# Introduction to R

## I. Using R for Statistical Tables and Plotting Distributions

The `R` suite of programs provides a simple way for statistical tables of just about any probability distribution of interest and also allows for easy plotting of the form of these distributions. In what follows below, R commands are set in `bold courier`. Note that **R** commands are CASE-SENSITIVE, so be careful when typing.

### General Syntax for Distribution Functions

There are four basic `R` commands that apply to the various distributions defined in `R`. Letting `DIST` denote the particular distribution and `parameters` the parameters to specify that distribution (see Table one), the basic syntax of the four basic commands are:

`dDIST(x, parameters)` — probability density of DIST evaluated at $x$.
`qDIST(p, parameters)` — returns $x$ satisfying Pr(DIST(parameters) $\leq x) = p$
`pDIST(x, parameters)` — returns Pr(DIST(parameters)$\leq x$)
`rDIST(n, parameters)` – generates $n$ random variables from DIST(parameters)

**Table 1.** Common continuous probability distributions in `R`. If certain parameters are not specified, the default is assumed. For example, a unit normal unless the mean and variance are specified, and a central distribution unless a noncentrality parameter is given. Other continuous distributions are given in Table 2 (below).

| Distribution | Syntax |
|---|---|
| **Normal** | `norm()` — unit normal (the default)<br>`norm(`$\mu, \sigma^2$`)` — normal with mean $\mu$ and variance $\sigma^2$. |
| **Student's t** | `t(df)` — central $t$ with $df$ degrees of freedom (default)<br>`t(df,ncp)` — noncentral $t$ with noncentrality parameter `ncp` |
| $\chi^2$ | `chisq(df)` – central $\chi^2$ with $df$ degrees of freedom (default)<br>`chisq(df,ncp)` – noncentral $\chi^2$ with noncenturality parameter `ncp` |
| $F$ | `f(df1,df2)` – central $F$ with $df1$ and $df2$ degrees of freedom (default)<br>`f(df1,df2,ncp)` — noncentral $F$ with noncenturality parameter `ncp` |

### Plotting Probability Distributions

One very powerful feature of `R` is its ability to quickly plot any number of functions for the desired distribution. The command used for plotting is the `curve` function:

- `curve(function,xlow,xhigh)` – plots `function` between xlow and xhigh

•**`curve(function,xlow,xhigh,n=500)`** – plots **`function`** using 500 equally-spaced points

---

**Example 1:** Suppose we wish to plot a (central) $\chi^2$ distribution with 5 degrees of freedom, say between 0 and 30. To do this in **R** type the following:

```
curve(dchisq(x, 5), 0, 30)
```

typing

```
curve(dchisq(x, 5), 0, 30, n = 500)
```

uses 500 (equally spaced) points to draw the curve

Suppose that we now wish to examine the effect of moving from a central $\chi^2$ distribution to an increasingly noncentral $\chi^2$. The distribution function for the noncentral $\chi^2$ distribution in **R** is given by **`dchisq(x, df,ncp)`**, where **`df`** are the degrees of freedom and **`ncp`** the noncentrality parameter. To add an additional curve to the one plotted, we use the **`add=TRUE`** command in the **`curve`** function. For example, to add a curve for e $\chi^2$ with noncentrality parameter 10,

```
curve(dchisq(x, 5,10), 0, 30,add=TRUE)
```

**R** also lets you specify the color, by using the **`col = "COLOR"`** command, for example

```
curve(dchisq(x, 5,10), 0, 30,add=TRUE,col="pink")
```

adds this curve in pink (**R** has a wide range of colors, so have fun).

---

**Example 2:** **R** can also be used to plot cumulative probability densities. For example,

```
curve(dt(x, 5), 0, 30)
```

plots the cumulative probabilities for a student's $t$ with 5 degrees of freedom for $x$ ranging from -3 to +3.

```
curve(pt(x,5,1), -3, 3, add=TRUE, col="green")
```

overlays a green curve for cumulative probability for a noncentral $t$ with noncentrality parameter 1.

We can also point the required $x$ values for a given $p$ value. For example, for a (central) F distribution with 2 and 20 degrees of freedom, since now $x$ (the probability) ranges from zero to one,

```
curve( qf(x,2,20), 0, 1)
```

---

### Obtaining $p$ and Critical Values

The **`pDIST(x, parameters)`** returns the $p$ value associated with a particular $x$ value drawn from that distribution, i.e., returns Pr(DIST(parameters)$\leq x$). Conversely, the **`qDIST(p, parameters)`** returns the $x$ value to give a particular $p$ value.

---

**Example 3:** . What is the $x$ value from a student's $t$ distribution with 12 degrees of freedom so that there is a 99 % probability that a random value is below $x$? Typing

```
qt(0.99,12)
```

in **R** returns **`2.680998`**.

Likewise, if we have a noncentral $t$ (with noncentrality parameter 1.75) and observe a value of 3.5, the probability of being this value or less is given by

```
pt(3.5,12,1.75)
```

or **`0.91566`**. The $p$ value is one minus this (as Pr(DIST $> x$) = 1-Pr(DIST $\leq x$), or

```
1- pt(3.5,12,1.75)
```

which equals **`0.08433496`**.

---

**Example 4:** Two-side confidence $\alpha$-level confidence intervals are given by $x_{min}, x_{max}$, where Pr(DIST $\leq (x_{min}) = (1-\alpha/2)$ and Pr(DIST $\leq (x_{max}) = 1-(1-\alpha/2)$. For example, an 99.9% confidence interval for a unit normal has $\alpha = 0.999$, and hence $(1-\alpha)/2 = 0.0005$. We can obtain upper and low values in **R** by typing

```
qnorm( 0.0005 )
```

and

```
qnorm( 0.9995 )
```

As an example of using other features of **R**, we can also compute these by first specifying to **R** the $\alpha$ value we wish to use by typing

```
alpha <- 0.999
```

This tells **R** that we have assigned alpha the value of 0.999. The lower and upper (respectively) values follow from

```
qnorm((1-alpha)/2)
```

and

```
qnorm( 1-1-alpha)/2
```

**R** again returns values of `-3.290527, 3.290527`.

Another useful **R** is that we can input a vector of values and for many of the distribution command **R** will output a vector of the designed values. We can define the vector of the upper and lower probabilities (for which we desire associated critical values) by

```
xvals <- c((1-alpha)/2, 1-(1-alpha)/2)
```

Here the command `c()` means create a vector from the (in this case two ) items in the list, and assign the variable `xvals` the value of this vector. Typing in

```
qnorm( xvals )
```

**R** again returns `-3.290527 3.290527`.

**Example 4:** While confidence intervals for normal and student $t$ variables are symmetric, those for $\chi^2$ and $F$ distributions are not. For example, the 95% confidence interval for an $F$ drawn from a distribution with 2 and 20 degrees of freedom is (lower value)

```
alpha <- 0.95
qf((1-alpha)/2,2,20)
```

or `0.02534988` and an upper value of

```
qf(1-(1-alpha)/2,2,20)
```

for `4.461255`.

## Other Continuous Probability Distributions

**Table 2.** Some other continuous probability distributions in **R**. If certain parameters are not specified, the default is assumed. For exact details of how the parameters are defined, see the **R** help file. Note that this is not a complete set, so check the help file if a distribution is not listed here, as **R** likely has it.

| Distribution | Syntax |
|---|---|
| **Exponential** | **`exp()`** — Exponential with rate 1 (the default)<br>**`exp(rate)`** — Exponential with rate set by user |
| **Uniform** | **`unif()`** — Uniform on (0,1) (the default)<br>**`unif(min, max)`** — Uniform over (min, max) |
| **Log Normal** | **`lnorm()`** —log normal with meanlog=0, SD of log = 1 (the default)<br>**`lnorm(meanlog,sdlog)`** —log normal with meanlog, SDlog |
| **Beta** | **`beta(shape1, shape2)`** —beta with shape parameters shape1, shape 2 |
| **Gamma** | **`gamma(shape)`** —gamma with shape user specified, scale=1. (the default)<br>**`gamma(shape,scale)`** — scale parameter set by user. |
| **Weibull** | **`weibull(shape)`** —Weibull with shape user specified, scale=1. (the default)<br>**`weibull(shape,scale)`** — scale parameter set by user. |
| **Wilcox** | **`wilcox(m,n)`** —Distribution for Wilcoxon rank sum statistic with sample sizes m and n. |

### Discrete Probability Distributions

Table 3 lists several of the discrete probability distributions avilable in **R**.

**Table 3.** Some of the discrete probability distributions in **R**.

| Distribution | Syntax |
|---|---|
| **Binomial** | **`binom(n,p)`** — Binomial with sample size $n$ and success probability $p$ |
| **Geometric** | **`geom(p)`** — Geometric with success probability $p$ |
| **Poisson** | **poism($\lambda$)** — Poisson with mean $\lambda$ |
| **Negative Binomial** | **`nbinom(tar,p,mu)`** — Negative Binomial with target $tar$, success probability $p$ and mean $\mu$ |

**Example 5.** In order to plot a discrete function, we must first generate a sequence of integers. **R** can be used to generate a sequence as follows:

```
x <- seq(0,25,1)
```

this generates a seqeunce from 0 to 25 in steps of one and then stores the resulting sequence in the variable x. More generaly to generate a sequence from a to b in steps of c, we would use **`seq(a,b,c)`**.

To plot the first 26 values (for 0 to 25) for a Poisson distribution with parameter $\lambda = 3$, in **R** using the above sequence, now type:

```
plot(dpois(x,3))
```

Adding the **`type =""`** command can change the points into a step latter graph (**`type = "s"`**) or a histogram (**`type ="h"`**), so that

```
plot(dpois(x,3),type="s")
```

plots a step latter function while

```
plot(dpois(x,3),type="h")
```

plots a histogram.

You will notice that the plot values are essentially zero from values greater than 10. If we wish more discrimation, we can plot the log of the probabilities. This is done in the **`dDIST`** command by using the option **`log=TRUE`**, which returns the log of the probabilities (**`log=FALSE`** is the default which is used unless otherwise specificed by the user). Hence, to plot the log of probabilities (as a histogram) for the 51 terms in a Binominal with $n = 50$ and success probability $p = 0.8$, type

```
x <- seq(0,50,1)
```

then

```
plot(dbinom(x,50,0.8,log=TRUE),type="h")
```

while

```
plot(dbinom(x,50,0.8),type="h")
```

plots the probabilities on a linear scale.