Lab 4

Moshe Weiss

11:59PM March 9, 2019

Note: the content of this lab is on the midterm exam (March 5) even though the lab itself is due after the midterm exam.

We now move on to simple linear modeling using the ordinary least squares algorithm.

Let's quickly recreate the sample data set from practice lecture 7:

```r
n = 20
x = runif(n)
beta_0 = 3
beta_1 = -2
y = beta_0 + beta_1 * x + rnorm(n, mean = 0, sd = 0.33)
```

Solve for the least squares line by computing b_0 and b_1 *without* using the functions `mean`, `cor`, `cov`, `var`, `sd` but instead computing it from the x and y quantities manually using base function such as `sum` and other basic operators. See the class notes.

```r
meanY = sum(y)/n
meanX = sum(x)/n
b_1 = (sum(x*y) - n*meanX*meanY)/(sum(x^2)-n*meanX^2)
b_0 = meanY - b_1*meanX
```

Verify your computations are correct using the `lm` function in R:

```r
lm_mod = lm(y~x)
b_vec = coef(lm_mod)
pacman::p_load(testthat)
expect_equal(b_0, as.numeric(b_vec[1]), tol = 1e-4)
expect_equal(b_1, as.numeric(b_vec[2]), tol = 1e-4)
```

6. We are now going to repeat one of the first linear model building exercises in history — that of Sir Francis Galton in 1886. First load up package `HistData`.

```r
pacman::p_load(HistData)
```

In it, there is a dataset called `Galton`. Load it up.

```r
galton = data.frame(Galton)
```

You now should have a data frame in your workspace called `Galton`. Summarize this data frame and write a few sentences about what you see. Make sure you report n, p and a bit about what the columns represent and how the data was measured. See the help file `?Galton`.

```r
summary(galton)
```

```
##      parent          child      
##  Min.   :64.00   Min.   :61.70  
##  1st Qu.:67.50   1st Qu.:66.20  
##  Median :68.50   Median :68.20  
##  Mean   :68.31   Mean   :68.09  
##  3rd Qu.:69.50   3rd Qu.:70.20  
##  Max.   :73.00   Max.   :73.70
```

```
str(galton)
```

```
## 'data.frame':    928 obs. of  2 variables:
##  $ parent: num  70.5 68.5 65.5 64.5 64 67.5 67.5 67.5 66.5 66.5 ...
##  $ child : num  61.7 61.7 61.7 61.7 61.7 62.2 62.2 62.2 62.2 62.2 ...
```

```
"p = 2: average mother and father height, and child height (females scaled by 1.08)"
```

```
## [1] "p = 2: average mother and father height, and child height (females scaled by 1.08)"
```

```
"n=928"
```

```
## [1] "n=928"
```

Find the average height (include both parents and children in this computation).

```
pHeights = galton$parent
cHeights = galton$child
n=length(galton$parent)
p=length(galton)
avg_height = (sum(pHeights*2) + sum(cHeights))/(3*n)
avg_height
```

```
## [1] 68.23495
```

If you were to use the null model, what would the RMSE be of this model be?

```
yBar = sum(cHeights/n)
sse = sum((cHeights - yBar)^2)
mse = sse/(n-2)
rmse = sqrt(mse)
paste("RMSE:",rmse)
```

```
## [1] "RMSE: 2.51930057898914"
```

Note that in Math 241 you learned that the sample average is an estimate of the "mean", the population expected value of height. We will call the average the "mean" going forward since it is probably correct to the nearest tenth of an inch with this amount of data.

Run a linear model attempting to explain the childrens' height using the parents' height. Use `lm` and use the R formula notation. Compute and report b_0, b_1, RMSE and R^2. Use the correct units to report these quantities.

```
lmHeights = lm(child~parent, data = Galton)
b_0 = as.numeric(coef(lmHeights)[1])
b_1 = as.numeric(coef(lmHeights)[2])
R2 = summary(lmHeights)$r.squared
RMSE = summary(lmHeights)$sigma
paste("b_0:",b_0)
```

```
## [1] "b_0: 23.9415301804085"
```

```
paste("b_1:",b_1)
```

```
## [1] "b_1: 0.646290581993716"
```

```
paste("R^2:",R2)
```

```
## [1] "R^2: 0.210462910561682"
```

```r
paste("RMSE:", RMSE)
```

```
## [1] "RMSE: 2.23854719318204"
```

Interpret all four quantities: b_0, b_1, RMSE and R^2.

b_0 is your intercept. b_1 is the change in child's height in a one unit change of the parent's height. RMSE is the range of normal variation in child height (give or take) R squared is how much better our model is than the null

How good is this model? How well does it predict? Discuss.

The goodness of a model is determined by its usefuleness, so the answer to that depends on the use-case. We do have an idea of how related parent height is to child height, with some error.

It is reasonable to assume that parents and their children have the same height? Explain why this is reasonable using basic biology and common sense.

Yes. Height is mostly genetic, and genes are inherited from both parents.

If they were to have the same height and any differences were just random noise with expectation 0, what would the values of β_0 and β_1 be?

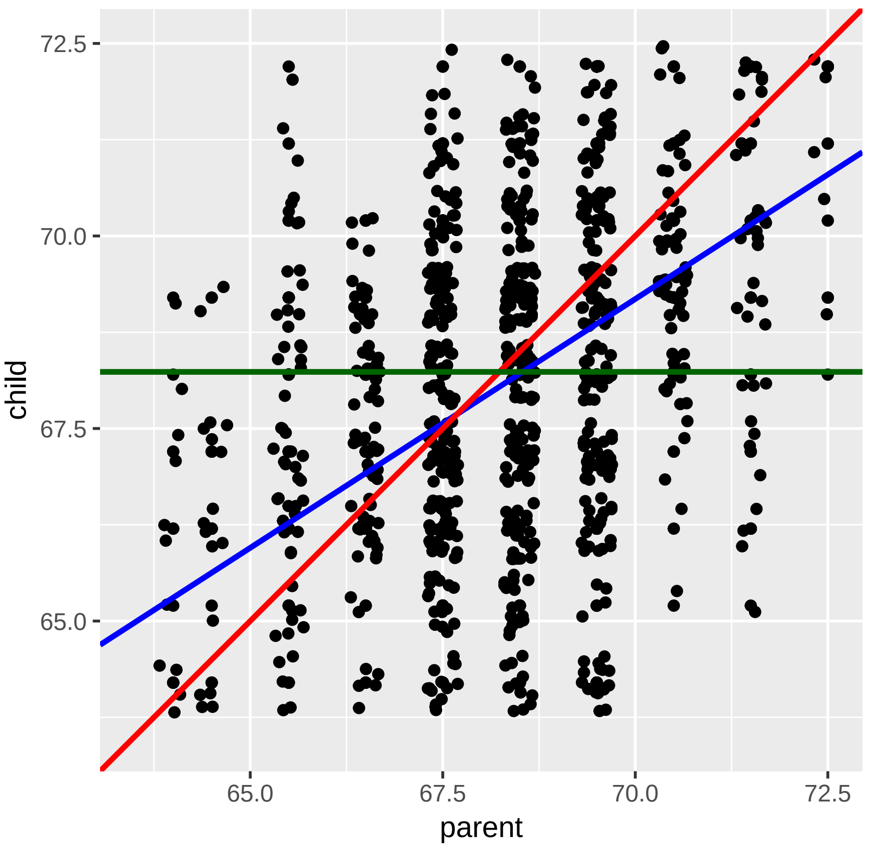
Beta 0 would equal average parent height, and beta 1 would equal 0.

Let's plot (a) the data in $\mathbb{D}$ as black dots, (b) your least squares line defined by b_0 and b_1 in blue, (c) the theoretical line β_0 and β_1 if the parent-child height equality held in red and (d) the mean height in green.

```r
pacman::p_load(ggplot2)
ggplot(Galton, aes(x = parent, y = child)) +
  geom_point() +
  geom_jitter() +
  geom_abline(intercept = b_0, slope = b_1, color = "blue", size = 1) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1) +
  geom_abline(intercept = avg_height, slope = 0, color = "darkgreen", size = 1) +
  xlim(63.5, 72.5) +
  ylim(63.5, 72.5) +
  coord_equal(ratio = 1)
```

```
## Warning: Removed 76 rows containing missing values (geom_point).
```

```
## Warning: Removed 92 rows containing missing values (geom_point).
```



Fill in the following sentence:

Children of short parents became taller on average and children of tall parents became Shorter on average.

Why did Galton call it “Regression towards mediocrity in hereditary stature” which was later shortened to “regression to the mean”?

Because of the above finding: children tended toward the mean.

Why should this effect be real?

The genetics of height is very complicated, but perhaps the average or dominant genetic encoding correlates to our mean.

You now have unlocked the mystery. Why is it that when modeling with y continuous, everyone calls it “regression”? Write a better, more descriptive and appropriate name for building predictive models with y continuous.

Because of Galton’s discovery regarding heredity and mean height.

Linear factor optimization

Create a dataset $\mathbb{D}$ which we call `Xy` such that the linear model as R^2 about 50% and RMSE approximately 1.

```
x = c(7,11,7,11,7,11)
y = c(6,9,8,10,7,8)
Xy = data.frame(x = x, y = y)
summary(Xy)
```

```
##        x            y        
##  Min.   : 7   Min.   : 6.00  
##  1st Qu.: 7   1st Qu.: 7.25  
##  Median : 9   Median : 8.00
##  Mean   : 9   Mean   : 8.00
##  3rd Qu.:11   3rd Qu.: 8.75
##  Max.   :11   Max.   :10.00
```

```
lmodel = lm(y~x)
summary(lmodel)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##          1          2          3          4          5          6
## -1.000e+00  6.106e-16  1.000e+00  1.000e+00 -8.882e-16 -1.000e+00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.5000     1.8819   1.860   0.1364
## x             0.5000     0.2041   2.449   0.0705 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1 on 4 degrees of freedom
## Multiple R-squared:    0.6,  Adjusted R-squared:    0.5
## F-statistic:     6 on 1 and 4 DF,  p-value: 0.07048
```

Create a dataset $\mathbb{D}$ which we call `Xy` such that the linear model as R^2 about 0% but x, y are clearly associated.

```
x = c(1,2,3,4,5)
y = c(1,2,1,2,1)
Xy = data.frame(x = x, y = y)
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    1    2    3    4    5
## -0.4  0.6 -0.4  0.6 -0.4
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.400e+00  6.633e-01   2.111    0.125
## x           3.511e-17  2.000e-01   0.000    1.000
##
## Residual standard error: 0.6325 on 3 degrees of freedom
## Multiple R-squared:  1.643e-31,  Adjusted R-squared:  -0.3333
## F-statistic: 4.93e-31 on 1 and 3 DF,  p-value: 1
```

Load up the famous iris dataset and drop the data for Species “virginica”.

```
#TO-DO
data(iris)
iris=as.data.frame(iris[iris$Species != "virginica", ])
```

If the only input x is Species and you are trying to predict y which is Petal.Length, what would a reasonable, naive prediction be under both Species? Hint: it's what we did in class.

```
x = iris$Species
y = iris$Petal.Length
sumRefCat = 0
sumAltCat = 0
n=numeric()
for(i in 1:length(x)){
  if(x[i]=='setosa'){
    sumRefCat = sumRefCat + y[i]
    n=i
  }else{
      sumAltCat = sumAltCat + y[i]
    }

}
b_0 = sumRefCat/n
b_1 = sumAltCat/(length(x)-n) - b_0
#x = {0,1}
#y = b_0 + b_1 * x
```

Prove that this is the OLS model by fitting an appropriate `lm` and then using the predict function to verify you get the same answers as you wrote previously.

```
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.260 -0.162  0.038  0.238  0.840
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.46200    0.05010   29.18   <2e-16 ***
## xversicolor   2.79800    0.07085   39.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3542 on 98 degrees of freedom
## Multiple R-squared:  0.9409, Adjusted R-squared:  0.9403
## F-statistic:  1560 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
predict(lm(y~x))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    13    14    15    16    17    18    19    20    21    22    23    24
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    25    26    27    28    29    30    31    32    33    34    35    36
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    37    38    39    40    41    42    43    44    45    46    47    48
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    49    50    51    52    53    54    55    56    57    58    59    60
## 1.462 1.462 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    61    62    63    64    65    66    67    68    69    70    71    72
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    73    74    75    76    77    78    79    80    81    82    83    84
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    85    86    87    88    89    90    91    92    93    94    95    96
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    97    98    99   100
## 4.260 4.260 4.260 4.260
```