A quick guide on how to use tapnet

Carsten F. Dormann

January 18, 2021

Abstract

This document presents the case study of Benadi et al. (2021), using the data of Tinoco et al. (2017). It thereby also serves as tutorial for the use of the "tapnet" package.

Contents

This supplement presents the workflow for the case study. It may also serve as tutorial for the use of tapnet functions.

References

Benadi, G., Dormann, C.F., Fründ, J., Stephan, R.& Vázquez, D.P. (2021) Quantitative prediction of interactions in bipartite networks based on traits, abundances, and phylogeny. *The American Naturalist*, in press.

Tinoco, B.A, Graham, C.H., Aguilar, J.M. & Schleuning, M. (2017) *Oikos* 126, 52–60. DOI: 10.1111/oik.02998

1 Data preparation

```
> library(tapnet)
> data(Tinoco)
```

First, we have to put all information into a single 'tapnet' object, using `make_tapnet`:

```
> # Produce tapnet objects using each network separately (1=Forest, 2=shrub, 3=cattle farm)
> tapnet_web1 <- make_tapnet(tree_low = plant_tree, tree_high = humm_tree,
+                            networks = networks[1], traits_low = plant_traits,
+                            traits_high = humm_traits, abun_low=plant_abun[1],
+                            abun_high=humm_abun[1], npems_lat = 4)
> tapnet_web2 <- make_tapnet(tree_low = plant_tree, tree_high = humm_tree,
+                            networks = networks[2], traits_low = plant_traits,
+                            traits_high = humm_traits, abun_low=plant_abun[2],
+                            abun_high=humm_abun[2], npems_lat = NULL)
```

Note:
(a) We use only 4 phylogenetic eigenvectors (PEMs, for "maps", and because PE is too short to be unambiguous) for the first network, but all for the other two for evaluation. The "value" NULL enforces the use of all PEMs.
(b) Because each web has only some species present, some PEMs will automatically be dropped (those relevant only for the missing species). As a consequence, web2 may now **not** have the same PEMs used for web1 (which are required for prediction from web1 to web 2). Let's check:

```
> colnames(tapnet_web1$networks[[1]]$pems$low) # names of fitted PEMs

[1] "V_1" "V_2" "V_3" "V_7"

> colnames(tapnet_web1$networks[[1]]$pems$high)

[1] "V_1" "V_3" "V_6" "V_8"

> colnames(tapnet_web2$networks[[1]]$pems$low) # names of PEMs all present

 [1] "V_1"  "V_2"  "V_3"  "V_5"  "V_7"  "V_8"  "V_10" "V_11" "V_16" "V_19"
[11] "V_20" "V_21" "V_22" "V_25" "V_27" "V_28"

> colnames(tapnet_web2$networks[[1]]$pems$high) # V_8 (high) is missing!

[1] "V_1"  "V_3"  "V_4"  "V_6"  "V_7"  "V_9"  "V_11" "V_12" "V_13"
```

We can see that for the lower level, all PEMs were computed for web2, but not for the higher level, where `V_8` is missing. We compute that, using the helper function `pems_from_tree`, and add it to the tapnet-object. (Because this function is not exported, we need to use the triple-colon when calling it explicitly from the package.) Then we do the same with web3.

```
> tapnet_web2$networks[[1]]$pems$high$V_8 <- tapnet:::pems_from_tree(humm_tree)[colnames(
+   tapnet_web2$networks[[1]]$web), "V_8"]
> colnames(tapnet_web2$networks[[1]]$pems$high) # check: complete!

 [1] "V_1"  "V_3"  "V_4"  "V_6"  "V_7"  "V_9"  "V_11" "V_12" "V_13" "V_8"
```

Using the `colnames(.)`-line we can check that they are now complete (not shown).
As an additional preliminary step, we may want to check for correlation between the phylogenetic eigenvectors and the observed traits:

```
> cor(cbind(tapnet_web1$networks[[1]]$pems$low, tapnet_web1$networks[[1]]$traits$low))

                         V_1         V_2         V_3         V_7
V_1                1.00000000 -0.03323508  0.19929407  0.01950265
V_2               -0.03323508  1.00000000 -0.02390179  0.02484880
V_3                0.19929407 -0.02390179  1.00000000  0.02538867
V_7                0.01950265  0.02484880  0.02538867  1.00000000
Corolla_length_mm  0.30147805 -0.14321786  0.23301956 -0.40838875
                  Corolla_length_mm
V_1                       0.3014780
V_2                      -0.1432179
V_3                       0.2330196
V_7                      -0.4083887
Corolla_length_mm         1.0000000

> cor(cbind(tapnet_web1$networks[[1]]$pems$high, tapnet_web1$networks[[1]]$traits$high))

                           V_1         V_3         V_6         V_8
V_1                  1.0000000 -0.21978007 -0.11948531 -0.14519738
V_3                 -0.2197801  1.00000000  0.07786362  0.01687728
V_6                 -0.1194853  0.07786362  1.00000000  0.28972218
V_8                 -0.1451974  0.01687728  0.28972218  1.00000000
Bill_length_mean_mm  0.5839393  0.23682088  0.10121502 -0.28068808
                    Bill_length_mean_mm
V_1                           0.5839393
V_3                           0.2368209
V_6                           0.1012150
V_8                          -0.2806881
Bill_length_mean_mm           1.0000000
```

In this case, correlations are moderate (-0.4 for the strongest lower-level and 0.58 for the higher-level traits) and indicate some phylogenetic signal in the observed trait. Note, however, that latent traits are linear combinations of phylogenetic traits and this correlation does not check for collinearity with such a construct. We shall do that after fitting.

2 *Fit web1 using the tapnet approach*

We here assume that all trait matches are best described using a normal distribution. Alternatively, we could use the shifted log-normal. Next, we evaluate the goodness-of-fit of this fit:

```
> fit_web1 <- fit_tapnet(tapnet = tapnet_web1, method="SANN") # very slow, but reliable
> #fit_web1 <- fit_tapnet(tapnet = tapnet_web1) # the default way
> #fit_web1ln <- fit_tapnet(tapnet = tapnet_web1, tmatch_type_obs = "shiftlnorm",
> #                         ini=fit_web1$opt$par*2) # requires some tempering with ini
> gof_web1_norm <- gof_tapnet(fit_web1)
> gof_web1_norm

$bc_sim_web
[1] 0.4999073

$cor_web
[1] 0.5132361

$net_indices
$net_indices[[1]]
```

```
          Index   Observed       Mean     Median       q2.5      q97.5
1   connectance  0.3250000  0.6970894  0.6875000  0.6118421  0.8015873
2          NODF 62.5076453 74.8523892 75.1287550 65.9083271 81.8014995
3 weighted NODF 39.1972477 56.0884635 56.4477148 47.6381727 63.4758859
4            H2  0.4496136  0.1325055  0.1323167  0.1127739  0.1516796
```

The goodness-of-fit function returns the similarity between fitted and observed network expressed as Bray-Curtis similarity (`bc_sim_web`), where 0.50 is not a bad value; as the correlation between fitted and observed number of interactions, expressed as Spearman correlation (`cor_web`), which is our key comparison criterion at 0.52; and, finally, some selected network indices were computed for the observed and repeated draws from the fitted multinomial distribution. In this case, none of the four indices includes the observed even in the 95% confidence interval (i.e. not good).
We can also have a look at the fitted model parameters:

```
> fit_web1

$par_opt
$par_opt$lat_low
       V_1        V_2        V_3        V_7
11.6145641  5.3955446 -0.4267704 11.8346621

$par_opt$lat_high
       V_1        V_3        V_6        V_8
-6.7383952 -3.2166957 -0.7118977  2.3225681

$par_opt$pem_shift
pem_shift
 3.211324

$par_opt$tmatch_width_pem
tmatch_width_pem
        1.383812

$par_opt$tmatch_width_obs
tmatch_width_obs1
         6.270236

$par_opt$delta
      delta
0.008112464


$tmatch_type_pem
[1] "normal"

$tmatch_type_obs
[1] "normal"

$lambda
[1] 0

$method
[1] "SANN"

$maxit
```

```
[1] 50000

$opt
$opt$par
              V_1               V_2               V_3               V_7
        2.4522598         5.3955446        -0.4267704        11.8346621
              V_1               V_3               V_6               V_8
       -6.7383952        -3.2166957        -0.7118977         2.3225681
        pem_shift  tmatch_width_pem tmatch_width_obs1             delta
        3.2113237         0.3248418         1.8358140        -4.8062081

$opt$value
[1] 1283.28

$opt$counts
function gradient
   50000       NA

$opt$convergence
[1] 0

$opt$message
NULL


attr(,"class")
[1] "fitted.tapnet"
attr(,"tapnet_name")
[1] "tapnet_web1"
```

The output is a bit confusing, as it contains the fitted parameters twice: first, under `par_opt` in the interpretable form, i.e. back-transformed for those parameters that were constraint (PEM 1, the standard deviations of the trait-matching function and δ); then again, under `opt`, in their untransformed form, as spit out by `optim`.
At least two things are interesting here:

1. The standard deviation of the trait-matching function (the normal, in this case) for the observed traits is rather wide (at 7.6, see `par_opt$tmatch_width_obs`). Typically, this indicates that the traits were not fitting very well to each other and the model did not find the observed traits useful. (We have seen much worse, with values > 1000, though.)

2. The value of δ is (practically) 1. A value of 1 indicates that traits (observed and latent) are as important as the abundance.

3. Putting the two previous points together: this model hinges on the matching of the phylogenetic-informed latent traits. One reason may be that the phylogenies code up the effect of the traits, so that the trait has no remaining additional effect. Parameter `lambda` imposes a shrinkage to prioritise the observed trait effect over the latent traits. (Imposing some shrinkge, e.g. setting `lambda=0.1`, in this case reduced predictive fit to below the abundance-only model.)

4. The absolute values of the PEM-parameters matters little.

With the fitted object, we can correlate the latent and the observed traits. The code is rather ugly, since we need to access data in the belly of the object:

```
> fitted_lin_low <- fit_web1$par_opt$lat_low[which(names(fit_web1$par_opt$lat_low) %in%
+                                colnames(tapnet_web1$networks[[1]]$pems$low))]
```

```
> fitted_lat_low <- as.vector(scale(rowSums(matrix(fitted_lin_low,
+                             nrow = nrow(tapnet_web1$networks[[1]]$pems$low),
+                             ncol = ncol(tapnet_web1$networks[[1]]$pems$low), byrow = TRUE) *
+                             tapnet_web1$networks[[1]]$pems$low)))
> cor(fitted_lat_low, tapnet_web1$networks[[1]]$traits$low)

     Corolla_length_mm
[1,]        -0.1439251

> fitted_lin_high <- fit_web1$par_opt$lat_high[which(names(fit_web1$par_opt$lat_high) %in%
+                                       colnames(tapnet_web1$networks[[1]]$pems$high))]
> fitted_lat_high <- as.vector(scale(rowSums(matrix(fitted_lin_high,
+                               nrow = nrow(tapnet_web1$networks[[1]]$pems$high),
+                               ncol = ncol(tapnet_web1$networks[[1]]$pems$high), byrow = TRUE
+                               tapnet_web1$networks[[1]]$pems$high)))
> cor(fitted_lat_high, tapnet_web1$networks[[1]]$traits$high)

     Bill_length_mean_mm
[1,]          -0.7381975
```

In neither case was there any correlation between latent and observed traits.
Finally, we can also check for correlation between the latent trait and the (independent) abundance of the species:

```
> cor(fitted_lat_low, tapnet_web1$networks[[1]]$abuns$low)

[1] -0.1084048

> cor(fitted_lat_high, tapnet_web1$networks[[1]]$abuns$high)

[1] 0.5678641
```

3 *Predict from fitted tapnet to new network*

Fitting characteristics are all nice and fine, but how good does tapnet predict to a new network?
To predict to a new network, we have to provide the tapnet fit-object and the abundances for that network. This allows for changing abundances, or indeed including or excluding species, independent from network observations. The tapnet-object itself is referenced by name in the fit, and is used to compute the phylogenetic information for the species in the new network. In this case, we provide the abundances based on the `tapnet_web2`-object we created earlier. We could however also simply make a list with the abundances of each level for the second network (see code in next section, or help page of `tapnet_predict`).

```
> preds2.tapnet <- predict_tapnet(fit=fit_web1, abuns=tapnet_web2$networks[[1]]$abuns)
> cor(as.vector(preds2.tapnet), as.vector(tapnet_web2$networks[[1]]$web))

[1] 0.2240482
```

So the correlation is actually not very high! Let's visualise that. To do so, we need to multiply the predictions by the number of observed interactions, as the predictions are probabilities that sum to one. Also, since interactions are approximately log-normally distributed, we depict the fit as log-log-plot.

```
> sum(tapnet_web2$networks[[1]]$web)

[1] 3979
```

```
> par(mar=c(5,5,1,1))
> plot(preds2.tapnet*3979 + 1, tapnet_web2$networks[[1]]$web + 1, log="xy", las=1,
+   xlab="predicted number of interactions + 1", ylab="observed number of interactions + 1")
> abline(0,1)
```

Finally, we can compute the multinomial log-likelihood of the data, given the prediction:

```
> dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.tapnet),
+           size=sum(tapnet_web2$networks[[1]]$web), log=T)

[1] -6330.075
```

4 *Using multiple networks*

In the tapnet approach, we can also fit several networks simultaneously, and use the resulting fit for prediction. For example, we can fit tapnet to the networks from shrub and cattle, and predict to (1 =) forest:

```
> data(Tinoco)
> tap <- make_tapnet(tree_low = plant_tree, tree_high = humm_tree, networks = networks[2:3],
+                    traits_low = plant_traits, traits_high = humm_traits,
+                    abun_low = plant_abun[2:3], abun_high=humm_abun[2:3] , npems_lat = 4)
> fit <- fit_tapnet(tap) # uses two networks for fitting!
> gof_tapnet(fit)

$bc_sim_web
[1] 0.3788003 0.4667962

$cor_web
[1] 0.3992802 0.5672018

$net_indices
$net_indices[[1]]
          Index   Observed       Mean     Median       q2.5      q97.5
1   connectance  0.2324561  0.7153267  0.7107843  0.6535088  0.7914439
2          NODF 43.5864979 81.1038697 81.5508300 72.3516295 86.5605941
3 weighted NODF 30.7805907 63.9681430 64.4677866 55.8142660 69.5472315
4            H2  0.5022179  0.1204438  0.1204516  0.1108967  0.1297563

$net_indices[[2]]
          Index   Observed       Mean     Median       q2.5      q97.5
1   connectance  0.3040936  0.6544900  0.6491228  0.6081871  0.7361111
2          NODF 56.5180793 80.9408035 81.1179531 74.2173170 86.4361991
3 weighted NODF 38.3391890 63.3156776 63.4773421 56.8836156 69.3672280
4            H2  0.3713669  0.1161482  0.1162131  0.1026874  0.1283566

> # predict to omitted forest network:
> pred1 <- predict_tapnet(fit, abuns=list("low"=plant_abun[[1]], "high"=humm_abun[[1]] ))
> cor(as.vector(pred1*sum(networks[[1]])), as.vector(networks[[1]]))

[1] 0.1188132
```

And we can do the same for the other two networks (first to 2 = shrubs, then to 3 = cattle):

```
[1] 0.1732161
```

```
[1] 0.4418444
```

If we compare this single-network predictions (see towards the end of this document), we notice a variable effect on performance: more is not necessarily better. This will probably depend on the similarity of the habitats and hence interacting species.

5 *Prediction baseline: use only abundances in the new web*

As a baseline, we compute the information contained in the abundances. If the hummingbirds had no preferences, but only interacted strictly with probability proportional to abundance, then abundant hummingbird species would interact with high probability with abundant plant species, but with low probability with rare plant species.

```
> preds2.abunonly <- (tapnet_web2$networks[[1]]$abuns$low /
+     sum(tapnet_web2$networks[[1]]$abuns$low)) %*% t(tapnet_web2$networks[[1]]$abuns$high /
+     sum(tapnet_web2$networks[[1]]$abuns$high)) * sum(tapnet_web2$networks[[1]]$web)
> cor(as.vector(preds2.abunonly), as.vector(tapnet_web2$networks[[1]]$web))
```

```
[1] 0.09473036
```

```
> par(mar=c(5,5,1,1))
> plot(preds2.abunonly + 1, tapnet_web2$networks[[1]]$web + 1, log="xy", las=1,
+   xlab="predicted number of interactions + 1", ylab="observed number of interactions + 1")
> abline(0,1)
```

And the log-likelihood:

```
> dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.abunonly),
+           size=sum(tapnet_web2$networks[[1]]$web), log=T)
```

```
[1] -9202.007
```

This means, our tapnet model is superior in prediction to the abundance-only model, indicating that the (latent and observed) traits do carry some information on the interactions. Note, however, that the prediction quality of the abundance-only is very low.

6 *Preparing tapnet data as data.frame for statistical models*

To prepare the data for a statistical model, we provide the function `tapnet2df`:

```
> web1.df <- tapnet2df(tapnet_web1)
> web2.df <- tapnet2df(tapnet_web2)
> head(web1.df)
```

```
        IDhigher                IDlower interactions     pemLV_1     pemLV_2
1 Coeligena iris Alloplectus peruvianum           13  0.19958316 -0.01721758
2 Coeligena iris    Macleania rupestris           22  0.06143519  0.01314354
3 Coeligena iris   Myrcianthes fragrans            0 -0.17608854  0.22809965
4 Coeligena iris           Salvia hirta            4  0.21167754 -0.01945177
5 Coeligena iris  Cavendishia bracteata            0  0.06143519  0.01314354
6 Coeligena iris      Mutisia lehmannii            6  0.20823846 -0.02230535
       pemLV_3      pemLV_7   pemHV_1    pemHV_3    pemHV_6   pemHV_8     abunL
1  0.298458999  0.110246748 0.3026318 0.09104722 -0.2849888 0.5647174  3.428571
2  0.014975823  0.006401991 0.3026318 0.09104722 -0.2849888 0.5647174 38.529412
3 -0.006808424 -0.005932541 0.3026318 0.09104722 -0.2849888 0.5647174 69.500000
```

```
4  0.346816571  0.323487833 0.3026318 0.09104722 -0.2849888 0.5647174 10.783505
5  0.014975823  0.006401991 0.3026318 0.09104722 -0.2849888 0.5647174 26.111111
6 -0.310376666  0.013295514 0.3026318 0.09104722 -0.2849888 0.5647174  2.740741
     abunH traitLCorolla_length_mm traitHBill_length_mean_mm
1 3.950695                   48.1                      28.1
2 3.950695                   16.5                      28.1
3 3.950695                    1.3                      28.1
4 3.950695                   22.9                      28.1
5 3.950695                   24.5                      28.1
6 3.950695                   29.3                      28.1
```

These data.frames contain all the information coded in the tapnet object, but no matches of traits.

7 *Prediction using GAM of phylogeny, traits and trait matching*

To apply the approach of Brousseau et al. (2018) to our quantitative network, we use a negative binomial GAM. It uses, as predictors, the first two PEMs of each group, plus the observed traits, plus the squared trait difference, plus the abundances.
We augment the data.frames from the previous section by the trait-matching variables (in this case only one pair: bill and corolla length):

```
> web1.df.extended <- cbind.data.frame(web1.df, "match"=(web1.df$traitHBill_length_mean_mm -
+                                                    web1.df$traitLCorolla_length_mm)^2 )
> web2.df.extended <- cbind.data.frame(web2.df, "match"=(web2.df$traitHBill_length_mean_mm -
+                                                    web2.df$traitLCorolla_length_mm)^2 )
```

Now we can fit the model. Note that we believe this approach to be statistically incorrect, as it assumes that the observed interactions are independent, when (clearly?) they are not. Each bird selects a flower based on what is on offer; thus, a decision to visit one flower implies *not* visiting another, creating a negative dependence. Anyway.

```
> library(mgcv)
> gam2 <- gam(interactions ~ s(pemLV_1, pemHV_1, bs="ts", k=24) +s(pemLV_2, pemHV_3, bs="ts",
+            k=24) + s(traitLCorolla_length_mm, k=3) + s(traitHBill_length_mean_mm, k=3) +
+            s(match, k=3) + s(abunL, k=3) + s(abunH, k=3), data=web1.df.extended, family=nb,
+            gamma=1.4)
> summary(gam2)

Family: Negative Binomial(0.171)
Link function: log

Formula:
interactions ~ s(pemLV_1, pemHV_1, bs = "ts", k = 24) + s(pemLV_2,
    pemHV_3, bs = "ts", k = 24) + s(traitLCorolla_length_mm,
    k = 3) + s(traitHBill_length_mean_mm, k = 3) + s(match, k = 3) +
    s(abunL, k = 3) + s(abunH, k = 3)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -5.969      1.703  -3.505 0.000456 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                                   edf Ref.df Chi.sq  p-value
```

```
s(pemLV_1,pemHV_1)           1.480e+00 23.000 14.647 7.86e-05 ***
s(pemLV_2,pemHV_3)           1.724e-05 23.000  0.000 0.291704
s(traitLCorolla_length_mm)   1.772e+00  1.929 10.635 0.002754 **
s(traitHBill_length_mean_mm) 1.000e+00  1.000  0.442 0.506187
s(match)                     1.000e+00  1.000 18.939 1.38e-05 ***
s(abunL)                     1.917e+00  1.993 14.489 0.001103 **
s(abunH)                     1.000e+00  1.000 12.618 0.000382 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =     -5   Deviance explained = 49.3%
-REML =    203  Scale est. = 1         n = 160
```

```
> preds2.gam <- predict(gam2, newdata=web2.df.extended)
> cor(exp(preds2.gam), web2.df$interactions)

[1] -0.01210051
```

Here, the contributions of the different predictors can easily be discerned. Abundances and trait-matching are important, but phylogenetics are (apparently) not. Note that in this case corolla length and trait matching are high ($r = 0.84$) correlated, representing the same information. That could be avoided by standardising the trait values before fitting the model. However, in that case we lose the direct interpretation of 0 indicating the same length of corolla and bill.
Again, we can plot the result:

```
> par(mar=c(5,5,1,1))
> plot(exp(preds2.gam) +1 , web2.df$interactions + 1, log="xy", las=1, xlab="predicted
+      number of interactions + 1", ylab="observed number of interactions + 1")
> abline(0,1)
```

This prediction contains no information on the observed interactions.
And the log-likelihood:

```
> dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=exp(preds2.gam) /
+             sum(exp(preds2.gam)), size=sum(tapnet_web2$networks[[1]]$web), log=T)

[1] -144756.9
```

So these values are abysmally poor.

8 *Prediction using randomForest with phylogeny, traits and trait matching*

We can use the same data with a different algorithm, in this case randomForest (as implemented in ranger). It provides an assessment of which predictors are important to the fit:

```
> library(ranger)
> rf2 <- ranger(interactions ~ ., data=web1.df.extended[, -c(1, 2)], importance="impurity")
> rf2

Ranger result

Call:
 ranger(interactions ~ ., data = web1.df.extended[, -c(1, 2)],      importance = "impurity")

Type:                             Regression
Number of trees:                  500
```

```
Sample size:                       160
Number of independent variables:  13
Mtry:                              3
Target node size:                  5
Variable importance mode:          impurity
Splitrule:                         variance
OOB prediction error (MSE):        587.3797
R squared (OOB):                   0.09158901
```

```
> sort(importance(rf2), decreasing=T)
```

```
                    abunL                     match                    pemLV_2
                13019.553                 11604.037                  8112.197
  traitLCorolla_length_mm                   pemHV_3                     abunH
                 7171.039                  6542.867                  6139.501
                  pemHV_1                   pemLV_3                   pemLV_7
                 5590.034                  5279.739                  5176.233
traitHBill_length_mean_mm                   pemLV_1                   pemHV_6
                 3981.386                  3814.208                  1183.081
                  pemHV_8
                 1144.180
```

```
> preds2.ranger <- predict(rf2, data=web2.df.extended)$predictions
> cor(preds2.ranger, web2.df$interactions)
```

```
[1] 0.2220702
```

```
> par(mar=c(5,5,1,1))
> plot(preds2.ranger +1 , web2.df$interactions + 1, log="xy", las=1, xlab="predicted number
+      of interactions + 1", ylab="observed number of interactions + 1")
> abline(0,1)
```

And the log-likelihood:

```
> dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=preds2.ranger/sum(preds2.ranger),
+           size=sum(tapnet_web2$networks[[1]]$web), log=T)
```

```
[1] -11804.72
```

This model is better than the abundance-only, GAM – and tapnet.

9 *Cross-validation for all approaches*

Here we only present the code and results for the cross-validation, where the model it fit to one web and predicts to the other two, for all three combinations. We use the same settings and data preparation as in the sections above.

9.1 *Using tapnet*

```
> tapnet_web1 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[1],
+                            traits_low=plant_traits, traits_high=humm_traits,
+                            abun_low=plant_abun[1], abun_high=humm_abun[1], npems_lat=4)
> tapnet_web2 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[2],
+                            traits_low=plant_traits, traits_high=humm_traits,
+                            abun_low=plant_abun[2], abun_high=humm_abun[2], npems_lat=4)
> tapnet_web3 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[3],
+                            traits_low=plant_traits, traits_high=humm_traits,
+                            abun_low=plant_abun[3], abun_high=humm_abun[3], npems_lat=4)
```

```
> fit_web1 <- fit_tapnet(tapnet = tapnet_web1, method="SANN")
> fit_web2 <- fit_tapnet(tapnet = tapnet_web2, method="SANN")
> fit_web3 <- fit_tapnet(tapnet = tapnet_web3, method="SANN")

> -c(fit_web1$opt$value, fit_web2$opt$value, fit_web3$opt$value)

[1] -1278.490 -4319.151 -1905.161

> preds2.tapnet1 <- predict_tapnet(fit=fit_web1, abuns=tapnet_web2$networks[[1]]$abuns)
> preds3.tapnet1 <- predict_tapnet(fit=fit_web1, abuns=tapnet_web3$networks[[1]]$abuns)
> preds1.tapnet2 <- predict_tapnet(fit=fit_web2, abuns=tapnet_web1$networks[[1]]$abuns)
> preds3.tapnet2 <- predict_tapnet(fit=fit_web2, abuns=tapnet_web3$networks[[1]]$abuns)
> preds1.tapnet3 <- predict_tapnet(fit=fit_web3, abuns=tapnet_web1$networks[[1]]$abuns)
> preds2.tapnet3 <- predict_tapnet(fit=fit_web3, abuns=tapnet_web2$networks[[1]]$abuns)

> cors.tapnet <- c(
+   cor(as.vector(preds2.tapnet1), as.vector(tapnet_web2$networks[[1]]$web)),
+   cor(as.vector(preds3.tapnet1), as.vector(tapnet_web3$networks[[1]]$web)),
+   cor(as.vector(preds1.tapnet2), as.vector(tapnet_web1$networks[[1]]$web)),
+   cor(as.vector(preds3.tapnet2), as.vector(tapnet_web3$networks[[1]]$web)),
+   cor(as.vector(preds1.tapnet3), as.vector(tapnet_web1$networks[[1]]$web)),
+   cor(as.vector(preds2.tapnet3), as.vector(tapnet_web2$networks[[1]]$web))
+ )
> cors.tapnet

[1] 0.12964284 0.47227021 0.10815367 0.47481137 0.03190397 0.10874489

> ellCV.tapnet <- c(
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.tapnet1),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=as.vector(preds3.tapnet1),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=as.vector(preds1.tapnet2),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=as.vector(preds3.tapnet2),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=as.vector(preds1.tapnet3),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.tapnet3),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T)
+ )
> ellCV.tapnet

[1] -8022.705 -2700.642 -2887.601 -4527.046 -3144.584 -8730.398
```

9.2 *Fit of the baseline, abundance-only model*

```
> preds1.abunonly <- (tapnet_web1$networks[[1]]$abuns$low /
+         sum(tapnet_web1$networks[[1]]$abuns$low)) %*% t(tapnet_web1$networks[[1]]$abuns$high
+         sum(tapnet_web1$networks[[1]]$abuns$high)) / sum(tapnet_web1$networks[[1]]$web)
> preds2.abunonly <- (tapnet_web2$networks[[1]]$abuns$low /
+       sum(tapnet_web2$networks[[1]]$abuns$low)) %*% t(tapnet_web2$networks[[1]]$abuns$high /
+       sum(tapnet_web2$networks[[1]]$abuns$high)) / sum(tapnet_web2$networks[[1]]$web)
> preds3.abunonly <- (tapnet_web3$networks[[1]]$abuns$low /
+       sum(tapnet_web3$networks[[1]]$abuns$low)) %*% t(tapnet_web3$networks[[1]]$abuns$high /
+       sum(tapnet_web3$networks[[1]]$abuns$high)) / sum(tapnet_web3$networks[[1]]$web)
```

```
> cors.abun <- c(
+   cor(as.vector(preds2.abunonly), as.vector(tapnet_web2$networks[[1]]$web)),
+   cor(as.vector(preds3.abunonly), as.vector(tapnet_web3$networks[[1]]$web)),
+   cor(as.vector(preds1.abunonly), as.vector(tapnet_web1$networks[[1]]$web)),
+   cor(as.vector(preds3.abunonly), as.vector(tapnet_web3$networks[[1]]$web)),
+   cor(as.vector(preds1.abunonly), as.vector(tapnet_web1$networks[[1]]$web)),
+   cor(as.vector(preds2.abunonly), as.vector(tapnet_web2$networks[[1]]$web))
+ )
> cors.abun

[1] 0.09473036 0.48561887 0.24775426 0.48561887 0.24775426 0.09473036

> ellCV.abuns <- c(
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.abunonly),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=as.vector(preds3.abunonly),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=as.vector(preds1.abunonly),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=as.vector(preds3.abunonly),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=as.vector(preds1.abunonly),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=as.vector(preds2.abunonly),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T)
+ )
> ellCV.abuns

[1] -9202.007 -2706.214 -2149.702 -2706.214 -2149.702 -9202.007
```

Comparing these values to the fits of the tapnet we see that they are similar, sometimes tapnet is better, sometimes abundance-only.

9.3 *The GAM-approach*

```
> web3.df <- tapnet2df(tapnet_web3)
> web3.df.extended <- cbind.data.frame(web3.df, "match"=(web3.df$traitHBill_length_mean_mm -
+                                                          web3.df$traitLCorolla_length_mm)^2 )

> gam1 <- gam(interactions ~ s(pemLV_1, pemHV_1, bs="ts", k=24) +
+   s(pemLV_2, pemHV_3, bs="ts", k=24) + s(traitLCorolla_length_mm, k=3) +
+   s(traitHBill_length_mean_mm, k=3) + s(match, k=3) + s(abunL, k=3) +
+     s(abunH, k=3), data=web1.df.extended, family=nb, gamma=1.4)
> gam2 <- gam(interactions ~ s(pemLV_1, pemHV_1, bs="ts", k=24) +
+   s(pemLV_2, pemHV_3, bs="ts", k=24) + s(traitLCorolla_length_mm, k=3) +
+   s(traitHBill_length_mean_mm, k=3) + s(match, k=3) + s(abunL, k=3) +
+   s(abunH, k=3), data=web2.df.extended, family=nb, gamma=1.4)
> gam3 <- gam(interactions ~ s(pemLV_1, pemHV_1, bs="ts", k=24) +
+   s(pemLV_2, pemHV_3, bs="ts", k=24) + s(traitLCorolla_length_mm, k=3) +
+   s(traitHBill_length_mean_mm, k=3) + s(match, k=3) + s(abunL, k=3) +
+   s(abunH, k=3), data=web3.df.extended, family=nb, gamma=1.4)

> preds2.gam1 <- predict(gam1, newdata=web2.df.extended, type="response")
> preds3.gam1 <- predict(gam1, newdata=web3.df.extended, type="response")
> preds1.gam2 <- predict(gam2, newdata=web1.df.extended, type="response")
```

```
> preds3.gam2 <- predict(gam2, newdata=web3.df.extended, type="response")
> preds1.gam3 <- predict(gam3, newdata=web1.df.extended, type="response")
> preds2.gam3 <- predict(gam3, newdata=web2.df.extended, type="response")

> cors.gam <- c(
+   cor(preds2.gam1, web2.df$interactions),
+   cor(preds3.gam1, web3.df$interactions),
+   cor(preds1.gam2, web1.df$interactions),
+   cor(preds3.gam2, web3.df$interactions),
+   cor(preds1.gam3, web1.df$interactions),
+   cor(preds2.gam3, web2.df$interactions)
+ )

> ellCV.gam <- c(
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=preds2.gam1/sum(preds2.gam1),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=preds3.gam1/sum(preds3.gam1),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=preds1.gam2/sum(preds1.gam2),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=preds3.gam2/sum(preds3.gam2),
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=preds1.gam3/sum(preds1.gam3),
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=preds2.gam3/sum(preds2.gam3),
+             size=sum(tapnet_web2$networks[[1]]$web), log=T)
+ )

> cors.gam

[1] -0.01210051  0.33249537  0.25840986  0.13461081 -0.01344406 -0.01974305

> ellCV.gam

[1]  -77771.757   -6943.118   -2750.557   -6029.040  -26537.792 -162237.116
```

Compared to the tapnet, this GAM-approach is inferior in all instances.

9.4 *The randomForest approach*

Note that we now have to append the correct PEMs to the data. For the GAM, we only used the first PEM, but here the first 4 (or all)!

```
> tapnet_web1 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[1],
+               traits_low=plant_traits, traits_high=humm_traits, abun_low=plant_abun[1],
+               abun_high=humm_abun[1], npems_lat=NULL, use.all.pems=T)
> tapnet_web2 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[2],
+               traits_low=plant_traits, traits_high=humm_traits, abun_low=plant_abun[2],
+               abun_high=humm_abun[2], npems_lat=NULL, use.all.pems=T)
> tapnet_web3 <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=networks[3],
+               traits_low=plant_traits, traits_high=humm_traits, abun_low=plant_abun[3],
+               abun_high=humm_abun[3], npems_lat=NULL, use.all.pems=T)

> web1.df <- tapnet2df(tapnet_web1)
> web1.df.extended <- cbind.data.frame(web1.df, "match"=(web1.df$traitHBill_length_mean_mm -
+                                                        web1.df$traitLCorolla_length_mm)^2 )
```

```
> web2.df <- tapnet2df(tapnet_web2)
> web2.df.extended <- cbind.data.frame(web2.df, "match"=(web2.df$traitHBill_length_mean_mm -
+                                                    web2.df$traitLCorolla_length_mm)^2 )
> web3.df <- tapnet2df(tapnet_web3)
> web3.df.extended <- cbind.data.frame(web3.df, "match"=(web3.df$traitHBill_length_mean_mm -
+                                                    web3.df$traitLCorolla_length_mm)^2 )

> rf1 <- ranger(interactions ~ ., data=web1.df.extended[, -c(1, 2)], importance="impurity")
> rf2 <- ranger(interactions ~ ., data=web2.df.extended[, -c(1, 2)], importance="impurity")
> rf3 <- ranger(interactions ~ ., data=web3.df.extended[, -c(1, 2)], importance="impurity")

> head(sort(round(importance(rf1)), decreasing=T))

pemHV_12 pemHV_11 pemLV_26    abunH    match  pemLV_5
    4209     4068     4003     3867     3739     3696

> head(sort(round(importance(rf2)), decreasing=T))

   abunH    match  pemHV_6 pemLV_19  pemHV_8  pemLV_1
  114624    69253    49380    46645    46561    46267

> head(sort(round(importance(rf3)), decreasing=T))

   abunH pemHV_10 pemHV_13  pemHV_3    abunL pemHV_12
   29299    22659    22586    18543    18206    17579

> preds2.ranger1 <- predict(rf1, data=web2.df.extended)$predictions
> preds3.ranger1 <- predict(rf1, data=web3.df.extended)$predictions
> preds1.ranger2 <- predict(rf2, data=web1.df.extended)$predictions
> preds3.ranger2 <- predict(rf2, data=web3.df.extended)$predictions
> preds1.ranger3 <- predict(rf3, data=web1.df.extended)$predictions
> preds2.ranger3 <- predict(rf3, data=web2.df.extended)$predictions

> cors.rf <- c(
+   cor(preds2.ranger1, web2.df$interactions),
+   cor(preds3.ranger1, web3.df$interactions),
+   cor(preds1.ranger2, web1.df$interactions),
+   cor(preds3.ranger2, web3.df$interactions),
+   cor(preds1.ranger3, web1.df$interactions),
+   cor(preds2.ranger3, web2.df$interactions)
+ )

> ellCV.rf <- c(
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=preds2.ranger1/sum(preds2.ranger1
+             size=sum(tapnet_web2$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=preds3.ranger1/sum(preds3.ranger1
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=preds1.ranger2/sum(preds1.ranger2
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web3$networks[[1]]$web), prob=preds3.ranger2/sum(preds3.ranger2
+             size=sum(tapnet_web3$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web1$networks[[1]]$web), prob=preds1.ranger3/sum(preds1.ranger3
+             size=sum(tapnet_web1$networks[[1]]$web), log=T),
+   dmultinom(as.vector(tapnet_web2$networks[[1]]$web), prob=preds2.ranger3/sum(preds2.ranger3
+             size=sum(tapnet_web2$networks[[1]]$web), log=T)
+ )
```

```
> cors.rf

[1] 0.2380470 0.3783774 0.3359350 0.1308790 0.3992643 0.1721153

> ellCV.rf

[1] -12991.185  -4667.831  -3123.259  -5757.930  -2082.967 -10430.639
```

Compared to tapnet, the randomForest approach is consistently better in validation (correlations), but yields very poor fits in terms of log-likelihood.

9.5 *Summary of cross-validation results*

To summarise all of this, here are the **correlations**, sorted by overall prediction quality:

```
> all.cors.res <- rbind(cors.tapnet, cors.abun, cors.rf, cors.gam)
> all.cors.res <- cbind(all.cors.res, rowMeans(all.cors.res))
> colnames(all.cors.res) <- c("1 to 2", "1 to 3", "2 to 1", "2 to 3", "3 to 1", "3 to 2",
+                             "average")
> round(all.cors.res, 2)

            1 to 2 1 to 3 2 to 1 2 to 3 3 to 1 3 to 2 average
cors.tapnet   0.13   0.47   0.11   0.47   0.03   0.11    0.22
cors.abun     0.09   0.49   0.25   0.49   0.25   0.09    0.28
cors.rf       0.24   0.38   0.34   0.13   0.40   0.17    0.28
cors.gam     -0.01   0.33   0.26   0.13  -0.01  -0.02    0.11
```

And here the **log-likelihoods** on the hold-out (larger, i.e. less negative, is better):

```
> all.ellCV.res <- rbind(ellCV.tapnet, ellCV.abuns, ellCV.rf, ellCV.gam)
> all.ellCV.res <- cbind(all.ellCV.res, rowMeans(all.ellCV.res))
> colnames(all.ellCV.res) <- colnames(all.cors.res)
> round(all.ellCV.res)

             1 to 2 1 to 3 2 to 1 2 to 3 3 to 1  3 to 2 average
ellCV.tapnet  -8023  -2701  -2888  -4527  -3145   -8730   -5002
ellCV.abuns   -9202  -2706  -2150  -2706  -2150   -9202   -4686
ellCV.rf     -12991  -4668  -3123  -5758  -2083  -10431   -6509
ellCV.gam    -77772  -6943  -2751  -6029 -26538 -162237  -47045
```

In summary, these results show that tapnet, abundance-only and randomForest yield very similar performance on prediction to a new network, based only on abundances. In absolute terms, these predictions are poor. Among the possible explanations for the poor prediction we think we can exclude the very skewed distribution of interaction intensities, as the data can be fit satisfactorily (by tapnet and randomForest). A more likely explanation is that bipartite networks have no representation of interactions within a group, e.g. competitive interactions among hummingbirds. As they differ in size, a larger species not occurring in forest may "bully" smaller birds into deviating from their feeding preferences; or very similar species may display character displacement in the presence of the other.

9.6 *Fits*

Just for completeness, here also the information on the goodness-of-fit for all approaches. This is an inferior measure of an approache's performance if the aim is prediction. As sometimes people want to only use an approach in an exploratory way, e.g. to identify which elements contribute to describing the data, we show the same measures as before for the fit (again sorted by quality of prediction, not by quality of fit).

```
> fits.cors.res <- rbind(
+   "tapnet"=cbind(
+     cor(as.vector(tapnet_web1$networks[[1]]$web), as.vector(predict_tapnet(fit_web1,
+                                          abuns=tapnet_web1$networks[[1]]$abuns))),
+     cor(as.vector(tapnet_web2$networks[[1]]$web), as.vector(predict_tapnet(fit_web2,
+                                          abuns=tapnet_web2$networks[[1]]$abuns))),
+     cor(as.vector(tapnet_web3$networks[[1]]$web), as.vector(predict_tapnet(fit_web3,
+                                          abuns=tapnet_web3$networks[[1]]$abuns)))
+   ),
+   "abuns"=cbind(
+         cor(as.vector(tapnet_web1$networks[[1]]$web), as.vector(preds1.abunonly)),
+         cor(as.vector(tapnet_web2$networks[[1]]$web), as.vector(preds2.abunonly)),
+         cor(as.vector(tapnet_web3$networks[[1]]$web), as.vector(preds3.abunonly))
+   ),
+   "rf"=cbind(
+     cor(predict(rf1, data=web1.df.extended)$predictions, web1.df$interactions),
+     cor(predict(rf2, data=web2.df.extended)$predictions, web2.df$interactions),
+     cor(predict(rf3, data=web3.df.extended)$predictions, web3.df$interactions)
+   ),
+   "gam"=cbind(
+     cor(predict(gam1, data=web1.df.extended, type="response"), web1.df$interactions),
+     cor(predict(gam2, data=web2.df.extended, type="response"), web2.df$interactions),
+     cor(predict(gam3, data=web3.df.extended, type="response"), web3.df$interactions)
+   )
+ )
> fits.cors.res <- cbind(fits.cors.res, rowMeans(fits.cors.res))
> colnames(fits.cors.res) <- c("1 to 1", "2 to 2", "3 to 3", "average")
> round(fits.cors.res, 2)

     1 to 1 2 to 2 3 to 3 average
[1,]   0.58   0.70   0.69    0.66
[2,]   0.25   0.09   0.49    0.28
[3,]   0.93   0.92   0.91    0.92
[4,]   0.56   0.29   0.40    0.42
```

10 *Comparison with tapnet using marginal totals as abundances*

Often, no information on species abundances are available or reported (e.g. in the interaction web data base). For null models, we thus typically use marginal totals as substitute for external abundances, arguing that when species abundances are not strongly dependent on the network itself, these marginal totals should be highly correlated with a species' overall abundance in that habitat.
Here, we show how misleading this reasoning is for the situation of the Tinoco data. We follow the same approach as above, but now withhold the information of independent plant and pollinator abundance, and use marginal totals instead. For the test data, this leads to the weird situation that we know how often a species has been observed in an interaction, but pretend not to know with whom. (The situation would be that of two ecologists collecting the data side-by-side, with one only noting own the plants visited, but not the birds visiting them, and the other the other way around. Concieveable, but unlikely.)

```
> tapnet_web1.w <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=
+               networks[1], traits_low=plant_traits, traits_high=humm_traits, npems_lat=4)
> tapnet_web2.w <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=
+               networks[2], traits_low=plant_traits, traits_high=humm_traits, npems_lat=4)
```

```
> tapnet_web3.w <- make_tapnet(tree_low=plant_tree, tree_high=humm_tree, networks=
+               networks[3], traits_low=plant_traits, traits_high=humm_traits, npems_lat=4)

> fit_web1.w <- fit_tapnet(tapnet=tapnet_web1.w)
> fit_web2.w <- fit_tapnet(tapnet=tapnet_web2.w)
> fit_web3.w <- fit_tapnet(tapnet=tapnet_web3.w)

> preds2.tapnet1.w <- predict_tapnet(fit=fit_web1.w, abuns=tapnet_web2.w$networks[[1]]$abuns)
> preds3.tapnet1.w <- predict_tapnet(fit=fit_web1.w, abuns=tapnet_web3.w$networks[[1]]$abuns)
> preds1.tapnet2.w <- predict_tapnet(fit=fit_web2.w, abuns=tapnet_web1.w$networks[[1]]$abuns)
> preds3.tapnet2.w <- predict_tapnet(fit=fit_web2.w, abuns=tapnet_web3.w$networks[[1]]$abuns)
> preds1.tapnet3.w <- predict_tapnet(fit=fit_web3.w, abuns=tapnet_web1.w$networks[[1]]$abuns)
> preds2.tapnet3.w <- predict_tapnet(fit=fit_web3.w, abuns=tapnet_web2.w$networks[[1]]$abuns)

> cors.tapnet.w <- c(
+   cor(as.vector(preds2.tapnet1.w), as.vector(tapnet_web2.w$networks[[1]]$web)),
+   cor(as.vector(preds3.tapnet1.w), as.vector(tapnet_web3.w$networks[[1]]$web)),
+   cor(as.vector(preds1.tapnet2.w), as.vector(tapnet_web1.w$networks[[1]]$web)),
+   cor(as.vector(preds3.tapnet2.w), as.vector(tapnet_web3.w$networks[[1]]$web)),
+   cor(as.vector(preds1.tapnet3.w), as.vector(tapnet_web1.w$networks[[1]]$web)),
+   cor(as.vector(preds2.tapnet3.w), as.vector(tapnet_web2.w$networks[[1]]$web))
+ )
> cors.tapnet.w

[1] 0.8044350 0.7580522 0.8108440 0.5426476 0.7749173 0.7946178

> mean(cors.tapnet.w)

[1] 0.7475857
```

Please post comments, corrections or additions through `github.com/biometry/tapnet`.