

Package ‘gtexr’

May 8, 2026

Title Query the GTEx Portal API

Version 0.2.1

Description A convenient R interface to the Genotype-Tissue Expression (GTEx) Portal API. The GTEx project is a comprehensive public resource for studying tissue-specific gene expression and regulation in human tissues. Through systematic analysis of RNA sequencing data from 54 non-diseased tissue sites across nearly 1000 individuals, GTEx provides crucial insights into the relationship between genetic variation and gene expression. This data is accessible through the GTEx Portal API enabling programmatic access to human gene expression data. For more information on the API, see <https://gtexportal.org/api/v2/redoc>.

License MIT + file LICENSE

URL <https://docs.ropensci.org/gtexr/>,
<https://github.com/ropensci/gtexr>

BugReports <https://github.com/ropensci/gtexr/issues>

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Depends R (>= 4.2.0)

Imports cli, dplyr, httr2 (>= 1.0.0), purrr, rlang, tibble, tidyr

Suggests curl, httptest2, knitr, rmarkdown, spelling, stringr,
testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Alasdair Warwick [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-0800-2890>),
Benjamin Zuckerman [aut] (ORCID:
<https://orcid.org/0000-0002-0077-6074>),
Abraham Olvera-Barrios [aut] (ORCID:

<<https://orcid.org/0000-0002-3305-4465>>),
 Chuin Ying Ung [aut] (ORCID: <<https://orcid.org/0000-0001-8487-4589>>),
 Robert Luben [aut] (ORCID: <<https://orcid.org/0000-0002-5088-6343>>),
 Zhian N. Kamvar [rev] (ORCID: <<https://orcid.org/0000-0003-1458-7108>>)

Maintainer Alasdair Warwick <alasdair.warwick.19@ucl.ac.uk>

Repository CRAN

Date/Publication 2025-08-19 07:20:07 UTC

Contents

calculate_expression_quantitative_trait_loci	3
calculate_ieqtls	5
calculate_isqtls	7
calculate_splicing_quantitative_trait_loci	8
download	9
get_annotation	11
get_clustered_median_exon_expression	12
get_clustered_median_gene_expression	13
get_clustered_median_junction_expression	15
get_clustered_median_transcript_expression	16
get_collapsed_gene_model_exon	18
get_dataset_info	19
get_downloads_page_data	20
get_eqtl_genes	21
get_exons	23
get_expression_pca	24
get_file_list	25
get_fine_mapping	26
get_full_get_collapsed_gene_model_exon	28
get_functional_annotation	29
get_genes	30
get_gene_expression	31
get_gene_search	33
get_genomic_features	34
get_gwas_catalog_by_location	35
get_image	36
get_independent_eqtl	37
get_linkage_disequilibrium_by_variant_data	39
get_linkage_disequilibrium_data	40
get_maintenance_message	41
get_median_exon_expression	42
get_median_gene_expression	43
get_median_junction_expression	44
get_median_transcript_expression	46
get_multi_tissue_eqtls	47
get_neighbor_gene	49
get_news_item	50

get_sample_biobank_data	51
get_sample_datasets	53
get_service_info	55
get_significant_single_tissue_eqtls	56
get_significant_single_tissue_eqtls_by_location	57
get_significant_single_tissue_ieqtls	59
get_significant_single_tissue_isqtls	60
get_significant_single_tissue_sqtls	61
get_single_nucleus_gex	63
get_single_nucleus_gex_summary	64
get_sqtls_genes	66
get_subject	67
get_tissue_site_detail	68
get_top_expressed_genes	69
get_transcripts	71
get_variant	72
get_variant_by_location	73

Index	76
--------------	-----------

calculate_expression_quantitative_trait_loci
Calculate Expression Quantitative Trait Loci

Description

Calculate your own eQTLs

- This service calculates the gene-variant association for any given pair of gene and variant, which may or may not be significant.
- This requires as input a GENCODE ID, GTEx variant ID, and tissue site detail ID.

By default, the calculation is based on the latest GTEx release.

[GTEx Portal API documentation.](#)

Usage

```
calculate_expression_quantitative_trait_loci(
  tissueSiteDetailId,
  gencodeId,
  variantId,
  datasetId = "gtex_v8",
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or an Ontology ID.
gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Notes on output:

- Beta and standard error are recorded in columns `nes` and `error` respectively (see [GTEx FAQs](#))
- `variantId` contains (in order) chromosome, position, reference allele, alternative allele and human genome build separated by underscores. The reference and alternative alleles for "chr1_13550_G_A_b38" for example are "G" and "A" respectively.
- See examples for how to calculate minor and alternative allele frequencies.

Notes on input:

- Argument `variantId` also accepts RSIDs.

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Dynamic Association Endpoints: [calculate_ieqtls\(\)](#), [calculate_isqtls\(\)](#), [calculate_splicing_quantitative_trait_loci\(\)](#)

Examples

```
# perform request - returns a tibble with a single row
calculate_expression_quantitative_trait_loci(
  tissueSiteDetailId = "Whole_Blood",
  gencodeId = "ENSG00000203782.5",
  variantId = "rs79641866"
)

# unnest list columns with tidyr::unnest()
calculate_expression_quantitative_trait_loci(
  tissueSiteDetailId = "Whole_Blood",
  gencodeId = "ENSG00000203782.5",
  variantId = "rs79641866"
) |>
tidyr::unnest(c("data", "genotypes"))
```

```

# to calculate minor and alternative allele frequencies
calculate_expression_quantitative_trait_loci(
  tissueSiteDetailId = "Liver",
  gencodeId = "ENSG00000237973.1",
  variantId = "rs12119111"
) |>
dplyr::bind_rows(.id = "rsid") |>
tidyr::separate(
  col = "variantId",
  into = c(
    "chromosome",
    "position",
    "reference_allele",
    "alternative_allele",
    "genome_build"
  ),
  sep = "_"
) |>
# ...then ascertain alternative_allele frequency
dplyr::mutate(
  alt_allele_count = (2 * homoAltCount) + hetCount,
  total_allele_count = 2 * (homoAltCount + hetCount + homoRefCount),
  alternative_allele_frequency = alt_allele_count / total_allele_count
) |>
dplyr::select(
  rsid,
  beta = nes,
  se = error,
  pValue,
  minor_allele_frequency = maf,
  alternative_allele_frequency,
  chromosome:genome_build,
  tissueSiteDetailId
)

```

calculate_ieqtls

Calculate Ieqtls

Description

Calculate your own Cell Specific eQTLs.

- This service calculates the gene-variant association for any given pair of gene and variant, which may or may not be significant.
- This requires as input a GENCODE ID, GTEx variant ID, and tissue site detail ID.

By default, the calculation is based on the latest GTEx release.

[GTEx Portal API documentation.](#)

Usage

```
calculate_ieqtls(
  cellType,
  tissueSiteDetailId,
  gencodeId,
  variantId,
  datasetId = "gtex_v8",
  .return_raw = FALSE
)
```

Arguments

cellType	String. "Adipocytes", "Epithelial_cells", "Hepatocytes", "Keratinocytes", "Myocytes", "Neurons", "Neutrophils".
tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or an Ontology ID.
gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Dynamic Association Endpoints: [calculate_expression_quantitative_trait_loci\(\)](#), [calculate_isqtls\(\)](#), [calculate_splicing_quantitative_trait_loci\(\)](#)

Examples

```
# perform request
calculate_ieqtls(
  cellType = "Adipocytes",
  tissueSiteDetailId = "Adipose_Subcutaneous",
  gencodeId = "ENSG00000203782.5",
  variantId = "chr1_1099341_T_C_b38"
)
```

calculate_isqtls	<i>Calculate Isqtls</i>
------------------	-------------------------

Description

Calculate your own Cell Specific sQTLs.

- This service calculates the gene-variant association for any given pair of gene and variant, which may or may not be significant.
- This requires as input a GENCODE ID, GTEx variant ID, and tissue site detail ID.

By default, the calculation is based on the latest GTEx release.

[GTEx Portal API documentation](#).

Usage

```
calculate_isqtls(
  cellType,
  tissueSiteDetailId,
  phenotypeId,
  variantId,
  datasetId = "gtex_v8",
  .return_raw = FALSE
)
```

Arguments

cellType	String. "Adipocytes", "Epithelial_cells", "Hepatocytes", "Keratinocytes", "Myocytes", "Neurons", "Neutrophils".
tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"); use get_tissue_site_detail() to see valid values) or an Ontology ID.
phenotypeId	String. See GTEx portal FAQs for further details.
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Dynamic Association Endpoints: [calculate_expression_quantitative_trait_loci\(\)](#), [calculate_ieqtls\(\)](#), [calculate_splicing_quantitative_trait_loci\(\)](#)

Examples

```
# perform request
calculate_isqtls(
  cellType = "Neutrophils",
  tissueSiteDetailId = "Whole_Blood",
  phenotypeId = "chr1:15947:16607:clu_40980:ENSG00000227232.5",
  variantId = "chr1_1099341_T_C_b38"
)
```

```
calculate_splicing_quantitative_trait_loci
```

Calculate Splicing Quantitative Trait Loci

Description

[GTEx Portal API documentation.](#)

Usage

```
calculate_splicing_quantitative_trait_loci(
  tissueSiteDetailId,
  phenotypeId,
  variantId,
  datasetId = "gtex_v8",
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or an Ontology ID.
phenotypeId	String. See GTEx portal FAQs for further details.
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Dynamic Association Endpoints: [calculate_expression_quantitative_trait_loci\(\)](#), [calculate_ieqtls\(\)](#), [calculate_isqtls\(\)](#)

Examples

```
# perform request - returns a tibble with a single row
calculate_splicing_quantitative_trait_loci(
  tissueSiteDetailId = "Whole_Blood",
  phenotypeId = "chr1:15947:16607:clu_40980:ENSG00000227232.5",
  variantId = "chr1_14677_G_A_b38"
)
```

download

Download

Description

[GTEx Portal API documentation](#)

Usage

```
download(
  materialTypes = NULL,
  tissueSiteDetailIds = NULL,
  pathCategory = NULL,
  tissueSampleIds = NULL,
  sex = NULL,
  sortBy = "sampleId",
  sortDirection = "asc",
  searchTerm = NULL,
  sampleIds = NULL,
  subjectIds = NULL,
  ageBrackets = NULL,
  hardyScales = NULL,
  hasExpressionData = NULL,
  hasGenotype = NULL,
  .return_raw = FALSE
)
```

Arguments

materialTypes String, vector. Options: "Cells:Cell Line Viable", "DNA:DNA Genomic", "DNA:DNA Somatic", "RNA:Total RNA", "Tissue:PAXgene Preserved", "Tissue:PAXgene Preserved Paraffin-embedded", "Tissue:Fresh Frozen Tissue".

tissueSiteDetailIds Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use [get_tissue_site_detail\(\)](#) to see valid values) or Ontology IDs.

pathCategory	Character vector. Options: "adenoma", "amylacea", "atelectasis", "atherosclerosis", "atherosis", "atrophy", "calcification", "cirrhosis", "clean_specimens", "congestion", "corpora_albicantia", "cyst", "desquamation", "diabetic", "dysplasia", "edema", "emphysema", "esophagitis", "fibrosis", "gastritis", "glomerulosclerosis", "goiter", "gynecomastoid", "hashimoto", "heart_failure_cells", "hemorrhage", "hepatitis", "hyalinization", "hypereosinophilia", "hyperplasia", "hypertrophy", "hypoxic", "infarction", "inflammation", "ischemic_changes", "macrophages", "mastopathy", "metaplasia", "monckeberg", "necrosis", "nephritis", "nephrosclerosis", "no_abnormalities", "nodularity", "pancreatitis", "pigment", "pneumonia", "post_menopausal", "prostatitis", "saponification", "scarring", "sclerotic", "solar_elastosis", "spermatogenesis", "steatosis", "sweat_glands", "tma".
tissueSampleIds	Array of strings. A list of Tissue Sample ID(s).
sex	String. Options: "male", "female".
sortBy	String. Options: "sampleId", "ischemicTime", "aliquotId", "tissueSampleId", "hardyScale", "pathologyNotes", "ageBracket", "tissueSiteDetailId", "sex".
sortDirection	String. Options: "asc", "desc". Default = "asc".
searchTerm	String.
sampleIds	Character vector. GTE _x sample ID.
subjectIds	Character vector. GTE _x subject ID.
ageBrackets	The age bracket(s) of the donors of interest. Options: "20-29", "30-39", "40-49", "50-59", "60-69", "70-79".
hardyScales	Character vector. A list of Hardy Scale(s) of interest. Options: "Ventilator case", "Fast death - violent", "Fast death - natural causes", "Intermediate death", "Slow death".
hasExpressionData	Logical.
hasGenotype	Logical.
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Note: running this request with no filters (i.e. `download()`) raises an error.

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Biobank Data Endpoints: [get_sample_biobank_data\(\)](#)

Examples

```
download(
  materialTypes = "RNA:Total RNA",
  tissueSiteDetailIds = "Thyroid",
  pathCategory = "clean_specimens",
  sex = "male",
  ageBrackets = "50-59"
)
```

get_annotation	<i>Get Annotation</i>
----------------	-----------------------

Description

This service returns the list of annotations and allowed values by which a particular dataset can be subsetted. Results may be filtered by dataset.

[GTEx Portal API documentation](#)

Usage

```
get_annotation(
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Note: the output for this function appears to be incomplete currently.

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_annotation()
```

```
get_clustered_median_exon_expression  
    Get Clustered Median Exon Expression
```

Description

Find median transcript expression data along with hierarchical clusters.

- Returns median normalized transcript expression in tissues of all known transcripts of a given gene along with the hierarchical clustering results of tissues and transcripts, based on exon expression, in Newick format.
- The hierarchical clustering is performed by calculating Euclidean distances and using the average linkage method.
- **This endpoint is not paginated.**

By default, this endpoint queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_clustered_median_exon_expression(  
  gencodeIds,  
  datasetId = "gtex_v8",  
  tissueSiteDetailIds = NULL,  
  .return_raw = FALSE  
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: <code>"gtex_v8"</code> , <code>"gtex_snrnaseq_pilot"</code> .

tissueSiteDetailIds
Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use [get_tissue_site_detail\(\)](#) to see valid values) or Ontology IDs.

.return_raw Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_clustered_median_exon_expression(c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))

# clustering data is stored as an attribute "clusters"
result <- get_clustered_median_exon_expression(c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))
attr(result, "clusters")

# process clustering data with the ape package
# install.packages("ape")
# phylo_tree <- ape::read.tree(text = attr(result, "clusters")$tissue)
# plot(phylo_tree)
# print(phylo_tree)
```

get_clustered_median_gene_expression

Get Clustered Median Gene Expression

Description

Find median gene expression data along with hierarchical clusters.

- Returns median gene expression in tissues along with The hierarchical clustering results of tissues and genes, based on gene expression, in Newick format.

- Results may be filtered by dataset, gene or tissue, but at least one gene must be provided
- The hierarchical clustering is performed by calculating Euclidean distances and using the average linkage method.
- **This endpoint is not paginated.**

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_clustered_median_gene_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  .return_raw = FALSE
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_junction_exp](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_clustered_median_gene_expression(gencodeIds = c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))

# clustering data is stored as an attribute "clusters"
```

```
result <- get_clustered_median_gene_expression(c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))
attr(result, "clusters")

# process clustering data with the ape package
# install.packages("ape")
# phylo_tree <- ape::read.tree(text = attr(result, "clusters")$tissue)
# plot(phylo_tree)
# print(phylo_tree)
```

get_clustered_median_junction_expression
Get Clustered Median Junction Expression

Description

Find median junction expression data along with hierarchical clusters.

- Returns median junction read counts in tissues of a given gene from all known transcripts along with the hierarchical clustering results of tissues and genes, based on junction expression, in Newick format.
- Results may be filtered by dataset, gene or tissue, but at least one gene must be provided.
- The hierarchical clustering is performed by calculating Euclidean distances and using the average linkage method.
- **This endpoint is not paginated.**

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_clustered_median_junction_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. c("ENSG00000132693.12", "ENSG00000203782.5").
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".

`tissueSiteDetailIds` Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use `get_tissue_site_detail()` to see valid values) or Ontology IDs.

`.return_raw` Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: `get_clustered_median_exon_expression()`, `get_clustered_median_gene_expression()`, `get_clustered_median_transcript_expression()`, `get_expression_pca()`, `get_gene_expression()`, `get_median_exon_expression()`, `get_median_gene_expression()`, `get_median_junction_expression()`, `get_median_transcript_expression()`, `get_single_nucleus_gex()`, `get_single_nucleus_gex_summary()`, `get_top_expressed_genes()`

Examples

```
get_clustered_median_junction_expression(gencodeIds = c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))

# clustering data is stored as an attribute "clusters"
result <- get_clustered_median_junction_expression(c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))
attr(result, "clusters")

# process clustering data with the ape package
# install.packages("ape")
# phylo_tree <- ape::read.tree(text = attr(result, "clusters")$tissue)
# plot(phylo_tree)
# print(phylo_tree)
```

`get_clustered_median_transcript_expression`
Get Clustered Median Transcript Expression

Description

Find median transcript expression data of all known transcripts of a gene along with hierarchical clusters.

- Returns median normalized expression in tissues of all known transcripts of a given gene along with the hierarchical clustering results of tissues and genes, based on expression, in Newick format.
- Results may be filtered by dataset, gene or tissue, but at least one gene must be provided.
- The hierarchical clustering is performed by calculating Euclidean distances and using the average linkage method.
- **This endpoint is not paginated.**

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_clustered_median_transcript_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. c("ENSG00000132693.12", "ENSG00000203782.5").
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```

get_clustered_median_transcript_expression(gencodeIds = c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))

# clustering data is stored as an attribute "clusters"
result <- get_clustered_median_transcript_expression(c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))
attr(result, "clusters")

# process clustering data with the ape package
# install.packages("ape")
# phylo_tree <- ape::read.tree(text = attr(result, "clusters")$tissue)
# plot(phylo_tree)
# print(phylo_tree)

```

```
get_collapsed_gene_model_exon
```

Get Collapsed Gene Model Exon

Description

This service returns the collapsed exons in the gene model of the given gene. Gene-level and exon-level expression quantification were based on the GENCODE annotation, collapsed to a single transcript model for each gene using an algorithm developed by the GTEx analysis team.

By default, this service queries the models used by the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```

get_collapsed_gene_model_exon(
  gencodeId,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

`gencodeId` String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_](#)
[get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_collapsed_gene_model_exon(gencodeId = "ENSG00000132693.12")
```

get_dataset_info	<i>Get Dataset Info</i>
------------------	-------------------------

Description

[GTEx Portal API documentation](#)

Usage

```
get_dataset_info(  
  datasetId = NULL,  
  organizationName = NULL,  
  .return_raw = FALSE  
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
organizationName	String. Options: "GTEx Consortium" "Kid's First".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

Examples

```
get_dataset_info(datasetId = "gtex_v8", organizationName = "GTEx Consortium")
```

```
get_downloads_page_data
```

Get Downloads Page Data

Description

Retrieves all the files belonging to the given `project_id` for display on the Downloads Page

[GTEx Portal API documentation](#)

Usage

```
get_downloads_page_data(project_id, .return_raw = FALSE)
```

Arguments

`project_id` String. Options: "gtex", "adult-gtex", "egtex".

`.return_raw` Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Note: The GTEx Portal API documentation states "GTEx currently has one project available: gtex". However, `project_id` values "adult-gtex" and "egtex" both return results, whereas "gtex" does not (see examples).

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_](#)
[get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#),
[get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
# "adult-gtex" (default `project_id` value) and "egtex" both return results
get_downloads_page_data("adult-gtex")
egtex <- get_downloads_page_data("egtex")
egtex

# ... "gtex" does not
get_downloads_page_data("gtex")

# get details for whole blood methylation data, including download URL
purrr::pluck(
  egtex$children,
  1,
  "folders",
  "Methylation - EPIC Array",
  "children",
  "folders",
  "mQTLs",
  "children",
  "files",
  "WholeBlood.mQTLs.regular.txt.gz"
)
```

get_eqtl_genes

Get Eqtl Genes

Description

Retrieve eGenes (eQTL Genes).

- This service returns eGenes (eQTL Genes) from the specified dataset.
- eGenes are genes that have at least one significant cis-eQTL acting upon them.
- Results may be filtered by tissue. By default, the service queries the latest GTEx release.

For each eGene, the results include the allelic fold change (`log2AllelicFoldChange`), p-value (`pValue`), p-value threshold (`pValueThreshold`), empirical p-value (`empiricalPValue`), and q-value (`qValue`).

- The `log2AllelicFoldChange` is the allelic fold change (in log2 scale) of the most significant eQTL.
- The `pValue` is the nominal p-value of the most significant eQTL.
- The `pValueThreshold` is the p-value threshold used to determine whether a cis-eQTL for this gene is significant. For more details see <https://gtexportal.org/home/documentationPage#staticTextAnalysisMethods>.
- The `empiricalPValue` is the beta distribution-adjusted empirical p-value from FastQTL.
- The `qValues` were calculated based on the empirical p-values. A false discovery rate (FDR) threshold of ≤ 0.05 was applied to identify genes with a significant eQTL.

[GTEx Portal API documentation.](#)

Usage

```
get_eqtl_genes(
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Static Association Endpoints: [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_by_location\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_sqtl_genes\(\)](#)

Examples

```
get_eqtl_genes(c("Whole_Blood", "Artery_Aorta"))
```

get_exons

*Get Exons***Description**

This service returns exons from all known transcripts of the given gene.

- A versioned GENCODE ID is required to ensure that all exons are from a single gene.
- A dataset ID or both GENCODE version and genome build must be provided.
- Although annotated exons are not dataset dependent, specifying a dataset here is equivalent to specifying the GENCODE version and genome build used by that dataset.

[GTEx Portal API documentation](#)

Usage

```
get_exons(
  gencodeIds,
  gencodeVersion = NULL,
  genomeBuild = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. c("ENSG00000132693.12", "ENSG00000203782.5").
gencodeVersion	String (default = "v26"). GENCODE annotation release. Either "v26" or "v19".
genomeBuild	String. Options: "GRCh38/hg38", "GRCh37/hg19". Default = "GRCh38/hg38".
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_gene_search\(\)](#), [get_genes\(\)](#), [get_genomic_features\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_neighbor_gene\(\)](#), [get_transcripts\(\)](#)

Examples

```
get_exons(gencodeIds = c("ENSG00000132693.12", "ENSG00000203782.5"))
```

get_expression_pca *Get Expression Pca*

Description

Find gene expression PCA data.

- Returns gene expression PCA (principal component analysis) in tissues.
- Results may be filtered by tissue, sample, or dataset.

By default, the service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_expression_pca(
  tissueSiteDetailIds,
  datasetId = "gtex_v8",
  sampleId = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
sampleId	String. <code>^GTEx-[A-Z0-9]{5}-[0-9]{4}-SM-[A-Z0-9]{5}\$</code>
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .

- .verbose Logical. If TRUE (default), print paging information. Set to FALSE globally with `options(list(gtexpr.verbose = FALSE))`.
- .return_raw Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_expression_pca(tissueSiteDetailIds = c(
  "Adipose_Subcutaneous",
  "Whole_Blood"
))

get_expression_pca(
  tissueSiteDetailIds = "Adipose_Subcutaneous",
  sampleId = "GTEx-1117F-0226-SM-5GZZ7"
)
```

get_file_list

Get File List

Description

Get all the files in GTEx dataset for Download page

[GTEx Portal API documentation](#)

Usage

```
get_file_list(.return_raw = FALSE)
```

Arguments

- .return_raw Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

The returned tibble includes a nested list column, "filesets". This details files, sub-categorised by fileset (see examples section).

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
# Column "filesets" is a list column
get_file_list()

# Get "GTEx Analysis V9" file list
gtex_v9_files <- get_file_list() |>
  dplyr::filter(name == "GTEx Analysis V9") |>
  dplyr::pull(filesets)

# "GTEx Analysis V9" filesets
names(gtex_v9_files[[1]])

# "GTEx Analysis V9", "snRNA-Seq Data" fileset files
names(gtex_v9_files[[1]][["snRNA-Seq Data"]]$files)
```

get_fine_mapping

Get Fine Mapping

Description

Retrieve Fine Mapping Data

- Finds and returns Fine Mapping data for the provided list of genes
- By default, this endpoint fetches data from the latest GTEx version

The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#)

Usage

```
get_fine_mapping(
  gencodeIds,
  datasetId = "gtex_v8",
  variantId = NULL,
  tissueSiteDetailIds = NULL,
  page = 0,
```

```

    itemsPerPage = getOption("gtexr.itemsPerPage"),
    .verbose = getOption("gtexr.verbose"),
    .return_raw = FALSE
  )

```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: <code>"gtex_v8"</code> , <code>"gtex_snrnaseq_pilot"</code> .
variantId	String. A gtex variant ID.
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. <code>"Whole_Blood"</code> ; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: `get_eqtl_genes()`, `get_independent_eqtl()`, `get_multi_tissue_eqtls()`, `get_significant_single_tissue_eqtls()`, `get_significant_single_tissue_eqtls_by_location()`, `get_significant_single_tissue_ieqtls()`, `get_significant_single_tissue_isqtls()`, `get_significant_single_sqtl_genes()`

Examples

```

# search by gene
get_fine_mapping(gencodeIds = c(
  "ENSG00000132693.12",
  "ENSG00000203782.5"
))

# optionally filter for a single variant and/or one or more tissues
get_fine_mapping(
  gencodeIds = c(
    "ENSG00000132693.12",
    "ENSG00000203782.5"
  ),

```

```

variantId = "chr1_153228363_A_G_b38",
tissueSiteDetailIds = c(
  "Whole_Blood",
  "Thyroid"
)
)

```

```

get_full_get_collapsed_gene_model_exon
  Get Full Get Collapsed Gene Model Exon

```

Description

This service allows the user to query the full Collapsed Gene Model Exon of a specific gene by gencode ID

[GTEx Portal API documentation](#)

Usage

```

get_full_get_collapsed_gene_model_exon(
  gencodeId,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_full_get_collapsed_gene_model_exon(gencodeId = "ENSG00000203782.5")
```

```
get_functional_annotation
```

Get Functional Annotation

Description

This endpoint retrieves the functional annotation of a certain chromosome location. Default to most recent dataset release.

[GTEx Portal API documentation](#)

Usage

```
get_functional_annotation(
  datasetId = "gtex_v8",
  chromosome,
  start,
  end,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
start	Integer.
end	Integer.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).

- `.verbose` Logical. If TRUE (default), print paging information. Set to FALSE globally with `options(list(gtexpr.verbose = FALSE))`.
- `.return_raw` Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Datasets Endpoints: `get_annotation()`, `get_collapsed_gene_model_exon()`, `get_downloads_page_data()`, `get_file_list()`, `get_full_get_collapsed_gene_model_exon()`, `get_linkage_disequilibrium_by_variant_data()`, `get_linkage_disequilibrium_data()`, `get_sample_datasets()`, `get_subject()`, `get_tissue_site_detail()`, `get_variant()`, `get_variant_by_location()`

Examples

```
get_functional_annotation(chromosome = "chr1", start = 192168000, end = 192169000)
```

get_genes

Get Genes

Description

This service returns information about reference genes. A genome build and GENCODE version must be provided.

- Genes are searchable by gene symbol, GENCODE ID and versioned GENCODE ID.
- Versioned GENCODE ID is recommended to ensure unique ID matching.
- By default, this service queries the genome build and GENCODE version used by the latest GTEx release.

[GTEx API Portal documentation](#)

Usage

```
get_genes(
  geneIds,
  gencodeVersion = "v26",
  genomeBuild = "GRCh38/hg38",
  page = 0,
  itemsPerPage = getOption("gtexpr.itemsPerPage"),
  .verbose = getOption("gtexpr.verbose"),
  .return_raw = FALSE
)
```

Arguments

geneIds	A character vector of gene symbols, versioned gencodeIds, or unversioned gencodeIds.
gencodeVersion	String (default = "v26"). GENCODE annotation release. Either "v26" or "v19".
genomeBuild	String. Options: "GRCh38/hg38", "GRCh37/hg19". Default = "GRCh38/hg38".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_gene_search\(\)](#), [get_genomic_features\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_neighbor_gene\(\)](#), [get_transcripts\(\)](#)

Examples

```
get_genes(c("CRP", "IL6R"))
```

get_gene_expression *Get Gene Expression*

Description

Find normalized gene expression data.

- Returns normalized gene expression in tissues at the sample level.
- Results may be filtered by dataset, gene or tissue, but at least one gene must be provided.

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```

get_gene_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  attributeSubset = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
attributeSubset	String. Examples include but are not limited to "sex", "ageBracket"
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: `get_clustered_median_exon_expression()`, `get_clustered_median_gene_expression()`, `get_clustered_median_junction_expression()`, `get_clustered_median_transcript_expression()`, `get_expression_pca()`, `get_median_exon_expression()`, `get_median_gene_expression()`, `get_median_junction_expression()`, `get_median_transcript_expression()`, `get_single_nucleus_gex()`, `get_single_nucleus_gex_summary()`, `get_top_expressed_genes()`

Examples

```

# multiple genes, selected tissues
get_gene_expression(

```

```

    gencodeIds = c(
      "ENSG00000132693.12",
      "ENSG00000203782.5"
    ),
    tissueSiteDetailIds = c("Thyroid", "Whole_Blood")
  )

# single gene, selected (single) tissue
get_gene_expression(
  gencodeIds = "ENSG00000132693.12",
  tissueSiteDetailIds = "Whole_Blood"
)

# subset by sex
get_gene_expression(
  gencodeIds = "ENSG00000132693.12",
  tissueSiteDetailIds = "Whole_Blood",
  attributeSubset = "sex"
)

# subset by age bracket
get_gene_expression(
  gencodeIds = "ENSG00000132693.12",
  tissueSiteDetailIds = "Whole_Blood",
  attributeSubset = "ageBracket"
)

```

get_gene_search

Get Gene Search

Description

Find genes that are partial or complete match of a gene_id

- gene_id could be a gene symbol, a gencode ID, or an Ensemble ID
- Gencode Version and Genome Build must be specified

[GTEx Portal API documentation](#)

Usage

```

get_gene_search(
  geneId,
  gencodeVersion = "v26",
  genomeBuild = "GRCh38/hg38",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

geneId	String. A gene symbol, a gencode ID, or an Ensemble ID.
gencodeVersion	String (default = "v26"). GENCODE annotation release. Either "v26" or "v19".
genomeBuild	String. Options: "GRCh38/hg38", "GRCh37/hg19". Default = "GRCh38/hg38".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_genes\(\)](#), [get_genomic_features\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_neighbor_gene\(\)](#), [get_transcripts\(\)](#)

Examples

```
get_gene_search("CRP")
```

get_genomic_features *Get Genomic Features*

Description

[GTEx API Portal documentation](#)

Usage

```
get_genomic_features(.featureId, datasetId = "gtex_v8", .return_raw = FALSE)
```

Arguments

.featureId	String. A genomic feature e.g. GENCODE ID, RSID or GTEx Variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

This endpoint takes a path parameter "featureId".

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_gene_search\(\)](#), [get_genes\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_neighbor_gene\(\)](#), [get_transcripts\(\)](#)

Examples

```
# gene symbol
get_genomic_features("brca1")

# GENCODE ID
get_genomic_features("ENSG00000132693.12")

# RSID
get_genomic_features("rs1815739")

# GTEx variant ID
get_genomic_features("chr11_66561023_G_GTTA_b38")
```

```
get_gwas_catalog_by_location
  Get Gwas Catalog By Location
```

Description

Find the GWAS Catalog on a certain chromosome between start and end locations.

[GTEx API Portal documentation](#)

Usage

```
get_gwas_catalog_by_location(
  start,
  end,
  chromosome,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

start	Integer.
end	Integer.
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_gene_search\(\)](#), [get_genes\(\)](#), [get_genomic_features\(\)](#), [get_neighbor_gene\(\)](#), [get_transcripts\(\)](#)

Examples

```
get_gwas_catalog_by_location(start = 1, end = 10000000, chromosome = "chr1")
```

get_image

Get Image

Description

[GTEx Portal API documentation](#)

Usage

```
get_image(
  tissueSampleIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

tissueSampleIds	Array of strings. A list of Tissue Sample ID(s).
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

Examples

```
get_image()

# filter by `tissueSampleId`
result <- get_image(tissueSampleIds = "GTEx-1117F-0526")
print(result)

# note that `pathologyNotesCategories` (if present) is a list column
print(result$pathologyNotesCategories)
```

get_independent_eqtl *Get Independent Eqtl*

Description

Retrieve Independent eQTL Data

- Finds and returns Independent eQTL Data data for the provided list of genes
- By default, this endpoint fetches data from the latest GTEx version

The retrieved data is split into pages with items_per_page entries per page

[GTEx portal API documentation](#)

Usage

```
get_independent_eqtl(
  gencodeIds,
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
```

```

  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: `get_eqtl_genes()`, `get_fine_mapping()`, `get_multi_tissue_eqtls()`, `get_significant_single_tissue_eqtls()`, `get_significant_single_tissue_eqtls_by_location()`, `get_significant_single_tissue_ieqtls()`, `get_significant_single_tissue_isqtls()`, `get_significant_single_eqtl_genes()`

Examples

```

# search by gene
get_independent_eqtl(gencodeIds = c(
  "ENSG00000132693.12",
  "ENSG00000203782.5"
))

# optionally filter for a single variant and/or one or more tissues
get_independent_eqtl(
  gencodeIds = c(
    "ENSG00000132693.12",
    "ENSG00000203782.5"
  ),
  tissueSiteDetailIds = c(
    "Whole_Blood",

```

```

    "Thyroid"
  )
)

```

```
get_linkage_disequilibrium_by_variant_data
```

Get Linkage Disequilibrium By Variant Data

Description

Find linkage disequilibrium (LD) data for a given variant

[GTEx Portal API documentation](#)

Usage

```

get_linkage_disequilibrium_by_variant_data(
  variantId,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

variantId	String. A gtex variant ID.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_linkage_disequilibrium_by_variant_data("chr1_159245536_C_T_b38")
```

```
get_linkage_disequilibrium_data
  Get Linkage Disequilibrium Data
```

Description

Find linkage disequilibrium (LD) data for a given gene.

This endpoint returns linkage disequilibrium data for the cis-eQTLs found associated with the provided gene in a specified dataset. Results are queried by gencode ID. By default, the service queries the latest GTEx release. Specify a dataset ID to fetch results from a different dataset.

[GTEx Portal API documentation](#)

Usage

```
get_linkage_disequilibrium_data(
  gencodeId,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_linkage_disequilibrium_data(gencodeId = "ENSG00000132693.12")
```

```
get_maintenance_message
```

Get Maintenance Message

Description

Getting all the maintenance messages from the database that are enabled.

[GTEx Portal API documentation.](#)

Usage

```
get_maintenance_message(
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Note this typically returns an empty tibble.

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Admin Endpoints: [get_news_item\(\)](#)

Examples

```
get_maintenance_message()
```

```
get_median_exon_expression  
Get Median Exon Expression
```

Description

Find median exon expression data.

- Returns median exon read counts, in tissues, of a collapsed gene model.
- Results may be filtered by dataset, gene or tissue, but at least one gene must be provided

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_median_exon_expression(  
  gencodeIds,  
  datasetId = "gtex_v8",  
  tissueSiteDetailIds = NULL,  
  page = 0,  
  itemsPerPage = getOption("gtexr.itemsPerPage"),  
  .verbose = getOption("gtexr.verbose"),  
  .return_raw = FALSE  
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: <code>"gtex_v8"</code> , <code>"gtex_snrnaseq_pilot"</code> .
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. <code>"Whole_Blood"</code> ; use get_tissue_site_detail() to see valid values) or Ontology IDs.
<code>page</code>	Integer (default = 0).

itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
# median exon expression values for CRP, filtered for whole blood
get_median_exon_expression(
  gencodeIds = "ENSG00000132693.12",
  tissueSiteDetailIds = "Whole_Blood"
)
```

get_median_gene_expression

Get Median Gene Expression

Description

Find median gene expression data along with hierarchical clusters.

- Returns median gene expression in tissues.
- By default, this endpoint queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_median_gene_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexpr.itemsPerPage"),
```

```

    .verbose = getOption("gtexr.verbose"),
    .return_raw = FALSE
  )

```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_median_gene_expression(gencodeIds = "ENSG00000132693.12")
```

```
get_median_junction_expression
```

Get Median Junction Expression

Description

Find junction gene expression data.

- Returns median junction read counts in tissues of a given gene from all known transcripts.
- Results may be filtered by dataset or tissue.

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_median_junction_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. c("ENSG00000132693.12", "ENSG00000203782.5").
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_median_junction_expression(gencodeIds = "ENSG00000132693.12")
```

```
get_median_transcript_expression
      Get Median Transcript Expression
```

Description

Find median transcript expression data of all known transcripts of a gene.

- Returns median normalized expression in tissues of all known transcripts of a given gene.
- Results may be filtered by dataset or tissue.

By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_median_transcript_expression(
  gencodeIds,
  datasetId = "gtex_v8",
  tissueSiteDetailIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. c("ENSG00000132693.12", "ENSG00000203782.5").
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
get_median_transcript_expression(gencodeIds = "ENSG00000132693.12")
```

`get_multi_tissue_eqtls`*Get Multi Tissue Eqtls*

Description

Find multi-tissue eQTL Metasoft results.

- This service returns multi-tissue eQTL Metasoft results for a given gene and variant in a specified dataset.
- A Versioned GENCODE ID must be provided.
- For each tissue, the results include: m-value (`mValue`), normalized effect size (`nes`), p-value (`pValue`), and standard error (`se`).
- The m-value is the posterior probability that an eQTL effect exists in each tissue tested in the cross-tissue meta-analysis (Han and Eskin, PLoS Genetics 8(3): e1002555, 2012).
- The normalized effect size is the slope of the linear regression of normalized expression data versus the three genotype categories using single-tissue eQTL analysis, representing eQTL effect size.
- The p-value is from a t-test that compares observed NES from single-tissue eQTL analysis to a null NES of 0.

By default, the service queries the latest GTEx release. The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#)

Usage

```
get_multi_tissue_eqtls(
  gencodeId,
  variantId = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_by_location\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_sqtl_genes\(\)](#)

Examples

```
# search by gene
get_multi_tissue_eqtls(gencodeId = "ENSG00000132693.12")

# note that 'tissues' is a list-column
x <- get_multi_tissue_eqtls(gencodeId = "ENSG00000132693.12",
  variantId = "chr1_159476920_T_C_b38")

x$tissues[[1]]
```

get_neighbor_gene *Get Neighbor Gene*

Description

Find all neighboring genes on a certain chromosome around a position with a certain window size.

[GTEx API Portal documentation](#)

Usage

```
get_neighbor_gene(
  pos,
  chromosome,
  bp_window,
  page = 0,
  gencodeVersion = "v26",
  genomeBuild = "GRCh38/hg38",
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

pos	Integer.
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
bp_window	Integer.
page	Integer (default = 0).
gencodeVersion	String (default = "v26"). GENCODE annotation release. Either "v26" or "v19".
genomeBuild	String. Options: "GRCh38/hg38", "GRCh37/hg19". Default = "GRCh38/hg38".
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_gene_search\(\)](#), [get_genes\(\)](#), [get_genomic_features\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_transcripts\(\)](#)

Examples

```
get_neighbor_gene(pos = 1000000, chromosome = "chr1", bp_window = 10000)
```

get_news_item

Get News Item

Description

Getting all the news items from the database that are current.

[GTEx Portal API documentation.](#)

Usage

```
get_news_item(
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Admin Endpoints: [get_maintenance_message\(\)](#)

Examples

```
get_news_item()
```

 get_sample_biobank_data

Get Sample (Biobank Data)

Description

[GTEx Portal API documentation](#)

Usage

```

get_sample_biobank_data(
  draw = NULL,
  materialTypes = NULL,
  tissueSiteDetailIds = NULL,
  pathCategory = NULL,
  tissueSampleIds = NULL,
  sex = NULL,
  sortBy = "sampleId",
  sortDirection = "asc",
  searchTerm = NULL,
  sampleIds = NULL,
  subjectIds = NULL,
  ageBrackets = NULL,
  hardyScales = NULL,
  hasExpressionData = NULL,
  hasGenotype = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

draw	Integer.
materialTypes	String, vector. Options: "Cells:Cell Line Viable", "DNA:DNA Genomic", "DNA:DNA Somatic", "RNA:Total RNA", "Tissue:PAXgene Preserved", "Tissue:PAXgene Preserved Paraffin-embedded", "Tissue:Fresh Frozen Tissue".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"); use get_tissue_site_detail() to see valid values) or Ontology IDs.
pathCategory	Character vector. Options: "adenoma", "amylacea", "atelectasis", "atherosclerosis", "atherosis", "atrophy", "calcification", "cirrhosis", "clean_specimens",

"congestion", "corpora_albicantia", "cyst", "desquamation", "diabetic", "dysplasia", "edema", "emphysema", "esophagitis", "fibrosis", "gastritis", "glomerulosclerosis", "goiter", "gynecomastoid", "hashimoto", "heart_failure_cells", "hemorrhage", "hepatitis", "hyalinization", "hypereosinophilia", "hyperplasia", "hypertrophy", "hypoxic", "infarction", "inflammation", "ischemic_changes", "macrophages", "mastopathy", "metaplasia", "monckeberg", "necrosis", "nephritis", "nephrosclerosis", "no_abnormalities", "nodularity", "pancreatitis", "pigment", "pneumonia", "post menopausal", "prostatitis", "saponification", "scarring", "sclerotic", "solar_elastosis", "spermatogenesis", "steatosis", "sweat_glands", "tma".

tissueSampleIds	Array of strings. A list of Tissue Sample ID(s).
sex	String. Options: "male", "female".
sortBy	String. Options: "sampleId", "ischemicTime", "aliquotId", "tissueSampleId", "hardyScale", "pathologyNotes", "ageBracket", "tissueSiteDetailId", "sex".
sortDirection	String. Options: "asc", "desc". Default = "asc".
searchTerm	String.
sampleIds	Character vector. GTEx sample ID.
subjectIds	Character vector. GTEx subject ID.
ageBrackets	The age bracket(s) of the donors of interest. Options: "20-29", "30-39", "40-49", "50-59", "60-69", "70-79".
hardyScales	Character vector. A list of Hardy Scale(s) of interest. Options: "Ventilator case", "Fast death - violent", "Fast death - natural causes", "Intermediate death", "Slow death".
hasExpressionData	Logical.
hasGenotype	Logical.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Biobank Data Endpoints: [download\(\)](#)

Examples

```
get_sample_biobank_data(tissueSiteDetailIds = "Whole_Blood")
```

get_sample_datasets *Get Sample (Datasets)*

Description

This service returns information of samples used in analyses from all datasets. Results may be filtered by dataset ID, sample ID, subject ID, sample metadata, or other provided parameters. By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_sample_datasets(
  datasetId = "gtex_v8",
  sampleIds = NULL,
  tissueSampleIds = NULL,
  subjectIds = NULL,
  ageBrackets = NULL,
  sex = NULL,
  pathCategory = NULL,
  tissueSiteDetailIds = NULL,
  aliquotIds = NULL,
  autolysisScores = NULL,
  hardyScales = NULL,
  ischemicTimes = NULL,
  ischemicTimeGroups = NULL,
  rins = NULL,
  uberonIds = NULL,
  dataTypes = NULL,
  sortBy = "sampleId",
  sortDirection = "asc",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
sampleIds	Character vector. GTEx sample ID.
tissueSampleIds	Array of strings. A list of Tissue Sample ID(s).
subjectIds	Character vector. GTEx subject ID.

ageBrackets	The age bracket(s) of the donors of interest. Options: "20-29", "30-39", "40-49", "50-59", "60-69", "70-79".
sex	String. Options: "male", "female".
pathCategory	Character vector. Options: "adenoma", "amylacea", "atelectasis", "atherosclerosis", "atherosis", "atrophy", "calcification", "cirrhosis", "clean_specimens", "congestion", "corpora_albicantia", "cyst", "desquamation", "diabetic", "dysplasia", "edema", "emphysema", "esophagitis", "fibrosis", "gastritis", "glomerulosclerosis", "goiter", "gynecomastoid", "hashimoto", "heart_failure_cells", "hemorrhage", "hepatitis", "hyalinization", "hypereosinophilia", "hyperplasia", "hypertrophy", "hypoxic", "infarction", "inflammation", "ischemic_changes", "macrophages", "mastopathy", "metaplasia", "monckeberg", "necrosis", "nephritis", "nephrosclerosis", "no_abnormalities", "nodularity", "pancreatitis", "pigment", "pneumonia", "post menopausal", "prostatitis", "saponification", "scarring", "sclerotic", "solar_elastosis", "spermatogenesis", "steatosis", "sweat_glands", "tma".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
aliquotIds	Character vector.
autolysisScores	Character vector. Options: "None", "Mild", "Moderate", "Severe".
hardyScales	Character vector. A list of Hardy Scale(s) of interest. Options: "Ventilator case", "Fast death - violent", "Fast death - natural causes", "Intermediate death", "Slow death".
ischemicTimes	Integer.
ischemicTimeGroups	Character vector. Options: "<= 0", "1 - 300", "301 - 600", "601 - 900", "901 - 1200", "1201 - 1500", "> 1500".
rins	Integer, vector.
uberonIds	Character vector of Uberon IDs (e.g. "UBERON:EFO_0000572"; use <code>get_tissue_site_detail()</code> to see valid values).
dataTypes	Character vector. Options: "RNASEQ", "WGS", "WES", "OMNI", "EXCLUDE".
sortBy	String. Options: "sampleId", "ischemicTime", "aliquotId", "tissueSampleId", "hardyScale", "pathologyNotes", "ageBracket", "tissueSiteDetailId", "sex".
sortDirection	String. Options: "asc", "desc". Default = "asc".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexpr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexpr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_sample_datasets()
```

get_service_info	<i>Get Service Info</i>
------------------	-------------------------

Description

General information about the GTEx service.

[GTEx Portal API documentation.](#)

Usage

```
get_service_info(.return_raw = FALSE)
```

Arguments

`.return_raw` Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

Examples

```
get_service_info()
```

```
get_significant_single_tissue_eqtls
```

Get Significant Single Tissue Eqtls

Description

Find significant single tissue eQTLs.

- This service returns precomputed significant single tissue eQTLs.
- Results may be filtered by tissue, gene, variant or dataset.
- To search by gene, use the versioned GENCODE ID.
- To search by variant, use the dbSNP rs ID (snpId).

By default, the service queries the latest GTEx release and the retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#).

Usage

```
get_significant_single_tissue_eqtls(
  gencodeIds = NULL,
  variantIds = NULL,
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>variantIds</code>	Character vector. Gtex variant IDs.
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Details

Note: although the GTEx Portal API documentation says to use the dbSNP rsID when searching by variant, this returns no results. Instead use gtex variant IDs e.g. use "chr1_153209640_C_A_b38" instead of "rs1410858".

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_by_location\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_sqtls\(\)](#), [get_sqtln_genes\(\)](#)

Examples

```
# search by gene
get_significant_single_tissue_eqtls(gencodeIds = c(
  "ENSG00000132693.12",
  "ENSG00000203782.5"
))

# search by variant - must be variantId (not rsid)
get_significant_single_tissue_eqtls(variantIds = "chr1_153209640_C_A_b38")

# filter by gene/variant and tissue site - either `gencodeIds` or `variantIds`
# should be supplied as a minimum
get_significant_single_tissue_eqtls(
  gencodeIds = c(
    "ENSG00000132693.12",
    "ENSG00000203782.5"
  ),
  variantIds = "chr1_153209640_C_A_b38",
  tissueSiteDetailIds = "Whole_Blood"
)
```

get_significant_single_tissue_eqtls_by_location

Get Significant Single Tissue eQTLs By Location

Description

Find significant single tissue eQTLs using Chromosomal Locations.

- This service returns precomputed significant single tissue eQTLs.
- Results may be filtered by tissue, and/or dataset.

By default, the service queries the latest GTEx release. Since this endpoint is used to support a third party program on the portal, the return structure is different from other endpoints and is not paginated.

[GTEx Portal API documentation](#)

Usage

```
get_significant_single_tissue_eqtls_by_location(
  tissueSiteDetailId,
  start,
  end,
  chromosome,
  datasetId = "gtex_v8",
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or an Ontology ID.
start	Integer.
end	Integer.
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_sqtls\(\)](#), [get_sqtl_genes\(\)](#)

Examples

```
get_significant_single_tissue_eqtls_by_location(
  tissueSiteDetailId = "Artery_Aorta",
  start = 10000,
  end = 250000,
  chromosome = "chr11"
```

```
)
```

```
get_significant_single_tissue_ieqtls
```

```
Get Significant Single Tissue Ieqtls
```

Description

Retrieve Interaction eQTL Data.

- This service returns cell type interaction eQTLs (ieQTLs), from a specified dataset.
- Results may be filtered by tissue
- By default, the service queries the latest GTEx release.

The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#)

Usage

```
get_significant_single_tissue_ieqtls(  
  gencodeIds,  
  variantIds = NULL,  
  tissueSiteDetailIds = NULL,  
  datasetId = "gtex_v8",  
  page = 0,  
  itemsPerPage = getOption("gtexr.itemsPerPage"),  
  .verbose = getOption("gtexr.verbose"),  
  .return_raw = FALSE  
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>variantIds</code>	Character vector. Gtex variant IDs.
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_sqtls\(\)](#), [get_sqtls_genes\(\)](#)

Examples

```
get_significant_single_tissue_ieqtls(c(
  "ENSG00000132693.12",
  "ENSG00000203782.5"
))
```

```
get_significant_single_tissue_isqtls
  Get Significant Single Tissue Isqtls
```

Description

Retrieve Interaction sQTL Data.

- This service retrieves cell type interaction sQTLs (isQTLs), from a specified dataset.
- Results may be filtered by tissue
- By default, the service queries the latest GTEx release.

The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#).

Usage

```
get_significant_single_tissue_isqtls(
  gencodeIds,
  variantIds = NULL,
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeIds	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
variantIds	Character vector. Gtex variant IDs.
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEX specific IDs (e.g. "Whole_Blood"; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: `get_eqtl_genes()`, `get_fine_mapping()`, `get_independent_eqtl()`, `get_multi_tissue_eqtls()`, `get_significant_single_tissue_eqtls()`, `get_significant_single_tissue_eqtls_l`, `get_significant_single_tissue_ieqtls()`, `get_significant_single_tissue_sqtls()`, `get_sqtln_genes()`

Examples

```
get_significant_single_tissue_isqtls(gencodeIds = c(
  "ENSG00000065613.9",
  "ENSG00000203782.5"
))
```

```
get_significant_single_tissue_sqtls
```

Get Significant Single Tissue Sqtls

Description

Retrieve Single Tissue sQTL Data.

- This service returns single tissue sQTL data for the given genes, from a specified dataset.
- Results may be filtered by tissue

- By default, the service queries the latest GTEx release.

The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#).

Usage

```
get_significant_single_tissue_sqtls(
  gencodeIds = NULL,
  variantIds = NULL,
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>variantIds</code>	Character vector. Gtex variant IDs.
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_l\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_sqtl_genes\(\)](#)

Examples

```
# search by gene
get_significant_single_tissue_sqtlts(gencodeIds = c(
  "ENSG00000065613.9",
  "ENSG00000203782.5"
))
```

```
get_single_nucleus_gex
```

Get Single Nucleus Gex

Description

Retrieve Single Nucleus Gene Expression Data for a given Gene.

[GTEx Portal API documentation](#)

Usage

```
get_single_nucleus_gex(
  gencodeIds,
  datasetId = "gtex_snrnaseq_pilot",
  tissueSiteDetailIds = NULL,
  excludeDataArray = TRUE,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

<code>gencodeIds</code>	A character vector of Versioned GENCODE IDs, e.g. <code>c("ENSG00000132693.12", "ENSG00000203782.5")</code> .
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: <code>"gtex_v8"</code> , <code>"gtex_snrnaseq_pilot"</code> .
<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. <code>"Whole_Blood"</code> ; use <code>get_tissue_site_detail()</code> to see valid values) or Ontology IDs.
<code>excludeDataArray</code>	String. Options are <code>TRUE</code> or <code>FALSE</code>
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .

`.verbose` Logical. If TRUE (default), print paging information. Set to FALSE globally with `options(list(gtexpr.verbose = FALSE))`.

`.return_raw` Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex_summary\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
# Search for one or more genes - returns a tibble with one row per tissue.
# Column "cellTypes" now contains a tibble of expression summary data, with
# one row for each cell type
get_single_nucleus_gex(gencodeIds = c(
  "ENSG00000203782.5",
  "ENSG00000132693.12"
))

# `excludeDataArray = FALSE` - expression values are stored under "celltypes"
# in an additional column called "data"
response <- get_single_nucleus_gex(
  gencodeIds = "ENSG00000132693.12",
  excludeDataArray = FALSE,
  itemsPerPage = 2
)

response

# "cellTypes" contains a tibble of data with one row for each
# cell type e.g. for Breast_Mammary_Tissue
response$cellTypes[[2]]

# when `excludeDataArray = FALSE`, expression values are stored in "data"
# e.g. for Breast_Mammary_Tissue, Epithelial cell (luminal):
response$cellTypes[[2]]$data[[1]]
```

`get_single_nucleus_gex_summary`

Get Single Nucleus Gex Summary

Description

Retrieve Summarized Single Nucleus Gene Expression Data.

[GTEx Portal API documentation](#)

Usage

```
get_single_nucleus_gex_summary(
  datasetId = "gtex_snrnaseq_pilot",
  tissueSiteDetailIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
tissueSiteDetailIds	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_top_expressed_genes\(\)](#)

Examples

```
# all tissues
get_single_nucleus_gex_summary()

# filter for specific tissue
```

```

get_single_nucleus_gex_summary(tissueSiteDetailIds = c(
  "Breast_Mammary_Tissue",
  "Skin_Sun_Exposed_Lower_leg"
))

```

get_sqtl_genes *Get Sqtl Genes*

Description

Retrieve sGenes (sQTL Genes).

- This service returns sGenes (sQTL Genes) from the specified dataset.
- Results may be filtered by tissue.
- By default, the service queries the latest GTEx release.

The retrieved data is split into pages with `items_per_page` entries per page

[GTEx Portal API documentation](#).

Usage

```

get_sqtl_genes(
  tissueSiteDetailIds = NULL,
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

<code>tissueSiteDetailIds</code>	Character vector of IDs for tissues of interest. Can be GTEx specific IDs (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or Ontology IDs.
<code>datasetId</code>	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with <code>options(list(gtexr.itemsPerPage = 100000))</code> .
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with <code>options(list(gtexr.verbose = FALSE))</code> .
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Static Association Endpoints: [get_eqtl_genes\(\)](#), [get_fine_mapping\(\)](#), [get_independent_eqtl\(\)](#), [get_multi_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls\(\)](#), [get_significant_single_tissue_eqtls_l\(\)](#), [get_significant_single_tissue_ieqtls\(\)](#), [get_significant_single_tissue_isqtls\(\)](#), [get_significant_single_tissue_isqtls_l\(\)](#)

Examples

```
get_sqtl_genes("Whole_Blood")
```

get_subject

Get Subject

Description

This service returns information of subjects used in analyses from all datasets. Results may be filtered by dataset ID, subject ID, sex, age bracket or Hardy Scale. By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_subject(
  datasetId = "gtex_v8",
  sex = NULL,
  ageBrackets = NULL,
  hardyScale = NULL,
  subjectIds = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
sex	String. Options: "male", "female".
ageBrackets	The age bracket(s) of the donors of interest. Options: "20-29", "30-39", "40-49", "50-59", "60-69", "70-79".

<code>hardyScale</code>	String A Hardy Scale of interest. Options: "Ventilator case", "Fast death - violent", "Fast death - natural causes", "Intermediate death", "Slow death".
<code>subjectIds</code>	Character vector. GTEx subject ID.
<code>page</code>	Integer (default = 0).
<code>itemsPerPage</code>	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
<code>.verbose</code>	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
<code>.return_raw</code>	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if `.return_raw = TRUE`.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
get_subject()
```

```
get_tissue_site_detail
```

Get Tissue Site Detail

Description

Retrieve all tissue site detail information in the database

[GTEx Portal API documentation](#)

Usage

```
get_tissue_site_detail(
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexpr.itemsPerPage"),
  .verbose = getOption("gtexpr.verbose"),
  .return_raw = FALSE
)
```

Arguments

datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_variant\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
# returns a tibble with one row per tissue
get_tissue_site_detail()

# `eqtlSampleSummary` and `rnaSeqSampleSummary` are list columns
bladder_site_details <- get_tissue_site_detail() |>
  dplyr::filter(tissueSiteDetailId == "Bladder")

purrr::pluck(bladder_site_details, "eqtlSampleSummary", 1)

purrr::pluck(bladder_site_details, "rnaSeqSampleSummary", 1)
```

```
get_top_expressed_genes
```

Get Top Expressed Genes

Description

Find top expressed genes for a specified tissue.

- Returns top expressed genes for a specified tissue in a dataset, sorted by median expression.
- When the optional parameter filterMtGene is set to true, mitochondrial genes will be excluded from the results. By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_top_expressed_genes(
  tissueSiteDetailId,
  datasetId = "gtex_v8",
  filterMtGene = TRUE,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

tissueSiteDetailId	String. The ID of the tissue of interest. Can be a GTEx specific ID (e.g. "Whole_Blood"; use get_tissue_site_detail() to see valid values) or an Ontology ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
filterMtGene	Logical. Exclude mitochondrial genes.
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Expression Data Endpoints: [get_clustered_median_exon_expression\(\)](#), [get_clustered_median_gene_expression\(\)](#), [get_clustered_median_junction_expression\(\)](#), [get_clustered_median_transcript_expression\(\)](#), [get_expression_pca\(\)](#), [get_gene_expression\(\)](#), [get_median_exon_expression\(\)](#), [get_median_gene_expression\(\)](#), [get_median_junction_expression\(\)](#), [get_median_transcript_expression\(\)](#), [get_single_nucleus_gex\(\)](#), [get_single_nucleus_gex_summary\(\)](#)

Examples

```
get_top_expressed_genes(tissueSiteDetailId = "Artery_Aorta")
```

get_transcripts	<i>Get Transcripts</i>
-----------------	------------------------

Description

Find all transcripts of a reference gene.

- This service returns information about transcripts of the given versioned GENCODE ID.
- A genome build and GENCODE version must be provided.
- By default, this service queries the genome build and GENCODE version used by the latest GTEx release.

[GTEx API Portal documentation](#)

Usage

```
get_transcripts(
  gencodeId,
  gencodeVersion = "v26",
  genomeBuild = "GRCh38/hg38",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

gencodeId	String. A Versioned GENCODE ID of a gene, e.g. "ENSG00000065613.9".
gencodeVersion	String (default = "v26"). GENCODE annotation release. Either "v26" or "v19".
genomeBuild	String. Options: "GRCh38/hg38", "GRCh37/hg19". Default = "GRCh38/hg38".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Reference Genome Endpoints: [get_exons\(\)](#), [get_gene_search\(\)](#), [get_genes\(\)](#), [get_genomic_features\(\)](#), [get_gwas_catalog_by_location\(\)](#), [get_neighbor_gene\(\)](#)

Examples

```
get_transcripts(gencodeId = "ENSG00000203782.5")
```

```
get_variant
```

```
Get Variant
```

Description

This service returns information about a variant, including position, dbSNP RS ID, the reference allele, the alternative allele, and whether the minor allele frequency is $\geq 1\%$. For GTEx v6p, there is also information about whether the whole exome sequence and chip sequencing data are available. Results may be queried by GTEx variant ID (variantId), dbSNP RS ID (snpId) or genomic location (chromosome and pos). Variants are identified based on the genotype data of each dataset cohort, namely, are dataset-dependent. Each variant is assigned a unique GTEx variant ID (i.e. the primary key). Not all variants have a mappable dbSNP RS ID. By default, this service queries the latest GTEx release.

[GTEx Portal API documentation](#)

Usage

```
get_variant(
  snpId = NULL,
  variantId = NULL,
  datasetId = "gtex_v8",
  chromosome = NULL,
  poss = NULL,
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)
```

Arguments

snpId	String
variantId	String. A gtex variant ID.
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
poss	Integer, vector.
page	Integer (default = 0).

itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexpr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexpr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant_by_location\(\)](#)

Examples

```
# search by rsid
get_variant(snpId = "rs1410858")

# search by variantId
get_variant(variantId = "chr1_153209640_C_A_b38")

# search by chromosome and position
get_variant(
  chromosome = "chr1",
  pos = 153209600:153209700
)
```

get_variant_by_location

Get Variant By Location

Description

This service allows the user to query information about variants on a certain chromosome at a certain location.

[GTEx Portal API documentation](#)

Usage

```

get_variant_by_location(
  start,
  end,
  chromosome,
  sortBy = "pos",
  sortDirection = "asc",
  datasetId = "gtex_v8",
  page = 0,
  itemsPerPage = getOption("gtexr.itemsPerPage"),
  .verbose = getOption("gtexr.verbose"),
  .return_raw = FALSE
)

```

Arguments

start	Integer.
end	Integer.
chromosome	String. One of "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY".
sortBy	String. Options: "sampleId", "ischemicTime", "aliquotId", "tissueSampleId", "hardyScale", "pathologyNotes", "ageBracket", "tissueSiteDetailId", "sex".
sortDirection	String. Options: "asc", "desc". Default = "asc".
datasetId	String. Unique identifier of a dataset. Usually includes a data source and data release. Options: "gtex_v8", "gtex_snrnaseq_pilot".
page	Integer (default = 0).
itemsPerPage	Integer (default = 250). Set globally to maximum value 100000 with options(list(gtexr.itemsPerPage = 100000)).
.verbose	Logical. If TRUE (default), print paging information. Set to FALSE globally with options(list(gtexr.verbose = FALSE)).
.return_raw	Logical. If TRUE, return the raw API JSON response. Default = FALSE

Value

A tibble. Or a list if .return_raw = TRUE.

See Also

Other Datasets Endpoints: [get_annotation\(\)](#), [get_collapsed_gene_model_exon\(\)](#), [get_downloads_page_data\(\)](#), [get_file_list\(\)](#), [get_full_get_collapsed_gene_model_exon\(\)](#), [get_functional_annotation\(\)](#), [get_linkage_disequilibrium_by_variant_data\(\)](#), [get_linkage_disequilibrium_data\(\)](#), [get_sample_datasets\(\)](#), [get_subject\(\)](#), [get_tissue_site_detail\(\)](#), [get_variant\(\)](#)

Examples

```
get_variant_by_location(  
  start = 153209600,  
  end = 153209700,  
  chromosome = "chr1"  
)
```

Index

- * **Admin Endpoints**
 - get_maintenance_message, 41
 - get_news_item, 50
- * **Biobank Data Endpoints**
 - download, 9
 - get_sample_biobank_data, 51
- * **Datasets Endpoints**
 - get_annotation, 11
 - get_collapsed_gene_model_exon, 18
 - get_downloads_page_data, 20
 - get_file_list, 25
 - get_full_get_collapsed_gene_model_exon, 28
 - get_functional_annotation, 29
 - get_linkage_disequilibrium_by_variant_data, 39
 - get_linkage_disequilibrium_data, 40
 - get_sample_datasets, 53
 - get_subject, 67
 - get_tissue_site_detail, 68
 - get_variant, 72
 - get_variant_by_location, 73
- * **Dynamic Association Endpoints**
 - calculate_expression_quantitative_trait_loci, 3
 - calculate_ieqtls, 5
 - calculate_isqtls, 7
 - calculate_splicing_quantitative_trait_loci, 8
- * **Expression Data Endpoints**
 - get_clustered_median_exon_expression, 12
 - get_clustered_median_gene_expression, 13
 - get_clustered_median_junction_expression, 15
 - get_clustered_median_transcript_expression, 16
 - get_expression_pca, 24
 - get_gene_expression, 31
 - get_median_exon_expression, 42
 - get_median_gene_expression, 43
 - get_median_junction_expression, 44
 - get_median_transcript_expression, 46
 - get_single_nucleus_gex, 63
 - get_single_nucleus_gex_summary, 64
 - get_top_expressed_genes, 69
- * **GTE Portal API Info**
 - get_service_info, 55
- * **Histology Endpoints**
 - get_image, 36
- * **Metadata Endpoints**
 - get_dataset_info, 19
- * **Reference Genome Endpoints**
 - get_exons, 23
 - get_gene_search, 33
 - get_genes, 30
 - get_genomic_features, 34
 - get_gwas_catalog_by_location, 35
 - get_neighbor_gene, 49
 - get_transcripts, 71
- * **Static Association Endpoints**
 - get_eqtl_genes, 21
 - get_fine_mapping, 26
 - get_independent_eqtl, 37
 - get_multi_tissue_eqtls, 47
 - get_significant_single_tissue_eqtls, 56
 - get_significant_single_tissue_eqtls_by_location, 57
 - get_significant_single_tissue_ieqtls, 59
 - get_significant_single_tissue_isqtls, 60
 - get_significant_single_tissue_sqtls, 61

- get_sqtl_genes, 66
- calculate_expression_quantitative_trait_loci, 3, 6–8
- calculate_ieqtls, 4, 5, 7, 8
- calculate_isqtls, 4, 6, 7, 8
- calculate_splicing_quantitative_trait_loci, 4, 6, 7, 8
- download, 9, 52
- get_annotation, 11, 19, 20, 26, 29, 30, 39, 41, 55, 68, 69, 73, 74
- get_clustered_median_exon_expression, 12, 14, 16, 17, 25, 32, 43–45, 47, 64, 65, 70
- get_clustered_median_gene_expression, 13, 13, 16, 17, 25, 32, 43–45, 47, 64, 65, 70
- get_clustered_median_junction_expression, 13, 14, 15, 17, 25, 32, 43–45, 47, 64, 65, 70
- get_clustered_median_transcript_expression, 13, 14, 16, 16, 25, 32, 43–45, 47, 64, 65, 70
- get_collapsed_gene_model_exon, 12, 18, 20, 26, 29, 30, 39, 41, 55, 68, 69, 73, 74
- get_dataset_info, 19
- get_downloads_page_data, 12, 19, 20, 26, 29, 30, 39, 41, 55, 68, 69, 73, 74
- get_eqtl_genes, 21, 27, 38, 48, 57, 58, 60–62, 67
- get_exons, 23, 31, 34–36, 49, 71
- get_expression_pca, 13, 14, 16, 17, 24, 32, 43–45, 47, 64, 65, 70
- get_file_list, 12, 19, 20, 25, 29, 30, 39, 41, 55, 68, 69, 73, 74
- get_fine_mapping, 22, 26, 38, 48, 57, 58, 60–62, 67
- get_full_get_collapsed_gene_model_exon, 12, 19, 20, 26, 28, 30, 39, 41, 55, 68, 69, 73, 74
- get_functional_annotation, 12, 19, 20, 26, 29, 29, 39, 41, 55, 68, 69, 73, 74
- get_gene_expression, 13, 14, 16, 17, 25, 31, 43–45, 47, 64, 65, 70
- get_gene_search, 24, 31, 33, 35, 36, 49, 71
- get_genes, 24, 30, 34–36, 49, 71
- get_genomic_features, 24, 31, 34, 34, 36, 49, 71
- get_gwas_catalog_by_location, 24, 31, 34, 35, 35, 49, 71
- get_image, 36
- get_independent_eqtl, 22, 27, 37, 48, 57, 58, 60–62, 67
- get_linkage_disequilibrium_by_variant_data, 12, 19, 20, 26, 29, 30, 39, 41, 55, 68, 69, 73, 74
- get_linkage_disequilibrium_data, 12, 19, 20, 26, 29, 30, 39, 40, 55, 68, 69, 73, 74
- get_maintenance_message, 41, 50
- get_median_exon_expression, 13, 14, 16, 17, 25, 32, 42, 44, 45, 47, 64, 65, 70
- get_median_gene_expression, 13, 14, 16, 17, 25, 32, 43, 43, 45, 47, 64, 65, 70
- get_median_junction_expression, 13, 14, 16, 17, 25, 32, 43, 44, 44, 47, 64, 65, 70
- get_median_transcript_expression, 13, 14, 16, 17, 25, 32, 43–45, 46, 64, 65, 70
- get_multi_tissue_eqtls, 22, 27, 38, 47, 57, 58, 60–62, 67
- get_neighbor_gene, 24, 31, 34–36, 49, 71
- get_news_item, 42, 50
- get_sample_biobank_data, 10, 51
- get_sample_datasets, 12, 19, 20, 26, 29, 30, 39, 41, 53, 68, 69, 73, 74
- get_service_info, 55
- get_significant_single_tissue_eqtls, 22, 27, 38, 48, 56, 58, 60–62, 67
- get_significant_single_tissue_eqtls_by_location, 22, 27, 38, 48, 57, 57, 60–62, 67
- get_significant_single_tissue_ieqtls, 22, 27, 38, 48, 57, 58, 59, 61, 62, 67
- get_significant_single_tissue_isqtls, 22, 27, 38, 48, 57, 58, 60, 60, 62, 67
- get_significant_single_tissue_sqtls, 22, 27, 38, 48, 57, 58, 60, 61, 61, 67
- get_single_nucleus_gex, 13, 14, 16, 17, 25, 32, 43–45, 47, 63, 65, 70
- get_single_nucleus_gex_summary, 13, 14, 16, 17, 25, 32, 43–45, 47, 64, 64, 70
- get_sqtl_genes, 22, 27, 38, 48, 57, 58, 60–62, 66

`get_subject`, [12](#), [19](#), [20](#), [26](#), [29](#), [30](#), [39](#), [41](#),
[55](#), [67](#), [69](#), [73](#), [74](#)
`get_tissue_site_detail`, [12](#), [19](#), [20](#), [26](#), [29](#),
[30](#), [39](#), [41](#), [55](#), [68](#), [68](#), [73](#), [74](#)
`get_tissue_site_detail()`, [4](#), [6–9](#), [13](#), [14](#),
[16](#), [17](#), [22](#), [24](#), [27](#), [32](#), [38](#), [42](#), [44–46](#),
[51](#), [54](#), [56](#), [58](#), [59](#), [61–63](#), [65](#), [66](#), [70](#)
`get_top_expressed_genes`, [13](#), [14](#), [16](#), [17](#),
[25](#), [32](#), [43–45](#), [47](#), [64](#), [65](#), [69](#)
`get_transcripts`, [24](#), [31](#), [34–36](#), [49](#), [71](#)
`get_variant`, [12](#), [19](#), [20](#), [26](#), [29](#), [30](#), [39](#), [41](#),
[55](#), [68](#), [69](#), [72](#), [74](#)
`get_variant_by_location`, [12](#), [19](#), [20](#), [26](#),
[29](#), [30](#), [39](#), [41](#), [55](#), [68](#), [69](#), [73](#), [73](#)