Week 10 - Homework

STAT 420, Summer 2023, D. Unger

Scott Girten NetID: sgirten2

Directions

Students are encouraged to work together on homework. However, sharing, copying or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible.

- Be sure to remove this section if you use this `.Rmd` file as a template.
- You may leave the questions in your final document.

Exercise 1 (Simulating Wald and Likelihood Ratio Tests)

In this exercise we will investigate the distributions of hypothesis tests for logistic regression. For this exercise, we will use the following predictors.

```
sample_size = 150
set.seed(120)
x1 = rnorm(n = sample_size)
x2 = rnorm(n = sample_size)
x3 = rnorm(n = sample_size)
```

Recall that

$$p(\mathbf{x}) = P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

Consider the true model

$$\log\left(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1$$

where

- $\beta_0 = 0.4$
- $\beta_1 = -0.35$

(a) To investigate the distributions, simulate from this model 2500 times. To do so, calculate

$$P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

for an observation, and then make a random draw from a Bernoulli distribution with that success probability. (Note that a Bernoulli distribution is a Binomial distribution with parameter $n = 1$. There is no direction function in R for a Bernoulli distribution.)

Each time, fit the model:

$$\log\left(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

Store the test statistics for two tests:

- The Wald test for $H_0 : \beta_2 = 0$, which we say follows a standard normal distribution for "large" samples
- The likelihood ratio test for $H_0 : \beta_2 = \beta_3 = 0$, which we say follows a χ^2 distribution (with some degrees of freedom) for "large" samples

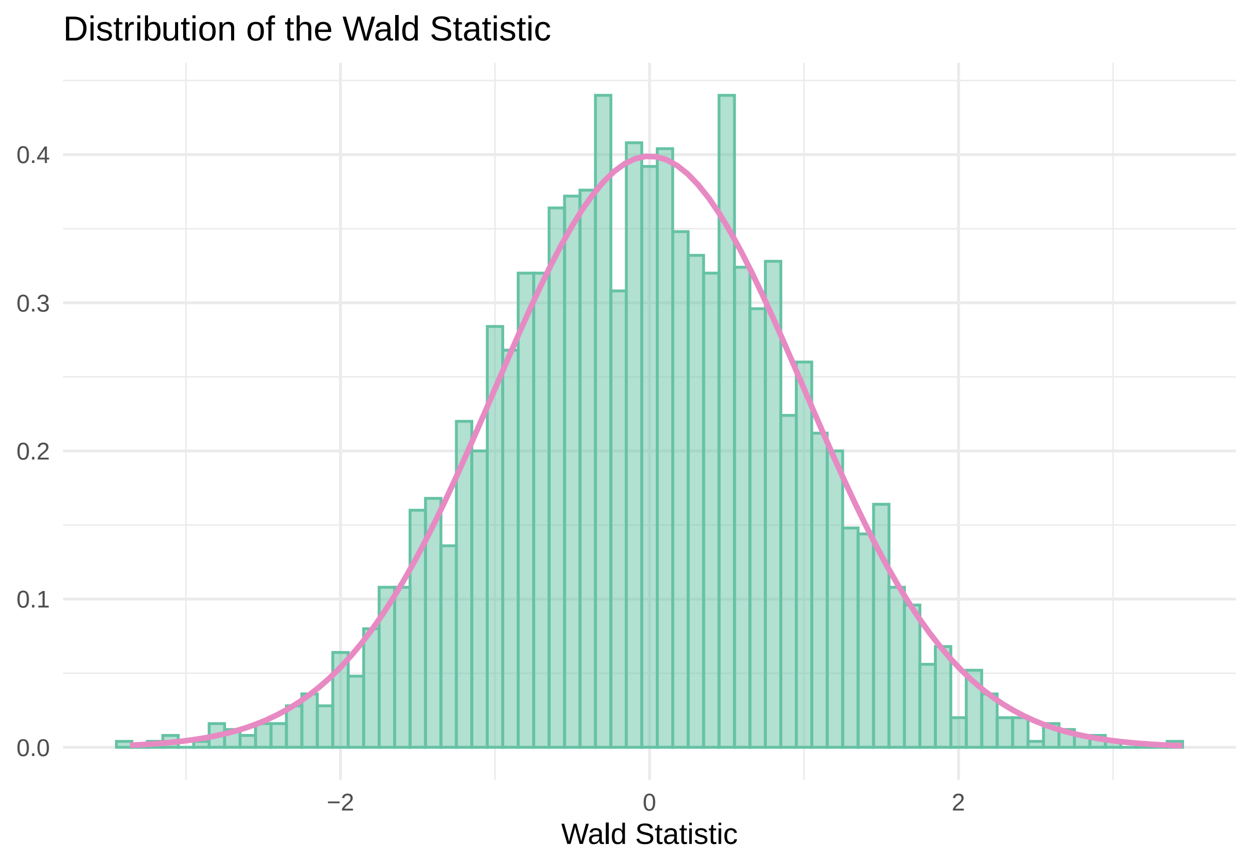
Solution:

```
library(tidyverse)


# Function for simulating logistic data
sim_logistic_data = function(sample_size = 150, beta_0 = 0.4, beta_1 = -0.35, beta_2 = 0, beta_3 = 0){

  eta = beta_0 + beta_1*x1 + beta_2*x2  + beta_3*x3

  p = 1 / (1 + exp(-eta))
  y = rbinom(n = sample_size, size = 1, prob = p)

  return(data.frame(y, x1, x2, x3))
}


# Number of simulations
iter = 2500


# Dataframe for storing results
results = tibble('Wald' = rep(0,iter),
                 'Likelihood Ratio' = rep(0, iter))

# Simulation
for(i in 1:iter){
  # Simulate logistic data
  sim_data = sim_logistic_data()

  # Fit models
  fit_glm = glm(y ~ x1 + x2 + x3, data = sim_data, family = binomial)
  fit_glm_small = glm(y ~ x1, data = sim_data, family = binomial)
  # Wald test
  wald_b2 = summary(fit_glm)$coefficients[3,3]

  # Likelihood Ratio
  like_ratio = anova(fit_glm_small, fit_glm, test = 'LRT')[2,4]

  # Update results
  results$Wald[i] = wald_b2
  results$`Likelihood Ratio`[i] = like_ratio

}
```

(b) Plot a histogram of the empirical values for the Wald test statistic. Overlay the density of the true distribution assuming a large sample.

Solution:

```
p_wald = ggplot(results, aes(x = Wald)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, color = '#66c2a5', fill = '#66c2a5', alpha = 0.5
  stat_function(fun = dnorm, args = list(mean=0, sd=1), color = '#e78ac3', size = 1) +
  theme_minimal() +
  labs(x = 'Wald Statistic',
       y = NULL,
       title = 'Distribution of the Wald Statistic')

p_wald
```



(c) Use the empirical results for the Wald test statistic to estimate the probability of observing a test statistic larger than 1. Also report this probability using the true distribution of the test statistic assuming a large sample.

Solution:

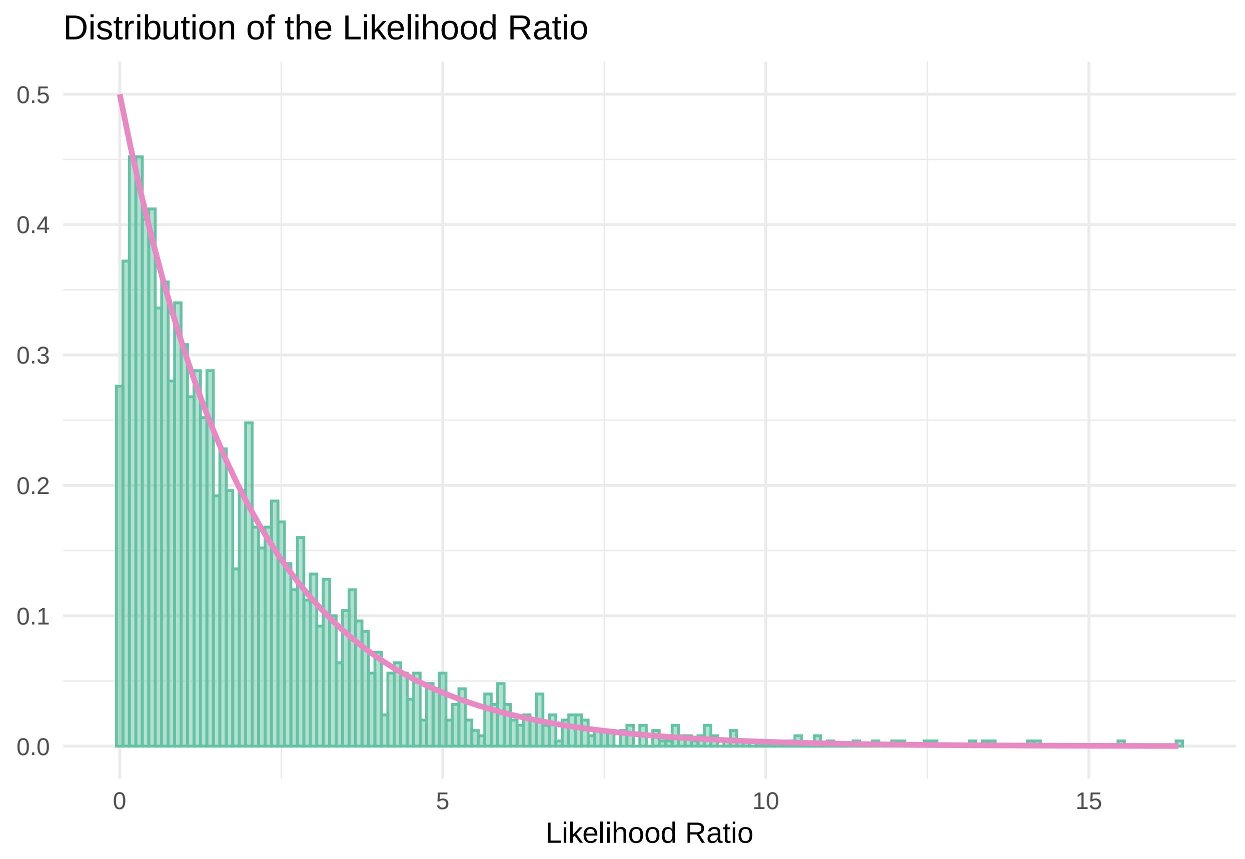
```
empirical_prob = length(results$Wald[results$Wald > 1]) / iter
true_prob = pnorm(1, lower.tail = FALSE)
```

The probability of observing a Wald test statistic larger than 1 from the empirical results is 0.1524 and the same probability from the true distribution is 0.1587.

(d) Plot a histogram of the empirical values for the likelihood ratio test statistic. Overlay the density of the true distribution assuming a large sample.

Solution:

```
p_llr = ggplot(results, aes(x = `Likelihood Ratio`)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, color = '#66c2a5', fill = '#66c2a5', alpha = 0.5
  stat_function(fun = dchisq, args = list(df = 2), color = '#e78ac3', size = 1) +
  theme_minimal() +
  labs(x = 'Likelihood Ratio',
       y = NULL,
       title = 'Distribution of the Likelihood Ratio')

p_llr
```



(e) Use the empirical results for the likelihood ratio test statistic to estimate the probability of observing a test statistic larger than 5. Also report this probability using the true distribution of the test statistic assuming a large sample.

Solution:

```r
empirical_prob2 = length(results$`Likelihood Ratio`[results$`Likelihood Ratio` > 5]) / iter
true_prob2 = pchisq(5,df = 2, lower.tail = FALSE)
```

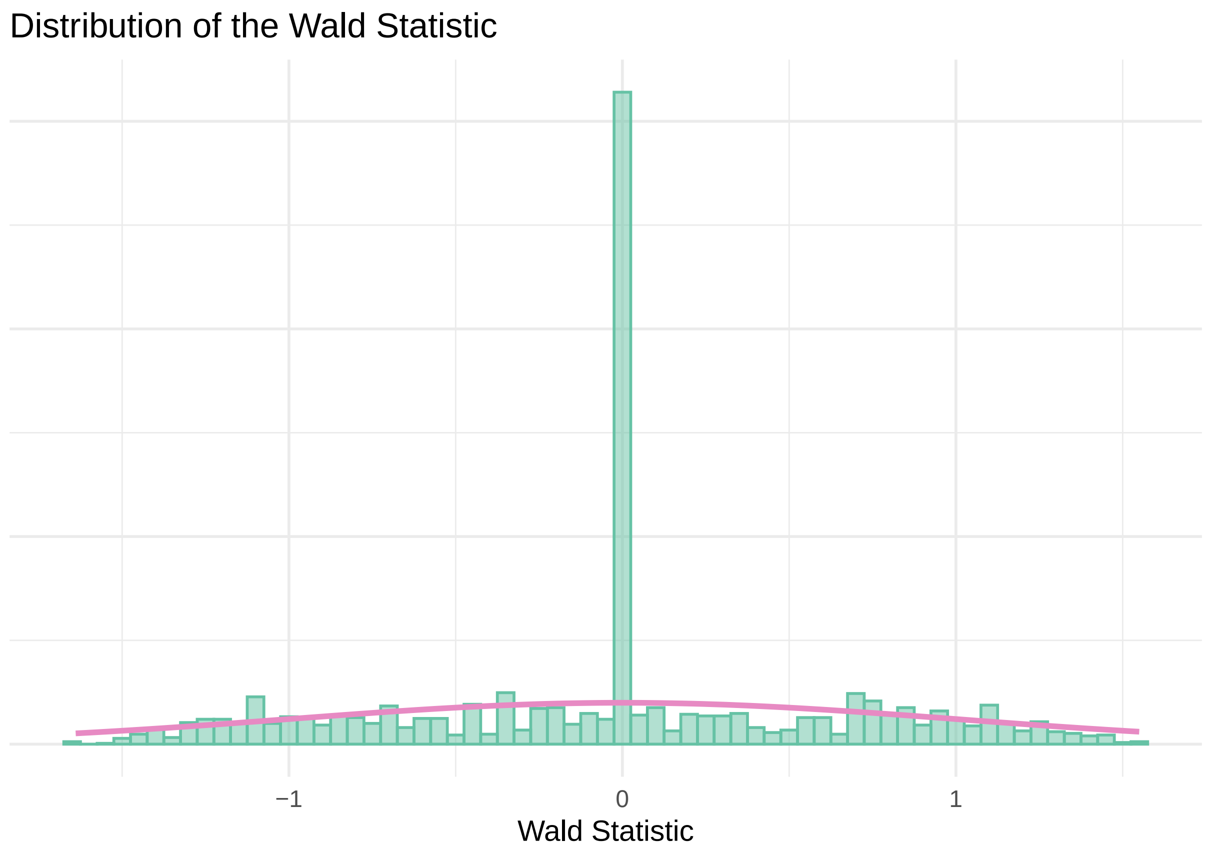
The probability of observing a χ^2 test statistic larger than 5 from the empirical results is 0.0872 and the same probability from the true distribution is 0.0821.

(f) Repeat **(a)**-**(e)** but with simulation using a smaller sample size of 10. Based on these results, is this sample size large enough to use the standard normal and χ^2 distributions in this situation? Explain.

```r
sample_size = 10
set.seed(120)
x1 = rnorm(n = sample_size)
x2 = rnorm(n = sample_size)
x3 = rnorm(n = sample_size)
```

Solution:

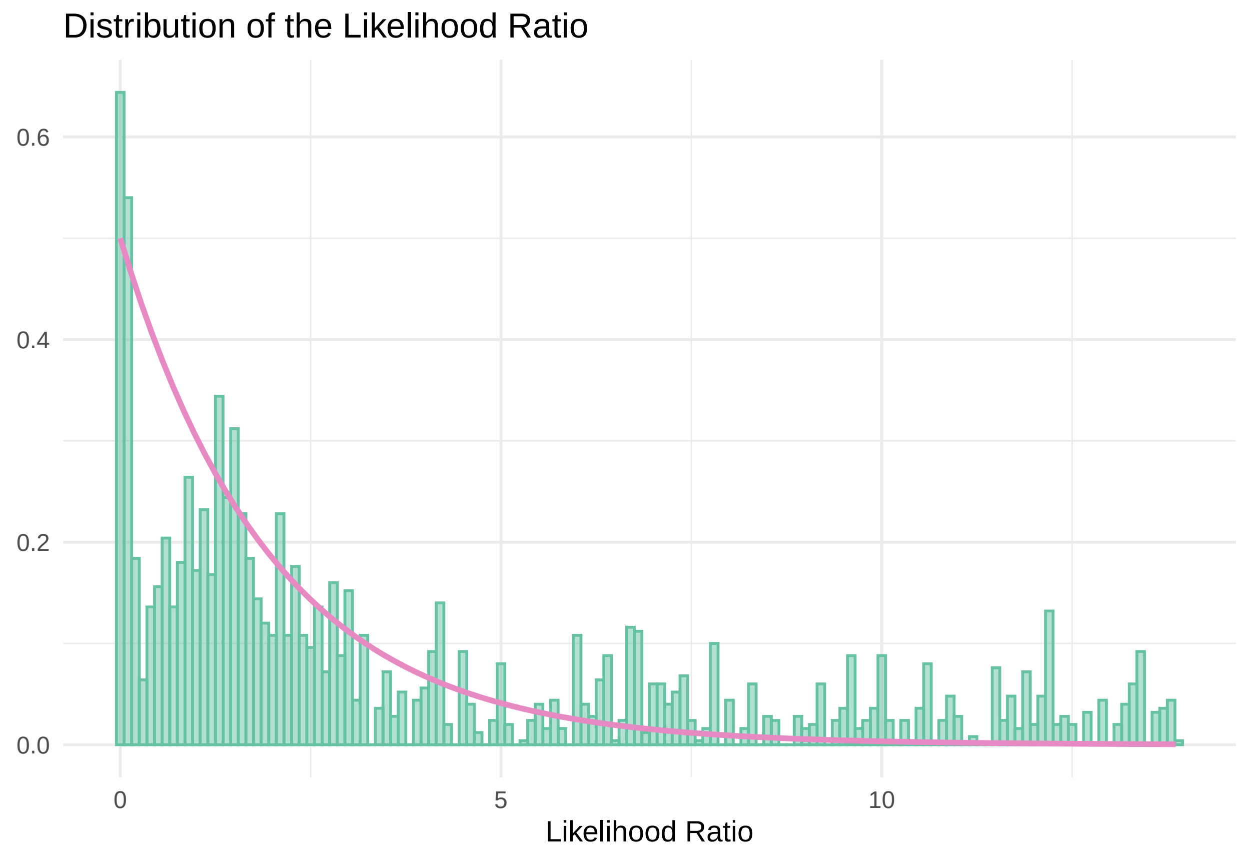
```r
# Dataframe for storing results
results2 = tibble('Wald' = rep(0,iter),
                 'Likelihood Ratio' = rep(0, iter))

# Simulation
for(i in 1:iter){
  # Simulate logistic data
  sim_data = sim_logistic_data(sample_size = sample_size)

  # Fit models
  fit_glm = glm(y ~ x1 + x2 + x3, data = sim_data, family = binomial)
  fit_glm_small = glm(y ~ x1, data = sim_data, family = binomial)

  # Wald test
  wald_b2 = summary(fit_glm)$coefficients[3,3]

  # Likelihood Ratio
  like_ratio = anova(fit_glm_small, fit_glm, test = 'LRT')[2,4]

  # Update results
  results2$Wald[i] = wald_b2
  results2$`Likelihood Ratio`[i] = like_ratio

}

# Histogram of Wald statistic
p_wald2 = ggplot(results2, aes(x = Wald)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.05, color = '#66c2a5', fill = '#66c2a5', alpha = 0.
  stat_function(fun = dnorm, args = list(mean=0, sd=1), color = '#e78ac3', size = 1) +
  theme_minimal() +
  labs(x = 'Wald Statistic',
       y = NULL,
       title = 'Distribution of the Wald Statistic')

p_wald2
```



```
empirical_prob = length(results2$Wald[results2$Wald > 1]) / iter
true_prob = pnorm(1, lower.tail = FALSE)
```

The probability of observing a Wald test statistic larger than 1 from the empirical results is 0.0804 and the same probability from the true distribution is 0.1587.

```
p_llr2 = ggplot(results2, aes(x = `Likelihood Ratio`)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, color = '#66c2a5', fill = '#66c2a5', alpha = 0.5
  stat_function(fun = dchisq, args = list(df = 2), color = '#e78ac3', size = 1) +
  theme_minimal() +
  labs(x = 'Likelihood Ratio',
       y = NULL,
       title = 'Distribution of the Likelihood Ratio')

p_llr2
```



```
empirical_prob2 = length(results2$`Likelihood Ratio`[results2$`Likelihood Ratio` > 5]) / iter
true_prob2 = pchisq(5,df = 2, lower.tail = FALSE)
```

The probability of observing a χ^2 test statistic larger than 5 from the empirical results is 0.3016 and the same probability from the true distribution is 0.0821.

A sample size of 10 is not large enough to use for the standard normal and χ^2 distributions in this example. The empirical distributions for the larger sample size of 150 closely resembled the theoretical distributions for both the Wald statistic and the Likelihood Ratio. However, with a smaller sample size of 10 the empirical distributions of both statistics do not align well with the theoretical distributions of those statistics.

Exercise 2 (Surviving the Titanic)

For this exercise use the `ptitanic` data from the `rpart.plot` package. (The `rpart.plot` package depends on the `rpart` package.) Use `?rpart.plot::ptitanic` to learn about this dataset. We will use logistic regression to help predict which passengers aboard the Titanic will survive based on various attributes.

```
# install.packages("rpart")
# install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
data("ptitanic")
```

For simplicity, we will remove any observations with missing data. Additionally, we will create a test and train dataset.

```r
ptitanic = na.omit(ptitanic)
set.seed(2021)
trn_idx = sample(nrow(ptitanic), 300)
ptitanic_trn = ptitanic[trn_idx, ]
ptitanic_tst = ptitanic[-trn_idx, ]
```

(a) Consider the model

$$\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_3 x_4$$

where

$$p(\mathbf{x}) = P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

is the probability that a certain passenger survives given their attributes and

- x_1 is a dummy variable that takes the value 1 if a passenger was 2nd class.
- x_2 is a dummy variable that takes the value 1 if a passenger was 3rd class.
- x_3 is a dummy variable that takes the value 1 if a passenger was male.
- x_4 is the age in years of a passenger.

Fit this model to the training data and report its deviance.

Solution:

```r
library(tidyverse)
ptitanic_trn = ptitanic_trn %>%
  mutate(x_1 = if_else(pclass == '2nd', 1, 0),
         x_2 = if_else(pclass == '3rd', 1, 0),
         x_3 = if_else(sex == 'male', 1, 0),
         x_4 = age,
         y = if_else(survived == 'survived', 1, 0))

mod_titanic = glm(y ~ x_1 + x_2 + x_3  + x_4 + x_3*x_4, data = ptitanic_trn, family = binomial)

dev = summary(mod_titanic)$deviance
```

The deviance for the fitted training model is 259.1653.

(b) Use the model fit in **(a)** and an appropriate statistical test to determine if class played a significant role in surviving on the Titanic. Use $\alpha = 0.01$. Report:

- The null hypothesis of the test
- The test statistic of the test
- The p-value of the test
- A statistical decision
- A practical conclusion

Solution:

```
mod_titanic_small = glm(y ~ x_3  + x_4, data = ptitanic_trn, family = binomial)

anova_small = anova(mod_titanic_small, mod_titanic, test = 'LRT')
#anova_small
pval = anova_small[2,5]
d_stat = anova_small[2,4]
```

$H_0 : \beta_1 = \beta_2 = 0$

$D := 53.2895$

$p - value : 1.5905 \times 10^{-11}$

Decision: Reject

At a level of $\alpha = 0.01$ I reject the null hypothesis that $H_0 : \beta_1 = \beta_2 = 0$ and assume that either β_1 or β_2 are not equal to 0.

(c) Use the model fit in **(a)** and an appropriate statistical test to determine if an interaction between age and sex played a significant role in surviving on the Titanic. Use $\alpha = 0.01$. Report:

- The null hypothesis of the test
- The test statistic of the test
- The p-value of the test
- A statistical decision
- A practical conclusion

Solution:

```
wald_stat = summary(mod_titanic)$coefficients[6,3]
wald_pval = summary(mod_titanic)$coefficients[6,4]
```

$H_0 : \beta_5 = 0$

$W :=$ -3.2012

$p - value : 0.0014$

Decision: Fail to Reject

At a level of $\alpha = 0.01$ I fail to reject the null hypothesis that $H_0 : \beta_5 = 0$ and assume that the interaction of gender and age is not significant in the model.

(d) Use the model fit in **(a)** as a classifier that seeks to minimize the misclassification rate. Classify each of the passengers in the test dataset. Report the misclassification rate, the sensitivity, and the specificity of this classifier. (Use survived as the positive class.)

Solution:

```
library(kableExtra)

# Create model variables
ptitanic_tst = ptitanic_tst %>%
    mutate(x_1 = if_else(pclass == '2nd', 1, 0),
          x_2 = if_else(pclass == '3rd', 1, 0),
          x_3 = if_else(sex == 'male', 1, 0),
          x_4 = age,
```

Table 1: Model Performance

Metric	Value
Misclassification Rate	0.2172
Sensitivity	0.5714
Specificity	0.9374

```r
        y = if_else(survived == 'survived', 1, 0))


# Misclassification rate
misclassify = mean(if_else(predict(mod_titanic, newdata = ptitanic_tst) > 0, 'survived', 'died') != pti

# Confusion matrix
titanic_pred = if_else(predict(mod_titanic, newdata = ptitanic_tst, type = 'response') > 0.5,  'survive
conf_mat = table(predicted = titanic_pred, actual = ptitanic_tst$survived)

# Sensitivity
sens = conf_mat[2,2] / sum(conf_mat[,2])

# Specificity
spec = conf_mat[1,1] / sum(conf_mat[,1])

# Table for output
display_table = tibble('Metric' = c('Misclassification Rate', 'Sensitivity', 'Specificity'),
                       'Value' = c(misclassify, sens, spec))

display_table %>%
  kbl(caption = 'Model Performance') %>%
  kable_styling()
```

Exercise 3 (Breast Cancer Detection)

For this exercise we will use data found in wisc-train.csv and wisc-test.csv, which contain train and test data, respectively. `wisc.csv` is provided but not used. This is a modification of the Breast Cancer Wisconsin (Diagnostic) dataset from the UCI Machine Learning Repository. Only the first 10 feature variables have been provided. (And these are all you should use.)

- UCI Page
- Data Detail

You should consider coercing the response to be a factor variable if it is not stored as one after importing the data.

(a) The response variable `class` has two levels: `M` if a tumor is malignant, and `B` if a tumor is benign. Fit three models to the training data.

- An additive model that uses `radius`, `smoothness`, and `texture` as predictors

- An additive model that uses all available predictors
- A model chosen via backwards selection using AIC. Use a model that considers all available predictors as well as their two-way interactions for the start of the search.

For each, obtain a 5-fold cross-validated misclassification rate using the model as a classifier that seeks to minimize the misclassification rate. Based on this, which model is best? Relative to the best, are the other two underfitting or over fitting? Report the test misclassification rate for the model you picked as the best.

Solution:

```r
library(boot)



wisc_train = read_csv('wisc-train.csv')
wisc_test = read_csv('wisc-test.csv')

# Check is response variable is factor and convert to factor is necessary
is.factor(wisc_train$class)
```

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

```r
wisc_train$class = as.factor(wisc_train$class)
wisc_test$class = as.factor(wisc_test$class)



# 3 models to fit
mod1 = glm(class ~ radius + smoothness + texture, data = wisc_train, family = binomial)
mod2 = glm(class ~ ., data = wisc_train, family = binomial)

# AIC backward selection
mod_aic = glm(class ~ .^2, data = wisc_train, family = binomial)
mod3 = step(mod_aic, direction = 'backward', criteria = 'AIC', trace = FALSE)

# Cross validation for each model
mod1_cv = cv.glm(wisc_train, mod1, K = 5)$delta[1]
mod2_cv = cv.glm(wisc_train, mod2, K = 5)$delta[1]
mod3_cv = cv.glm(wisc_train, mod3, K = 5)$delta[1]

# Table for output
cv_table = tibble('Model' = c('Model 1', 'Model 2', 'Model 3'),
                  'Cross-Validated Misclassification Rate' = c(mod1_cv, mod2_cv, mod3_cv))


cv_table %>%
  kbl(caption = 'Model Training Performance') %>%
  kable_styling()
```

Model 1 is the best performing model based on the cross validated misclassification rate. The other two models are overfitting the training data. I would consider Model 1 (misclassification rate 0.0805) to be the best model since it represents the best combination of performance (minimizing misclassification) and a limited set of predictors that helps to make the model more explainable.

(b) In this situation, simply minimizing misclassifications might be a bad goal since false positives and false negatives carry very different consequences. Consider the `M` class as the "positive" label. Consider each of the probabilities stored in `cutoffs` in the creation of a classifier using the **additive** model fit in **(a)**.

Table 2: Model Training Performance

Model	Cross-Validated Misclassification Rate
Model 1	0.0805
Model 2	0.1041
Model 3	0.0909

```r
cutoffs = seq(0.01, 0.99, by = 0.01)
```

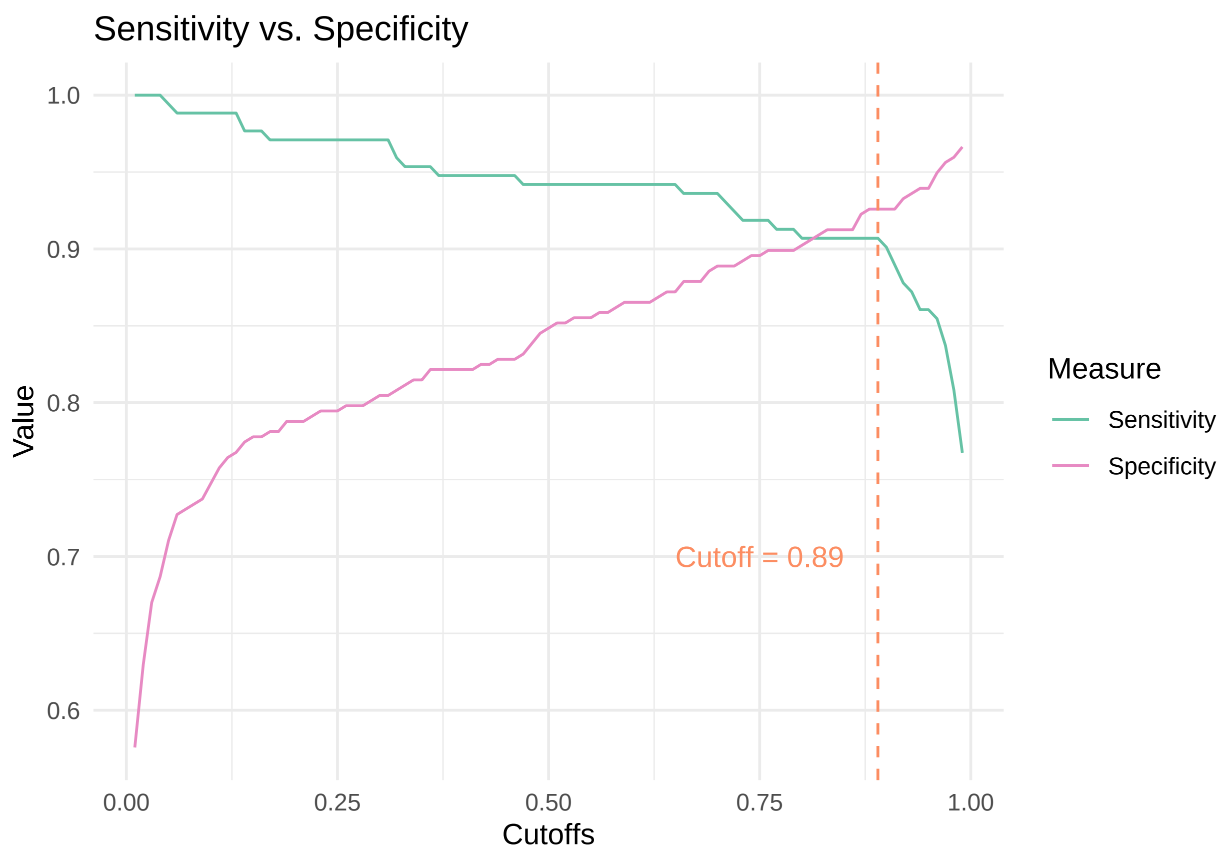
That is, consider each of the values stored in `cutoffs` as c. Obtain the sensitivity and specificity in the test set for each of these classifiers. Using a single graphic, plot both sensitivity and specificity as a function of the cutoff used to create the classifier. Based on this plot, which cutoff would you use? (0 and 1 have not been considered for coding simplicity. If you like, you can instead consider these two values.)

$$\hat{C}(\mathbf{x}) = \begin{cases} 1 & \hat{p}(\mathbf{x}) > c \\ 0 & \hat{p}(\mathbf{x}) \le c \end{cases}$$

Solution:

```r
# Predictions for test data and a vector of the actual value
wisc_pred = predict(mod2, newdata = wisc_test, type = 'response')
wisc_actual = wisc_test$class

# Dataframe to hold results
results_pred = tibble('Cutoffs' = rep(0,99),
                      'Sensitivity' = rep(0, 99),
                      'Specificity' = rep(0,99))


for(i in 1:length(cutoffs)) {
  # Set cutoff variable
  tmp_cutoff = cutoffs[i]

  # prediction at a specific cutoff
  tmp_pred = if_else(wisc_pred > tmp_cutoff,  'M', 'B')

  # Confusion Matrix
  tmp_conf_mat = table(predicted = tmp_pred, actual = wisc_actual)

  # Sensitivity
  tmp_sens = tmp_conf_mat[2,2] / sum(tmp_conf_mat[,2])

  #Specificity
   spec = tmp_conf_mat[1,1] / sum(tmp_conf_mat[,1])


  # Add results to table
  results_pred$Cutoffs[i] = cutoffs[i]
  results_pred$Sensitivity[i] = tmp_sens
  results_pred$Specificity[i] = spec
}
```

```r
results_pred = results_pred %>%
  pivot_longer(cols = c(Sensitivity, Specificity), names_to = 'Measure', values_to = 'Value')

p_results = ggplot(results_pred, aes(x = Cutoffs, Value, color = Measure)) +
  geom_line() +
  scale_color_manual(values = c('#66c2a5', '#e78ac3')) +
  geom_vline(xintercept = 0.89, linetype = 'dashed', color = '#fc8d62') +
  annotate('text', x = 0.75, y = 0.70, label = 'Cutoff = 0.89', color = '#fc8d62') +
  theme_minimal() +
  labs(title = 'Sensitivity vs. Specificity')


p_results
```



I would use a cutoff of 0.89 for classifying since that is the cutoff that maximizes both Sensitivity and Specificity.