Simulation study

Load R package

```
library(DFAclust)
#> Le chargement a nécessité le package : TMB
#> Le chargement a nécessité le package : RcppEigen
```

Reproducible time series

Let's simulate some data. We will create time series for 15 species with 20 time steps and simulate observation error time series for each species. We start by simulating 4 latent trends, then we create sets of factor loadings from 2 simulated cluster centres that we will build and finally, using the simulated latent trends and factor loadings, we get simulated time series. This is a reproducible example but normally, you already have your data.

Set parameters

```
seed_id <- 0 # starting seed

id_vec <- c() # vector of species id

n_sp_init <- 15 # number of species time series

nb_group_exp <- 2 # number of expected clusters

cum_perc <- c(10,5) # distribution of species among clusters

n_y <- 20 # number of time steps

n_y_burn <- 6 # first steps to burn

n_lt <- 4 # number of latent trends
```

Simulate latent trends

```
y_init <- data.frame(t(rep(NA,n_y+n_y_burn)))

for(i in 1:n_lt){

    set.seed(i+10)

    y_ts <- c()

    y_ts[1] <- rnorm(n = 1, mean = 0, sd = 1)

    for (t in 2:(n_y+n_y_burn)) {

        r.w <- rnorm(n = 1, mean = 0, sd = 1)

        y_ts[t] <- y_ts[t - 1] + r.w

    }

    y_ts <- y_ts + abs(min(y_ts))+1

    y_ts <- exp(log(y_ts)-mean(log(y_ts)))

    y_init[i,] <- y_ts

}
```

Simulate cluster centres

```
for(g in 1:nb_group_exp){

    nb_sp_g <- cum_perc[g]

    assign(paste0("nb_sp_g",g),nb_sp_g)
```

```r
    id_vec <- c(id_vec,rep(g,nb_sp_g))

    for(lt in 1:n_lt){

        seed_id <- seed_id + 1

        set.seed(seed_id)

        mean_u_g <- runif(1, -1, 1)

        lf_u_g <- rnorm(nb_sp_g, mean_u_g, 0.1)

        assign(paste0("mean_u",lt,"_g",g),mean_u_g) # mean of loading factors in group g for
latend trend lt

        assign(paste0("lf_u",lt,"_g",g),lf_u_g) # loading factors for each ts of group g for
latend trend lt

    }
}

id_vec <- id_vec[1:n_sp_init]
```

Simulate species time series

```r
y <- data.frame(t(rep(NA,(n_y+n_y_burn+2))))

obs_se <- data.frame(t(rep(NA,(n_y+n_y_burn+1))))

for(i in 1:n_sp_init){ # get simulated ts from loadings

    set.seed(i)

    noise <- rnorm((n_y+n_y_burn),0,0.2)

    y[i,1] <- obs_se[i,1] <- sprintf("SP%03d",i)

    y_ts <- rep(0,(n_y+n_y_burn))

    g <- id_vec[i]

    i_g <- which(which(id_vec==g)==i) # new index for i in group g

    for(lt in 1:n_lt){

        lf_u_g <- get(paste0("lf_u",lt,"_g",g))

        y_ts <- y_ts + as.numeric(y_init[lt,])*lf_u_g[i_g]

    }

    y_ts <- y_ts + noise

    y_ts <- y_ts + abs(min(y_ts)) + 1

    y_ts <- exp(scale(log(y_ts)))

    y[i,2:(n_y+n_y_burn+1)] <- y_ts

    y[i,(n_y+n_y_burn+2)] <- id_vec[i]

    obs_se[i,2:(n_y+n_y_burn+1)] <- abs(rnorm((n_y+n_y_burn),0,0.1))

    obs_se[obs_se>1] <- 1
}
```

Specify data in the right format

Three datasets should be provided for the analysis:

- `y_ts_mat` a matrix of species time-series in rows and years in columns, with species names' codes as row names and years as column names.
- `y_uncert_ts` a matrix of uncertainty in species time-series in rows and years in columns, with species names' codes as row names and years as column names.
- `species_name_ex` a dataset with two columns, one for species names and the other for species names' codes.

```r
y_ts_mat <- as.matrix(y[,(1+n_y_burn):(n_y+n_y_burn)]) # species time series

y_uncert_ts <- as.matrix(obs_se[,(1+n_y_burn):(n_y+n_y_burn)]) # observation error on time
        series

colnames(y_ts_mat) <- colnames(y_uncert_ts) <- c(1998:(1998+n_y-1)) # add years as column names

rownames(y_ts_mat) <- rownames(y_uncert_ts) <- y$X1 # add species names as row names

species_name_ex <- data.frame(name_long=sprintf("species %03d",1:nrow(y_ts_mat)), code_sp=y$X1)
        # species names and code
```

Data specification

The format and completeness of data can be checked using the `prepare_data` function. This function allows the user to log standard errors if they are not and to check for missing values in the standard error input. It also handles zeros in time series by replacing them with a percentage of the base year value (1% by default). This step is not mandatory, but is strongly recommended before using the `fit_dfa` function to run the DFA analysis. `perc_replace` is the proportion of the mean index value used to replace zeros in the species time series with `0.01` as the default (1%). It can also transform the observation error time series to their logarithmic values if they are not initially in logarithmic values (in this case, set `se_log` to `TRUE`).

```r
data_ready_dfa <- prepare_data(data_ts = y_ts_mat,data_ts_se = y_uncert_ts, se_log = TRUE,
        perc_replace = 0.01)
```

Run the DFA analysis

To run the DFA, there are several options. `nfac` corresponds to the number of latent trends. It can be specified by the user but the default is 0 to look for the optimal number of latent trends between `mintrend` and `maxtrend`. `center_option` allows the function to handle time-series centered according to the first year (`center_option` = 0) or mean-centred, which is the default (`center_option` = 1). `control` is a `list` of control options for `MakeADFun()` (default is list()).

```r
dfa_result <- fit_dfa(data_ts = data_ready_dfa$data_ts,data_ts_se =
        data_ready_dfa$data_ts_se,min_year = data_ready_dfa$min_year, max_year =
        data_ready_dfa$max_year, species_name_ordre =
        data_ready_dfa$species_name_ordre,species_sub = species_name_ex, nfac = 0, mintrend =
        1, maxtrend = 3, AIC = TRUE, center_option = 1, silent = TRUE, control = list())
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      0      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 248.1951 248.1951 248.1951 248.1951 318.0417 248.1951
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 6.202108e-05 2.618929e-04 1.934270e-05 1.200676e-04 3.691432e-05 1.572992e-04
#> AIC:  696.083453828981
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      0      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 171.0695 171.0695 171.0695 171.0695 171.0695 171.0695
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 2.777084e-05 3.026463e-04 4.092772e-05 1.693197e-05 7.291589e-05 8.313069e-05
#> AIC:  430.139005431581
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      0      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 170.9144 170.9144 170.9144 170.9144 170.9144 170.9144
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 7.041854e-05 7.307525e-05 4.842229e-05 1.405623e-04 3.535907e-05 9.586422e-05
#> AIC:  455.828792274029
```

`AIC` values showed what is optimal number of latent trends (here 2 latent trends).

Run the clustering analysis

Once the latent trends are estimated, the clustering analysis can be run using the function `cluster_dfa` and the number of iterations `nboot` can be specified. A higher number of iterations will help to estimate the stability of clusters but it takes also more time to compute.

```r
cluster_result <- cluster_dfa(data_dfa = dfa_result, species_sub = species_name_ex, nboot =
        100)
```

Plot results of DFA and clustering

Finally, the results can be plotted using the function `plot_dfa_result`.

```
dfa_result_plot <- plot_dfa_result(data_dfa = dfa_result, sdRep = cluster_result$sdRep,
        species_sub = species_name_ex, group_dfa = cluster_result$group_dfa, min_year =
        data_ready_dfa$min_year, species_name_ordre = data_ready_dfa$species_name_ordre)
#> Using name_long as id variables
```

Species time-series

The columns are set as follows:

- `code_sp`: code for species names
- `Year`: year
- `value_orig`: input values for species time-series
- `se_orig`: input values for standard errors of species time-series
- `value`: back transformed values for species time-series (should be identical to `value_orig`)
- `se`: back transformed values for species time-series (should be identical to `se_orig`)
- `pred`: predicted values for species time-series from DFA
- `pred_se`: predicted values for standard error of species time-series from DFA
- `name_long`: species names
- `pred.value_exp`: predicted values for species time-series from DFA, back transformed
- `pred_se.value_exp`: predicted values for standard error of species time-series from DFA, back transformed
- `se.value_exp`: back transformed values for species time-series for pred.value_exp
- `value_1`: values for species time-series standardised by the first value
- `pred.value_exp_1`: predicted values for species time-series from DFA standardised by the first value
- `se.value_exp_1`: back transformed values for species time-series standardised by the first value
- `pred_se.value_exp_1`: predicted values for standard error of species time-series from DFA standardised by the first value

```
head(dfa_result_plot$data_to_plot_sp)
#>   code_sp Year value_orig     se_orig     value          se       pred
#> 1   SP001 1998  0.4332800 0.010278773 0.4332800 0.010278773 -0.8426023
#> 2   SP001 2006  1.5711481 0.076317575 1.5711481 0.076317575  0.7548547
#> 3   SP001 1999  0.4102787 0.038767161 0.4102787 0.038767161 -1.3653313
#> 4   SP001 2007  4.7775548 0.016452360 4.7775548 0.016452360  1.0697025
#> 5   SP001 2000  0.4305714 0.005380504 0.4305714 0.005380504 -1.2518156
#> 6   SP001 2001  0.4678779 0.137705956 0.4678779 0.137705956 -1.1970411
#>     pred_se   name_long pred.value_exp pred_se.value_exp se.value_exp
#> 1 0.1604777 species 001      0.4305886        0.06909988  0.004453587
#> 2 0.1748108 species 001      2.1273024        0.37187534  0.119906214
#> 3 0.1831348 species 001      0.2552961        0.04675359  0.015905341
#> 4 0.1836379 species 001      2.9145123        0.53521485  0.078602050
#> 5 0.1781344 species 001      0.2859851        0.05094378  0.002316691
#> 6 0.1908569 species 001      0.3020867        0.05765533  0.064429568
#>      value_1 pred.value_exp_1 se.value_exp_1 pred_se.value_exp_1
#> 1  1.0000000        1.0000000    0.010278773           0.1604777
#> 2  3.6261725        4.9404528    0.276740692           0.8636443
#> 3  0.9469135        0.5929003    0.036709149           0.1085807
#> 4 11.0264829        6.7686710    0.181411662           1.2429844
#> 5  0.9937485        0.6641725    0.005346868           0.1183120
#> 6  1.0798510        0.7015670    0.148701912           0.1338989
```

DFA latent trends

The columns are set as follows:

- `Year`: year
- `variable`: latent trend id
- `value`: latent trend values
- `se`: standard error of latent trends
- `rot_tr`: rotated values for latent trends
- `x_mc_ts`: mean-centred latent trend values
- `x_mc_sd`: standard error of mean-centred latent trends

```
head(dfa_result_plot$data_to_plot_tr)
#>         variable Year      value        se     rot_tr    x_mc_ts   x_mc_sd
#> 1 Latent trend 1 1998  0.0000000 0.0000000  0.0000000 -2.1026075 0.5898141
#> 2 Latent trend 1 1999 -1.3045419 0.4319011 -1.2738699 -3.4070119 0.7127264
#> 3 Latent trend 1 2000 -1.0213820 0.4402599 -1.1866980 -3.1237477 0.7253654
#> 4 Latent trend 1 2001 -0.8846386 0.4756138 -1.1997870 -2.9870649 0.7386626
#> 5 Latent trend 1 2002 -0.8204051 0.4209879 -0.5760105 -2.9229031 0.6051698
#> 6 Latent trend 1 2003  1.1031343 0.4732641  1.4294700 -0.9992846 0.3946202
```

DFA loading factors

The columns are set as follows for `data_loadings`:

- `code_sp`: code for species names
- `variable`: latent trend id
- `value`: loading factors
- `se.value`: standard error of loading factors
- `name_long`: species names
- `PC1`: position of species on PCA axis

```
head(dfa_result_plot$data_loadings)
#>   code_sp       variable      value se.value   name_long        PC1
#> 1   SP001 Latent trend 1  0.3803342       NA species 001 -0.2444518
#> 2   SP001 Latent trend 2 -0.1262525       NA species 001 -0.2444518
#> 3   SP002 Latent trend 1  0.3554302       NA species 002 -0.2414481
#> 4   SP002 Latent trend 2 -0.1481433       NA species 002 -0.2414481
#> 5   SP003 Latent trend 1  0.4162919       NA species 003 -0.3194116
#> 6   SP003 Latent trend 2 -0.1972671       NA species 003 -0.3194116
```

The columns are set as follows for `exp_var_lt`:

- `name_long`: species names
- `Latent trend n`: variance of species time-series explained by latent trend n
- `Random noise`: variance of species time-series explained by random noise
- `all`: total explained variance

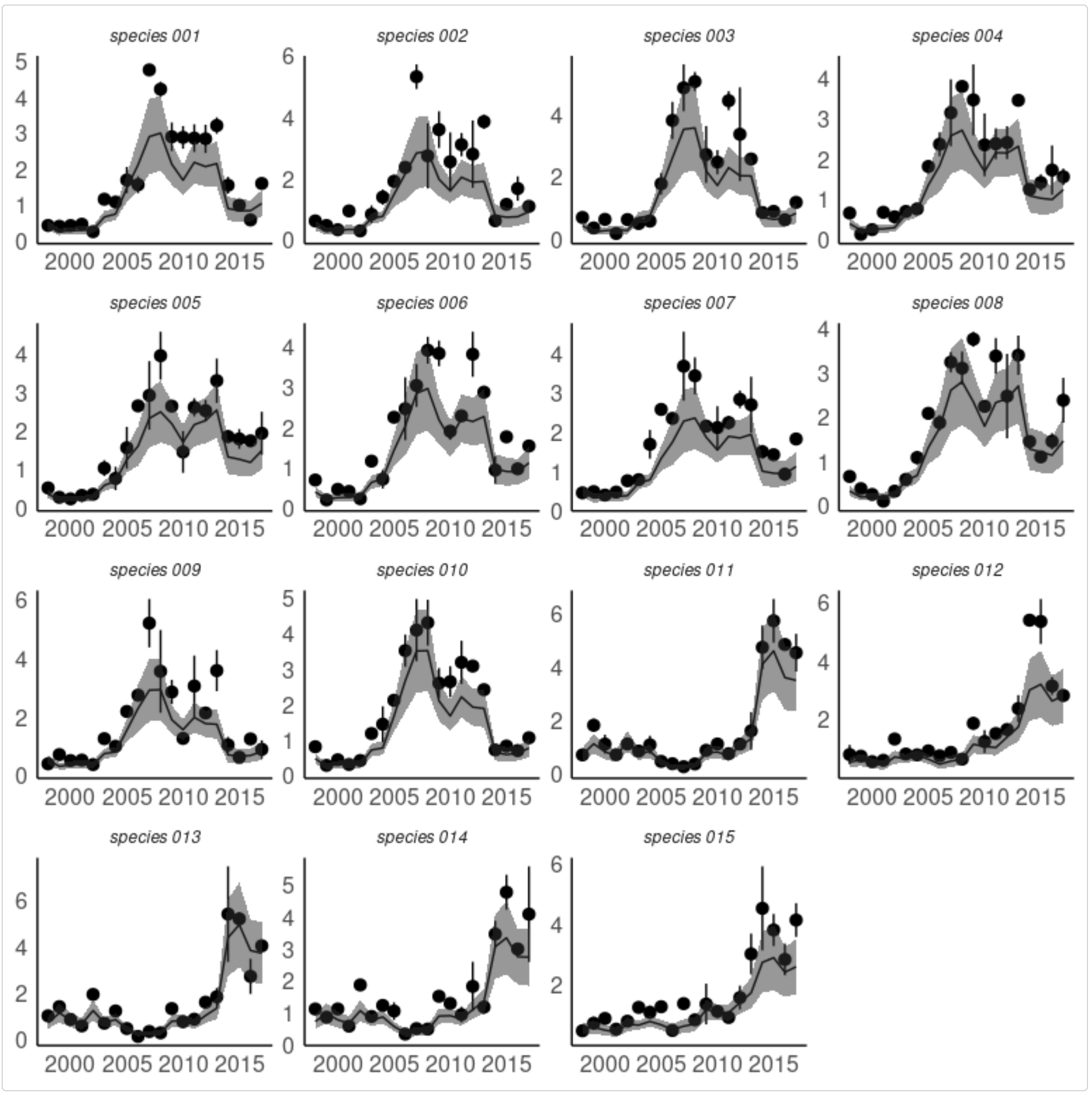
```
head(dfa_result_plot$exp_var_lt)
#>     name_long Latent trend 1 Latent trend 2 Random noise       all
#> 1 species 001      0.7935470    0.075212919    0.1312401 1.1580205
#> 2 species 002      0.7003173    0.104645660    0.1950370 1.1459665
#> 3 species 003      0.7362226    0.142197975    0.1215794 1.4953570
#> 4 species 004      0.8175171    0.027827233    0.1546556 1.0300839
#> 5 species 005      0.8870264    0.002177359    0.1107963 0.9061694
#> 6 species 006      0.8154270    0.054110617    0.1304624 1.1469534
```

Plots

Plot species time-series

Species time-series are plotted with initial data as black dots (with standard errors as vertical segments) and fitted time-series from DFA are displayed as continuous lines (with 95% CI as shaded grey area).

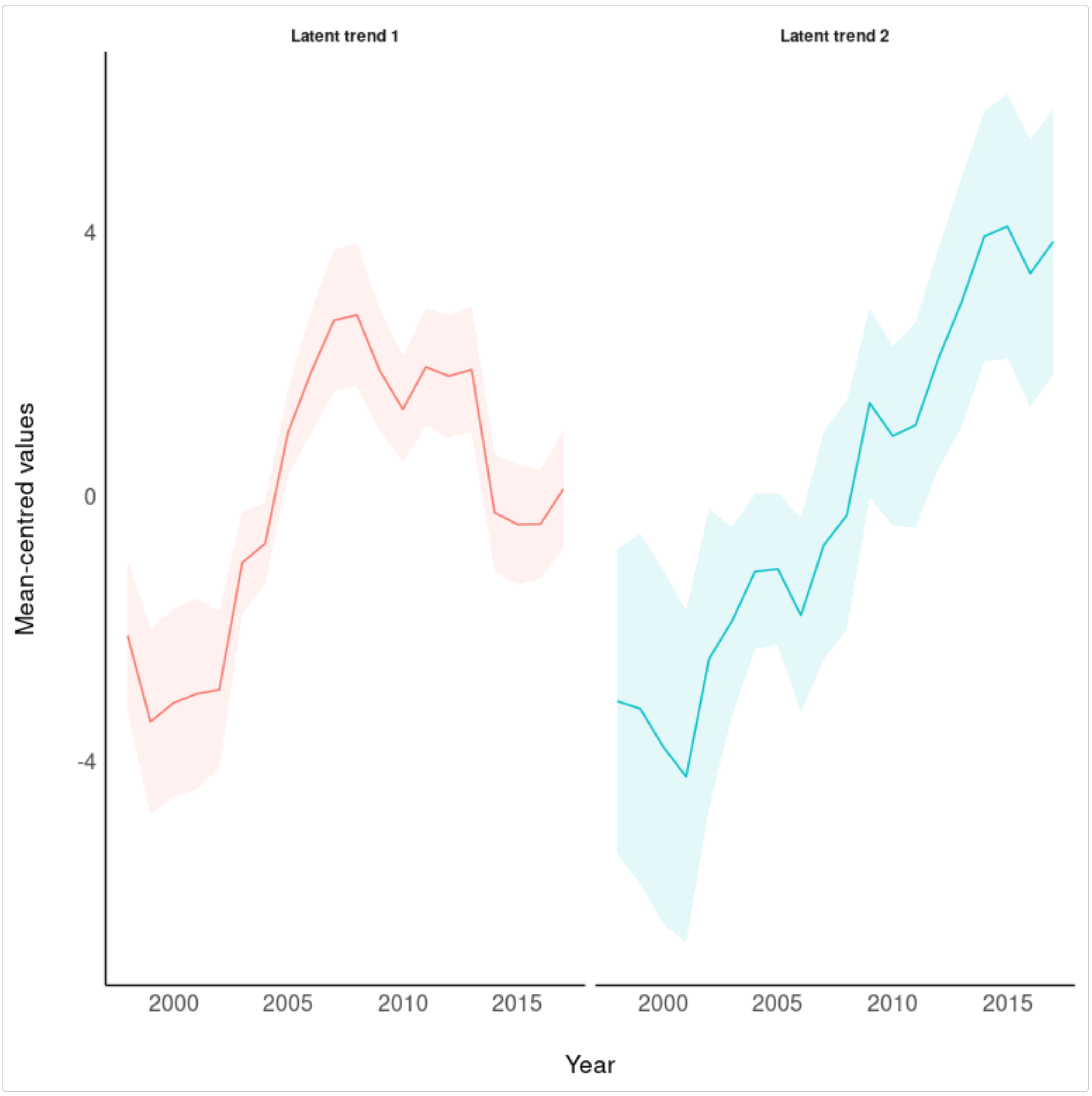
```
dfa_result_plot$plot_sp
```



Plot mean-centred latent trends

Latent trends (mean-centred) obtained from DFA are displayed as continuous lines (with 95% CI as shaded grey). This is the output of interest for latent trends (i.e. 1st tool of the toolbox).

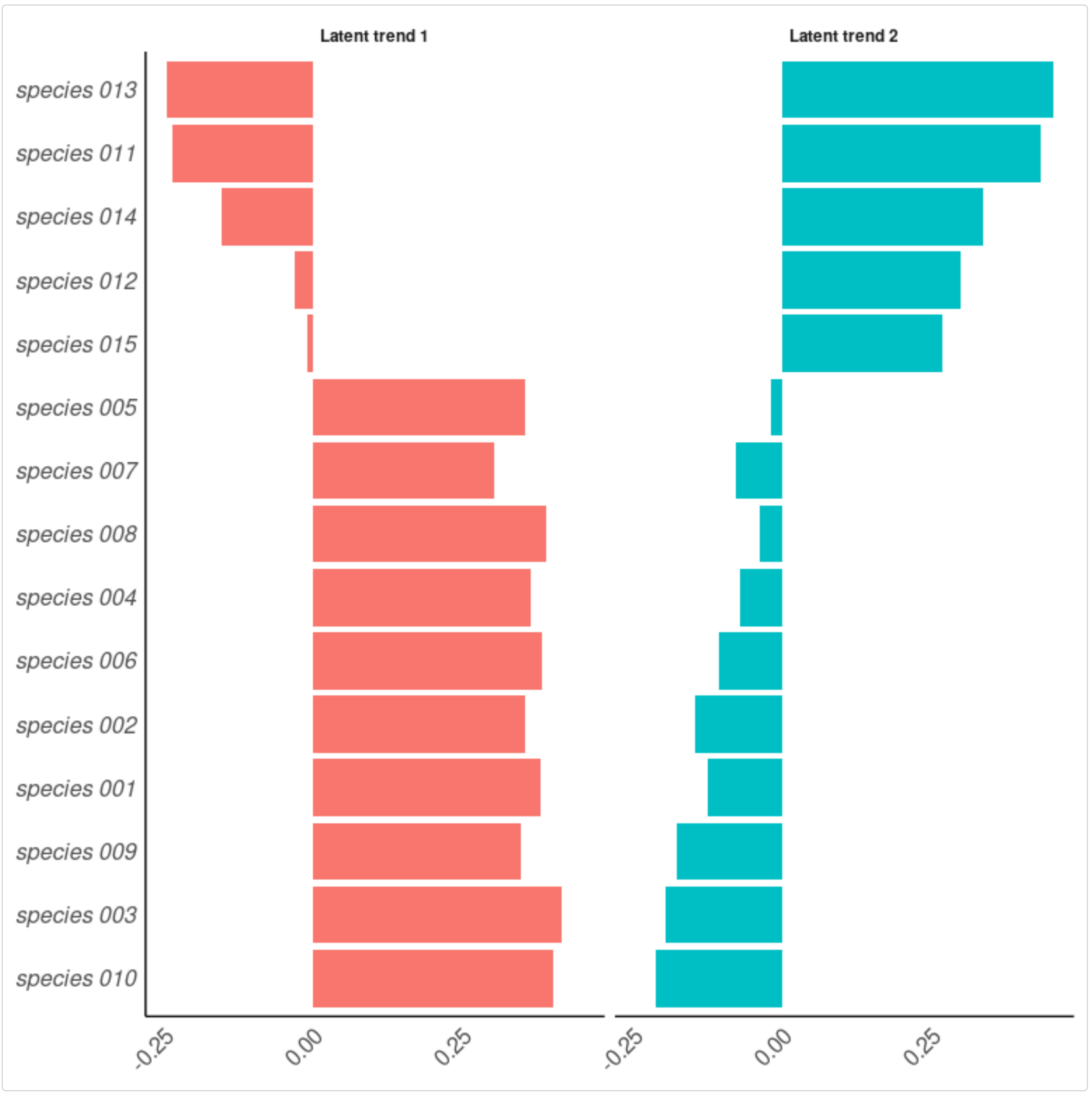
```
dfa_result_plot$plot_tr
```



Plot factor loadings

Factor loadings are shown for each species and each latent trend. Here, species 011 to 015 on the one hand and species 001 to 010 on the other hand share similar factor loadings.

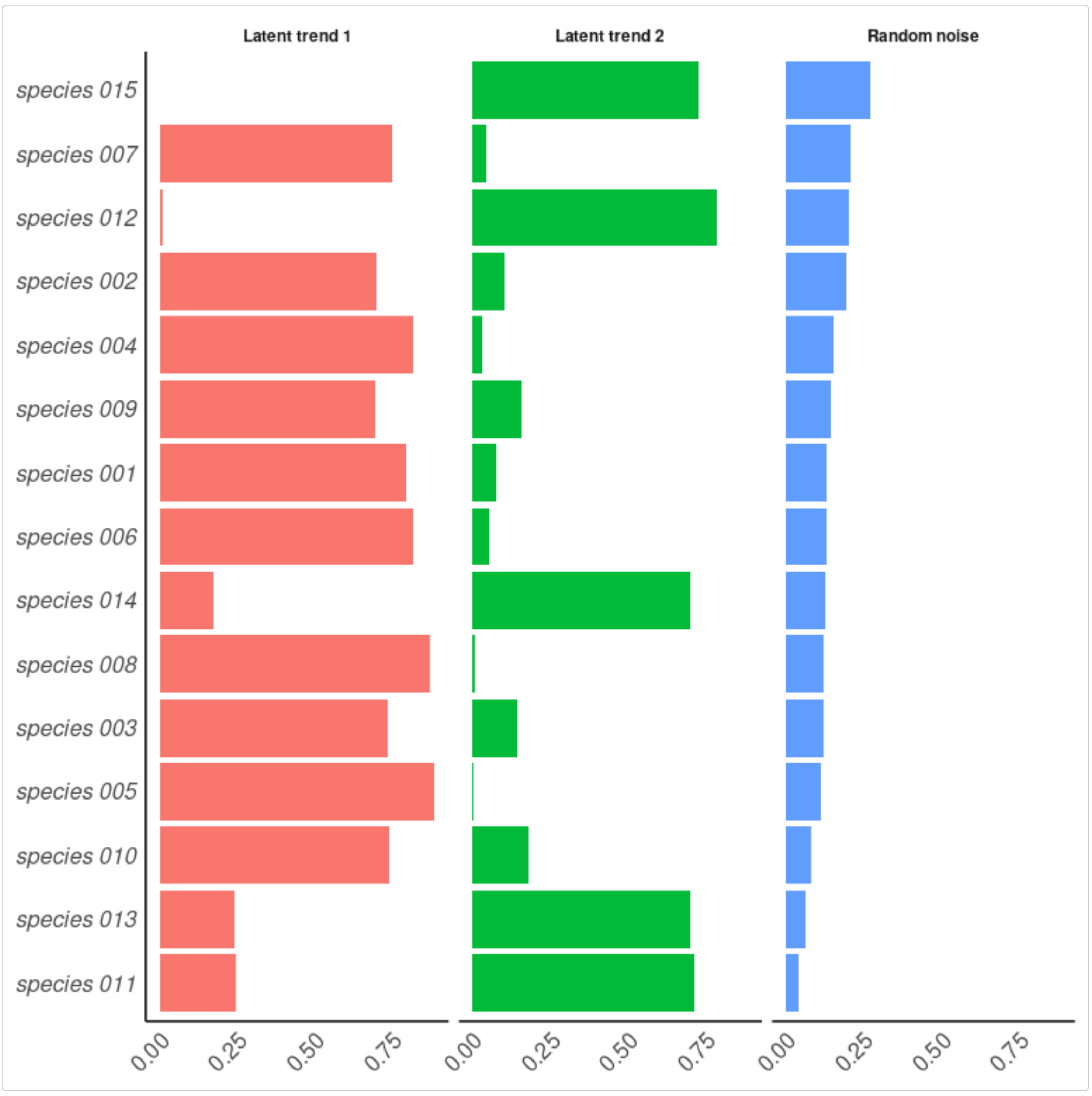
```
dfa_result_plot$plot_ld
```



Plot percentage of variance explained by latent trends

The percentage of variance in species time-series captured by each latent trend and random noise are shown.

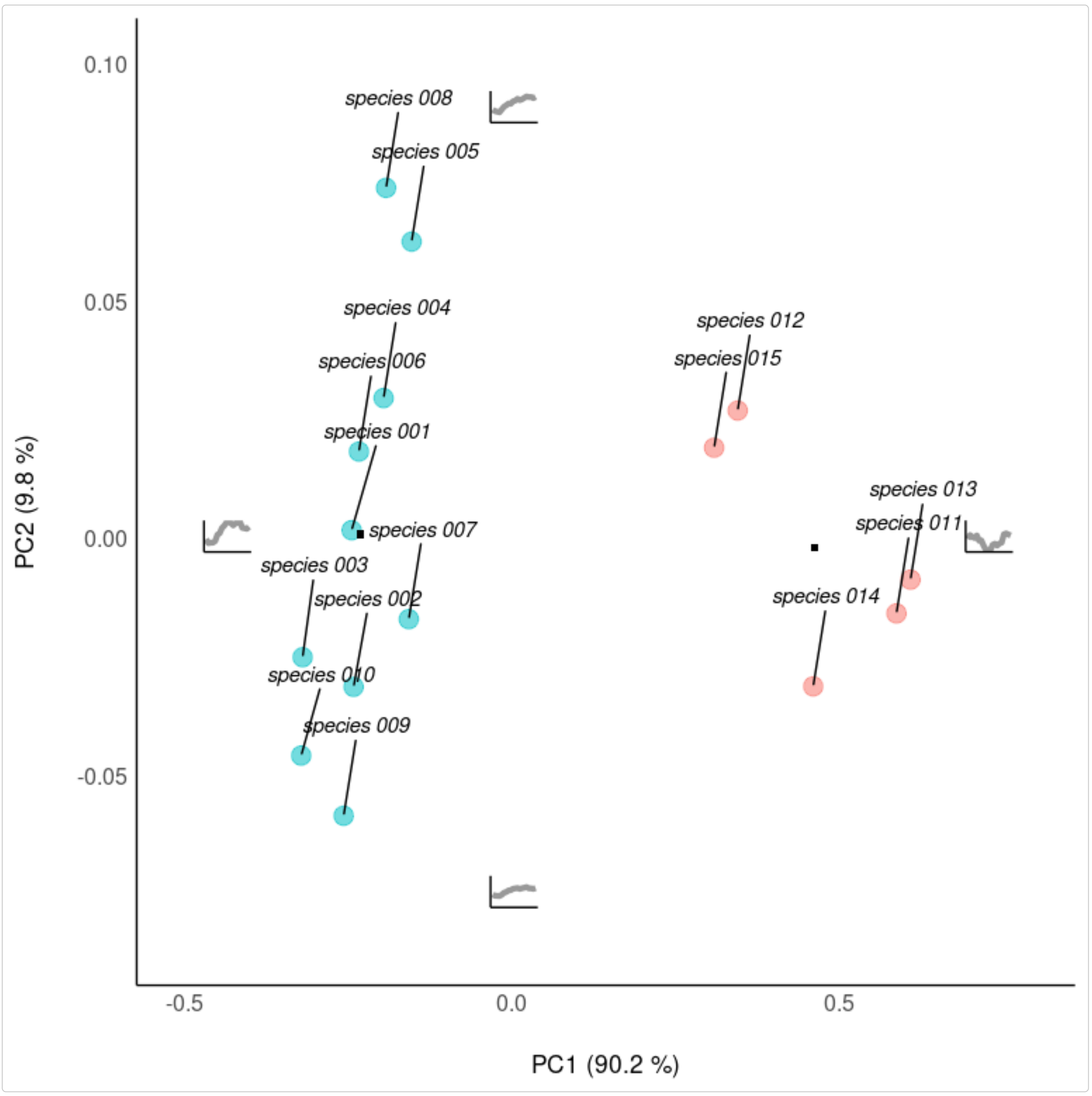
```
dfa_result_plot$plot_perc_var
```



Plot species clusters on first factorial plan

The ordinal plot of the first factorial plan shows the two clusters (that were expected by construction). This is the output of interest for cluster visualisation (i.e. 2nd and 3rd tools of the toolbox).

```
dfa_result_plot$plot_sp_group[[1]]
```



Plot species clusters on second factorial plan

```
dfa_result_plot$plot_sp_group[[2]]
#> [1] NA
```

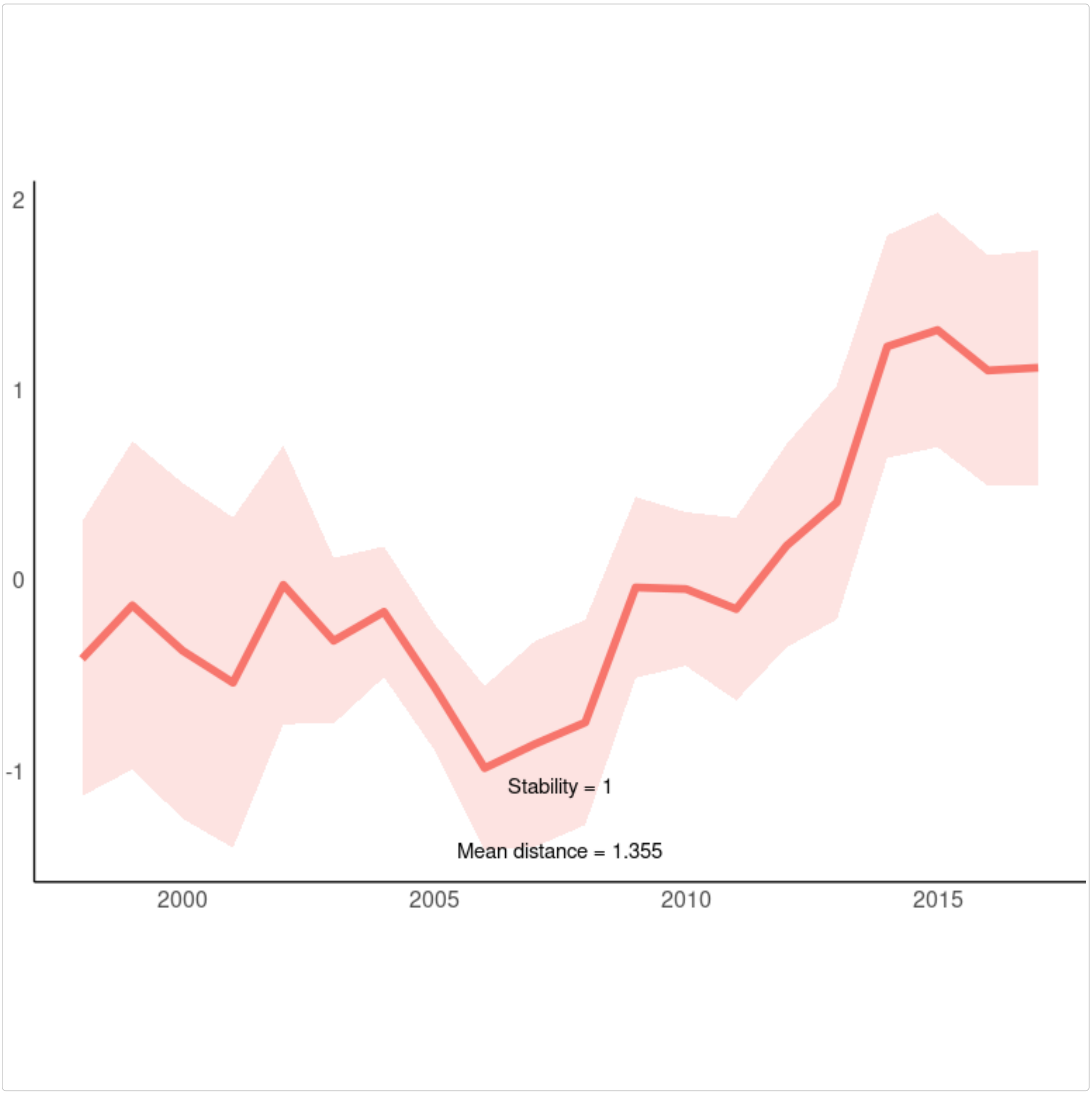
Plot species clusters on third factorial plan

```
dfa_result_plot$plot_sp_group[[3]]
#> [1] NA
```

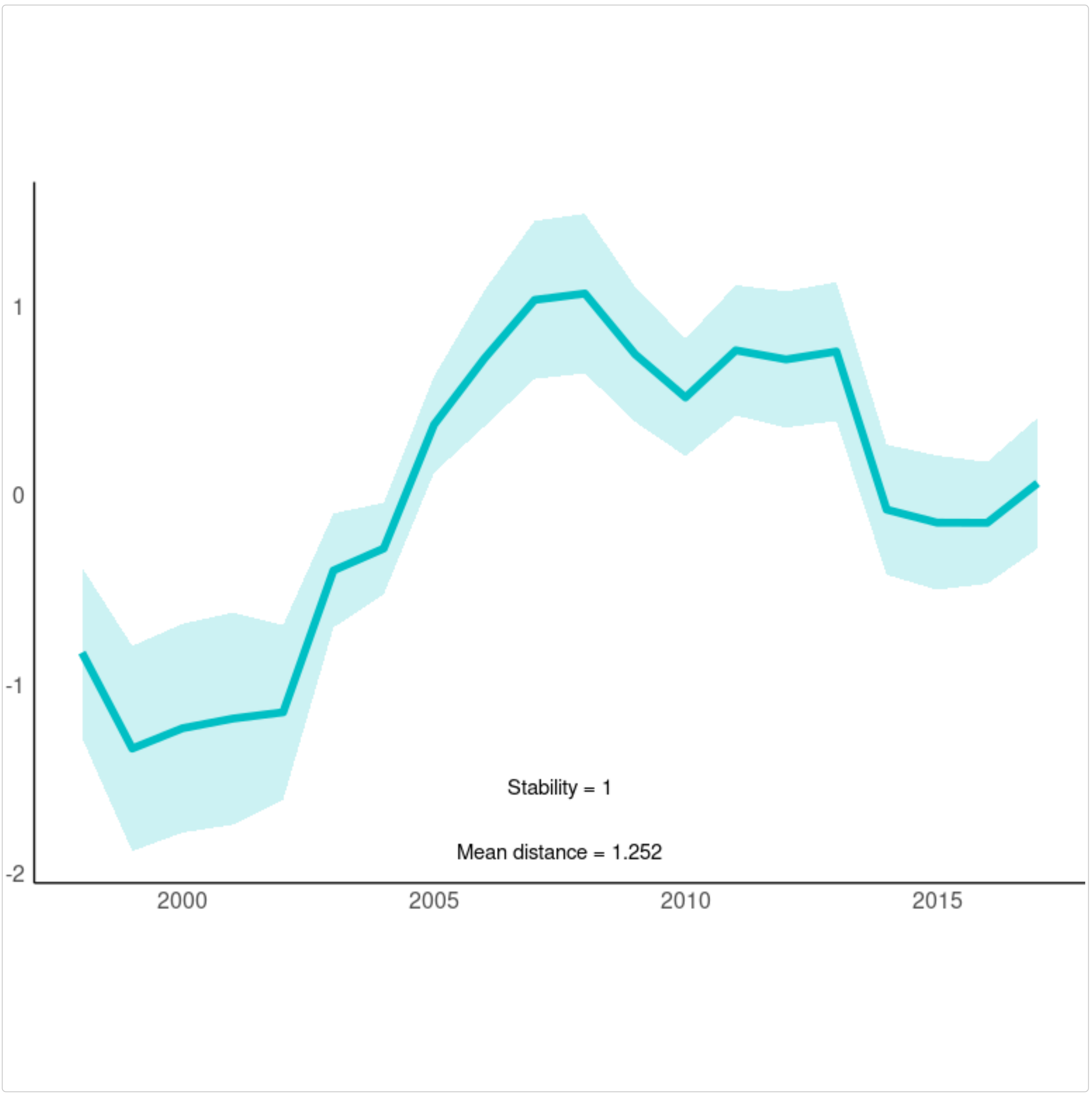
Plot time-series of cluster centres

These are time-series of a virtual species that would be at the exact centre of the cluster in the factorial space. This is therefore an average of time-series of species that belong to the cluster.

```
dfa_result_plot$plot_group_ts$g1
```



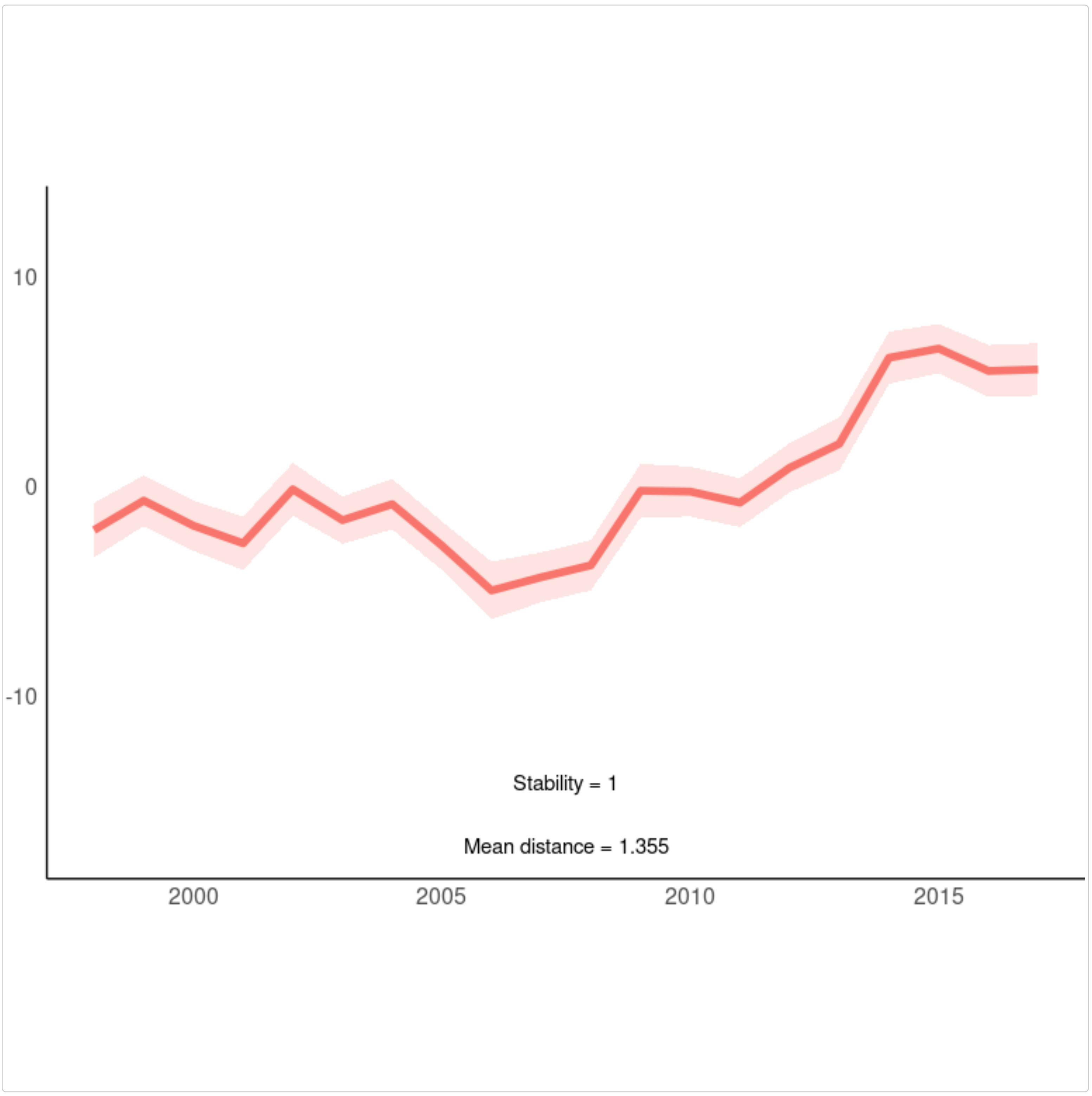
```
dfa_result_plot$plot_group_ts$g2
```



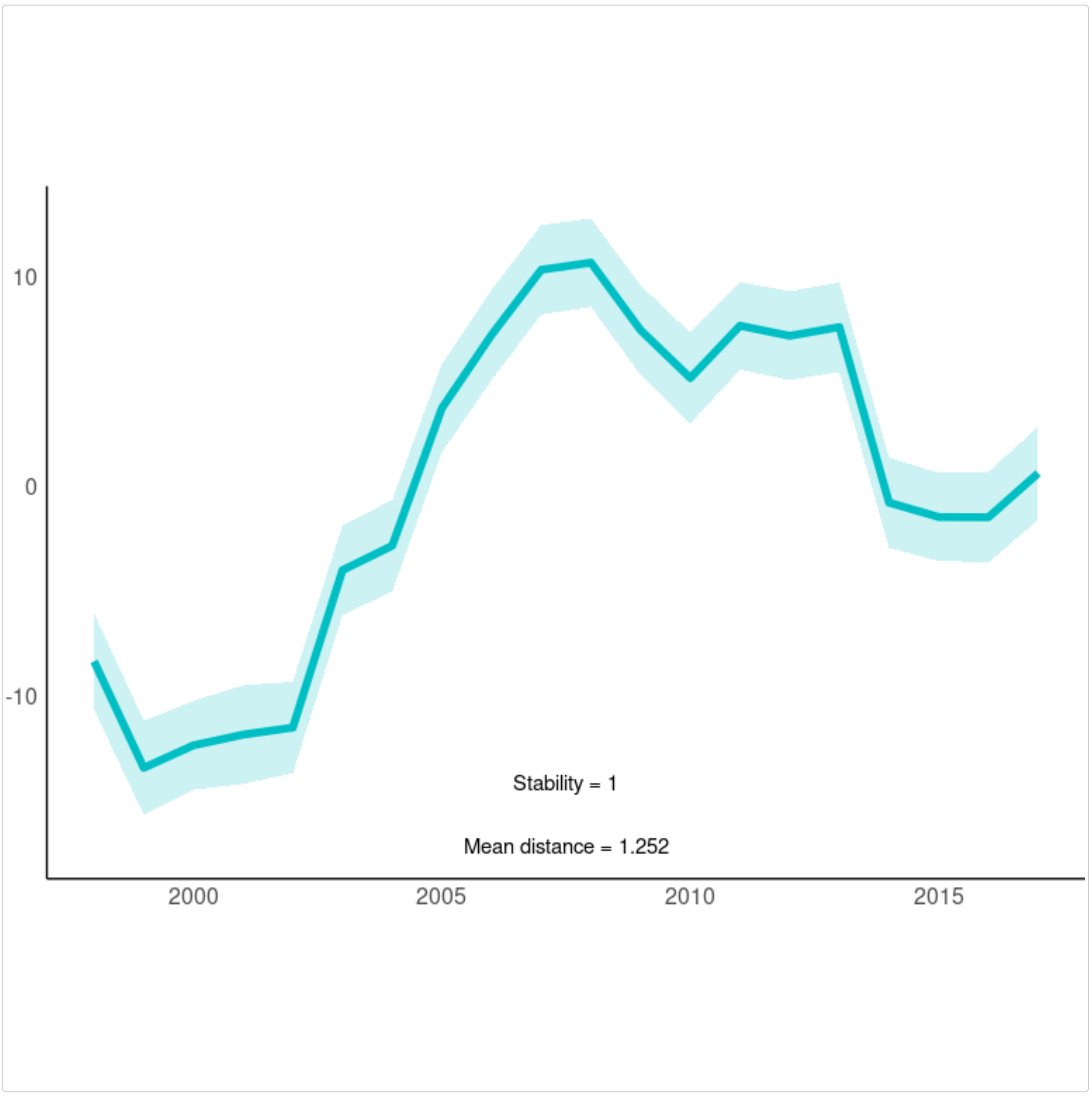
Plot time-series of cluster centres from sdRep

Similar as the previous plot but the 95% CI are narrower as the standard error was estimated during the optimisation phase. This is the output of interest for cluster centres.

```
dfa_result_plot$plot_group_ts2$g1
```



```
dfa_result_plot$plot_group_ts2$g2
```



Detailed information on DFA

This is the summary of the optimisation output of DFA. Row names correpsonds to the output of the cpp model: `log_re_sp` log of standard error for random effect by species, `re_sp` random effect by species, `x` latent trends, `x_mc` mean-centred latent trends, `x_pred` cluster centres, `x_pred2` weighted cluster centres, `x_sp` fitted species time-series, `Z` matrix of factor loadings of species, `Z_pred` matrix of factor loadings of cluster centres.

```
head(dfa_result_plot$sdRep)
#>              Estimate Std. Error
#> log_re_sp -0.9420075  0.1686078
#> log_re_sp -0.7491587  0.1672591
#> log_re_sp -0.8524115  0.1723132
#> log_re_sp -0.9184570  0.1748564
#> log_re_sp -1.1492955  0.1806140
#> log_re_sp -0.9497807  0.1696515
```

Main results of species clustering

The columns are set as follows:

- `code_sp`: code for species names
- `PC1`: coordinate on PCA first axis
- `PC2`: coordinate on PCA second axis
- `group`: cluster id
- `Xn`: rotated loading factors for each latent trend *n*
- `uncert`: species stability into its cluster
- `name_long`: species names

```
head(dfa_result_plot$group$kmeans_res[[1]])
#>   code_sp        PC1         PC2 group        X1          X2 uncert   name_long
#> 1   SP001 -0.2444518  0.00179774     2 0.4007416  0.00000000      1 species 001
#> 2   SP002 -0.2414481 -0.03122342     2 0.3840024 -0.02862203      1 species 002
#> 3   SP003 -0.3194116 -0.02498027     2 0.4572411 -0.05606992      1 species 003
#> 4   SP004 -0.1956230  0.02966840     2 0.3683649  0.04596495      1 species 004
#> 5   SP005 -0.1527768  0.06266422     2 0.3435808  0.09403024      1 species 005
#> 6   SP006 -0.2333477  0.01838843     2 0.3977317  0.01973561      1 species 006
```

Detail information on clustering

Cluster barycentres

The columns are set as follows:

- `group`: group number
- `X1`: factor loading on latent trend 1
- `X2`: factor loading on latent trend 2
- `PC1`: coordinate on PCA first axis
- `PC2`: coordinate on PCA second axis

```
dfa_result_plot$group$kmeans_res[[2]]
#>                      group         X1          X2        PC1           PC2
#> kmeans_center_row       1 -0.2411838 0.296987964  0.4628365 -0.0018475349
#> kmeans_center_row.1     2  0.3885699 0.004742292 -0.2314182  0.0009237674
```

Variance captured by PCA first two axes

```
dfa_result_plot$group$kmeans_res[[3]]
#> [1] 0.90159078 0.09840922
```

Cluster position in the first factorial plan

- `group`: group number
- `PC1`: coordinate on PCA first axis
- `PC2`: coordinate on PCA second axis

```
dfa_result_plot$group$centroids
#>                      group        PC1           PC2
#> kmeans_center_row       1  0.4628365 -0.0018475349
#> kmeans_center_row.1     2 -0.2314182  0.0009237674
```

Cluster stability

Stability max is 1 and indicates that species are always attributed to this cluster in all simulation. Stability is always above 0.5, any cluster with a lower stability being disolved during the clustering analysis.

```
dfa_result_plot$group$stability_cluster_final
#> [1] 1 1
```

Cluster dispersion

Cluster dispersion corresponds to the average distance between species and cluster centre. This allows the user to compare whether the different clusters are composed of species with very similar or fairly heterogenous time-series.

```
dfa_result_plot$group$mean_dist_clust
#>           mean_dist
#> cluster_1  1.355101
#> cluster_2  1.251968
```

Data for time-series of cluster centres

It corresponds to the data used to produce the plots of cluster centre time-series shown above.

```
head(dfa_result_plot$trend_group2)
#>           group year   Estimate Std..Error
#> x_pred2     all 1998 -10.382179  1.3142601
#> x_pred2.1    g1 1998  -2.065151  0.6579215
#> x_pred2.2    g2 1998  -8.317029  1.1643761
#> x_pred2.3   all 1999 -14.051041  1.2405315
#> x_pred2.4    g1 1999  -0.660125  0.6164866
#> x_pred2.5    g2 1999 -13.390916  1.1455201
```