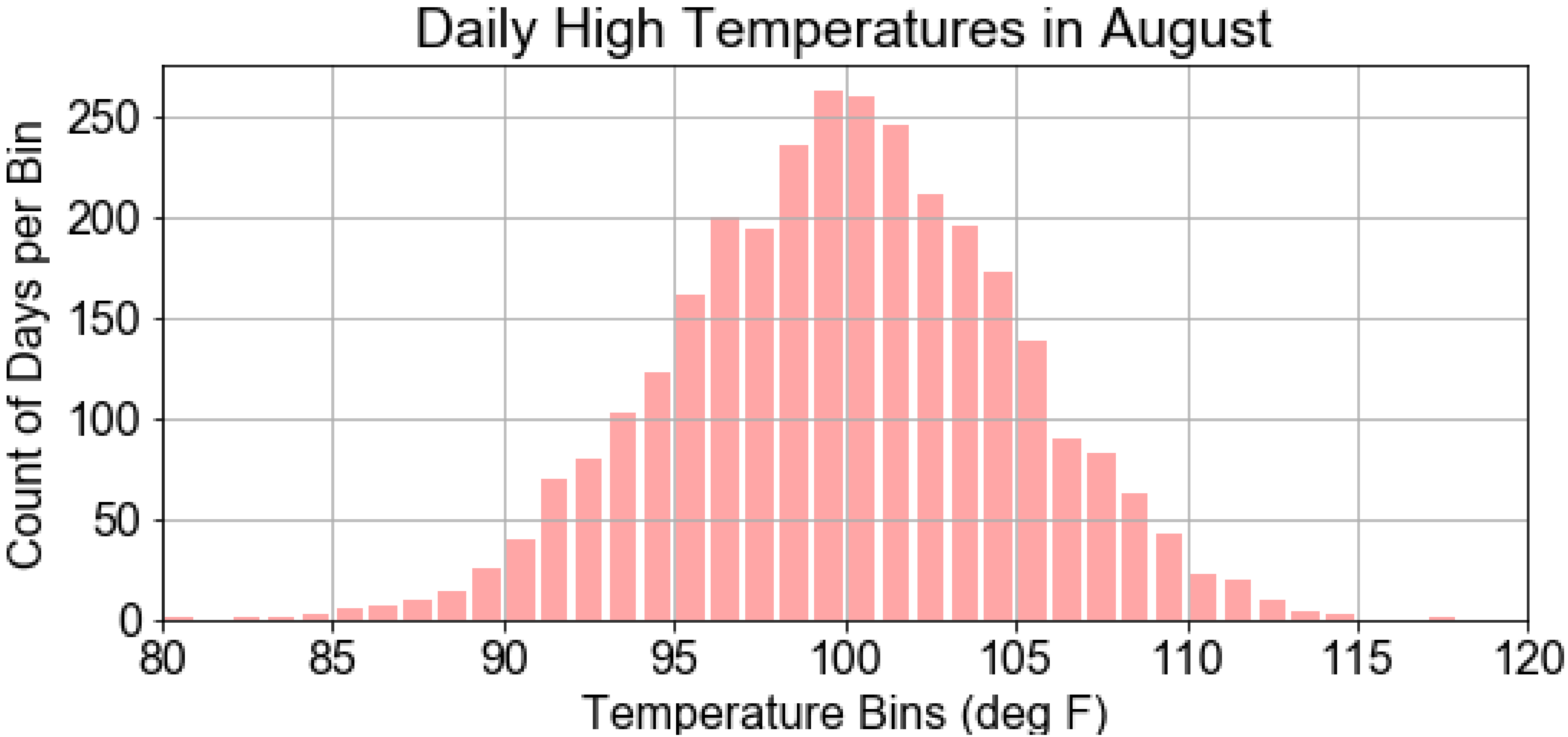
Inferential Statistics Concepts

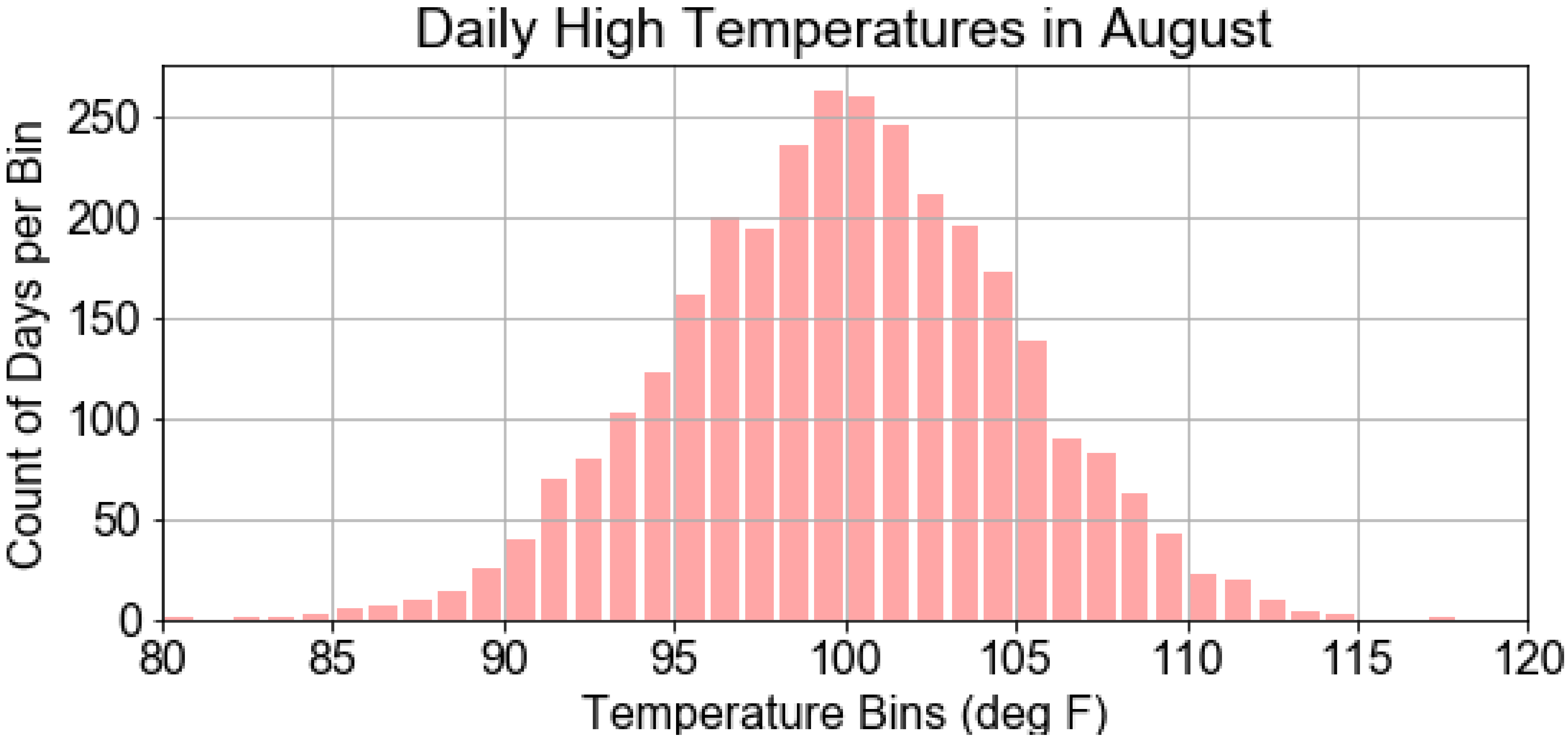
INTRODUCTION TO LINEAR MODELING IN PYTHON

Jason Vestuto
Data Scientist

Probability Distribution



Populations and Statistics



Sampling the Population

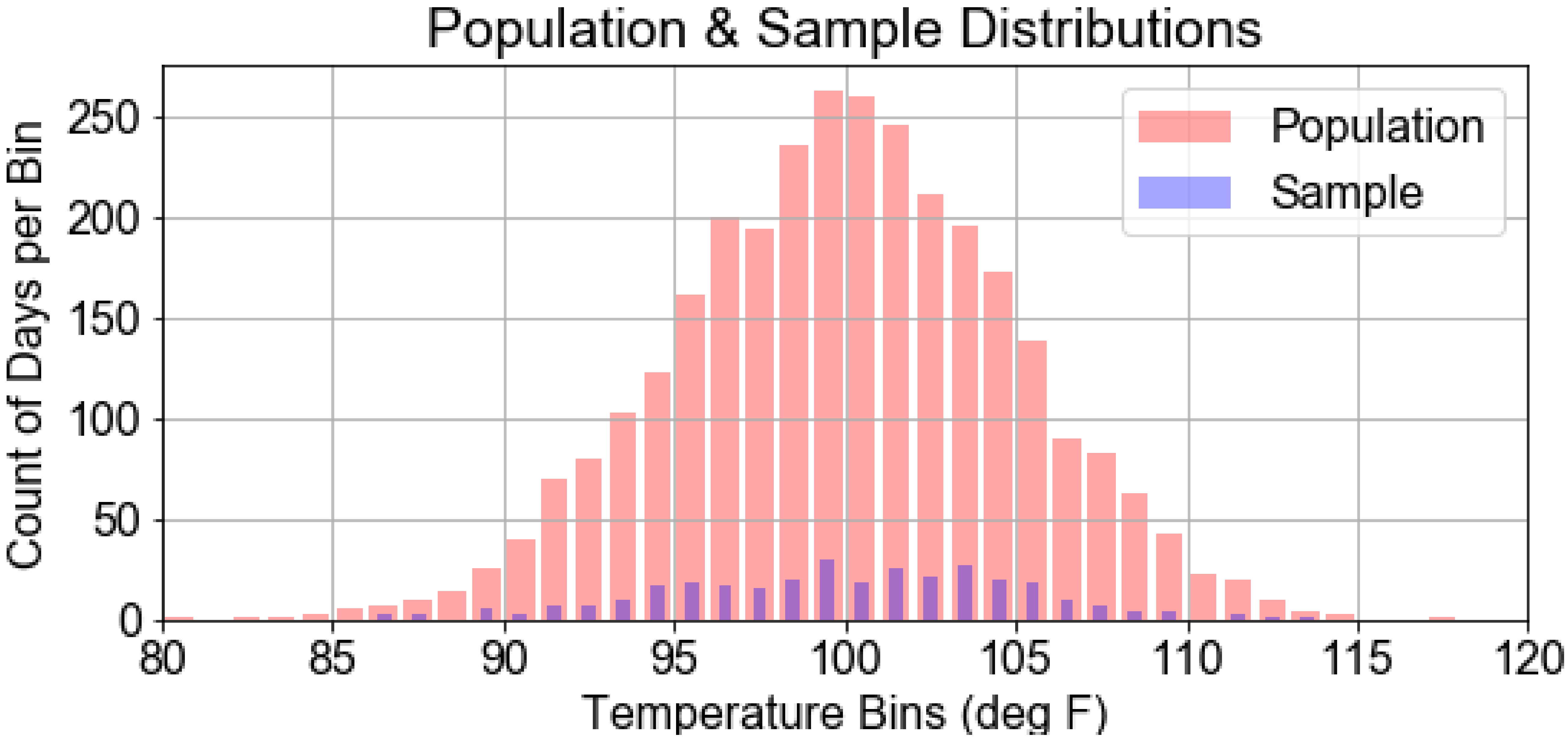
Population statistics vs Sample statistics

```
print( len(month_of_temps), month_of_temps.mean(), month_of_temps.std() )
print( len(decade_of_temps), decade_of_temps.mean(), decade_of_temps.std() )
```

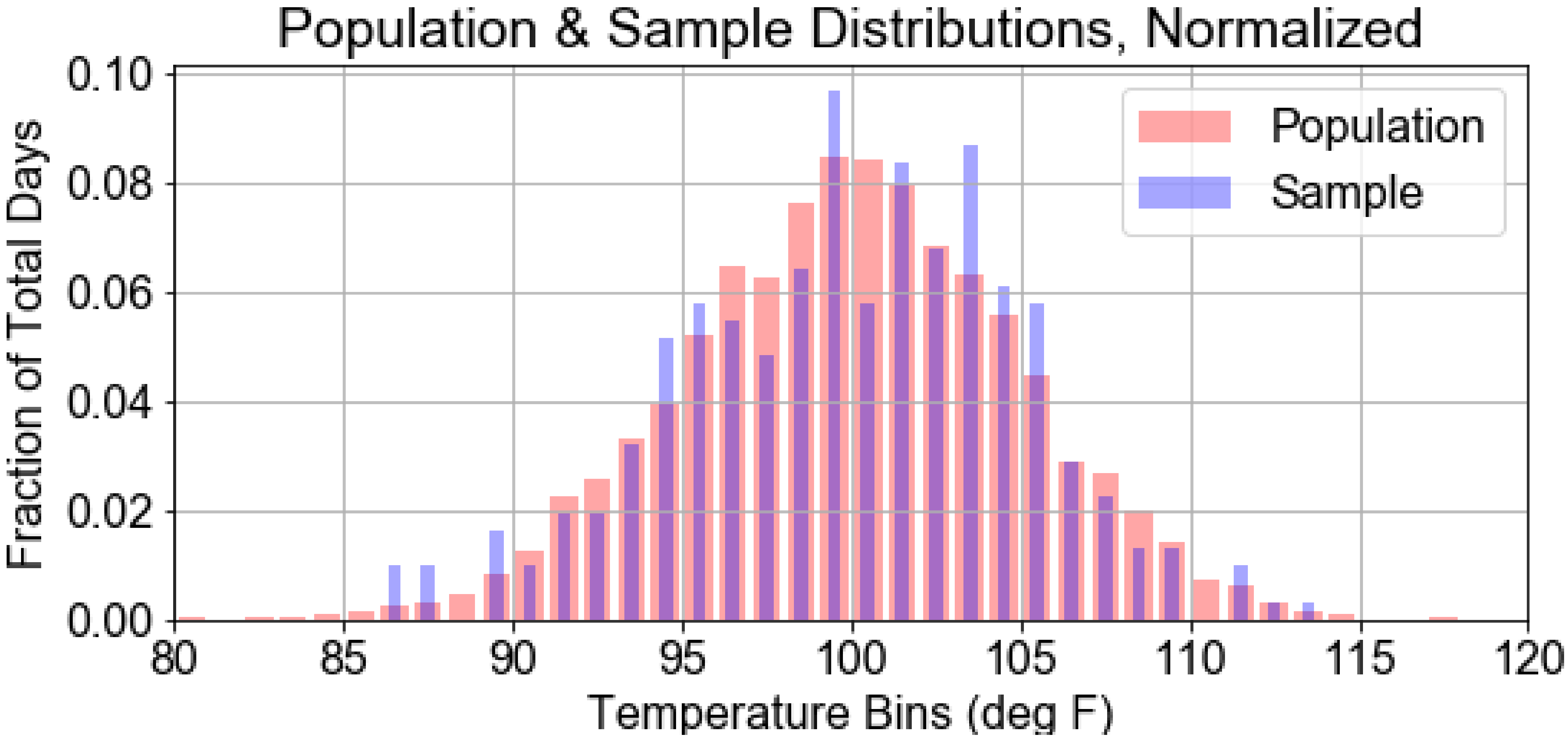
Draw a Random Sample from a Population

```
month_of_temps = np.random.choice(decade_of_temps, size=31)
```

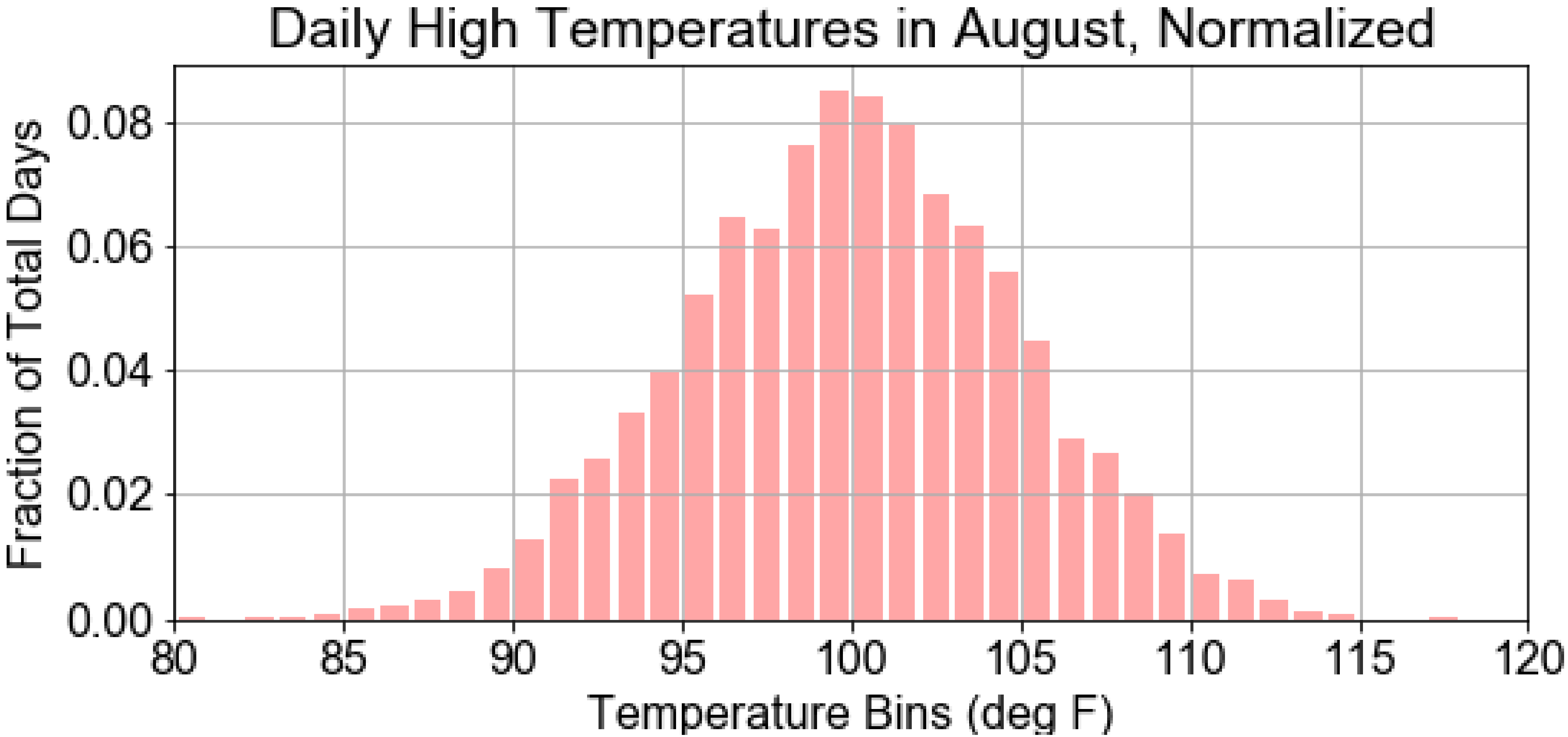
Visualizing Distributions



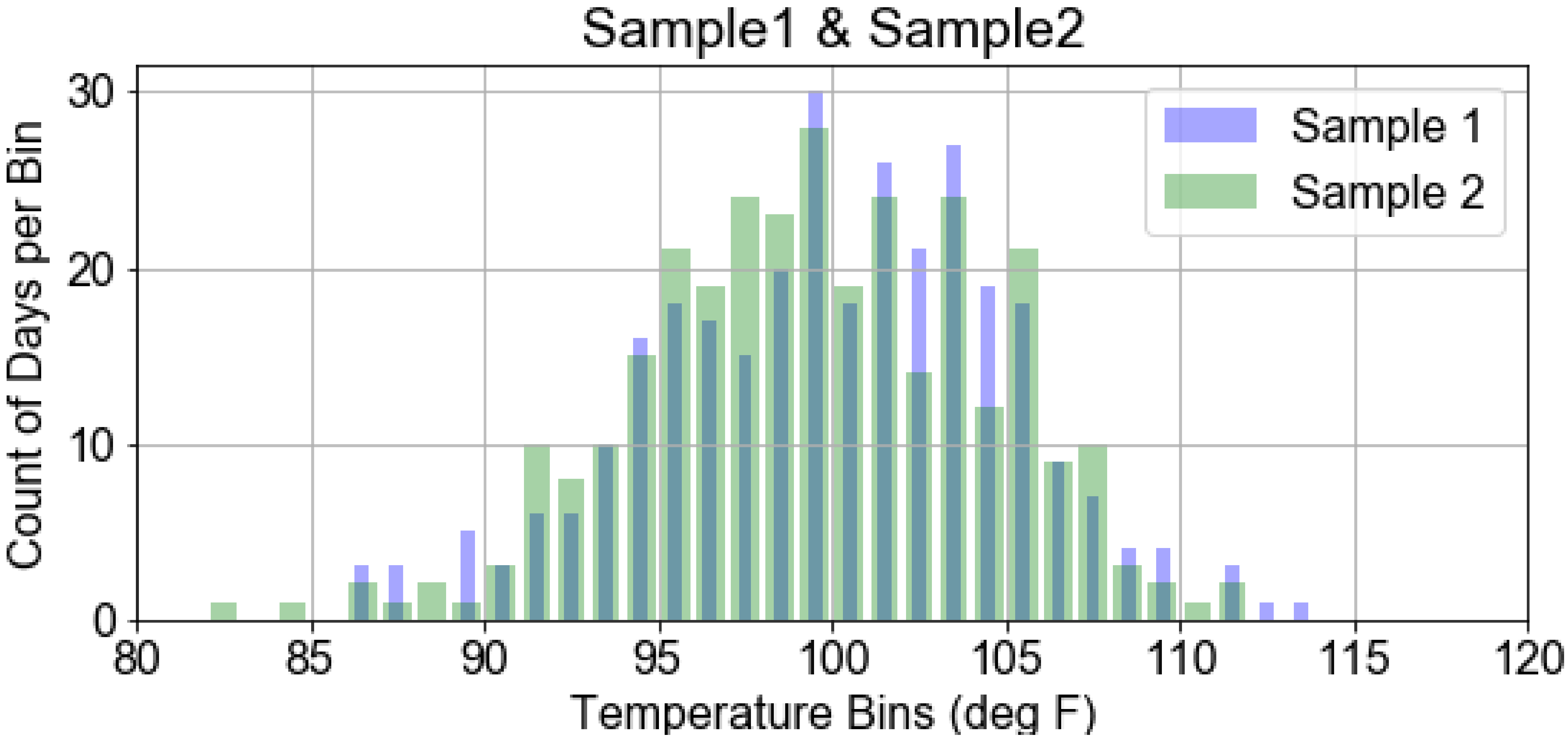
Visualizing Distributions



Probability and Inference



Visualizing Distributions



Resampling

```
# Resampling as Iteration
num_samples = 20
for ns in range(num_samples):
    sample = np.random.choice(population, num_pts)
    distribution_of_means[ns] = sample.mean()
```

```
# Sample Distribution Statistics
mean_of_means = np.mean(distribution_of_means)
stdev_of_means = np.std(distribution_of_means)
```

Let's practice!

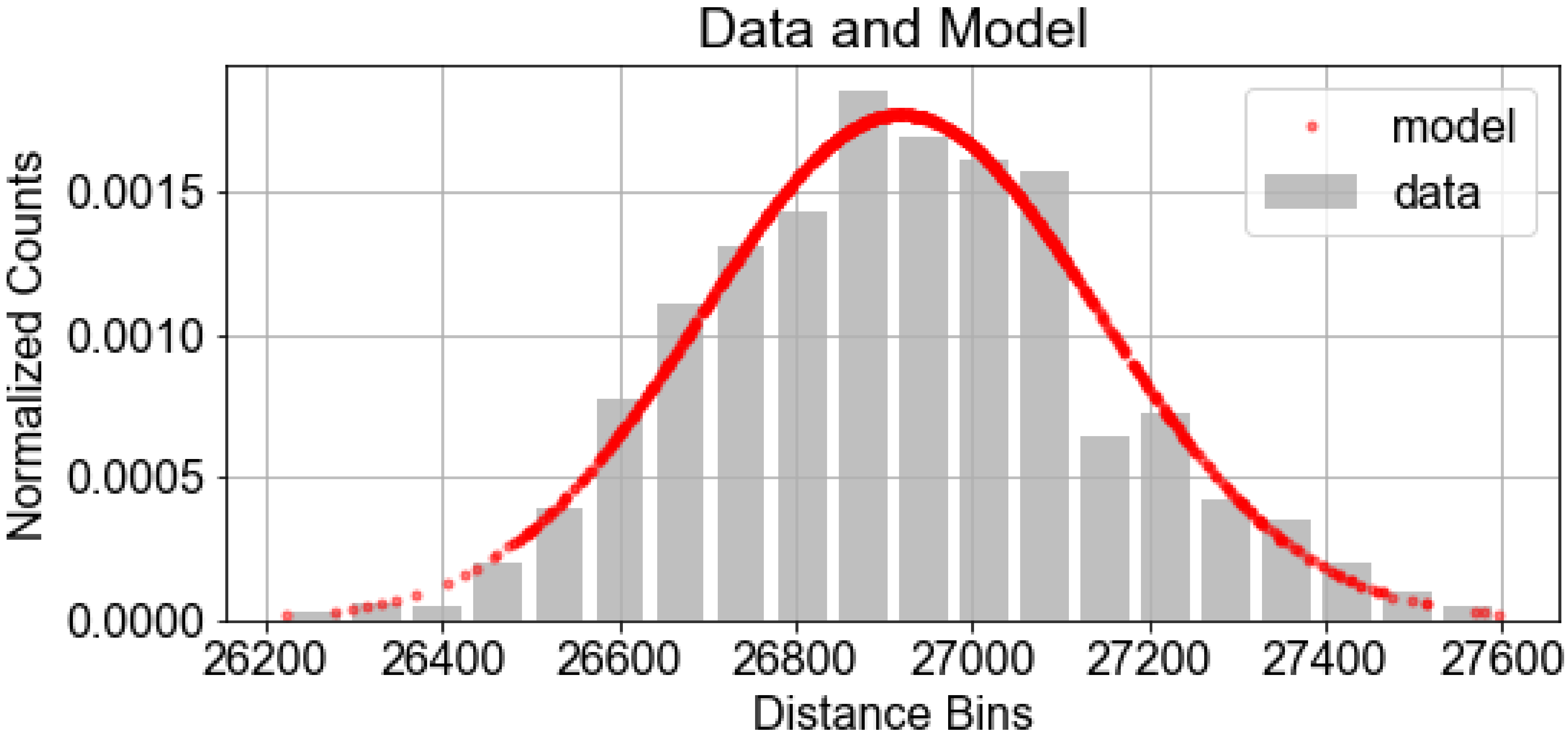
INTRODUCTION TO LINEAR MODELING IN PYTHON

Model Estimation and Likelihood

INTRODUCTION TO LINEAR MODELING IN PYTHON

Jason Vestuto
Data Scientist

Estimation


Data and Model
model
data
Normalized Counts
0.0000
0.0005
0.0010
0.0015
26200
26400
26600
26800
27000
27200
27400
27600
Distance Bins

Estimation

```
# Define gaussian model function
def gaussian_model(x, mu, sigma):
    coeff_part = 1/(np.sqrt(2 * np.pi * sigma**2))
    exp_part = np.exp( - (x - mu)**2 / (2 * sigma**2) )
    return coeff_part*exp_part
```

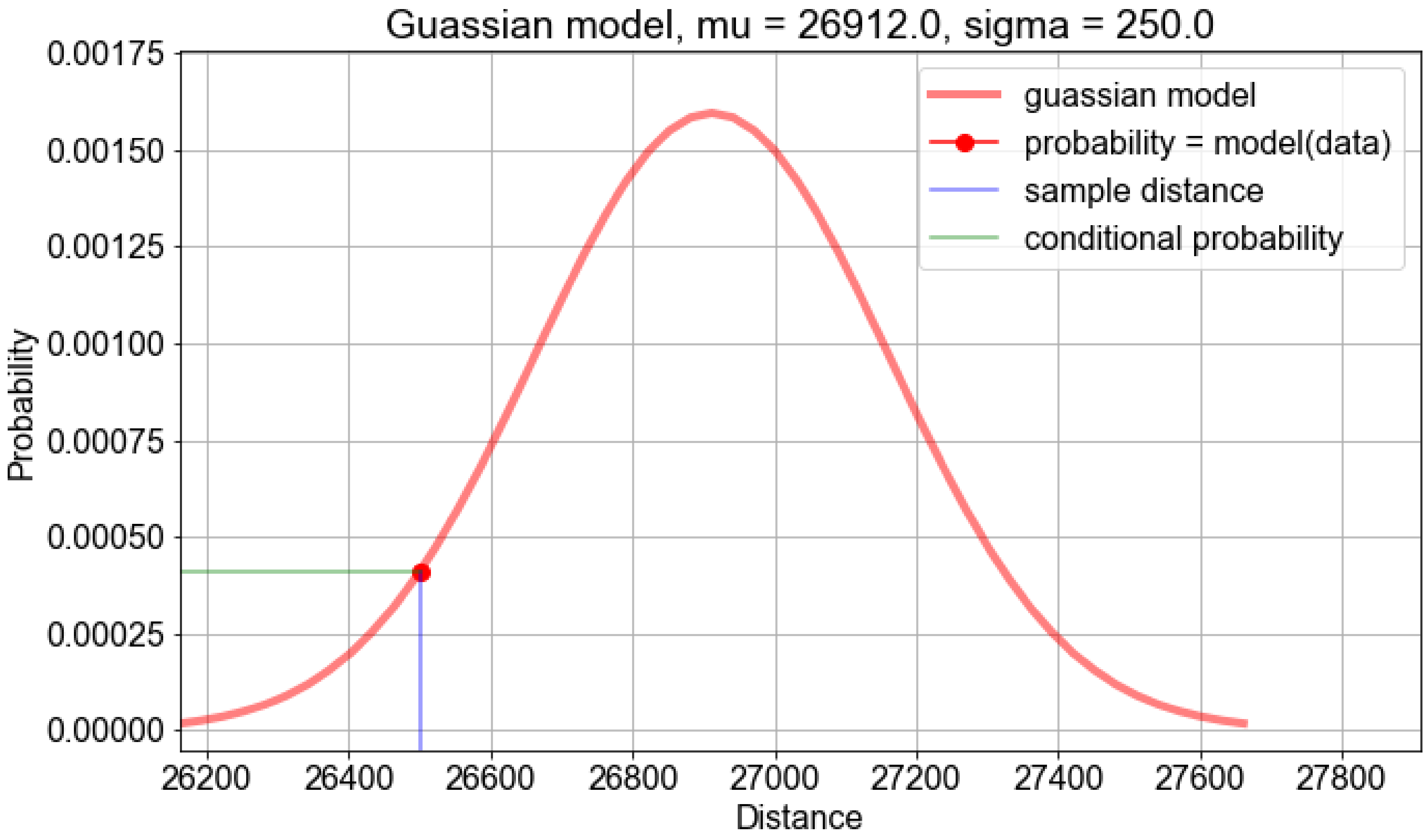
```
# Compute sample statistics
mean = np.mean(sample)
stdev = np.std(sample)
```

```
# Model the population using sample statistics
population_model = gaussian(sample, mu=mean, sigma=stdev)
```

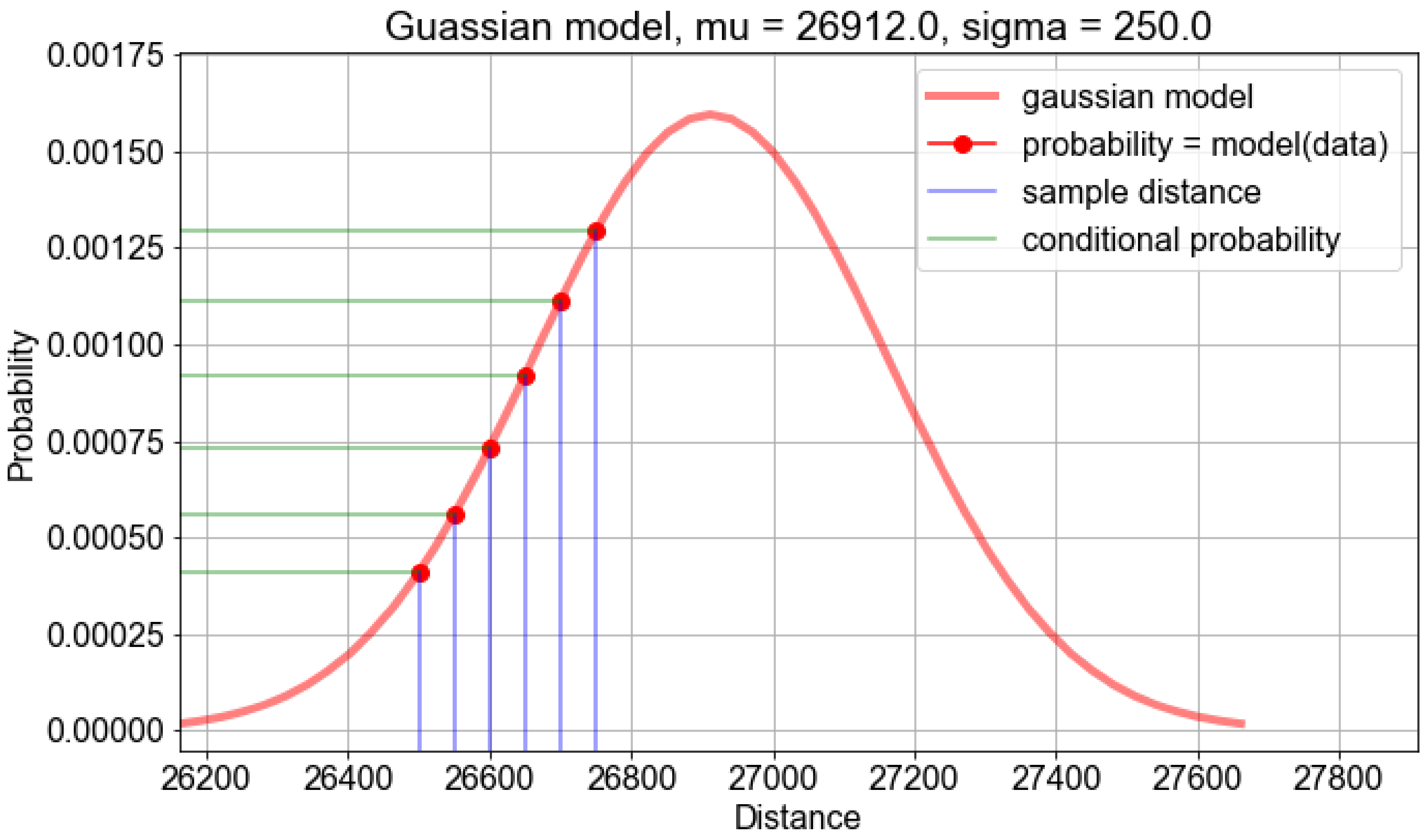
Likelihood vs Probability

- Conditional Probability: $P(\text{outcome A}|\text{given B})$
- Probability: $P(\text{data}|\text{model})$
- Likelihood: $L(\text{model}|\text{data})$

Computing Likelihood



Computing Likelihood



Likelihood from Probabilities

```
# Guess parameters
mu_guess = np.mean(sample_distances)
sigma_guess = np.std(sample_distances)
```

```
# For each sample point, compute a probability
probabilities = np.zeros(len(sample_distances))
for n, distance in enumerate(sample_distances):
    probabilities[n] = gaussian_model(distance, mu=mu_guess, sigma=sigma_guess)
```

```
likelihood = np.product(probs)
loglikelihood = np.sum(np.log(probs))
```

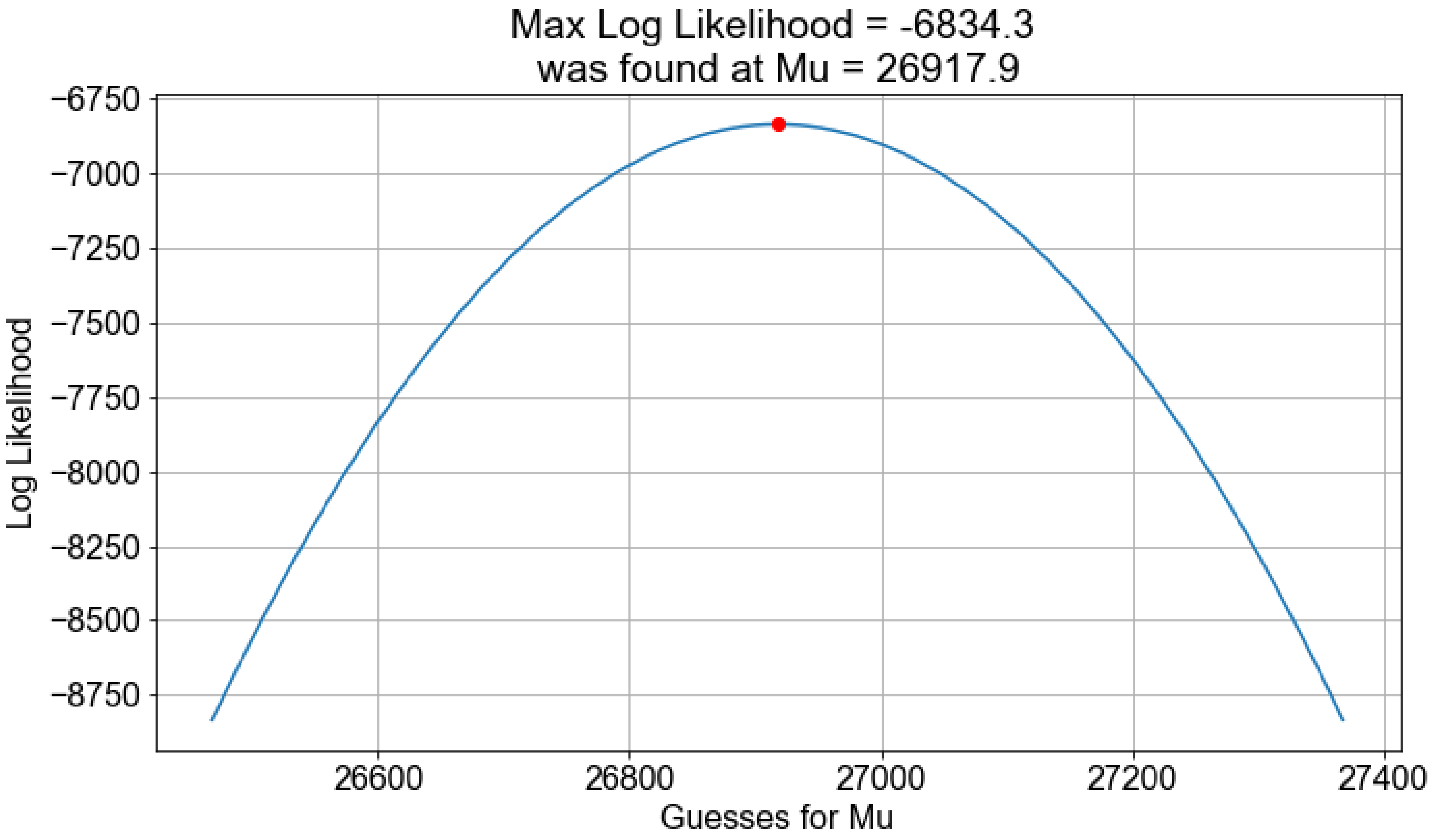
Maximum Likelihood Estimation

```
# Create an array of mu guesses
low_guess = sample_mean - 2*sample_stdev
high_guess = sample_mean + 2*sample_stdev
mu_guesses = np.linspace(low_guess, high_guess, 101)
```

```
# Compute the loglikelihood for each guess
loglikelihoods = np.zeros(len(mu_guesses))
for n, mu_guess in enumerate(mu_guesses):
    loglikelihoods[n] = compute_loglikelihood(sample_distances, mu=mu_guess, sigma=sample_stdev)
```

```
# Find the best guess
max_loglikelihood = np.max(loglikelihoods)
best_mu = mu_guesses[loglikelihoods == max_loglikelihood]
```

Maximum Likelihood Estimation



Let's practice!

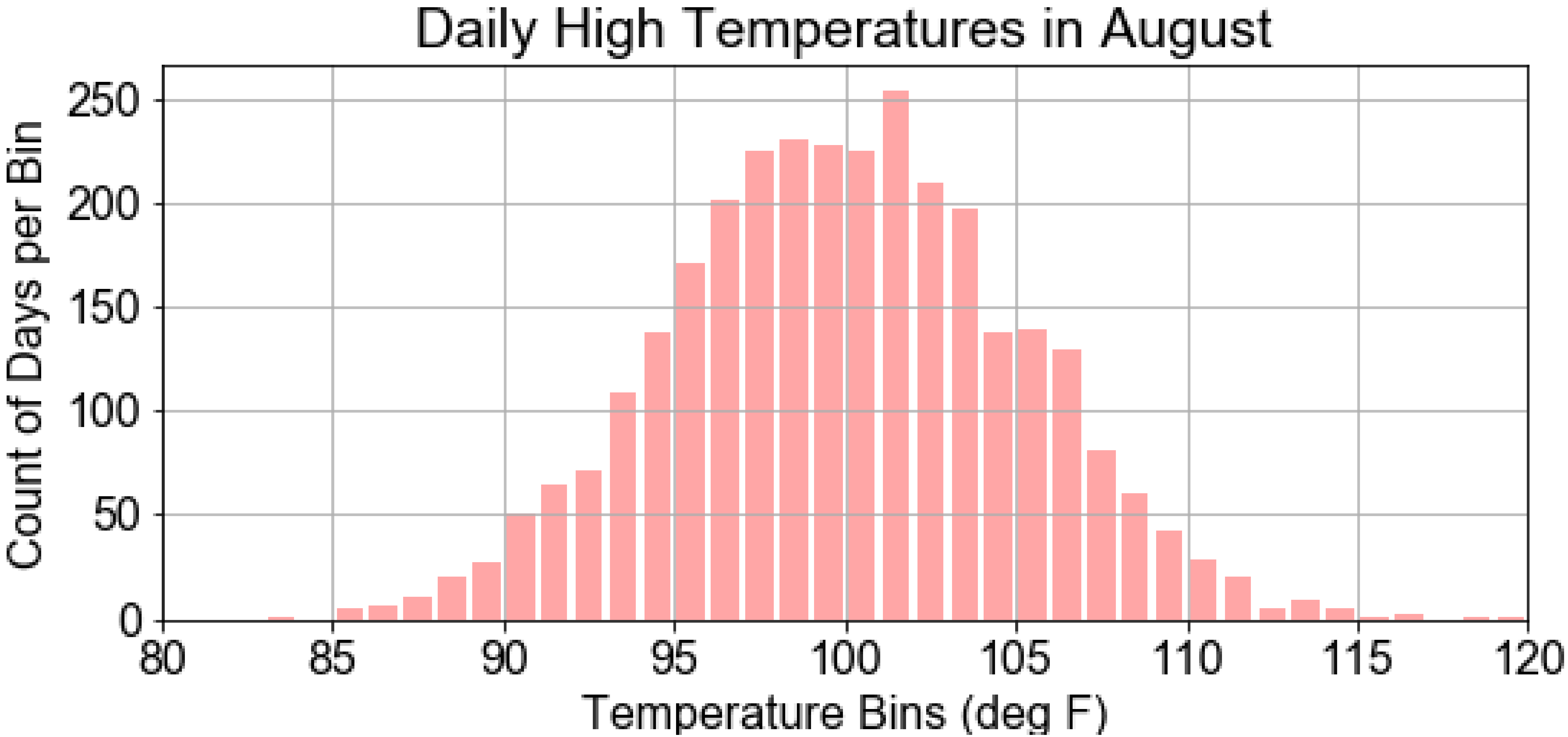
INTRODUCTION TO LINEAR MODELING IN PYTHON

Model Uncertainty and Sample Distributions

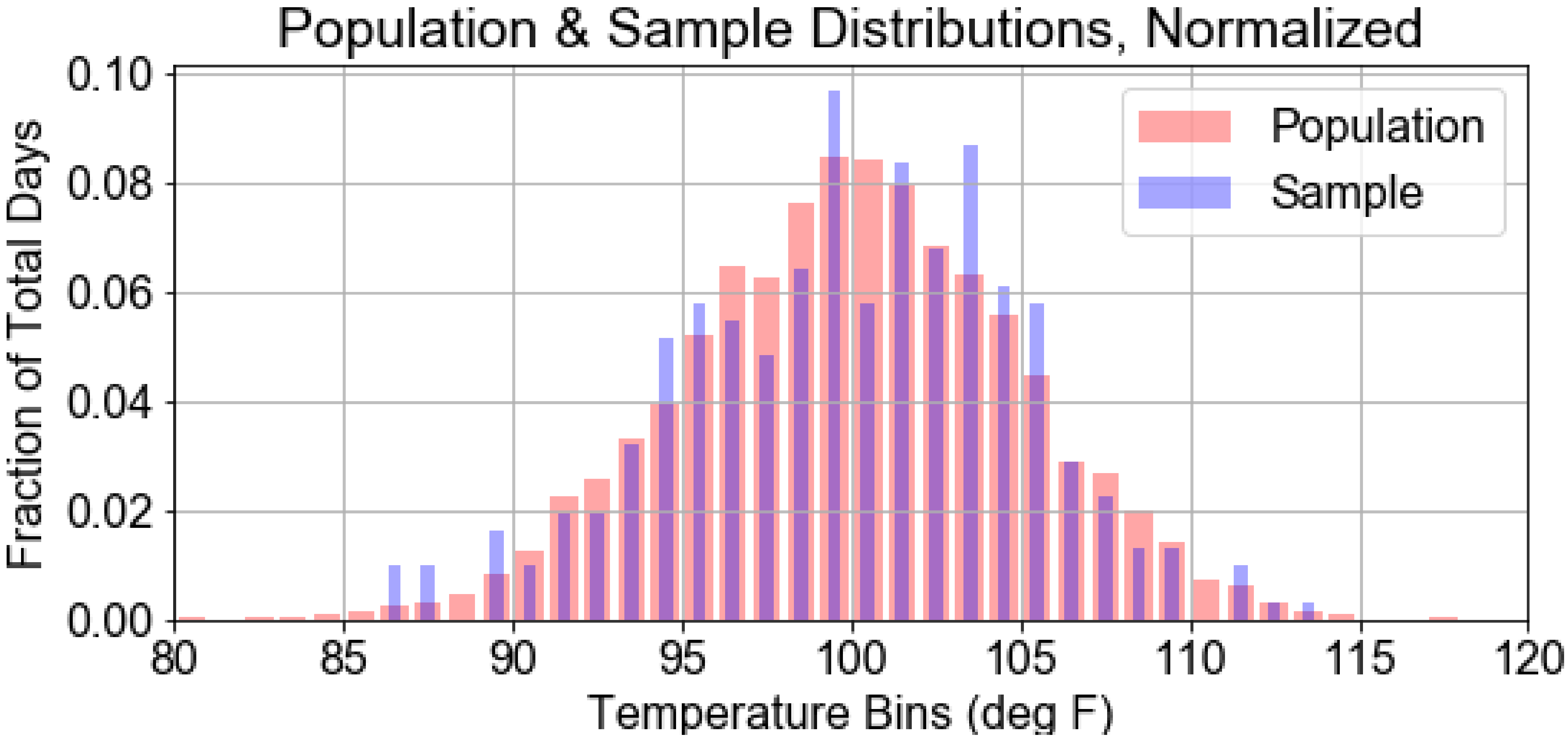
INTRODUCTION TO LINEAR MODELING IN PYTHON

Jason Vestuto
Data Scientist

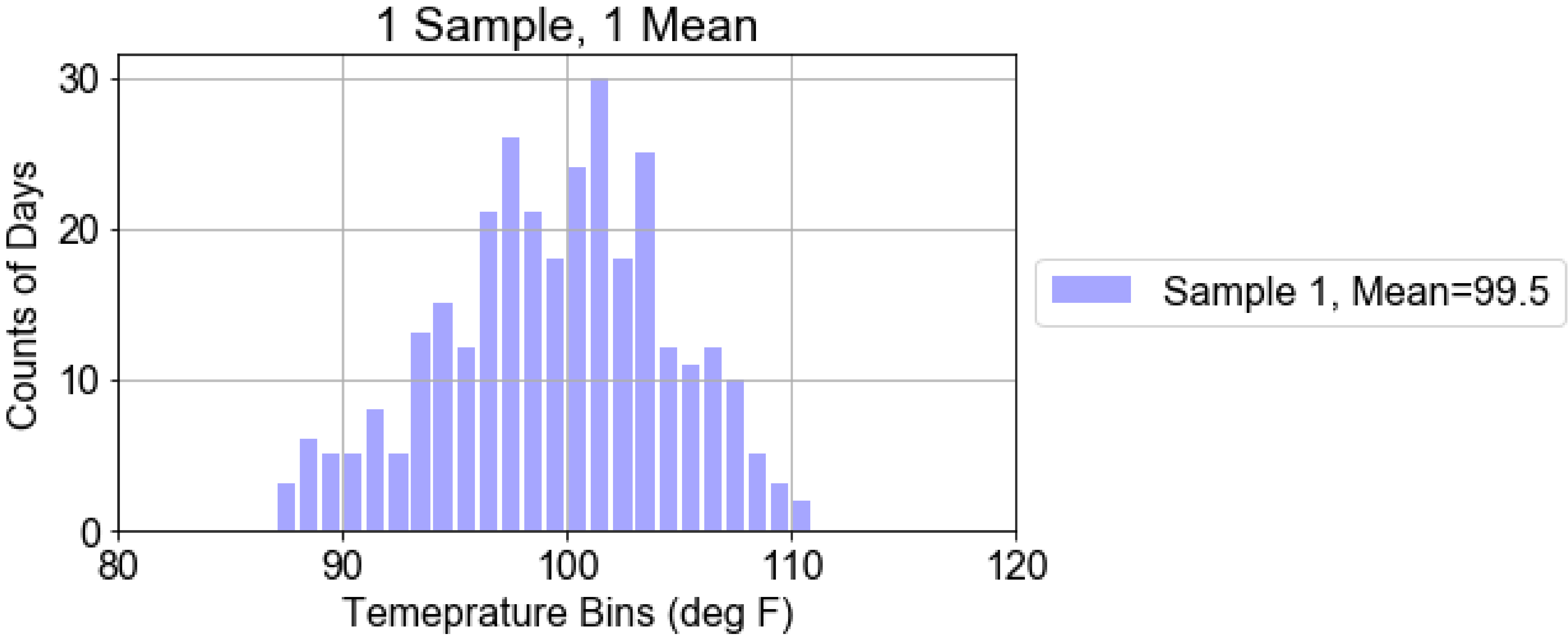
Population Unavailable



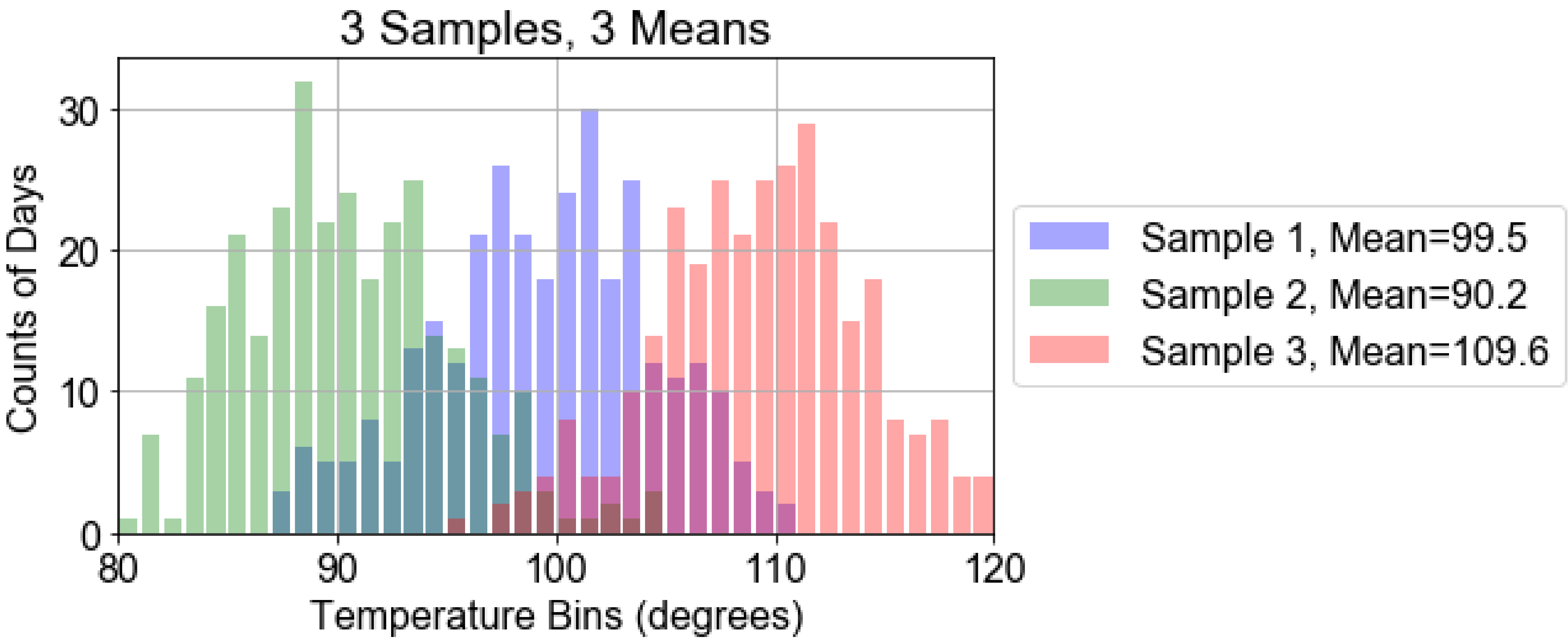
Sample as Population Model



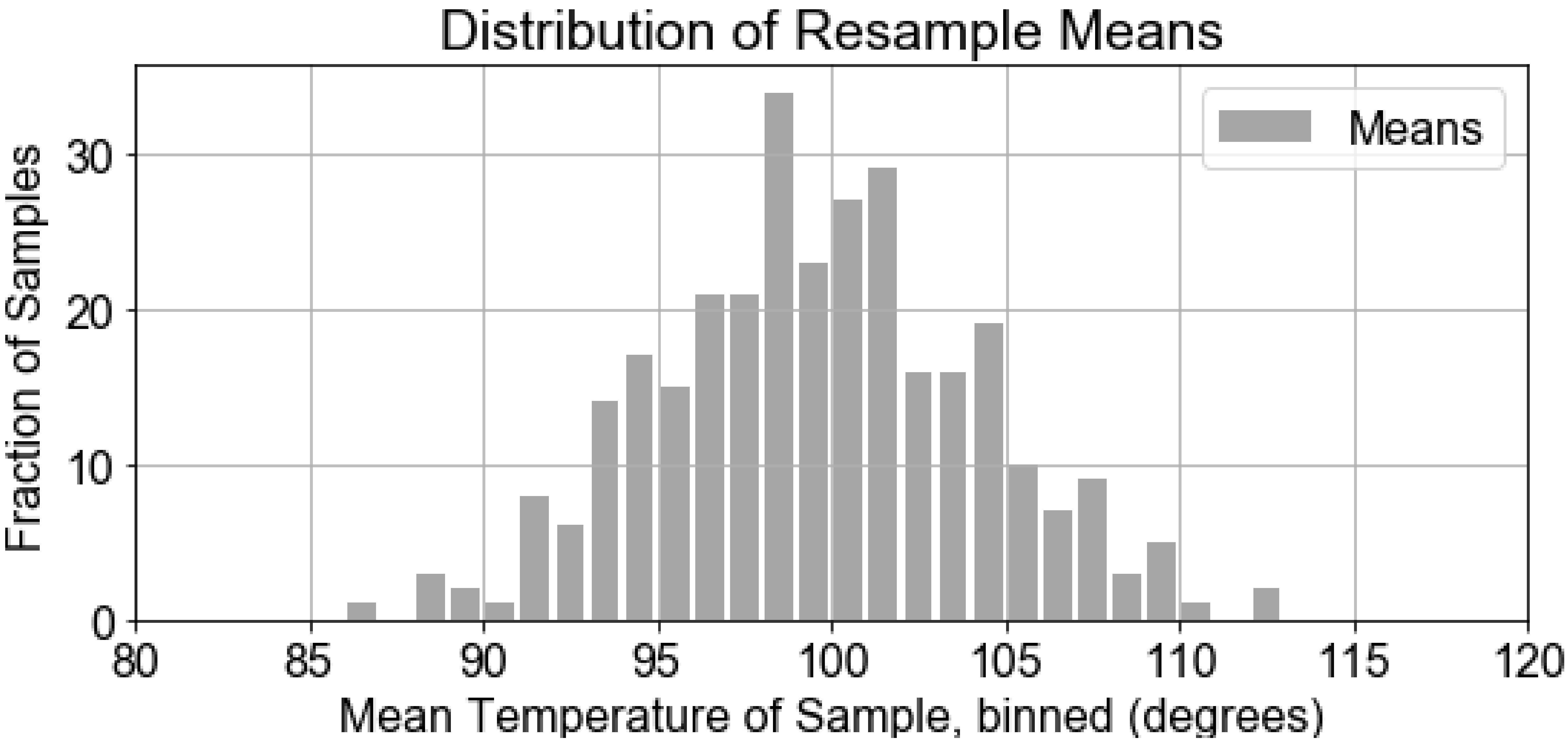
Sample Statistic



Bootstrap Resampling



Resample Distribution



Bootstrap in Code

```python
# Use sample as model for population
population_model = august_daily_highs_for_2017
```

```python
# Simulate repeated data acquisitions by resampling the "model"
for nr in range(num_resamples):
    bootstrap_sample = np.random.choice(population_model, size=resample_size, replace=True)
    bootstrap_means[nr] = np.mean(bootstrap_sample)
```

```python
# Compute the mean of the bootstrap resample distribution
estimate_temperature = np.mean(bootstrap_means)
```

```python
# Compute standard deviation of the bootstrap resample distribution
estimate_uncertainty = np.std(bootstrap_means)
```

Replacement

```python
# Define the sample of notes
sample = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

```python
# Replace = True, repeats are allowed
bootstrap_sample = np.random.choice(sample, size=4, replace=True)
print(bootstrap_sample)
```

```
C C F G
```

Replacement

```
# Replace = False
bootstrap_sample = np.random.choice(sample, size=4, replace=False)
print(bootstrap_sample)
```

```
C G A F
```

```
# Replace = True, more lengths are allowed
bootstrap_sample = np.random.choice(sample, size=16, replace=True)
print(bootstrap_sample)
```

```
C C F G C G A E F D G B B A E C
```

Let's practice!

Model Errors and Randomness

INTRODUCTION TO LINEAR MODELING IN PYTHON

Jason Vestuto
Data Scientist

DataCamp

Types of Errors

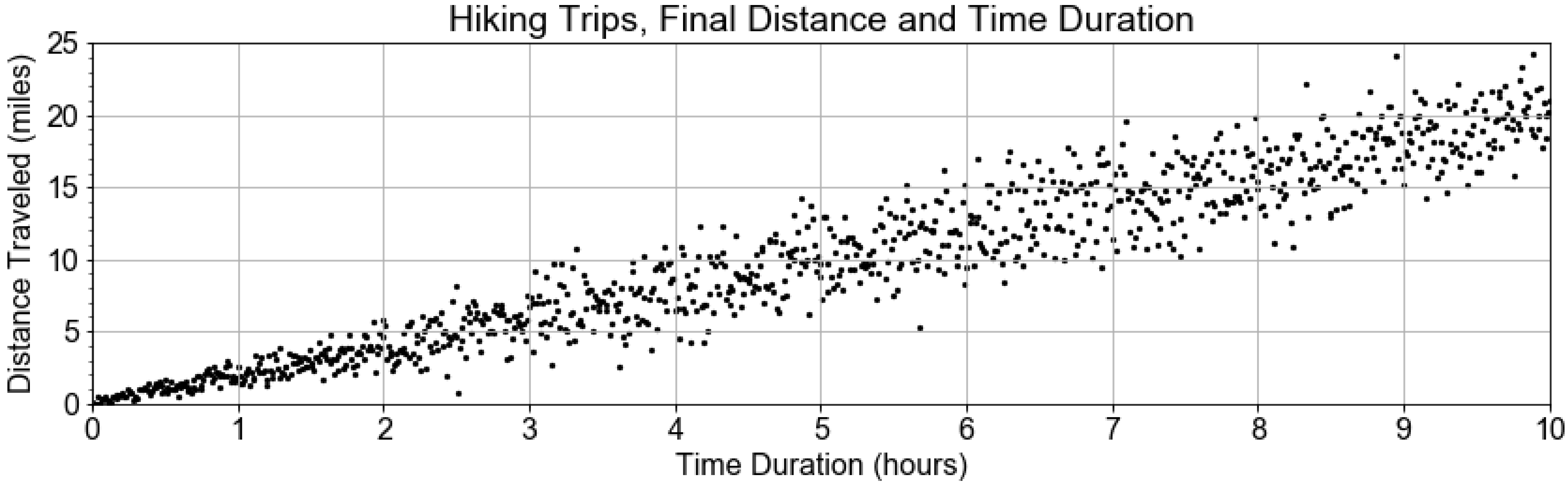
1. Measurement error
   - e.g.: broken sensor, wrongly recorded measurements
2. Sampling bias
   - e.g: temperatures only from August, when days are hottest
3. Random chance

Null Hypothesis

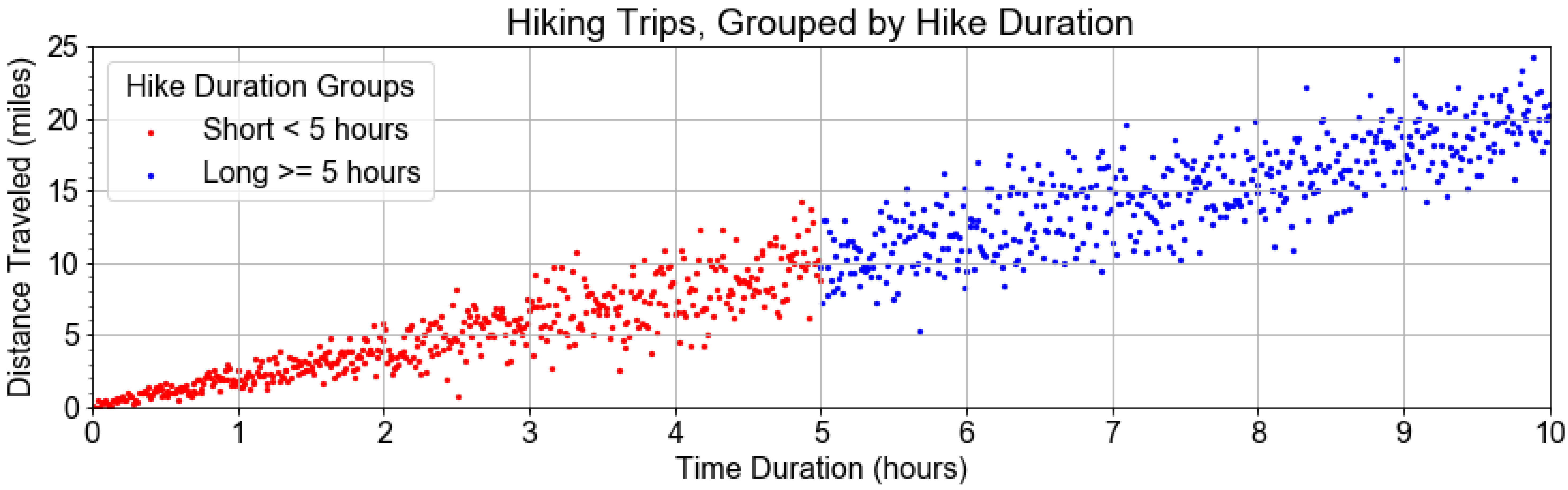
Question: Is our effect due a relationship or due to random chance?

Answer: check the Null Hypothesis.

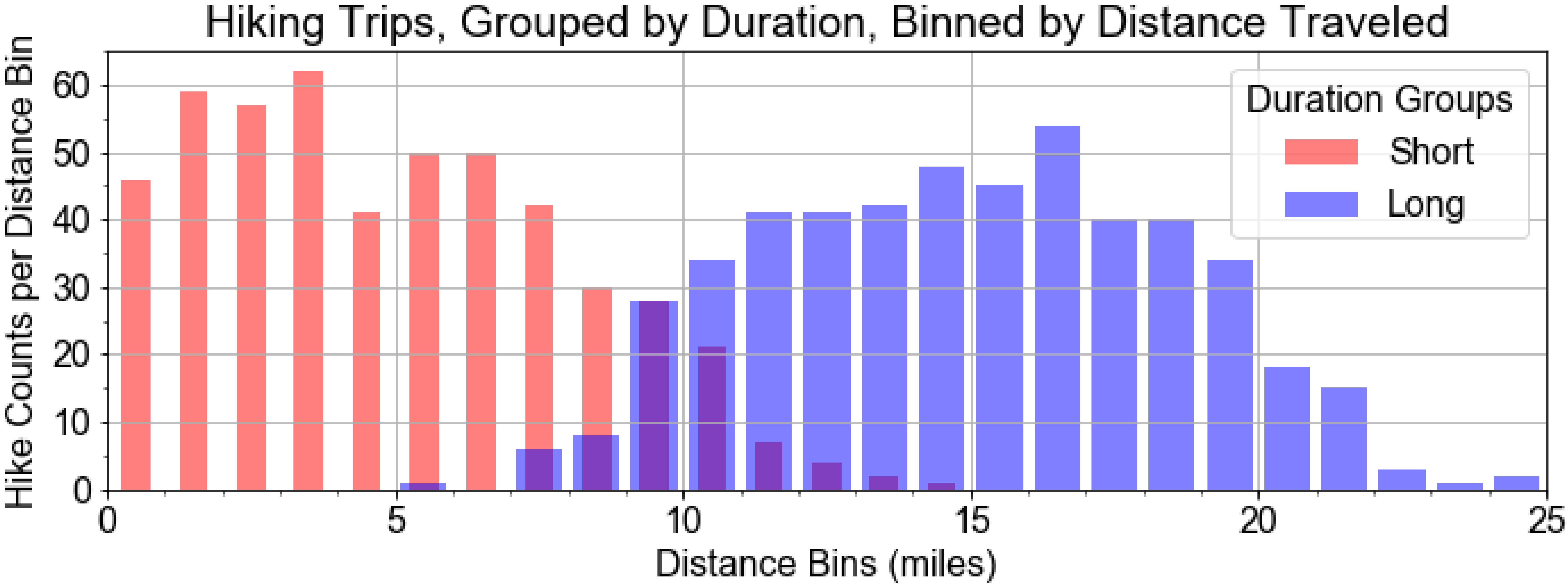
Ordered Data



Grouping Data



Grouping Data



Test Statistic

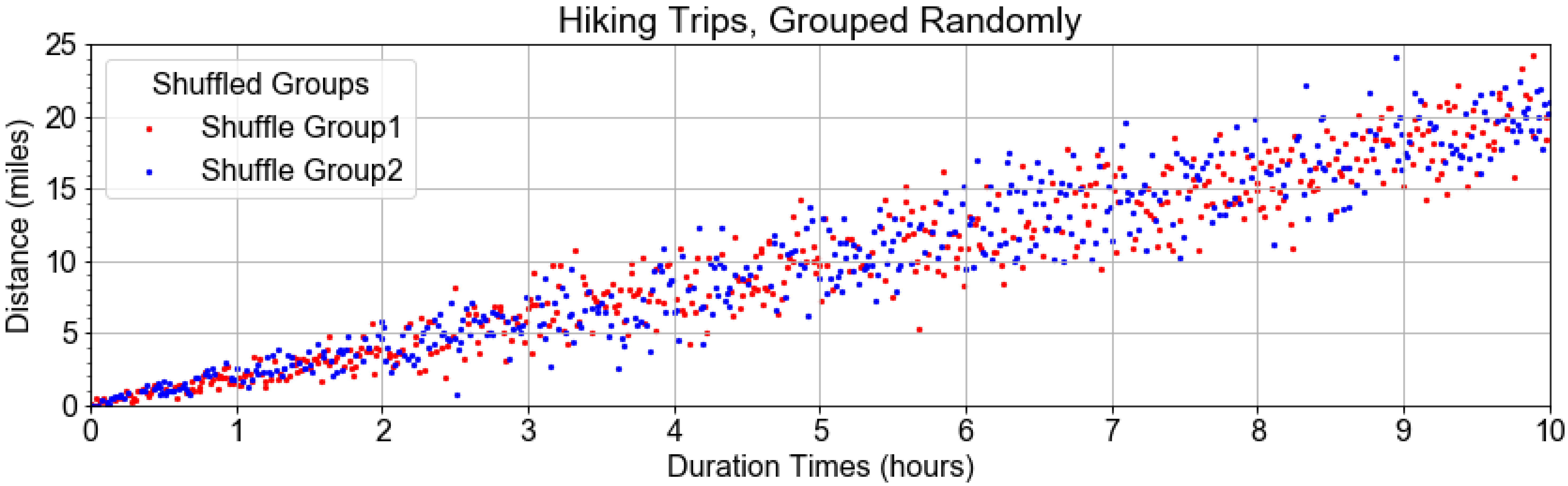
```
# Group into early and late times
group_short = sample_distances[times < 5]
group_long = sample_distances[times > 5]
```

```
# Resample distributions
resample_short = np.random.choice(group_short, size=500, replace=True)
resample_long = np.random.choice(group_long, size=500, replace=True)
```

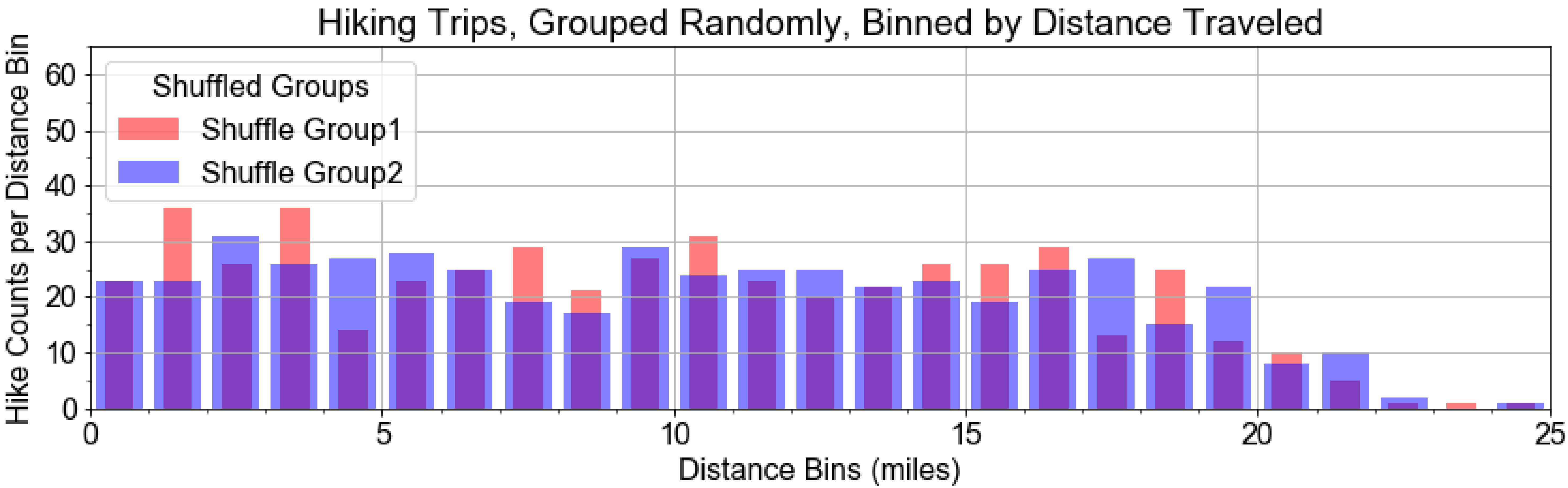
```
# Test Statistic
test_statistic = resample_long - resample_short
```

```
# Effect size as mean of test statistic distribution
effect_size = np.mean(test_statistic)
```

Shuffle and Regrouping



Shuffling and Regrouping



Shuffle and Split

```
# Concatenate and Shuffle
shuffle_bucket = np.concatenate((group_short, group_long))
np.random.shuffle(shuffle_bucket)
```

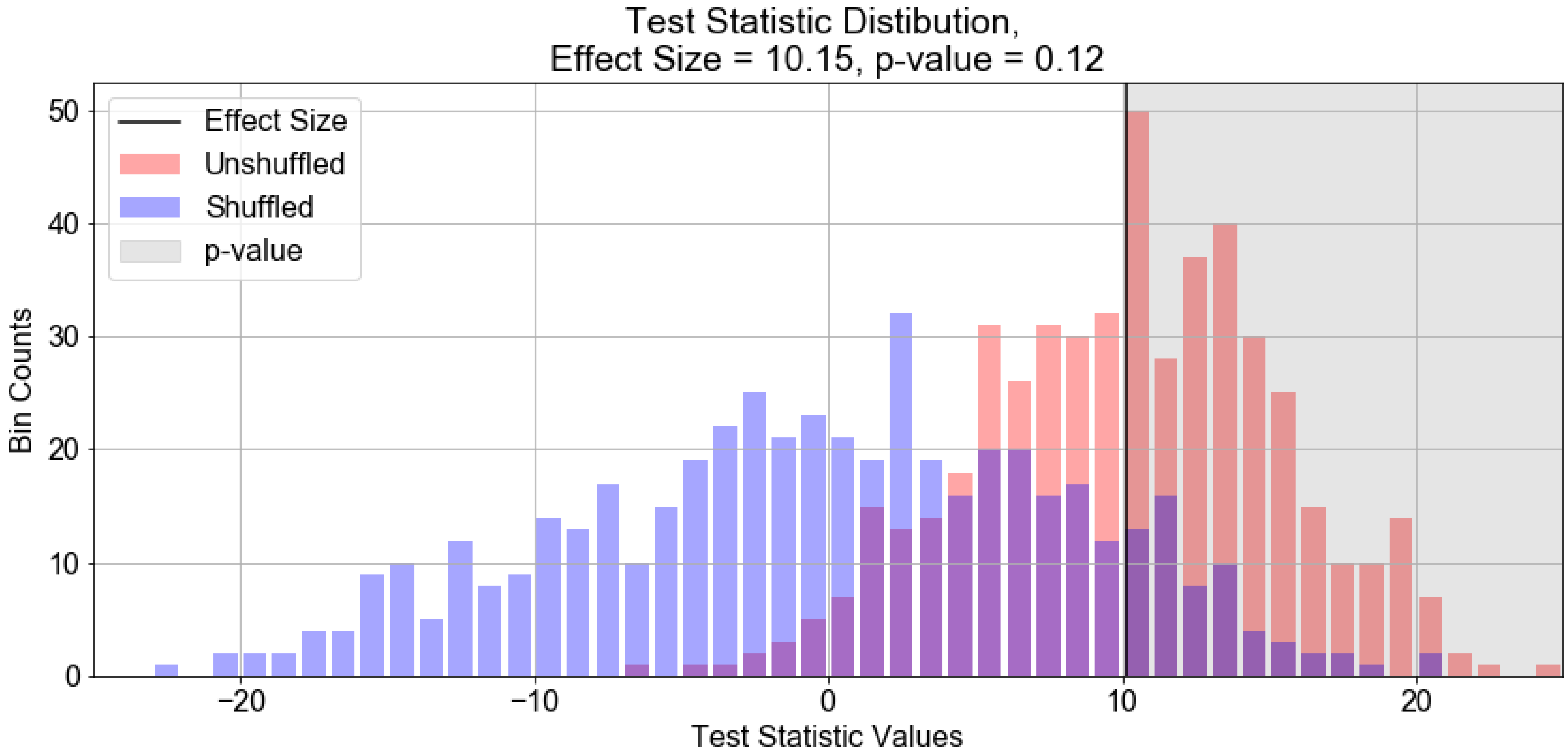
```
# Split in the middle
slice_index = len(shuffle_bucket)//2
shuffled_half1 = shuffle_bucket[0:slice_index]
shuffled_half2 = shuffle_bucket[slice_index+1:]
```

Resample and Test Again

```
# Resample shuffled populations
shuffled_sample1 = np.random.choice(shuffled_half1, size=500, replace=True)
shuffled_sample2 = np.random.choice(shuffled_half2, size=500, replace=True)
```

```
# Recompute effect size
shuffled_test_statistic = shuffled_sample2 - shuffled_sample1
effect_size = np.mean(shuffled_test_statistic)
```

p-Value


Test Statistic Distibution,
Effect Size = 10.15, p-value = 0.12
Effect Size
Unshuffled
Shuffled
p-value
Bin Counts
0
10
20
30
40
50
−20
−10
0
10
20
Test Statistic Values

Let's practice!

INTRODUCTION TO LINEAR MODELING IN PYTHON

Looking Back, Looking Forward

INTRODUCTION TO LINEAR MODELING IN PYTHON

Jason Vestuto
Data Scientist

Exploring Linear Relationships

- Motivation by Example Predictions
- Visualizing Linear Relationships
- Quantifying Linear Relationships

Building Linear Models

- Model Parameters
- Slope and Intercept
- Taylor Series
- Model Optimization
- Least-Squares

Model Predictions

- Modeling Real Data
- Limitations and Pitfalls of Predictions
- Goodness-of-Fit

Model Parameter Distributions

- modeling parameters as probability distributions
- samples, populations, and sampling
- maximizing likelihood for parametric shapes
- bootstrap resampling for arbitrary shapes
- test statistics and p-values

Goodbye and Good Luck!

INTRODUCTION TO LINEAR MODELING IN PYTHON