Package: JATSdecoder (via r-universe)

October 30, 2026

Title A Metadata and Text Extraction and Manipulation Tool Set

Date 2026-04-09

Version 1.3.1

Maintainer Ingmar Böschen <ingmar.boeschen@uni-hamburg.de>

Description Provides a function collection to extract metadata, sectioned text and study characteristics from scientific articles in 'NISO-JATS' format. Articles in PDF format can be converted to 'NISO-JATS' with the 'Content ExtRactor and MINEr' ('CERMINE', <https://github.com/CeON/CERMINE>). For convenience, two functions bundle the extraction heuristics: JATSdecoder() converts 'NISO-JATS'-tagged XML files to a structured list with elements title, author, journal, history, 'DOI', abstract, sectioned text and reference list. study.character() extracts multiple study characteristics like number of included studies, statistical methods used, alpha error, power, statistical results, correction method for multiple testing, software used. The function get.stats() extracts all statistical results from text and recomputes p-values for many standard test statistics. It performs a consistency check of the reported with the recalculated p-values. An estimation of the involved sample size is performed based on textual reports within the abstract and the reported degrees of freedom within statistical results. In addition, the package contains some useful functions to process text (text2sentences(), text2num(), ngram(), strsplit2(), grep2()). See Böschen, I. (2021) <doi:10.1007/s11192-021-04162-z> Böschen, I. (2021) <doi:10.1038/s41598-021-98782-3>, Böschen, I. (2023) <doi:10.1038/s41598-022-27085-y>, and Böschen, I. (2024) <doi:10.48550/arXiv.2408.07948>.

Depends R (>= 3.1.1)

Imports utils, stats, NLP, openNLP

License GPL-3

URL https://github.com/ingmarboeschen/JATSdecoder

BugReports https://github.com/ingmarboeschen/JATSdecoder/issues

Language en-US

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Ingmar Böschen [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1159-3991>)

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

Date/Publication 2026-04-09 09:43:39 UTC

RemoteUrl https://github.com/cran/JATSdecoder

RemoteRef HEAD

RemoteSha a5cdccc42de2f1556fdfb8cb5fec83fa558b3b0b

Contents

allStats . . . 3
est.ss . . . 4
get.abstract . . . 5
get.aff . . . 6
get.alpha.error . . . 7
get.assumptions . . . 8
get.author . . . 8
get.category . . . 9
get.country . . . 10
get.doi . . . 10
get.editor . . . 11
get.history . . . 12
get.journal . . . 12
get.keywords . . . 13
get.method . . . 14
get.multi.comparison . . . 14
get.n.studies . . . 15
get.outlier.def . . . 16
get.power . . . 16
get.R.package . . . 17
get.references . . . 18
get.sig.adjectives . . . 18
get.software . . . 19
get.stats . . . 20
get.subject . . . 22
get.tables . . . 23
get.test.direction . . . 23
get.text . . . 24
get.title . . . 25

get.type 26
get.vol 26
grep2 27
has.interaction 28
JATSdecoder 28
letter.convert 30
ngram 31
pCheck 32
standardStats 33
strsplit2 35
study.character 35
text2num 38
text2sentences 39
vectorize.text 40
which.term 40

Index **42**

`allStats` *allStats*

Description

Extracts statistical results within a text string and outputs a vector of sticked results, e.g.: c("t(12)=1.2, p>.05","r's(33)>.7, ps<.05"), that can be further processed with `standardStats`. This function is implemented in `get.stats` which returns the results of `allStats` and `standardStats`. Besides only plain textual input, `get.stats` enables direct processing of different file formats (NISO-JATS coded XML, DOCX, HTML) without text preprocessing.

Usage

```
allStats(x)
```

Arguments

x	A character string that may contain statistical results.

Value

Vector with sticked results. Empty, if no result is detected.

Source

A minimal web application that extracts statistical results from single documents with `get.stats` is hosted at: https://www.get-stats.app/

References

Böschen (2021). "Evaluation of JATSdecoder as an automated text extraction tool for statistical results in scientific reports." *Scientific Reports.* doi: 10.1038/s41598-021-98782-3.

See Also

study.character for extracting multiple study characteristics at once.

get.stats for extracting statistical results from textual input and different file formats.

Examples

```
x<-c("The mean difference of scale A was significant (beta=12.9, t(18)=2.5, p<.05)",
"The ANOVA yielded significant results on factor A (F(2,18)=6, p<.05, eta(g)2<-.22).",
"The correlation of x and y was r=.37.")
allStats(x)
```

---

est.ss *est.ss*

Description

Function to estimate studies sample size by maximizing different conservative estimates. Performs four different extraction heuristics for sample sizes mentioned in abstract, text and statistical results.

Usage

```
est.ss(
  abstract = NULL,
  text = NULL,
  stats = NULL,
  standardStats = NULL,
  quantileDF = 0.9,
  max.only = FALSE,
  max.parts = TRUE
)
```

Arguments

abstract	an abstract text string.
text	the main text string to process (usually method and result sections). If text has content, arguments "stats" and "standardStats" are deactivated and filled with results by get.stats(text).
stats	statistics extracted with get.stats(x)$stats (only active if no text is submitted).
standardStats	standard statistics extracted with get.stats(x)$standardStats (only active if no text is submitted).
quantileDF	quantile of (df1-1)+(df2+2) to extract.
max.only	Logical. If TRUE only the final estimate will be returned, if FALSE all sub estimates are returned as well.
max.parts	Logical. If FALSE outputs all captured sample sizes in sub inputs.

Details

Sample size extraction from abstract:
- Extracts N= from abstract text and performs position-of-speech search with list of synonyms of sample units

Sample size extraction from text:
- Unifies and extracts textlines with age descriptions, than computes sum of hits as nage - Unifies and extracts all "numeric male-female" patterns than computes sum of first male/female hit - Unifies and extracts textlines with participant description than computes sum of first three hits as ntext

Sample size extraction from statistical results:
- Extracts "N=" in statistical results extracted with allStats() that contain p-value: e.g.: chi(2, N=12)=15.2, p<.05

Sample size extraction by degrees of freedom with result of standardStats(allStats()):
- Extracts df1 and df2 if possible and neither containing a ".", than calculates specified quantile of (df1+1)+(df2+2) (at least 2 group comparison assumed)

Value

Numeric vector with extracted sample sizes by input and estimated sample size.

See Also

`study.character` for extracting multiple study characteristics at once.

Examples

```
a<-"One hundred twelve students participated in our study."
est.ss(abstract=a)
x<-"Our sample consists of three hundred twenty five undergraduate students.
    The F-test indicates significant differences in means F(2,102)=3.21, p<.05."
est.ss(text=x)
```

`get.abstract` *get.abstract*

Description

Extracts abstract tag from NISO-JATS coded XML file or text as vector of abstracts.

Usage

```
get.abstract(
  x,
  sentences = FALSE,
  remove.title = TRUE,
  letter.convert = TRUE,
  cermine = FALSE
)
```

Arguments

x	a NISO-JATS coded XML file or text.
sentences	Logical. If TRUE abstract is returned as vector of sentences.
remove.title	Logical. If TRUE removes section titles in abstract.
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
cermine	Logical. If TRUE and if 'letter.convert=TRUE' CERMINE specific letter correction is carried out (e.g. inserting of missing operators to statistical results).

Value

Character. The abstract/s text as floating text or vector of sentences.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <abstract>Some abstract</abstract> some text"
get.abstract(x)
x<-"Some text <abstract>Some abstract</abstract> TEXT <abstract with subsettings>
Some other abstract</abstract> Some text "
get.abstract(x)
```

---

get.aff *get.aff*

Description

Extracts the affiliation tag information from NISO-JATS coded XML file or text as a vector of affiliations.

Usage

```
get.aff(x, remove.html = FALSE, letter.convert = TRUE)
```

Arguments

x	a NISO-JATS coded XML file or text.
remove.html	Logical. If TRUE removes all html tags.
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.

Value

Character vector with the extracted affiliation name/s.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <aff>Some affiliation</aff> some text"
get.aff(x)
x<-"TEXT <aff>Some affiliation</aff> TEXT <aff>Some other affiliation</aff> TEXT"
get.aff(x)
```

`get.alpha.error` *get.alpha.error*

Description

Extracts reported and corrected alpha error from text and 1-alpha confidence intervalls.

Usage

```
get.alpha.error(x, p2alpha = TRUE, output = "list")
```

Arguments

x	text string to process.
p2alpha	Logical. If TRUE detects and extracts alpha errors denoted with a critical p-value (may lead to some false positive detections).
output	One of c("list","vector"). If output="list" returns a list containing: alpha_error, corrected_alpha, alpha_from_CI, alpha_max, alpha_min. If output="vector" returns unique alpha errors but no distinction of types.

Value

Numeric. Vector with identified alpha-error/s.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-c("The threshold for significance was adjusted to .05/2",
"Type 1 error rate was alpha=.05.")
get.alpha.error(x)
x<-c("We used p<.05 as level of significance.",
     "We display .95 CIs and use an adjusted alpha of .10/3.",
     "The effect was significant with p<.025.")
get.alpha.error(x)
```

get.assumptions *get.assumptions*

Description

Extracts the mentioned statistical assumptions from a text string by a dictionary search of 22 common statistical assumptions.

Usage

```
get.assumptions(x, hits_only = TRUE)
```

Arguments

x	text string to process.
hits_only	Logical. If TRUE returns the detected assumtions only, else a hit matrix with all potential assumptions is returned.

Value

Character. Vector with identified statistical assumption/s.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-"Sphericity assumption and gaus-marcov was violated."
get.assumptions(x)
```

get.author *get.author*

Description

Extracts author tag information from NISO-JATS coded XML file or text.

Usage

```
get.author(x, paste = "", short.names = FALSE, letter.convert = FALSE)
```

Arguments

x	a NISO-JATS coded XML file or text.
paste	if paste!="" author list is collapsed to one cell with seperator specified (e.g. paste=";").
short.names	Logical. If TRUE fully available first names will be reduced to single letter abbreviation.
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.

Value

Character vector with the extracted author name/s.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

---

get.category *get.category*

Description

Extracts category tag/s from NISO-JATS coded XML file or text as vector of categories.

Usage

```
get.category(x)
```

Arguments

x	a NISO-JATS coded XML file or text.

Value

Character vector with the extracted category name/s.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <article-categories>Some category</article-categories> some text"
get.category(x)
```

get.country *get.country*

Description

Extracts country tag from NISO-JATS coded XML file or text as vector of unique countries.

Usage

```
get.country(x, unifyCountry = TRUE)
```

Arguments

x	a NISO-JATS coded XML file or text.
unifyCountry	Logical. If TRUE replaces country name with standardised country name.

Value

Character vector with the extracted country name/s.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <country>UK</country> some text <country>England</country>
    Text<country>Berlin, Germany</country>"
get.country(x)
```

get.doi *get.doi*

Description

Extracts articles doi from NISO-JATS coded XML file or text.

Usage

```
get.doi(x)
```

Arguments

x	a NISO-JATS coded XML file or text.

Value

Character string with the extracted doi.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

`get.editor` *get.editor*

Description

Extracts editor tag from NISO-JATS coded XML file or text as vector of editors.

Usage

```
get.editor(x, role = FALSE, short.names = FALSE, letter.convert = FALSE)
```

Arguments

`x`	a NISO-JATS coded XML file or text.
`role`	Logical. If TRUE adds role to editor name, if available.
`short.names`	Logical. If TRUE reduces fully available first names to one letter abbreviation.
`letter.convert`	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.

Value

Character string with the extracted editor name/s.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.history *get.history*

Description

Extracts available publishing history tags from NISO-JATS coded XML file or text and compute pubDate and pubyear.

Usage

```
get.history(x, remove.na = FALSE)
```

Arguments

x	a NISO-JATS coded XML file or text.
remove.na	Logical. If TRUE hides non available date stamps.

Value

Character vector with the extracted dates of publishing history.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.journal *get.journal*

Description

Extracts journal tag from NISO-JATS coded XML file or text.

Usage

```
get.journal(x)
```

Arguments

x	a NISO-JATS coded XML file or text.

Value

Character string with the extracted journal name.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <journal-title>PLoS One</journal-title> some text"
get.journal(x)
```

`get.keywords` *get.keywords*

Description

Extracts keyword tag/s from NISO-JATS coded XML file or text as vector of keywords.

Usage

```
get.keywords(
  x,
  paste = "",
  letter.convert = TRUE,
  include.max = length(keyword)
)
```

Arguments

`x`	a NISO-JATS coded XML file or text.
`paste`	if paste!="" keyword list is collapsed to one cell with seperator specified (e.g. paste=";").
`letter.convert`	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
`include.max`	a maximum number of keywords to extract.

Value

Character vector with extracted keyword/s.

See Also

`JATSdecoder` for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <kwd>Keyword 1</kwd>, <kwd>Keyword 2</kwd> some text"
get.keywords(x)
get.keywords(x,paste(", "))
```

get.method *get.method*

Description

Extracts statistical methods mentioned in text.

Usage

```
get.method(x, add = NULL, cermine = FALSE)
```

Arguments

x	text to extract statistical methods from.
add	possible new end words of method as vector.
cermine	Logical. If TRUE CERMINE specific letter conversion will be performed.

Value

Character. Vector with identified statistical method/s

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-"We used multiple regression analysis and
two sample t tests to evaluate our results."
get.method(x)
```

get.multi.comparison *get.multi.comparison*

Description

Extracts alpha-/p-value correction method for multiple comparisons from list with 15 correction methods.

Usage

```
get.multi.comparison(x)
```

Arguments

x	text string to process.

Value

Character. Identified author/method of multiple comparison correction procedure.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-"We used Bonferroni corrected p-values."
get.multi.comparison(x)
```

---

get.n.studies *get.n.studies*

Description

Extracts number of studies/experiments from text.

Usage

```
get.n.studies(x, tolower = TRUE)
```

Arguments

x	text string to process.
tolower	Logical. If TRUE lowerises text and search patterns for processing.

Value

Numeric number of identified number of studies. Returns '1' as standard output.

See Also

study.character for extracting multiple study characteristics at once.

get.outlier.def *get.outlier.def*

Description

Extracts outlier/extreme value definition/removal in standard deviations, if present in text.

Usage

```
get.outlier.def(x, range = c(1, 10))
```

Arguments

x	Character. A text string to process.
range	Numeric vector with length=2. Possible result space of extracted value/s in standard deviations. Use 'c(0,Inf)' for no restriction.

Value

Numeric. Vector with identified outlier definition in standard deviations.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-"We removed 4 extreme values that were 3 SD above mean."
get.outlier.def(x)
```

get.power *get.power*

Description

Extracts a priori power and empirial power values from text.

Usage

```
get.power(x)
```

Arguments

x	text string to process.

Value

Numeric. Identified power values.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
x<-"We used G*Power 3 to calculate the needed sample with
beta error rate set to 12% and alpha error to .05."
get.power(x)
```

---

get.R.package *get.R.package*

Description

Extracts mentioned R packages from text.

Usage

```
get.R.package(x, update.package.list = FALSE)
```

Arguments

x	text string to process.
update.package.list	Logical. If TRUE update of list with available packages is downloaded from CRAN with utils::available.packages().

Value

Character. Vector with identified R package/s.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
get.R.package("We used the R Software packages lme4 (and psych).")
```

get.references *get.references*

Description

Extracts reference list from NISO-JATS coded XML file or text as vector of references.

Usage

```
get.references(
  x,
  letter.convert = FALSE,
  remove.html = FALSE,
  extract = "full"
)
```

Arguments

x	a NISO-JATS coded XML file or text.
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
remove.html	Logical. If TRUE removes all HTML tags.
extract	part of refernces to extract (one of "full" or "title").

Value

Character vector with extracted references from reference list.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.sig.adjectives *get.sig.adjectives*

Description

Extracts adjectives used for in/significance out of list with 37 potential adjectives.

Usage

```
get.sig.adjectives(x, unique_only = FALSE)
```

Arguments

x	text string to process.
unique_only	Logical. If TRUE returns unique hits only.

Value

Character. Vector with identified adjectives.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
get.sig.adjectives(
 x<-"We found very highly significance for type 1 effect"
)
```

---

get.software *get.software*

Description

Extracts mentioned software from text by dictionary search for 63 software names (object: .software_names).

Usage

```
get.software(x, add.software = NULL)
```

Arguments

x	text string to process.
add.software	a text vector with additional software name patterns to search for.

Value

Character. Vector with identified statistical software/s.

See Also

study.character for extracting multiple study characteristics at once.

Examples

```
get.software("We used the R Software and Excel 4.0 to analyse our data.")
```

get.stats *get.stats*

Description

Extracts statistical results from text string, XML, CERMXML, HTML or DOCX files. The result is a list with a vector containing all identified sticked results and a matrix containing the reported standard statistics and recalculated p-values if computation is possible.

Usage

```
get.stats(
  x,
  output = "both",
  stats.mode = "all",
  recalculate.p = TRUE,
  checkP = FALSE,
  alpha = 0.05,
  criticalDif = 0.02,
  alternative = "undirected",
  estimateZ = FALSE,
  T2t = FALSE,
  R2r = FALSE,
  select = NULL,
  rm.na.col = TRUE,
  cermine = FALSE,
  warnings = TRUE
)
```

Arguments

x	NISO-JATS coded XML or DOCX file path or plain textual content.
output	Select the desired output. One of c("both", "allStats", "standardStats").
stats.mode	Select a subset of test results by p-value checkability for output. One of: c("all", "checkable", "computable", "uncomputable").
recalculate.p	Logical. If TRUE recalculates p-values of test results if possible.
checkP	Logical. If TRUE observed and recalculated p-values are checked for consistency.
alpha	Numeric. Defines the alpha level to be used for error assignment.
criticalDif	Numeric. Sets the absolute maximum difference in reported and recalculated p-values for error detection.
alternative	Character. Select test sidedness for recomputation of p-values from t-, r- and beta-values. One of c("undirected", "directed"). If "directed" is specified, p-values for directed null-hypothesis are added to the table but still require a manual inspection on consistency of the direction.

estimateZ	Logical. If TRUE detected beta-/d-value is divided by reported standard error "SE" to estimate Z-value ("Zest") for observed beta/d and recompute p-value. Note: This is only valid, if Gauss-Marcov assumptions are met and a sufficiently large sample size is used. If a Z- or t-value is detected in a report of a beta-/d-coefficient with SE, no estimation will be performed, although set to TRUE.
T2t	Logical. If TRUE capital letter T is treated as t-statistic.
R2r	Logical. If TRUE capital letter R is treated as correlation.
select	Select specific standard statistics only (e.g.: c("t", "F", "Chi2")).
rm.na.col	Logical. If TRUE removes all columns with only NA from standardStats.
cermine	Logical. If TRUE CERMINE specific letter conversion will be peformed on allStats results.
warnings	Logical. If FALSE warning messages are omitted.

Value

If output="all": list with two elements. E1: vector of extracted results by `allStats` and E2: matrix of standard results by `standardStats`.
If output="allStats": vector of extracted results by `allStats`.
If output="standardStats": matrix of standard results by `standardStats`.

Note

In order to extract statistical test results reported in simple/unpublished PDF documents with get.stats(), the function *pdf_text()* from the pdftools package may help to extract textual content from pdf files (be aware that tabled content may cause corrupt text).

Source

A minimal web application that extracts statistical results from single documents with `get.stats` is hosted at: https://www.get-stats.app/

Statistical results extracted with `get.stats` can be analyzed and used to identify articles stored in the PubMed Central library at: https://www.scianalyzer.com/.

References

Böschen (2021). "Evaluation of JATSdecoder as an automated text extraction tool for statistical results in scientific reports." *Scientific Reports.* doi: 10.1038/s41598-021-98782-3.

See Also

`study.character` for extracting different study characteristics at once.

Examples

```
## Extract results from plain text input
x<-c("The mean difference of scale A was significant (beta=12.9, t(18)=2.5, p<.05).",
"The ANOVA yielded significant results on
 faktor A (F(2,18)=6, p<.05, eta(g)2<-.22)",
```

```
"the correlation of x and y was r=.37.")
get.stats(x)

## Extract results from native NISO-JATS XML file
# download example XML file via URL if a connection is possible
x<-"https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0114876&type=manuscript"
# file name
file<-paste0(tempdir(),"/file.xml")
# download URL as "file.xml" in tempdir() if a connection is possible
tryCatch({
  readLines(x,n=1)
  download.file(x,file)
  },
  warning = function(w) message(
  "Something went wrong. Check your internet connection and the link address."),
  error = function(e) message(
  "Something went wrong. Check your internet connection and the link address.")
)
# apply get.stats() to file
if(file.exists(file)) get.stats(file)
```

`get.subject` *get.subject*

Description

Extracts subject tag/s from NISO-JATS coded XML file or text as vector of subjects.

Usage

```
get.subject(x, letter.convert = TRUE, paste = "")
```

Arguments

`x`	a NISO-JATS coded XML file or text.
`letter.convert`	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
`paste`	if paste!="" subject list is collapsed to one cell with seperator specified (e.g. paste=";").

Value

Character vector with extracted subject/s.

See Also

`JATSdecoder` for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

Examples

```
x<-"Some text <subject>Some subject</subject> some text"
get.subject(x)
x<-"Some text <subject>Some subject</subject> TEXT ...
<subject>Some other subject</subject> Some text "
get.subject(x)
get.subject(x,paste=", ")
```

---

get.tables *get.tables*

Description

Extracts HTML tables as vector of tables.

Usage

```
get.tables(x)
```

Arguments

x	HTML file or text with html tags.

Value

Character vector with extracted HTML tables.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

---

get.test.direction *get.test.direction*

Description

Extracts mentioned test direction/s (one sided, two sided, one and two sided) from text.

Usage

```
get.test.direction(x)
```

Arguments

x	text string to process.

Value

Character.

See Also

study.character for extracting multiple study characteristics at once.

---

get.text *get.text*

Description

Extracts main textual content from NISO-JATS coded XML file or text as sectioned text.

Usage

```
get.text(
  x,
  sectionsplit = "",
  grepsection = "",
  letter.convert = TRUE,
  greek2text = FALSE,
  sentences = FALSE,
  paragraph = FALSE,
  cermine = "auto",
  rm.table = TRUE,
  rm.formula = TRUE,
  rm.xref = TRUE,
  rm.media = TRUE,
  rm.graphic = TRUE,
  rm.ext_link = TRUE
)
```

Arguments

x	a NISO-JATS coded XML file or text.
sectionsplit	search patterns for section split (forced to lower case), e.g. c("intro", "method", "result", "discus").
grepsection	search pattern to reduce text to specific section namings only.
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
greek2text	Logical. If TRUE some greek letters and special characters will be unified to textual representation (important to extract stats).
sentences	Logical. IF TRUE text is returned as sectioned list with sentences.

paragraph	Logical. IF TRUE "<New paragraph>" is added at the end of each paragraph to enable manual splitting at paragraphs.
cermine	Logical. If TRUE CERMINE specific error handling and letter conversion will be applied. If set to "auto" file name ending with 'cermxml$' will set cermine=TRUE.
rm.table	Logical. If TRUE removes <table> tag from text.
rm.formula	Logical. If TRUE removes <formula> tags.
rm.xref	Logical. If TRUE removes <xref> tag (citing) from text.
rm.media	Logical. If TRUE removes <media> tag from text.
rm.graphic	Logical. If TRUE removes <graphic> and <fig> tag from text.
rm.ext_link	Logical. If TRUE removes <ext link> tag from text.

Value

List with two elements. 1: Character vector with section title/s, 2: Character vector with floating text of sections or list with vector of sentences per section/s if sentences=TRUE.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

---

get.title *get.title*

Description

Extracts article title from NISO-JATS coded XML file or text.

Usage

```
get.title(x)
```

Arguments

x	a NISO-JATS coded XML file or text.

Value

Character string with extracted article title.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.type *get.type*

Description

Extracts article type from NISO-JATS coded XML file or text.

Usage

```
get.type(x)
```

Arguments

x	a NISO-JATS coded XML file or text.

Value

Character string with extracted article type.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.vol *get.vol*

Description

Extracts volume, first and last page from NISO-JATS coded XML file or text.

Usage

```
get.vol(x)
```

Arguments

x	a NISO-JATS XML coded file or text.

Value

Character string with extracted journal volume.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

grep2 *grep2*

Description

Extension of grep(). Allows to identify and extract cells with/without multiple search patterns that are connected with AND.

Usage

```
grep2(pattern, x, value = TRUE, invert = FALSE, perl = FALSE)
```

Arguments

pattern	Character vector containing regular expression as cells to be matched in the given character vector.
x	A character vector where matches are sought, or an object which can be coerced by as.character to a character vector. Long vectors are supported.
value	Logical. If FALSE, a vector containing the (integer) indices of the matches determined by grep2 is returned, and if TRUE, a vector containing the matching elements themselves is returned.
invert	Logical. If TRUE return indices or values for elements that do not match.
perl	Logical. Should Perl-compatible regexps be used?

Value

grep2(value = FALSE) returns a vector of the indices of the elements of x that yielded a match (or not, for invert = TRUE). This will be an integer vector unless the input is a long vector, when it will be a double vector.

grep2(value = TRUE) returns a character vector containing the selected elements of x (after coercion, preserving names but no other attributes).

See Also

grep

Examples

```
x<-c("ab","ac","ad","bc","bad")
grep2(c("a","b"),x)
grep2(c("a","b"),x,invert=TRUE)
grep2(c("a","b"),x,value=FALSE)
```

has.interaction *has.interaction*

Description

Identifies mentiones of interaction/moderator/mediator effect in text.

Usage

```
has.interaction(x)
```

Arguments

x	text string to process.

Value

Character vector with type/s of identified interaction/moderator/mediator effect.

See Also

study.character for extracting multiple study characteristics at once.

JATSdecoder *JATSdecoder*

Description

Function to extract and restructure NISO-JATS coded XML file or text into a list with metadata and text as selectable elements. Use CERMINE to convert PDF to CERMXML files.

Usage

```
JATSdecoder(
  x,
  sectionsplit = c("intro", "method", "result", "study", "experiment", "conclu",
    "implica", "discussion"),
  grepsection = "",
  sentences = FALSE,
  paragraph = FALSE,
  abstract2sentences = TRUE,
  output = "all",
  letter.convert = TRUE,
  unify.country.name = TRUE,
  greek2text = FALSE,
  warning = TRUE,
```

```
  countryconnection = FALSE,
  authorconnection = FALSE
)
```

Arguments

x	a NISO-JATS coded XML file or text.
sectionsplit	search patterns for section split of text parts (forced to lower case), e.g. c("intro", "method", "result", "discus").
grepsection	search pattern in regex to reduce text to specific section only.
sentences	Logical. IF TRUE text is returned as sectioned list with sentences.
paragraph	Logical. IF TRUE "<New paragraph>" is added at the end of each paragraph to enable manual splitting at paragraphs.
abstract2sentences	Logical. IF TRUE abstract is returned as vector with sentences.
output	selection of specific results to output c("all", "title", "author", "affiliation", "journal", "volume", "editor", "doi", "type", "history", "country", "subject", "keywords", "abstract", "sections", "text", "tables", "captions", "references").
letter.convert	Logical. If TRUE converts hexadecimal and HTML coded characters to Unicode.
unify.country.name	Logical. If TRUE tries to unify country name/s with list of country names from worldmap().
greek2text	Logical. If TRUE converts and unifies several greek letters to textual representation, e.g.: "alpha".
warning	Logical. If TRUE outputs a warning if processing CERMINE converted PDF files.
countryconnection	Logical. If TRUE outputs country connections as vector c("A - B","A - C", ...).
authorconnection	Logical. If TRUE outputs connections of a maximum of 50 involved authors as vector c("A - B","A - C", ...).

Value

List with extracted meta data, sectioned text and references.

Note

A short tutorial on how to work with JATSdecoder and the generated outputs can be found at: https://github.com/ingmarboeschen/JATSdecoder

Source

An interactive web application for selecting and analyzing extracted article metadata and study characteristics for articles linked to PubMed Central is hosted at: https://www.scianalyzer.com/

The XML version of PubMed Central database articles can be downloaded in bulk from: https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/

References

Böschen (2021). "Software review: The JATSdecoder package - extract metadata, abstract and sectioned text from NISO-JATS coded XML documents; Insights to PubMed Central's open access database." *Scientometrics.* doi: 10.1007/s1119202104162z.

See Also

`study.character` for extracting different study characteristics at once.

`get.stats` for extracting statistical results from textual input and different file formats.

Examples

```
# download example XML file via URL
x<-"https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0114876&type=manuscript"
# file name
file<-paste0(tempdir(),"/file.xml")
# download URL as "file.xml" in tempdir() if a connection is possible
tryCatch({
readLines(x,n=1)
download.file(x,file)
},
warning = function(w) message(
  "Something went wrong. Check your internet connection and the link address."),
error = function(e) message(
  "Something went wrong. Check your internet connection and the link address."))
# convert full article to list with metadata, sectioned text and reference list
if(file.exists(file)) JATSdecoder(file)
# extract specific content (here: abstract and text)
if(file.exists(file)) JATSdecoder(file,output=c("abstract","text"))
# or use specific functions, e.g.:
if(file.exists(file)) get.abstract(file)
if(file.exists(file)) get.text(file)
```

`letter.convert` *letter.convert*

Description

Converts and unifies most hexadecimal and some HTML coded letters to Unicode characters. Performs CERMINE specific error correction (inserting operators, where these got lost while conversion).

Usage

```
letter.convert(x, cermine = FALSE, greek2text = FALSE, warning = TRUE)
```

Arguments

x	text string to process.
cermine	Logical. If TRUE CERMINE specific error handling and letter conversion will be applied.
greek2text	Logical. If TRUE some greek letters and special characters will be unified to textual representation (important to extract stats).
warning	Logical. If TRUE prints warning massage if CERMINE specific letter conversion was performed.

Value

Character. Text with unified and corrected letter representation.

Examples

```
x<-c("five &#x0003c; ten","five &lt; ten")
letter.convert(x)
```

---

ngram *ngram*

Description

Extracts ngram bag of words around words that match a search pattern. Note: If an input contains the search pattern twice, only the ngram bag of words of the last hit is detected. Consider individual text splitting with text2sentences() or strsplit2() before applying ngram().

Usage

```
ngram(
  x,
  pattern,
  ngram = c(-3, 3),
  tolower = FALSE,
  split = FALSE,
  exact = FALSE
)
```

Arguments

x	vector of text strings to process.
pattern	a search term pattern to extract the ngram bag of words.
ngram	a vector of length=2 that defines the number of words to extract from left and right side of pattern match.
tolower	Logical. If TRUE converts text and pattern to lower case.
split	Logical. If TRUE splits text input at "[.,;:] " before processing. Note: You may consider other text splits before.
exact	Logical. If TRUE only exact word matches will be proceses

Value

Character. Vector with +-n words of search pattern.

Examples

```
text<-"One hundred twenty-eight students participated in our Study,
that was administred in thirteen clinics."
ngram(text,pattern="study",ngram=c(-1,2))
```

---

pCheck *pCheck*

Description

Wrapper function for a standardStats data frame to check extracted and recalculated p-value for consistency

Usage

```
pCheck(stats, alpha = 0.05, criticalDif = 0.02, add = TRUE, warnings = TRUE)
```

Arguments

stats	Data frame. A data frame object of standard stats that was created by get.stats() or standardStats()
alpha	Numeric. Set the alpha level of tests.
criticalDif	Numeric. Defines the absolute threshold of absolute differences in extracted and recalculated p-value to be labeled as inconsistency.
add	Logical. If TRUE the result of Pcheck are added to the input data frame.
warnings	Logical. If FALSE warning messages are omitted.

Value

A data frame with error report on each entry in the result of a standard stats data frame.

Examples

```
## Extract and check results from plain text input with get.stats(x,checkP=TRUE)
get.stats("some text with consistent or inconsistent statistical results:
t(12)=3.4, p<.05 or t(12)=3.4, p>=.05",checkP=TRUE)
## Check standardStats extracted with get.stats(x)$standardStats
pCheck(get.stats("some text with consistent or inconsistent statistical results:
t(12)=3.4, p<.05 or t(12)=3.4, p>=.05")$standardStats)
```

standardStats *standardStats*

Description

Extracts and restructures statistical standard results like Z, t, Cohen's d, F, eta^2, r, R^2, chi^2, BF_10, Q, U, H, OR, RR, beta values into a matrix. Performs a recomputation of two- and one-sided p-values if possible. This function is implemented in `get.stats` which returns the results of `allStats` and `standardStats`. Besides only plain textual input, `get.stats` enables direct processing of different file formats (NISO-JATS coded XML, DOCX, HTML) without text preprocessing.

Usage

```
standardStats(
  x,
  stats.mode = "all",
  recalculate.p = TRUE,
  alternative = "undirected",
  checkP = FALSE,
  alpha = 0.05,
  criticalDif = 0.02,
  estimateZ = FALSE,
  T2t = FALSE,
  R2r = FALSE,
  select = NULL,
  rm.na.col = TRUE,
  warnings = TRUE
)
```

Arguments

`x`	result vector by `allStats` or chracter vector with a statistical test result per cell, e.g. c("t(12)=1.2, p>.05","chi2(2)=12.7, p<.05")
`stats.mode`	Select subset of standard stats. One of: c("all", "checkable", "computable", "uncomputable").
`recalculate.p`	Logical. If TRUE recalculates p values (for 2 sided test) if possible.
`alternative`	Character. Select test sidedness for recomputation of p-values from t-, r- and beta-values. One of c("undirected", "directed"). If "directed" is specified, p-values for directed null-hypothesis are added to the table but still require a manual inspection on consistency of the direction.
`checkP`	Logical. If TRUE observed and recalculated p-values are checked for consistency.
`alpha`	Numeric. Defines the alpha level to be used for error assignment.
`criticalDif`	Numeric. Sets the absolute maximum difference in reported and recalculated p-values for error detection.

estimateZ	Logical. If TRUE detected beta-/d-values are divided by the reported standard error "SE" to estimate Z-values ("Zest") for observed beta/d and computation of p-values. Further, t-values that are reported without degrees of freedom will be treated as an estimate for Z. Note: This is only valid, if Gauss-Marcov assumptions are met and a sufficiently large sample size is used. If a Z- or t-value with degrees of freedom is detected in a report of a beta-/d-coefficient with SE, no estimation of Z will be performed, although set to TRUE.
T2t	Logical. If TRUE capital letter T is treated as t-statistic.
R2r	Logical. If TRUE capital letter R is treated as correlation.
select	Select specific standard statistics only (e.g.: c("t", "F", "Chi2")).
rm.na.col	Logical. If TRUE removes all columns with only NA.
warnings	Logical. If FALSE warning messages are omitted.

Value

Matrix with recognized statistical standard results and recalculated p-values. Empty, if no result is detected.

Source

A minimal web application that extracts statistical results from single documents with `get.stats` is hosted at: https://www.get-stats.app/

Statistical results extracted with `get.stats` can be analyzed and used to identify articles stored in the PubMed Central library at: https://www.scianalyzer.com/.

References

Böschen (2021). "Evaluation of JATSdecoder as an automated text extraction tool for statistical results in scientific reports." *Scientific Reports.* doi: 10.1038/s41598-021-98782-3.

See Also

`study.character` for extracting multiple study characteristics at once.

`get.stats` for extracting statistical results from textual input and different file formats.

Examples

```
x<-c("t(38.8)<=>1.96, p<=>.002","F(2,39)<=>4, p<=>.05",
"U(2)=200, p>.25","Z=2.1, F(20.8,22.6)=200, p<.005,
BF(01)>4","chi=3.2, r(34)=-.7, p<.01, R2=76%.")
standardStats(x)
```

strsplit2 *strsplit2*

Description

Extension of strsplit(). Makes it possible to split lines before or after a pattern match without removing the pattern.

Usage

```
strsplit2(x, split, type = "remove", perl = FALSE)
```

Arguments

x	text string to process.
split	pattern to split text at.
type	one out of c("remove", "before", "after").
perl	Logical. If TRUE uses perl expressions.

Value

A list of the same length as x, the i-th element of which contains the vector of splits of x[i].

Examples

```
x<-"This is some text, where text is the split pattern of the text."
strsplit2(x,"text","after")
```

study.character *study.character*

Description

Extracts study characteristics out of a NISO-JATS coded XML file. Use CERMINE to convert PDF to CERMXML files.

Usage

```
study.character(
  x,
  stats.mode = "all",
  recalculate.p = TRUE,
  alternative = "auto",
  estimateZ = FALSE,
  T2t = FALSE,
```

```
  R2r = FALSE,
  selectStandardStats = NULL,
  checkP = TRUE,
  criticalDif = 0.02,
  alpha = 0.05,
  p2alpha = TRUE,
  alpha_output = "list",
  captions = TRUE,
  text.mode = 1,
  update.package.list = FALSE,
  add.software = NULL,
  quantileDF = 0.9,
  N.max.only = FALSE,
  output = "all",
  rm.na.col = TRUE
)
```

Arguments

x	NISO-JATS coded XML file.
stats.mode	Character. Select subset of standard stats. One of: c("all", "checkable", "computable").
recalculate.p	Logical. If TRUE recalculates p values (for 2 sided test) if possible.
alternative	Character. Select sidedness of recomputed p-values for t-, r- and Z-values. One of c("auto", "undirected", "directed"). If set to "auto" 'alternative' will be be set to 'directed' if get.test.direction() detects one-directional hypotheses/tests in text. If no directional hypotheses/tests are dtected only "undirected" recomputed p-values will be returned.
estimateZ	Logical. If TRUE detected beta-/d-value is divided by reported standard error "SE" to estimate Z-value ("Zest") for observed beta/d and recompute p-value. Note: This is only valid, if Gauss-Marcov assumptions are met and a sufficiently large sample size is used. If a Z- or t-value is detected in a report of a beta-/d-coefficient with SE, no estimation will be performed, although set to TRUE.
T2t	Logical. If TRUE capital letter T is treated as t-statistic when extracting statistics with get.stats().
R2r	Logical. If TRUE capital letter R is treated as correlation when extracting statistics with get.stats().
selectStandardStats	Select specific standard statistics only (e.g.: c("t", "F", "Chi2")).
checkP	Logical. If TRUE observed and recalculated p-values are checked for consistency.
criticalDif	Numeric. Sets the absolute maximum difference in reported and recalculated p-values for error detection.
alpha	Numeric. Defines the alpha level to be used for error assignment of detected incosistencies.

p2alpha	Logical. If TRUE detects and extracts alpha errors denoted with critical p-value (what may lead to some false positive detections).
alpha_output	One of c("list", "vector"). If alpha_output = "list" a list with elements: alpha_error, corrected_alpha, alpha_from_CI, alpha_max, alpha_min is returned. If alpha_output = "vector" unique alpha errors without a distinction of types is returned.
captions	Logical. If TRUE captions text will be scanned for statistical results.
text.mode	Numeric. Defines text parts to extract statistical results from (text.mode=1: abstract and full text, text.mode=2: method and result section, text.mode=3: result section only).
update.package.list	Logical. If TRUE updates available R packages with utils::available.packages() function.
add.software	additional software names to detect as vector.
quantileDF	quantile of (df1+1)+(df2+1) to extract for estimating sample size.
N.max.only	return only maximum of estimated sample sizes.
output	output selection of specific results c("doi", "title", "year", "Nstudies", "methods", "alpha_error", "power", "multi_comparison_correction", "assumptions", "OutlierRemovalInSD", "InteractionModeratorMediatorEffect", "test_direction", "sig_adjectives", "software", "Rpackage", "stats", "standardStats", "estimated_sample_size").
rm.na.col	Logical. If TRUE removes all columns with only NA in extracted standard statistics.

Value

List with extracted study characteristics.

Note

A short tutorial on how to work with JATSdecoder and the generated outputs can be found at: https://github.com/ingmarboeschen/JATSdecoder

Source

An interactive web application for selecting and analyzing extracted article metadata and study characteristics for articles linked to PubMed Central is hosted at: https://www.scianalyzer.com/

The XML version of PubMed Central database articles can be downloaded in bulk from: https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/

References

Böschen (2023). "Evaluation of the extraction of methodological study characteristics with JATSdecoder." *Scientific Reports.* doi: 10.1038/s41598-022-27085-y.

Böschen (2021). "Evaluation of JATSdecoder as an automated text extraction tool for statistical results in scientific reports." *Scientific Reports.* doi: 10.1038/s41598-021-98782-3.

See Also

JATSdecoder for simultaneous extraction of meta-tags, abstract, sectioned text and reference list.

get.stats for extracting statistical results from textual input and different file formats.

Examples

```
# download example XML file via URL
x<-"https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0114876&type=manuscript"
# file name
file<-paste0(tempdir(),"/file.xml")
# download URL as "file.xml" in tempdir() if a connection is possible
tryCatch({
readLines(x,n=1)
download.file(x,file)
},
warning = function(w) message(
  "Something went wrong. Check your internet connection and the link address."),
error = function(e) message(
  "Something went wrong. Check your internet connection and the link address."))
# convert full article to list with study characteristics
if(file.exists(file)) study.character(file)
```

---

text2num *text2num*

Description

Converts special annotated number and written numbers in a text string to a fully digit representation. Can handle numbers with exponent, fraction, percent, e+num, products and written representation (e.g. 'fourtys-one') of all absolut numbers up to 99,999 (Note: gives wrong output for higher spelled numbers). Process is performed in the same order as its arguments.

Usage

```
text2num(
  x,
  exponent = TRUE,
  percentage = TRUE,
  fraction = TRUE,
  e = TRUE,
  product = TRUE,
  words = TRUE
)
```

Arguments

x	text string to process.
exponent	Logical. If TRUE values with exponent are converted to a digit representation.
percentage	Logical. If TRUE percentages are converted to a digit representation.
fraction	Logical. If TRUE fractions are converted to a digit representation.
e	Logical. If TRUE values denoted with 'number e+number' (e.g. '2e+2') or number*10^number are converted to a digit representation.
product	Logical. If TRUE values products are converted to a digit representation.
words	Logical. If TRUE written numbers are converted to a digit representation.

Value

Character. Text with unified digital representation of numbers.

Examples

```
x<-c("numbers with exponent: 2^2, -2.5^2, (-3)^2, 6.25^.5, .2^-2 text.",
     "numbers with percentage: 2%, 15 %, 25 percent.",
     "numbers with fractions: 1/100, -2/5, -7/.1",
     "numbers with e: 10e+2, -20e3, .2E-2, 2e4",
     "numbers as products: 100*2, -20*.1, 2*10^3",
     "written numbers: twenty-two, one hundred fourty five, fifteen percent",
     "mix: one hundred ten is not 1/10 is not 10^2 nor 10%/5")
text2num(x)
```

---

text2sentences *text2sentences*

Description

Converts floating text to a vector with sentences via fine-tuned regular expressions.

Usage

```
text2sentences(x)
```

Arguments

x	text string to process.

Value

Character vector with sentences compiled from floating text.

Examples

```
x<-"Some text with result (t(18)=1.2, p<.05). This shows how text2sentences works."
text2sentences(x)
```

vectorize.text *vectorize.text*

Description

Converts vector of text to a list of vectors with words within each cell. Note: punctuation will be removed.

Usage

```
vectorize.text(x)
```

Arguments

x	text string to vectorize.

Value

Character vector with one word per cell.

Examples

```
text<-"One hundred twenty-eight students participated in our
Study, that was administred in thirteen clinics."
vectorize.text(text)
```

which.term *which.term*

Description

Returns search element/s from vector that is/are present in text or returns search term hit vector for all terms.

Usage

```
which.term(x, terms, tolower = TRUE, hits_only = FALSE)
```

Arguments

x	text string to process.
terms	search term vector.
tolower	Logical. If TRUE converts search terms and text to lower case.
hits_only	Logical. If TRUE returns search pattern/s, that were found in text and not a search term hit vector.

Value

Binary hit vector with search term named elements if hits_only=FALSE.

Character vector with identified search term elements if hits_only=TRUE.

Examples

```
text<-c("This demo demonstrates how which.term works.",
       "The result is a simple 0, 1 coded vector for all search patterns or
        a vector including the identified patterns only.")
which.term(text,c("Demo","example","work"))
which.term(text,c("Demo","example","work"),tolower=TRUE,hits_only=TRUE)
```

Index

allStats, 3, *3*, *21*, *33*

est.ss, 4

get.abstract, 5
get.aff, 6
get.alpha.error, 7
get.assumptions, 8
get.author, 8
get.category, 9
get.country, 10
get.doi, 10
get.editor, 11
get.history, 12
get.journal, 12
get.keywords, 13
get.method, 14
get.multi.comparison, 14
get.n.studies, 15
get.outlier.def, 16
get.power, 16
get.R.package, 17
get.references, 18
get.sig.adjectives, 18
get.software, 19
get.stats, *3, 4*, 20, *21*, *30*, *33, 34*, *38*
get.subject, 22
get.tables, 23
get.test.direction, 23
get.text, 24
get.title, 25
get.type, 26
get.vol, 26
grep, *27*
grep2, 27

has.interaction, 28

JATSdecoder, *6, 7*, *9–13*, *18*, *22, 23*, *25, 26*, 28, *38*

letter.convert, 30

ngram, 31

pCheck, 32

standardStats, *3*, *21*, 33, *33*
strsplit2, 35
study.character, *4, 5*, *7, 8*, *14–17*, *19*, *21*, *24*, *28*, *30*, *34*, 35

text2num, 38
text2sentences, 39

vectorize.text, 40

which.term, 40