Bayesian Regression and Classification

Nikolas Tsigkas

April 2021

1 Bayesian Polynomial Regression on Linköping Temperature data

The given dataset contains daily temperature measurements in Linköping over one year. The aim is to fit a second degree polynomial to the data. More formally, the model can be formulated as:

$$T_i = \beta_0 + \beta_1 t_i + \beta_2 t_i^2 + \epsilon_i \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2) \qquad i \in [1, .., 365]$$

Where T_i is the temperature at time i measured in $^\circ C$ and $t_i = \frac{i}{365}$ is the time of the year. The used prior for the β_i parameters, is the conjugate prior, i.e.:

$$\beta \sim N(\mu_{\mathbf{0}}, \sigma_0^2 \Omega_0^{-1})$$
$$\sigma^2 \sim \text{Scale-Inv-}\chi^2(\nu_0, \sigma_0^2)$$

The first step is to choose suitable values for the hyperparameters μ (the mean vector), σ_0 (the scale parameter for the variance), ν_0 (the degrees of freedom for the variance) and Ω_0 (the precision matrix for the normal multivariate normal distribution). This can be done by drawing values from the prior of β and plotting the implied regression curves to see whether they agree with the data. After some iterations, the chosen suitable values were:

$$\mu = \begin{bmatrix} -15 \\ 120 \\ -110 \end{bmatrix}, \quad \sigma_0 = 1, \quad \nu_0 = 8, \quad \Omega_0 = \frac{1}{20}\mathbf{I}_3$$

The resulting plot for these prior values is shown below in figure 1, together with the observed temperature data, where one can see that the the curves cover the span of the data quite well.

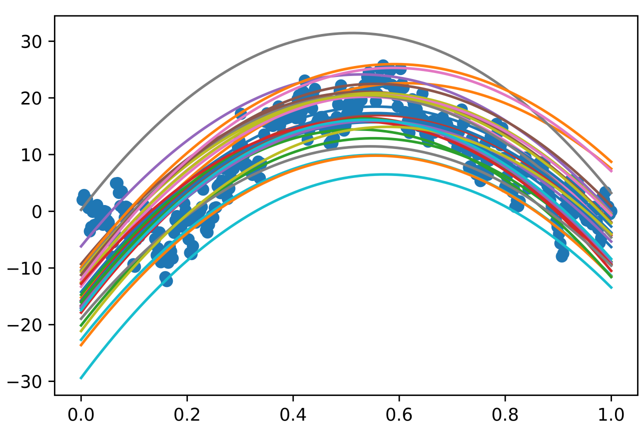

Figure 1: Regression simulated from the conjugate prior of β

Since the conjugate prior was chosen, the posterior can be evaluated analytically as:

$$\beta|\sigma, \mathbf{y}, \mathbf{X} \sim N(\mu_n, \sigma^2\Omega_n^{-1})$$
$$\sigma^2 \sim \text{Scale-Inv-}\chi^2(\nu_n, \sigma_n^2)$$

Where:

$$\mu_n = (\mathbf{X}^T\mathbf{X} + \Omega_0)^{-1}(\mathbf{X}^T\mathbf{X}\hat{\beta} + \Omega_0\mu_0)$$
$$\Omega_n = \mathbf{X}^T\mathbf{X} + \Omega_0$$
$$\nu_n = \nu_0 + n$$
$$\sigma_n^2 = \tfrac{1}{\nu_n}(\mathbf{y}^T\mathbf{y} + \mu_0^T\Omega_0\mu_0 + \mu_n^T\Omega_n\mu_n)$$

And $\hat{\beta} = \mathbf{X}^\dagger\mathbf{y}$ is the ordinary least squares estimate[1].

From these expressions, one can now draw posterior samples to get a posterior model given the data. The histograms for each individual parameter are shown below in figure 2 after 10 000 simulated draws.

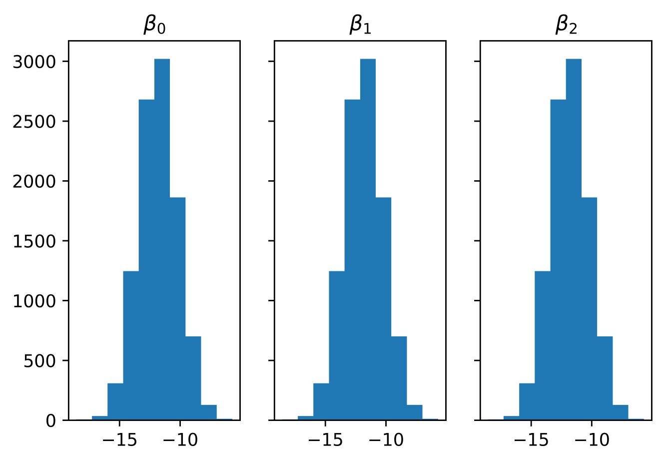

Figure 2: Histogram over the regression parameters generated from 10 000 simulated draws

From these 10 000 draws, 10 000 polynomial curves can be drawn as: $y^{(i)} = \mathbf{x}^T\beta^{(i)} + \sigma_i\xi_i$, where $\xi_i \overset{iid}{\sim} N(0,1)$, $i = 1, .., 10000$, and $\mathbf{x} = [1, t, t^2]$ is the time values. After computing these implied temperature values from these regression curves for each day of the year, the 2.5th, 50th 97.5th percentiles where evaluated for each day. These results are shown in figure 3.

The interval between the two extreme percentiles capture the uncertainty quite well, with most of the values lying between the two. The model is however failing to capture some of the data around March and November.

From the simulations, one can also explore other features in the data, for which there might be no easy to derive probability density. One such example is the time of the year with the highest expected temperature $\tilde{t}$. From the model, this quantity can be found by evaluating the derivative of the temperature with regard to time, to find the unique stationary point which will yield the maximum value of the polynomial.

$$\frac{dT}{dt} = \beta_1 + 2\beta_2\tilde{t} = 0 \implies \quad \tilde{t} = -\frac{\beta_1}{\beta_2}$$

This random variable follows no known distribution (although the theory of quotient distributions may give some reasonable guidance for approximations), but can be approximated via simulation. Using the same random draws as before, the histogram in figure 4 was obtained.

[1] $\dagger$ denotes the pseudo-inverse, i.e. $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$

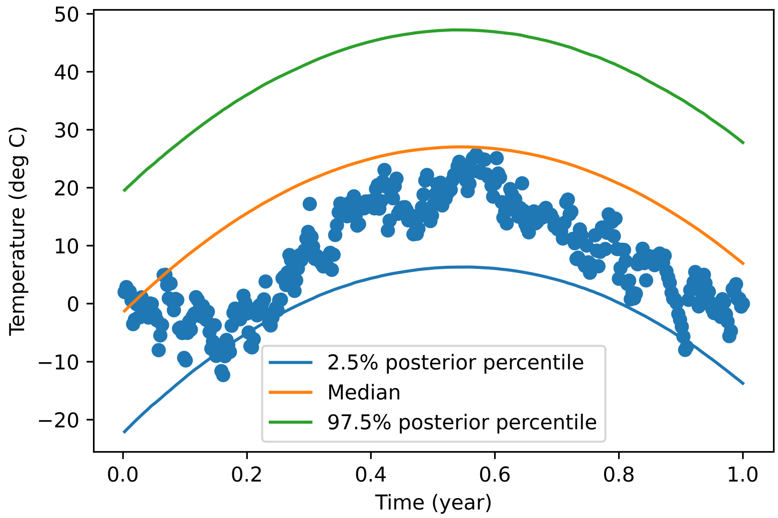

Figure 3: Posterior regression for the 2.5th, 50th and 97.5th percentiles

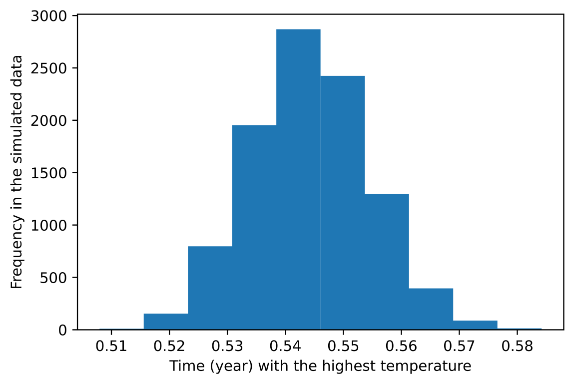

Figure 4: Histogram of the distribution of the time with the highest expected temperature

One can see that the time with the expected highest temperature is very centered around the values $0.5 - 0.6$, corresponding to calendar dates between the 2nd of July and 7th of August.

This method of polynomial regression can be generalized to fit higher order polynomials, which could have been suitable given what was displayed in figure 3, where some of the data at the beginning of the year was not captured by the regression intervals. However, with such methods, one runs the risk of overfitting the model to the data which yields a model that does not generalize well. For such methods, regularization techniques such as *Lasso* or *Tikhonov Regularization* (also called *Ridge Regression*) may be suitable. These regularization methods are equivalent, in the Bayesian sense, to these prior distributions:

$$\beta_{i,Lasso}|\sigma \overset{iid}{\sim} \text{Laplace}(0, \frac{\sigma^2}{\lambda}) \quad \beta_{i,Ridge}|\sigma \overset{iid}{\sim} N(0, \frac{\sigma^2}{\lambda})$$

Where $\lambda \geq 0$ is the regularization hyperparameter, which can be interpreted as a penalty term that penalizes solutions whose β-terms have too high of a magnitude. Thus, increasing the value of λ will result in a more "smooth" polynomial. Too high values could however result in a model that underfits the data. One way of selecting the magnitude of λ is by treating it as a random variable, and letting it follow a prior distribution, such as the scaled inverse chi squared distribution.

```
""" Useful packages """
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats as st

np.random.seed(1234) # set seed for reproducability

""" Helper functions for drawing samples and calculating posterior parameters
    """
# Generates num random draws from a scale-inverse-chi2 distribution
def draw_chi2inv(df, scale, num):
    x = st.chi2.rvs(df-1, scale, size = num)
    sigma_sq = (df-1)*scale/x
    return sigma_sq

# Generates multivariate normal random draws for different sigmas
def draw_mvnorm(mu, Omega_inv, sigma_sq):
    beta = np.zeros((len(sigma_sq),3))
    for i, sig2 in enumerate(sigma_sq):
        beta[i] = st.multivariate_normal.rvs(mean = mu, cov=sig2*Omega_inv,
            size = 1)
    return np.transpose(beta)

# Function that generates posterior parameters for the posterior
def post_params(X, y, mu_0, Omega_0, sigma_0):
    n = len(y)

    X_dagger =np.linalg.pinv(X)
    XTX = X.T@X
    beta_hat = X_dagger @ y

    mu_n = np.linalg.inv(Omega_0 + XTX) @ XTX@beta_hat + Omega_0@mu_0
    Omega_n = XTX + Omega_0
    nu_n = nu_0 + n

    yTy = y.T@y
    mu_Omega_mu_0 = mu_0.T @ Omega_0 @ mu_0
    mu_Omega_mu_n = mu_n.T @ Omega_0 @ mu_n

    sigma_n2 = (nu_0*sigma_0**2 + yTy + mu_Omega_mu_0 + mu_Omega_mu_n )*1/nu_n

    return [mu_n, Omega_n, nu_n, sigma_n2]

# Returns num random draws from the posterior of beta and sigma^2,
# given the posterior parameters as input
def post_draws(mu_n, Omega_n, nu_n, sigma_n2, num):
    sigma_2     = draw_chi2inv(nu_n, sigma_n2, num)
    beta_post   = draw_mvnorm(mu_n, np.linalg.inv(Omega_n),sigma_2)
    return [beta_post, sigma_2]

""" Import data"""
data = pd.read_csv("TempLinkoping.txt", sep = "\t+")
```

```

Y = data.temp.values
# Data matrix consisting of 1, temp, temp^2
X = np.column_stack(((np.ones_like(data.time.values), data.time.values, data.
    time.values**2)))

# Plot the data
plt.scatter(data.time.values,Y)
plt.xlabel("Time (year)")
plt.ylabel("Temperature (deg C)")
plt.show

# Prior parameters
mu_0 = np.array([-15,120,-110])
Omega_0 = 0.05*np.eye(3)
Omega_0_inv  = np.linalg.inv(Omega_0)
nu_0 = 8
sigma_0 = 1

# Draw 50 draws from the prior of beta
sigma_sq = draw_chi2inv(nu_0-1, sigma_0**2, 30)
beta_prior = draw_mvnorm(mu_0, Omega_0_inv, sigma_sq)

# Check visually if the priors generate a reasonable regression curve
zero_to_one = np.linspace(0,1,100)
X_test = np.column_stack((np.ones(100), zero_to_one, zero_to_one**2))

# By plotting the data
# together with the simulated regression curves from the prior
Y_prior = X_test@beta_prior
plt.plot(zero_to_one, Y_prior)
plt.scatter(data.time.values,Y)
plt.show
# Change the prior parameters until the results look reasonable

[mu_n, Omega_n, nu_n, sigma_n2] = post_params(X, Y, mu_0, Omega_0, sigma_0)
[beta_post, sigma2_post] = post_draws(mu_n, Omega_n, nu_n, sigma_n2, 10000)

# Histograms of the beta coefficients
fig, axs = plt.subplots(1,3, sharey = True)

for i in range(3):
    axs[i].hist(beta_post[0,:])

axs[0].set_title(r"$\beta_0$")
axs[1].set_title(r"$\beta_1$")
axs[2].set_title(r"$\beta_2$")

post_temp = np.zeros((365,10000))
for i in range(10000):
    post_temp[:,i] = X@beta_post[:,i] + np.sqrt(sigma2_post[i])*st.norm.rvs(1)

# Calculate the 2.5, 50 and 97.5 percentiles of the temperatures of each day
post_0025 = np.percentile(post_temp, 2.5,  axis=1)
```

```
post_0500 = np.percentile(post_temp, 50,   axis=1)
post_0975 = np.percentile(post_temp, 97.5, axis=1)

# Plot implied regression curves
plt.plot(X[:,1], post_0025)
plt.plot(X[:,1], post_0500)
plt.plot(X[:,1], post_0975)

plt.legend(["2.5% posterior percentile","Median", "97.5% posterior percentile"
    ])
plt.xlabel("Time (year)")
plt.ylabel("Temperature (deg C)")

plt.scatter(data.time.values,Y)
plt.show
plt.savefig("temp_data_posterior_curves.png", dpi = 1000)

# Find the simulated distribution of the time with maximal temperature
# from the posterior distribution of beta
time_max = -0.5*beta_post[1]/beta_post[2]

# Plot the distribution of time_max
plt.hist(time_max)
plt.xlabel("Time (year) with the highest temperature")
plt.ylabel("Frequency in the simulated data")
```

2 Logistic Regression on Women Worker Data

The given dataset contains 7 explanatory variables (the income of their husband, their years of education, their years of experience and the square of their years of experience, their age, and the number of small and large children respectively) on 200 women, and whether the women work or not. One way to predict whether a woman works or not, given their features, is by fitting a logistic regression model to the data.

Formally, the model is:

$$P[y_i = 1|\mathbf{x}_i] = \frac{1}{1 + e^{-\mathbf{x}_i^T\beta}} = \Lambda(\mathbf{x}_i^T\beta) \quad \mathbf{x}_i, \beta \in \mathbb{R}^8$$

Where $y_i = 1$ represents a working woman with features $\mathbf{x}_i$ and $\Lambda()$ is the cdf of the logistic distribution.

This model comes with the difficulty of choosing a suitable prior for the β-vector, as the posterior will not come in the form of a nice analytical expression from a known density kernel. However, by using a second order Taylor expansion of the log-posterior around its mode, an approximate posterior with a nice form can be derived.

$$\ln p(\beta|y) \approx \ln p(\tilde{\beta}|y) + (\beta - \tilde{\beta})^T \nabla_\beta(\ln p(\tilde{\beta}|y)) + \frac{1}{2}(\beta - \tilde{\beta})^T H_{\ln p}(\tilde{\beta})(\beta - \tilde{\beta})$$

From the definition of the mode, the first order term in the expression vanishes, as the function is maximized at this point. Using this, the above equation can be solved for the posterior density, resulting in.

$$\ln p(\beta|y) \approx p(\tilde{\beta}|y) \exp\left[\frac{1}{2}(\beta - \tilde{\beta})^T H_{\ln p}(\tilde{\beta})(\beta - \tilde{\beta})\right] \propto \exp\left[\frac{1}{2}(\beta - \tilde{\beta})^T H_{\ln p}(\tilde{\beta})(\beta - \tilde{\beta})\right]$$

From this expression, it can be seen that the posterior is approximately proportional to the multivariate Gaussian kernel, with mean vector equal to the mode, and covariace matrix equal to the negative inverse hessian of the log-posterior (which can be interpreted as the observed information) i.e.:

$$\beta|y \overset{appr.}{\sim} N(\tilde{\beta}, -H_{\ln p}^{-1}(\tilde{\beta}))$$

The parameters $\tilde{\beta}$ and $-H_{\ln p}^{-1}(\tilde{\beta})$ can be evaluated numerically, by running a suitable optimization algorithm on the problem:

$$\max_\beta \ln p(\beta|y) \Longleftrightarrow \max_\beta \ln \sum_1^n p(y_i|\beta) + \ln p(\beta)$$

The first term comes from the logistic function defined above, and the second from the prior. The used prior for was $\beta \sim N(0, 100I)$. As the objective of the optimization was to get estimates of both the mean and covariance matrix of the approximate posterior, a quasi-Newton optimization algorithm is suitable, as it uses an estimator for the hessian to find stationary points. The Broyden-Fletcher-Goldfarb-Shanno algorithm provided the following estimators:

$$\tilde{\beta} = \begin{bmatrix} 0.628 & -0.012 & 0.18 & 0.168 & -0.145 & -0.082 & -1.359 & -0.0247 \end{bmatrix}^T$$

$$-H_{\ln p}^{-1}(\tilde{\beta}) = \begin{bmatrix} 0.169 & 0.007 & -0.004 & 0.024 & -0.071 & -0.004 & 0.072 & -0.176 \\ 0.007 & 0.001 & -0.001 & 0.001 & -0.003 & 0 & 0.004 & -0.007 \\ -0.004 & -0.001 & 0.005 & -0.001 & 0.005 & -0.001 & -0.015 & 0.004 \\ 0.024 & 0.001 & -0.001 & 0.008 & -0.025 & -0.001 & 0.009 & -0.025 \\ -0.071 & -0.003 & 0.005 & -0.025 & 0.082 & 0.002 & -0.027 & 0.074 \\ -0.004 & 0 & -0.001 & -0.001 & 0.002 & 0 & 0.001 & 0.004 \\ 0.072 & 0.004 & -0.015 & 0.009 & -0.027 & 0.001 & 0.174 & -0.076 \\ -0.176 & -0.007 & 0.004 & -0.025 & 0.074 & 0.004 & -0.076 & 0.183 \end{bmatrix}$$

From this, one can evaluate individual features. For instance, the coefficient corresponding to the number of small children β_6 follows the distribution $\beta_6 \sim N(-1.359, 0.417)$ (the second parameter here is the standard deviation). From this, the 95% equal tail credible interval was determined as: $[-1.9599, -0.54127]$. This indicates that the number of small children that a woman has, with a high probability, correlates negatively with the probability that she works.

One can also do inference on out of sample data. For instance, a 37 year old woman with:

1. a husband whose income is 13
2. 8 years of education
3. 11 years of work experience
4. 2 small children (under the age of 7)

can be represented, in the model, as:

$$\mathbf{x} = \begin{bmatrix} 1 & 13 & 8 & 11 & 1.21 & 37 & 2 & 0 \end{bmatrix}^T$$

Given that the approximate posterior distribution of the coefficients is known, The distribution of the probability that she works can be evaluated through simulations, by drawing β-vectors, multiplying them with her data, and passing the product through the logistic cdf. The resulting histogram for this, when 10 000 draws from β was performed, is shown in figure 5.

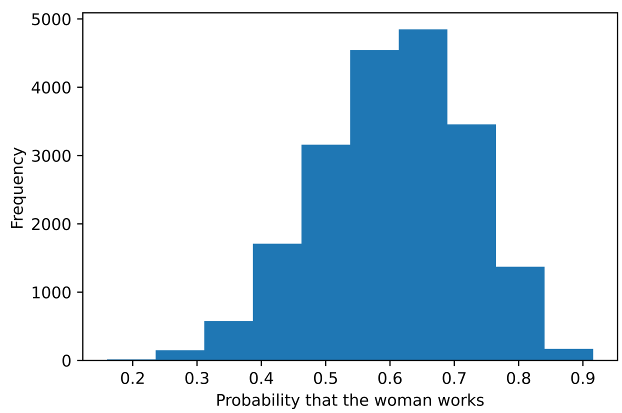

Figure 5: Histogram over the probability that the 37 year old woman works

Similarly, one can examine the distribution of the number of women working, given a population of eight women with the same features as the one mentioned above. This is equivalent of drawing samples from a $\text{Bin}(8, \Lambda(\mathbf{x}_i^T \beta))$, with $\beta \sim N(\tilde{\beta}, -H_{\ln p}^{-1}(\tilde{\beta}))$. The resulting histogram, for 10 000 draws from this distribution, is shown in figure 6.

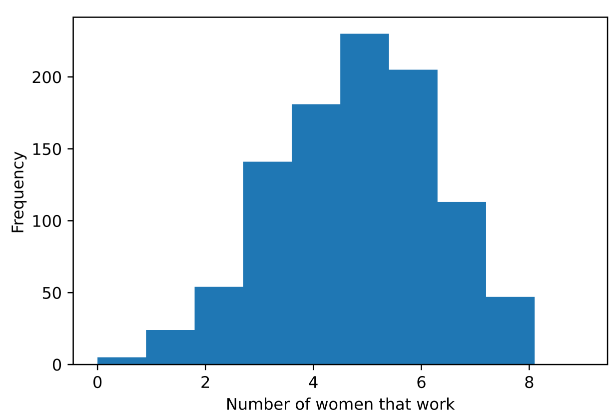

Figure 6: Histogram over the number of women out of the eight that work

```python
""" Useful packages """
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats as st
from scipy.optimize import minimize as fmin

np.random.seed(1234) # set seed for reproducability

""" Helper functions """
# Returns the log posterior of the logistic function
def log_posterior(beta, y, X, mu, Cov):
    log_prior = np.log( st.multivariate_normal.pdf(beta, mean=mu, cov=Cov) )
    regr = X@beta
    logL = np.sum( regr * y - np.log(1+np.exp(regr) ) )
    return (logL + log_prior)

# Returns the posterior distribution of the probability that a particular sample
    works
def posterior_prob(x, mu, Cov, num):
    beta = st.multivariate_normal.rvs(mean = mu, cov = Cov, size = num)
    return st.logistic.cdf( np.matmul( x , np.transpose( beta ) ) )

# Returns the posterior distribution of the number of samples that work
def posterior_num(x, mu, Cov, num):
    n = len(x)
    prob = posterior_prob(x, mu, Cov, num)
    mult_pred  = np.zeros(num)
    for i, p in enumerate(prob):
        mult_pred[i] = st.binom.rvs(n, p)
    return mult_pred

""" Import the data """
data = pd.read_csv("WomenWork.dat", sep = "    ")

# Binary response variable
y = data.Work.values

# Covariates
X = data.iloc[:, 1:].values

# Prior parameters
mu_0 = np.zeros(8)
Cov_0 = 100*np.eye(8)

# Define the log posterior as a function of beta only, for the optimization
# Also, multiply with -1, as we seek the maximum
# and the algorithm performs minimization
def obj_fun(beta):
    return -log_posterior(beta, y, X, mu_0, Cov_0)

# Find the max of the log posterior using a quasi-Newton method
# Which also returns a numerical approximation of H^-1
x0 = 1*np.ones(8) # Initial guess
opt_res = fmin(obj_fun, x0, method = "BFGS", options={'disp' : True})

# The inverse of J, no minus sign
# as the objective function already has "switched signs"
```

```
J_inv = opt_res["hess_inv"]
# The mode, i.e. the variables at optimum
post_mode = opt_res["x"]

# Extract parameters for the coefficient of "NSmallChild",
# i.e. the second to last covariate
mu_SC  = post_mode[-2]
sd_SC = J_inv[-2,-2]**0.5

eq_tail_CI_SC =  [mu_SC-st.norm.ppf(0.925)*sd_SC, mu_SC+st.norm.ppf(0.975)*sd_SC]

""" Data for a woman with:
- a husband with income 13
- 8 years of education
- 11 years of work experience
- 37 years of age
- 2 small children (under the age of 7)
"""
x_1w = np.array([1, 13, 8, 11, (11.0/10.0)**2, 37, 0, 0])

# The posterior probability that this particular woman works
post_prob = posterior_prob(x_1w, post_mode, J_inv, 20000)

plt.hist(post_prob)
plt.xlabel("Probability that the woman works")
plt.ylabel("Frequency")
plt.savefig("work_prob_hist.png", dpi = 1500)
plt.show

# The posterior distribution of the nunmber of women working
post_num = posterior_num(x_1w, post_mode, J_inv, 1000)

plt.hist(post_num, range=(0,9))
plt.xlabel("Number of women that work")
plt.ylabel("Frequency")
plt.savefig("work_num_hist.png", dpi = 1500)
plt.show
```