Binary Response Regression Notes

Daniel J. Eck

Contents

In these notes we discuss generalized linear models (GLMs) for binary data with an emphasis on models arising from exponential family theory. Our main focus will be on logistic regression. We will also consider probit regression and propensity score estimation in causal inference.

Background: exponential family motivation

Suppose that we have a sample of data (y_i, x_i), $i = 1, \ldots, n$ where y_i is a binary response variable and x_i is a vector of predictors taking values in $\mathbb{R}^p$. Recall from the exponential family notes that the log likelihood of the exponential family is of the form

$$l(\theta) = \langle y, \theta \rangle - c(\theta), \tag{1}$$

where y is a vector statistic having components y_i and $\theta \in \mathbb{R}^n$ is the canonical parameter vector. In those notes θ is unconstrained and the likelihood (1) corresponds to a saturated regression model, one parameter for every observation.

We motivate generalized linear models (GLMs) beginning with their formulation within the context of exponential family theory. These models were referred to as canonical linear submodels in the exponential family notes. The various parameterizations discussed in those notes arose naturally from general exponential family theory, and those parameterizations were linked to each other via explicit (or implicit) mappings which preserved maximum likelihood estimation. A canonical linear submodel of an exponential family is a submodel having parameterization

$$\theta = M\beta$$

where $\theta \in \mathbb{R}^n$ is the canonical parameter vector corresponding to the original saturated exponential family, $\beta \in \mathbb{R}^p$ is the canonical parameter vector for the submodel, M is the model matrix with rows x_i^T. The matrix M is usually called the *model matrix* in the terminology used by the `R` functions `lm` and `glm`. We will use this notation moving forward in these notes unless otherwise stated. The submodel log likelihood is

$$l(\beta) = \langle M^T y, \beta \rangle - c(M\beta). \tag{2}$$

Recall that we have four parameters with relationships between them presented in Figure 1: the saturated model canonical and mean value parameters θ and μ and the canonical affine submodel canonical and mean value parameters β and $\tau = M^T \mu$. The observed equals expected property for the submodel is

$$\hat{\tau} = M^T \hat{\mu} = M^T y. \tag{3}$$

We cannot actually solve these equations for $\hat{\mu}$ because M the mapping $\mu \to M^T \mu$ is usually not one-to-one (the $n > p$ case where $M \in \mathbb{R}^{n \times p}$ and is full rank). Hence we cannot determine $\hat{\theta}$ and $\hat{\beta}$ from them either. The only way to determine the MLE is to maximize the log likelihood (2) for β to obtain $\hat{\beta}$ and then $\hat{\theta} = M\hat{\beta}$ and $\hat{\mu} = \nabla c(\hat{\theta})$ and $\hat{\tau} = M^T \hat{\mu}$.

In an exponential family GLM, the saturated model canonical parameter vector θ is "linked' ' to the saturated model mean value parameter vector through the change-of-parameter mappings $g(\theta)$. We can reparameterize $\theta = M\beta$ and write

$$\mathrm{E}_\theta(Y) = \mu = g(M\beta)$$

which implies that we can write

$$g^{-1}\left(\mathrm{E}_\theta(Y)\right) = M\beta.$$

Therefore, a linear function of the canonical submodel parameter vector is linked to the mean of the exponential family through the inverse change-of-parameter mapping g^{-1}. This is the basis of exponential family generalized linear models with link function g^{-1}.

$$\begin{tikzcd}
\theta \arrow[r, shift left] \arrow[d, shift left, "\mu = \nabla c(\theta)"] & \beta \arrow[l, shift left, "\theta = M\beta"] \arrow[d, shift left, "\tau = \nabla_\beta c(M\beta)"] \\
\mu \arrow[u, shift left] \arrow[r, shift left, "\tau = M^T \mu"] & \tau \arrow[l, shift left] \arrow[u, shift left]
\end{tikzcd}$$

Figure 1: A depiction of the transformations necessary to change between parameterizations. Arrows going in opposite directions specify transformations and their inverses. M is a known model matrix of full column rank, a is a known offset vector, and c is the cumulant function for the exponential family model.

Logistic regression model

The exponential family motivation is important as it allows us to develop general mathematical properties for models arising from the theory. Here we will switch gears to motivating specific models from a practitioner's point-of-view. We begin with the logistic regression model. The logistic regression model is one of the most widely used and studied GLMs in practice. It is the most important model for categorical response data, being commonly used for a wide variety of applications. The logistic regression model is used for analyzing a binary response variable, $y_i \in \{0, 1\}$ where a 1 encodes a "success" and a 0 encodes a "failure." The logistic regression model allows for users to model the probability of success as a function of covariates.

For a binary response variable Y and a vector of predictors X, let

$$p(x) = P(Y = 1|X = x).$$

The logistic regression model is then

$$p(x) = P(Y = 1|X = x) = \frac{\exp(x^T\beta)}{1 + \exp(x^T\beta)}. \tag{4}$$

Equivalently, the *logit* (log-odds) of the response variable has a linear relationship in the canonical submodel parameters:

$$\operatorname{logit}(p(x)) = \log\left(\frac{p(x)}{1 - p(x)}\right) = x^T\beta.$$

In vector notation, we can express the above as

$$\mathbf{p} = \frac{\exp(M\beta)}{1 + \exp(M\beta)} = \frac{1}{1 + \exp(-M\beta)} \quad \text{and} \quad \operatorname{logit}(\mathbf{p}) = M\beta$$

where the above $\exp(\cdot)$ and $\operatorname{logit}(\cdot)$ operations are understood as componentwise operations.

To see where the logit link in the specification of $p(x)$ comes from, consider the log likelihood of the binomial distribution

$$\begin{aligned} l(\beta) &\propto \sum_{i=1}^n y_i \log\left(\frac{p_i}{1 - p_i}\right) + \sum_{i=1}^n \log(1 - p_i) \\ &= \sum_{i=1}^n y_i\theta_i - \sum_{i=1}^n \log(1 + e^{\theta_i}) \end{aligned}$$

where

$$\theta_i = \log\left(\frac{p_i}{1 - p_i}\right) \qquad \text{and} \qquad p_i = \frac{e^{\theta_i}}{1 + e^{\theta_i}}.$$

We recognize the above as $p_i = g(\theta_i)$ and $\theta_i = g^{-1}(p_i)$ where g is the change-of-parameter map from canonical to mean-value parameter values for Bernoulli models.

Thus we see that the logistic regression model is the same as the canonical linear submodel of an exponential family with $\theta_i = x_i^T\beta$ which in vector notation is $\theta = M\beta$.

Takeaways:

- The logistic regression model is an exponential family model whose log likelihood can be written in canonical form. As such, the nice properties discussed over the last four lectures hold for this model.
- Note the differences between logistic and linear regression: the logistic regression model does not possess an additive error structure (ie signal plus noise); the change of parameters map g is not the identity function; the mean-value parameter is a success probability.
- In the first point above, it is interesting to note how John A Nelder, one of the creators of GLMs, identifies the statistics of his day as too focused on mathematical properties of error distributions instead of studying the mechanisms of signal which is of interest to scientists and technologists (See Section 2 of this paper).

Example: CCSO data

```
rm(list = ls())
library(tidyverse)
```

We will demonstrate logistic regression modeling on the Champaign County Sheriff's Office (CCSO) data frame. This data frame consists of $n = 67764$ observations and 30 variables in total. We will only consider a few of these variables in these notes, and we will only consider observations with complete cases.

We now load in the CCSO data from the stat528materials package. This work-in-progress R package contains materials for our course, including documentation on the CCSO data set that will be useful for these notes and your homework. We will perform most of our data wrangling operations using functionality in the tidyverse. Note these data wrangling steps can also be performed using functionality in the data.table package or using basic R. We use the tidyverse because it provides a framework that allows one to read and write code with relative ease. Some may say that the data.table provides a more powerful set of tools for handling large data sets, but these debates are beyond the scope of this lecture. You may want to play around with both!

```
#install.packages("devtools")
#devtools::install_github("DEck13/stat528materials")
library(stat528materials)
data("CCSO")

## data wrangling
CCSO_small = CCSO %>%
  mutate(atleastone = ifelse(daysInJail > 0,1,0)) %>%
  filter(crimeCode == "OTHER TRAFFIC OFFENSES") %>%
  filter(race %in% c("Asian/Pacific Islander","Black","White","Hispanic")) %>%
  filter(sex %in% c("Female","Male")) %>%
  dplyr::select(atleastone, arrestAge, sex, race, bookingDate) %>%
  mutate(race = fct_drop(race), sex = fct_drop(sex))
CCSO_small = CCSO_small[complete.cases(CCSO_small), ]

head(CCSO_small)
```

```
## # A tibble: 6 x 5
##   atleastone arrestAge sex    race     bookingDate
##        <dbl>     <dbl> <fct>  <fct>    <date>
## 1          0        22 Male   White    2011-01-01
## 2          0        26 Male   White    2011-01-01
## 3          0        32 Female White    2011-01-01
## 4          0        22 Male   White    2011-01-02
## 5          0        35 Male   Hispanic 2011-01-02
## 6          0        35 Male   Hispanic 2011-01-02
```

```
dim(CCSO_small)
```

```
## [1] 5916    5
```

Fit model

In this analysis we will investigate the propensity of incarcerations lasting longer than one day for crimes encoded as "other traffic offenses". The response variable is `atleastone` where a 1 indicates an incarceration lasting longer than one day and a 0 indicates an incarceration lasting shorter than 1 day. We will also determine whether or not any demographic variables drive the propensity of incarcerations lasting longer than one day. The variables under consideration are: Age (age at arrest), Sex (Male or Female), Race (Asian/Pacific Islander, Black, Hispanic, and White).

We can fit a basic main effects model in an instant using the glm function in R.

```
m1 = glm(atleastone ~ -1 + race + sex + arrestAge, data = CCSO_small,
         family = "binomial", x = "TRUE")
```

Now let's unpack the glm function call above. We decided that we wanted to fit an exponential family regression model with log likelihood taking the general form

$$l(\beta) = \langle M^T y, \beta \rangle - c(M\beta),$$

where y is the vector of responses β is the submodel canonical statistic vector corresponding to the model matrix M specified by the formula in the glm function call above. The first few rows of M are displayed below

```
M = m1$x
head(M)
```

```
##   raceAsian/Pacific Islander raceBlack raceHispanic raceWhite sexMale arrestAge
## 1                          0         0            0         1       1        22
## 2                          0         0            0         1       1        26
## 3                          0         0            0         1       0        32
## 4                          0         0            0         1       1        22
## 5                          0         0            1         0       1        35
## 6                          0         0            1         0       1        35
```

MLE The specific log likelihood for the logistic regression model can then be written as

$$l(\beta) \propto \sum_{i=1}^{n} y_i x_i^T \beta - \log\left(1 + \exp(x_i^T \beta)\right),$$

where the x_is are the rows of the design matrix M and the y_is are the components of the response vector y (the `atleastone` variable corresponding to incarcerations lasting longer than one day). The glm function then maximizes the above log likelihood with respect to β using IRLS.

The glm model object `m1` has $\hat{\beta}$ stored among many other useful quantities, some of which will be discussed shortly. We can view summary information for $\hat{\beta}$ and the fitting process using the summary function

```
summary(m1)
```

```
##
## Call:
## glm(formula = atleastone ~ -1 + race + sex + arrestAge, family = "binomial",
##     data = CCSO_small, x = "TRUE")
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## raceAsian/Pacific Islander -4.389865   0.523612  -8.384  < 2e-16 ***
## raceBlack                  -1.876550   0.144601 -12.977  < 2e-16 ***
## raceHispanic               -2.804549   0.173349 -16.179  < 2e-16 ***
## raceWhite                  -3.043226   0.147160 -20.680  < 2e-16 ***
## sexMale                     0.739834   0.105380   7.021 2.21e-12 ***
## arrestAge                   0.007705   0.003186   2.418   0.0156 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8201.3  on 5916  degrees of freedom
## Residual deviance: 4668.7  on 5910  degrees of freedom
## AIC: 4680.7
##
## Number of Fisher Scoring iterations: 6
```

The Estimate column in the above summary table is $\hat{\beta}$. The standard error column contains estimates of the square root of the variances of the estimated submodel canonical parameter vector $\hat{\beta}$. Recall from the asymptotic theory of maximum likelihood estimation that

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, \Sigma^{-1}),$$

where Σ^{-1} is the inverse of the Fisher information matrix. We can extract these same standard errors using the vcov function

```r
sqrt(diag(vcov(m1)))
```

```
## raceAsian/Pacific Islander                  raceBlack
##                0.523611637                0.144600942
##               raceHispanic                  raceWhite
##                0.173349091                0.147160126
##                    sexMale                  arrestAge
##                0.105379771                0.003186262
```

These values are the same as those in the Std. Error column in the above summary table

```r
all.equal(summary(m1)$coef[, 2], sqrt(diag(vcov(m1))))
```

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

Inference in logistic regression

The CCSO example connects the exponential family that we have developed to R-based data analysis. We will now discuss inference in logistic regression models. We begin with β. Recall that β exhibits a linear relationship when the response variable takes the form $\text{logit}(p(x))$, where $p(x)$ is the conditional probability of success at covariate value x.

The logit function has another names, the log-odds ratio. Therefore, a unit increase in one predictor variables x_j corresponds to an increase of β_j (estimated by $\hat{\beta}_j$) in the log-odds ratio with everything else being held fixed. This interpretation is similar as that in linear regression, except it is harder. A unit increase in x_j corresponds to an increase of β_j (estimated by $\hat{\beta}_j$) increase in the log of the ratio of the probability of success to the probability of failure.

The odds ratio itself is hard to interpret, so in many cases simple heuristics suffice. Just like in linear regression, the following rules hold:

- $\beta > 0$: increasing X implies that $\mathbb{P}(Y = 1|X = x)$ increases.
- $\beta = 0$: changing X implies that $\mathbb{P}(Y = 1|X = x)$ do not change.

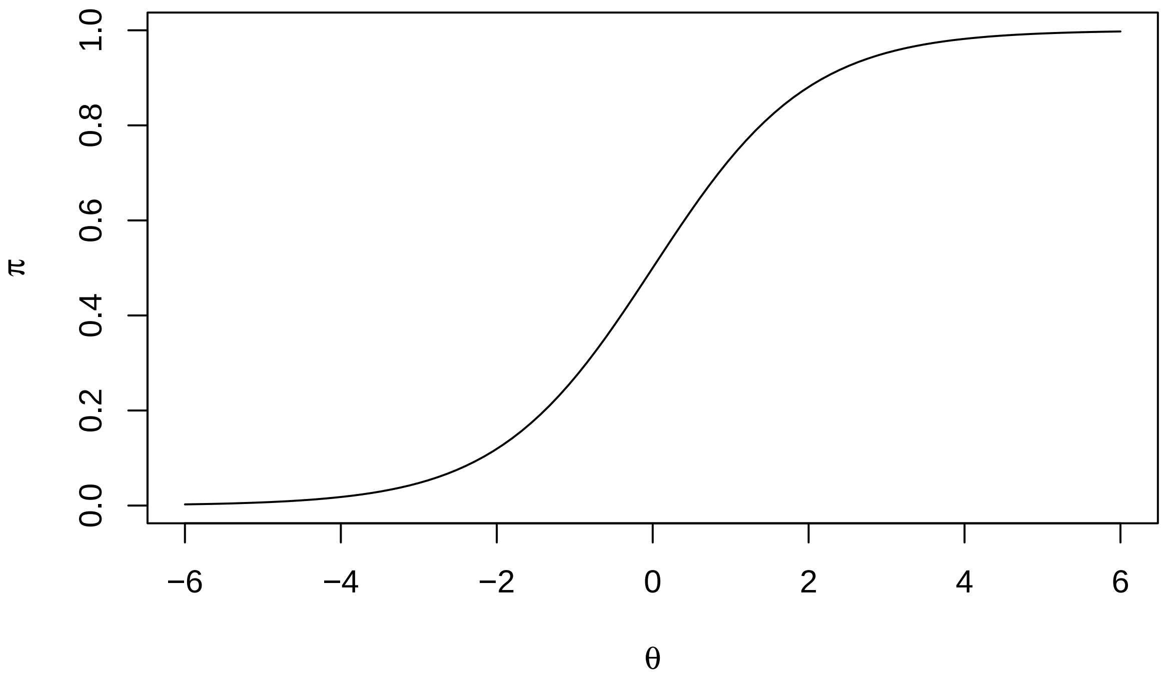
- $\beta < 0$: increasing X implies that $\mathbb{P}(Y = 1|X = x)$ decreases.

Note that the similarities between logistic and linear regression models stop when we discuss the relationship between β and $p(x) = \mathbb{P}(Y = 1|X = x)$. In linear regression there is no "second parameterization," the link function g^{-1} is the identity function, so the slope $\nabla_x E(Y|X = x) = \beta$ in linear regression. In logistic regression, we have that

$$\nabla_x p(x) = \nabla_x \mathrm{E}(Y|X = x) = \beta p(x)(1 - p(x)),$$

where $\mathrm{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$. The relationship between $p(x)$ and the linear predictor $\theta(x) = x^T\beta$ is shown below

```
library(faraway)
curve(ilogit(x),-6,6, xlab=expression(theta), ylab=expression(pi))
```



The logistic curve is almost linear in its midrange. This means that for modeling responses that are all of moderate probability, logistic and linear regression will not behave very differently. The curve approaches one at its upper end and zero at its lower end but never reaches these bounds. This means that logistic regression will never predict that anything is inevitable or impossible.

Now that we have a decent idea of what the submodel canonical parameter vector β is in a logistic regression model we can discuss statistical inference corresponding to estimates of β.

Inference and goodness-of-fit testing

Note that some contents from this section are similar to what was covered in the exponential family theory notes, and are included here for completeness.

Wald inference for regression coefficients

As in linear regression we can make inferences about β_j using the Wald statistic corresponding to the hypothesis test

$$H_o : \beta_j = 0, \qquad H_a : \beta_j \neq 0,$$

which is given by

$$\frac{\hat{\beta}_j}{\text{se}(\hat{\beta}_j)} \sim N(0,1),$$

where this distributional relationship holds under the null hypothesis $\beta_j = 0$. Similarly, we can form a confidence interval

$$\hat{\beta}_j \pm z_{\alpha/2}\text{se}(\hat{\beta}_j)$$

where $0 < \alpha < 1$ is some error threshold. In a similar vein, we can construct $(1-\alpha) \times 100\%$ confidence intervals for the odds ratio (not log-odds ratio) e^{β_j} given as

$$(e^{L_j}, e^{U_j})$$

where $L_j = \hat{\beta}_j - z_{\alpha/2}\text{se}(\hat{\beta}_j)$ and $U_j = \hat{\beta}_j + z_{\alpha/2}\text{se}(\hat{\beta}_j)$ when the Wald confidence interval is reported.

The $(1-\alpha) \times 100\%$ Wald confidence interval for the response variable $\text{logit}(p(x))$ at a particular x follows from the Delta method. We have that $\text{logit}(p(x)) = x^T\beta$ which implies that $\nabla_\beta \text{logit}(p(x)) = x$. Therefore,

$$\sqrt{n}\left(\text{logit}(\hat{p}(x)) - \text{logit}(p(x))\right) \xrightarrow{d} N(0, x^T\Sigma^{-1}x).$$

We can report the following $(1-\alpha) \times 100\%$ Wald confidence interval for the log-odds ratio

$$\text{logit}(\hat{p}(x)) \pm z_{\alpha/2}\sqrt{x^T\widehat{\Sigma}^{-1}x}.$$

Note that, while commonly used in practice, Wald inference can be suboptimal in certain settings. See Section 5.2.6 of Agresti [2013] for specifics.

Deviance and likelihood ratio testing

Let $l(\mu; y)$ denote the the log-likelihood of a GLM in the mean-value parameter μ. Let $l(\hat{\mu}; y)$ denote the MLE of the log likelihood for the model. The saturated model with $\hat{\mu} = y$ has the maximum achievable log likelihood $l(y; y)$. This model is not useful in practice, but it does allow for comparison to other model fits. The deviance is defined by

$$-2\left[l(\hat{\mu}; y) - l(y; y)\right].$$

This is the likelihood-ratio for testing the null hypothesis that the model against the general alternative (ie, the saturated model). The deviance has reference distribution

$$-2\left[l(\hat{\mu}; y) - l(y; y)\right] \approx \chi^2_{\text{df}}$$

where $\text{df} = n - p$, n is the sample size, and p is the number of model parameters. For large samples we can use the deviance statistic to test nested models. Denote $\mathcal{M}_o$ and $\mathcal{M}_a$, respectively, as the null model and the alternative model, and let $\hat{\mu}_o$ and $\hat{\mu}_a$, respectively, be the estimated mean-value parameter vectors under $\mathcal{M}_o$ and $\mathcal{M}_a$. Then the difference in deviances between the null model and the alternative model is,

$$-2\left[l(\hat{\mu}_o; y) - l(\hat{\mu}_a; y)\right] = -2\left[l(\hat{\mu}_o; y) - l(y; y)\right] - \left\{-2\left[l(\hat{\mu}_a; y) - l(y; y)\right]\right\} \approx \chi^2_{\text{df}},$$

where $\text{df} = p_a - p_o$, p_a is the number of model parameters in model $\mathcal{M}_a$, and p_o is the number of model parameters in model $\mathcal{M}_o$.

Example: CCSO data (continued)

We are now equipped to understand the summary table that is returned by summary.glm.

```
summary(m1)
```

```
##
## Call:
## glm(formula = atleastone ~ -1 + race + sex + arrestAge, family = "binomial",
##     data = CCSO_small, x = "TRUE")
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## raceAsian/Pacific Islander -4.389865   0.523612  -8.384  < 2e-16 ***
## raceBlack                  -1.876550   0.144601 -12.977  < 2e-16 ***
## raceHispanic               -2.804549   0.173349 -16.179  < 2e-16 ***
## raceWhite                  -3.043226   0.147160 -20.680  < 2e-16 ***
## sexMale                     0.739834   0.105380   7.021 2.21e-12 ***
## arrestAge                   0.007705   0.003186   2.418   0.0156 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8201.3  on 5916  degrees of freedom
## Residual deviance: 4668.7  on 5910  degrees of freedom
## AIC: 4680.7
##
## Number of Fisher Scoring iterations: 6
```

We can use the deviance and degrees of freedom objects to perform the likelihood ratio test to determine whether or not the main effects model fits the data well. In this case where the intercept is suppressed, the null model forces the intercept to be zero. This is equivalent to assuming that the Bernoulli success probability is fixed at $p = 1/2$ for every subject.

```
## compare with saturated model
pchisq(m1$deviance, df = m1$df.residual, lower = FALSE)
```

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

```
## use Chi squared testing to directly compare submodels
# null deviance
m1$null.deviance
```

```
## [1] 8201.317
```

```
# deviance of model with p = 1/2
-2 * 5916 * log(1/2)
```

```
## [1] 8201.317
```

```
pchisq(m1$null.deviance - m1$deviance, df = m1$df.null - m1$df.residual,
       lower = FALSE)
```

```
## [1] 0
```

```
## use LRT testing in the anova function
# here the null model allows for an intercept term to be present
m_null = glm(atleastone ~ 1, family = "binomial", data = CCSO_small)
anova(m_null, m1, test = "LRT")
```

```
## Analysis of Deviance Table
##
```

```
## Model 1: atleastone ~ 1
## Model 2: atleastone ~ -1 + race + sex + arrestAge
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      5915     4982.7
## 2      5910     4668.7  5   313.99 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can say that the fitted model fits the data better than a null model or a saturated model at any reasonably chosen error tolerance.

Let's consider the smaller model that ignores the Sex variable. A likelihood ratio test shows that the larger model is preferable at any reasonably chosen significance level α.

```
m_small = glm(atleastone ~ -1 + race + arrestAge, family = "binomial", data = CCSO_small)

## built in likelihood ratio test using anova.glm
anova(m_small, m1, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: atleastone ~ -1 + race + arrestAge
## Model 2: atleastone ~ -1 + race + sex + arrestAge
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      5911     4724.4
## 2      5910     4668.7  1   55.709 8.401e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## perform the above directly
pchisq(55.709, df = 1, lower = FALSE)
```

```
## [1] 8.403269e-14
```

Information criteria agree with the above.

```
AIC(m_small); AIC(m1)
```

```
## [1] 4734.438
```

```
## [1] 4680.728
```

```
BIC(m_small); BIC(m1)
```

```
## [1] 4767.865
```

```
## [1] 4720.841
```

Mean-value parameters

The canonical parameterization scale is bit awkward for interpretation, although summary tables provide some insight to which components of the submodel canonical parameter vector may be driving the data generating process (under the assumed model). R software provides functionality for estimating the mean value parameters $p(x) = \mathrm{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$, associated with every individual in the study. The function used to obtain mean value parameter estimates is `predict.glm` with `type = "response"` specified

```
p1 = predict(m1, type = "response", se.fit = TRUE)
```

Under the hood, we specified a model with a formula call and a data frame. A model matrix M is then created and $\hat{\beta}$ is estimated with the initial call to `glm`. The saturated model canonical parameter is expressed as $\hat{\theta} = M\hat{\beta}$, and this results in estimated mean value parameters of the form $\hat{\mu} = \nabla c(\hat{\theta})$. Recall that the submodel log likelihood can be expressed as

$$\begin{aligned} l(\beta) &\propto \sum_{i=1}^{n} y_i x_i^T \beta - \log\left(1 + \exp(x_i^T \beta)\right) \\ &= \sum_{i=1}^{n} y_i \theta_i - \log\left(1 + \exp(\theta_i)\right) = l(\theta) \end{aligned}$$

where $\theta_i = x_i^T \beta$. Therefore the mean value parameters are

$$\nabla c(\theta) = \nabla \log\left(1 + \exp(\theta)\right) = \frac{e^\theta}{1 + e^\theta},$$

which are estimated by plugging in $\hat{\theta} = M\hat{\beta}$ to the above. This is what `predict.glm` does to obtain $\hat{\mu}$ for a model with model matrix M.

```r
## estimates from GLM
p1.fit = as.numeric(p1$fit)

## estimates from hand
betahat = m1$coefficients
foo = as.numeric(exp(M %*% betahat) / (1 + exp(M %*% betahat)))

## the same
head(cbind(p1.fit, foo))
```

```
##          p1.fit        foo
## [1,] 0.10584708 0.10584708
## [2,] 0.10879965 0.10879965
## [3,] 0.05750466 0.05750466
## [4,] 0.10584708 0.10584708
## [5,] 0.14245614 0.14245614
## [6,] 0.14245614 0.14245614
```

```r
all.equal(p1.fit, foo)
```

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

Good, they are in fact equal.

Standard errors $\text{se}(\hat{\mu})$ that are computed from `predict.glm` also follow from straighforward exponential family results.

There are noted problems with Wald based confidence intervals for mean value parameters in logistic regression (the success probability). See here for more details in the context of binomial sampling proportions.

Be careful with inferences

It is important to remember that a canonical linear submodel is not unique to a particular model matrix or a particular model matrix, but is uniqely identified by the span of a model matrix. We demonstrate this point using the CCSO data. Here we create two new model matrices with same span as our original model and demonstrate that the likelihood has the same maximized value and mean-value parameter estimates are the same for all considered models.

```r
## fit model without suppressing the intercept
m2 = glm(atleastone ~ race + sex + arrestAge, data = CCSO_small,
         family = "binomial", x = "TRUE")
M3 = m1$x %*% eigen(vcov(m1))$vec
m3 = glm(CCSO_small$atleastone ~ -1 + M3, family = "binomial")

## coefficient estimates are different
mat = cbind(coef(m1), coef(m2), coef(m3))
mat
```

```
##                                   [,1]        [,2]        [,3]
## raceAsian/Pacific Islander -4.38986549 -4.38986549 -4.99124291
## raceBlack                  -1.87654964  2.51331585  3.81765861
## raceHispanic               -2.80454861  1.58531688 -0.02217119
## raceWhite                  -3.04322627  1.34663922  0.63194478
## sexMale                     0.73983403  0.73983403  0.76013870
## arrestAge                   0.00770504  0.00770504 -0.04255169
```

```r
## maximized likelihood and estimates of mean-value parameters are
## the same
AIC(m1); AIC(m2); AIC(m3)
```

```
## [1] 4680.728

## [1] 4680.728

## [1] 4680.728
```

```r
all.equal(p1.fit, as.numeric(predict(m2, type = "response")))
```

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

```r
all.equal(p1.fit, as.numeric(predict(m3, type = "response")))
```

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

Profile likelihood based confidence intervals

Say that we want a confidence interval for a parameter vector β in a model $\mathcal{M}$. Let $\mathcal{M}_o(\beta_o)$ be the same model as $\mathcal{M}$, except that β is fixed at β_o. Then the likelihood ratio tests the hypothesis that

$$H_o : \beta = \beta_o \qquad H_a : \beta \neq \beta_o$$

and this test produces a P-value (from a χ^2 approximation). Then

$$\{\beta_o : \text{P-value} > \alpha\}$$

is a $(1-\alpha) \times 100\%$ confidence set (usually a confidence interval) for β. This interval is a profile likelihood confidence interval. Standard software estimates this interval.

```r
## profile based method
beta.profile = confint(m1)
```

```
## Waiting for profiling to be done...
```

```
beta.profile
```

```
##                                     2.5 %      97.5 %
## raceAsian/Pacific Islander -5.59295728 -3.48648322
## raceBlack                  -2.16230780 -1.59523096
## raceHispanic               -3.14837654 -2.46857072
## raceWhite                  -3.33481810 -2.75773448
## sexMale                     0.53693218  0.95034741
## arrestAge                   0.00141719  0.01391126
```

```
## Wald based method
se = sqrt(diag(vcov(m1)))
cbind(betahat + qnorm(0.025) * sqrt(diag(vcov(m1))),
      betahat + qnorm(0.975) * sqrt(diag(vcov(m1))))
```

```
##                                      [,1]       [,2]
## raceAsian/Pacific Islander -5.416125438 -3.3636055
## raceBlack                  -2.159962276 -1.5931370
## raceHispanic               -3.144306583 -2.4647906
## raceWhite                  -3.331654813 -2.7547977
## sexMale                     0.533293472  0.9463746
## arrestAge                   0.001460082  0.0139500
```

Diagnostics (will revisit)

Model diagnostics for logistic regression fits are not as prevalent as those in linear regression.

Esarey and Pierce [2012] developed a method to assess the fit of binary-dependent variable models and compare this method to existing approaches. This method compares a model's predicted probability $\mathbb{P}(Y = 1)$ to the observed frequency of $Y = 1$ in the data set. If the model is a good fit to the data, subsets of the data with $\mathbb{P}(Y = 1) \approx m$ should have about m proportion of cases for which $Y = 1$. The process is analogous to plotting fitted and observed values of the dependent variable against one another, a commonly used fit diagnostic for continuous dependent variable models.

The difficulty is that (unlike Y) the true success probability p is not observable. But if we collect all the observations with values of $\hat{p} \approx m$, approximately m proportion of those cases should be $Y = 1$ when $\hat{p}$ is a good estimate of p. That is,

$$\mathbb{P}(Y = 1|\hat{p} = m) \approx m$$

when our model provides good estimates of p.

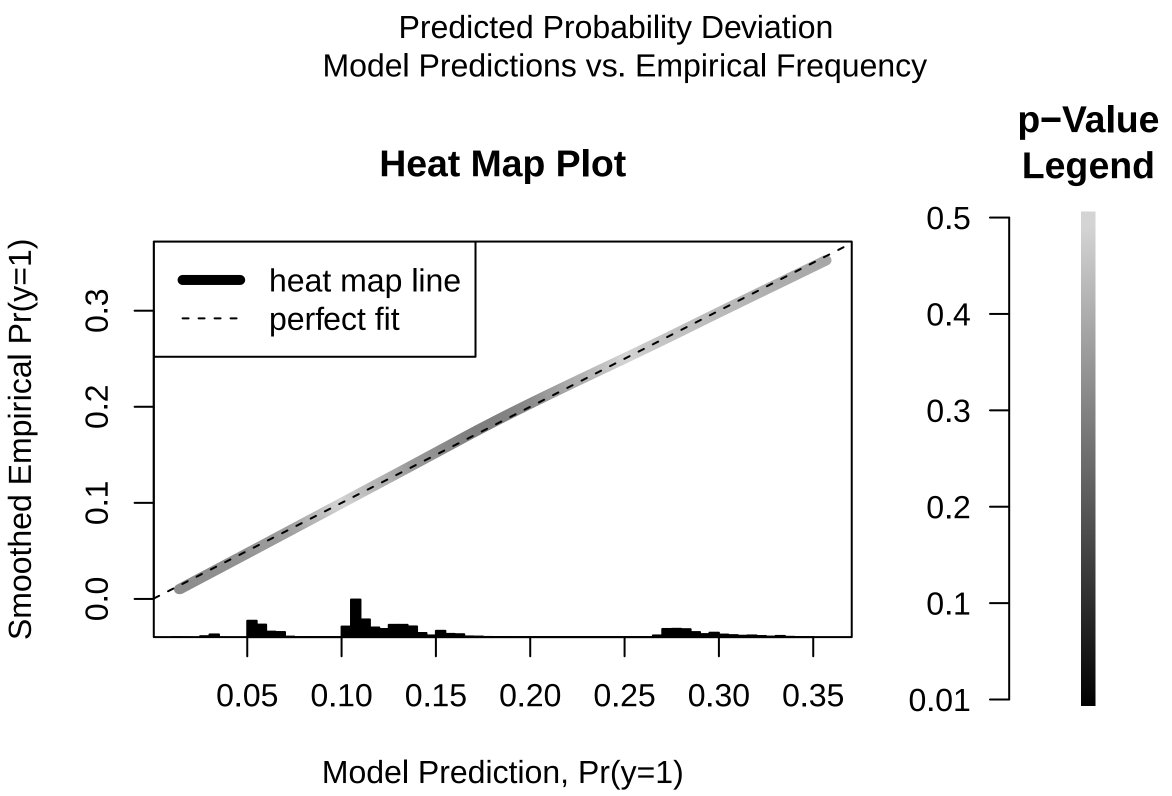
In brief, the approach in Esarey and Pierce [2012] is to estimate an empirical model, sort observations according to the predicted p of the model, and then determine whether this predicted probability is an accurate predictor of $R(p) = \mathbb{P}(Y = 1|p)$ by plotting them against each other. In a perfectly fitting model, $p = R(p)$ for all values of p and the plot is a straight 45 degree line through the origin. We will revisit details of this method when cover diagnostics in more detail.

This methodology is implemented in the `heatmapFit` package. We see that our fitted model exhibits great fit.

```
library(heatmapFit)
y = CCSO_small$atleastone
heatmap.fit(y, p1.fit)
```

```
##
## Calculating optimal loess bandwith...
## aicc Chosen Span =  0.9899469
##
```

```
## Generating Bootstrap Predictions...
##   |                                                                         |
```

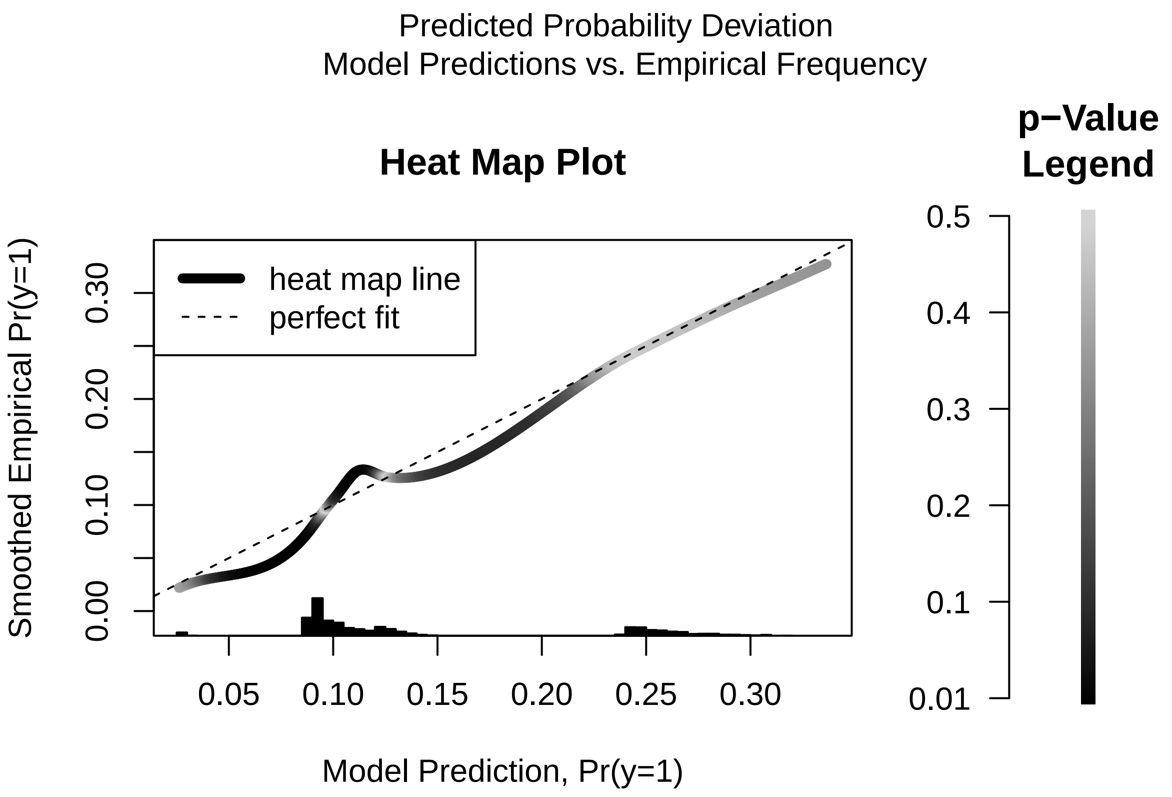


```
##
##
## *******************************************
## 0% of Observations have one-tailed p-value <= 0.1
## Expected Maximum = 20%
## *******************************************
```

We can see that the model which omits sex does not fit the data as well.

```
heatmap.fit(y, predict(m_small, type = "response"))
```

```
##
## Calculating optimal loess bandwith...
## aicc Chosen Span =  0.6419477
##
## Generating Bootstrap Predictions...
##   |                                                                         |
```



```
##
##
## ******************************************
## 34.11089% of Observations have one-tailed p-value <= 0.1
## Expected Maximum = 20%
## ******************************************
```

Appendix: nonparametric cases bootstrap

We introduce a nonparametric bootstrap procedure to estimate the uncertainty of estimates of model parameters. The idea of the bootstrap is to apply the same estimation task used in the original data set to many resamples of the observed data in order to estimate uncertainty of the modeling task. The intuition is that if the original data is large enough, then resamples from the original data mimic sampling from the data-generating process.

Suppose that we have a sample of data (y_i, x_i), $i = 1, \ldots, n$ where y_i is a binary response variable and x_i is a vector of predictors taking values in $\mathbb{R}^p$. For the bootstrapping cases method to estimate uncertainty of estimates of the submodel canonical parameter vector, the procedure is as follows:

1. Sample (y_i^*, x_i^*), $i = 1, \ldots, n$ with replacement from the original sample (y_i, x_i), $i = 1, \ldots, n$.
2. Calculate $\hat{\beta}^*$ using the resampled data in step 1 in the same way that $\hat{\beta}$ is calculated using the original data.
3. Repeat steps 1-2 a total of B times to for the bootstrap distribution of $\hat{\beta}$.

We can then estimate the uncertainty using Wald-type confidence intervals where the standard errors are calculated using the bootstrap sample of B values of $\hat{\beta}^*$

$$\hat{\beta} \pm Z_{1-\alpha/2}\hat{se}(\hat{\beta})$$

where $Z_{1-\alpha/2}$ is the $(1-\alpha/2)$th quantile of a standard normal distribution.

Acknowledgments

These notes borrow materials from Trevor Park's STAT 426 notes and Charles Geyer's notes on exponential families and other topics. We also borrow materials from Agresti [2013], Faraway [2016], and Sundberg [2019].

References

Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, 2013.

Justin Esarey and Andrew Pierce. Assessing fit quality and testing for misspecification in binary-dependent variable models. *Political Analysis*, 20(4):480–500, 2012.

Julian J Faraway. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press, 2016.

Rolf Sundberg. *Statistical Modelling by Exponential Families*, volume 12. Cambridge University Press, 2019.