SOFTWARE

Open Access

Software symptomcheckR: an R package for analyzing and visualizing symptom checker triage performance

Marvin Kopka[1*] and Markus A. Feufel[1]

Abstract

Background A major stream of research on symptom checkers aims at evaluating the technology's *predictive accuracy*, but apart from general trends, the results are marked by high variability. Several authors suggest that this variability might in part be due to different assessment methods and a lack of standardization. To improve the reliability of symptom checker evaluation studies, several approaches have been suggested, including standardizing input procedures, the generation of test vignettes, and the assignment of gold standard solutions for these vignettes. Recently, we suggested a third approach––test-theoretic metrics for standardized performance reporting–– to allow systematic and comprehensive comparisons of symptom checker performance. However, calculating these metrics is time-consuming and error prone, which could hamper the use and effectiveness of these metrics.

Results We developed the R package symptomcheckR as an open-source software to assist researchers in calculating standard metrics to evaluate symptom checker performance individually and comparatively and produce publication-ready figures. These metrics include accuracy (by triage level), safety of advice (i.e., rate of correctly or overtriaged cases), comprehensiveness (i.e., how many cases could be entered or were assessed), inclination to overtriage (i.e., how risk-averse a symptom checker is) and a capability comparison score (i.e., a score correcting for case difficulty and comprehensiveness that enables a fair and reliable comparison of different symptom checkers). Each metric can be obtained using a single command and visualized with another command. For the analysis of individual or the comparison of multiple symptom checkers, single commands can be used to produce a comprehensive performance profile that complements the standard focus on accuracy with additional metrics that reveal strengths and weaknesses of symptom checkers.

Conclusions Our package supports ongoing efforts to improve the quality of vignette-based symptom checker evaluation studies by means of standardized methods. Specifically, with our package, adhering to reporting standards and metrics becomes easier, simple, and time efficient. Ultimately, this may help users gain a more systematic understanding of the strengths and limitations of symptom checkers for different use cases (e.g., all-purpose symptom checkers for general medicine versus symptom checkers that aim at improving triage in emergency departments), which can improve patient safety and resource allocation.

Keywords Symptom checker, R package, Standardization, Reporting guidelines, Accuracy, Triage, Software

*Correspondence:
Marvin Kopka
marvin.kopka@tu-berlin.de
Full list of author information is available at the end of the article

Background

Symptom checkers––systems in which laypeople input symptoms to receive potential diagnoses and triage advice [1]––are gaining popularity and drawing attention among health professionals and lay users as well as in the research community [2–4]. Research in this field focuses, on one hand, on the potential impact of symptom checkers on individual users and healthcare systems [5–13], and on the accuracy of these systems on the other [2, 3, 14–17]. For users, it is crucial that symptom checkers give safe advice and prevent potential harm [18], whereas for healthcare systems overtriage could inflate costs and strain scarce resources (e.g., due to unnecessary emergency department visits) [5, 14]. Thus, accurate performance is of great importance for symptom checker success, emphasizing the need of high standards for performance evaluations.

Evaluation studies of symptom checkers show high variability, however, with average accuracy estimates ranging from 27 to 90% [2, 3]. The reasons for the wide range of accuracy estimates are not entirely clear, although a first set of factors might relate to the choice of different evaluation methods, including testing procedures [19], types of case vignettes tested [19–21], and the gold standard solutions assigned to these cases [21, 22]. For instance, not all symptom checkers may be tested with every vignette [20] as some symptom checkers are designed for specialized tasks (such as only addressing pediatric cases), whereas other symptom checkers restrict the types of symptoms that may be entered and processed [16]. As a result, accuracy cannot be effectively compared between these symptom checkers. A second factor might relate to the evaluation metrics used. For instance, to account for different goals such as avoiding individual harm and avoiding unnecessary demand on healthcare systems, some studies report additional metrics such as the safety of advice, although the exact metrics reported differ between studies [14, 16, 23, 24].

As a remedy researchers proposed solutions to standardize evaluation methodologies: Painter et al. proposed several requirements, including standardizing the number of inputters, developing a standardized way of determining a gold standard solution to a case, or developing more reliable vignettes that are more representative for real-world cases [21]. El-Osta et al. examined variability in the vignette creation processes and urged the field to use real-world data instead of artificial vignettes [22]. Meczner et al. examined inputter variability and proposed that coding a vignette as solved (in)correctly by a symptom checker should involve multiple coders and a synthesis of their assessments [25].

In a more recent study, our research team proposed guidelines to enhance symptom checker *reporting*. The metrics we developed focus solely on triage accuracy (as opposed to diagnostic accuracy), because the (final) diagnosis is invariably made by a healthcare professional and is thus a less relevant feature of symptom checkers [16, 17, 26]. The guidelines include various metrics that provide insights into individual symptom checker performance as well as performance comparisons [20]. Most importantly, we suggested quality indicators to control for bias in comparative accuracy estimates (e.g., such as how many cases a symptom checker could be tested with) and to guide the selection of symptom checkers for a specific use case (e.g., implementing it in an emergency department or using it for at-home testing). Using these quality indicators, comparability across different studies can be enhanced and implementation can be guided in a manner that is specific to each use case. To account for sources of bias in the accuracy estimate, we developed a 'Capability Comparison Score' based on classic test theory that adjusts for the difficulty and number of cases entered to allow reliable comparisons between different symptom checkers.

Although most of the proposed metrics can be calculated easily, it is neither cost efficient nor practical for researchers to calculate all metrics by hand. To solve this problem, software solutions can be used. Currently, there is only the psych package [27] available, which can calculate item difficulty, but no other metrics specific for symptom checker evaluations. Since no software is available to assess the performance of one symptom checker or a comparison of multiple ones in a standardized way, future studies are likely to continue reporting differing metrics, which limits the comparability between them. To improve quality standards in symptom checker research, we developed an R package named *symptomcheckR*, which assists users in calculating and reporting standardized metrics on symptom checker performance.

Implementation

The R package was developed to include several metrics for evaluating the (comparative) performance of symptom checkers using the data of the above-mentioned publication as a case study [20]. The metrics complement the commonly used single accuracy measure by shedding light both on its strengths *and* weaknesses. The package is optimized for ease-of-use to allow symptom checker researchers, developers, policymakers, and other stakeholders to quickly analyze the performance of single or multiple symptom checkers. To achieve this, we adhered to key usability principles, ensuring the software is (a) effective by enabling users to generate comprehensive metrics, (b) efficient through providing single commands for each outcome and a unified command structure for all metrics, and (c) easy to use by including an example dataset to simplify data wrangling and commands inspired by natural language [28, 29]. The package builds on the previously published packages dplyr [30], tidyr [31], ggplot2 [32] and ggpubr [33]. It is available on CRAN with open-source code and licensed under the GNU General Public License.

Metrics

In this section, we describe the developed metrics. These metrics originated from a previous study where we analyzed two separate evaluations of the same symptom checkers using identical vignettes, which produced significantly different results [20]. Consequently, we reviewed the literature to identify previously utilized metrics and adapted these concepts into a standardized reporting guideline. The first five metrics are designed to evaluate the performance of an individual symptom checker, but they may be used to compare different symptom checkers as well. For a comparative analysis, the subsequent metrics are necessary: item difficulty can be used to assess the difficulty of vignettes across symptom checkers, whereas the capability comparison score serves as a metric to account for potential sources of bias that may affect accuracy (e.g., how many easy and difficult vignettes could be entered). This ensures more reliable comparisons of the capabilities of different symptom checkers. All metrics in the symptomcheckR package are summarized in Table 1.

Accuracy

Accuracy is defined as the proportion of cases that a symptom checker successfully solves. Although this metric provides an initial insight into the performance of an individual symptom checker, it does not account for varying levels of case urgency or the difficulty of the cases. It can be calculated as:

$$A_{SC} = \frac{1}{n_{SC}}\sum\nolimits_{V=1}^{n_{SC}} X_{SC,V} \text{ with } X_{SC,V} = \begin{cases} 1, R_{SC,V} = T_V \\ 0, R_{SC,V} \neq T_V \end{cases}$$

Table 1 Metrics included in the symptomcheckR package to evaluate the performance of individual symptom checkers and to compare multiple symptom checkers

Metric	R command
Accuracy	get_accuracy()
Accuracy for each triage level	get_accuracy_by_triage()
Safety of advice	get_safety_of_advice()
Comprehensiveness	get_comprehensiveness()
Inclination to overitriage	get_inclination_to_overtriage()
Capability Comparison Score (CCS)	get_ccs()

where A denotes the accuracy, n_{SC} the number of cases a symptom checker was tested with, V the vignette's number, $X_{SC,V}$ whether a case was solved correctly (with $R_{SC,V}$ denoting the recommendation's triage level and T_V the correct triage level).

Accuracy by triage level

To gain more comprehensive insights on how symptom checkers perform in different scenarios, accuracy should be calculated for each triage level separately. For example, some symptom checkers do not advise self-care [16] and are thus not suitable to use on such cases. However, this information is not inferable from an aggregate accuracy. Another example is a symptom checker for emergency departments, which should distinguish particularly well between emergency and non-emergency cases. The following metric thus examines the use-case specific accuracy. The accuracy for each triage level can be calculated as:

$$A_{SC,L} = \frac{1}{n_{SC,L}}\sum\nolimits_{V=1}^{n_{SC,L}} X_{SC,V,L} \text{ with } X_{SC,V,L} = \begin{cases} 1, R_{SC,V} = T_V \\ 0, R_{SC,V} \neq T_V \end{cases}$$

where $A_{SC,L}$ denotes the accuracy the accuracy for a symptom checker on triage level L, $n_{SC,L}$ the number of cases a symptom checker was tested with on the triage level L, V the vignette's number, and $X_{SC,V,L}$ whether a case was solved correctly (with $R_{SC,V}$ denoting the recommendation's triage level and T_V the correct triage level).

Safety of advice

The safety of advice gives an impression on how safe recommendations by a symptom checker are. This might be particularly relevant when evaluating the potential harm of a symptom checker. It indicates the percentage of recommendations that are categorized as being of equal or greater urgency than what is appropriate for a given case and can be calculated as:

$$S_{SC} = \frac{1}{n_{SC}}\sum\nolimits_{V=1}^{n_{SC,L}} X_{SC,V} \text{ with } X_{SC,V} = \begin{cases} 1, R_{SC,V} \geq T_V \\ 0, R_{SC,V} < T_V \end{cases}$$

where S_{SC} denotes the safety, n_{SC} the number of cases a symptom checker was tested with, V the vignette's number, and $X_{SC,V}$ whether the recommendation of a symptom checker SC for a vignette V was safe (with R denoting the recommendation's triage level – higher values indicating higher urgency – and T the correct triage level).

Comprehensiveness

Not all symptom checkers allow entering all cases. If only few symptoms can be entered, a symptom checker might be beneficial for a specific use case, but not for broad implementation. Further, entering only selected cases can bias the accuracy. Thus, the comprehensiveness metric accounts for how many cases could be entered in a symptom checker and can be calculated as:

$$C_{SC} = \frac{1}{n}\sum_{V=1}^{n_{SC}} P_{SC,V}$$

where C_{SC} denotes the comprehensiveness, n the total number of vignettes in the set, and $P_{SC,V}$ whether a symptom checker provided a recommendation for this vignette.

It is important to note that an evaluation should be conducted with a specific aim and thus the same set of vignettes should be used for all symptom checkers [34]. If, for example, a symptom checker that only accepts pediatric cases is included, it will have a low comprehensiveness for general cases. Additionally, researchers should clarify in their descriptions why symptom checkers might have a low comprehensiveness (e.g., because it was a specialized symptom checker or because of aborting data entry according to stopping rules in the protocol).

Inclination to overtriage

Whereas providing safe advice is essential to protect individuals from harm, frequently giving advice with (unnecessarily) high urgency can result in increased healthcare utilization due to the use of a symptom checker [35]. This, in turn, can lead to increased healthcare expenditures and reduced availability of care resources for individuals [5, 10]. Thus, assessing a symptom checker's inclination to overtriage is especially valuable from a systems perspective and can be quantified as the proportion of 'overtriage' errors among all incorrect triage recommendations. It can be calculated as:

$$ITO_{SC} = \frac{\sum_{V \in SC, R_{SC,V} > T_V} 1}{\sum_{V \in SC, R_{SC,V} \neq T_V} 1}$$

where ITO_{SC} denotes the symptom checker's inclination to overtriage, V the vignette, SC the symptom checker, $R_{SC,V}$ the recommendation of a symptom checker SC for the vignette V and T_V the correct triage level for the vignette V.

Item difficulty

When testing multiple symptom checkers with the same cases, some cases might be solved by all symptom checkers and some by none. Item difficulty can be used to determine how difficult a vignette is for symptom checkers to solve. It describes the proportion of symptom checkers that were able to solve a vignette. Thus, an item difficulty of 1 means that the case was easy to solve (as all symptom checkers solved it) and an item difficulty of 0 means that it is particularly difficult (as none solved it correctly). It can be calculated as:

$$ID_V = \frac{C_V}{T_V}$$

where ID_V denotes the item difficulty of a vignette V, C_V the number of symptom checkers that solved a vignette correctly and T_V the total number of symptom checkers that assessed the case.

Capability comparison score

Because not all symptom checkers can be tested with all cases and those cases that can be entered differ in their difficulty [20], solely comparing different symptom checkers according to their accuracy results in biased conclusions. Thus, the capability comparison score accounts for the fact that (a) not all symptom checkers are tested with the same cases and (b) these cases differ in difficulty. It allows more reliable performance comparisons between different symptom checkers and can be calculated as:

$$CCS_{SC} = \frac{\left(\left(\sum_1^V (X_{SC,V} * (1 - ID_V)) - \sum_1^V ((1 - X_{SC,V}) * ID_V)\right)/n_{SC,V}\right) + 1}{2} * 100 \text{ with } X_{SC,V} = \begin{cases} 0, R_{SC,V} = T_V \\ 1, R_{SC,V} \neq T_V \end{cases}$$

where CCS_{SC} denotes the resulting score, SC the symptom checker that is being assessed, $X_{SC,V}$ whether the advice was correct (with R denoting the recommendation's triage level – higher values indicating higher urgency – and T the correct triage level), ID the item difficulty, V the vignette, and $n_{SC,V}$ the number of cases that were entered in the symptom checker.

Inter-rater reliability

In some studies, the same symptom checkers are tested by multiple individuals. To quantify the degree of agreement between the inputters' results, inter-rater reliability can be calculated. Since this data is typically ordinal, we implemented a two-way, absolute agreement, average-measures, mixed intra-class correlation [36].

Visualization

The symptomcheckR can be used to create publication-ready stacked bar plots to visualize all metrics for individual symptom checkers: accuracy (by triage level), safety of advice, comprehensiveness, and inclination to overtriage. Additionally, it can be used to create double-sided bar charts for illustrating the capability comparison score. All charts are color coded for intuitive understanding: desirable outcomes are shown in green, while undesirable ones are red. The color shades are chosen in accordance with inclusive design standards, ensuring they are distinguishable to individuals with color vision deficiencies. The package also includes two additional commands: one to visualize the performance of a single symptom checker across all metrics, and another for a side-by-side performance comparison of multiple symptom checkers. These commands return a ggplot class object, which can be further customized to meet various design requirements and preferences. For combined performance visualizations, the command returns a ggarrange class object.

Included dataset

In the R package, we included a dataset derived from a previous study on the accuracy of different symptom checkers [16, 37]. This study tested different freely accessible symptom checkers in 2020 using a set of 45 vignettes, initially developed by Semigran et al. [17]. These vignettes include both common and uncommon conditions across various medical disciplines with 15 cases each for emergency cases, non-emergency cases and self-care cases. The dataset comprises several symptom checkers with varying degrees of comprehensiveness and can thus be used as an example dataset to demonstrate the different functions of the package.

Results

To demonstrate the usage of the symptomcheckR package, we conduct a full analysis of the included dataset using all commands available in the package. This analysis conforms to the reporting standards recommended for symptom checker audit studies [20]. We present the analysis of both evaluating a single symptom checker and comparing multiple symptom checkers.

Dataset

The dataset can be loaded using the *data(symptomcheckRdat*a) command. It comprises 22 symptom checkers which were tested with 45 vignettes each, yielding a total sample size of 990 observations. 19.6% (194/990) are missing data, i.e., include cases in which a symptom checker did not provide a recommendation.

Analysis of individual symptom checker performance

As an individual symptom checker, we selected Ask NHS, because it contains missing data and can be used to demonstrate all commands. The first step is analyzing its accuracy. This can be done using the *get_accuracy()* command. It includes the arguments *data* for specifying the dataset, *correct* (as a string) to indicate the column in which correct responses are stored as a Boolean (TRUE or FALSE), and *CI* (TRUE or FALSE) to indicate whether 95% confidence intervals should be obtained. It returns a single accuracy value (or three values with the confidence interval):

```
accuracy
1 0.6060606
```

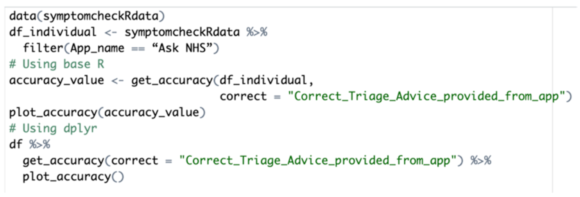
This result can be visualized using *plot_accuracy()*, see Fig. 7. The exemplary code in Fig. 1 first obtains a new data frame containing only ASK NHS data and then shows the analysis using base R and using dplyr.

Next, the accuracy can be analyzed for each triage level using the *get_accuracy_by_triage()* function. It includes the same arguments as the *get_accuracy()* command and adds a *triagelevel* (as a string) argument denoting the column in which the correct triage solution is stored. The output can look like this:

```
 Goldstandard_solution accuracy
 <chr>              <dbl>
1 Emergency           0.333
2 Non-Emergency          0.727
3 Self-care          0.8
```

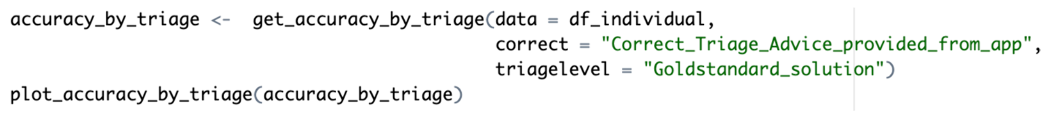
This can again be visualized using *plot_accuracy_by_triage_level()*. The code in Fig. 2 shows this process.

Next, users can visualize the safety of the advice. This can be done using the *get_safety_of_advice()* command with the arguments *data* for specifying the dataset, *triagelevel_correct* (as a string) for specifying the column in which the correct triage level solutions are stored, *triagelevel_advice* (as a string) for specifying the column in which the symptom checker recommendations are stored, *order_triagelevel* (as a vector) for specifying the order of triage levels, starting with the level of highest urgency, and *CI* (TRUE or FALSE) to indicate whether 95% confidence intervals for the percentage should be obtained. The output looks like this:



```
data(symptomcheckRdata)
df_individual <- symptomcheckRdata %>%
  filter(App_name == “Ask NHS”)
# Using base R
accuracy_value <- get_accuracy(df_individual,
                               correct = "Correct_Triage_Advice_provided_from_app")
plot_accuracy(accuracy_value)
# Using dplyr
df %>%
  get_accuracy(correct = "Correct_Triage_Advice_provided_from_app") %>%
  plot_accuracy()
```

Fig. 1 Code to obtain symptom checker accuracy



```
accuracy_by_triage <-  get_accuracy_by_triage(data = df_individual,
                                              correct = "Correct_Triage_Advice_provided_from_app",
                                              triagelevel = "Goldstandard_solution")
plot_accuracy_by_triage(accuracy_by_triage)
```

Fig. 2 Code to obtain and visualize symptom checker accuracy for each triage level

```
$raw_numbers
# A tibble: 2 × 2
# Groups:   safety [2]
  safety          n
  <chr>        <int>
1 safe advice      22
2 unsafe advice    11

$percentage
# A tibble: 1 × 1
  safety_percentage
            <dbl>
1            66.7
```

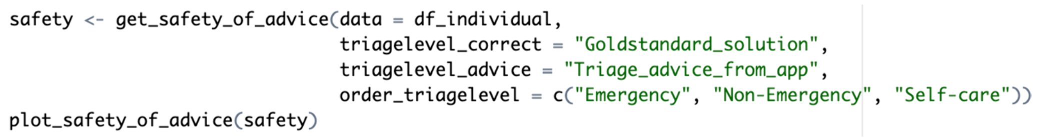
It can be visualized using *plot_safety_of_advice()*, which takes the input of the first command again. The code is shown in Fig. 3.

Afterwards, the comprehensiveness can be of interest. It can be calculated with the command *get_comprehensiveness()* with the arguments *data* for specifying the dataset, *triagelevel_advice* (as a string) for specifying the column in which the symptom checker recommendations are stored, *vector_not_entered* (as a vector) for specifying all values that are coded as no recommendation from a symptom checker or no possibility to enter the case, and *CI* (TRUE or FALSE) to indicate whether 95% confidence intervals for the percentage should be obtained.. The output looks like this:

```
$raw_numbers
# A tibble: 2 × 2
# Groups:   gave_advice [2]
  gave_advice        n
  <chr>           <int>
1 Advice given       33
2 Advice not given    12

$percentage
# A tibble: 1 × 1
  comprehensiveness_percentage
                       <dbl>
1                       73.3
```

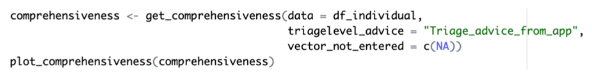
It can again be visualized using *plot_comprehensiveness()* with the result of *get_comprehensiveness()* as the input. The code is shown in Fig. 4.

Lastly, the inclination to overtriage can be calculated using *get_inclination_overtriage()* with the arguments *data* for specifying the dataset, *triagelevel_correct* (as a string) for specifying the column in which the correct triage level solutions are stored, *triagelevel_advice* (as a string) for specifying the column in which the symptom checker recommendations are stored, *order_triagelevel* (as a vector) for specifying the order of triage levels, and *CI* (TRUE or FALSE) to indicate whether 95% confidence intervals for the percentage should be obtained. The triage levels are sorted by urgency, starting with the highest urgency first and the lowest urgency last. The output looks like this:



```
safety <- get_safety_of_advice(data = df_individual,
                               triagelevel_correct = "Goldstandard_solution",
                               triagelevel_advice = "Triage_advice_from_app",
                               order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"))
plot_safety_of_advice(safety)
```

Fig. 3 Code to obtain and visualize the safety of a symptom checker's advice



```
comprehensiveness <- get_comprehensiveness(data = df_individual,
                                           triagelevel_advice = "Triage_advice_from_app",
                                           vector_not_entered = c(NA))
plot_comprehensiveness(comprehensiveness)
```

Fig. 4 Code to obtain and visualize the comprehensiveness of a symptom checker's advice

```
$raw_numbers
# A tibble: 2 × 2
# Groups:   overtriage_undertriage [2]
  overtriage_undertriage     n
  <chr>                  <int>
1 overtriage                11
2 undertriage               22

$percentage
# A tibble: 1 × 1
  inclination_to_overtriage_percentage
                                 <dbl>
1                                 33.3
```

It can be visualized using plot_inclination_overtriage(). The code is shown in Fig. 5.

To get a comprehensive overview of a symptom checker's performance, all metrics can be visualized in a single plot using *plot_performance_single()* with the arguments *data* for specifying the dataset, *triagelevel_correct* (as a string) for specifying the column in which the correct triage level solutions are stored, *triagelevel_advice* (as a string) for specifying the column in which the symptom checker recommendations are stored, *order_triagelevel* (as a vector) for specifying the order of triage levels, and *vector_not_entered* (as a vector) for specifying all values that are coded as no recommendation from a symptom checker or no possibility to enter the case. The code is shown in Fig. 6.

The resulting figure can be seen in Fig. 7.

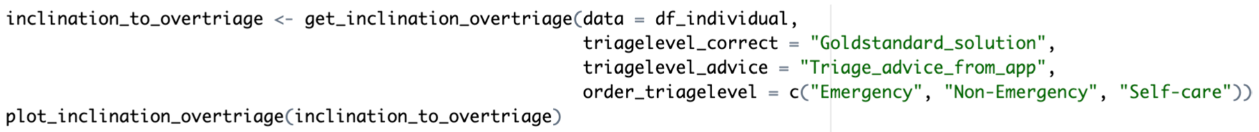
In some cases, multiple people might input the same vignettes into the symptom checkers. Then, only the accuracy (and accuracy for each triage level) can be calculated by creating a new variable that codes the vignette as solved or unsolved according to the researchers' algorithm (e.g., only coding cases solved by both inputters as correct or coding cases correct if at least one author solved it correctly). All other metrics should be reported for each inputter separately as they cannot be calculated meaningfully summed up across all inputters. The inter-rater reliability for these raters can be obtained using the command *get_irr()* with the arguments *data* for specifying the dataset, *ratings* (as a vector) for specifying the columns in which the different ratings are stored, and *order_triagelevel* (as a vector) for specifying the order of triage levels.

Performance comparison of multiple symptom checkers

The same commands can be used to compare multiple symptom checkers. To change the functions' output to comprise multiple symptom checkers, the *apps* argument (as a string) can be added to indicate the column in which the names of different symptom checkers are stored. A full analysis with the same commands as those employed for the evaluation of individual symptom checkers could be conducted as shown in Fig. 8.

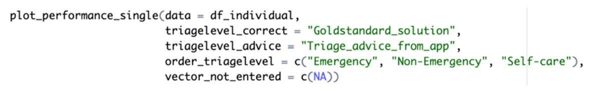
Additionally, users can calculate the item difficulty and a capability comparison score to compare different symptom checkers. The item difficulty can be obtained using *get_item_difficulty()* with the arguments *data* for specifying the dataset, *correct* (as a string) to indicate the column in which correct responses are stored as a Boolean (TRUE or FALSE), and *vignettes* (as a string) to indicate the column in which the vignettes (as numbers or characters) are stored. The capability comparison score can be calculated using *get_ccs()* with the same arguments and an *apps* (as a string) argument indicating the column in which different symptom checker names are stored. It can also be calculated for different triage levels using *get_ccs_by_triage()* with the additional argument *triagelevel* (as a string) to indicate the column in which the correct triage level solutions are stored. Both can be visualized using *plot_ccs()* and *plot_ccs_by_triage()*, see Fig. 11. An exemplary code can look like the code shown in Fig. 9.

Finally, the full dataset can be analyzed and visualized with the command *plot_performance_multiple()* with the arguments *data* for specifying the dataset, *triagelevel_correct* (as a string) for specifying the column in which the correct triage level solutions are stored, *triagelevel_advice* (as a string) for specifying the column in which the symptom checker recommendations are stored, *order_triagelevel* (as a vector) for specifying the order of triage levels, *vector_not_entered* (as a vector) for specifying



```
inclination_to_overtriage <- get_inclination_overtriage(data = df_individual,
                                                        triagelevel_correct = "Goldstandard_solution",
                                                        triagelevel_advice = "Triage_advice_from_app",
                                                        order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"))
plot_inclination_overtriage(inclination_to_overtriage)
```

Fig. 5 Code to obtain and visualize a symptom checker's inclination to overtriage



```
plot_performance_single(data = df_individual,
                        triagelevel_correct = "Goldstandard_solution",
                        triagelevel_advice = "Triage_advice_from_app",
                        order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"),
                        vector_not_entered = c(NA))
```

Fig. 6 Code to visualize all performance metrics of one symptom checker

all values that are coded as no recommendation from a symptom checker or no possibility to enter the case, *vignettes* (as a string) to indicate the column in which the vignettes (as numbers or characters) are stored, and *apps* (as a string) to indicate the column in which the names of different symptom checkers are stored. This results in a figure containing a comparison of all symptom checkers in the dataset across all metrics (see Fig. 11). In this figure, the performance of all metrics is readily apparent. For instance, to identify a symptom checker suitable for general implementation, one should first examine the comprehensiveness section. This helps to rule out symptom checkers that are limited to entering certain cases only. Subsequently, the capability comparison scores can be examined to pinpoint symptom checkers that perform well. After narrowing down the choices, they can be evaluated with respect to their safety to ensure there is no potential harm to users. Additionally, examining the inclination to overtriage can be crucial to determine if it might unduly burden healthcare resources. This process can be repeated and adapted to various use-cases, aiding in selecting the most appropriate symptom checker for a specific use-case. A tabular summary of the figure can be found in the supplementary material. The code to obtain such a figure is shown in Fig. 10.

Discussion

Whereas existing packages such as the psych package offer item difficulty calculation but lack metrics specifically tailored to symptom checkers, the symptomcheckR package presented in this paper is designed to help analyze and visualize various performance metrics of individual symptom checkers and for performance comparisons. While previous studies often focused solely on reporting accuracy, the metrics described reveal potential sources of bias in accuracy and allow drawing more reliable conclusions about the strengths and weaknesses of symptom checkers. For instance, as can be seen in Fig. 11, WebMD had medium accuracy overall. However, a more detailed examination reveals that it is among the best-performing symptom checkers for identifying non-emergency care cases, yet one of the worst performing for self-care cases. WebMD also shows a high comprehensiveness, as all cases could be entered. In contrast, Healthy Children appears to have low overall accuracy and low accuracy for both emergency and non-emergency cases, yet it performs well with self-care cases. In terms of comprehensiveness, it evaluated only few cases because it is a pediatric symptom checker that was tested with general cases. Therefore, despite its low performance in general comparisons with other symptom checkers, it may be effective for identifying self-care cases in children. Such nuances would remain hidden if the analysis were limited to accuracy and become apparent when examining the performance of symptom checkers in such a comparative figure. This way, our package contributes to ongoing research efforts aimed at standardizing evaluation methods and enhancing the quality of symptom checker assessments. There is an increasing number of studies offering recommendations or stating requirements to improve symptom checker assessments based on exploring the effect of different methodological variations (e.g., inputter instructions or gold standard solution assignment) [21, 22]. Because only few studies tend to implement these standards, we believe it is crucial to supplement empirical research with user-friendly software to facilitate implementation of these standards.

Our package has some limitations: it focuses mainly on triage accuracy and does not include commands for assessing diagnostic performance (e.g., evaluating the top diagnosis, the top three, or top twenty diagnoses [17]). However, our accuracy commands may be used for diagnostic accuracy by coding the corresponding responses in a new variable as true or false and using the *get_accuracy()* command. Secondly, users may have collected and stored their data in formats different from our example dataset. Because our package requires a specific data format, users will need to adjust their data format accordingly. To assist with this, we provide an example dataset to facilitate data wrangling. Lastly, the package incorporates current reporting standards. As new metrics emerge, they can be integrated into future versions of the package.

Conclusions

The symptomcheckR package is the first software that enables users to analyze the performance of symptom checkers using multiple metrics and produce publication-ready figures. It also allows more reliable comparisons of different symptom checkers, comprehensive insights into various aspects of their performance, and increases transparency in symptom checker audit studies. Consequently, users can determine the most appropriate symptom checker for a specific use case (e.g., integration in an emergency department) and identify factors that may influence accuracy estimates (such as the exclusive testing of simpler vignettes). These functionalities make the package especially useful for

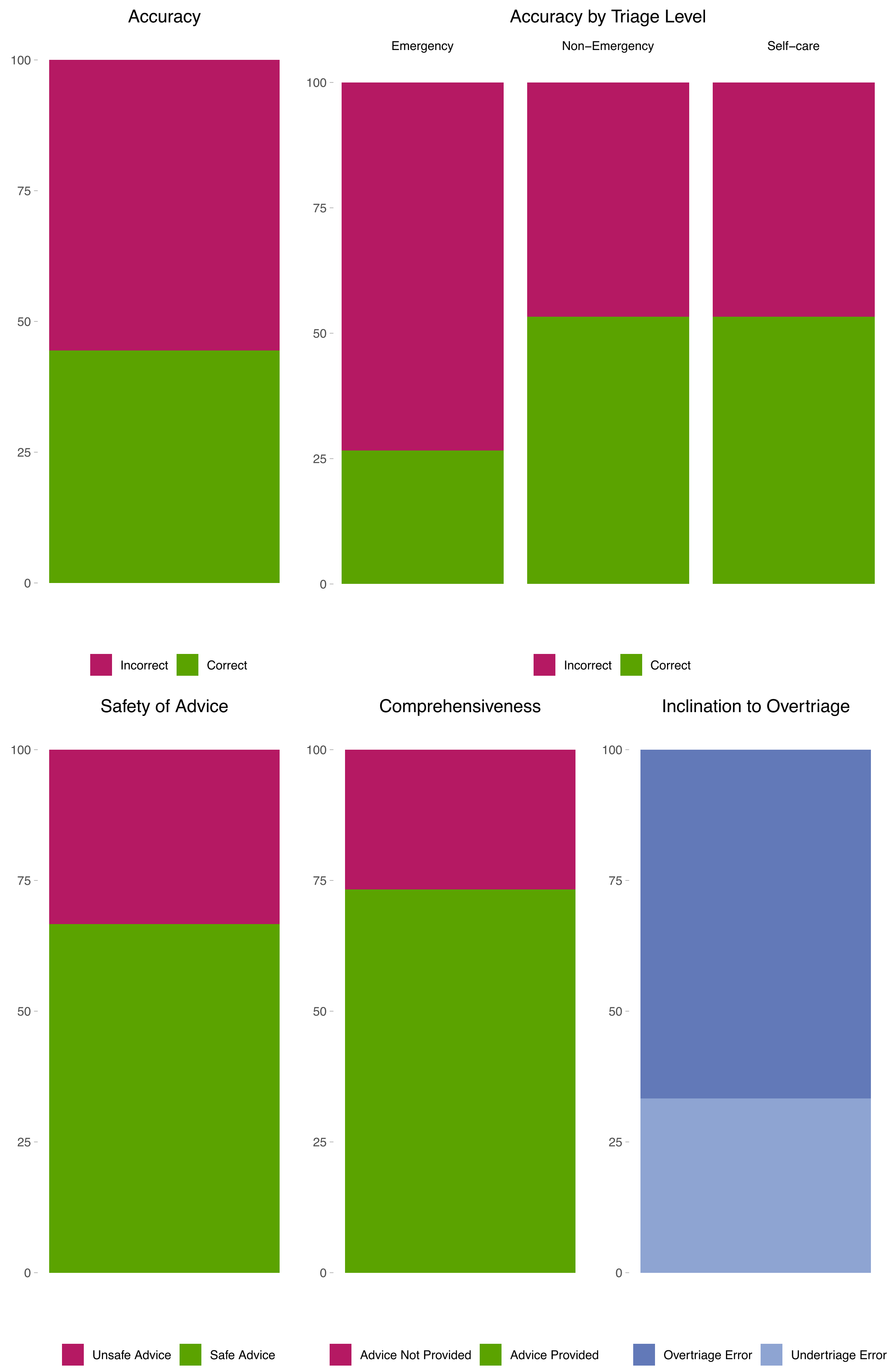
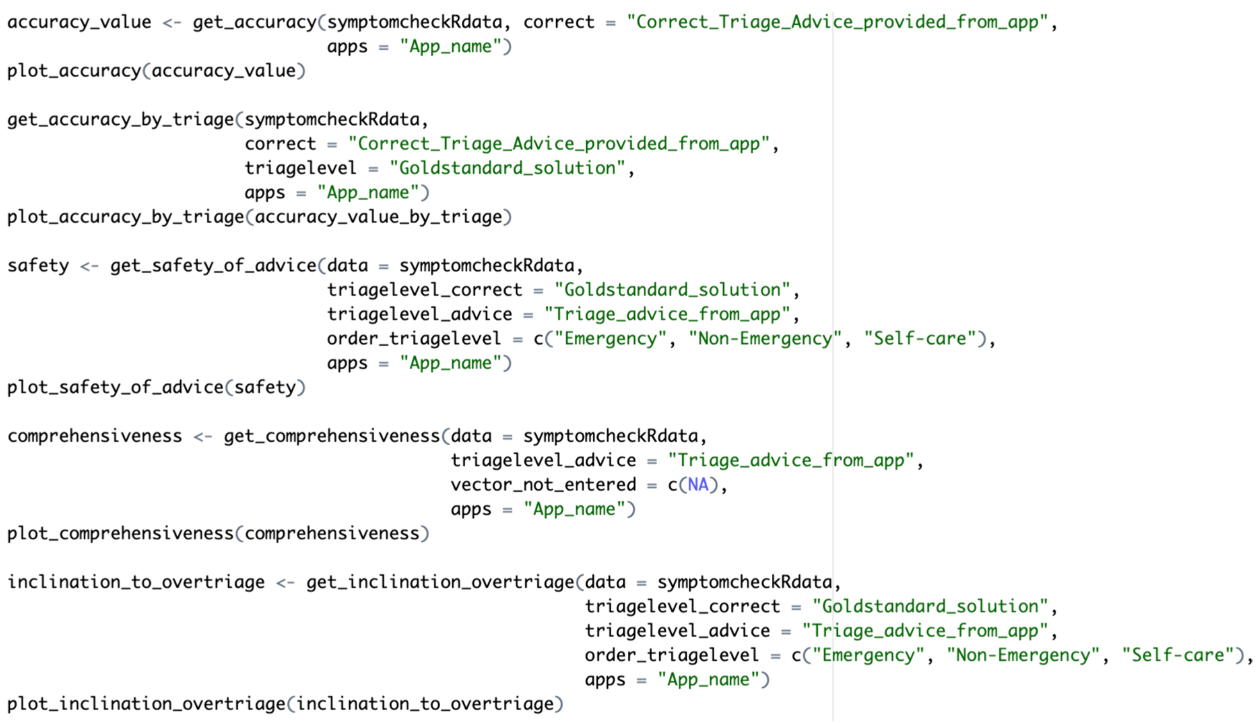
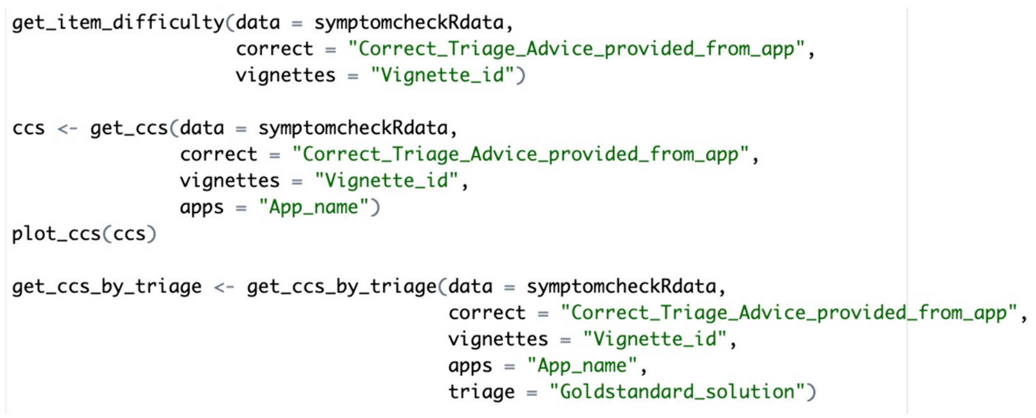


Fig. 7 Publication-ready figure from using the plot_performance_single() function. The figure visualizes all relevant metrics for a symptom checker's triage performance (in this case Ask NHS)



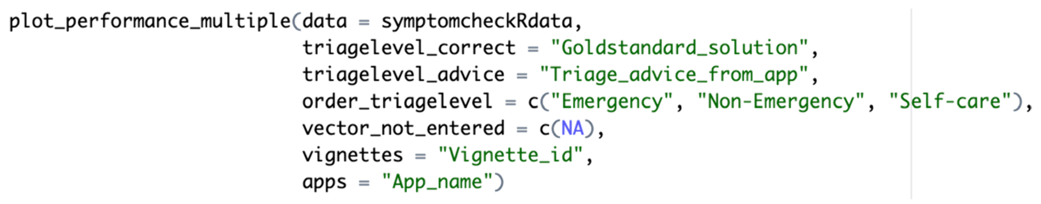
```
accuracy_value <- get_accuracy(symptomcheckRdata, correct = "Correct_Triage_Advice_provided_from_app",
                               apps = "App_name")
plot_accuracy(accuracy_value)

get_accuracy_by_triage(symptomcheckRdata,
                       correct = "Correct_Triage_Advice_provided_from_app",
                       triagelevel = "Goldstandard_solution",
                       apps = "App_name")
plot_accuracy_by_triage(accuracy_value_by_triage)

safety <- get_safety_of_advice(data = symptomcheckRdata,
                               triagelevel_correct = "Goldstandard_solution",
                               triagelevel_advice = "Triage_advice_from_app",
                               order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"),
                               apps = "App_name")
plot_safety_of_advice(safety)

comprehensiveness <- get_comprehensiveness(data = symptomcheckRdata,
                                           triagelevel_advice = "Triage_advice_from_app",
                                           vector_not_entered = c(NA),
                                           apps = "App_name")
plot_comprehensiveness(comprehensiveness)

inclination_to_overtriage <- get_inclination_overtriage(data = symptomcheckRdata,
                                                        triagelevel_correct = "Goldstandard_solution",
                                                        triagelevel_advice = "Triage_advice_from_app",
                                                        order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"),
                                                        apps = "App_name")
plot_inclination_overtriage(inclination_to_overtriage)
```

Fig. 8 Complete code to compare the performance of different symptom checkers



```
get_item_difficulty(data = symptomcheckRdata,
                    correct = "Correct_Triage_Advice_provided_from_app",
                    vignettes = "Vignette_id")

ccs <- get_ccs(data = symptomcheckRdata,
               correct = "Correct_Triage_Advice_provided_from_app",
               vignettes = "Vignette_id",
               apps = "App_name")
plot_ccs(ccs)

get_ccs_by_triage <- get_ccs_by_triage(data = symptomcheckRdata,
                                       correct = "Correct_Triage_Advice_provided_from_app",
                                       vignettes = "Vignette_id",
                                       apps = "App_name",
                                       triage = "Goldstandard_solution")
```

Fig. 9 Code to calculate item difficulty of a vignette and the Capability Comparison Score (CCS) of a symptom checker overall and for each triage level



```
plot_performance_multiple(data = symptomcheckRdata,
                          triagelevel_correct = "Goldstandard_solution",
                          triagelevel_advice = "Triage_advice_from_app",
                          order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"),
                          vector_not_entered = c(NA),
                          vignettes = "Vignette_id",
                          apps = "App_name")
```

Fig. 10 Code to visualize all performance metrics of multiple symptom checkers

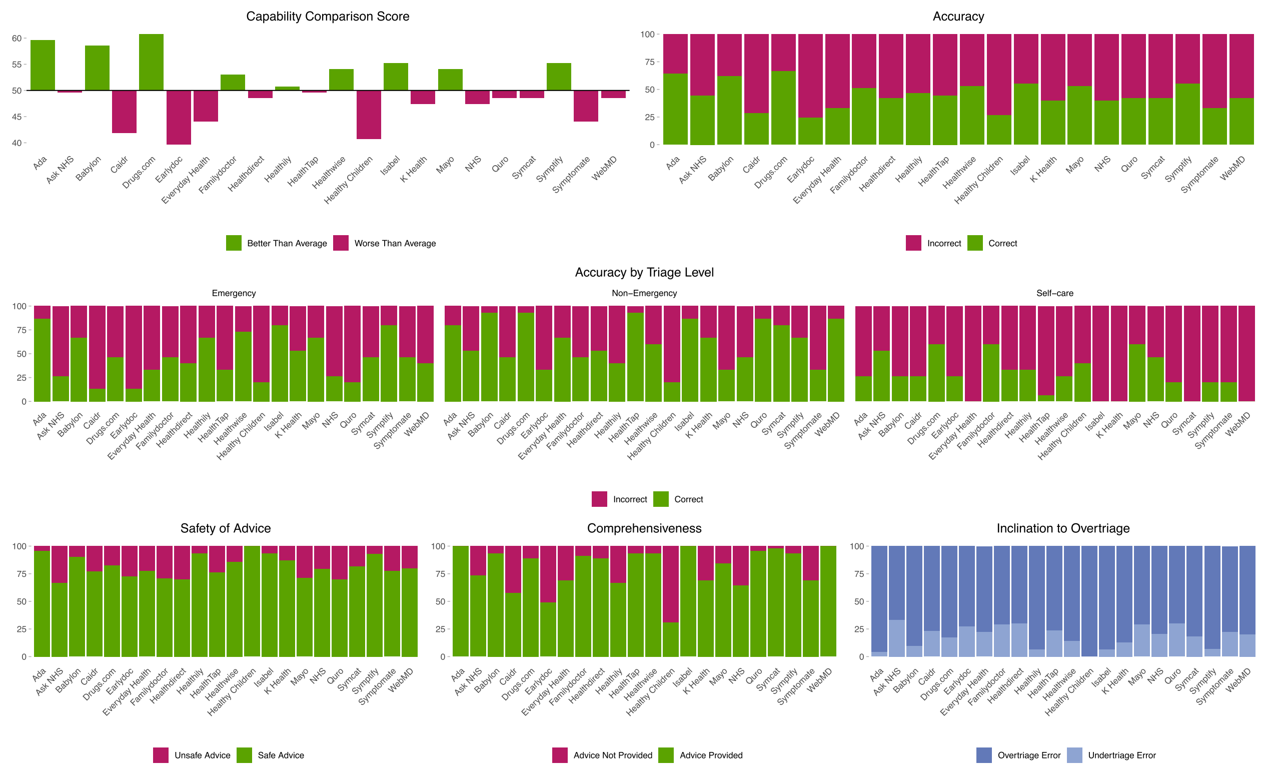


Fig. 11 Publication-ready figure from using the plot_performance_multiple() function. The figure visualizes all relevant metrics for comparing different symptom checkers' triage performance

researchers, as well as for developers and regulatory bodies. We thus encourage these stakeholders to utilize the symptomcheckR package. If used widely, reliable, transparent, and easy-to-use evaluation and reporting standards may help to realize the potential of digital health innovations to improve patient safety and optimize the allocation of healthcare resources. We thus invite the community to contribute to improvements of the package and to develop their own software for other parts of symptom checker evaluation methodology.

Availability and requirements

- Project name: symptomcheckR.
- Project home page: https://github.com/ma-kopka/symptomcheckR
- Operating system: Platform independent.
- Programming language: R.
- Other requirements: dplyr, tidyr, ggplot2, ggpubr, irr.
- License: GNU General Public License.
- Any restrictions to use by non-academics: According to GNU General Public License.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s44247-024-00096-7.

Supplementary Material 1.

Acknowledgements
ChatGPT, an AI by OpenAI, has been used in debugging the code and editing the manuscript.

Authors' contributions
MK conceived of the project, developed the package, and wrote the first draft of the manuscript. MAF provided critical input and all authors worked on manuscript development.

Funding
Open Access funding enabled and organized by Projekt DEAL. The authors received no financial support for the research. It was funded by the home institution of MAF.

Availability of data and materials
The datasets or analyzed during the current study are available in the symptomcheckR package, available on CRAN: https://cran.r-project.org/web/packages/symptomcheckR/index.html

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Author details
[1]Division of Ergonomics, Department of Psychology and Ergonomics (IPA), Technische Universität Berlin, Marchstr. 23, 10587 Berlin, Germany.

Received: 5 February 2024 Accepted: 7 May 2024
Published online: 22 July 2024

References
1. Napierala H, Kopka M, Altendorf MB, Bolanaki M, Schmidt K, Piper SK, et al. Examining the impact of a symptom assessment application on patient-physician interaction among self-referred walk-in patients in the emergency department (AKUSYM): study protocol for a multi-center, randomized controlled, parallel-group superiority trial. Trials. 2022;23(1):791.
2. Wallace W, Chan C, Chidambaram S, Hanna L, Iqbal FM, Acharya A, et al. The diagnostic and triage accuracy of digital and online symptom checker tools: a systematic review. NPJ Digit Med. 2022;5(1):118.
3. Riboli-Sasco E, El-Osta A, Alaa A, Webber I, Karki M, El Asmar ML, et al. Triage and Diagnostic Accuracy of Online Symptom Checkers: Systematic Review. J Med Internet Res. 2023;2(25):e43803.
4. Pairon A, Philips H, Verhoeven V. A scoping review on the use and usefulness of online symptom checkers and triage systems: How to proceed? Front Med. 2023;6(9):1040926.
5. Turner J, Knowles E, Simpson R, Sampson F, Dixon S, Long J, et al. Impact of NHS 111 Online on the NHS 111 telephone service and urgent care system: a mixed-methods study. Health Serv Deliv Res. 2021;9(21):1–148.
6. Aboueid S, Meyer S, Wallace JR, Mahajan S, Chaurasia A. Young Adults' Perspectives on the Use of Symptom Checkers for Self-Triage and Self-Diagnosis: Qualitative Study. JMIR Public Health Surveill. 2021;7(1):e22637.
7. Kopka M, Feufel MA, Balzer F, Schmieding ML. The Triage Capability of Laypersons: Retrospective Exploratory Analysis. JMIR Form Res. 2022;6(10):e38977.
8. Aboueid S, Liu RH, Desta BN, Chaurasia A, Ebrahim S. The Use of Artificially Intelligent Self-Diagnosing Digital Platforms by the General Public: Scoping Review. JMIR Med Inform. 2019;7(2):e13445.
9. Kopka M, Schmieding ML, Rieger T, Roesler E, Balzer F, Feufel MA. Determinants of Laypersons' Trust in Medical Decision Aids: Randomized Controlled Trial. JMIR Hum Factors. 2022;9(2):e35219.
10. Gottliebsen K, Petersson G. Limited Evidence of Benefits of Patient Operated Intelligent Primary Care Triage Tools: Findings of a Literature Review. BMJ Health Care Inform. 2020;27(1):e100114.
11. Verzantvoort NCM, Teunis T, Verheij TJM, van der Velden AW. Self-Triage for Acute Primary Care via a Smartphone Application: Practical, Safe and Efficient? PLoS ONE. 2018;13(6):e0199284.
12. Kopka M, Scatturin L, Napierala H, Fürstenau D, Feufel MA, Balzer F, et al. Characteristics of Users and Nonusers of Symptom Checkers in Germany: Cross-Sectional Survey Study. J Med Internet Res. 2023;20(25):e46231.
13. Arellano Carmona K, Chittamuru D, Kravitz RL, Ramondt S, Ramírez AS. Health Information Seeking From an Intelligent Web-Based Symptom Checker: Cross-sectional Questionnaire Study. J Med Internet Res. 2022;24(8):e36322.
14. Ceney A, Tolond S, Glowinski A, Marks B, Swift S, Palser T. Accuracy of online symptom checkers and the potential impact on service utilisation. Wilson FA, editor. PLoS ONE. 2021;16(7):e0254088.
15. Gräf M, Knitza J, Leipe J, Krusche M, Welcker M, Kuhn S, et al. Comparison of physician and artificial intelligence-based symptom checker diagnostic accuracy. Rheumatol Int. 2022;42(12):2167–76.
16. Schmieding ML, Kopka M, Schmidt K, Schulz-Niethammer S, Balzer F, Feufel MA. Triage Accuracy of Symptom Checker Apps: 5-Year Follow-up Evaluation. J Med Internet Res. 2022;24(5):e31810.
17. Semigran HL, Linder JA, Gidengil C, Mehrotra A. Evaluation of Symptom Checkers for Self Diagnosis and Triage: Audit Study. BMJ. 2015;8(351):1–9.
18. Chambers D, Cantrell AJ, Johnson M, Preston L, Baxter SK, Booth A, et al. Digital and Online Symptom Checkers and Health Assessment/Triage Services for Urgent Health Problems: Systematic Review. BMJ Open. 2019;9(8):e027743.
19. Ilicki J. Challenges in evaluating the accuracy of AI-containing digital triage systems: A systematic review. PLoS ONE. 2022;17(12):e0279636.
20. Kopka M, Feufel MA, Berner ES, Schmieding ML. How suitable are clinical vignettes for the evaluation of symptom checker apps? A test theoretical perspective. Digit Health. 2023;9:20552076231194930.
21. Painter A, Hayhoe B, Riboli-Sasco E, El-Osta A. Online Symptom Checkers: Recommendations for a Vignette-Based Clinical Evaluation Standard. J Med Internet Res. 2022;24(10):e37408.
22. El-Osta A, Webber I, Alaa A, Bagkeris E, Mian S, Sharabiani M Taghavi Azar, et al. What is the suitability of clinical vignettes in benchmarking the performance of online symptom checkers? An audit study. BMJ Open. 2022;12(4):e053566.
23. Chan F, Lai S, Pieterman M, Richardson L, Singh A, Peters J, et al. Performance of a new symptom checker in patient triage: Canadian cohort study. PLoS ONE. 2021;16(12):e0260696.
24. Fraser HSF, Cohan G, Koehler C, Anderson J, Lawrence A, Pateña J, et al. Evaluation of Diagnostic and Triage Accuracy and Usability of a Symptom Checker in an Emergency Department: Observational Study. JMIR Mhealth Uhealth. 2022;10(9):e38364.
25. Meczner A, Cohen N, Qureshi A, Reza M, Blount E, Malak T. Accuracy as a composite measure for the assessment of online symptom checkers in vignette studies: Evaluation of current practice and recommendations (Preprint). Journal of Medical Internet Research; 2023 Jun [cited 2024 Jan 5]. Available from: http://preprints.jmir.org/preprint/49907
26. Hill MG, Sim M, Mills B. The Quality of Diagnosis and Triage Advice Provided by Free Online Symptom Checkers and Apps in Australia. Med J Aust. 2020;212(11):514–9.
27. Revelle W. psych: Procedures for Psychological, Psychometric, and Personality Research. Evanston, Illinois: Northwestern University; 2022. Available from: https://CRAN.R-project.org/package=psych
28. Bevan N, Carter J, Earthy J, Geis T, Harker S. New ISO Standards for Usability, Usability Reports and Usability Measures. In: Kurosu M, editor. Human-Computer Interaction Theory, Design, Development and Practice. Cham: Springer International Publishing; 2016 [cited 2024 Jan 5]. p. 268–78. (Lecture Notes in Computer Science; vol. 9731). Available from: http://link.springer.com/https://doi.org/10.1007/978-3-319-39510-4_25
29. Good J, Howland K. Programming language, natural language? Supporting the diverse computational activities of novice programmers. J Vis Lang Comput. 2017;39:78–92.
30. Wickham H. dplyr: A Grammar of Data Manipulation. 2023. Available from: https://dplyr.tidyverse.org
31. Wickham H. tidyr: Tidy Messy Data. 2023. Available from: https://tidyr.tidyverse.org
32. Wickham H. ggplot2: Elegant Graphics for Data Analysis. 2016. Available from: https://ggplot2.tidyverse.org
33. Kassambara A. ggpubr: "ggplot2" Based Publication Ready Plots. 2023. Available from: https://rpkgs.datanovia.com/ggpubr/
34. Kopka M, Napierala H, Privoznik M, Sapunova D, Zhang S, Feufel M. Evaluating self-triage accuracy of laypeople, symptom-assessment apps, and large language models: A framework for case vignette development using a representative design approach (RepVig). medRxiv; 2024 [cited 2024 Apr 3]. p. 2024.04.02.24305193. Available from: https://www.medrxiv.org/content/https://doi.org/10.1101/2024.04.02.24305193v1
35. Winn AN, Somai M, Fergestrom N, Crotty BH. Association of Use of Online Symptom Checkers With Patients' Plans for Seeking Care. JAMA Netw Open. 2019;2(12):1–3.
36. Hallgren KA. Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial. TQMP. 2012;8(1):23–34.
37. Schmieding ML, Kopka M, Schmidt K, Schulz-Niethammer S, Balzer F, Feufel M. Data Set on Accuracy of Symptom Checker Apps in 2020. Zenodo; 2022 [cited 2023 Dec 15]. Available from: https://zenodo.org/record/6054092

Publisher's Note