Package: drugfindR (via r-universe)

May 30, 2026

Title Investigate iLINCS for candidate repurposable drugs

Version 1.1.0

Description This package provides a convenient way to access the LINCS Signatures available in the iLINCS database. These signatures include Consensus Gene Knockdown Signatures, Gene Overexpression signatures and Chemical Perturbagen Signatures. It also provides a way to enter your own transcriptomic signatures and identify concordant and discordant signatures in the LINCS database.

License GPL-3 + file LICENSE

Encoding UTF-8

URL https://github.com/CogDisResLab/drugfindR, https://cogdisreslab.github.io/drugfindR/

BugReports https://github.com/CogDisResLab/drugfindR/issues

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

biocViews FunctionalPrediction, DifferentialExpression, GeneSetEnrichment, SingleCell, Network

Imports tibble, rlang, dplyr, purrr, readr, stringr, stats, lifecycle, S4Vectors, httr2, curl, DFplyr

Depends R (>= 4.5.0)

Suggests AnnotationDbi, BiocStyle, biocthis, codemetar, devtools, here, httptest2, jsonlite, knitr, rmarkdown, testthat (>= 3.0.0), tidyverse, usethis

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

X-schema.org-applicationCategory Genomics

X-schema.org-keywords LINCS, iLINCS, drug repurposing, drug discovery, transcriptomics, gene expression, gene knockdown, gene overexpression, chemical perturbagen, drugfindR

X-schema.org-isPartOf https://bioconductor.org

Collate 'utilities.R' 'consensusConcordants.R' 'drugfindR-package.R' 'filterSignature.R' 'getConcordants.R' 'getSignature.R' 'prepareSignature.R' 'investigateSignature.R' 'investigateTarget.R'

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev

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

Date/Publication 2026-04-28 13:06:11 UTC

RemoteUrl https://github.com/bioc/drugfindR

RemoteRef HEAD

RemoteSha 8b653fc88529ccd468e21a659224a289a93ee17a

Contents

`consensusConcordants` *Generate a Consensus list of Targets* **[Stable]**

Description

This function takes a list of (optionally split) concordance dataframes and returns a ranked list of gene or drug targets that have been chose for their maximal similarity to the signature

Usage

```
consensusConcordants(..., paired = FALSE, cutoff = 0.321, cellLine = NULL)
```

Arguments

`...`	One or Two (see paired) Data Frames with the concordants
`paired`	Logical indicating whether you split the dataframes by up and down regulated in prior analysis
`cutoff`	A similarity cutoff value. Defaults to 0.321
`cellLine`	A character vector of Cell Lines you are interested in.

Value

A tibble with the filtered and deduplicated results

Examples

```
# Create mock concordants data for demonstration
mockConcordants <- data.frame(
    signatureid = paste0("SIG", 1:10),
    treatment = c(
        "TP53", "TP53", "MYC", "MYC", "EGFR",
        "EGFR", "KRAS", "BRCA1", "BRCA1", "PIK3CA"
    ),
    cellline = c(
        "A375", "PC3", "A375", "MCF7", "A375",
        "PC3", "A375", "A375", "MCF7", "A375"
    ),
    time = rep("24H", 10),
    concentration = rep(NA, 10),
    sig_direction = rep("DOWN", 10),
    sig_type = rep("single", 10),
    similarity = c(
        0.85, 0.72, -0.68, -0.45, 0.55,
        0.38, 0.42, 0.51, 0.33, 0.29
    ),
    pValue = rep(0.001, 10)
)

# Example 1: Basic consensus with default cutoff
consensus <- consensusConcordants(mockConcordants)
nrow(consensus) # Targets with |similarity| >= 0.321

# Example 2: Consensus with higher cutoff
consensus_strict <- consensusConcordants(mockConcordants, cutoff = 0.5)
nrow(consensus_strict) # Fewer targets with higher threshold

# Example 3: Filter by cell line
consensus_A375 <- consensusConcordants(mockConcordants, cellLine = "A375")
unique(consensus_A375$CellLine) # Only A375


# Network-dependent examples using real iLINCS data
# Get the L1000 signature for LINCSKD_28
kdSignature <- getSignature("LINCSKD_28")

# Get concordant gene knockdown signatures
concordantSignatures <- getConcordants(kdSignature, ilincsLibrary = "KD")

# Get the consensus list with different parameters
consensus <- consensusConcordants(concordantSignatures, cutoff = 0.5)

# Paired analysis example
filteredUp <- filterSignature(kdSignature, direction = "up", threshold = 0.5)
```

```
filteredDown <- filterSignature(kdSignature, direction = "down", threshold = -0.5)
concordants_up <- getConcordants(filteredUp, ilincsLibrary = "KD")
concordants_down <- getConcordants(filteredDown, ilincsLibrary = "KD")
consensus <- consensusConcordants(concordants_up, concordants_down, paired = TRUE)
```

`filterSignature` *Filter the L1000 Signature* **[Stable]**

Description

This function filters the L1000 signature to a given threshold, identifying up-regulated, down-regulated, or both up- and down-regulated genes. The function supports both absolute threshold filtering and proportional filtering based on quantiles of the expression data.

Usage

```
filterSignature(signature, direction = "any", threshold = NULL, prop = NULL)
```

Arguments

`signature`	A data.frame, tibble, or DataFrame containing the L1000 signature. Must contain a column named "Value_LogDiffExp" with log fold-change values.
`direction`	Character string specifying the direction to filter. Must be one of "up" (up-regulated genes only), "down" (down-regulated genes only), or "any" (both up- and down-regulated genes). Defaults to "any".
`threshold`	Numeric value or vector specifying the log fold-change threshold(s). Can be: * A single positive value: Creates symmetric thresholds ($\pm threshold$) * A vector of two values: First value is the down-regulated threshold, second value is the up-regulated threshold Cannot be specified together with `prop`. One of `threshold` or `prop` must be provided.
`prop`	Numeric value between 0 and 1 specifying the proportion of genes to select from the top and bottom of the expression distribution. For example, `prop = 0.1` selects the top 10% most up-regulated and bottom 10% most down-regulated genes. Cannot be specified together with `threshold`.

Details

The filtering process follows these steps:

1. Input validation: Checks data frame structure and parameter consistency
2. Threshold calculation: Computes filtering thresholds based on either absolute values (`threshold`) or quantiles (`prop`)
3. Direction-based filtering: Applies the computed thresholds according to the specified direction

When using `threshold`:

- Single value: Genes with |logFC| >= threshold are retained
- Two values: Genes with logFC <= `threshold[1]` OR logFC >= `threshold[2]`

When using `prop`:

- Thresholds are calculated as quantiles of the expression distribution
- Down threshold = quantile(logFC, prop)
- Up threshold = quantile(logFC, 1 - prop)

Value

A tibble containing the filtered L1000 signature with the same structure as the input but containing only genes that meet the filtering criteria.

See Also

`\link{getSignature}` for retrieving L1000 signatures from iLINCS, `\link{prepareSignature}` for preparing custom signatures, `\link{getConcordants}` for finding concordant signatures

Examples

```
# Create a mock signature for demonstration
mockSignature <- data.frame(
    signatureID = rep("MOCK001", 20),
    Name_GeneSymbol = paste0("GENE", 1:20),
    ID_geneid = 1:20,
    Value_LogDiffExp = c(
        -3.5, -2.8, -2.1, -1.5, -1.2, -0.8, -0.5, -0.3,
        -0.1, 0.1, 0.3, 0.6, 0.9, 1.2, 1.6, 2.0, 2.4, 2.9, 3.3, 3.8
    )
)

# Example 1: Filter by symmetric absolute threshold
# Keeps genes with |logFC| >= 1.5
filteredSymmetric <- filterSignature(mockSignature, threshold = 1.5)
nrow(filteredSymmetric) # Should return 8 genes

# Example 2: Filter by asymmetric absolute thresholds
# Keeps genes with logFC <= -2.0 OR logFC >= 2.5
filteredAsymmetric <- filterSignature(mockSignature, threshold = c(-2.0, 2.5))
nrow(filteredAsymmetric) # Should return 5 genes

# Example 3: Filter by proportion (top and bottom 20%)
filteredProportion <- filterSignature(mockSignature, prop = 0.2)
nrow(filteredProportion) # Should return 8 genes (4 up + 4 down)

# Example 4: Filter only up-regulated genes by threshold
upRegulated <- filterSignature(mockSignature, direction = "up", threshold = 1.0)
all(upRegulated$Value_LogDiffExp >= 1.0) # Should be TRUE

# Example 5: Filter only down-regulated genes by threshold
downRegulated <- filterSignature(mockSignature, direction = "down", threshold = 1.0)
```

```
all(downRegulated$Value_LogDiffExp <= -1.0) # Should be TRUE

# Network-dependent examples using real iLINCS data
# Get the L1000 signature for LINCSKD_28
kdSignature <- getSignature("LINCSKD_28")

# Filter for top 5% most extreme genes
topExtreme <- filterSignature(kdSignature, prop = 0.05)

# Get top 20% most up-regulated genes
topUpregulated <- filterSignature(kdSignature, direction = "up", prop = 0.2)
```

`getConcordants` *Get concordant signatures from iLINCS database* **[Stable]**

Description

This function queries the iLINCS (Integrative Library of Integrated Network-based Cellular Signatures) database to find signatures that are concordant (similar) to a given input signature.

Usage

```
getConcordants(signature, ilincsLibrary = "CP")
```

Arguments

`signature`	A data.frame, tibble, or S4Vectors::DataFrame containing the signature data. Must conform to iLINCS signature structure with columns: * `signatureID`: Signature identifier * `ID_geneid`: Gene IDs * `Name_GeneSymbol`: Gene symbols * `Value_LogDiffExp`: Log fold-change values * `Significance_pvalue`: Statistical significance p-values Use `prepareSignature()` to ensure proper formatting.
`ilincsLibrary`	Character string specifying the iLINCS library to search. Must be one of: * `"CP"`: Chemical Perturbagen library (default) * `"KD"`: Knockdown library * `"OE"`: Overexpression library

Details

The function performs the following steps:

1. Validates input parameters
2. Creates a temporary file with signature data
3. Detects signature direction from expression values
4. Sends a multipart POST request to the iLINCS API
5. Processes the JSON response into a standardized tibble
6. Cleans up temporary files

The signature direction is determined as follows:

- `"Up"`: All expression values are greater than or equal to zero
- `"Down"`: All expression values are less than or equal to zero
- `"Any"`: Mixed positive and negative values

Value

A data structure containing concordant signatures. The return type matches the input signature type: * tibble for data.frame or tibble inputs * S4Vectors::DataFrame for DataFrame inputs

Contains the following columns: * `signatureid`: Unique signature identifier * `compound` or `treatment`: Drug/treatment name * `concentration`: Drug concentration (CP library only) * `time`: Treatment duration * `cellline`: Cell line used * `similarity`: Similarity score (rounded to 8 decimal places) * `pValue`: Statistical significance (rounded to 20 decimal places) * `sig_direction`: Signature direction ("Up", "Down", or "Any")

API Details

This function interfaces with the iLINCS web service API. The signature is uploaded as a tab-separated file and analyzed against the specified library. Results are returned as JSON and parsed into a tibble.

Error Handling

The function will stop execution with informative error messages for:

- Invalid signature data types (must be data.frame, tibble, or DataFrame)
- Invalid signature structure (missing required columns, wrong order, etc.)
- Missing values in signature data
- Unsupported iLINCS library names
- HTTP errors from the iLINCS API
- Invalid or empty API responses

References

iLINCS Portal: http://www.ilincs.org/

Pilarczyk et al. (2020). Connecting omics signatures and revealing biological mechanisms with iLINCS. Nature Communications, 11(1), 4058.

See Also

`[ prepareSignature() ]` for signature preparation, `[ filterSignature() ]` for signature filtering, `[ investigateSignature() ]` for signature investigation

Examples

```
# Input validation examples (no API calls)
# These demonstrate proper signature structure
mockSig <- data.frame(
    signatureID = rep("TEST", 3),
    ID_geneid = c("123", "456", "789"),
    Name_GeneSymbol = c("TP53", "MYC", "EGFR"),
    Value_LogDiffExp = c(1.5, -2.0, 0.8)
)

# Validate library parameter (should produce error)
tryCatch(
    getConcordants(mockSig, ilincsLibrary = "INVALID"),
    error = function(e) message("Expected error: invalid library")
)

# This example requires network access to the iLINCS API

# Load example differential expression data
dge_file <- system.file("extdata", "dCovid_diffexp.tsv",
    package = "drugfindR"
)
dge_data <- read.delim(dge_file)

# Prepare signature to ensure proper structure
signature <- prepareSignature(
    dge_data[1:50, ],
    geneColumn = "hgnc_symbol",
    logfcColumn = "logFC",
    pvalColumn = "PValue"
)

# Find concordant chemical perturbagens
cpConcordants <- getConcordants(signature, ilincsLibrary = "CP")
head(cpConcordants)

# Find concordant knockdown signatures
kdConcordants <- getConcordants(signature, ilincsLibrary = "KD")
head(kdConcordants)

# Find concordant overexpression signatures
oeConcordants <- getConcordants(signature, ilincsLibrary = "OE")
head(oeConcordants)

# Works with different data frame types
signatureDf <- as.data.frame(signature)
cpConcordantsDf <- getConcordants(signatureDf, "CP")

# Works with S4Vectors::DataFrame
signatureDataFrame <- S4Vectors::DataFrame(signature)
cpConcordantsDataFrame <- getConcordants(signatureDataFrame, "CP")
# Returns S4Vectors::DataFrame to match input type
```

getSignature *Get the L1000 Signature from iLINCS* **[Stable]**

Description

This function acts as the entrypoint to the iLINCS database. This takes in an ID and returns the signature after making a call to the iLINCS database. The function automatically detects whether the signature is an L1000 signature based on the signature ID and metadata tables, and retrieves all available genes for comprehensive signature analysis.

Usage

```
getSignature(sigId)
```

Arguments

sigId	character. The ilincs signature_id

Value

a tibble with the signature data containing the following columns: * `signatureID`: The signature identifier * `ID_geneid`: Gene IDs (Entrez) * `Name_GeneSymbol`: Gene symbols * `Value_LogDiffExp`: Log fold-change values * `Significance_pvalue`: Statistical significance p-values

Examples

```
# Input validation example (no API call)
# Demonstrates proper signature ID format validation
tryCatch(
    getSignature(""), # Empty string should error
    error = function(e) message("Expected error: empty signature ID")
)


# These examples require network access to the iLINCS API

# Get the L1000 signature for LINCSKD_28
kdSignature <- getSignature("LINCSKD_28")
head(kdSignature)

# Get an overexpression signature (L1000 status is automatically detected)
oeSignature <- getSignature("LINCSOE_1000")
head(oeSignature)

# Check the structure of retrieved signature
str(kdSignature)
```

`investigateSignature` *Investigate a given DGE dataset* **[Stable]**

Description

This function takes a DGE Data frame and then finds concordant signatures to that. This generates an L1000 signature from the DGE dataset and then uploads that signature to iLINCS to find the relevant concordant (or discordant) signatures

Usage

```
investigateSignature(
  expr,
  outputLib,
  filterThreshold = NULL,
  filterProp = NULL,
  similarityThreshold = 0.2,
  paired = TRUE,
  outputCellLines = NULL,
  geneColumn = "Symbol",
  logfcColumn = "logFC",
  pvalColumn = "PValue",
  sourceName = "Input",
  sourceCellLine = NA,
  sourceTime = NA,
  sourceConcentration = NA
)
```

Arguments

`expr`	A dataframe that has differential gene expression analysis
`outputLib`	The library to search
`filterThreshold`	The Filtering threshold.
`filterProp`	The Filtering proportion.
`similarityThreshold`	The Similarity Threshold
`paired`	Logical. Whether to query iLINCS separately for up and down regulated genes
`outputCellLines`	A character vector of cell lines to restrict the output search to.
`geneColumn`	The name of the column that has gene symbols
`logfcColumn`	The name of the column that has log_2 fold-change values
`pvalColumn`	The name of the column that has p-values
`sourceName`	(Optional) An annotation column to identify the signature by name

`sourceCellLine` (Optional) An annotation column to specify the cell line for the input data

`sourceTime` (Optional) An annotation column to specify the time for the input data

`sourceConcentration` (Optional) An annotation column to specify the concentration for the input data

Value

A tibble with the the similarity scores and signature metadata

Examples

```
# Input validation example (no API calls)
mockExpr <- data.frame(
    Symbol = c("TP53", "MYC"),
    logFC = c(2.5, -1.8),
    PValue = c(0.001, 0.01)
)

# Validate library parameter (should produce error)
tryCatch(
    investigateSignature(mockExpr, outputLib = "INVALID"),
    error = function(e) message("Expected error: invalid library")
)


# This function makes multiple API calls to iLINCS and may take several minutes

# Load differential expression data
inputSignature <- read.table(
    system.file("extdata", "dCovid_diffexp.tsv", package = "drugfindR"),
    header = TRUE
)

# Investigate the signature against chemical perturbagen library
investigatedSignature <- investigateSignature(
    inputSignature,
    outputLib = "CP",
    filterThreshold = 0.5,
    geneColumn = "hgnc_symbol",
    logfcColumn = "logFC",
    pvalColumn = "PValue"
)
head(investigatedSignature)
```

`investigateTarget` *Investigate concordant signatures for a gene or drug* **[Stable]**

Description

Given the name of a target (gene knockdown/overexpression or compound) this high–level convenience wrapper:

Usage

```
investigateTarget(
  target,
  inputLib,
  outputLib,
  filterThreshold = 0.85,
  similarityThreshold = 0.321,
  paired = TRUE,
  inputCellLines = NULL,
  outputCellLines = NULL
)
```

Arguments

`target`	Character scalar. Gene symbol (for KD/OE libraries), or drug / compound name (for CP library) used to locate source signatures.
`inputLib`	Character ("OE", "KD", or "CP"). Library from which source signatures for `target` are drawn.
`outputLib`	Character ("OE", "KD", or "CP"). Library queried for concordant signatures.
`filterThreshold`	Numeric in \(0,1]. Minimum absolute (or directional) change used to retain genes in each source signature prior to concordance. Default `0.85` is conservative; consider lowering (e.g. `0.5`) for broader coverage.
`similarityThreshold`	Numeric in [0,1]. Minimum similarity score retained in the final consensus result set. Default `0.321` (~ upper third).
`paired`	Logical. If `TRUE` (default) computes concordance separately for up and down regulated gene sets; if `FALSE` uses all selected genes together.
`inputCellLines`	Optional character vector restricting the search for source signatures to specified cell line(s). If `NULL` all available are considered.
`outputCellLines`	Optional character vector restricting target signatures (during consensus formation) to specified cell line(s). If `NULL` all are considered.

Details

1. Locates iLINCS source signatures for the target in the specified input library.
2. Optionally filters by source cell line(s).
3. Retrieves each source signature and filters genes by direction and magnitude.
4. Queries iLINCS for concordant signatures in the chosen output library.
5. Computes paired or unpaired consensus concordance across up/down regulated sets.

6. Returns an augmented tibble of similarity scores and rich source/target metadata.

The paired workflow evaluates concordance separately for up- and down-regulated genes and then combines (via `consensusConcordants()`) the two result sets. When `paired = FALSE` a single aggregate signature (direction = "any") is used.

Network access: This function performs remote API calls (unless tests are run under a mocking context such as `httptest2::with_mock_api()`). Examples are wrapped in `\donttest{}` to avoid false negatives on CRAN / Bioconductor builders without network access.

Errors are raised if:

- No source signatures match `target` in the requested `inputLib` (empty set).
- Invalid library codes are supplied.

Internally this function orchestrates: `getSignature()`, `filterSignature()`, `getConcordants()` and `consensusConcordants()`. It returns a vertically concatenated result across all matching source signatures.

Value

A tibble (data frame) with one row per consensus concordant target signature. Typical columns include:

- `Source` / `Target` – gene or compound names.
- `Similarity` – numeric concordance score in [-1,1].
- `SourceSignature`, `TargetSignature` – iLINCS signature identifiers.
- `SourceCellLine`, `TargetCellLine` – originating cell lines (if applicable).
- `SourceConcentration`, `TargetConcentration` – dosing information for CP.
- `SourceTime`, `TargetTime` – time point metadata.

Thresholds

- `filterThreshold` controls gene selection within each source signature. It is passed to `filterSignature()` as the absolute (or directional) threshold.
- `similarityThreshold` is applied when forming the consensus concordants to discard low similarity entries.

See Also

`getSignature()`, `filterSignature()`, `getConcordants()`, `consensusConcordants()`, `prepareSignature()` for lower-level operations.

Examples

```
# Input validation examples (no API calls)
# Demonstrate library parameter validation
tryCatch(
    investigateTarget(target = "TP53", inputLib = "INVALID", outputLib = "CP"),
    error = function(e) message("Expected error: invalid inputLib")
)
```

```
tryCatch(
    investigateTarget(target = "TP53", inputLib = "KD", outputLib = "INVALID"),
    error = function(e) message("Expected error: invalid outputLib")
)


# This function makes multiple API calls to iLINCS and may take several minutes
# Basic paired investigation of a knockdown signature against compound library
set.seed(1)
res <- investigateTarget(
    target = "AATK",
    inputLib = "KD",
    outputLib = "CP",
    filterThreshold = 0.5,
    similarityThreshold = 0.3,
    paired = TRUE
)
head(res)

# Unpaired (aggregate) workflow — often faster, returns a single consensus set
res_unpaired <- investigateTarget(
    target = "AATK", inputLib = "KD", outputLib = "CP",
    filterThreshold = 0.5, similarityThreshold = 0.3, paired = FALSE
)
head(res_unpaired)

# Restrict source signatures to specific cell lines (if available)
# and target signatures to a subset of cell lines during consensus
res_filtered <- investigateTarget(
    target = "AATK", inputLib = "KD", outputLib = "CP",
    outputCellLines = c("MCF7"),
    filterThreshold = 0.5, similarityThreshold = 0.3
)
head(res_filtered)

# Using httptest2 (if installed) to mock network calls:
# httptest2::with_mock_api({
#   mock_res <- investigateTarget("AATK", "KD", "CP", filterThreshold = 0.5)
#   print(head(mock_res))
# })
```

`prepareSignature` *Prepare an L1000 Signature from a given differential gene expression output* **[Stable]**

Description

This function takes a differential gene expression output from any pipeline like edgeR or DeSeq2 or any that give you the gene symbol, log_2 fold-change and p-value and transforms that into an

L1000 signature for later processing.

Usage

```
prepareSignature(
  dge,
  geneColumn = "Symbol",
  logfcColumn = "logFC",
  pvalColumn = "PValue"
)
```

Arguments

dge	A dataframe-like object that has the differential gene expression information
geneColumn	The name of the column that has gene symbols
logfcColumn	The name of the column that has log_2 fold-change values
pvalColumn	The name of the column that has p-values

Value

A tibble with the L1000 signature.

Examples

```
# Load example differential expression data from package
dge_file <- system.file("extdata", "dCovid_diffexp.tsv",
    package = "drugfindR"
)
dge_data <- read.delim(dge_file)

# Prepare signature with p-values (standard workflow)
signature <- prepareSignature(
    dge_data,
    geneColumn = "hgnc_symbol",
    logfcColumn = "logFC",
    pvalColumn = "PValue"
)
head(signature)

# Prepare signature without p-values
signature_no_pval <- prepareSignature(
    dge_data,
    geneColumn = "hgnc_symbol",
    logfcColumn = "logFC",
    pvalColumn = NA
)
head(signature_no_pval)

# Custom column names example
custom_dge <- data.frame(
    Gene = c("TP53", "MYC", "BRCA1", "EGFR"),
```

```
    FC = c(2.5, -1.8, 3.2, -2.1),
    Pval = c(0.001, 0.01, 0.0001, 0.005)
)

custom_signature <- prepareSignature(
    custom_dge,
    geneColumn = "Gene",
    logfcColumn = "FC",
    pvalColumn = "Pval"
)
print(custom_signature)
```

Index