Package 'symptomcheckR'

April 16, 2024

Type Package

Title Analyzing and Visualizing Symptom Checker Performance

Version 0.1.3

Maintainer Marvin Kopka <marvin.kopka@tu-berlin.de>

Description Easily analyze and visualize the performance of symptom checkers. This package can be used to gain comprehensive insights into the performance of single symptom checkers or the performance of multiple symptom checkers. It can be used to easily compare these symptom checkers across several metrics to gain an understanding of their strengths and weaknesses. The metrics are developed in Kopka et al. (2023) <doi:10.1177/20552076231194929>.

License GPL-3

URL https://github.com/ma-kopka/symptomcheckR

BugReports https://github.com/ma-kopka/symptomcheckR/issues

Depends R (>= 3.5.0)

Imports dplyr (>= 1.0.0), ggplot2 (>= 3.2.0), ggpubr (>= 0.6.0), tidyr (>= 1.3.0), irr (>= 0.84.1)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author Marvin Kopka [cre, aut] (<https://orcid.org/0000-0003-3848-1471>), Malte L. Schmieding [aut] (<https://orcid.org/0000-0002-5844-7791>), Markus A. Feufel [aut] (<https://orcid.org/0000-0003-0563-8831>)

Repository CRAN

Date/Publication 2024-04-16 20:40:06 UTC

R topics documented:

`get_accuracy` *get_accuracy*

Description

Calculates the accuracy of one or multiple symptom checkers

Usage

```
get_accuracy(data, correct, apps = NULL, CI = FALSE)
```

Arguments

`data`	A dataframe
`correct`	A string indicating the column name storing if the symptom checker solved the case (TRUE or FALSE)
`apps`	A string indicating the column name storing the app names (optional)
`CI`	A Boolean (TRUE or FALSE) indicating whether 95% confidence intervals should be output (optional)

Value

A data frame object containing the accuracy of the symptom checker or the accuracy of multiple symptom checkers. Use the apps argument to calculate this metric for multiple symptom checkers.

Examples

```
data(symptomcheckRdata)
accuracy <- get_accuracy(
  data = symptomcheckRdata,
  correct = "Correct_Triage_Advice_provided_from_app",
  apps = "App_name",
  CI = TRUE
  )
```

`get_accuracy_by_triage` *get_accuracy_by_triage*

Description

Calculates the accuracy on each triage level for one or multiple symptom checkers

Usage

```
get_accuracy_by_triage(data, correct, triagelevel, apps = NULL, CI = FALSE)
```

Arguments

`data`	A dataframe
`correct`	A string indicating the column name storing if the symptom checker solved the case (TRUE or FALSE)
`triagelevel`	A string indicating the column name storing the correct triage solutions
`apps`	A string indicating the column name storing the app names (optional)
`CI`	A Boolean (TRUE or FALSE) indicating whether 95% confidence intervals should be output (optional)

Value

A data frame object containing the accuracy on each triage level (of one or multiple symptom checkers) or the accuracy of multiple symptom checkers. Use the apps argument to calculate this metric for multiple symptom checkers.

Examples

```
data(symptomcheckRdata)
accuracy_by_triage <- get_accuracy_by_triage(
  data = symptomcheckRdata,
  correct = "Correct_Triage_Advice_provided_from_app",
  triagelevel = "Goldstandard_solution",
  apps = "App_name",
  CI = TRUE
  )
```

get_ccs *get_ccs*

Description

Calculates the Capability Comparison Score (CCS) for each symptom checker

Usage

```
get_ccs(data, correct, vignettes, apps)
```

Arguments

data	A dataframe
correct	A string indicating the column name storing if the symptom checker solved the case (TRUE or FALSE)
vignettes	A string indicating the column name storing the vignette or vignette number
apps	A string indicating the column name storing the app names

Value

A data frame object containing the capability comparison score for each symptom checker.

Examples

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

get_ccs_by_triage *get_ccs_by_triage*

Description

Calculates the Capability Comparison Score (CCS) for each symptom checker and each triage level

Usage

```
get_ccs_by_triage(data, correct, vignettes, apps, triagelevel)
```

Arguments

data	A dataframe
correct	A string indicating the column name storing if the symptom checker solved the case (TRUE or FALSE)
vignettes	A string indicating the column name storing the vignette or vignette number
apps	A string indicating the column name storing the app names
triagelevel	A string indicating the column name storing the correct triage solutions

Value

A data frame object containing the capability comparison score for each symptom checker on each triage level.

Examples

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

---

get_comprehensiveness *get_comprehensiveness*

Description

Calculates the comprehensiveness for one or multiple symptom checkers

Usage

```
get_comprehensiveness(
  data,
  triagelevel_advice,
  vector_not_entered,
  apps = NULL,
  CI = FALSE
)
```

Arguments

data	A dataframe
triagelevel_advice	A string indicating the column name storing the recommendation of a symptom checker for a case
vector_not_entered	A vector indicating the values in which missing values are coded (e.g., as NA or a specified value such as -99)
apps	A string indicating the column name storing the app names
CI	A Boolean (TRUE or FALSE) indicating whether 95% confidence intervals should be output (optional)

Value

A list containing both a raw number and the percentage of comprehensiveness for one or multiple symptom checkers

Examples

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

---

get_inclination_to_overtriage *get_inclination_to_overtriage*

Description

Calculates the inclination to overtriage for one or multiple symptom checkers

Usage

```
get_inclination_to_overtriage(
  data,
  triagelevel_correct,
  triagelevel_advice,
  order_triagelevel,
  apps = NULL,
  CI = FALSE
)
```

Arguments

data	A dataframe
triagelevel_correct	A string indicating the column name storing the correct triage solutions
triagelevel_advice	A string indicating the column name storing the recommendation of a symptom checker for a case
order_triagelevel	A vector indicating the order of triage levels. The triage level with highest urgency should be the first value and the triage level with lowest urgency the last value.
apps	A string indicating the column name storing the app names (optional)
CI	A Boolean (TRUE or FALSE) indicating whether 95% confidence intervals should be output (optional)

Value

A list containing both a raw number and the percentage of the inclination to overtriage for one or multiple symptom checkers

Examples

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

---

get_irr *get_irr*

Description

Calculates the inter-rater reliability of multiple raters using a two-way, absolute agreement, average-measures, mixed intra-class correlation

Usage

```
get_irr(data, ratings, order_triagelevel)
```

Arguments

data	A dataframe
ratings	A vector indicating the column names storing the ratings of reach rater
order_triagelevel	A vector indicating the order of triage levels. The triage level with highest urgency should be the first value and the triage level with lowest urgency the last value.

Value

A list containing the inter-rater reliability

Examples

```
## Not run:
#' irr <- get_irr(
  data = df,
  ratings = c("datarater1", "datarater2", "datarater3"),
  order_triagelevel = c("Emergency", "Non-Emergency", "Self-care"),
  )

## End(Not run)
```

---

get_item_difficulty *get_item_difficulty*

Description

Calculates the item difficulty for each case / vignette

Usage

```
get_item_difficulty(data, correct, vignettes)
```

Arguments

data	A dataframe
correct	A string indicating the column name storing if the symptom checker solved the case (TRUE or FALSE)
vignettes	A string indicating the column name storing the vignette or vignette number

Value

A data frame object containing the item difficulty for each vignette

Examples

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

`get_safety_of_advice` *get_safety_of_advice*

Description

Calculates the safety of advice for one or multiple symptom checkers

Usage

```
get_safety_of_advice(
  data,
  triagelevel_correct,
  triagelevel_advice,
  order_triagelevel,
  apps = NULL,
  CI = FALSE
)
```

Arguments

`data`	A dataframe
`triagelevel_correct`	A string indicating the column name storing the correct triage solutions
`triagelevel_advice`	A string indicating the column name storing the recommendation of a symptom checker for a case
`order_triagelevel`	A vector indicating the order of triage levels. The triage level with highest urgency should be the first value and the triage level with lowest urgency the last value.
`apps`	A string indicating the column name storing the app names (optional)
`CI`	A Boolean (TRUE or FALSE) indicating whether 95% confidence intervals should be output (optional)

Value

A list containing both a raw number and the percentage of safe advice for one or multiple symptom checkers

Examples

```
data(symptomcheckRdata)
safety_of_advice <- 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",
  CI = TRUE
  )
```

---

plot_accuracy *plot_accuracy*

Description

Plots the accuracy for one or multiple symptom checkers

Usage

```
plot_accuracy(input)
```

Arguments

input	A dataframe containing the output of get_accuracy()

Value

A ggplot object visualizing the accuracy for either one or multiple symptom checkers

Examples

```
data(symptomcheckRdata)
accuracy <- get_accuracy(
  data = symptomcheckRdata,
  correct = "Correct_Triage_Advice_provided_from_app",
  apps = "App_name"
  )
accuracy_plot <- plot_accuracy(accuracy)
```

plot_accuracy_by_triage *plot_accuracy_by_triage*

Description

Plots the accuracy on each triage level for one or multiple symptom checkers

Usage

```
plot_accuracy_by_triage(input)
```

Arguments

input	A dataframe containing the output of get_accuracy_by_triage()

Value

A ggplot object visualizing the accuracy on each triage level for either one or multiple symptom checkers

Examples

```
data(symptomcheckRdata)
accuracy_by_triage <- get_accuracy_by_triage(
  data = symptomcheckRdata,
  correct = "Correct_Triage_Advice_provided_from_app",
  triagelevel = "Goldstandard_solution",
  apps = "App_name"
  )
accuracy_triage_plot <- plot_accuracy_by_triage(accuracy_by_triage)
```

plot_ccs *plot_ccs*

Description

Plots the Capability Comparison Score (CCS) for multiple symptom checkers

Usage

```
plot_ccs(input)
```

Arguments

input	A dataframe containing the output of get_ccs()

Value

A ggplot object visualizing the CCS for multiple symptom checkers

Examples

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

`plot_comprehensiveness` *plot_comprehensiveness*

Description

Plots the comprehensiveness for one or multiple symptom checkers

Usage

```
plot_comprehensiveness(input)
```

Arguments

`input`	A dataframe containing the output of get_comprehensiveness()

Value

A ggplot object visualizing the comprehensiveness for either one or multiple symptom checkers

Examples

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

plot_inclination_to_overtriage *plot_inclination_to_overtriage*

Description

Plots the inclination to overtriage for one or multiple symptom checkers

Usage

```
plot_inclination_to_overtriage(input)
```

Arguments

input	A dataframe containing the output of get_inclination_to_overtriage()

Value

A ggplot object visualizing the inclination to overtriage for either one or multiple symptom checkers

Examples

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

plot_performance_multiple *plot_performance_multiple*

Description

Plots the all performance metrics for all symptom checkers in dataframe

Usage

```
plot_performance_multiple(
  data,
  triagelevel_correct,
  triagelevel_advice,
  order_triagelevel,
  vector_not_entered,
  vignettes,
  apps
)
```

Arguments

data	A dataframe
triagelevel_correct	A string indicating the column name storing the correct triage solutions
triagelevel_advice	A string indicating the column name storing the recommendation of a symptom checker for a case
order_triagelevel	A vector indicating the order of triage levels. The triage level with highest urgency should be the first value and the triage level with lowest urgency the last value.
vector_not_entered	A vector indicating the values in which missing values are coded (e.g., as NA or a specified value such as -99)
vignettes	A string indicating the column name storing the vignette or vignette number
apps	A string indicating the column name storing the app names

Value

A ggplot object visualizing all performance metrics for all symptom checkers in dataframe

Examples

```
data(symptomcheckRdata)
performance_plot <- 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"
  )
```

plot_performance_single *plot_performance_single*

Description

Plots all performance metrics for one symptom checker

Usage

```
plot_performance_single(
  data,
  triagelevel_correct,
  triagelevel_advice,
  order_triagelevel,
  vector_not_entered
)
```

Arguments

data	A dataframe
triagelevel_correct	A string indicating the column name storing the correct triage solutions
triagelevel_advice	A string indicating the column name storing the recommendation of a symptom checker for a case
order_triagelevel	A vector indicating the order of triage levels. The triage level with highest urgency should be the first value and the triage level with lowest urgency the last value.
vector_not_entered	A vector indicating the values in which missing values are coded (e.g., as NA or a specified value such as -99)

Value

A ggplot object visualizing all performance metrics for one symptom checker

Examples

```
data(symptomcheckRdata)
performance_plot <- plot_performance_single(
  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)
  )
```

plot_safety_of_advice *plot_safety_of_advice*

Description

Plots the safety of advice for one or multiple symptom checkers

Usage

```
plot_safety_of_advice(input)
```

Arguments

input	A dataframe containing the output of get_safety_of_advice()

Value

A ggplot object visualizing the safety of advice for either one or multiple symptom checkers

Examples

```
data(symptomcheckRdata)
safety_of_advice <- 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"
  )
safety_plot <- plot_safety_of_advice(safety_of_advice)
```

symptomcheckRdata *Data on the performance of different symptom checkers*

Description

Dataset generated by Schmieding et al. that tested different symptom checkers with 45 vignettes in 2020. It includes the solution to the case vignettes and the advice each symptom checker gave.

Usage

```
data(symptomcheckRdata)
```

Format

An object of class `"data.frame"`

App_name The name of the app used in this evaluation)

Vignette_id An identificator referencing the number of each vignette (same numbers indicate the same vignettes)

Triage_advice_from_app The triage advice the app recommended for this case (Emergency, Non-Emergency or Self-care)

Goldstandard_solution The goldstandard solution for this vignette

Correct_Triage_Advice_provided_from_app A Boolean whether the app provided the correct advice)

References

This data set was created by Schmieding et al.: https://doi.org/10.5281/zenodo.6054092

Examples

```
data(symptomcheckRdata)
head(symptomcheckRdata)
```

Index