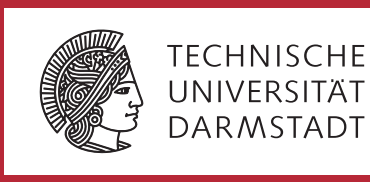
How to use probability density functions to track performance self-assessment

Applied Cognitive Science
Yannik P. Frisch, Maximilian A. Gehrke
March 10, 2020



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 *metacoognitive 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 $2x2$ 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 colums 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]. This can be adressed by excplicitly modeling the connection between a subject's performance and metacognition. The meta-d' measure [x] makes use of the fact that given gaussian variance assumptions (?) at type-1 level, the shapes of the type-2 distributions are known even if they are not themselves gaussian (WHY ?). Therefore the optimal type-2 performance is constrained by one's type-1 performance. E.g. given a particular type-1 variance structure and bias, the form of the type-2 ROC is completely determined. So, given a subject's actual type-2 performance, one could obtain the underlying type-1 sensitivity, labeled meta-d' [x], that is robust to changes in the bias and recovers simulated changes in metacognitive sensitivity. For a metacognitively ideal observer, meta-d' should be equal to d'. To measure this ideality, [] defined *metacognitive efficiency* as meta-d'/d', or by the more stable variants meta-$d' - d'$ or logmeta-d'/d'. However, this measurement is unable to discriminate between different causes of a change in metacognitive efficience. E.g. 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. A similar bias-free approach to model metacogntivie accuracy ist the *Stochastic Detection and Retrieval Model (SDRM)* which we do not want to cover here.
A somewhat different approach uses so-called *one-shot* discrepancy measures to quantify metacognition. A general confidence rating (e.g. asked before the trial) is compared to the actual performance on a variety of tasks, but it should be clear from the above (WHY ?) that using a single rating of performance will not result in a good distinction between the bias and the sensitivity, nor will it enable to measure the efficiency. In contrasst, collecting trial-by-trial measures of performance and metacognitive judgements allows to get a picture of an individuals bias, sensitivity and efficiency.
To get a different view-point on the domain, one could formalize metacognitive confidence as a probability judgement directed towards one's own actions.

1.1 Formalizing metacognition as probability judgements

Metacognition has a normative interpretation as the accuracy of a probability judgement about one's own performance and one advantage is the possibility to elicit a meaningful measure of the bias. A lot of literature is a availabe on how to measure this accuracy, but in the following we want to focus on one of them, called the *Brier Score*. To define this score, [x] first defined the *Probability Score (PS)* as the squared difference between a probability rating f for an event, and it's actual occurence c (0 or 1 for binary events).

$$PS = (f - c)^2$$

The *Brier Score (BS)* can then be defined as the mean value of the PS averaged across all estimates.

$$BS = \frac{1}{N} \sum_i (f_i - c_i)^2$$

This score is the equivalent of the ϕ measurement explained above and can be decomposed, shown by [x], by:

$$BS = O + C - R$$

where O is the *Outcome index*, reflecting the variance of the outcome event c. The *Calibration* expresses the goodness of fit between the probability assessments and the corresponding proportion of correct responses. It quantifies the discrepancy between the mean performance level in categorcy (e.g. 60%) and its associated rating (e.g. 80%). Last, the *Resolution* R encodes the variance of the probability assessments, measuring the extent to which correct and incorrect answers are assigned to different probability categories (EXPLAIN THESE FURTHER).
The next chapter gives an overview on how we measured the brier score of several subjects for different tasks, before we present the results of our measurements and finally come to a conclusion on our atttempt to measure metacognition.

2 Method

Task: describe your ideas and your realization of the task

2.1 Designing the Questionnaire (MAG)

We started out by designing a questionnaire in LaTeXand 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 **??**.

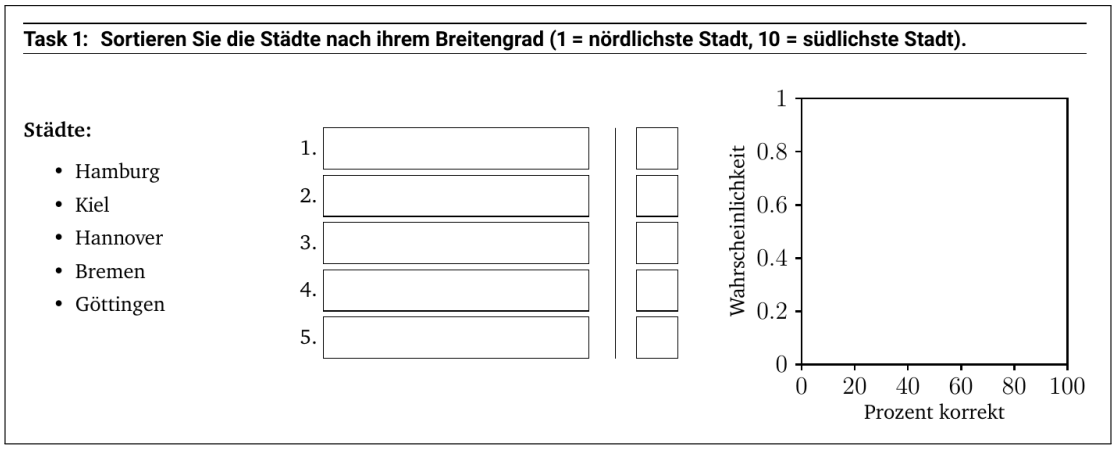


Figure 1: Example closed-form sorting task.

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 **??**.

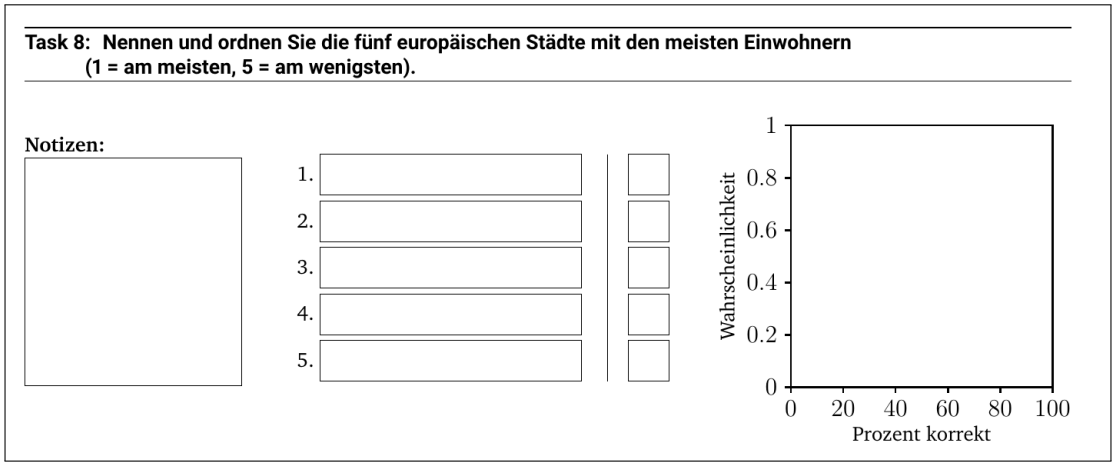
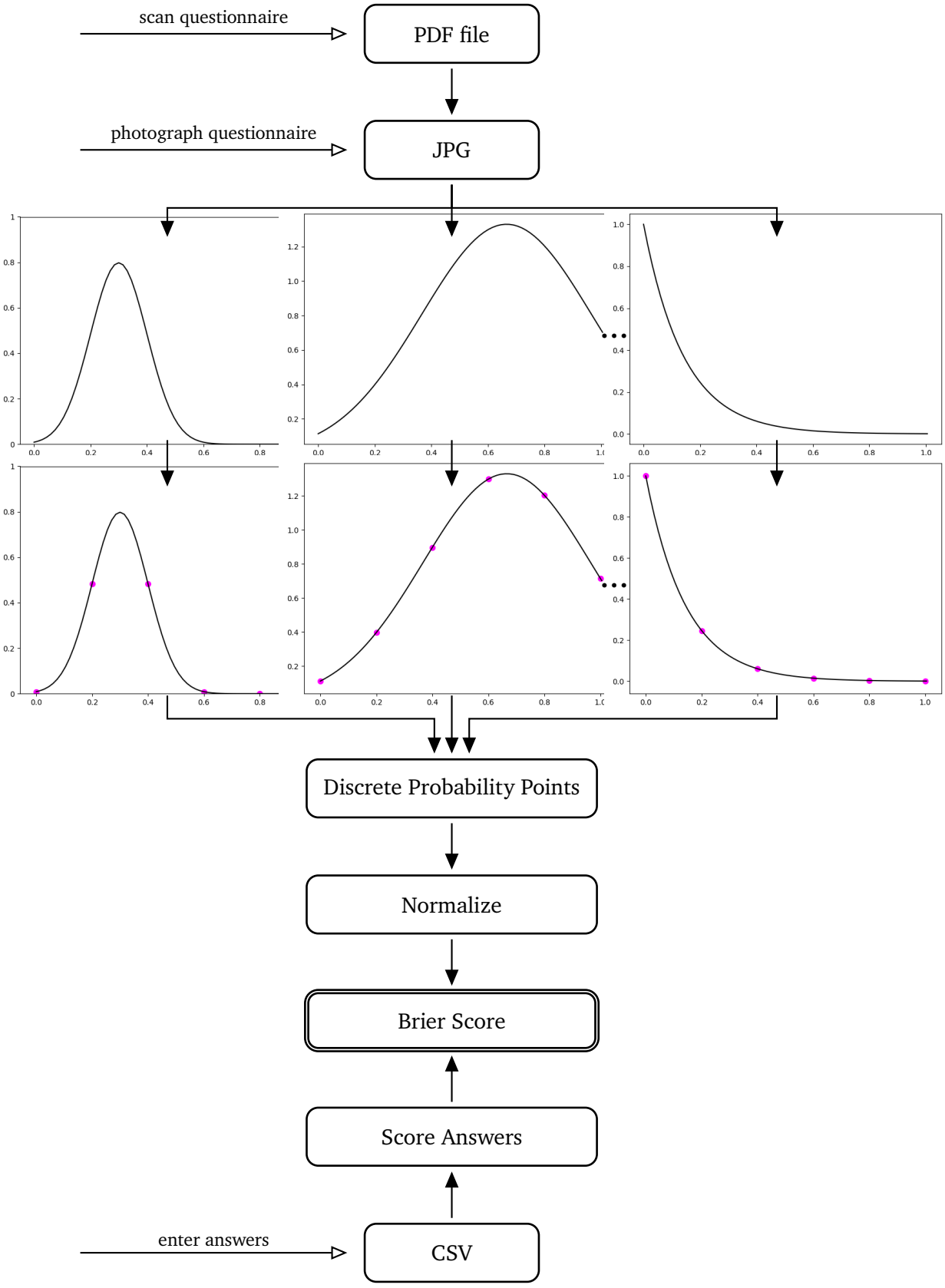


Figure 2: Example open end sorting task.

2.2 Processing Pipeline (MAG)



The pipeline that we used to process the questionnaire can be seen in the figure above. We start out by digitizing the image. This is important so that we are able to apply computer vision to extract the probability density functions. For convenience we provided two ways: either scan the questionnaire as a PDF file or photograph each page of the questionnaire and upload the pages as JPEG's. In the first case, we use the python package "pdf2image" to convert each PDF page into a single JPEG page, which is not necessary in the second case, cause the JPEG's are already available.

After the digitization of the questionnaire, the probability density functions are detected on the questionnaire and extracted as single images. From these images we build a digital representation of each probability density functions and read discrete probabilities at the desired locations. Afterwards we normalize the probabilities in order for them to sum to one. This way, the subjects do not need to care about drawing a normalized probability function and can rather concentrate on which area they assign a higher percentage and which areas they assign a lower percentage.

For a meaningful comparison, we need need to insert the answers of the subjects. We do this by filling a CSV file for each subject by hand. The answers from the CSV file are imported into our program and subsequently scored. An automatic scoring improves the objectivity and avoids errors. Finally, we use the points that each person achieved for each tasks and the normalized probabilities from the probability density functions to calculate the Brier score, which quantifies uncertainties.

The remaining segments of this sections examine the processing pipeline in more detail.

2.3 Extracting Probability Density Functions (MAG)

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 regarding some constraints.

1. The horizontal and vertical length of the contour had to be approximately the same length. We allowed for exactly 8% variation. So one side could be up to 8% longer than the other side. This factor is necessary, because of the difficulty to scan a paper perfectly aligned.

2. The coordinate system has a height of 4.5cm. A Din A4 paper is exactly 29.7cm high. This means that each pdf takes about 0.15% of the height of the image. Allowing for some deviation, the height of the contour had to be between 10 and 20% of the height of the JPEG.

3. The coordinate system has a width of 4.5cm. A Din A4 paper is exactly 21cm wide. This means that each pdf takes about 0.21% of the width of the image. Allowing for some deviation, the width of the contour had to be between 15 and 25% of the width of the JPEG.

4. Last, we excluded all contours which were lying on the left half of the page.

Constraint 1 makes sure that we find squares. Constraint 2 and 3 make sure we find the squares with the correct size. Constraint 4 makes sure that we do not take any squares from the left side of the page. We designed the questionnaire in a way that all pdfs are located at the right half of the page.

Because we calculate the extraction of the pdfs with percentages relative to the size of a Din A4 paper, it is independent of the DPI and resolution of the scanner or photo camera with which the questionnaire is copied.

2.4 Extracting Probabilites from probability density functions (YF)

2.5 Scoring the answers (YF)

2.6 Calculating the Brier Score (YF)

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 12 subjects, our program read the probability density functions with the help of computer vision and we analyzed the the Brier score of the data.

4.1 Finding the correct type of task

In comparison to many other experiments, achieving a high level of objectivity, reliability and validity was quite challenging. This is mainly due to our preconditions:

1. Each task must be answerable in a few minutes
2. The task type allows the design of easy, moderate and difficult tasks
3. The task type does not allow subjects to easily assess the exact number of points they achieve for a task. The idea behind this is that we want to avoid simple counting of correct answers. We are rather interested in the implicit metacognition that subjects exhibit.

Some of our top choices were math, spelling, grammar, translation, estimation, mapping and fill out tasks (see table **??**). We discarded all of these types, because none of them allowed us to define an objective, reliable and valid evaluation procedure that makes it hard for the subjects to track their exact performance.

Task type	Objectivity	Reliability	Validity
Math	-		
Translation	-	-	
Spelling			
Grammar			
Estimation			
Mapping			
Fill out			
Sorting	+	+	+

Table 1: Task types and their criteria for test quality.

Objectivity: Especially a high level of evaluation objectivity was hard to ensure for many tasks. For example math tasks allow for a fine grained assessment, which is what we are looking for, but lack objectivity. It is not possible to write down an evaluation scheme that considers all possibilities. In addition, even if we can write down such an evaluation scheme, it would need loads of resources and is susceptible to errors. Any possible answer that we forget has to be added to the evaluation scheme and incorporated in all previous evaluations. Another problem is the amount of expertise that the experiment administrator needs. Last, we would not be able to ask questions of different areas.

Reliability: Per se reliability was given for all the task types in table **??**. In the end we only care about the performance and the respective self-assessment of the subjects. At this stage, we do not really care why people are good or bad at specific tasks, we are only interested in getting an idea how well people self-assess themselves on average. However, a task type can be problematic if it biases the subject to draw the probability density function differently. We decided to use the sorting task.

Validity:

4.2 Finding good questions

4.3 Biases drawing the probability density functions.

4.4 Finding the correct metric

4.5 Explaining probability density functions

The idea of using probability density functions for gathering uncertainty was very intrieging to us.

4.6 Computer Vision

One of our main difficulties was to find a good method to gather probability density functions. It was clear that we needed a digital representation of the probability density functions that subjects would draw. This way the computer can read the probabilities from the probability density function without a human painfully measuring the distances with a ruler.

We initially had the idea that subjects draw the probability density functions on a computer inside a dedicated window. However, we decided against it, because it is rather hard to draw on the computer with a mouse. Drawing by hand is far more accurate and easier for subjects. Which is why we decided to develop the questionnaire in a way that let us use computer vision to detect and extract the probability functions as images. We can then use these probability density function images to read of the probability for certain points and calculate the metrics we are interested in.

We recommend this or a similar approach. An automatic processing pipeline is much less error prone and ensures a high level of objectivity. If correctly implemented the evaluation is independent of the instructor.

In the future, we think it would be beneficial to incorporate even more computer vision. When designing the questionnaire, we added little squares on the right of the answer panels. We used these to evaluate the subject's answers by assigning numbers from one to five. This helped us to created a CSV file with the ordering the subjects chose for each task. Originally we intended to read the squares with the help of computer vision and then apply Machine Learning to recognize the hand written digits. However, due to time constraints we were not able to do so. This would be a great next step or even skipping the hand evaluation and reading and recognizing the answers instead.

4.7 Brier Score