

## 1 Abstract

Python ermöglicht es, die Vorteile einer ausgewachsenen Programmiersprache mit der Flexibilität einer interaktiven Skriptsprache zu verbinden. Die Standardbibliothek und die hohe Qualität von wissenschaftlichen Drittmodulen erlaubt eine schnelle Entwicklung von Programmen. ObsPy erweitert Python um Import- und Exportroutinen für Wellenformdatenformate und Metadaten und enthält häufig benötigte Routinen für diverse Aufgabenfelder in der Seismologie. Es steht zudem die breite Palette an Funktionalität von numerischen Modulen zur Matrizenprogrammierung wie NumPy (<http://numpy.scipy.org>) oder SciPy (<http://scipy.org>) zur Verfügung. Zusammen mit z.B. IPython und Matplotlib stellt ObsPy eine mächtige, textbasierte, leicht erlernbare, interaktive Umgebung dar, die komplett aus freien Softwarepaketen besteht und leicht erweiterbar ist. ObsPy ist modular aufgebaut, mit dem Ziel die Abhängigkeiten zu minimieren und steht unter der GPL/LGPLv3 als Open Source Software zur Verfügung.

## 2 Eine kurze Einführung in Python

```
1 import glob
2 from obspy.core import read
3
4 for file in glob.glob("*.z"):
5     st = read(file)
6     print "%s %s %f" % (st[0].stats.station, str(st[0].stats.starttime),
7                          st[0].data.mean(), st[0].data.std())
```

Der Einrückungsgrad markiert zusammengehörige Codeblöcke. (siehe *for*-Schleife)

## 3 Lesen und Schreiben

ObsPy erweitert Python um Import- und Exportroutinen für MiniSEED, GSE2, SAC, SEISAN und die Seismic Handler Formate Q und ASCII.

Hierbei werden sämtliche Daten in einem Stream Objekt gespeichert.

```
1 >>> from obspy.core import read
2 >>> stream_object = read('BW.FURT..EHZ')
3 >>> print stream_object
4      1 Trace(s) in Stream:
5          BW.FURT..EHZ | 2010-02-06T04:53:00.000000Z - 2010-02-06T05:03:00.000000Z | 200.0 Hz,
6          120001 samples
```

Einfaches Auslesen multiplexer Datenströme: Jede Spur wird in einem eigenen Trace Objekt innerhalb des Stream Objekts gespeichert. Jeder Trace hat ein Stats Objekt, das die Metainformationen enthält. Die Daten selbst sind als numpy.ndarrays vorhanden und können somit mit den mächtigen Routinen von NumPy und SciPy weiterverarbeitet werden. Die Stream und Trace Objekte haben diverse Methoden, z.B. die plot() Methode des Stream Objekts:

```
6 >>> stream_object.plot()
```

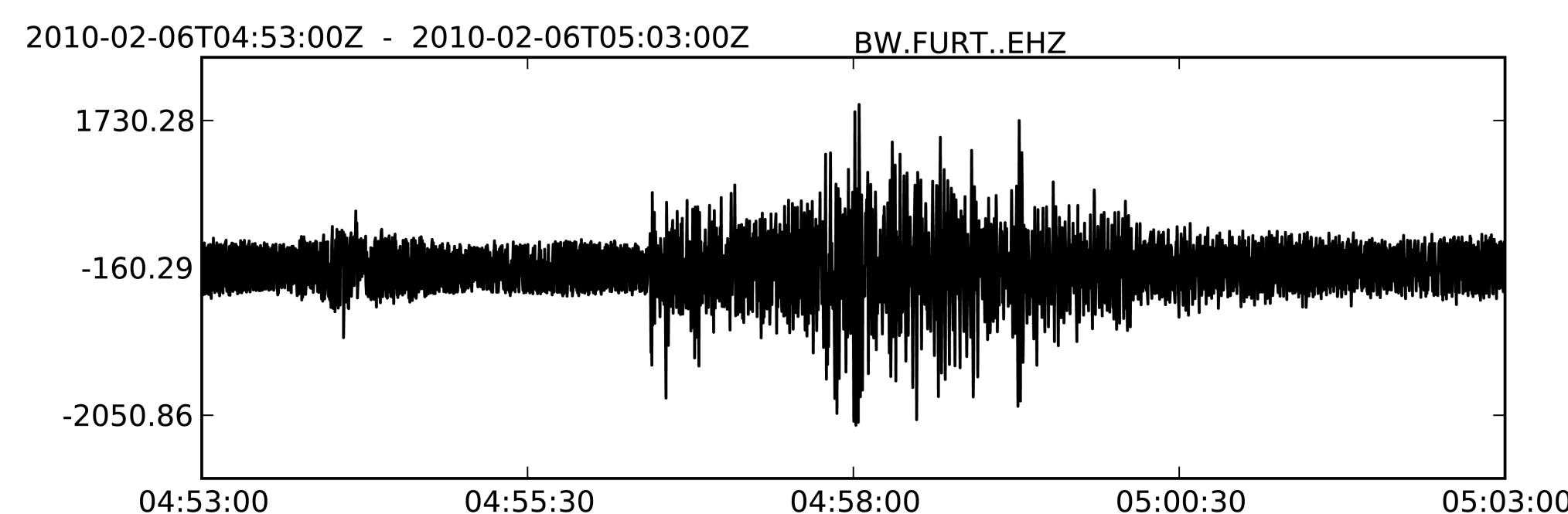


Abbildung 1: Grafische Darstellung des Stream Objekts. Erstellt mit obspy.imaging.

Geschrieben werden die Daten mit der write() Methode.

```
7 >>> stream_object.write('BW.FURT..EHZ', format='GSE2')
```

## 4 SEED/XML-SEED Unterstützung

XML-SEED [4] ist eine XML-Repräsentation des Dataless-SEED Formats [1]. ObsPy erlaubt die Verlustfreie Konvertierung zwischen beiden Formaten. So wird aus

```
1 0500097ANMO +34.946200-106.456700 ...
```

die folgende XML-Ausgabe

```
1 <station_identifier blockette="050">
2   <station_call_letters>ANMO</station_call_letters>
3   <latitude>+34.946200</latitude>
4   <longitude>-106.456700</longitude>
```

## 5 obspy.imaging (Basemap)

Diese Anwendung generiert einen topographischen Kartenausschnitt mitsamt Stationen und einigen Herdflächenlösungen. Dazu werden zunächst via obspy.seishub die Stationsmetadaten in dem Bereich abgefragt und anschließend alle Events im Gebiet. Die Karte wird mit Hilfe des Basemap Toolkits von Matplotlib dargestellt und die Herdflächenlösungen stammen aus obspy.imaging.

Das gesamte Skript ist einschließlich Kommentare und Leerzeilen nur 80 Zeilen lang.

### Codeauschnitte

Abruf der Stationsmetadaten via obspy.seishub.

```
11 client = obspy.seishub.Client(base_url = SERVER)
12 stations = client.station.getList('BW', '*', 
13                                     min_latitude=47.69, max_latitude=47.81,
14                                     min_longitude=12.75, max_longitude=12.95)
```

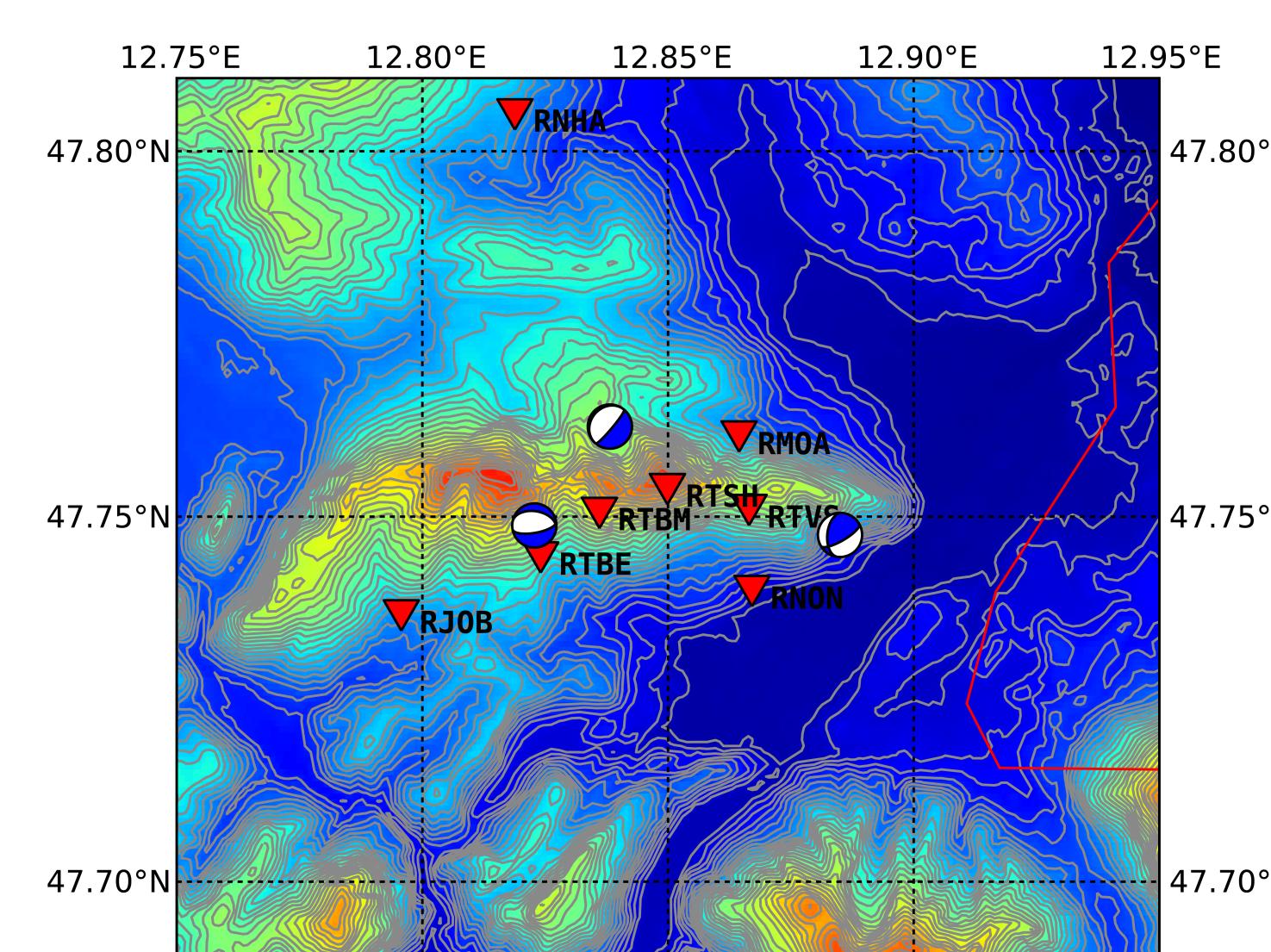


Abbildung 5: Ausgabe des Skriptes. Es zeigt das Gebiet Hochstaufen. Rote Dreiecke kennzeichnen vorhandene seismische Stationen.

## 6 obspy.signal (Instrumentenkorrektur)

Dieses Beispiel zeigt die Korrektur eines STS2-Seismometers auf ein 1 Hz Instrument mit Hilfe von obspy.signal.

Die Daten werden mit obspy.core eingelesen und anschließend wird der Mittelwert abgezogen.

```
s.tr.data = tr.data - tr.data.mean()
```

Jetzt werden die Daten mit Hilfe der seisSim() Funktion aus obspy.signal korrigiert. sts2 und onehzinst sind Python Dictionaries, die Informationen zur den Instrumentenantworten enthalten.

```
17 data2 = seisSim(tr.data,
18                   tr.stats.sampling_rate, sts2,
19                   inst_sim=onehzinst, water_level=600.0)
20 data2 = data2 / sts2["sensitivity"]
```

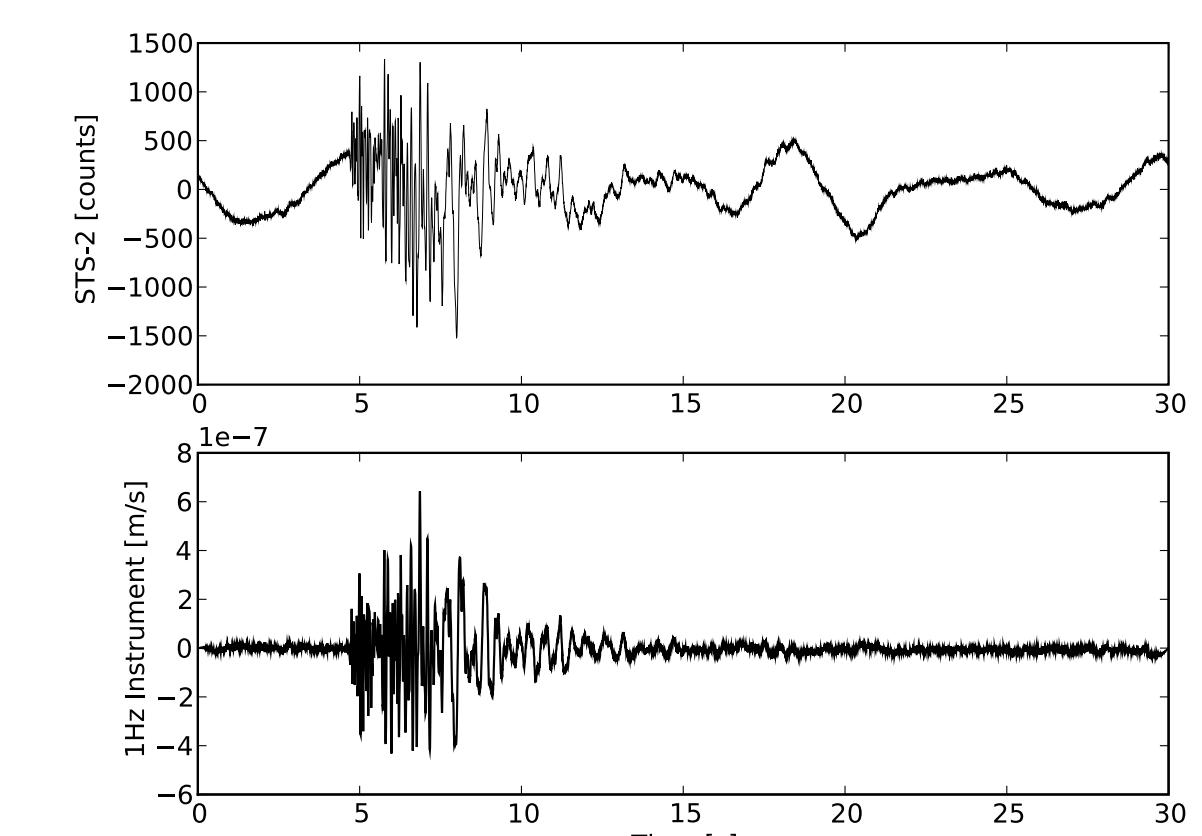


Abbildung 4: Instrumenkorrektur eines STS2 auf ein 1Hz Instrument.

## 7 obspy.imaging

Das obspy.imaging Modul bietet Routinen zur grafischen Darstellung von Seismogrammen, Spektrogrammen und Herdflächenlösungen an.

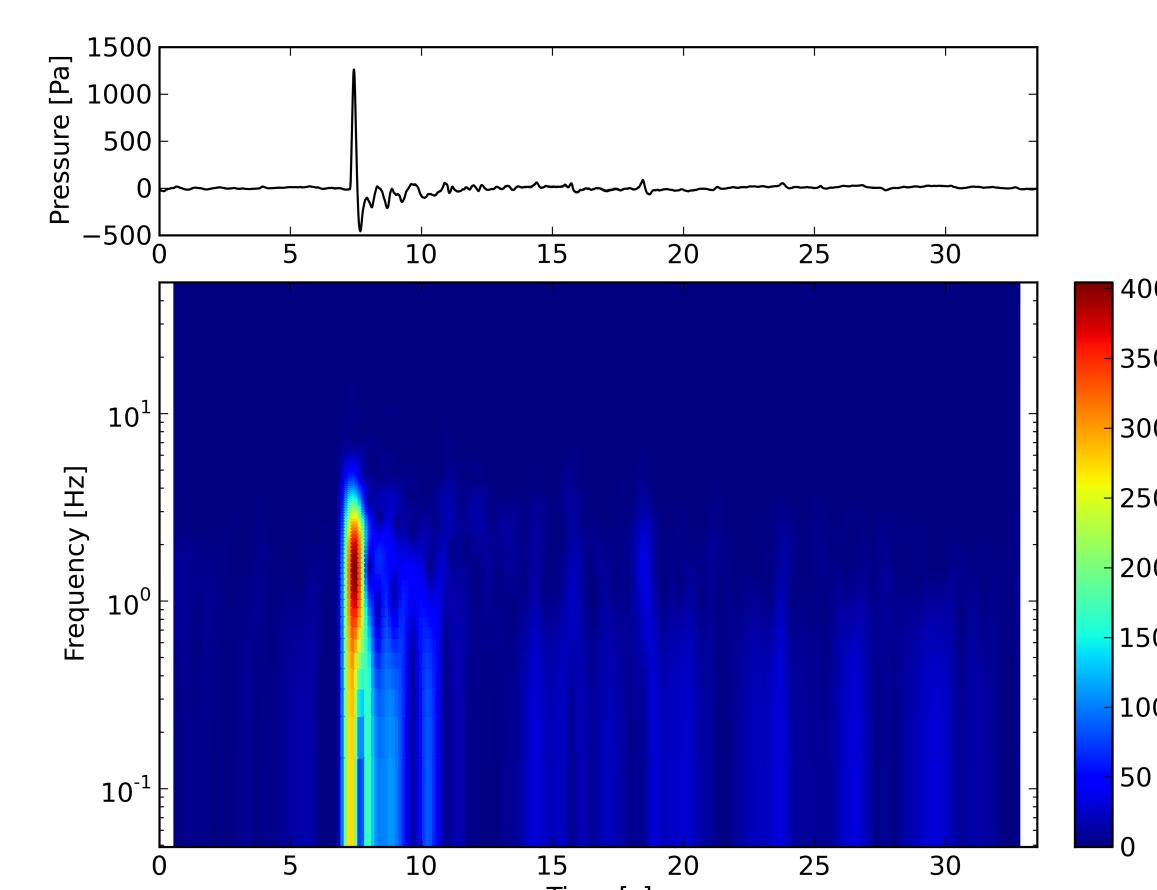


Abbildung 2: Spektrogramm, das mit Hilfe von obspy.imaging.spectrogram erstellt wurde.

Herdflächenlösungen können als *[Strike, Dip, Rake]* oder mit sechs unabhängigen Komponenten des Momententensors angegeben werden.

```
mt = [[264.98, 45.00, -159.99], [130.79, 98.0, 
    [0.99, -2.00, 1.01, 0.92, 0.48, 0.15],
    [-2.39, 1.04, 1.35, 0.57, -2.94, -0.94]]]
```

Jede Lösung wird in einen Graphen eingezzeichnet.

```
for i, t in enumerate(mt):
    ax.add_collection(Beach(t, size=100,
                           width=35, xy=(x,y), linewidth=.6))
```

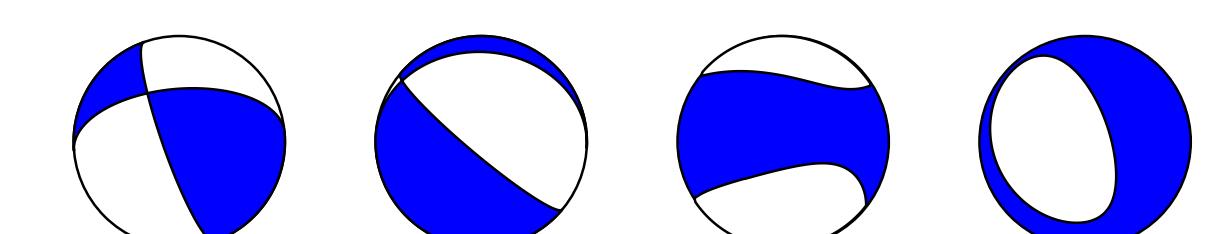


Abbildung 3: Eine Sammlung von Herdflächenlösungen, erstellt mit obspy.imaging.beachball.

## 8 http://www.obspy.org, weitere Informationen

Auf der Homepage des Projekts <http://www.obspy.org> befindet sich eine ausführliche Dokumentation, ein Tutorial, Installationsanleitungen für mehrere Plattformen, zahlreiche Beispiele und viele weitere Hinweise, die den Einstieg in ObsPy erleichtern.

Mehrere Entwickler verschiedenster Institutionen beteiligen sich an ObsPy und sind auf der Homepage offen für Vorschläge, Anregungen und Kritik.

### Weitere Funktionen von ObsPy

- **obspy.signal:** Filtern, Triggern, Rotieren, Magnitudenbestimmung, Instrumentenkorrektur, Koordinatentransformationen
- **obspy.arclink, obspy.fissures, obspy.seishub:** Datenaquisition von ArcLink/WebDC, IRIS DMC DHI/Fissures und SeisHub ([www.seishub.org](http://www.seishub.org)) [2].

## 9 Literatur

- [1] AHERN, Timothy K. ; DOST, Bernard: *SEED Reference Manual - standard for the exchange of earthquake data - SEED format version 2.4*. IRIS DMC, January 2009
- [2] BARSCH, Robert: *Web-based technology for storage and processing of multi-component data in seismology*, LMU München, Diss., 2009
- [3] BEYREUTHER, Moritz ; BARSCH, Robert ; KRISCHER, Lion ; MEGIES, Tobias ; BEHR, Yannik ; WASSERMANN, Joachim: ObsPy: A Python Toolbox for Seismology. In: *SRL* (2010)
- [4] TSUBOI, S. ; TROMP, J. ; KOMATITSCH, D.: An XML-SEED Format for the Exchange of Synthetic Seismograms. In: *EOS Transactions of American Geophysical Union. SF31B-03*, 2004

Abruf der Eventdaten.

```
21 events = client.event.getList(min_latitude=47.69,
22                               max_latitude=47.81, min_longitude=12.75,
23                               max_longitude=12.95)
```

Erzeugung der Basemap.

```
45 m = Basemap(projection='merc', lon_0=13, lat_0=48, resolution='h',
46           llcrnrlon=12.75, llcrnrlat=47.69,
47           urcrnrlon=12.95, urcrnrlat=47.81)
```

Zeichnen der Herdflächenlösungen.

```
73 ax = plt.gca()
74 for i in range(len(events_focmc)):
75     b = Beach(events_focmc[i], xy=(x[i], y[i]), width=1000,
               linewidth=1)
```

```
77 ax.add_collection(b)
```