Homework 4 Arrays and Lists

James Young

6/20/2019

# Instructions

There are six exercises below. You are required to provide solutions for at least four of the five. You are required to solve at least one exercise in R, and at least one in SAS. You are required to provide five solutions, each solution will be worth 10 points. Thus, you may choose to provide both R and SAS solutions for a single exercise, or you may solve all five problems, mixing the languages as you wish. The first four exercise refer to formula from the previous homework, you may reuse code as you wish.

#### Experimental

Again, you will be allowed to provide one solution using Python. Elaborate on the similarities and differences between Python arrays vs R or IML.

# Exercise 1.Please Grade

### Part a.

We will calculate a number of required replicates for a range of mean differences, comparable to calories per serving estimates found in Wansink, Table 1.

Let be a sequence of means from 320-420, incremented by 10. Let be 270. Assume a pooled standard deviation of 150.

Calculate Cohen’s for the pairs of means , letting . Calculate the required replicates for these same pairs of means. You may reuse code or functions from previous homework at your discretion.

To show your results, either create and print a matrix with one colum for effect size and one column for replicates, or plot required replicates versus effect size (effect size will be the independent variable). What does this tell you about the number of observations required to detect medium-size effects? You may include reference lines in your plot to illustrate.

m1 = seq(320,420,by=10)  
  
# function definition  
  
cohen.d <- function(m1, s1, m2, s2) {  
 sdpooled <- ((((s1\*\*2 + s2\*\*2))/2)^(1/2))  
 (abs(m1 - m2))/sdpooled  
   
}  
  
# function definition  
  
required.replicates <- function (m1, s1, m2, s2) {  
 CV <- (((s1\*\*2+s2\*\*2)/2)^(1/2))/((m1+m2)/2)  
 Diff <- (m1-m2)/((m1+m2)/2)  
 sdpooled <- ((((s1\*\*2 + s2\*\*2))/2)^(1/2))  
 2 \* ((CV/Diff)\*\*2) \* (((1-(0.05/2))+(1-0.2)))\*\*2  
}

cohen <- cohen.d(m1, 150, 270, 150)  
replicates <- required.replicates(m1, 150, 270, 150)  
  
matrix(   
 c(cohen, replicates),   
 nrow=11,   
 ncol=2)

## [,1] [,2]  
## [1,] 0.333333333333 56.71125000000  
## [2,] 0.400000000000 39.38281250000  
## [3,] 0.466666666667 28.93431122449  
## [4,] 0.533333333333 22.15283203125  
## [5,] 0.600000000000 17.50347222222  
## [6,] 0.666666666667 14.17781250000  
## [7,] 0.733333333333 11.71720041322  
## [8,] 0.800000000000 9.84570312500  
## [9,] 0.866666666667 8.38923816568  
## [10,] 0.933333333333 7.23357780612  
## [11,] 1.000000000000 6.30125000000

# Exercise 2 Please Grade

Create a table to show the required replicates for a range of combinations of and . Do this in steps as follows:

### Part a.

Define two matrices, one for CV and one for Diff. Each will matrix will be 5 rows by 6 columns. Let the rows in CV be the sequence and let the columns of Diff be the squence . The matrices should look like:

Define and print your matrices in the code below.

CV <- seq(8,28, by=4)  
Diff <- seq(5,25,by=5)  
  
matrix(   
 c(CV, CV, CV, CV, CV, CV),   
 nrow=6,   
 ncol=6)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 8 8 8 8 8 8  
## [2,] 12 12 12 12 12 12  
## [3,] 16 16 16 16 16 16  
## [4,] 20 20 20 20 20 20  
## [5,] 24 24 24 24 24 24  
## [6,] 28 28 28 28 28 28

matrix(   
 c(Diff, Diff, Diff, Diff, Diff, Diff),   
 nrow=5,   
 ncol=6)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 5 5 5 5 5 5  
## [2,] 10 10 10 10 10 10  
## [3,] 15 15 15 15 15 15  
## [4,] 20 20 20 20 20 20  
## [5,] 25 25 25 25 25 25

### Part b.

Calculate require replicates for each combination of CV and Diff. Use the same values for and as from Homework 2 and 3. You should be able to reuse coce from previous exercises, and you should not use iteration.

Print the result below. The result should be a matrix.

required.replicates <- function (CV, Diff) {  
  
 2 \* ((CV/Diff)\*\*2) \* ((qnorm(1-(0.05/2))+qnorm(1-0.2)))\*\*2  
}  
  
CV = seq(8, 28, by=4)  
Diff = seq(5, 25, by=4)  
  
CV1 = required.replicates(CV, 5)  
CV2 = required.replicates(CV, 10)  
CV3 = required.replicates(CV, 15)  
CV4 = required.replicates(CV, 20)  
CV5 = required.replicates(CV, 25)  
  
  
matrix(   
 c(CV1, CV2, CV3, CV4, CV5),   
 nrow=6,   
 ncol=5)

## [,1] [,2] [,3] [,4]  
## [1,] 40.1862642399 10.0465660600 4.4651404711 2.51164151499  
## [2,] 90.4190945397 22.6047736349 10.0465660600 5.65119340873  
## [3,] 160.7450569595 40.1862642399 17.8605618844 10.04656605997  
## [4,] 251.1641514992 62.7910378748 27.9071279444 15.69775946870  
## [5,] 361.6763781588 90.4190945397 40.1862642399 22.60477363493  
## [6,] 492.2817369384 123.0704342346 54.6979707709 30.76760855865  
## [,5]  
## [1,] 1.60745056959  
## [2,] 3.61676378159  
## [3,] 6.42980227838  
## [4,] 10.04656605997  
## [5,] 14.46705512635  
## [6,] 19.69126947753

To check your work, repeat the calculations using the rule of thumb from the previous exercises. What is largest deviation of the rule of thumb from the exact calculation?

# standard deviation of CV is about 7.4  
rule <- function(m1, m2){  
 16/((abs(m1-m2))/7.4)  
}  
  
rule(5,8)

## [1] 39.4666666667

# Exercise 3 Please Grade

In this exercise, we’ll use your norm.pdf function to illustrate how the formula for required replicates finds a compromise between Type I and Type II error rates. This is also a way to test your normal probability function over a range of arguments.

**Do not print the vectors you create for this exercise in the final typeset submission** We will check the results by examining the plots, and printing the vectors themselves will unnecessarily clutter your report. If you get stuck, use the built normal functions to create your plots.

### Part a.

Generate a squence of values from incremented by ; let this be x. Calculate the probability of each value of x using the norm.pdf function from Homework 3, letting mu=0 and sd=1. Name the result p.null.

Calculate the effect size for 1936 versus 2006, calories per recipe, as in Homework 2 and 3. Repeat the calculation for the probability of x, but this time use mean= effect size. Name this result p.alt.

The results will be the distribution of the expected value of the difference between means; the first is the expectation under the null hypothesis (the true effect size is 0) while the second is the expectation assuming the measured is the true .

seq = seq(-3,4,by=0.1)  
  
# function definition  
norm.pdf <- function(x) {  
 (1/((2\*pi)^(1/2))) \* exp(-((x-0)^2)/((2\*1)^2))  
   
}  
  
p.null = norm.pdf(seq)  
  
#Mean effect size 1936 vs 2016  
x =cohen.d(268.1, 124.8, 384.4, 168.3)  
x

## [1] 0.784987603959

norm.pdf <- function(x) {  
 (1/((2\*pi)^(1/2))) \* exp(-((x-0.7849876)^2)/((2\*1)^2))}  
  
p.alt = norm.pdf(seq)

### Part b.

Repeat the calculations of p.null and p.alt, but this time let sigma = where . Name these p.null.10 and p.alt.10. These calculations narrow the distribbutions by an amount proportional to standard error.

sig10 = sqrt(2/10)  
sig10

## [1] 0.4472135955

seq = seq(-3,4,by=0.1)  
  
  
# function definition  
sig10 = sqrt(2/10)  
norm.pdf <- function(x) {  
 (1/((0.4472136\*2\*pi)^(1/2))) \* exp(-((x-0)^2)/((2\*0.4472136)^2))  
   
}  
  
 p.null.10 = norm.pdf(seq)

norm.pdf <- function(x) {  
 (1/((0.4472136\*2\*pi)^(1/2))) \* exp(-((x-0.7849876)^2)/((2\*0.4472136)^2))}  
  
p.alt.10 = norm.pdf(seq)

### Part c.

Repeat the calculations of p.null and p.alt, but this time let sigma = where is the minimun mumber of replicates for calories per recipe, 1936 versus 2006, as calculated previously. Call these p.null.req and p.alt.req.

#previous replicates was 10.xx so I will round up to 11 to have adequate replicates  
sigrep = sqrt(2/11)  
sigrep

## [1] 0.426401432711

# function definition  
  
norm.pdf <- function(x) {  
 (1/((0.4264014\*2\*pi)^(1/2))) \* exp(-((x-0)^2)/((2\*0.4264014)^2))  
   
}  
  
p.null.req = norm.pdf(seq)  
  
norm.pdf <- function(x) {  
 (1/((0.4264014\*2\*pi)^(1/2))) \* exp(-((x-0.7849876)^2)/((2\*0.4260414)^2))}  
  
p.alt.req = norm.pdf(seq)

### Part d.

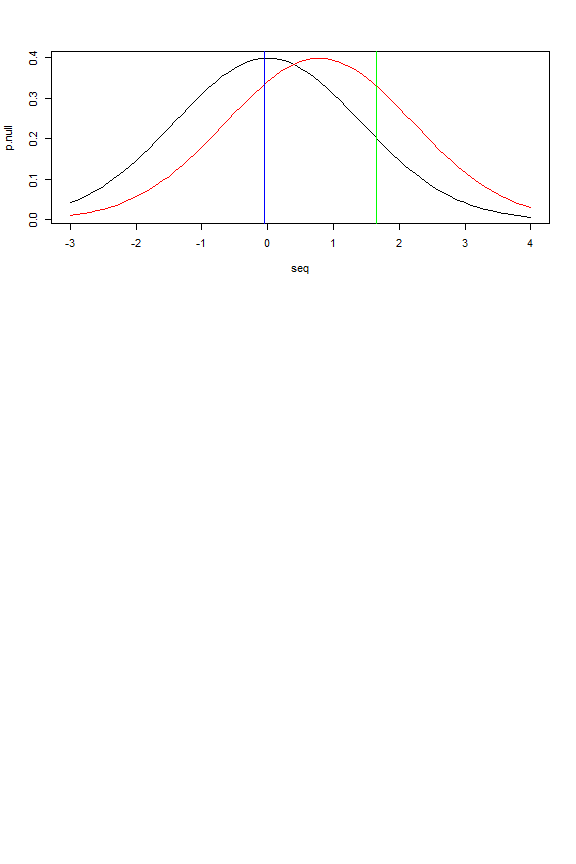
Plot p.null versus x as a black line and in the same plot add p.alt vs x as a red line. Add a green vertical line at and a blue vertical line at , using values as in previous exercises. The green line represents the critical value for Type I error, and the error under the black curve to the left of the green line is the probability of that error (97.5%). The area under the red curve, to the left of the green line, represents the achieved Type II error rate, the blue line represents the desired Type II rate.

Repeat the plot with p.null.10 and p.alt.10, but this time add vertical lines at and at . The lines representing critical values for Type I and Type II error should move closer as the distributions narrow.

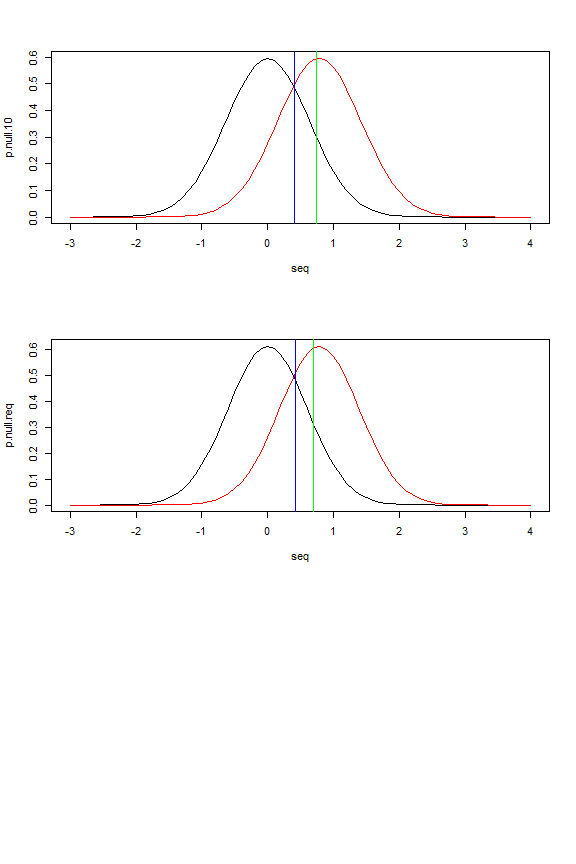
Repeat the plot with p.null.req and p.alt.req, but this time add vertical lines at and at , where is the minimum replicates. Do the lines for Type I and Type II error overlap?

It will improve the readability of the three plots if you plot all three in the chunk below. The arguments inside the braces specify the dimensions of the plot, while par(mfrow=c(3,1)) combines three plots into one graph.

#p  
par(mfrow=c(3,1))  
plot(p.null~seq, type ="l", col = "black")  
points(p.alt~ seq, type = "l", col = "red")  
abline(v = qnorm(1-0.05), col = "green")  
abline(v = 0.785-qnorm(1-0.2), col = "blue")  
#p.10  
par(mfrow=c(3,1))



plot(p.null.10~ seq, type ="l", col = "black")  
points(p.alt.10~ seq, type = "l", col = "red")  
abline(v = qnorm(1-0.05)\*sqrt(2/10), col = "green")  
abline(v = 0.785-qnorm(1-0.2)\*sqrt(2/10), col = "blue")  
#p.req  
plot(p.null.req~ seq, type ="l", col = "black")  
points(p.alt.req~ seq, type = "l", col = "red")  
abline(v = qnorm(1-0.05)\*sqrt(2/11), col = "green")  
abline(v = 0.785-qnorm(1-0.2)\*sqrt(2/11), col = "blue")



If you choose to solve this with SAS, I’ve included code in the SAS template to create the graphs, since combining plots in IML is not as easy as in R.

# Exercise 4 Please Grade

In this, we compare the normal and Poisson distributions, using the functions you’ve written previously. This is also a way to test your normal and Poisson functions over a range of arguments.

**Do not print the vectors you create for this exercise in the final typeset submission** We will check the results by examining the plots, and printing the vectors themselves will unnecessarily clutter your report. If you get stuck, use the built functions to create your plots. However, the final submission must call your functions.

### Part a

Create a sequence of from , incremented by 0.1. Calculate the normal likelihood for each , assuming and . Also calculate Poisson probability of each given a lambda=1.

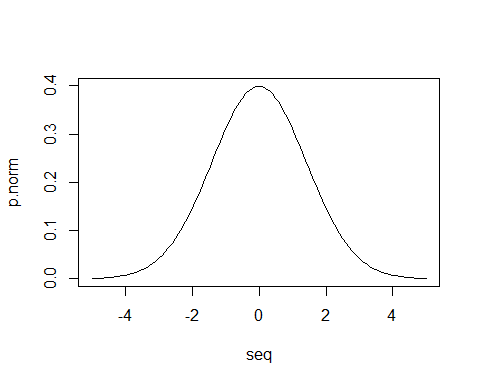
Plot both sets of probablities against x as lines, using a different color for each curve. Make sure that both curves fit in the plot; you may need to determine minimum and maximum values and set these as graphic parameters (see ylim).

Warning: if you do this in SAS, you may have to adjust the lower bound of .

seq =seq(-5,5,by=0.1)  
  
#e^(-1) ~ 0.36787944117   
pois.pmf <- function(x){  
 ( 0.36787944117\*(-1)\*\*(x))/(factorial(x))  
}  
  
norm.pdf <- function(x) {  
 (1/((2\*pi)^(1/2))) \* exp(-((x-0)^2)/((2\*1)^2))  
   
}  
  
p.norm = norm.pdf(seq)  
p.pois = pois.pmf(seq)

## Warning in gamma(x + 1): NaNs produced

plot(p.norm~seq, type ="l", col = "black")  
points(p.pois~ seq, type = "l", col = "red")



###This tells me poisson does not handle negative integers but the normal pdf still behaves properly

Does this graph tell you if your Normal PDF function behaves properly? Does your Poisson handle negative or non-integer values as expected?

### Part b

Create a sequence of using mean and standard deviation for servings per recipe from 1936.

Calculate the normal and Poission probability for each as in part a, again using mean and standard deviation from servings per recipe, 1936. The length of this vector should be the same length as the vector as in part a (), so you will need to calculate an interval based on the range x\_b and the number of elements in x\_a

Show the the length of both vectors are similar by calling length for each.

Repeat the plot from part a with this sequence.

If you choose to solve this with SAS, I’ve included code in the SAS template to create the graphs, since combining plots in IML is not as easy as in R.

range = (12.9-5\*13.3)- (12.9+5\*13.3)  
range

## [1] -133

seq1 = seq(12.9-5\*13.3, 12.9+5\*13.3, by=1.33)  
length(seq)

## [1] 101

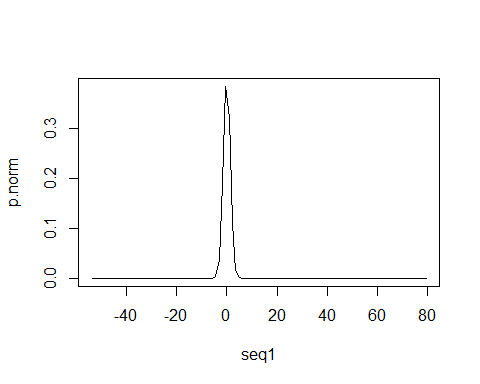
length(seq1)

## [1] 101

p.norm = norm.pdf(seq1)  
p.pois = pois.pmf(seq1)

## Warning in gamma(x + 1): NaNs produced

plot(p.norm~seq1, type ="l", col = "black")  
points(p.pois~ seq1, type = "l", col = "red")

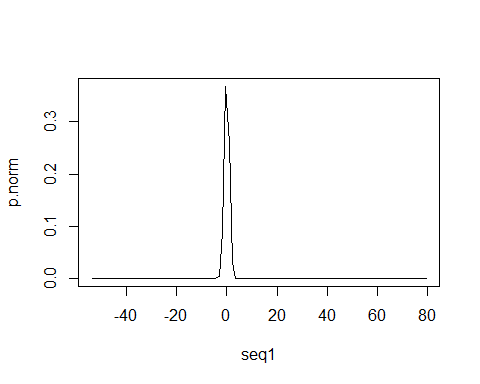


To check you work, duplicate the plots by calling built in normal and Poisson functions. Does the system Poisson function handle negative differently than your function?

p.norm = dnorm(seq1)  
p.pois = dpois(seq1, -1)

## Warning in dpois(seq1, -1): NaNs produced

plot(p.norm~seq1, type ="l", col = "black")  
points(p.pois~ seq1, type = "l", col = "red")



###As far as I can tell the system poisson does not handle negative integers differently.

# Exercise 5

Consider the table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Rate | 23000 | 24000 | 25000 | 26000 | 27000 | 28000 | 29000 |
| Yield | 111.4216 | 155.0326 | 181.1176 | 227.5800 | 233.4623 | 242.1753 | 231.3890 |

Suppose we wish to determine the linear relationship between per Rate and Yield. We can determine this by solving a system of linear equations, of the form

We write this in matrix notation as

We might write this as

and find a solution by computing .

However, an exact solution for the inverse, require square matrices, so commonly we use the *normal* equations,

(where is the transpose of ). We then find

### Answer

Define appropriate X and y matrices (y can be a vector in R) in the chunk below.

Multiply the transpose of X by X, then use solve (R) or inv (IML) to find the inverse. Multiply this by the product of transpose X and y to find hat.beta.

Print your hat.beta.

To check your work, calculate the values predicted by your statistical model. Compute hat.y by multiplying X and hat.beta,

Plot y vs the independent variable (the second column of X) as points, and hat.y vs independent variable as a line, preferably a different colors. The hat.y values should fall a straight line that interpolates y values.

You can also compare your result to the R function (set eval=TRUE).

summary(lm(y~X))

#### Alternative methods

You can also compute by passing both and as arguments to solve.

Alternatively, you can install the MASS library and use ginv to compute a generalized inverse . Use this to compute in the chunk below:

library(MASS)

# Exercise 6

Given a vector of mean estimates , a vector of standard deviations and a vector of sample sizes , we can calculate a one-way analysis of variance by

and

where is the mean of and . The test statistic is which is distributed as

### Part a

Calculate MSW and MSB for Calories per Serving from Wansink Table 1. You can use the variables CaloriesPerServingMean and CaloriesPerServingSD defined below. Let

Use array functions and arithmetic for your calculations, you should not need iteration (for loops). Do not hard code values for and , calculate these from the CaloriesPerServingMean or CaloriesPerServingSD.

Print both MSB and MSW.

CaloriesPerServingMean <- c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4)  
CaloriesPerServingSD <- c(124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3)

### Part b

Calculate an F-ratio and a for this , using the distribution with and degrees of freedom. Use . Compare these values to the corresponding values reported in Wansink Table 1.

You can check your results by entering appropriate values in an online calculator like <http://statpages.info/anova1sm.html> .