ECEN321: Engineering Statistics
Laboratory Session 1
Probability Densities

Due: 9:00 a.m., Monday 23 March 2020

Below is the task description of the first lab. The labs have two objectives. The first is to help you understand the material. The second is that yoiu learn how to write a more formal report. The report you hand in should have a number of sections:

1. a short introduction describing and motivating the task (why is it important),
2. a theoretical section describing in more detail the task at hand and the approach taken (this section usually includes equations), and
3. a results section that describes the experimental setup, the results, and a discussion of what they mean.

You may add a conclusion and you may include your program as an Appendix. The report does not have to be very long.

It is preferred if you use Matlab/Octave or Python (`import numpy`) to do the labs of this course. However, if you have a really strong preference for some other language, that is fine.

In this lab we will generate some random quantities and use simple estimators for their properties. Commands you might not have seen before, but which you might need, are listed after each paragraph.

1 Normal Density

Generate a vector of 1000 normal (i.e., Gaussian) random variables having mean 2.5 and variance 16 (note that we want the variance to be 16, not the standard deviation). `randn()` / `numpy.random.randn()`

Find the sample mean and variance of the sample. `mean() var() std()`

Generate a histogram of the data using 30 bins. You might want to generate your own grid for the histogram using `linspace()` rather than leaving it up to `hist()` to decide its own. In this case you probably want three or four times σ each side of the mean.

We would like to compare the histogram with the true pdf. There are two scaling factors we must apply here:

1. The first is that the sum of entries in the histogram is N, whereas the integral of the pdf is 1. So to convert the histogram counts to probabilities we must divide by N.
2. The second is that for the pdf to represent a probability, we need to consider some small region δx over which an integral is performed. To convert the histogram counts to probability *densities* we need to divide by δx

Recall that the equation for the pdf of a normal random variable is

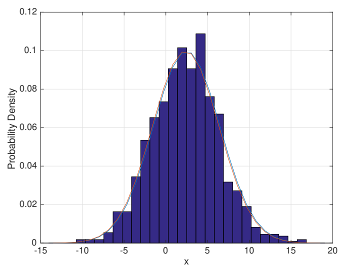
$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

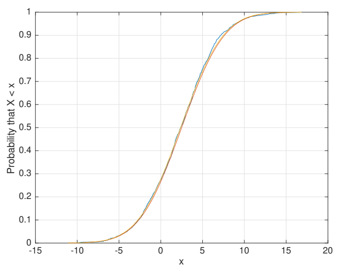
Overlay thee plots on one figure: 1. the histogram, 2. the pdf of the actual variable, and 3. the pdf using $\widehat{\mu}$ and $\widehat{\sigma}^2$ estimated from the data.

Now let's compare the empirical cdf of the sample with that of the population. We can generate the empirical cdf by plotting the data versus its rank/N. You might want to add an additional point which is slightly smaller than any of the data in the set, so that the probability that $\boldsymbol{X} < x$ is actually zero at that point and/or you might want to offset the rank by $1/(2N)$. `sort()` `erf()`

You'll notice that the sample and theoretical cdfs are much less distinguishable than are the pdfs. You can look more closely at a particular point on the plots by zooming using the mouse.

The plots should look something like this:





2 Student *t*

Now generate a matrix of size $M \times 10\,000$ of the same random variables as before, with $M = 5$.

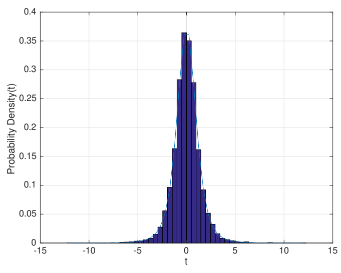
Generate a t value for each column by subtracting the true mean from the sample mean for each column, and dividing by the *sample* standard deviation and then multiplying by the square root of M (Note that by default, many Matlab functions operate on columns, unless we specify otherwise, or *unless there is only one row.* This last point can be a trap, since, we might sometimes end up with just one row without realising it, so it is a good idea to always specify the dimension you mean. This is also true in numpy, although the issues are slightly different.)

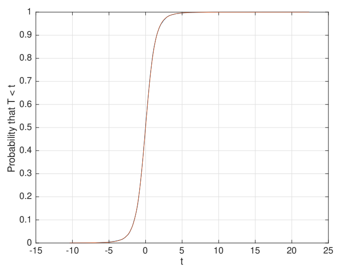
Generate a histogram of the data. The t distribution for this number of degrees of freedom has heavy tails, so you'll probably want to cover a large range e.g., to ± 10 or even more. (What happens to the histogram if the range is too small?)

Compare this with the theoretical pdf for $\nu = M - 1 = 4$ degrees of freedom (`tpdf()` or `gamma()` or maybe `gammaln()`).

$$p(x) = \frac{\Gamma((\nu + 1)/2)}{\sqrt{\nu\pi}\,\Gamma(\nu/2)\,(1 + x^2/\nu)^{(\nu+1)/2}}$$

Plot the sorted data versus the rank, and compare this with the theoretical cdf. `tcdf()`





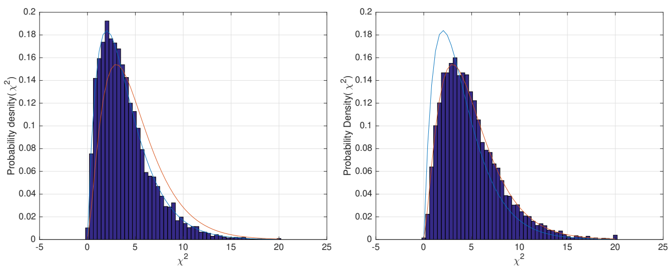
3 Chi Squared Density

Generate a chi squared random variable by multiplying the sample variance for each column (of the matrix from Section 2) by $\frac{(5-1)}{\sigma^2}$. Generate a histogram for this also.

Compare this with the theoretical pdf for $\nu = M - 1 = 4$ degrees of freedom `chi2pdf()`:

$$p(x) = \frac{(1/2)^{\nu/2}}{\Gamma(\nu/2)} x^{\nu/2-1} e^{-x/2}$$

Also generate a chi squared random variable from the sum of differences from the *true mean*, divided by σ^2. Compare this with the theoretical pdf for $\nu = M = 5$ degrees of freedom:


Probability desnity(χ^2)
χ^2
Probability Density(χ^2)
χ^2