Package 'bdchecks'

October 12, 2022

Title Biodiversity Data Checks

Description Supplies a Shiny app and a set of functions to perform and managing data checks for biodiversity data.

Version 0.1.7

Date 2019-02-07

License GPL-3 | file LICENSE

URL https://github.com/bd-R/bdchecks

BugReports https://github.com/bd-R/bdchecks/issues

Depends R (>= 2.10)

Imports bdDwC, DT, data.table, finch, methods, knitr, rgbif, shiny, shinyBS, shinydashboard, shinyjs, spocc, yaml

LazyData true

RoxygenNote 6.1.1

Encoding UTF-8

Suggests testthat, covr

NeedsCompilation no

Author Povilas Gibas [aut, cre] (<https://orcid.org/0000-0001-5311-6021>), Tomer Gueta [aut] (<https://orcid.org/0000-0003-1557-8596>), Vijay Barve [aut] (<https://orcid.org/0000-0002-4852-2567>), Thiloshon Nagarajah [aut], Yohay Carmel [aut] (<https://orcid.org/0000-0002-5883-0184>)

Maintainer Povilas Gibas <povilasgibas@gmail.com>

Repository CRAN

Date/Publication 2019-02-18 15:00:07 UTC

R topics documented:

createDCclassMain *Create a data check object*

Description

Create a data check object from a given YAML file

Usage

```
createDCclassMain(DCyaml)
```

Arguments

DCyaml	Data check entry as a list (originally imported as a YAML file)

Value

Data check object

createDCclassMeta *Create a data check metadata object*

Description

Create a data check metadata object from a given slot in a data check list object

Usage

```
createDCclassMeta(DCmeta)
```

Arguments

DCmeta	Data check metadata entry as a list.

Value

Data check metadata object

dataBats *Australian bats biodiversity data*

Description

A dataset of Australian bats (*chiroptera*) (1000 records) downloaded using `rgbif` package.

dataCheck-class *Create Data Check Class*

Description

Create Data Check Class

Show method for dataCheck objects

Usage

```
## S4 method for signature 'dataCheck'
show(object)
```

Arguments

object	a dataCheck object

Slots

`name` of a data check

`meta` meta-data for a data check of a dataCheckMeta class

`input` options for a data check

`func` expression to execute

dataCheckFlag-class *Combined Data Checks Class*

Description

Combined Data Checks Class

Show method for dataCheckFlag objects

Usage

```
## S4 method for signature 'dataCheckFlag'
show(object)
```

Arguments

object	a dataCheckFlag object

Slots

`DC` names of performed data checks

`flags` list of performed data checks in a dataCheckFlag_SINGLE class

`dataOrig` original data before data checks

`dataMod` modified data after data checks

`dataCheckFlag_SINGLE-class` *Single Data Check Flag Class*

Description

Single Data Check Flag Class

Slots

`name` of performed data check

`target` column for performed data check

`flag` class

`result` logical vector

`dataCheckMeta-class` *Data Check Meta-Data Class*

Description

Data Check Meta-Data Class

Slots

`description` of data check

`flags` for specific data check

`pseudocode` for this datachecks

`source` creators information

`DCtest` *Data checks in YAML format*

Description

Data checks in YAML format

`DC_basisOfRecordBadlyFormed` *Data check basisOfRecordBadlyFormed Check if the specific nature of the data record is within controlled vocabulary*

Description

This data check answers: "Is basis of record within controlled vocabulary??" question.
Data check will pass if **dwc:basisOfRecord was matched to vocabulary** and will fail if **dwc:basisOfRecord could not be unambiguously matched to vocabulary**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `basisOfRecord=HumanObservation,basisOfRecord=LIVING_SPECIMEN`, such data check would return `Passed`.
Example of entries that will fail: `basisOfRecord=Human specimen`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

dwc:basisOfRecord was matched to vocabulary

sampleFailData

dwc:basisOfRecord could not be unambiguously matched to vocabulary

targetDWCField

basisOfRecord

checkCategory

Record_level Terms

References

None

Examples

```
performDC(DC = DC_basisOfRecordBadlyFormed, DATA = bdchecks::dataBats)
```

DC_classUnknown *Data check classUnknown The full scientific name of the class in which the taxon is classified is within given vocabulary*

Description

This data check answers: "Is scientific name of the class known??" question.
Data check will pass if **Taxonomic rank Class was interpreted correctly** and will fail if **Taxonomic rank Class value cannot be interpreted**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `class=Unicorns`, such data check would return `Passed`.
Example of entries that will fail: `class=Birds`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Taxonomic rank Class was interpreted correctly

sampleFailData

Taxonomic rank Class value cannot be interpreted

targetDWCField

class

checkCategory

Taxon

References

None

Examples

```
performDC(DC = DC_classUnknown, DATA = bdchecks::dataBats)
```

DC_coordinatePrecisionMismatch

Data check coordinatePrecisionMismatch Check if decimal values given in decimalLatitude and decimalLongitude does agree witha a decimal representation of the precision of the coordinates (coordinatePrecision)

Description

This data check answers: "Is decimal number in langitude and latitude not smaller than a given decimal precision??" question.
Data check will pass if **The number of decimal places of latitude (dwc:decimalLatitude) and longitude (dwc:decimalLongitude) is in agreement with the supplied coordinate precision (dwc:coordinatePrecision)** and will fail if **The number of decimal places of latitude (dwc:decimalLatitude) or longitude (dwc:decimalLongitude) is not in agreement with the supplied coordinate precision (dwc:coordinatePrecision)**.
Dimension of this data check is **Consistency** and it's flagging type is: **FLAG**
Example of entries that will pass: `coordinatePrecision=1e-5, decimalLatitude=-35.123`, such data check would return `Passed`.
Example of entries that will fail: `coordinatePrecision=0.5, decimalLatitude=-35.123456`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The number of decimal places of latitude (dwc:decimalLatitude) and longitude (dwc:decimalLongitude) is in agreement with the supplied coordinate precision (dwc:coordinatePrecision)

sampleFailData

The number of decimal places of latitude (dwc:decimalLatitude) or longitude (dwc:decimalLongitude) is not in agreement with the supplied coordinate precision (dwc:coordinatePrecision)

targetDWCField

decimalLatitude,decimalLongitude

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_coordinatePrecisionMismatch, DATA = bdchecks::dataBats)
```

`DC_coordinatesZero` *Data check coordinatesZero Check if decimal values given in decimal-Latitude and decimalLongitude are not zero*

Description

This data check answers: "Is decimal number in langitude and latitude not zero??" question.
Data check will pass if **Decimal latitude and longitude are not zero (0) degrees** and will fail if **Decimal latitude and longitude are both zero (0) degrees**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `decimalLatitude=-1`, `decimalLongitude=1`, such data check would return `Passed`.
Example of entries that will fail: `decimalLatitude=0`, `decimalLongitude=0`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Decimal latitude and longitude are not zero (0) degrees

sampleFailData

Decimal latitude and longitude are both zero (0) degrees

targetDWCField

decimalLatitude,decimalLongitude

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_coordinatesZero, DATA = bdchecks::dataBats)
```

`DC_countryMismatch` *Data check countryMismatch Check if given country match given country code.*

Description

This data check answers: "Does country and country code match??" question.
Data check will pass if **Country name (dwc:country) and ISO country code (dwc:countryCode) match** and will fail if **Country name (dwc:country) and ISO country code (dwc:countryCode) do not match**.
Dimension of this data check is **Consistency** and it's flagging type is: **FLAG**
Example of entries that will pass: `country=Australia, countryCode=AU`, such data check would return `Passed`.
Example of entries that will fail: `country=Australia, countryCode=4`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Country name (dwc:country) and ISO country code (dwc:countryCode) match

sampleFailData

Country name (dwc:country) and ISO country code (dwc:countryCode) do not match

targetDWCField

country,countryCode

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_countryMismatch, DATA = bdchecks::dataBats)
```

DC_countryNameUnknown *Data check countryNameUnknown Check if the name of the country or major administrative unit in which the location occurs is within given vocabulary*

Description

This data check answers: "Is country known??" question.
Data check will pass if **Country name (dwc:country) is in vocabulary** and will fail if **Country name (dwc:country) not in vocabulary**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `country=Australia`, such data check would return `Passed`.
Example of entries that will fail: `country=Austend`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Country name (dwc:country) is in vocabulary

sampleFailData

Country name (dwc:country) not in vocabulary

targetDWCField

country

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_countryNameUnknown, DATA = bdchecks::dataBats)
```

DC_dataGeneralised *Data check dataGeneralised Chech if dataGeneralizations is a NULL column (ie, latitude and longitude were not generalized)*

Description

This data check answers: "Is dataGeneralizations NULL??" question.
Data check will pass if **Latitude and longitude values have not been generalized as indicated by dwc:dataGeneralizations is NULL** and will fail if **Latitude and longitude values may have been generalized as indicated by dwc:dataGeneralizations is not NULL**.
Dimension of this data check is **Resolution** and it's flagging type is: **FLAG**
Example of entries that will pass: `dataGeneralizations=`, such data check would return `Passed`.
Example of entries that will fail: `dataGeneralizations=record placed on 0.1 degree grid`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Latitude and longitude values have not been generalized as indicated by dwc:dataGeneralizations is NULL

sampleFailData

Latitude and longitude values may have been generalized as indicated by dwc:dataGeneralizations is not NULL

targetDWCField

dataGeneralizations

checkCategory

Record_level Terms

References

None

Examples

```
performDC(DC = DC_dataGeneralised, DATA = bdchecks::dataBats)
```

DC_dateIdentifiedInFuture *Data check dateIdentifiedInFuture Check if record identification date is not in the future*

Description

This data check answers: "?" question.
Data check will pass if **Date (dwc:dateIdentified) is not in the future** and will fail if **Date (dwc:dateIdentified) is in the future**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `dateIdentified` <= `TODAY`, such data check would return `Passed`.
Example of entries that will fail: `dateIdentified` > `TODAY`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Date (dwc:dateIdentified) is not in the future

sampleFailData

Date (dwc:dateIdentified) is in the future

targetDWCField

dateIdentified

checkCategory

Identification

References

None

Examples

```
performDC(DC = DC_dateIdentifiedInFuture, DATA = bdchecks::dataBats)
```

`DC_dateNull` *Data check dateNull Check if eventDate, year, verbatimEventDate are not NULL*

Description

This data check answers: "Is eventDate, year or verbatimEventDate not NULL??" question.
Data check will pass if **Date information is present** and will fail if **No date information**.
Dimension of this data check is **Completeness** and it's flagging type is: **FLAG**
Example of entries that will pass: `year=2000` + `eventDate=2000.01.01` + `verbatimEventDate=2000.01.01`, such data check would return `Passed`.
Example of entries that will fail: `year=NULL` + `eventDate=NULL` + `verbatimEventDate=NULL`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Date information is present

sampleFailData

No date information

targetDWCField

eventDate,year,verbatimEventDate

checkCategory

Event

References

None

Examples

```
performDC(DC = DC_dateNull, DATA = bdchecks::dataBats)
```

`DC_dayInvalid` *Data check dayInvalid Check if event day is valid (1 <= integer <= 31)*

Description

This data check answers: "?" question.
Data check will pass if **The value given for event day is between 1 and 31** and will fail if **The value given for event day is less than 1 or greater than 31**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `day=1,day=31`, such data check would return `Passed`.
Example of entries that will fail: `day=32,day=0,day=1.1,day=-1`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The value given for event day is between 1 and 31

sampleFailData

The value given for event day is less than 1 or greater than 31

targetDWCField

day

checkCategory

Event

References

None

Examples

```
performDC(DC = DC_dayInvalid, DATA = bdchecks::dataBats)
```

DC_depthOutOfRange *Data check depthOutOfRange Check if depthOutOfRange is not out of range.*

Description

This data check answers: "Is depth within range??" question.
Data check will pass if **Minimum depth is greater than or equal to zero (0) and maximum depth is less than 11,000 meters** and will fail if **Minimum depth is less than zero (0) or maximum depth is greater than 11,000 meters**.
Dimension of this data check is **Likeliness** and it's flagging type is: **FLAG**
Example of entries that will pass: `minimumDepthInMeters=1,maximumDepthInMeters=100`, such data check would return `Passed`.
Example of entries that will fail: `maximumDepthInMeters=19380`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Minimum depth is greater than or equal to zero (0) and maximum depth is less than 11,000 meters

sampleFailData

Minimum depth is less than zero (0) or maximum depth is greater than 11,000 meters

targetDWCField

minimumDepthInMeters,maximumDepthInMeters

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_depthOutOfRange, DATA = bdchecks::dataBats)
```

DC_elevationOutOfRange

Data check elevationOutOfRange Check if elevationOutOfRange is not out of range.

Description

This data check answers: "Is elevation within range??" question.
Data check will pass if **Minimum elevation in meters is greater or equal to zero and maximum elevation in meters is less than 10,000 meters** and will fail if **Minimum elevation in meters is less than zero and/or maximum elevation in meters is greater than 10,000**.
Dimension of this data check is **Likeliness** and it's flagging type is: **FLAG**
Example of entries that will pass: `minimumElevationInMeters=1,maximumElevationInMeters=100`, such data check would return `Passed`.
Example of entries that will fail: `maximumElevationInMeters=19375`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Minimum elevation in meters is greater or equal to zero and maximum elevation in meters is less than 10,000 meters

sampleFailData

Minimum elevation in meters is less than zero and/or maximum elevation in meters is greater than 10,000

targetDWCField

minimumElevationinMeters,maximumElevationInMeters

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_elevationOutOfRange, DATA = bdchecks::dataBats)
```

`DC_eventDateInFuture` *Data check eventDateInFuture Check if event date is not in the future*

Description

This data check answers: "?" question.
Data check will pass if **dwc:eventDate is valid** and will fail if **dwc:eventDate is in the future**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `eventDate=2001-01-01`, such data check would return `Passed`.
Example of entries that will fail: `eventDate=2230-12-31`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

dwc:eventDate is valid

sampleFailData

dwc:eventDate is in the future

targetDWCField

eventDate

checkCategory

Event

References

None

Examples

```
performDC(DC = DC_eventDateInFuture, DATA = bdchecks::dataBats)
```

`DC_identifiedDateImprobable` *Data check identifiedDateImprobable Check if identification date is between Linnaeus and current date*

Description

This data check answers: "?" question.
Data check will pass if **The date of identification (dwc:dateIdentified) is post Linnaeus (1753) to the current date** and will fail if **The date of identification (dwc:dateIdentified) falls prior to Linnaeus (1753) or after the current date.**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `dateidentified=2000-01-01`, such data check would return `Passed`.
Example of entries that will fail: `dateidentified=1573-02-14`,`dateidentified=1000-01-01`,`dateidentified=3000-0` such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The date of identification (dwc:dateIdentified) is post Linnaeus (1753) to the current date

sampleFailData

The date of identification (dwc:dateIdentified) falls prior to Linnaeus (1753) or after the current date.

targetDWCField

dateIdentified

checkCategory

Identification

References

None

Examples

```
performDC(DC = DC_identifiedDateImprobable, DATA = bdchecks::dataBats)
```

`DC_individualcountInvalid` *Data check individualcountInvalid Check if the number of individuals represented present at the time of the occurrence is and intiger*

Description

This data check answers: "?" question.
Data check will pass if **The count of individuals is a valid integer** and will fail if **The count of individuals is not an integer and therefore invalid**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `individualCount=1`, such data check would return `Passed`.
Example of entries that will fail: `individualCount=0.3`,`individualCount=-1`,`individualCount=0`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The count of individuals is a valid integer

sampleFailData

The count of individuals is not an integer and therefore invalid

targetDWCField

individualCount

checkCategory

Occurrence

References

None

Examples

```
performDC(DC = DC_individualcountInvalid, DATA = bdchecks::dataBats)
```

`DC_modifiedInFuture` *Data check modifiedInFuture Check if date on which the resource was changed is not in the future*

Description

This data check answers: "?" question.
Data check will pass if **dcterms:modified date is valid** and will fail if **dcterms:modified is in the future**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `dcterms:modified=2000-01-01`, such data check would return `Passed`.
Example of entries that will fail: `dcterms:modified=2230-12-31`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

dcterms:modified date is valid

sampleFailData

dcterms:modified is in the future

targetDWCField

modified

checkCategory

Record_level Terms

References

None

Examples

```
performDC(DC = DC_modifiedInFuture, DATA = bdchecks::dataBats)
```

DC_monthInvalid *Data check monthInvalid Check if event month is valid (1 <= integer <= 12)*

Description

This data check answers: "?" question.
Data check will pass if **The event month is between 1 and 12** and will fail if **The event month is less than 1 or is greater than 12**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `month=1`, such data check would return `Passed`.
Example of entries that will fail: `month=14,month=0`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The event month is between 1 and 12

sampleFailData

The event month is less than 1 or is greater than 12

targetDWCField

month

checkCategory

Event

References

None

Examples

```
performDC(DC = DC_monthInvalid, DATA = bdchecks::dataBats)
```

DC_namePublishedYearInFuture

Data check namePublishedYearInFuture Check if year in which scientific name was published is not in the future

Description

This data check answers: "?" question.
Data check will pass if **dwc:namePublishedInYear is valid** and will fail if **dwc:namePublishedInYear is in the future**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `namePublishedInYear=2000`, such data check would return `Passed`.
Example of entries that will fail: `namePublishedInYear=2230`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

dwc:namePublishedInYear is valid

sampleFailData

dwc:namePublishedInYear is in the future

targetDWCField

namePublishedInYear

checkCategory

Taxon

References

None

Examples

```
performDC(DC = DC_namePublishedYearInFuture, DATA = bdchecks::dataBats)
```

`DC_occurrenceIdNotGuid` *Data check occurrenceIdNotGuid Check if an identifier for the occurrence is a globally uniquely identifier (currently we use regex pattern solution is implemented)*

Description

This data check answers: "?" question.
Data check will pass if **occurrenceID is a globally uninique identifier (GUID)** and will fail if **occurrenceID is an integer, assuring that it is not a globally uninique identifier (GUID)**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `3cfe9ab4-79f8-4afd-8da5-723183ef16a3`, such data check would return `Passed`.
Example of entries that will fail: `occurrenceID=42`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

occurrenceID is a globally uninique identifier (GUID)

sampleFailData

occurrenceID is an integer, assuring that it is not a globally uninique identifier (GUID)

targetDWCField

occurrenceID

checkCategory

Occurrence

References

None

Examples

```
performDC(DC = DC_occurrenceIdNotGuid, DATA = bdchecks::dataBats)
```

DC_precisionRangeMismatch

Data check precisionRangeMismatch Check if precision range mismatch is between 0 and 1

Description

This data check answers: "?" question.
Data check will pass if **The coordinate precision (dwc:coordinatePrecision) is between zero (minimum) and one (maximum)** and will fail if **The coordinate precision (dwc:coordinatePrecision), as a decimal representation, is outside the range of zero (minimum) and one (maximum)**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `coordinatePrecision=0`,`coordinatePrecision=0.5`,`coordinatePrecision=1`, such data check would return `Passed`.
Example of entries that will fail: `coordinatePrecision=3`,`coordinatePrecision=-1`, such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The coordinate precision (dwc:coordinatePrecision) is between zero (minimum) and one (maximum)

sampleFailData

The coordinate precision (dwc:coordinatePrecision), as a decimal representation, is outside the range of zero (minimum) and one (maximum)

targetDWCField

coordinatePrecision

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_precisionRangeMismatch, DATA = bdchecks::dataBats)
```

DC_uncertaintyRangeMismatch

Data check uncertaintyRangeMismatch Check if geopoint uncertainty range mismatch is a positive integer (meters).

Description

This data check answers: "?" question.
Data check will pass if **Geopoint uncertainty (dwc:coordinateUncertaintyInMeters) is a whole number and greater than zero (meters)** and will fail if **Geopoint uncertainty (dwc:coordinateUncertaintyInMeters) should be a whole number and greater than zero (meters)**.
Dimension of this data check is **Conformance** and it's flagging type is: **FLAG**
Example of entries that will pass: `coordinateUncertaintyInMeters=1`, such data check would return `Passed`.
Example of entries that will fail: `coordinateUncertaintyInMeters=0.002,coordinateUncertaintyInMeters=-1,coor` such data check would return `Failed`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

Geopoint uncertainty (dwc:coordinateUncertaintyInMeters) is a whole number and greater than zero (meters)

sampleFailData

Geopoint uncertainty (dwc:coordinateUncertaintyInMeters) should be a whole number and greater than zero (meters)

targetDWCField

coordinateUncertaintyInMeters

checkCategory

Location

References

None

Examples

```
performDC(DC = DC_uncertaintyRangeMismatch, DATA = bdchecks::dataBats)
```

`DC_yearMissing` *Data check yearMissing Check if year information is not missing*

Description

This data check answers: "Is year information present??" question.
Data check will pass if **The value for dwc:year is valid** and will fail if **The value for dwc:year is NULL**.
Dimension of this data check is **Completeness** and it's flagging type is: **FLAG**
Example of entries that will pass: `year=2000`, such data check would return `Passed`.
Example of entries that will fail: `year=`, such data check would return `Faield`.

Format

An object of class `"dataCheck"`, see `dataCheck` for details.

samplePassData

The value for dwc:year is valid

sampleFailData

The value for dwc:year is NULL

targetDWCField

year

checkCategory

Event

References

None

Examples

```
performDC(DC = DC_yearMissing, DATA = bdchecks::dataBats)
```

exportDataCheck *Export Data Checks*

Description

Method that exports data checks

Usage

```
exportDataCheck(object)

## S4 method for signature 'dataCheckFlag'
exportDataCheck(object)
```

Arguments

object	A result of data checks (data check flag class)

Value

A data.frame that contains original users data.frame modified according to data checks

exportDC *Export data checks as R objects*

Description

'exportDC()' is a function for exporting data checks from YAML file to rda and/or roxygen2 file.

Usage

```
exportDC(pathYAML = "./inst/extdata/dataChecks.yaml", exportRDA = TRUE,
  exportROX = TRUE, pathRDA = "./data/", pathROX = "./R/",
  idRDA = "DC_", idROX = "DC_")
```

Arguments

pathYAML	Path to a YAML file.
exportRDA	Should function export data check to a .rda file.
exportROX	Should function create a documentation in .R for roxygen2.
pathRDA	Path for .rda file.
pathROX	Path for .R (roxygen2) file.
idRDA	ID in the beginning of file for .rda file
idROX	ID in the beginning of file for .R file (helps to differentiate other functions from data checks).

filterDataCheck *Filter Data Checks*

Description

'filterDataCheck()' is a function that filters data check result according to filtering vector.

Usage

```
filterDataCheck(DCresult, DCfilts)
```

Arguments

DCresult	Object of a dataCheckFlag generated with 'perforDataCheck()'
DCfilts	A list containing filtering targets and status generated with 'generateDCfilts()'

Value

A data.frame that is filtered according to given vector

generateDCfilts *Filter Data Checks*

Description

'generateDCfilts()' is a function that generates vector for filtering data checks result table according to 'DT::datatable()' 'selectedCells' object.

Usage

```
generateDCfilts(DCresultSummary, selectedCells, filters = c("P", "F",
  "M"))
```

Arguments

DCresultSummary	Summary table for a dataCheckFlag class (must be filterable in 'DT')
selectedCells	Cells selected in 'DT::datatable'
filters	Vector that contains names for passed, failed and missing data checks

Value

A list that contains name of a data checks, it's target and filtering status

generateRoxygenComment *Generate roxygen2 documentation from data check object*

Description

'generateRoxygenComment()' is a function for generating roxygen2 comments for a given data check. It's not super flexible as it just inserts metadata into a hard coded skeleton.

Usage

```
generateRoxygenComment(DC)
```

Arguments

DC	Data check to generate documentation for.

Value

Data check description in a roxygen2 comment style

getDC *Load Data Checks*

Description

'getDC()' is a function for importing data checks from a YAML file as a dataCheck object.

Usage

```
getDC(pathYAML = "../inst/extdata/dataChecks.yaml")
```

Arguments

pathYAML	Path to a YAML file.

Value

A list of data checks

performDataCheck *Perform Data Checks*

Description

'performDataCheck()' is a function for performing all available data checks on a give data set

Usage

```
performDataCheck(data = NULL, DCadd = NULL, DConly = NULL,
  verbose = TRUE, DCstand = ls(pos = ("package:bdchecks"), pattern =
  "^DC_"))
```

Arguments

data	Data set to perform data checks
DCadd	Character vector of names for additional data checks to be performed
DConly	Character vector of names for data checks that should be performed (ie perform only these data checks)
verbose	Message which data check is being performed
DCstand	Character vector of standardize data checks

Value

Object of a dataCheckFlag class (combined result for all performed data checks)

Examples

```
performDataCheck(dataBats)
```

performDC *Perform Data Checks*

Description

Method that performs data check on a given dataset

Usage

```
performDC(DC, DATA)

## S4 method for signature 'dataCheck'
performDC(DC, DATA)
```

Arguments

DC	A data check as a dataCheck class object
DATA	a data frame to perform data check on

Value

A vector of logical values that determine if data check was passed on specific entry in a given DATA object

---

runbdchecks *Launch bdchecks Shiny Application*

Description

'bdchecks' is a function that starts biodiversity check 'shiny' app.

Usage

```
runbdchecks()
```

Value

'shiny::runApp()' result within browser.

---

summary_DC *Summarise Data Checks*

Description

'summary_DC()' is a function that calculated statistics for how many data checks passed. It's main input is an object of a dataCheckFlag class and output is a summary table.

Usage

```
summary_DC(inputFlag, fancy = TRUE, filteringDT = FALSE)
```

Arguments

inputFlag	Object of a dataCheckFlag class
fancy	Should ouput be returned in a rst format
filteringDT	Should output be returned as a summary table that could be parsed with a 'DT' pacakge

Value

A data.frame or rst table with summary statistics

Examples

```
result <- performDataCheck(dataBats)
# Fancy summary table (for usage in reports)
summary_DC(result)
# object of class used for data filtering data.frame
summary_DC(result, fancy = FALSE, filteringDT = TRUE)
```

Index