Package 'scfetch'

November 21, 2023

Type Package

Title Access and Format Single-Cell RNA-Seq Datasets from Public Resources

Version 0.5.0

Maintainer Yabing Song <songyb0519@gmail.com>

Description The goal of 'scfetch' is to access and format single-cell RNA-seq datasets. It can be used to download single-cell RNA-seq datasets from widely used public resources, including GEO <https://www.ncbi.nlm.nih.gov/geo/>, Zenodo <https://zenodo.org/>, CELLxGENE <https://cellxgene.cziscience.com/>, Human Cell Atlas <https://www.humancellatlas.org/>, PanglaoDB <https://panglaodb.se/index.html> and UCSC Cell Browser <https://cells.ucsc.edu/>. And, it can also be used to perform object conversion between SeuratObject <https://satijalab.org/seurat/>, loom <http://loompy.org/>, h5ad <https://scanpy.readthedocs.io/en/stable/>, SingleCellExperiment <https://bioconductor.org/packages/release/bioc/html/scran.html>, CellDataSet <http://cole-trapnell-lab.github.io/monocle-release/> and cell_data_set <https://cole-trapnell-lab.github.io/monocle3/>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

biocViews

Imports Biobase, curl, data.table, dplyr, GEOquery, jsonlite, magrittr, Matrix, openxlsx, parallel, pbapply, purrr, rPanglaoDB, reticulate, Seurat, tibble, tools, utils, SingleCellExperiment, SummarizedExperiment, scater, LoomExperiment, httr, rlang, tidyr, methods

Suggests knitr, rmarkdown, scRNAseq, BiocStyle, htmltools, sceasy, SeuratDisk, SeuratWrappers, zellkonverter, GEOfastq

VignetteBuilder knitr

Additional_repositories https://showteeth.github.io/drat/

URL https://github.com/showteeth/scfetch

BugReports https://github.com/showteeth/scfetch/issues

Depends R (>= 2.10)

NeedsCompilation no

Author Yabing Song [aut, cre]

Repository CRAN

Date/Publication 2023-11-21 18:20:05 UTC

R topics documented:

`Bam2Fastq` *Convert bam files to fastq files.*

Description

Convert bam files to fastq files.

Usage

```
Bam2Fastq(
  bam.folder = NULL,
  bam.path = NULL,
  bam.type = c("10x", "other"),
  pair.end = NULL,
  bamtofastq.path = NULL,
  bamtofastq.paras = "--nthreads 4",
  sort.name = FALSE,
  sort.thread = 4
)
```

Arguments

`bam.folder`	Folder contains bam files, obtained from `DownloadSRA`. Default: NULL.
`bam.path`	Paths of bams. `bam.folder` and `bam.path` cannot be both NULL. Default: NULL.
`bam.type`	The source of bam files, choose from 10x (e.g. CellRanger) or other. Default: 10x.
`pair.end`	The bam files are pair-end or single-end, used when `bam.type` is other. Default: NULL (identify with flag).
`bamtofastq.path`	Path to 10x bamtofastq (`bam.type` is 10x) or samtools (`bam.type` is other). Default: NULL (conduct automatic detection with bamtofastq_linux/samtools).
`bamtofastq.paras`	Parameters for `bamtofastq.path`. Default: "–nthreads 4".
`sort.name`	Logical value, whether the bam files are sorted by name, required when `bam.type` is other. Default: FALSE.
`sort.thread`	The number of threads for bam sorting, used when `bam.type` is other. Default: 4.

Value

NULL or paths of failed bams.

Examples

```
## Not run:
# need users to provide prefetch.path and bamtofastq.path
GSE138266.runs <- ExtractRun(acce = "GSE138266", platform = "GPL18573")
GSE138266.down <- DownloadBam(
  gsm.df = GSE138266.runs, prefetch.path = "/path/to/prefetch",
  out.folder = "/path/to/output"
)
GSE138266.convert <- Bam2Fastq(
  bam.folder = "/path/to/output",
  bamtofastq.path = "/path/to/bamtofastq_linux or samtools",
  bamtofastq.paras = "--nthreads 4"
)

## End(Not run)
```

DownloadBam *Download bam.*

Description

Download bam.

Usage

```
DownloadBam(
  gsm.df,
  bam.type = c("10x", "other"),
  prefetch.path = NULL,
  samdump.path = NULL,
  out.folder = NULL,
  prefetch.paras = "-X 100G",
  samdump.paras = ""
)
```

Arguments

gsm.df	Dataframe contains GSM and Run numbers, obtained from ExtractRun.
bam.type	The source of bam files to download, choose from 10x (e.g. CellRanger) or other. Default: 10x.
prefetch.path	Path to prefetch. Default: NULL (conduct automatic detection).
samdump.path	Path to sam-dump, used when bam.type is other. Default: NULL (conduct automatic detection).

out.folder	Output folder. Default: NULL (current working directory).
prefetch.paras	Parameters for prefetch. This should not contain –type or -T values. Default: "-X 100G".
samdump.paras	Parameters for sam-dump. Default: "".

Value

Dataframe contains failed runs or NULL.

Examples

```
## Not run:
# need users to provide prefetch.path
GSE138266.runs <- ExtractRun(acce = "GSE138266", platform = "GPL18573")
GSE138266.down <- DownloadBam(
  gsm.df = GSE138266.runs, bam.type = "10x",
  prefetch.path = "/path/to/prefetch",
  out.folder = "/path/to/output"
)

## End(Not run)
```

DownloadSRA *Download SRA.*

Description

Download SRA.

Usage

```
DownloadSRA(
  gsm.df,
  prefetch.path = NULL,
  out.folder = NULL,
  prefetch.paras = "-X 100G"
)
```

Arguments

gsm.df	Dataframe contains GSM and Run numbers, obtained from ExtractRun.
prefetch.path	Path to prefetch. Default: NULL (conduct automatic detection).
out.folder	Output folder. Default: NULL (current working directory).
prefetch.paras	Parameters for prefetch. Default: "-X 100G".

Value

Dataframe contains failed runs or NULL.

Examples

```
## Not run:
# need users to provide the prefetch.path and out.folder
GSE186003.runs <- ExtractRun(acce = "GSE186003", platform = "GPL24247")
GSE186003.down <- DownloadSRA(
  gsm.df = GSE186003.runs, prefetch.path = "/path/to/prefetch",
  out.folder = "/path/to/output"
)

## End(Not run)
```

ExportSeurat *Export SeuratObject to Other Formats.*

Description

Export SeuratObject to Other Formats.

Usage

```
ExportSeurat(
  seu.obj,
  assay = NULL,
  reduction = NULL,
  to = c("SCE", "AnnData", "CellDataSet", "cell_data_set", "loom"),
  anndata.file = NULL,
  loom.file = NULL,
  conda.path = NULL,
  ...
)
```

Arguments

seu.obj	A seurat object.
assay	Which assay to use. Default: NULL (get with DefaultAssay).
reduction	Name of DimReduc to set to main reducedDim in cds.
to	The target format, chosen from "SCE" (SingleCellExperiment), "AnnData", "CellDataSet", "cell_data_set", "loom". Default: "SCE".
anndata.file	File used to save AnnData results. Default: NULL.
loom.file	File used to save loom results. Default: NULL.
conda.path	Conda environment path, used when to is "AnnData". Default: NULL.
...	Parameter for as.SingleCellExperiment, sceasy::convertFormat, as.CellDataSet, as.cell_data_set, SaveLoom, corresponding to to.

Value

Object corresponding to to.

Examples

```
## Not run:
library(Seurat)
# export to SingleCellExperiment
sce.obj <- ExportSeurat(seu.obj = pbmc_small, assay = "RNA", to = "SCE")
# export to CellDataSet
cds.obj <- ExportSeurat(seu.obj = pbmc_small, assay = "RNA", reduction = "tsne", to = "CellDataSet")
# export to cell_data_set
cds3.obj <- ExportSeurat(seu.obj = pbmc_small, assay = "RNA", to = "cell_data_set")
# export to AnnData, need users to provide the conda path and the output file
ExportSeurat(
  seu.obj = pbmc_small, assay = "RNA", to = "AnnData", conda.path = "/path/to/anaconda3",
  anndata.file = "/path/to/pbmc_small.h5ad"
)
# export to loom, need users to provide the output file
ExportSeurat(
  seu.obj = pbmc_small, assay = "RNA", to = "loom",
  loom.file = "/path/to/pbmc_small.loom"
)

## End(Not run)
```

ExtractCBComposition *Extract Cell Type Composition of UCSC Cell Browser Datasets.*

Description

Extract Cell Type Composition of UCSC Cell Browser Datasets.

Usage

```
ExtractCBComposition(
  json.folder = NULL,
  sample.df = NULL,
  all.samples.df = NULL,
  collection = NULL,
  sub.collection = NULL,
  organ = NULL,
  disease = NULL,
  organism = NULL,
  project = NULL,
  fuzzy.match = TRUE,
  cell.num = NULL
)
```

Arguments

json.folder	Folder contains datasets json files, same as json.folder of ShowCBDatasets. Default: NULL (current working directory).
sample.df	Dataframe contains used datasets. Default: NULL.
all.samples.df	Dataframe contains all samples metadata, obtained with ShowCBDatasets. Default: NULL. sample.df and all.samples.df cannot be both NULL.
collection	The collection of the datasets, corresponds to shortLabel column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
sub.collection	The sub-collection of the datasets, corresponds to subLabel column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
organ	The organ of the datasets, corresponds to body_parts column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
disease	The disease of the datasets, corresponds to diseases column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
organism	The specie of the datasets, corresponds to organisms column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
project	The project of the datasets, corresponds to projects column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
fuzzy.match	Logical value, whether to perform fuzzy match with provided attribute values. Default: TRUE.
cell.num	Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL.

Value

Dataframe contains sample information and cell type composition.

Examples

```
## Not run:
# lazy mode, load datasets json files locally, need users to provide json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, json.folder = NULL, update = FALSE)
# cell number is between 1000 and 2000
hbb.sample.df <- ExtractCBDatasets(
  all.samples.df = ucsc.cb.samples, organ = c("brain", "blood"),
  organism = "Human (H. sapiens)", cell.num = c(1000, 2000)
)
hbb.sample.ct <- ExtractCBComposition(json.folder = NULL, sample.df = hbb.sample.df)

## End(Not run)
```

`ExtractCBDatasets` *Extract UCSC Cell Browser Datasets with Attributes.*

Description

Extract UCSC Cell Browser Datasets with Attributes.

Usage

```
ExtractCBDatasets(
  all.samples.df,
  collection = NULL,
  sub.collection = NULL,
  organ = NULL,
  disease = NULL,
  organism = NULL,
  project = NULL,
  fuzzy.match = TRUE,
  cell.num = NULL
)
```

Arguments

`all.samples.df`	Dataframe contains all samples metadata, obtained with `ShowCBDatasets`.
`collection`	The collection of the datasets, corresponds to `shortLabel` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`sub.collection`	The sub-collection of the datasets, corresponds to `subLabel` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`organ`	The organ of the datasets, corresponds to `body_parts` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`disease`	The disease of the datasets, corresponds to `diseases` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`organism`	The specie of the datasets, corresponds to `organisms` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`project`	The project of the datasets, corresponds to `projects` column of `ShowCBDatasets`, obtain available values with `StatDBAttribute`. Default: NULL (without filtering).
`fuzzy.match`	Logical value, whether to perform fuzzy match with provided attribute values. Default: TRUE.
`cell.num`	Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL(without filtering).

Value

Dataframe contains filtered datasets.

Examples

```
## Not run:
# lazy mode, load datasets json files locally, need users to provide json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, json.folder = NULL, update = FALSE)
# cell number is between 1000 and 2000
hbb.sample.df <- ExtractCBDatasets(
  all.samples.df = ucsc.cb.samples, organ = c("brain", "blood"),
  organism = "Human (H. sapiens)", cell.num = c(1000, 2000)
)

## End(Not run)
```

ExtractCELLxGENEMeta *Extract Metadata of CELLxGENE Datasets with Attributes.*

Description

Extract Metadata of CELLxGENE Datasets with Attributes.

Usage

```
ExtractCELLxGENEMeta(
  all.samples.df,
  organism = NULL,
  ethnicity = NULL,
  sex = NULL,
  tissue = NULL,
  disease = NULL,
  assay = NULL,
  suspension.type = NULL,
  cell.type = NULL,
  cell.num = NULL
)
```

Arguments

`all.samples.df`	All detail information of CELLxGENE datasets, obtained with `ShowCELLxGENEDatasets`.
`organism`	The organism of the datasets, choose from "Homo sapiens", "Mus musculus", "Callithrix jacchus", "Macaca mulatta", "Sus scrofa domesticus", one or multiple values. Default: NULL (All).

`ethnicity` The ethnicity of the datasets, choose from "Asian", "European", "unknown", "na", "African", "Bangladeshi", "British", "Irish", "East Asian", "African American", "Hispanic or Latin American", "African American or Afro-Caribbean", "European American", "Finnish", "Indian", "Japanese", "Korean", "Malaysian", "Singaporean Chinese", "American", "Pacific Islander", "admixed ancestry", "Eskimo", "Han Chinese", "Greater Middle Eastern (Middle Eastern, North African or Persian)", "multiethnic", "Jewish Israeli", "South Asian", "Oceanian", "Chinese", one or multiple values. Default: NULL (All).

`sex` The sex of the datasets, choose from "female", "male", "unknown", one or multiple values. Default: NULL (All).

`tissue` The tissue of the datasets, obtain available values with `StatDBAttribute`. One or multiple values. Default: NULL (All).

`disease` The disease of the datasets, obtain available values with `StatDBAttribute`. One or multiple values. Default: NULL (All).

`assay` The assay of the datasets, choose from "10x 3' v1", "10x 3' v2", "10x 3' v3", "10x 3' transcription profiling", "10x 5' v1", "10x 5' v2", "10x 5' transcription profiling", "10x multiome", "10x scATAC-seq", "sci-RNA-seq", "Drop-seq", "Smart-seq", "Smart-seq2", "Smart-seq v4", "snmC-Seq2", "Visium Spatial Gene Expression", "Seq-Well", "Seq-Well S3", "Patch-seq", "sci-Plex", "BD Rhapsody Targeted mRNA", "BD Rhapsody Whole Transcriptome Analysis", "Slide-seqV2", "GEXSCOPE technology", "inDrop", "microwell-seq", "CEL-seq2", "STRT-seq", "DroNc-seq", "MERFISH", "scATAC-seq", "MARS-seq", "TruDrop", one or multiple values. Default: NULL (All).

`suspension.type` The suspension type of the datasets, choose from "nucleus", "cell", "na", one or multiple values. Default: NULL (All).

`cell.type` The cell type of the datasets, obtain available values with `StatDBAttribute`. One or multiple values. Default: NULL (All).

`cell.num` Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL(without filtering).

Value

Dataframe contains filtered datasets.

References

https://gist.github.com/ivirshup/f1a1603db69de3888eacb4bdb6a9317a

Examples

```
# all available datasets
all.cellxgene.datasets <- ShowCELLxGENEDatasets()
# human 10x v2 and v3 datasets
human.10x.cellxgene.meta <- ExtractCELLxGENEMeta(
  all.samples.df = all.cellxgene.datasets,
  assay = c("10x 3' v2", "10x 3' v3"),
```

```
  organism = "Homo sapiens"
)
```

`ExtractGEOExp` *Extract Raw Count Matrix or Fortmat Supplementary Files to 10x.*

Description

Extract Raw Count Matrix or Fortmat Supplementary Files to 10x.

Usage

```
ExtractGEOExp(
  pf.obj,
  acce,
  supp.idx = 1,
  down.supp = FALSE,
  timeout = 3600,
  supp.type = c("count", "10x"),
  out.folder = NULL,
  gene2feature = TRUE
)
```

Arguments

`pf.obj`	GEO object of platform.
`acce`	GEO accession number.
`supp.idx`	The index of supplementary files to download. Default: 1.
`down.supp`	Logical value, whether to download supplementary files to create count matrix. If TRUE, always download supplementary files. If FALSE, use `ExpressionSet` (If contains non-integer or emoty, download supplementary files automatically). Default: FALSE.
`timeout`	Timeout for `download.file`. Default: 3600.
`supp.type`	The type of downloaded supplementary files, choose from count (count matrix file or single count matrix file) and 10x (cellranger output files, contains barcodes, genes/features and matrix). Default: count.
`out.folder`	Output folder to save 10x files. Default: NULL (current working directory).
`gene2feature`	Logical value, whether to rename `genes.tsv.gz` to `features.tsv.gz`. Default: TRUE.

Value

Count matrix (`supp.type` is count) or NULL (`supp.type` is 10x).

ExtractGEOExpSupp *Extract Raw Count Matrix from Supplementary Files.*

Description

Extract Raw Count Matrix from Supplementary Files.

Usage

```
ExtractGEOExpSupp(acce, timeout = 3600, supp.idx = 1)
```

Arguments

acce	GEO accession number.
timeout	Timeout for download.file. Default: 3600.
supp.idx	The index of supplementary files to download. Default: 1.

Value

A dataframe.

ExtractGEOExpSupp10x *Fortmat Supplementary Files to 10x.*

Description

Fortmat Supplementary Files to 10x.

Usage

```
ExtractGEOExpSupp10x(
  acce,
  supp.idx = 1,
  timeout = 3600,
  out.folder = NULL,
  gene2feature = TRUE
)
```

Arguments

acce	GEO accession number.
supp.idx	The index of supplementary files to download. Default: 1.
timeout	Timeout for download.file. Default: 3600.
out.folder	Output folder to save 10x files. Default: NULL (current working directory).
gene2feature	Logical value, whether to rename genes.tsv.gz to features.tsv.gz. Default: TURE.

ExtractGEOExpSuppAll *Extract Raw Count Matrix from Supplementary Files or Fortmat Supplementary Files to 10x.*

Description

Extract Raw Count Matrix from Supplementary Files or Fortmat Supplementary Files to 10x.

Usage

```
ExtractGEOExpSuppAll(
  acce,
  supp.idx = 1,
  timeout = 3600,
  supp.type = c("count", "10x"),
  out.folder = NULL,
  gene2feature = TRUE
)
```

Arguments

acce	GEO accession number.
supp.idx	The index of supplementary files to download. Default: 1.
timeout	Timeout for download.file. Default: 3600.
supp.type	The type of downloaded supplementary files, choose from count (count matrix file or single count matrix file) and 10x (cellranger output files, contains barcodes, genes/features and matrix). Default: count.
out.folder	Output folder to save 10x files. Default: NULL (current working directory).
gene2feature	Logical value, whether to rename genes.tsv.gz to features.tsv.gz. Default: TRUE. Default: TURE.

Value

Count matrix (supp.type is count) or NULL (supp.type is 10x).

ExtractGEOInfo *Extract GEO Study Information.*

Description

Extract GEO Study Information.

Usage

```
ExtractGEOInfo(pf.obj, sample.wise = FALSE)
```

Arguments

pf.obj	GEO object of platform.
sample.wise	Logical value, whether to extract sample-wise information. Default: FALSE.

Value

A dataframe.

ExtractGEOMeta *Extract Sample Metadata from GEO.*

Description

Extract Sample Metadata from GEO.

Usage

```
ExtractGEOMeta(acce, platform = NULL, ...)
```

Arguments

acce	GEO accession number.
platform	Platform information/field. Default: NULL (all platforms).
...	Parameters for getGEO.

Value

Dataframe contains all metadata of provided GEO accession number.

Examples

```
# users may need to set the size of the connection buffer
# Sys.setenv("VROOM_CONNECTION_SIZE" = 131072 * 60)
# extract metadata of specified platform
GSE200257.meta <- ExtractGEOMeta(acce = "GSE200257", platform = "GPL24676")
```

ExtractGEOSubMeta *Extract Sample Metadata.*

Description

Extract Sample Metadata.

Usage

```
ExtractGEOSubMeta(pf.obj)
```

Arguments

pf.obj	GEO object of platform.

Value

A dataframe.

ExtractHCAMeta *Extract Metadata of Human Cell Atlas Projects with Attributes.*

Description

Extract Metadata of Human Cell Atlas Projects with Attributes.

Usage

```
ExtractHCAMeta(
  all.projects.df,
  organism = NULL,
  sex = NULL,
  organ = NULL,
  organ.part = NULL,
  disease = NULL,
  sample.type = NULL,
  preservation.method = NULL,
  protocol = NULL,
  suspension.type = NULL,
  cell.type = NULL,
  cell.num = NULL,
  sequencing.type = NULL
)
```

Arguments

`all.projects.df`	All detail information of HCA projects, obtained with `ShowHCAProjects`.
`organism`	The organism of the projects, choose from "Homo sapiens", "Mus musculus", "Macaca mulatta", "canis lupus familiaris", one or multiple values. Default: NULL (All).
`sex`	The sex of the projects, choose from "female", "male", "mixed", "unknown", one or multiple values. Default: NULL (All).
`organ`	The organ of the projects (e.g. brain), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`organ.part`	The organ part of the projects (e.g. cortex), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`disease`	The disease of the projects (e.g. normal), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`sample.type`	The sex of the projects, choose from "specimens", "organoids", "cellLines", one or multiple values. Default: NULL (All).
`preservation.method`	The preservation method of the projects (e.g. fresh), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`protocol`	The protocol of the projects (e.g. 10x 3' v2), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`suspension.type`	The suspension type of the projects, choose from "single cell", "single nucleus", "bulk cell", "bulk nuclei", one or multiple values. Default: NULL (All).
`cell.type`	The cell type of the projects (e.g. neuron), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).
`cell.num`	Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL(without filtering).
`sequencing.type`	The sequencing instrument type of the projects (e.g. illumina hiseq 2500), obtain available values with `StatDBAttribute`, one or multiple values. Default: NULL (All).

Value

Dataframe contains filtered projects.

References

https://bioconductor.org/packages/release/bioc/html/hca.html

Examples

```
# all available projects
```

```
all.hca.projects <- ShowHCAProjects()
# all human projects
all.human.projects <- ExtractHCAMeta(all.projects.df = all.hca.projects, organism = "Homo sapiens")
# all human and 10x 3' v2
all.human.10x.projects <- ExtractHCAMeta(
  all.projects.df = all.hca.projects,
  organism = "Homo sapiens",
  protocol = c("10x 3' v2", "10x 3' v3")
)
```

`ExtractPanglaoDBComposition` *Extract Cell Type Composition of PanglaoDB Datasets.*

Description

Extract Cell Type Composition of PanglaoDB Datasets.

Usage

```
ExtractPanglaoDBComposition(
  sra = NULL,
  srs = NULL,
  species = NULL,
  protocol = NULL,
  tissue = NULL,
  local.data = TRUE
)
```

Arguments

`sra`	The SRA identifier of the datasets, obtain available values with `StatDBAttribute`, one or multiple value. Default: NULL (All).
`srs`	The SRS identifier of the datasets, obtain available values with `StatDBAttribute`, one or multiple value. Default: NULL (All).
`species`	The species of the datasets, choose from "Homo sapiens", "Mus musculus", one or multiple value. Default: NULL (All).
`protocol`	Protocol used to generate the datasets, choose from "10x chromium", "drop-seq", "microwell-seq", "C1 Fluidigm", "inDrops", "Smart-seq2", "CEL-seq", one or multiple value. Default: NULL (All).
`tissue`	The tissue of the datasets, obtain available values with `StatDBAttribute`, one or multiple value. Default: NULL (All).
`local.data`	Logical value, whether to use local data (PanglaoDB is no longer maintained). Default: TRUE.

Value

Dataframe contains sample metadata, cluster, cell number and cell type information.

Examples

```
human.composition <- ExtractPanglaoDBComposition(
  species = "Homo sapiens",
  protocol = c("Smart-seq2", "10x chromium")
)
```

`ExtractPanglaoDBMeta` *Extract Metadata of scRNA-seq Datasets in PanglaoDB.*

Description

Extract Metadata of scRNA-seq Datasets in PanglaoDB.

Usage

```
ExtractPanglaoDBMeta(
  species = NULL,
  protocol = NULL,
  tissue = NULL,
  cell.num = NULL,
  show.cell.type = TRUE,
  local.data = TRUE
)
```

Arguments

`species`	The species of the datasets, choose from "Homo sapiens", "Mus musculus", one or multiple value. Default: NULL (All).
`protocol`	Protocol used to generate the datasets, choose from "10x chromium", "drop-seq", "microwell-seq", "C1 Fluidigm", "inDrops", "Smart-seq2", "CEL-seq", one or multiple value. Default: NULL (All).
`tissue`	The tissue of the datasets, obtain available values with `StatDBAttribute`. Default: NULL (All).
`cell.num`	Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL.
`show.cell.type`	Logical value, whether to show inferred cell type. Default: TRUE.
`local.data`	Logical value, whether to use local data (PanglaoDB is no longer maintained). Default: TRUE.

Value

Dataframe contains SRA, SRS, Tissue, Protocol, Species, Cells, CellType (inferred).

Examples

```
human.meta <- ExtractPanglaoDBMeta(
  species = "Homo sapiens",
  protocol = c("Smart-seq2", "10x chromium"),
  cell.num = c(1000, 2000)
)
```

`ExtractRun` *Extract Runs with GEO Accession Number or GSM Number.*

Description

Extract Runs with GEO Accession Number or GSM Number.

Usage

```
ExtractRun(gsm = NULL, acce = NULL, platform = NULL, parallel = TRUE, ...)
```

Arguments

`gsm`	GSM number. Default: NULL (use `acce`).
`acce`	GEO accession number. Default: NULL (use `gsm`). `acce` and `gsm` cannot be both NULL.
`platform`	Platform information/field. Default: NULL (all platforms).
`parallel`	Logical value, whether to process GSM parallelly. Default: TRUE.
`...`	Parameters for `getGEO`. Used when `acce` is not NULL.

Value

Dataframe contains GSM and Runs.

Examples

```
## Not run:
GSE186003.runs <- ExtractRun(acce = "GSE186003", platform = "GPL24247", parallel = FALSE)

## End(Not run)
```

`ExtractZenodoMeta` *Prepare Dataframe with Zenodo DOIs.*

Description

Prepare Dataframe with Zenodo DOIs.

Usage

```
ExtractZenodoMeta(doi, file.ext = c("rdata", "h5ad"))
```

Arguments

`doi`	A vector of Zenodo DOIs, should start with "10.5281/zenodo.".
`file.ext`	The valid file extension for download. When NULL, use all files. Default: c("rdata", "h5ad").

Value

Dataframe contains files with valid extension in given Zenodo DOI.

Examples

```
zebrafish.df <- ExtractZenodoMeta(doi = "10.5281/zenodo.7243603")
ExtractZenodoMeta(doi = "10.5281/zenodo.48065") # Restricted Access
# vector of dois
multi.dois <- ExtractZenodoMeta(doi = c(
  "1111", "10.5281/zenodo.7243603",
  "10.5281/zenodo.7244441"
))
```

`ImportSeurat` *Convert Other Formats to SeuratObject.*

Description

Convert Other Formats to SeuratObject.

Usage

```
ImportSeurat(
  obj = NULL,
  assay = "RNA",
  from = c("SCE", "AnnData", "CellDataSet", "cell_data_set", "loom"),
  count.assay = "counts",
  data.assay = "logcounts",
  slot = "counts",
  anndata.file = NULL,
  loom.file = NULL,
  conda.path = NULL,
  ...
)
```

Arguments

obj	Other formats object (eg: SingleCellExperiment, CellDataSet). Default: NULL (used when from is "AnnData").
assay	Assay name to store expression matrices in SeuratObject. Default: RNA.
from	The source formats, chosen from "SCE" (SingleCellExperiment), "AnnData", "CellDataSet", "cell_data_set". Default: "SCE".
count.assay	The assay of source formats to save raw counts, used when from is "SCE" or cell_data_set. Default: counts.
data.assay	The assay of source formats to save log transformed counts, used when from is "SCE" or cell_data_set. Default: logcounts.
slot	Slot to store expression data as, used when from is "CellDataSet". Default: counts.
anndata.file	The file contains AnnData. Default: NULL.
loom.file	The file contains loom. Default: NULL.
conda.path	Conda environment path, used when from is "AnnData". Default: NULL.
...	Parameter for as.Seurat, sceasy::convertFormat, as.Seurat, as.Seurat, as.Seurat, corresponding to from.

Value

A Seurat object.

Examples

```
## Not run:
# import data from SingleCellExperiment
seu.obj <- ImportSeurat(
  obj = sce.obj, from = "SCE", count.assay = "counts",
  data.assay = "logcounts", assay = "RNA"
)
# import data from CellDataSet
```

```
seu.obj <- ImportSeurat(obj = cds.obj, from = "CellDataSet", count.assay = "counts", assay = "RNA")
# import data from cell_data_set
seu.obj <- ImportSeurat(
  obj = sce.obj, from = "cell_data_set", count.assay = "counts",
  data.assay = "logcounts", assay = "RNA"
)
# import data from AnnData, need users to provide the file for conversion
seu.obj <- ImportSeurat(anndata.file = "path/to/h5ad", from = "AnnData", assay = "RNA")
# import data from loom, need users to provide the file for conversion
seu.obj <- ImportSeurat(loom.file = "path/to/loom", from = "loom")

## End(Not run)
```

`PanglaoDBComposition` *All Sample Composition of PanglaoDB Datasets*

Description

A dataset contains all sample composition of PanglaoDB datasets.

Usage

```
PanglaoDBComposition
```

Format

A data frame with 19,449 rows and 8 variables:

SRA The SRA identifier

SRS The SRS identifier

Tissue The tissue of the sample

Protocol The protocol used to generate this sample

Species The species of this sample

Cluster Seurat cluster

Cells Cluster cell number

Cell Type Predicted cluster cell types

PanglaoDBMeta *All Sample Metadata of PanglaoDB Datasets*

Description

A dataset contains all sample metadata and cell types of PanglaoDB datasets.

Usage

```
PanglaoDBMeta
```

Format

A data frame with 1,368 rows and 7 variables:

SRA The SRA identifier

SRS The SRS identifier

Tissue The tissue of the sample

Protocol The protocol used to generate this sample

Species The species of this sample

Cells Total cell number of this sample

CellType Predicted cell types, separated by comma

ParseCBDatasets *Download UCSC Cell Browser Datasets.*

Description

Download UCSC Cell Browser Datasets.

Usage

```
ParseCBDatasets(
  sample.df = NULL,
  all.samples.df = NULL,
  collection = NULL,
  sub.collection = NULL,
  organ = NULL,
  disease = NULL,
  organism = NULL,
  project = NULL,
  fuzzy.match = TRUE,
  cell.num = NULL,
  timeout = 1000,
  merge = TRUE
)
```

Arguments

sample.df	Dataframe contains used datasets. Default: NULL.
all.samples.df	Dataframe contains all samples metadata, obtained with ShowCBDatasets. Default: NULL. sample.df and all.samples.df cannot be both NULL.
collection	The collection of the datasets, corresponds to shortLabel column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
sub.collection	The sub-collection of the datasets, corresponds to subLabel column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
organ	The organ of the datasets, corresponds to body_parts column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
disease	The disease of the datasets, corresponds to diseases column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
organism	The specie of the datasets, corresponds to organisms column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
project	The project of the datasets, corresponds to projects column of all.samples.df, obtain available values with StatDBAttribute. Default: NULL (without filtering).
fuzzy.match	Logical value, whether to perform fuzzy match with provided attribute values. Default: TRUE.
cell.num	Cell number filter. If NULL, no filter; if one value, lower filter; if two values, low and high filter. Deault: NULL.
timeout	Maximum request time when loading data online. Default: 1000.
merge	Logical value, whether to merge Seurat list. Default: FALSE.

Value

Seurat object (if merge is TRUE) or list of Seurat objects (if merge is FALSE).

Examples

```
## Not run:
# lazy mode, load datasets json files locally, need users to provide json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, json.folder = NULL, update = FALSE)
# cell number is between 1000 and 2000
hbb.sample.df <- ExtractCBDatasets(
  all.samples.df = ucsc.cb.samples, organ = c("brain", "blood"),
  organism = "Human (H. sapiens)", cell.num = c(1000, 2000)
)
hbb.sample.seu <- ParseCBDatasets(sample.df = hbb.sample.df)
# test 10x and matrix load
complex.df <- ucsc.cb.samples[c(1, 927, 379), ] # two 10x and one matrix
```

```
complex.seu.list <- ParseCBDatasets(sample.df = test.df, merge = F)

## End(Not run)
```

`ParseCELLxGENE` *Download CELLxGENE Datasets.*

Description

Download CELLxGENE Datasets.

Usage

```
ParseCELLxGENE(
  meta,
  file.ext = c("rds", "h5ad"),
  out.folder = NULL,
  timeout = 3600,
  quiet = FALSE,
  parallel = TRUE
)
```

Arguments

`meta`	Metadata used to download, can be from `ExtractCELLxGENEMeta`, should contain dataset_id, rds_id/h5ad_id (depend on `file.ext`) and name columns.
`file.ext`	The valid file extension for download. When NULL, use "rds" and "h5ad". Default: c("rds", "h5ad").
`out.folder`	The output folder. Default: NULL (current working directory).
`timeout`	Maximum request time. Default: 3600.
`quiet`	Logical value, whether to show downloading progress. Default: FALSE (show).
`parallel`	Logical value, whether to download parallelly. Default: TRUE. When "libcurl" is available for `download.file`, the parallel is done by default (`parallel` can be FALSE).

Value

Dataframe contains failed datasets or NULL.

References

https://gist.github.com/ivirshup/f1a1603db69de3888eacb4bdb6a9317a

Examples

```
## Not run:
# all available datasets
all.cellxgene.datasets <- ShowCELLxGENEDatasets()
# human 10x v2 and v3 datasets
human.10x.cellxgene.meta <- ExtractCELLxGENEMeta(
  all.samples.df = all.cellxgene.datasets,
  assay = c("10x 3' v2", "10x 3' v3"),
  organism = "Homo sapiens"
)
# download, need to provide the output folder
ParseCELLxGENE(meta = human.10x.cellxgene.meta, out.folder = "/path/to/output")

## End(Not run)
```

ParseGEO *Download Matrix from GEO and Load to Seurat.*

Description

Download Matrix from GEO and Load to Seurat.

Usage

```
ParseGEO(
  acce,
  platform = NULL,
  down.supp = FALSE,
  supp.idx = 1,
  timeout = 3600,
  data.type = c("sc", "bulk"),
  supp.type = c("count", "10x"),
  out.folder = NULL,
  gene2feature = TRUE,
  merge = TRUE,
  ...
)
```

Arguments

acce	GEO accession number.
platform	Platform information/field. Disable when down.supp is TRUE. Default: NULL (disable).
down.supp	Logical value, whether to download supplementary files to create count matrix. If TRUE, always download supplementary files. If FALSE, use ExpressionSet (If contains non-integer or empty, download supplementary files automatically). Default: FALSE.

supp.idx	The index of supplementary files to download. This should be consistent with platform. Default: 1.
timeout	Timeout for download.file. Default: 3600.
data.type	The data type of the dataset, choose from "sc" (single-cell) and "bulk" (bulk). Default: "sc".
supp.type	The type of downloaded supplementary files, choose from count (count matrix file or single count matrix file) and 10x (cellranger output files, contains barcodes, genes/features and matrix). Default: count.
out.folder	Output folder to save 10x files. Default: NULL (current working directory).
gene2feature	Logical value, whether to rename genes.tsv.gz to features.tsv.gz. Default: TRUE.
merge	Logical value, whether to merge Seurat list when there are multiple 10x files (supp.type is 10x). Default: FALSE.
...	Parameters for getGEO.

Value

If data.type is "sc", return Seurat object (if merge is TRUE) or Seurat object list (if merge is FALSE). If data.type is "bulk", return count matrix.

Examples

```
## Not run:
# the supp files are count matrix
GSE94820.seu <- ParseGEO(acce = "GSE94820", down.supp = TRUE, supp.idx = 1, supp.type = "count")
# the supp files are cellranger output files: barcodes, genes/features and matrix
# need users to provide the output folder
GSE200257.seu <- ParseGEO(
  acce = "GSE200257", down.supp = TRUE, supp.idx = 1, supp.type = "10x",
  out.folder = "/path/to/output/folder"
)

## End(Not run)
```

ParseHCA *Download Human Cell Atlas Datasets.*

Description

Download Human Cell Atlas Datasets.

Usage

```
ParseHCA(
  meta,
  file.ext = c("rds", "rdata", "h5", "h5ad", "loom"),
  out.folder = NULL,
  timeout = 3600,
  quiet = FALSE,
  parallel = TRUE
)
```

Arguments

meta	Metadata used to download, can be from ExtractHCAMeta, should contain entryId and name catalog.
file.ext	The valid file extension for download. When NULL, use "rds", "rdata", "h5", "h5ad", "loom". Default: c("rds", "rdata", "h5", "h5ad", "loom").
out.folder	The output folder. Default: NULL (current working directory).
timeout	Maximum request time. Default: 3600.
quiet	Logical value, whether to show downloading progress. Default: FALSE (show).
parallel	Logical value, whether to download parallelly. Default: TRUE. When "libcurl" is available for download.file, the parallel is done by default (parallel can be FALSE).

Value

Dataframe contains failed projects or NULL.

Examples

```
## Not run:
# all available projects
all.hca.projects <- ShowHCAProjects()
# all human and 10x 3' v2
all.human.10x.projects <- ExtractHCAMeta(
  all.projects.df = all.hca.projects,
  organism = "Homo sapiens",
  protocol = c("10x 3' v2", "10x 3' v3")
)
# download, need users to provide the output folder
ParseHCA(meta = all.human.10x.projects, out.folder = "/path/to/output")

## End(Not run)
```

ParsePanglaoDB *Parse PanglaoDB Data.*

Description

Parse PanglaoDB Data.

Usage

```
ParsePanglaoDB(
  meta,
  cell.type = "All",
  include.gene = NA,
  exclude.gene = NA,
  merge = FALSE
)
```

Arguments

meta	Metadata contains "SRA", "SRS", "Tissue", "Protocol", "Species", can be obtained with ExtractPanglaoDBMeta.
cell.type	Extract samples with specified cell types. For samples without SRS (notused), this value can only be "All" or "None", or these samples will be filtered. Obtain available values with StatDBAttribute, one or multiple value. Default: "All".
include.gene	Include cells expressing the genes. Default: NA.
exclude.gene	Exclude cells expressing the genes. Default: NA.
merge	Logical value, whether to merge Seurat list. Default: FALSE.

Value

Seurat object (if merge is TRUE) or list of Seurat objects (if merge is FALSE).

Examples

```
## Not run:
hsa.meta <- ExtractPanglaoDBMeta(
  species = "Homo sapiens",
  protocol = c("Smart-seq2", "10x chromium"),
  show.cell.type = TRUE, cell.num = c(1000, 2000)
)
hsa.seu <- ParsePanglaoDB(hsa.meta, merge = TRUE)

## End(Not run)
```

ParseZenodo *Download Data with Zenodo DOI.*

Description

Download Data with Zenodo DOI.

Usage

```
ParseZenodo(
  doi = NULL,
  file.ext = c("rdata", "rds", "h5ad"),
  doi.df = NULL,
  out.folder = NULL,
  timeout = 1000,
  quiet = FALSE,
  parallel = TRUE
)
```

Arguments

doi	A vector of Zenodo DOIs to download. Default: NULL.
file.ext	The valid file extension for download. When NULL, use all files. Default: c("rdata", "rds", "h5ad").
doi.df	DOI dataframe for download. This is useful when something wrong happens in downloading (e.g. MD5 verification failure, DownloadZenodo will return a dataframe contains failure terms.). Default: NULL. It is required to provide either doi or doi.df.
out.folder	The output folder. Default: NULL (current working directory).
timeout	Maximum request time. Default: 1000.
quiet	Logical value, whether to show downloading progress. Default: FALSE (show).
parallel	Logical value, whether to download parallelly. Default: TRUE. When "libcurl" is available for download.file, the parallel is done by default (parallel can be FALSE).

Value

When successful, NULL. When MD5 verification failure, a dataframe contains failure terms.

Examples

```
## Not run:
# need users to provide the output folder
multi.dois.parse <- ParseZenodo(
  doi = c(
    "1111", "10.5281/zenodo.7243603",
```

```
    "10.5281/zenodo.7244441"
  ),
  file.ext = c("rdata", "rds"),
  out.folder = "/path/to/outfoder"
)

## End(Not run)
```

`SCEAnnData` *Data Format Conversion between SingleCellExperiment and AnnData.*

Description

Data Format Conversion between SingleCellExperiment and AnnData.

Usage

```
SCEAnnData(
  from = c("SingleCellExperiment", "AnnData"),
  to = c("AnnData", "SingleCellExperiment"),
  sce = NULL,
  anndata.file = NULL,
  slot.name = "counts",
  ...
)
```

Arguments

`from`	The source data format to convert, chosen from SingleCellExperiment and AnnData. Default: SingleCellExperiment.
`to`	The target data format to convert, chosen from AnnData and SingleCellExperiment. Default: AnnData.
`sce`	The SingleCellExperiment object to convert. Default: NULL.
`anndata.file`	File used to save or contains AnnData results. Default: NULL.
`slot.name`	Slot name used to save count matrix, used when converting from AnnData to SingleCellExperiment. Default: counts.
`...`	Parameters for `writeH5AD` and `readH5AD`.

Value

NULL or SingleCellExperiment.

Examples

```
## Not run:
library(scRNAseq)
seger <- SegerstolpePancreasData()
SCEAnnData(from = "SingleCellExperiment", to = "AnnData", sce = seger, X_name = "counts")
# need users to provide the output file
sce <- SCEAnnData(
  from = "AnnData", to = "SingleCellExperiment",
  anndata.file = "path/to/seger.h5ad"
)

## End(Not run)
```

SCELoom *Data Format Conversion between SingleCellExperiment and loom.*

Description

Data Format Conversion between SingleCellExperiment and loom.

Usage

```
SCELoom(
  from = c("SingleCellExperiment", "loom"),
  to = c("loom", "SingleCellExperiment"),
  sce = NULL,
  loom.file = NULL,
  ...
)
```

Arguments

from	The source data format to convert, chosen from SingleCellExperiment and AnnData. Default: SingleCellExperiment.
to	The target data format to convert, chosen from AnnData and SingleCellExperiment. Default: loom.
sce	The SingleCellExperiment object to convert. Default: NULL.
loom.file	File used to save or contains loom results. Default: NULL.
...	Parameters for `sceasy::convertFormat` and `sceasy::convertFormat`.

Value

NULL or SingleCellExperiment.

Examples

```
## Not run:
# convert from loom to SingleCellExperiment, need users to provide the loom file
sce.obj <- SCELoom(
  from = "loom", to = "SingleCellExperiment",
  loom.file = "path/to/loom"
)
# convert from SingleCellExperiment to loom, need users to provide the loom file
SCELoom(
  from = "SingleCellExperiment", to = "loom", sce = sce.obj,
  loom.file = "path/to/loom"
)

## End(Not run)
```

`ShowCBDatasets` *Show All Available Datasets in UCSC Cell Browser.*

Description

Show All Available Datasets in UCSC Cell Browser.

Usage

```
ShowCBDatasets(lazy = TRUE, json.folder = NULL, update = FALSE, quiet = FALSE)
```

Arguments

`lazy`	Logical value, whether to always load datasets online or locally. Default: TRUE (load locally).
`json.folder`	Folder used to save or load datasets json files. Default: NULL (current working directory).
`update`	Logical value, whther to update local datasets json. Default: FALSE. For the first time, you should set `lazy` TRUE and `update` TRUE to save json files.
`quiet`	Logical value, whether to show downloading progress. Default: FALSE (show).

Value

Dataframe contains all available datasets.

Examples

```
## Not run:
# first time run (lazy mode), need users to provide json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, update = TRUE)
# second time run (lazy mode), need users to provide json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, update = FALSE)

## End(Not run)
```

ShowCELLxGENEDatasets *Show All Available Datasets in CELLxGENE.*

Description

Show All Available Datasets in CELLxGENE.

Usage

```
ShowCELLxGENEDatasets()
```

Value

Dataframe contains all available datasets.

References

https://gist.github.com/ivirshup/f1a1603db69de3888eacb4bdb6a9317a

Examples

```
# all available datasets
all.cellxgene.datasets <- ShowCELLxGENEDatasets()
```

ShowHCAProjects *Show All Available Projects in Human Cell Atlas.*

Description

Show All Available Projects in Human Cell Atlas.

Usage

```
ShowHCAProjects(catalog = NULL)
```

Arguments

catalog	The catalog of the projects. Different catalogs may share some projects. Choose from "dcp29", "dcp30", "dcp1", "lm2", "lm3", one or multiple values. Default: NULL (all catalogs, remove duplicated projects).

Value

Dataframe contains all available projects.

Examples

```
# all available projects
all.hca.projects <- ShowHCAProjects()
```

SplitSRA *Split SRA to fastq Files and Format to 10x Standard Style.*

Description

Split SRA to fastq Files and Format to 10x Standard Style.

Usage

```
SplitSRA(
  sra.folder = NULL,
  sra.path = NULL,
  fastq.type = c("10x", "other"),
  split.cmd.path = NULL,
  sratools.path = NULL,
  split.cmd.paras = NULL,
  split.cmd.threads = NULL,
  format.10x = TRUE,
  remove.raw = FALSE
)
```

Arguments

sra.folder	Folder contains all sras, obtained from DownloadSRA. Default: NULL.
sra.path	Paths of sras. sra.folder and sra.path cannot be both NULL. Default: NULL.
fastq.type	The source of fastq files, choose from 10x (use --split-files to split sra) or other (use --split-3 to split sra). Default: 10x.
split.cmd.path	The full command path used to split, can be path to parallel-fastq-dump, fasterq-dump and fastq-dump. Default: NULL (conduct automatic detection).
sratools.path	Path to sratoolkit bin. When split.cmd.path is path to parallel-fastq-dump, it requires sratoolkit. Default: NULL (conduct automatic detection).
split.cmd.paras	Parameters for split.cmd.path. Default: NULL.
split.cmd.threads	Threads, used when split.cmd.path is path to parallel-fastq-dump or fasterq-dump. Default: NULL (1).
format.10x	Logical value, whether to format split fastqs to 10x standard format. Default: TRUE.
remove.raw	Logical value, whether to remove old split fastqs (unformatted), used when format.10x is TRUE. Default: FALSE.

Value

NULL or paths of failed sras.

Examples

```
## Not run:
# need users to provide the prefetch.path, sra.folder, split.cmd.path, sratools.path and out.folder
GSE186003.runs <- ExtractRun(acce = "GSE186003", platform = "GPL24247")
GSE186003.down <- DownloadSRA(
  gsm.df = GSE186003.runs, prefetch.path = "/path/to/prefetch",
  out.folder = "/path/to/output"
)
GSE186003.split <- SplitSRA(
  sra.folder = "/path/to/output",
  split.cmd.path = "/path/to/parallel-fastq-dump",
  sratools.path = "/path/to/sra/bin", fastq.type = "10x",
  split.cmd.threads = 4
)

## End(Not run)
```

StatDBAttribute *Stat Database Attributes.*

Description

Stat Database Attributes.

Usage

```
StatDBAttribute(
  df,
  filter,
  database = c("PanglaoDB", "UCSC", "CELLxGENE", "HCA")
)
```

Arguments

df	All metadata, can be PanglaoDBMeta and obtained with ShowCBDatasets, ShowCELLxGENEDatasets, and ShowHCAProjects.
filter	Vector of attributes.
database	Database name, choose from "PanglaoDB", "UCSC", "CELLxGENE", "HCA". Default: "PanglaoDB".

Value

List of attributes information, including attribute, value and number.

Examples

```
## Not run:
# PanglaoDB
StatDBAttribute(df = PanglaoDBMeta, filter = c("species", "protocol"), database = "PanglaoDB")
# UCSC Cell Browser, need users to provide the json folder
ucsc.cb.samples <- ShowCBDatasets(lazy = TRUE, json.folder = NULL, update = FALSE)
StatDBAttribute(df = ucsc.cb.samples, filter = c("organism", "organ"), database = "UCSC")
# CELLxGENE
all.cellxgene.datasets <- ShowCELLxGENEDatasets()
StatDBAttribute(
  df = all.cellxgene.datasets, filter = c("organism", "sex"),
  database = "CELLxGENE"
)
# HCA
all.hca.projects <- ShowHCAProjects()
StatDBAttribute(df = all.hca.projects, filter = c("organism", "sex"), database = "HCA")

## End(Not run)
```

Index