STATS 330

Handout 12
Investigating GLMs and GOF using Simulation

STATS 330

Department of Statistics, University of Auckland

Investigating GLMs using Simulation

For the linear model, the estimators of the regression parameters are exactly unbiased, and the confidence intervals and prediction intervals have exactly the desired coverage. This is not the case for generalized linear models fitted to count or proportion data.

For GLMs we rely on having enough data so that the central limit theory to apply. Then the estimators will be approximately unbiased, and the intervals will have approximately correct coverage. We know that the central limit theorem will not apply if the data are too sparse.

Simulation for generalized linear models

In this handout we will do some simulations to investigate just how much data we need for the central limit theorem to kick in.

We can very easily modify our linear model simulation code to use it for simulating generalized linear models. This requires

(1) Specifying the (inverse) link function

(2) Changing the distribution from which we simulate y

(3) Using the `glm` function for the model fit.

Simulation for generalized linear models

Poisson regression

```r
x = 1:10; b0 = 0; b1 = 0.1;
## Calculating expected response values.
means <- exp(b0 + b1*x)  ## (1) Inverse link function used here.
## Setting the number of data sets and models to simulate.
n.sims <- 1000
## Creating vectors in which to store estimates.
est.b0 <- est.b1 <- numeric(n.sims)
## Creating matrices in which to store CIs.
ci.b0 <- ci.b1 <- matrix(0, nrow = n.sims, ncol = 2)
for (i in 1:n.sims) {
    ## Simulating the responses.
    y <- rpois(10, means)  ## (2) Poisson distribution used here.
    ## Fitting the model.
    fit <- glm(y ~ x, family = poisson)  ## (3) GLM used here.
    ## Extracting the model parameters.
    est.b0[i] <- coef(fit)[1]
    est.b1[i] <- coef(fit)[2]
    ## Extracting the CIs.
    ci.b0[i, ] <- confint(fit)[1, ]
    ci.b1[i, ] <- confint(fit)[2, ]
}
```

Simulation for generalized linear models

Poisson regression

We can re-run the above code a bunch of times, each time changing some of the choices of parameter value, number of observations, choice of confidence interval[1], etc.

[1]Recall that `confint.default` uses a different method for calculating the CI

Simulation for generalized linear models

Poisson regression

We can re-run the above code a bunch of times, each time changing some of the choices of parameter value, number of observations, choice of confidence interval[1], etc.

- With $b0 = 0$ and $b1 = 1$ we see that...

[1] Recall that `confint.default` uses a different method for calculating the CI

Simulation for generalized linear models

Poisson regression

We can re-run the above code a bunch of times, each time changing some of the choices of parameter value, number of observations, choice of confidence interval[1], etc.

- With $b0 = 0$ and $b1 = 1$ we see that...

- With $b0 = 0$ and $b1 = 0.1$ we see that...

[1] Recall that `confint.default` uses a different method for calculating the CI

Simulation for generalized linear models
Logistic regression

To simulate logistic regression modelling, we need to simulate proportions under a logistic model. Recall that logistic regression is on the log-odds scale, that is, its uses the logit link function. We can calculate the corresponding probabilities using the `plogis` function—it is the inverse of the logit link function.

In addition, don't forget that we also need to specify the number of trials associated with each observation.

Simulation for generalized linear models

Logistic regression

```r
x = 1:10; b0 = -3; b1 = 1; n = rep(100, length(x))
## Calculating expected response values.
probs <- plogis(b0 + b1*x)  ## (1) Inverse logit link function.
## Setting the number of data sets and models to simulate.
n.sims <- 1000
## Creating vectors in which to store estimates.
est.b0 <- est.b1 <- numeric(n.sims)
## Creating matrices in which to store CIs.
ci.b0 <- ci.b1 <- matrix(0, nrow = n.sims, ncol = 2)
for (i in 1:n.sims) {
    ## Simulating the responses.
    y <- rbinom(length(x), size = n, probs)  ## (2) Binomial used here.
    ## Fitting the model.
    fit <- glm(cbind(y, n - y) ~ x, family = binomial)  ## (3) GLM used here.
    ## Extracting the model parameters.
    est.b0[i] <- coef(fit)[1]
    est.b1[i] <- coef(fit)[2]
    ## Extracting the CIs.
    ci.b0[i, ] <- confint(fit)[1, ]
    ci.b1[i, ] <- confint(fit)[2, ]
}
```

Simulation for generalized linear models

Logistic regression

We can re-run this code a bunch of times under different scenarios. In addition to each time changing some of the choices of parameter value, number of observations, choice of confidence interval, we can now also change the number of trials n for each observation.

Simulation for generalized linear models

Logistic regression

We can re-run this code a bunch of times under different scenarios. In addition to each time changing some of the choices of parameter value, number of observations, choice of confidence interval, we can now also change the number of trials n for each observation.

- With $b0 = -3$ and $b1 = 1$ and $n = 100$ we see that...

Simulation for generalized linear models

Logistic regression

We can re-run this code a bunch of times under different scenarios. In addition to each time changing some of the choices of parameter value, number of observations, choice of confidence interval, we can now also change the number of trials n for each observation.

- With $b0 = -3$ and $b1 = 1$ and $n = 100$ we see that. . .

- With $b0 = -3$ and $b1 = 1$ and $n = 10$ we see that. . .

Simulation methods for goodness-of-fit

We are now going to use simulation to look at the sampling distribution of the deviance statistic.

Recall that we can use the deviance as an overall test of goodness of fit in a generalized linear model, by assuming it has an approximate chi-square distribution. This approximation requires both sufficient sample size, and for the data not to be sparse. Let's investigate. . .

Let's simulate some Poisson regression data

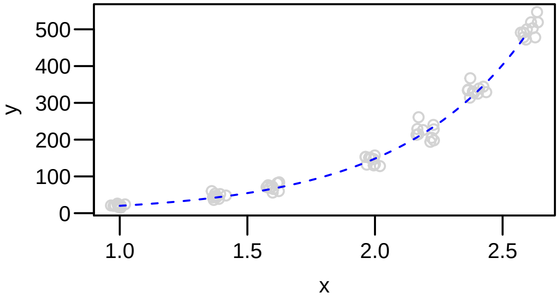
Suppose we have an explanatory variable X with many repeated values. It has the seven unique values of $x = 1.4, 1.6, 1.0, 2.0, 2.2, 2.4, 2.6$ repeated ten times each.
Say we have $\log \lambda_i = \log \mu_i = 1 + 2x$, that is, $\lambda_i = \mu_i = \exp(1 + 2x)$.

```
x = rep(c(1.4, 1.6, 1, 2.0, 2.2, 2.4, 2.6), each = 10)
n = length(x)
y = rpois(n, lambda=exp(1+2*x))
xy.df = data.frame(x=x, y=y)
```

Let's simulate some Poisson regression data...

Let's investigate the fruits of our simulation (with the underlying model superimposed)

```
plot(y~jitter(x), col="lightgray", data=xy.df)
xs <- seq(min(x), max(x), length=1e3)
lines(xs, exp(1+2*xs), lty=2, col="blue")
```



That's encouraging...

Let's simulate some Poisson data...

Let's analyze these data.

```
s0 <- glm(y~x, family=poisson, data=xy.df)
summary(s0)
## Call:
## glm(formula = y ~ x, family = poisson, data = xy.df)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0384     0.0566    18.3   <2e-16 ***
## x             1.9876     0.0243    81.7   <2e-16 ***
## ---
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 9590.671  on 69  degrees of freedom
## Residual deviance:   69.537  on 68  degrees of freedom
```

6,

We know the dispersion parameter is 1 as this data is a random sample from a Poisson distribution—so we would have no reason to believe we would have any evidence against this assumption. We would, in practice, test this by calculating $\Pr\left(\chi^2_{df=68} > 69.537\right) \approx 0.425$.

Using simulations to verify our practice. . .

This p-value is calculated in R as follows:

```
1-pchisq(deviance(s0), df.residual(s0))
## [1] 0.4255
```

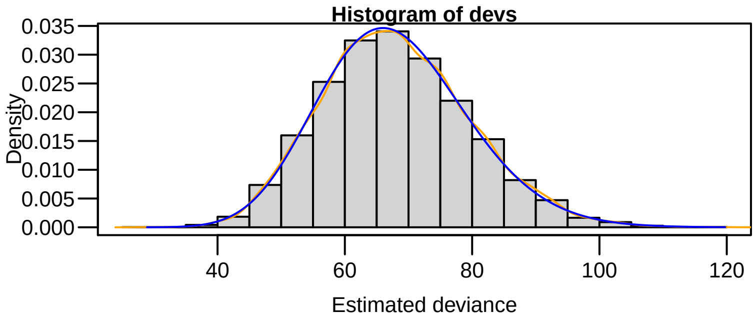
Let's randomly generate residual deviances to see if in fact they are consistent with coming from the claimed $\chi^2_{df=68}$ distribution.

```
Nsim <- 1e4
devs <- numeric(Nsim)
for (i in 1:Nsim) {
  ysim = rpois(n, lambda=exp(1+2*x))
  mod_i = glm(ysim~x, family=poisson)
  devs[i] = deviance(mod_i)
}
```

Using simulations to verify our practice...

Let's look at the histogram of the 10,000 estimates of deviances (`devs`) while also overlying a density plot (in orange) of these simulated data and of the claimed $\chi^2_{df=68}$ distribution. Here's how:

```r
hist(devs, freq = FALSE)
lines(density(devs), col="orange")
ds <- seq(min(devs), max(devs), length=1e3)
lines(ds, dchisq(ds, df=df.residual(s0)), col="blue")
```



Using simulations to verify our practice. . .

As you can see, the deviances appear to behave as if coming from a $\chi^2_{df=68}$ distribution since the blue and orange lines coincide. We would expect that about 5% of times we would reject the null hypothesis that the data comes from a Poisson distribution (i.e., has dispersion parameter $\phi = 1$). In other words, there is a 5% chance that we randomly obtain a deviance that is too large and hence falsely reject our null hypothesis.

Let's see how many of these `Nsim=10000` simulations 'fail' when we compare them to the 95% quantile of the $\chi^2_{df=68}$ distribution.

```r
mean(devs > qchisq(0.95, df=df.residual(s0)))
## [1] 0.0517
```

This is very comforting and tell us that these deviances 'behave' according to our theory.
Let's investigate a scenario where they don't quite work out.

Let's simulate some Poisson regression data

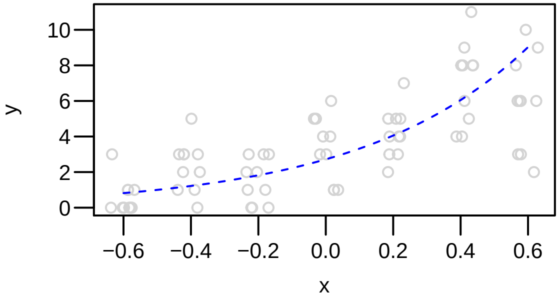
Suppose we have an explanatory variable X with many repeated values. It has the seven unique values of $x = -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6$ repeated ten times each. Compared to before, the values have been shifted to the LHS.
Say we have $\log \lambda_i = \log \mu_i = 1 + 2x$, that is, $\lambda_i = \mu_i = \exp(1 + 2x)$.

```
x <- rep(c(-0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6), each = 10)
n <- length(x)
y <- rpois(n, lambda=exp(1+2*x))
xy2.df <- data.frame(x=x, y=y)
```

Let's simulate some Poisson regression data...

Let's investigate the fruits of our simulation (with the underlying model superimposed).

```
plot(y~jitter(x), col="lightgray", data=xy2.df)
xs <- seq(min(x), max(x), length=1e3)
lines(xs, exp(1+2*xs), lty=2, col="blue")
```



That's encouraging...

Let's simulate some Poisson data...

Let's analyze these data.

```
s1 <- glm(y~x, family=poisson, data=xy2.df)
## Call:
## glm(formula = y ~ x, family = poisson, data = xy2.df)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0904     0.0749   14.55   <2e-16 ***
## x             1.5217     0.1766    8.61   <2e-16 ***
## ---
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 167.781  on 69  degrees of freedom
## Residual deviance:  85.062  on 68  degrees of freedom
```

We know the dispersion parameter is 1 as this data is a random sample from a Poisson distribution—so we would have no reason to believe we would have any evidence against this assumption. We would test this, in practice, by calculating $\Pr\left(\chi^2_{df=68} > 85.062\right) \approx 0.079$.

Using simulations to verify our practice. . .

This is calculated in R as follows:

```r
pchisq(deviance(s1), df.residual(s1), lower.tail = FALSE)
## [1] 0.07896
```

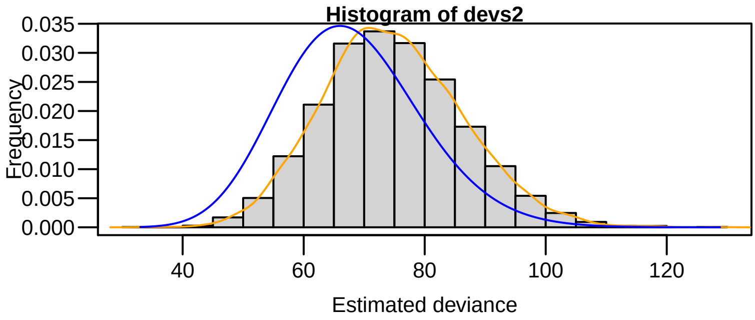
Let's see how many of these Nsim=10000 simulations 'fail' when we compare them to the 95% quantile of the $\chi^2_{df=68}$ distribution.

```r
Nsim <- 1e4
devs2 <- numeric(Nsim)
for (i in 1:Nsim) {
  ysim = rpois(n, lambda=exp(1+2*x))
  mod_i = glm(ysim~x, family=poisson)
  devs2[i] = deviance(mod_i)
}
```

Using simulations to verify our practice. . .

Let's look at the (probability) histogram of the 10,000 estimates of deviances (`devs2`) while also overlying a density plot (in orange) of these simulated data and the claimed $\chi^2_{df=68}$ distribution. Here's how:

```
hist(devs2, prob=TRUE)
lines(density(devs2), col="orange")
ds <- seq(min(devs2), max(devs), length=1e3)
lines(ds, dchisq(ds, df=df.residual(s1)), col="blue")
```



Using simulations to verify our practice...

There seems to be something wrong going on here. The distribution of `devs2` is at odds with these deviances coming from a $\chi^2_{df=68}$ distribution with `devs2` being to the RHS of what 'theory' says it should be. This may be a problem—unlike before this shows that we would reject the null hypothesis more times than we should. Again, let's compare our simulated deviances to the 95% quantile of the $\chi^2_{df=68}$ distribution.

```
mean(devs2 > qchisq(0.95, df=df.residual(s1)))
## [1] 0.1255
```

In this case the deviances aren't well approximated by a $\chi^2_{df=68}$ distribution. This is due to the fact that this is an approximation based on expected counts, typically, being bigger than the value 5. In this case this is not true and, so, the disparity. This can be corrected mathematically, but this requires greater knowledge of asymptotic distributional theory—and is quite tedious.

Using simulations to verify our practice...

So what do we do now?

We can use the deviances we have simulated to get around the inadequacy of the approximation to the $\chi^2_{df=68}$ distribution.

That is, instead of comparing our observed deviance of 85.06 to the $\chi^2_{df=68}$ distribution, we compare it to the simulated deviance values. We want to know how extreme our observed deviance is compared to these simulated values. Here's how we do it:

```
round(mean(devs2 > deviance(s1)), 3)
## [1] 0.185
```

which is markedly bigger than the p-value from erroneous assumption check
$\Pr\left(\chi^2_{df=68} > 85.062\right) \approx 0.079$.

Using simulations to verify our practice. . .

So the moral of all this is that:

We can check our underlying assumptions via simulation to see when our theory works (or when it doesn't) and use these as a basis of doing a corrected calculation.

Simulation allows us to investigate underlying behavior without the burden of having to do quite unwieldy mathematics.

The distribution of R^2

We'll finish this handout with another example. Here, we'll return to the linear model scenario, and suppose we are interested in the distribution of the R^2.

Let's generate a linear model where we have, say, $x_i \sim \text{Normal}(0,\ \sigma = 1)$ and $y_i = 1 + 2x + \varepsilon_i$ where, $\varepsilon_i \overset{iid}{\sim} \text{Normal}(0,\ \sigma = 1)$, say.

Clearly we know the true value of R^2 is not equal to zero as, by design, x is significant. We'll denote the true value of R^2 by ρ^2.

It can be shown that the density function of R^2 is given by the following formula, where p is the number of explanatory variables. In terms of ρ^2 it is the following brute:

$$\begin{aligned} f(r^2) &= \frac{1}{\Gamma\left(\frac{n-1}{2}\right)\ \Gamma\left(\frac{n-p-1}{2}\right)} \left(1-\rho^2\right)^{\frac{n-1}{2}} \left(1-r^2\right)^{\frac{n-p-2}{2}} \left(r^2\right)^{\frac{p-3}{2}} \\ &\times \sum_{i=0}^{\infty} \frac{(\rho^2 r^2)^i\ \Gamma^2\left(\frac{n-1}{2}+i\right)}{i!\ \Gamma\left(\frac{p-1}{2}+i\right)} \quad \text{for} \quad 0 < r^2 < 1. \end{aligned}$$

The distribution of R^2

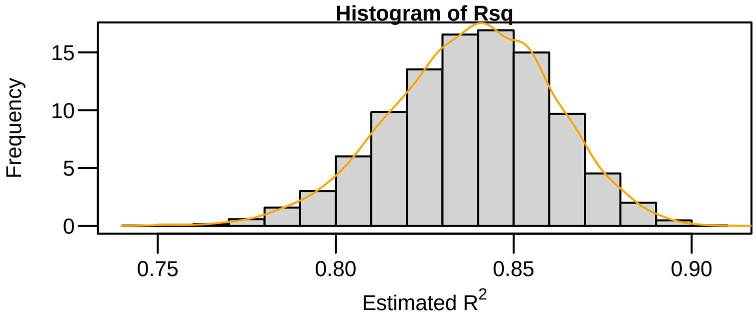
This is ghastly beyond all belief (unless you're a real math nerd) and I think it best if we leave insight about the distribution to simulation. In short, let's simulate our way out of this nightmare.

```
n <- 1e2
x <- rnorm(n)  # mean = 0 and sd=1 by default
Nsim <- 1e4
Rsq <- numeric(Nsim)
for (i in 1:Nsim) {
  ysim <- rnorm(n, mean=1+2*x)  # sd=1 by default
  mod_i <- lm(ysim~x)
  Rsq[i] <- summary(mod_i)$r.squared
}
head(Rsq, 5)
## [1] 0.8264 0.8321 0.8609 0.8412 0.8567
```

The distribution of R^2

Let's see how these data are distributed:

```r
hist(Rsq, prob=TRUE)
lines(density(Rsq), col="orange")
```



The distribution of R^2

We can very easily compute other statistics, such as the mean, median, standard deviation, and quantiles for these R^2 values:

```
c(mean(Rsq), median(Rsq), sd(Rsq))
## [1] 0.83809 0.83928 0.02278
quantile(Rsq, c(0.025, 0.975))
##   2.5%  97.5%
## 0.7906 0.8801
```

all without recourse to using the terrifying formula above!

In short—when life gets hard, try using simulation to make it easier.