

# Package ‘rangeModelMetadata’

May 9, 2026

**Type** Package

**Title** Provides Templates for Metadata Files Associated with Species  
Range Models

**Version** 0.1.5

**Maintainer** Cory Merow <cory.merow@gmail.com>

**Description** Range Modeling Metadata Standards (RMMS) address three challenges: they (i) are designed for convenience to encourage use, (ii) accommodate a wide variety of applications, and (iii) are extensible to allow the community of range modelers to steer it as needed. RMMS are based on a data dictionary that specifies a hierarchical structure to catalog different aspects of the range modeling process. The dictionary balances a constrained, minimalist vocabulary to improve standardization with flexibility for users to provide their own values. Merow et al. (2019) <[DOI:10.1111/geb.12993](https://doi.org/10.1111/geb.12993)> describe the standards in more detail. Note that users who prefer to use the R package 'ecospat' can obtain it from <<https://github.com/ecospat/ecospat>>.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 3.5.0)

**Suggests** dismo, googlesheets4, knitr, rmarkdown, rgbif, dplyr, sf

**Imports** shiny, spocc, utils, BIEN, terra

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Cory Merow [aut, cre],  
Brian Maitner [aut],  
Hannah Owens [aut],  
Jamie Kass [aut],  
Brian Enquist [aut],  
Rob Guralnik [aut],  
Damaris Zurrell [aut],  
Christian Koenig [aut]

**Repository** CRAN

**Date/Publication** 2023-10-16 22:50:02 UTC

## Contents

cleanForCSV . . . . .	2
csvToRMM . . . . .	3
rmmAutofillBIEN . . . . .	4
rmmAutofillEnvironment . . . . .	5
rmmAutofillPackageCitation . . . . .	6
rmmAutofillSpoc . . . . .	7
rmmCheckEmpty . . . . .	8
rmmCheckFinalize . . . . .	9
rmmCheckMissingNames . . . . .	10
rmmCheckName . . . . .	11
rmmCheckShiny . . . . .	12
rmmCheckValue . . . . .	13
rmmCleanNULLs . . . . .	14
rmmDataDictionary . . . . .	15
rmmFamilies . . . . .	16
rmmSuggest . . . . .	16
rmmTemplate . . . . .	17
rmmToCSV . . . . .	18
<b>Index</b>	<b>19</b>

---

cleanForCSV

*Helper function for non-string metadata in rmmToCSV*

---

### Description

Cleans up metadata instances that get messy if one tries to write them directly to csv tables (i.e. extent objects, bibtex objects.)

### Usage

```
cleanForCSV(x = NULL)
```

### Arguments

x                    An rmm entry that returned to the rmmToCSV function.

### Details

This is a utility function for use by rmmToCSV.

### Value

Reformatted element for use in rmmToCSV function.

**Author(s)**

Hannah Owens <hannah.owens@gmail.com>, Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>

**See Also**

Other csvConversion: [csvToRMM\(\)](#), [rmmToCSV\(\)](#)

---

csvToRMM

*Create rangeModelMetaData ('rmm') object from a .csv File*

---

**Description**

Takes user-input .csv file and converts it to a rangeModelMetaData ('rmm') object.

**Usage**

```
csvToRMM(csv, family = NULL)
```

**Arguments**

csv	A character file path to the csv file.
family	character string; specifies an application profile (use case) by specifying the families of entities that should be included. Specifying NULL includes all entities. Use <a href="#">rmmFamilies()</a> to see supported values.

**Details**

See Examples.

**Value**

An rmm object that was read from the supplied .csv text file.

**Author(s)**

Hannah Owens <hannah.owens@gmail.com>

**See Also**

Other csvConversion: [cleanForCSV\(\)](#), [rmmToCSV\(\)](#)

**Examples**

```
csv <- "somePathOnYourMachine/rmm_example.csv";  
## Not run: temp <- csvToRMM(csv);
```

---

rmmAutofillBIEN      *Add occurrence metadata from a BIEN query to an rmm object*

---

### Description

This function populates occurrence field in an rmm object with output from a BIEN\_occurrence\_... query

### Usage

```
rmmAutofillBIEN(rmm, occurrences)
```

### Arguments

rmm	an rmm list
occurrences	an occurrence data.frame obtained from a BIEN occurrence query

### Details

See Examples.

### Value

a range model metadata list

### Author(s)

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

### See Also

[BIEN\\_occurrence\\_species](#)

Other autofill: [rmmAutofillEnvironment\(\)](#), [rmmAutofillPackageCitation\(\)](#), [rmmAutofillspocc\(\)](#)

### Examples

```
## Not run:  
rmm <- rmmTemplate()  
xs <- BIEN::BIEN_occurrence_species(species="Xanthium strumarium")  
rmmAutofillBIEN(rmm = rmm, occurrences = xs)  
  
## End(Not run)
```

---

`rmmAutofillEnvironment`*Add relevant environmental data information to an rmm object*

---

**Description**

This can be used with environmental layers used for fitting or transferring

**Usage**

```
rmmAutofillEnvironment(rmm, env, transfer)
```

**Arguments**

<code>rmm</code>	an rmm list
<code>env</code>	a SPatRaster object
<code>transfer</code>	0 if not transfer, 1:n for n environments that you're transferring to

**Details**

See Examples.

**Value**

a range model metadata list

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other autofill: [rmmAutofillBIEN\(\)](#), [rmmAutofillPackageCitation\(\)](#), [rmmAutofillspocc\(\)](#)

**Examples**

```
## Not run:
rmm <- rmmTemplate()
rasterFiles <- list.files(path=paste(system.file(package='dismo'), '/ex', sep=''),
                          pattern='grd', full.names=TRUE)
#make a stack of the rasters
env <- terra::rast(rasterFiles)
# for fitting environment
rmm <- rmmAutofillEnvironment(rmm,env,transfer=0)
# for the first environment that you're transferring to
rmm <- rmmAutofillEnvironment(rmm,env,transfer=1)
# for the second environment that you're transferring to, etc.
rmm <- rmmAutofillEnvironment(rmm,env,transfer=2)
```

```
## End(Not run)
```

---

```
rmmAutofillPackageCitation
```

```
Add all package citations to an rmm object
```

---

## Description

Using bibtex citations

## Usage

```
rmmAutofillPackageCitation(rmm, packages)
```

## Arguments

rmm	an rmm list
packages	a vector of quoted package names

## Details

See Examples.

## Value

a range model metadata list

## Author(s)

Brian Maitner <bmaitner@gmail.com>, Cory Merow <cory.merow@gmail.com>

## See Also

Other autofill: [rmmAutofillBIEN\(\)](#), [rmmAutofillEnvironment\(\)](#), [rmmAutofillspocc\(\)](#)

## Examples

```
rmm <- rmmTemplate()  
rmm <- rmmAutofillPackageCitation(rmm,c('terra', 'sf'))
```

---

rmmAutofillspocc	<i>Add occurrence metadata from a spocc query to an rmm object</i>
------------------	--

---

## Description

This function populates occurrence field in an rmm object with output from a spocc query

## Usage

```
rmmAutofillspocc(rmm, occ)
```

## Arguments

rmm	an rmm list
occ	Output from <a href="#">occ</a>

## Details

See Examples.

## Value

a range model metadata list

## Author(s)

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

## See Also

[occ](#)

Other autofill: [rmmAutofillBIEN\(\)](#), [rmmAutofillEnvironment\(\)](#), [rmmAutofillPackageCitation\(\)](#)

## Examples

```
## Not run:  
rmm <- rmmTemplate()  
xs <- spocc::occ("Xanthium strumarium")  
rmmAutofillspocc(rmm = rmm, occ = xs)  
  
## End(Not run)
```

---

`rmmCheckEmpty`*Check an rmm object for empty fields*

---

## Description

Identify empty fields in an rmm object and classify these into obligate and optional fields.

## Usage

```
rmmCheckEmpty(rmm, family = c("base"))
```

## Arguments

<code>rmm</code>	a range model metadata list
<code>family</code>	an rmm family, "base" by default

## Details

See Examples.

## Value

A dataframe containing empty fields labeled as obligate, optional, or suggested.

## Author(s)

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

## See Also

Other check: [rmmCheckFinalize\(\)](#), [rmmCheckMissingNames\(\)](#), [rmmCheckName\(\)](#), [rmmCheckValue\(\)](#), [rmmCleanNULLs\(\)](#)

## Examples

```
#First, make an empty rmm object:
rmm <- rmmTemplate()
#Next, we check for empty fields:
empties1 <- rmmCheckEmpty(rmm = rmm)
#If looks like there are quite a few empty obligate fields. Let's populate a few:
rmm$data$occurrence$taxon <- "Acer rubrum"
rmm$data$environment$variableNames <- "Bio1"
#Now, if we run rmmCheckEmpty again, we see there are 2 fewer empty, obligate fields
empties2 <- rmmCheckEmpty(rmm = rmm)
```

---

rmmCheckFinalize	<i>Run a final check of an rmm object</i>
------------------	---

---

**Description**

Check an rmm object for non-standard and missing values and fields

**Usage**

```
rmmCheckFinalize(rmm, family = c("base"))
```

**Arguments**

rmm	a range model metadata list
family	The rmm family to check the rmm against

**Details**

See Examples.

**Value**

Prints feedback to point out possible errors.

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other check: [rmmCheckEmpty\(\)](#), [rmmCheckMissingNames\(\)](#), [rmmCheckName\(\)](#), [rmmCheckValue\(\)](#), [rmmCleanNULLs\(\)](#)

**Examples**

```
rmm <- rmmTemplate() # Make an empty template
rmmCheckFinalize(rmm)
```

rmmCheckMissingNames *Check for missing fields*

---

**Description**

Identify obligate fields that are missing

**Usage**

```
rmmCheckMissingNames(rmm, family = c("base"))
```

**Arguments**

rmm	a range model metadata list
family	The rmm family to check the rmm against

**Details**

See Examples.

**Value**

A vector of names that are missing from the rmm object.

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other check: [rmmCheckEmpty\(\)](#), [rmmCheckFinalize\(\)](#), [rmmCheckName\(\)](#), [rmmCheckValue\(\)](#), [rmmCleanNULLs\(\)](#)

**Examples**

```
rmm <- rmmTemplate() # Make an empty template
```

---

rmmCheckName	<i>Check field names of a range model metadata list against conventions</i>
--------------	---

---

**Description**

Identify nonstandard fields

**Usage**

```
rmmCheckName(  
  rmm,  
  cutoff_distance = 3,  
  returnData = F,  
  interactiveCorrections = FALSE  
)
```

**Arguments**

rmm	a range model metadata list
cutoff_distance	number of allowed different characters to match standardized names
returnData	logical. If FALSE, the function will return the (possibly) corrected rmm object. If TRUE, the function will return a data.frame containing information on incorrect names.
interactiveCorrections	logical. If TRUE, the user will be prompted to indicate whether the proposed correction should be accepted, thereby modifying the 'rmm' object. If FALSE, suggestions will just be printed to the screen and users can edit them manually.

**Details**

See Examples.

**Value**

Either an rmm list object ('returnData=FALSE') or a data.frame containing information on possible name errors ('returnData=TRUE').

**Note**

Names returned by this check may be either incorrectly named or correctly named but missing from the data dictionary.

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other check: [rmmCheckEmpty\(\)](#), [rmmCheckFinalize\(\)](#), [rmmCheckMissingNames\(\)](#), [rmmCheckValue\(\)](#), [rmmCleanNULLs\(\)](#)

**Examples**

```
rmm <- rmmTemplate() # Make an empty template
rmm$dataPrep$biological$taxonomicHarmonization$taxonomy_source <- "The Plant List"
# Add a new, non-standard field
rmm.1 <- rmmCheckName(rmm)
# Checking the names should identify the new, non-standard field we've added ("taxonomy_source")
```

---

rmmCheckShiny

*RangeModelMetadata Check in Shiny*

---

**Description**

Run shiny app to visualize rmm check functions

**Usage**

```
rmmCheckShiny()
```

**Details**

See Examples.

**Value**

None

**Note**

This function launches a shiny app in the default web browser

**Author(s)**

Jamie Kass <jamie.m.kass@gmail.com>

**Examples**

```
## Not run:
rmm1 <- rmmTemplate()
rmm1 <- rmmAutofillPackageCitation(rmm1,c('raster','sp'))
rasterFiles <- list.files(path=paste(system.file(package='dismo'), '/ex', sep=''),
                          pattern='grd', full.names=TRUE)

make a stack of the rasters
env <- terra::rast(rasterFiles)
# for fitting environment
rmm1 <- rmmAutofillEnvironment(rmm1,env,transfer=0)
# for transfer environment 1 (assuming different than for fitting)
rmm1 <- rmmAutofillEnvironment(rmm1,env,transfer=1)
# for transfer environment 2 (assuming different than 1)
rmm1 <- rmmAutofillEnvironment(rmm1,env,transfer=2)

## End(Not run)
## Not run: rmmCheckShiny(rmm1)
```

---

rmmCheckValue	<i>Check values of a range model metadata list against commonly used values</i>
---------------	---

---

**Description**

Identify nonstandard values

**Usage**

```
rmmCheckValue(rmm, cutoff_distance = 3, returnData = F)
```

**Arguments**

rmm	a range model metadata list
cutoff_distance	The maximum allowable similarity (Levenshtein (edit) distance) for use in fuzzy matching.
returnData	Should a dataframe containing information on matched and unmatched values be returned? Default is FALSE

**Details**

See Examples.

**Value**

Text describing identical, similar and non-matched values for rmm entities with suggested values. If `returnData = T`, a dataframe is returned containing 5 columns: `field` (the rmm entity), `exact_match` (values that appear correct), `partial_match` (values that are partial\_match to common values), `not_matched` (values that are dissimilar from accepted values), `partial_match_suggestions` (suggested values for partial\_match values).

**Note**

Names returned by this check may be either incorrectly named or correctly named but missing from the data dictionary.

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other check: [rmmCheckEmpty\(\)](#), [rmmCheckFinalize\(\)](#), [rmmCheckMissingNames\(\)](#), [rmmCheckName\(\)](#), [rmmCleanNULLs\(\)](#)

**Examples**

```
rmm <- rmmTemplate() #First, we create an empty rmm template
rmm$data$environment$variableNames <- c("bio1", "bio 2", "bio3", "cromulent")
#We add 3 of the bioclim layers, including a spelling error (an extra space) in bio2,
# and a word that is clearly not a climate layer, 'cromulent'.
rmmCheckValue(rmm = rmm)
#Now, when we check the values, we see that bio1 and bio2 are reported as exact matches,
#while 'bio 2' is flagged as a partial match with a suggested value of 'bio2',
# and cromulent is flagged as not matched at all.
#If we'd like to return a dataframe containing this information in a perhaps more useful format:
rmmCheckValue_output <- rmmCheckValue(rmm = rmm, returnData = TRUE)
```

---

rmmCleanNULLs

*Remove NULL entries range model metadata list*

---

**Description**

Check if fields are NULL in a range model metadata list and toss

**Usage**

```
rmmCleanNULLs(rmm)
```

**Arguments**

rmm                    a range model metadata list

**Details**

See Examples.

**Value**

printout to the console

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

**See Also**

Other check: [rmmCheckEmpty\(\)](#), [rmmCheckFinalize\(\)](#), [rmmCheckMissingNames\(\)](#), [rmmCheckName\(\)](#), [rmmCheckValue\(\)](#)

**Examples**

```
# see vignette('rmm_vignette')
```

---

rmmDataDictionary      *Open range model metadata dictionary.*

---

**Description**

For viewing only

**Usage**

```
rmmDataDictionary(excel = FALSE)
```

**Arguments**

excel                    logical; open in excel?

**Value**

If 'excel==FALSE', returns a data.frame, if 'excel==TRUE' it returns nothing but attempts to open the metadata dictionary in excel.

**Examples**

```
dd=rmmDataDictionary()
```

---

rmmFamilies	<i>Print supported family names for rmm objects</i>
-------------	---

---

**Description**

Used to see options to for specifying an rmm object template

**Usage**

```
rmmFamilies()
```

**Value**

a vector of characters, indicating which RMMS families are supported

**Examples**

```
rmmFamilies()
```

---

rmmSuggest	<i>Suggest inputs for a range model metadata list</i>
------------	---

---

**Description**

Supply fields to receive suggested inputs

**Usage**

```
rmmSuggest(charString, fullFieldDepth = FALSE)
```

**Arguments**

`charString` string referencing fields of the form 'field1\$field2' or 'field1\$field2\$field3', etc.

`fullFieldDepth` print all fields below the current field depth

```
rmm1 <- rmmTemplate() rmmSuggest('dataPrep',fullFieldDepth=FALSE) rmm-
Suggest('dataPrep',fullFieldDepth=TRUE) rmmSuggest('dataPrep$errors$duplicateRemoval')
rmmSuggest('dataPrep$errors$duplicateRemoval$rule') rmmSuggest('model') rmm-
Suggest('modelFit$algorithmSettings$') rmmSuggest('modelFit$algorithmSettings$maxent$')
rmmSuggest('$modelFit$algorithmSettings$maxent$featureSet')
```

**Details**

See Examples.

**Value**

list of suggestions

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>,

---

rmmTemplate

*Range modeling metadata*

---

**Description**

Make an empty metadata list

**Usage**

```
rmmTemplate(family = NULL)
```

**Arguments**

family            character string; specifies an application profile (use case) by specifying the families of entities that should be included. Specifying NULL includes all entities. Use 'rmmFamilies' to see supported values.

**Details**

See Examples.

**Value**

a range model metadata list

**Author(s)**

Cory Merow <cory.merow@gmail.com>, Brian Maitner <bmaitner@gmail.com>

**Examples**

```
rmm1 <- rmmTemplate()  
rmm2 <- rmmTemplate(family=c('base'))  
str(rmm2)
```

rmmToCSV

*Create .csv File From rangeModelMetaData Object***Description**

Takes user-input rangeModelMetaData object and from it generates a .csv file that can be used to document range model metadata for a variety of applications.

**Usage**

```
rmmToCSV(x = rmmTemplate(family = NULL), filename = NULL)
```

**Arguments**

`x` An object of class `rmm` that the user wishes transposed into a .csv file.  
`filename` The name of the transcription .csv file.

**Details**

See Examples.

**Value**

An data frame containing all the information from an `rmm` object.

**Author(s)**

Hannah Owens <hannah.owens@gmail.com>, Cory Merow <cory.merow@gmail.com>

**See Also**

Other csvConversion: [cleanForCSV\(\)](#), [csvToRMM\(\)](#)

**Examples**

```
rmm <- rmmTemplate()
rasterFiles <- list.files(path=paste(system.file(package='dismo'), '/ex', sep=''),
                        pattern='grd', full.names=TRUE)
#make a stack of the rasters
env <- terra::rast(rasterFiles)
# for fitting environment
rmm <- rmmAutofillEnvironment(rmm,env,transfer=0)
# for the first environment that you're transferring to
rmm <- rmmAutofillEnvironment(rmm,env,transfer=1)
# for the second environment that you're transferring to, etc.
rmm <- rmmAutofillEnvironment(rmm,env,transfer=2)
## Not run:
tmp <- rmmToCSV(rmm, file='somePathOnYourMachine/rmm_example.csv')

## End(Not run)
```

# Index

## \* **autofill**

rmmAutofillBIEN, [4](#)  
rmmAutofillEnvironment, [5](#)  
rmmAutofillPackageCitation, [6](#)  
rmmAutofillspocc, [7](#)

## \* **check**

rmmCheckEmpty, [8](#)  
rmmCheckFinalize, [9](#)  
rmmCheckMissingNames, [10](#)  
rmmCheckName, [11](#)  
rmmCheckValue, [13](#)  
rmmCleanNULLs, [14](#)

## \* **csvConversion**

cleanForCSV, [2](#)  
csvToRMM, [3](#)  
rmmToCSV, [18](#)

BIEN\_occurrence\_species, [4](#)

cleanForCSV, [2](#), [3](#), [18](#)

csvToRMM, [3](#), [3](#), [18](#)

occ, [7](#)

rmmAutofillBIEN, [4](#), [5–7](#)  
rmmAutofillEnvironment, [4](#), [5](#), [6](#), [7](#)  
rmmAutofillPackageCitation, [4](#), [5](#), [6](#), [7](#)  
rmmAutofillspocc, [4–6](#), [7](#)  
rmmCheckEmpty, [8](#), [9](#), [10](#), [12](#), [14](#), [15](#)  
rmmCheckFinalize, [8](#), [9](#), [10](#), [12](#), [14](#), [15](#)  
rmmCheckMissingNames, [8](#), [9](#), [10](#), [12](#), [14](#), [15](#)  
rmmCheckName, [8–10](#), [11](#), [14](#), [15](#)  
rmmCheckShiny, [12](#)  
rmmCheckValue, [8–10](#), [12](#), [13](#), [15](#)  
rmmCleanNULLs, [8–10](#), [12](#), [14](#), [14](#)  
rmmDataDictionary, [15](#)  
rmmFamilies, [16](#)  
rmmSuggest, [16](#)  
rmmTemplate, [17](#)  
rmmToCSV, [3](#), [18](#)