PHP 2550 - HW#2

Blain Morin

October 12, 2018

Part A

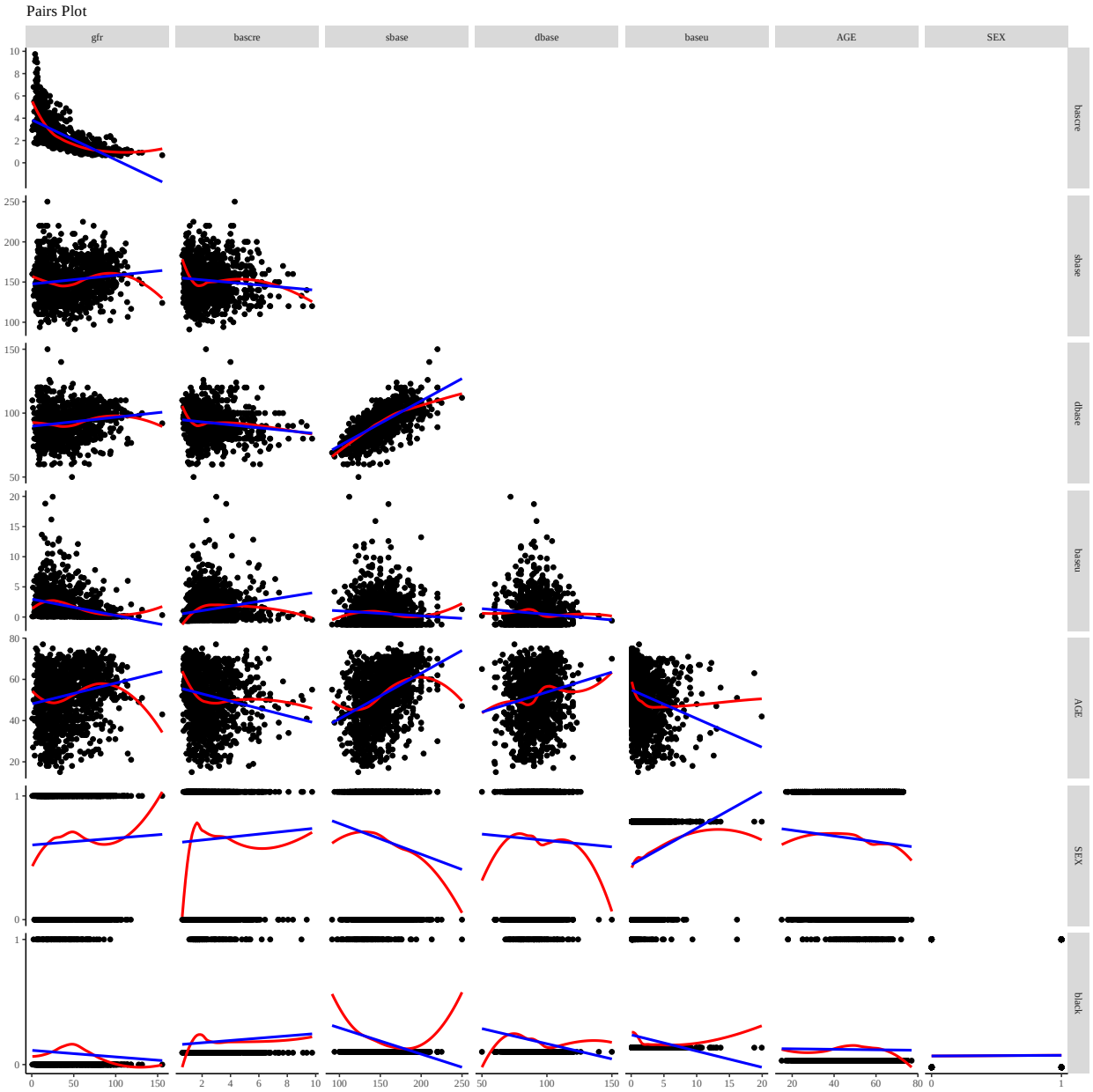
Before model fitting, we first did some exploratory analysis of the data. Here is a summary table of the variables relavant to the assignment (observations with missing values were dropped from analysis):

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:

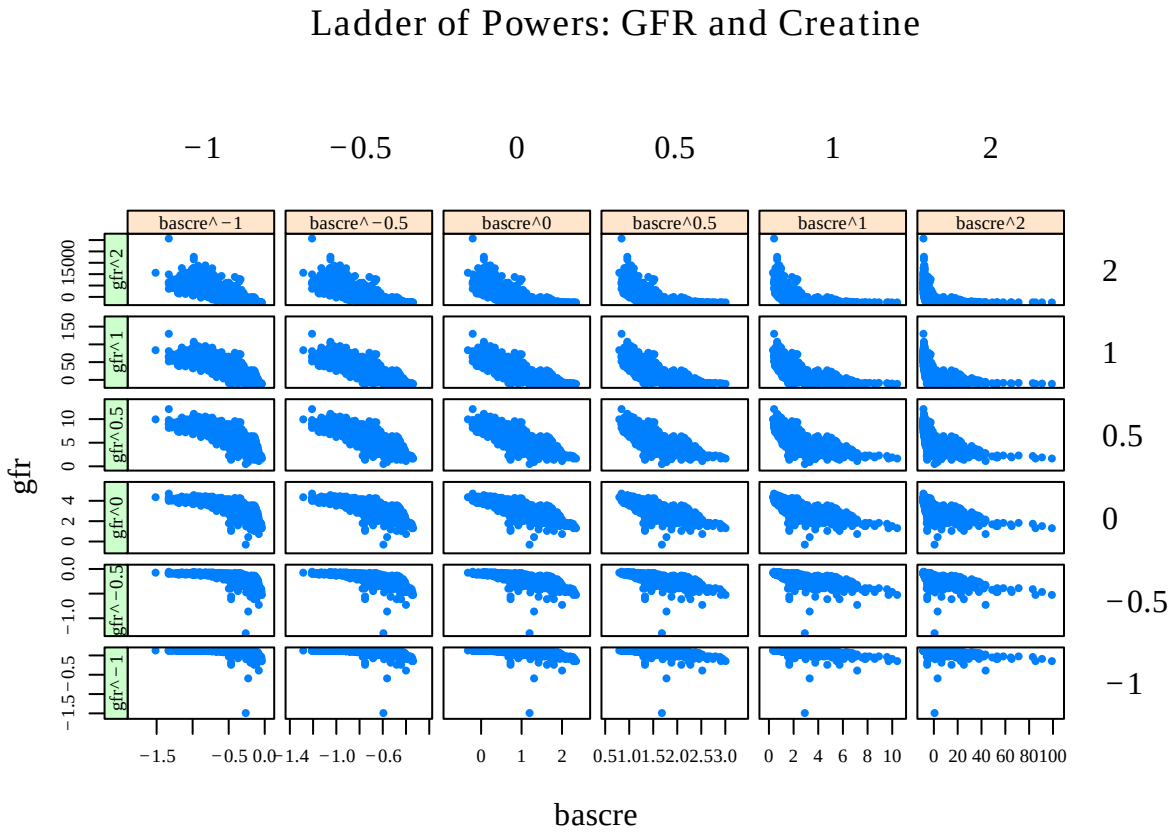


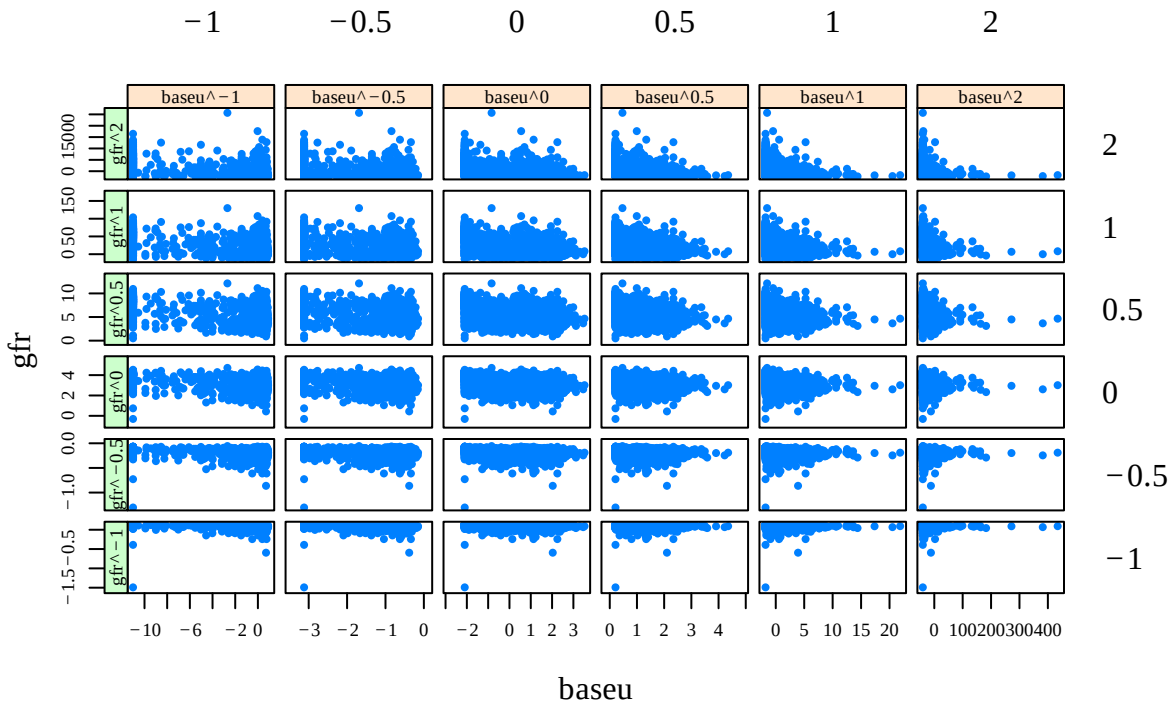
In the pairs plot above, the blue line is a linear model fit and the red line is a LOESS smooth fit. The first column of plots contains the relationship of our dependent variable (GFR) with all the independent variables in our data set. If the blue line has a positive slope, this indicates a positive 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):

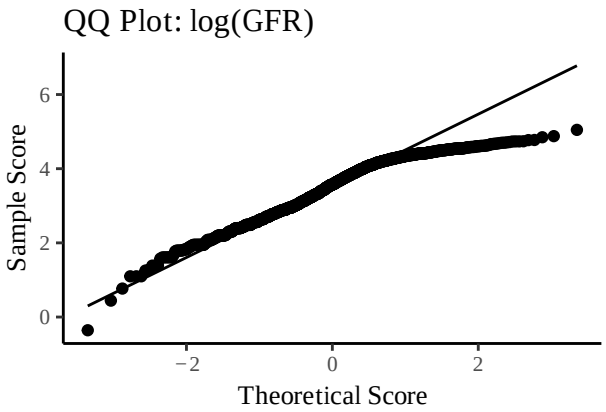


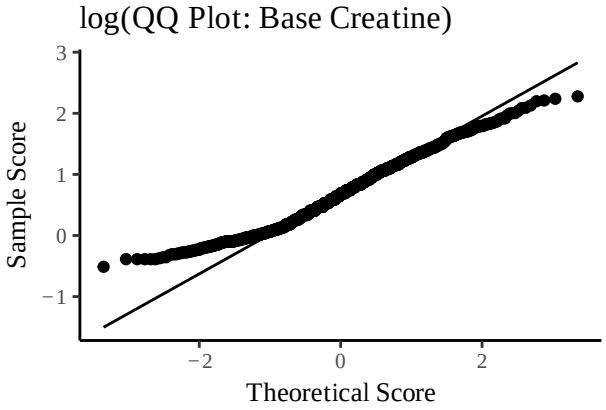


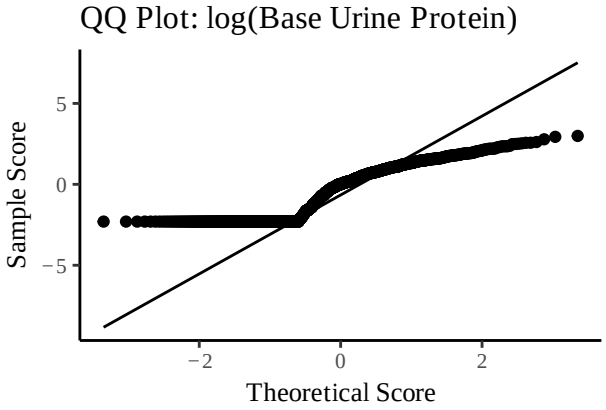
In the ladder plots above, we are searching for the combinations that make a linear trend and create a good spread. From visual inspection it, it looks like using a log transformation for GFR, creatine and urine protein may be a good transformation candidate.

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:

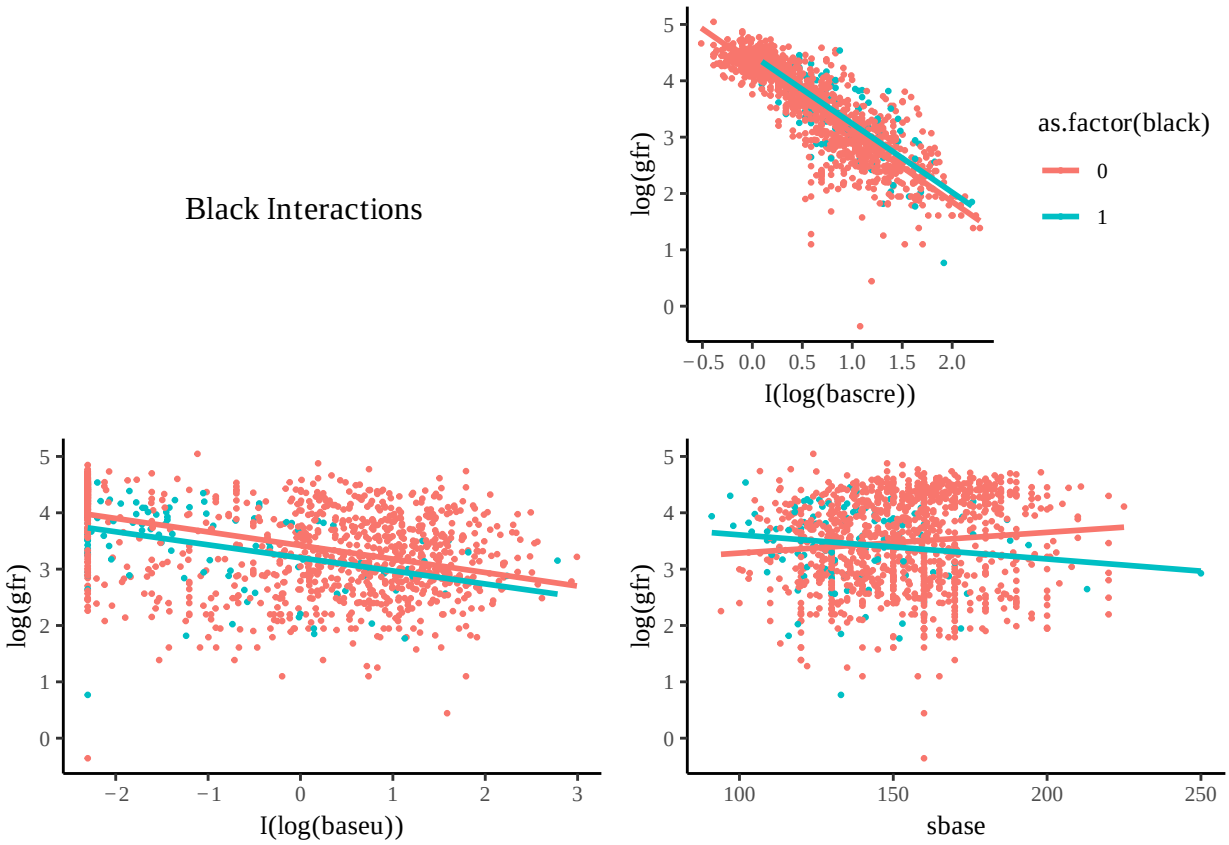
We see from the improvement in r squared that model 2 and 3, which used transformed variables, is a better fit than model 1, which does not use transformed variables. Since the AIC is lower in model 2, we know that including log(urine protein) significantly improves our model fit, despite the possible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

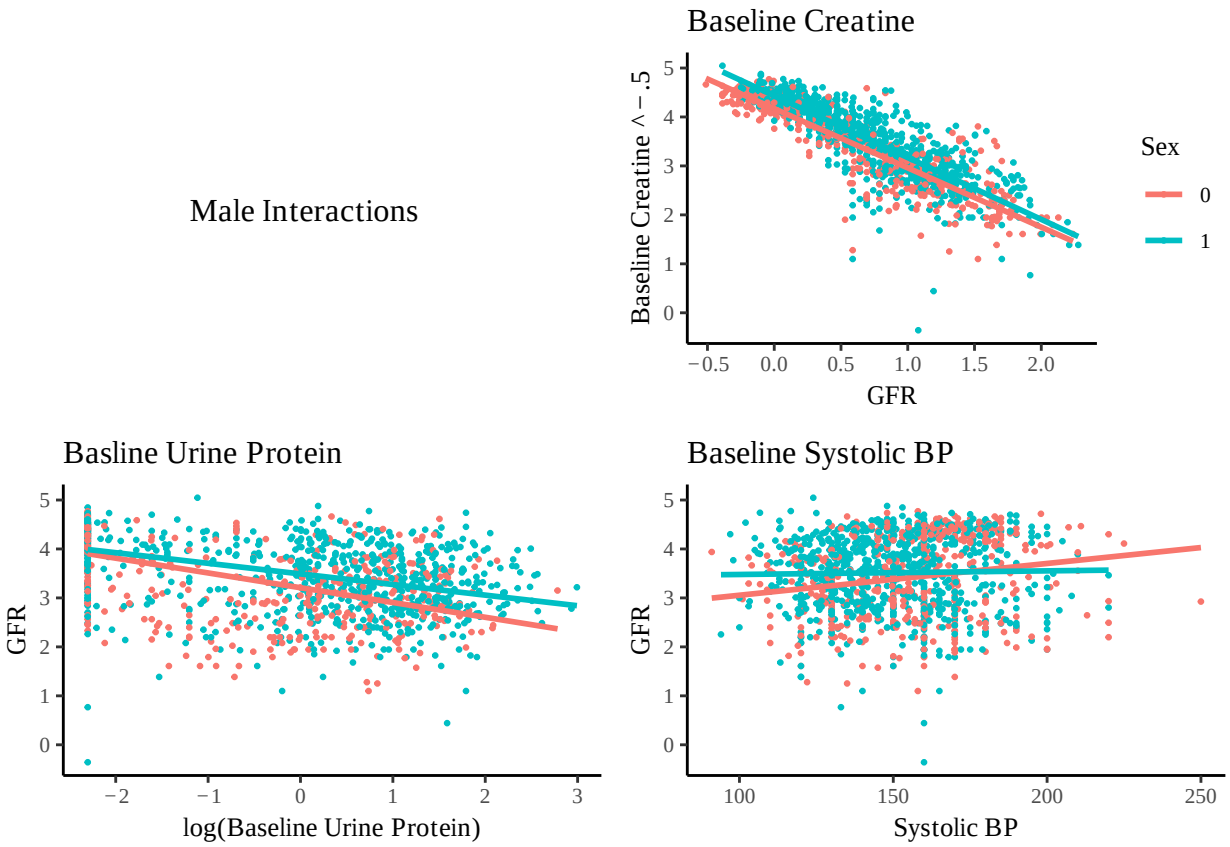
Next, we checked for variable interactions. From intuition, we decided to graphically check for interactions between race and creatine, urine protein, and blood pressure and also interactions between sex and creatine, urine protein, and blood pressure:

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 ()

Black 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:

We see that the interaction terms are not significant.

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

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.

```
set.seed(10)

y = rnorm(n =100, mean = 10, sd = 2)
x = rnorm(n = 100, mean = 3, sd = 1)
```

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

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

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

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)

```
reg1 = lm(y ~ x)

reg1.sum = summary.lm(reg1)

reg1.p = reg1.sum$coefficients[2,4]
```

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

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

```
set.seed(88)

n = 100
n.sims = 1000

xs = matrix(0, nrow = n, ncol = n.sims)
ys = matrix(0, nrow = n, ncol = n.sims)

for (i in 1:n.sims) {

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

}

for (i in 1:n.sims) {

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

}



ps = c()

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?]

```
length(which(ps < .05)) / n.sims
```

```
## [1] 0.048
```

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]

```
set.seed(99)

xs2 = matrix(0, nrow = n, ncol = n.sims)
ys2 = matrix(0, nrow = n, ncol = n.sims)

for (i in 1:n.sims) {

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

}

for (j in 1:n.sims) {

  for (i in 1:n) {

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

  }


}


ps2 = c()
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


}


length(which(ps2 < .05)) / n.sims
```

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

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.

```
set.seed(100)
params = 3
sigma = diag(1, nrow = 3, ncol = 3)


mv.ps = matrix(0, nrow = n.sims, ncol = params)

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.

```
apply(mv.ps, 2, function(x) length(which(x < .05)))
```

```
## [1] 44 55 63
```

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?

```
min.p = c()

for (i in 1:n.sims) {

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

}


length(which(min.p < .05))
```

```
## [1] 151
```

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?

```
set.seed(4)

params = c(3, 5, 10, 20, 50, 90)

results = matrix(0, nrow = length(params), ncol = 2)

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])

  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

  }

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



}
```

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.

```
set.seed(4)

params = 1:99

results2 = matrix(0, nrow = length(params), ncol = 2)

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])

  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

  }

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



}
```