Multiple Regression

Zijian Du

February 17, 2018

```r
yx= read.csv("sim-reg-data.csv")
print(summary(yx))
```

```
##        y                 x1                x2
##  Min.   :-2.2770   Min.   :0.00348   Min.   : 0.004425
##  1st Qu.: 0.4562   1st Qu.:2.33474   1st Qu.: 2.398081
##  Median : 1.2172   Median :4.83693   Median : 4.775174
##  Mean   : 1.2110   Mean   :4.90519   Mean   : 4.920732
##  3rd Qu.: 2.0133   3rd Qu.:7.48680   3rd Qu.: 7.440223
##  Max.   : 5.2494   Max.   :9.99582   Max.   : 9.999662
##        x3                  x4                  x5
##  Min.   :0.0001646   Min.   :0.001629   Min.   :0.001979
##  1st Qu.:0.2444854   1st Qu.:0.247012   1st Qu.:0.248805
##  Median :0.5096922   Median :0.514615   Median :0.515297
##  Mean   :0.5017638   Mean   :0.506709   Mean   :0.506733
##  3rd Qu.:0.7567860   3rd Qu.:0.761047   3rd Qu.:0.761768
##  Max.   :0.9999581   Max.   :1.006039   Max.   :1.005441
```

firstly do a multiple regression using all x variables

```
##
## Call:
## lm(formula = y ~ ., data = yx)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.024640   0.089157  -0.276    0.782
## x1           0.025323   0.007686   3.295    0.001 **
## x2           0.035590   0.007742   4.597 4.55e-06 ***
## x3           4.248433  10.888697   0.390    0.696
## x4          -5.126662   7.773133  -0.660    0.510
## x5           2.767445   7.835586   0.353    0.724
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

compare out of sample error using x1, x2 and just x3

```r
# number of rows (samples)
n = nrow(yx)
nd =100
set.seed(99)
# define function to calculate RMSE
rmse =function(y, yhat){sqrt(mean((y-yhat)^2))}
n_train = floor(0.75*n)
# we want to do nd times and stores the rmse into the matrix
resM=matrix(0.0, nd, 2)
for(i in 1:nd)
{
  # sample n samples, create permutation and get n_train of them
  ii =sample(1:n, n_train)
  dftrain =yx[ii,]
  dftest=yx[-ii,]
  # use model to do regression and stroe result into the matrix
  lm12 = lm(y~x1+x2,dftrain)
  resM[i,1]= rmse(dftest$y, predict(lm12, dftest))
  lm3 = lm(y~x3, dftrain)
  resM[i,2]=rmse(dftest$y, predict(lm3, dftest))
}
print(resM)
```

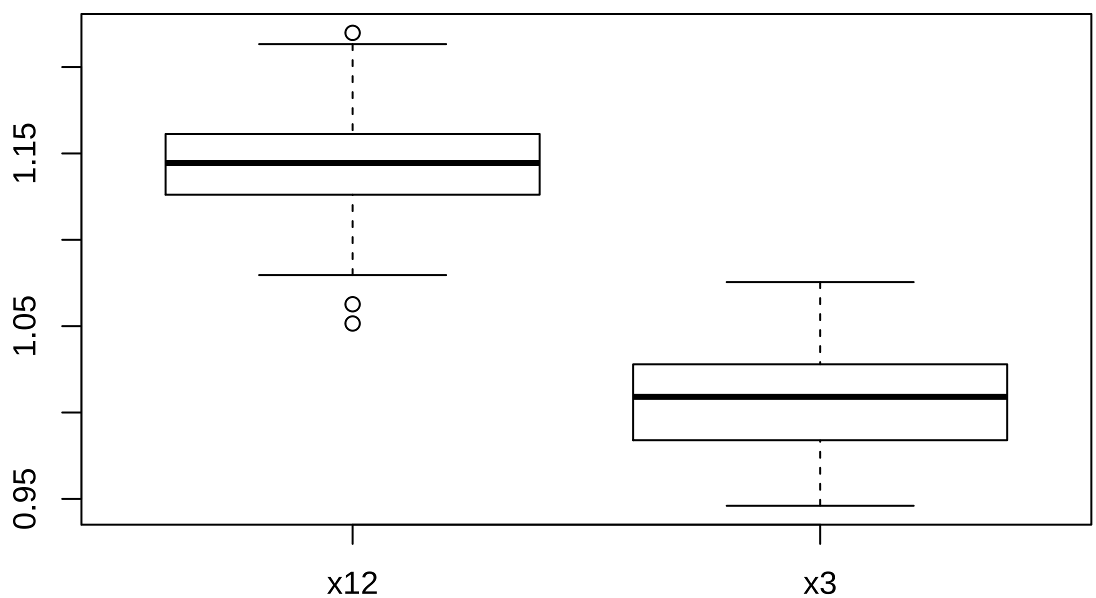
```
##             [,1]      [,2]
##   [1,] 1.084532 0.9465677
##   [2,] 1.159760 1.0207619
##   [3,] 1.140393 1.0304673
##   [4,] 1.081111 0.9760524
##   [5,] 1.193746 1.0260841
##   [6,] 1.136701 0.9894967
##   [7,] 1.178023 1.0447490
##   [8,] 1.184070 1.0323363
##   [9,] 1.108348 0.9626128
##  [10,] 1.176449 1.0366901
##  [11,] 1.143812 0.9970594
##  [12,] 1.156065 1.0229532
##  [13,] 1.151103 0.9960485
##  [14,] 1.143408 0.9990172
##  [15,] 1.082749 0.9581638
##  [16,] 1.186524 1.0530382
##  [17,] 1.171890 1.0755028
##  [18,] 1.145289 1.0100585
##  [19,] 1.172381 1.0191388
##  [20,] 1.137630 0.9861111
##  [21,] 1.180952 0.9981352
##  [22,] 1.149686 1.0090806
##  [23,] 1.167521 1.0558872
##  [24,] 1.150178 1.0277853
##  [25,] 1.062680 0.9661059
##  [26,] 1.108091 0.9790073
##  [27,] 1.144355 1.0209601
```

```
##  [28,] 1.193359 1.0280426
##  [29,] 1.116866 0.9704890
##  [30,] 1.123000 0.9763653
##  [31,] 1.113875 1.0202733
##  [32,] 1.136630 0.9685605
##  [33,] 1.152949 1.0281195
##  [34,] 1.142520 0.9935414
##  [35,] 1.113631 1.0025701
##  [36,] 1.194388 1.0481231
##  [37,] 1.157918 1.0169777
##  [38,] 1.143830 1.0312722
##  [39,] 1.155738 1.0268724
##  [40,] 1.122530 0.9789579
##  [41,] 1.112717 0.9763913
##  [42,] 1.149733 1.0301176
##  [43,] 1.144565 1.0178586
##  [44,] 1.159356 0.9935109
##  [45,] 1.119263 0.9836522
##  [46,] 1.130186 0.9509780
##  [47,] 1.160347 1.0463073
##  [48,] 1.113564 1.0032611
##  [49,] 1.219810 1.0413093
##  [50,] 1.163381 1.0398078
##  [51,] 1.140961 1.0151551
##  [52,] 1.153670 1.0018993
##  [53,] 1.092996 0.9720844
##  [54,] 1.171326 1.0034216
##  [55,] 1.090527 0.9709416
##  [56,] 1.162226 1.0334637
##  [57,] 1.116269 0.9921441
##  [58,] 1.152287 1.0462352
##  [59,] 1.086699 0.9594445
##  [60,] 1.136125 0.9842839
##  [61,] 1.145939 1.0229774
##  [62,] 1.118734 0.9800884
##  [63,] 1.144952 1.0486906
##  [64,] 1.129948 1.0114813
##  [65,] 1.127878 0.9937986
##  [66,] 1.140320 1.0189492
##  [67,] 1.109268 0.9460153
##  [68,] 1.155567 1.0063649
##  [69,] 1.134475 1.0090669
##  [70,] 1.148127 0.9934541
##  [71,] 1.147409 1.0167974
##  [72,] 1.147652 1.0031857
##  [73,] 1.139542 0.9990304
##  [74,] 1.199937 1.0252645
##  [75,] 1.132503 1.0033179
##  [76,] 1.171519 1.0359515
##  [77,] 1.200366 1.0616450
##  [78,] 1.149828 0.9931991
##  [79,] 1.124358 0.9754937
##  [80,] 1.172001 1.0532724
##  [81,] 1.169434 1.0373105
```

```
##  [82,] 1.172918 1.0184050
##  [83,] 1.051536 0.9557478
##  [84,] 1.130450 0.9972808
##  [85,] 1.137225 0.9723699
##  [86,] 1.150057 1.0209316
##  [87,] 1.165861 1.0172905
##  [88,] 1.147746 1.0337237
##  [89,] 1.101526 0.9809385
##  [90,] 1.079561 0.9636960
##  [91,] 1.131713 1.0147510
##  [92,] 1.184355 1.0753226
##  [93,] 1.213280 1.0711508
##  [94,] 1.118285 0.9726385
##  [95,] 1.142860 1.0145129
##  [96,] 1.133813 1.0094608
##  [97,] 1.141899 0.9748961
##  [98,] 1.147745 1.0036927
##  [99,] 1.132330 1.0241265
## [100,] 1.168358 1.0088372
```

boxplot for the resM

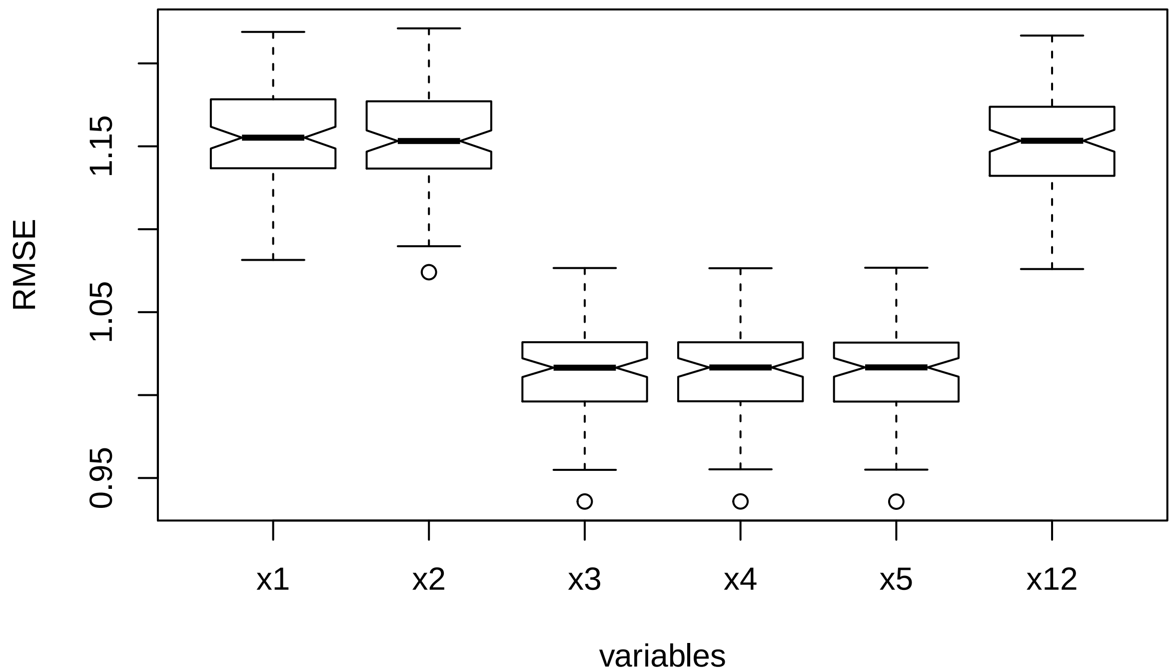
```r
colnames(resM)=c("x12","x3")
boxplot(resM)
```



it can be observed that using x3 for regression is indeed better compared with using x1 and x2.

now try each regression variable x1,2,3,4,5 and compare with x12

```
n =nrow(yx)
nd =100
set.seed(98)
resM =matrix(0.0, nd, 6)
for(i in 1:nd)
{
  ii = sample(1:n, n_train)
  dftrain=yx[ii,]
  dftest =yx[-ii,]
  lm1=lm(y~x1, dftrain)
  lm2=lm(y~x2, dftrain)
  lm3=lm(y~x3, dftrain)
  lm4=lm(y~x4, dftrain)
  lm5=lm(y~x5, dftrain)
  lm6=lm(y~x1+x2, dftrain)
  # calculate RMSE
  resM[i,1]= rmse(dftest$y, predict(lm1, dftest))
  resM[i,2]=rmse(dftest$y, predict(lm2, dftest))
  resM[i,3] =rmse(dftest$y, predict(lm3, dftest))
  resM[i,4]=rmse(dftest$y, predict(lm4, dftest))
  resM[i,5]=rmse(dftest$y, predict(lm5,dftest))
  resM[i,6]=rmse(dftest$y, predict(lm6, dftest))
}
colnames(resM)=c("x1","x2","x3","x4","x5","x12")
boxplot(resM, range=1.5, notch=TRUE, plot = TRUE, xlab="variables", ylab="RMSE")
```



using any of x3, x4, x5 to do regression will have lower out of sample prediction RMSE. Using just x1, x2 or combination of x1 and x2 have similar out of sample RMSE and much higher compared to x3, x4, x5

compare the analytical solution using least square as loss function with the result given by linear model in R

```
x=yx[,-1]
y=yx$y
# convert the dataframe to numerical matrix first
y=data.matrix(y)
x= data.matrix(x)
xtx=t(x)%*%x
# compute the analytical value of beta hat
bhat = solve(xtx)%*%t(x)%*%y
print(bhat)
```

```
##           [,1]
## x1  0.02446189
## x2  0.03466823
## x3  6.15928915
## x4 -6.07225532
## x5  1.79263977
```

```
lmf = lm(y~., data.frame(x,y))
lmf$coefficients
```

```
## (Intercept)          x1          x2          x3          x4          x5
## -0.02464036  0.02532344  0.03558996  4.24843259 -5.12666170  2.76744491
```

the number calculated from analytical solution and lm model doesnt match exactly

now first order condition

```
t(x)%*%(y-x%*%matrix(bhat, ncol=1))
```

```
##             [,1]
## x1 -3.482625e-08
## x2 -5.630732e-08
## x3 -3.777574e-09
## x4 -3.825048e-09
## x5 -3.822846e-09
```

first order condition satisfies

now get the sigma and standard errors for linear regression fits

```
print(summary(lm1)$sigma)
```

```
## [1] 1.150693
```

```
print(summary(lm2)$sigma)
```

```
## [1] 1.148451
```

```
print(summary(lm3)$sigma)
```

```
## [1] 1.013627
```

```
print(summary(lm4)$sigma)
```

```
## [1] 1.013851
```

```
print(summary(lm5)$sigma)
```

```
## [1] 1.013542
```

```
print(summary(lm6)$sigma)
```

```
## [1] 1.145336
```

get the value of standard error

```
x=yx[,-1]
y=yx$y
# convert the dataframe to numerical matrix first
```

```r
y=data.matrix(y)
x= data.matrix(x)
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm1)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.008028412 0.007984268 9.612978094 7.976270263 7.995922593
```

```r
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm2)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.008012771 0.007968713 9.594250253 7.960731029 7.980345073
```

```r
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm3)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.007072101 0.007033215 8.467919733 7.026169798 7.043481224
```

```r
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm4)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.007073665 0.007034771 8.469793174 7.027724267 7.045039522
```

```r
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm5)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.007071506 0.007032623 8.467207571 7.025578889 7.042888859
```

```r
print(sqrt(diag(solve(t(x)%*%x)))*summary(lm6)$sigma)
```

```
##          x1          x2          x3          x4          x5
## 0.007991039 0.007947101 9.568229311 7.939140418 7.958701266
```

```r
summary(lm1)
```

```
## 
## Call:
## lm(formula = y ~ x1, data = dftrain)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.4913 -0.7652  0.0138  0.7936  3.6937 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1.07076    0.05699  18.789  < 2e-16 ***
## x1           0.03098    0.01012   3.062  0.00224 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.151 on 1498 degrees of freedom
## Multiple R-squared:  0.00622,    Adjusted R-squared:  0.005557 
## F-statistic: 9.376 on 1 and 1498 DF,  p-value: 0.002237
```

```r
summary(lm2)
```

```
## 
## Call:
## lm(formula = y ~ x2, data = dftrain)
## 
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4097 -0.7653  0.0219  0.7711  3.5504
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.02594    0.05777  17.758  < 2e-16 ***
## x2           0.03928    0.01005   3.907 9.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.148 on 1498 degrees of freedom
## Multiple R-squared:  0.01009,    Adjusted R-squared:  0.009428
## F-statistic: 15.27 on 1 and 1498 DF,  p-value: 9.75e-05
```

```r
summary(lm3)
```

```
##
## Call:
## lm(formula = y ~ x3, data = dftrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5802 -0.6428  0.0430  0.6582  3.0865
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.28380    0.05153   5.508 4.27e-08 ***
## x3           1.88088    0.08920  21.086  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.014 on 1498 degrees of freedom
## Multiple R-squared:  0.2289, Adjusted R-squared:  0.2284
## F-statistic: 444.6 on 1 and 1498 DF,  p-value: < 2.2e-16
```

```r
summary(lm4)
```

```
##
## Call:
## lm(formula = y ~ x4, data = dftrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5761 -0.6414  0.0446  0.6602  3.0803
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.27496    0.05193   5.295 1.37e-07 ***
## x4           1.87994    0.08924  21.065  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.014 on 1498 degrees of freedom
## Multiple R-squared:  0.2285, Adjusted R-squared:  0.228
```

```
## F-statistic: 443.7 on 1 and 1498 DF,  p-value: < 2.2e-16
```

```r
summary(lm5)
```

```
##
## Call:
## lm(formula = y ~ x5, data = dftrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5763 -0.6432  0.0385  0.6552  3.0777
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.27395    0.05191   5.277 1.51e-07 ***
## x5           1.88208    0.08923  21.093  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.014 on 1498 degrees of freedom
## Multiple R-squared:  0.229,  Adjusted R-squared:  0.2285
## F-statistic: 444.9 on 1 and 1498 DF,  p-value: < 2.2e-16
```

```r
summary(lm6)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = dftrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4053 -0.7441  0.0276  0.7810  3.5065
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.88136    0.07485  11.775  < 2e-16 ***
## x1           0.03048    0.01007   3.026  0.00252 **
## x2           0.03889    0.01003   3.879  0.00011 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.145 on 1497 degrees of freedom
## Multiple R-squared:  0.01611,    Adjusted R-squared:  0.01479
## F-statistic: 12.25 on 2 and 1497 DF,  p-value: 5.261e-06
```

correlation

demeaning the data is equivalent getting rid of the intercept term since now that the data mean are at the origin

```r
xyd = read.csv("sim-reg-data.csv")
lmd = lm(y~.,xyd)
```

```r
summary(lmd)
```

```
##
## Call:
## lm(formula = y ~ ., data = xyd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.024640   0.089157  -0.276    0.782
## x1           0.025323   0.007686   3.295    0.001 **
## x2           0.035590   0.007742   4.597 4.55e-06 ***
## x3           4.248433  10.888697   0.390    0.696
## x4          -5.126662   7.773133  -0.660    0.510
## x5           2.767445   7.835586   0.353    0.724
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

now demeaned data

```r
xydd = xyd
for (i in 2:6)
{
  xydd[[i]] = xydd[[i]]-mean(xydd[[i]])
}
lmdd=lm(y~.,xydd)
summary(lmdd)
```

```
##
## Call:
## lm(formula = y ~ ., data = xydd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.211044   0.022515  53.788  < 2e-16 ***
## x1           0.025323   0.007686   3.295    0.001 **
## x2           0.035590   0.007742   4.597 4.55e-06 ***
## x3           4.248433  10.888697   0.390    0.696
## x4          -5.126662   7.773133  -0.660    0.510
## x5           2.767445   7.835586   0.353    0.724
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

```r
mean(xyd$y)
```

```
## [1] 1.211044
```

now look at the correlation between y, x1-5 yhat and e

```r
# cbind: combine into a new data.frame
fmat = cbind(xyd, lmd$fitted,lmd$residuals)
names(fmat)[c(7,8)]=c("yhat","e")
cor(fmat)
```

```
##               y           x1            x2            x3            x4
## y    1.00000000 8.462773e-02  8.654911e-02  4.802826e-01  4.801103e-01
## x1   0.08462773 1.000000e+00  2.637943e-02  3.653650e-02  3.620588e-02
## x2   0.08654911 2.637943e-02  1.000000e+00 -9.683881e-03 -9.707390e-03
## x3   0.48028258 3.653650e-02 -9.683881e-03  1.000000e+00  9.999510e-01
## x4   0.48011029 3.620588e-02 -9.707390e-03  9.999510e-01  1.000000e+00
## x5   0.48031024 3.661684e-02 -9.893077e-03  9.999518e-01  9.999054e-01
## yhat 0.49334026 1.715403e-01  1.754349e-01  9.735321e-01  9.731829e-01
## e    0.86983641 6.002102e-17 -6.686977e-17 -2.697639e-17 -3.822676e-17
##                 x5          yhat             e
## y     4.803102e-01  4.933403e-01  8.698364e-01
## x1    3.661684e-02  1.715403e-01  6.002102e-17
## x2   -9.893077e-03  1.754349e-01 -6.686977e-17
## x3    9.999518e-01  9.735321e-01 -2.697639e-17
## x4    9.999054e-01  9.731829e-01 -3.822676e-17
## x5    1.000000e+00  9.735882e-01 -7.551840e-17
## yhat  9.735882e-01  1.000000e+00 -1.037494e-17
## e    -7.551840e-17 -1.037494e-17  1.000000e+00
```

the residuals is uncorrelated with the fitted value is because the residual is from a indepedent distribution (a gaussian for example)

the square of y-yhat correlation is the same as the R square in multiple regression

orthogonalized regression

```r
lmfy=lm(y~x1+x2+x3+x4+x5, xyd)
summary(lmfy)
```

```
##
## Call:
```

```
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5, data = xyd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.024640   0.089157  -0.276    0.782
## x1           0.025323   0.007686   3.295    0.001 **
## x2           0.035590   0.007742   4.597 4.55e-06 ***
## x3           4.248433  10.888697   0.390    0.696
## x4          -5.126662   7.773133  -0.660    0.510
## x5           2.767445   7.835586   0.353    0.724
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

regress x5 on x1-4 and then replace x5 with the residue of this regression

```r
# regress x5 on other xs
lmf5=lm(x5~x1+x2+x3+x4, xyd)
# get the residuals of x5 on other xs
e5=lmf5$residuals
# combine the data.frame with the residuals
xyde=cbind(xyd[,1:5],e5)
# regress again
lmfe=lm(y~.,xyde)
summary(lmfe)
```

```
##
## Call:
## lm(formula = y ~ ., data = xyde)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.010466   0.079613  -0.131 0.895429
## x1           0.025350   0.007685   3.298 0.000989 ***
## x2           0.035531   0.007740   4.591  4.7e-06 ***
## x3           6.942769   7.769706   0.894 0.371660
## x4          -5.054789   7.770469  -0.651 0.515436
## e5           2.767445   7.835586   0.353 0.723984
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

the coefficients from this regression has the same x1-x4 coefficients compared to last regression

the coefficients for e5 is the same as the coefficients for x5 in the last regression

this can be explained by the fact that the coefficients for x5 is only determined by the part not explained by x1-x4 altogether, which is the residual/orthogonal portion of x5

```r
lmf=lm(y~.,xyd)
shat=summary(lmf)$sigma
shat/sqrt(sum(e5^2))
```

```
## [1] 7.835586
```

this number is the standard error for the coefficient of x5

R2 from regression of x5 on x1-x4 is 0.2434

run the regression of y on just x5 and compare with if run regression ofy on x1-5

```r
lm5=lm(y~x5, xyd)
summary(lm5)
```

```
##
## Call:
## lm(formula = y ~ x5, data = xyd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5624 -0.6660  0.0408  0.6587  3.8983
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.25053    0.04532   5.528 3.67e-08 ***
## x5           1.89550    0.07744  24.478  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.014 on 1998 degrees of freedom
## Multiple R-squared:  0.2307, Adjusted R-squared:  0.2303
## F-statistic: 599.2 on 1 and 1998 DF,  p-value: < 2.2e-16
```

```
lm5$coefficients
```

```
## (Intercept)          x5
##   0.2505312   1.8955035
```

```
lm_all=lm(y~x1+x2+x3+x4+x5, xyd)
summary(lm_all)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5, data = xyd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4575 -0.6490  0.0287  0.6639  4.0229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.024640   0.089157  -0.276    0.782
## x1           0.025323   0.007686   3.295    0.001 **
## x2           0.035590   0.007742   4.597 4.55e-06 ***
## x3           4.248433  10.888697   0.390    0.696
## x4          -5.126662   7.773133  -0.660    0.510
## x5           2.767445   7.835586   0.353    0.724
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 1994 degrees of freedom
## Multiple R-squared:  0.2434, Adjusted R-squared:  0.2415
## F-statistic: 128.3 on 5 and 1994 DF,  p-value: < 2.2e-16
```

```
lm_all$coefficients
```

```
## (Intercept)          x1          x2          x3          x4          x5
## -0.02464036  0.02532344  0.03558996  4.24843259 -5.12666170  2.76744491
```

the std error for x5 coefficient in multiple regression(7.835) is much higher than that of single regression(0.07744). this is due to the fact that as we add x's the size of residue can go down and since variance of estimated parameters are inversely proportional to the residue size, the variance will increases in multiple regression hence result in higher standard error in predictions, which is in essense a bias variance trade off problem.