Basic Stats Tutorial

Team Stats

Monday, February 09, 2015

### 1. Generating sample datasets

For each common distribution, R provides a function to generate a random dataset. These function names all begin with 'r' , followed by the distribution's shortcut name.

The first parameter, n, is always the size of the requested sample. After this, each distribution type has its own specific parameters. Type ?*funcname* to view the function's help file, which will give details on each parameter.

Available functions include: runif, rbinom, rnorm, rexp, rf, rgamma, rgeom, rlnorm, rpois, etc.

sampleUnif <- runif(100,min=1,max=100) #A vector of 100 variables between 1 and 10.  
sampleNorm <- rnorm(100, mean=50, sd=3) #A vector of 100 variables with a mean of 50 and sd of 3.  
sampleExp <- rexp(100, rate=0.05) #mean = 1/rate   
  
#preview our data  
head(sampleUnif)

## [1] 81.83172 31.45500 33.83788 52.66453 95.18099 55.17258

head(sampleNorm)

## [1] 49.08902 51.63849 45.34908 48.74987 47.77956 46.00821

head(sampleExp)

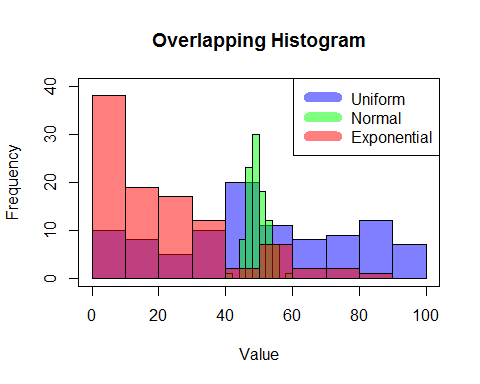
## [1] 23.57702 52.32127 14.45028 53.33372 7.50905 28.83343

We can provide some simple visualization for this. There are many options for plotting data. You will have to check out the help files to get details, or do a google search for anything you are trying to achieve.

R comes with standard functions for plotting data in formats such as lines, curves, and histograms. Separate functions and parameters will allow you to update colors, add legends, and change display style. Especially using the *par* function, you can make changes to the defaults.

There is also a common library, ggplot2, which you can install to perform more elegant plotting. We won't go into that right now.

hist(sampleUnif, col=rgb(0,0,1,0.5),xlim=c(0,100), ylim=c(0,40), main="Overlapping Histogram", xlab="Value")  
hist(sampleNorm, col=rgb(0,1,0,0.5), add=T)  
hist(sampleExp, col=rgb(1,0,0,0.5), add=T)  
  
legend("topright", c("Uniform","Normal","Exponential"),col=c(rgb(0,0,1,0.5),rgb(0,1,0,0.5),rgb(1,0,0,0.5)), lwd=10)  
box() #decorative box around data



### 2. Sampling from a larger population

You can also grab a random sample out of a larger dataset you have imported. *sample* takes a sample of the specified size from the elements of x using either with or without replacement.

You can also include a vector of weighted probabilities. These need not sum to one, but must all be non-negative with at least one non-zero.

sample(20) #randomly select 20 objects from 1:20 (in essence 'shuffle' them)

## [1] 10 11 19 13 8 14 1 3 5 9 2 20 15 6 7 18 12 17 4 16

sample(20,5) #randomly select 5 numbers from 1:20, without replacement.

## [1] 7 5 16 1 6

sample(c(10:20),5,replace=TRUE) #Choose 5 from 10:20, WITH replacement.

## [1] 16 15 16 16 12

sample(c(1:10),5,replace=FALSE,prob=c(1,2,1,2,1,2,1,2,1,2)) #Select 5 from 1:10, where the probability of choosing an even number is twice as high.

## [1] 8 1 6 4 2

Working with samples from data frames is slightly more complicated, but still easy to do.

data(cars) #load an included tutorial dataset (stored in 'cars' variable by default)  
str(cars) #provides structural information on the data. 50 obs. of 2 variables

## 'data.frame': 50 obs. of 2 variables:  
## $ speed: num 4 4 7 7 8 9 10 10 10 11 ...  
## $ dist : num 2 10 4 22 16 10 18 26 34 17 ...

head(cars) #preview the data

## speed dist  
## 1 4 2  
## 2 4 10  
## 3 7 4  
## 4 7 22  
## 5 8 16  
## 6 9 10

cars[sample(nrow(cars),2),] #Get a random size 2 subset of cars data frame

## speed dist  
## 49 24 120  
## 25 15 26

### 3. Simple descriptive statistics functions

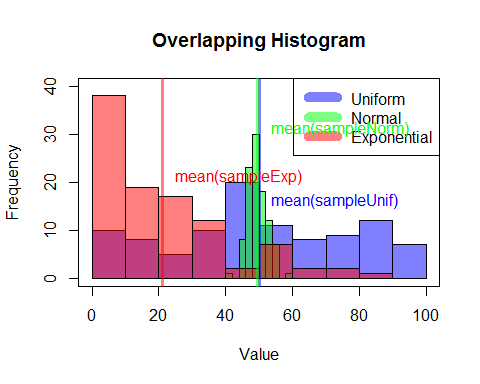
We can easily calculate min, max, mean, variance, covariance, and standard deviation.

First, let's calculate the means of our sample distributions and add them to the histogram plot.

meanUnif <- mean(sampleUnif)  
meanNorm <- mean(sampleNorm)   
meanExp <- mean(sampleExp)   
cat("meanUnif:",meanUnif,"\nmeanNorm:",meanNorm,"\nmeanExp:",meanExp,"\n")

## meanUnif: 49.96642   
## meanNorm: 49.51833   
## meanExp: 21.11965

abline(v=meanUnif, col=rgb(0,0,1,0.5), lwd=3)  
abline(v=meanNorm, col=rgb(0,1,0,0.5), lwd=3)  
abline(v=meanExp, col=rgb(1,0,0,0.5), lwd=3)  
  
text(meanUnif,15, "mean(sampleUnif)", adj = c(-.1, -.1))  
text(meanNorm,30, "mean(sampleNorm)", adj = c(-.1, -.1))  
text(meanExp,20,"mean(sampleExp)", adj = c(-.1, -.1))



Now let's work instead with the provided tutorial dataset and calculate other statistics.

head(cars) #Our dataset from earlier. 50 observations of 2 variables.

## speed dist  
## 1 4 2  
## 2 4 10  
## 3 7 4  
## 4 7 22  
## 5 8 16  
## 6 9 10

min(cars$dist)

## [1] 2

max(cars$dist)

## [1] 120

mean(cars$dist)

## [1] 42.98

var(cars$dist)

## [1] 664.0608

cov(cars$dist,cars$speed) #defaults to pearson method, but can also set kendall or spearman.

## [1] 109.9469

cov(cars)#computes covariance of all variables in a provided dataframe or matrix.

## speed dist  
## speed 27.95918 109.9469  
## dist 109.94694 664.0608

cor.test(cars$dist,cars$speed) #can also test for correlation between paired samples.

##   
## Pearson's product-moment correlation  
##   
## data: cars$dist and cars$speed  
## t = 9.464, df = 48, p-value = 1.49e-12  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.6816422 0.8862036  
## sample estimates:  
## cor   
## 0.8068949

sd(cars$dist)

## [1] 25.76938

### 4. Discrete probability functions

The same distributions used to generate random data sets can be used to compute density, distribution, and quantile functions. Use the same function name shortcuts from above, but instead of 'r' append them with 'd', 'p', and 'q' respectively.

For discrete probability functions, we could look at:

* Binomial
* Hypergeometric
* Geometric
* Poisson

#Please replace these with meaningful values! Then, set eval=TRUE to run it.  
dbinom(x=cars$dist,size=20,prob=0.5)  
dhyper(x,m,n,k)  
dgeom(p,prob)  
dpois(p,lambda)

### 5. Continuous probability functions

Continuous probability functions such as:

* Normal
* Student's t
* Exponential
* Gamma
* Chi-squared

#Please replace these with meaningful values! Then, set eval=TRUE to run it.  
dnorm(p, mean, sd)  
dt(p, df)  
dexp(p, rate)  
dgamma(p, shape, rate=rate) ; dgamma(p,shape,scale=scale)  
dchisq(p,df)

### 5. Pre-class survey questions

If I multiply a dataset by 10, what kind of change would happen to its variance?

v1 <- var(sampleUnif); v1

## [1] 738.4241

v2 <- var(sampleUnif\*10); v1

## [1] 738.4241

cat("The variance is",v2/v1,"times as large.")

## The variance is 100 times as large.

### 6. Learn more! Fun tutorial games!!!

#install.packages(swirl)  
library(swirl)  
swirl()  
#Now follow the directions! Woo!