Linear Algebra

Bill Chung

May 26, 2022

Welcome

House keeping

- Please be on time and `turn your camera on`
- Please free to ask questions any time.

Recommendd Books

- `Linear Algebra and its application` by David C.Lay 4th edition
- Linear and Nonlinear Programming by Stephen G. Nash and Ariela Sofer
- The fundamental theorem of linear algebra, Strang, Gilbert
- if you can read Korean: :LINK TO RIDI:

Dancing with Wu Li Masters

- `Young man, in mathmatics, you don't understand things. You just get used to them` by John Von Neumann from *Dancing with Wu Li Masters*

Who is John Von Neumann?

- Leonoid Kantorovich (1912 - 1986): `A new method of solving some classes of extrmal problems (1937)`
- George B. Dantzig (1914 - 2005) : `SIMPLEX (1947)`
- Jerzy Neyman (1894 - 1982) : `Confidence Interval, P-value`
- John Von Neumann : `The duality theorem (1947)`

Schedule

Week	Topic	Key concepts
1	Attributes and method of `vector` and `matrix`	see notes below
2	Slight detour to probabilities: Joint, conditional, marginal and Bayes formula. Markov chain, eigenvalue, eigenvectors	Linear combinations
3	What is rref(A) and what does it tell you about your matrix?	Basis, subspace, space, span, projection, inverse
4	Fundamental four subspaces of matrix. Given a vector, can you find out where it `lives`?	Shall we span?
5	Projection, projection, projection	linear combination, change of basis
6	Findings vector multiplication that looks like projection	projection, orthogonal matrix, spanning Space
7	Change of basis and solving systems of equations	matrix decomposition
8	It does not matter how slowly you move as long as you are making progress	eignevalue, eigenvector, Markov chain
9	Eignedecomposition	eigenvalue, eigenvector, eigenspace, nullspace
10	Markov chain	irreducible, reduccible, ergodic, regular, absorbing MC. What type of matrix do you have?
11	Singular value decomposition	SVD and PCA
12	Meeting matrix again	PSD, PD, ID, NSD, ND, Condition number, symmetric matrix, gram matrix, diagonailzable matrix
13	SIMPLEX method and The duality theorem	*The Martians*

Notations

$$\mathbb{A} \cdot \vec{x} = \vec{b}$$

$$\vec{v}$$

$$\mathbb{A}$$

vectors

Attribute

- Size of a vector
- Direction that it can `move`
- Direction that it can `see`
- Norm
- Subspace where it `lives`
- Space where it `lives`

Method

- Span
- linear combination
- transpose
- dot product
- projection

Space

- Contains ∞ number of subspaces

Subspace

- Created by spanning a vector or set of vectors
- Always contains $\vec{0}$ and closed under `addition` and `multiplication`
- basis
- Has orthogonal complement subspace (`they are like best friends`)

matrix

Attribute

- Dimension of matrix
- Column Space, $C(\mathbb{A})$, Left Nullspace, $N(\mathbb{A}^T)$
- Row Space, $R(\mathbb{A})$, Nullspace, $N(\mathbb{A})$
- Input space (related to domain)
- Output space (related to codomain and Range)
- basis

- eigenvalue, eignevector
- singular value, singular vector
- condition number
- Rank
- PD, PSD, ID, ND, NSD
- Rank-nullity theorem
- inverse (not every square matrix has it..)
- Gram matrix

method

- transpose
- inverse
- decomposition
    - singular value decomposition
    - eigen decomposition
- projection
- rref($\mathbb{A}$)

Solving systems of equations

- Homogeneous equations
- Homogeneous equations
- Augmented matrix

How to create matrix and vector in R

```
a1 <- matrix(c(3,0,-1,-5,2,4), nrow =1, byrow = T)

a2 <- matrix(c(3,0,-1,5,2,4), nrow =1, byrow = T)

A <- rbind(a1,a2)

print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1   -5    2    4
## [2,]    3    0   -1    5    2    4
```

```
print(a2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1    5    2    4
```

```
a1 <- matrix(c(3,0,-1,-5,2,4),nrow=1,byrow=T)
print(a1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1   -5    2    4
```

```
a2 <- matrix(c(3,0,-1,5,2,4),nrow=1,byrow=T)

A <- rbind(a1,a2)
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1   -5    2    4
## [2,]    3    0   -1    5    2    4
```

```
Rank(A)
```

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

Definiations

Linear combination

$$\mathbb{A}\vec{x} = \vec{b}$$

Subspace

- If $\vec{v}_1, ..\vec{v}_p \in R^n$, then Span$\{\vec{v}_1, ..\vec{v}_p\}$ is called the subset of R^n by these vectors.

Linear combination, Projection and transformation

$$\mathbb{A}\vec{x} = \vec{b}$$

How to create a matrix

```
a1 <- matrix(c(3,0,-1,-5,2,4), nrow =1, byrow = T)

a2 <- matrix(c(3,0,-1,5,2,4), nrow =1, byrow = T)

A <- rbind(a1,a2)

x <- c(5,-2,3,-2,5,-1.3)
x
```

```
## [1]  5.0 -2.0  3.0 -2.0  5.0 -1.3
```

```
b<- x/Norm(x)

Norm(b)
```

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

```
A%*%x
```

```
##      [,1]
## [1,] 26.8
## [2,]  6.8
```

```
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1   -5    2    4
## [2,]    3    0   -1    5    2    4
```

```
B <- A[,c(1,2,5)]
D <- A[,-c(1,2,5)]

B
```

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

```
D
```

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

select columns 1, 3 and 6 and put them into $\mathbb{B}$
select columns 2, 4 and 5 and put them into $\mathbb{N}$

```
B <- A[,c(1,3,6)]
B
```

```
##      [,1] [,2] [,3]
## [1,]    3   -1    4
## [2,]    3   -1    4
```

$$\mathbb{B} \cdot \vec{x}_B + \mathbb{N} \cdot \vec{x}_N = \mathbb{A} \cdot \vec{x}$$

Creating sample vector

```
#randomly selects number
a <- sample(-5:5, replace=TRUE, 12)
#find out number of elements in the vector
length(a)
```

```
## [1] 12
```

```
A <- matrix(a, ncol = 4, byrow= TRUE)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    4    1    3
## [2,]   -4    0    5    3
## [3,]    4    2   -2    1
```

```
A <- matrix(sample(-5:5, replace=TRUE, 12), ncol = 4, byrow= TRUE)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    0    5
## [2,]    2   -2   -3    1
## [3,]    3    2    2   -2
```

```r
b <- matrix(sample(-5:5, replace=TRUE, 3), ncol = 1, byrow= TRUE)

H <- cbind(A,b)
rref(H)
```

```
##      [,1] [,2] [,3]      [,4] [,5]
## [1,]    1    0    0 -1.272727   -3
## [2,]    0    1    0  6.272727    7
## [3,]    0    0    1 -5.363636   -5
```

- Go over solving systems of equations with inf solutions
- How to pick one solution

Problems

example 1

```r
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    0    5
## [2,]    2   -2   -3    1
## [3,]    3    2    2   -2
```

```r
dim(A)
```

```
## [1] 3 4
```

```r
Rank(A)
```

```
## [1] 3
```

```r
a1 <- matrix(c(3,0,-1,-5,2,4), nrow =1, byrow = T)
a2 <- matrix(c(3,0,-1,5,2,4), nrow =1, byrow = T)
a3 <- matrix(c(3,0,-1,5,2,4), nrow =1, byrow = T)

A <- rbind(a1,a2,a3)

print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    0   -1   -5    2    4
## [2,]    3    0   -1    5    2    4
## [3,]    3    0   -1    5    2    4
```

```r
dim(A)
```

```
## [1] 3 6
```

```
rref(A)
```

```
##      [,1] [,2]       [,3] [,4]      [,5]     [,6]
## [1,]    1    0 -0.3333333    0 0.6666667 1.333333
## [2,]    0    0  0.0000000    1 0.0000000 0.000000
## [3,]    0    0  0.0000000    0 0.0000000 0.000000
```

```
B <- A[,c(1,4)]
D <- A[,-c(1,4)]

x <- c(1,2,5,-2.2,4,1)
b <- A%*%x
A%*%x
```

```
##      [,1]
## [1,]   21
## [2,]   -1
## [3,]   -1
```

```
G <- t(B)%*%B
invG <- inv(G)

xB <- invG%*%t(B)%*%b

B%*%xB
```

```
##      [,1]
## [1,]   21
## [2,]   -1
## [3,]   -1
```

problem to solve

```
a1 <- matrix(c(3,0,-1,-5,2,4,5), nrow =1, byrow = T)
a2 <- matrix(c(3,0,-1,5,2,4,3.5), nrow =1, byrow = T)
a3 <- matrix(c(3,0,-1,5,2,4,-2.2), nrow =1, byrow = T)

A <- rbind(a1,a2,a3)
A
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    3    0   -1   -5    2    4  5.0
## [2,]    3    0   -1    5    2    4  3.5
## [3,]    3    0   -1    5    2    4 -2.2
```

```
x <- c(1,-2,3,5,-5.5,-1,3)
b<- A%*%x
```

steps using rref

```
rref(A)
```

```
##      [,1] [,2]       [,3] [,4]      [,5]     [,6] [,7]
## [1,]    1    0 -0.3333333    0 0.6666667 1.333333    0
## [2,]    0    0  0.0000000    1 0.0000000 0.000000    0
## [3,]    0    0  0.0000000    0 0.0000000 0.000000    1
```

break A in to B and D

```
B <- A[,c(1,4,7)]
D <- A[,-c(1,4,7)]

G <- t(B)%*%B
invG <- inv(G)

xB <- invG%*%t(B)%*%b

B%*%xB
```

```
##       [,1]
## [1,] -25.0
## [2,]  20.5
## [3,]   3.4
```

```
A%*%x
```

```
##       [,1]
## [1,] -25.0
## [2,]  20.5
## [3,]   3.4
```

find xB

shows Ax BxB are the same

$$\mathbb{A}\vec{x} = \mathbb{B}\vec{x}_B$$

Conditional Probablity example

from live sectoin note in wk2

In each week of a class, you are either caught up or behind.

- The probability that you are caught up in Week 1 is 0.7.

- If you are caught up in a given week, the probability that you will be caught up in the next week is 0.7.
- If you are behind in a given week, the probability that you will be caught up in the next week is 0.4.
- **What is the probability that you are caught up in week 3?**
- `Identify as many ways to improve this proof as you can`:

Conditional probability with not so good notation

- If you are caught up in a week, there are two possibilities for the previous week: caught up and behind.
- Let $P(X)$ be the probability of being caught up.
  - In week 1, the probability of being caught up $P(X) = .7$.
  - In week 1, the probability of being behind is $P(Y) = 1 - .7 = .3$.
- We first break down the probability for week 2:

$$P(X) = .7 \cdot .7 + .3 \cdot .4 = .61$$

Now we can repeat the process for week 3:

$$P(X) = .61 * .7 + .39 * .4 = .583$$

- Let C_i be the event that you are caught up in week i.
  - Given:
    - $P(C_1) = 0.7$
    - $P(C_{i+1}|C_i) = 0.7$
- Let C_i^C be the event that you are behind in week i
  - $P(C_{i+1}|C_i^C) = 0.4$.
- `For week 2`, we can partition the sample space into $\{C_1, B_1\}$ and apply the law of total probability:

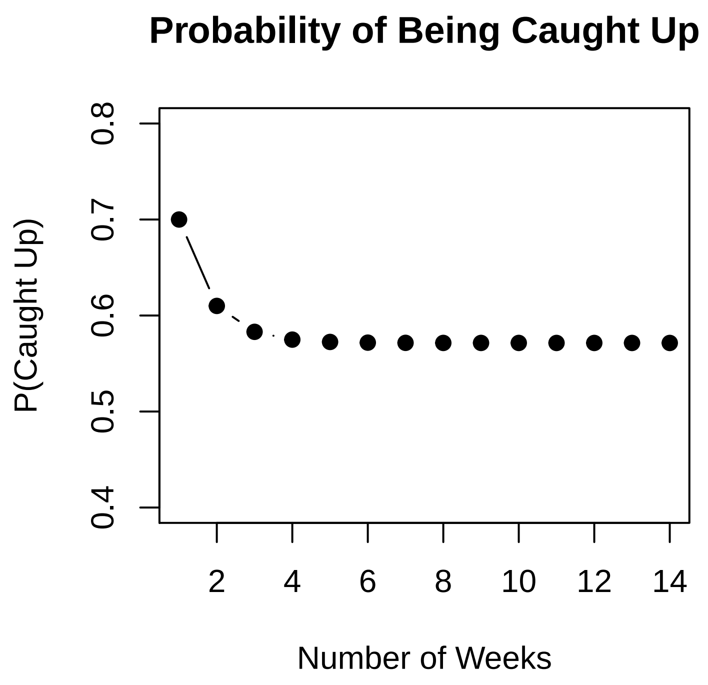
$$\begin{aligned} P(C_2) &= P(C_1)P(C_2|C_1) + P(B_1)P(C_2|B_1) \\ &= 0.7 \cdot 0.7 + 0.3 \cdot 0.4 = 0.61 \end{aligned}$$

- Next, repeat the process for `week 3`:

$$\begin{aligned} P(C_3) &= P(C_2)P(C_3|C_2) + P(B_2)P(C_3|B_2) \\ &= 0.7 \cdot 0.61 + 0.39 \cdot 0.4 = 0.58 \end{aligned}$$

Solving it using R

- You can write a function in R and solve it



Solving it using matrix

Given: - The probability of getting caught up with homework in this week only depends on the the outcome of the previous period.

- The transition matrix, $\mathbb{P}$, has nonzero values such that it is `regular`
- Since $\mathbb{P}$ is regular, it has limiting matrix

	C_i	C_i^C
C_{i+1}	0.7	0.4
C_{i+1}^C		

- Above matrix contains the given information:
- Let C_i be the event that you are caught up in week i.
  - $P(C_{i+1}|C_i) = 0.7$
- Let C_i^C be the event that you are behind in week i
  - $P(C_{i+1}|C_i^C) = 0.4$.
- Then, we can fill in the blank:

	C_i	C_i^C
C_{i+1}	0.7	0.4
C_{i+1}^C	0.3	0.6

And if we multiply the above matrix by the initial state vector, see what you get

$$[0.7, 0.3]^T$$

```
P <- matrix(c(0.7,0.4,0.3,0.6), nrow=2, byrow =T)
print(P)
```

```
##      [,1] [,2]
## [1,]  0.7  0.4
## [2,]  0.3  0.6
```

```
print(P%^%2)
```

```
##      [,1] [,2]
## [1,] 0.61 0.52
## [2,] 0.39 0.48
```

```
print(P%^%1000)
```

```
##           [,1]      [,2]
## [1,] 0.5714286 0.5714286
## [2,] 0.4285714 0.4285714
```

Solving it using eigenvalue

- Will talk about this more later in the class

```
######################
# Using eignevalues
######################
myeigen <- eigen(P)    #gets you the eigenvalues and eigenvectors

## getting the eigenvalues and eigenvectors into vector and matrix.

lambda <- myeigen$values   #eigenvalues

E <- myeigen$vectors     #corresponding eigenvectors

print(lambda)
```

```
## [1] 1.0 0.3
```

```r
print(E)
```

```
##      [,1]       [,2]
## [1,]  0.8 -0.7071068
## [2,]  0.6  0.7071068
```

```r
p_vector <- function(x){
y <- sum(abs(x))
x <- abs(x)/y
return(x)
}

#converting the eigenvector corresponding to eigenvalue = 1
p_vector(E[,1])
```

```
## [1] 0.5714286 0.4285714
```

Space and subspace

- Domain, codomain (Range, C($\mathbb{A}$))

Domain, codomain, Range

- You will see the following notation from time to time

$$T : R^n \rightarrow R^m$$

- the above notation is saying that `matrix T will be used to multiply vector with size of n and the resulting vector will have size m`
- And we will get into the details later.
- Vector resize within a space which consist of so many subspaces.
- When you put vectors into a matrix, you get two space, I call them `input` and `output space`. Input space can be divided into `row space` and `nullspace`, and `output space` can be divided into `column space` and `left null space`
- Think of domain as `row space` and `codomain` as `output space` and `range` as `column space`

Rank nullity theorem

If $\mathbb{A}$ has n columns, then $\text{Rank}(\mathbb{A}) + \dim \text{Nul}(\mathbb{A}) = n$

- see page 156 for the invertible matrix theorem (continued)

Invertible Linear Transformation

- A linear transformation $\mathbb{T} : R^n \rightarrow R^n$ is said to be `invertible` if there exists a funciton $\mathbb{S} : R^n \rightarrow R^n$ such that

$$\mathbb{S}(\mathbb{T}(\vec{x})) = \vec{x} \text{ for all } \vec{x} \text{ in } R^n$$
$$\mathbb{T}(\mathbb{S}(\vec{x})) = \vec{x} \text{ for all } \vec{x} \text{ in } R^n$$

where dim($\mathbb{A}$) = n by n, $\vec{x}, \vec{b} \in R^n$ - $\mathbb{A}$ is the standard matrix for $\mathbb{T}$

$$\mathbb{A}\vec{x} = \vec{b}$$

- $\mathbb{A}^{-1}$ is the standard matrix for $\mathbb{S}$

$$\mathbb{A}^{-1}\vec{b} = \vec{x}$$

```r
r1 <- c(0,1,-4)
r2 <- c(2,-3,2)
r3 <- c(5,-8,9)
A <- rbind(r1,r2,r3)
print(rref(A))
```

```
##    [,1] [,2] [,3]
## r1    1    0    0
## r2    0    1    0
## r3    0    0    1
```

- Above matrix is invertible matrix based on `rref()`
- Inverse transformation `undo` the transformation

```
x <- c(3,6,9)

#to use the same notation
T <- A
b<- T%*%x
print("Before the transformation")
```

```
## [1] "Before the transformation"
```

```
print(x)
```

```
## [1] 3 6 9
```

```
print("After the transformaiton")
```

```
## [1] "After the transformaiton"
```

```
print(T%*%x)
```

```
##    [,1]
## r1  -30
## r2    6
## r3   48
```

When $\mathbb{A}$ is not a square matrix

- With respect to $\mathbb{A}$, $\vec{b}$ is in your `range` and $\vec{x}$ is in `domain`.
- $\mathbb{A}$ transform vectors in domain to range.
- $\mathbb{A}^{-1}$ can transform values in range back to domain, when the $\mathbb{A}$ involved 1-1 transformation.

```
#chapter 1.1 example 3

r1 <- c(0,3,-6,6,4,-5)
r2 <- c(3,-7,8,-5,8,9)
r3 <- c(3,-9,12,-9,6,15)

A <- rbind(r1,r2,r3)

rref(A)
```

```
##    [,1] [,2] [,3] [,4] [,5] [,6]
## r1    1    0   -2    3    0  -24
## r2    0    1   -2    2    0   -7
## r3    0    0    0    0    1    4
```

Group exercise or Homework

- Break out session.

Part 1 (10 min)

(1) Create 3 by 3 nonsingular matrix, and call it $\mathbb{A}$

- What is the rank of $\mathbb{A}$

(2) Create 3 by 3 singular matrix and call it $\mathbb{F}$

- What is the rank of $\mathbb{F}$

(3) Can you express $\vec{v}_1 = [\ 6\ 4\ 1]$ as a linear combination of $\mathbb{A}$ or

$$\mathbb{F}$$

If not, what is the closest value you can express? How do you know?

Part 2 (15 min)

(1) 3 by 10 matrix given below and convert it to rref

```
set.seed(100)
A <- matrix(rnorm(30),ncol = 10)
print(A)
```

```
##              [,1]      [,2]       [,3]        [,4]       [,5]        [,6]
## [1,] -0.50219235 0.8867848 -0.5817907 -0.35986213 -0.2016340 -0.02931671
## [2,]  0.13153117 0.1169713  0.7145327  0.08988614  0.7398405 -0.38885425
## [3,] -0.07891709 0.3186301 -0.8252594  0.09627446  0.1233795  0.51085626
##             [,7]      [,8]       [,9]      [,10]
## [1,] -0.9138142 0.7640606 -0.8143791  0.2309445
## [2,]  2.3102968 0.2619613 -0.4384506 -1.1577295
## [3,] -0.4380900 0.7734046 -0.7202216  0.2470760
```

```
Rank(A)
```

```
## [1] 3
```

(2) Identify multiple basis (i.e., set of basis vectors that can span $C(\mathbb{A})$)

(3) Find 5 different solution to [6 2.5 3]

Part 3 (5 min)

(1) Identify column and row rank of the following matrix

```r
set.seed(100)
A <- matrix(rnorm(10),ncol = 2)
print(A)
```

```
##             [,1]       [,2]
## [1,] -0.50219235  0.3186301
## [2,]  0.13153117 -0.5817907
## [3,] -0.07891709  0.7145327
## [4,]  0.88678481 -0.8252594
## [5,]  0.11697127 -0.3598621
```

```r
Rank(A)
```

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

(2) Is the following vector in the span of C($\mathbb{A}$))?

```r
set.seed(110)
v1 <- matrix(rnorm(5),ncol = 1)
print(v1)
```

```
##           [,1]
## [1,] 0.2911952
## [2,] 1.3888632
## [3,] 0.6490100
## [4,] 1.4778760
## [5,] 0.4387201
```

```r
Ab <- cbind(A,v1)
Rank(Ab)
```

```
## [1] 3
```

(3)

- see page 173 for adding $\mathbb{I}$

```r
set.seed(100)
A <- matrix(rnorm(18),nrow=3)
print(A)
```

```
##             [,1]      [,2]       [,3]        [,4]       [,5]        [,6]
## [1,] -0.50219235 0.8867848 -0.5817907 -0.35986213 -0.2016340 -0.02931671
## [2,]  0.13153117 0.1169713  0.7145327  0.08988614  0.7398405 -0.38885425
## [3,] -0.07891709 0.3186301 -0.8252594  0.09627446  0.1233795  0.51085626
```

```r
Rank(A)
```

```
## [1] 3
```

```
rref(A)
```

```
##      [,1] [,2] [,3]       [,4]      [,5]         [,6]
## [1,]    1    0    0  1.3013129 3.1020226  0.867073274
## [2,]    0    1    0  0.2316328 1.6561283 -0.003415636
## [3,]    0    0    1 -0.1516676 0.1932849 -0.703259440
```

```
Rank(t(A))
```

```
## [1] 3
```

```
AT <- t(A)
print(AT)
```

```
##             [,1]        [,2]        [,3]
## [1,] -0.50219235  0.13153117 -0.07891709
## [2,]  0.88678481  0.11697127  0.31863009
## [3,] -0.58179068  0.71453271 -0.82525943
## [4,] -0.35986213  0.08988614  0.09627446
## [5,] -0.20163395  0.73984050  0.12337950
## [6,] -0.02931671 -0.38885425  0.51085626
```

```
rref(AT)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
## [4,]    0    0    0
## [5,]    0    0    0
## [6,]    0    0    0
```

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

nullspace <- -inv(t(B)%*%B)%*%t(B)%*%D
I <- diag(3)
nullspace <- rbind(nullspace,I)

dim(nullspace)
```

```
## [1] 6 3
```

```
round(A%*%nullspace,2)
```

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

Fundamental four subspaces

Fundamental four subspaces of Matrix

- $R(\mathbb{A}), N(\mathbb{A}), C(\mathbb{A}), N(\mathbb{A}^T)$

Column space basis

use rref($\mathbb{A}$)

```
A = matrix(c(-3,6,-1,1,-7,1,-2,2,3,-1,2,-4,5,8,-4),nrow=3,ncol=5,byrow=TRUE)
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   -3    6   -1    1   -7
## [2,]    1   -2    2    3   -1
## [3,]    2   -4    5    8   -4
```

```
dim(A)
```

```
## [1] 3 5
```

```
Rank(A)
```

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

```
rref(A)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1   -2    0   -1    3
## [2,]    0    0    1    2   -2
## [3,]    0    0    0    0    0
```

```
b1 <- c(9,4,3)

Ab <- cbind(A,b1)
Rank(Ab)
```

```
## [1] 3
```

```
B <- A[,c(1,3)]
N <- A[,-c(1,3)]

B
```

```
##      [,1] [,2]
## [1,]   -3   -1
## [2,]    1    2
## [3,]    2    5
```

```r
N
```

```
##      [,1] [,2] [,3]
## [1,]    6    1   -7
## [2,]   -2    3   -1
## [3,]   -4    8   -4
```

```r
#BXb + NXn = B1

G <- (t(B)%*%B)

#B^T*B*Xb  = B^T*B1
#inv(G)B^T*B*Xb  = inv(B)*B^T*B1
#Xb

## estimated coefficients
x_hat<- inv(G)%*%t(B)%*%b1
x_hat
```

```
##           [,1]
## [1,] -3.692308
## [2,]  2.312821
```

```r
## estimated value
B%*%x_hat
```

```
##           [,1]
## [1,] 8.7641026
## [2,] 0.9333333
## [3,] 4.1794872
```

```r
#using all features
x <- c(x_hat[1],0,x_hat[2],0,0)
A%*%x
```

```
##           [,1]
## [1,] 8.7641026
## [2,] 0.9333333
## [3,] 4.1794872
```

$$\mathbb{A}\vec{X} = \vec{b}_1$$

```r
library(wooldridge)
wooldridge::hprice1
```

```
##      price assess bdrms lotsize sqrft colonial   lprice  lassess  llotsize
## 1  300.000  349.1     4    6126  2438        1 5.703783 5.855359  8.720297
## 2  370.000  351.5     3    9903  2076        1 5.913503 5.862210  9.200593
## 3  191.000  217.7     3    5200  1374        0 5.252274 5.383118  8.556414
## 4  195.000  231.8     3    4600  1448          1 5.273000 5.445875  8.433811
## 5  373.000  319.1     4    6095  2514          1 5.921578 5.765504  8.715224
## 6  466.275  414.5     5    8566  2754          1 6.144775 6.027073  9.055556
## 7  332.500  367.8     3    9000  2067          1 5.806640 5.907539  9.104980
## 8  315.000  300.2     3    6210  1731          1 5.752573 5.704449  8.733916
## 9  206.000  236.1     3    6000  1767          0 5.327876 5.464255  8.699514
## 10 240.000  256.3     3    2892  1890          0 5.480639 5.546349  7.969704
## 11 285.000  314.0     4    6000  2336          1 5.652489 5.749393  8.699514
## 12 300.000  416.5     5    7047  2634          1 5.703783 6.031887  8.860357
## 13 405.000  434.0     3   12237  3375          1 6.003887 6.073044  9.412219
## 14 212.000  279.3     3    6460  1899          0 5.356586 5.632287  8.773385
## 15 265.000  287.5     3    6519  2312          1 5.579730 5.661223  8.782476
## 16 227.400  232.9     4    3597  1760          1 5.426711 5.450609  8.187856
## 17 240.000  303.8     4    5922  2000          0 5.480639 5.716370  8.686430
## 18 285.000  305.6     3    7123  1774          1 5.652489 5.722277  8.871084
## 19 268.000  266.7     3    5642  1376          1 5.590987 5.586124  8.637994
## 20 310.000  326.0     4    8602  1835          1 5.736572 5.786897  9.059750
## 21 266.000  294.3     3    5494  2048          1 5.583496 5.684599  8.611412
## 22 270.000  318.8     3    7800  2124          1 5.598422 5.764564  8.961879
## 23 225.000  294.2     3    6003  1768          0 5.416101 5.684260  8.700015
## 24 150.000  208.0     4    5218  1732          0 5.010635 5.337538  8.559870
## 25 247.000  239.7     3    9425  1440          1 5.509388 5.479388  9.151121
## 26 275.000  294.1     3    6114  1932          0 5.616771 5.683920  8.718336
## 27 230.000  267.4     3    6710  1932          0 5.438079 5.588746  8.811355
## 28 343.000  359.9     3    8577  2106          1 5.837730 5.885826  9.056840
## 29 477.500  478.1     7    8400  3529          1 6.168564 6.169820  9.035987
## 30 350.000  355.3     4    9773  2051          1 5.857933 5.872962  9.187379
## 31 230.000  217.8     4    4806  1573          1 5.438079 5.383577  8.477620
## 32 335.000  385.0     4   15086  2829          0 5.814130 5.953243  9.621523
## 33 251.000  224.3     3    5763  1630          1 5.525453 5.412984  8.659213
## 34 235.000  251.9     4    6383  1840          1 5.459586 5.529032  8.761394
## 35 361.000  354.9     4    9000  2066          1 5.888878 5.871836  9.104980
## 36 190.000  212.5     4    3500  1702          0 5.247024 5.358942  8.160519
## 37 360.000  452.4     4   10892  2750          1 5.886104 6.114567  9.295784
## 38 575.000  518.1     5   15634  3880          1 6.354370 6.250168  9.657204
## 39 209.001  289.4     4    6400  1854          1 5.342339 5.667810  8.764053
## 40 225.000  268.1     2    8880  1421          0 5.416101 5.591360  9.091557
## 41 246.000  278.5     3    6314  1662          1 5.505332 5.629418  8.750525
## 42 713.500  655.4     5   28231  3331          1 6.570182 6.485246 10.248176
## 43 248.000  273.3     4    7050  1656          1 5.513429 5.610570  8.860783
## 44 230.000  212.1     3    5305  1171          0 5.438079 5.357058  8.576406
## 45 375.000  354.0     5    6637  2293          1 5.926926 5.869297  8.800415
## 46 265.000  252.1     3    7834  1764          1 5.579730 5.529826  8.966228
## 47 313.000  324.0     3    1000  2768          0 5.746203 5.780744  6.907755
## 48 417.500  475.5     4    8112  3733          0 6.034285 6.164367  9.001100
## 49 253.000  256.8     3    5850  1536          1 5.533390 5.548297  8.674197
## 50 315.000  279.2     4    6660  1638          1 5.752573 5.631928  8.803875
## 51 264.000  313.9     3    6637  1972          1 5.575949 5.749074  8.800415
## 52 255.000  279.8     2   15267  1478          0 5.541264 5.634075  9.633449
## 53 210.000  198.7     3    5146  1408          1 5.347107 5.291796  8.545975
## 54 180.000  221.5     3    6017  1812          1 5.192957 5.400423  8.702344
## 55 250.000  268.4     3    8410  1722          1 5.521461 5.592478  9.037177
## 56 250.000  282.3     4    5625  1780          1 5.521461 5.642970  8.634976
## 57 209.000  230.7     4    5600  1674          1 5.342334 5.441118  8.630522
## 58 258.000  287.0     4    6525  1850         1 5.552959 5.659482  8.783396
## 59 289.000  298.7     3    6060  1925         1 5.666427 5.699440  8.709465
## 60 316.000  314.6     4    5539  2343         0 5.755742 5.751302  8.619569
## 61 225.000  291.0     3    7566  1567         0 5.416101 5.673323  8.931419
## 62 266.000  286.4     4    5484  1664         1 5.583496 5.657390  8.609590
## 63 310.000  253.6     6    5348  1386         1 5.736572 5.535758  8.584478
## 64 471.250  482.0     5   15834  2617         1 6.155389 6.177944  9.669915
## 65 335.000  384.3     4    8022  2321         1 5.814130 5.951424  8.989944
## 66 495.000  543.6     4   11966  2638         1 6.204558 6.298213  9.389825
## 67 279.500  336.5     4    8460  1915         1 5.633002 5.818598  9.043104
## 68 380.000  515.1     4   15105  2589         1 5.940171 6.244361  9.622781
## 69 325.000  437.0     4   10859  2709         0 5.783825 6.079933  9.292749
## 70 220.000  263.4     3    6300  1587         1 5.393628 5.573674  8.748305
## 71 215.000  300.4     3   11554  1694         0 5.370638 5.705115  9.354787
## 72 240.000  250.7     3    6000  1536         1 5.480639 5.524257  8.699514
## 73 725.000  708.6     5   31000  3662         0 6.586172 6.563291 10.341743
## 74 230.000  276.3     3    4054  1736         1 5.438079 5.621487  8.307459
## 75 306.000  388.6     2   20700  2205         0 5.723585 5.962551  9.937889
## 76 425.000  252.5     3    5525  1502         0 6.052089 5.531411  8.617039
## 77 318.000  295.2     4   92681  1696         1 5.762052 5.687653 11.436919
## 78 330.000  359.5     3    8178  2186         1 5.799093 5.884714  9.009203
## 79 246.000  276.2     4    5944  1928         1 5.505332 5.621125  8.690138
## 80 225.000  249.8     3   18838  1294         0 5.416101 5.520660  9.843632
## 81 111.000  202.4     4    4315  1535         1 4.709530 5.310246  8.369853
## 82 268.125  254.0     3    5167  1980         1 5.591453 5.537334  8.550048
## 83 244.000  306.8     4    7893  2090         1 5.497168 5.726196  8.973732
## 84 295.000  318.3     3    6056  1837         1 5.686975 5.762994  8.708805
## 85 236.000  259.4     3    5828  1715         0 5.463832 5.558371  8.670429
## 86 202.500  258.1     3    6341  1574         0 5.310740 5.553347  8.754792
## 87 219.000  232.0     2    6362  1185         0 5.389072 5.446737  8.758098
## 88 242.000  252.0     4    4950  1774         1 5.488938 5.529429  8.507143
##      lsqrft
## 1  7.798934
## 2  7.638198
## 3  7.225482
## 4  7.277938
## 5  7.829630
## 6  7.920810
## 7  7.633853
## 8  7.456455
## 9  7.477038
## 10 7.544332
## 11 7.756196
## 12 7.876259
## 13 8.124150
## 14 7.549083
## 15 7.745868
## 16 7.473069
## 17 7.600903
## 18 7.480992
## 19 7.226936
## 20 7.514800
## 21 7.624619
## 22 7.661057
## 23 7.477604
## 24 7.457032
## 25 7.272398
## 26 7.566311
## 27 7.566311
## 28 7.652546
## 29 8.168770
## 30 7.626083
## 31 7.360740
## 32 7.947679
## 33 7.396335
## 34 7.517521
## 35 7.633369
## 36 7.439559
## 37 7.919356
## 38 8.263591
## 39 7.525101
## 40 7.259116
## 41 7.415777
## 42 8.111028
## 43 7.412160
## 44 7.065613
## 45 7.737616
## 46 7.475339
## 47 7.925880
## 48 8.224967
## 49 7.336937
## 50 7.401231
## 51 7.586803
## 52 7.298445
## 53 7.249926
## 54 7.502186
## 55 7.451241
## 56 7.484369
## 57 7.422971
## 58 7.522941
## 59 7.562681
## 60 7.759187
## 61 7.356918
## 62 7.416980
## 63 7.234177
## 64 7.869784
## 65 7.749753
## 66 7.877776
## 67 7.557473
## 68 7.859027
## 69 7.904335
## 70 7.369601
## 71 7.434848
## 72 7.336937
## 73 8.205765
## 74 7.459339
## 75 7.698483
## 76 7.314553
## 77 7.436028
## 78 7.689829
## 79 7.564239
## 80 7.165493
## 81 7.336286
## 82 7.590852
## 83 7.644919
## 84 7.515889
## 85 7.447168
## 86 7.361375
## 87 7.077498
## 88 7.480992
```

```
library(tidyverse)
glimpse(hprice1)
```

```
## Rows: 88
## Columns: 10
## $ price    <dbl> 300.000, 370.000, 191.000, 195.000, 373.000, 466.275, 332.500~
## $ assess   <dbl> 349.1, 351.5, 217.7, 231.8, 319.1, 414.5, 367.8, 300.2, 236.1~
## $ bdrms    <int> 4, 3, 3, 3, 4, 5, 3, 3, 3, 3, 4, 5, 3, 3, 3, 4, 4, 3, 3, 4, 3~
## $ lotsize  <dbl> 6126, 9903, 5200, 4600, 6095, 8566, 9000, 6210, 6000, 2892, 6~
## $ sqrft    <int> 2438, 2076, 1374, 1448, 2514, 2754, 2067, 1731, 1767, 1890, 2~
## $ colonial <int> 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1~
## $ lprice   <dbl> 5.703783, 5.913503, 5.252274, 5.273000, 5.921578, 6.144775, 5~
## $ lassess  <dbl> 5.855359, 5.862210, 5.383118, 5.445875, 5.765504, 6.027073, 5~
## $ llotsize <dbl> 8.720297, 9.200593, 8.556414, 8.433811, 8.715224, 9.055556, 9~
## $ lsqrft   <dbl> 7.798934, 7.638198, 7.225482, 7.277938, 7.829630, 7.920810, 7~
```

```
mod1 <- lm(price ~ lotsize + sqrft, data = hprice1)
summary(mod1)
```

```
##
## Call:
## lm(formula = price ~ lotsize + sqrft, data = hprice1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -109.995  -36.210   -5.553   27.848  207.081
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.932e+00  2.351e+01   0.252  0.80141
## lotsize     2.113e-03  6.466e-04   3.269  0.00156 **
## sqrft       1.334e-01  1.140e-02  11.702  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.31 on 85 degrees of freedom
## Multiple R-squared:  0.6631, Adjusted R-squared:  0.6552
## F-statistic: 83.67 on 2 and 85 DF,  p-value: < 2.2e-16
```

```
round(mod1$coefficients,2)
```

```
## (Intercept)     lotsize       sqrft
##        5.93        0.00        0.13
```

```
head(hprice1)
```

```
##     price assess bdrms lotsize sqrft colonial   lprice  lassess llotsize
## 1 300.000  349.1     4    6126  2438        1 5.703783 5.855359 8.720297
## 2 370.000  351.5     3    9903  2076        1 5.913503 5.862210 9.200593
## 3 191.000  217.7     3    5200  1374        0 5.252274 5.383118 8.556414
## 4 195.000  231.8     3    4600  1448        1 5.273000 5.445875 8.433811
## 5 373.000  319.1     4    6095  2514        1 5.921578 5.765504 8.715224
## 6 466.275  414.5     5    8566  2754        1 6.144775 6.027073 9.055556
##     lsqrft
## 1 7.798934
## 2 7.638198
## 3 7.225482
## 4 7.277938
## 5 7.829630
## 6 7.920810
```

```
B <- as.matrix(hprice1[,c(4,5)])
B <- cbind(1,B)
class(B)
```

```
## [1] "matrix" "array"
```

```
#gram matrix
G <- t(B)%*%B

xb <- inv(G)%*%t(B)%*%hprice1$price
xb
```

```
##                [,1]
##         5.932414240
## lotsize 0.002113495
## sqrft   0.133362017
```

```
#last line
#residual is always orthogonal to features you have in your dataset.
round(B[,c(2)]%*%mod1$residuals,3)
```

```
##      [,1]
## [1,]    0
```

```
#error and residual
```

Codomain

- `column space + left nullspace`
- column space (range) is $\mathbb{R}^3$

use orth()

```
C_A = orth(A)
C_A
```

```
##             [,1]        [,2]
## [1,]  0.03354216  0.99686846
## [2,] -0.36102371 -0.05472854
## [3,] -0.93195322  0.05707950
```

Left Nullspace($\mathbb{A}^T$)

```
N_AT <- null(t(A))
N_AT
```

```
##             [,1]
## [1,] -0.07161149
## [2,] -0.93094934
## [3,]  0.35805744
```

Basis for output space

```
OUT <- cbind(C_A, N_AT)
rref(OUT)
```

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

Row space basis

Using transformation

using orth()

```
C_AT <- orth(t(A))
C_AT
```

```
##            [,1]        [,2]
## [1,] -0.1940942  0.29847614
## [2,]  0.3881884 -0.59695228
## [3,] -0.4519731  0.08359359
## [4,] -0.7098521 -0.13128896
## [5,]  0.3216637  0.72824124
```

Nullspace basis

- Suppose $T(\vec{x}) = A\vec{x}$, then the `kernel` or null space of such T can be found as below.

using nullspace()

```
N_A <- nullspace(A)
N_A
```

```
##              [,1]       [,2]        [,3]
## [1,]  0.04416189  0.3476413 -0.86627634
## [2,]  0.41326313  0.5658536 -0.04450838
## [3,]  0.85611211 -0.2202874  0.08531064
## [4,] -0.25090190  0.5572485  0.32464681
## [5,]  0.17715416  0.4471048  0.36730213
```

Basis spanning the input space

```
IN <- cbind(C_AT, N_A)
IN
```

```
##             [,1]        [,2]        [,3]       [,4]        [,5]
## [1,] -0.1940942  0.29847614  0.04416189  0.3476413 -0.86627634
## [2,]  0.3881884 -0.59695228  0.41326313  0.5658536 -0.04450838
## [3,] -0.4519731  0.08359359  0.85611211 -0.2202874  0.08531064
## [4,] -0.7098521 -0.13128896 -0.25090190  0.5572485  0.32464681
## [5,]  0.3216637  0.72824124  0.17715416  0.4471048  0.36730213
```

- Suppose we have $\vec{H} = [a-3b, b-a, a, b]^T$, this can be written as linear combination of two vectors $a\vec{v}_1$ and $b\vec{v}_2$ where $\vec{v}_1 = [1, -1, 1, 0]$ and $\vec{v}_2 = [-3, 1, 0, 1]$.

- This is very useful technique of expressing a subspace of $\vec{H}$ as the linear combination of some small collectoin of vectors.

- Subspace of $\vec{H} \in \text{Span}\{\vec{v}_1, \vec{v}_2\}$

How to find the basis of null space

- Step 1: Given $\mathbb{A}$, find its `rref`
- Step 2: Solve for $\vec{x}$ in $\mathbb{A}\vec{x} = \vec{0}$
- Step 3: express $\vec{x}$ as linear combination of smaller vectors.
- Step 4: identify basis spanning the null space

```
r1 <- c(-3,6,-1,1,-7)
r2 <- c(1,-2,2,3,-1)
r3 <- c(2,-4,5,8,-4)

A <- rbind(r1,r2,r3)

rref(A)
```

```
##    [,1] [,2] [,3] [,4] [,5]
## r1    1   -2    0   -1    3
## r2    0    0    1    2   -2
## r3    0    0    0    0    0
```

```
n1 <- c(2,1,0,0,0)
n2 <- c(1,0,-2,1,0)
n3 <- c(-3,0,2,0,1)

#########################################
# Any vector in the null space with A
#########################################

print(A%*%n1)
```

```
##    [,1]
## r1    0
## r2    0
## r3    0
```

```
print(A%*%(n1+n2+n3))
```

```
##    [,1]
## r1    0
## r2    0
## r3    0
```

```
print(round(A%*%(100*n1+0.1*n2-305*n3),3))
```

```
##    [,1]
## r1    0
## r2    0
## r3    0
```

Group exercise or Homework

- Get 5 matrices from class
- Given a matrix and a vector,

(1) find out where the vector lives

- Space and subspace

(2) basis of the subspace
(3) Provide a vector that is not in the span of these two subspaces

Concept check questions

- What is the relationship between $C(A)$ and $N(A^T)$?
- What is the relationship between $R(A)$ and $N(A)$?
- What is the relationship between $C(A)$ and $R(A)$?
- What is the relationship between $N(A)$ and $N(A^T)$?
- If the basis spanning $C(A)$ are given, can you find out the basis spanning $N(A^T)$?
- If the basis spanning $R(A)$ are given, can you find out the basis spanning $N(A^T)$?

Projection matrix

Review

- Given: Suppose $A \in R^{nxn}$ and A^{-1} exist, then the following can be said
  - The columns of A is the basis of R^n
  - rank $A =$ n
  - $NulA = \{\vec{0}\}$
  - dim $NulA = 0$
  - $A^{-1}A = I$
  - $AA^{-1} = I$
  - The Linear transformation $\vec{x} \mapsto A\vec{x}$ is one-to-one
  - A^T is an invertible matrix

Space, subspace, orthogonal complement subspace

- Let S be space of R^n, A is R^{mxn} matrix.
- Let $C(A)$ and $N(A^T)$ be the column space and left nullspace of A
- $C(A)$ and $N(A^T)$ are orthogonal complement subspace of each other.
- Then, any vector, $\vec{x} \in S$ but $\vec{x} \notin C(A)$ or $\vec{x} \notin N(A^T)$ can be expressed by the linear combination of basis of $C(A)$ and $N(A^T)$

Change of basis

Given: $\vec{y} \notin C(A)$, and Rank of $A = 2$, and $\vec{y} \in R^3$

Problem 1

- Let $\hat{\vec{y}} \subset C(A)$ where $\vec{C}_1$ and $\vec{C}_2$ are the basis of $C(A)$
- Find $\hat{\vec{y}}$ that minimizes $||\vec{y} - \hat{\vec{y}}||$

Solution:

- let C and N be the matrix that contains the basis of $C(A)$ and $N(A^T)$
- Since: $C\vec{x} = \hat{\vec{y}}$ and $C\vec{x} + N\vec{z} = \vec{y}$
- Simplify the expression

$$\begin{aligned} C^TC\vec{x} &= C^T\vec{y} \\ \vec{x} &= (C^TC)^{-1}C^T\vec{y} \end{aligned}$$

- Then,

$$C(C^TC)^{-1}C^T\vec{y} = \hat{\vec{y}}$$

- $C(C^TC)^{-1}C^T$ is called **projection matrix***

$\mathbb{A}$

$\hat{\vec{b}}$

$$\mathbb{A}\dot{}\vec{x} = \vec{b}$$
$$\mathbb{B}\dot{}\vec{x}_B + \mathbb{D}\dot{}\vec{x}_D = \vec{b}$$

Projection matrix

$$\mathbb{I} = \mathbb{P} + \mathbb{B}$$
$$\vec{y} = \mathbb{P}\vec{y} + \mathbb{B}\vec{y}$$

- where P and B are the `projection matrices` for $C(A)$ and $N(A^T)$

Example

```
A <- matrix(c(1,1,2,-1,3,3,1,2,4), nrow = 3, byrow=TRUE)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]   -1    3    3
## [3,]    1    2    4
```

```
Rank(A)
```

```
## [1] 3
```

```
dim(A)
```

```
## [1] 3 3
```

```
b <- c(1,4,-4)


x <- inv(A)%*%b

A%*%x
```

```
##      [,1]
## [1,]    1
## [2,]    4
## [3,]   -4
```

DOT Product

$$\hat{\vec{y}} = P_{\vec{u}}^{\vec{y}} = \frac{\vec{y}\boldsymbol{.}\vec{u}}{\vec{u}\boldsymbol{.}\vec{u}}\vec{u}$$

where

$\vec{y}\boldsymbol{.}\vec{u}$ and $\vec{u}\boldsymbol{.}\vec{u}$ are scalar quantity.

Projection tells you the `length` of the `projected vector`, $\hat{\vec{y}}$ in terms of the vector that is `being projected onto` $\vec{u}$

```
# y will be projected onto u
y <- matrix(c(7,6),nrow=2)
u <- matrix(c(4,2),nrow=2)
u0 <- matrix(c(16,8),nrow=2)
```

Using projection matrix

```
## using projection matrix
P <- u%*%(solve(t(u)%*%u)%*%t(u))
print(P)
```

```
##      [,1] [,2]
## [1,]  0.8  0.4
## [2,]  0.4  0.2
```

```
print(P%*%y)
```

```
##      [,1]
## [1,]    8
## [2,]    4
```

Using Projection formula on to $\vec{u}$

```
print(drop((t(y)%*%u)/(t(u)%*%u))*u)
```

```
##      [,1]
## [1,]    8
## [2,]    4
```

```
print(drop((t(y)%*%u)/(t(u)%*%u)))
```

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

Using Projection formula on to $\vec{u}_0$

```
print(drop((t(y)%*%u0)/(t(u0)%*%u0))*u0)
```

```
##      [,1]
## [1,]    8
## [2,]    4
```

```
print(drop((t(y)%*%u0)/(t(u0)%*%u0)))
```

```
## [1] 0.5
```

Orthogonal

- Two vectors $\vec{v_1}$ and $\vec{v_2} \in R^m$ are orthogonal, if $\vec{v_1} \cdot \vec{v_2} = 0$
- Note that the dot product produce scalar quantity 0 not $\vec{0}$
- Notice $\vec{v}_1$ is size of 3 vector and `orth( )` returns normalized $\vec{v}_1$

```
v1 <- c(3,4,5)
```

Orthonormal basis

```
mybasis <- matrix(c(1,2,3,4,5,6),nrow=3)
print(mybasis)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
print(orthonormalization(mybasis))
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.2672612  0.8728716  0.4082483
## [2,] 0.5345225  0.2182179 -0.8164966
## [3,] 0.8017837 -0.4364358  0.4082483
```

```
Z <- (orthonormalization(mybasis))
# z is orthonormal basis of codomain (I called it output space)

A <- matrix(c(4,3,5,6,8,10,5,12,13),nrow=3, byrow=T)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    4    6    5
## [2,]    3    8   12
## [3,]    5   10   13
```

```
c(Norm(A[1,]),Norm(A[2,]),Norm(A[3,])) #norm of each row vectors in A (i.e., sample)
```

```
## [1]  8.774964 14.730920 17.146428
```

```
B <- A%*%Z
print(B)
```

```
##           [,1]       [,2]       [,3]
## [1,]  8.285098  2.6186147 -1.2247449
## [2,] 14.699368 -0.8728716 -0.4082483
## [3,] 17.104719  0.8728716 -0.8164966
```

```
print(Z)
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.2672612  0.8728716  0.4082483
## [2,] 0.5345225  0.2182179 -0.8164966
## [3,] 0.8017837 -0.4364358  0.4082483
```

Explain the following

```
print(Z%*%B[1,])
```

```
##      [,1]
## [1,]    4
## [2,]    6
## [3,]    5
```

```
print(Z%*%B[2,])
```

```
##      [,1]
## [1,]    3
## [2,]    8
## [3,]   12
```

```
print(Z%*%B[3,])
```

```
##      [,1]
## [1,]    5
## [2,]   10
## [3,]   13
```

Normalizing the basis

```
c_A <- orth(v1)
print(c_A)
```

```
##           [,1]
## [1,] 0.4242641
## [2,] 0.5656854
## [3,] 0.7071068
```

```
#notice what happens when you dot v1 and c_A
print(v1%*%c_A)
```

```
##          [,1]
## [1,] 7.071068
```

```
Norm(v1)
```

```
## [1] 7.071068
```

Diagonal matrix

```
D1 <- diag(c(5,2,10),3,3)
print(D1)
```

```
##      [,1] [,2] [,3]
## [1,]    5    0    0
## [2,]    0    2    0
## [3,]    0    0   10
```

```
print(inv(D1)) #notice when the diagonal elements has zero in it, D1 becomes singular.
```

```
##      [,1] [,2] [,3]
## [1,]  0.2  0.0  0.0
## [2,]  0.0  0.5  0.0
## [3,]  0.0  0.0  0.1
```

```
print(D1 %^% 3) # using the function in expm
```

```
##      [,1] [,2] [,3]
## [1,]  125    0    0
## [2,]    0    8    0
## [3,]    0    0 1000
```

Orthogonal matrix

$$U^{-1} = U^T$$

- Let W be a subspace of R^n and let $\vec{y} \in R^n$ but $\vec{y} \notin W$.
- Then, $\hat{\vec{y}} \in W$ that is the closest approximation of $\vec{y}$ is the $\vec{y}$ projected onto W

Proerty of matrx that is not square, but has orthonormal basis

```
v <- matrix(c(2,1,2),nrow=3)
O <- orthonormalization(v)
print(O)
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.6666667 -0.2357023 -0.7071068
## [2,] 0.3333333  0.9428090  0.0000000
## [3,] 0.6666667 -0.2357023  0.7071068
```

```r
U <- cbind(O[,1],O[,2])
print(t(U)%*%U)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Suppose C is matrix that contains orthonormal basis of W. Since there exist $\vec{y} \notin W$, C can't be square matrix.

However, the basis in C can still be `orthonormal`.

Let C be retangular matrix with orthonormal basis,

$$\vec{y} = C\vec{x}_w + N\vec{x}_N$$

where - N is the basis spanning orthogonal complement subspace of W. Then,

$$C^T\vec{y} = C^TC\vec{x}_w$$

Since C is matrix that contains orthonormal basis, C^TC becomes identify matrix.

$$C^T\vec{y} = \vec{x}_W$$

Now, the location of $\hat{\vec{y}}$ in terms of the basis in C can be expressed as below

$$C\vec{x}_W = \hat{\vec{y}}$$

Solving for $\vec{x}_W$

$$\vec{x}_W = C^T\hat{\vec{y}}$$

Sub the above expression of $\vec{x}_W$ to the following equation

$$C^T\vec{y} = C^TC\vec{x}_w$$

$$C^T\vec{y} = C^TC(C^T\hat{\vec{y}})$$

Then,

$$CC^T\vec{y} = \hat{\vec{y}}$$

Gram-Schmidt Process

- Let $\{\vec{x}_1, \vec{x}_2 ... \vec{x}_p\}$ be basis for a nonzero subspace W of R^n where $p < n$. Gram-Schimidt process converts $\{\vec{x}_1, \vec{x}_2 ... \vec{x}_p\}$ to $\{\vec{v}_1, \vec{v}_2 ... \vec{v}_p\}$ where $\{\vec{v}_1, \vec{v}_2 ... \vec{v}_p\}$ are orthogonal basis for W
- Gram-Schimit process is projecting one set of basis to another basis that is orthogonal to them.
- Notice the `orthonormalization( )` in `R` returns 3 x 3 matrix. This function in R returns the basis spanning the subspace that is orthogonal to subspace spanned by $\vec{v}_1$

```r
GS <- orthonormalization(v1)
print(GS)
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.4242641  0.9055385  0.0000000
## [2,] 0.5656854 -0.2650357  0.7808688
## [3,] 0.7071068 -0.3312946 -0.6246950
```

Gram-Schmidt

Group exercise or Homework

Gram-Schmidt Process

```
set.seed(100)

A <- matrix(rnorm(10), ncol = 2)
A
```

```
##              [,1]       [,2]
## [1,] -0.50219235  0.3186301
## [2,]  0.13153117 -0.5817907
## [3,] -0.07891709  0.7145327
## [4,]  0.88678481 -0.8252594
## [5,]  0.11697127 -0.3598621
```

- Perform gram schmit process and explain the result

Find the basis spanning the four subspaces using R built-in function

```
r1 <- c(1,3,4,5,6)
r2 <- c(1,5,4,5,3)
r3 <- c(1,-2,4,7,6)

A <- cbind(r1,r2,r3)
print(A)
```

```
##      r1 r2 r3
## [1,]  1  1  1
## [2,]  3  5 -2
## [3,]  4  4  4
## [4,]  5  5  7
## [5,]  6  3  6
```

Find the basis spanning the nullspace without using R built-in function

- You can use `rref()`

Projection and MLE

```
#6.2 Example 1
u1 <- c(3,1,1)
u2 <- c(-1,2,1)
u3 <- c(-0.5, -2, 7/2)

print(t(u1)%*%u2)
```

```
##      [,1]
## [1,]    0
```

```
print(t(u3)%*%u2)
```

```
##      [,1]
## [1,]    0
```

```
print(t(u1)%*%u3)
```

```
##      [,1]
## [1,]    0
```

```
#page 399, example 2
y <- c(6,1,-8)

A <- cbind(u1,u2,u3)
print(A)
```

```
##      u1 u2   u3
## [1,]  3 -1 -0.5
## [2,]  1  2 -2.0
## [3,]  1  1  3.5
```

- is $\vec{y}$ in $C(\mathbb{A})$?

```
Rank(A)
```

```
## [1] 3
```

Since $\mathbb{A}$ is full rank, we can get $\curvearrowleft$ the following way

```
x <- inv(A)%*%y
print(A%*%x)
```

```
##      [,1]
## [1,]    6
## [2,]    1
## [3,]   -8
```

```
print("++++++++++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++++++++++"
```

```
############################################
# PROJECTION
############################################

# since each column vector of A are orthogonal we can use projection
# as well
x1 <- y%*%u1/(Norm(u1)^2)
x2 <- y%*%u2/(Norm(u2)^2)
x3 <- y%*%u3/(Norm(u3)^2)

# then using these coordinate you can get the following result as well
x <- c(x1, x2, x3)
print(A%*%x)
```

```
##      [,1]
## [1,]    6
## [2,]    1
## [3,]   -8
```

Orthogonal projection

- Very important concept and may take a few days of practice.
- see page 340.
- Suppose you have $\vec{u}$ and denote its subspace by L, and you have $\vec{y}$ that is not in the span of $\vec{u}$

projecting $\vec{y}$ onto L

$$\text{proj}_L\vec{y} = \hat{y} = \frac{\vec{y}\vec{u}}{\vec{u}\vec{u}}\vec{u}$$

```
# Example 3 (see slide 8)
y <- c(7,6)
u <- c(4,2)

hat_y <- y%*%u/(Norm(u)^2)*u
residual <- y - hat_y

print(y)
```

```
## [1] 7 6
```

```
print(hat_y + residual)
```

```
## [1] 7 6
```

```
#######################################################
# what is the relationship between hat_y and residuel?
#######################################################

print(round(hat_y %*% residual,3))
```

```
##      [,1]
## [1,]    0
```

```
# Example 3 (see slide 8)
# the same problem, but solved without using Norm()
y <- c(7,6)
u <- c(4,2)

#++++++++++++++++++++++++++++++++++++++++++++++
#what would be the physical meaning of this?
#see I wonder by Sam. =)
#++++++++++++++++++++++++++++++++++++++++++++++
print((y%*%u)/(u%*%u))
```

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

```
hat_y <- (y%*%u)/(u%*%u)*u

print(hat_y)
```

```
## [1] 8 4
```

```
residual <- y - hat_y

#############################################
# Will this always be zero?
# Why or why not?
#############################################
print(hat_y%*%residual)
```

```
##      [,1]
## [1,]    0
```

```
# example 6, see slide 13
# Orthonomal colums
#######################################################
#  Special property of matrix with orthonormal columns
#######################################################

u1 <- c(1/sqrt(2), 1/sqrt(2), 0)
u2 <- c(2/3, -2/3, 1/3)
U <- cbind(u1, u2)
x <- c(sqrt(2), 3)

##########################
print("Printing the norm of the vectors")
```

```
## [1] "Printing the norm of the vectors"
```

```
print(Norm(u1))
```

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

```
print(Norm(u2))
```

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

```
print("++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++"
```

```
print(round(t(U)%*%U,2))
```

```
##    u1 u2
## u1  1  0
## u2  0  1
```

```
print("=======================")
```

```
## [1] "======================="
```

```
##################################################
RHS <- U%*%x
print(U%*%x)
```

```
##      [,1]
## [1,]    3
## [2,]   -1
## [3,]    1
```

```
print("++++++++++++++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++++++++++++++"
```

```
print(Norm(U%*%x))
```

```
## [1] 3.316625
```

```
print(Norm(x))
```

```
## [1] 3.316625
```

```
print("++++++++++++++++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++++++++++++++++"
```

```
##################################################
print(t(U)%*%RHS)
```

```
##        [,1]
## u1 1.414214
## u2 3.000000
```

```
print(x)
```

```
## [1] 1.414214 3.000000
```

```
y <- matrix(c(1,2,5,7), nrow = 4)

A <- matrix(c(1,-1,3,2,1,4,4,1), nrow = 4, byrow = TRUE)
print(A)
```

```
##      [,1] [,2]
## [1,]    1   -1
## [2,]    3    2
## [3,]    1    4
## [4,]    4    1
```

Discussion

- $\mathbb{U}$ transformed a vector in R^2 to R^3.
- The size of vector changed, but the norm of the vector did not change.
- Recall that $\mathbb{A}^{-1}\vec{b}$ only works when $\mathbb{A}$ is singular.
- But notice, when $\mathbb{U}$ has orthonormal columns, we can use $\mathbb{U}^T$ to transform the `RHS` be to row space!

```
#page 351, example 3
u1 <- c(2,5,-1)
u2 <- c(-2,1,1)
y <- c(1,2,3)

#since the norm is not 1
#you still need to normalize it
print(Norm(u1))
```

```
## [1] 5.477226
```

```
U <- cbind(u1,u2)

#using Gram matrix
print("using gram matrix")
```

```
## [1] "using gram matrix"
```

```
x_hat<- inv(t(U)%*%U)%*%t(U)%*%y
print(U%*%x_hat)
```

```
##      [,1]
## [1,] -0.4
## [2,]  2.0
## [3,]  0.2
```

```
# using projection
print("using projection")
```

```
## [1] "using projection"
```

```
x1 <- y%*%u1/(Norm(u1)^2)
x2 <- y%*%u2/(Norm(u2)^2)
x <- rbind(x1,x2)
print(U%*%x)
```

```
##      [,1]
## [1,] -0.4
## [2,]  2.0
## [3,]  0.2
```

Orthogonal complement subspace

$$R(\mathbb{A})^{\perp} = N(\mathbb{A})$$

$$C(\mathbb{A})^{\perp} = N(\mathbb{A}^T)$$

- Orthogonal basis for subspace $\mathbb{W}$ is a basis for $\mathbb{W}$ that is also an orthogonal set
- $\mathbb{U} \in R^{mbyn}$ has orthogonal columns if and only if $\mathbb{U}^T\mathbb{U} = \mathbb{I}$

Angle between vectors

- This concept can be extended to beyond R^3

$$\vec{u}\vec{v} = ||\vec{u}||||\vec{v}||cos(\theta)$$

Difference between projecting onto orthogonal basis vs basis

- Explain what will be difference

More on matrix with orthonormal columns

- Orthonormal columns

$$
\begin{aligned}
\mathbb{U}\vec{x} &= ||\vec{x}|| \\
(\mathbb{U}\vec{x})(\mathbb{U}\vec{y}) &= \vec{x}\vec{y} \\
(\mathbb{U}\vec{x})(\mathbb{U}\vec{y}) = 0 &\text{ if and only if } \vec{x}\vec{y} = 0
\end{aligned}
$$

Orthogonal decomposition

- Projecting $\vec{y}$ on the the orthogonal basis or orthogonal complement subsapce (i.e., this is linear regression)
- Given baiss, you can create orthonormal basis that spans the same space. `The Gram-schmidt process`

```r
# see the example 1 from Chapter 6
# page 362

r1 <- c(4,0)
r2 <- c(0,2)
r3 <- c(1,1)

#your feature
A <- rbind(r1,r2,r3)

#your response
b <- c(2,0,11)
```

$$
\begin{aligned}
\mathbb{A}\vec{x} &= \vec{b} \\
\mathbb{A}^T\mathbb{A}\vec{x} &= \mathbb{A}^T\vec{b} \\
\mathbb{G}\vec{x} &= \mathbb{A}^T\vec{b} \\
\mathbb{G}^{-1}\mathbb{G}\vec{x} &= \mathbb{G}^{-1}\mathbb{A}^T\vec{b} \\
\vec{x} &= \mathbb{G}^{-1}\mathbb{A}^T\vec{b}
\end{aligned}
$$

- Above set of equations require set of assumptions. can you identify them?

```r
# see the example 1 from Chapter 6
# page 362 continue

# this tells you the linear combination of
# column vectors of A that will get you y_hat

x <- inv(t(A)%*%A)%*%t(A)%*%b

#########################################
#what is the physical meaning of this x?
# x is the least sqaure solution
#########################################
print(x)
```

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

```
print("++++++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++++++"
```

```
print(b)
```

```
## [1]  2  0 11
```

```
print("++++++++++++++++++++++++++++++")
```

```
## [1] "++++++++++++++++++++++++++++++"
```

```
###########################################
# predict the value
############################################
y_hat <-A%*%x
print(y_hat)
```

```
##    [,1]
## r1    4
## r2    4
## r3    3
```

```
print("------------------------------")
```

```
## [1] "------------------------------"
```

```
residual <- b - y_hat
print(residual)
```

```
##    [,1]
## r1   -2
## r2   -4
## r3    8
```

```
#########################################
# Why is this value zero? can anyone explain?
#########################################
print(round(t(y_hat)%*%residual,4))
```

```
##      [,1]
## [1,]    0
```

```
# see the example 2 from Chapter 6
# page 363 continue

v1 <- c(1,1,1,1,1,1)
v2 <- c(1,1,0,0,0,0)
v3 <- c(0,0,1,1,0,0)
v4 <- c(0,0,0,0,1,1)
b  <- c(-3,-1,0,2,5,1)

A <- cbind(v1,v2,v3,v4)

# Will the gram matrix invertible?
print(Rank(A))
```

```
## [1] 3
```

```
print("===================")
```

```
## [1] "==================="
```

```
# is b in C(A)
Ab <- cbind(A,b)
print(Rank(Ab))
```

```
## [1] 4
```

Group exercise or Homework

1. Will the gram matrix be invertible?

2.

Is $\vec{b}$ in C($\mathbb{A}$)?

3.

How can we get $\hat{\vec{x}}$?

$$\mathbb{A}\vec{x} = \vec{b}$$
$$\mathbb{A}^T\mathbb{A}\vec{x} = \mathbb{A}^T\vec{b}$$

4.

- Note that this process is different from when you have ∞ number of solution

```
###########################################
# Create gram matrix, but this is singular
###########################################
G <- t(A)%*%A

# creating gram matrix requir multiplying RHS
# by AT
RHS <- t(A)%*%b

hyperplane <- cbind(G,RHS)
rref(hyperplane)
```

```
##    v1 v2 v3 v4
## v1  1  0  0  1  3
## v2  0  1  0 -1 -5
## v3  0  0  1 -1 -2
## v4  0  0  0  0  0
```

$$x_1 = 3 - x_4$$
$$x_2 = -5 + x_4$$
$$x_3 = -2 + x_4$$
$$x_4 = \text{free}$$

```
shift <- c(3,-5,-2,0)
basis <- c(-1,1,1,1)

#One solution
print(A%*%shift)
```

```
##      [,1]
## [1,]   -2
## [2,]   -2
## [3,]    1
## [4,]    1
## [5,]    3
## [6,]    3
```

```
#Another solution
print("===========================")
```

```
## [1] "==========================="
```

```
x_another <- shift + 0.01*basis
print(A%*%x_another)
```

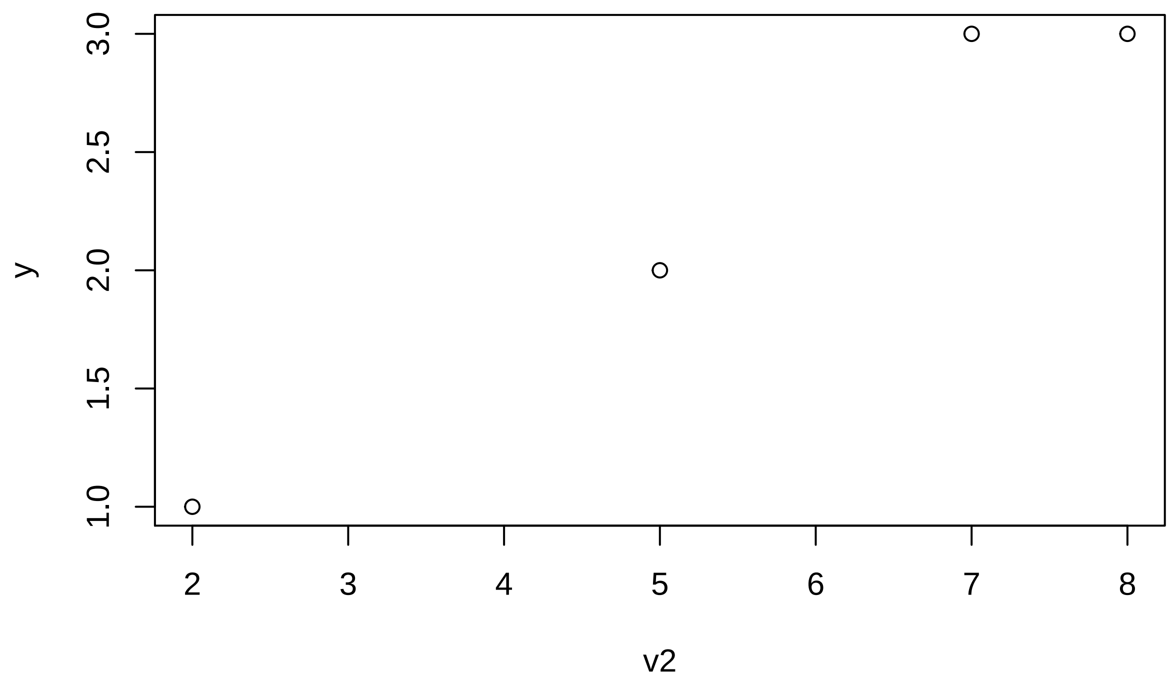
```
##      [,1]
## [1,]   -2
## [2,]   -2
## [3,]    1
## [4,]    1
## [5,]    3
## [6,]    3
```

```
# page 370 example 1

v1 <- c(1,1,1,1)
v2 <- c(2,5,7,8)
y <- c(1,2,3,3)

A<- cbind(v1,v2)

plot(v2,y)
```



```
# fractions() is function in MASS
fractions(inv(t(A)%*%A)%*%t(A)%*%y)
```

```
##    [,1]
## v1  2/7
## v2 5/14
```

MLE

- See page 317, is the following model linear?

$$y = \beta_0 + \beta_1 + \beta_2 x^2$$

- In the equation above β_0 is the constant term, what would be the effect of leaving this constant out of the model? Explain the effect using the following terms: `hyperplane` and `subspace`

Covarianc matrix

```
x1 <- matrix(c(1,2,1,4,2),nrow=5)
x2 <- matrix(c(4,2,13,2,1),nrow=5)
x3 <- matrix(c(7,8,1,3,4),nrow=5)
x4 <- matrix(c(8,4,5,5,6),nrow=5)

X <- cbind(x1,x2,x3,x4)

print(X)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7    8
## [2,]    2    2    8    4
## [3,]    1   13    1    5
## [4,]    4    2    3    5
## [5,]    2    1    4    6
```

using built in function

```
cov(X)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  1.50 -3.25 -0.50 -0.75
## [2,] -3.25 24.30 -8.55 -0.55
## [3,] -0.50 -8.55  8.30  0.80
## [4,] -0.75 -0.55  0.80  2.30
```

step by step

```
rSum <- colSums(X)/5
mean <- matrix(rep(rSum,5),nrow=5, byrow=TRUE)
M <- X - mean
S <- t(M)%*%M/4
S
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  1.50 -3.25 -0.50 -0.75
## [2,] -3.25 24.30 -8.55 -0.55
## [3,] -0.50 -8.55  8.30  0.80
## [4,] -0.75 -0.55  0.80  2.30
```

Additional problem that estimates sample covariance LINK

- Think about how $X_i - \bar{x}$ will change if X_i are all centered. Then, $\bar{x}$ will be zero.
- Can you express the numerator using vector multiplication?

Bayes Formula

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A^C)P(A^C)}$$

```r
c.table <- array(data = c(5,15,10,20,35,15), dim=c(3,2),
                 dimnames = list(Group=c("A","B","C"),
                                 TestResult = c("Positive", "Negative")))
c.table
```

```
##      TestResult
## Group Positive Negative
##     A        5       20
##     B       15       35
##     C       10       15
```

```r
#condition on X
c_X <- c.table/rowSums(c.table)
c_X
```

```
##      TestResult
## Group Positive Negative
##     A      0.2      0.8
##     B      0.3      0.7
##     C      0.4      0.6
```

```r
#joint
c_j <- c.table/sum(c.table)
c_j
```

```
##      TestResult
## Group Positive Negative
##     A     0.05     0.20
##     B     0.15     0.35
##     C     0.10     0.15
```

Given that `Test Result` is Positive, what will be `P(A|P)`, `P(B|P)`, `P(C|P)`?

Related to CLT

```r
a <- rnorm(25)
A <- matrix(rnorm(25), nrow=5)
A
```

```
##            [,1]       [,2]        [,3]       [,4]       [,5]
## [1,] -0.2217942 -0.1016292  1.32223096 -0.4470622  0.9804641
## [2,]  0.1829077  1.4032035 -0.36344033 -1.7385979 -1.3988256
## [3,]  0.4173233 -1.7767756  1.31906574  0.1788648  1.8248724
## [4,]  1.0654023  0.6228674  0.04377907  1.8974657  1.3812987
## [5,]  0.9702020 -0.5222834 -1.87865588 -2.2719255 -0.8388519
```

```
diag(A)
```

```
## [1] -0.2217942  1.4032035  1.3190657  1.8974657 -0.8388519
```