

How to use probability density functions to track performance self-assessment

Applied Cognitive Science

Yannik P. Frisch, Maximilian A. Gehrke

March 10, 2020



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Centre for
Cognitive
Science

Abstract

Several studies suggest that humans are rather poor when judging their performance after they have fulfilled a task. Probability density functions is a good way to measure the own evaluation of performance. To our knowledge, this has not yet been implemented. Which is why we wanted to test if and how one can use probability density functions in an experiment to extract the self-evaluation of task performance.

We designed a questionnaire where subjects have to draw probability density functions after executing simple sorting tasks. We evaluated each task using a norm scoring function and assigned each task 5 points. Last, we calculated the Brier score, a score for measuring uncertainty, and came to the conclusion that humans are about average in evaluating their performance.

The main aspect of this report is the design of a questionnaire which uses probability density functions to assess performance self-evaluation.

1 Introduction

The question of how to predict peoples performance on a task has been of great interest in the psychology literature. To be able to make qualified statements about this ability, one has to define it first. This can be achieved by the framework of *metacognition* including the terms *metacognitive sensitivity*, *metacognitive bias* and *metacognitive efficiency* [x].

The first is used to express how good a subject is at differing between his or her own correct and incorrect answers. A useful initial approach is the 2×2 confidence-accuracy table, labeled "type 2 SDT table" by [x] that is the equivalent of the usual "type-1" SDT table [x] applied to the *metacognition* framework. Common measures of the association between the rows and the columns of the table in the type-1 case are the ϕ -correlation [x] and the Goodman-Kruskall gamma coefficient G [x]. It is well known that both measurements are affected by (metacognitive?) bias [x], and [x] showed that this also holds for the type-2 application of the measurements.

A standard way to remove the influence of the bias would be using d' [x] which will be constant given different biases and also has several approaches to metacognitive sensitivity [x]. But type-2 d' is also affected by changes in the metacognitive bias (HOW?) [x].

One way to remove this issue is the use of non-parametric analysis, that does not make the equal-variance gaussian assumptions, e.g. ROC analysis [x], that can be applied to type-2 data.

A further complication for using the above methods to measure metacognitive sensitivity is the fact that all these measures are affected by the task performance (HOW?) [x].

- One solution approach: Use psychophysical techniques to control for differences in performance
- Alternative approach: Explicitly model the connection between performance and metacognition
- Meta- d' measure exploits the fact that given gaussian variance assumptions at type 1 level, the shapes of the type 2 distribution are known even if they are not themselves gaussian.
- Therefore max type 2 performance is constrained by one's type 1 performance
- Given a particular type 1 variance structure and bias, the form of the type 2 ROC is completely determined.
- Given the subject's actual type 2 performance, we can obtain the underlying type 1 sensitivity, labeled meta- d' .
- meta- d' is robust to changes in bias and recovers simulated changes in metacognitive sensitivity
- For a metacognitively ideal observer, meta- d' should be equal to d'
- We can define "metacognitive efficiency" as meta- d' / d'
- Efficiency of 1 encodes an ideal value
- Closely related measure is meta- $d' - d'$ or $\log(\text{meta-}d' / d')$
- Unable to discriminate between different causes of a change in metacognitive efficiency (Trial-to-trial variability in the placement of confidence criteria results in decreasing efficiency as well as additional noise in the evidence used to make the confidence rating does).
- Similar bias-free approach to model metacognitive accuracy is the "Stochastic Detection and Retrieval Model" (SDRM)
 - Measures metacognitive accuracy
 - Also able to model different potential causes of metacognitive inaccuracy
- Different approach to the above: Use 'one-shot' discrepancy measures to quantify metacognition: Compare general confidence rating (generally; asked before the trial) with performance on a variety of tasks. BUT: Using a single rating of performance, it is not possible to distinct bias from sensitivity, nor measure efficiency.
- In contrast, collecting trial-by-trial measures of performance and metacognitive judgements allows to get a picture of an individuals bias, sensitivity and efficiency.

-
- Metacognitive confidence can be formalized as a **probability judgement** directed towards one's own actions.
 - Metacognition has a normative interpretation as the accuracy of a probability judgement about one's own performance.
 - Advantage of this framework: Meaningful measure of bias can be elicited.
 - "Probability Score": $PS = (f - c)^2$
 - f : probability rating
 - c : actual occurrence ($c=1/0$ for binary events)
 - "Brier Score": Mean value of the PS averaged across estimates.
 - Brier score is analogous to ϕ .
 - Can be decomposed into: $PS = O + C - R$
 - O : "Outcome index", reflecting the variance of the outcome event c
 - C : "Calibration", the goodness of fit between probability assessments and the corresponding proportion of correct responses. Quantifies the discrepancy between the mean performance level in a category (e.g. 60%) and its associated rating (e.g. 80%).
 - R : "Resolution, the variance of the probability assessments. Measuring the extent to which correct and incorrect answers are assigned to different probability categories (Is subtracted; larger variance is better).

2 Method

Task: describe your ideas and your realization of the task

2.1 Designing the Questionnaire

We started out by designing a questionnaire in \LaTeX and using the corporate design of our university. We separated each task clearly from the one another and wrote the instructions in the headline. The body of each tasks consists of task related information on the left, space for the answer in the middle and an empty coordinate system on the right. It was important to us to keep this structure to increase reliability across the tasks.

We decided to use sorting tasks, because of their high objectivity (see section 4 for a discussion on question types). The first seven tasks were closed-form sorting tasks. Directly after executing each task, the subjects were asked to fill out the coordinate system with a probability density function over their performance. We decided to ask the subjects to sort five terms in a predefined order. We decided to use two easy items, three medium and two hard items. We randomized the location of the items as well as the order of the answer possibilities. An example can be seen in figure ??.

Task 1: Sortieren Sie die Städte nach ihrem Breitengrad (1 = nördlichste Stadt, 10 = südlichste Stadt).

Städte:

- Hamburg
- Kiel
- Hannover
- Bremen
- Göttingen

1.

☐

2.

☐

3.

☐

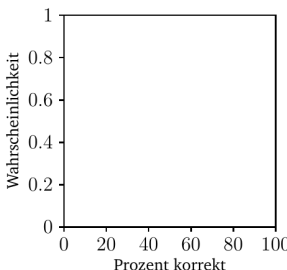
4.

☐

5.

☐

Wahrscheinlichkeit



Prozent korrekt

Figure 1: Example closed-form sorting task.

After the seven tasks, we asked the subjects to estimate their performance over all the previous tasks, by drawing another probability density function. With this we want to learn how well humans can average their performance on several task.

The eighth task was an open end sorting tasks, which we decided to incorporate out of curiosity how the self-assessment would change in comparison to closed form sorting tasks. The task can be seen in figure ??.

Task 8: Nennen und ordnen Sie die fünf europäischen Städte mit den meisten Einwohnern (1 = am meisten, 5 = am wenigsten).

Notizen:

1.

☐

2.

☐

3.

☐

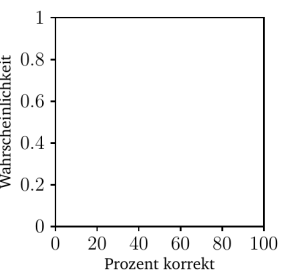
4.

☐

5.

☐

Wahrscheinlichkeit



Prozent korrekt

Figure 2: Example open end sorting task.

2.2 Processing Pipeline

2.3 Extracting Probability Density Functions

To extract the probability density functions from a scanned PDF into an image, we used computer vision to detect the probability density functions on the JPEG and simply cut out the detected regions from the JPEG. The cutout process is fairly easy. In python, images are stored as an array of numbers. As soon as we get the area of the pdf as pixels, we can simply enter these pixel indices in the array and extract that part of the image. Then we save it using OpenCV a free computer vision library in python.

The hard part is to identify the probability density functions in the image. However, we designed our questionnaire in a way that reduces the detection of the pdf to the detection of a big square. If we can reliably detect the coordinate system, which the subjects use to draw their pdfs in, we can extract the pdf if we only look at the pixels which lie inside this square.

To detect squares, we need to detect vertical and horizontal lines first. We did exactly that and looked at all contours that could be build with horizontal and vertical lines. This will output all lines on their own, but also all triangles, rectangles and squares. Everything that forms a contour. Now we sorted the contours. It is important to sort the contours (or later pdfs), regarding their position on the page. To the computer all pdfs look the same, so we have to make sure that we assign the correct pdf to the correct task. Because all tasks are ordered in ascending order, we can sort the pdfs from north to south.

Next, we had to find the correct square from the contours. We did this by iterating over the contours and testing each contour on some

3 Results

4 Discussion

What to do?

- discuss challenges that you faced during implementation,
- reflect your solution
- give an outlook

Our universal goal was to design an experiment which can understand how well people estimate their performance. Therefore is our study a preliminary study. For this we designed an experiment, where subjects had to solve several five item sorting tasks. After finishing each task, the subject had to draw a probability density function over their performance. The questionnaire had two conditions. The first condition was sorting without active recall. The five answers to the question were already given in a randomized order and had to be sorted in the write order. The second condition was sorting with active recall. The question was open and the subject had to know the items and the correct ordering. In total we had 7 questions for condition one and one question for condition two.

After collecting the answers from XX subjects, our program read the probability density functions with the help of computer vision and we analyzed the the Brier score of the data.

Objectivität Finding the correct type of questions.

Finding the correct questions.

Finding the correct metric.

Explaining the concept of probability density functions.

Computer Vision.