MLDS 401: Homework 5
Due: Nov 6, 3:00pm
Professor Malthouse

1. Use the following data:

```
dat = data.frame(
  x1=c(2.23,2.57,2.87,3.1,3.39,2.83,3.02,2.14,3.04,3.26,3.39,2.35,
    2.76,3.9,3.15),
  x2=c(9.66,8.94,4.4,6.64,4.91,8.52,8.04,9.05,7.71,5.11,5.05,8.51,
    6.59,4.9,6.96),
  y=c(12.37,12.66,12,11.93,11.06,13.03,13.13,11.44,12.86,10.84,
    11.2,11.56,10.83,12.63,12.46))
```

   (a) Generate a scatterplot matrix and comment. [*Answer: The scatterplot matrix show very little relationship between y and either x variable. There is a substantial correlation between x_1 and x_2.*]

   (b) Regress y on x_1. Is the overall model significant? Examine the residuals and comment. [*Answer: The model is not significant, with $P = .995$. There is no pattern in the residual plot.*]

   (c) Regress y on x_2. Is the overall model significant? Examine the residuals and comment. [*Answer: No, $P = .106$, so we cannot conclude that x_2 has an effect on y. No pattern in the residuals.*]

   (d) Regress y on both x_1 and x_2. Is the overall model significant? Examine the residuals and comment. [*Answer: The overall model is significant with $P = 0.01507$. We conclude at least one predictor is significant. The residuals show an outlier (observation 3) There is also a systematic pattern, suggesting a lack of fit.*]

   (e) Discuss the implications of this example on forward and backward selection. Assume that you will use a significance level for entry into the model of 0.05. [*Answer: The implication is that neither forward nor (forward) stepwise regression would work, because neither would allow any variables in. Backward would work in this case, because both variables would enter and never be dropped.*]

   (f) For you to think about but not turn in: how could you generate more data sets like this?

2. ACT problem 5.2 (Ridge and lasso regression) [*Answer: Since $\mathbf{X} = \mathbf{I}$, $\mathbf{X}\beta = \beta$. Also note that $\lambda \geq 0$.*

(a) *Ridge minimizes* $Q = \sum_i [(y_i - \beta_i)^2 + \lambda \beta_i^2]$. *Note that*

$$\frac{\partial Q}{\partial \beta_i} = -2(y_i - \beta_i) + 2\lambda\beta_i = -2y_i + 2\beta_i(1+\lambda).$$

If we set this equal to 0, solve for β_i *and substitute* $\hat{\beta}_i = y_i$ *we get the result* $\hat{\beta}_i = y_i/(1+\lambda)$.

(b) *For lasso, we can try the same approach: minimize* $Q = \sum_i [(y_i - \beta_i)^2 + \lambda|\beta_i|]$. *Note that*

$$\frac{\partial Q}{\partial \beta_i} = -2(y_i - \beta_i) + \lambda \text{sgn}(\beta_i).$$

- *Assume* $y_i > \lambda/2 > 0$. *For* $\beta_i > 0$,

$$\frac{\partial Q}{\partial \beta_i} = -2(y_i - \beta_i) + \lambda = 0 \qquad \Longrightarrow \quad \beta_i = y_i - \frac{\lambda}{2} > 0.$$

  Note that the case where $\beta_i < 0$ *leads to a contradiction:*

$$\frac{\partial Q}{\partial \beta_i} = -2(y_i - \beta_i) - \lambda = 0 \qquad \Longrightarrow \quad \beta_i = y_i + \frac{\lambda}{2} > 0,$$

  but β_i *was assumed negative.*

- *Assume* $y_i < -\lambda/2 < 0$. *For* $\beta_i < 0$,

$$\frac{\partial Q}{\partial \beta_i} = -2(y_i - \beta_i) - \lambda = 0 \qquad \Longrightarrow \quad \beta_i = y_i + \frac{\lambda}{2} < 0,$$

  since $y_i < -\lambda/2$. *We get a similar contradition for* $\beta_i > 0$.

- *Assume* $|y_i| < \lambda/2$. *It is difficult to work with the derivative. Let us return to the relevant part of the objective function, dropping all terms that do not involve* β_i*:* $\min_{\beta_i}[-2y_i\beta_i + \beta_i^2 + \lambda|\beta_i|]$.
  - *For* $\beta_i \geq 0$, *we get*

$$Q = [-2y_i\beta_i + \beta_i^2 + \lambda\beta_i] = [\beta_i^2 + \underbrace{(\lambda - 2y_i)}_{>0}\beta_i.]$$

    Since $|y_i| < \lambda/2$, $\lambda - 2y_i > 0$ *and* Q *increases with* β_i. *The minimum is at* $\beta_i = 0$.
  - *For* $\beta_i \leq 0$, *we get*

$$Q = [-2y_i\beta_i + \beta_i^2 - \lambda\beta_i] = [\beta_i^2 - \underbrace{(\lambda + 2y_i)}_{>0}\underbrace{\beta_i}_{\leq 0},]$$

    which also increases as β_i *becomes smaller, so it also has a minimum at 0.*

- *See Figure 6.10 in JWHT. Ridge shrinks by a constant multiplicative constant $1/(1+\lambda)$. For $|y_i| \geq \lambda/2$ lasso shrinks by an additive amount $\lambda/2$ toward 0, and for $|y_i| \geq \lambda/2$ lasso sets the slope to exactly 0.*

]

3. This is a variation of JWHT problem 10 on page 263. You will learn how to simulate data where you know the true parameters as a way of evaluating different methods. You will also learn some tricks for generating random data. Assume the following model:

$$y_i = \beta_0 + \sum_{j=1}^{15} x_{ij}\beta_j + e_i,$$

where $\beta_0 = 3$, $\beta_1 = 1$, $\beta_2 = -1$, $\beta_3 = 1.5$, $\beta_4 = 0.5$, $\beta_5 = -0.5$, and $\beta_j = 0$ for $j > 5$. Notice that $x_6, \ldots, x_{15}$ have no effect on y. We will generate a training set from this model of size 100, a very large test set of size 10,000, estimate various models using only the training data, and compare their accuracy on the test set. Note that you know the "truth," i.e., $\beta = (3, 1, -1, 1.5, 0.5, -0.5, 0, \ldots, 0)^\mathsf{T}$).

   (a) We would like for the column vectors of x to be correlated (since if they are uncorrelated the problem is fairly trivial). Let $\mathbf{x} = (x_1, \ldots, x_p)^\mathsf{T}$ (in our case $p = 15$) be a multivariate normal random vector with mean $E(\mathbf{x}) = 0$ and covariance matrix Σ. There are no functions in R (that I know of) to generate $\mathbf{x}$ directly, but it is easy to generate uncorrelated random variables and multiply them by a certain matrix. Let $\mathbf{z} = (z_1, \ldots, z_p)^\mathsf{T}$ be a vector of uncorrelated standard normal random variables, i.e., $E(z_j) = 0$, $V(\mathbf{z}) = \mathbf{I}$, the identity matrix (so that the correlation between any two columns is 0 and the variance of each column is 1). We want to find $p \times p$ matrix $\mathbf{A}$ so that if we let $\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{z}$ then

$$V(\mathbf{x}) = V(\mathbf{A}^\mathsf{T}\mathbf{z}) = \mathbf{A}^\mathsf{T}V(\mathbf{z})\mathbf{A} = \mathbf{A}^\mathsf{T}\mathbf{A} = \Sigma.$$

   We can find this with a *Cholesky* decomposition. For this part, suppose $p = 4$ and we want

$$\Sigma = \begin{pmatrix} 1 & 0.9 & 0.9 & 0.9 \\ 0.9 & 1 & 0.9 & 0.9 \\ 0.9 & 0.9 & 1 & 0.9 \\ 0.9 & 0.9 & 0.9 & 1 \end{pmatrix}$$

   Find the Cholesky decomposition (turn this in) and confirm that $\mathbf{A}^\mathsf{T}\mathbf{A} = \Sigma$. Hint: the `chol` function gives the decomposition, `t` does transposes, and `%*%` does matrix multiplication:

```
sigma = matrix(0.9, nrow=4, ncol=4) + .1*diag(4)
A = chol(sigma)
t(A) %*% A
```

[*Answer: Here is the matrix:*]

```
> A
     [,1]      [,2]      [,3]      [,4]
[1,]    1 0.9000000 0.9000000 0.9000000
[2,]    0 0.4358899 0.2064742 0.2064742
[3,]    0 0.0000000 0.3838859 0.1233919
[4,]    0 0.0000000 0.0000000 0.3635146
```

(b) Generate 1000 random vectors with the matrix Σ from the previous part. Submit the variance matrix of $\mathbf{x}$ and answer whether it approximately equals Σ. Hint:

```
Z = matrix(rnorm(4000), nrow=1000)
X = Z %*% A
```

[*Answer: Here is mine. The off-diagonal elements are roughly 0.9.*]

```
> Z = matrix(rnorm(4000), nrow=1000)
> X = Z %*% A
> var(X)
          [,1]      [,2]      [,3]      [,4]
[1,] 1.0264333 0.9169385 0.9014179 0.9008901
[2,] 0.9169385 1.0034628 0.8953122 0.8941671
[3,] 0.9014179 0.8953122 0.9685689 0.8755908
[4,] 0.9008901 0.8941671 0.8755908 0.9852622
```

(c) Now generate a $10,100 \times 15$ matrix of x variables so that each x_j has variance 1 and $\text{cov}(x_j, x_k) = 0.9$ for $j \neq k$. Generate y values so that $\sigma_e^2 = V(e_i) = 9$. Hint: regenerate the $\mathbf{X}$ matrix using the commands given above so that it is 10100×15 and then compute y as follows (you have to modify the code above to generate $\mathbf{X}$):

```
set.seed(12345)
# generate a new Z, A and X
beta = c(1,-1,1.5,0.5,-0.5,rep(0,10))
e = rnorm(10100)*3
y = 3 + X %*% beta + e
```

[*Answer: Here is my code:*]

```
set.seed(12345)
Z <- matrix(rnorm(10100*15), nrow=10100)
sigma <- matrix(.9, nrow=15, ncol=15) + (1-.9)*diag(15)
A <- chol(sigma)
X <- Z %*% A
dat = data.frame(X)
beta = c(1,-1,1.5,0.5,-0.5,rep(0,10))
e = rnorm(10100)*3
y = 3 + X %*% beta + e
```

(d) Estimate the true model (i.e., include only $x_1, \ldots, x_5$) using OLS. Submit your estimate of σ_e^2, the slopes and R^2. Do the 7 estimates roughly equal the true parameter values (e.g., within two standard errors)? Do the slopes have the correct signs? Are they significant? Do 95% confidence intervals cover the true values? Note: this is the case where you have a strong theory telling you which variables "cause" y. This is the ideal case, but you often do not have such a theory. Hint: make data frames first:

```
dat = data.frame(X)
dat$y <- y
train <- c(rep(T,100), rep(F, 10000))
```

[*Answer: There is a sign flip for X4. Only X1 is significant. The CIs all cover the three parameters. The CIs are very wide because of the multicollinearity. RMSE*=3.178 $\approx$ 3.]

```
> fit = lm(y~X1+X2+X3+X4+X5, data=dat[train,])
> summary(fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.97489    0.23117  12.869  < 2e-16 ***
X1           2.02348    0.65066   3.110  0.00215 **
X2          -0.26888    0.60649  -0.443  0.65801
X3           0.25106    0.70952   0.354  0.72384
X4          -0.05875    0.66340  -0.089  0.92953
X5          -0.21244    0.63987  -0.332  0.74025
Residual standard error: 3.178 on 194 degrees of freedom
Multiple R-squared:  0.2623,Adjusted R-squared:  0.2433
F-statistic:  13.8 on 5 and 194 DF,  p-value: 1.568e-11

> confint(fit)
                2.5 %   97.5 %
```

```
(Intercept)  2.5189494 3.430826
X1           0.7402107 3.306758
X2          -1.4650464 0.927281
X3          -1.1483049 1.650431
X4          -1.3671585 1.249665
X5          -1.4744357 1.049564
>
```

(e) Apply the estimated model in the previous part to the `test` data set (with 10,000 observations) and report the value of MSE (Note that you generated data so that $\sigma_e^2 = 9$). Hint:

```
mean((test$y-predict(fit, test))^2)
```

[*Answer: It is a bit high:* $9.56 > 9$.]

```
> mean((dat$y[!train]-predict(fit, dat[!train,]))^2)
[1] 9.56228
```

(f) Now estimate an OLS model with all 15 predictors. Submit your parameter estimates. Are the coefficients approximately equal to their true values (e.g., within two standard errors)? Are $x_1, \ldots, x_5$ significant? Are their signs right?

[*Answer: The first five variables should be significant, only X1 is. Note also that X10 is significant, but this is a false positive. The signs are right. The CIs are very wide and cover the true values except for X10.*]

```
> fit = lm(y~., data=dat[train,])
> summary(fit)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.97012    0.23355  12.717  < 2e-16 ***
X1           1.99214    0.69812   2.854  0.00482 **
X2          -0.36751    0.67776  -0.542  0.58830
X3           0.57158    0.80521   0.710  0.47869
X4           0.56792    0.77790   0.730  0.46628
X5          -0.06722    0.72300  -0.093  0.92602
X6           0.75323    0.77423   0.973  0.33189
X7          -0.78470    0.68988  -1.137  0.25683
X8           0.43861    0.77289   0.567  0.57107
X9           0.86298    0.68410   1.261  0.20873
X10         -1.66101    0.71354  -2.328  0.02101 *
X11         -0.19689    0.70553  -0.279  0.78051
X12         -0.24271    0.72508  -0.335  0.73821
X13          0.01414    0.65530   0.022  0.98281
```

```
X14          0.04868    0.72769   0.067  0.94673
X15         -0.19374    0.68928  -0.281  0.77897
```

(g) Apply the estimated model in the previous part to the `test` data set (with 10,000 observations) and report the value of MSE.

[*Answer: It even higher than before. Maybe we are overfitting.*]

```
> mean((dat$y[!train]-predict(fit, dat[!train,]))^2)
[1] 9.989624
```

(h) Now estimate a stepwise regression model on all 15 predictors. Report the final model. Did the "right" variables $(x_1, \ldots, x_5)$ come into the model?

[*Answer: Stepwise does not pick the right variables. Of the right ones only X1 comes in, and X9 and X10 should not enter.*]

```
> fit2 = step(fit)
> summary(fit2)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9717     0.2221  13.382  < 2e-16 ***
X1            2.2219     0.5593   3.973 9.98e-05 ***
X9            0.9401     0.5541   1.697   0.0914 .
X10          -1.4193     0.5494  -2.583   0.0105 *
```

(i) Apply the estimated model in the previous part to the `test` data set (with 10,000 observations) and report the value of MSE. How does this value compare with the other two models?

[*Answer: MSE is even worse because stepwise did not pick the right variables. It took the decoys instead!*]

```
> mean((dat$y[!train]-predict(fit2, dat[!train,]))^2)
[1] 10.04558
```

(j) Now estimate a ridge regression models on all 15 predictors and use `cv.glmnet` to pick λ. Generate and submit a ridge trace.

[*Answer: I get an optimal lambda of 0.904203.*]

(k) Apply the estimated ridge model in the previous part to the `test` data set and report the value of MSE.

[*Answer: Ridge does OK, and better then the full or stepwise models.*]

```
> cat("ridge:", mean((dat$y[!train] - yhat)^2), "\n")      # compute test set MSE
ridge: 9.56331
```

(l) Now estimate a lasso regression models on all 15 predictors and use `cv.glmnet` to pick λ. Generate and submit a ridge trace.

[*Answer: My lambda min is 0.06529077*]

(m) Apply the estimated ridge model in the previous part to the `test` data set and report the value of MSE. [*Answer: Lasso is close to ridge, but a little worse. This is surprising since there are 10 variables that should be dropped.*]

```
> cat("lasso:", mean((dat$y[!train] - yhat)^2), "\n")       # compute test set MSE
lasso: 9.693385
```

(n) The file `hw5.R` has some R code that I have written to run all of these models, apply them to the test set, and print their respective values of MSE. You can read it in with the following command

```
> source("hw5.R")
```

This reads in a function called `hw5`, which has three arguments: `beta` = true slope vector, `rho` = correlation between the x vectors and `sigmae` = standard deviation of the errors. For example, if you type `hw5()` you will run the function with the default values, `rho`=0.9 and `sigmae`=3 (so that the error variance is 9). You can enter different values with, e.g., `hw5(rho=0.5, sigmae=2)`. Call this function 9 times under the following conditions and note which model(s) perform best in each case. Briefly summarize your findings, i.e., under what circumstances do you recommend ridge regression? Stepwise? No selection or shrinkage?

Condition	`rho`	`sigmae`
High noise, High multicollinearity	0.9	5
Moderate noise, High multicollinearity	0.9	3
Low noise, High multicollinearity	0.9	1
High noise, Moderate multicollinearity	0.5	5
Moderate noise, Moderate multicollinearity	0.5	3
Low noise, Moderate multicollinearity	0.5	1
High noise, Low multicollinearity	0.1	5
Moderate noise, Low multicollinearity	0.1	3
Low noise, Low multicollinearity	0.1	1

[*Answer: Ridge does best in high-noise situations for all values of ρ. Stepwise (and lasso) do best in low-noise situations. For mid-level noise, ridge is best for moderate or high multicollinearity, but stepwise is better for low multicollinearity.*]

ρ	σ_e	OLS5	OLS15	Step	Ridge	Lasso
.9	5	26.56	27.75	27.26	**25.94**	26.22
.9	3	9.56	9.99	10.05	**9.56**	9.69
.9	1	1.06	1.11	**1.09**	1.20	1.10
.5	5	26.56	27.75	28.51	**26.86**	27.48
.5	3	9.56	9.99	9.94	**9.88**	10.07
.5	1	1.06	1.11	**1.09**	1.16	1.11
.1	5	26.56	27.75	27.56	**27.23**	27.73
.1	3	9.56	9.99	**9.82**	9.93	9.92
.1	1	1.06	1.11	**1.09**	1.13	1.10

(o) For you to think about, but not turn in: how do your conclusions change if you increase the size of your training sample? Values of `beta`?

4. This problem gives you a taste of feature engineering. You have data from a German book company. The company would like to have a machine learning model to predict how much each customer will spend if sent an offer based on previous purchase history. You will build a model based on a previous offer sent on 2014/11/25.

   (a) The file `customer2.csv` has three variables:

   - `id`: customer id, primary key.
   - `train`: indicates if the customer is in the training (1) or test (0) test.
   - `target`: the amount spent in response to the offer sent on 2014/11/25.

   Read the data into software of your choice, compute the natural log of target+1 = `logtarg` and print out basic descriptive statistics.

```
> customer = read.csv("customer2.csv") %>% mutate(logtarg = log(target+1))
> summary(customer)
       id                train             target            logtarg
 Min.   :     957   Min.   :0.0000   Min.   :  0.000   Min.   :0.0000
 1st Qu.: 4448960   1st Qu.:0.0000   1st Qu.:  0.000   1st Qu.:0.0000
 Median : 8090750   Median :0.0000   Median :  0.000   Median :0.0000
 Mean   : 8563488   Mean   :0.3308   Mean   :  3.241   Mean   :0.2529
 3rd Qu.:13378724   3rd Qu.:1.0000   3rd Qu.:  0.000   3rd Qu.:0.0000
 Max.   :16456238   Max.   :1.0000   Max.   :739.480   Max.   :6.6073
```

   (b) The file `orders.csv` is the transaction file, with one record for each unique item purchased. Here is information about the variables:

   - `id`: customer id, matches the customer table.
   - `orddate`: date of order.

- `ordnum`: order number, which uniquely identifies an order
- `category`: gives metadata on the category of an item, e.g., history, cooking, how-to, etc. We will not use this variable for now.
- `qty`: number of the item purchased.
- `price`: unit price. Note that `price*qty` gives the total amount spent on the item.

Read in the file and create a new variable, time (years) since the transaction as `2014/11/25 - orddate`. Compute basic descriptives. Hint: in R/dplyr use

```
> ord=read.csv("orders.csv") %>% mutate(t=as.numeric(as.Date("2014/11/25")
  - as.Date(orddate, "%d%b%Y"))/365.25)
> summary(ord)
       id              orddate              ordnum           category          qty
 Min.   :     957   Length:353687      Min.   :   1018   Min.   : 1.00   Min.   :
 1st Qu.: 3929256   Class :character   1st Qu.: 397351   1st Qu.:14.00   1st Qu.:
 Median : 6353495   Mode  :character   Median : 728198   Median :20.00   Median :
 Mean   : 6791632                      Mean   : 692588   Mean   :32.55   Mean   :
 3rd Qu.: 8720240                      3rd Qu.:1004519   3rd Qu.:36.00   3rd Qu.:
 Max.   :16456238                      Max.   :1256189   Max.   :99.00   Max.   :3502
     price                t
 Min.   :   0.000   Min.   :0.002738
 1st Qu.:   5.113   1st Qu.:1.229295
 Median :   8.666   Median :2.729637
 Mean   :  11.495   Mean   :2.958282
 3rd Qu.:  12.782   3rd Qu.:4.528405
 Max.   :5010.660   Max.   :7.058179
```

(c) Roll up/aggregate the transaction file so that you have one record per customer and the following variables. I will call this the "RFM" table. Submit basic summary statistics.

- `id`: customer id.
- `tof`: time on file—years since the first order, hint: maximum value of `t`.
- `r`: recency—years since the most recent order.
- `f`: frequency of orders. Hint: in dplyr see the `n_distinct` function.
- `m`: monetary—total amount spent in the past.

```
> fm = ord %>%
+   group_by(id) %>%
+   summarise(tof=max(t), r = min(t), f=n_distinct(ordnum), m= round(sum(price*qty),2
       id                tof                 r                 f                  m
```

```
 Min.   :     957   Min.   :0.002738   Min.   :0.002738   Min.   :  1.000   Min.   :
 1st Qu.: 4448960   1st Qu.:1.338809   1st Qu.:0.303901   1st Qu.:  2.000   1st Qu.:
 Median : 8090750   Median :3.800137   Median :0.851472   Median :  4.000   Median :
 Mean   : 8563488   Mean   :3.681231   Mean   :1.439085   Mean   :  6.111   Mean   :
 3rd Qu.:13378724   3rd Qu.:6.036961   3rd Qu.:2.031485   3rd Qu.:  8.000   3rd Qu.:
 Max.   :16456238   Max.   :7.058179   Max.   :7.058179   Max.   :160.000   Max.   :4
```

(d) Join the customer and RFM tables. Regress `logtarg` on `log(tof)`, `log(r)`, `log(f)`, and `log(m+1)` using only the training data. Show a summary of the fitted model.

```
> all = left_join(customer, fm, by="id")
> train = (all$train==1)  # create logical train variable
> fit = lm(logtarg ~ log(tof) + log(r)+log(f) + log(m+1), all[train, ])
> summary(fit)
Call: lm(logtarg ~ log(tof) + log(r) + log(f) + log(m+1), all[train,])

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.07231    0.06126  -1.180  0.23791
log(tof)    -0.06587    0.02064  -3.192  0.00142 **
log(r)      -0.07658    0.01297  -5.904 3.76e-09 ***
log(f)       0.11925    0.02658   4.486 7.41e-06 ***
log(m + 1)   0.04550    0.01741   2.614  0.00898 **

Residual standard error: 0.9146 on 5546 degrees of freedom
Multiple R-squared:  0.05328,Adjusted R-squared:  0.0526
F-statistic: 78.03 on 4 and 5546 DF,  p-value: < 2.2e-16
```

(e) Apply the model from the previous part to the test set and compute the mean squared error on the test set.

```
> yhat = predict(fit, all[!train,])
> mean((all$logtarg[!train]-yhat)^2)
[1] 0.8087568
```

5. Submit your final bike model.

(a) Give a rational for why you included each variable, i.e., why would the variable cause bike demand? Document what your model, e.g., what variables did you combine to form composites, etc. Give a correlation matrix and VIFs for the variables in the final model.

(b) Apply ridge regression to your final model (generate a ridge trace) to evaluate how robust your conclusions are. For example, if some of your variables flip signs in the ridge trace, then your conclusions about the variable are not stable. Also show a lasso trace.