Package 'gclink'

May 8, 2026

Title Gene-Cluster Discovery, Annotation and Visualization

Version 1.1

Description Performs end-to-end analysis of gene clusters—such as photosynthesis, carbon/nitrogen/sulfur cycling, carotenoid, antibiotic, or viral marker genes (e.g., capsid, polymerase, integrase)—from genomes and metagenomes. It parses Basic Local Alignment Search Tool (BLAST) results in tab-delimited format produced by tools like NCBI BLAST+ and Diamond BLASTp, filters Open Reading Frames (ORFs) by length, detects contiguous clusters of reference genes, optionally extracts genomic coordinates, merges functional annotations, and generates publication-ready arrow plots. The package works seamlessly with or without the coding sequences input and skips plotting when no functional groups are found. For more details see Li et al. (2023) <doi:10.1038/s41467-023-42193-7>.

URL https://github.com/LiuyangLee/gclink

BugReports https://github.com/LiuyangLee/gclink/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr (>= 1.1.4), gggenes (>= 0.5.1), ggplot2 (>= 3.5.2), utils

Depends R (>= 3.5)

LazyData false

Config/check/use_internal_tz TRUE

NeedsCompilation no

Author Liuyang Li [aut, cre] (ORCID: <https://orcid.org/0000-0001-6004-9437>)

Maintainer Liuyang Li <cyanobacteria@yeah.net>

Repository CRAN

Date/Publication 2025-09-02 21:10:08 UTC

Contents

`blastp_df` *BLASTP Results for test Proteins*

Description

A dataset containing BLASTP alignment results for proteins from test genomes of bacteria, including alignment metrics.

Usage

```
blastp_df
```

Format

A data frame with multiple rows and 12 variables:

qaccver Character. Protein query identifier.

saccver Character. Subject protein identifier from reference databases.

pident Numeric. Percentage identity between query and subject sequences.

length Numeric. Length of the aligned sequence region.

mismatch Numeric. Number of mismatches in the alignment.

gapopen Numeric. Number of gap openings in the alignment.

qstart Numeric. Start position in query sequence.

qend Numeric. End position in query sequence.

sstart Numeric. Start position in subject sequence.

send Numeric. End position in subject sequence.

evalue Numeric. Expect value for the alignment.

bitscore Numeric. Bit score for the alignment.

---

eggnog_df *EggNOG Functional Annotation Results*

Description

A dataset containing EggNOG functional annotation results, including taxonomic classification, functional categories, and domain information.

Usage

```
eggnog_df
```

Format

A data frame with multiple rows and 21 variables:

#query Character. Protein query identifier (e.g., "Houyiibacteriaceae–LLY-WYZ-15_3—k141_356661_13").

seed_ortholog Character. Subject protein identifier from EggNOG database (e.g., "1168065.DOK_14784").

evalue Numeric. Expect value for the annotation.

score Numeric. Annotation bitscore.

eggNOG_OGs Character.

max_annot_lvl Character.

COG_category Character.

Description Character. Functional description (e.g., "Bacterial regulatory proteins, tetR family").

Preferred_name Character. Gene name if available (e.g., "cobB").

GOs Character.

EC Character. Enzyme Commission number if applicable (e.g., "2.7.7.49").

KEGG_ko Character. KEGG Orthology identifier if available (e.g., "ko:K00986").

KEGG_Pathway Character.

KEGG_Module Character.

KEGG_Reaction Character.

KEGG_rclass Character.

BRITE Character.

KEGG_TC Character.

CAZy Character.

BiGG_Reaction Character.

PFAMs Character.

gclink *Gene-Cluster Discovery, Annotation and Visualization*

Description

`gclink()` performs **all steps** from raw blastp output to a publication-ready gene-cluster plot **in one call**. It dynamically adapts its workflow:

- If `in_seq_data` is provided: Extracts coordinates/sequences, merges data, and generates plots.
- If `in_seq_data` is `NULL`: Skips sequence-dependent steps, returning only the cluster table.
- If `in_GC_group` is `NULL`: Omits functional-group merging and plotting.
- Supports custom gene lists (e.g., photosynthesis, viral genes) via `in_gene_list` for universal cluster detection in bacteria, archaea, or phages.

Gene clusters are identified via a density threshold (`AllGeneNum` and `MinConSeq`) due to incomplete gene annotation coverage.

Usage

```
gclink(
  in_blastp_df = blastp_df,
  in_seq_data = seq_data,
  in_gene_list = photosynthesis_gene_list,
  in_GC_group = PGC_group,
  AllGeneNum = 50,
  MinConSeq = 25,
  apply_length_filter = FALSE,
  down_IQR = 10,
  up_IQR = 10,
  apply_evalue_filter = FALSE,
  min_evalue = 1,
  apply_score_filter = FALSE,
  min_score = 10,
  orf_before_first = 0,
  orf_after_last = 0,
  orf_range = "All",
 levels_gene_group = c("bch", "puh", "puf", "crt", "acsF", "assembly", "regulator",
    "hypothetical ORF"),
 color_theme = c("#3BAA51", "#6495ED", "#DD2421", "#EF9320", "#F8EB00", "#FF0683",
    "#956548", "grey"),
  genome_subset = NULL
)
```

Arguments

in_blastp_df	A data.frame from blast/blastp-style output with columns: • qaccver: Genome + contig name (separated by "---"), e.g., "Kuafubacteriaceae--GCA_01670353 – Genome: "Kuafubacteriaceae--GCA_016703535.1" ("--" separator), – Contig: "JADJBV010000001.1", – ORF: "JADJBV010000001.1_150" ("_" separator), – Position: "150". • saccver: Gene name + metadata (separated by "_"), e.g., "bchC_Methyloversatilis_sp_RAC08_B – Gene: "bchC", – Metadata(Optional): "Methyloversatilis_sp_RAC08_BSY238_2447_METR". eggNOG (evolutionary gene genealogy Nonsupervised Orthologous Groups) format results are supported by renaming annotation columns (e.g., "GOs") to saccver. Default: blastp_df.

in_seq_data	NULL or a data.frame with:
	SeqName ORF identifier (Prodigal format: >ORF_id # start # end # strand # ...).
	Sequence ORF sequence.
	Example: "Kuafubacteriaceae--GCA_016703535.1---JADJBV010000001.1_1 # 74 # 1018 # 1 # ..." Can be imported from **Prodigal** FASTA using:
	`seq_data <- Biostrings::readBStringSet("Prodigal.fasta",format="fasta", nrec=-1L, ski`
	`  data.frame(Sequence = .) %>%`
	`  tibble::rownames_to_column("SeqName")`
	NULL skips coordinate extraction and plotting. Default: seq_data.

in_gene_list	Character vector of reference genes for cluster detection. Default: photosynthesis_gene_list.
in_GC_group	NULL or a data.frame mapping genes to functional groups (columns: gene, gene_group). NULL skips plotting. Default: PGC_group.
AllGeneNum	Integer; max ORFs per cluster. Default: 50.
MinConSeq	Integer; min consecutive reference genes per cluster. Default: 25.
apply_length_filter	Logical; whether to apply length filtering based on IQR. If FALSE, skips down_IQR and up_IQR filtering. Default: FALSE.
down_IQR	Numeric; lower-bound scale factor for IQR length filtering (see length_filter). Default: 10.
up_IQR	Numeric; upper-bound scale factor for IQR length filtering (see length_filter). Default: 10.
apply_evalue_filter	Logical; whether to filter BLAST hits by e-value cutoff. Requires column evalue in input data. Default: FALSE.
min_evalue	Numeric; maximum e-value threshold for retaining BLAST hits. Hits with e-value > cutoff will be removed. Typical values: 1e-3 (loose) to 1e-10 (strict). Default: 1.

`apply_score_filter`
: Logical; whether to filter BLAST hits by bitscore cutoff. Requires column `bitscore` in input data. Default: `FALSE`.

`min_score`
: Numeric; minimum bitscore threshold for retaining BLAST hits. Hits with bitscore < cutoff will be removed. For short proteins (~100 aa), 30-40 is typical; for long proteins (>300 aa), 50-100 is recommended. Default: `10`.

`orf_before_first`
: Integer; number of hypothetical ORFs to insert **before the first gene** of each detected cluster. Useful when the upstream annotation is incomplete or low-confidence; insertion is bounded by the first ORF of the contig. Default: `0`.

`orf_after_last`
: Integer; number of hypothetical ORFs to append **after the last gene** of each detected cluster. Helpful when the downstream annotation is incomplete or low-confidence; insertion is bounded by the last ORF of the contig. Default: `0`.

`orf_range`
: Character. ORF inclusion mode:
  - `"All"`: Include all ORFs + annotations (default)
  - `"OnlyAnnotated"`: Keep only annotated ORFs
  - `"IgnoreAnnotated"`: Include all ORFs without annotation merging

`levels_gene_group`
: Character vector; factor levels for gene-group legends (must include "hypothetical ORF" in case some genes remain unclassified). Ignored if `in_GC_group` is `NULL`. Default: `c('bch','puh','puf','crt','acsF','assembly','regulator','hypothetical ORF')`.

`color_theme`
: Character vector; colours for `gc_plot`, matched to the length and order of `levels_gene_group`. Ignored if plotting is skipped. Default: `c('#3BAA51','#6495ED','#DD2421','#EF9320','#F8EB00','`

`genome_subset`
: Character vector or `NULL`; genomes to retain. If `NULL`, all genomes are retained. Default: `NULL`.

Value

A named list with:

`GC_meta` Annotated cluster table (`data.frame`).

`GC_seq` FASTA sequences (if `in_seq_data` provided).

`GC_plot` ggplot object (if `in_seq_data` and `in_GC_group` provided).

Examples

```
# Case 1: Using blastp result with Full pipeline (Find Cluster + Extract FASTA + Plot Cluster)
 data(blastp_df)
 data(seq_data)
 data(photosynthesis_gene_list)
 data(PGC_group)
 gc_list <- gclink(in_blastp_df = blastp_df,
                   in_seq_data = seq_data,
                   in_gene_list = photosynthesis_gene_list,
                   in_GC_group  = PGC_group,
                   AllGeneNum = 50,
```

```
                    MinConSeq  = 25,
                    apply_length_filter = TRUE,
                    down_IQR   = 10,
                    up_IQR     = 10,
                    orf_before_first = 0,
                    orf_after_last = 0,
              levels_gene_group = c('bch','puh','puf','crt','acsF','assembly','regulator',
                                    'hypothetical ORF'),
                    color_theme = c('#3BAA51','#6495ED','#DD2421','#EF9320','#F8EB00',
                                    '#FF0683','#956548','grey'),
                    genome_subset = NULL)
 gc_meta = gc_list[["GC_meta"]]
 gc_seq = gc_list[["GC_seq"]]
 gc_plot = gc_list[["GC_plot"]]
 head(gc_meta)
 head(gc_seq)
 print(gc_plot)

# Case 2: Using eggNOG result with Full pipeline (Find Cluster + Extract FASTA + Plot Cluster)
 data(eggnog_df)
 data(seq_data)
 data(KO_group)
 KOs = c("K02291","K09844","K20611","K13789",
         "K09846","K08926","K08927","K08928",
         "K08929","K13991","K04035","K04039",
         "K11337","K03404","K11336","K04040",
         "K03403","K03405","K04037","K03428",
         "K04038","K06049","K10960","K11333",
         "K11334","K11335","K08226","K08226",
         "K09773")
 rename_KOs = paste0("ko:", KOs)
 eggnog_df$qaccver = eggnog_df$`#query`
 eggnog_df$saccver = eggnog_df$KEGG_ko
 eggnog_df$evalue = eggnog_df$evalue
 eggnog_df$bitscore = eggnog_df$score
 eggnog_df$gene = eggnog_df$KEGG_ko
 gc_list_2 = gclink(in_blastp_df = eggnog_df,
                    in_seq_data = seq_data,
                    in_gene_list = rename_KOs,
                    in_GC_group  = KO_group,
                    AllGeneNum = 50,
                    MinConSeq  = 25,
                    apply_evalue_filter = FALSE,
                    min_evalue = 1,
                    apply_score_filter = TRUE,
                    min_score = 10,
                    orf_before_first = 1,
                    orf_after_last = 1,
                    levels_gene_group = c('bch','puh','puf','crt',
                                          'acsF','assembly','hypothetical ORF'),
                    color_theme = c('#3BAA51','#6495ED','#DD2421','#EF9320',
                                    '#F8EB00','#FF0683','grey'))
 gc_meta_2 = gc_list_2[["GC_meta"]]
```

```
gc_seq_2 = gc_list_2[["GC_seq"]]
gc_plot_2 = gc_list_2[["GC_plot"]]
head(gc_meta_2)
head(gc_seq_2)
print(gc_plot_2)

# SOLUTION FOR "Viewport has zero dimension(s)" ERROR in print(gc_plot)
# ------------------------------------------------
# When you see this error, try these 3 solutions:

# 1. RESIZE RSTUDIO PLOT PANEL (Quickest fix)
# Simply click and drag the right border of the plot panel wider

# 2. LAUNCH IN NEW WINDOW
# dev.new()
# print(gc_plot)

# 3. SAVE DIRECTLY TO FILE
# ggplot2::ggsave('gc_plot.pdf', gc_plot, width=20,height=3)
```

---

gc_add *Complete Gene Clusters by Adding Missing ORFs*

Description

Expands gene cluster tables to include **all ORFs** (annotated and hypothetical) within contigs, normalizing cluster representations for downstream analysis and plotting. Ensures consistent ORF spacing/length across clusters by inserting missing rows.

Usage

```
gc_add(
  Data = sbgc,
  Annotation = bin_genes,
  orf_before_first = 0,
  orf_after_last = 0,
  orf_range = "All"
)
```

Arguments

Data	A `data.frame` of annotated ORFs with required columns: • `qaccver`: ORF identifier (genome—contig_orf_position format). • `genome`, `contig`: Genome and contig names. • `gene`: Gene symbol (NA for hypothetical ORFs). • `orf_position`: Absolute ORF position on the contig. • `gene_cluster`: Cluster identifier.

`Annotation` A `data.frame` of full ORF annotations (e.g., from `orf_extract`). Must include `qaccver` and `orf_position`.

`orf_before_first` Integer. Hypothetical ORFs to insert **before** the first annotated ORF in each cluster (bounded by contig start). Default: `0`.

`orf_after_last` Integer. Hypothetical ORFs to append **after** the last annotated ORF (bounded by contig end). Default: `0`.

`orf_range` Character. Controls ORF inclusion and annotation merging:

- `"All"`: Include every ORF in the contig range and merge all annotations (default).
- `"OnlyAnnotated"`: Keep only ORFs present in `Annotation` and merge their annotations.
- `"IgnoreAnnotated"`: Include all ORFs but skip merging with `Annotation`.

Details

- **Hypothetical ORFs** are inserted as rows with `gene = NA`.
- Output is always sorted by `gene_cluster` and `orf_position`.
- Progress messages are printed to console with timestamps.
- Contig bounds are respected—insertions never exceed actual ORF positions in `Annotation`.

Value

A `data.frame` with one row per ORF (real/hypothetical), sorted by `gene_cluster` and `orf_position`. Added columns:

`GC_orf_position` Relative position within cluster (1-indexed).

`GC_present_length` Count of annotated ORFs in the cluster.

`GC_absent_length` Count of inserted hypothetical ORFs.

`GC_length` Total ORFs (`GC_present_length` + `GC_absent_length`).

---

gc_cal *Identify and Extract Gene Clusters from Scaled BLAST Data*

Description

This function screens contigs for regions that contain a pre-defined set of "reference" genes (e.g., photosynthetic genes, viral genes) arranged in a continuous block. Contigs are first coarsely filtered by the minimum number of reference genes they carry, then finely scanned for clusters that satisfy user- defined density and contiguity criteria. Each detected cluster is returned with a unique `gene_cluster` identifier.

Usage

```
gc_cal(
  Data = bin_genes,
  in_gene_list = photosynthesis_gene_list,
  AllGeneNum = 30,
  MinConSeq = 15
)
```

Arguments

`Data`	A data frame produced by `orf_extract` (i.e., a scaled BLAST table). Must include the columns `genome_contig`, `gene`, and `orf_position`.
`in_gene_list`	A character vector of "reference" gene symbols (e.g., `photosynthesis_gene_list`) that are expected to appear in the target cluster(s).
`AllGeneNum`	Integer. Maximum total ORF count (annotated plus hypothetical) that the algorithm is allowed to span when defining a cluster (default: 30).
`MinConSeq`	Integer. Minimum number of **reference genes** that must be present **and consecutive** within the candidate cluster (default: 15). Must satisfy `1 <= MinConSeq <= AllGeneNum`.

Details

1. **Coarse filter**: Contigs with fewer than `MinConSeq` reference genes are discarded.
2. **Fine scan**: For each remaining contig, the algorithm slides a window that can encompass up to `AllGeneNum` consecutive ORFs and retains windows that contain at least `MinConSeq` reference genes in uninterrupted order.
3. **Cluster labelling**: Each valid cluster receives a unique ID (`genome_contig---1`, `genome_contig---2`, ...).

Value

A data frame identical in structure to `Data` but filtered to contain only those rows that belong to valid clusters. An extra column `gene_cluster` (format: `genome_contig---N`) is added to uniquely label every cluster.

---

gc_cluster *Identify Breakpoints of Gene Clusters within a Contig*

Description

Internal helper used by `gc_cal`. Given the ordered positions of *reference* genes on a contig, this function returns the genomic coordinates that mark the boundary of each candidate cluster. A boundary is declared whenever the gap between two successive reference genes exceeds the maximum spacing allowed by the cluster definition (`AllGeneNum - MinConSeq`).

Usage

```
gc_cluster(Data = orf_position.tmp, AllGeneNum = 30, MinConSeq = 15)
```

Arguments

`Data`	A numeric vector (ascending order) of ORF positions that carry one of the reference genes of interest. Usually the vector `orf_position.tmp` created inside `gc_cal`.
`AllGeneNum`	Integer. Maximum genomic span (in ORF count) that the algorithm is allowed to cover when defining a single cluster. Passed unchanged from `gc_cal`.
`MinConSeq`	Integer. Minimum number of consecutive reference genes required to form a cluster. Passed unchanged from `gc_cal`.

Details

- If the gap between two consecutive reference genes is larger than `AllGeneNum - MinConSeq`, the left-hand gene is recorded as the **last member** of the preceding cluster.
- The final reference gene is always appended to the vector as the last boundary, ensuring the rightmost cluster is not lost.

Value

A numeric vector containing every position that marks the **end** of a putative gene cluster. These values are subsequently used as breakpoints to slice the contig into candidate regions in the downstream functions `gc_position()` and `gc_range()`.

`gc_plot` *Plot Scaled Gene Clusters with Arrows*

Description

Generates a compact, publication-ready arrow plot of gene clusters previously processed with `gc_scale`. Each cluster is displayed on its own horizontal track, with gene directionality, labels, and user-defined colour schemes.

Usage

```
gc_plot(
  data = GC_meta,
 color_theme = c("#3BAA51", "#6495ED", "#DD2421", "#EF9320", "#F8EB00", "#FF0683",
    "#956548", "grey")
)
```

Arguments

data	A data frame produced by gc_scale, must include the columns Pstart, Pend, Pgenome, gene_group, Pdirection, and gene_label.
color_theme	Character vector of colours, **in the same order** as the factor levels of gene_group. Defaults to an 8-colour palette; supply fewer or more colours as needed.

Value

A ggplot object that can be further customised or printed. The plot uses gggenes::geom_gene_arrow() and gggenes::geom_gene_label() with the following elements:

- One facet per Pgenome (cluster).
- Genes coloured by gene_group.
- Arrow direction encoded by Pdirection.
- Optional gene labels inside arrows.

---

gc_position *Extract ORF Positions for One Specific Gene Cluster*

Description

Internal helper used by gc_cal. Given the ordered positions of **all** reference genes on a contig (Data) and the vector of cluster breakpoints produced by gc_cluster (Cluster), this function slices Data to return the positions that belong to the EachCluster-th cluster.

Usage

```
gc_position(
  Data = orf_position.tmp,
  Cluster = cluster.tmp,
  EachCluster = eachcluster
)
```

Arguments

Data	A numeric vector (ascending order) of ORF positions that carry one of the reference genes of interest. Usually the vector orf_position.tmp created inside gc_cal.
Cluster	Numeric vector returned by gc_cluster; each element marks the **last position** of a candidate cluster.
EachCluster	Integer index specifying which cluster to extract (1 = first cluster, 2 = second cluster, ...).

Value

A numeric vector containing the ORF positions that constitute the requested gene cluster.

gc_range *Determine ORF Range for a Candidate Gene Cluster*

Description

Internal helper used by gc_cal. After gc_position has isolated the ORF positions belonging to a single cluster, this function **validates** and **trims** that range so that the final span (distance between the first and last retained ORF) does not exceed AllGeneNum. The goal is to retain the **largest contiguous block** that still satisfies the user-defined size limit.

Usage

```
gc_range(Norf_position = Norf_position, AllGeneNum = 20, MinConSeq = 10)
```

Arguments

Norf_position	Numeric vector of ORF positions (ascending) that belong to the current candidate cluster (output from gc_position).
AllGeneNum	Integer. Maximum allowed genomic span (in ORF count) for the final cluster.
MinConSeq	Integer. Minimum number of consecutive reference genes required for the cluster.

Details

- For every reference gene in Norf_position, the function evaluates whether a window of at least MinConSeq consecutive reference genes centred on that gene can fit within AllGeneNum consecutive ORFs.
- Genes that pass the test are collected in retain.site.
- The **minimal** and **maximal** positions in retain.site are then used to slice the full ORF range, guaranteeing that the final cluster length <= AllGeneNum.

Value

A numeric vector containing the **final ORF positions** that define the validated gene cluster. If no valid block can be produced, the vector will be empty.

gc_scale *Scale Gene-Cluster Coordinates for Visualization*

Description

Prepares a gene-cluster annotation table for downstream plotting by converting absolute genomic coordinates into relative positions, ensuring that every cluster starts at 0 and is oriented consistently. Hypothetical ORFs (originally labeled with NA) and missing labels are replaced with placeholders, and factor levels are set as requested.

Usage

```
gc_scale(GC_meta = GC_meta, levels_gene_group = levels_gene_group)
```

Arguments

GC_meta	A data frame containing gene-cluster information. Must include the columns gene_cluster, gene_group, gene_label, start, end, and direction (numeric: 1 for forward, -1 for reverse).
levels_gene_group	Character vector specifying the desired factor levels for gene_group. Group names should appear in the order required for plotting legends.

Details

- Absolute start/end values are **not** modified; scaled values are stored in new columns (Pstart, Pend).
- Pgenome can be swapped for any unique identifier (e.g., Genome) downstream if each genome contains only one cluster.

Value

The input data frame with the following **new or overwritten** columns:

gene_label Empty string ("") if originally NA.

gene_group Set to "hypothetical ORF" if originally NA, then coerced to a factor using levels_gene_group.

Pgenome Factor version of gene_cluster; levels follow the order of appearance in the data.

Pstart, Pend Relative start and end coordinates (numeric) within each cluster, scaled so that the left-most gene starts at 0.

Pdirection Logical vector: TRUE for forward, FALSE for reverse.

KO_group *KEGG Orthology (KO) Group Classification*

Description

A dataset mapping KEGG Orthology (KO) identifiers to their functional groups and gene symbols.

Usage

```
KO_group
```

Format

A data frame with multiple rows and 3 variables:

gene Character. KEGG Orthology identifier (e.g., "K04035").

gene_group Character. Functional group (e.g., "acsF").

gene_label Character. Gene symbol (e.g., "acsF").

length_filter *Remove Length Outliers from BLAST Results*

Description

Filters BLAST hits by removing ORFs whose gene (protein) length is an outlier within the corresponding gene group, as defined by the inter-quartile range (IQR). Hits whose length falls outside the interval `[Q1 - down_IQR * IQR, Q3 + up_IQR * IQR]` are discarded.

Usage

```
length_filter(Data = bin_genes, down_IQR = 1.5, up_IQR = 1.5)
```

Arguments

`Data`	A data frame containing BLAST results. Must include the columns `gene` (gene symbol) and `length` (ORF length in amino acids).
`down_IQR`	Numeric multiplier applied to the IQR for the **lower** bound (default: 1.5).
`up_IQR`	Numeric multiplier applied to the IQR for the **upper** bound (default: 1.5).

Details

- Filtering is performed **within each gene group**; outliers are determined independently for every gene symbol.
- Progress messages report the number of rows before and after filtering.
- Missing values in `length` are ignored when computing quantiles.

Value

The input data frame with outlier rows removed. The returned object is **ungrouped** regardless of the input grouping.

`orf_extract` *Extract ORF and Genome Information from BLAST or BLASTP Results*

Description

This function parses BLASTP result tables to extract structured genome, contig, ORF, and gene information from the query and subject identifiers. It is designed for downstream analyses requiring explicit separation of genome, contig, and ORF identifiers from concatenated BLAST headers.

Usage

```
orf_extract(bin_genes = blastp_df)
```

Arguments

`bin_genes`	A data frame containing BLASTP results with at least 2 standard columns: `qaccver`, `saccver`. the column of qaccver should include both of the genome name and predicted contig name, which is concatenated by a separator "—". for example, for the qaccver "p__Myxococcota–c__Kuafubacteria–o__Kuafubacteriales–f__Kuafubacteriaceae–GCA_016703535.1—JADJBV010000001.1_150", the genome name is "p__Myxococcota–c__Kuafubacteria–o__Kuafubacteriales–f__Kuafubacteriaceae–GCA_016703535.1", the contig name is "JADJBV010000001.1", the orf name is "JADJBV010000001.1_150", and the orf_position is "150". the column of saccver must include the gene name and may include the gene information, which are concatenated by a separator "_". for example, for the saccver "bchC_Methyloversatilis_sp_RAC the gene name is "bchC", the gene information is Methyloversatilis_sp_RAC08_BSY238_2447_METR that can help you understand the source of gene.

Value

The original data frame with six additional columns:

`genome` Genome identifier extracted from `qaccver`.

`contig` Contig identifier extracted from `qaccver`.

`orf` Full ORF identifier extracted from `qaccver`.

`genome_contig` Concatenated genome and contig IDs (genome—contig).

`gene` Gene symbol extracted from `saccver`.

`orf_position` Numeric ORF position extracted from the ORF identifier.

orf_locate *Parse ORF Coordinates from Prodigal FASTA Headers*

Description

Extracts ORF identifiers, start/end positions and strand orientation directly from the FASTA headers produced by Prodigal. The resulting table is ready for downstream gene-cluster analyses.

Usage

```
orf_locate(in_seq_data = seq_data)
```

Arguments

in_seq_data	A data frame with two columns:
	`SeqName` ORF identifier (Prodigal format: `>ORF_id # start # end # strand # ...`).
	`Sequence` ORF sequence.
	Example: `"Kuafubacteriaceae--GCA_016703535.1---JADJBV010000001.1_1 # 74 # 1018 # 1 # ..."` Can be imported from **Prodigal** FASTA using:
	`seq_data <- Biostrings::readBStringSet("Prodigal.fasta",format="fasta", nrec=-1L, ski`
	`data.frame(Sequence = .) %>%`
	`tibble::rownames_to_column("SeqName")`

Value

A data frame

PGC_group *Photosynthesis Gene Classification Groups*

Description

A dataset mapping photosynthesis-related genes to their functional groups and subunits.

Usage

```
PGC_group
```

Format

A data frame with multiple rows and 3 variables:

gene Character. Gene identifier (e.g., "bchB").

gene_group Character. Functional group (e.g., "bch").

gene_label Character. Subunit designation (e.g., "B").

photosynthesis_gene_list *Photosynthesis Gene List*

Description

A comprehensive list of genes involved in photosynthetic processes.

Usage

```
photosynthesis_gene_list
```

Format

A character vector containing gene identifiers:

Each element represents a photosynthesis-related gene symbol (e.g., "acsF", "bchB").

seq_data *Genomic Sequence Data with Annotations*

Description

A dataset containing DNA sequences from test bacteria with detailed annotation metadata. The first column combines multiple annotation elements separated by semicolons.

Usage

```
seq_data
```

Format

A data frame with multiple rows and 2 variables:

SeqName Character. Combined annotation fields separated by semicolons, containing:

- `ID`: Sequence identifier (e.g., "1_7")
- `partial`: Completion status ("00" for complete, "01" for partial)
- `start_type`: Translation initiation codon (e.g., "GTG", "ATG")
- `rbs_motif`: Ribosome binding site motif (e.g., "GGAG/GAGG")
- `rbs_spacer`: RBS spacer length (e.g., "5-10bp")
- `gc_cont`: GC content (e.g., "0.673")

Sequence Character. DNA sequence (when available) in FASTA format

Index