Package 'rcicr'

July 13, 2016

Type Package

Title Reverse-Correlation Image-Classification Toolbox

Version 0.3.4.1

Date 2016-07-13

URL http://ron.dotsch.org/rcicr

Author Ron Dotsch <r.dotsch@uu.nl>

Maintainer Ron Dotsch <r.dotsch@uu.nl>

Description

Functions to generate stimuli and analyze data of reverse correlation image classification experiments (psychophysical tasks aimed at visualizing cognitive mental representations of faces).

License GPL-2

Imports matlab, aspace, jpeg, dplyr, scales, utils

Repository CRAN

Repository/R-Forge/Project rcicr

Repository/R-Forge/Revision 44

Repository/R-Forge/DateTimeStamp 2016-07-13 08:51:07

Date/Publication 2016-07-13 14:52:26

NeedsCompilation no

R topics documented:

`rcicr-package` *Reverse-Correlation Image-Classification Toolbox*

Description

Toolbox with functions to generate stimuli and analyze data of reverse correlation image classification experiments. Reverse correlation is a psychophysical technique originally derived from signal detection theory. This package focuses on visualizing internal representations of participants using visual stimuli in a perceptual taks.

Details

Package:	rcicr
Type:	Package
Version:	0.3.2
Date:	2015-03-18
License:	GPL-2

Generating stimuli

Load the package with `library(rcicr)`. Then generate stimuli with:

```
generateStimuli2IFC(base_face_files, n_trials = 770)
```

This will generate stimuli for 770 trials of a 2 images forced choice reverse correlation image classification task with sinusoid noise. By default the stimuli will have a resolution of 512 x 512 pixels. The stimuli will be saved as jpegs to a folder called stimuli in your current working directory, and an .Rdata file will be saved that contains the stimulus parameters necessary for analysis.

The `base_face_files` argument is a list of jpegs that should be used as base images for the stimuli. The base_face_files variable might look like this:

```
base_face_files <- list('male'='male.jpg', 'female'='female.jpg')
```

For each jpeg a set of stimuli will be created using the same noise patterns as for the other sets. The jpeg should have the resolution that you want the stimuli to have. By default this should be 512 x 512 pixels. If you want a different size, resize your base image to either 128 x 128 or 256 x 256 for smaller stimuli, or 1024 x 1024 for bigger stimuli. In that case, also set the `img_size` parameter accordingly.

You are now ready to collect data with the stimuli you just created. The stimuli are named according to their sequence number when generating and whether the original noise is superimposed or the negative/inverted noise. Stimuli with the same sequence number should be presented side by side

in the same trial. Record which stimulus a participant selected at any given trial (the original, or the inverted). At the very least be sure that in your data file the connection can be made between the response key of the participant and which stimulus was selected on each trial. Use any presentation software you like (I recommend python-based open source alternatives, like PsychoPy, Expyriment, or OpenSesame).

Data analysis

Analyzing reverse correlation data is all about computing classification images. Use the following function for your data collected using the 2 images forced choice stimuli:

```
ci <- generateCI2IFC(stimuli, responses, baseimage, rdata)
```

The `stimuli` paramater should be a vector containing the sequence numbers of the stimuli that were presented in the trials of the task. The `responses` parameter contains, in the order of the stimuli vector, the response of a participant to those stimuli (coded 1 if the original stimulus was selected and -1 if the inverted stimulus was selected). The `baseimage` parameter is a string specifying which base image was used (not the file name, but the name in the list of `base_face_files`. So for the stimuli generated above, either `'male'` or `'female'`, depending on which set of stimuli was presented to the participant whose data you're analyzing). Finally, rdata is a string pointing to the .RData file that was created automatically when you generated the stimuli. It contains the parameters for each stimulus, necessary to create the classification image.

By default jpg's of the classification images will be saved automatically. The returned values can be used later to optimally rescale the noise relative to the base image. For instance, if you have a list of ci's from various participants (i.e., a list of the values returned by several calls to `generateCI2IFC`, one for each participant), you can use the autoscale function to generate classification images that are scaled identically and therefore straightforward to compare:

```
scaled_cis <- autoscale(cis, saveasjpegs = TRUE)
```

Computing CIs for many participants or conditions

Data analysis as described above can be automatized for a batch of participants or conditions using `batchGenerateCI2IFC`. Please see instructions for that function.

Note

Currently, the package is still in alpha stage. Much may still change. It only supports 2 Image Forced Choice tasks, although the underlying functions can be used for other versions of the reverse correlation task. It also only supports sinusoid noise. In the future, it will support Gaussian white noise, as well as additional variants of the task.

If you use this package for your experiments, please cite the package in your publications. Use `citation('rcicr')` to print the appropriate citation for the current version of the package.

Author(s)

Ron Dotsch <rdotsch@gmail.com> (http://ron.dotsch.org/) Maintainer: Ron Dotsch <rdotsch@gmail.com>

References

Dotsch, R., & Todorov, A. (2012). Reverse correlating social face perception. Social Psychological and Personality Science, 3 (5), 562-571.

Dotsch, R., Wigboldus, D. H. J., Langner, O., & Van Knippenberg, A. (2008). Ethnic out-group faces are biased in the prejudiced mind. Psychological Science, 19, 978-980.

Examples

```
#simple examples will be added soon.
```

`autoscale` *Determines optimal scaling constant for a list of ci's*

Description

Determines optimal scaling constant for a list of ci's

Usage

```
autoscale(cis, saveasjpegs = TRUE, targetpath = "./cis")
```

Arguments

`cis`	List of cis, each of which are a list containing the pixel matrices of at least the noise pattern (`$ci`) and if the noise patterns need to be written to jpegs, als the base image (`$base`)
`saveasjpegs`	Boolean, when set to true, the autoscaled noise patterns will be combined with their respective base images and saved as jpegs (using the key of the list as name)
`targetpath`	Optional string specifying path to save jpegs to (default: ./cis)

Value

List of scaled noise patterns and determind scaling factor

`batchGenerateCI` *Generates multiple classification images by participant or condition*

Description

Generate classification image for any reverse correlation task that displays independently generated alternatives.

Usage

```
batchGenerateCI(data, by, stimuli, responses, baseimage, rdata,
  saveasjpeg = TRUE, targetpath = "./cis", label = "", antiCI = FALSE,
  scaling = "autoscale", constant = 0.1)
```

Arguments

data	Data frame
by	String specifying column name that specifies the smallest unit (participant, condition) to subset the data on and calculate CIs for
stimuli	String specifying column name in data frame that contains the stimulus numbers of the presented stimuli
responses	String specifying column name in data frame that contains the responses coded 1 for original stimulus selected and -1 for inverted stimulus selected.
baseimage	String specifying which base image was used. Not the file name, but the key used in the list of base images at time of generating the stimuli.
rdata	String pointing to .RData file that was created when stimuli were generated. This file contains the contrast parameters of all generated stimuli.
saveasjpeg	Boolean stating whether to additionally save the CI as jpeg image
targetpath	Optional string specifying path to save jpegs to (default: ./cis)
label	Optional string to insert in file names of jepgs to make them easier to identify
antiCI	Optional boolean specifying whether antiCI instead of CI should be computed
scaling	Optional string specifying scaling method: `none`, `constant`, `independent` or `autoscale` (default)
constant	Optional number specifying the value used as constant scaling factor for the noise (only works for `scaling='constant'`)

Details

This funcions saves the classification images by participant or condition as jpeg to a folder and returns the CIs.

Value

List of classification image data structures (which are themselves lists of pixel matrix of classification noise only, scaled classification noise only, base image only and combined)

---

batchGenerateCI2IFC *Generates multiple 2IFC classification images by participant or condition*

Description

Generate classification image for 2 images forced choice reverse correlation task.

Usage

```
batchGenerateCI2IFC(data, by, stimuli, responses, baseimage, rdata,
  saveasjpeg = TRUE, targetpath = "./cis", antiCI = FALSE,
  scaling = "autoscale", constant = 0.1, label = "")
```

Arguments

data	Data frame
by	String specifying column name that specifies the smallest unit (participant, condition) to subset the data on and calculate CIs for
stimuli	String specifying column name in data frame that contains the stimulus numbers of the presented stimuli
responses	String specifying column name in data frame that contains the responses coded 1 for original stimulus selected and -1 for inverted stimulus selected.
baseimage	String specifying which base image was used. Not the file name, but the key used in the list of base images at time of generating the stimuli.
rdata	String pointing to .RData file that was created when stimuli were generated. This file contains the contrast parameters of all generated stimuli.
saveasjpeg	Boolean stating whether to additionally save the CI as jpeg image
targetpath	Optional string specifying path to save jpegs to (default: ./cis)
antiCI	Optional boolean specifying whether antiCI instead of CI should be computed
scaling	Optional string specifying scaling method: `none`, `constant`, `matched`, `independent`, or `autoscale` (default)
constant	Optional number specifying the value used as constant scaling factor for the noise (only works for `scaling='constant'`)
label	Optional string to insert in file names of jepgs to make them easier to identify

Details

This funcions saves the classification images by participant or condition as jpeg to a folder and returns the CIs.

Value

List of classification image data structures (which are themselves lists of pixel matrix of classification noise only, scaled classification noise only, base image only and combined)

---

computeCumulativeCICorrelation

Computes cumulative trial CIs correlations with final/target CI

Description

Computes cumulative trial CIs correlations with final/target CI.

Usage

```
computeCumulativeCICorrelation(stimuli, responses, baseimage, rdata,
  targetci = list(), step = 1)
```

Arguments

stimuli	Vector with stimulus numbers (should be numeric) that were presented in the order of the response vector. Stimulus numbers must match those in file name of the generated stimuli
responses	Vector specifying the responses in the same order of the stimuli vector, coded 1 for original stimulus selected and -1 for inverted stimulus selected.
baseimage	String specifying which base image was used. Not the file name, but the key used in the list of base images at time of generating the stimuli.
rdata	String pointing to .RData file that was created when stimuli were generated. This file contains the contrast parameters of all generated stimuli.
targetci	List Target CI object generated with rcicr functions to correlate cumulative CIs with
step	Step size in sequence of trials to compute correlations with

Details

Use for instance for plotting curves of trial-final/target CI correlations to estimate how many trials are necessary in your task

Value

Vector containing correlation between cumulative CI and final/target CI

---

generateCI *Generates classification image*

Description

Generate classification image for for any reverse correlation task.

Usage

```
generateCI(stimuli, responses, baseimage, rdata, saveasjpeg = TRUE,
  filename = "", targetpath = "./cis", antiCI = FALSE,
  scaling = "independent", constant = 0.1)
```

Arguments

stimuli	Vector with stimulus numbers (should be numeric) that were presented in the order of the response vector. Stimulus numbers must match those in file name of the generated stimuli
responses	Vector specifying the responses in the same order of the stimuli vector, coded 1 for original stimulus selected and -1 for inverted stimulus selected.
baseimage	String specifying which base image was used. Not the file name, but the key used in the list of base images at time of generating the stimuli.

rdata	String pointing to .RData file that was created when stimuli were generated. This file contains the contrast parameters of all generated stimuli.
saveasjpeg	Boolean stating whether to additionally save the CI as jpeg image
filename	Optional string to specify a file name for the jpeg image
targetpath	Optional string specifying path to save jpegs to (default: ./cis)
antiCI	Optional boolean specifying whether antiCI instead of CI should be computed
scaling	Optional string specifying scaling method: none, constant, matched, or independent (default)
constant	Optional number specifying the value used as constant scaling factor for the noise (only works for scaling='constant')

Details

This funcions saves the classification image as jpeg to a folder and returns the CI. Your choice of scaling matters. The default is 'matched', and will match the range of the intensity of the pixels to the range of the base image pixels. This scaling is non linear and depends on the range of both base image and noise pattern. It is truly suboptimal, because it shifts the 0 point of the noise (that is, pixels that would have not changed base image at all before scaling may change the base image after scaling and vice versa). It is however the quick and dirty way to see how the CI noise affects the base image.

For more control, use 'constant' scaling, where the scaling is independent of the base image and noise range, but where the choice of constant is arbitrary (provided by the user with t the constant parameter). The noise is then scale as follows: scaled <- (ci + constant) / (2*constant). Note that pixels can take intensity values between 0 and 1 If your scaled noise exceeds those values, a warning will be given. You should pick a higher constant (but do so consistently for different classification images that you want to compare). The higher the constant, the less visible the noise will be in the resulting image.

When creating multiple classification images a good strategy is to find the lowest constant that works for all classification images. This can be automatized using the autoscale function.

Value

List of pixel matrix of classification noise only, scaled classification noise only, base image only and combined

---

generateCI2IFC *Generates 2IFC classification image*

Description

Generate classification image for 2 images forced choice reverse correlation task. This function exists for backwards compatibility. You can also just use generateCI(), which this function wraps.

Usage

```
generateCI2IFC(stimuli, responses, baseimage, rdata, saveasjpeg = TRUE,
  filename = "", targetpath = "./cis", antiCI = FALSE,
  scaling = "independent", constant = 0.1)
```

Arguments

stimuli	Vector with stimulus numbers (should be numeric) that were presented in the order of the response vector. Stimulus numbers must match those in file name of the generated stimuli
responses	Vector specifying the responses in the same order of the stimuli vector, coded 1 for original stimulus selected and -1 for inverted stimulus selected.
baseimage	String specifying which base image was used. Not the file name, but the key used in the list of base images at time of generating the stimuli.
rdata	String pointing to .RData file that was created when stimuli were generated. This file contains the contrast parameters of all generated stimuli.
saveasjpeg	Boolean stating whether to additionally save the CI as jpeg image
filename	Optional string to specify a file name for the jpeg image
targetpath	Optional string specifying path to save jpegs to (default: ./cis)
antiCI	Optional boolean specifying whether antiCI instead of CI should be computed
scaling	Optional string specifying scaling method: `none`, `constant`, `matched`, or `independent` (default)
constant	Optional number specifying the value used as constant scaling factor for the noise (only works for `scaling='constant'`)

Details

This funcions saves the classification image as jpeg to a folder and returns the CI. Your choice of scaling matters. The default is `'matched'`, and will match the range of the intensity of the pixels to the range of the base image pixels. This scaling is non linear and depends on the range of both base image and noise pattern. It is truly suboptimal, because it shifts the 0 point of the noise (that is, pixels that would have not changed base image at all before scaling may change the base image after scaling and vice versa). It is however the quick and dirty way to see how the CI noise affects the base image.

For more control, use `'constant'` scaling, where the scaling is independent of the base image and noise range, but where the choice of constant is arbitrary (provided by the user with t the `constant` parameter). The noise is then scale as follows: `scaled <- (ci + constant) / (2*constant)`. Note that pixels can take intensity values between 0 and 1 If your scaled noise exceeds those values, a warning will be given. You should pick a higher constant (but do so consistently for different classification images that you want to compare). The higher the constant, the less visible the noise will be in the resulting image.

When creating multiple classification images a good strategy is to find the lowest constant that works for all classification images. This can be automatized using the `autoscale` function.

Value

List of pixel matrix of classification noise only, scaled classification noise only, base image only and combined

---

`generateCINoise` *Generate classification image noise pattern based on set of stimuli (matrix: trials, parameters), responses (vector), and sinusoid*

Description

Generate classification image noise pattern based on set of stimuli (matrix: trials, parameters), responses (vector), and sinusoid

Usage

```
generateCINoise(stimuli, responses, p)
```

Arguments

`stimuli`	Matrix with one row per trial, each row containing the 4092 parameters for the original stimulus
`responses`	Vector containing the response to each trial (1 if participant selected original , -1 if participant selected inverted; this can be changed into a scale)
`p`	3D patch matrix (generated using `generateNoisePattern()`)

Value

The classification image as pixel matrix

`generateGabor` *Generate single gabor patch*

Description

Generate single gabor patch

Usage

```
generateGabor(img_size, cycles, angle, phase, sigma, contrast)
```

Arguments

img_size	Integer specifying size of gabor patch in number of pixels
cycles	Integer specifying number of cycles the sinusoid should span
angle	Value specifying the angle (rotation) of the sinusoid
phase	Value specifying phase of sinusoid
sigma	of guassian mask on top of sinusoid
contrast	Value between -1.0 and 1.0 specifying contrast of sinusoid

Value

The sinusoid image with size img_size.

Examples

```
generateSinusoid(512, 2, 90, pi/2, 1.0)
```

---

generateNoiseImage *Generate single noise image based on parameter vector*

Description

Generate single noise image based on parameter vector

Usage

```
generateNoiseImage(params, p)
```

Arguments

params	Vector with each value specifying the contrast of each patch in noise
p	3D patch matrix (generated using generateNoisePattern())

Value

The noise pattern as pixel matrix

Examples

```
#params <- rnorm(4092) # generates 4092 normally distributed random values
#s <- generateNoisePattern(img_size=256)
#noise <- generateNoiseImage(params, p)
```

generateNoisePattern *Generate sinusoid noise pattern*

Description

Generate sinusoid noise pattern

Usage

```
generateNoisePattern(img_size = 512, nscales = 5, noise_type = "sinusoid",
  sigma = 25, pre_0.3.0 = FALSE)
```

Arguments

img_size	Integer specifying size of the noise pattern in number of pixels
nscales	Integer specifying the number of incremental spatial scales. Defaults to 5. Higher numbers will add higher spatial frequency scales.
noise_type	String specifying noise pattern type (defaults to sinusoid; other options: gabor).
sigma	Number specifying the sigma of the Gabor patch if noise_type is set to gabor (defaults to 25)
pre_0.3.0	Boolean specifying whether the noise pattern should be created in a way compatible with older versions of rcicr (< 0.3.0). If you are starting a new project, you should keep this at the default setting (FALSE). There is no reason to set this to TRUE, with the sole exception to recreate behavior of rcicr prior to version 0.3.0.

Value

List with two elements: the 3D noise matrix with size img_size, and and indexing matrix with the same size to easily change contrasts.

Examples

```
generateNoisePattern(256)
```

generateSinusoid *Generate single sinusoid patch*

Description

Generate single sinusoid patch

Usage

```
generateSinusoid(img_size, cycles, angle, phase, contrast)
```

Arguments

img_size	Integer specifying size of sinusoid patch in number of pixels
cycles	Integer specifying number of cycles sinusoid should span
angle	Value specifying the angle (rotation) of the sinusoid
phase	Value specifying phase of sinusoid
contrast	Value between -1.0 and 1.0 specifying contrast of sinusoid

Value

The sinusoid image with size img_size.

Examples

```
generateSinusoid(512, 2, 90, pi/2, 1.0)
```

---

generateStimuli2IFC *Generates 2IFC stimuli*

Description

Generate stimuli for 2 images forced choice reverse correlation task.

Usage

```
generateStimuli2IFC(base_face_files, n_trials = 770, img_size = 512,
  stimulus_path = "./stimuli", label = "rcic", use_same_parameters = TRUE,
  seed = 1, maximize_baseimage_contrast = TRUE, noise_type = "sinusoid",
  nscales = 5, sigma = 25)
```

Arguments

base_face_files	List containing base face file names (jpegs) used as base images for stimuli
n_trials	Number specifying how many trials the task will have (function will generate two images for each trial per base image: original and inverted/negative noise)
img_size	Number specifying the number of pixels that the stimulus image will span horizontally and vertically (will be square, so only one integer needed)
stimulus_path	Path to save stimuli and .Rdata file to
label	Label to prepend to each file for your convenience
use_same_parameters	Boolean specifying whether for each base image, the same set of parameters is used, or unique set is created for each base image
seed	Integer seeding the random number generator (for reproducibility)

`maximize_baseimage_contrast`
: Boolean specifying wheter the pixel values of the base image should be rescaled to maximize its contrast.

`noise_type`
: String specifying noise pattern type (defaults to `sinusoid`; other options: `gabor`).

`nscales`
: Integer specifying the number of incremental spatial scales. Defaults to 5. Higher numbers will add higher spatial frequency scales.

`sigma`
: Number specifying the sigma of the Gabor patch if noise_type is set to `gabor` (defaults to 25)

Details

Will save the stimuli as jpeg's to a folder, including .Rdata file needed for analysis of data after data collection. This .Rdata file contains the parameters that were used to generate each stimulus.

Value

Nothing, everything is saved to files.

`simulateNoiseIntensities` *Simulate pixel intensity range for noise*

Description

Simulate pixel intensity range for noise

Usage

```
simulateNoiseIntensities(nrep = 1000, img_size = 512)
```

Arguments

`nrep`
: Number of replications

`img_size`
: Size of noise pattern in pixels (one value equal for width and height)

Value

Matrix with range of noise intensities for each replication

Index