Package 'hypervolume'

August 24, 2014

Type Package

Title High-dimensional kernel density estimation and geometry operations.

Version 1.0.2

Date 2014-08-24

Author Benjamin Blonder

Maintainer Benjamin Blonder <bblonder@gmail.com>

Description Estimates the shape and volume of high-dimensional objects and performs set operations: intersection / overlap, union, unique components, inclusion test, and non-convex feature detection. Can measure the n-dimensional ecological hypervolume and perform species distribution modeling.

License GPL-3

Depends Rcpp, rgl, methods, R (>= 3.0.0)

LinkingTo Rcpp

Imports MASS, geometry, igraph

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-24 23:04:09

R topics documented:

`hypervolume-package` *N-dimensional hypervolume operations*

Description

Estimates the shape and volume of high-dimensional objects and performs set operations: intersection / overlap, union, unique components, and inclusion test. Can measure the n-dimensional ecological hypervolume and perform species distribution modeling.

Details

Frequently asked questions

What quantile threshold should I use?

A value of 0.0 will ensure that the final hypervolume encloses all observed points. A value of 0.1 is more robust to outliers.

How do I know the hypervolume algorithm has converged on the correct answer?

You can generate multiple hypervolumes, then plot computed volume against the number of Monte Carlo samples. You should choose a large enough number of Monte Carlo samples to constrain variation in volume to within your desired tolerance.

What fraction of points should I use for set operations? If I use all the points the algorithm is too slow.

The set operations scale approximately quadratically with the number of points, so choosing a smaller point fraction will improve runtimes quadratically. However, using fewer points decreases the accuracy of the computation and may result in some points being identified as 'out' when they should be 'in', and could potentially bias estimates of computed volumes. Choose a large enough point fraction that the computed volume converges to a value within your desired tolerance.

How can I export hypervolume data to another program for plotting?

Each hypervolume object stores a slot called `RandomUniformPointsThresholded`. Points in this field are a random uniform sample of the hypervolume, so you can export these points for graphing or inference in other programs.

Why does the hypervolume look blocky, and not smooth around the edges?

The algorithms use a hyperbox kernel for computational efficiency. For small sample sizes or widely spaced points, the kernel bandwidth may be large relative to the spread of points. As a result, the hypervolume will appear to have jagged edges. This effect cannot be avoided in the current implementation.

I have a multi-core machine. How can I speed up the algorithms?

You can run multiple hypervolume commands in parallel using several possible packages. See http://cran.r-project.org/web/views/HighPerformanceComputing.html for details.

I don't understand what the 'uniformly random' output means.

The hypervolume algorithm proceeds by 1) computing a kernel density estimate, 2) thresholding this estimate to include some quantile of the total probability density, 3) defining the volume as all points whose density value exceeds this threshold, and 4) sampling random points from within this volume, such that the point density of the final object is constant within the volume and zero outside the volume. This is the stochastic equivalent of a binary definition of an object (points are either in or out) and is necessary to do set operations. Thus the object is completely defined by the 'edge' of these points and no further contouring should be done.

Why do some parts of the hypervolume appear to have higher point density?

This is because each pairplot is a projection of a high-dimensional object to two dimensions - so 'thicker' regions of the object will effectively show more points.

Why do I not get the same answer if I run the same code repeatedly?

The algorithms are stochastic and depend on the state of the random number generator. If results are unreliable, increase the number of Monte Carlo samples. Alternative you can make results repeatable by fixing the random number generator seed in your code, e.g. `set.seed(3)`.

Author(s)

Benjamin Blonder <bblonder@gmail.com>

References

Blonder, B., Lamanna, C., Violle, C. and Enquist, B. J. (2014), The n-dimensional hypervolume. Global Ecology and Biogeography, 23: 595-609. doi: 10.1111/geb.12146

`estimate_bandwidth` *Silverman bandwidth estimator for hypervolumes.*

Description

Estimates bandwidth vector from data using a Silverman bandwidth estimator applied independently to each axis of the data. This approach provides a heuristic when no other methods are available to choose kernel bandwidth.

Usage

```
estimate_bandwidth(data)
```

Arguments

`data`	m x n matrix or data frame, where m is the number of observations and n the number of dimensions.

Details

The Silverman estimator is defined as 1.06 * sd(X) * m^(-1/5) where m is the number of observations and X is the data vector in each dimension. Note that this estimator is optimal only for univariate normal data and not for the box kernels implemented by the hypervolume algorithms.

Value

Vector of length n with each entry corresponding to the estimated bandwidth along each axis.

Examples

```
data(iris)
print(estimate_bandwidth(iris[,1:4]))
```

---

finch *Data and demo for Darwin's finches*

Description

Data for nine morphological traits for five species of Darwin's finches occurring on Isabela Island. Demonstration analysis of hypervolume overlaps between all pairs of species.

Usage

```
data(finch)
```

Format

A data frame with 146 observations on the following 10 variables.

`Species` a factor with levels `Geospiza fortis` `fortis Geospiza fortis platyrhyncha` `Geospiza fuliginosa parvula` `Geospiza heliobates` `Geospiza prosthemelas prosthemelas`

`BodyL` a numeric vector

`WingL` a numeric vector

`TailL` a numeric vector

`BeakW` a numeric vector

`BeakH` a numeric vector

`LBeakL` a numeric vector

`UBeakL` a numeric vector

`N.UBkL` a numeric vector

`TarsusL` a numeric vector

Source

Data are from Snodgrass RE and Heller E (1904) Papers from the Hopkins-Stanford Galapagos Expedition, 1898-99. XVI. Birds. Proceedings of the Washington Academy of Sciences 5: 231-372.

References

Blonder, B., Lamanna, C., Violle, C., Enquist, B. The n-dimensional hypervolume. Global Ecology and Biogeography (in press)

Examples

```
demo('finch', package='hypervolume')
```

`get_volume` *Extract volume*

Description

Extract volume from Hypervolume or HypervolumeList object

Usage

```
## S3 method for class 'Hypervolume'
get_volume(object)
## S3 method for class 'HypervolumeList'
get_volume(object)
```

Arguments

`object`	A `Hypervolume` or `HypervolumeList` object

Value

A named numeric vector with the volume of each input hypervolume

`hypervolume` *Hypervolume construction*

Description

Constructs a hypervolume from a set of observations via thresholding a kernel density estimate of the observations. Assumes a hyperbox kernel.

Usage

```
hypervolume(data, repsperpoint=1000, bandwidth,
  quantile = 0, name = NULL,
  verbose = T, warnings = T)
```

Arguments

`data`	A m x n matrix or data frame, where m is the number of observations and n is the dimensionality.
`repsperpoint`	The number of random points to generate in the kernel around each data point. Larger values are needed in higher dimensions, and generally produce more accurate results.
`bandwidth`	A scalar or a n x 1 vector corresponding to the half-width of the box kernel in each dimension. If a scalar input, the single value is used for all dimensions. Bandwidth also can be estimated using `estimate_bandwidth` if necessary (not recommended).
`quantile`	A number in [0,1), corresponding to the fraction of probability density to exclude from the hypervolume. A value of 0 encloses all data, while a value closer to 1 excludes more data. Note that this is a requested value; due to the discrete nature of the estimation procedure the obtained quantile may differ. A value of 0 can always be obtained.
`name`	A string to assign to the hypervolume for later output and plotting. Defaults to the name of the variable if NULL.
`verbose`	Logical value; print diagnostic output if true.
`warnings`	Logical value; checks for several potential issues in the input data if true. Checks for high variance in standard deviations between dimensions (indicating axis scale problems), highly correlated dimensions (indicating axis choice problems), and low number of observations (indicating algorithm applicability problems).

Details

Constructs a kernel density estimate by overlaying hyperbox kernels on each datapoint, then sampling uniformly random points from each kernel. Kernel density at each point is then determined by a range query on a recursive partitioning tree and used to resample these random points to a uniform density and fixed number, from which a volume can be inferred.

Value

A Hypervolume-class object corresponding to the inferred hypervolume.

Examples

```
data(iris)
hv1 = hypervolume(subset(iris, Species=="setosa")[,1:4],bandwidth=0.2)
summary(hv1)
```

Hypervolume-class *Class hypervolume*

Description

A class used to store a stochastic description of a hypervolume.

Objects from the Class

Objects can be created by calls of the form `new("Hypervolume", ...)`.

Slots

`Name`: Object of class `"character"`. A string naming the hypervolume, used in plotting.

`Data`: Object of class `"matrix"`. If available, the raw data used to construct the hypervolume. Defaults to a one-row NaN vector for hypervolumes returned by set operations.

`Dimensionality`: Object of class `"numeric"`. The dimensionality of the hypervolume.

`Volume`: Object of class `"numeric"`. The volume of the hypervolume, in units of the product of all dimensions.

`PointDensity`: Object of class `"numeric"`. The number density of the uniformly sampled random points characterizing the hypervolume.

`Bandwidth`: Object of class `"numeric"`. If available, the bandwidth vector used to construct the hypervolume. Defaults to a one-row NaN vector for hypervolumes returned by set operations.

`DisjunctFactor`: Object of class `"numeric"`. The ratio of the inferred volume to the volume of a hypervolume constructed from the same data with disjunct data points (i.e. no kernels overlap). Varies from zero to one. High values suggest that bandwidth should be increased.

`RepsPerPoint`: Object of class `"numeric"`. If available, the number of random points used per observation to construct the hypervolume. Defaults to NaN for hypervolumes returned by set operations.

`QuantileThresholdDesired`: Object of class `"numeric"`. If available, the quantile requested by the user and used to construct the hypervolume. Defaults to NaN for hypervolumes returned by set operations.

`QuantileThresholdObtained`: Object of class `"numeric"`. If available, the quantile obtained by the hypervolume algorithm. Defaults to NaN for hypervolumes returned by set operations.

`RandomUniformPointsThresholded`: Object of class `"matrix"` A set of uniformly random points guaranteed to be in the hypervolume.

`ProbabilityDensityAtRandomUniformPoints`: Object of class `"numeric"` A vector of integers proportional to the probability density at each uniformly random point in the hypervolume. Defaults to a 1-valued vector for hypervolumes returned by set operations because set operations are well defined for volumes and not for probability density functions.

Methods

Summary and plot methods are available.

`HypervolumeList-class` *Class HypervolumeList*

Description

A convenience class used to streamline plotting and other hypervolume output.

Objects from the Class

Objects can be created by calls of the form `new("HypervolumeList", ...)`.

Slots

`HVList`: Object of class `"list"`. A list of hypervolumes.

`hypervolume_importance`
Hypervolume variable importance

Description

Assesses the contribution of each variable to the total hypervolume as a rough metric of variable importance.

Usage

```
hypervolume_importance(hv)
```

Arguments

`hv`	A hypervolume for which the importance of each variable should be calculated.

Details

The algorithm proceeds by comparing the n-dimensional input hypervolume's volume to all possible n-1 dimensional hypervolumes where each variable of interest has been deleted. The importance score reported is the ratio of the n-dimensional hypervolume relative to each of the n-1 dimensional hypervolumes. Larger values indicate that a variable makes a proportionally higher contribution to the overall volume.

This function is currently experimental. Other definitions of variable importance may also be possible. The algorithm currently cannot be used on set operations hypervolumes because the variable deletion process is not well defined for objects that are not associated with a particular set of observations.

Value

A named vector with importance scores for each axis. Note that these scores are not dimensionless but rather have linear units.

Examples

```
data(iris)
hv1 = hypervolume(subset(iris, Species=="setosa")[,1:4],reps=1000,bandwidth=0.2)
imp = hypervolume_importance(hv1)

print(imp)
```

---

hypervolume_inclusion_test

Inclusion test

Description

Determines if a set of points are within a hypervolume.

Usage

```
hypervolume_inclusion_test(hv, points, reduction_factor = 1, verbose = T)
```

Arguments

hv	n-dimensional hypervolume to compare against
points	Candidate points. A m x n matrix or dataframe, where m is the number of candidate points and n is the number of dimensions.
reduction_factor	A number in (0,1] that represents the fraction of random points sampled from the hypervolume for the stochastic inclusion test. Larger values are more accurate but computationally slower.
verbose	Logical value; print diagnostic output if true.

Details

Tests if a point is in a hypervolume by determining if it is within a characteristic distance of at least one point from a uniformly random sample of the hypervolume.

Value

A m x 1 logical vector indicating whether each candidate point is in the hypervolume.

Examples

```
# construct a hypervolume of points in the unit square [0,1] x [0,1]
data = data.frame(x=runif(100,min=0,max=1), y=runif(100,min=0,max=1))
hv = hypervolume(data, reps=1000, bandwidth=0.1)

# test if (0.5,0.5) and (-1,1) are in - should return TRUE FALSE
hypervolume_inclusion_test(hv, points=data.frame(x=c(0.5,-1),y=c(0.5,-1)))
```

---

hypervolume_set *Set operations (intersection / union / unique components)*

Description

Computes the intersection, union, and unique components of two hypervolumes.

Usage

```
hypervolume_set(hv1, hv2, reduction_factor = 1, verbose = T, check_memory = T)
```

Arguments

hv1	A n-dimensional hypervolume
hv2	A n-dimensional hypervolume
reduction_factor	A number in (0,1] that represents the fraction of random points sampled from each hypervolume for the stochastic inclusion test. Larger values are more accurate but computationally slower.
verbose	Logical value; print diagnostic output if true.
check_memory	Logical value; returns information about expected memory usage if true.

Details

Uses the inclusion test approach to identify points in the first hypervolume that are or are not within the second hypervolume and vice-versa. The intersection is the points in both hypervolumes, the union those in either hypervolume, and the unique components the points in one hypervolume but not the other.

By default, the function uses check_memory=TRUE which will provide an estimate of the computational cost of the set operations. The function should then be re-run with check_memory=FALSE

if the cost is acceptable. This algorithm's memory and time cost scale quadratically with the number of input points, so large datasets can have disproportionately high costs. This error-checking is intended to prevent the user from large accidental memory allocation.

If the computational cost is too high, a small value of reduction_factor can be specified to reduce the number of points randomly sampled for this stochastic analysis. Linear decreases in this parameter produce quadratic improvements in cost.

Value

If check_memory is false, returns a HypervolumeList object, with four items in its HVList slot:

`Intersection`	The intersection of hv1 and hv2
`Union`	The union of hv1 and hv2
`Unique_1`	The unique component of hv1 relative to hv2
`Unique_2`	The unique component of hv2 relative to hv1

Note that the output hypervolumes will have lower random point densities than the input hypervolumes.

You may find it useful to define a Jaccard-type fractional overlap between hv1 and hv2 as hv_set@HVList$Intersection@Volu
/ hv_set@HVList$Union@Volume.

If check_memory is true, instead returns a scalar with the expected number of pairwise comparisons.

Examples

```
data(iris)
hv1 = hypervolume(subset(iris, Species=="setosa")[,1:4],reps=1000,bandwidth=0.2,warn=FALSE)
hv2 = hypervolume(subset(iris, Species=="virginica")[,1:4],reps=1000,bandwidth=0.2,warn=FALSE)

hv_set = hypervolume_set(hv1, hv2, reduction_factor = 0.5, check_memory=FALSE)

# no overlap found
get_volume(hv_set)
```

---

nonconvexfeatures *Non-convex feature detection (dents and holes)*

Description

Infers the presence of non-convex features (dents and holes) within a hypervolume. The algorithm's approach is to 1) compute a convex hull that encloses the data points, representing the maximal hypervolume and simplest linear inequality constraints that could describe the data; 2) sample a random set of points from this hull, 3) compute a thresholded kernel density estimate of the data points, representing the multidimensional hypervolume that is the niche; 4) sample a random set of points from this hypervolume; 4) determine the subset of random points that are within the convex hull and are not within the hypervolume; 5) identify non-convex features by determining the connectivity of these random points; and 6) segment these random points into holes or dents by determining if they touch the convex boundary.

Usage

```
nonconvexfeatures(hv,
  numconvexhullpoints = NULL, memory_reduction_factor = 1,
  check_convexhull = TRUE, check_memory = TRUE,
  minnumpoints = 5, npointsdesired = 1000,
  inflationfactors = seq(1, 2, by = 0.2),
  plotdiag = FALSE)
```

Arguments

hv	The hypervolume to be analyzed. Must have a non-empty Data slot, i.e. cannot be the output of set operations. The convex hull must be drawn around data points rather than uniformly random points. The kernel density estimate is not smooth due to the underlying hyperbox kernel. Thus delineating a convex hull around the hypervolume's random points necessarily will cause the algorithm to detect non-convex features where there are jagged transitions.
numconvexhullpoints	Number of points in hv@Data to use to delineate the convex hull. Smaller numbers will produce exponentially faster results but also much less accurate delineation of the convex hull and potential omission of non-convex features. The algorithm does preferentially sample points far away from the data centroid to minimize this effect. Leaving this parameter as NULL results in the algorithm using all data points to delineate the convex hull.
memory_reduction_factor	A number between 0 and 1 indicating the fraction of random points in the convex hull and hypervolume to use for set operations. Larger values are more accurate but use more computational resources.
check_convexhull	If TRUE, calculates convex hull and estimates memory usage necessary to perform hit-and-run sampling for generating uniformly random points, then exits. If FALSE, prints resource usage and continues algorithm. It is very useful for preventing crashes to check the estimated memory usage on large or high dimensional datasets before running the full algorithm.
check_memory	If TRUE, estimates the memory usage required to perform set operations between the convex hull and hypervolume, then exits. If FALSE, prints resource usage and continues algorithm. It is very useful for preventing crashes to check the estimated memory usage on large or high dimensional datasets before running the full algorithm.
minnumpoints	Number of random points in the smallest non-convex feature to be retained by the algorithm. Very small non-convex features are assumed to be false positives and pruned from the algorithm's output.
npointsdesired	Number of random points to sample from the convex hull. Larger values are more accurate.
inflationfactors	A vector of numbers representing multiplicative factors used when segmenting non-convex features. Features separated by at least a critical distance multiplied

by each inflation factor are inferred to be separate objects. Larger values produce fewer but larger features. The algorithm returns a HypervolumeList of non-convex features for each value in inflationfactors, so that the user can see how the choice of critical distance affects inference of the number and size of non-convex features. A value of approximately two seem to work well in practice.

plotdiag — Generate diagnostic plot of main steps in algorithm. If true, will create a file called 'diag.pdf'. Only works in n=2 dimensions.

Details

The algorithm's computational complexity can scale exponentially with the number of data points in the hypervolume as well as the dimensionality. For this reason, the algorithm prints extensive diagnostic output describing the resources allocated at each step. The algorithm also terminates by default before computationally intensive steps to warn the user of the expected resources needed; this behavior can be disabled by setting the appropriate flags (check_convexhull and check_memory to false).

Value

A list of HypervolumeList objects, suitable for plotting. Each item in the list corresponds to the non-convex features inferred for a given value of inflationfactors and is named accordingly. Each HypervolumeList has a HVList slot. In this slot, the first list item is the original hypervolume. All subsequent list items, if any, are individual non-convex features. They are named 'Dent' if a dent and 'Hole' if a hole.

Typically the user will want to access the last item in the list, as segmentation of non-convex features tends to be most accurate for larger inflation factors.

Author(s)

Benjamin Blonder

References

Blonder, B. Is niche geometry non-convex, and how would we know? (in preparation)

See Also

demo(iris_nonconvexity)

Examples

```
# generate annulus data
data_annulus <- data.frame(matrix(data=runif(4000),ncol=2))
names(data_annulus) <- c("x","y")
data_annulus  <- subset(data_annulus,
sqrt((x-0.5)^2+(y-0.5)^2) > 0.4 & sqrt((x-0.5)^2+(y-0.5)^2) < 0.5)

# MAKE HYPERVOLUME (low reps for fast execution)
hv_annulus <- hypervolume(data_annulus,bandwidth=0.1,name='annulus',reps=500)
```

```
# DETECT FEATURES (low npoints for fast execution)
features_annulus <- nonconvexfeatures(hv_annulus,
check_memory = FALSE, check_convexhull=FALSE,
inflationfactors = 2, minnumpoints=10)

# ANALYZE FOR THE LARGEST INFLATION FACTOR
features_annulus_largest <- features_annulus[[length(features_annulus)]]

# PLOT RESULTS
plot(features_annulus_largest)
```

`plot.HypervolumeList` *Plot a hypervolume or list of hypervolumes*

Description

Plots a single hypervolume or multiple hypervolumes as either a pairs plot (all axes) or a 3D plot (a subset of axes). The hypervolume is drawn as a uniformly random set of points guaranteed to be in the hypervolume.

Usage

```
## S3 method for class 'HypervolumeList'
plot(x, npmax = 1000,
  colors = rainbow(length(x@HVList)), names = NULL,
  reshuffle = TRUE, showdensity = TRUE, showdata = TRUE,
  darkfactor = 0.5, cex.random = 0.5, cex.data = 0.75,
  cex.axis = 0.75, cex.names = 1, cex.legend = 0.75,
  legend = TRUE, varlims = NULL, pairplot = TRUE,
  whichaxes = NULL, ...)
```

Arguments

`x`	A Hypervolume or HypervolumeList object. The objects to be plotted.
`npmax`	An integer indicating the maximum number of random points to be sampled from each hypervolume. Lower values result in faster plotting and smaller file sizes but less accuracy.
`colors`	A vector of colors to be used to plot each hypervolume, in the same order as the input hypervolumes.
`names`	A vector of strings in the same order as the input hypervolumes. Used to draw the legend.
`reshuffle`	A logical value relevant when pair=TRUE. If false, each hypervolume is drawn on top of the previous hypervolume; if true, all points of all hypervolumes are randomly shuffled so no hypervolume is given visual preference during plotting.

showdensity	A logical value indicating if the probability density of each hypervolume should be drawn by modulating alpha values (more transparent for lower probability density). Note that this has no effect when probability density is not relevant, i.e. for hypervolumes that are the output of set operations.
showdata	A logical value indicating if the original data should be drawn on top of the uniformly random points. Note that this has no effect if the hypervolume is not associated with data points, e.g. for those that are the output of set operations.
darkfactor	A value in [0,1] that modulates the color of data points, if shown. Values closer to 0 make data points more black, while values closer to 1 make data points closer to the input color.
cex.random	cex value for uniformly random points.
cex.data	cex value for data points.
cex.axis	cex value for axes, if pair=T.
cex.names	cex value for variable names printed on the diagonal, if pair=T.
cex.legend	cex value for the legend text
legend	Logical value indicating if a legend should be plotted, if pair=T
varlims	A list of two-element vectors corresponding to the axes limits for each dimension. If a single two-element vector is provided it is re-used for all axes.
pairplot	If true, a pair plot is produced. If false, a 3D plot is produced.
whichaxes	A length-three vector of integer IDs corresponding to the axes to be plotted when pair=F.
...	

Value

None; used for the side-effect of producing a plot.

Examples

```
data(iris)
hv1 = hypervolume(subset(iris, Species=="setosa")[,1:4],reps=1000,bandwidth=0.2)

# choose fixed axes
plot(hv1, pair=TRUE, npmax=500, varlims=list(c(3,6),c(2,5),c(0,3),c(-1,1)))
```

---

quercus *Data and demo for Quercus (oak) tree distributions*

Description

Data for occurrences of Quercus alba and Quercus rubra based on geographic observations. Demonstration analysis of how to use hypervolumes for species distribution modeling using WorldClim data.

Usage

```
data(quercus)
```

Format

A data frame with 3779 observations on the following 3 variables.

`Species` a factor with levels `Quercus alba` `Quercus rubra`

`Latitude` a numeric vector

`Longitude` a numeric vector

Source

Occurrence data come from the BIEN2 database (http://bien.nceas.ucsb.edu/bien/). Climate data are from WorldClim.

References

Blonder, B., Lamanna, C., Violle, C., Enquist, B. The n-dimensional hypervolume. Global Ecology and Biogeography (in press)

Examples

```
demo('quercus', package='hypervolume')
```

---

summary.Hypervolume *Summary of hypervolume*

Description

Prints basic information about Hypervolume or HypervolumeList structure.

Usage

```
## S3 method for class 'Hypervolume'
summary(object, ...)
## S3 method for class 'HypervolumeList'
summary(object, ...)
```

Arguments

`object`	The hypervolume to summarize
`...`	

Value

None; used for the side-effect of printing.

Index