Package ‘safetyGraphics’

December 14, 2022

**Title**  Interactive Graphics for Monitoring Clinical Trial Safety

**Version** 2.1.1

**Maintainer** Jeremy Wildfire <jwildfire@gmail.com>

**Description** A framework for evaluation of clinical trial safety. Users can interactively explore their data using the included 'Shiny' application.


**Depends** R (>= 4.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Suggests** ggplot2 (>= 3.3.0), knitr (>= 1.34), rmarkdown (>= 2.10), shinydashboard (>= 0.7.1), shinytest (>= 1.5.0), testthat (>= 3.0.4), usethis (>= 2.0.1), listviewer (>= 3.0.0), shinybusy (>= 0.2.2), shinyWidgets (>= 0.6.1)

**Imports** dplyr (>= 1.0.0), DT (>= 0.19), datamods (>= 1.1.5), htmlwidgets (>= 1.5.0), jsonlite (>= 1.7.0), magrittr (>= 2.0.0), purrr (>= 0.3.0), rclipboard (>= 0.1.3), rlang (>= 0.4.11), safetyData (>= 1.0.0), safetyCharts (>= 0.3), shiny (>= 1.6.0), shinyjs (>= 2.0.0), sortable (>= 0.4.4), stringr (>= 1.4.0), tidyR (>= 1.2.0), yaml (>= 2.2.1)

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Jeremy Wildfire [cre, aut], Becca Krouse [aut], Preston Burns [aut], Xiao Ni [aut], James Buchanan [aut], Susan Duke [aut]

**Repository** CRAN

**Date/Publication** 2022-12-14 22:00:02 UTC
R topics documented:

app_startup .................................................. 3
chartsNav ....................................................... 4
chartsTab ......................................................... 4
chartsTabUI ..................................................... 5
detectStandard .................................................. 5
evaluateStandard ................................................ 6
filterTab ........................................................... 7
filterTabChecks .................................................. 8
filterTabUI ......................................................... 8
generateMappingList ......................................... 9
homeTab ........................................................... 9
homeTabUI ........................................................ 10
loadCharts ......................................................... 10
loadChartsUI ..................................................... 11
loadData .......................................................... 11
loadDataUI ........................................................ 12
makeChartConfig ................................................ 12
makeChartExport ............................................... 13
makeChartParams ............................................... 14
makeChartSummary .............................................. 14
makeMapping ..................................................... 15
makeMeta ........................................................... 15
mappingColumn .................................................. 16
mappingColumnUI ............................................... 17
mappingDomain .................................................. 17
mappingDomainUI ............................................... 18
mappingSelect .................................................... 18
mappingSelectUI ................................................ 19
mappingTab ........................................................ 19
mappingTabUI ................................................... 20
prepareChart ...................................................... 20
safetyGraphicsApp ............................................... 21
safetyGraphicsInit ............................................... 22
safetyGraphicsServer .......................................... 22
safetyGraphicsUI .................................................. 23
settingsCharts ................................................... 23
settingsChartsUI ............................................... 24
settingsCode ..................................................... 24
settingsCodeUI .................................................. 25
settingsData ...................................................... 25
settingsDataUI ................................................... 26
settingsMapping .................................................. 26
settingsMappingUI .............................................. 27
settingsTab ....................................................... 27
settingsTabUI ..................................................... 28

Index 29
Description

Prepare inputs for safetyGraphics app - run before app is initialized.

Usage

```r
app_startup(
    domainData = NULL,
    meta = NULL,
    charts = NULL,
    mapping = NULL,
    autoMapping = NULL,
    filterDomain = NULL,
    chartSettingsPaths = NULL
)
```

Arguments

domainData: named list of data.frames to be loaded in to the app. Sample AdAM data from the safetyData package used by default.

meta: data frame containing the metadata for use in the app. If no metadata is provided (default value is NULL), metadata is generated by `makeMeta()`.

charts: list of charts in the format produced by `safetyGraphics::makeChartConfig()`.

mapping: list specifying the initial mapping values for each data mapping for each domain (e.g. `list(aes=list(id_col='USUBJID', seq_col='AESEQ'))`).

autoMapping: boolean indicating whether the app should attempt to automatically detect data standards and generate mappings for the data provided. Values specified in the `mapping` parameter overwrite automatically generated mappings when both are found. Defaults to true.

filterDomain: domain used for the data/filter tab. Demographics ("dm") is used by default. Using a domain that is not one record per participant is not recommended.

chartSettingsPaths: path(s) where customization functions are saved relative to your working directory. All charts can have initialization (e.g. `myChart_Init.R`) and static charts can have charting functions (e.g. `myGraphic_Chart.R`). All R files in this folder are sourced and files with the correct naming convention are linked to the chart. See the Custom Charts vignette for more details.

Value

List of elements for used to initialize the shiny app with the following parameters:

- "meta" List of configuration metadata
"charts" List of charts
"domainData" List of domain level data sets
"mapping" Initial Data Mapping
"standards" List of domain level data standards

**chartsNav**  
*Adds a navbar tab that initializes the Chart Module UI*

**Description**  
Adds a navbar tab that initializes the Chart Module UI

**Usage**  
`chartsNav(chart, ns)`

**Arguments**
- **chart**: chart metadata
- **ns**: namespace

**chartsTab**  
*Server for chart module, designed to be re-used for each chart generated.*

**Description**  
Server for chart module, designed to be re-used for each chart generated.

**Usage**  
`chartsTab(input, output, session, chart, data, mapping)`

**Arguments**
- **input**: Input objects from module namespace
- **output**: Output objects from module namespace
- **session**: An environment that can be used to access information and functionality relating to the session
- **chart**: list containing a safetyGraphics chart object like those returned by `makeChartConfig`
- **data**: named list of current data sets (Reactive)
- **mapping**: tibble capturing the current data mappings (Reactive)
chartsTabUI

UI for chart module, designed to be re-used for each chart generated.

Description
UI for chart module, designed to be re-used for each chart generated.

Usage
chartsTabUI(id, chart)

Arguments
id module id
chart list containing chart specifications like those returned by makeChartConfig.

detectStandard
Detect the data standard used for a data set

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

Usage
detectStandard(data, domain = NULL, meta = NULL)

Arguments
data A data frame in which to detect the data standard - required.
domain the domain to evaluate - should match a value of meta$domain. Uses the first value in meta$domain if no value is provided.
meta the metadata containing the data standards.

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 by default. Additional standards can be added by modifying the "meta" data set included as part of this package.

Value
A data frame describing the detected standard for each "text_key" in the provided metadata. Columns are "domain", "text_key", "column" and "standard".
Evaluate a data set against a data standard

Description

Determines whether the required data elements in a data standard are found in a given data frame.

Usage

evaluateStandard(data, meta, domain, standard)

Arguments

data A data frame in which to detect the data standard
meta the metadata containing the data standards.
domain the domain to evaluate - should match a value of meta$domain
standard standard to evaluate

Value

a list describing to what degree the data set matches the data standard. The "match" property describes compliance with the standard as "full", "partial" or "none". The "checks" property is a list of the data elements expected for the standard and whether they are "valid" in the given data set. "total_checks", "valid_checks" and "invalid_checks" provide counts of the specified checks. "match_percent" is calculated as valid_checks/total_checks. "mapping" is a data frame describing the detected standard for each "text_key" in the provided metadata. Columns are "text_key", "current" containing the name of the matched column or field value in the data and "match" a boolean indicating whether the data matches the standard.

Examples

# Match is TRUE
evaluateStandard(
  data=safetyData::adam_adlbc,
  meta=safetyCharts::meta_labs,
  domain="labs",
  standard="adam"
)

# Match is FALSE
evaluateStandard(
  data=safetyData::adam_adlbc,
filterTab

meta=safetyCharts::meta_labs,
domain="labs",
standard="sdm"
)

filterTab  

Server for the filter module in datamods::filter_data_ui

Description

Server for the filter module in datamods::filter_data_ui

Usage

filterTab(
  input,
  output,
  session,
  domainData,
  filterDomain,
  current_mapping,
  tabID = "Filtering",
  filterVars = NULL
)

Arguments

input  Shiny input object
output Shiny output object
session Shiny session object
domainData list of data files for each domain
filterDomain domain to use for filtering (typically "dm")
current_mapping current data mapping
tabID  ID for the tab containing the filter UI (used for testing)
filterVars Variables to use for filtering (used for testing)

Value

filtered data set
filterTabChecks  
*Checks for whether the current data and settings support a filter tab*

**Description**
Checks for whether the current data and settings support a filter tab

**Usage**

```
filterTabChecks(domainData, filterDomain, current_mapping)
```

**Arguments**
- **domainData**: list of data files for each domain
- **filterDomain**: domain to use for filtering (typically "dm")
- **current_mapping**: current data mapping (REACTIVE)

**Value**
reactive that returns a boolean indicating whether the checks passed and filtering can be initialized

filterTabUI  
*UI for the filter module in datamods::filter_data_ui*

**Description**
UI for the filter module in datamods::filter_data_ui

**Usage**

```
filterTabUI(id)
```

**Arguments**
- **id**: module id
generateMappingList  

Convert mapping data.frame to a list

Description

Convert mapping data.frame to a list

Usage

generateMappingList(settingsDF, domain = NULL, pull = FALSE)

Arguments

settingsDF  data frame containing current mapping
domain  mapping domain to return (returns all domains as a named list by default)
pull  call pull() the value for each parameter - needed for testing only. default: FALSE

homeTab  

Server for the filter module in datamods::filter_data_ui

Description

Server for the filter module in datamods::filter_data_ui

Usage

homeTab(input, output, session)

Arguments

input  mod input
output  mod output
session  mod session
homeTabUI  
__UI for the home module__

**Description**

UI for the home module

**Usage**

```r
homeTabUI(id)
```

**Arguments**

- `id`  
  module id

---

loadCharts  
__Server for the chart loading module used in safetyGraphicsInit()__

**Description**

Server for the chart loading module used in safetyGraphicsInit()

**Usage**

```r
loadCharts(input, output, session, charts = makeChartConfig())
```

**Arguments**

- `input`  
  Shiny input object
- `output`  
  Shiny output object
- `session`  
  Shiny session object
- `charts`  
  list containing chart specifications like those returned by `makeChartConfig`.  


**loadChartsUI**

*UI for the chart loading module used in safetyGraphicsInit()*

**Description**

UI for the chart loading module used in safetyGraphicsInit()

**Usage**

```r
loadChartsUI(id, charts = makeChartConfig())
```

**Arguments**

- `id` module id
- `charts` list containing chart specifications like those returned by `makeChartConfig`.

---

**loadData**

*Server for the data loading module used in safetyGraphicsInit()*

**Description**

Server for the data loading module used in safetyGraphicsInit()

**Usage**

```r
loadData(input, output, session, domain)
```

**Arguments**

- `input` Shiny input object
- `output` Shiny output object
- `session` Shiny session object
- `domain` data domain to be loaded
**makeChartConfig**

**Description**

Converts YAML chart configuration files to an R list and binds workflow functions. See the vignette about creating custom charts for more details.

**Usage**

```r
makeChartConfig(
  dirs,  
  packages = "safetyCharts",  
  packageLocation = "config",  
  sourceFiles = FALSE
)
```

**Arguments**

- **dirs**  
  path to one or more directories containing yaml config files (relative to working directory)
- **packages**  
  installed packages names containing yaml config files in the /inst/packageLocation folder
- **packageLocation**  
  inst folder where yaml config files (and possibly R functions referenced in yaml workflow) are located in packages
- **sourceFiles**  
  boolean indicating whether to source all R files found in dirs.

---

**loadDataUI**

*UI for the data loading module used in safetyGraphicsInit()*

**Description**

UI for the data loading module used in safetyGraphicsInit()

**Usage**

```r
loadDataUI(id, domain = NULL)
```

**Arguments**

- **id**  
  module id
- **domain**  
  character vector with domains to be loaded
**Value**

returns a named list of charts derived from YAML files. Each element of the list contains information about a single chart, and has the following parameters:

- "env" Environment for the chart. Must be set to "safetyGraphics" or the chart is dropped.
- "name" Name of the chart. Also the name of the element in the list - e.g. charts$aeExplorer$name is "aeExplorer"
- "label" Short description of the chart
- "type" Type of chart; options are: 'htmlwidget', 'module', 'plot', 'table', 'html' or 'plotly'.
- "domain" Data domain. Should correspond to one or more domains in meta
- "package" Primary package (if any). Other packages can be loaded directly in workflow functions.
- "order" Integer order in which to display the chart. If order is a negative number, the chart is dropped.
- "export" Logical flag indicating whether the chart can be exported to an html report. True by default, except for when type is module.
- "path" Path to YAML file
- "links" Named list of link names/urls to be shown in the chart header.
- "workflow" List of functions names used to render chart. See vignette for details.
- "functions" List of functions for use in chart rendering. These functions must be located in the global environment or package field of the YAML config. Function names must include either the name or workflow fields of the YAML config.

---

**makeChartExport**

**Make Chart Export**

**Description**

Creates R code that allows chart to be exported

**Usage**

makeChartExport(chart, mapping)

**Arguments**

- **chart** chart object like the one generated by makeChartConfig().
- **mapping** mapping object like the one generated by makeMapping().

**Value**

returns a character vector that can be saved as R code.
**makeChartParams**

*Make Chart Parameters*

**Description**

Updates raw data and mapping for use with a specific chart

**Usage**

```
makeChartParams(data, chart, mapping)
```

**Arguments**

- **data** list of domain-level data
- **chart** list containing chart specifications
- **mapping** data frame with current mapping

**makeChartSummary**

*html chart summary*

**Description**

makes a nicely formatted html summary for a chart object

**Usage**

```
makeChartSummary(chart, showLinks = TRUE, class = "chart-header")
```

**Arguments**

- **chart** list containing chart specifications
- **showLinks** boolean indicating whether to include links
- **class** character to include as class
**makeMapping**

Create data mapping based on data standards and user input

**Description**

Create data mapping based on data standards and user input

**Usage**

```r
makeMapping(domainData, meta, autoMapping, customMapping)
```

**Arguments**

- `domainData`: named list of data.frames to be loaded in to the app. Sample AdAM data from the safetyData package used by default.
- `meta`: data frame containing the metadata for use in the app.
- `autoMapping`: boolean indicating whether the app should use `safetyGraphics::detectStandard()` to detect data standards and automatically generate mappings for the data provided. Values specified in the `customMapping` parameter overwrite auto-generated mappings when both are found. Defaults to true.
- `customMapping`: optional list specifying initial mapping values within each data mapping (e.g. `list(aes=list(id_col='USUBJID', seq_col='AESEQ'))`).

**Value**

List containing data standard information and mapping

- "mapping" Initial Data Mapping
- "standards" List of domain level data standards (or NULL if autoMapping is false)

**makeMeta**

Create a metadata object table for a set of charts

**Description**

Generates metadata object for a list of charts. `makeMeta()` looks for metadata in 3 locations for each chart object:

- Domain-level metadata saved as `meta_chart$name` in the chart$package namespace
- Chart-specific metadata saved as `meta_chart$domain` in the chart$package namespace
- Chart-specific metadata saved directly to the chart object as chart$meta

After checking all charts, all metadata files are stacked in to a single dataframe and returned. If duplicate metadata rows (domain + text_key) are found, an error is thrown.
mappingColumn

Usage

makeMeta(charts)

Arguments

charts list of safetyGraphics chart objects for which to create metadata

Value
tibble of metadata with the following columns:

domain Data domain
text_key Text key indicating the setting name. '---' delimiter indicates a field level data mapping
col_key Key for the column mapping
field_key Key for the field mapping (if any)
type type of mapping - "field" or "column"
label Label
description Description
multiple Mapping supports multiple columns/fields
standard_adam Default values for the ADaM data standard
standard_sdtm Default values for the SDTM data standard

mappingColumn Server that facilitates the mapping of a column data (and any associated fields)

Description

Server that facilitates the mapping of a column data (and any associated fields)

Usage

mappingColumn(input, output, session, meta, data)

Arguments

input Shiny input object
output Shiny output object
session Shiny session object
meta metadata data frame for the object
data current data file for the domain

Value

A reactive data.frame providing the current value for text_key associated with the selected column
**mappingColumnUI**

*UI that facilitates the mapping of a column data (and any associated fields)*

**Description**

UI that facilitates the mapping of a column data (and any associated fields)

**Usage**

mappingColumnUI(id, meta, data, mapping = NULL)

**Arguments**

- **id**: module id
- **meta**: metadata for the column (and related fields)
- **data**: current data file for the domain
- **mapping**: current data mapping for the column (and related fields)

---

**mappingDomain**

*Server that facilitates the mapping of a full data domain*

**Description**

Server that facilitates the mapping of a full data domain

**Usage**

mappingDomain(input, output, session, meta, data)

**Arguments**

- **input**: Shiny input object
- **output**: Shiny output object
- **session**: Shiny session object
- **meta**: metadata for the domain
- **data**: clinical data for the domain

**Value**

A reactive data frame containing the mapping for the domain
**mappingDomainUI**

*UI that facilitates the mapping of a full data domain*

**Description**

UI that facilitates the mapping of a full data domain

**Usage**

```r
mappingDomainUI(id, meta, data, mapping = NULL)
```

**Arguments**

- **id**: module id
- **meta**: metadata for the domain
- **data**: data file for the domain
- **mapping**: current data mapping

**mappingSelect**

*Server that facilitates the mapping of a single data element (column or field) with a simple select UI*

**Description**

Server that facilitates the mapping of a single data element (column or field) with a simple select UI

**Usage**

```r
mappingSelect(input, output, session)
```

**Arguments**

- **input**: Shiny input object
- **output**: Shiny output object
- **session**: Shiny session object

**Value**

A reactive containing the selected column
**mappingSelectUI**

*UI that facilitates the mapping of a single data element (column or field) with a simple select UI*

**Description**

UI that facilitates the mapping of a single data element (column or field) with a simple select UI

**Usage**

```r
mappingSelectUI(id, label, choices = NULL, default = NULL)
```

**Arguments**

- **id**: unique id for the UI
- **label**: label associated with the control
- **choices**: a list of options for the control
- **default**: default value for the control

**Value**

returns the selected value wrapped in a reactive().

---

**mappingTab**

*Server for mapping tab covering of all data domains*

**Description**

Server for mapping tab covering of all data domains

**Usage**

```r
mappingTab(input, output, session, meta, domainData)
```

**Arguments**

- **input**: Shiny input object
- **output**: Shiny output object
- **session**: Shiny session object
- **meta**: metadata for all domains
- **domainData**: clinical data for all domains

**Value**

list of mappings for all domains
mappingTabUI  
*UI for mapping tab covering of all data domains*

**Description**

UI for mapping tab covering of all data domains

**Usage**

```r
mappingTabUI(id, meta, domainData, mappings = NULL, standards = NULL)
```

**Arguments**

- `id`: module id
- `meta`: metadata for all domains
- `domainData`: list of data files for each domain
- `mappings`: optional data frame containing stacked mappings for all domains
- `standards`: optional list of data standards like the ones generated by detectStandard()

**prepareChart**

*Prepare a chart object for safetyGraphics*

**Description**

Sets default values and binds needed functions to a chart object based on chart type.

**Usage**

```r
prepareChart(chart)
```

**Arguments**

- `chart`: chart object like the one generated by makeChartConfig().

**Value**

returns the chart object with a new functions object added.
safetyGraphicsApp

Run the core safetyGraphics App

Description
Run the core safetyGraphics App

Usage
safetyGraphicsApp(
  domainData = list(labs = safetyData::adam_adlbc, aes = safetyData::adam_adae, dm =
    safetyData::adam_adsl),
  meta = NULL,
  charts = NULL,
  mapping = NULL,
  autoMapping = TRUE,
  filterDomain = "dm",
  chartSettingsPaths = NULL,
  runNow = TRUE
)

Arguments

domainData named list of data.frames to be loaded in to the app. Sample AdAM data from the safetyData package used by default

meta data frame containing the metadata for use in the app. If no metadata is provided, metadata is generated by makeMeta().

charts list of charts in the format produced by safetyGraphics::makeChartConfig()

mapping list specifying the initial mapping values for each data mapping for each domain (e.g. list(aes= list(id_col='USUBJID', seq_col='AESEQ')).

autoMapping boolean indicating whether the app should attempt to automatically detect data standards and generate mappings for the data provided. Values specified in the mapping parameter overwrite automatically generated mappings when both are found. Defaults to true.

filterDomain domain used for the data/filter tab. Demographics ("dm") is used by default. Using a domain that is not one record per participant is not recommended.

chartSettingsPaths path(s) where customization functions are saved relative to your working directory. All charts can have initialization (e.g. myChart_Init.R) and static charts can have charting functions (e.g. myGraphic_Chart.R). All R files in this folder are sourced and files with the correct naming convention are linked to the chart. See the Custom Charts vignette for more details.

runNow Should the shiny app object created be run directly? Helpful when writing functions to dispatch to shinyapps, rsconnect, or shinyproxy.
safetyGraphicsInit  
App to select charts, load data and then initialize the core safetyGraphics app

Description
App to select charts, load data and then initialize the core safetyGraphics app

Usage
safetyGraphicsInit(
  charts = makeChartConfig(),
  delayTime = 1000,
  maxSize = NULL
)

Arguments
- charts: chart object
- delayTime: time (in ms) between drawing app UI and starting server. Default set to 1000 (1 second), but could need to be higher on slow machine.
- maxSize: maximum file size in MB allowed for file upload

safetyGraphicsServer  
Server for core safetyGraphics app including Home, Mapping, Filter, Charts and Settings modules.

Description
This function returns a server function suitable for use in shiny::runApp()

Usage
safetyGraphicsServer(
  input,
  output,
  session,
  meta,
  mapping,
  domainData,
  charts,
  filterDomain
)
safetyGraphicsUI

Arguments

- **input**: Shiny input object
- **output**: Shiny output object
- **session**: Shiny session object
- **meta**: data frame containing the metadata for use in the app.
- **mapping**: current mapping
- **domainData**: named list of data.frames to be loaded in to the app.
- **charts**: list of charts to include in the app
- **filterDomain**: domain used for the data/filter tab. Demographics ("dm") is used by default. Using a domain that is not one record per participant is not recommended.

safetyGraphicsUI(id, meta, domainData, mapping, standards)

Description

UI for the core safetyGraphics app including Home, Mapping, Filter, Charts and Settings modules.

Usage

safetyGraphicsUI(id, meta, domainData, mapping, standards)

Arguments

- **id**: module ID
- **meta**: data frame containing the metadata for use in the app.
- **domainData**: named list of data.frames to be loaded in to the app.
- **mapping**: data.frame specifying the initial values for each data mapping. If no mapping is provided, the app will attempt to generate one via detectStandard()
- **standards**: a list of information regarding data standards. Each list item should use the format returned by safetyGraphics::detectStandard.
**settingsCharts**  
*Server for settings tab showing details for the charts loaded in the app*

**Description**  
Server for settings tab showing details for the charts loaded in the app

**Usage**  
`settingsCharts(input, output, session, charts)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Shiny input object</td>
</tr>
<tr>
<td>output</td>
<td>Shiny output object</td>
</tr>
<tr>
<td>session</td>
<td>Shiny session object</td>
</tr>
<tr>
<td>charts</td>
<td>list data frame summarizing the charts</td>
</tr>
</tbody>
</table>

**settingsChartsUI**  
*UI for settings tab showing details for the charts loaded in the app*

**Description**  
UI for settings tab showing details for the charts loaded in the app

**Usage**  
`settingsChartsUI(id)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>module id</td>
</tr>
</tbody>
</table>
### settingsCode

*Server for settings tab providing code to re-start the app with current data/settings*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Shiny input object</td>
</tr>
<tr>
<td>output</td>
<td>Shiny output object</td>
</tr>
<tr>
<td>session</td>
<td>Shiny session object</td>
</tr>
<tr>
<td>mapping</td>
<td>mapping</td>
</tr>
<tr>
<td>charts</td>
<td>charts</td>
</tr>
<tr>
<td>domainData</td>
<td>data list</td>
</tr>
</tbody>
</table>

### Description

Server for settings tab providing code to re-start the app with current data/settings

### Usage

```r
settingsCode(input, output, session, mapping, charts, domainData)
```

### Arguments

- **input**: Shiny input object
- **output**: Shiny output object
- **session**: Shiny session object
- **mapping**: mapping
- **charts**: charts
- **domainData**: data list

---

### settingsCodeUI

*UI for settings tab providing code to re-start the app with current data/settings*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>module ID</td>
</tr>
</tbody>
</table>

### Description

UI for settings tab providing code to re-start the app with current data/settings

### Usage

```r
settingsCodeUI(id)
```

### Arguments

- **id**: module ID
settingsData  

Server for settings tab showing current data

Description

Server for settings tab showing current data

Usage

settingsData(input, output, session, domains)

Arguments

- input: Shiny input object
- output: Shiny output object
- session: Shiny session object
- domains: named list of the data.frames for each domain

settingsDataUI  

UI for settings tab showing current data

Description

UI for settings tab showing current data

Usage

settingsDataUI(id)

Arguments

- id: module id
settingsMapping

Server for settings tab showing current mapping

Description

Server for settings tab showing current mapping

Usage

settingsMapping(input, output, session, metadata, mapping)

Arguments

- input: Shiny input object
- output: Shiny output object
- session: Shiny session object
- metadata: Data mapping metadata used for initial loading of app
- mapping: reactive data frame representing the current metadata mapping. columns = "domain", "text_id" and "current"

settingsMappingUI

UI for settings tab showing current mapping

Description

UI for settings tab showing current mapping

Usage

settingsMappingUI(id)

Arguments

- id: module id
settingsTab

Server for the setting page

Description
Server for the setting page

Usage
settingsTab(input, output, session, domains, metadata, mapping, charts)

Arguments
- input: Shiny input object
- output: Shiny output object
- session: Shiny session object
- domains: domains
- metadata: metadata
- mapping: mapping
- charts: charts

settingsTabUI

UI for the settings tab

Description
UI for the settings tab

Usage
settingsTabUI(id)

Arguments
- id: module ID
Index

app_startup, 3
chartsNav, 4
chartsTab, 4
chartsTabUI, 5
detectStandard, 5
evaluateStandard, 6
filterTab, 7
filterTabChecks, 8
filterTabUI, 8
generateMappingList, 9
homeTab, 9
homeTabUI, 10
loadCharts, 10
loadChartsUI, 11
loadData, 11
loadDataUI, 12
makeChartConfig, 4, 5, 10, 11, 12
makeChartExport, 13
makeChartParams, 14
makeChartSummary, 14
makeMapping, 15
makeMeta, 15
mappingColumn, 16
mappingColumnUI, 17
mappingDomain, 17
mappingDomainUI, 18
mappingSelect, 18
mappingSelectUI, 19
mappingTab, 19
mappingTabUI, 20
prepareChart, 20
safetyGraphicsApp, 21
safetyGraphicsInit, 22
safetyGraphicsServer, 22
safetyGraphicsUI, 23
settingsCharts, 24
settingsChartsUI, 24
settingsCode, 25
settingsCodeUI, 25
settingsData, 26
settingsDataUI, 26
settingsMapping, 27
settingsMappingUI, 27
settingsTab, 28
settingsTabUI, 28