

SAMPLING, DISTRIBUTIONS, SIMULATIONS AND BOOTSTRAPPING IN R.

General advice: Try to avoid loops, use matrices!!!

SAMPLING

```
#sampling without replacement
sample(1:10, 10)

#sampling with replacement
sample(1:10, 10, replace=T)

library(MASS)
#A numeric vector of velocities in km/sec of 82 galaxies
data(galaxies)
gal<-galaxies/1000
hist(sample(gal, 100, replace=T))
```

Distributions in R

One convenient use of R is to provide a comprehensive set of statistical tables. Functions are provided to evaluate the cumulative distribution function $P(X \leq x)$, the probability density function and the quantile function (given q , the smallest x such that $P(X \leq x) > q$), and to simulate from the distribution.

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max

Weibull weibull shape, scale
Wilcoxon wilcox m, n

Prefix the name given here by *d* for the density, *p* for the CDF, *q* for the quantile function and *r* for simulation (*random deviates*). The first argument is *x* for *dxxx*, *q* for *pxxx*, *p* for *qxxx* and *n* for *rx* (except for *rhyper* and *rwilcox*, for which it is *nn*). The non-centrality parameter *ncp* is currently only available for the CDFs and a few other functions: see the on-line help for current details.

SIMULATIONS

There are about 20 distributions available including the normal, the binomial, the exponential, the logistic, Poisson, etc. Each of these can be accessed for density ('d'), cumulative density ('p'), quantile ('q') or simulation ('r' for random deviates). Thus, to find out the probability of a normal score of value *x* from a distribution with mean=*m* and sd = *s*,
`pnorm(x,mean=m, sd=s)`, eg.

```
pnorm(1,mean=0,sd=1)
[1] 0.8413447
same as
pnorm(1)            #default values of mean=0, sd=1 are used)
pnorm(1,1,10)      #parameters may be passed if in default order or by name
```

Simulating a simple correlation between two variables based upon their correlation with a latent variable:

```
samplesize<-1000
size.r<-0.6
theta<-rnorm(samplesize,0,1)            #generate some random normal deviates
e1<-rnorm(samplesize,0,1)            #generate errors for x
e2<-rnorm(samplesize,0,1)            #generate errors for y
weight<-sqrt(size.r)            #weight as a function of correlation
x<-weight*theta+e1*sqrt(1-size.r) #combine true score (theta) with
error
y<-weight*theta+e2*sqrt(1-size.r)
cor(x,y)            #correlate the resulting pair
df<-data.frame(cbind(theta,e1,e2,x,y)) #form a data frame to hold all
of the elements
round(cor(df),2)            #show the correlational structure
pairs(df)            #plot the correlational structure
```

Simulating multivariate structures using R

The following example shows how to simulate a multivariate structure with a particular measurement model and a particular structural model. This example produces data suitable for demonstrations of regression, correlation, factor analysis, or structural equation modeling.

The particular example assumes that there are 3 measures of ability (GREV, GREQ, GREA), two measures of motivation (achievement motivation and anxiety), and three measures of performance (Prelims, GPA, MA). These titles are, of course, arbitrary and can be changed easily.

```
#R Control sequence to generate numberofcases of latent variables Xi
and then produce errorful (observed)
# data for numberofvariable items (with true scores reliability of
trueweight)
#done using matrices for generality

title<- "Simulation study" #<---title goes here
numberofcases<-1000
numberofvariables<-8
numberoflatent<-3

                                     #structural model
effect<-matrix(c(1,0,.7,
                 0,1,.6,
                 0,.0,.39),nrow=numberoflatent,byrow=TRUE)

                                     #measurement model
model<-matrix(c(.9,.8,.7,0,0,0,0,0,
               0,0,.6,.8,.7,0,0,0,
               0,0,0,0,0,.7,.6,.5),nrow=numberofvariables,ncol=numberoflatent,byrow=FA
LSE)

tmodel<-t(model)      #transpose of model
model%*%tmodel        #show the resulting latent structure

communality<-diag(model%*%tmodel)      #find how much to weight true
#scores and errors given the measurement model
  uniqueness<-1-communality
  errorweight<-sqrt(uniqueness)
  errorweight<-diag(errorweight)      #how much to weight the errors

truescores<-
matrix(rnorm(numberofcases*(numberoflatent)),numberofcases) #create
#true scores for the latent variables. Matrix 1000 by 3.
  round(cor(truescores),2)
  truescores<-truescores%*%effect      #create true scores to reflect
#structural relations

  observedscore<-truescores%*%tmodel
  round(cor(observedscore),2)          #show the true score correlation
matrix (without error)
  error<- matrix(rnorm(numberofcases*(numberofvariables)),numberofcases)
#create normal error scores
  error<-error%*%errorweight          #matrix 1000 by 8.
  observedscore<-observedscore+error  #matrix 1000 by 8.
  round(cor(observedscore),2)        #show the correlation matrix
                                     #give the data "realistic"

properties

GREV<-round(observedscore[,1]*100+500,0)
GREQ<-round(observedscore[,2]*100+500,0)
```

```

GREV<-round(observedscore[,3]*100+500,0)
Ach<-round(observedscore[,4]*10+50,0)
Anx<-round(-observedscore[,5]*10+50,0)
Prelim<-round(observedscore[,6]+10,0)
GPA<-round(observedscore[,7]*.5+4,2)
MA<-round(observedscore[,8]*.5+3,1)

data<-data.frame(GREV,GREQ,GREA,Ach,Anx,Prelim,GPA,MA)
summary(data) #basic summary statistics
round(cor(data),2) #show the resulting correlations
#it is, of course, identical to the
#previous one

```

Bootstrapping the Sample Median--1

```

X<-rgamma(100,2,1)
summary(X)
set.seed(101); nsamp<-1000; res<-numeric(nsamp)
for (i in 1:nsamp) res[i] <- median(sample(X, replace=T))
se.b<-sqrt(var(res))
se.b
quantile(res, p = c(0.025, 0.975))
par(mfrow=c(1,2))
hist(res)
qqnorm(res)

```

Bootstrapping the Sample Median--2

```

library(boot)
X<-rgamma(100,2,1)
summary(X)
set.seed(101)
X.boot<-boot(X, function(y, i) median(y[i]), R=1000)
#boot: Generate 'R' bootstrap replicates of a statistic applied to data.
#R: number of replicas
#The function used:
# must take at least two arguments. The first argument passed
# will always be the original data. The second will be a vector
# of indices, frequencies or weights which define the bootstrap
#sample.
X.boot
boot.ci(X.boot, conf = c(0.95, 0.99), type = c("norm", "perc"))
# Nonparametric Bootstrap Confidence Intervals
#This function generates 5 different types of equi-tailed two-sided
# nonparametric confidence intervals. These are the first order
# normal approximation, the basic bootstrap interval, the
# studentized bootstrap interval, the bootstrap percentile

```

```
# interval, and the adjusted bootstrap percentile (BCa) interval.  
par(mfrow=c(1,2))  
plot(X.boot)
```

Bootstrapping a Trimmed Mean

```
X<-rgamma(100,2,1)
```

```
tm <- mean(X, trim = 0.10)
```

```
set.seed(101)
```

```
nsamp <- 1000
```

```
res <- numeric(nsamp)
```

```
for (i in 1:nsamp) res[i] <- mean(sample(X, replace = TRUE), trim=0.10)
```

```
hist(res)
```

```
abline(v = tm, lty = 4)
```

```
sd(res)
```

```
quantile(res, p = c(0.05, 0.95))
```