**Details**

For 300

1. Code for 48 x 354 input matrix: to run dynGENIE3 with improved conditions and Get\_link\_list [Code\_1\_links\_for\_300](https://drive.google.com/file/d/1EaX_A_LwF9Zy_TVHsgCsG3ziIzRrYcwm/view?usp=sharing)

For 17,000

1. Code to import data and find genes that throw out error [CheckforErrorGenes](https://drive.google.com/file/d/1dI6wL759QmDz1XinT_3aVMmUFUsmpfG9/view?usp=sharing)
2. Code to filter them out and to run dynGENIE3 & Get\_link\_list with names with no max count [ToRunAFinalTime\_edited](https://drive.google.com/file/d/1JjDchdHCzTwrdvuBjM5dOJJnkaP18hdg/view?usp=sharing)
3. Code for filtering Target by CCM genes and Regulator by CCM and TF genes [FilteringInterestedGenes](https://drive.google.com/file/d/1VF7Pdb1QOx-s7xnR-z-MfcJt9iZjWOim/view?usp=sharing)

For source file (they all similar, except the part in Get\_link\_list function, shown in the table below)

1. dynGENIE3.py: original dynGENIE3 file from Github
2. dynGENIE3\_mod.py: same with the original file, just adding comments and some code that enable tracking the process (green highlighted)
3. dynGENIE3\_mod\_for.py: change ndenumerate to nested for loop (yellow highlighted)

**Note: why many source file?** for 48 x 354 input matrix, any source files can be used ( though dynGENIE3.py is not recommended), but for the larger 48 x 17,777 input matrix, the dynGENIE3\_mod\_for.py is recommended as it is faster

| dynGENIE3.py | dynGENIE3\_mod.py | dynGENIE3\_mod\_for.py |
| --- | --- | --- |
| # Get the non-ranked list of regulatory links  vInter = [(i,j,score) for (i,j),score in ndenumerate(VIM) if i in input\_idx and i!=j]    # Rank the list according to the weights of the edges  vInter\_sort = sorted(vInter,key=itemgetter(2),reverse=True) #sort vInter by score  nInter = len(vInter\_sort)    # Random permutation of edges with score equal to 0  flag = 1  i = 0  while flag and i < nInter:  (TF\_idx,target\_idx,score) = vInter\_sort[i]  if score == 0:  flag = 0  else:  i += 1    if not flag:  items\_perm = vInter\_sort[i:]  items\_perm = random.permutation(items\_perm)  vInter\_sort[i:] = items\_perm    # Write the ranked list of edges  nToWrite = nInter  if isinstance(maxcount,int) and maxcount >= 0 and maxcount < nInter:  nToWrite = maxcount | # Get the non-ranked list of regulatory links  vInter = list()  for (i,j),score in ndenumerate(VIM) :  if i in input\_idx and i!=j:  vInter.append((i,j,score))  if i%100 == 0:  print(vInter[-1])    #if regulator = 'all' then input\_idx will be 0 to ngenes-1 -> so i will be each of thme in turn  # (0,0), 0.4353463 -> (0,0, 0.4353463)  # index, x in np.ndenumerate(a) produces (0, 0) 1  # may be I could add the progress bar here or just print(vInter[-1]))      # Rank the list according to the weights of the edges  vInter\_sort = sorted(vInter,key=itemgetter(2),reverse=True) #sort vInter by score  nInter = len(vInter\_sort)    # Random permutation of edges with score equal to 0  flag = 1  i = 0  while flag and i < nInter:  (TF\_idx,target\_idx,score) = vInter\_sort[i]  if score == 0:  flag = 0  else:  i += 1    if not flag:  items\_perm = vInter\_sort[i:]  items\_perm = random.permutation(items\_perm)  vInter\_sort[i:] = items\_perm    # Write the ranked list of edges  nToWrite = nInter  if isinstance(maxcount,int) and maxcount >= 0 and maxcount < nInter:  nToWrite = maxcount  print(nToWrite)  else:  print("No Maxcount, will write", nToWrite) | # Get the non-ranked list of regulatory links  vInter = list()  for i in range(VIM.shape[0]) :  for j in range(VIM.shape[1]):  if i in input\_idx and i!=j:  score = VIM[i,j]  vInter.append((i,j,score))  if i%100 == 0:  print(vInter[-1])  if i == 17777:  print("finish")    #if regulator = 'all' then input\_idx will be 0 to ngenes-1 -> so i will be each of thme in turn  # (0,0), 0.4353463 -> (0,0, 0.4353463)  # index, x in np.ndenumerate(a) produces (0, 0) 1  # may be I could add the progress bar here or just print(vInter[-1]))      # Rank the list according to the weights of the edges  vInter\_sort = sorted(vInter,key=itemgetter(2),reverse=True) #sort vInter by score  nInter = len(vInter\_sort)    # Random permutation of edges with score equal to 0  flag = 1  i = 0  while flag and i < nInter:  (TF\_idx,target\_idx,score) = vInter\_sort[i]  if score == 0:  flag = 0  else:  i += 1    if not flag:  items\_perm = vInter\_sort[i:]  items\_perm = random.permutation(items\_perm)  vInter\_sort[i:] = items\_perm    # Write the ranked list of edges  nToWrite = nInter  if isinstance(maxcount,int) and maxcount >= 0 and maxcount < nInter:  nToWrite = maxcount  print(nToWrite)  else:  print("No Maxcount, will write", nToWrite) |
| Cannot track | Can track | Can track &  Use *for* instead of *ndenumerate* (which is slower) |