Homework1 - Machine Learning

Kamal

Homework 1

Your name here

Kun Li

Some informative viewing

I highly recommend watching this series by 3blue1brown to get a sense of what matrix operations are from a vector space perspective and how to visualize them: Essence of Linear Algebra (https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab) . So far, we will cover the first 4 videos in the series, the rest are enrichment …or review if you've already done statistics. We might cover topics like convolution and polynomial multiplication when we get to deep learning, which are a bit different from the vector representation of matrices. The foundation you build here will be helpful then.

Matrix Multiplication (2.5pts)

You are a linear algebra expert from Youtube university and have been hired by a local space cadet to help them navigate some vector spaces. They have provided you with the following matrices:

```
A <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, byrow = TRUE)
B <- matrix(c(7, 8, 9, 10, 11, 12), nrow = 3, byrow = TRUE)
```

1. Compute the matrix product $C = AB$. What are the dimensions of C? (0.5pts)

```
C <- A %*% B
dim(C)
```

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

```
C
```

```
##      [,1] [,2]
## [1,]   58   64
## [2,]  139  154
```

2. Compute the outer product of the first column of A with the first row of B. What is the result? (0.5pts)

```
outer_product <- A[, 1] %*% t(B[1, ])
outer_product
```

```
##      [,1] [,2]
## [1,]    7    8
## [2,]   28   32
```

3. Verify that the matrix product C can be expressed as a sum of outer products of the columns of A and the rows of B. (0.5pts)

```
C_outer_sum <- A[, 1] %*% t(B[1, ]) + A[, 2] %*% t(B[2, ]) + A[, 3] %*% t(B[3, ]) # s
um of outer products
all.equal(C, C_outer_sum) # TRUE proves they are equal
```

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

```
C_outer_sum
```

```
##      [,1] [,2]
## [1,]   58   64
## [2,]  139  154
```

4. Explain in your own words the relationship between matrix multiplication and outer products. (0.5pts) Matrix multiplication can be decomposed into the sum of outer products created by the rows of the matrix. Each outer product represents the contribution of a specific column of the first matrix and a specific row of the second matrix to the overall product.

5. Implement a function that takes two matrices as input and returns their product using the outer product method. (0.5pts)

```
matrix_multiply_outer <- function(A, B) {
  if (ncol(A) != nrow(B)) {
    stop("Incompatible dimensions for matrix multiplication") # check dim
  }
  result <- matrix(0, nrow = nrow(A), ncol = ncol(B))
  for (i in 1:ncol(A)) {
    result <- result + A[, i] %*% t(B[i, ])
  }
  return(C)

}
matrix_multiply_outer(A, B)
```

```
##      [,1] [,2]
## [1,]   58   64
## [2,]  139  154
```

Chick-Weight (7.5pts)

You are a farmer named Bob and your chickens are getting too fat. Analyze the ChickWeight dataset in R before your chickens get taken away by the ASPCA. Your task is to explore the relationship between diet and weight gain over time in chicks. Perform the following analyses:

Load the ChickWeight dataset and display its structure. If it's not already included, you can load the attached chick_weight.csv

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(ggcorrplot)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data("ChickWeight")
str(ChickWeight)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   578 o
bs. of  4 variables:
##  $ weight: num  42 51 59 64 76 93 106 125 149 171 ...
##  $ Time  : num  0 2 4 6 8 10 12 14 16 18 ...
##  $ Chick : Ord.factor w/ 50 levels "18"<"16"<"15"<..: 15 15 15 15 15 15 15 15 15 1
5 ...
##  $ Diet  : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "formula")=Class 'formula'  language weight ~ Time | Chick
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##  - attr(*, "outer")=Class 'formula'  language ~Diet
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##  - attr(*, "labels")=List of 2
##   ..$ x: chr "Time"
##   ..$ y: chr "Body weight"
##  - attr(*, "units")=List of 2
##   ..$ x: chr "(days)"
##   ..$ y: chr "(gm)"
```

```
head(ChickWeight)
```

```
##   weight Time Chick Diet
## 1     42    0     1    1
## 2     51    2     1    1
## 3     59    4     1    1
## 4     64    6     1    1
## 5     76    8     1    1
## 6     93   10     1    1
```

```
# If you need to load from a CSV file, uncomment the line below
ChickWeight <- read.csv("chick_weight.csv")
```

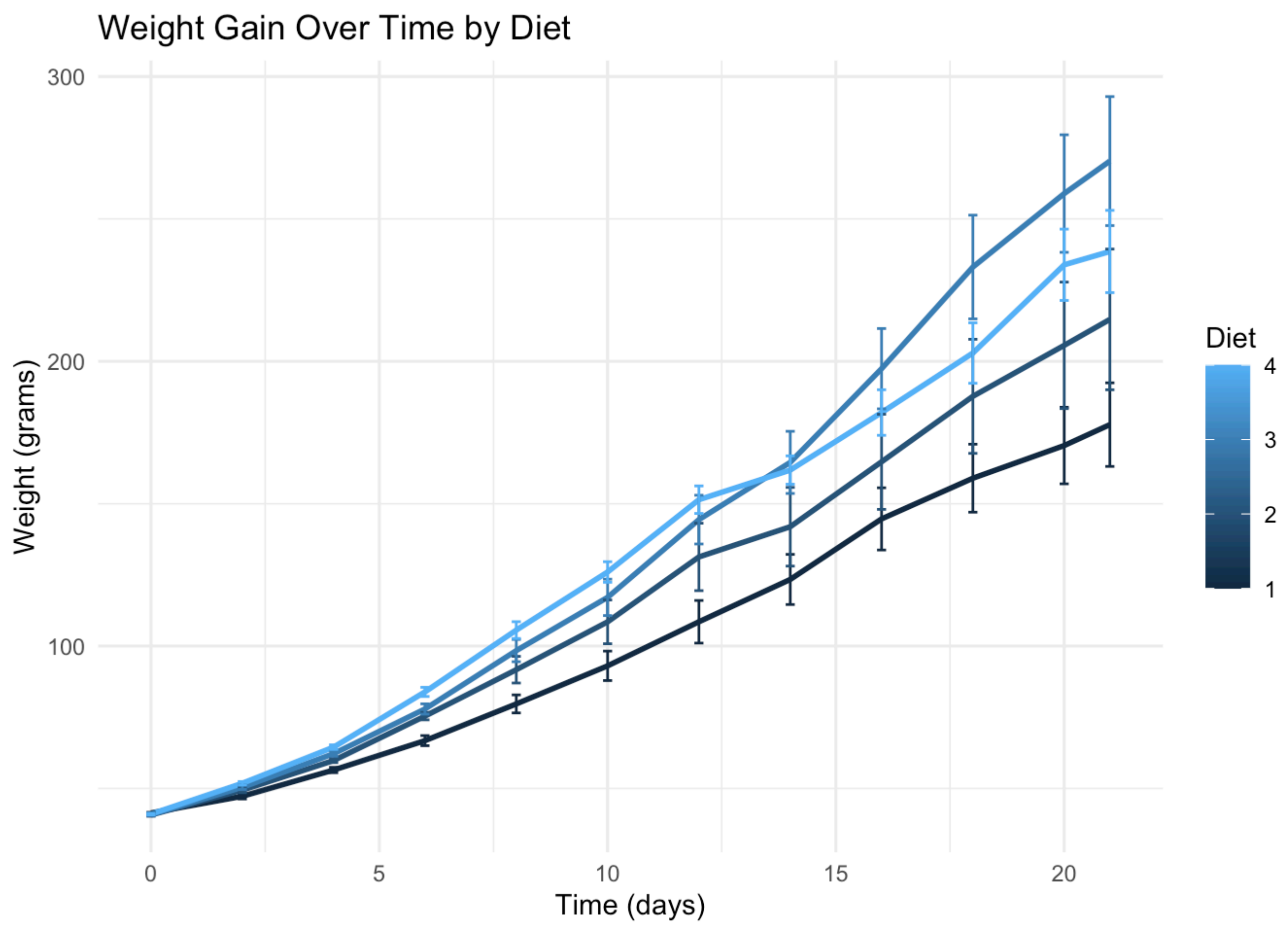
1. Create a summary table showing the average weight of chicks for each diet at each time point. (0.5pts)

```
summary_table <- ChickWeight %>%
  group_by(Diet, Time) %>%
  summarize(mean_weight = mean(weight), .groups = 'drop')
print(summary_table)
```

```
## # A tibble: 48 × 3
##     Diet  Time mean_weight
##    <int> <int>       <dbl>
##  1     1     0        41.4
##  2     1     2        47.2
##  3     1     4        56.5
##  4     1     6        66.8
##  5     1     8        79.7
##  6     1    10        93.1
##  7     1    12       109.
##  8     1    14       123.
##  9     1    16       145.
## 10     1    18       159.
## # ℹ 38 more rows
```

2. Visualize the weight gain over time for each diet using a line plot with error bars representing the standard error of the mean using ggplot. (0.5pts)

```
p <- ggplot(ChickWeight, aes(x = Time, y = weight, color = Diet, group = Diet)) +
  stat_summary(fun = mean, geom = "line", linewidth = 1) +
  stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.2) +
  labs(title = "Weight Gain Over Time by Diet",
       x = "Time (days)", y = "Weight (grams)") +
  theme_minimal()
print(p)
```

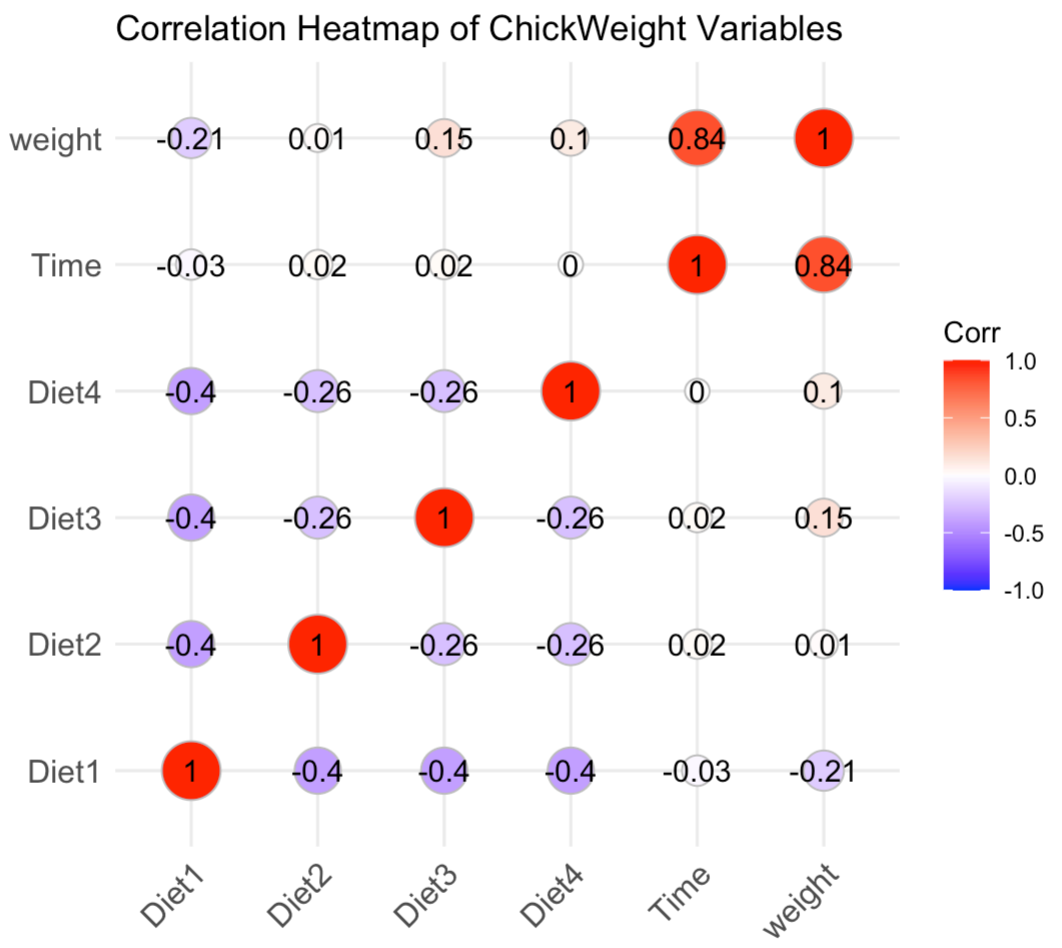


3. Use the ggcorrplot package to create a correlation heatmap of the numeric variables in the dataset. Convert the diet column to dummy variables (columns) and include them in the correlation analysis. (1pts)

```r
# Ensure 'Diet' is treated as a categorical variable (factor)
ChickWeight$Diet <- as.factor(ChickWeight$Diet)
numeric_data <- model.matrix(~ 0+Diet + Time + weight, data = ChickWeight)
corr_matrix <- cor(numeric_data)



ggcorrplot(corr_matrix,
  method = "circle",     # Use circles to represent correlation
  type = "full",        # Show full correlation matrix
  lab = TRUE,
  title = "Correlation Heatmap of ChickWeight Variables"
)
```



4. Calculate a slope coefficient for each diet and time combination using a custom function. HINT: use the lm() function inside calculate slope and add 0+ in the front of the independent variables to get slopes for all but no intercepts, otherwise the lm function will drop one of the diet columns to avoid collinearity. (1pts)

```
li_model <- lm(weight ~ 0 + Diet + Diet:Time, data = ChickWeight)

summary(li_model)
```

```
##
## Call:
## lm(formula = weight ~ 0 + Diet + Diet:Time, data = ChickWeight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -135.425  -13.757   -1.311   11.069  130.391
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## Diet1       30.9310     4.2468   7.283 1.09e-12 ***
## Diet2       28.6336     5.8972   4.855 1.55e-06 ***
## Diet3       18.2503     5.8972   3.095  0.00207 **
## Diet4       30.7921     5.9209   5.201 2.77e-07 ***
## Diet1:Time   6.8418     0.3408  20.076  < 2e-16 ***
## Diet2:Time   8.6091     0.4590  18.757  < 2e-16 ***
## Diet3:Time  11.4229     0.4590  24.887  < 2e-16 ***
## Diet4:Time   9.7144     0.4670  20.802  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.07 on 570 degrees of freedom
## Multiple R-squared:  0.9424, Adjusted R-squared:  0.9416
## F-statistic:  1166 on 8 and 570 DF,  p-value: < 2.2e-16
```

5. Write a function that calculates residual sum of squares (RSS), and then compare the minimal value to find the optimal slopes for each parameter and parameter combination. (1pts)

```
# Function to calculate RSS from a model
calculate_rss <- function(model) {
  sum(residuals(model)^2)
}
```

You can loop through different models and calculate RSS for each

to find the optimal slopes for each parameter and parameter combination.

You can work through it manually as well

```
results <- data.frame(
  Diet = character(),
  RSS = numeric()
)

unique_diets <- levels(ChickWeight$Diet) # get unique diets

# loop
for (d in unique_diets) {

  subset_data <- ChickWeight[ChickWeight$Diet == d, ]
  model <- lm(weight ~ Time, data = subset_data)
  rss_value <- calculate_rss(model)
  results <- rbind(results, data.frame(Diet = d, RSS = rss_value))
}

print(results)
```

```
##   Diet       RSS
## 1    1 235211.53
## 2    2 201864.21
## 3    3 172400.72
## 4    4  52055.58
```

```
#Diet4 has the lowest RSS
```

6. Use anova to compare the RSS to see if they're significant – compare the F statistic. Use the built in anova function. (0.5pts)

```
time_model <- lm(weight ~ Time, data = ChickWeight)
anova(time_model,li_model)
```

```
## Analysis of Variance Table
##
## Model 1: weight ~ Time
## Model 2: weight ~ 0 + Diet + Diet:Time
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    576 872212
## 2    570 661532  6    210680 30.255 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(li_model)
```

```
##
## Call:
## lm(formula = weight ~ 0 + Diet + Diet:Time, data = ChickWeight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -135.425  -13.757   -1.311   11.069  130.391
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## Diet1       30.9310     4.2468   7.283 1.09e-12 ***
## Diet2       28.6336     5.8972   4.855 1.55e-06 ***
## Diet3       18.2503     5.8972   3.095  0.00207 **
## Diet4       30.7921     5.9209   5.201 2.77e-07 ***
## Diet1:Time   6.8418     0.3408  20.076  < 2e-16 ***
## Diet2:Time   8.6091     0.4590  18.757  < 2e-16 ***
## Diet3:Time  11.4229     0.4590  24.887  < 2e-16 ***
## Diet4:Time   9.7144     0.4670  20.802  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.07 on 570 degrees of freedom
## Multiple R-squared:  0.9424, Adjusted R-squared:  0.9416
## F-statistic:  1166 on 8 and 570 DF,  p-value: < 2.2e-16
```

7. Fit a linear model to assess the effect of diet and time on weight. Use backwards selection to find the best model just against the p-values of the coefficients. Use the same approach with the 0 + leading the independent variables to ensure all lines are present. (1pts)

```
full_model <- lm(weight ~ 0 + Diet + Time, data = ChickWeight)
summary(full_model)
```

```
##
## Call:
## lm(formula = weight ~ 0 + Diet + Time, data = ChickWeight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -136.851  -17.151   -2.595   15.033  141.816
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## Diet1  10.9244     3.3607   3.251  0.00122 **
## Diet2  27.0905     4.0816   6.637 7.42e-11 ***
## Diet3  47.4238     4.0816  11.619  < 2e-16 ***
## Diet4  41.1578     4.0828  10.081  < 2e-16 ***
## Time    8.7505     0.2218  39.451  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35.99 on 573 degrees of freedom
## Multiple R-squared:  0.9354, Adjusted R-squared:  0.9348
## F-statistic:  1659 on 5 and 573 DF,  p-value: < 2.2e-16
```

```
# I am not so sure about which linaer model that you are talking about. if we still u
se li_model,according to the summary above, we don't have to remove any variable
# if we use the new full_model, we still don't have to remove any variable according
to the summary below
```

8. Iteratively enhance with backwards selection. When the F statistic becomes insignificant, stop. Do not use the step function, implement your own F test based backwards selection. (1pts)

```
# double check data types
ChickWeight$Diet <- factor(ChickWeight$Diet)
ChickWeight$Time <- as.numeric(ChickWeight$Time)



f_test_backward_selection <- function(model, alpha = 0.05) { #set alpha = 0.05
  repeat {
    summ <- summary(model)
    coefs <- summ$coefficients
    pvals <- coefs[, "Pr(>|t|)"]

    # find the largest p-value
    worst_p <- max(pvals, na.rm = TRUE)
    worst_term <- names(which.max(pvals))

    if (is.na(worst_p) || worst_p <= alpha) {
      message("All remaining terms are significant. Stop.")
      break
    }

    message("Dropping term: ", worst_term, " (p = ", round(worst_p, 4), ")")

    # remove it
    new_formula <- update(formula(model), paste(". ~ . -", worst_term))
    model <- lm(new_formula, data = model$model)
  }
  return(model)
}


best_model_p <- f_test_backward_selection(li_model, alpha = 0.05)
```

```
## All remaining terms are significant. Stop.
```

```
# check print
summary(best_model_p)
```

```
##
## Call:
## lm(formula = weight ~ 0 + Diet + Diet:Time, data = ChickWeight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -135.425  -13.757   -1.311   11.069  130.391
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## Diet1       30.9310     4.2468   7.283 1.09e-12 ***
## Diet2       28.6336     5.8972   4.855 1.55e-06 ***
## Diet3       18.2503     5.8972   3.095  0.00207 **
## Diet4       30.7921     5.9209   5.201 2.77e-07 ***
## Diet1:Time   6.8418     0.3408  20.076  < 2e-16 ***
## Diet2:Time   8.6091     0.4590  18.757  < 2e-16 ***
## Diet3:Time  11.4229     0.4590  24.887  < 2e-16 ***
## Diet4:Time   9.7144     0.4670  20.802  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.07 on 570 degrees of freedom
## Multiple R-squared:  0.9424, Adjusted R-squared:  0.9416
## F-statistic:  1166 on 8 and 570 DF,  p-value: < 2.2e-16
```

9. Create a quadratic line with just weight vs time (quadratic vs linear) – calculate RSS with results from quadratic to see if it's better. (0.5pts)

```
# Linear model
linear_model <- lm(weight ~ Time, data = ChickWeight)
linear_rss <- sum(residuals(linear_model)^2)

# Quadratic model
quadratic_model <- lm(weight ~ Time + I(Time^2), data = ChickWeight)
quadratic_rss <- sum(residuals(quadratic_model)^2)

cat("Linear model RSS:", linear_rss, "\n")
```

```
## Linear model RSS: 872212.2
```

```
cat("Quadratic model RSS:", quadratic_rss, "\n")
```

```
## Quadratic model RSS: 850266.3
```

```
cat("Improvement in RSS:", linear_rss - quadratic_rss, "\n")
```

```
## Improvement in RSS: 21945.91
```

```
cat("Percentage improvement:", (linear_rss - quadratic_rss) / linear_rss * 100, "%
\n")
```

```
## Percentage improvement: 2.51612 %
```

```
# Quadratic model is slightly better than linear model
```

10. Generate a null model of chick-weight to hypothetically use for forwards selection. (0.5pts)

```
# Null model (intercept only)
null_model <- lm(weight ~ 1, data = ChickWeight)
null_rss <- sum(residuals(null_model)^2)
summary(null_model)
```

```
##
## Call:
## lm(formula = weight ~ 1, data = ChickWeight)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -86.82 -58.82 -18.82  41.93 251.18
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  121.818      2.956   41.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.07 on 577 degrees of freedom
```

```
cat("Null model (intercept only):\n")
```

```
## Null model (intercept only):
```

```
print(summary(null_model))
```

```
##
## Call:
## lm(formula = weight ~ 1, data = ChickWeight)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -86.82 -58.82 -18.82  41.93 251.18
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  121.818      2.956   41.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.07 on 577 degrees of freedom
```

```
cat("Null model RSS:", null_rss, "\n")
```

```
## Null model RSS: 2914556
```

```
# Calculate R-squared for null model
total_ss <- sum((ChickWeight$weight - mean(ChickWeight$weight))^2)
null_r_squared <- 1 - (null_rss / total_ss)
cat("Null model R-squared:", null_r_squared, "\n")
```

```
## Null model R-squared: -2.220446e-16
```

Bonus: Show that slope of linear regression is pearsons correlation r times the ratio of standard deviations for a simple linear model. (1pts)

You can show it analytically by mashing together the equations for ß_1 and r, or you can show it numerically by simulating some data and fitting a linear model and calculating the correlation coefficient and standard deviations.

The equation for sd(X) in terms of sum of squares is:

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - X)^2}$$

where X is the predictor variable, n is the number of observations, and X is the mean of X.

The equation for sd(Y) in terms of sum of squares is:

$$s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - Y)^2}$$

where Y is the response variable, n is the number of observations, and Y is the mean of Y.

The equation for cov(X,Y) is:

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - X)(Y_i - Y)$$

where X is the predictor variable, Y is the response variable, n is the number of observations, and X and Y are the means of X and Y, respectively.

The equation to calculate pearson's correlation is:

$$r = \frac{cov(X,Y)}{s_x s_y}$$

where $cov(X,Y)$ is the covariance between the predictor variable X and the response variable Y, s_x is the standard deviation of X, and s_y is the standard deviation of Y.

The numerical equation for ß_1 in a simple linear regression is:

$$\beta_1 = \frac{\Sigma_{i=1}^{n}(x_i - x)(y_i - y)}{\Sigma_{i=1}^{n}(x_i - x)^2}$$

The symbolic representation of the same equation is:

$$\beta_1 = \frac{cov(X, Y)}{s_x^2}$$

where $cov(X, Y)$ is the covariance between the predictor variable X and the response variable Y, and s_x^2 is the variance of X.

Can you prove the following by apply in algebra with the equations above:

$$\beta_1 = r\frac{s_y}{s_x}$$

where r is Pearson's correlation coefficient, s_y is the standard deviation of the response variable, and s_x is the standard deviation of the predictor variable.

Write the solution by hand and upload a photo of your proof; or, if using a numerical comparison, submit the code here:

Define linear regression: $y=\beta_0+\beta_1 x$

$$\beta_1 = \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n} (x_i-\bar{x})^2}$$ (according to least squares methods)

$$= \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})(y_i-\bar{y})^2}{\left(\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2 \sum_{i=1}^{n} (y_i-\bar{y})^2}\right)^2}$$

$$= \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1} (x_i-\bar{x})^2 \sum_{i=1}^{n} (y_i-\bar{y})^2}} \cdot \frac{\sum_{i=1}^{n} (y_i-\bar{y})^2}{\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2 \sum_{i=1}^{n} (y_i-\bar{y})^2}} \quad 1)$$

we know

$$corr(x,y) = \frac{cov(x,y)}{\sqrt{var(x)}\sqrt{var(y)}} = \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i-\bar{y})^2}} \quad (2)$$

according to (1) (2)

$$\beta_1 = corr(x,y)\frac{\cancel{\sum_{i=1}^{n} (y_i-\bar{y})^2}}{\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2 \cancel{\sum_{i=1}^{n} (y_i-\bar{y})^2}}} = corr(x,y)\frac{\sqrt{\sum_{i=1}^{n} (y_i-\bar{y})^2}}{\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2}}$$

and

$$sd(x) = \sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2}, \quad sd(y) = \sqrt{\sum_{i=1}^{n} (y_i-\bar{y})^2}$$

$\Rightarrow \beta_1 = corr(x,y)\frac{sd(y)}{sd(x)}$, so the slope equals to pearsons correlation times $\frac{sd(y)}{sd(x)}$

proof