Naive Bayes Classification Assignment

Assignment Description

This assignment is designed to test your knowledge of Naive Bayes Classification. It closely mirrors our naive_bayes_penguins.qmd from lectures 10/1 and 10/3. We reflect back on the true vs fake news dataset from the beginning of the semester and apply the new skills in our bayesian toolbox.

This assignment is worth 16 points and is due by 10:00am on October 15th. Each section has a number of points noted. To turn in this assignment, render this qmd and save it as a pdf, it should look beautiful. If you do not want warning messages and other content in the rendered pdf, you can use `message = FALSE, warning = FALSE` at the top of each code chunk as it appears in the libraries code chunk below.

Load Libraries

```
library(bayesrules)
library(tidyverse)
library(e1071)
library(janitor)
```

Read in data

```
data(fake_news)
```

Challenge

Exercise 14.7 Fake news: three predictors

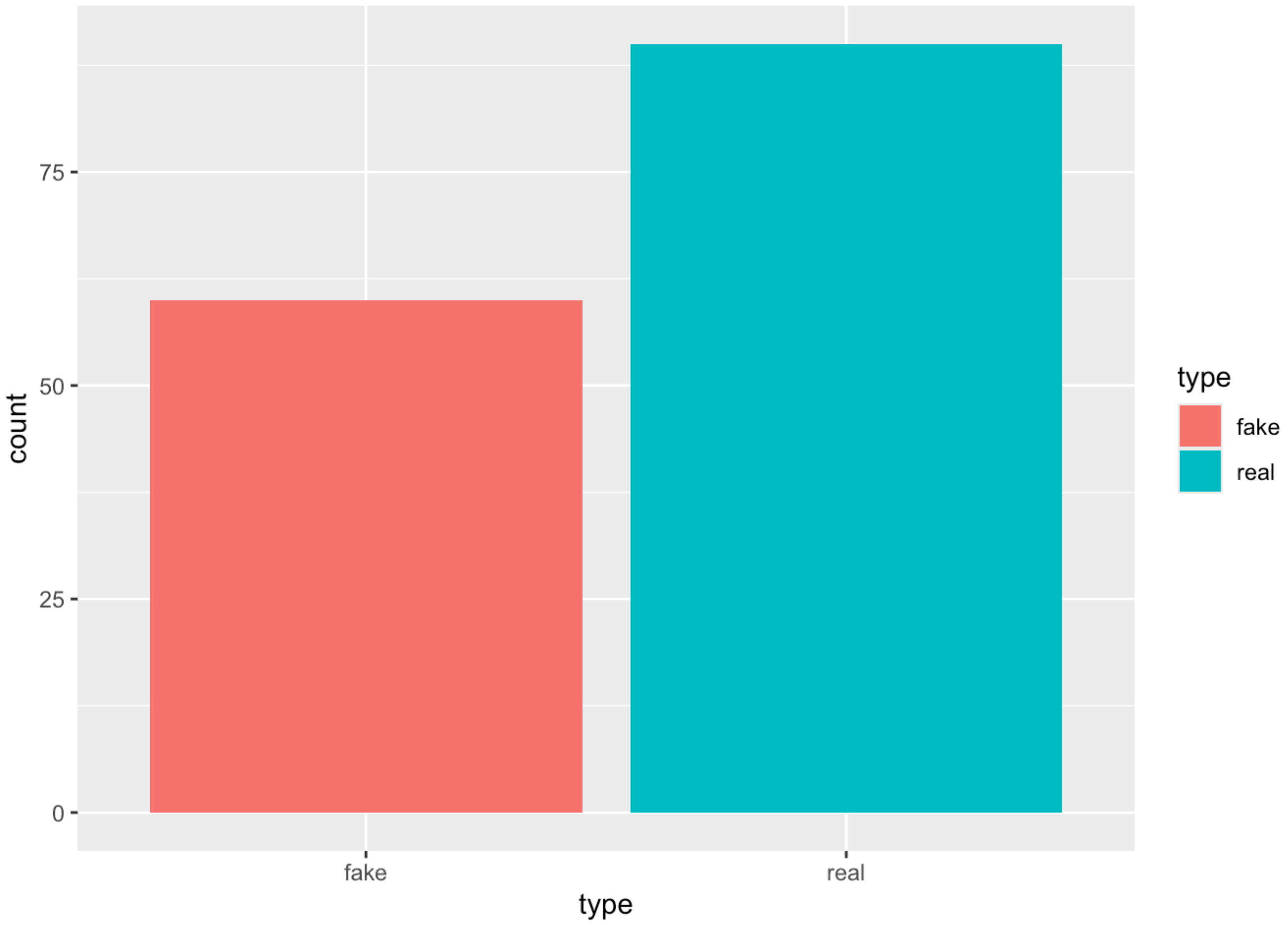
Suppose a ***new news article*** is posted online – it has a 15-word title, 6% of its words have negative associations, and its title *doesn't* have an exclamation point. We want to know if it is fake or real

Visualization (Exploratory Data Analysis) - 2 points

Below, insert a code chunk(s) and use `ggplot` to visualize the features of the data we are interested in. This can be one or multiple visualizations

- Type (fake vs real)

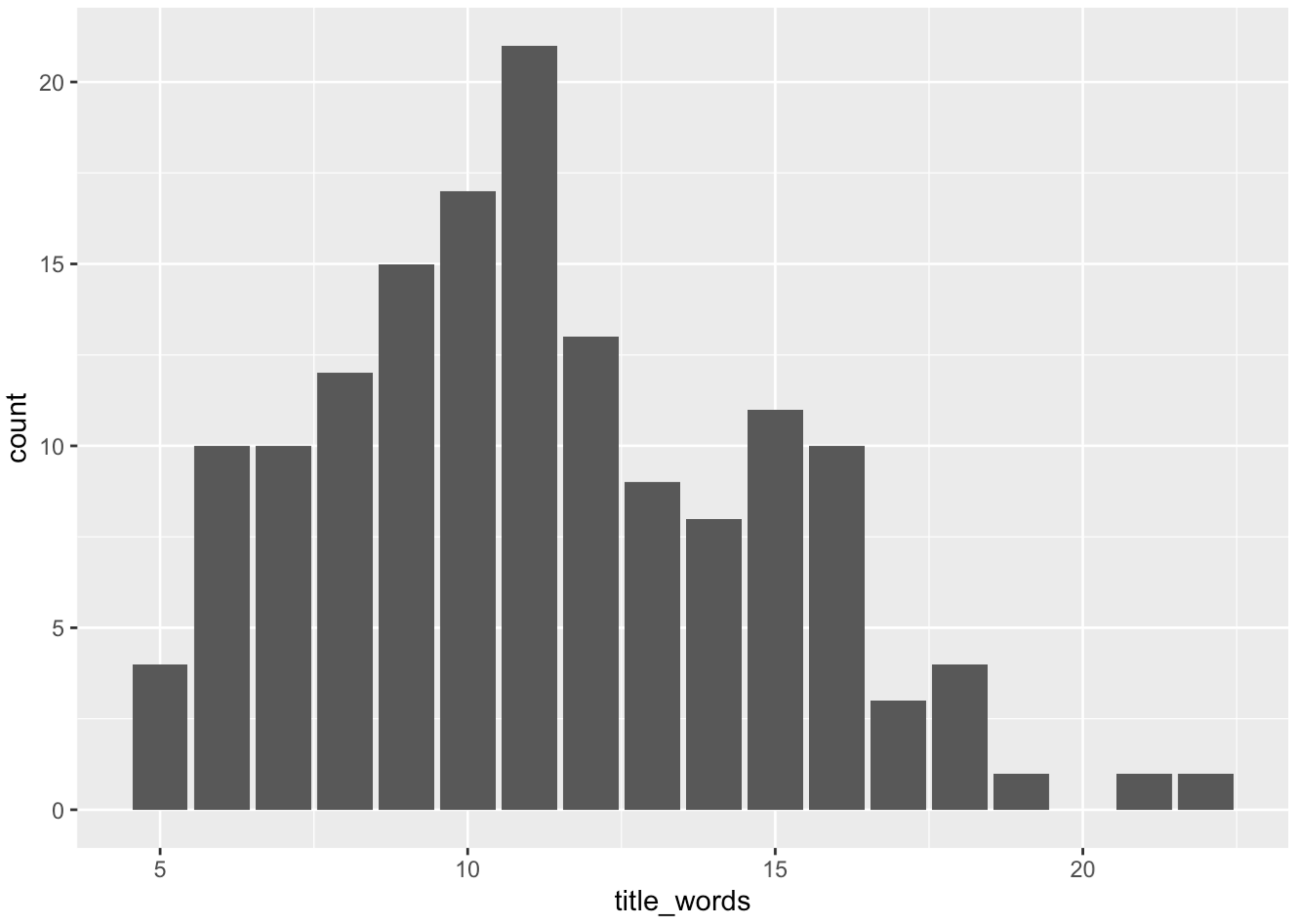
  ```
  ggplot(data = fake_news, aes(x = type, fill = type)) + geom_bar()
  ```



- Number of words in the title (numeric value)

```r
ggplot(data = fake_news, aes(x = title_words, fill = title_words)) + geom_bar()
```

```
Warning: The following aesthetics were dropped during statistical transformation:
fill.
ℹ This can happen when ggplot fails to infer the correct grouping structure in
  the data.
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```

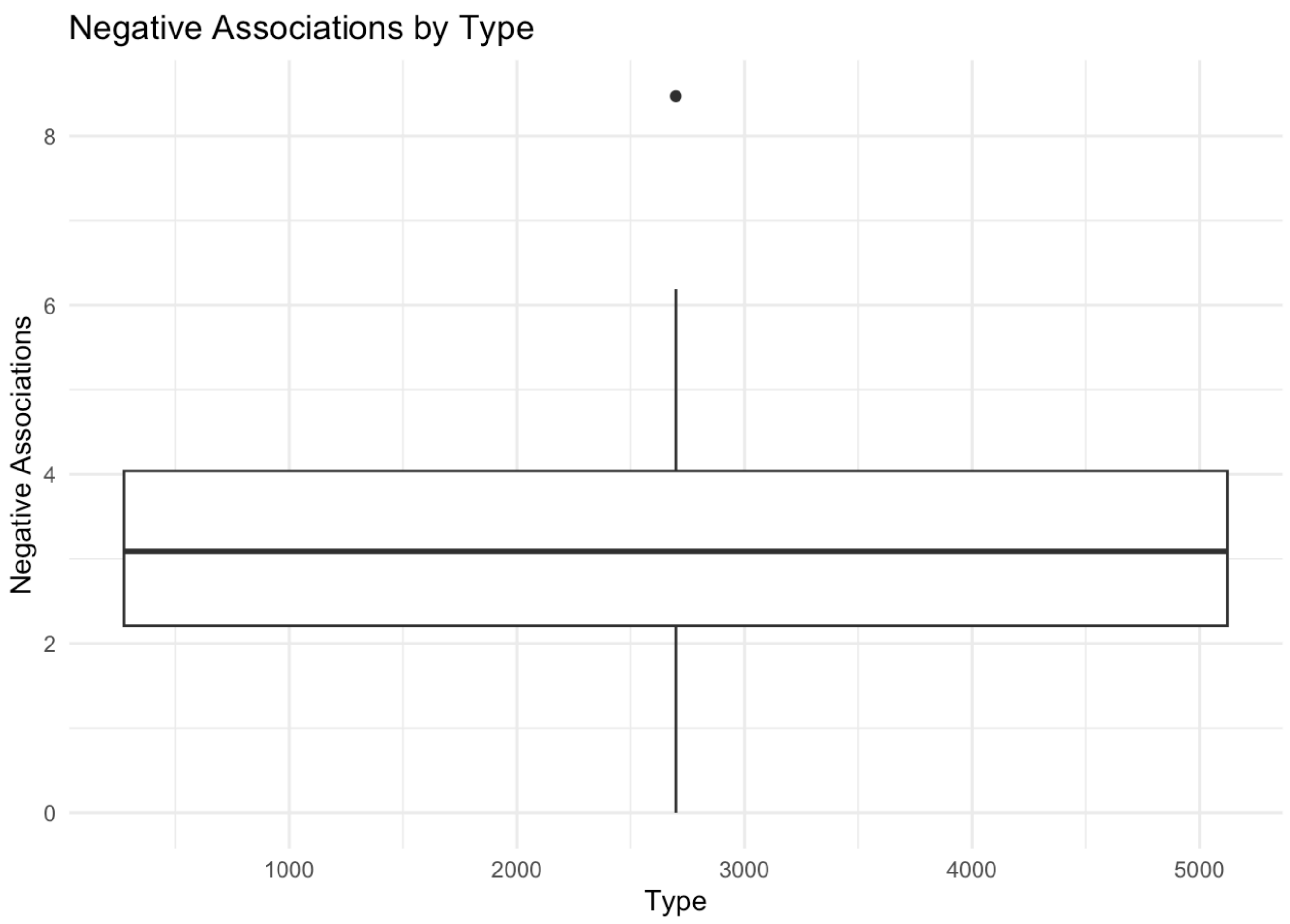


- Negative associations (numeric value)

```
ggplot(fake_news, aes(x = text_words, y = negative, fill = text_words)) +
  geom_boxplot() + theme_minimal() +
  labs(title = "Negative Associations by Type", x = "Type", y = "Negative Association
```

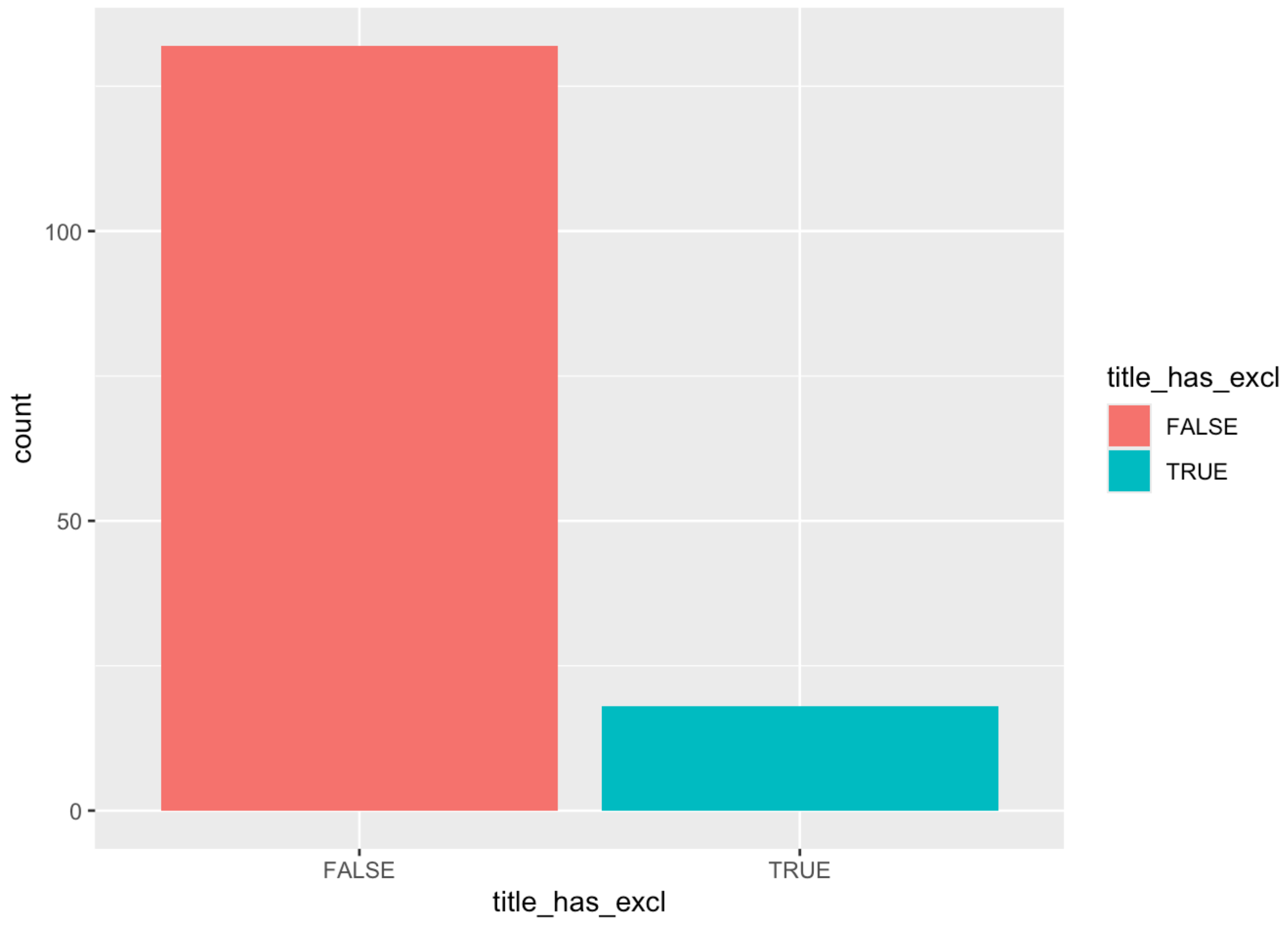
```
Warning: Continuous x aesthetic
ℹ did you forget `aes(group = ...)`?

Warning: The following aesthetics were dropped during statistical transformation:
fill.
ℹ This can happen when ggplot fails to infer the correct grouping structure in
  the data.
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```



- Exclamation point in the title (true vs false)

```r
ggplot(data = fake_news, aes(x = title_has_excl, fill = title_has_excl)) + geom_bar()
```



Interpretation of Visualization - 2 points

Below, write a few sentences explaining whether or not this ***new news article*** is true or fake solely using your visualization above

- The news article has a higher chance of being true from the lack of exclamation points. There are <25 exclamation marks compared to >125. More exclamation marks may mean a false signal of importance.

Perform Naive Bayes Classification - 3 points

Based on these three features (15-word title, 6% of its words have negative associations, and its title *doesn't* have an exclamation point), utilize naive Bayes classification to calculate the posterior probability that the article is real. Do so using `naiveBayes()` with `predict()`.

Below, insert the code chunks and highlight your answer

```
naive_model_hints <- naiveBayes(
  type ~ title_words + negative + title_has_excl, data = fake_news)
```

```
naive_model_hints
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
fake real 
 0.4  0.6 

Conditional probabilities:
      title_words
Y          [,1]     [,2]
  fake 12.31667 3.743884
  real 10.42222 3.204554

      negative
Y          [,1]     [,2]
  fake 3.606333 1.466429
  real 2.806556 1.190917

      title_has_excl
Y           FALSE       TRUE
  fake 0.73333333 0.26666667
  real 0.97777778 0.02222222
```

```r
test_data <- data.frame(
  title_words = 15,
  negative = 0.06,
  title_has_excl = 'FALSE'
)
```

```r
posterior_probability <- predict(naive_model_hints, test_data, type = "raw")

posterior_probability
```

```
          fake      real
[1,] 0.3639842 0.6360158
```

```r
posterior_probability_real <- 0.6360158
```

`{naiveBayes(type ~ title_words + negative + title_has_excl, data = .)}`

Break Down the Model - 5 points

Similar to the penguins example, we are going to break down the model we created above. To do this we need to find:

- Probability(15 - word title| article is real) using `dnorm()`

```
real_articles <- fake_news %>%
      filter(type == "real" & !is.na(title_words))

mu_real_title <- mean(real_articles$title_words)
sd_real_title <- sd(real_articles$title_words)

mu_real_title
```

```
[1] 10.42222
```

```
sd_real_title
```

```
[1] 3.204554
```

```
prob_15_word_title_given_real <- dnorm(15, mean = mu_real_title, sd = sd_real_title)

prob_15_word_title_given_real
```

```
[1] 0.0448761
```

```
prob_15_word_title_given_real <- 0.0448761
```

- Probability(6% of words have negative associations | article is real) using `dnorm()`

```
real_articles <- fake_news %>%
      filter(type == "real" & !is.na(negative))


mu_real_negative <- mean(real_articles$negative)
sd_real_negative <- sd(real_articles$negative)


mu_real_negative
```

```
[1] 2.806556
```

```
sd_real_negative
```

```
[1] 1.190917
```

```
prob_6percent_negative_given_real <- dnorm(0.06, mean = mu_real_negative, sd = sd_rea

prob_6percent_negative_given_real
```

```
[1] 0.02344583
```

```
prob_6percent_negative_given_real <- 0.02344583
```

- Probability(no exclamation point in title | article is real)

```
prob_no_excl_given_real <- mean(real_articles$title_has_excl == 0)

prob_no_excl_given_real
```

```
[1] 0.9777778
```

```
prob_no_excl_given_real <- 0.9777778
```

  - Multiply these probabilities and save as the object `probs_real`

```
probs_real = prob_15_word_title_given_real * prob_6percent_negative_given_real * prob_
```

```
probs_real <- 0.001028776
```

- Probability(15 - word title| article is fake) using `dnorm()`

```
fake_articles <- fake_news %>%
filter(type == "fake" & !is.na(title_words))

mu_fake_title <- mean(fake_articles$title_words)
sd_fake_title <- sd(fake_articles$title_words)

mu_fake_title
```

```
[1] 12.31667
```

```
sd_fake_title
```

```
[1] 3.743884
```

```
prob_15_word_title_given_fake <- dnorm(15, mean = mu_fake_title, sd = sd_fake_title)

prob_15_word_title_given_fake
```

```
[1] 0.08242149
```

```
prob_15_word_title_given_fake <- dnorm(15, mean = mu_fake_title, sd = sd_fake_title)

prob_15_word_title_given_fake
```

```
[1] 0.08242149
```

```
prob_15_word_title_given_fake <- 0.08242149
```

- Probability(6% of words have negative associations | article is fake) using `dnorm()`

```
fake_articles <- fake_news %>%
filter(type == "fake" & !is.na(negative))

mu_fake_negative <- mean(fake_articles$negative)
sd_fake_negative <- sd(fake_articles$negative)

mu_fake_negative
```

```
[1] 3.606333
```

```
sd_fake_negative
```

```
[1] 1.466429
```

```
prob_6percent_negative_given_fake <- dnorm(0.06, mean = mu_fake_negative, sd = sd_fak

prob_6percent_negative_given_fake
```

```
[1] 0.01461117
```

```
prob_6percent_negative_given_fake <- 0.01461117
```

- Probability(no exclamation point in title | article is fake)

```
prob_no_excl_given_fake <- mean(fake_articles$title_has_excl == 0)

prob_no_excl_given_fake
```

```
[1] 0.7333333
```

```
prob_no_excl_given_fake <- 0.7333333
```

  - Multiply these probabilities and save as the object **`probs_fake`**

```
probs_fake = prob_no_excl_given_fake * prob_6percent_negative_given_fake * prob_15_word_t

probs_fake
```

```
[1] 0.0008831345
```

```
probs_fake <- 0.0008831345
```

Lastly divide your **`probs_real`** by the sum of **`probs_real`** and **`probs_fake`** to see if you can reproduce the output from `naiveBayes()` above

```
output = probs_real + probs_fake/probs_real

output
```

```
[1] 0.859461
```

Confusion Matrix - 2 points

Calculate a confusion matrix by first mutating a column to fake_news called `predicted_type` . Then, use `tabyl()` to create the matrix

```
fake_news <- fake_news %>%
  mutate(predicted_type = ifelse(posterior_probability_real >
                                   0.5, "Real", "Fake"))
```

```
confusion_matrix <- fake_news %>%
  tabyl(type, predicted_type) %>%
  adorn_totals("row")

confusion_matrix
```

```
  type Real
  fake   60
  real   90
 Total  150
```

How can our model be improved? - 2 points

Think about the results of the confusion matrix, is the model performing well? Try creating a new model that uses all of the features in the fake_news dataset to make a prediction on type (fake vs true). Then, create a new confusion matrix to see if the model improves.

```
model2 <- naiveBayes(type ~ ., data = fake_news)

new_predict <- predict(model2, newdata = fake_news)

matrix2 <- table(fake_news$type, new_predict)

matrix2_tidy <- as.data.frame.matrix(matrix2) %>%
  adorn_totals("row")

matrix2_tidy
```

```
 fake real
   58    2
    2   88
Total   90
```