Seismic field data processing example

— Towards real world —

Yang Liu

Bureau of Economic Geology
Jackson School of Geosciences
The University of Texas at Austin
yangliu1979@gmail.com

July 24, 2010

Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Outline

1. Assignment

2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Assignment

Field seismic data are always complicated, a tutorial is given for setting up a simple processing workflow.

Tutorial workflow

- Conversion to RSF format
- Initial data checking
- Initial signal analysis and quality control
  - First break mute
  - Subsampling with anti-aliasing filter
  - Ground-roll attenuation with time-frequency analysis tool
- Initial velocity analysis
- Brute stacking
- Toy prestack time migration

Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Outline

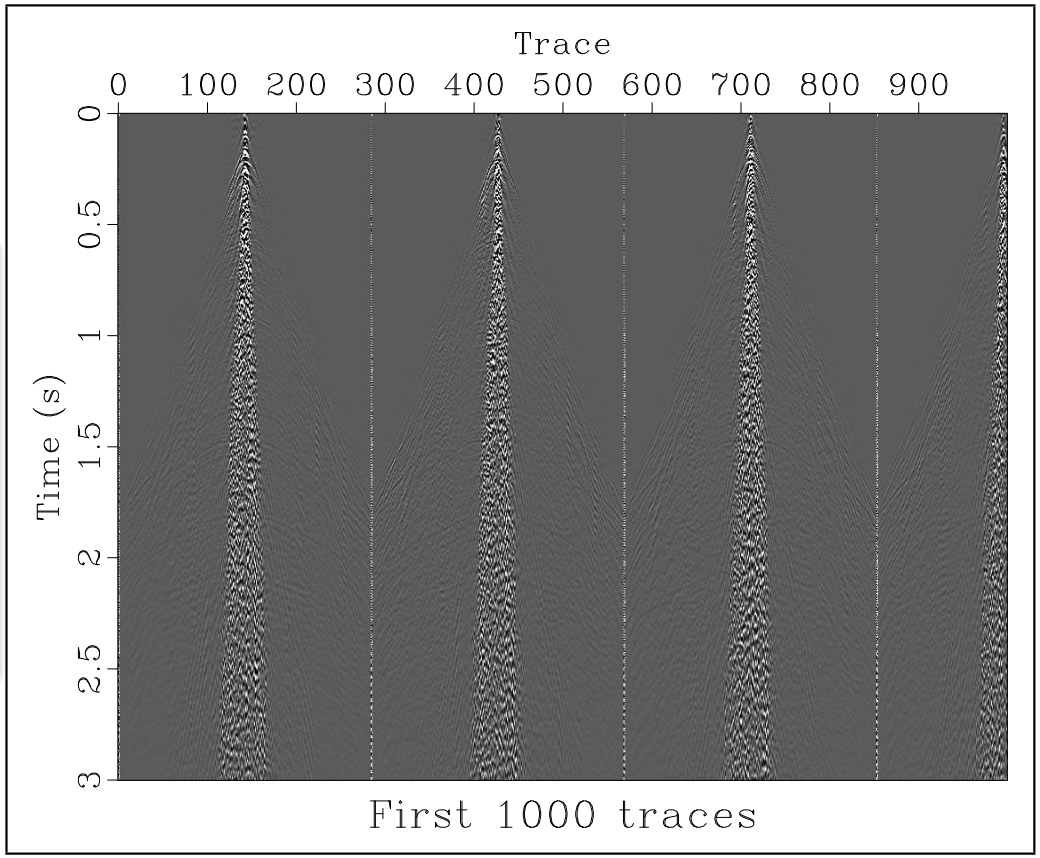
1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Convert to RSF format

```
from rsf.proj import *

##########################################################
tgz = '2D_Land_data_2ms.tgz'
Fetch(tgz,
      server='http://www.freeusp.org',
      top='RaceCarWebsite/TechTransfer/Tutorials/Processing_2D',
      dir='Data')

files = map(lambda x: 'Line_001.'+x,Split('TXT SPS RPS XPS sgy'))
Flow(files,tgz,
     'gunzip -c $SOURCE | tar -xvf -',stdin=0,stdout=-1)
##########################################################

Flow('line tline','Line_001.sgy','segyread tfile=${TARGETS[1]}')
```

Display some shots

```
Result('first','line',
       '''
       window n2=1000 |
       agc rect1=250 rect2=100 |
       grey title="First 1000 traces"
       ''')
```



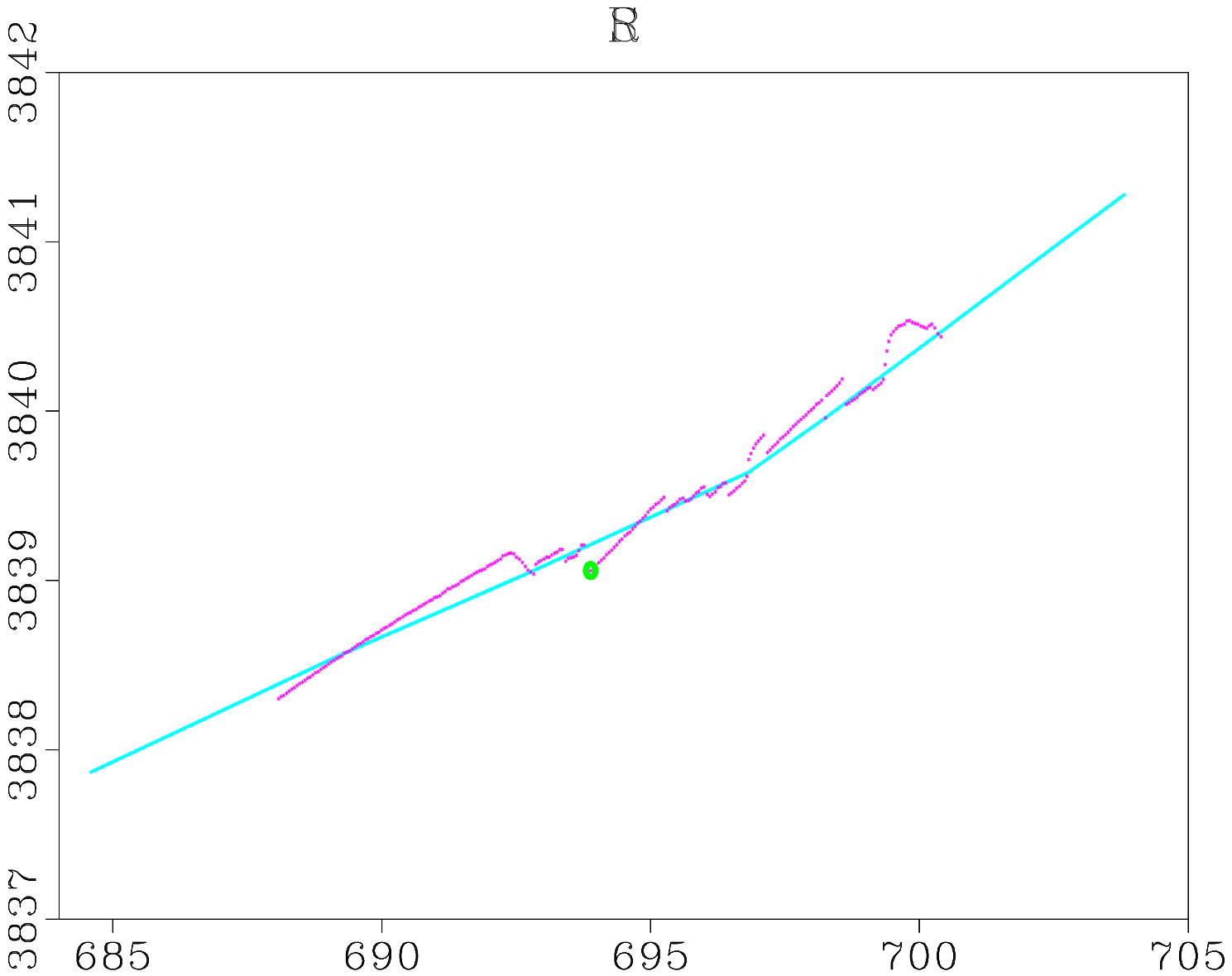
True geometry

```
lines = {'S':251,'R':782}

color = {'S':4, 'R':2}

for case in 'SR':

    Flow(case+'.asc','Line_001.%cPS' % case,'''awk 'NR > 20 {print $8, " ", $9}' ''')

    Flow(case,case+'.asc',

         '''

         echo in=$SOURCE data_format=ascii_float n1=2 n2=%d | dd form=native

         ''' % lines[case],stdin=0)

    Plot(case,

         '''

         scale dscale=0.001 | dd type=complex |

         graph symbol=* title=%c plotcol=%d

         min1=684 max1=705 min2=3837 max2=3842

         ''' % (case,color[case]))
```

True geometry

```
Plot('s118','S',
     '''
     window n2=1 f2=117 | scale dscale=0.001 | dd type=complex |
     graph symbol=O wanttitle=n plotcol=3 symbolsz=4 plotfat=10
     min1=684 max1=705 min2=3837 max2=3842
     ''')
Result('SRO','R S s118','Overlay')
```

Shots and receivers position

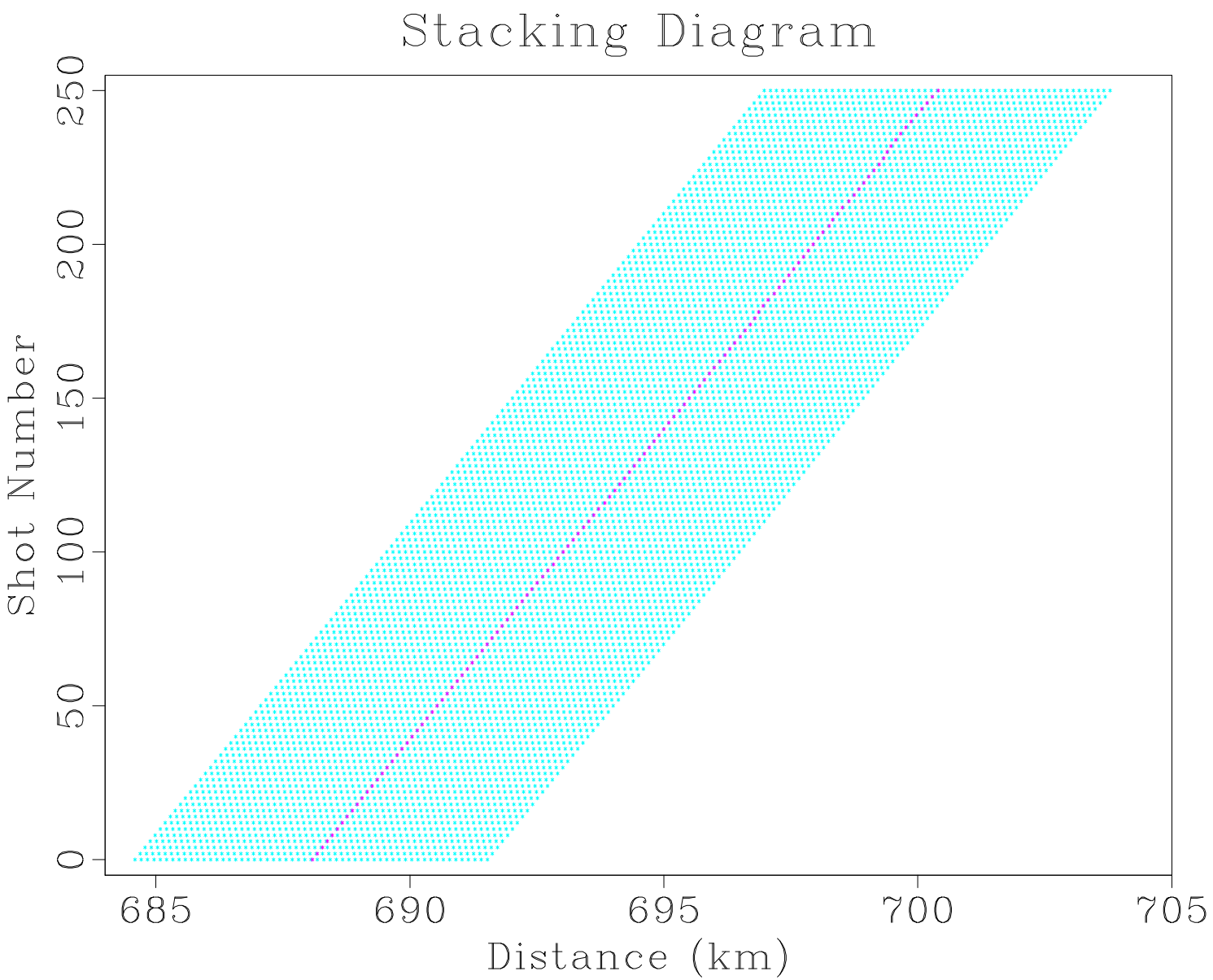


Arrange receiver coordinates

```
# Arrange receiver coordinates

shots = []

for shot in range(lines['S']):

    line = 'line%d' % shot

    Flow(line,'R','window f2=%d n2=282' % (2*shot))

    shots.append(line)

Flow('rece',shots,'rcat axis=3 ${SOURCES[1:%d]}' % len(shots))
```

Plot stacking diagram

```
Plot('rec','rece',
     '''
     window n1=1 j2=5 j3=2 | scale dscale=0.001 |
     rtoc | math output="input+I*x2" |
     graph symbol=* plotcol=%d title="Stacking Diagram"
     label1=Distance unit1=km label2="Shot Number"
     min1=684 max1=705
     ''' % color['R'])
Plot('sou','S',
     '''
     window n1=1 j2=2 | scale dscale=0.001 |
     rtoc | math output="input+I*x1" |
     graph symbol=* plotcol=%d wanttitle=n wantaxis=n min1=684 max1=705
     ''' % color['S'])
Result('diagram','rec sou','Overlay')
```


Stacking Diagram
Shot Number
0
50
100
150
200
250
685
690
695
700
705
Distance (km)

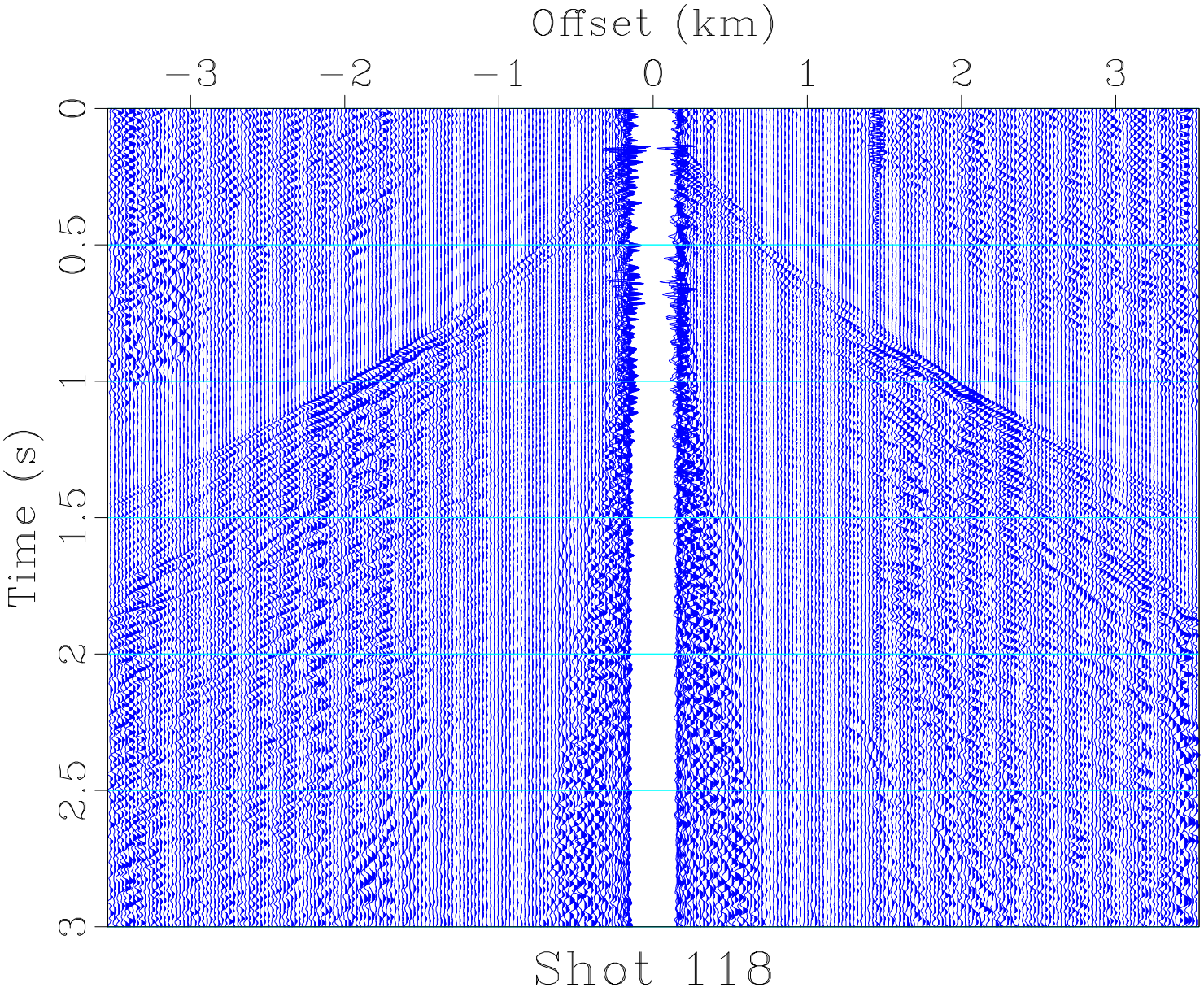
Plot shots using true geometry

```
# Display one shot using true geometry

Flow('sour','S','spray axis=2 n=282')

Flow('sx','sour','window n1=1 | scale dscale=0.001')

Flow('sy','sour','window n1=1 f1=1 | scale dscale=0.001')

Flow('rx','rece','window n1=1 | scale dscale=0.001')

Flow('ry','rece','window n1=1 f1=1 | scale dscale=0.001')


Flow('offset','sx sy rx ry',

     '''

     math SX=${SOURCES[0]} SY=${SOURCES[1]}

     RX=${SOURCES[2]} RY=${SOURCES[3]}

     output="sqrt((RX-SX)^2+(RY-SY)^2)"

     ''')
```

Plot shots using true geometry

```
Flow('shot118','lines','window n2=1 f2=117')

Flow('offset118','offset','window n2=1 f2=117')

Flow('boff1','offset118','window n1=141 | math output="-input"')

Flow('boff2','offset118','window f1=141')

Flow('boff118','boff1 boff2','cat axis=1 ${SOURCES[1]}')

Result('shot118','shot118 boff118',

       '''

       agc rect1=50 rect2=50 |

       wiggle xpos=${SOURCES[1]} transp=y yreverse=y poly=y

       wherexlabel=t wheretitle=b title="Shot 118"

       ''')
```

Plot shot 118



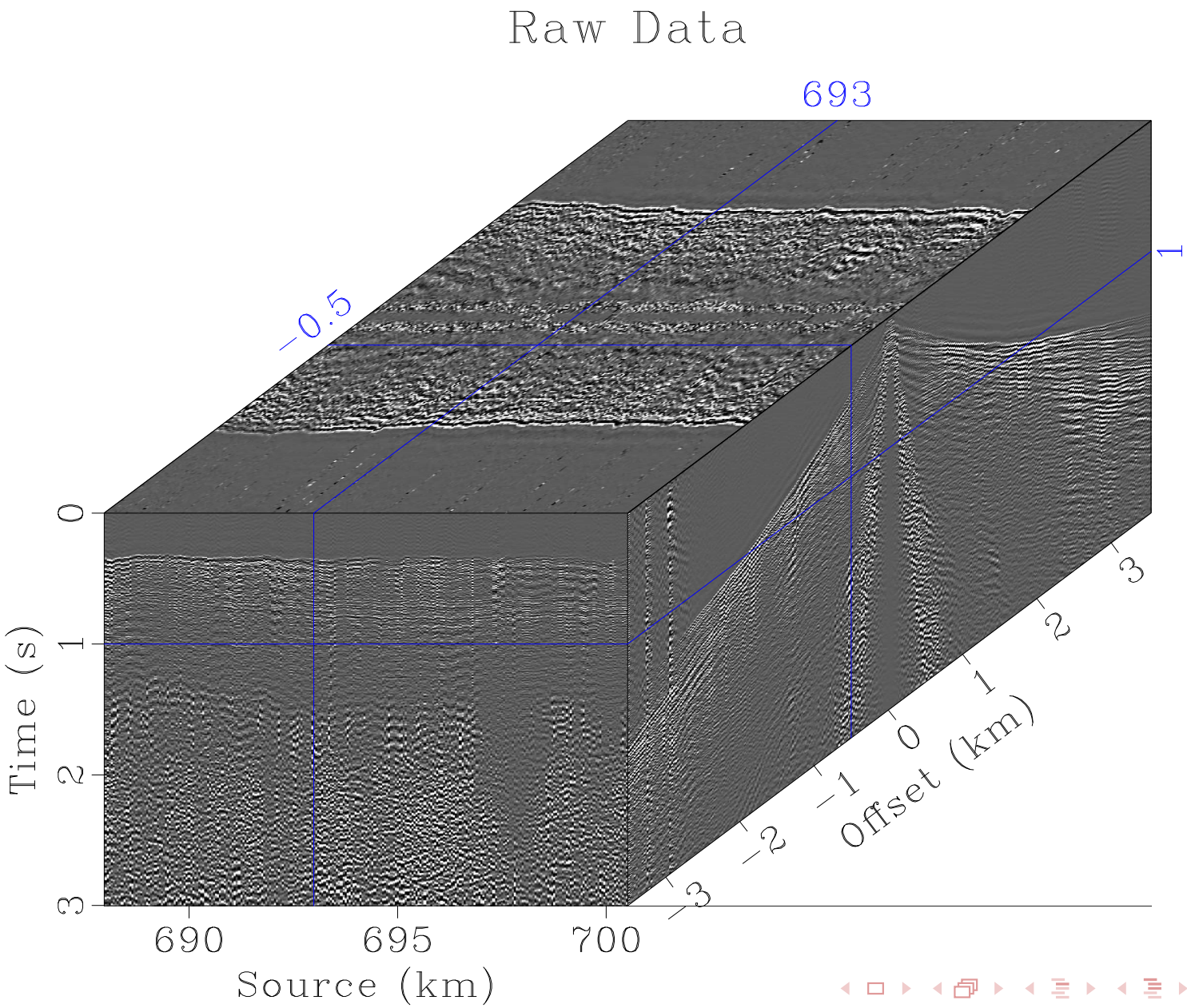
Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Map to regular shot gathers

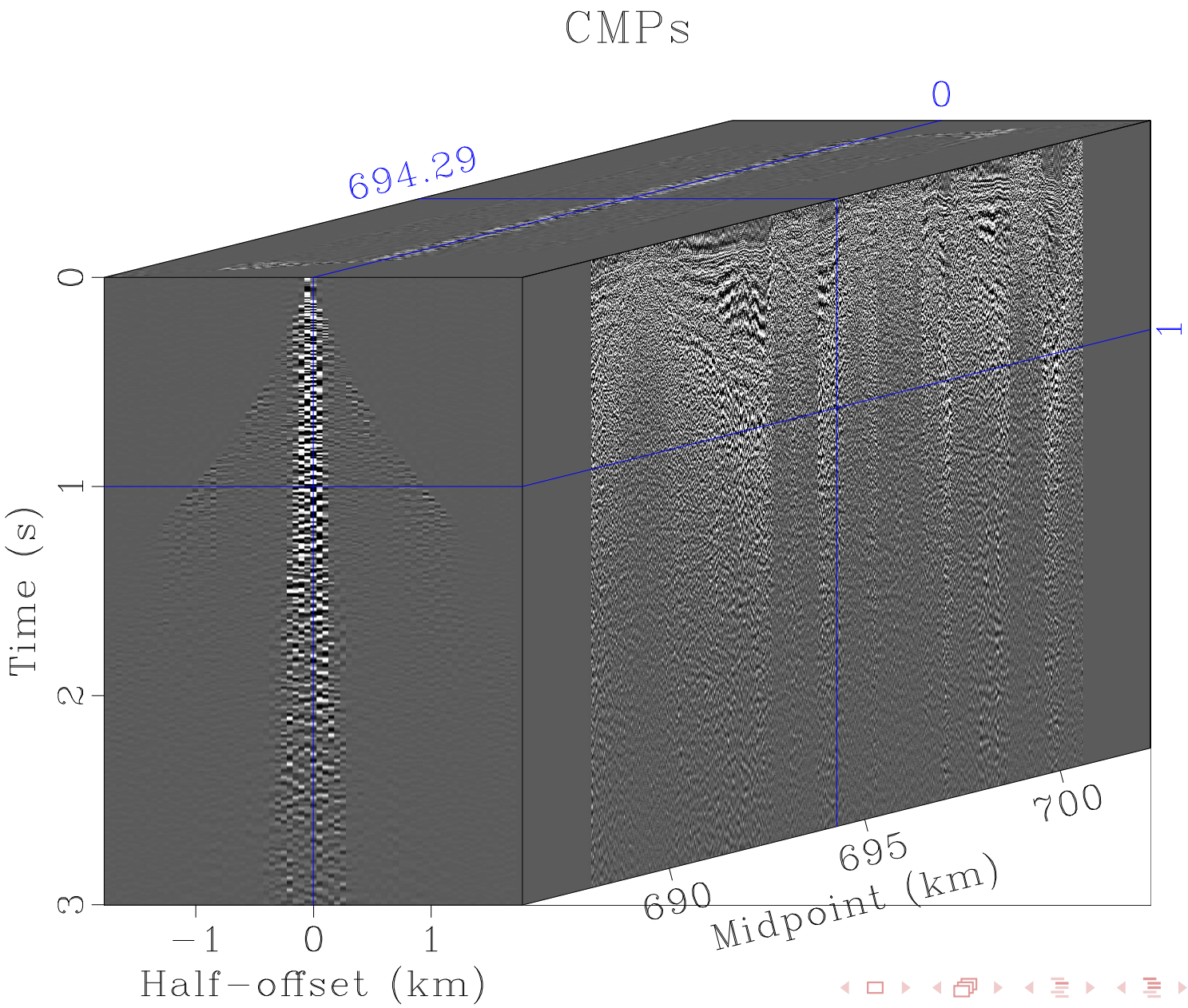
```
Flow('lines','line',
     '''
     intbin yk=cdpt | window f3=2 |
     put
     label2=Source d2=0.05  o2=688  unit2=km
     label3=Offset d3=0.025 o3=-3.5 unit3=km
     label1=Time unit1=s
     ''')

Result('lines',
       '''
       byte gainpanel=each |
       grey3 frame1=500 frame2=100 frame3=120 flat=n
       title="Raw Data"
       ''')
```

2-D shot gathers (suppose regular coordinates)



Convert shots to CMPs

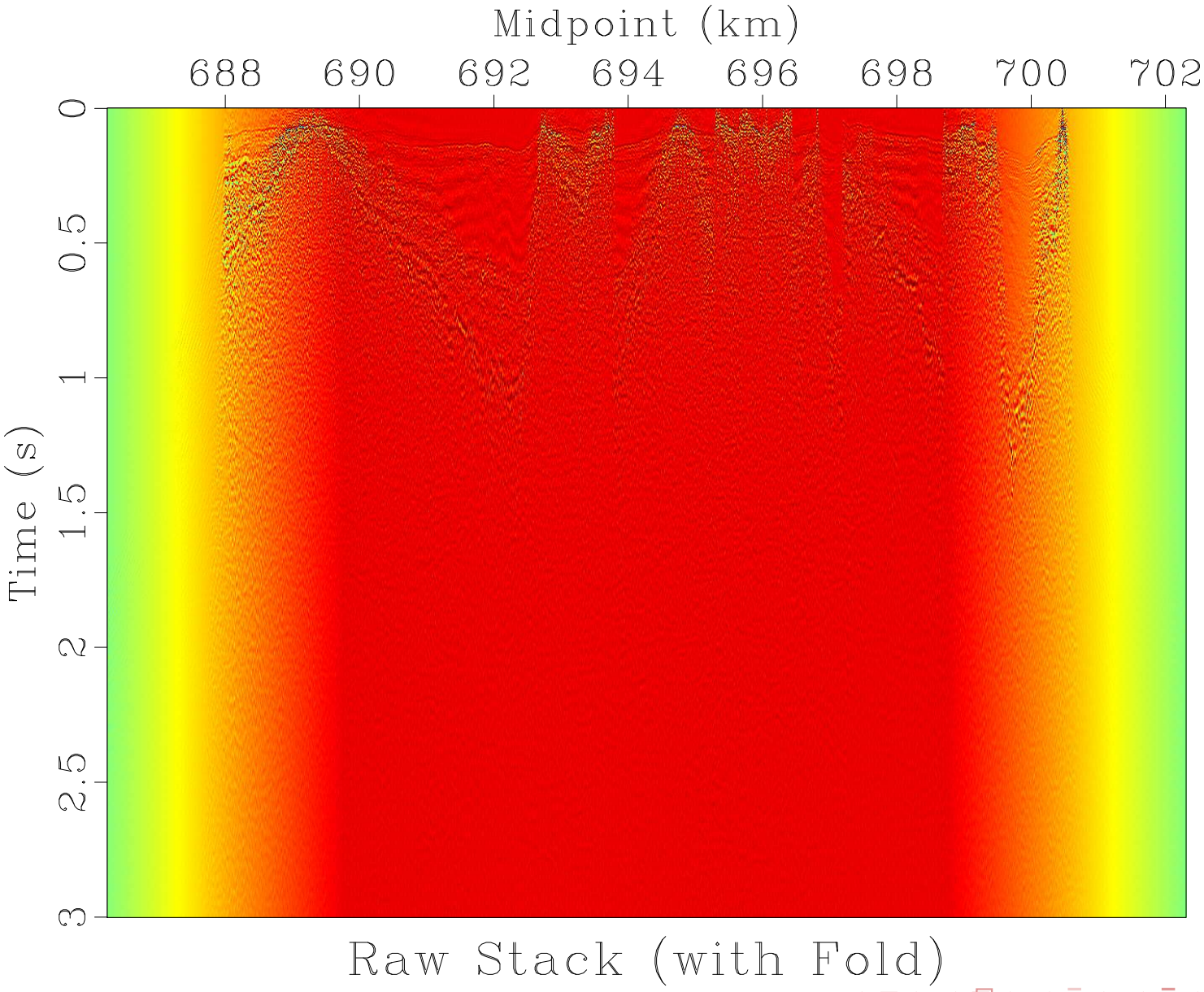
```
Flow('rcmps mask','lines',
     '''
     transp memsize=1000 plane=23 |
     shot2cmp mask=${TARGETS[1]} half=n |
     put o2=-1.75 label2="Half-offset"
     ''')
Result('rcmps',
       '''
       byte gainpanel=each |
       grey3 frame1=500 frame2=35 frame3=643 flat=n
       point1=0.8 point2=0.4 title="CMPs"
       ''')
```


CMPs
0
694.29
1
Time (s)
0
1
2
3
−1
0
1
Half−offset (km)
690
695
700
Midpoint (km)

Raw stacking

```
Flow('fold','mask','dd type=float | stack axis=1 norm=n')
Flow('rstack','rcmps','stack')
Result('rstack','fold rstack',
       '''
       spray axis=1 n=1501 d=0.002 o=0 label=Time unit=s |
       add scale=1,1000 ${SOURCES[1]} |
       grey color=j title="Raw Stack (with Fold)"
       ''')
```

Raw stacking



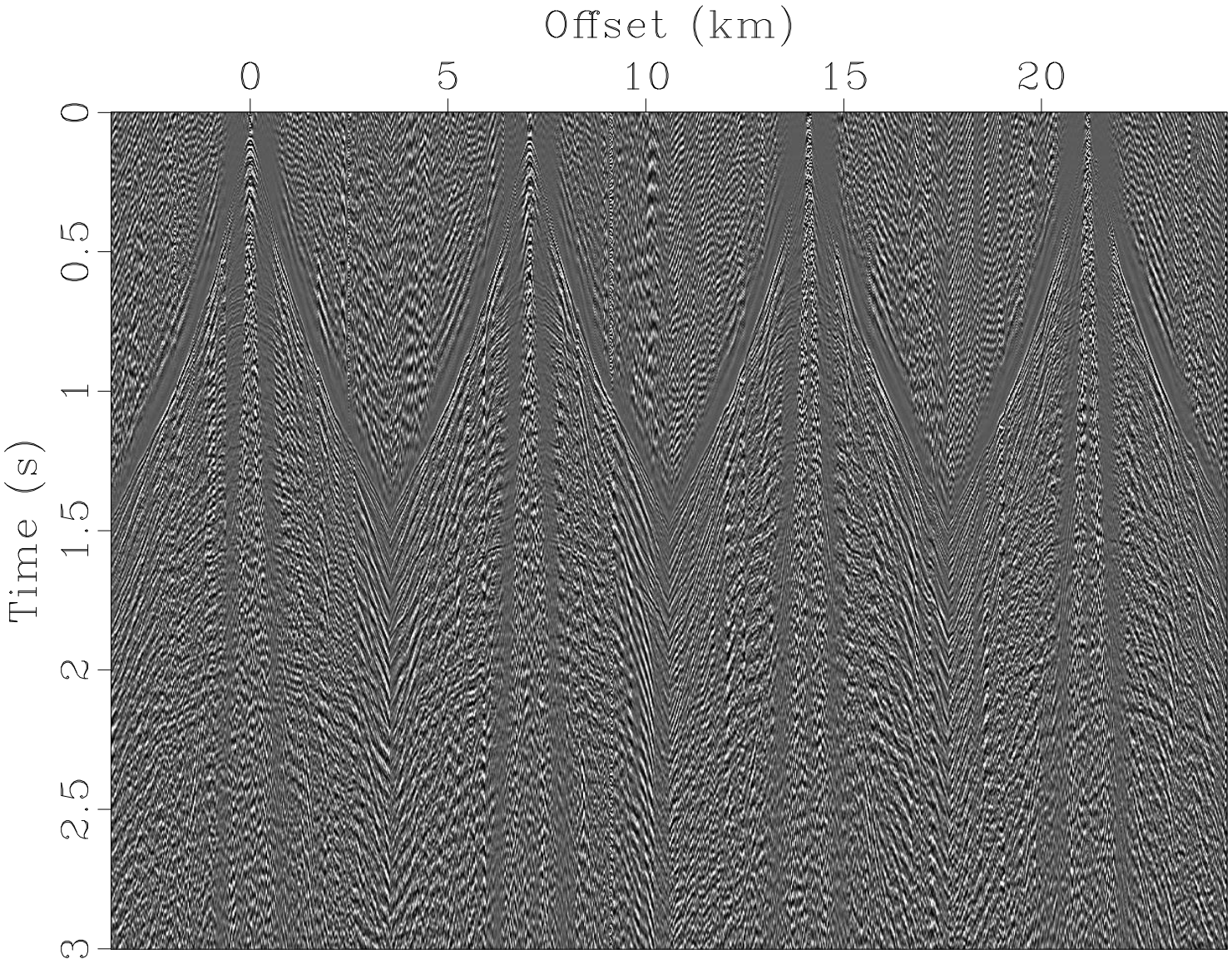
Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

First break mute

```
# Select 4 shots every tenth sequential shot
Flow('inpmute','lines',
     '''
     window f2=198 j2=10 n2=4 |
     transp plane=23
     ''')
Result('inpmute',
       '''
       put n2=1128 n3=1 |
       agc rect1=50 rect2=20 | grey wanttitle=n
       ''')
```

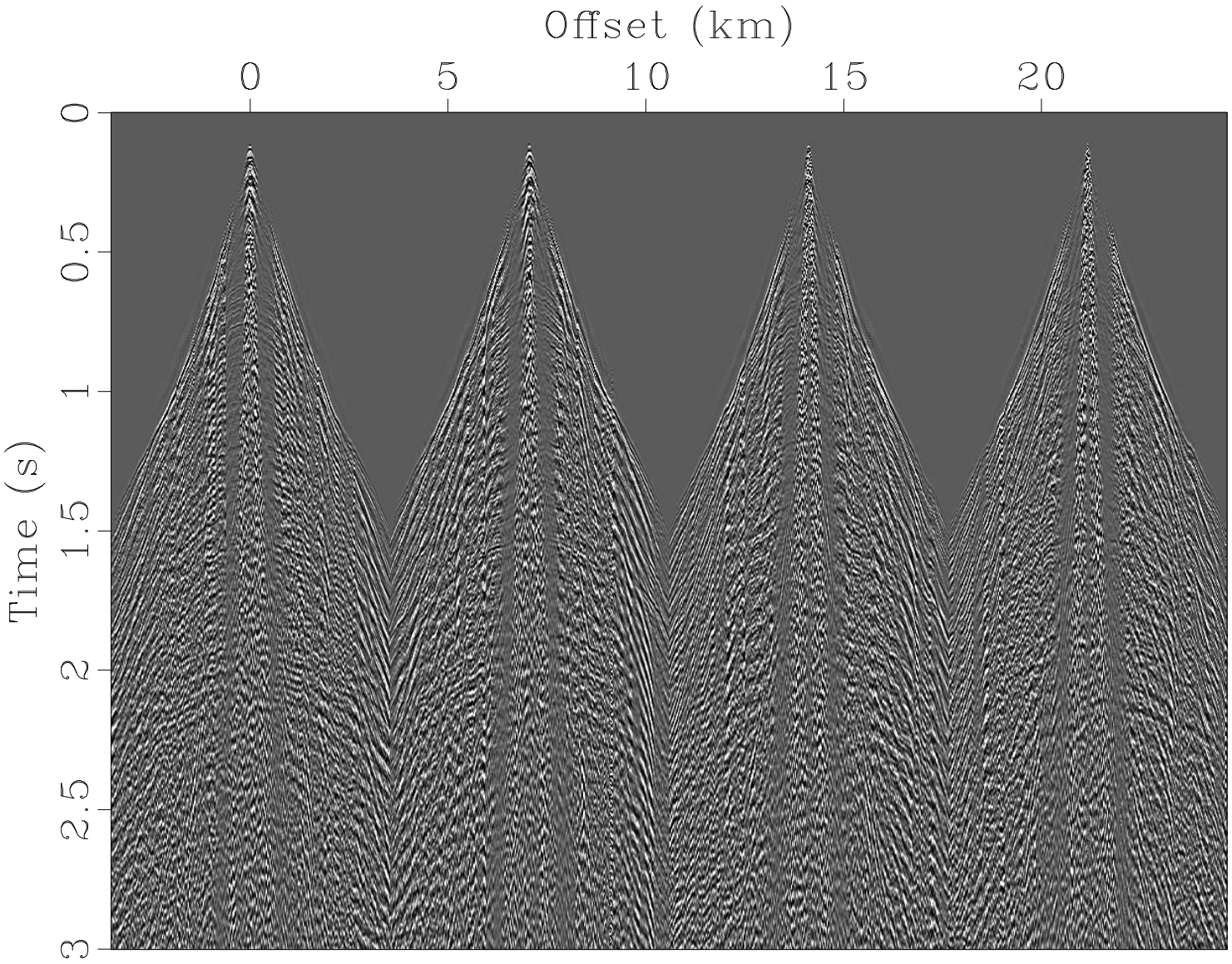
Input



First break mute

```
# Select muting parameter for background noise
Flow('outmute','inpmute',
     '''
     mutter t0=0.1 v0=5.2
     ''')
Result('outmute',
       '''
       put n2=1128 n3=1 |
       agc rect1=50 rect2=20 | grey wanttitle=n
       ''')
```

Output



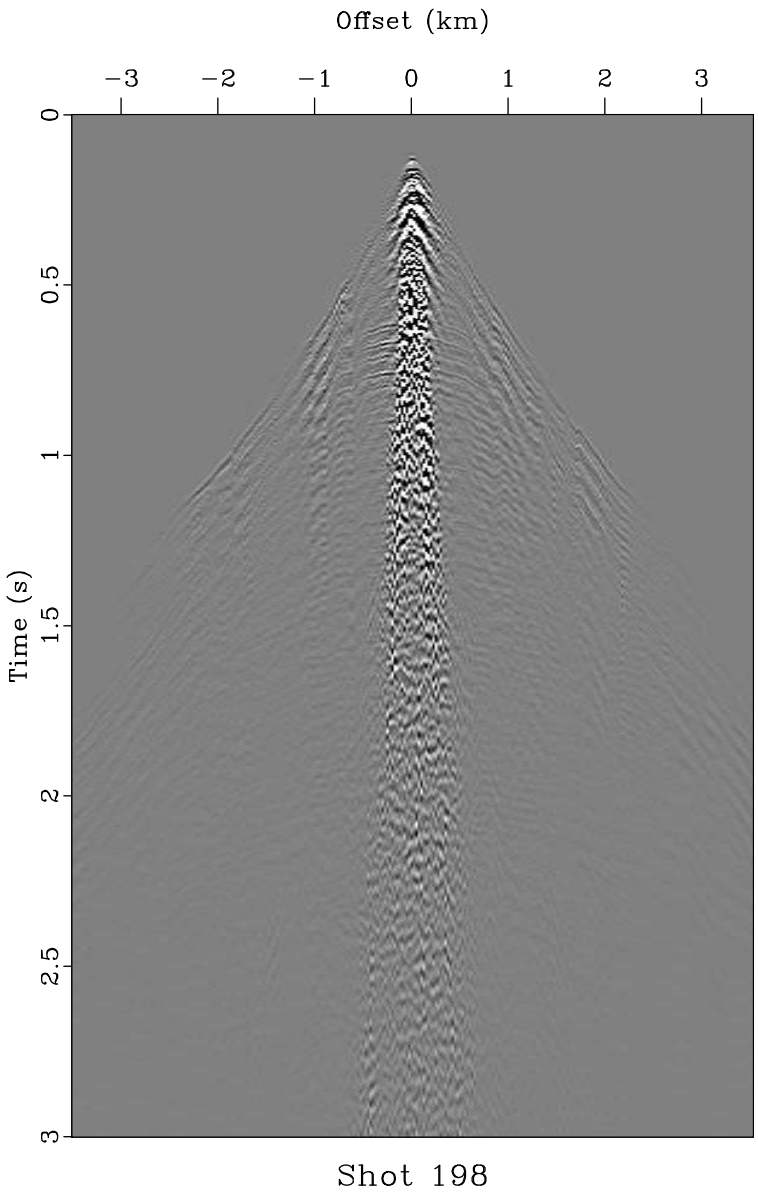
Apply muting on all shots

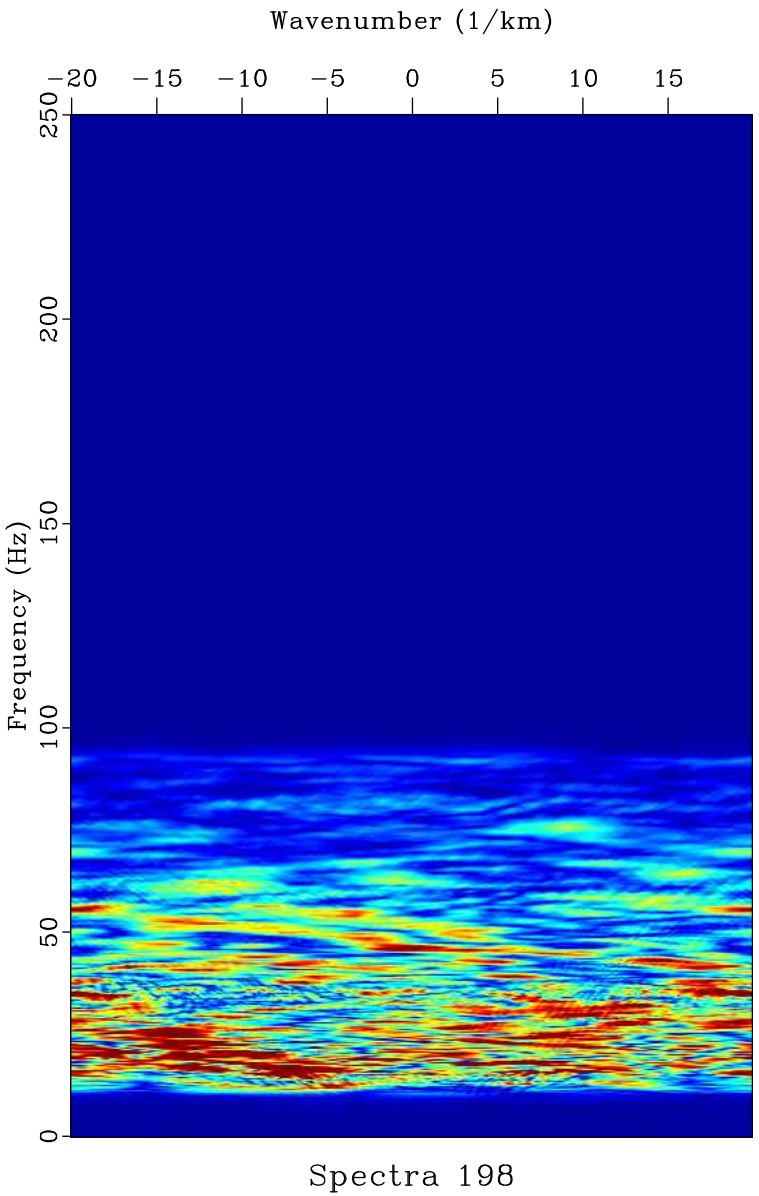
```
# First muting for all shots
Flow('mutes','lines',
     '''
     transp memsize=1000 plane=23 |
     mutter t0=0.1 v0=5.2 |
     transp memsize=1000 plane=23
     ''' )
```

```
[yourcomputer directory]$ sfin lines.rsf
lines.rsf:
    in="$RSFDATA/.../.../lines.rsf@"
    esize=4 type=float form=native
    n1=1501        d1=0.002       o1=0          label1="Time" unit1="s"
    n2=251         d2=0.05        o2=688        label2="Source" unit2="km"
    n3=282         d3=0.025       o3=-3.5       label3="Offset" unit3="km"
106243782 elements 424975128 bytes
```

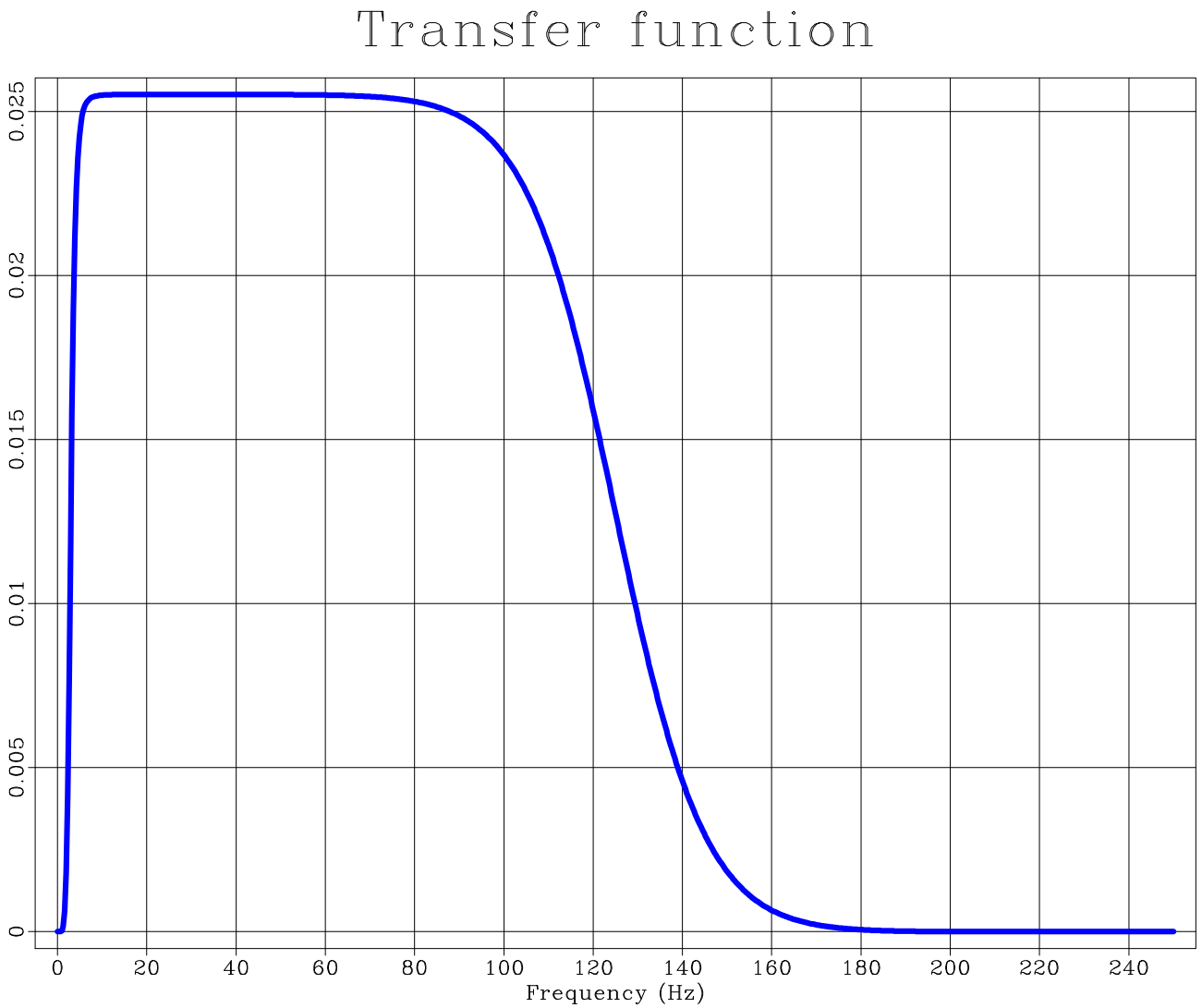
```
# Shot 198
Flow('shot198','mutes','window n2=1 f2=198')
Plot('shot198','grey title="Shot 198" labelfat=4 titlefat=4')

# Spectra
Flow('spec198','shot198','spectra2')
Plot('spec198',
     '''
     grey color=j yreverse=n title="Spectra 198" bias=0.08
     label1=Frequency unit1=Hz label2=Wavenumber unit2=1/km
     labelfat=4 titlefat=4
     ''')
Result('input198','shot198 spec198','SideBySideAniso')
```

Input


Offset (km)
−3
−2
−1
0
1
2
3
Time (s)
0
0.5
1
1.5
2
2.5
3
Shot 198


Wavenumber (1/km)
−20
−15
−10
−5
0
5
10
15
Frequency (Hz)
250
200
150
100
50
0
Spectra 198

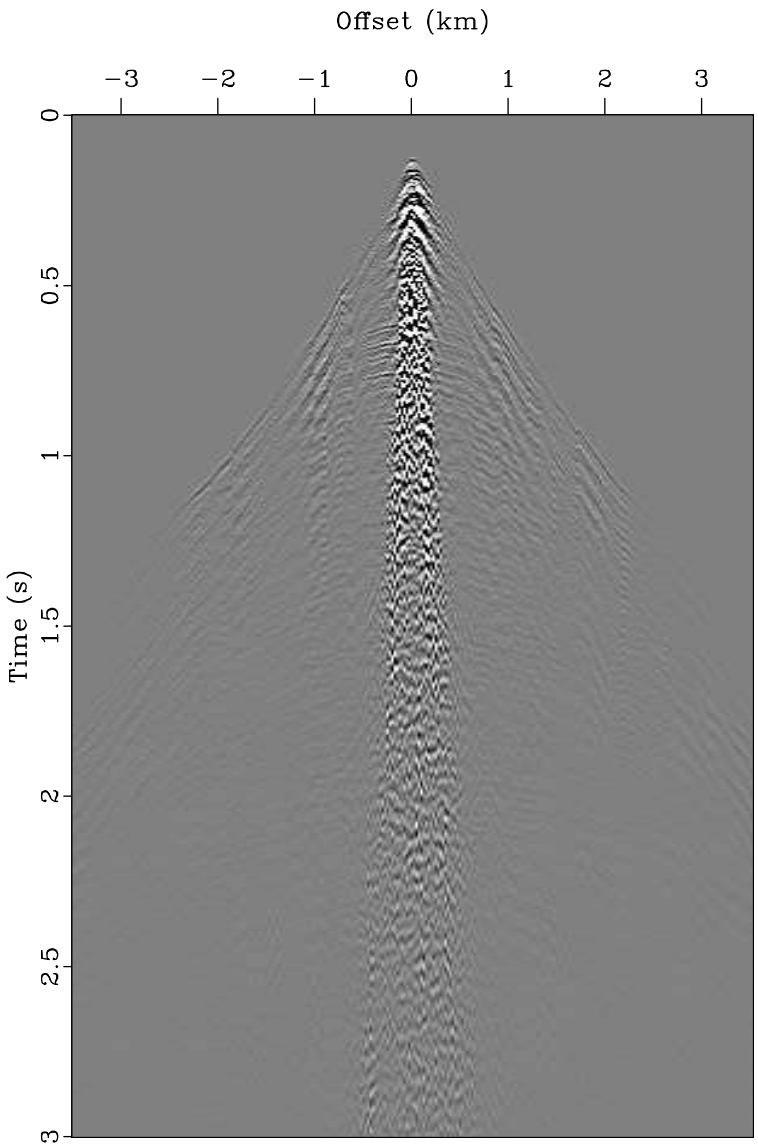
```
# Test anti-aliasing filter
Flow('spike',None,'spike n1=1501 d1=0.002 o1=0 k1=750 mag=1 nsp=1')
Flow('bandp','spike','bandpass flo=3 fhi=125 nphi=8 ')
Result('sbandp','bandp',
       '''
       spectra |
       graph title="Transfer function"
       labelsz=4. plotfat=10 grid=y
       ''')
```


Transfer function
0.025
0.02
0.015
0.01
0.005
0
0
20
40
60
80
100
120
140
160
180
200
220
240
Frequency (Hz)

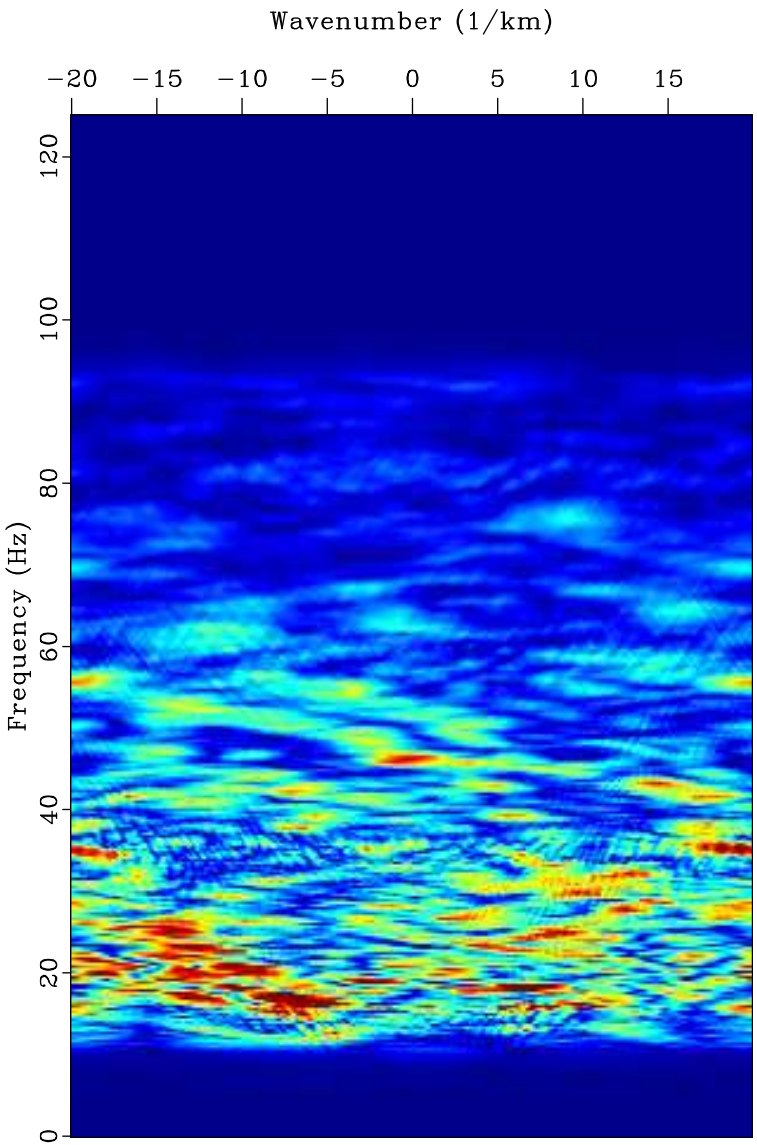
Subsampling – output

```
# Subsampling all shots to 4ms
Flow('subsample','mutes',
     'bandpass flo=3 fhi=125 nphi=8 | window j1=2')
Flow('subshot198','subsample','window n2=1 f2=198')
Plot('subshot198','grey title="Subsampled 198" labelfat=4 titlefat=4')
# Spectra
Flow('subspec198','subshot198','spectra2')
Plot('subspec198',
     '''
     grey color=j yreverse=n title="Spectra 198" bias=0.08
     label1=Frequency unit1=Hz label2=Wavenumber unit2=1/km
     labelfat=4 titlefat=4
     ''')
Result('output198','subshot198 subspec198','SideBySideAniso')
```

Output



Subsampled 198

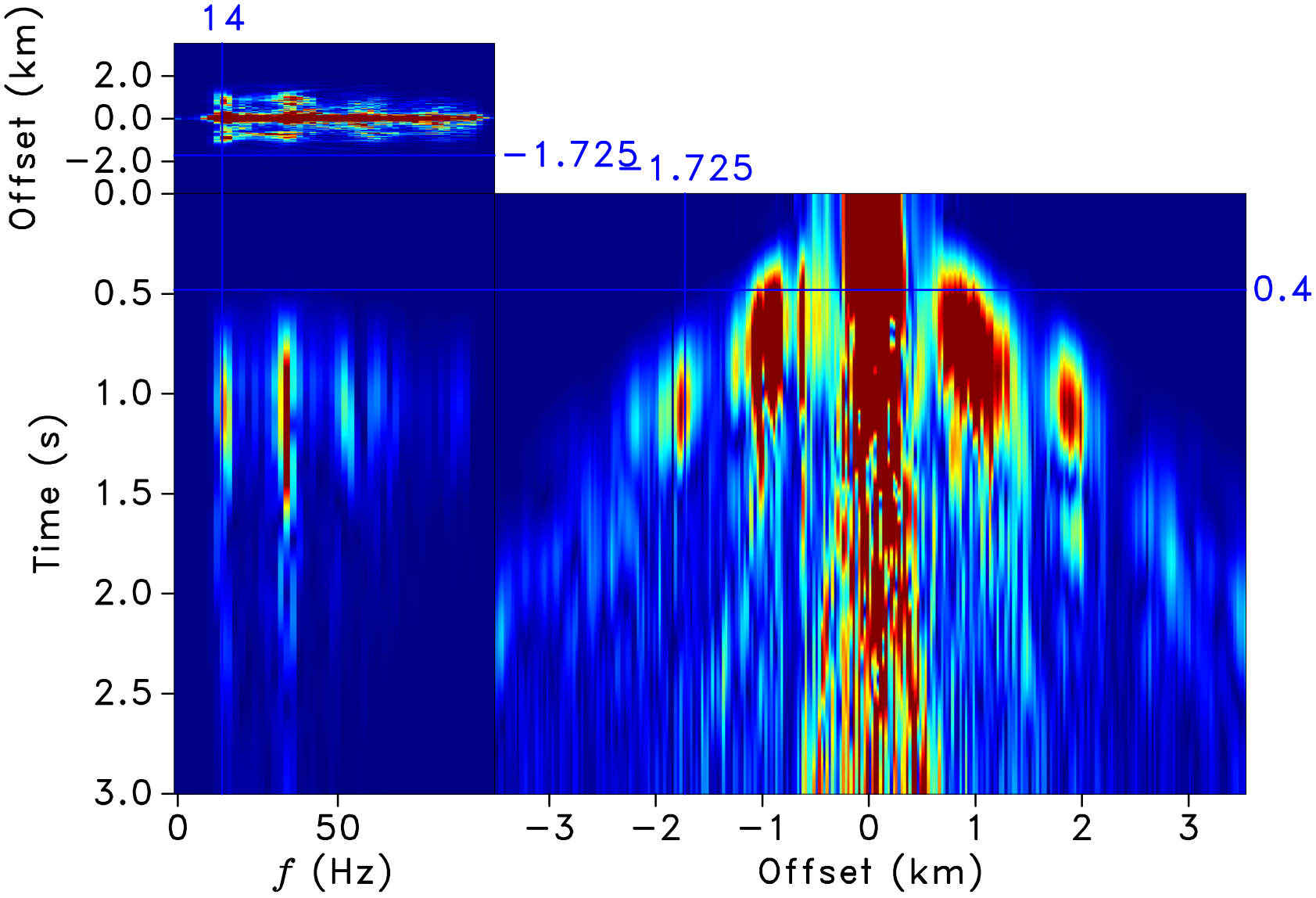


Spectra 198

Calculate time-frequency spectra

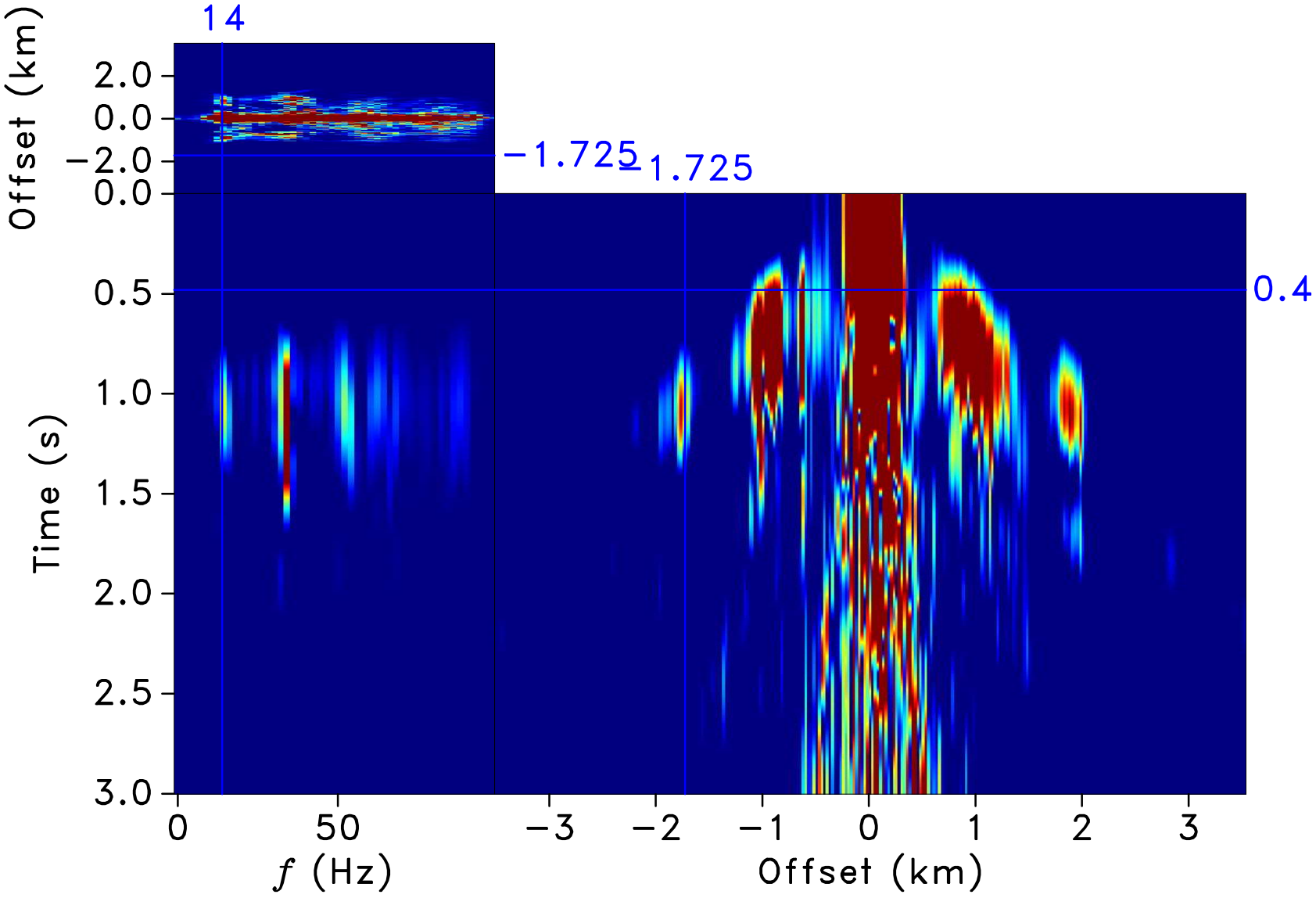
```
Flow('ltft198','subshot198',
     '''
     ltft rect=20 verb=n nw=50 dw=2 niter=50
     ''')
Result('ltft198',
       '''
       math output="abs(input)" | real |
       byte allpos=y gainpanel=100 pclip=99 |
       grey3 color=j  frame1=120 frame2=7 frame3=71 label1=Time flat=y
       unit1=s label3=Offset label2="\F5 f \F-1" unit3=km
       screenht=10 screenratio=0.7 parallel2=n format2=%3.1f
       point1=0.8 point2=0.3 wanttitle=n labelfat=4 font=2 titlefat=4
       ''')
```

Local time-frequency (LTF) spectra



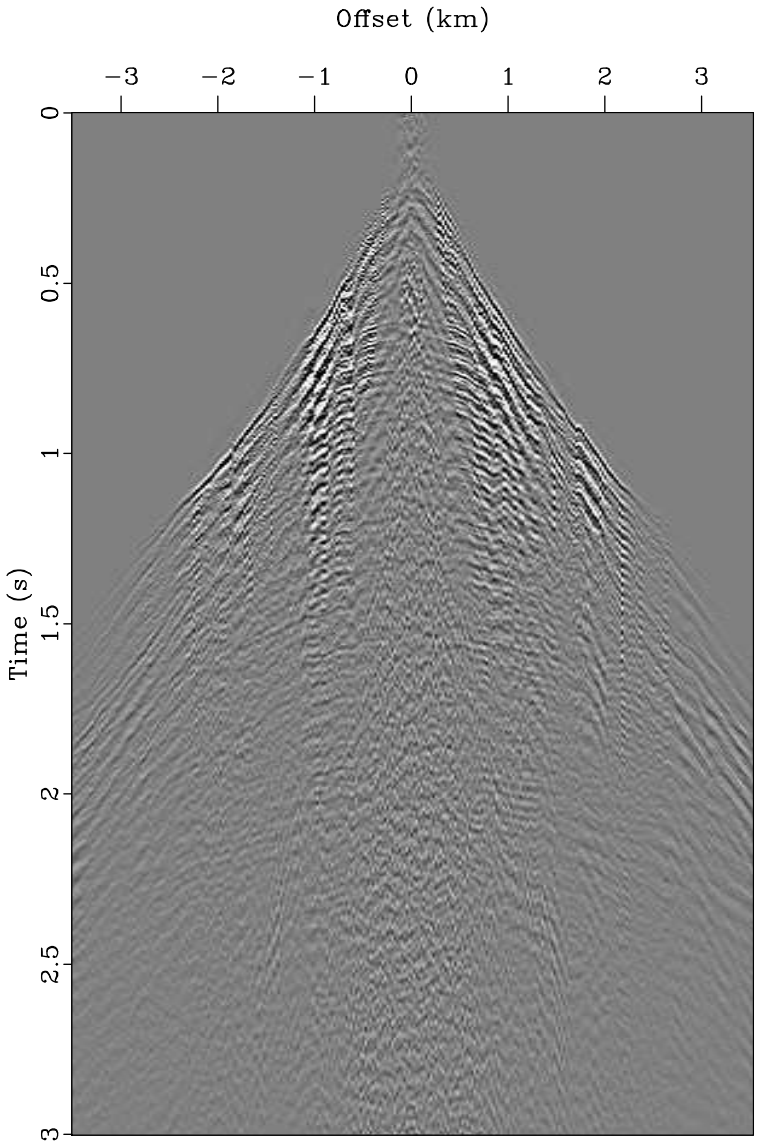
```
Flow('thr198','ltft198',
     '''
     transp plane=23 memsize=1000 |
     threshold2 pclip=25 verb=y |
     transp plane=23 memsize=1000
     ''')
Result('thr198',
       '''
       math output="abs(input)" | real |
       byte allpos=y gainpanel=100 pclip=99 |
       grey3 color=j  frame1=120 frame2=7 frame3=71 label1=Time flat=y
       unit1=s label3=Offset label2="\F5 f \F-1" unit3=km
       screenht=10 screenratio=0.7 parallel2=n format2=%3.1f
       point1=0.8 point2=0.3 wanttitle=n labelfat=4 font=2 titlefat=4
       ''')
```

Thresholded LTF spectra

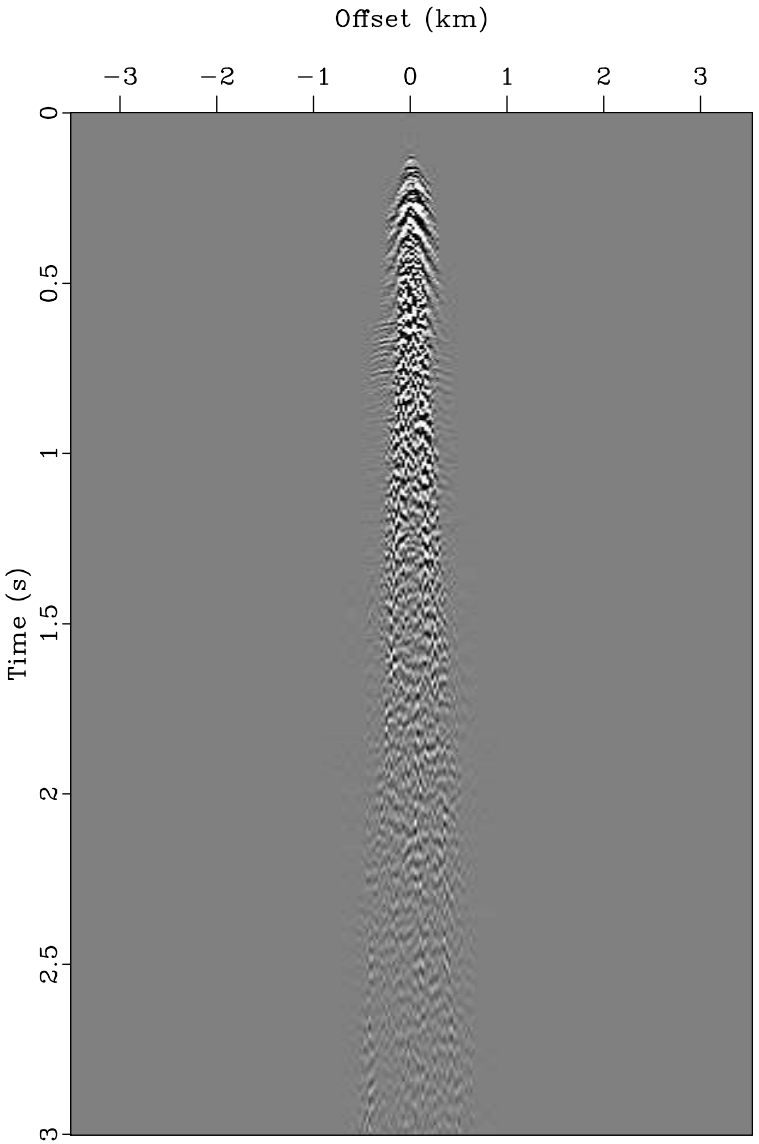


```
Flow('noise198','thr198','ltft inv=y | mutter t0=-0.5 v0=0.7')
Plot('noise198',
     'grey title="Ground-roll 198" unit2=km labelfat=4 titlefat=4')

Flow('signal198','subshot198 noise198','add scale=1,-1 ${SOURCES[1]}')
Plot('signal198',
     'grey title="Ground-roll removal" labelfat=4 titlefat=4')
Result('sn198','signal198 noise198','SideBySideAniso')
```

Signal and noise separation



Ground−roll removal



Ground−roll 198

Process the whole dataset

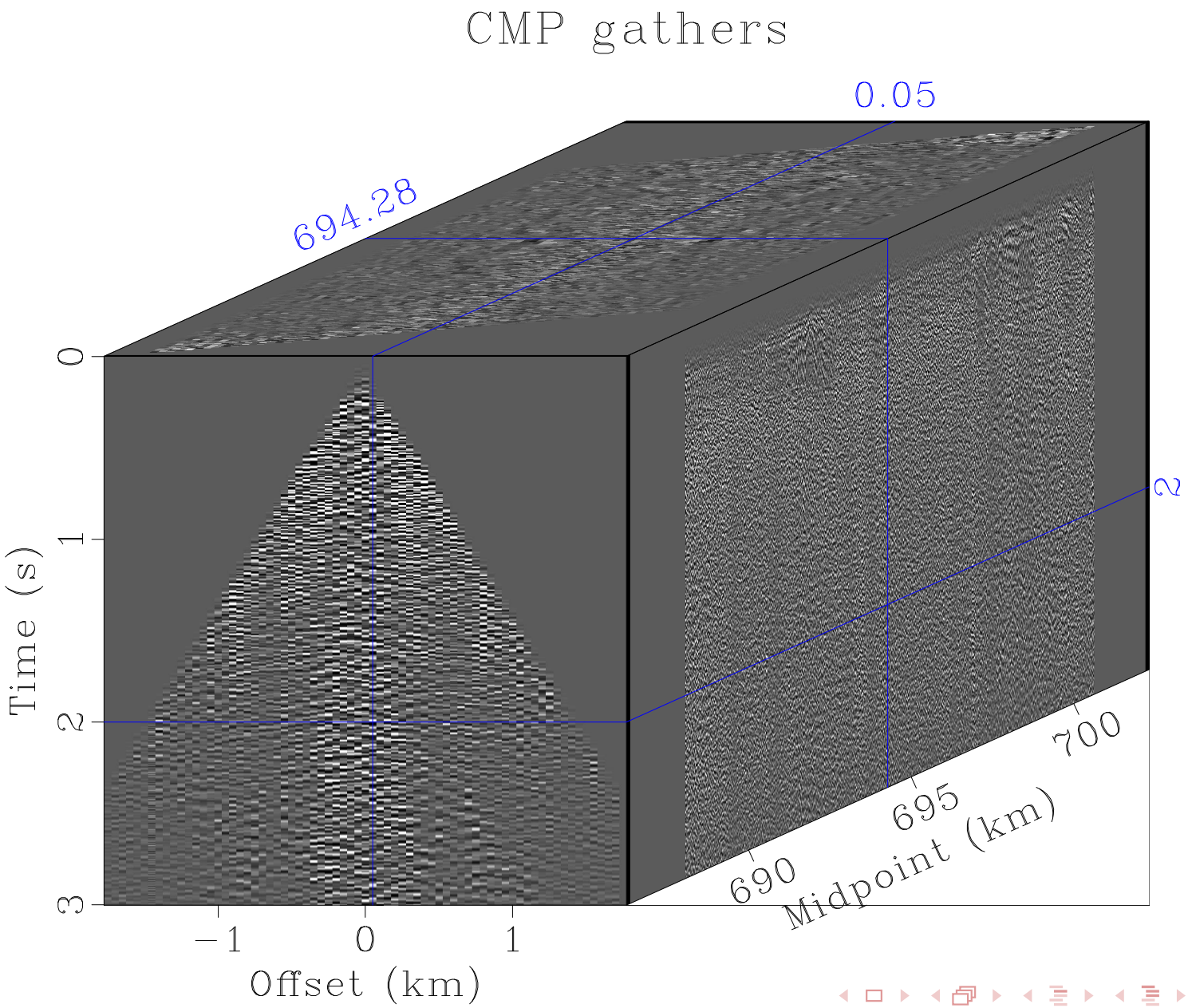
```
Flow('ltfts','subsample',
     '''
     ltft rect=20 verb=y nw=50 dw=2 niter=50
     ''',split=[3,282],reduce="cat axis=4")
Flow('thresholds','ltfts',
     '''
     transp plane=24 memsize=1000 | threshold2 pclip=25 verb=y  |
     transp plane=24 memsize=1000
     ''',split=[3,251])
Flow('noises','thresholds',
     '''
     ltft inv=y | transp plane=23 memsize=1000 |
     mutter t0=-0.5 v0=0.7 | transp plane=23 memsize=1000
     ''')
Flow('signals','subsample noises','add scale=1,-1 ${SOURCES[1]}')
```

Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

Convert shots to CMPs

```
Flow('cmps','signals',
     '''
     transp memsize=1000 plane=23 |
     mutter v0=3. |
     shot2cmp half=n | put o2=-1.75 label2="Half-offset"
     ''')
Result('cmps',
       '''
       byte gainpanel=each |
       grey3 frame1=500 frame2=36 frame3=642 flat=n
       title="CMP gathers" point1=0.7 label2=Offset label3=Midpoint
       ''')
```


CMP gathers
0.05
694.28
2
Time (s)
0
1
2
3
−1
0
1
Offset (km)
690
695
700
Midpoint (km)

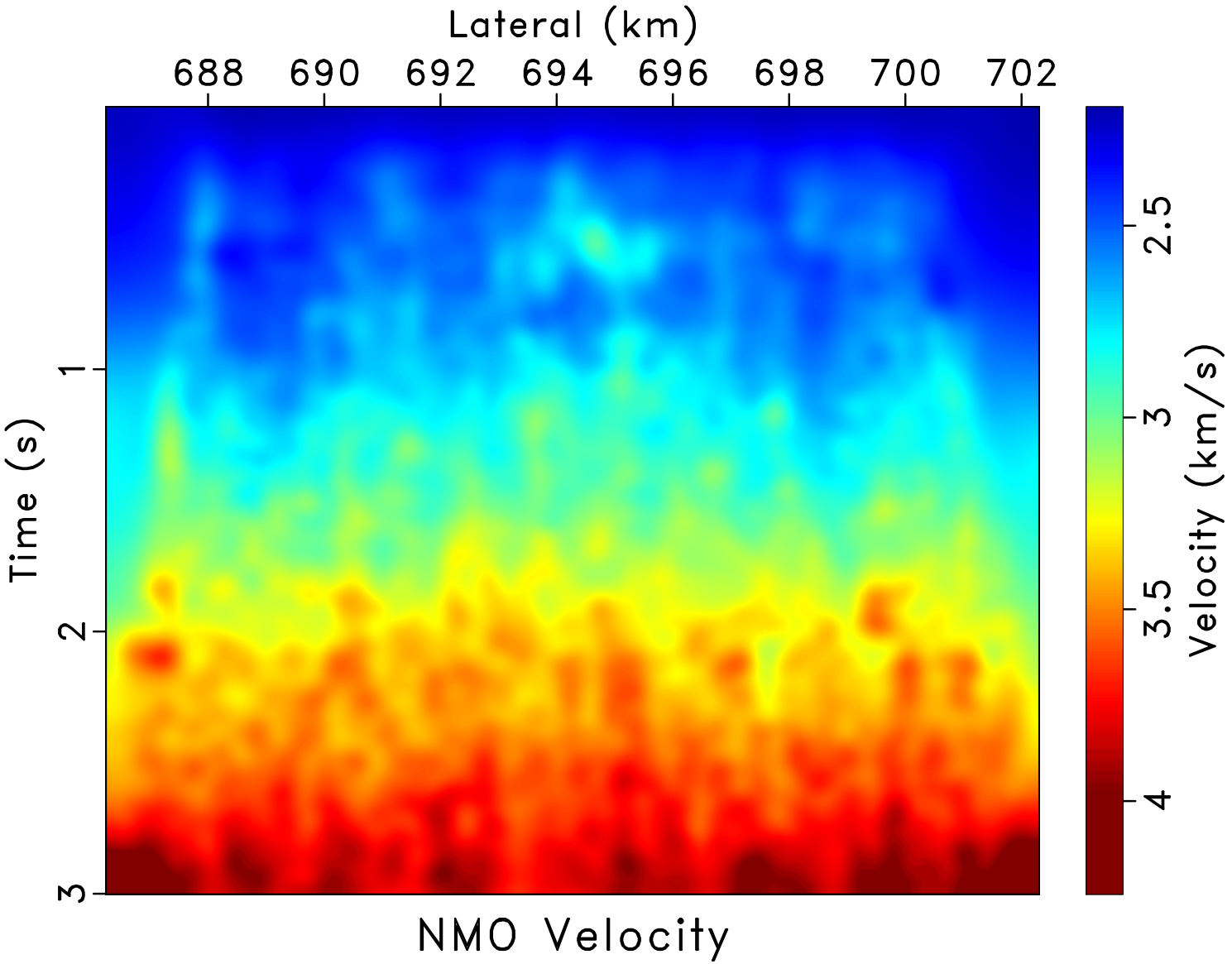
Velocity scan parameters

```
# Set up velocity scan parameters
v0 = 2.15
dv = 0.025
nv = 100
```

Velocity scan for all CMPs

```
# Velocity scanning for all CMP gathers
Flow('scn','cmps',
     '''
     vscan semblance=y v0=%g nv=%d dv=%g half=y str=0 |
     mutter v0=0.9 t0=-4.5 inner=y
     ''' % (v0,nv,dv),split=[3,1285])
Flow('vel','scn','pick rect1=15 rect2=25 gate=100 an=10 | window')
Result('vel',
       '''
       grey title="NMO Velocity" label1="Time" label2="Lateral"
       color=j scalebar=y allpos=y bias=2.1 barlabel="Velocity"
       barreverse=y o2num=1 d2num=1 n2tic=3 labelfat=4 font=2 titlefat=4
       ''')
```

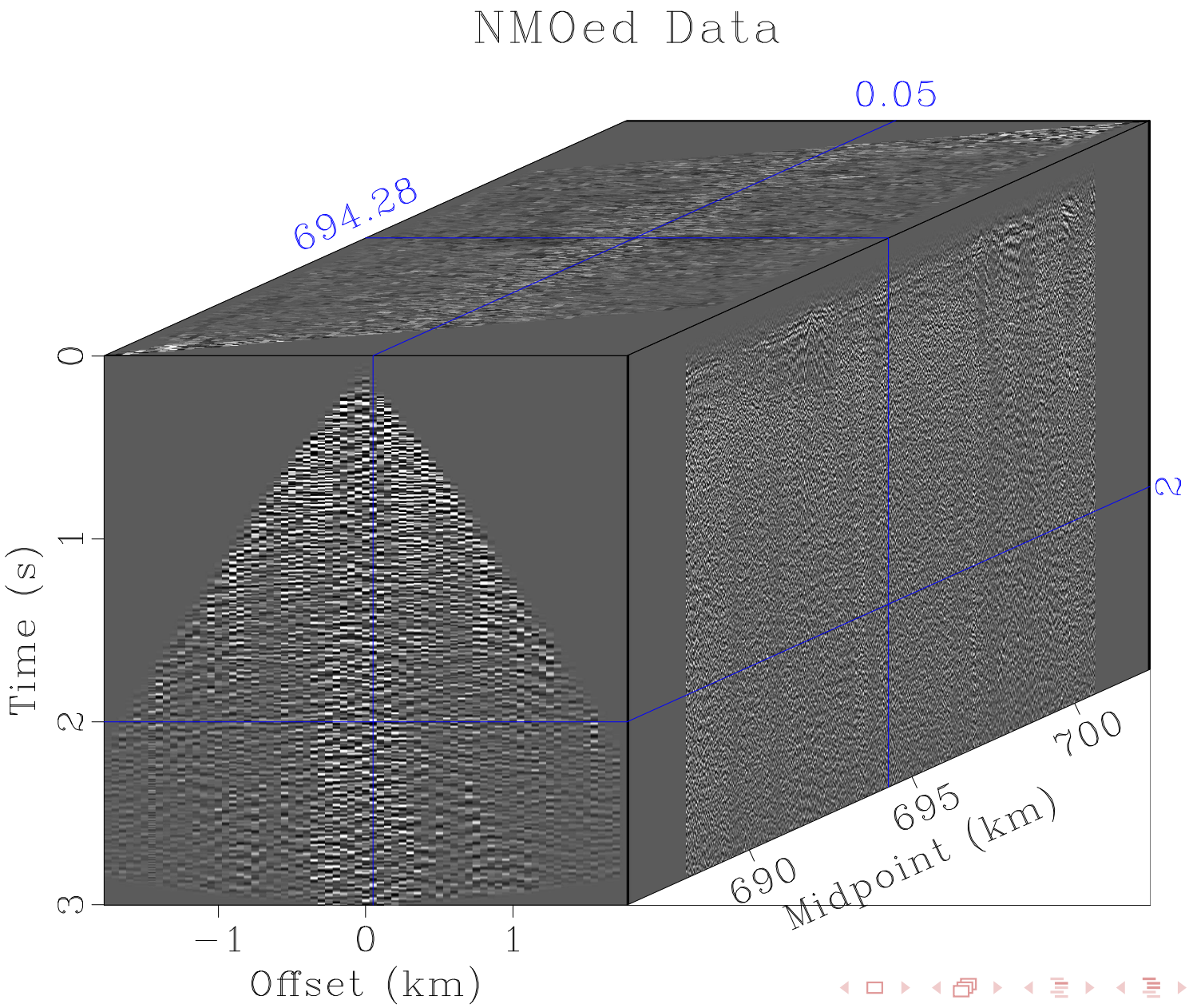
Display NMO velocity



Outline

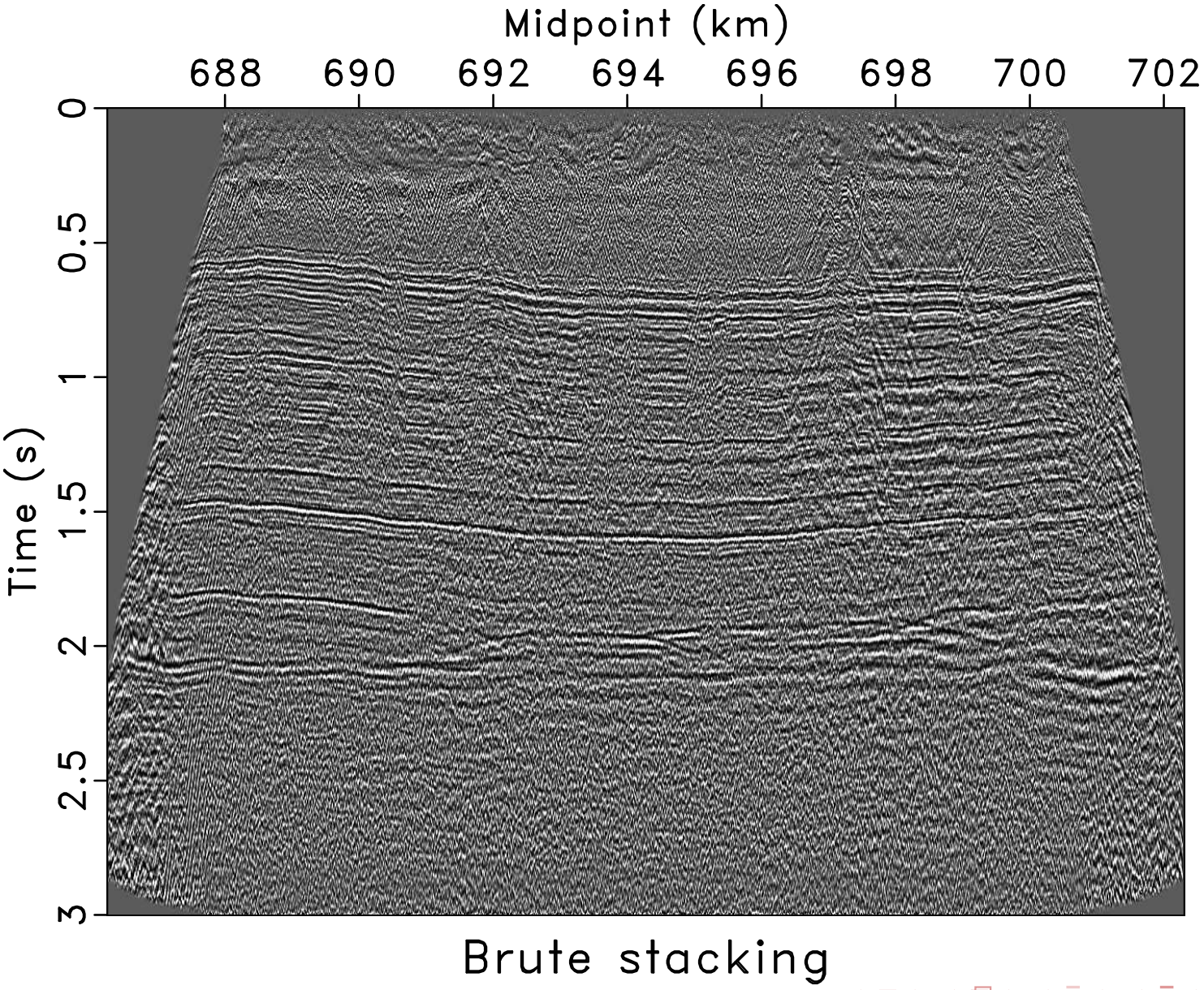
1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

```
Flow('nmo','cmps vel',
     '''
     nmo velocity=${SOURCES[1]} half=y
     ''')
Result('nmo',
       '''
       byte gainpanel=e |
       grey3 frame1=500 frame2=36 frame3=642 flat=n
       title="NMOed Data" point1=0.7
       label2=Offset label3=Midpoint
       ''')
```


NMOed Data
0.05
694.28
2
0
1
2
3
Time (s)
−1
0
1
Offset (km)
690
695
700
Midpoint (km)

```
# Brute stacking
Flow('bstack','nmo','stack')
Result('bstack',
       '''
       agc rect1=50 |
       grey title="Brute stacking" labelfat=4 font=2 titlefat=4
       ''')
```

Display brute stacking result

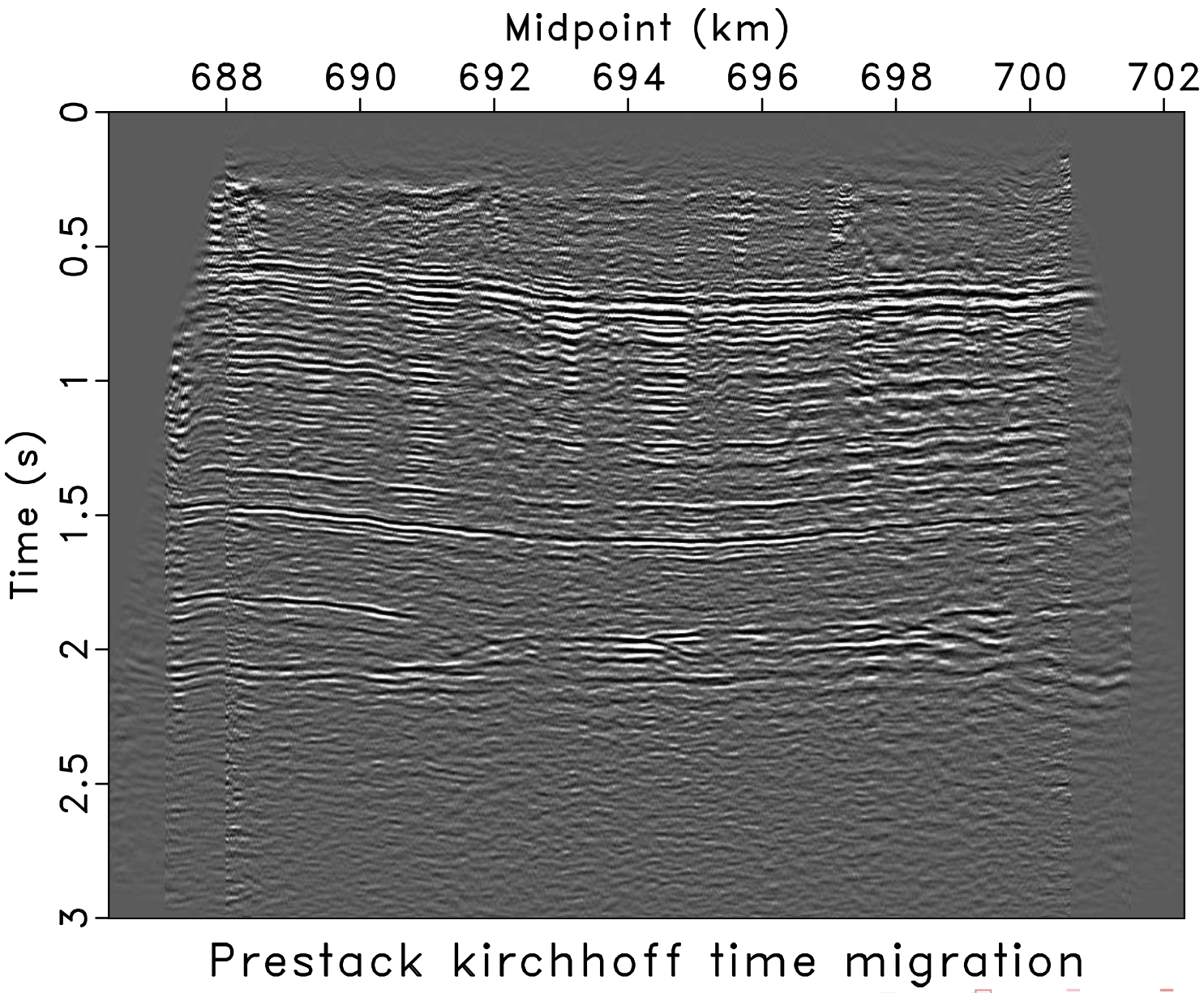


Outline

1. Assignment
2. Simple workflow
   - Conversion to RSF format
   - Initial data checking
   - Initial signal analysis and quality control
   - Initial velocity analysis
   - Brute stacking
   - Toy migration

```
# Prestack Kirchhoff time migration
Flow('tcmps','cmps','transp memsize=1000 plane=23')
Flow('pstm','tcmps vel',
     '''
     mig2 vel=${SOURCES[1]} apt=5 antialias=1
     ''',split=[3,71,[0]],reduce='add')
Result('pstm',
       '''
       grey title="Prestack kirchhoff time migration"
       labelfat=4 font=2 titlefat=4
       ''')

End()
```

Display migration result



To be continued ...

Modify/add your own modules, improve the final result ...

Software

MADAGASCAR

http://www.ahay.org

```
cd $RSFSRC/book/rsf/usp/data/
pscons lock
cd $RSFSRC/book/rsf/usp/
scons school10.read
```

Thanks for attention and have a fun!