project\_rmd

library(MASS) # Load library  
  
### Load data  
df <- MPV::table.b1  
head(df)  
col\_names <- c('Games won (per 14-game season)',  
 'Rushing yards (season)',  
 'Passing yards (season)',  
 'Punting average (yards/punt)',  
 'Field Goal Percentage (FGs made/FGs attempted)',  
 'Turnover differential (turnovers acquired - turnovers lost)',  
 'Penalty yards (season)',  
 'Percent rushing (rushing plays/total plays)',  
 'Opponents\' rushing yards (season)',  
 'Opponents\' passing yards (season)')  
  
team\_names <- c("Washington", "Minnesota", "New England", "Oakland", "Pittsburgh", "Baltimore",  
 "Los Angeles", "Dallas", "Atlanta", "Buffalo", "Chicago", "Cincinnati", "Cleveland",  
 "Denver", "Detroit", "Green Bay", "Houston", "Kansas City", "Miami", "New Orleans",  
 "New York Giants", "New York Jets", "Philadelphia", "St. Louis", "San Diego",  
 "San Francisco", "Seattle", "Tampa Bay")  
  
### Graphical data displays  
x\_labels <- c('Games won (per 14-game season)',  
 'Rushing yards (season)',  
 'Passing yards (season)',  
 'Punting average (yards/punt)',  
 'Field Goal Percentage (FGs made/FGs attempted)',  
 'Turnover differential (turnovers acquired - turnovers lost)',  
 'Penalty yards (season)',  
 'Percent rushing (rushing plays/total plays)',  
 'Opponents\' rushing yards (season)',  
 'Opponents\' passing yards (season)')  
var\_names <- c('y', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9')  
par(mfrow = c(3,4))  
for (i in 1:ncol(df)) {  
 hist(df[,i], probability = TRUE,  
 main = var\_names[i],  
 xlab = x\_labels[i])  
 lines(density(df[,i]))  
}  
dev.off()  
  
pairs(df) # pairs matrix  
  
# Reference: https://stackoverflow.com/questions/9439619/replace-all-values-in-a-matrix-0-1-with-0/9439694  
# Reference: https://stackoverflow.com/questions/3192791/find-indices-of-non-zero-elements-in-matrix/3193207  
### correlation matrix  
cor\_df <- abs(cor(df))  
# diag(cor\_df) <- 0  
cor\_df[lower.tri(cor\_df)] <- 0  
cor\_df <- as.matrix(cor\_df)  
cor\_df <- ifelse(cor\_df < 0.5, 0, cor\_df)  
which(cor\_df != 0, arr.ind = TRUE)  
# correlation > 0.5  
write.table(round(cor\_df, 2), file = 'correlation.txt', sep = ',')  
  
# Reference: https://stackoverflow.com/questions/39731068/how-to-let-a-matrix-minus-vector-by-row-rather-than-by-column  
# Reference: https://stackoverflow.com/questions/3444889/how-to-use-the-sweep-function  
# Reference: http://www.sthda.com/english/articles/39-regression-model-diagnostics/160-multicollinearity-essentials-and-vif-in-r/  
### VIF pp.117-118, pp.296-297 in Textbook  
ones <- rep(1, nrow(df))  
X <- as.matrix(cbind(ones, df[,2:ncol(df)]))  
  
sample\_means <- colMeans(X)  
sample\_sd <- sqrt(diag(cov(X)))  
X\_centered <- sweep(X, 2, sample\_means)  
X\_standardized <- sweep(X\_centered, 2, sample\_sd, FUN = "/")  
W <- X\_standardized[,2:ncol(X\_standardized)]  
  
C <- solve(t(W) %\*% W)  
# 1 or more large VIFs indicate multicollinearity  
diag(C) # All VIF's are fairly controlled  
car::vif(model\_1) # x1, x7, and x8 are relatively large (around 5)  
  
### Modeling  
# often the variables aren't entirely normal  
# there are limited observations  
# the response seems to have two peaks  
model\_0 <- lm(y~1, df)  
model\_1 <- lm(y~., df)  
  
# Significance of regression  
# F: 8.8458  
# p-value: 5.303e-05  
anova(model\_0, model\_1)  
  
# Check contribution of each term  
# most aren't significant (only x2 > 0.05)  
summary(model\_1)  
round(summary(model\_1)$coefficients[,4], 4)  
  
# BoxCox  
boxcox(df[-28,]$y~., data=df[-28,]) # one response is 0 (28th)  
  
# natural log is not so good  
boxcox(log(df[-28,]$y)~., data=df[-28,])  
  
# 3/4 also works  
boxcox(df[-28,]$y^(3/4)~., data=df[-28,])  
model\_2 <- lm(y^(3/4)~., df)  
model\_2b <- lm(y^(3/4)~1, df)  
anova(model\_2, model\_2b)  
summary(model\_2) # same issue  
round(summary(model\_2)$coefficients[,4], 4)  
  
# forward selection  
forward1 <- MASS::stepAIC(model\_0,  
 scope = list(upper=model\_1, lower=model\_0),  
 direction = c('forward'))  
summary(forward1) # lm(formula = y ~ x8 + x2 + x7 + x9, data = df)  
  
# backward elimination  
backward1 <- MASS::stepAIC(model\_1, direction = c('backward'))  
summary(backward1) # lm(formula = y ~ x2 + x7 + x8 + x9, data = df)  
  
# stepwise regression  
step1 <- MASS::stepAIC(model\_0,  
 scope = list(upper=model\_1, lower=model\_0),  
 direction = c('both'))  
summary(step1) # lm(formula = y ~ x8 + x2 + x7 + x9, data = df)  
  
# all possible regressions  
best1 <- leaps::regsubsets(x = y~., data = df, nvmax = 10)  
best1\_sum <- summary(best1)  
p.m <- 2:10  
n <- nrow(df)  
MS\_Res <- best1\_sum$rss / (n-p.m)  
aic <- n \* log(best1\_sum$rss / n) + 2 \* p.m  
data.frame(  
 adjrsq = which.max(best1\_sum$adjr2),  
 rsq = which.max(best1\_sum$rsq),  
 CP = which.min(best1\_sum$cp),  
 MSRes = which.min(MS\_Res),  
 bic = which.min(best1\_sum$bic),  
 aic = which.min(aic)  
)  
  
# voted 3: CP, bic  
# voted 4: adjrsq, MSRes, aic  
# 3: x2, x7, x8  
best\_1a <- lm(y~x2 + x7 + x8, data = df)  
summary(best\_1a)  
round(summary(best\_1a)$coefficients[,4],4)  
anova(model\_0, best\_1a)  
  
# 4: x2, x7, x8, x9  
best\_1b <- lm(y~x2 + x7 + x8 + x9, data = df)  
summary(best\_1b)  
round(summary(best\_1b)$coefficients[,4],4)  
anova(model\_0, best\_1b)  
  
### Residual analysis  
beta\_hat\_calc <- function(X, y) {  
 X <- as.matrix(X)  
 beta\_hat <- solve(t(X) %\*% X) %\*% t(X) %\*% y  
 return(beta\_hat)  
}  
SS\_Res\_calc <- function(y, beta\_hat, X) {  
 SS\_Res <- (t(y) %\*% y) - (t(beta\_hat) %\*% t(X) %\*% y)  
 return(SS\_Res)  
}  
H\_calc <- function(X) {  
 X <- as.matrix(X)  
 H <- X %\*% solve(t(X) %\*% X) %\*% t(X)  
 return(H)  
}  
ones <- rep(1, nrow(df))  
  
# residuals  
e <- best\_1a$residuals  
  
# standardized residuals  
standardized\_res <- sapply(1:n, function(x) e[x] / sqrt(MS\_Res))  
  
# studentized residuals  
beta\_hat <- best\_1a$coefficients  
X <- cbind(ones, df[,c('x2','x7','x8')])  
H <- H\_calc(X)  
SS\_Res <- SS\_Res\_calc(y = df$y, beta\_hat = beta\_hat, X = X)  
p <- ncol(X)  
MS\_Res <- SS\_Res / (n - p)  
H\_diag <- diag(H)  
studentized\_residuals <- sapply(1:n, function(x) {  
 e[x] / sqrt(MS\_Res \* (1 - H\_diag[x]))  
})  
  
# PRESS residuals  
PRESS\_res <- sapply(1:n, function(x) e[x] / (1 - H\_diag[x]))  
  
# R-Student residuals  
S\_squared <- sapply(1:n, function(x) {  
 ((n - p) \* MS\_Res - ((e[x]^2) / (1 - H\_diag[x]))) / (n - p - 1)  
})  
  
R\_student\_res <- sapply(1:n, function(x) {  
 e[x] / sqrt(S\_squared[x] \* (1 - H\_diag[x]))  
})  
  
# residual table  
res\_table <- data.frame(  
 Observation = seq(1, n),  
 Residual = e,  
 Standardized\_Residuals = standardized\_res,  
 Studentized\_Residuals = studentized\_residuals,  
 R\_Student = R\_student\_res,  
 PRESS = PRESS\_res  
)  
res\_table <- round(res\_table, 4)  
abs(res\_table)  
write.table(res\_table, file = 'res\_table.txt', sep = ',')  
  
norm\_prob\_plot <- function(residual\_var, x\_label,  
 main\_title = 'Normal Probability Plot',  
 y\_label = 'Probability', n\_size=n) {  
 ones <- rep(1, n)  
 sorted\_residuals <- sort(residual\_var)  
 cumulative\_probability <- (1:n\_size - 0.5) / n\_size  
 plot(sorted\_residuals, cumulative\_probability, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label)  
 X\_temp <- cbind(ones, sorted\_residuals)  
 beta\_hat\_temp <- beta\_hat\_calc(X=X\_temp,y=cumulative\_probability)  
 abline(beta\_hat\_temp)  
}  
norm\_prob\_plot(residual\_var = best\_1a$residuals, x\_label = 'Sorted Residuals')  
  
order(e, decreasing = FALSE)  
e[order(e, decreasing = FALSE)]  
  
# res vs fitted  
y\_hat <- best\_1a$fitted.values  
res\_vs\_fitted\_plot <- function(residual\_var,  
 main\_title,  
 y\_label,  
 x\_label = 'Predicted Response',  
 pred\_response = y\_hat) {  
 plot(pred\_response, residual\_var, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label,  
 ylim = c(min(residual\_var)-sd(residual\_var),  
 max(residual\_var)+sd(residual\_var)))  
}  
par(mfrow = c(2,2))  
res\_vs\_fitted\_plot(residual\_var = e,  
 main\_title = 'Residuals vs. Predicted Response',  
 y\_label = 'Residuals')  
res\_vs\_fitted\_plot(residual\_var = studentized\_residuals,  
 main\_title = 'Studentized Residuals vs. Predicted Response',  
 y\_label = 'Studentized Residuals')  
res\_vs\_fitted\_plot(residual\_var = R\_student\_res,  
 main\_title = 'R-student vs. Predicted Response',  
 y\_label = 'R-student')  
res\_vs\_fitted\_plot(residual\_var = PRESS\_res,  
 main\_title = 'PRESS Residuals vs. Predicted Response',  
 y\_label = 'PRESS Residuals')  
  
# res vs regressor  
# R-student  
res\_vs\_regressor <- function(residual\_var,  
 main\_title1, main\_title2, main\_title3,  
 ylabel,  
 X\_df=X) {  
 x2 <- X\_df[,2]; x7 <- X\_df[,3]; x8 <- X\_df[,4]  
 plot(x2, residual\_var,  
 main = main\_title1,  
 ylab = ylabel,  
 xlab = col\_names[2])  
 abline(h = 0)  
   
 plot(x7, residual\_var,  
 main = main\_title2,  
 ylab = ylabel,  
 xlab = col\_names[7])  
 abline(h = 0)  
   
 plot(x8, residual\_var,  
 main = main\_title2,  
 ylab = ylabel,  
 xlab = col\_names[8])  
 abline(h = 0)  
}  
  
par(mfrow = c(2,2))  
res\_vs\_regressor(residual\_var = R\_student\_res,  
 main\_title1 = 'R-Student Residuals vs. x2',  
 main\_title2 = 'R-Student Residuals vs. x7',  
 main\_title3 = 'R-Student Residuals vs. x8',  
 ylabel = 'R-Student Residuals',  
 X\_df = X)  
par(mfrow = c(2,2))  
plot(df$x2, df$y,  
 main = 'y vs x2',  
 xlab = 'x2', ylab = 'y')  
plot(df$x7, df$y,  
 main = 'y vs x7',  
 xlab = 'x7', ylab = 'y')  
points(43.8, 2, pch = 3, col = 'red')  
plot(df$x8, df$y,  
 main = 'y vs x8',  
 xlab = 'x8', ylab = 'y')  
  
### model validation  
# VIFs  
car::vif(best\_1a)  
  
# data splitting  
SS\_T\_calc <- function(y) {  
 n <- length(y)  
 SS\_T <- (t(y) %\*% y) - ((sum(y)^2) / n)  
 return(SS\_T)  
}  
SS\_T <- SS\_T\_calc(y = df$y)  
PRESS <- sum(PRESS\_res^2)  
R\_squared\_pred <- 1 - PRESS / SS\_T  
  
# data split  
df\_split <- data.frame(y = df$y, x2 = df$x2, x7 = df$x7, d8 = df$x8)  
n\_train <- n \* 0.5  
n\_test <- n \* 0.5  
idx <- 1:n  
set.seed(1)  
train\_idx <- sort(sample(idx, n\_train))  
test\_idx <- idx[!(idx %in% train\_idx)]  
model\_test <- lm(y~x2+x7+x8, data = df\_split[train\_idx,])  
y\_pred <- predict(model\_test, df\_split[test\_idx,])  
test\_error <- sum((y\_pred - df\_split[test\_idx,1])^2) # 62.40468  
  
# compare to intercept-only  
model\_base <- lm(y~1, data = df\_split[train\_idx,])  
y\_pred <- predict(model\_base, df\_split[test\_idx,])  
test\_error <- sum((y\_pred - df\_split[test\_idx,1])^2) # 185  
  
# compare to full model  
model\_full <- lm(y~., data = df[train\_idx,])  
y\_pred <- predict(model\_full, df[test\_idx,])  
test\_error <- sum((y\_pred - df[test\_idx,1])^2) # 295.5793  
  
  
### polynomial  
# Reference: https://www.ics.uci.edu/~jutts/201-F13/Lecture13.pdf  
# Reference: https://stats.stackexchange.com/questions/25975/how-to-add-second-order-terms-into-the-model-in-r/25977  
X\_poly <- as.data.frame(cbind(y=df$y, poly(X[,2:ncol(X)], degree = 2)))  
model\_6 <- lm(y~., data=X\_poly)  
summary(model\_6)  
  
### CI's  
X <- as.matrix(X)  
C <- solve(t(X) %\*% X)  
SS\_Res <- SS\_Res\_calc(y = df$y, beta\_hat = coef(best\_1a), X = X)  
sigma\_hat\_squared <- SS\_Res / (n - p)  
  
standard\_errors <- sqrt(as.vector(sigma\_hat\_squared) \* diag(C))  
alpha <- 0.05  
t\_value <- qt(p = 1 - alpha / 2, df = n - p)  
round(coef(best\_1a) + t\_value \* standard\_errors, 4)  
round(coef(best\_1a) - t\_value \* standard\_errors, 4)  
apply(X, 2, function(x) c(min(x), max(x)))  
  
### Unit normal scaling  
sample\_means <- colMeans(df)  
sample\_sd <- sqrt(diag(cov(df)))  
df\_centered <- sweep(df, 2, sample\_means)  
df\_normal\_scaled <- sweep(df\_centered, 2, sample\_sd, FUN = "/")  
  
model\_3b <- lm(y~1, data = df\_normal\_scaled)  
anova(model\_3b, model\_3)  
model\_3 <- lm(y~x2+x7+x8, data = df\_normal\_scaled)  
round(summary(model\_3)$coefficients[,4], 4)  
  
# CI  
X <- df\_normal\_scaled[,c('x2', 'x7', 'x8')]  
X <- as.matrix(cbind(ones, X))  
C <- solve(t(X) %\*% X)  
SS\_Res <- SS\_Res\_calc(y = df\_normal\_scaled$y, beta\_hat = coef(model\_3), X = X)  
sigma\_hat\_squared <- SS\_Res / (n - p)  
  
standard\_errors <- sqrt(as.vector(sigma\_hat\_squared) \* diag(C))  
alpha <- 0.05  
t\_value <- qt(p = 1 - alpha / 2, df = n - p)  
round(coef(model\_3) + t\_value \* standard\_errors, 4)  
round(coef(model\_3) - t\_value \* standard\_errors, 4)