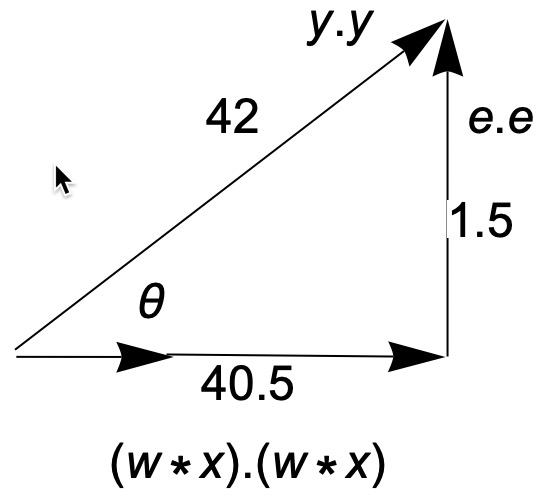
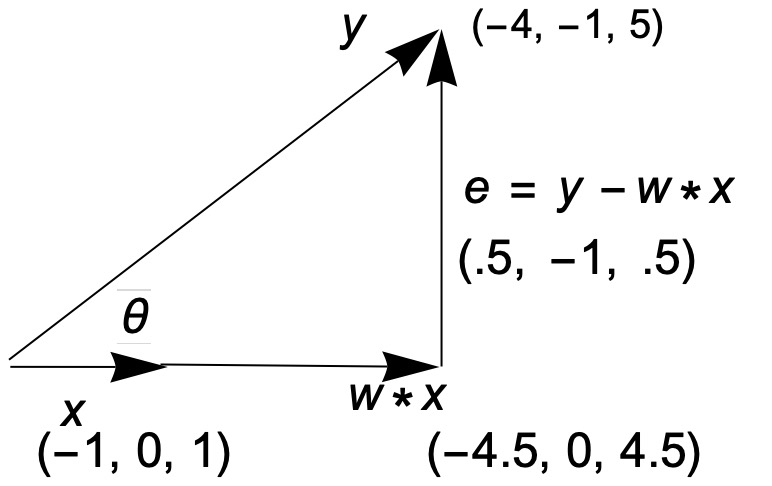
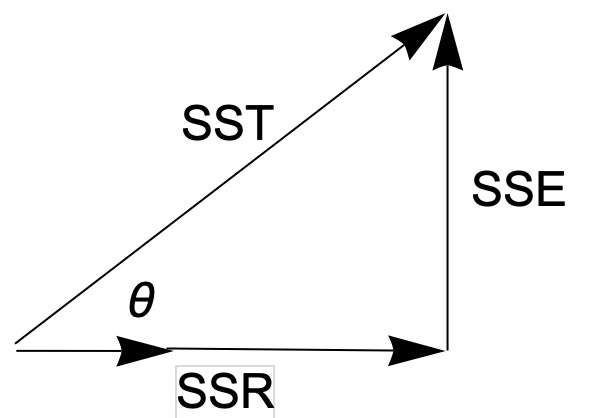


*Arizona State University*

*Polytechnic Campus*

*fulton schools*

**Linear Regression is a Right Triangle of Squared (centered) Vectors**



These right triangles show this documents’ main example using the centered feature vector **x** topredict centered target **y** (as close as possible). So, given a raw feature vector X = {3.0, 4.0, 5.0} , and then subtracting its’mean of 4 => x = { -1.0, 0.0, 1.0}, a “centered vector. Similarly for raw target vector Y = { 6.0, 9.0, 15.0} and then subtracting its’ mean of 10 => y = { -4, -1, 5}, a centered vector. So, given centered variables, every possible regression line goes through 0,0 and so its’ intercept “**b**” = 0. The slope **w** is easy to calc geometrically:

**Since x .( y – w \* x) == 0** at perpendicularity, **w** can be immediately calculated as: **w** = x.y/x.x = 4.5

”**b**” can then be calculated from “**w**” and the raw means: **b**= Ybar- **w**\* Xbar = -8. The “raw equation is then:

Y = 4.5 X -8

Since this is a right triangle ( when length of error is minimum), all the usual basic geometry holds. For example, adjacent vector length/hypotenuse vector length = cosine theta = **r**. Then SSR/SST is then cos2 theta ( **r2** in statistics).

The closest (linear) prediction is when w = 4.5, making the error vector perpendicular to **x,** and so a right triangle.

(Note: **w** is the regression coefficient and cosine of theta is the correlation coefficient ,r**.**

The squared lengths of each of these components equate to *variances*, and comprise the statistical triangle.

**abstract**: This doc lays out a subset of the fundamental structure of a geometric approach to *linear* regression. I use a detailed toy example so that the reader can follow along mentally. The context of this exposition is the concepts and operations within a Vector Space. Given the approach here, the reader can actually *see* the intuitive basis of calculations of the parameters: regression coefficient, the regression intercept, the correlation coefficient, various sums of squares such as SST, SSR, SSE, MSE, RMSE and their correlates. The intuition derives from visualizing/finding/analyzing, and synthesizing the properties of *right triangles* inherent in the data space, surprising, but true, what I call the statistical triangle.

This development will show that, when talking about *variance*, that notion can be recast as talking about the *squared length of a (centered) vector. In fact, the statistical triangle is a right triangle of variances!*

From the *intuition* and *vocabulary* gained from this exercise, the student is then prepared to apply these ideas in the context of data algorithms, such as *Spark LinearRegression, and Logistic Regresion.* So, the plan is: first gain intuition and a vocabulary for linear regression in the context of a detailed simple example, then apply those ideas and vocabulary to other algorithms. An actual Spark Linear Regression run of the example is shown in this document and equivalences drawn with the geometric approach.

An equivalent set of Scala 3 functions will be developed by the student to do these same calculations.

So, in summary: *Geometry for intuition, Statistics/Machine Learning packages for production.*

<IFT512AdvancedBigDataAnalytics/AI>

Draft version 4.11: 2023-09-22

r.r.

**Science before Statistics & Machine Learning**

Before starting to work on any project, it’s a good idea to stop and consider the *science* that

informs the effort. What *is* the goal, that is, what are the *estimand(s)*? What is to be calculated, discovered, explored, or inquired of? In particular, before running a statistics or machine learning package, what do I already know about the context and how does that influence what I can expect to know? In this simple case below I am going to state my assumptions in the form of a causal diagram ( a directed acyclic graph – DAG). This is a *minimal heuristic t*o describe some of the underlying assumptions I will use. This needs to come first in your explorations.

( as an aside: see your Spark documentation for a similar idea used to describe a Spark jobs’ execution plan).

*Goal- Objective- Estimand*

For this project, I am going to investigate the relation of the square-footage of a house to its’ market price ( say, for a real-estate agent). My assumed DAG, below, looks like this, and the arrow direction indicates *my* assessment of *causality*: Square-footage *influences* price. (This is the science before ML/statistics part, and, a minimal assessment of causality should be a pre-condition for any analytic work)!

*Square-Footage → Price*

this is a very modest influence diagram(dag = directed acyclic graph) and says the bare minimum:

1. I think that square footage feature *directly* influences price, some how?

2. I choose to represent this influence as some function ”f” : price = f ( square-footage)

3. My diagram doesn’t tell me what that function is and, I have to decide that based on other ( evidence based) considerations.

a. the simplest function to consider would be a *constant* and could be the average of all the

prices ( or just a swag)! (We will see how cost-effective its’ “SSE” in another discussion).

b. I could try a linear function price = w \* square-footage + b (w = slope, b= intercept)

c. I could consider a polynomial or splines, but I am going with linear for now

So, I decide to try a *linear function* as my *model*.

**model:** price **=** f w,b (square-footage) = w \* square-footage + b

The (made-up) example below is intended to provide the data to estimate/predict the price of a (new) home, given its square footage. The known ( labeled) ‘historical’ data on square footage versus prices is shown in the table below.

**Variable Space & Subject Space – a preliminary review**

A few vector space ideas are introduced first that the student already knows but maybe has not explicitly labeled. Let me use the training set below to illustrate these two ways to look at *any* data set.

This historical data set below is know as the *training data/set* since it will be used to calculate the  *parameters* (*estimands, w, b in this case*) of a *model* for home pricing. I will relabel this data in familiar terms of X and Y. The X values are the *features* while the Y values are the *labeled* points, also called the *historical* variable or sometimes the *ground truth*. So the *feature* of the first observation is its’ X value, ‘3’. The labeled point for that first observation is ‘6’. So, the first training example is ( 3, 6)

( In classical statistics, the X values are the independent variables while the Y values are the dependent values.

Price versus Square footage Training Data

|  |  |  |
| --- | --- | --- |
| Training example index “i”  i = 1… m  Observation Id of “m” training examples  m=3in this case | Square Footage(1000s) “X” (features)  j =1 . . . n  n=1 in this case | Price ($100000s)  “Y” (labeled point) |
| 1 | 3 | 6 |
| 2 | 4 | 9 |
| 3 | 5 | 15 |

*Variable Space ( X, Y) coordinates, is the**Scatter Plot Space*

Reading *across* this matrix, we see the pairs (3,6), (4,9), (5, 15). If I plot these I get the familiar *scatter* plot that is situated in *variable* space, since the plotting axes are the variables X and Y

The *scatter* plot is invaluable in order to detect outliers, spot a trend, as in this case where the prices follow the square footage in a roughly linear fashion. Big values of X seem to go with big values of Y, and hence, indicate a possible *association*. Also, the points don’t cluster about the origin, suggesting a non-zero mean. The caveat here is that the *scatter* plot emphasizes the *entities* at the expense of the *variables*. That is, it is hard to pick out the effect or the relation of the X variables to the Y variables, since the entities get in the way! It gets much worse when there are three or more features. Scatter plots are necessary, but not sufficient (when you can actually plot them). Below, I show the scatter plot of the entities together with the same entities with their means subtracted, called *centered* variables or *scaled* variables. I use lower case for these centered/scaled variables

and **bold** for the fact that they are vectors. So, **x**, **y** are the centered variables.

That is: **x** = **X** – mean(**X)**, **y** = **Y –** mean(**Y).**  Where I will write mean(X) = Xbar and mean(Y) =Ybar

*Raw Variable-Space Plot: (* showingraw variables and their centered counterparts)



Notice that the centered variables have the same standard deviation and correlations as do the raw data. That is, the interrelations between X and Y remain the same after this shift of origin and we will find that the centered variables are easier to work with. (We will transform back to the raw perspective subsequently). Plus which, a linear fit would go through the origin, making “b” = 0. We will use that fact to calculate the slope of a “centered line” and then separately calculate its “b” intercept).

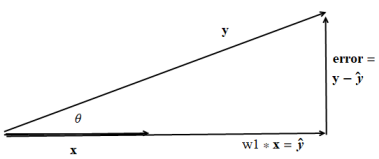
*Subject Space Plot ( switch the axes from variables to subjects/observations/entities )*

Now read *down* the table “Price versus Square footage Training Data”. Now X = {3,4,5}, Y = { 6, 9 15}. The axes are now the subjects/entities, of which there are three. Along each subject axes is plotted its X value and Y value. These are collected as the two vectors X, Y which show how the *variables* relate to each other. **But**-- what if there were 15 features? That would mean 15 subject/entity axes! Whoa! Finding 15 dimensional graph paper is tough, even using Google. The key idea though, is that we don’t have to work in higher dimensional space, since there are really only *three* objects to plot: an *origin*, and two *vectors as points* as shown below. Even if X, Y had 100 components, there would still be only three objects to plot. And, even better for us, given any two vectors from a common origin, they form a 2D *plane* we can easily work within!

X , Y are raw vectors, x, y are their centered vectors and Xbar, Ybar are their constant component vectors. e.g. X = ( 3,4,5) Xbar = (4, 4, 4) , x = (-1, 0, 1)

Y = (6, 9, 15) Ybar = (10,10,10) y = (-4, -1, 5)

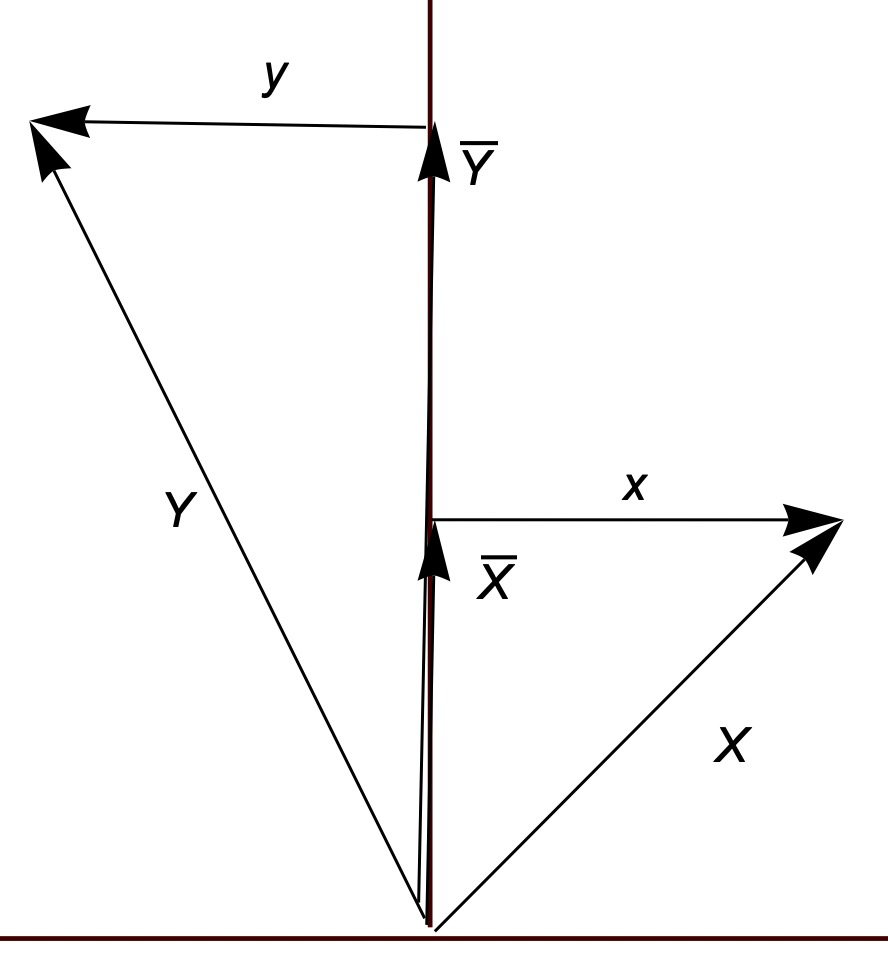
The Statistical Triangle I have so far:



This is the centered variable triangle that will result **if** ,I pick the right multiple of **x** ( the “w”) that gets as close as possible to the tip of **y** and so results in a right triangle.

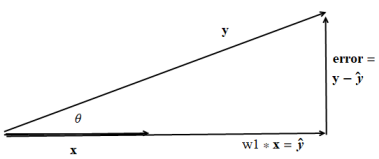
**A Tale of two complementary vector spaces**

As far as vector spaces go, the centered  **x, y** vectors in general, are perpendicular to their constant space vectors **Xbar**, **Ybar**. These two *vector spaces* are *orthogonal.* That means we can optimize in the two spaces independently, as we we will see.



It will be convenient and easier to analyze the X, Y relationship, if we first scale these two vectors by subtracting off their means. That is, we will see that we can break up any vector into two components: its’ *mean* vector and the vector that represents the *variation* about that mean ( see diagram above). It turns out that these two components are in perpendicular ( orthogonal) vector spaces to each other and so we can do **optimization** analyses separately on the means and the variations, as you will see in a moment. The statistical triangle below explicitly shows all the linear statistics we might want, and, I/you can do most of it mentally! Just stay with me for a few more lines. Plotting the **centered** vectors we get:

**The centered variables describing the The Statistical Triangle:**



-4, -1, 5

.5, -1, .5

-1, 0, 1

-4.5, 0, 4.5

*micro-summary of the statistical triangle*

x, y are the centered variables:

w1 will be calculated below and shown to be 4.5, so 4.5 \* x is the regression line

w1 is the regression coefficient ( and is the best we can do)

y-yhat is the error ( residual) vector and is of minimum length when the

regression vector is 4.5 \* x, i.e when 4.5 \* x is perpendicular to the error vector

This error-squared length =( .25 +1 + .25) = 1.5 = SSE in statistics ( error variance)

The squared length of 4.5\* x = = 40.5 = SSR in statistics ( regression variance)

The squared length of y = 42.0 =SST in stats // that’s life! (total variance)

cosine theta is the correlation coefficient **r ,** length of regression line / length of y

and of course, ol’ Pythagoras weighs in with a right triangle!

SST = SSR + SSE

**How this all works-- Least Squares ( Gauss’ discovery)**

*Calculating the regression coefficient, w1, in variation space, using centered vectors*

**1.** Since **x** and **y** aren’t on the same line, we can’t extend **x** to exactly match **y,** we can only extend **x** so that it is as *close as possible*. That means that **extended x** is directly under the tip of **y**. That is, **extended x** is perpendicular to **(y – yhat)**. Where yhat = w1 \* x

This happens only when the length of the error vector is minimized, that is, this minimizes the sum of squares error. So, we extend **x** by an amount that will get closest to the tip of **y**, That extension scalar w1, is the scalar *regression coefficient, w1.*

**x** = { -1, 0, 1) // the centered X vector

**y** = { -4,-1, 5} // the centered Y vector

Further, the vector **x** and the **error** vector are orthogonal. That means that the dot product of **x** with the **error** vector, equates to zero. That’s the geometry and that will give us the equation for w1.

**yhat** is the best estimate of y and is w1 \* x

The dot product will be indicated by a “dot”

x. y = |x| |y| Cos theta

**x** . (**y-yhat**) = **x** . **y** – **x** . w1 \***x**  ==0 // the dot product is distributive

w1 = **x** . **y** /(**x** . **x**) = 9/2 = 4.5 this is the regression coefficient,

**x** . **y** = 4 + 0 + 5 = 9

**x** . **x** = 1 + 0 + 1 = 2

**error** = **y** – **yhat** = {0.5, -1, 0.5}// residual vector

**error** . **error** = .25 + 1 + .25 = 1.5 // SSE

**y** . **y** = SST = 42 //that’s life!

**yhat** = w1 **x** = 4.5 \* **x**

**yhat** . **yhat** = SSR = 40.5

yhat is the regression vector

**NOTE: SST = SSR + SSE** Pythagoras!And is the basis for **the statistical triangle**

correlation coefficient = r = cosine *theta* =

1. length of the regression line over length of y : |w1 \* x| / |y| ( adjacent over hypotenuse!)

where vertical bars indicate Euclidean norm (l^2) ( the usual length calculation)

*another way to calc cosine theta*

2. x . y / ( norm(x) \* norm ( y)) since x.y = |x| \* |y| \* cos theta

= .981981

r^2 = r \* r = .9642n == (cos theta)^2 == SSR/SST

Notice that since r = cosine theta, then cosine ^2 = r2

The proportion of y regression variance accounted for by w\* x regression variance is the below ratio

**yhat** . **yhat** / **y** . **y** = SSR/SST i.e. sum of squares regression /total sum squares

MSE = **error** . **Error** / 3 = 1.5/3 = 0.5 = SSE/3

RMSE = sqrt ( MSE) = 0.707 sqrt(SSE/3)

If we only wanted the regression coefficient, we could stop here. If, we also wanted the regression intercept, “b”, we can get that easily as follows. **Ybar** = b + w1 \* **Xbar**

Intercept: b = 10 – 4.5 \* 4 = -8

The raw final equation is:

Y = 4.5 X - 8

**Actual Machine Learning (Spark Linear Regression ) run**

Just to see how this actually works within ML, I have put the data into a *DataFrame*, invoked an *Estimator*, then used the constructed *Model* to potentially evaluate additional DataFrame Inputs.

I used the simple data above for the single *features* and *label* of this DataFrame.

Note that this linear regression uses a denominator of “n” for the MSE, while a more accurate value would be n-1, for samples. Of course, if there are more than hundreds of data points, this doesn’t matter.

Note: Spark is looking for particular types of data columns with specific names, so you have to transform your raw data into that format:

1. The *feature* value(s) column must be Spark *Dense vectors*.

Below. I constructed these feature vectors by hand, usually best to use *VectorAssembler*

2.The feature value column must be named: *features*

*3.*  The labeled column must be type double and labeled as *label*

Note: I set up these names when I created the DataFrame, see below:

( Note: since Spark is written in Scala, I can take any Scala Sequence and construct a DataFrame, or a Dataset from it. See below. The student will be well advised to learn this *fused* OOP/FP **production** level language).

**Running IntelliJ Community Edition ( Spark does not yet support Scala 3, so I used Scala 2 here)**

\*\* You will need a couple of imports to get the needed algorithms plus a build file with the

Spark libraries\*\*\*

import org.apache.spark.ml.regression.{LinearRegression}

import org.apache.spark.ml.linalg.{ Vectors, Vector}

//Setting up training data as a DataFrame, as is necessary for ML computations.

val trainingData = spark.createDataFrame(Seq(

(6.0, Vectors.dense(3.0)),

(9.0, Vectors.dense(4.0)),

(15.0, Vectors.dense(5.0))

)).toDF("label", "features") // this created a DataFrame and named its columns

trainingData.show() //> +-----+--------+

//| |label|features|

//| +-----+--------+

//| | 6.0| [3.0]|

//| | 9.0| [4.0]|

//| | 15.0| [5.0]|

//| +-----+--------+

val lr = new LinearRegression() // here is the estimator

val lrModel = lr.fit(trainingData) // here is the parameterized model

//lrModel : org.apache.spark.ml.regression.LinearRegressionModel

println(s"Coefficients: ${lrModel.coefficients} Intercept: ${lrModel.intercept}")

//> Coefficients: [4.500000000000027] Intercept: -8.000000000000108

val trainingSummary = lrModel.summary

trainingSummary.residuals.show() //> +-------------------+

//| | residuals|

//| +-------------------+

//| | 0.5000000000000284|

//| |-0.9999999999999982|

//| |0.49999999999997335|

//| +-------------------+

println(s"RMSE: ${trainingSummary.rootMeanSquaredError}")

//> RMSE: 0.707106781186547

println(s"r2: ${trainingSummary.r2}") //> r2: 0.9642857142857143

Compare these outcomes with the geometric development

Now, this model can make new predictions when new *sqft* data comes in.

Finally, these *geometric* ideas can be generalized to analyze additional features sets.

**Generalizing to multivariate linear regression**

Consider two feature vectors **x1** and **x2** , centered and **y** centered.

The regression coefficients w1 and w2 are found when w1 and w2 are chosen to make the regression line extend to the tip of **y**. That geometry means that **x1** and **x2** are perpendicular to the error vector

yhat = w1 **x1** + w2 **x2**

error = y - ( w1 **x1** + w2 **x2**)

**x1** . error == 0

**x2** . error == 0

Two equations and two unknowns, w1, w2

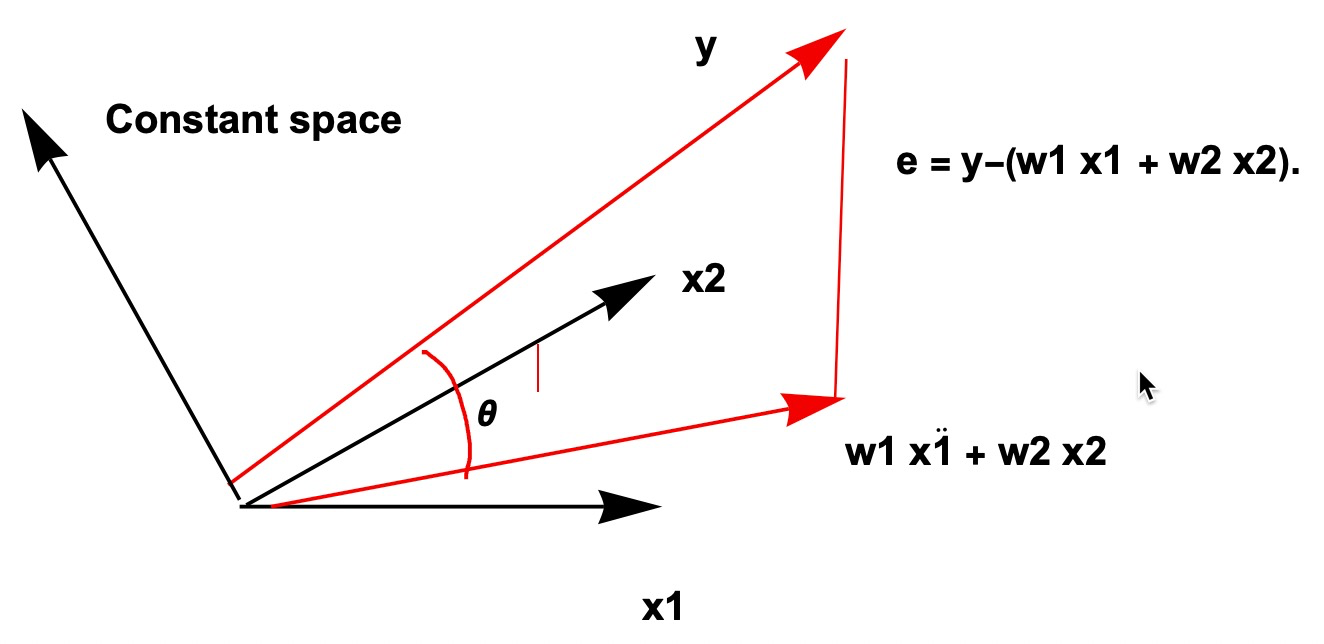
Intercept b = Ybar – (w1 \*X1bar + w2 \* X2bar)

That’s it! All the same stats can be calculated from the red statistical triangle below

**yhat** = w1 **x1** + w2 **x2** an estimate of **y**

\

Statistical triangle lies in a plane formed by y and yhat



**References:**

Cohen, M.(2018) *Linear Algebra- principles, theory, code*

McElreath, R.(2023) *Statistical Rethinking*, CRC

Wickens,T, (1984) *Geometric Multivariate Analysis*