Function: cooccurence()

Depends: None

Argument	Default	Description
character1		Integer: Number indicating the row of the first character of interest in the extinct dataset.
character2		Integer: Number indicating the row of the second character of interest in the extinct dataset.
extinctDataset		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.

Description:

Given two characters, this function calculates the percentage of instances where either the two characters are both present or both absent in the dataset. The output of this value will range between 0 (meaning no correlation) to 1 (meaning perfect correlation).

Function: createCorrelationMatrix()

Depends: cooccurence()

Argument	Default	Desciption
extinctDataset		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.

Description:

Creates a square matrix which shows the correlation of presence and absence between all characters of the extinct dataset.

Function: calculateGlobalCharacterLoss()

Depends: None; function interally defined

Argument	Default	Description
extinctDataset		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.

Description:

For all characters in the extinct dataset, this function calculates the percent of extinct taxa that are missing that character. The output of this function is a vector containing the percentage of taxa missing a character for each character in the data.

Function: FPSsample()

Depends: None

Argument	Default	Description
weights		Vector: vector with length equal to the number of characters for a single taxon. This vector stores the relative probabilities that any given character will be returned.

Description:

This function acts as a weighted sampler which will randomly select an index value based on the relative probabilities of each index being selected. The output of this function is the index which is selected based on the weights vector. This is an implementation of a genetic algorithm known as "fitness proportionate selection" (FPS) or "roulette wheel selection". For an input, the function takes a vector where the values of in the vector are the relative probabilities that each index will be selected. The values in the *weights* vector must be greater than or equal to 0, and can be greater than 1. Indices with weights of 0 have no chance of being selected, which means sampling without replacement can be implemented by changing the weight of selected values to 0 before rerunning the sampler.

Function: computeWeights()

Depends: None

Argument	Default	Description
initialWeights		Vector: same length as number of characters in the dataset. Each value is the relative weight that each character is selected using a sampler such as FPSsample()
characterCorrelations		Matrix: square matrix representing the calculated co-occurences of characters in the extinct dataset. This is calculated by the function createCorrelationMatrix()
removedChars		Vector: running list of characters that have been previously removed previously during the same simulation.
whichLostChar		Integer: for a given taxon, the number of characters that have been lost in the current simulation. This number is used with the vector *whichLostChar* to determine the last character which was lost.
globalLoss		Vector: same length as number of characters in the dataset, representing the global absence of each character in the extinct dataset across all taxa. For the first simulation, this vector is equal to the *initialWeights*. This vector must be inputed separately from *initialWeights* because this information must remain unchanged while *initialWeights* is dynamically changing.

Description:

Once a character has been chosen to be lost by the algorithm, the computeWights() function updates the weights for the next run of the simulation. This function first calculates the conditional probability of choosing the other characters given information about the last character that was chosen by multiplying the previous weights by their correlations with the last character lost. This vector of conditional probabilities are then divided by the probability of losing the focal character to put the weights into a Bayesian framework. The output of this function is then used as the weights for the next simulation for a given taxon. Note that dividing by the global probability of the focal character does not change the weights (since the weights are all relative) and is only included here to maximize the number of simulations (and more importantly, decimal multiplications) before the weights fall out of the range of double precision.

Function: withoutReplacement()

Depends: None

Argument	Default	Description
weights		Vector: same length as number of characters in the dataset. Each value is the relative weight that each character is selected using a sampler such as FPSsample().
characterLost		Integer: the character which is lost. This is typically the output of a sampler such as FPSsample().

Description:

This function updates the weights such that the sampler continues without replacement. This is done by setting the weight of the last chosen character to 0, thereby prohibiting the sampler from choosing that character again.

Function: updateCharsLost()

Depends: None

Argument	Default	Description
charsLost		Vector: running list of characters that have been previously removed previously during the same simulation.
lastCharLost		Integer: the character which is lost. This is typically the output of a sampler such as FPSsample().
numCharsLost		Integer: for a given taxon, the number of characters that have been lost in the current simulation.

Description:

This function updates the running list of lost characters.

Function: chooseCharsToRemove()

Depends: FPSsample()
computeWeights()
withoutReplacement()
updateCharsLost()

Argumentdoes	Default	Desciption
numCharsToLose		Integer: The number of characters that a single taxon is simulated to lose.
weights		Vector: Relative probabilities that each character will be lost. When losing the first character, the weights assigned based on the percentage that the characters are missing in the entire extinct dataset. Future iterations are then updated based on the a correlation matrix and characters which have been lost in previous iterations. Note that characters with weight 0 will never be chosen to be lost. This vector is calculated using the function createGlobalLoss().
correlations		Matrix: A square matrix that includes information about the percent of taxa which show correlations in the presence of absence of characters. This matrix is generated using the function createCorrelationMatrix().
globalLoss		Vector: A vector with the same length as the number of taxa in the study containing the percentage of taxa missing a character for each character in the data. This is the output of the function createGlobalLoss(). For the first character lost, this vector is equal to the *weights* input
softChars		Integer: A single number equaling the number soft characters for one taxon in the dataset. Note that the soft characters should appear as the last rows of the datasets, and that these characters will be preferentially lost first.
maxLength		Integer: Maximum number of characters that will be lost for all taxa and all simulations. This number is used to pad the output with NA values so that the outputs of different taxa and simulations can be combined into a matrix.
totalCharsLost		Integer: The number of characters that have been lost for all taxa within the current simulation. This number is used to terminate the current simulation if the desired number of characters are lost
totalCharsToBeLost		Integer: The total number of characters that can be lost across all taxa in a single simulation. When the total number of characters lost equals this number, the simulation terminates.

Description:

This function is called iteratively for each taxon, and simulates which characters should be lost. This function outputs a list including two values: the first value is the current number of characters lost across all taxa for the current simulation, and the other is a vector of the character IDs that should be lost, and is placed as the result of a single column of matrix generated by simulateCharacterLoss(). In

order to ensure that the output can be used in a matrix, the output is padded with NA values such that the lengths of the outputs of all iterations of this function is the same.

Function: chooseCharsNoLinkage()

Depends: None

Argument	Default	Description
numCharsToLose		Integer: number of characters per taxon that should be lost during the simulation.
globalLoss		Vector: same length as number of characters in the dataset, representing the global absence of each character in the extinct dataset across all taxa.
maxLength	NULL	Integer: the maximum number of characters to be lost. If specified as NULL, the length of the output will equal the *numCharsToLose.* Specifying a number smaller than this will truncate the result while specifying a number larger will pad the end of the output with NAs.

Description:

This function uses the built-in sample function to determine the characters that are lost during simulation. The weights are specified via the *globalLoss* vector, and these weights are not dynamically updated during the simulation.

Function: chooseCharsNoLinkage()

Depends: None

Argument	Default	Description
numCharsToLose		Integer: number of characters per taxon that should be lost during the simulation.
globalLoss		Vector: same length as number of characters in the dataset, representing the global absence of each character in the extinct dataset across all taxa.
maxLength	NULL	Integer: the maximum number of characters to be lost. If specified as NULL, the length of the output will equal the *numCharsToLose.* Specifying a number smaller than this will truncate the result while specifying a number larger will pad the end of the output with NAs.

Description:

This function uses the built-in sample function to determine the characters that are lost during simulation. The weights are specified via the *globalLoss* vector, and these weights are not dynamically updated during the simulation.

Function: adjustSoftCharCap

Depends: None

Argument	Default	Description
charsToRemove		Vector: how many characters a given taxon should lose in the simulation (of length # of taxa* # of simulations) .
numSims		Numeric: number of simulations to perform.
extantDataset		Dataframe: dataset used for analysis.
cap		Numeric: Guaranteed number of characters in every taxa after simulation of character loss.

Description:

Extinct datasets with already high amounts of missing data may be selected to lose all characters. To prevent this from happening, this function limits the simulation of missing data such that each taxon in the study has at least the “cap” amount of characters.

Function: adjustNumCharsToRemove()

Depends: None

Argument	Default	Description
charsToRemove		Vector: how many characters a given taxon should lose in the simulation (of length # of taxa* # of simulations) .
numSims		Numeric: number of simulations to perform.
averageCharactersMostMissing		Numeric: number of missing characters in the group with the most missing data.
numTaxa		Numeric: Number of taxa included in the dataset.
within		Numeric: Ratio of tolerance. Data loss adjustment will cease when the amount of missing data is above threshold (within * averageCharactersMostMissing).
extantDataset		Dataframe: dataset used for analysis.

Description:

Since character matrices of extinct data will likely already contain missing characters, the number of characters to lose should be adjusted. Without this adjustment, some taxa will be simulated to lose characters that are already missing, and thereby under-simulate the desired amount of data loss. This function adjusts the amount of data that should be simulated until the number of characters lost falls within a threshold of the goal.

Function: simulateCharacterLoss()

Depends: createCorrelationMatrix()
createGlobalLoss()
chooseCharsToRemove()
chooseCharsNoLinkage()

Argument	Default	Desciption
extinctDataset		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.
extantDataset		Matrix: Character matrix of extant taxa. Columns represent different taxa, while rows represent different morphological characters. However, missing data is coded with NA while all other values indicate character presence. Note that this is different than the coding scheme used in the extinctDataset.
numSims	100	Integer: how many simulations the algorithm should perform
numCharsToLose	1	Integer: The number of characters to lose per taxon. If a uniform distribution of characters is chosen, each taxon will lose these many characters. If a normal or Poisson distribution is chosen, this number represents the mean number of characters each taxon will lose.
linkage	FALSE	Boolean: When false, random algorithm is used. When true, linkage algorithm is activated. Note that linkage can be activated no matter what distribution of missing data is chosen.
softChars	13	Integer: Number of soft characters in the datasets. These soft characters should be the last rows of the dataset, and they will be preferentially lost first. Including soft characters in the algorithm improves the algorithm's performance.
sigma	NULL	Integer: The standard deviation which is to be used if a normal distribution of characters is selected. If the normal distribution is chosen and no sigma is specified, the code will halt.
normDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a normal distribution.
poisDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a Poisson distribution.
characterAdjustment	FALSE	Boolean: Changing to true will activate **adjustNumCharsToRemove().**
withinPercent	0.05	Numeric: Ratio of tolerance. Data loss adjustment will cease when the amount of missing data is above threshold (within * averageCharactersMostMissing).
cap	NULL	Numeric: Guaranteed number of characters in every taxa after simulation of character loss.

protectedTaxa	0	Numeric Vector: indices of taxa which will not lose data. Used to prevent data loss in groups with the most missing data when using the correction factor.

Description:

Main function to run when simulating character loss. Given the extinct and extant datasets, this function calls other functions that perform the calculations necessary for either the linkage or random character loss algorithm. This function is designed to generate character loss for any number of simulations. The final output of this function is a 3-dimensional array where columns represent different taxa, each row contains the numerical IDs of the characters that are simulated to be lost. This matrix is extended in the z-direction where the z-axis represents simulation number.

Function: removeCharacters()

Depends: None

Argument	Default	Description
taxaNum		Integer: number which specifies which taxon to remove characters from.
dataMatrix		Array: 3D array of extant dataset where every matrix in the z direction is a copy of the full extant dataset. The removeCharacters() function will edit each matrix in the array and remove characters based on the *lossMatrix*.
lossMatrix		Array: 3D array which contain which characters are simulated to be lost. This information is applied to the *dataMatrix*.
simNum		Integer: number which specifies which simulation of character loss to apply to the extant dataset.

Description:

This function applies the *lossMatrix* to the *dataMatrix* for a specified taxon and simulation number. Since the simulations of character loss only generate a listing of which characters should be lost, the removeCharacters() function provides a mechanism to remove these characters from an extant dataset. Removing characters from the extant dataset is done by iteratively calling this function in the applyToExtant() function.

Function: applyToExtant()

Depends: None

Argument	Default	Description
simArray		Array: Array containing the characters which should be lost for each taxon for each simulation.
extantDataset		Matrix: Character matrix of extant taxa. Columns represent different taxa, while rows represent different morphological characters. However, missing data is coded with NA while all other values indicate character presence. Note that this is different than the coding scheme used in the extinctDataset.

Description:

This function iteratively calls the removeCharacters() function and generates missing data according to the simArray.

Function: applyToExtinct()

Depends: None

Argument	Default	Description
simArray		Array: Array containing the characters which should be lost for each taxon for each simulation in the extinct dataset.

Description:

This function iteratively calls the removeCharacters() function and generates missing data according to the simArray.

Function: MATRIXList()

Depends: None

Argument	Default	Description
data		List: A list containing multiple taxon by character matrices.
axes		The number of Principle Coordinate axes to calculate.
method		Based on the vegdist() function in the vegan package. The following distance metrics can be calculated: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao" or "cao".
GroupBins	groupBins	Numeric: A vector containing the number bins. Do not change, brings in information from a global assignment.
GroupsVector	groupsVector	Numeric: A vector containing information about the bin that each time bin belongs to. Do not change, brings in information from a global assignment.

Description:

This function calculates a distance matrix from a taxon by character matrix then performs a Principle Coordinate analysis. Use "manhattan" for discrete characters, "euclidean" for continuous characters. This function will also output a mean PCO matrix for *T*-tests.

The MATRIXList function on the Ordination-Disparity.Source.R is slightly different to the one contained in the CorrectionFunction.R. If using the correction factor, the section below can be ignored. In the Ordination-Disparity.Source.R once the script has calculated PCO, the data can then be divided up into specific taxonomic groups. In script provided, the data are split into three groups, a total group (which was used in the study), but also into additional two taxonomic groupings. As the script requires this division (because it deals with lists), there must always be two groups present here. A group can contain as few as 2 taxa.

```
PCO.Output.Total<-PCO.Output[1:24,]
PCO.Output.Taxonomic.Group.1<-PCO.Output[1:14,]
PCO.Output.Taxonomic.Group.2<-PCO.Output[15:24,]
```

The numbers here refer to the taxa you wish to include in each grouping. There were 24 taxa in the extant dataset, we wanted to include them all in our study. However, we give future users the potential to split up taxa if they wish and run the analysis simultaneously. Set these numbers to your desired divisions of taxa.

In R, when using square brackets the settings are [ROWS, COLUMNS]. The colon acts as a separator. This "`PCO.Output.Taxonomic.Group.1`" will therefore make a subgroup containing taxa 1 *to* 14.

Function: DisparityCalc()

Depends: None

Argument	Default	Description
data		List: A list containing multiple PCO matrices.
axes		The number of Principle Coordinate axes present in ‘data’.
iterations	1000	Number of bootstrap iterations to perform for sample size correction.
C.I.L	0.025	Lower confidence interval.
C.I.U	0.975	Upper confidence interval.

Description:

This function calculates 5 disparity metrics from the imported PCO list: sumo of ranges sum of variances, product of ranges, product of variances, and mean Euclidean distance.

Function: calculate.pco()

Depends: None

Argument	Default	Description
input		Matrix: A taxon by character matrix.
axes		Numeric: The number of Principle Coordinate axes to calculate.
method		Character: Based on the vegdist() function in the vegan package. The following distance metrics can be calculated: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao" or "cao".

Description:

Similar to the MATRIXList() function but does not take in lists of data. Use "manhattan" for discrete characters, "euclidean" for continuous characters.

Function: calculate.pco.list()

Depends: None

Argument	Default	Description
input		List: A list of taxon by character matrices.
axes		Numeric: The number of Principle Coordinate axes to calculate.
method		Character: Based on the vegdist() function in the vegan package. The following distance metrics can be calculated: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao" or "cao".

Description:

Similar to the MATRIXList() function but does not output a mean PCO matrix.

Function: calculate.ordination.difference()

Depends: calculate.pco()
calculate.pco.list()
simulateCharacterLoss()

Argument	Default	Description
pco.data		List: A list of the PCO matrices that have undergone data removal.
no.loss.pco		Matrix: The original PCO matrix calculated from a taxon by character matrix with no data removal.
axes		Numeric: The number of Principle Coordinate axes to calculate.
method		Character: Based on the vegdist() function in the vegan package. The following distance metrics can be calculated: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao" or "cao".

Description:

This function calculates the distance (preferable Euclidean) between taxa that were derived from the original, no data removal PCO analysis, and the taxa that underwent data removal. The script then finds the difference between the unaltered PCO matrix and the list of data removal matrices. In other words, the script calculates how far taxa have moved in space following data removal.

Function: correctionFactor()

Depends: CorrectionFunctions.R

Argument	Default	Desciption
Data		Matrix: Character matrix
numSims	100	Integer: how many simulations the algorithm should perform
numCharsToLose	1	Integer: The number of characters to lose per taxon. If a uniform distribution of characters is chosen, each taxon will lose these many characters. If a normal or Poisson distribution is chosen, this number represents the mean number of characters each taxon will lose.
linkage	FALSE	Boolean: When false, random algorithm is used. When true, linkage algorithm is activated. Note that linkage can be activated no matter what distribution of missing data is chosen.
softChars	13	Integer: Number of soft characters in the datasets. These soft characters should be the last rows of the dataset, and they will be preferentially lost first. Including soft characters in the algorithm improves the algorithm's performance.
sigma	NULL	Integer: The standard deviation which is to be used if a normal distribution of characters is selected. If the normal distribution is chosen and no sigma is specified, the code will halt.
normDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a normal distribution.
poisDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a Poisson distribution.
characterAdjustment	FALSE	Boolean: Changing to true will activate **adjustNumCharsToRemove().**
withinPercent	0.05	Numeric: Ratio of tolerance. Data loss adjustment will cease when the amount of missing data is above threshold (within * averageCharactersMostMissing).
axes	10	The number of Principle Coordinate axes to calculate.
cap	NULL	Numeric: Guaranteed number of characters in every taxa after simulation of character loss.
protectedTaxa	0	Numeric Vector: indices of taxa which will not lose data. Used to prevent data loss in groups with the most missing data when using the correction factor.

Description:

This function wraps all of the functions required to calculate the disparity correction factor and will output a series of files that correspond to the number of bins present in your dataset. These bins will be arranged in order, bin one will be output first, then bin 2, and so on until the final bin. Each of these files will contain rarefied disparity metrics.

Function: DisparityNoCorr()

Depends: calculate.pco()
calculate.pco.list()
simulateCharacterLoss()

Argument	Default	Description
Data		Matrix: A file containing the taxa by character data matrix.
axes	10	Numeric: The number of Principle Coordinate axes present in 'data'.
method	manhattan	Character: Based on the vegdist() function in the vegan package. The following distance metrics can be calculated: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao" or "cao".
iterations	1000	Numeric: Number of bootstrap iterations to perform for sample size correction.
C.I.L	0.025	Numeric: Lower confidence interval.
C.I.U	0.975	Numeric: Upper confidence interval.

Description:

This function wraps all of the functions required to calculate rarefied disparity profiles without any correction factor. The function will output a series of files that correspond to the number of bins present in your dataset. These bins will be arranged in order, bin one will be output first, then bin 2, and so on until the final bin.

Function: DisparityTTest()

Depends: None

Argument	Default	Description
data		Matrix: A PCO matrix with taxa in the first column, time bins in the second column, and the PCO data in the subsequent columns.
replicates	1000	Numeric: Number of iterations to calculate a *t*-distribution.
upper.bound	0.975	Numeric: Upper confidence interval.
lower.bound	0.025	Numeric: Lower confidence interval.
method	All	Character: Four commonly used disparity metrics – sum of ranges, sum of variances, product of ranges, and sum of variances.

Description:

This function uses the *t*-distribution to calculate significant differences in disparity among time bins. This script is based on the t-test script of Anderson and Friedman (2012). We added additional disparity metrics to their script. Output is a symmetric matrix of p-values. Note that the output is not corrected for multiple comparisons so the R function p.adjust() may be required.

Depending on the number of comparisons you perform, you are likely to also need to do some type of *p*-value correction (e.g. Bonferroni). This can be achieved using the "`p.adjust`" function in R. For example, say you were looking at four adjacent time bins (a, b, c, and d) calculated using the "`DisparityTTest`" function and were interested in the adjusted *p*-value between them (a vs. b, b vs. c, c vs. d). Say we obtained p-values of 0.035, 0.0012, and 0.081. By typing the following line of code you can correct for multiple comparisons.

```
p.adjust(c(0.035, 0.0012, 0.081), method="bonferroni")
[1] 0.1050 0.0036 0.2430
```

Each value here has been adjusted by the Bonferroni correction method. In essence, this method multiplies the uncorrected *p*-value by the number of comparisons (in this case, 3 comparisons were made). 0.035 was corrected to 0.1050, 0.0012 was corrected to 0.0036, and so on.

Function: PCOsimulateAndanalyze()

Depends: Ordination-Disparity.Source.R

Argument	Default	Description
importExtinct		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.
importExtant		Matrix: Character matrix of extant taxa. Columns represent different taxa, while rows represent different morphological characters. However, missing data is coded with NA while all other values indicate character presence. Note that this is different than the coding scheme used in the extinctDataset.
importPCO		Matrix: PCO matrix of the original dataset with no data removal.
numSims	100	Integer: how many simulations the algorithm should perform
charsToLose	50	Integer: The number of characters to lose per taxon. If a uniform distribution of characters is chosen, each taxon will lose these many characters. If a normal or Poisson distribution is chosen, this number represents the mean number of characters each taxon will lose.
linkage	TRUE	Boolean: When false, random algorithm is used. When true, linkage algorithm is activated. Note that linkage can be activated no matter what distribution of missing data is chosen.
softChars	2	Integer: Number of soft characters in the datasets. These soft characters should be the last rows of the dataset, and they will be preferentially lost first. Including soft characters in the algorithm improves the algorithm's performance.
sigma	NULL	Integer: The standard deviation which is to be used if a normal distribution of characters is selected. If the normal distribution is chosen and no sigma is specified, the code will halt.
normDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a normal distribution.
poisDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a Poisson distribution.
raw.dist.method	manhattan	Character: The distance method to calculate inter-taxon distances from their discrete characters.
pco.dist.method	euclidean	Character: The distance method used to calculate inter-taxon distances in PCO space.
axes	10	Numeric: Number of PCO axes to calculate.

Description: This function wraps all of the functions required to calculate the changes in PCO space occupation as data are removed. The function outputs the sum of squared distances for each simulation between the original position of taxa with no missing data, and their new location with data removal.

Function: plot.pco()

Depends: PCOsimulateAndanalyze()

Argument	Default	Description
pco.data		List: PCO matrices output from **PCOsimulateAndanalyze()** that have undergone data removal. All simulations are contained in the list PCO.Sim. PCO.Sim was set as a global assignment hence exists out of the **PCOsimulateAndanalyze()** function.
no.loss.pco		Matrix: PCO matrix of taxa with no data correction.
Add	FALSE	Boolean: The first time this function is run Add must be set to false to set up the limits of the chart and plot the position of taxa that have undergone no data removal. Once the first plot it set up, change Add to true in order to plot the output from **PCOsimulateAndanalyze()** contained in the PCO.Sim vector.
xlimit		Numeric: Similar to xlim – provides the limits of the x-axis plots.
ylimit		Numeric: Similar to ylim – provides the limits of the y-axis plots.
colour	red	Numeric: Number of PCO axes to calculate.

Description:

This function plots PCO spaces of the datasets that underwent data removal, and the original PCO matrix with no data removal as a comparison. PCOsimulateAndanalyze() will output a list with the vector name PCO.Sims that can be plotted to see how much taxa have moved in morphospace following data removal. Note that PCO.Sims will contain all the simulations and so plotting all of these may make the plot cluttered. To avoid this, try plotting just a few of the simulations.

This could be done with - PCO.Sims.Reduce <- list(PCO.Sims[[1]], PCO.Sims[[2]])

Function: DisparitysimulateAndanalyze()

Depends: Ordination-Disparity.Source.R

Argument	Default	Desciption
importExtinct		Matrix: Character presence/absence matrix of extinct taxa. Columns represent different taxa, while rows represent different morphological characters. Values of 1 indicate character presence while values of 0 indicate character absence.
importExtant		Matrix: Character matrix of extant taxa. Columns represent different taxa, while rows represent different morphological characters. However, missing data is coded with NA while all other values indicate character presence. Note that this is different than the coding scheme used in the extinctDataset.
numSims	10	Integer: how many simulations the algorithm should perform
charsToLose	1	Integer: The number of characters to lose per taxon. If a uniform distribution of characters is chosen, each taxon will lose these many characters. If a normal or Poisson distribution is chosen, this number represents the mean number of characters each taxon will lose.
linkage	FALSE	Boolean: When false, random algorithm is used. When true, linkage algorithm is activated. Note that linkage can be activated no matter what distribution of missing data is chosen.
softChars	0	Integer: Number of soft characters in the datasets. These soft characters should be the last rows of the dataset, and they will be preferentially lost first. Including soft characters in the algorithm improves the algorithm's performance.
sigma	NULL	Integer: The standard deviation which is to be used if a normal distribution of characters is selected. If the normal distribution is chosen and no sigma is specified, the code will halt.
normDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a normal distribution.
poisDist	FALSE	Boolean: Changing the argument to true will distribute the number of characters lost among taxa using a Poisson distribution.
axes	10	Numeric: Number of PCO axes to calculate.

Description:

This function wraps all of the functions required to remove data from a taxon – character matrix and then calculate disparity. The function will place all the data from an analysis into a text file titled Disparity.output.txt and can continually update as more simulations are run.

See MATRIXList() for information on how to break up into taxonomic groups,
Depending on how many groups you set-up in MATRIXList will govern whether you need to alter some of the disparity code. The DisparityList deposits the disparity results into a series of empty vectors once calculated.
Each metric will have a section that looks like this. In the example script shown below, this code is for

saving the sum of ranges data.

```
#Refer to README for changing empty vectors
Disparity.Out<-cbind(Final.Mean[1], Quant.Vecs[1,1], Quant.Vecs[2,1],
 Final.Mean[2], Quant.Vecs[1,2], Quant.Vecs[2,2], Final.Mean[3], Quant.Vecs[1,3],
 Quant.Vecs[2,3])
```

You can search within the script for "`#Refer to README for changing empty vectors`". It will appear five times.

This code is currently set-up to receive three groups from the MATRIXList function. To set it up for two groups, or more than three, you will need to make some alterations to each metric's code.

To set up for two...

```
Disparity.Out<-cbind(Final.Mean[1], Quant.Vecs[1,1], Quant.Vecs[2,1],
 Final.Mean[2], Quant.Vecs[1,2], Quant.Vecs[2,2])
```

To set up for more than two...

```
Disparity.Out<-cbind(Final.Mean[1], Quant.Vecs[1,1], Quant.Vecs[2,1],
 Final.Mean[2], Quant.Vecs[1,2], Quant.Vecs[2,2], Final.Mean[3], Quant.Vecs[1,3],
 Quant.Vecs[2,3], Final.Mean[4], Quant.Vecs[1,4], Quant.Vecs[2,4])
```

Essentially, you will need to continue to duplicate the "`Final.Mean[`*n*`], Quant.Vecs[1,`*n*`], Quant.Vecs[2,`*n*`]`" line until *n* matches the number of groups you assigned in the MATRIXList section.

The same principle applies to the other metrics. Here's Product of variances as an example.

```
P.Disparity.Out.V<-cbind(P.Final.Mean.V[1], P.Quant.Vecs.V[1,1],
P.Quant.Vecs.V[2,1], P.Final.Mean.V[2], P.Quant.Vecs.V[1,2], P.Quant.Vecs.V[2,2],
P.Final.Mean.V[3], P.Quant.Vecs.V[1,3], P.Quant.Vecs.V[2,3], P.Final.Mean.V[n],
P.Quant.Vecs.V[1,n], P.Quant.Vecs.V[2,n])
```

If this section is incorrectly defined then the error output will likely be a "`subscript out of bounds`" type error.

The disparity data will be output as a separate text file (currently "`disparity.output.txt`", a space delimited text file). The first column will contain the mean number of characters removed during a simulation and the rest of that row will contain the disparity data pertaining to that simulation. The final column will be the number of PCO axes included in the simulation. The number of columns in the output will depend on how many groups went through the simulation. In the mammal example, three groups were tested. Each group would have a separate column for the mean, lower confidence interval (LCI), and upper confidence interval (UCI) for a disparity metric. Five disparity metrics will be calculated for each group (sum of ranges, sum of variances, product of ranges, product of variances, and the mean pairwise distance). Therefore, the total number of columns will consist of 5 disparity metrics with each having a mean, LCI, and UCI for each group, the number of PCO axes, and the mean number of characters removed. In our example there were three groups, there was a total of 47. This can be broken down to 5 disparity metrics * 3 values (mean, LCI, UCI) * 3 taxonomic groups + mean number of characters removed + number of PCO axes.