Big Data & Predictive Analytics

Lab 3 - Simulation & Hypothesis Testing

Objectives

At the end of this Lab, you should be able to

- gain a better understanding of standard distributions
- simulate a distribution for a random variable
- compute summary statistics and confidence intervals
- interpret the summary statistics of a distribution Note that this Lab illustrates the example simulation discussed in the Lectures on Simulation and Hypothesis Testing. In this Lab, you will estimate the range of expected profitability for a lemonade stand. The profitability of the lemonade stand depends on the number of customers arriving, the profit from the drinks they order, and the tips the customer may or may not choose to leave. The distribution of possible profits is thus, the joint distribution of customer arrivals, items ordered, and tips. In practice, such a complex distribution demands the use of simulation.

Simulate Probability Distributions

Exercise 1

1. Create a function `dist_summary` that:
   - Takes as inputs: a distribution, and a string with the name given to the distribution
   - Converts the input distribution data into a 1-dimensional ndarray with axis labels using `pd.Series(dist_data)`
   - Plots a histogram of the values using the Pandas hist method.
   - Then returns the summary statistics for the distribution.
2. Create a function `sim_normal` that:
   - Takes as inputs: a list of numbers `nums` (that contains the sizes `n` of the distributions to be generated), the mean, and the standard deviation of the normal distribution
   - Uses the normal function from the Python `numpy.random` library to generate random values drawn from the Normal distribution.
   - Calculates the summary statistics using the previously created function `dist_summary`
3. Run the function `sim_normal` for the input `nums = [100, 1000, 10000, 100000]` with `mean=600` and `stand_dev=30`.
4. Examine the returned results and note the following:
   - The median and mean converge to the theoretical values as the number of realizations (computed values) increases from 100 to 100000. Likewise, the confidence intervals converge to their theoretical values.

- The histogram of computed values comes to resemble the 'bell-shaped curve' of the theoretical Normal distribution as the number of realizations increases. Note that the histograms are affected by the quantization or binning of the values, which gives a somewhat bumpy appearance.

Exercise 2

1. Create a nearly identical function to `sim_normal`, called `sim_poisson` which generates a random Poisson distribution (instead of the normal one) using the `numpy.random` library
2. Run the function sim_poisson for the input `nums = [100000, 1000000]` and `mean=600`.
3. Examine the results for the Poisson distribution, and compare them to those for the Normal distribution. Note the following:
   - The mean and median (shown here as the 50% quantile) are at the theoretical values for the Normal distribution.
   - The 95% two-sided confidence intervals differ only slightly from those for the Normal distribution.
   - The values generated from a Poisson distribution are integers, which are reflected in the integer values for all the summary statistics. Further, this property leads to the uneven binning seen in the histogram.
   - Despite the uneven binning, the general shape of the histogram is nearly identical to that for the Normal distribution.
   - Overall, it is safe to conclude that for the large value of the mean for the number of customer arrivals there is no substantial difference between the Normal and Poisson distributions.

Simulate Specialized Random Variables

Exercise 3

Consider the following simulation code:

In [ ]:
```
# distribution of profits
def gen_profits(num):
    import numpy.random as nr
    unif = nr.uniform(size = num)#use uniform random numbers
    out = [5 if x < 0.3 else (3.5 if x < 0.6 else 4) for x in unif]
    # encode the function defined with probabilities
    return out
```

In [ ]:
```
# distribution of tips
def gen_tips(num):
    import numpy.random as nr
    unif = nr.uniform(size = num)
    out=[0 if x < 0.5 else (0.25 if x < 0.7 else (1.0 if x < 0.9 \
  else 2.0)) for x in unif]
    return out
```

Note: The function `gen_profits` generates random draws from a uniform distribution using the uniform function from the `numpy.random` library. Based on the values generated, the profit is computed using nested if else statements in the list defined in comprehension.

- Run the simulation function gen profits for the input `nums=[100000]` and use the function `dist_summary` to show the distribution of the simulated profits.
- Examine the results and observe that the distribution of profits per customer visit is as expected by looking at the frequencies of each profit value. Further, the median value is the most frequent profit level of 4.0.
- Likewise, proceed with the given function `gen_tips` and make relevant observations.

Simulate Lemonade Stand Income

Exercise 4

Consider the following code:

In [ ]:
```python
def sim_lemonade(num, mean = 600, sd = 30, pois = False):
    """Simulate the daily income for a lemonade stand.
    num: The number of simulations to run.
    mean: The mean number of visitors per day.
    sd: The standard deviation of the number of visitors per day
    pois: If `true` use the poisson distribution to model the
          number of visitors per day, otherwise use the normal
          distribution.
    """
    import numpy.random as nr
    import numpy as np

    ## number of customer arrivals
    if pois:
        arrivals = nr.poisson(lam = mean, size = num)
    else:
        arrivals = nr.normal(loc = mean, scale = sd, size = num)

    print(dist_summary(arrivals, 'customer arrivals per day'))

    ## Compute distibution of average profit per arrival using gen_profits



    ## Total profits are profit per arrival times number of arrivals.



    ## Compute distribution of average tips per arrival using gen_tips



    ## Compute average tips per day



    ## Compute total profits plus total tips.



    ## compute P(total_take < 3000)


    # return distribution summary
    return(dist_summary(total_take, 'total net per day'))
```

1. Read this code, and take note of the comments, in order to fill out the blanks and complete this function. Depending on the value of the pois argument, customer arrivals can be simulated from either a Normal or Poisson distribution.
2. Run the simulation sim_limonade for 100,000 values.
3. Examine each of the plots and the corresponding summary statistics. Note the following:
    - The Normal distribution of the customer arrivals is as expected with a mean and median of 600.
    - The distribution of profits per arrival appears as was observed previously.
    - Note that the distribution of total profits per day is a complex distribution which would be difficult to handle except by simulation.
    - The distribution of tips per arrival appears as was observed previously.
    - The distribution of total tips per day is a complex distribution which would be difficult to handle except by simulation.
    - The distribution of the final total net profit per day is the sum of the distribution of total profits per day and the distribution of total tips per day. This final distribution is even more complex with five peaks.
4. Compute the probability of making a profit that is less than 3000.
5. Enter the following command to run the simulation again, this time assuming a mean of 1200 customers per day with a standard deviation of 20: `sim_lemonade(100000, 1200, 20)`
6. Review the resulting statistics and plots, comparing them with the simulation results for a 600 daily customer average.