LAB 5 MATH 463

1. Simulation

A *simulation* of a distribution is a method to generate a random variable X which has this distribution.

For example, to simulate the distribution which assigns probability $1/6$ to each element of $\{1, 2, 3, 4, 5, 6\}$, roll a balanced 6-sided die.

Since it is impractical to roll dice, it is easier for the computer to act as if it is rolling dice:

```
> sample(c(1,2,3,4,5,6),size=1)
[1] 5
```

generates a random sample of size 1 from $\{1, 2, 3, 4, 5, 6\}$, i.e. simulates the distribution which assigns probability $1/6$ to each outcome.

We can generate 100 instances of a standard normal via

```
> x = rnorm(n=100,mean=0,sd=1)
```

If you simulate many instances of a distribution, say $X_1, X_2, \ldots, X_{100}$, and then look at the histogram of the results, the histogram should look like the probability density function:

```
> #the flag prob=T makes the histogram have total area 1
> hist(x,prob=T)
> #compare with standard normal density
> a = seq(from=-3,to=3,by=0.1)
> b = dnorm(a,mean=0,sd=1)
> lines(a,b)
```

Note that as n increases, the histogram becomes closer to the density which you are simulating: Simulate 10000 standard normals, make the histogram, and compare with the density function.

Thus, if you don't know the distribution of a random variable, but are able to simulate it, you can approximate numerically it's distribution by the histogram of the simulations.

Verify that the sample mean $\bar{X}$ from a random sample of size 10 has a Normal distribution with standard deviation $1/\sqrt{10}$ by simulation:

To get 800 sample means (simulations of $\bar{X}$) write a R script (program)

```
> #create a vector to store the 800 sample means.
> sampmeans = 1:800
> #make a loop to simulate 800 times using 'for'
> for(i in 1:800){
+   #generate 10 normals
+   randomnorms = rnorm(n=10,mean=0,sd=1)
+   #set i-th component of sampmeans equal to the mean
+   #of these 10 random normals
```

```
+   sampmeans[i]=mean(randomnorms)
+ }
```

Make a histogram of sampmeans and compare to the normal distribution with mean 0 and standard deviation $1/\sqrt{10}$.

2. Simulating a linear model

Suppose that

$$g(t) = \begin{cases} 0 & \text{if } t \leq 30 \\ \beta_1(t-30) & \text{if } 30 < t \leq 60, \\ \beta_1 30 + \beta_2(t-60) & \text{if } t > 60\,. \end{cases}$$

A model for random variables $(Y_1, \ldots, Y_n)$ given values $(a_1, a_2, \ldots, a_n)$ is that

$$Y_i = g(a_i) + \epsilon_i\,, \tag{1}$$

where ε_i are i.i.d. Normal$(0, \sigma^2)$.

In the first part of this lab,

- *we were given* values $Y_1, \ldots, Y_{100}$, values $a_1, \ldots, a_n$, and
- *assumed* that the data came from this model,
- *were not told* the values β_1, β_2 or σ^2
- and asked to *estimate* β_1 and β_2 and *test the hypothesis* that $\beta_1 = \beta_2$.

How good is the method we used to do this test? Can we detect small differences between β_1 and β_2 if they exist, using the given amount of data?

One way to answer this question is to use simulation. We can simulate data using this model for β_1 and β_2 close to each other, and see if we can detect the difference.

First, we need some values for the a_i's. The range of values for the data we actually analyzed last week was $[0, 120]$. So take 100 data points uniformly in this interval:

```
> sizesim = 100
> a = runif(n=sizesim,min=0,max=120)
```

We need to create the function g for given values of β_1, β_2:

```
> g = function(a,b1,b2){
+   if(a<=30){v = 0}
+   if((30<a)&&(a<=60)){v = b1*(a-30)}
+   if(a>60){v = 30*b1+b2*(a-60)}
+   v
+ }
```

Let us simulate the data when $\beta_1 = 1$ and $\beta_2 = 1.1$ and $\sigma^2 = 30$.

```
> simsigma = 30
> # create the function g1(a)=g(a,1,1.1)
> g1 = function(a){g(a,1,1.1)}
> # apply to a
> ga = apply(matrix(a),1,FUN=g1)
> # simulate normals
> ep = rnorm(n=sizesim,mean=0,sd=simsigma)
```

```
> # create y
> y = ga + ep
```

Plot y vs. a and see if your eye can detect the change in slope. Add the line $y = g(a)$ to the plot:

```
> plot(y~a)
> da = seq(0,120,0.2)
> dy = apply(matrix(da),1,FUN=g1)
> lines(da,dy)
```

Now test the hypothesis that $\beta_1 = \beta_2$ on the simulated data: This is what we did last week

```
> testdata = data.frame(y,a)
> xf1=function(a){
+   if((a>30)&&(a<=60)){
+     v = 1}
+   else{
+     v = 0}
+   v
+   }
> testdata$x1 = apply(as.matrix(testdata$a),1,xf1)
> xf2=function(a){
+   if(a>60){
+     v = 1}
+   else{
+     v = 0}
+   v
+   }
> testdata$x2 = apply(as.matrix(testdata$a),1,xf2)
> testdata$z1=(testdata$a-30)*testdata$x1 + 30*testdata$x2
> testdata$z2=(testdata$a-60)*testdata$x2
> testlm = lm(y~z1+z2-1,data=testdata)
> fitcoef = summary(testlm)$coef
> fitcov  = summary(testlm)$cov.unscaled
> sighat  = summary(testlm)$sigma
> SEsq = (fitcov[1,1]+fitcov[2,2]-2*fitcov[1,2])*sighat^2
> SE = sqrt(SEsq)
> Tstat = (fitcoef[2]-fitcoef[1])/SE
```

The T-statistic equals 0.85, and thus the p-value equals 0.4.

What effect does making σ smaller have? Now redo this with $\sigma = 3$. Can the difference in slopes be detected?

What about using more data? Redo this (again with $\sigma = 30$) but now with $n = 1000$? Can the difference in slopes be detected? Find an n for which you can detect the difference.