Extract Regional/Local Seismic Data

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

January 14, 2020

Say someone notified you that there was some felt ground shaking experienced in the heighborhood. How can you quickly extract seismic data from IRIS to show data recorded in the vicinity?

There are several steps:

- identify the location of the observation: need Lat-Lon-Z
- identify the local time of the event
- convert local time to GMT time
- extract from IRIS the local available stations
- edit the list of stations (this may take some interaction)
- determine if just vertical, or 3 components are required
- (for a quick look, just use vertical)
- Find the closest high-quality station
- get the first arrival time.
- check catalog for events recorded
- attempt to determine source location
- order traces using distance
- plot seismic record to observe moveout
- locate event?
- filter seismic records to enhance signals

1 Initial Set Up

First you must start **R** and invoke the necessary libraries. These libraries have a variety of functions that help facilitate the analysis.

```
> library(IRISSeismic)
> library(RSEIS)
> library(GEOmap)
> library(RPMG)
>  ###   library(rFDSN) #This is on CRAN, maybe do not need this anymore
>  library(lubridate)
>  library(maps)
```

```
>  library(maptools)
> library(lubridate)
> ########  these are extra codes specific for this particular event:
>
> ## source("~/Site/NC_TA/CODE/makeNCmap.R")
>
> source("~/Site/NC_TA/CODE/NCmap.R")
> source("~/Site/TA_DATA/CODE/stream2GH1.R")
> source("~/Progs/R_stuff/deblank.R")
> source("~/Vignettes/IRIS/CODE/CHANchooser.R")
>
>
>
```

Next set up the location and the timing.

In this example I know in advance that there is a quarry near by and I expect the signal is from a quarry blast:

In many cases you can get a first guess from google.earth or other online maps. You only really need the Lat-Lon of a starting place.

```
> ### examples:
> pratson.quarry = list(lat=35+55/60+57.27/3600, lon=-(79+8/60+56.58/3600) , z=0)
> pratson.quarry$latitude = pratson.quarry$lat
> pratson.quarry$longitude = pratson.quarry$lon
> pratson.quarry$notes = 'pratson.quarry'
> Chapel_Hill=list()
> Chapel_Hill$lat=c(35.927613)
> Chapel_Hill$lon=c(280.959373)
> pratson.home = list(lat=0, lon=0, z=0)
> pratson.home$lat = 35.94455
> pratson.home$lon = -79.140308333333
> pratson.home$z = 0.2
>
```

Next you need a time to start looking. All seismic data is referred back to UTM or GMT time. So if your time is recorded in local time-zone, you must convert back to GMT for extracting the data. Once your data is in GMT zone, if you want to report anything, convert back to local time for communicating to the press or neighbors.

In this case Pratson reported feeling an event at 3 PM on Nov. 4, 2019. The suspect quarry is: American Stone Company.

```
> ####  note time of felt event:
> LocalTime<-list(
+ Year = 2019,
+ Month = 11,
+ Day = 4,
+     Hour = 12+3,
+     Minute=0,
+     Second = 0,
+     Place='NC',
+     Zone='EST'
+ )
> ##### convert to GMT
> L = LocalTime
> dat1 = paste(formatC(L$Year, width = 4), formatC(L$Month,
+         width = 2, flag = 0), formatC(L$Day, width = 2, flag = 0),
+     sep = "-")
>  tim1 = paste(formatC(L$Hour, width = 2, flag = 0), formatC(L$Minute,
+         width = 2, flag = 0), formatC(L$Second, width = 2, flag = 0),
+         sep = ":")
> pb.txt = paste(dat1, tim1, sep=' ')
> x <- lubridate::ymd_hms(pb.txt, tz = "America/New_York")
> T.GMT =  lubridate::with_tz(x, "GMT")
> ##  Note:
```

```
> tdif = T.GMT - x
> ### should be zero, its the same time, just different zones
>
> #############  set before and after to form a time-window for extracting data
>
> #### these are in seconds:
> T.BEF = 5*60
> T.AFT= 1800
> starttime = T.GMT-T.BEF
> endtime =  T.GMT+T.AFT
>
>
```

2 Query IRIS for Data

Set up an event location and a radius (in degrees) for searching the IRIS data base.

At first we will extract everything in a small radius and then we will narrow down by editing the available stations:

```
> e = pratson.quarry
> e$rad = 3
> iris <- new("IrisClient")
>   SELstations <- IRISSeismic::getStation(iris,network="*",station="*",location="*",channel="*",
+                                          starttime=starttime,
+                                          endtime=endtime,
+                                          lat=e$latitude,long=e$longitude,maxradius=e$rad )
> sta.info1 = names(SELstations)
> NSTA = length(SELstations$station)
>
>
```

Or, we could get a more extensive data base:

```
> if(FALSE)
+     {
+         AVAIL = vector(mode='list')
+
+         for(i in 1:length(SELstations$station) )
+             {
+
+                 h= SELstations[i,]
+
+                 channelH <- getAvailability(iris,h$network,h$station,"*","",starttime,endtime,
+                                             lat=e$latitude,long=e$longitude,maxradius=5)
+
+                 PH = paste(h$station, paste(channelH$channel, collapse=' ')  )
+                 ## print(PH)
+                 AVAIL[[i]] = channelH
+             }
+
+         names(AVAIL)<-SELstations$station
+
+
+ ######  this may take a few minutes
+ ###  could save this to a file for later use, so you do not have to repeat
+ save(file='AVAIL.RDATA', AVAIL)
+ }
> load('AVAIL.RDATA')
> iw = which(names(AVAIL) == 'V58A')
> iris <- new('IrisClient')
```

```
> GOOD.Z = vector(mode='list')
> m = 0
> for(i in 1:length(AVAIL))
+ {
+     h =AVAIL[[i]]
+     iz = Vertz(h)
+     cat(paste(i, iz), sep='\n')
+     if(!is.na(iz[1]) )
+         {
+
+             hj = GETirisZ(h, iz[1], starttime, endtime , iris )
+             if(is.na(hj) ) next
+             m = m +1
+             GOOD.Z[[m]] =  hj
+         }
+ }
>
```

Here we need to make a decision on what we want to see. We could restrict our data selection to a much more limited data set, at least for first inspection, and then later go back and get more data. When you have large data sets, manipulation and editting becomes cumbersome. Judicious editting is useful.

Let's say we only want stations that are in the state of North Carolina and have sample rates of 100 samp/s (probably good stations, useful for recording local events).

```
> ####   get stations in North Carolina
>  g1 = grep('NC', SELstations$sitename)
> ### search through these and extract only ones that have 3 Components and high sample rate.
> if(FALSE)
+     {
+
+
+         ##### this is a quarry blast -
+    #####      the first station to record was V58A in pittsboro (I think)
+         w1 = which(names(AVAIL)=='V58A')
+         h = AVAIL[[w1[1]]]
+         sta = 'V58A'
+         Achan = 'HH*'
+         net =  'N4'
+         #### this should give us 3 channels
+
+         V.ALL = vector(mode='list')
+
+         i.record = which(h$channel=='HH1')
+         Achan = h$channel[i.record]
+         net =   h$network[i.record]
+         sta = h$station[i.record]
+         scale = h$scale[i.record]
+
+
+         V58.1  <- getDataselect(iris, net,sta,"00",'HH1',starttime, endtime,
+                                              inclusiveEnd=FALSE,ignoreEpoch=TRUE)
+
+         H1 = DECON::stream2GHnosens(V58.1, DEST='.',
+                                 STREAM=TRUE, sensitivity = 1, scalefactor = 1/scale, gain = 1 )
+                           attr(H1,'channel')<-h[i.record,]
+
+         V.ALL[[1]] <- H1
+
+
+         i.record = which(h$channel=='HH2')
+         Achan = h$channel[i.record]
```

```
+         net =   h$network[i.record]
+         sta = h$station[i.record]
+         scale = h$scale[i.record]
+
+
+          V58.2  <- getDataselect(iris, net,sta,"00",'HH2',starttime, endtime,
+                                              inclusiveEnd=FALSE,ignoreEpoch=TRUE)
+         H1 = DECON::stream2GHnosens(V58.2, DEST='.',
+                              STREAM=TRUE, sensitivity = 1, scalefactor = 1/scale, gain = 1 )
+                        attr(H1,'channel')<-h[i.record,]
+
+
+         V.ALL[[2]] <- H1
+
+
+
+           i.record = which(h$channel=='HHZ')
+         Achan = h$channel[i.record]
+         net =   h$network[i.record]
+         sta = h$station[i.record]
+         scale = h$scale[i.record]
+
+
+         V58.Z  <- getDataselect(iris, net,sta,"00",'HHZ',starttime, endtime,
+                                              inclusiveEnd=FALSE,ignoreEpoch=TRUE)
+
+          H1 = DECON::stream2GHnosens(V58.Z, DEST='.',
+                              STREAM=TRUE, sensitivity = 1, scalefactor = 1/scale, gain = 1 )
+                        attr(H1,'channel')<-h[i.record,]
+
+
+         V.ALL[[3]] <- H1
+
+                i.record = which(h$channel=='BDF')
+         Achan = h$channel[i.record]
+         net =   h$network[i.record]
+         sta = h$station[i.record]
+         scale = h$scale[i.record]
+
+
+         V58.F  <- getDataselect(iris, net,sta,"*",Achan,starttime, endtime,
+                                              inclusiveEnd=FALSE,ignoreEpoch=TRUE)
+
+          H1 = DECON::stream2GHnosens(V58.F, DEST='.',
+                              STREAM=TRUE, sensitivity = 1, scalefactor = 1/scale, gain = 1 )
+                        attr(H1,'channel')<-h[i.record,]
+
+         V.ALL[[4]] <- H1
+
+       save(file='V.ALL.RDATA', V.ALL)
+
+
+         load('V.ALL.RDATA', verbose=TRUE)
+
+         V58.rs =  prepSEIS(V.ALL )
+
+          V58.rs$units =c('m/s', 'm/s','m/s','Pa')
+
+
+         V58.rs$units
```

```
+          swig(V58.rs, YAX=2, TIT = "Americn Stone Co.: Quarry Blast")
+
+
+         ###   RPMG::jpng(file='QuarryBlast_V58A.png', width=10, height=6)
+
+         RSEIS::swig(V58.rs, WIN=c(165, 225), SHOWONLY=TRUE, YAX =2,
+               TIT = "GMT",
+              prefilt=list(fl=1/2, fh=8,  type="BP", proto="BU")
+              )
+         title(main="Americn Stone Co.: Quarry Blast, 15.7 km", sub='Unfiltered' )
+
+
+
+        F58 =  FILT.SEISN(V58.rs, sel = 1:length(V58.rs$JSTR),
+      FILT = list(ON = TRUE, fl = 1/2, fh = 8, type = "BP",
+       proto = "BU", RM=FALSE, zp=TRUE ), TAPER = 0.1, POSTTAPER = 0.1)
+
+         F58$COMPS = c("N", "E", "V", "F")
+         F58$KNOTES = c("N", "E", "V", "F")
+          F58$units =c('m/s', 'm/s','m/s','Pa')
+
+         ################################  plot 4 components
+        ##    F58$KNOTES  =  F58$units
+
+         source("~/Vignettes/IRIS/CODE/PLOT.SEISN2.R")
+         source("~/Vignettes/IRIS/CODE/Yaxis.expo.R")
+
+
+
+         RPMG::jpng(file='QuarryBlast_V58A.png', width=10, height=7)
+         #
+
+         PLOT.SEISN2(F58,
+                    dt = F58$info$dt,
+                    sel =c(1:4) ,
+                    WIN =c(165, 225) ,
+                    labs=F58$units ,
+                    notes = F58$KNOTES ,
+                    subnotes=NA,
+                    tags ='' ,
+                    sfact = 1,
+                    LOG = "",
+                    COL = c('black', 'black','black', 'blue') ,
+                    add = 1,
+                    pts = FALSE,
+                    YAX = 2,
+                    TIT = "GMT",
+                    SHIFT = NULL,
+                    COLLAPSE=FALSE,
+                    rm.mean = TRUE,
+                    UNITS = F58$units,
+                    MARK = TRUE,
+                    xtickfactor = 1)
+         title(main="Americn Stone Co.: Quarry Blast at V58A, 15.7 km",
+       sub='Filtered: BP 0.5-8 Hz')
+
+         dev.off()
+
+
+  #####   alternative: scale each trace prior to plotting
```

```
+
+
+
+
+         for(i in 1:3 )
+             {
+                 F58$JSTR[[i]] = F58$JSTR[[i]]*1e06
+             }
+
+
+         Myunits  =  c(paste('\\mu', 'm/s', sep=' '),
+                       paste('\\mu', 'm/s', sep=' '),
+                       paste('\\mu', 'm/s', sep=' '), 'PA' )
+
+
+           RPMG::jpng(file='QuarryBlast2_V58A.png', width=10, height=7)
+
+          PLOT.SEISN(F58,
+                     dt = F58$info$dt,
+                     sel =c(1:4) ,
+                     WIN =c(165, 225) ,
+                     labs=F58$units ,
+                     notes = F58$KNOTES ,
+                     subnotes=NA,
+                     tags ='' ,
+                     sfact = 1,
+                     LOG = "",
+                     COL = c('black', 'black','black', 'blue') ,
+                     add = 1,
+                     pts = FALSE,
+                     YAX = 2,
+                     TIT = "GMT",
+                     SHIFT = NULL,
+                     COLLAPSE=FALSE,
+                     rm.mean = TRUE,
+                     UNITS = Myunits,
+                     MARK = TRUE,
+                     xtickfactor = 1)
+         title(main="Americn Stone Co.: Quarry Blast at V58A, 15.7 km",
+       sub='Filtered: BP 0.5-8 Hz')
+          dev.off()
+
+
+           swig(F58, WIN=c(165, 180), SHOWONLY=TRUE)
+
+ GOOD.stations = vector(mode='list')
+ M = 0   ## counter
+
+ for(i in 1:length(g1) )
+     {
+
+         j = g1[i]
+
+         h = AVAIL[[j]]
+
+         ###  w = which(h$samplerate>=100 & h$samplerate < 200)
+         chan = h$channel
+         w.A1 = substr(chan, 1,1)
+         w.A2 = substr(chan, 2,2)
+         w.A3 = substr(chan, 3,3)
```

```
+
+         codez = (w.A1=='H' | w.A1=='B' )
+         instz = (w.A2=='H')
+         #  compz =
+         w = which( w.A1=='H' | w.A1=='B' )
+         nw = length(w)
+         if(nw>0)
+             {
+                 for(k in 1:nw )
+                     {
+
+                         i.record = w[k]
+                         Achan = h$channel[i.record]
+                         net =   h$network[i.record]
+                         sta = h$station[i.record]
+                         scale = h$scale[i.record]
+
+
+                         st3 <- getDataselect(iris, net,sta,"00",Achan,starttime, endtime,
+                                              inclusiveEnd=FALSE,ignoreEpoch=TRUE)
+                         H1 = DECON::stream2GHnosens(st3, DEST='.',
+                             STREAM=TRUE, sensitivity = 1, scalefactor = 1/scale, gain = 1 )
+                         attr(H1,'channel')<-h[i.record,]
+                         M = M +1
+                         GOOD.stations[[M]] = H1
+                     }
+
+             }
+     }
+
+
+
+
+ save(file='GOOD.stations.RDATA', GOOD.stations)
+
+ }
> load('GOOD.stations.RDATA')
> N = length(GOOD.stations)
>
>
```

The results of the last data retrieval come in a format that is common in RSEIS. It isimply a list of seismic traces with a small amount of header information. These traces can be combined into a slightly different structure that is more suitable for plotting on the same scale.

The purpose of this is collect the common information from the seismic traces (onset times, end times, station names, channels names) so that they can be plotted and manipulated by swig.

```
> GH = prepSEIS(GOOD.stations)
> GH = prepSEIS(GOOD.Z)
> swig(GH, SHOWONLY=TRUE)
>
>
```

Replot, this time with some kind of more sensible ordering:

```
> ### lat lon for each station
>
> stas = list(name=vector(mode='character'),
+     comp=vector(mode='character')   ,
+     lat=vector(mode='numeric'),
+     lon=vector(mode='numeric'),
```

```
+     z=vector(mode='numeric'))
> for(i in 1:length(GOOD.Z) )
+     {
+         p = GOOD.Z[[i]]
+         a.p = attr(p, "channel")
+
+         stas$name[i] = a.p$station
+         stas$lat[i] = a.p$latitude
+         stas$lon[i] = a.p$longitude
+         stas$z[i] = a.p$elevation/1000
+          stas$comp[i] = a.p$channel
+
+     }
> STA = list(LATITUDE=vector(mode='numeric'), LONGITUDE=vector(mode='numeric'),
+     ELEVATION=vector(mode='numeric'), CODE=vector(mode='character'),
+     NAME=vector(mode='character'), NETWORK=vector(mode='character'))
> for(i in 1:length(GOOD.Z) )
+     {
+         p = GOOD.Z[[i]]
+         a.p = attr(p, "channel")
+
+         STA$NAME[i] = a.p$station
+         STA$LATITUDE[i] = a.p$latitude
+         STA$LONGITUDE[i] = a.p$longitude
+         STA$ELEVATION[i] = a.p$elevation
+         STA$NETWORK[i] = a.p$net
+          STA$CODE[i] = a.p$channel
+     }
> dees = distaz(e$lat, e$lon, stas$lat, stas$lon)
> ### surface wave velocity = 3.6 km/s
> t.surf = dees$dist/3.6
> ORD1 = order(t.surf)
> swig(GH, sel=ORD1,  SHOWONLY=TRUE)
> if(FALSE)
+     {
+         #####  view one good trace - near the blast.
+
+         b = swig(GH, sel=which.min(t.surf) )
+
+ ##      pickV58A = Jtim( 308 , hr= 19 , mi= 57 ,sec= 47.4694802761078 )
+
+        pickV58A = b$WPX
+
+         save(file='pickV58A.RDATA', pickV58A)
+     }
>
```

Next, get the arival of the closest (or best recorded) station then subtract of the travel time to that station is an estimate of the source time.

```
> load(file='pickV58A.RDATA')
> ############  here get the arival of the closest (or best recorded) station
> #####   then subtract of the travel time to that station is an estimate of the source time
> ##   source.time = pickV58A
> source.time = pickV58A
> source.time$sec = source.time$sec - t.surf[ stas$name=='V58A']
> t.source = source.time
> Wground =setwpix(phase = 'G', col = 'brown', yr = t.source$yr, jd = t.source$jd,
+      hr = t.source$hr, mi = t.source$mi, sec = t.source$sec+t.surf[ORD1],
+     dur = 0, name = stas$name[ORD1] ,
+       comp ='V' , dispcomp =stas$comp[ORD1]  )
```

```
> ####  turn on all the predicted arrival picks:
> Wground$onoff = rep(1, length(Wground$onoff) )
> swig(GH, sel=ORD1 , APIX=Wground  )
>
>
>
>
>
```

Next consider filtering the signals - this can be done in swig, or outside of swig to get a specific filter applied to all the traces.

Use TELES package to plot a seismic section arranged by distance:

```
> library(TELES)
>  event=list(
+      yr=t.source$yr,
+      mo=t.source$mo,
+      dom=t.source$dom,
+      hr=t.source$hr,
+      mi=t.source$mi,
+      sec=t.source$sec    ,
+      lat=e$lat,
+      lon= e$lon,
+      mag= 1 ,
+      depth=e$z)
>      event$z =event$depth
>      event$jd = getjul(event$yr, event$mo, event$dom)
> for(i in 1:length(GOOD.Z) )
+     {
+         GOOD.Z[[i]]$lat = GOOD.Z[[i]]$coords$lat
+
+         GOOD.Z[[i]]$lon = GOOD.Z[[i]]$coords$lon
+                GOOD.Z[[i]]$z = stas$z[i]
+
+     }
>   FILT = list(ON = TRUE, fl = 1/2, fh = 3, type = "BP",
+       proto = "BU", RM=FALSE, zp=TRUE )
> FILT.Z = GOOD.Z
> for(i in 1:length(GOOD.Z) )
+     {
+
+        z = GOOD.Z[[i]]
+
+        zz1 = AUGMENTbutfilt(z$amp, 1/3, 2,  z$dt,    type='BP' ,  proto="BU", zp=TRUE, pct=0.2 )
+
+        FILT.Z[[i]]$amp = zz1
+
+     }
> dees = distaz(e$lat, e$lon, stas$lat, stas$lon)
> XMCOL=setXMCOL()
> kol = sample(XMCOL[2:length(XMCOL) ] , length(FILT.Z))
> kol[stas$name=='V58A' ] = 'black'
> V = seq(from=2, to=4, length=5)
>  APAL = GPAL()
> #########   plot here
> LIMS = PrepGSNdisplay(FILT.Z, 1:length(FILT.Z) , event, ylim=c(-10, 200) )
>    SurfaceWaves(V, col=APAL[1:length(V)] )
> tagXY = ADDTRACES(FILT.Z, 1:length(FILT.Z), event, col=kol )
> JTT = TELES::GET.TTCURVES(Z=0, DIS=tagXY$x )
> ADDttcurves(JTT, sel= 1:4  )
> ##  WHY.pos = rep(0, length=length(tagXY$x))
```

```
> ptop = par('usr')[4]
> WHY.pos = rep(ptop, length=length(tagXY$x))
> text(tagXY$x, WHY.pos , labels = stas$name, pos=2 , srt=-90, cex=0.8, xpd=TRUE)
> ##
> ## points(tagXY$x, tees$tt, col='black' , cex=1.4)
>
> if(FALSE)
+     {
+ v = 3.6
+ vdeg = 110/3.6
+ abline(0, vdeg )
+
+  ADDttcurves(TTCURVES, sel= WCURVES, col=TT.COL )
+
+
+
+ H = addSTAINFO(GOOD.Z, stas)
+
+ J = H
+
+      names(attributes(J))
+      stalat = attr(H, "stalat")
+      staname = attr(H, "staname")
+ }
>
>
```