GLM Diagnostics Notes

Daniel J. Eck

Contents

In these notes we discuss diagnostics for generalized linear models (GLMs) with discrete responses that arise naturally from exponential family theory. These diagnostics extend to other cases as well. First, we load in the needed software packages.

```
rm(list = ls())
library(tidyverse)
library(ggplot2)
library(data.table)
library(heatmapFit)
#install.packages("devtools")
#devtools::install_github("jhlee1408/assessor")
library(assessor)
```

Model assessment

Esarey and Pierce (2012)

We investigated the Esarey and Pierce [2012] method for assessing the fit of binary response models. We will now provide more details of this method. The basic plan is to compare a model's predicted probability, or $\hat{p}$, to an in-sample empirical estimate of $\mathbb{P}(y = 1|\hat{p})$. If

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

then the model fits the data well.

Let $R(\hat{p}) = \mathbb{P}(y = 1|\hat{p})$. The first task is to empirically estimate $R(\hat{p})$, as the true probabilities p are unobserved. In an ideal setting where one has k cases where $\hat{p} = m$, then

$$R(\hat{p}) = \frac{1}{k}\sum_{i=1}^{n} y_i 1(\hat{p}_i = m).$$

However, this ideal setting will likely never happen in interesting applications with continuous covariates. Since this ideal will likely not be satisfied, Esarey and Pierce [2012] use a smoothed local linear regression to estimate $R(\hat{p})$. At each point m, a vector of coefficients γ is chosen to minimize the weighted sum of squared errors,

$$\sum_{i=1}^{n} \left(y_i - (\gamma_0 + \gamma_1(m - \hat{p}_i))\right)^2 K(m - \hat{p}_i), \tag{1}$$

where K is a kernel smoother. The idea of kernel smoothing in this context is for the smoother K to, in a principled way, give large weight to observations with corresponding predicted probabilities that are close to m, and give little weight to observations with corresponding predicted probabilities that are far from m. The authors use the classical loess smoother for their method,

$$K(x) = (1 - |x|^3)^3.$$

With this choice of smoother specified, the estimate of $R(\hat{p})$ at m is given as

$$R(\hat{p} = m) = \hat{\gamma}_0,$$

the prediction for the fitted line at $\hat{p} = m$. This method is performed for several $m \in (0, 1)$. See here for more details on kernal smoothing.

CCSO example

We now load in the CCSO data corresponding to "Other Traffic Offenses", fit the same basic model as before to investigate the effect that demographic variables have on the propensity that a person will spend at least one day in jail, and then use the Esarey and Pierce [2012] method to diagnose the fit of this model. First we fit the model and obtain $\hat{p}$.

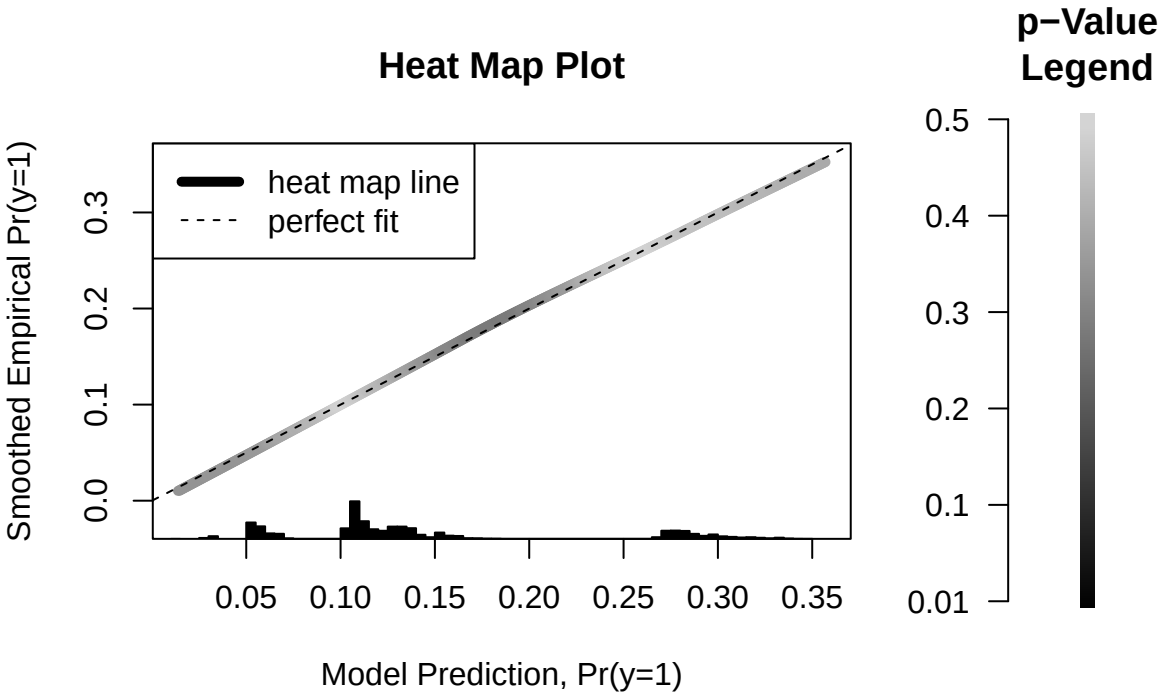
```
library(stat528materials)
data("CCSO")

## 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")) %>%
  select(atleastone, arrestAge, sex, race, bookingDate) %>%
  mutate(race = fct_drop(race), sex = fct_drop(sex))
CCSO_small = CCSO_small[complete.cases(CCSO_small), ]
m1 = glm(atleastone ~ -1 + race + sex + arrestAge,
         data = CCSO_small,
         family = "binomial", x = "TRUE")
p1 = predict(m1, type = "response", se.fit = TRUE)
p1.fit = as.numeric(p1$fit)
y = CCSO_small$atleastone
```

We now call the heatmap.fit function which implements the Esarey and Pierce [2012] method and provides a visual and numerical diagnostic for the fit of our simple model.

```
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%
## ****************************************
```

Quasi-Empirical Residual Distribution Functions (QERDF)

We now present the methodology of Yang [2021] for assessing the fit of regression models with discrete outcomes (note that this methodology has been expanded upon in Yang [2024] and implemented in Lee and Yang [2024]). The method uses "Quasi-empirical residual distribution functions" as the main tool. The idea is to extend the probability integral transform to discrete settings. Recall that if Y is continuous, then for any fixed value $s \in (0, 1)$, we have that

$$\mathbb{P}(F(Y|x) \leq s) = s. \tag{2}$$

For discrete data we have a similar result when $s = F(k|x)$ for some value k, ie,

$$\mathbb{P}\left(Y \leq k | F(k|x) = s\right) = s.$$

The idea in Yang [2021] is to use subsets of the data for which $F(k|x) \approx s$ to estimate $\Pr(Y \leq k|F(k|x) \approx s)$ instead. Definite decisions about which data points to include will not be made. Instead, closeness will be judged by a kernel smoother. This method has a similar intuition to the methodology in Esarey and Pierce [2012].

Theoretical notions of closeness

We define the conditional range of the distribution function given $X = x$ as a grid

$$\Delta(x) = \{F(k|x) : k = 0, 1, ...\}.$$

If X contains continuous components, when X varies in regression, there might be a subset of observations for which $s \in \Delta(x)$ holds approximately. Equivalently, the distance between s and $\Delta(x)$,

$$d(s, \Delta(x)) = \min\{|s - \eta|, \eta \in \Delta(x)\},$$

is small in this case.

Conditioning on $X = x$, denote $F^{(-1)}(\cdot|x)$ as the general inverse function of $F(\cdot|x)$ such that

$$F^{(-1)}(s|x) = \inf\{y : s \leq F(y|x) < 1\} \qquad \text{for} \qquad s \in (0, 1).$$

Here, we exclude $\{1\}$ to avoid boundary effects. Removing this point is not a concern, because Equation (2) always holds on the boundary. Denote

$$H^+(s; x) = F(F^{(-1)}(s|x)|x).$$

It can be shown that $H^+(s; x)$ is the smallest interior point on the grid $\Delta(x)$ that is larger than or equal to s. In a similar manner we can define $H^-(s; x)$ as the largest interior point on the grid $\Delta(x)$ that is smaller than or equal to s.

Now define $H(s; x)$ as the proximal interpolator which maps s to its nearest neighbor on $\Delta(x)$. In other words, $H(s; x)$ is defined to be whichever $H^-(s; x)$ or $H^+(s; x)$ is closest to s. It follows that

$$d(s, \Delta(x) \setminus \{1\}) = |H(s; x) - s|.$$

When s is "close to" being on the grid given x in the sense that $H(s; x) \approx s$, we have an approximation to Equation (2),

$$P(F(Y|x) \leq H(s; x)) = H(s; x) \approx s,$$

QERDF in practice

Now consider a sample (Y_i, X_i^T), $i = 1, ..., n$, where Y is a discrete response variable and X is a vector of covariates containing at least one continuous covariate, and also consider a fixed value of s. To realize the above idea, we use a kernel function $K(\cdot)$ to give large weight to observations whose grid is close to s. Note that several key theoretical quantities in the previous section now require estimation. We will change notation slightly by adding a β vector to make this explicit.

We assign weights to observations depending on the normalized distance between s and $H(s; X_i; \beta)$, that is,

$$K\left(\frac{H(s; X_i; \beta) - s}{\epsilon_n}\right),$$

where ε_n is a small bandwidth. Then, we define the **quasi-empirical residual distribution function** as

$$\hat{U}(s; \beta) = \sum_{i=1}^{n} W_n(s; X_i, \beta) 1\left[F_\beta(Y_i|X_i) \leq H(s; X_i; \beta)\right],$$

where $1(\cdot)$ is the indicator function,

$$W_n(s; X_i, \beta) = \frac{K\left(\frac{H(s;X_i;\beta)-s}{\epsilon_n}\right)}{\sum_{j=1}^{n} K\left(\frac{H(s;X_j;\beta)-s}{\epsilon_n}\right)},$$

and $K(\cdot)$ is a bounded, symmetric, and Lipschitz continuous kernel. In practice, β is unknown; let $\hat{\beta}$ be the corresponding estimator. Then everything in the above can be estimated via plug-in. Asymptotic results and bandwidth selection are beyond the scope of this course. More details can be found in Yang [2021].

Essentially, $\hat{U}(s; \beta)$ is an estimator of the probability

$$\mathbb{P}(Y \leq k|F(k|X) = s) = \mathbb{P}(F(Y|X) \leq s|F(k|X) = s),$$

if there exists such an integer k satisfying the condition $F(k|X) = s$. This probability equals s under the true model.

Numerical examples

We will now consider some numerical examples to demonstrate the effectiveness of the quasi-empirical residual distribution function method for diagnosing Poisson model fits. We first load in a snapshot of the source code from the supplement of Yang [2021].

```
source("yang2021supp.R")
```

We simulate some data to assess performance. In one simulation configuration we will test how the method performs when evaluated with respect to a true data generating model. We will compare this ideal setting to configurations with misspecification.

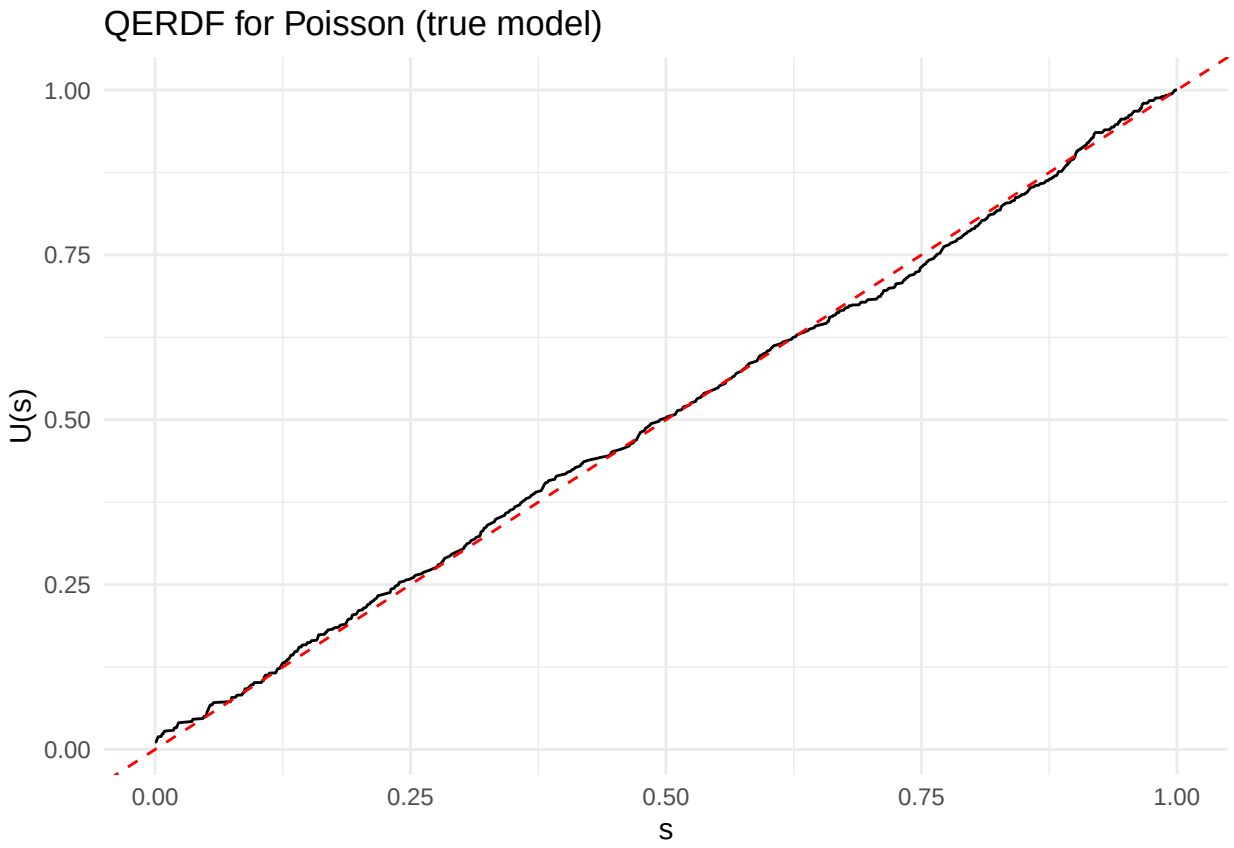
```
## true model
set.seed(13)
n = 500
p = 2
beta = rep(1,p+1)
M = cbind(1, matrix(rnorm(n*p), nrow = n, ncol = p))
Y = rpois(n = n, lambda = exp(M %*% beta))
m1 = glm(Y ~ -1 + M, family = "poisson")
lambdahat = m1$fitted.values
```

We can see that the method indicates that the model fits the data very well, as expected.

```
## calculate bandwidth parameter
h = bandwidthp(y = Y, lambdaf = lambdahat)
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multistart 1 of 1 |Multis
```

```
## diagnostic plot using the marginesti function
data.frame(s = seq(0,1,length = 1001)) %>%
  mutate(y = sapply(s, function(u) {
    marginesti(u, y = Y, lambdaf = lambdahat)
  })) %>%
  ggplot() +
  aes(x = s, y = y) +
  geom_line() +
  geom_abline(col = "red", lty = 2) +
  labs(title = "QERDF for Poisson (true model)",
       x = "s", y = "U(s)") +
  theme_minimal()
```



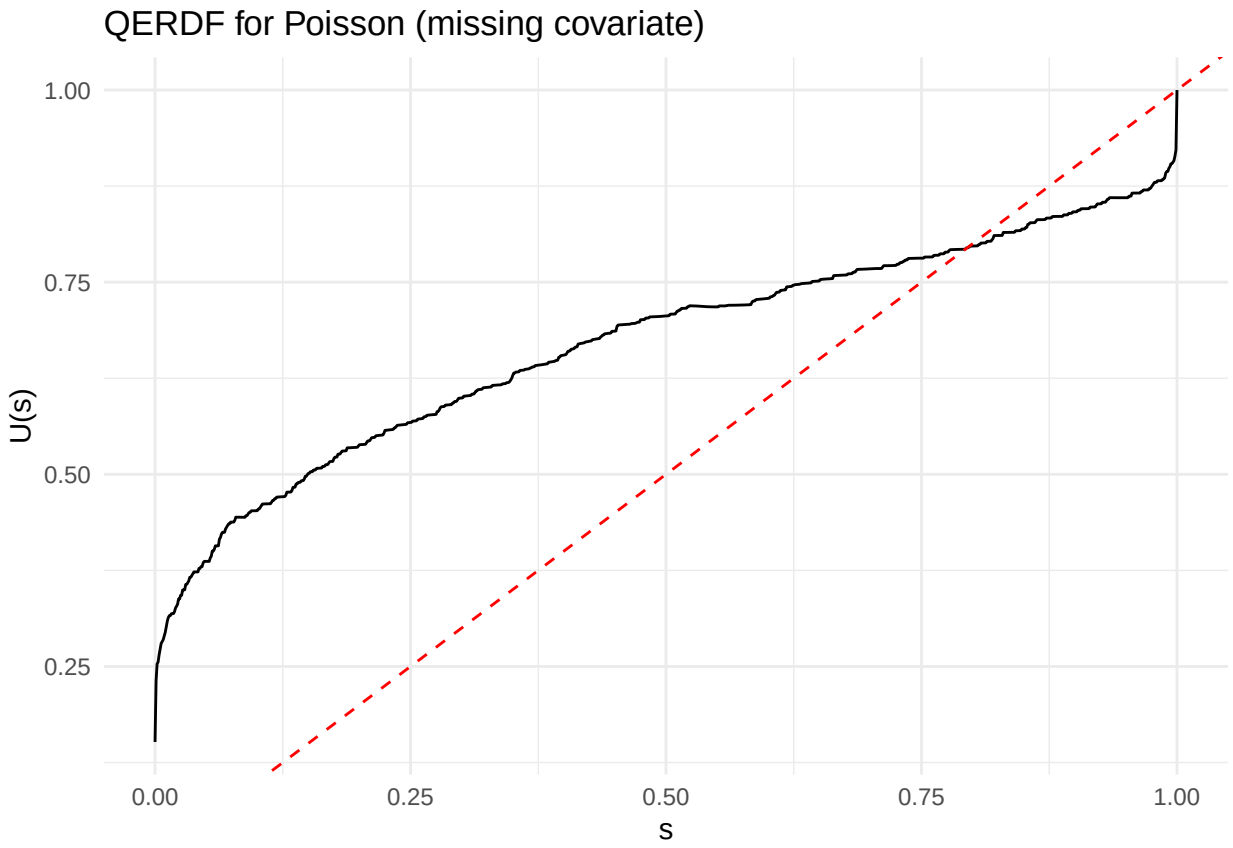
We can see that the model with a predictor removed does not fit the data well, also as expected.

```
m2 = glm(Y ~ -1 + M[, -p], family = "poisson")
lambdahat = m2$fitted.values

## calculate bandwidth parameter
h = bandwidthp(y = Y, lambdaf = lambdahat)
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multistart 1 of 1 |Multis
```

```
## diagnostic plot using the marginesti function
data.frame(s = seq(0,1,length = 1001)) %>%
  mutate(y = sapply(s, function(u) {
    marginesti(u, y = Y, lambdaf = lambdahat)
  })) %>%
  ggplot() +
  aes(x = s, y = y) +
  geom_line() +
  geom_abline(col = "red", lty = 2) +
  labs(title = "QERDF for Poisson (missing covariate)",
       x = "s", y = "U(s)") +
  theme_minimal()
```



Prediction and classification

In this section we briefly demonstrate some prediction and classification metrics that are used for binary and count response models.

Binary response models

Our demonstration will involve an analysis of 2022 baseball data where we model home runs as a function of launch speed (how hard one hits a baseball) and launch angle (the angle of the ball off the bat). Relevant data and software is loaded.

```
library(lubridate)
library(caret)
library(pROC)
library(PresenceAbsence)
dat = read_csv("sc-hr-2022.csv")
head(dat)
```

```
## # A tibble: 6 x 4
##   events    launch_speed launch_angle    HR
##   <chr>            <dbl>        <dbl> <dbl>
## 1 field_out         61.5           41     0
## 2 force_out         95.3          -10     0
## 3 field_out         93.8           34     0
## 4 field_out         83             40     0
## 5 field_out         87.9           -2     0
## 6 field_out         92.3           30     0
```

Confusion matrix

We fit a basic main effects only logistic regression model.

```
m1 = glm(HR ~ launch_speed + launch_angle, data = dat,
         family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(m1)
```

```
##
## Call:
## glm(formula = HR ~ launch_speed + launch_angle, family = "binomial",
##     data = dat)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -37.718109   0.512615  -73.58   <2e-16 ***
## launch_speed   0.327228   0.004677   69.96   <2e-16 ***
## launch_angle   0.085318   0.001510   56.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30074  on 85815  degrees of freedom
## Residual deviance: 17379  on 85813  degrees of freedom
## AIC: 17385
##
## Number of Fisher Scoring iterations: 9
```

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

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

This model fits the data extremely well. We now present a confusion matrix which shows correct and incorrect classifications, the overall classification accuracy, the accuracy that we could achieve if we flipped a weighted coin at random (with probability being the max of the marginal empirical success probability or one minus the marginal empirical success probability), and the Sensitivity and Specificity.

- **Sensitivity (true positive rate)** is the probability of a positive test result, conditioned on the individual truly being positive.
- **Specificity (true negative rate)** is the probability of a negative test result, conditioned on the individual truly being negative.

We can see below that this data has a high no information rate (there are relatively few home runs), and the main effects only model barely outperforms random guessing.

```
preds_m1 = predict(m1, type = "response")
y = dat$HR
confusionMatrix(
  data = as.factor(as.numeric(preds_m1 >= 0.5)),
  reference = as.factor(y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction     0     1
##          0 81610  2949
##          1   575   682
##
##                Accuracy : 0.9589
##                  95% CI : (0.9576, 0.9603)
##     No Information Rate : 0.9577
##     P-Value [Acc > NIR] : 0.035
##
##                   Kappa : 0.263
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9930
##             Specificity : 0.1878
##          Pos Pred Value : 0.9651
##          Neg Pred Value : 0.5426
##              Prevalence : 0.9577
##          Detection Rate : 0.9510
##    Detection Prevalence : 0.9854
##       Balanced Accuracy : 0.5904
##
##        'Positive' Class : 0
##
```

Modeling of homeruns is much improved by the addition of a quadratic term for launch angle.

```
m2 = glm(HR ~ launch_speed + poly(launch_angle, 2),
         data = dat, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
anova(m1, m2, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: HR ~ launch_speed + launch_angle
## Model 2: HR ~ launch_speed + poly(launch_angle, 2)
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     85813      17379
## 2     85812       9823  1     7556 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
preds_m2 = predict(m2, type = "response")
confusionMatrix(
  data = as.factor(as.numeric(preds_m2 >= 0.5)),
  reference = as.factor(y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 81417  1395
##          1   768  2236
##
```

```
##                Accuracy : 0.9748
##                  95% CI : (0.9737, 0.9758)
##     No Information Rate : 0.9577
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.661
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9907
##             Specificity : 0.6158
##          Pos Pred Value : 0.9832
##          Neg Pred Value : 0.7443
##              Prevalence : 0.9577
##          Detection Rate : 0.9487
##    Detection Prevalence : 0.9650
##       Balanced Accuracy : 0.8032
##
##        'Positive' Class : 0
##
```

ROC curve

A receiver operating characteristic (ROC) curve is a plot of the true positive rate (sensitivity) against the false positive rate (specificity) of a classifier as the threshold which differentiates predicted class membership changes. The area under the ROC curve (AUC) is important tool for measuring the accuracy of a classifier.

The following is taken from Nahm [2022]: The AUC is widely used to measure the accuracy of diagnostic tests. The closer the ROC curve is to the upper left corner of the graph, the higher the accuracy of the test because in the upper left corner, the sensitivity = 1 and the false positive rate = 0 (specificity = 1). The ideal ROC curve thus has an AUC = 1.0. However, when the coordinates of the x-axis (1 – specificity) and the y-axis correspond to 1 : 1 (i.e., true positive rate = false positive rate), a graph is drawn on the 45 degree diagonal (y = x) of the ROC curve (AUC = 0.5). Such a situation corresponds to determining the presence or absence of disease by random guessing with probability 0.5. Therefore, for any diagnostic technique to be meaningful, the AUC must be greater than 0.5, and in general, it must be greater than 0.8 to be considered acceptable. In addition, when comparing the performance of two or more diagnostic tests, the ROC curve with the largest AUC is considered to have a better diagnostic performance.

```
roc_m1 = roc(y, preds_m1)
```

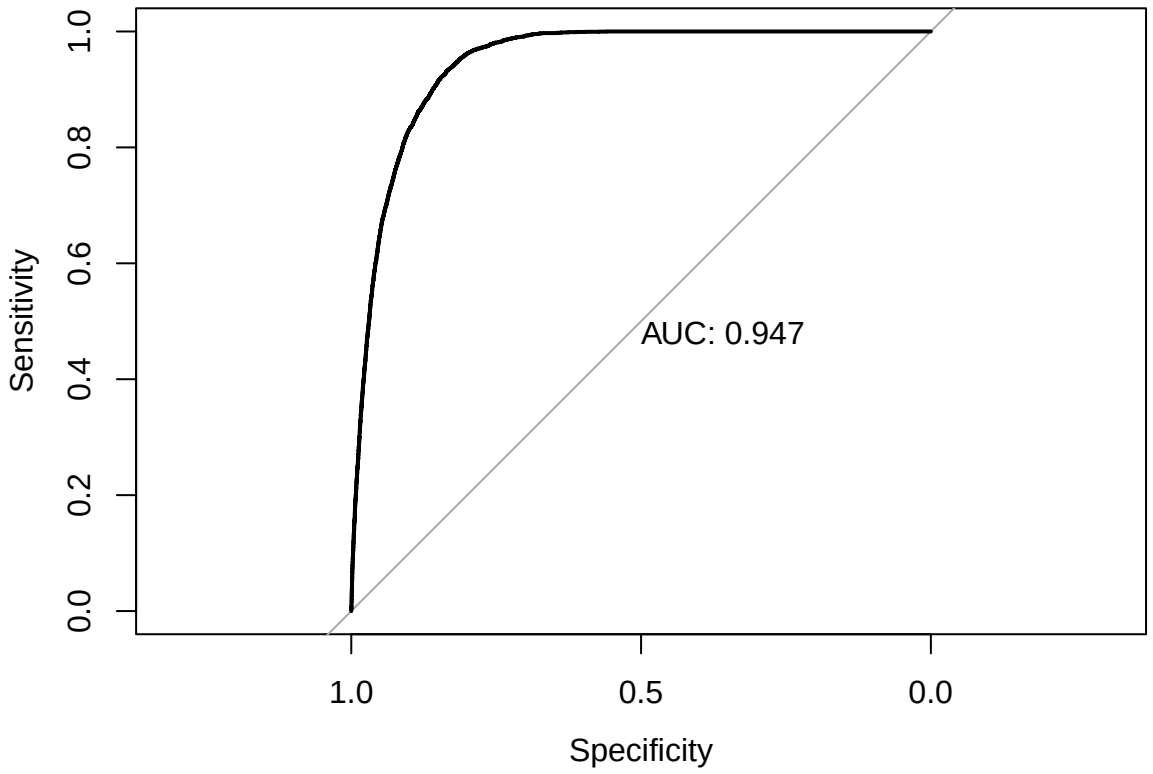
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc_m1
```

```
##
## Call:
## roc.default(response = y, predictor = preds_m1)
##
## Data: preds_m1 in 82185 controls (y 0) < 3631 cases (y 1).
## Area under the curve: 0.9466
```

```
plot(roc_m1, print.auc = TRUE)
```



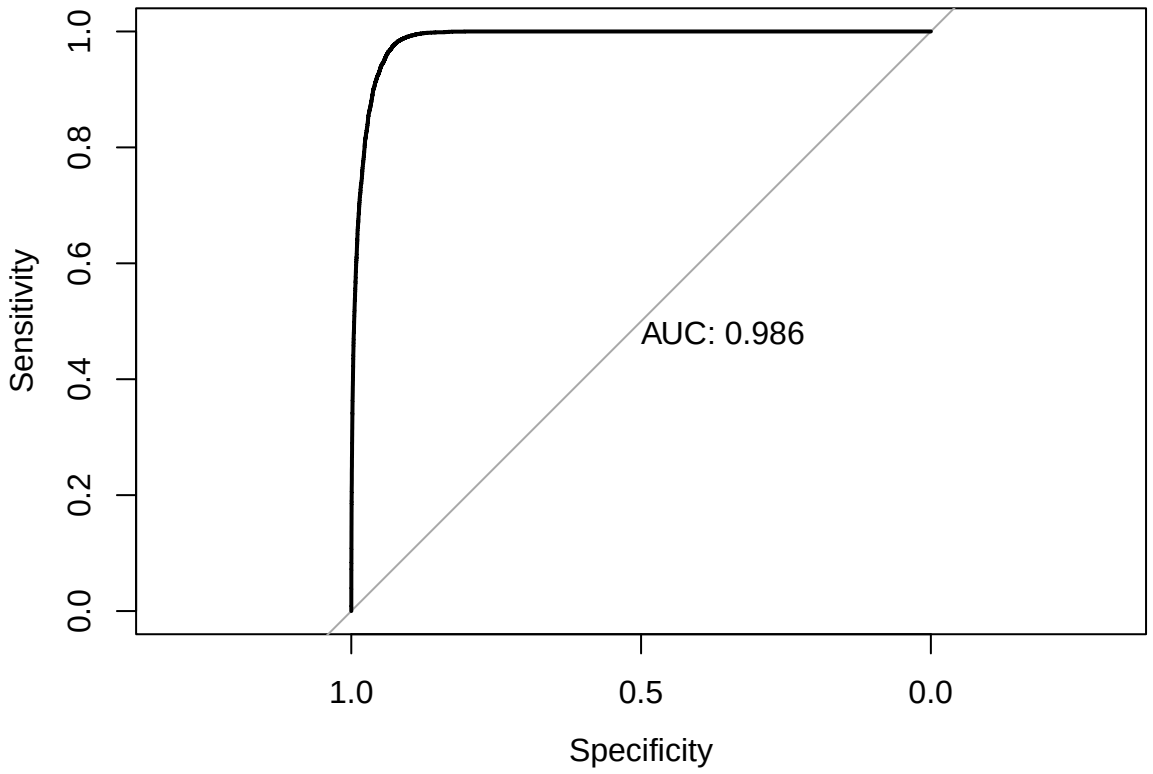

```
roc_m2 = roc(y, preds_m2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
roc_m2
```

```
##
## Call:
## roc.default(response = y, predictor = preds_m2)
##
## Data: preds_m2 in 82185 controls (y 0) < 3631 cases (y 1).
## Area under the curve: 0.9864
```

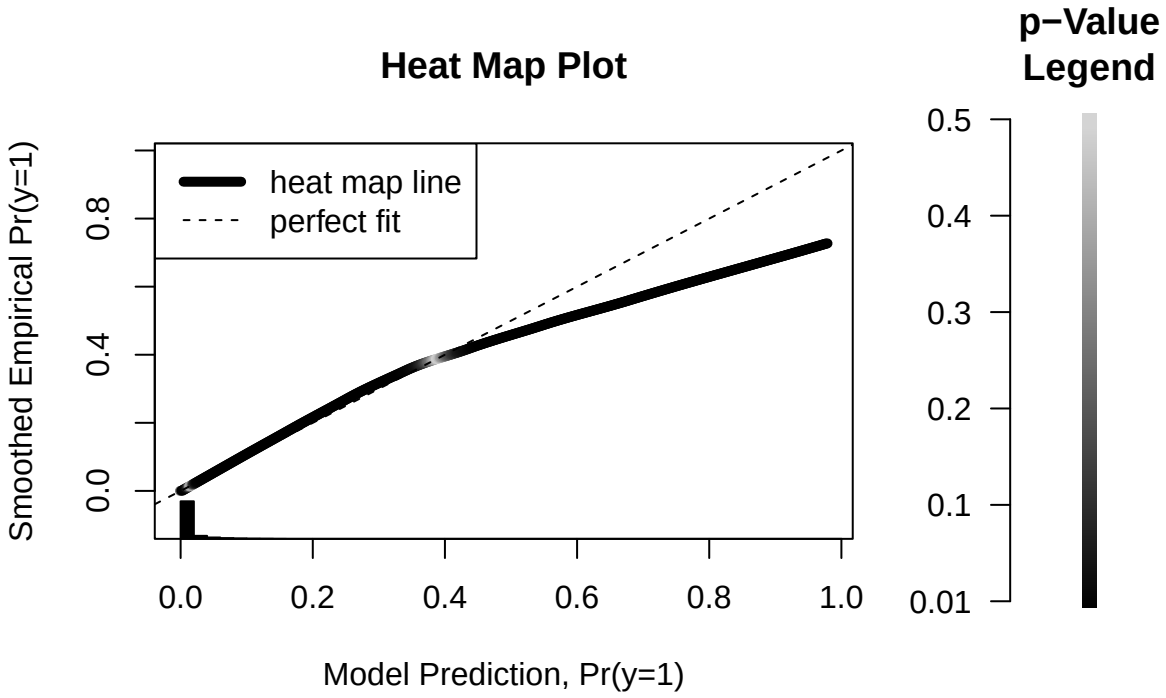
```
plot(roc_m2, print.auc = TRUE)
```



The quadratic model also exhibits better performance according to Esarey and Pierce's method for assessing the fit of binary response models.

```
heatmap.fit(y = y, preds_m1)
```

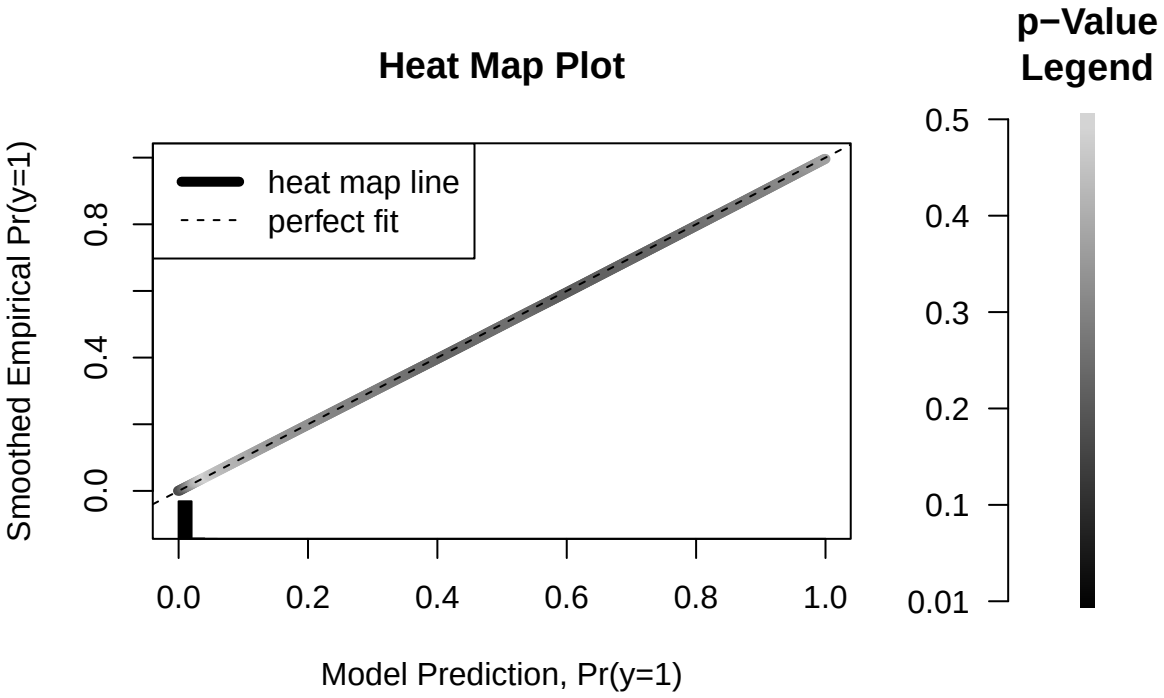
```
## Collapsing large data set into bins based on predicted Pr(y)...
## Data collapsed into 1615 weighted bins.
##
## Calculating optimal loess bandwith...
## aicc Chosen Span =  0.8604987
##
## Generating Bootstrap Predictions...
##   |                                                                      |
```




```
##
##
## ****************************************
## 52.45642% of Observations have one-tailed p-value <= 0.1
## Expected Maximum = 20%
## ****************************************
```

```
heatmap.fit(y = y, preds_m2)
```

```
## Collapsing large data set into bins based on predicted Pr(y)...
## Data collapsed into 1439 weighted bins.
##
## Calculating optimal loess bandwith...
## aicc Chosen Span =  0.9899513
##
## Generating Bootstrap Predictions...
##   |                                                                      |
```




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

Note that our confusion matrix example used 0.5 as the threshold for differentiating predicted class membership. We may be able improve classification accuracy by changing the threshold. The `optimal.thresholds` function in the `PresenceAbsence` package can be useful for finding desirable thresholds.

```
optimal.thresholds(DATA = data.frame(ID = seq_along(nrow(dat)),
                                     obs = y,
                                     pred = preds_m2)) %>%
  as.data.frame()
```

```
## Warning in optimal.thresholds(DATA = data.frame(ID = seq_along(nrow(dat)), :
## req.sens defaults to 0.85
```

```
## Warning in optimal.thresholds(DATA = data.frame(ID = seq_along(nrow(dat)), :
## req.spec defaults to 0.85
```

```
## Warning in optimal.thresholds(DATA = data.frame(ID = seq_along(nrow(dat)), :
## costs assumed to be equal
```

```
##          Method       pred
## 1       Default 0.50000000
## 2     Sens=Spec 0.08000000
## 3  MaxSens+Spec 0.04000000
## 4      MaxKappa 0.38000000
## 5        MaxPCC 0.53000000
```

```
## 6  PredPrev=Obs 0.40000000
## 7        ObsPrev 0.04231146
## 8       MeanProb 0.04231146
## 9     MinROCdist 0.06000000
## 10       ReqSens 0.21000000
## 11       ReqSpec 0.01000000
## 12          Cost 0.53000000
```

Out-of-sample classification

We now investigate the out-of-sample classification performance of our final logistic regression with a quadratic term for launch angle. We split our original data set into a training set and a testing set. The training data is for model fitting. It will contain 75% of observations. The classification accuracy of our trained model will be assessed on the test set.

```
index = sample(1:nrow(dat), size=round(0.75*nrow(dat)), replace = FALSE)
train = dat[index, ]
test = dat[-index, ]
```

We fit our model on the training data.

```
m_train = glm(HR ~ launch_speed + poly(launch_angle, 2),
         data = train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

We get the binary response vector from the test daa set and obtain conditional probability estimates for observations in the test data set.

```
y_test = test$HR
preds_test = predict(m_train, newdata = test)
```

We assess classification accuracy and the sensitivity and specificity of our fitted logistic regression model on the out-of-sample test set.

```
confusionMatrix(
  data = as.factor(as.numeric(preds_test >= 0.5)),
  reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20422   458
##          1   105   469
##
##                Accuracy : 0.9738
##                  95% CI : (0.9715, 0.9759)
##     No Information Rate : 0.9568
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6121
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.9949
##             Specificity : 0.5059
##          Pos Pred Value : 0.9781
##          Neg Pred Value : 0.8171
##              Prevalence : 0.9568
##          Detection Rate : 0.9519
##    Detection Prevalence : 0.9732
##       Balanced Accuracy : 0.7504
##
##        'Positive' Class : 0
##
```

```
roc_test = roc(y_test, preds_test)
```

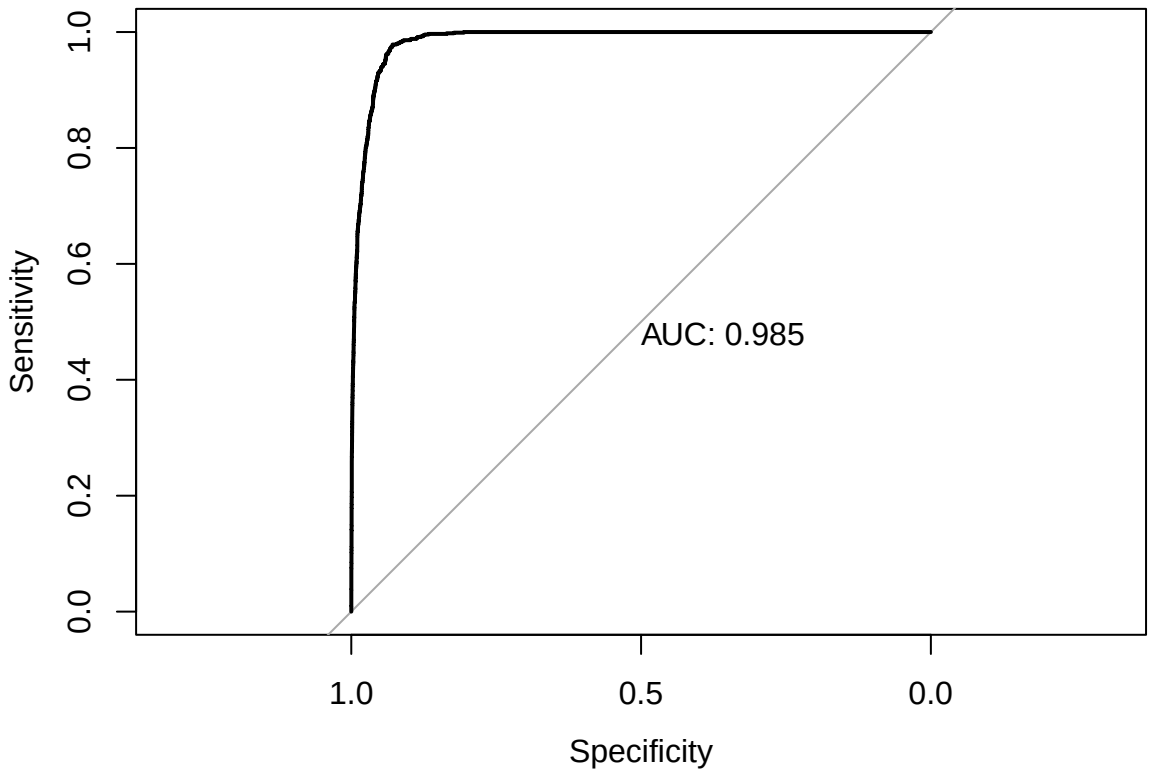
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc_test
```

```
##
## Call:
## roc.default(response = y_test, predictor = preds_test)
##
## Data: preds_test in 20527 controls (y_test 0) < 927 cases (y_test 1).
## Area under the curve: 0.9855
```

```
plot(roc_test, print.auc = TRUE)
```



References

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

Jaehoon Lee and Lu Yang. Assessor: A library for statistical assessment, 2024. URL https://jhlee1408.github.io/assessor/.

Francis Sahngun Nahm. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, 75(1):25–36, 2022.

Lu Yang. Assessment of regression models with discrete outcomes using quasi-empirical residual distribution functions. *Journal of Computational and Graphical Statistics*, 30(4):1019–1035, 2021.

Lu Yang. Double probability integral transform residuals for regression models with discrete outcomes. *Journal of Computational and Graphical Statistics*, 33(3):787–803, 2024.