Assignment 1

Hsuan Lee

1 Partial least squares

1. Download the corn data and store it in your assignment folder.

```
corn <- readRDS("data/corn.RDS")
```

First, let us check the data.

```
head(corn) # for the first six row
```

```
## # A tibble: 6 x 704
##   Moisture   Oil Protein Starch  `1100`  `1102`  `1104`  `1106`  `1108`  `1110`
##      <dbl> <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     10.4  3.69    8.75   64.8 -0.0227 -0.0228 -0.0229 -0.0229 -0.0229 -0.0228
## 2     10.4  3.72    8.66   64.9 -0.0219 -0.0221 -0.0222 -0.0222 -0.0222 -0.0222
## 3     10.3  3.50    9.12   63.6 -0.0209 -0.0210 -0.0211 -0.0212 -0.0212 -0.0212
## 4     10.3  3.50    9.39   63.3 -0.0236 -0.0238 -0.0239 -0.0239 -0.0239 -0.0239
## 5     10.3  3.66    8.95   64.1 -0.0152 -0.0153 -0.0154 -0.0155 -0.0155 -0.0155
## 6     10.3  3.51    8.73   64.3 -0.0157 -0.0159 -0.0160 -0.0160 -0.0160 -0.0160
## # ... with 694 more variables: `1112` <dbl>, `1114` <dbl>, `1116` <dbl>,
## #   `1118` <dbl>, `1120` <dbl>, `1122` <dbl>, `1124` <dbl>, `1126` <dbl>,
## #   `1128` <dbl>, `1130` <dbl>, `1132` <dbl>, `1134` <dbl>, `1136` <dbl>,
## #   `1138` <dbl>, `1140` <dbl>, `1142` <dbl>, `1144` <dbl>, `1146` <dbl>,
## #   `1148` <dbl>, `1150` <dbl>, `1152` <dbl>, `1154` <dbl>, `1156` <dbl>,
## #   `1158` <dbl>, `1160` <dbl>, `1162` <dbl>, `1164` <dbl>, `1166` <dbl>,
## #   `1168` <dbl>, `1170` <dbl>, `1172` <dbl>, `1174` <dbl>, `1176` <dbl>, ...
```

```
tail(corn) # for the last six rows
```

```
## # A tibble: 6 x 704
##   Moisture   Oil Protein Starch    `1100`    `1102`    `1104`    `1106`   `1108`
##      <dbl> <dbl>   <dbl>  <dbl>     <dbl>     <dbl>     <dbl>     <dbl>    <dbl>
## 1     9.87  3.10    8.65   65.4 -0.00169  -0.00182  -0.00191  -0.00195  -1.94e-3
## 2    10.8   3.25    7.88   65.2 -0.0174   -0.0175   -0.0176   -0.0176   -1.76e-2
## 3    10.1   3.49    8.59   65.7 -0.000436 -0.000556 -0.000639 -0.000673 -6.55e-4
## 4    10.4   3.34    8.03   65.1 -0.0179   -0.0180   -0.0181   -0.0181   -1.81e-2
## 5    10.6   3.18    8.13   65.2 -0.00680  -0.00689  -0.00696  -0.00698  -6.97e-3
## 6    11.0   3.33    8.43   64.9 -0.0153   -0.0154   -0.0155   -0.0155   -1.55e-2
## # ... with 695 more variables: `1110` <dbl>, `1112` <dbl>, `1114` <dbl>,
## #   `1116` <dbl>, `1118` <dbl>, `1120` <dbl>, `1122` <dbl>, `1124` <dbl>,
## #   `1126` <dbl>, `1128` <dbl>, `1130` <dbl>, `1132` <dbl>, `1134` <dbl>,
```

```
## #   `1136` <dbl>, `1138` <dbl>, `1140` <dbl>, `1142` <dbl>, `1144` <dbl>,
## #   `1146` <dbl>, `1148` <dbl>, `1150` <dbl>, `1152` <dbl>, `1154` <dbl>,
## #   `1156` <dbl>, `1158` <dbl>, `1160` <dbl>, `1162` <dbl>, `1164` <dbl>,
## #   `1166` <dbl>, `1168` <dbl>, `1170` <dbl>, `1172` <dbl>, `1174` <dbl>, ...
```

```
# check the dimentionality
dim(corn)
```

```
## [1]  80 704
```

The dataset has 80 rows and 704 columns, which means that there are 704 features and 80 observations. The data is a high dimension data.

2. Pick a property (`Moisture`, `Oil`, `Starch`, or `Protein`) to predict.

Moisture is pick as the outcome feature.

3. Split your data into a training (80%) and test (20%) set.

```
set.seed(9252568)

# remove the unused features in the data
corn <- corn %>%
  select(-Oil, -Starch, -Protein)

corn_samp <- corn[sample(nrow(corn)),] # reordering the data
train <- seq(1, nrow(corn) * 0.8)
test <- seq(max(train) + 1, nrow(corn))

corn_train <- corn_samp[train,]
corn_test <- corn_samp[test,]
```

4. Use the function `plsr` from the package `pls` to estimate a partial least squares model, predicting the property using the NIR spectroscopy measurements in the training data. Make sure that the features are on the same scale. Use leave-one-out cross-validation (built into plsr) to estimate out-of-sample performance.

```
pls_model.1 <- plsr(Moisture ~.,
                    data = corn_train,
                    scale = T,
                    center = T, # the default is to perform mean center
                    validation = "LOO")
```

5. Find out which component best predicts the property you chose. Explain how you did this.

```
summary(pls_model.1)
```

```
## Data:    X dimension: 64 700
##  Y dimension: 64 1
## Fit method: kernelpls
## Number of components considered: 62
##
## VALIDATION: RMSEP
```

```
## Cross-validated using 64 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.3674   0.2940   0.2524   0.2255   0.1980   0.1851   0.1765
## adjCV       0.3674   0.2939   0.2523   0.2254   0.1979   0.1850   0.1764
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.1661   0.1516   0.1464    0.1435    0.1437    0.1456    0.1510
## adjCV   0.1659   0.1512   0.1462    0.1432    0.1433    0.1453    0.1507
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.1571    0.1606    0.1490    0.1526    0.1502    0.1528    0.1571
## adjCV    0.1566    0.1599    0.1484    0.1519    0.1496    0.1515    0.1563
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.1569    0.1594    0.1565    0.1604    0.1603    0.1674    0.1696
## adjCV    0.1555    0.1585    0.1556    0.1595    0.1594    0.1663    0.1685
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.1782    0.1836    0.1803    0.1807    0.1819    0.1808    0.1779
## adjCV    0.1769    0.1823    0.1790    0.1794    0.1805    0.1794    0.1765
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV       0.1762    0.1747    0.1752    0.1755    0.1758    0.1760    0.1758
## adjCV    0.1748    0.1734    0.1738    0.1741    0.1744    0.1746    0.1744
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.1756    0.1755    0.1755    0.1755    0.1755    0.1755    0.1755
## adjCV    0.1743    0.1741    0.1741    0.1741    0.1741    0.1741    0.1741
##        49 comps  50 comps  51 comps  52 comps  53 comps  54 comps  55 comps
## CV       0.1755    0.1755    0.1755    0.1755    0.1755    0.1755    0.1755
## adjCV    0.1741    0.1741    0.1741    0.1741    0.1741    0.1741    0.1741
##        56 comps  57 comps  58 comps  59 comps  60 comps  61 comps  62 comps
## CV       0.1755    0.1755    0.1755    0.1755    0.1755    0.1755    0.1755
## adjCV    0.1741    0.1741    0.1741    0.1741    0.1741    0.1741    0.1741
##
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           98.51    99.58    99.85    99.92    99.96    99.98    99.99
## Moisture    38.01    56.17    67.41    75.90    80.37    82.95    86.23
##           8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X           99.99    99.99    100.00    100.00     100.0    100.00    100.00
## Moisture    90.06    91.22     92.27     93.09      93.3     93.66     94.33
##           15 comps  16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
## X           100.00    100.00    100.00     100.0    100.00    100.00    100.00
## Moisture     95.03     95.71     96.15      96.5     97.27     97.38     97.99
##           22 comps  23 comps  24 comps  25 comps  26 comps  27 comps  28 comps
## X           100.00    100.00    100.00    100.00    100.00    100.00    100.00
## Moisture     98.29     98.53     98.72     98.95     99.21     99.38     99.54
##           29 comps  30 comps  31 comps  32 comps  33 comps  34 comps  35 comps
## X           100.00    100.00    100.00    100.00    100.00    100.00    100.00
## Moisture     99.71     99.79     99.87     99.91     99.94     99.97     99.98
##           36 comps  37 comps  38 comps  39 comps  40 comps  41 comps  42 comps
## X           100.00       100       100       100       100       100       100
## Moisture     99.99       100       100       100       100       100       100
##           43 comps  44 comps  45 comps  46 comps  47 comps  48 comps  49 comps
## X              100       100       100       100       100       100       100
## Moisture       100       100       100       100       100       100       100
##           50 comps  51 comps  52 comps  53 comps  54 comps  55 comps  56 comps
## X              100       100       100       100       100       100       100
## Moisture       100       100       100       100       100       100       100
```

```
##           57 comps  58 comps  59 comps  60 comps  61 comps  62 comps
## X              100       100       100       100       100       100
## Moisture       100       100       100       100       100       100
```

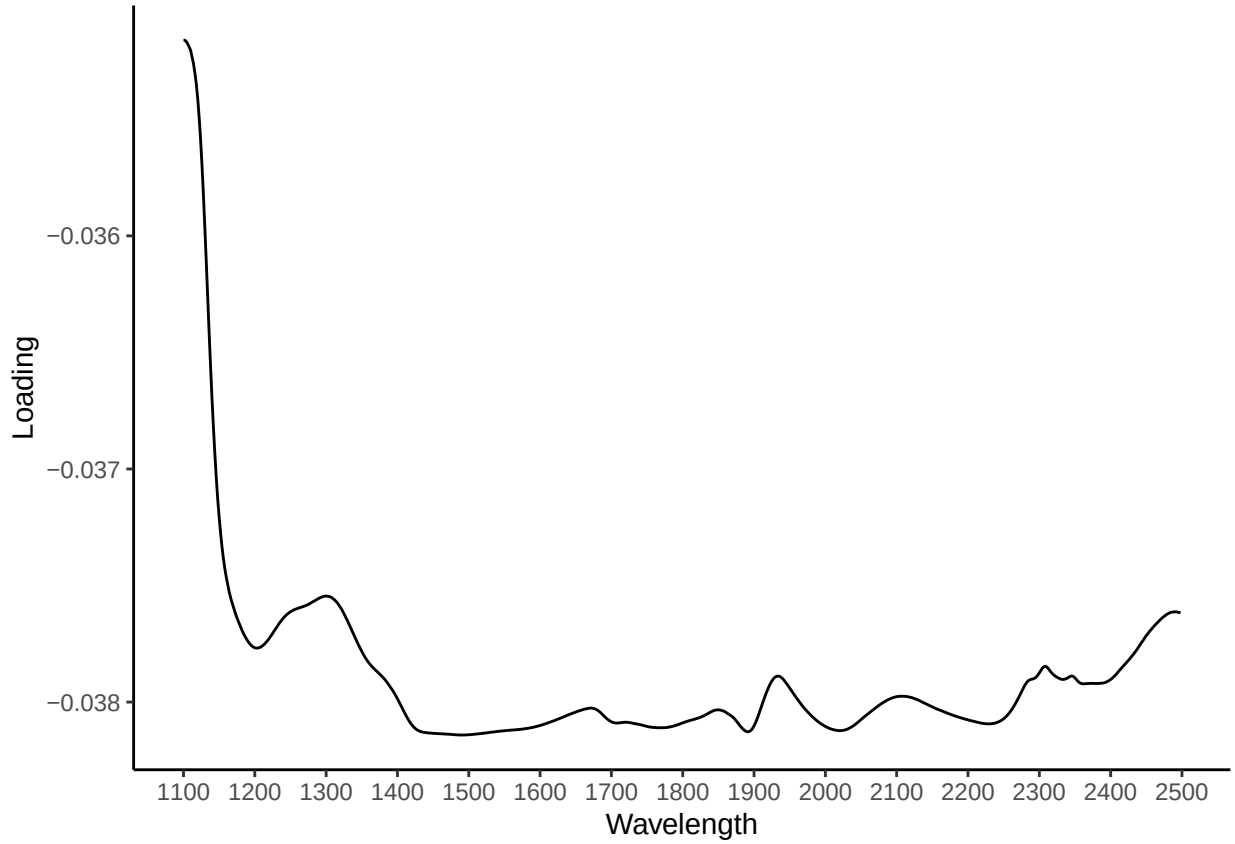
This table allows us to determine the percentage of the variance of the response variable explained by the PLS component. The first PLS component offers the best prediction of the response variable, which explains 38.01% of the variation in `Moisture`.

6. Create a plot with on the x-axis the wavelength, and on the y-axis the strength of the loading for this component. Explain which wavelengths are most important for predicting the property you are interested in.

```
Wavelength <- names(corn_train)[-1] # extract the features
Loading <- pls_model.1$loadings[1:700] # extract the loadings

plot <- data.frame(Wavelength = as.numeric(Wavelength), Loading = Loading)

plot %>%
  ggplot(aes(x = Wavelength, y = Loading)) +
  geom_line() +
  scale_x_continuous(n.breaks = 20) +
  theme_classic()
```



As can be seen from the plot, the wavelengths between approximately 1450 and 1600 are the most important for predicting Moisture, as this range holds the most powerful loading values.

7. Pick the number of components included in the model based on the "one standard deviation" rule (`selectNcomp()`). Create predictions for the test set using the resulting model.

```
#??selectNcomp
selectNcomp(pls_model.1,
            method = "onesigma") # one standard deviation
```

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

```
# predict the test set using the resulting model
pred_pls_test <- predict(pls_model.1, newdata = corn_test,
        ncomp = 8)
```

8. Compare your PLS predictions to a LASSO linear regression model where lambda is selected based on cross-validation with the one standard deviation rule (using `cv.glmnet`).

```
# first extract the predictors and outcome variable, then make the data as matrix
x_train <- corn_train %>%
  select(-Moisture)
x_train <- as.matrix(x_train)

y_train <- corn_train %>%
  select(Moisture)
y_train <- as.matrix(y_train)

x_test <- corn_test %>%
  select(-Moisture)
x_test <- as.matrix(x_test)

y_test <- corn_test %>%
  select(Moisture)
y_test <- as.matrix(y_test)
```

Now, fit the model on training data:

```
lasso_model.1 <- cv.glmnet(x = x_train,
                           y = y_train,
                           nfold = 10, # default
                           family = "gaussian",
                           alpha = 1)
```

And predict the test set using the LASSO model,

```
pred_lasso_test <- predict(lasso_model.1, newx = x_test, "lambda.min")
```

We use MSE to compare the two models, i.e., PLS model and LASSO model:

```
# create a function for computing MSE
mse <- function(y_true, y_pred){
  mse = mean((y_true - y_pred)^2)
  return(mse)
}

# MSE of PLS model
```

```r
PLS_MSE <- mse(y_true = corn_test$Moisture, y_pred = pred_pls_test)

PLS_MSE
```

```
## [1] 0.02771147
```

```r
# MSE of LASSO model
LASSO_MSE <- mse(y_true = corn_test$Moisture, y_pred = pred_lasso_test)

LASSO_MSE
```

```
## [1] 0.07253661
```

As the MSE of the PLS model is lower than that of the LASSO model, the PLS model possesses better performance in our case.