Package 'ruminate'

March 14, 2024

Title A Pharmacometrics Data Transformation and Analysis Tool

Version 0.2.2

Description Exploration of pharmacometrics data involves both general tools (transformation and plotting) and specific techniques (non-compartmental analysis). This kind of exploration is generally accomplished by utilizing different packages. The purpose of 'ruminate' is to create a 'shiny' interface to make these tools more broadly available while creating reproducible results.

License BSD_2_clause + file LICENSE

BugReports https://github.com/john-harrold/ruminate/issues

URL https://ruminate.ubiquity.tools/

Encoding UTF-8

Depends R (>= 4.2.0)

Imports digest, dplyr, DT, flextable, formods (>= 0.1.3), ggplot2, onbrand (>= 1.0.3), PKNCA (>= 0.10.2), plotly, rhandsontable, rlang, shiny, shinyAce, shinyWidgets, rxode2 (>= 2.1.2), stats, stringr, tidyr, utils, yaml

Suggests clipr, gridExtra, knitr, nlmixr2lib, nonmem2rx, prompter, rmarkdown, readxl, rxode2et, shinydashboard, testthat (>= 3.0.0), ubiquity

RoxygenNote 7.3.1

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author John Harrold [aut, cre] (<https://orcid.org/0000-0003-2052-4373>)

Maintainer John Harrold <john.m.harrold@gmail.com>

Repository CRAN

Date/Publication 2024-03-14 03:20:02 UTC

R topics documented:

`apply_route_map` *Applies Route Mapping to Dataset*

Description

Used to convert nonstandard dose route values (i.e. "IV") to standard values ("intravascular").

Usage

```
apply_route_map(route_map = list(), route_col = NULL, DS = NULL)
```

Arguments

route_map	List with names corresponding to the route replacement and a vector of regular expressions to match.
route_col	Column name with the route data.
DS	Dataframe containing the dataset.

Value

Dataset with the route mapping applied.

Examples

```
if(system.file(package="readxl") !=""){
library(readxl)
#loading a dataset
data_file =  system.file(package="formods","test_data","TEST_DATA.xlsx")
myDS = readxl::read_excel(path=data_file, sheet="DATA")

 route_map = list(
   intravascular = c("^(?i)iv$"),
   extravascular = c("^(?i)sc$", "^(?i)oral")
 )

utils::head(myDS[["ROUTE"]])

myDS = apply_route_map(route_map = route_map,
                       route_col = "ROUTE",
                       DS        = myDS)

utils::head(myDS[["ROUTE"]])
}
```

---

dose_records_builder *Builds Dose Records Dataframe*

Description

Takes information about columns in dataset and constructs the dosing records.

Usage

```
dose_records_builder(
  NCA_DS = NULL,
  dose_from = NULL,
  col_id = NULL,
  col_time = NULL,
  col_ntime = NULL,
  col_route = NULL,
```

```
  col_dose = NULL,
  col_cycle = NULL,
  col_dur = NULL,
  col_evid = NULL,
  col_analyte = NULL,
  col_group = NULL
)
```

Arguments

NCA_DS	Dataset containing dosing records.
dose_from	Method of dose extraction either "cols" or "rows".
col_id	Name of column with subject ID.
col_time	Name of column with time since first dose.
col_ntime	Name of column with time since the last dose (required with dose_from="cols").
col_route	Name of column with route information.
col_dose	Name of column with last dose given.
col_cycle	Name of column with dose cycle (required with dose_from="cols").
col_dur	Name of column with dose duration.
col_evid	Name of column with event ID (required with dose_from="rows").
col_analyte	Name of column with analyte (optional).
col_group	Names of columns with grouping information (optionl).

Value

list containing the following elements

- isgood: Return status of the function.
- msgs: Messages to be passed back to the user.
- dose_rec:

Examples

```
if(system.file(package="readxl") !=""){
library(dplyr)
library(readxl)
library(stringr)

# Example data file:
data_file =  system.file(package="formods","test_data","TEST_DATA.xlsx")

# Dataset formatted to extract dosing from columns
DS_cols = readxl::read_excel(path=data_file, sheet="DATA")        |>
 dplyr::filter(EVID == 0)                                         |>
 dplyr::filter(DOSE %in% c(3))                                    |>
 dplyr::filter(str_detect(string=Cohort, "^MD"))                  |>
```

```
 dplyr::filter(CMT == "C_ng_ml")

drb_res = dose_records_builder(
 NCA_DS     = DS_cols,
 dose_from  = "cols",
 col_id     = "ID",
 col_time   = "TIME_DY",
 col_ntime  = "NTIME_DY",
 col_route  = "ROUTE",
 col_cycle  = "DOSE_NUM",
 col_dose   = "DOSE",
 col_group  = "Cohort")

utils::head(drb_res$dose_rec)

# Dataset formatted to extract dosing from rows (records)
DS_rows = readxl::read_excel(path=data_file, sheet="DATA")        |>
 dplyr::filter(DOSE %in% c(3))                                    |>
 dplyr::filter(str_detect(string=Cohort, "^MD"))                  |>
 dplyr::filter(CMT %in% c("Ac", "C_ng_ml"))

drb_res = dose_records_builder(
 NCA_DS     = DS_rows,
 dose_from  = "rows",
 col_id     = "ID",
 col_time   = "TIME_DY",
 col_ntime  = "NTIME_DY",
 col_route  = "ROUTE",
 col_dose   = "AMT",
 col_evid   = "EVID",
 col_group  = "Cohort")

utils::head(drb_res$dose_rec)
}
```

---

mk_figure_ind_obs *Creates Figures of Individual Observations from PKNCA Result*

Description

Takes the output of PKNCA and creates ggplot figures faceted by subject id highlighting of certain NCA aspects (e.g. points used for half-life)

Usage

```
mk_figure_ind_obs(
  nca_res,
  OBS_LAB = "Concentration ===CONCUNITS===",
  TIME_LAB = "Time ===TIMEUNITS===",
  OBS_STRING = "Observation",
```

```
  BLQ_STRING = "BLQ",
  NA_STRING = "Missing",
  log_scale = TRUE,
  scales = "fixed",
  nfrows = 4,
  nfcols = 3
)
```

Arguments

nca_res	Output of PKNCA.
OBS_LAB	Label of the observation axis with optional ===CONCUNITS=== placeholder for units.
TIME_LAB	Label of the time axis with optional ===TIMEUNITS=== placeholder for units.
OBS_STRING	Label for observation data.
BLQ_STRING	Label for BLQ data.
NA_STRING	Label for missing data.
log_scale	Boolean variable to control y-scale (TRUE: Log 10, FALSE: linear).
scales	String to determine the scales used when faceting. Can be either "fixed", "free", "free_x", or "free_y".
nfrows	Number of facet rows per page.
nfcols	Number of facet cols per page.

Value

List containing the element figures which is a list of figure pages ("Figure 1", "Figure 2", etc.). Each of these is a also a list containing two elements:

- gg: A ggplot object for that page.
- notes: Placeholder for future notes, but NULL now.

Examples

```
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"

# We need a state variable to be define
sess_res = NCA_test_mksession(session=list(),
     id     = id,
     id_UD  = id_UD,
     id_DW  = id_DW,
     id_ASM = id_ASM,
     full_session=FALSE)

state = sess_res$state
```

```
# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This is the raw PKNCA output
pknca_res = NCA_fetch_ana_pknca(state, current_ana)

# Building the figure
mk_res = mk_figure_ind_obs(nca_res = pknca_res)
mk_res$figures$Figure_1$gg
```

`mk_table_ind_obs` *Creates Tables of Individual Observations from PKNCA Result*

Description

Takes the output of PKNCA and creates a tabular view of the individual observation data. This can be spread out of over several tables (pages) if necessary.

Usage

```
mk_table_ind_obs(
  nca_res,
  obnd = NULL,
  not_sampled = "NS",
  blq = "BLQ",
  digits = 3,
  text_format = "text",
  max_height = 7,
  max_width = 6.5,
  max_row = NULL,
  max_col = 9,
  notes_detect = NULL,
  rows_by = "time"
)
```

Arguments

`nca_res`	Output of PKNCA.
`obnd`	onbrand reporting object.
`not_sampled`	Character string to use for missing data when pivoting.
`blq`	Character string to use to indicate data below the level of quantification (value of 0 in the dataset).
`digits`	Number of significant figures to report (set to `NULL` to disable rounding)
`text_format`	Either `"md"` for markdown or `"text"` (default) for plain text.
`max_height`	Maximum height of the final table in inches (A value of `NULL` will use 100 inches).

max_width	Maximum width of the final table in inches (A value of NULL will use 100 inches).
max_row	Maximum number of rows to have on a page. Spillover will hang over the side of the page..
max_col	Maximum number of columns to have on a page. Spillover will be wrapped to multiple pages.
notes_detect	Vector of strings to detect in output tables (example c("NC", "BLQ")).
rows_by	Can be either "time" or "id". If it is "time", there will be a column for time and separate column for each subject ID. If rows_by is set to "id" there will be a column for ID and a column for each individual time.

Value

List containing the following elements

- isgood: Boolean indicating the exit status of the function.
- one_table: Dataframe of the entire table with the first lines containing the header.
- one_body: Dataframe of the entire table (data only).
- one_header: Dataframe of the entire header (row and body, no data).
- tables: Named list of tables. Each list element is of the output
- msgs: Vector of text messages describing any errors that were found. format from build_span.

Examples

```
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"

# We need a state variable to be define
sess_res = NCA_test_mksession(session=list(),
     id     = id,
     id_UD  = id_UD,
     id_DW  = id_DW,
     id_ASM = id_ASM,
     full_session=FALSE)

state = sess_res$state

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This is the raw PKNCA output
pknca_res = NCA_fetch_ana_pknca(state, current_ana)

# Building the figure
mk_res = mk_table_ind_obs(nca_res = pknca_res)
mk_res$tables[["Table 1"]]$ft
```

`mk_table_nca_params` *Create Tabular Output from PKNCA Results*

Description

Create paginated tables from PKNCA to use in reports and Shiny apps.

Usage

```
mk_table_nca_params(
  nca_res,
  type = "individual",
  grouping = "interval",
  not_calc = "NC",
  obnd = NULL,
  nps = NULL,
  mult_str = "*",
  infinity = "inf",
  digits = NULL,
  text_format = "text",
  notes_detect = NULL,
  max_height = 7,
  max_width = 6.5,
  max_row = NULL,
  max_col = NULL
)
```

Arguments

`nca_res`	Output of PKNCA.
`type`	Type of table to generate. Can be either `"individual"` or `"summary"]`.
`grouping`	How to group columns in tables. Can be either `"interval"` or `"parameter"]`.
`not_calc`	Text string to replace NA values with to indicated values were not calculated.
`obnd`	onbrand reporting object.
`nps`	NCA parameter summary table with the following columns. • parameter: PKNCA Paramter name. • text: Name used in text output. • md: Name used markdown output. • latex: Name used in latex output. • description: Verbose textual description of the parameter.
`mult_str`	Text string to replace * values in units.
`infinity`	Text string to replace infinity in time intervals in column headers.
`digits`	Number of significant figures to report (set to `NULL` to disable rounding)

text_format	Either "md" for markdown or "text" (default) for plain text.
notes_detect	Vector of strings to detect in output tables (example c("NC", "BLQ")).
max_height	Maximum height of the final table in inches (A value of NULL will use 100 inches).
max_width	Maximum width of the final table in inches (A value of NULL will use 100 inches).
max_row	Maximum number of rows to have on a page. Spillover will hang over the side of the page..
max_col	Maximum number of columns to have on a page. Spillover will be wrapped to multiple pages.

Value

list containing the following elements

- raw_nca: Raw PKNCA output.
- isgood: Boolean indicating the exit status of the function.
- one_table: Dataframe of the entire table with the first lines containing the header.
- one_body: Dataframe of the entire table (data only).
- one_header: Dataframe of the entire header (row and body, no data).
- tables: Named list of tables. Each list element is of the output
- msgs: Vector of text messages describing any errors that were found. format from build_span.

Examples

```
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"

# We need a state variable to be define
sess_res = NCA_test_mksession(session=list(),
     id     = id,
     id_UD  = id_UD,
     id_DW  = id_DW,
     id_ASM = id_ASM,
     full_session=FALSE)

state = sess_res$state

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This is the raw PKNCA output
pknca_res = NCA_fetch_ana_pknca(state, current_ana)

# Parameter reporting details from the ruminate configuration
nps  = state[["NCA"]][["nca_parameters"]][["summary"]]
```

```
# Building the figure
mk_res = mk_table_nca_params(nca_res = pknca_res, nps=nps, digits=3)
mk_res$tables[["Table 1"]]$ft
```

`NCA_add_int` *Adds Analysis Interval to Current Analysis*

Description

Takes the start time, stop time, and NCA parameters and adds them to the intervals table

Usage

```
NCA_add_int(state, interval_start, interval_stop, nca_parameters)
```

Arguments

`state`	NCA state from `NCA_fetch_state()`
`interval_start`	Interval start time (numeric).
`interval_stop`	Interval stop time (numeric).
`nca_parameters`	list of NCA parameters in the interval

Value

State with interval added to the current analysis.

`NCA_append_report` *Append Report Elements*

Description

Takes an NCA state object and appends any reportable elements for the specified report type. On NCA analyses that are in a "good" state will be reported. Those not in a good state will be ignored.

Usage

```
NCA_append_report(state, rpt, rpttype, gen_code_only = FALSE)
```

Arguments

`state`	NCA state from `NCA_fetch_state()`
`rpt`	Report with the current content of the report which will be appended to in this function. For details on the structure see the documentation for `FM_generate_report`.
`rpttype`	Type of report to generate (supported "xlsx", "pptx", "docx").
`gen_code_only`	Boolean value indicating that only code should be generated (`FALSE`).

Value

list containing the following elements

- isgood: Return status of the function.
- hasrptele: Boolean indicator if the module has any reportable elements.
- code: Code to create report elements.
- msgs: Messages to be passed back to the user.
- rpt: Report with any additions passed back to the user.

See Also

FM_generate_report

Examples

```
# We need a state object to use below
sess_res = NCA_test_mksession(session=list(), full_session=FALSE)
state = sess_res$state

# here we need an empty report object for tabular data
rpt = list(summary = list(), sheets=list())

# Now we append the report indicating we want
# Excel output:
rpt_res = NCA_append_report(state,
 rpt           = rpt,
 rpttype       = "xlsx",
 gen_code_only = TRUE)

# Shows if report elements are present
rpt_res$hasrptele

# Code chunk to generate report element
cat(paste(rpt_res$code, collapse="\n"))
```

---

nca_builder *Builds NCA Code from ui Elements*

Description

Takes the current analysis in the state object and creates the code to run the analysis

Usage

```
nca_builder(state)
```

Arguments

state	NCA state from NCA_fetch_state() JMH update the return list below

Value

NCA state with the NCA for the current analysis built.

Examples

```
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"

# We need a module variables to be defined
sess_res = NCA_test_mksession(session=list(),
     id     = id,
     id_UD  = id_UD,
     id_DW  = id_DW,
     id_ASM = id_ASM,
     full_session=FALSE)

state = sess_res$state

state = nca_builder(state)
```

NCA_fetch_ana_ds *Fetch Analysis Dataset*

Description

Fetches the dataset used for the specified analysis

Usage

```
NCA_fetch_ana_ds(state, current_ana)
```

Arguments

state	NCA state from NCA_fetch_state()
current_ana	Current value in the analysis

Value

Dataset from the ds field of FM_fetch_ds()

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
```

```
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_fetch_ana_pknca *Fetch PKNCA Results Object*

Description

Fetches the PKNCA output for a specified analysis

Usage

```
NCA_fetch_ana_pknca(state, current_ana)
```

Arguments

state	NCA state from NCA_fetch_state()
current_ana	Current value in the analysis

Value

Dataset from the ds field of FM_fetch_ds()

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
```

```
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_fetch_code *Fetch Module Code*

Description

Fetches the code to generate results seen in the app

Usage

```
NCA_fetch_code(state)
```

Arguments

state	NCA state from NCA_fetch_state()

Value

Character object vector with the lines of code

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)
```

```
# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_fetch_current_ana *Fetches Current Analysis*

Description

Takes an NCA state and returns the current active analysis

Usage

```
NCA_fetch_current_ana(state)
```

Arguments

state	NCA state from NCA_fetch_state()

Value

List containing the details of the current analysis. The structure of this list is the same as the structure of stateNCAanas in the output of NCA_fetch_state().

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)
```

```
# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_fetch_current_obj *Fetches the Current Analysis Object*

Description

Takes the current state and object type and returns the currently selected object. For example if you have specified figure, it will look at the output figure selected and the figure number of that figure and return the ggplot object for that. by subject id highlighting of certain NCA aspects (e.g. points used for half-life)

Usage

```
NCA_fetch_current_obj(state, obj_type)
```

Arguments

state	NCA state from NCA_fetch_state()
obj_type	Type of object to return (either "table" or "figure").

Value

List with a format that depends on the obj_type. For figures:

- ggplot: ggplot object of the figure.
- isgood: Return status of the function.
- msgs: Messages to be passed back to the user.

For tables:

- df: Dataframe of the current table.
- ft: Flextable object of the current table.
- notes: Any table notes to be included.
- isgood: Return status of the function.
- msgs: Messages to be passed back to the user.

Examples

```
# We need a state object to use below
sess_res = NCA_test_mksession(session=list(), full_session=FALSE)
state = sess_res$state

# Current active table:
res = NCA_fetch_current_obj(state, "table")
res$ft

# Current active figure:
res = NCA_fetch_current_obj(state, "figure")
res$ggplot
```

NCA_fetch_data_format *Fetches Details About Data Requirements*

Description

Use this to get information about data formats.

Usage

```
NCA_fetch_data_format(
  MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")
)
```

Arguments

MOD_yaml_file	Module configuration file with MC as main section.

Value

List with details about the data formats

Examples

```
NCA_fetch_data_format()
```

NCA_fetch_ds *Fetch Module Datasets*

Description

Fetches the datasets contained in the module

Usage

```
NCA_fetch_ds(state)
```

Arguments

state	NCA state from NCA_fetch_state()

Value

list containing the following elements

- isgood: Return status of the function.
- hasds: Boolean indicator if the module has any datasets
- msgs: Messages to be passed back to the user.
- ds: List with datasets. Each list element has the name of the R-object for that dataset. Each element has the following structure:
  - label: Text label for the dataset
  - MOD_TYPE: Short name for the type of module.
  - id: module ID
  - DS: Dataframe containing the actual dataset.
  - DSMETA: Metadata describing DS
  - code: Complete code to build dataset.
  - checksum: Module checksum.
  - DSchecksum: Dataset checksum.

Examples

```
# We need a state object to use below
sess_res = NCA_test_mksession(session=list(), full_session=FALSE)
state = sess_res$state

myDs = NCA_fetch_ds(state)
```

NCA_fetch_np_meta *Fetches NCA Parameter Meta Information*

Description

This provides meta information about NCA parameters. This includes parameter names, text descriptions, formatting (md and LaTeX).

Usage

```
NCA_fetch_np_meta(
  MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")
)
```

Arguments

MOD_yaml_file	Module configuration file with MC as main section.

Value

List with the following elements:

- choices: List parameter choices grouped by values specified in the module configuration file.
- summary: Data frame with meta data about the NCA parameters with the following columns:
  - parameter: Name of parameter in PKNCA.
  - text: Name of parameter in plain text.
  - md: Parameter name formatted in Markdown.
  - latex: Parameter name formatted using LaTeX.
  - description: Verbose description in plain text for the parameter.

Examples

```
NCA_fetch_np_meta()
```

NCA_fetch_PKNCA_meta *Fetches PKNCA Metadata*

Description

Compiles Metadata from PKNCA

Usage

```
NCA_fetch_PKNCA_meta()
```

Value

Dataframe containing PKCNA metadata for NCA parameters.

Examples

```
PKNCA_meta = NCA_fetch_PKNCA_meta()
utils::head(PKNCA_meta)
```

NCA_fetch_state *Fetch ruminate State*

Description

Merges default app options with the changes made in the UI

Usage

```
NCA_fetch_state(
  id,
  input,
  session,
  FM_yaml_file,
  MOD_yaml_file,
  id_ASM,
  id_UD,
  id_DW,
  react_state
)
```

Arguments

id	Shiny module ID
input	Shiny input variable
session	Shiny session variable
FM_yaml_file	App configuration file with FM as main section.
MOD_yaml_file	Module configuration file with MC as main section.
id_ASM	ID string for the app state management module used to save and load app states
id_UD	ID string for the upload data module used to save and load app states
id_DW	ID string for the data wrangling module used to save and load app states
react_state	Variable passed to server to allow reaction outside of module (NULL)

Value

list containing the current state of the app including default values from the yaml file as well as any changes made by the user. The list has the following structure:

- yaml: Full contents of the supplied yaml file.
- MC: Module components of the yaml file.
- NCA:
  - ana_cntr: Analysis counter.
  - anas: List of analyses: Each analysis has the following structure:

    * ana_dsview: Dataset view/ID (name from DSV) selected as a data source for this analysis.
    * ana_scenario: Analysis scenario selected in the UI
    * checksum: checksum of the analysis (used to detect changes in the analysis).
    * code: Code to generate analysis from start to finish or error messages if code generation/analysis failed.
    * code_components: List containing the different components from code
    * col_conc: Column from ana_dsview containing the concentration data.
    * col_dose: Column from ana_dsview containing the dose amount.
    * col_dur: Column from ana_dsview containing the infusion duration or N/A if unused.
    * col_group: Columns from ana_dsview containing other grouping variables.
    * col_id: Column from ana_dsview containing the subject IDs.
    * col_ntime: Column from ana_dsview containing the nominal time values
    * col_route: Column from ana_dsview containing the dosing route.
    * col_time: Column from ana_dsview containing the time values.
    * id: Character id (`ana_idx`).
    * idx: Numeric id (1).
    * include_units: Boolean variable indicating in units should included in the analysis.
    * interval_range: Vector with the first element representing he beginning of the interval and the second element containing the end of the interval.
    * intervals: List of the intervals to include.
    * isgood: Current status of the analysis.
    * key: Analysis key acts as a title/caption (user editable)
    * msgs: Messages generated when checking configuration and analysis options.
    * nca_config: List of NCA configuration options for this analysis.
    * nca_object_name: Prefix for NCA objects associated with this analyis.
    * nca_parameters: NCA parameters selected for calculation in the UI.
    * notes: Analysis notes (user editable)
    * objs: List of names and values for objects created with generated code.
    * sampling: Sampling method either "sparse" or "serial"
    * units_amt: Amount units.
    * units_conc: Concentration units.
    * units_dose: Dosing units.
    * units_time: Time units.
- current_ana: Currently selected analysis (list name element from anas).
- DSV: Available data source views (see `FM_fetch_ds`)
- checksum: This is an MD5 sum of the module (checksum of the analysis checksums).
- nca_config: List of PKNCA configuration options for this analysis.
- nca_parameters: List with two elements
    * choices: List consisting of "Common Parameters" and "Other" (used for grouping in the UI). Each of these is a list of text parameter names with a value of the PKNCA parameter name.
    * summary: Summary table with the following columns:

        - parameter: PKNCA Paramter name.
        - text: Name used in text output.
        - md: Name used markdown output.
        - latex: Name used in latex output.
        - description: Verbose textual description of the parameter.
    - ui: Current value of form elements in the UI.
    - ui_ana_map: Map between UI element names and analysis in the object you get from NCA_fetch_current_ana
    - ui_ids: Vector of UI elements for the module.
    - ui_hold: List of hold elements to disable updates before a full ui referesh is complete.
- MOD_TYPE: Character data containing the type of module "NCA"
- id: Character data containing the module id module in the session variable.
- FM_yaml_file: App configuration file with FM as main section.
- MOD_yaml_file: Module configuration file with MC as main section.

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
```

```
                        react_state     = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

`NCA_find_col` *Determines Default Column Name*

Description

Based on the current analysis, value from the UI, an optional list of patterns to search, and column names from a dataset this function tries to find a default value for a column in the analysis (e.g. subject id, dose, concentration, etc).

Generally the following is done:

- If curr_ui has a non-NULL, non-"" value it is compared to dscols. If it is found there that value is returned.
- If not then the patterns are considered. If the patterns from the YAML file are not NULL they are compared sequentially to the columns names. The first match found is returned.
- If nothing is found then the first value of dscols is returned.

Usage

```
NCA_find_col(
  curr_ana = NULL,
  curr_ui = NULL,
  patterns = NULL,
```

```
  dscols,
  null_ok = FALSE
)
```

Arguments

curr_ana	Current value in the analysis
curr_ui	Current value in UI
patterns	List of regular expression patterns to consider.
dscols	Columns from the dataset.
null_ok	Logical value indicating if a null result (nothing found) is OK (default: FALSE)

Value

Name of column found based on the rules above.

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)
```

```
# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_init_state *Initialize NCA Module State*

Description

Creates a list of the initialized module state

Usage

```
NCA_init_state(FM_yaml_file, MOD_yaml_file, id, id_UD, id_DW, session)
```

Arguments

FM_yaml_file	App configuration file with FM as main section.
MOD_yaml_file	Module configuration file with MC as main section.
id	ID string for the module.
id_UD	ID string for the upload data module used to handle uploads or the name of the list element in react_state where the data set is stored.
id_DW	ID string for the data wrangling module to process any uploaded data
session	Shiny session variable (in app) or a list (outside of app)

Value

list containing an empty NCA state

NCA_load_scenario *Loads Pre-Defined Scenario*

Description

Loads a pre-defined analysis scneario from the NCA YAML config file.

Usage

```
NCA_load_scenario(state, ana_scenario)
```

Arguments

state	NCA state from NCA_fetch_state()
ana_scenario	Short name of the analysis scenario to load from the config file.

Value

NCA state object with the scenario loaded and relevant notifications set.

NCA_mkactive_ana *Fetch PKNCA Results Object*

Description

Fetches the PKNCA output for a specified analysis

Usage

```
NCA_mkactive_ana(state, ana_id)
```

Arguments

state	NCA state from NCA_fetch_state()
ana_id	Analysis ID to make active.

Value

State with the analysis ID made active. JMH add to example script below

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
```

```
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_new_ana *Initialize New Analysis*

Description

Creates a new NCA analysis in an NCA module

Usage

```
NCA_new_ana(state)
```

Arguments

state	NCA state from NCA_fetch_state()

Value

NCA state object containing a new empty analysis and that analysis is set as the current active analyisis

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
```

```
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_process_current_ana

Processes Current Analysis to be Run

Description

Takes the current analysis and checks different aspects to for any issues to make sure it's good to go.

Usage

```
NCA_process_current_ana(state)
```

Arguments

state	NCA state from NCA_fetch_state()

Value

Current analysis list with isgood and msgs set

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)
```

```
# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_Server *Fetch Non-Compartmental Analysis State*

Description

Merges default app options with the changes made in the UI

Usage

```
NCA_Server(
  id,
  FM_yaml_file = system.file(package = "formods", "templates", "formods.yaml"),
  MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml"),
  id_ASM = "ASM",
  id_UD = "UD",
  id_DW = "DW",
  deployed = FALSE,
  react_state = NULL
)
```

Arguments

id	An ID string that corresponds with the ID used to call the modules UI elements
FM_yaml_file	App configuration file with FM as main section.
MOD_yaml_file	Module configuration file with MC as main section.
id_ASM	ID string for the app state management module used to save and load app states
id_UD	ID string for the upload data module used to save and load app states
id_DW	ID string for the data wrangling module used to save and load app states
deployed	Boolean variable indicating whether the app is deployed or not.
react_state	Variable passed to server to allow reaction outside of module (NULL)

Value

list containing the current state of the app including default values from the yaml file as well as any changes made by the user. The list has the following structure:

- yaml: Full contents of the supplied yaml file.
- MC: Module components of the yaml file.
- NCA:
  - isgood: Boolean object indicating if the file was successfully loaded.
  - checksum: This is an MD5 sum of the contents element and can be used to detect changes in the state.
- MOD_TYPE: Character data containing the type of module "NCA"
- id: Character data containing the module id module in the session variable.
- FM_yaml_file: App configuration file with FM as main section.
- MOD_yaml_file: Module configuration file with MC as main section.

Examples

```
if(interactive()){
# original file: inst/templates/ruminate.R
library(formods)
library(ruminate)

# These are suggested packages
library(shinydashboard)
#library(ggpubr)
#library(plotly)
#library(shinybusy)
library(prompter)
#library(utils)

tags$style("@import url(https://use.fontawesome.com/releases/v6.4.0/css/all.css);")

# You can copy these locally and customize them for your own needs. Simply
# change the assignment to the local copy you've modified.
formods.yaml  = system.file(package="formods",  "templates",  "formods.yaml")
ASM.yaml      = system.file(package="formods",  "templates",  "ASM.yaml")
UD.yaml       = system.file(package="formods",  "templates",  "UD.yaml")
DW.yaml       = system.file(package="formods",  "templates",  "DW.yaml")
FG.yaml       = system.file(package="formods",  "templates",  "FG.yaml")
NCA.yaml      = system.file(package="ruminate", "templates",  "NCA.yaml")


# Name of  file to indicate we need to load testing data
ftmptest = file.path(tempdir(), "ruminate.test")


# Making sure that the deployed object is created
if(!exists("deployed")){
  deployed = FALSE
```

```
}

# Making sure that the run_dev object is created
if(file.exists(file.path(tempdir(), "RUMINTE_DEVELOPMENT"))){
  run_dev  = TRUE
}else{
  run_dev  = FALSE
}

# If the SETUP.R file exists we source it
if(file.exists("SETUP.R")){
  source("SETUP.R")
}


# If the DEPLOYED file marker existrs we set deployed to TRUE
if(file.exists("DEPLOYED")){
  deployed = TRUE
}

CSS <- "
.wrapfig {
  float: right;
  shape-margin: 20px;
  margin-right: 20px;
  margin-bottom: 20px;
}
"

#https://fontawesome.com/icons?from=io
logo_url =
  "https://raw.githubusercontent.com/john-harrold/ruminate/main/man/figures/logo.png"
data_url =
  "https://github.com/john-harrold/formods/raw/master/inst/test_data/TEST_DATA.xlsx"
run_url  =
  "https://runruminate.ubiquity.tools/"
use_url  =
  "https://useruminate.ubiquity.tools/"
main_url =
  "https://ruminate.ubiquity.tools/"
issue_url =
  "https://github.com/john-harrold/ruminate/issues"

intro_text = tags$p(
"Ruminate is a shiny module for pharmacometric data processing,
visualization, and analysis. It consists of separate shiny modules that
provide interfaces into common R packages and provides the underlying code.
This is done to facilitate usage of those packages and to provide reproducible
analyses." ,
tags$li( "To find out more visit  ",
        tags$a("ruminate.ubiquity.tools", href=main_url),""),
tags$li( "To give it a try you can download a test dataset ",
        tags$a("here", href=data_url),""),
```

```
tags$li( "Go to  ",
        tags$a("useruminate.ubiquity.tools", href=use_url)," for a video
        demonstrating how to use ruminate"),
tags$li( "If you run into any problems, have questions, or want a feature please
        visit the ",
        tags$a("issues", href=issue_url)," page")
)


ui <- shinydashboard::dashboardPage(
  skin="black",
  shinydashboard::dashboardHeader(title="ruminate"),
  shinydashboard::dashboardSidebar(
     shinydashboard::sidebarMenu(
       shinydashboard::menuItem("Load/Save",
                                tabName="loadsave",
                                icon=icon("arrow-down-up-across-line")) ,
     shinydashboard::menuItem("Transform Data", tabName="wrangle", icon=icon("shuffle")),
     shinydashboard::menuItem("Visualize",    tabName="plot",   icon=icon("chart-line")),
      shinydashboard::menuItem("NCA",         tabName="nca",    icon=icon("chart-area")),
     shinydashboard::menuItem("App Info",     tabName="sysinfo", icon=icon("book-medical"))
     )
  ),
  shinydashboard::dashboardBody(
  tags$head(
    tags$style(HTML(CSS))
  ),
    shinydashboard::tabItems(
       shinydashboard::tabItem(tabName="nca",
               shinydashboard::box(title="Run Non-Compartmental Analysis", width=12,
               fluidRow( prompter::use_prompt(),
               column(width=12,
               htmlOutput(NS("NCA",  "NCA_ui_compact")))))
               ),
       shinydashboard::tabItem(tabName="loadsave",
         #     shinydashboard::box(title=NULL, width=12,
               shinydashboard::tabBox(
                 width = 12,
                 title = NULL,
                 shiny::tabPanel(id="load_data",
                          title=tagList(shiny::icon("file-arrow-up"),
                                        "Load Data"),
                   fluidRow(
                     column(width=6,
                       div(style="display:inline-block;width:100%",
                       htmlOutput(NS("UD", "ui_ud_load_data"))),
                       htmlOutput(NS("UD", "ui_ud_clean")),
                       htmlOutput(NS("UD", "ui_ud_select_sheets")),
                       htmlOutput(NS("UD", "ui_ud_text_load_result"))),
                     column(width=6,
                         tags$p(
                             tags$img(
                             class = "wrapfig",
```

```
                        src   = logo_url,
                        width = 150,
                        alt = "formods logo" ),
                      intro_text
                      ))
            ),
            fluidRow(
              column(width=12,
                     div(style="display:inline-block;vertical-align:top",
                               htmlOutput(NS("UD", "ui_ud_data_preview")))
                     ))
            ),
            shiny::tabPanel(id="save_state",
                     title=tagList(shiny::icon("arrow-down-up-across-line"),
                                   "Save or Load Analysis"),
            fluidRow(
              column(width=5,
                     div(style="display:inline-block;vertical-align:top",
              htmlOutput(NS("ASM", "ui_asm_compact"))
              ))
              )
            )
          )
    #     ),
          ),
shinydashboard::tabItem(tabName="wrangle",
    shinydashboard::box(title="Transform and Create Views of Your Data", width=12,
        fluidRow(
        column(width=12,
        htmlOutput(NS("DW",  "DW_ui_compact")))))
        ),
shinydashboard::tabItem(tabName="plot",
        shinydashboard::box(title="Visualize Data", width=12,
        htmlOutput(NS("FG",  "FG_ui_compact")))),
shinydashboard::tabItem(tabName="sysinfo",
    #     box(title="System Details", width=12,
          shinydashboard::tabBox(
            width = 12,
            title = NULL,
            shiny::tabPanel(id="sys_packages",
                     title=tagList(shiny::icon("box-open"),
                                   "Installed Packages"),
            htmlOutput(NS("ASM", "ui_asm_sys_packages"))
            ),
            shiny::tabPanel(id="sys_modules",
                     title=tagList(shiny::icon("cubes"),
                                   "Loaded Modules"),
            htmlOutput(NS("ASM", "ui_asm_sys_modules"))
            ),
            shiny::tabPanel(id="sys_log",
                     title=tagList(shiny::icon("clipboard-list"),
                                   "Log"),
            verbatimTextOutput(NS("ASM", "ui_asm_sys_log"))
```

```
                ),
                shiny::tabPanel(id="sys_options",
                         title=tagList(shiny::icon("sliders"),
                                       "R Options"),
                htmlOutput(NS("ASM", "ui_asm_sys_options"))
                )
        #       )
              ))
      )
    )
  )

# Main app server
server <- function(input, output, session) {

  # Empty reactive object to track and react to
  # changes in the module state outside of the module
  react_FM = reactiveValues()

  # Module IDs and the order they are needed for code generation
  mod_ids = c("UD", "DW", "FG", "NCA", "MB")

  # If the ftmptest file is present we load test data
  if(file.exists(ftmptest)){
    NCA_test_mksession(
      session,
      id     = "NCA",
      id_UD  = "UD",
      id_DW  = "DW",
      id_ASM = "ASM"
    )
  }

  # Module servers
  formods::ASM_Server( id="ASM",
                       deployed      = deployed,
                       react_state   = react_FM,
                       FM_yaml_file  = formods.yaml,
                       MOD_yaml_file = ASM.yaml,
                       mod_ids       = mod_ids)
  formods::UD_Server(  id ="UD", id_ASM = "ASM",
                       deployed         = deployed,
                       react_state      = react_FM,
                       MOD_yaml_file    = UD.yaml,
                       FM_yaml_file     = formods.yaml)
  formods::DW_Server(  id="DW",       id_ASM = "ASM",
                       id_UD = "UD",
                       deployed         = deployed,
                       react_state      = react_FM,
                       MOD_yaml_file    = DW.yaml,
                       FM_yaml_file     = formods.yaml)
  formods::FG_Server(  id="FG",     id_ASM = "ASM",
                       id_UD = "UD", id_DW = "DW",
```

```
                        deployed         = deployed,
                        react_state      = react_FM,
                        MOD_yaml_file    = FG.yaml,
                        FM_yaml_file     = formods.yaml)
  ruminate::NCA_Server(id    ="NCA", id_ASM = "ASM",
                        id_UD = "UD", id_DW  = "DW",
                        deployed         = deployed,
                        react_state      = react_FM,
                        MOD_yaml_file    = NCA.yaml,
                        FM_yaml_file     = formods.yaml)

}

shinyApp(ui, server)
}
```

NCA_set_current_ana *Sets Current Analysis*

Description

Takes an NCA state and an analysis list and sets that figure list as the value for the active figure

Usage

```
NCA_set_current_ana(state, ana)
```

Arguments

state	NCA state from NCA_fetch_state()
ana	Analysis list from NCA_fetch_current_ana

Value

State with the current analysis updated

Examples

```
library(ruminate)
# Module IDs
id     = "NCA"
id_UD  = "UD"
id_DW  = "DW"
id_ASM = "ASM"
# We need session and input variables to be define
 sess_res = NCA_test_mksession(session=list(),
      id     = id,
      id_UD  = id_UD,
      id_DW  = id_DW,
```

```
      id_ASM = id_ASM,
      full_session=FALSE)

# Extracting the session and input variables
session      = sess_res$session
input        = sess_res$input
react_state  = list()

# We also need configuration files
FM_yaml_file  = system.file(package = "formods",  "templates", "formods.yaml")
MOD_yaml_file = system.file(package = "ruminate", "templates", "NCA.yaml")

# Getting the current module state
state = NCA_fetch_state(id             = id,
                        input          = input,
                        session        = session,
                        FM_yaml_file   = FM_yaml_file,
                        MOD_yaml_file  = MOD_yaml_file,
                        id_ASM         = id_ASM,
                        id_UD          = id_UD,
                        id_DW          = id_DW,
                        react_state    = react_state)

# Pulls out the active analysis
current_ana = NCA_fetch_current_ana(state)

# This will get the dataset associated with this analysis
ds = NCA_fetch_ana_ds(state, current_ana)

# After making changes you can update those in the state
state = NCA_set_current_ana(state, current_ana)

# You can use this to check the current analysis
current_ana = NCA_process_current_ana(state)

# This will pull out the code for the module
fc_res = NCA_fetch_code(state)

# This will use patterns defined for the site to detect
# columns. In this example we are detecting the id column:
id_col = NCA_find_col(
  patterns = state[["MC"]][["detect_col"]][["id"]],
  dscols   = names(ds$DS))

# This creates a new analysis
state = NCA_new_ana(state)
```

NCA_test_mksession *Populate Session Data for Module Testing*

Description

Populates the supplied session variable for testing.

Usage

```
NCA_test_mksession(
  session,
  id = "NCA",
  id_UD = "UD",
  id_DW = "DW",
  id_ASM = "ASM",
  full_session = TRUE
)
```

Arguments

session	Shiny session variable (in app) or a list (outside of app)
id	An ID string that corresponds with the ID used to call the modules UI elements
id_UD	An ID string that corresponds with the ID used to call the UD modules UI elements
id_DW	An ID string that corresponds with the ID used to call the DW modules UI elements
id_ASM	An ID string that corresponds with the ID used to call the ASM modules UI elements
full_session	Boolean to indicate if the full test session should be created (default TRUE).

Value

list with the following elements

- isgood: Boolean indicating the exit status of the function.
- session: The value Shiny session variable (in app) or a list (outside of app) after initialization.
- input: The value of the shiny input at the end of the session initialization.
- state: App state.
- rsc: The react_state components.

Examples

```
sess_res = NCA_test_mksession(session=list(), full_session=FALSE)
```

`ruminate` *ruminate: Shiny app and module to facilitate pharamacometrics analysis*

Description

This is done by creating a Shiny interface to different tools for data transformation (`dplyr` and `tidyr`), plotting (`ggplot2`), and noncompartmental analysis (`PKNCA`). These results can be reported in Excel, Word or PowerPoint. The state of the app can be saved and loaded at a later date. When saved, a script is generated to reproduce the different actions in the Shiny interface.

Runs the pharmacometrics ruminate app.

Usage

```
ruminate(
  host = "127.0.0.1",
  port = 3838,
  server_opts = list(shiny.maxRequestSize = 30 * 1024^2),
  mksession = FALSE
)
```

Arguments

`host`	Hostname of the server ("127.0.0.1")
`port`	Port number for the app (3838)
`server_opts`	List of options (names) and their vlues (value) e.g. `list(shiny.maxRequestSize = 30 * 1024^2)`.
`mksession`	Boolean value, when TRUE will load test session data for app testing

Value

Nothing is returned, this function just runs the built-in ruminate app.

Author(s)

Maintainer: John Harrold <john.m.harrold@gmail.com> (ORCID)

See Also

https://ruminate.ubiquity.tools/

Examples

```
if (interactive()) {
ruminate()
}
```

`run_nca_components` *Runs NCA for the Current Analysis*

Description

Takes the current state and runs the current analysis in that state.

Usage

```
run_nca_components(
  state,
  components = c("nca", "fg_ind_obs", "tb_ind_obs", "tb_ind_params")
)
```

Arguments

`state`	NCA state from `NCA_fetch_state()`
`components`	List of components to run. By default it will run all of the following. If you just need to regenerate a figure based on the current nca results you can just specify that component. These are the valid components: • nca: Run NCA analysis • fg_ind_obs: Build the figure(s) with the indiviudal observations. • tb_ind_obs: Build the table(s) with the indiviudal observations. • tb_ind_params: Build the table(s) with the indiviudal parameters.

Value

List with the following components:

- isgood: Return status of the function.
- msgs: Error messages if any issues were encountered.
- nca_res: PKNCA results if run was successful.

Examples

```
# We need a state object to use below
sess_res = NCA_test_mksession(session=list(), full_session=FALSE)
state = sess_res$state

state = run_nca_components(state, components="tb_ind_params")
```

Index