# General preprocessing with SAC and .tcsh

Rob Porritt, G. G.
University of Arizona

Goal:

Make record section plots for 3 components of motion

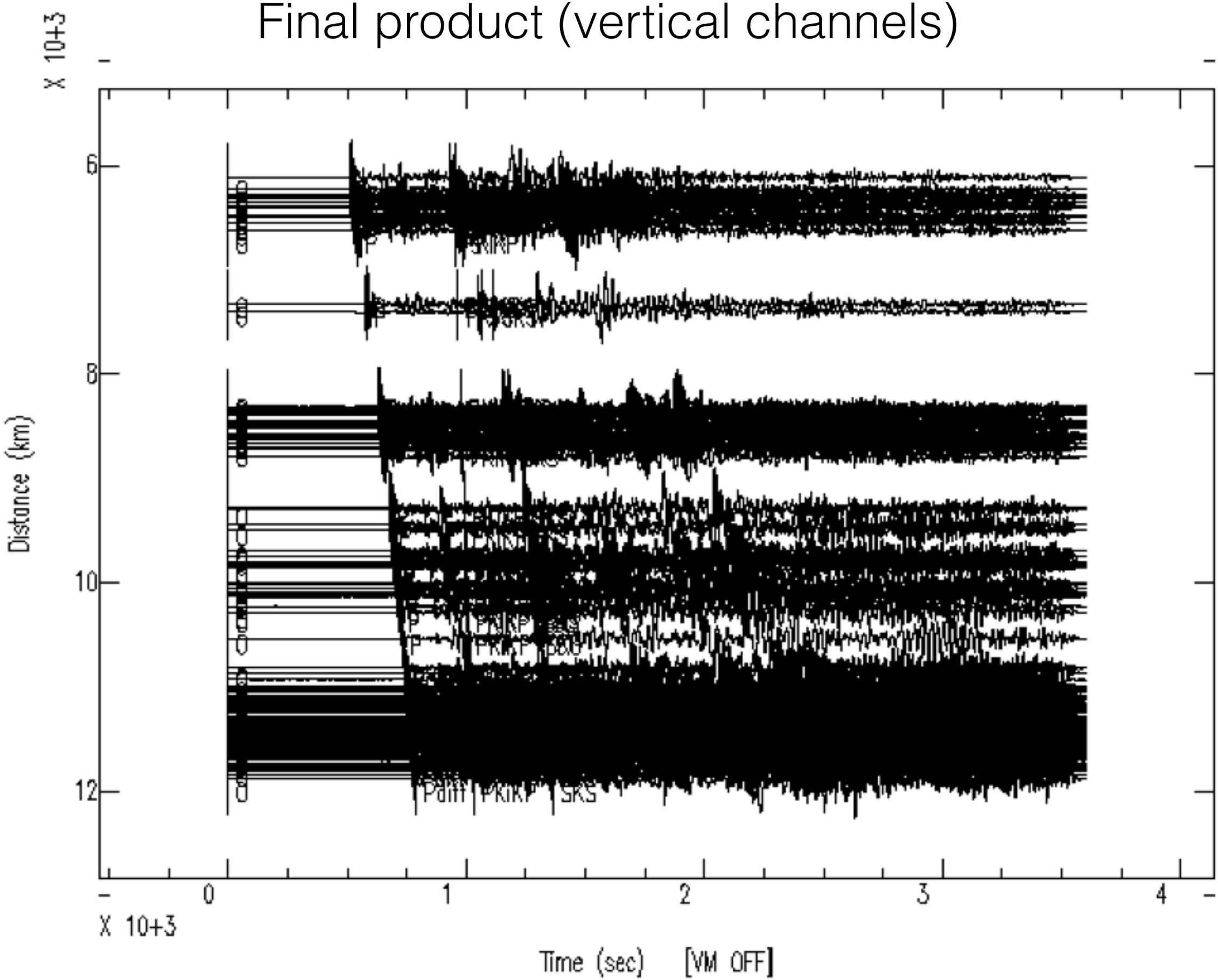
Starting point:

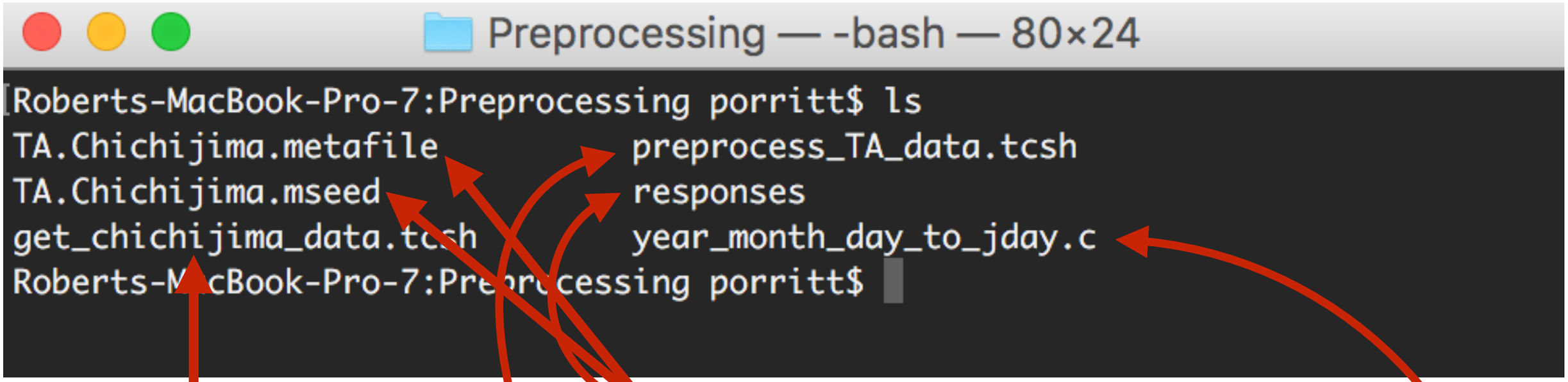
1. Miniseed data starting at event origin time with 3 component broadband data
2. Metadata file associated with the miniseed data
3. RESP files associated with waveform data

Necessary software:

c compiler (gcc and/or clang)
sac - also needs X window system, such as XQuartz
mseed2sac
evalresp (comes with sac)
taup

# Final product (vertical channels)






shell script to pull data from iris with perl fetch scripts

metadata file, miniseed file, and responses directory from shell script

Heavily commented shell script to convert from miniseed to record sections

c source code to calculate the day of the year from year, month, and day of month

# Dissecting the script

```
echo "Calling sac for conversion to ground motion and adding event information to header"

sac << EOF
cuterr fillz
cut on b 0 3600
read *SAC
sync
rmean
rtrend
taper
cd responses
transfer from evalresp to vel freqlimits $f1 $f2 $f3 $f4
cd ../
ch o GMT $OriginYear $OriginJDay $OriginHour $OriginMinute $OriginSecond
ch mag $Magnitude evlo $EventLongitude evla $EventLatitude evdp $EventDepth
write over
quit
EOF
```

window the data from the origin to 1 hour later

read all sac waveform data and synchronize them

remove 0 frequency offsets and compensate for digitization

remove the instrument response

update the headers

# Dissecting the script

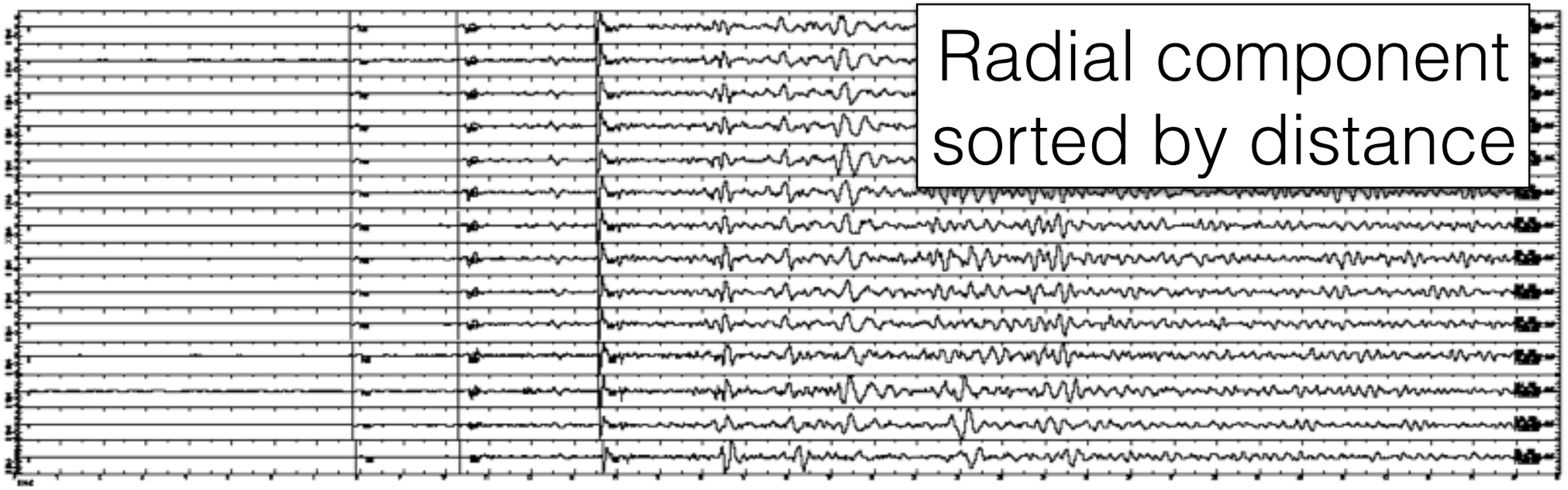
```
echo "Rotating sac files to radial and transverse"
foreach station (`ls *.SAC | awk -F. '{print $2}' | sort | uniq`)

# Calls sac for rotation of this event
# the sac command rotate to gcp uses the cmpaz header information and the event location to rotate into standard radial and transverse coordinate frames. The write command used here writes new files; one for each file in memory and therefore we can control the names of the output files
# After rotating and writing, the script re-reads the new files and changes the header variable "kcmpnm" which is the name of the channel
echo "Rotating $station to great circle path"

sac << EOF
read *.${station}.*.BHE.*.SAC *.${station}.*.BHN.*.SAC
rotate to gcp
write TA.${station}.BHR.SAC TA.${station}.BHT.SAC
read TA.${station}.BHR.SAC
ch kcmpnm BHR
write over
read TA.${station}.BHT.SAC
ch kcmpnm BHT
write over
quit
EOF

# Move the waveforms into the event directory with our naming convention
mv TA.${station}.BHR.SAC TA.${station}.BHT.SAC $evtdir
mv *.${station}.*.BHZ.*.SAC ${evtdir}/TA.${station}.BHZ.SAC
rm *.${station}.*BHN.*.SAC *.${station}.*.BHE.*.SAC
echo "Finished with $station into $evtdir"

end # End loop over each station
```

Read East and North files
rotate to great circle path
Write rotated files
Update headers

# Dissecting the script

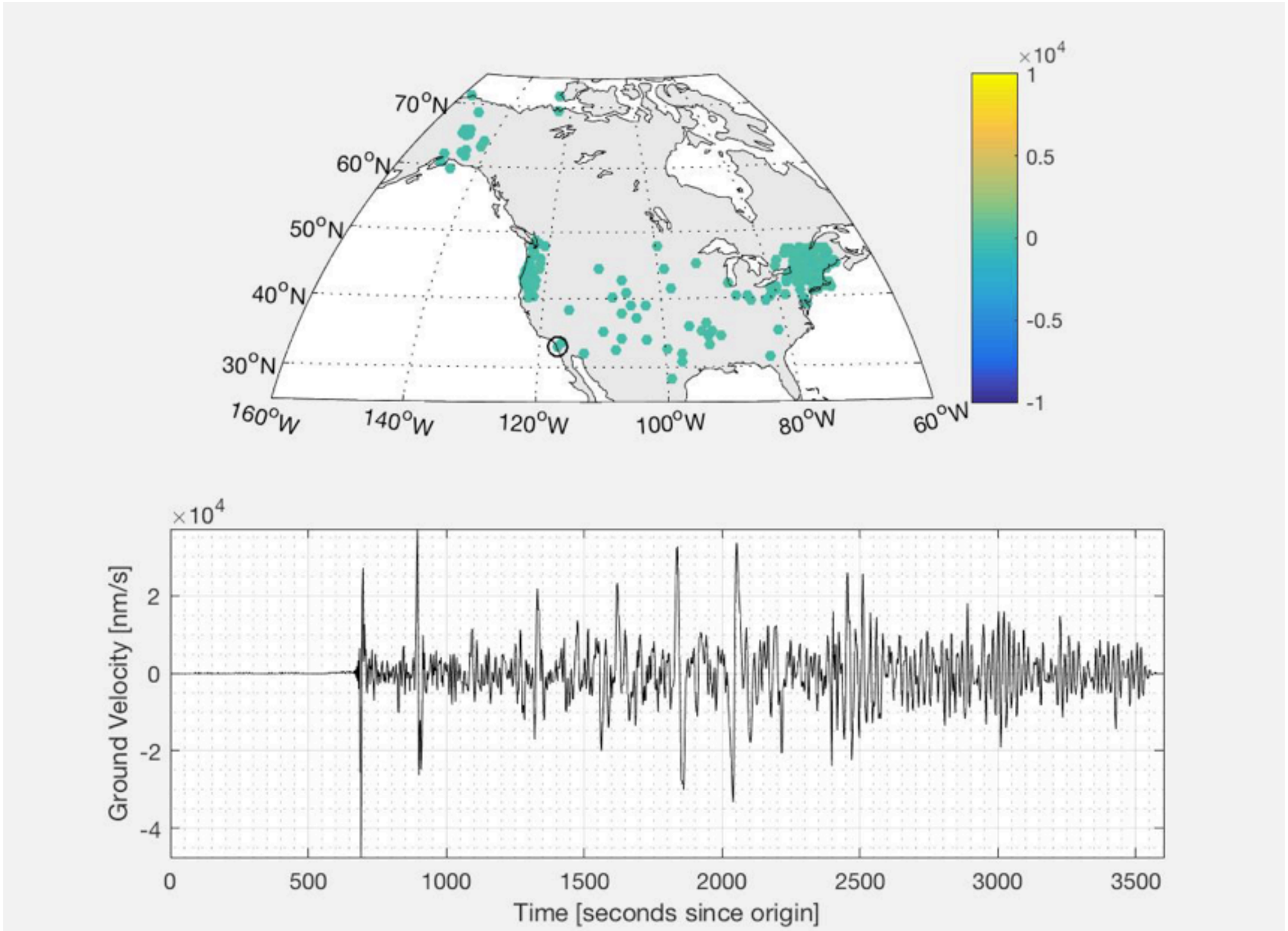
```
# Finally, lets move to the event directory and make a plot of the waveforms
# Uses the signal stacking subprocess to make a record section and then the unix "sleep" command to wait
10 seconds. This command also uses the sac filter command (bp bu co 0.01 0.1) to do a bandpass (bp)
butterworth (bu) filter with corners (co) at 100 (0.01) seconds and 10 (0.1) seconds.
cd $evtdir
sac << EOF
qdp off
read *BHZ*SAC
bp bu co 0.01 0.1
sss
prs ref off labels off
sleep 10
quitsub
read *BHR*SAC
bp bu co 0.01 0.1
sss
prs ref off labels off
sleep 10
quitsub
read *BHT*SAC
bp bu co 0.01 0.1
sss
prs ref off labels off
sleep 10
quit
EOF
```

For each component,
turn off quick and dirty plotting
read data
filter between 10 and 100 seconds period
enter signal stacking subprocess (sss)
plot record section
sleep for 10 seconds with plot up


Radial component
sorted by distance

Ok, cool, so we've got waveforms, now what?

# TA Ground motion movie



Preprocessing — -bash — 98×15

```
guest-wireless-upc-1606-10-120-106-164:Preprocessing porritt$ ls
TA.Chichijima.metafile          make_chichijima_movie.m         year_month_day_to_jday
TA.Chichijima.mseed             preprocess_TA_data.tcsh         year_month_day_to_jday.c
decimate_sac_data.tcsh          responses
get_chichijima_data.tcsh        sac_list_to_4D
guest-wireless-upc-1606-10-120-106-164:Preprocessing porritt$
```

Script to decimate by factor of 100 via sac

Matlab script to create the movie

Directory with c source and makefile to convert from sac files to ascii

# Step 1: Compile sac_list_to_4D

sac_list_to_4D — -bash — 98×34

```
guest-wireless-upc-1606-10-120-106-164:Preprocessing porritt$ ls
TA.Chichijima.metafile          make_chichijima_movie.m         year_month_day_to_jday
TA.Chichijima.mseed             preprocess_TA_data.tcsh         year_month_day_to_jday.c
decimate_sac_data.tcsh          responses
get_chichijima_data.tcsh        sac_list_to_4D
guest-wireless-upc-1606-10-120-106-164:Preprocessing porritt$ cd sac_list_to_4D/
guest-wireless-upc-1606-10-120-106-164:sac_list_to_4D porritt$ ls
Makefile                sac_list_to_4D          sac_list_to_4D.c
guest-wireless-upc-1606-10-120-106-164:sac_list_to_4D porritt$ cat Makefile
CC = clang
FLAGS = -Wall -O3 -lm
SACLIBS = `sac-config -c -l sacio sac`
BIN = sac_list_to_4D
SRC = sac_list_to_4D.c
INSTALL_DIR = ~/bin


all :: $(BIN)


$(BIN) ::
        $(CC) $(SRC) $(FLAGS) $(SACLIBS) -o $(BIN)

install ::
        install -s $(BIN) $(INSTALL_DIR)

clean ::
        rm $(BIN)

guest-wireless-upc-1606-10-120-106-164:sac_list_to_4D porritt$ make
clang sac_list_to_4D.c -Wall -O3 -lm `sac-config -c -l sacio sac` -o sac_list_to_4D
guest-wireless-upc-1606-10-120-106-164:sac_list_to_4D porritt$ make install
install -s sac_list_to_4D ~/bin
guest-wireless-upc-1606-10-120-106-164:sac_list_to_4D porritt$
```

# Step 2: run decimate_sac_data.tcsh

```
#!/bin/tcsh

# Simple script to decimate the data by a factor of 100 and low pass filter at 10
seconds

# Little extra to set it looping over each event
foreach event (`ls -d Event_*/`)

cd $event

# Sac set of commands to do the actual decimation
sac << EOF
read *BHZ.SAC
lp bu co 0.1
decimate 5
decimate 5
decimate 4
write append .decimated
quit
EOF

# Little script to do the 4D command
ls *.decimated > list.bhz.deci
sac_list_to_4D list.bhz.deci bhz.4d.deci

cd ../

end
```

Each original trace is 144,001 samples

This makes a giant (~1GB) data file

Such a big file is unwieldy and causes crashes

Step 3: copy the make_chichijima_movie.m to the Event directory and navigate there in Matlab. Run the matlab script.

Go ahead and read the script. It's only 112 lines, half comments, half blanks space, and half commands.

Step 3a: read and organize data from sac_list_to_4D
Step 3b: set map parameters
Step 3c: set waveform parameters
Step 3d: Loop through each point in time
Step 3e: find the latitude, longitude, and amplitudes at this time step.
Step 3f: make a scatter plot of amplitudes
Step 3g: plot waveform with sliding bar indicating time point
Step 3h: save the frame
Step 3i: write the video from frames

If time remaining, go ahead and check out sac_list_to_4D.c

It is an example of using c code to read a list of sac files and calling the sacio library.