

# PHP 2550 - HW#2

*Blain Morin*

*October 12, 2018*

## Part A

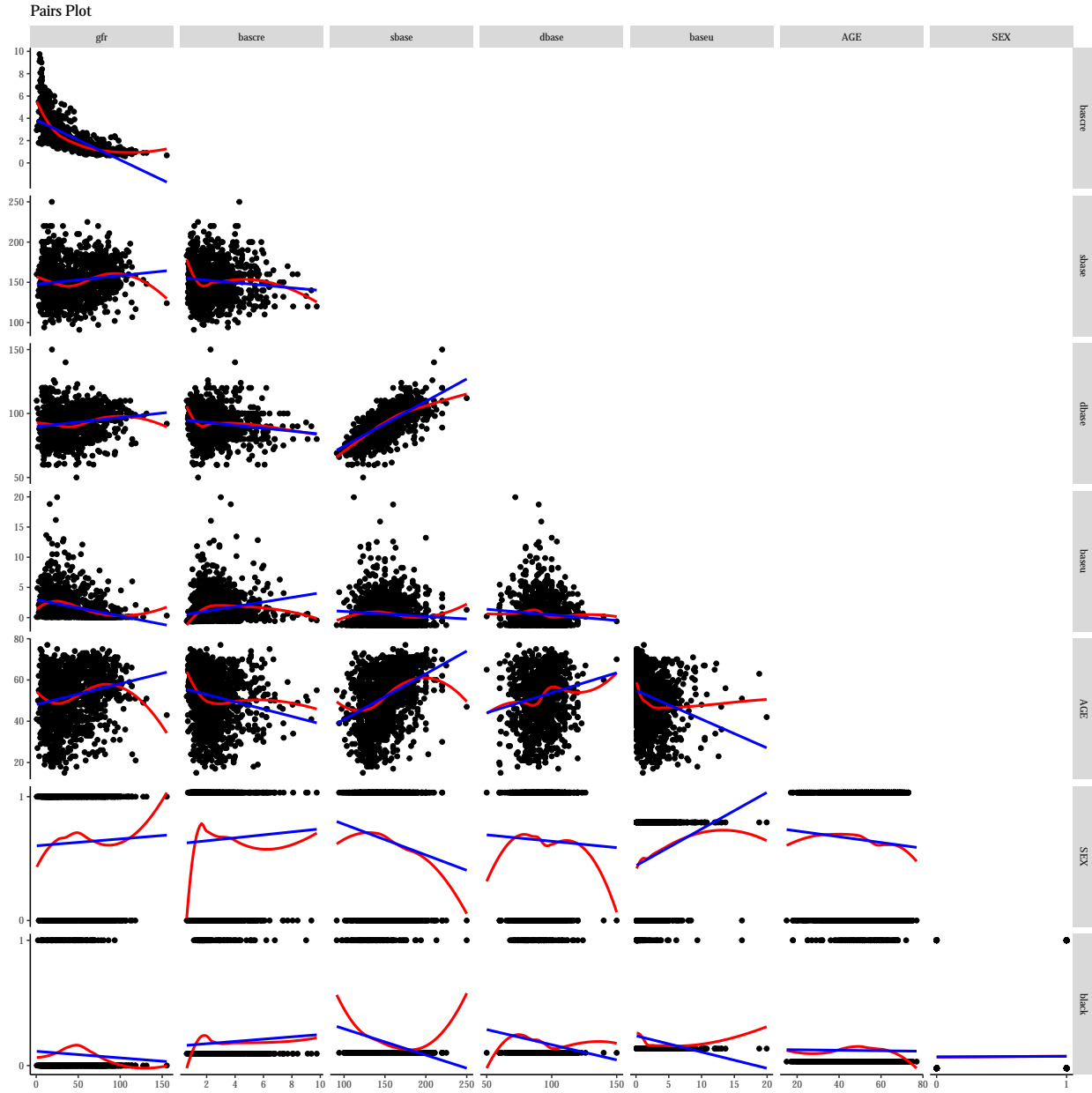
Before model fitting, we first did some exploratory analysis of the data. Here is a summary table of the variables relevant to the assignment (observations with missing values were dropped from analysis):

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 variable. For example, the positive slope in the GFR and sex plot may indicate the males have a higher GFR than women on average.

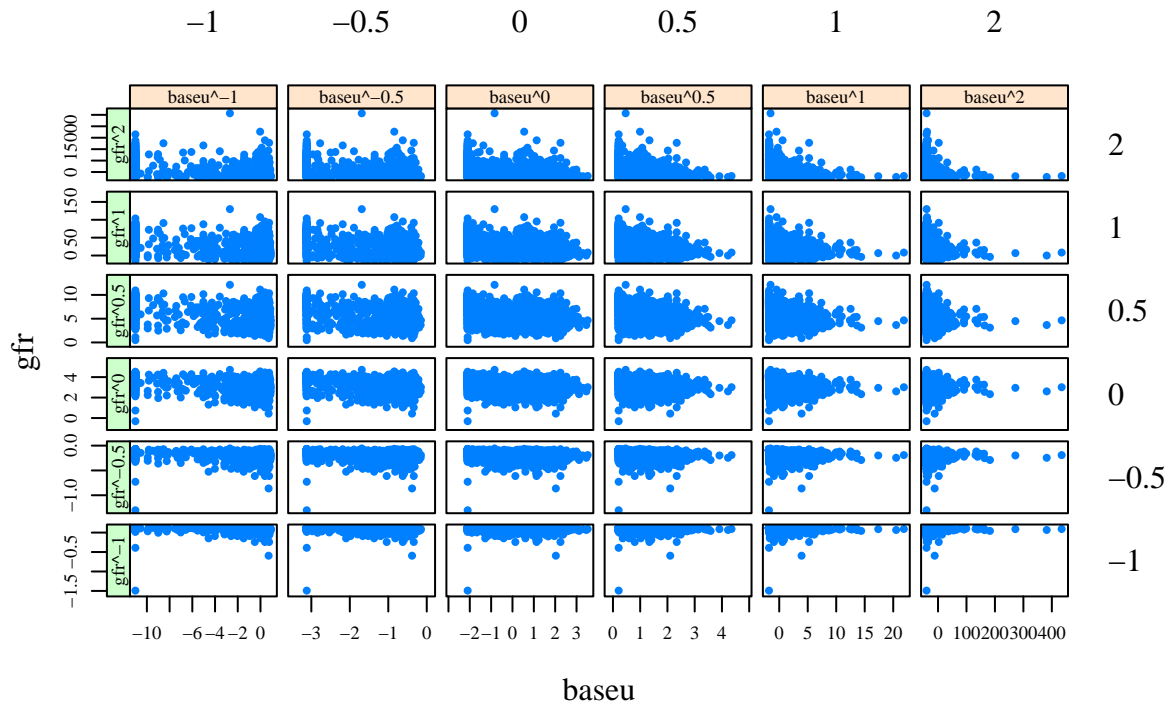
We see from the pairs plot that the relationship between GFR and baseline creatine and the relationship between GFR and baseline urine protien may not be linear. We also see in the baseline urine protien plot that the errors in the linear regression are higher for lower GFR values and they are for higher values. These heterogeneous errors 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 conclusions.

We then explored transformations of GFR, baseline creatine, and baseline urine protein.

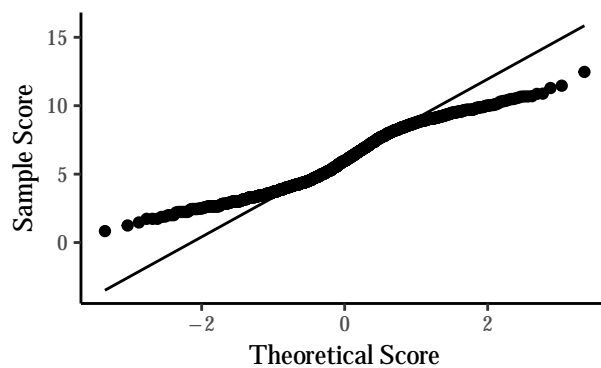
```
ladder(gfr ~ baseu, data = baseline,
       main.in = "Ladder of Powers: GFR and Urine Protein",
       fontfamily = "CM Sans")
```

## Ladder of Powers: GFR and Urine Protein

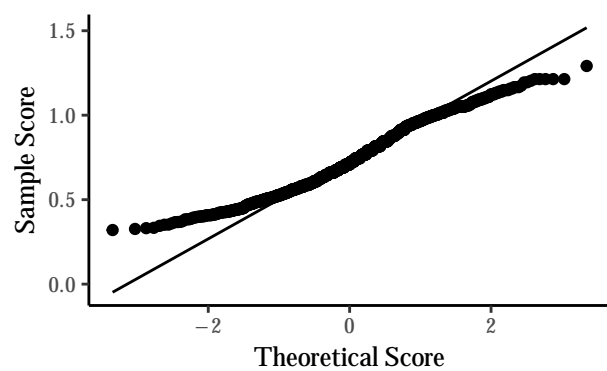


## Transformation Choices

QQ Plot:  $GFR^{.5}$



QQ Plot:  $\text{Base Creatine}^{-.5}$



QQ Plot:  $\log(\text{Base Urine Protein})$

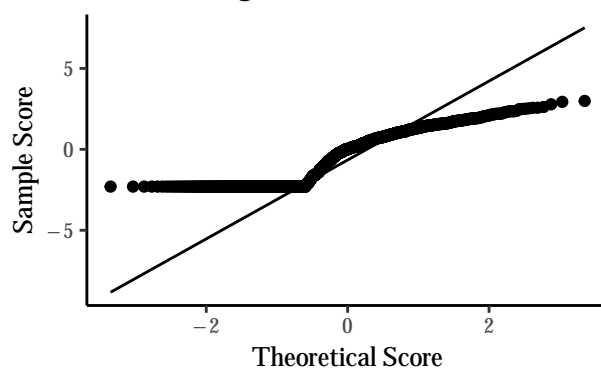
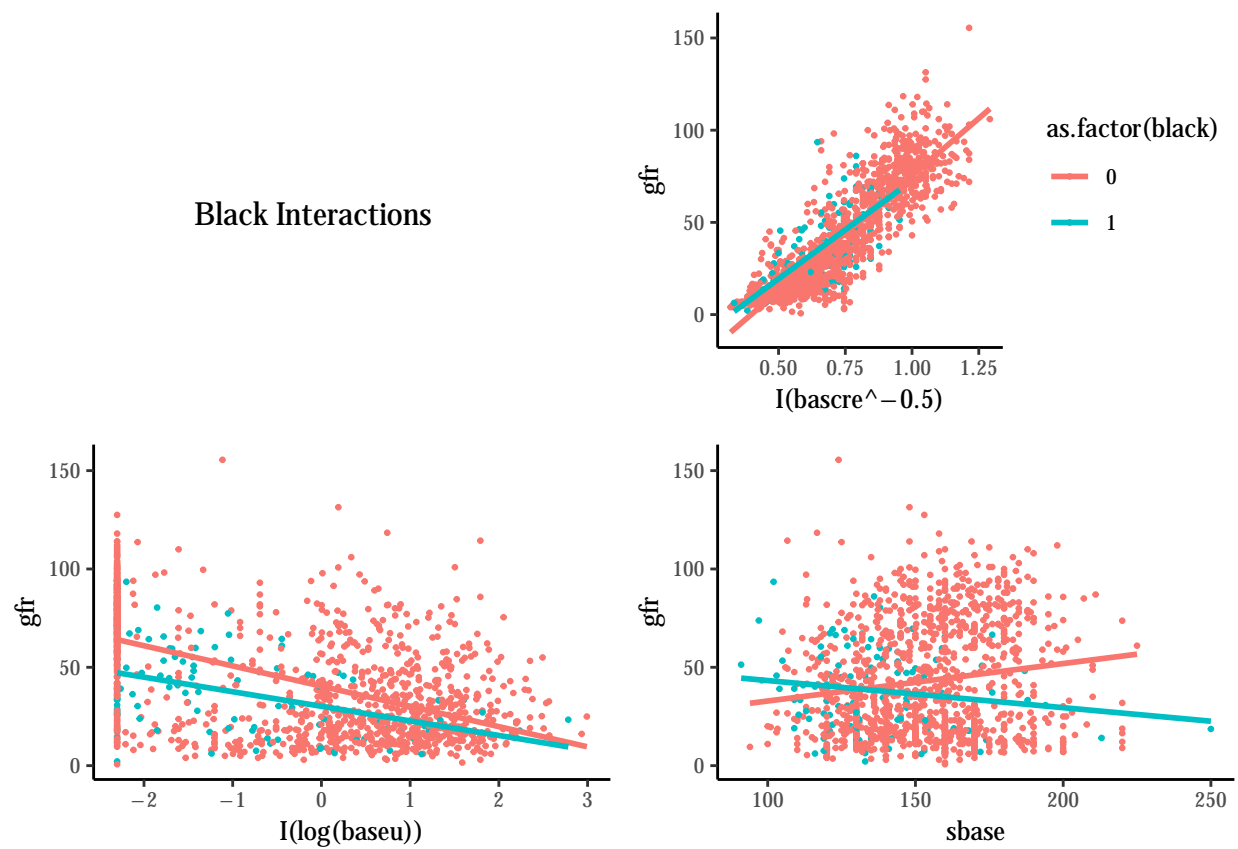


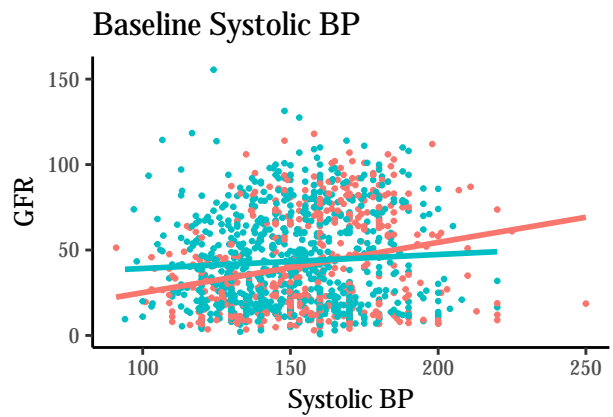
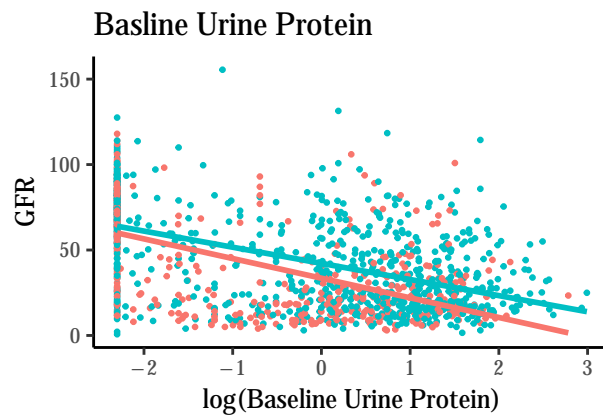
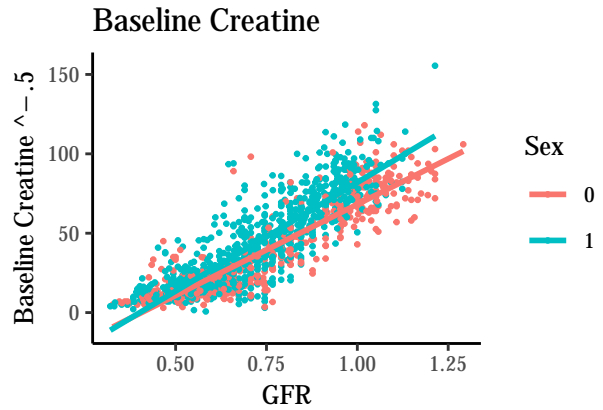
Table 2:

	<i>Dependent variable:</i>		
	gfr	I(gfr <sup>0.5</sup> )	
	(1)	(2)	(3)
bascre	−13.614*** (0.391)		
baseu	−2.391*** (0.246)		
I(bascre <sup>−0.5</sup> )		9.607*** (0.179)	10.123*** (0.159)
log(baseu)		−0.145*** (0.025)	
sbase	0.055** (0.025)	0.0004 (0.001)	0.0003 (0.001)
AGE	0.067 (0.045)	−0.008*** (0.003)	−0.004 (0.003)
as.factor(SEX)1	5.690*** (1.115)	0.756*** (0.062)	0.705*** (0.063)
as.factor(black)1	−3.393* (1.889)	0.272** (0.108)	0.423*** (0.106)
Constant	63.589*** (4.124)	−1.122*** (0.247)	−1.648*** (0.234)
Observations	1,249	1,249	1,249
R <sup>2</sup>	0.570	0.783	0.777
Akaike Inf. Crit.	10,861.770	3,655.775	3,688.445
<i>Note:</i> *p<0.1; **p<0.05; ***p<0.01			

## Black Interactions



## Male Interactions



```
attach(baseline)

lm4 = lm(I(gfr $\wedge$ .5) ~ I(bascre $\wedge$ (-.5)) +
  log(baseu) +
  sbase + AGE +
  as.factor(SEX) + as.factor(black) +
  as.factor(black):sbase +
  as.factor(SEX):sbase)

lm4$AIC = AIC(lm4)

stargazer(lm4, keep.stat = c("aic", "rsq", "n"), header = FALSE)

detach(baseline)
```

Table 3:

	<i>Dependent variable:</i>
	I(gfr <sup>0.5</sup> )
I(bascre <sup>-0.5</sup> )	9.589*** (0.179)
log(baseu)	-0.143*** (0.025)
sbase	0.004* (0.002)
AGE	-0.008*** (0.003)
as.factor(SEX)1	1.487*** (0.407)
as.factor(black)1	0.924 (0.621)
sbase:as.factor(black)1	-0.005 (0.004)
sbase:as.factor(SEX)1	-0.005* (0.003)
Constant	-1.617*** (0.350)
Observations	1,249
R <sup>2</sup>	0.784
Akaike Inf. Crit.	3,655.586
<i>Note:</i>	*p<0.1; **p<0.05; ***p<0.01



## 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)
```

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  $y \sim x[,1] + x[,2] + \dots$  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))

}
```