BayesianChondruleCode

July 12, 2023

[47]:
```
import numpy as np
from scipy.stats import norm
import scipy.integrate as integrate
import matplotlib.pyplot as plt
```

[48]:
```
def Brec_fn(Btype,Bamb,x,PSE):
    '''
    type:
        0 = Constant field, non-spinning chondrule
        1 = Constant field, spinning condrule
        2 = Sinusoidally varying field, non-spinning chondrule
        3 = Sinusoidally varying field, spinning chondrule
    '''
    if Btype == 0:
        #here PSE is the paleointensity standard error
        return np.exp(-(x-Bamb)**2 /(2*(PSE)**2))/(np.sqrt(2*np.pi)*PSE)

    elif Btype == 1:
        return 1/Bamb*np.ones(len(x))

    elif Btype == 2:
        # not defined at 0 so get as close as possible without introducing␣
 ↪other artifacts
        #n = np.minimum((x/Bamb)**2,np.ones(x.shape)*(1-1e-4))
        return 2/(np.pi*Bamb) * 1/np.sqrt(1-(x/Bamb)**2)

    elif Btype == 3:
        # arctanh not defined at 1 so get as close as possible
        n = np.maximum((x/Bamb)**2,np.ones(x.shape)*1e-9)
        return 2/(np.pi*Bamb) * np.arctanh(np.sqrt(1-n))

    else:
        raise NameError("Type not recognized")
```

[49]:
```
def likelihood_table(chond_pints, chond_sigmas, Btype=0,
                     Bamb_min=10, Bamb_max=250, Bamb_int=2, dx=0.1, PSE=False):
    '''
```

```
    Produces a table of likelihoods for a chosen Brec function and the given␣
↪data.

    chond_pints: (N,) np array of chondrule paleointensities
    chond_sigmas: (N,) np array of 1-sigma uncertainties in chondrule␣
↪paleointensities
    Brec: function describing recorded field given as type value for Brec()
        [0] = Constant field, non-spinning chondrule
         1 = Constant field, spinning condrule
         2 = Sinusoidally varying field, non-spinning chondrule
         3 = Sinusoidally varying field, spinning chondrule
    Bamb_min: Minimum ambient field to consider in microtesla [10]
    Bamb_max: Maximum ambient field to consider in microtesla [250]
    Bamb_int: Ambient field step size to use in calculations in microtesla [2]
    PSE: Paleointensity standard error.  This is used for the constant ambient␣
↪field, no-spin case to approximate the uncertainty on the true value. By␣
↪default, the function automatically computes the standard error of the␣
↪dataset. One can manually enter a number, for example a small one, to␣
↪approximate a Dirac delta function (e.g., true single ambient field).
    '''

    N = len(chond_pints)
    if not PSE:
        PSE = np.std(chond_pints)/np.sqrt(len(chond_pints))

    # loop over Bamb
    Bamb_list = np.arange(Bamb_min,Bamb_max+Bamb_int,Bamb_int)

    likelist = np.zeros((N,len(Bamb_list)))

    for i,Bamb in enumerate(Bamb_list):
        x = np.arange(0,Bamb+dx,dx)

        # extends range of integration to cover full Bamb range
        if Btype == 0:
            x = np.arange(0,Bamb+5*PSE+dx,dx)

        # deal with asymptote in sinusoidally varying, non-spinning case
        if Btype == 2:
            x = np.concatenate([x[:-1],np.linspace(x[-2],x[-1],100)[:-2]])

        # Brec
        Brec_x = np.tile(Brec_fn(Btype,Bamb,x,PSE),(N,1)).T

        # PDF
        x_t = np.tile(x,(N,1)).T
```

```
        PDF_x = norm.pdf(x_t, np.tile(chond_pints,(len(x),1)),
                         np.tile(chond_sigmas,(len(x),1)))

        likelist[:,i] = integrate.
 ↪cumtrapz(Brec_x*PDF_x,x_t,axis=0,initial=0)[-1]

    # assemble output
    likelihood = {}
    likelihood['table'] = likelist.T
    likelihood['total'] = np.nansum(np.nanprod(likelist.T,axis=1))*Bamb_int
    likelihood['normalized'] =np.nanprod(likelist.T,axis=1)*Bamb_int/
 ↪likelihood['total']
    likelihood['Bamb'] = Bamb_list

    return likelihood, Brec_x, PDF_x
```

0.1 Example 1: Find posterior likelihood function and total likelihood for single model

[50]:
```
    '''
    Example run with Semarkona (Fu et al. 2014) data. See manuscript text for␣
 ↪additional explanation.
    '''

chond_pints = np.array([37.0, 40.0, 24.5, 11.5, 21.0, 0.0, 0.0])

chond_sigmas = np.array([11.0, 13.5, 8.7, 4.7, 5.5, 8.7, 8.7])

# Btype = 3 corresponds to sinusoidally varying field, spinning chondrule
# It is important to choose Bamb_min and Bamb_max so that the >>0 portions of␣
 ↪the distribution are fully covered
# In other words, the posterior PDF at Bamb_min and Bamb_max should ~0;␣
 ↪otherwise the 'total' variable will not be representative
# Choose the Bamb_int variable as a balance between resolution and computation␣
 ↪time


test1 ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=3,
                                  Bamb_min=10., Bamb_max=250., Bamb_int=2.)
```

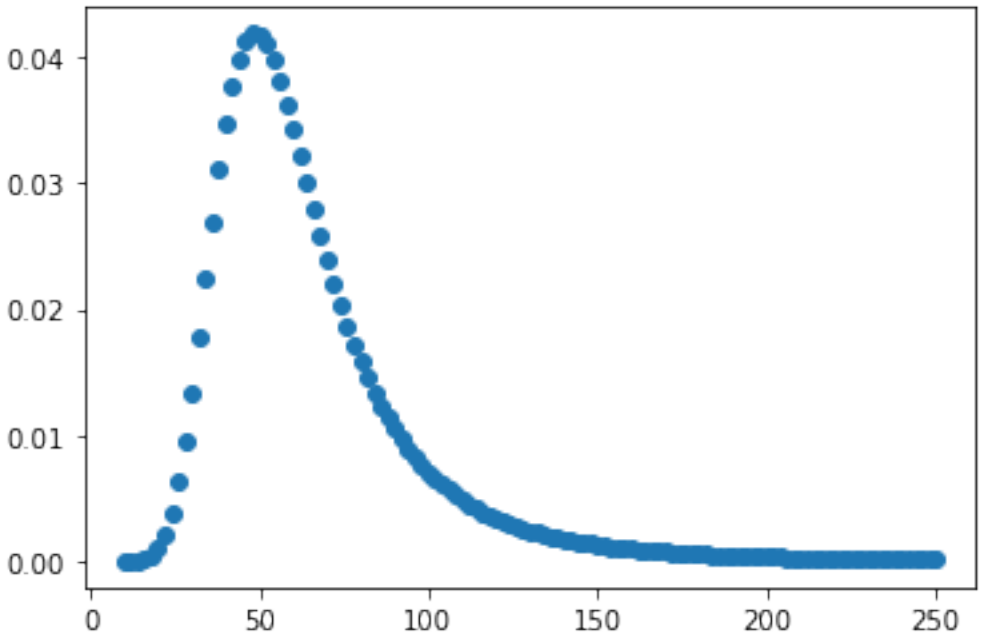
[51]:
```
# Explanation of output variables:

# test1 contains normalized posterior distributions of paleointensity Bamb:
#test1['normalized']
```

```
# test1 constains an array of Bamb values computed:

#test1['Bamb']
# test1 contains the total (marginal) likelihood of the utilized model given␣
 ↪the dataset:
test1['total']

#Brec and PDF are used internally by the likelihood_table function and should␣
 ↪not be commonly needed.
#Brec contains arrays of Brec (recorded field) values for the set of specified␣
 ↪Bamb (true ambient field)
#PDF contains arrays of normalized probabilities for each chondrule data point␣
 ↪at a given Brec
```

[51]: 1.702289177451611e-11

```
[52]: # Plot the posterior probability density function for the paleointensity

plt.scatter(test1['Bamb'],test1['normalized'])
```

[52]: <matplotlib.collections.PathCollection at 0x7fa098c7e2b0>



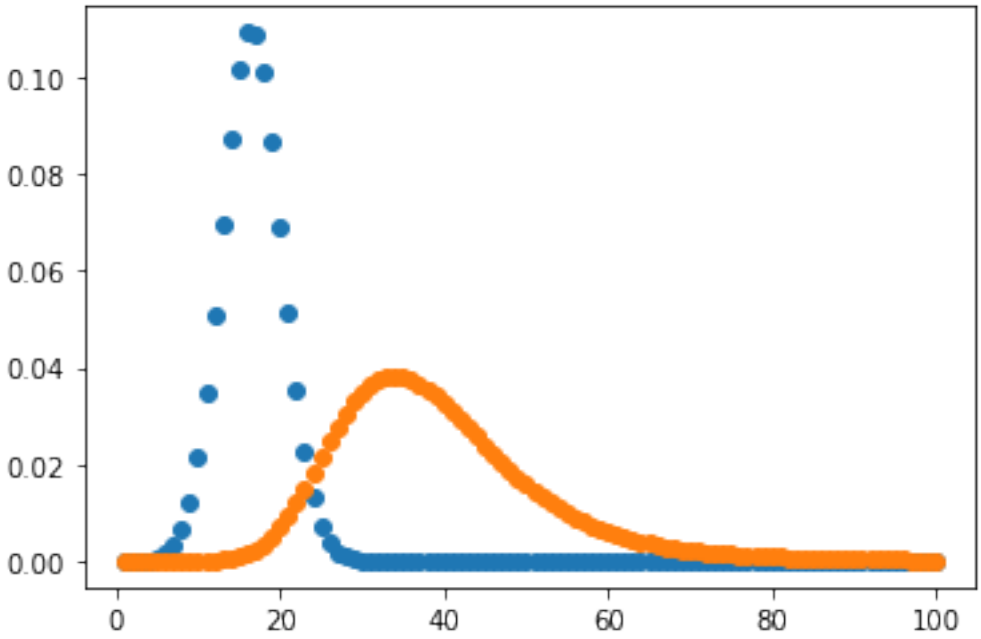
0.2 Example 2: Model selection, spinning vs non-spinning chondrule in constant ambient field

[53]:
```
# Comparison of two models with constant ambient field

chond_pints = np.array([37.0, 40.0, 24.5, 11.5, 21.0, 0.0, 0.0])

chond_sigmas = np.array([11.0, 13.5, 8.7, 4.7, 5.5, 8.7, 8.7])

# run for constant field, no chondrule spin
nospin ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=0,
                                   Bamb_min=1., Bamb_max=100., Bamb_int=1)

# version with artificially small "standard error" on constant Bamb to simulate␣
 ↪Dirac delta
#nospin ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=0,
#                                    Bamb_min=1., Bamb_max=100., Bamb_int=1,␣
 ↪PSE=0.1)

# run for constant field, yes chondrule spin
yesspin ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=1,
                                   Bamb_min=1., Bamb_max=100., Bamb_int=1)

print('Bayes Factor in favor of spin')
bayesfactor=yesspin['total']/nospin['total']
print(bayesfactor)
print('Probability of spin')
print(1-(1+bayesfactor)**(-1))
```

```
Bayes Factor in favor of spin
3.938989835657311
Probability of spin
0.7975294476655035
```

[54]:
```
plt.scatter(nospin['Bamb'],nospin['normalized'])
plt.scatter(yesspin['Bamb'],yesspin['normalized'])
```

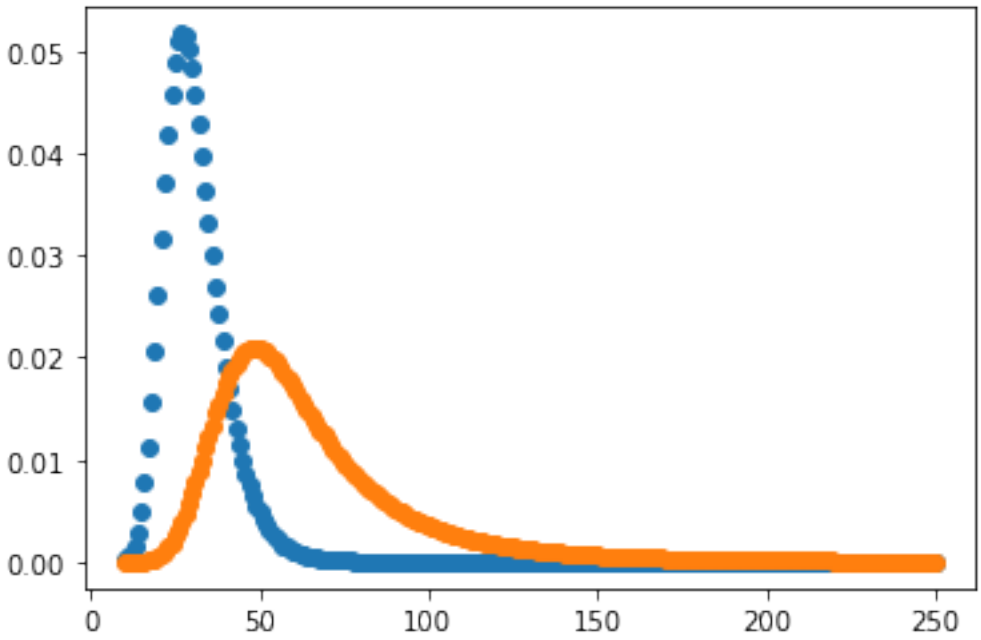
[54]:
```
<matplotlib.collections.PathCollection at 0x7fa078a843a0>
```



0.3 Example 3: Model selection, spinning vs non-spinning chondrule in sinusoidal ambient field

[55]:
```
# Comparison of two models with sinusoidal ambient field

chond_pints = np.array([37.0, 40.0, 24.5, 11.5, 21.0, 0.0, 0.0])

chond_sigmas = np.array([11.0, 13.5, 8.7, 4.7, 5.5, 8.7, 8.7])

# run for constant field, no chondrule spin
nospin ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=2,
                                   Bamb_min=10., Bamb_max=250., Bamb_int=1.)
# run for constant field, yes chondrule spin
yesspin ,Brec,PDF= likelihood_table(chond_pints, chond_sigmas, Btype=3,
                                   Bamb_min=10., Bamb_max=250., Bamb_int=1.)

print('Bayes Factor in favor of spin')
bayesfactor=yesspin['total']/nospin['total']
print(bayesfactor)
print('Probability of spin')
print(1-(1+bayesfactor)**(-1))
```

```
Bayes Factor in favor of spin
1.8418803359682752
Probability of spin
0.6481203000198517
```

[56]:
```
plt.scatter(nospin['Bamb'],nospin['normalized'])
plt.scatter(yesspin['Bamb'],yesspin['normalized'])
```

[56]: <matplotlib.collections.PathCollection at 0x7fa0b9f49700>