

# Package ‘epigraphdb’

July 22, 2025

**Title** Interface Package for the 'EpiGraphDB' Platform

**Version** 0.2.3

**Description** The interface package to access data from the 'EpiGraphDB' <<https://epigraphdb.org>> platform. It provides easy access to the 'EpiGraphDB' platform with functions that query the corresponding REST endpoints on the API <<https://api.epigraphdb.org>> and return the response data in the 'tibble' data frame format.

**URL** <https://mrcieu.github.io/epigraphdb-r/>

**BugReports** <https://github.com/MRCIEU/epigraphdb-r/issues>

**License** GPL-3

**Suggests** testthat, roxygen2, knitr, rmarkdown, spelling, devtools, usethis, pkgdown, styler, lintr, covr, igraph, gtools, stringr, dplyr, ggplot2

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Imports** magrittr, tibble, httr, glue, purrr, jsonlite

**VignetteBuilder** knitr

**Language** en-GB

**NeedsCompilation** no

**Author** Yi Liu [cre, aut],  
Valeria Haberland [aut],  
Marina Vabistsevits [aut],  
Tom Gaunt [aut],  
MRC IEU [cph]

**Maintainer** Yi Liu <[yi6240.liu@bristol.ac.uk](mailto:yi6240.liu@bristol.ac.uk)>

**Repository** CRAN

**Date/Publication** 2022-01-14 19:52:45 UTC

## Contents

confounder . . . . .	2
cypher . . . . .	3
drugs_risk_factors . . . . .	4
genetic_cor . . . . .	5
literature_gwas . . . . .	6
mappings_gene_to_protein . . . . .	6
meta_nodes_list . . . . .	7
meta_nodes_list_node . . . . .	8
meta_nodes_search_node . . . . .	9
meta_rels_list . . . . .	10
meta_rels_list_rel . . . . .	11
mr . . . . .	11
obs_cor . . . . .	13
ontology_gwas_efo . . . . .	14
pathway . . . . .	15
pqtl . . . . .	15
pqtl_list . . . . .	17
pqtl_pleio . . . . .	18
protein_in_pathway . . . . .	19
query_epigraphdb . . . . .	19
xqtl_multi_snp_mr . . . . .	21
xqtl_single_snp_mr . . . . .	22
<b>Index</b>	<b>24</b>

---

confounder	<i>MR evidence on confounding traits between exposure and outcome</i>
------------	---

---

## Description

[GET /confounder](#)

## Usage

```
confounder(
  exposure_trait = NULL,
  outcome_trait = NULL,
  type = c("confounder", "intermediate", "reverse_intermediate", "collider"),
  pval_threshold = 1e-05,
  mode = c("table", "raw")
)
```

**Arguments**

exposure_trait	A trait name, e.g. "Body mass index", leaving exposure_trait as NULL will return MR information related to a specific outcome. <b>NOTE:</b> exposure_trait and outcome_trait cannot be both NULL.
outcome_trait	A trait name, e.g. "Coronary heart disease", leaving outcome_trait as NULL will return MR information related to a specific exposure_trait. <b>NOTE:</b> exposure_trait and outcome_trait cannot be both NULL.
type	One in ["confounder", "intermediate", "reverse_intermediate", "collider"] Refer to <a href="#">the confounder view in web application</a> for details
pval_threshold	P-value threshold
mode	If mode = "table", returns a data frame (a <a href="#">tibble</a> as per <a href="#">tidyverse</a> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <a href="#">httr</a> .

**Value**

Data from GET /confounder

**Examples**

```
## Not run:
confounder(exposure_trait = "Body mass index", outcome_trait = "Coronary heart disease")

## End(Not run)
```

---

cypher *Send a query in Cypher to EpiGraphDB*

---

**Description**

**NOTE:** this function is intended for advanced uses. Regular users are recommended to use standard query functions

**Usage**

```
cypher(query, mode = c("table", "raw"))
```

**Arguments**

query	A Cypher query.
mode	If mode = "table", returns a data frame (a <a href="#">tibble</a> as per <a href="#">tidyverse</a> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <a href="#">httr</a> .

## Examples

```
## Not run:  
cypher("MATCH (n:Gwas) RETURN n LIMIT 2")  
  
## End(Not run)
```

---

drugs\_risk\_factors      *Drugs for risk factors*

---

## Description

`GET /drugs/risk-factors`

## Usage

```
drugs_risk_factors(trait, pval_threshold = 1e-08, mode = c("table", "raw"))
```

## Arguments

trait	A trait name
pval_threshold	P-value threshold
mode	If mode = "table", returns a data frame (a <code>tibble</code> as per <code>tidyverse</code> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <code>httr</code> .

## Value

Data from `GET /drugs/risk-factors`

## Examples

```
## Not run:  
drugs_risk_factors(trait = "Body mass index")  
  
## End(Not run)
```

---

genetic_cor	<i>Genetic correlations between traits</i>
-------------	--

---

## Description

GET /genetic-cor

## Usage

```
genetic_cor(trait, cor_coef_threshold = 0.8, mode = c("table", "raw"))
```

## Arguments

trait	name of the trait, e.g. "body mass index"
cor_coef_threshold	correlation coefficient threshold
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from GET /genetic\_cor

## Examples

```
## Not run:
genetic_cor(trait = "Body mass index") %>%
  dplyr::glimpse()

## End(Not run)

# Use a different threshold
## Not run:
genetic_cor(trait = "Body mass index", cor_coef_threshold = 0.4) %>%
  dplyr::glimpse()

## End(Not run)
```

---

literature_gwas	<i>Literature evidence regarding a GWAS trait</i>
-----------------	---

---

**Description**

GET /literature/gwas

**Usage**

```
literature_gwas(trait, semmed_predicate = NULL, mode = c("table", "raw"))
```

**Arguments**

trait	A trait name
semmed_predicate	Either NULL which returns entries from all predicates, or a SemMed predicate e.g. "DIAGNOSES" or "ASSOCIATED_WITH"
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

**Value**

Data from GET /literature/gwas

**Examples**

```
## Not run:
literature_gwas(trait = "Body mass index")

## End(Not run)
```

---

mappings_gene_to_protein	<i>Return protein uniprot_id from associated genes</i>
--------------------------	--

---

**Description**

POST /mappings/gene-to-protein

**Usage**

```
mappings_gene_to_protein(
  gene_name_list = NULL,
  gene_id_list = NULL,
  by_gene_id = FALSE,
  mode = c("table", "raw")
)
```

**Arguments**

gene\_name\_list List of HGNC symbols of the genes (default)  
 gene\_id\_list List of Ensembl gene IDs (when by\_gene\_id == TRUE)  
 by\_gene\_id Search for gene ids (Ensembl gene IDs) instead of gene names (HGNC symbols)  
 mode If mode = "table", returns a data frame (a **tibble** as per **tidyverse** convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by **httr**.

**Value**

Data from POST /mappings/gene-to-protein

**Examples**

```
# By HGNC symbols
## Not run:
mappings_gene_to_protein(gene_name_list = c("GCH1", "MYOF"))

## End(Not run)

# By Ensembl Ids
## Not run:
mappings_gene_to_protein(gene_id_list = c("ENSG00000162594", "ENSG00000113302"), by_gene_id = TRUE)

## End(Not run)
```

---

meta_nodes_list	<i>List meta nodes (e.g. Gwas, Gene, etc.)</i>
-----------------	--

---

**Description**

**GET /meta/nodes/list**

**Usage**

```
meta_nodes_list(mode = c("raw"))
```

**Arguments**

mode If mode = "table", returns a data frame (a **tibble** as per **tidyverse** convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by **httr**.

**Value**

Data from GET /meta/nodes/list

**Examples**

```
## Not run:
meta_nodes_list()

## End(Not run)
```

---

meta\_nodes\_list\_node *List nodes under a meta node*

---

**Description**

GET /meta/nodes/{meta\_node}/list

**Usage**

```
meta_nodes_list_node(
  meta_node,
  full_data = TRUE,
  limit = 10,
  offset = 0,
  mode = c("table", "raw")
)
```

**Arguments**

meta_node	Name of a meta node (e.g. Gwas). Use meta_nodes_list to get the full list of meta nodes.
full_data	When False, only return the id and name fields (their specific names differ in specific nodes) for a node. This is useful if you want your queries to return results faster with smaller amount of data requested.
limit	Max number of items to retrieve.
offset	Number of items to skip. Use limit and offset in combination to do pagination.
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

**Value**

Data from GET /meta/nodes/{meta\_node}/list



**Examples**

```
# List the first 5 Gwas nodes, with only id and name fields
## Not run:
meta_nodes_list_node(meta_node = "Gwas", full_data = FALSE, limit = 5)

## End(Not run)

# List the 6th - 10th Disease nodes, with full properties
## Not run:
meta_nodes_list_node(meta_node = "Disease", full_data = TRUE, limit = 5, offset = 0)

## End(Not run)
```

---

meta\_nodes\_search\_node

*Search a node by its id field, or its name field*

---

**Description**

[GET /meta/nodes/{meta\\_node}/search](#)

**Usage**

```
meta_nodes_search_node(
  meta_node,
  id = NULL,
  name = NULL,
  limit = 10,
  full_data = TRUE,
  mode = c("table", "raw")
)
```

**Arguments**

meta_node	Name of a meta node (e.g. Gwas). Use meta_nodes_list to get the full list of meta nodes.
id	The id field of a node (e.g. "ieu-a-2" for a Gwas). Use EpiGraphDB web UI to get a sense of what those ids are for entities.
name	The name field of a node (e.g. "body mass index" for a Gwas). Use EpiGraphDB web UI to get a sense of what those names are for entities.
limit	Max number of items to retrieve.
full_data	When False, only return the id and name fields (their specific names differ in specific nodes) for a node. This is useful if you want your queries to return results faster with smaller amount of data requested.
mode	If mode = "table", returns a data frame (a <a href="#">tibble</a> as per <a href="#">tidyverse</a> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <a href="#">httr</a> .

**Value**

Data from GET /meta/nodes/{meta\_node}/search

**Examples**

```
# Search Gwas nodes
## Not run:
meta_nodes_search_node(meta_node = "Gwas", name = "body mass index")

## End(Not run)
```

---

meta_rels_list	<i>List meta rels (e.g. MR, etc.)</i>
----------------	---------------------------------------

---

**Description**

`GET /meta/rels/list`

**Usage**

```
meta_rels_list(mode = c("raw"))
```

**Arguments**

mode	If mode = "table", returns a data frame (a <code>tibble</code> as per <code>tidyverse</code> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <code>httr</code> .
------	--

**Value**

Data from GET /meta/rels/list

**Examples**

```
## Not run:
meta_rels_list()

## End(Not run)
```

---

meta\_rels\_list\_rel      *List relationships under a meta relationship*

---

## Description

`GET /meta/rels/{meta_rel}/list`

## Usage

```
meta_rels_list_rel(meta_rel, limit = 10, offset = 0, mode = c("table", "raw"))
```

## Arguments

meta_rel	Name of a meta relationship (e.g. MR). Use meta_rels_list to get the full list of meta relationships.
limit	Max number of items to retrieve.
offset	Number of items to skip. Use limit and offset in combination to do pagination.
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from `GET /meta/rels/{meta_rel}/list`

## Examples

```
# List the first 5 MR relationships
## Not run:
meta_rels_list_rel(meta_rel = "MR_EVE_MR", limit = 5)

## End(Not run)
```

---

mr      *Return information related to Mendelian Randomisation*

---

## Description

`GET /mr`

**Usage**

```
mr(
  exposure_trait = NULL,
  outcome_trait = NULL,
  pval_threshold = 1e-05,
  mode = c("table", "raw")
)
```

**Arguments**

`exposure_trait` A trait name, e.g. "Body mass index", leaving `exposure_trait` as NULL will return MR information related to a specific outcome. **NOTE:** `exposure_trait` and `outcome_trait` cannot be both NULL.

`outcome_trait` A trait name, e.g. "Coronary heart disease", leaving `outcome_trait` as NULL will return MR information related to a specific `exposure_trait`. **NOTE:** `exposure_trait` and `outcome_trait` cannot be both NULL.

`pval_threshold` P-value threshold

`mode` If `mode = "table"`, returns a data frame (a `tibble` as per `tidyverse` convention). If `mode = "raw"`, returns a raw response from EpiGraphDB API with minimal parsing done by `httr`.

**Value**

Data from GET /mr

**Examples**

```
# Returns a data frame
## Not run:
mr(exposure_trait = "Body mass index", outcome_trait = "Coronary heart disease")

## End(Not run)

# Returns raw response
## Not run:
mr(
  exposure_trait = "Body mass index", outcome_trait = "Coronary heart disease",
  mode = "raw"
) %>% str()

## End(Not run)

# Use a different threshold
## Not run:
mr(exposure_trait = "Body mass index", pval_threshold = 1e-8)

## End(Not run)
```

---

obs_cor	<i>Observational correlations between traits</i>
---------	--

---

## Description

GET /obs-cor

## Usage

```
obs_cor(trait, cor_coef_threshold = 0.8, mode = c("table", "raw"))
```

## Arguments

trait	name of the trait, e.g. "body mass index"
cor_coef_threshold	correlation coefficient threshold
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from GET /obs-cor

## Examples

```
## Not run:
obs_cor(trait = "Body mass index (BMI)") %>%
  dplyr::glimpse()

## End(Not run)

# Use a different threshold
## Not run:
obs_cor(trait = "Body mass index (BMI)", cor_coef_threshold = 0.8) %>%
  dplyr::glimpse()

## End(Not run)
```

---

ontology\_gwas\_efo      *Ontology association between EFO term and Gwas*

---

## Description

GET /ontology/gwas-efo

## Usage

```
ontology_gwas_efo(  
  trait = NULL,  
  efo_term = NULL,  
  fuzzy = TRUE,  
  mode = c("table", "raw")  
)
```

## Arguments

trait	trait name, e.g. "body mass"
efo_term	EFO term, e.g. "systolic blood pressure"
fuzzy	whether query with exact matching (FALSE) or fuzzy matching (default, TRUE)
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from GET /ontology/gwas-efo

## Examples

```
## Not run:  
ontology_gwas_efo(trait = "blood", fuzzy = FALSE)  
  
## End(Not run)  
  
## Not run:  
ontology_gwas_efo(efo_term = "blood pressure", fuzzy = FALSE)  
  
## End(Not run)
```

---

pathway	<i>Pathway evidence</i>
---------	-------------------------

---

**Description**

`GET /pathway`

**Usage**

```
pathway(trait, pval_threshold = 1e-05, mode = c("table", "raw"))
```

**Arguments**

trait	A trait name
pval_threshold	P-value threshold
mode	If mode = "table", returns a data frame (a <code>tibble</code> as per <code>tidyverse</code> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <code>httr</code> .

**Value**

Data from GET /pathway

**Examples**

```
## Not run:  
pathway(trait = "Body mass index")  
  
## End(Not run)
```

---

pqtl	<i>Return information related to the pQTL analysis</i>
------	--

---

**Description**

`GET /pqtl/`

**Usage**

```
pqtl(  
  query,  
  rtype = c("mrres", "simple", "sglmr", "inst", "sense"),  
  pvalue = 0.05,  
  searchflag = c("traits", "proteins"),  
  mode = c("table", "raw")  
)
```

**Arguments**

query	(Required) A protein coding gene name or a trait name, e.g. "ADAM19" or "Inflammatory bowel disease" which cannot be NULL.
rtype	(Optional) A type of data to be extracted, which can be one of these options: <ol style="list-style-type: none"> <li>1. simple: Basic summary</li> <li>2. mrres: MR results (DEFAULT)</li> <li>3. sglm: Single SNP MR results</li> <li>4. inst: SNP information</li> <li>5. sense: Sensitivity analysis <b>NOTE:</b> mrres is a DEFAULT option.</li> </ol>
pvalue	(Optional) A pvalue threshold for MR results with the DEFAULT set to 0.05. <b>NOTE:</b> this threshold applies to any rtype chosen.
searchflag	(Required) A flag to indicate whether you are searching for proteins or traits which cannot be NULL. If query is a protein name, then this flag should be "proteins"; if query is a trait, this flag should be "traits". <b>NOTE:</b> if the wrong flag is chosen for query, there will be no result returned.
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

**Value**

Data from GET /pqt1/

**Examples**

```
# Returns a data frame of MR results, while searching for proteins
## Not run:
pqt1(query = "ADAM19", searchflag = "proteins")

## End(Not run)

# Returns a data frame with SNP information, while searching for traits
## Not run:
pqt1(
  query = "Inflammatory bowel disease",
  rtype = "inst",
  searchflag = "traits"
)

## End(Not run)

# Change a pvalue threshold (the default is 0.05)
## Not run:
pqt1(
  query = "Inflammatory bowel disease",
  rtype = "inst",
  pvalue = 1.0,
  searchflag = "traits"
```



```

)

## End(Not run)

# Returns raw response if mode="raw"
## Not run:
pctl(
  query = "ADAM19", searchflag = "proteins",
  mode = "raw"
) %>% str()

## End(Not run)

```

---

pctl_list	<i>Return a list of all proteins/exposures or traits/outcomes available in the database</i>
-----------	---

---

## Description

`GET /pctl/list/`

## Usage

```
pctl_list(flag = c("exposures", "outcomes"), mode = c("table", "raw"))
```

## Arguments

flag	(Optional) A flag which indicates whether the list of exposures (if "exposures") or outcomes (if "outcomes") should be returned. The DEFAULT is "exposures".
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from `GET /pctl/list/`

## Examples

```

# Returns a list of available proteins (exposures)
## Not run:
pctl_list()

## End(Not run)

# Returns a list of available traits (outcomes)
## Not run:
pctl_list(flag = "outcomes")

## End(Not run)

```

---

`pqtl_pleio`*Return information related to the pleiotropy of SNPs*

---

## Description

```
GET /pqtl/pleio/
```

## Usage

```
pqtl_pleio(  
  rsid = NULL,  
  prflag = c("proteins", "count"),  
  mode = c("table", "raw")  
)
```

## Arguments

<code>rsid</code>	(Required) A SNP identified by rsID which cannot be NULL.
<code>prflag</code>	(Optional) A flag which determines whether the number (if "count") or names (if "proteins") of the associated proteins should be returned. The DEFAULT value is "proteins".
<code>mode</code>	(Optional) If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

## Value

Data from GET /pqtl/pleio/

## Examples

```
# Returns a data frame of associated proteins  
## Not run:  
pqtl_pleio(rsid = "rs1260326")  
  
## End(Not run)  
  
# Returns a number of associated proteins  
## Not run:  
pqtl_pleio(rsid = "rs1260326", prflag = "count")  
  
## End(Not run)
```

---

protein\_in\_pathway      *For the list of proteins, returns their associated pathway data*

---

### Description

POST /protein/in-pathway

### Usage

```
protein_in_pathway(uniprot_id_list, mode = c("table", "raw"))
```

### Arguments

uniprot\_id\_list      A list of protein UniProt IDs

mode                  If mode = "table", returns a data frame (a **tibble** as per **tidyverse** convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by **httr**.

### Value

Data from POST /protein/in-pathway

### Examples

```
## Not run:  
protein_in_pathway(uniprot_id_list = c("014933", "060674", "P32455"))  
  
## End(Not run)
```

---

query\_epigraphdb      *Send data request to an EpiGraphDB API endpoint*

---

### Description

This is a general purpose function to send data request which can be used when there has not been an R equivalent package function to an API endpoint. Underneath this is a wrapper around **httr** functions with better handling of returned status.

**Usage**

```
query_epigraphdb(
  route,
  params = NULL,
  mode = c("raw", "table"),
  method = c("GET", "POST"),
  retry_times = 3,
  retry_pause_min = 1
)
```

**Arguments**

route	An EpiGraphDB API endpoint route, e.g. <code>"/mr"</code> or <code>"/confounder"</code> . Consult the <a href="#">EpiGraphDB API documentation</a> .
params	A list of parameters associated with the query endpoint.
mode	<code>c("raw", "table")</code> , if <code>"table"</code> then the query handler will try to convert the returned data to a tibble dataframe. NOTE: The default mode is <code>"raw"</code> which will NOT convert the returned response to a dataframe. This is different to functions that query topic endpoints which default to return a dataframe. Explicitly specify <code>mode = "table"</code> when needed.
method	Type of HTTP (GET, POST, PUT, etc.) method. NOTE: When sending a POST request where a specific parameter is specified as a list on the API, and if the equivalent in R is a vector of length 1, you should wrap this parameter in <code>I()</code> , e.g. <code>I(c("APOE"))</code> to avoid auto unboxing. For details, please refer to <a href="#">httr::POST</a>
retry_times	Number of times the function will retry the request to the API.
retry_pause_min	Minimum number of seconds to wait for the next retry.

**Value**

Data from an EpiGraphDB API endpoint.

**Examples**

```
# GET /mr
# equivalent to `mr(exposure_trait = "Body mass index", outcome_trait = "Coronary heart disease")`
## Not run:
query_epigraphdb(
  route = "/mr",
  params = list(
    exposure_trait = "Body mass index",
    outcome_trait = "Coronary heart disease"
  ),
  mode = "table"
)

## End(Not run)
```

```
# GET /meta/nodes/Gwas/list
## Not run:
query_epigraphdb(
  route = "/meta/nodes/Gwas/list",
  params = list(
    limit = 5,
    offset = 0
  )
) %>% str(1)

## End(Not run)

# POST /protein/ppi
## Not run:
query_epigraphdb(
  route = "/protein/ppi",
  params = list(
    uniprot_id_list = c("P30793", "Q9NZM1", "O95236")
  ),
  method = "POST"
)

## End(Not run)

# error handling
## Not run:
tryCatch(
  query_epigraphdb(
    route = "/mr",
    params = list(
      exposure_trait = NULL,
      outcome_trait = NULL
    )
  ),
  retry_times = 0
),
error = function(e) {
  message(e)
}
)

## End(Not run)
```

---

xqtl\_multi\_snp\_mr

*Multi SNP QTL MR evidence*

---

## Description

GET /xqtl/multi-snp-mr

**Usage**

```
xqtl_multi_snp_mr(
  exposure_gene = NULL,
  outcome_trait = NULL,
  mr_method = c("IVW", "Egger"),
  qtl_type = c("eQTL", "pQTL"),
  pval_threshold = 1e-05,
  mode = c("table", "raw")
)
```

**Arguments**

exposure_gene	Name of the exposure gene
outcome_trait	Name of the outcome trait
mr_method	"IVW" or "Egger"
qtl_type	"eQTL" or "pQTL"
pval_threshold	P-value threshold
mode	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

**Value**

Data from GET /xqtl/multi-snp-mr

**Examples**

```
## Not run:
xqtl_multi_snp_mr(outcome_trait = "Coronary heart disease")

## End(Not run)
```

---

xqtl\_single\_snp\_mr      *Single SNP QTL MR evidence*

---

**Description**

GET /xqtl/single-snp-mr

**Usage**

```
xqtl_single_snp_mr(
  exposure_gene = NULL,
  outcome_trait = NULL,
  snp = NULL,
  qtl_type = c("eQTL", "pQTL"),
```

```
pval_threshold = 1e-05,  
mode = c("table", "raw")  
)
```

### Arguments

<code>exposure_gene</code>	Name of the exposure gene
<code>outcome_trait</code>	Name of the outcome trait
<code>snp</code>	SNP rsid
<code>qtl_type</code>	"eQTL" or "pQTL"
<code>pval_threshold</code>	P-value threshold
<code>mode</code>	If mode = "table", returns a data frame (a <b>tibble</b> as per <b>tidyverse</b> convention). If mode = "raw", returns a raw response from EpiGraphDB API with minimal parsing done by <b>httr</b> .

### Value

Data from GEET /xqtl/single-snp-mr

### Examples

```
## Not run:  
xqtl_single_snp_mr(outcome_trait = "Coronary heart disease")  
  
## End(Not run)
```

# Index

confounder, [2](#)  
cypher, [3](#)  
  
drugs\_risk\_factors, [4](#)  
  
genetic\_cor, [5](#)  
  
literature\_gwas, [6](#)  
  
mappings\_gene\_to\_protein, [6](#)  
meta\_nodes\_list, [7](#)  
meta\_nodes\_list\_node, [8](#)  
meta\_nodes\_search\_node, [9](#)  
meta\_rels\_list, [10](#)  
meta\_rels\_list\_rel, [11](#)  
mr, [11](#)  
  
obs\_cor, [13](#)  
ontology\_gwas\_efo, [14](#)  
  
pathway, [15](#)  
pqt1, [15](#)  
pqt1\_list, [17](#)  
pqt1\_pleio, [18](#)  
protein\_in\_pathway, [19](#)  
  
query\_epigraphdb, [19](#)  
  
xqtl\_multi\_snp\_mr, [21](#)  
xqtl\_single\_snp\_mr, [22](#)