Statistics Homework

This an individual homework and is due at 11:59 pm on Friday, May 30. In addition to plotting, there are a number of built-in Python functions that you may find helpful, so feel free to submit this assignment to Canvas in notebook form.

1 Playing with Probability Distributions

We have introduced a number of probability distributions in class and described how they relate to physical processes. This question will have you derive quantities and manipulate probability distribution functions (PDFs) to gain insight into how they work

a) You are an exoplanet astronomer who has just developed new Starshade^{TM} technology that allows you to directly image exoplanet systems. Your goal is to measure how close each planet is to its host star to determine whether it is in the "Habitable Zone". You expect $\sim$1% of all exoplanets to fall within this special spatial region based on previous studies of other systems with different techniques. After directly observing 44 exoplanets using your method, you determine that 3 are within the Habitable Zone.

   i) What distribution governs the probability that some number of these 44 exoplanets will be found in the Habitable Zone? Justify your answer.

   ii) What is the expected probability that you would observe 3 or more systems in the habitable zone? Based on this, does your study warrant publication and further investigation?

b) You are an astroparticle physicist studying the interaction of high energy cosmic rays within our atmosphere. To identify the rate of incident cosmic rays, you have launched a balloon experiment with a large detector (essentially a "bucket" to collect particles with a precise clock to measure the time between events). When your balloon lands, you go to collect your data, only to find that the cosmic rays have fried your clock! So much for accurate timing of specific events. Based on housekeeping data you noted down from watching the balloon launch and land, you can at least pin down the total flight time of the balloon to 48 ± 3 minutes. During the flight, you collected 2465 particle events.

   i) What is the Poisson error on the total number of events recorded?

ii) What is the estimated number of cosmic ray events per second and its associated measurement error?

iii) You go to the literature and find a previous experiment that probed the rate of cosmic rays during a Solar flare. This experiment measured the rate during these flares to be 55.3 ± 0.2 events per minute. Is your experiment consistent with taking place during a period of enhanced solar activity? Justify your answer.

2 Archival Observations of a Star

You are taking deep observations of a star over the course of one week of observing time. Before arriving at the telescope, you study the published results for this star and note 30 measurements of the flux (in arbitrary units) that have been taken over the past three years. You can find their measurements with one sigma (68%) confidence limits (i.e., uncertainties or errors) in the FITS catalog file `prob2.cat` on Canvas. You have taken note of when the observations were taken as the number of days after the first observation.

Calculate the following quantities for these flux measurements. You may do this by hand or via computational methods/functions, so long as you detail the choice of functions and parameters:

a) unweighted mean

b) median

c) error on the unweighted mean

d) sample variance about the unweighted mean

e) weighted mean

f) error on the weighted mean

3 Modelling Previous Observations

a) Plot the flux measurements from the previous problem (with descriptive axes!) as:

i) A histogram of the flux values.

ii) A light curve, which plots the flux as a function of time, with confidence limits.

b) Assuming a model with constant flux with respect to time, and using a χ^2 fit, compute the best fit value of the constant with the PYTHON model fitter (see computational supplement).

i) What is the best fit χ^2 value?

ii) What is the probability that the χ^2 value would be randomly greater than this value for our constant model?

iii) Is this an acceptable fit?

iv) What is the best-fit constant value and what are the 68% and the 95% confidence limits on that value?

v) In part a) you plotted the data with one sigma (68%) confidence limits, which delineate a range which is believed to contain the true answer 68% of the time for a repeated experiment. Do the plotted confidence limits match this expectation?

c) How does the best-fit constant value compare to what you calculated in problem 2?

4 Follow-up Observations of Your Star

You take new measurements of the star from problem 2 on shorter timescales than previous measurements. The flux values derived from your images are listed in the FITS catalog file `prob4.cat` available on Canvas. However, things didn't always work out as well as you hoped. You took careful notes of every observation, and made sure to note when possibly detrimental events happened, which are noted in the other columns of the catalog (0 – condition did not occur; 1 – condition occurred). Note that the time scale of your observations are now in hours instead of days.

a) Plot the light curve for your observations with uncertainties. Is a constant model an acceptable fit to the data? If not, do any data points warrant removal? If so, exclude those data points and recompute the constant model quality fit.

b) After examining the light curve, your observing team suggests fitting a constant plus linear model (flux $= A \times t + B$) to the cleaned data remaining from part a). What are the best-fit parameters?

c) Is the new model (constant+linear) an acceptable fit to the data? If so, what are the 68% (1σ) confidence limits on the fit parameters?

d) Compare the χ^2 values of the constant and constant+linear fits. Is the use of the constant+linear model justified? Quantify your answer in terms of the probability that the improvement in χ^2 value by the addition of the extra term is due to random fluctuations.

5 Observations with unknown errors

A collaborator of yours reaches out to you after having observed what he believes to be a new type of variable star. He believes this star is characterized by a slow, steady increase in brightness, followed by a sharp drop, and thought he was observing during this "slow-burn" increase. He swears he can see the trend in his data, but none of his models produce acceptable fits! He has sent you his measurements with their associated errors (`prob5.cat` on Canvas) to ask for your assistance.

a) Plot the light curve for your observations with uncertainties, and attempt to compute a fit of your choosing. What do you notice? Comment qualitatively and quantitatively on why your collaborator is having trouble.

b) Choose a model (function) to test against these data. Improve your collaborator's analysis by **bootstrapping** his sample and recording the best-fit model parameters for your chosen model on each iteration.

c) Plot the posterior distributions of the model parameters you have recorded. Report their 68% confidence intervals and select (and justify!) a value to use as the best-fit. Plot the best fitted model against the original data.

Statistics Homework: Computational Supplement

To complete this statistics homework we suggest you explore how to use a PYTHON library to fit various datasets with arbitrary models. This may also be useful your Final Projects. You are not required to use the package described below to produce fits to your data, but make sure you select a package that handles minimization and confidence intervals properly!

Fitting Models

In class, we were introduced to χ^2 fits. To implement these fits computationally, we introduce the LMFIT package, a least-squares minimization tool that can fit arbitrary minimization functions.

To run the code below, make sure you have `lmfit` installed so you can import LMFIT functionality:

```
pip install lmfit
```

Creating a Model and Fitting

To fit a model with LMFIT to a particular set of data, we need to define a residual function to minimize. Say we want to fit a linear model to our data:

```
from lmfit import Minimizer, minimize, create_params, fit_report

def residual_linear(pars,xdata,ydata,uncertainty=None):
  parvals = pars.valuesdict()
  constant = pars['const']
  slope = pars['slope']

  model = constant + slope*xdata

  if uncertainty is None:
    return (model - ydata)

  return (model - ydata) / uncertainty
```

- `xdata` is a NUMPY array representing independent data, such as a catalog column.

- `ydata` is a NUMPY array representing the dependent data, always a 1D array.
- `uncertainty` is a NUMPY array representing the one sigma confidence limits of the dependent data, always a 1D array. It is not required, in which case the fit will technically not be χ^2, but simply least-squares minimization

Additionally, LMFIT has a number of built-in models you may find useful, and you are welcome to play around with those, as there is slightly different notation.

Once a model is selected, you will need to create a set of initial parameters:

```
fit_params_linear = create_params(const=some_guess,slope=some_guess2)
```

Then, you create a `Minimizer` object and conduct the fit (using whatever variable arrays play the roles of `xdata`, `ydata`, and `uncertainty`):

```
mini_linear = Minimizer(residual_linear, fit_params_linear, fcn_args=(xdata,),
                fcn_kws={'ydata': ydata,'uncertainty': uncertainty})
out_linear = mini_linear.minimize()
```

Obtaining Fit Parameters and Evaluating the Quality of a Fit

The final result of calling `minimize()` is an object containing a plethora of information about the fit parameters and fit itself.

To obtain best-fit parameters, you can use the name of the parameter to obtain the best-fit value, e.g.:

```
out_linear.params['slope'].value
```

In the case of a single-parameter fit, the 1-σ (68%) error can be obtained via

```
out_constant.params['constant'].stderr
```

To determine whether the fit is acceptable (and see a description of the parameters and **rough** errors), you can access the `fit_report`:

```
print(fit_report(out_linear))
```

which should produce something similar to:

```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 7
    # data points      = 100
    # variables        = 2
    chi-square         = 95.7001480
    reduced chi-square = 0.97653212
    Akaike info crit   = -0.39503410
    Bayesian info crit = 4.81530627
[[Variables]]
```

```
    const:  5.06089490 +/- 0.06078775 (1.20%) (init = 5.028229)
    slope: -0.00877874 +/- 0.01049643 (119.57%) (init = -0.04086189)
[[Correlations]] (unreported correlations are < 0.100)
    C(const, slope) = -0.8650
```

from which you can read off χ^2 and χ^2_ν values!

Calculating Confidence Limits

The errors reported in the `fit_report` use covariances estimated during the fit. These approximations are useful, but do not always contain a full understanding of the parameter space and fit statistics. Following the procedure discussed in class, we need to fix the value whose error we want to describe, and vary the other parameters until we obtain a set difference in the χ^2 value of the fit. Fortunately, LMFIT has the power to do that!

Using the `Minimizer()` object and output of the fit previously created, you can use the built-in `conf_interval` function:

```
from lmfit import conf_interval, report_ci

ci_linear = conf_interval(mini_linear, out_linear, sigmas=[1,2,3])
report_ci(ci_linear)
```