Stat 5421 Lecture Notes: Overdispersion

Charles J. Geyer

November 15, 2023

Contents

1 License

2 R

- The version of R used to make this document is 4.3.2.
- The version of the `rmarkdown` package used to make this document is 2.25.

3 Quasilikelihood and Estimating Equations

3.1 Variance Functions

In linear models (fit by R function `lm`) we assume the components of the response vector are independent, normally distributed, and same variance (homoscedastic). The variance is unrelated to the mean. The mean of each component can be any real number. The common variance of all the components can be any positive real number.

In generalized linear models (fit by R function `glm`) none of these assumptions hold except the components of the response vector are independent.

- There are constraints on the means.

- There is only one parameter for binomial and Poisson models, so the variance and mean cannot be separately varied.

For binomial regression,

- components of the response vector are independent Binomial(n_i, π_i) distributed.
- Means $\mu_i = n_i \pi_i$ satisfy $0 \le \mu_i \le n_i$.
- Variances are a function of means

$$\operatorname{var}(y_i) = n_i V(\pi_i) = n_i \pi_i (1 - \pi_i)$$

  where

$$V(\pi) = \pi(1 - \pi)$$

For Poisson regression,

- components of the response vector are independent Poisson(μ_i) distributed.
- Means μ_i satisfy $0 \le \mu_i < \infty$.
- Variances are a function of means

$$\operatorname{var}(y_i) = V(\mu_i) = \mu_i$$

  where

$$V(\mu) = \mu$$

3.2 Modeling without Models

Everything we have done so far relied on statistical models (families of probability distributions). Now we are going to do something different: statistics without models. We are only going to make assumptions about means and variances, not about whole probability distributions.

In generalized linear model theory, this approach is called quasilikelihood, although the approach can be explained without even defining this term, and we shall do so.

We assume the means are given by the same function of the parameters ("coefficients") as in binomial and Poisson regression. Thus we will have the same maximum likelihood estimates (MLE) of "coefficients" and means with and without overdispersion. Except since we don't have a model, we don't have a likelihood, thus these cannot be maximum *likelihood* estimates. We say they are maximum *quasilikelihood* estimates, or just estimates. But they are the *same* estimates as the MLE for ordinary binomial or Poisson regression.

The difference is that we do not assume the same variance function as ordinary binomial or Poisson regression. The variance function could be anything, but for simplicity we (like everybody else) assume the variance function is a constant times the ordinary variance function

- The variance is $\phi n_i V(\pi_i)$ for binomial.
- The variance is $\phi V(\mu_i)$ for Poisson.

Because we assumed the estimates of coefficients and means are the same as for ordinary binomial or Poisson regression, we can estimate the means without knowing ϕ. Call those estimates $\widehat{\mu}_i$.

3.3 Estimating the Dispersion Parameter

For Poisson, by assumption,

$$\frac{(y_i - \mu_i)^2}{\phi V(\mu_i)}$$

has variance one. Thus

$$\sum_{i=1}^{n} \frac{(y_i - \mu_i)^2}{\phi V(\mu_i)}$$

has variance n. Hence

$$\hat{\phi} = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \mu_i)^2}{V(\mu_i)}$$

would be a good estimate of ϕ except that we don't know the μ_i.

So we plug in estimated values

$$\hat{\phi} = \frac{1}{n - p} \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

and as usual divide by $n - p$ instead of n where p is the number of "coefficients" (the number of parameters needed to specify the $\hat{\mu}_i$).

This division by $n-p$ has no exact theory justifying it. We know that in linear models, dividing by $n-p$ gives an unbiased estimate of variance (or so we are told, this is proved in the theory class 5102 Slides 31–38, Deck 5). But we are not doing linear models, so dividing by $n - p$ is just an analogy, not real math. Nevertheless, it is the conventional thing to do. (Of course, it is correct for large n, but for large n the difference between n and $n - p$ is inconsequential.)

For binomial, the corresponding estimate is

$$\hat{\phi} = \frac{1}{n - p} \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{n_i V(\hat{\mu}_i / n_i)}$$

4 Example: Agresti Section 4.7.4

```
library(CatDataAnalysis)
data(table_4.7)
names(table_4.7)
```

```
## [1] "litter" "group"  "n"      "y"
```

```
sapply(table_4.7, class)
```

```
##    litter     group         n         y
## "integer" "integer" "integer" "integer"
```

R function `glm` wants binomial data with sample sizes greater than one presented as a two-column matrix whose columns are successes and failures, so we have to make that. Instead we are given `y` equals successes and `n` equals totals (successes + failures), apparently.

```
with(table_4.7, all(0 <= y & y <= n))
```

```
## [1] TRUE
```

```
with(table_4.7, range(n))
```

```
## [1]  1 17
```

```
resp <- with(table_4.7, cbind(dead = y, alive = n - y))
resp
```

```
##       dead alive
##  [1,]    1     9
##  [2,]    4     7
##  [3,]    9     3
##  [4,]    4     0
##  [5,]   10     0
##  [6,]    9     2
```

```
##  [7,]    9     0
##  [8,]   11     0
##  [9,]   10     0
## [10,]    7     3
## [11,]   12     0
## [12,]    9     1
## [13,]    8     0
## [14,]    9     2
## [15,]    4     2
## [16,]    7     2
## [17,]   14     0
## [18,]    7     5
## [19,]    9     2
## [20,]    8     5
## [21,]    5     9
## [22,]   10     0
## [23,]   10     2
## [24,]    8     5
## [25,]   10     0
## [26,]    3    11
## [27,]   13     0
## [28,]    3     1
## [29,]    8     0
## [30,]    5     8
## [31,]   12     0
## [32,]    1     9
## [33,]    1     2
## [34,]    1    12
## [35,]    0    12
## [36,]    4    10
## [37,]    2     7
## [38,]    2    11
## [39,]    1    15
## [40,]    0    11
## [41,]    0     4
## [42,]    0     1
## [43,]    0    12
## [44,]    0     8
## [45,]    1    10
## [46,]    0    14
## [47,]    1    13
## [48,]    0    11
## [49,]    0     3
## [50,]    0    13
## [51,]    2     7
## [52,]    2    15
## [53,]    0    15
## [54,]    0     2
## [55,]    1    13
## [56,]    0     8
## [57,]    0     6
## [58,]    0    17
```

Variable litter is just sequence numbers

```r
with(table_4.7, identical(litter, seq(along = litter)))
```

```
## [1] TRUE
```

Group is numeric but we need to make it a factor.

```r
dat <- transform(table_4.7, group = as.factor(group))
```

Now just for comparison, we will fit logistic regression

```r
gout.logistic <- glm(resp ~ group, data = dat, family = binomial)
summary(gout.logistic)
```

```
##
## Call:
## glm(formula = resp ~ group, family = binomial, data = dat)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.1440     0.1292   8.855  < 2e-16 ***
## group2       -3.3225     0.3308 -10.043  < 2e-16 ***
## group3       -4.4762     0.7311  -6.122 9.22e-10 ***
## group4       -4.1297     0.4762  -8.672  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 509.43  on 57  degrees of freedom
## Residual deviance: 173.45  on 54  degrees of freedom
## AIC: 252.92
##
## Number of Fisher Scoring iterations: 5
```

Now allow for overdispersion

```r
gout.over <- glm(resp ~ group, data = dat, family = quasibinomial)
summary(gout.over)
```

```
##
## Call:
## glm(formula = resp ~ group, family = quasibinomial, data = dat)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1440     0.2187   5.231 2.81e-06 ***
## group2       -3.3225     0.5600  -5.933 2.18e-07 ***
## group3       -4.4762     1.2375  -3.617 0.000656 ***
## group4       -4.1297     0.8061  -5.123 4.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 2.864945)
##
##     Null deviance: 509.43  on 57  degrees of freedom
## Residual deviance: 173.45  on 54  degrees of freedom
## AIC: NA
```

```
##
## Number of Fisher Scoring iterations: 5
```

We see that we do have exactly the same coefficients

```
all.equal(coef(gout.logistic), coef(gout.over))
```

```
## [1] TRUE
```

And exactly the same mean values

```
all.equal(fitted(gout.logistic), fitted(gout.over))
```

```
## [1] TRUE
```

Now try predicted values

```
pout.over <- predict(gout.over, newdata = data.frame(group = factor(1:4)),
    type = "response", se.fit = TRUE)
pout.over
```

```
## $fit
##          1          2          3          4
## 0.75840979 0.10169492 0.03448276 0.04807692
##
## $se.fit
##          1          2          3          4
## 0.04006599 0.04709542 0.04055320 0.03550672
##
## $residual.scale
## [1] 1.692615
```

Are these the same as the predicted values for logistic regression?

```
pout.logistic <- predict(gout.logistic,
    newdata = data.frame(group = factor(1:4)),
    type = "response", se.fit = TRUE)
pout.logistic
```

```
## $fit
##          1          2          3          4
## 0.75840979 0.10169492 0.03448276 0.04807692
##
## $se.fit
##          1          2          3          4
## 0.02367106 0.02782406 0.02395890 0.02097744
##
## $residual.scale
## [1] 1
```

```
all.equal(pout.logistic$fit, pout.over$fit)
```

```
## [1] TRUE
```

```
pout.over$se.fit / pout.logistic$se.fit
```

```
##        1        2        3        4
## 1.692615 1.692615 1.692615 1.692615
```

What is that?

```
phi.hat <- summary(gout.over)$dispersion
phi.hat
```

```
## [1] 2.864945
```

```
sqrt(phi.hat)
```

```
## [1] 1.692615
```

Indeed we are just inflating the estimated variance by a factor of 2.864945 and the estimated standard deviation by a factor of 1.692615.

4.1 Testing for Overdispersion

One might think that, since we have no model, we cannot do hypothesis tests about the dispersion. But hypothesis tests only need a distribution under the null hypothesis, and we do have that. Null hypothesis is ordinary binomial; alternative hypothesis is overdispersed binomial.

We can do the test by simulation, as we did in the section on the parametric bootstrap in the notes on Chapter 9.

```
# set random number generator seed for reproducibility
set.seed(42)

n <- dat$n
nboot <- 999
mu.hat <- fitted(gout.logistic)
phi.star <- double(nboot)
for (iboot in 1:nboot) {
    y.star <- rbinom(length(n), n, mu.hat)
    resp.star <- cbind(dead = y.star, alive = n - y.star)
    gout.over <- glm(resp.star ~ group, data = dat, family = quasibinomial)
    phi.star[iboot] <- summary(gout.over)$dispersion
}
all.phi.values <- c(phi.star, phi.hat)
mean(all.phi.values >= phi.hat)
```

```
## [1] 0.001
```

None of the simulated dispersion values are as large as the observed value. This is very strong evidence for overdispersion.

4.2 On Not Testing for Overdispersion

But the natural idea of only using `family = quasibinomial` when there seems to be evidence for it using the test in the preceeding section is actually the Wrong Thing.

When one does a composite procedure having two or more steps, one must consider the composite procedure as the procedure. This seems obvious: what one does is what one does. But that is not what most people do when doing composite procedures they invented. In this example, the idea is to use ordinary binomial if the test of overdispersion accepts the null hypothesis and to use overdispersed binomial if the test of overdispersion rejects the null hypothesis, where "use ordinary binomial" or "use overdispersed binomial" means *pretend no pretest had been done*. This is clearly the Wrong Thing. The composite procedure is not the simple procedure that is the second step of the composite procedure.

Hence the conclusion is that if you want to use overdispersion, then use it. *Always.* Don't use a composite procedure that is the Wrong Thing.

This is a general principle of statistics. This seemingly reasonable idea — many naive users of statistics invent it — has zero theoretical justification, is recommended by zero textbooks, and is obviously wrong once you think about it in the right way.

Every multistage procedure invented by newbies has this problem. When you do stage one, and then proceed with stage two *pretending that stage one never happened and that its results satisfy the assumptions for stage two* and so on (maybe stages three and four), you are in the realm of complete bogosity.

Of course, if you bootstrap the whole multistage procedure, then that is the Right Thing but that is not what we are talking about there.

5 Example: Agresti Section 4.7.2

We refit the horseshoe crab data from Section 4.3.2 allowing for overdispersion.

```
# clean up R global environment
rm(list = ls())

data(table_4.3)
names(table_4.3)
```

```
## [1] "color"  "spine"  "width"  "satell" "weight" "y"
```

We found out in homework 3 that the apparent best fitting model when we did not use overdispersion was

```
gout.no.over <- glm(satell ~ color + weight, family = poisson, data = table_4.3)
summary(gout.no.over)
```

```
##
## Call:
## glm(formula = satell ~ color + weight, family = poisson, data = table_4.3)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.614e-01  3.008e-01   0.869  0.38496
## color       -1.728e-01  6.155e-02  -2.808  0.00499 **
## weight       5.459e-04  6.749e-05   8.088 6.05e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 552.79  on 170  degrees of freedom
## AIC: 914.09
##
## Number of Fisher Scoring iterations: 6
```

If we allow for overdispersion

```
gout.over <- glm(satell ~ color + weight, family = quasipoisson,
    data = table_4.3)
summary(gout.over)
```

```
##
## Call:
## glm(formula = satell ~ color + weight, family = quasipoisson,
##     data = table_4.3)
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2613642  0.5370711   0.487    0.627
## color       -0.1728172  0.1098793  -1.573    0.118
## weight       0.0005459  0.0001205   4.531  1.1e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 3.18719)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 552.79  on 170  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

we see that everything is the same except that we are allowing for a lot of overdispersion.