sunmoon Group

John D. Baker

May 25, 2020

Contents

sunmoon Overview

`sunmoon` is a collection of basic astronomical algorithms The key verbs are `moons`, `sunriseset0` and `sunriseset1.`
All of these verbs were derived from BASIC programs published in *Sky & Telescope* magazine in the 1990's. The rest of the verbs in `sunmoon` are mostly date and trigonometric utilities.

sunmoon Interface

calmoons	*calendar dates of new and full moons*
moons	*times of new and full moons for n calendar years*
sunriseset0	*computes sun rise and set times - see group documentation*
sunriseset1	*computes sun rise and set times - see group documentation*

sunriseset0 *v*– sunrise and sunset times

This verb has been adapted from a BASIC program submitted by Robin G. Stuart *Sky & Telescope's* shortest sunrise/set program contest. Winning entries were listed in the March 1995 Astronomical Computing column.

The J version of this algorithm has been vectorized. It can compute any number of sunrise and sunset times in one call.

The `(y)` argument is a `5*n` floating point table where:

```
0{ is latitude in degrees with northern latitudes positive.
1{ is longitude in degrees with western longitudes negative.
2{ is western time zones expressed as positive whole hours.
3{ is the month number.
4{ is the day number.
```

The result is a numeric table with four rows. To handle the cases when the sun never rises or sets the first two elements of the corresponding result columns are:

```
0{ is NORISESET an invalid hour indicating no rise or set
1{ is 0 when the sun never rises
1{ is 1 when the sun never sets
```

Warning: this algorithm breaks for latitudes close to the South pole.

The original BASIC code has been slightly modified to use control structures in place of GOTO's and line numbers.

Adapted from:

```
/* Sunrise/set by R. G. Stuart,  Mexico City, Mexico */
PI = 3.14159265#: DR = PI / 180: RD = 1 / DR
INPUT "Lat, Long (deg)"; B5, L5
INPUT "Time zone (hrs)"; H
B5 = DR * B5
INPUT "Month, day"; M, D
N = INT(275 * M / 9) - 2 * INT((M + 9) / 12) + D - 30
L0 = 4.8771 + .0172 * (N + .5 - L5 / 360)
C = .03342 * SIN(L0 + 1.345)
C2 = RD * (ATN(TAN(L0 + C)) - ATN(.9175 * TAN(L0 + C)) - C)
SD = .3978 * SIN(L0 + C): CD = SQR(1 - SD * SD)
SC = (SD * SIN(B5) + .0145) / (COS(B5) * CD)
IF ABS(SC) <= 1 THEN
  C3 = RD * ATN(SC / SQR(1 - SC * SC))
  R1 = 6 - H - (L5 + C2 + C3) / 15
```

```
  HR = INT(R1): MR = INT((R1 - HR) * 60)
  PRINT USING "Sunrise at ##:##"; HR; MR
  S1 = 18 - H - (L5 + C2 - C3) / 15
  HS = INT(S1): MS = INT((S1 - HS) * 60)
  PRINT USING "Sunset at ##:##"; HS; MS
ELSEIF SC > 1 THEN
  PRINT "Sun up all day"
ELSEIF SC < -1 THEN
  PRINT "Sun down all day"
END IF
END
```

```
monad: ntRiseset =. sunriseset0 flBLHMD

  NB. rise and set times at Dog Lake today (daylight savings)
  td=. (44 + 19%60),(- 76 + 21%60), 4 , }. today 0
  sunriseset0 td

  NB. rise and set times on June 30 on Greenwich meridian
  t0=.   0 0 0 6 30   NB. equator
  t1=.  49 0 0 6 30   NB. north - lat of western US/Canada border
  t2=. _47 0 0 6 30   NB. south - southern Chile and Argentina
  t3=.  75 0 0 6 30   NB. far north (sun always up)
  t4=. _75 0 0 6 30   NB. far south (sun always down)

  sunriseset0 t0

  sunriseset0 t0 , t1 , t2 , t3 ,: t4
```

```
NB. times on equator for March 21 for all 1 hour time zones
sunriseset0  0 0 ,"1 (,.i. 24) ,"1 ] 3 21

NB. times for calendar year 1995 on the Greenwich meridian
md95=.  47 0 0 ,"1 }."1 yeardates 1995
rs095=. sunriseset0 md95
```

sunriseset1 *v*– sunrise and sunset times

This verb has been adapted from a BASIC program submitted by James Brimhall to *Sky & Telescope's* "shortest sunrise/set program" contest. Winning entries were listed in the March 1995 Astronomical Computing column.

The `(y)` argument of `sunriseset1` is a `6*n` floating point table where:

```
0{ is latitude in degrees with northern latitudes positive.
1{ is longitude in degrees with western longitudes negative.
2{ is western time zones expressed as positive whole hours.
3{ is the month number.
4{ is the day number.
5{ is the year.
```

The result is a numeric table with four rows. To handle the cases when the sun never rises or sets the first two elements of the corresponding result columns n are:

```
0{ is NORISESET an invalid hour that indicates no rise or set.
1{ is 0 when the sun never rises.
1{ is 1 when the sun never sets.
```

Adapted from:

```
/* Sunrise/set by James Brimhall, St. Albans, WV */
PI = 3.1415927#: DR = PI / 180: DD = 360 / 365.25636#: DIM D(20)
DATA 0,31,59,90,120,151,181,212,243,273,304,334
FOR C = 1 TO 12: READ D(C): NEXT C
INPUT "Lat, long"; LA, LO
INPUT "Month, day, year"; M, D, Y
IF Y / 4 = INT(Y / 4) THEN FOR C = 3 TO 12: D(C) = D(C) + 1: NEXT C
DY = D(M) + D: DY = DY - LO / 360: DY = DY + .5
TH = 9.357001 + DD * DY + 1.914 * SIN(DR * (DD * DY - 3.97))
C3 = .3978 * COS(DR * TH)
DC = -1 / DR * ATN(C3 / (SQR(1 - C3 ^ 2)))
IF LA < 0 THEN A1 = 90 + LA - DC ELSE A1 = 90 - LA + DC: PRINT
IF A1 < -50 / 60 THEN PRINT "Sun never rises": GOTO 200
IF LA < 0 THEN A2 = -90 - LA - DC ELSE A2 = LA - 90 + DC
IF A2 >= -50 / 60 THEN PRINT "Sun never sets": GOTO 200
C1 = (SIN(-DR * 50 / 60) - SIN(DR * DC) * SIN(DR * LA)) / (COS(DR * DC) * COS(DR * LA))
T2 = (1 / DR) * ATN(SQR(1 - C1 ^ 2) / C1): IF C1 >= 0 THEN T1 = 360 - T2
IF C1 < 0 THEN T2 = 180 + T2: T1 = 360 - T2
TR = T1 / 15 - 12: TS = T2 / 15
ET = -.1276 * SIN(DR * (DD * DY - 3.97)) - .1511 * SIN(DR * (2 * DD * DY + 17.86))
TR = TR - ET: TS = TS - ET
INPUT "Time zone (h)"; TZ
TC = -TZ - LO / 15: R = TR + TC: S = TS + TC
C2 = (SIN(DR * DC) - SIN(DR * LA) * SIN(-DR * 50 / 60)) / (COS(DR * LA) - SIN(DR * LA) * SIN(-DR * 50 / 60))
Z1 = (1 / DR) * ATN(SQR(1 - C2 ^ 2) / C2): IF C2 >= 0 THEN Z2 = 360 - Z1
IF C2 < 0 THEN Z1 = 180 + Z1: Z2 = 360 - Z1
```

```
PRINT "Sunrise "; INT(R); "h"; INT(600 * (R - INT(R))) / 10;
PRINT "m a.m., azimuth "; INT(10 * Z1) / 10
PRINT "Sunset "; INT(S); "h"; INT(600 * (S - INT(S))) / 10;
PRINT "m p.m., azimuth "; INT(10 * Z2) / 10
200 END
```

```
monad:  ntRiseset =. sunriseset1 flBLHMDY

  NB. rise and set times at Dog Lake today (daylight savings)
  td=. (44 + 19%60),(- 76 + 21%60), 4 , 1 |. today 0
  sunriseset1 td

  NB. rise and set times on June 30 1995 on Greenwich meridian
  t0=.   0 0 0 6 30 1995  NB. equator
  t1=.  49 0 0 6 30 1995  NB. north - lat of western US/Canada border
  t2=. _47 0 0 6 30 1995  NB. south - southern Chile and Argentina
  t3=.  75 0 0 6 30 1995  NB. far north (sun always up)
  t4=. _75 0 0 6 30 1995  NB. far south (sun always down)

  sunriseset1 t0

  sunriseset1 t0 , t1 , t2 , t3 ,: t4

  NB. compare algorithms
  sun1=. sunriseset1 t0 , t1 , t2 , t3 ,: t4
  sun0=. sunriseset0 }:"1 t0 , t1 , t2 , t3 ,: t4
  sun1 - sun0
```

```
NB. times on equator for March 21 1995 for all 1 hour time zones
sunriseset1 0 0 ,"1 (,.i. 24) ,"1 ] 3 21 1995

NB. times for calendar year 1995 on the Greenwich meridian
mdy95=. 47 0 0 ,"1 ] 1 |."1 yeardates 1995
rs195=. sunriseset1 mdy95
```

sunmoon Source Code

```
NB.*sunmoon s-- computes sun rise/set times and full new moon dates.
NB.
NB. verbatim:
NB.
NB. interface word(s):
NB. ----------------------------------------------------------------------------
NB.  calmoons    - calendar dates of new and full moons
NB.  moons       - times of new and full moons for n calendar years
NB.  sunriseset0 - computes sun rise and set times - see group documentation
NB.  sunriseset1 - computes sun rise and set times - see group documentation
NB.
NB. author:  John D. Baker -- bakerjd99@gmail.com
NB. created: 2010feb12
NB. ----------------------------------------------------------------------------
NB. 2010feb12 sunmoon group converted to class
NB. 2012oct03 documentation adjusted for (jodliterate)

coclass 'sunmoon'

NB.*end-header

NB. interface words (IFACEWORDSsunmoon) group
IFACEWORDSsunmoon=: <;._1 ' calmoons moons sunriseset0 sunriseset1'

NB. indicates sun never rises or sets in (sunriseset0) and (sunriseset1) results
NORISESET=: 99
```

```
NB. root words (ROOTWORDSsunmoon) group
ROOTWORDSsunmoon=: <;._1 ' IFACEWORDSsunmoon ROOTWORDSsunmoon calmoons sunriseset0 sunriseset1 today yeardates'

NB. arc tangent
arctan=: _3&o.

calmoons=: 3 : 0

NB.*calmoons v-- calendar dates of new and full moons. 0's denote
NB. new moons and 1's denote full moons.
NB.
NB. monad:  it =. calmoons ilYears
NB.
NB.   calmoons 1900 2000

NB. compute Julian dates and convert to calendar
j=. moons y
t=. fromjulian <. {. j

NB. attach new (0) and full (1) bits
j=. 0 [ t=. (, |: {: j) ,"0 1 ,/ t

NB. eliminate year overlap and duplicate dates
~. t #~  (1 {"1 t) e. y
)

NB. cosine radians
cos=: 2&o.
```

```
fromjulian=: 3 : 0

NB.*fromjulian v-- converts Julian day numbers to dates, converse
NB. (tojulian).
NB.
NB. monad:  itYYYYMMDD =. fromjulian nlJulian
NB.
NB.   juldayno=. 1 tojulian 17770704 19530702 20000101 20331225
NB.   fromjulian juldayno
NB.
NB. dyad:  i[1,2]YYYYMMDD =. fromjulian nlJulian
NB.
NB.   0 fromjulian juldayno   NB. monad
NB.   1 fromjulian juldayno

0 fromjulian y
:
NB. Gregorian Calendar correction
b=.      2299161 <: y
jalpha=. <. 36524.25 %~ _0.25 + y - 1867216
ja=.     (y * -. b) + b * y + 1 + jalpha - <. 0.25 * jalpha

jb=. ja + 1524
jc=. <. 6680.0 + ((jb - 2439870) - 122.1) % 365.25
jd=. <. (365 * jc) + 0.25 * jc
je=. <. (jb - jd) % 30.6001
```

```
id=. (jb - jd) - <. 30.6001 * je
mm=. je - 1
mm=. mm - 12 * mm > 12

iyyy=. jc - 4715
iyyy=. iyyy - mm > 2
iyyy=. iyyy - iyyy <: 0

NB. convert result format
if. x do. 100 #. |: iyyy , mm ,: id  else. |: iyyy , mm ,: id end.
)

moons=: 3 : 0

NB.*moons v-- times of new and full moons for n calendar years.
NB.
NB. The  result  is  rank 3 numeric array  where  ({.  moons) are
NB. Julian day numbers and ({: moons)  is  a  logical  mask  with
NB. (0)'s denoting new moons and (1)'s denoting full moons.
NB.
NB. monad:  ftJulian=. moons ilYears
NB.
NB.        moons 1996 1997 2002

NB.  vector J                          scalar Basic
y=. , y
r1=. 1r180p1                          NB.  R1=3.14159265/180
k0=. <. 12.3685 * y - 1900            NB.  K0=INT((Y-1900)*12.3685)
```

```
t=.  (y - 1899.5) % 100                NB.  T=(Y-1899.5)/100
t2=. *: t [ t3=. t^3                   NB.  T2=T*T: T3=T*T*T
j0=. 2415020 + 29 * k0                 NB.  J0=2415020+29*K0
f0=. (0.0001178*t2) - 0.000000155*t3   NB.  F0=0.0001178*T2-0.000000155*T3
f0=. f0 + 0.75933 + 0.53058868*k0      NB.  F0=F0+0.75933+0.53058868*K0
f0=. (f0-(0.000837*t))-0.000335*t2     NB.  F0=F0-0.000837*T-0.000335*T2
m0=. k0 * 0.08084821133                NB.  M0=K0*0.08084821133
m0=. 359.2242 + 360 * 1 | m0           NB.  M0=360*(M0-INT(M0))+359.2242
m0=. m0 - 0.0000333*t2                 NB.  M0=M0-0.0000333*T2
m0=. m0 - 0.00000347*t3                NB.  M0=M0-0.00000347*T3
m1=. k0 * 0.07171366128                NB.  M1=K0*0.07171366128
m1=. 306.0253 + 360 * 1 | m1           NB.  M1=360*(M1-INT(M1))+306.0253
m1=. m1 + 0.0107306*t2                 NB.  M1=M1+0.0107306*T2
m1=. m1 + 0.00001236*t3                NB.  M1=M1+0.00001236*T3
b1=. k0 * 0.08519585128                NB.  B1=K0*0.08519585128
b1=. 21.2964 + 360 * 1 | b1            NB.  B1=360*(B1-INT(B1))+21.2964
b1=. b1 - 0.0016528*t2                 NB.  B1=B1-0.0016528*T2
b1=. b1 - 0.00000239*t3                NB.  B1=B1-0.00000239*T3

NB. rank conjuntion vectorizes BASIC loop
k9=. i. 29                             NB.  FOR K9=0 TO 28
j=.  j0 +"1 0 ] 14*k9                  NB.  J=J0+14*K9
f=.  f0 +"1 0 ] 0.765294*k9            NB.  F=F0+0.765294*K9
k=.  k9 % 2                            NB.  K=K9/2
m5=. r1 * m0 +"1 0 k*29.10535608       NB.  M5=(M0+K*29.10535608)*R1
m6=. r1 * m1 +"1 0 k*385.81691806      NB.  M6=(M1+K*385.81691806)*R1
b6=. r1 * b1 +"1 0 k*390.67050646      NB.  B6=(B1+K*390.67050646)*R1
```

```
f=.  f - 0.4068 * sin m6                 NB.  F=F-0.4068*SIN(M6)
f=.  f + (0.1734 - 0.000393*t) *"1 1 sin m5   NB. F=F+(0.1734-0.000393*T)*SIN(M5)
f=.  f + 0.0161 * sin 2*m6               NB.  F=F+0.0161*SIN(2*M6)
f=.  f + 0.0104 * sin 2*b6               NB.  F=F+0.0104*SIN(2*B6)
f=.  f - 0.0074 * sin m5-m6              NB.  F=F-0.0074*SIN(M5-M6)
f=.  f - 0.0051 * sin m5+m6              NB.  F=F-0.0051*SIN(M5+M6)
f=.  f + 0.0021 * sin 2*m5               NB.  F=F+0.0021*SIN(2*M5)
f=.  f + 0.0010 * sin m6 -~ 2*b6         NB.  F=F+0.0010*SIN(2*B6-M6)
j=.  j + f                               NB.  J=J+INT(F): F=F-INT(F)
u=.  0 1 $~ # k9                         NB. IF U=0 THEN PRINT " NEW MOON ";
j ,: |: (#y) # ,: u                      NB. IF U=1 THEN PRINT "FULL MOON ";
)

NB. round (y) to nearest (x) (e.g. 1000 round 12345)
round=: [ * [: (<.) 0.5 + %~

NB. sine radians
sin=: 1&o.

sunriseset0=: 3 : 0

NB.*sunriseset0 v-- computes sun rise and set times - see group documentation.
NB.
NB. monad:  itHM =. sunriseset0 ilBLHMD | ftBLHMD

NB. latitude, longitude, time-zone, month, day !(*)=. b l h m d
y=. # b [ 'b l h m d'=. |: tabit y
```

```
b=. dr * b [ rd =. % dr=. 1r180p1

NB. day number within year
n=. _30 + d + (<.9 %~ 275 * m) - 2 * <. 12 %~ m + 9

NB. sun's mean longitude
lg0=. 4.8771 + 0.0172 * (n + 0.5) - 1 % 360

NB. equation of time
c=. 0.03342 * sin lg0 + 1.345
c2=. rd * c -~ (arctan tan lg0 + c) - arctan 0.9175 * tan lg0 + c
cd=. %: 1 - *: sd=. 0.3978 * sin lg0 + c
sc=. (0.0145 + sd * sin b) % cd * cos b

NB. to handle the three cases enmass without redundant calculations
NB. a boolean table is computed.  1's in each row satisfy a case.
items=. i. #sc [ cases=. (<&_1 , 1&>:@| ,: 1&<) sc

NB. set result table to sun never sets
hrmn=. |: (y , 4)$ NORISESET , 1 0 0

NB. adjust for the sun's declination and atmospheric refraction
pos=. items #~ 1 { cases
c3=. rd * arctan (pos{sc) % %: 1 - *: pos{sc
lc=. (pos{l) + pos{c2

NB. time zone adjusted sunrise times
```

```
st=. (6 - pos{h) - (lc + c3) % 15
mn=. <.(st - hr) * 60 [ hr=. <. st
hrmn=. hr (<0;pos)} hrmn
hrmn=. mn (<1;pos)} hrmn

NB. time zone adjusted sunset times
st=. (18 - pos{h) - (lc - c3) % 15
mn=. <.(st - hr) * 60 [ hr=. <. st
hrmn=. hr (<2;pos)} hrmn
hrmn=. mn (<3;pos)} hrmn

NB. sun never rises and result table with rows hr,mr,hs,ms
pos=. items #~ 0 {cases
0 (<1;pos)} hrmn
)

sunriseset1=: 3 : 0

NB.*sunriseset1 v-- computes sun rise and set times - see group documentation.
NB.
NB. monad:  itHM =. sunriseset1 flBLHMDY | ftBHMDY

NB. latitude, longitude, time-zone, month, day, year !(*)=. la lo tz m d y
y=. # la [ 'la lo tz m d y'=. |: tabit y
dr=. 1r180p1 [ dd=. 360 % 365.25636 [ rt=. 50r60

NB. days into year with leap year adjustment
dm=. 0 31 59 90 120 151 181 212 243 273 304 334
```

```
dl=. (2 {. dm) , >: 2 }. dm
bl=. 0 = 4 | y [ m=. <: m
dy=. d + ((-.bl) * m { dm) + bl * m { dl
dy=. 0.5 + dy - lo % 360

NB. (th) angle Earth has moved since winter solstice
th=. 9.357001 + (dd * dy) + 1.914 * sin dr * (dd * dy) - 3.97
c3=. 0.3978 * cos dr * th
dc=. (- % dr) * arctan c3 % %: 1 - c3 ^ 2

NB. adjust for positive and negative latitudes
bl=. la < 0
a1=. ((-.bl) * (90 - la) + dc) + bl * (90 + la) - dc
a2=. ((-.bl) * (la - 90) + dc) + bl * (_90 - la) - dc

NB. sun never rises or sets masks
nvset =. a2 >: - rt [ nvrise=. a1 < - rt

NB. corrections
drla=. dr * la   [ drdc=. dr * dc
c1=. ((sin - dr * rt) - (sin drdc) * sin drla) % (cos drdc) * cos drla
t2=. dr %~ arctan (%: 1 - c1 ^ 2) % c1
t1=. 360 - t2 [ bl=. c1 < 0
t2=. (t2 * -.bl) + bl * 180 + t2
t1=. (t1 * -.bl) + bl * 360 - t2

NB. first order equation of time
```

```
et=. 0.1511 * sin dr * 17.86 + 2 * dddy=. dd * dy
et=. (_0.1276 * sin dr * dddy - 3.97) - et
drla=. drdc=. dddy=. 0

NB. time zone adjusted rise and set times
tr=. (t1 % 15) - 12 [ ts=. t2 % 15
tr=. tr - et [ ts=. ts - et
s=. ts + tc [ r=.tr + tc [ tc=.(-tz) - lo % 15
hrmn=. (<. r) ,: 1 round 60 * 1|r
hrmn=. hrmn , (<.12 + s) ,: 1 round 60 * 1|s

NB. adjust for when sun never rises or sets
hrmn=. hrmn *"1 -. bl [ bl=. nvset +. nvrise
hrmn=. NORISESET (<0;bl # pos) } hrmn [ pos=. i. {: $ hrmn
1 (<1;nvset # pos) } hrmn
)

NB. promotes only atoms and lists to tables
tabit=: ]`,:@.(1&>:@(#@$))^:2

NB. tan radians
tan=: 3&o.

today=: 3 : 0

NB.*today v-- returns todays date.
NB.
```

```
NB. monad:  ilYYYYMMDD =. today uu
NB.
NB.   today 0    NB. ignores argument
NB.
NB. dyad:  iaYYYYMMDD =. uu today uu
NB.
NB.   0 today 0

3&{.@(6!:0) ''
:
0 100 100 #. <. 3&{.@(6!:0) ''
)

yeardates=: 3 : 0

NB.*yeardates v-- returns all valid dates for n calendar years.
NB.
NB. The monad returns an integer table with YYYY MM DD rows.  The
NB. dyad returns dates as a list of YYYYMMDD integers.
NB.
NB. This algorithm  uses  a  series of outer-products  and  ravel
NB. reductions to form a cross  product rather  than  the  direct
NB. catalog verb ({).
NB.
NB. monad:  itYYYYMMDD =. yeardates ilYears
NB.
NB.   yeardates 2000
NB.
```

```
NB.   yeardates 2001 + i. 100  NB. all dates in 21st century
NB.
NB.
NB. dyad:  ilYYYYMMDD =. uu yeardates ilYears
NB.
NB.   0 yeardates 2001
NB.
NB.   yeardates~  1999 2000 2001   NB. useful idiom

NB. generate all possible dates in years
days =. ,/ (,y) ,"0 1/ ,/ (>: i. 12) ,"0/ >: i. 31

NB. remove invalid dates
days #~ valdate days
:
NB. convert to yyyy mm dd format
0 100 100 #. yeardates y
)

NB.POST_sunmoon post processor.

smoutput IFACE=:  (0 : 0)
NB. (sunmoon) interface word(s):
NB. ----------------------------
NB. calmoons     NB. calendar dates of new and full moons
NB. moons        NB. times of new and full moons for n calendar years
NB. sunriseset0  NB. computes sun rise and set times - see long documentation
NB. sunriseset1  NB. computes sun rise and set times - see long documentation
```

```
)

cocurrent 'base'
coinsert  'sunmoon'
```

Index