Lab 9

Adam Shen

November 18, 2020

Packages

```r
library(ResourceSelection)
library(dplyr)
library(broom)
library(ggplot2)
theme_set(theme_bw())
```

Horseshoe crab data

```r
crabs <- read.table("./hcrabs.txt", header=TRUE)
```

Create the response variable

```r
crabs <- crabs %>%
  mutate(y = ifelse(Satellites >= 1, 1, 0)) %>%
  relocate(y, .before=everything())
```

The `.before=everything()` is not really needed since `relocate()` by default moves everything to the front. However, I usually still specify it so that my code is more readable to those who are unfamiliar with the `relocate()` function.

Fit a simple logistic regression model

```r
m1 <- glm(y ~ Width, family=binomial, data=crabs)
summary(m1)
```

```
##
## Call:
## glm(formula = y ~ Width, family = binomial, data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0281  -1.0458   0.5480   0.9066   1.6942
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.3508     2.6287  -4.698 2.62e-06 ***
## Width         0.4972     0.1017   4.887 1.02e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 225.76  on 172  degrees of freedom
## Residual deviance: 194.45  on 171  degrees of freedom
## AIC: 198.45
##
## Number of Fisher Scoring iterations: 4
```

Check for model usefulness

Using model log-likelihoods

We first fit a null model, which will be our reduced model.

```
m0 <- glm(y ~ 1, family=binomial, data=crabs)
```

The hypothesis we wish to test is:

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_A : \beta_1 \neq 0$$

The test statistic is computed as:

$$G^2 = 2(\mathcal{L}_{\text{full}} - \mathcal{L}_{\text{reduced}})$$

where $\mathcal{L}$ represents the log-likelihood of the respective model. We can obtain the log-likelihood of a GLM using the `logLik()` function. You may wish to wrap any calculations involving `logLik()` with `as.numeric()` so that it only displays the value of the log-likelihood.

```
logLik(m0)
```

```
## 'log Lik.' -112.8793 (df=1)
```

```
logLik(m1)
```

```
## 'log Lik.' -97.22633 (df=2)
```

```
as.numeric(logLik(m1))
```

```
## [1] -97.22633
```

The value of the test statistic is

```
(Gsq1 <- as.numeric(2*(logLik(m1) - logLik(m0))))
```

```
## [1] 31.30586
```

For sufficiently large n,

$$G^2 \sim \chi^2_{p-q}$$

where p is the number of parameters in the full model, and q is the number of parameters in the reduced model. Here, $p = 2$ and $q = 1$. The p-value of this test is the area to the right of `Gsq`:

```
pchisq(Gsq1, df=2-1, lower.tail=FALSE)
```

```
## [1] 2.204134e-08
```

Since the p-value is less than 0.05, we reject the null hypothesis. There is evidence to support that $\beta_1 \neq 0$, i.e. the model is useful.

Using deviance

The hypothesis we wish to test is again:

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_A : \beta_1 \neq 0$$

The test statistic is computed as:

$$G^2 = \mathcal{D}_{\text{reduced}} - \mathcal{D}_{\text{full}}$$

where $\mathcal{D}$ represents the deviance of the respective model. We can obtain the deviance of a GLM using the `deviance()` function.

```r
deviance(m0)
```

```
## [1] 225.7585
```

```r
deviance(m1)
```

```
## [1] 194.4527
```

```r
(Gsq2 <- deviance(m0) - deviance(m1))
```

```
## [1] 31.30586
```

Note that `Gsq2` has the same value as `Gsq1` and the same distribution. As such, the p-values and the conclusion of the hypothesis test will be the same.

Note that for a simple logistic regression model, we can also use the `anova` command to get the p-value of this test:

```r
anova(m1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##       Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                    172     225.76
## Width  1   31.306       171     194.45 2.204e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is the same p-value obtained as above, so we will reach the same conclusion.

Augmenting a logistic regression model

With base-R, `fitted()` will return the fitted values on the response scale (probabilities).

```r
head(fitted(m1))
```

```
##         1         2         3         4         5         6
## 0.8482329 0.2380991 0.6404177 0.4951254 0.6404177 0.3736172
```

However, when we use `broom::augment()` with a GLM:

```r
augment(m1)
```

```
## # A tibble: 173 x 8
##        y Width .fitted .resid .std.resid    .hat .sigma .cooksd
##    <dbl> <dbl>   <dbl>  <dbl>      <dbl>   <dbl>  <dbl>   <dbl>
##  1     1  28.3  1.72    0.574      0.577 0.0123    1.07 0.00113
##  2     0  22.5 -1.16   -0.737     -0.747 0.0257    1.07 0.00423
##  3     1  26    0.577   0.944      0.947 0.00708   1.07 0.00202
##  4     0  24.8 -0.0195 -1.17      -1.18  0.0101    1.07 0.00503
##  5     1  26    0.577   0.944      0.947 0.00708   1.07 0.00202
##  6     0  23.8 -0.517  -0.967     -0.975 0.0166    1.07 0.00512
##  7     0  26.5  0.826  -1.54      -1.55  0.00754   1.06 0.00874
##  8     0  24.7 -0.0692 -1.15      -1.15  0.0106    1.07 0.00504
##  9     0  23.7 -0.566  -0.948     -0.956 0.0174    1.07 0.00510
## 10     0  25.6  0.378  -1.34      -1.35  0.00739   1.06 0.00547
## # ... with 163 more rows
```

It should be noted that the values in the `.fitted` column are on the predictor scale (log odds) rather than on the response scale (probabilities). This is quite evident as we cannot have probabilities that are negative or greater than 1. Looking at the documentation using both `?broom::augment.glm` and `?stats::predict.glm`, we see that we need to make the adjustment:

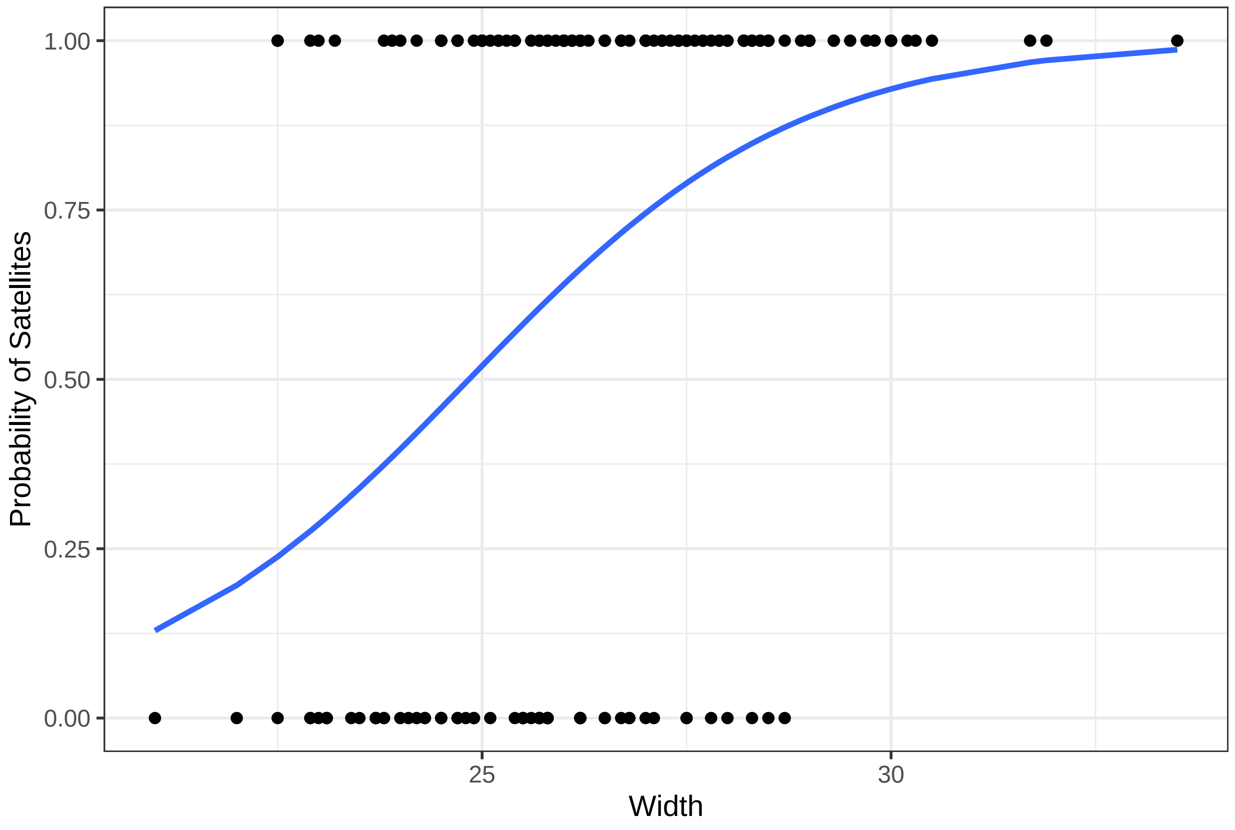
```
augment(m1, type.predict="response")
```

```
## # A tibble: 173 x 8
##        y Width .fitted .resid .std.resid    .hat .sigma .cooksd
##    <dbl> <dbl>   <dbl>  <dbl>      <dbl>   <dbl>  <dbl>   <dbl>
##  1     1  28.3   0.848  0.574      0.577 0.0123    1.07 0.00113
##  2     0  22.5   0.238 -0.737     -0.747 0.0257    1.07 0.00423
##  3     1  26     0.640  0.944      0.947 0.00708   1.07 0.00202
##  4     0  24.8   0.495 -1.17      -1.18  0.0101    1.07 0.00503
##  5     1  26     0.640  0.944      0.947 0.00708   1.07 0.00202
##  6     0  23.8   0.374 -0.967     -0.975 0.0166    1.07 0.00512
##  7     0  26.5   0.695 -1.54      -1.55  0.00754   1.06 0.00874
##  8     0  24.7   0.483 -1.15      -1.15  0.0106    1.07 0.00504
##  9     0  23.7   0.362 -0.948     -0.956 0.0174    1.07 0.00510
## 10     0  25.6   0.593 -1.34      -1.35  0.00739   1.06 0.00547
## # ... with 163 more rows
```

Visualize the fit

```
m1_aug <- augment(m1, type.predict = "response")

ggplot(m1_aug, aes(x=Width))+
  geom_point(aes(y=y))+
  geom_line(aes(y=.fitted), colour="#3366FF", size=1)+
  labs(y="Probability of Satellites")
```



Variance-covariance matrix

```
vcov(m1)
```

```
##             (Intercept)       Width
## (Intercept)   6.9101576 -0.26684761
## Width        -0.2668476  0.01035012
```

Point estimate and large sample CI for odds ratio

Recall that our fitted model has form:

$$\ln\left(\frac{\hat{\pi}_i}{1-\hat{\pi}_i}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

This means that a unit change in x_i will result in a change in **log-odds** by $\hat{\beta}_1$ units. If we are interested in the change in **odds** with respect to a unit change in x_i, we should exponentiate both sides to get rid of the ln.

```
exp(coef(m1))
```

```
##  (Intercept)        Width
## 4.326214e-06 1.644162e+00
```

Large sample confidence intervals (i.e. using standard normal distribution) can be computed using:

```
exp(confint.default(m1))
```

```
##                    2.5 %       97.5 %
## (Intercept) 2.503452e-08 0.0007476128
## Width       1.346936e+00 2.0069749360
```

Point estimate and large sample CI for probabilities

```
# Extract coefficients
b <- coef(m1)

# Compute point estimate on predictor scale: eta = xh' %*% beta
xh <- c(1, 26.5)
xht <- t(xh)
logit_pihat <- xht %*% b

# Convert point estimate to response scale
pihat <- exp(logit_pihat)/(1+exp(logit_pihat))

# Calculate SE of point estimate on predictor scale
v_logit_pihat <- xht %*% vcov(m1) %*% xh
se_logit_pihat <- sqrt(v_logit_pihat)

# Compute lower and upper bounds on predictor scale
lower_logitpi <- logit_pihat - 1.96*se_logit_pihat
upper_logitpi <- logit_pihat + 1.96*se_logit_pihat

# Convert lower and upper bounds to response scale
lower_pi <- exp(lower_logitpi)/(1+exp(lower_logitpi))
upper_pi <- exp(upper_logitpi)/(1+exp(upper_logitpi))

# Display results
c(fit=pihat, lower=lower_pi, upper=upper_pi)
```

```
##       fit     lower     upper
## 0.6954646 0.6120528 0.7677476
```

As an alternative, to go from the predictor scale (log odds) to the response scale (probabilities), we can use `plogis()`:

$$\hat{\pi} = \texttt{plogis}(\hat{\eta}) = \frac{e^{\hat{\eta}}}{1 + e^{\hat{\eta}}}$$

To do the opposite, i.e. response scale (probabilities) to predictor scale (log odds), we can use `qlogis()`:

$$\hat{\eta} = \texttt{qlogis}(\hat{\pi}) = \ln\left(\frac{\hat{\pi}}{1 - \hat{\pi}}\right)$$

In the above example, we need to convert values on the predictor scale (log odds) to the response scale (probabilities) so we need to use `plogis()`. Comparing the output using math with using `plogis`:

```
matrix(
  c(pihat, lower_pi, upper_pi,
    plogis(logit_pihat), plogis(lower_logitpi), plogis(upper_logitpi)),
  nrow=3, ncol=2,
  dimnames = list(c("estimate", "lower", "upper"), c("Math", "plogis"))
)
```

```
##               Math    plogis
## estimate 0.6954646 0.6954646
## lower    0.6120528 0.6120528
## upper    0.7677476 0.7677476
```

With 95% confidence, the probability of at least one satellite for a crab with shell width 26.5 cm is between 0.612 and 0.768.

Hosmer-Lemeshow test for goodness of fit

```
(htest <- with(m1_aug, hoslem.test(y, .fitted, g=10)))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  y, .fitted
## X-squared = 4.3855, df = 8, p-value = 0.8208
```

The hypotheses we are testing are:

$$H_0 : \pi_i = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)} \quad \text{vs} \quad H_A : \pi_i \neq \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}$$

The p-value of this test is greater than 0.05. We fail to reject the null hypothesis at the 5% level of significance and conclude that there is insufficient evidence that suggests a lack of fit.

```
names(htest)
```

```
## [1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
## [7] "expected"
```

```
htest[c(1,2,3,4,6,7)]
```

```
## $statistic
## X-squared
##  4.385541
##
## $parameter
## df
##  8
```

```
##
## $p.value
## [1] 0.8207722
##
## $method
## [1] "Hosmer and Lemeshow goodness of fit (GOF) test"
##
## $observed
##
## cutyhat         y0 y1
##   [0.129,0.362] 14  5
##   (0.362,0.458] 10  8
##   (0.458,0.527]  5 10
##   (0.527,0.605] 10  9
##   (0.605,0.652]  5 11
##   (0.652,0.716]  7 11
##   (0.716,0.785]  4 12
##   (0.785,0.842]  4 16
##   (0.842,0.888]  3 15
##   (0.888,0.987]  0 14
##
## $expected
##
## cutyhat              yhat0      yhat1
##   [0.129,0.362] 13.6106317  5.3893683
##   (0.362,0.458] 10.3756848  7.6243152
##   (0.458,0.527]  7.4488451  7.5511549
##   (0.527,0.605]  8.0174489 10.9825511
##   (0.605,0.652]  5.9045476 10.0954524
##   (0.652,0.716]  5.7010064 12.2989936
##   (0.716,0.785]  3.9407017 12.0592983
##   (0.785,0.842]  3.7332657 16.2667343
##   (0.842,0.888]  2.3498224 15.6501776
##   (0.888,0.987]  0.9180457 13.0819543
```

The values required to compute the value of the test statistic by hand can be found within the the `observed` and `expected` components of `htest`. A demonstration of what these values represent can be found in *Module 9.9*.