L14Ex_Acetylene_Rick_Davila

Rick Davila

6/01/2020

Perform data housekeeping - upload, name columns, display to make sure it reads properly, etc.

```
knitr::opts_chunk$set(echo = TRUE)

library('xlsx') # Needed to read data
```

```
## Warning: package 'xlsx' was built under R version 4.0.4
```

```
library(readxl)
library(ridge) # Needed for ridge regression in Example 9.2
```

```
## Warning: package 'ridge' was built under R version 4.0.4
```

```
library(pls) # Needed for principal component regression in Example 9.3
```

```
## Warning: package 'pls' was built under R version 4.0.4
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(MASS) # Needed for ginv() function
library(car) # Needed for vif() function
```

```
## Loading required package: carData
```

```
library(xtable)

rm(list = ls())

# Load data
Ex91 <- read.xlsx(
  "data-ex-9-1.xlsx",
  sheetIndex = 1, sheetName=NULL, rowIndex=NULL,
  startRow=NULL, endRow=NULL, colIndex= c(1,2,3,4,5),
  as.data.frame=TRUE, header=TRUE, colClasses=NA,
  keepFormulas=FALSE, encoding="unknown")

# Give labels to data columns
names(Ex91) <- c("Obs",
                 "conversion",
                 "temp",
                 "heptane",
                 "time")
attach(Ex91)

# Output data to make sure it reads properly
out <- as.data.frame(c(Ex91))
colnames(out) <- c("Obs",
                 "conversion",
                 "temp",
                 "heptane",
                 "time")
tab <- (xtable(out, digits=c(0,0,1,0,1,4)))
print(tab, type="html")
```

	Obs	conversion	temp	heptane	time
1	1	49.0	1300	7.5	0.0120
2	2	50.2	1300	9.0	0.0120
3	3	50.5	1300	11.0	0.0115
4	4	48.5	1300	13.5	0.0130
5	5	47.5	1300	17.0	0.0135
6	6	44.5	1300	23.0	0.0120
7	7	28.0	1200	5.3	0.0400
8	8	31.5	1200	7.5	0.0380
9	9	34.5	1200	11.0	0.0320
10	10	35.0	1200	13.5	0.0260
11	11	38.0	1200	17.0	0.0340
12	12	38.5	1200	23.0	0.0410
13	13	15.0	1100	5.3	0.0840
14	14	17.0	1100	7.5	0.0980
15	15	20.5	1100	11.0	0.0920
16	16	29.5	1100	17.0	0.0860

```
# Output data structure and dimensions
str(Ex91)
```

'data.frame': 16 obs. of 5 variables: $ Obs : num 1 2 3 4 5 6 7 8 9 10 … $ conversion: num 49 50.2 50.5 48.5 47.5 44.5 28 31.5 34.5 35 … $ temp : num 1300 1300 1300 1300 1300 1300 1200 1200 1200 1200 … $ heptane : num 7.5 9 11 13.5 17 23 5.3 7.5 11 13.5 … $ time : num 0.012 0.012 0.0115 0.013 0.0135 0.012 0.04 0.038 0.032 0.026 …

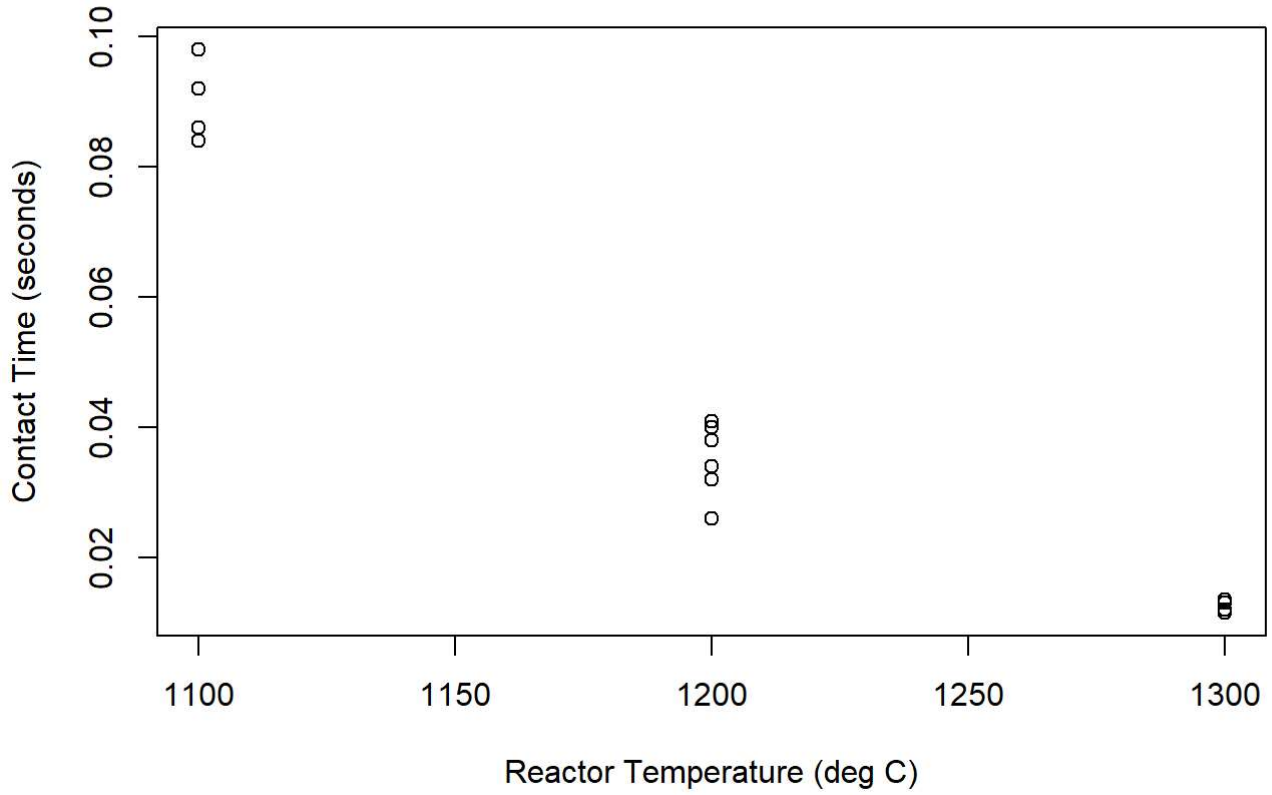
```
dim(Ex91)
```

[1] 16 5

Example 9.1 (p. 290-292)

Create scatterplot of Contact Time versus Reactor Temperature

```
plot(temp,time,
     xlab = "Reactor Temperature (deg C)",
     ylab = "Contact Time (seconds)")
```



Scale regressor variables using unit normal scaling

```
P <- conversion
Ts <- (temp - mean(temp))/sd(temp)
Hs <- (heptane - mean(heptane))/sd(heptane)
Cs <- (time - mean(time))/sd(time)
```

Fit second order model using scaled regressors, to include two-way interaction terms. Compare to Table 9.2 on p. 293

Note: Can do this model the “long way”, i.e. lm(y ~ x1 + x2 + x3 + x1x2 + x2x3… etc.) or using polym() function in R.

Model output for the polym() shortcut uses a particular notation. Don’t get confused - know what you’re looking at.

Model both ways and use each model to decipher the polym() output.

```
# model the "long way"
TsHs <- Ts*Hs
TsCs <- Ts*Cs
HsCs <- Hs*Cs
Ts_sqrd <- Ts*Ts
Hs_sqrd <- Hs*Hs
Cs_sqrd <- Cs*Cs

model.91 <- lm(P ~ Ts + Hs + Cs +
                 TsHs + TsCs + HsCs +
                 Ts_sqrd + Hs_sqrd + Cs_sqrd)

summary(model.91)
```

```
##
## Call:
## lm(formula = P ~ Ts + Hs + Cs + TsHs + TsCs + HsCs + Ts_sqrd +
##     Hs_sqrd + Cs_sqrd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3499 -0.3411  0.1297  0.5011  0.6720
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.8958     1.0916  32.884 5.26e-08 ***
## Ts            4.0038     4.5087   0.888 0.408719
## Hs            2.7783     0.3071   9.048 0.000102 ***
## Cs           -8.0423     6.0707  -1.325 0.233461
## TsHs         -6.4568     1.4660  -4.404 0.004547 **
## TsCs        -26.9804    21.0213  -1.283 0.246663
## HsCs         -3.7681     1.6553  -2.276 0.063116 .
## Ts_sqrd     -12.5236    12.3238  -1.016 0.348741
## Hs_sqrd      -0.9727     0.3746  -2.597 0.040844 *
## Cs_sqrd     -11.5932     7.7063  -1.504 0.183182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9014 on 6 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9943
## F-statistic: 289.7 on 9 and 6 DF,  p-value: 3.225e-07
```

```
anova(model.91)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
	<int>	<dbl>	<dbl>	<dbl>	<dbl>
Ts	1	1896.6769391	1896.6769391	2334.0918815	5.272579e-09
Hs	1	56.3015883	56.3015883	69.2859588	1.631506e-04
Cs	1	0.4407947	0.4407947	0.5424516	4.892028e-01
TsHs	1	143.4211476	143.4211476	176.4971827	1.124511e-05
TsCs	1	14.1014346	14.1014346	17.3535321	5.906382e-03
HsCs	1	1.2241399	1.2241399	1.5064532	2.656486e-01
Ts_sqrd	1	0.9480496	0.9480496	1.1666904	3.215755e-01
Hs_sqrd	1	3.8806445	3.8806445	4.7756055	7.153516e-02
Cs_sqrd	1	1.8390524	1.8390524	2.2631779	1.831824e-01
Residuals	6	4.8755843	0.8125974	*NA*	*NA*

1-10 of 10 rows

```
# model using polym()

model.91b <- lm(P ~ polym(Ts, Hs, Cs, degree = 2, raw=TRUE))
summary(model.91b)
```

```
##
## Call:
## lm(formula = P ~ polym(Ts, Hs, Cs, degree = 2, raw = TRUE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3499 -0.3411  0.1297  0.5011  0.6720
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                  35.8958     1.0916  32.884
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.0.0   4.0038     4.5087   0.888
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)2.0.0 -12.5236    12.3238  -1.016
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.1.0   2.7783     0.3071   9.048
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.1.0  -6.4568     1.4660  -4.404
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.2.0  -0.9727     0.3746  -2.597
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.0.1  -8.0423     6.0707  -1.325
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.0.1 -26.9804    21.0213  -1.283
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.1.1  -3.7681     1.6553  -2.276
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.0.2 -11.5932     7.7063  -1.504
##                                             Pr(>|t|)
## (Intercept)                                 5.26e-08 ***
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.0.0 0.408719
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)2.0.0 0.348741
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.1.0 0.000102 ***
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.1.0 0.004547 **
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.2.0 0.040844 *
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.0.1 0.233461
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)1.0.1 0.246663
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.1.1 0.063116 .
## polym(Ts, Hs, Cs, degree = 2, raw = TRUE)0.0.2 0.183182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9014 on 6 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9943
## F-statistic: 289.7 on 9 and 6 DF,  p-value: 3.225e-07
```

```
anova(model.91b)
```

	Df <int>	Sum Sq <dbl>	Mean Sq <dbl>	F value <dbl>	
polym(Ts, Hs, Cs, degree = 2, raw = TRUE)	9	2118.833791	235.4259767	289.7203	3.2
Residuals	6	4.875584	0.8125974	*NA*	

2 rows

Reproduce Table 9.3 on p. 295.

```
y <- (conversion - mean(conversion))/sqrt(sum((conversion - mean(conversion))^2))
x_1 <- (temp - mean(temp))/sqrt(sum((temp - mean(temp))^2))
x_2 <- (heptane - mean(heptane))/sqrt(sum((heptane - mean(heptane))^2))
x_3 <- (time - mean(time))/sqrt(sum((time - mean(time))^2))

x_12 <- x_1*x_2
x_12n <- (x_12 - mean(x_12))/sqrt(sum((x_12 - mean(x_12))^2))
x_13 <- x_1*x_3
x_13n <- (x_13 - mean(x_13))/sqrt(sum((x_13 - mean(x_13))^2))
x_23 <- x_2*x_3
x_23n <- (x_23 - mean(x_23))/sqrt(sum((x_23 - mean(x_23))^2))

x_11 <- x_1*x_1
x_11n <- (x_11 - mean(x_11))/sqrt(sum((x_11 - mean(x_11))^2))
x_22 <- x_2*x_2
x_22n <- (x_22 - mean(x_22))/sqrt(sum((x_22 - mean(x_22))^2))
x_33 <- x_3*x_3
x_33n <- (x_33 - mean(x_33))/sqrt(sum((x_33 - mean(x_33))^2))

# reproduce table 9.3
table_9pt3 <- data.frame(cbind(Obs,y,x_1,x_2,x_3,
                               x_12n,x_13n,x_23n,
                               x_11n,x_22n,x_33n))
out <- table_9pt3
colnames(out) <- c("Obs $i$","$y$","$x_1$","$x_2$","$x_3$",
                   "$x_1x_2$","$x_1x_3$","$x_2x_3$",
                   "$x_1^2$","$x_2^2$","$x_3^2$")
tab <- (xtable(out, digits=c(0,0,5,5,5,5,5,5,5,5,5,5)))
print(tab, type="html")
```

	Obs i	y	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	x_1^2	x_2^2	x_3^2
1	1	0.27979	0.28022	-0.22544	-0.23106	-0.33783	-0.02098	0.30974	0.07828	-0.04091	-0.03436
2	2	0.30583	0.28022	-0.15704	-0.23106	-0.25390	-0.02098	0.23679	0.07828	-0.13259	-0.03436
3	3	0.31234	0.28022	-0.06584	-0.23514	-0.14199	-0.02592	0.14076	0.07828	-0.20382	-0.02719
4	4	0.26894	0.28022	0.04817	-0.22289	-0.00210	-0.01111	0.01975	0.07828	-0.21088	-0.04833
5	5	0.24724	0.28022	0.20777	-0.21881	0.19375	-0.00617	-0.14055	0.07828	-0.06774	-0.05512
6	6	0.18214	0.28022	0.48138	-0.23106	0.52948	-0.02098	-0.44410	0.07828	0.59301	-0.03436
7	7	-0.17590	-0.04003	-0.32577	-0.00255	-0.00410	0.25902	0.07314	-0.29746	0.15287	-0.23562
8	8	-0.09995	-0.04003	-0.22544	-0.01887	-0.02168	0.26184	0.08894	-0.29746	-0.04091	-0.23430
9	9	-0.03486	-0.04003	-0.06584	-0.06784	-0.04966	0.27030	0.08992	-0.29746	-0.20382	-0.21829
10	10	-0.02401	-0.04003	0.04817	-0.11680	-0.06964	0.27876	0.04334	-0.29746	-0.21088	-0.18421
11	11	0.04109	-0.04003	0.20777	-0.05152	-0.09762	0.26748	0.01990	-0.29746	-0.06774	-0.22564
12	12	0.05194	-0.04003	0.48138	0.00561	-0.14558	0.25761	0.08177	-0.29746	0.59301	-0.23552
13	13	-0.45800	-0.36029	-0.32577	0.35653	0.45274	-0.29604	-0.46680	0.32877	0.15287	0.24361
14	14	-0.41460	-0.36029	-0.22544	0.47078	0.29447	-0.47378	-0.42060	0.32877	-0.04091	0.59999
15	15	-0.33865	-0.36029	-0.06584	0.42182	0.04267	-0.39761	-0.05888	0.32877	-0.20382	0.43520
16	16	-0.14335	-0.36029	0.20777	0.37285	-0.38899	-0.32143	0.42688	0.32877	-0.06774	0.28850

Using Table 9.3 data, define an X matrix and perform eigensystem analysis of X'X. Compare to Table 9.10 on p. 316

```
#X <- cbind(matrix(1,length(Distance),1),as.matrix(Num_Cases),as.m#atrix(Distance))

X <- cbind(as.matrix(x_1),as.matrix(x_2),as.matrix(x_3),
           as.matrix(x_12n),as.matrix(x_13n),as.matrix(x_23n),
           as.matrix(x_11n),as.matrix(x_22n),as.matrix(x_33n))

xtx <- t(X) %*% X

ev <- eigen(xtx)$vectors
lambda <- eigen(xtx)$values

# eigenvectors
as.matrix(ev)
```

```
##              [,1]        [,2]         [,3]        [,4]        [,5]        [,6]
##  [1,] -0.33869118 -0.10576185  0.649400352  0.01210303 -0.14255274  0.24948972
##  [2,] -0.13235854 -0.33901571 -0.001449742 -0.72472286  0.58389085 -0.02070227
##  [3,]  0.41362212  0.09799876 -0.469273272 -0.07524924  0.01846900 -0.01603244
##  [4,]  0.21951175 -0.54005584  0.087054424  0.36162698  0.16633108 -0.36623974
##  [5,] -0.44916992 -0.08641133 -0.287944901  0.18937808  0.09446653 -0.02922040
##  [6,] -0.25275845  0.51719668 -0.054712522 -0.34488485 -0.20103944 -0.31549409
##  [7,]  0.40557509  0.07476183  0.441686924 -0.21950491 -0.14430892 -0.54068922
##  [8,] -0.02555615 -0.53163292 -0.221360350 -0.34290917 -0.73441660  0.07075512
##  [9,]  0.46660059  0.09714164  0.143135377 -0.13272171  0.03477733  0.63655026
##               [,7]        [,8]         [,9]
##  [1,] -0.221473239 -0.53874409  0.174279714
##  [2,] -0.011303829 -0.02878024 -0.003323561
##  [3,] -0.168848097 -0.71289256  0.236874442
##  [4,] -0.589696292  0.11070661  0.002547737
##  [5,]  0.062038024  0.15041172  0.797341195
##  [6,] -0.620187593  0.14841251  0.008333273
##  [7,]  0.318930658  0.04769084  0.411690042
##  [8,] -0.002471479  0.07563638  0.005021362
##  [9,] -0.290458587  0.36852341  0.328881761
```

```
# eigenvalues
as.matrix(lambda)
```

```
##              [,1]
##  [1,] 4.205230e+00
##  [2,] 2.161999e+00
##  [3,] 1.138677e+00
##  [4,] 1.040475e+00
##  [5,] 3.852305e-01
##  [6,] 4.953807e-02
##  [7,] 1.362526e-02
##  [8,] 5.127803e-03
##  [9,] 9.693644e-05
```

Calculate kappa values and compare to p. 298

```
kappa <- max(lambda)/lambda
as.matrix(kappa)
```

```
##              [,1]
##  [1,]     1.000000
##  [2,]     1.945066
##  [3,]     3.693085
##  [4,]     4.041644
##  [5,]    10.916141
##  [6,]    84.888858
##  [7,]   308.634770
##  [8,]   820.084287
##  [9,] 43381.314197
```

Example 9.2 (p.307-308)

Perform ridge regression (by hand) using the standardized data from Table 9.3. Define the parameter k as a constant and vary to reproduce Table 9.8 on p. 308

Note: The textbook leaves off the intercept term. This is because the response and the regressors are both centered and scaled. Due to rounding, the output will give a small B_0 coefficient.

We could have also dropped the vector of ones from X

```
# create identity matrix (9x9)
I_matrix <- diag(9)

# k matrix
k_matrix <- c(0.000,0.001,0.004,0.008,0.016,0.032,0.064,0.128,0.512)

# create blank matrix for storage (9x9)
mstore <- matrix(nrow = 9, ncol = length(k_matrix))

for (kval in seq(1:length(k_matrix))) {
  matrix_in <- xtx + k_matrix[kval]*I_matrix
  beta_R <- ginv(matrix_in, tol = .Machine$double.eps) %*% t(X) %*%  y

  mstore[,kval] <- as.matrix(beta_R[,1])
}

out <- as.data.frame(mstore)
colnames(out) <- c("$0.000$","$0.001$","$0.004$",
                   "$0.008$","$0.016$","$0.032$",
                   "$0.064$","$0.128$","$0.512$")

rownames(out) <- c("$beta_{R1}$","$beta_{R2}$","$beta_{R3}$",
                   "$beta_{R12}$","$beta_{R13}$","$beta_{R23}$",
                   "$beta_{R11}$","$beta_{R22}$","$beta_{R33}$")

tab <- (xtable(out, digits=c(0,4,4,4,4,4,4,4,4,4)))
print(tab, type="html")
```

	0.000	0.001	0.004	0.008	0.016	0.032	0.064	0.128	0.512
$beta_{R1}$	0.3365	0.6765	0.6359	0.6001	0.5670	0.5390	0.5121	0.4805	0.3784
$beta_{R2}$	0.2335	0.2240	0.2197	0.2172	0.2147	0.2117	0.2066	0.1971	0.1554
$beta_{R3}$	-0.6759	-0.2133	-0.2672	-0.3135	-0.3515	-0.3735	-0.3799	-0.3723	-0.3108
$beta_{R12}$	-0.4800	-0.4479	-0.3913	-0.3437	-0.2880	-0.2329	-0.1863	-0.1509	-0.1045
$beta_{R13}$	-2.0340	-0.2776	-0.1352	-0.1018	-0.0810	-0.0676	-0.0570	-0.0455	-0.0093
$beta_{R23}$	-0.2657	-0.2173	-0.1536	-0.1020	-0.0433	0.0123	0.0562	0.0848	0.0991
$beta_{R11}$	-0.8345	0.0642	0.1213	0.1262	0.1253	0.1248	0.1257	0.1229	0.0827
$beta_{R22}$	-0.0904	-0.0731	-0.0621	-0.0558	-0.0509	-0.0480	-0.0464	-0.0444	-0.0342
$beta_{R33}$	-1.0009	-0.2450	-0.1313	-0.0825	-0.0455	-0.0266	-0.0250	-0.0338	-0.0585

There is also a package to perform ridge regression automatically

```
# example using automated calculation ... for k = 0.032
linRidgeMod <- linearRidge(y ~ x_1 + x_2 + x_3 + x_12n + x_13n + x_23n + x_11n + x_22n + x_33n,
                           lambda = 0.032)
print(summary(linRidgeMod), type="html")
```

```
##
## Call:
## linearRidge(formula = y ~ x_1 + x_2 + x_3 + x_12n + x_13n + x_23n +
##     x_11n + x_22n + x_33n, lambda = 0.032)
##
##
## Coefficients:
##               Estimate Scaled estimate Std. Error (scaled) t value (scaled)
## (Intercept) -6.122e-17              NA                  NA               NA
## x_1          5.390e-01       5.390e-01           3.704e-02           14.553
## x_2          2.117e-01       2.117e-01           2.780e-02            7.614
## x_3         -3.735e-01      -3.735e-01           3.722e-02           10.035
## x_12n       -2.329e-01      -2.329e-01           4.671e-02            4.987
## x_13n       -6.758e-02      -6.758e-02           1.441e-02            4.691
## x_23n        1.228e-02       1.228e-02           4.688e-02            0.262
## x_11n        1.248e-01       1.248e-01           4.347e-02            2.870
## x_22n       -4.804e-02      -4.804e-02           3.021e-02            1.590
## x_33n       -2.663e-02      -2.663e-02           4.998e-02            0.533
##             Pr(>|t|)
## (Intercept)       NA
## x_1          < 2e-16 ***
## x_2         2.66e-14 ***
## x_3          < 2e-16 ***
## x_12n       6.15e-07 ***
## x_13n       2.72e-06 ***
## x_23n         0.7934
## x_11n         0.0041 **
## x_22n         0.1117
## x_33n         0.5941
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge parameter: 0.032
##
## Degrees of freedom: model 5.891 , variance 5.173 , residual 6.609
```

```
# create blank matrix for storage (9x9)
mstore2 <- matrix(nrow = 9, ncol = length(k_matrix))

# loop through the values
for (kval in seq(1:length(k_matrix))) {
  linRidgeMod <- linearRidge(y ~ x_1 + x_2 + x_3 + x_12n + x_13n + x_23n + x_11n + x_22n + x_33
n,
                           lambda = k_matrix[kval])

  beta_R <- linRidgeMod$coef
  mstore2[,kval] <- as.matrix(beta_R[,1])
}

out <- as.data.frame(mstore2)
colnames(out) <- c("$0.000$","$0.001$","$0.004$",
                   "$0.008$","$0.016$","$0.032$",
                   "$0.064$","$0.128$","$0.512$")

rownames(out) <- c("$beta_{R1}$","$beta_{R2}$","$beta_{R3}$",
                   "$beta_{R12}$","$beta_{R13}$","$beta_{R23}$",
                   "$beta_{R11}$","$beta_{R22}$","$beta_{R33}$")

tab <- (xtable(out, digits=c(0,4,4,4,4,4,4,4,4,4)))
print(tab, type="html")
```

	0.000	0.001	0.004	0.008	0.016	0.032	0.064	0.128	0.512
$beta_{R1}$	0.3365	0.6765	0.6359	0.6001	0.5670	0.5390	0.5121	0.4805	0.3784
$beta_{R2}$	0.2335	0.2240	0.2197	0.2172	0.2147	0.2117	0.2066	0.1971	0.1554
$beta_{R3}$	-0.6759	-0.2133	-0.2672	-0.3135	-0.3515	-0.3735	-0.3799	-0.3723	-0.3108
$beta_{R12}$	-0.4800	-0.4479	-0.3913	-0.3437	-0.2880	-0.2329	-0.1863	-0.1509	-0.1045
$beta_{R13}$	-2.0340	-0.2776	-0.1352	-0.1018	-0.0810	-0.0676	-0.0570	-0.0455	-0.0093
$beta_{R23}$	-0.2657	-0.2173	-0.1536	-0.1020	-0.0433	0.0123	0.0562	0.0848	0.0991
$beta_{R11}$	-0.8345	0.0642	0.1213	0.1262	0.1253	0.1248	0.1257	0.1229	0.0827
$beta_{R22}$	-0.0904	-0.0731	-0.0621	-0.0558	-0.0509	-0.0480	-0.0464	-0.0444	-0.0342
$beta_{R33}$	-1.0009	-0.2450	-0.1313	-0.0825	-0.0455	-0.0266	-0.0250	-0.0338	-0.0585

Example 9.3 (p.316-319)

Perform Principal Component Regression using the pcr() function. Display $coefficients, $loadings, and $scores. Compare to Table 9.12 on p. 318 and Table 9.10 on p. 3.16

```
pcr.model <- pcr(y ~ x_1 + x_2 + x_3 + x_12n + x_13n + x_23n + x_11n + x_22n + x_33n)


pcr.model$coefficients
```

```
## , , 1 comps
##
##                 y
## x_1    0.119286603
## x_2    0.046616509
## x_3   -0.145677187
## x_12n -0.077311759
## x_13n  0.158197077
## x_23n  0.089021205
## x_11n -0.142843030
## x_22n  0.009000843
## x_33n -0.164336136
##
## , , 2 comps
##
##                 y
## x_1    0.118761294
## x_2    0.044932650
## x_3   -0.145190437
## x_12n -0.079994165
## x_13n  0.157767881
## x_23n  0.091590072
## x_11n -0.142471695
## x_22n  0.006360273
## x_33n -0.163853643
##
## , , 3 comps
##
##                 y
## x_1    0.50964555
## x_2    0.04406003
## x_3   -0.42765338
## x_12n -0.02759474
## x_13n -0.01555067
## x_23n  0.05865775
## x_11n  0.12338659
## x_22n -0.12687999
## x_33n -0.07769821
##
## , , 4 comps
##
##                 y
## x_1    0.50680870
## x_2    0.21392899
## x_3   -0.41001559
## x_12n -0.11235708
## x_13n -0.05993930
## x_23n  0.13949587
## x_11n  0.17483670
## x_22n -0.04650496
## x_33n -0.04658936
##
## , , 5 comps
```

```
##
##                y
## x_1    0.50548074
## x_2    0.21936825
## x_3   -0.40984354
## x_12n -0.11080761
## x_13n -0.05905930
## x_23n  0.13762307
## x_11n  0.17349238
## x_22n -0.05334645
## x_33n -0.04626539
##
## , , 6 comps
##
##                y
## x_1    0.55955409
## x_2    0.21488133
## x_3   -0.41331834
## x_12n -0.19018487
## x_13n -0.06539240
## x_23n  0.06924421
## x_11n  0.05630568
## x_22n -0.03801128
## x_33n  0.09169783
##
## , , 7 comps
##
##                y
## x_1    0.47464724
## x_2    0.21054775
## x_3   -0.47805014
## x_12n -0.41625848
## x_13n -0.04160870
## x_23n -0.16851894
## x_11n  0.17857507
## x_22n -0.03895878
## x_33n -0.01965613
##
## , , 8 comps
##
##                y
## x_1    0.75486617
## x_2    0.22551732
## x_3   -0.10725074
## x_12n -0.47384071
## x_13n -0.11984290
## x_23n -0.24571327
## x_11n  0.15376945
## x_22n -0.07829981
## x_33n -0.21133755
##
## , , 9 comps
##
##                y
```

```
## x_1    0.33648681
## x_2    0.23349593
## x_3   -0.67589625
## x_12n -0.47995686
## x_13n -2.03395608
## x_23n -0.26571830
## x_11n -0.83454188
## x_22n -0.09035419
## x_33n -1.00085767
```

```
pcr.model$loadings
```

```
##
## Loadings:
##       Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7 Comp 8 Comp 9
## x_1   -0.339  0.106 -0.649        -0.143  0.249  0.221  0.539  0.174
## x_2   -0.132  0.339        -0.725  0.584
## x_3    0.414         0.469                        0.169  0.713  0.237
## x_12n  0.220  0.540         0.362  0.166 -0.366  0.590 -0.111
## x_13n -0.449         0.288  0.189                       -0.150  0.797
## x_23n -0.253 -0.517        -0.345 -0.201 -0.315  0.620 -0.148
## x_11n  0.406        -0.442 -0.220 -0.144 -0.541 -0.319         0.412
## x_22n         0.532  0.221 -0.343 -0.734
## x_33n  0.467        -0.143 -0.133         0.637  0.290 -0.369  0.329
##
##                Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7 Comp 8 Comp 9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111
## Cumulative Var  0.111  0.222  0.333  0.444  0.556  0.667  0.778  0.889  1.000
```

```
pcr.model$scores
```

```
##         Comp 1       Comp 2      Comp 3        Comp 4        Comp 5
## 1  -0.28690261 -0.392872499 -0.28913032 -4.739909e-02 -0.2787385504
## 2  -0.25674961 -0.335363264 -0.32062355 -1.002155e-02 -0.1428408383
## 3  -0.21428631 -0.232931019 -0.35561835  2.031625e-02  0.0003526641
## 4  -0.17935422 -0.057765472 -0.36277947  3.712793e-02  0.1205896097
## 5  -0.12433170  0.261802466 -0.35237199  8.711463e-06  0.1737614026
## 6  -0.01573294  1.042049111 -0.26452817 -2.033993e-01 -0.0357791054
## 7  -0.31459412 -0.005685441  0.30221559  3.022900e-01 -0.2530074625
## 8  -0.33817857 -0.090649401  0.25482581  2.858091e-01 -0.0582057530
## 9  -0.37811434 -0.134788686  0.19865201  2.187104e-01  0.1502325054
## 10 -0.39378521 -0.084373112  0.17102538  1.481121e-01  0.2291050319
## 11 -0.40604548  0.039499473  0.23741504 -2.022899e-02  0.2159274568
## 12 -0.46186296  0.420139869  0.41701560 -4.886373e-01 -0.1302162345
## 13  0.90606065  0.309895052  0.10436827  3.166369e-01 -0.1423239975
## 14  1.14468703  0.070336943  0.02935362  1.476504e-01  0.0206675579
## 15  0.84966978 -0.257836893  0.05777543 -8.798317e-02  0.1194636896
## 16  0.46952059 -0.551457124  0.17240511 -6.189925e-01  0.0110120237
##          Comp 6       Comp 7       Comp 8        Comp 9
## 1   0.037813186 -0.020364838 -0.013626588  0.0005664330
## 2   0.022186452 -0.015566902 -0.003188625 -0.0005153534
## 3   0.009343832 -0.006574025  0.001877054 -0.0042355236
## 4  -0.020655638 -0.002853180  0.022454676  0.0024547420
## 5  -0.039520801  0.013789090  0.023000535  0.0044743588
## 6  -0.011779547  0.033121931 -0.025369790 -0.0026718571
## 7  -0.010681129  0.040704227  0.007333541  0.0014426376
## 8  -0.023995771  0.038242091  0.011933256 -0.0009601786
## 9  -0.018160807  0.019607921 -0.010278321 -0.0019575291
## 10  0.023231572 -0.018688238 -0.046079272  0.0035492776
## 11  0.020605869 -0.047874692  0.017771805 -0.0036880204
## 12  0.052817010 -0.025672909  0.017650600  0.0015209655
## 13 -0.110622081 -0.061154541 -0.002643078  0.0001967562
## 14  0.147192252  0.008654892  0.002414531  0.0014284114
## 15  0.004127371  0.025042797  0.007886915 -0.0026069071
## 16 -0.081901771  0.019586376 -0.011137238  0.0010017874
## attr(,"class")
## [1] "scores"
```

Perform Principal Component Regression (by hand) using the following steps:

Define matrix of eigenvectors of xTx, where X is defined by standardized variables (Table 9.3)

```
xTx <- t(X) %*% X
```

Calculate Z; omit intercept row/column because variables are standardized. Compare to $scores output from pcr().

```
# ev is the eigenvectgor matrix

Z <- X %*% as.matrix(ev)
Z
```

```
##              [,1]         [,2]        [,3]          [,4]          [,5]
##  [1,] -0.28690261  0.392872499  0.28913032 -4.739909e-02 -0.2787385504
##  [2,] -0.25674961  0.335363264  0.32062355 -1.002155e-02 -0.1428408383
##  [3,] -0.21428631  0.232931019  0.35561835  2.031625e-02  0.0003526641
##  [4,] -0.17935422  0.057765472  0.36277947  3.712793e-02  0.1205896097
##  [5,] -0.12433170 -0.261802466  0.35237199  8.711463e-06  0.1737614026
##  [6,] -0.01573294 -1.042049111  0.26452817 -2.033993e-01 -0.0357791054
##  [7,] -0.31459412  0.005685441 -0.30221559  3.022900e-01 -0.2530074625
##  [8,] -0.33817857  0.090649401 -0.25482581  2.858091e-01 -0.0582057530
##  [9,] -0.37811434  0.134788686 -0.19865201  2.187104e-01  0.1502325054
## [10,] -0.39378521  0.084373112 -0.17102538  1.481121e-01  0.2291050319
## [11,] -0.40604548 -0.039499473 -0.23741504 -2.022899e-02  0.2159274568
## [12,] -0.46186296 -0.420139869 -0.41701560 -4.886373e-01 -0.1302162345
## [13,]  0.90606065 -0.309895052 -0.10436827  3.166369e-01 -0.1423239975
## [14,]  1.14468703 -0.070336943 -0.02935362  1.476504e-01  0.0206675579
## [15,]  0.84966978  0.257836893 -0.05777543 -8.798317e-02  0.1194636896
## [16,]  0.46952059  0.551457124 -0.17240511 -6.189925e-01  0.0110120237
##               [,6]         [,7]         [,8]          [,9]
##  [1,]  0.037813186  0.020364838  0.013626588  0.0005664330
##  [2,]  0.022186452  0.015566902  0.003188625 -0.0005153534
##  [3,]  0.009343832  0.006574025 -0.001877054 -0.0042355236
##  [4,] -0.020655638  0.002853180 -0.022454676  0.0024547420
##  [5,] -0.039520801 -0.013789090 -0.023000535  0.0044743588
##  [6,] -0.011779547 -0.033121931  0.025369790 -0.0026718571
##  [7,] -0.010681129 -0.040704227 -0.007333541  0.0014426376
##  [8,] -0.023995771 -0.038242091 -0.011933256 -0.0009601786
##  [9,] -0.018160807 -0.019607921  0.010278321 -0.0019575291
## [10,]  0.023231572  0.018688238  0.046079272  0.0035492776
## [11,]  0.020605869  0.047874692 -0.017771805 -0.0036880204
## [12,]  0.052817010  0.025672909 -0.017650600  0.0015209655
## [13,] -0.110622081  0.061154541  0.002643078  0.0001967562
## [14,]  0.147192252 -0.008654892 -0.002414531  0.0014284114
## [15,]  0.004127371 -0.025042797 -0.007886915 -0.0026069071
## [16,] -0.081901771 -0.019586376  0.011137238  0.0010017874
```

Obtain least squares estimators for alpha

```
alpha_hat <- ginv(t(Z) %*% Z, tol = .Machine$double.eps) %*% t(Z) %*%  y

as.matrix(alpha_hat)
```

```
##              [,1]
##  [1,] -0.352198731
##  [2,]  0.004966905
##  [3,]  0.601915680
##  [4,] -0.234391618
##  [5,]  0.009315548
##  [6,]  0.216735789
##  [7,]  0.383372953
##  [8,] -0.520133632
##  [9,] -2.400619956
```

Define vector of ones and zeros to determine cutoff of principal components to use

```
b_p <- c(1,1,1,1,1,0,0,0,0)
```

Define alpha_pc as per p.315 (equation not numbered). Note that this is term-by-term multiplication, not the dot product

```
alpha_pc <- alpha_hat * b_p
```

Calculate beta_pc using Equation 9.10. Compare results to Table 9.12 on p. 318.Vary B vector to obtain standardized estimates in columns A, B, C, D, and E

```
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##              [,1]
##  [1,]  0.50548074
##  [2,]  0.21936825
##  [3,] -0.40984354
##  [4,] -0.11080761
##  [5,] -0.05905930
##  [6,]  0.13762307
##  [7,]  0.17349238
##  [8,] -0.05334645
##  [9,] -0.04626539
```

Calculate beta_pc using Equation 9.10. Compare results to Table 9.12 on p. 318.Vary B vector to obtain standardized estimates in columns A, B, C, D, and E

Column A

```
b_p <- c(1,0,0,0,0,0,0,0,0)
alpha_pc <- alpha_hat * b_p
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##               [,1]
##  [1,]  0.119286603
##  [2,]  0.046616509
##  [3,] -0.145677187
##  [4,] -0.077311759
##  [5,]  0.158197077
##  [6,]  0.089021205
##  [7,] -0.142843030
##  [8,]  0.009000843
##  [9,] -0.164336136
```

Column B

```
b_p <- c(1,1,0,0,0,0,0,0,0)
alpha_pc <- alpha_hat * b_p
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##              [,1]
##  [1,]  0.118761294
##  [2,]  0.044932650
##  [3,] -0.145190437
##  [4,] -0.079994165
##  [5,]  0.157767881
##  [6,]  0.091590072
##  [7,] -0.142471695
##  [8,]  0.006360273
##  [9,] -0.163853643
```

Column C

```
b_p <- c(1,1,1,0,0,0,0,0,0)
alpha_pc <- alpha_hat * b_p
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##             [,1]
##  [1,]  0.50964555
##  [2,]  0.04406003
##  [3,] -0.42765338
##  [4,] -0.02759474
##  [5,] -0.01555067
##  [6,]  0.05865775
##  [7,]  0.12338659
##  [8,] -0.12687999
##  [9,] -0.07769821
```

Column D

```
b_p <- c(1,1,1,1,0,0,0,0,0)
alpha_pc <- alpha_hat * b_p
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##             [,1]
##  [1,]  0.50680870
##  [2,]  0.21392899
##  [3,] -0.41001559
##  [4,] -0.11235708
##  [5,] -0.05993930
##  [6,]  0.13949587
##  [7,]  0.17483670
##  [8,] -0.04650496
##  [9,] -0.04658936
```

Column E

```
b_p <- c(1,1,1,1,1,0,0,0,0)
alpha_pc <- alpha_hat * b_p
beta_pc <- as.matrix(ev) %*% alpha_pc
as.matrix(beta_pc)
```

```
##             [,1]
##  [1,]  0.50548074
##  [2,]  0.21936825
##  [3,] -0.40984354
##  [4,] -0.11080761
##  [5,] -0.05905930
##  [6,]  0.13762307
##  [7,]  0.17349238
##  [8,] -0.05334645
##  [9,] -0.04626539
```