UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Advanced Regression

Assignment 2

Regression and Bayesian Regression Techniques

Julian Albert
ALBJUL005

Department of Statistical Sciences
University of Cape Town
South Africa
April 23, 2019

Contents

Question 1 **2**
a) LOOCV and Principal Components 2
b) Regression Comparisons 4
Principal Component Regression 4
Lasso Regression 5
Comparison 5
c) Shrinkage in Ridge Regression 5
d) Bayesian Ridge Regression 6

Question 2 **8**
a) Modelling Profiles of One Group 8
i) Posterior Specification and Gibbs Sampling 9
ii) Fitting, Credibility Intervals and Visual Comparison 10
b) Modelling Profiles for Both Groups 12
i) Posterior Distributions 12
ii) Gibbs Sampling and Tuning 13
iii) Model Fit 13
c) Modelling for Auto-Correlation in the Data 14

Question 1

In this question I will consider the prostate dataset from the ElemStatLearn [2] package in R. The dataset is aimed at examining the correlation between the level of a prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. The dataset is already categorised into training and testing sets with 67 (69.07%) of the observations being used as a training set and the other 30 (30.93%) being used as the testing set. I will utilise various regression techniques such as ordinary least squares (OLS), ridge, principal component, lasso and bayesian ridge regression and compare the effectiveness of each. Before we begin I chose to standardise the dataset such that,

$$\vec{x}_i = \frac{\vec{x}_i - \mu_{x_i}}{\sigma_{x_i}}, \quad \vec{y} = \frac{\vec{y} - \mu_y}{\sigma_y}. \tag{1}$$

Where $\mathbf{X}$ is the $n \times p$ predictor matrix composed of $\vec{x}_i$ predictor vectors for $i = 1, 2, \ldots, p$ and $\mu_{x_i},\ \sigma_{x_i}$ representing the mean and standard deviation for each predictor. Similar notation is used for our respinse vector (only one response).

a) LOOCV and Principal Components

The first technique to be investigated is a shrinkage method for regression known as ridge regression. Shrinkage methods are more continuous and this do not suffer as much from high variability as conventional subset methods. Ridge regression "shrinks" the regression coefficients by imposing a penalty (complexity parameter) on their size. The ridge coefficients minimize a penalized residual sum of squares given by

$$\hat{\beta}_{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \quad [1]. \tag{2}$$

From Equation 2 we can see that the last summation excludes the β_0 term from the penalty parameter λ as penalisation of the intercept would make the procedure dependent on the origin chosen for the response. This is problematic as it would add a constant c to each of the responses y_i which does not simply correspond to a shift of the predictions by the same amount c. As such we can estimate β_0 such that

$$\beta_0 = \bar{\vec{y}} = \frac{1}{N} \sum_{i=1}^{N} y_i \quad [1].$$

The main criticism of penalised models involve selecting an appropriate value for the complexity parameter (often denoted *lambda*). For the purpose of this question we will select λ such that it minimises a cross-validation error in a procedure known as leave-one-out cross-validation (LOOCV). To perform this we re-write Equatio n 2 in matrix form for easier computation, thus we obtain a new form the the regression coefficients

$$\hat{\beta}_{ridge} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}' \vec{y}. \tag{3}$$

Where $\boldsymbol{I}$ is the $p \times p$ identity matrix to penalise each coeffiecient by λ. Notice here that by setting the complexity parameter $\lambda = 0$ we would obtain the standard OLS regression

coefficients. LOOCV utilises a single observation from the original sample as the validation (testing) data with the remaining observations as the training set. This is repeated such that each observation in the sample is used once as the validation data. By finding the parameter of interest (λ) that minimises the error between training and testing we assume this to be the value that best generalises the model. To perform the LOOCV procedure we can implement Algorithm 1.

Algorithm 1 LOOCV for Ridge Regression

procedure CVRidge(x, λ) ▷ The CV error for data (x) and complexity parameter (λ)
    Shuffle the Data.
    Define n observations and p predictors with response $\vec{y}$.
    $\mathbf{I} \leftarrow$ matrix ▷ Initialise $(p) \times (p)$ Identity matrix
    for $i \leftarrow 1, n$ **do**
        Set an index equal to i.

        Subset training data to be not row i of the shuffled data.
        Denoting the predictors as $\boldsymbol{X}_{-i}$ and response as $\vec{y}_{-i}$

        Subset testing data to be row i of the shuffled data.
        Denoting the predictors as $\boldsymbol{X}_i$ and response as y_i

        Calculate the $\hat{\beta}_{-i}$ coefficients as per Equation 3 using $\boldsymbol{X}_{-i}$ and $\vec{y}_{-i}$
        Calculate prediction $\hat{f}(\boldsymbol{X}_i) = \boldsymbol{X}_i \hat{\beta}_{-i}$
        Cross-Validation Error is $CV_i = y_i - \hat{f}(\boldsymbol{X}_i)$
    end for
    Total CV Error (CV_{total}) is $\frac{1}{n}\sum_{i=1}^{n}(CV_i)^2$
    return CV_{total}
end procedure

The optimal value of λ can be shown to be

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \vec{x}_i' \hat{\beta}_{ridge}(\lambda)}{1 - \boldsymbol{H}_i(\lambda)} \right)^2 \quad , \tag{4}$$

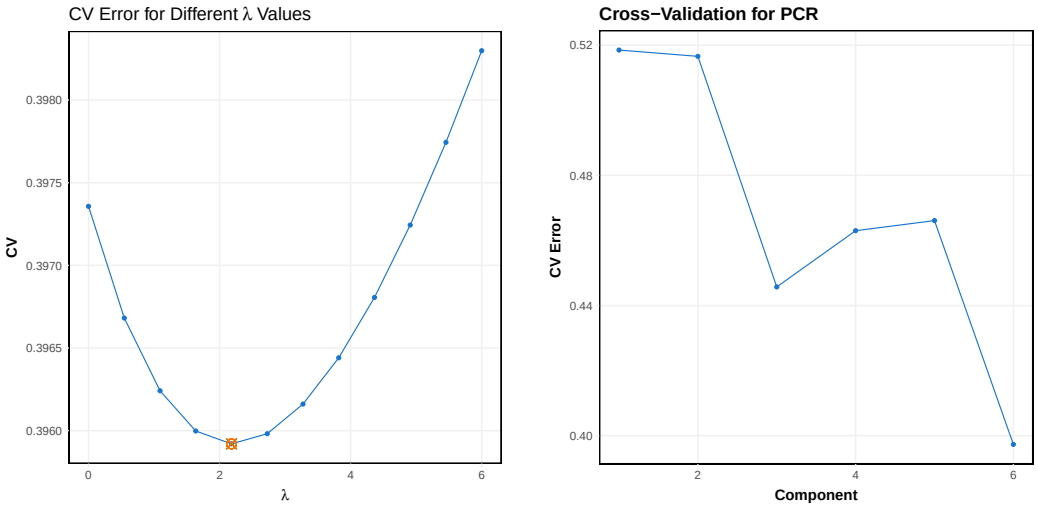
however, the dataset is sufficiently small thus the results from Algorithm 1 should yield similar results to Equation 4. Implementing Algorithm 1 I can obtain results seen in Figure 1 where the minimum CV error is 0.406 corresponding to an optimal complexity parameter $\lambda^* = 2.73$.

The second part of this question is to identify the optimal number of principal components necessary for undertaking principal components regression (PCR). To do this we consider the standardised (centered) data $\mathbf{X}$ and $\vec{y}$. We can perform a singular value decomposition such that $\mathbf{X} = \boldsymbol{UDV'}$ where the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are both orthonormal sets of vectors denoting the left and right singular vectors of $\mathbf{X}$ respectively and $\mathbf{D}$ denotes diagonal matrix containing the non-negative singular values. Now define the coordinates of the input variables along the $j-th$ axis as $z_j = \boldsymbol{X}v_j$ representing the principal components of $\mathbf{X}$. Thus, $\sigma_{z_j} = \frac{d_j^2}{N}$, we can now calculate the cumulative sum of variation explained by each component seen in Table 1.

Component	1	2	3	4	5	6	7
% Var. Explained	42.663	65.399	77.987	86.678	93.170	97.510	100.000

Table 1: Percentage Variance Explained by Principal Components for PCR.

From Table 1 we can determine that $\approx 93\%$ would represent an adequate amount of variance explained, as such we could choose 5 components and have reduced the dimensionality of our problem. Figure 1b shows the LOOCV results for PCR which shows that all components (OLS) is best with a CV error of 0.409.



(a) LOOCV for Ridge

(b) LOOCV for Principal Components

Figure 1: Leave-one-out Cross-Validation Error for Ridge Regression Complexity Parameter and Optimal no. of Principal Components for PCR.

b) Regression Comparisons

In this question I am concerned with comparing OLS, ridge, principal component and lasso regression results. I have already implemented ridge regression above and setting the penalty parameter $\lambda = 0$ yields OLS results. Thus, here I must carry out principal component regression (PCR) and lasso regression.

Principal Component Regression

Now let γ denote the vector of estimated regression coefficients obtained by OLS regression of the response vector $\vec{y}$ on the data matrix $\boldsymbol{Z} = \boldsymbol{XV}$ such that $\hat{\vec{\gamma}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\vec{y}$. Our regression coefficients now become $\hat{\vec{\beta}}_{ols} = \boldsymbol{V}\hat{\vec{\gamma}}$ with predictions $\hat{\vec{f}}_{ols}(\boldsymbol{X}) = \boldsymbol{X}\hat{\vec{\beta}}_{ols}$ [1].

[1] We can denote $\boldsymbol{Z}_k = \boldsymbol{XV}_k$ to be the reduced form for the optimal number of principal components when undertaking principal components regression.

Lasso Regression

Lasso regression imposes a different penalty on the residual sum of squares to shrink the regression coefficients toward zero given by

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \quad [1]. \tag{5}$$

We notice that the lasso uses an $L1$ penalty whilst ridge regression uses an $L2$ penalty. While there is no closed form expression for $\hat{\beta}_{lasso}$ we can code an optimisation algorithm for the objective in Equation 5 and use a LOOCV procedure to find the value of the penalty parameter λ that minimises the CV error.

Comparison

Table 2 shows the results of LOOCV for ridge and lasso techniques. OLS and PCR do not suffer from the problem of choosing this parameter, where cross-validation is generally accepted as a valid method it has it's shortcomings making it a weakness of ridge and lasso. PCR does however suffer from choosing an optimal number of components (this is done using LOOCV but can also be carried out using variance explained and scree plots). For lasso we optimise our beta parameters for a given lambda and then change lambda to give different optimal beta's. The regression coefficients associated with the lambda that minimises CV error are chosen as optimal. We also note that setting complexity parameter to zero in both cases returns the OLS estimate and where the number of principal components M is equal to the number of predictions p principal component regression obtains the same result as OLS as the principal components span the columns space of $\mathbf{X}$ [1]. The results show that PCR (with $M = 6$) optimises the in-sample MSE whilst ridge and lasso perform slightly worse, however, the performance of lasso is sensitive to changes in the initial beta estimates and often converges to the OLS solution with $\lambda = 0$. The results imply no discernible difference in performance between the methods. The similar performance of the methods is likely due to the optimal values of λ being similar, as such to choose an optimal method for this small dataset and without a test set is not necessarily ideal.

Method	λ	CV-Error	MSE
OLS			0.333
Ridge	2.18	0.406	0.335
PCR_M		0.409	0.333
Lasso	1.64	0.340	0.340
Bayes	1.97		0.335

Table 2: Comparison of Results for Different Regression Procedures.

c) Shrinkage in Ridge Regression

Consider the singular value decomposition $\boldsymbol{X} = \boldsymbol{UDV'}$, if we construct our $\vec{\hat{\beta}}$ estimate from normal OLS such that $\vec{\hat{\beta}} = (\boldsymbol{X'X})^{-1}\boldsymbol{X'}\vec{y}$ we can substitute in the singular value decomposition

to obtain

$$\vec{\hat{\beta}} = ((\boldsymbol{UDV}')'\boldsymbol{UDV}')^{-1}(\boldsymbol{UDV}')'\vec{y} = \boldsymbol{VD}^{-1}\boldsymbol{U}'\vec{y} \tag{6}$$

$$= \sum_{j=1}^{p} \frac{1}{d_j} \langle\ \vec{u}_j, \vec{y}\ \rangle \vec{v}_j \tag{7}$$

We can solve similarly for the ridge regression coefficients to get

$$\boldsymbol{V}(D^2 + \lambda \boldsymbol{I})^{-1}\boldsymbol{DU}'\vec{y} \tag{8}$$

$$= \sum_{j=1}^{p} \frac{d_j}{d_j^2 + \lambda} \langle\ \vec{u}_j, \vec{y}\ \rangle \vec{v}_j \quad . \tag{9}$$

We can see that the shrinkage in ridge is a function of the singular values, this is illustrated in Figure 2 where smaller values of d_j imply greater shrinkage on the coefficients and the magnitude is effected by the choice of penalty term.

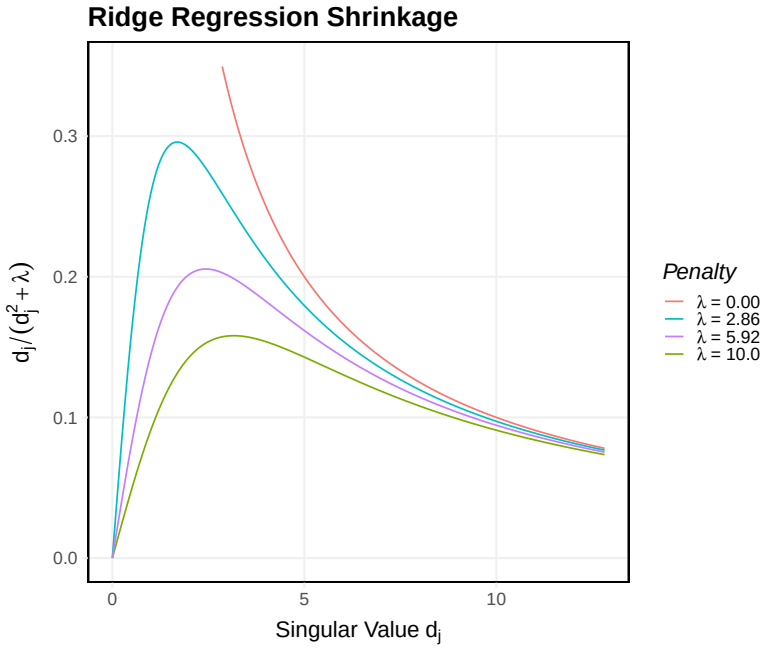


Figure 2: Shrinkage for Ridge Regression given Different Complexity Parameters.

d) Bayesian Ridge Regression

We can also form the above modelling approach in terms of a Bayesian framework. To do this we formulate prior distributions and derive appropriate posteriors, this formulation is very similar for the proceeding sections and as such is not carried out in full here, rather the results

are shown for comparison. First we note that the choice of prior will affect the value implied on the complexity parameter, $\left(\frac{\sigma^2}{\tau^2} \equiv \lambda\right)$, shown in Figure 3. Clearly the uninformative prior (prior 1) has too great a variance, whilst prior 2 seems to imply $\lambda \approx 1.9$ "close" to our optimal λ_{ridge} and λ_{lasso}.

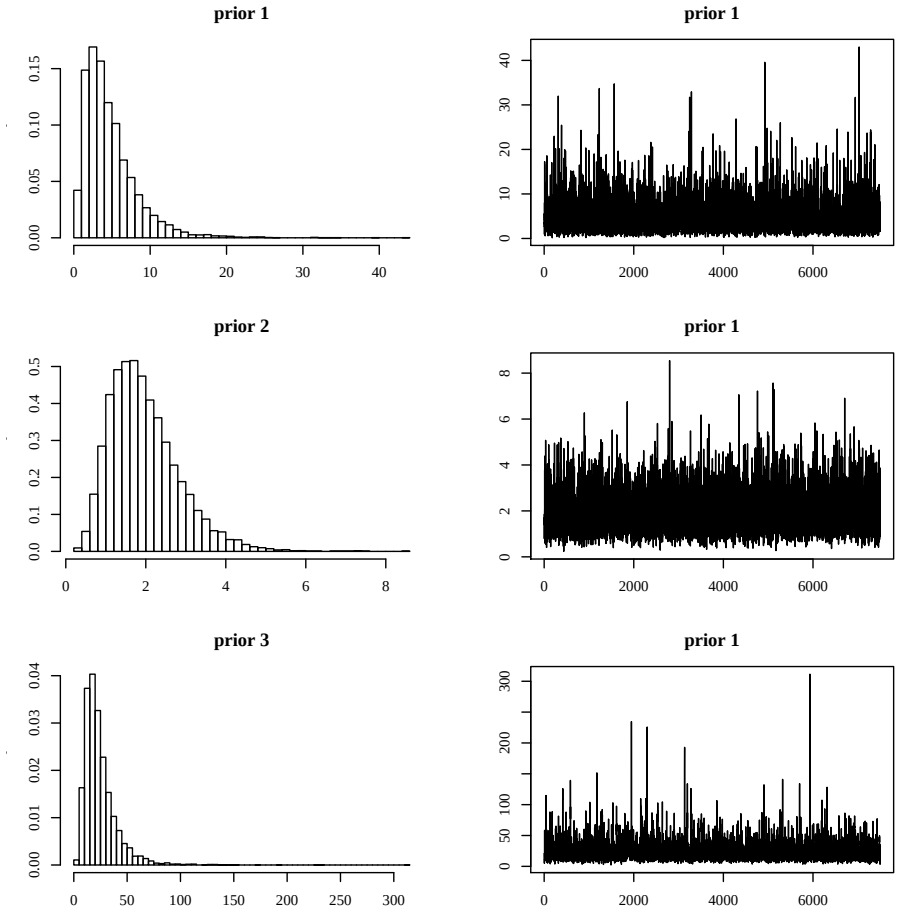
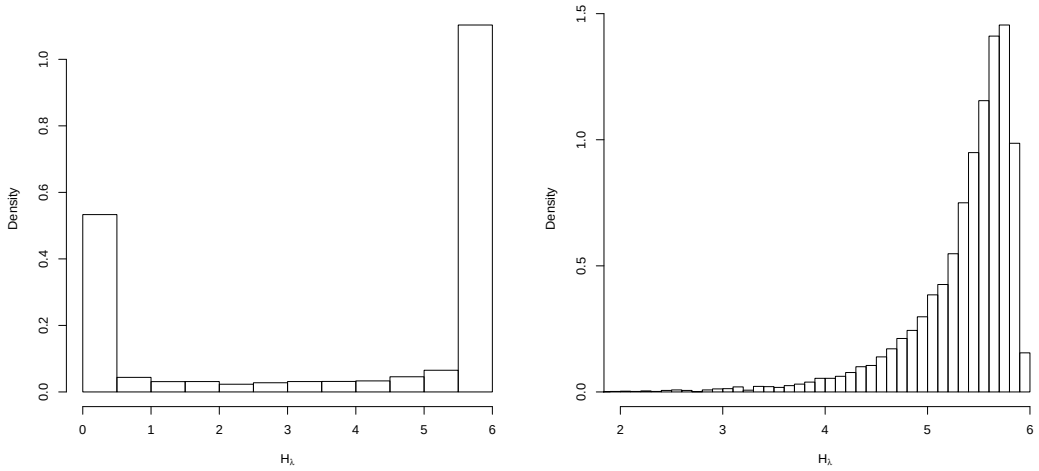


Figure 3: Effect of Prior on the Posterior Distribution for λ.

We can further examine the resulting affect of our prior choice on the degrees of freedom for the penalty parameter. We can see from Figure 4 that before tuning our model implies a bi-modal distribution for the degrees of freedom, this is nonsensical as we cannot have a model that is fully specified and null, if we wanted a fully smooth curve we would want something on the right (higher degrees of freedom) and if we want a linear representation (horizontal line) we would want low degrees of freedom (to the left). Thus, we tune our priors to imply a distribution that is left-skewed in the hope that this model will generalise well without over-fitting as seen in Figure 4b.



(a) Before Tuning

(b) After Tuning

Figure 4: Effect of Prior on the Degrees of Freedom for λ.

We can take the prior distribution settings and substitute them in to the Gibbs sampler to obtain Figure 3 (prior 2) from which we take the mean to get our optimal λ_{bayes} which is then substituted into the ridge regression formulae to generate a preditions and an in-sample MSE as per Table 2.

Question 2

In this question we consider a Bayesian regression approach to a paper modelling the log_2 gametocyte density present in patients over time. The dataset contains 50 patients observed over 7 time points separated into two distinct groups of 25 patients each. In 2a and 2b we will undertake Gibb's sampling to sample from the appropriate posterior distributions, a process that iteratively updates the parameters until a stationary distribution is obtained (this is when we assume that the stationary distribution is the appropraite posterior distribution).

a) Modelling Profiles of One Group

Here we consider patients in group 1 only, we aim to model

$$y_{i,j} = \beta_0 + \beta_1 t_j + \sum_{k=1}^{L^*} \tilde{\phi}_k(|t_j - \xi_k^{(1)}) + e_{i,j} \quad \forall\ j$$

where $\xi_k^{(1)}$ represents the knot vector for group 1, t_j represents time point j and L^* represents the number of knots such that we have $L = L^* + 2(\beta_0$ and $\beta_1)$ parameters. From bayesian statistics we know that to acquire the necessary posterior distributions we need to specify a likelihood $\mathrm{L}(y_{ij}|\cdot)$ and prior distributions $\pi(\cdot)$ for the parameters.

i) Posterior Specification and Gibbs Sampling

In this question we are given the necessary priors and a distribution for y to obtain the likelihood and can thus find the posteriors to be given by;

$$p(\phi|\cdot, y) = \pi(\phi)\mathrm{L}(y_{ij}|\cdot)$$
$$p(\phi|\cdot, y) \sim \mathcal{MVN}(\mu_0, \Sigma_0)$$

With

$$\Sigma_0 = \left(\frac{1}{\sigma_\epsilon^2}\boldsymbol{X'X} + \frac{1}{\sigma_\phi^2}\boldsymbol{D}\right)^{-1} \quad ; \quad \mu_0 = \Sigma_0 \frac{1}{\sigma_\epsilon^2}\boldsymbol{X'}\vec{y}$$

And the posterior for the ϕ variance is

$$p(\sigma_\phi^2|\cdot, y) = \pi(\sigma_\phi^2)\pi(\phi|\sigma_\phi^2)$$
$$p(\sigma_\phi^2|\cdot, y) \sim \mathcal{IG}(a, b)$$

With

$$a = \left(i_3 + \frac{L^*}{2}\right) \quad ; \quad b = \left(i_4 + \frac{1}{2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right)$$

Here the posterior is not dependent on the likelihood as σ_ϕ^2 is not directly estimated form the data but rather is hierarchical to the estimation of ϕ. We also need a posterior for the error variance calculated to be,

$$p(\sigma_\epsilon^2|\cdot, y) = \pi(\sigma_\epsilon^2)\mathrm{L}(y_{ij}|\cdot)$$
$$p(\sigma_\epsilon^2|\cdot, y) \sim \mathcal{IG}(a, b)$$

With

$$a = \left(i_1 + \frac{J}{2}\right) \quad ; \quad b = \left(i_2 + \frac{1}{2}(\vec{y} - \boldsymbol{X}\vec{\phi})'(\vec{y} - \boldsymbol{X}\vec{\phi})\right)$$

We can first examine the choice of $i_j,\ j = \{1, 2, 3, 4\}$. The specification for the priors impact the prior distribution for our degrees of freedom ($tr(\boldsymbol{H})$). We can implement a search for parameters i_j (which affect Σ_0 in Equation 10) and generate a histogram of the degrees of freedom to ensure our prior choice is sensible.

$$\boldsymbol{H} = \frac{1}{\sigma_\epsilon^2}\boldsymbol{X}\Sigma_0\boldsymbol{X'} \tag{10}$$
$$edf = tr(\boldsymbol{H}) \tag{11}$$

The resulting choice for the priors is depicted in Figure 5, It is important to note that 5 is the maximum number of degrees of freedom representing OLS (most wiggly) whilst 0 degrees of freedom represent a straight line. The distribution in Figure 5 is ideal as it is a smoother representation of our data (but not too smooth) implying that the model should generalise well.

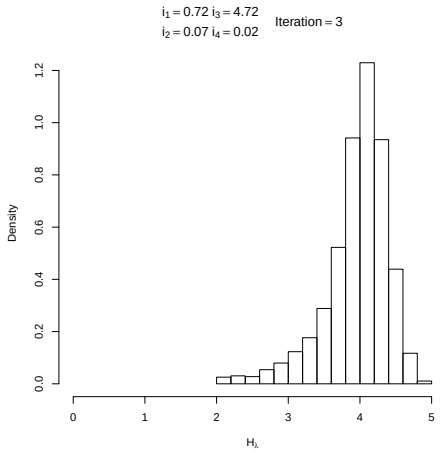
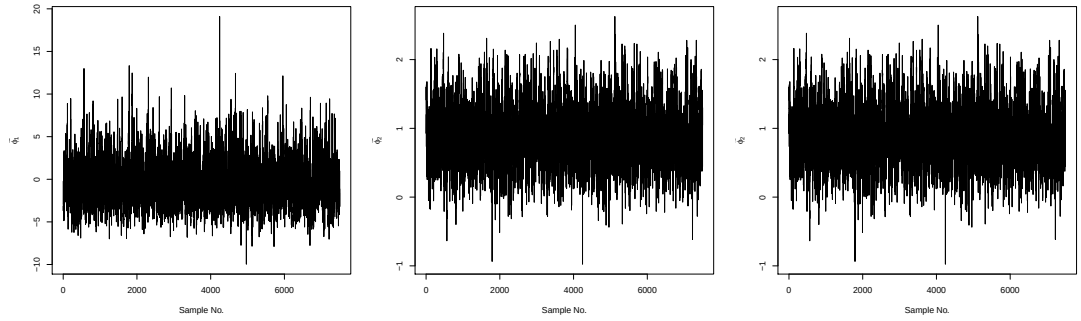


Figure 5: Distribution for the Effective Degrees of Freedom

ii) Fitting, Credibility Intervals and Visual Comparison

After obtaining the posterior distributions above we can undertake Gibbs sampling to sample from the appropriate posterior distributions. In this question we will assume that each individual person has their own $\beta_0, \beta_1, \tilde{\phi}_1, \tilde{\phi}_2$ and $\tilde{\phi}_3$ such that our results should be a prediction matrix that is 25 by 7. We can plot the posterior samples after burning half the samples, Figure 6 shows relatively stationary distributions of the $\tilde{\phi}_i$ parameters. Stationarity allows us to assume that we are correctly sampling from the true posterior distribution [2]. Once we obtain 7500 sampled values for $\sigma_\epsilon^2, \sigma_\phi^2$ and $\vec{\tilde{\phi}}_i$ from the assumed posterior distribution we can take the means of each parameter to obtain estimates for their true values. From this we can obtain 25 predicted profiles by taking $\hat{\vec{y}} = \boldsymbol{X}\vec{\tilde{\phi}}$, where each profile has its own estimates for ϕ.



(a) Posterior $\tilde{\phi}_1$ Samples (b) Posterior $\tilde{\phi}_2$ Samples (c) Posterior $\tilde{\phi}_3$ Samples

Figure 6: Post Burn-in $\tilde{\phi}_i$ Samples for the First Profile

[2] These distributions are obtained from $\sigma_\epsilon(i_1, i_2)$ and $\sigma_\phi(i_3, i_4)$ inverse gamma distributions with the hyperparamters all set to 0.001

An advantage of bayesian methods is the construction of credibility intervals where we can summarize the prediction uncertainty by giving a range of values on the posterior probability distribution that includes 95% of the probability - this is called a "95% credibility interval". To generate these Bayesian credibility intervals we first obtain $\hat{y_{i,j}}$ predictions for each of our post burn-in samples (thus we would have $\hat{y}_{i,j}$ for $i = 1, 2, \ldots, 25$; $j = 1, 2, \ldots, 7500$). We can then use R's quantile() function to produce sample quantiles corresponding to given probabilities (2.5% and 97.5% for a 95% interval for the fitted profile). The function treats the smallest observation as corresponds to a probability of 0 and the largest to a probability of 1. The resulting intervals are plotted for a random sample of 4 profiles and are depicted by the dashed lines in Figure 7. The intervals are visually fairly tight representing a visually good prediction, this is supported by the close superimposition of "true" and "predicted" response variable values in each of the four profiles.

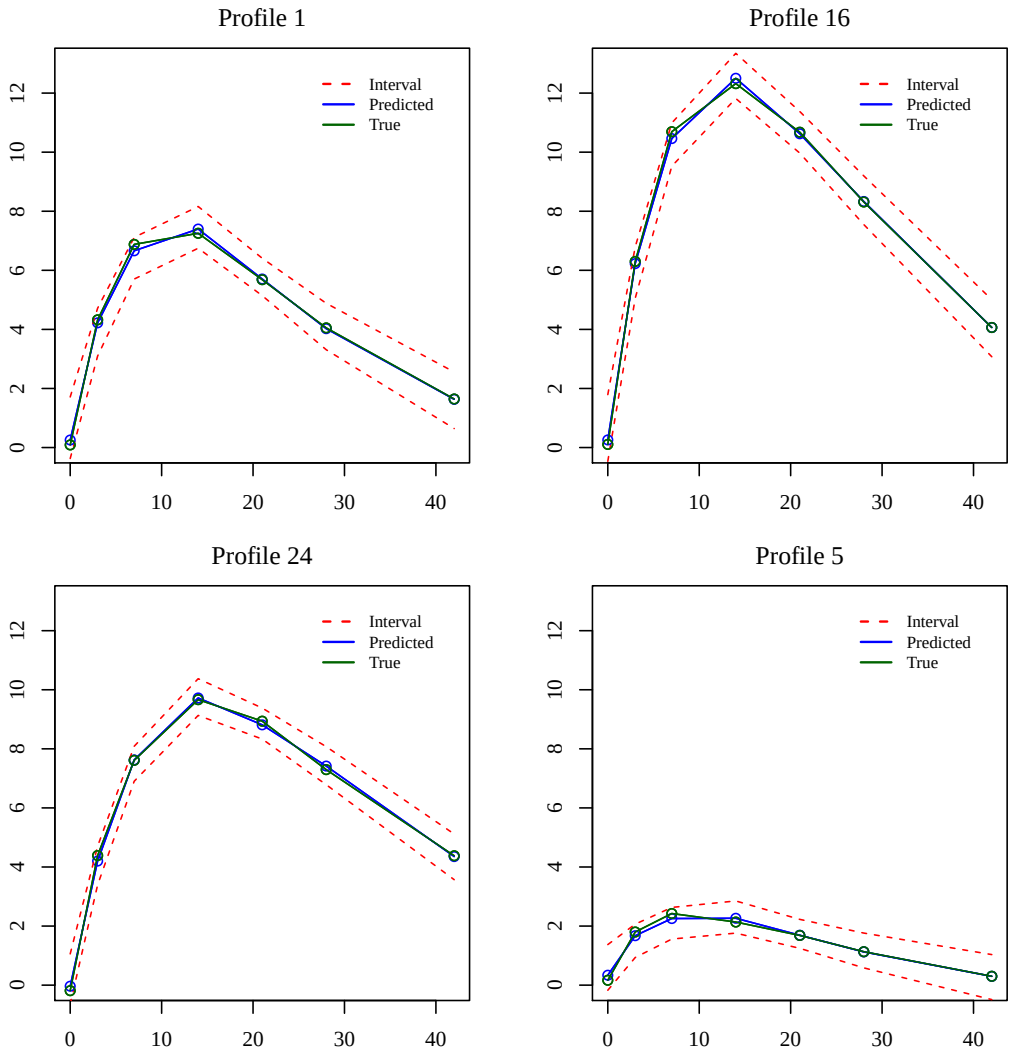


Figure 7: Bayesian Credibility Intervals for a Sample of Profiles

b) Modelling Profiles for Both Groups

Here we consider patients in both groups, such that the model is now;

$$y_{i,j} = \beta_0 + \beta_1 t_j + \beta_2 group_i + \sum_{k=1}^{L^*} \tilde{\phi}_k(|t_j - \xi_k^{(1)}) + e_{i,j} \quad \forall\ j,\ i = 1, \ldots, 25$$

$$y_{i,j} = \beta_0 + \beta_1 t_j + \beta_2 group_i + \sum_{k=1}^{L^*} \tilde{\phi}_k(|t_j - \xi_k^{(2)}) + e_{i,j} \quad \forall\ j,\ i = 26, \ldots, 50$$

The difference now is that we have "group" as an indicator variable which is coded 1 for group 1 and 0 for group 2. To model this we are going to assume that the β_0, β_1 and β_2 are common to all individuals, whilst each individual has a distinct $\tilde{\phi}_i$. This allows us to construct one big design matrix.

$$\overbrace{\begin{bmatrix} X^{(1)} & Z^{(1)} & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ X^{(1)} & 0 & Z^{(2)} & \ldots & \ldots & \ldots & \ldots & 0 \\ \vdots & \vdots & & \ddots & \ldots & \ldots & \ldots & 0 \\ X^{(1)} & 0 & \ldots & 0 & Z^{(25)} & \ldots & \ldots & 0 \\ X^{(2)} & 0 & \ldots & 0 & 0 & Z^{(26)} & \ldots & 0 \\ \vdots & \vdots & & 0 & 0 & 0 & \ddots & 0 \\ X^{(2)} & 0 & \ldots & 0 & 0 & 0 & \ldots & Z^{(50)} \end{bmatrix}}^{{}_{350}\boldsymbol{X}_{153}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \tilde{\phi}_{1,1} \\ \vdots \\ \tilde{\phi}_{1,3} \\ \tilde{\phi}_{2,1} \\ \vdots \\ \tilde{\phi}_{2,3} \\ \vdots \\ \tilde{\phi}_{50,3} \end{bmatrix}}_{{}_{153}\vec{\phi}_1} = \overbrace{\begin{bmatrix} y_{1,1} \\ \vdots \\ y_{1,7} \\ y_{2,1} \\ \vdots \\ y_{2,7} \\ \vdots \\ y_{50,7} \end{bmatrix}}^{{}_{350}\vec{y}_1}$$

i) Posterior Distributions

The posteriors are constructed in a similar manner as in the previous question, however, we now utilise the big design matrix as well as the long ϕ and y vectors in the following posteriors;

$$p(\phi|\cdot, y) = \pi(\phi)\mathrm{L}(y_{ij}|\cdot)$$
$$p(\phi|\cdot, y) \sim \mathcal{MVN}(\mu_0, \Sigma_0)$$

With

$$\Sigma_0 = \left(\frac{1}{\sigma_\epsilon^2}\boldsymbol{X}'\boldsymbol{X} + \frac{1}{\sigma_{\tilde{\phi}}^2}\boldsymbol{D}\right)^{-1} \quad ; \quad \mu_0 = \Sigma_0 \frac{1}{\sigma_\epsilon^2}\boldsymbol{X}'\vec{y}$$

And the posterior for the ϕ variance is

$$p(\sigma_{\tilde{\phi}}^2|\cdot, y) = \pi(\sigma_{\tilde{\phi}}^2)\pi(\phi|\sigma_{\tilde{\phi}}^2)$$
$$p(\sigma_{\tilde{\phi}}^2|\cdot, y) \sim \mathcal{IG}(a, b)$$

With

$$a = \left(i_3 + \frac{50L^* + 3}{2}\right) \quad ; \quad b = \left(i_4 + \frac{1}{2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right)$$

Lastly the posterior for the error variance is calculated to be,

$$p(\sigma_\epsilon^2|\cdot, y) = \pi(\sigma_\epsilon^2)\mathrm{L}(y_{ij}|\cdot)$$
$$p(\sigma_\epsilon^2|\cdot, y) \sim \mathcal{IG}(a, b)$$

With

$$a = \left(i_1 + \frac{50J}{2}\right) \quad ; \quad b = \left(i_2 + \frac{1}{2}(\vec{y} - \boldsymbol{X}\vec{\phi})'(\vec{y} - \boldsymbol{X}\vec{\phi})\right)$$

ii) Gibbs Sampling and Tuning

Unlike the previous question tuning for the i_j values should be relatively innefective unless we specify large values of i_j as the shape and rate parameters all sum with a large value. The large sample means that tuning should theoretically be less effective and the tuning process is more computationally intensive, thus the potential performance gain is not worth the required effort to tune. As such we select the parameters to be equivalent to those found in the previous question. We can again see a visually good fit, presuming a common β vector for all individuals has the model fitting as well as before.

iii) Model Fit

When comparing Figure 7 with Figure 8 that the fit for profile 1 appears slightly worse in the latter, however, visual checking of all profiles would be unnecessarily intensive. We cannot necessarily compare models via MSE as the tuning is largely subjective in question 2a yet yields high variability in results. We also note that specifying an uninformative set of priors resulting in a model that has a very low MSE (this is probably due to the degrees of freedom sampling occurring majority at the highest MSE possible). As such tuning the model resulted in a lower MSE.

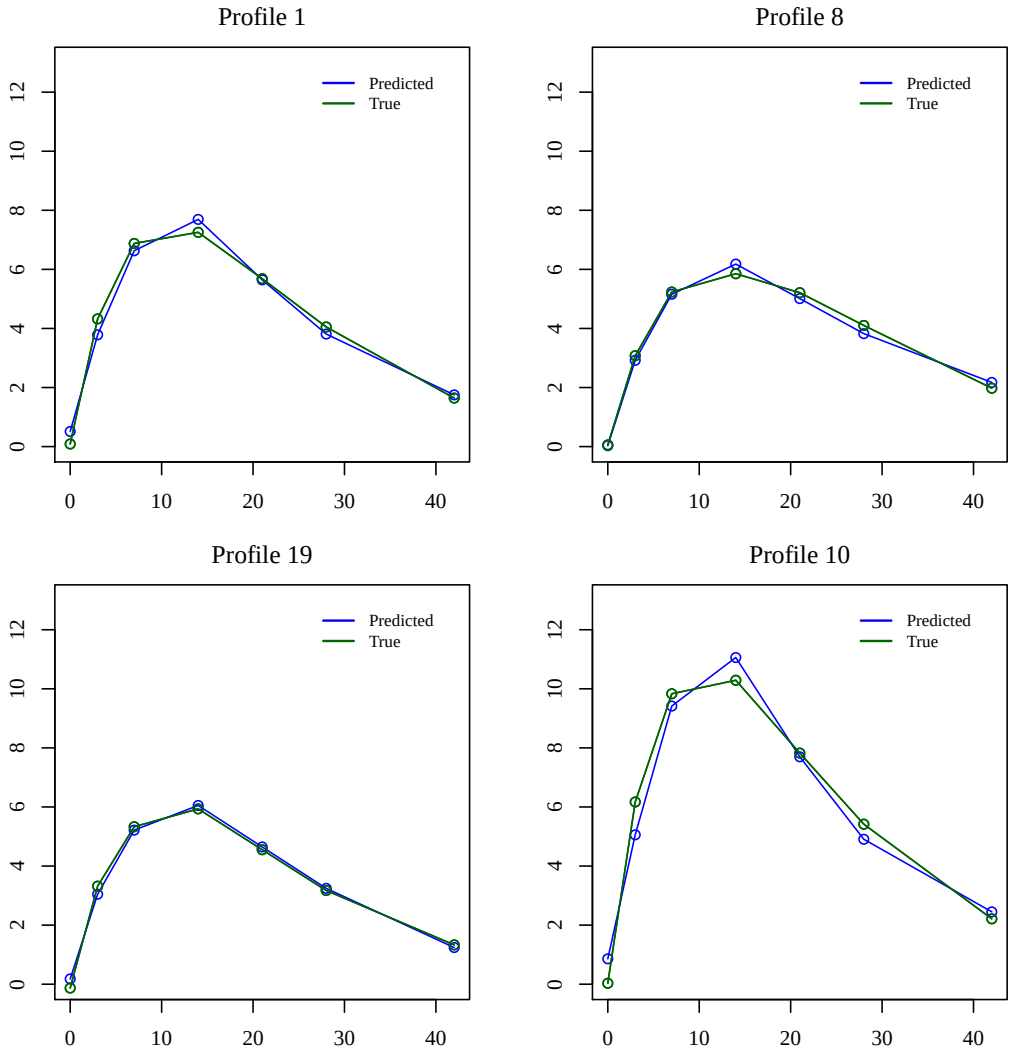


Figure 8: Fits for a Sample of Profiles

c) Modelling for Auto-Correlation in the Data

To account for potential auto-correlation we can let

$$e_{i,j} \sim \mathcal{MVN}(\vec{0}, \boldsymbol{V})$$

where $\mathbf{V}$ is a covariance matrix. Now let

$$L(\vec{y}|\cdot) \propto |\boldsymbol{V}|^{-\frac{N}{2}} \exp\left[\frac{1}{2}\sum_i (y_i - \boldsymbol{X}\vec{\phi}\,)'\boldsymbol{V}^{-1}(y_i - \boldsymbol{X}\vec{\phi}\,)\right]$$

A common conjugate prior for the covariance matrix is known to be the inverse Wishardt $f(\boldsymbol{V}, \boldsymbol{\Psi}, b) = \mathcal{IW}(\boldsymbol{\Psi}, b)$ distribution. As $\boldsymbol{\Psi}$ and b are hyper-parameters we can simply set them

to be $\boldsymbol{I}$ and b respectively. Now we have a posterior for the covariance matrix $\boldsymbol{V}$ as

$$\begin{aligned}
\pi(\boldsymbol{V}|\cdot) &\propto \pi(\boldsymbol{V})L(\vec{y}|\cdot) \\
&\propto |\boldsymbol{V}|^{-\left(\frac{b+p+1}{2}\right)} \exp\left[-\frac{1}{2}tr(\boldsymbol{V}^{-1})\right] |\boldsymbol{V}|^{-\left(\frac{N}{2}\right)} \exp\left[\frac{1}{2}tr\left\{\sum_i \left(y_i - \boldsymbol{X}\vec{\phi}\right)' \boldsymbol{V}^{-1}\left(y_i - \boldsymbol{X}\vec{\phi}\right)\right\}\right] \\
&\propto |\boldsymbol{V}|^{-\left(\frac{N+b+p+1}{2}\right)} \exp\left[-\frac{1}{2}tr\left\{\boldsymbol{V}^{-1} + \boldsymbol{V}^{-1}\sum_i \left(y_i - \boldsymbol{X}\vec{\phi}\right)\left(y_i - \boldsymbol{X}\vec{\phi}\right)'\right\}\right] \\
&\propto |\boldsymbol{V}|^{-\left(\frac{N+b+p+1}{2}\right)} \exp\left[-\frac{1}{2}tr\left\{\boldsymbol{V}^{-1}(\boldsymbol{I} + \boldsymbol{S}_{\boldsymbol{X}\boldsymbol{\phi}})\right\}\right] \\
&\sim \mathcal{IW}(\boldsymbol{I} + \boldsymbol{S}_{\boldsymbol{X}\boldsymbol{\phi}}, N + b)
\end{aligned}$$

This conjugacy allows easy incorporation into Markov chain Monte Carlo (MCMC) approaches based on Gibbs sampling. We also need to update our ϕ parameter such that

$$\begin{aligned}
\pi(\phi_i|\cdot) &\propto \pi(\phi_i)L(\vec{y}|\cdot) \\
&\propto (\sigma_\phi^2)^{\frac{L^*}{2}} \exp\left[-\frac{1}{2}\vec{\phi_i}'\boldsymbol{D}\vec{\phi_i}\right] |\boldsymbol{V}|^{-\left(\frac{N}{2}\right)} \exp\left[\frac{1}{2}\sum_i \left(y_i - \boldsymbol{X}\vec{\phi_i}\right)' \boldsymbol{V}^{-1}\left(y_i - \boldsymbol{X}\vec{\phi_i}\right)\right] \\
&\propto \exp\left[-\frac{1}{2}\left\{\sum_i \left(y_i - \boldsymbol{X}\vec{\phi_i}\right)' \boldsymbol{V}^{-1}\left(y_i - \boldsymbol{X}\vec{\phi_i}\right) + \frac{1}{\sigma_\phi^2}\vec{\phi_i}'\boldsymbol{D}\vec{\phi_i}\right\}\right] \\
&\propto \exp\left[-\frac{1}{2}\left\{\vec{\phi_i}'\left(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X} + \frac{1}{\sigma_\phi^2}\right)\vec{\phi_i} - 2\vec{\phi_i}'\boldsymbol{X}'\boldsymbol{V}^{-1}\vec{y} + \dots\right\}\right] \\
&\sim \mathcal{MVN}(\boldsymbol{\Sigma}_0,\ \vec{\mu_0})
\end{aligned}$$

with

$$\Sigma_0 = \left(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X} + \frac{1}{\sigma_\phi^2}\right)^{-1}, \quad \mu_0 = \boldsymbol{\Sigma}_0\boldsymbol{X}'\boldsymbol{V}^{-1}\vec{y}$$

Lastly we can get

$$\begin{aligned}
\pi(\sigma_\phi^2|\cdot) &\propto \pi(\sigma_\phi^2)\pi(\phi_i|\cdot) \\
&\propto (\sigma_\phi^2)^{-(i_3+1)} \exp\left[\frac{i_4}{\sigma_\phi^2}\right] (\sigma_\phi^2)^{-\frac{50L^*}{2}} \exp\left[-\frac{1}{\sigma_\phi^2}\vec{\phi_i}'\boldsymbol{D}\vec{\phi_i}\right] \\
&\propto (\sigma_\phi^2)^{-(i_3+\frac{50L^*}{2}+1)} \exp\left[-\frac{1}{\sigma_\phi^2}\left(i_4 + \vec{\phi_i}'\boldsymbol{D}\vec{\phi_i}\right)\right] \\
&\sim \mathcal{IG}\left(i_3 + \frac{50L^*}{2}, i_4 + \vec{\phi_i}'\boldsymbol{D}\vec{\phi_i}\right)
\end{aligned}$$

- END -

References

[1] Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[2] from the book's webpage, M., port, R., and packaging by Kjetil B Halvorsen. *ElemStatLearn: Data Sets, Functions and Examples from the Book: "The Elements of Statistical Learning, Data Mining, Inference, and Prediction" by Trevor Hastie, Robert Tibshirani and Jerome Friedman*, 2015. R package version 2015.6.26.

Appendix

Derivations

$$
\begin{aligned}
p(\phi|\cdot, y) &\propto \pi(\phi)\mathrm{L}(y_{ij}|\cdot) \\
&\propto \exp\left[-\frac{1}{2\sigma_\phi^2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right]\exp\left[-\frac{1}{2\sigma_\epsilon^2}\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)'\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)\right] \\
&\propto \exp\left[-\frac{1}{2}\left\{\frac{1}{\sigma_\phi^2}\vec{\phi}'\boldsymbol{D}\vec{\phi}+\frac{1}{\sigma_\epsilon^2}\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)'\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)\right\}\right] \\
&\propto \exp\left[-\frac{1}{2}\left\{\frac{1}{\sigma_\phi^2}\vec{\phi}'\boldsymbol{D}\vec{\phi}+\frac{1}{\sigma_\epsilon^2}\left(\vec{\phi}'\boldsymbol{X}'\boldsymbol{X}\vec{\phi}-2\vec{y}'\boldsymbol{X}\vec{\phi}+\dots\right)\right\}\right] \\
&\propto \exp\left[\vec{\phi}'\left(\frac{1}{\sigma_\phi^2}\boldsymbol{D}+\frac{1}{\sigma_\epsilon^2}\boldsymbol{X}'\boldsymbol{X}\right)\vec{\phi}-\frac{2}{\sigma_\epsilon^2}\vec{y}'\boldsymbol{X}\vec{\phi}\right] \\
p(\phi|\cdot, y) &\sim \mathcal{MVN}(\mu_0, \Sigma_0)
\end{aligned}
$$

With

$$
\Sigma_0 = \left(\frac{1}{\sigma_\epsilon^2}\boldsymbol{X}'\boldsymbol{X}+\frac{1}{\sigma_\phi^2}\boldsymbol{D}\right)^{-1} \quad ; \quad \mu_0 = \Sigma_0\frac{1}{\sigma_\epsilon^2}\boldsymbol{X}'\vec{y}
$$

We can get the posterior for ϕ variance

$$
\begin{aligned}
p(\sigma_\phi^2|\cdot, y) &\propto \pi(\sigma_\phi^2)\pi(\phi|\sigma_\phi^2) \\
&\propto (\sigma_\phi^2)^{-(i_3+1)}(\sigma_\phi^2)^{150/3}\exp\left[-\frac{1}{2\sigma_\phi^2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right]\exp\left[-\frac{i_4}{\sigma_\phi^2}\right] \\
&\propto (\sigma_\phi^2)^{-(i_3+\frac{150}{2})}\exp\left[-\frac{1}{\sigma_\phi^2}\left(i_4+\frac{1}{2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right)\right] \\
p(\sigma_\phi^2|\cdot, y) &\sim \mathcal{IG}(a, b)
\end{aligned}
$$

With

$$
a = \left(i_3+\frac{150}{2}\right) \quad ; \quad b = \left(i_4+\frac{1}{2}\vec{\phi}'\boldsymbol{D}\vec{\phi}\right)
$$

We also need a posterior for the error variance calculated to be,

$$
\begin{aligned}
p(\sigma_\epsilon^2|\cdot, y) &\propto \pi(\sigma_\epsilon^2)\mathrm{L}(y_{ij}|\cdot) \\
&\propto (\sigma_\epsilon^2)^{-(i_1+1)}(\sigma_\epsilon^2)^{-350/2}\exp\left[-\frac{1}{2\sigma_\epsilon^2}\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)'\left(\vec{y}-\boldsymbol{X}\vec{\phi}\right)\right]\exp\left[-\frac{i_2}{\sigma_\epsilon^2}\right] \\
p(\sigma_\epsilon^2|\cdot, y) &\sim \mathcal{IG}(a, b)
\end{aligned}
$$

With

$$
a = \left(i_1+\frac{350}{2}\right) \quad ; \quad b = \left(i_2+\frac{1}{2}(\vec{y}-\boldsymbol{X}\vec{\phi})'(\vec{y}-\boldsymbol{X}\vec{\phi})\right)
$$

R Code

Listing 1: Question 1 Code

```
1
2  # UCT Assignment
3  # Author: Julian Albert
4  # Date:   26/03/2019
5
6  rm(list = ls()); dev.off()
7  # Assignment prelim script
8  source("../../../../PrelimScript/assignment_prelim_script.R")
9  ## 0. Load required packages
10 p_load(ElemStatLearn, corrplot, MASS)
11
12 # Question 1 ----
13
14 ## 1.1 Read in the Data and OLS ----
15 set.seed(123)
16 dat_prostate <- get('prostate', asNamespace('ElemStatLearn'))
17 dat_train <- dat_prostate %>% dplyr::filter(train == TRUE) %>%
18   dplyr::select(train)
19 dat_prostate_wo_train <- dat_prostate %>% dplyr::filter(train == TRUE) %>%
20    dplyr::select(-c(train, svi, gleason))
21 dat_std_prostate <- apply(dat_prostate_wo_train, 2,
22                            function(x) (x - mean(x))/(sd(x))) %>%
23   cbind(., dat_train)
24
25 ## 1.1.1 Standardise the Data then split
26 train_dat <- dat_std_prostate # training predictors
27
28 X <- train_dat %>%  dplyr::select(-c(lpsa, train)) %>% as.matrix()
29 Y <- train_dat %>%  dplyr::select(lpsa) %>% as.matrix()
30 n <- NROW(X) ; p <- NCOL(X)
31
32 ### some quick EDA
33 dat_corr <- cor(train_dat %>%  dplyr::select(-train))
34 round(dat_corr*lower.tri(dat_corr), 3)
35
36 ## 1.1.2 OLS
37 ols.beta_hat <- solve(crossprod(X)) %*% t(X) %*% Y
38 ols.fhat <- X %*% ols.beta_hat %>% as.numeric()
39
40 ## 1.2 Ridge Regresion ----
41
42 func.cv_general <- function(data, lambda = NULL, method = 'Ridge',
43                             lasso.obj = NULL, lasso.pars = NULL)
44 {
45
46   # Define Parameters and Matrices/Vectors
47   data_shuffled <- sample_frac(data, 1L) %>%
48      dplyr::select(-train)
49   n <- NROW(data_shuffled); p <- NCOL(data_shuffled) - 1
50   I_mat <- diag(p); fhat <- numeric(n)
51
52   ### PCR Stuff
53   if(method != 'PCR') cv.error <- numeric(n) else{
54     cv.error <-  matrix(0, n, p) # X.svd <- svd(tmp.X_train)
55     U <- X.svd$u
56     D <- diag(X.svd$d) # D_{i} >= D_{i+1} forall i
57     V <- X.svd$v # principal components of X
58   }
59
60   # Cross-Validation
61   for(i in 1:n){
62     idx <- i
63     ### Training Data
64     tmp.X_train <- data_shuffled[-idx, ] %>%
65        dplyr::select(-lpsa) %>% as.matrix()
66     tmp.Y_train <- data_shuffled[-idx, ] %>%
67        dplyr::select(lpsa) %>% as.matrix()
68     ### Testing Data
69     tmp.X_test <- data_shuffled[idx, ] %>%
```

```
      dplyr::select(-lpsa) %>% as.matrix()
    tmp.Y_test <- data_shuffled[idx, ] %>%
      dplyr::select(lpsa) %>% as.matrix()

    if(method == 'Ridge'){
      ### Get Beta Coefficients
      beta_hat <- solve(crossprod(tmp.X_train) + lambda*I_mat) %*% t(tmp.X_train) %*% tmp.Y_train
      ### Predictions
      fhat[i] <- tmp.X_test %*% beta_hat
      cv.error[i] <- (fhat[i] - tmp.Y_test)^2
    }else if(method == 'Lasso'){
      lasso.obj_func <- match.fun(lasso.obj)
      init_par <- lasso.pars
      ### Get Beta Coefficients
      beta_hat <- optim(init_par, lasso.obj_func, lambda = lambda)$par
      ### Predictions
      fhat[i] <- tmp.X_test %*% beta_hat
      cv.error[i] <- (fhat[i] - tmp.Y_test)^2
    }else if(method == 'PCR'){
      for(j in 1:p){
        # wikipedia
        tmp.V <- V[, 1:j] %>% as.matrix()
        Z <- apply(tmp.V, 2, function(x) tmp.X_train %*% x)
        pcr.gamma <- solve(crossprod(Z)) %*% t(Z) %*% tmp.Y_train # ?
        beta_hat <- tmp.V %*% pcr.gamma
        fhat[j] <- tmp.X_test %*% beta_hat %>% as.numeric()
        cv.error[i, j] <- (fhat[j] - tmp.Y_test)^2
      }
    }

  }

  CV <- cv.error %>% as.matrix()
  return(colMeans(CV))
}

### Range of Lambda Values, calculate CVs, Plot
lambda_tries <- seq(0, 6, length.out = 12) %>% as.matrix()
ridge_cvs <- apply(lambda_tries, 1, function(x)
  func.cv_general(train_dat, x, 'Ridge'))
df_ridge <- tibble(lambda = lambda_tries, CV = ridge_cvs)

### Optimal Values
ridge.optimal_pair <- df_ridge %>% dplyr::filter(CV == min(CV))
ridge.optimal_lambda <- ridge.optimal_pair %>%  dplyr::select(lambda) %>% as.numeric()
ridge.optimal_CV <- ridge.optimal_pair %>%  dplyr::select(CV) %>% as.numeric()

setwd('../Figs')
pdf('CV_Ridge.pdf')
ggplot(df_ridge, aes(x = lambda, y = CV)) +
  geom_line(col = 'dodgerblue3') +
  geom_point(col = 'dodgerblue3') +
  geom_point(aes(x = ridge.optimal_lambda, y = ridge.optimal_CV),
             col = 'darkorange2', shape = 13, size = 4) +
  labs(title = bquote('CV Error for Different' ~ lambda ~ 'Values'),
       x = bquote(lambda)) +
  theme_university()
dev.off()

### Get Predictions for Ridge
I_mat <- diag(p)
ridge.beta_hat <- solve(crossprod(X) + ridge.optimal_lambda*I_mat) %*% t(X) %*% Y
ridge.fhat <- X %*% ridge.beta_hat %>% as.numeric()

## 1.3 Principal Component Regression ----

X.svd <- svd(X)
U <- X.svd$u
D <- diag(X.svd$d) # D_{i} >= D_{i+1} forall i
V <- X.svd$v # principal components of X

# plot(X.svd$d)
# cumsum((((X.svd$d)^2)/n / sum(((X.svd$d)^2)/n))*100)
```

```r
# plot(cumsum((((X.svd$d)^2)/n / sum(((X.svd$d)^2)/n))*100))

# Var. Explained
cumsum((((X.svd$d)^2)/n / sum(((X.svd$d)^2)/n))*100) # 5

df_plot_pc <- data.frame(x = 1:6, y = X.svd$d)

setwd('../Figs')
pdf('PCR_components.pdf')
ggplot(df_plot_pc, aes(x = x, y = y)) +
  geom_line(col = 'dodgerblue3') +
  geom_point(col = 'dodgerblue3') +
  geom_hline(yintercept = 5.476453, col = 'darkorange2', lty = 'dashed') +
  geom_vline(xintercept = 5, col = 'darkorange2', lty = 'dashed') +
  labs(title = 'Singular Values for PCR',
       x = 'Component', y = 'Singular Value') +
  theme_university()
dev.off()

### var explained
PCR_cvs <- func.cv_general(train_dat, method = 'PCR') # 7
df_plot_pc_cv <- data.frame(x = 1:6, y = PCR_cvs)

setwd('../Figs')
pdf('PCR_components_CV.pdf')
ggplot(df_plot_pc_cv, aes(x = x, y = y)) +
  geom_line(col = 'dodgerblue3') +
  geom_point(col = 'dodgerblue3') +
  labs(title = 'Cross-Validation for PCR',
       x = 'Component', y = 'CV Error') +
  theme_university()
dev.off()

# cross-validation
Z <- apply(V[, 1:6], 2, function(x) X %*% x)
pcr.gamma <- solve(crossprod(Z)) %*% t(Z) %*% Y # ?
pcr.beta_hat <- V[, 1:6] %*% pcr.gamma
pcr7.fhat <- X %*% pcr.beta_hat %>% as.numeric()

## 1.6 Lasso ----
lasso.obj_func <- function(pars, lambda)
{
  lambda <- lambda
  beta <- pars
  q <- 1

  tmp <- sum((Y - X %*% beta)^2)
  penalty <- lambda * sum(abs(beta)^q)
  lasso.obj <- tmp + penalty
  return(lasso.obj)
}

set.seed(123)
init_par <- runif(p, -5, 5)
lasso_cvs <- apply(lambda_tries, 1, function(x)
  func.cv_general(train_dat, x, 'Lasso', lasso.obj_func, init_par))
df_lasso <- tibble(lambda = lambda_tries, CV = lasso_cvs)

### Optimal Values
lasso.optimal_pair <- df_lasso %>% dplyr::filter(CV == min(CV))
lasso.optimal_lambda <- lasso.optimal_pair %>%  dplyr::select(lambda) %>% as.numeric()
lasso.optimal_CV <- lasso.optimal_pair %>%  dplyr::select(CV) %>% as.numeric()

lasso.beta_hat <- optim(init_par, lasso.obj_func, lambda = lasso.optimal_lambda)$par
lasso.fhat <- X %*% lasso.beta_hat %>% as.numeric()

## 1.7 Compare Results of OLS, Ridge, Lasso and PCR ----


# # bayes.optimal_lambda <- mean(rr2$sigma2/rr2$tau2)
# ### Get Predictions for Ridge
# I_mat <- diag(p)
# bayes.beta_hat <- solve(crossprod(X) + bayes.optimal_lambda*I_mat) %*% t(X) %*% Y
```

```r
# bayes.fhat <- X %*% bayes.beta_hat %>% as.numeric()


df_calc_MSE <- tibble(OLS = ols.fhat,
                      Ridge = ridge.fhat,
                      PCR = pcr7.fhat,
                      Lasso = lasso.fhat,
                      Bayes = bayes.fhat)

df_plot_pred <- df_calc_MSE %>%
  gather(Method, value) %>%
  mutate(x = rep(1:67, 5),
         y = rep(Y, 5),
         Method = factor(Method))

setwd('../Figs')
pdf('Regression_fits.pdf')
ggplot(df_plot_pred, aes(x = x)) +
  geom_point(aes(y = y)) +
  geom_line(aes(y = value, col = Method)) +
  labs(title = 'Regression Fits', x = '', y ='') +
  theme_university() +
  theme(legend.direction = 'vertical',
        legend.position = 'right',
        legend.key.width = unit(1.5, 'line'))
dev.off()

MSEs <- lapply(df_calc_MSE, function(x) mean((x-Y)^2)) %>%
  bind_cols()

## 1.7 Shrinkage ----

tmp.djs <- X.svd$d
tmp.djs_plot <- seq(0, max(tmp.djs), length.out = 1000)
lambda_tries_plot <- seq(0, 10, length.out = 50) %>% as.matrix()
tmp.shrunk <- apply(lambda_tries_plot, 1,
                    function(x) tmp.djs_plot/((tmp.djs_plot)^2 + x))

df_plot_shrink <- data.frame(lambda0 = tmp.shrunk[, 1],
                             lambda2.86 = tmp.shrunk[, 15],
                             lambda5.92 = tmp.shrunk[, 30],
                             lambda10 = tmp.shrunk[, 50])

df_plot_shrink <- df_plot_shrink %>% gather(Value, Coeff.) %>%
  mutate(x = rep(tmp.djs_plot, 4),
         Value = factor(Value))

setwd('../Figs')
pdf('shrinkage.pdf')
ggplot(df_plot_shrink, aes(x = x)) +
  geom_line(aes(y = Coeff.,
                col = Value)) +
  ylim(c(0, 0.35)) +
  scale_colour_discrete(name = "Penalty",
                        breaks = c("lambda0", "lambda2.86",
                                   "lambda5.92", "lambda10"),
                        labels = c(bquote(lambda ~ "= 0.00"),
                                   bquote(lambda ~ "= 2.86"),
                                   bquote(lambda ~ "= 5.92"),
                                   bquote(lambda ~ "= 10.0"))) +
  theme_university() +
  theme(legend.direction = 'vertical',
        legend.position = 'right',
        legend.key.width = unit(1, 'line')) +
  labs(title = 'Ridge Regression Shrinkage',
       x = bquote('Singular Value'~d[j]),
       y = bquote(d[j]/(d[j]^2 + lambda)))
dev.off()

## 1.8 Bayesian Ridge Regression ----

ridge.regression.bayes <-function(X, Y, D, s2, t2,
                                  i1 = 0.001, i2 = 0.001,
```

```r
                                i3 = 0.001, i4 = 0.001,
                                ndraws = 15000){
  n <- length(Y) #the sample size
  d <- NCOL(X)

  betas <- matrix(NA, ncol = floor(ndraws/2), nrow = d)
  yhat <- matrix(NA, n, floor(ndraws/2))
  sigma2 <- NULL
  tau2 <- NULL
  icounter <- 0

  #do the sampling here and only retain the second half of the samples
  for (i in 1:ndraws){
    #sample beta
    lambda <- s2/t2
    C <- solve(crossprod(X)/s2 + D/t2 )
    mean_beta <- C %*% crossprod(X, Y)/s2
    b <- mvrnorm(n = 1, mu = mean_beta, Sigma = C)

    #sample s2
    shape_s2 <- i1 + n/2
    rate_s2 <- 0.5*crossprod( Y - X%*%b ) + i2
    s2 <- 1/rgamma(1, shape = shape_s2, rate = rate_s2)

    #sample tau2
    shape_tau2 <- i3 + d/2
    rate_tau2 <- 0.5*t(b) %*% D %*% b + i4
    t2 <- 1/rgamma(1, shape = shape_tau2, rate = rate_tau2)

    if (i > floor(ndraws/2)){
      icounter <- icounter + 1
      betas[,icounter] <- b
      sigma2[icounter] <- s2
      tau2[icounter] <- t2
    }
  } #end of simulations

  list(betas = betas, sigma2 = sigma2,
       tau2 = tau2, yhat = yhat)
}

D <- diag(NCOL(X))

rr1 <- ridge.regression.bayes(X=X, Y= Y, D=D, s2=10, t2=100,
                              i1=0.0001, i2=0.0001, i3=0.0001, i4=0.0001)

rr2 <- ridge.regression.bayes(X=X, Y= Y, D=D, s2=10, t2=100,
                              i1=2, i2=11, i3=3.5, i4=2)

rr3 <- ridge.regression.bayes(X=X, Y= Y, D=D, s2=10, t2=100,
                              i1=2, i2=0.01, i3=6, i4=0.01)

rr4 <- ridge.regression.bayes(X=X, Y= Y, D=D, s2=10, t2=100,
                              i1=2, i2=0.01, i3=6, i4=0.01)

setwd('../Figs')
pdf('q1_prioronpost.pdf')
par(mfrow=c(3,2), family="serif", mai=c(0.4,0.4,0.4,0.4))
hist(rr1$sigma2/rr1$tau2, prob=TRUE, main="prior 1", xlab = expression(lambda), breaks=50)
plot(rr1$sigma2/rr1$tau2, type="l", ylab = expression(lambda), xlab="", main="prior 1")

hist(rr2$sigma2/rr2$tau2, prob=TRUE, main="prior 2", xlab = expression(lambda), breaks=50)
plot(rr2$sigma2/rr2$tau2, type="l", ylab = expression(lambda), xlab="", main="prior 1")
bayes.optimal_lambda <- mean(rr2$sigma2/rr2$tau2)


hist(rr3$sigma2/rr3$tau2, prob=TRUE, main="prior 3", xlab = expression(lambda), breaks=50)
plot(rr3$sigma2/rr3$tau2, type="l", ylab = expression(lambda), xlab="", main="prior 1")
dev.off()

df_lambda <- function(x){
  #the degrees of freedom for penalised spline regression model
  if(is.nan(x)==TRUE){
```

```
    NaN
  }else {
    if (x==0){
      ncol(X)
    }else{
      if (x>100000){
        0
      }else{
        sum(diag(X%*%solve(crossprod(X) + x*D, t(X))))
      }
    }
  }
}

#sample from the prior distribution fo s2 and tau2
s2 <- 1/rgamma(10000, shape=0.1, rate=0.1)
tau2 <- 1/rgamma(10000, shape=0.1, rate=0.1)
lambda_prior <- s2/tau2

pdf('q1_dof_beforetune.pdf')
hist(na.omit(sapply(lambda_prior, FUN=df_lambda)), main="",
     xlab=expression(H[lambda]), prob=TRUE, breaks = 20)

#sample from the prior distribution fo s2 and tau2
s2 <- 1/rgamma(10000, shape=2, rate=.01)
tau2 <- 1/rgamma(10000, shape=6, rate=.01)
lambda_prior <- s2/tau2

pdf('q1_dof_aftertune.pdf')
hist(na.omit(sapply(lambda_prior, FUN=df_lambda)), main="",
     xlab=expression(H[lambda]), prob=TRUE, xlim=c(2,6), breaks = 50)

# --- END --- #
```

Listing 2: Question 2a Code

```

# UCT Assignment
# Author: Julian Albert
# Date:   04/04/2019

rm(list = ls()); dev.off()
# Assignment prelim script
source("../../../../PrelimScript/assignment_prelim_script.R")
## 0. Load required packages
p_load(mvtnorm, MASS, psych, parallel)

# Question 2 ----

## 2.1 Read in the Data ----
set.seed(123)

### All the predictor and response data
data <- read.csv('Gametocyte_Data.csv',
                 header = T, sep =';') %>% as_tibble()
### time data
dat_times <- c(0, 3, 7, 14, 21, 28, 42)
### knot data
dat_knot1 <- c(0.5, 10, 25)

### seperate groups
Group1 <- data %>%
  dplyr::filter(grp == 1) %>%
  dplyr::select(-c(profile_num, grp))

plot(as.numeric(Group1[1, ])~dat_times, ylim = c(0, 12), col ='red',
     type = 'o', main = 'Group 1', ylab = 'Response', xlab = 'Time')
for(i in 1:dim(Group1)[1]) lines(as.numeric(Group1[i, ])~dat_times,
                                 type = 'o', col = 'red')

Group2 <- data %>%
  dplyr::filter(grp != 1) %>%
  dplyr::select(-c(profile_num, grp))
```

```r

plot(as.numeric(Group2[1, ])~dat_times, ylim = c(0, 12), col = 'blue',
     type = 'o', main = 'Group 2', ylab = 'Response', xlab = 'Time')
for(i in 1:dim(Group2)[1]) lines(as.numeric(Group2[i, ])~dat_times,
                                 type = 'o', col = 'blue')

### both groups with Group variable
dat_all <- cbind(bind_rows(Group1, Group2), factor(data$grp)) %>%
  as_tibble() %>% rename(Group = `factor(data$grp)`)

### PARAMETERS
N <- NROW(dat_all)
n1 <- NROW(Group1)
n2 <- NROW(Group2)

J <- length(dat_times)
Lstar <- length(dat_knot1)
L <- length(dat_knot1) + 2 # add two for betas
knot.function <- function(x) return(x*log(x))

I <- diag(Lstar)
D <- matrix(0, L, L)
D[(Lstar:L), (Lstar:L)] <- I

basis_matrix <- matrix(0, J, Lstar)
for(j in 1:J){
  basis_matrix[j, ] <- abs(dat_times[j] - dat_knot1) %>%
    knot.function()
}

Design_matrix <- cbind(1, dat_times, basis_matrix)

## 2.2 Gibbs sampler ----
### paramaters
i1 <- 0.001; i2 <- 0.001
i3 <- 0.001; i4 <- 0.001
nsamps <-  15000
### Inpute Data
predictors <- Design_matrix
response <- dat_all %>% dplyr::filter(Group == 1) %>%
  dplyr::select(-Group) %>% as.matrix()

func.gibbs <- function(predictors, # the predictor variables
           response, # the response variable
           D,
           nsamps = 15000, # the no. of iterations used in the Gibbs sampler
           i1 = 0.001, i2 = 0.001, # pars for prior for error variance
           i3 = 0.001, i4 = 0.001 # pars for prior for phi variance
           )
{
  ### Data Matrices
  X <- predictors
  XtX <- crossprod(X)
  Y <- response

  ### Initial Parameter Specifications
  var_e <- 1
  var_phi <- 1
  icounter <- 0
  Sigma_e <- numeric()
  Sigma_Phi <- numeric()
  Phi <- matrix(0, L, floor(nsamps/2))

  ### Posterior Sampling
  for (i in 1:nsamps){

    # Phi Given All Else
    sigma_0 <- solve(1/var_e * XtX + 1/var_phi * D)
    mu_0 <- 1/var_e * (sigma_0 %*% t(X) %*% Y)
    phi_posterior <- mvrnorm(1, mu_0, sigma_0)

    # Error Variance Given All Else
    a_e <- (i1 + 7/2)
```

```
    b_e <- (i2 + 1/2 * crossprod(Y - X %*% phi_posterior))
    var_e <- 1/rgamma(1, a_e, b_e)

    # Error Phi Given All Else
    a_phi <- (i3 + Lstar/2)
    b_phi <- (i4 + 1/2 *  t(phi_posterior) %*% D %*% phi_posterior)
    var_phi <- 1/rgamma(1, a_phi, b_phi)

    if (i > floor(nsamps/2)){
      icounter <- icounter + 1
      Sigma_e[icounter] <- var_e
      Sigma_Phi[icounter] <- var_phi
      Phi[, icounter] <- phi_posterior
    }
  } #end of simulations

  return(list(Phi = Phi,
              Sigma_Phi = Sigma_Phi,
              Sigma_e = Sigma_e))
}

func.gibbs_statcheck <- function(predictors, # the predictor variables
                      response, # the response variable
                      D,
                      nsamps = 15000, # the no. of iterations used in the Gibbs sampler
                      i1 = 0.9, i2 = 0.1, # pars for prior for error variance
                      i3 = 5, i4 = 0.03 # pars for prior for phi variance
                      )
{
  ### Data Matrices
  X <- predictors
  XtX <- crossprod(X)
  Y <- response

  ### Initial Parameter Specifications
  var_e <- 0.1
  var_phi <- 0.1
  icounter <- 0
  Sigma_e <- numeric()
  Sigma_Phi <- numeric()
  Phi <- matrix(0, L, floor(nsamps/2))

  ### Posterior Sampling
  for (i in 1:nsamps){

    # Phi Given All Else
    sigma_0 <- solve(1/var_e * XtX + 1/var_phi * D)
    mu_0 <- 1/var_e * (sigma_0 %*% t(X) %*% Y)
    phi_posterior <- mvrnorm(1, mu_0, sigma_0)

    # Error Variance Given All Else
    a_e <- (i1 + 7/2)
    b_e <- (i2 + 1/2 * crossprod(Y - X %*% phi_posterior))
    var_e <- 1/rgamma(1, a_e, b_e)

    # Error Phi Given All Else
    a_phi <- (i3 + Lstar/2)
    b_phi <- (i4 + 1/2 * t(phi_posterior) %*% D %*% phi_posterior)
    var_phi <- 1/rgamma(1, a_phi, b_phi)

    if (i > floor(nsamps/2)){
      icounter <- icounter + 1
      Sigma_e[icounter] <- var_e
      Sigma_Phi[icounter] <- var_phi
      Phi[, icounter] <- phi_posterior
    }
  } #end of simulations

  return(list(Phi = Phi,
              Sigma_Phi = Sigma_Phi,
              Sigma_e = Sigma_e))
}
```

```r
test1 <- func.gibbs_statcheck(predictors, response[1,], D = D)
setwd('../Figs')
pdf('q2a_phi1.pdf')
plot(test1$Phi[1, ], type = 'l',
     xlab = 'Sample No.', ylab = bquote(tilde(phi[1])))
pdf('q2a_phi2.pdf')
plot(test1$Phi[2, ], type = 'l',
     xlab = 'Sample No.', ylab = bquote(tilde(phi[2])))
pdf('q2a_phi3.pdf')
plot(test1$Phi[3, ], type = 'l',
     xlab = 'Sample No.', ylab = bquote(tilde(phi[3])))

## 2.3 Let's GO! ----

### Run Gibbs in parallel
no.cores <- detectCores()
cl <- makeCluster(no.cores)
clusterExport(cl, c("L", 'J', 'Lstar'))
clusterEvalQ(cl, library(MASS))
q2a_gibbs.out <- parApply(cl = cl, X = response, MARGIN = 1,
                          FUN =func.gibbs, predictors = predictors,
                          D = D, nsamps = nsamps,
                          i1 = i1, i2 = i2, i3 = i3, i4 = i4)
stopCluster(cl)

### should get a 25x7 prediction matrix, Credibility Intervals
names(q2a_gibbs.out) <- seq(1, 25) %>% as.character()
yhats <- lapply(q2a_gibbs.out, function(x) predictors %*% x$Phi)
y.lowers <- lapply(yhats, apply, 1, function(x) quantile(x, 0.025))
y.means <- lapply(yhats, apply, 1, function(x) mean(x))
y.uppers <- lapply(yhats, apply, 1, function(x) quantile(x, 0.975))
y.fitted <- lapply(q2a_gibbs.out,
                   function(x) predictors %*% rowMeans(x$Phi))

### get the MSE
yhat_long <- lapply(q2a_gibbs.out,
                    function(x) predictors %*% rowMeans(x$Phi)) %>%
  bind_rows() %>% as.matrix() %>% as.vector()

response_long <- as.vector(t(response)) # 175 x 1
q2a.MSE <- mean((response_long - yhat_long)^2)

set.seed(420)
tmp.sample <- c(1, sample(NROW(Group1), 3))

setwd('../Figs')
pdf('credibility_intervals.pdf')
par(mfrow=c(2,2), family="serif", mai=c(0.4,0.4,0.4,0.4))
for(i in tmp.sample){
  plot(as.numeric(Group1[i, ])~dat_times, ylim = c(0, 13),
       col ='darkgreen', type = 'o', ylab = 'Response', xlab = 'Time',
       main = bquote('Profile' ~ .(i)))
  lines(y.lowers[[i]]~dat_times, type = 'l', col = 'red', lty = 2)
  lines(y.means[[i]]~dat_times, type = 'o', col = 'blue')
  lines(y.uppers[[i]]~dat_times, type = 'l', col = 'red', lty = 2)
  lines(y.fitted[[i]]~dat_times, type = 'l', col = 'blue')
  lines(as.numeric(Group1[i, ])~dat_times,
        type = 'o', col = 'darkgreen')
  legend(x= 26, y=13, bty="n",
         col=c('red', 'blue', 'darkgreen'),
         cex=0.8, c("Interval" ,"Predicted", 'True'),
         lty = c(2, 1, 1), y.intersp=1, lwd = c(1.5, 1.5, 1.5))
}
dev.off()

## 2.4 Tune priors ----

#sample from the prior distribution of var_e and var_phi
num_tuning_samps <- 10000
tmp.shapes_e <- round(seq(0.5, 1.5, length.out = 10), 2)
tmp.rates_e <- round(seq(0.05, 0.15, length.out = 10), 2)

tmp.shapes_phi <- round(seq(4.5, 5.5, length.out = 10), 2)
```

```
tmp.rates_phi <- round(seq(0.01, 0.05, length.out = 10), 2)

set.seed(123)
for(i in 1:length(tmp.shapes_e)){
### cycle through rates and shapes
tmp.var_e <- 1/rgamma(num_tuning_samps, tmp.shapes_e[i], tmp.rates_e[i])
tmp.var_phi <- 1/rgamma(num_tuning_samps, tmp.shapes_phi[i], tmp.rates_phi[i])
edf <- numeric(num_tuning_samps)
X <- predictors
XtX <- crossprod(X)
### get different edf values
  for(j in 1:num_tuning_samps){
    tmp.sigma_0 <- solve(1/tmp.var_e[j] * XtX + 1/tmp.var_phi[j] * D)
    H <- X %*% tmp.sigma_0 %*% t(X) * 1/tmp.var_e[j]
    edf[j] <- tr(H)
  }
### plot this bad boi
  if( i == 3){
  setwd('../Figs')
  pdf('q2a_tuning_is.pdf')
  hist(edf, breaks = 20,
       xlab = expression(tr(H)), prob=T,
       xlim = c(0, 5),
       main = bquote(atop(i[1] == .(tmp.shapes_e[i]),
                          i[2] == .(tmp.rates_e[i])) ~
                     atop(i[3] == .(tmp.shapes_phi[i]),
                          i[4] == .(tmp.rates_phi[i])) ~~~~~
                     'Iteration' == .(i) ))
  dev.off()
  }

} # iteration 10

### Nice Left Skew... Not too skew
tuned_shape_e <- tmp.shapes_e[3]
tuned_rate_e <- tmp.rates_e[3]
tuned_shape_phi <- tmp.shapes_phi[3]
tuned_rate_phi <- tmp.rates_phi[3]

### rerun with sensical priors
no.cores <- detectCores()
cl <- makeCluster(no.cores)
clusterExport(cl, c("L", 'J', 'Lstar'))
clusterEvalQ(cl, library(MASS))
q2a_gibbs.out2 <- parApply(cl = cl, X = response, MARGIN = 1,
                           FUN =func.gibbs, predictors = predictors,
                           D = D, nsamps = nsamps,
                           i1 = tuned_shape_e, i2 = tuned_rate_e,
                           i3 = tuned_shape_phi, i4 = tuned_rate_phi)
stopCluster(cl)

### should get a 25x7 prediction matrix
names(q2a_gibbs.out2) <- seq(1, 25) %>% as.character()
### get the MSE
yhat2_long <- lapply(q2a_gibbs.out2,
                     function(x) predictors %*% rowMeans(x$Phi)) %>%
  bind_rows() %>% as.matrix() %>% as.vector()

### get the MSE
q2a.MSE2 <- mean((response_long-yhat2_long)^2)


# --- END --- #
```

Listing 3: Question 2b Code

```

# UCT Assignment
# Author: Julian Albert
# Date:   12/04/2019

rm(list = ls()); dev.off()
# Assignment prelim script
```

```r
source("../../../../PrelimScript/assignment_prelim_script.R")
## 0. Load required packages
p_load(mvtnorm, MASS, psych, Matrix)

# Question 2 ----

## 2.1 Read in the Data ----
set.seed(123)

### All the predictor and response data
data <- read.csv('Gametocyte_Data.csv',
                 header = T, sep =';') %>% as_tibble()
### time data
dat_times <- c(0, 3, 7, 14, 21, 28, 42)
### knot data
dat_knot1 <- c(0.5, 10, 25)
dat_knot2 <- c(0.5, 5, 25)

### seperate groups
Group1 <- data %>%
  dplyr::filter(grp == 1) %>%
  dplyr::select(-c(profile_num, grp))

Group2 <- data %>%
  dplyr::filter(grp != 1) %>%
  dplyr::select(-c(profile_num, grp))

### both groups with Group variable
dat_all <- cbind(bind_rows(Group1, Group2), factor(data$grp)) %>%
  as_tibble() %>% rename(Group = `factor(data$grp)`)

### PARAMETERS
N <- NROW(dat_all)
n1 <- NROW(Group1)
n2 <- NROW(Group2)
J <- length(dat_times)
Lstar <- length(dat_knot1)

no.rows <- N*J
no.cols <- 3 + Lstar*N

knot.function <- function(x) return(x*log(x))

I <- diag(Lstar*N)
D <- matrix(0, no.cols, no.cols)
D[((Lstar+1):no.cols), ((Lstar+1):no.cols)] <- I

basis_matrix <- matrix(0, no.rows, no.cols)
basis_matrix[, 1] <- 1
basis_matrix[, 2] <- dat_times
basis_matrix[1:no.rows/2, 3] <- 1

Z1 <- t(apply(as.matrix(dat_times), 1,
        function(x) abs(x - dat_knot1) %>%
          knot.function()))

Z2 <- t(apply(as.matrix(dat_times), 1,
              function(x) abs(x - dat_knot2) %>%
                knot.function()))

Z1_mat <- diag(25) %x% Z1
Z2_mat <- diag(25) %x% Z2

basis_matrix[1:dim(Z1_mat)[1],
             4:(dim(Z1_mat)[2]+3)] <- Z1_mat

basis_matrix[(dim(Z1_mat)[1]+1):dim(basis_matrix)[1],
             (dim(Z1_mat)[2]+4):dim(basis_matrix)[2]] <- Z2_mat

## 2.3 Gibbs sampler ----

func.gibbs <- function(predictors, # the predictor variables
                       response, # the response variable
```

```r
                      D,
                      nsamps = 15000, # the no. of iterations used in the Gibbs sampler
                      i1 = 0.001, i2 = 0.001, # pars for prior for error variance
                      i3 = 0.001, i4 = 0.001 # pars for prior for phi variance
)
{
  ### Data Matrices
  X <- predictors # 350 153
  XtX <- crossprod(X) # 153 153
  Y <- response # 50 7
  Y_long <- as.vector(t(Y)) %>% as.matrix() # 350   1

  ### Initial Parameter Specifications
  var_e <- 1
  var_phi <- 1
  icounter <- 0
  Sigma_e <- numeric()
  Sigma_Phi <- numeric()
  Phi <- matrix(0, no.cols, floor(nsamps/2)) # 153 7500

  ### Posterior Sampling
  for (i in 1:15000){

    # Phi Given All Else
    sigma_0 <- solve(1/var_e * XtX + 1/var_phi * D)
    mu_0 <- 1/var_e * (sigma_0 %*% t(X) %*% Y_long)
    phi_posterior <- mvrnorm(1, mu_0, sigma_0)

    # Error Variance Given All Else
    a_e <- (i1 + no.rows/2)
    b_e <- (i2 + 1/2 * crossprod(Y_long - X %*% phi_posterior))
    var_e <- 1/rgamma(1, a_e, b_e)

    # Error Phi Given All Else
    a_phi <- (i3 + no.cols/2)
    b_phi <- (i4 + 1/2 *  t(phi_posterior) %*% D %*% phi_posterior)
    var_phi <- 1/rgamma(1, a_phi, b_phi)

    if (i > floor(nsamps/2)){
      icounter <- icounter + 1
      Sigma_e[icounter] <- var_e
      Sigma_Phi[icounter] <- var_phi
      Phi[, icounter] <- phi_posterior
    }

  } #end of simulations

  return(list(Phi = Phi,
              Sigma_Phi = Sigma_Phi,
              Sigma_e = Sigma_e))
}

## 2.3 Let's GO!

### paramaters
i1 <- tuned_shape_e; i2 <- tuned_rate_e
i3 <- tuned_shape_phi; i4 <- tuned_rate_phi

predictors <- basis_matrix
response <- dat_all %>% dplyr::select(-Group) %>% as.matrix()

set.seed(123)
q2b_gibbs.out <- func.gibbs(predictors, response, D, 15000,
                            i1, i2, i3, i4)
q2b_params <- rowMeans(q2b_gibbs.out$Phi)
q2b_yhat <- predictors %*% q2b_params # mse = 0.1106431
q2b_mse <- mean((q2b_yhat-Y_long)^2)

q2b_y_plot <- matrix(Y_long, nrow = 50, byrow = TRUE)
q2b_yhat_plot <- matrix(q2b_yhat, nrow = 50, byrow = TRUE)

set.seed(123)
tmp.sample <- c(1, sample(NROW(Group1), 3))
```

```
154
155 setwd('../Figs')
156 pdf('q2b_fits.pdf')
157 par(mfrow=c(2,2), family="serif", mai=c(0.4,0.4,0.4,0.4))
158 for(i in tmp.sample){
159   plot(as.numeric(Group1[i, ])~dat_times, ylim = c(0, 13),
160        col ='darkgreen', type = 'o', ylab = 'Response', xlab = 'Time',
161        main = bquote('Profile' ~ .(i)))
162   lines(q2b_yhat_plot[i, ]~dat_times, type = 'o', col = 'blue')
163   lines(as.numeric(Group1[i, ])~dat_times,
164         type = 'o', col = 'darkgreen')
165   legend(x= 26, y=13, bty="n",
166          col=c('blue', 'darkgreen'),
167          cex=0.8, c("Predicted", 'True'),
168          lty = c(1, 1), y.intersp=1, lwd = c(1.5, 1.5))
169 }
170 dev.off()
```

Plagiarism Declaration Form

A copy of this form, completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department.

COURSE CODE:	**STA5090Z**
COURSE NAME:	**Advanced Regression**
STUDENT NAME:	**Julian Albert**
STUDENT NUMBER:	**ALBJUL005**
GROUP NUMBER:	**N/A**

PLAGIARISM DECLARATION

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.
- This tutorial/report/project is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.
- Agreement to this statement does not exonerate me from the University's plagiarism rules.

Signature:  Date: April 23, 2019