Reindeer Project Analysis

2024-03-22

```
knitr::opts_chunk$set(echo=TRUE)
#install.packages("caret")
#install.packages("vip")
#install.packages("pls")
#install.packages("ggpubr")
library(pls)
```

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

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

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

```
library(ggpubr)
library(knitr)

#load data
reindeer_data<- read.csv("reindeer_dataframe.csv")
(reindeer_data)
```

```
##    Year Reindeer_Total sa_temp_min sa_temp_max   prec lynx wolverine
## 1  2000         155907       -3.77        4.46 609.19  6.5        10
## 2  2001         188964       -5.20        3.66 622.38  6.5         2
## 3  2002         210992       -5.30        3.85 547.62  8.0         4
## 4  2003         243864       -4.69        4.44 570.93 10.0         3
## 5  2004         249378       -4.38        4.51 579.67  3.0         3
## 6  2005         251282       -3.97        4.87 669.02  3.5        11
```

```
## 7  2006         262226       -4.21        4.47 562.71  5.5         8
## 8  2007         277586       -4.08        4.66 648.26 12.0         7
## 9  2008         268639       -4.59        4.00 575.61  9.0         6
## 10 2009         276284       -4.75        4.09 513.54  9.0         7
## 11 2010         272523       -5.85        3.15 634.77 15.0        10
## 12 2011         274969       -3.40        5.08 636.50 11.5         4
## 13 2012         256095       -5.18        3.53 625.62 13.0         9
## 14 2013         234044       -4.34        5.38 561.49 10.0         1
## 15 2014         211227       -4.07        4.56 536.42  5.5         8
##    carcass_weight avg_price sami_pop higher_ed basic_school permit_to_build
## 1        23.45000  35.72815    59972      40.1         11.2             228
## 2        26.26250  39.15278    59664      40.6          9.2             250
## 3        28.62308  34.94238    59180      41.3          8.4             144
## 4        27.80000  26.47889    58897      42.9          9.5             270
## 5        26.51429  21.90753    58280      43.9         10.3             273
## 6        24.97857  28.09892    57867      44.9         10.5             483
## 7        24.65000  42.38590    57513      46.8         11.2             489
## 8        25.38462  50.10350    56915      47.5         11.8             358
## 9        23.45385  51.05887    56520      46.6         12.0             267
## 10       25.10000  52.47420    56142      46.0         12.2             186
## 11       23.40000  53.56327    55934      46.3         11.8             415
## 12       21.44444  51.57997    55635      45.8         11.2             273
## 13       24.00000  49.86143    55631      45.5         11.9             368
## 14       27.70000  52.84608    55652      45.6         11.6             299
## 15       21.70000  57.11831    55619      42.9         12.6             353
##    underconstruction radio computer internet greenhouse_gases_emitted
## 1               2571    57       25       27                    55003
## 2               2717    56       33       35                    56248
## 3               3924    58       31       35                    54935
## 4               3221    58       36       42                    55543
## 5               3725    58       36       44                    55908
## 6               3181    55       47       55                    54787
## 7               4112    54       51       60                    54760
## 8               5595    53       56       66                    56450
## 9               5610    54       59       71                    54927
## 10              4017    53       65       73                    52436
## 11               415    56       68       77                    54732
## 12              6314    55       70       80                    53798
## 13              6107    60       70       80                    53166
## 14              5638    59       75       85                    53407
## 15              6972    64       75       88                    53806
##    ag_area_decares_finnmark
## 1                     169.5
## 2                     184.1
## 3                     202.0
## 4                     218.0
## 5                     228.1
## 6                     232.1
## 7                     244.9
## 8                     251.0
## 9                     251.1
## 10                    264.0
## 11                    255.6
## 12                    270.6
```

```
## 13                    281.0
## 14                    285.1
## 15                    286.9
```

```
# I found the plots easier to read when the variable names were clear:
sami_pop<- reindeer_data$sami_pop
sa_temp_min<- reindeer_data$sa_temp_min
sa_temp_max<- reindeer_data$sa_temp_max
prec <- reindeer_data$prec
lynx<-reindeer_data$lynx
wolverine <- reindeer_data$wolverine
carcass_weight<-reindeer_data$carcass_weight
avg_price_meat<-reindeer_data$avg_price
radio <- reindeer_data$radio
higher_ed <-reindeer_data$higher_ed
basic_school <-reindeer_data$basic_school
permit_to_build <- reindeer_data$permit_to_build
underconstruction <- reindeer_data$underconstruction
computer <- reindeer_data$computer
internet <- reindeer_data$internet
greenhouse_gases_emitted <- reindeer_data$greenhouse_gases_emitted
ag_area_decares_finnmark <- reindeer_data$ag_area_decares_finnmark


# Split the data into predictor variables [X] and response variable [Y]:
total_reindeer <- (reindeer_data$Reindeer_Total)
predictors <- data.frame(sami_pop,
                         sa_temp_min,
                         sa_temp_max,
                         prec,
                         lynx,
                         wolverine,
                         carcass_weight,
                         avg_price_meat,
                         radio,
                         higher_ed,
                         basic_school,
                         permit_to_build,
                         underconstruction,
                         computer,
                         internet,
                         greenhouse_gases_emitted,
                         ag_area_decares_finnmark)

#Mean-center and scale the predictors dataframe:
predictors_scaled <- as.data.frame(scale(predictors))
reindeer<-cbind(total_reindeer,predictors_scaled)
summary(reindeer)
```

```
##  total_reindeer      sami_pop        sa_temp_min       sa_temp_max
##  Min.   :155907   Min.   :-1.0568   Min.   :-2.0293   Min.   :-1.9335
##  1st Qu.:222636   1st Qu.:-0.9470   1st Qu.:-0.6803   1st Qu.:-0.6462
##  Median :251282   Median :-0.2395   Median : 0.2114   Median : 0.2425
##  Mean   :242265   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:270581   3rd Qu.: 0.8159   3rd Qu.: 0.6763   3rd Qu.: 0.4917
```

```
##  Max.   :277586   Max.   : 1.6884   Max.   : 1.7052   Max.   : 1.7707
##       prec              lynx           wolverine       carcass_weight
##  Min.   :-1.7407   Min.   :-1.5868   Min.   :-1.6305   Min.   :-1.642443
##  1st Qu.:-0.6758   1st Qu.:-0.7265   1st Qu.:-0.8466   1st Qu.:-0.705653
##  Median :-0.2905   Median : 0.1338   Median : 0.2508   Median : 0.006758
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000
##  3rd Qu.: 0.8175   3rd Qu.: 0.6357   3rd Qu.: 0.7212   3rd Qu.: 0.664651
##  Max.   : 1.6690   Max.   : 1.8545   Max.   : 1.5050   Max.   : 1.707467
##  avg_price_meat        radio           higher_ed        basic_school
##  Min.   :-1.8705   Min.   :-1.2354   Min.   :-1.8357   Min.   :-2.1681
##  1st Qu.:-0.6883   1st Qu.:-0.7300   1st Qu.:-0.6532   1st Qu.:-0.5173
##  Median : 0.5906   Median :-0.2246   Median : 0.4448   Median : 0.1431
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.7812   3rd Qu.: 0.4492   3rd Qu.: 0.7193   3rd Qu.: 0.6796
##  Max.   : 1.2295   Max.   : 2.4707   Max.   : 1.2895   Max.   : 1.2986
##  permit_to_build   underconstruction    computer          internet
##  Min.   :-1.6636   Min.   :-2.1965   Min.   :-1.6155   Min.   :-1.6881
##  1st Qu.:-0.5189   1st Qu.:-0.6110   1st Qu.:-0.9839   1st Qu.:-0.8983
##  Median :-0.3739   Median :-0.1466   Median : 0.1646   Median : 0.2369
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5259   3rd Qu.: 0.7679   3rd Qu.: 0.9111   3rd Qu.: 0.8539
##  Max.   : 1.7855   Max.   : 1.5351   Max.   : 1.2557   Max.   : 1.3228
##  greenhouse_gases_emitted ag_area_decares_finnmark
##  Min.   :-1.9309          Min.   :-2.0037
##  1st Qu.:-0.7451          1st Qu.:-0.5155
##  Median : 0.1099          Median : 0.2612
##  Mean   : 0.0000          Mean   : 0.0000
##  3rd Qu.: 0.5318          3rd Qu.: 0.7142
##  Max.   : 1.5535          Max.   : 1.2589
```

PLS MODEL 1:

```
###MODEL 1 ###
# Split the data into a training set and test set:
#training data: the rows 1-11:
training_data <- reindeer[1:11, c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17)]
(training_data)
```

```
##    total_reindeer   sami_pop sa_temp_min sa_temp_max       prec       lynx
## 1          155907  1.6883695   1.1411778   0.2425224  0.3569006 -0.5831097
## 2          188964  1.4941349  -1.0385430  -1.0863676  0.6461551 -0.5831097
## 3          210992  1.1889091  -1.1909710  -0.7707562 -0.9933189 -0.1529468
## 4          243864  1.0104403  -0.2611600   0.2093002 -0.4821346  0.4206037
## 5          249378  0.6213405   0.2113669   0.3255781 -0.2904679 -1.5868232
## 6          251282  0.3608896   0.8363218   0.9235786  1.6689623 -1.4434356
## 7          262226  0.1376459   0.4704945   0.2591336 -0.6623978 -0.8698850
## 8          277586 -0.2394719   0.6686509   0.5747449  1.2136990  0.9941543
## 9          268639 -0.4885715  -0.1087320  -0.5215893 -0.3795030  0.1338285
## 10         276284 -0.7269503  -0.3526168  -0.3720892 -1.7406874  0.1338285
## 11         272523 -0.8581217  -2.0293251  -1.9335350  0.9178656  1.8544802
##      wolverine carcass_weight avg_price_meat      radio  higher_ed basic_school
## 1   1.19149609   -0.706550013    -0.65371114  0.1123059 -1.8356628    0.1430696
## 2  -1.31691673    0.605903530    -0.35220824 -0.2246118 -1.6245053   -1.5077330
## 3  -0.68981352    1.707467100    -0.72288976  0.4492236 -1.3288847   -2.1680541
## 4  -1.00336512    1.323378133    -1.46801125  0.4492236 -0.6531806   -1.2601127
```

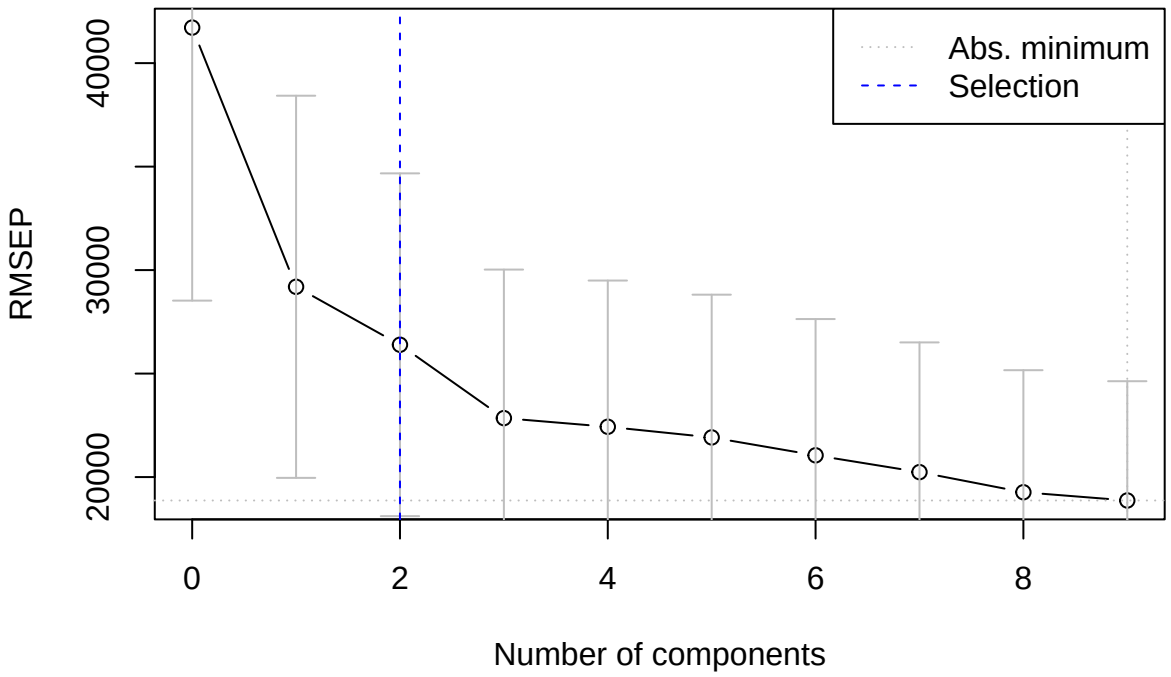
```
## 5  -1.00336512    0.723399369    -1.87047220  0.4492236 -0.2308656   -0.5997916
## 6   1.50504769    0.006758072    -1.32538435 -0.5615294  0.1914495   -0.4347114
## 7   0.56439288   -0.146569835    -0.06756547 -0.8984471  0.9938481    0.1430696
## 8   0.25084128    0.196238541     0.61188827 -1.2353648  1.2894686    0.6383103
## 9  -0.06271032   -0.704755207     0.69599879 -0.8984471  0.9093851    0.8033906
## 10  0.25084128    0.063422732     0.82060373 -1.2353648  0.6559961    0.9684709
## 11  1.19149609   -0.729882521     0.91648491 -0.2246118  0.7826906    0.6383103
##    permit_to_build underconstruction   computer    internet
## 1       -0.8237835       -0.96951279 -1.6155070 -1.68807678
## 2       -0.6038413       -0.88642470 -1.1561211 -1.29320502
## 3       -1.6635628       -0.19952523 -1.2709676 -1.29320502
## 4       -0.4038938       -0.59960006 -0.9838514 -0.94769223
## 5       -0.3739017       -0.31277543 -0.9838514 -0.84897429
## 6        1.7255465       -0.62236392 -0.3521958 -0.30602562
## 7        1.7855307       -0.09253509 -0.1225029 -0.05923076
## 8        0.4758749        0.75143501  0.1646133  0.23692306
## 9       -0.4338860        0.75997146  0.3368830  0.48371791
## 10      -1.2436731       -0.14659926  0.6814224  0.58243585
## 11       1.0457252       -2.19648482  0.8536921  0.77987173
##    greenhouse_gases_emitted
## 1                0.29739555
## 2                1.37812368
## 3                0.23836783
## 4                0.76614510
## 5                1.08298507
## 6                0.10989573
## 7                0.08645825
## 8                1.55347073
## 9                0.23142339
## 10              -1.93090093
## 11               0.06215272
```

```r
# testing data: the rows 12-15:
y_test <- reindeer[12:nrow(reindeer), c("total_rein")]
test <- reindeer[12:nrow(reindeer), c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17)]


# This code runs the partial least squsare regression:
pls_model<- plsr(training_data$total_reindeer ~ ., data= training_data,validation = "LOO")
# this is to veiw the summary, which is important in determining what to do next:
summary(pls_model)
```

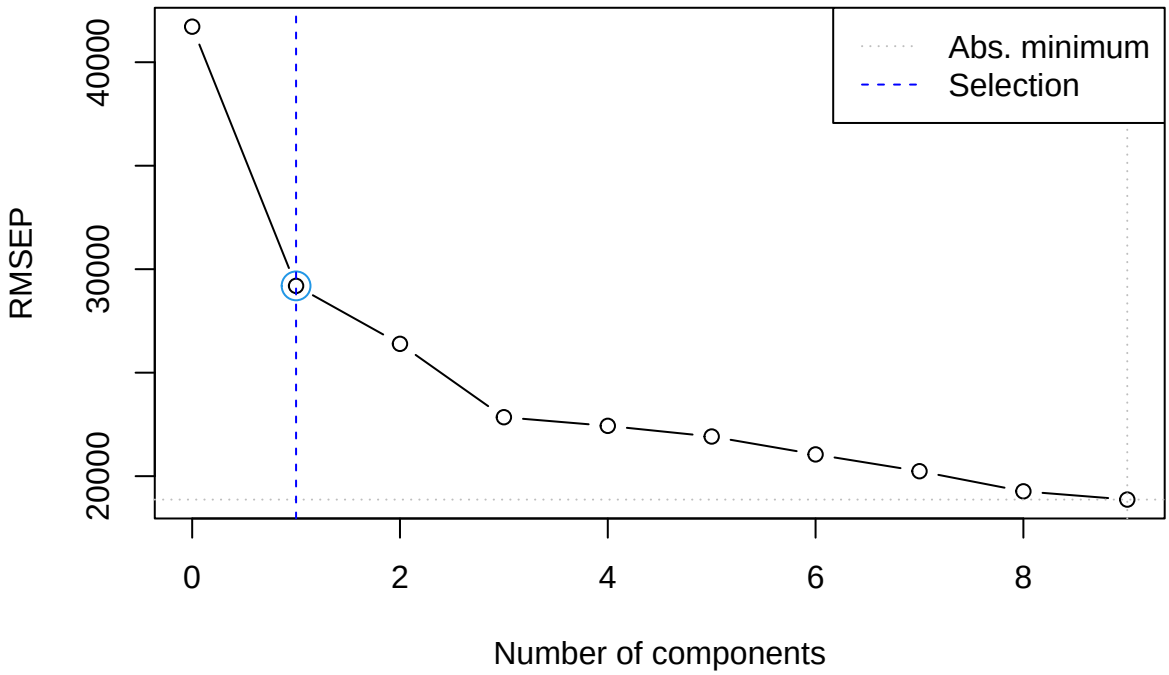
```
## Data:    X dimension: 11 16
##  Y dimension: 11 1
## Fit method: kernelpls
## Number of components considered: 9
##
## VALIDATION: RMSEP
## Cross-validated using 11 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           41717    29193    26393    22849    22431    21915    21052
## adjCV        41717    28847    25231    21858    21455    20927    20073
##        7 comps  8 comps  9 comps
## CV       20240    19267    18869
## adjCV    19297    18371    17991
```

```
##
## TRAINING: % variance explained
##                                1 comps  2 comps  3 comps  4 comps  5 comps
## X                                42.79    53.16    65.39    77.93    88.97
## training_data$total_reindeer     71.70    96.11    98.34    99.21    99.72
##                                6 comps  7 comps  8 comps  9 comps
## X                                94.97    97.13       99    99.43
## training_data$total_reindeer     99.95    99.99      100   100.00
```

```r
# This is the mathematical way to determine the number of components to use:
ncomp.onesigma <- selectNcomp(pls_model, method = "onesigma", plot = TRUE)
```




```r
ncomp.permut <- selectNcomp(pls_model, method = "randomization", plot = TRUE)
```



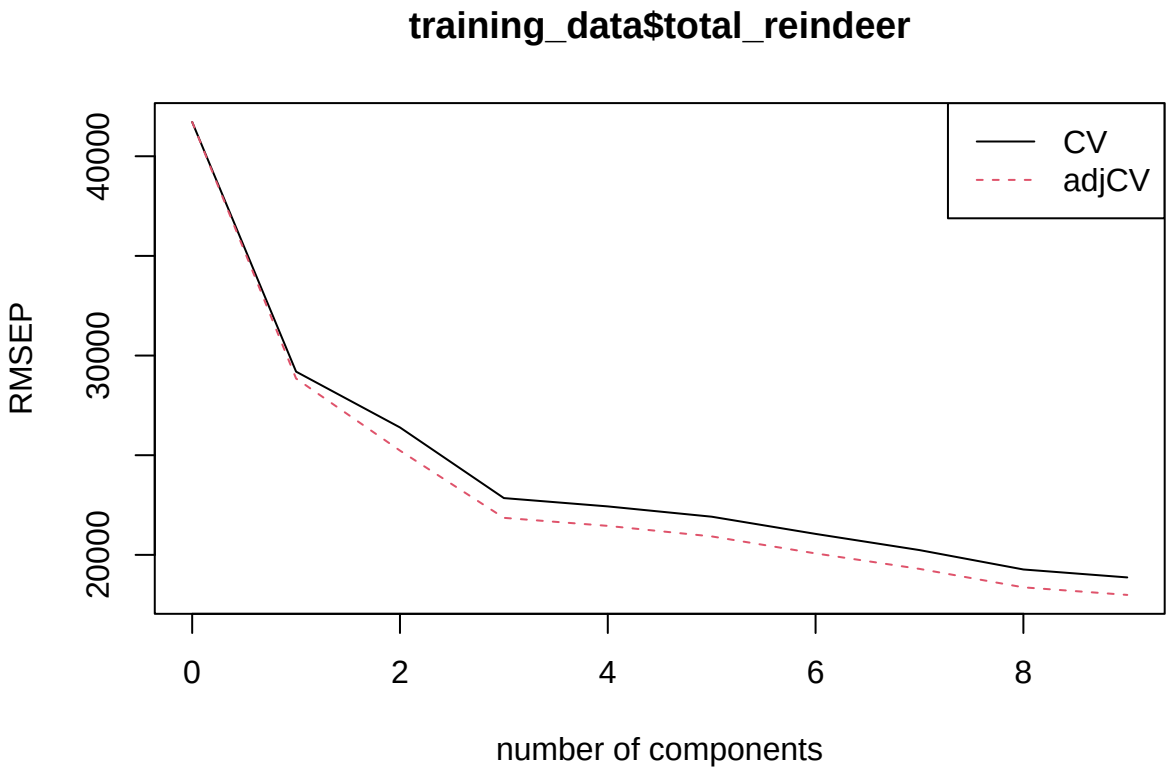

```
(ncomp.onesigma)
```

```
## [1] 2
```

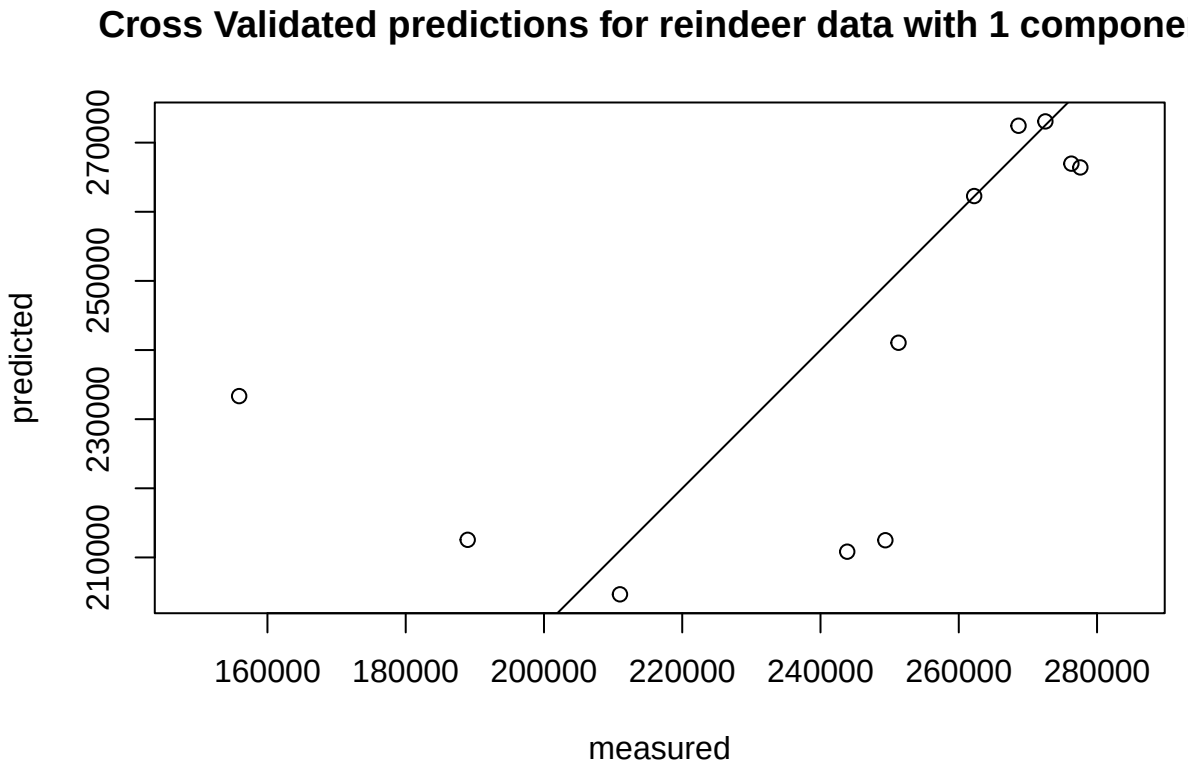
```
(ncomp.permut)
```

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

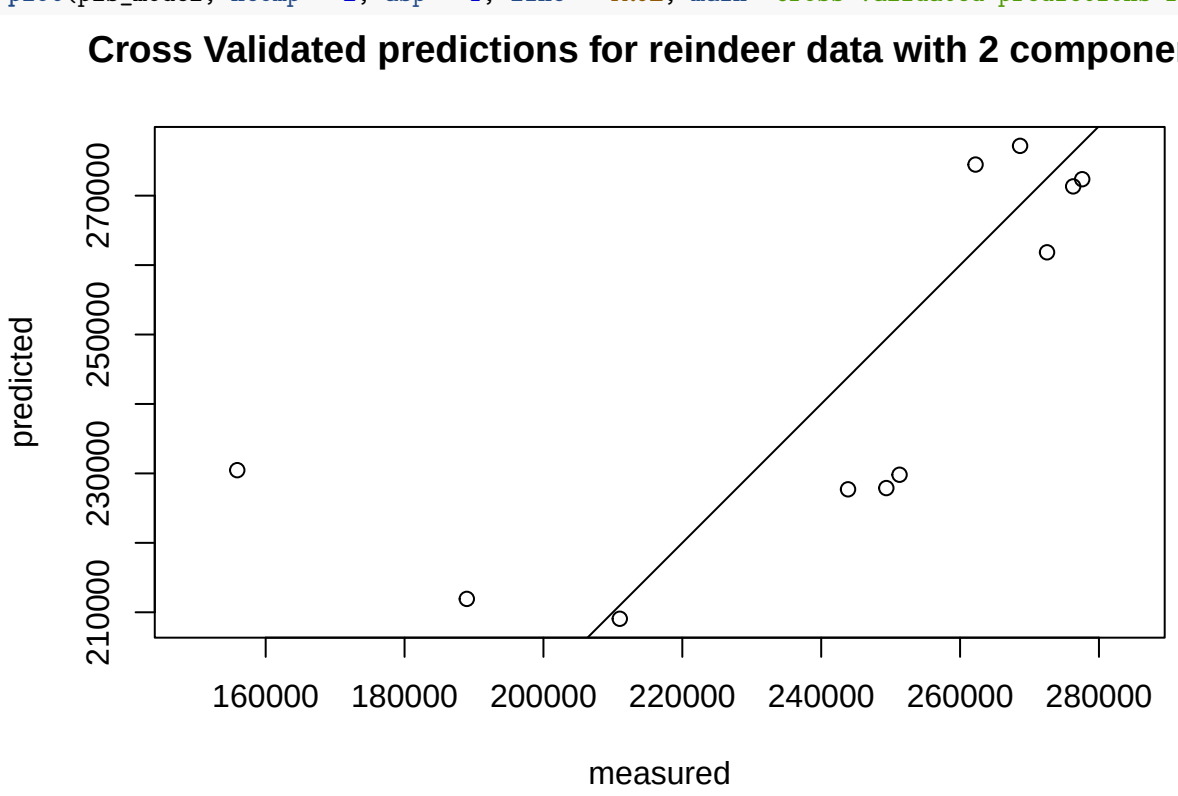
```
#RMSEP over Components: this is the visual way to determine the number of components: 3
rme<- plot(RMSEP(pls_model), legendpos = "topright")
```



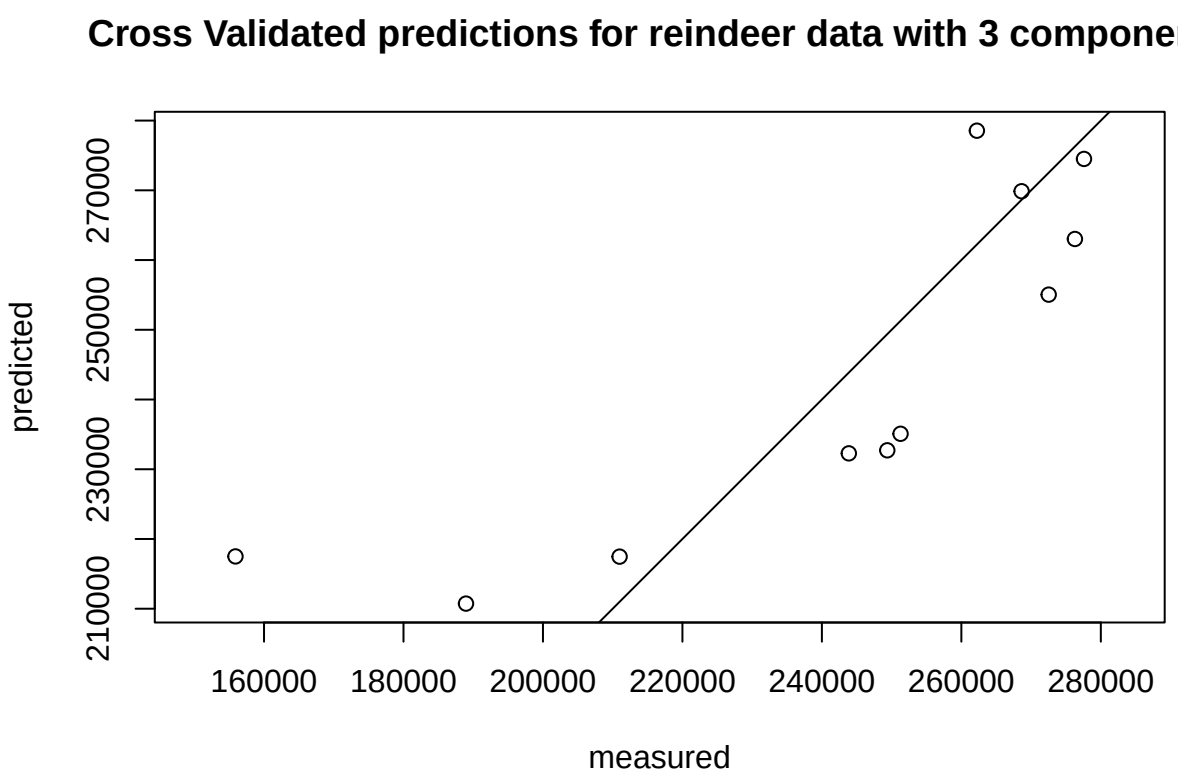

```
# these plots let us examine how well our model fits the data based on the number of
# components we selected:
plot(pls_model, ncomp = 1, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data wi
```



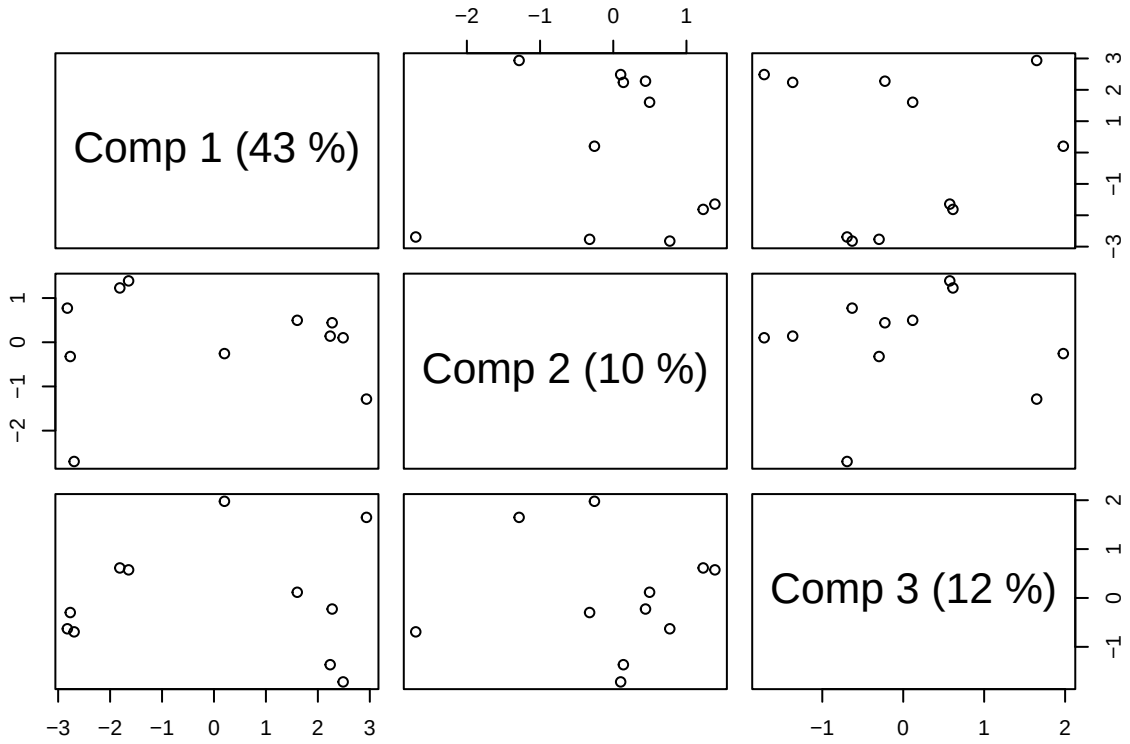
```
plot(pls_model, ncomp = 2, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data wi
```




```
plot(pls_model, ncomp = 3, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data wi
```



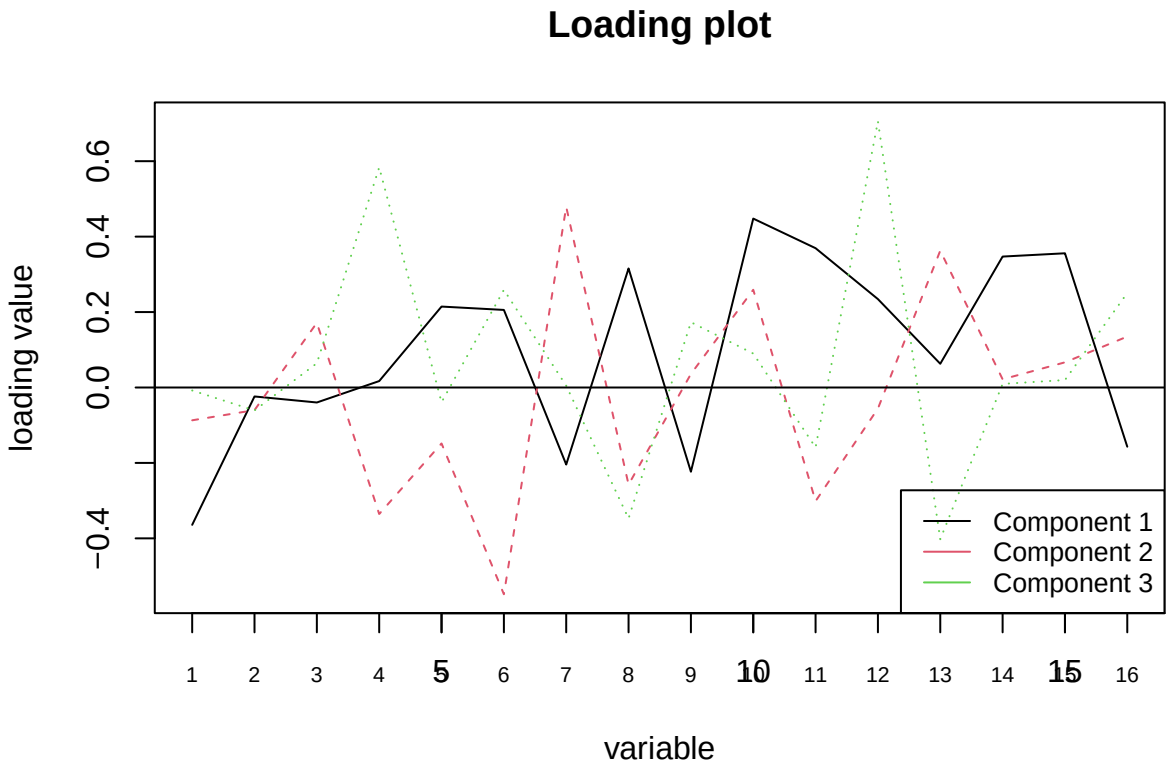
```r
# this is a way to visually inspect the data for any outliers or oddities:
plot(pls_model, plottype = "scores", comps = 1:3)
```




```r
explvar(pls_model)
```

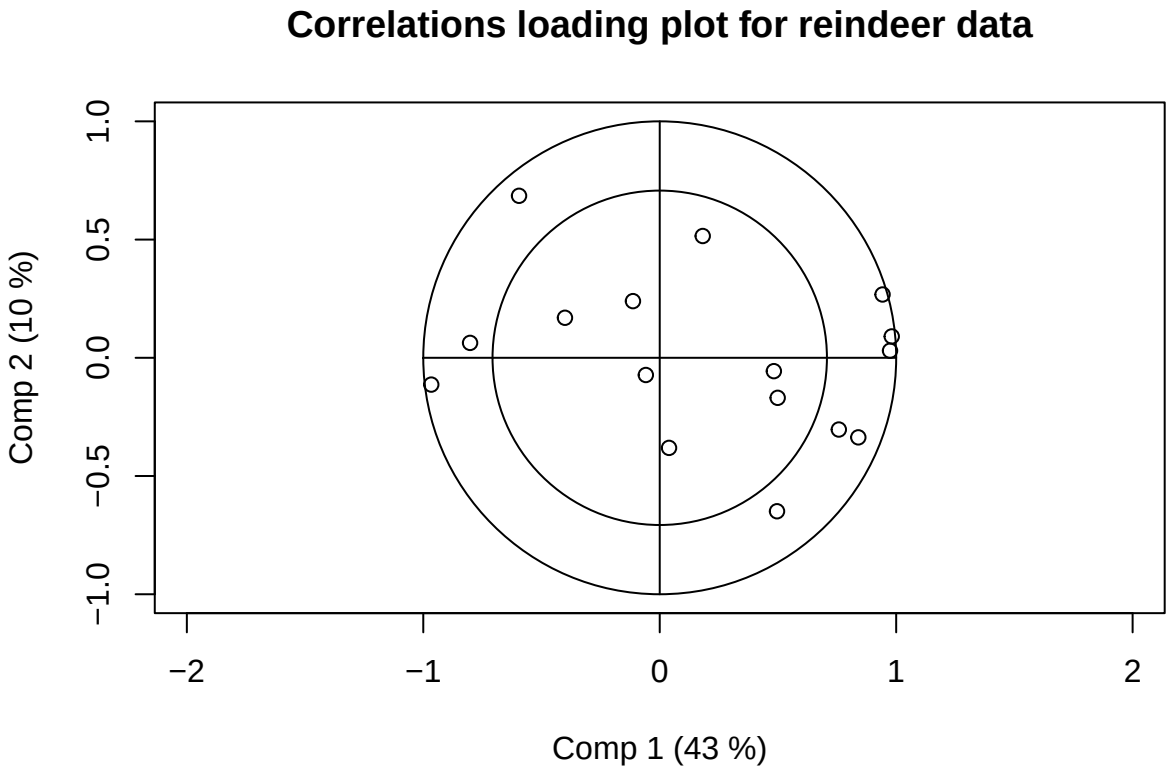
```
##     Comp 1     Comp 2     Comp 3     Comp 4     Comp 5     Comp 6     Comp 7
## 42.7890099 10.3757255 12.2262262 12.5439794 11.0332729  6.0047502  2.1600274
##     Comp 8     Comp 9
##  1.8684789  0.4290587
```

```r
# these plots show variable loading value on component: in simple terms, how much the
# variable has an effect on the component, with points near zero having little effect
# plot 1
plot(pls_model, "loadings", comps = 1:3,main="Loading plot")
legend("bottomright", legend = paste("Component", 1:3), col = 1:3, lty = 1, cex = 0.8)
abline(h = 0)
axis(side = 1, at = 1:17, labels = colnames(pls_model$Xvar), cex.axis = 0.7)
```

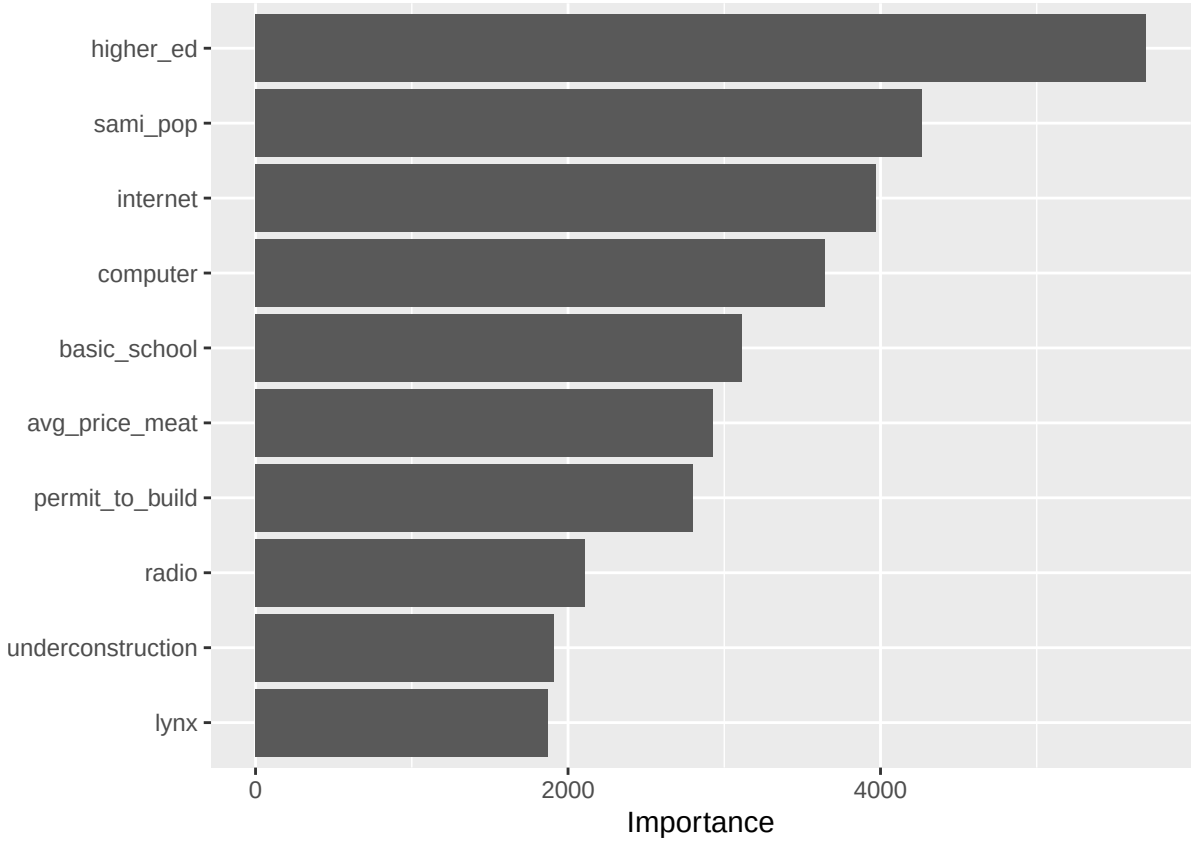



```
# plot 2
plot(pls_model, plottype = "correlation", ncomp=1:3, legendpos = "bottomleft",main="Correlations loadin
```

```
## Warning in plot.window(...): "ncomp" is not a graphical parameter

## Warning in plot.window(...): "legendpos" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "ncomp" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "legendpos" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "ncomp" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "legendpos" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "ncomp" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "legendpos" is not
## a graphical parameter

## Warning in box(...): "ncomp" is not a graphical parameter

## Warning in box(...): "legendpos" is not a graphical parameter

## Warning in title(...): "ncomp" is not a graphical parameter

## Warning in title(...): "legendpos" is not a graphical parameter
```



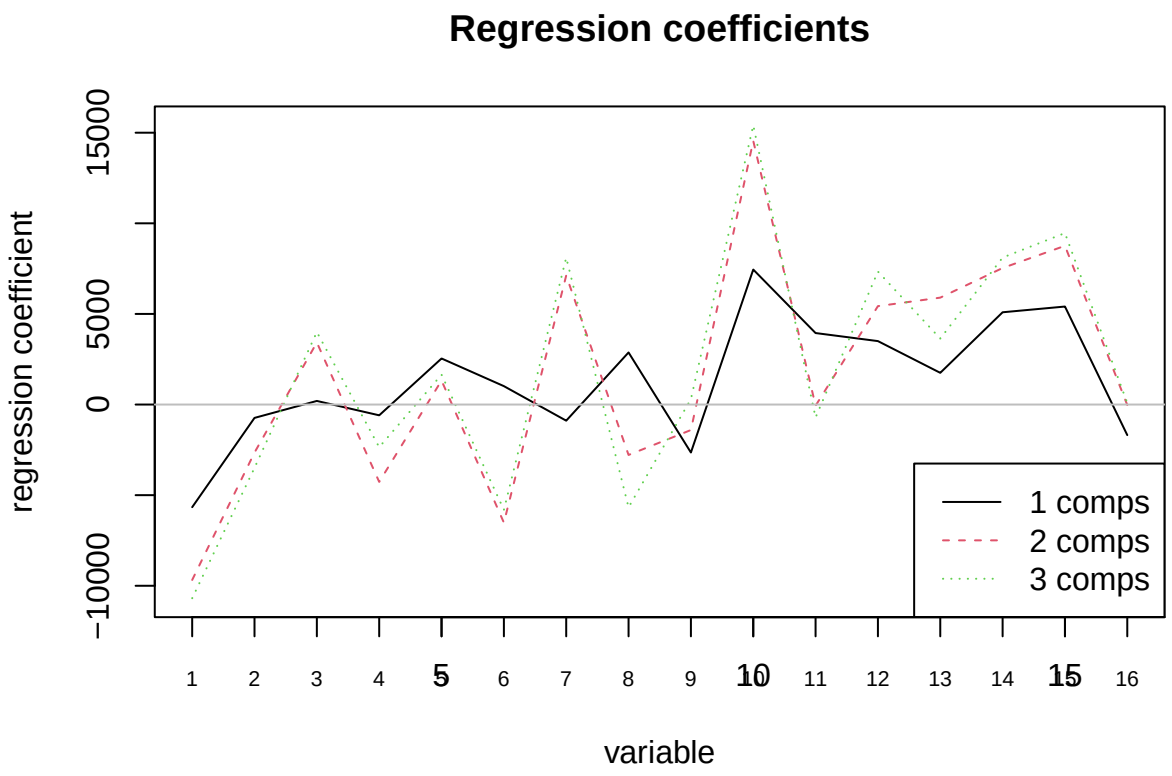

```
# the vi and vip function shows the "variable importance in projection" and tells us
# which variables had the strongest effect on the model
vip(pls_model)
```




```
vi(pls_model)
```

```
## # A tibble: 16 x 2
##    Variable                 Importance
##    <chr>                         <dbl>
##  1 higher_ed                     5701.
##  2 sami_pop                      4267.
##  3 internet                      3971.
##  4 computer                      3645.
##  5 basic_school                  3113.
##  6 avg_price_meat                2927.
##  7 permit_to_build               2802.
##  8 radio                         2111.
##  9 underconstruction             1908.
## 10 lynx                          1873.
## 11 wolverine                     1630.
## 12 carcass_weight                1625.
## 13 greenhouse_gases_emitted      1351.
## 14 prec                          1108.
## 15 sa_temp_min                    834.
## 16 sa_temp_max                    606.
```

```
# the regression coefficient plot looks at how strongly each variable related to the
# outcome variable. This plot shows 3 components:
plot(pls_model, plottype = "coef", ncomp=1:3, legendpos = "bottomright",main="Regression coefficients")
axis(side = 1, at = 1:17, labels = colnames(pls_model$Xvar), cex.axis = 0.7)
```




```
# This is the predict feature of the model for each of the number of the components
# suggested above.
prediction_1 = predict(pls_model, comps=1, newdata=test)
prediction_2 = predict(pls_model, comps=2, newdata=test)
prediction_3 = predict(pls_model, comps=3, newdata=test)
#this is the actual number of reindeer
actual = (reindeer$total_reindeer[12:15])
# print to console for a visual comparison
(prediction_1)
```

```
##    training_data$total_reindeer
## 12                     279017.0
## 13                     282770.6
## 14                     277472.7
## 15                     273606.9
```

```
(prediction_2)
```

```
##    training_data$total_reindeer
## 12                     243523.9
## 13                     238672.3
## 14                     283851.5
## 15                     237195.9
```

```
(prediction_3)
```

```
##    training_data$total_reindeer
## 12                     236388.4
## 13                     244008.4
## 14                     240146.4
## 15                     236093.9
```

```
(actual)
```

```
## [1] 274969 256095 234044 211227
```

```
# Outcomes for the 3 component choices:

# From the permutation method: 1 component:
print("RMSE for prediction::actual with ncomp = 1")
```

```
## [1] "RMSE for prediction::actual with ncomp = 1"
```

```
sqrt(mean((prediction_1 - actual)^2))
```

```
## [1] 40327.65
```

```
# Calculate mean absolute error
mae1 <- mean(abs(prediction_1 - actual))
# Calculate R-squared
r_squared1 <- 1 - sum((actual - prediction_1)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae1))
```

```
## [1] "Mean Absolute Error:  34133.0370657873"
```

```
print(paste("R-squared: ", r_squared1))
```

```
## [1] "R-squared:  -1.85503136964292"
```

```
# From the one-sigma method: 2 components:
print("RMSE for prediction::actual with ncomp = 2")
```

```
## [1] "RMSE for prediction::actual with ncomp = 2"
```

```
sqrt(mean((prediction_2 - actual)^2))
```

```
## [1] 33344.85
```

```
# Calculate mean absolute error
mae2 <- mean(abs(prediction_2 - actual))
# Calculate R-squared
r_squared2 <- 1 - sum((actual - prediction_2)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae2))
```

```
## [1] "Mean Absolute Error:  31161.0648867631"
```

```
print(paste("R-squared: ", r_squared2))
```

```
## [1] "R-squared:  -0.95192252993155"
```

```
# From the visual method: 3 components:
print("RMSE for prediction::actual with ncomp = 3")
```

```
## [1] "RMSE for prediction::actual with ncomp = 3"
```

```
sqrt(mean((prediction_3 - actual)^2))
```

```
## [1] 23927.75
```

```
# Calculate mean absolute error
mae3 <- mean(abs(prediction_3 - actual))
# Calculate R-squared
```

```
r_squared3 <- 1 - sum((actual - prediction_3)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae3))
```

```
## [1] "Mean Absolute Error:  20409.1208267018"
```

```
print(paste("R-squared: ", r_squared3))
```

```
## [1] "R-squared:  -0.0050982776057007"
```

PLS MODEL 2:

```
############ MODEL 2 ##############
# Build a new data frame based on the top 9 variables:
predictors2 <- data.frame(sami_pop,
                          avg_price_meat,
                          computer,
                          higher_ed,
                          underconstruction,
                          radio,
                          basic_school,
                          permit_to_build,
                          internet)

predictors_scaled2 <- as.data.frame(scale(predictors2))
reindeer2<-cbind(total_reindeer,predictors_scaled2)
summary(reindeer2)
```

```
##  total_reindeer      sami_pop       avg_price_meat       computer
##  Min.   :155907   Min.   :-1.0568   Min.   :-1.8705   Min.   :-1.6155
##  1st Qu.:222636   1st Qu.:-0.9470   1st Qu.:-0.6883   1st Qu.:-0.9839
##  Median :251282   Median :-0.2395   Median : 0.5906   Median : 0.1646
##  Mean   :242265   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:270581   3rd Qu.: 0.8159   3rd Qu.: 0.7812   3rd Qu.: 0.9111
##  Max.   :277586   Max.   : 1.6884   Max.   : 1.2295   Max.   : 1.2557
##    higher_ed       underconstruction     radio          basic_school
##  Min.   :-1.8357   Min.   :-2.1965   Min.   :-1.2354   Min.   :-2.1681
##  1st Qu.:-0.6532   1st Qu.:-0.6110   1st Qu.:-0.7300   1st Qu.:-0.5173
##  Median : 0.4448   Median :-0.1466   Median :-0.2246   Median : 0.1431
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.7193   3rd Qu.: 0.7679   3rd Qu.: 0.4492   3rd Qu.: 0.6796
##  Max.   : 1.2895   Max.   : 1.5351   Max.   : 2.4707   Max.   : 1.2986
##  permit_to_build      internet
##  Min.   :-1.6636   Min.   :-1.6881
##  1st Qu.:-0.5189   1st Qu.:-0.8983
##  Median :-0.3739   Median : 0.2369
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5259   3rd Qu.: 0.8539
##  Max.   : 1.7855   Max.   : 1.3228
```

```
#split data:
#training data:
training_data2 <- reindeer2[1:11, c(1,2,3,4,5,6,7,8,9)]
(training_data2)
```

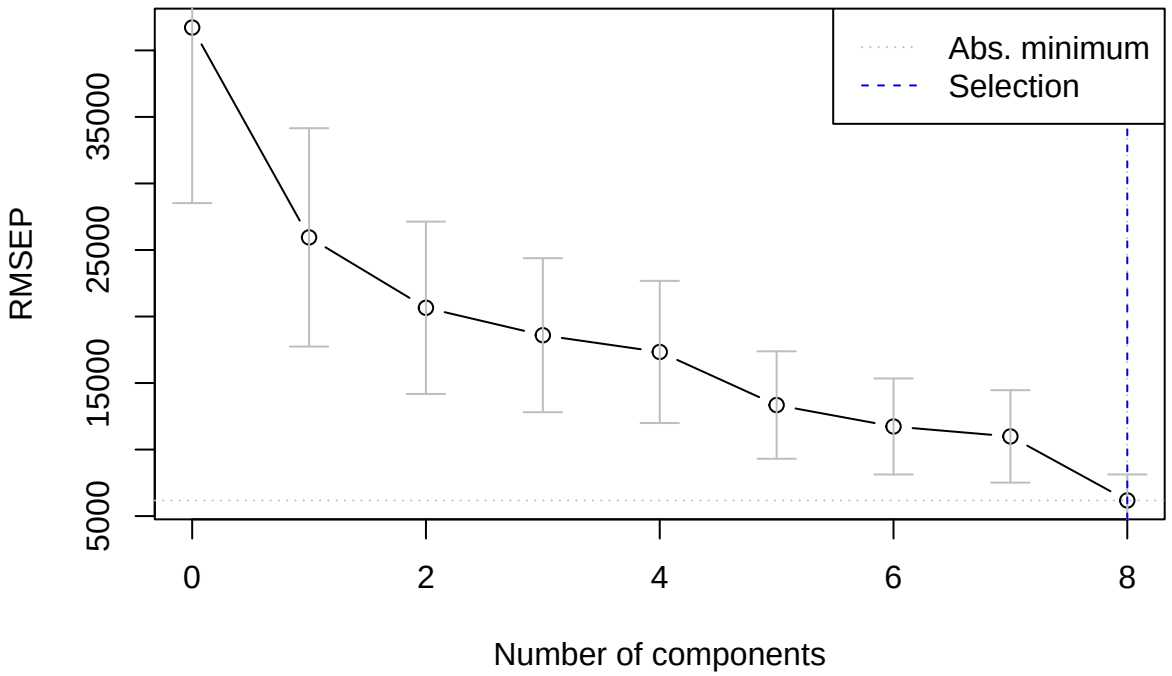
```
##    total_reindeer   sami_pop avg_price_meat   computer  higher_ed
```

```
## 1          155907  1.6883695    -0.65371114 -1.6155070 -1.8356628
## 2          188964  1.4941349    -0.35220824 -1.1561211 -1.6245053
## 3          210992  1.1889091    -0.72288976 -1.2709676 -1.3288847
## 4          243864  1.0104403    -1.46801125 -0.9838514 -0.6531806
## 5          249378  0.6213405    -1.87047220 -0.9838514 -0.2308656
## 6          251282  0.3608896    -1.32538435 -0.3521958  0.1914495
## 7          262226  0.1376459    -0.06756547 -0.1225029  0.9938481
## 8          277586 -0.2394719     0.61188827  0.1646133  1.2894686
## 9          268639 -0.4885715     0.69599879  0.3368830  0.9093851
## 10         276284 -0.7269503     0.82060373  0.6814224  0.6559961
## 11         272523 -0.8581217     0.91648491  0.8536921  0.7826906
##    underconstruction      radio basic_school permit_to_build
## 1        -0.96951279  0.1123059    0.1430696      -0.8237835
## 2        -0.88642470 -0.2246118   -1.5077330      -0.6038413
## 3        -0.19952523  0.4492236   -2.1680541      -1.6635628
## 4        -0.59960006  0.4492236   -1.2601127      -0.4038938
## 5        -0.31277543  0.4492236   -0.5997916      -0.3739017
## 6        -0.62236392 -0.5615294   -0.4347114       1.7255465
## 7        -0.09253509 -0.8984471    0.1430696       1.7855307
## 8         0.75143501 -1.2353648    0.6383103       0.4758749
## 9         0.75997146 -0.8984471    0.8033906      -0.4338860
## 10       -0.14659926 -1.2353648    0.9684709      -1.2436731
## 11       -2.19648482 -0.2246118    0.6383103       1.0457252
```

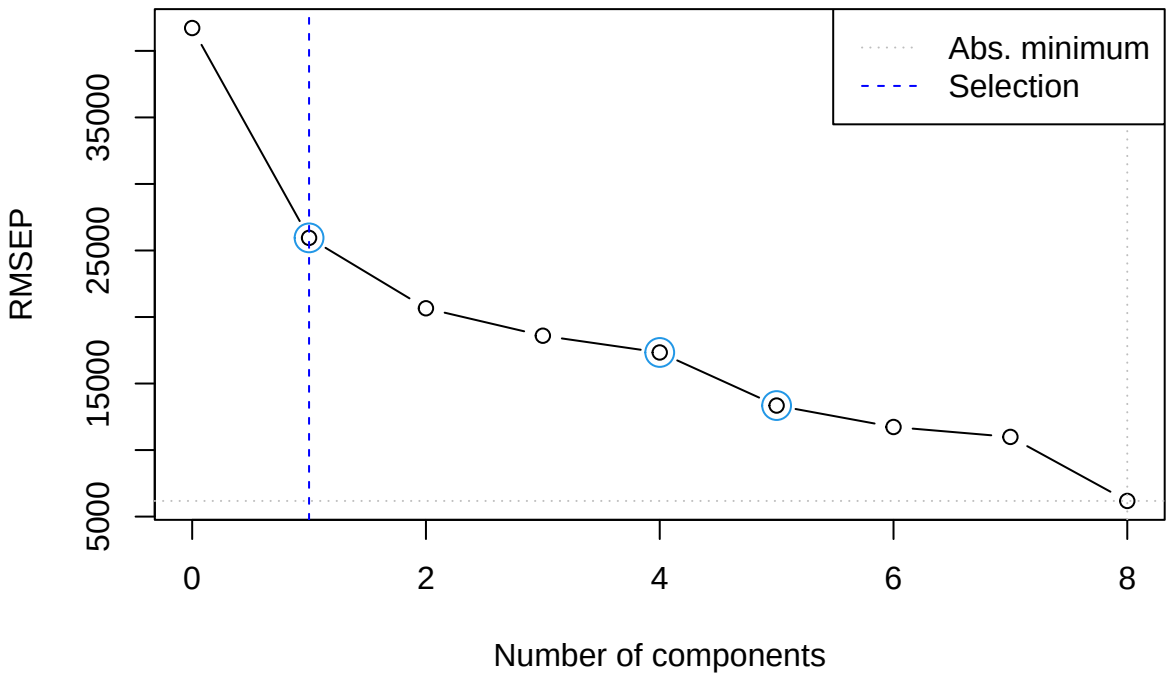
```
# testing data:
y_test2 <- reindeer2[12:nrow(reindeer2), c("total_rein")]
test2 <- reindeer2[12:nrow(reindeer2), c(1,2,3,4,5,6,7,8,9)]

# This code runs the partial least squares regression
pls_model2<- plsr(total_reindeer ~ ., data= training_data2,validation = "LOO")
# this is to veiw the summary, which is important in determining what to do next:
summary(pls_model2)
```

```
## Data:    X dimension: 11 8
##  Y dimension: 11 1
## Fit method: kernelpls
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 11 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           41717    25947    20656    18595    17336    13346    11735
## adjCV        41717    25682    19926    17821    16594    12819    11408
##        7 comps  8 comps
## CV       10989     6178
## adjCV    10584     5907
##
## TRAINING: % variance explained
##                  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X                  62.10    70.92    83.31    95.15    97.59    99.71    99.97
## total_reindeer     73.32    94.72    98.06    98.59    98.82    99.08    99.67
##                  8 comps
## X                 100.00
## total_reindeer     99.97
```

```r
# This is the mathematical way to determine the number of components to use:
ncomp.onesigma <- selectNcomp(pls_model2, method = "onesigma", plot = TRUE)
```




```r
ncomp.permut <- selectNcomp(pls_model2, method = "randomization", plot = TRUE)
```



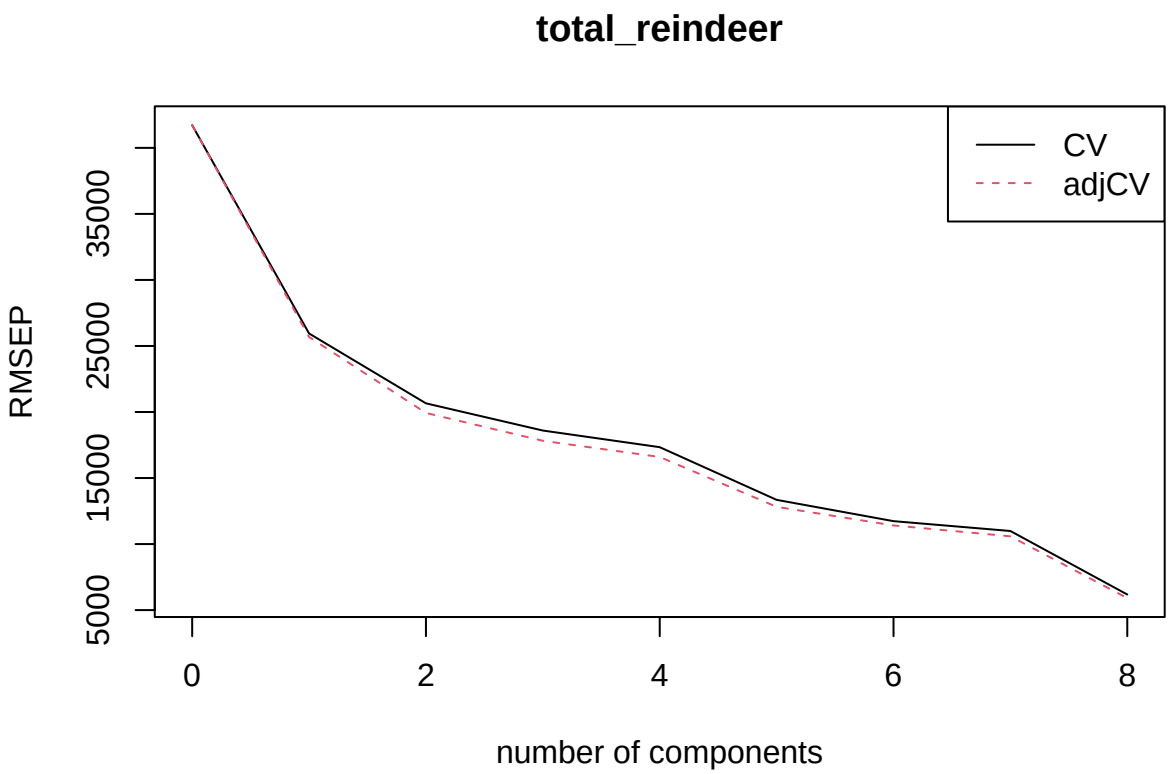

```r
(ncomp.onesigma)
```

```
## [1] 8
```

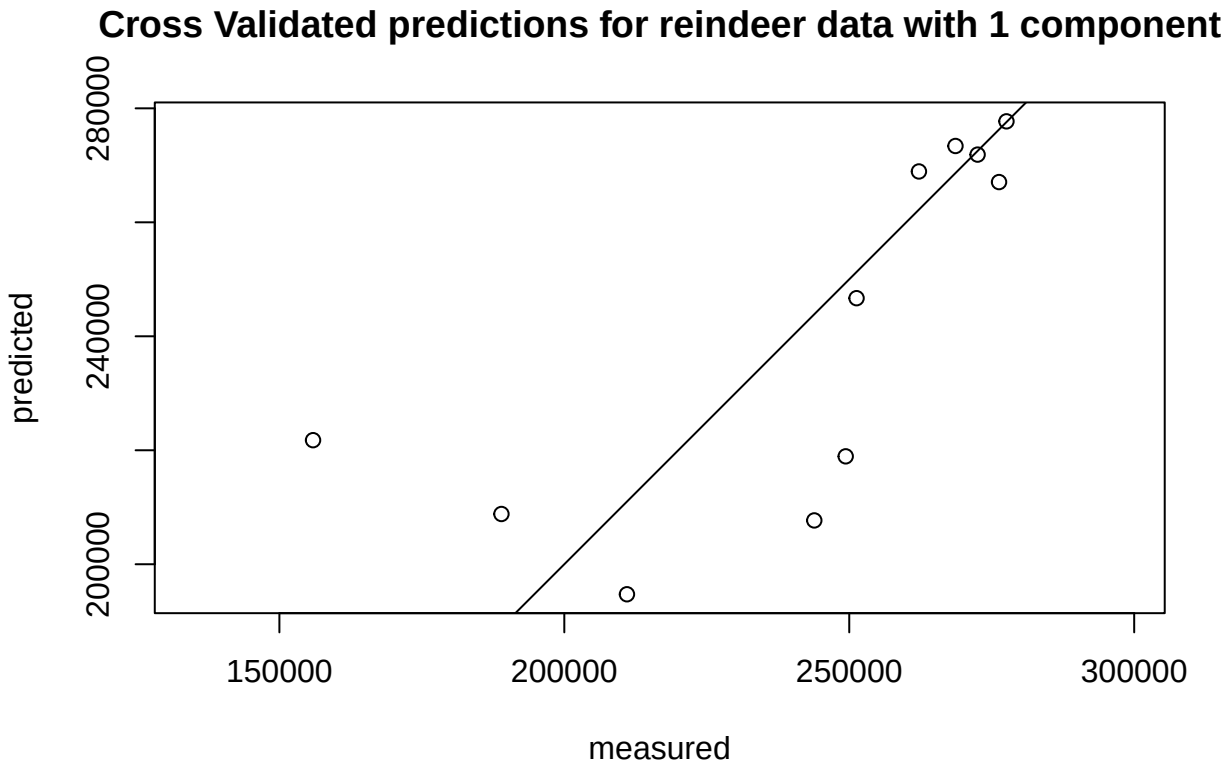
```r
(ncomp.permut)
```

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

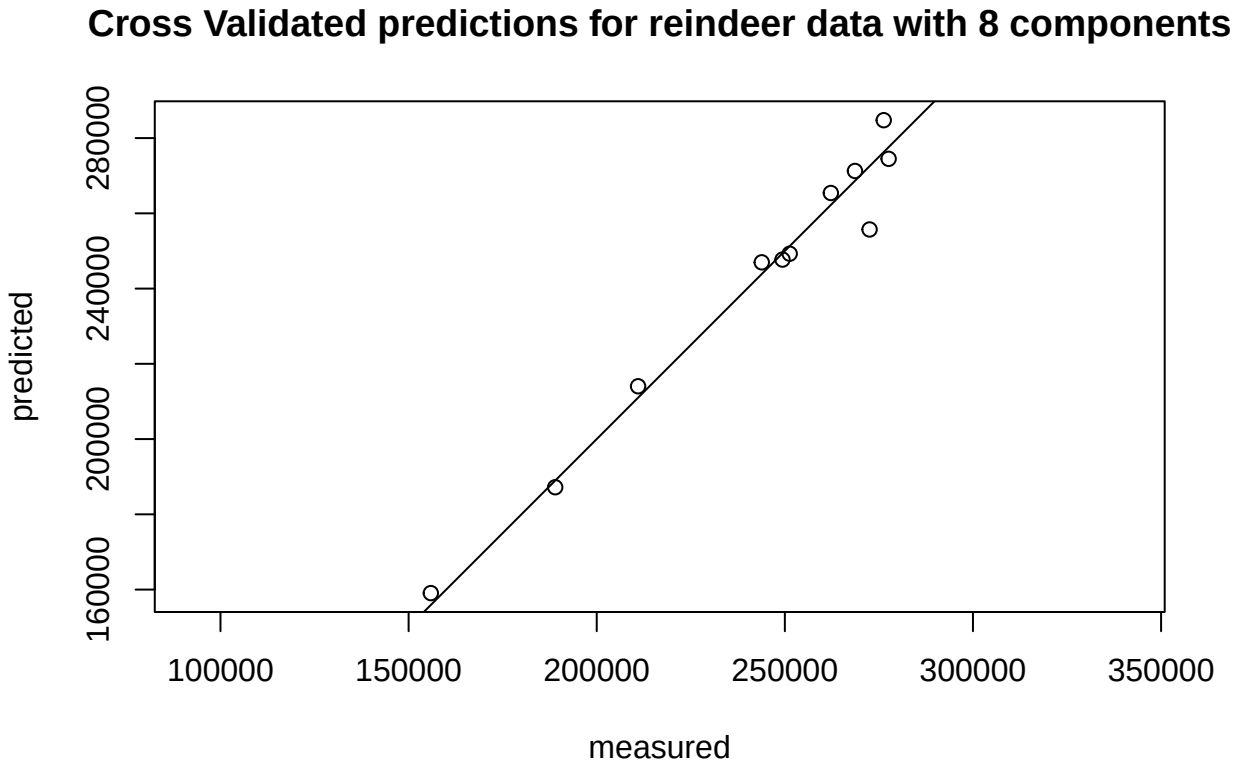
```r
#RMSEP over Components: this is the visual way to determine the number of components: 6
rme<- plot(RMSEP(pls_model2), legendpos = "topright")
```



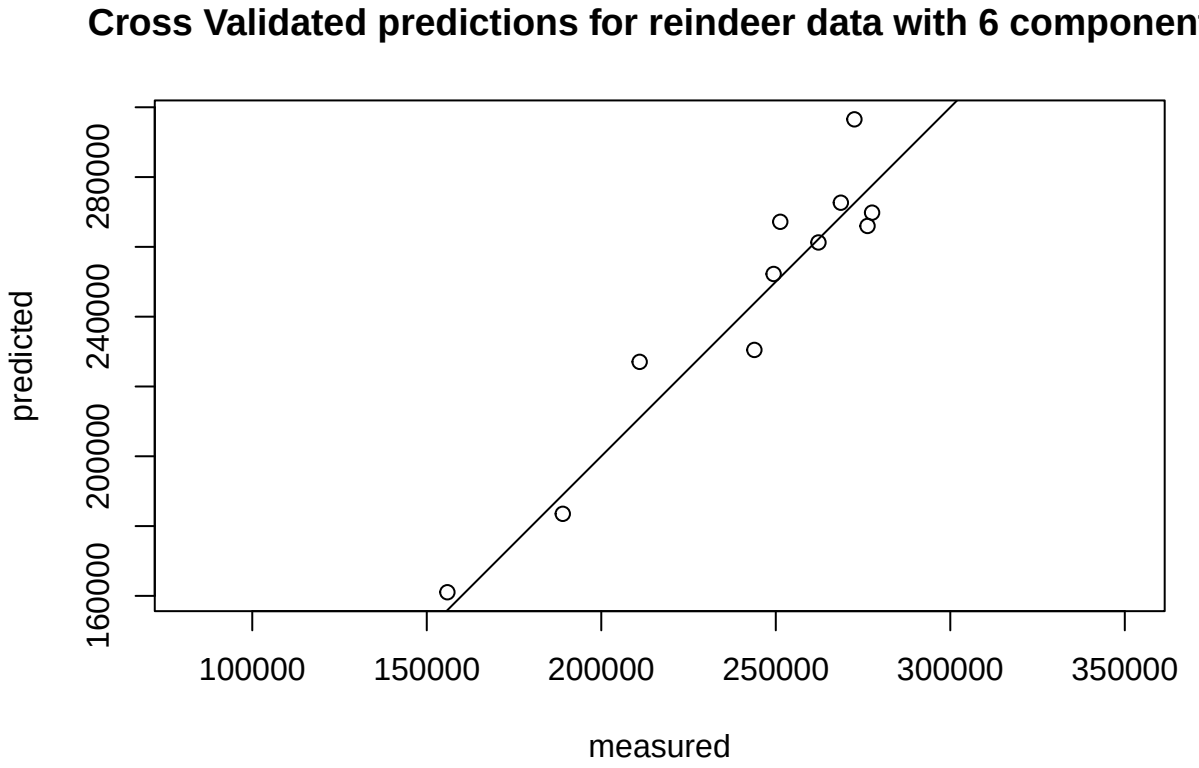

```r
# these plots let us examine how well our model fits the data based on the number of
# components we selected:
plot(pls_model2, ncomp = 1, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data w
```



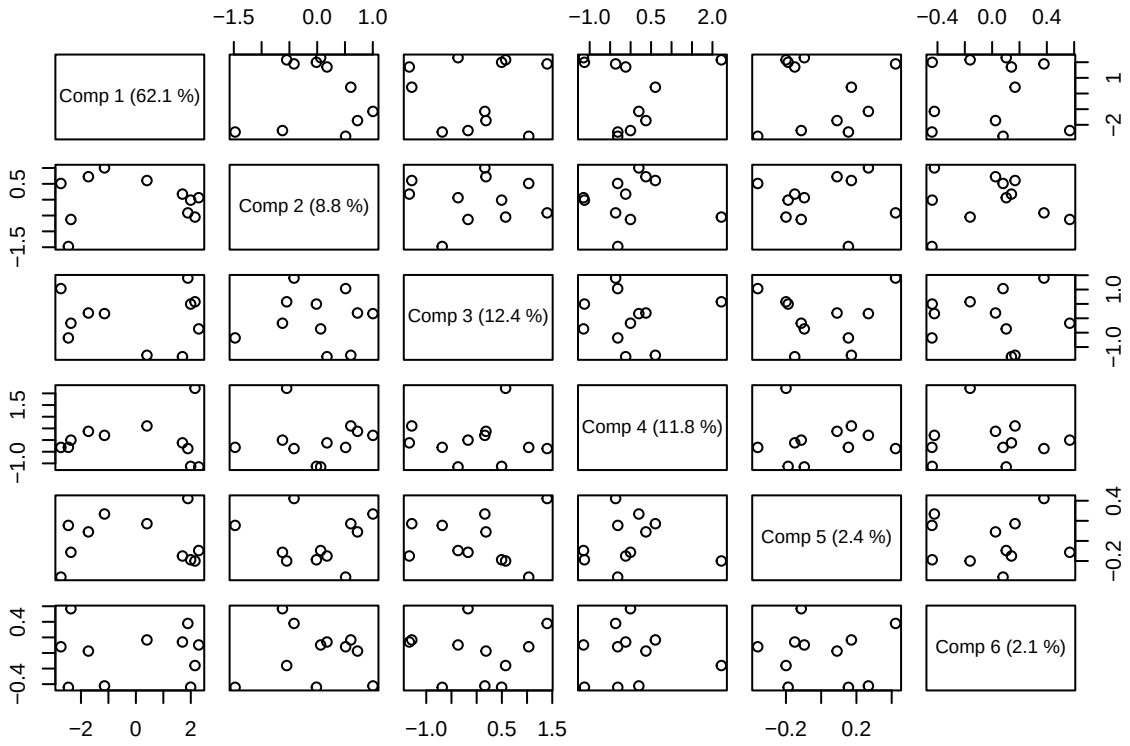

```
plot(pls_model2, ncomp = 8, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data w
```




```
plot(pls_model2, ncomp = 6, asp = 1, line = TRUE, main="Cross Validated predictions for reindeer data w
```




```
# this is a way to visually inspect the data for any outliers or oddities:
plot(pls_model2, plottype = "scores", comps = 1:6)
```

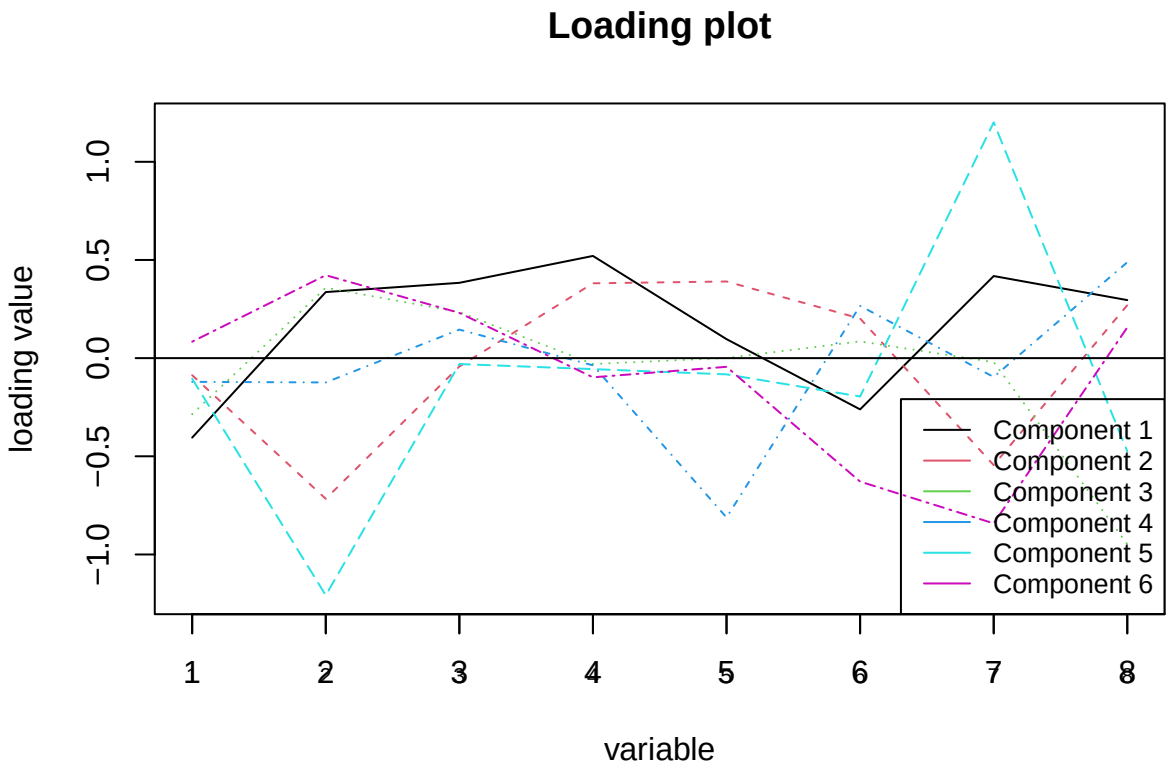



```
explvar(pls_model2)
```

```
##      Comp 1      Comp 2      Comp 3      Comp 4      Comp 5      Comp 6
## 62.10210500  8.82199025 12.38368312 11.84017507  2.44290988  2.11688691
```

```
##      Comp 7      Comp 8
##  0.26331771  0.02893207
```

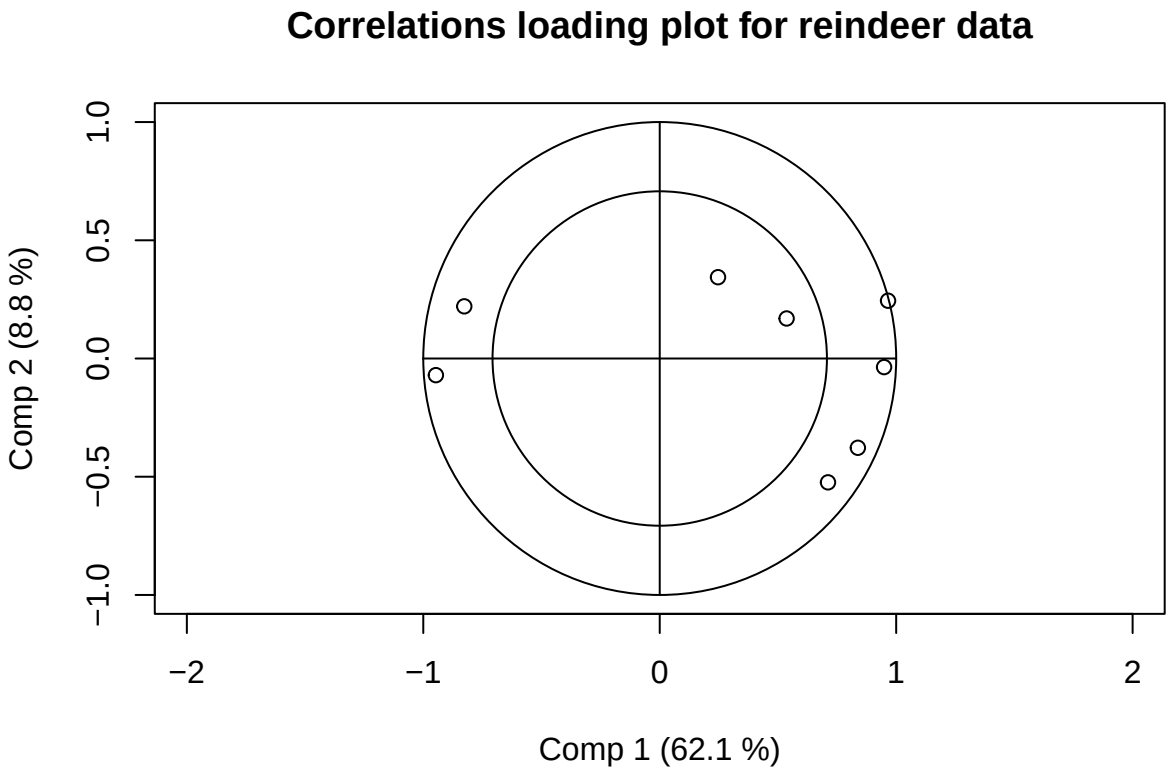
```r
# these plots show variable loading value on component: in simple terms, how much the
# variable has an effect on the component, with points near zero having little effect
# plot 1#
plot(pls_model2, "loadings", comps = 1:6,main="Loading plot")
legend("bottomright", legend = paste("Component", 1:6), col = 1:9, lty = 1, cex = 0.8)
abline(h = 0)
axis(side = 1, at = 1:9, labels = colnames(pls_model2$Xvar), cex.axis = 0.7)
```



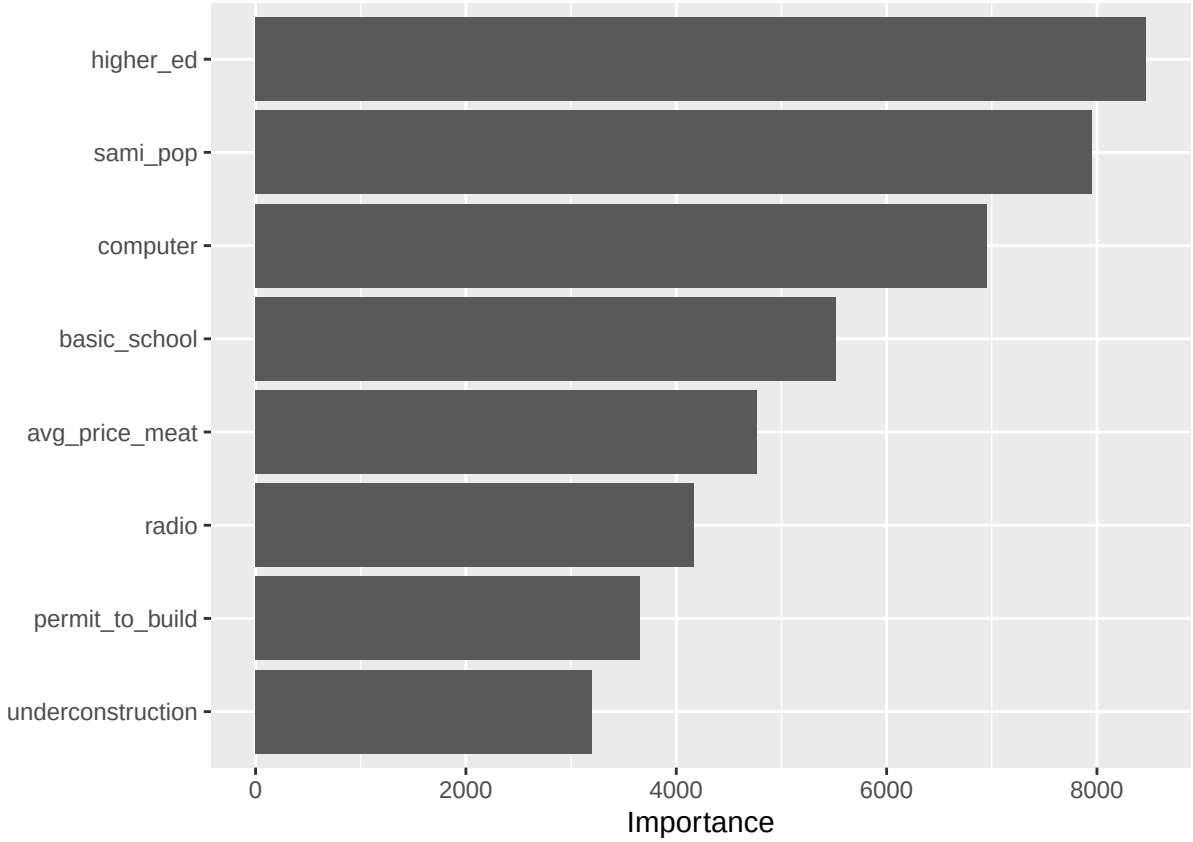

```r
# plot 2
plot(pls_model2, plottype = "correlation", ncomp=1:6, legendpos = "bottomleft",main="Correlations loadi
```

```
## Warning in plot.window(...): "ncomp" is not a graphical parameter

## Warning in plot.window(...): "legendpos" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "ncomp" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "legendpos" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "ncomp" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "legendpos" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "ncomp" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "legendpos" is not
## a graphical parameter

## Warning in box(...): "ncomp" is not a graphical parameter
```

```
## Warning in box(...): "legendpos" is not a graphical parameter
## Warning in title(...): "ncomp" is not a graphical parameter
## Warning in title(...): "legendpos" is not a graphical parameter
```



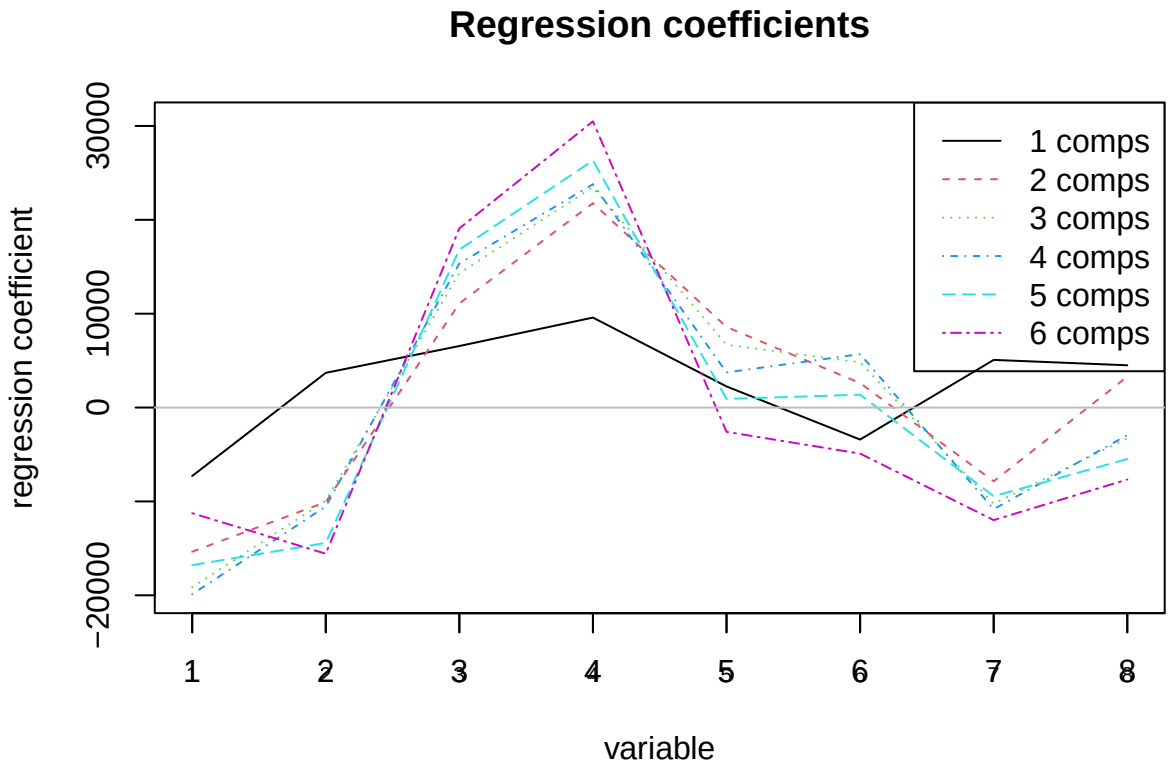

```
# the vi and vip function shows the "variable importance in projection" and tells us
# which variables had the strongest effect on the model
vip(pls_model2)
```




```
vi(pls_model2)
```

```
## # A tibble: 8 x 2
##   Variable          Importance
##   <chr>                  <dbl>
## 1 higher_ed              8468.
## 2 sami_pop               7949.
## 3 computer               6955.
## 4 basic_school           5516.
## 5 avg_price_meat         4770.
## 6 radio                  4165.
## 7 permit_to_build        3658.
## 8 underconstruction      3196.
```

```
# the regression coefficient plot looks at how strongly each variable related to the
# outcome variable. This plot shows 3 components:
plot(pls_model2, plottype = "coef", ncomp=1:6, legendpos = "topright",main="Regression coefficients")
axis(side = 1, at = 1:9, labels = colnames(pls_model2$Xvar), cex.axis = 0.7)
```




```
# This is the predict feature of the model for each of the number of the components
# suggested above.
prediction2_1 = predict(pls_model, ncomp = 1, newdata = test)
prediction2_8 = predict(pls_model2, ncomp = 8, newdata = test)
prediction2_6 = predict(pls_model2, ncomp = 6, newdata=test)
# actual outcome:
actual = (reindeer$total_reindeer[12:15])
# print to console for a visual comparison
(prediction2_1)
```

```
## , , 1 comps
##
##    training_data$total_reindeer
## 12                     279017.0
## 13                     282770.6
## 14                     277472.7
## 15                     273606.9
```

```
(prediction2_8)
```

```
## , , 8 comps
##
##    total_reindeer
## 12       268893.2
## 13       256645.0
## 14       282742.2
## 15       208797.1
```

```
(prediction2_6)
```

```
## , , 6 comps
```

```
##
##    total_reindeer
## 12       282177.7
## 13       258534.9
## 14       271662.1
## 15       207010.4
```

```
(actual)
```

```
## [1] 274969 256095 234044 211227
```

```
# Outcomes for the 3 component choices:

# From the permutation method: 1 component:
print("RMSE for prediction::actual with ncomp = 1")
```

```
## [1] "RMSE for prediction::actual with ncomp = 1"
```

```
sqrt(mean((prediction2_1 - actual)^2))
```

```
## [1] 40327.65
```

```
# Calculate mean absolute error
mae2_1 <- mean(abs(prediction2_1 - actual))
# Calculate R-squared
r_squared2_1 <- 1 - sum((actual - prediction2_1)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae2_1))
```

```
## [1] "Mean Absolute Error:  34133.0370657873"
```

```
print(paste("R-squared: ", r_squared2_1))
```

```
## [1] "R-squared:  -1.85503136964292"
```

```
# From the one-sigma method: 8 components:
print("RMSE for prediction::actual with ncomp = 8")
```

```
## [1] "RMSE for prediction::actual with ncomp = 8"
```

```
sqrt(mean((prediction2_8 - actual)^2))
```

```
## [1] 24569.48
```

```
# Calculate mean absolute error
mae2_8 <- mean(abs(prediction2_8 - actual))
# Calculate R-squared
r_squared2_8 <- 1 - sum((actual - prediction2_8)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae2_8))
```

```
## [1] "Mean Absolute Error:  14438.4806837827"
```

```
print(paste("R-squared: ", r_squared2_8))
```

```
## [1] "R-squared:  -0.059733794641674"
```

```
# From the visual method: 6 components:
print("RMSE for prediction::actual with ncomp = 6")
```

```
## [1] "RMSE for prediction::actual with ncomp = 6"
```

```r
sqrt(mean((prediction2_6 - actual)^2))
```

```
## [1] 19305.55
```

```r
# Calculate mean absolute error
mae2_6 <- mean(abs(prediction2_6 - actual))
# Calculate R-squared
r_squared2_6 <- 1 - sum((actual - prediction2_6)^2) / sum((mean(actual) - actual)^2)
# Print MAE and R-squared
print(paste("Mean Absolute Error: ", mae2_6))
```

```
## [1] "Mean Absolute Error:  12870.804719289"
```

```r
print(paste("R-squared: ", r_squared2_6))
```

```
## [1] "R-squared:  0.345711252963841"
```