Computational Biology I: Quantitative Data Analysis
Frühjahrsemester 2019

Exercise 4

Athos Fiori and Pascal Grobecker

Central limit theorem

This part of the exercise deals with the following questions: How does the distribution of sums of random variates look like? Do limiting distributions exist? If so, how can they be characterized? Specifically, we will focus on the sums of random variates drawn from the following distributions :

- Uniform : $P(x) = \frac{1}{b-a}, \quad a \leq x \leq b, \quad a < b.$
- Exponential: $P(x) = ae^{-ax}, \quad 0 \leq x \leq \infty, \quad a > 0.$
- Pareto : $P(x) = \frac{b}{x^{b+1}}, \quad 1 \leq x \leq \infty, \quad b > 0.$

You can plot each probability density function for reasonable values of x in order to get an idea of the shapes the distributions take.

Scipy

SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python (http://docs.scipy.org/doc/scipy/reference/tutorial/general.html). Today we will start using Scipy in order to generate random variates (RVS) from the three distributions mentioned above. Have a look at the `import` statements and the pre-defined RVS functions in the `exercise4.py` file in order to understand how you can access functions from Scipy.

In `exercise4.py`, 7 functions are already provided :

- `uniformPDF`, `exponentialPDF`, `paretoPDF`
  Input : An array x, the parameters (a and/or b) of the distribution.
  Output : An array of probabilities $P(x|a, b)$.
- `uniformRVS`, `exponentialRVS`, `paretoRVS`, `normalRVS`
  Input : The size of the returned array of random variates m, the parameters (a and/or b, or μ, σ) of the distribution.
  Output : An array of random variates.

How to submit the exercise

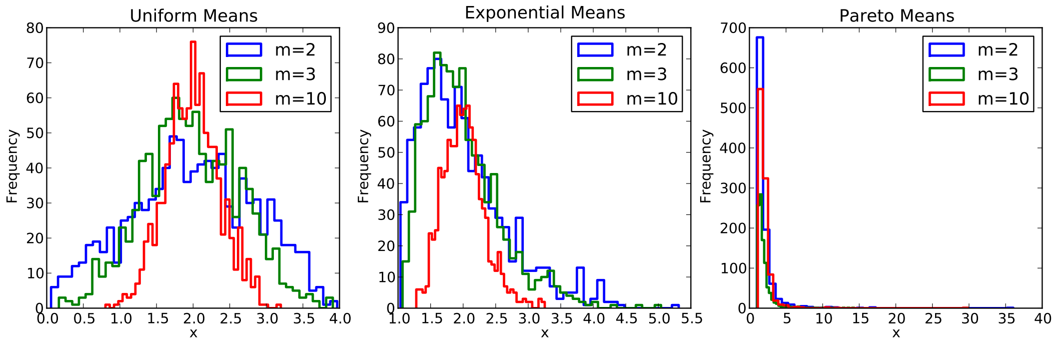
Submit your completed `Exercise4.py` file by email to biocomp1-bioz@unibas.ch. The submission deadline is Thursday, **28th March**, midnight.

1. Check that your functions are working properly before submitting them!
2. Don't forget to attach your exercise file!
3. Don't change the name of the file or the name of any function within the file!
4. Please send your solutions from the same email address that you used to registering at the course.

Homework

1. Complete the functions `uniformMean`, `exponentialMean` and `paretoMean` which calculate the mean of m random variates drawn from the respective distribution. (Use the predefined functions to generate the random variates.)

2. Complete the functions `uniformMeans`, `exponentialMeans` and `paretoMeans` which extend the previous functions such that they return an array of n means, each computed from m random variates generated from the corresponding distribution.
   Tip: You can use the `np.reshape` and `np.mean` functions, it might make your life easier. Pay special attention to axis attribute of `np.mean`.
   Tip: In order to get a feeling of the behavior of such means, have a look at your results by running `plot_DistributionMeans(dist)` which plot the histogram of $n = 1000$ means of a given distribution (`dist=0` : uniform, `dist=1` : exponential, `dist=2` : pareto), for $m = 2, 3, 10$ and the default distribution parameters. Your plots should look similar to the following ones.



3. As discussed in the lecture, cumulants can be derived from the moment generating function of a probability distribution. As it turns out the first four cumulants can be easily expressed in terms of the first four moments (have a look at the lecture slides). Once a distribution is specified you can derive analytical expressions for the cumulants from the analytical expressions of the moments. On the other hand, you can also calculate the *sample* moments of a data set. For example, the first sample moment is just the mean over all data points. In the following we will analyse the sample cumulants of a data set derived from its sample means. Complete the function `fourCumulants` which calculates the first four sample cumulants (mean, variance, skewness, kurtosis) for a given data set of n values.
   Important: Implement cumulants as given in the lecture! Do not use *skew* and *kurtosis* functions from *scipy* library! You are allowed to use `np.mean` and `np.var`.

4. Complete the function `cumulantsOfMeans` which, given one of the three distributions above, calculates the first four sample cumulants from $n = 10^4$ means of $m = \{1, 2, ..., M\}$ random variates (as computed in point 2.). Note, that the function takes one input argument `dist` which simply specifies the distribution as `dist=0` = uniform, `dist=1` = exponential and `dist=2` = Pareto distribution. Your function must return an array of size $4 \times M$ of your four cumulants for $m = 1, ..., M$.
   Tip: Check on the behavior of the four cumulants as a function of m for all three distributions by using the provided `plot_cumulantsOfMeans` function. I takes the distribution (`dist` $\in \{0, 1, 2\}$) as input (cumulants shown in log-space for the pareto distribution).

5. Complete the function `zScore` which returns the z-score (aka standard score) of a given vector x of n points:

$$z = \frac{x - \text{mean}(x)}{\text{std}(x)}$$

In this equation, one first removes the mean of x from every data point of x, which imply that z have a mean 0. Secondly, one divides by the standard deviation of x, which imply that the standard deviation of z is 1. You can verify this with your function `fourCumulants`.

6. Complete the function `gaussianApproximation` which should return n normally distributed random variables with mean μ and standard deviation σ. Those n random variables are computed from the means of $m = 10$ random variates from one of the three distributions mentioned above (so using one the functions from point 2). You have to pick the one which you think is most suitable.
Tip: In order to return n random variables with the desired mean and standard deviation, first use z-score (mean 0 and standard deviation 1). Then you can again transform your random variables as $z \cdot \sigma + \mu$ to obtain the desired distribution.
Tip: You can compare your results with the Gaussian distribution predefined in `normalRVS` using the predefined function `plot_gaussianApproximation`. Check for different values of mean and variance.

Theoretical Questions

1. Given the three probability distribution functions at the beginning of the exercise sheet, compute analytically the median of each distribution as a function of their respective parameters. The median x_m is defined as
$$\int_{x_{\min}}^{x_m} P(x)dx = \frac{1}{2}$$
where $x_{\min}$ is the smallest value on the domain of the distribution.

2. Given the three probability distribution functions at the beginning of the exercise sheet, compute analytically the mean of each distribution as a function of their respective parameters. The mean $\langle x \rangle$ is defined as
$$\langle x \rangle = \int_{x_{\min}}^{x_{\max}} xP(x)dx$$
where $x_{\min}$ and $x_{\max}$ are respectively the smallest and highest value on the domain of the distributions :
   - Uniform : $[x_{\min}, x_{\max}] = [a, b]$
   - Exponential : $[x_{\min}, x_{\max}] = [0, \infty]$
   - Pareto : $[x_{\min}, x_{\max}] = [1, \infty]$

   Tip : To calculate the integral for the exponential distribution, you can use integration by part :
$$\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx$$

3. Assume we have a quantity r which (based on out information I) has a distribution $P(r|D)$. We are now to make a best guess x at the true value of r. Let's assume that the *cost* of predicting x when the true value is r depends on the relative squared-error $(r-x)^2/r^2$. Specifically, we will assume the loss function $L(x,r) = (r-x)^2/r^2$. The optimal guess x_* minimizes the expected loss
$$\langle L(x) \rangle = \int L(x,r)P(r|I)dr.$$
Taking the derivative of $\langle L(x) \rangle$ with respect to x, you will find that x_* depends on the expectation values $\langle 1/r \rangle$ and $\langle 1/r^2 \rangle$. What is the optimal guess x_* in terms of these two expectation value?