Package 'AIG'

May 21, 2018

Type Package

Title Automatic Item Generator

Version 0.1.9

Date 2018-05-18

Maintainer Bao Sheng Loe (Aiden) <bsl28@cam.ac.uk>

Description A collection of Automatic Item Generators used mainly for psychological research. This package can generate linear syllogistic reasoning, arithmetic and 2D/3D/Double 3D spatial reasoning items. It is recommended for research purpose only.

License GPL-3

Imports mgcv, stats, dplyr, rgl, magrittr

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Author Bao Sheng Loe (Aiden) [aut, cre, cph],
David Condon [ctb, cph],
Francis Smart [ctb, cph]

Repository CRAN

Date/Publication 2018-05-21 18:51:28 UTC

R topics documented:

AIG *AIG: A package for studying Automatic Item Generators.*

Description

The AIG package contains generators for studying automatically generated item types.

Details

The package will include several generators for psychological research purposes. Currently, it can only generate linear syllogistic reasoning ,arithmetic, and 2D/3D spatial reasoning items.

Linear Syllogism Generator

This function creates linear syllogsitic reasoning items according to certain specifications given by the arguments.
`lisy`

Arithmetic Generator

This function creates arithmetic items according to certain specifications given by the arguments.
`arith`

Spatial 2D Reasoning Generator

This function creates spatial 2D reasoning items according to certain specifications given by the arguments.
`spatial2d`

Spatial 3D Reasoning Generator

This function creates spatial 3D reasoning items according to certain specifications given by the arguments.
`spatial3d`

The `spatial3d_mirror` creates the mirror image of the figure created using the spatial3d function.

The `spatial3dDouble` creates the 2 figures in the image using the spatial3dDouble function.

The `spatial3d_mirrorDouble` creates variations of the figures in a single image based on one of the four methods.

References

Sternberg, R. J. (1980). Representation and process in linear syllogistic reasoning. Journal of Experimental Psychology: General, 109(2), 119.

Embretson, S. E. (1999). Generating items during testing: Psychometric issues and models. Psychometrika, 64(4), 407-433.

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement*, 14(3), 237-245.

`arith` *ICAR Arithmetic Generator*

Description

A basic arithmetic item generator for ICAR.

Usage

```
arith(numProbs = 5, numOps = 1, magnitude = 1, arithOps = "add",
  allowNeg = FALSE, sameValue = FALSE, order = "mixed")
```

Arguments

`numProbs`	Number of desired problems
`numOps`	sets the number of operations in the stimuli (range of values is 1 to 5)
`magnitude`	sets the range of values for the integers in the stimuli by powers of 10 (range of values is 1 to 5)
`arithOps`	There are two options in this argument, either 'add' or 'subtract', "multiply.
`allowNeg`	Allow the use of negative values. Only for "add" arithmetic operator. Not allowed for "subtract".
`sameValue`	Deciding if you want the values to be the same or different values.
`order`	There are three options to order the values presented, ("mixed", "descending", "ascending"). Does not matter when same Value = TRUE.

Details

The arithmetic function can currently generate additive and substractive operators. For the additive operators, you have the option of using both negative and positive numerical values. For the substractive operators, you are currently restricted to only using positive numerical values. There are also options for you to select the order of the values. It can be in ascending, descending or mixed order. Currently, there are 4 types of distractors that are generated now. 1a,1b distractor types are developed for addition and subtraction questions, whereas, 2a,2b distractor types are developed for multiplication questions.

Distactor type 1a is used for all arithmetic operators. Distractor type 1a is generated such that the values are very near +1/+2/+3 or -1/-2/-3 from the actual answers. Distractor type 1b is generated

such as it takes values from a range of values (i.e. number of values * (max(magnitude-1) + 1)). In this instance, distractor type 1b cannot be used in multiplication items because the range will be too big. Hence, elimination of distractors becomes very easy.

Distractor type 2a is used in multiplication questions where the values are not the same. The distractors are generated such that one of the numbers are randomly assigned a value of 1,2 or 3. The product of the new values are considerd as one distractor. It will repeat until 5 unique distractors are created. If the multiplication question uses the same value, then a slightly different approach is adopted. In this case, the numbers are randomly drawn three times before we calcuate the product of the values. Again, this process is repeated until 5 unique distractors are achieved.

Value

Question The arithmetic questions that are automatically generated.

Code First digit is the magnitude. Subsequent digits correspond to the positional values in the question. 0 is positive value, 1 is negative value.

Answers The answer to the question generated.

Response Options Eight response options.

Author(s)

Aiden Loe and David Condon

See Also

lisy, spatial2d, spatial3d

Examples

```
arith(numProbs=4, numOps = 4, magnitude = 3, arithOps = "subtract",
 allowNeg = FALSE, sameValue=FALSE, order = "mixed")
```

---

lisy *Linear Syllogism Generator*

Description

This function generates linear syllogistic reasoning items. This is for research purposes.

Usage

```
lisy(seed = 1, nclues = 4, nspread = 5, clone = NULL,
  incidental = "names", linear = FALSE, antonym = "both", ninfer = 1,
  direct = "ob", Ndist = 4, dist = "mixed", distprob = 0.5,
  itemSet = "random", items = NULL, scales = NULL)
```

Arguments

seed	Generates the same question again from local computer.
nclues	Generates the number of sentences to make up the item.
nspread	Calculates the spread of possible incidentals in total.
clone	Null means that every generated item may or may not have a different position in the inference. If given a numeric value, then the items will have the same inference position.
incidental	Tells the function whether the item features are 'names' or 'objects'.
linear	Linear is the line of thought. If linear is TRUE, then the comparions are going in either forward or backward direction. The direction is based on the direct argument. When linear is FALSE, some extra incidentals (e.g. random names) that are not useful are also generated in the statement. The hypothesis is that people may have to create two or more mental models, completing with each other to find the correct answer amongst the response options.
antonym	Determine whether to use both antonyms ('both') or only one type ("first" or "second").
ninfer	Generate answers that requires a X amount of inference from the items. Up to 3 is the maximum.
direct	Deciding on whether the statements are organised in an ordered ("of" = ordered forward / "ob" = ordered backward), randomly selected ordered ('alt' = alternative) fashion or unordered('mixed') fashion. Note. 'alt' can only be used when ninfer is equal to 3.
Ndist	Returns the number of distractors per question.
dist	Select the type of distractors. You have three options ('mixed', 'invalid','false'). If dist='false', then the number of false distractors must be less than the number of clues by 1.
distprob	Calculates the number of comparison variation for the distractors.
itemSet	This is the choice of itemset you want. If itemSet='random' then the generator will randomly select one ('People', 'Fruits', 'Superheroes'). Change itemset='own' if you are using your own item set.
items	Input own incidental features, with at least 10 of them. Default incidental features (either names, objects or superheros) are used when items = NULL.
scales	Input own antonyms. At least 2 antonyms (i.e."bigger","smaller"). Default antonyms are used when scales = NULL.

Details

There are several things to note. To use own item set, please have at least 10 incident features (e.g. 10 different names). In order for antonyms comparison to work, please ensure that you have at least 2 antonyms The function will stop if the criteria is not met. The genearation of items are slower if you have a huge item set (e.g. In the millions!).

When nspread and nclue is = 3. This means that there are 3 sentences, and only 3 names. This makes it impossible to generate an invalid distractor. As such, only the false distractors will be created. Since there are only three clues, then at most 2 false distractors can be created.

When nspread and nclues are the same, all the names of the distractors will be taken from the names that are used in the clues. As nspread value increases, the likelihood of having names not taken from the clues increases. Making the distractors fairly easy as there is a higher likelihood that the names taken from the matrix might not appear in the clues. Hence, keeping the value of nspread and nclue as close as possible is recommended.

This function only generates items that requires up to 3 inferences. As the required inferences increases, then number of clues needed also increases. Inference is the implied comparison between sentences which allows the test taker to make an inform decision. When ninfer = 1 and the antonym is declared as either 'first' or 'second', then the correct answer will always be the opposite of the antonym used in the sentence. When ninfer = 2, the correct answer will be in the right direction.

Direct is the direction of the clue provided. In the function, the argument direct = "ob" means that solving the items requires the test taker to work 'ordered backward'. If it is 'of', it means 'ordered forward'. Finally, if it is 'alt', then it means the clues are not in order. direct = 'alt' can only be used when ninfer = 3. The clues provided in the question are useful for the first three arguments. However, when direct = "mixed", this means that the sentences are randomly placed. Making it difficult for the participant to form a linear array of the item. In this case, the clues are not useful, so NA is given in the output instead.

When linear = TRUE, the sentence structure will be in a linear order. i.e. when antonym = "first" or antonym = "second" and direct = "of", the names will follow in a linear sequence (A > B, B > C, C > D). However, when antonym = "first" or antonym = "second" and direct = "ob", then the sentence structure changes to becomes (C > D, B > C, A > B). When antonym = "both", the names will still follow a linear sequence either forward or backward, but the antonyms will interchange between sentence (i.e. A > B, C < B, C > D). Nevertheless, 'A' will always be bigger than the 'D'. The argument direct = 'alt' cannot be used when linear = TRUE.

When linear = TRUE and infer = 3, the last sentence will not be one of the clues for the inference. If you want to study distance effect, then what is recommended is to generate the items with ninfer = 3, and remove the last clue in the sentence structure.

When linear = FALSE, there might be a possible of having just a single mental model, or having completing mental models. This is random and will depend on the seed selected. The statements are randomly placed but in the general order of the direct argument provided. The clues remain in the order given based on the direct argument.

When distprob = 0.5, the distribution of the antonym for the distractors will be mixed. When distprob is either 1 or 0, then only one of the two antonym will be used. This is only used if one wishes to study distractor analysis.

Author(s)

Aiden Loe and Francis Smart

References

Leth-Steensen, C., & Marley, A. A. J. (2000). A model of response time effect in symbolic comparison. *Psychological Review, 107*, 62-100.

Sternberg, R. J. (1980). Representation and process in linear syllogistic reasoning. Journal of Experimental Psychology: General, 109(2), 119.

Sedek, G., & Von Hecker, U. (2004). Effects of subclinical depression and aging on generative reasoning about linear orders: Same or different processing limitations?. *Journal of Experimental Psychology: General, 133(2)*, 237.

See Also

arith, spatial2d, spatial3d

Examples

```
#Generate an item with default item set
lisy(seed=10,nclues=4,nspread=6,clone = NULL,incidental='names',linear=FALSE,
    antonym="first",ninfer = 3, direct='ob', Ndist=3,
    dist="mixed",distprob=0.5,itemSet='random',
    items= NULL,scales = NULL)

#Item set
superheroes <- c('Spider man','Super man','Batman','Wolverine',
'Catwoman','Thor','The Shadow','Silver Surfer', 'Flash','Wonder woman',
'Mr. Fantastic', 'Aqua man', "Hawkeye", 'Starfire', 'Venom', "General Zod")

#Antonym
compare <- c("taller","shorter", "older", "younger",
 "smaller", "bigger","stronger", "weaker")

#Generate item with own dataset
lisy(seed=1,nclues=4,nspread=6,clone = NULL,incidental='names',linear=FALSE,
    antonym="first",ninfer = 3, direct='ob',
    Ndist=3, dist="mixed",distprob=0.5,
    itemSet='own',items= superheroes, scales = compare)

#loop through 30 items
run <- NULL
for(i in 1:30){
 run[[i]]<-AIG::lisy(seed=i,nclues=4,nspread=5,clone = 1,incidental='names',linear=TRUE,
                     dist="false",distprob=0.5,itemSet='random',
                     antonym="both",ninfer = 3, direct='of', Ndist=3,
                    items= NULL,scales = NULL)
}
```

---

spatial2d *Spatial 2D Reasoning Items*

Description

This function generates a 2 or 3 dimensional display figure with 4 2D distractors and 1 2D answer.

Usage

```
spatial2d(items, wd = NULL, view = "2D", seed = NULL, degree = 360,
  ansDegree = 180, test = 1)
```

Arguments

items	The number of items to generate.
wd	This is the working directory to save the figures in. If not provided, the file will be saved in the current working directory.
view	There are three options ("2D", "top", "bottom") to view the display figure from different prespectives.
seed	To generate the same set of item(s) on the local computer.
degree	This allows users to change the rotation of the display figure by increasing or decreasing the value.
ansDegree	This will allow users to change the degree of rotation for the correct figure.
test	This will generate the specific value in the name of the saved images.

Details

To see the 3D display figure, change the view argument to either 'top' or 'bottom'. For 3D figures, some of the cubes may be hidden in sight. Hence, the 2D answer may not seem correct. So one would need to rotate the display figure several times to ensure that none of the cubes are hidden. The rotation can be done by changing the degree value in the function. To ensure that the same image is generated again, please provide a seed value. Currently angle of the view from the top and the bottom is fixed. It may be better to generate one item at a time for the 3D displayed figures. This is not a problem for 2D items.

By default the actual answer is always at half of the display polar coordinates. The display figure polar coordinates by default is 360. Increasing the degrees will make the figure rotate clock-wise and decreasing the degrees will make the figure rotate anti-clockwise.

4 distractors are generated. 2 of the generated distractors are a mirror image of the displayed figure. The difference in the polar coordinates between the two distractors are fixed at 120. The third distractor is a figure wihout the first square that is generated by the display item. The fourth distractor is a figure with an additional square added to the distractor. Currently, the polar coordinates are fixed for the third and fourth distractor as well.

There are occassions where a mirror distractor will not work. In such an instance, the item together with the distractors will not be useful for the test. Hence, it will be a good item to remove the item from the test. It is also a good idea to keep track on your local computer which seed it is so you can avoid having to regenerate the item. That is where the seed will come in handy. The seed correspond to the number position of which the item is created. For example, if you want to recreate item/figure number 19, then all you need do so is to keep the argument item = 1, but change the argument seed = 19. This will re-generate the number 19th item. Bear in mind that the created figures will use item 19 as the file name. So if you try to re-generate the 19th item 10 times, then all it does is to rewrite on the same image for 10 times. Hence, you will not get replicates.

Finally, to automatically generate more than 1 item, just make sure that the argument seed = NULL, and put in how many values you want in the argument item. See the examples for help.

Value

ans Return the matrix that generates the display figure and answer.

mirror Return the matrix that generates a mirror figure.

dist3 Return the matrix that generates the distractor without the first square.

dist4 Return the matrix that generates the distractor that adds an extra square.

Author(s)

Aiden Loe and Francis Smart

See Also

lisy, arith, spatial3d

Examples

```
#2D display figure (Generate the 19th item given the seed value)
spatial2d(items=1, wd=NULL,view="2D", seed=19, degree=360, ansDegree= 180, test = 2)

#2D display figure (Generate 10 items)
#seed must be NULL
#spatial2d(items=10, wd=NULL,view="2D", seed=NULL, degree=360, ansDegree= 180, test = 2)

#wd<- "~/desktop"
#3D display figure (top view)
#spatial2d(items=1, wd=wd,view="top", seed=1, degree=320, ansDegree=160, test = 2)

#3D display figure (bottom view)
#spatial2d(items=1, wd=wd,view="bottom", seed=1, degree=320, ansDegree=100, test = 2)
```

---

spatial3d *Spatial 3D Reasoning Item*

Description

This function generates a 3 dimensional display figure. It acts as a wrap because the creation of the figure is done using functions from the rgl package.

Usage

```
spatial3d(seed = 1, angle = pi/1.3, x = 0.3, y = 3, z = 0.8,
  cubes = 8, axis = TRUE, filled = "yes")
```

Arguments

seed	To generate the same set of item(s) on the local computer.
angle, x, y, z	See details
cubes	The number of connected cubes to generate.
axis	Showing the axis is helpful when first testing the function.
filled	yes, no, random will determine the colour of the cubes.

Details

For 3D figures, some of the cubes may be hidden in sight when automatically generated. Hence, one would need to rotate the display figure several times to ensure that none of the cubes are hidden. To ensure that the same image is generated again, please provide a seed value. You will need to use the rgl post script function to save the figure. Currently, the function returns the a list object that is used to generate the display figure. The figure matrix in the list object can be used for the spatial3d_mirror to generate a mirror image of the display figure.

The arguments angle, x, y, z represents the rotation of angle radians based on the x, y and z axis. This is a wrapper to the rotationMatrix function from the rgl package. Changing the values in the arguments angle, x, y, z coordinates allows one to programmatically change angles to study potential cognitive operators at work.

You can also rotate the figure interactively by clicking on the figure and moving it in different direction.

Value

figure Return the matrix that generates the display figure.

rotationMatrix Return the cordinates of x,y,z,w

Author(s)

Aiden Loe and Francis Smart

References

https://en.wikipedia.org/wiki/Radian

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement*, 14(3), 237-245.

See Also

lisy, arith, spatial2d, spatial3d_mirror

Examples

```
item <- spatial3d(seed=4, angle=pi/1.3, x=0.3,y=4,z=0.8,cubes=8,axis = TRUE)

# To save the figure (not run)
# library(rgl)
# item <- spatial3d(seed=4, angle=pi/1.3, x=0.3,y=4,z=0.8,cubes=8,axis = TRUE)

# save in pdf
# wd<- '~/desktop'
# item <- 1

# save <- paste0(wd,"/display3d_",item,".pdf")
# rgl.postscript(save,"pdf")

# save in png
```

```
# rgl.snapshot(filename="image3D.png",fmt="png")
```

`spatial3dDouble` *Spatial 3D Reasoning Item (double shapes)*

Description

This function generates a 3 dimensional display figure. It acts as a wrap because the creation of the figure is done using functions from the rgl package.

Usage

```
spatial3dDouble(seed = 1, angle = pi/1.3, x = 0.3, y = 3, z = 0.8,
  cubes = 8, axis = TRUE, filled = "yes")
```

Arguments

`seed`	To generate the same set of item(s) on the local computer.
`angle, x, y, z`	See details
`cubes`	The number of connected cubes together.
`axis`	Showing the axis is helpful when first testing the function.
`filled`	yes, no, random will determine the colour of the cubes. Ignore for now.

Details

For 3D figures, some of the cubes may be hidden in sight when automatically generated. Hence, one would need to rotate the display figure several times to ensure that none of the cubes are hidden. To ensure that the same image is generated again, please provide a seed value. You will need to use the rgl post script function to save the figure. Currently, the function returns the a list object that is used to generate the display figure. The figure matrix in the list object can be used for the `spatial3d_mirror` to generate a mirror image of the display figure.

The arguments angle, x, y, z represents the rotation of angle radians based on the x, y and z axis. This is a wrapper to the rotationMatrix function from the rgl package. Changing the values in the arguments angle, x, y, z coordinates allows one to programmatically change angles to study potential cognitive operators at work.

You can also rotate the figure interactively by clicking on the figure and moving it in different direction.

Value

figure Return the matrix that generates the display figure.

rotationMatrix Return the cordinates of x,y,z,w

Author(s)

Aiden Loe and Francis Smart

References

https://en.wikipedia.org/wiki/Radian

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement*, 14(3), 237-245.

See Also

lisy, arith, spatial2d, spatial3d_mirrorDouble

Examples

```
item <- spatial3dDouble(seed=4, angle=pi/1.3, x=0.3, y=4, z=0.8, cubes=9, axis = TRUE)

# To save the figure (not run)
# library(rgl)
# item <- spatial3dDouble(seed=4, angle=pi/1.3, x=0.3, y=4, z=0.8, cubes=9, axis = TRUE)

# save in pdf
# wd<- '~/desktop'
# item <- 1

# save <- paste0(wd,"/display3d_",item,".pdf")
# rgl.postscript(save,"pdf")

# save in png
# rgl.snapshot(filename="image3D.png",fmt="png")
```

---

spatial3d_mirror *Mirror Spatial 3D Reasoning Item*

Description

This function generates the mirror image of the 3 dimensional display figure. It acts as a wrap because the creation of the figure is done using functions from the rgl package.

Usage

```
spatial3d_mirror(obj, angle = pi/1.3, x = 0.3, y = 3, z = 0.8,
  cubes = 1, axis = TRUE)
```

Arguments

obj	An object with class of threeD.
angle, x, y, z	See details.
cubes	The number of connected cubes to generate.
axis	Showing the axis is helpful when first testing the function.

Details

For 3D figures, some of the cubes may be hidden in sight when automatically generated. Hence, one would need to rotate the display figure several times to ensure that none of the cubes are hidden.

The arguments angle, x, y, z represents the rotation of angle radians based on the x, y and z axis. This is a wrapper to the rotationMatrix function from the rgl package. Changing the values in the arguments angle, x, y, z coordinates allows one to programmatically change angles to study potential cognitive operators at work.

You can also rotate the figure interactively by clicking on the figure and moving it in different direction.

Value

figure Return the matrix that generates the mirror image of the display figure.

Author(s)

Aiden Loe

References

https://en.wikipedia.org/wiki/Radian

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement*, 14(3), 237-245.

See Also

lisy, arith, spatial2d, spatial3d

Examples

```
display <- spatial3d(seed=4, angle=pi/1.3, x=0.3,y=4,z=0.8, cubes=9, axis = TRUE)
display_mirror <- spatial3d_mirror(display, angle=pi/1.3, x=0.3,y=4,z=0.8, cubes=9, axis = TRUE)

# To save the figure (not run)
# library(rgl)
# display_mirror <- spatial3d_mirror(display, angle=pi/1.3, x=0.3,y=4,z=0.8, cubes=9, axis = TRUE)
# wd<- '~/desktop'

# save in pdf
# item <- 1
# rgl.postscript(save,"pdf")

# save in png
# rgl.snapshot(filename="image3D.png",fmt="png")
```

`spatial3d_mirrorDouble` *Mirror Spatial 3D Reasoning Item (2)*

Description

This function generates the variation of the images of the 3 dimensional display figure. There are four methods in total. See details for more information. It acts as a wrap because the creation of the figure is done using functions from the rgl package.

Usage

```
spatial3d_mirrorDouble(obj, angle = pi/1.3, x = 0.3, y = 3, z = 0.8,
  method = "one", axis = TRUE)
```

Arguments

`obj`	An object with class of threeD.
`angle, x, y, z`	See details
`method`	There are 4 methods. See details.
`axis`	Showing the axis is helpful when first testing the function.

Details

For 3D figures, some of the cubes may be hidden in sight when automatically generated. Hence, one would need to rotate the display figure several times to ensure that none of the cubes are hidden. This may be even more so when it comes to 2 figures within an image. The longer the number of connected cubes, the more likely it will overlap with each other. About 8 is more or less right.

The arguments angle, x, y, z represents the rotation of angle radians based on the x, y and z axis. This is a wrapper to the rotationMatrix function from the rgl package. Changing the values in the arguments angle, x, y, z coordinates allows one to programmatically change angles to study potential cognitive operators at work.

You can also rotate the figure interactively by clicking on the figure and moving it in different direction.

There are 4 methods to create distractors. The first method generates, the first image as a mirror, and the second image as normal. The second method is vice versa from the first method. The third method generates both mirror images. The first method generates correct rotation but the image is always short of one cube.

Value

figure Return the matrix that generates the mirror image of the display figure.

Author(s)

Aiden Loe

References

https://en.wikipedia.org/wiki/Radian

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement*, 14(3), 237-245.

See Also

lisy, arith, spatial2d, spatial3dDouble

Examples

```
display <- spatial3dDouble(seed=4, angle=pi/1.3, x=0.3,y=4,z=0.8,cubes=8, axis = TRUE)
display_mirror <- spatial3d_mirrorDouble(display, angle=pi/1.3, x=0.3, y=4, z=0.8,
                                         method="one", axis = TRUE)

# To save the figure (not run)
# library(rgl)
# display_mirror <- spatial3d_mirrorDouble(display, angle=pi/1.3, x=0.3, y=4, z=0.8,
#                                            method="one", axis = TRUE)
# wd<- '~/desktop'

# save in pdf
# item <- 1
# save <- paste0(wd,"/mirror3d_",item,".pdf")
# rgl.postscript(save,"pdf")

# save in png
# rgl.snapshot(filename="image3D.png",fmt="png")
```

Index