Package: ggchord (via r-universe)

November 3, 2026

Title Layered Chord, Circular and Linear Sequence Visualization

Version 0.16.6

Description A 'ggplot2'-based package for multi-sequence alignment chord diagrams, linear collinearity plots and single circular sequence maps. A layered grammar combines sequence backbones, interval links, genes, general biological features, labels, quantitative tracks, genomic axes and restriction sites. Feature placement is expressed with reusable Position objects, while role-specific scales and themes keep data mapping separate from geometry and appearance.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports ggplot2 (>= 4.0.0), grDevices, grid, rlang, scales

VignetteBuilder knitr

Suggests knitr, rmarkdown, qpdf, dplyr, purrr, tidyr, svglite, testthat (>= 3.0.0), readxl, digest, GenomicRanges, InteractionSet, IRanges, S4Vectors, jsonlite, Biostrings, colorspace

Encoding UTF-8

LazyData true

URL https://github.com/DangJem/ggchord, https://dangjem.github.io/ggchord/

BugReports https://github.com/DangJem/ggchord/issues

NeedsCompilation no

Config/roxygen2/version 8.0.0

Roxygen list(markdown = TRUE)

Config/Needs/website pkgdown, jsonlite, pkgload

Repository https://dangjem.r-universe.dev

Date/Publication 2026-11-03 04:13:03 UTC

RemoteUrl https://github.com/dangjem/ggchord

RemoteRef HEAD

RemoteSha f6e0840cca515a3cabac39ef2bb3dfea5eaf4d11

Contents

+.ggchord . . . 4
as.data.frame.ggchord_validation . . . 4
as_ggchord_feature . . . 5
as_ggchord_link . . . 6
as_ggchord_seq . . . 7
build_ggchord_cds . . . 8
bundle_ggchord_ribbons . . . 9
clean_ggchord_data . . . 11
coord_chord . . . 13
coord_circular . . . 14
coord_collinear . . . 15
deduplicate_ggchord_ribbons . . . 17
diagnose_ggchord . . . 18
export_ggchord_layout . . . 20
filter_ggchord_ribbons . . . 21
filter_restriction_sites . . . 23
find_common_features . . . 24
find_ggchord_loci . . . 25
find_primer_bindings . . . 26
find_restriction_sites . . . 28
focus_ggchord_data . . . 29
focus_ggchord_track . . . 30
gene_data_example . . . 31
geom_feature . . . 32
geom_feature_label . . . 34
geom_feature_label_repel . . . 35
geom_feature_plasmid . . . 37
geom_gene . . . 39
geom_gene_label . . . 40
geom_gene_label_repel . . . 42
geom_link_line . . . 45
geom_link_ribbon . . . 46
geom_primer . . . 48
geom_primer_label_repel . . . 49
geom_restriction_site . . . 50
geom_seq . . . 52
geom_seq_break . . . 53
geom_seq_center_label . . . 55
geom_seq_label . . . 56
geom_seq_region . . . 57
geom_seq_scale . . . 59
geom_track_tile . . . 60

get_chord_layout . . . 63
ggchord . . . 64
ggchord_api_contract . . . 65
ggchord_data_contract . . . 67
ggchord_layout_contract . . . 69
ggplot2-helpers . . . 71
grid-helpers . . . 79
guide_ggchord_legend . . . 80
link_ggchord_clusters . . . 82
map_ggchord_position . . . 84
merge_ggchord_ribbons . . . 85
optimize_ggchord_layout . . . 87
plasmid_example_pUC19 . . . 88
position_feature_stack . . . 88
position_plasmid . . . 89
position_strand . . . 90
position_track . . . 90
prepare_ggchord_data . . . 91
primer_catalog . . . 93
print.ggchord_clean . . . 94
project_ggchord_features . . . 94
project_ggchord_links . . . 96
read_bed . . . 97
read_blast . . . 98
read_fasta_lengths . . . 99
read_genbank . . . 100
read_gff3 . . . 102
read_paf . . . 103
restriction_site_example . . . 104
ribbon_data_example . . . 105
scale_feature_fill_plasmid . . . 105
scale_feature_shape_manual . . . 106
scale_gene_fill_manual . . . 106
scale_link . . . 107
scale_ribbon_alpha_continuous . . . 108
scale_ribbon_fill_stepsn . . . 109
scale_seq_colour_manual . . . 110
scale_seq_position_continuous . . . 110
scale_seq_ring_manual . . . 111
scale_track_fill_gradientn . . . 112
scale_track_value_continuous . . . 113
segment_ggchord_data . . . 114
seq_data_example . . . 115
single_gene_example . . . 116
single_genome_example . . . 116
stat_ribbon_bundle . . . 116
stat_ribbon_density . . . 117
theme_ggchord . . . 118

validate_ggchord_data 132
validate_ggchord_hierarchy 134
view_ggchord 135

Index **137**

`+.ggchord` *Combine a ggchord plot with ggplot2 objects*

Description

Uses ggplot2's standard composition semantics for layers, scales, themes, coordinates and annotations.

Usage

```
## S3 method for class 'ggchord'
e1 + e2
```

Arguments

`e1`	A ggchord object
`e2`	A ggplot2 component.

Value

A ggchord object

`as.data.frame.ggchord_validation`
Coerce a validation result to a flat data.frame

Description

Combines the `errors` and `warnings` tables into a single data.frame and adds a `severity` column, which is convenient for filtering, exporting or printing the full report programmatically.

Usage

```
## S3 method for class 'ggchord_validation'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A "`ggchord_validation`" object.
`row.names`	Ignored.
`optional`	Ignored.
`...`	Ignored.

Value

A data.frame with columns `table`, `category`, `row`, `column`, `message` and `severity`.

See Also

ggchord_data_contract

`as_ggchord_feature` *Convert intervals to generic ggchord features*

Description

`GRanges` coordinates are already one-based inclusive and are not shifted. Strand `*` is retained as `source_strand`, while the output uses `strand = "."` and `directionality = "none"`. Use the returned table in `geom_feature(data = ...)` to preserve nondirectional intervals; `geom_gene()` requires directional gene input. No unknown strand is invented as `+`.

Usage

```
as_ggchord_feature(x, mapping = NULL)
```

Arguments

`x`	A data frame or `GRanges` object.
`mapping`	Optional named character vector mapping output roles to source columns, as in `as_ggchord_seq()`.

Value

An ordinary feature data frame with `accver`, `start`, `end`, `strand`, `directionality`, retained metadata and conversion provenance. Conflicting metadata names receive a `metadata_` prefix; the attribute's `column_map` records the output name.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(gene_data_example)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
renamed <- genes |>
  rename(chrom = accver, left = start, right = end)
```

```
features <- as_ggchord_feature(renamed,
  mapping = c(accver = "chrom", start = "left", end = "right")
)
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_feature(data = features)
```

`as_ggchord_link` *Convert paired genomic intervals to ggchord links*

Description

For `GInteractions`, the first stored anchor becomes query and the second becomes subject. Their order is never sorted or swapped by this adapter. A source object that already canonicalized its anchors cannot recover their earlier biological host/virus roles; establish those roles before conversion. Negative-strand anchor endpoints run from end to start. Unknown-strand anchors retain increasing coordinates, their `*` strand and `orientation_known = FALSE`; increasing order is not direction evidence. Interval interactions do not imply a pairwise alignment: identity and alignment length are not inferred from anchor widths.

Usage

```
as_ggchord_link(x, mapping = NULL)
```

Arguments

`x`	A data frame or `GInteractions` object.
`mapping`	Optional named character vector mapping canonical link roles to source columns, as in `as_ggchord_seq()`.

Value

An ordinary data frame with `qaccver`, `saccver`, `qstart`, `qend`, `sstart`, `send` and source metadata. Paired objects also retain anchor indices, source row/name, anchor strands, anchor metadata prefixed with `query_`/`subject_`, and object-level metadata in the conversion attribute.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
as_ggchord_link(links)
```

as_ggchord_seq *Convert sequence information to ggchord input*

Description

Explicit adapters return ordinary data frames. Optional interval packages are loaded only when their objects are supplied; drawing never installs or runs external data preparation. Supported S4 inputs supplied directly to ggchord or its layers use the same adapters. Identifiers are kept verbatim.

Usage

```
as_ggchord_seq(x, mapping = NULL, lengths = NULL, ids = NULL)
```

Arguments

x	A data frame, GRanges, Seqinfo, GInteractions, or a Biostrings XStringSet/XString object.
mapping	Optional named character vector output_role = "input_column" for data-frame inputs. For example c(accver = "chrom", length = "size").
lengths	Optional named numeric vector of known sequence lengths. Fills unknown lengths and must agree with any known source lengths.
ids	Optional character vector of explicit identifiers for unnamed XStringSet/XString inputs only. Existing source names must agree.

Details

For a GRanges object, lengths come from its Seqinfo, never from the maximum observed feature endpoint. A GInteractions object uses the same information from its common regions. Only sequence names used by the supplied intervals are returned; a Seqinfo object returns all its entries. Unknown lengths require explicit lengths or corrected source information.

XStringSet and XString inputs use their stored sequence widths. Set names are identifiers; unnamed sequences require explicit ids. Explicit IDs must agree with existing nonempty names, so identifiers and accession versions are never silently rewritten. sequence_unit records "bp", "aa", or "character"; amino-acid widths are not nucleotide coordinates. These non-nucleotide tables remain available for data workflows, but ggchord() rejects them as genomic sequence input. Project amino-acid intervals with build_ggchord_cds() and project_ggchord_features() or project_ggchord_links() before using known nucleotide sequence lengths. Empty sequences are rejected like other zero lengths.

Value

An ordinary data frame with accver, length and retained metadata. Its ggchord_conversion attribute records input class, coordinate system, column renaming, object-level metadata, and upstream import/conversion provenance when present. No genomic coordinates or accession versions are guessed or rewritten.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
renamed <- seqs |>
  rename(chrom = accver, size = length)
as_ggchord_seq(renamed, mapping = c(accver = "chrom", length = "size"))
```

---

build_ggchord_cds *Build an explicit spliced CDS coordinate map*

Description

Builds a mapping from observed protein residues to genomic coding bases. CDS rows sharing `feature_id` form one coding model. Their genomic intervals must be non-overlapping and have one sequence and known strand. Segments are ordered in transcription direction, increasing on `+` and decreasing on `-`. An explicit `segment_order` overrides genomic sorting, allowing a declared circular-origin join. A `segments` list-column can declare `segment_order` for transcription order, including reverse-strand and origin joins; otherwise its pieces follow genomic strand order. Generic `segment_index` identifiers are preserved as source metadata and do not define transcription order. Nested phases may be supplied or derived from an explicit first phase or `codon_start - 1`; no phase is guessed from the genomic envelope. Trans-spliced models spanning different sequence IDs are rejected.

Usage

```
build_ggchord_cds(cds_data, genetic_code = NULL, partial = c("error", "trim"))
```

Arguments

`cds_data`	Data frame with `feature_id`, `accver`, ordered one-based inclusive `start`/`end`, `strand` (`+`/`-`) and GFF3 `phase` (0–2, numeric or literal text as returned by `read_gff3()`). `.` remains unknown; unknown initial phase requires explicit `codon_start` for nested CDS. If `type` is present, select CDS rows explicitly before building a map. Alternatively supply one row per CDS with a `segments` list-column and explicit initial `phase` or `codon_start` (1–3). Nested genomic pieces retain explicit `segment_order` when supplied and are never replaced by the parent's outer envelope.
`genetic_code`	Optional positive integer code ID, scalar or named by CDS `feature_id`. Must agree with a supplied `genetic_code` column. Missing IDs remain unknown; they do not prevent coordinate-only projection.
`partial`	`"error"` (default) or explicit `"trim"`.

Details

GFF3 phase counts bases from a segment's biological 5-prime end to the next codon boundary. Only the first phase is skipped; later phases validate the accumulated coding length, retaining bases of codons split across exons. `partial = "error"` rejects partial ends. Explicit `"trim"` permits a nonzero first phase, partial-end flags and incomplete terminal codons, and excludes bases without complete codons. Residue one then means the first complete observed codon, not an inferred position in an unobserved protein.

Optional `partial5`, `partial3`, `genetic_code`, and `protein_id` columns must agree within each model. A nonzero initial phase requires partial5; if absent it is recorded as partial5. A terminal remainder similarly records partial3 unless a conflicting explicit flag is supplied. Genetic-code IDs are retained as metadata: coordinate projection does not translate sequence, identify stop codons, or infer a genetic code. `codon_count` includes any source terminal stop codon. Frameshifts, ribosomal slippage and translation exceptions or non-join location operators cannot be represented by a simple triplet map and are rejected when declared. No full protein length or identity is inferred.

Value

A `ggchord_cds` list containing a `models` summary, transcription- ordered `segments` per feature, original `source_data` and provenance. Segment `cds_start`/`cds_end` identify complete-codon nucleotide positions; genomic coordinates and original row numbers remain available.

See Also

https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md

ggchord_data_contract

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
coding <- features |>
  filter(anno == "AmpR", type == "CDS") |>
  select(accver, start, end, strand) |>
  mutate(feature_id = paste(accver, "AmpR", sep = ":"), phase = 0)
cds <- build_ggchord_cds(coding, genetic_code = 11)
cds$features
```

`bundle_ggchord_ribbons` *Bundle dense alignment ribbons explicitly*

Description

Aggregates nearby alignment ribbons on a normalized query/subject grid. Bundling is always explicit: `geom_link_ribbon()` continues to draw one ribbon per input row unless the returned data are supplied by the user. Groups use relative endpoint direction, so reversing both ends can remain in the same group. The envelope follows the first row's absolute directions; length and identity are length-weighted source means (length is rounded), not statistics of a newly computed alignment. Conflicting extra metadata become typed missing values. Use `group_by` to keep source categories separate and retain the original input with the source-row report. Single-base endpoint rows stay independent because their endpoint order cannot define an absolute aggregate direction. Density can still count them without merging their geometry.

Usage

```
bundle_ggchord_ribbons(
  ribbon_data,
  seq_data,
  bins = 80L,
  min_bundle = 2L,
  weight = c("length", "count", "pident"),
  group_by = NULL
)
```

Arguments

`ribbon_data`	Alignment data in ggchord ribbon format.
`seq_data`	Sequence data containing `accver` and `length`.
`bins`	Positive integer number of normalized midpoint bins per sequence, default 80.
`min_bundle`	Minimum number of rows required for aggregation, default 2. Smaller groups remain as individual ribbons.
`weight`	Bundle weight: `"length"`, `"count"`, or `"pident"`. Identity weight is aligned length multiplied by identity proportion.
`group_by`	Optional character columns that must agree before ribbons can be bundled.

Value

A list with `data` and `report`. The data include `.bundle_n`, `.bundle_weight`, and `.bundle_density`; the report maps each output row back to its source rows.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
```

```
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
out <- bundle_ggchord_ribbons(links, seqs, bins = 30)
out$report
```

---

clean_ggchord_data *Clean ggchord input data with explicit, report-driven policies*

Description

Applies a conservative, predictable set of cleaning policies to the three ggchord tables and returns the cleaned copies plus a full report. The original data frames are never modified. Nothing is dropped silently: every change (including every dropped row) is recorded in `report` with the original row number, the reason, the original value(s) and the new value(s). Its historical sorting default differs deliberately from preparation's direction-preserving default. Use `reversed_interval = "keep"` for directional links and circular-origin features; sorting endpoints loses information that a later workflow cannot reconstruct.

Usage

```
clean_ggchord_data(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  unknown_id = c("drop", "error", "keep"),
  out_of_range = c("clip", "drop", "error", "keep"),
  reversed_interval = c("sort", "drop", "error", "keep"),
  invalid_pident = c("clip", "drop", "error", "keep"),
  empty_annotation = c("keep", "drop", "replace"),
  replacement_annotation = "unannotated"
)
```

Arguments

`seq_data`	data.frame/tibble, required. Must contain `accver` and `length`; used as the coordinate reference for clipping. Reference IDs must always be non-missing, non-empty and unique, regardless of `unknown_id`.
`ribbon_data`	data.frame/tibble or `NULL`, optional. Alignment results. Other input types are rejected rather than treated as absent.
`gene_data`	data.frame/tibble or `NULL`, optional. Gene annotation data. Other input types are rejected rather than treated as absent.
`unknown_id`	Character, default `"drop"`. Policy for rows whose sequence ID in `ribbon_data` or `gene_data` is missing, empty or unknown: `"drop"` removes them, `"error"` stops with a message, `"keep"` leaves them unchanged. A kept row without a reference sequence is reported, but its coordinates and annotations are not cleaned and it is not guaranteed to be drawable. IDs are never guessed or replaced. Other

structural checks, including finite numeric coordinates and valid gene strands, still apply.

`out_of_range` Character, default `"clip"`. Policy for coordinates outside `[1, sequence length]`: `"clip"` clamps them, `"drop"` removes the row, `"error"` stops, `"keep"` leaves them unchanged. Coordinates are one-based inclusive, so `start == end` is a valid single-base interval and is retained. With `"clip"`, intervals wholly outside their sequence are removed and reported; partially overlapping intervals retain their intersection, even a single base.

`reversed_interval`
Character, default `"sort"`. Policy for intervals with `start > end`: `"sort"` swaps the endpoints so the feature draws stably (the original direction is recorded in the report), `"drop"` removes the row, `"error"` stops, `"keep"` leaves them unchanged.

`invalid_pident` Character, default `"clip"`. Policy for `pident` outside `[0, 100]`: `"clip"` clamps them, `"drop"` removes the row, `"error"` stops, `"keep"` leaves them unchanged.

`empty_annotation`
Character, default `"keep"`. Policy for missing/empty anno in gene data: `"replace"` fills them with `replacement_annotation`, `"drop"` removes the row, `"keep"` leaves them unchanged.

`replacement_annotation`
Character, default `"unannotated"`. Replacement annotation used when `empty_annotation = "replace"`.

Value

A list with four components: `seq_data`, `ribbon_data`, `gene_data` (cleaned copies) and `report` (a data.frame with columns `table`, `row` (original row number), `column`, `reason`, `original_value`, `new_value` and `action`).

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
# Deliberate diagnostic error in a copy; source observations remain unchanged.
problem <- links |>
  slice_head(n = 2) |>
  mutate(qaccver = replace(qaccver, 2, "absent"))
cleaned <- clean_ggchord_data(seqs, problem, unknown_id = "drop")
cleaned$report
```

coord_chord *Chord diagram coordinate system*

Description

Controls global rotation, aspect ratio, coordinate limits, clipping and the strategy used to fit chord geometry and labels.

Usage

```
coord_chord(
  rotation = 45,
  ratio = 1,
  xlim = NULL,
  ylim = NULL,
  expand = FALSE,
  clip = "off",
  fit = c("labels", "geometry", "manual")
)
```

Arguments

rotation	Global counterclockwise layout rotation in degrees, default 45, measured from the positive x axis. This rotates the complete layout; it does not change the biological strand or sequence orientation.
ratio	Fixed y/x aspect ratio, default 1.
xlim, ylim	Optional user limits. Explicit limits take priority over automatically fitted limits.
expand	Logical. Expand coordinate limits, default FALSE. Automatic fitting already includes a small safety margin; set TRUE to request the additional ggplot2 coordinate expansion.
clip	Whether drawing is clipped to the panel, default "off".
fit	Fitting strategy: "labels" includes measured label boxes, "geometry" fits geometric elements only, and "manual" requires explicit xlim and ylim.

Value

A Coord object for ggplot2 + composition

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
ggchord(seq_data = seqs) +
  geom_seq() +
  coord_chord(rotation = 20)
```

`coord_circular` *Single-sequence circular coordinate system*

Description

Single-sequence circular coordinate system

Usage

```
coord_circular(
  rotation = 0,
  direction = c("clockwise", "counterclockwise"),
  gap = 0,
  ratio = 1,
  xlim = NULL,
  ylim = NULL,
  expand = FALSE,
  clip = "off",
  fit = c("labels", "geometry", "manual")
)
```

Arguments

`rotation`	Angle in degrees at which the genomic origin is placed.
`direction`	Direction of increasing genomic coordinates.
`gap`	Opening in degrees. Zero draws a closed circle.
`ratio`	Fixed y/x aspect ratio, default 1.
`xlim`, `ylim`	Optional user limits. Explicit limits take priority over automatically fitted limits.
`expand`	Logical. Expand coordinate limits, default FALSE. Automatic fitting already includes a small safety margin; set TRUE to request the additional ggplot2 coordinate expansion.
`clip`	Whether drawing is clipped to the panel, default `"off"`.
`fit`	Fitting strategy: `"labels"` includes measured label boxes, `"geometry"` fits geometric elements only, and `"manual"` requires explicit `xlim` and `ylim`.

Value

A Coord object for ggplot2 composition.

coord_collinear *Linear multi-sequence collinearity coordinates*

Description

Draws sequences as left-aligned horizontal backbones, ordered from top to bottom by `geom_seq(seq_order = ...)`. All sequences share one linear bp scale. `seq_orientation = -1` reverses a sequence together with its features, links and axis without modifying the input data.

Usage

```
coord_collinear(
  ratio = 1,
  xlim = NULL,
  ylim = NULL,
  expand = FALSE,
  clip = "off",
  fit = c("labels", "geometry", "manual"),
  align = c("left", "center", "right"),
  anchor = NULL,
  orientation = c("manual", "auto"),
  min_support = 0.1,
  segments = NULL,
  segment_gap = 100
)
```

Arguments

`ratio`	Fixed y/x aspect ratio, default 1.
`xlim`, `ylim`	Optional user limits. Explicit limits take priority over automatically fitted limits.
`expand`	Logical. Expand coordinate limits, default FALSE. Automatic fitting already includes a small safety margin; set TRUE to request the additional ggplot2 coordinate expansion.
`clip`	Whether drawing is clipped to the panel, default `"off"`.
`fit`	Fitting strategy: `"labels"` includes measured label boxes, `"geometry"` fits geometric elements only, and `"manual"` requires explicit `xlim` and `ylim`.
`align`	Display alignment: `"left"` (default), `"center"`, or `"right"`. Applied to the full sequence or the interval supplied by `anchor`.
`anchor`	Optional anchor positions as a numeric vector named by `accver`, or a data frame with `accver` and either `position` or `start`/`end`. Every displayed sequence must have exactly one anchor. Missing and duplicate anchors are errors; multiple-copy targets must be selected explicitly before constructing the coordinate system.
`orientation`	`"manual"` (default) uses sequence directions supplied by `geom_seq()`. `"auto"` derives display directions from interval-link evidence and records support and unresolved cases.

`min_support` Minimum directional imbalance in `[0, 1]` for an automatic change. It is the absolute difference between forward and reverse support divided by their sum, default 0.1. A tie is always unresolved.

`segments` Optional data frame with unique window `accver`, `row_id` and `orientation` (1 or -1). Every displayed accession must be declared. Each window has its own local coordinates and reversible source metadata.

`segment_gap` Positive display gap in bp scale units when `segments` is supplied, default 100. It never implies genomic continuity.

Details

In this coordinate system, `geom_seq(seq_gap = ...)` specifies the distance between consecutive backbones as a proportion of the longest backbone's width (default 0.3). Positive Position offsets point upward and negative offsets downward, independently of sequence orientation. Sequence labels appear to the left of their backbone. Automatic feature labels use local horizontal rows; fixed radial labels are vertical and tangent labels are horizontal. Label side `"outside"` means above and `"inside"` below.

Explicit sequence radius, curvature, ring mappings/scales and sequence-label radius cannot be converted and produce an error. Remove these arguments when switching from `coord_chord()`. Circular-only primer, restriction-site and centre-label layers require `coord_circular()`; use `geom_feature()` for general linear annotations. Cross-track links may cross intervening tracks. Explicit link control points use the exported linear layout x/y units; the longest backbone has width 2 (the longest complete display row when `segments` is supplied). The circular `external = FALSE` feature-label constraint cannot be converted; remove it to use linear label rows. Gene/feature repel layout modes share the linear row strategy rather than selecting circular label tracks.

`align` and `anchor` control a reversible display translation. With no anchor, the selected edge or centre of each complete sequence is placed at x = 0. With an anchor, the selected edge or centre of each anchor interval is placed at x = 0 instead. Left and right refer to the displayed interval, after orientation has been applied. Genomic input coordinates stay unchanged. Sequence order is never changed by alignment or automatic orientation.

`orientation = "auto"` evaluates interval links against earlier displayed sequences and chooses the direction supported by the larger aligned-interval weight. The first sequence fixes the reference direction. Ambiguous, insufficiently supported and unconnected sequences retain their initial direction, with evidence recorded in the exported layout diagnostics. This is a display heuristic, not a biological strand assignment or a guarantee of a global optimum. It cannot be combined with an explicit `geom_seq(seq_orientation = ...)`.

With `segments`, each accession remains an independent window but windows sharing `row_id` occupy one display row. Their table order is left to right; `align` applies to the whole row. `segment_gap` reserves display space in bp scale units, independently of omitted genomic length. Internal ends receive break marks. Use `segment_ggchord_data()` to crop/split input first. Local reversal comes from the table's `orientation`, so automatic orientation, explicit sequence orientation and `anchor` conflict with this mode. Ribbon adjacency follows display rows; same-row ribbons are omitted with a report, and explicit same-row pairs or point links are unsupported.

Value

A ggplot2 Coord object for composition with a ggchord plot.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
ggchord(seq_data = seqs, ribbon_data = links) +
  geom_seq() +
  geom_link_ribbon() +
  geom_seq_label() +
  coord_collinear(orientation = "auto")
```

---

deduplicate_ggchord_ribbons

Deduplicate alignment ribbons

Description

Removes fully duplicated, coordinate-near-duplicated or highly overlapping alignment blocks within each (query, subject) pair and keeps the best representative per `keep`.

Usage

```
deduplicate_ggchord_ribbons(
  ribbon_data,
  tolerance = 0,
  by = c("exact", "coordinates", "overlap"),
  keep = c("best_pident", "longest", "first"),
  min_reciprocal_overlap = 0.9
)
```

Arguments

`ribbon_data`	data.frame in ribbon_data format.
`tolerance`	Numeric, default 0. Maximum absolute difference (in bp) allowed on each of `qstart`/`qend`/`sstart`/`send` for two rows to count as coordinate duplicates (used with `by = "coordinates"`).
`by`	Character, default `"exact"`. `"exact"` removes rows with identical pair and coordinates; `"coordinates"` additionally treats rows within `tolerance` bp as duplicates; `"overlap"` treats blocks whose reciprocal overlap is at least `min_reciprocal_overlap` on both sequences as duplicates. Each interval must meet the threshold, not only the shorter one. Opposite relative endpoint directions are kept as distinct relationships by all methods. Single-base endpoints use supplied strand/direction evidence; otherwise their direction is unknown.

keep — Character, default "best_pident". Which representative to keep: "best_pident" (highest pident), "longest" (longest length) or "first" (first occurrence in the input).

min_reciprocal_overlap — Numeric (0-1), default 0.9. Reciprocal overlap threshold used with by = "overlap".

Value

A list with data (deduplicated data frame with source_rows attribute) and report (n_input/n_kept/n_removed plus a data.frame of removed rows with row, duplicate_of and reason).

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
out <- deduplicate_ggchord_ribbons(links, by = "overlap")
out$report
```

---

diagnose_ggchord *Inspect one target-size render without changing the plot*

Description

Builds on a temporary device of the requested kind and measures the actual single panel after gtable layout, then refines physical geometry within a bounded render-local context. The original plot and its layout cache are left unchanged. Standard saving continues to use ggplot2::ggsave().

Usage

```
diagnose_ggchord(
  plot,
  width = 7,
  height = 5,
  units = c("in", "cm", "mm", "px"),
  device = c("png", "pdf", "svg"),
  dpi = 144,
  min_size = 2,
  min_linewidth = 0.2,
  colour_vision = FALSE
)
```

Arguments

plot	A ggchord plot.
width, height	Positive target canvas dimensions.
units	Dimensions in "in", "cm", "mm", or "px".
device	Target device: "png", "pdf", or "svg". It must be available in the current R environment. Only this device is inspected.
dpi	Raster resolution; also defines pixel dimensions.
min_size, min_linewidth	Advisory minimum text size and line width, both in millimetres. These do not modify the supplied style.
colour_vision	Request optional simulated colour-vision hints.

Details

Results describe evidence, not a general quality certificate. Checks cover registered vertex bounds, measured text, small text/lines, oversized guides, and existing label/track diagnostics. Objects outside a panel with clip = "off" are legal when they still fit the canvas. Unregistered custom grobs, arrowhead extents, font embedding/fallback and cross-application transparency are not certified. Device warnings remain in the report.

Grayscale suggestions compare luminance of opaque discrete mapped colours (difference below 0.05); continuous gradients are deliberately excluded. With colour_vision = TRUE, optional colorspace public simulations for protan/deutan/tritan at severity 1 in linear RGB flag Lab distances below 10. These heuristic thresholds ignore spatial context, alpha compositing and viewing conditions, and never automatically change the plot or fail a run.

Value

A ggchord_diagnostics list with measurement, checks, text and guide measurements, colour suggestions, existing label_diagnostics, annotation_registry, track_registry, track_diagnostics, and captured device warnings. Field units and provenance are described in ggchord_layout_contract. measurement$status concerns panel measurement only; it never means that all quality or portability checks passed.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
p <- ggchord(seq_data = seqs) +
  geom_seq()
diagnose_ggchord(p, width = 8, height = 6, units = "in", device = "png")
```

`export_ggchord_layout` *Export computed ggchord geometry*

Description

Builds a ggchord plot and returns selected computed geometry in a stable, layer-aware structure. Every exported row records its source layer and, where applicable, its source input row. Original mapped columns already attached to the geometry are preserved. This is the recommended interface for consuming layout tables; `get_chord_layout()` remains available for the complete layout list. Both retain their documented fields. See ggchord_layout_contract for units, provenance and stability boundaries.

Usage

```
export_ggchord_layout(
  plot,
  include = c("seq", "ribbon", "gene", "feature", "labels", "axis", "track"),
  original_data = FALSE
)
```

Arguments

`plot`	A ggchord plot.
`include`	Character vector selecting any of `"seq"`, `"ribbon"`, `"gene"`, `"feature"`, `"labels"`, `"link"`, `"restriction"`, `"track"`, and `"axis"`. Point links and restriction sites require explicit selection.
`original_data`	Logical. Include the resolved per-layer input tables under `original_data`, default `FALSE`.

Value

An object of class `ggchord_layout_export`. Selected component names contain data frames. Circular layouts additionally contain an `annotation_registry` data frame. Its inner rows expose the physical track, radial width and boundaries; outer rows expose region, sector, annotation class, dominant and actual band, spill reason, leader-crossing count, slot, bounding box and leader corridor. Explicitly inside primer and restriction callouts register measured inner boxes rather than exterior bands or slots. The `metadata` member records rotation, aspect ratio, limits, coordinate units and transformation state. Collinear layouts instead include `sequence_layout`, containing each sequence's horizontal range, backbone y, orientation, units per bp, reversible x offset, anchor and source origin. `orientation_diagnostics` records directional support, confidence and unresolved cases. Position lanes remain local offsets and are not circular physical tracks. Their `label_diagnostics` records placed or hidden labels after shared linear collision handling, including source layer, row, reason and leader collisions. `connection_report` records retained and omitted ribbon input rows with a selection reason. When converted inputs are present, `conversion_reports` retains their conversion attributes, including source class, column renaming, sequence information and object metadata. Sequence break marks are included under `seq`.

Segmented linear displays retain one `sequence_layout` row per window, including `row_id`, segment order, break flags and display `gap_bp`. Source accession, origin/end/full length, local orientation, x offset and units per bp describe each reversible piece; gaps have no source mapping. Automatic segment break paths have `.decoration = "segment_break"` in the sequence geometry. `gap_bp` is display spacing, not omitted length. Quantitative geometry is selected with `track`; `track_registry` records each ID/sequence band, source layers, numerical limits, unit and physical measurement status. `track_diagnostics` records retained/missing/clipped source rows. Numerical track axes and explicit bp reference bars are included under `axis`; `axis_selection` records selected sequence axes. Explicit label shrinking adds `label_fit_diagnostics`, preserving original text, requested/minimum/final sizes and fallback or hiding reasons. `render_context` distinguishes layout-only estimates from actual panel measurements. For a target-size measured report use `diagnose_ggchord()`.

Examples

```
library(dplyr)
library(ggchord)
data(seq_data_example)
data(ribbon_data_example)
p <- ggchord(seq_data_example, ribbon_data_example) +
  geom_seq() +
  geom_link_ribbon()
exported <- export_ggchord_layout(p, include = c("seq", "ribbon"))
names(exported)
```

`filter_ggchord_ribbons`

Filter alignment ribbons before plotting

Description

Keeps the ribbon rows that satisfy all requested criteria and optionally sorts them. The returned data frame keeps every extra column and the original column order; the original row numbers are attached as the `source_rows` attribute. Each non-NULL numeric threshold must be one finite number. Columns used by a numeric criterion must be numeric; missing values fail that criterion.

Usage

```
filter_ggchord_ribbons(
  ribbon_data,
  seq_ids = NULL,
  min_pident = NULL,
  max_pident = NULL,
  min_length = NULL,
  max_evalue = NULL,
  min_bitscore = NULL,
  min_query_coverage = NULL,
  min_subject_coverage = NULL,
```

```
  keep_pairs = NULL,
  drop_self_links = TRUE,
  sort_by = NULL
)
```

Arguments

ribbon_data	data.frame with at least qaccver and saccver columns (normally the full ribbon_data format).
seq_ids	Optional character vector. Keep only rows where both the query and the subject are in seq_ids.
min_pident, max_pident	Optional numeric. Lower/upper bounds on pident.
min_length	Optional numeric. Lower bound on length.
max_evalue	Optional numeric. Upper bound on evalue (requires an evalue column).
min_bitscore	Optional numeric. Lower bound on bitscore (requires a bitscore column).
min_query_coverage	Optional numeric (0-100). Lower bound on the query coverage. Uses the qcovs column when present; otherwise computed from qlen and the query interval (requires qlen).
min_subject_coverage	Optional numeric (0-100). Lower bound on the subject coverage, computed from slen and the subject interval (requires an slen column).
keep_pairs	Optional data.frame/list/matrix describing an undirected set of sequence pairs. A data.frame or matrix with the first two columns used as query/subject IDs, or a list of length-2 character vectors. A row is kept when its query/subject pair matches any pair in either direction.
drop_self_links	Logical, default TRUE. Remove rows where qaccver == saccver.
sort_by	Optional character vector of column names. Prefix a name with "-" or "desc:" for descending order (e.g. c("pident", "-evalue")).

Value

A list with data (the filtered data frame, with source_rows attribute) and report (n_input/n_kept/n_removed, removed_by_reason, removed_rows and kept_rows).

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
```

```
out <- filter_ggchord_ribbons(links, min_pident = 95, min_length = 500)
out$report
```

`filter_restriction_sites` *Filter restriction sites for display*

Description

Filter restriction sites for display

Usage

```
filter_restriction_sites(
  sites,
  set = c("all", "unique", "unique_dual", "six_plus", "unique_6plus", "commercial",
    "reference"),
  enzymes = NULL,
  min_site_length = NULL,
  cuts = NULL,
  window = NULL,
  commercial_only = FALSE,
  parent_set = c("all", "commercial_all", "commercial_nonredundant")
)
```

Arguments

`sites`	Result from `find_restriction_sites()`.
`set`	Display preset. `"reference"` reuses the enzyme-set profile stored for an exactly recognized reference sequence. It errors when no such profile is available. A saved `"None"` profile returns no sites.
`enzymes`, `min_site_length`, `cuts`, `window`, `commercial_only`	Additional filters.
`parent_set`	Enzyme catalogue subset. `"commercial_nonredundant"` keeps one stable commercial representative per equivalent recognition group and site, preferring the database's preferred enzyme when present.

Value

A row subset whose biological coordinates are unchanged.

See Also

`ggchord_data_contract`

Examples

```
data(plasmid_example_pUC19)
sites <- find_restriction_sites(plasmid_example_pUC19,
  enzymes = c("EcoRI", "BamHI", "HindIII"))
unique_sites <- filter_restriction_sites(sites, set = "unique")
```

`find_common_features` *Find common biological features in DNA sequences*

Description

Searches the built-in common-feature database (or a compatible custom database) using curated exact-sequence annotations, exact DNA, and the detection mode stored for each database record. Restriction sites are intentionally outside this API; use `find_restriction_sites()` for those. Protein matching uses a standard-code six-frame translation, without alternative initiation, splicing or origin-crossing protein matches. Source `geneticCode` metadata does not select a translation table. For nonstandard codes use `mode = "dna"` or external code-aware analysis. `confidence` is a deterministic heuristic label, not a probability or experimental confirmation. `identity` and `coverage` describe the reported match method, not independently validated biological function.

Usage

```
find_common_features(
  sequence,
  database = NULL,
  types = NULL,
  features = NULL,
  circular = TRUE,
  mode = c("auto", "dna", "protein"),
  min_protein_identity = 0.9,
  min_protein_coverage = 0.8,
  resolve = c("best", "all")
)
```

Arguments

`sequence`	DNA text, a named character vector, or a data frame with `accver` and `sequence`.
`database`	Optional normalized database list or custom data frame.
`types`, `features`	Optional feature-type and feature-name/ID filters.
`circular`	Search across the sequence origin.
`mode`	Matching mode. `"auto"` first uses curated annotations for an exactly known sequence, then follows each record's detection mode. Records marked `exactProteinMatch` use exact protein followed by near-exact DNA; other records use exact DNA. `"protein"` explicitly enables exact and approximate protein matching.
`min_protein_identity`, `min_protein_coverage`	Approximate protein thresholds, each one finite number in `[0, 1]`.
`resolve`	Return deterministic best annotations or every candidate.

Value

A data frame directly usable by geom_feature() and geom_feature_label_repel(). feature_class and feature_shape provide semantic metadata and a geometry hint; neither affects radial stacking. Explicit user aesthetics and scales still take priority. Its segments list-column preserves the biological feature's segment structure.

See Also

ggchord_data_contract

---

find_ggchord_loci *Find genomic loci around explicitly identified features*

Description

Selects targets by feature_id and expands their genomic bounds using biological upstream/downstream directions. For a negative-strand target, upstream extends towards larger genomic coordinates. Unknown or bidirectional strands require equal upstream and downstream distances. Windows are clipped to the known full sequence length; clipping is reported.

Usage

```
find_ggchord_loci(
  seq_data,
  feature_data,
  targets,
  upstream = 0,
  downstream = 0
)
```

Arguments

seq_data	Sequence table containing unique accver and positive integer length values. Optional genome and genome_id columns are retained in the returned windows.
feature_data	Feature table containing feature_id, accver, start, end, and optionally strand. Missing strand means unknown.
targets	Non-empty character vector of distinct feature_id values. Selection is exact; no annotation text or accession versions are guessed.
upstream, downstream	Single non-negative integer numbers of bases added before and after each target in its biological direction.

Details

Repeated rows with the same `feature_id` represent segments of one target and must belong to the same sequence and strand. Their outer bounds define the target window. Different targets always produce separate windows, including overlapping targets on the same sequence. No intervening sequence is removed, concatenated or locally reversed. Ordered one-based inclusive genomic coordinates are required; origin-spanning circular features and protein coordinates need an explicit conversion before this operation.

The returned `locus_id` equals `target_id`. Pass the returned table as `loci` to `focus_ggchord_data()` to create independent focused sequences named by those IDs, including multiple windows from one source sequence.

Value

An ordinary data frame in target order with `target_id`, `locus_id`, `accver`, `start`, `end`, `strand`, unexpanded `target_start` and `target_end`, requested extensions, and `clipped_left`/`clipped_right`. The `target_source_rows` list column identifies feature input rows used for each target. Coordinates refer to the supplied sequence table; no implicit change from focused coordinates to an earlier source is made.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(gene_data_example)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
genes <- genes |>
  mutate(feature_id = paste(accver, start, end, sep = ":"))
find_ggchord_loci(seqs, genes, targets = genes$feature_id[1], upstream = 100, downstream = 100)
```

`find_primer_bindings` *Find sequencing-primer binding sites*

Description

Searches each distinct oligonucleotide once on both target strands and then attaches all catalogue aliases. Exact matching is the default. Optional `three_prime` matching uses the longest exact 3-prime suffix available, with no mismatches, and is intended for inspecting partial annealing rather than silently treating it as a full match. Binding hits are sequence-search results, not experimental validation. `melting_temperature` uses the Wallace composition estimate unless source annotations supply it; ambiguous bases do not contribute to that estimate.

Usage

```
find_primer_bindings(
  sequence,
  primers = NULL,
  set = c("universal", "all", "reference"),
  match = c("exact", "three_prime"),
  min_annealed_bases = 15L,
  circular = TRUE,
  unique_only = FALSE
)
```

Arguments

sequence	DNA text, a named character vector, or a data frame with accver and sequence.
primers	Optional custom primers with sequence and optional name and primer_id (one ID per distinct sequence), or a named character vector.
set	Bundled catalogue subset. reference returns authoritative primer annotations from a recognized bundled reference plasmid.
match	Full exact matching or longest exact 3-prime annealing.
min_annealed_bases	One finite integer of at least 8, the minimum suffix length for three_prime matching. Fractional values are rejected.
circular	Allow matches to cross the sequence origin.
unique_only	Keep only primer sequences with one binding site per target.

Value

One row per binding site. The aliases list-column preserves every catalogue name associated with the matched oligonucleotide.

See Also

ggchord_data_contract

Examples

```
target <- c(circle = "TTGCAAAAACG")
find_primer_bindings(
  target, primers = c(test = "ACGTTGCA"),
  min_annealed_bases = 8
)
```

`find_restriction_sites` *Find restriction pattern matches and cleavage coordinates*

Description

Find restriction pattern matches and cleavage coordinates

Usage

```
find_restriction_sites(
  sequence,
  enzymes = NULL,
  patterns = NULL,
  circular = TRUE,
  database = NULL,
  methylation = c("auto", "none", "dam_dcm")
)
```

Arguments

`sequence`	DNA text, a named character vector, or a data frame with `accver` and `sequence`.
`enzymes`	Optional enzyme names. A data frame is a compatibility spelling for `patterns`.
`patterns`	Optional custom definitions.
`circular`	Allow recognition and cleavage coordinates to wrap origin.
`database`	Optional parsed pattern database.
`methylation`	Methylation context. `"auto"` uses the audited context for recognized reference plasmids and otherwise makes no blocked-site claim; `"none"` disables blocking; `"dam_dcm"` evaluates source-backed Dam/Dcm sensitivities against the supplied sequence.

Value

One row per pattern match without display filtering or deduplication. Methylation-aware columns include `methylation_status`, `blocked_by`, `display_warning`, `methylation_motif`, and `sensitivity_source`. `enzyme_site_count` records the number of distinct cleavage/display positions for that enzyme on the complete supplied sequence, so it remains stable after display filtering. For an exactly recognized bundled reference sequence, the `reference_enzyme_profiles` attribute records its saved display set for use by `filter_restriction_sites()`.

See Also

`ggchord_data_contract`

`focus_ggchord_data` *Focus ggchord data on one or more loci*

Description

Crops sequence, gene and ribbon tables together and relocates retained coordinates to start at one in each output locus. Multiple loci from the same source sequence are supported; provide a unique `locus_id` column to control their output sequence IDs. Segmented annotations, cleavage positions and cached feature runs are cropped and relocated together. Source intervals, source origins and genomic left/right truncation flags are retained on the returned tables. Alignment cropping is a linear display projection, not a new alignment: identity is retained, and a scaled alignment length is marked as estimated.

Usage

```
focus_ggchord_data(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  loci,
  expand = 0,
  boundary = c("trim", "drop")
)
```

Arguments

`seq_data`	Sequence table containing `accver` and `length`.
`ribbon_data`	Optional ggchord alignment table.
`gene_data`	Optional gene or feature table.
`loci`	Data frame containing `accver`, `start`, and `end`, and optionally `locus_id`.
`expand`	Non-negative number of bases added on both sides, scalar or one value per locus.
`boundary`	Boundary policy. `"trim"` clips partially overlapping genes and alignments; `"drop"` keeps only rows fully contained in selected loci.

Value

A list of focused `seq_data`, `ribbon_data`, `gene_data`, normalized `loci`, and a compact `report` including retained/dropped source rows and truncated output rows. `.source_origin` is the original genomic position corresponding to local position one, including after repeated focusing.

See Also

`ggchord_data_contract`

focus_ggchord_track *Focus quantitative intervals or samples without bridging omitted windows*

Description

Applies the same explicit loci and output IDs as `focus_ggchord_data()` and `segment_ggchord_data()`. It never averages, interpolates, rescales values, or creates boundary samples. Missing values remain missing. Subsequent track layers group by the returned window accession, so separate windows cannot be joined by a line. Source coordinates and original row lineage survive repeated focusing.

Usage

```
focus_ggchord_track(
  data,
  seq_data,
  loci,
  type = c("point", "interval"),
  boundary = c("trim", "drop")
)
```

Arguments

`data`	Quantitative input table. Metadata and mapped values are retained.
`seq_data`	Sequence reference table.
`loci`	Explicit `accver`, `start`, `end` and optional `locus_id`, as in `focus_ggchord_data()`.
`type`	`"point"` requires `pos`; `"interval"` requires `start`/`end`.
`boundary`	`"trim"` (default) or `"drop"` for partly retained intervals.

Value

An ordinary data frame with focused coordinates. Its `ggchord_track_focus` attribute reports output-to-input rows and omitted input rows; `.source_*` columns retain earliest source coordinates.

See Also

`ggchord_data_contract`

Examples

```
library(ggchord)
library(dplyr)
data(plasmid_example_pUC19)
track_seq <- plasmid_example_pUC19 |>
  select(accver, length)
starts <- seq.int(1L, track_seq$length, by = 100L)
gc_fraction <- function(sequence) {
```

```
  bases <- strsplit(sequence, "")[[1]]
  mean(bases %in% c("G", "C"))
}
bins <- tibble(
  accver = track_seq$accver,
  start = starts,
  end = pmin(starts + 99L, track_seq$length)
) |>
  rowwise() |>
  mutate(gc = gc_fraction(substr(plasmid_example_pUC19$sequence, start, end))) |>
  ungroup()
loci <- tibble(accver = track_seq$accver, start = 501, end = 2000)
focus_ggchord_track(bins, track_seq, loci, type = "interval")
```

`gene_data_example` *Example gene annotation data*

Description

A compact plotting fixture generated deterministically from `examples/gene_track.tsv`. Eight features of varied visible lengths are distributed across each sequence and favour informative annotations. Because the source records are strongly strand-biased, the demonstration `strand` alternates between `"+"` and `"-"`; the biological source value remains available as `source_strand`.

Usage

```
gene_data_example
```

Format

A data frame containing the following columns:

- accver: Sequence accession/version or an unchanged custom identifier
- start: Gene start position
- end: Gene end position
- strand: Strand direction (+/-)
- anno: Gene annotation category
- source_strand: Strand direction in the source annotation file

geom_feature *Draw generic genomic features*

Description

A general layer for CDS, tRNA, rRNA, repeat, CRISPR, promoter or user-defined features. It supports directional arrows, blocks, notched chevrons and lollipops, all generated against the sequence's real local curve. Feature fill and geometry use independent role-specific scales.

Usage

```
geom_feature(
  mapping = NULL,
  data = NULL,
  shape = "arrow",
  width = NULL,
  arrow_head_length = 0.04,
  arrow_head_width = 1,
  arrow_head_style = c("shouldered", "flush", "triangle"),
  short_feature = c("auto", "wedge", "block"),
  segment_boundaries = TRUE,
  boundary_colour = "#666666",
  boundary_linewidth = 0.25,
  boundary_linetype = "22",
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping	Optional aesthetic mapping. Role aesthetics such as accver, start, end and strand may rename input columns; ordinary visual mappings are evaluated after geometry is generated.
data	data.frame with accver, start, end and strand; optional type, category and label. Factor start/end columns use their numeric labels, not their internal level codes. Coordinates must resolve to finite numbers. display_priority or logical prioritized_display. As an alternative to strand, directionality or direction may contain "forward", "reverse", "bidirectional", or "none".
shape	Fixed feature geometry. An explicitly supplied shape takes precedence over aes(feature_shape = ...): "arrow", "compact_arrow", "promoter_arrow", "primer_arrow", "marker", "block", "capped_line", "primer_arc", "chevron", or "lollipop". The default is "arrow". Use aes(feature_shape = type) together with scale_feature_shape_manual() to map categories to geometry.

`width` Optional numeric/vector/list controlling width in the shared feature geometry engine. In a circular feature-label composition, the base width responds to final label size within bounded limits.

`arrow_head_length`, `arrow_head_width`
Arrow-head dimensions in the sequence-local frame, shared with `geom_gene()`. Values above one for `arrow_head_width` create a protruding block-arrow head.

`arrow_head_style`
Arrowhead construction: `"shouldered"` keeps the requested protruding head, `"flush"` keeps it level with the body, and `"triangle"` draws a full triangular wedge.

`short_feature` Fallback for arrows too short to hold their requested head: an automatically compressed directional glyph, a wedge, or a block.

`segment_boundaries`
Draw explicit internal marks carried by a segmented feature. Contiguous source-segment joins use dotted dividers; a non-solid segment-level `line_style` may override them, while a `cleavage_arrows` coordinate adds or merges a biological cut mark. These marks never create additional arrowheads.

`boundary_colour`, `boundary_linewidth`, `boundary_linetype`
Appearance of internal segment joins.

`position` Feature placement. Use `"identity"`, `"strand"`, `"plasmid"`, `position_strand()`, `position_plasmid()`, or `position_feature_stack()`.

`show.legend`, `inherit.aes`
Standard ggplot2 layer arguments.

`...` Additional fixed polygon aesthetics.

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
seqs <- plasmid_example_pUC19 |>
  select(accver, length)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
# ASCII display copy for the default PDF help-example device.
features <- features |>
  mutate(anno = gsub("\u03b1", "alpha", anno, fixed = TRUE))
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_feature(
    data = features,
    shape = "block"
  ) +
  coord_circular()
```

`geom_feature_label` *Label generic genomic features*

Description

Label generic genomic features

Usage

```
geom_feature_label(
  mapping = NULL,
  data = NULL,
  label_orientation = c("feature", "horizontal", "radial", "tangent"),
  label_side = c("outside", "inside", "auto"),
  label_overlap = c("hide", "nudge", "allow"),
  label_wrap = NULL,
  label_rotation = NULL,
  label_radial_offset = 0.04,
  label_circum_offset = NULL,
  label_circum_limit = NULL,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping, data`	Standard layer inputs. Map feature text with `label`.
`label_orientation`	One of `"feature"`, `"horizontal"`, `"radial"`, or `"tangent"`. `"feature"` keeps both internal and adjacent labels tangent to the feature direction; compact labels move beside the feature.
`label_side`	One of `"inside"`, `"outside"`, or `"auto"`.
`label_overlap`	Fixed-label collision policy.
`label_wrap`	Optional character wrapping width.
`label_rotation`	Optional numeric/vector/list. Label rotation angle, default 0
`label_radial_offset`	Optional numeric/vector/list. Radial offset of labels, default 0.04
`label_circum_offset`	Optional numeric/vector/list. Circumferential offset of labels, default 0
`label_circum_limit`	Optional logical/vector/list. Whether to limit circumferential offset, default TRUE

position — Feature Position used for the label anchor. In `coord_collinear()`, an omitted Position shares the matching feature layer's lanes; an explicit Position changes only label anchors, leaving feature polygons in their own Position. If both inputs contain unique `feature_id` values, label inputs may be subsets or reordered; their accession, interval and strand must still agree. Without IDs, inheritance requires the same interval rows in order. Multiple matching geometry layers require an explicit label Position or unambiguous feature IDs.

show.legend, inherit.aes — Standard ggplot2 layer arguments.

... — Additional text parameters.

Value

A ggplot2 layer.

`geom_feature_label_repel`

Automatically arrange generic feature labels

Description

Automatically arrange generic feature labels

Usage

```
geom_feature_label_repel(
  mapping = NULL,
  data = NULL,
  label_layout = c("feature", "radial", "auto", "callout"),
  label_wrap = NULL,
  label_fit = c("wrap", "none", "ellipsis", "auto", "shrink"),
  label_max_lines = 2L,
  label_side = c("outside", "inside", "auto"),
  max_overlaps = Inf,
  external = TRUE,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  label_min_size = 1.5,
  label_segment_overlap = "fade",
  label_segment_overlap_alpha = 0.18,
  label_segment_linetype = "auto",
  ...
)
```

Arguments

mapping, data	Standard layer inputs. Map feature text with label.
label_layout	One of "feature", "radial", "auto", or "callout". The default "feature" follows plasmid-map convention: text is tried inside first, then assigned to a fixed-radius internal track. All non-external text is laid out character by character along its selected track radius rather than rotated as one straight string. A light leader appears when one or more intervening tracks separate the feature and its label. Choose another mode for unrestricted external callouts.
label_wrap, label_fit, label_max_lines	Text fitting controls. label_fit = "shrink" explicitly uses internal-first placement without truncating the text. Text size is bounded by label_min_size; unresolved labels keep their requested font and use adjacent/external placement or hiding. Other fitting modes retain their existing behavior.
label_side	One of "inside", "outside", or "auto".
max_overlaps	Maximum unresolved overlaps.
external	Whether the staged feature layout may place a label outside the circular backbone after inside and adjacent placement fail. When FALSE, unresolved labels are hidden instead of pushing inward forever. External labels are rendered as rounded callouts using a lightened form of the feature's resolved fill. When restriction-site labels coexist on a circular map, both retain their own layout semantics but register final boxes in one coordinate-owned exterior track region so they cannot occupy the same space.
position	Feature Position used for the label anchor. In coord_collinear(), an omitted Position shares the matching feature layer's lanes; an explicit Position changes only label anchors, leaving feature polygons in their own Position. If both inputs contain unique feature_id values, label inputs may be subsets or reordered; their accession, interval and strand must still agree. Without IDs, inheritance requires the same interval rows in order. Multiple matching geometry layers require an explicit label Position or unambiguous feature IDs.
show.legend, inherit.aes	Standard ggplot2 layer arguments.
label_min_size	Minimum text size in millimetres for explicit shrink, default 1.5. Must not exceed the requested text size. Original text, requested/final sizes and fitting reason are exported with the label.
label_segment_overlap	Character, default "fade". How portions of a leader line covered by another label are drawn: "fade" keeps them at reduced opacity, "clip" removes them, and "show" leaves the complete leader visible. This does not affect text wrapping, ellipsis or label hiding.
label_segment_overlap_alpha	Numeric in [0, 1], default 0.18. Relative opacity applied only to covered leader-line portions when label_segment_overlap = "fade".
label_segment_linetype	Character or numeric, default "auto". Leader-line linetype. "auto" draws solid lines, except for labels that were moved to the other side of their arc, which

are drawn dashed. Any other valid ggplot2 linetype (e.g. `"solid"`, `"dashed"`, `"dotted"`, or a numeric dash pattern) is used for all leader lines.

`...` Additional text parameters.

Details

Manual label rotation, offsets, orientation and overlap policy belong to `geom_feature_label()`. Their `label_*` controls are rejected by automatic layers rather than silently ignored.

Value

A composite text and leader-line layer.

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
seqs <- plasmid_example_pUC19 |>
  select(accver, length)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  slice_head(n = 4) |>
  select(accver, start, end, strand, type, anno)
# ASCII display copy for the default PDF help-example device.
features <- features |>
  mutate(anno = gsub("\u03b1", "alpha", anno, fixed = TRUE))
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_feature(data = features, position = position_plasmid()) +
  geom_feature_label_repel(data = features, position = position_plasmid()) +
  coord_circular()
```

`geom_feature_plasmid` *Draw SnapGene-style features on a circular plasmid*

Description

A concise plasmid-map preset built on `geom_feature()`. It uses the same generic feature engine, while selecting a compact circular track, protruding shouldered block arrows, and dashed internal segment joins.

Usage

```
geom_feature_plasmid(
  mapping = NULL,
  data = NULL,
  shape = "arrow",
  width = 0.058,
  arrow_head_length = 0.068,
```

```
  arrow_head_width = 1.42,
  arrow_head_style = "shouldered",
  short_feature = "auto",
  segment_boundaries = TRUE,
  boundary_colour = "#666666",
  boundary_linewidth = 0.25,
  boundary_linetype = "22",
  position = NULL,
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping	Optional aesthetic mapping. Role aesthetics such as accver, start, end and strand may rename input columns; ordinary visual mappings are evaluated after geometry is generated.
data	data.frame with accver, start, end and strand; optional type, category and label. Factor start/end columns use their numeric labels, not their internal level codes. Coordinates must resolve to finite numbers. display_priority or logical prioritized_display. As an alternative to strand, directionality or direction may contain "forward", "reverse", "bidirectional", or "none".
shape	Fixed feature geometry. An explicitly supplied shape takes precedence over aes(feature_shape = ...): "arrow", "compact_arrow", "promoter_arrow", "primer_arrow", "marker", "block", "capped_line", "primer_arc", "chevron", or "lollipop". The default is "arrow". Use aes(feature_shape = type) together with scale_feature_shape_manual() to map categories to geometry.
width	Track width; the default matches the compact feature bands used by desktop plasmid editors.
arrow_head_length, arrow_head_width	Arrow-head dimensions in the sequence-local frame, shared with geom_gene(). Values above one for arrow_head_width create a protruding block-arrow head.
arrow_head_style	Arrowhead construction: "shouldered" keeps the requested protruding head, "flush" keeps it level with the body, and "triangle" draws a full triangular wedge.
short_feature	Fallback for arrows too short to hold their requested head: an automatically compressed directional glyph, a wedge, or a block.
segment_boundaries	Draw explicit internal marks carried by a segmented feature. Contiguous source-segment joins use dotted dividers; a non-solid segment-level line_style may override them, while a cleavage_arrows coordinate adds or merges a biological cut mark. These marks never create additional arrowheads.
boundary_colour, boundary_linewidth, boundary_linetype	Appearance of internal segment joins.

position — Feature position. NULL (the default) automatically stacks overlapping features on compact lanes inside the plasmid backbone. Reuse the same explicit position_feature_stack() object for a separate label layer so polygons and labels share their lanes. Optional numeric display_priority (larger first) or logical prioritized_display can override the default longest-first ordering for overlapping features.

show.legend, inherit.aes — Standard ggplot2 layer arguments.

... — Additional fixed polygon aesthetics.

Value

A ggplot2 layer.

---

geom_gene *Add a gene arrow layer*

Description

A directed-gene shortcut using the generic geom_feature() constructor with arrow geometry and the independent gene_fill role. Gene input still requires a known + or - strand; use geom_feature() for unknown strands. The gene fill scale is kept independent from the ribbon's fill scale via a separate internal aesthetic used by the ribbon layer.

Usage

```
geom_gene(
  mapping = NULL,
  data = NULL,
  width = NULL,
  arrow_head_length = 0.04,
  arrow_head_width = 1,
  arrow_head_style = c("shouldered", "flush", "triangle"),
  short_feature = c("auto", "wedge", "block"),
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping — Default NULL (uses pre-computed data)

data — Default NULL (retrieved automatically from the layout)

width — Optional numeric/vector/list. Width of gene arrows, default 0.05

arrow_head_length, arrow_head_width — Shared arrow-head dimensions in the sequence-local frame.

arrow_head_style
: Arrowhead construction: shouldered, flush, or a full triangular wedge.

short_feature
: Fallback for arrows too short to hold their requested head: automatic wedge/block selection, a wedge, or a block.

position
: Feature placement. Use "identity", "strand", "plasmid", position_strand(), position_plasmid(), or position_feature_stack().

show.legend, inherit.aes
: Standard ggplot2 layer arguments.

...
: Additional arguments passed to geom_polygon()

Value

A ggplot2 layer. To annotate the genes with their labels, add a geom_gene_label() layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(gene_data_example)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
ggchord(seq_data = seqs, gene_data = genes) +
  geom_seq() +
  geom_gene()
```

---

geom_gene_label *Add a gene label layer*

Description

Draws the gene annotation labels on a chord diagram. This layer is independent from geom_gene(): add it after geom_gene() to annotate the gene arrows with their texts.

Usage

```
geom_gene_label(
  mapping = NULL,
  data = NULL,
  label_orientation = "horizontal",
  label_side = "outside",
  label_overlap = "hide",
  label_rotation = NULL,
  label_radial_offset = 0.04,
  label_circum_offset = NULL,
  label_circum_limit = NULL,
```

```
  label_wrap = NULL,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`	Default NULL (uses pre-computed data)
`data`	Default NULL (retrieved automatically from the layout)
`label_orientation`	Character, default `"horizontal"`. Text orientation relative to each sequence path: `"radial"`, `"tangent"`, or `"horizontal"`.
`label_side`	Character, default `"outside"`. Place labels outside the chord, use the strand-based `"auto"` side, or force `"inside"` placement.
`label_overlap`	Character, default `"hide"`. Fixed-label collision policy: omit later conflicting labels (`"hide"`), apply the legacy gentle adjustment (`"nudge"`), or keep all requested positions (`"allow"`). Input row order therefore provides a simple way to prioritize labels in `"hide"` mode.
`label_rotation`	Optional numeric/vector/list. Label rotation angle, default 0
`label_radial_offset`	Optional numeric/vector/list. Radial offset of labels, default 0.04
`label_circum_offset`	Optional numeric/vector/list. Circumferential offset of labels, default 0
`label_circum_limit`	Optional logical/vector/list. Whether to limit circumferential offset, default TRUE
`label_wrap`	Numeric or NULL, default NULL. When set, long gene annotations are wrapped at this many characters (e.g. 15), which makes the labels narrower and less prone to overlap.
`position`	Position adjustment. Supply the same `position_feature_stack()` used by a gene/feature layer when labels are drawn from separate data or without a geometry layer. In `coord_collinear()`, labels without an explicit Position share a matching geometry layer's lanes; explicit label Positions do not move the geometry layer. Unique `feature_id` columns allow label subsets and reordered labels when their accession, interval and strand agree with the geometry input. Multiple matching geometry layers require an explicit label Position or unambiguous feature IDs.
`show.legend, inherit.aes`	Standard ggplot2 layer arguments.
`...`	Additional arguments passed to `geom_text()`

Details

This is the deterministic, fixed-position label layer. By default labels sit outside their sequence and labels that would collide are omitted in input-row order. Use `label_overlap = "nudge"` to retain the legacy gentle adjustment, or `"allow"` to draw every label at its requested position. For automatic arrangement with leader lines, use `geom_gene_label_repel()` instead.

Value

A ggplot2 layer. To let the labels avoid each other and the genes (with leader lines), use `geom_gene_label_repel()` instead.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(gene_data_example)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand, anno)
ggchord(seq_data = seqs, gene_data = genes) +
  geom_seq() +
  geom_gene() +
  geom_gene_label()
```

---

geom_gene_label_repel *Add an automatically arranged gene label layer*

Description

Like `geom_gene_label()`, with deterministic external label placement. The default `"radial"` keeps horizontal text on a contour following each sequence; crowded names fan out along that contour. `"auto"` aligns left/right names in columns and uses radial placement elsewhere. `"arc"` rotates text along the sequence tangent.

Usage

```
geom_gene_label_repel(
  mapping = NULL,
  data = NULL,
  label_layout = "radial",
  label_wrap = NULL,
  label_fit = "wrap",
  label_max_lines = 2L,
  label_side = "outside",
  max_overlaps = Inf,
  label_segment_overlap = "fade",
  label_segment_overlap_alpha = 0.18,
  label_segment_linetype = "auto",
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  label_min_size = 1.5,
  ...
)
```

Arguments

mapping	Default NULL (uses pre-computed data)
data	Default NULL (retrieved automatically from the layout)
label_layout	Character, default "radial". Label layout: "radial" arranges horizontal external labels along sequence-relative contours, with straight normal leaders or a short normal departure followed by a long connector when displaced; "auto" uses shared vertical columns on the left/right and radial placement elsewhere; "arc" rotates text along the sequence tangent. The former "aligned" mode has been removed; use "auto" for side columns.
label_wrap	Numeric or NULL, default NULL. When set, long gene annotations are wrapped at this many characters (e.g. 15), overriding label_fit.
label_fit	Character, default "wrap". How labels whose initial boxes are crowded or unusually wide are fitted before deterministic layout: "wrap" preserves the complete annotation over multiple lines, "ellipsis" uses a single shortened line, "auto" wraps and adds an ellipsis only if text exceeds label_max_lines, and "none" disables adaptive fitting. Explicit "shrink" uses internal-first feature placement and reduces the complete text only as far as label_min_size. Labels that still cannot fit use the existing adjacent/external placement or hiding policy, at their requested size. This choice uses feature placement rather than the external layout selected by label_layout; other fitting modes are unchanged.
label_max_lines	Positive integer, default 2. Maximum number of lines used by adaptive "wrap" or "auto" fitting.
label_side	Character, default "outside". Which side of the arc the labels sit on. "auto" keeps the strand-based placement (same as before); "outside" moves labels that would be inside the chord (where they can overlap the ribbons) to the outside of their arc; "inside" does the opposite. Labels moved to the other side are connected with a dashed leader line (see label_segment_linetype).
max_overlaps	Numeric, default Inf. Hide labels that still overlap more than this many other labels after repulsion (ggrepel-style decluttering). Use a finite value to clean up crowded plots.
label_segment_overlap	Character, default "fade". How portions of a leader line covered by another label are drawn: "fade" keeps them at reduced opacity, "clip" removes them, and "show" leaves the complete leader visible. This does not affect text wrapping, ellipsis or label hiding.
label_segment_overlap_alpha	Numeric in [0, 1], default 0.18. Relative opacity applied only to covered leader-line portions when label_segment_overlap = "fade".
label_segment_linetype	Character or numeric, default "auto". Leader-line linetype. "auto" draws solid lines, except for labels that were moved to the other side of their arc, which are drawn dashed. Any other valid ggplot2 linetype (e.g. "solid", "dashed", "dotted", or a numeric dash pattern) is used for all leader lines.

position — Position adjustment. Supply the same position_feature_stack() used by a gene/feature layer when labels are drawn from separate data or without a geometry layer.

show.legend, inherit.aes — Standard ggplot2 layer arguments.

label_min_size — Minimum text size in millimetres for explicit label_fit = "shrink", default 1.5. It must not exceed the requested text size. Source text, requested/final size and fitting reason are retained in exported label geometry.

... — Additional text styles and drawing options. Automatic layouts own rotation, offsets, orientation and overlap policy; use geom_gene_label() for manual label_rotation, label_radial_offset, label_circum_offset, label_circum_limit, label_orientation or label_overlap. These manual controls are rejected here.

Details

The local outside/inside label organisation adapts concepts used by SnapGene and Geneious. ggchord keeps generic mode names and integrates those ideas into its own geometry system. See SnapGene feature labels and Geneious label options.

Low-level force, padding, orientation and segment arguments used by earlier releases have been removed and now produce an error. Use label_layout for automatic placement, or geom_gene_label() for manual rotation and offsets.

Value

A ggplot2 layer that draws labels and leader lines together.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(gene_data_example)
genes <- gene_data_example |>
  group_by(accver) |>
  slice_head(n = 2) |>
  ungroup() |>
  transmute(accver, start, end, strand = source_strand, anno)
ggchord(seq_data = seqs, gene_data = genes) +
  geom_seq() +
  geom_gene() +
  geom_gene_label_repel(label_wrap = 20)
```

`geom_link_line` *Draw point-to-point sequence links*

Description

Requires qaccver, saccver, qpos and spos. Positions use the same genomic coordinates as sequence tracks. Curves run from query to subject; an arrow supplied by `arrow()` therefore points to the subject by default.

Usage

```
geom_link_line(
  mapping = NULL,
  data = NULL,
  link_type = c("curve", "straight"),
  link_gap = NULL,
  link_ctrl_point = NULL,
  link_avoid = c("none", "smooth", "uniform"),
  link_branch = c("none", "query", "subject"),
  link_branch_fraction = 0.15,
  arrow = NULL,
  colour = "#59636D",
  alpha = 0.6,
  linewidth = 0.4,
  linetype = 1,
  position = "identity",
  show.legend = NA,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping, data`	Standard ggplot2 mapping and layer data, including data functions.
`link_type`	Either "curve" (default) or "straight".
`link_gap`	Explicit endpoint spacing. With the default `link_avoid = "none"`, `NULL` uses a fixed spacing of 0.035. Set `link_avoid` to `"smooth"` or `"uniform"` to opt in to entity-aware endpoint spacing.
`link_ctrl_point`	Bezier control points: two numbers, four numbers, or a list per input row.
`link_avoid`	One of "none" (default), "smooth", or "uniform". Explicit gap takes precedence.
`link_branch`	Share equal query or subject endpoints: "none", "query", "subject".
`link_branch_fraction`	Shared length relative to the shortest member, in (0, 0.5].
`arrow`	A grid arrow specification, or NULL.

`colour`, `alpha`, `linewidth`, `linetype`
: Fixed styles; mappings use link_* aesthetics. An explicitly supplied fixed value takes precedence over the matching mapping. Omit the fixed argument to retain a mapped style, including an inherited mapping when `inherit.aes = TRUE`.

`position`, `show.legend`, `inherit.aes`
: Standard ggplot2 layer arguments.

`...`
: Additional path parameters.

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
points <- links |>
  transmute(qaccver, saccver,
    qpos = floor((qstart + qend) / 2),
    spos = floor((sstart + send) / 2)
  )
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_link_line(data = points)
```

---

geom_link_ribbon *Add an alignment ribbon layer*

Description

Draws alignment ribbons. Identity is mapped to `ribbon_fill` by default; When pident is absent, the default is a fixed grey fill with no Identity guide. Alignment length and identity are never inferred from endpoints. Alternative mappings and legends are controlled with `aes()`, `scale_*()` and `guides()`.

Usage

```
geom_link_ribbon(
  mapping = NULL,
  data = NULL,
  link_ctrl_point = NULL,
  link_gap = NULL,
  link_avoid = c("uniform", "none", "smooth"),
```

```
  link_branch = c("none", "query", "subject"),
  link_branch_fraction = 0.15,
  fill = NULL,
  alpha = 0.42,
  colour = "#59636D",
  linewidth = 0.08,
  linetype = NULL,
  adjacent_only = NULL,
  pairs = NULL,
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping	Default NULL (uses pre-computed data)
data	Default NULL (retrieved automatically from the layout)
link_ctrl_point	Optional vector/list of Bezier edge controls. Two numbers specify one shared control; four numbers specify controls for the two ribbon edges. A per-row list may contain pairs of coordinate vectors. NULL uses the origin `c(0, 0)` in chord layouts and derives controls from the sequence endpoints in collinear layouts. With shared branches, controls shape the far ends while the junction follows the shared trunk.
link_gap	Optional numeric/vector controlling spacing between sequences and ribbon endpoints. With the default `link_avoid = "uniform"`, `NULL` uses one obstacle-aware clearance across each endpoint. Use `"smooth"` for clearance that changes only next to `geom_gene()` and `geom_feature()` polygons, or `"none"` for fixed spacing 0.035. Text and leader lines are ignored. Supplying a number disables the automatic rule and uses that exact spacing. Linear layouts reduce automatic baseline clearance to fit narrow channels; an explicit gap or feature clearance that cannot fit raises an error.
link_avoid	One of `"uniform"` (default), `"none"`, or `"smooth"`. Smooth avoidance changes clearance locally; uniform avoidance uses one clearance across an endpoint.
link_branch	Share equal query or subject endpoints: "none", "query", "subject".
link_branch_fraction	Shared length relative to the shortest member, in (0, 0.5].
fill	Optional fixed ribbon fill. `NULL` keeps the default Identity mapping.
alpha, colour, linewidth	Standard fixed ribbon styles. Explicit values take precedence over matching mappings; omitted defaults do not suppress user mappings. For example, omit `alpha` when mapping `ribbon_alpha`.
linetype	Fixed outline line type. The default `NULL` uses solid outlines for adjacent displayed sequences and dashed outlines for non-adjacent sequences. Chord adja-

cency includes the first/last pair. This follows the final display order, independently of alignment direction. Explicit fixed styles or `aes(ribbon_linetype = ...)` (also `aes(linetype = ...)`) replace the automatic styles. Use a matching scale for data mappings; automatic styles do not train a linetype scale. Global line-type mappings are used when `inherit.aes = TRUE`.

`adjacent_only` Whether only neighbouring sequences are connected. `NULL` uses TRUE in `coord_collinear()` and FALSE in `coord_chord()`. Set FALSE to retain cross-track ribbons. Chord neighbours include the first/last sequence pair.

`pairs` Optional data frame with `qaccver` and `saccver` selecting unordered comparison pairs. With `adjacent_only = NULL`, explicit pairs take precedence over the coordinate default. Omitted rows are reported by `export_ggchord_layout()`.

`position`, `show.legend`, `inherit.aes`
Standard ggplot2 layer arguments.

`...` Additional arguments passed to `geom_polygon()`

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
ggchord(seq_data = seqs, ribbon_data = links) +
  geom_seq() +
  geom_link_ribbon()
```

---

geom_primer *Draw primer binding ranges on a circular sequence*

Description

Primers are slender directional arcs drawn between the two lines of a double sequence backbone by default, not arrows in the feature stack. Add `geom_primer_label_repel()` for the matching unboxed label and leader.

Usage

```
geom_primer(
  mapping = NULL,
  data = NULL,
  side = c("between", "outside", "inside"),
```

```
  arc_offset = 0.045,
  arc_linewidth = 0.026,
  colour = "#A020F0",
  position = NULL,
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping, data	Standard layer inputs.
side	Backbone lane used by the primer arc. "between" uses the backbone centre line; "outside" and "inside" use arc_offset.
arc_offset	Distance from the backbone centre when side is "outside" or "inside".
arc_linewidth	Radial width of the primer arc.
colour	Primer arc colour.
position, show.legend, inherit.aes	Standard layer arguments.
...	Additional parameters passed to geom_feature().

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
seqs <- plasmid_example_pUC19 |>
  select(accver, length)
primers <- find_primer_bindings(plasmid_example_pUC19, set = "universal")
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_primer(data = primers) +
  geom_primer_label_repel(data = primers) +
  coord_circular()
```

---

geom_primer_label_repel

Arrange primer labels and leaders

Description

Primer labels are unboxed, use the primer colour, and show one-based inclusive binding coordinates by default. Their leaders start at the directional primer arc's arrow tip (the primer 3-prime end). Factor start/end columns use their numeric labels for both displayed text and retained binding-coordinate metadata, matching feature geometry.

Usage

```
geom_primer_label_repel(
  mapping = NULL,
  data = NULL,
  side = c("outside", "inside"),
  show_location = TRUE,
  colour = "#A020F0",
  max_overlaps = Inf,
  position = NULL,
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping, data	Standard layer inputs.
side	Side of the backbone on which the primer callout is arranged. "inside" retains the inner label layout, including when exterior restriction or feature labels share the map. The default is "outside".
show_location	Include the binding range in the label.
colour	Primer arc colour.
max_overlaps	Maximum unresolved overlaps.
position, show.legend, inherit.aes	Standard layer arguments.
...	Additional parameters passed to geom_feature().

Value

A composite text and leader-line layer.

---

geom_restriction_site *Draw restriction sites, labels and deterministic leaders*

Description

Draw restriction sites, labels and deterministic leaders

Usage

```
geom_restriction_site(
  mapping = NULL,
  data = NULL,
  label = TRUE,
  label_style = c("enzyme_position", "enzyme", "position"),
```

```
  label_order = c("auto", "enzyme_position", "position_enzyme"),
  label_side = c("outside", "inside", "auto"),
  leader = c("radial", "elbow", "straight"),
  min_label_gap = NULL,
  tick_length = 0.04,
  label_offset = 0.18,
  colour = "#202020",
  linewidth = 0.28,
  label_size = 3.2,
  fontface = NULL,
  family = NULL,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`, `data`	Standard layer inputs.
`label`	Draw labels.
`label_style`	Label enzyme and position, enzyme only, or position only.
`label_order`	Automatic mirrored label order, enzyme first, or position first.
`label_side`	Draw labels outside, inside, or automatically.
`leader`	Leader routing style. The default `"radial"` measures actual text boxes, separates site anchors from final radial label positions, and routes either a direct connector or an independent radial stub plus fan segment. All enzyme-facing text edges follow one close circular contour in genomic order. Dense lateral clusters use evenly spaced contour fans, while sparse clusters retain their natural angular positions. The former `"trunk"` alias was removed in v0.16.1; use `"radial"`. `"straight"` connects the tick tip directly to the final label edge, without detours around the sequence or other leaders. `"elbow"` uses one bend, starting with a local-normal departure where possible. Label positions and grouping are shared by all routing styles.
`min_label_gap`	Minimum genomic fraction between label slots. `NULL` derives a compact device-aware default from the rendered labels.
`tick_length`, `label_offset`	Local-normal distances.
`colour`, `linewidth`, `label_size`, `fontface`, `family`	Fixed appearance. With the default `fontface = NULL`, enzyme names are bold only when that enzyme cuts once in the supplied sequence; coordinates remain regular. An explicit `fontface` disables this automatic distinction.
`position`, `show.legend`, `inherit.aes`	Standard layer arguments.
`...`	Additional geom parameters.

Value

A ggplot2 layer.

See Also

ggchord_data_contract

`geom_seq` *Add a sequence arc layer*

Description

Draws arcs (or straight lines, depending on the curvature setting) representing sequences in the chord diagram. Sequence layout parameters (order, orientation, radius, curvature, colors, etc.) are specified here.

Usage

```
geom_seq(
  mapping = NULL,
  data = NULL,
  seq_order = NULL,
  seq_orientation = NULL,
  seq_gap = NULL,
  seq_radius = NULL,
  seq_curvature = NULL,
  seq_style = c("auto", "single", "double", "band"),
  seq_backbone_gap = 0.025,
  seq_backbone_width = 0.035,
  linewidth = 0.9,
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`	Default NULL (uses pre-computed data)
`data`	Default NULL (retrieved automatically from the layout)
`seq_order`	Optional character vector. Specifies the drawing order of sequences
`seq_orientation`	Optional numeric (1 or -1). Sequence orientation, default 1

`seq_gap` Optional numeric. Gap proportion between sequences, default 0.03 in chord coordinates. In `coord_collinear()`, a positive distance between consecutive backbones as a proportion of the longest backbone's width, default 0.3. The final sequence's gap is unused in linear layouts. Segmented linear displays measure it against the longest display row; windows in one row must request the same gap, and only gaps between rows affect vertical placement.

`seq_radius` Optional numeric (> 0). Sequence arc radius, default 1.0

`seq_curvature` Optional numeric. Signed arc bow (0=straight, 1=standard arc, negative=opposite bow), default 1.0. Finite positive and negative values are accepted without clipping; magnitudes above 1 amplify the bow.

`seq_style` Backbone geometry. `"auto"` uses a double backbone in `coord_circular()` and the historical single backbone in `coord_chord()`.

`seq_backbone_gap`
Gap between double backbone lines.

`seq_backbone_width`
Width of a band backbone.

`linewidth` Arc line width, default 0.9

`position`, `show.legend`, `inherit.aes`
Standard ggplot2 layer arguments.

`...` Additional fixed path aesthetics. `colour` (or `color`) sets `seq_colour`. Explicit fixed values take precedence over matching mappings; omit fixed arguments to retain mapped styles.

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
ggchord(seq_data = seqs) +
  geom_seq()
```

---

geom_seq_break *Mark cropped ends of linear sequences*

Description

Adds two short diagonal strokes at each truncated sequence end in `coord_collinear()`. Metadata returned by `focus_ggchord_data()` determines the truncated genomic ends; reversed sequence orientation moves each mark to the corresponding displayed end. These marks are decorations and do not change the source sequence or feature coordinates.

Usage

```
geom_seq_break(
  mapping = NULL,
  data = NULL,
  break_width = 0.02,
  break_height = 0.04,
  linewidth = 0.4,
  colour = "#3D4348",
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`, `data`	Standard layer mapping and sequence data. By default, use the plot's sequence table. Truncation is read from `.truncated_left`/`.truncated_right`, with source interval and length metadata as fallback when available.
`break_width`, `break_height`	Width and height of the complete break mark in linear layout units. These are layout distances, not millimetres.
`linewidth`, `colour`	Fixed line appearance. Ordinary visual aesthetics can also be mapped and controlled by their ggplot2 scales.
`position`, `show.legend`, `inherit.aes`	Standard ggplot2 layer arguments.
`...`	Additional parameters passed to `ggplot2::geom_path()`.

Value

A ggplot2 layer. Other ggchord coordinate systems are unsupported.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
loci <- tibble(
  accver = seqs$accver, start = floor(seqs$length * .25),
  end = floor(seqs$length * .75)
)
focused <- focus_ggchord_data(seqs, loci = loci)
ggchord(seq_data = focused$seq_data) +
  geom_seq() +
  geom_seq_break() +
  coord_collinear()
```

`geom_seq_center_label` *Add a circular-sequence centre label*

Description

Add a circular-sequence centre label

Usage

```
geom_seq_center_label(
  mapping = NULL,
  data = NULL,
  name = TRUE,
  length = TRUE,
  separator = "\n",
  name_style = list(),
  length_style = list(),
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`, `data`	Standard layer inputs.
`name`	Show the sequence name. A character value overrides the name.
`length`	Show the formatted sequence length.
`separator`	Separator between name and length.
`name_style`, `length_style`	Named lists of fixed text-style overrides for the name and length grobs. Supported fields are `colour`, `size`, `family`, `fontface`, and `lineheight`.
`show.legend`, `inherit.aes`	Standard ggplot2 layer arguments.
`...`	Additional arguments passed to `geom_text()`.

Value

A ggplot2 layer.

`geom_seq_label` *Add a sequence label layer*

Description

Places sequence labels at the midpoint of each sequence arc, radially offset from the arc. Labels can be styled via their radial offset, rotation, font size, text orientation and justification.

Usage

```
geom_seq_label(
  mapping = NULL,
  data = NULL,
  seq_label_radius = 1,
  seq_label_rotation = NULL,
  labels = NULL,
  seq_label_orientation = c("arc", "horizontal"),
  seq_label_hjust = NULL,
  seq_label_vjust = NULL,
  check_overlap = FALSE,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`	Default NULL (uses pre-computed data)
`data`	Default NULL (retrieved automatically from the layout)
`seq_label_radius`	Optional numeric/vector. Radial position of the labels as a multiplier of the sequence arc radius: 1 is on the arc, > 1 places the label outside (away from the chord center) and < 1 places it inside, default 1 Explicit values are incompatible with `coord_collinear()`, which places sequence labels beside the left endpoint of each horizontal track.
`seq_label_rotation`	Optional numeric/vector. Additional label rotation (degrees) on top of the arc-aligned orientation, default NULL (0). Ignored when `seq_label_orientation = "horizontal"`.
`labels`	Optional character vector. Override the label texts (defaults to the sequence labels from `geom_seq()` or the sequence IDs)
`seq_label_orientation`	Character, default "arc". Label text orientation: `"arc"` rotates the text along the sequence arc (and keeps it readable), `"horizontal"` draws every label horizontally, extending away from the chord center.

`seq_label_hjust`
: Optional numeric/vector. Horizontal justification of the labels, default NULL. The default arc orientation uses -0.2 so the text sits just inside the sequence; with `seq_label_orientation = "horizontal"` the justification is chosen automatically so the text extends away from the chord center.

`seq_label_vjust`
: Optional numeric/vector. Vertical justification of the labels, default NULL (0.5).

`check_overlap`
: Logical, default FALSE. When TRUE, labels that would overlap a previously drawn label are skipped (ggplot2's `geom_text()` option).

`position, show.legend, inherit.aes`
: Standard ggplot2 layer arguments.

`...`
: Additional arguments passed to `geom_text()`

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_seq_label()
```

---

geom_seq_region *Highlight regions along sequence arcs*

Description

Draws rectangular bands on sequence arcs for one or more coordinate intervals. This is useful for marking loci, repeats, CRISPR arrays, or other user-defined regions without turning them into gene arrows.

Usage

```
geom_seq_region(
  mapping = NULL,
  data = NULL,
  fill = "#F59E0B",
  colour = "#B45309",
  alpha = 0.25,
  region_width = 0.08,
  region_offset = 0,
  region_side = c("inside", "outside", "auto"),
```

```
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping	Default NULL (uses pre-computed data)
data	data.frame with at least accver, start, end; optional label, category and color. Factor start/end columns use their numeric labels, not level codes.
fill	Character. Default fill colour for regions, default "#F59E0B".
colour	Outline colour, default "#B45309".
alpha	Numeric (0-1). Region alpha, default 0.25.
region_width	Numeric. Band width in chord radius units, default 0.08.
region_offset	Numeric. Radial offset from the sequence arc, default 0.
region_side	Character. "inside", "outside" or "auto"; "auto" places regions inside the chord when possible.
position, show.legend, inherit.aes	Standard ggplot2 layer arguments.
...	Additional arguments passed to geom_polygon()

Value

A ggplot2 layer.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
region <- tibble(
  accver = links$qaccver[1], start = min(links$qstart[1], links$qend[1]),
  end = max(links$qstart[1], links$qend[1])
)
ggchord(seq_data = seqs) +
  geom_seq() +
  geom_seq_region(data = region)
```

`geom_seq_scale` *Add a shared base-pair scale bar to a linear display*

Description

Adds a free-standing reference bar in the empty space outside the sequence envelope. Its length uses the same bp-to-x conversion as every backbone in `coord_collinear()`, including independent windows on shared display rows. Display gaps are never interpreted as genomic length. Local reversals and original-source axis labels do not change the bar's bp scale. Chord and circular coordinates have local sequence axes instead and are rejected.

Usage

```
geom_seq_scale(
  mapping = NULL,
  data = NULL,
  length = NULL,
  label = NULL,
  location = c("bottomleft", "bottomright", "topleft", "topright"),
  offset = grid::unit(c(2, 6), "mm"),
  tick_height = grid::unit(1.5, "mm"),
  label_gap = grid::unit(1.5, "mm"),
  linewidth = 0.4,
  colour = "#3D4348",
  size = 3,
  position = "identity",
  show.legend = FALSE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping, data`	Standard ggplot2 layer inputs. Optional `scale_length` maps one positive integer bp length per bar; `label` supplies its text. Multiple bars require explicit `x` and `y` layout positions in `data` (or their role mappings), to avoid accidental overprinting.
`length`	Optional fixed positive integer length in bp. Cannot be combined with a `scale_length` mapping. `NULL` uses input lengths or an automatic reference length.
`label`	Optional fixed character label, or a function receiving the chosen bp lengths. `NULL` uses input labels or formatted `"... bp"` text.
`location`	Corner of the sequence envelope used for automatic placement.
`offset`	Two non-negative distances, horizontal then vertical. Absolute grid units are supported; plain numeric values are layout-unit distances.

`tick_height`, `label_gap`
: Non-negative absolute grid units or layout-unit numbers for the end ticks and space between the bar and caption. Physical conversion uses the current render context when available; device-free builds retain an explicitly estimated measurement status.

`linewidth`, `colour`, `size`
: Fixed line colour/width and text size. Text size follows ordinary ggplot2 millimetres. These may also use normal mappings and scales when not supplied as fixed arguments.

`position`, `show.legend`, `inherit.aes`
: Standard ggplot2 layer arguments.

`...`
: Additional fixed aesthetics passed to `ggplot2::layer()`.

Details

The bar does not claim to span one source interval. An explicit length can therefore exceed a single window and still be a valid distance reference in the shared linear metric. With no length, a small whole-number 1/2/5 reference is chosen from displayed window lengths. No source coordinates are modified and no missing genomic region is manufactured.

To change ticks and labels on sequence axes instead, use `scale_seq_position_continuous()`. A position scale configures existing axes; this geom adds an independent distance reference.

Value

One independent ggplot2 layer. Exported axis geometry retains the bp length, input row and measurement status for each bar.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
ggchord(seq_data = seqs) +
  geom_seq() +
  coord_collinear() +
  geom_seq_scale(length = 10000)
```

---

geom_track_tile *Draw explicit quantitative tracks along sequence backbones*

Description

Tile and bar layers require `accver`, `start`, `end`; lines require `accver` and `pos`. Map `track_value` for numeric height and `track_fill` or `track_colour` for visual mapping. Values are never inferred from column names. `group` separates line series; positions must increase strictly within each sequence/series. NA height breaks lines or omits bars, never becomes zero. Inf is rejected. Tiles may overlap in input drawing order; no averaging, interpolation or implicit aggregation is performed.

Usage

```
geom_track_tile(
  mapping = NULL,
  data = NULL,
  position,
  fill = "#86AFB9",
  colour = NA,
  alpha = 1,
  linewidth = 0.25,
  linetype = 1,
  bounds = c("error", "clip"),
  show.legend = NA,
  inherit.aes = FALSE,
  ...
)

geom_track_line(
  mapping = NULL,
  data = NULL,
  position,
  colour = "#387D88",
  alpha = 1,
  linewidth = 0.4,
  linetype = 1,
  closed = FALSE,
  bounds = c("error", "clip"),
  show.legend = NA,
  inherit.aes = FALSE,
  ...
)

geom_track_bar(
  mapping = NULL,
  data = NULL,
  position,
  baseline = 0,
  fill = "#86AFB9",
  colour = NA,
  alpha = 1,
  linewidth = 0.25,
  linetype = 1,
  bounds = c("error", "clip"),
  show.legend = NA,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping, data	Standard ggplot2 mapping and layer input.
position	A position_track() declaration. Every layer requires one.
fill, colour, alpha, linewidth, linetype	Fixed styles; fill and colour mappings use the dedicated track roles and scoped scales.
bounds	"error" (default) or "clip" for genomic coordinates. Clipping is reported and never changes source values.
show.legend, inherit.aes	Standard layer arguments.
...	Extra geom parameters.
closed	Explicitly close a complete circular line series.
baseline	Explicit numeric bar baseline, default zero. It is included in scale training and must remain representable under limits/transform.

Details

All coordinates are window-local, one-based inclusive. Interval bins occupy their full end - start + 1 bases, including one-base bins; line samples use the specified positions without binning.

Use focus_ggchord_track() for source windows. Lines never connect distinct windows, even when they share a display row. Circular lines close only with explicit closed = TRUE; missing endpoints are not joined across NA.

Value

A ggplot2 layer with independent input and a shared named band.

Examples

```
library(ggchord)
library(dplyr)
data(plasmid_example_pUC19)
track_seq <- plasmid_example_pUC19 |>
  select(accver, length)
starts <- seq.int(1L, track_seq$length, by = 100L)
gc_fraction <- function(sequence) {
  bases <- strsplit(sequence, "")[[1]]
  mean(bases %in% c("G", "C"))
}
bins <- tibble(
  accver = track_seq$accver,
  start = starts,
  end = pmin(starts + 99L, track_seq$length)
) |>
  rowwise() |>
  mutate(gc = gc_fraction(substr(plasmid_example_pUC19$sequence, start, end))) |>
  ungroup()
doc_theme <- theme_ggchord_publication(
```

```
  axis.text = element_text(size = 9), axis.text.check.overlap = TRUE
) +
  theme(legend.position = "bottom")
p_tracks <- ggchord(seq_data = track_seq) +
  geom_seq() +
  geom_seq_center_label() +
  geom_track_bar(aes(track_value = gc, track_fill = gc),
    data = bins,
    position = position_track("gc", width = unit(6, "mm"))
  ) +
  scale_track_value_continuous("gc", limits = c(0, 1), unit = "GC fraction") +
  scale_track_fill_gradientn("gc", colours = c("#EDF1EA", "#8AAE9C", "#345F58")) +
  coord_circular() +
  doc_theme
```

`get_chord_layout` *Get the chord layout from a ggchord plot*

Description

Returns the layout owned by an explicit ggchord plot. This avoids ambiguous cross-talk when several plots are built in one R session. For layer-aware tabular results, prefer `export_ggchord_layout()`. See ggchord_layout_contract for fields, units and cache ownership. Device/context changes invalidate cached estimates, including ordinary label layouts. `build = FALSE` then reports that no current layout is cached.

Usage

```
get_chord_layout(plot, build = TRUE)
```

Arguments

`plot`	A ggchord plot.
`build`	Logical. Build the layout when it is not cached, default TRUE.

Value

A chord layout list containing the computed geometry (sequence arcs, ribbon polygons, gene arrows, axis elements, extremes, colors, etc.)

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
p <- ggchord(seq_data = seqs) +
  geom_seq()
get_chord_layout(p)$seqs
```

ggchord *Layered chord, circular and collinear sequence plots*

Description

Visualize multi-sequence alignments, a single circular sequence or linear collinearity using ggplot2's layered grammar. The constructor supplies data and validation. Add geometry with `geom_*()`, select a global view with `coord_chord()`, `coord_circular()` or `coord_collinear()`, and use mappings/scales for data-driven styles and themes for non-data appearance. The layout is computed lazily when the plot is built (e.g. via `print()`, `ggsave()`, or `ggplot_build()`).

Usage

```
ggchord(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  debug = FALSE,
  validate = c("warn", "error", "none"),
  ...
)
```

Arguments

`seq_data`	data.frame/tibble with accver and length, or a `GRanges`, `Seqinfo` or `GInteractions` object with known full sequence lengths, or a named nucleotide Biostrings `XStringSet`. Unnamed sequence objects must first use `as_ggchord_seq()` with explicit `ids`. Optional S4 inputs use `as_ggchord_seq()`. When present, `sequence_unit` must be `"bp"`; amino-acid intervals require an explicit CDS projection, and arbitrary character widths are not bp lengths.
`ribbon_data`	Optional alignment table with qaccver, saccver, qstart, qend, sstart and send. length and pident are optional unless requested by a mapping, filter or statistic. A `GInteractions` object uses `as_ggchord_link()` without inventing identity or alignment length.
`gene_data`	Optional table with accver, start, end and strand. Annotation (anno) may be omitted. Directional `GRanges` are accepted; use `geom_feature(data = ...)` for unknown-strand intervals.
`debug`	Logical. Whether to output debug information, default FALSE
`validate`	Character, default `"warn"`. How to run the structured input-data validation (see `validate_ggchord_data`): `"warn"` emits a single summary warning when the data has problems and caches the full report on the plot object (`p$ggchord$validation`); `"error"` stops on severe problems; `"none"` skips the diagnostic validation (the cheap structural checks that prevent crashes are still performed).
`...`	Reserved for clear migration errors from removed constructor arguments. Use `labs()`, `coord_chord()` and `theme()` for presentation.

Value

A ggchord object (inherits from ggplot) to which geom_* layers can be added with +

See Also

ggchord_api_contract, ggchord_data_contract, ggchord_layout_contract, `view_ggchord()`

Examples

```
library(dplyr)
library(ggchord)
data(seq_data_example)
data(ribbon_data_example)
data(gene_data_example)

seqs <- seq_data_example |>
  select(accver, length)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
p <- ggchord(
  seq_data = seqs,
  ribbon_data = links,
  gene_data = genes
) +
  geom_seq() +
  geom_link_ribbon() +
  geom_gene()
print(p)
```

---

ggchord_api_contract *Candidate API and migration contract*

Description

The v0.16.4 API candidate consolidates the plotting, data and output contracts established in v0.16.1–v0.16.3. Candidate freezing is a development baseline, not a claim of v1.0.0 compatibility or complete platform/device validation. Correctness fixes remain possible with documented evidence and migration.

Plotting responsibilities

`ggchord()` supplies reference sequences and optional default input tables. Each layer retains its own data, mapping, parameters and identity. `aes()` selects data roles, scales control mapping rules/guides, geoms draw entities, Positions place them locally, coords select the canvas mode, and themes control non-data appearance. Explicit fixed styles take precedence over matching mappings; omit a fixed argument to retain its mapping. User scales take precedence over automatically inferred defaults.

geom_gene() remains the strictly directed gene shortcut over the generic geom_feature() constructor. Unknown-strand features remain unknown and use the generic feature entrance. The plasmid shortcut retains its distinct defaults. Fixed feature shape is shape; mapped shape is feature_shape. Gene/feature widths use width, label controls use label_*, and point/area links use link_gap and link_ctrl_point. These names do not merge the distinct geometries or their defaults.

Chord, circular and collinear remain separate coordinate modes. Circular maps reject inter-sequence links; collinear maps reject circular-only restriction/primer/centre-label layers. geom_seq_scale() is a collinear bp reference bar; scale_seq_position_continuous() configures genomic axes. A matching name never implies that every geom supports every coord.

Names and migration errors

The colour spelling is preferred in documentation; existing color spelling aliases remain supported. Selected ggplot2/grid reexports retain the original upstream object and follow that dependency's public contract. Their internals and future upstream signatures are not frozen by ggchord.

Retired arguments remain errors with replacement guidance; they do not execute compatibility code. In v0.16.1, fixed feature_shape, old width, label and ribbon endpoint names were removed in favour of the names above. gene_offset/feature_offset were also removed then; use position = position_strand(offset = ...). Explicit strand-named offsets are signed, so negate the former minus-strand magnitude when migrating it. The old v1.0.0 removal deadline no longer applies.

Automatic gene/feature labels own their rotation, offsets, orientation and overlap policy. Manual label_rotation, label_radial_offset, label_circum_offset, label_circum_limit, label_orientation and label_overlap supplied through their ... now fail rather than being ignored. Use geom_gene_label() or geom_feature_label() for manual placement, or choose automatic geometry with label_layout. The old "aligned" label request becomes label_layout = "auto"; restriction leader = "trunk" becomes leader = "radial".

Constructor unknown arguments stop. Ordinary geom, scale, guide and theme extras retain their documented forwarding and upstream error/warning rules; there is no promise that all unknown names fail uniformly. Geometric roles must be evaluated before layout, whereas supported visual mappings may use after_stat()/after_scale(). Unknown legacy label prefixes never imply that a similarly named replacement parameter exists.

Data, units and results

Input accver and query/subject identifiers are preserved without adding or removing versions. Biological strand, display orientation and link direction remain distinct. Coordinates are one-based inclusive bp after explicit external-format adaptation. Missing values, defaults and NULL are defined per responsibility, not by one package-wide coercion rule. Standalone cleaning defaults to sorting reversed endpoints; preparation retains direction by default. Readers retain file/files with shared path expansion. See ggchord_data_contract for source rows, optional dependencies, transformations, aggregation estimates and search limits.

Native layout distances, bp, angles, numerical track domains and physical mm are separate quantities. Track IDs share numerical domains and consistent band declarations; their physical placement is not a numeric value scale. Existing documented return members remain supported. Prefer export_ggchord_layout() for layer-aware tables; get_chord_layout() remains available for the complete layout. Preview paths and target-size diagnostic reports retain separate purposes. See

ggchord_layout_contract for stable fields, provenance, measurement status and plot/render cache ownership.

See Also

ggchord_data_contract, ggchord_layout_contract

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
seqs <- plasmid_example_pUC19 |>
  select(accver, length)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
# ASCII display copy for the default PDF help-example device.
features <- features |>
  mutate(anno = gsub("\u03b1", "alpha", anno, fixed = TRUE))
p <- ggchord(seq_data = seqs) +
  geom_seq() +
  geom_feature(data = features) +
  coord_circular()
export_ggchord_layout(p, include = "feature")$metadata$coordinate
```

---

ggchord_data_contract *Data workflow and biological coordinate contracts*

Description

ggchord separates reading, adaptation, validation, explicit cleaning and preparation. None of these steps runs BLAST, infers a missing full sequence length from features, verifies homology experimentally, or adds plot layers.

Details

Read The six `read_*()` functions parse local files. Both `file` and `files` accept path vectors and glob patterns. When both are supplied, `file` precedes `files`; matches expand in pattern order, and each normalized path is read once, retaining its first supplied spelling. Missing, empty, unmatched or directory paths fail. Text input must be UTF-8 (an initial BOM is accepted by table readers). Convert other encodings explicitly before reading. Import attributes record format, files and output counts. Tabular readers and FASTA lengths retain physical `.source_line`; optional `.source_file` gives row-level file identity. GenBank instead retains record/feature indices and source text in its report. Malformed field counts stop, rather than shifting fields between rows. Numeric conversion problems in BLAST/GFF3 warn and retain missing values for inspection and validation.

Adapt `as_ggchord_*()` accepts documented tables or optional interval/sequence objects. Required IDs and positive integral coordinates are checked immediately; zero-based tables must be converted explicitly. Metadata name collisions receive prefixes and a column map. Required coordinate NA is an error, while unrelated metadata NA survives. Seqinfo, object metadata, names and upstream provenance are retained in the conversion attribute. Optional packages are never installed automatically.

Validate `validate_ggchord_data()` reports all collected issues without changing input. `strict = TRUE` stops after collection if errors exist. Logical argument errors stop immediately. This validator describes constructor seq/ribbon/gene tables; generic unknown-strand features belong in preparation's named feature collections. Coordinate checks do not prove an alignment or a biological annotation.

Clean `clean_ggchord_data()` explicitly applies selected policies and reports original rows and old/new values. Its historical `reversed_interval = "sort"` is retained; use `"keep"` for directional links or circular-origin features. Sorting cannot recover their original meaning later. Unknown-ID `"keep"` is an audit policy, not a guarantee that the result is drawable. NULL means an absent optional table, and an empty table remains a table.

Prepare `prepare_ggchord_data()` adapts first, validates, then runs only requested steps in the documented order. Its cleaning default is `reversed_interval = "keep"`, because downstream operations use endpoint directions. Adaptation errors therefore occur before cleaning; clean/drop malformed IDs separately when needed. Named features/links remain independent collections. Reports distinguish stage-local rows from composed original ribbon source sets.

Coordinates and direction

Genomic intervals are one-based inclusive: `start == end` is one base. The explicit adapters enforce integral coordinates. Legacy constructor validation, cleaning and raw focus accept finite numeric positions; their success alone does not certify integral genomic coordinates. Prefer integral windows for biological data and use adapters before these legacy entries when strict integer checks are required. BED/PAF half-open `[a,b)` becomes `[a+1,b]`; zero-width records cannot become nonempty intervals and are rejected. Link endpoint order can encode relative direction; it is not a circular-origin join. Feature `start > end` is context-dependent: circular plotting can split an origin join, while linear window workflows require explicit ordered segments and an envelope. A raw focus locus uses min/max of its bounds, not an origin wrap. `strand` is biological annotation, `orientation` is display choice, and link `direction` is relative. Single-base link endpoints have no order: supplied strand or relative direction can provide evidence, otherwise direction remains unknown. Merge and bundle retain these rows independently instead of manufacturing aggregate endpoint directions. Unknown orientation is kept separate from known direction in density and deduplication. Unknown strands remain unknown in adapters; GFF3's explicit legacy `unstranded = "plus"` is an exception, with `source_strand` retained. Prefer `unstranded = "keep"` and generic feature layers when direction is unknown.

Plotting uses a continuous display axis, not a base-cell raster: feature/axis chord anchors use pos/length, point-link anchors use (pos-1)/length. These historical graphical offsets do not change stored coordinates. Single-base features can use a point or widened glyph. No new endpoint convention is imposed on plots in this data-workflow revision.

Selection, projection and aggregation

Focus crops all selected tables and composes earliest source coordinates. Overlapping windows can copy a row. Ribbon clipping interpolates an endpoint correspondence, rounds to bp and marks estimated length; identity is retained, not recalculated. Segment preparation creates separate windows and explicit gaps; it never concatenates their biological sequences. Position mapping is invertible within the selected linear window and rejects segmented gaps. CDS projection uses observed complete codons and explicit phase/partial policies; it does not translate with the stored genetic-code ID. Protein identity/length remain source-prefixed metadata when links are projected.

Filtering retains full rows. Deduplication selects representatives and reports discarded rows; it does not combine their statistics. Merge creates envelopes and reports source sets, lengths and identities. Bundle groups by pair, relative endpoint direction and midpoint bins; both endpoints may be reversed together in one group. Envelope endpoints follow the first row's absolute direction. Bundle length is a rounded length-weighted mean of source lengths, identity is a length-weighted mean, and conflicting extra metadata become typed NA. These summaries are not new alignments. Supply `group_by` for metadata that must remain separate. Density retains every input row and measures bin weight relative to that pair's maximum bin; it is not coverage or statistical probability. Optimization suggests display order/orientation without changing biological strand or query/subject roles.

Search helpers

Cluster links use declared membership, with explicit paralog and segment policies, not sequence alignment. Primer exact and three-prime suffix searches report zero mismatches; a partial suffix is not full-oligo validation. Wallace temperature is a simple composition estimate, not an experimental prediction. Restriction results are motif matches and source-backed cut offsets; missing cuts remain unknown. Methylation status uses supplied/reference context and a limited sensitivity catalogue, not a complete digestion model. Linear context never wraps sequence ends. Common-feature confidence is a deterministic heuristic rank, not a probability or experimental confirmation. Its six-frame protein search uses the standard codon table without alternative starts, splicing or circular-origin protein matches. Source `geneticCode` metadata is not interpreted as a selectable translation table. Use exact DNA mode or an external code-aware analysis for nonstandard translation.

See Also

`prepare_ggchord_data`, `read_genbank`, `as_ggchord_feature`, `project_ggchord_links`

---

ggchord_layout_contract

Layout results, units and ownership

Description

`export_ggchord_layout()` is the recommended interface for consuming layer-aware geometry tables. `get_chord_layout()` retains the complete layout list and its documented geometry, bounds, colour and label members. Neither entrance is deprecated. `view_ggchord()` returns a temporary image path; `diagnose_ggchord()` returns evidence for one requested render.

Geometry and provenance

Selected export components are data frames; absent components may be empty data frames without columns. Nonempty geometry has `layer_id`, `component` and `source_row`. An ID identifies one layer occurrence in that plot; editing/reordering layers can change IDs. A source row indexes the resolved input for that layer/component, not necessarily a physical file line. Generated decorations use missing source rows. Aggregates may carry `source_rows` membership instead of identifying one observation. `original_data = TRUE` includes resolved inputs for selected nonempty components, after role mapping and input adaptation. It is not a raw-file archive. File line/record provenance and conversion attributes retain their separate meanings from ggchord_data_contract.

Cartesian geometry and bounding boxes use native layout data units. Genomic source coordinates remain one-based inclusive bp; they are not converted to Cartesian distances. Angles use degrees except explicitly internal sequence-reference angles. Text sizes use mm. Metadata retains `coordinate_space`, `units`, `coordinate`, `rotation`, `rotation_applied`, `coord_transform_applied`, `ratio`, `fit`, `xlim`, `ylim` and `clip`. Exported coordinates already include layout rotation, before the ggplot2 coord transform. Native data units are not millimetres or genomic bp.

Coordinate-specific tables

Circular `annotation_registry` retains its documented identity, owner, component/source row, inner track/radius/width/boundaries, outer region, sector, band, dominant band, slot, spill reason, crossing count, bounding box and leader corridor fields. Radii, widths and boxes use layout data units; tracks/slots are build-local resource indices. Missing values denote fields that do not apply. These resources are distinct from named quantitative track IDs and Position lanes.

Collinear `sequence_layout` retains reversible window/source origins, horizontal range, backbone y, display orientation, units per bp, x offset and anchor. Segmented rows also retain row/window order, breaks and display `gap_bp`; omitted gaps have no source mapping. `orientation_diagnostics`, `connection_report` and `label_diagnostics` retain support, selection and placed/hidden-label evidence, including layer/row provenance where applicable.

`track_registry` retains track ID, accession, placement, source layers, numerical `limits` in original value units, optional `unit`, domain status and measurement status. Native offsets/widths use layout units; `_mm` fields are physical mm contributions. `normal_min`/`normal_max` use layout units. `track_diagnostics` records input rows, retention/reason, coordinate clipping and the original bar baseline. `axis_selection` records selected sequence axes. `label_fit_diagnostics` retains original text, requested, minimum/final sizes in mm and fallback/hiding reasons. Empty diagnostic tables need not have a fixed column prototype.

Measurement and cache scope

`render_context` and diagnostic `measurement` distinguish `estimated`, `unavailable` and `measured` panel status. `panel_mm`, `canvas_mm` and, when measured, `panel_bounds_mm` use physical mm. `units_per_mm` and `axis_units_per_inch` convert layout units along x/y; `units_per_inch` is a conservative scalar text metric. `geometry_metric_status` and `converged` describe refinement accuracy separately from panel measurement. Estimated contexts can have missing dimensions and convergence. Device/basis/font conditions state the source and limits of the evidence. Refinement counts and private solver/candidate/cache structures are implementation details.

A plot composition owns a fresh layout cache; a build/getter may populate it. Device/context changes invalidate estimates, even for ordinary labels. `get_chord_layout(build = FALSE)` re-

quires a current cache. Previews and diagnostics render independent copies and preserve the source cache. Reference-path reuse lasts only for one geometry calculation. Temporary render contexts restore their predecessor on normal or exceptional exit. Ordinary R assignment retains ggplot2 reference semantics for ggproto objects; arbitrary direct mutation of shared layer environments is not tracked as a new plot composition.

Stability and diagnostic limits

Existing documented members above remain supported. Additional columns may be attached for input metadata or geometry-specific needs; consumers should select named fields, not fixed column positions/counts. Numeric layouts can change with size, device, font and legitimate collision handling. Resource numbers and diagnostic reason details are not persistent biological IDs. No candidate layout, private cache or iteration count is a stable result.

Diagnostics retain `measurement`, `checks`, `text`, `guides`, `colour_suggestions`, `label_diagnostics`, `annotation_registry`, `track_registry`, `track_diagnostics` and `warnings`. Text/guide bounds and line/text sizes use mm; colour distances use the named model in each row. A measured panel does not certify collision-free geometry, font embedding, fallback, arbitrary grobs or cross-application portability. Preview paths live in a temporary directory and may be removed by cleanup; use ordinary `ggplot2::ggsave()` to keep a final file.

See Also

ggchord_data_contract, `export_ggchord_layout()`, `get_chord_layout()`, `view_ggchord()`, `diagnose_ggchord()`

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
p <- ggchord(seq_data = seqs) +
  geom_seq()
geometry <- export_ggchord_layout(p, include = "seq")
geometry$seq |>
  distinct(layer_id, component, source_row)
```

---

ggplot2-helpers *Selected ggplot2 helpers for ggchord plots*

Description

ggchord re-exports a deliberately small set of ggplot2 helpers that are directly useful when constructing, annotating, styling, or saving a chord plot. Cartesian geoms, coordinates, facets, generic scales, and complete theme presets remain in ggplot2 so IDE completion stays focused.

Usage

```
aes(x, y, ...)

after_stat(x)

after_scale(x)

annotate(
  geom,
  x = NULL,
  y = NULL,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  xend = NULL,
  yend = NULL,
  ...,
  na.rm = FALSE
)

labs(
  ...,
  title = waiver(),
  subtitle = waiver(),
  caption = waiver(),
  tag = waiver(),
  dictionary = waiver(),
  alt = waiver(),
  alt_insight = waiver()
)

ggtitle(label, subtitle = waiver())

guides(...)

theme(
  ...,
  line,
  rect,
  text,
  title,
  point,
  polygon,
  geom,
  spacing,
  margins,
  aspect.ratio,
```

```
    axis.title,
    axis.title.x,
    axis.title.x.top,
    axis.title.x.bottom,
    axis.title.y,
    axis.title.y.left,
    axis.title.y.right,
    axis.text,
    axis.text.x,
    axis.text.x.top,
    axis.text.x.bottom,
    axis.text.y,
    axis.text.y.left,
    axis.text.y.right,
    axis.text.theta,
    axis.text.r,
    axis.ticks,
    axis.ticks.x,
    axis.ticks.x.top,
    axis.ticks.x.bottom,
    axis.ticks.y,
    axis.ticks.y.left,
    axis.ticks.y.right,
    axis.ticks.theta,
    axis.ticks.r,
    axis.minor.ticks.x.top,
    axis.minor.ticks.x.bottom,
    axis.minor.ticks.y.left,
    axis.minor.ticks.y.right,
    axis.minor.ticks.theta,
    axis.minor.ticks.r,
    axis.ticks.length,
    axis.ticks.length.x,
    axis.ticks.length.x.top,
    axis.ticks.length.x.bottom,
    axis.ticks.length.y,
    axis.ticks.length.y.left,
    axis.ticks.length.y.right,
    axis.ticks.length.theta,
    axis.ticks.length.r,
    axis.minor.ticks.length,
    axis.minor.ticks.length.x,
    axis.minor.ticks.length.x.top,
    axis.minor.ticks.length.x.bottom,
    axis.minor.ticks.length.y,
    axis.minor.ticks.length.y.left,
    axis.minor.ticks.length.y.right,
    axis.minor.ticks.length.theta,
```

```
  axis.minor.ticks.length.r,
  axis.line,
  axis.line.x,
  axis.line.x.top,
  axis.line.x.bottom,
  axis.line.y,
  axis.line.y.left,
  axis.line.y.right,
  axis.line.theta,
  axis.line.r,
  legend.background,
  legend.margin,
  legend.spacing,
  legend.spacing.x,
  legend.spacing.y,
  legend.key,
  legend.key.size,
  legend.key.height,
  legend.key.width,
  legend.key.spacing,
  legend.key.spacing.x,
  legend.key.spacing.y,
  legend.key.justification,
  legend.frame,
  legend.ticks,
  legend.ticks.length,
  legend.axis.line,
  legend.text,
  legend.text.position,
  legend.title,
  legend.title.position,
  legend.position,
  legend.position.inside,
  legend.direction,
  legend.byrow,
  legend.justification,
  legend.justification.top,
  legend.justification.bottom,
  legend.justification.left,
  legend.justification.right,
  legend.justification.inside,
  legend.location,
  legend.box,
  legend.box.just,
  legend.box.margin,
  legend.box.background,
  legend.box.spacing,
  panel.background,
```

```
  panel.border,
  panel.spacing,
  panel.spacing.x,
  panel.spacing.y,
  panel.grid,
  panel.grid.major,
  panel.grid.minor,
  panel.grid.major.x,
  panel.grid.major.y,
  panel.grid.minor.x,
  panel.grid.minor.y,
  panel.ontop,
  panel.widths,
  panel.heights,
  plot.background,
  plot.title,
  plot.title.position,
  plot.subtitle,
  plot.caption,
  plot.caption.position,
  plot.tag,
  plot.tag.position,
  plot.tag.location,
  plot.margin,
  strip.background,
  strip.background.x,
  strip.background.y,
  strip.clip,
  strip.placement,
  strip.text,
  strip.text.x,
  strip.text.x.bottom,
  strip.text.x.top,
  strip.text.y,
  strip.text.y.left,
  strip.text.y.right,
  strip.switch.pad.grid,
  strip.switch.pad.wrap,
  complete = FALSE,
  validate = TRUE
)

element_blank()

element_line(
  colour = NULL,
  linewidth = NULL,
  linetype = NULL,
```

```
  lineend = NULL,
  color = NULL,
  linejoin = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  inherit.blank = FALSE,
  size = deprecated(),
  ...
)

element_rect(
  fill = NULL,
  colour = NULL,
  linewidth = NULL,
  linetype = NULL,
  color = NULL,
  linejoin = NULL,
  inherit.blank = FALSE,
  size = deprecated(),
  ...
)

element_text(
  family = NULL,
  face = NULL,
  colour = NULL,
  size = NULL,
  hjust = NULL,
  vjust = NULL,
  angle = NULL,
  lineheight = NULL,
  color = NULL,
  margin = NULL,
  debug = NULL,
  inherit.blank = FALSE,
  ...
)

margin(t = 0, r = 0, b = 0, l = 0, unit = "pt", ...)

rel(x)

ggsave(
  filename,
  plot = get_last_plot(),
  device = NULL,
  path = NULL,
  scale = 1,
```

```
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  limitsize = TRUE,
  bg = NULL,
  create.dir = FALSE,
  ...
)

last_plot()

waiver()

expansion(mult = 0, add = 0)

position_identity()

PositionIdentity
```

Arguments

x, y	Aesthetic expressions for mappings; x is also the staged expression or relative size, depending on the helper.
...	Additional named mappings, guide specifications, theme elements, fixed annotation aesthetics or device arguments, as appropriate for the called helper.
geom, xmin, xmax, ymin, ymax, xend, yend, na.rm	For annotate(), the geometry name, positional bounds and endpoints, and whether missing values are removed silently. In theme(), geom instead sets geometry defaults.
title, subtitle, caption, tag, dictionary, alt, alt_insight, label	Plot labels, title, subtitle, caption and tag; dictionary replaces variable labels and alt/alt_insight provide accessible descriptions. In theme(), title supplies inherited title styling.

line, rect, text, point, polygon, spacing, margins, aspect.ratio, axis.title, axis.title.x, axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left, axis.title.y.right, axis.text, axis.text.x, axis.text.x.top, axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right, axis.text.theta, axis.text.r, axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.theta, axis.ticks.r, axis.minor.ticks.x.top, axis.minor.ticks.x.bottom, axis.minor.ticks.y.left, axis.minor.ticks.y.right, axis.minor.ticks.theta, axis.minor.ticks.r, axis.ticks.length, axis.ticks.length.x, axis.ticks.length.x.top, axis.ticks.length.x.bottom, axis.ticks.length.y, axis.ticks.length.y.left, axis.ticks.length.y.right, axis.ticks.length.theta, axis.ticks.length.r, axis.minor.ticks.length, axis.minor.ticks.length.x, axis.minor.ticks.length.x.top, axis.minor.ticks.length.x.bottom, axis.minor.ticks.length.y, axis.minor.ticks.length.y.left, axis.minor.ticks.length.y.right, axis.minor.ticks.length.theta, axis.minor.ticks.length.r, axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y.left, axis.line.y.right, axis.line.theta, axis.line.r, legend.background, legend.margin, legend.spacing, legend.spacing.x, legend.spacing.y, legend.key, legend.key.size, legend.key.height, legend.key.width, legend.key.spacing, legend.key.spacing.x, legend.key.spacing.y, legend.key.justification, legend.frame, legend.ticks, legend.ticks.length, legend.axis.line, legend.text, legend.text.position, legend.title, legend.title.position, legend.position, legend.position.inside, legend.direction, legend.byrow, legend.justification, legend.justification.top, legend.justification.bottom, legend.justification.left, legend.justification.right, legend.justification.inside, legend.location, legend.box, legend.box.just, legend.box.margin, legend.box.background, legend.box.spacing, panel.background, panel.border, panel.spacing, panel.spacing.x, panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor, panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x, panel.grid.minor.y, panel.ontop, panel.widths, panel.heights, plot.background, plot.title, plot.title.position, plot.subtitle, plot.caption, plot.caption.position, plot.tag, plot.tag.position, plot.tag.location, plot.margin, strip.background, strip.background.x, strip.background.y, strip.clip, strip.placement, strip.text, strip.text.x, strip.text.x.bottom, strip.text.x.top, strip.text.y, strip.text.y.left, strip.text.y.right, strip.switch.pad.grid, strip.switch.pad.wrap, complete, validate

Theme elements and layout settings. Line, rectangle and text elements control appearance; spacing and margins use physical units, positions and justifications control placement. More specific axis, legend, panel, plot and strip elements inherit from their parents. complete marks a complete theme; validate checks element types. Consult ggplot2::theme for each element and its accepted values.

colour, linewidth, linetype, lineend, color, linejoin, arrow, arrow.fill, inherit.blank, size, fill, family, face, hjust, vjust, angle, lineheight, margin, debug
: Element styling: colours, fill, line width and type, line ends and joins, arrows, font family/face/size, alignment, angle, line spacing and text margins. inherit.blank follows blank parent elements; debug shows text bounds. size for line/rectangle elements is deprecated upstream; use linewidth.

t, r, b, l, unit
: Top, right, bottom and left margins and their physical unit.

filename, plot, device, path, scale, width, height, units, dpi, limitsize, bg, create.dir
: Save settings: destination filename/path, plot, device, scale multiplier, width/height and units, raster resolution, size limit, background and permission to create the directory.

mult, add
: Multiplicative and additive scale expansion, each a single value or lower/upper pair.

Details

Mapping helpers are aes(), after_stat() and after_scale(); ggplot2::stage() combines stages of one aesthetic. Use labs() and ggtitle() for titles, guides() for guide selection, and theme() with the element_*() helpers for appearance. annotate() adds fixed annotations. margin() and rel() specify theme sizes; waiver() and expansion() specify automatic or expanded scale settings. position_identity() and PositionIdentity are the upstream identity placement helpers. ggsave() saves a supplied plot; last_plot() retrieves ggplot2's last plot. Each symbol is the original ggplot2 function or object, and can also be called with its ggplot2:: namespace. unit() and arrow() come from grid.

---

grid-helpers *Physical units and direction arrows*

Description

These are the original grid functions, available after loading ggchord. Use physical units for theme spacing and arrows for point links.

Usage

```
unit(x, units, data = NULL)

arrow(angle = 30, length = unit(0.25, "inches"), ends = "last", type = "open")
```

Arguments

x
: Numeric unit values.

units
: Unit names, such as mm, inches or npc.

data
: Optional supplementary data for special grid units.

angle	Arrow-head angle in degrees.
length	Arrow-head length as a unit object.
ends	Which ends receive arrows: first, last or both.
type	Arrow-head style: open or closed.

Value

A grid unit or arrow object.

Examples

```
unit(2, "mm")
arrow(length = unit(2, "mm"), type = "closed")
```

---

guide_ggchord_legend *Guides for ggchord role aesthetics*

Description

Thin wrappers around ggplot2::guide_legend() and ggplot2::guide_colourbar() with compact defaults suitable for chord diagrams. All ordinary ggplot2 guide arguments remain available.

Usage

```
guide_ggchord_legend(
  title = ggplot2::waiver(),
  theme = NULL,
  position = NULL,
  direction = NULL,
  override.aes = list(),
  nrow = NULL,
  ncol = NULL,
  reverse = FALSE,
  order = 0,
  size_scale = 1,
  ...
)

guide_ggchord_colourbar(
  title = ggplot2::waiver(),
  theme = NULL,
  nbin = NULL,
  display = "raster",
  alpha = NA,
  draw.ulim = TRUE,
  draw.llim = TRUE,
  angle = NULL,
```

```
  position = NULL,
  direction = NULL,
  reverse = FALSE,
  order = 0,
  available_aes = c("colour", "color", "fill", "ribbon_fill", "gene_fill",
    "feature_fill", "region_fill"),
  size_scale = 1,
  ...
)

guide_ggchord_colorbar(
  title = ggplot2::waiver(),
  theme = NULL,
  nbin = NULL,
  display = "raster",
  alpha = NA,
  draw.ulim = TRUE,
  draw.llim = TRUE,
  angle = NULL,
  position = NULL,
  direction = NULL,
  reverse = FALSE,
  order = 0,
  available_aes = c("colour", "color", "fill", "ribbon_fill", "gene_fill",
    "feature_fill", "region_fill"),
  size_scale = 1,
  ...
)
```

Arguments

`title`	Guide title.
`theme`	Optional guide-specific theme.
`position`, `direction`	Guide position and direction.
`override.aes`	A list of legend-key aesthetic overrides.
`nrow`, `ncol`	Legend key layout.
`reverse`	Whether to reverse the key order.
`order`	Guide order.
`size_scale`	Positive multiplier for keys, spacing and typography. ggchord's automatically generated guides derive this from the active export device; direct calls default to 1.
`...`	Additional arguments passed to the corresponding ggplot2 guide.
`nbin`	Number of colourbar bins.
`display`	Colourbar display method.
`alpha`	Colourbar alpha override.

`draw.ulim`, `draw.llim`
: Whether to draw upper/lower limit ticks.

`angle`
: Tick-label angle.

`available_aes`
: Aesthetics accepted by the guide.

Value

A ggplot2 guide object.

---

link_ggchord_clusters *Convert explicit homology-cluster membership to feature links*

Description

Connects feature intervals belonging to the same declared cluster for explicitly requested sequence pairs. Membership is an input assertion, not an inferred alignment: no identity, alignment length or score is invented. Both sequence and feature identifiers are kept exactly as supplied.

Usage

```
link_ggchord_clusters(
  feature_data,
  membership,
  pairs,
  paralogs = c("drop", "all"),
  segment_policy = c("error", "span"),
  max_links = 10000
)
```

Arguments

`feature_data`
: Feature data frame with `feature_id`, `accver`, `start`, `end`, and optional `strand`, accepted by `as_ggchord_feature()`.

`membership`
: Data frame with `feature_id` and `cluster_id`. A feature can belong to one cluster; extra columns are retained in the report.

`pairs`
: Data frame with explicit `qaccver` and `saccver` comparisons.

`paralogs`
: How to handle multiple members per cluster and sequence: `"drop"` (default) or explicit `"all"`.

`segment_policy`
: `"error"` (default) rejects repeated feature IDs; `"span"` explicitly uses a validated discontinuous feature's outer bounds.

`max_links`
: Positive integer upper limit for generated rows, default 10000. Exceeding it stops before Cartesian products are allocated.

Details

The default `paralogs = "drop"` retains only comparisons containing exactly one member on each side. Ambiguous one-to-many/many-to-many comparisons are omitted and reported. `"all"` explicitly requests their Cartesian product. Its total size is checked against `max_links` before any links are expanded. Pair row order and its query/subject roles are retained. Self-pairs and repeated pairs (including the reversed pair) must be resolved by the caller.

Feature IDs normally identify one row. If repeated rows deliberately describe one discontinuous feature, `segment_policy = "span"` requires one sequence, one strand, and non-overlapping intervals. The link then anchors the feature's outer genomic bounds; real segments remain in the `qsegments` and `ssegments` list-columns. This span is a feature anchor, never an exon-to-exon alignment, and does not remove intervening sequence. Nested `segments` fields in a single feature row are retained unchanged.

Unknown feature IDs in membership are reported and omitted. Malformed identifiers, conflicting cluster assignments, ambiguous repeated features, unsupported origin-crossing intervals and unknown pair sequence IDs stop with an actionable error. Feature IDs not assigned to a cluster are also listed in the report. Negative strands reverse their link endpoints; unknown strands retain increasing coordinates and `orientation_known = FALSE`.

Value

A list with `data`, an ordinary link data frame, and `report`. Links retain cluster/member IDs, strands, source feature/membership rows and segment tables. `report$comparisons` lists member counts, output counts and retention/omission reasons for each requested pair and clusters represented on either side (clusters absent from both sides are omitted); `report$members` retains membership metadata with resolution status. `report$valid` is false when membership contains unknown feature IDs.

See Also

ggchord_data_contract

Examples

```
library(ggchord)
library(dplyr)
data(plasmid_example_pUC19)
data(plasmid_example_pETDuet_1)
homolog_source <- bind_rows(plasmid_example_pUC19, plasmid_example_pETDuet_1)
homolog_seq <- homolog_source |>
  select(accver, length)
cluster_features <- homolog_source |>
  find_common_features() |>
  filter(type == "CDS", anno == "AmpR") |>
  select(accver, start, end, strand, anno) |>
  arrange(match(accver, homolog_source$accver)) |>
  mutate(feature_id = paste(accver, start, end, sep = ":"))
# Membership is computed from exact strand-corrected CDS DNA equality, not gene names.
reverse_complement <- function(sequence) {
  bases <- strsplit(sequence, "")[[1]]
```

```
  chartr("ACGT", "TGCA", paste(rev(bases), collapse = ""))
}
coding_sequences <- cluster_features |>
  left_join(select(homolog_source, accver, sequence), by = "accver") |>
  rowwise() |>
  mutate(
    coding_dna = substr(sequence, start, end),
    coding_dna = if (strand == "-") reverse_complement(coding_dna) else coding_dna
  ) |>
  ungroup()
cds_dna <- pull(coding_sequences, coding_dna)
membership <- tibble(
  feature_id = cluster_features$feature_id,
  cluster_id = paste0("exact_CDS_", match(cds_dna, unique(cds_dna)))
)
homologs <- link_ggchord_clusters(cluster_features, membership,
  pairs = tibble(qaccver = homolog_seq$accver[1], saccver = homolog_seq$accver[2])
)
loci <- find_ggchord_loci(homolog_seq, cluster_features,
  targets = cluster_features$feature_id[cluster_features$anno == "AmpR"][1],
  upstream = 100, downstream = 100
)
gene_palette <- c("+" = "#56887C", "-" = "#B58169", "." = "#AAB9AE")
doc_theme <- theme_ggchord_publication(
  axis.text = element_text(size = 9), axis.text.check.overlap = TRUE
) +
  theme(legend.position = "bottom")
p_homologs <- ggchord(seq_data = homolog_seq) +
  geom_link_ribbon(data = homologs$data, fill = "#ADC5B9", alpha = .6) +
  geom_seq() +
  geom_gene(data = cluster_features) +
  geom_seq_label(size = 4) +
  coord_collinear() +
  scale_gene_fill_manual(values = gene_palette) +
  doc_theme
```

`map_ggchord_position` *Convert genomic positions to or from a linear ggchord layout*

Description

Maps positions for ordinary ggplot2 annotation layers without modifying source tables. Inverse mapping returns the chosen relative or source bp. In a segmented layout, specify the window accession rather than its parent source accession. Positions outside that window, including display gaps, produce an error instead of being extrapolated into omitted sequence.

Usage

```
map_ggchord_position(plot, accver, position, inverse = FALSE, source = FALSE)
```

Arguments

plot	A ggchord plot using coord_collinear().
accver	Sequence identifiers, scalar or one per position.
position	Numeric genomic positions, or layout x when inverse = TRUE.
inverse	Convert layout x back to bp.
source	Use original source coordinates after focusing.

Value

A data frame with accver, position, x and backbone y.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
p <- ggchord(seq_data = seqs) +
  geom_seq() +
  coord_collinear()
map_ggchord_position(p, seqs$accver[1], position = 100)
```

---

merge_ggchord_ribbons *Merge adjacent or overlapping alignment blocks of the same sequence pair*

Description

Merges alignment blocks that belong to the same (query, subject) pair, are adjacent (gap <= max_gap) or overlapping on both sequences, and are compatible. The merged pident is weighted by alignment length. Merging is deliberately conservative: blocks whose merged query and subject spans would differ by more than max(max_gap, 1), whose pident differs by more than min_pident_difference, or whose orientation differs (when require_same_orientation) are left unmerged. With the default orientation check, subject bounds must also progress in the order implied by the relative direction; crossing block orders remain separate. Query/subject identifiers always define separate groups, even when group_by selects additional metadata. An unmerged row retains its original alignment length. For a merged row, length is the query envelope span, not a newly computed alignment length; .length_estimated marks this display summary. Source lengths and identities are retained in the report. Overlap can count aligned bases more than once in the weighted identity; gaps can add unaligned bases to the envelope. Use the original blocks for alignment statistics. Rows with a single-base endpoint remain independent: endpoint order cannot define an absolute direction for an aggregate envelope.

Usage

```
merge_ggchord_ribbons(
  ribbon_data,
  max_gap = 0,
  min_pident_difference = 0,
  require_same_orientation = TRUE,
  group_by = c("qaccver", "saccver"),
  extra_columns = c("consistent", "first")
)
```

Arguments

`ribbon_data`	data.frame in ribbon_data format.
`max_gap`	Numeric, default 0. Maximum gap (in bp) allowed between two blocks on both sequences for them to be merged.
`min_pident_difference`	Numeric, default 0. When > 0, two blocks are only merged when their pident difference is <= this value.
`require_same_orientation`	Logical, default `TRUE`. Only merge blocks with the same relative query/subject direction and compatible progression of their bounds. Reversing both endpoints preserves relative direction. `FALSE` explicitly permits a coarser mixed-direction summary.
`group_by`	Character vector, default `c("qaccver", "saccver")`. Additional grouping columns; query/subject IDs always remain grouping keys.
`extra_columns`	How to handle columns other than the required ribbon fields when multiple rows are merged. `"consistent"` (default) keeps a value only when all source rows agree and otherwise stores a typed missing value; `"first"` retains the first source row's value.

Value

A list with `data` (merged data frame with `source_rows` attribute) and `report` (data.frame with `output_row`, `from_rows`, `n_merged`, and list-columns `source_length` and `source_pident` for every output row).

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send, pident, length)
out <- merge_ggchord_ribbons(links, max_gap = 100)
out$report
```

`optimize_ggchord_layout` *Optimize sequence order and orientation for ribbon readability*

Description

Searches deterministic sequence-order and orientation changes that reduce weighted ribbon centre-line crossings and long chord spans. It never modifies the supplied data and never accepts a worse arrangement. Inputs above 2,000 ribbons use a deterministic 64-bin approximation and record `approximate = TRUE` in the returned report.

Usage

```
optimize_ggchord_layout(
  seq_data,
  ribbon_data,
  seq_order = NULL,
  seq_orientation = NULL,
  optimize = c("order", "orientation"),
  weight = c("length", "count", "pident"),
  max_iter = 50L
)
```

Arguments

`seq_data`	Sequence data containing `accver` and `length`.
`ribbon_data`	Alignment data in ggchord ribbon format.
`seq_order`	Optional initial complete sequence order. The default uses the row order of `seq_data`.
`seq_orientation`	Optional initial orientation (1 or -1), scalar, complete vector, or named vector.
`optimize`	Any of `"order"` and `"orientation"`.
`weight`	Objective weight: `"length"`, `"count"`, or `"pident"`.
`max_iter`	Positive integer maximum number of improving search rounds.

Value

An object of class `ggchord_layout_optimization` containing `seq_order`, `seq_orientation`, before/after scores and a report.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
suggestion <- optimize_ggchord_layout(seqs, links, weight = "count")
suggestion$seq_order
```

---

plasmid_example_pUC19 *Circular plasmid sequence examples*

Description

Thirteen one-row data frames generated from every reference FASTA record under `examples/plasmid/`. The file stem is used unchanged as both `accver` and the display `label`; the DNA sequence is copied without biological modification. Feature and restriction annotations are generated at run time with `find_common_features()` and `find_restriction_sites()`.

Format

A one-row data frame with `accver`, display `label`, `length`, and the DNA `sequence`. The historical `plasmid_example_pUC19c` spelling is a compatibility alias of `plasmid_example_pUC19`, including its `accver`. It must not be confused with an independently imported GenBank pUC19c record such as L09137.2.

Source

`examples/plasmid/`.

---

position_feature_stack *Stack overlapping genes or features on local lanes*

Description

Assigns overlapping intervals on the same sequence and side to radial lanes. Explicit display priority is ordered first, then intervals by decreasing length. Every interval takes the nearest collision-free lane; feature type, name and colour never choose a lane. The resulting anchors are shared with gene label layers because stacking is solved before polygons and labels are generated. In `coord_collinear()`, these local lanes become parallel horizontal bands; they do not represent circular physical tracks or radii.

Usage

```
position_feature_stack(
  spacing = 0.1,
  side = c("strand", "outside", "inside"),
  base_position = NULL
)
```

Arguments

spacing	Positive minimum radial distance between adjacent feature lanes. Circular plasmid layout enlarges the first transition into a main label corridor and reserves at least one label track between deeper overlapping feature lanes.
side	Legacy track-side selector retained for compatibility when base_position is NULL. Its historical names retain their existing geometry.
base_position	Optional identity, strand, or plasmid Position applied before interval-overlap detection and lane allocation.

Value

A ggplot2 Position object for geom_gene() or geom_feature().

---

position_plasmid *Place both feature strands on one shared band*

Description

Place both feature strands on one shared band

Usage

```
position_plasmid(offset = -0.1)
```

Arguments

offset	Signed local-normal offset. Negative values place the shared band inside a circular sequence and positive values place it outside.

Value

A ggplot2 Position object.

position_strand *Place features on opposite sides according to strand*

Description

Positive signed offsets point away from the plot centre and negative offsets point inward. A scalar is expanded to `+ = offset` and `- = -offset`; explicitly strand-named values are already signed and are not flipped again. In `coord_collinear()`, the positive local normal points upward and the negative local normal downward, regardless of sequence orientation.

Usage

```
position_strand(offset = 0.1)
```

Arguments

`offset`	A finite scalar or a flexible per-sequence/per-strand specification.

Value

A ggplot2 Position object.

position_track *Place a quantitative layer in an explicit sequence-local band*

Description

Layers with the same track ID share a band and numerical scale across all displayed sequences and windows. Their declarations must agree. Outside means positive local normal (up in linear coordinates), inside negative. The coordinate system resolves physical placement; feature-stack lanes and sequence ring radii are unrelated to track IDs.

Usage

```
position_track(
  track,
  side = c("outside", "inside"),
  order = 1,
  width = grid::unit(4, "mm"),
  gap = grid::unit(1, "mm")
)
```

Arguments

track	One stable nonempty character ID.
side	"outside" or "inside".
order	Finite ordering priority, nearest band first. Ties use ID order.
width, gap	Positive band width and nonnegative preceding gap. Absolute grid units (mm, cm, inches or points) are resolved against the render context. Numeric values retain native layout units. Numeric per-sequence vectors/lists follow the usual sequence parameter formats.

Value

A ggplot2 Position object for the three track geoms.

Examples

```
position_track("depth", width = unit(4, "mm"))
```

---

prepare_ggchord_data *Prepare named ggchord tables with an explicit audit trail*

Description

Converts inputs using the existing as_ggchord_*() adapters and returns named feature/link collections alongside the constructor's three tables. Collections are passed explicitly to layers, for example geom_feature(data = result$features$sites); preparation never adds layers. Import files first with the appropriate reader. No file format, alignment, sequence length, genome membership or biological identity is guessed.

Usage

```
prepare_ggchord_data(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  features = list(),
  links = list(),
  mapping = list(),
  steps = list()
)
```

Arguments

seq_data	Sequence input accepted by as_ggchord_seq().
ribbon_data	Optional paired intervals accepted by as_ggchord_link().
gene_data	Optional directional gene input accepted by as_ggchord_feature(). Nondirectional intervals belong in features.

features, links	Named lists of feature or paired-interval inputs.
mapping	Named list with optional seq, ribbon, gene column mappings, and features/links lists of mappings named like the respective collections. Each column mapping is a named character vector.
steps	Named list of explicit helper argument lists. Supported names: clean, filter, deduplicate, merge, bundle, optimize. Data arguments are supplied by preparation and cannot be overridden.

Details

By default this only converts and validates. Optional operations run in the fixed order clean, filter, deduplicate, merge, bundle, optimize. Each entry in steps is a named list of arguments for the corresponding existing helper. An empty list requests its defaults. clean affects only the constructor tables; the other operations affect ribbon_data only. Preparation sets cleaning's reversed_interval = "keep" by default to preserve directional endpoints. Explicit "sort" cannot be followed by deduplication, merging, bundling or optimization, which use those endpoints. Those operations also reject previously sorted input whose direction contradicts the endpoints; supply the original directional link table. Geometry, named feature/link tables and the original inputs are never silently filtered, merged or reoriented. Optimization returns a suggestion. Expensive pairwise duplicate diagnostics are disabled in validation here; request steps = list(deduplicate = list(...)) to deduplicate explicitly.

An optional genome column in seq_data groups contigs as metadata. accver must remain globally unique; contigs are never concatenated. Stable feature IDs and parent ID lists are preserved when supplied, not manufactured from row numbers. Repeated IDs can describe feature segments; uniqueness for a specific join is checked by the consumer, such as find_ggchord_loci().

Value

A list with seq_data, ribbon_data, gene_data, features, links and report. report$valid describes final validity, with validation before/after, feature_issues, per-link validations, conversions, steps, effective step_arguments, a ribbon source_rows list (original input rows contributing to each output row), pair_counts, and optional layout_suggestion. Structural conversion errors stop; biological reference/bounds errors remain in the report and are never repaired by default. Helper stage reports number rows in that stage's input.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
data(gene_data_example)
```

```
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
prepared <- prepare_ggchord_data(seqs, links, features = list(genes = genes))
prepared$report
```

`primer_catalog` *Inspect the bundled sequencing-primer catalogue*

Description

The catalogue stores one searchable record per distinct oligonucleotide. Set `aliases = TRUE` to obtain the source rows, including multiple biological names that share one sequence. The source direction is retained only as a descriptive hint; binding strand is determined from the target sequence.

Usage

```
primer_catalog(set = c("all", "universal"), aliases = FALSE)
```

Arguments

`set`	Return all records or only the commonly used universal subset.
`aliases`	Return source alias records instead of the sequence catalogue.

Value

A data frame.

See Also

`ggchord_data_contract`

Examples

```
head(primer_catalog("universal"))
head(primer_catalog("universal", aliases = TRUE))
```

print.ggchord_clean *Print a cleaned ggchord data result*

Description

Shows the dimensions of the cleaned tables and the first rows of the change report.

Usage

```
## S3 method for class 'ggchord_clean'
print(x, ...)
```

Arguments

x	A "ggchord_clean" object.
...	Ignored.

Value

The object invisibly.

See Also

ggchord_data_contract

project_ggchord_features *Project protein domains or genomic features through a spliced CDS map*

Description

Protein input uses one-based inclusive observed-residue coordinates. Genomic output preserves exon gaps in segments and uses the parent's biological strand; its outer envelope is not a claim that introns encode the domain. The inverse projects only intersections with coding bases, reports excluded bases, and uses the CDS feature ID as its protein coordinate namespace. This namespace is not an inferred protein accession. Inputs may supply a genomic segments list-column to describe disjoint genomic pieces.

Usage

```
project_ggchord_features(
  feature_data,
  cds,
  from = c("protein", "genome"),
  codon_boundary = c("error", "expand")
)
```

Arguments

feature_data	Domain/feature data frame with feature_id, parent_id (or a single-valued parent_ids list-column), start and end.
cds	A map from build_ggchord_cds().
from	"protein" (default) or "genome".
codon_boundary	Inverse projection policy: "error" or "expand".

Details

Each domain retains its explicit feature_id, single CDS parent, source interval and source row. Source metadata survives with source_ prefixes on name conflicts. Multiple biological parents must be expanded explicitly before projection. Unknown parent, intronic-only, out-of-range, reversed, inconsistent-strand and ambiguous codon-boundary input produce errors. codon_boundary = "expand" explicitly permits an inverse interval to cover partial codons and reports this expansion; it does not modify source bases.

Value

Ordinary feature table with projected accver, start, end, strand, directionality, segments, parent IDs, genetic-code metadata, source coordinates and a ggchord_projection provenance attribute.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
coding <- features |>
  filter(anno == "AmpR", type == "CDS") |>
  select(accver, start, end, strand) |>
  mutate(feature_id = paste(accver, "AmpR", sep = ":"), phase = 0)
cds <- build_ggchord_cds(coding, genetic_code = 11)
# Select residues for display; this is not a detected domain.
selected <- tibble(
  feature_id = "selected_residues", parent_id = coding$feature_id,
  start = 2, end = 20
)
project_ggchord_features(selected, cds)
```

`project_ggchord_links` *Project explicitly ungapped protein links to genomic coding segments*

Description

Splits paired protein intervals wherever either CDS crosses a segment boundary, retaining query/subject roles and genomic endpoint directions. This is a codon-coordinate correspondence, not a new nucleotide alignment. Source protein identity and length remain prefixed metadata and are never emitted as nucleotide `pident` or alignment `length`. General gapped alignments require explicit per-column alignment information and cannot be projected by this endpoint-only helper.

Usage

```
project_ggchord_links(link_data, cds, ungapped = FALSE)
```

Arguments

`link_data`	Data frame with `qfeature_id`, `sfeature_id` identifying CDS models and ordered inclusive protein `qstart`, `qend`, `sstart`, `send`. Query and subject residue spans must be equal. If present, `gapopen` must be zero and source `length` must equal this residue span.
`cds`	A map from `build_ggchord_cds()`.
`ungapped`	Must explicitly be `TRUE`, confirming an ungapped residue correspondence rather than inferring it from equal endpoint spans.

Value

A paired genomic interval table with CDS parent IDs, source row, piece number, genomic strands, `projected_bases`, source-prefixed input metadata and projection provenance. No alignment identity is computed.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
coding <- features |>
  filter(anno == "AmpR", type == "CDS") |>
  select(accver, start, end, strand) |>
  mutate(feature_id = paste(accver, "AmpR", sep = ":"), phase = 0)
cds <- build_ggchord_cds(coding, genetic_code = 11)
# Self-correspondence of one real CDS, solely to demonstrate coordinate conversion.
```

```
# The same observed residues are paired, so the correspondence is ungapped.
selected <- tibble(
  qfeature_id = coding$feature_id, sfeature_id = coding$feature_id,
  qstart = 1, qend = 20, sstart = 1, send = 20
)
project_ggchord_links(selected, cds, ungapped = TRUE)
```

`read_bed` *Read BED intervals as ggchord features*

Description

Reads standard BED3 through BED12. Starts are converted from zero-based half-open intervals to one-based inclusive intervals; accession identifiers are kept verbatim. Blank lines, comments, and custom-track header lines are skipped. Every data row in a file must have the same number of fields. Zero-width insertion records cannot be represented as nonempty genomic intervals and produce an explicit error.

Usage

```
read_bed(file = NULL, files = NULL, source_file = FALSE)
```

Arguments

`file`	Optional path vector or glob patterns; read before `files`.
`files`	Optional paths or wildcard patterns for multiple BED files.
`source_file`	Logical. Add a `.source_file` column, default `FALSE`.

Details

BED names are labels, not guaranteed unique feature IDs. Unspecified strand remains `"."` and `directionality = "none"`; use `geom_feature()` for these records. Optional fields retain their BED names and coordinate conventions, including zero-based `thickStart`, `thickEnd`, and relative `blockStarts`. BED12 additionally creates a `segments` list-column of one-based inclusive block intervals after checking block counts, boundaries and non-overlap. Blocks are retained in genomic order on both strands. No colour or shape is inferred from BED display fields. BED extensions beyond column 12 require an explicit data-frame conversion with `as_ggchord_feature()`.

Value

A feature data frame with `accver`, `start`, `end`, `strand`, `directionality`, original `chromStart`/`chromEnd`, optional BED fields, and `.source_line` (the physical file line). A `ggchord_import` attribute records the input/output coordinate conventions and files.

See Also

https://genome.ucsc.edu/FAQ/FAQformat.html#format1

`ggchord_data_contract`

Examples

```
library(dplyr)
data(plasmid_example_pUC19)
feature_source <- find_common_features(plasmid_example_pUC19)
features <- feature_source |>
  select(accver, start, end, strand, type, anno)
bed <- tempfile(fileext = ".bed")
bed_rows <- features |>
  transmute(accver, start0 = start - 1L, end)
write.table(bed_rows, bed,
  sep = "\t", quote = FALSE, row.names = FALSE, col.names = FALSE
)
read_bed(bed)
unlink(bed)
```

---

read_blast *Read one or more BLAST tabular output files into ribbon_data format*

Description

Both `file` and `files` accept literal path vectors or wildcard patterns. Repeated resolved paths are read once, in first-occurrence order. Standard `-outfmt 6` and `-outfmt 7` use the default 12 tabular fields; the historical extended 17-column format is optional. Omitting `-outfmt` entirely produces pairwise text, which this tabular reader does not parse.

Usage

```
read_blast(
  file = NULL,
  files = NULL,
  format = c("auto", "outfmt6", "outfmt7", "custom"),
  col_names = NULL,
  comment = "#",
  source_file = FALSE,
  ...
)
```

Arguments

`file`	Optional path vector or glob patterns; read before `files`.
`files`	Optional character vector of BLAST tabular output files (literal paths and/or wildcard patterns). All matched files are read and combined into one data.frame.
`format`	Character. `"auto"` (default) detects the column layout from the file; `"outfmt6"` requires the standard 12-column layout, `"outfmt7"` parses its `# Fields:` declaration, and `"custom"` requires `col_names`.
`col_names`	Optional character vector naming the columns in the file, used with `format = "custom"` or to override auto-detection.

comment	Character comment character, default "#" (BLAST outfmt 7 header lines start with #).
source_file	Logical. Add a .source_file column when reading one or more files, default FALSE.
...	Additional arguments passed to utils::read.table() (e.g. na.strings).

Details

The reader also parses BLASTp tabular output. It preserves the producer's numeric endpoints and does not infer or convert amino-acid/nucleotide units. Protein intervals require an explicit CDS projection before genomic plotting; reading a protein HSP does not establish an ungapped nucleotide correspondence. See project_ggchord_features() and project_ggchord_links().

Value

A data.frame with the required ribbon columns first (qaccver, saccver, qstart, qend, sstart, send) followed by any preserved optional columns. A ggchord_import attribute records files, per-file output row counts and the one-based inclusive coordinate convention.

See Also

ggchord_data_contract

Examples

```
## Not run:
library(dplyr)
# Standard 12-column records; requires the source checkout or tutorial input bundle.
source <- read_blast("examples/blastn-standard/MT108731.1__MT118296.1.tsv")
links <- source |>
  select(qaccver, saccver, qstart, qend, sstart, send)

## End(Not run)
```

---

read_fasta_lengths *Read one or more FASTA files into seq_data format*

Description

file reads a single FASTA file; files reads multiple files at once and combines them. files accepts a character vector of literal paths and/or wildcard patterns (e.g. "examples/fasta/*.fna").

Usage

```
read_fasta_lengths(
  file = NULL,
  files = NULL,
  header_delim = NULL,
  source_file = FALSE
)
```

Arguments

file	Optional path vector or glob patterns; read before files.
files	Optional character vector of FASTA files (literal paths and/or wildcard patterns). All matched files are read and combined.
header_delim	Optional character. When given, each header is split at every occurrence of this delimiter and only the first piece is kept.
source_file	Logical. Add a .source_file column when reading one or more files, default FALSE.

Value

A data.frame with columns accver and length. A ggchord_import attribute records files and per-file output row counts.

See Also

ggchord_data_contract

Examples

```
## Not run:
library(dplyr)
# Requires the source checkout or tutorial input bundle.
source <- read_fasta_lengths("examples/fasta/MT108731.1.fna")
seqs <- source |>
  select(accver, length)

## End(Not run)
```

---

read_genbank *Import local GenBank or GBFF records using an optional mature parser*

Description

Uses the public Bio.SeqIO.parse(..., "genbank") API in an explicitly selected Python environment with Biopython installed. The R package jsonlite exchanges structured temporary JSON with that process. This function never installs packages, downloads records, or runs during plot building. It reads all selected records into memory and is intended for small to moderate local annotation files, not chromosome-scale streaming.

Usage

```
read_genbank(
  file = NULL,
  files = NULL,
  feature_types = NULL,
  lengths = NULL,
  python = "python3",
  unsupported = c("error", "report"),
  source_file = FALSE
)
```

Arguments

file	Optional path vector or glob patterns; read before files.
files	Optional paths or wildcard patterns for several files.
feature_types	Optional feature types to select. NULL retains all.
lengths	Optional named numeric vector of known full sequence lengths, following as_ggchord_seq().
python	Interpreter name or path, default "python3". It must already have the optional Biopython package installed. No environment is modified.
unsupported	"error" or "report" for locations without a safe local interval representation.
source_file	Add .source_file columns to output tables, default FALSE. File provenance is retained in the report regardless.

Details

Record IDs, including accession versions, are retained exactly as returned by the source parser. Full parsed record annotations, feature qualifiers, source locations, and original file text are retained in report. Feature IDs use explicit ID, then protein_id for CDS, then locus_tag, then a non-placeholder parser feature ID. Missing IDs remain missing. Shared locus_tag values do not invent gene/CDS parent relationships; explicit Parent qualifiers become parent_ids.

Simple and joined local locations retain one-based inclusive segments in parser order. Before/after fuzzy bounds retain their source descriptors and partial flags; they are displayed at the reported boundary rather than treated as exact biology. Remote locations, missing or unsupported fuzzy positions, zero-width sites, and mixed-strand joined features are never flattened into guessed intervals. unsupported = "error" stops by default; "report" retains these features in report$unplaced_features instead. An absent ORIGIN sequence does not imply an unknown declared length. Truly missing/zero lengths require explicit lengths; feature endpoints are never used to infer full lengths. Protein records require an explicit protein-coordinate workflow instead of this nucleotide reader.

Value

A list with seq_data, named features$annotation, and report. Tables can be supplied to prepare_ggchord_data(). Feature rows include feature_id, parent_ids, segments, source record/feature indices, qualifiers, source_location, partial_start/partial_end, and location_operator. report$records retains the complete parsed source CDS rows also expose source codon_start, genetic_code, biological partial5/partial3, and translation_exception qualifiers for explicit downstream coding-coordinate models. Missing CDS codon_start uses the INSDC default

1, marked by codon_start_source = "format-default". Bound-only flags remain in bound_partial5/bound_partial3; CDS partial flags also record initial codon offset or terminal coding remainder, with their evidence in partial5_source/partial3_source. They describe the observed coding extent, not proof of a complete biological protein. No per-segment phase values are invented. segment_order preserves the source's declared coding-piece order independently of segment_index. metadata, raw_files retains original text, and unplaced_features identifies every selected feature not placed under "report".

See Also

https://biopython.org/docs/latest/api/Bio.SeqIO.html

ggchord_data_contract

Examples

```
## Not run:
# Requires the source checkout or tutorial input bundle and optional Biopython.
imported <- read_genbank("examples/format-inputs/pUC19c-L09137.2.gb")
imported$report

## End(Not run)
```

---

read_gff3 *Read one or more GFF3 files into gene_data format*

Description

file reads a single GFF3 file; files reads multiple files at once and combines them. files accepts a character vector of literal paths and/or wildcard patterns (e.g. "examples/gff3/*.gff3").

Usage

```
read_gff3(
  file = NULL,
  files = NULL,
  feature_types = "CDS",
  anno_from = c("product", "Name", "gene", "ID"),
  unstranded = c("plus", "drop", "keep"),
  source_file = FALSE
)
```

Arguments

file	Optional path vector or glob patterns; read before files.
files	Optional character vector of GFF3 files (literal paths and/or wildcard patterns). All matched files are read and combined.
feature_types	Character vector of feature types to keep, default "CDS".

anno_from	Character vector of GFF3 attribute keys, tried in order to fill the anno column.
unstranded	"plus" (legacy default) assigns unknown strands to +; "drop" removes them. Use "keep" with geom_feature() to retain them as nondirectional . intervals. source_strand always records the input.
source_file	Logical. Add a .source_file column when reading one or more files, default FALSE.

Value

A data.frame with accver, start, end, strand, anno followed by type, source, score, phase, attributes, source_strand, feature_id (decoded GFF3 ID, or missing) and a parent_ids list-column. Repeated IDs for discontinuous features and multiple parents are retained; selecting feature types may omit parents. A ggchord_import attribute records files, per-file output row counts and the one-based inclusive coordinate convention.

See Also

ggchord_data_contract

Examples

```
## Not run:
library(dplyr)
# Requires the source checkout or tutorial input bundle.
source <- read_gff3("examples/gff3/MT108731.1.gff3", unstranded = "keep")
features <- source |>
  select(accver, start, end, strand, anno)

## End(Not run)
```

---

read_paf *Read PAF mappings as ggchord links*

Description

Converts zero-based half-open PAF query/target intervals to one-based inclusive endpoints. Query endpoints increase; target endpoints decrease for relative strand "-". This display convention preserves relative orientation and does not infer a biological strand for either sequence.

Usage

```
read_paf(file = NULL, files = NULL, source_file = FALSE)
```

Arguments

file	Optional path vector or glob patterns; read before files.
files	Optional paths or wildcard patterns for multiple PAF files.
source_file	Logical. Add a .source_file column, default FALSE.

Details

length is the reported alignment block length, including gaps, and pident is 100 * nmatch / length using the reported PAF fields. Both counts may be approximate when the producer did not compute an alignment. The importer does not recompute an alignment, validate a CIGAR, or claim exact base identity. pident_basis records this ratio's provenance. Mapping quality 255 means unavailable: mapq becomes NA, while source_mapq retains the original value. Optional typed tags are retained verbatim in the tags list-column, including unknown tags and their order.

Value

A link data frame with canonical endpoints, qlen, slen, nmatch, length, pident, mapq, relative orientation, original zero-based coordinates, tags and .source_line. The ggchord_import attribute records input/output coordinate conventions and files.

See Also

https://github.com/lh3/miniasm/blob/master/PAF.md

ggchord_data_contract

Examples

```
## Not run:
library(dplyr)
# Requires the source checkout or tutorial input bundle.
links <- read_paf("examples/format-inputs/emales-pair.paf")
links |>
  select(qaccver, saccver, qstart, qend, sstart, send)

## End(Not run)
```

---

restriction_site_example

Restriction sites for the minimal single-genome fixture

Description

Restriction sites for the minimal single-genome fixture

Format

A data frame returned by find_restriction_sites().

Source

Generated from examples/single-genome/minimal.fasta by data-raw/generate_example_data.R.

ribbon_data_example *Example alignment data*

Description

A compact alignment fixture generated deterministically from the BLAST files under `examples/blastn`. Alignments shorter than 300 bases are omitted, and dense sequence pairs retain at most three spatially distributed representatives so the default chord remains readable. Sparse pairs are preserved unchanged.

Usage

```
ribbon_data_example
```

Format

A data frame containing standard alignment columns (qaccver, saccver, pident, etc.)

scale_feature_fill_plasmid *Plasmid-map feature presets*

Description

Plasmid-map feature presets

Usage

```
scale_feature_fill_plasmid(..., limits = NULL)

scale_feature_shape_plasmid(..., limits = NULL, guide = "none")
```

Arguments

`...`	Additional scale arguments.
`limits`	Optional feature-type order.
`guide`	Guide used by the shape preset.

Value

A feature fill or shape scale.

scale_feature_shape_manual *Feature shape scale*

Description

Maps feature categories to the geometry types understood by `geom_feature()`, including full/compact/promoter/primer arrows, markers, blocks, capped lines, chevrons, and lollipops. `capped_line` is an unfilled interval with short terminal caps. Shape values affect the actual feature geometry as well as its legend key.

Usage

```
scale_feature_shape_manual(
  ...,
  values,
  name = "Feature",
  limits = NULL,
  guide = "none"
)
```

Arguments

`...`	Arguments passed to `ggplot2::discrete_scale()`.
`values`	A named or unnamed character vector of feature geometry names.
`name`	Scale and guide title, default `"Feature"`.
`limits`	Optional category order. Named `values` use their name order by default so shape and fill guides can merge cleanly.
`guide`	Guide specification. The default is `"none"` because `geom_feature()` combines shape silhouettes into its fill legend when both aesthetics describe the same feature categories. Supply `guide_ggchord_legend()` for a separate shape guide.

Value

A ggplot2 discrete scale for the `feature_shape` aesthetic.

scale_gene_fill_manual *Gene, feature and region fill scales*

Description

Gene, feature and region fill scales

Usage

```
scale_gene_fill_manual(..., values)

scale_feature_fill_manual(..., values, limits = NULL)

scale_region_fill_manual(..., values)
```

Arguments

...	Arguments passed to ggplot2::scale_fill_manual().
values	A set of fill values to map data values to.
limits	Optional display order. Named feature values use their name order by default.

Value

A ggplot2 scale.

---

scale_link *Scales for point links*

Description

Point-link aesthetics are independent of sequence and ribbon aesthetics.

Usage

```
scale_link_colour_manual(..., values)

scale_link_color_manual(..., values)

scale_link_alpha_manual(..., values)

scale_link_linewidth_manual(..., values)

scale_link_linetype_manual(..., values)

scale_link_colour_gradientn(...)

scale_link_color_gradientn(...)

scale_link_alpha_continuous(...)

scale_link_linewidth_continuous(...)
```

Arguments

...	Arguments passed to the corresponding ggplot2 scale.
values	Manual aesthetic values.

Value

A ggplot2 scale.

---

scale_ribbon_alpha_continuous
Ribbon alpha, outline and linetype scales

Description

Ribbon alpha, outline and linetype scales

Usage

```
scale_ribbon_alpha_continuous(..., range = NULL)

scale_ribbon_alpha_manual(..., values)

scale_ribbon_colour_manual(..., values)

scale_ribbon_color_manual(..., values)

scale_ribbon_linetype_manual(..., values)
```

Arguments

...	Arguments passed to the corresponding ggplot2 scale.
range	Output alpha range for a continuous scale. NULL uses the corresponding ggplot2 scale's default range.
values	Manual aesthetic values.

Value

A ggplot2 scale.

scale_ribbon_fill_stepsn *Ribbon fill scales*

Description

Ribbon fill scales

Usage

```
scale_ribbon_fill_stepsn(
  ...,
  colours,
  values = NULL,
  colors,
  guide = ggplot2::guide_coloursteps(available_aes = "ribbon_fill")
)

scale_ribbon_fill_gradientn(
  ...,
  colours,
  values = NULL,
  colors,
  guide = ggplot2::guide_colourbar(available_aes = "ribbon_fill")
)

scale_ribbon_fill_manual(..., values)

scale_ribbon_fill_identity(...)
```

Arguments

`...`	Arguments passed to the corresponding ggplot2 scale.
`colours, colors`	Gradient colours.
`values`	Manual values, or positions for gradient colours where supported by the corresponding ggplot2 scale.
`guide`	A guide object; defaults accept the `ribbon_fill` aesthetic.

Value

A ggplot2 scale.

scale_seq_colour_manual

Sequence colour scales

Description

Sequence colour scales

Usage

```
scale_seq_colour_manual(..., values)

scale_seq_color_manual(..., values)
```

Arguments

`...`	Arguments passed to `ggplot2::scale_colour_manual()`.
`values`	A set of aesthetic values to map data values to.

Value

A ggplot2 scale.

scale_seq_position_continuous

Genomic sequence position scale

Description

Controls major/minor genomic position breaks and labels independently for every sequence. The default domain is each sequence's `[0, length]` range; linear source-coordinate axes use the retained original genomic interval. This scale configures sequence axes in all three coordinate modes. To add an independent bp reference bar in a linear display, use `geom_seq_scale()`.

Usage

```
scale_seq_position_continuous(
  name = ggplot2::waiver(),
  breaks = ggplot2::waiver(),
  minor_breaks = ggplot2::waiver(),
  labels = ggplot2::waiver(),
  limits = NULL,
  expand = ggplot2::waiver(),
  oob = scales::censor,
  transform = "identity",
```

```
  n.breaks = 6,
  coordinates = c("relative", "source"),
  sequences = NULL,
  ...
)
```

Arguments

`name`	Scale name; position guides are disabled by default.
`breaks`, `minor_breaks`, `labels`, `limits`, `expand`, `oob`, `transform`, `n.breaks`	Standard continuous-scale controls.
`coordinates`	In linear layouts, use relative locus coordinates or original source coordinates retained by `focus_ggchord_data()`. This affects axis breaks and labels, not the underlying genomic geometry. Breaks returned in transformed scale space are converted back to genomic positions in every coordinate mode. Explicit `breaks = NULL` and `minor_breaks = NULL` suppress the corresponding ticks.
`sequences`	Optional character vector selecting sequence axes by their displayed `accver`. `NULL` keeps all axes; `character()` hides every axis. Focused or segmented views use window IDs, not parent source accessions. Unknown or undisplayed identifiers produce an error. This controls axis lines, ticks and labels together without dropping sequence geometry.
`...`	Additional arguments passed to `ggplot2::continuous_scale()`.

Value

A ggplot2 continuous scale for the `seq_position` role.

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
ggchord(seq_data = seqs) +
  geom_seq() +
  coord_collinear() +
  scale_seq_position_continuous(breaks = seq(0, 100000, by = 20000))
```

---

scale_seq_ring_manual *Explicit sequence ring scale*

Description

Maps user-defined ring identifiers to positive sequence radii. Ring membership is supplied with `geom_seq(aes(seq_ring = ...))`; ggchord never guesses the number, order, or spacing of rings.

Usage

```
scale_seq_ring_manual(
  ...,
  values,
  name = "Ring",
  limits = NULL,
  guide = "none"
)
```

Arguments

...	Arguments passed to ggplot2::discrete_scale().
values	A named or unnamed numeric vector of positive radii.
name	Scale title. The guide is hidden by default because the scale controls geometry rather than a visible legend aesthetic.
limits	Optional ring order.
guide	Guide specification, default "none".

Value

A ggplot2 discrete scale for the seq_ring role.

---

scale_track_fill_gradientn

Set a track's fill or colour mapping

Description

Set a track's fill or colour mapping

Usage

```
scale_track_fill_gradientn(track, ..., colours, guide = NULL)

scale_track_fill_manual(track, ..., values)

scale_track_colour_gradientn(track, ..., colours, guide = NULL)

scale_track_colour_manual(track, ..., values)
```

Arguments

track	Track ID.
...	Standard ggplot2 scale arguments.
colours	Gradient colours.
guide	Optional standard guide for the fill/colour scale.
values	Manual category colours.

Value

A standard scale scoped to the named track's visual role.

Examples

```
scale_track_fill_gradientn("gc", colours = c("navy", "white", "orange"))
```

`scale_track_value_continuous` *Set a named track's numerical height scale*

Description

Uses an independent continuous scale for each track ID. The same ID shares its domain across sequences, windows and layers. The scale controls height, not colour. A numerical axis is drawn by default; `guide = "none"` hides it.

Usage

```
scale_track_value_continuous(
  track,
  name = track,
  limits = NULL,
  breaks = ggplot2::waiver(),
  labels = ggplot2::waiver(),
  transform = "identity",
  oob = scales::censor,
  unit = NULL,
  guide = c("axis", "none"),
  ...
)
```

Arguments

`track`	Track ID from `position_track()`.
`name`	Axis title, default the track ID.
`limits`, `breaks`, `labels`, `transform`, `oob`	Standard continuous-scale controls.
`unit`	Optional data-unit label retained in track exports.
`guide`	`"axis"` or `"none"` for the numerical track axis.
`...`	Additional ggplot2 continuous-scale arguments.

Value

A ggplot2 scale, scoped to this track.

Examples

```
scale_track_value_continuous("coverage", limits = c(0, 100), unit = "reads")
```

`segment_ggchord_data` *Prepare independent windows for a segmented linear display*

Description

Crops through `focus_ggchord_data()` while retaining one sequence per window. Pass `segments` to `coord_collinear()` to place those sequences in shared display rows with explicit gaps and local reversals. Coordinates in the returned tables remain window-local and are never concatenated.

Usage

```
segment_ggchord_data(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  windows,
  point_data = NULL,
  cross_gap = c("split", "drop"),
  max_links = 10000
)
```

Arguments

`seq_data`	Full sequence table with unique `accver` and integer lengths.
`ribbon_data`	Optional paired interval table.
`gene_data`	Optional gene or generic feature table, including segments.
`windows`	Table with `accver`, `start`, `end`, unique `locus_id`, optional `row_id` (defaults to source `accver`) and `orientation` (1 or -1, defaults to 1). Bounds must be ordered, integral and within the sequence.
`point_data`	Optional point links with `qaccver`, `saccver`, `qpos`, `spos`.
`cross_gap`	How to handle interval links clipped by selected windows.
`max_links`	Positive integer upper bound on candidate ribbon/point copies. Increase explicitly for larger, intentionally overlapping views.

Details

`cross_gap = "split"` retains only the parts of each ribbon's linear endpoint correspondence that lie in both selected windows. Omitted regions are never bridged by one ribbon polygon. `"drop"` retains only links fully contained in a query and subject window. This is a display projection, not realignment. Point links survive only when both endpoints are retained. Overlapping windows can deliberately duplicate a source row; provenance identifies each copy, and `max_links` bounds the candidate expansion before allocation. Order in the window table is display order within each row.

Value

A list containing focused `seq_data`, `ribbon_data`, `gene_data`, `point_data`, `segments` for the coordinate system, reversible source to window `transforms`, and a `report` with input row numbers for every output connection and omitted input rows. Each window retains its own source accession, origin, full length and truncation flags.

See Also

`ggchord_data_contract`

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
windows <- tibble(
  accver = rep(seqs$accver[1], 2),
  start = c(1, floor(seqs$length[1] * .65)),
  end = c(floor(seqs$length[1] * .4), seqs$length[1]),
  locus_id = c("left_window", "right_window"), row_id = seqs$accver[1]
)
segment_ggchord_data(slice_head(seqs, n = 1), windows = windows)
```

---

seq_data_example *Example sequence data*

Description

Sequence length data for ggchord demonstration

Usage

```
seq_data_example
```

Format

A data frame containing columns: accver, length

single_gene_example *Minimal single-genome gene fixture*

Description

Minimal single-genome gene fixture

Format

A data frame with `accver`, `start`, `end`, `strand`, and `anno`.

Source

`examples/single-genome/features.tsv`.

single_genome_example *Minimal single-genome sequence fixture*

Description

Minimal single-genome sequence fixture

Format

A one-row data frame with `accver` and `length`.

Source

`examples/single-genome/minimal.fasta`.

stat_ribbon_bundle *Bundle ribbons as a ggchord stat*

Description

A layer form of `bundle_ggchord_ribbons()` that computes bundles before the shared chord geometry and exposes `after_stat(bundle_n)`, `after_stat(bundle_weight)`, and `after_stat(density)`.

Usage

```
stat_ribbon_bundle(
  mapping = NULL,
  data = NULL,
  bins = 80L,
  min_bundle = 2L,
  weight = c("length", "count", "pident"),
  group_by = NULL,
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

`mapping`, `data`	Standard layer mapping and data arguments.
`bins`, `min_bundle`, `weight`, `group_by`	Passed to `bundle_ggchord_ribbons()`.
`position`, `show.legend`, `inherit.aes`	Standard ggplot2 layer arguments.
`...`	Ribbon geometry and appearance arguments passed to `geom_link_ribbon()`.

Value

A ggchord ribbon layer.

See Also

`ggchord_data_contract`

---

stat_ribbon_density *Compute local ribbon density as a ggchord stat*

Description

Keeps one output ribbon per input row while measuring density in normalized query/subject midpoint bins. The computed variables are the same as `stat_ribbon_bundle()`, so density can be mapped with `aes(ribbon_alpha = after_stat(density))`.

Usage

```
stat_ribbon_density(
  mapping = NULL,
  data = NULL,
  bins = 80L,
```

```
  weight = c("length", "count", "pident"),
  group_by = NULL,
  position = "identity",
  show.legend = TRUE,
  inherit.aes = FALSE,
  ...
)
```

Arguments

mapping, data	Standard layer mapping and data arguments.
bins	Positive integer number of normalized midpoint bins per sequence.
weight	Density weight: "length", "count", or "pident".
group_by	Optional columns that define independent density groups.
position, show.legend, inherit.aes	Standard ggplot2 layer arguments.
...	Ribbon geometry and appearance arguments passed to geom_link_ribbon().

Value

A ggchord ribbon layer.

See Also

ggchord_data_contract

---

theme_ggchord *ggchord themes*

Description

Complete themes for chord diagrams. Besides ordinary ggplot2 theme elements, these functions control automatic sequence axes, chord labels and the appearance and placement of guides belonging to each ggchord role. Palettes, breaks, labels and guide ordering remain scale/guide concerns.

Usage

```
theme_ggchord(
  axis = NULL,
  axis.line = NULL,
  axis.ticks = NULL,
  axis.minor.ticks = NULL,
  axis.text = NULL,
  axis.gap = NULL,
  axis.ticks.length = NULL,
  axis.minor.ticks.length = NULL,
```

```
  axis.text.offset = NULL,
  axis.text.orientation = NULL,
  axis.text.check.overlap = NULL,
  seq.label = NULL,
  gene.label = NULL,
  gene.label.segment = NULL,
  feature.label = NULL,
  feature.label.segment = NULL,
  restriction.label = NULL,
  restriction.label.segment = NULL,
  seq.center.label = NULL,
  seq.center.name = NULL,
  seq.center.length = NULL,
  legend.seq = NULL,
  legend.seq.position = NULL,
  legend.seq.direction = NULL,
  legend.seq.background = NULL,
  legend.seq.key = NULL,
  legend.seq.key.size = NULL,
  legend.seq.key.width = NULL,
  legend.seq.key.height = NULL,
  legend.seq.text = NULL,
  legend.seq.text.position = NULL,
  legend.seq.title = NULL,
  legend.seq.title.position = NULL,
  legend.seq.margin = NULL,
  legend.seq.spacing = NULL,
  legend.link = NULL,
  legend.link.position = NULL,
  legend.link.direction = NULL,
  legend.link.background = NULL,
  legend.link.key = NULL,
  legend.link.key.size = NULL,
  legend.link.key.width = NULL,
  legend.link.key.height = NULL,
  legend.link.text = NULL,
  legend.link.text.position = NULL,
  legend.link.title = NULL,
  legend.link.title.position = NULL,
  legend.link.margin = NULL,
  legend.link.spacing = NULL,
  legend.ribbon = NULL,
  legend.ribbon.position = NULL,
  legend.ribbon.direction = NULL,
  legend.ribbon.background = NULL,
  legend.ribbon.key = NULL,
  legend.ribbon.key.size = NULL,
  legend.ribbon.key.width = NULL,
```

```
  legend.ribbon.key.height = NULL,
  legend.ribbon.text = NULL,
  legend.ribbon.text.position = NULL,
  legend.ribbon.title = NULL,
  legend.ribbon.title.position = NULL,
  legend.ribbon.margin = NULL,
  legend.ribbon.spacing = NULL,
  legend.ribbon.ticks = NULL,
  legend.ribbon.ticks.length = NULL,
  legend.ribbon.axis.line = NULL,
  legend.gene = NULL,
  legend.gene.position = NULL,
  legend.gene.direction = NULL,
  legend.gene.background = NULL,
  legend.gene.key = NULL,
  legend.gene.key.size = NULL,
  legend.gene.key.width = NULL,
  legend.gene.key.height = NULL,
  legend.gene.text = NULL,
  legend.gene.text.position = NULL,
  legend.gene.title = NULL,
  legend.gene.title.position = NULL,
  legend.gene.margin = NULL,
  legend.gene.spacing = NULL,
  legend.feature = NULL,
  legend.feature.position = NULL,
  legend.feature.direction = NULL,
  legend.feature.background = NULL,
  legend.feature.key = NULL,
  legend.feature.key.size = NULL,
  legend.feature.key.width = NULL,
  legend.feature.key.height = NULL,
  legend.feature.text = NULL,
  legend.feature.text.position = NULL,
  legend.feature.title = NULL,
  legend.feature.title.position = NULL,
  legend.feature.margin = NULL,
  legend.feature.spacing = NULL,
  legend.region = NULL,
  legend.region.position = NULL,
  legend.region.direction = NULL,
  legend.region.background = NULL,
  legend.region.key = NULL,
  legend.region.key.size = NULL,
  legend.region.key.width = NULL,
  legend.region.key.height = NULL,
  legend.region.text = NULL,
  legend.region.text.position = NULL,
```

```
  legend.region.title = NULL,
  legend.region.title.position = NULL,
  legend.region.margin = NULL,
  legend.region.spacing = NULL,
  ...
)

theme_ggchord_minimal(
  axis = NULL,
  axis.line = NULL,
  axis.ticks = NULL,
  axis.minor.ticks = NULL,
  axis.text = NULL,
  axis.gap = NULL,
  axis.ticks.length = NULL,
  axis.minor.ticks.length = NULL,
  axis.text.offset = NULL,
  axis.text.orientation = NULL,
  axis.text.check.overlap = NULL,
  seq.label = NULL,
  gene.label = NULL,
  gene.label.segment = NULL,
  feature.label = NULL,
  feature.label.segment = NULL,
  restriction.label = NULL,
  restriction.label.segment = NULL,
  seq.center.label = NULL,
  seq.center.name = NULL,
  seq.center.length = NULL,
  legend.seq = NULL,
  legend.seq.position = NULL,
  legend.seq.direction = NULL,
  legend.seq.background = NULL,
  legend.seq.key = NULL,
  legend.seq.key.size = NULL,
  legend.seq.key.width = NULL,
  legend.seq.key.height = NULL,
  legend.seq.text = NULL,
  legend.seq.text.position = NULL,
  legend.seq.title = NULL,
  legend.seq.title.position = NULL,
  legend.seq.margin = NULL,
  legend.seq.spacing = NULL,
  legend.link = NULL,
  legend.link.position = NULL,
  legend.link.direction = NULL,
  legend.link.background = NULL,
  legend.link.key = NULL,
```

```
  legend.link.key.size = NULL,
  legend.link.key.width = NULL,
  legend.link.key.height = NULL,
  legend.link.text = NULL,
  legend.link.text.position = NULL,
  legend.link.title = NULL,
  legend.link.title.position = NULL,
  legend.link.margin = NULL,
  legend.link.spacing = NULL,
  legend.ribbon = NULL,
  legend.ribbon.position = NULL,
  legend.ribbon.direction = NULL,
  legend.ribbon.background = NULL,
  legend.ribbon.key = NULL,
  legend.ribbon.key.size = NULL,
  legend.ribbon.key.width = NULL,
  legend.ribbon.key.height = NULL,
  legend.ribbon.text = NULL,
  legend.ribbon.text.position = NULL,
  legend.ribbon.title = NULL,
  legend.ribbon.title.position = NULL,
  legend.ribbon.margin = NULL,
  legend.ribbon.spacing = NULL,
  legend.ribbon.ticks = NULL,
  legend.ribbon.ticks.length = NULL,
  legend.ribbon.axis.line = NULL,
  legend.gene = NULL,
  legend.gene.position = NULL,
  legend.gene.direction = NULL,
  legend.gene.background = NULL,
  legend.gene.key = NULL,
  legend.gene.key.size = NULL,
  legend.gene.key.width = NULL,
  legend.gene.key.height = NULL,
  legend.gene.text = NULL,
  legend.gene.text.position = NULL,
  legend.gene.title = NULL,
  legend.gene.title.position = NULL,
  legend.gene.margin = NULL,
  legend.gene.spacing = NULL,
  legend.feature = NULL,
  legend.feature.position = NULL,
  legend.feature.direction = NULL,
  legend.feature.background = NULL,
  legend.feature.key = NULL,
  legend.feature.key.size = NULL,
  legend.feature.key.width = NULL,
  legend.feature.key.height = NULL,
```

```
  legend.feature.text = NULL,
  legend.feature.text.position = NULL,
  legend.feature.title = NULL,
  legend.feature.title.position = NULL,
  legend.feature.margin = NULL,
  legend.feature.spacing = NULL,
  legend.region = NULL,
  legend.region.position = NULL,
  legend.region.direction = NULL,
  legend.region.background = NULL,
  legend.region.key = NULL,
  legend.region.key.size = NULL,
  legend.region.key.width = NULL,
  legend.region.key.height = NULL,
  legend.region.text = NULL,
  legend.region.text.position = NULL,
  legend.region.title = NULL,
  legend.region.title.position = NULL,
  legend.region.margin = NULL,
  legend.region.spacing = NULL,
  ...
)

theme_ggchord_dark(
  axis = NULL,
  axis.line = NULL,
  axis.ticks = NULL,
  axis.minor.ticks = NULL,
  axis.text = NULL,
  axis.gap = NULL,
  axis.ticks.length = NULL,
  axis.minor.ticks.length = NULL,
  axis.text.offset = NULL,
  axis.text.orientation = NULL,
  axis.text.check.overlap = NULL,
  seq.label = NULL,
  gene.label = NULL,
  gene.label.segment = NULL,
  feature.label = NULL,
  feature.label.segment = NULL,
  restriction.label = NULL,
  restriction.label.segment = NULL,
  seq.center.label = NULL,
  seq.center.name = NULL,
  seq.center.length = NULL,
  legend.seq = NULL,
  legend.seq.position = NULL,
  legend.seq.direction = NULL,
```

```
  legend.seq.background = NULL,
  legend.seq.key = NULL,
  legend.seq.key.size = NULL,
  legend.seq.key.width = NULL,
  legend.seq.key.height = NULL,
  legend.seq.text = NULL,
  legend.seq.text.position = NULL,
  legend.seq.title = NULL,
  legend.seq.title.position = NULL,
  legend.seq.margin = NULL,
  legend.seq.spacing = NULL,
  legend.link = NULL,
  legend.link.position = NULL,
  legend.link.direction = NULL,
  legend.link.background = NULL,
  legend.link.key = NULL,
  legend.link.key.size = NULL,
  legend.link.key.width = NULL,
  legend.link.key.height = NULL,
  legend.link.text = NULL,
  legend.link.text.position = NULL,
  legend.link.title = NULL,
  legend.link.title.position = NULL,
  legend.link.margin = NULL,
  legend.link.spacing = NULL,
  legend.ribbon = NULL,
  legend.ribbon.position = NULL,
  legend.ribbon.direction = NULL,
  legend.ribbon.background = NULL,
  legend.ribbon.key = NULL,
  legend.ribbon.key.size = NULL,
  legend.ribbon.key.width = NULL,
  legend.ribbon.key.height = NULL,
  legend.ribbon.text = NULL,
  legend.ribbon.text.position = NULL,
  legend.ribbon.title = NULL,
  legend.ribbon.title.position = NULL,
  legend.ribbon.margin = NULL,
  legend.ribbon.spacing = NULL,
  legend.ribbon.ticks = NULL,
  legend.ribbon.ticks.length = NULL,
  legend.ribbon.axis.line = NULL,
  legend.gene = NULL,
  legend.gene.position = NULL,
  legend.gene.direction = NULL,
  legend.gene.background = NULL,
  legend.gene.key = NULL,
  legend.gene.key.size = NULL,
```

```
  legend.gene.key.width = NULL,
  legend.gene.key.height = NULL,
  legend.gene.text = NULL,
  legend.gene.text.position = NULL,
  legend.gene.title = NULL,
  legend.gene.title.position = NULL,
  legend.gene.margin = NULL,
  legend.gene.spacing = NULL,
  legend.feature = NULL,
  legend.feature.position = NULL,
  legend.feature.direction = NULL,
  legend.feature.background = NULL,
  legend.feature.key = NULL,
  legend.feature.key.size = NULL,
  legend.feature.key.width = NULL,
  legend.feature.key.height = NULL,
  legend.feature.text = NULL,
  legend.feature.text.position = NULL,
  legend.feature.title = NULL,
  legend.feature.title.position = NULL,
  legend.feature.margin = NULL,
  legend.feature.spacing = NULL,
  legend.region = NULL,
  legend.region.position = NULL,
  legend.region.direction = NULL,
  legend.region.background = NULL,
  legend.region.key = NULL,
  legend.region.key.size = NULL,
  legend.region.key.width = NULL,
  legend.region.key.height = NULL,
  legend.region.text = NULL,
  legend.region.text.position = NULL,
  legend.region.title = NULL,
  legend.region.title.position = NULL,
  legend.region.margin = NULL,
  legend.region.spacing = NULL,
  ...
)

theme_ggchord_publication(
  axis = NULL,
  axis.line = NULL,
  axis.ticks = NULL,
  axis.minor.ticks = NULL,
  axis.text = NULL,
  axis.gap = NULL,
  axis.ticks.length = NULL,
  axis.minor.ticks.length = NULL,
```

```
  axis.text.offset = NULL,
  axis.text.orientation = NULL,
  axis.text.check.overlap = NULL,
  seq.label = NULL,
  gene.label = NULL,
  gene.label.segment = NULL,
  feature.label = NULL,
  feature.label.segment = NULL,
  restriction.label = NULL,
  restriction.label.segment = NULL,
  seq.center.label = NULL,
  seq.center.name = NULL,
  seq.center.length = NULL,
  legend.seq = NULL,
  legend.seq.position = NULL,
  legend.seq.direction = NULL,
  legend.seq.background = NULL,
  legend.seq.key = NULL,
  legend.seq.key.size = NULL,
  legend.seq.key.width = NULL,
  legend.seq.key.height = NULL,
  legend.seq.text = NULL,
  legend.seq.text.position = NULL,
  legend.seq.title = NULL,
  legend.seq.title.position = NULL,
  legend.seq.margin = NULL,
  legend.seq.spacing = NULL,
  legend.link = NULL,
  legend.link.position = NULL,
  legend.link.direction = NULL,
  legend.link.background = NULL,
  legend.link.key = NULL,
  legend.link.key.size = NULL,
  legend.link.key.width = NULL,
  legend.link.key.height = NULL,
  legend.link.text = NULL,
  legend.link.text.position = NULL,
  legend.link.title = NULL,
  legend.link.title.position = NULL,
  legend.link.margin = NULL,
  legend.link.spacing = NULL,
  legend.ribbon = NULL,
  legend.ribbon.position = NULL,
  legend.ribbon.direction = NULL,
  legend.ribbon.background = NULL,
  legend.ribbon.key = NULL,
  legend.ribbon.key.size = NULL,
  legend.ribbon.key.width = NULL,
```

```
  legend.ribbon.key.height = NULL,
  legend.ribbon.text = NULL,
  legend.ribbon.text.position = NULL,
  legend.ribbon.title = NULL,
  legend.ribbon.title.position = NULL,
  legend.ribbon.margin = NULL,
  legend.ribbon.spacing = NULL,
  legend.ribbon.ticks = NULL,
  legend.ribbon.ticks.length = NULL,
  legend.ribbon.axis.line = NULL,
  legend.gene = NULL,
  legend.gene.position = NULL,
  legend.gene.direction = NULL,
  legend.gene.background = NULL,
  legend.gene.key = NULL,
  legend.gene.key.size = NULL,
  legend.gene.key.width = NULL,
  legend.gene.key.height = NULL,
  legend.gene.text = NULL,
  legend.gene.text.position = NULL,
  legend.gene.title = NULL,
  legend.gene.title.position = NULL,
  legend.gene.margin = NULL,
  legend.gene.spacing = NULL,
  legend.feature = NULL,
  legend.feature.position = NULL,
  legend.feature.direction = NULL,
  legend.feature.background = NULL,
  legend.feature.key = NULL,
  legend.feature.key.size = NULL,
  legend.feature.key.width = NULL,
  legend.feature.key.height = NULL,
  legend.feature.text = NULL,
  legend.feature.text.position = NULL,
  legend.feature.title = NULL,
  legend.feature.title.position = NULL,
  legend.feature.margin = NULL,
  legend.feature.spacing = NULL,
  legend.region = NULL,
  legend.region.position = NULL,
  legend.region.direction = NULL,
  legend.region.background = NULL,
  legend.region.key = NULL,
  legend.region.key.size = NULL,
  legend.region.key.width = NULL,
  legend.region.key.height = NULL,
  legend.region.text = NULL,
  legend.region.text.position = NULL,
```

```
  legend.region.title = NULL,
  legend.region.title.position = NULL,
  legend.region.margin = NULL,
  legend.region.spacing = NULL,
  ...
)

theme_ggchord_plasmid(
  axis = NULL,
  axis.line = NULL,
  axis.ticks = NULL,
  axis.minor.ticks = NULL,
  axis.text = NULL,
  axis.gap = NULL,
  axis.ticks.length = NULL,
  axis.minor.ticks.length = NULL,
  axis.text.offset = NULL,
  axis.text.orientation = NULL,
  axis.text.check.overlap = NULL,
  seq.label = NULL,
  gene.label = NULL,
  gene.label.segment = NULL,
  feature.label = NULL,
  feature.label.segment = NULL,
  restriction.label = NULL,
  restriction.label.segment = NULL,
  seq.center.label = NULL,
  seq.center.name = NULL,
  seq.center.length = NULL,
  legend.seq = NULL,
  legend.seq.position = NULL,
  legend.seq.direction = NULL,
  legend.seq.background = NULL,
  legend.seq.key = NULL,
  legend.seq.key.size = NULL,
  legend.seq.key.width = NULL,
  legend.seq.key.height = NULL,
  legend.seq.text = NULL,
  legend.seq.text.position = NULL,
  legend.seq.title = NULL,
  legend.seq.title.position = NULL,
  legend.seq.margin = NULL,
  legend.seq.spacing = NULL,
  legend.link = NULL,
  legend.link.position = NULL,
  legend.link.direction = NULL,
  legend.link.background = NULL,
  legend.link.key = NULL,
```

```
  legend.link.key.size = NULL,
  legend.link.key.width = NULL,
  legend.link.key.height = NULL,
  legend.link.text = NULL,
  legend.link.text.position = NULL,
  legend.link.title = NULL,
  legend.link.title.position = NULL,
  legend.link.margin = NULL,
  legend.link.spacing = NULL,
  legend.ribbon = NULL,
  legend.ribbon.position = NULL,
  legend.ribbon.direction = NULL,
  legend.ribbon.background = NULL,
  legend.ribbon.key = NULL,
  legend.ribbon.key.size = NULL,
  legend.ribbon.key.width = NULL,
  legend.ribbon.key.height = NULL,
  legend.ribbon.text = NULL,
  legend.ribbon.text.position = NULL,
  legend.ribbon.title = NULL,
  legend.ribbon.title.position = NULL,
  legend.ribbon.margin = NULL,
  legend.ribbon.spacing = NULL,
  legend.ribbon.ticks = NULL,
  legend.ribbon.ticks.length = NULL,
  legend.ribbon.axis.line = NULL,
  legend.gene = NULL,
  legend.gene.position = NULL,
  legend.gene.direction = NULL,
  legend.gene.background = NULL,
  legend.gene.key = NULL,
  legend.gene.key.size = NULL,
  legend.gene.key.width = NULL,
  legend.gene.key.height = NULL,
  legend.gene.text = NULL,
  legend.gene.text.position = NULL,
  legend.gene.title = NULL,
  legend.gene.title.position = NULL,
  legend.gene.margin = NULL,
  legend.gene.spacing = NULL,
  legend.feature = NULL,
  legend.feature.position = NULL,
  legend.feature.direction = NULL,
  legend.feature.background = NULL,
  legend.feature.key = NULL,
  legend.feature.key.size = NULL,
  legend.feature.key.width = NULL,
  legend.feature.key.height = NULL,
```

```
  legend.feature.text = NULL,
  legend.feature.text.position = NULL,
  legend.feature.title = NULL,
  legend.feature.title.position = NULL,
  legend.feature.margin = NULL,
  legend.feature.spacing = NULL,
  legend.region = NULL,
  legend.region.position = NULL,
  legend.region.direction = NULL,
  legend.region.background = NULL,
  legend.region.key = NULL,
  legend.region.key.size = NULL,
  legend.region.key.width = NULL,
  legend.region.key.height = NULL,
  legend.region.text = NULL,
  legend.region.text.position = NULL,
  legend.region.title = NULL,
  legend.region.title.position = NULL,
  legend.region.margin = NULL,
  legend.region.spacing = NULL,
  ...
)
```

Arguments

`axis` NULL to show automatic sequence axes, or ggplot2::element_blank() to hide the complete axis decoration.

`axis.line`, `axis.ticks`, `axis.minor.ticks`
Axis line elements.

`axis.text` Axis text element.

`axis.gap`, `axis.ticks.length`, `axis.minor.ticks.length`, `axis.text.offset`
Scalar grid::unit() objects controlling physical axis spacing. A negative `axis.gap` moves circular ticks and labels inside the sequence ring; the plasmid preset uses this convention.

`axis.text.orientation`
One of `"horizontal"`, `"parallel"`, `"perpendicular"`, or a finite numeric angle.

`axis.text.check.overlap`
Whether overlapping axis labels are hidden.

`seq.label`, `gene.label`, `feature.label`, `restriction.label`, `seq.center.label`, `seq.center.name`, `seq.center.length`
Text elements for sequence, gene, feature, restriction-site and circular centre labels.

`gene.label.segment`, `feature.label.segment`, `restriction.label.segment`
Line elements for gene, feature and restriction-site leader lines.

legend.seq, legend.ribbon, legend.gene, legend.feature, legend.region
: NULL to show and inherit common settings, or ggplot2::element_blank() to hide every guide for that role.

legend.seq.position, legend.ribbon.position, legend.gene.position, legend.feature.position, legend.region.position
: Per-role guide positions.

legend.seq.direction, legend.ribbon.direction, legend.gene.direction, legend.feature.direction, legend.region.direction
: Per-role directions.

legend.seq.background, legend.ribbon.background, legend.gene.background, legend.feature.background, legend.region.background
: Per-role backgrounds.

legend.seq.key, legend.ribbon.key, legend.gene.key, legend.feature.key, legend.region.key
: Per-role key elements.

legend.seq.key.size, legend.seq.key.width, legend.seq.key.height, legend.ribbon.key.size, legend.ribbon.key.width, legend.ribbon.key.height, legend.gene.key.size, legend.gene.key.width, legend.gene.key.height, legend.feature.key.size, legend.feature.key.width, legend.feature.key.height, legend.region.key.size, legend.region.key.width, legend.region.key.height
: Per-role key dimensions.

legend.seq.text, legend.ribbon.text, legend.gene.text, legend.feature.text, legend.region.text
: Per-role text elements.

legend.seq.text.position, legend.ribbon.text.position, legend.gene.text.position, legend.feature.text.position, legend.region.text.position
: Per-role text positions.

legend.seq.title, legend.ribbon.title, legend.gene.title, legend.feature.title, legend.region.title
: Per-role title elements.

legend.seq.title.position, legend.ribbon.title.position, legend.gene.title.position, legend.feature.title.position, legend.region.title.position
: Per-role title positions.

legend.seq.margin, legend.ribbon.margin, legend.gene.margin, legend.feature.margin, legend.region.margin
: Per-role margins.

legend.seq.spacing, legend.ribbon.spacing, legend.gene.spacing, legend.feature.spacing, legend.region.spacing
: Per-role spacing.

`legend.link`, `legend.link.position`, `legend.link.direction`, `legend.link.background`, `legend.link.key`, `legend.link.key.size`, `legend.link.key.width`, `legend.link.key.height`, `legend.link.text`, `legend.link.text.position`, `legend.link.title`, `legend.link.title.position`, `legend.link.margin`, `legend.link.spacing`
: Point-link guide appearance; follows the same role-specific rules.

`legend.ribbon.ticks`, `legend.ribbon.axis.line`
: Ribbon colourbar lines.

`legend.ribbon.ticks.length`
: Ribbon colourbar tick length.

`...`
: Standard ggplot2 theme arguments, including common text, plot and legend settings. Common legend settings also supply role inheritance. Removed underscore-style arguments produce a migration error.

Value

A complete ggplot2 theme carrying ggchord appearance settings.

Appearance settings

Ordinary text, panel, title and shared legend settings follow ggplot2. Sequence-axis settings control automatic axis decoration, not genomic break selection; use `scale_seq_position_continuous()` for breaks and labels. Role text and leader elements style non-data annotation. Role legend settings inherit common legend appearance, while scale and guide objects continue to own palettes, breaks and ordering. Physical spacings require scalar absolute `grid::unit()` values; `axis.text.orientation` is scalar. The preset themes share this signature and provide appearance defaults.

Standard ggplot2 settings

Standard text, plot title/caption/tag, backgrounds, plot margins and common legend settings remain available through `...`, for example `theme_ggchord(text = element_text(size = 12))`. Only ggchord-specific settings appear as explicit formals for IDE argument completion. An IDE that additionally expands upstream dots may still offer upstream suggestions; this package controls its own function signature.

`validate_ggchord_data` *Validate ggchord input data before plotting*

Description

Performs structured validation of `seq_data`, `ribbon_data` and `gene_data` so that problems can be found, understood and fixed before plotting. The result is a `"ggchord_validation"` object with a `valid` flag, `errors` (severe problems that make the plot misleading), `warnings` (drawable but noteworthy issues), per-category counts, a data summary, the original row numbers of every problem, and a list of automatically fixable issues.

Usage

```
validate_ggchord_data(
  seq_data,
  ribbon_data = NULL,
  gene_data = NULL,
  strict = FALSE,
  check_coordinates = TRUE,
  check_duplicates = TRUE,
  check_self_links = TRUE
)
```

Arguments

seq_data	data.frame/tibble, required. Basic sequence information (columns accver, length).
ribbon_data	data.frame/tibble, optional. Alignment results (columns qaccver, saccver, qstart, qend, sstart, send).
gene_data	data.frame/tibble, optional. Gene annotation data (columns accver, start, end, strand; anno is optional).
strict	Logical. When TRUE, collect issues and then stop if any severe problem is found. When FALSE (default), return the full diagnostic report without stopping.
check_coordinates	Logical, default TRUE. Whether to check that ribbon/gene coordinates stay inside [1, sequence length].
check_duplicates	Logical, default TRUE. Whether to look for fully duplicated, near-duplicated and highly overlapping records. Reciprocal overlap requires the intersection to cover at least 90 percent of each interval, including descending ribbon endpoints. Reversed feature bounds may cross a circular origin and are excluded from linear overlap diagnostics; exact duplicates are still reported.
check_self_links	Logical, default TRUE. Whether to flag alignment rows where qaccver == saccver.

Value

A "ggchord_validation" object (a list) with at least:

valid Logical: TRUE when there are no severe errors.

errors data.frame of severe issues (table, category, row, column, message).

warnings data.frame of non-severe issues (same columns).

summary Per-category counts (table, category, severity, n).

data_summary Counts of sequences/ribbons/genes, unknown IDs, out-of-range rows, etc.

invalid_rows Named list of original row numbers per problem category.

cleanable data.frame of fixable issues with suggested actions.

print() and summary() methods are provided.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
data(ribbon_data_example)
links <- ribbon_data_example |>
  select(qaccver, saccver, qstart, qend, sstart, send)
data(gene_data_example)
genes <- gene_data_example |>
  transmute(accver, start, end, strand = source_strand)
result <- validate_ggchord_data(seqs, links, genes)
result$valid
```

---

validate_ggchord_hierarchy

Validate explicit genome and contig relationships

Description

Checks relational keys without deriving genome membership, changing sequence order or joining contigs into a continuous sequence. Original input tables and their metadata are returned unchanged. Sequence lengths and annotation coordinates can be checked separately with validate_ggchord_data().

Usage

```
validate_ggchord_hierarchy(seq_data, genomes, strict = FALSE)
```

Arguments

seq_data	A data frame containing accver and genome foreign keys.
genomes	A data frame containing a unique genome primary key.
strict	Logical. Stop if relational errors are present; warnings about genomes without contigs do not cause failure.

Details

A sequence belongs to exactly one genome. Sequence accver values must be globally unique, including when local contig names would be reused by two genomes. Supply distinct explicit identifiers before validation; this function never prefixes or rewrites them. Genome rows with no contigs are reported as warnings, and remain in the returned genome table.

Value

A list with `valid`, unchanged `seq_data` and `genomes`, an `issues` data frame (`table`, `row`, `column`, `category`, `severity`, `message`), and `contigs`, a genome-order table with `genome`, `n_contigs`, and `sequence_rows`/`accvers` list-columns for each valid unique genome key.

See Also

ggchord_data_contract

Examples

```
library(dplyr)
data(seq_data_example)
seqs <- seq_data_example |>
  select(accver, length)
contigs <- seqs |>
  mutate(genome = accver)
genomes <- tibble(genome = seqs$accver)
validate_ggchord_hierarchy(contigs, genomes)
```

---

view_ggchord *Preview a ggchord plot at its intended export size*

Description

Renders a plot with `ggsave()` into a temporary PNG or SVG and opens a small static preview page in the configured IDE viewer or web browser. This is an export-size preview, not an interactive plot conversion; the original plot and its standard `ggsave()` workflow remain unchanged. ggchord inputs are rendered on a private copy, preserving their layout cache. See ggchord_layout_contract for the distinct preview/export/diagnostic results. Automatic height fitting does not resize an overwide legend. If decorations exceed the requested width, increase `width` or use, for example, `guides(seq_colour = guide_ggchord_legend(nrow = 2))`.

Usage

```
view_ggchord(
  plot = ggplot2::last_plot(),
  width = 12.39,
  height = 9.71,
  units = c("in", "cm", "mm", "px"),
  device = c("png", "svg"),
  dpi = 144,
  bg = NULL,
  viewer = c("auto", "ide", "browser", "none")
)
```

Arguments

plot	A ggchord or ggplot object, default last_plot().
width	Positive output width. With no explicit dimensions, coord_chord() uses 12.39 inches while coord_circular() derives its width from the annotation envelope.
height	Positive output height. With no explicit dimensions, coord_chord() uses 9.71 inches while coord_circular() derives both dimensions. Use NULL to fit the sequence, labels and legends while preserving equal coordinate units. Explicit width/height values are always respected.
units	Output units: "in", "cm", "mm", or "px".
device	Preview device, "png" or "svg". SVG output requires the optional svglite package or a working base Cairo SVG device.
dpi	Positive raster resolution, default 144.
bg	Optional background colour passed to ggsave().
viewer	How to open the preview: "auto" prefers getOption("viewer") and falls back to a browser in interactive sessions; "ide", "browser", and "none" select an explicit destination. "none" is useful in scripts and tests.

Value

The normalized temporary image path, invisibly.

RStudio plot pane and export-size rendering

Label packing, guide sizing, and fitted limits use the active graphics device's physical dimensions. Printing p in a small or narrow RStudio Plots pane can therefore produce a substantially different layout from an export, and very small panes may not have enough room for all radial labels. Resize or Zoom the pane and print again for a larger direct preview. For size-sensitive chord plots, use view_ggchord() while composing, then save the final figure with ggsave() at an explicit width and height matching the intended output. Preview measurements restore the previously active graphics device.

Examples

```
library(dplyr)
data(seq_data_example)
p <- ggchord(seq_data_example) +
  geom_seq()
if (interactive()) view_ggchord(p)
```

Index

∗ **datasets**
gene_data_example, 31
ribbon_data_example, 105
seq_data_example, 115
∗ **layout results**
ggchord_layout_contract, 69
+.ggchord, 4

aes *(ggplot2-helpers)*, 71
after_scale *(ggplot2-helpers)*, 71
after_stat *(ggplot2-helpers)*, 71
annotate *(ggplot2-helpers)*, 71
arrow *(grid-helpers)*, 79
arrow(), *45*
as.data.frame.ggchord_validation, 4
as_ggchord_feature, 5, *69*
as_ggchord_feature(), *82*, *91*, *97*
as_ggchord_link, 6
as_ggchord_link(), *64*, *91*
as_ggchord_seq, 7
as_ggchord_seq(), *5, 6*, *64*, *91*, *101*

build_ggchord_cds, 8
build_ggchord_cds(), *7*, *95, 96*
bundle_ggchord_ribbons, 9
bundle_ggchord_ribbons(), *116, 117*

clean_ggchord_data, 11
coord_chord, 13
coord_chord(), *16*, *48*, *53*, *64*
coord_circular, 14
coord_circular(), *16*, *53*, *64*
coord_collinear, 15
coord_collinear(), *48*, *53*, *56*, *59*, *64*, *85*, *88*, *90*, *114*

deduplicate_ggchord_ribbons, 17
diagnose_ggchord, 18
diagnose_ggchord(), *21*, *69*, *71*

element_blank *(ggplot2-helpers)*, 71
element_line *(ggplot2-helpers)*, 71
element_rect *(ggplot2-helpers)*, 71
element_text *(ggplot2-helpers)*, 71
expansion *(ggplot2-helpers)*, 71
export_ggchord_layout, 20
export_ggchord_layout(), *48*, *63*, *66*, *69*, *71*

filter_ggchord_ribbons, 21
filter_restriction_sites, 23
filter_restriction_sites(), *28*
find_common_features, 24
find_common_features(), *88*
find_ggchord_loci, 25
find_ggchord_loci(), *92*
find_primer_bindings, 26
find_restriction_sites, 28
find_restriction_sites(), *23, 24*, *88*, *104*
focus_ggchord_data, 29
focus_ggchord_data(), *26*, *30*, *111*, *114*
focus_ggchord_track, 30
focus_ggchord_track(), *62*

gene_data_example, 31
geom_feature, 32, *39*
geom_feature(), *16*, *25*, *37*, *49, 50*, *66*, *97*, *103*, *106*
geom_feature_label, 34
geom_feature_label(), *37*, *66*
geom_feature_label_repel, 35
geom_feature_label_repel(), *25*
geom_feature_plasmid, 37
geom_gene, 39, *40*
geom_gene(), *33*, *38*, *66*
geom_gene_label, 40, *40*, *42*, *44*
geom_gene_label(), *44*, *66*
geom_gene_label_repel, *41*, 42, *42*
geom_link_line, 45
geom_link_ribbon, 46
geom_link_ribbon(), *117, 118*

geom_primer, 48
geom_primer_label_repel, 49
geom_primer_label_repel(), *48*
geom_restriction_site, 50
geom_seq, 52
geom_seq_break, 53
geom_seq_center_label, 55
geom_seq_label, 56
geom_seq_region, 57
geom_seq_scale, 59
geom_seq_scale(), *110*
geom_track_bar *(*geom_track_tile*)*, 60
geom_track_line *(*geom_track_tile*)*, 60
geom_track_tile, 60
get_chord_layout, 63
get_chord_layout(), *20*, *66*, *69*, *71*
ggchord, 64
ggchord(), *65*
ggchord_api_contract, 65, *65*
ggchord_data_contract, *5, 6*, *8–10*, *12*, *18*, *22, 23*, *25–30*, *52*, *65, 66*, 67, *67*, *70, 71*, *83*, *86, 87*, *92–97*, *99, 100*, *102–104*, *115*, *117, 118*, *134, 135*
ggchord_layout_contract, *19, 20*, *63*, *65*, *67*, 69, *135*
ggplot2-helpers, 71
ggplot2::continuous_scale(), *111*
ggplot2::discrete_scale(), *106*, *112*
ggplot2::element_blank(), *130, 131*
ggplot2::ggsave(), *18*, *71*
ggplot2::guide_colourbar(), *80*
ggplot2::guide_legend(), *80*
ggplot2::layer(), *60*
ggplot2::scale_colour_manual(), *110*
ggplot2::scale_fill_manual(), *107*
ggsave *(*ggplot2-helpers*)*, 71
ggtitle *(*ggplot2-helpers*)*, 71
grid-helpers, 79
grid::unit(), *130*
guide_ggchord_colorbar *(*guide_ggchord_legend*)*, 80
guide_ggchord_colourbar *(*guide_ggchord_legend*)*, 80
guide_ggchord_legend, 80
guide_ggchord_legend(), *106*
guides *(*ggplot2-helpers*)*, 71

labs *(*ggplot2-helpers*)*, 71
labs(), *64*

last_plot *(*ggplot2-helpers*)*, 71
link_ggchord_clusters, 82

map_ggchord_position, 84
margin *(*ggplot2-helpers*)*, 71
merge_ggchord_ribbons, 85

optimize_ggchord_layout, 87

plasmid_example_pBluescript_II_SK_plus *(*plasmid_example_pUC19*)*, 88
plasmid_example_pBR322 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pCAMBIA1300 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pcDNA3_1_plus *(*plasmid_example_pUC19*)*, 88
plasmid_example_pDONR221 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pEarleyGate_201 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pET_28a_plus *(*plasmid_example_pUC19*)*, 88
plasmid_example_pETDuet_1 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pSB1C3 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pSpCas9_BB_2A_GFP_PX458 *(*plasmid_example_pUC19*)*, 88
plasmid_example_pTRE_Tight_BI *(*plasmid_example_pUC19*)*, 88
plasmid_example_pTRIPZ *(*plasmid_example_pUC19*)*, 88
plasmid_example_pUC19, 88
plasmid_example_pUC19c *(*plasmid_example_pUC19*)*, 88
position_feature_stack, 88
position_feature_stack(), *33*, *39–41*, *44*
position_identity *(*ggplot2-helpers*)*, 71
position_plasmid, 89
position_plasmid(), *33*, *40*
position_strand, 90
position_strand(), *33*, *40*
position_track, 90
position_track(), *62*, *113*
PositionIdentity *(*ggplot2-helpers*)*, 71
prepare_ggchord_data, *69*, 91
prepare_ggchord_data(), *101*
primer_catalog, 93

print.ggchord_clean, 94
project_ggchord_features, 94
project_ggchord_features(), *7*, *99*
project_ggchord_links, *69*, 96
project_ggchord_links(), *7*, *99*

read_bed, 97
read_blast, 98
read_fasta_lengths, 99
read_genbank, *69*, 100
read_gff3, 102
read_gff3(), *8*
read_paf, 103
rel *(*ggplot2-helpers*)*, 71
restriction_site_example, 104
ribbon_data_example, 105

scale_feature_fill_manual *(*scale_gene_fill_manual*)*, 106
scale_feature_fill_plasmid, 105
scale_feature_shape_manual, 106
scale_feature_shape_plasmid *(*scale_feature_fill_plasmid*)*, 105
scale_gene_fill_manual, 106
scale_link, 107
scale_link_alpha_continuous *(*scale_link*)*, 107
scale_link_alpha_manual *(*scale_link*)*, 107
scale_link_color_gradientn *(*scale_link*)*, 107
scale_link_color_manual *(*scale_link*)*, 107
scale_link_colour_gradientn *(*scale_link*)*, 107
scale_link_colour_manual *(*scale_link*)*, 107
scale_link_linetype_manual *(*scale_link*)*, 107
scale_link_linewidth_continuous *(*scale_link*)*, 107
scale_link_linewidth_manual *(*scale_link*)*, 107
scale_region_fill_manual *(*scale_gene_fill_manual*)*, 106
scale_ribbon_alpha_continuous, 108
scale_ribbon_alpha_manual *(*scale_ribbon_alpha_continuous*)*, 108
scale_ribbon_color_manual *(*scale_ribbon_alpha_continuous*)*, 108
scale_ribbon_colour_manual *(*scale_ribbon_alpha_continuous*)*, 108
scale_ribbon_fill_gradientn *(*scale_ribbon_fill_stepsn*)*, 109
scale_ribbon_fill_identity *(*scale_ribbon_fill_stepsn*)*, 109
scale_ribbon_fill_manual *(*scale_ribbon_fill_stepsn*)*, 109
scale_ribbon_fill_stepsn, 109
scale_ribbon_linetype_manual *(*scale_ribbon_alpha_continuous*)*, 108
scale_seq_color_manual *(*scale_seq_colour_manual*)*, 110
scale_seq_colour_manual, 110
scale_seq_position_continuous, 110, *132*
scale_seq_position_continuous(), *60*, *66*
scale_seq_ring_manual, 111
scale_track_colour_gradientn *(*scale_track_fill_gradientn*)*, 112
scale_track_colour_manual *(*scale_track_fill_gradientn*)*, 112
scale_track_fill_gradientn, 112
scale_track_fill_manual *(*scale_track_fill_gradientn*)*, 112
scale_track_value_continuous, 113
segment_ggchord_data, 114
segment_ggchord_data(), *16*, *30*
seq_data_example, 115
single_gene_example, 116
single_genome_example, 116
stat_ribbon_bundle, 116
stat_ribbon_bundle(), *117*
stat_ribbon_density, 117

theme *(*ggplot2-helpers*)*, 71
theme(), *64*
theme_ggchord, 118
theme_ggchord_dark *(*theme_ggchord*)*, 118
theme_ggchord_minimal *(*theme_ggchord*)*, 118

theme_ggchord_plasmid *(*theme_ggchord*)*,
        118
theme_ggchord_publication
        *(*theme_ggchord*)*, 118

unit *(*grid-helpers*)*, 79
utils::read.table(), *99*

validate_ggchord_data, *64*, 132
validate_ggchord_data(), *134*
validate_ggchord_hierarchy, 134
view_ggchord, 135
view_ggchord(), *65*, *69*, *71*

waiver *(*ggplot2-helpers*)*, 71