ALPERC: R implementation

Luca Pegoraro

08 aprile, 2022

ALPERC()

Description

ALPERC is a model-agnostic Active Learning (AL) algorithm for noisy physical experiments with multiple responses based on nonparametric ranking and clustering.

For further information on the algorithm, please refer to the article "R. Arboretti, R. Ceccato, L. Pegoraro, L. Salmaso, Active learning for noisy physical experiments with multiple responses."

The functions have been implemented in the following environment, using the following package versions:

```
#load the package
library(ALPERC)
```

```
## [1] "R version 4.1.1 (2021-08-10)"
```

```
##       ALPERC       gplots RColorBrewer       plotly   factoextra      usedist        purrr       forca
##      "0.1.0"      "3.1.1"      "1.1-2"    "4.9.4.1"      "1.0.7"      "0.4.0"      "0.3.4"      "0.5
##        dplyr      ggplot2        akima
##      "1.0.7"      "3.3.5"    "0.6-2.2"
```

Usage

ALPERC(n_add, D_cand, sigma_cand, S, strategy, varimp_distance, n_clust, n_boot, alpha_rank, seed_rank, paral)

Arguments

Argument	Description
n_add	The size of the batch to add at the AL iteration. An integer.
D_cand	A data frame including the candidate experimental configurations for AL. The first column is a factor column that contains the identifier (ID) of the candidates, the following columns include the factor levels. The column headers (except from the first) are the names of the factors.
sigma_cand	A data frame including the predictive uncertainty of the candidate points in D_cand. The first column is a factor column that contains the identifier (ID) of the candidates, the following columns include the uncertainty of predictions of the candidate configurations for each response (>2). The column headers (except from the first) are the names of the responses.
S	A data frame including the variable importance of each factor. The first column is a character column that contains the names of the factors (equal to the column headers in D_cand), the following columns include the variable importance of the factors corresponding to each response (>2). The column headers (except from the first) are the names of the responses. If variable importance is not available, set S=NULL.

Argument	Description
strategy	It is either “exploration” (only unique candidates) or “exploitation” (can include replicates).
varimp__distance	It is either TRUE or FALSE. In the first case, the assignment of each candidate configuration to a cluster is adjusted by the importance of each dimension. If varimp__distance=FALSE, S is not used.
n__clust	The number of clusters, it can either be NULL or an integer. If n__clust=NULL, the best number of clusters is chosen with the Silhouette method, otherwise it is set equal to the number indicated.
n__boot	The number of bootstrap iterations for the permutations used for the nonparametric ranking procedure.
alpha__rank	The significance level for the computation of the nonparametric ranking procedure.
seed__rank	A seed that is used to initialize the permutations.
paral	Either TRUE or FALSE. It indicates whether parallel computation should be employed (default is FALSE). If paral=TRUE, a cluster must be initialized with “parallel” and “doParallel” libraries.

Outputs

The function provides a list with 8 elements:

Element	Description
$strategy	Is the strategy chosen for ALPERC implementation.
$seed__rank	Is the seed chosen to initialize the nonparametric ranking procedure.
$varimp__distance	Indicates whether the variable importance is used or not.
$clust__choice	A tibble that includes the number of clusters and corresponding Silhouette index.
$nclust__criterion	Indicates the strategy used for the selection of the number of clusters.
$best__nclust	The number of clusters considered.
$LAMBDA	A tibble that includes the ID of each candidate, the factor levels combination, the position in the rank and the cluster to which each configuration is assigned.
$D__add__j	A tibble that includes the n__add configurations from D__add that should be added according to ALPERC.

Example

Here we show an example of the application of ALPERC function. First, we load the input data frames:

```
D_cand
```

```
## # A tibble: 50 x 7
##    rowid     A     B     C     D     E     F
##    <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 1       0.6   0.2   0     0.2   0.2   0
##  2 2       0.2   0.2   0.6   1     0.6   0.4
##  3 3       1     0.4   0.8   1     0.4   0.6
##  4 4       0     0.6   1     0.6   0.6   0.2
##  5 5       0.4   0.2   0     0.4   0.8   0.2
##  6 6       0.4   0     0.4   0.8   0     0.6
##  7 7       0     0.8   0     0     0.4   0.8
##  8 8       0     0.2   0.4   0.6   0.4   0
##  9 9       0.4   0.6   0.2   0.8   1     0
## 10 10      0     0.8   0.2   1     0.8   0.8
## # ... with 40 more rows
```

```
sigma_cand
```

```
## # A tibble: 50 x 8
##    rowid X6.d.Borehole X6.d.OTL.circuit X6.d.Piston X6.d.Piston.Mod X6.d.Robot.arm X6.d.Rosenbrock.Fu
##    <fct>         <dbl>            <dbl>       <dbl>           <dbl>          <dbl>
##  1 1            0.0245           0.0327      0.0837          0.0836         0.148
##  2 2            0.0185           0.0440      0.0718          0.0589         0.0834
##  3 3            0.0582           0.0177      0.0549          0.0563         0.0869
##  4 4            0.0175           0.0989      0.0676          0.0626         0.100
##  5 5            0.0218           0.0342      0.0696          0.0679         0.110
##  6 6            0.0353           0.0411      0.0928          0.0715         0.0894
##  7 7            0.0345           0.0968      0.0486          0.0627         0.137
##  8 8            0.0132           0.0642      0.0900          0.0717         0.0877
##  9 9            0.0268           0.0753      0.0459          0.0498         0.0926
## 10 10           0.0289           0.0423      0.0378          0.0398         0.107
## # ... with 40 more rows
```

```
S
```

```
## # A tibble: 6 x 8
##   X     X6.d.Borehole X6.d.OTL.circuit X6.d.Piston X6.d.Piston.Mod X6.d.Robot.arm X6.d.Rosenbrock.Fun
##   <chr>         <dbl>            <dbl>       <dbl>           <dbl>          <dbl>
## 1 A           0.585            0.529       0.0239          0.0656         0.0110                   0
## 2 B           0.00204          0.360       0.606           0.483          0.431                    0
## 3 C           0.00187          0.0851      0.358           0.366          0.428                    0
## 4 D           0.0409           0.0247      0.0202          0.162          0.149                    0
## 5 E           0.0304           0.00387     0.00248         0.00128        0.231                    0
## 6 F           0.478            0.00178     0.00507         0.00300        0.176                    0
```

Then, we use the function:

```
##Here we run the function with parallel computation. This step can be avoided by setting the argument p
library(parallel)
library(doParallel)
```

```
##initialize cluster
cl <- makePSOCKcluster(4)
registerDoParallel(cl)

results<-ALPERC(n_add=5,
                D_cand=D_cand,
                sigma_cand=sigma_cand,
                S=S,
                strategy="exploration",
                varimp_distance=T,
                n_clust=NULL,
                n_boot=500,
                alpha_rank=0.1,
                seed_rank=2105,
                paral=TRUE)
```

```
## [1] "Start NPC ranking"
## [1] "2022-04-08 14:27:20 PDT"
## Time difference of 12.99025 secs
## [1] "End NPC ranking"
```

```
stopCluster(cl)
```

Finally, we print the results:

```
results
```

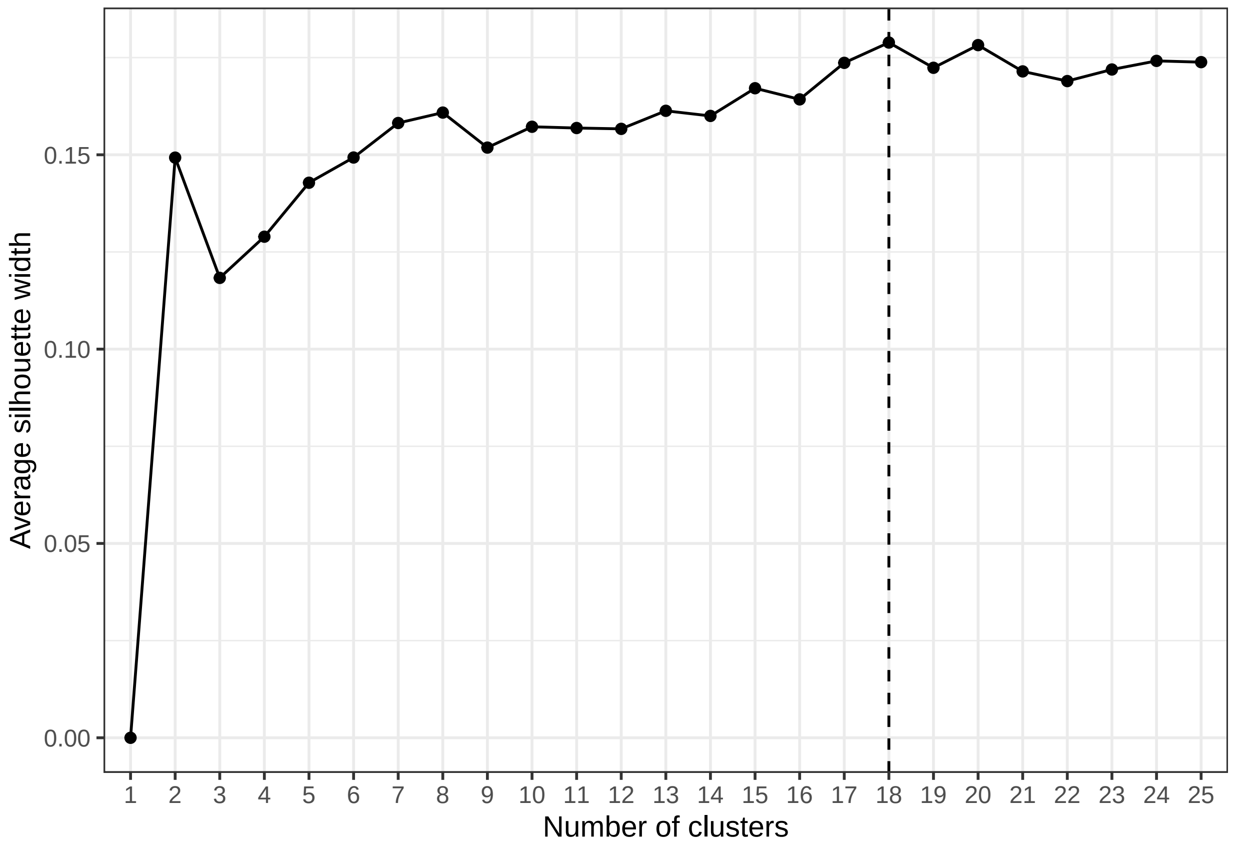
```
## $strategy
## [1] "exploration"
##
## $seed_rank
## [1] 2105
##
## $varimp_distance
## [1] TRUE
##
## $clust_choice
## # A tibble: 25 x 2
##    clusters     y
##    <fct>    <dbl>
##  1 1        0
##  2 2        0.149
##  3 3        0.118
##  4 4        0.129
##  5 5        0.143
##  6 6        0.149
##  7 7        0.158
##  8 8        0.161
##  9 9        0.152
## 10 10       0.157
## # ... with 15 more rows
##
## $nclust_criterion
## [1] "silhouette"
##
## $best_nclust
## [1] 18
##
## $LAMBDA
## # A tibble: 50 x 9
##    rowid     A     B     C     D     E     F  rank cluster
##    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>   <int>
##  1 26      0.8   0.8   1     0     1     0.2    50      10
##  2 25      0     0.2   1     0.2   0.8   0.6    49       9
##  3 37      0.6   0.4   1     0     0.6   0.6    48      15
##  4 38      0.6   0     0     0.2   0.4   1      47      16
##  5 7       0     0.8   0     0     0.4   0.8    46       1
##  6 1       0.6   0.2   0     0.2   0.2   0      30       1
##  7 4       0     0.6   1     0.6   0.6   0.2    30       2
##  8 9       0.4   0.6   0.2   0.8   1     0      30       1
##  9 11      0.8   0     1     0.2   0.2   0.6    30       4
## 10 15      1     0.4   1     0.8   1     0.6    30       1
## # ... with 40 more rows
##
## $D_add_j
## # A tibble: 5 x 7
##   rowid     A     B     C     D     E     F
##   <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 26      0.8   0.8     1   0     1     0.2
## 2 25      0     0.2     1   0.2   0.8   0.6
## 3 37      0.6   0.4     1   0     0.6   0.6
## 4 38      0.6   0       0   0.2   0.4   1
## 5 7       0     0.8     0   0     0.4   0.8
```

silhouette_plot()

A function that gets as input an "ALPERC" element (list) and provides a Silhouette plot, indicating the best number of clusters.

Example

```
silhouette_plot(results)
```



ALPERC_2D_viz()

A function that produces a 2D visualization of D_add_j and LAMBDA, together with the mean response (or uncertainty) as predicted by the model. It currently works only for hetGP and ranger models.

Usage

ALPERC_2D_viz(D_add_j=D_add_j, LAMBDA=LAMBDA, model=model, viz_factors=viz_factors, fixed_factors=fixed_factors, fixed_factor_levels=fixed_factor_levels, what_plot=what_plot)

Arguments

Argument	Description
D_add_j	A tibble, that is the output of ALPERC() function.
LAMBDA	A tibble, that is the output of ALPERC() function.
model	A hetGP or ranger (trained via caret) model.
viz_factors	A character vector that includes the names of 2 factors that will be used in the plot.
fixed_factors	A character vector that includes the names of the factors that will be fixed. They are all the input variables except from those included in viz_factors.
fixed_factor_levels	A numeric vector that includes the chosen levels of the fixed_factors. The order must correspond to the order in fixed_factors
what_plot	A character vector that is either "mean" or "uncertainty". In the first case, the mean response is plotted, in the latter case the uncertainty is plotted.

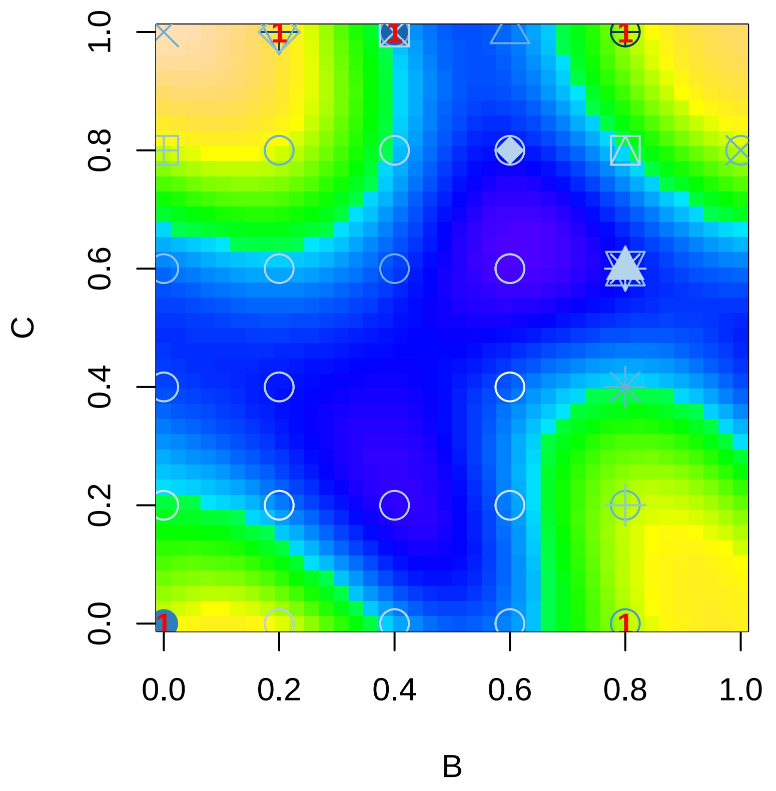
Outputs

A 2D contour plot (blue-low, yellow-high) that shows how the mean response or uncertainty (from the model) varies with respect to viz_factors, when the fixed_factors are constrained. The symbols in the plot correspond to the configurations proposed in LAMBDA, and the different symbols indicate the different clusters. The color of the symbols corresponds to the position in the rank, from white (low position, small uncertainty) to blue (high position, high uncertainty). The points in D_add_j are indicated with a red number, that shows the number of replicates of that configuration (a configuration is considered as replicated if another point shares the same configuration of viz_factors, as fixed_factors are not considered).

Example

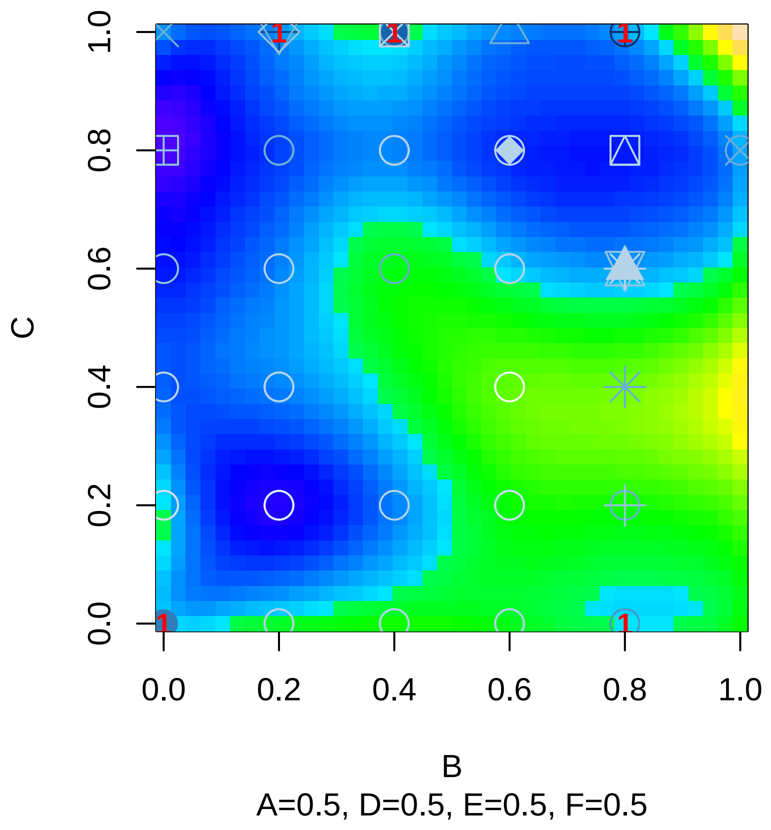
```
#here load the models with readRDS(). We load 7 models
```

```
plot2D_mean<-ALPERC_2D_viz(D_add_j=results$D_add_j,
                           LAMBDA=results$LAMBDA,
                           model=model5,
                           viz_factors=c("B", "C"),
                           fixed_factors=c("A","D","E","F"),
                           fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                           what_plot="mean")
```



A=0.5, D=0.5, E=0.5, F=0.5

```
plot2D_uncert<-ALPERC_2D_viz(D_add_j=results$D_add_j,
                             LAMBDA=results$LAMBDA,
                             model=model5,
                             viz_factors=c("B", "C"),
                             fixed_factors=c("A","D","E","F"),
                             fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                             what_plot="uncertainty")
```



ALPERC_2D_viz_multiv()

A function equivalent to ALPERC_2D_viz(), but provides a visualization of multiple responses in one plot. The contour plot is obtained as the average of the mean responses (or uncertainty) from the models. For this reason, raw data (responses) should be transformed to 0-1 before fitting the models. It currently works only for hetGP and ranger models.

Usage

ALPERC_2D_viz_multiv(D_add_j=D_add_j, LAMBDA=LAMBDA, model=model, viz_factors=viz_factors, fixed_factors=fixed_factors, fixed_factor_levels=fixed_factor_levels, what_plot=what_plot)

Arguments

Argument	Description
D_add_j	A tibble, that is the output of ALPERC() function.
LAMBDA	A tibble, that is the output of ALPERC() function.
model	A list of hetGP or ranger (trained via caret) models.
viz_factors	A character vector that includes the names of 2 factors that will be used in the plot.
fixed_factors	A character vector that includes the names of the factors that will be used fixed. They are all the input variables except from those included in viz_factors.
fixed_factor_levels	A numeric vector that includes the chosen levels of the fixed_factors. The order must correspond to the order in fixed_factors

Argument	Description
what_plot	A character vector that is either "mean" or "uncertainty". In the first case, the average mean response is plotted, in the latter case the average uncertainty is plotted.

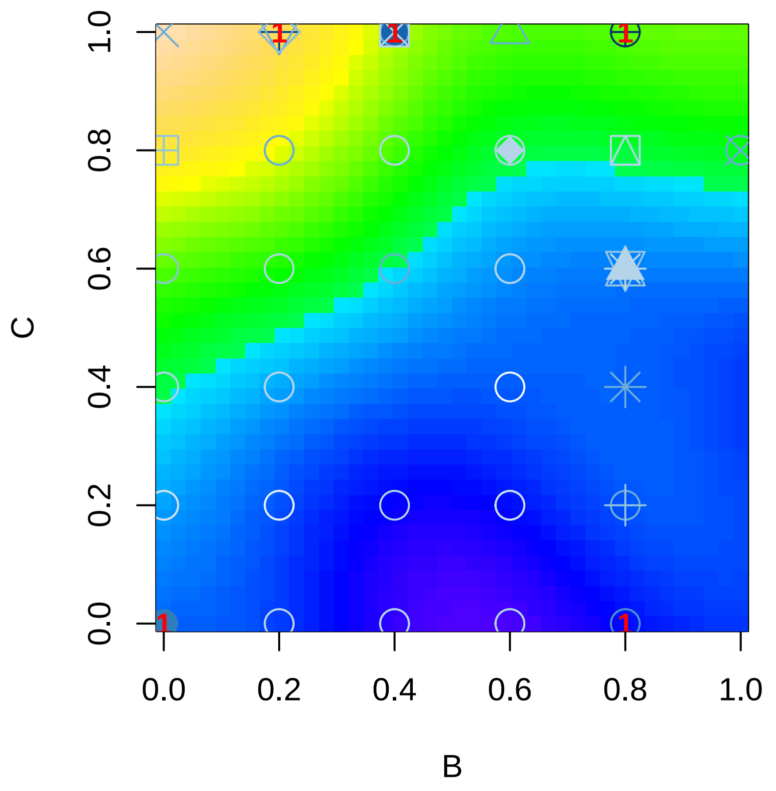
Outputs

A 2D contour plot (blue-low, yellow-high) that shows how the average response or average uncertainty (over the responses as predicted by the models) varies with respect to viz_factors, when the fixed_factors are constrained. The symbols in the plot correspond to the configurations proposed in LAMBDA, and the different symbols indicate the different clusters. The color of the symbols corresponds to the position in the rank, from white (low position, small uncertainty) to blue (high position, high uncertainty).

Example

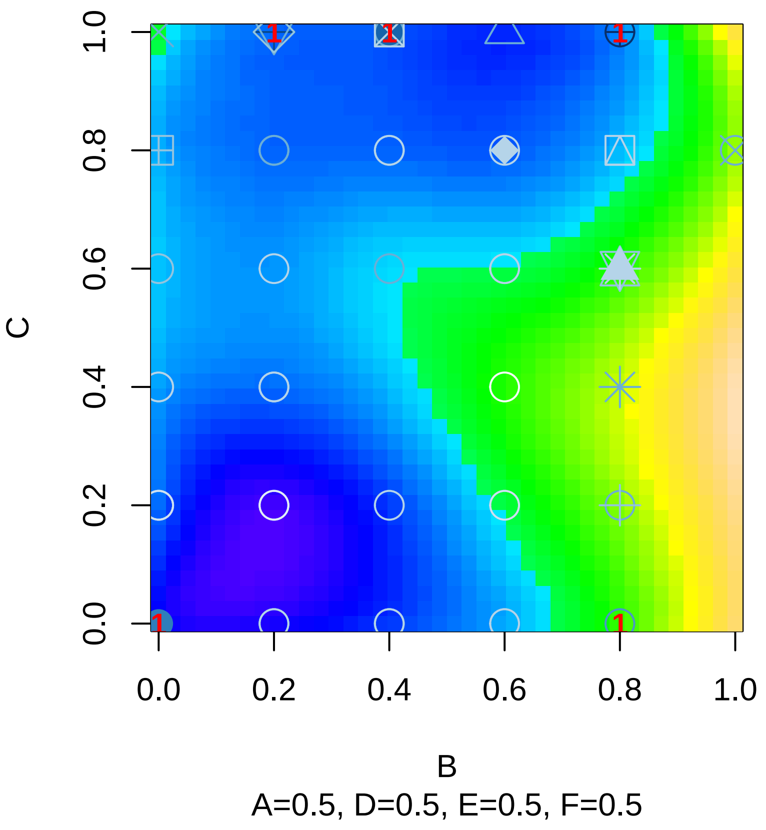
```
model_list<-list(model1,
                 model2,
                 model3,
                 model4,
                 model5,
                 model6,
                 model7
                 )
```

```
plot2D_multiv_m<-ALPERC_2D_viz_multiv(D_add_j=results$D_add_j,
                      LAMBDA=results$LAMBDA,
                      model=model_list,
                      viz_factors=c("B", "C"),
                      fixed_factors=c("A","D","E","F"),
                      fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                      what_plot="mean")
```



A=0.5, D=0.5, E=0.5, F=0.5

```
plot2D_multiv_u<-ALPERC_2D_viz_multiv(D_add_j=results$D_add_j,
                          LAMBDA=results$LAMBDA,
                          model=model_list,
                          viz_factors=c("B", "C"),
                          fixed_factors=c("A","D","E","F"),
                          fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                          what_plot="uncertainty")
```



ALPERC_3D_viz()

A function that provides a 3D visualization of D_add_j and LAMBDA (from an ALPERC element), together with the mean response or uncertainty as predicted by the model. It is equivalent to ALPERC_2D_viz(), but produces an interactive surface plot in 3 dimensions. It currently works only for hetGP and ranger models.

Usage

ALPERC_3D_viz(D_add_j=D_add_j, LAMBDA=LAMBDA, model=model, viz_factors=viz_factors, fixed_factors=fixed_factors, fixed_factor_levels=fixed_factor_levels, what_plot=what_plot, n_pred_points=n_pred_point
)

Arguments

Argument	Description
D_add_j	A tibble, that is the output of ALPERC() function.
LAMBDA	A tibble, that is the output of ALPERC() function.
model	A hetGP or ranger (trained via caret) model.
viz_factors	A character vector that includes the names of 2 factors that will be used in the plot.
fixed_factors	A character vector that includes the names of the factors that will be fixed. They are all the input variables except from those included in viz_factors.
fixed_factor_levels	A numeric vector that includes the chosen levels of the fixed_factors. The order must correspond to the order in fixed_factors

Argument	Description
what_plot	A character vector that is either “mean” or “uncertainty”. In the first case, the mean response is plotted, in the latter case the uncertainty is plotted.
n_pred_points	An integer, that indicates the number of points (randomly sampled) from the surface, for which uncertainty of prediction is reported. If not specified, 10k points are plotted.

Outputs

An interactive 3D surface plot that shows how the response (from the model) or uncertainty varies with respect to viz_factors, when the fixed_factors are constrained. In the bottom plane the candidate configurations are shown with different colors, corresponding to the different clusters. A number indicates the number of replicates (only viz_factors are considered). Red points indicate the mean uncertainty of predictions in some locations.

Example

```
plot3D_m<-ALPERC_3D_viz(D_add_j=results$D_add_j,
                        LAMBDA=results$LAMBDA,
                        model=model5,
                        viz_factors=c("B", "C"),
                        fixed_factors=c("A","D","E","F"),
                        fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                        what_plot="mean",
                        n_pred_points=100)
plot3D_m
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please
```

```
plot3D_u<-ALPERC_3D_viz(D_add_j=results$D_add_j,
                        LAMBDA=results$LAMBDA,
                        model=model5,
                        viz_factors=c("B", "C"),
                        fixed_factors=c("A","D","E","F"),
                        fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                        what_plot="uncertainty",
                        n_pred_points=0)
plot3D_u
```

ALPERC_3D_viz_multiv()

A function equivalent to ALPERC_3D_viz(), but provides a visualization of multiple responses in one plot. The surface is obtained as the average of the responses (or uncertainty) from the models. For this reason, raw data (responses) should be transformed to 0-1 before fitting the models. It is equivalent to ALPERC_2D_viz_multiv(), but produces an interactive surface plot in 3 dimensions. It currently works only for hetGP and ranger models.

Usage

ALPERC_3D_viz_multiv(D_add_j=D_add_j, LAMBDA=LAMBDA, model=model, viz_factors=viz_factors, fixed_factors=fixed_factors, fixed_factor_levels=fixed_factor_levels, what_plot=what_plot, n_pred_points=n_pred_point
)

Arguments

Argument	Description
D_add_j	A tibble, that is the output of ALPERC() function.
LAMBDA	A tibble, that is the output of ALPERC() function.
model	A list of hetGP or ranger (trained via caret) models. Maximum 10.
viz_factors	A character vector that includes the names of 2 factors that will be used in the plot.
fixed_factors	A character vector that includes the names of the factors that will be used fixed. They are all the input variables except from those included in viz_factors.
fixed_factor_levels	A numeric vector that includes the chosen levels of the fixed_factors. The order must correspond to the order in fixed_factors
what_plot	A character vector that is either "mean" or "uncertainty". In the first case, the average mean response is plotted, in the latter case the average uncertainty is plotted.
n_pred_points	An integer, that indicates the number of points (randomly sampled) from the surface, for which uncertainty of the prediction is reported. If not specified, 10k points are plotted.

Outputs

An interactive 3D surface plot that shows how the mean response (average from the models) or the mean uncertainty (average from the models) varies with respect to viz_factors, when the fixed_factors are constrained. In the bottom plane the candidate configurations are shown with different colors, corresponding to the different clusters. A number indicates the number of replicates (only viz_factors are considered). Red points indicate the mean uncertainty of predictions (calculated as the mean uncertainty from all models in the same location) in some locations.

Example

```
plot3D_multiv_m<-ALPERC_3D_viz_multiv(D_add_j=results$D_add_j,
                           LAMBDA=results$LAMBDA,
                           model=model_list,
                           viz_factors=c("B", "C"),
                           fixed_factors=c("A","E","D","F"),
                           fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                           what_plot="mean",
                           n_pred_points=100)
plot3D_multiv_m
```

```
plot3D_multiv_u<-ALPERC_3D_viz_multiv(D_add_j=results$D_add_j,
                           LAMBDA=results$LAMBDA,
                           model=model_list,
                           viz_factors=c("B", "C"),
                           fixed_factors=c("A","E","D","F"),
                           fixed_factor_levels=c(0.5,0.5,0.5,0.5),
                           what_plot="uncertainty",
                           n_pred_points=0)
plot3D_multiv_u
```