Package ‘pepr’

October 16, 2020

Type    Package
Title   Reading Portable Encapsulated Projects
Version 0.4.0
Date    2020-10-15
Maintainer Michal Stolarczyk <michal@virginia.edu>
Description A PEP, or Portable Encapsulated Project, is a dataset that subscribes to the PEP structure for organizing metadata. It is written using a simple YAML + CSV format, it is your one-stop solution to metadata management across data analysis environments. This package reads this standardized project configuration structure into R.

Imports yaml, stringr, pryr, data.table, methods
Suggests knitr, testthat, rmarkdown
VignetteBuilder knitr
License BSD_2_clause + file LICENSE

BugReports https://github.com/pepkit/pepr
RoxygenNote 7.1.0
NeedsCompilation no

Author Nathan Sheffield [aut, cph],
       Michal Stolarczyk [aut, cre]

Repository CRAN
Date/Publication 2020-10-16 06:10:02 UTC

R topics documented:

.appendAttrs .......................................................... 2
.applyAmendments .................................................. 3
.applyImports .......................................................... 3
.checkSection .......................................................... 4
.deriveAttrs ............................................................ 4
.duplicateAttrs ........................................................ 5
Append constant attributes across all the samples

Append constant attributes across all the samples

Usage

.appendAttrs(.Object)
Arguments

- .Object an object of Project-class

Value

- an object of Project-class

---

**Arguments**

- .applyAmendments

**Value**

- possibly updated config

---

**Arguments**

- .applyImports

**Value**

- config data enriched in imported sections, if imports existed in the input
.checkSection  
*Check for a section existence in a nested list*

**Description**
Check for a section existence in a nested list

**Usage**
```
.checkSection(object, sectionNames)
```

**Arguments**
- `object` list to inspect
- `sectionNames` vector or characters with section names to check for

**Value**
logical indicating whether the sections were found in the list

**Examples**
```r
l = list(a=list(b="test"))
.checkSection(l,c("a","b"))
.checkSection(l,c("c","b"))
```

---

.deriveAttrs  
*Derive attributes*

**Description**
Derive attributes

**Usage**
```
.deriveAttrs(.Object)
```

**Arguments**
- `.Object` an object of "Project"

**Value**
an object of "Project"
**.duplicateAttrs**

*Duplicate a selected attribute across all the samples*

**Description**

Duplicate a selected attribute across all the samples

**Usage**

```
.duplicateAttrs(.Object)
```

**Arguments**

`.Object` an object of "Project"

**Value**

an object of "Project"

---

**.expandList**

*Recursively try to expand list of strings*

**Description**

Recursively try to expand list of strings

**Usage**

```
.expandList(x)
```

**Arguments**

`x` list, possibly of strings that are paths to expand

**Value**

list of strings with paths expanded

**Examples**

```
x = list(a=list(b=list(c="~/test.txt")))
.expandList(x)
```
**.expandPath**  
**Expand system path**

**Description**
This function expands system paths (the non-absolute paths become absolute) and replaces the environment variables (e.g, `${HOME}`) with their values.

**Usage**
```
.expandPath(path)
```

**Arguments**
- **path**: file path to expand. Potentially any string

**Details**
Most importantly strings that are not system paths are returned untouched.

**Value**
Expanded path or untouched string

**Examples**
```
string = "https://www.r-project.org/
.expandPath(string)
path = "$HOME/my/path/string.txt"
.expandPath(path)
```

**.getSubscript**  
**Get list subscript**

**Description**
Based on available list element names and subscript value determine index of the element requested.

**Usage**
```
.getSubscript(lst, i)
```

**Arguments**
- **lst**: list to search subscript for
- **i**: character or numeric to determine final list index
**Value**

numeric index of the requested element in the list

**Examples**

```python
l = list(a="a", b="b")
.getSubscript(l, 1) == .getSubscript(l, "a")
```

**.implyAttrs**

**Imply attributes**

**Description**

Imply attributes

**Usage**

```
.implyAttrs(.Object)
```

**Arguments**

- `.Object` an object of "Project"

**Value**

an object of "Project"

**.inferProjectName**

**Infer project name**

**Description**

Based on dedicated config section or PEP enclosing dir

**Usage**

```
.inferProjectName(cfg, filename)
```

**Arguments**

- `cfg` config data
- `filename` path to the config file

**Value**

string project name
.isAbsolute

Determine whether a path is absolute.

Description
Determine whether a path is absolute.

Usage
.isAbsolute(path)

Arguments
path The path to check for seeming absolute-ness.

Value
Flag indicating whether the path appears to be absolute.

.listifyDF

Listify data frame columns

Description
This function turns each data frame column into a list, so that its cells can contain multiple elements.

Usage
.listifyDF(DF)

Arguments
DF an object of class data.frame

Value
an object of class data.frame

Examples
dataFrame=mtcars
listifiedDataFrame=.listifyDF(dataFrame)
.loadConfig

Description
Loads a PEP config file

Usage
.loadConfig(filename = NULL, amendments = NULL)

Arguments
filename file path to config file
amendments amendments to activate

See Also
https://pep.databio.org/

.loadSampleAnnotation

Description
Read sample annotation from disk

Usage
.loadSampleAnnotation(.Object)

Arguments
.Object an object of "Project"

Value
an object of "Project"
.loadSubsampleAnnotation

Load single subsample annotation

Description
Load single subsample annotation

Usage
.loadSubsampleAnnotation(.Object, path)

Arguments

,Object an object of "Project"

path string, a path to the subsample table to read and incorporate

Value
an object of "Project"

.makeAbsPath

Create an absolute path from a primary target and a parent candidate.

Description
Create an absolute path from a primary target and a parent candidate.

Usage
.makeAbsPath(perhapsRelative, parent)

Arguments

,perhapsRelative Path to primary target directory.

,parent a path to parent folder to use if target isn’t absolute.

Value
Target itself if already absolute, else target nested within parent.
.matchesAndRegexes

Description

Create a list of matched files in the system and unmatched regular expressions

Usage

.matchesAndRegexes(rgx)

Arguments

rgx

string to expand in the system

Value

a list of all the elements after possible expansion

.mergeAttrs

Merge samples defined in sample table with ones in subsample table(s)

Description

Merge samples defined in sample table with ones in subsample table(s)

Usage

.mergeAttrs(.Object)

Arguments

.Object

an object of "Project"

Value

an object of "Project"
.modifySamples

**Description**
Perform all the sample attribute modifications

**Usage**
```
.modifySamples(object)
```

**Arguments**
- **object**
  an object of "Project"

**Value**
modified Project object

---

.printNestedList

**Description**
Prints a nested list in a way that looks nice

**Usage**
```
.printNestedList(lst, level = 0)
```

**Arguments**
- **lst**
  list object to print
- **level**
  the indentation level

**Details**
Useful for displaying the config of a PEP

**Examples**
```
projectConfig = system.file("extdata",
  "example_peps-master",
  "example_basic",
  "project_config.yaml",
  package = "pepr")
p = Project(file = projectConfig)
.printNestedList(config(p), level=2)
```
.reformat

Description
Check config spec version and reformat if needed

Usage
.reformat(object)

Arguments
object an object of "Config"

Value
an object of "Config"

.removeAttrs

Description
Remove attributes across all the samples

Usage
.removeAttrs(.Object)

Arguments
.Object an object of "Project"

Value
an object of "Project"
activateAmendments

activateAmendments

Description

This method switches between the amendments within the "Project" object.

Usage

activateAmendments(.Object, amendments)

## S4 method for signature 'Project, character'
activateAmendments(.Object, amendments)

Arguments

.Object an object of class "Project"

.amendments character with the amendment name

.strformat

Format a string like python’s format method

Description

Given a string with environment variables (encoded like ${VAR} or $VAR), and other variables (encoded like {VAR}) this function will substitute both of these and return the formatted string, like the Python str.format() method. Other variables are populated from a list of arguments. Additionally, if the string is a non-absolute path, it will be expanded.

Usage

.strformat(string, args, parent = NULL)

Arguments

string String with variables encoded
args named list of arguments to use to populate the string
parent a directory that will be used to make the path absolute

Examples

.strformat("/{VAR1}{VAR2}_file", list(VAR1="hi", VAR2="hello"))
.strformat("$/HOME/{VAR1}{VAR2}_file", list(VAR1="hi", VAR2="hello"))
checkSection

Details
To check what are the amendments names call listAmendments(p), where p is the object of "Project" class

Methods (by class)

- .Object = Project, amendments = character: activate amendments in a "Project" object

Examples

projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package = "pepr")
p = Project(file = projectConfig)
availAmendments = listAmendments(p)
activateAmendments(p,availAmendments[1])

checkSection
Check for existence of a section in the Project config

Description
This function checks for the section/nested sections in the config YAML file. Returns TRUE if it exist(s) or FALSE otherwise.

Usage
checkSection(object, sectionNames)

## S4 method for signature 'Config'
checkSection(object, sectionNames)

Arguments

- object: object of "Config"
- sectionNames: the name of the section or names of the nested sections to look for

Details
Element indices can be used instead of the actual names, see Examples.

Value

a logical indicating whether the section exists
Methods (by class)

- Config: checks for existence of a section in "Config" objects

Examples

```r
projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package="pepr")
p=Project(projectConfig)
checkSection(config(p),sectionNames = c("amendments","newLib"))
checkSection(config(p),sectionNames = c("amendments",1))
```

Description

This method can be used to view the config slot of the "Project" class

Usage

```r
config(object)
```

## S4 method for signature 'Project'

```r
config(object)
```

Arguments

- object: an object of "Project"

Value

project config

Methods (by class)

- Project: Extract "Project" of the object of "Project"

Examples

```r
projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package="pepr")
p=Project(projectConfig)
config(p)
```
Config-class

Config objects and specialized list objects and expand string attributes

Description

Config objects are used with the "Project" object

Usage

Config(file, amendments = NULL)

Arguments

file a character with project configuration yaml file
amendments a character with the amendments names to be activated

Value

an object of "Config" class

Examples

projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package="pepr")
c=Config(projectConfig)

fetchSamples

Collect samples fulfilling the specified requirements

Description

This function collects the samples from a data.table-class object that fulfill the requirements of an attribute attr specified with the fun argument

Usage

fetchSamples(samples, attr = NULL, func = NULL, action = "include")

Arguments

samples an object of data.table-class class
attr a string specifying a column in the samples
func an anonymous function, see Details for more information
action a string (either include or exclude) that specifies whether the function should select the row or exclude it.
getSample

Details

The anonymous function provided in the func argument has to return an integer that indicate the rows that the action should be performed on. Core expressions which are most useful to implement the anonymous function are:

- `which` with inequality signs: `==`, `>`, `<
- `grep`

Examples

```r
projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package="pepr")
p = Project(projectConfig)
s = sampleTable(p)
fetchSamples(s,attr = "sample_name", func=function(x){ which(x=="pig_0h") },action="include")
fetchSamples(s,attr = "sample_name", func=function(x){ which(x=="pig_0h") },action="exclude")
fetchSamples(s,attr = "sample_name", func=function(x){ grep("pig_",x) },action="include")
```

getSample

Extract samples

Description

This method extracts the samples

Usage

```r
getSample(.Object, sampleName)

## S4 method for signature 'Project,character'
getSample(.Object, sampleName)
```

Arguments

- .Object: An object of Project class
- sampleName: character the name of the sample

Value

data.table one row data table with the sample associated metadata

Methods (by class)

- .Object = Project, sampleName = character: extracts the sample from the "Project" object
getSubsample

Examples

```r
projectConfig = system.file(
  "extdata",
  "example_peps-master",
  "example_basic",
  "project_config.yaml",
  package = "pepr"
)
p = Project(projectConfig)
sampleName = "frog_1"
getSample(p, sampleName)
```

description

This method extracts the samples

Usage

```r
getSubsample(.Object, sampleName, subsampleName)
```

## S4 method for signature 'Project,character,character'

getSubsample(.Object, sampleName, subsampleName)

Arguments

- **.Object**: An object of Project class
- **sampleName**: character the name of the sample
- **subsampleName**: character the name of the subsample

Value

- data.table: one row data table with the subsample associated metadata

Methods (by class)

- **.Object = Project, sampleName = character, subsampleName = character**: extracts the subsamples from the "Project" object
Examples

```r
projectConfig = system.file(
    "extdata",
    "example_peps-master",
    "example_subtable1",
    "project_config.yaml",
    package = "pepr"
)
p = Project(projectConfig)
sampleName = "frog_1"
subsampleName = "sub_a"
getSubsample(p, sampleName, subsampleName)
```

---

listAmendments  List amendments

Description

Lists available amendments within a "Project" object.

Usage

```r
listAmendments(.Object)
```

## S4 method for signature 'Project'
listAmendments(.Object)

Arguments

`.Object`  an object of "Project"

Details

The amendments can be activated by passing their names to the `activateAmendments` method

Value

names of the available amendments

Methods (by class)

- Project: list amendments in a "Project" object
Examples

```r
projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package = "pepr")
p = Project(file = projectConfig)
availAmendemtns = listAmendments(p)
```

Description

Make selected sections absolute using config path

Usage

```r
makeSectionsAbsolute(object, sections, cfgPath)
```

## S4 method for signature 'Config,character,character'

```r
makeSectionsAbsolute(object, sections, cfgPath)
```

Arguments

- `object` "Config"
- `sections` character set of sections to make absolute
- `cfgPath` character absolute path to the config YAML file

Value

Config with selected sections made absolute

Methods (by class)

- `object = Config, sections = character, cfgPath = character`: Make selected sections absolute using config path from "Project"
Project Description

Package documentation

Author(s)

Michal Stolarczyk, Nathan Sheffield

References

GitHub: https://github.com/pepkit/pepr, Documentation: https://code.databio.org/pepr/

Project

The constructor of a class representing a Portable Encapsulated Project

Description

This is a helper that creates the project with empty samples and config slots

Usage

Project(file = NULL, amendments = NULL)

Arguments

file a character with project configuration yaml file

amendments a character with the amendments names to be activated

Examples

projectConfig = system.file("extdata", "example_peps-master", "example_amendments1", "project_config.yaml", package="pepr")
p=Project(projectConfig)
**Project-class**

*Portable Encapsulated Project object*

**Description**

Provides an in-memory representation and functions to access project configuration and sample annotation values for a PEP.

**Details**

Can be created with the constructor: "Project"

**Slots**

- `file` character vector path to config file on disk.
- `samples` a data table object holding the sample metadata
- `config` a list object holding contents of the config file

**sampleTable**

*View samples in the objects of "Project"

**Description**

This method can be used to view the samples slot of the "Project" class

**Usage**

```r
sampleTable(object)
```

```r
## S4 method for signature 'Project'
sampleTable(object)
```

**Arguments**

- `object` an object of "Project"

**Value**

A data.table with the with metadata about samples

**Methods (by class)**

- Project: extract sample table from a "Project"
Examples

```
projectConfig = system.file("extdata", "example_peps-master", 
"example_amendments1", "project_config.yaml", package="pepr")
p=Project(projectConfig)
sampleTable(p)
```

## Description

You can subset `Config` by identifier or by position using the `\[, \[\]` or `$` operator. The string will be expanded if it’s a path.

## Usage

```
# S4 method for signature 'Config'
x[i]

# S4 method for signature 'Config'
x[[i]]

# S4 method for signature 'Config'
x$name
```

## Arguments

- **x**: a "Config" object.
- **i**: position of the identifier or the name of the identifier itself.
- **name**: name of the element to access.

## Value

An element held in "Config" object

## Examples

```
projectConfig = system.file("extdata", "example_peps-master", 
"example_amendments1", "project_config.yaml", package="pepr")
c=Config(projectConfig)
c[[2]]
c[2]
c[["sample_table"]]
c$sample_table
```