Tutorial 4:
Calculating Receiver functions

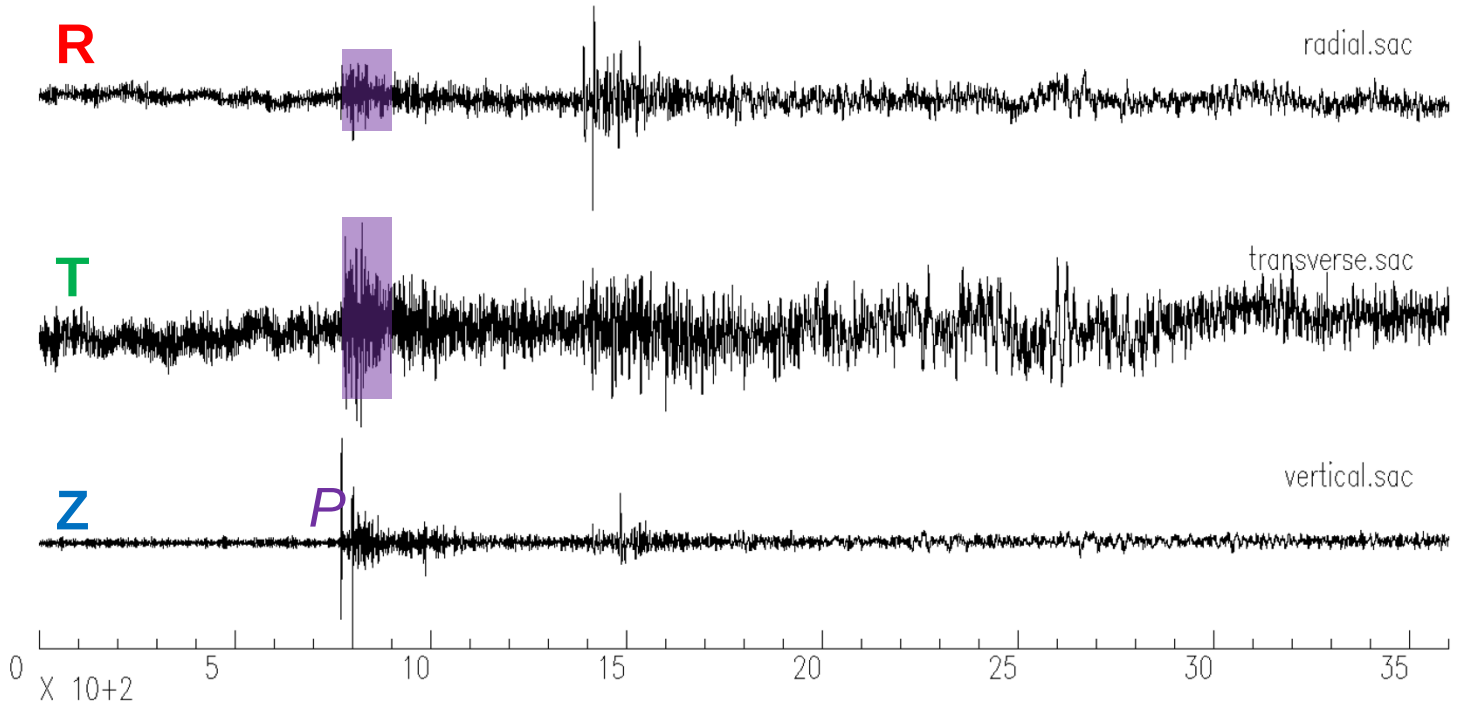
OBS training workshop, VUW, April 14-16, 2025

https://github.com/paudetseis

Software for calculating and post-processing teleseismic receiver functions:

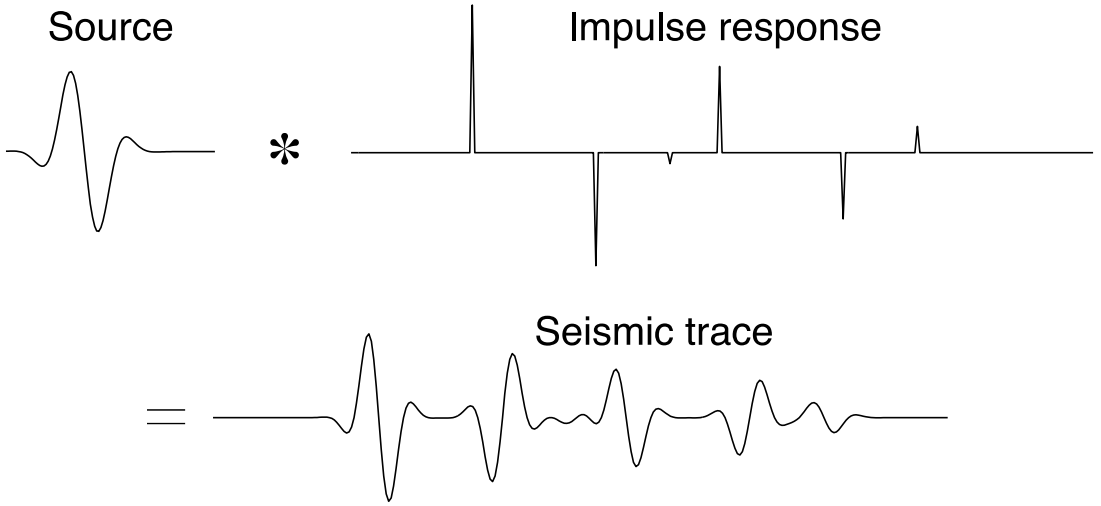
1. Automated workflow for calculating RFs from FDSN archives
2. Post-processing includes:
    1. Visualization
    2. H-k stacking
    3. Harmonic decomposition
    4. CCP stacking

Receiver functions: General principles



To understand **receiver functions**, note how the P pulse is followed by a *coda*, particularly on the horizontal components.

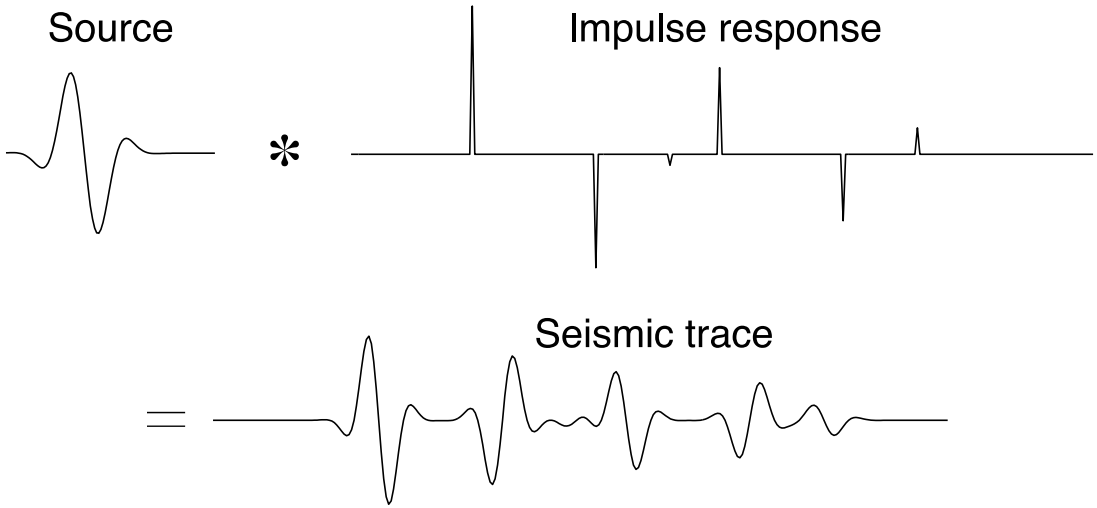
Receiver functions



$$g(t) = s(t) * w(t) = \int_{-\infty}^{\infty} s(\tau) w(t-\tau)\, d\tau \quad \leftrightarrow \quad G(\omega) = S(\omega) W(\omega)$$

- In the frequency domain, convolution is a *spectral product* – that is, any frequency present in both the source and the impulse response will be present in the data. But any frequency missing in the source will not, except as noise – it's impossible to recover frequencies that aren't in the source pulse.

Receiver functions



$$G(\omega) = S(\omega)W(\omega) \quad \text{so} \quad W(\omega) \approx \frac{G(\omega)S^*(\omega)}{S(\omega)S^*(\omega) + \delta}$$

- So to recover the impulse response, we need to perform a *deconvolution*, recovering the impulse response within the limits of the available frequencies. There are a number of algorithms for this (which I won't go into). Note that to deconvolve, you need an estimate of the source time function!

RfPy implementation

- Preprocessing:
  - Choose Event settings (start and end times, min and max magnitudes)
  - Choose Geometry settings (Phase ‘P’ or ‘PP’, min and max great arc circle distances
  - Choose Processing settings (sampling rate, window length, component alignment, near-surface Vp and Vs, SNR window length, pre-filtering options)
  - Choose Deconvolution settings (method, Gaussian filter, water level)
- ALL of these settings have default values

RfPy deconvolution

- Wiener: Use pre-event noise spectrum in regularization

$$W(\omega) \approx \frac{G(\omega)S^*(\omega)}{S(\omega)S^*(\omega) + N(\omega)N^*(\omega)}$$

- Water level: Set regularization as percent of maximum spectral amplitude

$$W(\omega) \approx \frac{G(\omega)S^*(\omega)}{S(\omega)S^*(\omega) + \delta * \max(|S(\omega)|^2)}$$

- Multitaper: Average spectra obtained from multiple orthogonal tapers

$$W(\omega) \approx \frac{\sum G(\omega)S^*(\omega)}{\sum S(\omega)S^*(\omega) + \sum N(\omega)N^*(\omega)}$$

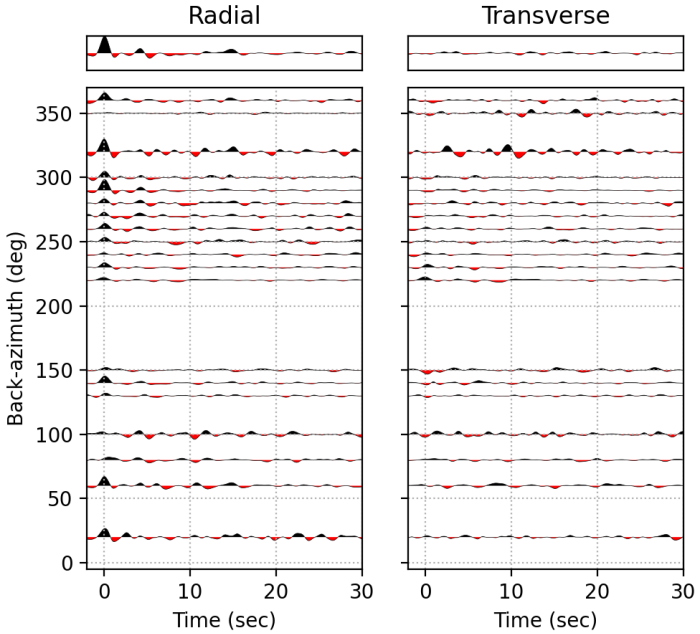
RfPy implementation

- Defaults:
  - Event:
    - Start and end times are those for the station
    - magnitude > 6.0
  - Geometry:
    - Phase is 'P'
    - distances between 30 and 90 degrees
  - Processing:
    - Sampling rate = 10 Hz
    - window length = 120 sec
    - component alignment = Z-R-T
    - near-surface Vp = 6 km/s; Vs = 3.5 km/s
    - SNR window length = 30 sec
    - No pre-filtering
  - Deconvolution:
    - Wiener

- Run `rfpy_calc` with the `stdb` station/network file to download the data and calculate RFs automatically
- Once the RFs are calculated, they can be re-calculated with different Processing and Deconvolution parameters using `rfpy_recalc`
- RFs can be visualized using `rfpy_plot` with various options (e.g., binning by back-azimuth or slowness or ray parameter, filtering, etc.)
- Quality controls for post-processed RFs are:
  - SNR: Signal-to-noise ratio on vertical (default > 5dB)
  - SNRh: SNR on radial horizontal (default > 0 dB)
  - CC: Cross-correlation between radial and predicted radial from convolution of Z with radial RF
  - Outlier: Outlier analysis of variance in RF signals within specified lag times

RfPy visualization

Back-azimuth binning



Slowness binning