

Supplementary material

Michael Dietze

February 10, 2017

The article “Validity, precision and limitations of seismic-based rockfall monitoring” by Michael Dietze, Solmaz Mohadjer, Jens M. Turowski, Todd Ehlers and Niels Hovius, submitted to Natural Hazards and Earth System Sciences, uses records of terrestrial laser scanning and broadband seismometers to investigate rockfall activity in a part of the Lauterbrunnen Valley, Bernese Oberland, Switzerland. This supplementary material provides the seismic signals of ten rockfall events, co-detected by both methods.

The documentation explains the general work that is needed to reproduce the main results discussed in the manuscript. However, there may be some changes in the parameter settings that cannot be fully implemented, here. One example is the setup of the STA/LTA picking routine, which needed adjustments simply due to the short lengths of the clipped seismic signals.

Content of the supplementary material

- Readme.pdf - the information document you are reading by now
- ASCII files labelled event_*n*.txt - seismic data of ten rockfall events
- seismic_stations.txt - summary of the seismic stations
- lidar.txt - lidar-detected rockfall information

Software requirements

As far as possible, all data preparation, analysis and visualisation steps were made using free and open software. To work with the seismic signals the following software needs to be or is recommended to be installed:

- R
- RStudio

Furthermore, the following R-packages need to be installed. Please consider that this documentation cannot guide you through the sometimes rocky paths to successful installations of R-packages. The internet is indeed a very good ressource to solve issues during installation.

- raster (`install.packages(pkgs = "raster")`)
- sp (`install.packages(pkgs = "sp")`)
- devtools (`install.packages(pkgs = "devtools")`)
- eseis (`devtools::install_github(repo = "coffeemugger/eseis")`)

The package eseis is the central tool for working with seismic data in R. It is hosted on GitHub. All other packages are hosted on CRAN.

Loading the rockfall data sets

The content of the zip-archive `rockfalls.zip` needs to be extracted. The following code will import the data set of the first event to R. Note that `"~/Downloads/"` is the path to the extracted files and may need adjustments.

```
## load event 1
event_1 <- read.table(file = "~/Downloads/event_1.txt",
                      header = TRUE,
                      sep = ",",
                      stringsAsFactors = FALSE)
```

```
## show structure of the imported data set
str(event_1)
```

```
## 'data.frame':    12199 obs. of  5 variables:
## $ Time : chr  "2014-10-12 22:45:20.005" "2014-10-12 22:45:20.009" "2014-10-12 22:45:20.015" "2014-10-12 22:45:20.019"
## $ LAU02: int   61436 61467 61449 61432 61445 61451 61458 61444 61448 61451 ...
## $ LAU03: int   59069 59070 59062 59060 59059 59060 59066 59061 59062 59080 ...
## $ LAU04: int   63202 63174 63162 63163 63145 63157 63168 63154 63161 63143 ...
## $ LAU05: int   63488 63489 63492 63512 63510 63527 63541 63508 63455 63445 ...
```

The data set is a data frame with 12199 samples. The first element (column) is the time stamp as POSIXct-string. The subsequent four elements contain the raw seismic signals of the vertical component from four stations: LAU02 to LAU05.

To set these data into context, the seismic station summary file is needed, as well. It can be loaded in a similar way:

```
## load station summary file
stations <- read.table(file = "~/Downloads/seismic_stations.txt",
                      header = TRUE,
                      stringsAsFactors = FALSE)
```

```
## show content of the imported data set
print(stations)
```

```
##      ID      name      x      y      z sensor_type logger_type
## 1 LAU02  Gate of China 415766.2 5157910 1524      TC120s   Cube3ext
## 2 LAU03  Blatters Herbs 415398.0 5156814 1595      TC120s   Cube3ext
## 3 LAU04  Mosquito Fabric 416236.0 5158777  832      TC120s   Cube3ext
## 4 LAU05    Funny Rain 416035.2 5157843  888      TC120s   Cube3ext
```

Each seismic station has an ID (corresponding to the names of the seismic signals of each event), a station name, location information in UTM coordinates, the elevation of deployment and information about the sensor and logger type.

Finally, to load the lidar-based rockfall events, the following code is needed. The volumes are given in m^3 , the coordinates in the UTM system. IDs correspond to those of the event files.

```
## load event 1
lidar <- read.table(file = "~/Downloads/lidar.txt",
                   header = TRUE,
                   sep = ",",
                   stringsAsFactors = FALSE)
```

```
## show content of the imported data set
print(lidar)
```

```
##      ID volume..m.3. volume_error..m.3.  x..UTM.  y..UTM.
## 1      1      0.201      0.0046020 415511.2 5156535
## 2      2      0.063      0.0066810 415522.5 5156542
## 3      3      0.201      0.0046020 415541.4 5156844
```

## 4	4	0.175	0.0113370	415565.5	5156845
## 5	5	0.053	0.0043365	415591.4	5156934
## 6	6	0.416	0.0208125	415950.0	5158213
## 7	7	0.891	0.0384285	415952.1	5157829
## 8	8	0.258	0.0138285	416005.0	5157897
## 9	9	0.192	0.0100230	416116.3	5158797
## 10	10	2.338	0.0854475	416037.4	5158649

Further requirements

Due to copyright reasons it was not possible to also deliver the digital elevation model (DEM) used to run the localisation routines. You may need to provide such a DEM by your own. The DEM used by the authors was provided by swissALTI3D.

General workflows

Again, this documentation cannot give detailed into the R-package `eseis`. Please see the documentation of this package, e.g, by typing `?eseis` in the R console. Most of the following text uses event 7 as example, because it corresponds to figure 7 of the accompanying manuscript. There are a few preparation steps needed:

```
## load package
library("eseis")

## load event 7
data <- read.table(file = "~/Downloads/event_7.txt",
                   header = TRUE,
                   sep = ",",
                   stringsAsFactors = FALSE)

## assign time and signal data
t <- as.POSIXct(x = data$Time, tz = "UTC")
s <- as.list(data[,2:5])
```

Picking event onsets and durations

The following code snippet filters the raw signals of all stations to the frequency window of interest for rockfall detection. The result is tapered to remove edge effects. Then, the signal envelope is calculated and the STA/LTA picking routine is applied to the first trace (i.e., LAU02, “Gate of China”). Note that due to the short length of the signal trace the STA/LTA setup differs from the values proposed in the manuscript.

```
## filter signal
s_f <- signal_filter(data = s,
                    dt = 1/200,
                    f = c(10, 30))

## taper signal to remove egde effects
s_f <- signal_taper(data = s_f, n = 1000)

## calculate envelopes
s_e <- signal_envelope(data = s_f)

## pick events
```

```
events <- signal_stalta(data = s_e[[1]],
                        time = t,
                        dt = 1/200,
                        sta = 100,
                        lta = 5000,
                        freeze = TRUE,
                        on = 5,
                        off = 3)
```

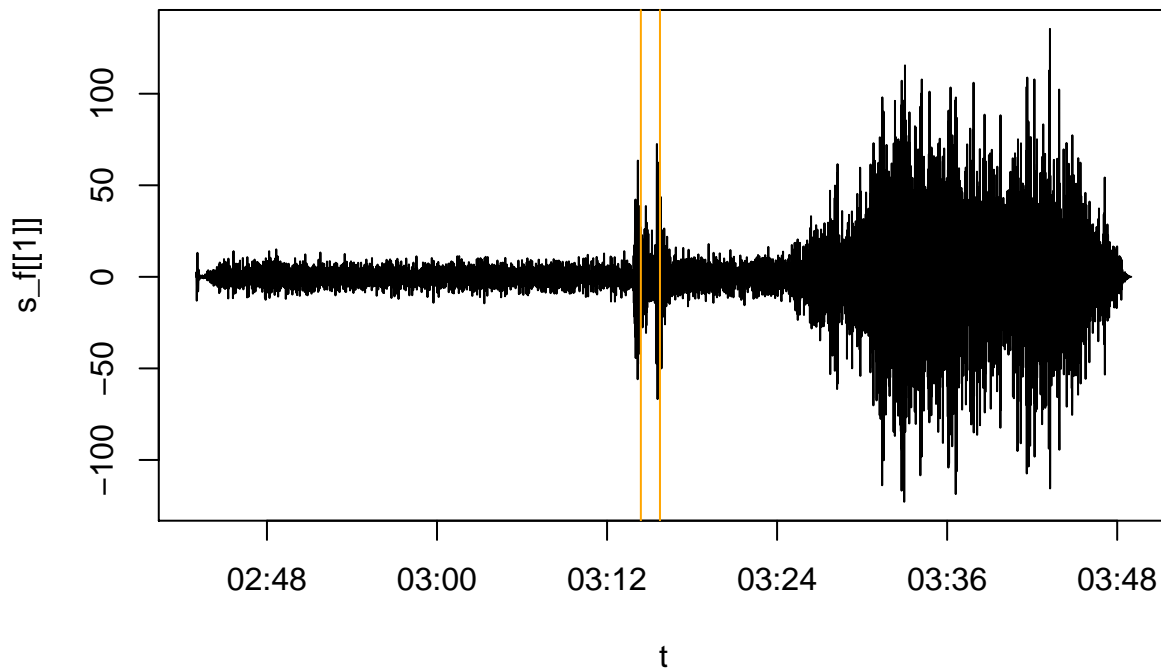
```
print(events)
```

```
##   ID          start duration
## 1  1 2014-09-25 07:03:14 0.6710000
## 2  2 2014-09-25 07:03:15 0.5310001
```

```
## plot signal and event onsets
```

```
plot(x = t, y = s_f[[1]], type = "l")
```

```
abline(v = events$start, col = "orange")
```



Deconvolution of signals

Deconvolution is needed to remove the instrument response of the data and to plot it in meaningful units (10 $\log_{10} \text{ m}^2/\text{s}^2$, dB).

```
## check that used sensors and loggers are in library
```

```
list_sensor()[[1]]
```

```
## $ID
```

```
## [1] "TC120s"
```

```
##
```

```
## $name
```

```
## [1] "Trillium Compact 120s"
```

```

##
## $manufacturer
## [1] "Nanometrics"
##
## $type
## [1] "broadband seismometer"
##
## $n_components
## [1] 3
##
## $comment
## [1] "Data taken from data base of Arnaud Burtin"
##
## $poles
## [1] 3.691e-02+3.702e-02i 3.691e-02-3.702e-02i -3.430e+02+0.000e+00i
## [4] -3.700e+02+4.670e+02i -3.700e+02-4.670e+02i -8.360e+02+1.522e+03i
## [7] -8.360e+02-1.522e+03i -4.900e+03+4.700e+03i -4.900e+03-4.700e+03i
## [10] -6.900e+03+0.000e+00i -1.500e+04+0.000e+00i
##
## $zeros
## [1] 0+ 0i 0+ 0i -392+ 0i -1960+ 0i -1490+1740i -1490-1740i
##
## $s
## [1] 749.1
##
## $k
## [1] 4.34493e+17
list_logger()[[1]]

## $ID
## [1] "Cube3ext"
##
## $name
## [1] "Cube 3ext"
##
## $manufacturer
## [1] "Omnirecs"
##
## $type
## [1] "The greatest data logger in the world"
##
## $n_components
## [1] 3
##
## $comment
## [1] ""
##
## $AD
## [1] 2.4414e-07
## deconvolve signal
s_d <- signal_deconvolve(data = s,
                        dt = 1/200,
                        sensor = "TC120s",

```

```
logger = "Cube3ext")
```

Plotting the waveforms of seismic data

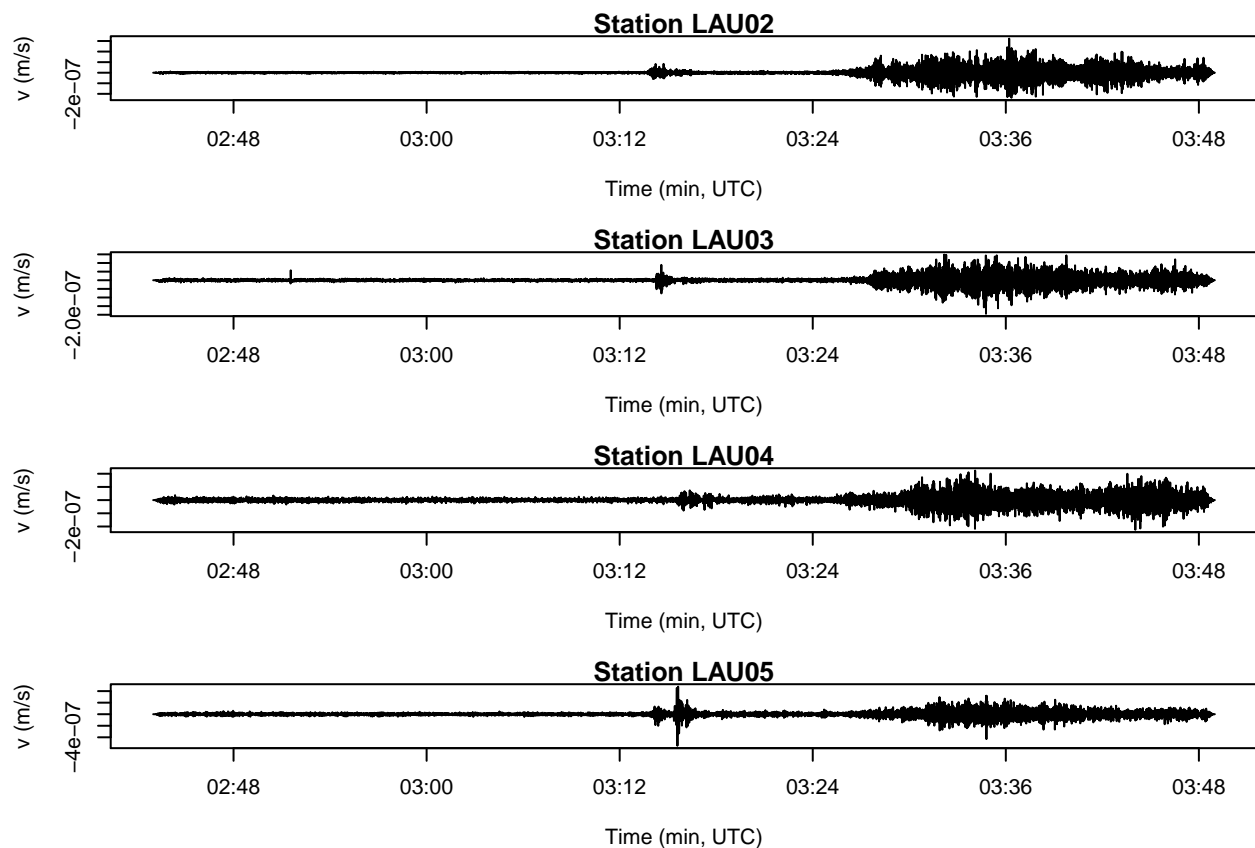
```
## filter (and taper) deconvolved signals
s_f_plot <- signal_filter(data = s_d,
                          dt = 1/200,
                          f = c(1, 90),
                          p = 0.01)

## prepare plot area
par(mfcol = c(4, 1), mar = c(5, 4, 1, 0))

## plot signals
for(i in 1:length(s_f_plot)) {

  plot(x = t,
       y = s_f_plot[[i]],
       type = "l",
       main = paste("Station", stations$ID[i]),
       xlab = "Time (min, UTC)",
       ylab = "v (m/s)")

}
```



Calculating and plotting power spectral density estimates (PSD)

```
## truncate the signal and time vectors to the actual event
t_lim <- as.POSIXct(c("2014-09-25 07:03:12 UTC",
                     "2014-09-25 07:03:49 UTC"),
                  tz = "UTC")

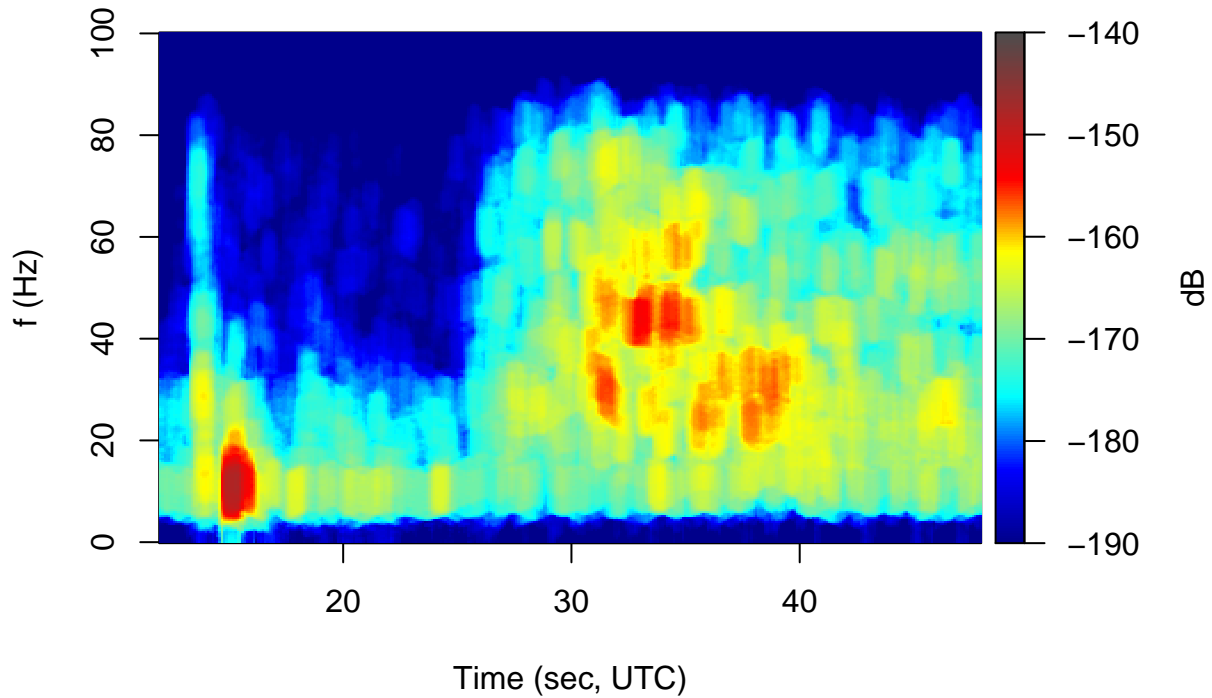
t_cut <- t[t >= t_lim[1] & t <= t_lim[2]]
s_cut <- lapply(X = s_f_plot, FUN = function(x) {

  x[t >= t_lim[1] & t <= t_lim[2]]
})

## calculate PSD
PSD <- signal_spectrogram(data = s_cut,
                          time = t_cut,
                          dt = 1/200,
                          Welch = TRUE,
                          window = 1.0,
                          overlap = 0.9,
                          window_sub = 0.8,
                          overlap_sub = 0.9,
                          multitaper = TRUE)

## plot PSD of station LAU05
plot_spectrogram(data = PSD[[4]],
                 legend = "dB",
                 zlim = c(-190, -140),
                 xlab = "Time (sec, UTC)",
                 ylab = "f (Hz)",
                 main = "PSD, station Funny Rain")
```

PSD, station Funny Rain



Locating events

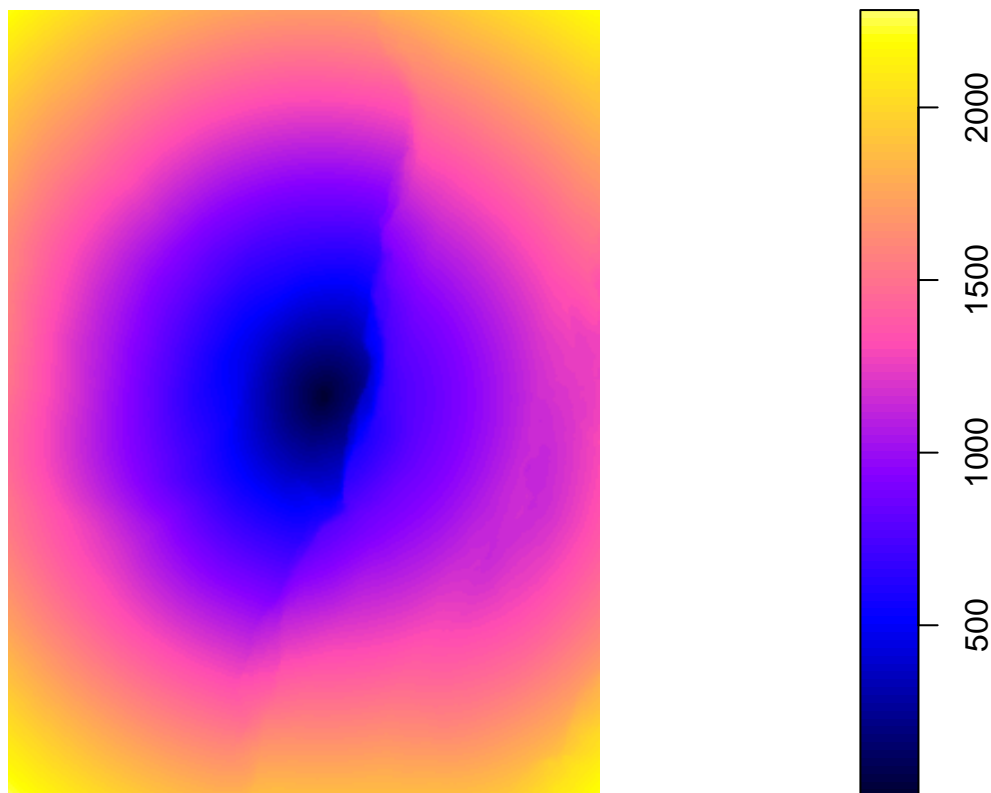
Locating events requires a DEM (see above). This DEM is used for the topography correction process. The workflow requires first to generate distance maps and inter-station distances, which are both required in the next step: signal migration.

Generate distance maps and inter-station distance matrix

```
## Loading required package: sp
## [1] "Processing station distances"
## generate distance data sets
D_data <- spatial_distance(stations = stations[,3:4],
                           dem = dem)

## plot distance map for station LAU02
plot(D_data$maps[[1]])

## show interstation distances
print(D_data$stations)
```



```
##           1           2           3           4
## 1    0.0000 1479.734 1625.5933  846.7072
## 2 1479.7343    0.000 3454.8019 3212.6337
## 3 1625.5933 3454.802    0.0000  998.1759
## 4  846.7072 3212.634  998.1759    0.0000
```

Migrate the signals

```
## filter signals to localisation window
s_migrate <- signal_filter(data = s,
                           dt = 1/200,
                           f = c(10, 20))

## calculate envelopes
s_migrate <- signal_envelope(data = s_migrate)

## convert signal list to matrix
s_migrate <- do.call(rbind, s_migrate)

## truncate time and signals to event +/- 2 seconds
t_lim <- as.POSIXct(c("2014-09-25 07:03:11 UTC",
                     "2014-09-25 07:03:21 UTC"),
                  tz = "UTC")

t_migrate <- t[t >= t_lim[1] & t <= t_lim[2]]
s_migrate <- s_migrate[,t >= t_lim[1] & t <= t_lim[2]]

## migrate the signal of rockfall 7
```

```

P <- spatial_migrate(data = s_migrate,
                     d_stations = D_data$stations,
                     d_map = D_data$maps,
                     v = 2700,
                     dt = 1/200,
                     normalise = TRUE)

## [1] "No snr given. Will be calculated from signals"
## remove values below likelihood threshold
P_min <- 0.97
P@data@values[P@data@values < quantile(P@data@values,
                                       P_min,
                                       na.rm = TRUE)] <- NA

## plot output as hillshade overlay and add lidar points
plot(hs,
     col = grey.colors(250),
     legend = FALSE,
     ann = FALSE,
     axes = FALSE)
plot(P,
     add = TRUE,
     col = adjustcolor(col = rev(heat.colors(250)),
                       alpha.f = 0.5))
points(x = lidar$x, y = lidar$y)

```

