Package ‘Racmacs’

September 4, 2023

Type    Package
Title   Antigenic Cartography Macros
Version 1.2.7
Date    2023-09-01

Description  A toolkit for making antigenic maps from immunological assay data, in order to quantify and visualize antigenic differences between different pathogen strains as described in Smith et al. (2004) <doi:10.1126/science.1097211> and used in the World Health Organization influenza vaccine strain selection process. Additional functions allow for the diagnostic evaluation of antigenic maps and an interactive viewer is provided to explore antigenic relationships amongst several strains and incorporate the visualization of associated genetic information.

Depends    R (>= 4.0)
Imports    Rcpp, jsonlite, ks, brotli, shiny, shinyFiles, shinyjs, htmlwidgets, ggplot2 (>= 3.0.0), htmltools, rmarchingcubes, shape, ellipsis, MASS, magrittr, igraph, dplyr, vctrs, rlang
Suggests   testthat, r3js, knitr, rmarkdown, rstudioapi, plotly, geometry, gdata, stringr, tibble, tidyr, base64enc, lifecycle, MCMCpack
LinkingTo  Rcpp, RcppArmadillo, RcppProgress, RcppEnsmallen, rapidjsonr
Encoding   UTF-8
RooxygenNote 7.2.3
VignetteBuilder knitr

BugReports  https://github.com/acorg/Racmacs/issues
License    AGPL-3
NeedsCompilation yes
Author     Sam Wilks [aut, cre]
Maintainer  Sam Wilks <sw463@cam.ac.uk>
Repository  CRAN
Date/Publication  2023-09-04 09:40:05 UTC

R topics documented:

acmap .................................................. 4
acmapAttributes ...................................... 5
addOptimization ...................................... 6
adjustedLogTiterTable ............................... 7
adjustedTiterTable ................................. 8
agAttributes ......................................... 9
agCohesion .......................................... 10
agGroups ........................................... 11
agHomologousSr ..................................... 11
agLabIDs ............................................ 12
agReactivityAdjustments ......................... 12
agSequences ....................................... 13
applyMapTransform ................................. 14
applyPlotspec ...................................... 14
as.json ............................................. 15
blob ................................................ 15
blobsize ........................................... 16
bootstrapBlobs ..................................... 17
bootstrapMap ....................................... 18
checkHemisphering .................................. 20
colBases ........................................... 21
deprecated_functions .............................. 22
dilutionStepsize ................................... 22
dimensionTestMap .................................. 23edit_agNames ...................................... 24edit_srNames ....................................... 25
export_viewer ...................................... 26
getOptimization .................................... 26
ggplot.acmap ....................................... 27
htmlAdjustedTiterTable ......................... 29
htmlMergeReport .................................. 29
htmlTiterTable .................................... 30
keepBestOptimization ............................. 30
keepOptimizations ................................. 31
keepSingleOptimization ......................... 31
layerNames ........................................ 32
listOptimizations ................................. 33
logtiterTable ...................................... 33
logtiterTableLayers ............................... 34
make.acmap ........................................ 34
map-table-distances .............................. 36
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapBootstrapCoords</td>
</tr>
<tr>
<td>mapComment</td>
</tr>
<tr>
<td>mapDescription</td>
</tr>
<tr>
<td>mapDimensions</td>
</tr>
<tr>
<td>mapDistances</td>
</tr>
<tr>
<td>mapGadget</td>
</tr>
<tr>
<td>mapName</td>
</tr>
<tr>
<td>mapRelaxed</td>
</tr>
<tr>
<td>mapResiduals</td>
</tr>
<tr>
<td>mapStress</td>
</tr>
<tr>
<td>mapTransformation</td>
</tr>
<tr>
<td>matchStrains</td>
</tr>
<tr>
<td>mergeMaps</td>
</tr>
<tr>
<td>mergeReport</td>
</tr>
<tr>
<td>moveTrappedPoints</td>
</tr>
<tr>
<td>optimizationProperties</td>
</tr>
<tr>
<td>optimizeAgReactivity</td>
</tr>
<tr>
<td>optimizeMap</td>
</tr>
<tr>
<td>orderPoints</td>
</tr>
<tr>
<td>plot.acmap</td>
</tr>
<tr>
<td>pointStress</td>
</tr>
<tr>
<td>procrustesData</td>
</tr>
<tr>
<td>procrustesMap</td>
</tr>
<tr>
<td>ptAnnotations</td>
</tr>
<tr>
<td>ptBaseCoords</td>
</tr>
<tr>
<td>ptBootstrapBlob</td>
</tr>
<tr>
<td>ptBootstrapCoords</td>
</tr>
<tr>
<td>ptClades</td>
</tr>
<tr>
<td>ptCoords</td>
</tr>
<tr>
<td>ptDrawingOrder</td>
</tr>
<tr>
<td>ptLeverage</td>
</tr>
<tr>
<td>ptOpacity</td>
</tr>
<tr>
<td>ptStyles</td>
</tr>
<tr>
<td>ptTriangulationBlob</td>
</tr>
<tr>
<td>RacMerge.options</td>
</tr>
<tr>
<td>RacOptimizer.options</td>
</tr>
<tr>
<td>RacViewer</td>
</tr>
<tr>
<td>RacViewer-shiny</td>
</tr>
<tr>
<td>RacViewer.options</td>
</tr>
<tr>
<td>randomizeCoords</td>
</tr>
<tr>
<td>read.acmap</td>
</tr>
<tr>
<td>read.titerTable</td>
</tr>
<tr>
<td>realignMap</td>
</tr>
<tr>
<td>realignOptimizations</td>
</tr>
<tr>
<td>recalculateStress</td>
</tr>
<tr>
<td>reflectMap</td>
</tr>
<tr>
<td>relaxMap</td>
</tr>
<tr>
<td>relaxMapOneStep</td>
</tr>
</tbody>
</table>
acmap

Generate a new acmap object

Description

This function generates a new acmap object, the base object for storing map data in the Racmacs package.

Usage

acmap(
  ag_names = NULL,
  sr_names = NULL,
  titer_table = NULL,
  ag_coords = NULL,
  sr_coords = NULL,
  check_duplicates = TRUE,
acmapAttributes

Arguments

- `ag_names` : Antigen names
- `sr_names` : Sera names
- `titer_table` : Table of titer data
- `ag_coords` : Antigenic coordinates for an optimization run record (optional)
- `sr_coords` : Sera coordinates for an optimization run record (optional)
- `check_duplicates` : Issue a warning if duplicate antigen or sera names are found
- ... Further arguments passed to addOptimization()

Details

The fundamental unit of the Racmacs package is the acmap object, short for Antigenic Cartography MAP. This object contains all the information about an antigenic map. You can read in a new acmap object from a file with the function `read.acmap()` and create a new acmap object within an R session using the `acmap()` function.

Value

Returns the new acmap object

See Also

See `optimizeMap()` for generating new optimizations estimating antigen similarity from the acmap titer data.

Other functions for working with map data: `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.acmap()`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints`, `subsetMap()`
Usage

numAntigens(map)
numSera(map)
numSeraGroups(map)
numPoints(map)
numOptimizations(map)
numLayers(map)

Arguments

map The acmap data object

Value

A number relating to the attribute

See Also

Other map attribute functions: adjustedLogTiterTable(), adjustedTiterTable(), dilutionStepsize(), logtiterTableLayers(), mapDescription(), mapName(), titerTableLayers(), titerTable()

addOptimization Add a new optimization to an acmap object

Description

Function to add a new optimization to an acmap object, with specified values.

Usage

addOptimization(
  map,
  ag_coords = NULL,
  sr_coords = NULL,
  number_of_dimensions = NULL,
  minimum_column_basis = "none",
  fixed_column_bases = NULL,
  ag_reactivity_adjustments = NULL
)
adjustedLogTiterTable

Arguments

map The acmap data object
ag_coords Antigen coordinates for the new optimization (0 if not specified)
sr_coords Sera coordinates for the new optimization (0 if not specified)
number_of_dimensions The number of dimensions of the new optimization
minimum_column_basis The minimum column basis to use for the new optimization
fixed_column_bases A vector of fixed column bases with NA for sera where the minimum column basis should be applied
ag_reactivity_adjustments A vector of antigen reactivity adjustments to apply to each antigen. Corresponding antigen titers will be adjusted by these amounts when calculating column bases and table distances.

Value

Returns the acmap data object with new optimization added (but not selected).

See Also

Other functions for working with map data: acmap(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints, read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()

adjustedLogTiterTable  Get the reactivity adjusted log titer table

Description

Return the log titer table plus any antigen reactivity adjustments.

Usage

adjustedLogTiterTable(map, optimization_number = 1)

Arguments

map An acmap object
optimization_number The optimization number from which to take any antigen reactivity adjustments
Value

A numeric matrix of adjusted log titers.

See Also

Other map attribute functions: acmapAttributes, adjustedTiterTable(), dilutionStepsize(), logtiterTableLayers(), mapDescription(), mapName(), titerTableLayers(), titerTable()
Description

These functions get and set the antigen attributes for a map.

Usage

agIDs(map)
agIDs(map) <- value
agDates(map)
agDates(map) <- value
agReference(map)
agReference(map) <- value
agNames(map)
agNames(map) <- value
agExtra(map)
agExtra(map) <- value
agPassage(map)
agPassage(map) <- value
agLineage(map)
agLineage(map) <- value
agReassortant(map)
agReassortant(map) <- value
agStrings(map)
agStrings(map) <- value
agContinent(map)
agContinent(map) <- value

Arguments

map  The acmap data object
value New value to set

Value

Returns either the requested attribute when using a getter function or
the updated acmap object when using the setter function.

See Also

srAttributes()

Other antigen and sera attribute functions: agGroups(), agHomologousSr(), agLabIDs(), agSequences(),
ptAnnotations, ptClades, srAttributes, srGroups(), srHomologousAgs(), srSequences()
agCohesion

Check map cohesion

Description

Checks the vertex connectivity of points in a map (the minimum number of points needed to remove from the map to eliminate all paths from one point to another point). This is for checking for example if after merging maps you only have a small number of points in common between separate groups of points, leading to a situation where groups of points cannot be robustly positioned relative to each other. If the vertex connectivity is smaller than the number of map dimensions + 1 then this will certainly be occurring and will lead to an unstable map solution. mapCohesion() returns the minimum vertex connectivity found between any given points, while agCohesion() and srCohesion() return the vertex connectivity between each pair of antigens and sera as a table helping to diagnose which antigens and sera are forming separate groups. Note that for these purposes only detectable titers count as connections and non-detectable titers are ignored.

Usage

agCohesion(map)
srCohesion(map)
mapCohesion(map)

Arguments

map An acmap object

Value

A scalar real value.

See Also

Other map diagnostic functions: bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs(), unstableMaps
Getting and setting antigen groups

Description
These functions get and set the antigen groupings for a map.

Usage
agGroups(map)
agGroups(map) <- value

Arguments
map The acmap object
value A character or factor vector of groupings to apply to the antigens

Value
A factor vector of groupings.

See Also
Other antigen and sera attribute functions:
agAttributes, agHomologousSr(), agLabIDs(), agSequences(),
ptAnnotations, ptClades, srAttributes, srGroups(), srHomologousAgs(), srSequences()

Get homologous sera for each antigen

Description
Gets the indices of homologous sera for each antigen in an antigenic map. See also the function
srHomologousAgs() for getting and setting the homologous antigens reciprocally.

Usage
agHomologousSr(map)

Arguments
map An acmap object

Value
A list, where each entry is a vector of indices for homologous sera, or a length 0 vector where no
homologous serum is present
See Also

Other antigen and sera attribute functions: `agAttributes`, `agGroups()`, `agSequences()`, `ptAnnotations`, `ptClades`, `srAttributes`, `srGroups()`, `srHomologousAgs()`, `srSequences()`

---

**agLabIDs**

*Getting and setting antigen lab id information*

Description

Getting and setting antigen lab id information

Usage

```r
agLabIDs(map)

agLabIDs(map) <- value
```

Arguments

- `map` The acmap data object
- `value` A list of character vectors with lab ids information for each point

Value

A character vector of antigen laboratory IDs

See Also

Other antigen and sera attribute functions: `agAttributes`, `agGroups()`, `agHomologousSr()`, `agSequences()`, `ptAnnotations`, `ptClades`, `srAttributes`, `srGroups()`, `srHomologousAgs()`, `srSequences()`

---

**agReactivityAdjustments**

*Get and set antigen reactivity adjustments*

Description

Get and set antigen reactivity adjustments

Usage

```r
agReactivityAdjustments(map)

agReactivityAdjustments(map) <- value
```
Ag Sequences

Arguments

map  The acmap object
value A vector of antigen reactivity adjustments to apply

Value

A numeric vector of antigen reactivity adjustments

See Also

Other functions for working with map data: acmap(), addOptimization(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints(), read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()

Description

Getting and setting antigen sequence information

Usage

agSequences(map, missing_value = ".")

agSequences(map) <- value

agNucleotideSequences(map, missing_value = ".")

agNucleotideSequences(map) <- value

Arguments

map  The acmap data object
missing_value Character to use to fill in portions of the sequence matrix where sequence data is missing.
value  A character matrix of sequences with rows equal to the number of antigens

Value

A character matrix of sequences, where each row represents an antigen.

See Also

Other antigen and sera attribute functions: agAttributes, agGroups(), agHomologousSr(), agLabIDs(), ptAnnotations, ptClades, srAttributes, srGroups(), srHomologousAgs(), srSequences()
applyMapTransform  

Apply the current map transformation

Description
Applies the map transformation associated with a selected optimization run to a set of coordinates.

Usage
```
applyMapTransform(coords, map, optimization_number = 1)
```

Arguments
- `coords`: Coordinates to transform
- `map`: The acmap object
- `optimization_number`: The optimization number

Value
An acmap object with transformation applied

See Also
Other functions relating to map transformation: `reflectMap()`, `rotateMap()`, `translateMap()`

applyPlotspec  

Apply a plotspec from another acmap

Description
Copy point style from matching antigens and sera in another acmap

Usage
```
applyPlotspec(map, source_map)
```

Arguments
- `map`: The acmap object
- `source_map`: An acmap object from which to copy point styles

Value
Returns the acmap object with updated point styles (unmatched point styles unchanged)
as.json

See Also

Other map point style functions: ptDrawingOrder(), ptOpacity, ptStyles

---

as.json

Convert map to json format

Description

Convert map to json format

Usage

as.json(map, pretty = FALSE, round_titers = FALSE)

Arguments

map The map data object
pretty Should json be output prettily with new lines and indentation?
round_titers Should titers be rounded to the nearest integer before outputting

Value

Returns map data as .ace json format

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints, read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()

---

blob

Plot a blob object

Description

Plot a blob object such as that return from agBootstrapBlob() using the polygon() function.

Usage

blob(x, col, border, lwd, alpha = 1, ...)
Arguments

- **x**: The blob object to plot
- **col**: Color for the blob fill
- **border**: Color for the blob outline
- **lwd**: Line width for the blob outline
- **alpha**: Blob opacity
- **...**: Additional arguments to pass to `polygon()`

Value

No return value, called for the side effect of plotting the blobs.

See Also

Other additional plotting functions: `blobsize()`

---

**blobsize**

*Calculate size of a blob object*

Description

Returns either the area (for 2D blobs) or volume (for 3D blobs)

Usage

`blobsize(blob)`

Arguments

- **blob**: The blob object

Value

A numeric vector

See Also

Other additional plotting functions: `blob()`
bootstrapBlobs  

**Calculate bootstrap blob data for an antigenic map**

**Description**

This function takes a map for which the function `bootstrapMap()` has already been applied and draws contour blobs for each point illustrating how point position varies in each bootstrap repeat. The blobs are calculated using kernel density estimates according to these point distribution and drawn so as to encompass a given proportion of this variation according to the parameter `conf.level`. A `conf.level` set at 0.95 for example will draw blobs that are calculated to encompass 95% of the positional variation seen in the bootstrap repeats. Note however that the accuracy of these estimates will depend on the number of bootstrap repeats performed, for example whether 100 or 1000 repeats were performed in the initial calculations using `bootstrapMap()`.

**Usage**

```r
bootstrapBlobs(
  map,
  conf.level = 0.68,
  smoothing = 6,
  gridspacing = 0.25,
  antigens = TRUE,
  sera = TRUE,
  method = "ks"
)
```

**Arguments**

- **map**  
  The acmap data object
- **conf.level**  
  The proportion of positional variation captured by each blob
- **smoothing**  
  The amount of smoothing to perform when performing the kernel density estimate, larger equates to more smoothing
- **gridspacing**  
  Grid spacing to use when calculating blobs, smaller values will produce more accurate blobs with smoother edges but will take longer to calculate.
- **antigens**  
  Should blobs be calculated for antigens
- **sera**  
  Should blobs be calculated for sera
- **method**  
  One of "MASS", the default, or "ks", specifying the algorithm to use when calculating blobs in 2D. 3D will always use ks::kde.

**Value**

Returns an acmap object that will then show the corresponding bootstrap blobs when viewed or plotted.
See Also

Other map diagnostic functions: \texttt{agCohesion()}, \texttt{bootstrapMap()}, \texttt{checkHemisphering()}, \texttt{dimensionTestMap()}, \texttt{logtiterTable()}, \texttt{map-table-distances}, \texttt{mapBootstrapCoords}, \texttt{mapDistances()}, \texttt{mapRelaxed()}, \texttt{mapResiduals()}, \texttt{pointStress}, \texttt{ptBootstrapBlob}, \texttt{ptBootstrapCoords()}, \texttt{ptLeverage}, \texttt{ptTriangulationBlob}, \texttt{recalculateStress()}, \texttt{stressTable()}, \texttt{tableColbases()}, \texttt{tableDistances()}, \texttt{triangulationBlobs()}, \texttt{unstableMaps}

\begin{verbatim}
bootstrapMap

Perform a bootstrap on a map

Description

This function takes the map and original titer table, and performs a version of bootstrapping defined by the method argument. For each bootstrap run this process is performed and a record of the coordinates of points in the lowest stress solution is kept. See details for a description of the bootstrapping methods you can apply.

Usage

\begin{verbatim}
bootstrapMap(
  map,
  method,
  bootstrap_repeats = 1000,
  bootstrap_ags = TRUE,
  bootstrap_sr = TRUE,
  reoptimize = TRUE,
  optimizations_per_repeat = 100,
  ag_noise_sd = 0.7,
  titer_noise_sd = 0.7,
  options = list()
)
\end{verbatim}

Arguments

\begin{verbatim}
map

The map object

method

One of "resample", "bayesian" or "noisy" (see details)

bootstrap_repeats

The number of bootstrap repeats to perform

bootstrap_ags

For "resample" and "bayesian" methods, whether to apply bootstrapping across antigens

bootstrap_sr

For "resample" and "bayesian" methods, whether to apply bootstrapping across sera

reoptimize

Should the whole map be reoptimized with each bootstrap run. If FALSE, the map is simply relaxed from it's current optimization with each run.
\end{verbatim}
optimizations_per_repeat

When re-optimizing the map from scratch, the number of optimization runs to perform

ag_noise_sd

The standard deviation (on the log titer scale) of measurement noise applied per antigen when using the "noisy" method

titer_noise_sd

The standard deviation (on the log titer scale) of measurement noise applied per titer when using the "noisy" method

options

Map optimizer options, see RacOptimizer.options()

Details

**Bootstrapping methods:**

"resample": The resample bootstrap is the most standard bootstrap method, a random resample of the titer table data is taken with replacement. Depending on your specification, resampling is applied across either individual antigens, individual sera or both antigens and sera. In essence this method tries to let you see how robust the map is to inclusion of particular titer measurements or antigens or sera. Like most bootstrapping techniques it will prove give more reliable results the more antigens and sera you have in your map. It won’t work very well for a map of 5 sera and antigens for example, in this case a "noisy" bootstrap may be better.

"bayesian": The bayesian bootstrap is akin to the resampling bootstrap, but rather than explicitly resampling data, weights are assigned to each part of the titer table data according to random draws from a dirichilet distribution. Under this scheme, every data point will play at least some role in making the map, even if only weighted slightly. Sometimes this is helpful, if you know for example that the points in your map are highly dependent upon the presence of a few antigens / sera / titers to achieve reasonable triangulation of point positions and you don’t really want to risk removing them completely and ending up with bootstrap runs that are under-constrained, you might want to consider this approach. On the other hand this might be exactly what you don’t want and you want to know uncertainty that can be generated when certain subsets of the data are excluded completely, in that case you probably want to stick with the "resample" method.

"noisy": The noisy bootstrap, sometimes termed a smooth bootstrap involved adding normally distributed noise to each observation. The distribution of this noise can be parametrised through the ag_noise_sd and titer_noise_sd arguments. titer_noise_sd refers to the standard deviation (on the log scale) of noise added to each individual titer measurement in the table, while antigen_noise_sd refers to the standard deviation of noise applied to titers for each antigen. The reason for this distinction is that we have noticed with repeat measurements of influenza data there is often both a random noise per titer and a random noise per antigen, i.e. in one repeat titers may all be around one 2-fold higher on average, in addition to unbiased additional titer noise. If you wish to only simulate additional noise per titer and not a per antigen effect, simply set antigen_noise_sd to 0. Note that in order to use this most effectively it is best to have an idea of the amount and type of measurement noise you may expect in your data and set these parameters accordingly.

Value

Returns the map object updated with bootstrap information
checkHemisphering

See Also

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`

---

cHECKHemisphering Check for hemisphering or trapped points

Description

Check for hemisphering or trapped points

Usage

```r
checkHemisphering(
  map,
  optimization_number = 1,
  grid_spacing = 0.25,
  stress_lim = 0.1,
  options = list()
)
```

Arguments

- **map**
  - The acmap data object
- **optimization_number**
  - The map optimization number
- **grid_spacing**
  - When doing a grid search of more optimal point positions the grid spacing to use
- **stress_lim**
  - The stess difference to use when classifying a point as "hemisphering" or not
- **options**
  - A named list of options to pass to RacOptimizer.options()

Value

Returns a data frame with information on any points that were found to be hemisphering or trapped.

See Also

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`
Getting and setting column bases

Description

Functions to get and set column bases specified for an optimization run, either through the minimum column basis or through a vector of specified column bases.

Usage

```
minColBasis(map, optimization_number = 1)
minColBasis(map, optimization_number = 1) <- value
fixedColBases(map, optimization_number = 1)
fixedColBases(map, optimization_number = 1) <- value
```

Arguments

- `map` The acmap data object
- `optimization_number` The optimization run from which to get / set the data
- `value` New value to set

Details

In general a map can have column bases that are specified either through a minimum column basis or a vector of fixed column bases for each sera. When you call `minColBasis()`, it will return the minimum column basis if it has been set, or "fixed" if column bases have instead been fixed directly. The `colBases()` function will return the column bases as calculated for a given optimization run. Setting column bases through this function with `colBases()<-` will fix the column bases to the supplied vector of values.

Note that although the output from `colBases()` might be the same in a case where a minimum column basis was set or a case where column bases were set explicitly, when a minimum column basis is set, the column bases will still depend on the log titers recorded against a given sera, so changing the titers may therefore change the actual column bases calculated. For fixed column bases case, column bases will remain fixed at their values independently of measured titers.

Value

Returns either the requested attribute when using a getter function or the updated acmap object when using the setter function.

See Also

Other map optimization attribute functions: `mapComment()`, `mapDimensions()`, `mapStress()`, `mapTransformation()`, `ptBaseCoords()`, `ptCoords()`
Deprecated functions

Description

These functions still work but have been deprecated in favour of another function. Arguments will be passed onto the new function with a warning.

Usage

stressBlobs(...)

Arguments

... Arguments to pass to the new function

Value

Values from the new function

dilutionStepsize

Get or set the dilution stepsize associated with a map

Description

This defaults to 1 but can be changed using this function with knock-on effects for how < values are treated when maps are optimized or relaxed and the way stress is calculated, see details.

Usage

dilutionStepsize(map)

dilutionStepsize(map) <- value

Arguments

map The acmap object from which to get or set the dilution stepsize
value The dilution stepsize value to set
Details

Antigenic cartography was originally developed for HI titers which typically follow a 2-fold dilution series starting from 1/10, then 1/20, 1/40 etc. This represents a "dilution stepsize" of 1 when converted to the log2 scale. When no inhibition was recorded at the highest dilution, the value is typically recorded as <10 but the optimization regime effectively treats this as a <=5, the rationale being that, had the dilution series been continued to higher concentrations, the next lowest titer would have been a 5. Over time the method has also been applied to other neutralization assays that sometimes have a continuous read out with a lower end, in these cases a <10 really means a <10 since any other values like 9.8 or 7.62 would also be possible. To indicate these continuous cases, you can specify the dilution stepsize as 0. Equally, if the dilution regime followed a different pattern, you can also set that here.

Value

A number giving the current dilution stepsize setting for a map.

See Also

Other map attribute functions: acmapAttributes, adjustedLogTiterTable(), adjustedTiterTable(), logtiterTableLayers(), mapDescription(), mapName(), titerTableLayers(), titerTable()
test_proportion
    The proportion of data to be used as the test set for each test run
minimum_column_basis
    The minimum column basis to use
fixed_column_bases
    A vector of fixed column bases with NA for sera where the minimum column
    basis should be applied
number_of_optimizations
    The number of optimizations to perform when creating each map for the dimen-
    sion test
replicates_per_dimension
    The number of tests to perform per dimension tested
options
    Map optimizer options, see RacOptimizer.options()

Details
For each run, the ag-sr titers that were randomly excluded are predicted according to their relative
positions in the map trained without them. An RMSE is then calculated by comparing predicted
titers inferred from the map on the log scale to the actual log titers. This is done separately for
detectable titers (e.g. 40) and non-detectable titers (e.g. <10). For non-detectable titers, if the
predicted titer is the same or lower than the log-titer threshold, the error is set to 0.

Value
Returns a data frame with the following columns. "dimensions": the dimension tested, "mean_rmse_detectable"
: mean prediction rmse for detectable titers across all runs. "var_rmse_detectable" the variance of
the prediction rmse for detectable titers across all runs, useful for estimating confidence intervals.
"mean_rmse_nondetectable" and "var_rmse_nondetectable" the equivalent for non-detectable titers

See Also
Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(),
logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(),
mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob,
recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs(),
unstableMaps

edit_agNames  Edit antigen names in an acmap

Description
Edit antigen names in an acmap

Usage
edit_agNames(map, old_names, new_names)
**Arguments**

- **map**: The map data object to be updated
- **old_names**: Old names to be replaced
- **new_names**: Replacement for old names

**Value**

Returns the acmap object with antigen names updated.

**See Also**

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints()`, `read.acmap()`, `read.titerTable()`, `removePoints()`, `save.acmap()`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints()`, `subsetMap()`
**getOptimization**

**Description**

Gets the details associated with the currently selected or specified acmap optimization as a list.

**Usage**

```r
getOptimization(map, optimization_number = 1)
```

**Arguments**

- `map` The acmap data object
- `optimization_number` The optimization data to access

---

**export_viewer**

**Export the map viewer**

**Description**

Export a map in a standalone html viewer

**Usage**

```r
export_viewer(map, file, selfcontained = TRUE, ...)
```

**Arguments**

- `map` The acmap object
- `file` File to save HTML into
- `selfcontained` Whether to save the HTML as a single self-contained file (with external resources base64 encoded) or a file with external resources placed in an adjacent directory.
- `...` Further parameters to `view()`

**Value**

Called for the side effect of saving the viewer to an html file but invisibly returns the map viewer htmlwidget.

**See Also**

Other functions to view maps: `RacViewer.options()`, `RacViewer()`, `ggplot.acmap()`, `mapGadget()`, `plot.acmap()`, `setLegend()`, `view.acmap()`, `view.default()`, `view()`

---

**getOptimization**

**Get optimization details from an acmap object**

**Description**

Gets the details associated with the currently selected or specified acmap optimization as a list.
Value

Returns a list with information about the optimization

See Also

See listOptimizations() for getting information about all optimizations.

---

**ggplot.acmap**

Plot an antigenic map using ggplot

---

**Description**

Method for plotting an antigenic map as a ggplot object

**Usage**

```r
## S3 method for class 'acmap'
ggplot(
  data = NULL,
  mapping = NULL,
  optimization_number = 1,
  xlim = NULL,
  ylim = NULL,
  plot_ags = TRUE,
  plot_sr = TRUE,
  plot_blobs = TRUE,
  plot_hemisphering = TRUE,
  show_procrustes = TRUE,
  show_error_lines = FALSE,
  plot_stress = FALSE,
  indicate_outliers = "arrowheads",
  grid.col = "grey90",
  grid.lwd = 0.5,
  grid.margin.col = "grey50",
  grid.margin.lwd = grid.lwd,
  fill.alpha = 0.8,
  outline.alpha = 0.8,
  padding = 1,
  arrow_angle = 25,
  arrow_length = 0.2,
  margins = rep(0.5, 4),
  ...
)
```
Arguments

- **data**: The acmap to plot
- **mapping**: Default list of aesthetic mappings to use for plot, not currently used
- **optimization_number**: The optimization number to plot
- **xlim**: optional x axis limits
- **ylim**: optional y axis limits
- **plot_ags**: logical, should antigens be plotted
- **plot_sr**: logical, should antigens be plotted
- **plot_blobs**: logical, should stress blobs be plotted if present
- **plot_hemisphering**: logical, should hemisphering points be indicated, if tested for already with checkHemisphering() (and if present)
- **show_procrustes**: logical, should procrustes lines be shown, if present
- **show_error_lines**: logical, should error lines be drawn
- **plot_stress**: logical, should map stress be plotted in lower left corner
- **indicate_outliers**: how should points outside the plotting region be indicated, either FALSE, for not shown, or "arrowheads" for small arrowheads like in the viewer.
- **grid.col**: grid line color
- **grid.lwd**: grid line width
- **grid.margin.col**: grid margin color
- **grid.margin.lwd**: grid margin line width
- **fill.alpha**: alpha for point fill
- **outline.alpha**: alpha for point outline
- **padding**: padding at limits of the antigenic map, ignored if xlim or ylim set explicitly
- **arrow_angle**: angle of arrow heads drawn for procrustes lines
- **arrow_length**: length of arrow heads drawn for procrustes lines in cm
- **margins**: margins in inches for the plot
- **...**: additional arguments, not used
- **environment**: not used

Value

Returns the ggplot plot

See Also

Other functions to view maps: RacViewer.options(), RacViewer(), export_viewer(), mapGadget(), plot.acmap(), setLegend(), view.acmap(), view.default(), view()
htmlAdjustedTiterTable

Return an html formatted titer table with antigen reactivity adjustments applied

Description

Prints an html formatted titer table, visualising with colors things like which titers are the maximum for each sera.

Usage

htmlAdjustedTiterTable(map, optimization_number = 1)

Arguments

map
An acmap object
optimization_number
The optimization number from which to take the antigen reactivity adjustments.

Value

A list() with a Rac_html_merge_report and shiny.tag class that can be converted into an HTML string via as.character() and saved to a file with save_html().

htmlMergeReport
Return an html formatted merge report

Description

Prints an html formatted table merge report of a set of merged maps, visualising with colors how different titers have been merged together.

Usage

htmlMergeReport(map)

Arguments

map
An acmap object that was the result of merging several maps

Value

A list() with a Rac_html_merge_report and shiny.tag class that can be converted into an HTML string via as.character() and saved to a file with save_html().
htmlTiterTable  Return an html formatted titer table

Description
Prints an html formatted titer table, visualising with colors things like which titers are the maximum for each sera.

Usage
htmlTiterTable(map)

Arguments
map  An acmap object

Value
A list() with a Rac_html_merge_report and shiny.tag class that can be converted into an HTML string via as.character() and saved to a file with save_html().

See Also
htmlAdjustedTiterTable

keepBestOptimization
Keep only the lowest stress map optimization

Description
Keep only the lowest stress map optimization

Usage
keepBestOptimization(map)

Arguments
map  The acmap object

Value
An acmap object with only the lowest stress optimization kept
See Also

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.acmap()`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints`, `subsetMap()`

---

**keepOptimizations**  
*Keep specified optimization runs*

**Description**

Keep only data from specified optimization runs.

**Usage**

```r
test <- keepOptimizations(map, optimization_numbers)
```

**Arguments**

- `map`: The acmap object
- `optimization_numbers`: Optimizations to keep

**Value**

Returns the updated acmap object

**See Also**

Other functions to work with map optimizations: `optimizationProperties()`, `removeOptimizations()`, `sortOptimizations()`

---

**keepSingleOptimization**  
*Keep only a single optimization run*

**Description**

Keep only a single optimization run

**Usage**

```r
test <- keepSingleOptimization(map, optimization_number = 1)
```
layerNames

Arguments

map
The acmap object

optimization_number
The optimization run to keep

Value

An acmap object with only one optimization kept

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), layerNames(), orderPoints(), read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()

layerNames

Get and set map layer names

Description

Get and set map layer names

Usage

layerNames(map)

layerNames(map) <- value

Arguments

map
The acmap object

value
A vector of new layer names to apply to the map

Value

A character vector of layer names

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), orderPoints, read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()
listOptimizations

Description

 Gets the details associated with the all the optimizations of an acmap object as a list.

Usage

 listOptimizations(map)

Arguments

 map The acmap data object

Value

 Returns a list of lists with information about the optimizations

See Also

 See getOptimization() for getting information about a single optimization.

logtiterTable

Description

 Converts titers to the log scale via the transformation $\log_2(x/10)$, less-than values are reduced by 1 on the log scale and greater than values are increased by 1, hence <10 => -1 and >1280 => 8

Usage

 logtiterTable(map)

Arguments

 map The acmap object

Value

 Returns a matrix of titers converted to the log scale
See Also

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`

Other functions relating to map stress calculation: `mapDistances()`, `mapResiduals()`, `pointStress`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`

---

`logtiterTableLayers`  
*Return a list of logtiter table layers*

Description

Return a list of logtiter table layers

Usage

`logtiterTableLayers(map)`

Arguments

- `map`  
  An acmap data object

Value

A list of numeric matrices with logtiter values

See Also

Other map attribute functions: `acmapAttributes`, `adjustedLogTiterTable()`, `adjustedTiterTable()`, `dilutionStepsize()`, `mapDescription()`, `mapName()`, `titerTableLayers()`, `titerTable()`

---

`make.acmap`  
*Make an antigenic map from scratch*

Description

This is a wrapper function for first making a map with table data then, running optimizations to make the map otherwise done with `acmap()` followed by `optimizeMap()`.
Usage

make.acmap(
    titer_table = NULL,
    ag_names = NULL,
    sr_names = NULL,
    number_of_dimensions = 2,
    number_of_optimizations = 100,
    minimum_column_basis = "none",
    fixed_column_bases = NULL,
    sort_optimizations = TRUE,
    check_convergence = TRUE,
    verbose = TRUE,
    options = list(),
    ...
)

Arguments

titer_table       A table of titer data
ag_names          A vector of antigen names
sr_names          A vector of sera names
number_of_dimensions       The number of dimensions in the map
number_of_optimizations    The number of optimization runs to perform
minimum_column_basis    The minimum column basis for the map
fixed_column_bases     A vector of fixed values to use as column bases directly, rather than calculating them from the titer table.
sort_optimizations    Should optimizations be sorted by stress afterwards?
check_convergence      Should a basic check for convergence of lowest stress optimization runs onto a similar solution be performed.
verbose                Should progress messages be reported, see also RacOptimizer.options()
options                List of named optimizer options, see RacOptimizer.options()
...                    Further arguments to pass to acmap()

Value

Returns an acmap object that has optimization run results.

See Also

Other map optimization functions: RacOptimizer.options(), moveTrappedPoints(), optimizeMap(), randomizeCoords(), relaxMapOneStep(), relaxMap()
Description

Plot map vs table distances

Usage

plot_map_table_distance(
  map,
  optimization_number = 1,
  xlim,
  ylim,
  line_of_equality = TRUE
)

plotly_map_table_distance(
  map,
  optimization_number = 1,
  xlim,
  ylim,
  line_of_equality = TRUE
)

Arguments

map The acmap data object
optimization_number The optimization number from which to take map and table distances
xlim The x limits of the plot
ylim The y limits of the plot
line_of_equality Should the line x=y be added

Value

Returns the ggplot2 object

See Also

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(),
dimensionTestMap(), logtiterTable(), mapBootstrapCoords, mapDistances(), mapRelaxed(),
mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob,
recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs(),
unstableMaps
**mapBootstrapCoords**  
*Get bootstrap coordinates associated with a map*

**Description**

This can be used to get information about the bootstrap run results after `bootstrapMap()` has been run.

**Usage**

```r
mapBootstrap_ptBaseCoords(map)  
mapBootstrap_agCoords(map)  
mapBootstrap_srCoords(map)
```

**Arguments**

- `map`: The map object

**Value**

Returns a list of coordinate matrices for the points in each of the bootstrap runs

**See Also**

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`

---

**mapComment**  
*Get or set an optimization run comment*

**Description**

Get or set an optimization run comment

**Usage**

```r
mapComment(map, optimization_number = 1)  
mapComment(map, optimization_number = 1) <- value
```
Arguments

map
   The acmap data object
optimization_number
   The optimization run from which to get / set the data
value
   New value to set

Value

Gets or sets map comments for the optimization run.

See Also

Other map optimization attribute functions: `colBases()`, `mapDimensions()`, `mapStress()`, `mapTransformation()`, `ptBaseCoords()`, `ptCoords()`

mapDescription

Getting and setting the map description

Value

Returns either the requested attribute when using a getter function or the updated acmap object when using the setter function.

See Also

Other map attribute functions: `acmapAttributes()`, `adjustedLogTiterTable()`, `adjustedTiterTable()`, `dilutionStepsize()`, `logtiterTableLayers()`, `mapName()`, `titerTableLayers()`, `titerTable()`
mapDimensions

Get the current map dimensions

Description

Get the current map dimensions

Usage

mapDimensions(map, optimization_number = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>The acmap data object</td>
</tr>
<tr>
<td>optimization_number</td>
<td>The optimization run from which to get / set the data</td>
</tr>
</tbody>
</table>

Value

Returns the number of dimensions for the optimization run.

See Also

Other map optimization attribute functions: colBases(), mapComment(), mapStress(), mapTransformation(), ptBaseCoords(), ptCoords()

mapDistances

Return calculated map distances for an acmap

Description

Takes the acmap object and calculates euclidean distances between antigens and sera for the currently selected or specified optimization.

Usage

mapDistances(map, optimization_number = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>The acmap data object</td>
</tr>
<tr>
<td>optimization_number</td>
<td>The optimization number</td>
</tr>
</tbody>
</table>
mapGadget

Value

Returns a matrix of map distances with antigens as rows and sera as columns.

See Also

Other map diagnostic functions: \texttt{agCohesion()}, \texttt{bootstrapBlobs()}, \texttt{bootstrapMap()}, \texttt{checkHemisphering()}, \texttt{dimensionTestMap()}, \texttt{logtiterTable()}, \texttt{map-table-distances}, \texttt{mapBootstrapCoords}, \texttt{mapRelaxed()}, \texttt{mapResiduals()}, \texttt{pointStress}, \texttt{ptBootstrapBlob}, \texttt{ptBootstrapCoords()}, \texttt{ptLeverage}, \texttt{ptTriangulationBlob}, \texttt{recalculateStress()}, \texttt{stressTable()}, \texttt{tableColbases()}, \texttt{tableDistances()}, \texttt{triangulationBlobs()}, \texttt{unstableMaps}

Other functions relating to map stress calculation: \texttt{logtiterTable()}, \texttt{mapResiduals()}, \texttt{pointStress}, \texttt{recalculateStress()}, \texttt{stressTable()}, \texttt{tableColbases()}, \texttt{tableDistances()}

---

mapGadget  \hspace{1cm} \textit{Open a shiny gadget to view the map}

Description

This function is equivalent to running \texttt{runGUI()} and loading a map file, but this takes the acmap object to open as an input argument.

Usage

mapGadget(map)

Arguments

map \hspace{1cm} The acmap object to open in the GUI

Value

No value returned, called for the side effect of starting the gadget.

See Also

Other functions to view maps: \texttt{RacViewer.options()}, \texttt{RacViewer().export_viewer()}, \texttt{ggplot.acmap()}, \texttt{plot.acmap()}, \texttt{setLegend()}, \texttt{view.acmap()}, \texttt{view.default()}, \texttt{view()}
**mapName**

*Getting and setting the map name*

**Description**

Getting and setting the map name

**Usage**

```r
mapName(map)
mapName(map) <- value
```

**Arguments**

- **map**
  - The acmap data object
- **value**
  - New value to set

**Value**

Returns either the requested attribute when using a getter function or
the updated acmap object when using the setter function.

**See Also**

Other map attribute functions: `acmapAttributes`, `adjustedLogTiterTable()`, `adjustedTiterTable()`, `dilutionStepsize()`, `logTiterTableLayers()`, `mapDescription()`, `titerTableLayers()`, `titerTable()`

**mapRelaxed**

*Check if a map has been fully relaxed*

**Description**

Checks if the map optimization run can be relaxed further.

**Usage**

```r
mapRelaxed(map, optimization_number = 1, options = list())
```

**Arguments**

- **map**
  - The acmap data object
- **optimization_number**
  - The map optimization number
- **options**
  - List of named optimizer options, see `RacOptimizer.options()`
mapResiduals

Value

Returns TRUE or FALSE

See Also

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs(), unstableMaps

mapResiduals Get a table of residuals from an acmap

Description

This is the difference between the table distance and the map distance

Usage

mapResiduals(map, exclude_nd = FALSE, optimization_number = 1)

Arguments

map The acmap object
exclude_nd Should values associated with non-detectable measurements like <10 be set to NA
optimization_number The optimization number

Value

Returns a matrix of residuals, showing the residual error between map distance and table distance for each antigen-sera pair.

See Also

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs(), unstableMaps

Other functions relating to map stress calculation: logtiterTable(), mapDistances(), pointStress, recalculateStress(), stressTable(), tableColbases(), tableDistances()
mapStress

Calculate the current map stress

Description

Calculate the current map stress

Usage

mapStress(map, optimization_number = 1)

Arguments

<table>
<thead>
<tr>
<th>map</th>
<th>The acmap object</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimization_number</td>
<td>The optimization number for which to calculate stress</td>
</tr>
</tbody>
</table>

Value

A number giving the map stress

See Also

Other map optimization attribute functions: colBases(), mapComment(), mapDimensions(), mapTransformation(), ptBaseCoords(), ptCoords()

mapTransformation

Reading map transformation data

Description

These functions can be used to query and if necessary set the map transformation and map translation attributes for a given optimization run.

Usage

mapTransformation(map, optimization_number = 1)
mapTransformation(map, optimization_number = 1) <- value
mapTranslation(map, optimization_number = 1)
mapTranslation(map, optimization_number = 1) <- value

Arguments

<table>
<thead>
<tr>
<th>map</th>
<th>The acmap data object</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimization_number</td>
<td>The optimization run from which to get / set the data</td>
</tr>
<tr>
<td>value</td>
<td>New value to set</td>
</tr>
</tbody>
</table>
**matchStrains**

**Value**

Returns either the requested attribute when using a getter function or the updated acmap object when using the setter function.

**See Also**

Other map optimization attribute functions: `colBases()`, `mapComment()`, `mapDimensions()`, `mapStress()`, `ptBaseCoords()`, `ptCoords()`

---

**matchStrains**

*Find matching antigens or sera between 2 maps*

**Description**

Find matching antigens or sera between 2 maps

**Usage**

```r
match_mapAntigens(map1, map2)
match_mapSera(map1, map2)
```

**Arguments**

- `map1` The map to match names from.
- `map2` The map to match names to.

**Value**

Returns the indices of matching strains in map 2, or NA in the position of strains not found.

**See Also**

Other functions to compare maps: `procrustesData()`, `procrustesMap()`, `realignMap()`, `realignOptimizations()`
mergeMaps

Merging maps

Description

Functions to merge together two tables or maps.

Usage

mergeMaps(
  ..., method = "table",
  number_of_dimensions,
  number_of_optimizations,
  minimum_column_basis = "none",
  optimizer_options = list(),
  merge_options = list(),
  verbose = TRUE
)

Arguments

... acmaps to merge provided as either a list, or a series of separate arguments
method The merge method to use, see details.
number_of_dimensions For merging that generates new optimization runs, the number of dimensions.
number_of_optimizations For merging that generates new optimization runs, the number of optimization runs to do.
minimum_column_basis For merging that generates new optimization runs, the minimum column basis to use.
optimizer_options For merging that generates new optimization runs, optimizer settings (see RacOptimizer.options()).
merge_options Options to use when merging titers (see RacMerge.options()).
verbose Should progress messages be output?

Details

Maps can be merged in a number of ways depending upon the desired result.

Method ‘table’: As you would expect, this merges the tables of the two maps but does not attempt to create any new optimizations and any existing optimizations are lost.

Method ‘reoptimized-merge’: This merges the tables and then does a specified number of fresh optimizations from random starting coordinates, ignoring any pre-existing optimization runs. It’s exactly the same as doing a ‘table’ merge and running optimizeMap() on the merged table.
Method 'incremental-merge': This takes the currently selected optimization in the first map and then merges in the additional maps in turn. Each time any points not already found in the first map (or the last map in the incremental merge chain) are randomised and everything is relaxed, this is repeated the specified number of times and the process is repeated.

Method 'frozen-overlay': This fixes the positions of points in each map and tries to best match them simply through re-orientation. Once the best re-orientation is found, points that are in common between the maps are moved to the average position.

Method 'relaxed-overlay': This is the same as the frozen-overlay but points in the resulting map are then allowed to relax.

Method 'frozen-merge': In this version, positions of all points in the first map are fixed and remain fixed, so the original map does not change. The second map is then realigned to the first as closely as possible and then all the new points appearing in the second map are allowed to relax into their new positions. This is a way to merge in new antigens and sera into a map without affecting the first one at all (and was first implemented in lisp).

Value

Returns the merged map object

See Also

Other map merging functions: RacMerge.options(), htmlMergeReport(), mergeReport(), splitTiterLayers()
moveTrappedPoints

Description

Sometimes points in a map optimization run get trapped in local optima, this function tries to combat this by doing a grid search for each point individually moving points if a better optima is found. Note that this only performs grid searches individually so won’t find cases where a group of points are trapped together in a local optima.

Usage

moveTrappedPoints(
  map,
  optimization_number = 1,
  grid_spacing = 0.25,
  max_iterations = 10,
  options = list()
)

Arguments

map
  The acmap data object

optimization_number
  The map optimization number to apply it to

grid_spacing
  Grid spacing in antigenic units of the search grid to use when searching for more optimal positions

max_iterations
  The maximum number of iterations of searching for trapped points then relaxing the map to be performed

options
  List of named optimizer options, see RacOptimizer.options()

Details

The search is iterative, searching for and moving points that are found to be trapped before relaxing the map and searching again, stopping either when no more trapped points are found or max_iterations is reached.

Value

Returns the acmap object with updated coordinates (if any trapped points found)

See Also

Other map optimization functions: RacOptimizer.options(), make.acmap(), optimizeMap(), randomizeCoords(), relaxMapOneStep(), relaxMap()
optimizationProperties

*Get optimization properties*

**Description**

Utility functions to get a vector of all the map optimization properties.

**Usage**

```r
allMapStresses(map)
allMapDimensions(map)
```

**Arguments**

- `map` The acmap object

**Value**

A numeric vector of values

**See Also**

Other functions to work with map optimizations: `keepOptimizations()`, `removeOptimizations()`, `sortOptimizations()`

---

**optimizeAgReactivity**  *Optimize antigen reactivity adjustments*

**Description**

[Experimental]

**Usage**

```r
optimizeAgReactivity(
  map,
  optimization_number = 1,
  reactivity_stress_weighting = 1,
  fixed_ag_reactivities = rep(NA, numAntigens(map)),
  start_pars = rep(0, numAntigens(map)),
  reoptimize = FALSE,
  number_of_optimizations = 100,
  options = list()
)
```
Arguments

- **map**: The acmap object
- **optimization_number**: The optimization number for which to optimize antigen reactivity adjustments
- **reactivity_stress_weighting**: The weighting to apply when calculating how much antigen reactivity changes should additionally contribute to stress in the optimization regime (see details).
- **fixed_ag_reactivities**: A vector of fixed antigen reactivities, use NA values to distinguish the positions you would still like to be optimized.
- **start_pars**: A vector of starting parameters to use for the optimizer, you can still supply starting parameters for antigens listed in `fixed_ag_reactivities` but they will be ignored.
- **reoptimize**: Should the map be reoptimized from scratch (slower but more likely to explore other optima) when testing each reactivity adjustment or simply relaxed from it’s current coordinates (default)
- **number_of_optimizations**: If reoptimizing from scratch, how many optimization runs should be performed each time.
- **options**: A named list of additional options to pass to `RacOptimizer.options()`

Value

The acmap object is returned with antigen reactivity adjustments set to the value calculated in the optimizer. This can be queried with `agReactivityAdjustments()`.

---

**optimizeMap**

*Optimize an acmap*

Description

Take an acmap object with a table of titer data and perform optimization runs to try and find the best arrangement of antigens and sera to represent their antigenic similarity. Optimizations generated from each run with different random starting conditions will be added to the acmap object.

Usage

```r
optimizeMap(
  map,
  number_of_dimensions,
  number_of_optimizations,
  minimum_column_basis = "none",
  fixed_column_bases = NULL,
  titer_weights = NULL,
  sort_optimizations = TRUE,
)```
Arguments

map | The acmap data object
number_of_dimensions | The number of dimensions for the new map
number_of_optimizations | The number of optimization runs to perform
minimum_column_basis | The minimum column basis to use (see details)
fixed_column_bases | A vector of fixed values to use as column bases directly, rather than calculating them from the titer table.
titer_weights | An optional matrix of weights to assign each titer when optimizing
sort_optimizations | Should optimizations be sorted by stress afterwards?
check_convergence | Should a basic check for convergence of lowest stress optimization runs onto a similar solution be performed.
verbose | Should progress messages be reported, see also RacOptimizer.options()
options | List of named optimizer options, see RacOptimizer.options()

Details

This is the core function to run map optimizations. In essence, for each optimization run, points are randomly distributed in n-dimensional space, the L-BFGS gradient-based optimization algorithm is applied to move points into an optimal position. Depending on the map, this may not be a trivial optimization process and results will depend upon the starting conditions so multiple optimization runs may be required. For a full explanation see vignette("intro-to-antigenic-cartography").

Minimum column basis and fixed column bases:

Fixed column bases is a vector of fixed column bases for each sera, where NA is specified (the default) column bases will be calculated according to the minimum_column_basis setting. Again for a full explanation of column bases and what they mean see vignette("intro-to-antigenic-cartography").

Value

Returns the acmap object updated with new optimizations.

See Also

See relaxMap() for optimizing a given optimization starting from its current coordinates.

Other map optimization functions: RacOptimizer.options(), make.acmap(), moveTrappedPoints(), randomizeCoords(), relaxMapOneStep(), relaxMap()
### orderPoints

**Order antigens and sera**

**Description**

Functions to change the order of antigens and sera in a map

**Usage**

```r
orderAntigens(map, order)
orderSera(map, order)
```

**Arguments**

- `map`: The map data object
- `order`: The new order of points

**Value**

An acmap object with points reordered

**See Also**

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.acmap()`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints`, `subsetMap()`

---

### plot.acmap

**Plot an antigenic map**

**Description**

Method for plotting an antigenic map in two dimensions

**Usage**

```r
## S3 method for class 'acmap'
plot(
  x,
  optimization_number = 1,
  xlim = NULL,
  ylim = NULL,
  plot_ags = TRUE,
```
plot.acmap

plot_sr = TRUE,
plot_labels = FALSE,
plot_blobs = TRUE,
point_opacity = "automatic",
show_procrustes = TRUE,
show_error_lines = FALSE,
plot_stress = FALSE,
indicate_outliers = "arrowheads",
grid.col = "grey90",
grid.margin.col = "grey50",
outlier.arrow.col = grid.col,
fill.alpha = 0.8,
outline.alpha = 0.8,
procrustes.lwd = 2,
procrustes.col = "black",
procrustes.arr.type = "triangle",
procrustes.arr.length = 0.2,
procrustes.arr.width = 0.15,
label.offset = 0,
padding = 1,
cex = 1,
margins = rep(0.5, 4),
...

Arguments

x The acmap to plot
optimization_number The optimization number to plot
xlim optional x axis limits
ylim optional y axis limits
plot_ag optional antigens to be plotted
plot_sr optional antigens to be plotted
plot_labels should point labels be plotted, can be true, false or "antigens" or "sera"
plot_blobs logical, should stress blobs be plotted if present
point_opacity Either "automatic" or "fixed". "fixed" fixes point opacity to match those in ptFill() and ptOutline() and will not be altered in procrustes plots or by the fill.alpha and outline.alpha parameters.
show_procrustes logical, should procrustes lines be shown, if present
show_error_lines logical, should error lines be drawn
plot_stress logical, should map stress be plotted in lower left corner
indicate_outliers
how should points outside the plotting region be indicated, either FALSE, for
not shown, "arrowheads" for small arrowheads like in the viewer, or "arrows"
for arrows pointing from the edge of the plot margin, default is "arrowheads".

grid.col grid line color
grid.margin.col grid margin color
outlier.arrow.col outlier arrow color
fill.alpha alpha for point fill
outline.alpha alpha for point outline
procrustes.lwd procrustes arrow line width
procrustes.col procrustes arrow color
procrustes.arr.type procrustes arrow type (see shape::Arrows())
procrustes.arr.length procrustes arrow length (see shape::Arrows())
procrustes.arr.width procrustes arrow width (see shape::Arrows())
label.offset amount by which any point labels should be offset from point coordinates in
fractions of a character width
padding padding at limits of the antigenic map, ignored if xlim or ylim set explicitly
cex point size expansion factor
margins margins in inches for the plot, use NULL for default margins from par("mar")
... additional arguments, not used

Value
Called for the side effect of plotting the map but invisibly returns the map object.

See Also
Other functions to view maps: RacViewer.options(), RacViewer(), export_viewer(), ggplot.acmap(),
mapGadget(), setLegend(), view.acmap(), view.default(), view()

pointStress Get individual point stress

Description
Functions to get stress associated with individual points in a map.
Usage

```r
agStress(map, antigens = TRUE, optimization_number = 1)
srStress(map, sera = TRUE, optimization_number = 1)
srStressPerTiter(map, sera = TRUE, optimization_number = 1)
agStressPerTiter(map, antigens = TRUE, optimization_number = 1)
```

Arguments

- `map` The acmap data object
- `antigens` Which antigens to check stress for, specified by index or name (defaults to all antigens).
- `optimization_number` The optimization number
- `sera` Which sera to check stress for, specified by index or name (defaults to all sera).

Value

A numeric vector of point stresses

See Also

See `mapStress()` for getting the total map stress directly.

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`

Other functions relating to map stress calculation: `logtiterTable()`, `mapDistances()`, `mapResiduals()`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`

---

**procrustesData**

Return procrustes data on a map comparison

Description

Returns information about how similar point positions are in two maps, to get an idea of how similar antigenic positions are in for example maps made from two different datasets.
Usage

procrustesData(
  map,
  comparison_map,
  optimization_number = 1,
  comparison_optimization_number = 1,
  antigens = TRUE,
  sera = TRUE,
  translation = TRUE,
  scaling = FALSE
)

Arguments

map The acmap data object
comparison_map The acmap data object to procrustes against
optimization_number The map optimization to use in the procrustes calculation (other optimization runs are discarded)
comparison_optimization_number The optimization run int the comparison map to compare against
antigens Antigens to include (specified by name or index or TRUE/FALSE for all/none)
sera Sera to include (specified by name or index or TRUE/FALSE for all/none)
translation Should translation be allowed
scaling Should scaling be allowed (generally not recommended unless comparing maps made with different assays)

Value

Returns a list with information on antigenic distances between the aligned maps, and the rmsd of the point differences split by antigen points, serum points and total, or all points. The distances are a vector matching the number of points in the main map, with NA in the position of any points not found in the comparison map.

See Also

Other functions to compare maps: matchStrains, procrustesMap(), realignMap(), realignOptimizations()
Usage

procrustesMap(
  map,
  comparison_map,
  optimization_number = 1,
  comparison_optimization_number = 1,
  antigens = TRUE,
  sera = TRUE,
  translation = TRUE,
  scaling = FALSE,
  keep_optimizations = FALSE
)

Arguments

map               The acmap data object
comparison_map    The acmap data object to procrustes against
optimization_number The map optimization to use in the procrustes calculation (other optimization
                     runs are discarded)
comparison_optimization_number The optimization run int the comparison map to compare against
antigens           Antigens to include (specified by name or index or TRUE/FALSE for all/none)
sera               Sera to include (specified by name or index or TRUE/FALSE for all/none)
translation        Should translation be allowed
scaling             Should scaling be allowed (generally not recommended unless comparing maps
                     made with different assays)
keep_optimizations  Should all optimization runs be kept or only the one to which the procrustes was
                     applied.

Value

Returns an acmap object with procrustes information added, which will be shown when the map is
plotted. To avoid ambiguity about which optimization run the procrustes was applied to, only the
optimization run specified by optimization_number is kept in the map returned.

See Also

Other functions to compare maps: matchStrains, procrustesData(), realignMap(), realignOptimizations()
**ptAnnotations**  
*Getting and setting point annotation information*

**Description**  
Getting and setting point annotation information

**Usage**

```r
agAnnotations(map)
srAnnotations(map)
agAnnotations(map) <- value
srAnnotations(map) <- value
```

**Arguments**

- `map`: The acmap data object
- `value`: A list of character vectors with annotations information for each point

**Value**

A character vector of point annotations.

**See Also**

Other antigen and sera attribute functions: `agAttributes`, `agGroups()`, `agHomologousSr()`, `agLabIDs()`, `agSequences()`, `ptClades`, `srAttributes`, `srGroups()`, `srHomologousAgs()`, `srSequences()`

**ptBaseCoords**  
*Getting and setting base coordinates*

**Description**

These functions get and set the base coordinates for a given optimization run.

**Usage**

```r
ptBaseCoords(map, optimization_number = 1)
agBaseCoords(map, optimization_number = 1)
agBaseCoords(map, optimization_number = 1) <- value
srBaseCoords(map, optimization_number = 1)
srBaseCoords(map, optimization_number = 1) <- value
```
Get antigen or serum bootstrap blob information

Description
Get antigen or serum bootstrap blob information for plotting with the blob() function.

Usage

\begin{verbatim}
agBootstrapBlob(map, antigen, optimization_number = 1)
srBootstrapBlob(map, serum, optimization_number = 1)
agBootstrapBlobs(map, optimization_number = 1)
srBootstrapBlobs(map, optimization_number = 1)
ptBootstrapBlobs(map, optimization_number = 1)
\end{verbatim}

Arguments

- **map**: An acmap object
- **antigen**: The antigen to get the blob for
- **optimization_number**: Optimization number from which to get blob information
- **serum**: The serum to get the blob for
Value

Returns an object of class "blob" that can be plotted using the `blob()` function.

See Also

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableColbases()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`
ptClades  Getting and setting point clade information

Description

Getting and setting point clade information

Usage

agClades(map)

srClades(map)

agClades(map) <- value

srClades(map) <- value

Arguments

map  The acmap data object

value  A list of character vectors with clade information for each point

Value

A character vector of clade information.

See Also

Other antigen and sera attribute functions: agAttributes, agGroups(), agHomologousSr(), agLabIDs(), agSequences(), ptAnnotations, srAttributes, srGroups(), srHomologousAgs(), srSequences()

ptCoords  Getting and setting point coordinates

Description

Getting and setting of antigen and serum coordinates in a map optimization run (by default the currently selected one).
ptCoords

Usage

agCoords(map, optimization_number = 1)
srCoords(map, optimization_number = 1)
ptCoords(map, optimization_number = 1)

ptCoords(map, optimization_number = 1) <- value

agCoords(map, optimization_number = 1) <- value

srCoords(map, optimization_number = 1) <- value

Arguments

map The acmap object
optimization_number The optimization number from which to get / set the coordinates
value A matrix of new coordinates to set

Details

These functions get and set point coordinates in a map. By default these coordinates refer to the currently selected optimization run, unless otherwise specified through the optimization_number argument.

99% want to use but you should note that the outputs are actually the map base coordinates after the transformation and translation associated with the optimization run has been applied (see mapTransformation() and mapTranslation() for more details). When you set the antigen or serum coordinates through these functions, the transformed coordinates are "baked" in and the map transformation and translation are reset. Consequently if you want to apply a transformation to all coordinates generally, you are better off modifying the map translation and transformation directly, as is done by functions like rotateMap() and translateMap().

Value

Returns a matrix of point coordinates.

See Also

agBaseCoords() srBaseCoords() mapTransformation() mapTranslation()

Other map optimization attribute functions: colBases(), mapComment(), mapDimensions(), mapStress(), mapTransformation(), ptBaseCoords()
ptDrawingOrder

Get and set point drawing order in map

Description

Point drawing order is a vector of indices defining the order in which points should be draw when plotting or viewing a map. Points are indexed in the same order as antigens then followed by sera.

Usage

ptDrawingOrder(map)

ptDrawingOrder(map) <- value

Arguments

map An acmap object
value The point drawing order

Value

A numeric vector of point drawing order information

See Also

Other map point style functions: applyPlotspec(), ptOpacity, ptStyles

ptLeverage

Calculate point leverage

Description

These functions attempt to estimate leverage of each antigen, sera or titer by removing it from the data, relaxing the map, then calculating the rmsd of the procrustes comparison between the original and newly relaxed map. Column bases will be recalculated unless you have specified them as fixed with fixedColBases().

Usage

agLeverage(map, antigens = TRUE, sera = TRUE)
srLeverage(map, antigens = TRUE, sera = TRUE)
titerLeverage(map, antigens = TRUE, sera = TRUE)
Arguments

- **map**: An acmap object
- **antigens**: Antigens to include when calculating the rmsd of the procrustes (specified by name or index or TRUE/FALSE for all/none)
- **sera**: Sera to include when calculating the rmsd of the procrustes (specified by name or index or TRUE/FALSE for all/none)

Value

Returns a numeric vector of the leverage calculated for each of the points.

See Also

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`,
ptStyles  Getting and setting point plotting styles

Description
These functions get and set the styles to use for each point when plotting.

Usage

\begin{verbatim}
agShown(map)
srShown(map)
agShown(map) <- value
srShown(map) <- value
agSize(map)
srSize(map)
agSize(map) <- value
srSize(map) <- value
agFill(map)
srFill(map)
agFill(map) <- value
srFill(map) <- value
agOutline(map)
srOutline(map)
agOutline(map) <- value
srOutline(map) <- value
agOutlineWidth(map)
srOutlineWidth(map)
agOutlineWidth(map) <- value
srOutlineWidth(map) <- value
agRotation(map)
srRotation(map)
agRotation(map) <- value
srRotation(map) <- value
agAspect(map)
srAspect(map)
agAspect(map) <- value
srAspect(map) <- value
agShape(map)
srShape(map)
agShape(map) <- value
srShape(map) <- value
\end{verbatim}

Arguments

- **map** The acmap data object
- **value** New value to set
\textbf{ptTriangulationBlob}  

\textbf{Value}  

Returns either the requested attribute when using a getter function or the updated acmap object when using the setter function.

\textbf{See Also}  

Other map point style functions: \texttt{applyPlotspec()}, \texttt{ptDrawingOrder()}, \texttt{ptOpacity}

---

\textbf{ptTriangulationBlob}  \textit{Get antigen or serum triangulation blob information}

\textbf{Description}  

Get antigen or serum triangulation blob information for plotting with the \texttt{blob()} function.

\textbf{Usage}  

\begin{verbatim}
agTriangulationBlob(map, antigen, optimization_number = 1)
srTriangulationBlob(map, serum, optimization_number = 1)
agTriangulationBlobs(map, optimization_number = 1)
srTriangulationBlobs(map, optimization_number = 1)
ptTriangulationBlobs(map, optimization_number = 1)
\end{verbatim}

\textbf{Arguments}  

map \hspace{1cm} An acmap object  
antigen \hspace{1cm} The antigen to get the blob for  
optimization_number \hspace{1cm} Optimization number from which to get blob information  
serum \hspace{1cm} The serum to get the blob for

\textbf{Value}  

Returns an object of class "blob" that can be plotted using the \texttt{blob()} function.

\textbf{See Also}  

Other map diagnostic functions: \texttt{agCohesion()}, \texttt{bootstrapBlobs()}, \texttt{bootstrapMap()}, \texttt{checkHemisphering()}, \texttt{dimensionTestMap()}, \texttt{logtiterTable()}, \texttt{map-table-distances}, \texttt{mapBootstrapCoords}, \texttt{mapDistances()}, \texttt{mapRelaxed()}, \texttt{mapResiduals()}, \texttt{pointStress}, \texttt{ptBootstrapBlob}, \texttt{ptBootstrapCoords()}, \texttt{ptLeverage}, \texttt{recalculateStress()}, \texttt{stressTable()}, \texttt{tableColbases()}, \texttt{tableDistances()}, \texttt{triangulationBlobs()}, \texttt{unstableMaps}
RacMerge.options

Set acmap merge options

Description

This function facilitates setting options for the acmap titer merging process by returning a list of option settings.

Usage

RacMerge.options(sd_limit = NULL, dilution_stepsize = 1, method = NULL)

Arguments

sd_limit

When merging titers, titers that have a standard deviation of this amount or greater on the log2 scale will be set to "*" and excluded. Setting this to NA removes any limit. The default value will be NA, unless the titer merge method is specified as "lispmds" in which case the default is 1 and standard deviation is calculated by division by n, instead of n-1, in order to maintain backwards compatibility with previous approaches.

dilution_stepsize

The dilution stepsize to assume when merging titers (see dilutionStepsize())

method

The titer merging method to use, either a string of "conservative" or "likelihood", or a user defined function. See details.

Details

When merging measured titers, the general approach is to take the geometric mean and use that as the merged titer, however in particular when < values are present there are different options that can be employed. In older versions of Racmacs, < values were converted to maximum possible numeric titer after accounting for the dilution_stepsize factor, then the geometric mean was taken. This approach can be used by specifying the method as "likelihood" since, this approach gives a very rough approximation of the most likely mean numeric value. In contrast, the "conservative" method and current default returns the highest < value that satisfies all the values that were measured. As an example merging <10 and 20, (assuming dilution_stepsize = 1) would return a value of 10 with the "likelihood" method and <40 with the "conservative" method.

Value

Returns a named list of merging options

See Also

Other map merging functions: htmlMergeReport(), mergeMaps(), mergeReport(), splitTiterLayers()
RacOptimizer.options  

Set acmap optimization options

Description
This function facilitates setting options for the acmap optimizer process by returning a list of option settings.

Usage

RacOptimizer.options(  
dim_annealing = FALSE,  
method = “L-BFGS”,  
maxit = 1000,  
num_basis = 10,  
armijo_constant = 1e-04,  
wolfe = 0.9,  
min_gradient_norm = 1e-06,  
factr = 1e-15,  
max_line_search_trials = 50,  
min_step = 1e-20,  
max_step = 1e+20,  
num_cores = getOption("RacOptimizer.num_cores"),  
report_progress = NULL,  
ignore_disconnected = FALSE,  
progress_bar_length = options()$width  
)

Arguments

dim_annealing  Should dimensional annealing be performed
method     The optimization method to use
maxit     The maximum number of iterations to use in the optimizer
num_basis     Number of memory points to be stored (default 10).
armijo_constant     Controls the accuracy of the line search routine for determining the Armijo condition.
wolfe     Parameter for detecting the Wolfe condition.
min_gradient_norm     Minimum gradient norm required to continue the optimization.
factr     Minimum relative function value decrease to continue the optimization.
max_line_search_trials     The maximum number of trials for the line search (before giving up).
min_step     The minimum step of the line search.
max_step  The maximum step of the line search.
num_cores The number of cores to run in parallel when running optimizations
report_progress Should progress be reported
ignore_disconnected Should the check for disconnected points be skipped
progress_bar_length Progress bar length when progress is reported

Details
For more details, for example on "dimensional annealing" see vignette("intro-to-antigenic-cartography"). For details on optimizer settings like maxit see the underlying optimizer documentation at ens-mallen.org.

Value
Returns a named list of optimizer options

See Also
Other map optimization functions: make.acmap(), moveTrappedPoints(), optimizeMap(), randomizeCoords(), relaxMapOneStep(), relaxMap()
Arguments

- **map**
  - The map data object
- **show_procrustes**
  - Should procrustes lines be shown
- **show_group_legend**
  - Show an interactive legend detailing different groups as set by `agGroups()` and `srGroups()`
- **options**
  - A named list of viewer options supplied to `racviewer.options()`
- **width**
  - Width of the widget
- **height**
  - Height of the widget
- **elementId**
  - DOM element ID

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

Other functions to view maps: `RacViewer.options()`, `export_viewer()`, `ggplot.acmap()`, `mapGadget()`, `plot.acmap()`, `setLegend()`, `view.acmap()`, `view.default()`, `view()`

Description

Output and render functions for using RacViewer within Shiny applications and interactive Rmd documents.

Usage

RacViewerOutput(outputId, width = "100\%", height = "100\%")

renderRacViewer(expr, env = parent.frame(), quoted = FALSE)

Arguments

- **outputId**
  - output variable to read from
- **width, height**
  - Must be a valid CSS unit (like '100\%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
- **expr**
  - An expression that generates a RacViewer
- **env**
  - The environment in which to evaluate expr.
- **quoted**
  - Is expr a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.
RacViewer.options

Value
An output or render function that enables the use of the widget within Shiny applications.

See Also
Other shiny app functions: runGUI(), view.acmap()

RacViewer.options  Set viewer options

Description
This function facilitates setting racviewer options by returning a list of option settings.

Usage
RacViewer.options(
  point.opacity = NA,
  viewer.controls = "hidden",
  grid.display = "static",
  grid.col = "#cfcfcf",
  background.col = "#ffffff",
  show.names = FALSE,
  show.errorlines = FALSE,
  show.connectionlines = FALSE,
  show.titers = FALSE,
  xlim = NULL,
  ylim = NULL,
  translation = c(0, 0, 0),
  rotation = c(0, 0, 0),
  zoom = NULL
)

Arguments
point.opacity  Default opacity for unselected points, or "inherit" to take opacity from the color values themselves.
viewer.controls  Should viewer controls be shown or hidden by default?
grid.display  For 3d maps, should the grid be fixed in the background or enclose and rotate along with the map
grid.col  Color to use for the grid shown behind the map
background.col  Color for the viewer background
show.names  Toggle name labels on, can be true or false or "antigens" or "sera"
show.errorlines  Toggle error lines on
randomizeCoords

Description

Moves map coordinates back into random starting conditions, as performed before each optimization run. The maximum table distance is calculated then points are randomized in a box with side length equal to maximum table distance multiplied by table_dist_factor.

Usage

randomizeCoords(map, optimization_number = 1, table_dist_factor = 2)

Arguments

map
The acmap data object
optimization_number
The map optimization number to randomize
table_dist_factor
The expansion factor for the box size in which points are randomized.

Value

Returns an updated map object

See Also

Other map optimization functions: RacOptimizer.options(), make.acmap(), moveTrappedPoints(), optimizeMap(), relaxMapOneStep(), relaxMap()
**Description**

Reads an antigenic map file and converts it into an acmap data object.

**Usage**

```r
read.acmap(
  filename,
  optimization_number = NULL,
  sort_optimizations = FALSE,
  align_optimizations = FALSE
)
```

**Arguments**

- `filename` Path to the file.
- `optimization_number` Numeric vector of optimization runs to keep, the default, NULL, keeps information on all optimization runs.
- `sort_optimizations` Should optimizations be sorted in order of stress when the map data is read?
- `align_optimizations` Should optimizations be rotated and translated to match the orientation of the first optimization as closely as possible?

**Value**

Returns the acmap data object.

**See Also**

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints()`, `read.titerTable()`, `removePoints()`, `save.acmap()`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints()`, `subsetMap()`
read.titerTable  

Read in a table of titer data

Description

Reads in a table of titer data, converting it to a matrix of titers with labelled column and row names. Missing titers should be represented by an asterisk character.

Usage

read.titerTable(filepath)

Arguments

filepath Path to the table of titer data

Details

Currently supported file formats are .csv and .xls and .txt

Value

Returns a matrix of titers.

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints, read.acmap(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetCommonPoints, subsetMap()

realignMap  

Realign map to match another

Description

Realigns the coordinates of a map to match a target map as closely as possible, based on a procrustes analysis. Note that all optimization runs will be separately aligned to match as closely as possible the first optimization run of the target map.

Usage

realignMap(map, target_map, translation = TRUE, scaling = FALSE)
Arguments

map The acmap to realign.
target_map The acmap to realign to.
translation Should translation be allowed
scaling Should scaling be allowed (generally not recommended unless comparing maps made with different assays)

Value

Returns a map object aligned to the target map

See Also

Other functions to compare maps: `matchStrains`, `procrustesData()`, `procrustesMap()`, `realignOptimizations()`
recalculateStress

Recalculate the stress associated with an acmap optimization

Description
Recalculates the stress associated with the currently selected or user-specified optimization.

Usage
recalculateStress(map, optimization_number = 1)

Arguments
map The acmap data object
optimization_number The optimization number

Value
Returns the recalculated map stress for a given optimization

See Also
See pointStress() for getting the stress of individual points.

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(),
dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(),
mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage,
ptTriangulationBlob, stressTable(), tableColbases(), tableDistances(), triangulationBlobs(),
unstableMaps

Other functions relating to map stress calculation: logtiterTable(), mapDistances(), mapResiduals(),
pointStress, stressTable(), tableColbases(), tableDistances()

reflactMap

Reflect a map

Description
Reflects map coordinates

Usage
reflectMap(map, axis = "x", optimization_number = NULL)
**relaxMap**

**Arguments**

- `map` The acmap object
- `axis` Axis of reflection
- `optimization_number` The optimization number (or NULL to apply to all optimizations)

**Value**

An acmap object with reflection applied

**See Also**

Other functions relating to map transformation: `applyMapTransform()`, `rotateMap()`, `translateMap()`

---

**Description**

Optimize antigen and serum positions starting from their current coordinates in the selected or specified optimization.

**Usage**

```r
relaxMap(
  map,
  optimization_number = 1,
  fixed_antigens = FALSE,
  fixed_sera = FALSE,
  titer_weights = NULL,
  options = list()
)
```

**Arguments**

- `map` The acmap object
- `optimization_number` The optimization number to relax
- `fixed_antigens` Antigens to set fixed positions for when relaxing
- `fixed_sera` Sera to set fixed positions for when relaxing
- `titer_weights` An optional matrix of weights to assign each titer when optimizing
- `options` List of named optimizer options, see `RacOptimizer.options()`

**Value**

Returns an acmap object with the optimization relaxed.
relaxMapOneStep

**See Also**

See optimizeMap() for performing new optimization runs from random starting coordinates.

Other map optimization functions: RacOptimizer.options(), make.acmap(), moveTrappedPoints(), optimizeMap(), randomizeCoords(), relaxMapOneStep()

---

relaxMapOneStep  
Relax a map one step in the optimiser

**Description**

Relax a map one step in the optimiser

**Usage**

relaxMapOneStep(
  map,
  optimization_number = 1,
  fixed_antigens = FALSE,
  fixed_sera = FALSE,
  options = list()
)

**Arguments**

- **map**  
The acmap data object
- **optimization_number**  
The map optimization number
- **fixed_antigens**  
Antigens to set fixed positions for when relaxing
- **fixed_sera**  
Sera to set fixed positions for when relaxing
- **options**  
List of named optimizer options, see RacOptimizer.options()

**Value**

Returns an updated map object

**See Also**

Other map optimization functions: RacOptimizer.options(), make.acmap(), moveTrappedPoints(), optimizeMap(), randomizeCoords(), relaxMap()
removeOptimizations  
*Remove map optimizations*

**Description**
Remove all optimization run data from a map object

**Usage**
```
removeOptimizations(map)
```

**Arguments**
- `map`  
The acmap object

**Value**
An acmap object with all optimizations removed

**See Also**
Other functions to work with map optimizations: `keepOptimizations()`, `optimizationProperties()`, `sortOptimizations()`

removePoints  
*Remove antigens and sera*

**Description**
Functions to remove antigens and sera from a map

**Usage**
```
removeAntigens(map, antigens)
removeSera(map, sera)
```

**Arguments**
- `map`  
The map data object
- `antigens`  
Antigens to remove (specified by name or index)
- `sera`  
Sera to remove (specified by name or index)

**Value**
An acmap object with points removed
rotateMap

Rotate a map

Description

Apply a rotation to an antigenic map

Usage

rotateMap(map, degrees, axis = NULL, optimization_number = NULL)

Arguments

map
The acmap object
degrees
Degrees of rotation
axis
Axis of rotation (if 3D), specified as "x", "y", or "z"
optimization_number
The optimization number (or NULL to apply to all optimizations)

Value

An acmap object with rotation applied

See Also

Other functions relating to map transformation: applyMapTransform(), reflectMap(), translateMap()

runGUI

Open the Racmacs GUI

Description

This function opens the Racmacs GUI in a new window

Usage

runGUI()
Value

Nothing returned, called only for the side effect of starting the viewer.

See Also

Other shiny app functions: `RacViewer-shiny.view.acmap()`

---

**save.acmap**  
Save acmap data to a file

Description

Save acmap data to a file. The preferred extension is "ace", although the format of the file will be a json file of map data compressed using `xz` compression.

Usage

```r
save.acmap(
  map,  
  filename,
  compress = FALSE,
  pretty = !compress,
  round_titers = FALSE
)
```

Arguments

- `map`  
  The acmap data object.

- `filename`  
  Path to the file.

- `compress`  
  Should the file be xz compressed

- `pretty`  
  Should json be output prettily with new lines and indentation

- `round_titers`  
  Should titers be rounded when outputted (this is needed for acmacs web and lispmds compatibility)

Value

No return value, called for the side effect of saving the map data to the file.

See Also

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.coords()`, `save.titerTable()`, `subsetCommonPoints`, `subsetMap()`
save.coords

Save acmap coordinate data to a file

Description

Saves acmap coordinate data of all or specified antigens and sera to a .csv file.

Usage

```r
save.coords(
  map,
  filename,
  optimization_number = 1,
  antigens = TRUE,
  sera = TRUE
)
```

Arguments

- **map**: The acmap data object.
- **filename**: Path to the file.
- **optimization_number**: Optimization number from which to take coordinates
- **antigens**: Antigens to include, either as a numeric vector of indices or character vector of names.
- **sera**: Sera to include, either as a numeric vector of indices or character vector of names.

Value

No return value, called for the side effect of saving the coordinate data.

See Also

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.acmap()`, `save.titerTable()`, `subsetCommonPoints`, `subsetMap()`
save.titerTable  

Save titer data to a file

Description

Saves titer data of all or specified antigens and sera to a .csv file.

Usage

```
save.titerTable(map, filename, antigens = TRUE, sera = TRUE)
```

Arguments

- `map`: The acmap data object.
- `filename`: Path to the file.
- `antigens`: Antigens to include, either as a numeric vector of indices or character vector of names.
- `sera`: Sera to include, either as a numeric vector of indices or character vector of names.

Value

No return value, called for the side effect of saving the titer data to the file.

See Also

Other functions for working with map data: `acmap()`, `addOptimization()`, `agReactivityAdjustments()`, `as.json()`, `edit_agNames()`, `edit_srNames()`, `keepBestOptimization()`, `keepSingleOptimization()`, `layerNames()`, `orderPoints`, `read.acmap()`, `read.titerTable()`, `removePoints`, `save.acmap()`, `save.coords()`, `subsetCommonPoints`, `subsetMap()`

---

setLegend  

Set acmap legend

Description

This sets the acmap legend used when viewing a map for example.

Usage

```
setLegend(map, legend, fill, style.bottom = "8px", style.right = "8px")
```
sortOptimizations

Arguments

- **map**: The acmap object
- **legend**: A character vector of legend labels
- **fill**: The fill color to be used with the boxes that appear alongside the legend labels
- **style.bottom**: "bottom" style of the div, specifying how far from the bottom of the viewport the bottom of the legend is spaced.
- **style.right**: "right" style of the div, specifying how far from the right of the viewport the bottom of the legend is spaced.

Value

Returns the updated acmap object

See Also

Other functions to view maps: RacViewer.options(), RacViewer(), export_viewer(), ggplot.acmap(), mapGadget(), plot.acmap(), view.acmap(), view.default(), view()

---

**sortOptimizations**  
*Sort optimizations by stress*

Description

Sorts all the optimization runs for a given map object by stress (lowest to highest). Note that this is done by default when running optimizeMap().

Usage

sortOptimizations(map)

Arguments

- **map**: The acmap object

Value

An acmap object with optimizations sorted by stress.

See Also

Other functions to work with map optimizations: keepOptimizations(), optimizationProperties(), removeOptimizations()
**splitTiterLayers**

*Split a map made up from titer layers into a list of separate maps each with a titer table corresponding to one of the layers*

**Description**

Split a map made up from titer layers into a list of separate maps each with a titer table corresponding to one of the layers

**Usage**

```r
splitTiterLayers(map)
```

**Arguments**

- `map` An acmap object with titer table layers

**See Also**

Other map merging functions: `RacMerge.options()`, `htmlMergeReport()`, `mergeMaps()`, `mergeReport()`

---

**srAttributes**

*Getting and setting sera attributes*

**Description**

These functions get and set the sera attributes for a map.

**Usage**

```r
srIDs(map)
srIDs(map) <- value
srDates(map)
srDates(map) <- value
srReference(map)
srReference(map) <- value
srNames(map)
srNames(map) <- value
srExtra(map)
srExtra(map) <- value
srPassage(map)
srPassage(map) <- valuesrLineage(map)
srLineage(map) <- valuesrReassortant(map)
srReassortant(map) <- value
```
srGroups

srStrings(map)
srStrings(map) <- value
srSpecies(map)
srSpecies(map) <- value

Arguments

map The acmap data object
value New value to set

Value

Returns either the requested attribute when using a getter function or the updated acmap object when using the setter function.

See Also

agAttributes()
Other antigen and sera attribute functions: agAttributes, agGroups(), agHomologousSr(), agLabIDs(), agSequences(), ptAnnotations, ptClades, srGroups(), srHomologousAgs(), srSequences()

srGroups

Getting and setting sera groups

Description

These functions get and set the sera groupings for a map.

Usage

srGroups(map)
srGroups(map) <- value

Arguments

map The acmap object
value A character or factor vector of groupings to apply to the sera

Value

A factor vector of serum groups

See Also

Other antigen and sera attribute functions: agAttributes, agGroups(), agHomologousSr(), agLabIDs(), agSequences(), ptAnnotations, ptClades, srAttributes, srHomologousAgs(), srSequences()
srHomologousAgs

*Get and set homologous antigens for sera*

**Description**

Get and set indices of homologous antigens to sera in an antigenic map

**Usage**

srHomologousAgs(map)

srHomologousAgs(map) <- value

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>An acmap object</td>
</tr>
<tr>
<td>value</td>
<td>A list, where each entry is a vector of indices for homologous antigens, or a length 0 vector where no homologous antigen is present</td>
</tr>
</tbody>
</table>

**Value**

A list, where each entry is a vector of indices for homologous antigens, or a length 0 vector where no homologous antigen is present.

**See Also**

Other antigen and sera attribute functions: agAttributes(), agGroups(), agHomologousSr(), agLabIDs(), agSequences(), ptAnnotations, ptClades, srAttributes, srGroups(), srSequences()

srSequences

*Getting and setting sera sequence information*

**Description**

Getting and setting sera sequence information

**Usage**

srSequences(map, missing_value = ".")

srSequences(map) <- value

srNucleotideSequences(map, missing_value = ".")

srNucleotideSequences(map) <- value
standardizeStrainNames

Arguments

map  The acmap data object
missing_value  Character to use to fill in portions of the sequence matrix where sequence data is missing.
value  A character matrix of sequences with rows equal to the number of sera

Value

A character matrix of sequences with rows equal to the number of sera.

See Also

Other antigen and sera attribute functions: agAttributes, agGroups(), agHomologousSr(), agLabIDs(), agSequences(), ptAnnotations, ptClades, srAttributes, srGroups(), srHomologousAgs()

standardizeStrainNames

Standardize strain names

Description

This is a utility function to help standardise antigen names into a more consistent format, also attempting to break apart different components of the name.

Usage

standardizeStrainNames(
  names,
  default_species = NA,
  default_virus_type = "A",
  default_virus_subtype = "H3N2"
)

Arguments

names  Strain names to be standardised
default_species  Are the strains isolated from a particular species?
default_virus_type  Default virus type to be used (if no type found in name)
default_virus_subtype  Default virus subtype to be used (if no subtype found in name)

Value

Returns a tibble of standardised names and extracted information
stressTable

Get a stress table from an acmap

Description
Get a stress table from an acmap

Usage
stressTable(map, optimization_number = 1)

Arguments
map
   The acmap object
optimization_number
   The optimization number for which to calculate stresses

Value
Returns a matrix of stresses, showing how much each antigen and sera measurement contributes to stress in the selected or specified optimization.

See Also
Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), tableColbases(), tableDistances(), triangulationBlobs(), unstableMaps
Other functions relating to map stress calculation: logtiterTable(), mapDistances(), mapResiduals(), pointStress, recalculateStress(), tableColbases(), tableDistances()

subsetCommonPoints

Remove antigens and sera

Description
Functions to subset a list of maps to include only antigens, antigen groups, sera or serum groups that are in common between them.

Usage
subsetCommonAgs(maps)
subsetCommonSrGroups(maps)
subsetMap

Arguments

maps A list of map data objects

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints, read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetMap()

subsetMap Subset an antigenic map

Description

Subset an antigenic map to contain only specified antigens and sera

Usage

subsetMap(map, antigens = TRUE, sera = TRUE)

Arguments

map The antigenic map object
antigens Antigens to keep, defaults to all.
sera Sera to keep, defaults to all.

Value

Returns a new antigenic map containing only match antigens and sera

See Also

Other functions for working with map data: acmap(), addOptimization(), agReactivityAdjustments(), as.json(), edit_agNames(), edit_srNames(), keepBestOptimization(), keepSingleOptimization(), layerNames(), orderPoints, read.acmap(), read.titerTable(), removePoints, save.acmap(), save.coords(), save.titerTable(), subsetMap()
tableColbases

**Calculate column bases for a titer table**

**Description**

For more information on column bases, what they mean and how they are calculated see vignette("intro-to-antigenic-cartography")

**Usage**

```r
tableColbases(
  titer_table,  # The titer table
  minimum_column_basis = "none",  # The minimum column basis to assume
  fixed_column_bases = rep(NA, ncol(titer_table)),  # Fixed column bases to apply
  ag_reactivity_adjustments = rep(0, nrow(titer_table))  # Reactivity adjustments to apply on a per-antigen basis
)
```

**Arguments**

- `titer_table`  The titer table
- `minimum_column_basis`  The minimum column basis to assume
- `fixed_column_bases`  Fixed column bases to apply
- `ag_reactivity_adjustments`  Reactivity adjustments to apply on a per-antigen basis

**Value**

Returns a numeric vector of the log-converted column bases for the table

**See Also**

Other map diagnostic functions: `agCohesion()`, `bootstrapBlobs()`, `bootstrapMap()`, `checkHemisphering()`, `dimensionTestMap()`, `logtiterTable()`, `map-table-distances`, `mapBootstrapCoords`, `mapDistances()`, `mapRelaxed()`, `mapResiduals()`, `pointStress`, `ptBootstrapBlob`, `ptBootstrapCoords()`, `ptLeverage`, `ptTriangulationBlob`, `recalculateStress()`, `stressTable()`, `tableDistances()`, `triangulationBlobs()`, `unstableMaps`

Other functions relating to map stress calculation: `logtiterTable()`, `mapDistances()`, `mapResiduals()`, `pointStress`, `recalculateStress()`, `stressTable()`, `tableDistances()`
tableDistances

Return calculated table distances for an acmap

Description
Takes the acmap object and, assuming the column bases associated with the currently selected or specified optimization, returns the table distances calculated from the titer data. For more information on column bases and their role in antigenic cartography see vignette("intro-to-antigenic-cartography")

Usage
tableDistances(map, optimization_number = 1)

Arguments
map The acmap data object
optimization_number The optimization number

Value
Returns a matrix of numeric table distances

See Also
Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), triangulationBlobs(), unstableMaps

Other functions relating to map stress calculation: logtiterTable(), mapDistances(), mapResiduals(), pointStress, recalculateStress(), stressTable(), tableColbases()

titerTable

Getting and setting map titers

Description
Functions to get and set the map titer table. Note that when setting the titer table like this any titer table layer information is lost, this is normally not a problem unless the map is a result of merging two titer tables together previously and you then go on the merge the titers again.

These are underlying functions to get and set the "flat" version of the titer table only. When a map is merged, the titer tables are merged but a record of the original titers associated with each map are kept as titer table layers so that information on the original set of titers that made up the merge is not lost. At the same time, the merged titer version of the titer table is created and saved as
the titer_table_flat attribute. When you access titers through the `titerTable()` function, the flat version of the titer table is retrieved (only really a relevant distinction for merged maps). When you set titers through `titerTable<-()` titer table layers are lost. These functions allow you to manipulate the flat version without effecting the titer table layers information.

**Usage**

```r
> titerTable(map)
> titerTable(map) <- value
> titerTableFlat(map)
> titerTableFlat(map) <- value
```

**Arguments**

- `map`: The acmap object
- `value`: A character matrix of titers to set

**Value**

Returns a character matrix of titers.

**See Also**

`adjustedTiterTable()`, `htmlTiterTable()`

Other map attribute functions: `acmapAttributes`, `adjustedLogTiterTable()`, `adjustedTiterTable()`, `dilutionStepsize()`, `logtiterTableLayers()`, `mapDescription()`, `mapName()`, `titerTableLayers()`

---

**titerTableLayers**

**Getting and setting titer table layers**

**Description**

Functions to get and set the underlying titer table layers of a map (see details).

**Usage**

```r
> titerTableLayers(map)
> titerTableLayers(map) <- value
```

**Arguments**

- `map`: The acmap object
- `value`: A list of titer table character vectors to set
translateMap

Details

When you merge maps with mergeMaps() repeated antigen - serum titers are merged to create a new titer table but information on the original titers is not lost. The original titer tables, aligned to their new positions in the merged table, are kept as separate layers that can be accessed with these functions. If you have merged a whole bunch of different maps, these functions can be useful to check for example, variation in titer seen between a single antigen and serum pair.

Value

A list of character matrices of titers.

See Also

Other map attribute functions: acmapAttributes, adjustedLogTiterTable(), adjustedTiterTable(), dilutionStepsize(), logtiterTableLayers(), mapDescription(), mapName(), titerTable()

translateMap Translate a map

Description

Translates map coordinates

Usage

translateMap(map, translation, optimization_number = NULL)

Arguments

map The acmap object
translation Translation to apply (as vector or n x 1 matrix)
optimization_number The optimization number (or NULL to apply to all optimizations)

Value

An acmap object with transformation applied

See Also

Other functions relating to map transformation: applyMapTransform(), reflectMap(), rotateMap()
triangulationBlobs

Calculate triangulation blobs data for an antigenic map

Description

This function is to help give an idea of how well coordinated each point is in a map, and to give some idea of uncertainty in it's position. It works by moving each point in a grid search and seeing how the total map stress changes, see details.

Usage

```r
triangulationBlobs(
  map,
  optimization_number = 1,
  stress_lim = 1,
  grid_spacing = 0.25,
  antigens = TRUE,
  sera = TRUE,
  .check_relaxation = TRUE,
  .options = list()
)
```

Arguments

- `map`: The acmap data object
- `optimization_number`: The optimization number to check
- `stress_lim`: The blob stress limit
- `grid_spacing`: Grid spacing to use when searching map space and inferring the blob
- `antigens`: Should triangulation blobs be calculated for antigens
- `sera`: Should triangulation blobs be calculated for sera
- `.check_relaxation`: Should a check be performed that the map is fully relaxed (all points in a local optima) before the search is performed
- `.options`: List of named optimizer options to use when checking map relaxation, see `RacOptimizer.options()`

Details

The region or regions of the plot where total map stress is not increased above a certain threshold (`stress_lim`) are shown when the map is plotted. This function is really to check whether point positions are clearly very uncertain, for example the underlying titers may support an antigen being a certain distance away from a group of other points but due to the positions of the sera against which it was titrated the direction would be unclear, and you might see a blob that forms an arc or "banana" that represents this. Note that it is not really a confidence interval since a point may be
well coordinated in terms of the optimization but its position may still be defined by perhaps only one particular titer which is itself uncertain. For something more akin to confidence intervals you can use other diagnostic functions like bootstrapMap().

**Value**

Returns the acmap data object with triangulation blob information added, which will be shown when the map is plotted.

**See Also**

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), tableDistances(), unstableMaps

---

**unstableMaps**

Notes on unstable maps

**Description**

Tips for exploring maps that are difficult to find a consistent optimal solution for.

**Details**

Maps may be difficult to optimize or unstable for a variety of reasons, a common one with larger maps being simply that it is difficult to find a global optima and so many different local optima are found each time.

One approach that can sometimes help is to consider running the optimizer with options = list(dim_annnealing = TRUE) (see see vignette("intro-to-antigenic-cartography") for an explanation of the dimensional annealing approach). However be wary that in our experience, while applying dimensional annealing can sometimes significantly speed up finding a better minima, it can also sometimes be more prone to getting stuck in worse local optima.

If there are many missing or non-detectable titers it is also possible that points in map are too poorly connected to find a robust solution, to check this see mapCohesion().

**See Also**

Other map diagnostic functions: agCohesion(), bootstrapBlobs(), bootstrapMap(), checkHemisphering(), dimensionTestMap(), logtiterTable(), map-table-distances, mapBootstrapCoords, mapDistances(), mapRelaxed(), mapResiduals(), pointStress, ptBootstrapBlob, ptBootstrapCoords(), ptLeverage, ptTriangulationBlob, recalculateStress(), stressTable(), tableColbases(), tableDistances(), triangulationBlobs()
view

### Description

S3 method for viewing objects

### Usage

```r
view(x, ...)
```

### Arguments

- `x` The object to view
- `...` Additional arguments, not used.

### Value

When called on an acmap object, returns an htmlwidget object that can be used to interactively view the map. Otherwise by default it simply calls the print method of the respective object with no return value.

### See Also

Other functions to view maps: `RacViewer.options()`, `RacViewer()`, `export_viewer()`, `ggplot.acmap()`, `mapGadget()`, `plot.acmap()`, `setLegend()`, `view.acmap()`, `view.default()`
show_procrustes = NULL,
show_diagnostics = NULL,
num_optimizations = 1,
options = list()
)

Arguments

- **x** The acmap data object
- **optimization_number** The optimization number to view
- **...** Additional arguments to be passed to RacViewer()
- **.jsCode** Additional javascript code to be run after map has been loaded and rendered
- **.jsData** Any data to supply to the .jsCode function
- **select_ag** A vector of antigen indices to select in the plot
- **select_sr** A vector of serum indices to select in the plot
- **show_procrustes** If the map contains procrustes information, should procrustes lines be shown by default?
- **show_diagnostics** If the map contains diagnostics information like stress blobs or hemisphering, should it be shown by default?
- **num_optimizations** Number of optimization runs to send to the viewer for inclusion in the "optimizations" pane.
- **options** A named list of viewer options to pass to RacViewer.options()

Value

Returns an htmlwidget object

See Also

Other functions to view maps: RacViewer.options(), RacViewer(), export_viewer(), ggplot.acmap(), mapGadget(), plot.acmap(), setLegend(), view.default(), view()

Other shiny app functions: RacViewer-shiny, runGUI()
view.default

Default method for viewing objects

Description

Default method for viewing objects

Usage

```r
## Default S3 method:
view(x, ...)
```

Arguments

- `x` The object to view
- `...` Additional arguments, passed to print.

Value

No value returned, simply calls the print method on the object

See Also

Other functions to view maps: `RacViewer.options()`, `RacViewer()`, `export_viewer()`, `ggplot.acmap()`, `mapGadget()`, `plot.acmap()`, `setLegend()`, `view.acmap()`, `view()`
Index

* additional plotting functions
  blob, 15
  blobsize, 16
* antigen and sera attribute functions
  agAttributes, 9
  agGroups, 11
  agHomologousSr, 11
  agLabIDs, 12
  agSequences, 13
  ptAnnotations, 57
  ptClades, 60
  srAttributes, 84
  srGroups, 85
  srHomologousAgs, 86
  srSequences, 86
* functions for working with map data
  acmap, 4
  addOptimization, 6
  agReactivityAdjustments, 12
  as.json, 15
  edit_agNames, 24
  edit_srNames, 25
  keepBestOptimization, 30
  keepSingleOptimization, 31
  layerNames, 32
  orderPoints, 51
  read.acmap, 72
  read.titerTable, 73
  removePoints, 78
  save.acmap, 80
  save.coords, 81
  save.titerTable, 82
  subsetCommonPoints, 88
  subsetMap, 89
* functions relating to map stress calculation
  logtiterTable, 33
  mapDistances, 39
  mapResiduals, 42
  pointStress, 53
  recalculateStress, 75
  stressTable, 88
  tableColbases, 90
  tableDistances, 91
* functions relating to map transformation
  applyMapTransform, 14
  reflectMap, 75
  rotateMap, 79
  translateMap, 93
* functions to compare maps
  matchStrains, 44
  procrustesData, 54
  procrustesMap, 55
  realignMap, 73
  realignOptimizations, 74
* functions to view maps
  export_viewer, 26
  ggplot.acmap, 27
  mapGadget, 40
  plot.acmap, 51
  RacViewer, 68
  RacViewer.options, 70
  setLegend, 82
  view, 96
  view.acmap, 96
  view.default, 98
* functions to work with map optimizations
  keepOptimizations, 31
  optimizationProperties, 48
  removeOptimizations, 78
  sortOptimizations, 83
* map attribute functions
  acmapAttributes, 5
  adjustedLogTiterTable, 7
  adjustedTiterTable, 8
  dilutionStepsize, 22
  logtiterTableLayers, 34
  mapDescription, 38
  mapName, 41
map diagnostic functions
  agCohesion, 10
  bootstrapBlobs, 17
  bootstrapMap, 18
  checkHemisphering, 20
  dimensionTestMap, 23
  logtiterTable, 33
  map-table-distances, 36
  mapBootstrapCoords, 37
  mapDistances, 39
  mapRelaxed, 41
  mapResiduals, 42
  pointStress, 53
  ptBootstrapBlob, 58
  ptBootstrapCoords, 59
  ptLeverage, 62
  ptTriangulationBlob, 65
  recalculateStress, 75
  stressTable, 88
  tableColbases, 90
  tableDistances, 91
  triangulationBlobs, 94
  unstableMaps, 95

map merging functions
  htmlMergeReport, 29
  mergeMaps, 45
  mergeReport, 46
  RacMerge.options, 66
  splitTiterLayers, 84

map optimization attribute functions
  coIBases, 21
  mapComment, 37
  mapDimensions, 39
  mapStress, 43
  mapTransformation, 43
  ptBaseCoords, 57
  ptCoords, 60

map optimization functions
  make.acmap, 34
  moveTrappedPoints, 47
  optimizeMap, 49
  RacOptimizer.options, 67
  randomizeCoords, 71
  relaxMap, 76
  relaxMapOneStep, 77

map point style functions
  applyPlotspec, 14
  ptDrawingOrder, 62
  ptOpacity, 63
  ptStyles, 64

shiny app functions
  RacViewer-shiny, 69
  runGUI, 79
  view.acmap, 96

acmap, 4, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
acmapAttributes, 5, 8, 23, 34, 38, 41, 92, 93
addOptimization, 5, 6, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
adjustedLogTiterTable, 6, 7, 8, 23, 34, 38, 41, 92, 93
adjustedTiterTable, 6, 8, 8, 23, 34, 38, 41, 92, 93
adjustedTiterTable(.), 92
agAnnotations(ptAnnotations), 57
agAnnotations<-(ptAnnotations), 57
agAspect (ptStyles), 64
agAspect<-(ptStyles), 64
agAttributes, 9, 11–13, 57, 60, 85–87
agBaseCoords (ptBaseCoords), 57
agBaseCoords<-(ptBaseCoords), 57
agBootstrapBlob (ptBootstrapBlob), 58
agBootstrapBlobs (ptBootstrapBlob), 58
agBootstrapCoords (ptBootstrapCoords), 59
agClades (ptClades), 60
agClades<-(ptClades), 60
agCohesion, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
agContinent (agAttributes), 9
agContinent<-(agAttributes), 9
agCoords (ptCoords), 60
agCoords<-(ptCoords), 60
agDates (agAttributes), 9
agDates<-(agAttributes), 9
agExtra (agAttributes), 9
agExtra<-(agAttributes), 9
agFill (ptStyles), 64
agFill<-(ptStyles), 64
agGroups, 9, 11, 12, 13, 57, 60, 85–87
agGroups<-(agGroups), 11
agHomologousSr, 9, 11, 11, 12, 13, 57, 60, 85–87
agIDs (agAttributes), 9
agIDs <- (agAttributes), 9
agLabIDs, 9, 11, 12, 12, 13, 57, 60, 85–87
agLabIDs <- (agLabIDs), 12
agLeverage (ptLeverage), 62
agLineage (agAttributes), 9
agLineage <- (agLeverage), 62
agNames (agAttributes), 9
agName <- (agAttributes), 9
agNucleotideSequences (agSequences), 13
agNucleotideSequences <- (agSequences), 13
agOpacity <- (ptOpacity), 63
agOutline (ptStyles), 64
agOutline <- (ptStyles), 64
agOutlineWidth (ptStyles), 64
agOutlineWidth <- (ptStyles), 64
agPassage (agAttributes), 9
agPassage <- (agAttributes), 9
agReactivityAdjustments. 5, 7, 12, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
agReactivityAdjustments <- (agReactivityAdjustments), 12
agReassortant (agAttributes), 9
agReassortant <- (agAttributes), 9
agReference (agAttributes), 9
agReference <- (agAttributes), 9
agRotation (ptStyles), 64
agRotation <- (ptStyles), 64
agSequences. 9, 11, 12, 13, 57, 60, 85–87
agSequences <- (agSequences), 13
agShape (ptStyles), 64
agShape <- (ptStyles), 64
agShown (ptStyles), 64
agShown <- (ptStyles), 64
agSize (ptStyles), 64
agSize <- (ptStyles), 64
agStress (pointStress), 53
agStressPerTiter (pointStress), 53
agStrings (agAttributes), 9
agStrings <- (agAttributes), 9
agTriangulationBlob (ptTriangulationBlob), 65
agTriangulationBlobs (ptTriangulationBlob), 65
allMapDimensions (optimizationProperties), 48
allMapStresses (optimizationProperties), 48
applyMapTransform, 14, 76, 79, 93
applyPlotspec, 14, 62, 63, 65
as.json, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
.blob, 15, 16
blobsize, 16, 16
bootstrapBlobs, 10, 17, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
bootstrapMap, 10, 18, 19, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
checkHemisphering, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
colBases, 21, 38, 39, 43, 44, 58, 61
deprecated_functions, 22
dilutionStepsize, 6, 8, 22, 34, 38, 41, 92, 93
dilutionStepsize <- (dilutionStepsize), 22
dimensionTestMap, 10, 18, 20, 23, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
edit_agNames, 5, 7, 13, 15, 24, 25, 31, 32, 51, 72, 73, 79–82, 89
edit_srNames, 5, 7, 13, 15, 25, 25, 31, 32, 51, 72, 73, 79–82, 89
goViewer, 26, 28, 40, 53, 69, 71, 83, 96–98
fixedColBases (colBases), 21
fixedColBases <- (colBases), 21
ggplot.acmap, 26, 27, 40, 53, 69, 71, 83, 96–98
htmlAdjustedTiterTable, 29
htmlAdjustedTiterTable(), 8
htmlMergeReport, 29, 46, 66, 84
htmlTiterTable, 30
htmlTiterTable(), 92
keepBestOptimization, 5, 7, 13, 15, 25, 30, 32, 51, 72, 73, 79–82, 89
keepOptimizations, 31, 48, 78, 83
INDEX

keepSingleOptimization, 5, 7, 13, 15, 25, 31, 31, 32, 51, 72, 73, 79–82, 89
layerNames, 5, 7, 13, 15, 25, 31, 32, 32, 51, 72, 73, 79–82, 89
layerNames<-(layerNames), 32
listOptimizations, 33
logtiterTable, 10, 18, 20, 24, 33, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
logtiterTableLayers, 6, 8, 23, 34, 38, 41, 92, 93
make.acmap, 34, 47, 50, 68, 71, 77
map-table-distances, 36
mapBootstrap_agCoords
  (mapBootstrapCoords), 37
mapBootstrap_ptBaseCoords
  (mapBootstrapCoords), 37
mapBootstrap_srCoords
  (mapBootstrapCoords), 37
mapBootstrapCoords, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 93
mapCohesion (agCohesion), 10
mapComment, 21, 37, 39, 43, 44, 58, 61
mapComment<-(mapComment), 37
mapDescription, 6, 8, 23, 34, 38, 41, 92, 93
mapDescription<-(mapDescription), 38
mapDimensions, 21, 38, 39, 43, 44, 58, 61
mapDistances, 10, 18, 20, 24, 34, 36, 37, 39, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
mapGadget, 26, 28, 40, 53, 69, 71, 83, 96–98
mapName, 6, 8, 23, 34, 38, 41, 92, 93
mapName<-(mapName), 41
mapRelaxed, 10, 18, 20, 24, 34, 36, 37, 40, 41, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
mapResiduals, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
mapStress, 21, 38, 39, 43, 44, 58, 61
mapTransformation, 21, 38, 39, 43, 44, 58, 61
mapTransformation<-(mapTransformation), 43
mapTranslation (mapTransformation), 43
mapTranslation<-(mapTransformation), 43
match_mapAntigens (matchStrains), 44
match_mapSera (matchStrains), 44
matchStrains, 44, 55, 56, 74
mergeMaps, 30, 45, 46, 66, 84
mergeReport, 30, 46, 46, 66, 84
minColBasis (colBases), 21
minColBasis<-(colBases), 21
moveTrappedPoints, 35, 47, 50, 68, 71, 77
numAntigens (acmapAttributes), 5
numLayers (acmapAttributes), 5
numOptimizations (acmapAttributes), 5
numPoints (acmapAttributes), 5
numSera (acmapAttributes), 5
optimizationProperties, 31, 48, 78, 83
optimizeAgReactivity, 48
optimizeMap, 35, 47, 49, 68, 71, 77
orderAntigens (orderPoints), 51
orderPoints, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
orderSera (orderPoints), 51
plot.acmap, 26, 28, 40, 51, 69, 71, 83, 96–98
plot_map_table_distance
  (map-table-distances), 36
plotly_map_table_distance
  (map-table-distances), 36
pointStress, 10, 18, 20, 24, 34, 36, 37, 40, 42, 53, 59, 63, 65, 75, 88, 90, 91, 95
procrustesData, 44, 54, 56, 74
procrustesMap, 44, 55, 57, 74
ptAnnotations, 9, 11–13, 57, 60, 85–87
ptBaseCoords, 21, 38, 39, 43, 44, 57, 61
ptBootstrapBlob, 10, 18, 20, 24, 34, 36, 37, 40, 54, 58, 59, 63, 65, 75, 88, 90, 91, 95
ptBootstrapBlobs (ptBootstrapBlob), 58
ptBootstrapCoords, 10, 18, 20, 24, 34, 36, 37, 40, 54, 59, 59, 63, 65, 75, 88, 90, 91, 95
ptClades, 9, 11–13, 57, 60, 85–87
ptCoords, 21, 38, 39, 43, 44, 58, 60
ptCoords<-(ptCoords), 60
ptDrawingOrder, 15, 62, 63, 65
ptDrawingOrder<-(ptDrawingOrder), 62
ptLeverage, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 62, 65, 75, 88, 90, 91, 95
ptOpacity, 15, 62, 63, 65
ptStyles, 15, 62, 63, 64
ptTriangulationBlob, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
ptTriangulationBlobs
  (ptTriangulationBlob), 65

RacMerge.options, 30, 46, 66, 84
RacOptimizer.options, 35, 47, 50, 67, 71, 77
RacViewer, 26, 28, 40, 53, 68, 71, 83, 96–98
RacViewer-shiny, 69
RacViewer.options, 26, 28, 40, 53, 69, 70, 83, 96–98
RacViewerOutput (RacViewer-shiny), 69
randomizeCoords, 35, 47, 50, 68, 71, 77
read.acmap, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
read.titerTable, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89
realignMap, 44, 55, 56, 73, 74
realignOptimizations, 44, 55, 56, 74, 74
recalculateStress, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
reflectMap, 14, 75, 79, 93
relaxMap, 35, 47, 50, 68, 71, 76, 77
relaxMapOneStep, 35, 47, 50, 68, 71, 77, 77
removeAntigens (removePoints), 78
removeOptimizations, 31, 48, 78, 83
removePoints, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 78, 80–82, 89
removeSera (removePoints), 78
renderRacViewer (RacViewer-shiny), 69
rotateMap, 14, 76, 79, 93
runGUI, 70, 79, 97
save.acmap, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79, 80, 81, 82, 89
save.coords, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79, 80, 81, 82, 89
save.titerTable, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–81, 82, 89
setLegend, 26, 28, 40, 53, 69, 71, 82, 96–98
sortOptimizations, 31, 48, 78, 83
splitTiterLayers, 30, 46, 66, 84
srAnnotations (ptAnnotations), 57
srAnnotations (ptAnnotations), 57
srAspect (ptStyles), 64
srAspect (ptStyles), 64
srAttributes, 9, 11–13, 57, 60, 84, 85–87
srBaseCoords (ptBaseCoords), 57
srBaseCoords (ptBaseCoords), 57
srBootstrapBlob (ptBootstrapBlob), 58
srSize <- (ptStyles), 64
srSpecies (srAttributes), 84
srSpecies <- (srAttributes), 84
srStress (pointStress), 53
srStressPerTiter (pointStress), 53
srStrings (srAttributes), 84
srStrings <- (srAttributes), 84
srTriangulationBlob
  (ptTriangulationBlob), 65
srTriangulationBlobs
  (ptTriangulationBlob), 65
standardizeStrainNames, 87
stressBlobs (deprecated_functions), 22
stressTable, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
subsetCommonAgs (subsetCommonPoints), 88
subsetCommonPoints, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 88, 89
subsetCommonSrGroups
  (subsetCommonPoints), 88
subsetMap, 5, 7, 13, 15, 25, 31, 32, 51, 72, 73, 79–82, 89, 89
tableColbases, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
tableDistances, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95
titerLeverage (ptLeverage), 62
titerTable, 6, 8, 23, 34, 38, 41, 91, 93
titerTable <- (titerTable), 91
titerTableFlat (titerTable), 91
titerTableFlat <- (titerTable), 91
titerTableLayers, 6, 8, 23, 34, 38, 41, 92, 92
titerTableLayers <- (titerTableLayers), 92
translateMap, 14, 76, 79, 93
triangulationBlobs, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 94, 95
unstableMaps, 10, 18, 20, 24, 34, 36, 37, 40, 42, 54, 59, 63, 65, 75, 88, 90, 91, 95, 95
view, 26, 28, 40, 53, 69, 71, 83, 96, 97, 98
view.acmap, 26, 28, 40, 53, 69–71, 80, 83, 96, 96, 98
view.default, 26, 28, 40, 53, 69, 71, 83, 96, 97, 98