DS1EDP: Homework 07 – Solutions

1. Plot the Vote

Question 1:

```
def proportions_in_resamples():
    prop_c = make_array()
    for i in np.arange(5000):
        bootstrap = votes.sample()
        single_proportion =
               np.count_nonzero(bootstrap.column("vote") == "C") /
               bootstrap.num_rows
        prop_c = np.append(prop_c, single_proportion)
    return prop_c
```

Question 2:

```
c_lower_bound = percentile(2.5, sampled_proportions)
c_upper_bound = percentile(97.5, sampled_proportions)
```

Question 3:

```
bins = np.arange(-0.2,0.2,0.01)

def leads_in_resamples():
    leads = make_array()
    for i in np.arange(5000):
        bootstrap = votes.sample()
        c_proportion = np.count_nonzero(bootstrap.column("vote")
               == "C") / bootstrap.num_rows
        t_proportion = np.count_nonzero(bootstrap.column("vote")
               == "T") / bootstrap.num_rows
        leads = np.append(leads, c_proportion - t_proportion)
    return leads

sampled_leads = leads_in_resamples()
Table().with_column('Estimated Lead',
     sampled_leads).hist(bins=bins)
```

Question 4:

```
diff_lower_bound = percentile(2.5, sampled_leads)
diff_upper_bound = percentile(97.5, sampled_leads)
```

2. Interpreting Confidence Intervals

Question 1:

```
correct_option = 2
```

Question 2:

```
true_proportion_intervals = 9500
```

Question 3:

```
confidence_interval_80 = interval_2
confidence_interval_90 = interval_1
```

```
confidence_interval_99 = interval_3
```

Question 4:
```
candidates_tied = 2
```

Question 5:
```
cutoff_one_percent = 2
```

Question 6:
```
cutoff_ten_percent = 3
```

3. Triple Jump Distances vs. Vertical Jump Heights

Question 1:
```
jumps.scatter("triple", "vertical", fit_line=True)
linear_correlation = True
```

Question 2:
```
r_estimate = 0.5
```

Question 3:
```
def su(a):
    return (a - np.mean(a))/np.std(a)

def regression_parameters(t):
    r = np.mean(su(t.column(0)) * su(t.column(1)))
    slope = r * np.std(t.column(1)) / np.std(t.column(0))
    intercept = np.mean(t.column(1)) - slope *
          np.mean(t.column(0))
    return make_array(r, slope, intercept)

parameters = regression_parameters(jumps)
```

Question 4:
```
triple_record_vert_est = parameters.item(1) * 1829 +
     parameters.item(2)
```

Question 5:
```
estimation_accurate = False
```

4. Cryptocurrencies

Question 1:
```
def std_units(arr):
    return (arr - np.mean(arr)) / np.std(arr)

standard_btc = std_units(btc.column("open"))
standard_eth = std_units(eth.column("open"))

r = np.mean(standard_btc * standard_eth)
```

Question 2:

```
def eth_predictor(btc_price):
    parameters = regression_parameters(Table().with_columns("btc",
          btc.column("open"), "eth", eth.column("open")))
    slope = parameters.item(1)
    intercept = parameters.item(2)
    return slope * btc_price + intercept
```

Question 3:

```
prices_with_predictions = Table().with_columns(
    "btc", btc.column("open"),
    "eth", eth.column("open"),
    "eth_prediction", eth_predictor(btc.column("open")))
prices_with_predictions.scatter("btc")
```

Question 4:

```
regression_changes = [False, True, True]
```