Package ‘tm1r’

March 19, 2020

Type Package
Title The Integration Between 'IBM COGNOS TM1' and R
Version 1.1.4
Author Muhammed Ali Onder
Maintainer Muhammed Ali Onder <muhammedalionder@gmail.com>
Description Useful functions to connect to 'TM1' <https://www.ibm.com/uk-en/marketplace/planning-and-analytics> instance from R via REST API. With the functions in the package, data can be imported from 'TM1' via mdx view or native view, data can be sent to 'TM1', processes and chores can be executed, and cube and dimension metadata information can be taken.
License GPL (>= 2)
Encoding UTF-8
LazyData true
Imports jsonlite, httr
Depends R (>= 3.0.0)
NeedsCompilation no
URL http://github.com/muhammedalionder/tm1r
BugReports http://github.com/muhammedalionder/tm1r/issues
Repository CRAN
Date/Publication 2020-03-19 20:30:02 UTC

R topics documented:

tm1_api_request ............................................. 2
tm1_connection ............................................. 3
tm1_create_element ......................................... 4
tm1_create_mdx .............................................. 4
tm1_create_subset .......................................... 7
tm1_create_view ............................................ 7
tm1_delete_element ......................................... 8
tm1_delete_subset .......................................... 9
tm1_api_request

Description

Makes a api request to tm1 server with url and body specified

Usage

```
tm1_api_request(tm1_connection, url, body ="", type = "GET")
```

Arguments

- **tm1_connection**: tm1 connection object returned by the function `tm1_connection`
- **url**: URL address for rest api request
- **body**: body text of request
- **type**: type of api request. Requests in httr package are supported like GET, POST, DELETE, PATCH
Examples

```r
## Not run:
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
url <- "https://localhost:8881/api/v1/Cubes(\'SalesCube\')/Dimensions"
 tm1_api_request(con_obj, url, type = "GET")

## End(Not run)
```

tm1_connection

**TM1 Connection**

**Description**

Creates and returns a connection object to connect to TM1 via REST API.

**Usage**

```r
tm1_connection(adminhost = "localhost", httpport = ",",
username = "admin", password = "apple",
namespace="", ssl=TRUE, base_url="")
```

**Arguments**

- **adminhost**: adminhost of tm1 model
- **httpport**: httpport of tm1 model
- **username**: username to connect to tm1 model
- **password**: password of the username
- **namespace**: ID of namespace should be specified if there is CAM security. Should be blank for native security
- **ssl**: If UseSSL parameter is T in tm1s.cfg file, then TRUE. Else FALSE. Default is TRUE
- **base_url**: when connecting to cloud, this option can be used instead of adminhost and httpport

**Examples**

```r
## Not run:
tm1_connection("localhost", "8881", "admin", "apple")
tm1_connection(username="admin", password="apple",
base_url = "https://[Customer_Name].planning-analytics.ibmcloud.com/tm1/api/[Server Name]/")

## End(Not run)
```
**tm1_create_element**  
*TM1 Create New Element to a Dimension*

**Description**

Inserts a new element to the dimension

**Usage**

```
tm1_create_element(tm1_connection,  
    dimension, element, parent="", weight=1)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of dimension
- `element`: Name of new element
- `parent`: Name of parent of new element. Leave bland if there is no parent.
- `weight`: Weight of the element as a component to the parent. Default is 1

**Examples**

```r
## Not run:
tm1_create_element(tm1_connection("localhost", "8881", "admin", "apple"),  
    "month", "test", "Year")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_create_element(con_obj, "month", "test", "Year")
```

## End(Not run)

---

**tm1_create_mdx**  
*TM1 Generate mdx for a cube view*

**Description**

Returns mdx as a string to use in the function `tm1_get_mdx_view`
tm1_create_mdx

Usage

\[ \text{tm1\_create\_mdx}(\text{cube}, \text{rowdim}1, \text{rowsub}1, \text{rowel}1, \text{rowdim}2, \text{rowsub}2, \text{rowel}2, \text{rowdim}3, \text{rowsub}3, \text{rowel}3, \text{coldim}1, \text{colsub}1, \text{colel}1, \text{coldim}2, \text{colsub}2, \text{colel}2, \text{titledim}1, \text{titleel}1, \text{titledim}2, \text{titleel}2, \text{titledim}3, \text{titleel}3, \text{titledim}4, \text{titleel}4, \text{titledim}5, \text{titleel}5, \text{titledim}6, \text{titleel}6, \text{titledim}7, \text{titleel}7, \text{titledim}8, \text{titleel}8, \text{titledim}9, \text{titleel}9, \text{titledim}10, \text{titleel}10, \text{rowsuppress}, \text{colsuppress} ) \]

Arguments

cube Name of the cube
rowdim1 Name of dimension in 1st row
rowsub1 Subset of dimension in 1st row
rowel1 Element of dimension in 1st row. If multiple, seperated by ",". This should be passed if subset is not provided
rowdim2 Name of dimension in 2nd row
rowsub2 Subset of dimension in 2nd row
rowel2 Element of dimension in 2nd row. If multiple, seperated by ",". This should be passed if subset is not provided
rowdim3 Name of dimension in 3rd row
rowsub3 Subset of dimension in 3rd row
rowel3 Element of dimension in 3rd row. If multiple, seperated by ",". This should be passed if subset is not provided
coldim1 Name of dimension in 1st col
colsub1 Subset of dimension in 1st col
colel1 Element of dimension in 1st col. If multiple, seperated by ",". This should be passed if subset is not provided
coldim2 Name of dimension in 2nd col
colsub2 Subset of dimension in 2nd col
colel2 Element of dimension in 2nd col. If multiple, seperated by ",". This should be passed if subset is not provided
titledim1 Name of dimension in title
Examples

```r
## Not run:

tm1_create_mdx( "SalesCube", rowdim1="account1", rowel1 = "Sales", coldim1="month", colel1="Jan",
                 titledim1 = "actvsbud", titleel1 = "Actual",
                 titledim2 = "region", titleel2 = "Argentina",
                 titledim3 = "model", titleel3 = "S Series 1.8 L Sedan",
                 rowsuppress=TRUE, colsuppress = FALSE)

## End(Not run)
```
tm1_create_subset  

**TM1 Create New Subset to a Dimension**

**Description**

Creates a new subset to the dimension

**Usage**

```r
tm1_create_subset(tm1_connection, 
  dimension, subset, element="", mdx="", overwrite=TRUE)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of dimension
- `subset`: Name of new subset
- `element`: Name of elements seperated by | for static subset
- `mdx`: mdx of subset for dynamic subset
- `overwrite`: TRUE or FALSE. If TRUE, subset is overwritten

**Examples**

```r
## Not run:
tm1_create_subset(tm1_connection("localhost", "8881", "admin", "apple"), 
  "month", "Q1Months", element = "Jan|Feb|Mar")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_create_element(con_obj, "month", "all", mdx = "[month].MEMBERS")

## End(Not run)
```

tm1_create_view  

**TM1 Create View from mdx**

**Description**

Creates cube view with mdx

**Usage**

```r
tm1_create_view(tm1_connection, cube, view, mdx)
```
Arguments

- `tm1_connection`: `tm1` connection object returned by the function `tm1_connection`
- `cube`: Name of cube
- `view`: Name of view to be created
- `mdx`: MDX of view as a string

Examples

```r
## Not run:
mdx <- "SELECT
  NON EMPTY
  {[month].[Jan],[month].[Feb],[month].[Mar]}
  ON COLUMNS,
  NON EMPTY
  {[account1].[Price],[account1].[Units]}
  ON ROWS
FROM [SalesCube]
WHERE
  ([actvsbud].[actvsbud].[Actual],
  [region].[region].[Argentina],
  [model].[model].[S Series 1.8 L Sedan]
)"
tm1_create_view(
  tm1_connection("localhost", "8881", "admin", "apple"),
  "SalesCube", "test", mdx)

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_create_view(con_obj, "SalesCube", "test", mdx)

## End(Not run)
```
Arguments

- **tm1_connection**: tm1 connection object returned by the function `tm1_connection`
- **dimension**: Name of dimension
- **element**: Name of element
- **parent**: Name of parent of element. If parent is specified, component delete will be done. If parent is omitted, element will be deleted from dimension

Examples

```r
## Not run:
tm1_delete_element(tm1_connection("localhost", "8881", "admin", "apple"),
"month", "test", "Year")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_delete_element(con_obj, "month", "test")
## End(Not run)
```

---

**tm1_delete_subset**  
**TM1 Delete Subset**

Description

Deletes subset from dimensions

Usage

```
 tm1_delete_subset(tm1_connection, dimension, subset)
```

Arguments

- **tm1_connection**: tm1 connection object returned by the function `tm1_connection`
- **dimension**: Name of dimension
- **subset**: Name of subset

Examples

```r
## Not run:
 tm1_delete_subset(tm1_connection("localhost", "8881", "admin", "apple"),
"month", "test")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_delete_subset(con_obj, "month", "test")
## End(Not run)
```
tm1_delete_view  TM1 Delete View

Description
Deletes cube view

Usage
    tm1_delete_view(tm1_connection, cube, view)

Arguments
    tm1_connection   tm1 connection object returned by the function tm1_connection
    cube             Name of cube
    view             Name of view to be deleted

Examples
    # Not run:
    tm1_delete_view(
        tm1_connection("localhost", "8881", "admin", "apple"),
        "SalesCube", "test")

    con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
    tm1_delete_view(con_obj, "SalesCube", "test")

    # End(Not run)

tm1_get_config  TM1 Get Configuration

Description
Gets configuration of tm1 instance

Usage
    tm1_get_config(tm1_connection)

Arguments
    tm1_connection   tm1 connection object returned by the function tm1_connection
### Examples

```r
## Not run:
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_config(con_obj)
## End(Not run)
```

## Description

Gets list of cubes

## Usage

```r
tm1_get_cubes(tm1_connection, ShowControlObjects = FALSE)
```

## Arguments

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `ShowControlObjects`: If TRUE, control cubes are also listed. Default is FALSE

## Examples

```r
## Not run:
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_cubes(con_obj)
## End(Not run)
```

### tm1_get_cube_dimensions

#### Description

Gets dimensions of a cube

#### Usage

```r
tm1_get_cube_dimensions(tm1_connection, cube)
```
tm1_get_data

TM1 Get Data from a Cube

Description

 Gets data from a cube, Supports up-to 10 dimension for now

Usage

  tm1_get_data(tm1_connection, cube,
               element1="", element2="",
               element3="", element4="",
               element5="", element6="",
               element7="", element8="",
               element9="", element10="")

Arguments

  tm1_connection  tm1 connection object returned by the function tm1_connection
  cube            Name of a cube as a string
  element1        Element from 1st dimension of cube. Leave empty if there is no corresponding dimension
  element2        Element from 2nd dimension of cube. Leave empty if there is no corresponding dimension
  element3        Element from 3rd dimension of cube. Leave empty if there is no corresponding dimension
  element4        Element from 4th dimension of cube. Leave empty if there is no corresponding dimension
  element5        Element from 5th dimension of cube. Leave empty if there is no corresponding dimension
**tm1_get_dimensions**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>element6</td>
<td>Element from 6th dimension of cube. Leave empty if there is no corresponding dimension</td>
</tr>
<tr>
<td>element7</td>
<td>Element from 7th dimension of cube. Leave empty if there is no corresponding dimension</td>
</tr>
<tr>
<td>element8</td>
<td>Element from 8th dimension of cube. Leave empty if there is no corresponding dimension</td>
</tr>
<tr>
<td>element9</td>
<td>Element from 9th dimension of cube. Leave empty if there is no corresponding dimension</td>
</tr>
<tr>
<td>element10</td>
<td>Element from 10th dimension of cube. Leave empty if there is no corresponding dimension</td>
</tr>
</tbody>
</table>

**Examples**

```r
## Not run:
tm1_get_data(
  tm1_connection("localhost", "8881", "admin", "apple"),
  "SalesCube", "Actual", "Argentina", "Total", "Sales", "Jan")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_data(con_obj, "SalesCube", "Actual", "Argentina", "Total", "Sales", "Jan")
```

**Description**

Gets list of dimensions

**Usage**

`tm1_get_dimensions(tm1_connection, ShowControlObjects = FALSE)`

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `ShowControlObjects`: If TRUE, control dimensions are also listed. Default is FALSE

**Examples**

```r
## Not run:
tm1_get_dimensions(tm1_connection("localhost", "8881", "admin", "apple"))

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_dimensions(con_obj)
```

## End(Not run)
tm1_get_dimension_attributes

**TM1 Get Attributes of a Dimension**

**Description**

Gets attributes of a dimension

**Usage**

```r
tm1_get_dimension_attributes(tm1_connection, dimension)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of a dimension as a string

**Examples**

```r
## Not run:
tm1_get_dimension_attributes(
  tm1_connection("localhost", "8881", "admin", "apple"),
  "region")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_dimension_attributes(con_obj, "region")
## End(Not run)
```

tm1_get_dimension_elements

**TM1 Get Elements of a Dimension**

**Description**

Gets elements of a dimension

**Usage**

```r
tm1_get_dimension_elements(tm1_connection, dimension)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of a dimension as a string
Examples

```r
## Not run:
tm1_get_dimension_elements(
    tm1_connection("localhost", "8881", "admin", "apple"),
    "region")
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_dimension_elements(con_obj, "region")
## End(Not run)
```

---

### tm1_get_dimension_subsets

**TM1 Get Subsets of a Dimension**

#### Description

Gets subsets of a dimension

#### Usage

```r
tm1_get_dimension_subsets(tm1_connection, dimension)
```

#### Arguments

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of a dimension as a string

#### Examples

```r
## Not run:
tm1_get_dimension_subsets(
    tm1_connection("localhost", "8881", "admin", "apple"),
    "region")
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_dimension_subsets(con_obj, "region")
## End(Not run)
```
tm1_get_element  **TM1 Get Element of a Dimension**

**Description**
Gets element detail of a dimension. Name, UniqueName, Type, Level, Index, and Components. element or index should be specified.

**Usage**
```
tm1_get_element(tm1_connection, dimension, element='', index = 0)
```

**Arguments**
- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `dimension`: Name of a dimension as a string
- `element`: Name of element as a string
- `index`: Index of element as a numeric

**Examples**
```
## Not run:
tm1_get_element(
  tm1_connection("localhost", "8881", "admin", "apple"),
  "month", "Year")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_element(con_obj, "month", "", 7)

## End(Not run)
```

tm1_get_instances  **TM1 Get Instances**

**Description**
Returns the list of tm1 instances in the specified adminhost.

**Usage**
```
tm1_get_instances(adminhost = "localhost", port = "5898", ssl=TRUE, base_url = "")
```
**tm1_get_log**

**Arguments**

- **adminhost**: adminhost of tm1 models
- **port**: port of admin server
- **ssl**: If TRUE it will be accesses through https
- **base_url**: when connecting to cloud, this option can be used instead of adminhost and httpport

**Examples**

```r
## Not run:
tm1_get_instances()

 tm1_get_instances(adminhist = "localhost",
                   port = "5898", ssl = TRUE)

## End(Not run)
```

---

**tm1_get_log**  
**TM1 Get Logs of an instance**

**Description**

Gets server logs from a tm1 instance

**Usage**

```r
tm1_get_log(tm1_connection, lognumber)
```

**Arguments**

- **tm1_connection**: tm1 connection object returned by the function tm1_connection
- **lognumber**: Number of how many lines of logs you want. Default is 5

**Examples**

```r
## Not run:
tm1_get_log(tm1_connection("localhost", "8881", "admin", "apple"), 10)

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_log(con_obj)

## End(Not run)
```
**tm1_get_mdx_view**

**TM1 Get Data from an MDX View**

**Description**

Gets mdx view data

**Usage**

```r
tm1_get_mdx_view(tm1_connection, mdx, RowElementAsColumn = FALSE)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `mdx`: MDX of view as a string
- `RowElementAsColumn`: if False, row elements will be attached to rownames of data frame

**Examples**

```r
## Not run:
mdx <- "SELECT
    NON EMPTY
    {[month].[Jan],[month].[Feb],[month].[Mar]}
    ON COLUMNS,
    NON EMPTY
    {[account1].[Price],[account1].[Units]}
    ON ROWS
FROM [SalesCube]
WHERE
    ([actvsbud].[actvsbud].[Actual],
     [region].[region].[Argentina],
     [model].[model].[S Series 1.8 L Sedan])"

itm_get_mdx_view(
    tm1_connection("localhost", "8881", "admin", "apple"),
    mdx, RowElementAsColumn=FALSE)

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_mdx_view(con_obj,mdx)

## End(Not run)
```
**tm1_get_native_view**

*TM1 Get Data from a Native View*

**Description**

Gets native view data

**Usage**

```
tm1_get_native_view(tm1_connection, cube, view, RowElementAsColumn= FALSE)
```

**Arguments**

- `tm1_connection`: tm1 connection object returned by the function `tm1_connection`
- `cube`: Name of the cube
- `view`: Name of the view
- `RowElementAsColumn`: if False, row elements will be attached to rownames of data frame

**Examples**

```r
## Not run:
tm1_get_native_view(
  tm1_connection("localhost", "8881", "admin", "apple"),
  "SalesCube", "Default", RowElementAsColumn=FALSE)

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_native_view(con_obj, "SalesCube", "Default")
## End(Not run)
```

---

**tm1_get_subset_elements**

*TM1 Get Elements of a subset*

**Description**

Gets elements of a subset

**Usage**

```
tm1_get_subset_elements(tm1_connection, dimension, subset)
```
**tm1_logout**

**Arguments**

- `tm1_connection` tm1 connection object returned by the function `tm1_connection`
- `dimension` Name of a dimension as a string
- `subset` Name of a subset as a string

**Examples**

```r
## Not run:
tm1_get_subset_elements(
tm1_connection("localhost", "8881", "admin", "apple"),
"region", "default")
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_get_dimension_elements(con_obj, "region", "default")

## End(Not run)
```

---

**tm1_logout**

**TM1 Log Out**

**Description**

Logs out

**Usage**

```r
tm1_logout(tm1_connection)
```

**Arguments**

- `tm1_connection` tm1 connection object returned by the function `tm1_connection`

**Examples**

```r
## Not run:
tm1_logout(tm1_connection("localhost", "8881", "admin", "apple"))
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_logout(con_obj)

## End(Not run)
```
**tm1_run_chore**

**TM1 Run a Chore**

**Description**

Runs a chore

**Usage**

```
tm1_run_chore(tm1_connection, chore)
```

**Arguments**

- `tm1_connection` tm1 connection object returned by the function `tm1_connection`
- `chore` Name of a chore as a string

**Examples**

```r
## Not run:
tm1_run_chore(tm1_connection("localhost", "8881", "admin", "apple"), "test")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_run_chore(con_obj, "test")

## End(Not run)
```

**tm1_run_process**

**TM1 Run a Process**

**Description**

Runs a process

**Usage**

```
tm1_run_process(tm1_connection, process, par1name, par1value, par2name, par2value, par3name, par3value)
```
tm1_send_data

Arguments

tm1_connection tm1 connection object returned by the function tm1_connection
process Name of a process as a string
par1name Name of a parameter
par1value Value of a parameter
par2name Name of a parameter
par2value Value of a parameter
par3name Name of a parameter
par3value Value of a parameter

Examples

## Not run:
tm1_run_process(tm1_connection("localhost", "8881", "admin", "apple"), "test")
con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_run_process(con_obj, "test")

## End(Not run)

---

tm1_send_data TM1 Send Data to a Cube

Description

Send data to a cube, Supports up-to 10 dimension for now

Usage

tm1_send_data(tm1_connection, value, cube,
             element1, element2, element3, element4, element5,
             element6, element7, element8, element9, element10,
             increment)

Arguments

tm1_connection tm1 connection object returned by the function tm1_connection
value data value you want to send to cube
cube Name of a cube as a string
element1 Element from 1st dimension of cube. Leave empty if there is no dimension
element2 Element from 2nd dimension of cube. Leave empty if there is no dimension
element3 Element from 3rd dimension of cube. Leave empty if there is no dimension
element4 Element from 4th dimension of cube. Leave empty if there is no dimension
element5  Element from 5th dimension of cube. Leave empty if there is no dimension
element6  Element from 6th dimension of cube. Leave empty if there is no dimension
element7  Element from 7th dimension of cube. Leave empty if there is no dimension
element8  Element from 8th dimension of cube. Leave empty if there is no dimension
element9  Element from 9th dimension of cube. Leave empty if there is no dimension
element10 Element from 10th dimension of cube. Leave empty if there is no dimension
increment If TRUE, it will increment cube data by Value. If False, it will replace. This
          parameter is ignored in sending string values.

Examples

## Not run:
tm1_send_data(
  tm1_connection("localhost", "8881", "admin", "apple"),
  10,
  "SalesCube",
  "Actual", "Argentina", "S Series 1.8 L Sedan", "Units", "Jan")

con_obj <- tm1_connection("localhost", "8881", "admin", "apple")
tm1_send_data(con_obj,
  10,
  "SalesCube",
  "Actual", "Argentina", "S Series 1.8 L Sedan", "Units", "Jan",
  increment=TRUE)

## End(Not run)
Arguments

- `tm1_connection` tm1 connection object returned by the function `tm1_connection`
- `valueset` data frame or matrix object holding values you want to send to cube
- `cube` Name of a cube as a string
- `rowdim` Corresponding dimension of the elements on row
- `coldim` Corresponding dimension of the elements on column
- `titledim1` Name of dimension in title
- `titleel1` Element of dimension in corresponding titledim
- `titledim2` Name of dimension in title
- `titleel2` Element of dimension in corresponding titledim
- `titledim3` Name of dimension in title
- `titleel3` Element of dimension in corresponding titledim
- `titledim4` Name of dimension in title
- `titleel4` Element of dimension in corresponding titledim
- `titledim5` Name of dimension in title
- `titleel5` Element of dimension in corresponding titledim
- `titledim6` Name of dimension in title
- `titleel6` Element of dimension in corresponding titledim
- `titledim7` Name of dimension in title
- `titleel7` Element of dimension in corresponding titledim
- `titledim8` Name of dimension in title
- `titleel8` Element of dimension in corresponding titledim

Examples

```r
## Not run:
sdata <- tm1_connection("localhost", "8881", "admin", "apple")

#valueset
# Argentina Brazil
#Jan 1 2
#Feb 3 4
tm1_send_dataset(
sdata,
valueset = valueset, cube = "SalesCube",
rowdim = "month", coldim = "region",
titledim1 = "actvsbud", titleel1 = "Actual",
titledim2 = "model", titleel2 = "L Series 1.6 L Convertible",
titledim3 = "account1", titleel3 = "Units")

## End(Not run)
```
Index

tm1_api_request, 2
tm1_connection, 3
tm1_create_element, 4
tm1_create_mdx, 4
tm1_create_subset, 7
tm1_create_view, 7
tm1_delete_element, 8
tm1_delete_subset, 9
tm1_delete_view, 10
tm1_get_config, 10
tm1_get_cube_dimensions, 11
tm1_get_cubes, 11
tm1_get_data, 12
tm1_get_dimension_attributes, 14
tm1_get_dimension_elements, 14
tm1_get_dimension_subsets, 15
tm1_get_dimensions, 13
tm1_get_element, 16
tm1_get_instances, 16
tm1_get_log, 17
tm1_get_mdx_view, 18
tm1_get_native_view, 19
tm1_get_subset_elements, 19
tm1_logout, 20
tm1_run_chore, 21
tm1_run_process, 21
tm1_send_data, 22
tm1_send_dataset, 23