STAT448 - Assignment 2

Zhen Huang 74093323

5 May

```
library(tidyverse)
set.seed(3323)
```

(a) Set a seed at the beginning of your code equal to the last 4 numbers of your student id (or one of your student id's if you work in pairs)

```
load("~/Documents/OneDrive/UC/Semester2/Stat448_BigData/big_data/Ass-2-448/Residen.RData")
attach(Residen)
```

(b) Split the data set into a training set and a test set Splite the data set with 8:2 (train:test)

```
dim(Residen)
```

```
## [1] 372 109
```

```
sum(is.na(Residen))
```

```
## [1] 0
```

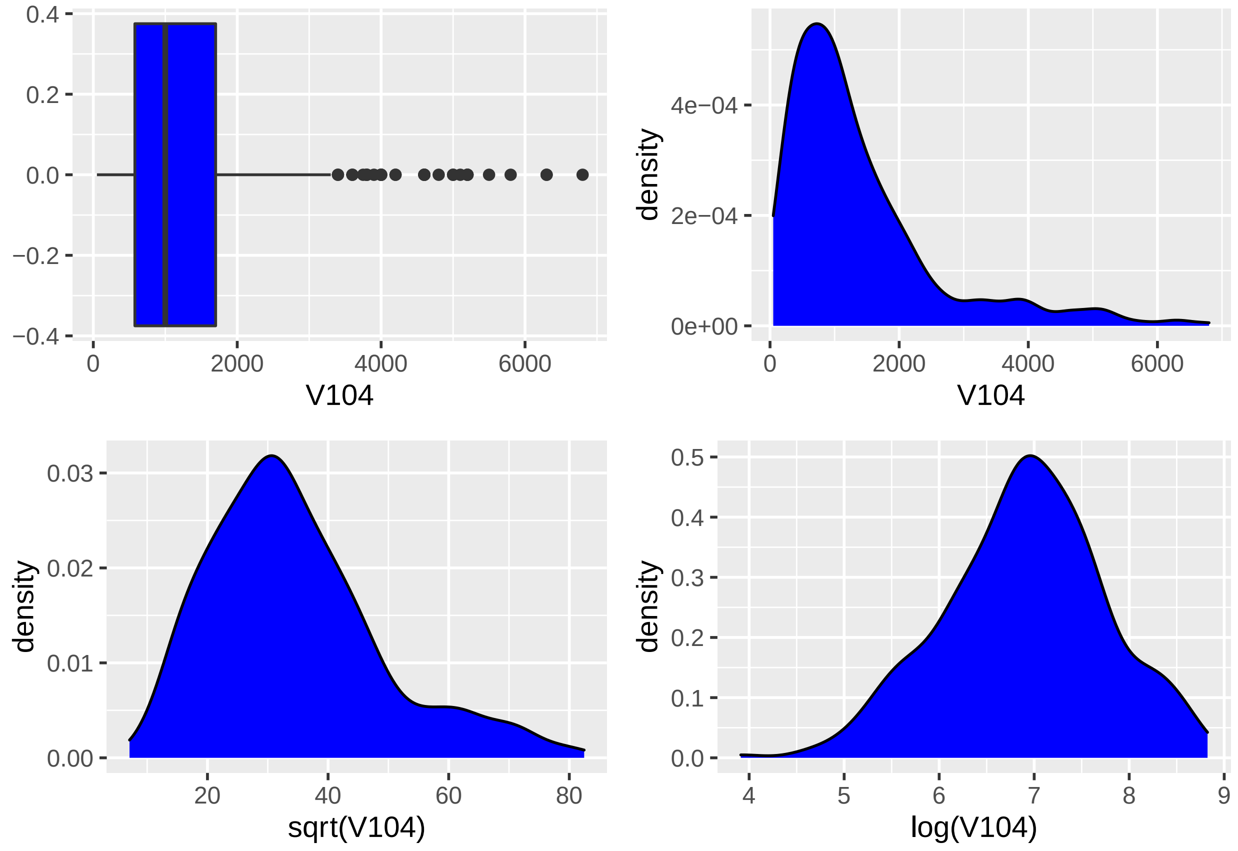
```
my_split <- sample(1:nrow(Residen), 0.8*nrow(Residen), replace=FALSE)
#res_train <- Residen[my_split,-c(1:4)]
#res_test <- Residen[-my_split,-c(1:4)]
res_train <- Residen[my_split,]
res_test <- Residen[-my_split,]
dim(Residen)[1] == dim(res_train)[1]+dim(res_test)[1] # check if split correct
```

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

(c) Fit a linear regression model on the training set to explain the "actual sales price" (V104) in terms of the of the other variables excluding the variable "actual construction costs" (V105). Report the test RMSE obtained. Detect potential outliers and skewness of V104 (response) and V105 (predictor)

```r
library(ggplot2)
if(!require(gridExtra)) install.packages("gridExtra")
```

```r
# V104 distribution
plot1=ggplot(Residen, aes(V104)) + geom_boxplot(fill="blue")
plot2=ggplot(Residen, aes(V104)) + geom_density(fill="blue")
plot3=ggplot(Residen, aes(sqrt(V104))) + geom_density(fill="blue")
plot4=ggplot(Residen, aes(log(V104))) + geom_density(fill="blue")
grid.arrange(plot1,plot2,plot3,plot4,nrow=2,ncol=2)
```



```r
# calculate skewness of V104
round(e1071::skewness(V104),2)
```

```
## [1] 1.86
```

```r
round(e1071::skewness(log(V104)),2)
```

```
## [1] -0.22
```

```r
round(e1071::skewness(sqrt(V104)),2)
```

```
## [1] 0.89
```

Fit a linear model by using original V104

```r
lm_mod <- lm(V104 ~ .-V105, data=res_train) # build linear regression model on training data set
lm_pred <- predict(lm_mod, newdata = res_test)
```

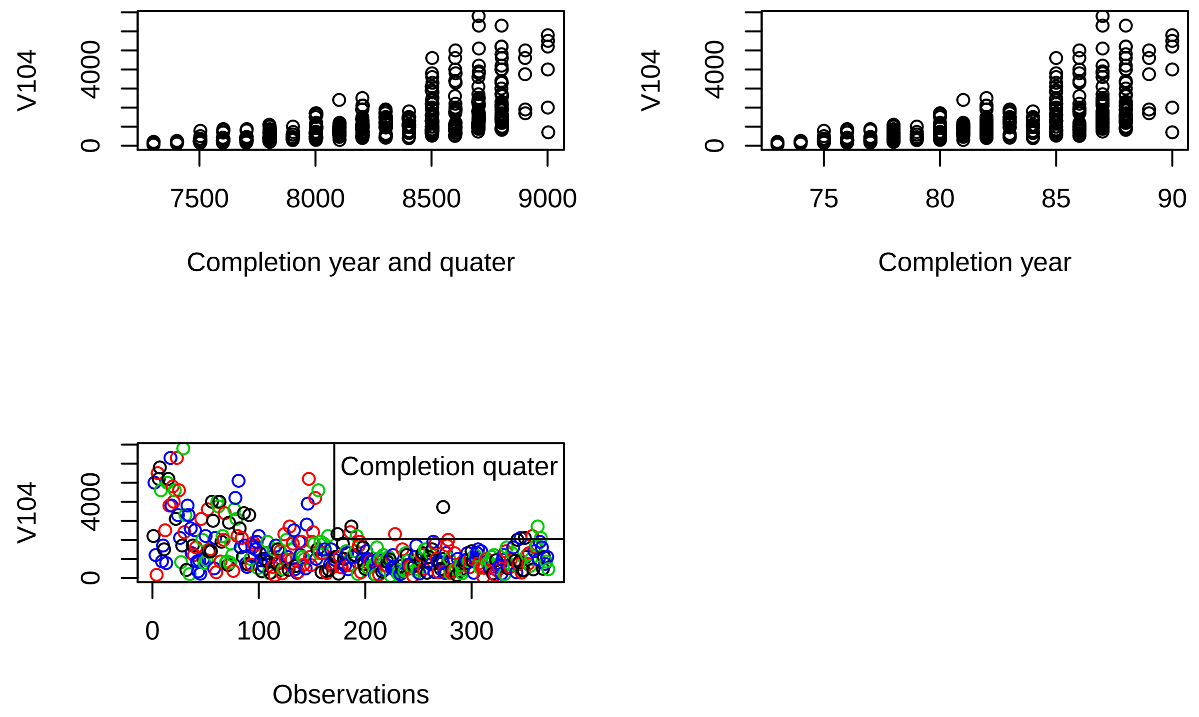
```
## Warning in predict.lm(lm_mod, newdata = res_test): prediction from a rank-
## deficient fit may be misleading
```

```r
lm_rmse <- sqrt(sum(lm_pred - res_test$V104)^2/length(res_test$V104))
c(RMSE = lm_rmse, R2=summary(lm_mod)$r.squared)
```

```
##        RMSE          R2
## 791.9099531   0.9909736
```

The *completion year* have a potential realtionship with *V104 (actual sales price)*, so it was kept for the following steps.

```r
# new variable of completion quarter and year
comp_quat <- paste0(Residen$`COMPLETION YEAR`,paste0(0,Residen$`COMPLETION QUARTER`))
par(mfrow=c(2,2))
plot(comp_quat, Residen$V104, xlab = "Completion year and quater", ylab = "V104")
plot(Residen$`COMPLETION YEAR`, Residen$V104, xlab = "Completion year", ylab = "V104")
plot(Residen$V104, col = Residen$`COMPLETION QUARTER`, xlab = "Observations", ylab = "V104")
legend("topright", legend = "", col = Residen$`COMPLETION QUARTER`, pch = 1, title = "Completion quater
```



Fit a linear model by using the log transformed (normalised distribution) V104

```
lm_sqrt_mod <- lm(log(V104) ~ .-V105, data=res_train) # build linear regression model on training data
lm_pred <- predict(lm_sqrt_mod, newdata = res_test)
```

```
## Warning in predict.lm(lm_sqrt_mod, newdata = res_test): prediction from a rank-
## deficient fit may be misleading
```

```
lm_sqrt_rmse <- sqrt(sum(exp(lm_pred) - res_test$V104)^2/length(res_test$V104))
c(RMSE = lm_sqrt_rmse, R2=summary(lm_sqrt_mod)$r.squared)
```

```
##        RMSE          R2
## 376.2512602   0.9505534
```

(d) Fit a linear regression model using stepwise selection on the training set. Report the test RMSE obtained. One way to implement stepwise selection by using `leaps::regsubsets`.

1. Forward selection

```
library(leaps)
# Forward and Backward Stepwise Selection
regfit.fwd <- regsubsets(V104~ .-V105, data=res_train, nvmax=107, method="forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 35 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to replace is
## not a multiple of replacement length
```

```
reg.summary <- summary(regfit.fwd)
vars_fwd <- c(adjr2=which.max(reg.summary$adjr2), cp=which.min(reg.summary$cp) ,bic=which.min(reg.summa
# define a empty vector, length is 3, for 3 metrices
rmse_fwd <- rep(NA,3)
# names of 3 metrices
names(rmse_fwd) <- names(vars_fwd)
# model matrix (design matrix) transform
test.mat <- model.matrix(V104~.-V105, data=res_test)

for(i in 1:3){
  # extract the optimal number of variables
  coefi=coef(regfit.fwd,id=vars_fwd[i])
  # row %*% coefficients (matrix multiply) give the predict values
  pred=test.mat[,names(coefi)]%*%coefi
  # calculate rmse and put into vector
  rmse_fwd[i] = sqrt(sum(pred - res_test$V104)^2/length(res_test$V104))
}
```

2. Backward selection

```r
regfit.bwd=regsubsets(V104~ .-V105, data=res_train, nvmax=107, method="backward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 35 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to replace is
## not a multiple of replacement length
```

```r
reg.summary=summary(regfit.bwd)
which.max(reg.summary$adjr2)
```

```
## [1] 44
```

```r
which.min(reg.summary$cp)
```

```
## [1] 23
```

```r
which.min(reg.summary$bic)
```

```
## [1] 15
```

```r
vars_bwd <- c(adjr2=which.max(reg.summary$adjr2), cp=which.min(reg.summary$cp) ,bic=which.min(reg.summa
# define a empty vector, length is 3, for 3 metrices
rmse_bwd <- rep(NA,3)
names(rmse_bwd) <- names(vars_bwd)

for(i in 1:3){
  # extract the optimal number of variables
  coefi=coef(regfit.bwd,id=vars_bwd[i])
  # row %*% coefficients (matrix multiply) give the predict values
  pred=test.mat[,names(coefi)]%*%coefi
  # calculate rmse and put into vector
  rmse_bwd[i] = sqrt(sum(pred - res_test$V104)^2/length(res_test$V104))
}
```

Reports of forward selection methods.

```r
c(RMSE = min(rmse_fwd), Measure=names(which.min(rmse_fwd)), Variables=vars_fwd[which.min(rmse_fwd)])
```

```
##                RMSE            Measure       Variables.cp
## "57.6291720851883"               "cp"               "18"
```

Reports of backward selection methods.

```r
c(RMSE = min(rmse_bwd), Measure=names(which.min(rmse_bwd)), Variables=vars_fwd[which.min(rmse_bwd)])
```

```
##                RMSE            Measure      Variables.bic
## "115.457607060865"              "bic"               "11"
```

Another way to do stepwise selection by using AIC measures with `MASS::stepAIC`.

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
model0 = lm(V104~1, data=res_train) # define a null model as initial step of forward/hybrid selection
stepAIC_fwd_mod <- MASS::stepAIC(model0, direction="forward",scope=list(upper=lm_mod,lower=model0), trac
stepAIC_fwd_pred <- predict(stepAIC_fwd_mod, newdata = res_test)
stepAIC_fwd_rmse <- sqrt(sum(stepAIC_fwd_pred - res_test$V104)^2/length(res_test$V104))

# Using linear regression model to be the initial step
stepAIC_bwd_mod <- MASS::stepAIC(lm_mod, direction="backward", trace = FALSE)
stepAIC_bwd_pred <- predict(stepAIC_bwd_mod, newdata = res_test)
stepAIC_bwd_rmse <- sqrt(sum(stepAIC_bwd_pred - res_test$V104)^2/length(res_test$V104))

# Hybrid selection
stepAIC_mod <- MASS::stepAIC(model0, direction="both",scope=list(upper=lm_mod,lower=model0), trace = FA
stepAIC_pred <- predict(stepAIC_mod, newdata = res_test)
stepAIC_rmse <- sqrt(sum(stepAIC_pred - res_test$V104)^2/length(res_test$V104))
```

Report of three stepwise methods using AIC measures

```r
c(RMSE = stepAIC_fwd_rmse, Variables=length(coefficients(stepAIC_fwd_mod)))
```

```
##      RMSE Variables
##  34.55121  19.00000
```

```r
c(RMSE = stepAIC_bwd_rmse, Variables=length(coefficients(stepAIC_bwd_mod)))
```

```
##      RMSE Variables
##  238.3094   45.0000
```

```r
c(RMSE = stepAIC_rmse, Variables=length(coefficients(stepAIC_mod)))
```

```
##      RMSE Variables
##  37.88764  15.00000
```

```r
library(glmnet)
```

```r
# Ridge Regression
x_train = model.matrix(V104 ~ .-V105, res_train)[,-1]
x_test = model.matrix(V104 ~ .-V105, res_test)[,-1]
y_train = res_train$V104
y_test = res_test$V104

# Error of Null model
sqrt(mean((mean(y_train) - y_test)^2))
```

(e) Fit a linear regression model using ridge regression on the training set, with λ chosen by cross validation. Report the test RMSE obtained.

```
## [1] 1298.092
```

```r
# generate a tuning grid, (10^10~0.01)
grid=10^seq(10,-2,length=100)
# alpha=0 ridge
ridge_mod = glmnet(x_train, y_train, alpha=0, lambda = grid, thresh = 1e-12)
```

```
## Warning: from glmnet Fortran code (error code -83); Convergence for 83th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```r
ridge_cv_out = cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on training data
bestlam = ridge_cv_out$lambda.min  # Select lamda that minimizes training MSE
# optimal lambda from cross validation
bestlam
```

```
## [1] 115.6424
```

```r
ridge_pred = predict(ridge_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data

c(RMSE = sqrt(mean((ridge_pred - y_test)^2))/length(y_test)) # Calculate test MSE
```

```
##     RMSE
## 3.029313
```

```r
ridge_coeff=predict(ridge_mod,type="coefficients",s=bestlam) #Display coefficients using lambda chosen
length(ridge_coeff[ridge_coeff!=0]) #Display only non-zero coefficients
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] 108
```

```r
lasso_mod = glmnet(x_train, y_train, alpha=1, lambda = grid)
lasso_cv_out = cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso regression model on training data
bestlam = lasso_cv_out$lambda.min  # Select lamda that minimizes training MSE
# optimal lambda from cross validation
bestlam
```

(f) Fit a linear regression model using lasso on the training set, with λ chosen by cross validation. Report the test RMSE obtained along with the number of non-zero coefficient estimates.

```
## [1] 2.068438
```

```r
lasso_pred <-  predict(lasso_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data

#lasso_pred <- exp(lasso_pred)


lasso_coeff=predict(lasso_mod,type="coefficients",s=bestlam) #Display coefficients using lambda chosen
c(RMSE = sqrt(mean((lasso_pred - y_test)^2))/length(y_test), "Number of non-zero coefficient estimates"
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
##                                     RMSE
##                                  3.09664
## Number of non-zero coefficient estimates
##                                 26.00000
```

```r
 #Display only non-zero coefficients
```

(g) Comment on the results obtained. How accurately can we predict the actual sale price? Is there much difference among the test errors resulting from these four approaches? The root mean squared error (RMSE) were calculated for the average (mean) of squared differences between prediction and actual sale price. The result of four mothods (10 models) are showed below table:

Model	RMSE	Variables selection
Null model	1401.975	-
lm	791.9099	-
lm (with log)	376.2513	-
forward selection (cp)	57.6292	18
forward selection (AIC)	34.5512	19
backward selection (bic)	115.4576	11
backward selection (AIC)	238.3094	45
Hybrid selection (AIC)	37.8876	15
Ridge	3.0293	108(with intercept)
Lasso	3.0952	26 (with intercept)

The RMSE in above table were calculated from the same test data set, so it is a relatively fair way to compare the accuracy between the models. Another way in answering the question of how accurately can we predict is using prediction interval if certain values of predictors are given.

The performance of nine models are better than the Null model. The multivariate linear regression had a 0.9909 R^2 , while its RMSE is 791.9099. The prediction error is relatively high compared with the mean of the actual sale price in test data set (1394.6). The overfitting and unstable of estimation may happened here. When fitting the multivariate linear model, a warning message "prediction from a rank-deficient fit may be misleading" suggested there were multicollinearity of predictors. We can also use `cor(Residen)` to observe the condition.

Stepwise selection improved the accurate of prediction, the RMSE decreased to 37.8876, meanwhile, it offered a parsimonious model with 15 predictors which are more important than the others. Stepwise selection was implemented by using `stepAIC`, but the forward and backward selection were also be tried to estimate the data. Different metrices (C_p, AIC, BIC) can be used in finding the optimal subset. The initial model of backward selection was from the multi-linear model, which might be the reason of their less well performance. Because the results of multi-linear model (using `lm`) only fitted the mode with 72 predictors and some important information lost.

Ridge and lasso model performaned better than multi-linear and stepwise models. Lasso offered a parsimonious model with 26 predictors, while the ridge model included all the 108 predictors. When the data set has a large number of predictors, the variance of OLS models increase, while ridge models perform a bias-variance trade-off with a L2 shrinkage penalty. Lasso models do similar with ridge but with L1 shrinkage penalty, which help to fit sparse models with less variables and more easier to be interpreted.