Package ‘flatxml’

October 13, 2022

Type Package

Title Tools for Working with XML Files as R Dataframes

Version 0.1.1

Maintainer Joachim Zuckarelli <joachim@zuckarelli.de>

Description On import, the XML information is converted to a dataframe that reflects the hierarchical XML structure. Intuitive functions allow to navigate within this transparent XML data structure (without any knowledge of 'XPath'). 'flatXML' also provides tools to extract data from the XML into a flat dataframe that can be used to perform statistical operations. It also supports converting dataframes to XML.

License GPL-3

BugReports https://github.com/jsugarelli/flatxml/issues

URL https://github.com/jsugarelli/flatxml/

Repository CRAN

Encoding UTF-8

LazyData true

Imports RCurl, xml2, httr, crayon

RoxygenNote 7.1.1

NeedsCompilation no

Author Joachim Zuckarelli [aut, cre]

Date/Publication 2020-12-01 21:40:02 UTC

R topics documented:

`flatxml` *flatXML: Tools for Working with XML Files as R Dataframes*

Description

`flatxml` provides functions to easily deal with XML files. When parsing an XML document with `fxml_importXMLFlat`, `flatxml` produces a special dataframe that is \'flat\' by its very nature but contains all necessary information about the hierarchical structure of the underlying XML document (for details on the dataframe see the reference for the `fxml_importXMLFlat` function). `flatxml` offers a set of functions to work with this dataframe. Apart from representing the XML document in a dataframe structure, there is yet another way in which `flatxml` relates to dataframes: the `fxml_toDataFrame` and `fxml_toXML` functions can be used convert XML data to dataframes and vice versa.

Each XML element, for example `<tag attribute="some value">Here is some text</tag>` has certain characteristics that can be accessed via the `flatxml` interface functions, after an XML document has been imported with `fxml_importXMLFlat`. These characteristics are:

- *value*: The (text) value of the element, `"Here is some text"` in the example above
- *attributes*: The XML attributes of the element, `attribute` with its value `"some value"` in the example above
- *children*: The elements on the next lower hierarchical level
- *parent*: The element of the next higher hierarchical level, i.e. the element to which the current element is a child
- *siblings*: The elements on the same hierarchical level as the current element

Structure of the flatxml interface

The `flatxml` interface to access these characteristics follows a simple logic: For each of the characteristics there are typically three functions available:

- `fxml_has...()`: Determines if the current XML element has (at least one instance of) the characteristic
- `fxml_num...()`: Returns the number of the characteristics of the current XML (e.g. the number of children elements)
- `fxml_get...()`: Returns (the IDs of) the respective characteristics of the current XML element (e.g. the children of the current element)

Functions to access the characteristics of an XML element

For values:

- `fxml_hasValue`
- `fxml_getValue`

For attributes:

- `fxml_hasAttributes`
- `fxml_numAttributes`
- `fxml_getAttribute` (note: no plural 's'!)
- `fxml_getAttributesAll` (get all attributes instead of a specific one)

For children:

- `fxml_hasChildren`
- `fxml_numChildren`
- `fxml_getChildren`

For parents:

- `fxml_hasParent`
- `fxml_getParent`

For siblings:

- `fxml_hasSiblings`
- `fxml_numSiblings`
- `fxml_getSiblings`

Functions for searching in the XML document

- `fxml_findPath` (search anywhere in the path to an XML element)
- `fxml_findPathFull` (find an element based on its complete path)
- `fxml_findPathRoot` (search in the path to an XML element starting at the top element [root node])
- `fxml_findPathBottom` (search in the path to an XML element starting at the lowest hierarchical level)

Functions for converting between XML and dataframe

- `fxml_toDataFrame` (converts a (flattened) XML document to a dataframe)
- `fxml_toXML` (converts a dataframe to an XML document)

Other functions

- `fxml_getElement` (name on an XML element (the `tag` in `<tag>. . . </tag>`)
- `fxml_getUniqueElements` (unique XML elements in the document)
- `fxml_getElementInfo` (all relevant information on an XML element (children, siblings, etc.)
- `fxml_getDepthLevel` (level of an element in the hierarchy of the XML document)

`fxml_findPath` *Finding XML elements*

Description

Finds all XML elements in an XML document that lie on a certain path, regardless of where exactly the path is found in the XML document. Sub-elements (children) of the elements on the search path are returned, too.

Usage

```
fxml_findPath(xmlflat.df, path, attr.only = NULL, attr.not = NULL)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`path`	A character vector representing the path to be searched. Each element of the vector is a hierarchy level in the XML document. Example: `path = c("tag1", "tag2")`.
`attr.only`	A list of named vectors representing attribute/value combinations the XML elements on the search path must match. The name of an element in the list is the XML elment name to which the attribute belongs. The list element itself is a named vector. The vector's elements represent different attributes (= the names of the vector elements) and their values (= vector elements). Example: `attr.only = list(tag1 = c(attrib1 = "Value 1", attrib2 = "Value 2"), tag2 = c(attrib3 = "Value 3"))` will only find those elements which lie on a path that includes `<tag1 attrib1 = "Value 1" attrib2 = "Value 2"><tag2 attrib3 = "Value 3">`.
`attr.not`	A list of vectors representing attribute/value combinations the XML elements on the search path must not match to be included in the results. See argument `attr.only` for details on the composition.

Details

With `fxml_findPath()` it does not matter where exactly in the hierarchy of the XML document the path is found. If, for example, `path = c("tag1", "tag2")` then the element with full XML path `<xml><testdoc><tag1><tag2>` would be found, too.

Other `fxml_findPath...()` functions allow for different search modes:

- `fxml_findPathRoot`: Search for path from the root node of the XML document downwards. Sub-elements are returned, too.
- `fxml_findPathFull`: Search for exact path (always starting from the root node). No sub-elements returned, as they have a different path than the search path.
- `fxml_findPathBottom`: Search for path from the bottom of the element hierarchy in the XML document.

Value

The IDs (`xmlflat.df$elemid.`) of the XML elements that are located on the provided path. Sub-elements of the elements on the search path are returned, too. `NULL`, if no elements where found.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_findPathRoot`, `fxml_findPathFull`, `fxml_findPathBottom`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Find all XML elements with <data><record><field> in their XML path
path <- c("data", "record", "field")
fxml_findPath(xml.dataframe, path)

# Find only those XML elements with <data><record><field> in their XML path that have the
# "name" attribute of the <field> element set to "Sex"
path <- c("data", "record", "field")
fxml_findPath(xml.dataframe, path, attr.only = list(field = c(name = "Sex")))
```

fxml_findPathBottom *Finding XML elements*

Description

Finds all XML elements in an XML document that lie on a certain path. The path of the found elements must end with the provided search path.

Usage

```
fxml_findPathBottom(xmlflat.df, path, attr.only = NULL, attr.not = NULL)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`path`	A character vector representing the path to be searched. Each element of the vector is a hierarchy level in the XML document. Example: `path = c("tag1", "tag2")`.
`attr.only`	A list of named vectors representing attribute/value combinations the XML elements on the search path must match. The name of an element in the list is the XML elment name to which the attribute belongs. The list element itself is a named vector. The vector's elements represent different attributes (= the names of the vector elements) and their values (= vector elements). Example: `attr.only = list(tag1 = c(attrib1 = "Value 1", attrib2 = "Value 2"), tag2 = c(attrib3 = "Value 3"))` will only find those elements which lie on a path that includes `<tag1 attrib1 = "Value 1" attrib2 = "Value 2"><tag2 attrib3 = "Value 3">`.
`attr.not`	A list of vectors representing attribute/value combinations the XML elements on the search path must not match to be included in the results. See argument `attr.only` for details on the composition.

Details

With `fxml_findPathRoot()`, the search always starts at the bottom of the element hierarchy of the XML document. Only if the path of an elemends ends with the provided search path, it is returned as a result. If, for example, `path = c("tag1", "tag2")` then the element with full XML path `<tag1><tag2><tag3>` would not be found, only if search path were `c("tag2", "tag3")`.

Other `fxml_findPath...()` functions allow for different search modes:

- `fxml_findPath`: Search for path anywhere in the XML document (not necessarily starting at the root node). Sub-elements are returned, too.
- `fxml_findPathRoot`: Search for path from the root node of the XML document downwards. Sub-elements are returned, too.
- `fxml_findPathFull`: Search for exact path (always starting from the root node). No sub-elements returned, as they have a different path than the search path.

Value

The IDs (`xmlflat.df$elemid.`) of the XML elements that are located on the provided path. `NULL`, if no elements where found.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_findPath`, `fxml_findPathRoot`, `fxml_findPathFull`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Find all XML elements that have a path ending with <record><field>
path <- c("record", "field")
fxml_findPathBottom(xml.dataframe, path)

# Find all XML elements that have a path ending with <record><field>, but only
# those which have the "name" attribute of the <field> element set to "Sex"
path <- c("record", "field")
fxml_findPathBottom(xml.dataframe, path, attr.only = list(field = c(name = "Sex")))
```

`fxml_findPathFull` *Finding XML elements*

Description

Finds all XML elements in an XML document that lie on a certain path. The path of the found elements must match exactly the search path.

Usage

```
fxml_findPathFull(xmlflat.df, path, attr.only = NULL, attr.not = NULL)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`path`	A character vector representing the path to be searched. Each element of the vector is a hierarchy level in the XML document. Example: `path = c("tag1", "tag2")`.

`attr.only`	A list of named vectors representing attribute/value combinations the XML elements on the search path must match. The name of an element in the list is the XML elment name to which the attribute belongs. The list element itself is a named vector. The vector's elements represent different attributes (= the names of the vector elements) and their values (= vector elements). Example: `attr.only = list(tag1 = c(attrib1 = "Value 1", attrib2 = "Value 2"), tag2 = c(attrib3 = "Value 3"))` will only find those elements which lie on a path that includes `<tag1 attrib1 = "Value 1" attrib2 = "Value 2"><tag2 attrib3 = "Value 3">`.
`attr.not`	A list of vectors representing attribute/value combinations the XML elements on the search path must not match to be included in the results. See argument `attr.only` for details on the composition.

Details

With `fxml_findPathRoot()`, the search always starts at the root node of the XML document. Only if an element has exactly the same path as the search path, it is returned as a result. If, for example, `path = c("tag1", "tag2")` then the element with full XML path `<tag1><tag2><tag3>` would not be found, only if search path were `c("tag1", "tag2", "tag3")`.

Other `fxml_findPath...()` functions allow for different search modes:

- `fxml_findPath`: Search for path anywhere in the XML document (not necessarily starting at the root node). Sub-elements are returned, too.
- `fxml_findPathRoot`: Search for path from the root node of the XML document downwards. Sub-elements are returned, too.
- `fxml_findPathBottom`: Search for path from the bottom of the element hierarchy in the XML document.

Value

The IDs (`xmlflat.df$elemid.`) of the XML elements that are located on the provided path. Sub-elements of the elements on the search path are not returned as they have a different search path. `NULL`, if no elements where found.

Author(s)

Joachim Zuckarelli `<joachim@zuckarelli.de>`

See Also

`fxml_findPath`, `fxml_findPathRoot`, `fxml_findPathBottom`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)
```

```
# Find all XML elements that have the exact path <root><data><record>
path <- c("root", "data", "record")
fxml_findPathFull(xml.dataframe, path)
```

`fxml_findPathRoot` *Finding XML elements*

Description

Finds all XML elements in an XML document that lie on a certain path. Search starts from the root node of the XML document. Sub-elements (children) of the elements on the search path are returned, too.

Usage

```
fxml_findPathRoot(xmlflat.df, path, attr.only = NULL, attr.not = NULL)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`path`	A character vector representing the path to be searched. Each element of the vector is a hierarchy level in the XML document. Example: `path = c("tag1", "tag2")`.
`attr.only`	A list of named vectors representing attribute/value combinations the XML elements on the search path must match. The name of an element in the list is the XML elment name to which the attribute belongs. The list element itself is a named vector. The vector's elements represent different attributes (= the names of the vector elements) and their values (= vector elements). Example: `attr.only = list(tag1 = c(attrib1 = "Value 1", attrib2 = "Value 2"), tag2 = c(attrib3 = "Value 3"))` will only find those elements which lie on a path that includes `<tag1 attrib1 = "Value 1" attrib2 = "Value 2"><tag2 attrib3 = "Value 3">`.
`attr.not`	A list of vectors representing attribute/value combinations the XML elements on the search path must not match to be included in the results. See argument `attr.only` for details on the composition.

Details

With `fxml_findPathRoot()`, the search always starts at the root node of the XML document. If, for example, `path = c("tag1", "tag2")` then the element with full XML path `<xml><testdoc><tag1><tag2>` would not be found, only if search path were `c("xml", "testdoc", "tag1", "tag2")`

Other `fxml_findPath...()` functions allow for different search modes:

- `fxml_findPath`: Search for path anywhere in the XML document (not necessarily starting at the root node). Sub-elements are returned, too.
- `fxml_findPathFull`: Search for exact path (always starting from the root node). No sub-elements returned, as they have a different path than the search path.

- `fxml_findPathBottom`: Search for path from the bottom of the element hierarchy in the XML document.

Value

The IDs (`xmlflat.df$elemid.`) of the XML elements that are located on the provided path. Sub-elements of the elements on the search path are returned, too. `NULL`, if no elements where found.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_findPath`, `fxml_findPathFull`, `fxml_findPathBottom`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Find all XML elements that have a path starting with <root><data><record><field>
path <- c("root", "data", "record", "field")
fxml_findPathRoot(xml.dataframe, path)

# Find all XML elements that have a path starting with <root><data><record><field>, but only
# those which have the "name" attribute of the <field> element set to "Sex"
path <- c("root", "data", "record", "field")
fxml_findPathRoot(xml.dataframe, path, attr.only = list(field = c(name = "Sex")))
```

`fxml_getAttribute` *Attributes of an XML element*

Description

Returns the value of a specific attribute of an XML element.

Usage

```
fxml_getAttribute(xmlflat.df, elemid, attrib.name)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.
`attrib.name`	Name of the attribute.

Value

The value of attribute `attrib.name` of the XML element with ID `elemid`. If the attribute is not existing, an error message is shown.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_hasAttributes`, `fxml_numAttributes`, `fxml_getAttributesAll`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Read the value of attribute "name" from the XML element with ID 4 (xml.dataframe$elemid. == 4)
fxml_getAttribute(xml.dataframe, 4, "name")
```

`fxml_getAttributesAll` *Attributes of an XML element*

Description

Returns all attributes of an XML element and their respective values.

Usage

```
fxml_getAttributesAll(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

A named vector containing the attribute values of all attributes of the XML element with ID `elemid`. The names of the vector are the names of the attributes. Returns `NULL` if the element has no attributes at all.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

fxml_hasAttributes, fxml_numAttributes, fxml_getAttribute

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get all attribute of the XML element with ID 4 (xml.dataframe$elemid. ==  4)
fxml_getAttributesAll(xml.dataframe, 4)
```

fxml_getChildren *Children of an XML element*

Description

Returns the children of an XML element.

Usage

```
fxml_getChildren(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

The IDs (xmlflat.df$elemid.) of the children of the XML element with ID elemid. If no children exist, NULL is returned.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

See Also

fxml_hasChildren, fxml_numChildren

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get all the children (sub-elements) of the XML element with ID 4 (xml.dataframe$elemid. ==  4)
fxml_hasChildren(xml.dataframe, 4)
```

`fxml_getDepthLevel` *Handling flat XML files*

Description

Hierarchical position of an XML element

Usage

```
fxml_getDepthLevel(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

The number of the hierarchy level of the XML element with ID `elemid`. The root node of the XML data has hierarchy level 1.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Determine hierarchy level of XML element with ID 3 (xml.dataframe$elemid. ==  3)
fxml_getDepthLevel(xml.dataframe, 3)
```

fxml_getElement *Handling flat XML files*

Description

Returns the element name of an XML element.

Usage

```
fxml_getElement(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

Name of the element identified by the ID (xmlflat.df$elemid.) elemid.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

See Also

fxml_getUniqueElements

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get the XML element with ID 3 (xml.dataframe$elemid. ==  3)
fxml_getElement(xml.dataframe, 3)
```

`fxml_getElementInfo` *Handling flat XML files*

Description

Returns summary information on an XML element.

Usage

```
fxml_getElementInfo(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

A list with the following elements:

- `value`: The value of the XML element; return value of the `fxml_getValue` function.
- `path`: A vector representing the path from the root element of the XML element document to the current element. Each XML element on the path is represented by a element of the vector. The vector elements are the names of the XML elements on the path.
- `depth.level`: The depth level (hierarchy level) of the XML element; return value of the `fxml_getDepthLevel` function.
- `attributes`: A named vector with the attributes of the XML element (vector elements are the attributes' values, names of the vector elements are the attributes' names; return value of the `fxml_getAttributesAll` function.
- `parent`: The parent of the XML element; return value of the `fxml_getParent` function.
- `children`: The children of the XML element; return value of the `fxml_getChildren` function.
- `siblings`: The siblings of the XML element; return value of the `fxml_getSiblings` function.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_getElement`, `fxml_getValue`, `fxml_getDepthLevel`, `fxml_getAttribute`, `fxml_getChildren`, `fxml_getParent`, `fxml_getSiblings`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get all relevant information on the XML element with ID 4 (xml.dataframe$elemid. ==  4)
fxml_getElementInfo(xml.dataframe, 4)
```

`fxml_getParent` *Parent of an XML element*

Description

Returns the parent of an XML element.

Usage

```
fxml_getParent(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

The ID (`xmlflat.df$elemid.`) of the parent node of the XML element with ID `elemid`. If no parent exists (because XML node `elemid` is the root node of the XML document) then `NULL` is returned.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_hasParent`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get the ID of the parent element of the XML element with ID 4 (xml.dataframe$elemid. ==  4)
fxml_getParent(xml.dataframe, 4)
```

fxml_getSiblings *Siblings of an XML element*

Description

Returns the siblings of an XML element, i.e. the elements on the same hierarchical level.

Usage

```
fxml_getSiblings(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

The IDs (xmlflat.df$elemid.) of the siblings of the XML element with ID elemid. If no siblings exist, NULL is returned.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

See Also

fxml_hasSiblings, fxml_getSiblings

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get all the siblings (elements on the same hierarchy level) of the XML element with ID 4
# (xml.dataframe$elemid. ==  4)
fxml_getSiblings(xml.dataframe, 4)
```

fxml_getUniqueElements

Handling flat XML files

Description

Returns the unique XML elements included in an XML document.

Usage

```
fxml_getUniqueElements(xmlflat.df)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.

Value

A vector with all the names of the elements included in the XML document `xmlflat.df`. Every tag is only returned once, even if it occurs multiple times in the document. The return vector is empty (`NULL`) if no elements exist.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_getElement`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Identify the unique XML elements
fxml_getUniqueElements(xml.dataframe)
```

`fxml_getValue` *Value of an XML element*

Description

Returns the value of an XML element.

Usage

```
fxml_getValue(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

The value of the XML element with ID `elemid`. `NA` is returned if the element has no value.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_hasValue`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Get the value of the XML element with ID 4 (xml.dataframe$elemid. ==  4)
fxml_hasValue(xml.dataframe, 4)
```

fxml_hasAttributes *Attributes of XML elements*

Description

Determines if an XML element has any attributes.

Usage

```
fxml_hasAttributes(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

TRUE if the the XML element with ID elemid has at least one attribute, FALSE otherwise.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

See Also

fxml_getAttribute, fxml_numAttributes, fxml_getAttributesAll

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Check if the XML element with ID 4 (xml.dataframe$elemid. ==  4) has any attributes
fxml_hasAttributes(xml.dataframe, 4)
```

`fxml_hasChildren` *Children of an XML element*

Description

Determines if an XML element has any children.

Usage

```
fxml_hasChildren(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

`TRUE`, if the the XML element with ID `elemid` has at least one child, `FALSE` otherwise.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_numChildren`, `fxml_getChildren`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Check, if the XML element with ID 4 (xml.dataframe$elemid. ==  4) has any
# children (sub-elements)
fxml_hasChildren(xml.dataframe, 4)
```

`fxml_hasParent` *Parent of an XML element*

Description

Determines, if an XML element has a parent element.

Usage

```
fxml_hasParent(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

`TRUE`, if a parent element for the XML element with ID `elemid` exists, `FALSE` otherwise (which would mean that the XML element is the root node of the XML document).

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_getParent`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Check if the XML element with ID 4 (xml.dataframe$elemid. ==  4) has a parent element
fxml_hasParent(xml.dataframe, 4)
```

fxml_hasSiblings *Siblings of an XML element*

Description

Determines if an XML element has any siblings, i.e. elements on the same hierarchical level.

Usage

```
fxml_hasSiblings(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

TRUE, if the the XML element with ID elemid has at least one sibling, FALSE otherwise.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

#' @seealso fxml_numSiblings, fxml_getSiblings

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Check if XML element with ID 4 (xml.dataframe$elemid. ==  4) has any siblings
# (elements on the same hierarchy level)
fxml_hasSiblings(xml.dataframe, 4)
```

`fxml_hasValue` *Value of an XML element*

Description

Determines if an XML element carries a value.

Usage

```
fxml_hasValue(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

`TRUE` if the XML element has a value (not being equal to `NA`), `FALSE` otherwise.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_getValue`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Check if element with ID 4 (xml.dataframe$elemid. ==  4) carries a value
fxml_hasValue(xml.dataframe, 4)
```

`fxml_importXMLFlat` *Handling flat XML files*

Description

Reads an XML document into a flat dataframe structure.

Usage

```
fxml_importXMLFlat(path)
```

Arguments

`path`	Path to the XML document. Can be either a local path or a URL.

Details

The XML document is parsed and stored in a dataframe structure (flat XML). The first four columns of a flat XML dataframe are standard columns. Their names all end with a dot. These columns are:

- `elem.`: The element identifier of the current XML element (without the tag delimiters < and >).
- `elemid.`: A unique, ascending numerical ID for each XML element. The first XML element is assigned 1 as its ID. This ID is used by many of the `flatxml` functions.
- `attr.`: Name of an attribute. For each attribute of an XML element the dataframe will have an additional row.
- `value.`: The value of either the attribute (if `attr.` is not `NA`) or the element itself (if `attr.` is `NA`). `value.` is `NA`, if the element has no value.

The columns after these four standard columns represent the 'path' to the current element, starting from the root element of the XML document in column 5 all the way down to the current element. The number of columns of the dataframe is therefore determined by the depth of the hierarchical structure of the XML document. In this dataframe representation, the hierarchical structure of the XML document becomes very easy to understand. All `flatxml` functions work with this flat XML dataframe.

If an XML element has N attributes it is represented by (N+1) rows in the flat XML dataframe: one row for the value (with `dataframe$value.` being `NA` if the element has no value) and one for each attribute. In the attribute rows, the names of the attributes are stored in the `attr.` field, their respecitive values in the `value.` field. Even if there are multiple rows for one XML element, the `elem.` and `elemid.` fields still have the same value in all rows (because the rows belong to the same XML element).

Value

A dataframe containing the XML document in a flat structure. See the Details section for more information on its structure.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

Examples

```
# Load example file with population data from United Nations Statistics Division
example <- system.file("worldpopulation.xml", package="flatxml")
# Create flat dataframe from XML
xml.dataframe <- fxml_importXMLFlat(example)
```

`fxml_numAttributes` *Attributes of XML elements*

Description

Determines the number of attributes of an XML element.

Usage

```
fxml_numAttributes(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

The number of attributes of the XML element with ID `elemid`.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

See Also

`fxml_hasAttributes`, `fxml_getAttribute`, `fxml_getAttributesAll`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Determine the number of attributes of the XML element with ID 4 (xml.dataframe$elemid. == 4)
fxml_numAttributes(xml.dataframe, 4)
```

fxml_numChildren *Children of an XML element*

Description

Determines the number of children of an XML element.

Usage

```
fxml_numChildren(xmlflat.df, elemid)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`elemid`	The ID of the XML element. The ID is the value of the `elemid.` field in the flat XML dataframe.

Value

The number of children of the XML element with ID `elemid`.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_hasChildren`, `fxml_getChildren`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Determine the number of children (sub-elements) of the XML element with ID 4
# (xml.dataframe$elemid. ==  4)
fxml_numChildren(xml.dataframe, 4)
```

fxml_numSiblings *Siblings of an XML element*

Description

Determines the number of siblings of an XML element, i.e. elements on the same hierarchical level.

Usage

```
fxml_numSiblings(xmlflat.df, elemid)
```

Arguments

xmlflat.df	A flat XML dataframe created with fxml_importXMLFlat.
elemid	The ID of the XML element. The ID is the value of the elemid. field in the flat XML dataframe.

Value

The number of siblings of the XML element with ID elemid.

Author(s)

Joachim Zuckarelli <joachim@zuckarelli.de>

#' @seealso fxml_hasSiblings, fxml_getSiblings

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Determine the number of siblings (elements on the same hierarchy level) of the XML element
# with ID 4 (xml.dataframe$elemid. ==  4)
fxml_numSiblings(xml.dataframe, 4)
```

`fxml_toDataFrame` *Converting between XML and dataframes*

Description

Converts an XML document to a dataframe.

Usage

```
fxml_toDataFrame(
  xmlflat.df,
  siblings.of,
  same.tag = TRUE,
  attr.only = NULL,
  attr.not = NULL,
  elem.or.attr = "elem",
  col.attr = "",
  include.fields = NULL,
  exclude.fields = NULL
)
```

Arguments

`xmlflat.df`	A flat XML dataframe created with `fxml_importXMLFlat`.
`siblings.of`	ID of one of the XML elements that contain the data records. All data records need to be on the same hierarchical level as the XML element with this ID.
`same.tag`	If `TRUE`, only elements of the same type (`xmlflat.df$elem.`) as the element `sibling.of` are considered as data records. If `FALSE`, *all* elements on the same hierarchical level as the element `sibling.of` are considered to be data records.
`attr.only`	A list of named vectors representing attribute/value combinations the data records must match. The name of an element in the list is the XML element name to which the attribute belongs. The list element itself is a named vector. The vector's elements represent different attributes (= the names of the vector elements) and their values (= vector elements). Example: `attr.only = list(tag1 = c(attrib1 = "Value 1", attrib2 = "Value 2"), tag2 = c(attrib3 = "Value 3"))` will only include `tag1` elements of the form `<tag1 attrib1 = "Value 1" attrib2 = "Value 2">` and `tag2` elements of the form `<tag2 attrib3 = "Value 3">` as data records.
`attr.not`	A list of vectors representing attribute/value combinations the XML elements must *not* match to be considered as data records. See argument `attr.only` for details.
`elem.or.attr`	Either `"elem"` or `"attr"`. Defines, if the names of the record fields (columns in the dataframe) are represented by the names (tags) of the respective XML elements (the children of the elements on the same level as `siblings.of`) (`"elem"`) or if the field names are given by some attribute of those tags (`"attr"`).

`col.attr`	If `elem.or.attr` is `"attr"` then `col.attr` specifies the name of the attribute that gives the record field / column names.
`include.fields`	A character vector with the names of the fields that are to be included in the result dataframe. By default, all fields from the XML document are included.
`exclude.fields`	A character vector with the names of the fields that should be excluded in the result dataframe. By default, no fields from the XML document are excluded.

Details

Data that can be read in are either represented in this way:

```
<record>
<field1>Value of field1</field1>
<field2>Value of field2</field2>
<field3>Value of field3</field3>
</record>
...
```

In this case `elem.or.attr` would need to be `"elem"` because the field names of the data records (`field1`, `field2`, `field3`) are the names of the elements.

Or, the XML data could also look like this:

```
<record>
<column name="field1">Value of field1</column>
<column name="field2">Value of field2</column>
<column name="field3">Value of field3</column>
</record>
...
```

Here, the names of the fields are attributes, so `elem.or.attr` would need to be `"attr"` and `col.attr` would be set to `"name"`, so `fxml_toDataframe()` knows where to look for the field/column names.

In any case, `siblings.of` would be the ID (`xmlflat.df$elemid.`) of one of the `<record>` elements.

Value

A dataframe with the data read in from the XML document.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_importXMLFlat`, `fxml_toXML`

Examples

```
# Load example file with population data from United Nations Statistics Division
# and create flat dataframe
example <- system.file("worldpopulation.xml", package="flatxml")
xml.dataframe <- fxml_importXMLFlat(example)

# Extract the data out of the XML document. The data records are on the same hierarchical level
# as element with ID 3 (xml.dataframe$elemid. ==  3).
# The field names are given in the "name" attribute of the children elements of element no. 3
# and its siblings
population.df <- fxml_toDataFrame(xml.dataframe, siblings.of=3, elem.or.attr="attr",
col.attr="name")
# Exclude the "Value Footnote" field from the returned dataframe
population.df <- fxml_toDataFrame(xml.dataframe, siblings.of=3, elem.or.attr="attr",
col.attr="name", exclude.fields=c("Value Footnote"))


# Load example file with soccer world cup data (data from
# https://www.fifa.com/fifa-tournaments/statistics-and-records/worldcup/index.html)
# and create flat dataframe
example2 <- system.file("soccer.xml", package="flatxml")
xml.dataframe2 <- fxml_importXMLFlat(example2)

# Extract the data out of the XML document. The data records are on the same hierarchical level
# as element with ID 3 (xml.dataframe$elemid. ==  3). #' # The field names are given as the name
# of the children elements of element no. 3 and its siblings.
worldcups.df <- fxml_toDataFrame(xml.dataframe2, siblings.of=3, elem.or.attr="elem")
```

`fxml_toXML` *Converting between XML and dataframes*

Description

Converts a dataframe to XML.

Usage

```
fxml_toXML(
  df,
  filename = NULL,
  element.tag = "record",
  indent = "\t",
  line.break = "\n",
  return.xml = FALSE
)
```

Arguments

`df`	The dataframe to be converted (also works with tibbles and the like)
`filename`	Name of the file to which the XML will be saved; default is `NULL` meaning no file is produced
`element.tag`	The tag name of the XML element that will carry the data (see example)
`indent`	Character(s) used for indentation to make the XML prettier; tabulator (`"\t"`) by default ("" will lead to no indentation)
`line.break`	Character(s) that is written at the end of each line of the XML (line break `"\n"` by default)
`return.xml`	If `TRUE`, the XML will be returned by the function; so you can decide if you want the function write a file, or just return the XML code, or both.

Value

If `return.xml == TRUE` the XML code is returned. If `filename` is not `NULL` then the XML is (additionally) written to the specified file.

Author(s)

Joachim Zuckarelli <`joachim@zuckarelli.de`>

See Also

`fxml_toDataFrame`

Examples

```
mydata<-data.frame(list(var1 = c("a", "b", "c"), var2 = c(1,2,3)))
fxml_toXML(mydata, return.xml = TRUE)
```

Index