Working examples with R

Statistics 427: R Programming

Module 16

2020

To Infinity, and Beyond! How about the Sol System?

Start with Newton's law of gravitation and look at the implication of the shape of the Earth's orbit is. Numerically solving the gravity equations to obtain directly the project path of the Earth around the sun.

There will be many calculations involved and will be using `R` for what its meant to do. Do try this at home!

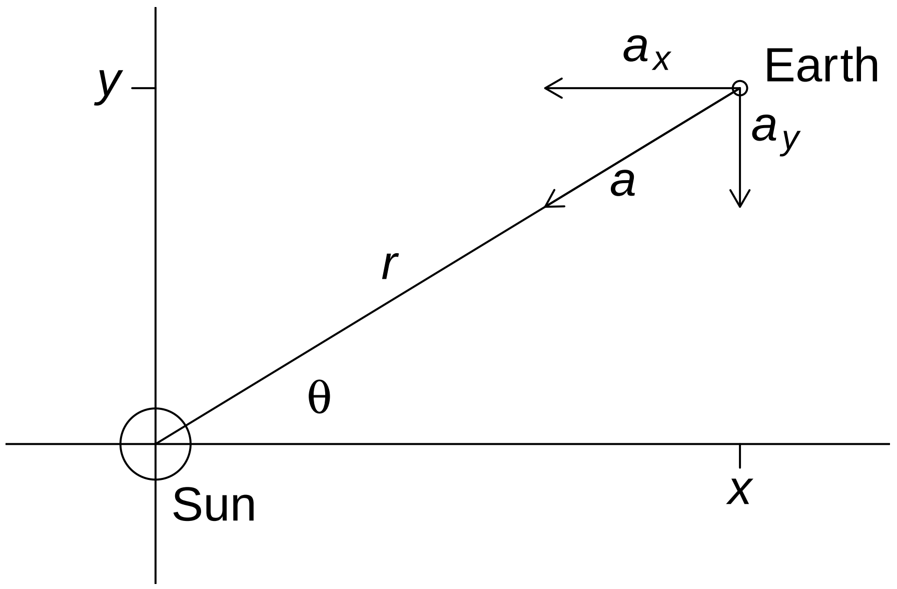
Newton's law

Newton's universal law states that there is a force of attraction between any two bodies (objects) having mass. The force decreases with the square of the distance between the two masses and it proportional to the product of the two masses. Newton was able to show mathematically that this law *implies* Kepler's laws of planetary motion; further it also predicts phenomena beyond Kepler's ellipses.

For instance, according to Newton's law, an object with sufficient velocity will not orbit another much larger object but will rather execute a flyby in which its trajectory is bent by the large object's gravity. *Apollo 13* made use of this when it was disabled on its way to the moon in 1970. They executed a flyby of the moon that sling-shotted (roadrunner) them around the moon and returned home safely (yay!).

Setup

Set up a coordinate system for the Earth (the name Earth is boring, I am going to call her Terra from now on; and it is unfair our planet has such a boring name considering the rest of our solar system's planets are freaking GODS! Why did we not name her Hera or Juno…) and the sun.



Setup

(1) the sun is at the origin with Earth currently at point (x, y)
(2) Denote mass of sun M, mass of Earth m
(3) goal is to calculate the trajectory of Earth using Newton's law and the changes in position over a small time interval (a few hours)
(4) Update Earth's position with new points (x, y)
(5) Calculate another position…

Velocities

To calculate different (x, y) pairs for positions, we need to know how fast everything is moving around out there. Let v_x and v_y be the velocities of x and y, respectively. If t is time and t is a very small interval of time, calculate a change in x by the product $v_x t$ and a change in y by the product $v_y t$. One thing to remember is that the values of x and y change during every time interval but the velocities of Earth in those directions will also change. Gravity is a force so it changes the velocity of an object. Force values will differ with every different distance of Earth from the sun. Every time interval will need a calculation of how much v_x and v_y change, in addition to changes in x and y.

Review of Newton's laws

Newton's second law of motion:

$$F = ma$$

F: force acting upon object
m: mass of Earth
a: acceleration of Earth

Review of Newton's laws

According to Newton's law of gravitation, the gravitational attraction force between the sun and Earth, denoted by F_{Mm}, obeys an inverse-square relationship:

$$F_{Mm} = -G\frac{Mm}{r^2}$$

G: universal gravity constant
M: mass of sun
m: mass of Earth
r: distance between the centers of sun and Earth

Review of Newton's laws

For gravity, these laws are expressions of the same force:

$$F = F_{Mm}$$

$$\therefore ma = -G\frac{Mm}{r^2}$$

Review of Newton's laws

Canceling the ms:

$$a = -G\frac{M}{r^2}$$

This acceleration is always in the direction of the sun. The previous equations are for a one-dimensional coordinate system. The negative sign for gravity designates that gravity acceleration tends to decrease the position of the Earth on a line connecting the Earth and the sun. The current general formalization of Newton's laws uses concepts form vector calculus for independence of coordinate system.

Application of gravity acceleration

Gravity acceleration has a component in both x and y directions. In the previous figure, the arrow labeled a depicts the gravitational acceleration pulling Earth in the direction of the sun. The trig functions explored in a previous module gives us

$$a_x = a\cos\theta$$

$$a_y = a\sin\theta$$

Yay trigonometry again!

θ is the angle made by the line connecting the Earth to the sun with the line given by the x-axis. Using Pythagoras theorem, we can express a, $\cos\theta$, $\sin\theta$, and r in the previous components all in coordinates of x and y

By the aid of Pythagoras

$$r^2 = x^2 + y^2$$
$$r = \sqrt{x^2 + y^2} = (x^2 + y^2)^{1/2}$$
$$\cos\theta = \frac{x}{y}$$
$$\sin\theta = \frac{y}{r}$$

substitutions in algebra

Now substitute back in

$$a_x = a\cos\theta = \left(-\frac{GM}{r^2}\right)\left(\frac{x}{r}\right) =$$
$$-\frac{GMx}{(x^2+y^2)^{3/2}}$$

and

$$a_y = a\sin\theta = \left(-\frac{GM}{r^2}\right)\left(\frac{y}{r}\right) =$$
$$-\frac{GMy}{(x^2+y^2)^{3/2}}$$

Velocities

It is the current accelerations of x and y, calculated for the current location of the Earth (x and y), that allow us to compute changes in the velocities of v_x and v_y. The amount of change in v_x during t is the product of the rate at which v_x changes and t; but the rate at which v_x changes is just the acceleration component a_x (same basic approach for v_y as well)

$$v_x = a_x t \textit{ and } v_y = a_y t$$

Move the Earth. . . that's where I keep all my stuff

Yes, move the Earth to its future positions with new values of x, y, v_x, v_y, and t are calculated as the old values (previous position in time) plus the changes.

$$x_{new} = x + x_{old}$$
$$y_{new} = y + y_{old}$$

$$\vdots$$

Odd and ends before launch

Needed:

(1) G, M, starting values of x, y, v_x, v_y, and the value of t
(2) Measurement units to use for distance, time, and mass
   (a) *distance* will use AU. 1 AU is the average distance from Earth to the sun (about 149,597,871 *km*)
   (b) *time* will use a sidereal year (time it takes for one revolution around the sun, from periapsis to periapsis (aka perihelion, where Earth is closest to)). On a side note, the sidereal year is about 20 minutes longer than the tropical year we use (solstice to solstice)
   (c) *mass* is great! we conveniently use the mass of Earth as 1 unit
(3) Think of the vectors needed; 2 (large) vectors `x` and `y`, with successive values calculated by a `for` loop
(4) Plot it

Outline of script

(1) set values for initial location, velocity, and direction of Earth
   (a) the major (long) axis of the elliptical orbit to lie on the x-axis
   (b) the periapsis is at a point slightly less than 1.0 on the x-axis and y is 0; this will be initial position of Earth
   (c) Earth will be crossing the x-axis vertically at the initial instant in the direction increasing for y (initial angle will be ϕ with value set at $\pi/2$)

(2) Set total duration of time for which the trajectory is to be calculated and the total number of small time intervals per sidereal year to use; total time is 1 sidereal year (other planets will need longer time duration) and small time intervals per sidereal year will be 10,000 (larger number will result in more accurate trajectory)

Still working on To-do list

(3) Set values for constants
(4) Rescale physical constants and computationally easy units
  (a) G is measured in meters^3 kilograms^{-1} seconds^{-2}
  (b) Distance is 1 AU
  (c) mass is 1 unit
  (d) time is sidereal year
  (e) convert G to new units with new expression $G_{new} = G(\text{meters per AU})^{-3}(\text{kilograms per earth mass})(\text{seconds per sidereal year})^2$
(5) Calculate needed initial quantities: `tot.inc` is different from the number of small intervals (`n.int`) in 1 sidereal year; creation of empty vectors to dump calculations into from the loop
(6) Initialize new vectors with initial values; the product GM appears in the velocity change calculations so it will be factored out and multiplied outside the loop

Almost there. . . almost there. . .

(7) The loop: `i` will go from `1:tot.inc` and need to calculate:
  (a) change in x
  (b) change in y
  (c) change in v_x
  (d) change in v_y
  (e) calculate and store x_{i+1}
  (f) calculate and store y_{i+1}
  (g) calculate and store $v_{x_{i+1}}$
  (h) calculate and store $v_{y_{i+1}}$
  (i) calculate and store t_{i+1}
(8) Plot y vs. x positions with `plot()`

Trajectory script

```
x0=0.983291 # Periapsis distance in astronomical units
y0=0           # Start on x-axis, at periapsis
phi=pi/2       # Initial direction angle of earth's orbit
v0=30.3        # Earth's initial velocity at periapsis, km/s

n.int=10000    # Number time increments per Earth yr
t.end=1        # Number of years duration of trajectory

G=6.67428e-11         # Gravitational constant, m^3/(kg*s^2)
m=6.0e24              # Mass of earth, kg
```

```
M=m*3.33e5            # Mass of sun
km.AU=149.6e6             # One astronomical unit, km
sec.yr=31558149.540     # Seconds in a sidereal year
```

Trajectory script

```
me=m/m      # Mass of earth expressed in earth units
Me=M/m      # Sun's mass in earth units
G=G*m*sec.yr^2/((km.AU^3)*(1000^3))  # New units of G=AU^2/(me*yr^2)
v0=v0*sec.yr/km.AU     # New units of v are AU/yr

v.x0=v0*cos(phi)      # Earth's velocity, x-component, at periapsis
v.y0=v0*sin(phi)      # Earth's velocity, y-component, at periapsis
dt=1/n.int            # Duration of one small time interval (delta t)
tot.int=round(t.end*n.int);  # Total number of small intervals (integer)

x=numeric(tot.int+1)  #  zero vector of ninc+1 for initial condition
y=x           # Allocate y the same as x
v.x=x         # Allocate v.x the same as x
v.y=x         # Allocate v.y the same as x
t=x           # Allocate t the same as x
```

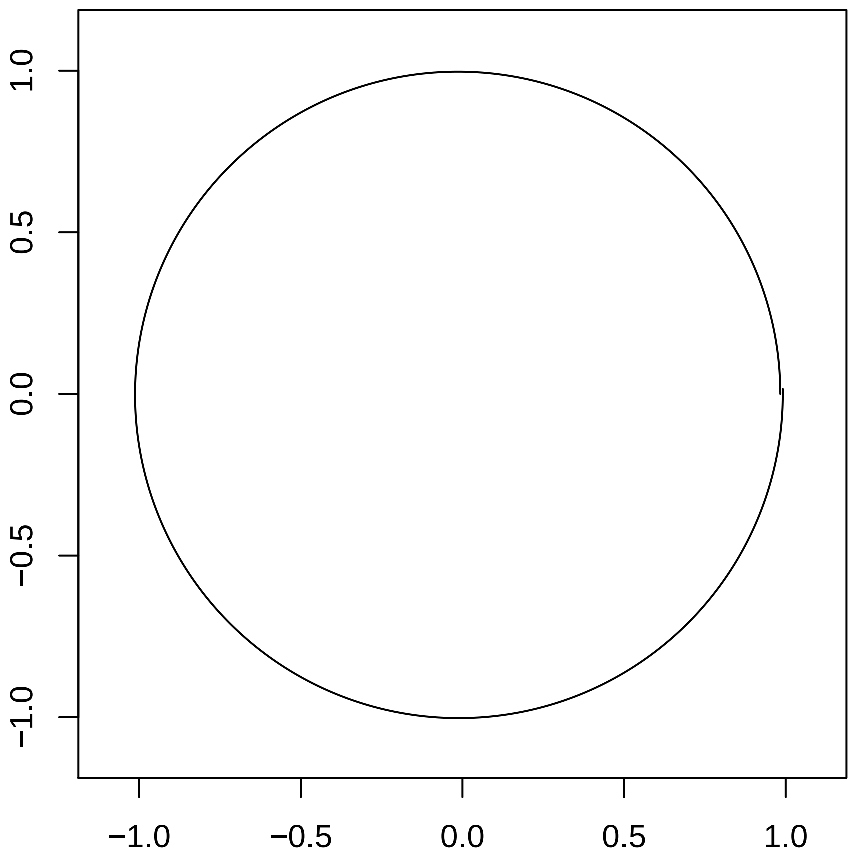
Trajectory script

```
x[1]=x0          # Initial x-position
y[1]=y0          # Initial y-position
v.x[1]=v.x0      # Initial x-direction velocity
v.y[1]=v.y0      # Initial y-direction velocity
t[1]=0           # Initial t
c=G*Me           # Pre-calculate a constant that appears repeatedly

for (i in 1:tot.int) {
  dx=v.x[i]*dt     # Change in x
  dy=v.y[i]*dt     # Change in y
  dv.x=-c*x[i]/(x[i]^2+y[i]^2)^(3/2)*dt    # Change in vx
  dv.y=-c*y[i]/(x[i]^2+y[i]^2)^(3/2)*dt    # Change in vy
  x[i+1]=x[i]+dx     # New value of x
  y[i+1]=y[i]+dy     # New value of y
  v.x[i+1]=v.x[i]+dv.x    # New value of vx
  v.y[i+1]=v.y[i]+dv.y    # New value of vy
  t[i+1]=t[i]+dt     # New value of t
}
```

Trajectory plot

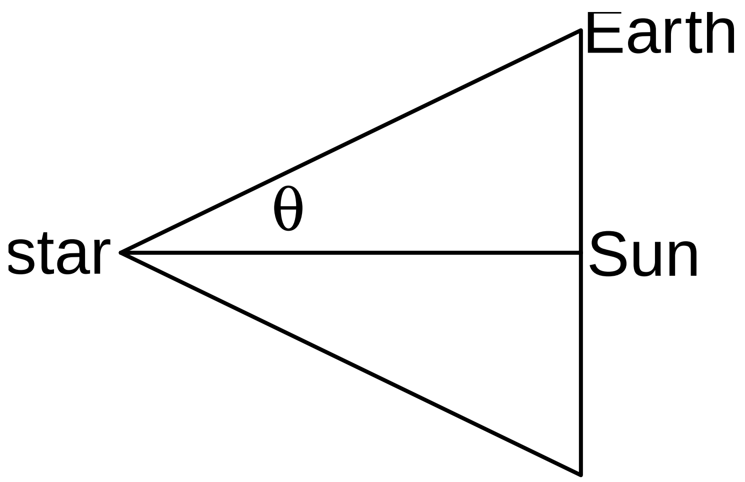
```
par(pin=c(4,4))      # Equal screen size in x and y directions
plot(x,y,type="l",xlim=c(-1.1,1.1),ylim=c(-1.1,1.1)) # Line plot of y vs x.
```



Distances to stars near the solar system

Accurately measuring distances to stars near our solar system use a method of triangulation called *parallax*. The angle to the star is measured when Earth is on one side of its orbit and then measure the angle gain six months later when Earth is on the opposite side. The amount by which the angle has changed by 2θ (half the amount of change in θ is called the parallax), we can construct a right triangle, shown in next slide

Parallax



Parallax

Using 1 AU as the (average) distance from Earth to the sun, the equation for distance from the sun to the star is $distance = 1/\tan\theta$. The angle represented by θ in the previous slide for Proxima Centauri (the closest star to the sun) is 3.276×10^{-6} radians. The distance in light years (LYs) from the fact that 1 LY is about 63279 AU (light travels from the sun to Earth in just over 8 minutes)

Distance to Proxima Centauri

```r
theta=3.276e-06
d.au=1/tan(theta); d.au
```

```
[1] 305250.3
```

```r
d.ly=d.au/63270; d.ly
```

```
[1] 4.824566
```

Light takes almost five years to get from Proxima Centauri to Earth

Projectile motion

Let x be the horizontal distance a projectile travels, y is the vertical distance the projectile can reach. If an object is thrown at an angle of θ radians at an initial velocity v_0 distance per time (use meters per second), the object's initial velocity in the x direction is $v_0 \cos\theta$ and the initial velocity in the y direction is $v_0 \sin\theta$.

Equation time!

If the object is thrown while on a level surface, the horizontal distance x traveled by the object after t seconds is described (discounting air resistance) by

$$x = t(v_0 \cos\theta)$$

The height of the object above the ground after t seconds, assuming it was initially released at a height of y_0 meters is

$$y = y_0 + t(v_0 \sin\theta) - \frac{gt^2}{2}$$

g: gravitational constant ($g \approx 9.81 m/s^2$)
Note that x is a linear function of t and the equation for y a quadratic function of t

More projectile

The object hits the ground when $y = 0$. The time t_{max} at which this happens is solved by the quadratic equation

$$t_{max} = \frac{-b - \sqrt{b^{-}4ac}}{2a}$$

(the + would be the minimum because this quadratic function opens down)
$a = -g/2$, $b = v_0 \sin\theta$, and $c = y_0$. The farthest distance traveled before hitting the ground is $x_{max} = t_{max}(v_0 \cos\theta)$

Projectile script

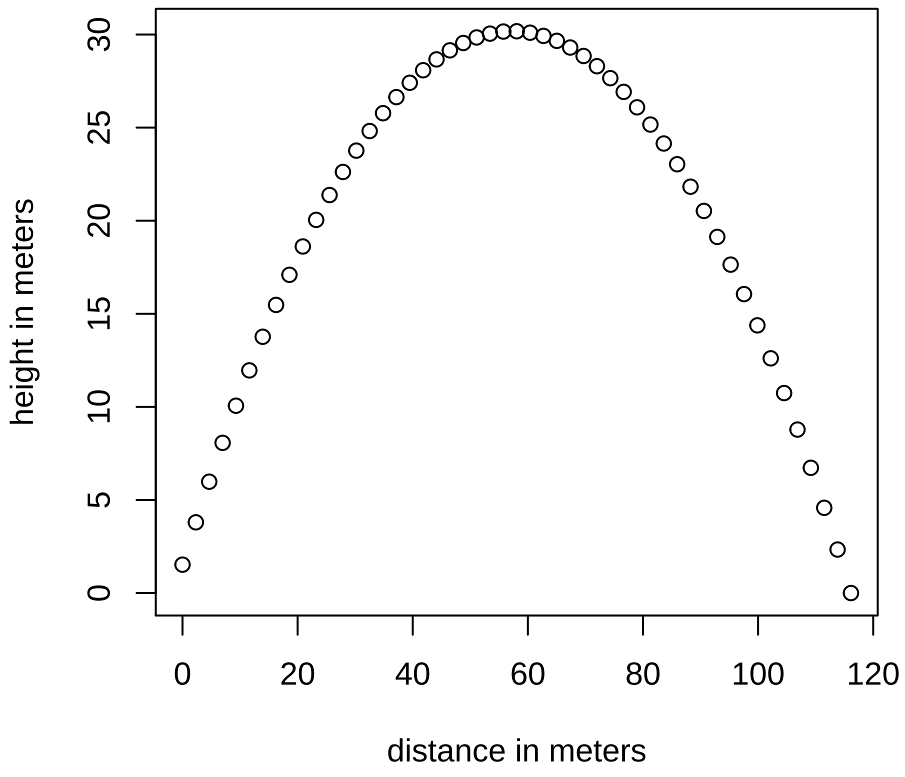
```r
mph=75           # Initial velocity, mph
angle=45         # Initial angle, degrees
height=5         # Initial height, ft

# Convert units to meters and seconds
v0=mph*1609.344/(60*60)  # Convert velocity to m/s
theta=2*pi*angle/360     # Convert angle to radians
y0=height/3.2808399      # Convert height to m
g=9.80665                # Gravitational acceleration constant

a=-g/2
b=v0*sin(theta)
c=y0
t.max=(-b-sqrt(b^2-4*a*c))/(2*a)  # flight max time
x.max=v0*cos(theta)*t.max         # max distance
```

Projectile script

```r
t=t.max*(0:50)/50  # Range of t values between 0 and t.max.
x=v0*cos(theta)*t
y=y0+v0*sin(theta)*t-g*t^2/2
plot(x,y,xlab='distance in meters',ylab='height in meters')
```



```r
t.max; x.max        # Print t.max,x.max
```

```
[1] 4.898511

[1] 116.1333
```

Beam me back up (like, yesterday!)

Newton's law of gravity can be used to derive a polar curve equation for the orbit of a planet around its star or the orbit of a satellite around a planet.

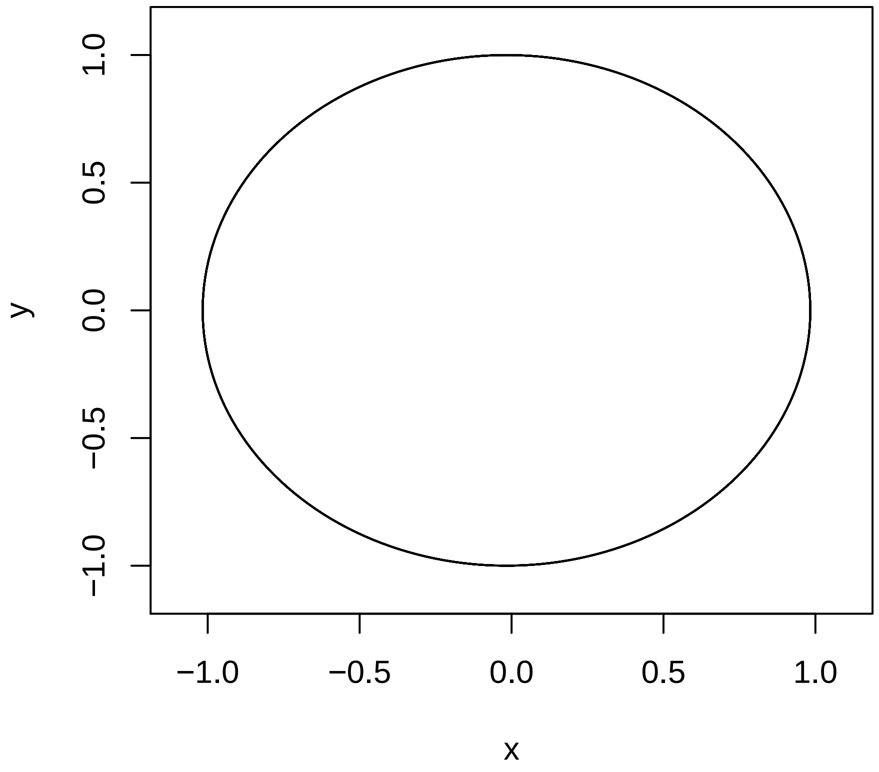
$$r = r_0 \frac{1+\varepsilon}{1+\varepsilon\cos\theta}$$

Eccentricity

The sun or body being orbited is assumed to be the origin. In the equation r_0 is the distance from the point of closest approach (periapsis) of the two bodies, and ε is the *eccentricity* (departure from circularity) of the orbit. The periapsis and eccentricity both depend on the initial velocity and direction of the movement of the orbiting body in ways that hurt my brain. For Earth's orbit around the sun, r_0 is about 0.98329 AU and ε is about 0.016711.

Eccentricity

```r
r0=0.98329; e=0.016711
theta=10*pi*(0:1000)/1000
r=r0*(1+e)/(1+e*cos(theta))
x=r*cos(theta); y=r*sin(theta)
plot(x,y,type="l",lty=1,xlim=c(-1.1,1.1),ylim=c(-1.1,1.1))
```



y
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
x