Problem 5 - Data analysis I

alexaoh

27 mai, 2021

All needed packages are run first in a chunk with `echo = FALSE`.

Import data.

```
id <- "1dNLfx9Dbs2gYIooUxA6HMxK_MPFwE3Hn"
d.bodyfat <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = T)
# pairs(d.bodyfat)
str(d.bodyfat)
```

```
#> 'data.frame':    243 obs. of  16 variables:
#>  $ bodyfat: num  12.3 6.1 25.3 10.4 28.7 20.9 19.2 12.4 4.1 11.7 ...
#>  $ age    : int  23 22 22 26 24 24 26 25 25 23 ...
#>  $ weight : num  70 78.7 69.9 83.9 83.7 ...
#>  $ height : num  172 184 168 184 181 ...
#>  $ bmi    : num  23.6 23.4 24.7 24.9 25.5 ...
#>  $ neck   : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 42.1 ...
#>  $ abdomen: num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5 88.6 ...
#>  $ hip    : num  94.5 98.7 99.2 101.2 101.9 ...
#>  $ chest  : num  93.1 93.6 95.8 101.8 97.3 ...
#>  $ thigh  : num  59 58.7 59.6 60.1 63.2 66 58.4 60 62.9 63.1 ...
#>  $ knee   : num  37.3 37.3 38.9 37.3 42.2 42 38.3 39.4 38.3 41.7 ...
#>  $ ankle  : num  21.9 23.4 24 22.8 24 25.6 22.9 23.2 23.8 25 ...
#>  $ biceps : num  32 30.5 28.8 32.4 32.2 35.7 31.9 30.5 35.9 35.6 ...
#>  $ forearm: num  27.4 28.9 25.2 29.4 27.7 30.6 27.8 29 31.1 30 ...
#>  $ wrist  : num  17.1 18.2 16.6 18.2 17.7 18.8 17.7 18.8 18.2 19.2 ...
#>  $ head   : num  59.1 64.1 52.2 57.9 58.6 ...
```

```
set.seed(1234)
samples <- sample(1:243, 180, replace = F)
d.body.train <- d.bodyfat[samples, ]
d.body.test <- d.bodyfat[-samples, ]
```

a)

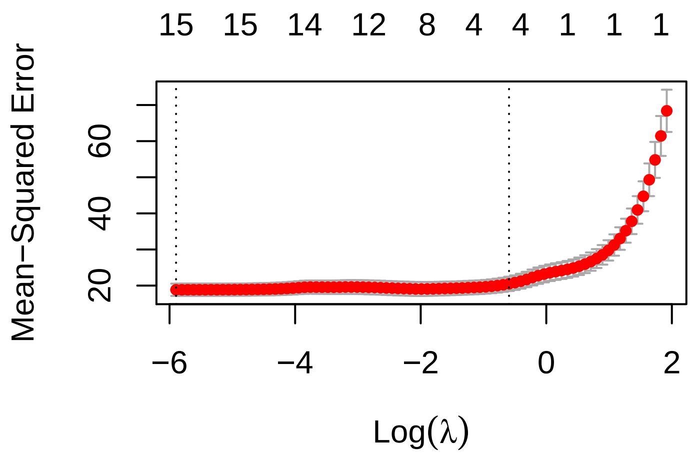
Lasso regression is performed below. The parameter λ is chosen via 10-fold cross-validation, where it is chosen to be the largest value of λ such that the mean cross-validated error is within one standard error of the minimum, following the principle of parsimony.

```
# Make data on correct format.
x.train <- model.matrix(bodyfat ~ ., data = d.body.train)[, -1]  # Remove the intercept.
y.train <- d.body.train$bodyfat
x.test <- model.matrix(bodyfat ~ ., data = d.body.test)[, -1]  # Remove the intercept.
```

```
y.test <- d.body.test$bodyfat

lasso.mod <- glmnet(x.train, y.train, alpha = 1)  # alpha = 1 gives Lasso.

cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.lasso)
```




```
(lambda.lasso <- cv.lasso$lambda.1se)
```

```
#> [1] 0.5523759
```

```
lasso.pred <- predict(lasso.mod, s = lambda.lasso, newx = x.test)
(lasso.mse <- mean((lasso.pred - y.test)^2))
```

```
#> [1] 18.87964
```

b)

Linear regression with all covariates.

```
lm.fit <- lm(bodyfat ~ ., data = d.body.train)
linreg.pred <- predict(lm.fit, newdata = d.body.test)
(linreg.mse <- mean((linreg.pred - d.body.test$bodyfat)^2))
```

```
#> [1] 22.10107
```

```
# Comparison.
(errors <- data.frame(lasso = lasso.mse, linreg = linreg.mse))
```

```
#>      lasso   linreg
#> 1 18.87964 22.10107
```

The data frame above compares the errors from the two methods. We can see that the error is smaller for the lasso compared to the linreg. The difference is most likely taking place because there are many covariates that are not good predictors of, or strongly related to, bodyfat.

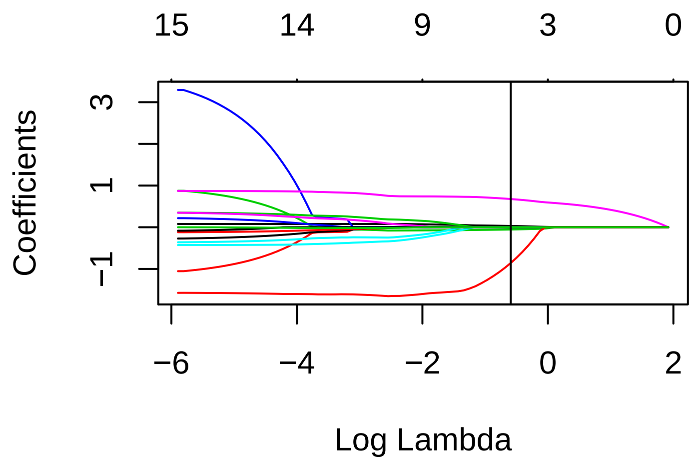
```
lasso.best <- glmnet(x.train, y.train, alpha = 1, lambda = lambda.lasso)
coef(lasso.best)
```

```
#> 16 x 1 sparse Matrix of class "dgCMatrix"
#>                      s0
#> (Intercept) -19.79483282
#> age           0.03073728
#> weight        .
#> height       -0.05366148
#> bmi           .
#> neck          .
#> abdomen       0.67712360
#> hip           .
#> chest         .
#> thigh         .
#> knee          .
#> ankle         .
#> biceps        .
#> forearm       .
#> wrist        -0.85988914
#> head          .
```

```r
summary(lm.fit)
```

```
#>
#> Call:
#> lm(formula = bodyfat ~ ., data = d.body.train)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -9.5012 -2.6305 -0.0839  2.9606  9.8508
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -1.948e+02  6.519e+01  -2.989  0.00323 **
#> age          7.882e-02  3.882e-02   2.031  0.04390 *
#> weight      -1.238e+00  3.983e-01  -3.107  0.00223 **
#> height       1.045e+00  3.609e-01   2.897  0.00429 **
#> bmi          3.887e+00  1.275e+00   3.048  0.00269 **
#> neck        -4.276e-01  2.587e-01  -1.653  0.10032
#> abdomen      8.740e-01  1.102e-01   7.928 3.24e-13 ***
#> hip         -2.990e-01  1.879e-01  -1.592  0.11342
#> chest       -1.265e-01  1.246e-01  -1.015  0.31139
#> thigh        3.639e-01  1.749e-01   2.081  0.03901 *
#> knee         2.478e-01  3.014e-01   0.822  0.41218
#> ankle       -3.749e-01  4.201e-01  -0.892  0.37345
#> biceps       3.747e-01  2.367e-01   1.583  0.11544
#> forearm     -1.080e-01  2.797e-01  -0.386  0.69978
#> wrist       -1.562e+00  6.453e-01  -2.420  0.01660 *
#> head         4.003e-03  8.173e-02   0.049  0.96099
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.148 on 164 degrees of freedom
#> Multiple R-squared:  0.7704, Adjusted R-squared:  0.7494
#> F-statistic: 36.68 on 15 and 164 DF,  p-value: < 2.2e-16
```

```r
plot(glmnet(x.train, y.train, alpha = 1), "lambda")
abline(v = log(lambda.lasso))
```



From the output above, it is apparent that many of the coefficients are set to zero by Lasso. This is the regularization effect of Lasso that makes the big difference in the estimates and also in the models overall. We can also see that the covariates that are deemed important, i.e. not set to zero, by Lasso are shrinked compared to the similar coefficients in the linear regression. This is, again, a consequence of the regularization effect of Lasso, with the goal of decreasing variance more than the increase in bias, such that the overall test error is decreased.

c)

A GAM is fit below.

```r
gam.fit <- gam(bodyfat ~ poly(age, 2) + ns(height, df = 3) + ns(abdomen, df = 4) +
    s(hip) + weight + bmi, data = d.body.train)

# ns with df = 4 gives knots at the given percentiles automatically.

gam.pred <- predict(gam.fit, newdata = d.body.test)
(gam.mse <- mean((gam.pred - d.body.test$bodyfat)^2))
```

```
#> [1] 20.37389
```

```r
(errors <- data.frame(errors, gam = gam.mse))  # All the errors again.
```

```
#>      lasso   linreg      gam
#> 1 18.87964 22.10107 20.37389
```

d)

PLS regression on training data is run below. It is run with 10-fold cross-validation also, such that one can see what the CV-error is for each possible number of principal components used.

```r
pls.fit <- plsr(bodyfat ~ ., scale = TRUE, validation = "CV", data = d.body.train)
summary(pls.fit)
```

```
#> Data:    X dimension: 180 15
```

```
#>  Y dimension: 180 1
#> Fit method: kernelpls
#> Number of components considered: 15
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV           8.309    6.174     4.92    4.724    4.675    4.598    4.525
#> adjCV        8.309    6.171     4.91    4.708    4.651    4.570    4.500
#>        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
#> CV       4.491    4.444     4.45     4.495     4.465     4.398     4.400
#> adjCV    4.467    4.425     4.43     4.469     4.431     4.372     4.376
#>        14 comps  15 comps
#> CV        4.397     4.384
#> adjCV     4.372     4.361
#>
#> TRAINING: % variance explained
#>          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
#> X          58.36    70.33    75.49    79.44    82.86    86.58    90.88    93.18
#> bodyfat    45.86    67.46    71.35    73.09    74.73    75.46    75.71    75.81
#>          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
#> X          94.95     96.34     96.83     97.73     98.53     98.84    100.00
#> bodyfat    75.87     76.01     76.62     76.88     77.01     77.04     77.04
```

The smallest number of components such that at least 95% of the covariate variance in the training data is explained is 10, as seen from the output above. Nine principal components explain 94.95, which is just short of 95, which is why we have to choose 10 components.

The MSE when using these components is reported below.

```r
pls.pred <- predict(pls.fit, d.body.test, ncomp = 10)
(pls.mse <- mean((pls.pred - d.body.test$bodyfat)^2))
```

```
#> [1] 19.63273
```

```r
(errors <- data.frame(errors, pls = pls.mse))  # All the errors again, for comparison.
```

```
#>      lasso   linreg      gam      pls
#> 1 18.87964 22.10107 20.37389 19.63273
```

e)

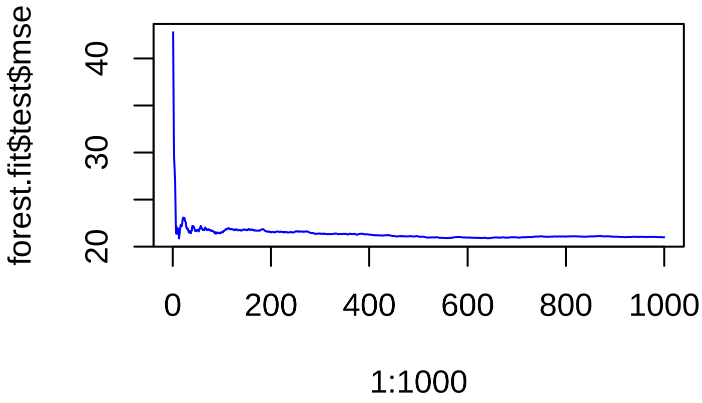
Below a random forest has been fitted. The number of covariates allowed to use in each split m is chosen to $p/3 = 5$, since this is a regression problem. Moreover, the amount of trees (which is not a tuning parameter!) is chosen to $B = 600$, based on the plot below, since the error is stable here.

```r
(m <- round(ncol(d.body.train)/3))  # regression.
```

```
#> [1] 5
```

```r
train.predictors <- d.body.train[, -1]  # remove bodyfat.
y.train <- d.body.train[, 1]  # only bodyfat.
test.predictors <- d.body.test[, -1]  # remove bodyfat.
y.test <- d.body.test[, 1]  # only bodyfat.

forest.fit <- randomForest(train.predictors, y = y.train, xtest = test.predictors,
    ytest = y.test, mtry = m, ntree = 1000, importance = T)
plot(1:1000, forest.fit$test$mse, col = "blue", type = "l")
```




```r
# Choose B = 600 for example, since the error seems to be stable after this
# amount.
(forest.mse <- forest.fit$test$mse[600])
```

```
#> [1] 20.97413
```

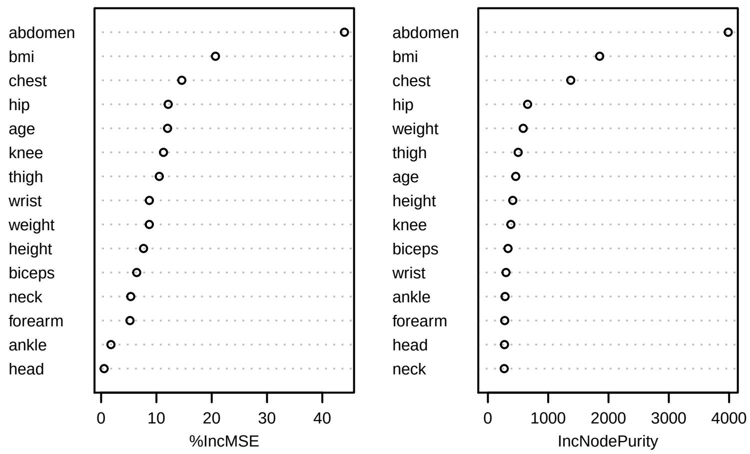
```r
(errors <- data.frame(errors, forest = forest.mse))  # All the errors again, for comparison.
```

```
#>      lasso   linreg      gam      pls   forest
#> 1 18.87964 22.10107 20.37389 19.63273 20.97413
```

```r
importance(forest.fit)
```

```
#>            %IncMSE IncNodePurity
#> age     12.0202131      461.3655
#> weight   8.7142198      585.4932
#> height   7.6759552      412.2492
#> bmi     20.6799744     1853.0991
#> neck     5.3552997      270.4659
#> abdomen 44.0021768     3985.2580
#> hip     12.1292684      658.5902
#> chest   14.5879518     1373.6264
#> thigh   10.5207320      501.9547
#> knee    11.2778401      380.5928
#> ankle    1.7741922      284.7778
#> biceps   6.4355891      333.7055
#> forearm  5.2171304      278.5587
#> wrist    8.7232517      299.9150
#> head     0.5465257      274.9359
```

```r
varImpPlot(forest.fit, main = "", cex = 0.5)  # To see what variables are important.
```



f)

The errors are printed again below. Based on these results, the lasso seems to do the best, and the linear regression is the worst. However, bear in mind that all the other methods are relatively similar in their errors, which means that these results may change when the splits into training and testing data sets is changed (e.g. if the seed is changed.)

```
errors
```

```
#>      lasso   linreg      gam      pls   forest
#> 1 18.87964 22.10107 20.37389 19.63273 20.97413
```