Probability Distributions and CI for Correlation

Probability Distributions

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

Distribution	R name	Additional arguments
Binomial	`binom`	`size, prob`
Chi-squared	`chisq`	`df, ncp`
Exponential	`exp`	`rate`
F	`f`	`df1, df2, ncp`
Normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
Student's t	`t`	`df, ncp`
Uniform	`unif`	`min, max`

Prefix the name given here by '`d`' --- for the density;
'`p`' --- for the CDF;
'`q`' --- for the quantile function;
'`r`' --- for simulation (random numbers).

Here are some examples.

```
> ## 2-tailed p-value for t(13) distribution
> 2*pt(-2.43, df = 13)

> ## upper 1% point for an F(2, 7) distribution
> qf(0.99, df1 = 2, df2 = 7)


> # To generate 100 random numbers from N(0,2) distribution
> x <- rnorm(100, mean = 0, sd = 2); x
> # by default, rnorm generates numbers from N(0, 1).
```

`> # to compute` $P(X = 12)$`, where` $X \sim Binomial(20, .45)$

```
> pbinom(12, size = 20, prob = 0.45)
[1] 0.942
```

A Confidence Interval for Correlation Coefficient

The following function is written to construct a p% (by default, 95%) confidence interval for the population correlation coefficient between two random variables, say, *X* and *Y*.

```
ci.corr <- function(x, y, p=.95)
{  r <- cor(x, y);  # compute the sample correlation
   n <- length(x)     # sample size
   z <- 0.5*log((1+r)/(1-r)) # Fisher's Z transforamtion
   width <- qnorm(1-(1-p)/2)/sqrt(n-3) # half width of the CI
   L <- z - width   # Lower bound for z
   U <- z + width   # Upper bound for z
   cbind(lower.bound=(exp(2*L)-1)/(exp(2*L)+1),
         upper.bound=(exp(2*U)-1)/(exp(2*U)+1))
}
```

To apply the above function, consider the following artificial data.

```
> n <- 100 # sample size
> x <- rexp(n, rate = 1)
> y <- 2.5 * x^2 + rnorm(n, mean = 0, sd = 0.5)
> ci.corr(x, y, p = 0.9)
     lower.bound upper.bound
[1,]      0.9013      0.9482
```