Regression Inference, Bootstrapping, and Prediction

Python Code Sample with the Snowy Plover

Brianna Hayes

2023-12-21

The Data

The Snowy Plover is a small bird native to the California coast. The small size of the bird makes it vulnerable to many predators, including people that may not seem them when walking on the beach. Currently, the Snowy Plover is considered endangered in the United States.

This data was collected by the **Point Reyes National Seashore**. Hopefully, larger newly-hatched Snowy Plover chicks are more likely to survive predators. The objective from constructing this data set was to see if the size and weight of an egg could predict the weight of the hatched chick. So, this data set observes the weight of a chick, the weight of it's egg from which it hatched, and the size and breadth of the egg.

Bird Weight - weight of the chick, measured in grams

Egg Weight - weight of the egg, measured in grams

Egg Length - length of the egg, measured in millimeters

Egg Breadth - widest diameter of the egg, measured in millimeters

```
birds = Table.read_table('snowy_plover.csv')
birds
```

```
## Egg Length | Egg Breadth | Egg Weight | Bird Weight
## 28.8       | 21.84       | 7.4        | 5.2
## 29.04      | 22.45       | 7.7        | 5.4
## 29.36      | 22.48       | 7.9        | 5.6
## 30.1       | 21.71       | 7.5        | 5.3
## 30.17      | 22.75       | 8.3        | 5.9
## 30.34      | 22.84       | 8.5        | 5.8
## 30.36      | 22.5        | 8.2        | 5.8
## 30.46      | 22.72       | 8.3        | 6
## 30.54      | 23.31       | 9          | 6.1
## 30.62      | 22.94       | 8.5        | 6.2
## ... (34 rows omitted)
```

Defining Functions for Regression and Boostrapping

Converts an array to standard units

```
def su(arr):
  return(arr-np.average(arr))/np.std(arr)
```

Correlation, r

```
def corr(tbl, x_col, y_col):
  x_su = su(tbl.column(x_col))
  y_su = su(tbl.column(y_col))
  return np.average(x_su*y_su)
```

Slope and Intercept, in a two-item array

```
def fit(tbl, x_col, y_col):
  r = corr(tbl, x_col, y_col)
  slope_xy = r * np.std(tbl.column(y_col)) / np.std(tbl.column(x_col))
  intercept_xy = np.average(tbl.column(y_col) - slope_xy * np.average(tbl.column(x_col)))
  slope_int = make_array(slope_xy, intercept_xy)
  return slope_int
```

Prediction of Y, given X

```
def fitted_y(tbl, x_col, y_col, given_x):
  slope = fit(tbl, x_col, y_col).item(0)
  intercept = fit(tbl, x_col, y_col).item(1)
  return slope*given_x+intercept
```

Slope and intercept of a resampled x and y, in a two-item array

```
def resampled_fit(tbl, x_col, y_col):
  resample = tbl.sample(tbl.num_rows, with_replacement=True)
  return fit(resample, x_col, y_col)
```

Generate a table of resampled slopes and intercepts

```
def bootstrap_fits(tbl, x_col, y_col, num_reps):
  resampled_slopes = make_array()
  resampled_ints = make_array()
  for i in np.arange(num_reps):
```

```
    resampled_coefs = resampled_fit(tbl, x_col, y_col)
    resampled_slope = resampled_coefs.item(0)
    resampled_int = resampled_coefs.item(1)
    resampled_slopes = np.append(resampled_slopes, resampled_slope)
    resampled_ints = np.append(resampled_ints, resampled_int)
  tbl_fits = Table().with_columns(
    'Slope', resampled_slopes,
    'Intercept', resampled_ints)
  return tbl_fits
```

Regression Inference for the Relationship between a Snowy Plover's Egg Weight and Bird Weight

Slope and Intercept

```
slope, intercept = fit(birds, 'Egg Weight', 'Bird Weight')
slope
```

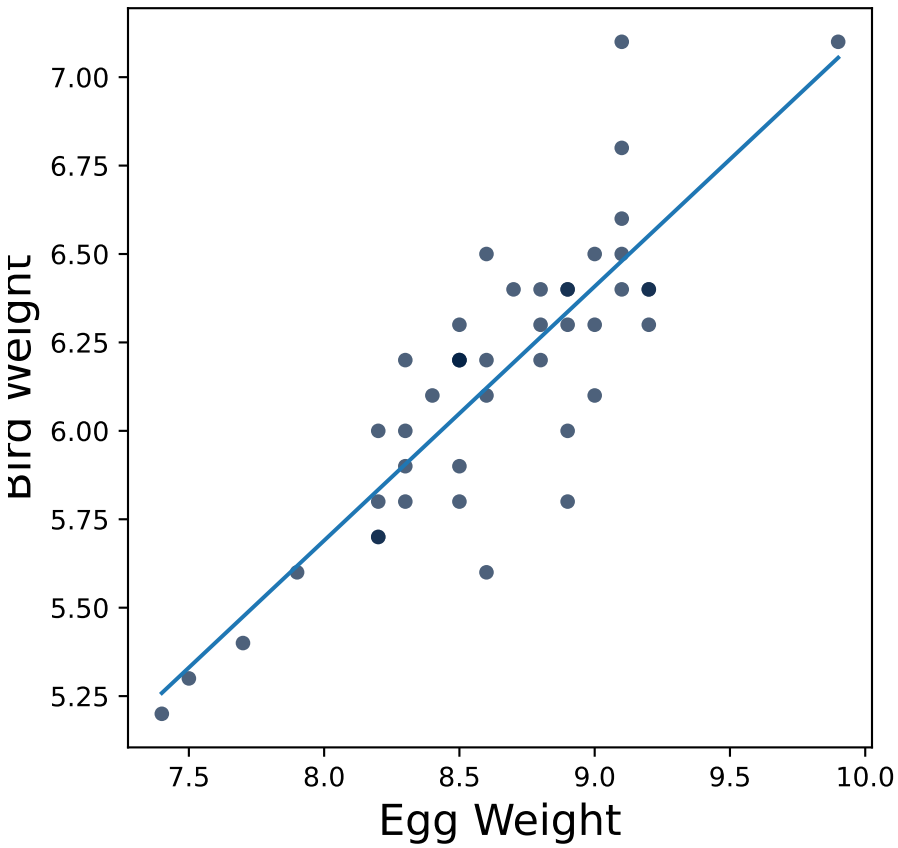
```
## 0.7185153448936793
```

```
intercept
```

```
## -0.058272261934289069
```

Plotting the line of best fit on a scatter plot

```
birds.scatter('Egg Weight', 'Bird Weight')
plt.plot([min(birds.column("Egg Weight")),
        max(birds.column("Egg Weight"))],
         [slope*min(birds.column("Egg Weight"))+intercept,
         slope*max(birds.column("Egg Weight"))+intercept])
```



One of the resident researchers has been observing an egg with a weight of 9 grams, making the predicted weight...

```
fitted_y(birds, 'Egg Weight', 'Bird Weight', 9)
```

```
## 6.408365842108824
```

However, this prediction has been made from a sample data set that contains only 44 observations. How might the predictions change based with a different sample of birds?

Bootstrap Prediction for an Egg Weighing 9 Grams

Generating a table of slopes and intercepts for 1000 bootstrap samples

```
bird_fits = bootstrap_fits(birds, 'Egg Weight', 'Bird Weight', 1000)
bird_fits
```

```
## Slope    | Intercept
## 0.701137 | 0.0904709
## 0.621581 | 0.789058
## 0.651155 | 0.500598
## 0.703858 | 0.0394975
## 0.76352  | -0.416132
## 0.602728 | 0.914112
## 0.780616 | -0.630545
## 0.759598 | -0.44351
## 0.78297  | -0.594541
## 0.687963 | 0.197558
## ... (990 rows omitted)
```

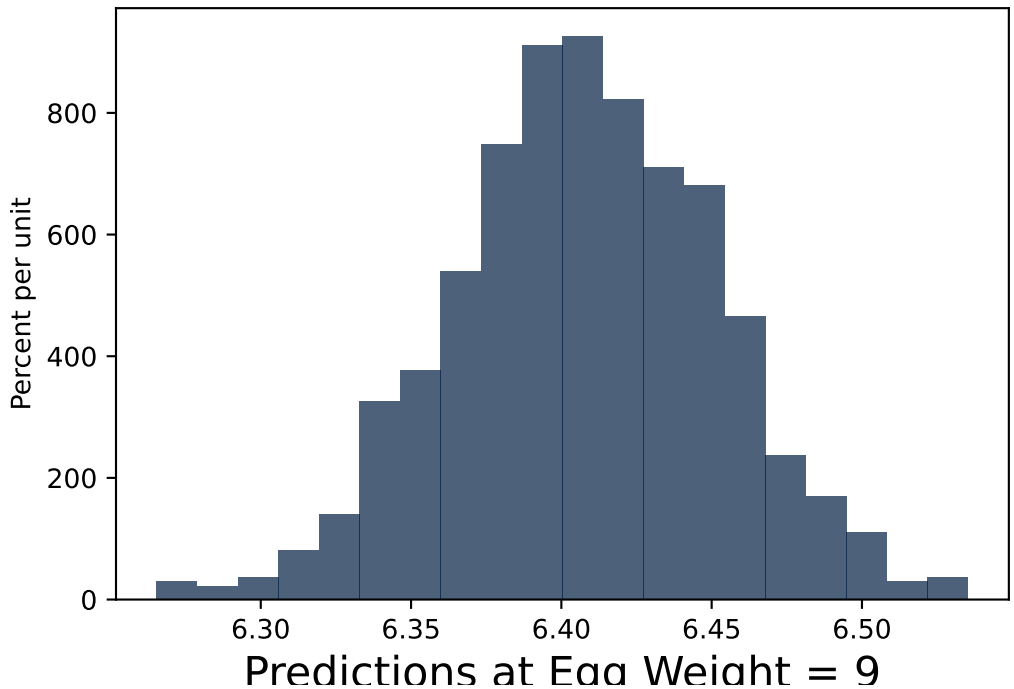
Predicted bird weight for a 9 gram egg, for each of the bootsrap regressions

```
egg_nine_predictions = make_array()
for i in np.arange(bird_fits.num_rows):
  egg_nine_predictions = np.append(egg_nine_predictions,
                                   bird_fits.column(0) * 9 + bird_fits.column(1))
egg_nine_predictions
```

```
## array([ 6.40069998,  6.38328267,  6.36099071, ...,  6.35797872,
##         6.383738  ,  6.43910779])
```

Histogram of predicted bird weights

```
prediction_tbl = Table().with_column(
  'Predictions at Egg Weight = 9', egg_nine_predictions
)
prediction_tbl.hist('Predictions at Egg Weight = 9', bins=20)
plt.show()
```



Constructing a 95% confidence interval

```
lower_bound = percentile(2.5, egg_nine_predictions)
upper_bound = percentile(97.5, egg_nine_predictions)

lower_bound, upper_bound
```

```
## (6.3201948283115961, 6.4938257303709186)
```