In [24]:
```
#Make an array that contains the list of points that match up with the distance along
#the trench of the present day day

#Make an array of lonlats that we want to search in
lonlat = numpy.r_['1,2,0',andesPresent2[:,0,0],andesPresent2[:,0,1]]

#The size of the region to find the closest point (in degrees)
region=10


########################################
##Make this append to the andesClean data
#Make an emtpy arry that stores the indexes of
indexArray=numpy.zeros((len(andesClean[:,0,0]),len(andesClean[0,:,0])))

d = numpy.zeros((andesClean.shape[0],andesClean.shape[1],andesClean.shape[2]+1))
d[:,:,:-1] = andesClean

#Loop through all the data in space and time
for ind,val in enumerate(andesClean[:,0,0]):
    for jind,valj in enumerate(andesClean[ind,:,0]):
        lonBirth=andesClean[ind,jind,0]
        latBirth=andesClean[ind,jind,1]

        #Now look up the nearest neighbor for each point
        index=coregPoint([lonBirth,latBirth],lonlat[:],region)

        if index=='inf':
            pass
        else:
            #and save the index and distance along the track
            indexArray[ind,jind]=index
            d[ind,jind,-1]=distAlongTrack[index]

#Add the subduction obliquity angle data and the ore deposit formation flag data
thetad=numpy.arctan(d[:,:,15]/d[:,:,14])
d2 = numpy.ones((d.shape[0],d.shape[1],d.shape[2]+2))
d2[:,:,:-2] = d
d2[:,:,-2]=thetad


##### Random data

########################################
##Make this append to the andesRandClean data
#Make an emtpy arry that stores the indexes of
indexArray=numpy.zeros((len(andesRandClean[:,0,0]),len(andesRandClean[0,:,0])))

d = numpy.zeros((andesRandClean.shape[0],andesRandClean.shape[1],andesRandClean.shape[2]+1))
d[:,:,:-1] = andesRandClean

# print andesPresent2[:,0,7]
# print distAlongTrack
#Loop through all the data in space and time
for ind,val in enumerate(andesRandClean[:,0,0]):
    for jind,valj in enumerate(andesRandClean[ind,:,0]):
        lonBirth=andesRandClean[ind,jind,0]
        latBirth=andesRandClean[ind,jind,1]

        #Now look up the nearest neighbor for each point
        index=coregPoint([lonBirth,latBirth],lonlat[:],region)

        if index=='inf':
            pass
        else:
            #and save the index and distance along the track
            indexArray[ind,jind]=index
#             print ind,jind,index
            d[ind,jind,-1]=distAlongTrack[index]

#Add the subduction obliquity angle data and the ore deposit formation flag data
thetad=numpy.arctan(d[:,:,15]/d[:,:,14])
d2Rand = numpy.zeros((d.shape[0],d.shape[1],d.shape[2]+2))
d2Rand[:,:,:-2] = d
d2Rand[:,:,-2]=thetad
```

```
##########################################
#Combine to Random and Ore Deposit datasets
d3=numpy.concatenate([d2,d2Rand])

print andesClean.shape, d2.shape
print andesRandClean.shape, d2Rand.shape

print numpy.shape(d3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-4f9a68a84866> in <module>()
     24
     25         #Now look up the nearest neighbor for each point
---> 26         index=coregPoint([lonBirth,latBirth],lonlat[:],region)
     27
     28         if index=='inf':

/Users/amirfoxus/mineralexplorationcourse/Week10/Utils_coreg.pyc in coregPoint(point, data, region)
    120     region - integer, same units as data
    121     '''
--> 122     tree = scipy.spatial.cKDTree(data)
    123     dists, indexes = tree.query(point,k=1,distance_upper_bound=region)
    124

ckdtree.pyx in scipy.spatial.ckdtree.cKDTree.__init__()

/Users/amirfoxus/miniconda3/envs/py2GEOL/lib/python2.7/site-packages/numpy/core/fromnumeric.pyc in amax(a,
 axis, out, keepdims, initial)
   2332     """
   2333     return _wrapreduction(a, np.maximum, 'max', axis, None, out, keepdims=keepdims,
-> 2334                           initial=initial)
   2335
   2336

/Users/amirfoxus/miniconda3/envs/py2GEOL/lib/python2.7/site-packages/numpy/core/fromnumeric.pyc in _wrapred
uction(obj, ufunc, method, axis, dtype, out, **kwargs)
     81                 return reduction(axis=axis, out=out, **passkwargs)
     82
---> 83     return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
     84
     85

ValueError: zero-size array to reduction operation maximum which has no identity
```