PHP 2550 - HW#2

Blain Morin

October 12, 2018

Part A

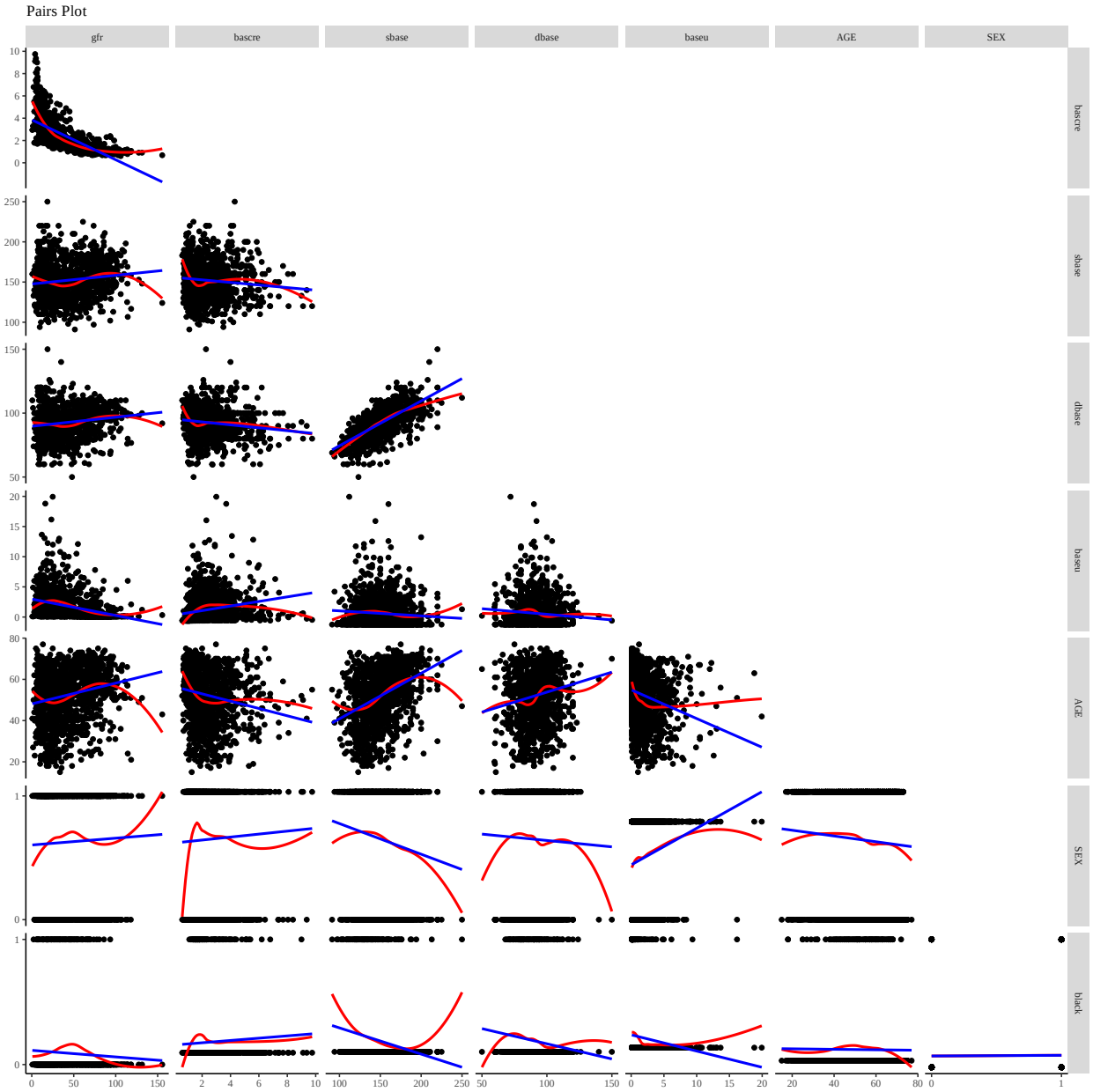
Before model fitting, we first did some exploratory analysis of the data. Here is a summary table of the variables relavant to the assignment (observations with missing values were dropped from analysis):

Table 1: Summary Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
gfr	1,249	42.59	28.38	0.70	17.93	66.00	155.50
bascre	1,249	2.33	1.40	0.60	1.26	3.00	9.75
sbase	1,249	152.18	23.08	91	135	169	250
dbase	1,249	92.73	11.54	50	85	100	150
baseu	1,249	1.82	2.31	0.10	0.10	2.68	19.93
AGE	1,249	52.40	13.10	15	43	62	77
SEX	1,249	0.63	0.48	0	0	1	1
black	1,249	0.09	0.29	0	0	0	1

We see that the data set contains 1,249 complete observations. The first six rows have the summary statistics for the continuous variables (in order: glomerular filtration rate, baseline creatine, systolic blood pressure, diastolic blood pressure, urine protein, and age). The last two rows are binary variables (sex = 1 if male, black = 1 if black). For the binary variables, the mean represents the proportion of observations with that characteristic.

We then looked at scatterplots between each of the variable:

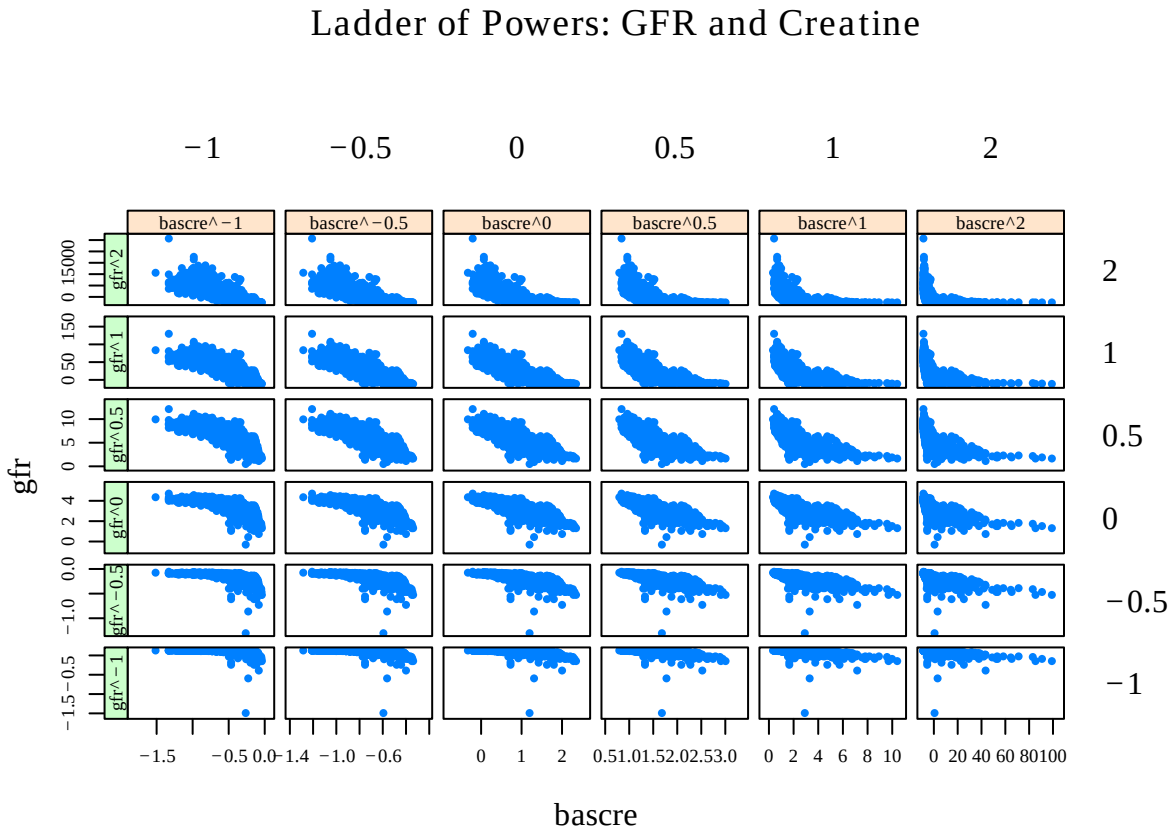


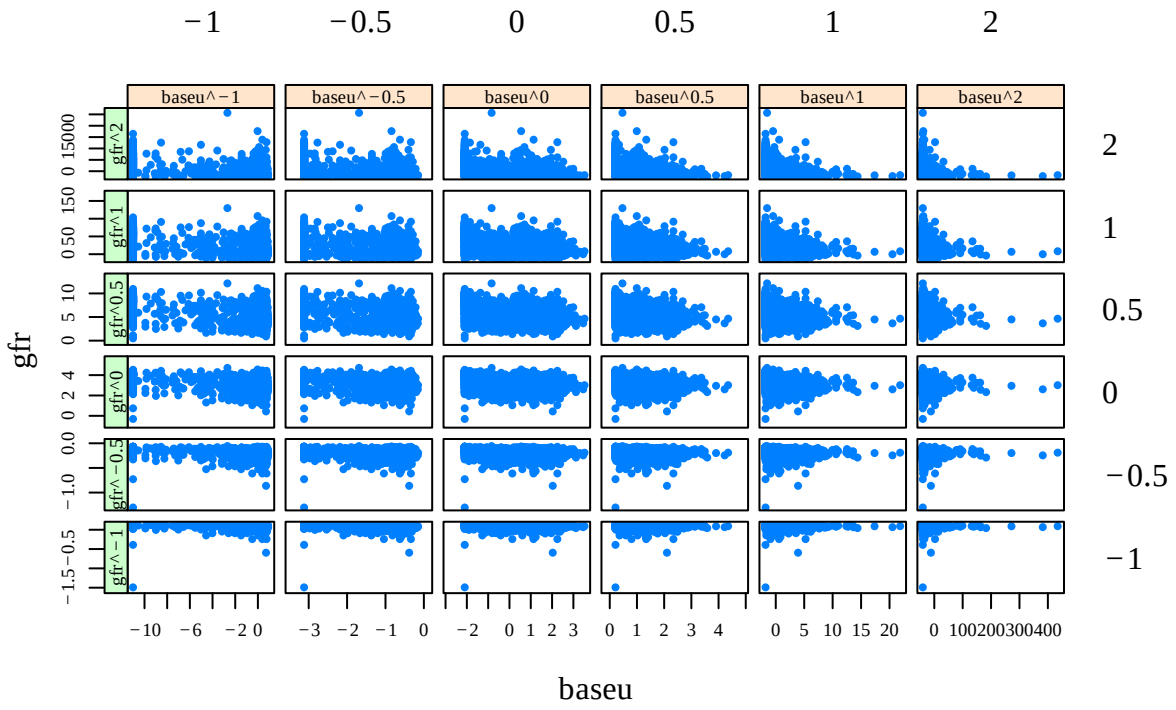
In the pairs plot above, the blue line is a linear model fit and the red line is a LOESS smooth fit. The first column of plots contains the relationship of our dependent variable (GFR) with all the independent variables in our data set. If the blue line has a positive slope, this indicates a positive relationship between GFR and the independent varbiable. For example, the positive slope in the GFR and sex plot may indicate the males have a higher GFR than women on average.

We see from the pairs plot that the relationship between GFR and baseline creatine and the relationship between GFR and baseline urine protien may not be linear. We also see in the baseline urine protien plot that the errors in the linear regression are higher for lower GFR values and they are for higher values. These heterogeneous erros would violate our linear model assumptions. We will try to fix these problems using transformations.

Another potential problem with our data is seen in the GFR and black plot. We see that there are observations for a larger range of GFRs for white people, whereas there are only observations of black people with low GFRs. This may represent a sampling bias, which would affect the validity of our counclusions.

We then explored transformations of GFR, baseline creatine, and baseline urine protein using ladder plots (which create scatterplots of all the combinations of common transformations):

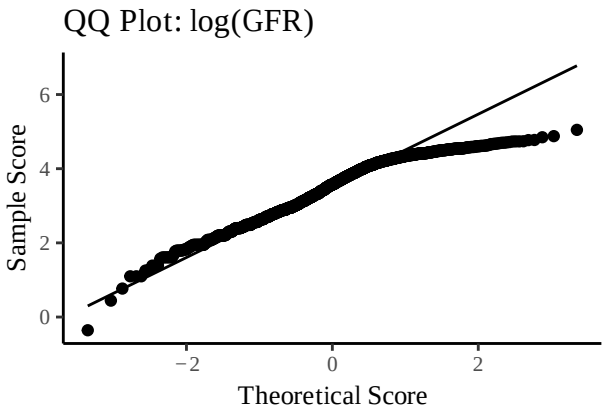


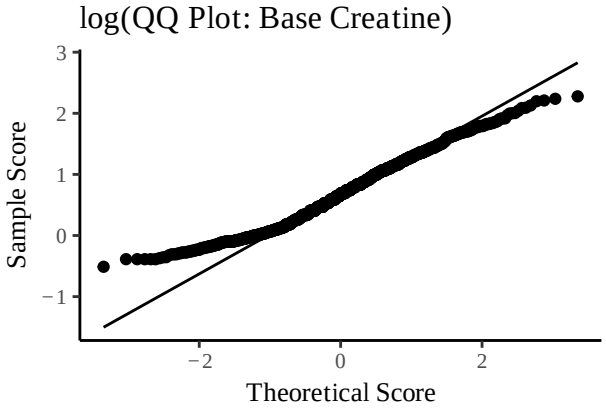


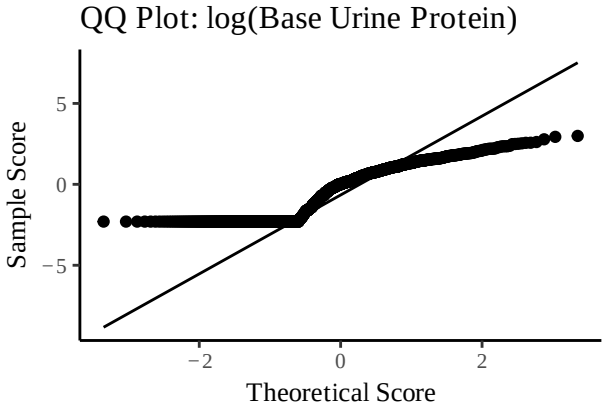
In the ladder plots above, we are searching for the combinations that make a linear trend and create a good spread. From visual inspection it, it looks like using a log transformation for GFR, creatine and urine protein may be a good transformation candidate.

We checked these transformations on quantile-quantile plots:

Transformation Choices







In the QQ plots above, the diagonal line represents a perfect normal distribution. We see that the distribution of log(GFR) is a bit left skewed. For log(creatine), the sample quantile is less spread out than a normal distribution, which indicates that the sample data has thin tails. In the urine protein plot, we see that many of the points in the lower quantiles have the same value. This horizontal portion may represent a measurement threshold cutoff.

We then ran three linear models. We did not include diastolic blood pressure in any of the models because of its high correlation with systolic blood pressure. In the first model we regressed an untransformed GFR on all the other independent variables (except diastolic). In the second model, we used transformed variables as identified above. The third model is uses transformed variables, but drops urine protein from the regression (because of the lack of normality seen in its QQ plot). Here are the results from the three regressions:

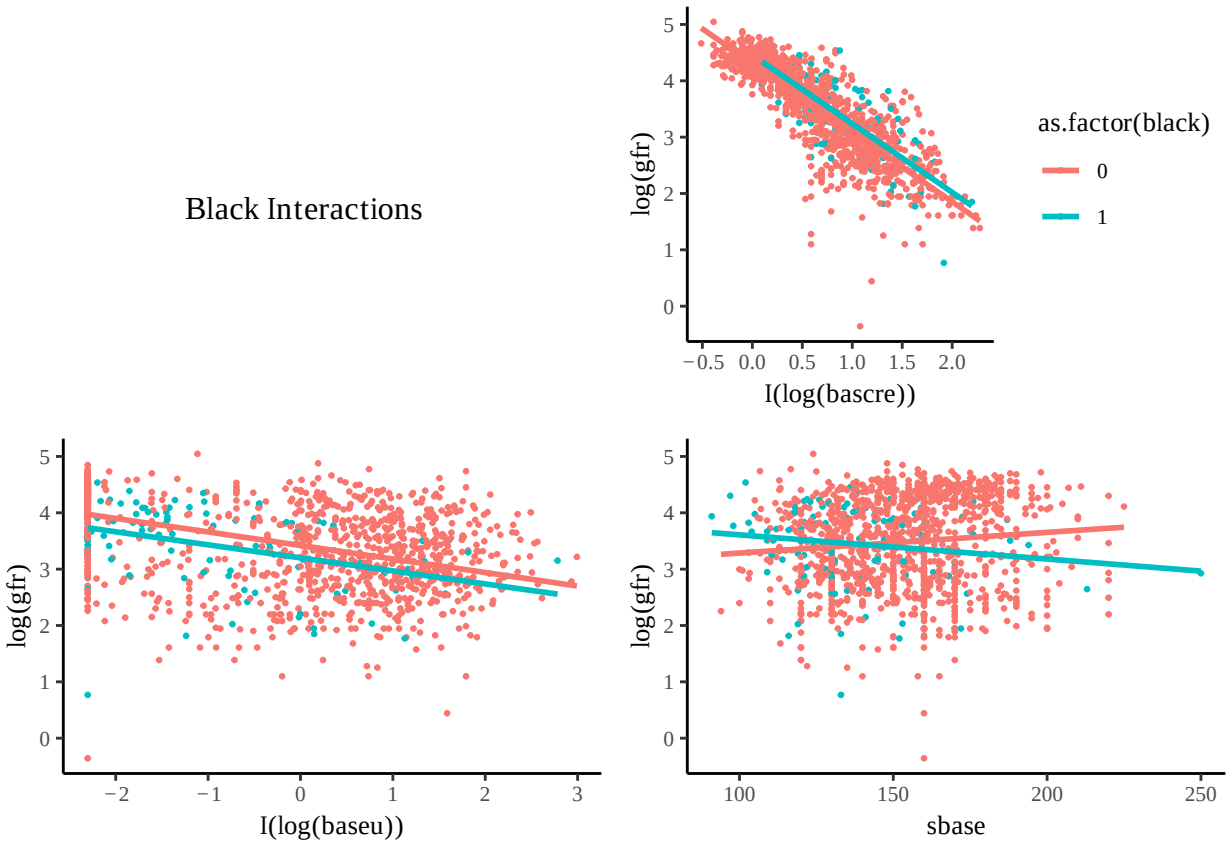
Table 2:

	Dependent variable:		
	gfr	log(gfr)	
	(1)	(2)	(3)
bascre	-13.614^{***}		
	(0.391)		
baseu	-2.391^{***}		
	(0.246)		
log(bascre)		-1.207^{***}	-1.252^{***}
		(0.025)	(0.022)
log(baseu)		-0.039^{***}	
		(0.009)	
sbase	0.055^{**}	-0.0002	-0.0002
	(0.025)	(0.001)	(0.001)
AGE	0.067	-0.002^{**}	-0.001
	(0.045)	(0.001)	(0.001)
as.factor(SEX)1	5.690^{***}	0.240^{***}	0.225^{***}
	(1.115)	(0.025)	(0.024)
as.factor(black)1	-3.393^{*}	0.115^{***}	0.153^{***}
	(1.889)	(0.042)	(0.042)
Constant	63.589^{***}	4.285^{***}	4.273^{***}
	(4.124)	(0.089)	(0.090)
Observations	1,249	1,249	1,249
R^2	0.570	0.738	0.734
Akaike Inf. Crit.	10,861.770	1,336.823	1,351.441
Note:		*p<0.1; **p<0.05; ***p<0.01	
		Standard errors in ()	

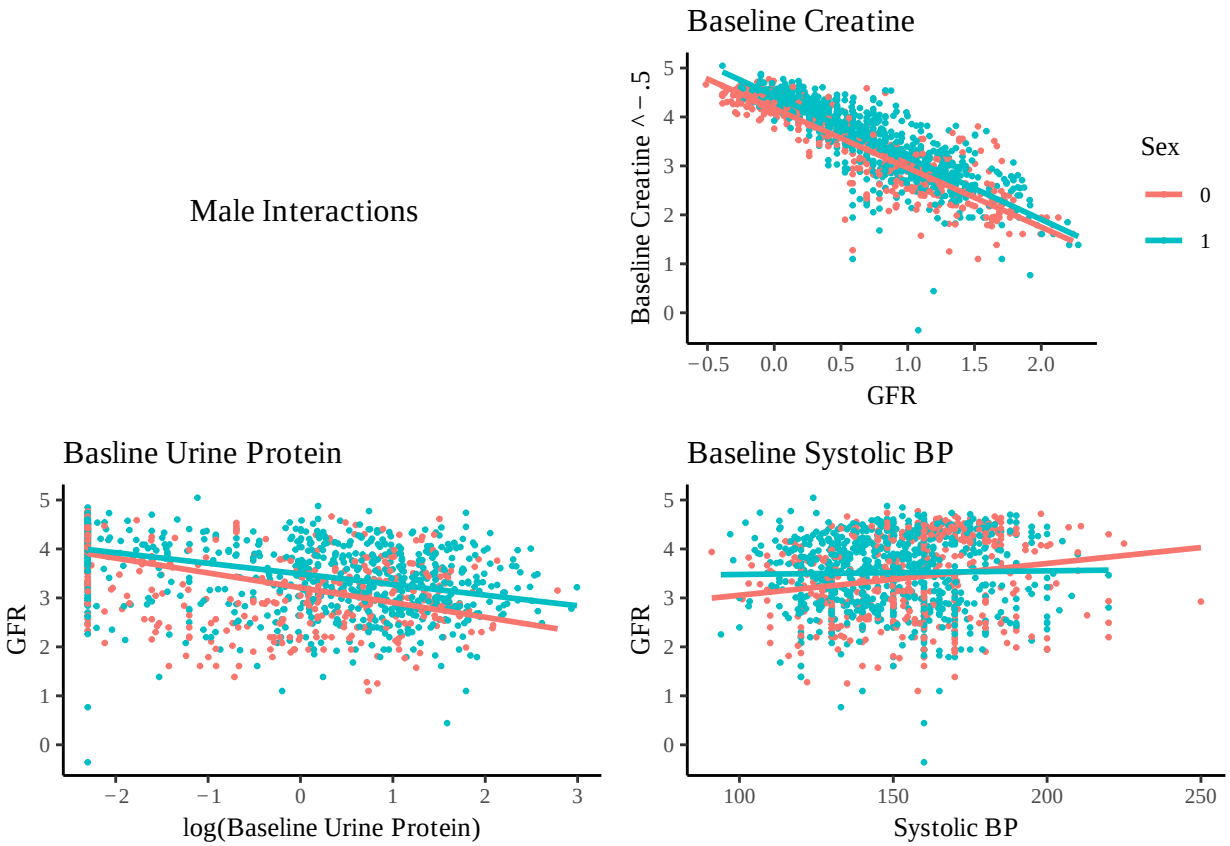
We see from the improvement in r squared that model 2 and 3, which used transformed variables, is a better fit than model 1, which does not use transformed variables. Since the AIC is lower in model 2, we know that including log(urine protein) significantly improves our model fit, despite the possible measurement errors found above.

Next, we checked for variable interactions. From intuition, we decided to graphically check for interactions between race and creatine, urine protein, and blood pressure and also interactions between sex and creatine, urine protein, and blood pressure:

Black Interactions



Male Interactions



In the scatter plots above, a significant interaction would surface as a difference in slopes between the two groups. Visually, it appears that there may be a significant interaction between race and blood pressure and also a significant interaction between sex and blood pressure. To check this significance, we added the two interactions to model (2) from above:

Table 3:

	Dependent variable:
	log(gfr)
I(bascre^(-0.5))	3.468***
	(0.072)
log(baseu)	−0.019**
	(0.010)
sbase	0.001
	(0.001)
AGE	−0.003**
	(0.001)
as.factor(SEX)1	0.516***
	(0.162)
as.factor(black)1	0.364
	(0.248)
sbase:as.factor(black)1	−0.001
	(0.002)
sbase:as.factor(SEX)1	−0.002
	(0.001)
Constant	0.777***
	(0.139)
Observations	1,249
R^2	0.734
Akaike Inf. Crit.	1,359.696
Note:	*p<0.1; **p<0.05; ***p<0.01

We see that the interaction terms are not significant.

We can also brute force check every combination of second order interactions:

Table 4: All Second Order Interactions

	Df	Sum Sq	Mean Sq	F value	p
log(bascre)	1	571.5904847	571.5904847	3410.7639860	0.0000000
log(baseu)	1	1.2328892	1.2328892	7.3568300	0.0067742
sbase	1	1.2274195	1.2274195	7.3241917	0.0068976
AGE	1	0.5527289	0.5527289	3.2982143	0.0695992
as.factor(SEX)	1	16.3375713	16.3375713	97.4886767	0.0000000
as.factor(black)	1	1.2382428	1.2382428	7.3887760	0.0066557
log(bascre):log(baseu)	1	1.3376246	1.3376246	7.9818013	0.0048015
log(bascre):sbase	1	0.0442876	0.0442876	0.2642708	0.6072937
log(bascre):AGE	1	0.3032074	0.3032074	1.8092831	0.1788428
log(bascre):as.factor(SEX)	1	0.6000198	0.6000198	3.5804056	0.0587001
log(bascre):as.factor(black)	1	0.0833793	0.0833793	0.4975366	0.4807184
log(baseu):sbase	1	0.0197923	0.0197923	0.1181035	0.7311595
log(baseu):AGE	1	0.1103703	0.1103703	0.6585954	0.4172136
log(baseu):as.factor(SEX)	1	0.2274625	0.2274625	1.3573022	0.2442320
log(baseu):as.factor(black)	1	0.0069996	0.0069996	0.0417677	0.8380971
sbase:AGE	1	0.2626262	0.2626262	1.5671290	0.2108630
sbase:as.factor(SEX)	1	0.8034322	0.8034322	4.7941974	0.0287436
sbase:as.factor(black)	1	0.2121114	0.2121114	1.2656999	0.2607949
AGE:as.factor(SEX)	1	0.1934202	0.1934202	1.1541663	0.2828899
AGE:as.factor(black)	1	0.2114347	0.2114347	1.2616616	0.2615565
as.factor(SEX):as.factor(black)	1	0.5085403	0.5085403	3.0345343	0.0817605
Residuals	1227	205.6259324	0.1675843	NA	NA

The rows highlighted in red in the table above show the variables with p values less than .05. We see the the interaction between log baseline creatine and lod baseline urine protein is significant. Adding this interaction to our model (2) from above, the new regression results are:

Table 5:

	Dependent variable:
	log(gfr)
log(bascre)	-1.188^{***}
	(0.025)
log(baseu)	-0.066^{***}
	(0.014)
sbase	-0.0005
	(0.001)
AGE	-0.002^{**}
	(0.001)
as.factor(SEX)1	0.242^{***}
	(0.025)
as.factor(black)1	0.125^{***}
	(0.042)
log(bascre):log(baseu)	0.047^{***}
	(0.017)
Constant	4.306^{***}
	(0.090)
Observations	1,249
R^2	0.739
Akaike Inf. Crit.	1,330.863
Note:	*p<0.1; **p<0.05; ***p<0.01

Although the new model has a better fit than model (2), the improvment is small. Since we did multiple comparisons to find the added interaction, we should be wary of including it in the model. Because the effect size and AIC improvement is relatively small, we elected to exclude the log baseline creatine and log baseline urine protein from our regression. Overall, our best model choice is model (2):

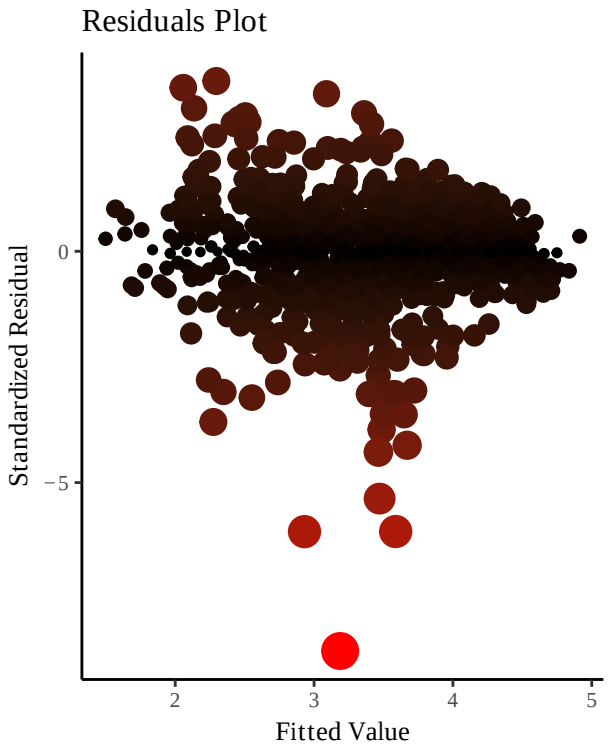
Table 6: Best Model

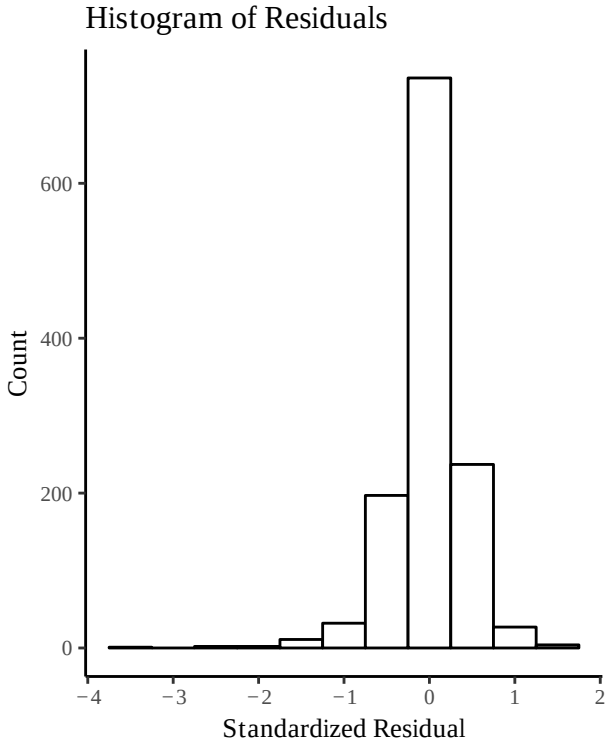
	Dependent variable:
	log(gfr)
log(bascre)	-1.207^{***}
	(0.025)
log(baseu)	-0.039^{***}
	(0.009)
sbase	-0.0002
	(0.001)
AGE	-0.002^{**}
	(0.001)
as.factor(SEX)1	0.240^{***}
	(0.025)
as.factor(black)1	0.115^{***}
	(0.042)
Constant	4.285^{***}
	(0.089)
Observations	1,249
R^2	0.738
Akaike Inf. Crit.	1,336.823
Note:	*p<0.1; **p<0.05; ***p<0.01
	Standard errors in ()

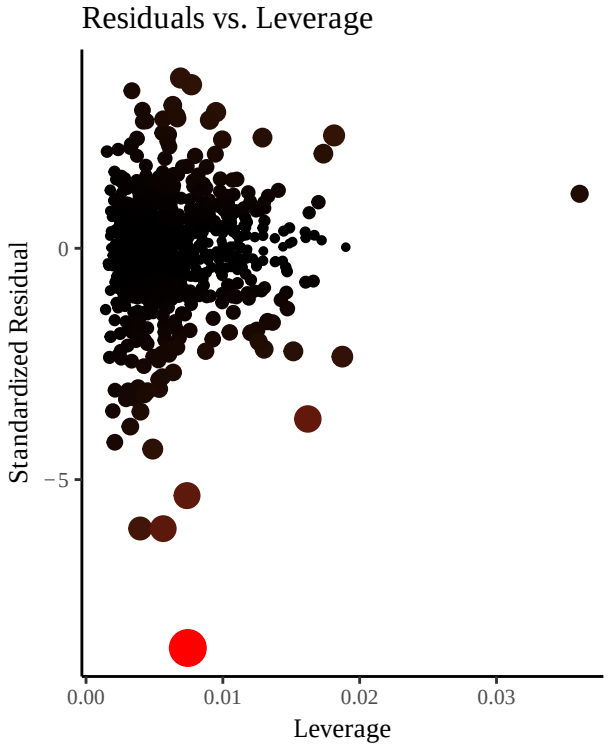
The coefficient for log(bascre) means that for a 10% increase in baseline creatine, we expect an 11% decrease in GFR (1.1^-1.207 = .89). The coefficient for log(baseu) has a similar interpretation: for a 10% increase in baseline urine protein, we expect a .4% decrease in GFR (1.1^.039 = .996). The coefficient on systolic blood pressure means that for a one unit increase of blood pressure, we expect a .0002% decrease in GFR (exp(-.0002 = .9998)). The coefficient on age means that for every additional year of age, we expect a .002% decrease in GFR (exp(-.002 = .998)). The coefficient on sex means that we expect males to have a 27% higher GFR measurements than those of females (exp(.24) = 1.27). The coefficient on black means that we expect blacks to have a 12% higher GFR measurement than non-blacks (exp(.115) = 1.12).

Overall, all of the variables are significant at the .05 level except for systolic blood pressure. Lastly, we checked some regression diagnostics:

Diagnostic Plots







In the "Residuals Plot", we see that the variance of the residuals appears to be higher in the middle of the distribution, which violates the assumptions of the linear model. The large points in red indicate highlight the more extreme residuals. These red values should be further investigated.

The histogram of residuals shows that the distrubution is left skewed. The skewness likely comes from the points that have the high residuals seen in the "Residuals Plot."

In the "Residuals vs. Leverage" plot, the larger red points represent the more influential points (in terms of Cook's Distance). We see that the most extreme residual is highly influential. Again, this observation should be investigated in more detail.

Part B

1. First you will explore type 1 and type 2 errors with a simple regression.

a. Simulate a random vector y of length 100 in which each element follows a normal distribution with mean 10 and variance 4. Then simulate a random vector x of length 100 with mean 3 and variance 1.

To simulate the random vectors, we used the rnorm function in R:

```
### Set seed for reproducibility
set.seed(10)

### Create vectors with 100 random draws
y = rnorm(n =100, mean = 10, sd = 2)
x = rnorm(n = 100, mean = 3, sd = 1)
```

b. Regress y on x and save the p-value for the slope (HINT: use the summary.lm function or summary(lmobject) and use the coef element of the summary object if using R)

The following R code regresses y on x and extracts the p values from the regression:

```
### Regress y on x
reg1 = lm(y ~ x)

### Extract the p value
reg1.sum = summary.lm(reg1)
reg1.p = reg1.sum$coefficients[2,4]

### Output p value
kable(reg1.p, format = "latex", col.names = "p value")
```

p value
0.5662843

We observe that the p value from the x coefficient is .566.

c. What do you think this p-value should be?

We expect the p value to be greater than .05, since we know that x and y are independent.

d. Now replicate this procedure 1000 times to develop a simulation.

We used loops to create X and Y matrices such that each column of the matrix represents a vector of 100 random draws. We then used a loop to regress each column of Y on its corresponding column in X. Here is the R code for our simulation:

```
### Set seed for reproducibility
set.seed(88)

### Set number of draws and number of simulations
n = 100
n.sims = 1000


### Initialize a matrix for x and y
xs = matrix(0, nrow = n, ncol = n.sims)
ys = matrix(0, nrow = n, ncol = n.sims)


### Draw 100 Xs 1000 times and store them in the X matrix
for (i in 1:n.sims) {

  xs[,i] = rnorm(n = 100, mean = 3, sd = 1)

}

### Draw 100 Ys 1000 times and store them in the y matrix
for (i in 1:n.sims) {

  ys[,i] = rnorm(n =100, mean = 10, sd = 2)

}


### Initialize a vector to store p values
ps = c()

### Iteratively regress a column of the y matrix
### on a column of the X matrix, then extract the p value
### and store it in the p vector
for (i in 1:n.sims) {

  reg = lm(ys[,i] ~ xs[,i])
  p = summary.lm(reg)
  reg.p = p$coefficients[2,4]
  ps[i] = reg.p

}
```

e. Calculate the proportion of times the p-value is less than 0.05. How does this match with your intuition? [HINT: Should the slope be related to the outcome? What type of error are you simulating?]

To calculate the proportion, we divided the count of p values $< .05$ by 1000 (the number of simulations):

```
### Calculate the proportion of p values which are less than .05
answer.e = length(which(ps < .05)) / n.sims

### Output Answer
```

```
kable(answer.e, format = "latex", col.names = "Proportion of p < .05") %>%
  kable_styling(latex_options = "HOLD_position")
```

Proportion of p < .05
0.048

We observe that the proportion of p values less than .05 is .048. This matches our earlier intiution that most of the p values should be higher than .05. The slope of the regression should not be related to the outcome because Y and X are independent. In other words, we know that the null hypothesis is true. The regressions that rejected the null are examples of Type I error (they are false positives).

f. Now simulate y and x together so that the conditional mean of y given x is 10+x and the conditional variance is 1. Repeat b-e above for this distribution. How does your answer differ from the first simulation? What are you simulating now? [HINT: Consider if the slope is actually related to the outcome]

We then repeated the simulation process, but this time the draws of Y were drawn from a normal distribution with a mean of 10+x and variance of 1. We then calculated the proportion of p values less than .05. Here is the R code for this simulation and calculation:

```
### Set seed for reproducibility
set.seed(356)


### Initialize x and y matrices
xs2 = matrix(0, nrow = n, ncol = n.sims)
ys2 = matrix(0, nrow = n, ncol = n.sims)


### Draw Xs and store as columns in X matrix
for (i in 1:n.sims) {

  xs2[,i] = rnorm(n = 100, mean = 3, sd = 1)

}


### Draw Ys that have a mean taken from the corresponding X location + 10
for (j in 1:n.sims) {

  for (i in 1:n) {

    ys2[i,j] = rnorm(n = 1, mean = xs2[i,j] + 10, sd = 1)

  }


}

### Initialize a vector for p values
ps2 = c()
```

```
### Iteratively regress a column of the y matrix
### on a column of the X matrix, then extract the p value
### and store it in the p vector
for (i in 1:n.sims) {

  reg = lm(ys2[,i] ~ xs2[,i])
  p = summary.lm(reg)
  reg.p = p$coefficients[2,4]
  ps2[i] = reg.p


}

### Calclulate the proportion of p values less than .05
answer2 = length(which(ps2 < .05)) / n.sims

### Output the proportion
kable(answer2, format = "latex", col.names = "Proportion of p < .05") %>%
  kable_styling(latex_options = "HOLD_position")
```

Proportion of p < .05
1

We see that the proportion of p values less than .05 is 1 for this simulation. This time, we know that Y is conditional on X. We thus expect the slope coefficient to be related to the outcome. Since we know that the alternative hypothesis is true (that the slope is non zero), this simulation shows us the statistical power of our regression. Since the regression found statistical significance in all of our trials, we know that we have high power. In other words, the probability of type 2 error is low.

2. Now we will investigate overfitting which occurs when you are making too complex a model. First we will simulate some data in which y and multiple x's are unrelated.

a. Extend the function you have written to carry out exercise 1 so that you now generate the same y vector as in 1a (i.e. 100 draws from N(10,4)) but you now generate 100 draws from an 3-variate multivariate normal X matrix with means 1, 2, 3 and covariance matrix equal to the identity matrix of rank 3 (i.e., each X variable has variance 1 and they are uncorrelated). [HiNT: use the mvrnorm function in the MASS library to generate multivariate normal draws].

b. Regress y on the X matrix and save the p-values from this regression. In R if you specify X as a matrix in the call to the lm() function, it will automatically use the formula y~x[,1]+x[,2]+... i.e. it will use the columns of the matrix as the predictors.

We followed the same procedure in part 1, except now the X variables are drawn from a 3 variate normal distribution. Here is the R code for this simulation:

```
### Set seed for reproducibility
set.seed(100)

### Set number of parameters and create sigma
params = 3
sigma = diag(1, nrow = 3, ncol = 3)
```

```
### Initialize a matrix for the p values to be stored in
mv.ps = matrix(0, nrow = n.sims, ncol = params)


### Draw Xs and Ys
### Regress Y on X
### Extract P values
### Store p values in the p value matrix
for (i in 1:n.sims) {

  x = mvrnorm(n = 100, mu = c(1,2,3), Sigma = sigma)
  y = rnorm(n =100, mean = 10, sd = 2)
  reg = lm(y ~ x)
  sum.reg = summary.lm(reg)
  reg.p = sum.reg$coefficients[I(1:params+1), 4]
  mv.ps[i,] = reg.p

}
```

c. Determine how many of the p-values for each variable are significant at the 0.05 level.

Here is the R code that calculates the proportion of p values less than .05 for each variable:

```
### Calculate the proportion of p values less than .05
### for each X (stored as columns)
answer3 = apply(mv.ps, 2, function(x) length(which(x < .05))/n.sims)

### Output p values
kable(t(as.data.frame(answer3)), format = "latex",
      col.names = c("X1", "X2", "X3"),
      caption = "Proportion of p values < .05",
      row.names = FALSE) %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 7: Proportion of p values $< .05$

X1	X2	X3
0.044	0.055	0.063

Since all of the variables are known to be independent, we expect these proportions to be around .05 (like it was in part 1). We observe that the proportions match our expectations.

d. Compute the minimum p-value for each regression and calculate how many times the minimum p-value is less than 0.05. Why is this answer different from those in question c?

Next, we grab the minimum p value from each regression and calculate the proportion of times the minimum value is less than .05:

```
### Initialize a vector to store minimum p values
min.p = c()
```

```
### Extract minimum p value for each regression
for (i in 1:n.sims) {

  min.p[i] = min(mv.ps[i,])

}

### Calculate the proportion where the minimum p value is < .05
answer4 = length(which(min.p < .05))/1000

### Output proportion
kable(answer4, format = "latex", col.names = "Proportion of min(p) < .05") %>%
  kable_styling(latex_options = "HOLD_position")
```

Proportion of min(p) < .05
0.151

We see that the proportion of minimum pvalues < .05 is .151. This number is different than the individual proportions because we are looking at p values from three variables instead of one. We see that this about triples the chance of finding a p value less than .05.

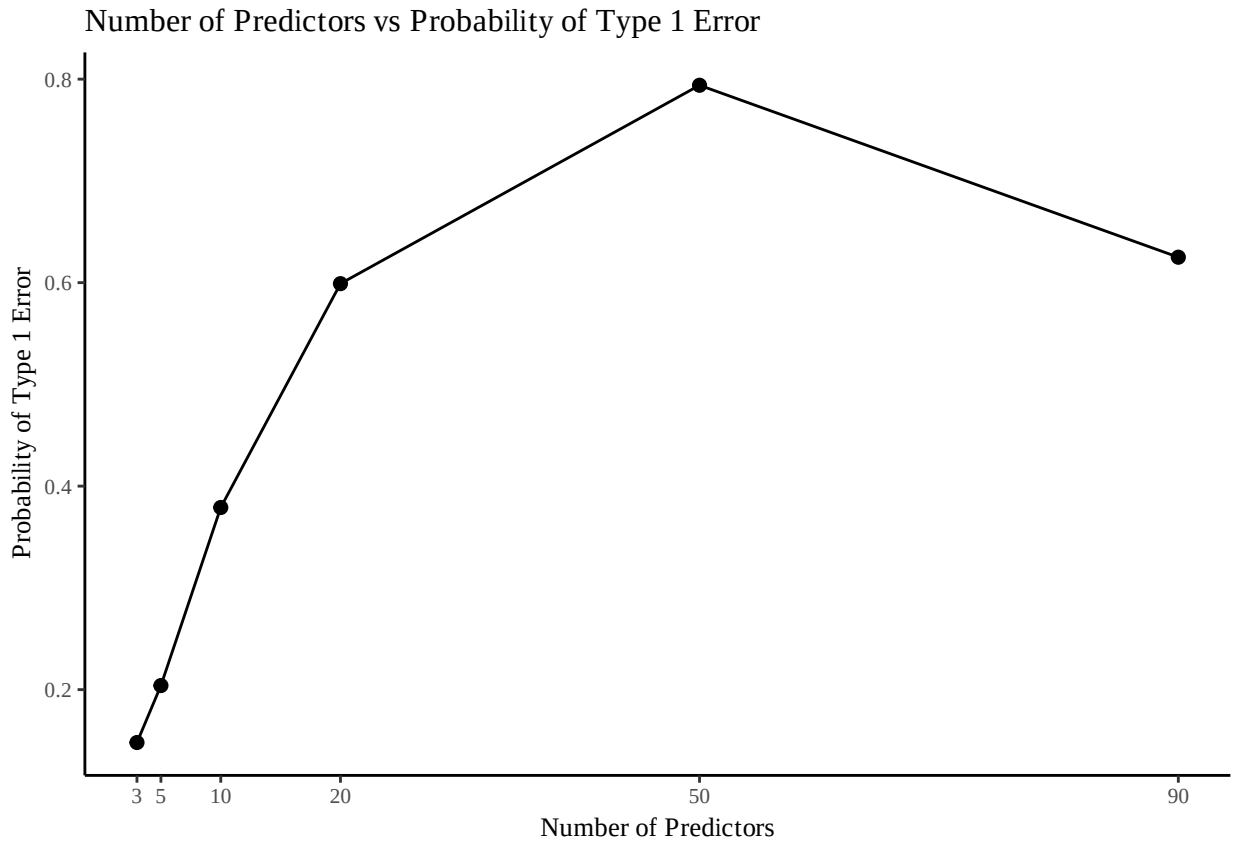
e. Now repeat this exercise changing the number of predictors P so that you do it for P = 3, 5, 10, 20, 50, 90. Compute the minimum p-value in each simulated regression and plot the number of times you find at least one significant variable for each P. What pattern do you see? How do you explain this?

We repeated the simulation for a vector of parameters:

```
### Set seed for reproducibility
set.seed(4)

### Vector of parameters to simulate
params = c(3, 5, 10, 20, 50, 90)

### Initialize a matrix to store results from each simulation
results = matrix(0, nrow = length(params), ncol = 2)

### Create sigma with dims that equal number of variables
### Initialize a matrix to store p values for each simulation
for (i in 1:length(params)){

  sigma = diag(1, nrow = params[i], ncol = params[i])
  results[i, 1] = params[i]
  mv.ps = matrix(0, nrow = n.sims, ncol = params[i])

### Draw Xs and Ys
### Regress Y on X
### Extract p values and store in mv.ps

  for (j in 1:n.sims) {

    x = mvrnorm(n = 100, mu = 1:params[i], Sigma = sigma)
    y = rnorm(n =100, mean = 10, sd = 2)
```

```
    reg = lm(y ~ x)
    sum.reg = summary.lm(reg)
    reg.p = sum.reg$coefficients[I(1:params[i]+1), 4]
    mv.ps[j,] = reg.p

  }

### Grab the minimum p from each regression
### Calculate the proportion of min(p) < .05
### Store the p in our results matrix

  min.p = apply(mv.ps, 1, min)
  results[i, 2] = length(which(min.p < .05))/n.sims



}
```

Here is a line plot of the results from our simulation:



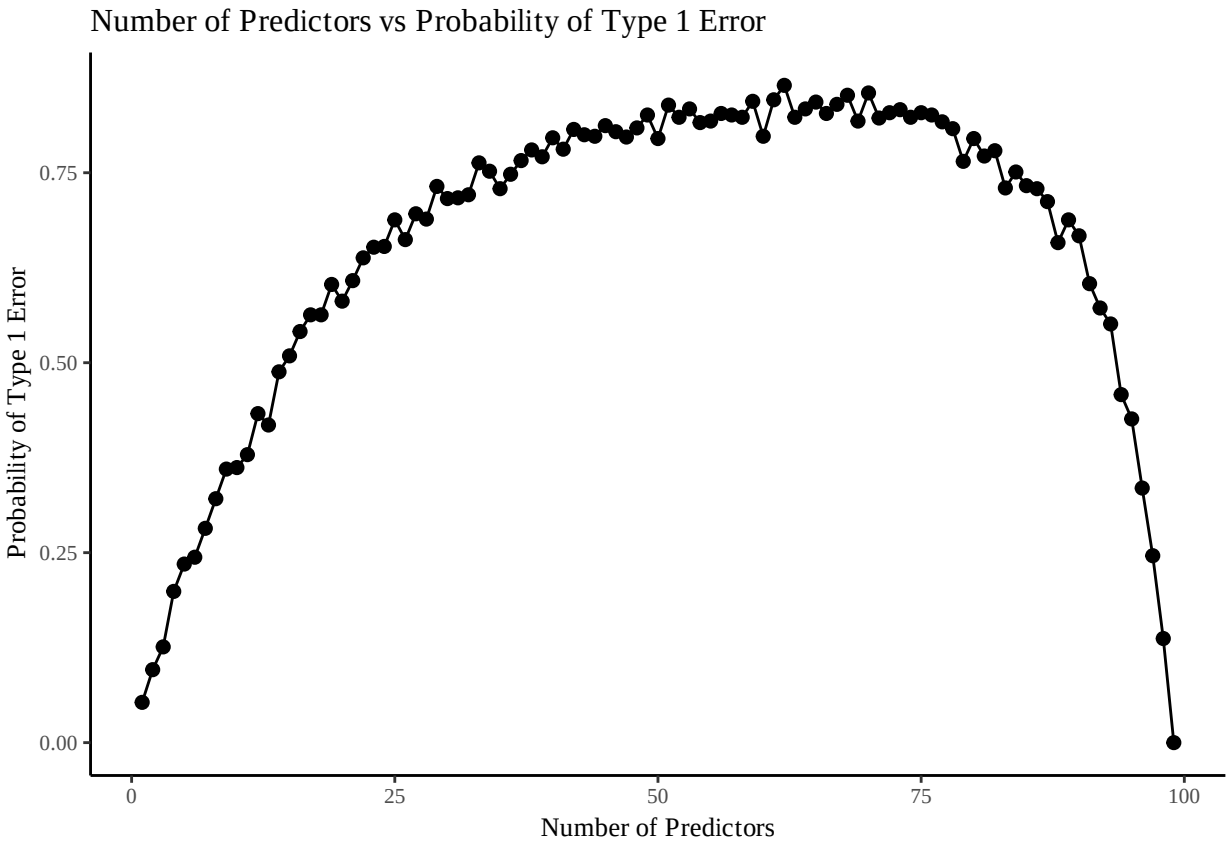
We see that the probability of type 1 error increases as the number of predictors goes to 50. After 50, the probability of type 1 error decreases.

e. Set P equal to every number from 1 to 99 (i.e. you try models with all possible number of predictors) and run more simulations to reduce simulation error.

Next, we repeate the above simulation using the number of predictors from 1 to 99:

```
### Set seed for reproducibility
set.seed(4)

### Set the number of predictors to check
params = 1:99

### Initialize a matrix to store our results in
results2 = matrix(0, nrow = length(params), ncol = 2)


### Create sigma with dims that equal number of variables
### Initialize a matrix to store p values for each simulation
for (i in 1:length(params)){

  sigma = diag(1, nrow = params[i], ncol = params[i])
  results2[i, 1] = params[i]
  mv.ps = matrix(0, nrow = n.sims, ncol = params[i])

### Draw Xs and Ys
### Regress Y on X
### Extract p values and store in mv.ps

  for (j in 1:n.sims) {

    x = mvrnorm(n = 100, mu = 1:params[i], Sigma = sigma)
    y = rnorm(n =100, mean = 10, sd = 2)
    reg = lm(y ~ x)
    sum.reg = summary.lm(reg)
    reg.p = sum.reg$coefficients[I(1:params[i]+1), 4]
    mv.ps[j,] = reg.p

  }

### Grab the minimum p from each regression
### Calculate the proportion of min(p) < .05
### Store the p in our results matrix

  min.p = apply(mv.ps, 1, min)
  results2[i, 2] = length(which(min.p < .05))/n.sims


}
```

Again, we create a line plot with the reults from our simulation:



We see that the probability of type 1 error increases until the number of predictors is around 75. We see that the probability decreases steeply after. In our simulation, the sample size for each regression is 100. As the number of predictors approaches the number of observations, the degrees of freedom approach 0. This in effect widens the confidence intervals around each slope estimate, which reduces power. Since we decreases the probability of declaring any of the predictors significant, we thus decrease the chance of a false positive. Thus, the probability of type 1 error decreases when the number of predictors approaches the sample size.