

VVnA R Package

Aous Abdo

Tuesday, August 12, 2014

Contents

1	Introduction	1
2	Functions	1
2.1	Projectile Motion	1
2.1.1	Projectile Motion in Vacuum	1
2.1.2	Projectile Motion in Air	4
2.1.3	Comparing Projectile Motion	6
2.2	The jit Function	7

1 Introduction

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

2 Functions

2.1 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 \mathbf{v} . The Inertial drag related 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.

2.1.1 Projectile Motion in Vacuum

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 θ_0 of 30 degrees, and at time $t=3$ seconds we get:

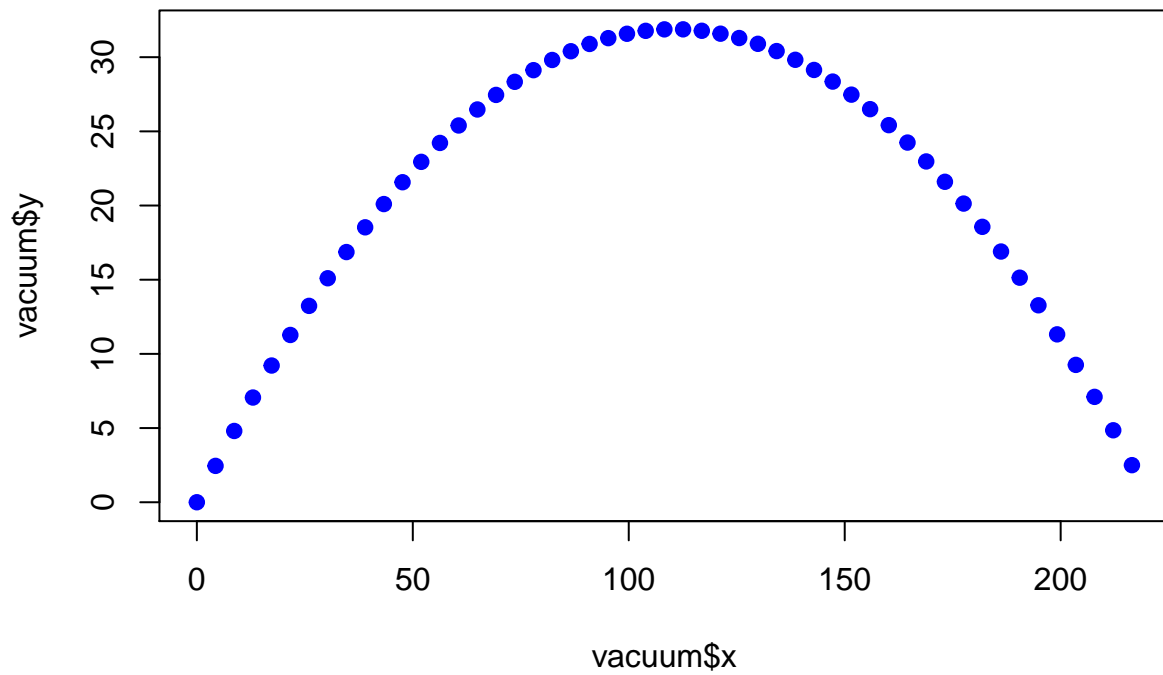
```
projectile(t = 3, y0 = 0, v0 = 30, theta0 = 30)
```

```
## $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 :

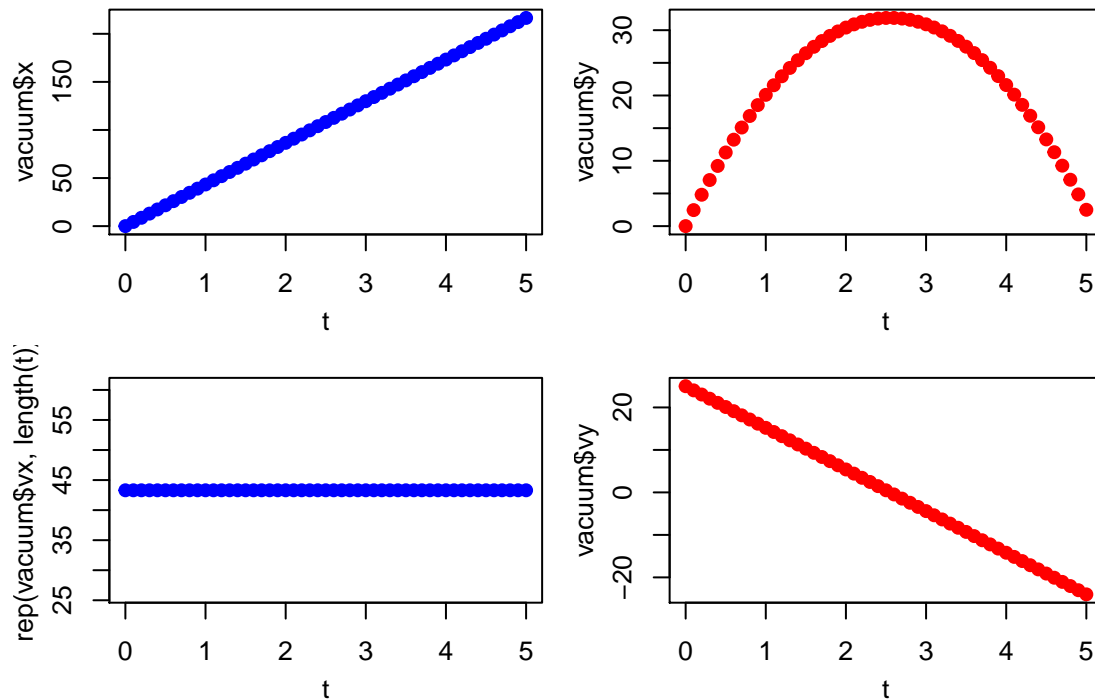
```
vacuum <- projectile(t = seq(0, 5, 0.1), y0 = 0, v0 = 50, theta0 = 30)
plot(vacuum$x, vacuum$y, pch = 19, col = "blue", main = "Projectile motion in vacuum")
```

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)
```

Projectile motion in vacuum



2.1.2 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 θ_0 of 30 degrees, drag coefficient b of 0.5, mass of projectile m of 5 kg, and at time $t=3$ seconds we get:

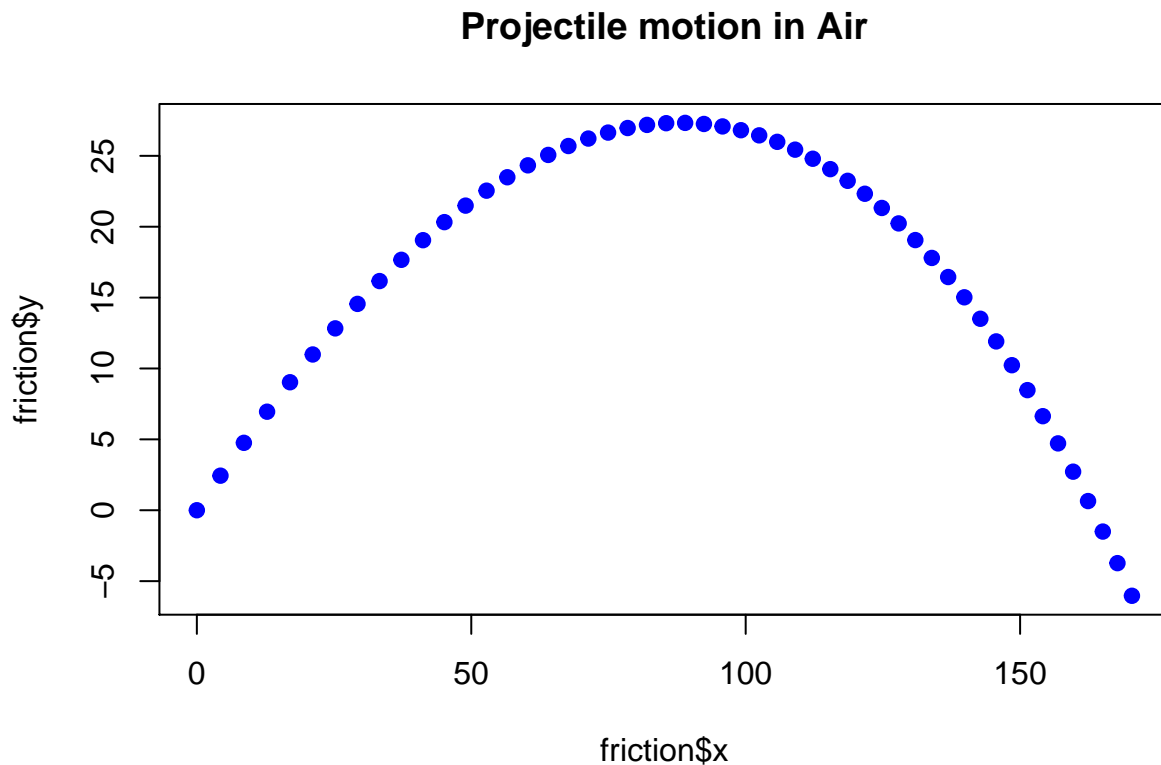
```
projFrictionLin(t = 3, y0 = 0, v0 = 30, theta0 = 30, b = 0.5,
  m = 5)
```

```
## $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:

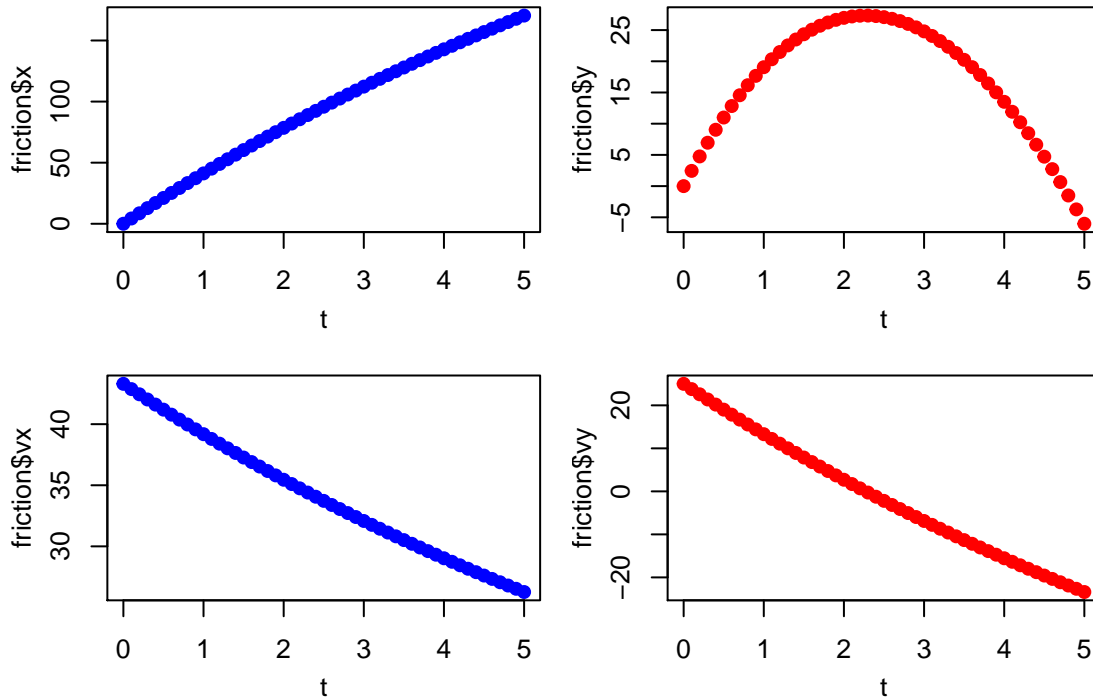
```
friction <- projFrictoinLin(t = seq(0, 5, 0.1), y0 = 0, v0 = 50,
  theta0 = 30, b = 0.5, m = 5)
plot(friction$x, friction$y, pch = 19, col = "blue", main = "Projectile motion in Air")
```



```
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
```

```
## 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)
```

Projectile motion in Air



2.1.3 Comparing Projectile Motion

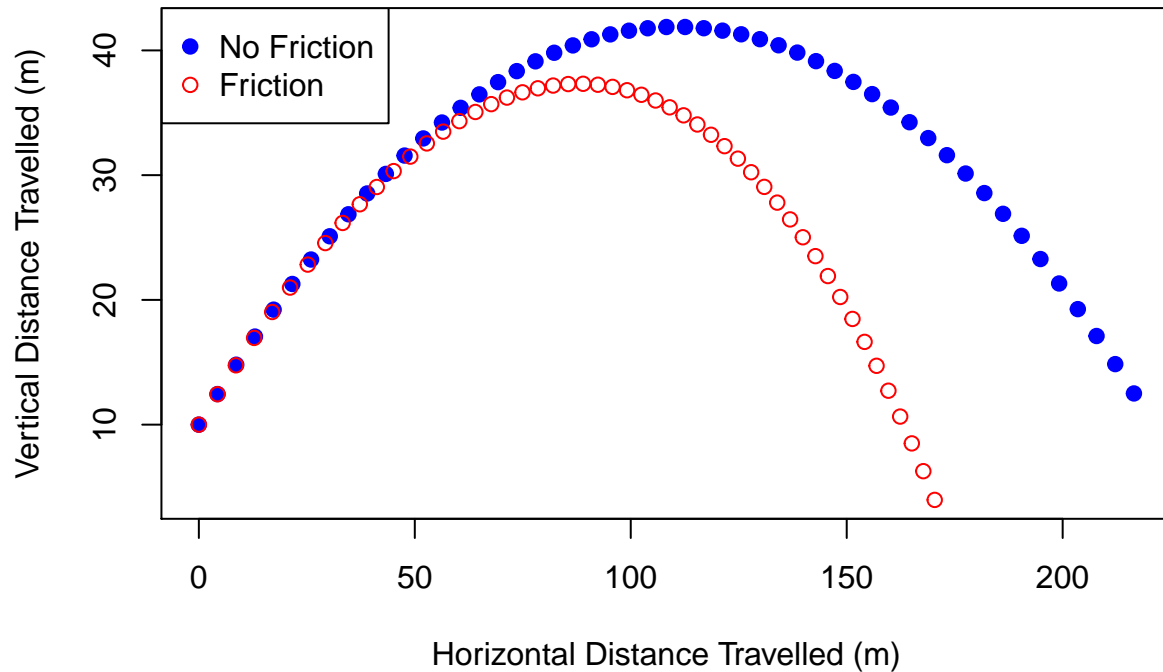
In the example below we compare projectile motion in vacuum to that in air:

```
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)
```

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



2.2 The jit Function

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

norm: randomizes a numerical vector by drawing samples from a normal 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 randomized, 40, and a standard deviation equal to 3, passed to the `mean` argument:

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

```
## [1] 38.94622 31.85091 40.57524 36.13563 36.59262
```

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

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

```
## [1] 39.75170 40.12106 39.49572 40.64177 39.21818
```

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")
```

