transition-count-matrices

June 25, 2015

1 Transition count matrices

A transition count matrix is a matrix that contains all the vocabulary (rows) and all of the positions (columns) and each of the elements is simply a count of the number of times that a transition occurs. This matrix is can be created from an alignment matrix or it may be found by counting all of the transitions in a given phylogenetic tree.

1.1 Data munging

We assume that trees are provided in the newick-tree format. Conveniently, in Python there is a package for handling newick trees.

- ete2

Basically we just use ete2 to store the tree structure for iterating and visualization. The newick trees produced by phylobayes are not directly following any standard type of newick tree that ete2 understands so if we load directly there is a loss of information. Specifically, the transitions at internal nodes will be ignored.

To *fix* this there is a function called **fix_tree** that reads the tree in to get the basic structure then it iterates over the newick formatted string to both confirm the structure and repopulate the missing informtion.

Before we can fix the tree we must standardize it first. For example we need to make the following types of substitutions

```
_Q:0.0367044:Q) --> _Q:0.0367044)
_Q:0.044464:Q:0.0512411:K) --> _Q:0.044464)
```

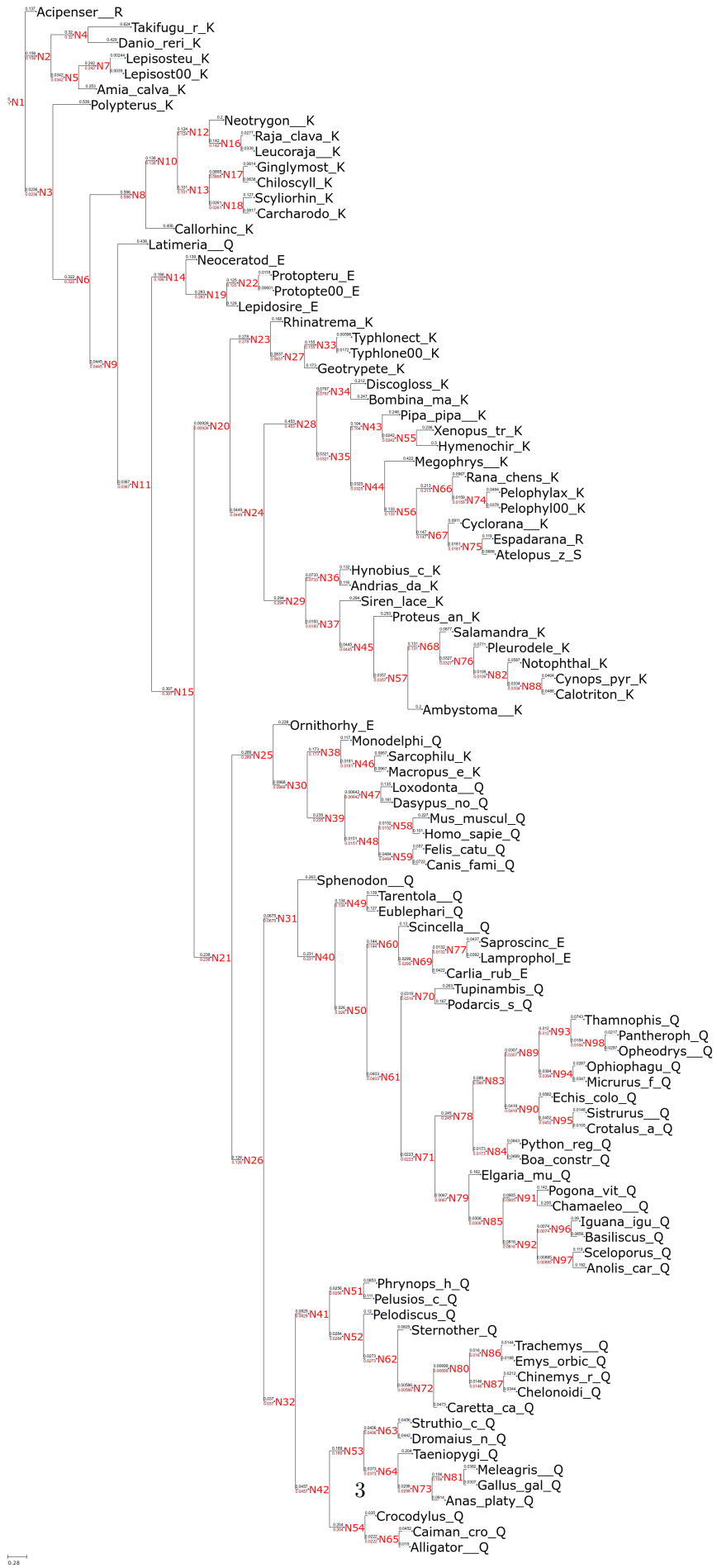
```
In [50]: import os
         from ete2 import Tree,TreeStyle
         from ete2 import faces, AttrFace
         from vertebratesLib import *

         def standardize_tree(nwTree):
             pattern1 = "[\w|:]+.+?[,|\)|\(]"
             parsedTree = nwTree

             for r in re.finditer(pattern1,nwTree):
                 matched = nwTree[r.start(0):r.end(0)]
                 parsed = re.sub(":[A-Z]","",matched)
                 multiples = re.findall(":\d\.[\de-]+",parsed)
                 trans = "".join(multiples)

                 if len(multiples) > 1:
                     parsed = parsed.replace(trans,multiples[0])
                 parsedTree = parsedTree.replace(matched,parsed)
```

```
        return re.sub("\)_[A-Z]:",")",parsedTree)[:-3]+";"

    ## standardize a phylobayes tree
    pbtree = "(Acipenser__R:0.137306:R:0.167816:K,((Takifugu_r_K:0.623751:K,Danio_reri_K:0.428613:
    nwtree = standardize_tree(pbtree)
    #print(pbtree)
    #print(nwree)

    ## fix phylobayes tree (standardize, add internal nodes and extract extra info)
    fixedTree,treeSummary = fix_tree(pbtree)
    for key in ['aa','pairs','transitions','dist','parent']:
        print("%s - %s"%(key, treeSummary["Danio_reri"][key]))

    print("...parent N4- %s"%treeSummary["N4"]['pairs'])

    ## plot the tree
    def my_layout(node):
        if node.is_leaf():
            name_face = AttrFace("name",fsize=25)
            faces.add_face_to_node(name_face, node, column=0, position="branch-right")
        else:
            name_face = AttrFace("name", fsize=20, fgcolor="red")
            faces.add_face_to_node(name_face, node, column=0, position="branch-right")

    ts = TreeStyle()
    ts.show_leaf_name = False
    ts.show_branch_length = True
    ts.show_branch_support = True
    ts.scale =  180
    ts.layout_fn = my_layout
    out = fixedTree.render("./figures/simple-tree.png", units="mm", tree_style=ts,dpi=400)
```

```
aa - K
pairs - ['KK']
transitions - ['0.428613:K']
dist - 0.428613
parent - N4
...parent N4- ['KK']
```

```
In [51]: from IPython.display import Image
         Image(filename='./figures/simple-tree.png')
```

Out[51]:

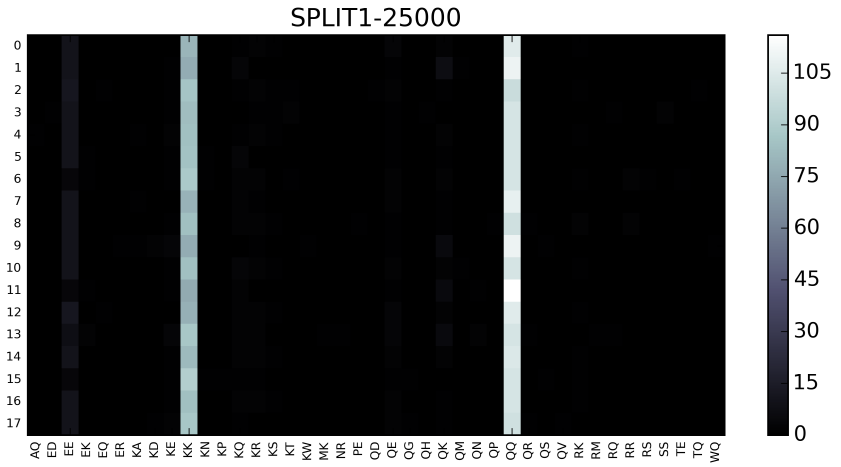

Acipenser__R
Takifugu_r_K
Danio_reri_K
Lepisosteu_K
Lepisost00_K
Amia_calva_K
Polypterus_K
N1
N2
N3
N4
N5
N6
N7
N8
Neotrygon__K
Raja_clava_K
Leucoraja__K
Ginglymost_K
Chiloscyll_K
Scyliorhin_K
Carcharodo_K
N10
N12
N13
N16
N17
N18
Callorhinc_K
Latimeria__Q
Neoceratod_E
Protopteru_E
Protopte00_E
Lepidosire_E
N9
N11
N14
N19
N22
Rhinatrema_K
Typhlonect_K
Typhlone00_K
Geotrypete_K
N23
N27
N33
Discogloss_K
Bombina_ma_K
Pipa_pipa__K
Xenopus_tr_K
Hymenochir_K
Megophrys__K
Rana_chens_K
Pelophylax_K
Pelophyl00_K
Cyclorana__K
Espadarana_R
Atelopus_z_S
N20
N24
N28
N34
N35
N43
N44
N55
N56
N66
N67
N74
N75
Hynobius_c_K
Andrias_da_K
Siren_lace_K
Proteus_an_K
Salamandra_K
Pleurodele_K
Notophthal_K
Cynops_pyr_K
Calotriton_K
Ambystoma__K
N15
N29
N36
N37
N45
N57
N68
N76
N82
N88
Ornithorhy_E
Monodelphi_Q
Sarcophilu_K
Macropus_e_K
Loxodonta__Q
Dasypus_no_Q
Mus_muscul_Q
Homo_sapie_Q
Felis_catu_Q
Canis_fami_Q
N21
N25
N30
N38
N39
N46
N47
N48
N58
N59
Sphenodon__Q
Tarentola__Q
Eublephari_Q
Scincella__Q
Saproscinc_E
Lamprophol_E
Carlia_rub_E
Tupinambis_Q
Podarcis_s_Q
N26
N31
N40
N49
N50
N60
N69
N70
N77
Thamnophis_Q
Pantheroph_Q
Opheodrys__Q
Ophiophagu_Q
Micrurus_f_Q
Echis_colo_Q
Sistrurus__Q
Crotalus_a_Q
Python_reg_Q
Boa_constr_Q
Elgaria_mu_Q
Pogona_vit_Q
Chamaeleo__Q
Iguana_igu_Q
Basiliscus_Q
Sceloporus_Q
Anolis_car_Q
N61
N71
N78
N79
N83
N84
N85
N89
N90
N91
N92
N93
N94
N95
N96
N97
N98
Phrynops_h_Q
Pelusios_c_Q
Pelodiscus_Q
Sternother_Q
Trachemys__Q
Emys_orbic_Q
Chinemys_r_Q
Chelonoidi_Q
Caretta_ca_Q
N32
N41
N51
N52
N62
N72
N80
N86
N87
Struthio_c_Q
Dromaius_n_Q
Taeniopygi_Q
Meleagris__Q
Gallus_gal_Q
Anas_platy_Q
Crocodylus_Q
Caiman_cro_Q
Alligator__Q
N42
N53
N54
N63
N64
N65
N73
N81
3
0.28

1.2 Distributions of transitions at the sample level

There are 18 samples at each position.

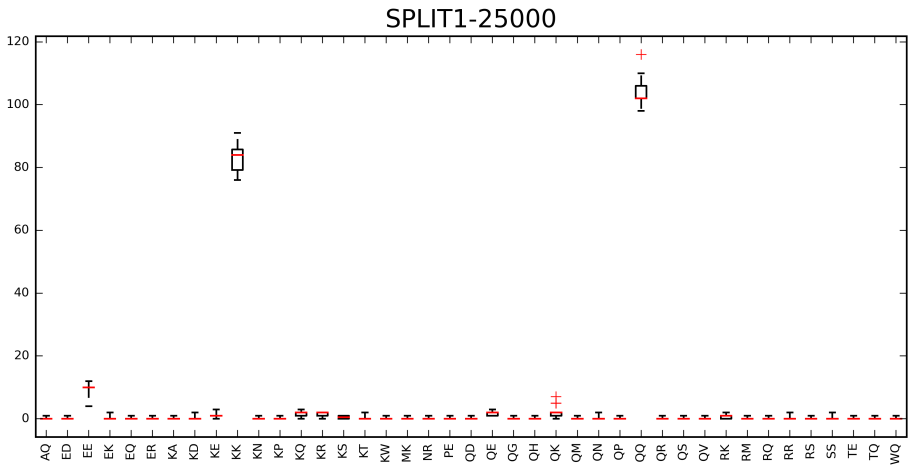
In [52]:
```
dataDir = None
for ddir in [os.path.join("..","data","herve-vertebrates"),\
             os.path.join("/","media","ganda","mojo","phylogenetic-models","herve-vertebrates")]:
    if os.path.isdir(ddir):
        dataDir = ddir

split = "SPLIT1"
position = "25000"
treeList = get_trees(split,position,dataDir)
countMatrix = np.zeros((len(treeList),len(TRANSITIONS)),)
t = 0
for t,pbTree in enumerate(treeList):
    fixedTree,treeSummary = fix_tree(pbTree)
    tlist = []
    for item in treeSummary.itervalues():
        tlist.extend(item['pairs'])
    counts = transitions_to_counts(tlist)
    countMatrix[t,:] = counts

## make example plots
figTitle = "%s-%s"%(split,position)
figName1 = os.path.join("figures","example-profile-heatmap-1.png")
profile_heatmap_plot(countMatrix,figName1,figTitle=figTitle,cmap=plt.cm.bone)
Image(filename=figName1)
```

Out[52]:



In [53]:
```
figName2 = os.path.join("figures","example-profile-bplot-1.png")
profile_box_plot(countMatrix,figName2,figTitle=figTitle)
Image(filename=figName2)
```

Out[53]:



In [ ]:

1.3 Data munging...

- runDataMungeVertebrates.py
- dataMungeVertebrates.py

Because the script is fairly complicated here we just validate that we get the same transition distribution as above.

In [54]:
```
## load data from data munge
outputDir = os.path.join("..","data","hv-compressed")
outputFile = os.path.join(outputDir,"%s.npz"%(split))
npz = np.load(outputFile)
summaryTree = npz['tree']
summarySpecies = {}
for key,item in npz.iteritems():
    if key == 'tree':
        continue
    summarySpecies[key] = item

splitPositions = get_positions(split,dataDir)
splitIndex = np.where(splitPositions==position)[0]
nonZero = np.where(summaryTree[splitIndex,:] != 0)[1]
print nonZero
for nz in nonZero:
    print("%s - %s"%(TRANSITIONS[nz],summaryTree[splitIndex,nz]))
```

```
[ 63 163 168 173 174 175 263 268 273 288]
EE - [ 9.]
KE - [ 1.]
KK - [ 83.]
KQ - [ 2.]
KR - [ 1.]
```

```
KS - [ 1.]
QE - [ 2.]
QK - [ 2.]
QQ - [ 104.]
RK - [ 1.]
```

There is a very larege reduction in data and this is what is being saved.

```
In [55]: print("Full Tree - %s x %s "%summaryTree.shape)
         for key,item in summarySpecies.iteritems():
             print("%s - %s x %s"%(key,item.shape[0],item.shape[1]))
```

```
Full Tree - 30000 x 400
Ginglymost - 30000 x 400
Sceloporus - 30000 x 400
Rhinatrema - 30000 x 400
Saproscinc - 30000 x 400
Echis_colo - 30000 x 400
Rana_chens - 30000 x 400
```

```
In [56]: print("approx total number of matrices saved: %s"%(30000 * 10 * len(npz.keys())))
```

```
approx total number of matrices saved: 2100000
```

We go from about 2.2 GB per split to 2.2 MB per file.

```
In [ ]:
```