DONDERS INSTITUTE

RavGen Color

Nick
Ann
Maud

Ann Hogenhuis

some code within the notes

Maud Megen

Background

Raven's matrices are a measure of abstract reasoning

Abstract reasoning ability is suuuuper relevant

1. mental health symptoms (e.g., autism & ADHD)
2. clinical outcomes (e.g., stroke, dementia & traumatic brain injury)
3. life outcomes (e.g., aging, mortality, educational attainment, income)

The 'gold standard' Raven's Matrices is copyrighted (costs 466.04), not digitized and crucially has very little stimuli.

Notable Open Source alternatives

International Cognitive Ability Resource (ICAR)

Hagen Matrices Test

The Matrix Reasoning Item Bank (MaRs-IB)

Notable Open Source alternatives

WHY BOTHER!?!?!

International Cognitive Ability Resource (ICAR)

n items = 11

Hagen Matrices Test

n items = 18

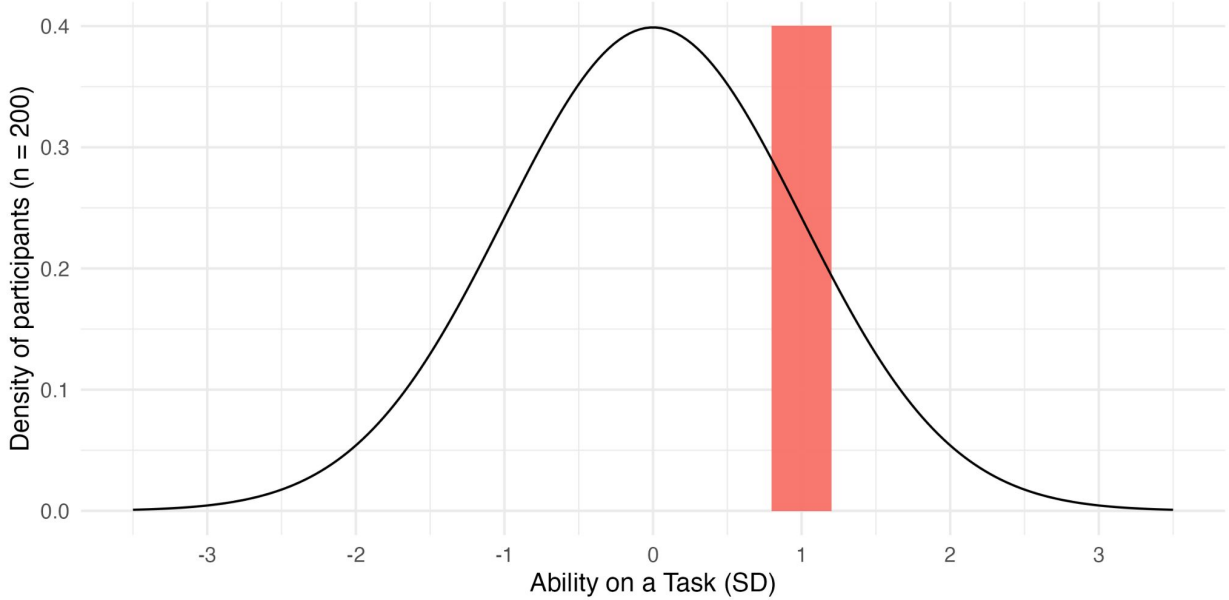
The Matrix Reasoning Item Bank (MaRs-IB)

n items = 80

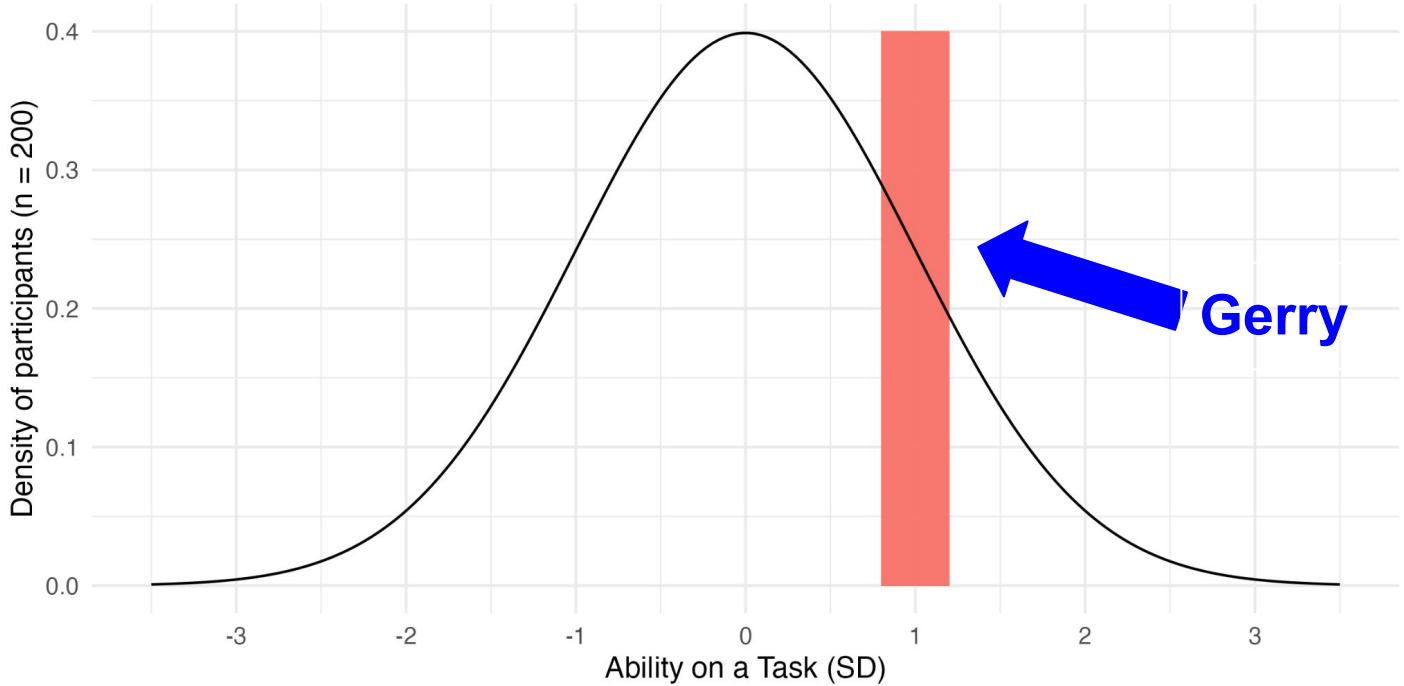
Issues with a low number of stimuli

Subjects remembering: Repeat testing, longitudinal designs & EMA

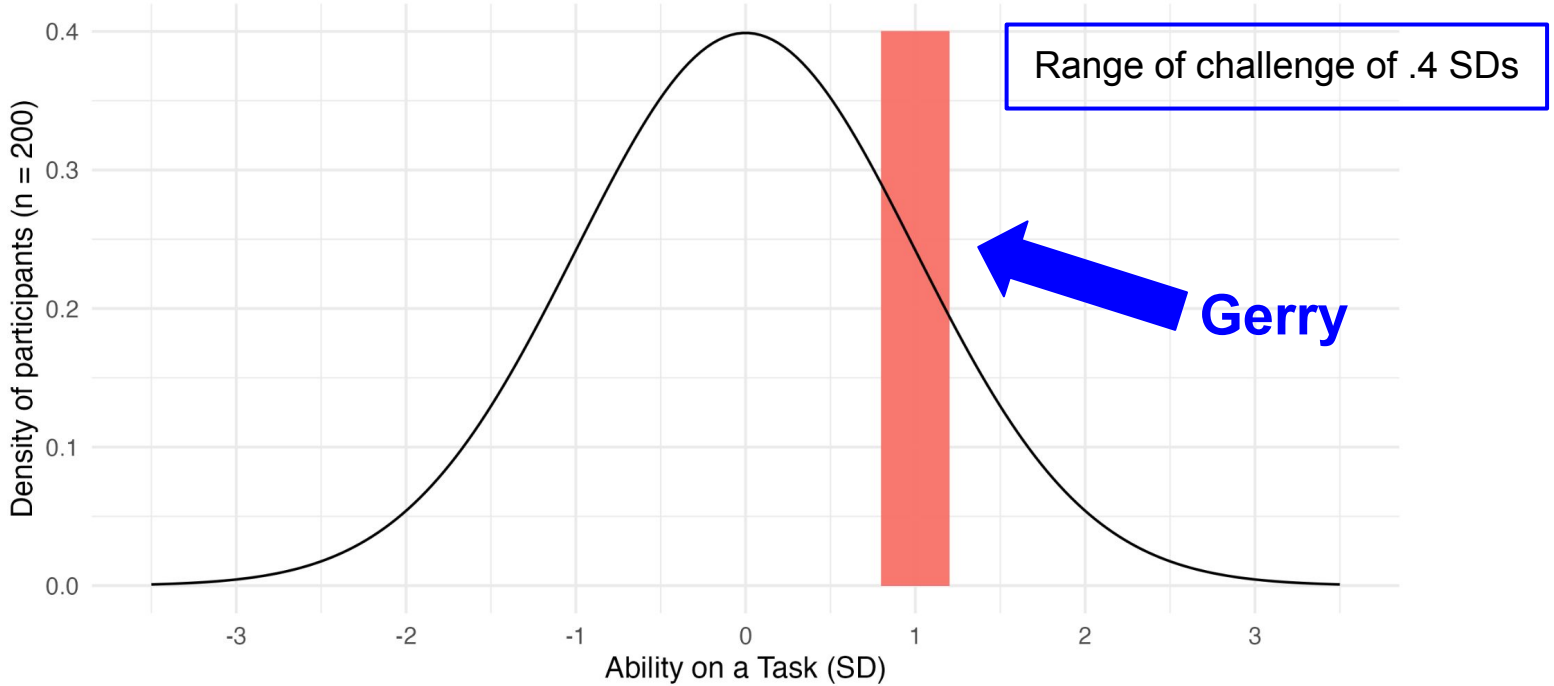
Cognitive training studies



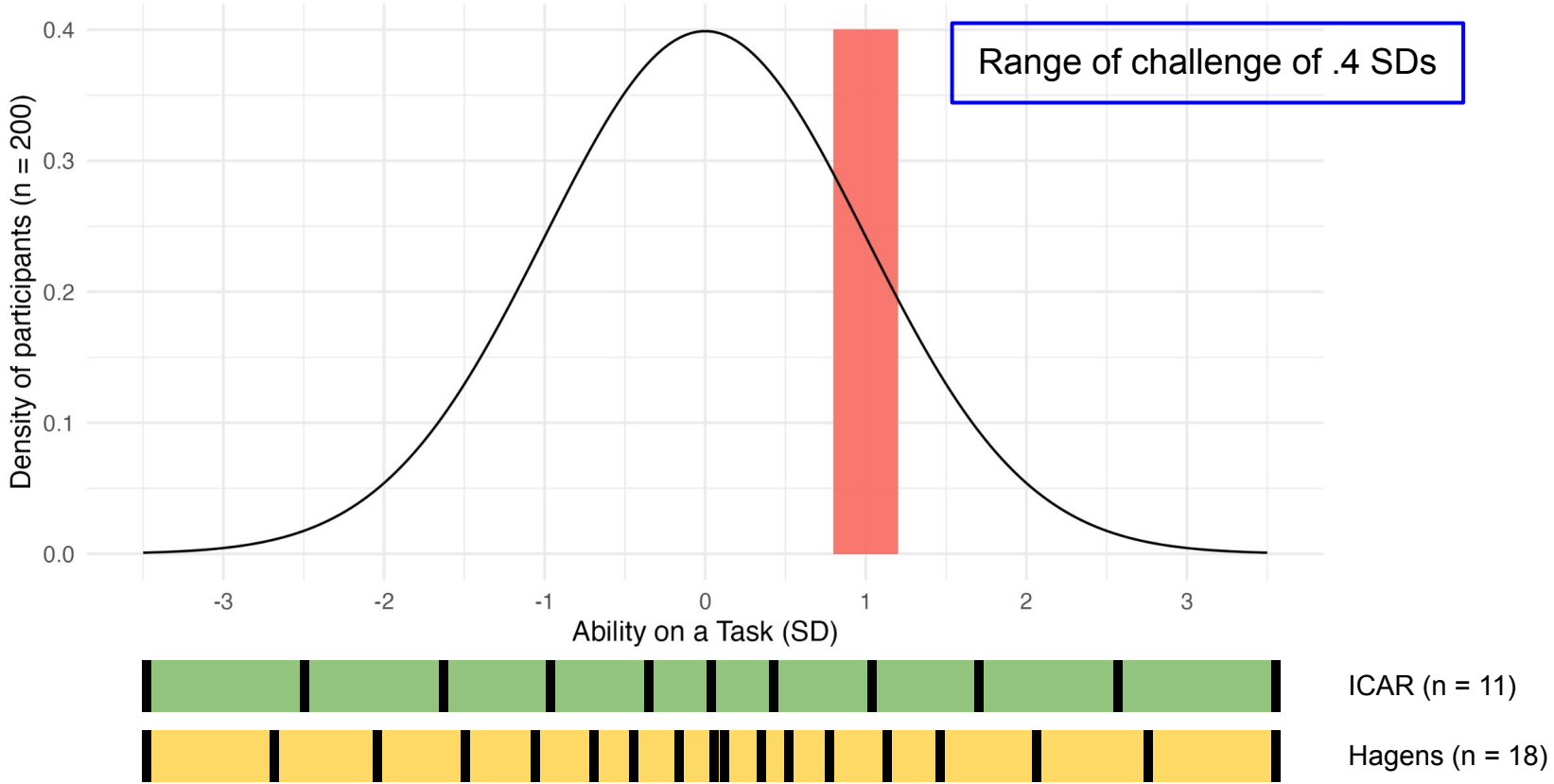
Issues with a low number of stimuli



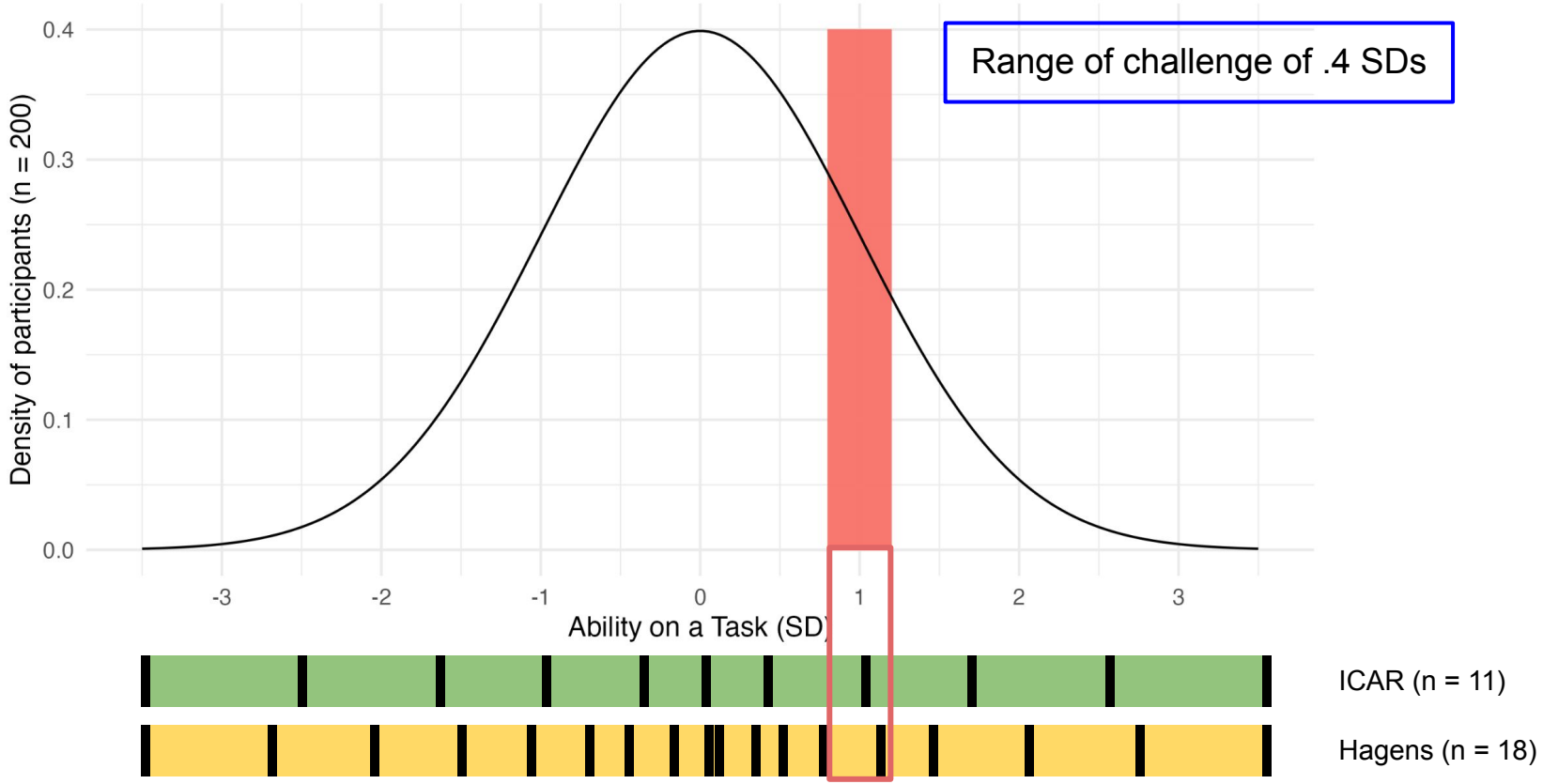
Issues with a low number of stimuli



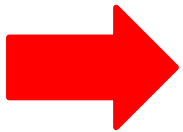
Issues with a low number of stimuli



Issues with a low number of stimuli



Matrix generation software

stimuli  **generation**

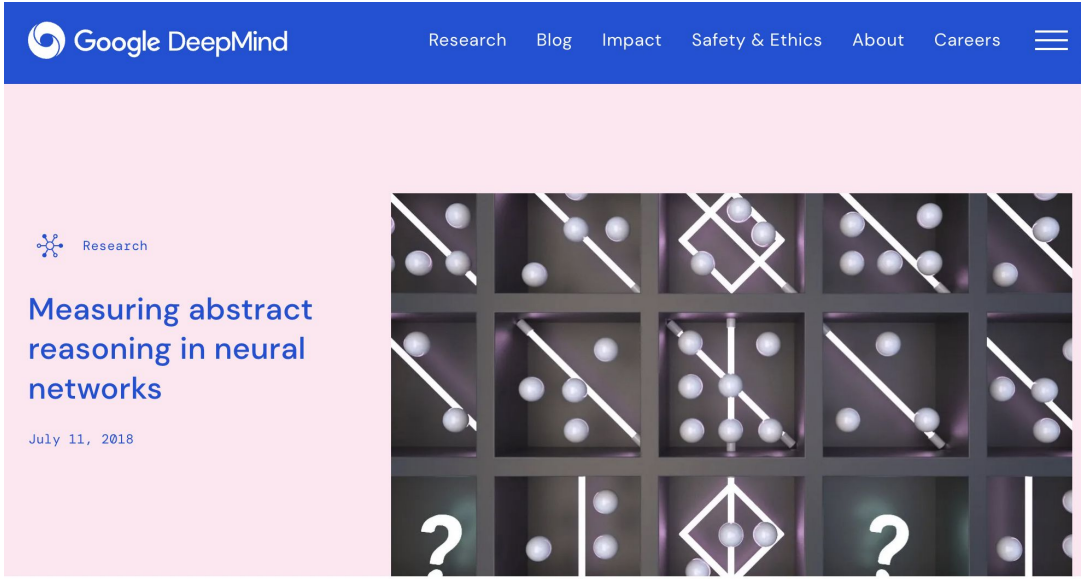
Matrix generation software

Sandia National Laboratories

Matrix generation software

Sandia National Laboratories

Deepmind (1.4m items)

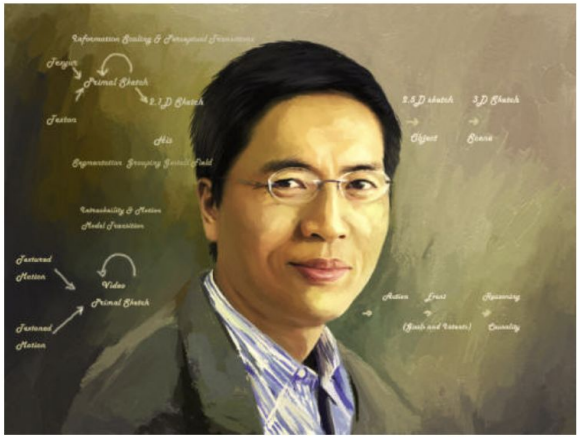


Matrix generation software

Sandia National Laboratories

Deepmind (1.4m items)

UCLA machine learning researchers



hi-res image*

Song-Chun Zhu
Professor, jointly
Statistics and Computer Science

This is Chi Zhang's personal website!

I'm a Ph.D. candidate in Computer Science (Artificial Intelligence) at UCLA advised by Professor Judea Pearl. My research interest is causal inference and machine learning. I'm actively (2023) looking for research scientist and post-doctoral researcher positions.

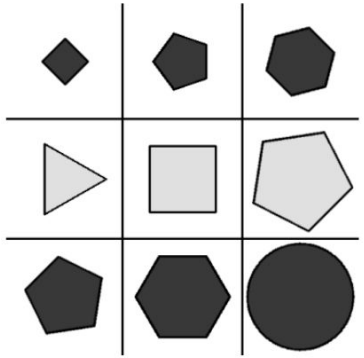
https://web.cs.ucla.edu/~zccc/

RAVEN generator

Generates matrices

Has 3 types of parameters

1. Type (layout)
2. Attributes (color, size etc)
3. Rules

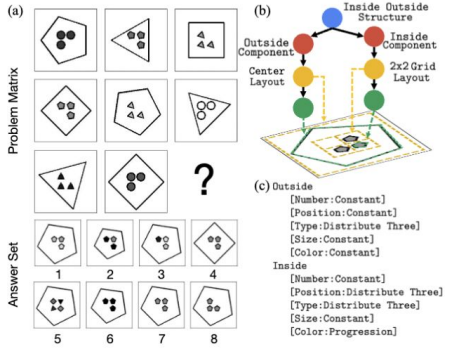


RAVEN: A Dataset for Relational and Analogical Visual rEasoNing

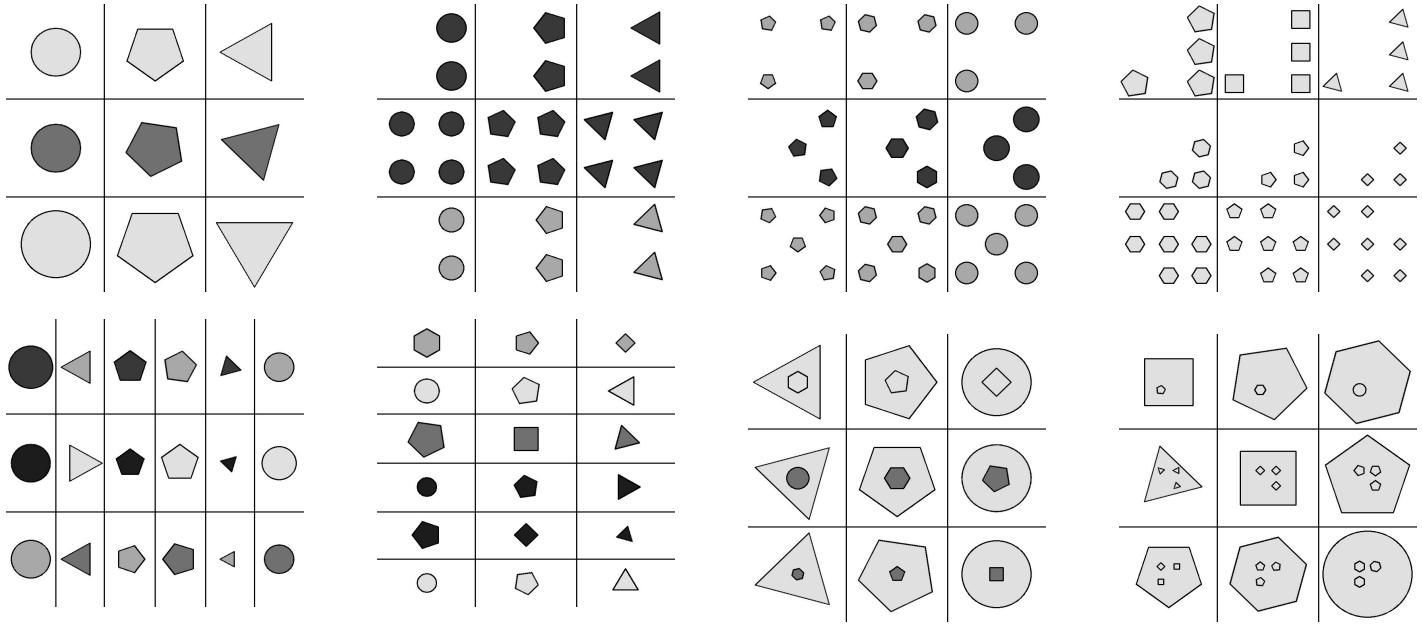
Chi Zhang*,1,2 Feng Gao*,1,2 Baoxiong Jia[1] Yixin Zhu[1,2] Song-Chun Zhu[1,2]
[1] UCLA Center for Vision, Cognition, Learning and Autonomy
[2] International Center for AI and Robot Autonomy (CARA)
{chi.zhang,f.gao,baoxiongjia,yixin.zhu}@ucla.edu, sczhu@stat.ucla.edu

Abstract

Dramatic progress has been witnessed in basic vision tasks involving low-level perception, such as object recognition, detection, and tracking. Unfortunately, there is still an enormous performance gap between artificial vision systems and human intelligence in terms of higher-level vision problems, especially ones involving reasoning. Earlier attempts in equipping machines with high-level reasoning have hovered around Visual Question Answering (VQA), one typical task associating vision and language understanding. In this work, we propose a new dataset, built in the context of Raven's Progressive Matrices (RPM) and aimed at lifting machine intelligence by associating vision with structural, relational, and analogical reasoning in a hierar-



Parameter: Type



```
from raven gen import Matrix, MatrixType, Ruleset, RuleType
list(MatrixType)
[<MatrixType.ONE_SHAPE: 1>, <MatrixType.FOUR_SHAPE: 2>, <MatrixType.FIVE_SHAPE: 3>, <MatrixType.NINE_SHAPE: 4>,
<MatrixType.TWO_SHAPE_VERTICAL_SEP: 5>, <MatrixType.TWO_SHAPE_HORIZONTAL_SEP: 6>, <MatrixType.SHAPE_IN_SHAPE: 7>,
<MatrixType.FOUR_SHAPE_IN_SHAPE: 8>]
```

Parameter: Attributes

Rule: **Constant**

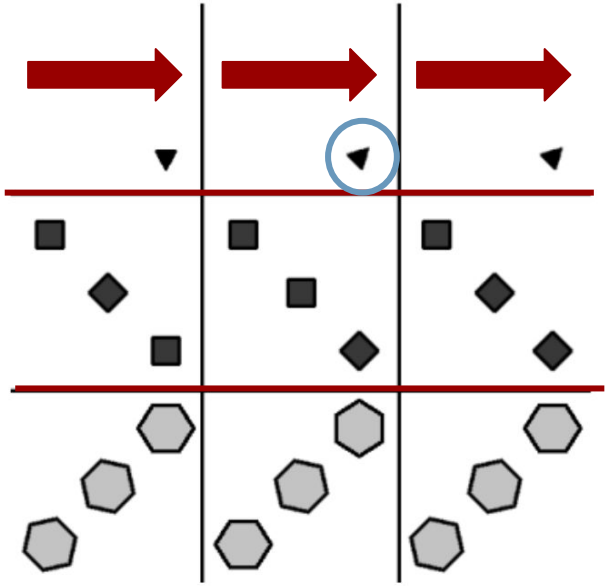
Shape is constant

Color is constant

Size is constant

Number is constant

Position is random noise



Parameter: Rules (applied rowwise)

Carpenter, Just & Shell 1990

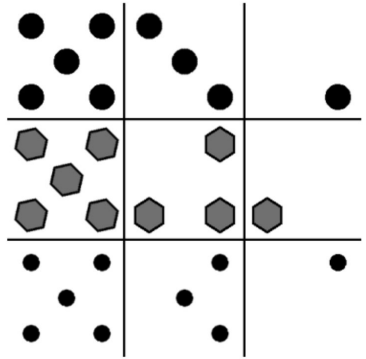
`CONSTANT:` same value in a row

`PROGRESSION:` quantitative increment or decrement occurs between adjacent entries

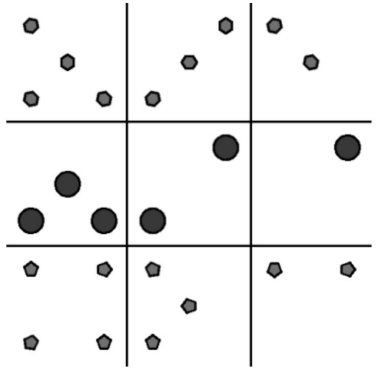
Position is noise!

We edit the attribute **number**.

Increments of 2



Increments of 1



Parameter: Rules (applied rowwise)

Carpenter, Just & Shell 1990

`CONSTANT:` same value in a row

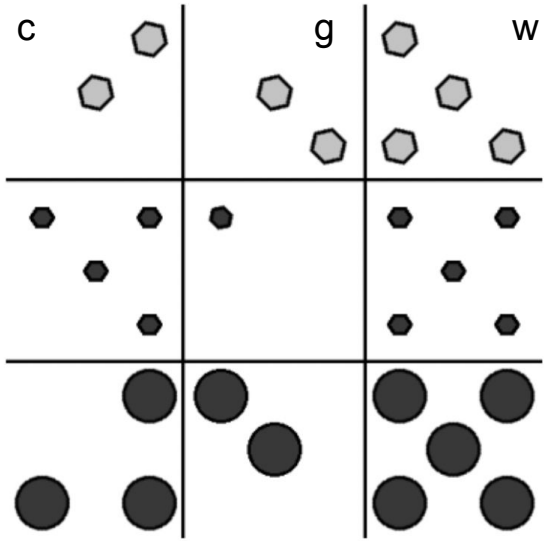
`PROGRESSION:` quantitative increment or decrement occurs between adjacent entries

`ARITHMETIC:` a figure from one column is added to (juxtaposed or superimposed) or subtracted from another figure to produce the third.

Position is noise!

We edit the attribute **number**.

c + g = w (numbers)



Parameter: Rules (applied rowwise)

Position is noise!
We edit the attribute **number**.

Carpenter, Just & Shell 1990

`CONSTANT:` same value in a row

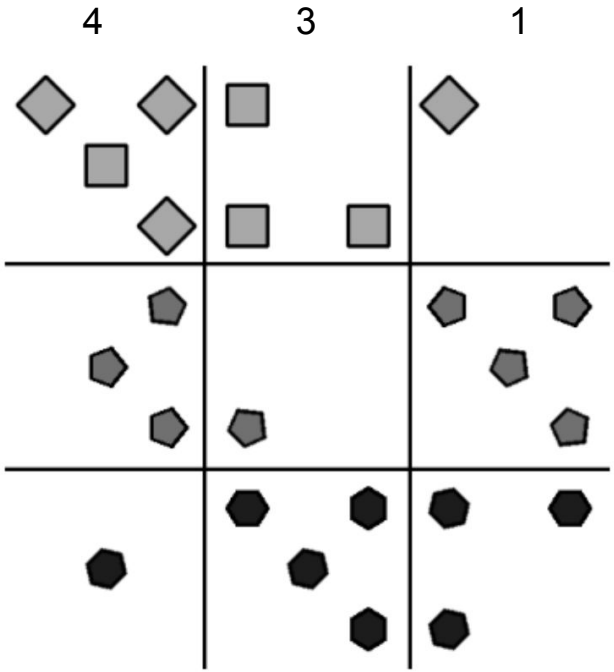
`PROGRESSION:` quantitative increment or decrement occurs between adjacent entries

`ARITHMETIC:` a figure from one column is added to (juxtaposed or superimposed) or subtracted from another figure to produce the third.

`DISTRIBUTE_THREE:` three values from a categorical

attribute (such as figure type) are distributed through a row

Dist 3 is also Dist 2?!?!?



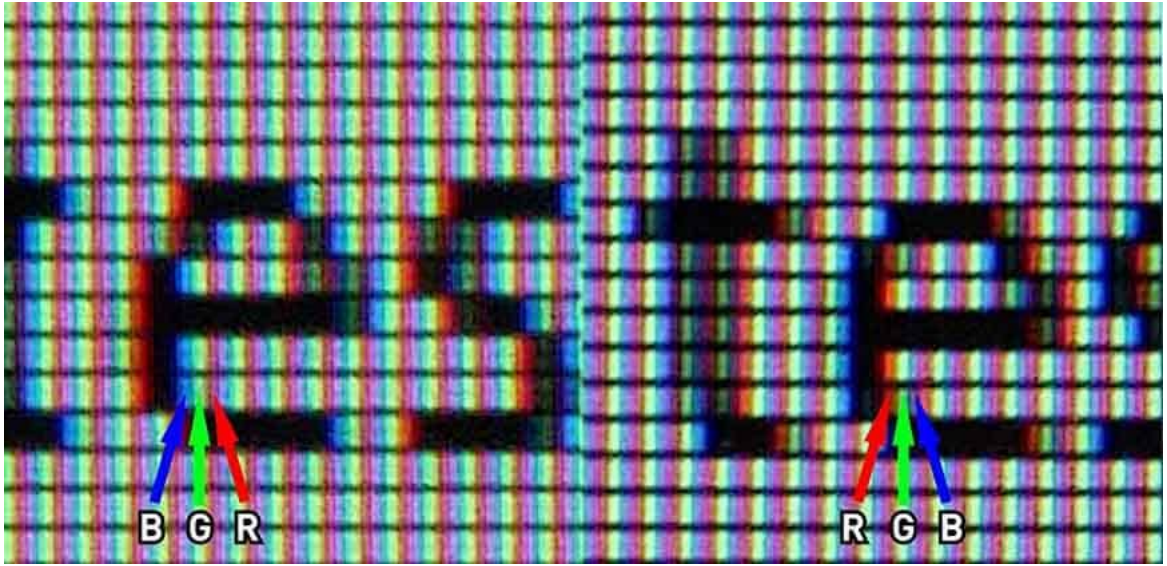
That is what we have. Now where we are going…

Goals & challenges faced

Color representation

Common system: RGB

Current system: BGR



Goals & challenges faced

Raven gen software

- Built on the premise of single int's (greyscale = 1 , color = 3)
- So change entire grid from (x) to (x,x,x)

Goal

```
rpm.save(".", "color")
```

Goals & challenges faced

Raven gen software

- **First** we tried to change this within the package by following the variable logic.
- Nested code

→ New approach

Start:

1. Attribute.py

Initial value options for color and makes it a unique value (Uniquesampleable, sampleable) according to the numbers in line 33.

Line 33-35

```
COLOR_VALUES = (255, 194, 184, 140, 121, 87, 32, 5, 0)
COLOR_MIN = 0
COLOR_MAX = len(COLOR_VALUES) - 1
```

Line 42 & 46

```
class AttributeType(Enum):
    COLOR = auto()
```

Line 59-61

```
@property
def value(self):
    return self.values[self.setting]
```

This will later be used in entity for self.color.value

Line 255-260

```
class Color(UniqueSampleable):

    def __init__(self, constraints):
        super(Color, self).__init__(name=AttributeType.COLOR,
                                    values=COLOR_VALUES,
                                    constraints=constraints)
```

2. Entity.py

Goals & challenges faced

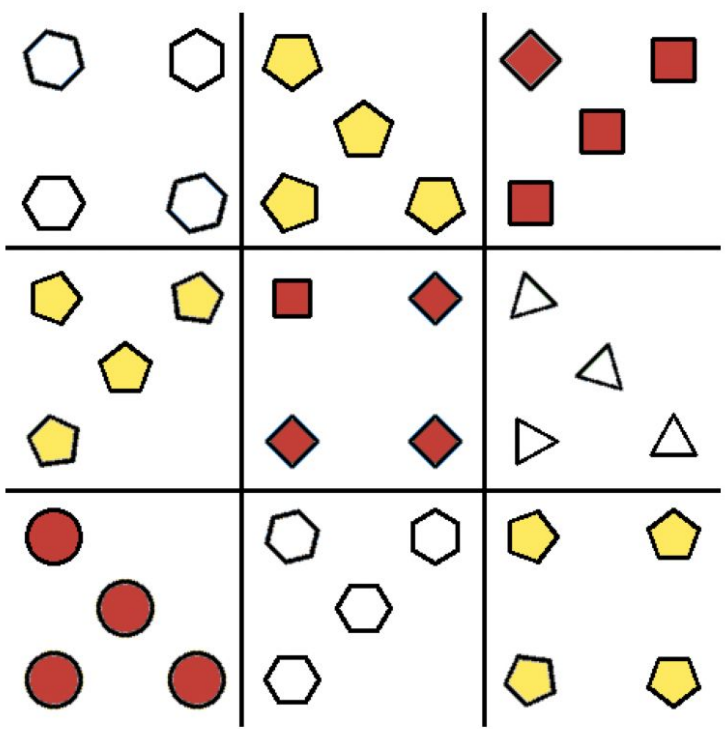
Raven gen software

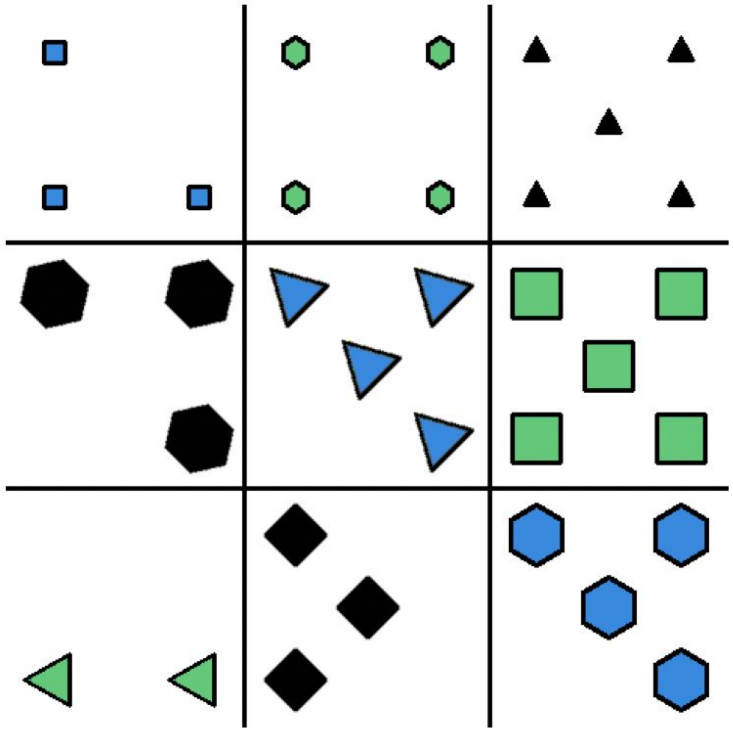
→ New approach

1. Extract matrix of the final image
2. Multiply grey single values to tuples of 3
3. Change to color according to the set grey values

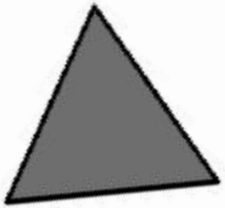
(to keep the rules)

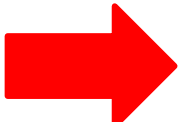
Example stimuli

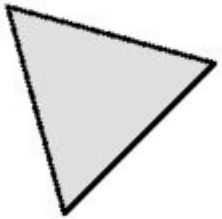


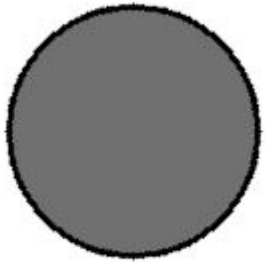


NWO Open Science Fund



RavGen Color  **RavGen Human**



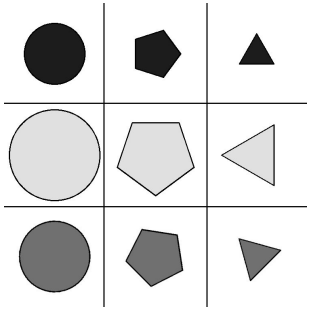


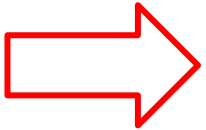
WP1 Designing matrix generation stimuli

Maud Megen

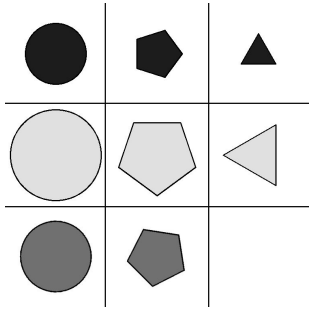
Researcher friendly output

From this





To this

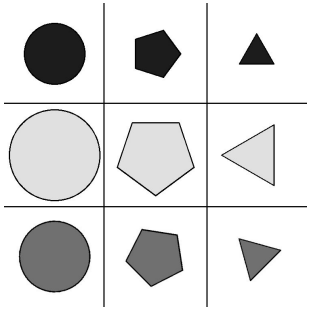


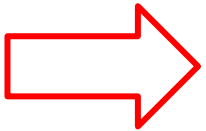
WP1 Designing matrix generation stimuli

Maud Megen

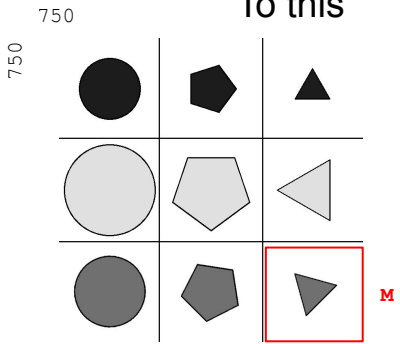
Researcher friendly output

From this





To this




```
M is the dimension of subseted space
M = (image_size/3) -5)*2 # times 2 because it is a 2D image
whole_img[whole_img == [corner_rightbottom:, corner_rightbottom:]] = [0, 0, 0] * M
```

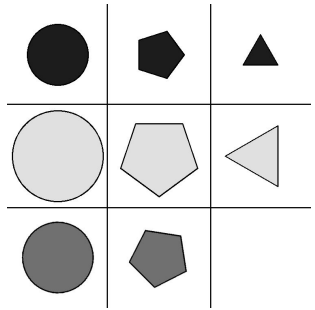
WP1 Designing matrix generation stimuli

Maud Megen

Researcher friendly output

10 parametrizations of 10 colored stimuli = 100 (colored) stimuli

500 png files



WP1 Designing matrix generation stimuli

- Human friendly attributes (e.g., size)
- Human possible combinations
- Making a bank of 1000 sampled combinations

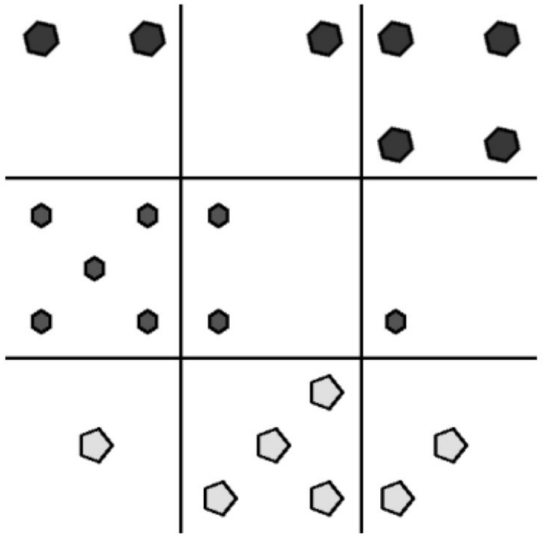
WP2 Testing, validating and disseminating matrix generation software & stimuli

- IRT modeling stimuli difficulty
  - Each stimulus in the bank will have a level
  - Yet, we will also have a range of difficulties for generator parameters

Thanks for your attention!

Some issues… how is this distribute 3…

Need to look into further



```
DistributeThree(name=<RuleType.DISTRIBUTE_THREE: 4>, attr=<AttributeType.NUMBER: 1>, params=None, value=None)
Constant(name=<RuleType.CONSTANT: 1>, attr=<AttributeType.SHAPE: 2>, params=None, value=None)
Constant(name=<RuleType.CONSTANT: 1>, attr=<AttributeType.SIZE: 3>, params=None, value=None)
Constant(name=<RuleType.CONSTANT: 1>, attr=<AttributeType.COLOR: 4>, params=None, value=None)
```

Extra slides/notes: Future to do

Color in save