Package 'safetyGraphics'

February 4, 2019

Title Create Interactive Graphics Related to Clinical Trial Safety

Version 0.7.3

Maintainer Jeremy Wildfire <jeremy_wildfire@rhoworld.com>

Description A framework for evaluation of clinical trial safety. Users can interactively explore their data using the 'Shiny' application or create standalone 'htmlwidget' charts. Interactive charts are built using 'd3.js' and 'webcharts.js' 'JavaScript' libraries.

Depends R (>= 3.5)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat, shinytest, knitr

Imports dplyr, htmlwidgets, purrr, stringr, jsonlite, shiny, magrittr, DT, shinyjs, rmarkdown, rlang

VignetteBuilder knitr

NeedsCompilation no

Author Jeremy Wildfire [cre, aut],
Becca Krouse [aut],
Preston Burns [aut],
Xiao Ni [aut],
James Buchanan [aut],
Susan Duke [aut],
Rho Inc. [cph]

Repository CRAN

Date/Publication 2019-02-04 20:40:03 UTC

R topics documented:

`adlbc` *Safety measures sample data*

Description

A dataset containing anonymized lab data from a clinical trial in the CDISC ADaM format. The structure is 1 record per measure per visit per participant. See a full description of the ADaM data standard here.

Usage

```
adlbc
```

Format

A data frame with 10288 rows and 46 variables.

STUDYID Study Identifier

SUBJID Subject Identifier for the Study

USUBJID Unique Subject Identifier

TRTP Planned Treatment

TRTPN Planned Treatment (N)

TRTA Actual Treatment

TRTAN Actual Treatment (N)

TRTSDT Date of First Exposure to Treatment

TRTEDT Date of Last Exposure to Treatment

AGE Age

AGEGR1 Age Group
AGEGR1N Age Group (N)
RACE Race
RACEN Race (N)
SEX Sex
COMP24FL Completers Flag
DSRAEFL Discontinued due to AE?
SAFFL Safety Population Flag
AVISIT Analysis Visit
AVISITN Analysis Visit (N)
ADY Analysis Relative Day
ADT Analysis Relative Date
VISIT Visit
VISITNUM Visit (N)
PARAM Parameter
PARAMCD Parameter Code
PARAMN Parameter (N)
PARCAT1 Parameter Category
AVAL Analysis Value
BASE Baseline Value
CHG Change from Baseline
A1LO Analysis Normal Range Lower Limit
A1HI Analysis Normal Range Upper Limit
R2A1LO Ratio to Low limit of Analysis Range
R2A1HI Ratio to High limit of Analysis Range
BR2A1LO Base Ratio to Analysis Range 1 Lower Lim
BR2A1HI Base Ratio to Analysis Range 1 Upper Lim
ANL01FL Analysis Population Flag
ALBTRVAL Amount Threshold Range
ANRIND Analysis Reference Range Indicator
BNRIND Baseline Reference Range Indicator
ABLFL Baseline Record Flag
AENTMTFL Analysis End Date Flag
LBSEQ Lab Sequence Number
LBNRIND Reference Range Indicator
LBSTRESN Numeric Result/Finding in Std Units

Source

https://github.com/RhoInc/data-library

checkColumnSetting *Check that a setting parameter has a matching data column*

Description

Checks that a single parameter from the settings list matches a column name in a specified data set

Usage

```
checkColumnSetting(key, settings, data)
```

Arguments

key	a list (like those provided by `getSettingKeys()`) defining the position of parameter in the settings object.
settings	The settings list used to generate a chart like `eDISH()`
data	A data frame to check for the specified column

Details

This function compares settings with column names as part of the `validateSettings()` function. More specifically, the function checks whether the `key` in a `settings` object matches a column in `"data"`.

Value

A list containing the results of the check following the format specified in `validateSettings()[["checkList"]]`

Examples

```
testSettings<-generateSettings(standard="AdAM")
testSettings$filters<-list()
testSettings$filters[[1]]<-list(value_col="RACE",label="Race")
testSettings$filters[[2]]<-list(value_col=NULL,label="No Column")
testSettings$filters[[3]]<-list(value_col="NotAColumn",label="Invalid Column")

#pass ($valid == TRUE)
safetyGraphics:::checkColumnSetting(key=list("id_col"),
                                    settings=testSettings, adlbc)

#pass
safetyGraphics:::checkColumnSetting(key=list("filters",1,"value_col"),
                                    settings=testSettings, adlbc)

#NULL column pass
safetyGraphics:::checkColumnSetting(key=list("filters",2,"value_col"),
                                    settings=testSettings, adlbc)
```

```
#invalid column fails
safetyGraphics:::checkColumnSetting(key=list("filters",3,"value_col"),
                                    settings=testSettings, adlbc)
```

---

checkFieldSettings *Check that a setting parameter has a matching data field*

Description

Checks that a single parameter from the settings list matches a field value in a specified data set

Usage

```
checkFieldSettings(fieldKey, settings, data)
```

Arguments

fieldKey	a list (like those provided by getSettingKeys()) defining the position of parameter in the settings object.
settings	The settings list used to generate a chart like eDISH()
data	A data frame to check for the specified field

Details

This function compares settings with field values as part of the validateSettings() function. More specifically, the function checks whether the fieldKey in a settings object matches a column/field combination in "data". The function makes 2 major assumptions about the structure of the settings object. First, it assumes that the first value in fieldKey is "settingName_values" and there is a corresponding "settingName_col" setting that defines the column to search for field-level data. Second, it expects that the value specified by key/settings is a list, and that each value in the list is a field of the variable above.

Value

A list containing the results of the check following the format specified in validateSettings()[["checkList"]]

Examples

```
testSettings<-generateSettings(standard="AdAM")

#list of 4 checks. all pass ($valid ==TRUE)
safetyGraphics:::checkFieldSettings(fieldKey=list("measure_values"),
                                    settings=testSettings, adlbc)
```

checkNumericColumns *Check that settings for mapping numeric data are associated with numeric columns*

Description

Check that settings for mapping numeric data are associated with numeric columns

Usage

```
checkNumericColumns(key, settings, data)
```

Arguments

key	a list (like those provided by getSettingKeys()) defining the position of parameter in the settings object.
settings	The settings list used to generate a chart like eDISH()
data	A data frame to check for the specified numeric column

Value

A list containing the results of the check following the format specified in validateSettings()[["checkList"]]

Examples

```
testSettings<-generateSettings(standard="AdAM")
#pass ($valid == TRUE)
safetyGraphics:::checkSettingProvided(key=list("id_col"),settings=testSettings)

#fails since filters aren't specified by default
safetyGraphics:::checkSettingProvided(key=list("filters"),settings=testSettings)

#fails since groups aren't specified by default
safetyGraphics:::checkSettingProvided(key=list("groups",1,"value_col"),settings=testSettings)
```

checkSettingProvided *Check that the user has provided a valid for a given settings parameter*

Description

Checks that a single required parameter from the settings list is provided by the user

Usage

```
checkSettingProvided(key, settings)
```

Arguments

key	a list (like those provided by getSettingKeys()) defining the position of parameter in the settings object.
settings	The settings list used to generate a chart like eDISH()

Value

A list containing the results of the check following the format specified in validateSettings()[["checkList"]]

Examples

```
testSettings<-generateSettings(standard="AdAM")

#pass ($valid == TRUE)
safetyGraphics:::checkSettingProvided(key=list("id_col"),
                                      settings=testSettings)

#fails since filters aren't specified by default
safetyGraphics:::checkSettingProvided(key=list("filters"),
                                      settings=testSettings)

#fails since groups aren't specified by default
safetyGraphics:::checkSettingProvided(key=list("groups",1,"value_col"),
                                      settings=testSettings)
```

---

compare_cols *Compares contents of 2 vectors*

Description

Function to compare contents of 2 vectors - used to summarize of which data columns are found in a given standard. Used in detectStandard() and validateSettings()

Usage

```
compare_cols(data_cols, standard_cols)
```

Arguments

data_cols	A character vector with column names in the data frame
standard_cols	A character vector with column names in the data standard

Value

A list summarizing the comparison between data_cols and standard_cols. List has character vectors for "matched_columns", "extra_columns" and "missing_columns" parameters, and a boolean "match" parameter indicating that there are no missing columns.

Examples

```
#match == FALSE
safetyGraphics:::compare_cols(data_cols=c("a","b","c"),
                              standard_cols=c("d","e","f"))

# match == TRUE
safetyGraphics:::compare_cols(names(adlbc),
                              safetyGraphics:::getRequiredColumns(standard="ADaM"))
```

`detectStandard` *Detect the data standard used for a data set*

Description

This function attempts to detect the data CDISC clinical standard used in a given R data frame.

Usage

```
detectStandard(data, domain = "labs")
```

Arguments

`data`	A data frame in which to detect the data standard
`domain`	The data domain for the data set provided. Default: `"labs"`.

Details

This function compares the columns in the provided `"data"` with the required columns for a given data standard/domain combination. The function is designed to work with the SDTM and AdAM CDISC(<https://www.cdisc.org/>) standards for clinical trial data. Currently, only "labs" is the only domain supported.

Value

A list containing the matching `"standard"` ("ADaM", "SDTM" or "None") and a list of `"details"` describing each standard considered. #'

Examples

```
detectStandard(adlbc)[["standard"]] #AdAM
detectStandard(iris)[["standard"]] #none

## Not run:
  detectStandard(adlbc,domain="AE") #throws error. AE domain not supported in this release.

## End(Not run)
```

eDISH *Create an eDISH widget*

Description

This function creates an interactive graphic for the Evaluation of Drug-Induced Serious Hepatotoxicity (eDISH)

Usage

```
eDISH(data, id_col = "USUBJID", value_col = "STRESN",
  measure_col = "TEST", normal_col_low = "STNRLO",
  normal_col_high = "STNRHI", visit_col = "VISIT",
  visitn_col = "VISITN", studyday_col = "DY", baseline = NULL,
  filters = NULL, group_cols = NULL, analysisFlag = NULL,
  measure_values = list(ALT = "Aminotransferase, alanine (ALT)", AST =
  "Aminotransferase, aspartate (AST)", TB = "Total Bilirubin", ALP =
  "Alkaline phosphatase (ALP)"), x_options = c("ALT", "AST", "ALP"),
  y_options = "TB", visit_window = 30, r_ratio_filter = TRUE,
  r_ratio_cut = 0, showTitle = TRUE, debug_js = FALSE,
  warningText = NULL, settings = NULL)
```

Arguments

data	A data frame containing the labs data. Data must be structured as one record per study participant per time point per lab measure.
id_col	Unique subject identifier variable name. Default: "USUBJID".
value_col	Lab result variable name. Default: "STRESN".
measure_col	Lab measure variable name. Default: "TEST".
normal_col_low	Lower limit of normal variable name. Default: "STNRLO".
normal_col_high	Upper limit of normal variable name. Default: "STNRHI".
visit_col	Visit variable name. Default: "VISIT".
visitn_col	Visit number variable name. Default: "VISITN".
studyday_col	Visit day variable name. Default: "DY".
baseline	An optional list defining which column "value_col" and values (one or more) values represent the baseline visit(s) of the study.
filters	An optional list of specifications for filters. Each filter is a nested, named list (containing the filter value column: "value_col" and associated label: "label") within the larger list. Default: NULL.
group_cols	An optional list of specifications for grouping columns. Each group column is a nested, named list (containing the group variable column: "value_col" and associated label: "label") within the larger list. Default: NULL.

analysisFlag	An optional list defining which column value_col and values values should be used to records for use in eDish and mDish analyses. Default: NULL.
measure_values	A list defining the data values from measure_col for the lab measures used in eDish evaluations. Default: list(ALT = 'Aminotransferase, alanine (ALT)',AST = 'Aminotransf
x_options	Specifies variable options for the x-axis using the key values from measure_values (e.g. "ALT"). When multiple options are specified, a control allowing the user to interactively change the x variable is shown. Default: c("ALT", "AST", "ALP").
y_options	Specifies variable options for the y-axis using the key values from measure_values (e.g. "TB"). When multiple options are specified, a control allowing the user to interactively change the y variable is shown. Default: c("TB", "ALP").
visit_window	Default visit window used to highlight eDish points where x and y measures occurred within the specified number of days. Editable by user after render. Default: 30.
r_ratio_filter	Specifies whether the R Ratio filter should be shown. R ratio is defined as: (ALT value/ULN for ALT) / (ALP value/ULN for ALP). Default: TRUE.
r_ratio_cut	Default cut point for R Ratio filter. Ignored when r_ratio_filter = FALSE. User can update this setting via the UI when r_ratio_filter = TRUE. Default: 0.
showTitle	Specifies whether the title should be drawn above the controls. Default: TRUE.
debug_js	print settings in javascript before rendering chart. Default: FALSE.
warningText	Informational text to be displayed near the top of the controls (beneath the title, if any). No warning is displayed if warningText = "". If warningText = NULL, default warning text will be displayed ("Caution: This interactive graphic is not validated. Any clinical recommendations based on this tool should be confirmed using your organizations standard operating procedures.").
settings	Optional list of settings arguments to be converted to JSON using jsonlite::toJSON(settings, auto_ If provided, all other function parameters are ignored. Default: NULL.

Examples

```
## Not run:

## Create eDISH figure customized to user data
eDISH(data=adlbc,
      id_col = "USUBJID",
      value_col = "AVAL",
      measure_col = "PARAM",
      visit_col = "VISIT",
      visitn_col = "VISITNUM",
      studyday_col = "ADY",
      normal_col_low = "A1LO",
      normal_col_high = "A1HI",
      measure_values = list(ALT = "Alanine Aminotransferase (U/L)",
                            AST = "Aspartate Aminotransferase (U/L)",
                            TB = "Bilirubin (umol/L)",
                            ALP = "Alkaline Phosphatase (U/L)"))
```

```
## Create eDISH figure using a premade settings list
group_cols_list <- list(
  list(value_col = "TRTP", label = "Treatment"),
  list(value_col = "SEX", label = "Sex"),
  list(value_col = "AGEGR1", label = "Age group")
)


filters_list <- list(
  list(value_col = "TRTA", label = "Treatment"),
  list(value_col = "SEX", label = "Sex"),
  list(value_col = "RACE", label = "RACE"),
  list(value_col = "AGEGR1", label = "Age group")
)

settingsl <- list(id_col = "USUBJID",
      value_col = "AVAL",
      measure_col = "PARAM",
      visit_col = "VISIT",
      visitn_col = "VISITNUM",
      studyday_col = "ADY",
      normal_col_low = "A1LO",
      normal_col_high = "A1HI",
      group_cols = group_cols_list,
      filters = filters_list,
      measure_values = list(ALT = "Alanine Aminotransferase (U/L)",
                            AST = "Aspartate Aminotransferase (U/L)",
                            TB = "Bilirubin (umol/L)",
                            ALP = "Alkaline Phosphatase (U/L)"))
eDISH(data=adlbc, settings = settingsl)


## End(Not run)
```

`eDISH-shiny` *Shiny bindings for eDISH*

Description

Output and render functions for using eDISH within Shiny applications and interactive Rmd documents.

Usage

```
eDISHOutput(outputId, width = "100%", height = "400px")

renderEDISH(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a eDISH
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

---

generateSettings *Generate a settings object based on a data standard*

Description

This function returns a settings object for the eDish chart based on the specified data standard.

Usage

```
generateSettings(standard = "None", chart = "eDish", partial = FALSE,
  partial_cols = NULL)
```

Arguments

standard	The data standard for which to create settings. Valid options are "SDTM", "AdAM" or "None". Default: "SDTM"
chart	The chart for which standards should be generated ("eDish" only for now) . Default: "eDish".
partial	Boolean for whether or not the standard is a partial standard. Default: "NULL".
partial_cols	Optional character vector of the matched cols if partial is TRUE. It will not be used if partial is FALSE Default: "NULL".

Details

The function is designed to work with the SDTM and AdAM CDISC(<https://www.cdisc.org/>) standards for clinical trial data. Currently, eDish is the only chart supported.

Value

A list containing the appropriate settings for the selected chart

Examples

```
generateSettings(standard="SDTM")
generateSettings(standard="SdTm") #also ok
generateSettings(standard="SDTM", partial=TRUE, partial_cols = c("USUBJID","TEST","STRESN"))
generateSettings(standard="AdAM")

generateSettings(standard="a different standard")
#returns shell settings list with no data mapping

## Not run:
generateSettings(standard="adam",chart="AEExplorer") #Throws error. Only eDish supported so far.

## End(Not run)
```

`getRequiredColumns` *Get a list of required columns*

Description

Get a list of required columns for a chart in a given data standard

Usage

```
getRequiredColumns(standard, chart = "eDish")
```

Arguments

`standard`	The data standard for which to fetch required columns Valid options are "SDTM", "AdAM".
`chart`	The chart for which standards should be generated ("eDish" only for now) . Default: `"eDish"`.

Value

A character vector of required data columns

Examples

```
safetyGraphics:::getRequiredColumns(standard="ADAM")
safetyGraphics:::getRequiredColumns(standard="SDTM")
```

getRequiredSettings *Get a list of required settings*

Description

Get a list of required settings for a given chart

Usage

```
getRequiredSettings(chart = "eDish",
  metadata = safetyGraphics::settingsMetadata)
```

Arguments

chart	The chart for which required settings should be returned ("eDish" only for now) . Default: "eDish".
metadata	The metadata file to be used

Value

List of lists specifying the position of matching named elements in the format `list("filters",2,"value_col")`, which would correspond to `settings[["filters"]][[2]][["value_col"]]`.

Examples

```
safetyGraphics:::getRequiredSettings(chart="eDish")
```

getSettingKeys *Get setting keys matching a pattern*

Description

Recursive function to find all keys matching a given text pattern in a settings list

Usage

```
getSettingKeys(patterns, settings, parents = NULL, matchLists = FALSE)
```

Arguments

patterns	List of text patterns to match with named elements in settings.
settings	List of settings used to generate a chart like eDISH().
parents	List containing the position of the parent list using recursive matches.
matchLists	Boolean indicating whether keys containing lists should be returned as matches.

Details

This function loops through all named elements (or "keys") in a `settings` list and returns items that match `patterns`. If `matchLists==FALSE` (the default), the function iteratively looks through the named elements in nested lists using the built-in `parents` parameter. The function returns a array of keys for all matches using a list of lists. Each key is defines the position of a matching key using an unnamed list. For example, `list("filters",2,"value_col")` would correspond to `settings[["filters"]][[2]][["value_col"]]`.

Value

List of lists specifying the position of matching named elements in the format `list("filters",2,"value_col")`, which would correspond to `settings[["filters"]][[2]][["value_col"]]`.

Examples

```
testSettings<-generateSettings(standard="AdAM")

# returns list of all matching values
safetyGraphics:::getSettingKeys(patterns=c("_col"),
                                settings=testSettings)

#finds the matching nested setting
safetyGraphics:::getSettingKeys(patterns=c("ALP"),
                                settings=testSettings)

#returns an empty list, since the only matching item is a list
safetyGraphics:::getSettingKeys(patterns=c("measure_values"),
                                settings=testSettings)

#finds the matching key associated with a list
safetyGraphics:::getSettingKeys(patterns=c("measure_values"),
                                settings=testSettings,
                                matchLists=TRUE)
```

---

getSettingsMetadata *Get metadata about chart settings*

Description

Retrieve specified metadata about chart settings from the data/settingsMetadata.Rda file.

Usage

```
getSettingsMetadata(charts = NULL, text_keys = NULL, cols = NULL,
  filter_expr = NULL, metadata = safetyGraphics::settingsMetadata)
```

Arguments

charts	optional vector of chart names used to filter the metadata. Exact matches only (case-insensitive). All rows returned by default.
text_keys	optional vector of keys used to filter the metadata. Partial matches for any of the strings are returned (case-insensitive). All rows returned by default.
cols	optional vector of columns to return from the metadata. All columns returned by default.
filter_expr	optional filter expression used to subset the data.
metadata	metadata data frame to be queried

Value

dataframe with the requested metadata or single metadata value

Examples

```
safetyGraphics:::getSettingsMetadata()
# Returns a full copy of settingsMetadata.Rda

safetyGraphics:::getSettingsMetadata(text_keys=c("id_col"))
# returns a dataframe with a single row with metadata for the id_col setting

safetyGraphics:::getSettingsMetadata(text_keys=c("id_col"), cols=c("label"))
# returns the character value for the specified row.
```

---

getSettingValue *Retrieve the value for a given named parameter*

Description

Returns the value for a named parameter (key) in a list settings

Usage

```
getSettingValue(key, settings)
```

Arguments

key	a list (like those provided by getSettingKeys()) defining the position of parameter in the settings object.
settings	The settings list used to generate a chart like eDISH()

Value

the value of the key/settings combo

Examples

```
safetyGraphics:::getSettingValue(list("a","b"),list(a=list(b="myValue"))) #returns "myValue"

testSettings<-generateSettings(standard="AdAM")
safetyGraphics:::getSettingValue(list("id_col"),testSettings)
safetyGraphics:::getSettingValue(list("measure_values","ALP"),testSettings)
safetyGraphics:::getSettingValue(list("NotASetting"),testSettings) #returns NULL
```

---

safetyGraphicsApp *Run the interactive safety graphics builder*

Description

Run the interactive safety graphics builder

Usage

```
safetyGraphicsApp()
```

---

settingsMetadata *Settings Metadata*

Description

Metadata about the settings used to configure safetyGraphics charts. One record per unique setting

Usage

```
settingsMetadata
```

Format

A data frame with 25 rows and 10 columns

chart_edish Flag indicating if the settings apply to the eDish Chart

text_key Text key indicating the setting name. '–' delimiter indicates a nested setting

label Label

description Description

setting_type Expected type for setting value. Should be "character", "vector", "numeric" or "logical"

setting_required Flag indicating if the setting is required

column_mapping Flag indicating if the setting corresponds to a column in the associated data

column_type Expected type for the data column values. Should be "character","logical" or "numeric"

column_required Flag indicating whether the associated data column should be considered required

field_mapping Flag indicating whether the setting corresponds to a field-level mapping in the data

adam Settings values for the ADaM standard

sdtm Settings values for the SDTM standard

Source

Created for this package

`textKeysToList` *Helper function to convert keys from text to nested lists*

Description

Convert settings keys from text vectors (using the "–" delimiter) to a list of lists

Usage

```
textKeysToList(textKeys)
```

Arguments

`textKeys`	a list (or vector) of character keys using the "–" delimiter to indicate hierarchy

Value

A list of unnamed lists, with position in the nested list indicating hierarchy

Examples

```
safetyGraphics:::textKeysToList("id_col")
#list(list("id_col"))

#list(list("id_col"),list("measure_col","label"))
safetyGraphics:::textKeysToList(c("id_col","measure_col--label"))
```

validateSettings *Compare a settings object with a specified data set*

Description

This function returns a list describing the validation status of a data set for a specified data standard

Usage

```
validateSettings(data, settings, chart = "eDish")
```

Arguments

data	A data frame to check against the settings object
settings	The settings list to compare with the data frame.
chart	The chart type being created ("eDish" only for now)

Details

This function returns a list describing the validation status of a settings/data combo for a given chart type. This list can be used to populate status fields and control workflow in the Shiny app. It could also be used to manually QC a buggy chart. The tool checks that all setting properties containing "_col" match columns in the data set via checkColumnSettings, and all properties containing "_values" match fields in the data set via checkFieldSettings.

Value

A list describing the validation state for the data/settings combination. The returned list has the following properties: - ‘valid‘ - boolean indicating whether the settings/data combo creates a valid chart - ‘status‘ - a string summarizing of the validation results - ‘checkList‘ - list of lists giving details about checks performed on individual setting specifications. Each embedded item has the following properties: - ‘key‘ - a list specifying the position of the property being checked. For example, ‘list("group_cols",1,"value_col")‘ corresponds to ‘settings[["group_cols"]][[1]][["value_col"]]‘ - ‘text_key‘ - list from ‘key‘ parsed to character with a "–" separator. - ‘value‘ - value of the setting - ‘check‘ - description of the check performed. - ‘valid‘ - a boolean indicating whether the check was passed - ‘message‘ - a string describing failed checks (where ‘valid=FALSE‘). returns an empty string when ‘valid==TRUE‘

@examples testSettings <- generateSettings(standard="adam") validateSettings(data=adlbc, settings=testSettings)
.$valid is TRUE testSettings$id_col <- "NotAColumn" validateSettings(data=adlbc, settings=testSettings)
.$valid is now FALSE

Index