Revisiting SEP tour with Madagascar and SCons

Sergey Fomel[1]

ABSTRACT

Many appreciative users were introduced to SEPlib (Claerbout, 1991) by an excellent article of Dellinger and Tálas (1992). In this paper, I show how to create a similar experience using Madagascar and SCons.

GETTING STARTED

Similarly to SU and SEPlib, RSF programs can be piped and executed from the command line, for example:

```
bash$ sfspike n1=1000 k1=300 title="\s200 Welcome to \c2 RSF" | \
sfbandpass fhi=2 phase=1 | sfwiggle | sfpen
```

If you are already familiar with SEPlib, you can find most of the familiar programs with the names prepended by "sf".

Typing a command without arguments, should produce a concise self-documentation.

```
bash$ sfbandpass
```

The recommended way of using RSF, however, is not with the command line but with SCons and "SConstruct" files.

Setting up

Open a file named "SConstruct" in your favorite editor and start it with a line

```
5 from rsf.proj import *
```

This line tells Python to load the RSF project module.

[1]**e-mail:** sergey.fomel@beg.utexas.edu

Obtaining the test data

Add a Fetch command as follows:

```
11 Fetch('Txx.HH','septour')
```

Now, by running

```
bash$ scons Txx.HH
```

you can instruct SCons to connect to an anonymous data server and extract (fetch) the data file "Txx.HH" from the "septour" directory.

Displaying the data

Add the following line to the `SConstruct` file:

```
17 Result('wiggle0','Txx.HH','wiggle')
```

Note that it does not matter if this line appears before or after the "Fetch" line. You are simply instructing SCons how to create a result plot from the input.

Run

```
bash$ scons wiggle0.view
```

If everything is setup correctly in your environment, you should see something like the following output in your terminal:

```
bash$ scons wiggle0.view
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
retrieve(["Txx.HH"], [])
< Txx.HH /path/to/RSF/bin/sfwiggle > Fig/wiggle0.vpl
/path/to/RSF/bin/sfpen Fig/wiggle0.vpl
```

and a figure similar to Figure 1 appearing on your screen.

PROCESSING EXERCISES

Windowing and plotting

Our next task is to window and plot a significant portion of the data. Add the following line to the `SConstruct` file:

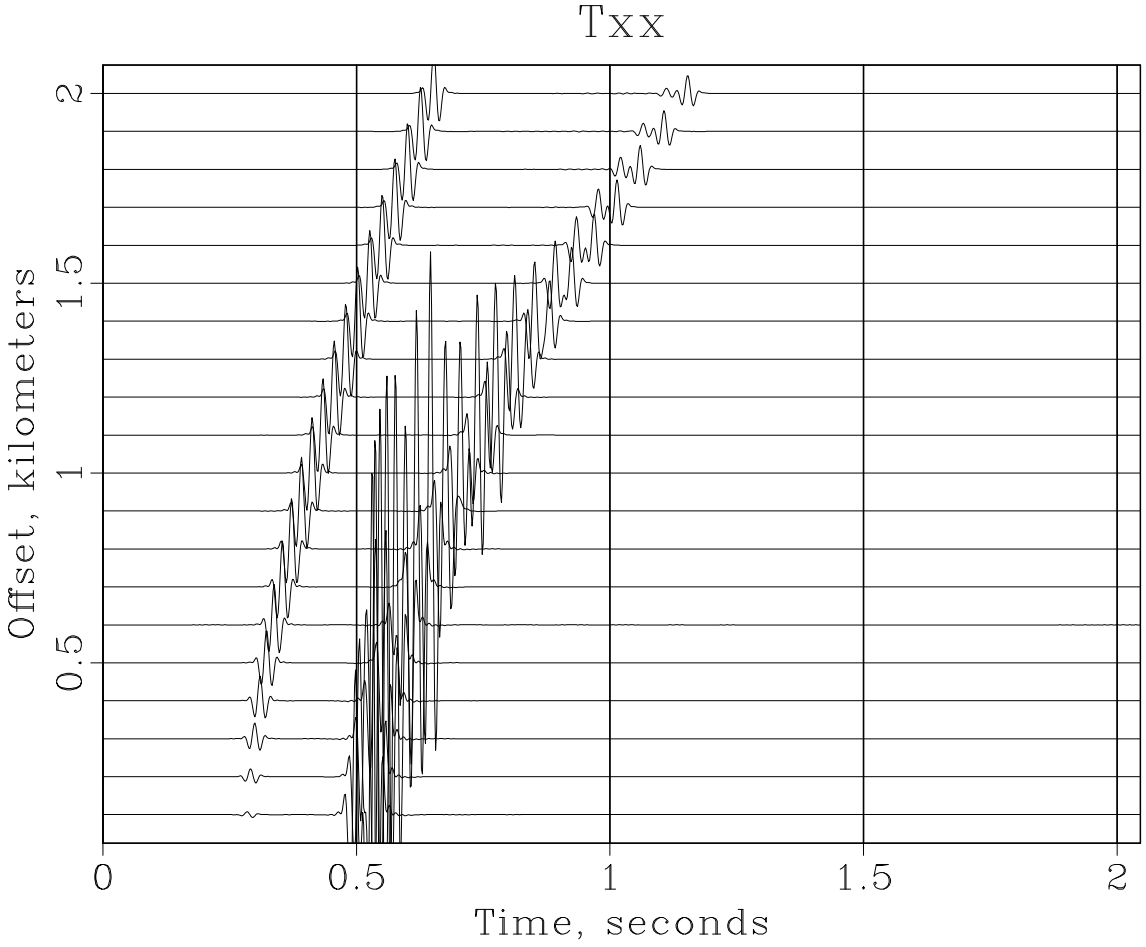

Figure 1: To see this figure on your screen, run `scons wiggle0.view`

```
23 Flow('windowed','Txx.HH','window n2=10 min1=0.4 max1=0.8')
```

The window command selects the first ten traces and the time window between 0.4 and 0.8 seconds.

We will plot the windowed data with three different plotting programs.

```
25 plotpar = '''
26 transp=y poly=y yreverse=y pclip=100 nc=20 allpos=n
27 unit2=km unit1=s label1=Time label2=Offset
28 '''
29
30 for plot in ('wiggle','contour','grey'):
```

For convenience, plotting parameters are put in a string called `plotpar`. A Python string can be enclosed in single, double, or triple quotes. Triple quotes allow the string to span multiple lines. In this case, we use triple quotes for convenience. Next, we loop (using Python's `for` construct) through three different programs (`wiggle`, `contour`, and `grey`). For each program, the command portion of `Result` is formed by concatenating two strings with Python's addition operator.

Try running `scons -Q wiggle.view`. You should see something like the following output in your terminal:

```
bash$ scons -Q wiggle.view
< Txx.HH /path/to/RSF/bin/sfwindow n2=10 n1=200 f1=200 > windowed.rsf
< windowed.rsf /path/to/RSF/bin/sfwiggle transp=y poly=y yreverse=y
pclip=100 nc=200 > Fig/wiggle.vpl
/path/to/RSF/bin/sfpen Fig/wiggle.vpl
```

and a figure similar to Figure 2 appearing on your screen. The `-Q` switch tells SCons to run in a quiet mode, suppressing verbose comments. We will use it from now on to save space. You can dismiss the figure by using the "q" key on the keyboard or by hitting the "quit" button.

Run `scons -Q view`, and you should see simply

```
bash$ scons -Q view
/path/to/RSF/bin/sfpen Fig/wiggle.vpl
```

Since the `wiggle.vpl` figure is up to date, SCons does not rebuild it. After quitting the figure, SCons will resume processing with

```
< windowed.rsf /path/to/RSF/bin/sfcontour transp=y poly=y yreverse=y
pclip=100 nc=200 > Fig/contour.vpl
/path/to/RSF/bin/sfpen Fig/contour.vpl
```

and a figure similar to Figure 3 appearing on your screen. Quitting the figure, produces

```
< windowed.rsf /path/to/RSF/bin/sfgrey transp=y poly=y yreverse=y
pclip=100 nc=200 > Fig/grey.vpl
/path/to/RSF/bin/sfpen Fig/grey.vpl
```

and Figure 4.

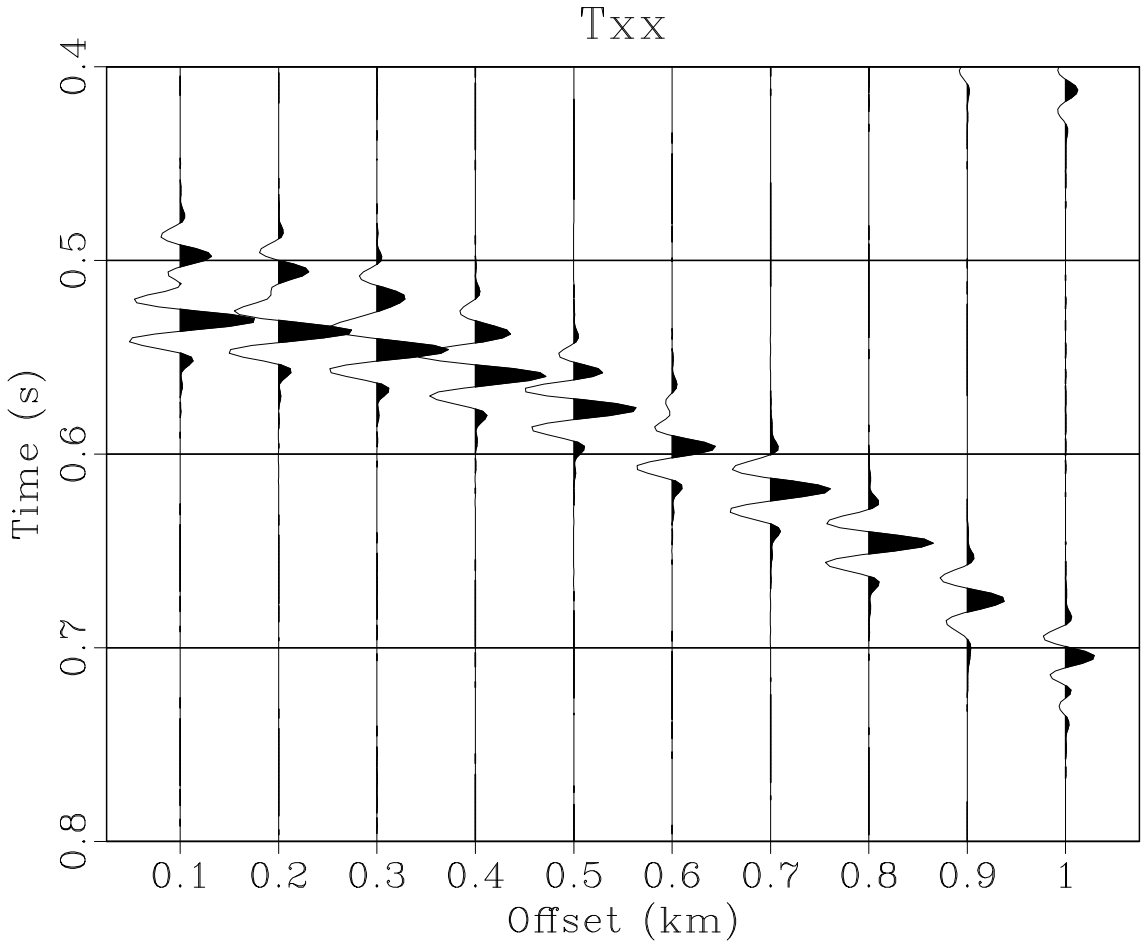

Figure 2: To see this figure on your screen, run `scons wiggle.view`

Resampling

The next example demonstrated simple signal processing using the Fast Fourier Transform. We will first subsample the original data and then recover the data using Fourier interpolation.

Subsampling is accomplished with `sfwindow`.

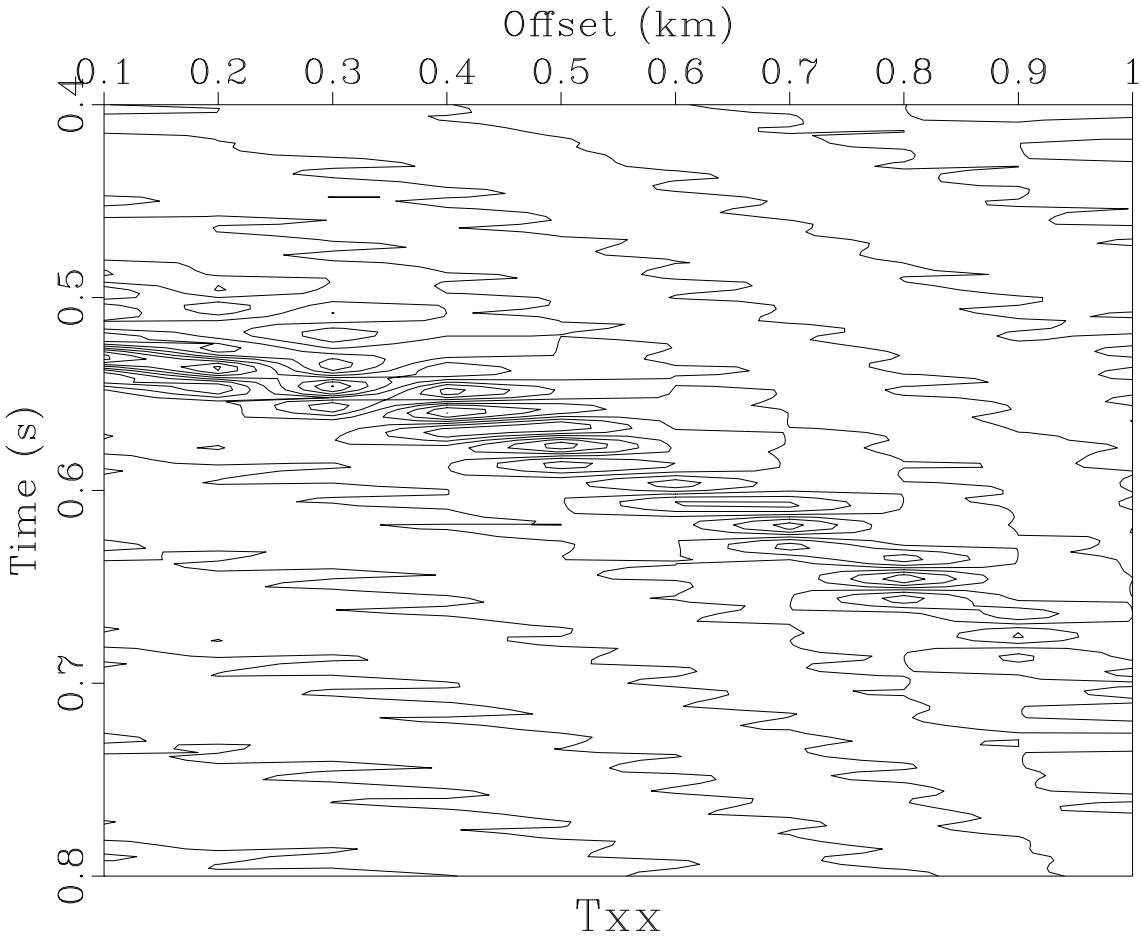

Figure 3: To see this figure on your screen, run `scons contour.view`

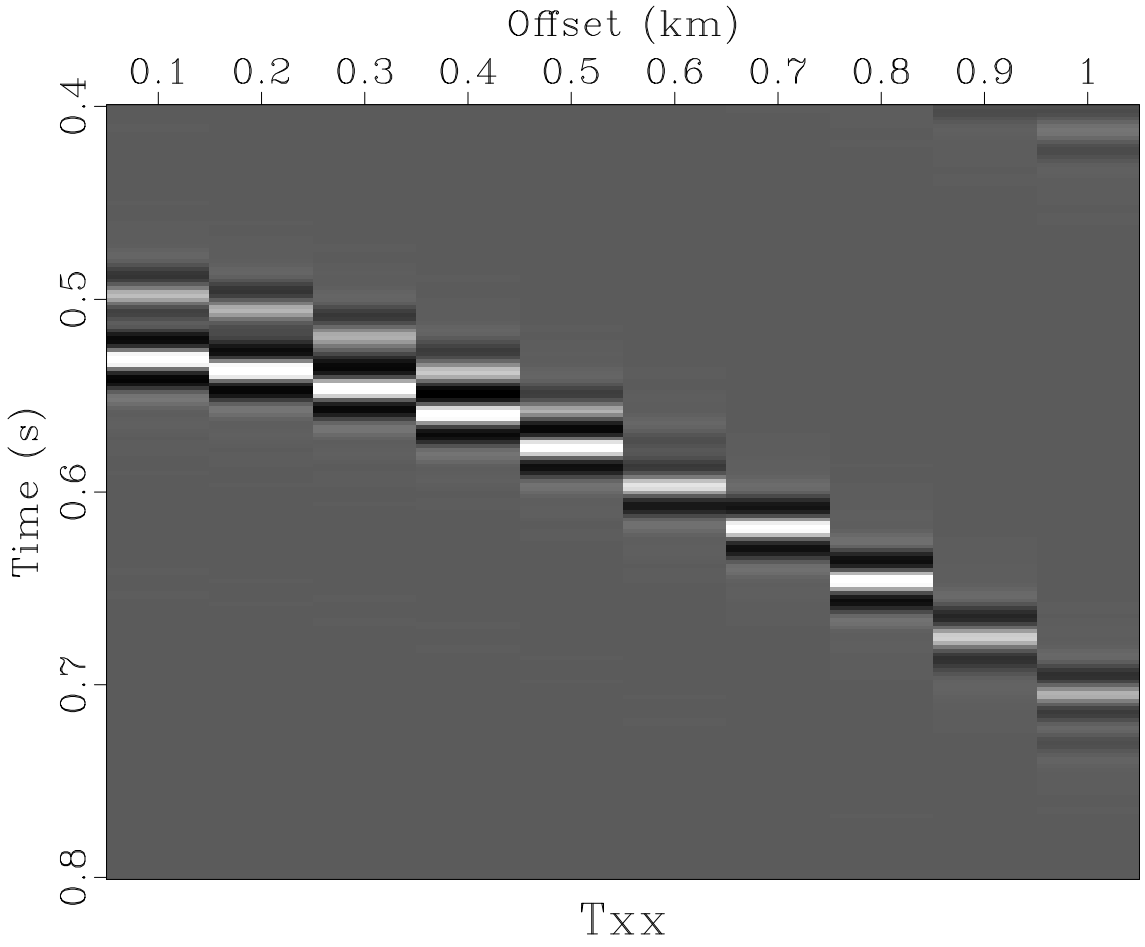

Figure 4: To see this figure on your screen, run `scons grey.view`

```
36
37 # decimate time axis by two
```

Running `scons -Q subsampled.rsf` produces

```
< windowed.rsf /path/to/RSF/bin/sfwindow j1=2 > subsampled.rsf
```

We can verify that the size of the first axis has decreased by running

```
sfin windowed.rsf subsampled.rsf.
```

Try also `sfwiggle < subsampled.rsf | sfpen` to quickly inspect the subsampled data on the screen.

To interpolate the data back to the original sampling, the following sequence of steps can be applied:

1. Fourier transform from time domain to frequency domain.
2. Pad the frequency axis
3. Inverse Fourier transform from frequency to time.

All three steps are conveniently combined into one using pipes.

```
39
40 # sinc interpolation in the Fourier domain
41 Flow('resampled','subsampled',
```

Why do we pad the Fourier domain to 102? The time length of the original data is 201 samples. In the frequency domain, it can be represented with 101 positive frequencies plus the zero frequency, which amounts to 102. Note that the output of `sffft1` does not contain negative frequencies.

Finally, we display the result. The reconstructed data is shown in Figure 5. Comparing this result with Figure 2, we can verify a fairly accurate reconstruction.

As an exercise, try subsampling the data by a factor of 4 and see if you can still reconstruct the original data with the Fourier method.

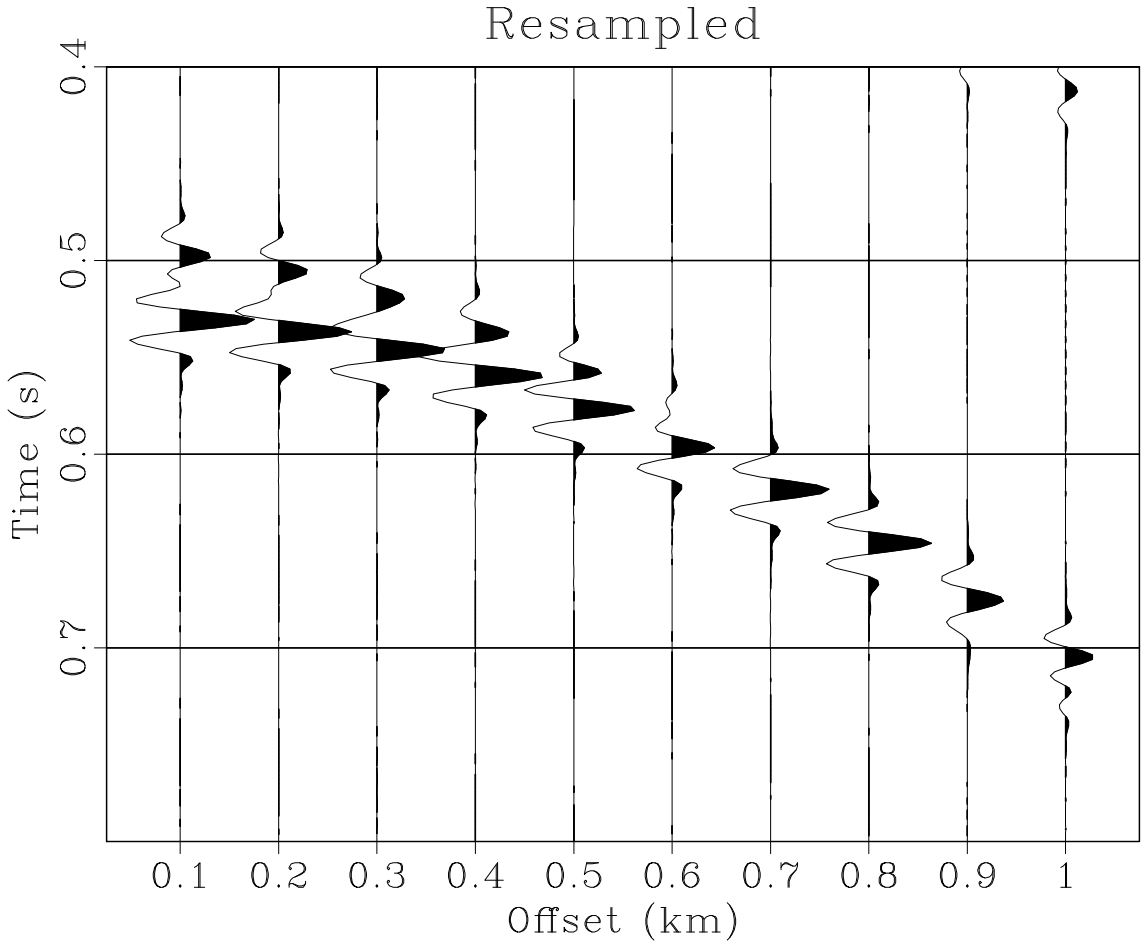

Figure 5: To see this figure on your screen, run `scons resampled.view`

Normal Moveout

The next example applies a simple constant-velocity NMO correction to the windowed data and pipes the result to a wiggle plotting command:

```
49
50 Result('nmo','windowed',
51         '''
52         nmostretch v0=2.05 half=n |
53         wiggle pclip=100 max1=0.6 poly=y
```

Running `scons -Q nmo.view` produces

```
< windowed.rsf /path/to/RSF/bin/sfnmostretch v0=2.05 half=n |
/path/to/RSF/bin/sfwiggle pclip=100 max1=0.6 poly=y > Fig/nmo.vpl
/path/to/RSF/bin/sfpen Fig/nmo.vpl
```

and Figure 6. Note that SCons does not recreate the `windowed.rsf` file if that file is up to date. You can experiment with the NMO velocity (2.05 km/s) or with plotting parameters to get different results. As Dellinger and Tálas (1992) point out, the NMO velocity of 2.05 km/s "appears to split the difference between two distinctly non-hyperbolic shear waves".

Advanced plotting

Sometimes, we need to combine different plots either by overlaying them on top of each other or by putting them side by side. Here is an example of accomplishing it with RSF and SCons.

Start by creating common plotting plotting arguments and plotting the data in greyscale.

```
59
60 plotpar = plotpar+' min1=.4 max1=.8 max2=1. min2=.05 poly=n'
61
62 Plot('grey','windowed',
```

Next, plot the wiggle traces twice: the fist time, using thick black lines (`plotcol=0 plotfat=10`), and the second time, using thinner white lines (`plotcol=7 plotfat=5`).

```
63      'grey wheretitle=t wherexlabel=b' + plotpar)
64 Plot('wiggle1','windowed',
65      'wiggle plotcol=0 plotfat=10' + plotpar)
66 Plot('wiggle2','windowed',
```

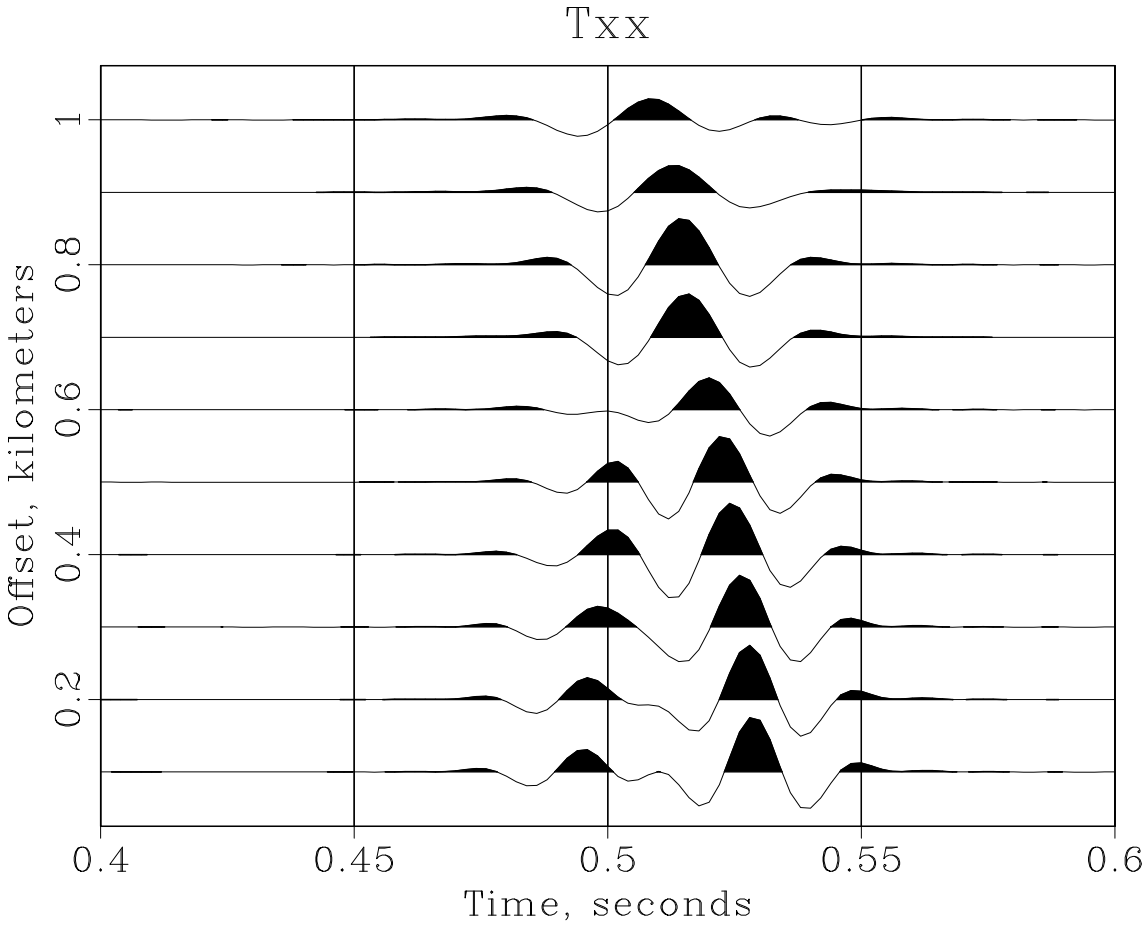

Figure 6: To see this figure on your screen, run `scons nmo.view`

The plots are combined by overlaying or by putting them side by side.

```
68
69 Result('overplot','grey wiggle1 wiggle2','Overlay')
```

The resultant plots are shown in Figures 7 and 8.

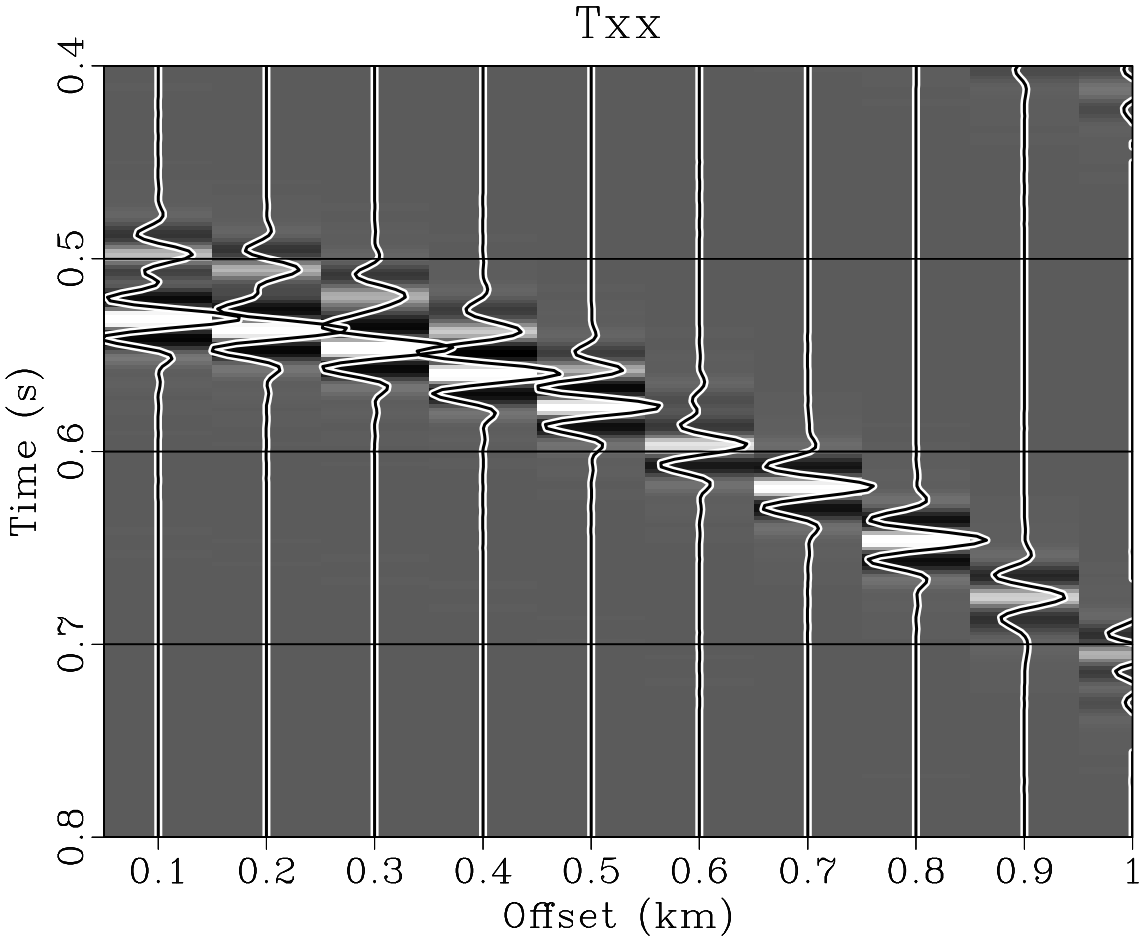

Figure 7: To see this figure on your screen, run `scons overplot.view`

CONCLUSIONS

This tour is not designed as a comprehensive manual. It simply gives a glimpse into working in a reproducible research environment with RSF and SCons. The reader is encouraged to experiment with the `SConstruct` file attached to this tour and included in the Appendix. For other documentation on RSF, please see

- Introduction to RSF
- Installation instructions

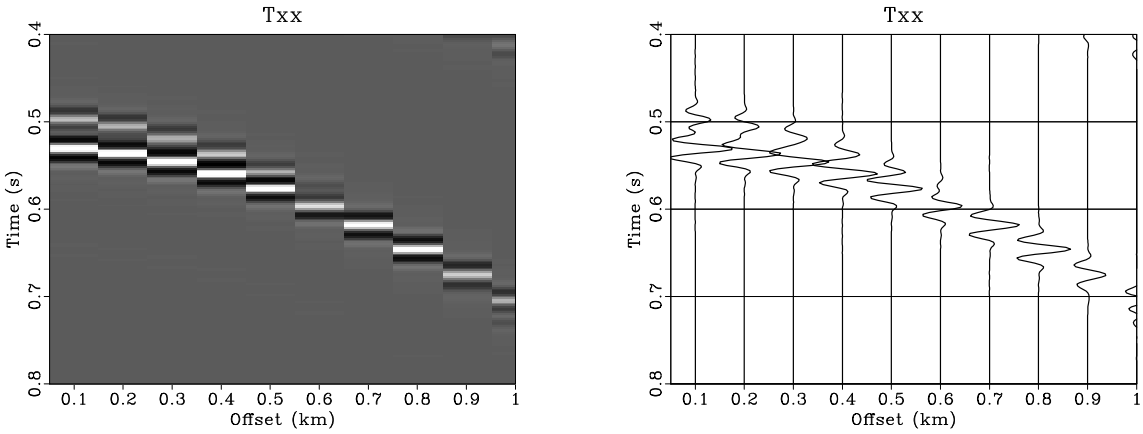

Figure 8: To see this figure on your screen, run `scons sidebyside.view`

- Self-documentation reference for RSF programs
- A guide to RSF programs
- A guide to RSF file format
- A guide to RSF programming interface
- A guide to programming with RSF
- A guide to SCons interface for reproducible computations

ACKNOWLEDGMENTS

Thanks to Joe Dellinger and Sándor Tálas for creating "SEP tour" and to James Rickett for updating it. Several generations of SEP students contributed to SEPlib. We try to preserve all their good ideas when refactoring SEPlib into RSF.

The test dataset used in this paper is courtesy of Beltram Nolte and L. Neil Frazer.

REFERENCES

Claerbout, J. F., 1991, Introduction to Seplib and SEP utility software, *in* SEP-70: Stanford Exploration Project, 413–436.

Dellinger, J., and S. Tálas, 1992, A tour of SEPlib for new users, *in* SEP-73: Stanford Exploration Project, 461–502.

SCONSTRUCT FILE

Here is a complete listing of the `SConstruct` file used in this example.

```
##############################################################
# Setting up
##############################################################

from rsf.proj import *

##############################################################
# Obtaining the test data
##############################################################

Fetch('Txx.HH','septour')

##############################################################
# Displaying the data
##############################################################

Result('wiggle0','Txx.HH','wiggle')

##############################################################
# Windowing and plotting
##############################################################

Flow('windowed','Txx.HH','window n2=10 min1=0.4 max1=0.8')

plotpar = '''
transp=y poly=y yreverse=y pclip=100 nc=20 allpos=n
unit2=km unit1=s label1=Time label2=Offset
'''

for plot in ('wiggle','contour','grey'):
    Result(plot,'windowed',plot + plotpar)

##############################################################
# Resampling
##############################################################

# decimate time axis by two
Flow('subsampled','windowed','window j1=2')

# sinc interpolation in the Fourier domain
Flow('resampled','subsampled',
```

```
42      'fft1 | pad n1=102 | fft1 inv=y opt=n | window max1=0.8')
43
44  Result('resampled','wiggle title=Resampled' + plotpar)
45
46  ##########################################################
47  # Velocity analysis and NMO
48  ##########################################################
49
50  Result('nmo','windowed',
51         '''
52         nmostretch v0=2.05 half=n |
53         wiggle pclip=100 max1=0.6 poly=y
54         ''')
55
56  ##########################################################
57  # Advanced plotting
58  ##########################################################
59
60  plotpar = plotpar+' min1=.4 max1=.8 max2=1. min2=.05 poly=n'
61
62  Plot('grey','windowed',
63       'grey wheretitle=t wherexlabel=b' + plotpar)
64  Plot('wiggle1','windowed',
65       'wiggle plotcol=0 plotfat=10' + plotpar)
66  Plot('wiggle2','windowed',
67       'wiggle plotcol=7 plotfat=3' + plotpar)
68
69  Result('overplot','grey wiggle1 wiggle2','Overlay')
70  Result('sidebyside','grey wiggle2','SideBySideIso')
71
72  ##########################################################
73  # Wrapping up
74  ##########################################################
75
76  End()
```