EcoAssocNet inference - python tutorial

January 17, 2020

Contents

1 Tutorial on using EcoAssocNet

This tutorials shows how to use the package on an example dataset to learn associations.

The following packages are required: - Data manipulation: Pandas, numpy - Plotting: matplotlib, seaborn - Machine learning: scikit-learn, tensorflow 1.5, keras

1.1 Part one: preparing the data

We offer a helper class DataPrep to automate data preprocessing, particularly that of environmental features.

[38]:
```
import warnings
warnings.filterwarnings("ignore")
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
tf.get_logger().setLevel('INFO')
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(color_codes=True)
```

1.1.1 Loading dataset

The data used here is provided as part of the examples folder. The data was obtained from ade4 (R package), it was produced as part of a paper from Choler et al 2005, provided within the examples/doc folder.

```
[3]: folder_data="../examples/Aravo/data/"
     file_env=folder_data+"env.csv"
     file_count=folder_data+"occur.csv"
```

```
[4]: env=pd.read_csv(file_env,sep=";",decimal=".")
     env.head()
```

```
[4]:    Aspect  Slope  Form  PhysD ZoogD  Snow
     0       7      2     1     50    no   140
     1       1     35     3     40    no   140
     2       5      0     3     20    no   140
     3       9     30     3     80    no   140
     4       9      5     1     80    no   140
```

```
[5]: counts=pd.read_csv(file_count,sep=";",decimal=".")
     names=counts.columns.tolist()
     occur=(counts>0).astype(int)
     counts.head()
```

```
[5]:    Agro.rupe  Alop.alpi  Anth.nipp  ...  Trif.alpi  Trif.badi  Trif.thal
     0          0          0          0  ...          0          0          0
     1          0          0          0  ...          0          0          0
     2          3          0          1  ...          0          0          0
     3          0          0          0  ...          0          0          0
     4          0          0          0  ...          0          0          0

     [5 rows x 82 columns]
```

1.1.2 Preprocessing environmental data

```
[8]: import sys
     sys.path.append('../../')
```

```
[9]: from ecoassocnet.Util.DataPrep import DataPrep
```

```
Using TensorFlow backend.
```

```
[10]: num_vars=['Slope','PhysD','Snow']
      cat_vars=['Aspect','Form','ZoogD']
      prep=DataPrep(num_std=["minmax"]*len(num_vars),cat_trt="onehot")
```

```
[11]: prep.load_dataset(feat=env,occur=occur,num=num_vars,cat=cat_vars)
```

```
[12]: prep.preprocess_numeric()
      prep.process_categoric()
      prep.combine_covariates()
```

```
[13]: prep.covariates.head()
```

```
[13]:       Slope  PhysD  Snow  Aspect_0  ...  Form_4  ZoogD_0  ZoogD_1  ZoogD_2
      0  0.057143  0.625   0.0       0.0  ...     0.0      0.0      1.0      0.0
      1  1.000000  0.500   0.0       1.0  ...     0.0      0.0      1.0      0.0
      2  0.000000  0.250   0.0       0.0  ...     0.0      0.0      1.0      0.0
      3  0.857143  1.000   0.0       0.0  ...     0.0      0.0      1.0      0.0
      4  0.142857  1.000   0.0       0.0  ...     0.0      0.0      1.0      0.0

      [5 rows x 19 columns]
```

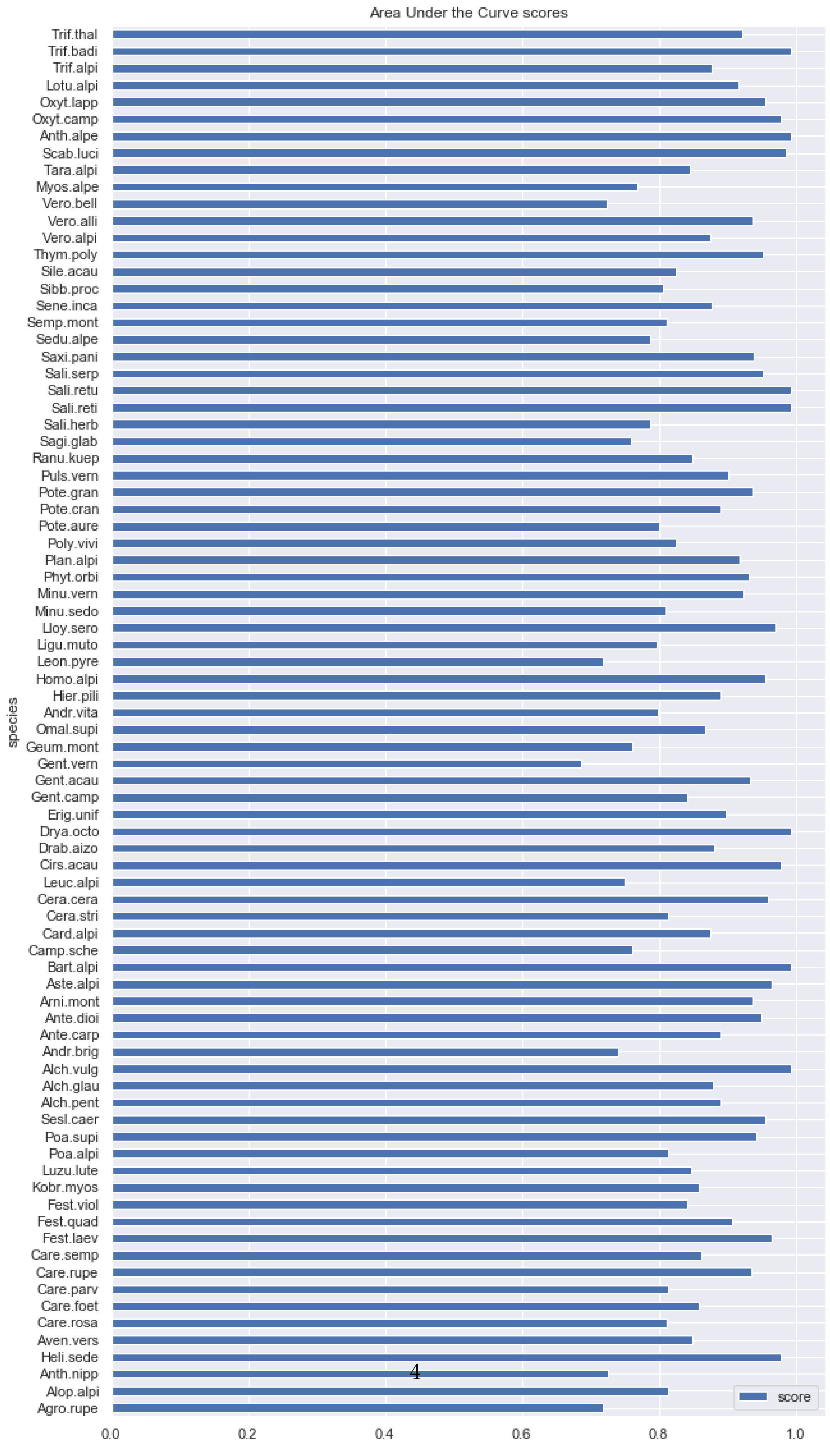
1.2 Part Two: Training the model

1.2.1 Habitat Suitability Model pretraining (Optional)

```
[14]: perfs,params=prep.pretrain_glms()
```

```
[15]: fig, ax=plt.subplots(1,1,figsize=(10,20))
      perfs[0].plot.barh(x='species',y='score',ax=ax,title='Area Under the Curve␣
       ↪scores')
```

```
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1bba4de59c8>
```

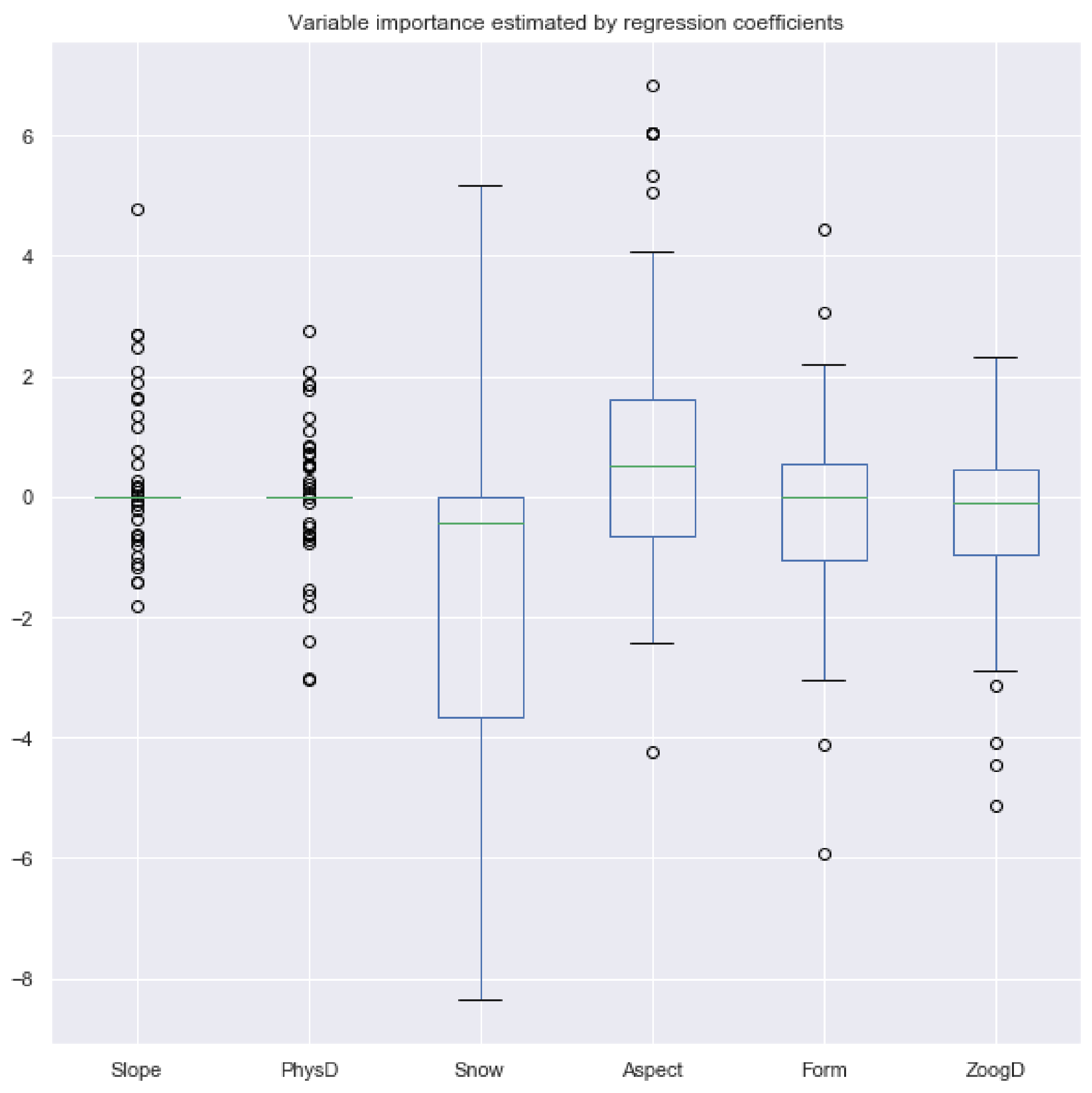

Area Under the Curve scores
species
score
Trif.thal
Trif.badi
Trif.alpi
Lotu.alpi
Oxyt.lapp
Oxyt.camp
Anth.alpe
Scab.luci
Tara.alpi
Myos.alpe
Vero.bell
Vero.alli
Vero.alpi
Thym.poly
Sile.acau
Sibb.proc
Sene.inca
Semp.mont
Sedu.alpe
Saxi.pani
Sali.serp
Sali.retu
Sali.reti
Sali.herb
Sagi.glab
Ranu.kuep
Puls.vern
Pote.gran
Pote.cran
Pote.aure
Poly.vivi
Plan.alpi
Phyt.orbi
Minu.vern
Minu.sedo
Lloy.sero
Ligu.muto
Leon.pyre
Homo.alpi
Hier.pili
Andr.vita
Omal.supi
Geum.mont
Gent.vern
Gent.acau
Gent.camp
Erig.unif
Drya.octo
Drab.aizo
Cirs.acau
Leuc.alpi
Cera.cera
Cera.stri
Card.alpi
Camp.sche
Bart.alpi
Aste.alpi
Arni.mont
Ante.dioi
Ante.carp
Andr.brig
Alch.vulg
Alch.glau
Alch.pent
Sesl.caer
Poa.supi
Poa.alpi
Luzu.lute
Kobr.myos
Fest.viol
Fest.quad
Fest.laev
Care.semp
Care.rupe
Care.parv
Care.foet
Care.rosa
Aven.vers
Heli.sede
Anth.nipp
Alop.alpi
Agro.rupe
0.0
0.2
0.4
0.6
0.8
1.0

```
[16]: biases=np.array(params[0]['b'])
      weights=np.concatenate([biases,params[0]['w']],axis=1)
      weights_df=pd.DataFrame(data=weights,columns=['bias']+prep.covariates.columns.
       ↪tolist())
```

```
[17]: for c in prep.groups.keys():
          if len(prep.groups[c])>1:
              weights_df[c]=weights_df[prep.groups[c]].sum(axis=1)
```

```
[18]: fig, ax=plt.subplots(1,1,figsize=(10,10))
      weights_df[num_vars+cat_vars].plot.box(ax=ax,title='Variable importance␣
       ↪estimated by regression coefficients')
```

```
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1bba66b5c08>
```



1.2.2 Training, validation data

[19]:
```
prep.train_test_split(meth="stratified",prob=0.8)
X_train=prep.covariates.iloc[prep.idx_train,:].values
X_test=prep.covariates.iloc[prep.idx_test,:].values

Y_train=counts.iloc[prep.idx_train,:].values
Y_test=counts.iloc[prep.idx_test,:].values
```

1.2.3 Setting up configuration files

```
[21]: from ecoassocnet.EcoAssoc import EcoAssoc, load_default_config
```

```
[22]: from ecoassocnet.Util.Util import avgnz
```

Computing the offset to be used

```
[23]: offsets=avgnz(counts)
```

```
[28]: training_config_file='../examples/Aravo/config/association_learning.ini'
```

To understand the use of each of the following parameters, refer to the documented default config file.

```
[30]: conf=load_default_config(training_config_file)
      for k in conf.keys():
          print("%s = %s" %(k,conf[k]))
```

```
exposure = True
use_covariates = True
intercept = False
fixedoccur = False
w_sigma2 = 1.0
archi_desc_file = examples/Aravo/config/hsm_archi.json
archi_plot_file = examples/Aravo/archi.png
plot = False
bias = True
offset = False
dist = negbin
assoc_plasticity = False
k = 4
use_reg = True
ar_sigma2 = 1.0
prior = lasso
lambda_lasso = 0.1
use_penalty = False
emb_initializer = uniform
fixed_rho = False
optim = sgd
sample_ratio = 0.2
use_valid = True
lr = 0.01
use_decay = False
lr_update_step = 10000
lr_update_scale = 0.5
batch_size = 1
max_iter = 5000
```

```
nprint = 1000
```

[35]:
```
ecoasso_model=EcoAssoc(config=training_config_file,labels=names,name_dataset="aravo",target="co
```

Hereafter, we launch the training for a few epochs (5)

[40]:
```
logg= ecoasso_model.
 ↪train_interaction_model(dataset=(X_train,Y_train),verbose=1,init_weights=weights,offset=offse
```

```
Splitting train and validation sets
Computation graph creation (static)
Computation graph initialization
Setting pretrained HSM weights
Begin training
iteration[ 1000 ]: average llh, obj, and valid_llh are  [-35.1035095
35.92741251 -37.04162254]
iteration[ 2000 ]: average llh, obj, and valid_llh are  [-32.04687284
32.82447647 -35.12422829]
iteration[ 3000 ]: average llh, obj, and valid_llh are  [-31.40496827
32.15929597 -34.58151703]
iteration[ 4000 ]: average llh, obj, and valid_llh are  [-30.35970771
31.07175282 -33.64297371]
iteration[ 5000 ]: average llh, obj, and valid_llh are  [-29.98409594
30.65965316 -33.92271309]
```

1.2.4 Evaluation

Here, we show how to evaluate a trained model given a test set

[41]:
```
perf_hsm, perf_im=ecoasso_model.evaluate_model(testdata=(X_test,Y_test))
```

```
Building graph...
Initializing...
Calculating llh of instances...
```

[42]:
```
print("Performance of HSM component" , perf_hsm)
```

```
Performance of HSM component {'microauc': 0.903777246727267}
```

[44]:
```
print("Performance of abundance component" , perf_im['pos_deviance'])
```

```
Performance of abundance component 0.42921730266340535
```

1.2.5 Usage for conditional predictions of abundance

[45]:
```
t=0
C=[x for x in range(len(names)) if x!=t]
Yc=Y_test[:,C]
```

```
habsuit,avg_abund=ecoasso_model.predict(X_test,C,Yc,t)
print('Predicting abundance of %s' % names[t])
(habsuit>0.5)*avg_abund
```

```
Predicting abundance of Agro.rupe
```

```
[45]: array([1.03985339, 1.03923207, 1.09525126, 1.05039894, 1.06305322,
             1.05340252, 1.06702837, 1.08231762, 1.05621757, 1.07785998,
             1.08209483, 1.05161302, 1.04389093, 0.        , 0.        ,
             1.00571548, 1.01322025, 0.99395305])
```

1.2.6 Unraveling associations

```
[46]: from ecoassocnet.Util.Util import plot_association, plot_assoc_clustered
```

```
[48]: assoc_df=ecoasso_model.compute_associations(save=False,norm=True)
```

We ignore intraspecific associations estimated.

```
[49]: assoc_df*=(np.identity(len(names))==0)
```

1.2.7 Applying biogeographic filtering

```
[55]: from ecoassocnet.Util.Util import cooccur, response_sim, biogeo_filter,␣
       ↪plot_dendrograms
```

```
[56]: cooc=cooccur(occur,names)
      respsim=response_sim(weights)
      sel_assoc=biogeo_filter(assoc_df,cooc,respsim,thoccur=0,thass=0.5,thresp=0.
       ↪5,m=len(names))
```

```
[57]: sel_assoc
```

```
[57]:            Agro.rupe  Alop.alpi  Anth.nipp  ...  Trif.alpi  Trif.badi  Trif.thal
      Agro.rupe  -0.000000   0.000000   0.566786  ...   0.000000  -0.000000  -0.000000
      Alop.alpi  -0.771278   0.000000   0.000000  ...   0.000000  -0.000000   0.000000
      Anth.nipp   0.000000   0.000000   0.000000  ...  -0.000000   0.828475  -0.000000
      Heli.sede  -0.000000  -0.000000  -0.000000  ...   0.000000   0.000000  -0.888695
      Aven.vers   0.571304  -0.000000   0.000000  ...  -0.000000   0.599935  -0.000000
      ...              ...        ...        ...  ...        ...        ...        ...
      Oxyt.lapp  -0.000000  -0.000000  -0.000000  ...   0.000000  -0.000000   0.000000
      Lotu.alpi   0.000000   0.000000   0.000000  ...  -0.000000   0.606419   0.596023
      Trif.alpi  -0.000000   0.965686   0.000000  ...   0.000000   0.000000   0.000000
      Trif.badi  -0.000000   0.000000   0.544881  ...  -0.000000   0.000000  -0.000000
      Trif.thal   0.000000   0.000000   0.804835  ...  -0.594169   0.859545  -0.000000
```

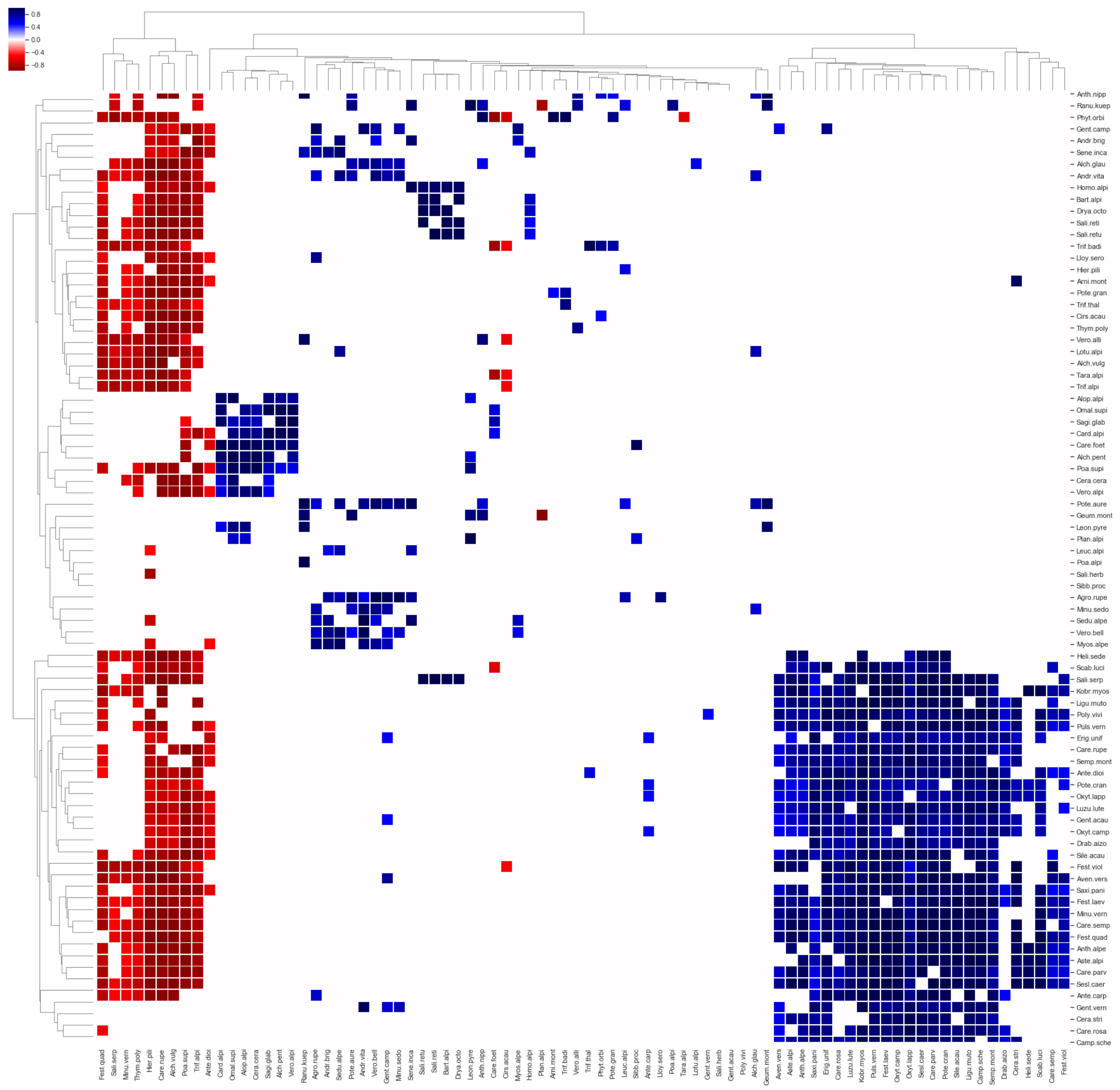
```
[82 rows x 82 columns]
```

2 Part Three: Association analysis

Hereafter, we show how to analyze the learnt association matrix. We illustrate on the final association matrix (obtained after full training). Particularly, we analyze the similarities in terms of associations by performing a (hierarchical) co-clustering of the association matrix.

[60]:
```
assoc_df=pd.read_csv('../examples/Aravo/results/plant_associations.csv',sep=";
↪",decimal=".",index_col=0)
```

[61]:
```
g=plot_assoc_clustered(assoc_df,file=None)
```

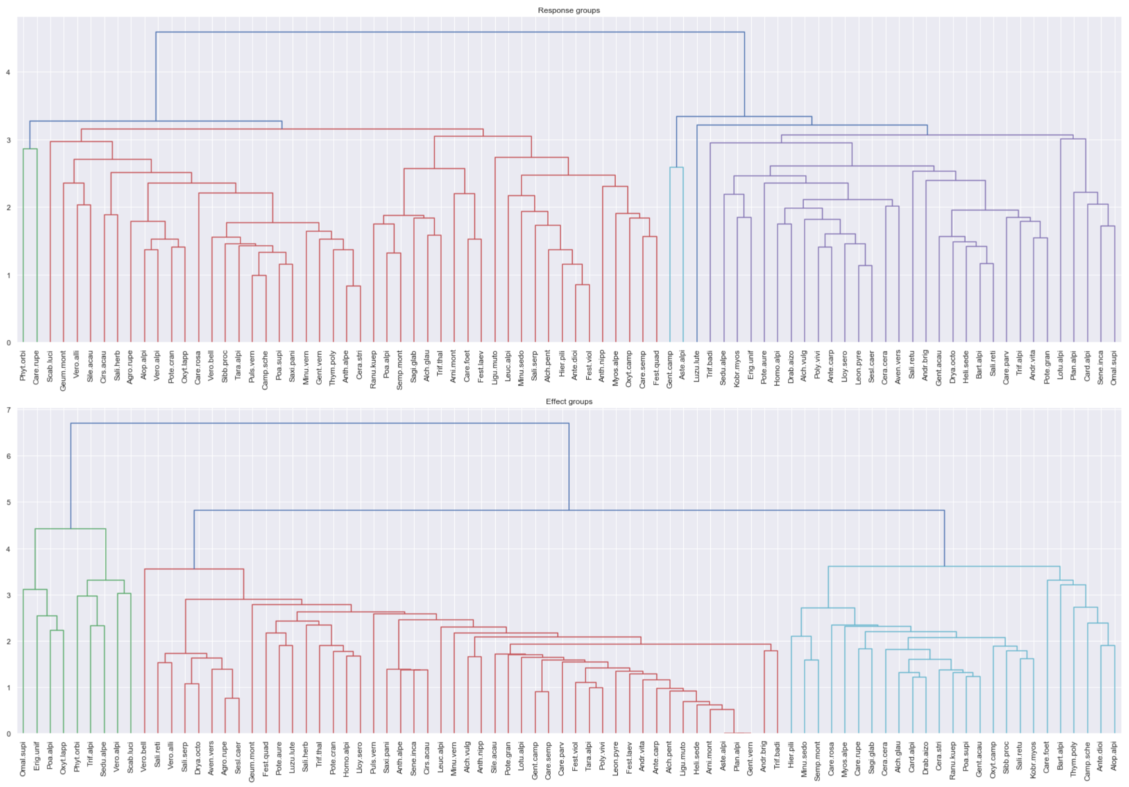


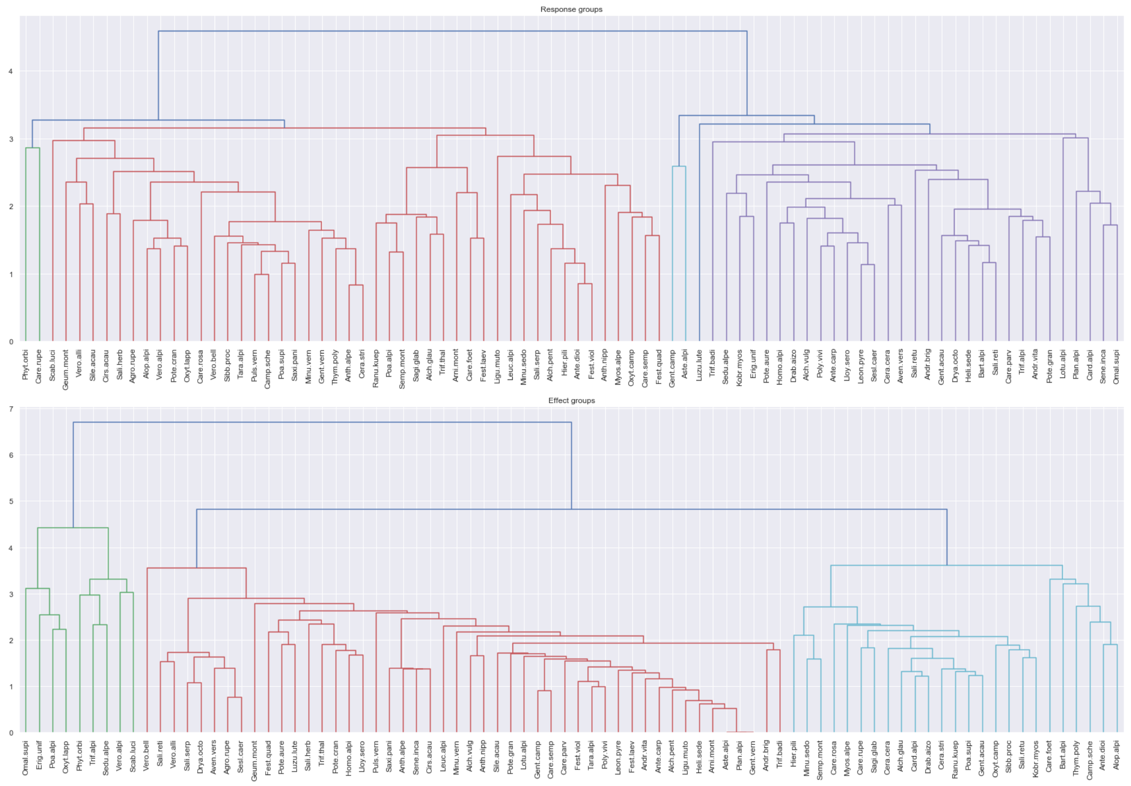
To retrieve the species clusters shown in the figure, use the object returned by the previous function and pass it to read_dendrogram as follow:

[62]:
```
labcol=[names[x] for x in g.dendrogram_col.reordered_ind]
labrow=[names[x] for x in g.dendrogram_row.reordered_ind]
```

[63]:
```
_, _, fig=plot_dendrograms(g=g,labx='Response groups',laby='Effect␣
↪groups',names=names)
fig
```

[63]:





[64]:
```
fig.savefig('../examples/Aravo/results/hierarchies.pdf',bbox_inches='tight')
```