MATH95010 - GROUP S16

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

Bayesian Linear Regression

Authors:
Ridha Bouhiaoui (CID: 01519353)
Bahram Keshtmand (CID: 01516821)
Adrian Law (CID: 01499874)
Ismail Riahi (CID: 01379199)
William Woodward (CID: 01515192)

With special thanks to Professor Nicholas Heard

Date: June 17, 2020

Contents

1 Introduction

Bayesian linear regression is a method of linear regression that incorporates Bayesian methods to model data. In this project, we will cover the theory behind Bayesian linear regression and its applications in choosing a suitable model for describing a simulated dataset. We will then apply the Bayesian linear regression methodology to a data set.

2 The Bayesian Linear Model

2.1 The Standard Linear Model under Normal Theory Assumptions

The equation for the standard linear model is:

$$Y = X\beta + \epsilon$$

where:
Y $\in$ R^n is a vector of (observable) data
X is an (n x m) design matrix
$\beta \in$ R^m is an unknown parameter
$\epsilon \in$ R^n is a non-observable n-vector, $E(\epsilon) = 0$
Under Normal Theory assumptions (NTA), we have that $\epsilon \sim N_n(0, \sigma^2\mathbf{I}\ )$ for $\sigma^2 > 0$

It is sometimes helpful to write the standard linear model in the equivalent form:

$$Y \sim N_n(X\beta, \sigma^2\mathbf{I})$$

In the frequentist setting, β and σ^2 are treated as fixed unknown parameters, whereas using the Bayesian approach we can treat β and σ^2 as random variables, and assign them prior distributions based on our prior knowledge.[1] There are several advantages of using the Bayesian approach, these include [2],[3]:

- It takes in to account any prior information and beliefs that we have about the parameters.
- It only involves probability calculations, rather than having to maximize functions

However, there are also some disadvantages:

- Probability calculations often include complicated integrals and ones that can't be solved analytically (although numerical techniques can approximate them with accuracy)
- Prior distributions can be hard to justify, especially if we have little prior knowledge of the parameters.

2.2 Prior Distributions for β and σ^2

For the purposes of this project, we will choose the following prior distributions for β and σ^2, due to their mathematical convenience [4]:

$$\beta \mid \sigma^2 \sim N_m(0, \sigma^2\mathbf{V})$$

$$\sigma^{-2} \sim Gamma(a, b)$$

where V is a (m x m) matrix, $a \in \mathbb{R}$, $b \in \mathbb{R}$ are constants that are chosen based on our prior beliefs of β and σ^2.

2.2.1 Choice of V, a and b

If we have strong prior beliefs about β and σ^2, then we want our prior beliefs to have a strong influence on the posterior distributions $\beta \mid y$ and $\sigma^2 \mid y$. In this case, we can choose the prior distributions to have a very small variance. These are "informative priors"[5]. Similarly, we can choose the prior distributions to have large variance if we have little knowledge of the parameters, these are "diffuse parameters"[5]. Noting that $E[\sigma^{-2}] = \frac{a}{b}$ and $Var[\sigma^{-2}] = \frac{a}{b^2}$ (standard facts about a gamma distribution), if we wish to make our prior distributions "informative priors", we could choose b to be large and **V** to be diagonal with small entries. Similarly, to make our prior distributions "diffuse priors" we could choose b to be small and **V** to be diagonal with large entries.

3 Simulating Data

In this section, we aim to simulate data for a general function with noise.

$$y = f(x) + noise$$

Where $noise \sim N(0, \sigma_{noise}^2)$

To do this, we will sample 'n' points from a uniform(0,1) distribution, apply the function and then add noise to each point. [6]
Code:

```
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
from scipy.special import gamma
from scipy.special import gammaln

def simulate(f, n, sigma_noise):

     x = np.random.uniform(0,1,n)

     noise = np.random.normal(0, sigma_noise**2, n)
```

```
    y = f(x) + noise
    plt.plot(x,y, '.')


    return [x,y]
```

If we want to simulate 50 data points, using the function $f(x) = e^x$ with sigma_noise = 0.25 :

```
simulate(lambda x: np.exp(x), 50, 0.25)
```

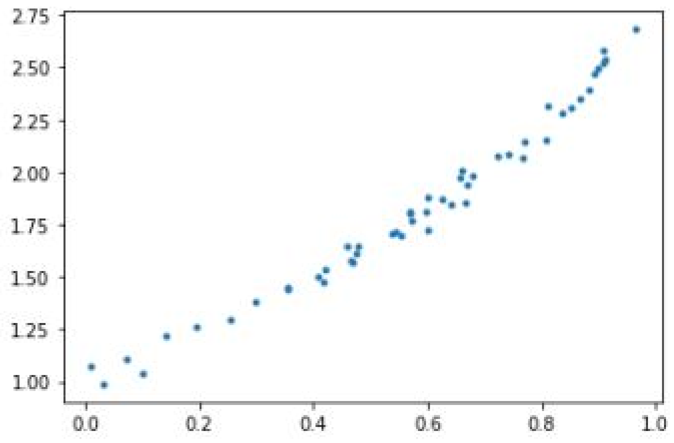


Figure 1: Graph showing 50 simulated data points

4 Design Matrix and Basis Functions

4.1 The Design Matrix

In linear regression, an observed value y_i is expressed as a linear combination of m regressors $[r_{i1}, ...r_{im}]$:

$$y_i = \beta_1 r_{i1} + ... + \beta_m r_{im} + \epsilon_i$$

Here, $[\beta_1, ..., \beta_m] = \beta$ and ϵ_i is the corresponding error term.
Thus, for all n observed values of y:

$$y = \begin{bmatrix} y_1 \\ ... \\ y_n \end{bmatrix} = \begin{bmatrix} r_{11} & ... & r_{1k} \\ ... & & ... \\ r_{n1} & ... & r_{nk} \end{bmatrix} \begin{bmatrix} \beta_1 \\ ... \\ \beta_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ ... \\ \epsilon_n \end{bmatrix}$$

This matrix of regressors is known as the design matrix.

4.2 Basis Functions

As seen in Section 4, a uniform distribution is used to generate n values of x, which in turn are used in the function $y = f(x) + noise$ to generate n values of y. To construct an n x m design matrix X, we use a set of basis functions $[\phi_0(x), \phi_1(x), ..., \phi_m(x)]$ and the simulated x values x_1 to x_n as follows:

$$X_{ij} = \phi_j(x_i)$$

Hence $y_i = \phi(x_i)^T \beta$, where $\phi(x_i) = [\phi_0(x_i), \phi_1(x_i), ..., \phi_m(x_i)]$

Linear models that use design matrices of this form are known as linear basis function models. [6]
Code for generating a design matrix X with a vector of basis functions p:

```
def designmatrix(f, n, sigma_noise , p):
    #m is the number of basis functions in p
    m= len(p)

    #simulating data from f, n and sigma_noise
    [x,y] = simulate(f, n, sigma_noise

    #constructing the design matrix from the simulated values
    X =  np.zeros([n, m])

    for i in range(n):
        for j in range(m):
            X[i,j]= p[j](x[i])

    return X
```

4.3 Types of Basis Functions

There are many types of vectors of basis functions that can be used. In this project, we will focus on three: polynomial, Gaussian and piecewise linear. Our aim is to find the set of basis functions that yields the most suitable model to describe the simulated data.

4.3.1 Polynomial Basis Functions

Polynomial basis functions are of the form $[1, x, ..., x^m]$.
Code used to return a vector of polynomial basis functions of length m+1:

```
def poly(k):
    poly = lambda x: x**k
    return poly

#input n: power of x
def polybasis(n):
    polybasis = [0]*(n+1)
    for i in range(n+1):
        polybasis[i] = poly(i)
    return polybasis
```

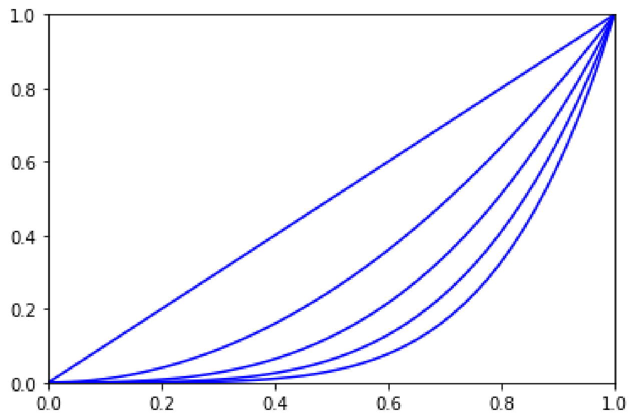


Figure 2: Polynomial basis functions up to degree 5 plotted on Python

4.3.2 Gaussian Basis Functions

Gaussian basis functions are of the form $[1, g(0, s), ..., g(m, s)]$, where

$$g(i, s) = \exp \frac{-(x - \mu_i)^2}{2s^2} \quad [6]$$

where s is the spacing, and its value depends on m. The value μ_i for g(i,s) is calculated by i x s.
Code used to return a vector of Gaussian basis functions of length m+2:

```
def gauss(i,s) :
    #mu_i is i*s
    gauss = lambda x: np.exp((-(x - (i*s))**2)/(2*((s)**2)))
    return gauss

#gaussbasis(n) gives a vector of Gaussian basis functions:
#[gauss(0,s), gauss(1,s)..., gauss(m,s)]

def gaussbasis(n):
    #to separate [0,1] x [0,1] into evenly spaced intervals
    s = 1/(n-1)
    gaussbasis = [0]*n

    for i in range(n):
        gaussbasis[i] = gauss(i,s)

    return gaussbasis
```

For example, the following is a plot of the Gaussian basis functions for n = 11:

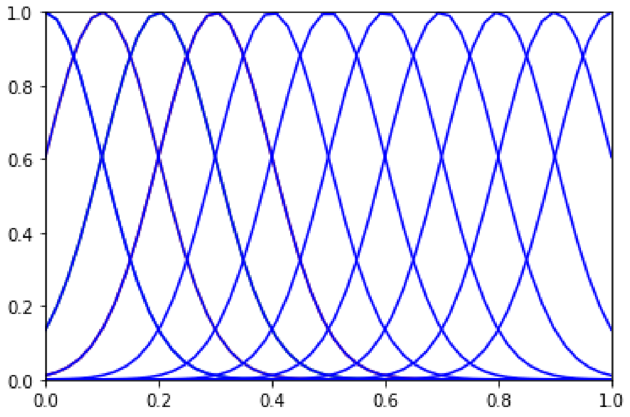


Figure 3: Gaussian basis functions plotted on Python

4.3.3 Piecewise Linear Basis Functions

Piecewise linear functions are of the form $[1, (x-t_0)_+, (x-t_1)_+, ..., (x-t_m)_+]$, where $(a)_+ = max(0, a)$ for a number a. The values $[t_0, t_1, ..., t_m]$, with $t_0 = 0$ and $t_m = 1$, are obtained by separating [0,1] into intervals of length 1/(m-1). Code used to return a vector of polynomial basis functions of length m+1:

```
def plus(a):
    return max(0,a)


def piecewise(t):
    piece = lambda x: plus(x-t)
    return piece

#we want a vector of basis functions with length k+1:
#(1, x, (x-t_1)_+,...,(x-t_(k-1))_+,(x-1)_+)
```

```
#assuming over the interval [0,1]
def piecewisebasis(n):
    #d is the distance between t_(n-1) and t_n
    d = 1/(n-1)
    piecewisebasis = [lambda x: 1]
    for i in range(n):
        t_i = i*d
        pc_ti = piecewise(t_i)
        piecewisebasis.append(pc_ti)

    return piecewisebasis
```

Below is a graphical representation of such functions:

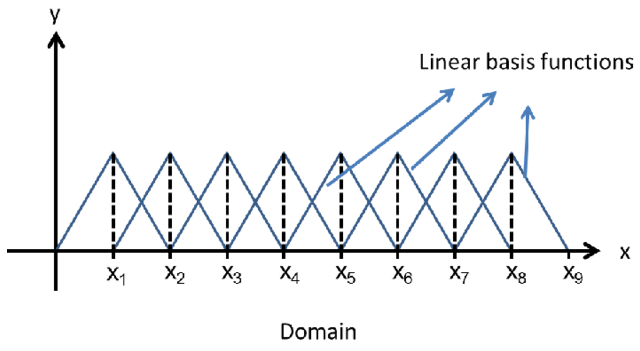


Figure 4: Image representing piecewise linear functions [7]

5 The Marginal Likelihood Density Function

Now that we have a design matrix X, we can find the marginal likelihood of $y|X$. The marginal likelihood density function will later be used to calculate the Bayes Factor, which in turn is used to compare the different types of models used in Bayesian linear regression.
As seen in section 2, y follows a multivariate normal distribution $N_n(X\beta, \sigma^2 I_n)$ in the standard linear model, and we assume that the prior distributions of $\beta|\sigma^2$ and σ^{-2} are given as $N_m(0, \sigma^2 V)$ and Gamma(a, b) respectively.
The calculation for the marginal likelihood of $y|X$ is as follows:
We can rewrite the distribution of y as a multivariate normal distribution with mean 0:

$$\beta|\sigma^2 \sim N_m(0, \sigma^2 V) \implies X\beta|\sigma^2 \sim N_n(0, \sigma^2 XVX^T) \implies y \sim N_n(0, \sigma^2(XVX^T + I_n))$$

Finding the determinant of $(XVX^T + I_n)$ by the matrix inversion lemma:

$$(XVX^T + I_n)^{-1} = I_n - X(X^TX + V^{-1})^{-1}X^T$$

$$\implies |XVX^T + I_n| = |V||X^TX + V^{-1}|$$

y can be written as a multivariate normal distribution with mean 0. Thus the density function of $y|X, \sigma^2$ is:

$$f(y|X, \sigma^2) = \frac{\exp\left(-\frac{1}{2\sigma^2}y^ty + \frac{1}{2\sigma^2}y^tX(X^TX + V^{-1})^{-1}X^Ty\right)}{(2\pi)^{n/2}\sigma^n|V|^{\frac{1}{2}}|X^TX + V^{-1}|^{\frac{1}{2}}}$$

Since σ^{-2} follows a gamma prior distribution, σ^2 follows an inverse gamma prior distribution. Now we can find $f(y|X)$ by marginalising the density function of $y|X, \sigma^2$ over the prior distribution for σ^2:

$$f(y|X) = \frac{b^a}{\Gamma(a)}\int_0^\infty \frac{\exp(-b/a)}{\sigma^{2(a-1)}} f(y|X, \sigma^2)d\sigma^{-2}$$

By comparing to the probability density function of Gamma$(a+n/2, b+\frac{1}{2}y^Ty - \frac{1}{2}y^TXm_n)^{a+n/2})$, we can simplify $f(y|X)$

$$f(y|X) = \frac{\Gamma(a+n/2)|V_n|^{\frac{1}{2}}b^a}{(2\pi)^{n/2}\Gamma(a)|V|^{\frac{1}{2}}(b + \frac{1}{2}y^Ty - \frac{1}{2}y^TXm_n)^{a+n/2}}$$

where

$$V_n = (X^TX + V^{-1})^{-1}$$

$$m_n = V_nX^Ty$$

We then find the natural logarithm of the marginal likelihood density function. The reason for doing this is that the log of the likelihood is more numerically stable, so the effect of any errors in the inputs would be lessened, allowing us to achieve more accurate results:

$$\log(f(y|X)) = \log(\Gamma(a+n/2)) + \frac{1}{2}\log|V_n| + a\log(b)$$

$$-\frac{1}{2}\log((2\pi)) - \log(\Gamma(a)) - \frac{1}{2}\log(|V|) - (a+n/2)\log(b + \frac{1}{2}y^Ty - \frac{1}{2}y^TXm_n)$$

The higher the value of the marginal likelihood for $y|X$, the more suitable the model X is for describing the simulated data.
To see how well a basis function model fits a simulated dataset of x and y values, we fix values for a, b and V, use the vector of basis functions p and the values of x to produce a design matrix X, and input y, X, a, b and V to find the log of the marginal likelihood density $y|X$. This can be used to compare the suitability of different vectors of basis functions for the dataset.
Code for finding the log of the marginal likelihood density:

```
def logmarginaly(y, X, V, a, b):
    #n = rows of X
    n = len(X)

    #defining X^t and y^t
    Xt = X.transpose()
    yt = y.transpose()

    V_n = inv(Xt.dot(X) + inv(V))
    m_n = V_n.dot(Xt.dot(y))
    detV_n = np.linalg.det(V_n)
    detV = np.linalg.det(V)

    #using the log of the given equation for f(y|X)
    logmarginal = gammaln(a+n/2) + 0.5 * np.log(detV_n) + a * np.log(b)
    - n/2 * np.log(2*np.pi) - gammaln(a) - 0.5 * np.log(detV)
    - (a+n/2) * np.log(b + 0.5 * yt.dot(y) - 0.5 * yt.dot(X.dot(m_n)))


    return logmarginal
```

6 Bayes Factor

6.1 Bayes Factor Definition and Meaning

The Bayes Factor [10] is a measure that compares how well 2 models are at describing a particular set of observations. It does this by comparing the marginal likelihoods of a set of observations under 2 models.

$$BayesFactor = \frac{p(y \mid X_1)}{p(y \mid X_2)} \quad [8]$$

Where y is the data, X_1 is the design matrix for the model using the first set of basis functions and X_2 is the design matrix for the model using the second set of basis functions.

The Bayes Factor is useful for comparing 2 or more models and choosing which is best.

6.2 Obtaining the Bayes Factor in Python

The function 'comparey' obtains a set of data, and compares the loglikelihood of y for 2 different models. The 'BayesFactor' function then uses these loglikelihoods to calculate the Bayes Factor:

```
def comparey(f, n, sigma_noise, p1, p2, V1, V2, a, b):
    [x,y] = simulate(f, n, sigma_noise)
    #constructing design matrix for vector of basis functions p1
    m1 = len(p1)
    X1 =  np.zeros([n, m1])

    for i in range(n):
        for j in range(m1):
            X1[i,j]= p1[j](x[i])

    mar_y1 = logmarginaly(y, X1, V1, a, b)

    #constructing design matrix for vector of basis functions p2
    m2 = len(p2)
    X2 =  np.zeros([n, m2])

    for i in range(n):
        for j in range(m2):
            X2[i,j]= p2[j](x[i])

    mar_y2 = logmarginaly(y, X2, V2, a, b)

    return (mar_y1,mar_y2)

def BayesFactor(f,n,sigma_noise,p1, p2, V1,V2, a, b):


    logfy1 = comparey(f,n,sigma_noise,p1,p2,V1,V2,a,b)[0]
    logfy2 = comparey(f,n,sigma_noise,p1,p2,V1,V2,a,b)[1]

    logB= logfy1 - logfy2

    B = np.exp(logB)

    return B
```

6.3 Interpreting the Bayes Factor

The Bayes Factor is great for comparing 2 models to each other, a high Bayes Factor indicates that the first model (in the numerator) is better. To help interpret the Bayes Factor, authors have created tables with levels of support: While the Bayes Factor is effective at choosing one model over another, it should be noted that a high Bayes Factor doesn't necessarily make the model in the numerator 'good'. For example, the model in the denominator may be so awful that comparing an average model to it would give a high Bayes Factor.

- 1-3: anecdotal
- 3-10: moderate
- 10-30: strong
- 30-100: very strong
- > 100: extreme

Figure 5: Table for interpreting the Bayes Factor [9]

7 Example

We will now consider an example in order to find the best set of basis functions to use in the design matrix,

7.1 Simulating

Let's consider $f(x) = log(1+x)$. We will firstly simulate data from this function with noise:

```
#set parameters we will use in function
f = lambda x: np.log(1+x)
n = 100
sigma_noise = 0.1
In[139]: simulate(f,n,sigma_noise)
```

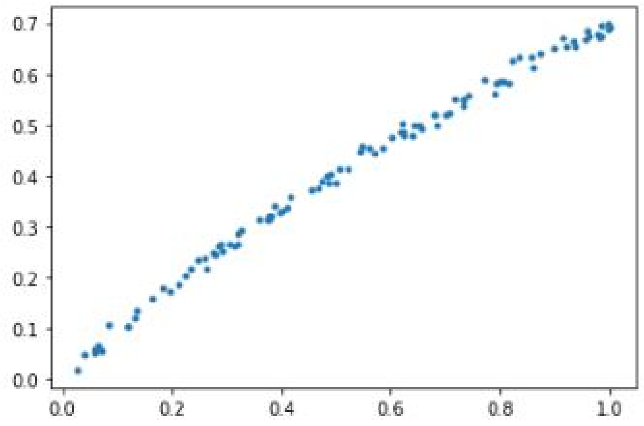


Figure 6: Graph showing 100 simulated data points

7.2 Choosing Optimal Basis Functions

We will start with the simple polynomial basis functions. The following python function obtains the best degree of polynomial to go up to in our basis functions. If we use a degree of polynomial that is too high, the data becomes ”overfitted” [6] but if we use a degree too low, it is too simple to represent the data well.

The function works by comparing the value of the marginal likelihood for y for each set of polynomial basis functions from degree 0 to degree n_poly. It outputs the value of n such that going up to that degree of polynomial gives the highest value for the likelihood of y.

```
def bestpoly(f, n, sigma_noise, a, b, n_poly):
    #n_poly is the degree that we will trial up to
    [x,y] = simulate(f, n, sigma_noise)

    poly_logy = []
    #for vectors [1] to [1, x, ... , x^(n_poly-1)]
    for i in range(n_poly):
        p = polybasis(i)
        m = len(p)
        V = np.identity(m)*k_v
        X = np.zeros([n, m])
        for i in range(n):
            for j in range(m):
                X[i,j]= p[j](x[i])

        logy = logmarginaly(y, X, V, a, b)
        poly_logy.append(logy)

    bestpoly = max(poly_logy)
    #best_n corresponds to the vector [1, x, ..., x^best_n]
    which maximises the marginal of y
    best_n =poly_logy.index(bestpoly)
    return  best_n
```

We then set the parameters to appropriate values and run the function:

```
a = 1
b = 1
k_v1= 1

In [44]: bestpoly(f,n,sigma_noise,a,b,200)
Out[44]: 2
```

So now we know that using the basis functions $\phi_0(x) = 1, \phi_1(x) = x, \phi_2(x) = x^2$ maximises the marginal of y.

The following 2 functions do the same thing for the Gaussian and piecewise linear basis functions:

```
def bestgauss(f, n, sigma_noise, a, b, n_gauss):
    [x,y] = simulate(f, n, sigma_noise)

    gauss_logy = []
    for i in range(2,n_gauss):
        p = gaussbasis(i)
        m = len(p)
        V = k_v1*(np.identity(m))
        X = np.zeros([n, m])
        for i in range(n):
            for j in range(m):
                X[i,j]= p[j](x[i])

        logy = logmarginaly(y, X, V, a, b)
        gauss_logy.append(logy)

    bestgauss = max(gauss_logy)

    best_n = gauss_logy.index(bestgauss) + 2

    return  best_n

def bestpiecewise(f, n, sigma_noise, a, b, n_piecewise):
    [x,y] = simulate(f, n, sigma_noise)

    piecewise_logy = []
    #for vectors [1] to [1, x, ... , x^(n_poly-1)]
    for i in range(2, n_piecewise + 1):
        p = piecewisebasis(i)
        m = len(p)
        V = np.identity(m)*k_v
        X = np.zeros([n, m])
        for i in range(n):
            for j in range(m):
                X[i,j]= p[j](x[i])

        logy = logmarginaly(y, X, V, a, b)
        piecewise_logy.append(logy)

    bestpiecewise = max(piecewise_logy)
    #best_n corresponds to the vector
    [1, x-t_0, ..., x-t_best_n,...,
    best_n =piecewise_logy.index(bestpiecewise) + 2
    return  best_n

In[121]: bestgauss(f,n,sigma_noise,a,b,200)
Out[121]: 3

In[138]: bestpiecewise(f,n,sigma_noise,a,b,200)
Out[138]: 57
```

So we now also have the number of Gaussian and piecewise linear basis functions that maximise the marginal likelihood of y for our data.

7.3 Using Bayes Factors to Choose a Model

We will now use Bayes Factors to compare the 3 models that we have. Since the Bayes Factor fluctuates slightly depending on the data, we will write a function that takes an average over 500 simulations. Additionally, to simplify notation, we will let model 1 be the polynomial basis functions, model 2 be the Gaussian basis functions and model 3 be the piecewise linear basis functions.

```
def BayesAverage(f,n,sigma_noise,p1,p2,V1,V2,a,b):

    total=0
    for i in range(500):
        total = total + BayesFactor(f,n,sigma_noise,p1,p2,V1,V2,a,b)
    avg= total/500
    return avg
#set our basis functions
p1 = polybasis(2)
k_v1 = 1
V1 = k_v1 * np.identity(len(p1))

p2= gaussbasis(3)
k_v2=  1
V2= k_v2 * np.identity(len(p2))

p3= piecewisebasis(57)
k_v3= 1
V3= k_v3 * np.identity(len(p3))

#calculate Bayes factors
In[188]: BayesAverage(f,n,sigma_noise,p1,p2,V1,V2,a,b)
Out[188]: 1.7489343632488685

In[190]: BayesAverage(f,n,sigma_noise,p3,p1,V3,V1,a,b)
Out[190]: 41.2498815247726
```

```
In[192]: BayesAverage(f,n,sigma_noise,p3,p2,V3,V2,a,b)
Out[192]: 65.5850069517603
```

The above results give us lots of information. Model 1 (polynomial basis functions) is slightly better than model 2 (Gaussian basis functions), however with the Bayes Factor only being about 1.75, this is only "anecdotal"[9]. The argument for choosing model 3 (piecewise linear basis functions) over models 1 and 2 is "very strong" [9], with the high Bayes Factors of 41.25 between model 3 and model 1 and 65.59 between model 3 and model 2. Therefore we can conclude that the best model to use given the function $f(x) = log(1 + x)$ and parameters used is model 3.

7.4 Varying Parameters

The conclusion we reached in the previous section was for the following parameters:

- a = 1
- b = 1
- n = 100
- V = identity (with size corresponding to the number of basis functions)
- sigma_noise = 0.1

We now aim to see the impact of varying the parameters n and V.

7.4.1 Varying n

We will investigate the impact of decreasing the number of data points in our simulation to 25, and then increasing it to 500. We will start by recalculating for each type of basis function the number of basis functions that maximises the marginal likelihood of y. We will then recalculate the Bayes Factors. In order to simplify comparisons, we will only calculate the Bayes Factor between model 3 and model 1, and between model 3 and model 2.

```
n = 25
In[231]: bestpoly(f,n,sigma_noise,a,b,100)
Out[231]: 2

bestgauss(f,n,sigma_noise,a,b,100)
Out[232]: 3

bestpiecewise(f,n,sigma_noise,a,b,100)
Out[233]: 9
```

```
#bestpiecewise has changed so edit p3
p3= piecewisebasis(9)

In[239]: BayesAverage(f,n,sigma_noise,p3,p2,V3,V2,a,b)
Out[239]: 1.772804662999448

In[240]: BayesAverage(f,n,sigma_noise,p3,p1,V3,V1,a,b)
Out[240]: 1.0673259580491956

n=500
In[256]: bestpoly(f,n,sigma_noise,a,b,100)
Out[256]: 6

In[257]: bestgauss(f,n,sigma_noise,a,b,100)
Out[257]: 5

In[258]: bestpiecewise(f,n,sigma_noise,a,b,200)
Out[258]: 150

#edit p1, p2 and p3
p1 = polybasis(6)
p2= gaussbasis(5)
p3= piecewisebasis(150)

In[261]: BayesAverage(f,n,sigma_noise,p3,p1,V3,V1,a,b)
Out[261]: 28521330.340348348

In[262]: BayesAverage(f,n,sigma_noise,p3,p2,V3,V2,a,b)
Out[262]: 7307303.576282859
```

It is easy to see that a decrease in n results in a smaller Bayes Factor, and an increase in n dramatically increases the Bayes Factor. This intuitively makes sense, as if model 3 is the best for fitting the data, then seeing more observations of data would make us even more certain that model 3 is better.

7.4.2 Varying V

We will now investigate the impact of changing V on the Bayes Factor. We will change V from the identity to 5 times the identity, and then to 0.5 times the identity. This represents a change in our prior beliefs about β. If multiplied by a high constant, the prior distribution of β has high variance representing weak prior beliefs, and if its multiplied by a low constant, the prior distribution has low variance representing strong prior beliefs.

```
k_v1 = 5
k_v2=  5
k_v3= 5
```

```
In[300]: bestpoly(f,n,sigma_noise,a,b,100)
Out[300]: 1
In[302]: bestgauss(f,n,sigma_noise,a,b,100)
Out[302]: 3
In[303]: bestpiecewise(f,n,sigma_noise,a,b,100)
Out[303]: 2
```

Interestingly, it can be seen that our piecewise basis functions reduce down to the simple case, where it is just a global linear polynomial between 0 and 1. Therefore, since our best polynomial basis is of degree 1, we would expect the Bayes Factor between these to be 1.

```
p1 = polybasis(1)
p2= gaussbasis(3)
p3= piecewisebasis(2)
In[305]: BayesAverage(f,n,sigma_noise,p3,p1,V3,V1,a,b)
Out[305]: 0.9999999999999996

In[304]: BayesAverage(f,n,sigma_noise,p3,p2,V3,V2,a,b)
Out[304]: 2.395249455483294
```

As expected, the Bayes Factor between polynomial basis and piecewise basis is very close to 1. Additionally, the Bayes Factor between the piecewise and Gaussian basis has reduced dramatically. This suggests that having little prior knowledge of β reduces our confidence in our choice of model. Similarly, choosing V to be a 0.5 times the identity:

```
k_v1 = 0.5
k_v2=  0.5
k_v3= 0.5

In[309]: bestpoly(f,n,sigma_noise,a,b,10)
Out[309]: 3

In[310]: bestgauss(f,n,sigma_noise,a,b,100)
Out[310]: 4

In[312]: bestpiecewise(f,n,sigma_noise,a,b,200)
Out[312]: 120

In[314]: BayesAverage(f,n,sigma_noise,p3,p2,V3,V2,a,b)
Out[314]: 5009.80785156057

In[315]: BayesAverage(f,n,sigma_noise,p3,p1,V3,V1,a,b)
Out[315]: 2222.0465877369143
```

So increasing our prior knowledge of β results in an even higher Bayes Factor, so increases our confidence in the choice of model. Again, this intuitively makes sense, as the more uncertainty we have about parameters, the less confidence we will have in our final selection.

8 Example of Bayesian Linear Regression

8.1 Introduction

This section focuses on applying the Bayesian linear regression methodology to effects of aging, alcohol consumption and tobacco consumption on the likelihood of developing (o)esophageal cancer. One data set is used, compiled in Ille-et-Vilaine (see Breslow Day). The data set is relatively small, so we are careful and conservative with the steps taken. We follow the methodology outlined in Lai Xing (2008), Gelman et. al. (2003) and Bishop (2007). The methodology has been well-researched, so our contribution is largely empirical. We discover that the estimated effects of aging and health habits are mostly in-line with our conjectures. We also discover that the effects are not big. Overall, the cancer risk is driven by many other life circumstances, genetic characteristics, and so on. The analysis has been performed in R.

8.2 Data

In this paper, we use data from a case-control study of (o)esophageal cancer in Ille-et-Vilaine, France. For details, see Breslow Day (1980)[12]. The available variables are the following.

- Age group: possible values are “25-34 years”, “35-44 years”, “45-54 years”, “55-64 years”, “65-74 years”, “75+ years”.
- Alcohol consumption: possible values are “0-39 gm/day”, “40-79 gm/-day”, “80-119 gm/day”, “120+ gm/day”.
- Tobacco consumption: possible values are “0-9 gm/day”, “10-19 gm/day”, “20-29 gm/day”, “30+ gm/day”
- Number of cancer cases.
- Number of controls (no evidence of (o)esophageal cancer).

  Overall, the data set contains records for 88 age / alcohol / tobacco combinations. The amount of information is scarce, the data set is small. For that reason, it is not feasible to treat age, alcohol consumption and tobacco consumption as categorical or ordinal variables. In fact, they were not such variables in the first place. What happened is that the world lost much valuable information. Somebody discretized the variables into just a few categories. To make most of what is left, we recode the variables back into their continuous, numeric form. Each category is recoded as its mean value, with the exception of the highest category. The highest category is recoded as its lower bound because of the absence of any other details. For example, age group "45-54 years" is coded as 49.5 while age group "75+ years" is coded as 75. We are aware of the "censoring" issues when it comes to the highest categories. Yet, this is the simplest objective way to treat the data in our case... The created numeric variables are labelled as Age, Alcohol and Tobacco. We are interested in effects of age, alcohol consumption and tobacco consumption on cancer incidence. The incidence of most phenomena depends on the size of the studied group, which may vary over time and demographic conditions. So it makes sense to consider variable

$$CancerRate = \frac{[number\ of\ cancer\ cases]}{[number\ of\ cancer\ cases + number\ of\ controls]}.$$

In addition to its relative "robustness" over various conditions, the newly created variable exhibits a distributional nicety. It is an average of conditionally independent Bernoulli trials. As such, it is approximately normally distributed by the Central Limit Theorem. The property justifies our using a normal linear model to relate Cancer Rate to the three predictors (Age, Alcohol and Tobacco).

8.3 Methodology

The focus is to explain variability in Cancer Rate with variability in Age, Alcohol and Tobacco. Two important aspects of the data set drive our modeling choice.

- The data set is relatively small. So no fancy, non-linear modeling is feasible.

- Dependent variable Cancer Rate is approximately normally distributed. This serves as a partial justification of utilizing a normal model, where the residuals are assumed to be normally distributed.

- The data set is even smaller than it seems. The provided information is limited since the three predictors are marked on a grid. They are allowed to take very few values. This limits our ability to study heteroskedastic and correlation effects. A simple i.i.d. model is a parsimonious choice, suitable for the data set in hand.

All in all, we assume the following structure. Let Y be the vector of Cancer Rate observations. Let X be a matrix of observations of a subset of 1, Age, Alcohol, Tobacco.

$$Y \sim N(X\beta, \sigma^2 I_n)$$

where
$\beta = (\beta_1, ..., \beta_p)$ is the vector of regression coefficients,
σ^2 is the residual variance,
I_n is an n-by-n identity matrix,

n is the sample size (88 in our case).

Regression parameters β and σ^2 are random. This is where our paradigm becomes Bayesian. The regression parameters come from the following prior distribution:

$$\tau \sim gamma(g, \lambda)$$
$$\sigma^2 = \frac{1}{2\tau}$$
$$V = I_p s_1^2$$
$$\beta \mid \sigma^2 \sim N(z, \sigma^2 V)$$

where λ is the scale, not the rate and z is a p-dimensional vector.

Quantities z, s1, g and λ are referred to as the hyper-parameters. In any particular Bayesian model they are specified. After observing the data, we make inference about β and σ^2 using their posterior distribution:

$$\tau \text{ given } [X, Y] \sim gamma(g + \frac{n}{2}, \frac{1}{a_n})$$
$$\sigma^2 = \frac{1}{2\tau}$$
$$\beta \text{ given } [X, Y, \sigma^2] \sim N(\beta_n, \sigma^2 V_n)$$

where

$$V_n = (V^{-1} + X^T X)^{-1}$$
$$B_n = V_n(V^{-1}z + X^T Y)$$
$$a_n = \frac{1}{\lambda} + z^T V^{-1} z + Y^T Y - \beta_n^T V_n^{-1} \beta_n$$

The posterior means can serve as estimates of β and σ . They have simple expressions [11]:

$$E[\beta | X, Y] = \beta_n$$
$$E[\sigma^2 | X, Y] = \frac{a_n}{2(g + (n/2) - 1)}$$

The formulas above are taken from Lai Xing (2008), pp 107 – 108... Naturally, the hyperparameters "order the music "[11] here. Their choice is important. There are two ways of specifying g and λ.

- Manual method: we can set g and λ to values reasonable from the point of view of prior research and experience.

- τ-inspired method: we notice that the cancer rate is always between 0 and 1. By the properties of binomial distribution, the highest standard deviation it can have is 0.5 (and even that is rare). The lowest standard deviation of any random variable is 0. We choose the middle (0.25) as a typical value of the standard deviation (σ). This prompts us to choose 1 / (2 * 0.252) as E[τ]. Next the standard deviation of τ is set as follows:

  $$Var[\tau]^{1/2} = s_2 E[\tau]$$

  where s_2 is some reasonably big number. Values of g and λ are then the solutions of system

  $$E[\tau] = g\lambda$$
  $$Var[\tau] = g\lambda^2$$

We experiment with both methods in our analysis. In general, we try to keep the prior distribution spread out, which means that both s_1 and s_2 must be reasonably large.

Together z, s_1, s_2, g and λ constitute the complete set of hyperparameters. We try many specifications of the hyperparameters. The ultimate choice is helped by ideas of empirical Bayes. For each specification of hyperparameters, we calculate the marginal, integrated likelihood. We then choose the specification corresponding to the highest marginal likelihood (or, equivalently, log-likelihood).

The analysis is performed in the following sequence of steps.

1) We choose the optimal set of hyperparameters using marginal log-likelihood.

2) For the optimal set of hyperparameters (optimal prior), we estimate the full model. This is the model containing the intercept and all the predictors: Age, Alcohol and Tobacco. We take a note of the terms which are statistically significant at the 5 % significance level. For that we use posterior credible intervals.

3) We polish the model using backward stepwise selection. This means that we drop the least significant terms one by one until all the remaining terms are significant. The outcome constitutes the optimal model... As a double-check, we compare the optimal model to the previous iterations using Bayesian Information Criterion (BIC).

4) We exploit the optimal model for inference regarding the effects of age, alcohol and tobacco on cancer risks.

5) We take a note of the ways in which the estimated model may be less than ideal.

8.4 Code

```
Bayesian_Linear_Model <- function(Y, X, Intercept = TRUE, z = NULL, g = NULL,
lambda = NULL, v_1 = 10, v_2 = 10, CI_Level = 0.95, MC.Scenarios = 1e4, MC.Seed = 1) {

################################################################################
# This functon implements Bayesian estimation of the normal linear model
# where the residual variance is unknown. The function exploits formulas on
# pp 107-108 of Lai & Xing. In the code below, we mimic the notation of
# Lai & Xing wherever appropriate.

library('MASS')
library('car')

################################################################################
# SETTINGS

Results_Folder <- 'Model'

################################################################################
# PRELIMINARIES

n <- dim(X)[1]
if(Intercept) {
X <- cbind(rep(1,n), X)
Parameter_Names <- c(names(X), 'Sigma Sq')
Parameter_Names[1] <- 'Intercept'
} else {
Parameter_Names <- c(names(X), 'Sigma Sq')
}
p <- dim(X)[2]

Y  <- as.numeric(Y)
X  <- as.matrix(X)

################################################################################
# HYPERPARAMETER SELECTION

if(is.null(z)) {
z <- rep(0,p)
}

if(is.null(g) || is.null(lambda)) {
tau_mean <- 1 / (2 * 0.25^2)
tau_std <- tau_mean * v_2
```

```
g <- tau_mean^2 / tau_std^2
lambda <- tau_std^2 / tau_mean
}

##############################################################################
# MODEL ESTIMATION

V <- v_1 * diag(p)
V_Inv <- solve(V)
V_n <- solve(V_Inv + t(X) %*% X)
beta_n <- V_n %*% (V_Inv %*% z + t(X) %*% Y)
a_n <- 1 / lambda + t(z) %*% V_Inv %*% z + t(Y) %*% Y - t(beta_n) %*% V_Inv %*% beta_n
sigma_sq_n <- a_n / (2 * (g + n / 2 - 1))

##############################################################################
# SIMULATION

cat('\nPOSTERIOR SIMULATION:')
param_post <- data.frame( array(rep(NA,MC.Scenarios*(p+1)), dim = c(MC.Scenarios,(p+1))) )
names(param_post) <- Parameter_Names

set.seed(MC.Seed)
for(scen in 1:MC.Scenarios) {
tau_r  <- rgamma(1, shape = g + n / 2, scale = 1 / a_n)
sigma_sq_r <- 1 / (2 * tau_r)
beta_r <- mvrnorm(n = 1, mu = beta_n, Sigma = V_n * sigma_sq_r)
param_post[scen,] <- c(beta_r, sigma_sq_r)

if(scen %% 1e3 == 0) {
cat(paste('\n    Simulated ', scen, ' random scenarios so far.', sep = ''))
}
}
cat('\n\n')

write.csv(param_post, paste(Results_Folder, '/Posterior_Distribution.csv', sep = '')
, row.names = FALSE)

##############################################################################
# MODEL DIAGNOSTICS

Estimates  <- data.frame( array(rep(NA,(p+1)*4), dim = c(p+1,4)) )
names(Estimates) <- c('Parameter', 'Estimate', 'CI Lower Bound', 'CI Upper Bound')
Estimates[,1] <- Parameter_Names
Estimates[,2] <- c(beta_n, sigma_sq_n)

for(j in 1:(p+1)) {
```

```
Estimates[j,3] <- quantile(param_post[,j], (1 - CI_Level) / 2)
Estimates[j,4] <- quantile(param_post[,j], CI_Level + (1 - CI_Level) / 2)
}

write.csv(Estimates, paste(Results_Folder, '/Estimates.csv', sep = ''),
row.names = FALSE)

################################################################################
# PLOTTING

jpeg(paste(Results_Folder, '/Posterior_Distribution.jpeg', sep=''), width = 480 * 3,
height = 240 * 3)
scatterplotMatrix(param_post, smooth = TRUE, regLine = FALSE, col = 'blue')
dev.off()

Estimates
```

8.5 Results

The hyperparameter selection is summarized in table 1. We try hyperparameter specifications until the marginal log-likelihood does not improve perceptibly. This involves exploring 20 different models. The hyperparameter specification in model 19 turns out to be optimal.

Whenever $s_2 = NA$ in table 1, this means that g and λ are set manually. They are not "inspired" by τ-related calculations.

The complete results, with all the estimates for each model, are available as a zip archive upon request. They are too big to include in this document. One pleasant finding that we make is that the marginal log- likelihood and parameter estimates do not vary substantially over the prior specifications. This means that our analysis is stable. It is robust. Our findings are not all over the place depending on the choice of prior. In fact, all 20 models point to essentially same relationships. This is a somewhat pleasant surprise given the size of the data set.

Model	Intercept	Age	Alcohol	Tobacco	s_1	s_2	g	λ	Marginal Log-likelihood
1	0	0	0	0	10	10	0.01	800	32.35142
2	0	0	0	0	3	10	0.01	800	32.41730
3	0	0	0	0	10	3	0.1111111	72	32.35169
4	0	0	0	0	3	3	0.1111111	72	32.41745
5	0	0	0	0	1	3	0.1111111	72	32.68782
6	0	0	0	0	3	1	1	8	32.41835
7	0	0	0	0	1	1	1	8	32.68333
8	0	0	0	0	1	NA	0.1111111	40	32.63579
9	0	0	0	0	1	NA	0.1111111	150	32.72167
10	0	0	0	0	1	NA	0.1111111	300	32.73730
11	0	0	0	0	1	NA	0.1111111	600	32.74512
12	0	0	0	0	1	NA	0.1111111	1200	32.74903
13	0	0	0	0	1	NA	0.1111111	2400	32.75098
14	0	0	0	0	1	NA	0.1111111	10000	32.75247
15	0	0	0	0	1	NA	0.1111111	40000	32.75282
16	0	0	0	0	1	NA	0.1111111	160000	32.75291
17	0	0	0	0	1	NA	0.1111111	640000	32.75293
18	0	-0.002	-0.002	-0.002	1	NA	0.1111111	640000	32.75280
19	0	0.002	0.002	0.002	1	NA	0.1111111	640000	32.75295
20	0	0.004	0.004	0.004	1	NA	0.1111111	640000	32.75286

For the optimal set of hyperparameters, we estimate the full model. The model is summarized in table 2. We note that most terms are statistically significant and the coefficients have expected signs. Most signs (directions of the relationships) are consistent with the common sense and our prior experience. The only surprise is that, after adjusting for the effects of Age and Alcohol, Tobacco is non-significant. Perhaps, esophageal cancer is less dependent on smoking than lung cancer is. Or, perhaps, our data set is too small and weak relationships are hard to detect.

Parameter	Estimate	95% CI Lower Bound	95% CI Upper Bound
Intercept	-0.32432	-0.56020	-0.08448
Age	0.00650	0.00291	0.01006
Alcohol	0.00232	0.00080	0.00384
Tobacco	0.00137	-0.00480	0.00727
σ^2	0.07927	0.05870	0.10630

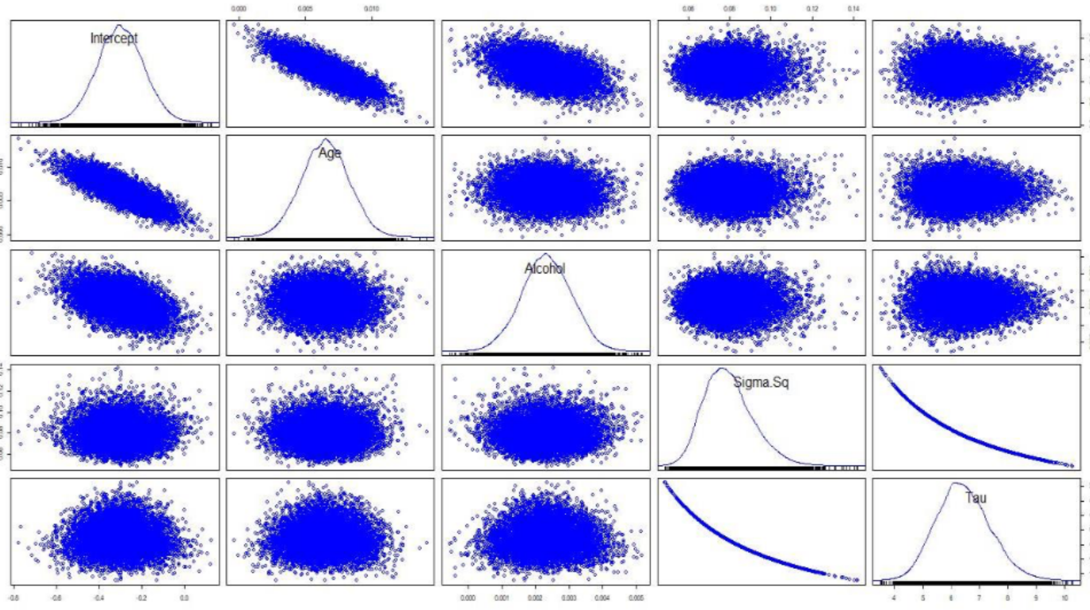
Table 2: estimate of model 19: z = (0, 0.002, 0.002, 0.002), s_1 = 1, g = 0.1111111, λ = 640000. Marginal log-likelihood = 32.75295, BIC = -38.64188. "CI" stands for "credible interval".

We drop Tobacco from the model and see that all other effects are statistically significant at the 5% significance level. We have arrived at the optimal linear model. Adding bells and whistles would not be wise given the sample size. The optimal model is summarized in table 3. We are reassured by the fact that the BIC score has decreased since the last iteration. The lower is the BIC score the better is the model.

Parameter	Estimate	95% CI Lower Bound	95% CI Upper Bound
Intercept	-0.30060	-0.51617	-0.08283
Age	0.00650	0.00290	0.01005
Alcohol	0.00232	0.00081	0.00382
σ^2	0.07945	0.05905	0.10662

Table 3: estimate of model 19: z = (0, 0.002, 0.002), s_1 = 1, g = 0.1111111, λ = 640000. Marginal log-likelihood = 31.66723, BIC = -40.94777. "CI" stands for "credible interval".

The posterior distribution implied by the optimal model is visualized in graph 1. Nothing looks out of place. No co-linearity between Age and Alcohol is detected. The residual variance σ^2 is estimated at a realistic level.



Graph 1: posterior distribution of (β, σ², τ) implied by the optimal model (model 21).

To illustrate that our inference is stable, we modify the full model for optimal hyper-parameters to impose zero prior mean on the regression coefficients (no prior knowledge of the direction of the relationships). As shown below, the data dictate all the decisions and the estimated model (table 4) is almost the same as the model where the regression coefficients have non-zero mean (table 2).

Parameter	Estimate	95% CI Lower Bound	95% CI Upper Bound
Intercept	-0.32431	-0.56019	-0.08447
Age	0.00650	0.00291	0.01006
Alcohol	0.00232	0.00080	0.00384
Tobacco	0.00137	-0.00480	0.00727
σ^2	0.07927	0.05870	0.10630

Table 4: estimate of model 17: z = (0, 0, 0, 0), s_1 = 1, g = 0.1111111, λ = 640000. Marginal log-likelihood = 32.75293, BIC = -38.64184. "CI" stands for "credible interval".

8.6 Conclusions

The unimportance of tobacco consumption is quite baffling. It is baffling because old people, people abusing alcohol and people abusing tobacco are quite different crowds. One membership does not imply another, or the lack of such. Our best explanation:

- smoking is more important for lung cancer than (o)esophageal cancer;
- the effect of smoking may be small enough not to be detected on 88 observations, but it may well be detected on 880 observations.

Based on the optimal model, our inference is the following:

- other things being equal, older people are more likely to get cancer,
- other things being equal, people consuming more alcohol are more likely to get cancer.

Note that alcohol is "pooled" in our study. It is quite conceivable that the effect of beer is very different from the effect of wine, which is very different from the effect of brandy, which is different from the effect of vodka, and so on.

This has been said many times before (in a way), so this is the last time: the major direction for future research is trying the estimated models on a much bigger data set and seeing if our conclusions are relatively stable over the sample size.

9 References

References in text:

- 1 Rencher AC; Schaalj GB. Linear Models in Statistics, 2nd Edition. Hoboken: John Wiley Sons, Inc; 2008, p277

- 2 Copp L. From: Quora. What is the advantages of Bayesian inference compared to statistical significance in experimental testing? Available from: https://www.quora.com/What-is-the-advantages-of-Bayesian-inference-compared-to-statistical-significance-in-experimental-testing [Accessed: 9th June 2020]
- 3 Rencher AC; Schaalj GB. Linear Models in Statistics, 2nd Edition. Hoboken: John Wiley Sons, Inc; 2008, p277
- 4 Rencher AC; Schaalj GB. Linear Models in Statistics, 2nd Edition. Hoboken: John Wiley Sons, Inc; 2008, p280
- 5 Rencher AC; Schaalj GB. Linear Models in Statistics, 2nd Edition. Hoboken: John Wiley Sons, Inc; 2008, p281
- 6 From: University of Toronto. Modeling Data with Linear Combinations of Basis Functions. Available from: `http://www.utstat.utoronto.ca/~radford/sta414.S11/week1b.pdf` [Accessed: 10th June 2020]
- 7 Image From: TU Wien. Basis Functions and Domain Partitioning. Available from: `https://www.iue.tuwien.ac.at/phd/singulani/disssu12.html` [Accessed: 10th June 2020]
- 8 Equation from: The PyMC Development Team. Bayes Factors and Marginal Likelihood. Available from:`https://docs.pymc.io/notebooks/Bayes_factor.html`[Accessed: 4th June 2020]
- 9 Table From: The PyMC Development Team. Bayes Factors and Marginal Likelihood. Available from:`https://docs.pymc.io/notebooks/Bayes_factor.html`[Accessed: 4th June 2020]
- 10 Kass RE; Raftery AE. Bayes Factors. Journal of the American Statistical Association, 1995, Vol. 90, No. 430, p776
- 11 Lai, T. L., Xing, H. (2008). Statistical Models and Methods for Financial Markets. Springer. p107-108
- 12 Breslow, N. E., Day, N. E. (1980). Statistical Methods in Cancer Research. Volume 1: The Analysis of Case- Control Studies. IARC Lyon / Oxford University Press.

Further Background Reading:

- Bishop, C. M. (2007). Pattern Recognition and Machine Learning (corr. 2nd printing ed). Springer.
- Gelman, A., Carlin, J. B., Stern, H. S., Rubin, D. B. (2003). Bayesian Data Analysis (2nd ed). Chapman and Hall / CRC.