WINRPACK: Convert WIN to 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) 962-0695

MARCH, 2008

Abstract

Convert WIN format data from Japan to R

1 Getting started

Win format data consists of 3 files: a pick file that has infomation about the arrival times and the phase arrival picks, and two files that store the waveform data. These are a channel file and a digital waveform file. The channel file is a meta-data file that has information on how the waveforms are stored, so they can be extracted by the C-code programs.

First read in the pickfile:

```
> library(WINRPACK)
> winpick = "970723.235151.840"
> zip = get1WINPICK(winpick)
```

This is a list structure that consists of a copy of the pickfile in WIN format and a break down of the pickfile after it has been parsed.

The structure is as follows:

```
> names(zip)
```

```
 [1] "PF"     "AC"     "LOC"    "MC"     "REFT"   "STA"    "LIP"    "E"
 [9] "infile" "winID1" "LEVEL"  "PICKER"
```

The documentation call will show how this data is stored in R, for example, help(get1WINPICK), which will show the list structure of the data collected by the conversion program.

```
    PF: original pick file
    AC: event card (a-card)
   LOC: location
    yr: year
    jd: julian day
    mo: month
   dom: day or month
    hr: hour
    mi: minute
   sec: sec
   lat: lat
   lon: lon
     z: depth (km)
   mag: magnitude
    MC: focal mechanism card
  REFT: Reference time for phase arrivals
    yr: year
    jd: julian day
    mo: month
   dom: day or month
    hr: hour
    mi: minute
   sec: sec
   STA: List of stations
   tag: id tag for station
  name: station name
  comp: component
    c3: other id tag
  ppol: polarity
  parr: p-aarival time relative to reference
  pflg: flag
  perr: p-error
  pres: p-residual
  sarr: s-arrival time relative to reference
  sflg: flag
  serr: s-error
  sres: s-residual
   LIP: error ellipse
     E: simple error card
infile: input file name
winID1: WIN ID number
 LEVEL: level
PICKER: name of picker
```

In WIN format the pcikfile contains information on where to extract the waveforms. To read in the waveforms one must first get information on the

channels stored in the binary file. On my computer the waveforms and the pickfiles are stored in different directories. This information must be indicated to the calling program, of course. The channel file contains information about the stations and their arrangement in the waveform file. For this example we have stored the pickfile in the local directory but typically these will be stored on disk in the a remote directory. The full path name to the pickfile and the waveform file directories must be supplied to the conversion programs.

Once we read in the pickfile, we extract the waveform ID from the pickfile and use that to find the correct associated waveform file.

As an example, we next read in the channel file to set the LAT-LON coordinates of the stations recorded on the waveform file. The program readwinch is later called again from the program XWINdata, so this step can often be skipped.

```
> fn = zip$winID1
> chans = readwinch(fn)
> m1 = match(zip$STA$name, chans$sta)
> zip$STA$lat = chans$lat[m1]
> zip$STA$lon = chans$lon[m1]
> zip$STA$z = chans$z[m1]
```

The next program actually reads in the channel file and then extracts the infromation from waveform data file.

```
> JH = XWINdata(fn, stasel = NULL, PLOT = FALSE)
> L = length(JH)
```

The argument stasel allows one to select only a subset of traces from the WIN data file.

The data is now in memory and it can be used for plotting and other analysis. For example, suppose I want to view one of the traces in the structure.

The structure JH has 80 elements or 80 traces. Lets consider the first one. To see the meta data structure, we print the names of the list:

```
> names(JH[[1]])

[1] "fn"     "sta"    "comp"   "dt"     "DATTIM" "N"      "units"  "amp"
```

We see that the structure has the meta data and the trace data (the waveform) in it. To plot the first trace, we can very simply try:

```
> y = JH[[1]]$amp
> x = seq(from = 0, by = JH[[1]]$dt, length = length(y))

> plot(x, y, type = "l", xlab = "sec", ylab = "amplitude", main = paste(JH[[1]]$sta,
+     JH[[1]]$comp, sep = " "))
```

The user can make a function to plot part or all of the traces or do other analysis.

More commonly the pickfiles may be stored in some remote directory, for example:

```
pickdir = '/data/gauss/sake/Fuji/Picks/2001'
wigdir = '/data/gauss/sake/Fuji/Wigs/2001'
```

The waveforms are further broken down into directories by month:

```
% ls /data/gauss/sake/Fuji/Wigs/2001
01  02  03  04  05  06  07  08  09  10  11  12
```

So to extract the correct file name associated with a pickfile, we must make sure we are pointing to the correct directory. The zip list has this information in it:

```
> wigdir = paste(sep = "/", "/data/gauss/sake/Fuji/Wigs", zip$LOC$yr,
+     formatC(zip$LOC$mo, width = 2, flag = "0"))
> testfn = paste(sep = "/", wigdir, zip$winID1)
```

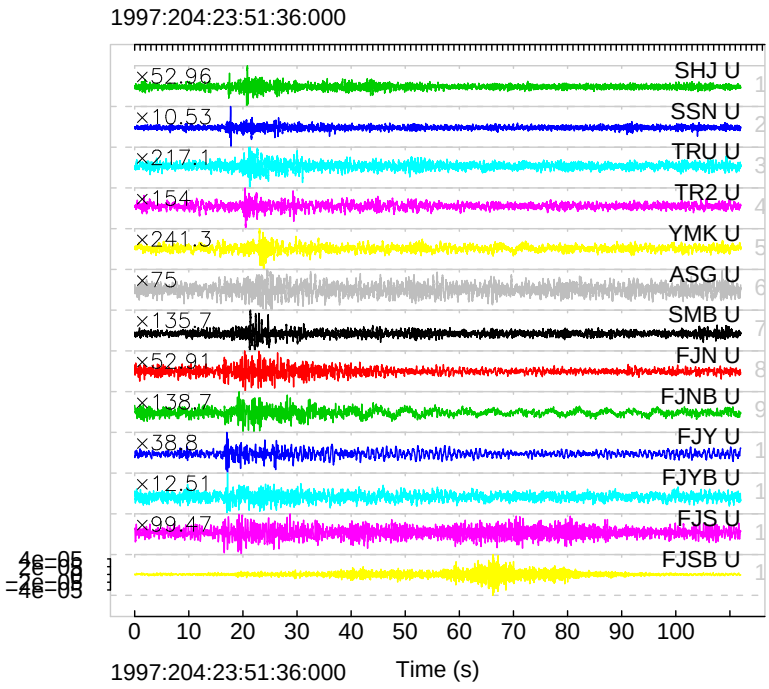
which can be used to replace the file name "fn" in the examples above.

To use the RSEIS software, first convert this structure with a program to reformat the list into a seismic structure:

```
> require(RSEIS)
> KH = prepSEIS(JH)
```

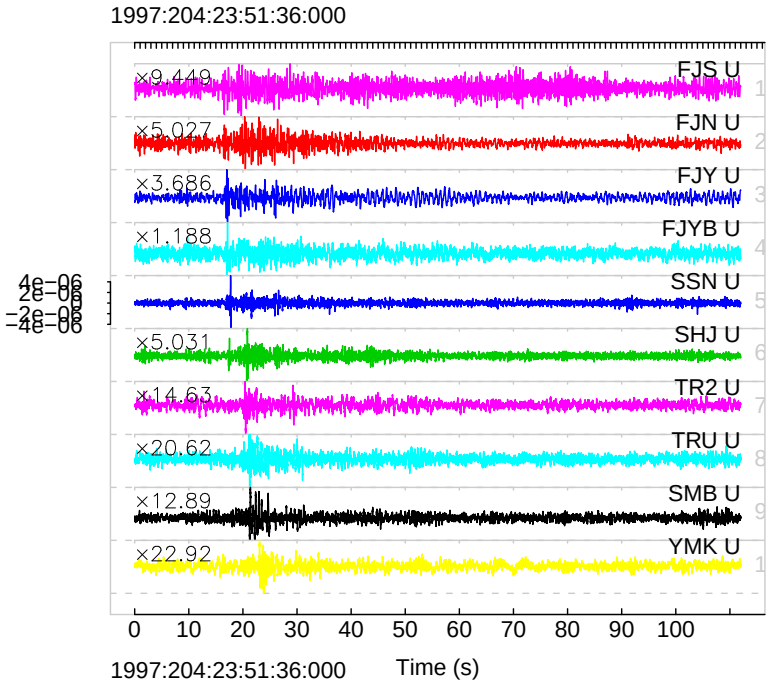
All this does is reorganize the data into a structure used by this package. Now the data can be plotted and analyzed with all the power in the RSEIS package.

```
> w1 = which(KH$COMPS == "V")
> PICK.GEN(KH, sel = w1, SHOWONLY = TRUE)
```



Not all the traces have associated phase picks. We can select off the ones that do and plot them according to arrival time. To sort these according to the times of the first arrivals one may do something like this:

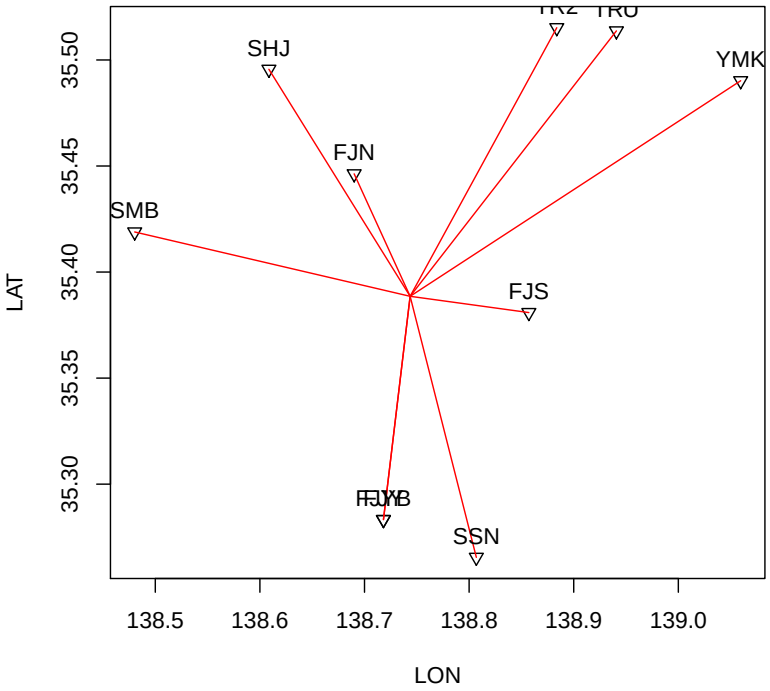
```
> match(zip$STA$name, KH$STNS[w1])
> w2 = w1[match(zip$STA$name, KH$STNS[w1])]
> B = order(zip$STA$parr)
> PICK.GEN(KH, sel = w2[B], SHOWONLY = TRUE)
```



Finally, to analyze the data using the full power of RSEIS, remove the SHOWONLY argument (or set to false) so that the program enters interactive mode. Buttons can be accessed this way, but I recommend looking over the documentation of RSEIS before proceeding.

To see a plot of the station configuration that recorded the event:

```
> plot(zip$STA$lon, zip$STA$lat, pch = 6, xlab = "LON", ylab = "LAT",
+     main = zip$winID1)
> segments(zip$STA$lon, zip$STA$lat, zip$LOC$lon, zip$LOC$lat,
+     col = "red")
> text(zip$STA$lon, zip$STA$lat, labels = zip$STA$name, pos = 3)
```


970723.235156
SHJ
YMK
FJN
SMB
FJS
SSN
35.50
35.45
35.40
35.35
35.30
LAT
138.5
138.6
138.7
138.8
138.9
139.0
LON