Auburn Drumline

Statistical Analysis for Marching Band Drum Core Improvement

Daniel Willis

O'R
O'REILLY
AUBURN
AUBURN
Joyent
WARRIORS
Occupancy
607
TAMA.
Peavey
Peavey

Bio

- 15 Year old freshman in High School
- I’m in the Marching Band and Concert Band
- Playing drums for 5 years
- I also play rock N Roll Drums with my father
- I have been to SCALE 5 times and spoke the past 3 years
- Professionally speaking since I was 12
- Spoken at Dockercon, O’Reilly Velocity and multiple Devopsdays

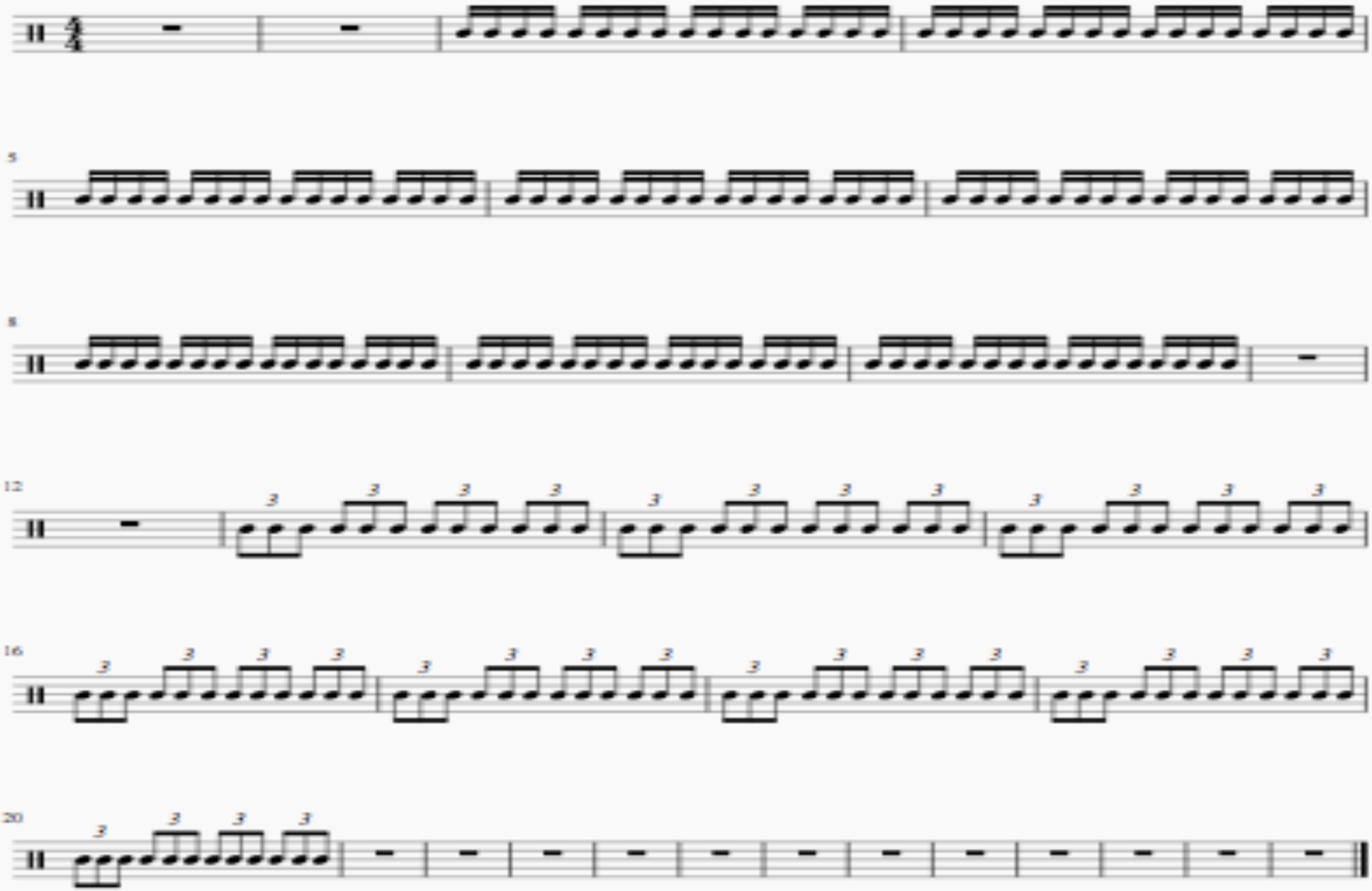
Outline

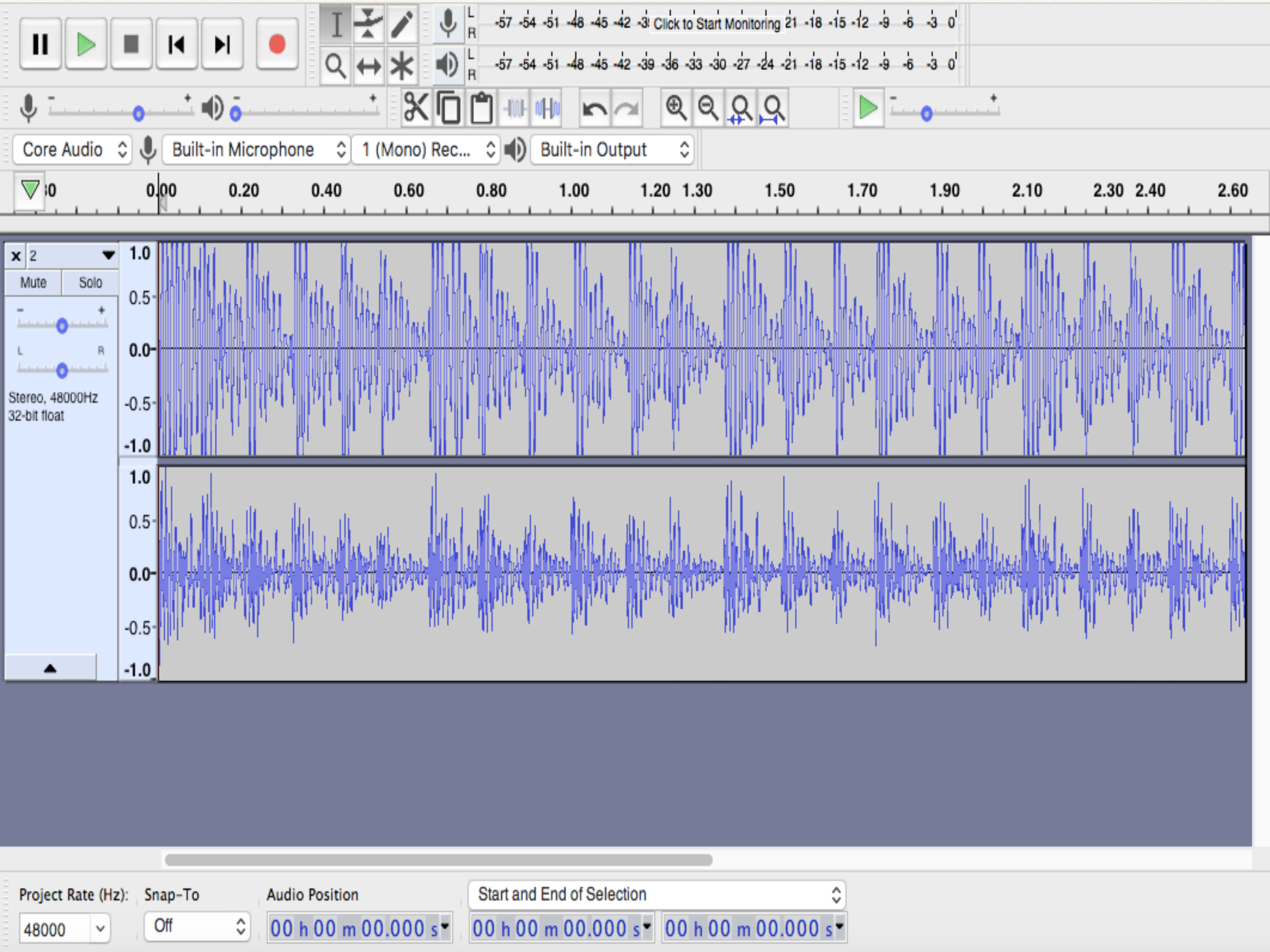
- The Idea
- Demonstration (recording a diddle exercise)
- Audacity
- History
- Digital Signal Analysis
- Statistical Analysis (Improvement)
- Summary (What I learned)

The Idea

- Record the same diddle sequence over a three month period.
- Use audio tools to extract timings, tempo and beat signals.
- Use statistical tools to look for patterns of improvement over time.

base scale





Core Audio

Built-in Microphone

1 (Mono) Rec...

Built-in Output

Stereo, 48000Hz
32-bit float

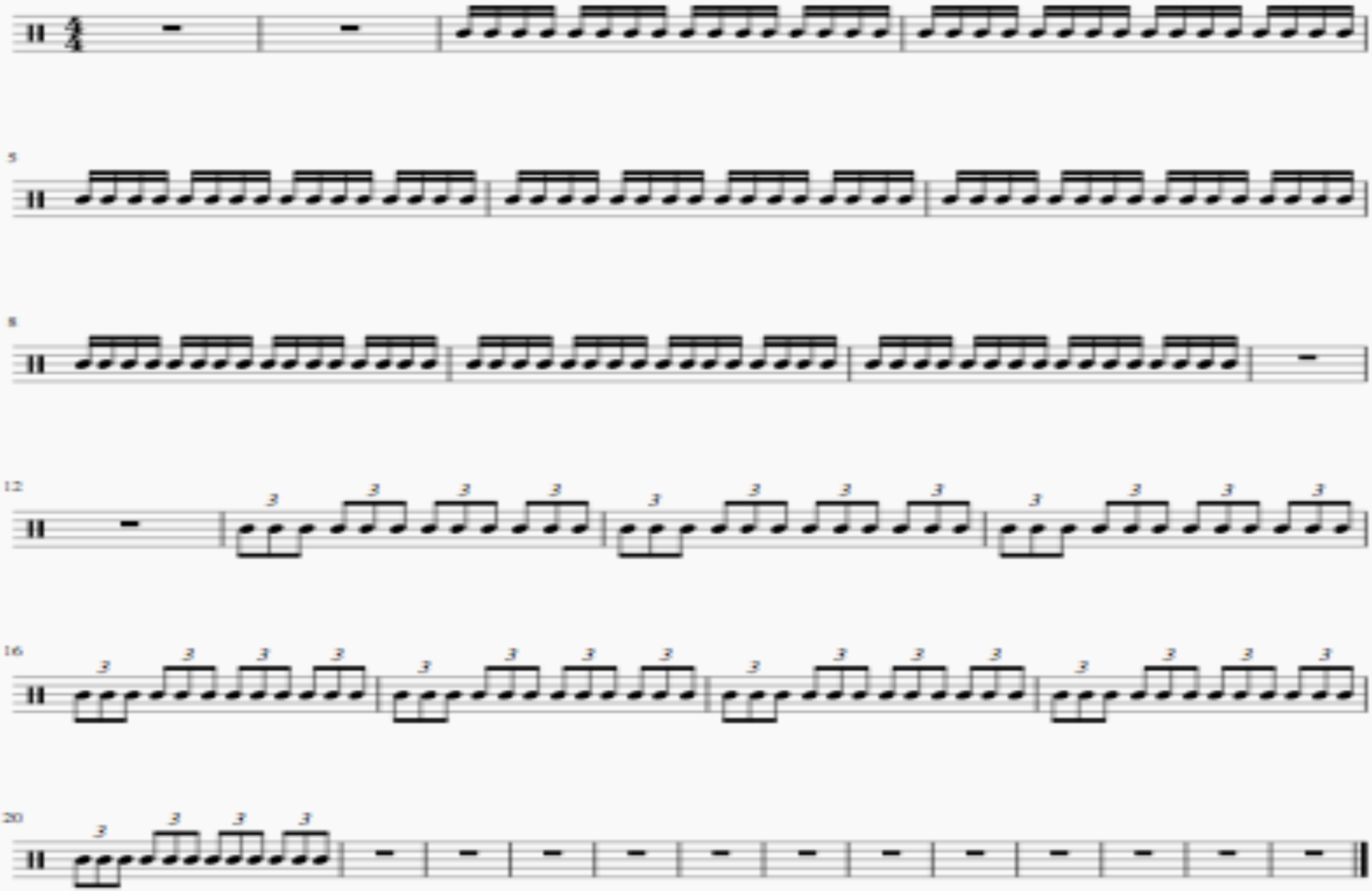
Project Rate (Hz): 48000

Snap-To: Off

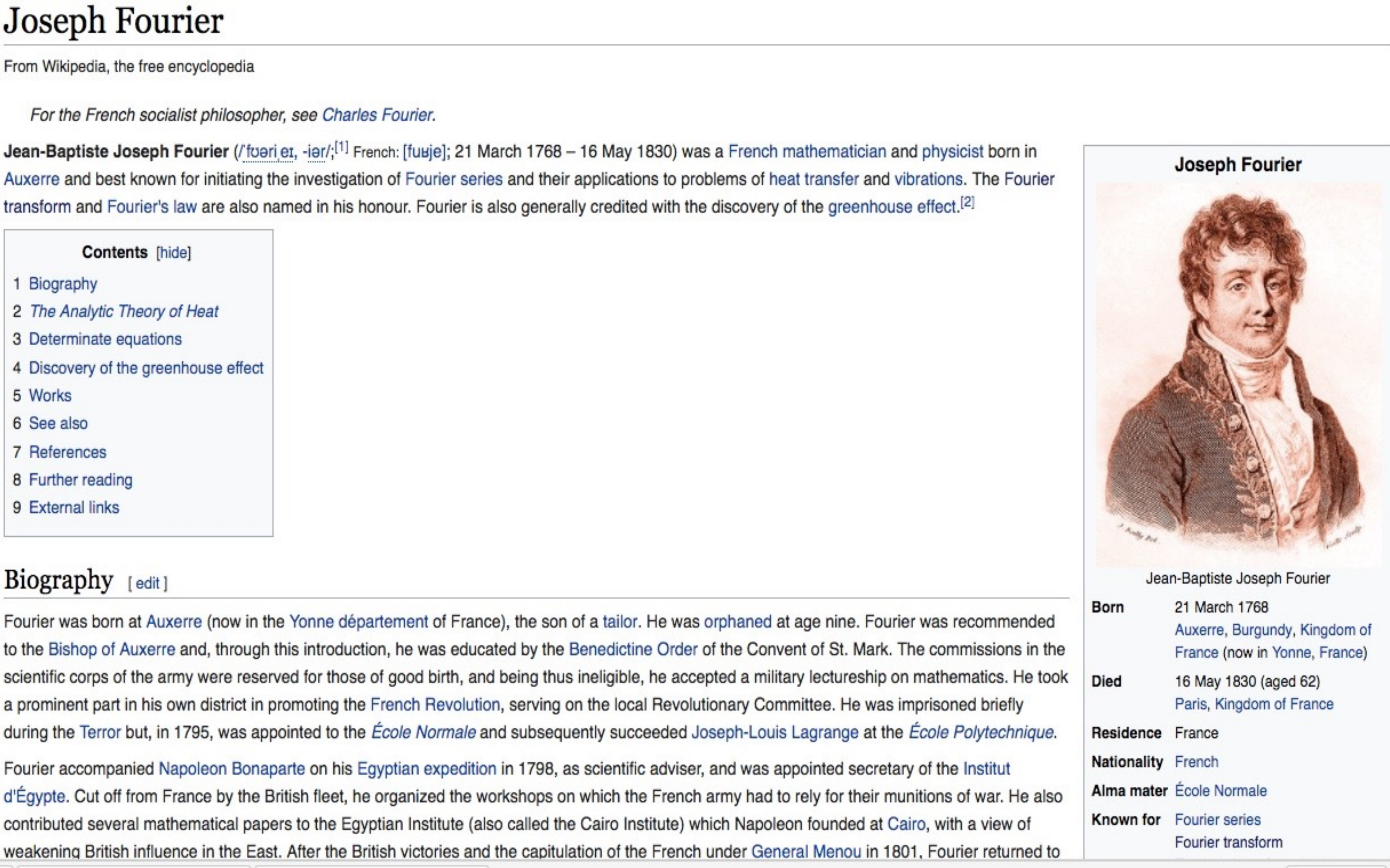
Audio Position: 00 h 00 m 00.000 s

Start and End of Selection: 00 h 00 m 00.000 s – 00 h 00 m 00.000 s

base scale



History



Joseph Fourier

From Wikipedia, the free encyclopedia

For the French socialist philosopher, see Charles Fourier.

Jean-Baptiste Joseph Fourier (/ˈfʊəri eɪ, -iər/;[1] French: [fuʁje]; 21 March 1768 – 16 May 1830) was a French mathematician and physicist born in Auxerre and best known for initiating the investigation of Fourier series and their applications to problems of heat transfer and vibrations. The Fourier transform and Fourier's law are also named in his honour. Fourier is also generally credited with the discovery of the greenhouse effect.[2]

Contents [hide]

1 Biography

2 *The Analytic Theory of Heat*

3 Determinate equations

4 Discovery of the greenhouse effect

5 Works

6 See also

7 References

8 Further reading

9 External links

Joseph Fourier

Jean-Baptiste Joseph Fourier

Born	21 March 1768 Auxerre, Burgundy, Kingdom of France (now in Yonne, France)
Died	16 May 1830 (aged 62) Paris, Kingdom of France
Residence	France
Nationality	French
Alma mater	École Normale
Known for	Fourier series Fourier transform

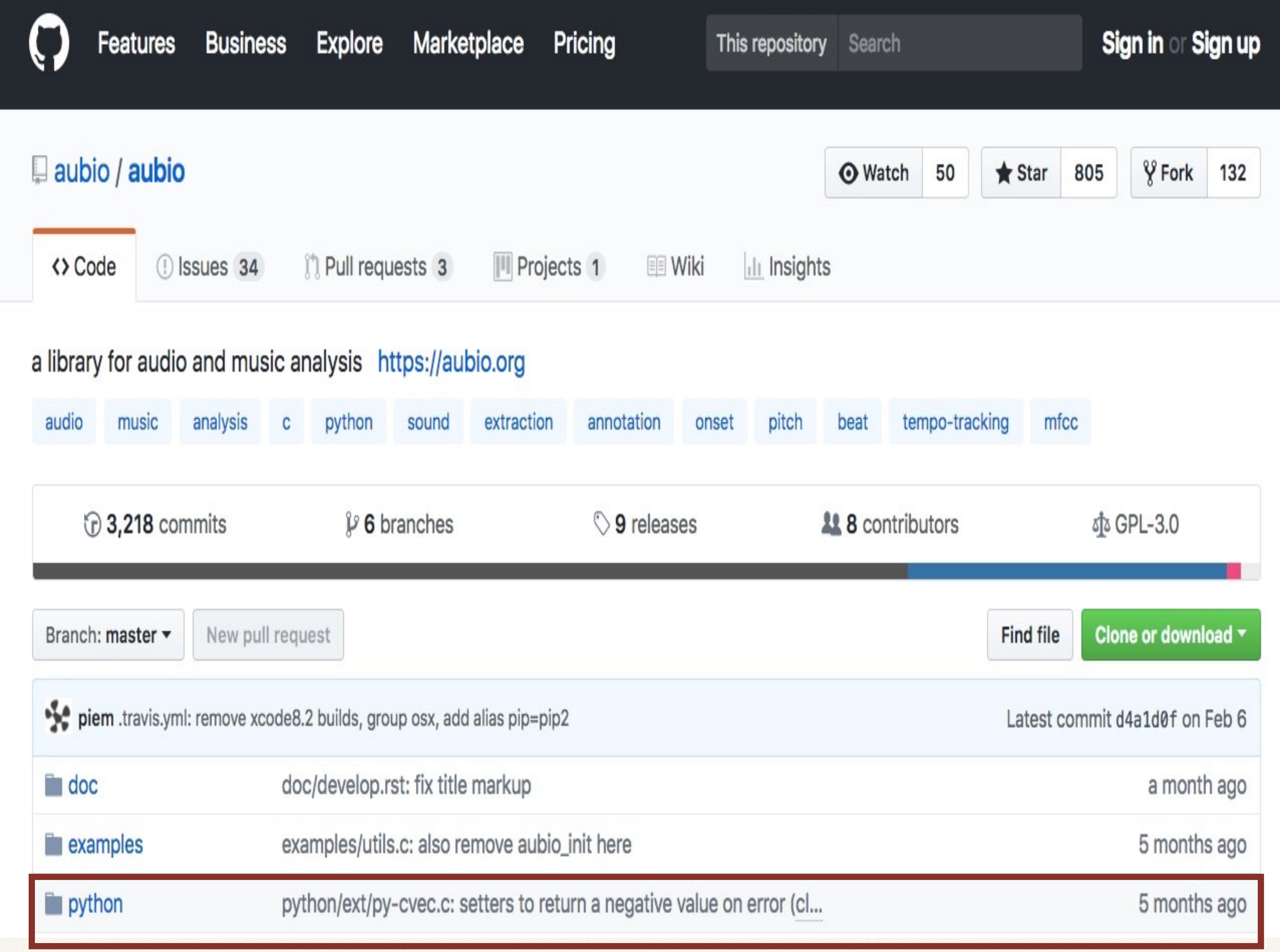
Biography [edit]

Fourier was born at Auxerre (now in the Yonne département of France), the son of a tailor. He was orphaned at age nine. Fourier was recommended to the Bishop of Auxerre and, through this introduction, he was educated by the Benedictine Order of the Convent of St. Mark. The commissions in the scientific corps of the army were reserved for those of good birth, and being thus ineligible, he accepted a military lectureship on mathematics. He took a prominent part in his own district in promoting the French Revolution, serving on the local Revolutionary Committee. He was imprisoned briefly during the Terror but, in 1795, was appointed to the *École Normale* and subsequently succeeded Joseph-Louis Lagrange at the *École Polytechnique*.

Fourier accompanied Napoleon Bonaparte on his Egyptian expedition in 1798, as scientific adviser, and was appointed secretary of the Institut d'Égypte. Cut off from France by the British fleet, he organized the workshops on which the French army had to rely for their munitions of war. He also contributed several mathematical papers to the Egyptian Institute (also called the Cairo Institute) which Napoleon founded at Cairo, with a view of weakening British influence in the East. After the British victories and the capitulation of the French under General Menou in 1801, Fourier returned to

Considered the Father of Digital Music

Digital Signal Analysis



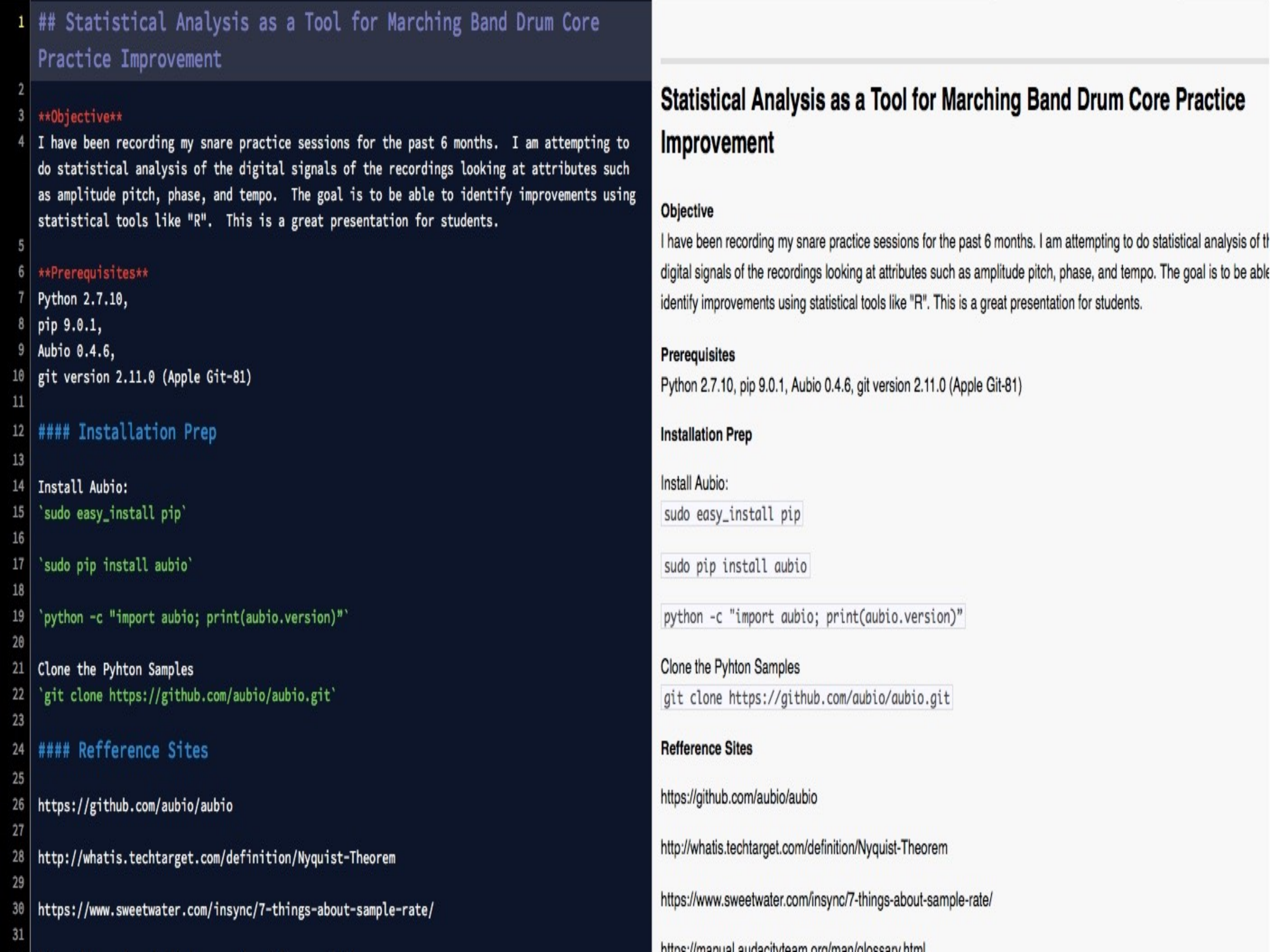
aubio / aubio

a library for audio and music analysis https://aubio.org

3,218 commits | 6 branches | 9 releases | 8 contributors | GPL-3.0

piem .travis.yml: remove xcode8.2 builds, group osx, add alias pip=pip2 Latest commit d4a1d0f on Feb 6

doc	doc/develop.rst: fix title markup	a month ago
examples	examples/utils.c: also remove aubio_init here	5 months ago
python	python/ext/py-cvec.c: setters to return a negative value on error (cl…	5 months ago



```
## Statistical Analysis as a Tool for Marching Band Drum Core Practice Improvement

**Objective**
I have been recording my snare practice sessions for the past 6 months.  I am attempting to do statistical analysis of the digital signals of the recordings looking at attributes such as amplitude pitch, phase, and tempo.  The goal is to be able to identify improvements using statistical tools like "R".  This is a great presentation for students.

**Prerequisites**
Python 2.7.10,
pip 9.0.1,
Aubio 0.4.6,
git version 2.11.0 (Apple Git-81)

#### Installation Prep

Install Aubio:
`sudo easy_install pip`

`sudo pip install aubio`

`python -c "import aubio; print(aubio.version)"`

Clone the Pyhton Samples
`git clone https://github.com/aubio/aubio.git`

#### Refference Sites

https://github.com/aubio/aubio

http://whatis.techtarget.com/definition/Nyquist-Theorem

https://www.sweetwater.com/insync/7-things-about-sample-rate/

```

Statistical Analysis as a Tool for Marching Band Drum Core Practice Improvement

Objective
I have been recording my snare practice sessions for the past 6 months. I am attempting to do statistical analysis of th digital signals of the recordings looking at attributes such as amplitude pitch, phase, and tempo. The goal is to be able identify improvements using statistical tools like "R". This is a great presentation for students.

Prerequisites
Python 2.7.10, pip 9.0.1, Aubio 0.4.6, git version 2.11.0 (Apple Git-81)

Installation Prep

Install Aubio:
`sudo easy_install pip`

`sudo pip install aubio`

`python -c "import aubio; print(aubio.version)"`

Clone the Pyhton Samples
`git clone https://github.com/aubio/aubio.git`

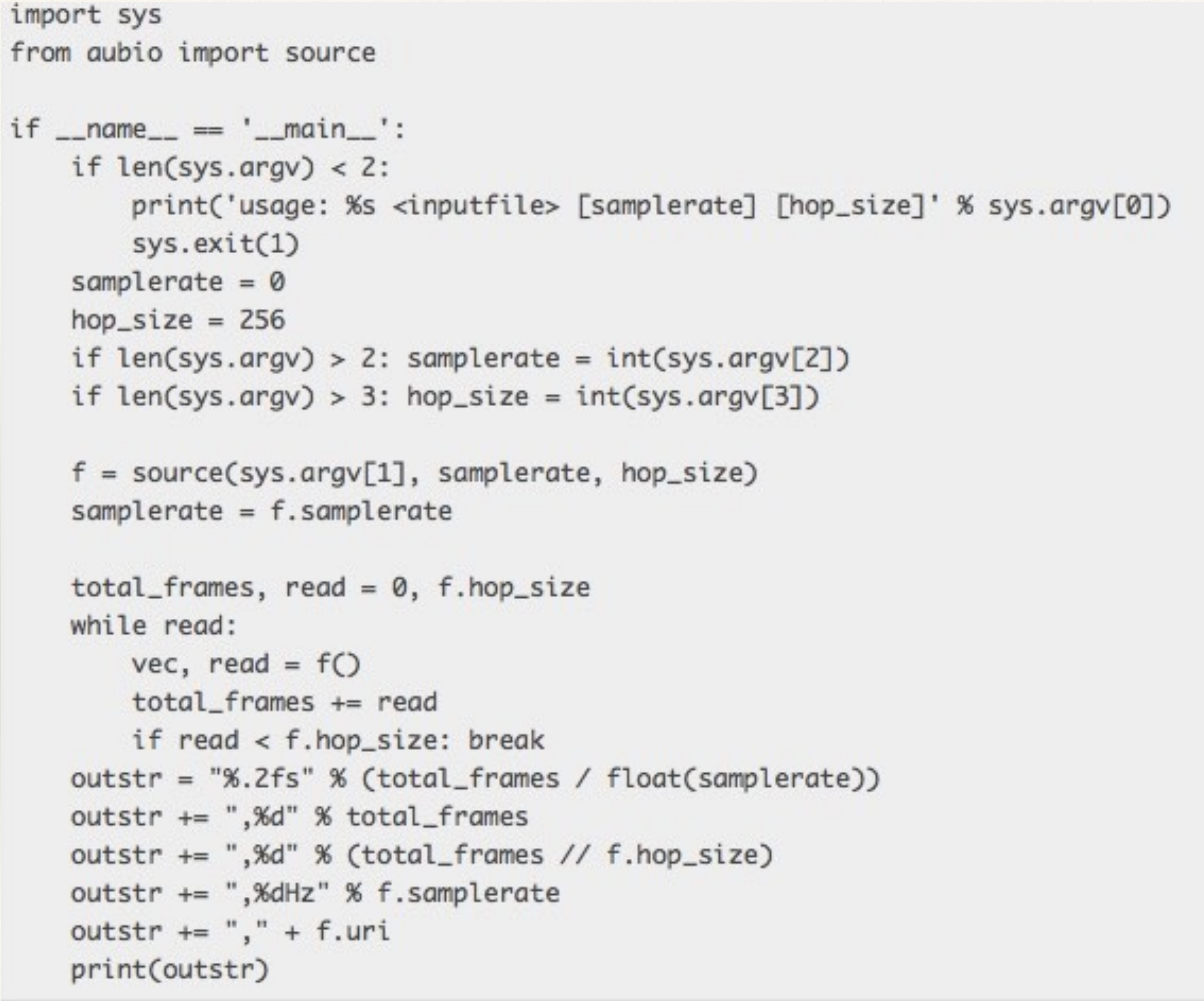
Refference Sites

https://github.com/aubio/aubio

http://whatis.techtarget.com/definition/Nyquist-Theorem

https://www.sweetwater.com/insync/7-things-about-sample-rate/

https://manual.audacityteam.org/man/glossary.html

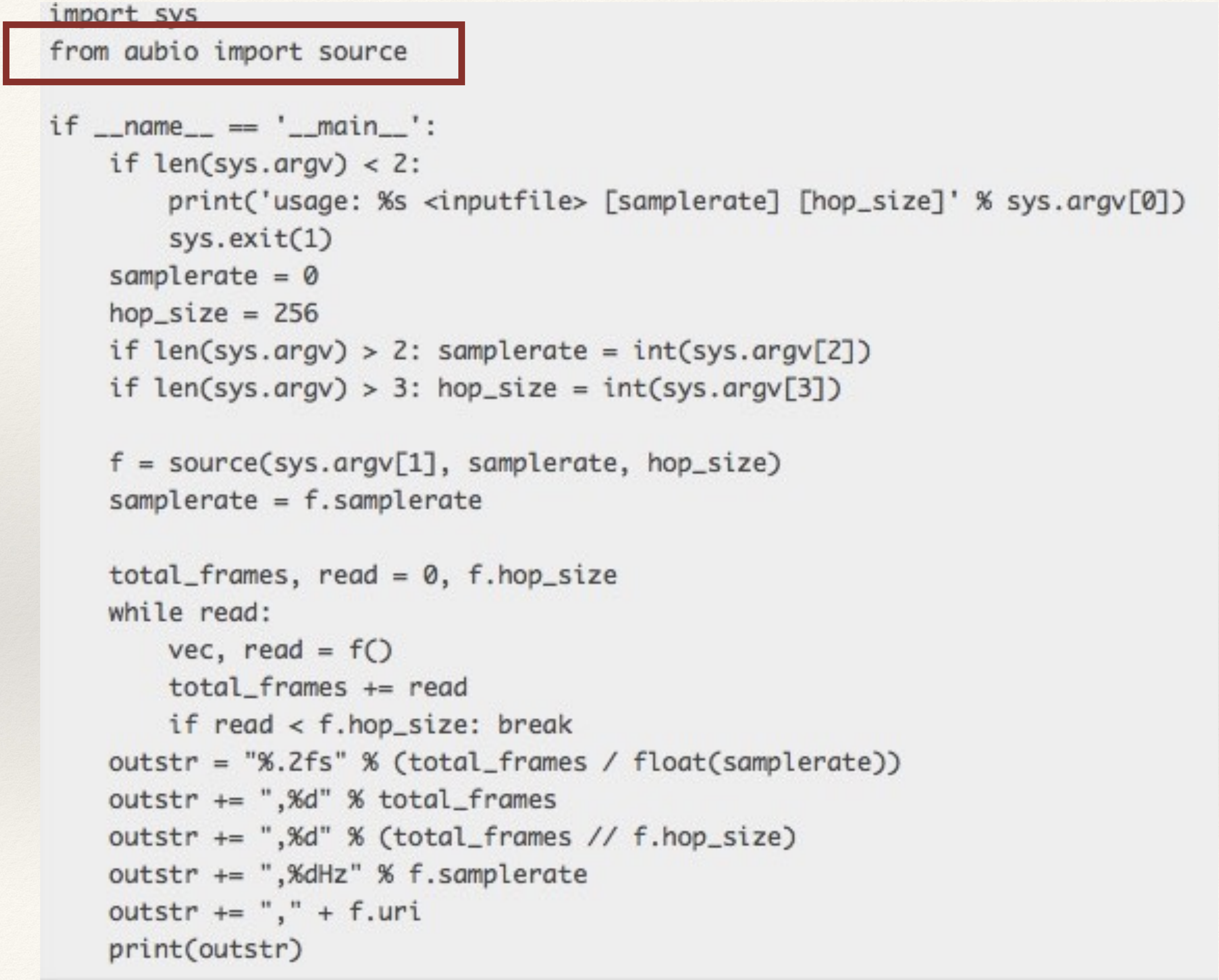


```python
import sys
from aubio import source

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('usage: %s <inputfile> [samplerate] [hop_size]' % sys.argv[0])
        sys.exit(1)
    samplerate = 0
    hop_size = 256
    if len(sys.argv) > 2: samplerate = int(sys.argv[2])
    if len(sys.argv) > 3: hop_size = int(sys.argv[3])

    f = source(sys.argv[1], samplerate, hop_size)
    samplerate = f.samplerate

    total_frames, read = 0, f.hop_size
    while read:
        vec, read = f()
        total_frames += read
        if read < f.hop_size: break
    outstr = "%.2fs" % (total_frames / float(samplerate))
    outstr += ",%d" % total_frames
    outstr += ",%d" % (total_frames // f.hop_size)
    outstr += ",%dHz" % f.samplerate
    outstr += "," + f.uri
    print(outstr)
```

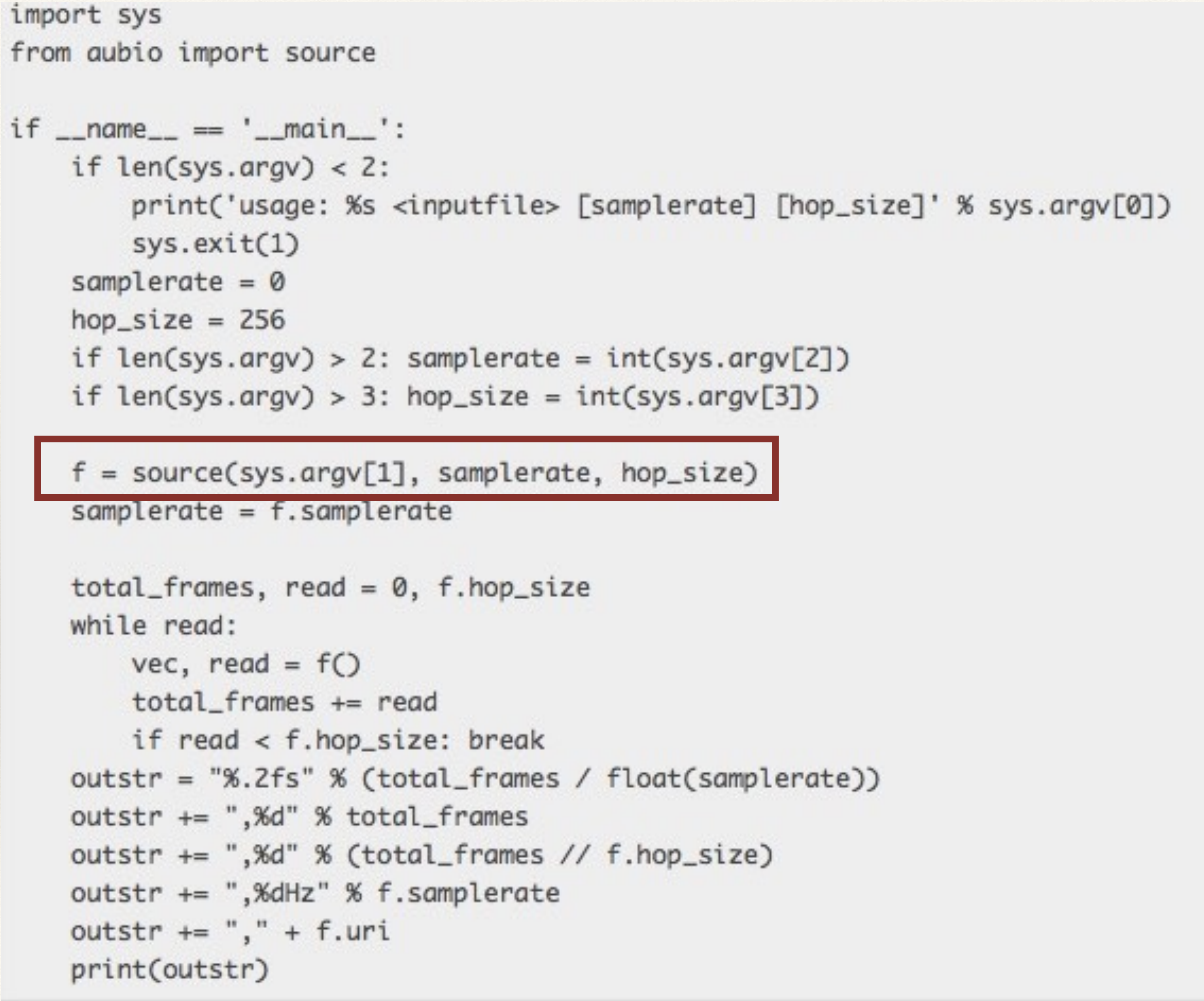


```
import sys
from aubio import source

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('usage: %s <inputfile> [samplerate] [hop_size]' % sys.argv[0])
        sys.exit(1)
    samplerate = 0
    hop_size = 256
    if len(sys.argv) > 2: samplerate = int(sys.argv[2])
    if len(sys.argv) > 3: hop_size = int(sys.argv[3])

    f = source(sys.argv[1], samplerate, hop_size)
    samplerate = f.samplerate

    total_frames, read = 0, f.hop_size
    while read:
        vec, read = f()
        total_frames += read
        if read < f.hop_size: break
    outstr = "%.2fs" % (total_frames / float(samplerate))
    outstr += ",%d" % total_frames
    outstr += ",%d" % (total_frames // f.hop_size)
    outstr += ",%dHz" % f.samplerate
    outstr += "," + f.uri
    print(outstr)
```

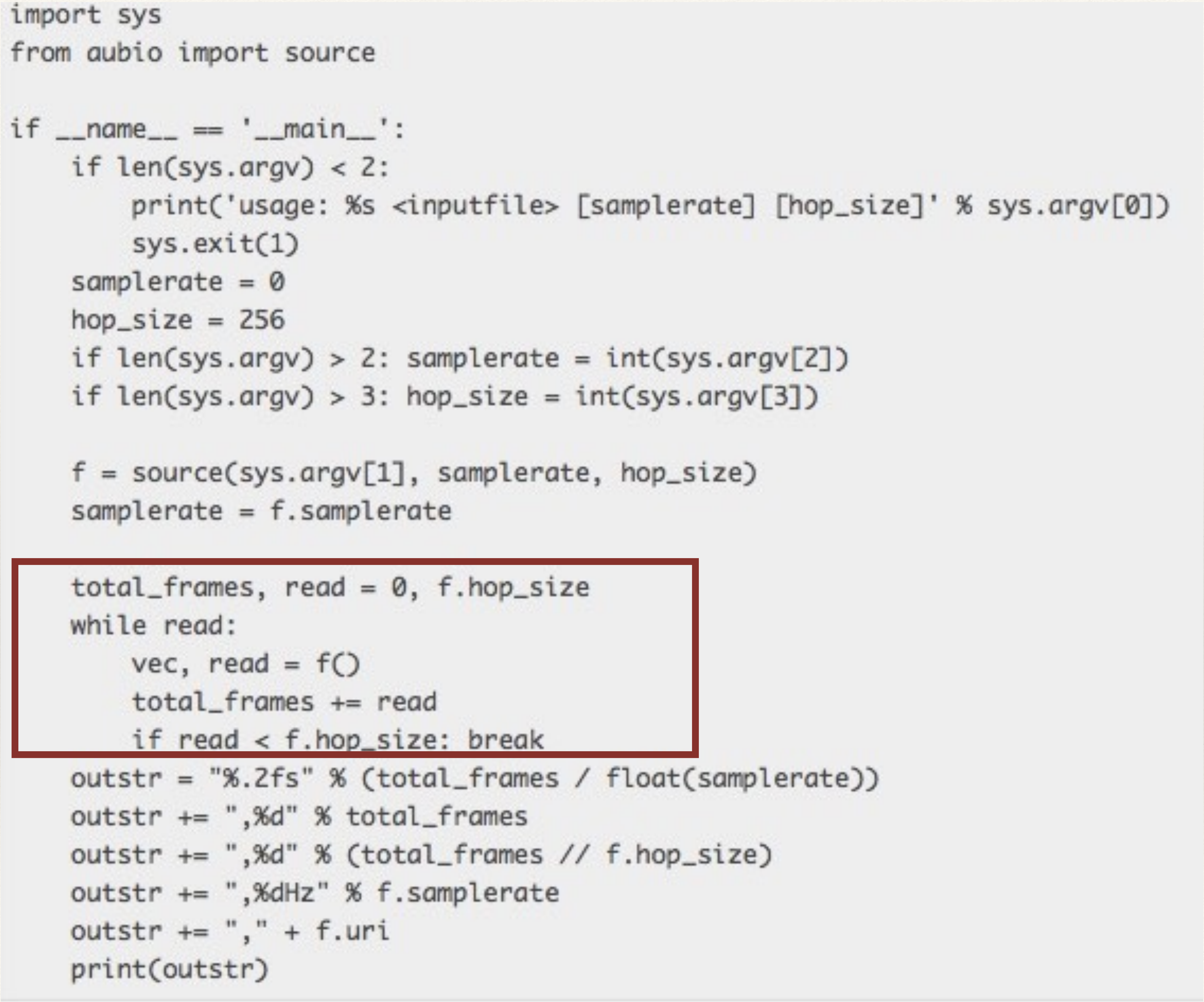


```
import sys
from aubio import source

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('usage: %s <inputfile> [samplerate] [hop_size]' % sys.argv[0])
        sys.exit(1)
    samplerate = 0
    hop_size = 256
    if len(sys.argv) > 2: samplerate = int(sys.argv[2])
    if len(sys.argv) > 3: hop_size = int(sys.argv[3])

    f = source(sys.argv[1], samplerate, hop_size)
    samplerate = f.samplerate

    total_frames, read = 0, f.hop_size
    while read:
        vec, read = f()
        total_frames += read
        if read < f.hop_size: break
    outstr = "%.2fs" % (total_frames / float(samplerate))
    outstr += ",%d" % total_frames
    outstr += ",%d" % (total_frames // f.hop_size)
    outstr += ",%dHz" % f.samplerate
    outstr += "," + f.uri
    print(outstr)
```

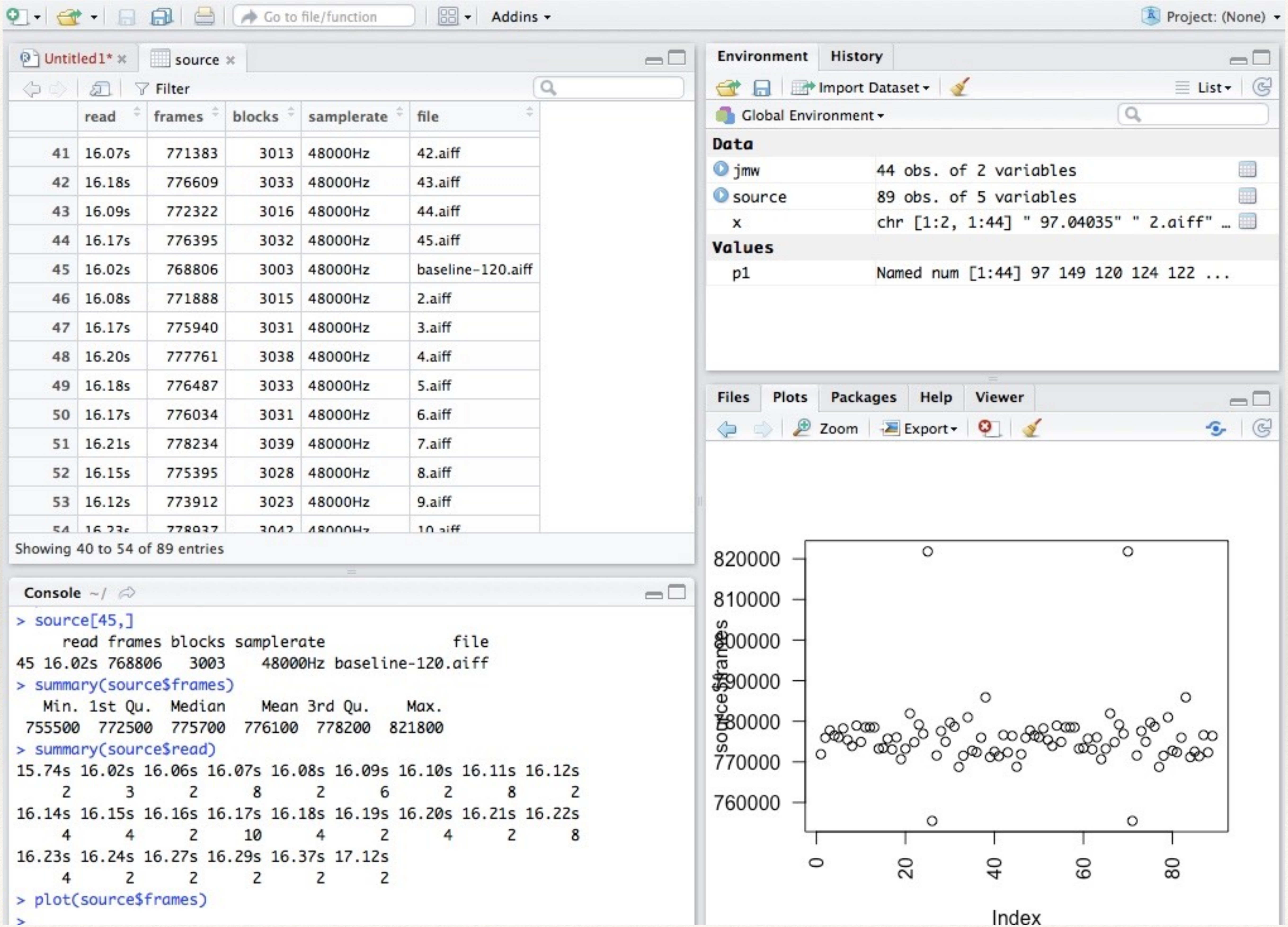


```python
import sys
from aubio import source

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('usage: %s <inputfile> [samplerate] [hop_size]' % sys.argv[0])
        sys.exit(1)
    samplerate = 0
    hop_size = 256
    if len(sys.argv) > 2: samplerate = int(sys.argv[2])
    if len(sys.argv) > 3: hop_size = int(sys.argv[3])

    f = source(sys.argv[1], samplerate, hop_size)
    samplerate = f.samplerate

    total_frames, read = 0, f.hop_size
    while read:
        vec, read = f()
        total_frames += read
        if read < f.hop_size: break
    outstr = "%.2fs" % (total_frames / float(samplerate))
    outstr += ",%d" % total_frames
    outstr += ",%d" % (total_frames // f.hop_size)
    outstr += ",%dHz" % f.samplerate
    outstr += "," + f.uri
    print(outstr)
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

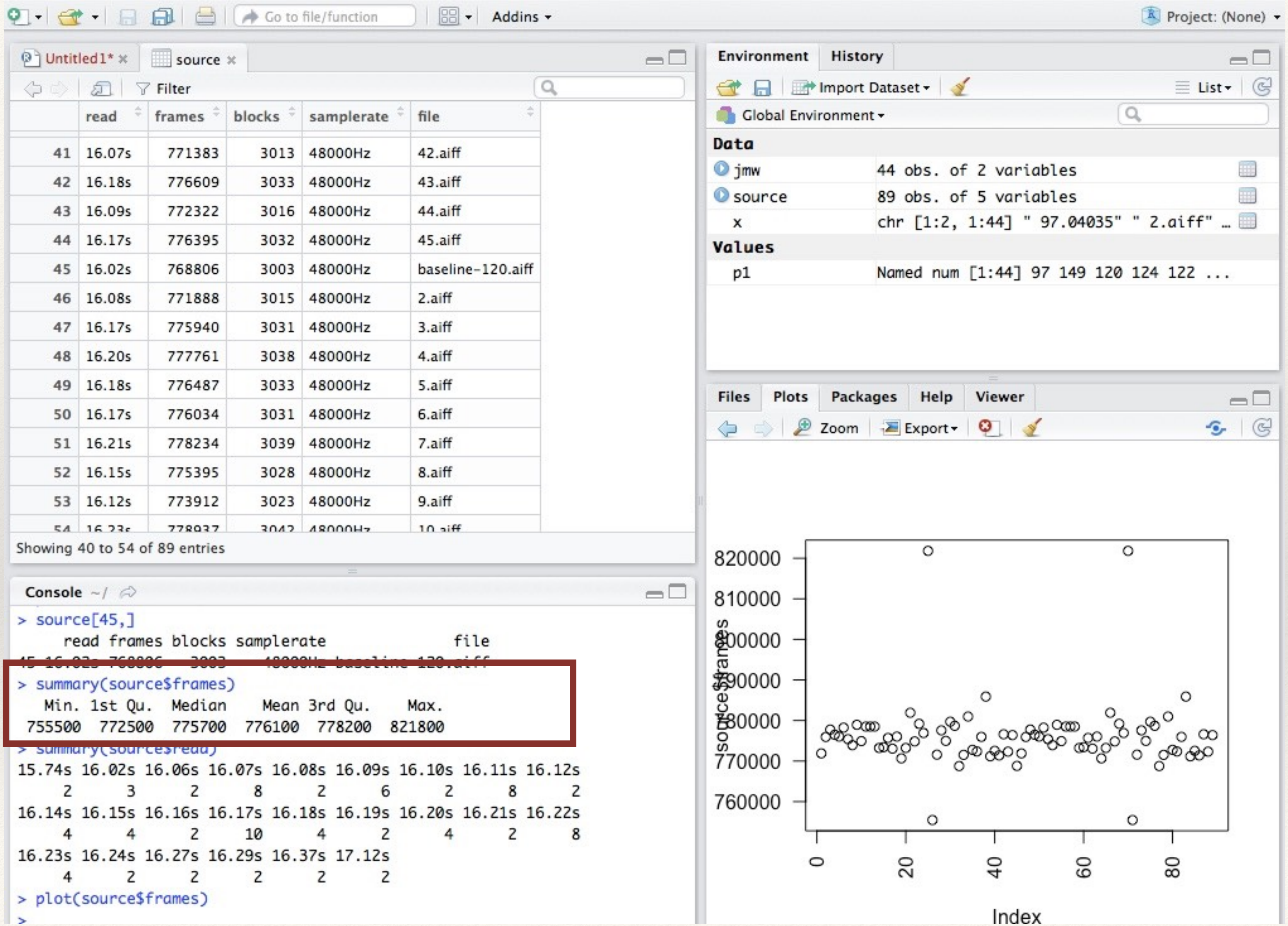
Data

jmw	44 obs. of 2 variables
source	89 obs. of 5 variables
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" …

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12s 
     2      3      2      8      2      6      2      8      2 
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22s 
     4      4      2     10      4      2      4      2      8 
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
>
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

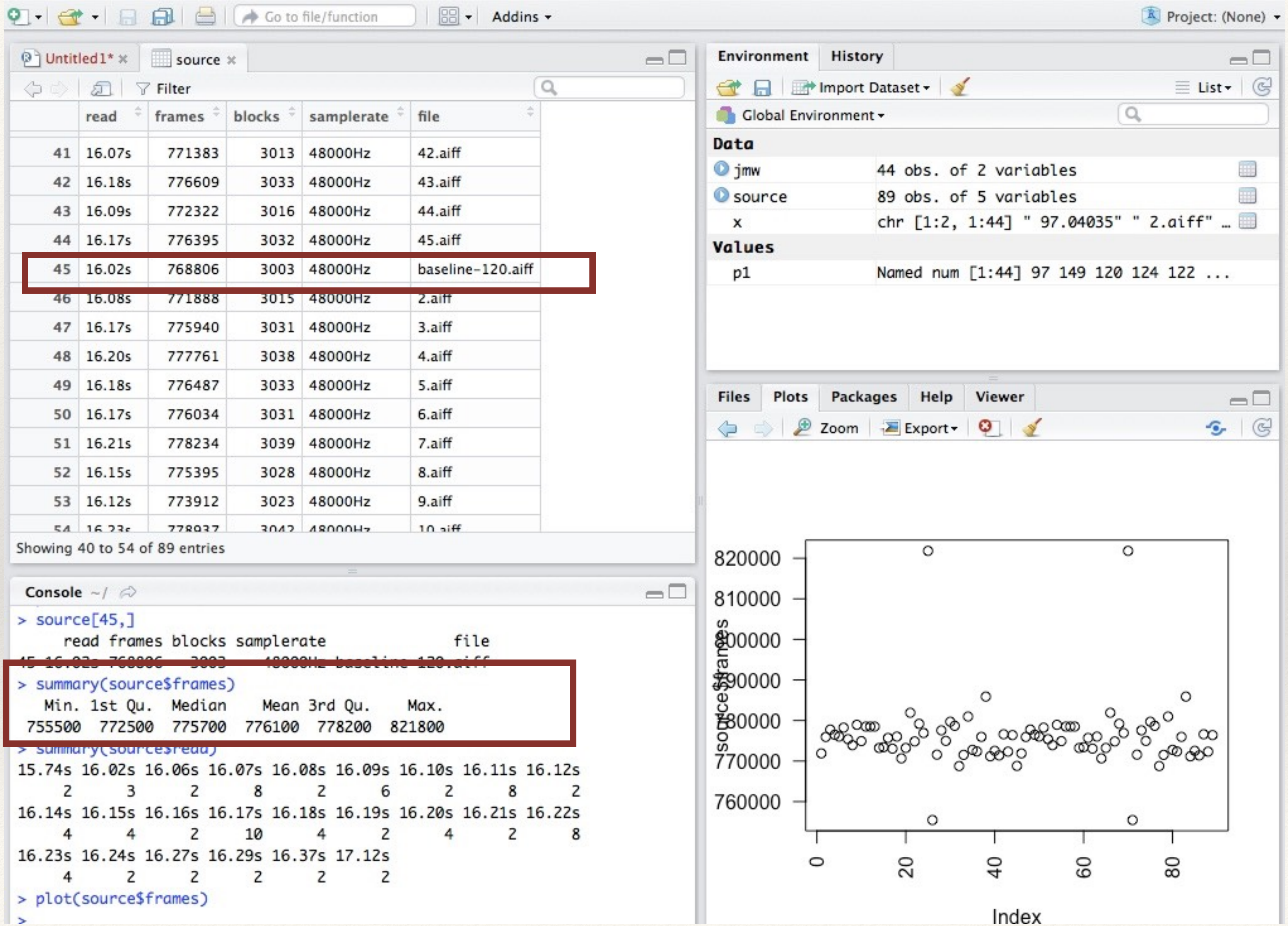
Data

jmw	44 obs. of 2 variables
source	89 obs. of 5 variables
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" ...

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 755500  772500  775700  776100  778200  821800
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12s
     2      3      2      8      2      6      2      8      2
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22s
     4      4      2     10      4      2      4      2      8
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s
     4      2      2      2      2      2
> plot(source$frames)
>
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

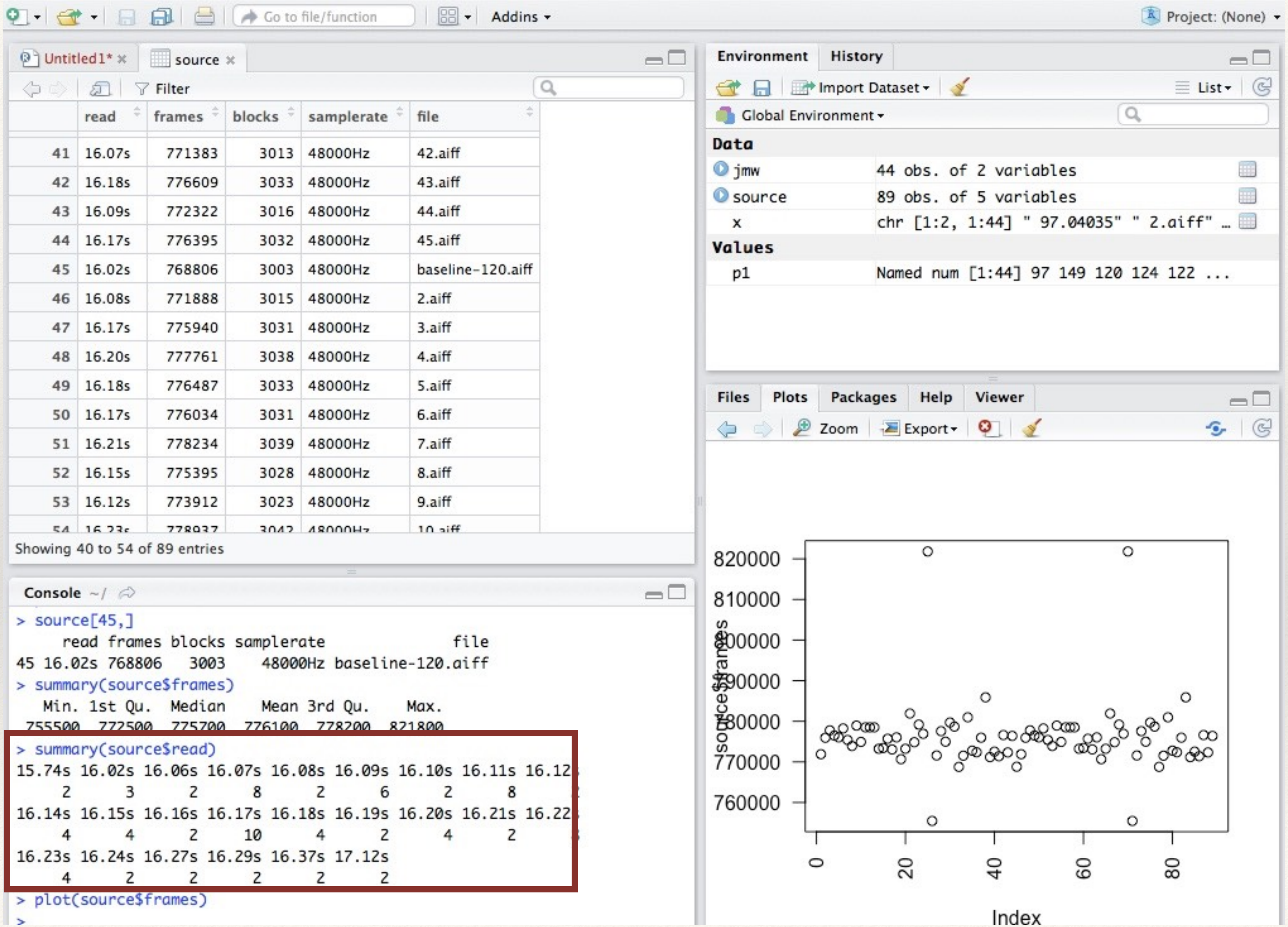
Data

jmw	44 obs. of 2 variables
source	89 obs. of 5 variables
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" …

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12s 
     2      3      2      8      2      6      2      8      2 
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22s 
     4      4      2     10      4      2      4      2      8 
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
>
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

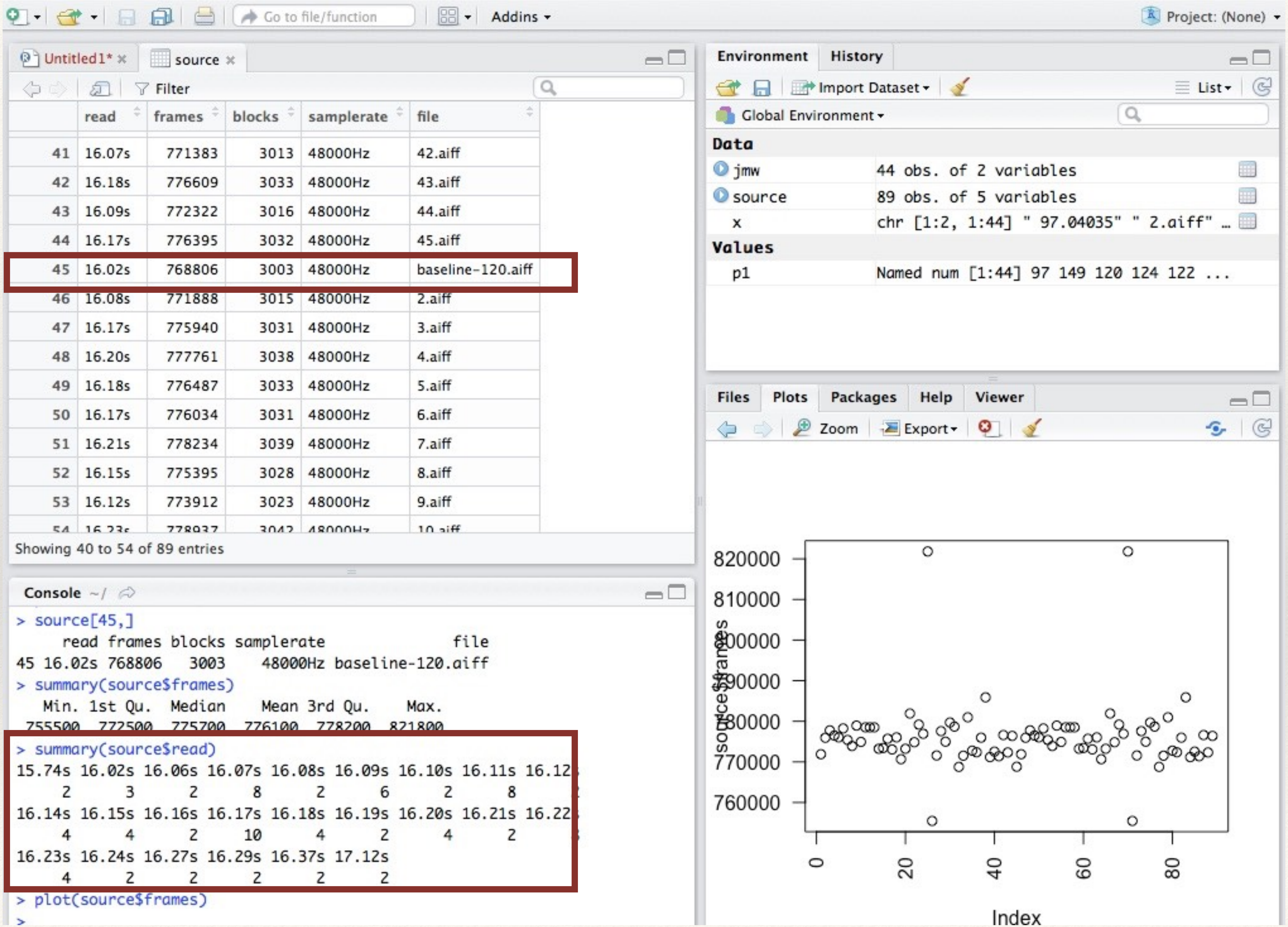
Data

jmw	44 obs. of 2 variables
source	89 obs. of 5 variables
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" …

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12
     2      3      2      8      2      6      2      8
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22
     4      4      2     10      4      2      4      2
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
> 
```

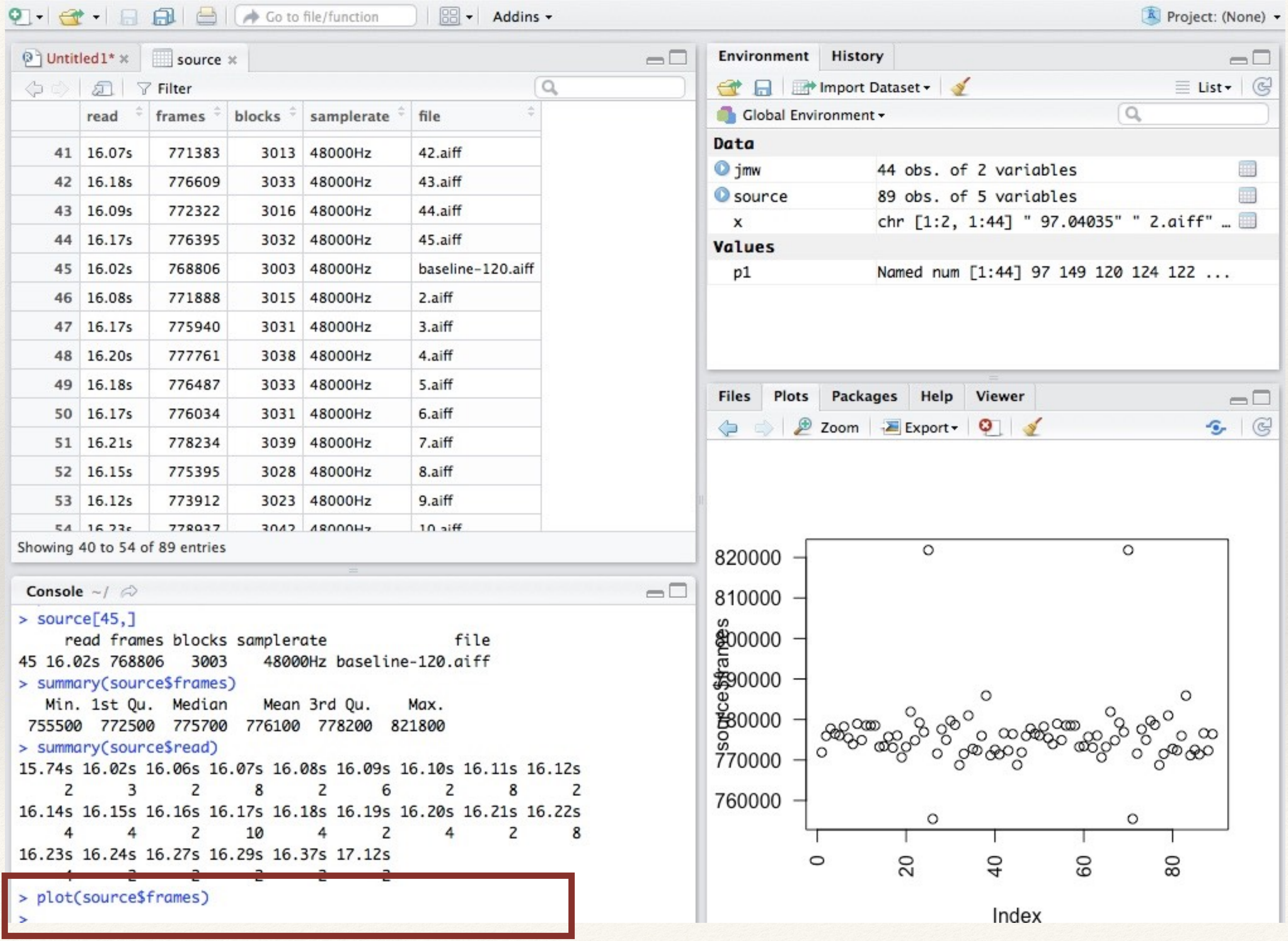


	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

```
Data
jmw        44 obs. of 2 variables
source     89 obs. of 5 variables
x          chr [1:2, 1:44] " 97.04035" " 2.aiff" …
Values
p1         Named num [1:44] 97 149 120 124 122 ...
```

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12
     2      3      2      8      2      6      2      8       
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22
     4      4      2     10      4      2      4      2       
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
>
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

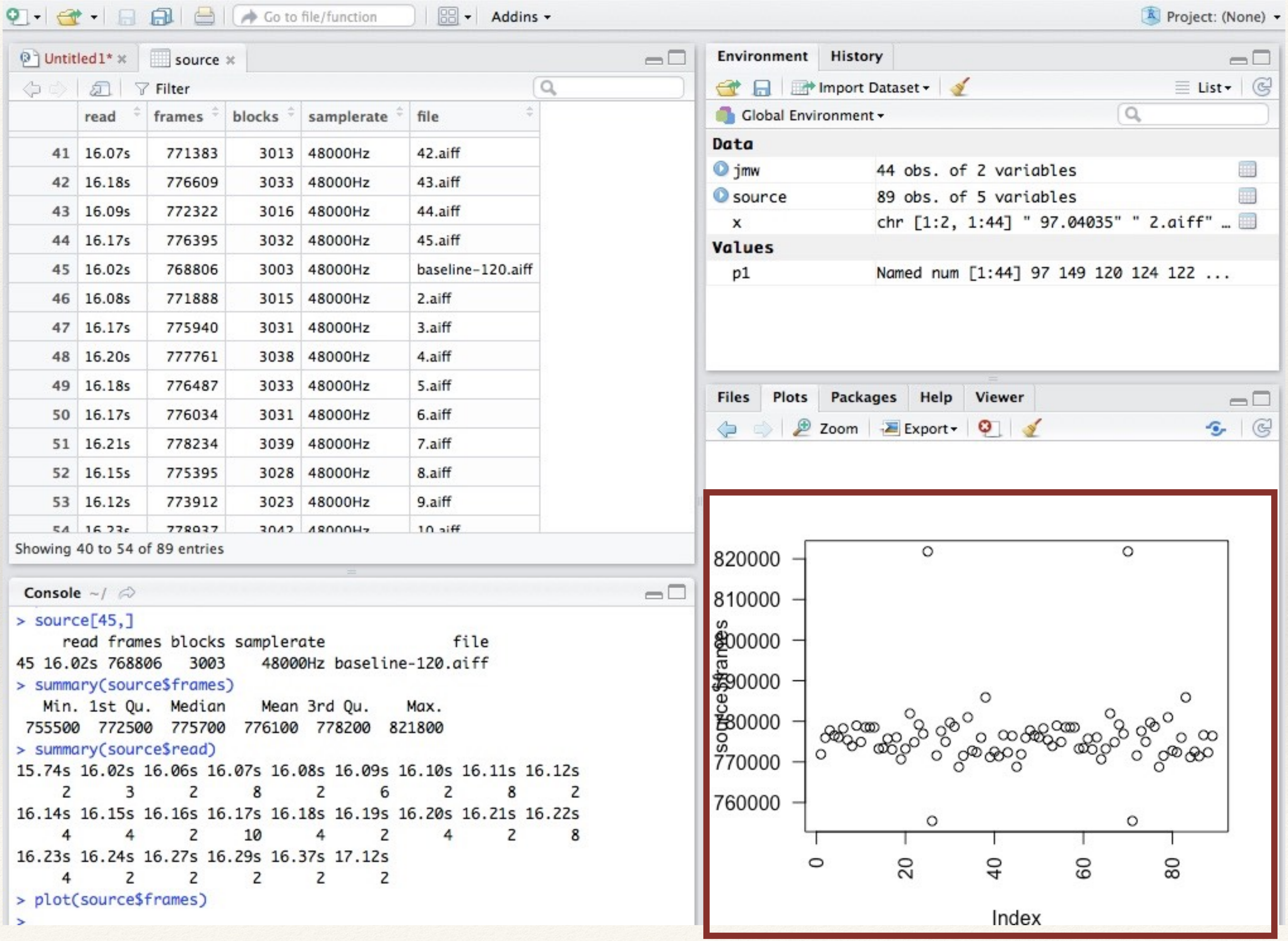
Data

jmw	44 obs. of 2 variables	
source	89 obs. of 5 variables	
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" ...	

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
    read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12s 
     2      3      2      8      2      6      2      8      2 
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22s 
     4      4      2     10      4      2      4      2      8 
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
> 
```



	read	frames	blocks	samplerate	file
41	16.07s	771383	3013	48000Hz	42.aiff
42	16.18s	776609	3033	48000Hz	43.aiff
43	16.09s	772322	3016	48000Hz	44.aiff
44	16.17s	776395	3032	48000Hz	45.aiff
45	16.02s	768806	3003	48000Hz	baseline-120.aiff
46	16.08s	771888	3015	48000Hz	2.aiff
47	16.17s	775940	3031	48000Hz	3.aiff
48	16.20s	777761	3038	48000Hz	4.aiff
49	16.18s	776487	3033	48000Hz	5.aiff
50	16.17s	776034	3031	48000Hz	6.aiff
51	16.21s	778234	3039	48000Hz	7.aiff
52	16.15s	775395	3028	48000Hz	8.aiff
53	16.12s	773912	3023	48000Hz	9.aiff
54	16.23s	778937	3042	48000Hz	10.aiff

Showing 40 to 54 of 89 entries

Data

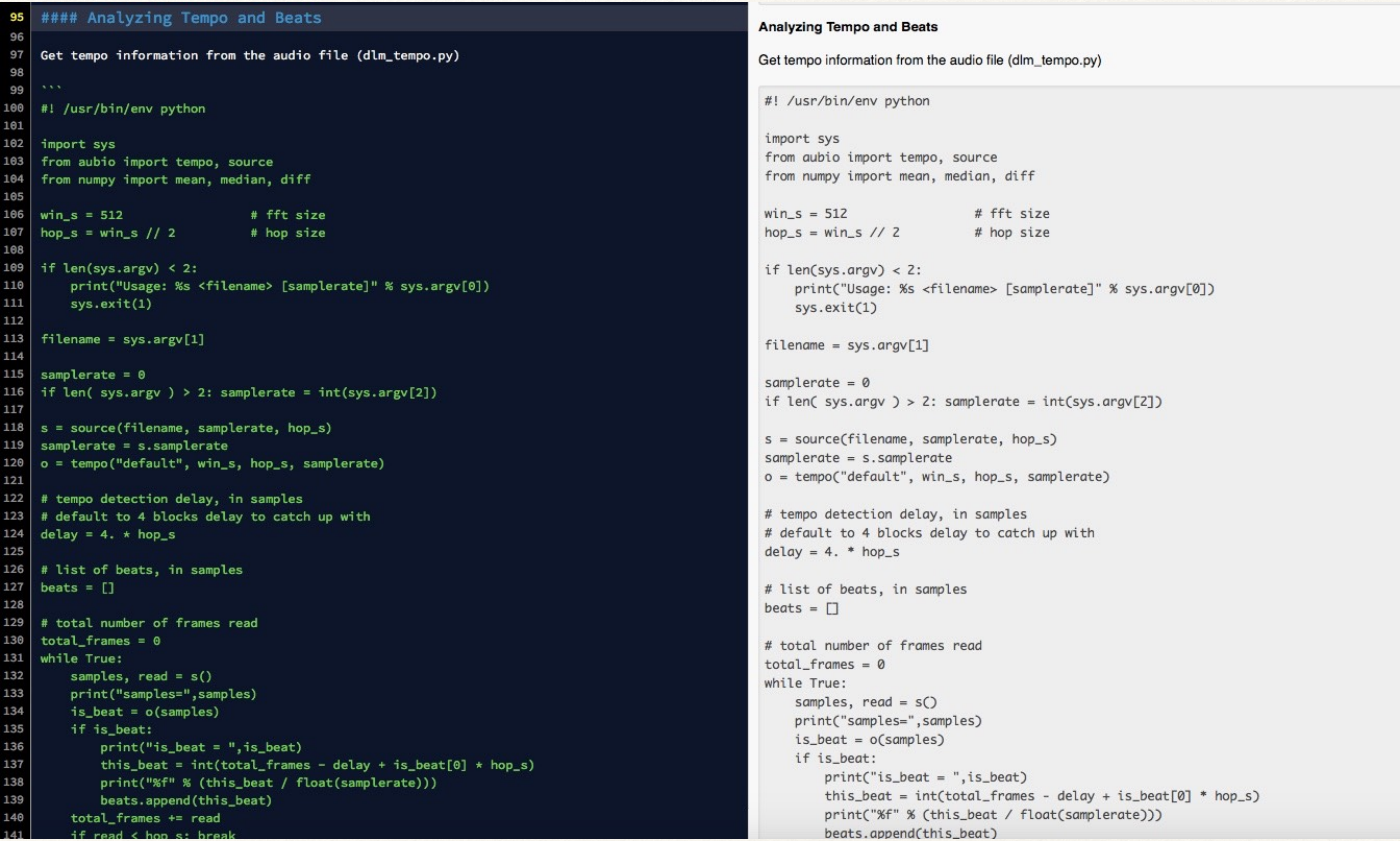
jmw	44 obs. of 2 variables
source	89 obs. of 5 variables
x	chr [1:2, 1:44] " 97.04035" " 2.aiff" …

Values

p1	Named num [1:44] 97 149 120 124 122 ...

```
> source[45,]
     read frames blocks samplerate              file
45 16.02s 768806   3003    48000Hz baseline-120.aiff
> summary(source$frames)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 755500  772500  775700  776100  778200  821800 
> summary(source$read)
15.74s 16.02s 16.06s 16.07s 16.08s 16.09s 16.10s 16.11s 16.12s 
     2      3      2      8      2      6      2      8      2 
16.14s 16.15s 16.16s 16.17s 16.18s 16.19s 16.20s 16.21s 16.22s 
     4      4      2     10      4      2      4      2      8 
16.23s 16.24s 16.27s 16.29s 16.37s 17.12s 
     4      2      2      2      2      2 
> plot(source$frames)
> 
```

Looking and Temp and Beats



~~~~
#### Analyzing Tempo and Beats

Get tempo information from the audio file (dlm_tempo.py)

```
#! /usr/bin/env python

import sys
from aubio import tempo, source
from numpy import mean, median, diff

win_s = 512                 # fft size
hop_s = win_s // 2          # hop size

if len(sys.argv) < 2:
    print("Usage: %s <filename> [samplerate]" % sys.argv[0])
    sys.exit(1)

filename = sys.argv[1]

samplerate = 0
if len( sys.argv ) > 2: samplerate = int(sys.argv[2])

s = source(filename, samplerate, hop_s)
samplerate = s.samplerate
o = tempo("default", win_s, hop_s, samplerate)

# tempo detection delay, in samples
# default to 4 blocks delay to catch up with
delay = 4. * hop_s

# list of beats, in samples
beats = []

# total number of frames read
total_frames = 0
while True:
    samples, read = s()
    print("samples=",samples)
    is_beat = o(samples)
    if is_beat:
        print("is_beat = ",is_beat)
        this_beat = int(total_frames - delay + is_beat[0] * hop_s)
        print("%f" % (this_beat / float(samplerate)))
        beats.append(this_beat)
    total_frames += read
    if read < hop_s: break
~~~~

Analyzing Tempo and Beats

Get tempo information from the audio file (dlm_tempo.py)

```
#! /usr/bin/env python

import sys
from aubio import tempo, source
from numpy import mean, median, diff

win_s = 512                 # fft size
hop_s = win_s // 2          # hop size

if len(sys.argv) < 2:
    print("Usage: %s <filename> [samplerate]" % sys.argv[0])
    sys.exit(1)

filename = sys.argv[1]

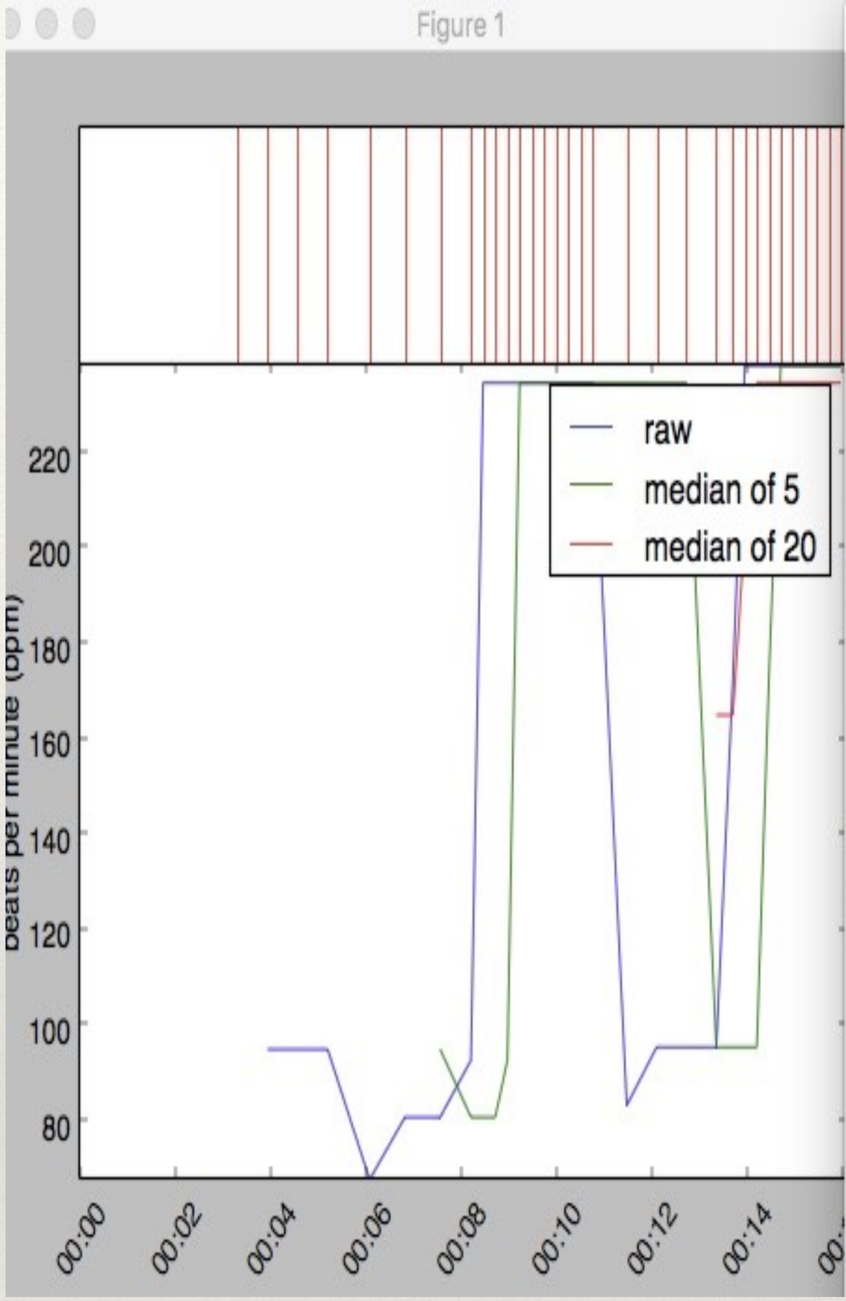
samplerate = 0
if len( sys.argv ) > 2: samplerate = int(sys.argv[2])

s = source(filename, samplerate, hop_s)
samplerate = s.samplerate
o = tempo("default", win_s, hop_s, samplerate)

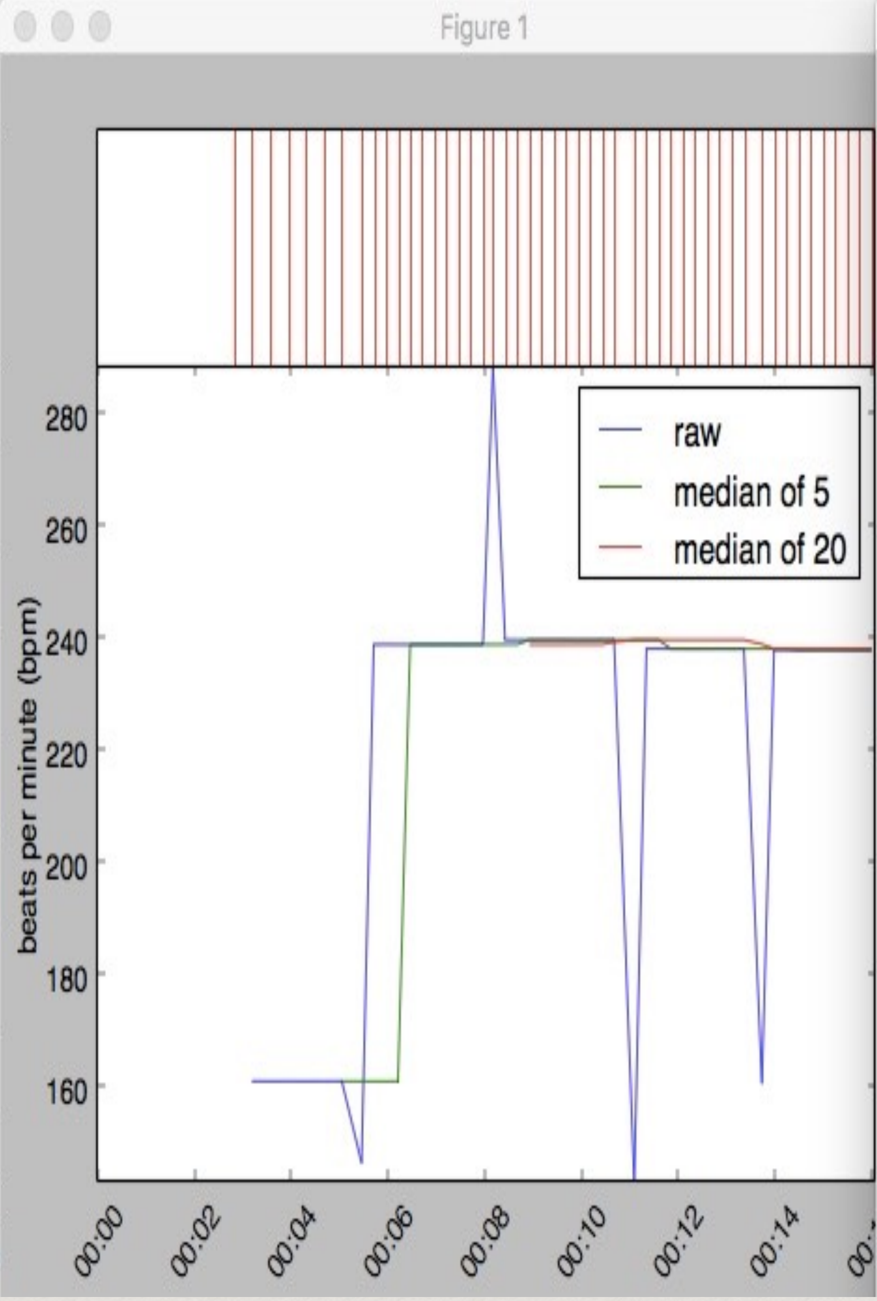
# tempo detection delay, in samples
# default to 4 blocks delay to catch up with
delay = 4. * hop_s

# list of beats, in samples
beats = []

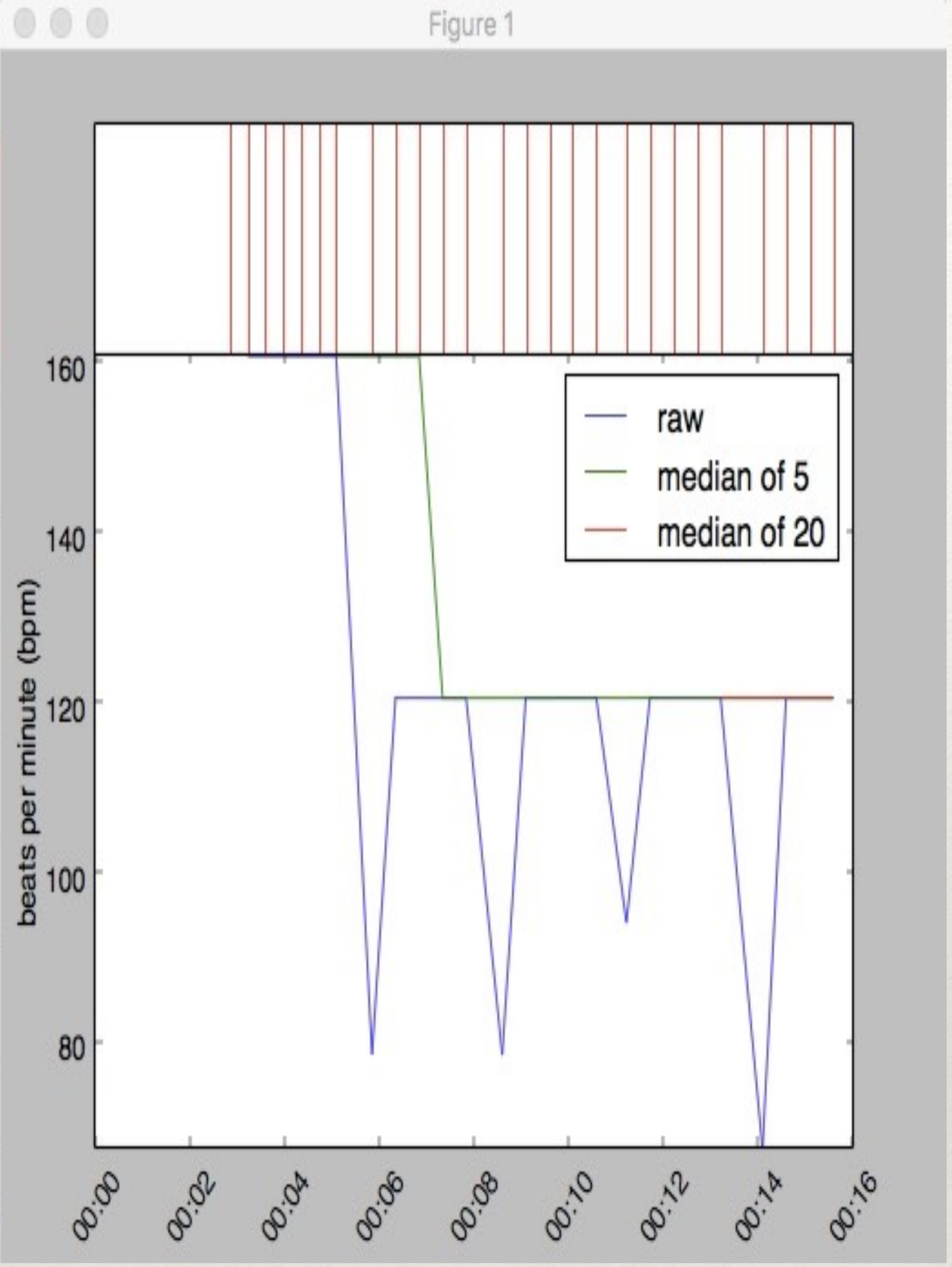
# total number of frames read
total_frames = 0
while True:
    samples, read = s()
    print("samples=",samples)
    is_beat = o(samples)
    if is_beat:
        print("is_beat = ",is_beat)
        this_beat = int(total_frames - delay + is_beat[0] * hop_s)
        print("%f" % (this_beat / float(samplerate)))
        beats.append(this_beat)
```



Day 1



Day 54



Baseline

Summary

Summary

- Although the data didn't show us exactly what we were hoping… it did tell us that we are on the right track.
- Next time we are going to record digital practices files as opposed to analog files.
- We are going to look for more improved tools (aubio was great but is very new and appear to be a little buggy)
- Also Audio as far as we can tell was designed more for music and not drum beats.