1D Raytracing, Velocity models in R

Jonathan M. Lees
University of North Carolina, Chapel Hill
Department of Geological Sciences
CB #3315, Mitchell Hall
Chapel Hill, NC 27599-3315
email: jonathan.lees@unc.edu
ph: (919) 962-1562

July 23, 2012

```
> options(width=60)
> options(prompt=" ")
 options(continue=" ")
 options(SweaveHooks=list(fig=function()
 par(mar=c(5.1, 4.1, 1.1, 2.1))))
 JPOST<-function(file="tmp", width = 8, height = 8)
   {
     postscript(file=file, width = width,
            height = height, paper = "special",
         horizontal = FALSE, onefile = TRUE, print.it = FALSE)
 }
 library(RSEIS)
 library(Rquake)
```

1 Rquake

In this document I will illustrate how to use **Rquake** , a non-linear earthquake location program.

2 Data Structures and Lists

2.1 Station File

Station location information can be stored in memory (in a list) or in a text file on disk. The station file is a table, with name, lat, lon, and elevation.

For example:

```
 fsta = "/Users/lees/Site/CHAC/staLLZ.txt"
 ###  system(paste(sep=" ", "cat", fsta), intern = TRUE )
```

```
CHAC0   -0.39377412    -78.15369741 3588
CHAC1   -0.366526404   -78.16962049  3606
CHAC2   -0.42485567    -78.2710065   4020
```

```
CHAC3   -0.4524493    -78.18676153    4328
CHAC4   -0.461317213   -78.21783387   4412
CHAC5   -0.351938598   -78.21809574   4000
CHAC6   -0.408928292   -78.20667762    3860
CHAC7   -0.39837847    -78.22075601    4109
CHAC8   -0.382639731  -78.2023599    3767
CHAC9   -0.323852103  -78.15061344  3762
```

These can be scanned in **R** with a simple command.

See **REIS** for more details on stations.

If the stations are in UTM coordinates, you may convert to Lat-Lon using the GEOmap package.

```
stas = scan(file=fsta,what=list(name="", lat=0, lon=0, z=0))
stas$z = stas$z/1000
```

Units in Rquake are in km, so the meters are converted.

REIS has a function for reading in the stations:

```
stas = setstas("stas")
```

2.2 Velocity Structure

The one-dimensional velocity model is also stored in file (or stored in memory in an **R** session). See **REIS** for details.

Sample velocity model stored on disk. In this case no estimates of error are provided, so they are set to zero. If S-wave velocity is not available, can use $V_s = V_p/\sqrt{3}$.

```
#MODEL WU COSO REGINAL FINE LAYERS REGIONAL VELOCITY MODEL
#P DEPTH   P VEL      PERR      S DEPTH    S VEL    SERR
   0.00     4.50      0.00      0.00        2.43    0.00
   0.50     4.51      0.00      0.50        2.59    0.00
   1.00     4.92      0.00      1.00        2.97    0.00
```

1.50	4.92	0.00	1.50	2.97	0.00
2.00	5.46	0.00	2.00	3.15	0.00
2.50	5.46	0.00	2.50	3.15	0.00
3.00	5.54	0.00	3.00	3.27	0.00
3.50	5.54	0.00	3.50	3.27	0.00
4.00	5.58	0.00	4.00	3.42	0.00
5.50	5.58	0.00	5.50	3.42	0.00
12.00	6.05	0.00	12.00	3.49	0.00
20.00	7.20	0.00	20.00	4.15	0.00

The following is a constructor for making a 1D velocity model suitable for use in RSEIS and Rquake:

```
VEL=list()
    VEL$'zp'=c(0,0.25,0.5,0.75,1,2,4,5,10,12)
    VEL$'vp'=c(1.1,2.15,3.2,4.25,5.3,6.25,6.7,6.9,7,7.2)
    VEL$'ep'=c(0,0,0,0,0,0,0,0,0,0)
    VEL$'zs'=c(0,0.25,0.5,0.75,1,2,4,5,10,12)
    VEL$'vs'=c(0.62,1.21,1.8,2.39,2.98,3.51,3.76,3.88,3.93,4.04)
    VEL$'es'=c(0,0,0,0,0,0,0,0,0,0)
    VEL$'name'='/data/wadati/lees/Site/Hengil/krafla.vel'
```

There are several default velocity models available in **REIS** . Function defaultVEL(i) will return one of 6 “standard” models used for different purposes.

If you have a velocity model on disk, you can read it in with **REIS** function, Get1Dvel.

To compare a set of different velocity models visually, try,

```
data(ASW.vel)
     data(wu_coso.vel)
     data(fuj1.vel)
     data(LITHOS.vel)
```

These can be plotted with the routine:

```
Comp1Dvels(c("ASW.vel","wu_coso.vel",  "fuj1.vel" , "LITHOS.vel"  ))
```

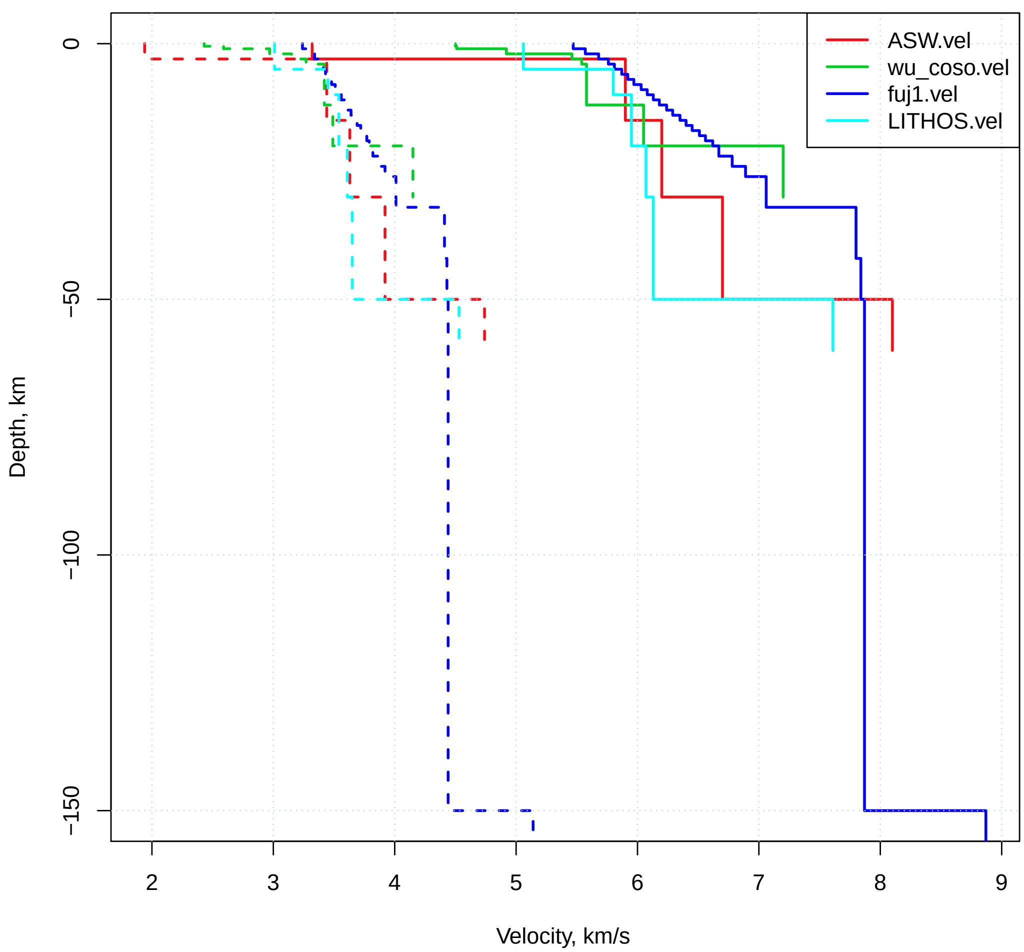

Figure 1: Comparison of 4 sample velocity models

2.3 Arrival Time List

The arrival times, or the picks are stored in in list mode, i.e. a list of vectors each with attributes relating to the arrival time pick.

These vectors are described as:

tag character tag the should be unique

name character, station name

comp character, component name

c3 character, three-component station id sta.hhh.BHZ

phase character, phase name

err numeric, error

pol character polarity, U, D, 0

flg numeric, flag, used in location

res numeric, travel time residual relative to model

dur numeric, duration

yr numeric, year

mo numeric, month

dom numeric, day-of-month

jd numeric, julian day

hr numeric, hour

mi numeric, minute

sec numeric, second

col numeric, or character, color for plotting in RSEIS

onoff numeric, less than 0 means do not use

A constructor for creating an empty pick list is cleanWPX. For many of the functions in RSEIS and Rquake the list must contain filled vectors for each element. use function repairWPX to fill out list elements that are deficient.

The arrival time list has one attribute, the “ID”. This can be used to identify earthquake with a unique tag or identifaication number or name.

For **Rquake** , the elements that are absolutely *required* are: name, phase, err, sec.

There are many different ways to store arrival time picks. It does not matter how these are stored, as long as they are read into R and formatted properly. By disassociating the input format from the analysis, we can simply write a short input, or conversion, routine to use all the codes as is.

We can thus store the data in any format we desire, perhaps for use in other non-R software.

2.3.1 Native (binary) R

The output of **swig** is binary R file, so the data can simply be loaded automatically.

2.3.2 UW format Pickfiles

loadUWpickfiles is a function that reads in a list of pickfiles stored on disk and returns a list of picked events.

Since UW pickfiles store the times relative to a common minute mark, and station information is not stored in the pickfile, this information is filled out in the code:

```
KF = vector(mode="list")
   for(i in 1:length(LF))
     {
       g1 = getpfile(LF[i])
       m1 = match(g1$STAS$name, stas$name)
       g1$STAS$lat = stas$lat[m1]
       g1$STAS$lon = stas$lon[m1]
       g1$STAS$z = stas$z[m1]
       w1 = which(!is.na(g1$STAS$lat))
       sec = g1$STAS$sec[w1]
```

```
    N = length(sec)

    Ldat = list(name = g1$STAS$name[w1],
      sec = g1$STAS$sec[w1],
      phase = g1$STAS$phase[w1],
      lat = g1$STAS$lat[w1],
      lon = g1$STAS$lon[w1],
      z = g1$STAS$z[w1],
      err = g1$STAS$err[w1],
      yr = rep(g1$LOC$yr, times = N),
      jd = rep(g1$LOC$jd, times = N),
      mo = rep(g1$LOC$mo, times = N),
      dom = rep(g1$LOC$dom, times = N),
      hr = rep(g1$LOC$hr, times = N),
      mi = rep(g1$LOC$mi, times = N))

    Ldat$err[Ldat$err <= 0] = 0.05
    Ksta = length(unique(Ldat$name))
   ###  cat(paste("#################      ", i, Ksta), sep = "\n")
    Ldat = LeftjustTime(Ldat)

    KF[[i]] = Ldat

  }
```

2.3.3 Travel Times

The raytracing programs and travel time calculations in **REIS** use slowness in calculating the raypaths.

For travel time calculations alone, pass the velocity structure in terms of velocity, not slowness.

```
fsta = "/Users/lees/Site/CHAC/staLLZ.txt"
stas = scan(file=fsta,what=list(name="", lat=0, lon=0, z=0))
stas$z = stas$z/1000
library(Rquake)
EL=list()
EL$lon=c(-78.1793410885)
EL$lat=c(-0.393611681711)
EL$depth=8.4
data(wu_coso.vel)
```

```
vel = wu_coso.vel
slness = 1/vel$vp
DZ = distaz(EL$lat, EL$lon, stas$lat, stas$lon)
Mray = many.time1D(DZ$dist, EL$depth, stas$z, length(vel$zp), vel$zp, vel$vp)
arrs = Mray$tt

###  dump data to file
```

For extracting raypaths, on the other hand, the proper way is to pass the slowness vector to the calling program:

```
rays = Ray.time1D(DZ$dist[1], EL$depth, stas$z, length(vel$zp), vel$zp, vel$vp)
plot(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod],type="n", xlab="distance, km" , yla
     abline(h=-vel$zp, lty=2, col=grey(0.80) )
     points(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod], pch=8, col='green')
     lines(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod])
     points(rays$rnod[rays$nnod] , -rays$znod[rays$nnod], pch=6, col='red', cex=2)
```

pdf
  2

```
rays = Ray.time1D(200, 0, 0, length(vel$zp), vel$zp, vel$vp)
plot(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod],type="n", xlab="distance, km" , yla
     abline(h=-vel$zp, lty=2, col=grey(0.80) )
     points(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod], pch=8, col='green')
     lines(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod])
     points(rays$rnod[rays$nnod] , -rays$znod[rays$nnod], pch=6, col='red', cex=2)
```

pdf
  2

Make a 3D plot of the ray paths using the RGL package.

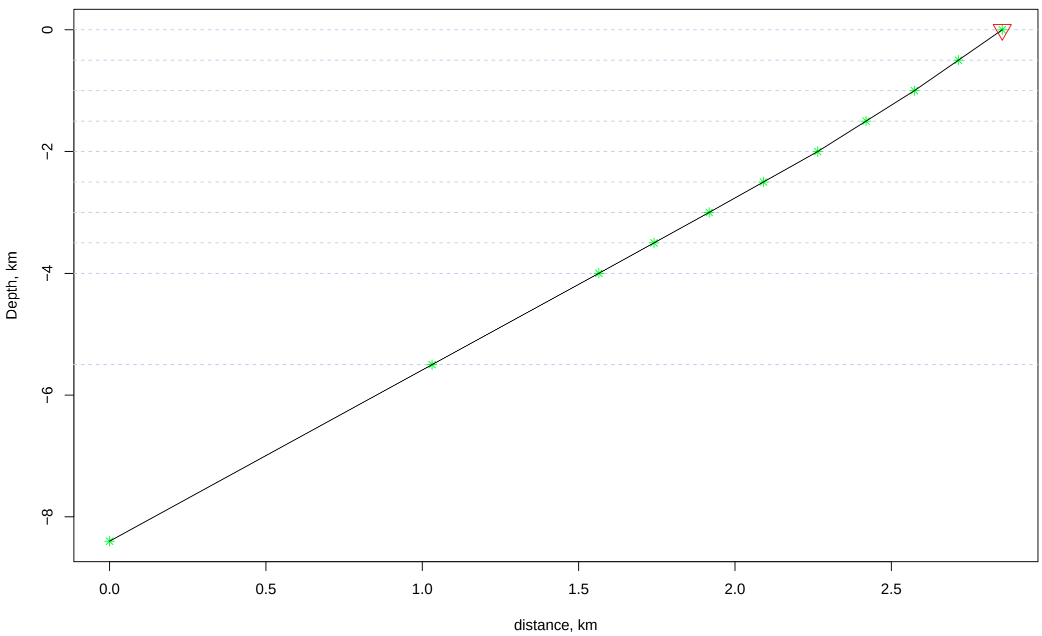

Figure 2: Raytracings with

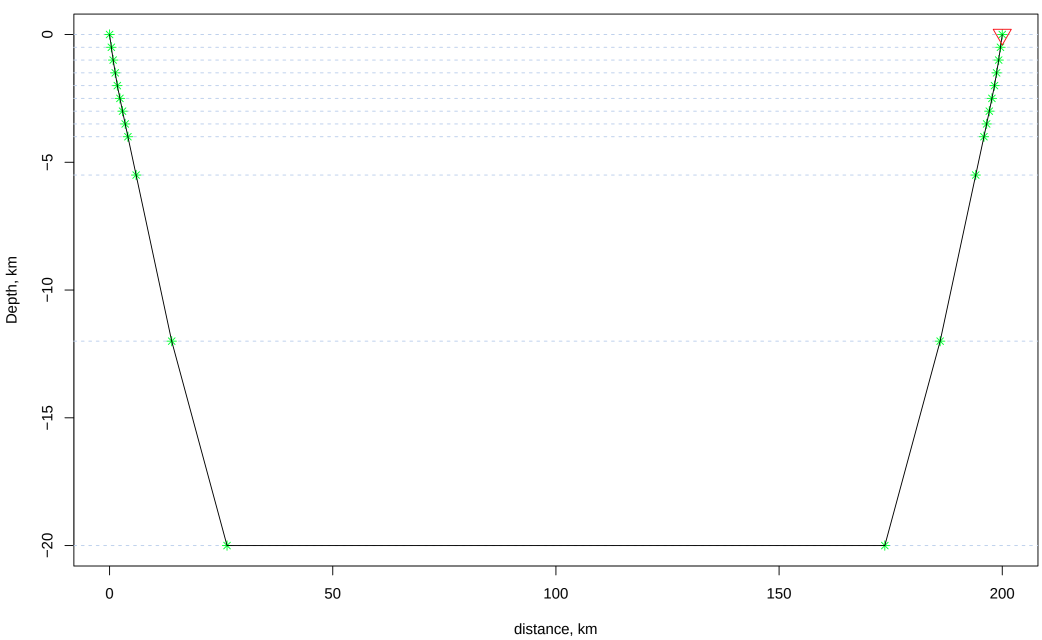

Figure 3: Raytracing done with long offset. Diffraction is returned.

```
proj=setPROJ(2, LAT0=EL$lat, LON0=EL$lon)
SXY = GLOB.XY(stas$lat, stas$lon, proj)
dee = sqrt( SXY$x^2 + SXY$y^2)
az = atan2(SXY$y, SXY$x)
rex = range(c(SXY$x, 0) )
rey = range(c(SXY$y, 0) )
rgl.open()
view3d( theta = 0, phi = -75)
###  put a plane at the top
 quads3d(c(rex[1],rex[2], rex[2],rex[1]),
         y = c(rey[1], rey[1], rey[2], rey[2])
         , z = rep(0, 4) , col='black', alpha=0.1 )
if(FALSE)
  {
for(i in 2:10 )
  {

    quads3d(c(rex[1],rex[2], rex[2],rex[1]),
         y = c(rey[1], rey[1], rey[2], rey[2])
         , z = rep(-vel$zp[i], 4) , col=rgb(0.7,1,0.7) , alpha=0.1 )
  }
}
for(i in 1:length(DZ$dist))
  {
    rays = Ray.time1D(dee[i], EL$depth, 0, length(vel$zp), vel$zp, vel$vp)

    zx=cos(az[i])*rays$rnod[1:rays$nnod]
    zy=sin(az[i])*rays$rnod[1:rays$nnod]
    zz = -rays$znod[1:rays$nnod]

     lines3d(cbind(zx, zy, zz) , col='red' )
    ####  color the stations
    points3d(cbind(zx[rays$nnod] ,zy[rays$nnod],zz[rays$nnod]), col=rgb(.6,.6,1))

  }
rgl.postscript(file="/Users/lees/Mss/SEIS_BOOK/RayTrace/FIGS/raypath3d.eps" , fmt="eps",
rgl.close()
```

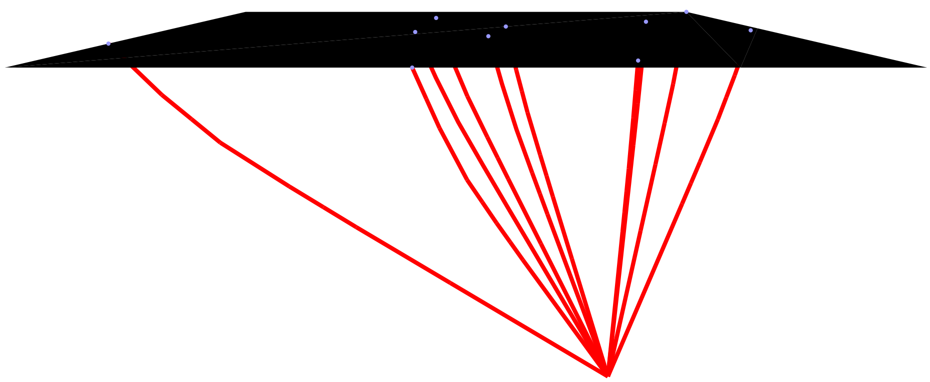

Figure 4: Shot with associated raypaths

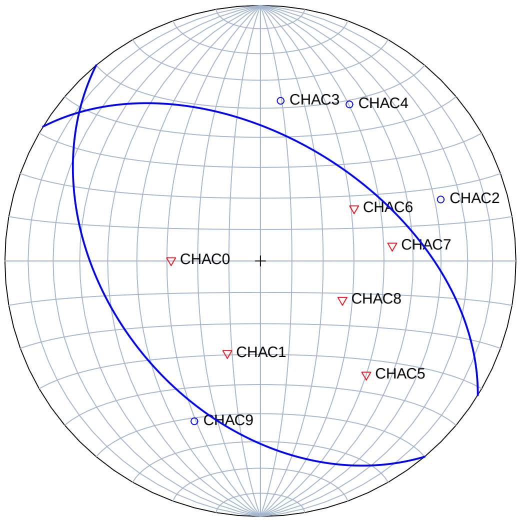

Figure 5: Focal sphere with raypath incident angles and possible focal solution shown as 2 perpendicular planes.

Once the earthquake hypocenter is known, the points where the rays exit the focal sphere can be determined and plotted.

In the next example I show how to plot the arrivals on the focal sphere.

```
sdr1 = c(140, 41, -76 )
a1 = SDRfoc(sdr1[1], sdr1[2], sdr1[3], PLOT=FALSE)
pol = rep("U", times=length(stas$name))
pol[match(c("CHAC2", "CHAC3", "CHAC4", "CHAC9"), stas$name) ] = "D"
```

```
AZ =  DZ$az
ang = Mray$angle
library(RFOC)
pch = rep(25, times=length(AZ))
pch[pol=="D"] = 21
col = rep('red', times=length(AZ))
col[pol=="D"] = 'blue'
```