STAT0030 Lab 9: Generalized Linear Modelling in R

In the last workshop we looked at numerical methods for fitting generalized linear models (GLMs). R has an extensive range of facilities for fitting GLMs and related models; in this workshop we will explore these.

1 The `glm` command

As you might expect, the specification of GLMs using `glm` is very similar to that of linear models using `lm` (see Lab 4). The main difference is the need to specify a probability distribution (Poisson, normal, gamma etc.) for the response, and a form for the link function. This is done using the `family` argument. For example:

```
> gamma.glm <- glm(y ~ x1 + x2, family=Gamma, data=mydataframe)
```

If, as here, you don't specify a link, R will use the canonical link for the chosen family of distributions (see Lab 8). The possible values of `family`, together with associated canonical links, are summarised in Table 1. Note that the canonical links are defined primarily on mathematical grounds, and are not necessarily the most suitable for applications. For example, for the gamma distribution the canonical link is the inverse: $g(\mu) = \mu^{-1}$. When using the gamma distribution, it is often more sensible to use a log link: $g(\mu) = \ln \mu$. This has to be stated explicitly:

```
> gamma.glm <- glm(y ~ x1 + x2, family=Gamma(link="log"),
+   data=mydataframe)
```

In general, it is good practice to state your link functions explicitly — then you can be sure that R hasn't made any silly decisions for you.

R commands such as `summary`, `predict`, `plot` and `resid` work in the same way for `glm` objects as for linear model (`lm`) objects. To explore these, we will start with an example.

2 Poisson regression: effect of El Niño on tropical storms

In the last workshop, we illustrated the IWLS algorithm by fitting a Poisson distribution to Pacific tropical storm counts. We will now do something a bit more interesting, which is to relate storm activity to the El Niño phenomenon. By way of background: 'El Niño' is the name given to the periodic warming of large areas of the tropical Pacific ocean (the intervening cool periods are referred to as 'La Niña' episodes). In climatology, it is well known that the state of El Niño in one year is related to tropical storm activity in the North-West Pacific the following year. This relationship can be used, for example, to try to forecast tropical storm activity.

For the last workshop, you downloaded file `nstorms.dat` which contained annual storm counts. For this workshop, you will also need file `nino3.dat`, which is available from the usual place. The file contains monthly values of an index of El Niño activity, from 1856 to 2000. The file contains 13 columns: the first is `Year`, and the remainder are the monthly index values. Download the file to your working directory. Start up R and change to this directory. Reload the Pacific storm count data from Lab 8 by reading the file `nstorms.dat` using `read.table` with the argument `header=TRUE`. Assign it to an object called `storm.data`. Now you're ready to start.

Distribution	**Value of `family`**	**Canonical link**
Binomial (or Bernoulli)	`binomial`	`logit`: $g(\mu) = \ln[\mu/(1-\mu)]$
Normal	`gaussian`	`identity`: $g(\mu) = \mu$
Gamma	`Gamma`	`inverse`: $g(\mu) = \mu^{-1}$
Inverse Gaussian	`inverse.gaussian`	`1/mu^2`: $g(\mu) = \mu^{-2}$
Poisson	`poisson`	`log`: $g(\mu) = \ln \mu$
Overdispersed Binomial	`quasibinomial`	`logit`: $g(\mu) = \ln[\mu/(1-\mu)]$
Overdispersed Poisson	`quasipoisson`	`log`: $g(\mu) = \ln \mu$
See Section 4	`quasi`	See Section 4

Table 1: Distributions supported by the `family` argument to the `glm` command, with associated canonical (i.e. default) link functions.

1. Use `read.table` (again, with `header=TRUE`) to read the data in `nino3.dat` into an object called `nino.data`.

2. You now have two separate data frames, `storm.data` and `nino.data`. It would be useful to merge these into a single data frame, with rows corresponding to cases. We haven't done much data manipulation since Lab 3, so here are the commands you need:

```
> nino.data$Year <- nino.data$Year + 1
> storm.data <- merge(storm.data,nino.data)
```

   When R merges the frames, it notices that they both have a column called `Year`, so it matches cases accordingly. By adding 1 to the years in `nino.data` frame before merging, we match the storm data to the *previous* year's El Niño values. This is necessary if we want to build a model that's useful for forecasting.

3. A full analysis of these data[1] reveals that storm counts are more strongly associated with September El Niño values than with those for any other month. To visualise the relationship, produce a scatterplot of `Storms` against `N3.m09` ('Niño 3 index, month 9').

4. On the strength of this, let's fit a GLM that uses September El Niño activity to predict tropical storm numbers. The Poisson is a natural first choice of distribution for count data, and a log link (which happens to be canonical in the Poisson case) is often sensible (for one thing, it guarantees that all fitted values will be positive):

```
> storm.model1 <-
+    glm(Storms ~ N3.m09,family=poisson(link="log"),data=storm.data)
> summary(storm.model1)
```

[1] See Richard Chandler's homepage at `http://www.homepages.ucl.ac.uk/~ucakarc/work/glmnotes/study1/study1.html` for more details if you're interested.

The `summary` is very similar to that for linear models — in particular, you get a table of coefficient estimates, test statistics and p-values. In addition, you get:

- A summary of the deviance residuals. We will explore these further in Section 2.1 below.
- A note about the dispersion parameter (ϕ, in the notation of Lab 8). For the Poisson distribution, $\phi = 1$.
- The null and residual deviance. The residual deviance is the quantity D introduced in Section 4.5 of Lab 8. The null deviance is the deviance for the model containing only an intercept term.
- Number of Fisher scoring iterations. This tells how many IWLS iterations were required to obtain the parameter estimates.
- The AIC (Akaike Information Criterion) for this model. We discussed this in Lab 4, in the context of linear model selection. To recap briefly: AIC is intended to provide an automatic means of model selection. It is defined as $-2 \ln L + 2k$, where $\ln L$ is the log-likelihood for a model and k is the number of parameters estimated. The object, if you're going to use AIC to choose models, is to find the model with the minimum AIC. Be aware, however, that 'automatic' model selection procedures are best avoided if possible — there is no substitute for intelligent thought!

By typing `summary(storm.model1,correlation=TRUE)` you can also see the correlation between the estimated coefficients. This is calculated from the inverse of the Fisher information matrix.

2.1 Residuals, fitted values and diagnostics

Having fitted a GLM, we will want to check it; also possibly to extract the fitted values for each case in the dataset, and to predict future values of the response. In the standard linear model, model checking is often based on residuals. For GLMs however, there is no unique definition of a 'residual', so you need to be a bit careful. Compare the output of

```
> storm.model1$residuals
```

with that of

```
> resid(storm.model1)
```

to see why! To understand what's going on, you need to know about the following types of residual (most of which coincide when the response distribution is normal):

Response residuals: these are the 'raw residuals' obtained by subtracting the fitted values from the observations. To verify this:

```
> resid(storm.model1,type="response")
> mu <- fitted(storm.model1)
> storm.data$Storms - mu
```

The `fitted` command returns the means $\{\mu_i : i = 1, \ldots, n\}$ according to the fitted model.

In general, response residuals are not very useful since they all come from distributions with different variances — hence it is difficult to make any meaningful comparisons among them.

Pearson residuals: A simple way to make the response residuals comparable is to divide each one by its modelled standard deviation — if the model is correct, the resulting residuals should all come from distributions with mean zero and variance 1. These are *Pearson residuals*. They are perhaps the most easily-understood of the 'useful' residual measures. For the Poisson distribution, the variance is equal to the mean so the Pearson residuals for the ith case is equal to $(y_i - \mu_i) / \sqrt{\mu_i}$. Check this:

```
> (storm.data$Storms-mu)/sqrt(mu)
> resid(storm.model1,type="pearson")
```

Alternatively,

```
> all.equal((storm.data$Storms-mu)/sqrt(mu),
+   resid(storm.model1,type="pearson"))
```

Since the Pearson residuals have variance 1 under the Poisson assumption, you should be able to use them to check it. **Exercise:** carry out this check. What do you think?

Deviance residuals: you have seen that the deviance for a GLM can be regarded as the equivalent of an error sum of squares in a linear model. One could argue that since the deviance is calculated from log-likelihoods, each of which is a sum of contributions from each case in your dataset, these contributions are in some sense equivalent to the squared residuals from a linear model. Carrying the analogy to its (not particularly helpful, in my view) conclusion, we could therefore take the square root of each of these deviance contributions, attach a + or - sign as appropriate, and claim that the resulting *deviance residuals* are the GLM equivalents of standard residuals from a linear model. The `resid` command returns deviance residuals by default. To check this:

```
> sum(resid(storm.model1)^2)
> storm.model1$deviance
```

You could be excused for thinking that the idea of deviance residuals is rather artificial. The following arguments are sometimes advocated in support of them:

- The deviance is a measure, in likelihood terms, of lack of fit (it's essentially the difference in log-likelihoods between the model you've got and a model that fits the data perfectly). Deviance residuals give an idea of the extent to which individual observations contribute to this lack of fit. Therefore, by looking at cases with large deviance residuals, you may be able to work out how to improve your model.
- For large samples, the deviance has an approximate χ^2 distribution. Since a χ^2 is a sum of squared standard normals, the deviance residuals might therefore be expected to look something like standard normals. In my view this is an *extremely* tenuous argument; however, it's used by R in its diagnostic plots for GLMs, so you need to know about it.

Working residuals: In the IWLS algorithm, at each stage of the iterative procedure you calculate the adjusted dependent variates $z_i = \eta_i + (y_i - \mu_i)\, g'(\mu_i)$ (Lab 8, end of Section 2). $(y_i - \mu_i)\, g'(\mu_i)$ is some sort of residual. In R, the `residuals` component of a `glm` object contains these quantities, which are called *working residuals*. This explains why `storm.model1$residuals` gave you a different result from `resid(storm.model1)`. The working residuals have no use in any statistical analysis, and therefore **you should ALWAYS use the `resid` command to extract residuals from a model object**.

We are now in a position to interpret the output of the `plot` command, when applied to a `glm` object:

```
> par(mfrow=c(2,2))
> plot(storm.model1)
```

This produces four plots, which are very similar to the ones you get when you `plot` a `lm` object; see also Lab 4 Section 2.3. The plots are as follows:

1. A plot of deviance residuals against linear predictors (i.e. ηs, in the notation of Lab 8). **NB** the x-axis is labelled *Predicted values*, which may lead you to think that the residuals are being plotted against μs (especially since the plot title is 'Residuals vs Fitted'!). For our storm model, we used a log link so the linear predictor is the log of the mean. To check this (and to check that the x-axis represents linear predictors rather than fitted values):

   ```
   > predict(storm.model1)
   > log(fitted(storm.model1))
   ```

   If your model is a good one, the residuals should be randomly scattered about zero, with constant variance.

2. A normal probability plot of standardised deviance residuals. This is based on the assumption that they should behave approximately like standard normal random variables. For this particular example, the approximation seems to work reasonably well. However, it can be absolutely dreadful if, for example, you have Poisson data with a small mean so that your observed counts are all 0, 1 or 2. Therefore you shouldn't worry too much about this plot if your data are highly discrete. In this particular example the plot seems to indicate that the Poisson model fits the data well, although you may have doubts about this based on (a) your analysis of the Pearson residuals (b) the plausibility of the Poisson process as a model for cyclone formation.

3. A scale-location plot, showing transformed residuals against linear predictors. Here, if the variance structure of the model is correct the points should appear to be randomly scattered about an average value of 1.

4. A plot of standardised Pearson residuals against leverage which, as with linear models, is designed to highlight both outliers and influential observations (see your notes for Lab 4 and *STAT0028* or *STAT0032*).

2.2 Predicting future values

The `predict` command above was used to extract the linear predictors from the fitted model. This command can also be used to calculate linear predictors, along with corresponding confidence intervals if necessary, for subsequent cases. For example, the value of the El Niño index in September 2000 was -0.226. We can use our model to forecast a distribution for the number of NW Pacific storms in 2001:

```
> newnino.data <- data.frame(N3.m09=-0.226)
> predict(storm.model1,newdata=newnino.data,se.fit=TRUE)
```

With `se.fit=TRUE`, the `predict` command returns the standard error associated with the calculated linear predictor. This is derived from the fact that the predictor itself can be written as $\hat{\eta} = \mathbf{x}^T\hat{\boldsymbol{\beta}}$ where $\mathbf{x}$ is the new vector of covariates. The variance, and hence standard error, of $\hat{\eta}$ can therefore be calculated from the covariance matrix of $\hat{\boldsymbol{\beta}}$.

Exercise: An approximate 95% confidence interval for η can be calculated as $\hat{\eta} \pm (1.96 \times \text{s.e.}(\hat{\eta}))$. Calculate this interval, and hence the corresponding interval for the mean of the 2001 storm count distribution.

2.3 Model comparison

Statistical models are often built with the aim of determining whether a particular covariate has a 'genuine' effect on the response. This can be done using t-tests for the regression coefficients in the model. However, if covariates are highly correlated this can yield misleading inferences unless sample sizes are large (recall the 'sheep energy requirement' example in Lab 4). Correlated covariates arise quite frequently in applications where experimental design is not possible. In such situations, the effect of a particular covariate can be assessed using deviance tests, as described in Lab 8. These tests should be based on the χ^2 distribution if the models' dispersion parameter is known, and on the F distribution otherwise. The procedure — known as 'analysis of deviance' — is identical to the analysis of variance for linear models, and hence you shouldn't be surprised to discover that we use the `anova` command:

```
> anova(storm.model1,test="Chi")
```

This tests sequentially whether the reduction in deviance attributable to each predictor in the model is significantly different from zero. We use a χ^2 test because the dispersion parameter is known for the Poisson case — otherwise we would have used `test="F"`.

The `anova` command can also be used to compare two nested models. For example, can we improve our model by adding the October El Niño?

```
> storm.model2 <- glm(Storms ~ N3.m09 + N3.m10, family=poisson(link="log"),
+     data=storm.data)
> summary(storm.model2)
> anova(storm.model1,storm.model2,test="Chi")
> anova(storm.model2,test="Chi")
```

Notice the different conclusions from the t-tests reported by `summary` and the deviance tests reported by `anova`. The deviance-based results are more reliable.

Other commands that help you to choose models — `drop`, `step` and so on — are available for `glm` objects as with `lm` objects. If you've forgotten what they do, consult your notes for Lab 4.

Dose ($\log_{10}$ CS_2 mg l^{-1})	Number of insects	Number of deaths
1.6907	59	6
1.7242	60	13
1.7552	62	18
1.7842	56	28
1.8113	63	52
1.8369	59	53
1.8610	62	61
1.8839	60	60

Table 2: Beetle mortality data (from A. Dobson: *An Introduction to Generalized Linear Models*)

3 Models for binary data

Suppose $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_N)^T$ is a vector of binomial random variables: $Y_i \sim Bin(n_i, \pi_i)$. This situation might arise when we are studying the effects of covariates upon the presence or absence of some attribute of interest and, within each of N groups of individuals, we record the number of individuals possessing this attribute. A GLM for $\mathbf{Y}$ can be expressed as

$$g(\pi_i) = \eta_i = \mathbf{x}_i^T \boldsymbol{\beta} \ ,$$

for an appropriate link function $g(.)$. The most common links for the binomial distribution are the logit (which is the canonical link — see Table 1), probit ($g(\mu) = \Phi^{-1}(\mu)$, where $\Phi(.)$ is the standard normal distribution function) and complementary log-log: $g(\mu) = \ln[-\ln(1-\mu)]$. A GLM for binomial data with a logit link is called a *logistic regression model*.

To fit GLMs for binary data in R, you need a vector of *proportions* (rather than counts) as a response; you also need a vector of group sizes (the ns in the binomial distributions), unless these are all equal to 1. Suppose your counts are stored in a vector `y`, and the group sizes in a vector `n`. Then a logistic regression model can be fitted using a command like

```
> my.glm <- glm(y/n ~ x1 + x2,family=binomial,weights=n,
+   data=mydataframe)
```

The argument `weights=n` *must* be included unless each subgroup is of size 1. For alternative link functions, use `family=binomial(link="probit")` or `family=binomial(link="cloglog")`.

3.1 Example: dose-response Curves

We will consider an example taken from Dobson (textbook for your G1 course). Table 2 shows numbers of insects dead after five hours' exposure to gaseous carbon disulphide at various concentrations. Create three vectors in R to hold these data: I suggest you call them `dose`, `totals` and `dead`. **NB** these data are *not* on the website — you have to create them yourself.

1. Produce a plot showing the proportion of deaths against the dose. Since the y axis is a proportion, ensure that the limits on this axis are set to 0 and 1.

2. Produce a plot of `log(dead/(totals-dead))` vs. `dose`. Does this tell you anything about the way in which the dose affects the proportion of deaths?

3. Carry out a logistic regression for the proportion of deaths as a function of dose. Store the fitted model in an object called `beetle.glm`. Obtain a `summary` and `plot` of the model.

4. Use the `plot(beetle.glm)` command to assess the fit of the model. Before executing this command, type `par(mfrow=c(2,2))` to put all four plots on one window.

   Do you see a pattern in the 'Residuals vs Fitted' plot? Try plotting the observed values against the fits (after first telling R to put one graph on the page again):

```
> par(mfrow=c(1,1))
> plot(fitted(beetle.glm),dead/totals,pch=15,xlim=c(0,1),ylim=c(0,1),
+      xlab="Fitted proportions",ylab="Observed proportions")
> abline(0,1,lty=2)
> title("Observed vs fitted proportions for beetle data")
```

   Does that help?

5. Finally, plot the fitted response curve over the data:

```
> plot(dose,dead/totals,pch=15,xlab="Dose",
+      ylab="Proportion killed",ylim=c(0,1))
> lines(dose,fitted(beetle.glm),type="l",lty=2)
```

6. Hopefully, you have concluded by now that the logistic model underpredicts at high and low doses, and overpredicts in the middle. This suggests that the logit is not the correct link function for this dataset. Perhaps the `probit` or `cloglog` link will provide a better fit. Fit the corresponding models, and examine their deviances. Which model fits best, in terms of deviance? Is this conclusion supported by residual analyses?

4 Quasi-likelihood fitting

The examples above should give you a good idea of how to do generalized linear modelling in R. We have so far treated a GLM as a way of specifying the distribution of a response variable given the values of relevant covariates. In some situations however, the choice of distribution may not be clear — or even if it *is* clear, the preferred distribution may not be in the exponential family and therefore the usual theory/algorithms cannot be applied. In such situations, an alternative is *quasi-likelihood fitting*. This is based on the observation that the IWLS algorithm depends only on the means and variances of the chosen distributions — and specifically, upon the way in which the variance changes as a function of the mean (e.g. for a Poisson distribution the variance is equal to the mean, for the normal distribution the variance is constant and for the gamma distribution the variance is proportional to the square of the mean). Therefore, if we can postulate a plausible variance-mean relationship in any particular problem we may be able to obtain parameter estimates using the IWLS algorithm in the usual way. In general, the resulting estimates will not be maximum likelihood estimates (they are called *quasi-likelihood* estimates), but it can be shown that for all practical purposes they can be treated in the same

way as MLEs. For example, *quasi-deviances* can be calculated by pretending that the data come from an exponential family distribution with the postulated variance-mean relationship, and these can be used to compare models using χ^2 and F-tests in the usual way.

Quasi-likelihood fitting is most commonly used to deal with overdispersion in Poisson and binomial models. Overdispersion occurs when the variance of the data is greater than that suggested by the fitted model (for example, in the Poisson case the variance ought to be equal to the mean). It often arises as a result of unobserved factors that affect the response variable, for example. In many applications, the only real problem caused by overdispersion is that if it is ignored, standard errors of parameter estimates will be incorrect. Hence, if we can pretend that our data arise from a 'Poisson-like' or 'binomial-like' distribution with dispersion parameter $\phi \neq 1$, we can use quasi-likelihood to allow for the overdispersion when fitting models. This is the motivation behind the `quasibinomial` and `quasipoisson` family options to the `glm` command (see Table 1).

4.1 Example: tropical storms revisited

When you were looking at the tropical storm data, you hopefully noticed that the variance of the Pearson residuals is rather less than it should be if the Poisson assumption is correct. This is an (extremely rare!) example of *underdispersion*. Let's therefore pretend that the data arise from a'Poisson-like' distribution with dispersion parameter $\phi < 1$:

```
> storm.model3 <-
+    glm(Storms ~ N3.m09,family=quasipoisson(link="log"),data=storm.data)
> summary(storm.model3)
> anova(storm.model3,test="F")
```

The fitted values are exactly the same as those for `storm.model1` (**Question:** why is this?), but you will notice that the standard errors have changed to allow for the underdispersion. Also note that we must use an F test in the analysis of deviance, because the dispersion parameter is no longer known.

4.2 Other uses of quasi-likelihood

The `quasibinomial` and `quasipoisson` families are provided for convenience, because they are often required. For less common problems, you have to use the `quasi` family. This requires the specification of both a link function *and* a variance-mean relationship. For example:

```
> my.glm <- glm(y ~ x,family=quasi(link="sqrt",variance="mu^2"),data=mydata)
```

This would fit a model with a square root link function, pretending that the responses come from exponential family distributions with variance proportional to the square of the mean. In fact, these are just gamma distributions! The theory of quasi-likelihood says that it doesn't matter if your data *aren't* gamma distributed, so long as the variance function is right. You might wonder why we bother with a `quasi` family then, since you may as well just use the `Gamma` (or whatever) family instead. The answer is that `quasi` allows you to be much more flexible about your choice of link and variance functions. For example, the `Gamma` family doesn't allow a square root link function. If you want a square root link in a gamma GLM (and this problem arises in the context of normal distributions with non-constant variance), you can deceive the system by using the `quasi` family instead of `Gamma`!

5 Miscellanea

5.1 Factor covariates

None of the examples in this workshop have involved factor covariates (e.g. levels of a treatment). The concepts involved (relating to the coding of the design matrix and the use of contrasts) are exactly the same as those for linear models and have been covered in Lab 4.

5.2 Contingency Tables

When response variables and covariates are both categorical, it is common to present the data in tabular form, and to base analysis on the resulting *contingency table* of counts — the classic example is the 2×2 table e.g.

		Disease status		
		Absent	Present	TOTAL
Patient group	Treatment	n_{11}	n_{12}	$n_{1.}$
	Control	n_{21}	n_{22}	$n_{2.}$

Denoting by μ_{jk} the expected number of observations in row j and column k of such a table (with obvious extensions to higher dimensions), the standard analysis of such a table is via a loglinear model whereby $\ln \mu_{jk}$ is expressed as a linear function of covariates. Depending on how the data were collected, such tables may be modelled using either the Poisson, multinomial or product-multinomial distributions for numbers of observations in each cell of the table. However, *providing your model contains effects corresponding to any fixed margins*, each of these distributions yields exactly the same likelihood function, and so R commands for Poisson regression can be used to analyse contingency tables. *STAT0028* notes and the books in the reading list of *STAT0032* contain more details on this, with some worked examples in R.