Package 'GMQL'

May 19, 2017

Type Package

Title GMQL function

Version 0.3.0

Author Simone Pallotta <simone.pallotta@mail.polimi.it>

Maintainer Polimi <yourself@somewhere.net>

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Depends R(>= 2.7.0), rscala(<= 1.0.15), httr(>= 1.2.1),
rtracklayer(<= 1.34.2), GenomicRanges(<= 1.26.4), xml2(<= 1.1.1)

VignetteBuilder knitr

Suggests BiocStyle, knitr, rmarkdown

R topics documented:

`compileQuery` *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
compileQuery(url, query)
```

Arguments

`username`	user name
`password`	user password

Details

si predilige il testo della query

Value

authentication token

cover *GMQL Operation: COVER*

Description

responds to the need of computing properties that reflect region's intersections, the operation with no grouping produces a single output sample, and all the metadata attributes of the contributing input samples in result dataset are assigned to the resulting single sample in input dataset Regions of the result sample are built from the regions of samples in input dataset according to the following condition:

Usage

```
cover(minAcc, maxAcc, groupBy = NULL, aggregates = NULL, input_data)
```

Details

Each resulting region r in input dataset is the contiguous intersection of at least minAcc and at most maxAcc contributing regions in the samples of result dataset; minAcc and maxAcc are called accumulation indexes.

Resulting regions may have new attributes, calculated by means of aggregate expressions over the attributes of the contributing regions. Jaccard Indexes are standard measures of similarity of the contributing regions, added as default region attributes. with grouping define instead, the samples are partitioned by groups, each with distinct values of grouping metadata attributes (i.e., homonym attributes in the operand schemas) and the cover operation is separately applied to each group, yielding to one sample in the result for each group

deleteDataset *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
deleteDataset(url = "http://genomic.elet.polimi.it/gmql-rest/datasets",
  datasetName)
```

difference *GMQL Operation: DIFFERENCE*

Description

produces one sample in the result for each sample of the left operand, by keeping the same metadata of the left operand sample and only those regions (with their schema and values) of the left operand sample which do not intersect with any region in the right operand sample. The optional joinby clause is used to extract a subset of couples from the cartesian product of two dataset Dleft x Dright on which to apply the DIFFERENCE operator: only those samples that have the same value for each attribute are considered when performing the difference.

Usage

```
difference(joinBy = NULL, left_input_data, right_input_data)
```

Arguments

joinBy	list of CONDITION object using metadata as value. The CONDITION available are EXACT,FULLNAME,DEFAULT. Every condition accept a string value.
left_input_data	string pointer taken from GMQL function
right_input_data	string pointer taken from GMQL function

Examples

```
## Not run:
startGMQL()
path = /.../dataset_name
r = read(path)
c = cover(2,3,input_data = r)
s = select("NOT(Patient_age < 70 AND provider=='Polimi')",input_dat = r)
d = difference(left_input_data = r, right_input_data = c)
d = difference(list(DEFAULT("antibody_target")),left_input_data = r, right_input_data = c)

## End(Not run)
```

downloadDataset *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
downloadDataset(datasetName, path = getwd())
```

execute *GMQL Function: EXECUTE*

Description

execute GMQL query. The function works only after invoking at least one materialize

Usage

```
execute()
```

Examples

```
startGMQL()
r = read(path)
r2 = read(path2)
s = select(input_data = r)
m = merge(groupBy = c("antibody_targer","cell_karyotype"),input_data = s)
materialize(input_data = m, dir_out = "/.../foldername")
materialize(s,"/.../foldername")
execute()
```

exportGMQL.gdm *Create GMQL dataset from Granges or GrangesList*

Description

Create GMQL dataset from Granges or GrangesList

Usage

```
exportGMQL.gdm(..., dir_out)
```

Arguments

...	set of Granges or a single GrangesList
dir_out	folder path where create a folder and write all the sample files

exportGMQL.gtf *Create GMQL dataset from Granges or GrangesList*

Description

Create GMQL dataset from Granges or GrangesList

Usage

```
exportGMQL.gtf(..., dir_out)
```

Arguments

...	set of Granges or a single GrangesList
dir_out	folder path where create a folder and write all the sample files

extend *GMQL Operation: EXTEND*

Description

It generates new metadata attributes as result of aggregate functions applied to sample region attributes and adds them to the existing metadata of the sample aggregate functions are applied sample by sample.

Usage

```
extend(metadata = NULL, input_data)
```

Arguments

metadata	a list of element key = value. value is an object of class OPERATOR. The functions aggregate available for extend function are: MIN,MAX,SUM,BAG,AVG,COUNT. Every operator accept a string value. only COUNT cannot have a value The key of list is mandatory; if all missed we create that based on function you choose
input_data	url-like "string" pointer returned from GMQL function

Examples

```
startGMQL()
path = /.../dataset_name
r = read(path)
e = extend(somma = SUM("pvalue"),c = COUNT(), m = AVG("score"),input_data = r)
```

flat *GMQL Operation: FLAT*

Description

returns the contiguous region that starts from the first end and stops at the last end of the regions which would contribute to each region of the COVER

Usage

```
flat(minAcc, maxAcc, groupBy = NULL, aggregates = NULL, input_data)
```

GMQLlogin *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
GMQLlogin(url, username = NULL, password = NULL)
```

Arguments

url	server address
username	user name
password	user password

Value

autentication Token

Examples

```
GMQLlogin(url = http://...../...)
GMQLlogin(url, username="pippo",password="baudo")
```

GMQLlogout *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
GMQLlogout(url)
```

Arguments

url	server address

Examples

```
GMQLlogin(url = http://...../...)
GMQLlogin(url, username="pippo",password="baudo")
```

GMQLregister *GMQL API web Service*

Description

Register to web Service Provider

Usage

```
GMQLregister(url, name, lastname, mail, username, password)
```

Arguments

url	server address
name	user name
lastname	user lastname
mail	user mail
username	user username
password	user password

Value

autentication Token

Examples

```
GMQLregister(url = http://...../...)
```

group *GMQL Operation: GROUP*

Description

GMQL Operation: GROUP

Usage

```
group()
```

histogram *GMQL Operation: HISTOGRAM*

Description

returns the non-overlapping regions contributing to the cover, each with its accumulation index value, which is assigned to the AccIndex region attribute.

Usage

```
histogram(minAcc, maxAcc, groupBy = NULL, aggregates = NULL, input_data)
```

importGMQL.gdm *Create GrangesList from dataset GMQL in GDM (delimited / tabulated) format file*

Description

Create GrangesList from dataset GMQL in GDM (delimited / tabulated) format file

Usage

```
importGMQL.gdm(datasetName = "/Users/simone/Downloads/DATA_SET_VAR_GDM")
```

Arguments

datasetName	dataset folder path

Examples

```
path <- "/..../..../..../DATA_SET_VAR_GDM"
grl <- importGMQL.gdm(path)
```

importGMQL.gtf *Create GrangesList from dataset GMQL in GTF format file*

Description

Create GrangesList from dataset GMQL in GTF format file

Usage

```
importGMQL.gtf(datasetName = "/Users/simone/Downloads/DATA_SET_VAR_GTF")
```

Arguments

datasetName	dataset folder path

Examples

```
path <- "/..../..../..../DATA_SET_VAR_GTF"
grl <- importGMQL.gtf(path)
```

materialize *GMQL Operation: MATERIALIZE*

Description

It saved the content of a dataset,whose name can be specified, that contains samples metadata and samples regions. To preserve the content of any dataset generated during a GMQL query, the dataset must be materialized. Any dataset can be materialized, however the operation is time expensive; for best performance, materialize the relevant data only.

Usage

```
materialize(input_data, dir_out)
```

Arguments

input_data	string pointer taken from GMQL function
dir_out	out path folder for default is working directory

Examples

```
startGMQL()
r = read(path)
r2 = read(path2)
s = select(input_data = r)
m = merge(groupBy = c("antibody_targer","cell_karyotype"),input_data = s)
materialize(input_data = m, dir_out = "/.../foldername")
materialize(s,"/.../foldername")
```

merge *GMQL Operation: MERGE*

Description

It builds a dataset consisting of a single sample having as regions all the regions of the input data and as metadata the union of all the attribute-values of the input samples. A groupby clause can be specified on metadata:the samples are then partitioned in groups, each with a distinct value of the grouping metadata attributes, and the operation is separately applied to each group, yielding to one sample in the result for each group. Samples without the grouping metadata attributes are disregarded.

Usage

```
merge(groupBy = NULL, input_data)
```

Arguments

input_data	url-like string "pointer" taken from GMQL function
metadata	a vector of metadata as string

Examples

```
startGMQL()
r = read(path)
s = select(input_data = r)
m = merge(groupBy = c("antibody_targer","cell_karyotype"),input_data = r)
m = merge(c("antibody_targer","cell_karyotype"),s)
m = merge(input_data = s)
```

metadataFromSample *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
metadataFromSample(url = "http://genomic.elet.polimi.it/gmql-rest/datasets/",
  datasetName, sampleName)
```

order *GMQL operation: ORDER*

Description

GMQL operation: ORDER

Usage

```
order(metadata_order = NULL, mtop = 0, mtopg = 0, regions_order = NULL,
  rtop = 0, rtopg = 0, input_data)
```

Arguments

metadata_order	list of ORDER object. the possbililty are ASC or DESC, every order class accept one string
mtop	0 is default meand that we not use it
mtopg	0 is default meand that we not use it
regions_order	list of ORDER object. the possbililty are ASC or DESC, every order class accept one string
rtop	0 is default meand that we not use it
rtopg	0 is default meand that we not use it
input_data	url-like "string" pointer taken from GMQL function

Examples

```
startGMQL()
path = /.../dataset_name
r = read(path)
c = cover(2,3,input_data = r)
s = select("NOT(Patient_age < 70 AND provider=='Polimi')",input_dat = r)
s = select("NOT(Patient_age < 70)",region_predicate = "NOT(qValue > 0.001)",
semi_join = list(EXACT("cell_type"),EXACT("age")),semi_join_dataset = c,input_data = r )

o = order(DESC(Region_Count), mtop = 2, input_data = s)
o = order(list(DESC(Region_Count)),regions_order = list(DESC(MutationCount),ASC(pvalue)),
mtop = 5,rtopg = 1, input_data = c)
```

project *GMQL Operation: PROJECT*

Description

GMQL Operation: PROJECT

Usage

```
project(metadata = NULL, regions = NULL, input_data)
```

Arguments

input_data	string pointer taken from GMQL function
predicate	string made up by logical oepration: AND,OR,NOT
region	region
semijoin	semijoin

read *GMQL Function: READ*

Description

Read a GMQL dataset from disk

Usage

```
read(DatasetPathFolder)
```

Arguments

DatasetPathFolder	folder path for GMQL dataset

Value

url-like string "pointer" to dataset

Examples

```
startGMQL()
path = /.../dataset_name
r = read(path)
```

regionFromSample *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
regionFromSample(url = "http://genomic.elet.polimi.it/gmql-rest/datasets/",
  datasetName, sampleName)
```

Details

start.time <- Sys.time() end.time <- Sys.time() time.taken <- end.time - start.time

runQuery *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
runQuery(url, fileName, query, output_gtf = T)
```

Arguments

username	user name
password	user password

Details

si predilige il testo della query

Value

authentication token

saveQuery *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
saveQuery(url, queryName, queryTxt)
```

Arguments

url	server address
queryName	query name
queryTxt	query text

select *GMQL Operation: SELECT*

Description

Extract a subset of samples from the input dataset. It returns all the samples which satisfy the predicate on metadata and / or returns those regions which satisfy the predicate on regions. Also semijoin clause are used to further select samples; When semijoin is defined it extract, based on the existence of certain metadata attributes defined in semijoin clause, those sample that are associated with at least one sample in an semijoin dataset

Usage

```
select(predicate = NULL, region_predicate = NULL, semi_join = NULL,
  semi_join_dataset = NULL, input_data)
```

Arguments

predicate	string predicate made up by logical oepration: AND,OR,NOT on metadata values
region_predicate	string predicate made up by logical oepration: AND,OR,NOT on schema region values
semi_join_dataset	url-like "string" pointer taken from GMQL function used in semijoin
input_data	url-like "string" pointer taken from GMQL function
semijoin	list of CONDITION object using metadata as value. The CONDITION available are EXACT,FULLNAME,DEFAULT. Every condition accept a string value.

Examples

```
startGMQL()
path = /.../dataset_name
r = read(path)
c = cover(2,3,input_data = r)
s = select("NOT(Patient_age < 70 AND provider=='Polimi')",input_dat = r)
s = select("NOT(Patient_age < 70)",region_predicate = "NOT(variant_type == 'SNP' OR pValue < 0.01)",
semi_join = list(DEFAULT("cell_type"),FULLNAME("age")),semi_join_dataset = c,input_data = r )
s = select("NOT(Patient_age < 70)",region_predicate = "NOT(qValue > 0.001)",
semi_join = list(EXACT("cell_type"),EXACT("age")),semi_join_dataset = c,input_data = r )
```

---

showDatasets *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showDatasets(url = "http://genomic.elet.polimi.it/gmql-rest/datasets")
```

---

showJobLog *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showJobLog(url, job_id)
```

Arguments

username	user name
password	user password

showJobs *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showJobs(url, dataset_name)
```

Arguments

username	user name
password	user password

showQueries *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showQueries(url)
```

Arguments

url	server address

showSamplesFromDataset *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showSamplesFromDataset(url = "http://genomic.elet.polimi.it/gmql-rest/datasets/",
datasetName)
```

showSchemaFromDataset *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
showSchemaFromDataset(datasetName)
```

startGMQL *start GMQL Server*

Description

Set and run GMQL server for executing GMQL query

Usage

```
startGMQL()
```

Examples

```
startGMQL()
```

stopJob *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
stopJob(url, job_id)
```

Arguments

username	user name
password	user password

summit *GMQL Operation: SUMMIT*

Description

returns regions that start from a position where the number of intersecting regions is not increasing afterwards and stops at a position where either the number of intersecting regions decreases, or it violates the max accumulation index).

Usage

```
summit(minAcc, maxAcc, groupBy = NULL, aggregates = NULL, input_data)
```

traceJob *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
traceJob(url, job_id)
```

Arguments

username	user name
password	user password

union *GMQL Operation: UNION*

Description

It is used to integrate homogeneous or heterogeneous samples of two datasets within a single dataset for each sample of either one of the input datasets, a sample is created in the result as follow its metadata are the same as in the original sample. Its regions are the same (in coordinates and attribute values) as in the original sample. Region attributes which are missing in an input dataset sample (w.r.t. themerged schema) are set to null. its schema is the schema of the first (left) input dataset (more properly, it will be the merging of the schemas of the two input datasets) ; new identifiers are assigned to each output sample; The merging of two schemas is performed by projecting the schema of the right dataset over the schema of the left one two region attributes are considered identical if they have the same name and type. For what concerns metadata, attributes of samples from the left (right) input dataset are prefixed with the strings LEFT (RIGHT), so as to trace the dataset to which they originally belonged.

Usage

```
union(left_input_data, right_input_data)
```

Arguments

left_input_data	url-like "string" pointer taken from GMQL function
right_input_data	url-like "string" pointer taken from GMQL function

Examples

```
r = read(path)
r2 = read(path2)
c = cover(2,3,input_data = r)
u = union(r2,c)
```

---

uploadSamples *GMQL API web Service*

Description

Allow access to web service GMQL as guest or registered user with username and password

Usage

```
uploadSamples(url = "http://genomic.elet.polimi.it/gmql-rest/datasets",
  datasetName, schemaName = NULL, ...)
```

Index