HW02p

Vyacheslav Takhalov

March 6, 2018

```
knitr::opts_chunk$set(error = TRUE) #this allows errors to be printed into the PDF
```

Welcome to HW02p where the "p" stands for "practice" meaning you will use R to solve practical problems. This homework is due 11:59 PM Tuesday 3/6/18.

You should have RStudio installed to edit this file. You will write code in places marked "TO-DO" to complete the problems. Some of this will be a pure programming assignment. Sometimes you will have to also write English.

The tools for the solutions to these problems can be found in the class practice lectures. I want you to use the methods I taught you, not for you to google and come up with whatever works. You won't learn that way.

To "hand in" the homework, you should compile or publish this file into a PDF that includes output of your code. To do so, use the knit menu in RStudio. You will need LaTeX installed on your computer. See the email announcement I sent out about this. Once it's done, push the PDF file to your github class repository by the deadline. You can choose to make this respository private.

For this homework, you will need the `testthat` libray.

```
pacman::p_load(testthat)
```

1. Source the simple dataset from lecture 6p:

```
Xy_simple = data.frame(
 response = factor(c(0, 0, 0, 1, 1, 1)), #nominal
 first_feature = c(1, 1, 2, 3, 3, 4),    #continuous
 second_feature = c(1, 2, 1, 3, 4, 3)    #continuous
)
X_simple_feature_matrix = as.matrix(Xy_simple[, 2 : 3])
y_binary = as.numeric(Xy_simple$response == 1)
```

Try your best to write a general perceptron learning algorithm to the following `Roxygen` spec. For inspiration, see the one I wrote in lecture 6.

```
#' This function implements the "perceptron learning algorithm" of Frank Rosenblatt (1957).
#'
#' @param Xinput      The training data features as an n x (p + 1) matrix where the first column is all
#' @param y_binary    The training data responses as a vector of length n consisting of only 0's and 1'
#' @param MAX_ITER    The maximum number of iterations the perceptron algorithm performs. Defaults to 1
#' @param w           A vector of length p + 1 specifying the parameter (weight) starting point. Defaul
#'                    \code{NULL} which means the function employs random standard uniform values.
#' @return            The computed final parameter (weight) as a vector of length p + 1
perceptron_learning_algorithm = function(Xinput, y_binary, MAX_ITER = 1000, w = NULL){
  if (is.null(w)){
    w = runif(ncol(Xinput)) #intialize a p+1-dim vector with random values
  }
  for (iter in 1 : MAX_ITER){
    for (i in 1 : nrow(Xinput)){
      x_i = Xinput[i, ]
      yhat_i = ifelse(x_i %*% w > 0, 1, 0)
      w = w + as.numeric(y_binary[i] - yhat_i) * x_i
```

```
    }
  }
  w
}
```

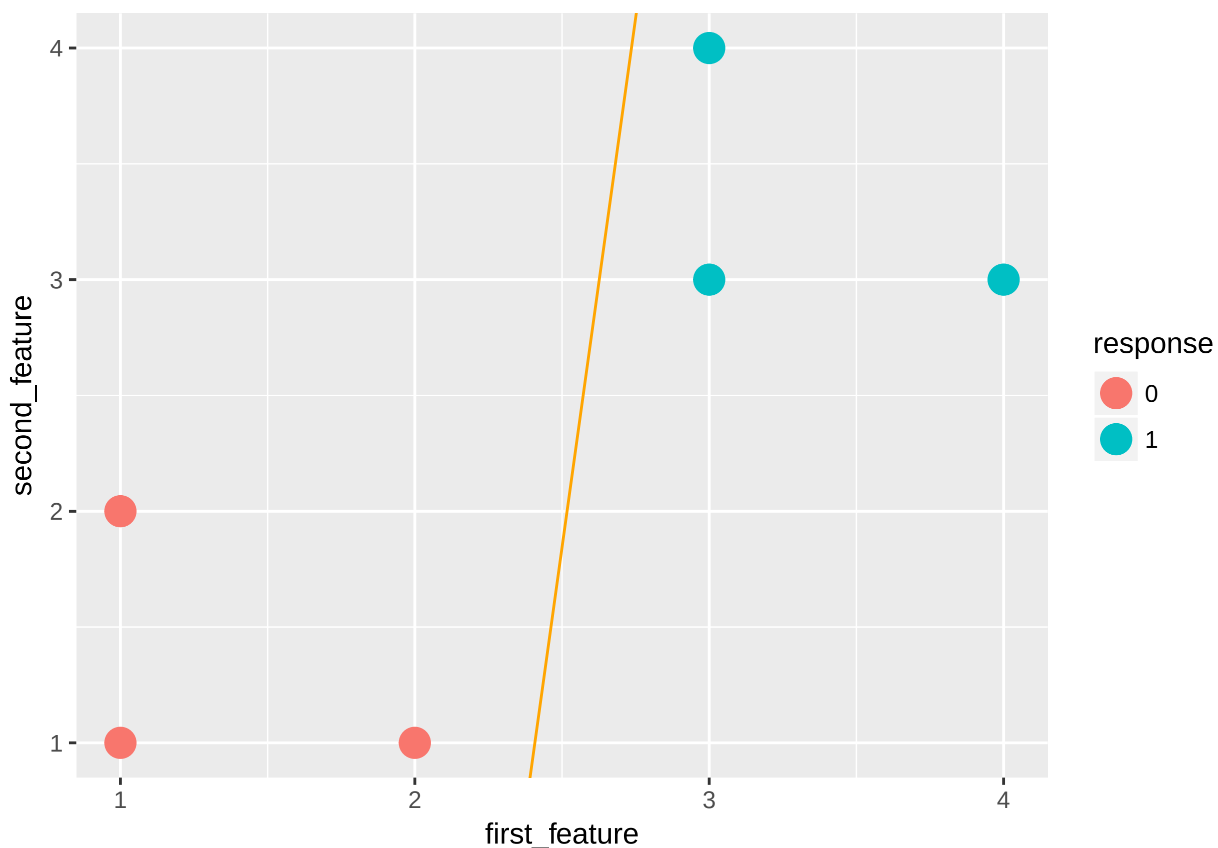
Run the code on the simple dataset above via:

```
w_vec_simple_per = perceptron_learning_algorithm(
  cbind(1, Xy_simple$first_feature, Xy_simple$second_feature),
  as.numeric(Xy_simple$response == 1))
w_vec_simple_per
```

```
## [1] -8.2016216  3.5684311 -0.3903183
```

Use the ggplot code to plot the data and the perceptron's g function.

```
pacman::p_load(ggplot2)
simple_viz_obj = ggplot(Xy_simple, aes(x = first_feature, y = second_feature, color = response)) +
  geom_point(size = 5)
simple_perceptron_line = geom_abline(
    intercept = -w_vec_simple_per[1] / w_vec_simple_per[3],
    slope = -w_vec_simple_per[2] / w_vec_simple_per[3],
    color = "orange")
simple_viz_obj + simple_perceptron_line
```



Why is this line of separation not "satisfying" to you?

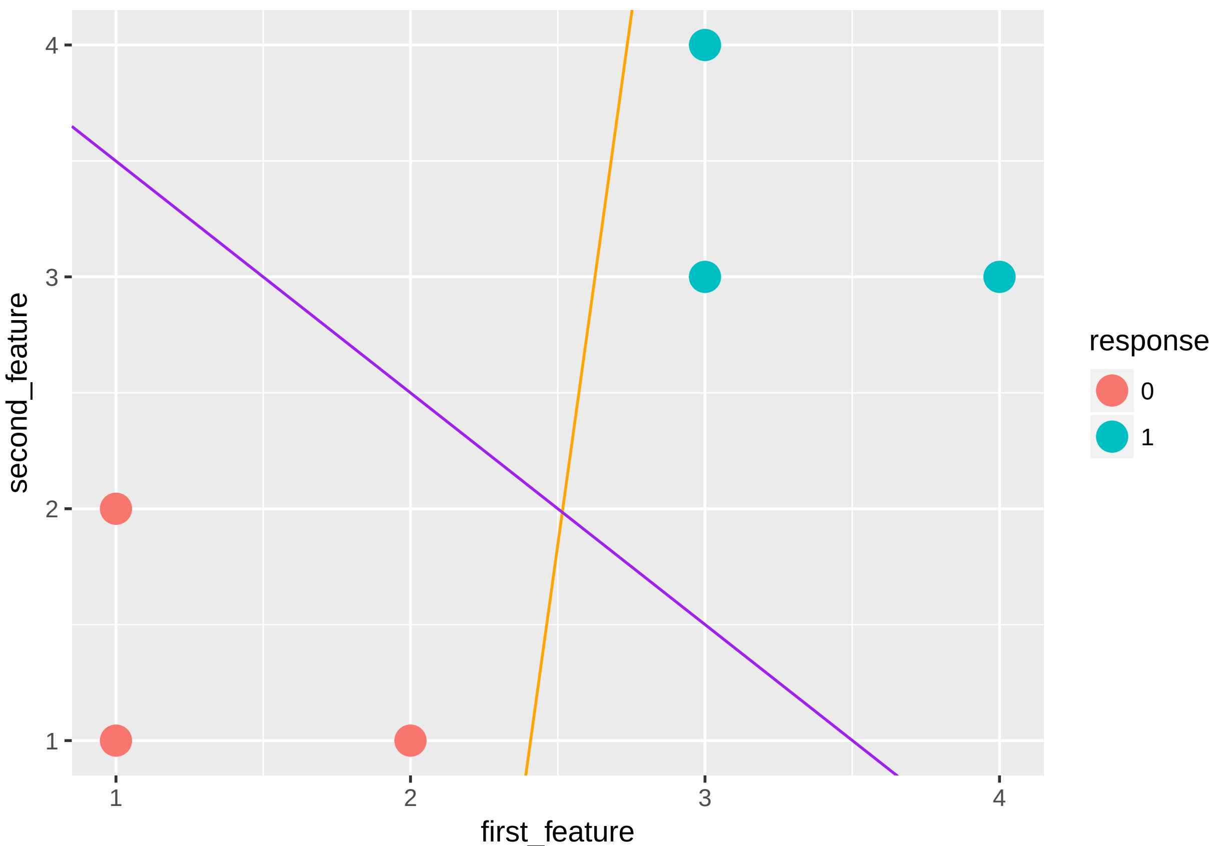
This line of separation is not "satisfying" to me because even though it separates the responses of 1s and 0s, there are better lines of separation. Plus there are many different lines that would separate the responses, but it is clear that a line like y = 4-x is what we are looking to produce. Using something like the SVM line would produce a more "satisfying" line.

2. Use the `e1071` package to fit an SVM model to `y_binary` using the predictors found in `X_simple_feature_matrix`. Do not specify the λ (i.e. do not specify the `cost` argument).

```
pacman::p_load(e1071)
Xy_simple_feature_matrix = as.matrix(Xy_simple[, 2 : 3])
lambda = 1e-9
n = nrow(Xy_simple_feature_matrix)
svm_model = svm(Xy_simple_feature_matrix, Xy_simple$response, kernel = "linear", scale = FALSE)
```

and then use the following code to visualize the line in purple:

```
w_vec_simple_svm = c(
  svm_model$rho, #the b term
  -t(svm_model$coefs) %*% X_simple_feature_matrix[svm_model$index, ] # the other terms
)
simple_svm_line = geom_abline(
    intercept = -w_vec_simple_svm[1] / w_vec_simple_svm[3],
    slope = -w_vec_simple_svm[2] / w_vec_simple_svm[3],
    color = "purple")
simple_viz_obj + simple_perceptron_line + simple_svm_line
```



Is this SVM line a better fit than the perceptron?

This SVM line is a better fit than the perceptron. Just as previously stated this SVM line looks more like the y = 4-x line we wanted to produce.

3. Now write pseudocode for your own implementation of the linear support vector machine algorithm respecting the following spec making use of the nelder mead `optim` function from lecture 5p. It turns out you do not need to load the package `neldermead` to use this function. You can feel free to define a function within this function if you wish.

Note there are differences between this spec and the perceptron learning algorithm spec in question #1. You should figure out a way to respect the `MAX_ITER` argument value.

For extra credit, write the actual code.

```
#' This function implements the hinge-loss + maximum margin linear support vector machine algorithm of
#'
#' @param Xinput      The training data features as an n x p matrix.
#' @param y_binary    The training data responses as a vector of length n consisting of only 0's and 1'
#' @param MAX_ITER    The maximum number of iterations the algorithm performs. Defaults to 5000.
#' @param lambda      A scalar hyperparameter trading off margin of the hyperplane versus average hinge
#'                    The default value is 1.
#' @return            The computed final parameter (weight) as a vector of length p + 1
linear_svm_learning_algorithm = function(Xinput, y_binary, MAX_ITER = 5000, lambda = 0.1){
  #TO-DO
}
```

If you wrote code (the extra credit), run your function using the defaults and plot it in brown vis-a-vis the previous model's line:

```
svm_model_weights = linear_svm_learning_algorithm(X_simple_feature_matrix, y_binary)
my_svm_line = geom_abline(
    intercept = svm_model_weights[1] / svm_model_weights[3],#NOTE: negative sign removed from intercept
    slope = -svm_model_weights[2] / svm_model_weights[3],
    color = "brown")
```

```
## Error in -svm_model_weights[2]: invalid argument to unary operator
```

```
simple_viz_obj  + my_svm_line
```

```
## Error in eval(expr, envir, enclos): object 'my_svm_line' not found
```

Is this the same as what the `e1071` implementation returned? Why or why not?

4. Write a $k = 1$ nearest neighbor algorithm using the Euclidean distance function. Respect the spec below:

```
#' This function implements the nearest neighbor algorithm.
#'
#' @param Xinput      The training data features as an n x p matrix.
#' @param y_binary    The training data responses as a vector of length n consisting of only 0's and 1'
#' @param Xtest       The test data that the algorithm will predict on as a n* x p matrix.
#' @return            The predictions as a n* length vector.
nn_algorithm_predict = function(Xinput, y_binary, Xtest){
  #TO-DO
}
```

Write a few tests to ensure it actually works:

```
#TO-DO
```

For extra credit, add an argument `k` to the `nn_algorithm_predict` function and update the implementation so it performs KNN. In the case of a tie, choose $\hat{y}$ randomly. Set the default `k` to be the square root of the size of $\mathcal{D}$ which is an empirical rule-of-thumb popularized by the "Pattern Classification" book by Duda, Hart and Stork (2007). Also, alter the documentation in the appropriate places.

```
#not required TO-DO --- only for extra credit
```

For extra credit, in addition to the argument `k`, add an argument `d` representing any legal distance function to the `nn_algorithm_predict` function. Update the implementation so it performs KNN using that distance function. Set the default function to be the Euclidean distance in the original function. Also, alter the documentation in the appropriate places.

```
#not required TO-DO --- only for extra credit
```

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

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

```
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 `cor`, `cov`, `var`, `sd` but instead computing it from the x and y quantities manually. See the class notes.

```
Sumx = 0
for(i in 1 : 20){
  Sumx = Sumx + (x[i]-mean(x))^2
}
Sx2 = (1/19) * Sumx
Sumy = 0
for(i in 1 : 20){
  Sumy = Sumy + (y[i]-mean(y))^2
}
Sy2 = (1/19) * Sumy
Sy = sqrt(Sy2)
Sumxy = 0
for(i in 1 : 20){
  Sumxy = Sumxy + (x[i]-mean(x)) * (y[i]-mean(y))
}
Sxy = Sumxy/19
b_1 = Sxy/Sx2
b_0 = mean(y) - b_1 * mean(x)
```

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

```
?lm
```

```
## starting httpd help server ... done
```

```
lm_mod = lm(y~x)
b_vec = coef(lm_mod)
expect_equal(b_0, as.numeric(b_vec[1]), tol = 1e-4) #thanks to Rachel for spotting this bug - the b_vec
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`.

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

In it, there is a dataset called `Galton`. Load it using the `data` command:

```r
data(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
?Galton
Galton
```

```
##     parent child
## 1     70.5  61.7
## 2     68.5  61.7
## 3     65.5  61.7
## 4     64.5  61.7
## 5     64.0  61.7
## 6     67.5  62.2
## 7     67.5  62.2
## 8     67.5  62.2
## 9     66.5  62.2
## 10    66.5  62.2
## 11    66.5  62.2
## 12    64.5  62.2
## 13    70.5  63.2
## 14    69.5  63.2
## 15    68.5  63.2
## 16    68.5  63.2
## 17    68.5  63.2
## 18    68.5  63.2
## 19    68.5  63.2
## 20    68.5  63.2
## 21    68.5  63.2
## 22    67.5  63.2
## 23    67.5  63.2
## 24    67.5  63.2
## 25    67.5  63.2
## 26    67.5  63.2
## 27    66.5  63.2
## 28    66.5  63.2
## 29    66.5  63.2
## 30    65.5  63.2
## 31    65.5  63.2
## 32    65.5  63.2
## 33    65.5  63.2
## 34    65.5  63.2
## 35    65.5  63.2
## 36    65.5  63.2
## 37    65.5  63.2
## 38    65.5  63.2
## 39    64.5  63.2
## 40    64.5  63.2
## 41    64.5  63.2
## 42    64.5  63.2
## 43    64.0  63.2
## 44    64.0  63.2
## 45    69.5  64.2
```

```
## 46    69.5  64.2
## 47    69.5  64.2
## 48    69.5  64.2
## 49    69.5  64.2
## 50    69.5  64.2
## 51    69.5  64.2
## 52    69.5  64.2
## 53    69.5  64.2
## 54    69.5  64.2
## 55    69.5  64.2
## 56    69.5  64.2
## 57    69.5  64.2
## 58    69.5  64.2
## 59    69.5  64.2
## 60    69.5  64.2
## 61    68.5  64.2
## 62    68.5  64.2
## 63    68.5  64.2
## 64    68.5  64.2
## 65    68.5  64.2
## 66    68.5  64.2
## 67    68.5  64.2
## 68    68.5  64.2
## 69    68.5  64.2
## 70    68.5  64.2
## 71    68.5  64.2
## 72    67.5  64.2
## 73    67.5  64.2
## 74    67.5  64.2
## 75    67.5  64.2
## 76    67.5  64.2
## 77    67.5  64.2
## 78    67.5  64.2
## 79    67.5  64.2
## 80    67.5  64.2
## 81    67.5  64.2
## 82    67.5  64.2
## 83    67.5  64.2
## 84    67.5  64.2
## 85    67.5  64.2
## 86    66.5  64.2
## 87    66.5  64.2
## 88    66.5  64.2
## 89    66.5  64.2
## 90    66.5  64.2
## 91    65.5  64.2
## 92    65.5  64.2
## 93    65.5  64.2
## 94    65.5  64.2
## 95    65.5  64.2
## 96    64.5  64.2
## 97    64.5  64.2
## 98    64.5  64.2
## 99    64.5  64.2
```

```
## 100   64.0  64.2
## 101   64.0  64.2
## 102   64.0  64.2
## 103   64.0  64.2
## 104   71.5  65.2
## 105   70.5  65.2
## 106   69.5  65.2
## 107   69.5  65.2
## 108   69.5  65.2
## 109   69.5  65.2
## 110   68.5  65.2
## 111   68.5  65.2
## 112   68.5  65.2
## 113   68.5  65.2
## 114   68.5  65.2
## 115   68.5  65.2
## 116   68.5  65.2
## 117   68.5  65.2
## 118   68.5  65.2
## 119   68.5  65.2
## 120   68.5  65.2
## 121   68.5  65.2
## 122   68.5  65.2
## 123   68.5  65.2
## 124   68.5  65.2
## 125   68.5  65.2
## 126   67.5  65.2
## 127   67.5  65.2
## 128   67.5  65.2
## 129   67.5  65.2
## 130   67.5  65.2
## 131   67.5  65.2
## 132   67.5  65.2
## 133   67.5  65.2
## 134   67.5  65.2
## 135   67.5  65.2
## 136   67.5  65.2
## 137   67.5  65.2
## 138   67.5  65.2
## 139   67.5  65.2
## 140   67.5  65.2
## 141   66.5  65.2
## 142   66.5  65.2
## 143   65.5  65.2
## 144   65.5  65.2
## 145   65.5  65.2
## 146   65.5  65.2
## 147   65.5  65.2
## 148   65.5  65.2
## 149   65.5  65.2
## 150   64.5  65.2
## 151   64.0  65.2
## 152   71.5  66.2
## 153   71.5  66.2
```

```
## 154   71.5  66.2
## 155   70.5  66.2
## 156   69.5  66.2
## 157   69.5  66.2
## 158   69.5  66.2
## 159   69.5  66.2
## 160   69.5  66.2
## 161   69.5  66.2
## 162   69.5  66.2
## 163   69.5  66.2
## 164   69.5  66.2
## 165   69.5  66.2
## 166   69.5  66.2
## 167   69.5  66.2
## 168   69.5  66.2
## 169   69.5  66.2
## 170   69.5  66.2
## 171   69.5  66.2
## 172   69.5  66.2
## 173   68.5  66.2
## 174   68.5  66.2
## 175   68.5  66.2
## 176   68.5  66.2
## 177   68.5  66.2
## 178   68.5  66.2
## 179   68.5  66.2
## 180   68.5  66.2
## 181   68.5  66.2
## 182   68.5  66.2
## 183   68.5  66.2
## 184   68.5  66.2
## 185   68.5  66.2
## 186   68.5  66.2
## 187   68.5  66.2
## 188   68.5  66.2
## 189   68.5  66.2
## 190   68.5  66.2
## 191   68.5  66.2
## 192   68.5  66.2
## 193   68.5  66.2
## 194   68.5  66.2
## 195   68.5  66.2
## 196   68.5  66.2
## 197   68.5  66.2
## 198   67.5  66.2
## 199   67.5  66.2
## 200   67.5  66.2
## 201   67.5  66.2
## 202   67.5  66.2
## 203   67.5  66.2
## 204   67.5  66.2
## 205   67.5  66.2
## 206   67.5  66.2
## 207   67.5  66.2
```

```
## 208   67.5  66.2
## 209   67.5  66.2
## 210   67.5  66.2
## 211   67.5  66.2
## 212   67.5  66.2
## 213   67.5  66.2
## 214   67.5  66.2
## 215   67.5  66.2
## 216   67.5  66.2
## 217   67.5  66.2
## 218   67.5  66.2
## 219   67.5  66.2
## 220   67.5  66.2
## 221   67.5  66.2
## 222   67.5  66.2
## 223   67.5  66.2
## 224   67.5  66.2
## 225   67.5  66.2
## 226   67.5  66.2
## 227   67.5  66.2
## 228   67.5  66.2
## 229   67.5  66.2
## 230   67.5  66.2
## 231   67.5  66.2
## 232   67.5  66.2
## 233   67.5  66.2
## 234   66.5  66.2
## 235   66.5  66.2
## 236   66.5  66.2
## 237   66.5  66.2
## 238   66.5  66.2
## 239   66.5  66.2
## 240   66.5  66.2
## 241   66.5  66.2
## 242   66.5  66.2
## 243   66.5  66.2
## 244   66.5  66.2
## 245   66.5  66.2
## 246   66.5  66.2
## 247   66.5  66.2
## 248   66.5  66.2
## 249   66.5  66.2
## 250   66.5  66.2
## 251   65.5  66.2
## 252   65.5  66.2
## 253   65.5  66.2
## 254   65.5  66.2
## 255   65.5  66.2
## 256   65.5  66.2
## 257   65.5  66.2
## 258   65.5  66.2
## 259   65.5  66.2
## 260   65.5  66.2
## 261   65.5  66.2
```

```
## 262   64.5  66.2
## 263   64.5  66.2
## 264   64.5  66.2
## 265   64.5  66.2
## 266   64.5  66.2
## 267   64.0  66.2
## 268   64.0  66.2
## 269   71.5  67.2
## 270   71.5  67.2
## 271   71.5  67.2
## 272   71.5  67.2
## 273   70.5  67.2
## 274   70.5  67.2
## 275   70.5  67.2
## 276   69.5  67.2
## 277   69.5  67.2
## 278   69.5  67.2
## 279   69.5  67.2
## 280   69.5  67.2
## 281   69.5  67.2
## 282   69.5  67.2
## 283   69.5  67.2
## 284   69.5  67.2
## 285   69.5  67.2
## 286   69.5  67.2
## 287   69.5  67.2
## 288   69.5  67.2
## 289   69.5  67.2
## 290   69.5  67.2
## 291   69.5  67.2
## 292   69.5  67.2
## 293   69.5  67.2
## 294   69.5  67.2
## 295   69.5  67.2
## 296   69.5  67.2
## 297   69.5  67.2
## 298   69.5  67.2
## 299   69.5  67.2
## 300   69.5  67.2
## 301   69.5  67.2
## 302   69.5  67.2
## 303   68.5  67.2
## 304   68.5  67.2
## 305   68.5  67.2
## 306   68.5  67.2
## 307   68.5  67.2
## 308   68.5  67.2
## 309   68.5  67.2
## 310   68.5  67.2
## 311   68.5  67.2
## 312   68.5  67.2
## 313   68.5  67.2
## 314   68.5  67.2
## 315   68.5  67.2
```

```
## 316   68.5  67.2
## 317   68.5  67.2
## 318   68.5  67.2
## 319   68.5  67.2
## 320   68.5  67.2
## 321   68.5  67.2
## 322   68.5  67.2
## 323   68.5  67.2
## 324   68.5  67.2
## 325   68.5  67.2
## 326   68.5  67.2
## 327   68.5  67.2
## 328   68.5  67.2
## 329   68.5  67.2
## 330   68.5  67.2
## 331   68.5  67.2
## 332   68.5  67.2
## 333   68.5  67.2
## 334   67.5  67.2
## 335   67.5  67.2
## 336   67.5  67.2
## 337   67.5  67.2
## 338   67.5  67.2
## 339   67.5  67.2
## 340   67.5  67.2
## 341   67.5  67.2
## 342   67.5  67.2
## 343   67.5  67.2
## 344   67.5  67.2
## 345   67.5  67.2
## 346   67.5  67.2
## 347   67.5  67.2
## 348   67.5  67.2
## 349   67.5  67.2
## 350   67.5  67.2
## 351   67.5  67.2
## 352   67.5  67.2
## 353   67.5  67.2
## 354   67.5  67.2
## 355   67.5  67.2
## 356   67.5  67.2
## 357   67.5  67.2
## 358   67.5  67.2
## 359   67.5  67.2
## 360   67.5  67.2
## 361   67.5  67.2
## 362   67.5  67.2
## 363   67.5  67.2
## 364   67.5  67.2
## 365   67.5  67.2
## 366   67.5  67.2
## 367   67.5  67.2
## 368   67.5  67.2
## 369   67.5  67.2
```

```
## 370   67.5  67.2
## 371   67.5  67.2
## 372   66.5  67.2
## 373   66.5  67.2
## 374   66.5  67.2
## 375   66.5  67.2
## 376   66.5  67.2
## 377   66.5  67.2
## 378   66.5  67.2
## 379   66.5  67.2
## 380   66.5  67.2
## 381   66.5  67.2
## 382   66.5  67.2
## 383   66.5  67.2
## 384   66.5  67.2
## 385   66.5  67.2
## 386   66.5  67.2
## 387   66.5  67.2
## 388   66.5  67.2
## 389   65.5  67.2
## 390   65.5  67.2
## 391   65.5  67.2
## 392   65.5  67.2
## 393   65.5  67.2
## 394   65.5  67.2
## 395   65.5  67.2
## 396   65.5  67.2
## 397   65.5  67.2
## 398   65.5  67.2
## 399   65.5  67.2
## 400   64.5  67.2
## 401   64.5  67.2
## 402   64.5  67.2
## 403   64.5  67.2
## 404   64.5  67.2
## 405   64.0  67.2
## 406   64.0  67.2
## 407   72.5  68.2
## 408   71.5  68.2
## 409   71.5  68.2
## 410   71.5  68.2
## 411   70.5  68.2
## 412   70.5  68.2
## 413   70.5  68.2
## 414   70.5  68.2
## 415   70.5  68.2
## 416   70.5  68.2
## 417   70.5  68.2
## 418   70.5  68.2
## 419   70.5  68.2
## 420   70.5  68.2
## 421   70.5  68.2
## 422   70.5  68.2
## 423   69.5  68.2
```

```
## 424   69.5  68.2
## 425   69.5  68.2
## 426   69.5  68.2
## 427   69.5  68.2
## 428   69.5  68.2
## 429   69.5  68.2
## 430   69.5  68.2
## 431   69.5  68.2
## 432   69.5  68.2
## 433   69.5  68.2
## 434   69.5  68.2
## 435   69.5  68.2
## 436   69.5  68.2
## 437   69.5  68.2
## 438   69.5  68.2
## 439   69.5  68.2
## 440   69.5  68.2
## 441   69.5  68.2
## 442   69.5  68.2
## 443   68.5  68.2
## 444   68.5  68.2
## 445   68.5  68.2
## 446   68.5  68.2
## 447   68.5  68.2
## 448   68.5  68.2
## 449   68.5  68.2
## 450   68.5  68.2
## 451   68.5  68.2
## 452   68.5  68.2
## 453   68.5  68.2
## 454   68.5  68.2
## 455   68.5  68.2
## 456   68.5  68.2
## 457   68.5  68.2
## 458   68.5  68.2
## 459   68.5  68.2
## 460   68.5  68.2
## 461   68.5  68.2
## 462   68.5  68.2
## 463   68.5  68.2
## 464   68.5  68.2
## 465   68.5  68.2
## 466   68.5  68.2
## 467   68.5  68.2
## 468   68.5  68.2
## 469   68.5  68.2
## 470   68.5  68.2
## 471   68.5  68.2
## 472   68.5  68.2
## 473   68.5  68.2
## 474   68.5  68.2
## 475   68.5  68.2
## 476   68.5  68.2
## 477   67.5  68.2
```

```
## 478   67.5  68.2
## 479   67.5  68.2
## 480   67.5  68.2
## 481   67.5  68.2
## 482   67.5  68.2
## 483   67.5  68.2
## 484   67.5  68.2
## 485   67.5  68.2
## 486   67.5  68.2
## 487   67.5  68.2
## 488   67.5  68.2
## 489   67.5  68.2
## 490   67.5  68.2
## 491   67.5  68.2
## 492   67.5  68.2
## 493   67.5  68.2
## 494   67.5  68.2
## 495   67.5  68.2
## 496   67.5  68.2
## 497   67.5  68.2
## 498   67.5  68.2
## 499   67.5  68.2
## 500   67.5  68.2
## 501   67.5  68.2
## 502   67.5  68.2
## 503   67.5  68.2
## 504   67.5  68.2
## 505   66.5  68.2
## 506   66.5  68.2
## 507   66.5  68.2
## 508   66.5  68.2
## 509   66.5  68.2
## 510   66.5  68.2
## 511   66.5  68.2
## 512   66.5  68.2
## 513   66.5  68.2
## 514   66.5  68.2
## 515   66.5  68.2
## 516   66.5  68.2
## 517   66.5  68.2
## 518   66.5  68.2
## 519   65.5  68.2
## 520   65.5  68.2
## 521   65.5  68.2
## 522   65.5  68.2
## 523   65.5  68.2
## 524   65.5  68.2
## 525   65.5  68.2
## 526   64.0  68.2
## 527   72.5  69.2
## 528   72.5  69.2
## 529   71.5  69.2
## 530   71.5  69.2
## 531   71.5  69.2
```

```
## 532   71.5  69.2
## 533   71.5  69.2
## 534   70.5  69.2
## 535   70.5  69.2
## 536   70.5  69.2
## 537   70.5  69.2
## 538   70.5  69.2
## 539   70.5  69.2
## 540   70.5  69.2
## 541   70.5  69.2
## 542   70.5  69.2
## 543   70.5  69.2
## 544   70.5  69.2
## 545   70.5  69.2
## 546   70.5  69.2
## 547   70.5  69.2
## 548   70.5  69.2
## 549   70.5  69.2
## 550   70.5  69.2
## 551   70.5  69.2
## 552   69.5  69.2
## 553   69.5  69.2
## 554   69.5  69.2
## 555   69.5  69.2
## 556   69.5  69.2
## 557   69.5  69.2
## 558   69.5  69.2
## 559   69.5  69.2
## 560   69.5  69.2
## 561   69.5  69.2
## 562   69.5  69.2
## 563   69.5  69.2
## 564   69.5  69.2
## 565   69.5  69.2
## 566   69.5  69.2
## 567   69.5  69.2
## 568   69.5  69.2
## 569   69.5  69.2
## 570   69.5  69.2
## 571   69.5  69.2
## 572   69.5  69.2
## 573   69.5  69.2
## 574   69.5  69.2
## 575   69.5  69.2
## 576   69.5  69.2
## 577   69.5  69.2
## 578   69.5  69.2
## 579   69.5  69.2
## 580   69.5  69.2
## 581   69.5  69.2
## 582   69.5  69.2
## 583   69.5  69.2
## 584   69.5  69.2
## 585   68.5  69.2
```

```
## 586   68.5  69.2
## 587   68.5  69.2
## 588   68.5  69.2
## 589   68.5  69.2
## 590   68.5  69.2
## 591   68.5  69.2
## 592   68.5  69.2
## 593   68.5  69.2
## 594   68.5  69.2
## 595   68.5  69.2
## 596   68.5  69.2
## 597   68.5  69.2
## 598   68.5  69.2
## 599   68.5  69.2
## 600   68.5  69.2
## 601   68.5  69.2
## 602   68.5  69.2
## 603   68.5  69.2
## 604   68.5  69.2
## 605   68.5  69.2
## 606   68.5  69.2
## 607   68.5  69.2
## 608   68.5  69.2
## 609   68.5  69.2
## 610   68.5  69.2
## 611   68.5  69.2
## 612   68.5  69.2
## 613   68.5  69.2
## 614   68.5  69.2
## 615   68.5  69.2
## 616   68.5  69.2
## 617   68.5  69.2
## 618   68.5  69.2
## 619   68.5  69.2
## 620   68.5  69.2
## 621   68.5  69.2
## 622   68.5  69.2
## 623   68.5  69.2
## 624   68.5  69.2
## 625   68.5  69.2
## 626   68.5  69.2
## 627   68.5  69.2
## 628   68.5  69.2
## 629   68.5  69.2
## 630   68.5  69.2
## 631   68.5  69.2
## 632   68.5  69.2
## 633   67.5  69.2
## 634   67.5  69.2
## 635   67.5  69.2
## 636   67.5  69.2
## 637   67.5  69.2
## 638   67.5  69.2
## 639   67.5  69.2
```

```
## 640   67.5  69.2
## 641   67.5  69.2
## 642   67.5  69.2
## 643   67.5  69.2
## 644   67.5  69.2
## 645   67.5  69.2
## 646   67.5  69.2
## 647   67.5  69.2
## 648   67.5  69.2
## 649   67.5  69.2
## 650   67.5  69.2
## 651   67.5  69.2
## 652   67.5  69.2
## 653   67.5  69.2
## 654   67.5  69.2
## 655   67.5  69.2
## 656   67.5  69.2
## 657   67.5  69.2
## 658   67.5  69.2
## 659   67.5  69.2
## 660   67.5  69.2
## 661   67.5  69.2
## 662   67.5  69.2
## 663   67.5  69.2
## 664   67.5  69.2
## 665   67.5  69.2
## 666   67.5  69.2
## 667   67.5  69.2
## 668   67.5  69.2
## 669   67.5  69.2
## 670   67.5  69.2
## 671   66.5  69.2
## 672   66.5  69.2
## 673   66.5  69.2
## 674   66.5  69.2
## 675   66.5  69.2
## 676   66.5  69.2
## 677   66.5  69.2
## 678   66.5  69.2
## 679   66.5  69.2
## 680   66.5  69.2
## 681   66.5  69.2
## 682   66.5  69.2
## 683   66.5  69.2
## 684   65.5  69.2
## 685   65.5  69.2
## 686   65.5  69.2
## 687   65.5  69.2
## 688   65.5  69.2
## 689   65.5  69.2
## 690   65.5  69.2
## 691   64.5  69.2
## 692   64.5  69.2
## 693   64.0  69.2
```

```
## 694   72.5  70.2
## 695   71.5  70.2
## 696   71.5  70.2
## 697   71.5  70.2
## 698   71.5  70.2
## 699   71.5  70.2
## 700   71.5  70.2
## 701   71.5  70.2
## 702   71.5  70.2
## 703   71.5  70.2
## 704   71.5  70.2
## 705   70.5  70.2
## 706   70.5  70.2
## 707   70.5  70.2
## 708   70.5  70.2
## 709   70.5  70.2
## 710   70.5  70.2
## 711   70.5  70.2
## 712   70.5  70.2
## 713   70.5  70.2
## 714   70.5  70.2
## 715   70.5  70.2
## 716   70.5  70.2
## 717   70.5  70.2
## 718   70.5  70.2
## 719   69.5  70.2
## 720   69.5  70.2
## 721   69.5  70.2
## 722   69.5  70.2
## 723   69.5  70.2
## 724   69.5  70.2
## 725   69.5  70.2
## 726   69.5  70.2
## 727   69.5  70.2
## 728   69.5  70.2
## 729   69.5  70.2
## 730   69.5  70.2
## 731   69.5  70.2
## 732   69.5  70.2
## 733   69.5  70.2
## 734   69.5  70.2
## 735   69.5  70.2
## 736   69.5  70.2
## 737   69.5  70.2
## 738   69.5  70.2
## 739   69.5  70.2
## 740   69.5  70.2
## 741   69.5  70.2
## 742   69.5  70.2
## 743   69.5  70.2
## 744   68.5  70.2
## 745   68.5  70.2
## 746   68.5  70.2
## 747   68.5  70.2
```

```
## 748   68.5  70.2
## 749   68.5  70.2
## 750   68.5  70.2
## 751   68.5  70.2
## 752   68.5  70.2
## 753   68.5  70.2
## 754   68.5  70.2
## 755   68.5  70.2
## 756   68.5  70.2
## 757   68.5  70.2
## 758   68.5  70.2
## 759   68.5  70.2
## 760   68.5  70.2
## 761   68.5  70.2
## 762   68.5  70.2
## 763   68.5  70.2
## 764   68.5  70.2
## 765   67.5  70.2
## 766   67.5  70.2
## 767   67.5  70.2
## 768   67.5  70.2
## 769   67.5  70.2
## 770   67.5  70.2
## 771   67.5  70.2
## 772   67.5  70.2
## 773   67.5  70.2
## 774   67.5  70.2
## 775   67.5  70.2
## 776   67.5  70.2
## 777   67.5  70.2
## 778   67.5  70.2
## 779   67.5  70.2
## 780   67.5  70.2
## 781   67.5  70.2
## 782   67.5  70.2
## 783   67.5  70.2
## 784   66.5  70.2
## 785   66.5  70.2
## 786   66.5  70.2
## 787   66.5  70.2
## 788   65.5  70.2
## 789   65.5  70.2
## 790   65.5  70.2
## 791   65.5  70.2
## 792   65.5  70.2
## 793   72.5  71.2
## 794   72.5  71.2
## 795   71.5  71.2
## 796   71.5  71.2
## 797   71.5  71.2
## 798   71.5  71.2
## 799   70.5  71.2
## 800   70.5  71.2
## 801   70.5  71.2
```

```
## 802   70.5  71.2
## 803   70.5  71.2
## 804   70.5  71.2
## 805   70.5  71.2
## 806   69.5  71.2
## 807   69.5  71.2
## 808   69.5  71.2
## 809   69.5  71.2
## 810   69.5  71.2
## 811   69.5  71.2
## 812   69.5  71.2
## 813   69.5  71.2
## 814   69.5  71.2
## 815   69.5  71.2
## 816   69.5  71.2
## 817   69.5  71.2
## 818   69.5  71.2
## 819   69.5  71.2
## 820   69.5  71.2
## 821   69.5  71.2
## 822   69.5  71.2
## 823   69.5  71.2
## 824   69.5  71.2
## 825   69.5  71.2
## 826   68.5  71.2
## 827   68.5  71.2
## 828   68.5  71.2
## 829   68.5  71.2
## 830   68.5  71.2
## 831   68.5  71.2
## 832   68.5  71.2
## 833   68.5  71.2
## 834   68.5  71.2
## 835   68.5  71.2
## 836   68.5  71.2
## 837   68.5  71.2
## 838   68.5  71.2
## 839   68.5  71.2
## 840   68.5  71.2
## 841   68.5  71.2
## 842   68.5  71.2
## 843   68.5  71.2
## 844   67.5  71.2
## 845   67.5  71.2
## 846   67.5  71.2
## 847   67.5  71.2
## 848   67.5  71.2
## 849   67.5  71.2
## 850   67.5  71.2
## 851   67.5  71.2
## 852   67.5  71.2
## 853   67.5  71.2
## 854   67.5  71.2
## 855   65.5  71.2
```

```
## 856   65.5  71.2
## 857   73.0  72.2
## 858   72.5  72.2
## 859   72.5  72.2
## 860   72.5  72.2
## 861   72.5  72.2
## 862   72.5  72.2
## 863   72.5  72.2
## 864   72.5  72.2
## 865   71.5  72.2
## 866   71.5  72.2
## 867   71.5  72.2
## 868   71.5  72.2
## 869   71.5  72.2
## 870   71.5  72.2
## 871   71.5  72.2
## 872   71.5  72.2
## 873   71.5  72.2
## 874   70.5  72.2
## 875   70.5  72.2
## 876   70.5  72.2
## 877   70.5  72.2
## 878   69.5  72.2
## 879   69.5  72.2
## 880   69.5  72.2
## 881   69.5  72.2
## 882   69.5  72.2
## 883   69.5  72.2
## 884   69.5  72.2
## 885   69.5  72.2
## 886   69.5  72.2
## 887   69.5  72.2
## 888   69.5  72.2
## 889   68.5  72.2
## 890   68.5  72.2
## 891   68.5  72.2
## 892   68.5  72.2
## 893   67.5  72.2
## 894   67.5  72.2
## 895   67.5  72.2
## 896   67.5  72.2
## 897   65.5  72.2
## 898   73.0  73.2
## 899   73.0  73.2
## 900   73.0  73.2
## 901   72.5  73.2
## 902   72.5  73.2
## 903   71.5  73.2
## 904   71.5  73.2
## 905   70.5  73.2
## 906   70.5  73.2
## 907   70.5  73.2
## 908   69.5  73.2
## 909   69.5  73.2
```

```
## 910   69.5  73.2
## 911   69.5  73.2
## 912   68.5  73.2
## 913   68.5  73.2
## 914   68.5  73.2
## 915   72.5  73.7
## 916   72.5  73.7
## 917   72.5  73.7
## 918   72.5  73.7
## 919   71.5  73.7
## 920   71.5  73.7
## 921   70.5  73.7
## 922   70.5  73.7
## 923   70.5  73.7
## 924   69.5  73.7
## 925   69.5  73.7
## 926   69.5  73.7
## 927   69.5  73.7
## 928   69.5  73.7
```

Galton is a matrix with 2 columns named "parent" and "child" and 928 rows with observations. Parent is a numeric vector with entries heights of the mid-parent. Child is a numeric vector with heights of the children. n = 928 p = 2 Data is in inches and the values of female children are multiplied by 1.08 Most Children were shorter than the mid-parent in the lower rows around rows 1-400. Afterwards in the rest of the rows the children in the data are recorded to be taller than the mid-parent.

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

```
colSums(Galton)
```

```
##  parent   child
## 63390.0 63186.1
```

```
colSums(Galton)[1] + colSums(Galton)[2]
```

```
##   parent
## 126576.1
```

```
avg_height = (colSums(Galton)[1] + colSums(Galton)[2]) / (928 + 928)
avg_height
```

```
##   parent
## 68.19833
```

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.

```
x = Galton$parent
y = Galton$child

r = cor(x, y)
s_x = sd(x)
s_y = sd(y)
ybar = mean(y)
```

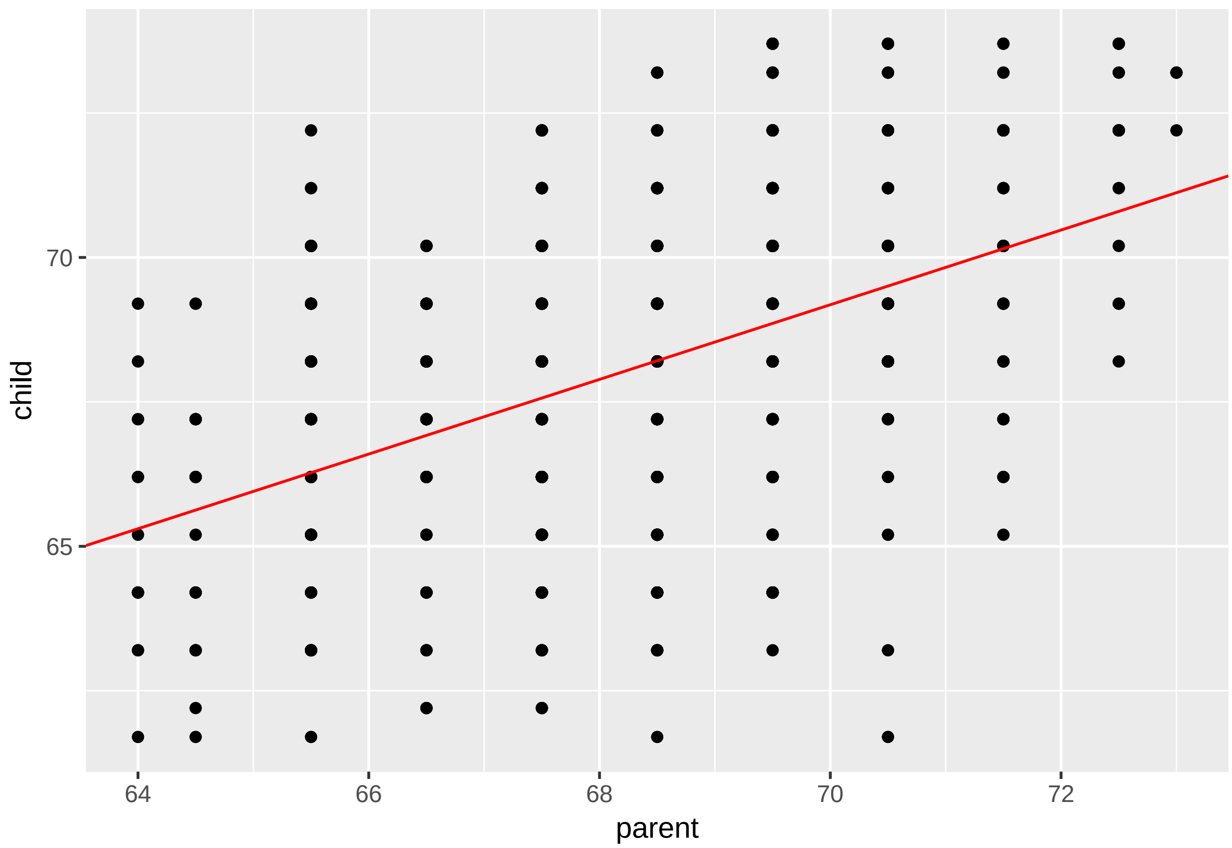
```r
xbar = mean(x)

b_1 = r * s_y / s_x
b_0 = ybar - b_1 * xbar
b_0
```

```
## [1] 23.94153
```

```r
b_1
```

```
## [1] 0.6462906
```

```r
simple_viz_obj = ggplot(Galton, aes(x = parent, y = child)) + geom_point()
simple_ls_regression_line = geom_abline(intercept = b_0, slope = b_1, color = "red")
simple_viz_obj + simple_ls_regression_line
```




```r
yhat = b_0 + b_1 * x #this is the g(x^*) function!
e = y - yhat
sse = sum(e^2)
mse = sse / length(y)
rmse = sqrt(mse)
sse
```

```
## [1] 4640.273
```

```r
mse
```

```
## [1] 5.000294
```

```r
rmse
```

```
## [1] 2.236134
```

```r
s_sq_y = var(y)
s_sq_e = var(e)
rsq = (s_sq_y - s_sq_e) / s_sq_y
rsq
```

```
## [1] 0.2104629
```

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

b__0 is the y-intercept and b__1 is the slope. As the height of the parent increases so does the height of the child, this is why the slope is positive. Since there can be no negative height our intecept is positive RMSE is about 2.2 so a child will be plus or minus 4.4 inches as compared to the parent. This is a pretty bad RMSE since 4.4 inches is a huge amount when considering height. As a result of this R^2 should be bad, and it is since it is about 21%

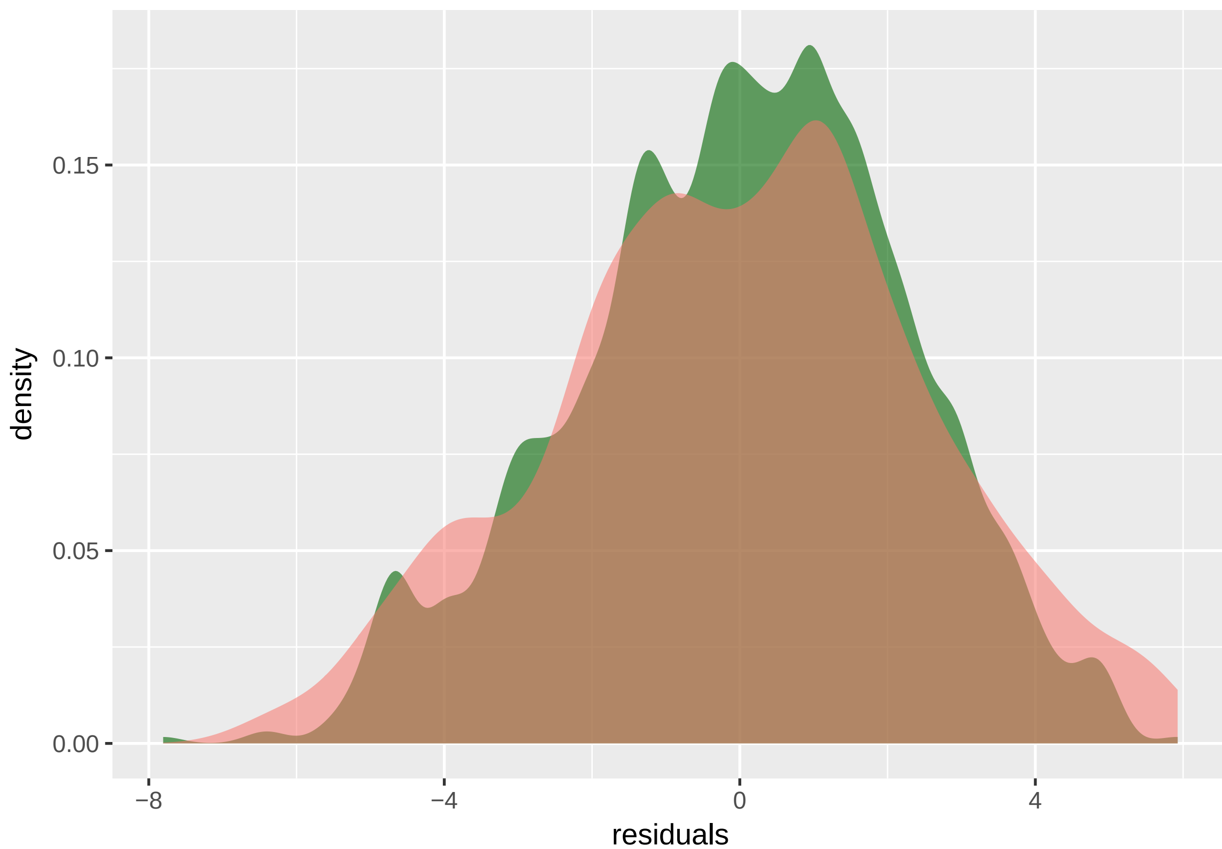
How good is this model? How well does it predict? Discuss.

This is a bad model becuase the height difference given the error +/-4.4 inches is too large. There are multiple sources of error. This will not predict well.

Now use the code from practice lecture 8 to plot the data and a best fit line using package `ggplot2`. Don't forget to load the library.

```r
Galton$null_residuals = y - mean(y)
Galton$residuals = e

ggplot(Galton) +
  stat_density(aes(x = residuals), fill = "darkgreen", alpha = 0.6, adjust = 0.5) +
  stat_density(aes(x = null_residuals, fill = "red", alpha = 0.6, adjust = 0.5)) +
  theme(legend.position = "none")
```

```
## Warning: Ignoring unknown aesthetics: adjust
```



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

This is a reasonable assumption. We would expect short parents creating short children, medium height parents creating medium children, and tall parents creating tall children. Basic biology would say that the height of the parents will reflect the height of the children. So this is reasonable

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?

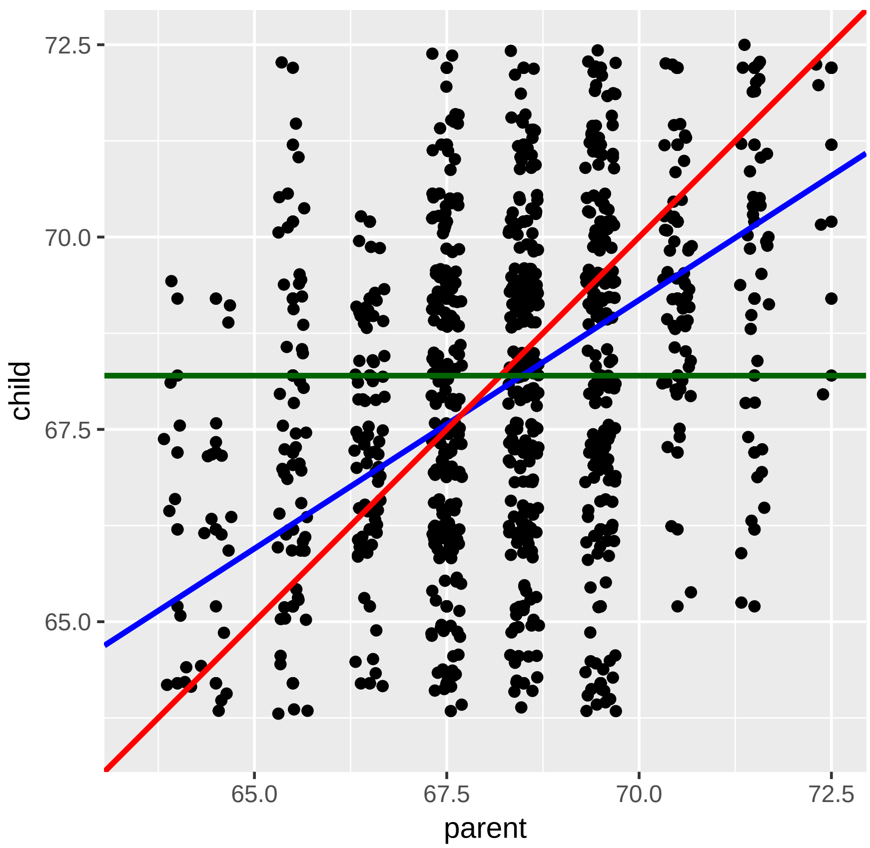
beta_0, the intercept, would be 0 and beta_1, the slope, would be 1

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.

```
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 88 rows containing missing values (geom_point).
```



Fill in the following sentence:

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

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

On average the hereditary stature was the same for the child as it was for the parent. Therefore the average height is the mean and the data tells us the output will also "regress" to this mean. Even if there is an extreme case with an extreme height, probably the next height will be average.

Why should this effect be real?

This effect should be real because of the law of large numbers. As more people are born the more chances there are to have average height, and the less chances there are of extreme heights.

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.

It is called "regression" because extreme data points tend to go back ("regress") to the average value of y. A more approprite name would be average y or y bar.