Correlation (Part 1 of 2)

Preview

October 21 & 28: Correlation

- Covariance and Linear Regression
- Matrix Algebra

November 4: Prediction

- Fitting Models and Machine Learning
- Cross-Validation, Regularization, and Ensembles

November 11 & 18: Causation

- Experimental Data
- Observational Causal Inference

November 25: Thanksgiving

December 2 & 9: Bonus Weeks!

- *Possible Topics*: Big Data, Text-As-Data, Networks, Spatial/Geographic Data, Advanced `R`, Advanced Visualizations (Interactives/Animations)

Correlation

By the end of this module you will be able to…

1. Compute covariance and correlation coefficients.
2. Estimate the slope of a line of best fit, plus confidence intervals and p-values.
3. Fit multivariable linear models using matrix algebra.

Covariance and Correlation

Covariance

Recall that the **variance** of a random variable is its expected squared distance from the mean:

$$\mathrm{Var}(X) = E[(X - E(X))^2]$$

The **covariance** extends that definition of variance to two random variables X and Y:

$$\mathrm{Cov}(X, Y) = E[(X - E(X))(Y - E(Y))]$$

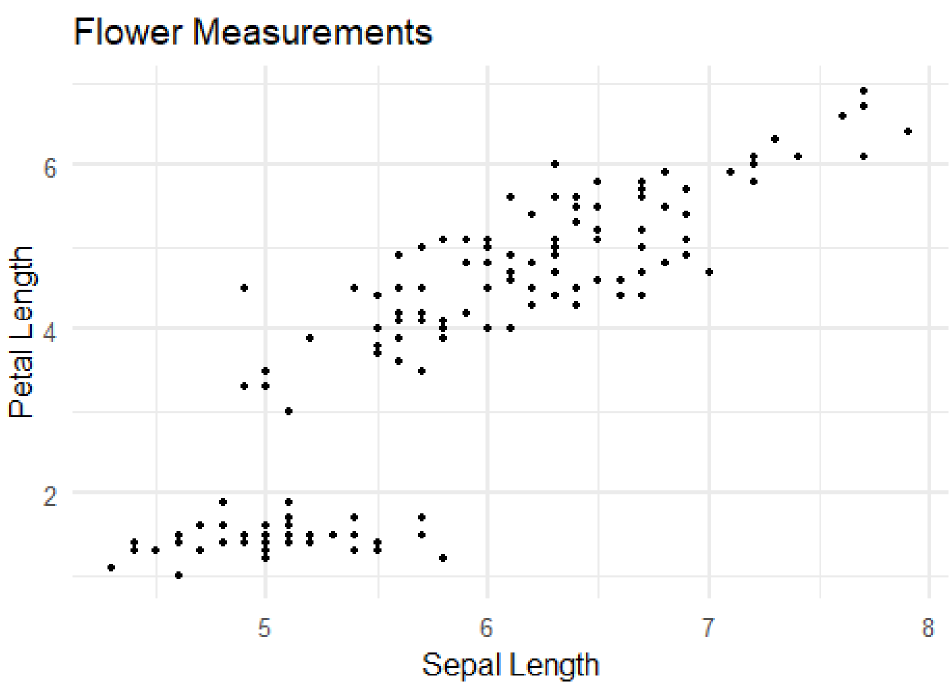
Covariance captures the degree of association between two variables. Does X tend to be high when Y is high?

Note that:

$$\mathrm{Cov}(X, X) = \mathrm{Var}(X)$$

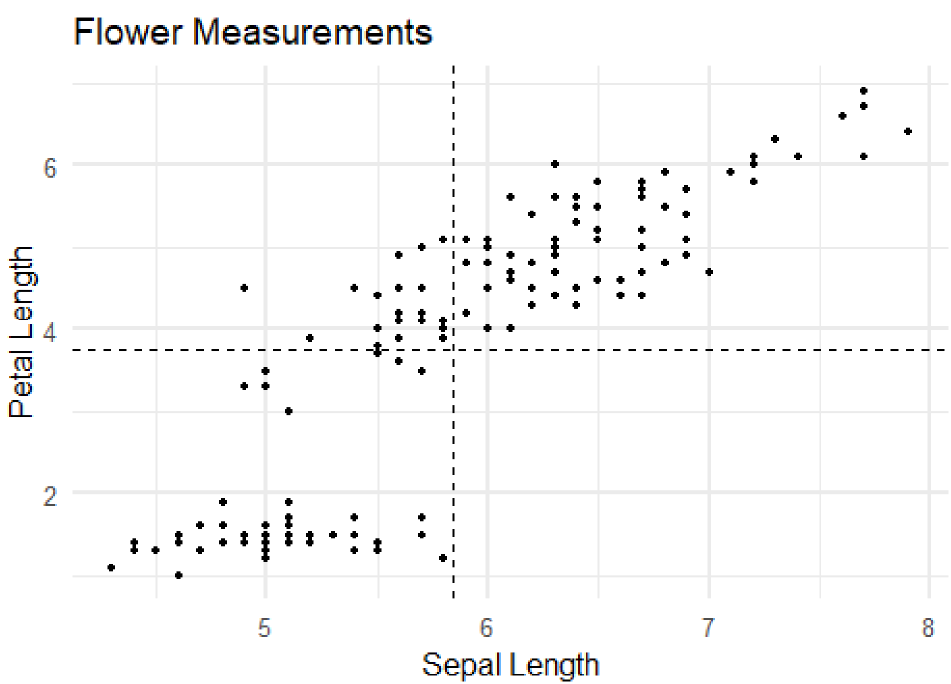
Covariance

```
flower_plot <- ggplot(data = iris) +
  geom_point(mapping = aes(x = Sepal.Length, y = Petal.Length)) +
  labs(x = 'Sepal Length', y = 'Petal Length', title = 'Flower Measur
flower_plot
```



Covariance

```
flower_plot <- flower_plot +
  geom_vline(xintercept = mean(iris$Sepal.Length), linetype = 'dashed
  geom_hline(yintercept = mean(iris$Petal.Length), linetype = 'dashed
flower_plot
```



Covariance

Because petal length tends to be larger than average whenever sepal length is larger than average (and vice versa) when you take the mean of all the the $(X - \bar{X})(Y - \bar{Y})$, you get a positive number.

```
cov(iris$Sepal.Length, iris$Petal.Length)
```

```
[1] 1.274315
```

When covariance is positive, X and Y tend to move together. When covariance is negative, X and Y tend to move in opposite directions.

Correlation Coefficients

The problem with covariance is that it's not easily interpretable. What does a covariance of 1.2743154 mean? How strong is that relationship?

The **correlation** coefficient solves that problem by standardizing the covariance.

$$\mathrm{Cor}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

This yields a value between -1 (perfectly anti-correlated) and $+1$ (perfectly correlated).

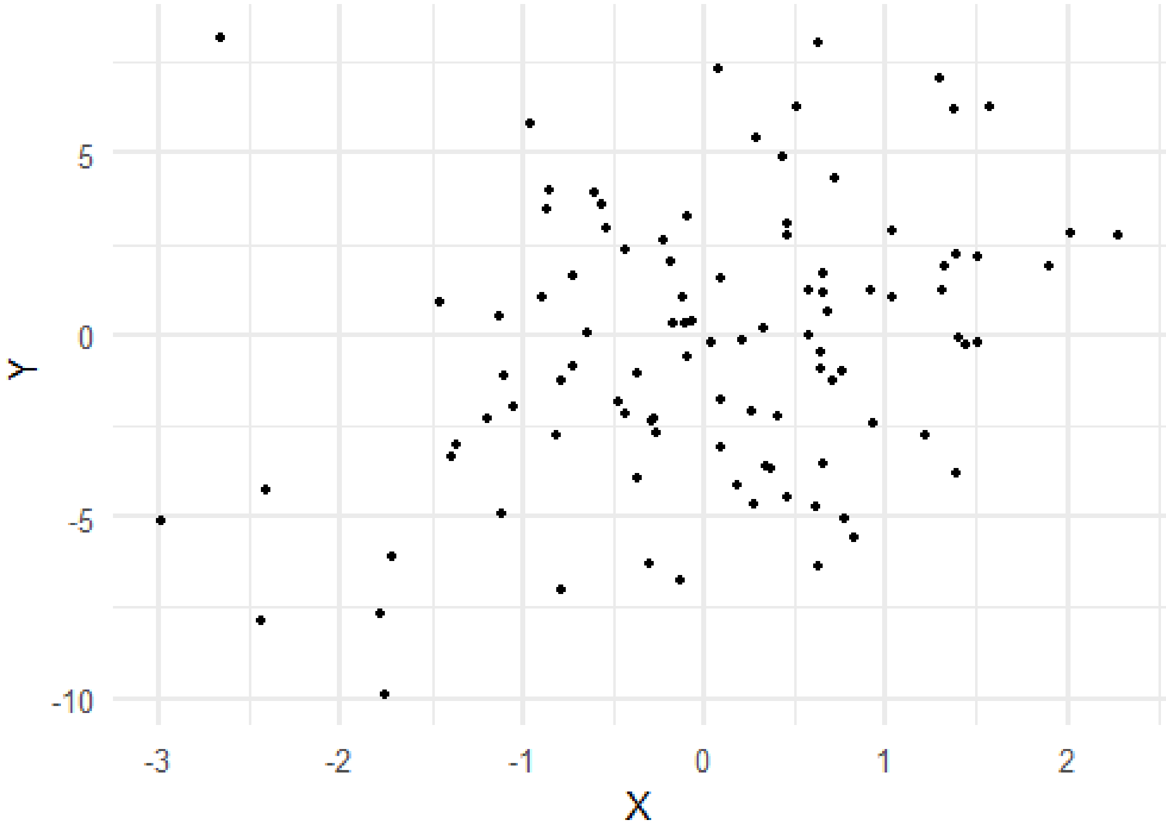
```
cor(iris$Sepal.Length, iris$Petal.Length)
```

```
[1] 0.8717538
```

```
cov(iris$Sepal.Length, iris$Petal.Length) / sd(iris$Sepal.Length) / s
```

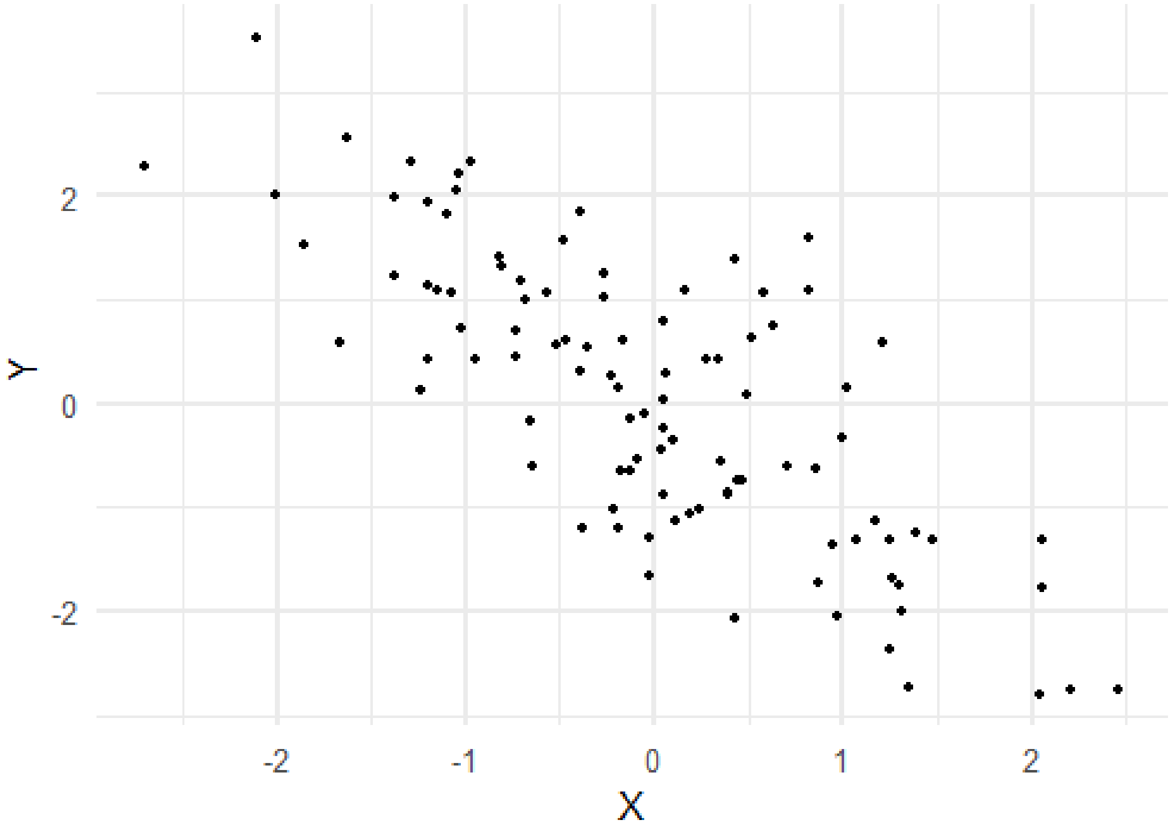
```
[1] 0.8717538
```

Let's Play "Guess The Correlation"



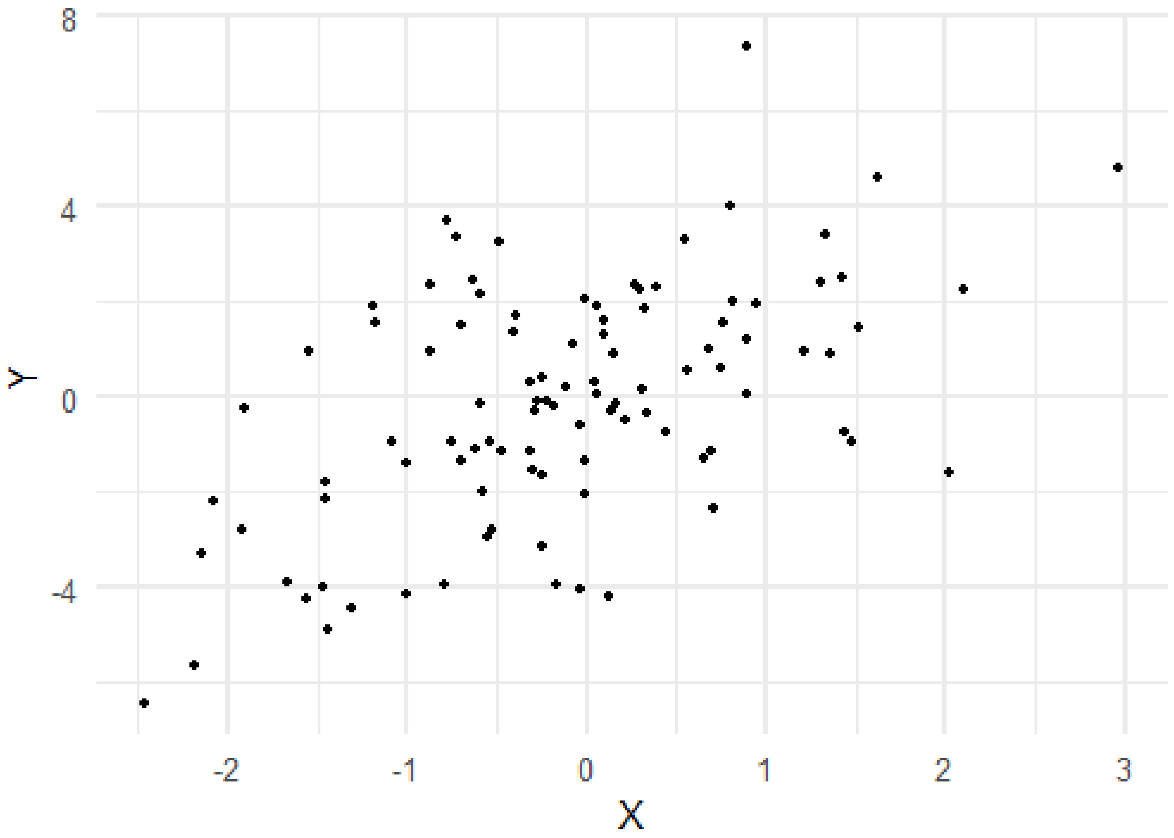
Actual Correlation: 0.3042285

Let's Play "Guess The Correlation"



Actual Correlation: -0.7702996

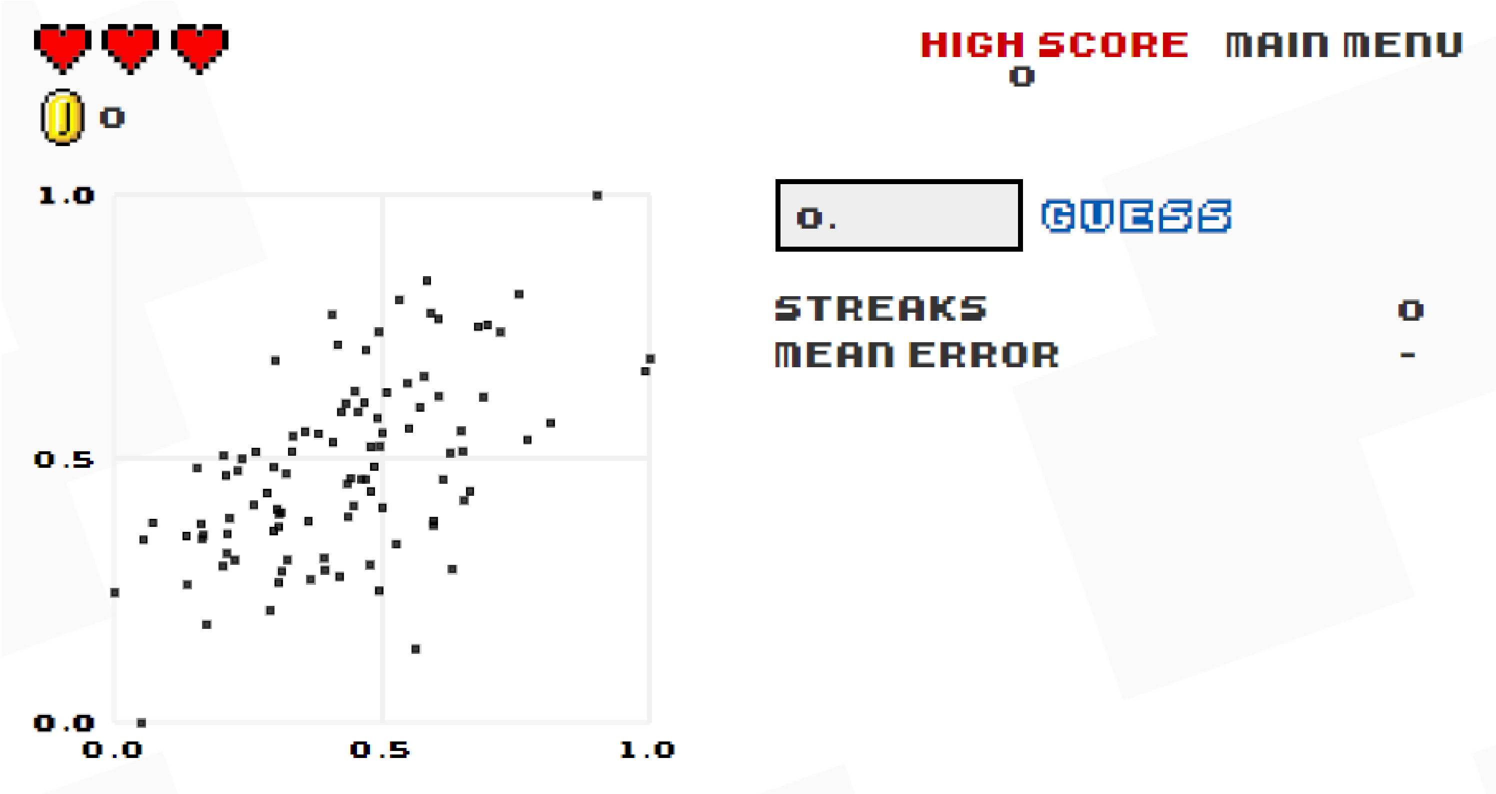
Let's Play "Guess The Correlation"



Actual Correlation: 0.5387565

Let's Play "Guess The Correlation"

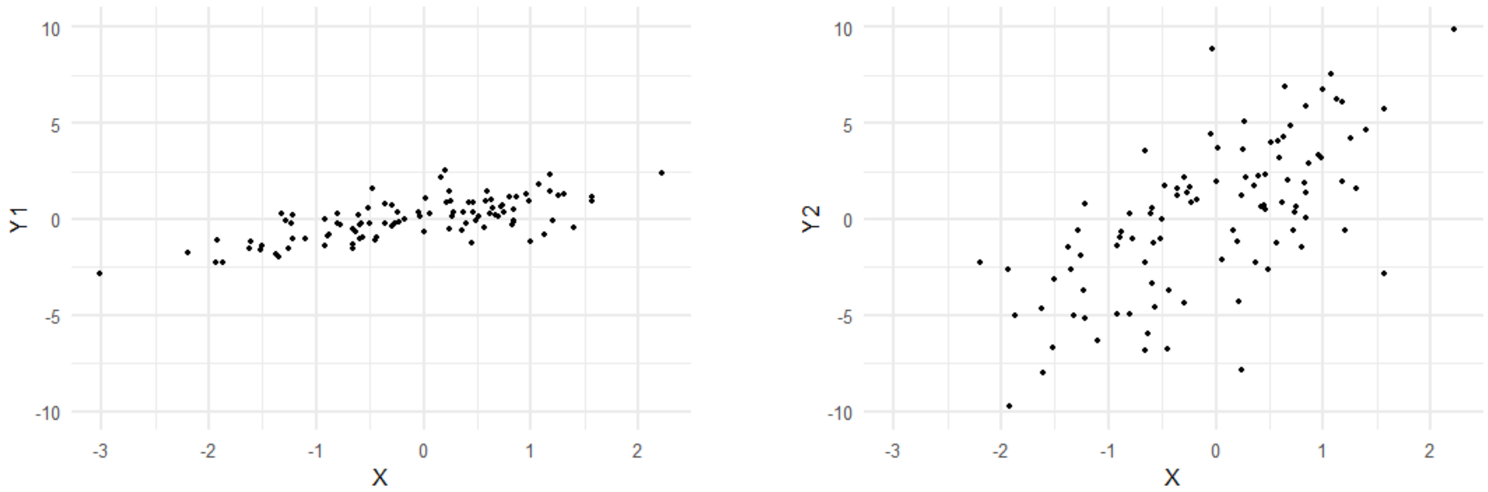
For more fun, try http://guessthecorrelation.com/



Linear Regression

Linear Regression

Correlation coefficients are nice, but limited. Both pairs of variables below have the same correlation coefficient (0.696 and 0.696).

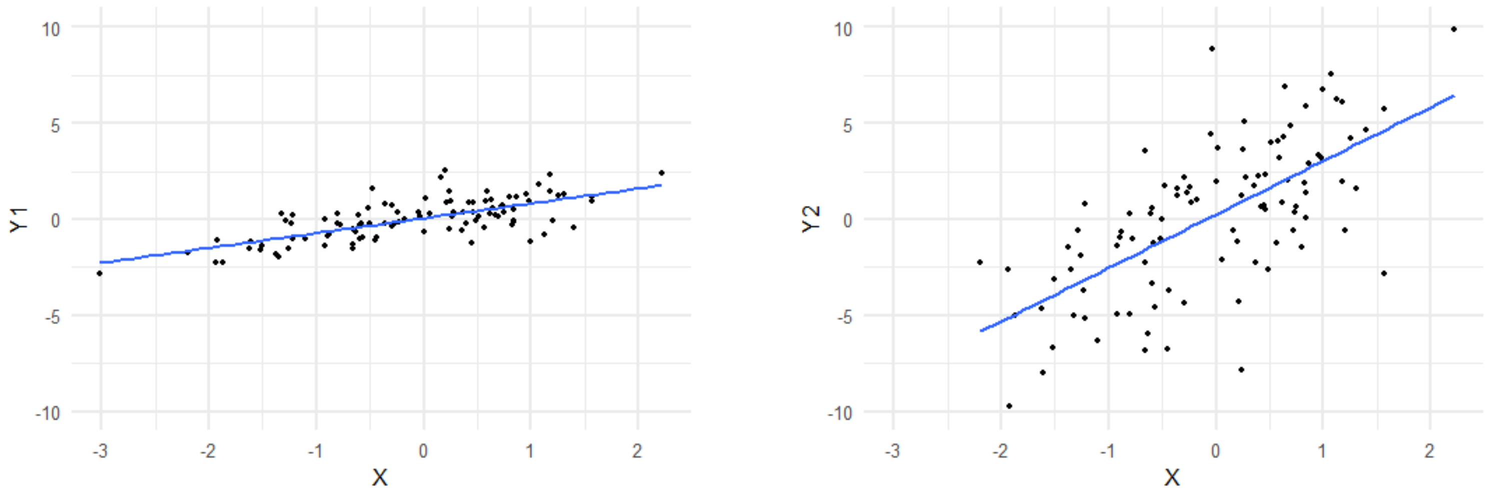


We want to find the **slope** of the relationship (the "line of best fit").

- When we increase X by 1, how much does Y increase or decrease, on average?

Linear Regression

Correlation coefficients are nice, but limited. Both pairs of variables below have the same correlation coefficient (0.696 and 0.696).



We want to find the **slope** of the relationship (the "line of best fit").

- When we increase X by 1, how much does Y increase or decrease, on average?

Linear Regression

The two-variable linear model looks like this:

$$Y = a + bX + \varepsilon$$

Terms:

- Y is a **vector** of outcomes
- X is a **vector** we're using to predict the outcome
- a is the y-intercept
- b is the slope of the relationship between X and Y, and
- ε is **vector** of random error
  - The difference between the true value of Y and the predicted value $a + bX$.

Linear Regression

$$Y = a + bX + \varepsilon$$

Example:

$$X = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} \qquad Y = \begin{bmatrix} 4 \\ 6 \\ 10 \end{bmatrix}$$

$a = 2, b = 2$

$$\underbrace{\begin{bmatrix} 4 \\ 6 \\ 10 \end{bmatrix}}_{Y} = \underbrace{2 \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{a} + \underbrace{2 \times \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}}_{bX} + \underbrace{\begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix}}_{\varepsilon}$$

The Line of Best Fit

The "line of best fit" is the one that minimizes error (specifically, the sum of squared errors).

```
ggplot(data = iris) +
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length)) +
  geom_smooth(mapping = aes(x= Petal.Width, y = Petal.Length),
              method = 'lm', se = FALSE)
```

Estimating The Line of Best Fit

To make things easier, we will ignore the y-intercept for now.

- Create new variables called X and Y, equal to Petal Width and Petal Length minus their means.

```
demeaned_iris <- iris %>%
  mutate(Y = Petal.Length - mean(Petal.Length), X = Petal.Width - mea
```

Exercise: What is the mean of X?

```
demeaned_iris$X %>% mean %>% round(4)
```

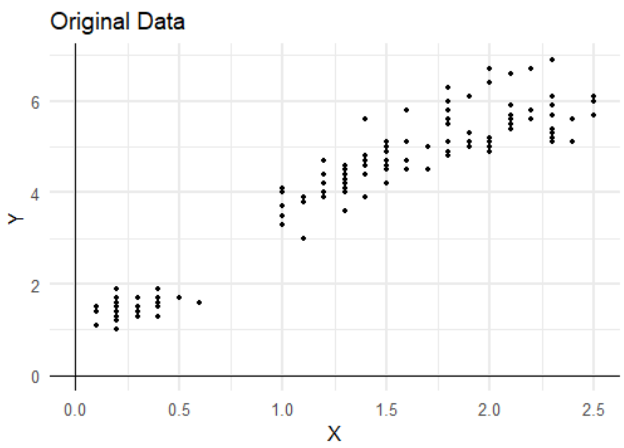
```
[1] 0
```

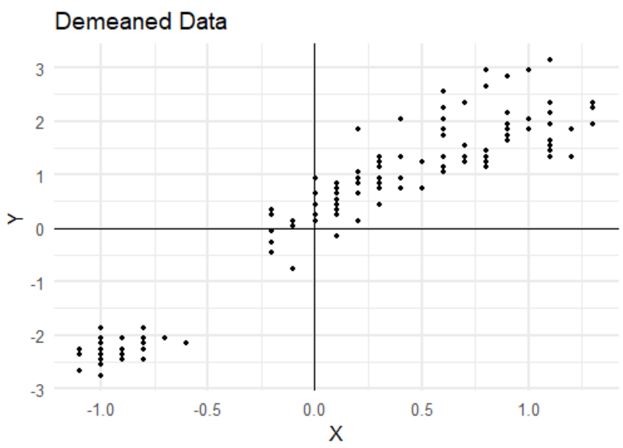
You'll thank me when we start doing the calculus.

Estimating The Line of Best Fit

```
ggplot(iris) +
  geom_point(mapping = aes(x=Pet
  labs(x='X',y='Y',title = 'Orig
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0)
```

```
ggplot(demeaned_iris) +
  geom_point(mapping = aes(x=X,y
  labs(x='X',y='Y',title = 'Deme
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0)
```





The data looks the same; we've just shifted it down and to the left.

Estimating The Line of Best Fit

The "best" line is the one that minimizes error. Specifically, we're going to find the line that minimizes the **sum of squared errors**.

$$Y = bX + \varepsilon$$

$$\varepsilon = Y - bX$$

Let's create a function called $f(b)$ equal to the sum of squared errors:

$$f(b) = \sum \varepsilon_i^2 = \sum (Y_i - bX_i)^2$$

$$f(b) = \sum Y_i^2 - \sum 2bX_iY_i + \sum b^2X_i^2$$

Three Steps to Minimize a Function?

Step 1: Take the derivative

$$f(b) = \sum Y_i^2 - \sum 2bX_iY_i + \sum b^2X_i^2$$

$$\frac{\partial f}{\partial b} = -2\sum X_iY_i + 2b\sum X_i^2$$

Step 2: Set Equal to Zero

$$-2\sum X_iY_i + 2b\sum X_i^2 = 0$$

Step 3: Solve for b

$$2\sum X_iY_i = 2b\sum X_i^2$$

$$b = \frac{\sum X_iY_i}{\sum X_i^2}$$

Slope Estimate

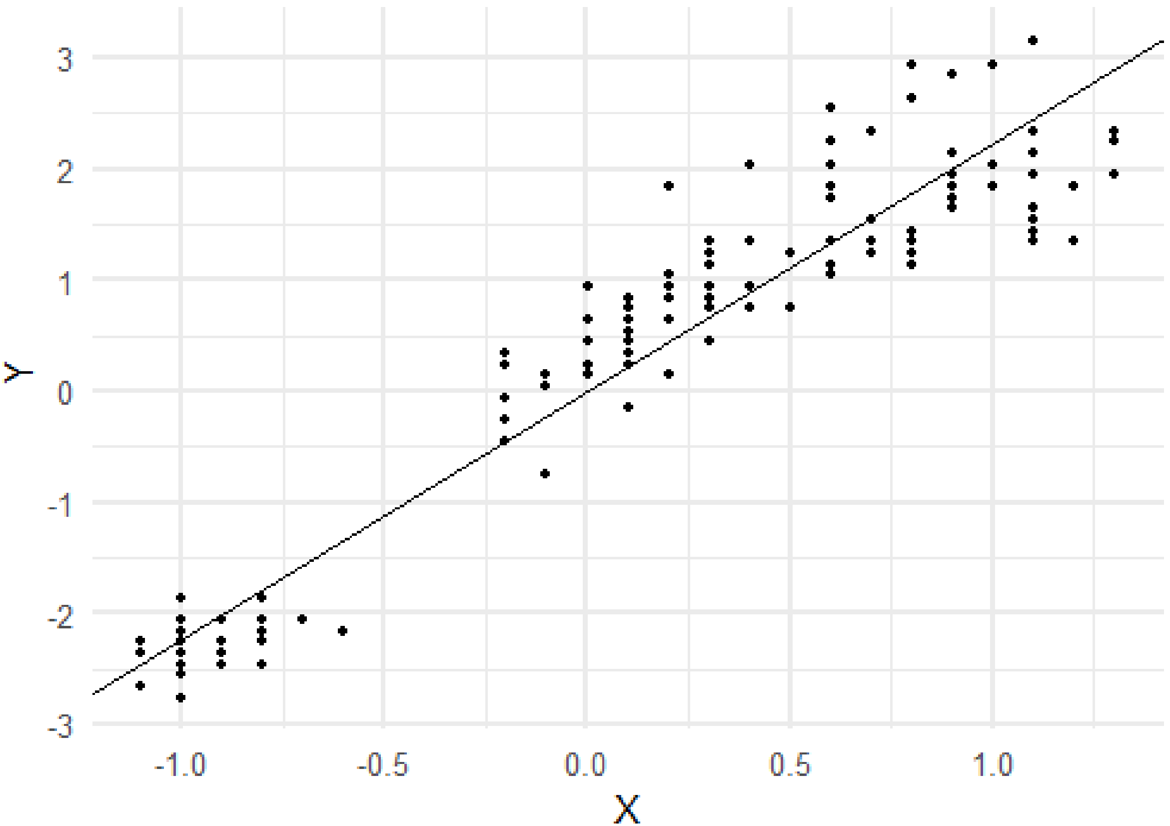
$$b = \frac{\sum X_i Y_i}{\sum X_i^2}$$

```
slope_estimate <-
  sum(demeaned_iris$X * demeaned_iris$Y) /
  sum(demeaned_iris$X^2)

slope_estimate
```

```
[1] 2.22994
```

Slope Estimate

```
ggplot(demeaned_iris) +
  geom_point(mapping = aes(x=X,y=Y)) +
  geom_abline(intercept = 0, slope = slope_estimate)
```



Some Terminology

The slope parameter b is called the **estimand**. It is the thing we are trying to estimate.

$\frac{\sum X_i Y_i}{\sum X_i^2}$ is the **estimator**. It is the equation we use to produce our estimate.

2.23 is our **estimate**.

- Typically, we denote estimates with little hats, like this: $\hat{b} = 2.23$.

Interesting Footnote

Notice that our estimator $\frac{\sum X_i Y_i}{\sum X_i^2}$ is equal to $\frac{\sum (X_i - 0)(Y_i - 0)}{\sum (X_i - 0)^2}$, which is equal to $\frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2}$ because $\bar{X} = 0$ and $\bar{Y} = 0$.

So another way of writing the estimator is $\hat{b} = \frac{\text{Cov}(X,Y)}{\text{Var}(X)}$.

```
slope_estimate
```

```
[1] 2.22994
```

```
cov(demeaned_iris$X, demeaned_iris$Y) / var(demeaned_iris$X)
```

```
[1] 2.22994
```

Residuals

The vector of observed errors $(\hat{\varepsilon})$, also known as the **residuals**, is equal to $Y - \hat{b}X$.

```
# Compute the epsilon vector
epsilon <- demeaned_iris$Y - slope_estimate * demeaned_iris$X
```

If we fit the line correctly, then the average error should equal zero.

```
mean(epsilon)
```

```
[1] 4.446053e-17
```

Statistical Inference

Statistical Inference

We now have a **point estimate** of the slope ($\hat{b} = 2.23$). What if we want an **interval estimate** (confidence intervals) and p-values?

Three Steps

1. Specify the Null Hypothesis $(b = 0)$
2. Generate Sampling Distribution of $\hat{b}$ assuming $b = 0$
3. Compare observed value of $\hat{b}$ to the sampling distribution

Where do we get the sampling distribution?

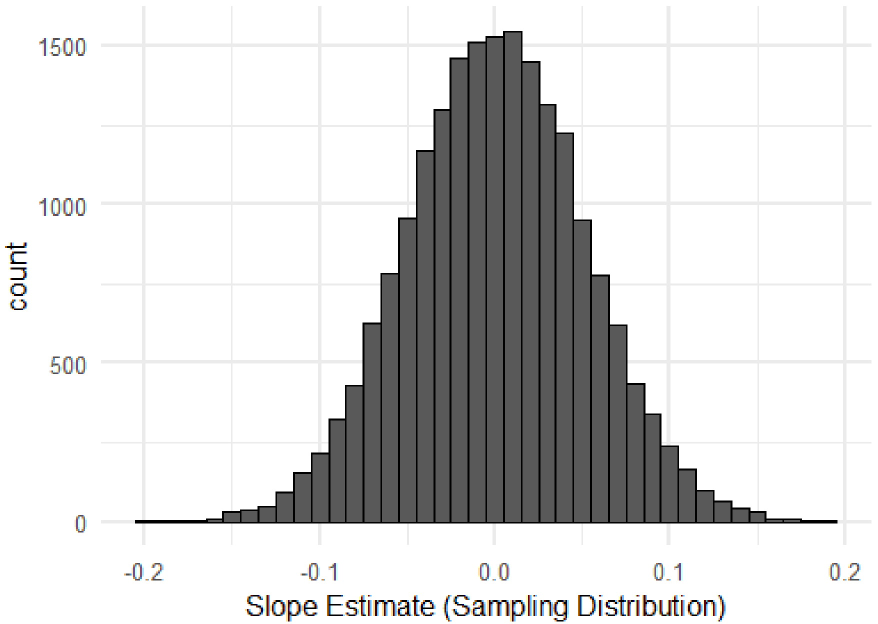
In a linear regression, the randomness comes from the error term ε. Imagine that we repeatedly draw a new ε vector with each sample.

```r
null_slope_estimate <- function(X, epsilon){

  # Randomly sample a vector of epsilons
  epsilon <- sample(epsilon, replace = TRUE)

  # null hypothesis: b = 0
  b <- 0

  # create a random dataset assuming the null hypothesis
  Y <- b*X + epsilon

  # Return the slope estimate
  sum(X * Y) / sum(X^2)
}

null_slope_estimate(X = demeaned_iris$X, epsilon)
```

```
[1] 0.01040073
```

Generate the Sampling Distribution

```r
sampling_distribution <- replicate(20000, null_slope_estimate(X = der

tibble(sampling_distribution) %>%
  ggplot() +
  geom_histogram(mapping = aes(x=sampling_distribution),
                 color = 'black', binwidth = 0.01) +
  labs(x='Slope Estimate (Sampling Distribution)') # a normally dist
```



P-values

```
sum(sampling_distribution > slope_estimate) # p-value effectively ze
```

```
[1] 0
```

Confidence Intervals

```
standard_error <- sd(sampling_distribution)
standard_error
```

```
[1] 0.05103464
```

```
confidence_interval <- c(slope_estimate - 1.96 * standard_error,
                         slope_estimate + 1.96 * standard_error)
confidence_interval
```

```
[1] 2.129913 2.329968
```

The One-Line Built-In R Function

The `lm()` function estimates the linear model parameters (slope + y-intercept) and computes confidence intervals and p-values.

```
linear_model_fit <- lm(formula = Y ~ X,
                       data = demeaned_iris)

coef(linear_model_fit) # get the coefficient estimates
```

```
  (Intercept)             X
-1.715932e-15  2.229940e+00
```

```
confint(linear_model_fit) # get the confidence intervals
```

```
                  2.5 %     97.5 %
(Intercept) -0.07715837 0.07715837
X            2.12837525 2.33150574
```

The One-Line Built-In R Function

```
summary(linear_model_fit)
```

```
Call:
lm(formula = Y ~ X, data = demeaned_iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.33542 -0.30347 -0.02955  0.25776  1.39453 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.716e-15  3.905e-02    0.00        1    
X            2.230e+00  5.140e-02   43.39   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4782 on 148 degrees of freedom
Multiple R-squared:  0.9271,	Adjusted R-squared:  0.9266 
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

Exercise

1. Check out the documentation for the `mtcars` dataset by typing `?mtcars`.
2. What is the mean fuel efficiency of cars (`mpg`), grouped by the number of cylinders (`cyl`)?
3. Do cars with a manual transmission have significantly higher/lower horsepower than those with an automatic transmission?
4. What is the correlation between horsepower (`hp`) and fuel efficiency (`mpg`)? Visualize the relationship.
5. Fit a linear model with horsepower as the predictor variable (X) and fuel efficiency as the outcome variable (Y). What is the slope of the relationship? What is the 95% confidence interval on that slope estimate?

Exercise (Solution)

What is the mean fuel efficiency of cars (`mpg`), grouped by the number of cylinders (`cyl`)?

```
mtcars %>%
  group_by(cyl) %>%
  summarize(mean_mpg = mean(mpg),
            num = n())
```

```
# A tibble: 3 x 3
    cyl mean_mpg   num
  <dbl>    <dbl> <int>
1     4     26.7    11
2     6     19.7     7
3     8     15.1    14
```

Exercise (Solution)

Do cars with a manual transmission have significantly higher/lower horsepower than those with an automatic transmission?

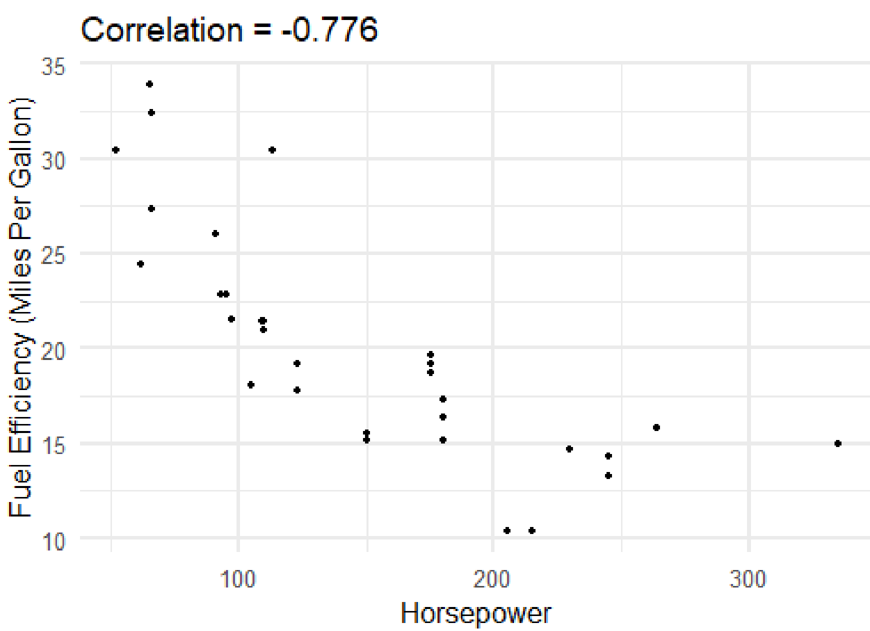
```
mtcars %>%
  t.test(hp ~ am, data = .)
```

```

	Welch Two Sample t-test

data:  hp by am
t = 1.2662, df = 18.715, p-value = 0.221
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -21.87858  88.71259
sample estimates:
mean in group 0 mean in group 1 
       160.2632        126.8462 
```

Exercise (Solution)

What is the correlation between horsepower (`hp`) and fuel efficiency (`mpg`)?

```
ggplot(data = mtcars) +
  geom_point(mapping = aes(x=hp, y=mpg)) +
  labs(x = 'Horsepower', y = 'Fuel Efficiency (Miles Per Gallon)',
       title = paste0('Correlation = ', cor(mtcars$hp, mtcars$mpg) %>
```



Exercise (Solution)

Fit a linear model with horsepower as the predictor variable (X) and fuel efficiency as the outcome variable (Y). What is the slope of the relationship? What is the 95% confidence interval on that slope estimate?

```
cars_linear_model <- lm(mpg ~ hp, data = mtcars)

coef(cars_linear_model)
```

```
(Intercept)          hp
30.09886054 -0.06822828
```

```
confint(cars_linear_model)
```

```
                  2.5 %     97.5 %
(Intercept) 26.76194879 33.4357723
hp          -0.08889465 -0.0475619
```

Multivariable Linear Regression

Multivariable Linear Regression

Suppose we want to explain the outcome as a function of **multiple** explanatory variables.

$$\text{mpg} = \alpha + \beta_1 \text{hp} + \beta_2 \text{wt} + \varepsilon$$

Fuel efficiency probably depends on both horsepower **and** weight. More powerful and heavier cars will tend to have lower fuel efficiency. We'd like to estimate the slope of both relationships simultaneously!

Vector Representation:

$$\underbrace{\begin{bmatrix} 21.0 \\ 21.0 \\ 22.8 \\ \vdots \\ 21.4 \end{bmatrix}}_{\text{mpg}} = \underbrace{\alpha \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{\alpha} + \underbrace{\beta_1 \times \begin{bmatrix} 110 \\ 110 \\ 93 \\ \vdots \\ 109 \end{bmatrix}}_{\beta_1 \text{hp}} + \underbrace{\beta_2 \times \begin{bmatrix} 2.62 \\ 2.875 \\ 2.32 \\ \vdots \\ 2.78 \end{bmatrix}}_{\beta_2 \text{wt}} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}}_{\varepsilon}$$

Multivariable Linear Regression

The challenge is to simultaneously estimate α, β_1, and β_2.

$$\text{mpg} = \alpha + \beta_1 \text{hp} + \beta_2 \text{wt} + \varepsilon$$

We've come as far as we can with scalar algebra. It's time you learned **matrix algebra**.

Matrix Algebra

Matrix Algebra

Recall from Week 1 that a **matrix** is a bunch of vectors squished together.

$$\text{hp} = \begin{bmatrix} 110 \\ 110 \\ 93 \\ \vdots \\ 109 \end{bmatrix} \qquad \text{wt} = \begin{bmatrix} 2.62 \\ 2.875 \\ 2.32 \\ \vdots \\ 2.78 \end{bmatrix}$$

$$X = \begin{bmatrix} 110 & 2.62 \\ 110 & 2.875 \\ 93 & 2.32 \\ \vdots & \vdots \\ 109 & 2.78 \end{bmatrix}$$

Matrix Algebra

The **dimension** of a matrix refers to the number of rows and columns. An $m \times n$ matrix has m rows and n columns.

```
dim(mtcars)
```

```
[1] 32 11
```

There are 32 rows and 11 columns in the mtcars data matrix.

Matrix Algebra

Adding and **subtracting** matrices is straightforward. Just add and subtract elementwise.

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 4 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 4 & 4 \\ 8 & 5 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 3 & 3 \\ 6 & 7 \\ 12 & 9 \end{bmatrix}$$

Multiplying and **dividing** is where it gets tricky.

- You can only multiply *some* matrices together (they must be **conformable**)
- And matrix division isn't really a thing. Instead, we multiply by the matrix's **inverse**.

Matrix Multiplication

Matrix Multiplication

First, let's introduce the **dot product** of two vectors.

$$a \cdot b = \sum a_i b_i$$

If $a = [3, 1, 2]$ and $b = [1, 2, 3]$, then the dot product of a and b equals:

$$a \cdot b = 3 \times 1 + 1 \times 2 + 2 \times 3 = 11$$

In `R`, a dot product can be computed like so:

```
A <- c(3,1,2)
B <- c(1,2,3)

# dot product
sum(A*B)
```

```
[1] 11
```

Matrix Multiplication

Exercise: Take the dot product of a and b.

$a = [1, 4, 5]$ and $b = [3, 2, 1]$

Answer:

$$a \cdot b = 1 \times 3 + 4 \times 2 + 5 \times 1 = 16$$

```
A <- c(1,4,5)
B <- c(3,2,1)

# dot product
sum(A*B)
```

```
[1] 16
```

Matrix Multiplication

When you multiply two matrices, you take a series of dot products.

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 4 & 4 \end{bmatrix}$$

To get the entry in the first row, first column of AB, take the dot product of:

- The first row of A, and
- The first column of B

Then do that for every row in A and every column in B.

$$AB = \begin{bmatrix} 1 \times 2 + 2 \times 4 & 1 \times 1 + 2 \times 4 \\ 2 \times 2 + 3 \times 4 & 2 \times 1 + 3 \times 4 \end{bmatrix} = \begin{bmatrix} 10 & 9 \\ 16 & 14 \end{bmatrix}$$

This is all very strange and confusing to get used to if you've never seen it before, but we'll soon see that it makes representing our multivariable linear regression problem a whole lot easier.

Matrix Multiplication

You can multiply matrices in `R` with the `%*%` command.

```
A <- cbind(c(1,2), c(2,3))
A
```

```
     [,1] [,2]
[1,]    1    2
[2,]    2    3
```

```
B <- cbind(c(2,4), c(1,4))
B
```

```
     [,1] [,2]
[1,]    2    1
[2,]    4    4
```

```
A %*% B
```

```
     [,1] [,2]
[1,]   10    9
[2,]   16   14
```

Matrix Multiplication

Exercise: Try multiplying these two matrices.

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 5 \\ 2 & 1 \end{bmatrix}$$

Answer:

$$AB = \begin{bmatrix} 4 \times 5 + 1 \times 2 & 4 \times 5 + 1 \times 1 \\ 1 \times 5 + 2 \times 2 & 5 \times 1 + 1 \times 2 \end{bmatrix} = \begin{bmatrix} 22 & 21 \\ 9 & 7 \end{bmatrix}$$

```r
A <- cbind(c(4,1), c(1,2))
B <- cbind(c(5,2), c(5,1))
A %*% B
```

```
     [,1] [,2]
[1,]   22   21
[2,]    9    7
```

Matrix Multiplication

This process -- taking the dot product of rows and columns -- means that you can only multiply two matrices AB if the row vectors of A are the same length as the column vectors in B.

- In other words, you can only multiply AB if the dimension of A is $m \times k$ and the dimension of B is $k \times n$.
- If this condition holds, then the two matrices are **conformable**.

Multiplying an $m \times k$ matrix with a $k \times n$ matrix yields an $m \times n$ matrix.

```
A <- cbind(c(3,1,2), c(2,2,2))
A
```

```
     [,1] [,2]
[1,]    3    2
[2,]    1    2
[3,]    2    2
```

```
B <- cbind(c(4,5), c(5,4), c(1,1
B
```

```
     [,1] [,2] [,3]
[1,]    4    5    1
[2,]    5    4    1
```

Exercise: Which can you multiply: AB or BA?

Matrix Multiplication

Answer: Both!

```
A %*% B
```

```
     [,1] [,2] [,3]
[1,]   22   23    5
[2,]   14   13    3
[3,]   18   18    4
```

```
B %*% A
```

```
     [,1] [,2]
[1,]   19   20
[2,]   21   20
```

But now try `A %*% A`. I can't do it here because `R` gets so mad it won't even render my slides.

Matrix Multiplication

To make matrices conformable for multiplication, sometimes you may need to take the **transpose** of a matrix. The transpose just takes the rows and turns them into columns.

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 2 \\ 3 & 3 \end{bmatrix} \qquad A' = \begin{bmatrix} 4 & 1 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

```
t(A)
```

```
     [,1] [,2] [,3]
[1,]    3    1    2
[2,]    2    2    2
```

```
t(A) %*% A
```

```
     [,1] [,2]
[1,]   14   12
[2,]   12   12
```

Matrix Multiplication

Multiplying a vector by its transpose is the same as taking the dot product with itself:

$$a = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$

$$a \cdot a = a'a = 1 \times 1 + 3 \times 3 + 4 \times 4 = 26$$

Hey, it's the **sum of squares**! That could be useful for something...

```
a <- c(1,3,4)

sum(a*a)
```

```
[1] 26
```

```
t(a) %*% a
```

```
     [,1]
[1,]   26
```

Matrix Inversion

Matrix Inversion

Before I can teach you how to **divide** matrices, I need to tell you about a very special matrix, called the **identity matrix**.

Remember how any number times 1 just equals the original number?

$$a \times 1 = a$$

This is called the **identity property**. It's what makes 1 a very special number.

The **identity matrix** (I) is basically the 1 of matrices.

$$AI = A$$

You multiply any matrix by I and you get the same matrix back.

Matrix Inversion

The identity matrix I_n is an $n \times n$ matrix with ones in the diagonal and zeroes everywhere else.

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise: Try multiplying this matrix with the following matrix A.

$$A = \begin{bmatrix} 2 & 1 & 5 \\ -2 & 8 & 100 \\ 7 & 42 & -42 \end{bmatrix}$$

Matrix Inversion

Answer:

```
diag(3) # Create the identity matrix in R with the `diag()` function
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

```
A <- rbind(c(2, 1, 5),
           c(-2, 8, 100),
           c(7, 42, -2))

# multiply AI
A %*% diag(3)
```

```
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]   -2    8  100
[3,]    7   42   -2
```

Matrix Inversion

Hey, remember how dividing $\frac{a}{b}$ is the same as multiplying $a \times \frac{1}{b}$?

$\frac{1}{b} = b^{-1}$ is called the **inverse** (or reciprocal) of b.

Exercise: What do you get when you multiply a number by its inverse?

- **Answer:** $a \times \frac{1}{a} = a^1 a^{-1} = a^0 = 1$

There is an equivalent concept in matrix algebra, called the **matrix inverse**.

$$AA^{-1} = I$$

Matrix Inversion

Good news! It's the 21st century. No one is going to make you solve for matrix inverses by hand.

There is literally a function in R called `solve()` which will do it for you.

```
solve(A)
```

```
           [,1]         [,2]         [,3]
[1,]  0.49976292 -0.025130394 -0.007112376
[2,] -0.08250356  0.004623044  0.024893314
[3,]  0.01659554  0.009127549 -0.002133713
```

```
A %*% solve(A)
```

```
              [,1]         [,2]         [,3]
[1,]  1.000000e+00 6.938894e-17 2.081668e-17
[2,] -2.220446e-16 1.000000e+00 0.000000e+00
[3,]  1.734723e-16 0.000000e+00 1.000000e+00
```

Matrix Inversion

Now that we know how to multiply by an inverse, we have what we need to perform matrix algebra.

Exercise: Solve this equation for A.

$$AB = C$$

Answer: Multiply both sides by B^{-1}

$$ABB^{-1} = CB^{-1}$$

$$AI = CB^{-1}$$

$$A = CB^{-1}$$

Matrix Inversion

Watch out for conformability! With matrices, it matters whether you multiply on the right or the left.

Exercise: Solve for B.

$$AB = C$$

Answer:

$$A^{-1}AB = A^{-1}C$$

$$IB = A^{-1}C$$

$$B = A^{-1}C$$

$$B \neq CA^{-1}$$

Back to Multivariable Regression

Multivariable Regression

This is the regression problem we wanted to solve:

$$\underbrace{\begin{bmatrix} 21.0 \\ 21.0 \\ 22.8 \\ \vdots \\ 21.4 \end{bmatrix}}_{\text{mpg}} = \underbrace{\alpha \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{\alpha} \underbrace{+ \beta_1 \times \begin{bmatrix} 110 \\ 110 \\ 93 \\ \vdots \\ 109 \end{bmatrix}}_{\beta_1 \text{hp}} \underbrace{+ \beta_2 \times \begin{bmatrix} 2.62 \\ 2.875 \\ 2.32 \\ \vdots \\ 2.78 \end{bmatrix}}_{\beta_2 \text{wt}} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}}_{\varepsilon}$$

Notice that we can restate it as a matrix multiplication problem:

$$\underbrace{\begin{bmatrix} 21.0 \\ 21.0 \\ 22.8 \\ \vdots \\ 21.4 \end{bmatrix}}_{\text{mpg}} = \underbrace{\begin{bmatrix} 1 & 110 & 2.62 \\ 1 & 110 & 2.875 \\ 1 & 93 & 2.32 \\ \vdots & \vdots & \vdots \\ 1 & 109 & 2.78 \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \end{bmatrix}}_{\beta} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}}_{\varepsilon} = X\beta + \varepsilon$$

Multivariable Regression

$X\beta$ is an $n \times 1$ vector of predicted values, and ε is an $n \times 1$ vector of errors.

$$Y = X\beta + \varepsilon$$

Just like before, we want to minimize the sum of squared errors:

$$\varepsilon \cdot \varepsilon = \varepsilon'\varepsilon = (Y - X\beta)'(Y - X\beta)$$

Minimizing this expression follows the same three steps we used with scalar calculus. Just be careful with the multiplication and division. Start by distributing the function:

$$f(X, Y, \beta) = (Y - X\beta)'(Y - X\beta) = Y'Y - 2(X\beta)'Y + (X\beta)'X\beta$$

Estimating The Regression Parameters

Step 1: Take the derivative with respect to β

$$f(X, Y, \beta) = Y'Y - 2(X\beta)'Y + (X\beta)'X\beta$$

$$\frac{\partial f}{\partial \beta} = -2X'Y + 2X'X\beta$$

Step 2: Set the derivative equal to zero

$$-2X'Y + 2X'X\beta = 0$$

Step 3: Solve for β

$$2X'X\beta = 2X'Y$$

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Estimate The Multivariable Regression

```r
# create the Y vector
Y <- mtcars$mpg

# create the X matrix
X <- mtcars %>%
  select(hp, wt) %>%
  mutate(intercept = 1) %>%
  as.matrix

head(X)
```

```
      hp    wt intercept
[1,] 110 2.620         1
[2,] 110 2.875         1
[3,]  93 2.320         1
[4,] 110 3.215         1
[5,] 175 3.440         1
[6,] 105 3.460         1
```

Estimate The Multivariable Regression

The vector of estimates that minimizes the sum of squared errors equals $(X'X)^{-1}(X'Y)$:

```
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% Y

beta_hat
```

```
                 [,1]
hp        -0.03177295
wt        -3.87783074
intercept 37.22727012
```

```
lm_fit <- lm(mpg ~ hp + wt, data = mtcars)
coef(lm_fit)
```

```
(Intercept)          hp          wt
37.22727012 -0.03177295 -3.87783074
```

What's Going On Here?

Previously, we showed that the line of best fit between `hp` and `mpg` had this slope:

```
bivariate_fit <- lm(mpg ~ hp, data = mtcars)
coef(bivariate_fit)
```

```
(Intercept)          hp 
30.09886054 -0.06822828 
```

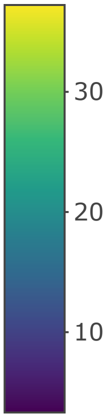
Now the slope is this half that...

```
multivariate_fit <- lm(mpg ~ hp + wt, data = mtcars)
coef(multivariate_fit)
```

```
(Intercept)          hp          wt 
37.22727012 -0.03177295 -3.87783074 
```

What's Going On Here?

What's Going On Here?



Exercise

Estimate a linear regression model from the `iris` dataset using Petal Length as the outcome variable and Sepal Width, Sepal Length, and Petal Width as the explanatory variables.

1. How does the coefficient on Sepal Width change from our previous bivariate regression with Petal Length and Petal Width alone? Why?

2. Conduct a t-test to see if the versicolor petals are significantly longer than setosa petals.

3. Try `lm(Petal.Length ~ Species, data = iris)`. Notice anything familiar?