vignette-DFAclust

Load R package

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

Reproducible time series

Let simulate some data. Here we will create time-series for 15 species with 20 time-steps and simulate time-series of observation errors for each species as well. We start by simulating 4 latent trends, then we create sets of factor loadings from 2 simulated cluster centres which we will built and finally using simulated latent trends and factor loading we obtain simulated time-series. Normally you will 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)
    id_vec <- c(id_vec,rep(g,nb_sp_g))

    for(lt in 1:n_sp_init){

        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

```
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),0,0.01)

    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

Data format and completeness can be check using the function `prepare_data`. This is not a mandatory step, but it is highly recommended before using the function `fit_dfa` to run the DFA analysis. It checks for missing values and zeros in the species time-series and observation error time-series. `perc_replace` correspond to the proportion of the average index value used to replace zeros in species time-series with `0.01` as default value (1 %). It can also transform the observation error time-series to get their log values if they are initially not in log values (in that 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 th enumber 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 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$species_name_ordre,species_sub = species_name_ex, nfac = 0, mintrend =
        1, maxtrend = 3, AIC = TRUE,center_option = 1, silent = TRUE, control = list())
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      0      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 213.4966 213.4966 213.4966 213.4966 213.4966 217.5346
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 7.925812e-05 4.242093e-05 6.494198e-05 3.034626e-05 5.232726e-05 1.967196e-05
#> AIC:  495.069180007586
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      1      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 28.51441 28.51441 28.51441 28.51441 28.51441 28.51441
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 2.187195e-04 5.645280e-05 1.661806e-04 4.446748e-05 1.360617e-04 6.060426e-05
#> Warning in core_dfa(data_ts = data_ts, data_ts_se = data_ts_se, nfac = i, :
#> Convergence issue:singular convergence (7)
#> AIC:  145.028815975003
#> Warning in fit_dfa(data_ts = data_ready_dfa$data_ts, data_ts_se =
#> data_ready_dfa$data_ts_se, : Convergence issue:singular convergence (7)
#> NLMINB   BFGS NLMINB   BFGS NLMINB   BFGS
#>      0      0      0      0      0      0
#>   NLMINB     BFGS   NLMINB     BFGS   NLMINB     BFGS
#> 23.45465 23.45465 23.45465 23.45465 23.45465 23.45465
#>       NLMINB         BFGS       NLMINB         BFGS       NLMINB         BFGS
#> 2.517525e-04 2.924885e-05 7.633662e-05 3.457630e-05 9.987615e-05 4.301544e-05
#> AIC:  160.909301626534
```

Run the clustering analysis

Once the latent trend are estimated, the clustering analysis can be run using the function `cluster_result` and the number of iteration `nboot` can be specified.

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

Plot results of DFA and clustering

Finally, the result can be plot 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 observation error 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
- `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  0.6150898 0.010278773 0.6150898 0.010278773 -0.8154946
#> 2   SP001 2006  3.0320684 0.076317575 3.0320684 0.076317575  0.8034784
#> 3   SP001 1999  0.3249538 0.038767161 0.3249538 0.038767161 -1.3804611
#> 4   SP001 2007  4.1893323 0.016452360 4.1893323 0.016452360  1.1143172
#> 5   SP001 2000  0.3033011 0.005380504 0.3033011 0.005380504 -1.3685443
#> 6   SP001 2001  0.3617251 0.137705956 0.3617251 0.137705956 -1.2454470
#>     pred_se   name_long pred.value_exp pred_se.value_exp se.value_exp   value_1
#> 1 0.1094888 species 001      0.4424205        0.04844010  0.006322369 1.0000000
#> 2 0.1254860 species 001      2.2332958        0.28024735  0.231400109 4.9294726
#> 3 0.1352990 species 001      0.2514626        0.03402264  0.012597537 0.5283030
#> 4 0.1333950 species 001      3.0474867        0.40651943  0.068924401 6.8109276
#> 5 0.1369571 species 001      0.2544771        0.03485246  0.001631913 0.4931005
#> 6 0.1334913 species 001      0.2878122        0.03842042  0.049811699 0.5880850
#>   pred.value_exp_1 se.value_exp_1 pred_se.value_exp_1
#> 1        1.0000000    0.010278773          0.10948884
#> 2        5.0479035    0.376205393          0.63344120
#> 3        0.5683792    0.020480809          0.07690115
#> 4        6.8882138    0.112055830          0.91885316
#> 5        0.5751930    0.002653129          0.07877678
#> 6        0.6505400    0.080982804          0.08684142
```

DFA latent trends

The columns are set as follows:

- `Year`: year
- `variable`: latent trend id
- `value`: latent trend values
- `se.value`: standard error of latent trends
- `rot_tr.value`: 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 2.095265 0.4650894
#> 2 Latent trend 1 1999  1.4515298 0.3985068  1.3622629 3.546843 0.6549423
#> 3 Latent trend 1 2000  1.4208242 0.4216589  1.3083817 3.516225 0.6766713
#> 4 Latent trend 1 2001  1.1045782 0.4039735  0.9909613 3.199949 0.6427041
#> 5 Latent trend 1 2002  1.3944914 0.3907760  1.3110451 3.489871 0.6343044
#> 6 Latent trend 1 2003 -0.3145916 0.3595322 -0.3929336 1.780765 0.4283698
```

DFA loading factors

The columns are set as follows:

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

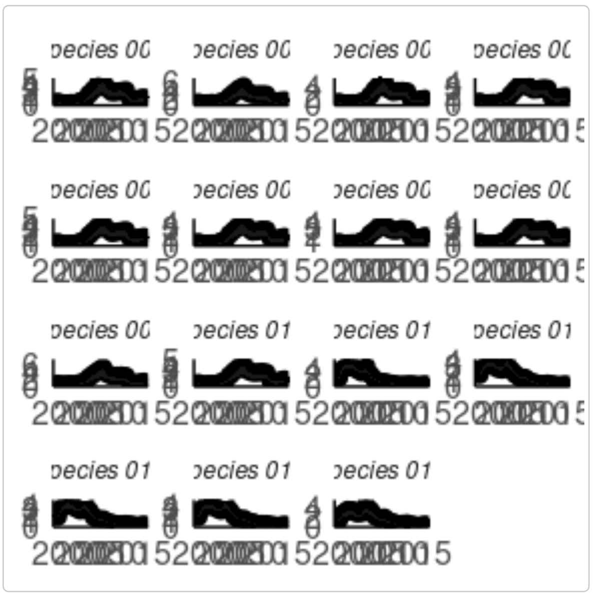
```
head(dfa_result_plot$data_loadings)
#>   code_sp       variable       value se.value   name_long        PC1
#> 1   SP001 Latent trend 1 -0.38864892       NA species 001 -0.1931130
#> 2   SP001 Latent trend 2 -0.02086085       NA species 001 -0.1931130
#> 3   SP002 Latent trend 1 -0.39396362       NA species 002 -0.1676149
#> 4   SP002 Latent trend 2  0.02313504       NA species 002 -0.1676149
#> 5   SP003 Latent trend 1 -0.41560759       NA species 003 -0.2076764
#> 6   SP003 Latent trend 2 -0.01269357       NA species 003 -0.2076764
```

- `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

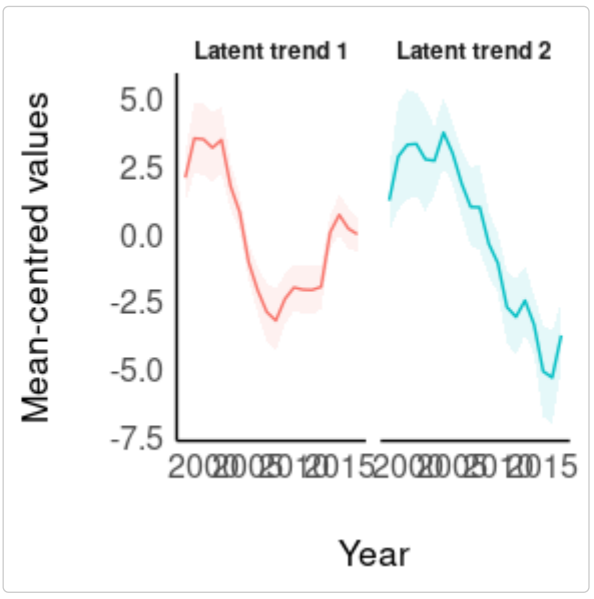
```
head(dfa_result_plot$exp_var_lt)
#>     name_long Latent trend 1 Latent trend 2 Random noise       all
#> 1 species 001      0.8974139    0.004945318   0.09764083 0.8490128
#> 2 species 002      0.8905433    0.005874025   0.10358268 0.8791223
#> 3 species 003      0.9269871    0.001653967   0.07135888 0.9399079
#> 4 species 004      0.8735296    0.025423515   0.10104686 0.7826477
#> 5 species 005      0.8540357    0.041045565   0.10491874 0.7510299
#> 6 species 006      0.8561437    0.016747063   0.12710921 0.7328955
```

Plots

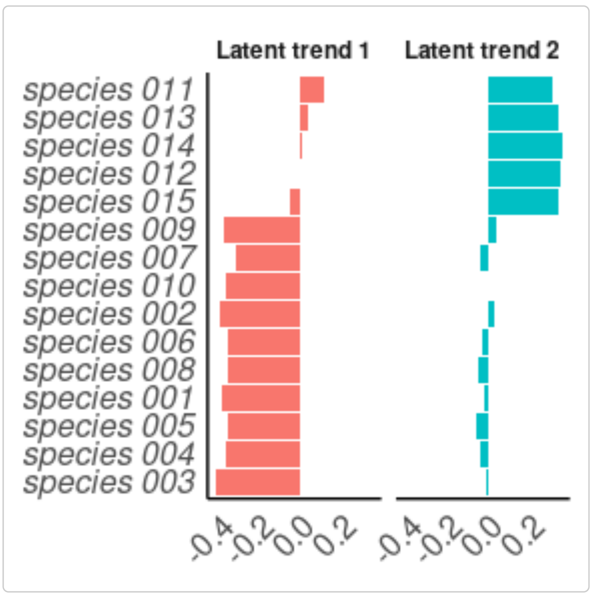
```
# Plot species time-series

dfa_result_plot$plot_sp
```



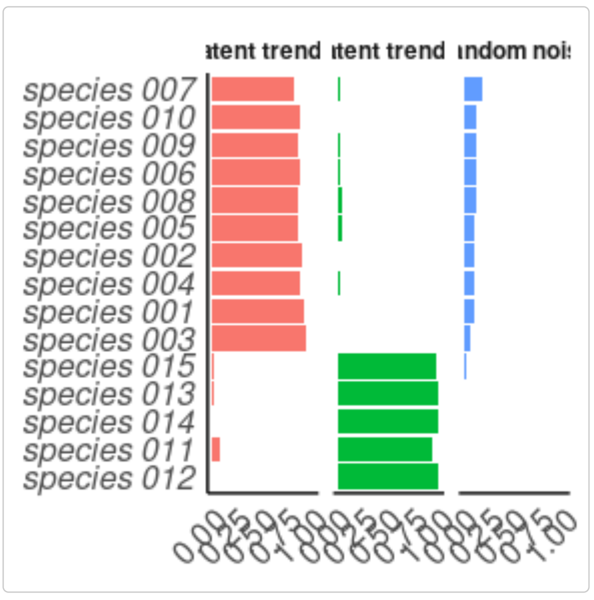
```
# Plot mean-centred latent trends

dfa_result_plot$plot_tr
```



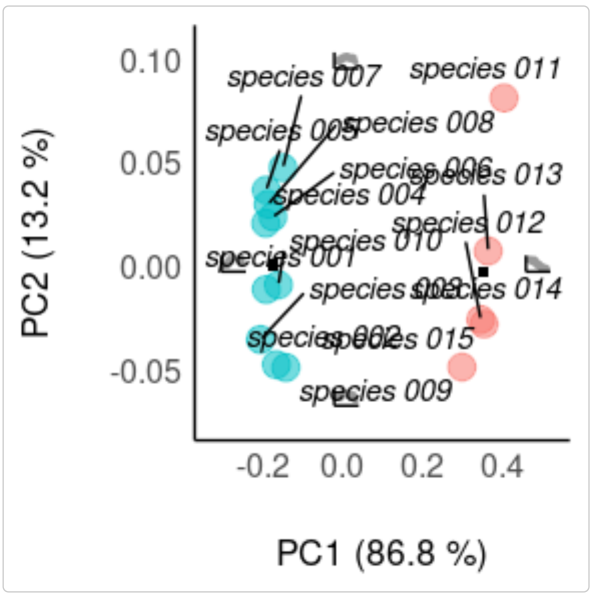
```r
# Plot loading factors
dfa_result_plot$plot_ld
```



```r
# Plot percentage of variance explained by latent trends
dfa_result_plot$plot_perc_var
```

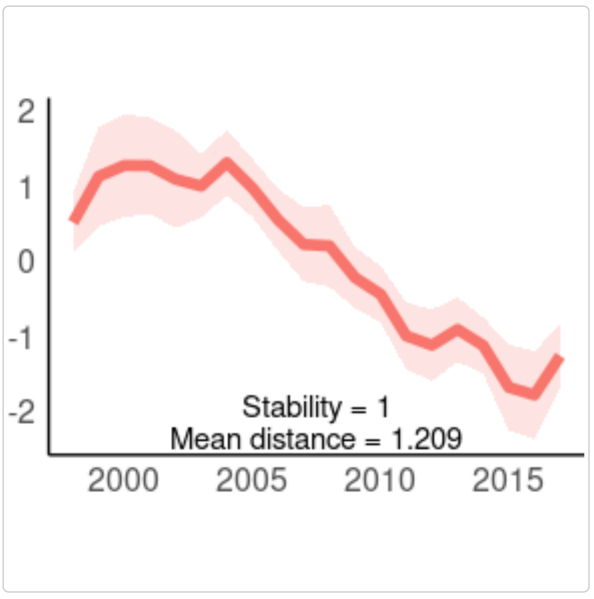


```r
# Plot species clusters on first factorial plan
dfa_result_plot$plot_sp_group[[1]]
```

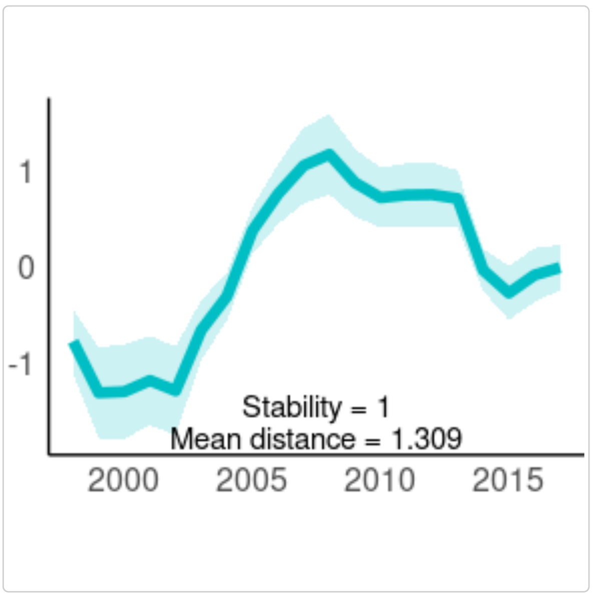


```r
# Plot species clusters on second factorial plan
dfa_result_plot$plot_sp_group[[2]]
#> [1] NA

# Plot species clusters on third factorial plan
dfa_result_plot$plot_sp_group[[3]]
#> [1] NA

# Plot time-series of cluster centres
```

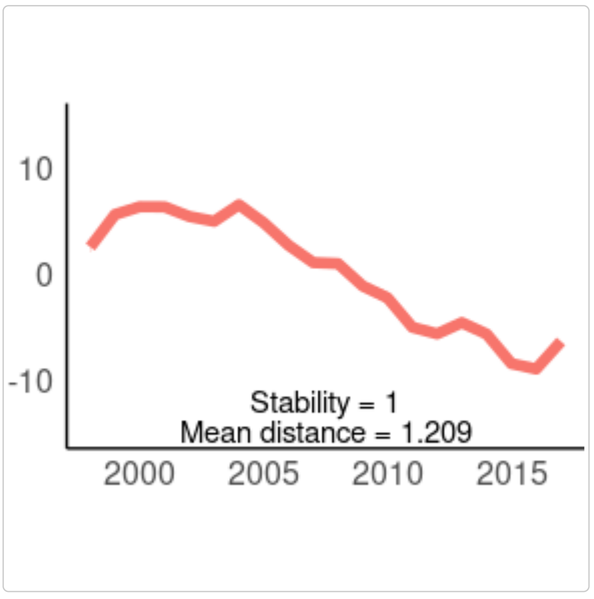
```
dfa_result_plot$plot_group_ts$g1
```



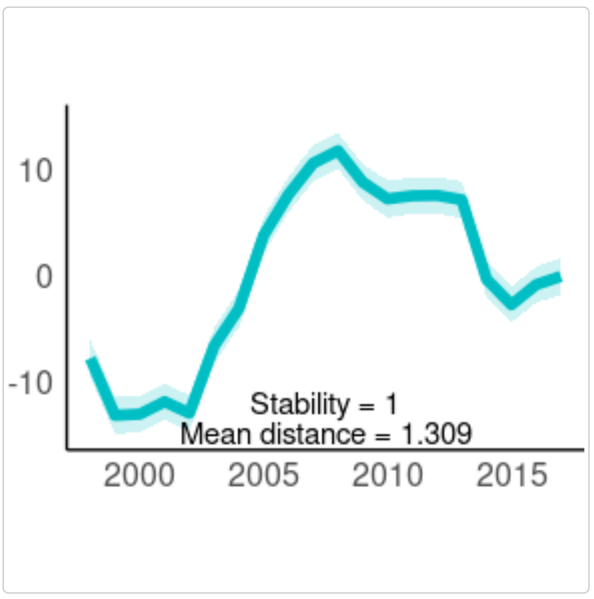
```
dfa_result_plot$plot_group_ts$g2
```



```
# Plot time-series of cluster centres from sdRep
dfa_result_plot$plot_group_ts2$g1
```



```
dfa_result_plot$plot_group_ts2$g2
```



Detailed information on DFA

```
# Summary of otimisation output of DFA

head(dfa_result_plot$sdRep)
#>            Estimate Std. Error
#> log_re_sp -1.245070  0.1664727
#> log_re_sp -1.198108  0.1744001
#> log_re_sp -1.351003  0.1767900
#> log_re_sp -1.268622  0.1735835
#> log_re_sp -1.270439  0.1850602
#> log_re_sp -1.186730  0.1690873
```

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
#> 1   SP001 -0.1931130 -0.01075859     2 -0.3892084  0.000000000      1
#> 2   SP002 -0.1676149 -0.04700400     2 -0.3921573  0.044217509      1
#> 3   SP003 -0.2076764 -0.03487045     2 -0.4156905  0.009600473      1
#> 4   SP004 -0.1943163  0.02120262     2 -0.3700558 -0.025615354      1
#> 5   SP005 -0.1931649  0.03719573     2 -0.3591082 -0.037330981      1
#> 6   SP006 -0.1763093  0.02430706     2 -0.3540989 -0.016712098      1
#>     name_long
#> 1 species 001
#> 2 species 002
#> 3 species 003
#> 4 species 004
#> 5 species 005
#> 6 species 006
```

Detail information on clustering

```
# Cluster barycentres

dfa_result_plot$group$kmeans_res[[2]]
#>                       group          X1           X2      PC1          PC2
#> kmeans_center_row         1  0.04280296 0.3382192105  0.35548 -0.002291884
#> kmeans_center_row.1       2 -0.36976910 0.0004031131 -0.17774  0.001145942

# Variance captured by PCA first two axes

dfa_result_plot$group$kmeans_res[[3]]
#> [1] 0.8682984 0.1317016

# Cluster position in the first factorial plan
```

```
dfa_result_plot$group$centroids
#>                       group      PC1          PC2
#> kmeans_center_row       1  0.35548 -0.002291884
#> kmeans_center_row.1     2 -0.17774  0.001145942

# Cluster stability

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

# Cluster dispersion

dfa_result_plot$group$mean_dist_clust
#>           mean_dist
#> cluster_1  1.208530
#> cluster_2  1.309337

# Cluster position in PCA

dfa_result_plot$group$pca_centre_list
#> [[1]]
#>          [,1]        [,2]        [,3]       [,4]
#> X1 -0.2626465 -0.18094820 -0.39377359 0.08398552
#> X2  0.1506297  0.04952946 -0.01752139 0.36855183
#>
#> [[2]]
#> [1] NA
#>
#> [[3]]
#> [1] NA

# PCA results

dfa_result_plot$group$myPCA
#> NULL

# Time-series of cluster barycentres

head(dfa_result_plot$trend_group2)
#>           group year   Estimate Std..Error
#> x_pred2     all 1998  -5.176600  0.9309405
#> x_pred2.1    g1 1998   2.565994  0.2485824
#> x_pred2.2    g2 1998  -7.742594  0.8918557
#> x_pred2.3   all 1999  -7.476007  0.9068120
#> x_pred2.4    g1 1999   5.627518  0.1860488
#> x_pred2.5    g2 1999 -13.103525  0.8868077
```

Empirical data

The following empirical example corresponds to the empirical analysis in the paper related to this R package. Preparation of the data from the raw data are not in the scope of the package, this is why required packages have to be loaded first.

Load additionnal R packages

```
require(stringr)
require(rnaturalearth)
require(sp)
require(ggplot2)
require(reshape2)
require(plyr)
require(dplyr)
require(emmeans)
require(arm)
require(see)
```

Load and prepare data

Bird data

Download and extract data

Breeding bird data for Sweden can be download from https://www.gbif.org/occurrence

/download?dataset_key=91fa1a0d-a208-40aa-8a6e-f2c0beb9b253 (an free account is necessary).Type of download: Darwin Core Archive. This will provide you a text file: “occurrence.txt”.

```
bird_se_raw <- read.csv("raw_data/occurrence.txt", header = T, sep="\t")
```

Subselecting and preparing data

We select columns and row of interest (i.e. birds).

```
bird_se_clean <- bird_se_raw[bird_se_raw$class=="Aves",c("class","order","family",
        "genus","species",

        "specificEpithet","infraspecificEpithet","taxonRank",

        "organismQuantity","decimalLatitude","decimalLongitude",

        "day","month","year","taxonKey","speciesKey","countryCode",

        "level1Gid","level2Gid","iucnRedListCategory")]
```

We add a code by species from species name and check for duplicate in species’ code. The link between species name and species name code is stored in a specific dataset `species_data`.

```
species_data <- data.frame(name_long = unique(bird_se_clean$species[bird_se_clean$taxonRank !=
        "GENUS"]))

species_data$code_sp <- paste0(toupper(substr(species_data$name_long, 1, 3)),
                               toupper(substr(sub(".* ", "", species_data$name_long), 1, 3)))

species_data$code_sp[species_data$name_long=="Corvus corax"] <- "CORCOX"

species_data$code_sp[species_data$name_long=="Phylloscopus trochilus"] <- "PHYTRU"

species_data$code_sp[species_data$name_long=="Saxicola rubicola"] <- "SAXRUI"

species_data$code_sp[species_data$name_long=="Sterna paradisaea"] <- "STEPAD"

species_data$genus <- sub(" .*", "", species_data$name_long)

species_data$species <- sub(".* ", "", species_data$name_long)

bird_se_clean <- merge(bird_se_clean, species_data[,c("code_sp", "name_long")],
                       by.x = c("species"), by.y = c("name_long"), all.x = T)

species_data$class <- bird_se_clean$class[match(species_data$code_sp, bird_se_clean$code_sp)]

species_data$order <- bird_se_clean$order[match(species_data$code_sp, bird_se_clean$code_sp)]

species_data$family <- bird_se_clean$family[match(species_data$code_sp, bird_se_clean$code_sp)]

species_data$iucnRedListCategory <-
        bird_se_clean$iucnRedListCategory[match(species_data$code_sp, bird_se_clean$code_sp)]
```

Geographical coordinates and routes

We prepare a specific dataset for coordinates of monitored routes by linking route numbers to their coordinates.

```
route_data <- paste0(bird_se_clean$decimalLatitude, sep="_", bird_se_clean$decimalLongitude)

route_data <- data.frame(code_route = paste0("R",str_pad(1:length(unique(route_data)), 3, pad =
        "0")),
                         coordinate_chr = unique(route_data))

route_data$lat <- as.numeric(sub("_.*", "", route_data$coordinate_chr))

route_data$lon <- as.numeric(sub(".*_", "", route_data$coordinate_chr))
```

We can then display route location on the map.

```
worldmap <- ne_countries(scale = 'medium', type = 'countries',returnclass = 'sf')

sweden_map_wgs84 <- worldmap[worldmap$sovereign=="Sweden",]
```

```
sweden_map_moll <- sf::st_transform(sweden_map_wgs84, "+proj=moll")

sweden_map_swe <- sf::st_transform(sweden_map_wgs84, "+init=epsg:3006")

route_data_coord <- route_data

coordinates(route_data_coord) <- ~lon+lat

proj4string(route_data_coord) <- CRS("+proj=longlat +datum=WGS84")

route_data_coord <- spTransform(route_data_coord, CRSobj = "+proj=moll")

route_data_moll <- as.data.frame(coordinates(route_data_coord))

route_data_moll$code_route <- route_data_coord$code_route

route_data_coord <- spTransform(route_data_coord, CRSobj = "+init=epsg:3006")

route_data_swe <- as.data.frame(coordinates(route_data_coord))

route_data_swe$code_route <- route_data_coord$code_route

ggplot() + geom_sf(data=sweden_map_swe) +
  geom_tile(data = route_data_swe, aes(x=lon, y=lat), width=25000, height=25000, alpha=0.5) +
  theme_void() + coord_sf(datum=NA)
```

We aggregate all the projections into on dataset.

```
route_data <- merge(route_data, route_data_moll, by="code_route", all=T)

route_data <- merge(route_data, route_data_swe, by="code_route", all=T)

names(route_data)[3:8] <- c("lat_wgs", "lon_wgs", "lon_moll", "lat_moll", "lon_swe", "lat_swe")

route_data$coordinate_chr <- NULL
```

We finally associate routes and coordinates with the main dataset on bird occurrence.

```
bird_se_clean <- merge(bird_se_clean, route_data[,c("code_route", "lat_wgs", "lon_wgs")],
                       by.x = c("decimalLatitude", "decimalLongitude"), by.y = c("lat_wgs",
        "lon_wgs"), all.x = T)

route_data$level1Gid <- bird_se_clean$level1Gid[match(route_data$code_route,
        bird_se_clean$code_route)]

route_data$level2Gid <- bird_se_clean$level2Gid[match(route_data$code_route,
        bird_se_clean$code_route)]
```

Finalise bird dataset

We need to incorporate 0s in the dataset when, a given year, a species was not present while the route was monitored.

```
bird_se_clean_tot <- dcast(bird_se_clean, countryCode+code_route+year~code_sp,
                           fun.aggregate = sum, value.var="organismQuantity")

bird_se <- melt(bird_se_clean_tot, id.vars = c("countryCode", "code_route", "year"))

names(bird_se)[4:5] <- c("code_sp", "abund")

bird_se$code_sp <- as.character(bird_se$code_sp)

bird_se <- bird_se[bird_se$code_sp!="NA",]
```

We add information on geographical coordinate and taxa in the final dataset `bird_se`.

```
bird_se <- merge(bird_se, route_data, by="code_route", all.x = T)

bird_se$order <- species_data$order[match(bird_se$code_sp, species_data$code_sp)]

bird_se$family <- species_data$family[match(bird_se$code_sp, species_data$code_sp)]

bird_se$genus <- species_data$genus[match(bird_se$code_sp, species_data$code_sp)]
```

```r
bird_se$species <- species_data$species[match(bird_se$code_sp, species_data$code_sp)]

bird_se$name_long <- species_data$name_long[match(bird_se$code_sp, species_data$code_sp)]

bird_se$iucnRedListCategory <- species_data$iucnRedListCategory[match(bird_se$code_sp,
        species_data$code_sp)]
```

Estimate species time-series

Now that the data are ready, we can estimate species time-series and standard errors of these time-series.

Select period of time

We first remove data from the first two years of the survey (1996 and 1997) because there was a low number of routes monitored in 1996 and 1997.

```r
bird_se_1998 <- droplevels(bird_se[bird_se$year>1997,])
```

Compute species time series

As we said above about the additional packages that needed to be loaded, it is not in the scope of this R package to obtain time-series from raw data and the user will normally have usable time-series. So we need to specify here an additional function adapted from the French Breeding Bird Survey analysis https://www.vigienature.fr/sites/vigienature/files/atoms/files/analysestoceps_0.zip) to estimate time-series from bird abundance data.

```r
get_ts <- function(data_bird_input){

  # d: data for species i

  d <- droplevels(data_bird_input)

  sp <- levels(as.factor(d$code_sp))

  # number of route followed by year

  nb_route <- tapply(rep(1,nrow(d)),d$year,sum)

  # number of route with species i by year

  nb_route_presence <- tapply(ifelse(d$abund>0,1,0),d$year,sum)

  year <- as.numeric(as.character(levels(as.factor(d$year))))

  firstY <- min(year)

  lastY <- max(year)

  timestep <- length(year)-1

  # table for analysis result

  threshold_occurrence <- 3

  tab_ana <- data.frame(year=rep(year,2),val=c(nb_route,nb_route_presence),LL = NA,UL=NA,
                        catPoint=NA,pval=NA,
                        curve=rep(c("route","presence"),each=length(year)))

  tab_ana$catPoint <- ifelse(tab_ana$val == 0,"0", ifelse(tab_ana$val < threshold_occurrence,
                                                          "inf_threshold",NA))


  # abundance by year

  abund <- tapply(d$abund,d$year,sum)

  threshold_abundance <- 5

  tab_fig <- data.frame(year=year,val=abund,LL = NA,UL=NA,catPoint=NA,pval=NA)

  tab_fig$catPoint <- ifelse(tab_fig$val == 0,"0",ifelse(tab_fig$val < threshold_abundance,
                                                         "inf_threshold",NA))

  # remove criteria
```

```r
remove_sp <- FALSE

# if first year empty

if(tab_fig$val[1]==0){remove_sp <- TRUE}

# if four consecutive years empty

ab_vec <- paste(tab_fig$val,collapse="")

if(str_detect(ab_vec, "0000")){remove_sp <- TRUE}

# if less than three consecutive years

ab_vec2 <- paste(sign(tab_fig$val),collapse="")

if(!str_detect(ab_vec2, "111")){remove_sp <- TRUE}

if(anyNA(tab_fig$catPoint) & anyNA(tab_ana$catPoint[tab_ana$curve=="presence"]) &
      remove_sp==F){

  # GLM abundance variation
  glm1 <- glm(abund~as.factor(code_route)+as.factor(year),data=d,family=quasipoisson)

  sglm1 <- summary(glm1)

  # mean-centered values

  con.mat <- diag(length(year)) - 1/length(year)

  colnames(con.mat) <- year # firstY:lastY

  rg <- ref_grid(glm1, nuisance = 'code_route')

  sglm2 <- summary(contrast(rg, as.data.frame(con.mat)))

  # as link function is log, estimates need to be back transformed from sglm1 (first year set
      to 1 and se to 0)

  coef_yr <- tail(matrix(sglm1$coefficients[,1]), timestep)

  coef_yr <- rbind(1, exp(coef_yr))

  error_yr <- tail(matrix(sglm1$coefficients[,2]), timestep)

  error_yr <- rbind(0, error_yr)*coef_yr # approximated se values

  log_error_yr <- tail(matrix(sglm1$coefficients[,2]), timestep)

  log_error_yr <- rbind(0, log_error_yr)

  pval <- c(1,tail(matrix(coefficients(sglm1)[,4]),timestep))

  # from sglm2 (mean value to 0)

  coef_yr_m0 <- exp(sglm2$estimate)

  error_yr_m0 <- sglm2$SE*coef_yr_m0 # approximated se values

  log_error_yr_m0 <- sglm2$SE

  pval_m0 <- sglm2$p.value

  # CIs

  glm1.sim <- sim(glm1)

  ci_inf_sim <- c(1, exp(tail(apply(coef(glm1.sim),2, quantile, .025), timestep)))

  ci_sup_sim <- c(1, exp(tail(apply(coef(glm1.sim),2, quantile, .975), timestep)))

  thresold_signif <- 0.05

  tab_res <- data.frame(year,val=coef_yr,val_m0=coef_yr_m0,
                        LL=ci_inf_sim,UL=ci_sup_sim,
                        catPoint=ifelse(pval<thresold_signif,"significatif",NA),pval)

  # cleaning out of range CIs
```

```
    tab_res$UL <- ifelse(nb_route_presence==0,NA,tab_res$UL)

    tab_res$UL <-  ifelse(tab_res$UL == Inf, NA,tab_res$UL)

    tab_res$UL <-  ifelse(tab_res$UL > 1.000000e+20, NA,tab_res$UL)

    tab_res$UL[1] <- 1

    tab_res$val <-  ifelse(tab_res$val > 1.000000e+20,1.000000e+20,tab_res$val)

    tab_res$val_m0 <-  ifelse(tab_res$val_m0 > 1.000000e+20,1.000000e+20,tab_res$val_m0)

    # overdispersion index
    dispAn <- sglm1$deviance/sglm1$null.deviance

    # class uncertainity

    if(dispAn > 2 | (median(nb_route_presence)<threshold_occurrence & median(abund)
        <threshold_abundance)) catIncert <- "Uncertain" else catIncert <-"Good"

    vecLib <-  NULL

    if(dispAn > 2 | median(nb_route_presence)<threshold_occurrence) {

      if(median(nb_route_presence)<threshold_occurrence) {

            vecLib <- c(vecLib,"too rare species")

          }

      if(dispAn > 2) {

         vecLib <- c(vecLib,"deviance")

       }
    }

    reason_uncert <-  paste(vecLib,collapse=" and ")

    # Store results

    tab_tot <- data.frame(code_sp=sp, year = tab_res$year, nb_year=timestep,
                          firstY = firstY, lastY = lastY,
                          relative_abundance = tab_res$val,
                          CI_inf = tab_res$LL, CI_sup = tab_res$UL,
                          Standard_error = error_yr,
                          Log_SE = log_error_yr,
                          p_value = tab_res$pval,
                          relative_abundance_m0 = tab_res$val_m0,
                          Standard_error_m0 = error_yr_m0,
                          Log_SE_m0 = log_error_yr_m0,
                          p_value_m0 = pval_m0, signif = !is.na(tab_res$catPoint),
                          nb_route,nb_route_presence,abundance=abund,
                          mediane_occurrence=median(nb_route_presence),
        mediane_ab=median(abund) ,
                          valid = catIncert, uncertanity_reason = reason_uncert)

  }
  else{
    tab_tot <- data.frame(code_sp=sp, year = year, nb_year=timestep,
                          firstY=firstY, lastY=lastY,
                          relative_abundance=NA,
                          CI_inf = NA, CI_sup = NA,
                          Standard_error = NA,
                          p_value = NA,
                          relative_abundance_m0 = NA,
                          Standard_error_m0 = NA,
                          Log_SE_m0 = NA,
                          p_value_m0 = NA,signif = NA,
                          nb_route,nb_route_presence,abundance=abund,
                          mediane_occurrence=median(nb_route_presence),
        mediane_ab=median(abund) ,
                          valid = NA, uncertanity_reason = NA)
  }

  return(tab_tot)
}
```

We can now compute time-series between 1998 and 2020 for each species in the Swedish Breeding Bird

Survey.

```
ts_bird_se_allcountry <- ddply(bird_se_1998, .(code_sp), .fun=get_ts, .progress="text")
```

We can then check, by plotting the estimated time-series, if they are similar to the one produced by the official institution in charge in Sweden Svensk fageltaxering (https://www.fageltaxering.lu.se/resultat/trender).

```
for(i in 1:length(levels(as.factor(bird_se$code_sp)))){

  sp <- levels(as.factor(ts_bird_se_allcountry$code_sp))[i]

  gp <- ggplot(ts_bird_se_allcountry[ts_bird_se_allcountry$code_sp==sp,],
         aes(year, relative_abundance)) + geom_line() + geom_text(x=2010, y=1, label=sp) +
    geom_line(aes(y=CI_inf), linetype="dashed") +
    geom_line(aes(y=CI_sup), linetype="dashed") +
    ylab("Relative abundance") + xlab("Years") +
    theme_modern()

  print(gp)

}
```

DFA cluster analysis for Swedish birds

Now that we have time-series and standard errors for each species, we can launch the analysis.

Farmland birds

We will first run the analysis on farmland birds. We therefore select the 15 farmland species.

```
species_sub <- species_farm <-  droplevels(species_data[species_data$code_sp %in% c(
  "FALTIN","VANVAN","ALAARV","HIRRUS","CORFRU",
  "SAXRUB","SYLCOM","ANTPRA","MOTFLA","LANCOL",
  "STUVUL","LINCAN","EMBCIT","EMBHOR","PASMON"),])
```

We then subselect the corresponding time-series and standard errors for farmland birds.

```
Obs <- ts_bird_se_allcountry[ts_bird_se_allcountry$code_sp %in% species_sub$code_sp,]
```

We finally produce a dataset of species time-series `y_farm_ts` and a dataset of standard errors `obs_se_farm_ts`.

```
y_farm <- dcast(Obs[,c("code_sp","relative_abundance_m0","year")],
           code_sp~year, fun.aggregate = sum, value.var = "relative_abundance_m0")

obs_se_farm <- dcast(Obs[,c("code_sp","Log_SE_m0","year")],
             code_sp~year, fun.aggregate = sum, value.var = "Log_SE_m0")

y_farm_ts <- as.matrix(y_farm[,2:ncol(y_farm)]) # species time series

y_uncert_ts <- as.matrix(obs_se_farm[,2:ncol(obs_se_farm)]) # standard error on time series

rownames(y_farm_ts) <- rownames(y_uncert_ts) <- y_farm$code_sp # add species names as row names
```

We can now run the `prepare_data` function.

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

Then the `fit_dfa` to launch the DFA analysis.

```
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_farm,
         nfac = 0, mintrend = 1, maxtrend = 5, AIC = TRUE, center_option = 1, silent = TRUE,
         control = list())
```

Then the `cluster_dfa` to launch the clustering analysis.

```r
cluster_result <- cluster_dfa(data_dfa = dfa_result, species_sub = species_farm, nboot = 500)
```

And finally the `plot_dfa_result` to get the result of the whole analysis.

```r
dfa_result_plot_farm <- plot_dfa_result(data_dfa = dfa_result, sdRep = cluster_result$sdRep,
        species_sub = species_farm, group_dfa = cluster_result$group_dfa, min_year =
        data_ready_dfa$min_year, species_name_ordre = data_ready_dfa$species_name_ordre)
```

Woodland birds

We will now run the analysis on woodland birds. We therefore select the 26 woodland species.

```r
species_sub <- species_forest <- droplevels(species_data[species_data$code_sp %in% c(
   "ACCNIS","TETBON","TRIOCH","COLOEN","DRYMAR",
  "DRYMIN","NUCCAR","GARGLA","PERATE","LOPCRI","POEPAL","POEMON",
  "SITEUR","CERFAM","TURVIS","PHOPHO","PHYCOL","PHYSIB","REGREG","FICHYP","FICALB",
  "ANTTRI","COCCOC","SPISPI","PYRPYR","EMBRUS"),])
```

We then subselect the corresponding time-series and standard errors for farmland birds.

```r
Obs <- ts_bird_se_allcountry[ts_bird_se_allcountry$code_sp %in% species_sub$code_sp,]
```

We finally produce a dataset of species time-series `y_farm` and a dataset of standard errors `obs_se_farm`.

```r
y_forest <- dcast(Obs[,c("code_sp","relative_abundance_m0","year")],
          code_sp~year, fun.aggregate = sum, value.var = "relative_abundance_m0")

obs_se_forest <- dcast(Obs[,c("code_sp","Log_SE_m0","year")],
               code_sp~year, fun.aggregate = sum, value.var = "Log_SE_m0")

y_forest_ts <- as.matrix(y_forest[,2:ncol(y_forest)]) # species time series

y_uncert_ts <- as.matrix(obs_se_forest[,2:ncol(obs_se_forest)]) # standard error on time series

rownames(y_forest_ts) <- rownames(y_uncert_ts) <- y_forest$code_sp # add species names as row
        names
```

We can now run the `prepare_data` function.

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

Then the `fit_dfa` to launch the DFA analysis.

```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_forest,
        nfac = 0, mintrend = 1, maxtrend = 5, AIC = TRUE, center_option = 1, silent = TRUE,
        control = list())
```

Then the `cluster_dfa` to launch the clustering analysis.

```r
cluster_result <- cluster_dfa(data_dfa = dfa_result, species_sub = species_forest, nboot = 500)
```

And finally the `plot_dfa_result` to get the result of the whole analysis.

```r
dfa_result_plot_forest <- plot_dfa_result(data_dfa = dfa_result, sdRep = cluster_result$sdRep,
        species_sub = species_forest, group_dfa = cluster_result$group_dfa, min_year =
        data_ready_dfa$min_year, species_name_ordre = data_ready_dfa$species_name_ordre)
```

Display results

We now can look at the main result for farmland and woodland birds.

```r
dfa_result_plot_farm$plot_sp_group[[1]]
```

```
dfa_result_plot_forest$plot_sp_group[[1]]
```