IMPRS Blackboard Lecture on Statistical methods – Exercises

Problems

Problem 1. Sampling from distributions in numpy. In order to familiarize yourself with some of the standard distributions generate histograms for each of the following problems using numpy.random:

1. Generate 10000 samples of a uniform distribution on the interval [0,27[. Show a histogram with matplotlib.
2. Generate 10000 samples describing the number of positive outcomes of a coin toss experiment with n=27 repetitions and a success probability of 0.45.
3. Generate 10000 samples describing the repeated drawing of balls from an urn with replacement. The balls come in 6 colors and we draw 27 times. The success probability for each color is 2/36, 4/36, 8/36, 12/36, 7/36, and 3/36. Plot the histograms of the number of balls for each color by using the corner plot library.
4. Draw 10000 samples describing the repeated drawing of balls from an urn without replacement (equivalent to a simultaneous draw) for two colors (6 ”good” and 35 ”bad” outcomes). We draw 27 times. Plot the histogram for ”good” outcomes.
5. Model the distribution of rare events with an average count rate of $c = 0.001$ cps for time intervals of 100 minutes. Plot the histogram of 10000 samples.
6. Draw 10000 samples of the waiting time (number of steps) distribution for 2 color urn drawings with replacement, i.e., the number of excess steps required to see the r-th success, for r = 9 and a success probability of 0.45. Plot the histogram.
7. Sample from the waiting time distribution (discrete) for the first success, as above for r=1 and a success probability of 0.1. Plot the histogram of the 10000 samples.
8. Draw from the waiting time distribution (continuous) for the r event with an average count rate of 0.001, r=5. Plot the histogram of the 10000 samples.
9. Same as before, just for the waiting time of the first event, i. e., r=1.
10. Waiting time distribution (continuous) for the r^{th} event when a fixed number of events n = 22.7 will certainly happen within the unit time interval. Draw 10000 samples for r=18.8 and show the histogram.
11. Draw from a normal random variable with a center of m=5 and the variance of v=3. Draw also from a log-normal distribution as implemented in numpy.random.lognormal with the same parameters. Plot histograms of 10000 samples of the normal distribution and of the logarithm of 10000 samples of the log-normal distribution.

Problem 2. Monte Carlo simulations of random variables based on uniform random numbers on [0,1[.

1. Sampling from binomial distribution. Use a uniformly distributed random variable to simulate the number of successes for 20 coin tosses (the sum of a Bernoulli sequence) for a success probability p=0.7. Plot a histogram of 10000 samples and compare with the probability mass function of the binomial distribution as provided by the scipy.stats library.

2. Sampling from exponential distributions. Use the quantile transformation (the inverse CDF) to generate exponentially distributed random numbers from uniformly distributed random numbers on [0,1[for a count rate of c=0.001. Compare with the probability mass function as provided by scipy.stats.

3. Sampling from a Poisson distribution. Follow example 3.41 in "Georgii: Stochastics" and implement the pseudo-code:

```
$v \leftarrow 1,\ k \leftarrow -1$
repeat $v \leftarrow U\ v,\ k \leftarrow k+1$
until $v \leftarrow \mathrm{e}^{-\lambda}$
$N_\lambda \leftarrow k$ .
```

   Compare with the probability mass function as provided by scipy.stats for $\lambda = 3.3$.

4. Polar method for sampling from normal distribution. Can be shown that for

$$X = (X_1, X_2) := 2\sqrt{-\log|Z|}\frac{Z}{|Z|}\ ,$$

   with $Z = (Z_1, Z_2)$ a random variable uniformly distributed on the unit disk, the variables X_1 and X_2 are independent and standard normally distributed. Z can be obtained by rejection sampling, resulting in the following pseudo-code:

```
repeat
$u \leftarrow 2U - 1,\ v \leftarrow 2V - 1,\ w \leftarrow u^2 + v^2$
until $w \leq 1$
$a \leftarrow \sqrt{(-2\log w)/w},\ X_1 \leftarrow au,\ X_2 \leftarrow av$ .
```

   Implement this algorithm and draw 10000 samples for the standard normal distribution.

Problem 3. Normality and central limit theorem.

1. Sums of i.i.d. (independent and identically distributed) uniformly random variables. Generate 10000 samples of the sum of 100000 i.i.d. random variables. To do this, write in in form of a python function and use numba to increase speed. Plot histogram and compare with the normal PDF and a suitably scaled binomial PMF.

2. Sums of Cauchy distributed random variables. Similar to the problem above, just for Cauchy distributed random variables. Plot the resulting histogram. Appreciate how strange it looks and contemplate why.

3. Normality and error propagation. Write a function to simulate the distribution of flux density ratios and spectral indices calculated from two flux densities when both have normally distributed measurement errors. Input of this function should be frequency ratio of the two frequencies at which the flux densities are measured, the spectral index, signal to noise ratio, and the absolute flux density at one of the frequencies. Write a second function with the same input that uses the uncertainties package in python to formally propagate the uncertainties. Plot the histogram of the flux density ratio and the flux density index. Are those histograms Gaussian? Now change the frequency ratio and/or the signal to noise ratio, compare the spectral index distributions with the values of the formal error propagation. Plot the histograms, again.

Problem 4. **Empirical data**

1. Use the astropy table package to read-in the machine readable table *"table_3_witzel18.txt"* and remind yourself how to access the data and the column entries. Import the table into pandas DataFrame and display the DataFrame. Use it to obtain average and standard deviation of flux density for each instrument.

2. Calculate the empirical CDFs of flux densities for each of the instruments.

3. Sample from the empirical flux densities with the naive method and with the so-called 'dithering' sampler and compare the resulting distributions with the original flux density distributions.

Problem 5. **Markov chains, ergodic theorem and asymptotic stationarity.**

1. Show by matrix multiplication in python that for the transition matrix

$$\Pi = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 \end{pmatrix} ,$$

applies

$$\begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 \end{pmatrix}^n \underset{n\to\infty}{\longrightarrow} \begin{pmatrix} 1/2 & 3/8 & 1/8 \\ 1/2 & 3/8 & 1/8 \\ 1/2 & 3/8 & 1/8 \end{pmatrix} .$$

2. MC walker on three outcomes. Write a function that realizes a Markov chain with the transition matrix above and stores the result of each step in a chain. Show that for a sufficient number of steps the fraction of times each outcome occurs approaches the probability of the limiting matrix. For an arbitrary starting point, show the frequency of occurrence of each state as a function of steps since the starting point.

Problem 6. Linear regression and tests

1. Use the presented set of analytical formulas to calculate a linear regression for $(x, y, \delta y)$. Make mock-data with individual error bars for each point and run the linear regression on the mock-data. Plot the results and draw 1000 parameter sets from the confidence interval of the regression parameters and over-plot the corresponding lines.

2. χ^2 test. Write a χ^2 function for the case above and minimize the the χ^2 with scipy.optimize.minimize. Plot the χ^2 minimization results and calculate the minimal χ^2-value, the reduced-χ^2 value, and the p-value of the fit. For the latter use scipy.stats.chi2.cdf.

3. student's t-test tests. Generate suited, normal distributed mock-data and execute the one-sided student's t-test for the expectation value for various population means.

4. Generate suited mock-data of several normal distributed random variables with equal variance and execute the two-sided t-test.

5. Generate suited general normal distributed mock-data with equal variance and run the relative t-test.

Problem 7. Sampling a 5-dimensional normal distribution with random mean and random covariance matrix using emcee.

Follow the Quickstart guide[1] of the emcee documentation to set up a multi-dimensional Gaussian distribution and to sample it with emcee functions. This knowledge will be instrumental for the next, much more complex problem.

Problem 8. Fitting realistic orbit data of the Galactic Center star S(0)-2 in order to constrain mass and distance of the supermassive black hole, Sgr A*.

1. Write a function in order to predict the projected RA and DEC position and the radial velocity (RV) of S-2 for any given time from the published orbital elements. These can be found in Do et al. (2019, Table 1[2]). In this paper, Equations S-1 to S-16 present all conventions used for the galactic center and can be directly implemented. In order to solve for the eccentric anomaly, Equation S-3, use the algorithm described in Alzener's overview article, Chapter 7, page 60[3]. It is easiest to write three functions, one to solve for the eccentric anomaly, one to compute the position and RV for a given time, and a wrapper to calculate orbits for an array of time points.

2. Choose 1000 linearly-spaced points in time for two S-2 orbits of 16 years around the 2018 closest approach. Plot the DEC vs. RA, RV vs. time, and the S-2 distance to Sgr A* and compare with Do et al. 2019 paper.

3. Generate a sequence of time-points representative of an actual observing sequence for 15 years starting in 2003. For simplicity assume that astrometry and RV are always measured simultaneously. Each year we observe 3-4 times between decimal year of 0.254 and 0.626 (April 15th to August 15th) and the probability of an observation to occur is equally distributed in this time frame. For the time of closest

[1]https://emcee.readthedocs.io/en/stable/tutorials/quickstart/
[2]https://arxiv.org/pdf/1907.10731.pdf
[3]http://www.ugastro.berkeley.edu/infrared10/adaptiveoptics/binary_orbit.pdf

approach, let's throw in 8 more uniformly distributed observations around $T_0 \pm 0.1$ [year]. Draw a realization of this observing scheme, add realistic Gaussian measurement errors (Do et al. 2019), and plot the same diagnostics as above.

4. χ^2 minimization: use scipy.optimize.curve_fit and the orbit generating functions to fit an orbit to the generated mock-data. Use boundaries that reflect the definition ranges of the individual parameters. Drive and print-out the formal fitting error from the resulting covariance matrix. Plot DEC vs. RA and RV vs. time for the mock data over-plot the best-fit orbit. Show the residuals as a function of time for RA, DEC, and RV, including measurement errors. Load the input and output parameters of the model and the fit into a Pandas DataFrame, compute the fractional best-fit error, and determine the fractional difference between ground truth and best-fit values.

5. Likelihood maximization: Write down a function for a Gaussian logarithmic likelihood based on our model. Use scipy.optimize.minimize to minimize the negative log-likelihood for a suited set of initial values and boundaries. Evaluate the results as before in form of a Pandas table and diagnostic plots as described above.

6. MCMC sampling of posterior distribution: write a function for generating the log of a flat prior on the ranges of the boundaries and a second function that generates the total log-probability, i. e., the sum of the log-prior and the log-likelihood. As in the sampling example of Problem 6, sample the posterior distribution with emcee around the maximum likelihood solution of the previous step. Use 32 walkers, 7000 steps. If it is too slow, use pool for multi-threading the sampler as described in the emcee documentation. Pickle the resulting sampler variable for later use. Look at diagnostic as parameter value as a function of step number or the auto-correlation time. Extract the sample chain and plot it in a corner plot. Over-plot 100 of the resulting orbits onto the mock-data in the DEC vs. RA and RV vs. time plots. Show the final results, including fractional errors and input and output differences in a Pandas table.

Homework

Problem 1. Read the proof of the central limit theorem and summarize in a few sentences the individual steps (without too much detail) of the proof.

Problem 2. Give a detailed description of data in context of your own project. What are the statistical properties of the raw data? What are the calibration and reduction steps? What are the statistical properties of the reduced data? What is the most limiting factor for the signal to noise and what are possible biases?

Problem 3. The MPIfR runs a VLBI correlator. How does the signal that is correlated look like? What are its statistical properties? And in particular, why can we down-mix the signal without losing essential information?

Problem 4. Similar to the MC simulations regarding the spectral index, set up a set of simulations for linear polarization data. In particular, simulate the statistical properties of the differential measurements for Stokes Q and U and the resulting properties for the polarization angle, polarization degree, and polarized flux density. Show the statistical properties of polarization angle for variable flux densities when the flux densities approach the noise floor.

Problem 5. Modify the S2 mock-data generation to make the orbit fit more realistic. In particular, separate astrometric measurements and RV measurements in time and introduce individual error bars for each point and under-estimate the error somewhat as described in the introduction of emcee[4]. Include an offset and proper motion of the origin of the coordinate system. In reality, this is the signature of the systematic errors of the astrometric reference frame. Additionally, think about a better prior for the orbital elements. Refer to the brilliant paper by Kosmo O'Neil et al. (2019)[5] about avoiding biases that can be introduced when flat priors are used for orbital elements.

[4]https://emcee.readthedocs.io/en/stable/tutorials/line/
[5]https://ui.adsabs.harvard.edu/abs/2019AJ....158....4O/abstract