

# Package ‘IDEATools’

July 21, 2025

**Title** Individual and Group Farm Sustainability Assessments using the IDEA4 Method

**Version** 3.5.2

**Description** Collection of tools to automate the processing of data collected through the IDEA4 method (see Zahm et al. (2018) <[doi:10.1051/cagri/2019004](https://doi.org/10.1051/cagri/2019004)> ). Starting from the original data collecting files this package provides functions to compute IDEA indicators, draw modern and aesthetic plots, and produce a wide range of reporting materials.

**Encoding** UTF-8

**URL** <https://davidcarayon.github.io/IDEATools/index.html>,  
<https://github.com/davidcarayon/IDEATools>

**BugReports** <https://github.com/davidcarayon/IDEATools/issues>

**RoxygenNote** 7.2.3

**Imports** data.table, jsonlite, readxl, tibble, stringi, ggimage, pdfutils, ggpubr, ggplot2, rlang, ggtext, rmarkdown, shiny, openxlsx

**License** GPL-3

**Depends** R (>= 4.1.0)

**VignetteBuilder** knitr

**Suggests** covr, knitr

**NeedsCompilation** no

**Author** David Carayon [aut, cre, cph] (ORCID:  
<<https://orcid.org/0000-0003-0217-2175>>)

**Maintainer** David Carayon <david.carayon@inrae.fr>

**Repository** CRAN

**Date/Publication** 2023-12-15 10:20:02 UTC

## Contents

compute_idea . . . . .	2
diag_idea . . . . .	3
jsonify2 . . . . .	5
old_idea . . . . .	5
plot_idea . . . . .	6
read_idea . . . . .	8
runGUI . . . . .	9
show_decision_rules . . . . .	9
show_tree_structure . . . . .	10
theme_idea . . . . .	10
write_idea . . . . .	11
<b>Index</b>	<b>13</b>

---

compute_idea	<i>Compute IDEA4 indicators and aggregation metrics</i>
--------------	---

---

### Description

Aggregates items from read\_idea() to produce IDEA4 indicators, components, dimensions and properties.

### Usage

```
compute_idea(data)
```

### Arguments

data            an object of class IDEA\_items produced with read\_idea()

### Details

This function is designed to compute IDEA scores for the dimensions and properties approaches. A copy of the decision rules used for the properties approach can be locally exported as an excel file with :

```
IDEATools::show_decision_rules()
```

Further information about decision rules can be found in this vignette :

```
vignette("decision_rules", package = "IDEATools")
```

**Value**

An object of class "IDEA\_data" with three attributes :

**metadata** a named list containing the 17 metadata entries about the farm

**dataset** a tibble containing the score computed for the 53 indicators, 13 components and 3 dimensions

**nodes** a list of tibbles, one per property plus a global one, which all describe the qualitative evaluation obtained for each leaf/node and for the final property.

**Examples**

```
library(IDEATools)
path <- system.file("example_data/idea_example_1.json", package = "IDEATools")
my_data <- read_idea(path)
computed_data <- compute_idea(my_data)
```

---

diag_idea	<i>Run a complete IDEA4 diagnosis</i>
-----------	---------------------------------------

---

**Description**

This is a wrapper function designed to compute a complete IDEA4 diagnosis with a single function. According to the user's input, functions from IDEATools will be sequentially called to produce the desired output.

**Usage**

```
diag_idea(
  input,
  output_directory,
  type = "single",
  export_type = c("report", "local"),
  plot_choices = c("dimensions", "trees", "radars"),
  report_format = "pdf",
  prefix = NULL,
  dpi = 320,
  append = FALSE,
  quiet = FALSE
)
```

**Arguments**

**input** a character vector with path names to a single file, multiple files or even a directory with IDEA data. File extensions can either be json or xls(x)

**output\_directory** the output directory for the rendered reports and/or plots. Defaults to "IDEATools\_output"

type	the type of analysis to perform. Can be "single" for single farm-related results, "group" for group-related results, "group_reference" for anonymous group-related results, or a combination of the latter, provided that the number of farms is at least 3. Note that "group_reference" is a special option designed to work with an online platform using this package (WEBIDEA)
export_type	the type of output to produce. Can be either "report" to produce compiled reports and/or "local" to write raw plots. If NULL, the algorithm will not produce any plots on machine and will return a list with the IDEA results.
plot_choices	the type of plots to be produced. Can be either "dimensions", "trees" or "radars" or a combination of these 3. Ignored if the export type is "report".
report_format	a string indicating the output format if type = "report". Can be a single format (e.g "pdf") or multiple formats (e.g. c("pdf", "xlsx")). Possible formats are "pdf", "docx", "pptx" and "xlsx"
prefix	a prefix which will be added to output files names. Typically, the name of the farm. Ignored if length(input) > 1 or in the case of a group analysis : the metadata\$MTD_01 field will be used to identify each farm.
dpi	ggplot output resolution.
append	In the case of a single excel report and if the input is an xlsx file, should the results be appended to the original file ?
quiet	A command to remove console printing.

## Details

This function is designed to provide the user a single function to use for a full IDEA4 diagnosis.

If the input is a single file, then a simple "read\_idea() |> compute\_idea() |> plot\_idea() |> write\_idea()" or "old\_idea() |> plot\_idea() |> write\_idea()" pipeline will be used. If export\_type is NULL, then the output of plot\_idea() will be returned.

If the input is a list of files and/or a directory, and if type is "single", then the single analysis pipelines are iterated over each file. If export\_type is NULL, then the multiple outputs of plot\_idea() are gathered in an unique list and returned.

If the input is a list of files and/or a directory, and if type is "group\*", then the "import" (read\_idea() |> compute\_idea() or old\_idea()) pipeline is iterated over each file and the results are gathered in an object of class "IDEA\_group\_data". This object introduced in the plot\_idea() |> write\_idea() pipeline will trigger a new algorithm suited to group analysis. If export\_type is NULL, then the output of plot\_idea() will be returned.

## Value

Either reports and/or raw plots in output\_directory or a named list with all the results.

## Examples

```
library(IDEATools)
path <- system.file("example_data/idea_example_1.json", package = "IDEATools")
group_path <- system.file("example_data", package = "IDEATools")
```

```
# Find your temporary directory (the output will be there)
tempdir <- tempdir()

# Run a full individual diagnosis with no export, with only trees
my_diagnosis <- diag_idea(
  input = path,
  output_directory = tempdir,
  type = "single",
  export_type = NULL,
  prefix = "Farm_A",
  plot_choices = "trees",
  dpi = 20, ## Can be much higher
  quiet = TRUE
)
```

---

jsonify2

*Convert IDEA excel files to json, without output*

---

### **Description**

Convert IDEA excel files to json, without output

### **Usage**

```
jsonify2(input, output)
```

### **Arguments**

input	the directory containing the xls/xlsx files to convert
output	the directory where the json files will be created

### **Value**

a list of json files exported to the output directory

---

old\_idea

*Read "old" IDEA data files*

---

### **Description**

This function is an alternative to the read/compute pipeline for older versions of IDEA data excel files.

### **Usage**

```
old_idea(input)
```

## Arguments

**input** a system path to the file containing the IDEA data. The file extension has to be xls or xlsx.

## Details

This function is designed to import data from "old" IDEA data files. It will most probably work for IDEA4 excel files which are later than 2019-01-01. This actually works because this function focuses on indicators directly computed in the excel file rather than items. The potential drawbacks being that no information about items are collected and that some metadata may be missed.

Note : For the farm id metadata, the full First/Last name will be used if found.

## Value

An object of class "IDEA\_data" with three attributes :

**metadata** a named list containing the 17 metadata entries about the farm

**dataset** a tibble containing the score computed for the 53 indicators, 13 components and 3 dimensions

**nodes** a list of tibbles, one per property plus a global one, which all describe the qualitative evaluation

## Examples

```
library(IDEATools)

## Importing from an old IDEA file
input <- "path_to_your_old_file.xlsx"
if(file.exists(input)) {
  computed_data <- old_idea(input)
}
```

---

plot\_idea

*Plot IDEA4 results*

---

## Description

Produces ggplots and/or SVG source code with IDEA data produced by either compute\_idea() or old\_idea() call.

## Usage

```
plot_idea(IDEA_data, choices = c("dimensions", "trees", "radars"))
```

## Arguments

IDEA_data	an IDEA_data or IDEA_group_data object
choices	Which type of plots should be produced ? Can be either "dimensions", "trees" or "radars" or a combination of these 3. Ignored if IDEA_data is of class IDEA_group_data.

## Details

This function will produce different plots depending on whether the input data is of class IDEA\_data or IDEA\_group\_data.

The IDEA\_data class implies that the data comes from an individual analysis pipeline, so individual plots will be produced according to the user's choices specified in the function call :

**dimensions** This option will produce histograms for dimensions, components, and indicators along with a polarised synthetic representation for components.

**trees** This option will produce the colored trees describing the qualitative agregation in the property approach.

**radars** This option will produce polarised histograms (also called 'radars') giving the score (in %) of each indicator, grouped by property

Important note : All 3 types of plots are required for the "report" option of the write\_idea() function.

The IDEA\_group\_data class can only be generated by a "diag\_idea()" group call which will iterate either "read\_idea() |> compute\_idea()" or "old\_idea()" on each data file before aggregating results and assigning this grouped class.

As the IDEA\_group\_data class implies that the data comes from a group analysis pipeline, the choices argument will be ignored. The plots produced for the dimension approach are almost the same as in the individual analysis pipeline but with the histograms being replaced by boxplots. Concerning properties, the only visualization currently considered as relevant is a matrix (or heatmap) of properties \* farms, with the cells colored according to the qualitative evaluation for each farm for a given property.

Note that plots are using a "theme\_idea()" theme defined in this package.

A copy of the blank canvas used for colored trees can be locally exported as svg files with :

```
IDEATools::show_canvas()
```

Further information about the colored trees ans canvas can be found in this vignette :

```
vignette("colored_trees", package = "IDEATools")
```

## Value

a named list of plots of class IDEA\_plots or IDEA\_group\_plots. The algorithm also adds a "data" attribute containing data introduced in the input of this function.

## Examples

```
library(IDEATools)

## Example given for a single analysis. See diag_idea() for a group analysis.
path <- system.file("example_data/idea_example_1.json", package = "IDEATools")
my_data <- read_idea(path)
computed_data <- compute_idea(my_data)

## Example without radars or dimensions
idea_plots <- plot_idea(computed_data, choices = c("trees"))
```

---

read\_idea

*Read IDEA4 items and metadata*

---

## Description

Reads and imports items and farm metadata from .xls, .xlsx or .json files containing IDEA4 data.

## Usage

```
read_idea(input)
```

## Arguments

**input** a system path to the file containing the IDEA data. The file extension can either be json or xls(x). For the latter, the version number must be  $\geq 4.2.0$

## Details

This function is designed to import items and farm metadata from a single IDEA data file. Errors will be produced if the input file does not contain any "metadonnees" field in the case of a json file or any "Notice" sheet in the case of an excel file.

The R code has been developed according to the newest versions of IDEA data collecting files (version numbers  $\geq 4.2.0$ ) and will produce an error if the version number is lower than 4.2.0 or can't be found in the 'Notice\$K4' cell of the input in the case of excel input. There are no limitations for json input files as they were introduced after version number 4.2.0.

For some older versions (from about 2019-01-01), you can replace the " read\_idea() |> compute\_idea() " pipeline by "old\_idea()" which will focus on indicators rather than items.

## Value

An object of class `IDEA_items` with two attributes :

**metadata** a named list containing the 17 metadata entries about the farm

**items** a tibble with the extracted 118 items found in the input

**Examples**

```
library(IDEATools)
path <- system.file("example_data/idea_example_1.json", package = "IDEATools")
my_data <- read_idea(path)
my_data
```

---

runGUI	<i>Minimal Shiny app for IDEATools</i>
--------	--

---

**Description**

This function loads a graphical user interface (GUI) to use IDEATools.

**Usage**

```
runGUI()
```

**Value**

Loads a shiny application

**Examples**

```
library(IDEATools)

## Only run this example in interactive R sessions
if (interactive()) {
  runGUI()
}
```

---

show_decision_rules	<i>Show decision rules</i>
---------------------	----------------------------

---

**Description**

Show decision rules

**Usage**

```
show_decision_rules(directory)
```

**Arguments**

directory      the directory where to output the decision rules

**Value**

Exports an excel file in the desired directory

**Examples**

```
library(IDEATools)
show_decision_rules(tempdir())
```

---

show\_tree\_structure     *Show the reference table used for building colored trees*

---

**Description**

Show the reference table used for building colored trees

**Usage**

```
show_tree_structure(directory)
```

**Arguments**

directory     the directory where to output the reference tables

**Value**

Exports an excel file in the desired directory

**Examples**

```
library(IDEATools)
show_tree_structure(tempdir())
```

---

theme\_idea     *IDEA ggplot2 theme*

---

**Description**

IDEA ggplot2 theme

**Usage**

```
theme_idea(base_size = 15, base_family = "")
```

**Arguments**

base\_size      base size  
 base\_family    base family

**Value**

a ggplot2 theme for IDEA

---

write_idea	<i>Write IDEA4 results and graphs to local disc</i>
------------	---

---

**Description**

This function allows the user to write IDEA4 results either as local PNG files or compiled in reports.

**Usage**

```
write_idea(
  IDEA_plots,
  output_directory,
  type = c("local", "report"),
  prefix = NULL,
  dpi = 320,
  report_format = "docx",
  append = FALSE,
  input_file_append = NULL,
  quiet = FALSE
)
```

**Arguments**

IDEA\_plots      an IDEA\_plots or IDEA\_group\_plots object from a plot\_idea() call.

output\_directory      the desired output directory for the rendered reports and/or plots. Defaults to "IDEATools\_output"

type              the type of output to produce. Can be either "report" to produce compiled reports or "local" to write raw plots as PNG files.

prefix            a prefix which will be added to output files names. Typically, the name of the farm. Ignored in the case of a group analysis : The metadata\$MTD\_01 field will then be used to identify each farm.

dpi                ggplot output resolution.

report\_format    a string indicating the output format if type = "report". Can be a single format (e.g "pdf") or multiple formats (e.g. c("pdf", "xlsx")). Possible formats are "pdf", "docx", "pptx" and "xlsx"

append	If the input is an xlsx format, should the individual output be appended to the original file ?
input_file_append	file path to an xlsx IDEA data spreadsheet
quiet	A command to remove console printing.

## Details

This function automatically creates in `output_directory` a subdirectory named after the system date for users to use the same `output_directory` for multiple diagnosis.

Inputs of class `IDEA_plots` can be generated by a classic `read_idea() |> compute_idea() |> plot_idea()` or `old_idea() |> plot_idea()` individual analysis pipelines. Inputs of class `IDEA_group_plots` can only be generated by a `plot_idea()` call in a group analysis conducted by `diag_idea`.

In the case of an individual analysis, another subdirectory is created with `prefix` as a name so that analyses are not mixed up. The user can choose if output should be raw plots (that can be used in custom reports) or pre-compiled reports with a large variety of available formats.

In the case of a group analysis, another subdirectory is created with a name like "Groupe\_number\_of\_farms" so that analyses are not mixed up. The user can again choose if output should be raw plots or pre-compiled reports.

If the `report_format` argument is set to either "docx" or "pptx", the report will be rendered using the `rmarkdown` package (and `officedown/officer` packages for the docx output) using a template stored in this package. For "pdf" output, LaTeX will be called with the `rmarkdown` package. For "xlsx" output, the `openxlsx` package will be used to sequentially produce Excel worksheets and files, using an internal R function.

Please note that an error will be produced if the input object does not contain all three "dimensions", "trees" and "radars" entries in the case of an individual analysis and if `type = "report"`.

## Value

Reports and/or raw plots in `output_directory`.

## Examples

```
library(IDEATools)
path <- system.file("example_data/idea_example_1.json", package = "IDEATools")
my_data <- read_idea(path)
computed_data <- compute_idea(my_data)
# Only plotting the radars as a minimal example
idea_plots <- plot_idea(computed_data, choices = "radars")
# Find your temporary directory
tempdir <- tempdir()

# Export as raw plots to your tempdir
write_idea(idea_plots,
  output_directory = tempdir,
  type = "local",
  prefix = "myFarm",
  dpi = 20 # Can be much higher
)
```

# Index

`compute_idea`, [2](#)

`diag_idea`, [3](#)

`jsonify2`, [5](#)

`old_idea`, [5](#)

`plot_idea`, [6](#)

`read_idea`, [8](#)

`runGUI`, [9](#)

`show_decision_rules`, [9](#)

`show_tree_structure`, [10](#)

`theme_idea`, [10](#)

`write_idea`, [11](#)