May 2022
update 26 Jan 2024

Example 2: AutoStatsQ – Using local waveform data and metadata

Download example data (1 GB): https://data.pyrocko.org/examples/autostatsq_example.tar

The example directory contains:

- **Configuration file**: `AutoStatsQ_settings.config`
- **Event catalog**: events.txt
    - 39 synthetic events with M 7.0-7.5
- Pyrocko **station file**: some_stations_testset.pf
- Flat **response functions** of the synthetic stations: `stations_flat_displacement.xml`
- **Continuous synthetic waveform data:** /data_sds
    - ~50 days for all stations; velocity-waveforms;
    - sampling rate: 0.5 Hz
    - **sds data structure**

While the names and locations of the stations are taken from existing stations, the waveforms are synthetic. That means the errors, which we will find (misorientation and gain) were synthetically added on purpose and do not reflect the quality of the truely existing stations.

This tutorial has three main parts:

1. A comprehensive explanation of the entire config file, including best practices for all settings.
2. A step by step guide to run AutoStatsQ.
3. A section with commonly observed problems and FAQs.

We recommend to first have a look at the different AutoStatsQ settings (part 1) and then to follow the step by step instructions to run AutoStatsQ (part 2).
The remaining parts of this tutorial are dedicated to common errors and questions.

This tutorial is work in progress.

I am happy to receive your comments, questions, suggestions and corrections.

Table of Contents

0. What can and cannot AutoStatsQ do?

All tests implemented in AutoStatsQ can be run with various settings. These settings must be adjusted to obtain best results for your set of stations.

The texts below provide only a short overview on what AutoStatsQ can do. Please see Petersen et al. 2019 for more complete methodological information and discussion.

Amplitude gain test:

- Several methods are implemented to check amplitude gains. Most commonly, P wave amplitudes (default is first arriving P phase) are compared either to a reference station, or to amplitudes of synthetic data. In both cases, the median result of all azimuthally distributed events should be considered to avoid bias from single noisy events or travel-path effects. Smaller amplitude differences can result from site-effects, while larger amplitude errors are more likely to result from instrument or meta data errors.
- AutoStatsQ has proven to be helpful to identify stations with **significantly wrong amplitude values**. From our experience, such errors often originate from errors in the meta data, like **missing or wrong amplification factors**. These errors are often, but not always consistent among the same network, and can be in the order of amplitude offsets with factor ten to thousands. Therefore such errors are easily identified and detected.
- **Smaller amplitude offsets**, in the order of factor <10 (or >0.1) are more difficult to detect and to interpret. The test is using teleseismic P phases. If at least ~10 well distributed events are used, and if the median is considered instead of the mean, influences of the travel-path can be ruled out. The mean and standard deviation will of course still be influenced from travel path variability.
- **Site effects** can lead to significantly higher or lower amplitudes compared to a reference station with other site conditions and to synthetic data. AutoStatsQ was developed having Moment Tensor inversions in mind. In such an inversion, synthetic data is fitted to match the observed waveforms. Offsets of amplitude gains result in large misfits, no matter if they result from transfer function errors, station problems or site effects.
- **AutoStatsQ outputs the median gain amplitude ratio, but it cannot determine the reason for suspicious values.**

PSD test:

- Power spectral density plots (PSDs) are computed for synthetic and observed data of all (deep) events and stations. PSDs of synthetic and observed waveforms are calculated for all station–component–event combinations in long time windows (>30 min). For each pair of synthetic and observed PSDs, the ratio of both PSDs is calculated. Subsequently, the median power ratio over all events is computed for each frequency band of each station and component. Frequency ranges with a stable PSD ratio are determined by line fits to a given number of frequency ratio points (default 25). Successive lines having a slope below a given threshold (default 10) set up the recommended frequency ranges. Higher thresholds and fitting more frequency ratio points can be used to search for large misfits due to erroneous amplitude gains or extremely noisy environments, as well as to verify the instrument corner frequency.

- The comparison of spectra of observed and synthetic waveforms can be performed as a **second check of large amplitude gain offsets**, in this case **across a wider range of frequencies**.
- Furthermore, **locked components** or components with **very high noise levels** can be identified visually from the output plots.
- Deviations between the synthetics and observed spectra at lower frequencies can serve as a **check of instrument corner frequencies**. If e.g. station meta data imply a lower instrument corner frequency of 100 s, but observed waveform spectra only match the synthetics at above ~30 s, the corner frequency is most likely wrong.
- **The test provides a frequency range, for which the syn. and obs. spectra are similar or have a constant offset value.**
- This test requires a manual check of the spectra plots before interpretation.

Orientation test:

- Rayleigh waves have a 90° phase shift between the vertical and the radial component. Waveforms of a set of shallow, teleseismic events are analyzed by rotating the radial traces in steps of 1°. For each step, the cross-correlation of the Hilbert-transformed Z component and the rotated R component is calculated. Correctly oriented sensors would result in a maximum cross-correlation value at 0° rotation.
- With rather coarse default settings and without removal of outliers or manual checks of waveforms, the orientation test can identify stations with misorientations >20° with uncertainties usually in the order of <10°. However, when adjusting the cross-correlation threshold (default 0.8) to higher values (e.g. 0.95) and using more events (>30), the standard deviation can be significantly reduced (<6°).
- In addition, the median value instead of the mean should always be considered as the final result to reduce the effects of outliers, which may originate from path effects. Plotting the obtained correction angles from the single events over backazimuth helps to identify dependencies of the results from backazimuth directions.
- The orientation test can easily identify components with switched polarities, e.g. North pointing to South, which corresponds to a correction value of 180°. Interchanged components (N-S is E-W & E-W is N-S) are also detected as correction angles of around 90°.
- In future, it is planned to add an option to compare the results of neighboring stations for each event. This will also reduce the effect of travel paths, because the travel paths of Rayleigh waves from teleseismic distances are very similar for closely located stations.
- Plots showing the results from single events over time help to identify temporal changes, as seen e.g. in chapter 3.2.2.

Timing test:

- The timing test uses a comparison of waveforms via cross-correlations, relative to other stations and to synthetic data. The resolution abilities is therefore limited by the sampling frequency of the Green's function database and the downsampled data (2s by default). Only errors in the order of >4 s are detected in this default mode. Errors in the origin time of the catalog are considered and do not influence the result.

1. The Configuration file:

The config file is used to provide all needed information to AutoStatsQ, including for example the stations which we want to check, the time range and the choice to download data or check local data. In addition it is used to specify which tests we want to run.

In the example config file, many options are set to **`false`** to avoid running the entire program in a single run. After the explanation of all config file settings, you will find a section where running AutoStatsQ is described – including setting the according parameters to **`true`** .

The config files are in yaml format, which can easily be read by python codes. "**#**" indicates comments in the config file.

1.1. General settings:

Every file starts with a section for the general settings, which are the working directory path and the path to the list of stations which we want to check:

```
--- !autostatsq.config.AutoStatsQConfig
Settings:
- !autostatsq.config.GeneralSettings
  work_dir: '.'
  list_station_lists: [some_stations_testset.pf]
```

In case of this example, the working directory is the directory in which the config file is located. Therefore **`work_dir`** ist set to **`'.'`**.
The list of stations is provided in the **`some_stations_testset.pf`** file. **`list_station_lists`** can take multiple files with stations as input (e.g. **`[file1.pf, file2.pf]`**).

AutoStatsQ can be run with single stations up to large networks:

- The orientation test uses a comparison between R and Z component of each single station, and in the PSD test, the observed traces are compared to synthetics. Both tests are independent to the the number of tested stations.
- For the gain test, the default mode is a comparison with respect to a reference station. This does only make sense, if at least about five stations are tested. Otherwise the test should be run with another method, comparing the amplitudes to synthetic data (see chapter 1.7.1).
- The timing tests uses corrections for wrong origin times which are determined from a comparison of theoretical and true arrival times at all stations. This correction is needed to avoid bias from wrong catalog times and can only work for at least 5 stations.

1.2. Catalog:

The next part of the config file contains the information on the earthquakes which are used to run the tests. AutoStatsQ uses subsets of deep and shallow events for the different tests which rely either on body waves or on surface waves, respectively.

The search settings (e.g. time range, backzimuth steps, min. magnitude) define how many events will be used in the tests. AutoStatsQ tries to find azimuthally well distributed events. In general, the more events are used, the better. The exact number of required events depends on the task – Do you want to find suspicious stations and investigate further in a second step, or do you want to obtain

correction factors/ angles as good as possible.
In general, a number of 10-15 events is usually fine for a first run to check for larger errors. To obtain for example stable corrections for the sensor orientation, we would want to use >20 events, but this could also be done in a second run if we know that there are misoriented stations from the first run.

If the time period of the stations is limited, it may help to follow the steps suggested in chapter 5.

Instead of downloading a catalog, it is also possible to use an **existing catalog** in pyrocko-format. The path to these locally stored catalogs need to be provided in the **`subset_fns: {}`** section**.**

```
- !autostatsq.config.CatalogConfig
  search_events: false
  # search gCMT catalog for events? We want to search for events, because
      we do not have a catalog of earthquakes yet. Therefore we later, in
      the step by step run example, set this to true.

  use_local_catalog: false
  # If we would have a catalog, we could add the path here.

  subset_of_local_catalog: true
  # This will generate the deep and shallow subset of the catalog which we
      will download.

  use_local_subsets: false
  # If we have already subsets of some catalog downloaded, we would
      indicate “true” here and set the path in the next option
      “subset_fns”.

  subset_fns: {}
  # e.g.
  # {'deep': 'catalog_deep_subset.txt',
  # 'shallow': 'catalog_shallow_subset.txt'}
  # In our example, we download a new catalog, so we keep this empty.

  # Settings for searching GCMT for teleseismic events:
  min_mag: 7.0  # should be large enough to observe in far distance
                # e.g. M>7 for distances larger than 4000 km
  max_mag: 8.0  # should not be too large (M<8) because a point source
                # approximation is used in forward modeling of synthetic
                # waveforms.

  tmin_str: '2015-01-01 00:00:00'
  tmax_str: '2021-12-01 00:00:00'
      # tmin and tmax must be defined according to the run time or the
      # time of interest of your stations. Generally it holds that the
      # longer the time span, the more teleseismic events are available.
  min_dist_km: 4000.0
      # The minimum distance to events should be significantly larger than
      # the network aperture to perform the inter-station comparison in
      # the gain test. If you do not perform that test, closer events can
      # be used. For the orientation test and to compare gains and PSDs to
      # synthetic data, regional events can in theory be included.
  max_dist_km: 9999999.0
      # Maximum event distance
  depth_options:
    deep: [25000, 600000]
      # [m] Depth range of earthquakes to emphasize body waves, used for
      # gain check, PSD comparison and timing test.
```

```
      shallow: [100, 40000]
        # [m] Depth range of earthquakes to emphasize Rayleigh waves, used
        # for the sensor orientation check.



    wedges_width: 10
        # Backazimuthal step for subset generation – see also plot in
        # chapter 2.1.
        # Adjust to smaller steps (1 deg) to get more events. This can
        # especially be helpful in case of short time ranges or to check for
        # a backazimuthal dependency of the results.

    mid_point: [46.98, 10.74]
        # Give a rough estimate of the midpoint of your array or network.
        # This is optional, if the midpoint is not provided a geographic
        # midpoint is calculated of all station locations is computed.
        # This midpoint is used to compute the distance to the teleseismic
        # events.

    ### Catalog plotting options ###
    plot_catalog_all: false
        # Plots the entire downloaded catalog on a map.

    plot_hist_wedges: false
        # Plot statistics of the catalog.

    plot_catalog_subset: false
        # Plot the deep and shallow selected subsets on maps.
```

1.3. Arrival time computation:

The next section of the config file defines the computation of the arrival times of body and surface waves. By default, the very first arriving P phase is used and the arrival time of a Rayleigh wave with v=4.0 km/s. However, both can be adjusted for example when only events in a certain, closer distance range are used. The P phase can be defined using the **`phase_select`** with pyrocko cake terminology. (Type "**`cake list-phase-map`**" into your terminal to get a list of phase names mapped to cake terminology.)

```
  - !autostatsq.config.ArrTConfig
    calc_first_arr_t: false
    phase_select: P|p|P(cmb)P(icb)P(icb)p(cmb)p|P(cmb)Pv(icb)p(cmb)p|
P(cmb)P<(icb)(cmb)p
    calc_est_R: false
    v_rayleigh: 4.0
```

1.4. Using local waveform data and station meta data:

The next config file section contains the settings to either download data and metadata – or to use local data. In this example we use waveform data and metadata, which are locally stored. Please refer to the other example instruction file for downloading data and metadata from an fdsn server. The settings that need to be adjusted are marked with yellow background color. 'local_data' and 'local_metadata' contain the paths pointing to the local data and metadata directories. Set 'local_waveforms_only' to True if you do not want to use any downloaded data. 'sds_structure' defines if

the local waveform data is stored in ‘SeisComP Data Structure’ directories (`<SDSdir>/Year/NET/STA/CHAN.TYPE/NET.STA.LOC.CHAN.TYPE.YEAR.DAY`). AutoStatsQ can also work with other directory structures, but will be significantly slower, because all files need to be searched to find the according data. Support for other directory set-ups could be added, but are currently not implemented.

```
- !autostatsq.config.MetaDataDownloadConfig
  download_data: false
      # set to true to download data
  download_metadata: false
      # set to true to download metadata

  local_data: []
      # Lists of paths to include local data.
  local_metadata: []
      # Lists of paths to include local station meta data.
  local_waveforms_only: true
      # Set to true to use only local waveforms, do not download data.
  sds_structure: false
      # Set to true if you have local waveform data which is saved in sds
      # structure directories.

  use_downmeta: false
      # Use the downloaded metadata, can just stay true at all times.

  channels_download: HH*
      # Define channels to download. * is wildcard.
  all_channels: false
      # Or indicate to download all channels of a station (not
      # recommended, can be time consuming).
  sites: [bgr]
      # Define fdsn server from which data is downloaded. Available are:
      # geofon, iris, orfeus, bgr, geonet, knmi, ncedc, scedc, usgs, bgr,
      # koeri, ethz, icgc, ipgp, ingv, isc, lmu, noa, resif, usp
  token: {}
      # If data is restricted, you can add a token here, the format is:
      # {‘geofon’: ‘/path/to/token’}

  dt_start: 0.1
      # Start of download before origin time in hours.
  dt_end: 1.5
      # End of download after origin time in hours.
      # The time window should be long enough to contain body and surface
      # wave phases and to allow bandpass filtering in low frequency
      # ranges.
```

1.5. Preprocessing (Downsampling, Restitution, Rotation):

In order to run the tests, AutoStatsQ needs restituted data, which means the instrument response of the seismometer is removed from the waveform signal. In addition, the traces are rotated into Z,R,T coordinate system, taking into account the channel information of the response file. Finally, the waveform data is downsampled, in oder to reduce computation time. If a test relying on the comparison of synthetic and recorded data is run, then the downsampled rate needs to be the same as the sampling rate of the Green’s function database (see step 6).

```
- !autostatsq.config.RestDownRotConfig
  rest_data: false
      # set to true to start restitution
  freqlim: [0.005, 0.01, 0.2, 0.25]
      # Frequency filter for restitution; first two entries give lower
      # corner frequency, last two entries are the upper corner
      # frequencies. Here chosen to match the sampling frequency after
      # downsampling (2 s).
  rotate_data: false
      # set to true to rotate to ZRT system
  deltat_down: 2.0
      # Sampling rate after downsampling, in [s].
```

1.6. Synthetic data:

The next section of the config file defines if and how synthetic data should be computed. The computation is based on the pyrocko precalculated Green's function databases, see also https://pyrocko.org/docs/current/apps/fomosto/. Default is to use the `global_2s` store, which is can be downloaded using pyrocko's `fomosto` tool in your terminal:

```
fomosto download kinherd global_2s
```

(see also https://greens-mill.pyrocko.org/).

Synthetic data is used for the PSD test, where spectra of synthetic and recorded data are compared. In addition, synthetic data can be used in the gain test, for a comparison of P wave amplitudes. However, the amplitude gain test can also be run relative to a reference station.

Synthetic data only needs to be computed if one of the aforementioned tests shall be performed. Otherwise, the Green's function database does not need to be downloaded.

In this example, you can decide if you want to run the PSD test. If so, please download the GF database. This might take some time depending on your internet connection. It is also possible to skip this step.

```
- !autostatsq.config.SynthDataConfig
  make_syn_data: false
  # start computation of synthetic waveform data
  engine_path:  path/to/gf_stores
  # path to GF store directory
  store_id:  global_2s
  # Name of the store which you want to used.
```

1.7. Quality checks:

AutoStatsQ currently contains three well tested quality checks and two which are still in development. In this example, we will only run the three stable checks on P wave amplitude gains,

PSDs and sensor orientations. For in-depth methodological descriptions, explanations and discussions, please refer to Petersen et al., 2019.

1.7.1 Amplitude gain test:

The first test section contains the settings for testing the amplitude gains. Several methods are implemented. Most often, maximum P wave amplitudes are compared either to a reference station, or to the amplitudes of synthetic data. The used phase is defined by the phase settings used to compute the arrival time (see 1.3 Arrival time computation).

In case of both, the comparison to a reference station and to synthetic data, the median result of all azimuthally distributed events is considered instead of the mean to avoid bias from single noisy events or travel-path effects.

While small amplitude differences can result from site-effects, larger amplitude errors are most often resulting from instrument or meta data errors.

```
- !autostatsq.config.GainfactorsConfig
  calc_gainfactors: false
      # start running the gain test

  gain_factor_method:
  - reference_nsl
  - [GR, BFO]
      # Sets the gain factor method. In the case of this example, the P
      # amplitudes are compared to the reference station GR.BFO. See below
      # this config section for all methods & recommendations.

  fband:
    corner_hp: 0.01
    corner_lp: 0.2
    order: 4
      # Settings for the bandpass filter, filtering is performed before
      # cutting the time windows. The passband of the filter can be
      # relative wide as in this example. It only needs to be adjusted in
      # few cases: In case of sensors with instrument lower corner
      # frequencies above 0.01 Hz, the corner_hp should be adjusted. And
      # in case of very noisy data, it may help to lower the corner_lp.
      # The filter settings can easily be checked visually by using the
      # debug_mode (see below).

  wdw_st_arr: 120
      # Start of time window before phase arrival time.
  wdw_sp_arr: 60
      # End of time window after phase arrival time. The time window
      # should be long enough to contain the full first arriving phase
      # even after filtering and allowing for slightly wrong arrival
      # times. Here it also includes some of the secondary arrivals.
      # Depending on the distance of the event to the station, these
      # phases may have larger amplitudes and therefore be more
      # appropriate to compare.
  taper_xfrac: 0.25
      # Taper of time window: fade in and fade out as fraction between 0.
      # and 1. of pyrocko.trace CosFader – can remain at default.

  snr_thresh: 1.0
```

```
    # SNR threshold, adjust only in case of many noisy events. Due to
    # using the median instead of the mean of all event's results, the
    # SNR threshold does not need to be strict.

debug_mode: false
    # Start an interactive debugging mode. This will open the used
    # waveform data with time window and filter settings as selected
    # above. Helpful to debug in case of suspicious results – but should
    # be turned off to run entire test, because it slows down the
    # process significantly and needs manual interaction.

components: [Z]
# component(s) to run the test on

plot_median_gain_on_map: false
# plot the gain test result on a map
plot_allgains: false
# plot single gain test results for each station & event
# (simple x-y-plot)
```

Gain check methods:

(a) relative to a reference station: **← We use this method in the example.**

```
gain_factor_method:
- reference_nsl
- [NET, STAT]
```

→ Should only be used if the distance of the network stations to the reference stations is small compared to the event distance. We are using a teleseismic catalog with a large minimum distance, therefore it is safe to assume that when neglecting site effects, the amplitudes of an event should be similar at the different stations.

→ In case of a short operation time period the number of large, teleseismic events might not be sufficient to run this test. Then the amplitude gain test should rather be based on closer, moderate events and be compared to synthetic amplitudes.

→ Can also be used to study site effects on stations.

(b) compared to synthetic data:

```
gain_factor_method: ['syn_comp']
```

→ Compares the amplitudes of each event at all stations to the amplitudes of the synthetic waveforms.
→ Can also be used when using closer, moderate earthquakes.

(c) compared to median of all events:

```
gain_factor_method: ['median_all_avail']
```

→ Compares the amplitudes of each event at all stations to median of all stations.

1.7.2 PSD test:

Power spectral density plots (PSDs) are computed for synthetic and observed data of all (deep) events and stations. PSDs of synthetic and observed waveforms are calculated for all station–component–event combinations in long time windows (>30 min). For each pair of synthetic and observed PSDs, the ratio of both PSDs is calculated. Subsequently, the median power ratio over all events is computed for each frequency band of each station and component.

Frequency ranges with a stable PSD ratio are determined by line fits to a given number of frequency ratio points (default `n_poly: 25`). Successive lines having a slope below a given threshold (default `norm_factor: 10`) set up the recommended frequency ranges. We recommend testing different values for these parameters. Higher thresholds and fitting more frequency ratio points can be used to search for large misfits due to erroneous amplitude gains or extremely noisy environments, as well as to verify the instrument corner frequency.

```
- !autostatsq.config.PSDConfig
  calc_psd: false
      # Start running the PSD check

  ### Complete time window:
  # We use a time window beginning before the first arriving P phase and
  # ending before the arrival of the fastest surface waves.

  dt_start: 60
      # Start of time window in [s] before first arrival time.
  dt_end: 120
      # End of time window in [s] before arrival of Rayleigh waves.


  ### Increments of time window to compute PSD:
  tinc: 600
      # Time increment (window shift time) to compute PSD.
  tpad: 200
      # Padding time appended on either side of the data windows.


  ### Line fitting parameters to determine frequency ranges with flat
  ### ratios between spectra of synthetic and observed data:
  n_poly: 25
      # number of points for line fit (see text above)
  norm_factor: 50
      # Line slope threshold (see text above)
  f_ign: 0.02
      # Frequency bandwidth to ignore in case of outliers/ instabilities
      # of ratio between observed and synthetic spectra.
  only_first: true
      # Output only the first, lowest frequency range with a stable ratio
      # between observed and synthetic data?


  ### Plotting:
  plot_psds: false
      # Plot PSDs of synthetic and observed data for each event and
      # station.
  plot_ratio_extra: false
      # Plot ratio of syn and obs PSDs for single events – only needed for
      #  debugging.
  plot_m_rat: false
```

```
    # Plot median ratio between syn and obs PSDs over frequency – only
    # needed for debugging
plot_flat_ranges: false
    # Output plot showing median ratio of PSDs and the frequency
    # range(s) for which this ratio is stable.
```

1.7.3 Orientation test:

The next part of the config files contains the settings for the orientation test.

Rayleigh waves have a 90° phase shift between the vertical and the radial component. Waveforms of a set of shallow, teleseismic events are analyzed by rotating the radial traces in steps of 1°. For each step, the cross-correlation of the Hilbert-transformed Z component and the rotated R component is calculated. Correctly oriented sensors would result in a maximum cross-correlation value at 0° rotation.
For example, a N component pointing towards N20E would result in a correction angle of -20°. The median of the correction angles of all events are used are the correction angle for one station. We recommend using the median instead of the mean of all events as the correction angle for a station, because the results of single events in one azimuthal direction may be biased by crustal structures. See Petersen et al. 2019 for more information.

For testing, we recommend to run the test using different filters (e.g. 0.01-0.05 Hz, 0.01-0.03 Hz, 0.02-0.05 Hz, etc) and minimum cross-correlation settings (e.g. 0.8, 0.85, 0.9, 0.05).

```
- !autostatsq.config.OrientConfig
  orient_rayl: false
  # Start orientation test

  bandpass: [3, 0.01, 0.05]
      # Bandpass filter [steepness, lower corner frequency, higher corner
      # frequency], should be chosen to emphasize Rayleigh waves. The
      # exact filter depends on the distance of the events to the station.
      # In our example, with distances >4000 km, frequencies of 0.01-0.03
      # Hz are dominant. In case of low cross-correlation values (due to
      # noise conditions) it can help to use a more narrow band, but this
      # may result in larger uncertainties originating for large cc values
      # for a wide range of similar rotations.

  start_before_ev: 30
  stop_after_ev: 600
      # Settings of time window, start before and end after theoretical
      # Rayleigh wave arrival. Must be long enough to contain large
      # portions of the surface waves. If too short, the waveforms can be
      # rather monochronous which can lead to mismatching of phases on the
      # Hilbert transformed R and the Z component.

  ccmin: 0.80
      # Threshold of cross-correlation value. If maximum cc value of an
      # event at a station is below this threshold, then this event will
      # be excluded from the analysis. Should be at least 0.8, but even
      # higher values provide more stable results.

  plot_heatmap: false
```

```
    # Plot heatmap showing cc value over correction angle for each
    # station and event. Usually the next, distribution plot is easier
    # to read.
plot_distr: false
    # Plot max. cross-correlation value over correction angle for each
    # station and event.
plot_orient_map_fromfile: false
    # Plot map with horizontal sensor orientation as arrows.
plot_angles_vs_events: false
    # Plot correction angle vs. event time for each station.
plot_angles_vs_baz: false
    # Plot correction angle vs. backazimuth to event for each station.


debug_mode: false
    # run test in debug mode, will open waveform data in used time
    # window and with filter in interactive waveform browser snuffler.
    # Only for debugging, otherwise slow.
```

1.7.4 Timing test:

The small implemented check for timing errors is based on the cross-correlation between the recorded traces and the synthetic vertical traces:
(1) First, for each event and station the two (syn. + obs.) traces are correlated to obtain the time shift for which the correlation is highest.
(2) In a second step the median time shift of each event over all stations is determined and the time shift values at the single stations are corrected for this median value. This is done to avoid errors from wrong origin times in the catalog, to take into account large deviations between origin time and centroid time, and to consider large path effects of the teleseismic test events which effect all stations in a similar manner.

The test is run in low frequency ranges (e.g. 0.01-0.10 Hz) and using synthetics computed from a global GFDB with a sampling of 2s. Therefore this test can only be applied to detect large timing errors in the order of several seconds. This error range is only useful to check prior to other seismological applications which use a similar frequency range as e.g. MT inversions.

```
- !autostatsq.config.TimingConfig
  timing_test: false
      # Start timing test.

  bandpass: [3, 0.01, 0.1]
      # bandpass filter for timing test,
      # [steepness, lower corner frequency, higher corner frequency]

  time_wdw: ['tP-120', 'tP+1200']
      # Time window – from 120 s before the first P phase until 1200 s
      # after that.

  cc_thresh: 0.6
      # Waveform cross-correlation threshold to consider event at a
      # station. As waveforms are manually selected, the threshold does
      # not need to be high.

  debug_mode: false
      # Interactive debugging mode.
```

1.7.5 TeleCheck:

This is a new, not well checked interactive test, which will not be covered in this manual. Simply leave tele_check: false.

```
- !autostatsq.config.TeleCheckConfig
  tele_check: false
```

1.8. Settings for plotting of maps:

In the last section of the config file, the maps to show the results of the orientation and gain tests can be adjusted:

```
- !autostatsq.config.maps
  map_size: [30.0, 30.0]
  pl_opt: ['automatic']  # [50, 8.3, 1000000, 'split']
      # [Center of map latitude, longitude, radius, gmt color scale name]
      # or  ['automatic'] to determine from data.
  pl_topo: false
      # If set to true, topographic data from ETOPO or SRTM will be
      # downloaded and used. This may slow down the plotting.
  outformat: png
      # Output format of the map (png or pdf).
```

2. Run AutoStatsQ example step-by-step:

2.1 Step 1: Catalog

We will now start with a very first run of AutoStatsQ. In this example, we use the catalog file events.txt which is provided in the example directory for all tests. Therefore, we set all search parameters to false.

```
search_events: false
use_local_catalog: false
subset_of_local_catalog: false
use_local_subsets: true
subset_fns: {'deep':'events.txt',
             'shallow': 'events.txt'}
```

and to also have a look at the output plots:

```
plot_catalog_all: true
plot_hist_wedges: true
plot_catalog_subset: true
```

We then run AutoStatsQ in the terminal:

```
autostatsq --config AutoStatsQ_settings.config --run -l INFO
```

The `-l info` defines the log level, the amount of terminal output while running AutoStatsQ.

Running this command, a new directory "*results*" is generated in your work directory. This directory contains a subdirectory "*catalog*", in which the catalogs and catalog plots are stored.

To avoid re-plotting the catalog plots when we continue, we set:

```
search_events: false
plot_catalog_all: false
plot_hist_wedges: false
plot_catalog_subset: false
```

2.2 Step 2: Arrival times

We now compute the **arrival times** of P and Rayleigh waves by setting:

```
calc_first_arr_t: true
calc_est_R: true
```

..and we rerun AutoStatsQ:

```
autostatsq --config AutoStatsQ_settings.config --run -l INFO
```

The calculation of arrival times may take few minutes. The arrival times are saved as numpy arrays in the /ttt directory. Next we set:

```
calc_first_arr_t: false
calc_est_R: false
```

…so that we do not rerun the calculation.

2.3 Step 3: Data & Metadata

In the example, we do not download waveform data and station meta data but use data that is stored locally in the directory './data_sds/with_errors/'. Therefore we add the paths to the directories that contain the data and metadata:

```
download_data: false
```

```
download_metadata: false

local_waveforms_only: true
    ### in this example we use only local waveforms, no download.
local_data: ['./data_sds/with_errors/']
sds_structure: true
    ### the locally stored data is in sds structure
use_downmeta: false
local_metadata: ['stations_flat_displacement.xml']
     ### in this example we use only local waveforms, no download.

channels_download: BH*
    ### component to test, synthetic example has BH channels
```

2.4 & 2.5 Step 4 & 5: Preprocessing and synthetic data:

Now we preprocess the data: Traces are rotated to ZRT components and the instrument response is removed. In our case, the sampling interval is already 0.5 Hz, therefore we set deltat_down to 0.0 which indicates that the downsampling is skipped.

```
rest_data: true
  freqlim:
  - 0.005
  - 0.01
  - 0.2
  - 0.22
rotate_data: true
deltat_down: 0.0
```

After running AutoStatsQ again (**`autostatsq --config AutoStatsQ_settings.config --run -l INFO)`**, three new directories exist in your working directory:

- `/rest`: Contains the restituted data (instrument response removed) – still with original sampling rate and component orientation.
- `/rrd`: Contains the restituted and downsampled data, rotated to ZRT.

After checking that these new directories exist, we reset the above parameters in the config file to `false`:

```
rest_data: false
rotate_data: false
```

→ → → **Now, all data is prepared to run the quality checks.**

Note, in this example we skip the computation of synthetic waveforms, which is needed for the PSD comparison.

2.6 Step 6: Running the tests:

Each test section (gain, PSD, orient, timing) has one parameter in the config file which indicates that the test should be run. These are:

- **calc_gainfactors**
- **calc_psd**
- **orient_rayl**
- **timing_test**

By setting them to **true**, the test will be started. Here we will only run the first and the third tests to check gains and misorientations.

2.6.1 Step 6.1: The amplitude gain test

We will run one test after another, starting with the amplitude gain check. We will run the test to check the vertical component amplitudes relative to the **synthetic** reference station GR.BRG. To start the test, we set the according parameters to true:

```
calc_gainfactors: true
gain_factor_method:
    ### in this example we compare to synthetic station GR.BRG
- reference_nsl
- - GR
  - BRG

fband:
  corner_hp: 0.01
  corner_lp: 0.2
  order: 4

taper_xfrac: 0.25

wdw_st_arr: 180
    ### make sure to include time window before arrival of first phase
wdw_sp_arr: 60
    ### 60 s is enough to have first phase fully covered

components:
- Z

plot_allgains: true
```

...and rerun AutoStatsQ:

```
autostatsq --config AutoStatsQ_settings.config --run -l INFO
```

The results are stored in the directory **/results/gains/**. Two output files are generated:

- `gains_median_and_meanZ.txt`: A file with the median, mean, standard deviation and number of used events for each station. Can be opened in any texteditor.
  - Median ratios are in the order of 0.9-0.99 for all stations except for GR.ASSE, where an gains for manipulated to show an example (94).
- `gains_all_eventsZ.txt`: A comma spread (csv) table containing the single results for each station and event. Best to open with a spreadsheet application.

Looking at the median gain rations of the different synthetic stations to GR.BRG, we see that the amplitudes are in the same order of magnitude at most stations. However, we included an error for one station, which has 100-fold increased gains:

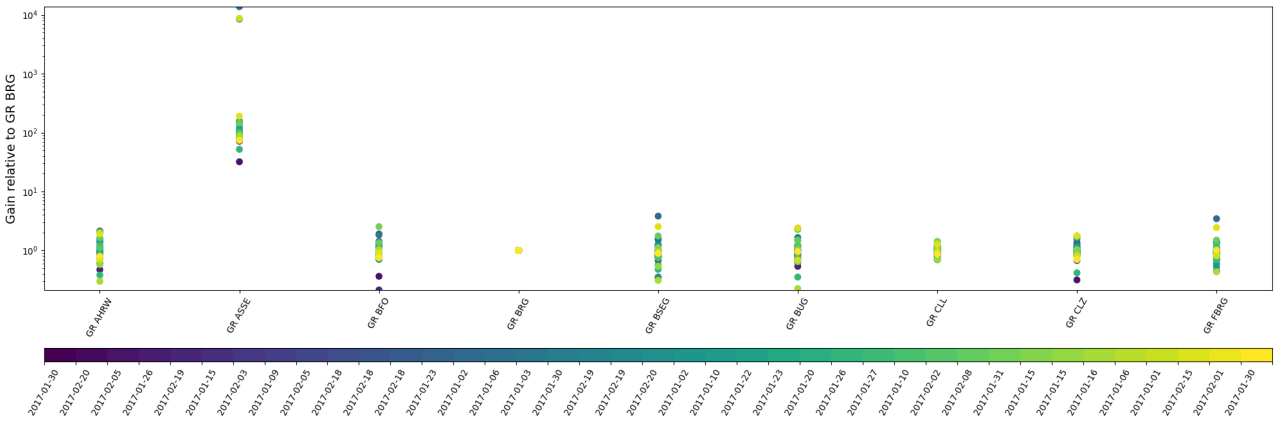

Fig. 1: Amplitude ratio of each station with respect to reference station BRG for all events. Values around 10^0=1 indicate very similar and therefore non-suspicious values. Note that the ***y axis is logarithmic.***
Small variations result from differences in station-event-backazimuths and random noise that was added to the waveforms.
(All waveforms synthetic, no real error.)
F

2.6.2 Step 6.2: The PSD test

We skip the PSD test here. Please refer to the other example for information and instructions of this test.

2.6.3 Step 6.3: The orientation test

To start the test, we set the PSD parameters back to false and the orientation test parameters to true:

```
orient_rayl: true
plot_distr: true
plot_orient_map_fromfile: true
plot_angles_vs_events: true
```

...and rerun AutoStatsQ:

```
autostatsq --config AutoStatsQ_settings.config --run -l INFO
```

The results of the tests are stored in the directory /results/orient, which contains different files:

- AllCorrectionAngles.yaml: Correction angles of all events obtained for every station.

- CorrectionAngles.yaml: Median, mean, standard deviation and number of events used for each station.
- *NET*_*STA*__distr.png: One figure for each station with subplots for each event showing the cross-correlation coefficient over correction angle for each event. All events with waveform data shown, even if below cross-correlation threshold.
- *NET*_*STA*_overtime.png: Correction angle for each event over event time. Can help to identify temporal changes e.g. due to a misorientation after swop of instruments.

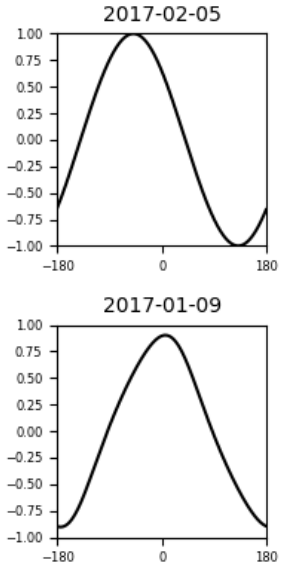

Fig. 2: Example plots showing coefficient of Z component and Hilbert transformed R component value over correction angle. The curve in the top panel shows that a correction of -50° is needed for the later events, while the sensor was correctly oriented on January 9th. A maximum at 0° indicates a station with perfectly oriented horizontal components. The 50 degree misorientation was introduced on purpose.*

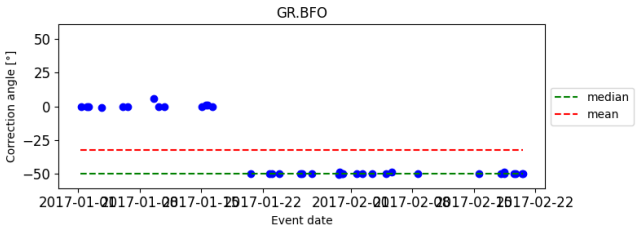

Fig. 3: Obtained correction angle for each event, synthetic station GR.BFO.The introduced shift in time between a correct orientation and a correction angle of -50° is clearly resolved.

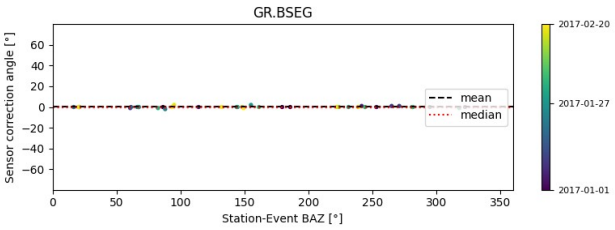

Fig. 4: Obtained correction angle for each event, sorted by back azimuth to station GR.BSEG (synthetic dataset). Grey area indicates the standard deviation around the mean (black line). Median as dotted red line. When using recorded instead of synthetic data, a backazimuthal dependence of correction angle can result from travel path effects of large tectonic features.

The result plots show a median and mean correction angle around 0° with very little scatter for all stations except for synthetic GR.BFO. In case of this example, we introduced a rotation of 50 degrees starting on the 16th of January.

In case of real, recorded data scatter of the single event results can have several reasons:

- Variations in the noise level → We have chosen a cross-correlation threshold of 0.8. By increasing the threshold or manually checking the waveforms, it is possible to exclude the events for which a station has a decreased similarity of Hilbert transformed R component and Z component.

- A too narrow frequency band can result in wide plateaus of correction angles with high cross-correlation values. If the frequency band is dominated by very low frequencies only, the resolution abilities can be limited.
  It can help to test different frequency bands, e.g. 0.01-0.05 Hz, 0.01-0.03 Hz, 0.02-0.05 Hz.

- The obtained angle for events from a certain back azimuth direction can be influenced by large scale subsurface features. In our case, the third plot does not show such a trend.

- The second plot, showing correction angle over event time, can help to understand if there is a jump over time, as for example due to maintenance work in the field including a correction of sensor orientation.

3. AutoStatsQ – result files, figures and html report

The above step by step instructions include examples to generate output files and figures.
Output files are in yaml and csv format. To read a yaml file for future processing use pyrocko's guts and import the top class of the file. For example, to read CorrectionAngles.yaml use this:

```
from pyrocko import guts
from autostatsq.orient import dict_stats_rota

cs = guts.load_all(filename='CorrectionAngles.yaml')

print(cs[0])
      # will output the entire file
print(cs[0].CorrectAngl_perStat_median)
      # outputs the median values as python dictionary:
      {'GR AHRW *': 0.0, 'GR ASSE *': 0.0, 'GR BFO *': -50.0, 'GR BRG *':
      0.0, 'GR BSEG *': 0.0, 'GR BUG *': 0.0, 'GR CLL *': 0.0, 'GR CLZ *':
      0.0, 'GR FBRG *': 0.0}
print(cs[0].CorrectAngl_perStat_median['GR BFO *'])
      # provides the correction angle for GR.BFO.* (again, this is a made-
      up example, the real station BFO is correctly oriented.)
      -50.0
```

In general, png or pdf figures (defined in the config file) can be generated for each plot. gmt (Wessel et al., 2013) is used to plot maps and all other plots rely on the python matplotlib library (Hunter et al., 2007). Plotting is optional, settings are made in the config file (see step-by-step instructions).

In addition to the figures and text files, AutoStatsQ can be used to generate an **html report**. The

report is generated using reveal (Copyright (C) 2020 Hakim El Hattab, http://hakim.se, and reveal.js contributors). The report combines figures, nice-to-look-at tables and background information to each method.

To generate a html report after running AutoStatsQ use the option:

```
autostatsq --config name_of_config_file --report
```

The report is saved in a directory result_report and the file index_report.html can be opened for example with firefox (**`firefox result_report/index_report.html`**).

Fig. 5: Front page of AutoStatsQ html report with interactive links to background information and relevant packages.

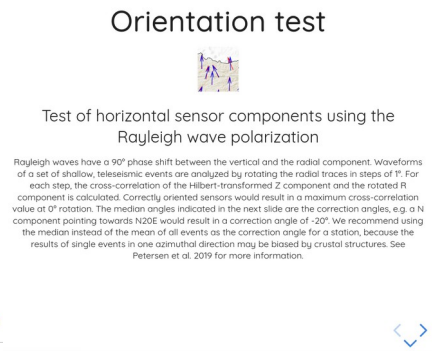

Fig. 6: First page of orientation test with test information.

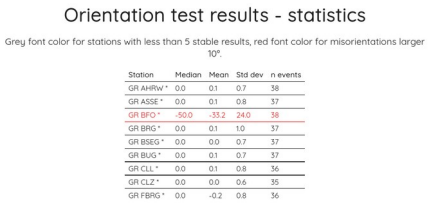

Fig. 7: Example result table in html report. Erroneous station emphasized.

4. Typically observed data and metadata error gallery

4.1 Gains & PSDs

4.1.2 Large offsets of PSDs

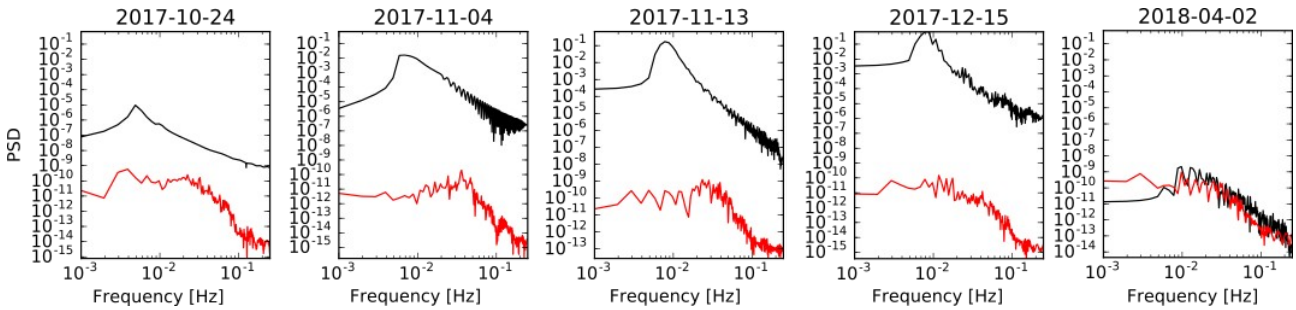

Fig. 8: Example of significant differences between the recorded and synthetic frequency spectra before the replacement of the acquisition system in March 2018. From Petersen et al., 2019.

4.2 Sensor orientation

4.2.1 Example: Inverse polarities of horizontal components

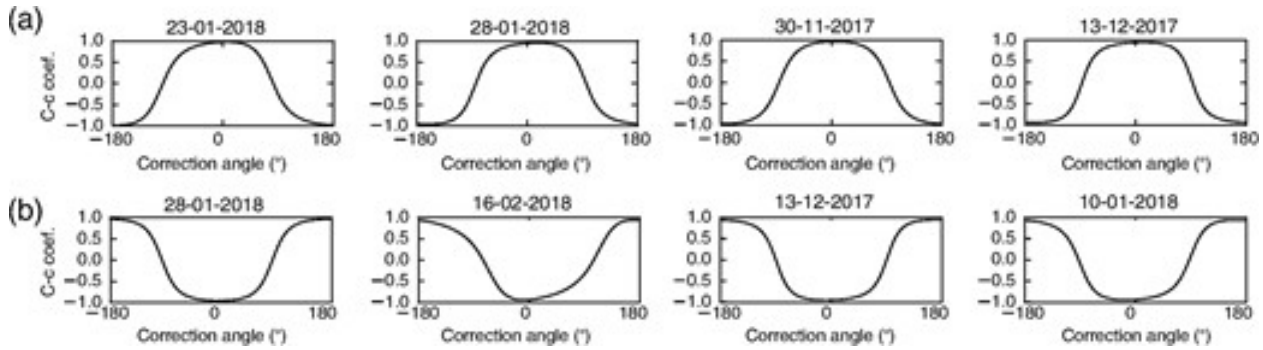

Fig. 9: (a) A correctly oriented sensor results in maximum cross-correlation values for a correction angle of 0°. (b) If a station has inverse polarities of the horizontal components, the resulting correction angle will be 180°. From Petersen et al., 2019.

4.2.2 Example: Temporal change of orientation:

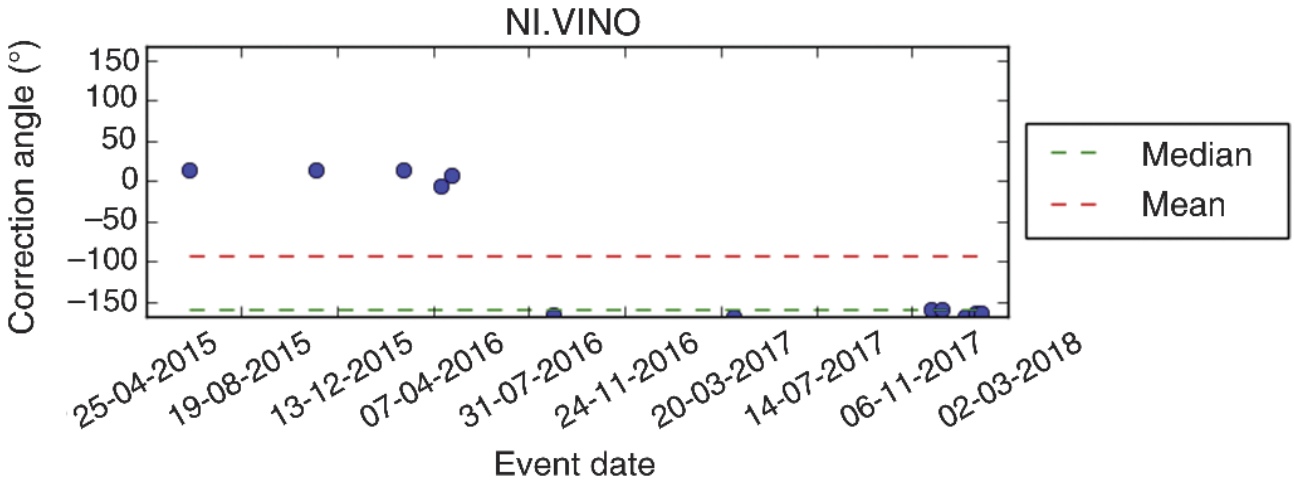

Fig. 10: Correction angle over time showing a temporal change of orientation. Such changes can be related to sudden flips of polarity or e.g. to maintenance work. From Petersen et al., 2019.

4.2.3 Example: Evaluation of misorientations from result map

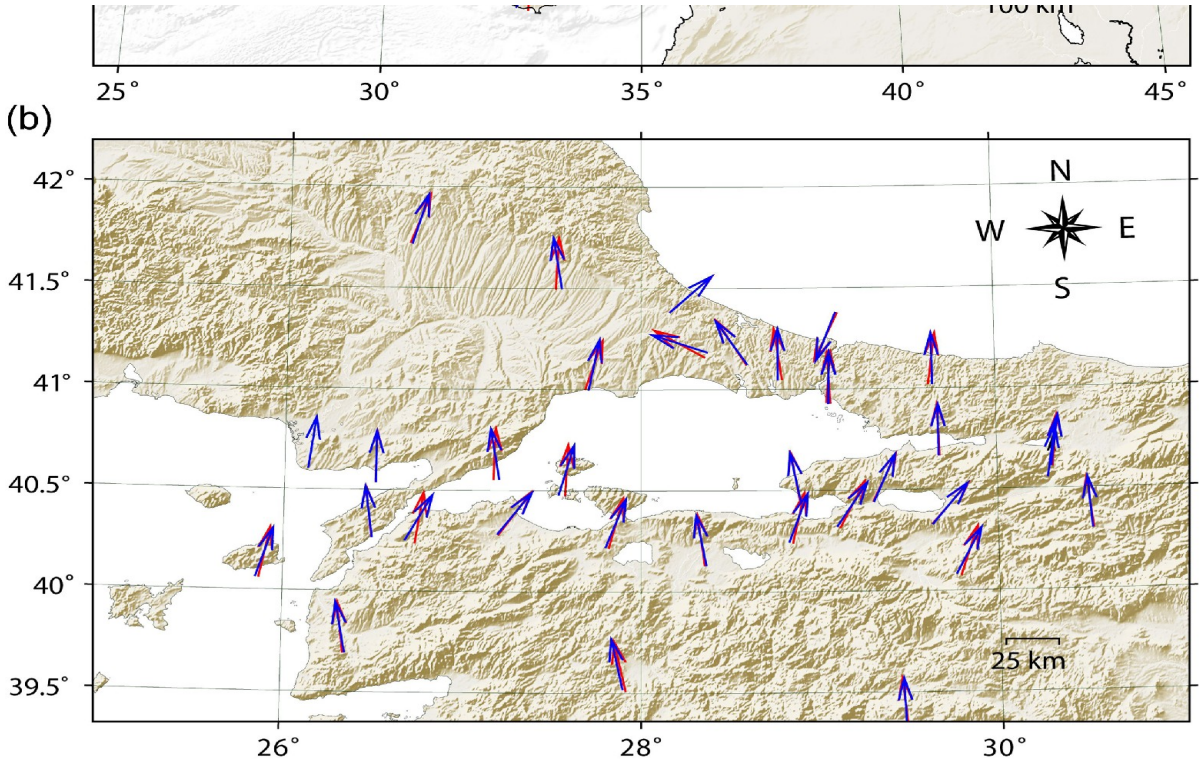

Fig. 11: Orientation of North components as obtained from AutoStatsQ (red) and a P polarization methods (blue). Arrows pointing not in North direction indicate misorientations. From Büyükakpınar et al., 2021.

5. AutoStatsQ commands overview:

- show basic commands/ help:

  ```
  autostatsq -h
  ```

- generate an example config file:

  ```
  autostatsq --generate_config
  ```

- run AutoStatsQ

  ```
  autostatsq --config name_of_config_file --run
  ```

- To get detailed error/ info logging, use the -l option:

  ```
  autostatsq --config name_of_config_file --run -l INFO
  ```

- Helpful for debugging: Forward terminal output into a logging file by using the option `--logoutput FILENAME`.

- To generate a html report after running AutoStatsQ use the option:

```
autostatsq --config name_of_config_file --report
```

The report is saved in a directory result_report and the file index_report.html can be opened for example with firefox.

6. FAQs:

My stations did only run for a few months so I do not find many teleseismic events. What do I do now?

- Set wedges_width to 1°
- Include earthquakes at shorter distances in the catalog search, these may also have smaller magnitudes.
- The orientation test can be run with closer and smaller events. The only requirement is that surface waves are generated and observed at the stations. At regional distances, even events with M>4 can be used. However, the test can be biased from travel path effects, if the earthquakes are not azimuthally well distributed.
- The PSD test can be run for region to local events, but for the generation of the synthetic data a more local Green's function database might be needed (see also https://greens-mill.pyrocko.org/).
- The gain test per default compares amplitudes among the stations of a network. This requires that inter-station distances are small compared to the distance to the earthquakes. However, if too little teleseismic events are found, local events can be used if switching to comparing to synthetics. As for the PSD test, for the generation of the synthetic data a more local Green's function database is required (see also https://greens-mill.pyrocko.org/).

References:

Büyükakpınar, P., Aktar, M., Maria Petersen, G., & Köseoğlu, A. (2021). Orientations of broadband stations of the KOERI seismic network (Turkey) from two independent methods: P-and Rayleigh-wave polarization. *Seismological Society of America*, *92*(3), 1512-1521.

Heimann, S., Kriegerowski, M., Isken, M., Cesca, S., Daout, S., Grigoli, F., Juretzek, C., Megies, T., Nooshiri, N., Steinberg, A., Sudhaus, H., Vasyura-Bathke, H., Willey, T., and Dahm, T. (2017). Pyrocko - an open-source seismology toolbox and library. v. 0.3. GFZ Data Services. http://doi.org/10.5880/GFZ.2.1.2017.001.

Hunter, J.D., 2007. Matplotlib: a 2D graphics environment, Comput. Sci. Eng., 9(3), 90–95.

Petersen, G. M., Cesca, S., Kriegerowski, M., & AlpArray Working Group. (2019). Automated quality control for large seismic networks: Implementation and application to the AlpArray seismic network. *Seismological Research Letters*, *90*(3), 1177-1190.

Wessel, P., Smith, W. H.~F., Scharroo, R., Luis, J.~F., and Wobbe, F. (2013). Generic Mapping Tools: Improved version released. EOS Trans. AGU, 94:409--410.