**FUNCLAB (IN FUNCLAB2.1 FOLDER)**

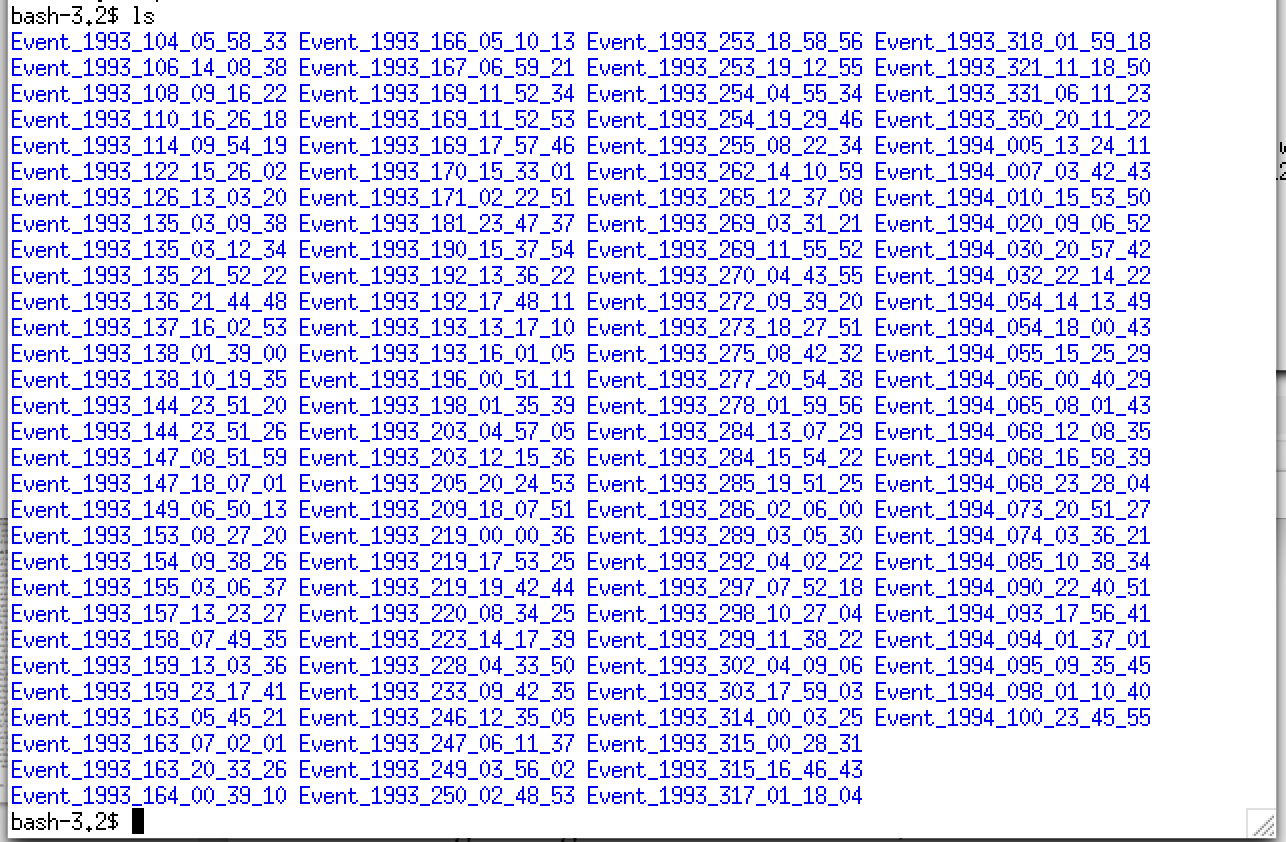
**Step 1: Formatting and loading data**

The Funclab package was designed by Kevin Eagar and Matthew Fouch (Arizona State University) to allow for easy data quality control and receiver function analysis techniques, such as H-k and CCP stacking. This package has been kept up-to-date by Robert Porritt, but this version is one I have personally edited to work in the workflow necessary for the joint inversion. There is also a manual for FUNCLAB in the PAPERs folder.

**Before loading data**

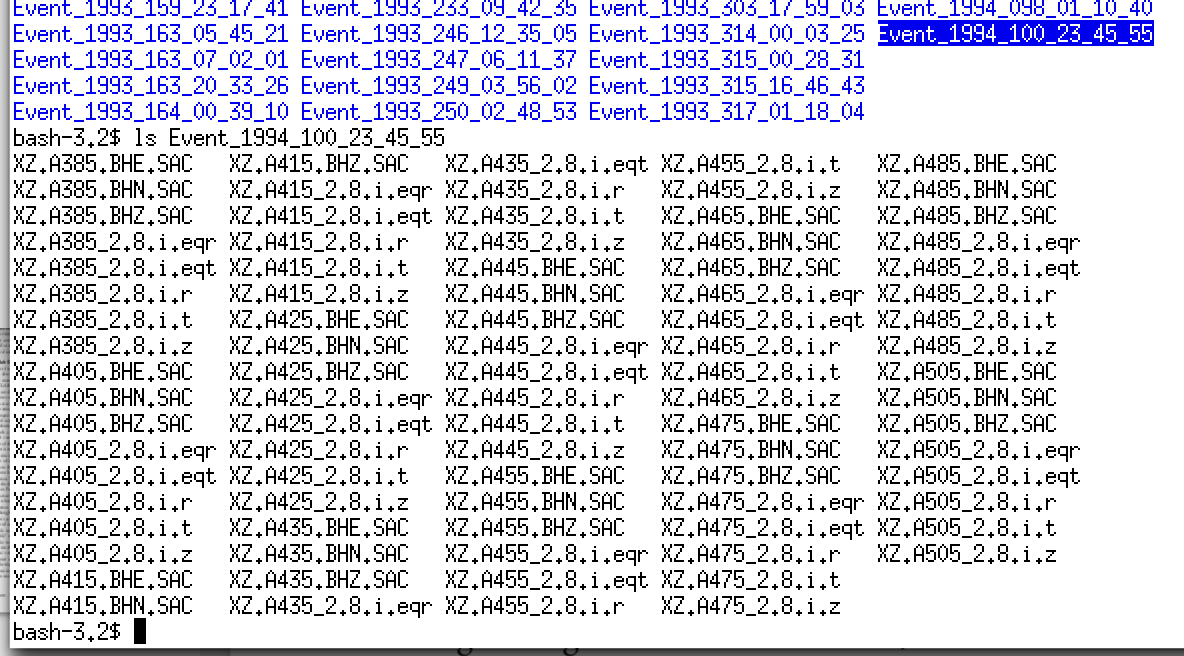
Funclab has a particular folder design for upload. Receiver functions for individual events need to be grouped and place in event folders. I recommend doing this alongside their components (radial and vertical channels) to avoid errors in the code, but all that is really needed is the radial receiver functions. You should try by making a new project and loading the folder in “RAWDATA” in the project example I provided (FUNC\_PROJ)

The format of the file structure should look like the following:



Data directory must have folders with the format “Event\_year\_jday\_hh\_mm\_ss” where jday is the julian day, and hh, mm, and ss are is the time that the event occurred.

Inside the event folders, SAC files should have names like the following:



Individual SAC files within Event folders should have format “NN.STNM\_GG.i.eqr” where NN is the network code, STNM is the station name, GG is the Gaussian alpha parameter used in receiver function calculation, and eqr(t) indicates either radial or tangential component receiver function.

**Open Funclab (in FN\_FUNCLAB/funclab1.5.5.dilatate)**

* Run “setup\_funclab.m” in MATLAB
* Run “funclab.m” in MATLAB
  + GUI should pop up outlining steps to load project or create new project
  + Warnings will pop up in MATLAB but can be ignored. These have to do with changes in the syntax of MATLAB in newer versions related to figure placement on the computer screen.
* **New Project Creation:**

1. Click “File” -> “New Project”
2. Need to name a folder for the Project (cannot already exist!)
3. Upload path to data directory (in example, CASC93\_ForFunc in RAWDATA folder)1
   1. A new folder containing the data will be copied into your project directory defined in (2). This is an EXACT copy of the data folder, so to save space, you can delete the data folder after it has been successfully added to your project.
4. Verify ray parameter and backazimuth are correct (necessary for CCP stacking analysis). **Ray parameter needs to be units of s/km**
5. FUNCLAB is a powerful tool. You can automatically or manually edit receiver functions by event or station by clicking “Edit” in the toolbar. A manual to Funclab comes with this package, and you can see 2 publications for some of the details of FUNCLAB in peer-reviewed form (Eager and Fouch, 2012; Porritt and Miller, 2017). This version of the code is well-represented by Eagar and Fouch (2012), but an updated version (with some bugs in it) can be downloaded from Rob Porritt’s personal webpage.

**Step 2: Calculation of adaptive CCP stacks**

**CCP Stacking analysis in Funclab**

In this version of the code (Funlab2.1), there is an option for “Dilating CCP Stacking \_\_\_” in the “Add-ons”. This approach is outlined briefly by Delph et al. 2015 (Tectonophysics) and 2017 (EPSL). This version differs from older funclab versions in that some of the Dilating CCP Stacking algorithms incorporate an elevation correction (elv + vel correction with bins defined in km or degs depending on the option you choose) prior to ray tracing and stacking. This is done by taking the highest elevation of the stations in the dataset, extending the model domain to that elevation, and filling in the velocity characteristics for the layers above sea level with those of the shallowest layer in the velocity model. **If your shallowest layer has a large Vp/Vs ratio, this may bias your results deeper in the model.** You can see the effect that the elevation correction can have in Herr & Delph (2023; JGR). You can still perform Dilating CCP stacking (with or without 3D velocity corrections) without an elevation correction (see supplemental material of Delph et al., 2021; EPSL for comparison sections). A general description of ACCP stacking is below:

1. Adaptive (or Dilating) CCP stacking is simple. A minimum number of hits in a bin, along with minimum and maximum bin width is set by the user. The CCP Stacking bin width starts at the minimum size, and if less than the user-defined minimum number of rays is obtained, the bin dilates by a predefined amount (set in the dBinR variable in addons/ccp\_dilate\_\*\_assign.m script). It does so until the maximum bin width is reached, after which it creates a stack based on the number of rays in a bin even if it is less than the minimum. \* If performing in preparation for a joint inversion, you should choose a bin spacing that is the same as your tomographic gridding in the surface wave inversions.
2. **Special Note:** In this version of the funclab, CCP stacks are computed for both normalized and un-normalized RFs. If you are interested in only the dilating aspect of the CCP stacking and not trying to make a dataset for a joint inversion, these will be computed and output in the final CCP volume in a different column than the un-normalized stacks. Regardless, high amplitudes (>0.8) in ~1 Hz receiver functions are difficult explain, likely the result of complex shallow structure or basin effects, and will not be fit well using the joint inversion of Julia et al., 2001. Be be sure to throw these RFs out.
3. **STEPS**
4. **1D Ray tracing:** All boxes will be grayed out except for Depth max and increment for resulting CCP model, and velocity model to migrate with. A tomography model is not necessary unless you would like to do a 3D correction (useful if CCP model is the end result, but cannot be used for receiver functions to be used in a joint inversion. For example of when useful, see Abgarmi et al. 2017, Geosphere). Run 1D ray tracing.
   1. **IMPORTANT**: Need to determine depth increment by ensuring the time spent in any bin << sampling rate of RF (See Delph et al 2015 for how to calculate this). Usually 0.5 km is sufficient around 1Hz).
5. **3D heterogeneity corrections** is an optional step. This is only necessary if you want to create 3D velocity corrected RFs following the approach of Eagar et al., 2010; EPSL). To do this, you can use to folder "tomo\_mod\_maker" to turn an x,y,z,vp,vs ASCII file into the proper format for the 3D velocity model correction. Matlab scripts xyzv2mat\*.m will require some editing to ensure your velocity model is properly created. An example is included for formatting reference.
6. **Time to depth migration:** All boxes remain grayed. Perform time-to-depth migration (not “technically” a migration, just a conversion…). Can skip 3D heterogeneity correction.
7. **CCP Assignment:** Boxes for latitudes, longitudes, minimum and max bin widths, bin spacing, and minimum hits per bin should now be editable. Bin width should always be >= bin spacing (if greater than, it acts as a lateral smoother). This code iterates to calculate the number of hits in a bin until a bin width is hit, so the larger the maximum bin width, the longer the computation time. Set to appropriate parameters and run CCP Assignment.
   1. **Note:** Bin width and spacing ARE IN DEGREES. Not appropriate for studies at high latitutes (need to slightly tweak ccp\_dilate\_stacking.m to work in kilometers)
   2. For the test dataset, run with depth range 0-100km (needs to be deeper than longest time for RF to use in joint inversion. Divide roughly by 8 to see how deep it needs to extend. I usually do 10 seconds, so 100 km is sufficient), Depth Inc 0.5 km, bin width 0.2 – 1 degree, lats 44 – 45, longs -125 - -120. Vel model was AK135. Hits per bin were set at 10.
8. **CCP Stacking:** Bootstrap Resamples and Bootstrap percentage should now be editable. **BE SURE BOOTSTRAP PERCENTAGE = 100** (**I think this is a bug in funclab)**. This number acts as a gain on RF amplitudes. This number is a percentage that is multiplied with the RF amplitudes, so 100 =100%, and will preserve true amplitudes. Run CCP Stacking. Bootstrap resampling is a common way to estimate errors (or at least the variation) within your dataset. The higher the number, the longer the calculations will take. Keep Bootstrap Resamples at 200 for example.
9. **Tricky step (need to run outside Funclab)**
   1. In the addon/ccp\_dilate folder, there is a script called **Getavgrp.m**. Copy this to the CCPDATA\_dilate folder and run. This code will output the average ray parameter in the shallowest bin for all gridpoints that have a results at ALL depths (i.e., profiles that do not have a result at 0 will be ignored).
10. Click on Export -> Export CCP Stacks in CCP Stacking Toolbar
    1. Compare output in this file to the outputs in the processed folder (ccp\_stacks\_jrd.txt)
       1. Outputs in texfile are:

long, lat, “0”, depth, Unnormalized CCP stacking amplitude, …

Unnorm. Bootstrap error, normalized CCP stacking amplitude, …

Norm. Bootstrap error, bin width, hits, …

average ray parameter in shallowest bin