VVnA R Package

Aous Abdo

Tuesday, August 12, 2014

Contents

1 Introduction **1**

2 Air Resistance **1**

2.1 Air Resistance at Low Speeds 1

3 Functions **2**

3.1 Projectile Motion 2

3.1.1 Projectile Motion in Vaccum 2

3.1.2 Projectile Motion in Air 4

3.1.3 Comparing Projectile Motion 6

3.2 The jit Function 7

4 Simulations **8**

4.1 Simulation Parameters: 8

1 Introduction

The VVnA "Validation, Verification, and Accreditation" package is a package intended for . . . .

2 Air Resistance

Air resistance is a friction force resulting from different atmoic phenomena and depends on the velocity of the projectile relative to air. For most objects, the direction of the force is always opposite to the main velocity vector.

2.1 Air Resistance at Low Speeds

At low speeds, the force of friction with air can be approximated as a summation of linear an duadratic terms:

$$\begin{aligned} f(v) &= f_{lin} + f_{quad} \\ &= bv + cv^2 \end{aligned}$$

For a spherical object of diametere D, the coefficients of friction are:

$$\begin{aligned} b &= \beta D \\ c &= \gamma D^2 \end{aligned}$$

Where

$$\beta = 1.6 \times 10^{-4}\, N.s/m^2$$
$$\gamma = 0.25\, N.s/m^4$$

For more details on the derivation of the equations used in this package please see this document

3 Functions

3.1 Projectile Motion

Projectile motion in vacuum and in air are calculated with the `projectile` and `projFrictionLin` functions respectively. when considering air friction effects on projectiles, we only consider the viscous drag which is related to the velocity `v`. The Inertial drag realted to the square of the velocity is not treated in this package.

In each of these two cases, a function will return the following projectile parameters:

1. `x`: Displacement in the horizontal direction as a function of time (in meters)
2. `vx`: Speed in the horizontal direction as a function of time (in m/s units)
3. `y`: Displacement in the vertical direction as a function of time (in meters)
4. `vy`: Speed in the vertical direction as a function of time (in m/s units)
5. `y_x`: Displacement in the vertical direction as a function of horizontal displacement (in meters)

In all cases, it is assumed that there are no motion in the lateral direction.

3.1.1 Projectile Motion in Vaccum

Arguments of the `projectile` function are:

1. `v0`: Initial velocity in m/s
2. `y0`: Initial height in m
3. `theta0`: Initial angle in degrees
4. `t`: Time of flight in seconds

For vectors of length `1` for all arguments, the function will return a list of projectile parameters for those arguments. For example for an initial velocity `v` of `30 m/s`, initial height `y` of `0 m`, initial projectile angle `theta0` of `30 degrees`, and at time `t=3 seconds` we get:

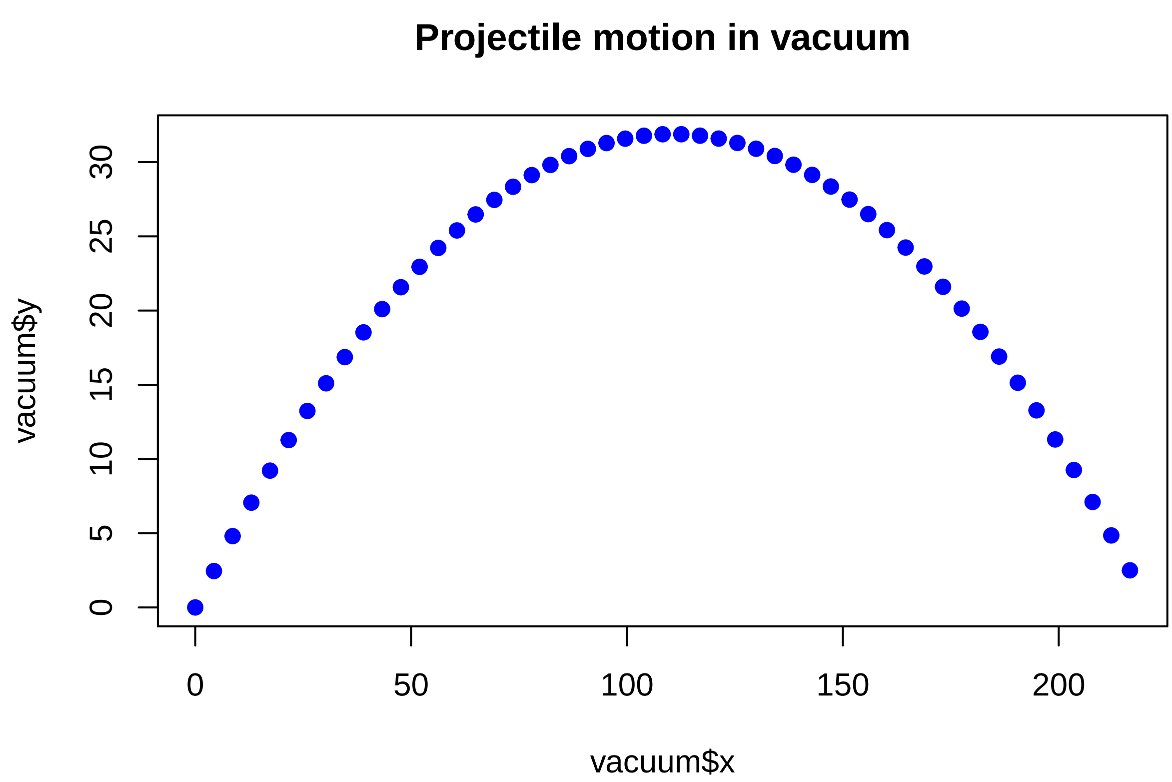
```
projectile(t = 3, y0 = 0, v0 = 30, theta0 = 30)
```

```
## $x
## [1] 77.94229
##
## $vx
## [1] 25.98076
##
## $y
## [1] 0.9
##
## $vy
## [1] -14.4
```

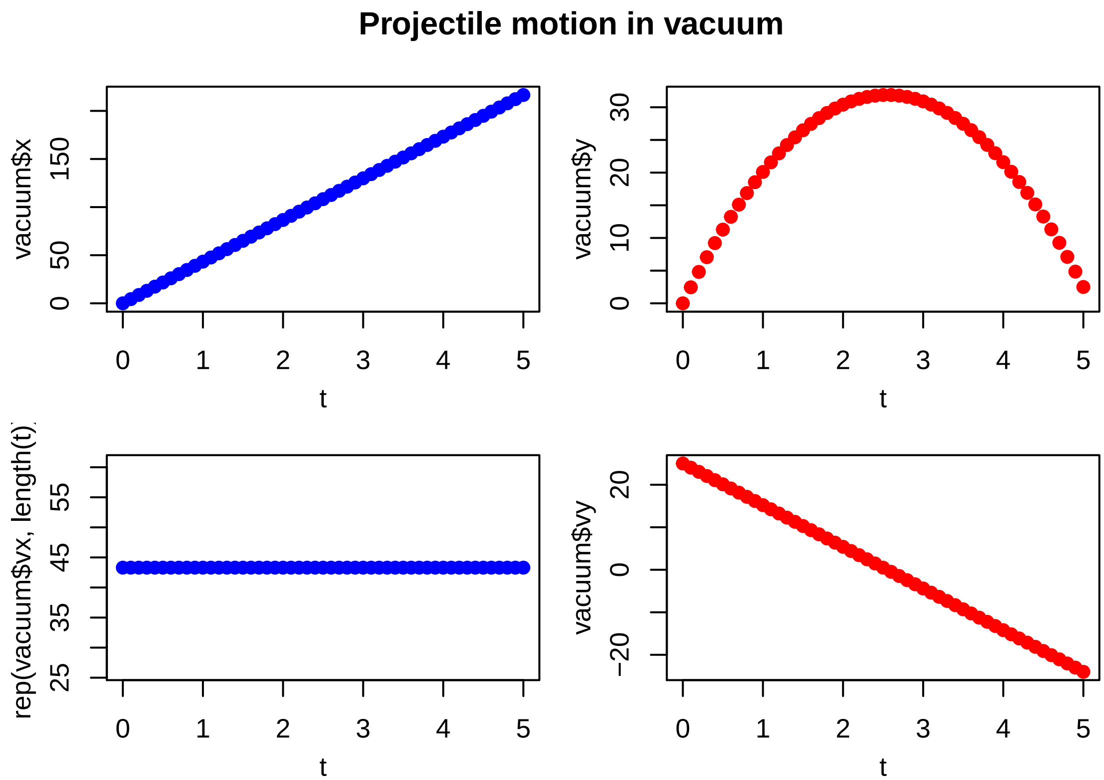
```
##
## $y_x
## [1] 0.9
```

One can also pass a vector of length `> 1` for any individual input parameter. This is most useful for the time parameter `t`:

```
vacuum <- projectile(t = seq(0, 5, 0.1), y0 = 0, v0 = 50, theta0 = 30)
plot(vacuum$x, vacuum$y, pch = 19, col = "blue", main = "Projectile motion in vacuum")
```




```
t <- seq(0, 5, 0.1)
vacuum <- projectile(t = t, y0 = 0, v0 = 50, theta0 = 30)
## prepare
## grid
par(mfcol = c(2, 2), mar = c(3.5, 3, 1, 1), oma = c(2, 2, 2,
    2), mgp = c(2.2, 1, 0))
## plot
## outputs
plot(t, vacuum$x, pch = 19, col = "blue")
plot(t, rep(vacuum$vx, length(t)), pch = 19, col = "blue")
plot(t, vacuum$y, pch = 19, col = "red")
plot(t, vacuum$vy, pch = 19, col = "red")
title("Projectile motion in vacuum", outer = TRUE)
```



3.1.2 Projectile Motion in Air

Arguments of the `projFrictionLin` function are:

1. `v0`: Initial velocity in m/s
2. `y0`: Initial height in m
3. `theta0`: Initial angle in degrees
4. `t`: Time of flight in seconds
5. `b`: drag coefficient in Newtons.seconds/meters
6. `m` mass of object in kg

For vectors of length `1` for all arguments, the function will return a list of projectile parameters for those arguments. For example for an initial velocity `v` of `30 m/s`, initial height `y` of `0 m`, initial projectile angle `theta0` of `30 degrees`, drag coeffiecient `b` of 0.5, mass of projectile `m` of 5 kg, and at time `t=3 seconds` we get:

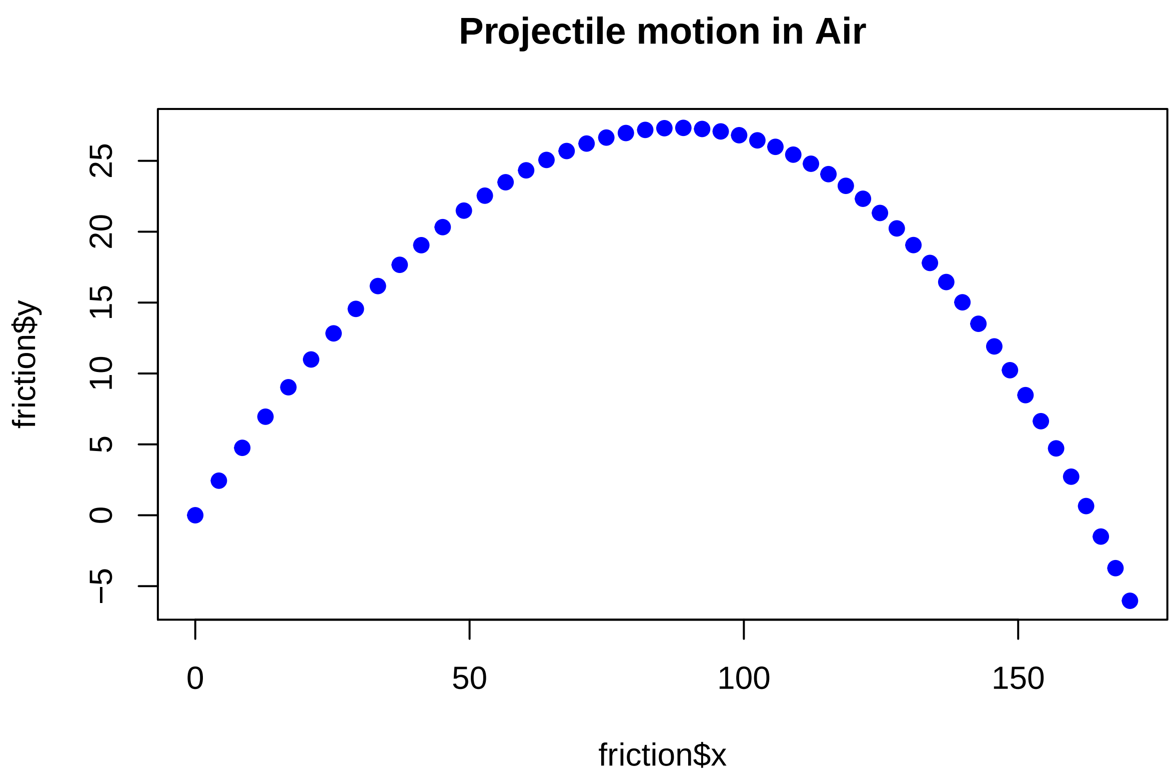
```
projFrictoinLin(t = 3, y0 = 0, v0 = 30, theta0 = 30, b = 0.5,
    m = 5)
```

```
## $x
## [1] 67.3374
##
## $vx
## [1] 19.24702
```

```
##
## $y
## [1] -1.124589
##
## $vy
## [1] -14.28754
##
## $y_x
## [1] -1.124589
```

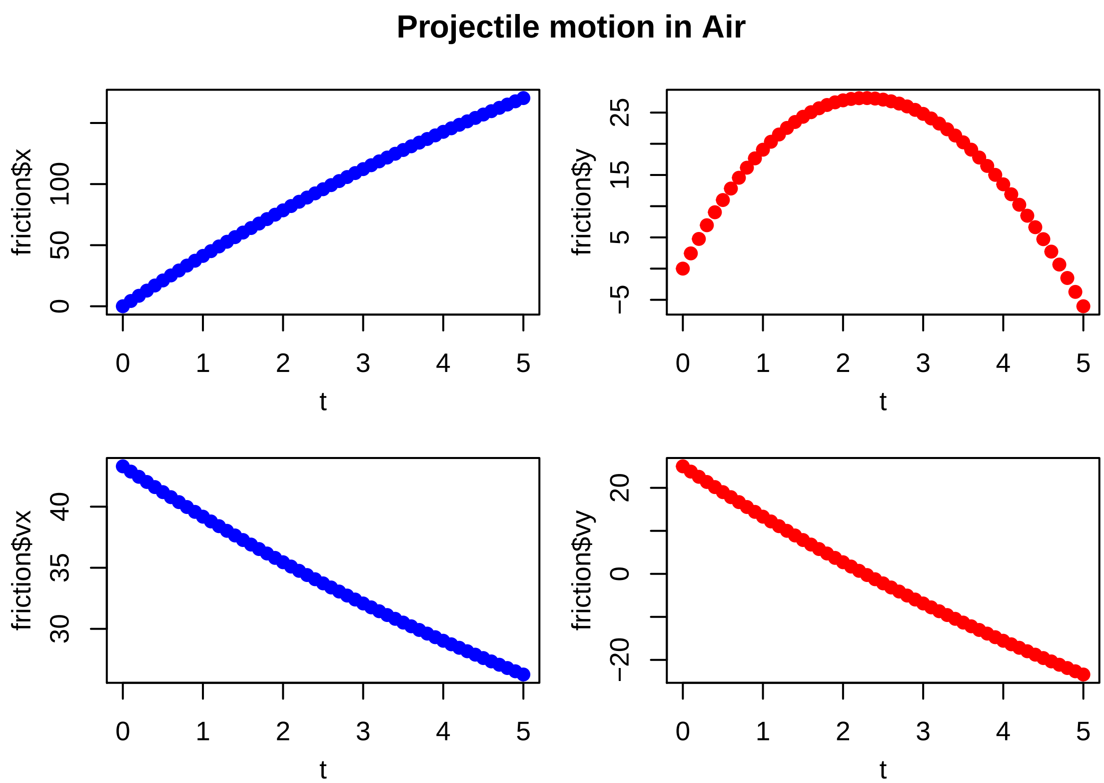
One can also pass a vector of length `> 1` for any individual input parameter. This is most useful for the time parameter `t`:

```
friction <- projFrictoinLin(t = seq(0, 5, 0.1), y0 = 0, v0 = 50,
    theta0 = 30, b = 0.5, m = 5)
plot(friction$x, friction$y, pch = 19, col = "blue", main = "Projectile motion in Air")
```




```
t <- seq(0, 5, 0.1)
friction <- projFrictoinLin(t = t, y0 = 0, v0 = 50, theta0 = 30,
    b = 0.5, m = 5)
## prepare
## grid
par(mfcol = c(2, 2), mar = c(3.5, 3, 1, 1), oma = c(2, 2, 2,
    2), mgp = c(2.2, 1, 0))
## plot
```

```r
## outputs
plot(t, friction$x, pch = 19, col = "blue")
plot(t, friction$vx, pch = 19, col = "blue")
plot(t, friction$y, pch = 19, col = "red")
plot(t, friction$vy, pch = 19, col = "red")
title("Projectile motion in Air", outer = TRUE)
```



3.1.3 Comparing Projectile Motion

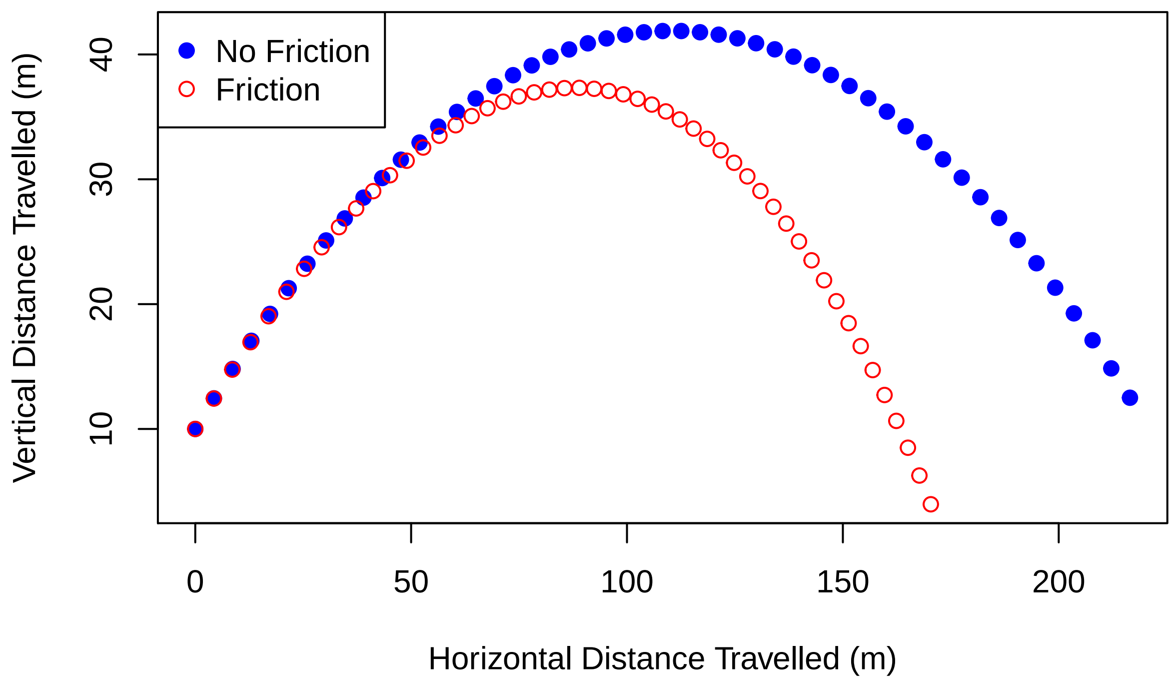
In the example below we compare projectile motion in vacuum to that in air:

```r
par(mfcol = c(1, 1))
vacuum <- projectile(t = seq(0, 5, 0.1), y0 = 10, v0 = 50, theta0 = 30)
friction <- projFrictoinLin(t = seq(0, 5, 0.1), y0 = 10, v0 = 50,
    theta0 = 30, b = 0.5, m = 5)

x <- vacuum$x
y <- vacuum$y
xf <- friction$x
yf <- friction$y_x

plot(x, y, col = "blue", pch = 19, ylim = c(min(y, yf), max(y,
    yf)), xlab = "Horizontal Distance Travelled (m)", ylab = "Vertical Distance Travelled (m)")
points(xf, yf, col = "red", pch = 21)
```

```r
legend(x = "topleft", legend = c("No Friction", "Friction"),
    col = c("blue", "red"), pch = c(19, 21))
```



3.2 The jit Function

The `jit` function is used to randmoize a given vector. It has two modes, `norm` and `uniform`:

`norm`: randomizes a numerical vector by drawing samples from a noraml distribution. `uniform`: randomizes a numerical vector by drawing samples from a uniform distribution.

In the example below we randomize the number `40`, `5` times by drawing from a normal distribution with mean equal to the value to be ranomized, `40`, and a standard deviation equal to 3, passed to the `mean` argument:

```r
jit(x = 40, n = 5, method = "norm", amount = 3)
```

```
## [1] 46.03186 32.81895 39.00934 43.70161 44.07641
```

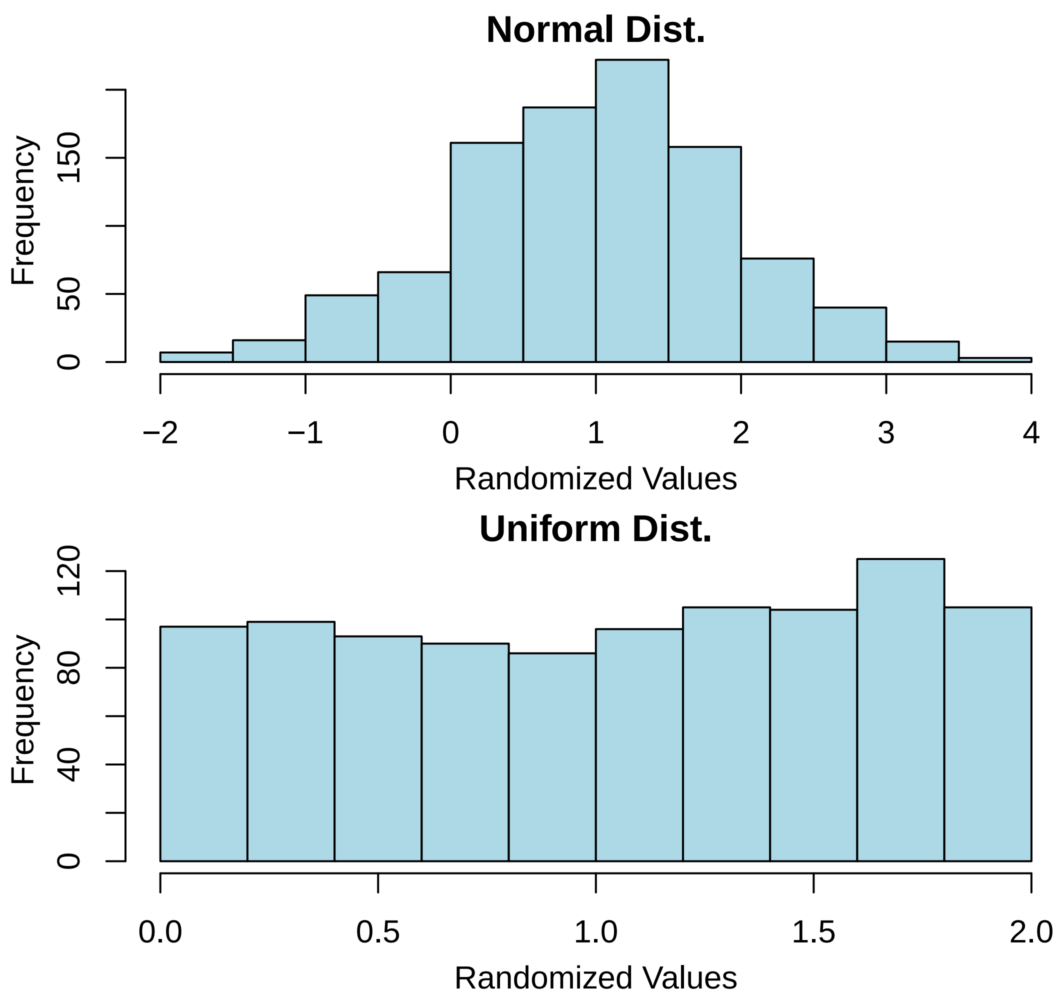
The example below does a similar ranomization, jittering, but with a uniform distribution:

```r
jit(x = 40, n = 5, method = "uniform")
```

```
## [1] 39.98230 39.37995 39.40619 40.61725 39.35667
```

Please notice that the `jit` function with the `uniform` method is essentially equivalent to the `jitter` function in R.

```
par(mfrow = c(2, 1), mar = c(3.5, 3, 1, 1), oma = c(2, 2, 2,
    2), mgp = c(2.2, 1, 0))
hist(jit(x = 1, n = 1000, method = "norm", amount = 1), main = "Normal Dist.",
    xlab = "Randomized Values", col = "lightblue")
hist(jit(x = 1, n = 1000, method = "uniform", amount = 1), main = "Uniform Dist.",
    xlab = "Randomized Values", col = "lightblue")
```



4 Simulations

4.1 Simulation Parameters:

1. θ: Angle with respect to the vertical direction.

2. ϕ: Lateral angle, angle with respect to the horizontal direction.
3. t: Time of firing.

We will be simulating the following uncertainities:

1. $\delta\theta$: Uncertainity in θ
2. $\delta\phi$: Uncertainity in ϕ
3. δt: Uncertainity in firing time t

A detailed example of generating simulations can be found in `example_2` in the `example` directory. An output of this exmaple is shown below: