**Chapter 5 – Model selection and evaluation**

The purpose of this chapter is

* Determine the most parsimonious model that best estimates the response
* Assess how well the model fits the data

**Section 5.1 – Variable selection**

In some situations, you will have a specific number of explanatory variables where you want to include all of them in the model. For example, in a designed experiment setting, you may have three explanatory variables and you want to include all of them in the model.

In other situations, due to knowledge of the subject matter, the specific variables and their form may lead to a particular model. For example, explanatory variables that are “non-significant” may still be included in the model.

In contrast, you may have access to a number of explanatory variables and you do not know which to include in a model. For example, this includes observational studies such as with placekicking data of Chapter 2. Of course, you would only like to choose those explanatory variables that are “important” rather than those that do not help estimate the response. The purpose of this section is to examine methods that help in this regard.

There will be MANY similarities here to what you would learn in a regression analysis course.

Information criteria

Suppose you would like to determine which of two estimated models are “best” to use for a data set. A LRT can be used, but one model needs to be “nested” within the other (all variables in one model need to be within the other model).

A residual deviance or even just a likelihood function evaluated at the parameter estimates could be used to compare the two models. You may think that the smaller the residual deviance (or larger the likelihood function), the “better” the model. Unfortunately, this is not good to do for similar reasons as why R2 generally should not be used to compare two models. What is this reason!?

*Information criteria* avoid these problems by using the likelihood function and adding a “penalty” to it. The general form is

IC(k) = -2log(Likelihood function evaluated at parameter estimates) + kr

where k is a numerical penalty and r is the total number of parameters in the model. The most common IC(k) are:

1. Akaike’s Information Criterion



1. Corrected AIC



1. Bayesian Information Criterion (BIC)



Comments:

* See Burnham and Anderson (*Sociological Methods & Research*, 2004, p. 261-304) for discussion
* 2n/(n-r-1) > 2; for large n relative to r, notice that 2n/(n-r-1) ≈ 2
* log(n) > 2 when n ≥ 8

With all three information criteria, the best estimated model has the lowest IC(k). Why?

Comments:

* AIC tends to favor larger models than AICc and BIC. This can be good or bad. Why?
* What if a large number of models have very similar IC(k) values?
* Which information criteria is best? There are various statistical evaluation methods to examine “best” and these are described in my book. Overall, one information criterion is not necessarily best for all statistical evaluation methods.
* The AIC() function can be used to calculate the information criteria, where the k argument is used to specify the appropriate value of k.

All-subsets regression

Now that we have numerical measures to decide which model is “best”, we need to determine which models to compare. One way is to compare “all possible” models that one could have!

Simply, if you have P total variables to choose from, one could have 2P different models (this depends – more on this soon). For example, if there are P = 3 explanatory variables x1, x2, and x3 to consider, the possible models include:

1. No explanatory variables
2. x1
3. x2
4. x3
5. x1, x2
6. x1, x3
7. x2, x3
8. x1, x2, x3

Thus, 23 = 8 possible models. From these models, the “best” model is the model with the smallest IC(k). Also, because “subsets” of all possible variables are considered, this method is referred to as all-subsets regression.

Of course, 2P can become large quickly as P grows. For this reason, special algorithms have been developed to search among the 2P models to find the best. We will use an algorithm known as the genetic algorithm when P is large. This algorithm works as follows:

1. Random sets of explanatory variables are put together into models. This is called the “first generation” of models.
2. From the best performing of these models, new models are put together to form a “second generation” of models.
3. This process continues for multiple generations with random additions and deletions (*mutations*) of explanatory variables to determine if better models can be formed.
4. Eventually the algorithm converges to a “best” model.

Due to the stochastic nature of the algorithm (notice the use of the word “random” above), different final models could result. Therefore, it is best to run the algorithm more than once to determine if even better models can be found.

Notice we are NOT really considering ALL possible models here even if all 2P models as described above are examined. For example, what about interactions or various transformations of variables? Thus, it is somewhat a misnomer to really think these methods are looking at all possible models. However, it does at least provide a good way to start to determine which models are best.

We will use the function glmulti() of the glmulti package to perform all-subsets regression. This function allows for both all 2P main-effects models to be examined and the genetic algorithm to be applied to search for models. In additional, all pairwise interactions can be included in the search too! This leads to 2P+A, where A = P(P – 1)/2, different possible models!

Example: Placekicking (AllSubsetsPlacekick.R, Placekick.csv)

How many main-effects models are there to examine?

> placekick <- read.table(file = "C:\\data\\Placekick.csv",

header = TRUE, sep = ",")

> head(placekick)

week distance change elap30 PAT type field wind good

1 1 21 1 24.7167 0 1 1 0 1

2 1 21 0 15.8500 0 1 1 0 1

3 1 20 0 0.4500 1 1 1 0 1

4 1 28 0 13.5500 0 1 1 0 1

5 1 20 0 21.8667 1 0 0 0 1

6 1 25 0 17.6833 0 0 0 0 1

> library(glmulti)

# Using "good ~ ." to include all explanatory variables

> search.1.aicc <- glmulti(y = good ~ ., data = placekick,

fitfunction = "glm", level = 1, method = "h", crit =

"aicc", family = binomial(link="logit"))

Initialization…

TASK: Exhaustive screening of candidate set.

Fitting…

After 50 models:

Best model: good~1+week+distance+change+PAT

Crit= 767.299644897567

Mean crit= 861.376317807727

After 100 models:

Best model: good~1+week+distance+change+PAT

Crit= 767.299644897567

Mean crit= 849.706367688597

<OUTPUT EDITED>

After 250 models:

Best model: good~1+distance+change+PAT+wind

Crit= 766.728784139471

Mean crit= 774.167191225549

Completed.

> print(search.1.aicc)

glmulti.analysis

Method: h / Fitting: glm / IC used: aicc

Level: 1 / Marginality: FALSE

From 100 models:

Best IC: 766.728784139471

Best model:

[1] “good ~ 1 + distance + change + PAT + wind”

Evidence weight: 0.0669551501665866

Worst IC: 780.47950528365

12 models within 2 IC units.

51 models to reach 95% of evidence weight.

> aa <- weightable(search.1.aicc)

> head(aa) #Not in program

model aicc weights

1 good ~ 1 + distance + change + PAT + wind 766.7288 0.06695515

2 good ~ 1 + week + distance + change + PAT + wind 767.1329 0.05470419

3 good ~ 1 + week + distance + change + PAT 767.2996 0.05032956

4 good ~ 1 + distance + change + PAT 767.3607 0.04881733

5 good ~ 1 + distance + PAT + wind 767.6899 0.04140743

6 good ~ 1 + distance + change + PAT + type + wind 768.1182 0.03342523

> cbind(model = aa[1:5,1], round(aa[1:5,c(2,3)], 3)) #In

program

model aicc weights

1 good ~ 1 + distance + change + PAT + wind 766.729 0.067

2 good ~ 1 + week + distance + change + PAT + wind 767.133 0.055

3 good ~ 1 + week + distance + change + PAT 767.300 0.050

4 good ~ 1 + distance + change + PAT 767.361 0.049

5 good ~ 1 + distance + PAT + wind 767.690 0.041

Comments:

* A model is specified in the y argument in order to obtain the response and explanatory variables. This does not mean that ONLY this model will be evaluated!
* The level argument is equal to 1 for main effects and 2 for the main effects with interactions.
* The method argument is equal to "h" for an exhaustive search (e.g., 2P models when level = 1) and "g" for the genetic algorithm.

Including the pairwise interactions leads to 236 different models! The genetic algorithm is then the best way to approach this problem.

> set.seed(87112811) # program used for book did not set a

seed

> search.gmarg.aicc <- glmulti(y = good ~ ., data =

placekick, fitfunction = "glm", level = 2,

marginality = TRUE, method = "g", crit = "aicc", family

= binomial(link = "logit"))

Initialization...

TASK: Genetic algorithm in the candidate set.

Initialization...

Algorithm started...

After 10 generations:

Best model: good~1+week+distance+change+elap30+PAT+type+field+wind+PAT:distance+PAT:change+type:week+field:week+field:change+wind:PAT+wind:field

Crit= 777.416474495123

Mean crit= 786.201784496673

Change in best IC: -9222.58352550488 / Change in mean IC: -9213.79821550333

After 20 generations:

Best model: good~1+week+distance+change+elap30+PAT+type+field+wind+PAT:distance+PAT:change+type:week+field:distance+wind:distance+wind:elap30+wind:field

Crit= 773.369488512734

Mean crit= 782.745699618429

Change in best IC: -4.0469859823894 / Change in mean IC: -3.45608487824404

<OUTPUT EDITED>

After 610 generations:

Best model: good~1+week+distance+change+PAT+field+wind+PAT:distance+wind:distance+wind:field

Crit= 763.973819647683

Mean crit= 768.63562968283

Change in best IC: 0 / Change in mean IC: 0

After 620 generations:

Best model: good~1+week+distance+change+PAT+field+wind+PAT:distance+wind:distance+wind:field

Crit= 763.973819647683

Mean crit= 768.63562968283

Change in best IC: 0 / Change in mean IC: 0

After 630 generations:

Best model: good~1+week+distance+change+PAT+field+wind+PAT:distance+wind:distance+wind:field

Crit= 763.973819647683

Mean crit= 768.619137549168

Improvements in best and average IC have bebingo en below the specified goals.

Algorithm is declared to have converged.

Completed.

> print(search.gmarg.aicc)

glmulti.analysis

Method: g / Fitting: glm / IC used: aicc

Level: 2 / Marginality: TRUE

From 100 models:

Best IC: 763.973819647683

Best model:

[1] "good ~ 1 + week + distance + change + PAT + field + wind + wind:distance + "

[2] " wind:field"

[1] "good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + "

[2] " wind:distance + wind:field"

Evidence weight: 0.039326682519735

Worst IC: 774.923121972759

28 models within 2 IC units.

56 models to reach 95% of evidence weight.

Convergence after 630 generations.

Time elapsed: 13.6449862003326 minutes.

> head(weightable(search.gmarg.aicc))

model

1 good ~ 1 + week + distance + change + PAT + field + wind + wind:distance + wind:field

2 good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + wind:distance + wind:field

3 good ~ 1 + distance + change + PAT + field + wind + PAT:distance + wind:distance + wind:field

4 good ~ 1 + distance + change + PAT + field + wind + wind:distance + wind:field

5 good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + wind:week + wind:distance + wind:field

6 good ~ 1 + distance + PAT + field + wind + PAT:distance + wind:distance + wind:field

aicc weights

1 763.9738 0.03932668

2 763.9738 0.03932668

3 764.0935 0.03704267

4 764.0935 0.03704267

5 764.1698 0.03565665

6 764.3575 0.03246224

The marginality = TRUE argument value instructs glmulti() to only consider those models where the appropriate main effects are always included corresponding to when an interaction is in a model. Note that this will reduce the number of models considered, but still there are too many to look at all of them.

When I ran the code again with the seed of 91051211, I obtain the following output:

> head(weightable(search.gmarg.aicc))

model

1 good ~ 1 + week + distance + change + PAT + type + field + wind + field:type + wind:distance + wind:type + wind:field

2 good ~ 1 + week + distance + change + PAT + type + field + wind + PAT:distance + field:type + wind:distance + wind:field

3 good ~ 1 + week + distance + change + PAT + type + field + wind + wind:distance + wind:field

4 good ~ 1 + week + distance + change + PAT + type + field + wind + field:type + wind:distance + wind:field

5 good ~ 1 + distance + change + PAT + type + field + wind + field:type + wind:distance + wind:field

6 good ~ 1 + week + distance + change + PAT + type + field + wind + field:type + wind:week + wind:distance + wind:field

aicc weights

1 765.3552 0.04174521

2 765.3552 0.04174521

3 765.3552 0.04174521

4 765.3552 0.04174521

5 765.3904 0.04101522

6 765.5400 0.03806065

And again using a seed of 11256012:

> head(weightable(search.gmarg.aicc))

model

1 good ~ 1 + distance + change + PAT + wind + PAT:distance + wind:distance

2 good ~ 1 + distance + change + PAT + wind + wind:distance

3 good ~ 1 + week + distance + change + PAT + wind + wind:distance

4 good ~ 1 + week + distance + change + PAT + wind + PAT:distance + wind:distance

5 good ~ 1 + week + distance + change + PAT + field + wind + wind:distance + wind:field

6 good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + wind:distance + wind:field

aicc weights

1 763.3304 0.03876313

2 763.3304 0.03876313

3 763.5875 0.03408860

4 763.5875 0.03408860

5 763.9738 0.02810025

6 763.9738 0.02810025

Comments:

* The last paragraph in Section 5.1.3 gives some good discussion about the application of these methods. In summary, it is best to only include those explanatory variables and those interactions among explanatory variables that make sense in the context of the problem.
* You may experience problems with initially using the glmulti package as detailed in a footnote in Section 5.1.3. These problems are due to the package relying on the rJava package. To avoid these problems, you need to use the same version of R (in terms of 32-bit or 64-bit) as the version of Java (32-bit or 64-bit) that you have on your computer. For example, I have encountered the problem in the following instance. I was using 32-bit Internet Explorer when I installed Java. This lead to the 32-bit version of Java to be installed on my computer. When I try to use 64-bit R, I received the following error message when trying to use the glmulti package:

> library(glmulti)

Loading required package: rJava

Error : .onLoad failed in loadNamespace() for 'rJava', details:

call: fun(libname, pkgname)

error: No CurrentVersion entry in Software/JavaSoft registry! Try re-installing Java and make sure R and Java have matching architectures.

Error: package ‘rJava’ could not be loaded

The solution is to use the same version of R as Java on your computer.

Stepwise search algorithms

These methods provide another way to find the “best” model from all possible models. The advantage these have over the previous methods is that they are simple and generally faster. The disadvantage is that they are not as good at finding the “best” model. I recommend avoiding these methods.

Why discuss these methods then???

* There may be instances where the previous methods can not be implemented easily. This includes situations where models are much more complicated than those we discuss in our class. Also, this includes situations where the glmulti package does not work!
* These methods are still prevalently used. You may need to read a paper which uses them, so it is good to understand how they are implemented.

There are three ways to implement these selection methods:

1. Forward selection
   1. Compute IC(k) for a model with no explanatory variables
   2. Compute IC(k) for all possible one explanatory variable models. Find the model that reduces the IC(k) value the most. If no model reduces IC(k) compared to the model in a), then use the model in a) as your “best” model.
   3. Compute IC(k) for all possible two explanatory variable models, where the model from step b) is used to start from. Find the model that reduces the IC(k) value the most. If no model reduces IC(k) compared to the model in b), then use the model in b) as your “best” model.
   4. Continue adding explanatory variables one at a time until no additional variable decreases the IC(K).
2. Backward selection – What do you think the steps are?
3. Alternating stepwise selection – What do you think the steps are?

Interactions are sometimes includes in this process as well. However, one needs to make sure that the main effects corresponding to an interaction are still in the model.

Example: Placekicking (stepwisePlacekick.R, Placekick.csv)

Below is forward selection implemented by the step() function using BIC:

> placekick <- read.table(file = "C:\\data\\Placekick.csv",

header = TRUE, sep = ",")

> empty.mod <- glm(formula = good ~ 1, family =

binomial(link = logit), data = placekick)

> full.mod <- glm(formula = good ~ ., family =

binomial(link = logit), data = placekick)

> forw.sel <- step(object = empty.mod, scope = list(upper =

full.mod), direction = "forward", k =

log(nrow(placekick)), trace = TRUE)

Start: AIC=1020.69

good ~ 1

Df Deviance AIC

+ distance 1 775.75 790.27

+ PAT 1 834.41 848.93

+ change 1 989.15 1003.67

<none> 1013.43 1020.69

+ elap30 1 1007.71 1022.23

+ wind 1 1010.59 1025.11

+ week 1 1011.24 1025.76

+ type 1 1011.39 1025.92

+ field 1 1012.98 1027.50

Step: AIC=790.27

good ~ distance

Df Deviance AIC

+ PAT 1 762.41 784.20

<none> 775.75 790.27

+ change 1 770.50 792.29

+ wind 1 772.53 794.32

+ week 1 773.86 795.64

+ type 1 775.67 797.45

+ elap30 1 775.68 797.47

+ field 1 775.74 797.53

Step: AIC=784.2

good ~ distance + PAT

Df Deviance AIC

<none> 762.41 784.20

+ change 1 759.33 788.38

+ wind 1 759.66 788.71

+ week 1 760.57 789.62

+ type 1 762.25 791.30

+ elap30 1 762.31 791.36

+ field 1 762.41 791.46

> anova(forw.sel)

Analysis of Deviance Table

Model: binomial, link: logit

Response: good

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev

NULL 1424 1013.43

distance 1 237.681 1423 775.75

PAT 1 13.335 1422 762.41

While “AIC” is listed in the output, it is really BIC that is calculated because of what was specified in the k argument.

Below is forward selection implemented by the step() function using AIC:

> forw.sel2 <- step(object = empty.mod, scope = list(upper=

full.mod), direction = "forward", k = 2, trace = TRUE)

Start: AIC=1015.43

good ~ 1

Df Deviance AIC

+ distance 1 775.75 779.75

+ PAT 1 834.41 838.41

+ change 1 989.15 993.15

+ elap30 1 1007.71 1011.71

+ wind 1 1010.59 1014.59

+ week 1 1011.24 1015.24

+ type 1 1011.39 1015.39

<none> 1013.43 1015.43

+ field 1 1012.98 1016.98

Step: AIC=779.75

good ~ distance

Df Deviance AIC

+ PAT 1 762.41 768.41

+ change 1 770.50 776.50

+ wind 1 772.53 778.53

<none> 775.75 779.75

+ week 1 773.86 779.86

+ type 1 775.67 781.67

+ elap30 1 775.68 781.68

+ field 1 775.74 781.74

Step: AIC=768.41

good ~ distance + PAT

Df Deviance AIC

+ change 1 759.33 767.33

+ wind 1 759.66 767.66

<none> 762.41 768.41

+ week 1 760.57 768.57

+ type 1 762.25 770.25

+ elap30 1 762.31 770.31

+ field 1 762.41 770.41

Step: AIC=767.33

good ~ distance + PAT + change

Df Deviance AIC

+ wind 1 756.69 766.69

+ week 1 757.26 767.26

<none> 759.33 767.33

+ elap30 1 759.11 769.11

+ type 1 759.13 769.13

+ field 1 759.33 769.33

Step: AIC=766.69

good ~ distance + PAT + change + wind

Df Deviance AIC

<none> 756.69 766.69

+ week 1 755.07 767.07

+ type 1 756.06 768.06

+ elap30 1 756.43 768.43

+ field 1 756.66 768.66

> anova(forw.sel2)

Analysis of Deviance Table

Model: binomial, link: logit

Response: good

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev

NULL 1424 1013.43

distance 1 237.681 1423 775.75

PAT 1 13.335 1422 762.41

change 1 3.077 1421 759.33

wind 1 2.646 1420 756.69

We obtain a different set of variables! Remember that the BIC will generally favor smaller models than the AIC.

Comments:

* Investigate backward and alternating stepwise on your own. Note that the three stepwise selection methods will not always obtain the same “best” models.
* While AICc is not implemented in step(), why would we generally expect the same model with this data set as with using AIC?
* Alternatively to using step(), one could use the AIC() function and estimate each model in the stepwise process.
* Interactions can be included in the process by specifying a full model that includes them.

The LASSO

The least absolute shrinkage and selection operator (LASSO) was proposed by Tibshirani (1996) and has been further developed since. The basic idea is to add a penalty to the log likelihood function (different from Firth’s penalty) and then maximize it to obtain estimates. This penalty is chosen to help extenuate the effects of those explanatory variables that are truly important, while keeping parameter estimates close to 0 for those parameters that are not truly important. The model with the smallest residual deviance is considered to the “best”.

Model averaging

When a number of models have IC(k) values similar to each other, this is a sign that uncertainty exists about which model is “best”. By choosing only one of these models, some explanatory variables (and interactions) will be removed from consideration that were included in other models with similar IC(k) values. This is equivalent to assuming the estimated regression parameters and their corresponding variances are equal to 0!

A different approach is to use model averaging. Essentially, you use more than one model and “average” the estimated regression parameters across the models. In particular, Bayesian model averaging (BMA) uses Bayesian methods to estimate a probability that each model of interest is the “correct” model. These probabilities are easily approximated with the help of the BIC.

Define BIC0 as the smallest BIC obtained from all-subsets regression and define BICm as the BIC for the model of interest out of M possible models. The difference between these two values is defined as

Δm = BICm – BIC0

The estimated probability that the model is the “correct” model is



We will treat ≈ as an equality for the remainder of this section.

Comments:

* What is the numerator for the model with the smallest BIC?
* What happens to the probability as BIC increases?
* When Δm = 2, . Thus, the probability for this model is about 1/3 of that for the model with the smallest BIC. Notice that the glmulti() output earlier always reported the number of models that were within “2 units” of the smallest IC(k).
* The “probability” here is a “posterior probability”.
* Although BIC is used in the approximation, the same expression is sometimes used with other information criteria. In those cases, the term “evidence weights” is used rather than probabilities.

Suppose θ is our parameter of interest to estimate. This could be a regression parameter or a linear function of regression parameters like a log of an odds ratio. The model averaged estimate of θ is



Questions:

* Suppose one model clearly has a smaller BIC than the others. What will happen to ?
* Suppose many models have a BIC close to the smallest BIC. What will happen to ?
* For the last bullet, what would be a potential advantage to using BMA over other variable selection methods?

As a simple example, suppose θ = βj. Then



When a model does not include the explanatory variable corresponding to βj,  in the above expression.

The estimated variance of  is

****

Answer the questions above again, but now with respect to the estimated variance.

Comments:

* BMA results in non-zero estimates of regression parameters corresponding to all explanatory variables of interest.
* Explanatory variables that are truly not important are unlikely to appear in the high probability models; thus, their corresponding model averaged parameter estimates are likely to be close to 0 and they will not contribute much to the variability of the model-averaged estimator.
* Explanatory variables that are truly important are likely to appear in the high probability models; thus, their corresponding model averaged parameter estimates will be “shrunk” somewhat toward 0 due to those models that do not include the variable. This ends up reducing bias caused by the selection of variables (see Section 5.1.5 for a discussion).
* The BMA and glmulti packages in R can be used for BMA. Overall, there is no package that automatically performs all of the calculations desired. We will focus on the glmulti package because it does at least make some of these calculation automatic.

Example: Placekicking (BMAPlacekick.R, Placekick.csv)

I will use the BIC with glmulti() and only look among main effects:

> search.1.bic <- glmulti(y = good ~ ., data = placekick,

fitfunction = "glm", level = 1, method = "h", crit =

"bic", family = binomial(link = "logit"))

<OUTPUT EDITED>

> print(search.1.bic)

glmulti.analysis

Method: h / Fitting: glm / IC used: bic

Level: 1 / Marginality: FALSE

From 100 models:

Best IC: 784.195620372141

Best model:

[1] "good ~ 1 + distance + PAT"

Evidence weight: 0.656889992065609

Worst IC: 810.039529019811

1 models within 2 IC units.

9 models to reach 95% of evidence weight.

> head(weightable(search.1.bic))

model bic weights

1 good ~ 1 + distance + PAT 784.1956 0.65688999

2 good ~ 1 + distance + change + PAT 788.3802 0.08106306

3 good ~ 1 + distance + PAT + wind 788.7094 0.06875864

4 good ~ 1 + week + distance + PAT 789.6186 0.04364175

5 good ~ 1 + distance 790.2689 0.03152804

6 good ~ 1 + distance + PAT + type 791.3004 0.01882426

> plot(search.1.bic, type = "w")



Comments:

* Remember that the BIC tends to favor smaller models, and this is the case here in comparison to us using the AICc before.
* The weights column provides the approximation to .
* The plot gives the approximation to  vs. the ordered model number. The vertical line is drawn at a place where the cumulative probabilities reach 0.95. Thus, it takes 9 models to cumulate to 0.95.

Below is how the model averaged regression estimators are found:

> parms <- coef(search.1.bic)

> # Renaming columns to fit nicely

> colnames(parms) <- c("Estimate", "Variance", "n.Models",

"Probability", "95%CI +/-")

> round(parms, digits = 3)

Estimate Variance n.Models Probability 95%CI +/-

field 0.000 0.000 29 0.026 0.010

set 0.001 0.000 28 0.026 0.012

elap30 0.000 0.000 29 0.027 0.001

type 0.003 0.000 30 0.028 0.017

week -0.002 0.000 35 0.062 0.007

wind -0.051 0.010 35 0.095 0.197

change -0.042 0.007 39 0.119 0.159

PAT 1.252 0.191 62 0.944 0.857

(Intercept) 4.624 0.271 100 1.000 1.021

distance -0.088 0.000 100 1.000 0.024

> parms.ord <- parms[order(parms[,4], decreasing = TRUE),]

> ci.parms <- cbind(lower = parms.ord[,1] - parms.ord[,5], upper

= parms.ord[,1] + parms.ord[,5])

> round(cbind(parms.ord[,1], ci.parms), digits = 3)

lower upper

(Intercept) 4.626 3.609 5.644

distance -0.088 -0.111 -0.064

PAT 1.252 0.394 2.109

change -0.042 -0.201 0.117

wind -0.051 -0.248 0.147

week -0.002 -0.008 0.005

type 0.003 -0.015 0.020

elap30 0.000 -0.001 0.001

field 0.000 -0.011 0.010

> round(exp(cbind(OR = parms.ord[,1], ci.parms))[-1,],

digits = 2)

OR lower upper

distance 0.92 0.89 0.94

PAT 3.50 1.48 8.24

change 0.96 0.82 1.12

wind 0.95 0.78 1.16

week 1.00 0.99 1.01

type 1.00 0.99 1.02

elap30 1.00 1.00 1.00

field 1.00 0.99 1.01

Comments:

* Comparison to only using distance and PAT in a model:

> best.fit <- glm(formula = good ~ distance + PAT,

data = placekick, family = binomial(link = "logit"))

> round(summary(best.fit)$coefficients,2)

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.52 0.45 9.95 0

distance -0.09 0.01 -7.81 0

PAT 1.34 0.38 3.53 0

* t-distribution based intervals are calculated by default, where degrees of freedom are averaged across models; to change these to normal-based intervals, use icmethod = "standard" as an argument in coef()
* Please see the program for how to compute model-averaged estimates of the probability of success.

The approximations to  can also be used simply as a variable selection tool without calculating model-averaged estimates. Thus, just one overall model would result from applying this method, like we saw for all-subsets regression and stepwise selection methods.

For each explanatory variable, we sum the approximate probabilities associated with those models where the variable appears. Explanatory variables that are “important” to include will have large probabilities, and those that are “not important” will have small probabilities.

Some very general guidance for what is large and small comes from Raftery (1995):

* + < 0.5, it’s more likely the variable should not be in the model.
  + Between 0.5 and < 0.75, “weak” evidence
  + Between 0.75 and 0.95, “positive” evidence
  + Between 0.95 and 0.99, “strong” evidence
  + > 0.99, “very strong” evidence

While I do not include an example here, there is an example in Section 5.4.2.

**Section 5.2 – Tools to assess model fit**

There will be many similarities in this section to what you would learn about when assessing the fit of a normal linear regression model. If you have not done this before or need a refresher, please see my Section 10.1-10.3 lecture notes from STAT 870 at [www.chrisbilder.com/  
stat870/schedule.htm](http://www.chrisbilder.com/stat870/schedule.htm).

This was a difficult section for us to write for the following reasons:

* The generalized linear models (GLMs) of Chapters 2 and 4 largely fit within the same framework, so we wanted to discuss them at the same time. This led us to use a little different notation to encompass all of these types of GLMs.
* There are enough differences among the models of Chapters 2 and 4 that some differentiation is still needed at times.
* The multicategory models of Chapter 3 are not GLMs, but the goals to evaluate these models are largely the same as for GLMs. Unfortunately, evaluation tools are much less developed and not implemented in statistical software packages. This led us to briefly talk about these models in Section 5.2.4, separate from GLMs.

Therefore, the discussion next is for the GLMs presented in Chapters 2 and 4. Watch for the few notational changes.

Explanatory variable pattern (EVP) form

In the placekicking data set, each of the 1,425 rows represented one observation:

> placekick<-read.table(file = "C:\\data\\placekick.csv",

header = TRUE, sep = ",")

> head(placekick)

week distance change elap30 PAT type field wind good

1 1 21 1 24.7167 0 1 1 0 1

2 1 21 0 15.8500 0 1 1 0 1

3 1 20 0 0.4500 1 1 1 0 1

4 1 28 0 13.5500 0 1 1 0 1

5 1 20 0 21.8667 1 0 0 0 1

6 1 25 0 17.6833 0 0 0 0 1

The “good” variable represents a Bernoulli response – 0 for a failure and 1 for a success. When examining how well a model for this type of response fits the data, it is often better to examine the data in explanatory variable pattern (EVP) form. This format has one row in the data set for each of the unique sets of explanatory variables.

We saw EVP form already in Chapter 2 when we converted the placekick data set (involving ONLY distance and good) to a binomial response format:

> w<-aggregate(formula = good ~ distance, data = placekick,

FUN = sum)

> n<-aggregate(formula = good ~ distance, data = placekick,

FUN = length)

> w.n<-data.frame(distance = w$distance, success = w$good,

trials = n$good, proportion = round(w$good/n$good,4))

> head(w.n)

distance success trials proportion

1 18 2 3 0.6667

2 19 7 7 1.0000

3 20 776 789 0.9835

4 21 19 20 0.9500

5 22 12 14 0.8571

6 23 26 27 0.9630

> tail(w.n)

distance success trials proportion

38 55 2 3 0.6667

39 56 1 1 1.0000

40 59 1 1 1.0000

41 62 0 1 0.0000

42 63 0 1 0.0000

43 66 0 1 0.0000

For example, there are 20 observations with distance = 21 in the placekicking data set. Of the 20, 19 are successes. Remember that when we estimated a model to this form of the data, we obtained the same parameter estimates as with the original Bernoulli form of the data.

Why is EVP form better than the original form of the data?

Many of our statistical measures that examine how well the models of Chapter 2 fit the data are based on continuous distributional approximations. These approximations work better with the EVP format as you will soon see.

Before evaluating the fit of models from Chapter 2, we will always need to convert the data to EVP form first.

Residuals

Let y denote the observed count. Thus, this can be the usual count from Chapter 4, but also the number of successes for a binomial response.

A residual in its simplest form is

Observed value – Estimated value

The residuals for Chapters 2 and 4 are then

Chapter 2 models: The residual is y – n

Chapter 4 models: The residual is y – 

where these residuals are calculated for EVERY observation in a data set. For example, the residual for the placekicking data set at 21 yards is

19 – 20×0.9676 = –0.3519

where  using the model from Chapter 2.

Notational changes:

* In order to write these residuals without specifying the Chapter number, I will use  to denote the estimated value from the model. Thus,

Chapter 2 models: n will be denoted by 

Chapter 4 models:  will be denoted by 

The residual then simply becomes y – .

* There will be m = 1, …, M “observations” rather than i = 1, …, n observations in a data set. The reason for this notation is that it can get confusing with respect to the models of Chapter 2 due to the data perhaps being observed in a binary response format (like for the placekicking data). Thus, the residuals are



for m = 1, …, M. For the Chapter 2 models, there are nm Bernoulli responses or trials for each m = 1, …, M.

Pearson residuals

In order to determine if a residual is unusual in size (i.e., an outlier), we need to take into account the variability in the residual. This is partially done by calculating Pearson residuals:



where  is the estimated variance for Ym depending on the model:

Logistic –

Probit –

Poisson –

For the distance of 21 yards, there were nm = 20 trials and ym = 19 successes observed. The Pearson residual is then



The calculation of Pearson residuals is simple, but they have one problem – the denominator fails to take into account the variation in the estimated value given in the numerator. Thus, a standardized Pearson residual is



where hm come from the “hat matrix”. If you have had a full regression course, this matrix should be very familiar! If not (or if you need a refresher), below is an explanation of the hat matrix:

Let **X** be the matrix of the explanatory variable data values with 1’s in the first column. Create a diagonal matrix, , with diagonal elements of  in the same order as the corresponding data listed in **X**. The hat matrix is **H** = **X**(**X**′**X**)-1**X**′. Note that this is essentially the same hat matrix used when fitting a regression model by weighted least squares.

Both em and rm have “approximate” standard normal distributions. Thus, values of em and rm outside of  could be considered to be outliers. HOWEVER, this approximated does not always work well. This approximation is discussed next for the Chapters 2 and 4 models:

**Chapter 2 models** – We are essentially making a normal approximation for binomial random variables. This approximation depends on nm being large relative to probability of success (say, πm). For example, consider the case with an estimated probability of success  = 0.9 for an EVP where ym = 1 success out of nm = 3 trials is observed. The Pearson residual is:



Using a strict normal approximation, one would think this observation is being poorly fit by the model. However, note that there are ONLY 4 possible values for em:

> y < -0:3

> n <- 3

> pi.hat <- 0.9

> e <- round((y - n\*pi.hat)/sqrt(n\*pi.hat\*(1-pi.hat)),

2)

> prob.norm <- round(pnorm(q = e), 4)

> prob.bin <- round(pbinom(q = y, size = n, prob =

pi.hat), 4)

> data.frame(y, e, prob.norm, prob.bin)

y e prob.norm prob.bin

1 0 -5.20 0.0000 0.001

2 1 -3.27 0.0005 0.028

3 2 -1.35 0.0885 0.271

4 3 0.58 0.7190 1.000

Thus, em is a DISCRETE random variable, and we used a CONTINUOUS random variable to make a judgment about it. The normal distribution approximation should not be expected to do well! Note that it is not just nm that one needs to examine. For example, if nm = 100 and  = 0.99, one can see that there are few likely possible values for ym in this case, so the normal approximation would likely not work well here too.

**Chapter 4 models** – We are essentially making a normal approximation for Poisson random variables. For this approximation to work,  can not be extremely small (say, need  > 0.5). Note that when  is extremely small, then one will expect to observe ym values that are equal to 0 or 1.

Because of these normal distribution approximation problems, I recommend being careful with making judgments about observations. Using simple guidelines, like em or rm being outside of ±2 or ±3, can be used to provide an initial impression about whether an observation is an outlier. One should then follow-up with a closer look at quantities like nm to make a more informed judgment.

Comments:

* The standardized Pearson residuals are better than the Pearson residuals to examine. I discuss Pearson residuals here because they lead nicely into the standardized version. Still, there may be cases where Pearson residuals are much easier to calculate (outside of models in Chapters 2 and 4), so this is another reason why they are introduced here.
* In what situation would you expect nm ≈ 1 for m = 1, …, M? How does this affect the use of em and rm to judge the model’s fit?
* The residual deviance can be written as  where dm is the specific contribution by the mth observation. The deviance residual is then based on dm:

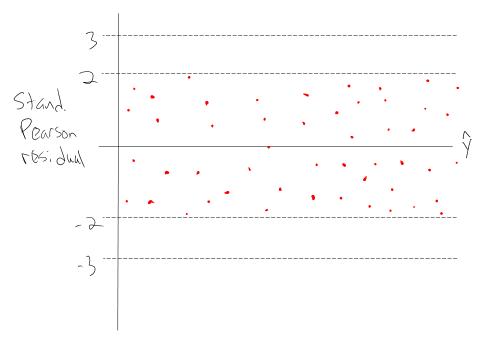


Thus, . The standardized deviance residual is . Please see Section 5.2.1 of the book for more details.

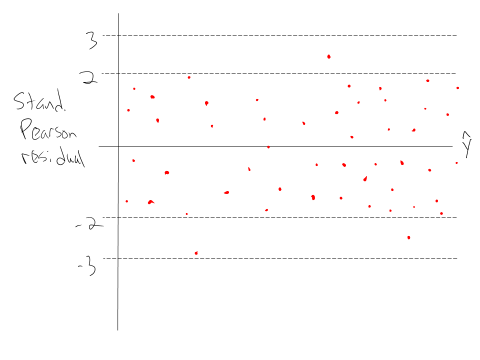
* The residuals() function with type = "pearson" calculates Pearson residuals.
* The rstandard() function with type = "pearson" calculates the standardized Pearson residuals.
* Suppose a normal distribution approximation does actually work and M = 100. Approximately how many standardized residuals would you expect outside of ±2?
* Generally, if the model is o.k., one should expect to see a random scattering of the points in a plot of the standardized Pearson residuals vs. an important component of the model, such as an explanatory variable or . If instead there is some type of trend, this suggests there is a problem with the model, such as perhaps a transformation is needed of an explanatory variable. One just needs to be careful with looking for trend, and more will be discussed about this in the examples.

Example: Plots with about 50 points plotted on each

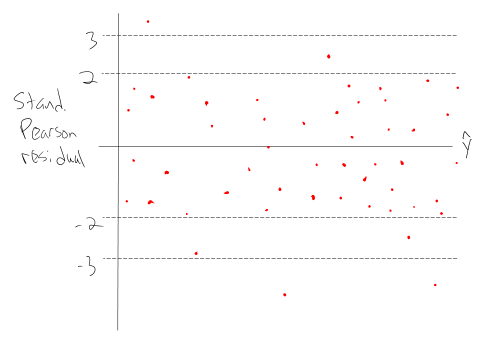
Random scattering of points, everything within ±2, assuming normal approximation works



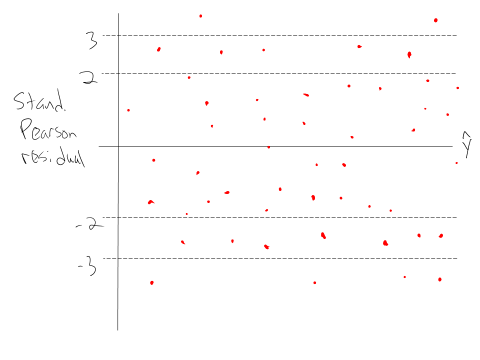
Random scattering of points, everything within ±3, assuming normal approximation works



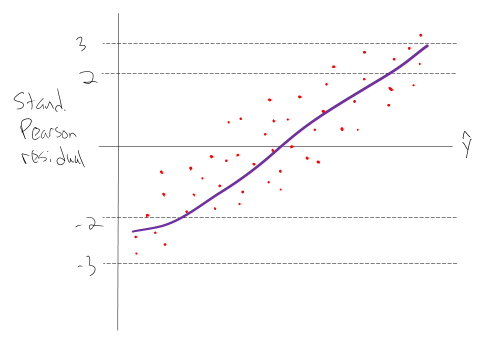
Random scattering of points, a few are outside of ±3, assuming normal approximation works



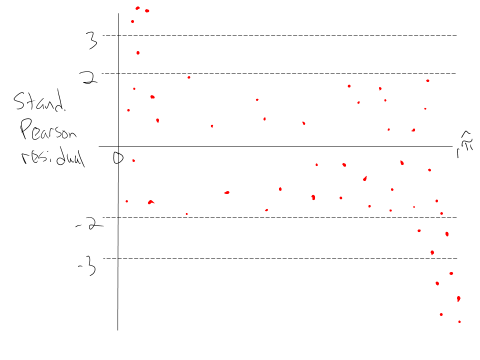
Random scattering of points, many more points than expected outside of ±2, assuming normal approximation works



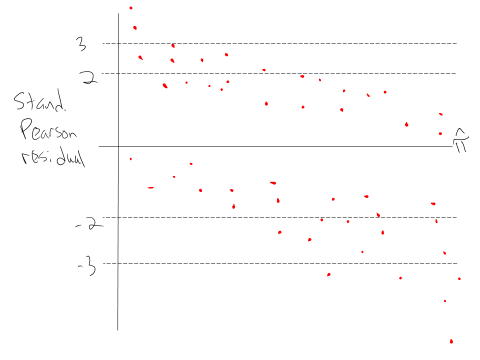
Trend in points (could also plot an explanatory variable on the x-axis)



Chapter 2 models: Points outside of ±3 when  is close to 0 and 1



Chapter 2 models: Purely binary responses (M = n); notice separation between points (due to positive and negative responses)



Example: Placekicking (PlacekickDiagnostics.R, Placekick.csv)

Consider the logistic regression model with distance as the only explanatory variable.

> mod.fit.bin <- glm(formula = success/trials ~ distance,

weights = trials, family = binomial(link = logit), data

= w.n)

> pi.hat <- predict(mod.fit.bin, type = "response")

> p.res <- residuals(mod.fit.bin, type = "pearson")

> s.res <- rstandard(mod.fit.bin, type = "pearson")

> lin.pred <- mod.fit.bin$linear.predictors

> w.n <- data.frame(w.n, pi.hat, p.res, s.res, lin.pred)

> round(head(w.n), digits = 3)

distance success trials prop pi.hat p.res s.res lin.pred

1 18 2 3 0.667 0.977 -3.571 -3.575 3.742

2 19 7 7 1.000 0.974 0.432 0.433 3.627

3 20 776 789 0.984 0.971 2.094 3.628 3.512

4 21 19 20 0.950 0.968 -0.444 -0.448 3.397

5 22 12 14 0.857 0.964 -2.136 -2.149 3.281

6 23 26 27 0.963 0.960 0.090 0.091 3.166

> # Standardized Pearson residual vs X plot

> plot(x = w.n$distance, y = w.n$s.res, xlab = "Distance",

ylab = "Standardized Pearson residuals", main =

"Standardized residuals vs. \n X")

> abline(h = c(3, 2, 0, -2, -3), lty = 3, col = "blue")

> # Add loess model to help visualize trend

> smooth.stand <- loess(formula = s.res ~ distance, data =

w.n, weights = trials)

> # Make sure that loess estimates are ordered by "X" for

the plots, so that they are displayed properly

> order.dist <- order(w.n$distance)

> lines(x = w.n$distance[order.dist], y =

predict(smooth.stand)[order.dist], lty = 3, col =

"red", lwd = 3)

See the first plot on the left corresponding to the code above.



Additional code in the program produces the other two plots. Because distance is the only variable in the model and it has a negative coefficient, the plot of the residuals against the linear predictor is just a mirror image of the plot against distance.

Comments:

* The residual at 18 yards is outside of the ±3 lines. I am not concerned about this because there are only 3 placekicks attempted from this distance (2 out of 3 were successes). Thus, a normal approximation would not be accurate. In fact, the only way this residual would not be outside of the ±3 lines is if all placekicks were successes! With binary outcomes, there are simply going to be examples like this where a failure occurs despite a high probability of success.
* The residual at 20 yards is also outside of the ±3 lines. In this case, there are 776 successes out of 789 trials (776/789 = 0.984) when the estimated probability of success is 0.971. How unusual is an occurrence like this? Use a binomial distribution to help make a judgment:



where I used

> 1-pbinom(q = 775, size = 789, prob = 0.971)

[1] 0.01720304

in R. Due to the small probability, I am somewhat concerned about this residual. Taking into my knowledge about football, I am even more concerned because there are two different types of placekicks typically observed at 20 yards – field goals and PATs. Perhaps another variable needs to be added to the model to account for this? More on this in Section 5.4!

* There are a few other standardized Pearson residuals outside of ±2, but within ±3. Are these observations of concern?
* If you are not familiar with loess regression models, they provide a way to determine trend in a scatter plot. Chapter 11 of my UNL STAT 870 course lecture notes discuss these models in detail. One important detail of their application is that predicted values from loess models can have a large amount of variability where data are sparse or near the extreme values of the variable on the x-axis. For our plots here, I focus on the middle of the plots and see some upward trend, suggesting there may be some problems.
* If there were not potential problems with a model, where would you expect the loess curve to be on a plot?

Goodness of fit (GOF)

Residual measures examine how well a model fits an individual observation. GOF statistics evaluate the model for all observations at once.

Two commonly used GOF statistics are the Pearson and residual deviance statistics:

Pearson statistic: 

Residual deviance statistic: D = -2log(Λ) for comparing the model of interest to the saturated model

Questions:

* Why do you think these statistics would be useful to examine?
* The statistics are used for a GOF hypothesis test. What are the hypotheses in words, rather than symbols?
* What types of values for X2 and D would lead to a rejection of the null hypothesis?

These statistics have approximate  distributions for “large” samples, where  denotes the number of regression parameters estimated in the model of interest. However, this result is only valid under the assumption that the number of unique sets of explanatory variable combinations in the data is fixed as the sample size increases.

As an alternative, we recommend calculating “Deviance/df”, i.e., . If the model is reasonable, this numerical value should not be too far from 1. Why?

How far away from 1 is still o.k.? General guidelines are

* Potential problem: 
* Poor fit: 

There are other measures of goodness-of-fit too. Please see Section 5.2.2 of my book.

Influence

An observation is influential to a model’s fit if removing or changing the observation results in a significant change in the parameter estimates or estimated response values (). There are a number of influence measures that can be calculated for an observation. They all focus on removing the observation from the data set and then examining how a particular statistic changes. Because removing each observation one at a time and re-estimating a model can be time consuming, approximations have been developed for these measures to avoid needing to re-estimate models.

Cook’s distance – This is an overall measure of how all parameter estimates would change if the mth observation is removed from the data set. Cook’s distance is



and it is always greater than 0. Those observations with CDm values much larger than the rest should be investigated further for potentially being influential. Others use the general guideline of CDm > 1 may have high influence and CDm > 4/M may have moderate influence.

Change in Pearson GOF statistic – This is a measure of how much X2 will change if the mth observation is removed from the data set. The “delta X-squared” statistic is



The statistic is just the square of the standardized Pearson statistic! A  approximation can be used with the statistic, but with the same caveats as discussed earlier for rm and a standard normal distribution approximation. For this reason, I recommend simply looking at  > 4 and  > 9 as thresholds.

Change in the residual deviance statistic – This is a measure of how much the residual deviance will change if the mth observation is removed from the data set. The “delta D” statistic is



The same approximation as used with  can be used here.

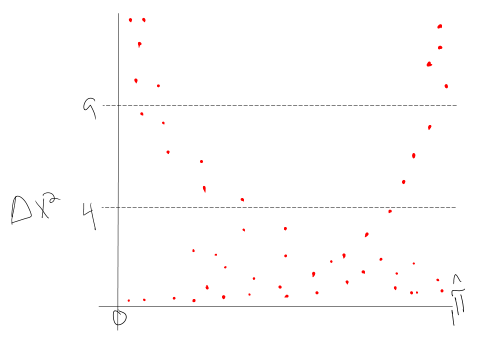
Hat matrix diagonal value (leverage) – This measure alone is sometimes used with normal linear regression models to identify outlying observations (in terms of their explanatory variable values), which may subsequently be influential. For GLMs, interpreting the hm becomes more difficult because a large value may not mean being outlying with respect to the explanatory variables (see Section 5.2.3). Overall, some individuals use the guidelines that values larger than 2p/M indicate “moderately high leverage,” and values larger than 3p/M indicate high leverage. Note that having a high amount of leverage is not necessarily bad. This is why I focus on this measure MUCH less than the others.

Calculation in R – Use the model fit object obtained from glm() with the following functions:

* hatvalues() calculates hm
* cooks.distance() calculates CDm
* rstandard() calculates  which then can be squared for 
* rstandard() calculates  (type = "deviance") which subsequently leads to the calculation of 

Section 5.2 discusses a general function written for the book named glmInflDiag() which automatically calculates these measures and produces plots. This function works well for GLMs in general. Section 5.4 discusses an alternative function specifically for logistic regression models named examine.logistic.reg(). We will use this function shortly.

Example: Plot with about 50 points plotted



Example: Placekicking (Placekick\_NotInBook.R, Examine.logistic.reg.R, Placekick.csv)

> placekick <- read.table(file =

"C:\\data\\placekick.csv", header = TRUE, sep = ",")

> w <- aggregate(formula = good ~ distance, data =

placekick, FUN = sum)

> n <- aggregate(formula = good ~ distance, data =

placekick, FUN = length)

> w.n <- data.frame(distance = w$distance, success =

w$good, trials = n$good, prop = round(w$good/n$good,4))

> mod.fit.bin <- glm(formula = success/trials ~ distance,

weights = trials, family = binomial(link = logit), data

= w.n)

> # summary(mod.fit.bin)

> source(file = "C:\\Rprograms\\Examine.logistic.reg.R")

> save.info1 <- examine.logistic.reg(mod.fit.obj =

mod.fit.bin, identify.points = TRUE)



Plots:

* (1,1): This is the same plot as we saw before
* (1,2): Observation #3 is EXTREME on the plot! Why do you think this occurs?
* (2,1): The y-axis numerical values are the squares of the y-axis numerical values on the (1,1) plot. What makes this plot helpful then is that the size of the plotting symbol is proportional to nm.
* (2,2): The same plot as in (2,1), but now the plotting symbol is proportional in size to Cook’s distance.
* Deviance/df is given below the plot

What do you think of the model?

When there are a few observations with nm or CDm values that are much larger than the rest, this can make examining the size of the plotting symbol not helpful for the other observations. For this reason, I have included a way to rescale the numerical values used as the plotting symbol size. Below are some examples.

> save.info2 <- examine.logistic.reg(mod.fit.obj =

mod.fit.bin, identify.points = TRUE, scale.n = sqrt,

scale.cookd = sqrt)



> one.fourth.root <- function(x) {

x^0.25

}

> one.fourth.root(16) # Example

[1] 2

> save.info3 <- examine.logistic.reg(mod.fit.obj =

mod.fit.bin, identify.points = TRUE, scale.n =

one.fourth.root, scale.cookd = one.fourth.root)



For those observations identified in the plots, it is often helpful to list out all their explanatory variable and response values with their residual and influence measures. Because there are not too many observations, I did this for all observations:

> data.frame(w.n[,-4], prop = round(w.n[,4],2), pi.hat =

round(mod.fit.bin$fitted.values, 2), std.Pear =

round(save.info1$stand.resid,2), deltaXsq =

round(save.info1$deltaXsq,2), cookd =

round(save.info1$cookd,2))

distance success trials prop pi.hat std.Pear deltaXsq cookd

1 18 2 3 0.67 0.98 -3.58 12.78 0.02

2 19 7 7 1.00 0.97 0.43 0.19 0.00

3 20 776 789 0.98 0.97 3.63 13.16 13.16

4 21 19 20 0.95 0.97 -0.45 0.20 0.00

5 22 12 14 0.86 0.96 -2.15 4.62 0.03

6 23 26 27 0.96 0.96 0.09 0.01 0.00

7 24 7 7 1.00 0.95 0.58 0.33 0.00

8 25 12 13 0.92 0.95 -0.44 0.19 0.00

9 26 8 9 0.89 0.94 -0.72 0.52 0.00

10 27 24 29 0.83 0.94 -2.48 6.13 0.09

11 28 20 22 0.91 0.93 -0.39 0.16 0.00

12 29 16 17 0.94 0.92 0.29 0.08 0.00

13 30 12 14 0.86 0.91 -0.76 0.58 0.00

14 31 10 11 0.91 0.90 0.05 0.00 0.00

15 32 23 30 0.77 0.89 -2.30 5.28 0.08

16 33 20 21 0.95 0.88 1.01 1.01 0.01

17 34 16 19 0.84 0.87 -0.37 0.13 0.00

18 35 12 14 0.86 0.86 0.01 0.00 0.00

19 36 18 22 0.82 0.84 -0.31 0.09 0.00

20 37 22 29 0.76 0.83 -0.97 0.94 0.02

21 38 23 28 0.82 0.81 0.18 0.03 0.00

22 39 22 28 0.79 0.79 -0.06 0.00 0.00

23 40 13 19 0.68 0.77 -0.91 0.83 0.01

24 41 7 10 0.70 0.75 -0.36 0.13 0.00

25 42 20 27 0.74 0.73 0.16 0.03 0.00

26 43 12 22 0.55 0.70 -1.67 2.79 0.07

27 44 13 18 0.72 0.68 0.40 0.16 0.00

28 45 11 18 0.61 0.65 -0.39 0.15 0.00

29 46 11 14 0.79 0.63 1.25 1.57 0.04

30 47 14 23 0.61 0.60 0.09 0.01 0.00

31 48 10 18 0.56 0.57 -0.15 0.02 0.00

32 49 8 12 0.67 0.54 0.88 0.77 0.02

33 50 10 19 0.53 0.52 0.10 0.01 0.00

34 51 11 15 0.73 0.49 2.00 3.98 0.17

35 52 5 13 0.38 0.46 -0.55 0.30 0.01

36 53 5 9 0.56 0.43 0.79 0.62 0.02

37 54 1 7 0.14 0.40 -1.43 2.04 0.05

38 55 2 3 0.67 0.37 1.06 1.12 0.01

39 56 1 1 1.00 0.35 1.37 1.89 0.01

40 59 1 1 1.00 0.27 1.63 2.67 0.01

41 62 0 1 0.00 0.21 -0.52 0.27 0.00

42 63 0 1 0.00 0.19 -0.49 0.24 0.00

43 66 0 1 0.00 0.14 -0.41 0.17 0.00

When there is only one explanatory variable, another good plot to examine is what we saw in Chapter 2:







Remember that the very large distance placekicks do not have many observations.

**Section 5.3 – Overdispersion**

We will focus on count responses in this section and briefly mention other types of responses at the end.

A limiting assumption for a Poisson distribution is that E(Y) = Var(Y) = μ. Sometimes, the sample variance of Y appears to be greater than the sample mean of Y for a data set. Evidence of this occurs in the horseshoe example. Below is a table that can be used as an ad-hoc assessment (Horseshoe\_Ch5.R):

> crab<-read.table(file = "c:\\horseshoe.txt", header =

TRUE)

> head(crab)

> groups<-ifelse(test = crab$width<23.25, yes = 1, no =

ifelse(test = crab$width<24.25, yes = 2, no =

ifelse(test = crab$width<25.25, yes = 3, no =

ifelse(test = crab$width<26.25, yes = 4, no =

ifelse(test = crab$width<27.25, yes = 5, no =

ifelse(test = crab$width<28.25, yes = 6, no =

ifelse(test = crab$width<29.25, yes = 7, no =

8)))))))

> crab.group<-data.frame(crab,groups)

> head(crab.group)

satellite width groups

1 8 28.3 7

2 0 22.5 1

3 9 26.0 4

4 0 24.8 3

5 4 26.0 4

6 0 23.8 2

> ybar<-aggregate(formula = satellite ~ groups, data =

crab, FUN = mean)

> var.y<-aggregate(formula = satellite ~ groups, data =

crab, FUN = var)

> group.name<-c("width <23.25", "23.25 <= width < 24.25",

"24.25 <= width < 25.25", "25.25 <= width < 26.25",

"26.25 <= width < 27.25", "27.25 <= width < 28.25",

"28.25 <= width < 29.25","width > 29.25")

> data.frame(group.name, mean.sat =round(ybar$satellite,2),

var.sat = round(var.y$satellite,2))

group.name mean.sat var.sat



1 width <23.25 1.00 2.77

2 23.25 <= width < 24.25 1.43 8.88

3 24.25 <= width < 25.25 2.39 6.54

4 25.25 <= width < 26.25 2.69 11.38

5 26.25 <= width < 27.25 2.86 6.89

6 27.25 <= width < 28.25 3.88 8.81

7 28.25 <= width < 29.25 3.94 16.88

8 width > 29.25 5.14 8.29

If the Poisson assumptions were satisfied, we would expect the mean.sat column to be approximately the same as the var.sat column. Obviously, this does not occur.

When the variance is larger than the mean, this is called overdispersion, and it is a violation of our model. Thus, inferences made using the model may be incorrect. What can you do when this occurs?

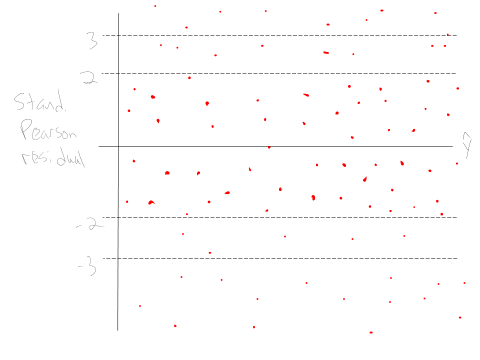
1. Find more explanatory variables that help explain the variability in the response variable! The additional variability could be due to not accounting for other explanatory variables. For example, perhaps crab weight plays an important role in estimating the mean number of satellites. By using width only in a model, there may be additional satellite variability at individual widths due to the omission of weight.
2. Check if it is a zero inflation problem.
3. Use Poisson generalized linear mixed models. These models are helpful if there is an identifiable cause for overdispersion, such as observations are collected over time on the same individuals. Please see Section 6.5.
4. Use quasi-Poisson regression models. These models do not assume a full parametric form for the model.
5. Use negative binomial regression models.

Options 4 and 5 should only be used when there is not an identifiable or measurable cause for the overdispersion.

Detection of overdispersion

We previously compared estimates of the mean counts to the variance of the counts to determine if overdispersion exists. This cannot always be done due there being more than one explanatory variable. Other approaches are:

1. Poor overall model fit as determined by deviance/df
2. More extreme standardized Pearson residuals than expected; this can be seen by examining a plot where overdispersion is often exhibited by a fairly uniform trend of extreme standardized Pearson values across the plot



Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.txt)

> mod.fit <- glm(formula = satellite ~ width, data = crab,

family = poisson(link = log))

> summary(mod.fit)

Call:

glm(formula = satellite ~ width, family = poisson(link = log), data = crab)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8526 -1.9884 -0.4933 1.0970 4.9221

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -3.30476 0.54224 -6.095 1.1e-09 \*\*\*

width 0.16405 0.01997 8.216 < 2e-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: 567.88 on 171 degrees of freedom

AIC: 927.18

Number of Fisher Scoring iterations: 6

> mod.fit$deviance / mod.fit$df.residual

[1] 3.320927

> round(c(1 + 2\*sqrt(2/mod.fit$df.residual), 1 +

3\*sqrt(2/mod.fit$df.residual)), 2)

[1] 1.22 1.32

Poor model fit is indicated by a large deviance/df.

> mu.hat <- mod.fit$fitted.values

> stand.resid <- rstandard(model = mod.fit, type =

"pearson")

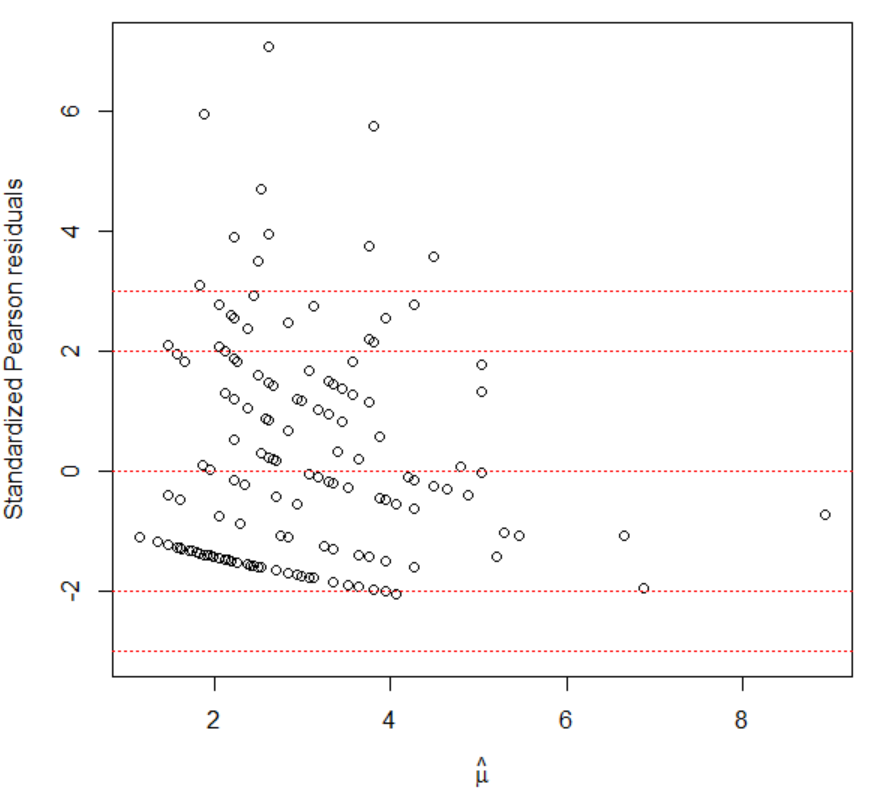
> plot(x = mu.hat, y = stand.resid, xlab =

expression(hat(mu)), ylab = "Standardized Pearson

residuals", ylim = c(min(c(-3, stand.resid)), max(c(3,

stand.resid))))

> abline(h = c(-3,-2,0,2,3), lty = "dotted", col = "red")



> sum(abs(stand.resid) > 3)

[1] 10

> length(stand.resid)

[1] 173

> sum(abs(stand.resid) > 3) / length(stand.resid)

[1] 0.05780347

> length(stand.resid)\*2\*(1-pnorm(q = 3)) #Expected number of

extreme residuals if N(0,1) works

[1] 0.4670647

The expected number of extreme residuals is larger than expected when using a standard normal approximation. Also, some residuals are MUCH larger than one would expect. Therefore, overdispersion appears to exist when only having width available as an explanatory variable.

Quasi-Poisson regression models

In order to find the maximum likelihood estimates for β0, …, βp we need



for r = 0, …, p. These equations are set equal to 0 and solved for β0, …, βp. In general, equations that are set equal to 0 and then solved for parameter estimates are called “estimating equations.” This makes sense here because our “estimates” result from these “equations.”

Wedderburn (1974) suggested using quasi-likelihood methods to find parameter estimates for a generalized linear model. In this setting, one assumes a particular relationship between the mean and variance, BUT no particular distribution for the response variable. In the count data situation, we can proceed in a similar manner as in Chapter 4, but with some adjustments. Let the relationship between the mean and variance be

Var(Y) = γE(Y)

for some constant γ, but do not assume Y has a Poisson distribution. Notice that when γ > 1, we would have overdispersion for the regular Poisson distribution. Models of this type are called quasi-Poisson.

The estimating equations used for Poisson regression simplify to



for r = 0, …, p, where there are m = 1, …, M observations, xmr is the mth observed value for the rth explanatory variable, and μm = exp(β0 + β1xm1 + + βpxmp). The quasi-likelihood approach uses Var(Ym) = γμm leading to the estimating equations of



for r = 0, …, p.

What happens when you estimate the parameters with this set of estimating equations?

1. Notice what happens when the above equation is set equal to 0: γ does not play a role. The exact same estimates for  as with Poisson regression will result.
2. The estimated covariance matrix for  is the same as with the Poisson regression model except it is multiplied by γ.
3. Diagnostic measures, like standardized residuals, are appropriately adjusted. For example,



for the ith observation.

1. Because γ is a parameter, it needs to be estimated as well. The standard approach is to use  = X2/(m-) where X2 is the Pearson statistic from the regular likelihood approach model.
2. The -2log(Λ) statistic for a LRT is divided by  to account for the overdispersion. The distribution of this new statistic is approximated by an F-distribution. The numerator degrees of freedom is the number of parameters testing (q) and the denominator degrees of freedom is the sample size minus the total number of parameters in full model (M – ). Similarly, profile LR intervals are found using the F-distribution rather than the chi-square distribution.
3. The AIC can no longer be used because likelihood methods are not being used. Instead, the QAIC is used to compare quasi-Poisson regression models. This statistic is the same as the AIC, but now with the likelihood function divided by . Do not compare a QAIC to an AIC from a Poisson regression model. Also, you should use the same value of  when comparing different quasi-Poisson regression models. Please see p. 313 for a discussion.

Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.txt)

The model is estimated by glm() again, but now the family option has changed to quasipoisson(link = log).

> mod.fit.quasi <- glm(formula = satellite ~ width, data =

crab, family = quasipoisson(link = log))

> summary(mod.fit.quasi)

Call:

glm(formula = satellite ~ width, family = quasipoisson(link = log), data = crab)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8526 -1.9884 -0.4933 1.0970 4.9221

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -3.30476 0.96729 -3.417 0.000793 \*\*\*

width 0.16405 0.03562 4.606 7.99e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 3.182205)

Null deviance: 632.79 on 172 degrees of freedom

Residual deviance: 567.88 on 171 degrees of freedom

AIC: NA

Number of Fisher Scoring iterations: 6

> vcov(mod.fit.quasi)

(Intercept) width

(Intercept) 0.93565064 -0.034334475

width -0.03433447 0.001268475

> vcov(mod.fit) \* sum.fit.quasi$dispersion

(Intercept) width

(Intercept) 0.93565064 -0.034334475

width -0.03433447 0.001268475

> anova(mod.fit.quasi, test = "F")

Analysis of Deviance Table

Model: quasipoisson, link: log

Response: satellite

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev F Pr(>F)

NULL 172 632.79

width 1 64.913 171 567.88 20.399 1.168e-05 \*\*\*

---

> Anova(mod.fit.quasi, test.statistic = "F")

Analysis of Deviance Table (Type II tests)

Response: satellite

SS Df F Pr(>F)

width 64.91 1 20.399 1.168e-05 \*\*\*

Residuals 544.16 171

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> #Wald interval with c = 1

> beta.ci <- confint.default(object = mod.fit.quasi, parm =

"width", level = 0.95)

> beta.ci

2.5 % 97.5 %

width 0.09423969 0.2338505

> 100\*(exp(beta.ci) - 1)

2.5 % 97.5 %

width 9.882309 26.34556

> #Profile likelihood ratio interval

> beta.ci <- confint(object = mod.fit.quasi, parm =

"width", level = 0.95)

Waiting for profiling to be done...

> beta.ci

2.5 % 97.5 %

0.09363759 0.23324826

> 100\*(exp(beta.ci) - 1)

2.5 % 97.5 %

9.816169 26.269491

The estimated model is .

Notice the output here is exactly the same as what we had for the regular Poisson regression model except for a few differences:

1. The “dispersion parameter” is the estimate of γ, say , and it is given to be 3.18.
2. The estimated standard deviations for the model parameter estimates are  = 0.96731 and  = 0.03562. The Poisson regression model had values of  = 0.5422 and  = 0.01996. Notice that

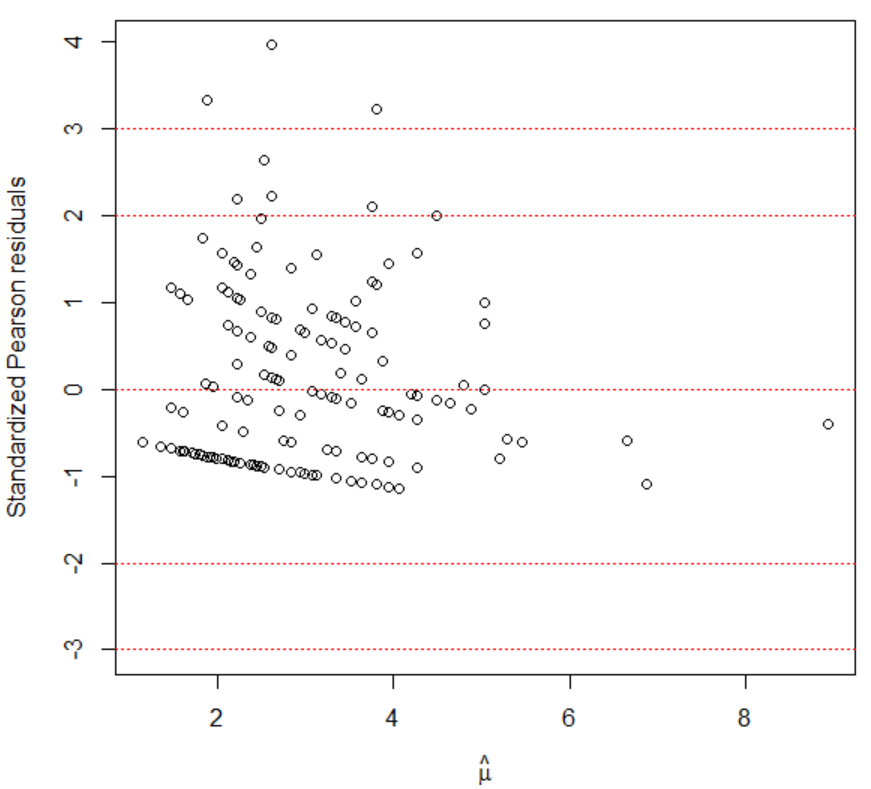
> 0.54222 \* sqrt(3.18)

[1] 0.9669168

> 0.01996 \* sqrt(3.18)

[1] 0.03559378

1. With the larger standard errors, the Wald statistics are smaller and the p-values larger.
2. A LRT for H0: β1 = 0 vs. Ha: β1 ≠ 0 with an F-distribution approximation results in a small p-value indicating that width is an important variable.
3. The 95% confidence intervals for PC are wider here than when using the Poisson model in Chapter 4. For example, the 95% profile LR interval was (13.28, 22.51) with a Poisson model.
4. There is no AIC listed because this is not a full likelihood method.
5. There are much fewer extreme residuals with this model than with the Poisson model:



Negative binomial regression models

One way to write the negative binomial distribution is

 for y = 0, 1, …

This distribution occurs when one is interested in the probability of y failures before the kth success (see Casella and Berger (2002, p. 95) if you are interested in more detail). For us, there are two important aspects to this distribution. First, the values of Y are non-negative integers just like a Poisson random variable. Second, the distribution can be rewritten as

 for y = 0, 1, …, and k>0

where E(Y) = μ and Var(Y) = μ + μ2/k. Contrast the relationship between Var(Y) and E(Y) here with what was used for the quasi-Poisson regression models.

As k0, we have more overdispersion. As k∞, we have E(Y) = Var(Y) = μ; thus, the same mean and variance as in the Poisson case. More in-depth information on how the negative binomial distribution comes about for count data is available on p. 553 of Agresti (2013).

Maximum likelihood estimation is used to estimate the regression parameters and k. Therefore, Wald and likelihood-based methods can be used for inferences. Information criteria can be used to choose among models.

Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.txt)

The glm() function cannot estimate a negative binomial regression model, so we will use the glm.nb() function from the MASS package:

> library(package = MASS)

> mod.fit.nb <- glm.nb(formula = satellite ~ width, data =

crab, link = log)

> summary(mod.fit.nb)

Call:

glm.nb(formula = satellite ~ width, data = crab, link = log, init.theta = 0.90456808)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.7798 -1.4110 -0.2502 0.4770 2.0177

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -4.05251 1.17143 -3.459 0.000541 \*\*\*

width 0.19207 0.04406 4.360 1.3e-05 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(0.9046) family taken to be 1)

Null deviance: 213.05 on 172 degrees of freedom

Residual deviance: 195.81 on 171 degrees of freedom

AIC: 757.29

Number of Fisher Scoring iterations: 1

Theta: 0.905

Std. Err.: 0.161

2 x log-likelihood: -751.291

> class(mod.fit.nb)



[1] "negbin" "glm" "lm"

The estimated negative binomial regression model is



R refers to the “dispersion parameter” estimate as  = 0.905, which is 1/. Thus,  = 1/0.905 = 1.1, and the estimated variance of Y is now  (remember that  depends on the value of x). This extra variability then shows up in the estimated variances for  and :

> vcov(mod.fit.nb)

(Intercept) width

(Intercept) 1.37225388 -0.051449169

width -0.05144917 0.001941126

> vcov(mod.fit.quasi)

(Intercept) width

(Intercept) 0.93565064 -0.034334475

width -0.03433447 0.001268475

> vcov(mod.fit)

(Intercept) width

(Intercept) 0.29402590 -0.0107895239

width -0.01078952 0.0003986151

> sum.fit.nb <- summary(mod.fit.nb)

> sum.fit <- summary(mod.fit)

> std.err <- data.frame(Poisson = sum.fit$coefficients[,2],

quasi = sum.fit.quasi$coefficients[,2],

nb = sum.fit.nb$coefficients[,2])

> std.err

Poisson quasi nb

(Intercept) 0.54224155 0.96729036 1.17143241

width 0.01996535 0.03561565 0.04405821

What affect will this have on inferences using the model?

Remember that as k∞, we have LESS overdispersion. Thus, as θ∞, we have more overdispersion, where θ = 0 corresponds to no overdispersion. Using this information and the standard error for  given in the output, we have an informal way to assess if overdispersion exists. Note that 0.905 – 2×0.161 > 0 so it appears that overdispersion exists (3 standard errors results in same conclusion).

Notes:

* LRTs are performed using anova() and Anova().
* Confidence intervals for β1 can be found through confint() and confint.default() – please see program
* Confidence intervals for μ can be found through predict(). For example, using the ci.mu() function from Chapter 4, below are the intervals for width = 23 and 33 from the three regression models:

> nb23 <- ci.mu(newdata = data.frame(width = 23),

mod.fit.obj = mod.fit.nb, alpha = 0.05)

> quasi23 <- ci.mu(newdata = data.frame(width = 23),

mod.fit.obj = mod.fit.quasi, alpha = 0.05)

> Pois23 <- ci.mu(newdata = data.frame(width = 23),

mod.fit.obj = mod.fit, alpha = 0.05)

> data.frame(type = c("Negative binomial", "Quasi-

Poisson", "Poisson"),

lower = round(c(nb23$lower, quasi23$lower,

Pois23$lower),2),

upper = round(c(nb23$upper, quasi23$upper,

Pois23$upper),2))

type lower upper

1 Negative binomial 1.01 2.05

2 Quasi-Poisson 1.16 2.21

3 Poisson 1.33 1.92

> nb33 <- ci.mu(newdata = data.frame(width = 33),

mod.fit.obj = mod.fit.nb, alpha = 0.05)

> quasi33 <- ci.mu(newdata = data.frame(width = 33),

mod.fit.obj = mod.fit.quasi, alpha = 0.05)

> Pois33 <- ci.mu(newdata = data.frame(width = 33),

mod.fit.obj = mod.fit, alpha = 0.05)

> data.frame(type = c("Negative binomial", "Quasi-

Poisson", "Poisson"),

lower = round(c(nb33$lower, quasi33$lower,

Pois33$lower),2),

upper = round(c(nb33$upper, quasi33$upper,

Pois33$upper),2))

type lower upper

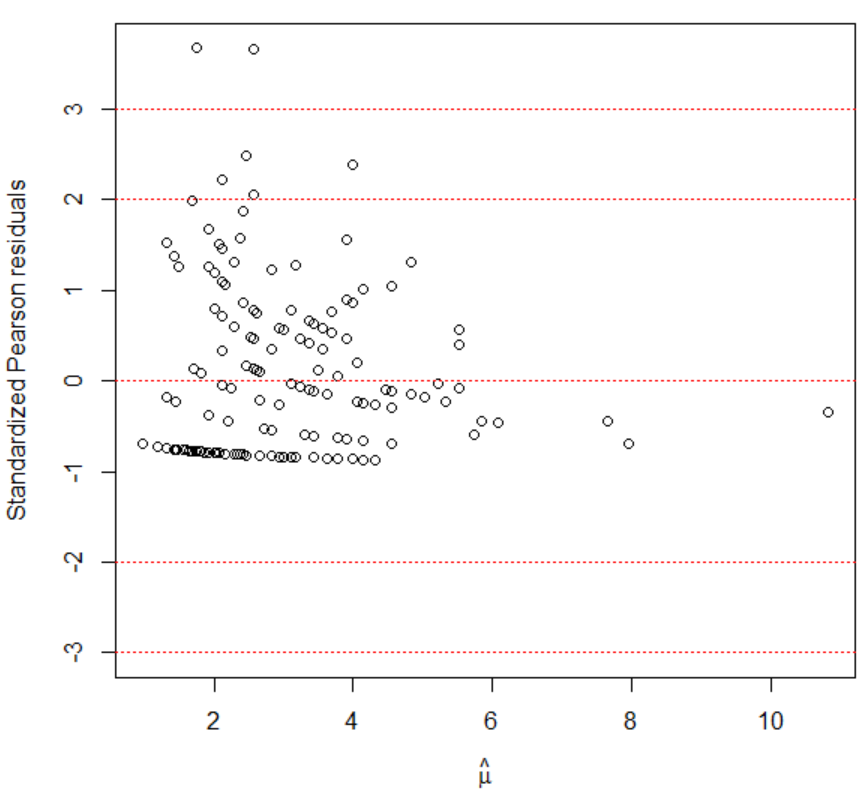
1 Negative binomial 5.45 17.73

2 Quasi-Poisson 5.29 12.82

3 Poisson 6.43 10.56

The intervals for the negative binomial model are shorter for the smaller width and longer for the larger width than the quasi-Poisson model. This occurs due to how the two models take into account the overdispersion (see equations for Var(Y)).

* There are much fewer extreme residuals with this model than with the Poisson model:



Choosing between the options to handle overdispersion

If it is clear why there is extra variability, take it into account! For example, include an additional explanatory variable. Also, if the data is observed in clusters, such as at time = 1, 2, and 3 an observation is observed for the same individual, then use a generalized linear mixed model.

If it is not clear why there is extra variability, the quasi-Poisson and negative binomial models provide viable options. To choose between the two, you should focus on the differences in how the models account for overdispersion:

Quasi-Poisson: Var(Y) = γμ

Negative binomial: Var(Y) = μ + μ2/k

Ver Hoef and Boveng (2007) suggest to examine the relationship between the mean and variance by plotting the squared residuals vs. the estimated mean from the Poisson regression model fit. If there is an increasing quadratic trend in the points, the negative binomial model is preferred. If there is mostly a linear trend, the quasi-Poisson is preferred.

Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.txt)

> set1 <- data.frame(res.sq = residuals(object = mod.fit,

type = "response")^2, mu.hat = mod.fit$fitted.values)

> fit.lin <- lm(formula = res.sq ~ mu.hat, data = set1)

> fit.quad <- lm(formula = res.sq ~ mu.hat + I(mu.hat^2),

data = set1)

> summary(fit.quad)

Call:

lm(formula = res.sq ~ mu.hat + I(mu.hat^2), data = set1)

Residuals:

Min 1Q Median 3Q Max

-12.409 -6.134 -2.673 -1.072 120.892

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -3.1635 7.1585 -0.442 0.659

mu.hat 6.1627 3.9398 1.564 0.120

I(mu.hat^2) -0.6097 0.4913 -1.241 0.216

Residual standard error: 16.4 on 170 degrees of freedom

Multiple R-squared: 0.0188, Adjusted R-squared: 0.007258

F-statistic: 1.629 on 2 and 170 DF, p-value: 0.1992

> plot(x = set1$mu.hat, y = set1$res.sq, xlab =

expression(hat(pi)), ylab = "Squared Residual")

> curve(expr = predict(object = fit.lin, newdata =

data.frame(mu.hat = x), type = "response"), col =

"blue", add = TRUE, lty = "solid")

> curve(expr = predict(object = fit.quad, newdata =

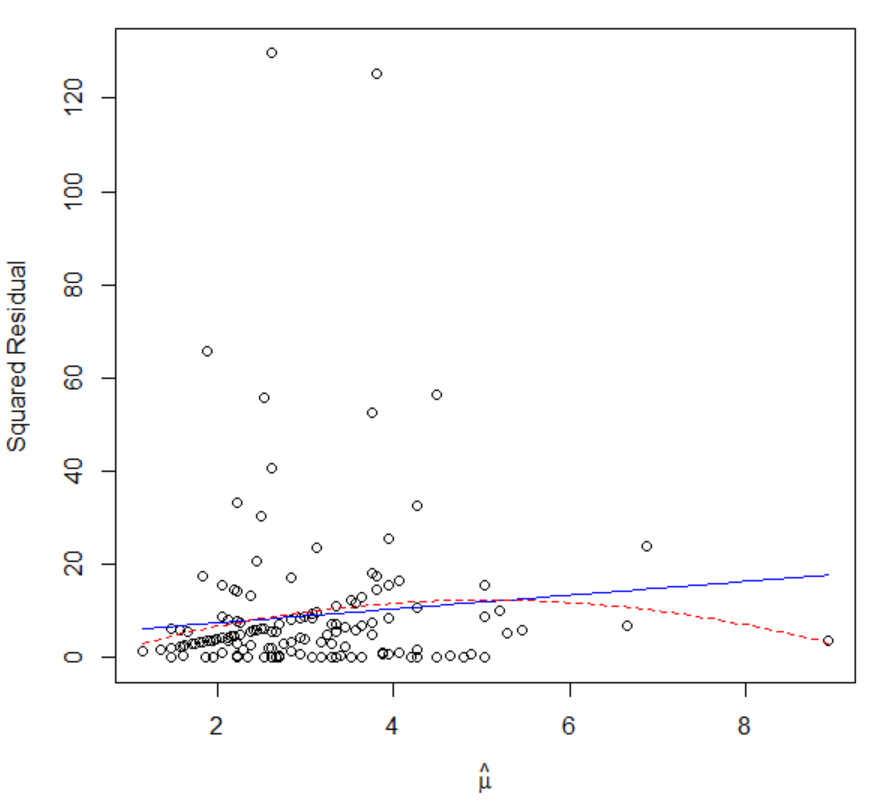
data.frame(mu.hat = x), type = "response"), col = "red",

add = TRUE, lty = "dashed")

> legend(x = 50, y = 1000, legend = c("Linear",

"Quadratic"), col = c("red", "blue"), lty = c("solid",

"dashed"), bty = "n")



At least with respect to the regression model, the quadratic term is non-significant. There may be one point at about  = 9 that is distorting a quadratic trend. If I had to choose here, I would likely go with the quasi-Poisson regression model (as long as there were no other explanatory variables available to explain the overdispersion).

Ver Hoef and Boveng (2007) also suggest to group the data first to help see trend better in the plot. My program shows how to do this. The overall conclusions are not different from here.

Overdispersion for other types of responses

A similar problem can occur with binomial or multinomial responses as well. For example, consider a setting where binomial observations are observed each with the same number of trials n. If the observed variance of the binomial observations is greater than the variance as given by the binomial distribution (nπ(1 – π) for each set of trials), then this a violation of the binomial assumptions. Simply, if n > 1 and ALL binomial observations were 0 and n, overdispersion would occur.

When overdispersion occurs, the options outlined on page 5.2 hold for a binomial response too, except a quasi-binomial model replaces the quasi-Poisson model and the beta-binomial regression model replaces the negative binomial regression model. The glm() function with family = quasibinomial(link = "logit") estimates the quasi-binomial model. The vglm() function of the VGAM package can estimate these models using a betabinomial() argument value for family.

Overdispersion cannot be measured when there are Bernoulli responses because these responses are only 0 or 1.

The multinomRob() function in the multinomRob package includes methods to handle overdispersion for multinomial data.

**Section 5.4: Examples**

Example: Placekicking (Placekick-FindBestModel.R, Examine.logistic.reg.R, placekick.mb.csv)

The data set used here has 13 more observations than the data set used for past placekick examples. The reason for why these observations are included (and then eventually excluded) will be given in the analysis.

The data used here also includes these additional variables:

* altitude: Official altitude in feet for the city where the placekick is attempted (not necessarily the exact altitude of the stadium)
* home: Binary variable denoting placekicks attempted at the placekicker's home (1) vs. away (0) stadium
* precip: Binary variable indicating whether precipitation is falling at game time (1) vs. no precipitation (0)
* temp72: Fahrenheit temperature at game time, where 72° is assigned to placekicks attempted within domed stadiums

If you would like to see a comprehensive examination of this data set, see my Master’s report. It is available from the KSU library (LD2668.R4 STAT 1996 B55). Also, see a summary of it in Bilder and Loughin (*Chance*, 1998). Note that I primarily used stepwise variable selection methods in those references, because these were the standard techniques at that time. We will be using all-subset variable selection methods here. Fortunately, the same model results (with AICc, not with BIC).

Below is a summary of the steps taken to find the “best” logistic regression model for the placekicking data set.

Variable selection

I tried to use the glmulti() function to search among the main-effects and two-way interactions. Because some interactions were not plausible for this problem, I chose to only examine the following interactions:

* distance with altitude, precip, wind, change, elap30, PAT, field, and temp72
* home with wind
* precip with type, field, and temp72

The exclude argument in glmulti() should allow for the exclusion of those interactions that I do not want to consider. Unfortunately, I came across an error in the function that does not allow this to work. Instead, I used glmulti() to search among the main-effects models, and then use forward selection to choose from among the 12 interactions. I used AICc as my information criterion, although AIC would likely result in the same model due to the large sample size.

> placekick.mb <- read.table("C:\\data\\Placekick.mb.csv",

header = TRUE, sep = ",")

> head(placekick.mb)

week distance altitude home type precip wind change

1 4 18 585 1 0 0 0 0

2 11 18 585 1 0 0 0 1

3 11 18 10 0 1 0 0 1

4 1 19 20 0 1 0 0 0

5 3 19 5 0 0 0 0 1

6 6 19 20 0 1 0 0 0

elap30 PAT field good temp72

1 24.3167 0 0 1 72

2 13.2000 0 0 1 72

3 1.5667 0 1 0 74

4 24.6333 0 1 1 79

5 25.6000 0 0 1 72

6 18.2500 0 1 1 82

> library(glmulti)

Loading required package: rJava

> mod.fit.full <- glm(formula = good ~ week + distance +

altitude + home + type + precip + wind + change +

elap30 + PAT + field, family = binomial(link = logit),

data = placekick.mb)

> summary(mod.fit.full)

Call:

glm(formula = good ~ week + distance + altitude + home + type + precip + wind + change + elap30 + PAT + field, family = binomial(link = logit), data = placekick.mb)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.9647 0.1689 0.2030 0.4606 1.5023

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.757e+00 5.691e-01 8.360 < 2e-16 \*\*\*

week -2.472e-02 1.964e-02 -1.259 0.20819

distance -8.838e-02 1.131e-02 -7.815 5.49e-15 \*\*\*

altitude 6.627e-05 1.085e-04 0.611 0.54120

home 2.168e-01 1.872e-01 1.158 0.24680

type 3.516e-01 2.900e-01 1.212 0.22539

precip -2.024e-01 4.472e-01 -0.453 0.65080

wind -6.599e-01 3.512e-01 -1.879 0.06027 .

change -3.210e-01 1.962e-01 -1.636 0.10187

elap30 3.704e-03 1.052e-02 0.352 0.72482

PAT 1.088e+00 3.697e-01 2.942 0.00326 \*\*

field -3.074e-01 2.643e-01 -1.163 0.24475

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1024.77 on 1437 degrees of freedom

Residual deviance: 765.74 on 1426 degrees of freedom

AIC: 789.74

Number of Fisher Scoring iterations: 6

> # All possible using AICc – I use an alternative way to

specify the full model here in comparison to what we

did earlier in the notes.

> search.1.aicc <- glmulti(y = mod.fit.full, level = 1,

method = "h", crit = "aicc", family = binomial(link =

"logit"))

Initialization...

TASK: Exhaustive screening of candidate set.

Fitting...

After 50 models:

Best model: good~1+distance

Crit= 791.070543909615

Mean crit= 913.654519390799

After 100 models:

Best model: good~1+distance+wind

Crit= 790.070921206521

Mean crit= 912.376246191703

<OUTPUT EDITED>

After 2000 models:

Best model: good~1+distance+wind+change+PAT

Crit= 780.80634381996

Mean crit= 783.524534831603

After 2050 models:

Best model: good~1+distance+wind+change+PAT

Crit= 780.80634381996

Mean crit= 783.524534831603

After 2100 models:

Best model: good~1+distance+wind+change+PAT

Crit= 780.80634381996

Mean crit= 783.523465948507

Completed.

> weightable(search.1.aicc)[1:20,]

model aicc weights

1 good ~ 1 + distance + wind + change + PAT 780.8063 0.03472394

2 good ~ 1 + week + distance + wind + change + PAT 781.1386 0.02940828

3 good ~ 1 + week + distance + change + PAT 781.1653 0.02901847

4 good ~ 1 + distance + change + PAT 781.2868 0.02730893

5 good ~ 1 + distance + wind + PAT 781.6147 0.02317858

6 good ~ 1 + distance + home + wind + change + PAT 781.7210 0.02197970

7 good ~ 1 + week + distance + home + wind + change + PAT 781.9811 0.01929937

8 good ~ 1 + week + distance + home + change + PAT 782.0795 0.01837210

9 good ~ 1 + week + distance + wind + PAT 782.1563 0.01768037

10 good ~ 1 + distance + PAT 782.1992 0.01730521

11 good ~ 1 + distance + home + change + PAT 782.2912 0.01652687

12 good ~ 1 + week + distance + PAT 782.3041 0.01642114

13 good ~ 1 + distance + type + wind + change + PAT 782.4182 0.01551002

14 good ~ 1 + distance + home + wind + PAT 782.5179 0.01475582

15 good ~ 1 + distance + altitude + wind + change + PAT 782.5242 0.01470994

16 good ~ 1 + distance + wind + change + elap30 + PAT 782.6700 0.01367538

17 good ~ 1 + distance + wind + change + PAT + field 782.7084 0.01341554

18 good ~ 1 + week + distance + type + wind + change + PAT 782.7914 0.01287009

19 good ~ 1 + week + distance + altitude + change + PAT 782.7933 0.01285774

20 good ~ 1 + distance + precip + wind + change + PAT 782.7976 0.01282999

> print(search.1.aicc)

glmulti.analysis

Method: h / Fitting: glm / IC used: aicc

Level: 1 / Marginality: FALSE

From 100 models:

Best IC: 780.80634381996

Best model:

[1] "good ~ 1 + distance + wind + change + PAT"

Evidence weight: 0.0347239351027185

Worst IC: 784.673641893819

20 models within 2 IC units.

90 models to reach 95% of evidence weight.

> AICc <- function(object) {

n <- length(object$y)

r <- length(object$coefficients)

AICc <- AIC(object) + 2\*r\*(r + 1)/(n - r - 1)

list(AICc = AICc, BIC = BIC(object))

}

> # Main effects model

> mod.fit <- glm(formula = good ~ distance + wind + change

+ PAT, family = binomial(link = logit), data =

placekick.mb)

> AICc(object = mod.fit)

$AICc

[1] 780.8063

$BIC

[1] 807.1195

> # Models with one interaction included

> mod.fit1 <- glm(formula = good ~ distance + wind + change

+ PAT + altitude + distance:altitude, family =

binomial(link = logit), data = placekick.mb)

<OUTPUT EDITED>

> mod.fit12 <- glm(formula = good ~ distance + wind +

change + PAT + precip + temp72 + precip:temp72, family

= binomial(link = logit), data = placekick.mb)

> inter <- c("distance:altitude", "distance:precip",

"distance:wind", "distance:change", "distance:elap30",

"distance:PAT", "distance:field", "distance:temp72",

"home:wind", "type:precip", "precip:field",

"precip:temp72")

> AICc.vec <- c(AICc(mod.fit1)$AICc, AICc(mod.fit2)$AICc,

AICc(mod.fit3)$AICc, AICc(mod.fit4)$AICc,

AICc(mod.fit5)$AICc, AICc(mod.fit6)$AICc,

AICc(mod.fit7)$AICc, AICc(mod.fit8)$AICc,

AICc(mod.fit9)$AICc, AICc(mod.fit10)$AICc,

AICc(mod.fit11)$AICc, AICc(mod.fit12)$AICc)

> all.AICc1 <- data.frame(inter = inter, AICc.vec)

> all.AICc1[order(all.AICc1[,2]), ]

inter AICc.vec

3 distance:wind 777.2594

6 distance:PAT 777.3321

7 distance:field 782.4092

4 distance:change 782.6573

9 home:wind 783.6260

1 distance:altitude 784.4997

10 type:precip 784.6068

2 distance:precip 784.6492

8 distance:temp72 784.6507

5 distance:elap30 784.6822

11 precip:field 785.7221

12 precip:temp72 786.5106

Comments:

* A number of models have a very similar AICc. Depending on the final outcome for a model here, Bayesian model averaging may be of interest to try.
* The “best” main effects model includes distance, wind, change, and PAT with a AICc = 780.8.
* Because AIC() does not calculate AICc, I wrote my own function for it.
* Because step() does not calculate AICc, I had to manually perform the forward selection.
* When examining the 12 interactions, I include corresponding main effects in the model even when they were not among distance, wind, change, and PAT. For example, notice that altitude is included as a main effect when determining the importance of distance:altitude with mod.fit1.
* The interaction that reduces the AICc the most is distance:wind, so this interaction is added to the model.
* Addition steps of forward selection are given in the corresponding program. The second step adds distance:PAT with a resulting AICc of 773.8. No other interactions reduce the AICc further.
* Of course, alternative paths could have been followed here. These paths may lead to a different final model.
  + If the genetic algorithm is used here with ALL main effects and ALL possible interactions, a number of other interactions are included. However, I had a lot of difficulty explaining these interactions (based on my knowledge of football), which is what originally led me to limit the interactions to those that make sense in terms of the problem.
  + If the BIC is used for the initial main effects model, the distance and PAT explanatory variables would be the only variables included.

Investigate possible improvements to the model

First, we need to convert the data to EVP form.

> # Convert data to EVP form; interactions are not needed

in aggregate() because they do not change the number

of unique combinations of explanatory variables.

> w <- aggregate(formula = good ~ distance + wind + change

+ PAT, data = placekick.mb, FUN = sum)

> n <- aggregate(formula = good ~ distance + wind + change

+ PAT, data = placekick.mb, FUN = length)

> w.n <- data.frame(w, trials = n$good, prop =

round(w$good/n$good, 4))

> head(w.n)

distance wind change PAT good trials prop

1 18 0 0 0 1 1 1.00

2 19 0 0 0 3 3 1.00

3 20 0 0 0 15 15 1.00

4 21 0 0 0 11 12 0.92

5 22 0 0 0 7 8 0.88

6 23 0 0 0 15 15 1.00

> nrow(w.n) # Number of EVPs (M)

[1] 124

> sum(w.n$trials) # Number of observations

[1] 1438

> # Estimates here match those had before converting data

to EVP form (not shown)

> mod.prelim1 <- glm(formula = good/trials ~ distance +

wind + change + PAT + distance:wind + distance:PAT,

family = binomial(link = logit), data = w.n, weights =

trials)

> round(summary(mod.prelim1)$coefficients, digits = 4)

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.4964 0.4814 9.3399 0.0000

distance -0.0807 0.0114 -7.0620 0.0000

wind 2.9248 1.7850 1.6385 0.1013

change -0.3320 0.1945 -1.7068 0.0879

PAT 6.7119 2.1137 3.1754 0.0015

distance:wind -0.0918 0.0457 -2.0095 0.0445

distance:PAT -0.2717 0.0980 -2.7726 0.0056

Are there transformations of the explanatory variables that may be helpful? Obviously, there are no transformations of wind, change, and PAT that need to be investigated because there are binary explanatory variables. With respect to distance, we can plot the standardized Pearson residuals vs. distance to look for trends. If trends exist, this suggests a transformation may be needed.

> # Plot of standardized Pearson residuals vs. distance

> stand.resid <- rstandard(model = mod.prelim1, type =

"pearson")

> plot(x = w.n$distance, y = stand.resid, ylim = c(min(-3,

stand.resid), max(3, stand.resid)), ylab =

"Standardized Pearson residuals", xlab = "Distance")

> abline(h = c(3, 2, 0, -2, -3), lty = 3, col = "blue")

> smooth.stand <- loess(formula = stand.resid ~ distance,

data = w.n, weights = trials)

> ord.dist <- order(w.n$distance)

> lines(x = w.n$distance[ord.dist], y =

predict(smooth.stand)[ord.dist], lty = "solid", col =

"red")



There is no discernable trend, so I do not think a transformation is necessary. I also tried fitting a model that included a squared distance term, and obtained an AICc = 775.72. This value is larger than the AICc = 773.8 that we had obtained previously for the model without the distance squared term. Note that when you do this information criteria comparison, it is VERY important that you calculate both values either using the EVP form or the binary form of the data. Why?

Assessing the model fit – preliminary model

> one.fourth.root <- function(x) {

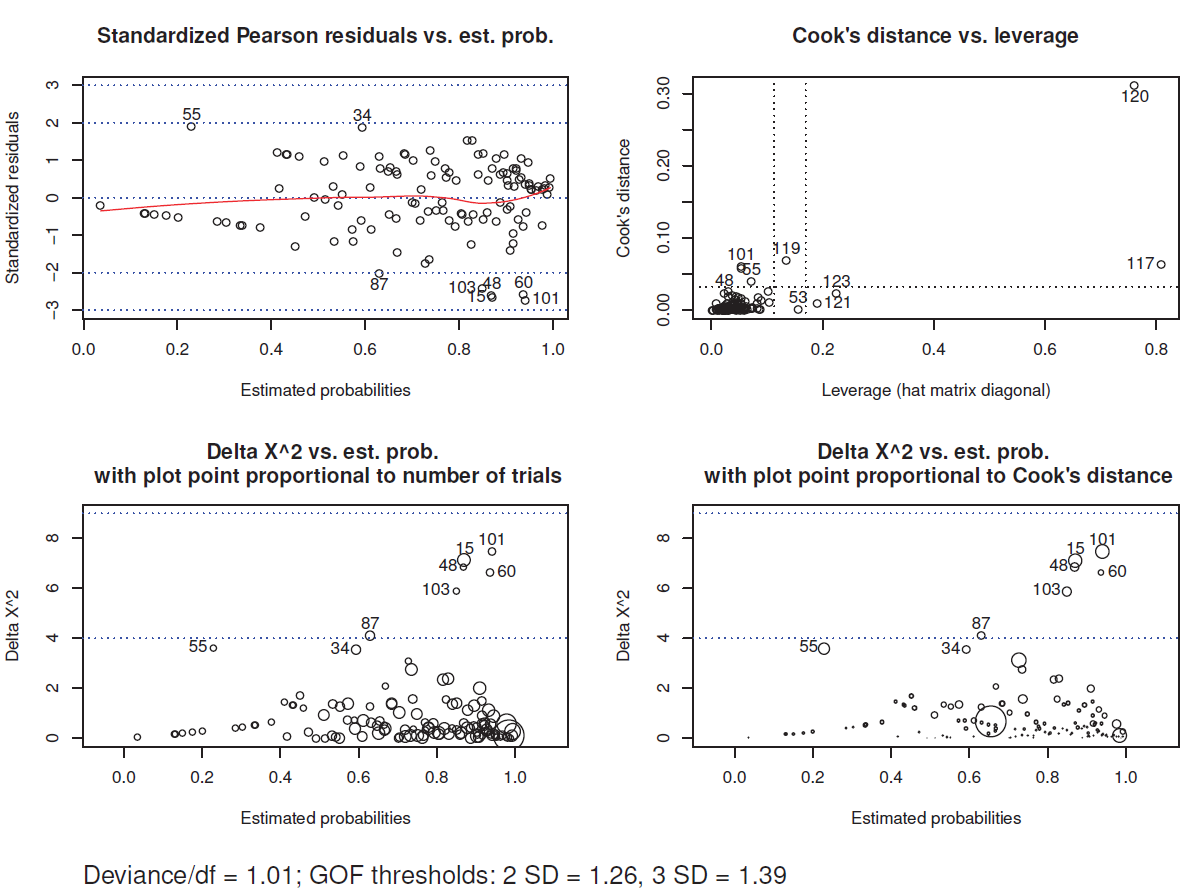
x^0.25

}

> save.info1 <- examine.logistic.reg(mod.fit.obj =

mod.prelim1, identify.points = TRUE, scale.n =

one.fourth.root, scale.cookd = sqrt)



Plots:

* (1,1): A few points are outside or close to these ±2 and ±3 lines
* (1,2): #120 has very large values in comparison to the others, so we definitely want to investigate it further; #117 has large leverage

* (2,1): Like in (1,1), we see a few points outside the lower threshold as we would expect; however, these points have relatively small plotting symbols (except perhaps #15) indicating that they may not really be unusual.

* (2,2): Notice that the very large Cook's distance value (#120, the largest circle on plot) has a relatively small squared standardized residual; this could indicate a very influential EVP that is “pulling” the model toward it to better its fit.

* Deviance/df: No evidence of any problems with the overall fit of the model

Next, I examined those EVPs identified in the plot further:

> w.n.diag1 <- data.frame(w.n, pi.hat =

round(save.info1$pi.hat, 2), std.res =

round(save.info1$stand.resid, 2), cookd =

round(save.info1$cookd, 2), h = round(save.info1$h, 2))

> p <- length(mod.prelim1$coefficients)

> ck.out <- abs(w.n.diag1$std.res) > 2 | w.n.diag1$cookd >

4/nrow(w.n) | w.n.diag1$h > 3\*p/nrow(w.n) # "|" means

"or"

> extract.EVPs <- w.n.diag1[ck.out,]

> extract.EVPs[order(extract.EVPs$distance),] # Order by

distance

distance wind change PAT good trials prop pi.hat

60 18 0 1 0 1 2 0.50 0.94

117 20 0 0 1 605 614 0.99 0.98

121 20 1 0 1 42 42 1.00 0.99

123 20 0 1 1 94 97 0.97 0.98

101 25 1 1 0 1 2 0.50 0.94

119 29 0 0 1 0 1 0.00 0.73

120 30 0 0 1 3 4 0.75 0.65

103 31 1 1 0 0 1 0.00 0.85

15 32 0 0 0 12 18 0.67 0.87

48 32 1 0 0 0 1 0.00 0.87

87 45 0 1 0 1 5 0.20 0.63

55 50 1 0 0 1 1 1.00 0.23

std.res cookd h

60 -2.57 0.01 0.01

117 0.32 0.06 0.81

121 0.52 0.01 0.19

123 -0.75 0.02 0.23

101 -2.73 0.06 0.05

119 -1.76 0.07 0.13

120 0.83 0.31 0.76

103 -2.43 0.03 0.03

15 -2.67 0.06 0.05

48 -2.62 0.02 0.02

87 -2.03 0.02 0.03

55 1.90 0.04 0.07

Comments:

* Most of these EVPs have a very small number of trials, so standard normal and chi-square approximations are not going to work well. For example, I am not concerned about #60 having rm = -2.57 because there are only two trials associated with it. A standard normal approximation for it does not make sense.
* The large hm values for #121 and #123 are likely due to the relatively large numbers of trials. Of course, these should be influential.
* The large Cook’s distance for #117 is likely due to the same reason as what we saw for #121 and #123. This EVP has 614 of the total 1438 trials in the data set!
* #15 has a low number of successes in comparison to its estimated probability of success, and it has a moderate number of trials. This causes me to have a little concern about it, but notice the standardized Pearson residual is between ±3.
* #120 really concerns me. Its explanatory variables correspond to an unusual type of placekick too – a PAT which is not 20 yards! Interestingly, #119 is a PAT not at 20 yards too, and it has a larger Cook’s distance value. We definitely need to investigate these problems further.

To determine how influential these #119 and #120 truly are, I TEMPORARILY removed them from the data set and refit the model. I chose to do both at the same time because they are both non-20 yard PATs.

> mod.prelim1.wo119.120 <- glm(formula = good/trials ~

distance + wind + change + PAT + distance:wind +

distance:PAT, family = binomial(link = logit), data =

w.n[-c(119, 120),], weights = trials)

> round(summary(mod.prelim1.wo119.120)$coefficients, digits

= 4)

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.4985 0.4816 9.3400 0.0000

distance -0.0807 0.0114 -7.0640 0.0000

wind 2.8770 1.7866 1.6103 0.1073

change -0.3308 0.1945 -1.7010 0.0889

PAT -12.0703 49.2169 -0.2452 0.8063

distance:wind -0.0907 0.0457 -1.9851 0.0471

distance:PAT 0.6666 2.4607 0.2709 0.7865

There is a dramatic change in the regression parameter estimate for distance:PAT! When both of these EVPs were in the data set, the estimate was -0.2717 with a Wald test p-value of 0.0056. It appears the presence of this interaction in the model was due to these EVPs consisting of just five placekicks.

Choices:

1. Leave these EVPs in the data set and keep the interaction in the model. In this case, I will just need to live with them being influential.
2. Take ALL non-20 yard PATs out of the data set and remove the interaction from the model. This will limit my population of inference.
3. Remove ALL PATs from the data set! Find the best model for just the field goals. Depending on interest, one could also find the best model for these PATs too in their own separate data set.

Overall, all three choices are justifiable. My preferences from best to worse are 2, 3, 1. Thus, I decided to go with 2.

With respect to 1, there is not enough information here (only 13 placekicks) to determine if the need for the interaction is a real trend or an anomaly of binary response data.

With respect to 3, one could do this if there was belief that the effects of the other variables differ based on whether or not the placekick was a PAT or field goal. I do not think this is the case. Plus, I think it is an important finding later in this example that the probability of success for 20-yard placekicks is different for field goals and PATs.

BIG NOTE: Removing observations from a data set should only occur when there is VERY GOOD justification for it! Just because an observation is an outlier or influential does NOT mean it should automatically be removed from the data set!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Assessing the model fit – revised model

First, the non-20 yard PATs need to be removed.

> # Remove non-20 yard PATs - "!" negates and "&" means

"and"

> placekick.mb2 <- placekick.mb[!(placekick.mb$distance

!= 20 & placekick.mb$PAT == 1),]

> nrow(placekick.mb2) # Number of observations after

13 were removed

[1] 1425

Now, the variable selection needs to be repeated due to the change in the data set! The code is in the corresponding program. As you might expect due to the small number of observations removed, the same terms for the model as before resulted (excluding the distance:PAT interaction). Below is information about the estimated model:

> # EVP form

> w2 <- aggregate(formula = good ~ distance + wind + change

+ PAT, data = placekick.mb2, FUN = sum)

> n2 <- aggregate(formula = good ~ distance + wind + change

+ PAT, data = placekick.mb2, FUN = length)

> w.n2 <- data.frame(w2, trials = n2$good, prop =

round(w2$good/n2$good, 2))

> # head(w.n2)

> nrow(w.n2) # Number of EVPs

[1] 119

> sum(w.n2$trials) # Number of observations

[1] 1425

> mod.prelim2 <- glm(formula = good/trials ~ distance +

wind + change + PAT + distance:wind, family =

binomial(link = logit), data = w.n2, weights = trials)

> summary(mod.prelim2)

Call:

glm(formula = good/trials ~ distance + wind + change + PAT + distance:wind, family = binomial(link = logit), data = w.n2, weights = trials)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.2386 -0.5836 0.1965 0.8736 2.2822

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.49835 0.48163 9.340 < 2e-16 \*\*\*

distance -0.08074 0.01143 -7.064 1.62e-12 \*\*\*

wind 2.87783 1.78643 1.611 0.10719

change -0.33056 0.19445 -1.700 0.08914 .

PAT 1.25916 0.38714 3.252 0.00114 \*\*

distance:wind -0.09074 0.04570 -1.986 0.04706 \*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 376.01 on 118 degrees of freedom

Residual deviance: 113.86 on 113 degrees of freedom

AIC: 260.69

Number of Fisher Scoring iterations: 5

Again, examine.logistic.reg() was used to assess the fit of model, and the corresponding code is included in the program. Similar results as before were found here. Below is the corresponding plot:



Some EVPs that were “potentially” influential were temporarily removed from the data set one at a time to determine if actual changes would occur to the model. Overall, the changes were relatively small, so I am satisfied with the model! My final estimated model is



Interpreting the model

See p. 24 and 30 of Bilder and Loughin (1998) for how the model can be used to make inferences. Note that the analyses for the paper were done using SAS, so the estimates and plots may be slightly different than those obtained from R.

Because the paper uses Wald confidence intervals for the odds ratios, I will generally use profile LR intervals here. The two should not be too different due to the large sample size.

> library(package = mcprofile)

> OR.name <- c("Change", "PAT", "Distance, 10-yard

decrease, windy", "Distance, 10-yard decrease, not

windy", "Wind, distance = 20", "Wind, distance = 30",

"Wind, distance = 40", "Wind, distance = 50", "Wind,

distance = 60")

> var.name <- c("int", "distance", "wind", "change", "PAT",

"distance:wind")

> K <- matrix(data = c(0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 1, 0,

0, -10, 0, 0, 0, -10,

0, -10, 0, 0, 0, 0,

0, 0, 1, 0, 0, 20,

0, 0, 1, 0, 0, 30,

0, 0, 1, 0, 0, 40,

0, 0, 1, 0, 0, 50,

0, 0, 1, 0, 0, 60),

nrow = 9, ncol = 6, byrow = TRUE, dimnames =

list(OR.name, var.name))

> # K # Check matrix

> linear.combo <- mcprofile(object = mod.prelim2, CM = K)

> ci.log.OR <- confint(object = linear.combo, level = 0.90,

adjust = "none")

> exp(ci.log.OR)

mcprofile - Confidence Intervals

level: 0.9

adjustment: none

Estimate lower upper

Change 0.7185 0.5223 0.991

PAT 3.5225 1.8857 6.785

Distance, 10-yard decrease, windy 5.5557 2.8871 12.977

Distance, 10-yard decrease, not windy 2.2421 1.8646 2.717

Wind, distance = 20 2.8950 0.7764 16.242

Wind, distance = 30 1.1683 0.5392 3.094

Wind, distance = 40 0.4715 0.2546 0.869

Wind, distance = 50 0.1903 0.0598 0.515

Wind, distance = 60 0.0768 0.0111 0.377

For the probability of success, below is an example of how to calculate profile LR intervals involving 20 yard placekicks under non-windy and no lead change conditions.

> K <- matrix(data = c(1, 20, 0, 0, 1, 0,

1, 20, 0, 0, 0, 0), nrow = 2, ncol =

6, byrow = TRUE, dimnames = list(c("PAT", "FG"),

var.name))

> linear.combo <- mcprofile(object = mod.prelim2, CM = K)

> ci.lin.pred <- confint(object = linear.combo, level =

0.90, adjust = "none")

> # as.matrix() is needed to obtain the proper class for

plogis()

> round(plogis(q = as.matrix(ci.lin.pred$estimate)), digits

= 3)

Estimate

PAT 0.984

FG 0.947

> round(plogis(q = as.matrix(ci.lin.pred$confint)), digits

= 3)

lower upper

[1,] 0.976 0.991

[2,] 0.921 0.966

Lin Elliott’s placekick corresponding to the paper discussion:

> K <- matrix(data = c(1, 42, 0, 1, 0, 0), nrow = 1, ncol =

6, byrow = TRUE)

> linear.combo <- mcprofile(object = mod.prelim2, CM = K)

> ci.lin.pred <- confint(object = linear.combo, level =

0.90, adjust = "none")

> round(plogis(q = as.matrix(ci.lin.pred$estimate)), digits

= 3) # as.numeric() and as.vector() do not work

Estimate

C1 0.685

> round(plogis(q = as.matrix(ci.lin.pred$confint)), digits

= 3)

lower upper

[1,] 0.628 0.738

Below are a few plots (see code in programs):



