Package: ORscraper (via r-universe)

October 6, 2026

Type Package

Title Extract Information from Clinical Reports from 'Oncomine Reporter' and NCBI 'ClinVar'

Version 0.1.1

Description Clinical reports generated by 'Oncomine Reporter' software contain critical data in unstructured PDF format, making manual extraction time-consuming and error-prone. 'ORscraper' provides a coherent suite of functions to automate this process, allowing researchers to parse reports, identify key biomarkers, extract genetic variant tables, and filter results. It also integrates with the NCBI 'ClinVar' API <https://www.ncbi.nlm.nih.gov/clinvar/> to enrich extracted data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

SystemRequirements poppler-cpp (>= 0.73)

Imports pdftools, stringr, readxl, rentrez

Suggests testthat (>= 3.0.0), rmarkdown, knitr, mockery, spelling

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/SamuelGonzalez0204/ORscraper

BugReports https://github.com/SamuelGonzalez0204/ORscraper/issues

Language en-US

Config/pak/sysreqs libicu-dev libjpeg-dev libxml2-dev libssl-dev libpoppler-cpp-dev poppler-data

Repository https://samuelgonzalez0204.r-universe.dev

Date/Publication 2026-03-10 20:37:03 UTC

RemoteUrl https://github.com/samuelgonzalez0204/orscraper

RemoteRef HEAD

RemoteSha 30ef489dbaa16a2884dd1231c8a43beb41bb8595

Contents

classify_biopsy . . . 2
extract_chip_id . . . 3
extract_fusions . . . 3
extract_intermediate_values . . . 4
extract_values_from_tables . . . 5
extract_values_start_end . . . 6
extract_variable . . . 7
filter_pathogenic_only . . . 7
narrow_text . . . 8
read_pdf_content . . . 9
read_pdf_files . . . 9
search_ncbi_clinvar . . . 10
search_value . . . 11

Index **12**

`classify_biopsy` *Determine the type of biopsy from identifiers*

Description

This function analyzes biopsy identifiers and categorizes them into specific types based on a defined rule.

Usage

```
classify_biopsy(biopsy_numbers)
```

Arguments

`biopsy_numbers`	Character vector. Identifiers of biopsies to classify.

Value

A character vector representing the type of Sample type: 1, biopsy 2, aspiration 3, cytology

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file

NB_values <- c()
NB_values <- extract_intermediate_values(NB_values, lines, "biopsia:")

biopsies_identifiers <- classify_biopsy(NB_values)
```

extract_chip_id *Extract numeric identifiers from file names*

Description

This function retrieves chip values from file names matching a specific pattern.

Usage

```
extract_chip_id(files)
```

Arguments

files	Character vector. File names to process.

Value

A character vector of chip identifiers extracted from the file names.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)

chips <- extract_chip_id(files)
```

extract_fusions *Extract fusion variants from text*

Description

This function identifies and extracts fusion variants from text lines based on specific patterns.

Usage

```
extract_fusions(lines, mutations)
```

Arguments

lines	Character vector. Lines of text to search for fusion variants.
mutations	Character vector. List of mutations to look for.

Value

A list of fusion variants identified in the text.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file

genes_file <- system.file("extdata/Genes.xlsx", package = "ORscraper")
genes <- readxl::read_excel(genes_file)
mutations <- unique(genes$GEN)

fusions <- extract_fusions(lines, mutations)
```

`extract_intermediate_values` *Extract intermediate values from text lines*

Description

This function retrieves unique matches for a search pattern within text lines.

Usage

```
extract_intermediate_values(list_input, lines, search_text)
```

Arguments

`list_input`	List. The list to append extracted values to.
`lines`	Character vector. The text lines to search within.
`search_text`	Character. The pattern to search for.

Value

An updated list with appended values.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file
NHC_Data <- NB_values <- dates <- textDiag <- c()
NHC_Data <- extract_intermediate_values(NHC_Data, lines, "NHC:")
NB_values <- extract_intermediate_values(NB_values, lines, "biopsia:")
dates <- extract_intermediate_values(dates, lines, "Fecha:")
textDiag <- extract_intermediate_values(textDiag, lines, "de la muestra:")
```

`extract_values_from_tables` *Extract values from tables within text*

Description

This function analyzes a subset of text lines, extracting information such as mutations, pathogenicity, frequencies, codifications and changes.

Usage

```
extract_values_from_tables(
  lines,
  mutations,
  genes_mutated = list(),
  pathogenicity = list(),
  frequencies = list(),
  codifications = list(),
  changes = list(),
  values = list(),
  start = "Variantes de secuencia de ADN",
  start2 = "   Variaciones del número de copias",
  end = "Genes analizados",
  end2 = "Comentarios adicionales sobre las variantes"
)
```

Arguments

`lines`	Character vector. Lines of text to process.
`mutations`	Character vector. List of known mutation identifiers.
`genes_mutated`	Ordered list to store extracted gene data.
`pathogenicity`	Ordered list to store extracted pathogenicity information.
`frequencies`	Ordered list to store extracted frequency data.
`codifications`	Ordered list to store extracted codification data.
`changes`	Ordered list to store extracted changes data.
`values`	Aggregated list of extracted information.
`start`	Starting marker for the relevant table section.
`start2`	Secondary starting marker for the table section, in case the table is divided in two pages.
`end`	text marker indicating the end of the subset.
`end2`	secondary end marker.

Value

A list containing extracted data: genes, pathogenicity, frequencies, codifications and changes.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file

genes_file <- system.file("extdata/Genes.xlsx", package = "ORscraper")
genes <- readxl::read_excel(genes_file)
mutations <- unique(genes$GEN)

TableValues <- extract_values_from_tables(lines, mutations)
mutateGenes <- TableValues[[1]]
pathogenity <- TableValues[[2]]
frequencies <- TableValues[[3]]
codifications <- TableValues[[4]]
changes <- TableValues[[5]]
```

---

extract_values_start_end

Extract values from start or end patterns

Description

This function appends extracted variable values based on start or end markers to a list.

Usage

```
extract_values_start_end(list_input, lines, pattern)
```

Arguments

list_input	List. The list to append extracted values to.
lines	Character vector. The text lines to search within.
pattern	Character. The pattern to search for.

Value

An updated list with appended values.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file
diagnostic <- gender <- tumor_cell_percentage <- quality <- c()
diagnostic <- extract_values_start_end(diagnostic, lines, ".*Diagnóstico:\\s")
gender <- extract_values_start_end(gender, lines, ".*Sexo:\\s*")
tumor_cell_percentage <- extract_values_start_end(
```

```
                                tumor_cell_percentage,
                                lines,
                                ".*% células tumorales:\\s")
  quality <- extract_values_start_end(
                                quality,
                                lines,
                            ".*CALIDAD DE LA MUESTRA /LIMITACIONES PARA SU ANÁLISIS:\\s")
```

`extract_variable` *Extract variable value from text lines*

Description

This function searches for a specific pattern in text lines and extracts the corresponding value.

Usage

```
extract_variable(lines, search_text)
```

Arguments

`lines`	Character vector. The lines of text to search within.
`search_text`	Character. The regular expression pattern to match.

Value

The extracted value as a character, or "Null" if not found.

`filter_pathogenic_only` *Filter for pathogenic results only*

Description

This function filters a list of pathogenicity classifications, retaining only those marked as "Pathogenic".

Usage

```
filter_pathogenic_only(pathogenic_list, related_list)
```

Arguments

`pathogenic_list`	List. A list of pathogenicity classifications.
`related_list`	List. A list of corresponding data to filter alongside pathogenicity.

Value

A list containing only the elements of the related list corresponding to "Pathogenic" classifications.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file

genes_file <- system.file("extdata/Genes.xlsx", package = "ORscraper")
genes <- readxl::read_excel(genes_file)
mutations <- unique(genes$GEN)

TableValues <- extract_values_from_tables(lines, mutations)
mutateGenes <- TableValues[[1]]
pathogenity <- TableValues[[2]]
frequencies <- TableValues[[3]]
changes <- TableValues[[5]]

pathogenic_mutations <- filter_pathogenic_only(pathogenity, mutateGenes)
pathogenic_changes <- filter_pathogenic_only(pathogenity, changes)
pathogenic_frequencies <- filter_pathogenic_only(pathogenity, frequencies)
```

---

narrow_text *Extract a subset of text based on start and end patterns*

Description

This function extracts lines from a text based on specified start and end markers.

Usage

```
narrow_text(
  start_text,
  start_text2 = "   Variaciones del número de copias",
  lines_total,
  text_limit,
  text_limit2 = "Comentarios adicionales sobre las variantes"
)
```

Arguments

start_text	Character. The text marker indicating the beginning of the subset.
start_text2	Character. An optional secondary start marker.
lines_total	Character vector. The full set of text lines.
text_limit	Character vector. The text marker indicating the end of the subset.
text_limit2	Character vector. An optional secondary end marker.

Value

A character vector containing the extracted lines.

`read_pdf_content` *Read content from a PDF file*

Description

This function extracts the text content from a PDF file and splits it into individual lines.

Usage

```
read_pdf_content(file_path)
```

Arguments

`file_path`	Character. The path to the PDF file.

Value

A character vector, where each element is a line from the PDF content.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])
head(lines)
```

`read_pdf_files` *Read all PDF files in a directory*

Description

This function scans a specified directory and retrieves all files with a .pdf extension.

Usage

```
read_pdf_files(path)
```

Arguments

`path`	Character. Path to the directory to scan for PDF files.

Value

A character vector with the full paths of the PDF files.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
```

`search_ncbi_clinvar` *Search for pathogenicity information in NCBI ClinVar*

Description

This function queries the NCBI ClinVar database for germline classifications based on gene and codification data.

Usage

```
search_ncbi_clinvar(pathogenicity, genes_mutated, total_codifications)
```

Arguments

`pathogenicity`	Ordered list. Existing pathogenicity data.
`genes_mutated`	Ordered list. Existing mutated gene data.
`total_codifications`	Ordered list. Existing mutated gen codification data.

Value

An updated list of pathogenicity classifications based on NCBI ClinVar search results.

Examples

```
InputPath <- system.file("extdata", package = "ORscraper")
files <- read_pdf_files(InputPath)
lines <- read_pdf_content(files[1])  # Example with the first file

genes_file <- system.file("extdata/Genes.xlsx", package = "ORscraper")

if (requireNamespace("readxl", quietly = TRUE)) {
  genes <- readxl::read_excel(genes_file)
  mutations <- unique(genes$GEN)

  TableValues <- extract_values_from_tables(lines, mutations)
  mutateGenes <- TableValues[[1]]
  pathogenity <- TableValues[[2]]
  codifications <- TableValues[[4]]
```

```
  search_pathogenity <- search_ncbi_clinvar(pathogenity, mutateGenes, codifications)
}
```

`search_value` *Search for a specific value in text lines*

Description

This function searches for a specific text pattern in a set of lines and extracts values that follow the pattern.

Usage

```
search_value(search_text, lines)
```

Arguments

`search_text`	Character. The pattern to search for in the text lines.
`lines`	Character vector. The lines of text to search within.

Value

A character vector with extracted values matching the search criteria.

Index

classify_biopsy, 2

extract_chip_id, 3
extract_fusions, 3
extract_intermediate_values, 4
extract_values_from_tables, 5
extract_values_start_end, 6
extract_variable, 7

filter_pathogenic_only, 7

narrow_text, 8

read_pdf_content, 9
read_pdf_files, 9

search_ncbi_clinvar, 10
search_value, 11