter, analogue of σ^2 in the normal linear model. $c(y,\phi)$ is a normalizing constant to make f_Y a probability density. The key component we work with is $(y\theta - b(\theta))$, which is the analogue of the RSS in linear regression.
Here are some examples:

Normal $f(y) = \frac{1}{2\pi\sigma^2} exp(-\frac{1}{2\sigma^2}(y-\mu)^2),\ \mu = x\beta$
$\log(f(y)) = \frac{1}{\sigma^2}(y\mu - \mu^2/2) - \frac{1}{2}(y^2/\sigma^2 + log(2\pi\sigma^2)$
$\rightarrow\ \theta = \mu = x\beta,\ \phi = \sigma^2$ and $b(\theta) = \theta^2/2$
Since $\theta = g(\mu) = \mu$, the link function $g()$ is the identity.

Poisson $f(y|\mu) = \frac{e^{-\mu}\mu^y}{y!}$
$\log(f(y)) = y\log(\mu) - \mu - \log(y!)$
$\rightarrow\ \theta = x\beta = \log(\mu),\ \mu = e^\theta,\ \phi = 1$ and $b(\theta) = \mu = e^\theta$
The link function $g(\mu) = \log(\mu) = \theta = x\beta$

Binomial $f(y|m,\pi) = \binom{m}{y}\pi^y(1-\pi)^{m-y}$
$\log(f(y)) = y\log(\frac{\pi}{1-\pi}) + m\log(1-\pi) + \log(\binom{m}{y})$
$\rightarrow\ \theta = \frac{\pi}{1-\pi},\ \pi = \frac{e^\theta}{1+e^\theta},\ \phi = 1$ and $b(\theta) = m\log(1+e^\theta)$
The link function $g(\mu) = \log(\frac{\pi}{1-\pi}) = \theta = x\beta$, the logit link

1.1 Properties of exponential families

If we assume f_Y has the form of equation (1), the log-likelihood can be written as

$$l(\theta,\phi|y) = \sum_i \big(\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i,\phi)\big)$$

If we want to maximize this likelihood, we take the derivative with respect to θ and set it to 0. This is called the score function or equation:

$$U(\theta_i) = \frac{dl}{d\theta_i} = \frac{y_i - b'(\theta_i)}{\phi}.$$

One can actually show that the

$$E[score] = 0\ \rightarrow\ E[\frac{y_i - b'(\theta_i)}{\phi}] = 0\ \rightarrow\ E[y_i] = b'(\theta_i)$$

and similarly from the second derivative of the likelihood one can show that

$$V(y_i) = \phi b''(\theta_i).$$

It is thus the $b()$ function in the f_Y that specifies the mean-variance relationship of the distribution. (Check this for the Poisson and Binomial examples above).

2 Iterative weighted least squares

How to we fit GLMs to data? We have two problems that need to be addressed; (1) Usually, $E(y_i) = \mu_i = g^{-1}(\theta_i = x_i\beta)$ is *nonlinear* in β
(2) Since $V(y_i)$ depends on the mean $E(y_i)$ we have non-constant variance.

Problem (2) is the easiest one to address: we will simply use *weighted least squares*. Problem (1) is more complicated. We will use an *iterative least squares* procedure for this. Together, this results in the Iterative (re)weighted least squares algorithm.

2.1 Weighted least squares

Let us say that $V(y_i) = \sigma_i^2$, not constant. We can write this for the observation vector as $V(y) = \sigma^2 V$, where V is then a relative variance matrix. If it diagonal, we are simply allowing for different error variance. If it contains non-diagonal elements, we are allowing for some correlation between observations. If we for now ignore this variance structure and use least squares we obtain the solution $\hat{\beta} = (X'X)^{-1}X'y$ which is unbiased even when V is not the identity and with variance

$$V(\hat{\beta}) = \sigma^2(X'X)^{-1}(X'VX)(X'X)^{-1}$$

Let us now try using weighted least squares instead. We minimize

$$\sum_i w_i(y_i - x_i\beta)^2 = \sum_i (w_i^{1/2}(y_i - x_i\beta))^2 = ||W^{1/2}y - W^{1/2}X\beta||^2 = ||\tilde{(}y) - \tilde{(}X)\beta||^2$$

where $\tilde{y} = W^{1/2}y$ and $\tilde{X} = W^{1/2}X$ are transformed variables. We solve the least squares problem in the transformed system and obtain

$$\hat{\beta} = (\tilde{X}'\tilde{X})^{-1}\tilde{X}'\tilde{y} = (X'WX)^{-1}X'Wy$$

Now, if $V(y) = \sigma^2 V$ and we use weights $W = V^{-1}$

$$V(\hat{\beta}) = \sigma^2(X'WX)^{-1}$$

which one can show is the best (minimum) variance estimator (unbiased) that we can find. So we should use weights w_i inversely proportional to the observation variance.

Of course, in practise we won't know the variances, σ_i^2. However, there is an easy way to estimate them. Let us assume we first fit a model using ordinary least squares. We obtain residuals, e_i. Now, one of the diagnostic plots we have used to check for constant error variance is to plot $|e_i|$ or e_i^2 versus the fitted values $\hat{y}_i$ or an x-variable. We will now use the tool to estimate the non-constant error variance instead. Under the model assumption $E[\epsilon_i] = 0$ and $V[\epsilon_i] = E[\epsilon_i^2] = \sigma_i^2$. We will use e_i^2 as a new response variable and use either the fitted values $\hat{y}_i$ or x-variables, or transformations thereof, as the predictors. We make a second regression for e_i^2 and use the fitted values from this second model as estimates $\hat{\sigma}_i$. We the use these as weights in a weighted least squares fit: $w_i = 1/\hat{\sigma}_i^2$. If necessary, we can iterate this process a couple of times until convergence (usually this procedure converges in just a few cycles).

We now know how to incorporate the non-constant variance. For GLMs we will know how the variance depends on the mean parameter, $V(y_i) = \phi b''(\theta_i)$, e.g. $V(y_i) = \pi_i(1 - \pi_i)$ when y is Binomial. In an iterative scheme, given an estimate $\hat{\pi}_i$ we can compute an estimate of $V(y_i)$ and therefore the weights, $w_i = 1/V(y_i) = 1/(\hat{\pi}_i(1 - \hat{\pi}_i))$ in the Binomial case.

2.2 Nonlinear regression

We are now ready to address the second problem, that the mean function is non-linear in the regression parameters β.

In nonlinear model, $y = f(x;\beta) + \epsilon,\ \epsilon \sim N(0, \sigma^2)$. We estimate β by minimizing the nonlinear least squares:

$$\min_\beta \sum_i (y_i - f(x_i;\beta)^2 = ||y - f(x;\beta)||^2.$$

We take the derivative of the NLS with respect to parameter β_j and set it to 0:

$$\frac{\partial NLS}{\partial \beta_j} = -2\sum_i (y_i - f(x_i;\beta)\frac{\partial f(x_i;\beta)}{\partial \beta_j} = 0,\ \forall\ j$$

We denote the gradients $\frac{\partial f(x_i;\beta)}{\partial \beta_j} = (\nabla f)_j$. Re-arranging the above expression we can write:

$$\sum_i y_i \frac{\partial f(x_i;\beta)}{\partial \beta_j} = \sum_i f(x_i;\beta)\frac{\partial f(x_i;\beta)}{\partial \beta_j},\ \forall\ j$$

or, stacking these for all j and writing in a matrix form:

$$(\nabla f)^T y = (\nabla f)^T f, \tag{1}$$

where both f and ∇f may depend on β in a complex fashion.

Solving equation (2) may not be easy, and generally does not lead to a closed-form expression for β. The solution is instead obtained through a sequence of linearized version of the nonlinear problem.

We write $y = f(x;\beta) + \epsilon$. We linearize f near a point β^0:

$$f(x;\beta) \simeq f(x;\beta^0) + \sum_j (\beta_j - \beta_j^0)\frac{\partial f}{\partial \beta_j}|_{\beta^0}, \tag{2}$$

where $\frac{\partial f}{\partial \beta_j}|_{\beta^0}$ is the gradient evaluated at β^0. The nonlinear surface f is thus approximated by the tangent plane at β^0. We denote $\frac{\partial f}{\partial \beta_j}|_{\beta^0} = Z_j^0$ and $f(x;\beta^0) - \sum_j \beta_j^0 Z_j^0 = W^0$, then we can write

$$f(x;\beta) \simeq W^0 + Z_j^0 \beta.$$

We think of Z_j^0 as the "new x-variables" and W^0 as a known intercept in a regression model. We thus update the β by regression y on Z^0, i.e. assume a model approximation

$$(y - W^0) = Z^0\beta + \epsilon.$$

This is a *linear model* so we can solve for β using least squares:

$$\hat{\beta} = ((Z^0)^T Z^0)^{-1}(Z^0)^T(y - W^0).$$

This gives us a new points $\beta^0 = \hat{\beta}$ to linearize f around. We repeat this process until convergence. Note, this algorithm can be quite sensitive to the chosen starting value β^0. You should try a few different starting points to make sure you are not converging to a local optimum.

Inference for non-linear models shares some similarities with linear models. Since the errors are additive and normally distributed, nested models can be compared using the $F - test$. Also, AIC and BIC can also be used for model selection since they are also based on just residual sums of squares. Testing and confidence intervals for parameter β_j is more complicated for nonlinear models. The reason is that since $\hat{\beta}$ is no longer linear in y, $\hat{\beta}$ does not in general follow a normal distribution and the pivot $\frac{(\hat{\beta}_j - \beta_j)}{se(\hat{\beta}_j)}$ is no longer t-distributed. In addition, the standard error $se(\hat{\beta}_j)$ may not be known. We got a simple expression for this when we had a closed for solution for β in the linear model: $V(\hat{\beta}) = \sigma^2(X'X)^{-1}$ and so $se(\hat{\beta}_j) = \hat{\sigma}(X'X)^{-1}_{jj}$. Since the nonlinear regression fit was obtained through iterations of least squares fit, we use the last linearization to obtain and estimate of the standard error. The final set of gradients (treated as x-variables in the final iteration) are collected into a matrix

$$F_{ij} = \frac{\partial f(x_i;\hat{\beta}^{final})}{\partial \beta_j},\ F = F_{ij}.$$

We approximate

$$V(\hat{\beta}^{final}) = \hat{\sigma^2}(F^T F)^{-1},\ se(\hat{\beta}_j) = \hat{\sigma}\sqrt{(F^T F)^{-1}_{jj}}.$$

From this we can construct confidence intervals:

$$CI[\beta_j] = [\hat{\beta}_j \pm t_{n-p}(1-\alpha/2)\hat{\sigma}\sqrt{(F^T F)^{-1}_{jj}}].$$

Are there any problems with this approximation? (1) The linear approximation at the final iteration may be a poor approximation, and then the CI baed on it can be misleading. If the f is very nonlinear near $\hat{\beta}^{final}$, then perhaps the CI should be asymmetric. (2) Columns of F can be correlated which would lead to instability in the estimates of β (collinearities).

We can check the linear approximation at the final iteration use the *profile function*,

$$\tau(\beta_j) = sign(\beta_j - \hat{\beta}_j)\sqrt{F(\beta_j)},\ F(\beta_j) = \frac{RSS(\beta_j) - RSS(\hat{\beta}_j)}{\hat{\sigma^2}}.$$

That is, we compute the RSS near the fit $\hat{\beta}$, changing the jth estimate to β_j and keeping other βs fixed at there estimate $\hat{\beta}$. If $\sqrt{F(\beta_j)}$ is close to linear in β_j, then $RSS(\beta_j)$ is near quadratic in β_j and thus the linear approximation is quite good. If this is the case, we trust the CIs above. If the profile function indicates that the f is not linear in β_j near the solution, we can't trust the above CIs. We can instead use the profile of the likelihood to construct intervals (for details see Inference class). What you do is interpolate the profile (or the likelihood) near $\hat{\beta}_j$ until the likelihood ratio to the left and to the right hits the level $\chi^2_1(1-\alpha)$ and read off the corresponding values of β_j that then constitute the upper and lower end of the interval. This is an approximate confidence interval but allows for asymmetry which the t-test does not. A second alternative is to resampling techniques like bootstrap (which we will cover in a future lecture if time permits).

We will work with NLS using a radioactive decay data set. We have data of radioactive counts from blood samples observed at times (days) after an injection of a radioactive substance.

```
> library(stats)
> time <- c(0, 1, 2, 3, 4, 7, 9, 11, 14, 16, 18, 21, 24, 29, 32,
+     35, 38, 46)
> count <- c(5149, 5435, 5358, 5462, 4755, 4457, 4538, 4378, 4189,
+     4195, 3743, 3855, 3569, 3045, 3091, 3064, 2729, 2477)
> dataf <- data.frame(cbind(count, time))
> names(dataf) <- c("count", "time")
```

We start by plotting the data:

```
> par(mfrow = c(1, 1))
> plot(dataf$time, dataf$count, xlab = "time", ylab = "count")
```

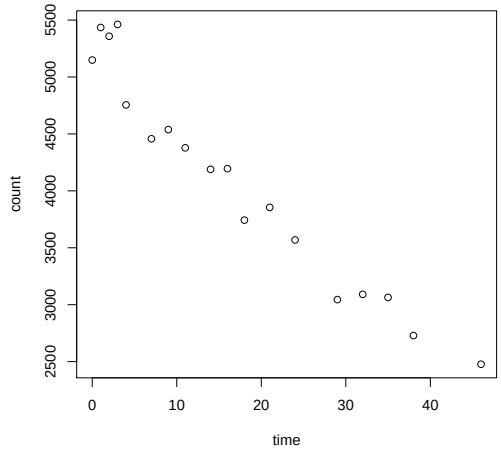

Figure 1: Decay count vs time.

In Figure 1 we see some evidence of a curvature in the data. A common model for radioactive counts is exponential decay. We attempt a NLS fit, $y_i = a + be^{-c*t_i} + \epsilon_i$, where y_i is the radioactive intensity and t_i is the time. a is the background radiation and b an indication of the dose of the substance at onset. c is decay rate parameter.

```
> m1 <- nls(count ~ a + b * exp(-cc * time), data = dataf, start = list(a = 2000,
+     b = 3000, cc = 0.02))
> m1s <- summary(m1)
> print(m1s)
```

```
Formula: count ~ a + b * exp(-cc * time)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
a  8.577e+02  1.108e+03   0.774 0.451089
b  4.532e+03  1.058e+03   4.282 0.000655 ***
cc 2.232e-02  8.323e-03   2.681 0.017092 *
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 185.4 on 15 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 2.762e-07
```

The starting values I picked here were informed. I know that the half-life of the radioactive substance if 40 days. Thus $a + b/2 = a + be^{c*40}$ and thus $c = log(2)/40$. I pick a as close to the minimum observed count and b as the maximum count minus a.
I try out a couple of different starting values and observe that I get convergence to the same final model. (Try this out at home.)

We check the residual diagnostics.

```
> par(mfrow = c(2, 2))
> ma <- m1s$p[1, 1] + m1s$p[2, 1] * exp(-m1s$p[3, 1] * dataf$time)
> plot(dataf$time, dataf$count, xlab = "time", ylab = "count",
+     main = "data")
> lines(dataf$time, ma)
> qqnorm(m1s$res)
> qqline(m1s$res)
> plot(ma, m1s$res, main = "residuals on fitted")
> plot(dataf$time, m1s$res, main = "residuals on time")
```

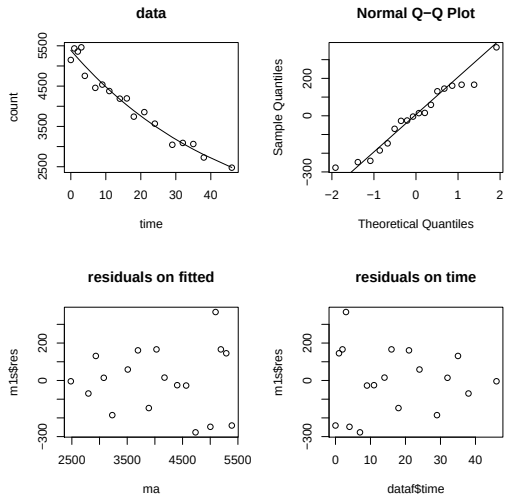

Figure 2: Diagnostics - NLS.

In Figure 2 we examine the fit and the residuals. The model looks like a good fit overall, but with a slight indication that the error variance is non-constant (in fact, count data is usually modeled as Poisson which has a variance linearly depended on the mean).

We can use subset models to test hypotheses of interest, e.g. that the background radiation $a = 0$.

```
> m2 <- nls(count ~ b * exp(-cc * time), data = dataf, start = list(b = 3000,
+     cc = 0.05))
> m2s <- summary(m2)
> print(m2s)

Formula: count ~ b * exp(-cc * time)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
b  5.356e+03  7.801e+01   68.66  < 2e-16 ***
cc 1.734e-02  8.904e-04   19.47 1.45e-12 ***
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 181.7 on 16 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 3.787e-06
```

We use an F-test to compare the two models:

```
> print(anova(m1, m2))

Analysis of Variance Table

Model 1: count ~ a + b * exp(-cc * time)
Model 2: count ~ b * exp(-cc * time)
  Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
1     15     515521
2     16     528080 -1 -12559  0.3654 0.5545
```

The test supports the hypothesis that the background radiation is 0.

Another issue with the original fit is that we didn't restrict the parameters a and b to be positive. We can guaranties this by doing a re-parametrization.

```
> m3 <- nls(count ~ exp(a) + exp(b) * exp(-cc * time), data = dataf,
+     start = list(a = log(2000), b = log(3000), cc = 0.05))
> m3s <- summary(m3)
> print(m3s)

Formula: count ~ exp(a) + exp(b) * exp(-cc * time)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a  6.754238   1.292369   5.226 0.000102 ***
b  8.418909   0.233543  36.049 5.48e-16 ***
cc 0.022316   0.008323   2.681 0.017092 *
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 185.4 on 15 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 6.904e-07
```

The overall fit (RSS) does not change, but this can help with interpretation.

Can we trust the t-tests in the model summary above? We check the profile function of the a, b and c estimates:

```
> pf <- profile(m1, alphamax = 0.05)
> par(mfrow = c(1, 3))
> plot(pf)
```

For the rate parameter, c the profile function indicates that the CIs and t-test based on the linear approximation works well (see Figure 3). For a and b we should be a bit concerned and perhaps use a profile method or try a re-sampling based approach (see upcoming lecture).

```
> print(confint(m1))
```

```
            2.5%        97.5%
a  -9.049410e+03 2.185836e+03
b   3.270914e+03 1.431937e+04
cc  5.075709e-03 4.116787e-02
```

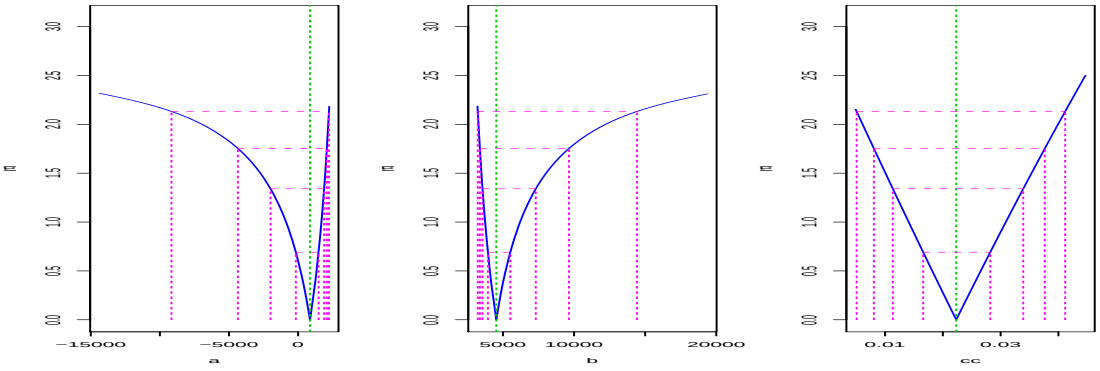

Figure 3: Profile plots.

2.3 IWLS

We are now ready to construct an algorithm to fit a generalized linear model to the data. We will incorporate the weighted least squares to deal with the non-constant variance and an iterative approach to handle the nonlinearity in β.

0 We have $E(y_i) = \mu_i,\ g(\mu_i) = \eta_i = x_i\beta$. We linearize the link function $g(\mu)$ near a points μ^0:
$g(\mu) \simeq g(\mu^0) + (\mu - \mu^0)\frac{dg(\mu)}{d\mu}|_{\mu^0}$
We denote $\eta^0 = g(\mu^0)$ the starting point of the linear predictor, and $\frac{dg(\mu)}{d\mu} = \frac{d\eta}{d\mu}|_{\eta^0}$

1 We denote the initial values $\beta^0 \rightarrow \eta_i^0 = x_i\beta^0 \rightarrow \mu_i^0 = g^{-1}(\eta_i^0)$

2 "Working values" $Z^0 = \eta^0 + (y - \mu^0)\frac{d\eta}{d\mu}|_0$, a linear approximation of $g()$ near η^0 with μ replaced by the data y (so kind of linearizing the data).
The "working weights", $(W^0)^{-1} = \left(\frac{d\eta}{d\mu}|_0\right)^2 V_0$, where $V_0 = b''(\theta_0) = \frac{d\mu}{d\theta}_0$ is the variance of y assuming the current β^0 (known for each distribution family). The working weights are thus inversely proportional to $V(Z^0)$
So the weights that we use are inversely proportional to the observation variance.

3 Update the estimate by regression Z^0 on X using weights W^0.
This gives us $\beta^{new} \rightarrow \eta^{new} = x\beta^{new} \rightarrow$ replace η^0 with this in step [1] and iterate until convergence.

One can show that the IWLS is the same as maximizing the loglikelihood (or a Newton-Rhapson approach to finding the zero-crossing of the derivative of the loglikelihood).

3 Validation and Inference

For linear and nonlinear regression models with additive gaussian errors, the F-test and t-test are our main preliminary investigative tools. Here, the RSS will be replaced with the *residual deviance*, which is just -2 times the maximized loglikelihood. For a model m we denote the corresponding deviance by $D(m)$. The quantity $D(m)/\phi$ is called the scaled deviance (RSS/σ^2 in linear models), and under the normal error assumption is distributed as χ^2_{n-p} (if the model is adequate). This also holds true, asymptotically (large n), in the GLM case.

$$\frac{D(m)}{\phi} \sim \chi^2_{n-p} \text{ if the model } m \text{ is adequate.}$$

Instead of the goodness-of-fit F-test, we have the χ^2-test. If the observed $D(m)/\phi$ exceeds $\chi^2_{n-p}(1-\alpha)$ we reject the model m's adequacy at level α. If this happens, you should try transforming the xs, other link functions, or perhaps switch to another distribution family (e.g. if the Binomial model doesn't work, try a Beta-Binomial).

The dispersion factor ϕ is assumed known in Binomial and Poisson models (equal to 1). It can also be estimated from the data as $\hat{\phi} = D(m)/(n-p)$ (compare $\hat{\sigma^2} = RSS/(n-p)$). If $\hat{\phi}$ is much larger than 1, we say that our data is overdispersed (compared with a standard Binomial or Poisson model.). If this happens we have two options; (1) a quick fix - inflating statistics with the factor $\hat{\phi}$; or (2) pick a model that allows for overdispersion.

3.1 ANODEV - analysis of deviance

The F-test used to compare nested models in linear regression is here replaced with ANODEV (analysis of deviance). Assume we have two models, m_0 and m_1 and m_0 is a simplified version of m_1 (nested). The number of parameters are p_0 and p_1 respectively, and the deviance is $D(m_0)$ and $D(m_1)$. Under the null hypothesis that m_0 is sufficient,

$$\frac{D(m_0) - D(m_1)}{\phi} \sim \chi^2_{p_1-p_0}.$$

We thus reject the model m_0 is the scaled difference of deviances exceeds $\chi^2_{p_1-p_0}(1-\alpha)$.

If we see evidence of overdispersion, or we are using a model where the dispersion factor ϕ is not known or given by the mean parameter, we can estimate $\hat{\phi} = D(m_1)/(n-p_1)$. We can then use an *approximate* F-test (approximate unless the data is normally distributed) where

$$\frac{D(m_0)-D(m_1)}{\hat{\phi}(p_1-p_0)} \sim F_{p_1-p_0,n-p_1} \text{ if } m_0 \text{ is sufficient.}$$

3.2 Subset selection

We can use backward selection based on ANODEV (χ^2 or F-test as above) or AIC or BIC. We can also use AIC and BIC to compare all subset models (see previous lecture).

3.3 z-test and CIs

As in NLS, the final $\hat{\beta}$ comes from a local linearization. The standard package outputs will provide you with standard errors estimated as

$$se(\hat{\beta}_j) = \sqrt{(X'W^{final}X)^{-1}_{jj}}\phi,$$

where W^{final} denotes the final weight function in the IWLS. The z-score is computed as

$$\frac{\hat{\beta}_j}{se(\hat{\beta}_j)}.$$

The $z-scores$ can be very misleading if;
(a) there is overdispersion ($\hat{\phi}$ is much greater than the assumed $\phi = 1$ of Poisson and Binomial models). One fix-it solution is to inflate all the se's by a factor $\sqrt{\hat{\phi}}$ (we still use a z-test after since the t-test came

from the normal error assumption, this fix-it just deals with the se estimate);
(b) as in the NLS case the final local linearization around $\hat{\beta}$ may be a poor representation of the loglikelihood, so the z-based CI may not be accurate. We can check this using profile functions as in NLS: the profile is given by

$$\tau(\beta_j) = \sqrt{\frac{D(\beta_j) - D(\hat{\beta}_j)}{\phi}}.$$

If the profile appears linear in β_j we can use the z-based interval. If not, we can use the approximate profile confidence intervals or use re-sampling based methods like bootstrap.

4 Demo 12

We will revisit the radioactive decay data. We will now analyze two traces, for a healthy and a patient with an illness. The hypothesis is that the disease alters the rate at which blood cells are regenerated and replaced and thus the decay rate for the two traces may differ.

```
> library(stats)
> library(MASS)
> time <- c(0, 1, 2, 3, 4, 7, 9, 11, 14, 16, 18, 21, 24, 29, 32,
+     35, 38, 46)
> count <- c(5149, 5435, 5358, 5462, 4755, 4457, 4538, 4378, 4189,
+     4195, 3743, 3855, 3569, 3045, 3091, 3064, 2729, 2477)
> count2 <- c(5121, 5263, 4930, 5266, 4521, 4434, 4590, 4290, 4005,
+     4123, 3659, 3785, 3538, 3159, 2946, 2956, 2643, 2500)
> dataf <- data.frame(cbind(count, count2, time))
> names(dataf) <- c("count", "count2", "time")
```

We start by reanalyzing trace 1 with GLM instead of NLS.

```
> gg <- glm(count ~ time, data = dataf, family = poisson)
> gs <- summary(gg)
> print(gs)

Call:
glm(formula = count ~ time, family = poisson, data = dataf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1583  -2.2911  -0.0216   2.1166   5.3168

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.584416   0.005528 1552.79   <2e-16 ***
time        -0.017220   0.000290  -59.39   <2e-16 ***
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 3844.9  on 17  degrees of freedom
Residual deviance:  117.1  on 16  degrees of freedom
AIC: 303.35

Number of Fisher Scoring iterations: 3

> plot(dataf$time, dataf$count)
> lines(dataf$time, gg$fit, col = "green")
```

The fit in Figure 4 is almost identical to the NLS fit. That is not so strange since the counts are quite hight and for large mean values a Poisson is well approximated by a normal distribution.

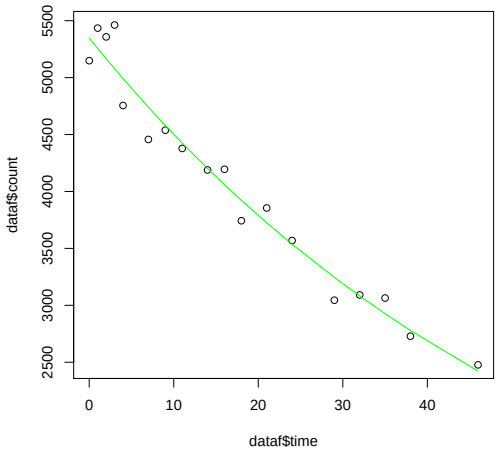

Figure 4: GLM fit to count data

Using the z-test, both coefficients are significant, but remember this may be misleading if the profile is nonlinear, or the scale factor far from 1 (overdispersion).

The goodness of fit test comes next:

```
> c("1-P(X2>Dev)=", 1 - pchisq(gg$dev, df = gs$df[2]))

[1] "1-P(X2>Dev)=" "0"
```

The Poisson model is rejected. There is evidence of overdispersion.

```
> par(mfrow = c(2, 2))
> plot(gg)
```

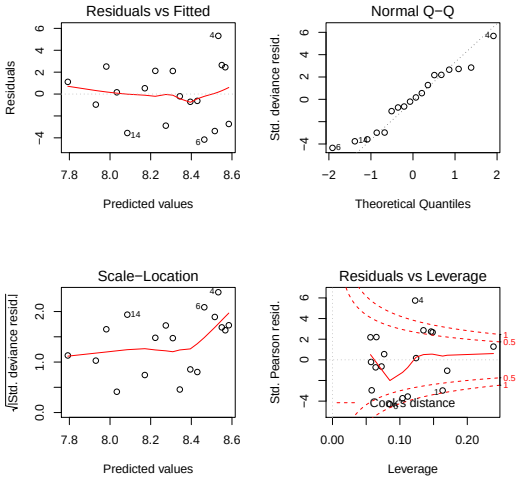

Figure 5: Diagnostics of the Poisson model

For now, let us continue without addressing this particular issue:

```
> c("Z-interval")
```

```
[1] "Z-interval"
```

```
> c("(Intercept)", round(gs$coef[1, 1] - 1.96 * gs$coef[1, 2],
+     8), round(gs$coef[1, 1] + 1.96 * gs$coef[1, 2], 8))
```

```
[1] "(Intercept)" "8.57358047"  "8.5952517"
```

```
> c("(time)", round(gs$coef[2, 1] - 1.96 * gs$coef[2, 2], 8), round(gs$coef[2,
+     1] + 1.96 * gs$coef[2, 2], 8))
```

```
[1] "(time)"      "-0.01778797" "-0.01665133"
```

We are concerned that the z-interval are misleading, not only because of the overdispersion but also because of the linear approximation in the last iteration not being very accurate. We can use the profile function to examine this.

```
> plot(profile(gg))
```

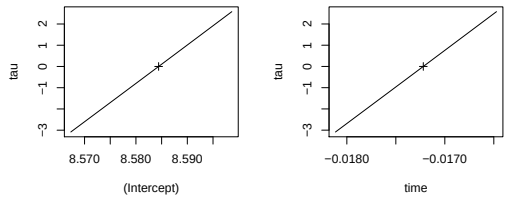

Figure 6: Profile functions for the Poisson fit

The `confint` function in R uses the profile to construct asymmetric CIs for the parameter estimates. As the profiles here indicate a linear approximation is adequate, the two confidence intervals almost completely agree.

```
> print(confint(gg))
```

```
                 2.5 %      97.5 %
(Intercept)  8.57356662  8.59523749
time        -0.01778852 -0.01665189
```

The estimated dispersion factor is estimated as

```
> c("phi-hat=", round(sum(residuals(gg, "pearson")^2)/gg$df.r,
+     5))
```

```
[1] "phi-hat=" "7.32128"
```

We now analyze the two traces jointly:

```
> A <- as.factor(c(rep(0, length(time)), rep(1, length(time))))
> Time <- rep(time, 2)
> Count <- c(count, count2)
> Dataf <- data.frame(cbind(Count, Time, A))
> names(Dataf) <- c("Count", "Time", "A")
> gg2 <- glm(Count ~ A * Time, data = Dataf, family = poisson)
> gs2 <- summary(gg2)
> par(mfrow = c(1, 1))
> plot(Time, Count)
> points(time, count, col = "red")
> points(time, count2, col = "green")
> lines(Time[A == 0], gg2$fit[A == 0], col = "red")
> lines(Time[A == 1], gg2$fit[A == 1], col = "green")
```

```
> print(gs2)
```

```
Call:
glm(formula = Count ~ A * Time, family = poisson, data = Dataf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6690  -1.8172  -0.2573   2.1118   5.3168

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.6172256  0.0123974 695.082  < 2e-16 ***
A           -0.0328095  0.0078745  -4.167 3.09e-05 ***
Time        -0.0178210  0.0006496 -27.436  < 2e-16 ***
A:Time       0.0006013  0.0004119   1.460    0.144
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 7366.39  on 35  degrees of freedom
Residual deviance:  215.75  on 32  degrees of freedom
AIC: 587.86

Number of Fisher Scoring iterations: 3
```

The patient*time interaction is not significant based on the z-test, and if anything, the z-test is too generous since the data is overdispersed. The interaction is unlikely to be needed. Also, notice that the curves are almost parallel (Figure 7).

The dispersion in the joint model is estimated as

```
> c("Dev/df=", round(gg2$dev/gs2$df[2], 5))
```

```
[1] "Dev/df=" "6.74225"
```

which is much greater than 1.

We ignore this for now, and construct profile intervals (which assumes $\phi = 1$ since the likelihood assumes this).

```
> print(confint(gg2))
```

```
                   2.5 %       97.5 %
(Intercept)  8.5929076634  8.641504762
A           -0.0482436118 -0.017376192
Time        -0.0190948693 -0.016548656
A:Time      -0.0002060413  0.001408709
```

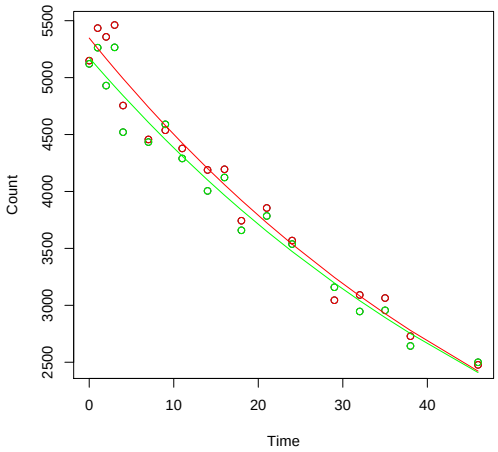

Figure 7: Poisson - two traces

The confidence intervals indicate that we can drop the patient*time interaction. There is no change in rate of decay, only an offset in the onset radioactive level.

```
> gg2b <- update(gg2, . ~ . - A:Time)
> anova(gg2b, gg2)

Analysis of Deviance Table

Model 1: Count ~ A + Time
Model 2: Count ~ A * Time
  Resid. Df Resid. Dev Df Deviance
1        33     217.88
2        32     215.75  1    2.131
```

We update the model, dropping the interaction. Using ANODEV, this test compares the increase in deviance to a χ^2 distribution on 1 degrees of freedom.

```
> c("Pvalue", round((1 - pchisq(2.131, 1)), 6))

[1] "Pvalue"   "0.144347"
```

The outcome of the test is that we can drop the interaction.

Can we drop the patient label alltogether?

```
> gg2c <- update(gg2, . ~ . - A - A:Time)
> anova(gg2c, gg2)

Analysis of Deviance Table

Model 1: Count ~ Time
Model 2: Count ~ A * Time
  Resid. Df Resid. Dev Df Deviance
1        34     239.21
2        32     215.75  2   23.462

> c("Pvalue", round((1 - pchisq(23.462, 1)), 6))

[1] "Pvalue" "1e-06"
```

No, the ANODEV does not support dropping the patient factor.

What happens when we take overdispersion into account? We use the approximate F-test:

```
> FF <- (gg2c$dev - gg2$dev)/(2 * (gg2$dev/gs2$df[2]))
> c("F-statistic=", round(FF, 6))

[1] "F-statistic=" "1.739939"

> c("Pvalue", round((1 - pf(FF, 2, gs2$df[2])), 6))

[1] "Pvalue"   "0.191728"
```

This test does support dropping the patient factor completely. The z-test is clearly too generous since we have overdispersion, but on the other hand the F-test is an ad-hoc approximation.

We explore using a different model, the negative binomial model, which allows for count data to have excess variance compared with a Poisson (variance proportional to μ^2).

```
> gg3 <- glm.nb(Count ~ A * Time, data = Dataf)
> gs3 <- summary(gg3)
> print(gs3)

Call:
glm.nb(formula = Count ~ A * Time, data = Dataf, init.theta = 848.7122187,
    link = log)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.80537  -0.86712  -0.09973   0.89403   2.02243

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.6151856  0.0316410 272.279   <2e-16 ***
A           -0.0321781  0.0200241  -1.607    0.108
Time        -0.0176904  0.0014757 -11.988   <2e-16 ***
A:Time       0.0005607  0.0009338   0.600    0.548
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

(Dispersion parameter for Negative Binomial(848.7122) family taken to be 1)

    Null deviance: 1320.445  on 35  degrees of freedom
Residual deviance:   35.575  on 32  degrees of freedom
AIC: 472.04

Number of Fisher Scoring iterations: 1


              Theta:  849
          Std. Err.:  241

 2 x log-likelihood:  -462.036
```

If we use the negative binomial model to construct confidence intervals we get the following results:

```
> print(confint(gg3))

                 2.5 %       97.5 %
(Intercept)  8.55335429  8.677201728
A           -0.07137420  0.007017876
Time        -0.02057498 -0.014803658
A:Time      -0.00126578  0.002387179
```

This model also suggests that the patient factor can be dropped from the model.

```
> gg3c <- update(gg3, . ~ . - A - A:Time)
> anova(gg3, gg3c)

Likelihood ratio tests of Negative Binomial Models

Response: Count
     Model    theta Resid. df    2 x log-lik.   Test    df LR stat.   Pr(Chi)
1     Time 755.8674        34       -465.4930
2 A * Time 848.7122        32       -462.0363 1 vs 2     2 3.456692 0.1775779
```

The ANODEV based on the negative binomial agrees with the findings so far.

What if we use backward AIC selection for the Poisson model?

```
> step(gg2)

Start:  AIC=587.86
Count ~ A * Time

         Df Deviance    AIC
<none>        215.75 587.86
- A:Time  1   217.88 587.99

Call:  glm(formula = Count ~ A * Time, family = poisson, data = Dataf)

Coefficients:
(Intercept)            A         Time       A:Time
  8.6172256   -0.0328095   -0.0178210    0.0006013

Degrees of Freedom: 35 Total (i.e. Null);  32 Residual
Null Deviance:            7366
Residual Deviance: 215.8         AIC: 587.9
```

This doesn't simplify the model at all. The AIC is less conservative than the Poisson ANODEV.

```
> step(gg3)

Start:  AIC=470.04
Count ~ A * Time

         Df Deviance    AIC
- A:Time  1   35.937 468.40
<none>        35.575 470.04

Step:  AIC=468.4
Count ~ A + Time

       Df Deviance     AIC
<none>       35.58  468.40
- A     1    38.82  469.63
- Time  1  1303.99 1734.80

Call:  glm.nb(formula = Count ~ A + Time, data = Dataf, init.theta = 838.6941628,
    link = log)

Coefficients:
(Intercept)            A         Time
    8.60121     -0.02285     -0.01685
```

```
Degrees of Freedom: 35 Total (i.e. Null);  33 Residual
Null Deviance:            1308
Residual Deviance: 35.58         AIC: 470.4
```

The backward selection using AIC and the negative binomial retains the patient factor as a main effect, but it is a close call. The conclusion we can draw from this is that it is very important to check the approximations and assumptions, especially overdispersion.

Let us revisit the heart disease data:

```
> SA <- data.frame(read.table("SA.dat", sep = "\t", header = T))
> SA$ldl <- log(SA$ldl)
> SA$age <- log(SA$age)
> SA$famhist <- SA$famhist - 1
> gg <- glm(chd ~ ldl + age + sbp + alcohol + alcind + tobacco +
+     tobind + famhist + typea + adiposity + obesity, "binomial",
+     data = SA)
> print(summary(gg))

Call:
glm(formula = chd ~ ldl + age + sbp + alcohol + alcind + tobacco +
    tobind + famhist + typea + adiposity + obesity, family = "binomial",
    data = SA)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8419  -0.7773  -0.4187   0.8838   2.9126

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -12.145033   2.687598  -4.519 6.22e-06 ***
ldl           1.357091   0.419875   3.232  0.00123 **
age           1.826944   0.618120   2.956  0.00312 **
sbp           0.008857   0.006660   1.330  0.18358
alcohol       0.006813   0.006553   1.040  0.29851
alcind       -0.296096   0.362703  -0.816  0.41429
tobacco       0.075608   0.035945   2.103  0.03543 *
tobind        0.490360   0.435641   1.126  0.26033
famhist       0.762985   0.284905   2.678  0.00741 **
typea         0.039715   0.015028   2.643  0.00822 **
adiposity    -0.006932   0.037065  -0.187  0.85165
obesity      -0.057064   0.054408  -1.049  0.29426
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 401.21  on 311  degrees of freedom
Residual deviance: 308.57  on 300  degrees of freedom
AIC: 332.57

Number of Fisher Scoring iterations: 5
```

The dispersion factor for the heart disease data is

```
> sum(residuals(gg, "pearson")^2/gg$df.r)

[1] 1.163297
```

which is not too far from 1. It does not look as if the heart disease data is overdispersed. Can we trust the z-based intervals? We check the profiles of the model.

```
> plot(profile(gg))
```

The profile plots (Figure 8) look very good. We can conclude that the z-based tests and confidence intervals are appropriate.

```
> confint(gg)
```

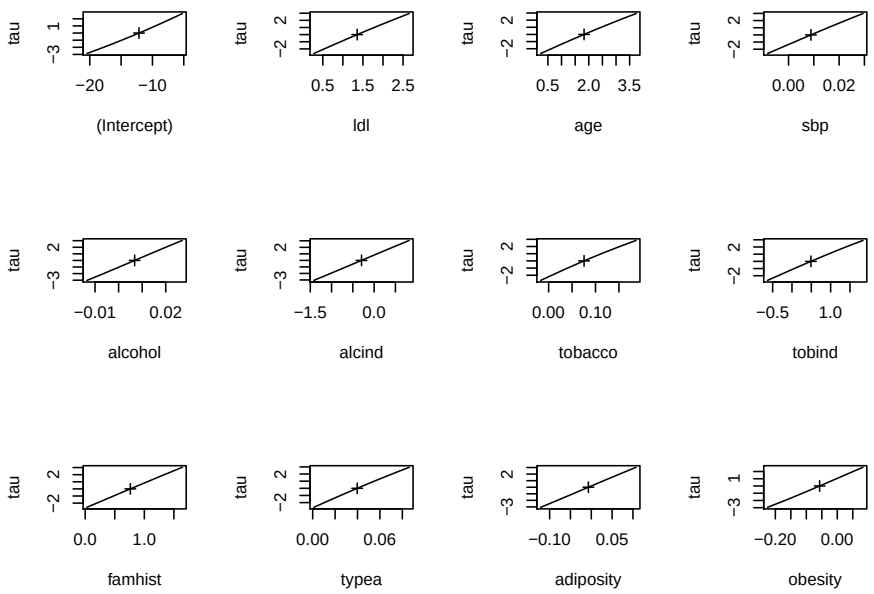

Figure 8: Profile plots - heart disease

```
                  2.5 %      97.5 %
(Intercept) -17.675838891 -7.10763982
ldl           0.552234715  2.20404982
age           0.649943149  3.08246218
sbp          -0.004070564  0.02218714
alcohol      -0.006136452  0.01973955
alcind       -1.008749380  0.41775682
tobacco       0.008139853  0.15009617
tobind       -0.343916831  1.37601499
famhist       0.207476326  1.32697467
typea         0.010983782  0.07010697
adiposity    -0.079821827  0.06637729
obesity      -0.166818265  0.04902196
```

We will now try all subset selection. The program `randomsplitsBin.R` performs random splits over a fraction of the data and compute prediction errors. We start by applying this to one random split only.

```
> frac <- 2/3
> ii <- sample(seq(1, dim(SA)[1]), round(dim(SA)[1] * frac))
> yy <- SA[ii, 10]
> xx <- as.matrix(SA[ii, -10])
> yyt <- SA[-ii, 10]
> xxt <- as.matrix(SA[-ii, -10])
> library(leaps)
> rleaps <- regsubsets(xx, yy, int = T, nbest = 100, nvmax = dim(SA)[2],
+     really.big = T, method = c("ex"))
```

```
> cleaps <- summary(rleaps, matrix = T)
> tt <- apply(cleaps$which, 1, sum)
> BICvec <- rep(0, dim(cleaps$which)[1])
> AICvec <- BICvec
> y <- SA[, 10]
> x <- as.matrix(SA[, -10])
> data1 <- as.data.frame(cbind(y, x))
> for (zz in (1:dim(cleaps$which)[1])) {
+     gg <- glm(y ~ x[, cleaps$which[zz, 2:dim(cleaps$which)[2]] ==
+         T], "binomial", subset = ii, data = data1)
+     AICvec[zz] <- gg$dev + 2 * tt[zz]
+     BICvec[zz] <- gg$dev + tt[zz] * log(length(yy))
+ }

> plot(tt, AICvec, xlab = "modelsize", ylab = "AIC")

> plot(tt, BICvec, xlab = "modelsize", ylab = "BIC")
```

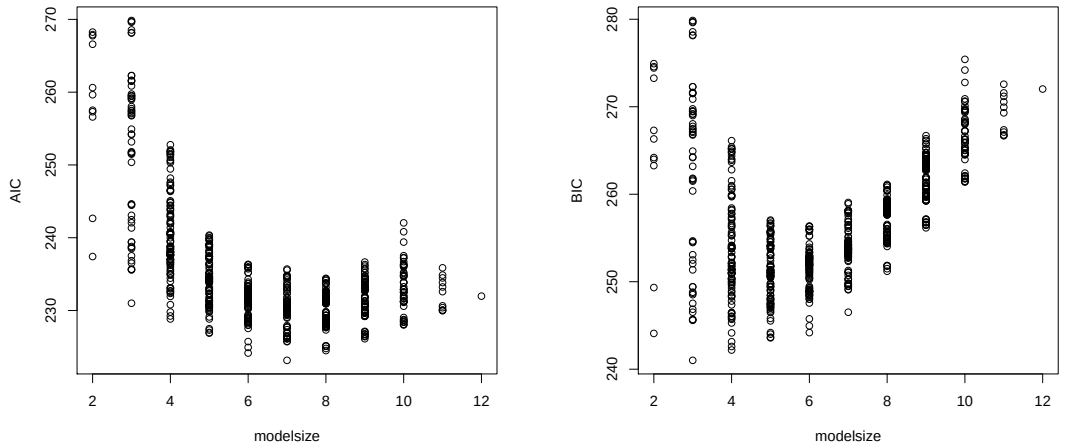

Figure 9: AIC and BIC

In Figure 10 we compare AIC and BIC values for different subset models. The winning models are obtained as

```
> print(aicmod <- cleaps$which[AICvec == min(AICvec), ])

(Intercept)         age         sbp   adiposity     obesity       typea
       TRUE        TRUE       FALSE       FALSE        TRUE        TRUE
    alcohol      alcind     tobacco      tobind     famhist         ldl
      FALSE       FALSE        TRUE       FALSE        TRUE        TRUE

> print(bicmod <- cleaps$which[BICvec == min(BICvec), ])

(Intercept)         age         sbp   adiposity     obesity       typea
       TRUE        TRUE       FALSE       FALSE       FALSE       FALSE
    alcohol      alcind     tobacco      tobind     famhist         ldl
      FALSE       FALSE        TRUE       FALSE       FALSE       FALSE
```

We try using the selected models for prediction on the test data.

```
> mmaic <- glm(y ~ x[, aicmod[2:length(aicmod)] == T], "binomial",
+     subset = ii)
```

```
> mmbic <- glm(y ~ x[, bicmod[2:length(bicmod)] == T], "binomial",
+     subset = ii)
> ppaic <- predict(mmaic, as.data.frame(x[-ii, ]), "response")
> ppbic <- predict(mmbic, as.data.frame(x[-ii, ]), "response")
> print(PEaicK <- sum(yyt != round(ppaic[-ii]))/length(yyt))

[1] 0.2596154

> print(PEbicK <- sum(yyt != round(ppbic[-ii]))/length(yyt))

[1] 0.3365385

> source("randomsplitsBin.R")
> rr <- randomsplitsBin(SA, 10, 2/3, 100)

> print(rr$modseltab)

                modselaic modselbic
 [1,] "age"       "100"     "97"
 [2,] "sbp"       "28"      "2"
 [3,] "adiposity" "13"      "4"
 [4,] "obesity"   "36"      "5"
 [5,] "typea"     "92"      "49"
 [6,] "alcohol"   "16"      "1"
 [7,] "alcind"    "5"       "0"
 [8,] "tobacco"   "80"      "34"
 [9,] "tobind"    "29"      "14"
[10,] "famhist"   "91"      "42"
[11,] "ldl"       "96"      "72"

> boxplot(cbind(rr$PEa, rr$PEb), names = c("AIC", "BIC"), main = "Prediction errors")

> boxplot(cbind(rr$modsizea, rr$modsizeb), names = c("AIC", "BIC"),
+     main = "Modelsizes")
```

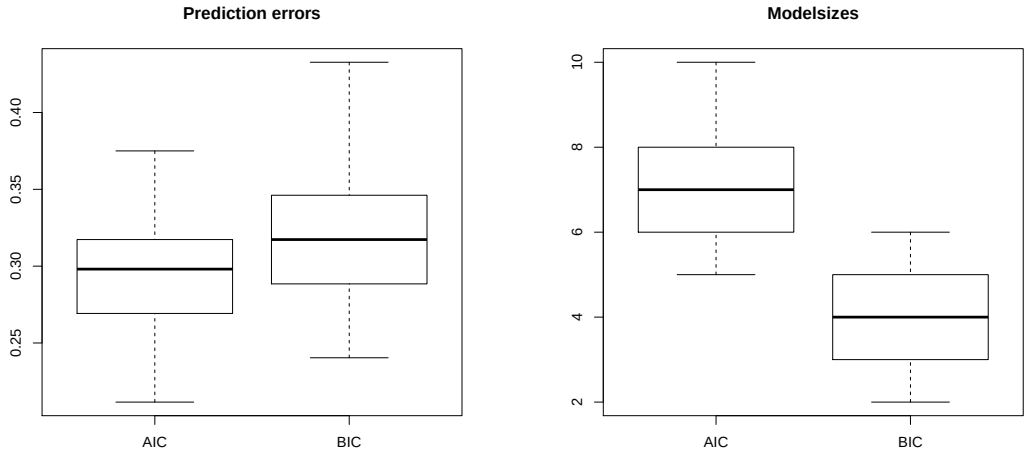

Figure 10: Prediction errors and modelsizes