

An Introductory Tutorial:
Learning R for Quantitative Thinking in the Life Sciences

Scott C Merrill

October 10rd, 2012

Chapter 7 Distributions!

R has a function that allows one to generate samples based on distributions such as the normal distribution. Compile the following to see some of the available distributions:

```
require(animation) # you will probably have to install this package

# sampling from a poisson distribution with different lambdas
# lambda = mean = standard deviation in a poisson distribution

# looping through lambdas from 1 to 20
#   with a 2 second pause between histograms
for (lam in 1:20) {
  true.pop = rpois(n=10000,lambda = lam)
  hist(true.pop)
  ani.pause(2) # creates a pause between histogram displays (animation package)
}

# Generating samples from a Beta distribution
# looping through a beta distribution with different shape parameters
#   with shape parameters 1 & 2 ranging from 1:10
for (shp1 in 1:50) {
  for (shp2 in 1:50) {
    true.pop = rbeta(n=10000,shape1 = shp1/5, shape2 = shp2/5)
    hist(true.pop, main = c("shape1 = ",shp1/5," shape2 = ",shp2/5))
    ani.pause(.1)
  }
}

# Generating samples from a Weibull distribution
# looping through a weibull distribution with different shape parameters
#   with shape and scale parameters ranging from 1:10
```

```

for (shp1 in 1:50) {
  for (scale1 in 1:50) {
    Weibull = rweibull(n=10000,shape = shp1/5, scale = scale1/5)
    hist(Weibull, main = c("shape = ",shp1/5," scale =",scale1/5))
    ani.pause(.1)
  }
}

# Now try a gamma distribution
for (shp1 in 1:50) {
  for (scale1 in 1:50) {
    Gamma.Distribution = rgamma(n=10000,shape = shp1/5, scale = scale1/5)
    hist(Gamma.Distribution, main = c("shape = ",shp1/5," scale =",scale1/5))
    ani.pause(.05)
  }
}

```

Simulation to obtaining probability density and quantile functions

We have been using the `r` –prefix which samples from the distribution that you have described following the `r`. For example `rnorm` generates samples from the normal distribution with a mean and standard deviation defined by the user.

The “`d`” prefix:

Alternatively, one could find the probability density function. For example `dnorm`, where instead of inputting the number of samples to obtain, one would enter values, with the probability density value returned. Try compiling:

```
dnorm(x = 3,mean = 5, sd = 1)
```

Hint: the `x` value is 2 standard deviations away from the mean. What would you expect the probability density function to equal?

The “`p`” prefix:

One could find the cumulative density function using the “`p`” prefix. For example `pnorm`, where instead of inputting the number of samples to obtain, one would enter values of interest, with the cumulative density value returned. Try compiling:

```
pnorm(q = 7, mean = 5, sd = 1)
```

Hint: the `q` value is 2 standard deviations above the mean. What would you expect the cumulative density function to equal?

The “q” prefix:

One could find the value of the quantile using the “p” prefix. For example `qnorm`, where instead of inputting the number of samples to obtain, one would enter cumulative probability of interest and have the corresponding `x` value returned. Try compiling:

```
qnorm(p= .1, mean = 5, sd = 1) # returns value of x with 10% of probabilities at or below
```

or

```
qnorm(p = .9, mean = 5, sd = 1) # returns value of x with 90% of probabilities at or below
```

Do you understand what is happening here?

For those of you who are feeling ambitious, you can glance through Vito Ricci’s distribution manuscript. It is fairly dense and has a lot of equations/jargon but has some really good information on there:

<http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf>

Exercise:

Simulate 10,50,200,1000,100000 data points in a sequence for a normal distribution with means looping from -10 to 10, and standard deviation looping from 1 to 5 using the distribution code:

```
# code for generating 500 samples from a normal distribution with a mean of 100  
# and a standard deviation of 10 looks like  
# rnorm(n = 500,mean=100,sd=10)
```

```
#Create a histogram for each distribution (you do not have to output each of these plots, just  
# send me the code and the last plot (i.e., n=100000, mean = 10, sd = 5)
```

```
# put these code sections into a three piece nested loop with loops for sample number, mean  
and standard deviation.
```

```
sample.number = c(10,50,200,1000,100000)
```

```
simulation.data = rnorm(n = x[loop.variable1], mean = [loop.variable2], sd = [loop.variable3])
```

```
hist(simulation.data)
```

Hint: you can create a for loop that runs from -10 to 10 using the same code structure that we have been using:

```
for (x in -10:10) {  
  do stuff in here  
}
```

Exercise:

Simulate data and histograms using the following distributions:

log normal distribution e.g.,

```
rlnorm(meanlog=.8,sdlog = .5, n = 1000)
```

exponential

```
rexp(n = 10000, rate = 2)
```

What is happening here?

hint: $f(x) = \lambda * \exp^{-\lambda x}$

Also, (from Ally's cheat sheet) you could check out these distributions:

```
rt(n, df) # 'Student' (t)
```

```
rf(n, df1, df2) # Fisher-Snedecor (F) (c2)
```

```
rchisq(n, df) # Pearson
```

```
rbinom(n, size, prob) # binomial
```

```
rgeom(n, prob) # geometric
```

```
rhyper(nn, m, n, k) # hypergeometric
```

```
rlogis(n, location=0, scale=1) # logistic
```

```
rlnorm(n, meanlog=0, sdlog=1) # lognormal
```

```
rnbinom(n, size, prob) # negative binomial
```

```
runif(n, min=0, max=1) # uniform
```

```
rwilcox(nn, m, n)
```