Package 'rhinotypeR'

April 7, 2026

Type Package

Title Rhinovirus genotyping

Version 1.5.0

Date 2025-10-21

Description
``rhinotypeR" is designed to automate the comparison of sequence data against prototype strains, streamlining the genotype assignment process. By implementing predefined pairwise distance thresholds,
this package makes genotype assignment accessible to researchers and public health professionals. This tool enhances our epidemiological toolkit by enabling more efficient surveillance and analysis of
rhinoviruses (RVs) and other viral pathogens with complex genomic landscapes.
Additionally, ``rhinotypeR" supports comprehensive visualization and analysis of
single nucleotide polymorphisms (SNPs) and amino acid substitutions,
facilitating in-depth genetic and evolutionary studies.

License MIT + file LICENSE

Encoding UTF-8

Imports Biostrings, methods, MSA2dist, msa

LazyData false

biocViews Sequencing, Genetics, Phylogenetics, Visualization, MultipleSequenceAlignment, MultipleComparison

Depends R (>= 4.5.0)

Suggests knitr, rmarkdown, BiocManager, BiocStyle, testthat (>= 3.0.0)

VignetteBuilder knitr

Vignettes `Introduction to rhinotypeR` = ``vignettes/rhinotypeR.Rmd"

URL https://github.com/omicscodeathon/rhinotypeR

BugReports https://github.com/omicscodeathon/rhinotypeR/issues

RoxygenNote 7.3.3

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/rhinotypeR

git_branch devel

git_last_commit cf7ced9

git_last_commit_date 2025-11-24

Repository Bioconductor 3.23

Date/Publication 2026-04-06

Author Martha Luka [aut, cre] (ORCID: <https://orcid.org/0000-0001-6217-4426>),
Ruth Nanjala [aut],
Winfred Gatua [aut],
Wafaa M. Rashed [aut],
Olaitan Awe [aut]

Maintainer Martha Luka <marthaluka20@gmail.com>

Contents

`alignToRefs` *Align user sequences to packaged rhinovirus prototype references*

Description

Combines imported sequences with the prototype rhinovirus references shipped in **rhinotypeR**. Performs a multiple sequence alignment using ClustalW, ClustalOmega, or MUSCLE. The aligned result is returned as a `DNAStringSet` with equal-width sequences, suitable for downstream distance calculations (e.g., `pairwiseDistances`). If `trimToRef = TRUE`, the final alignment is cropped to the non-gap span of a chosen reference sequence.

Usage

```
alignToRefs(
  seqData,
  method = c("ClustalW", "ClustalOmega", "Muscle"),
  trimToRef = TRUE,
  refName = "JN855971.1_A107",
  ...
)
```

Arguments

seqData	A Biostrings::DNAStringSet containing the user sequences to be aligned.
method	Character; one of "ClustalW", "ClustalOmega", or "Muscle". Passed to msa::msa.
trimToRef	logical(1). If TRUE (default), crop the aligned DNAStringSet to the span of the reference sequence (useful when user sequences extend beyond the targeted VP4/2 region).
refName	character(1). Name of the reference sequence that defines the trimming span when trimToRef = TRUE. Defaults to "JN855971.1_A107" (a VP4/2 prototype used as a convenient anchor; it has no special biological significance). Provide a different accession to change the trimming anchor.
...	Additional arguments forwarded to msa::msa (e.g., gap penalties).

Details

Packaged prototypes are loaded from system.file("extdata","prototypes.fasta", package = "rhinotypeR"). User sequences are appended and jointly aligned with msa::msa. The returned DNAStringSet has equal width. When trimToRef = TRUE, the alignment is cropped to the range covered by non-gap positions in refName; internal gaps within that span are preserved. For global removal of gap-containing columns across all sequences, see deleteMissingDataSites().

If you prefer to align/curate outside R, use getPrototypeSeqs to copy the prototype references to disk and import your curated alignment later.

Value

A Biostrings::DNAStringSet of aligned sequences (user + prototypes), all with identical width; if trimToRef = TRUE, cropped to the chosen reference's non-gap span.

Notes / Caveats

- **Method selection:** msa::msa expects a single method string; this wrapper enforces that with match.arg().
- **Homology:** Ensure sequences are homologous and from the same genomic region; MSA quality degrades on mixed loci.
- **Result class:** The function returns a DNAStringSet (not a DNAMultipleAlignment).

See Also

getPrototypeSeqs, pairwiseDistances, assignTypes

Examples

```
seqs_path <- system.file("extdata", "test.fasta", package = "rhinotypeR")
seqs <- Biostrings::readDNAStringSet(seqs_path)

aln_trim <- alignToRefs(seqs, method = "ClustalW", trimToRef = TRUE)

# Keep full joint alignment (no trimming)
aln_full <- alignToRefs(seqs, method = "ClustalW", trimToRef = FALSE)

# Use a different prototype as trimming anchor
# aln_sel <- alignToRefs(seqs, method = "Muscle", trimToRef = TRUE,
#                        refName = "AF343653.1_B27")
```

`assignTypes` *Assign rhinovirus genotypes based on pairwise distances to prototype strains*

Description

Compares query rhinovirus sequences to prototype reference strains using pairwise genetic distances and assigns a genotype when the nearest distance is below a user-specified threshold. If the nearest distance exceeds the threshold, the sequence is returned as ‘"unassigned"‘ **but the actual nearest distance and prototype reference are still reported**. This enables users to inspect borderline cases and optionally apply relaxed thresholds (e.g., up to 0.11) to account for evolving viral diversity or intra-type variation.

Usage

```
assignTypes(
  fastaData,
  model = "IUPAC",
  deleteGapsGlobally = FALSE,
  threshold = 0.105
)
```

Arguments

`fastaData` A [Biostrings::DNAStringSet-class] containing an **aligned** VP4/2 nucleotide alignment. The alignment must include the rhinovirus prototype sequences (same accessions as shipped with the package). If your alignment does not already include prototypes, you have two options:

- *In R (no download):* call [alignToRefs()] to align your sequences jointly with the packaged prototypes, then pass the result here.
- *External tools:* use [getPrototypeSeqs()] to export the prototypes, combine with your sequences, align/curate externally, then import the curated alignment.

`model` Character string specifying the substitution model used by [pairwiseDistances()] for distance calculation. Defaults to '"IUPAC"'.

`deleteGapsGlobally` Logical. If 'TRUE', sites containing gaps ('-') are removed across all sequences before distance calculation. Default: 'FALSE'.

`threshold` Numeric. Distance threshold for genotype assignment. Defaults to '0.105' (approximately 10.5 McIntyre et al. (2013). Use '0.095' for species B if desired.

Details

- Prototype accessions are distributed with the package and loaded from 'inst/extdata/prototypes.rda' (object 'prototypes'). - The function checks that all prototypes are present in 'fastaData'. If not, it stops with guidance to use [alignToRefs()] or [getPrototypeSeqs()]. - Distances are computed via [pairwiseDistances()] on the provided alignment. - Above-threshold queries are labeled '"unassigned"' while still reporting the nearest prototype and its distance to support interpretation and threshold adjustment.

Value

A 'data.frame' with columns:

query Query sequence name.

assignedType Assigned rhinovirus type, or '"unassigned"'.

distance Nearest pairwise distance to any prototype (reported even when unassigned).

reference Prototype accession corresponding to the nearest distance.

References

McIntyre, C. L., et al. (2013). Proposals for the classification of human rhinovirus species A, B and C into genotypically assigned types. *Journal of General Virology*, 94(8), 1791–1806.

See Also

[alignToRefs()] for in-R alignment with packaged prototypes; [getPrototypeSeqs()] to export prototypes for external alignment; [pairwiseDistances()] for distance calculation.

Examples

```
if (interactive()) {

  # Load example alignment shipped with the package
  test <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
  fastaD <- Biostrings::readDNAStringSet(test)

  # Assign types
  # Note: The input must include the prototype sequences.
  # If your alignment does not already contain prototypes,
  # use alignToRefs() before calling assignTypes() OR
  # use getPrototypeSeqs() to download the prototype sequences,
      # combine with your new sequences and align before importing to R
```

```
  try({
   res <- assignTypes(fastaD, model = "IUPAC", deleteGapsGlobally = FALSE, threshold = 0.105)
    head(res)
  })
}
```

---

countSNPs *Count pairwise SNPs between aligned DNA sequences*

Description

Computes the absolute number of single-nucleotide differences (SNPs) between all pairs of sequences in an aligned [Biostrings::DNAStringSet-class] object. Optionally removes any alignment columns that contain gaps or ambiguous bases before counting.

Usage

```
countSNPs(fastaData, deleteGapsGlobally = FALSE)
```

Arguments

fastaData	A [Biostrings::DNAStringSet-class] of aligned DNA sequences. Sequences must have identical width.
deleteGapsGlobally	Logical. If 'TRUE', sites containing gaps ('-' or '.') or ambiguous bases ('"N"', '"?"') are removed across all sequences before SNP counting. Default: 'FALSE'.

Value

A square integer matrix of SNP counts with sequence names as row and column dimnames.

See Also

[pairwiseDistances()] for distances; [SNPeek()], [plotAA()] for viz.

Examples

```
if (interactive()) {
  aln <- Biostrings::DNAStringSet(c(Seq1="ATGC", Seq2="ATGT", Seq3="ATGA"))
  countSNPs(aln)
  countSNPs(aln, deleteGapsGlobally = FALSE)
}
```

`getPrototypeSeqs` *Download rhinovirus prototype strains (optional)*

Description

Copies the packaged rhinovirus prototype strains (‘`prototypes.fasta`’) from **rhinotypeR** into a user-specified directory as ‘`RVRefs.fasta`’. Downloading to local storage is **not required** to use the package.

Usage

```
getPrototypeSeqs(destinationFolder, overwrite = TRUE)
```

Arguments

`destinationFolder`	`character(1)`. Path to an existing directory where the prototype FASTA will be written. The folder must already exist.
`overwrite`	`logical(1)`. Whether to overwrite an existing ‘`RVRefs.fasta`’ in the destination directory. Defaults to `TRUE`.

Details

Users have two equivalent workflows:

1. **In-R workflow (no download):** use `alignToRefs` to align your sequences against the packaged reference prototypes directly in R, then run `assignTypes` on the resulting alignment.
2. **External-tools workflow (with download):** use `getPrototypeSeqs` to save ‘`RVRefs.fasta`’ locally, *combine it with your new sequences*, and perform alignment using your preferred external tool. You can then bring the aligned FASTA back into R for genotype assignment.

Internally, this function uses `system.file` to locate the packaged prototype file and `file.copy` to copy it into the specified directory. If you are following the in-R workflow, you can skip this function entirely and proceed with `alignToRefs` followed by `assignTypes`.

Value

Prints a message noting the destination directory.

Author(s)

Martha Luka, Ruth Nanjala, Wafaa Rashed, Winfred Gatua, Olaitan Awe

See Also

`assignTypes`, `alignToRefs`

Examples

```
if (interactive()) {
  # --- In-R workflow (no download) ---
  # aln <- alignToRefs(query_fasta = "my_new_sequences.fasta", method = "Muscle")
  # typesDF <- assignTypes(aln)

  # --- External-tools workflow (with download) ---
  dest_dir <- tempdir() # specify a destination directory
  getPrototypeSeqs(destinationFolder = dest_dir)  # writes RVRefs.fasta
  # Now combine RVRefs.fasta with your new sequences and align using
  # external tools (e.g., MAFFT, MUSCLE). Then read the aligned FASTA
  # back into R and proceed with assignTypes().
  list.files(dest_dir)
}
```

---

overallMeanDistance *Overall Mean Pairwise Distance*

Description

Computes the overall mean pairwise distance between all sequences in an alignment. Optionally deletes columns containing gaps across all sequences before computing the distances.

Usage

```
overallMeanDistance(
  fastaData,
  model = "IUPAC",
  deleteGapsGlobally = FALSE,
  ...
)
```

Arguments

`fastaData`	A DNAStringSet containing aligned nucleotide sequences.
`model`	Character string specifying the substitution model to use. Defaults to '"IUPAC"'. Other options are passed to `dist.dna`.
`deleteGapsGlobally`	Logical; if `TRUE`, columns with gaps in any sequence are removed before computing distances. Default is `FALSE`.
`...`	Additional arguments passed to `dnastring2dist`.

Details

The function computes all pairwise distances using `dnastring2dist`, extracts the distance matrix, and reports the mean of the off-diagonal elements.

Value

A single numeric value giving the overall mean pairwise distance across all sequence comparisons (excluding diagonal/self comparisons).

See Also

pairwiseDistances, assignTypes

Examples

```
# Load example alignment
test <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
fastaD <- Biostrings::readDNAStringSet(test)

# Compute mean distance (using default IUPAC model)
overallMeanDistance(fastaD)

# Compute mean distance with gaps deleted
overallMeanDistance(fastaD, deleteGapsGlobally = TRUE)
```

---

pairwiseDistances *Compute pairwise genetic distances between aligned DNA sequences*

Description

Calculates pairwise genetic distances between all sequences in an aligned [Biostrings::DNAStringSet-class] object using [MSA2dist::dnastring2dist()]. Supports global gap deletion or default pairwise gap masking.

Usage

```
pairwiseDistances(fastaData, model = "IUPAC", deleteGapsGlobally = FALSE, ...)
```

Arguments

fastaData	A [Biostrings::DNAStringSet-class] object containing aligned DNA sequences. Sequences must all be the same width (e.g., from [msa::msa()] or [alignToRefs()]).
model	A character string specifying the substitution model to use. Defaults to '"IUPAC"', which accounts for ambiguous bases. Other models are passed to [ape::dist.dna()] through [MSA2dist::dnastring2dist()].
deleteGapsGlobally	Logical. If 'TRUE', all columns containing at least one gap ('-') are removed prior to distance calculation (global gap deletion). If 'FALSE' (default), gaps are handled pairwise during distance calculation.
...	Additional arguments passed to [ape::dist.dna()] via [MSA2dist::dnastring2dist()].

Details

- If ‘deleteGapsGlobally = TRUE‘, all gapped sites are removed across the entire alignment using [deleteMissingDataSites()]. - Otherwise, gaps are masked on a pairwise basis by [MSA2dist::dnastring2dist()].

Value

A numeric matrix of pairwise genetic distances, with row and column names corresponding to the sequence names in ‘fastaData‘.

See Also

* [countSNPs()] for converting distances to SNP counts. * [deleteMissingDataSites()] for global gap trimming.

Examples

```
if (interactive()) {

  aln <- DNAStringSet(c(
    Seq1 = "ATGC",
    Seq2 = "ATGT",
    Seq3 = "ATGA"
  ))

  # Default: IUPAC model with pairwise gap masking
  pairwiseDistances(aln)

  # Jukes and Cantor model
  pairwiseDistances(aln, model = "JC69")
}
```

---

plotAA *Visualize amino acid differences against a reference (fast zoom & highlighting)*

Description

‘plotAA()‘ plots per–position amino acid differences between an aligned set of protein sequences and a chosen reference. It reuses the same cache and fast redraw machinery as [SNPeek()], with an amino acid color map grouping residues into biochemical classes:

* Positively charged (Arg, His, Lys) – red * Negatively charged (Asp, Glu) – blue * Polar uncharged (Ser, Thr, Asn, Gln) – green * Nonpolar / aromatic / special (all others) – gold * Other/unknown (X, gaps) – grey

By default (no ‘xlim‘/‘center‘+‘window‘), the function plots the full alignment span and all sequences. Users can zoom to regions of interest and optionally emphasize specific sequence IDs without hiding the others.

Usage

```
plotAA(
  aln_aa_set,
  ref_name = NULL,
  xlim = NULL,
  center = NULL,
  window = NULL,
  showLegend = FALSE,
  colorMapAA = c(R = "red2", H = "red2", K = "red2", D = "blue2", E = "blue2", S =
   "green3", T = "green3", N = "green3", Q = "green3", A = "gold", V = "gold", I =
  "gold", L = "gold", M = "gold", F = "gold", W = "gold", P = "gold", G = "gold", Y =
   "gold", C = "gold"),
  colorFallback = "grey50",
  highlight_seqs = NULL,
  highlight_bg = grDevices::adjustcolor("#FFE082", alpha.f = 0.5),
  lwd_normal = 1,
  lwd_highlight = 2,
  seg_half_height = 0.35,
  y_cex = 0.8,
  line_col = "grey20",
  show_only_highlighted = FALSE
)
```

Arguments

aln_aa_set	A ‘Biostrings::AAStringSet‘ of aligned amino acid sequences (all sequences must be the same width).
ref_name	Character scalar; the sequence name in ‘aln_aa_set‘ to use as reference. If ‘NULL‘, the last sequence is used.
xlim	Optional integer length-2 vector giving the protein window to display, e.g. ‘c(5, 20)‘. If omitted, the entire span is shown.
center, window	Optional integers; alternatively specify a zoom by center position and window width (e.g., ‘center = 150, window = 50‘). Ignored if ‘xlim‘ is provided.
showLegend	Logical; if ‘TRUE‘, draws a compact legend for the five amino acid classes. Default: ‘FALSE‘.
colorMapAA	Named character vector mapping amino acids to colors. The default assigns one color per residue class (see Details).
colorFallback	Color used for any symbol not present in ‘colorMapAA‘ (e.g., ‘X‘, ‘-‘). Default: ‘"grey50"‘.
highlight_seqs	Optional character vector of sequence IDs (row names) to visually emphasize in the plot (adds a translucent row band and thicker mismatch strokes). Sequences are still shown unless ‘show_only_highlighted = TRUE‘.
highlight_bg	Fill color for emphasized rows (semi-transparent recommended). Default: translucent amber.
lwd_normal, lwd_highlight	Line widths for mismatch segments in normal vs highlighted rows. Defaults: ‘1‘ and ‘2‘.

`seg_half_height`
: Vertical half-height of each mismatch segment (in row units). Default: ‘0.35‘.

`y_cex`
: Expansion factor for y-axis (sequence labels). Default: ‘0.8‘.

`line_col`
: Baseline axis/line color. Default: ‘"grey20"‘.

`show_only_highlighted`
: Logical; if ‘TRUE‘, display only the rows whose IDs are listed in ‘highlight_seqs‘. Default: ‘FALSE‘.

Details

Internally, ‘plotAA()‘ calls [compute_cache()] with an amino acid–specific color map, then passes the result to [plot_window()] for fast drawing. The returned cache can be reused for repeated zooms without recomputation.

When ‘showLegend = TRUE‘, the legend shows only five biochemical classes (positively charged, negatively charged, polar uncharged, nonpolar/aromatic, and other).

Value

Invisibly returns the precomputed cache (class ‘"SNPeekCache"‘) used for plotting. The primary output is a base R plot drawn to the active device.

See Also

[SNPeek()] for the nucleotide version; [compute_cache()], [plot_window()] for internals.

Other visualization: `SNPeek()`, `plotDistances()`, `plotFrequency()`, `plotTree()`

Examples

```
# Create a protein alignment
aa <- Biostrings::AAStringSet(c(
  Ref = "MTEYKLVVVGYKL",
  S1  = "MTEYKLVILVVVG",
  S2  = "MTEYKLVVV-LVV"
))

# 1) Full span, default reference (last sequence)
plotAA(aa)

# 2) Choose a reference and zoom
plotAA(aa, ref_name = "Ref", xlim = c(3, 8))

# 3) Highlight a sequence of interest
plotAA(aa, ref_name = "Ref", xlim = c(3, 8),
       highlight_seqs = "S1", showLegend = TRUE)
```

plotDistances *Plot Pairwise Distance Heatmap*

Description

Generates a heatmap from a pairwise distance matrix, typically produced by pairwiseDistances.

Usage

```
plotDistances(distancesMatrix)
```

Arguments

distancesMatrix	A numeric matrix of pairwise distances, usually produced by pairwiseDistances or related functions.

Details

The function uses the base R heatmap function to visualize distances. Row and column clustering are disabled to preserve the input ordering.

Colors are drawn from heat.colors(256). The scale is set to "none" so the distances are displayed directly, not normalized by row or column.

Value

Invisibly returns the object from stats::heatmap (a list with components such as rowInd and colInd). The primary output is the heatmap drawn to the active device.

See Also

pairwiseDistances, plotTree

Other visualization: SNPeek(), plotAA(), plotFrequency(), plotTree()

Examples

```
# Example using built-in dataset
test <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
fastaD <- Biostrings::readDNAStringSet(test)

# Compute pairwise distances
dist_mat <- pairwiseDistances(fastaD)

# Plot heatmap of distances
 plotDistances(dist_mat)
```

plotFrequency *Plot Frequency of Assigned Rhinovirus Types*

Description

Generates a barplot showing the frequency of assigned rhinovirus types. Optionally displays a legend indicating species classification (A, B, C, Other).

Usage

```
plotFrequency(
  assignedTypesDF,
  showLegend = FALSE,
  sort = FALSE,
  las = 2,
  cex_names = 0.8
)
```

Arguments

assignedTypesDF	A data frame, typically the output of assignTypes, containing at least the columns query and assignedType.
showLegend	Logical; if TRUE, adds a legend mapping species (A, B, C, Other) to colors. Default is FALSE.
sort	Logical; if TRUE, bars are sorted descending by count. Default FALSE.
las	Integer; axis label style for type names (defaults to 2 = perpendicular for readability).
cex_names	Numeric; character expansion for x-axis names. Default 0.8.

Details

Types are grouped into species via the first letter *of the type token* after stripping common prefixes like "RV-" or "RV". Any "unassigned" (any case) is labeled as species "Other".

Colors:

- A = blue
- B = red
- C = green
- Other = grey

Value

Invisibly returns a data frame with columns: assignedType, count, species.

See Also

assignTypes

Other visualization: SNPeek(), plotAA(), plotDistances(), plotTree()

Examples

```
test <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
fastaD <- Biostrings::readDNAStringSet(test)
assigned <- try(assignTypes(fastaD), silent = TRUE)
if (!inherits(assigned, "try-error")) {
  plotFrequency(assigned, showLegend = TRUE)
}
```

---

plotTree *Plot a Phylogenetic Tree from a Distance Matrix*

Description

Generates a simple phylogenetic tree/dendrogram from a pairwise distance matrix using hierarchical clustering (complete linkage by default).

Usage

```
plotTree(distance_matrix, ...)
```

Arguments

distance_matrix	A numeric, symmetric matrix of pairwise distances (zeros on the diagonal).
...	Additional arguments passed to plot.hclust.

Details

The function converts the matrix to a dist object, performs hclust with method = "complete", and plots the resulting dendrogram.

Value

Invisibly returns the hclust object.

See Also

pairwiseDistances, plotDistances

Other visualization: SNPeek(), plotAA(), plotDistances(), plotFrequency()

Examples

```
# Example using built-in dataset
test <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
fastaD <- Biostrings::readDNAStringSet(test)

# Compute pairwise distances
dist_mat <- pairwiseDistances(fastaD)

# Plot tree
plotTree(dist_mat, lwd = 2)
```

`rhinotypeR` *rhinotypeR: VP4/2-based genotyping and visualization for human rhinoviruses*

Description

The **rhinotypeR** package provides a toolkit for automating rhinovirus genotyping using the VP4/2 genomic region—streamlining alignment, genotype assignment, and visualization.

Main Functions

Core Analysis Functions::

- `assignTypes`: Assign sequences to rhinovirus genotypes based on distances to prototype strains. Returns query ID, assigned type, distance to nearest prototype, and reference used.
- `pairwiseDistances`: Calculates pairwise genetic distances between all sequences using specified evolutionary models (e.g., IUPAC).
- `overallMeanDistance`: Computes the overall mean genetic distance across all sequences as a measure of diversity.

Alignment and Data Preparation::

- `alignToRefs`: Aligns user sequences with packaged prototype references using ClustalW, ClustalOmega, or MUSCLE. Optionally crops the alignment to a reference span.
- `getPrototypeSeqs`: Exports packaged prototype sequences to a local directory for external alignment workflows.
- `deleteMissingDataSites`: Performs global gap deletion (removes any column containing gaps in any sequence).

Visualization Functions::

- `plotDistances`: Creates a heatmap of pairwise genetic distances.
- `plotTree`: Generates a simple phylogenetic tree from a distance matrix.
- `SNPeek`: Visualizes single nucleotide polymorphisms (SNPs) with color-coded nucleotides (A=green, T=red, C=blue, G=yellow).
- `plotAA`: Displays amino acid differences with zooming and highlighting.

Input expectations

- *Format:* FASTA; DNA for VP4/2 (AA only for `plotAA`).
- *Region:* VP4/2; sequences must be homologous.
- *Length/QC:* Recommended $\geq$ 350 bp (typical 420 bp) and trimmed to the VP4/2 span.

Workflows

Workflow A: External Alignment:

1. Export prototypes: `getPrototypeSeqs("~/Desktop")`
2. Combine prototypes with your sequences
3. Align externally (e.g., MAFFT https://mafft.cbrc.jp/alignment/server/)
4. Import alignment: `Biostrings::readDNAStringSet("alignment.fasta")`
5. Assign genotypes: `assignTypes(alignment)`

Workflow B: Align in R:

```
# Align sequences with packaged prototypes
aln <- alignToRefs(
  seqData   = rhinovirusVP4,
  method    = "ClustalW",
  trimToRef = TRUE,
  refName   = "JN855971.1_A107"
)

# Assign genotypes
res <- assignTypes(
  aln,
  model = "IUPAC",
  deleteGapsGlobally = FALSE,
  threshold = 0.105
)

# Visualize results
plotFrequency(res)
```

Key Parameters

- **threshold:** Default 0.105 (approximately 10.5%) for VP4/2 region, following proposals in McIntyre et al. (2013).
- **trimToRef:** When TRUE, crops alignment to non-gap span of chosen prototype (helpful when reads extend beyond target region).
- **deleteGapsGlobally:** Global deletion vs. pairwise deletion (distance models).
- **model:** Evolutionary model for distance calculation (e.g., "IUPAC").

Bundled datasets

- `rhinovirusPrototypesVP4`: VP4/2 prototype references as a `Biostrings::DNAStringSet` (mirrors 'inst/extdata/prototypes.fasta'). Reported by McIntyre et al. (2013), *Journal of General Virology*, 94(8), 1791–1806. doi:10.1099/vir.0.0536860
- `rhinovirusVP4`: A rhinovirus VP4/2 alignment as a `Biostrings::DNAStringSet` for examples/tests; derived from Luka et al. (2020), *Open Forum Infectious Diseases*, 7(10), ofaa385. doi:10.1093/ofid/ofaa385

Getting Started

```
library(rhinotypeR)
data(rhinovirusVP4, package = "rhinotypeR")

aln <- alignToRefs(
  seqData   = rhinovirusVP4,
  method    = "ClustalW",
  trimToRef = TRUE
)
res <- assignTypes(aln, model = "IUPAC")
head(res)

d <- pairwiseDistances(rhinovirusVP4, model = "IUPAC")
plotDistances(d)

sampled <- d[1:30, 1:30]
plotTree(sampled, main = "Rhinovirus VP4/2 Tree")

SNPeek(rhinovirusVP4)
aa <- Biostrings::translate(rhinovirusVP4)
plotAA(aa)
```

Troubleshooting

- **Prototypes missing error:** Ensure prototypes are present. Use `alignToRefs()` or `getPrototypeSeqs()`.
- **Misaligned regions:** Use `trimToRef = TRUE` and choose an appropriate `refName`.
- **Gap issues:** Consider global deletion with `deleteMissingDataSites()` or pairwise deletion via distance models.
- **Truncated plots:** Increase device height or use `show_only_highlighted = TRUE`.

Author(s)

Maintainer: Martha Luka <marthaluka20@gmail.com> (ORCID)

Authors:

- Ruth Nanjala
- Winfred Gatua
- Wafaa M. Rashed
- Olaitan Awe

See Also

alignToRefs, assignTypes, pairwiseDistances, overallMeanDistance, plotDistances, plotTree, SNPeek, plotAA, getPrototypeSeqs

---

`rhinovirusPrototypesVP4`

Rhinovirus VP4/2 prototype references (DNAStringSet)

Description

A FASTA bundle of rhinovirus prototype sequences used as references for genotyping (VP4/2 region). Provided as a convenience dataset for examples, vignettes, and tests.

Usage

```
data(rhinovirusPrototypesVP4)
```

Format

A [Biostrings::DNAStringSet-class] containing near-aligned VP4/2 prototype sequences. Sequence names are accessions (optionally with type labels).

Details

These sequences mirror the prototypes shipped in '`inst/extdata/prototypes.fasta`'. You may use this object directly in workflows (e.g., to append to user sequences before alignment) or export prototypes to disk with [getPrototypeSeqs()].

Source

McIntyre, C. L., Knowles, N. J., & Simmonds, P. (2013). Proposals for the classification of human rhinovirus species A, B and C into genotypically assigned types.

References

McIntyre, C. L., Knowles, N. J., & Simmonds, P. (2013). Proposals for the classification of human rhinovirus species A, B and C into genotypically assigned types. *Journal of General Virology*, 94(8), 1791–1806. doi:10.1099/vir.0.0536860

Examples

```
data(rhinovirusPrototypesVP4)
rhinovirusPrototypesVP4
```

`rhinovirusVP4` *Example VP4/2 alignment (DNAStringSet)*

Description

A VP4/2 alignment used in examples and tests. Useful for demonstrating ‘alignToRefs()‘, ‘pairwiseDistances()‘, and ‘assignTypes()‘.

Usage

```
data(rhinovirusVP4)
```

Format

A [Biostrings::DNAStringSet-class] where all sequences have equal width (aligned). Names are sequence IDs.

Details

This dataset is provided for illustration of the rhinotypeR workflow. For related cohort data and background context on respiratory viruses in a coastal Kenya school cohort, see the sources and references below.

Source

Luka, M. M., et al. (2020). Molecular Epidemiology of Human Rhinovirus From 1-Year Surveillance Within a School Setting in Rural Coastal Kenya.

References

Luka, M. M., et al. (2020). Molecular Epidemiology of Human Rhinovirus From 1-Year Surveillance Within a School Setting in Rural Coastal Kenya. *Open Forum Infectious Diseases*, 7(10), ofaa385. doi:10.1093/ofid/ofaa385

Adema, I. et al. (2020).Surveillance of respiratory viruses among children attending a primary school in rural coastal Kenya (Wellcome Open Research, 5:63, v2). doi:10.12688/wellcomeopenres.15703.2

Examples

```
data(rhinovirusVP4)
rhinovirusVP4
# d <- pairwiseDistances(rhinovirusVP4, model = "IUPAC")
```

SNPeek *Visualize SNP differences against a reference (fast zoom & highlighting)*

Description

'SNPeek()' renders per–position nucleotide differences between an aligned set of DNA sequences and a chosen reference using a cached pre-computation for fast redraws. With no 'xlim'/'center'+'window', it plots the full genome span and **all** sequences. Users can zoom regions and optionally emphasize specific sequence IDs with(out) hiding the others.

Usage

```
SNPeek(
  aln_set,
  ref_name = NULL,
  xlim = NULL,
  center = NULL,
  window = NULL,
  showLegend = FALSE,
  colorMapNT = c(A = "green3", T = "red2", C = "blue2", G = "gold"),
  colorFallback = "grey50",
  highlight_seqs = NULL,
  highlight_bg = grDevices::adjustcolor("#FFE082", alpha.f = 0.5),
  lwd_normal = 1,
  lwd_highlight = 2,
  seg_half_height = 0.35,
  y_cex = 0.8,
  line_col = "grey20",
  show_only_highlighted = FALSE
)
```

Arguments

aln_set	A ['Biostrings::DNAStringSet'] of aligned sequences (all sequences must have identical width).
ref_name	Character; the sequence name in 'aln_set' to use as reference. If 'NULL', the last sequence is used.
xlim	Optional integer length-2 vector giving the genomic window, e.g. 'c(100, 200)'. If omitted, the entire span is shown.
center, window	Optionally specify a zoom by center position and window width (e.g., 'center = 5012, window = 600'). Ignored if 'xlim' is provided.
showLegend	Logical; if 'TRUE', draws a legend for the nucleotide color map. Default: 'FALSE'.
colorMapNT	Named character vector mapping nucleotides to colors. Default: 'c(A = "green3", T = "red2", C = "blue2", G = "gold")'.

colorFallback Color for any non-ATCG symbol (e.g., ‘N‘, ‘-‘). Default: ‘"grey50"‘.

highlight_seqs Optional character vector of sequence IDs (row names) to emphasize (adds a translucent row band and thicker mismatch strokes). All sequences are still shown unless ‘show_only_highlighted = TRUE‘.

highlight_bg Fill color for emphasized rows (use ‘grDevices::adjustcolor()‘ for transparency). Default: ‘adjustcolor("#FFE082", alpha.f = 0.5)‘.

lwd_normal, lwd_highlight
Line widths for mismatch segments in normal vs highlighted rows. Defaults: ‘1‘ and ‘2‘.

seg_half_height
Vertical half-height of each mismatch segment (row units). Default: ‘0.35‘.

y_cex Expansion factor for y-axis (sequence labels). Default: ‘0.8‘.

line_col Baseline axis/line color. Default: ‘"grey20"‘.

show_only_highlighted
Logical; if ‘TRUE‘, display only rows whose IDs are in ‘highlight_seqs‘. Default: ‘FALSE‘.

Details

Internally, ‘SNPeek()‘ builds a reusable cache of mismatch positions and color indices versus the reference, then subsets to the requested window before plotting for speed. The cache is returned invisibly so you can reuse it for interactive workflows without recomputation.

Value

Invisibly returns the precomputed cache (class ‘"SNPeekCache"‘) used for plotting. The primary output is a base R plot.

See Also

‘plotAA()‘ for the amino-acid analogue.

Other visualization: plotAA(), plotDistances(), plotFrequency(), plotTree()

Examples

```
fasta_file <- system.file("extdata", "input_aln.fasta", package = "rhinotypeR")
aln <- Biostrings::readDNAStringSet(fasta_file)

# Full span
SNPeek(aln)

# Zoom + highlight
SNPeek(aln, xlim = c(100, 200),
       highlight_seqs = c("MT177780.1", "MT177798.1"),
       showLegend = TRUE)
```

SNPeek-internal *Internal helpers for SNPeek/plotAA*

Description

Utilities used by SNPeek/plotAA for caching and fast drawing. These functions are documented for transparency but are **not** part of the public API.

Given an aligned set of sequences (first element treated as the reference), find positions where each non-reference sequence differs and assign colors based on a user-supplied map.

Precomputes mismatch positions vs. a chosen reference and compact color indices for quick re-drawing of windows without rescanning the alignment.

Keeps only the specified row indices from an `SNPeekCache`. Useful for drawing views that focus on a subset (e.g., highlighted sequences).

Low-level renderer used by higher-level plotting helpers. Plots mismatch ticks relative to a cached reference over a specified genomic span, with optional highlighting and a legend.

Usage

```
compareAndColorSequences(sequences, colorMap, colorFallback = "gray")

compute_cache(aln_set, ref_name = NULL, colorMap, colorFallback = "grey")

subset_cache(cache, keep_idx)

plot_window(
  cache,
  xlim = NULL,
  center = NULL,
  window = NULL,
  showLegend = FALSE,
  highlight_seqs = NULL,
  highlight_bg = "#FFE08280",
  lwd_normal = 1,
  lwd_highlight = 2,
  seg_half_height = 0.35,
  y_cex = 0.8,
  line_col = "grey20",
  show_only_highlighted = FALSE
)
```

Arguments

`sequences`	Character vector (named preferred) of equal-length, aligned sequences. The first element is treated as the reference.
`colorMap`	Named vector mapping symbols (e.g., nucleotides) to colors. Used to render mismatches. For symbols not present, `colorFallback` is used.

colorFallback	Color used when a symbol is not in colorMap.
aln_set	Biostrings::DNAStringSet or Biostrings::AAStringSet containing an aligned, equal-width alignment.
ref_name	Optional character; the sequence name to use as the reference. Defaults to the last sequence in aln_set.
cache	An object from compute_cache (class "SNPeekCache").
keep_idx	Integer vector of 1-based row indices to retain. Values outside the valid range are ignored.
xlim	Integer length-2 vector giving the genomic range to display. If omitted, you can provide center and window for quick zoom.
center	Integer center position for quick zoom (used with window).
window	Integer window width for quick zoom (used with center).
showLegend	Logical; draw a small legend for the color map.
highlight_seqs	Character vector of sequence names to visually highlight (background shading and thicker ticks).
highlight_bg	Background color for highlighted rows (can include alpha).
lwd_normal	Numeric line width for non-highlighted ticks.
lwd_highlight	Numeric line width for highlighted ticks.
seg_half_height	Half-height of mismatch ticks (in plot y-units).
y_cex	Character expansion for y-axis (sequence name) labels.
line_col	Color for axes/frame elements.
show_only_highlighted	Logical; if TRUE and highlight_seqs are provided, only those rows are drawn.

Value

A list (length = length(sequences) - 1) of data frames with columns position (integer), subsType (character), and color (character), one per non-reference sequence.

An object of class "SNPeekCache" (a list) with elements:

- genome_len (integer)
- seq_names (character)
- ref_label (character)
- ref_row (integer; row index of the reference)
- diffs_list (list of per-sequence mismatch positions & color indices)
- col_levels (character; palette including fallback as last element)
- colorMap, colorFallback

A pruned "SNPeekCache" object containing only the selected sequences. The original metadata (e.g., ref_row) is preserved and is not used directly by the drawing code.

Invisibly returns a list with xlim, n_sequences, and highlighted sequence names.

See Also

compute_cache, plot_window

compute_cache

Examples

```
## Not run:
seqs <- c(Ref="ATGC", S1="ATGT", S2="ATAC")
cmap <- c(A="green", T="red", C="blue", G="gold")
compareAndColorSequences(seqs, cmap)

## End(Not run)

## Not run:
f <- system.file("extdata","test.fasta", package="rhinotypeR")
aln <- Biostrings::readDNAStringSet(f)
cache <- compute_cache(aln, ref_name = tail(names(aln), 1))

## End(Not run)

## Not run:
cache2 <- subset_cache(cache, c(1, 3, 5))

## End(Not run)

## Not run:
cache <- compute_cache(aln)
plot_window(cache, center = 200, window = 150,
            highlight_seqs = cache$seq_names[1:3], showLegend = TRUE)

## End(Not run)
```

Index