BTRY 4030 - Fall 2018 - Homework 3 Q1

Greg Benton gwb67

Due Friday, October 12, 2018

Instructions:

Create your homework solution file by editing the "hw3-2018_q1.Rmd" Rmarkdown file provided. Your solution to this homework assignment should include the relevant R code and output (fit summaries, ANOVA tables and computed statistics, as well as requested plots) in addition to written comments where requested. Do not include output that is not relevant to the question. You should turn in a .pdf version of your compiled code.

You may discuss the homework problems and computing issues with other students in the class. However, you must write up your homework solution on your own. In particular, do not share your homework RMarkdown file with other students.

Question 1

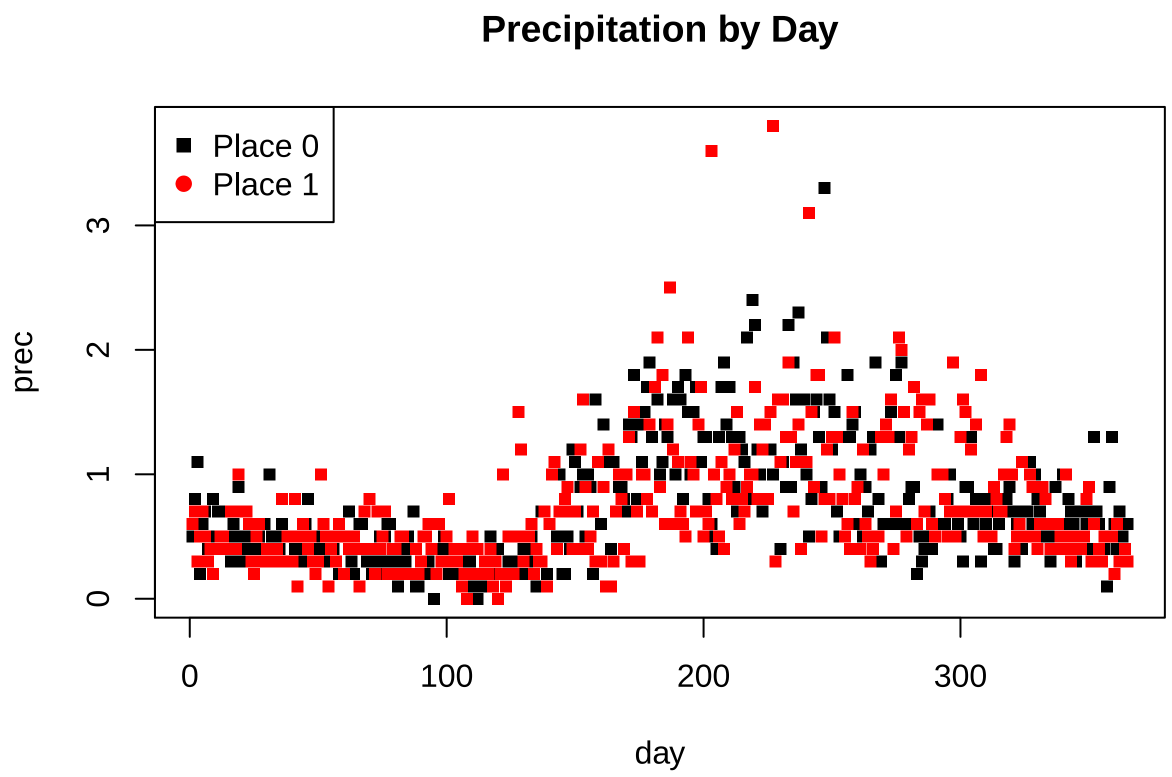
This question is based on the data in *precipitation.csv* which gives average daily precipitation for two Canadian cities. There are three columns indicating

– the day of year – the city (0 = Whitehorse, 1 = Yellowknife) – average precipitation between 1940 and 1990

a. Plot precipitation over time, indicating the city by different marks.

```
precip = read.csv("./precipitation.csv", header=T)
```

```
plot(prec ~ day, data=precip, pch=15, cex=0.8, col=1+precip$place,
     main="Precipitation by Day")
legend("topleft", pch=c(15, 19), col=1:2,
       legend=c("Place 0", "Place 1"))
```



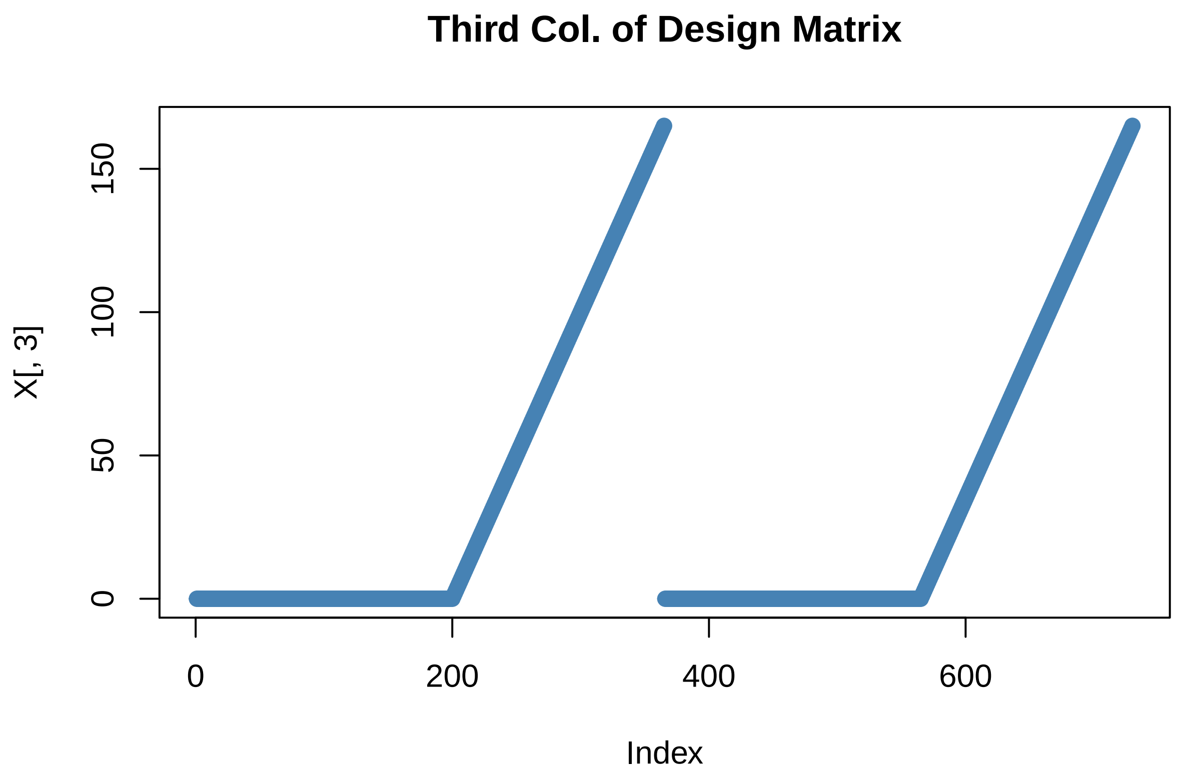
b. Comment on whether a linear model is appropriate here.

Probably not without some transformational work. The seasonality could be captured by a linear model with sinusoidal covariates, but the data appear to be heteroskedastic (variance is different in the winter than the summer).

c. Form a matrix of linear spline bases in d = day with columns $[1, d, (d-100)_+, (d-200)_+, (d-300)_+]$ where $(d-100)_+$ is $\max(d-100, 0)$. In `R` you can get this from `(d-100)*(d>100)` or `pmax(d-100,0)`. Plot the third column of this matrix.

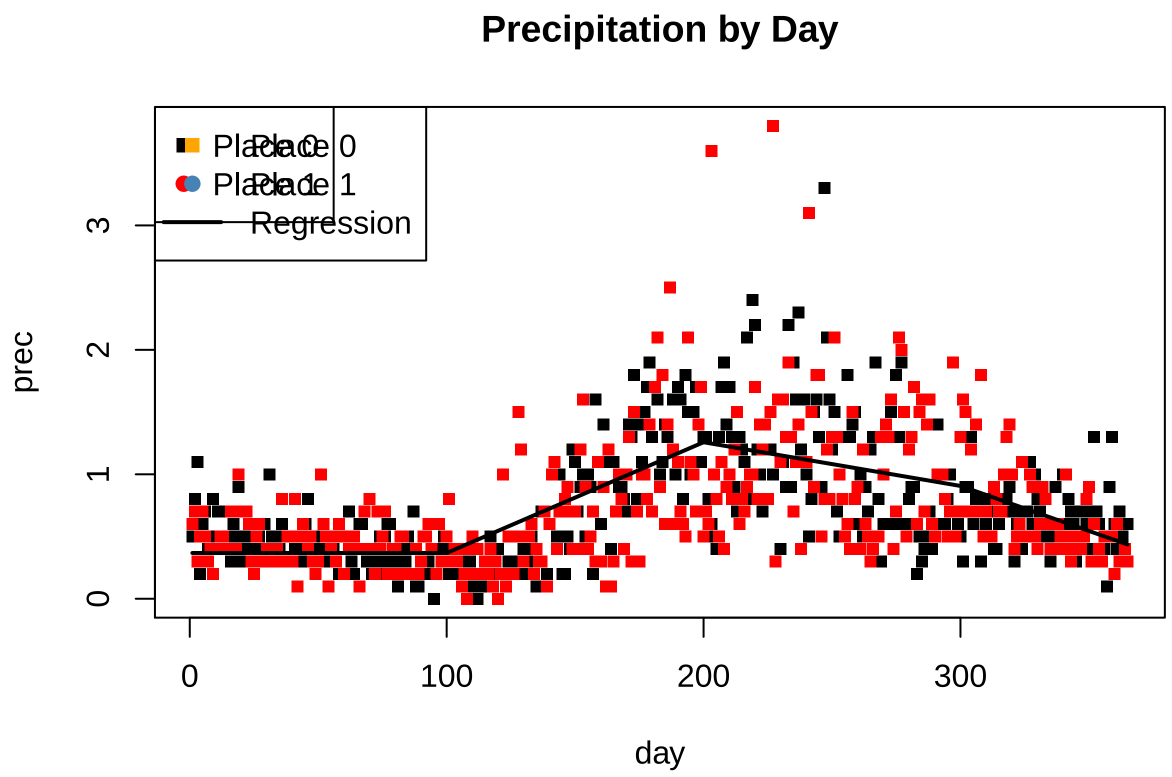
```r
n = nrow(precip)
d = precip$day
X = cbind(rep(1, n), (d-100)*(d>100), (d-200)*(d>200), (d-300)*(d>300))

plot(X[ , 3], pch=19, col="steelblue", main="Third Col. of Design Matrix")
```



d. Regress precipitation onto your basis in part c and plot the resulting function along with the data. Ie, plot the prediction for each day as a line through the data.

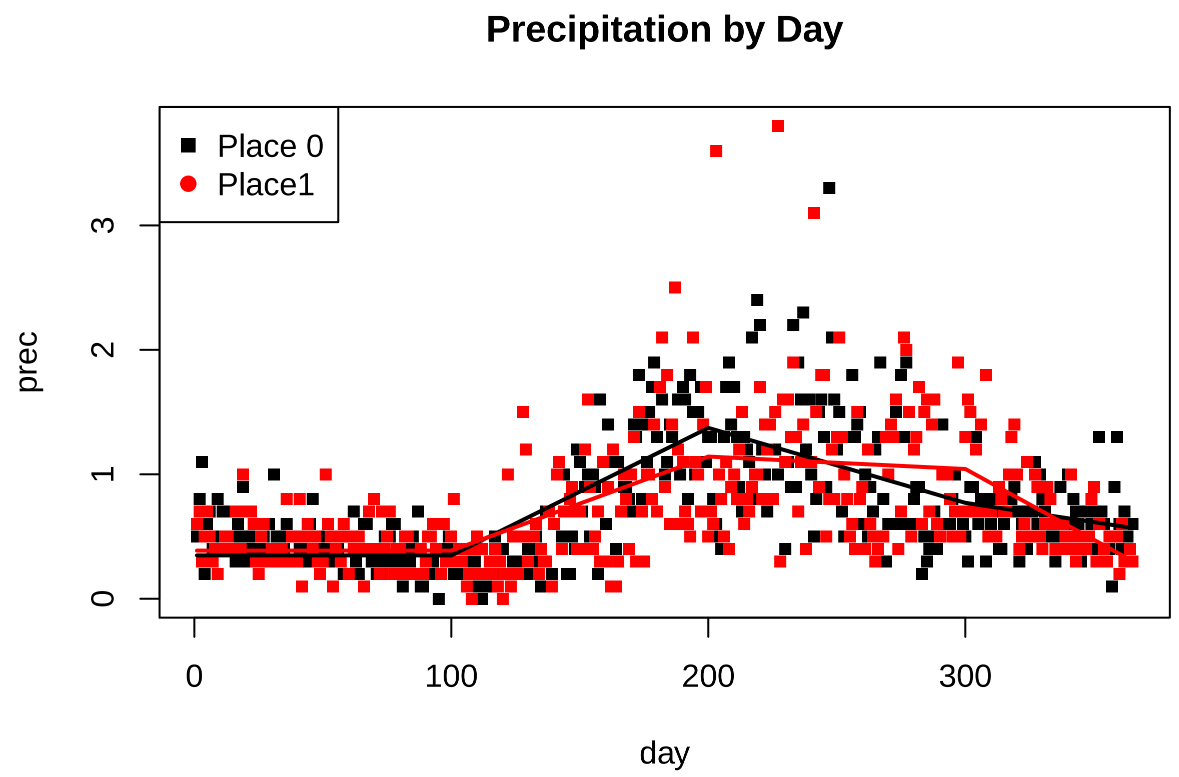
```
model_d = lm(precip$prec ~ X-1)  # We already have the intercept in X
prec_fits = model_d$fitted.values[1:365] # only need the first 365 (it will just be repeated here)
plot(prec ~ day, data=precip, pch=15, cex=0.8, col=1+precip$place,
     main="Precipitation by Day")
legend("topleft", pch=c(15, 19), col=1:2,
       legend=c("Place 0", "Place 1"))
lines(prec_fits ~ precip$day[1:365], lwd=2, col="black")
legend("topleft", pch=c(15, 19, NA), col=c("orange", "steelblue", "black"),
       lwd=c(NA, NA, 2), legend=c("Place 0", "Place 1", "Regression"))
```



e. Obtain a design matrix to allow you to plot different precipitation curves for each city with one model. Produce these plots.

```r
X_new = cbind(X, precip$place*X)
model_e = lm(precip$prec ~ X_new -1)
place_0_fits = model_e$fitted.values[precip$place==0]
place_1_fits = model_e$fitted.values[precip$place==1]

plot(prec ~ day, data=precip, pch=15, cex=0.8, col=1+precip$place,
     main="Precipitation by Day")
legend("topleft", pch=c(15, 19), col=1:2,
       legend=c("Place 0", "Place1"))
lines(place_0_fits, lwd=2, col = 1)
lines(place_1_fits, lwd=2, col = 2)
```



Question 2

This question follows on from Q1 of HW 2. As in that case, the file "brainweight.txt" contains measurements of **brain weight** (B, in kilograms), **body weight** (W, in kilograms), and **gestation period** (G, in days) of a sample of 96 mammal species. The following commands can be used to read the data into R (although you will need point the **setwd** command to the directory on your computer that contains the data file.).

Here we will look at variability. We'll start by re-fitting the model

```
brain_data =read.table('BrainWeight.txt',head=TRUE)
colnames(brain_data)=c("B","W","G")
brain_mod = lm(B~W+G,data=log(brain_data))
summary(brain_mod)
```

```
##
## Call:
## lm(formula = B ~ W + G, data = log(brain_data))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00286 -0.30372 -0.05242  0.37851  1.58788
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.45728    0.45848  -0.997    0.321
## W            0.55117    0.03236  17.033   <2e-16 ***
## G            0.66782    0.10875   6.141    2e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4902 on 93 degrees of freedom
## Multiple R-squared:  0.9501, Adjusted R-squared:  0.949
## F-statistic: 885.2 on 2 and 93 DF,  p-value: < 2.2e-16
```

a. Calculate the estimated covariance of the coefficients $\sigma^2(X^TX)^{-1}$.

```
design_mat = model.matrix(brain_mod)
cov_mat = summary(brain_mod)$sigma^2 * solve(t(design_mat) %*% design_mat)
cov_mat
```

```
##             (Intercept)           W           G
## (Intercept)  0.21020824  0.011792619 -0.049398988
## W            0.01179262  0.001047074 -0.002975323
## G           -0.04939899 -0.002975323  0.011825820
```

b. What prediction do you make for log(B) for a new animal with Body size 162 and Gestation period 143? What is the standard deviation of that prediction? It may help to write this in terms of $\mathbf{x}^T\hat{\beta}$ and use the results in Q4.

```
# prediction:
x = matrix(c(1, log(162), log(143)), nrow = 3)
pred = t(x) %*% brain_mod$coef
print(paste("The point prediction is:", pred))
```

```
## [1] "The point prediction is: 5.66111907688238"
```

To compute the variance of our prediction, we wish to find the variance of $x^T\hat{\beta}$. Knowing that $\hat{\beta}$ is an unbiased estimator for β this is

$$\begin{aligned}
&E[(x^T\hat{\beta} - x^T\beta)(x^T\hat{\beta} - x^T\beta)^T] && (1)\\
=&x^TE[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T]x && (2)\\
=&x^T\Sigma x && (3)
\end{aligned}$$

where Σ is the covariance matrix for the coefficients $\hat{\beta}$. Plugging this in below gives,

```
pred_var = t(x) %*% cov_mat %*% x
print(paste("The variance of the prediction is:", pred_var))
```

```
## [1] "The variance of the prediction is: 0.00800381422605338"
```

c. Calulate the VIFs for β_1 and β_2, verify that you get the same answer as the **vif** function in the **car** package.

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.5.1
```

```
## Loading required package: carData
```

```
# beta_1:
b1_mod = lm(W ~ G, data = log(brain_data))
vif_beta1 = 1/(1 - summary(b1_mod)$r.sq)
print(paste("VIF for Beta 1:", vif_beta1))
```

```
## [1] "VIF for Beta 1: 3.50783515334758"
```

```
# beta_2:
b2_mod = lm(G ~ W, data = log(brain_data))
vif_beta2 = 1/(1 - summary(b2_mod)$r.sq)
print(paste("VIF for Beta 2:", vif_beta2))
```

```
## [1] "VIF for Beta 2: 3.50783515334758"
```

```
print("From the `cars' package:")
```

```
## [1] "From the `cars' package:"
```

```
vif(brain_mod)
```

```
##        W        G 
## 3.507835 3.507835
```

d. Repeating the experiment from Homework 2, we'll transform our covariates to orthogonalize log(G) with respect to log(W)

```
brain_data$z = lm(G~W,data=log(brain_data))$resid
brain_mod2 = lm(log(B)~log(W)+z,data=brain_data)
```

Show (numerically) that the VIF for β_1 in part b is equal to the ratio of its variance in **brain_mod** to its variance in **brain_mod2**.

```
design_mat = model.matrix(brain_mod2)
mod2_cov_mat = summary(brain_mod2)$sigma^2 * solve(t(design_mat) %*% design_mat)

numeric_vif = cov_mat[2,2]/mod2_cov_mat[2,2]
print(paste("Numerically calculated VIF:", numeric_vif))
```

```
## [1] "Numerically calculated VIF: 3.50783515334758"
```

The VIF calculated in this way is the same as both versions calculated above (from the R^2 of the model of $W \sim G$ and from the 'cars' package).

e. Does the same conclusion hold for the variance of β_2 between the two models above?

```
guess_vif = cov_mat[3,3]/mod2_cov_mat[3,3]
print(paste("Numerically calculated VIF (beta 2):", guess_vif))
```

```
## [1] "Numerically calculated VIF (beta 2): 1"
```

The conclusion does not hold for β_2. This is because the coefficient for β_2 is identical in the two models; see Homework 2.

f. Do we get the same variance ratio if we fit an alternative model that ignores log(G)? Why not?

No, we will not.

```
reduced.mod = lm(B~W,data=log(brain_data))
```

```
## Warning in FUN(X[[i]], ...): NaNs produced
```

```
cov_mat[2,2]/summary(reduced.mod)$coef[2,2]
```

```
## [1] 0.05139406
```

The reason for this is that the VIF calculations keeps σ^2 the same. But when we drop $\log(G)$, the SSE increases and hense so does the estimated $\hat{\sigma}^2$.

bonus How would you generalize these calculations if you had more than two covariates?

Suppose we had the model

$$\mathbf{y} = X_1\beta_1 + X_2\beta_2 + \epsilon$$

then if we only used X_1 as covariates, we would have that

$$\text{var}(\hat{\beta}_1) = \sigma^2 (X_1^T X_1)^{-1}$$

To achieve this variance when we include the information for X_2, we take $Z = (I - H_1)X_2$ instead of X_2. That is, we regress out X_1 in which case we have

$$\text{var}\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \sigma^2 \begin{pmatrix} X_1^T X_1 & 0 \\ 0 & Z^T Z \end{pmatrix}^{-1} = \sigma^2 \begin{pmatrix} (X_1^T X_1)^{-1} & 0 \\ 0 & (Z^T Z)^{-1} \end{pmatrix}$$

since $X_1^T Z = 0$, thus achieving the same variance for $\hat{\beta}_1$.

Question 3

This question will perform a simulation based on the model in Question 2. Here we will see in simulation that the way we quantify uncertainty really is reasonable.

```r
brain_data = read.table("Brainweight.txt", head=T)
colnames(brain_data)=c("B","W","G")
```

a. Create 1000 new data sets each with the values of $log(G)$ and $log(W)$ using the coefficients and residual standard error estimated in Question 1 to produce the values in y (you will need to re-load the data and re-run **lm** to get this). On each data set, conduct an linear regression and record the estimated $\hat{\beta}$ its standard error, and $\hat{\sigma}$.

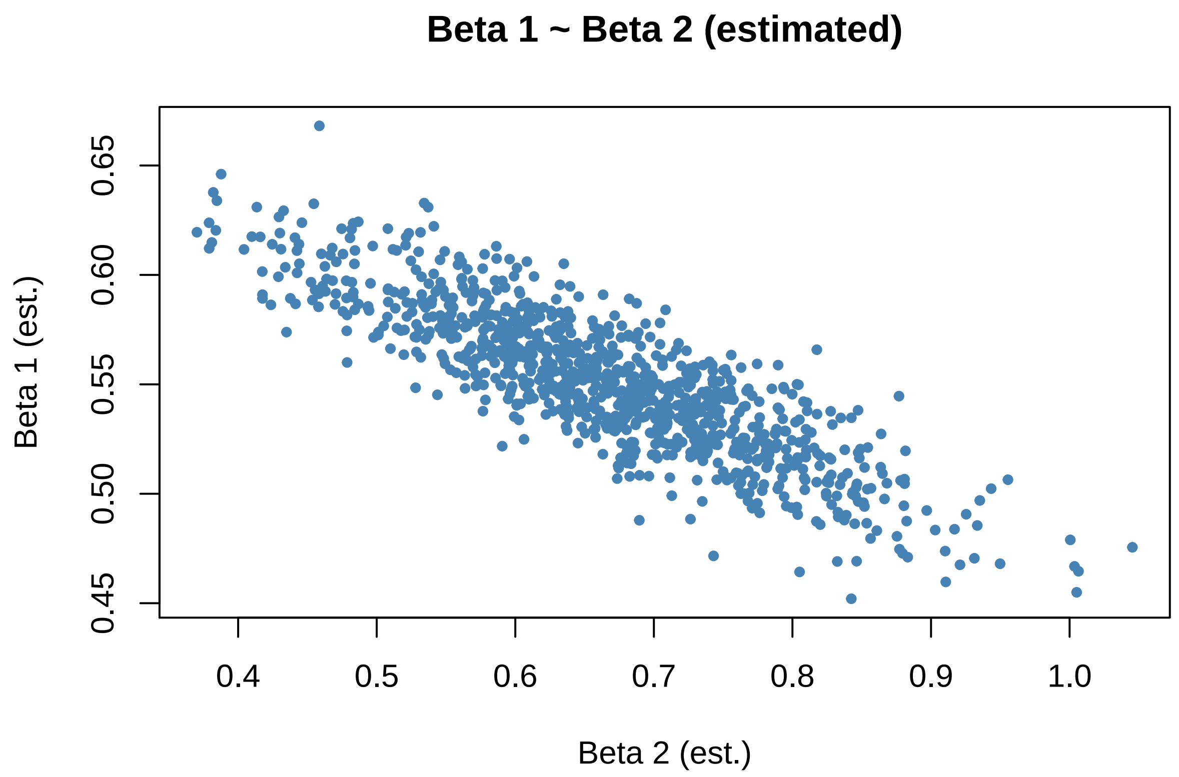
```r
# create model and get resid. std. error
original_model = lm(B ~ W + G, data=log(brain_data))
sig_hat = summary(original_model)$sigma

## initializations ##
n_data = 1000
new_data = array(NA, dim=c(n_data, nrow(brain_data)))
beta_hats = array(NA, dim=c(n_data, 3))
beta_std_errors = array(NA, dim=c(n_data, 3))
sigmas = rep(NA, n_data)

for (iter in 1:n_data) {
    new_data[iter, ] = rnorm(nrow(brain_data), mean=original_model$fitted.values, sd=sig_hat) # generat
    mod = lm(new_data[iter, ] ~ W + G, data=log(brain_data)) # build model
    beta_hats[iter, ] = mod$coeff # get coefficients
    beta_std_errors[iter, ] = summary(mod)$coeff[ , 2] # get out std errors
    sigmas[iter] = summary(mod)$sigma
}
```

b. Produce a scatter plot of the $\hat{\beta}_1$ and $\hat{\beta}_2$ values. Report their covariance. How does this relate to the relationship between log(G) and log(W)?

```r
plot(beta_hats[ , 2] ~ beta_hats[ , 3], pch=19, col="steelblue", cex=0.6,
     main="Beta 1 ~ Beta 2 (estimated)",
     ylab="Beta 1 (est.)", xlab="Beta 2 (est.)")
```



```r
print(paste("Covariance between Beta 1 and Beta 2:", cov(beta_hats[ , 2], beta_hats[ , 3])))
```

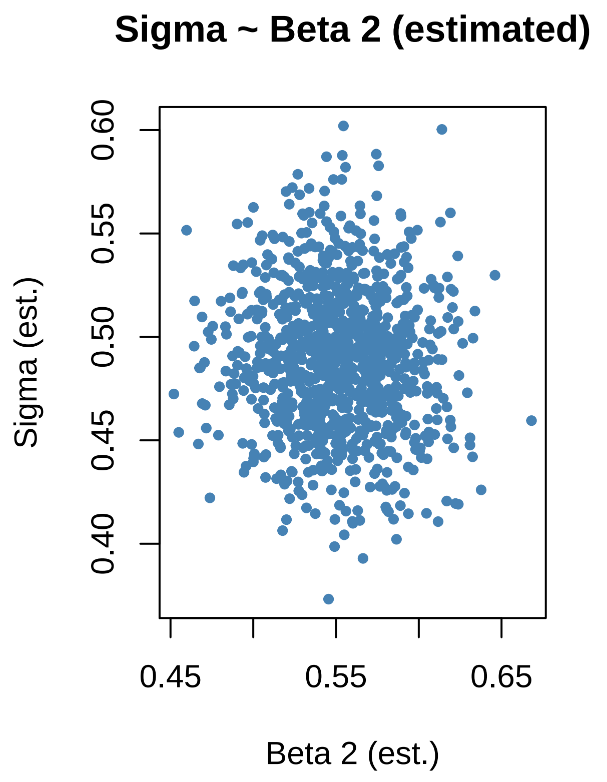
```
## [1] "Covariance between Beta 1 and Beta 2: -0.00316579109789055"
```

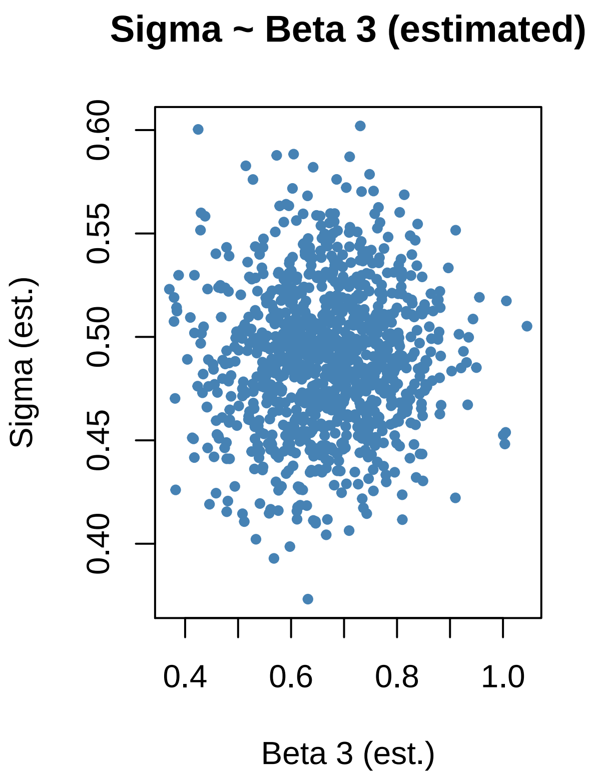
This indicates that if the values of log(G) were changed such that the corresponding coefficient was changed we would expect to see a relatively minor change in the coefficient of log(W) (due to the very small correlation coefficient between the two coefficients).

c. Construct scatterplots of $\hat{\sigma}$ versus each of $\hat{\beta}_1$ and $\hat{\beta}_2$. Is there a relationship here?

```r
par(mfrow = c(1,2))
plot(sigmas ~ beta_hats[ , 2],  pch=19, col="steelblue", cex=0.6,
     main="Sigma ~ Beta 2 (estimated)",
     ylab="Sigma (est.)", xlab="Beta 2 (est.)")

plot(sigmas ~ beta_hats[ , 3],  pch=19, col="steelblue", cex=0.6,
     main="Sigma ~ Beta 3 (estimated)",
     ylab="Sigma (est.)", xlab="Beta 3 (est.)")
```





From the above plots there does not seem to be any meaningful relationship between $\hat{\sigma}$ and either $\hat{\beta}_2$ or $\hat{\beta}_3$.

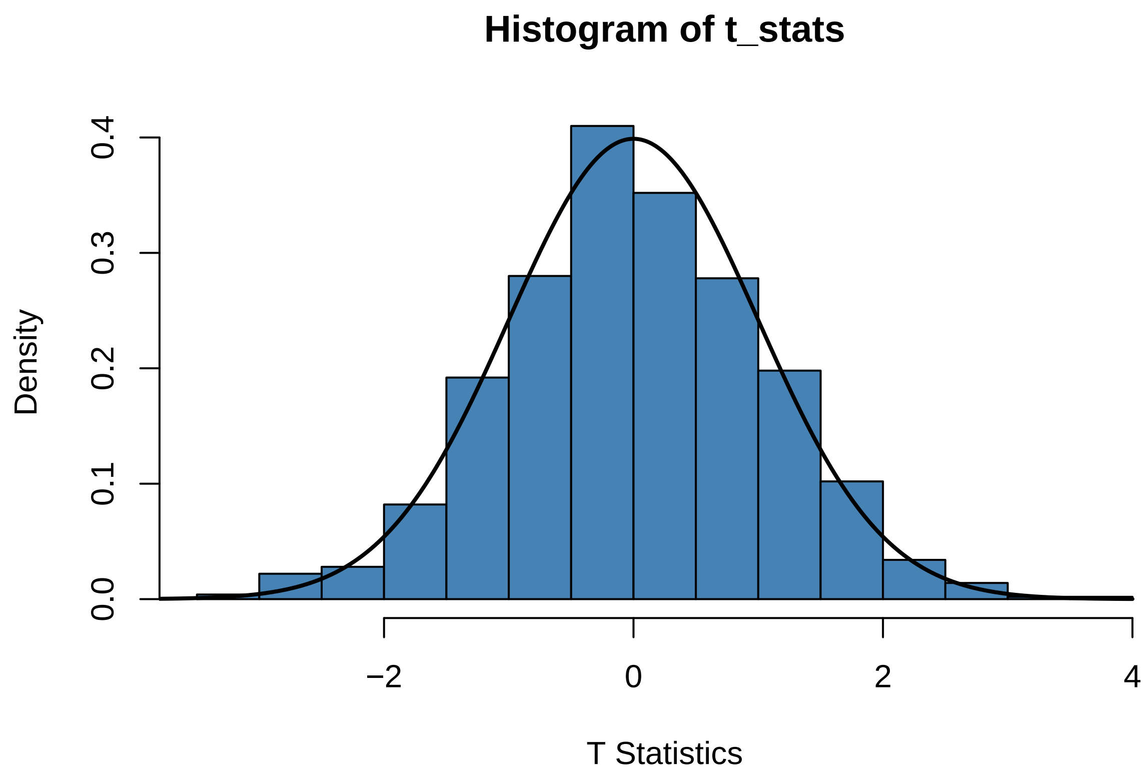
d. Construct a histogram of the 1000 t-statistics,

$$T_i = \frac{\hat{\beta}_{1i} - \beta_1}{\text{se}(\hat{\beta}_1)}$$

where β_1 is the coefficient for height. Overlay a t-density with the appropriate degrees of freedom.

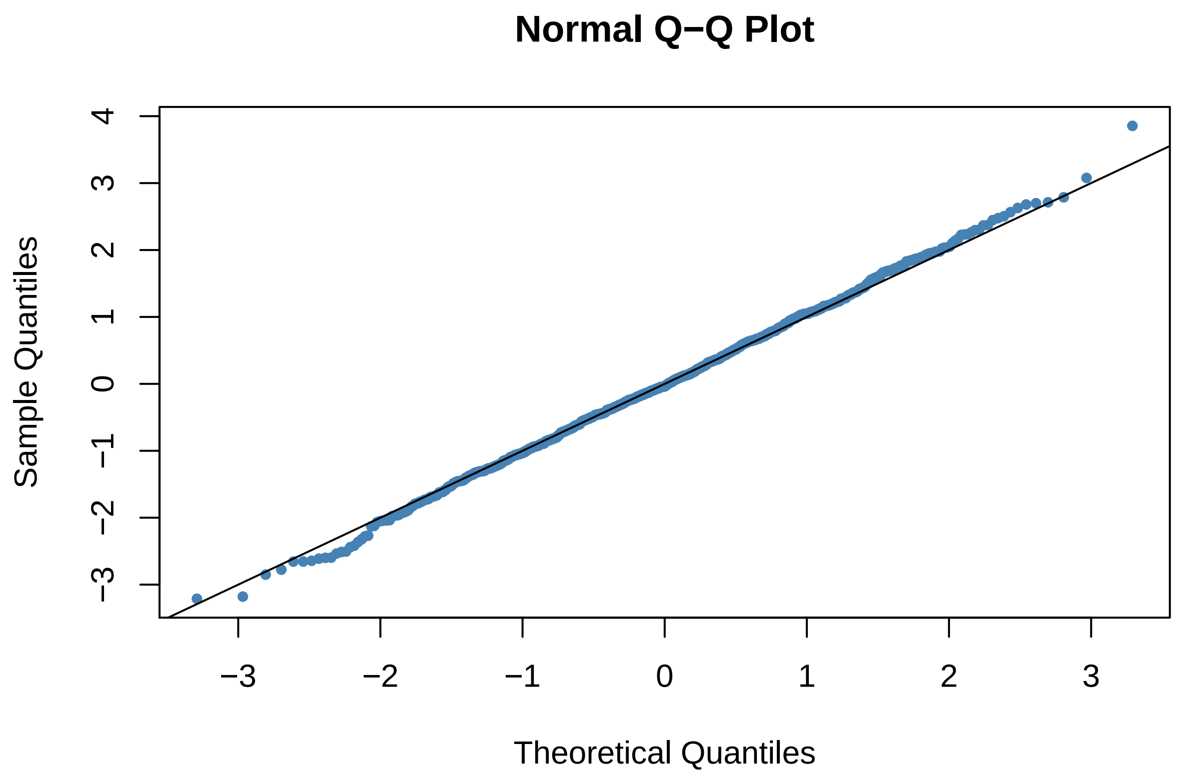
```r
og_beta1 = original_model$coefficients[2]
og_std_err = summary(original_model)$coeff[2,2]
t_stats = (beta_hats[ , 2] - og_beta1)/beta_std_errors[,2]

hist(t_stats, prob=T, col="steelblue", xlab="T Statistics")
domain = seq(-4, 4, by=0.01)
lines(dt(domain, df=(n_data - 2 - 1)) ~ domain, lwd=2)
```



e. Use the qqnorm function to construct a QQ plot of the 1000 simulated t-statistics and add a line through the origin with unit slope.

```
qqnorm(t_stats, pch=19, col="steelblue", cex=0.6)
abline(0,1)
```

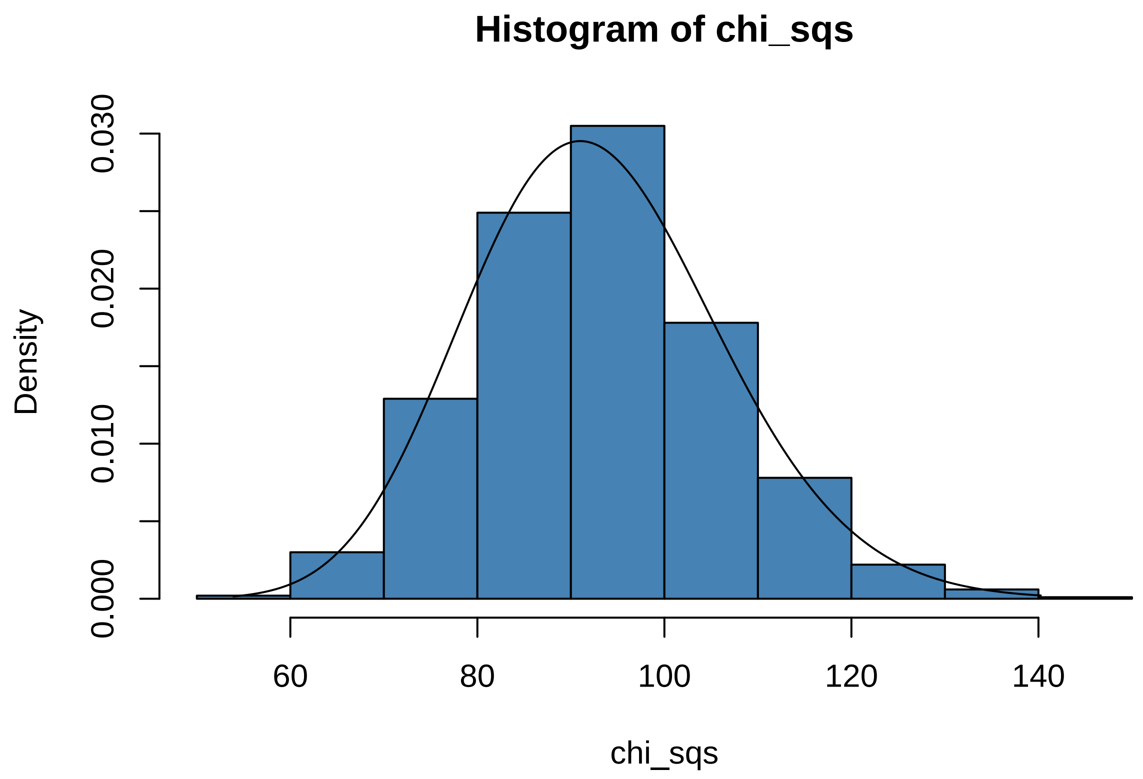


f. Construct a histogram of the 1000 chi-squared statistics of the form

$$\chi^2 = f\frac{\hat{\sigma}_i^2}{\sigma^2}$$

where σ^2 is the variance used to generate your errors, and f is the error degrees of freedom. Overlay a chi-squared density with f degrees of freedom.

```r
degrees = nrow(brain_data) - 2 - 1
chi_sqs= degrees*sigmas^2/sig_hat^2
hist(chi_sqs, prob=T, col="steelblue")
domain=seq(min(chi_sqs), max(chi_sqs), length.out=2000)
lines(dchisq(domain, df=degrees) ~ domain)
```



g. Report the mean and variance of the chi-square statistics and comment briefly on their values. Are they (approximately) what they should be?

```
mean(chi_sqs) # sample mean
```

```
## [1] 93.22855
```

```
nrow(brain_data) - 3 - 1 # theoretical mean
```

```
## [1] 92
```

```
var(chi_sqs) # sample var.
```

```
## [1] 175.5638
```

```
2 * (nrow(brain_data) - 3 - 1) # theoretical var.
```

```
## [1] 184
```

The sampled mean and variance of the chi-squared statistics taken from the values drawn in part a) are off by only 1-2% of what we expect the theoretical values to be, and given that the underlying σ^2 used is an estimate these values are approximately what they should be.

h. Verify your answer to part b in Q1 using these simulations (ie, obtain a prediction for each simulation and then take their variance).

```
design_mat = model.matrix(original_model)
cov_mat = summary(original_model)$sigma^2 * solve(t(design_mat) %*% design_mat)
cov_mat # covariance matrix from \hat{\sigma}^2 and design matrix
```

```
##              (Intercept)             W            G
## (Intercept)  0.21020824  0.011792619 -0.049398988
## W            0.01179262  0.001047074 -0.002975323
## G           -0.04939899 -0.002975323  0.011825820
```

```
cov(beta_hats) # covariance matrix from samples
```

```
##             [,1]         [,2]         [,3]
## [1,]  0.22249135  0.012652371 -0.052247528
## [2,]  0.01265237  0.001101236 -0.003165791
## [3,] -0.05224753 -0.003165791  0.012484213
```

All of these values are in agreement out to 2 or 3 decimal places, which is "close enough" to consider them to be in agreement.

For the specific prediction we can look at

```
var( beta_hats%*%c(1,log(163),log(142)))
```

```
##            [,1]
## [1,] 0.00888313
```

Which is the same as Q2b to four decimal places.

Question 4

a. Suppose that y has mean $(4.2, 3.9, -2.5)^T$ and covariance

$$\Sigma = \begin{pmatrix} 9.1 & 1.2 & -4.3 \\ 1.2 & 25.7 & -2.4 \\ -4.3 & -2.4 & 13.6 \end{pmatrix}$$

determine the mean and variance of $x = Ay + b$ for

$$A = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & -1 \end{pmatrix}$$

and $b = (-2.7, 4.5)^T$. *You may do these calculations in R if you wish.*

```
y_mean = c(4.2, 3.9, -2.5)
sig_mat = matrix(c(9.1, 1.2, -4.3,
             1.2, 25.7, -2.4,
             -4.3, -2.4, 13.6), nrow=3)
A = matrix(c(1/3, 1/3, 1/3,
             1/2, 1/2, -1), nrow=2,byrow=TRUE)
b = c(-2.7, 4.5)
```

We can use the linearity of expectation to compute

$$E[x] = E[Ay + b] = AE[y] + b,$$

which is computed below.

```
## Expected Value ##
expec = A %*% y_mean + b
expec
```

```
##            [,1]
## [1,] -0.8333333
## [2,] 11.0500000
```

Using the properties of variance we get

$$Var(x) = Var(Ay + b) = Var(Ay) = AVar(y)A^T = A\Sigma A^T,$$

which is computed below.

```r
## variance ##
var = A %*% sig_mat %*% t(A)
var
```

```
##           [,1]      [,2]
## [1,] 4.155556  2.783333
## [2,] 2.783333 29.600000
```

b. We've stated in class that if $\mathbf{x}$ has variance Σ then $A\mathbf{x}$ has variance $A\Sigma A^T$. Show that this is the case by explicitly calculating the covariance of the i and jth entries of $A\mathbf{x}$. It will help to know that

$$\begin{aligned}
cov(a_1x_1 + a_2x_2, b_1x_1 + b_2x_2) &= E[a_1x_1 + a_2x_2 - E(a_1x_1 + a_2x_2)][b_1x_1 + b_2x_2 - E(b_1x_1 + b_2x_2)] \\
&= E[a_1(x_1 - Ex_1) + a_2(x_2 - Ex_2)][b_1(x_1 - Ex_1) + b_2(x_2 - Ex_2)] \\
&= a_1b_1E(x_1 - Ex_1)^2 + a_2b_2E(x_2 - Ex_2)^2 + (a_1b_2 + b_2a_1)E(x_1 - Ex_1)(x_2 - Ex_2) \\
&= a_1b_1var(x_1) + a_2b_2var(x_2) + a_1b_2cov(x_1, x_2) + a_2b_1cov(x_2, x_1)
\end{aligned}$$

We want to compute $cov(A\mathbf{x}_i, A\mathbf{x}_j) = cov(\sum_{m=1}^n A_{im}x_m, \sum_{p=1}^n A_{jp}x_p)$. For ease of notation I will let the i^{th} and j^{th} rows of A be $\mathbf{a}$ and $\mathbf{b}$ respectively. We can write this out as an expectation and carry out the algebra to get:

$$\begin{aligned}
&E\left[\left(\sum_{m=1}^n a_mx_m - E\left(\sum_{m=1}^n a_mx_m\right)\right)\left(\sum_{p=1}^n b_px_p - E\left(\sum_{p=1}^n b_px_p\right)\right)\right] \\
=&E\left[\left(\sum_{m=1}^n a_m(x_m - E(x_m))\right)\left(\sum_{p=1}^n b_p(x_p - E(x_p))\right)\right] \\
=&E\left[\sum_{m=1}^n\sum_{p=1}^n a_mb_p(x_m - E(x_m))(x_p - E(x_p))\right] \\
=&\sum_{m=1}^n\sum_{p=1}^n a_mb_pE\left[(x_m - E(x_m))(x_p - E(x_p))\right] \\
=&\sum_{m=1}^n\sum_{p=1}^n a_mb_p\Sigma_{mp} \\
=&\sum_{m=1}^n a_m\left(\sum_{p=1}^n b_p\Sigma_{mp}\right) \\
=&\sum_{m=1}^n a_m\left(\Sigma\mathbf{b}^T\right) = \mathbf{a}\Sigma\mathbf{b}^T.
\end{aligned}$$

We can extend this result for arbitrary i and j to get that

$$cov(A\mathbf{x}, A\mathbf{x}) = A\Sigma A^T,$$

the desired result.

c. We have seen that our residuals $\hat{e}$ is orthogonal the predicted values $\hat{y}$; show that $\hat{e}$ is also uncorrelated with each column of X.

We need to consider the dot product of each column of X with $\hat{e}$, i.e. $X^T\hat{e}$. We can expand and simplify as

$$X^T\hat{e} = X^T(y - \hat{y}) = X^T(y - Hy) = X^T(y - X(X^TX)^{-1}X^Ty)$$
$$= X^Ty - X^TX(X^TX)^{-1}X^Ty = X^Ty - X^Ty = 0.$$

Since $\hat{e}$ is orthogonal to all columns of X, they must be uncorrelated.

d. Hence show that our estimate of the variance $\hat{\sigma}^2$ is uncorrelated with $\hat{\beta}$, confirming the plots in Question 3.

The only source of randomness in $\hat{\sigma}^2$ is the residuals, $\hat{e}$. We know that $\hat{\beta}$ lives in the column-space of X, i.e. it is a linear combination of the columns of X, and therefore from part e is orthogonal (and thus uncorrelated) to $\hat{e}$. Therefore $\hat{\sigma}^2$, a function of $\hat{e}$, will be uncorrelated with $\hat{\beta}$.

e. In linear regression, we generally assume $\epsilon \sim N(0, \sigma^2 I)$ but in spacial statistics and time series we sometimes allow the errors to be correlated: $\epsilon \sim N(0, \Sigma)$ instead. When this is the case, what is the variance of the least squares estimate $\hat{\beta}$?

We just carry out the variance/covariance calculation in the standard way:

$$var(\hat{\beta}) = var((X^TX)^{-1}X^Ty) = (X^TX)^{-1}X^T var(y) X((X^TX)^{-1})^T = (X^TX)^{-1}X^T\Sigma X(X^TX)^{-1}.$$

So the variance of the least squares estimate is $(X^TX)^{-1}X^T\Sigma X(X^TX)^{-1}$.