# Package: DIETCOST (via r-universe)

October 16, 2026

**Type** Package

**Title** Calculate the Cost and Environmental Impact of a Ideal Diet

**Version** 1.0.0.0

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**Imports** readxl, rlang, dplyr, tidyselect, stats, xlsx, magrittr

**Depends** R (>= 2.10)

**LazyData** true

**Description** Easily perform a Monte Carlo simulation to evaluate the cost and carbon, ecological, and water footprints of a set of ideal diets. Pre-processing tools are also available to quickly treat the data, along with basic statistical features to analyze the simulation results — including the ability to establish confidence intervals for selected parameters, such as nutrients and price/emissions. A 'standard version' of the datasets employed is included as well, allowing users easy access to customization. This package brings to R the 'Python' software initially developed by Vandevijvere, Young, Mackay, Swinburn and Gahegan (2018) <doi:10.1186/s12966-018-0648-6>.

**License** MIT + file LICENSE

**URL** https://github.com/hbracarense/dietcost

**BugReports** https://github.com/hbracarense/dietcost/issues

**NeedsCompilation** no

**Author** Henrique Bracarense [cre, aut] (ORCID: <https://orcid.org/0009-0001-5964-9969>), Thais Marquezine [aut] (ORCID: <https://orcid.org/0000-0002-9415-5817>), Rafael Claro [aut] (ORCID: <https://orcid.org/0000-0001-9690-575X>)

**Maintainer** Henrique Bracarense <hbracarense@hotmail.com>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2025-05-09 14:10:16 UTC

**RemoteUrl** https://github.com/cran/DIETCOST

**RemoteRef** HEAD

**RemoteSha** a523dc8fe73721988674473fc0743816dc9b101e

# Contents

add_float_range . . . 3
add_range . . . 4
addConstraintData . . . 4
addEmissionData . . . 5
addFoodGroupsConstraintData . . . 5
addNutrientData . . . 6
addPriceData . . . 6
calculateGroupedResults . . . 7
calculateResults . . . 7
check_function . . . 8
check_id_defined . . . 8
check_match_food_price . . . 9
check_match_individual_diet . . . 9
check_min_exists . . . 10
check_nom_num_df . . . 10
check_non_num . . . 11
check_spelling . . . 11
check_variety . . . 12
checkLinkedFoods . . . 12
checks_optional_food_groups . . . 13
checkZeroDiff . . . 13
converts_dataframe . . . 14
convertWeeklyFoodGroups . . . 14
convertWeeklyNutrientTargets . . . 15
createFoodData . . . 15
createFoodGroupData . . . 16
createNutrientTargets . . . 16
createRandomMeal . . . 17
diff_calc . . . 18
energy_conversor . . . 19
food_groups . . . 19
foodData . . . 22
foodGroupData . . . 23
foods . . . 23
getDifference . . . 25
getFoodGroupServes . . . 25
getNutrients . . . 26
getPerc . . . 26
join_function . . . 27
monteCarlo . . . 27
monteCarloSimulation . . . 29

nutrient_targets . . . . . 30
nutrientDataCalculation . . . . . 32
permitted_individuals . . . . . 32
priceEmissionData . . . . . 33
printResults . . . . . 33
random_plan . . . . . 34
redmeat_check . . . . . 34
remove_suffix . . . . . 35
sample_safe . . . . . 35
sauces_protein_discretionary_change . . . . . 36
standard_name_check . . . . . 36
starchy_fill . . . . . 37
treat_df . . . . . 37
treat_groups_df . . . . . 38
unique_values . . . . . 38
upload_data . . . . . 39

**Index** **40**

---

`add_float_range` *Float range*

---

## Description

Checks if a numeric variable is within a continuous float range.

## Usage

```
add_float_range(variable, min, max)
```

## Arguments

| | |
|---|---|
| `variable` | Numeric variable. |
| `min` | Minimum possible value. |
| `max` | Maximum possible value. |

## Value

No return value,only performs a check.

## add_range *Discrete range*

### Description

Checks if a variable is within a discrete range.

### Usage

```
add_range(variable, range, message)
```

### Arguments

| | |
|---|---|
| variable | variable. |
| range | Allowed range. |
| message | Message to be printed in case of failure. |

### Value

No return value, only performs a check.

## addConstraintData *Food constraint data addition*

### Description

Adds nutrients constraint data, according to chosen diet, to foods dataframe.

### Usage

```
addConstraintData(filepath, df, diets, max_scale, override_min = NULL)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored.. |
| df | Foods dataframe. |
| diets | Chosen diets. Constraint sheets in foods dataset must be of format 'constraints_DIETNAME_diet_foods', then the parameter passed will be DIETNAME. Can be a vector of diets in format c('DIETNAME1','DIETNAME2',...,'DIETNAMEN'). |
| max_scale | Maximum scale. |
| override_min | If is not null, overrides all minimum values. |

### Value

Foods dataframe with constraints columns.

## addEmissionData *Emission data addition*

### Description

Adds emission data to foods dataframe.

### Usage

```
addEmissionData(filepath, df, emission_cols = NULL)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored. |
| df | Foods dataframe. |
| emission_cols | Optional parameter. Emission column names if standard dataset isn't used. |

### Value

Food dataframe with emission data.

## addFoodGroupsConstraintData *Food group constraint data addition*

### Description

Adds serves constraints to food groups dataframe

### Usage

```
addFoodGroupsConstraintData(filepath, df, diets)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored.. |
| df | Food groups dataframe. |
| diets | Chosen diets. Constraint sheets in foods dataset must be of format 'constraints_DIETNAME_diet_food_g then the parameter passed will be DIETNAME. Can be a vector of diets in format c('DIETNAME1','DIETNAME2',...,'DIETNAMEN'). |

### Value

Food groups dataframe with added constraint data.

## addNutrientData *Nutrients data addition*

### Description

Adds nutrients data to foods dataframe.

### Usage

```
addNutrientData(filepath, df)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored.. |
| df | Foods dataframe. |

### Value

Foods dataframe with nutrient columns.

## addPriceData *Price data addition*

### Description

Adds price data to foods dataframe.

### Usage

```
addPriceData(filepath, df)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored.. |
| df | Foods dataframe. |

### Value

Foods dataframe with added price data.

`calculateGroupedResults` *Calculates grouped results for a Monte Carlo Simulation*

## Description

Calculates a confidence interval for price and footprints obtained through a Monte Carlo Simulation, grouped by food groups. This function should be employed only if the standard table supplied with this package is utilized. Prints a .xlsx file in the home directory.

## Usage

```
calculateGroupedResults(path_file, report_path, confidence_interval)
```

## Arguments

| | |
|---|---|
| `path_file` | A string containing the path to the folder containing the .csv files created in the monteCarlo function. |
| `report_path` | A string containing the path to where the report will be saved. |
| `confidence_interval` | A float. Must be either 0.01, 0.05 or 0.1. |

## Value

No R object return, prints an Excel workbook.

`calculateResults` *Calculates results for a Monte Carlo Simulation*

## Description

Calculates a confidence interval for several parameters obtained through a Monte Carlo Simulation. This function should be employed only if the standard table supplied with this package is utilized. Prints a .xlsx file in the home directory.

## Usage

```
calculateResults(path_file, report_path, confidence_interval)
```

## Arguments

| | |
|---|---|
| `path_file` | A string containing the path to the folder containing the .csv files created in the monteCarlo function. |
| `report_path` | A string containing the path to where the report will be saved. |
| `confidence_interval` | A float. Must be either 0.01, 0.05 or 0.1. |

### Value

No R object return, prints an Excel workbook.

---

`check_function` *Missing value check*

---

### Description

Checks if there are any missing values in a given column from the dataset.

### Usage

```
check_function(name, column)
```

### Arguments

| | |
|---|---|
| `name` | Column in which missing values will be sought. |
| `column` | Column name, in string format. |

### Value

No return, only performs a check.

---

`check_id_defined` *ID mismatch check*

---

### Description

Checks if a given food has an ID assigned but is absent in another dataset.

### Usage

```
check_id_defined(df1, df2, value)
```

### Arguments

| | |
|---|---|
| `df1` | First dataframe. |
| `df2` | Second dataframe. |
| `value` | Dataset name. |

### Value

No return, only performs a check.

## check_match_food_price

*Food/price mismatch check*

### Description

Checks if all foods have a price.

### Usage

```
check_match_food_price(df)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |

### Value

No return, only performs a check.

## check_match_individual_diet

*Individual/diet mismatch check*

### Description

Checks if all individuals have a matching diet.

### Usage

```
check_match_individual_diet(df)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |

### Value

No return, only performs a check.

## check_min_exists *Minimum intake food groups check*

### Description

Looks for a minimum value and sets zero if missing,

### Usage

```
check_min_exists(df, check, col)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| check | Boolean variable to permit optional food group. |
| col | Minimum percentage intake column name. |

### Value

Dataframe.

## check_nom_num_df *Applies non-nummeric value check to entire dataframe*

### Description

Checks if values supposed to be numeric are in fact numeric.

### Usage

```
check_nom_num_df(df)
```

### Arguments

| | |
|---|---|
| df | Dataframe columns. |

### Value

No return, only performs a check.

## check_non_num *Non-numeric check*

### Description

Checks if values supposed to be numeric are in fact numeric.

### Usage

```
check_non_num(df)
```

### Arguments

| | |
|---|---|
| df | Dataframe column. |

### Value

No return, only performs a check.

## check_spelling *Spellcheck*

### Description

Checks if two datasets have the same spelling in names column.

### Usage

```
check_spelling(df1, df2, condition)
```

### Arguments

| | |
|---|---|
| df1 | First dataframe. |
| df2 | Second dataframe. |
| condition | Column to be joined. |

### Value

No return, only performs a check.

## check_variety *Variety check*

### Description

Checks if varieties are into the allowed range (1,2 or 3).

### Usage

```
check_variety(df)
```

### Arguments

| | |
|---|---|
| df | Dataframe variety column. |

### Value

No return, only performs a check.

## checkLinkedFoods *Linked foods check*

### Description

Checks if lower bound linked foods serves are lower or equal to higher bound linked foods serves.

### Usage

```
checkLinkedFoods(df, low, high)
```

### Arguments

| | |
|---|---|
| df | Random meal plan. |
| low | Vector of lower bound food IDs. |
| high | Vector of higher bound food IDs. |

### Value

Differences dataframe.

## checks_optional_food_groups *Optional food groups check*

### Description

If discretionary foods, alcohol or takeaway are permitted, looks for a minimum value and sets zero if missing,

### Usage

```
checks_optional_food_groups(check, value)
```

### Arguments

| | |
|---|---|
| check | Boolean variable to permit optional food group. |
| value | Minimum percentage of energy intake from optional food group. |

### Value

Minimum percentage of energy intake from optional food group.

## checkZeroDiff *All zero difference check*

### Description

Checks if differences dataframe is all zeroes.

### Usage

```
checkZeroDiff(diff)
```

### Arguments

| | |
|---|---|
| diff | Differences dataframe |

### Value

Boolean. TRUE if all zeroes, FALSE otherwise.

converts_dataframe *Weekly conversion*

## Description

Converts data from daily to weekly

## Usage

```
converts_dataframe(df, exclusion_cols)
```

## Arguments

| | |
|---|---|
| df | Dataframe. |
| exclusion_cols | Columns (non-numerical or percentage) that conversion won't be applied. |

## Value

Weekly dataframe.

convertWeeklyFoodGroups *Food group serves conversion*

## Description

Converts food group serves dataframe to weekly values.

## Usage

```
convertWeeklyFoodGroups(df, diet, individual)
```

## Arguments

| | |
|---|---|
| df | Food group serves dataframe. |
| diet | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_food_groups' sheet in dataset. |
| individual | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |

## Value

Converted food group serves dataframe.

## Examples

```
food_groups_wk <- convertWeeklyFoodGroups(DIETCOST::food_groups, 'C', 'man');
```

## convertWeeklyNutrientTargets *Nutrient targets conversion*

### Description

Converts nutrient targets dataframe to weekly values.

### Usage

```
convertWeeklyNutrientTargets(df, diet, person, nutrient_constraints = NULL)
```

### Arguments

| | |
|---|---|
| df | Nutrient targets dataframe. |
| diet | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_foods' sheet in dataset. |
| person | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |
| nutrient_constraints | Optional parameter. Vector of nutrients column names to be used if not all nutrients are to be used as constraints. |

### Value

Converted nutrient targets dataframe.

### Examples

```
nutrient_targets_wk <- convertWeeklyNutrientTargets(DIETCOST::nutrient_targets, 'C', 'man')
```

## createFoodData *Food data creation*

### Description

Creates a food data dataframe

### Usage

```
createFoodData(filepath, redmeat_ids)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored. |
| redmeat_ids | Vector of redmeat IDs. |

### Value

Food dataframe.

---

`createFoodGroupData` *Food group data creation*

---

### Description

Creates and populates a food group data dataframe

### Usage

```
createFoodGroupData(df)
```

### Arguments

| | |
|---|---|
| `df` | Foods dataframe. |

### Value

Food group dataframe.

---

`createNutrientTargets` *Nutrients data addition*

---

### Description

Adds nutrients data to foods dataframe.

### Usage

```
createNutrientTargets(
  filepath,
  allow_alcohol = TRUE,
  allow_discretionary = TRUE,
  allow_takeaway = TRUE,
  alcohol_perc_max = NULL,
  discretionary_perc_max = NULL,
  takeaway_perc_max = NULL
)
```

### Arguments

| | |
|---|---|
| `filepath` | Path in which the dataset, in .xlsx format, is stored.. |
| `allow_alcohol` | Boolean variable checking if alcohol is permitted. Default TRUE. |
| `allow_discretionary` | Boolean variable checking if discretionary foods are permitted. Default TRUE. |
| `allow_takeaway` | Boolean variable checking if takeaway is permitted. Default TRUE. |
| `alcohol_perc_max` | Optional parameter. Defines maximum energy intake derived from alcohol. |
| `discretionary_perc_max` | Optional parameter. Defines maximum energy intake derived from discretionary foods. |
| `takeaway_perc_max` | Optional parameter. Defines maximum energy intake derived from takeaway. |

### Value

Nutrient targets dataframe.

---

## `createRandomMeal` *Random meal plan*

---

### Description

Creates a random meal plan.

### Usage

```
createRandomMeal(
  foods_df,
  targets_df,
  person,
  diet,
  allowed_varieties,
  min_serve_size_difference,
  allow_discretionary = TRUE,
  allow_alcohol = TRUE,
  allow_takeaway = TRUE,
  emission_cols = NULL,
  nutrient_cols = NULL
)
```

### Arguments

| | |
|---|---|
| foods_df | Foods dataframe. |
| targets_df | Nutrient targets dataframe. |
| person | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |
| diet | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_foods' sheet in dataset. |
| allowed_varieties | Permitted food varieties. Can be a vector of the following: 1,2 and/or 3. |
| min_serve_size_difference | Multiplier to serve difference. A float between 0 and 1. |
| allow_discretionary | Boolean variable checking if discretionary foods are permitted. Default TRUE. |
| allow_alcohol | Boolean variable checking if alcohol is permitted. Default TRUE. |
| allow_takeaway | Boolean variable checking if takeaway is permitted. Default TRUE. |
| emission_cols | Optional parameter. Emission column names if standard dataset isn't used. |
| nutrient_cols | Optional parameter. Nutrients column names if standard dataset isn't used. |

### Value

Random meal plan dataframe.

### Examples

```
foods_df <- createRandomMeal(foods_df = DIETCOST::foods,
                             targets_df = DIETCOST::nutrient_targets,
                             person = 'man',
                             diet = 'C',
                             allowed_varieties = c(1,2,3),
                             min_serve_size_difference = 0.5,
                             allow_takeaway = TRUE,
                             allow_alcohol = TRUE,
                             allow_discretionary = TRUE)
```

---

## diff_calc *Difference calculator*

---

### Description

Calculates difference between values of random meal plan created and targets logged.

### Usage

```
diff_calc(val, min, max)
```

### Arguments

| | |
|---|---|
| val | Value to be evalueted. |
| min | Minimum constraint. |
| max | Maximum constraint. |

### Value

Difference.

---

## energy_conversor *MJ to KJ conversion*

---

### Description

Converts energy values in megajoules (MJ) to kilojoules (KJ),

### Usage

```
energy_conversor(df, min, max)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| min | Minimum energy column name. Default 'energy_mj_min'. |
| max | Maximum energy column name. Default 'energy_mj_max'. |

### Value

No return, only performs a check.

---

## food_groups *Food groups dataset example*

---

### Description

A set of data containing commonly available food groups based on a Brazilian typical diet.

### Usage

```
food_groups
```

**Format**

A dataframe with 12 rows and 74 columns:

**food_group** Food group, i.e. 'Fruit' or 'Vegetable'
**food_group_id** Numerical code for dood group
**man_min_g_C** Weekly minimal current diet intake for males, in grams
**man_max_g_C** Weekly maximal current diet intake for males, in grams
**man_target_g_C** Weekly target current diet intake for males, in grams
**man_min_serve_C** Weekly minimal current diet intake for males, in serves
**man_max_serve_C** Weekly maximal current diet intake for males, in serves
**man_target_serve_C** Weekly target current diet intake for males, in serves
**woman_min_g_C** Weekly minimal current diet intake for females, in grams
**woman_max_g_C** Weekly maximal current diet intake for females, in grams
**woman_target_g_C** Weekly target current diet intake for females, in grams
**woman_min_serve_C** Weekly minimal current diet intake for females, in serves
**woman_max_serve_C** Weekly maximal current diet intake for females, in serves
**woman_target_serve_C** Weekly target current diet intake for females, in serves
**boy_min_g_C** Weekly minimal current diet intake for boys, in grams
**boy_max_g_C** Weekly maximal current diet intake for boys, in grams
**boy_target_g_C** Weekly target current diet intake for boys, in grams
**boy_min_serve_C** Weekly minimal current diet intake for boys, in serves
**boy_max_serve_C** Weekly maximal current diet intake for boys, in serves
**boy_target_serve_C** Weekly target current diet intake for boys, in serves
**girl_min_g_C** Weekly minimal current diet intake for girls, in grams
**girl_max_g_C** Weekly maximal current diet intake for girls, in grams
**girl_target_g_C** Weekly target current diet intake for girls, in grams
**girl_min_serve_C** Weekly minimal current diet intake for girls, in serves
**girl_max_serve_C** Weekly maximal current diet intake for girls, in serves
**girl_target_serve_C** Weekly target current diet intake for girls, in serves
**man_min_g_PF** Weekly minimal EAT-Lancet diet intake for males, in grams
**man_max_g_PF** Weekly maximal EAT-Lancet diet intake for males, in grams
**man_target_g_PF** Weekly target EAT-Lancet diet intake for males, in grams
**man_min_serve_PF** Weekly minimal EAT-Lancet diet intake for males, in serves
**man_max_serve_PF** Weekly maximal EAT-Lancet diet intake for males, in serves
**man_target_serve_PF** Weekly target EAT-Lancet diet intake for males, in serves
**woman_min_g_PF** Weekly minimal EAT-Lancet diet intake for females, in grams
**woman_max_g_PF** Weekly maximal EAT-Lancet diet intake for females, in grams
**woman_target_g_PF** Weekly target EAT-Lancet diet intake for females, in grams

**woman_min_serve_PF** Weekly minimal EAT-Lancet diet intake for females, in serves
**woman_max_serve_PF** Weekly maximal EAT-Lancet diet intake for females, in serves
**woman_target_serve_PF** Weekly target EAT-Lancet diet intake for females, in serves
**boy_min_g_PF** Weekly minimal EAT-Lancet diet intake for boys, in grams
**boy_max_g_PF** Weekly maximal EAT-Lancet diet intake for boys, in grams
**boy_target_g_PF** Weekly target EAT-Lancet diet intake for boys, in grams
**boy_min_serve_PF** Weekly minimal EAT-Lancet diet intake for boys, in serves
**boy_max_serve_PF** Weekly maximal EAT-Lancet diet intake for boys, in serves
**boy_target_serve_PF** Weekly target EAT-Lancet diet intake for boys, in serves
**girl_min_g_PF** Weekly minimal EAT-Lancet diet intake for girls, in grams
**girl_max_g_PF** Weekly maximal EAT-Lancet diet intake for girls, in grams
**girl_target_g_PF** Weekly target EAT-Lancet diet intake for girls, in grams
**girl_min_serve_PF** Weekly minimal EAT-Lancet diet intake for girls, in serves
**girl_max_serve_PF** Weekly maximal EAT-Lancet diet intake for girls, in serves
**girl_target_serve_PF** Weekly target EAT-Lancet diet intake for girls, in serves
**man_min_g_H** Weekly minimal healthy diet intake for males, in grams
**man_max_g_H** Weekly maximal healthy diet intake for males, in grams
**man_target_g_H** Weekly target healthy diet intake for males, in grams
**man_min_serve_H** Weekly minimal healthy diet intake for males, in serves
**man_max_serve_H** Weekly maximal healthy diet intake for males, in serves
**man_target_serve_H** Weekly target healthy diet intake for males, in serves
**woman_min_g_H** Weekly minimal healthy diet intake for females, in grams
**woman_max_g_H** Weekly maximal healthy diet intake for females, in grams
**woman_target_g_H** Weekly target healthy diet intake for females, in grams
**woman_min_serve_H** Weekly minimal healthy diet intake for females, in serves
**woman_max_serve_H** Weekly maximal healthy diet intake for females, in serves
**woman_target_serve_H** Weekly target healthy diet intake for females, in serves
**boy_min_g_H** Weekly minimal healthy diet intake for boys, in grams
**boy_max_g_H** Weekly maximal healthy diet intake for boys, in grams
**boy_target_g_H** Weekly target healthy diet intake for boys, in grams
**boy_min_serve_H** Weekly minimal healthy diet intake for boys, in serves
**boy_max_serve_H** Weekly maximal healthy diet intake for boys, in serves
**boy_target_serve_H** Weekly target healthy diet intake for boys, in serves
**girl_min_g_H** Weekly minimal healthy diet intake for girls, in grams
**girl_max_g_H** Weekly maximal healthy diet intake for girls, in grams
**girl_target_g_H** Weekly target healthy diet intake for girls, in grams
**girl_min_serve_H** Weekly minimal healthy diet intake for girls, in serves
**girl_max_serve_H** Weekly maximal healthy diet intake for girls, in serves
**girl_target_serve_H** Weekly target healthy diet intake for girls, in serves

## Source

Elaborated by authors.

---

`foodData` *Single-function food dataframe creation*

---

## Description

Creates foods dataframe, with emission, nutrients, constraints and price data, in a single function.

## Usage

```
foodData(
  filepath = filepath,
  redmeat_ids,
  diets,
  max_scale,
  emission_cols = NULL,
  override_min = NULL
)
```

## Arguments

| | |
|---|---|
| `filepath` | Path in which the dataset, in .xlsx format, is stored.. |
| `redmeat_ids` | Vector of unique food IDs that are redmeat. |
| `diets` | Chosen diets. Constraint sheets in foods dataset must be of format 'constraints_DIETNAME_diet_foods', then the parameter passed will be DIETNAME. Can be a vector of diets in format c('DIETNAME1','DIETNAME2',...,'DIETNAMEN'). |
| `max_scale` | Maximum scale. Default is two. |
| `emission_cols` | Optional parameter. Emission column names if standard dataset isn't used. |
| `override_min` | If is not null, overrides all minimum values |

## Value

Foods dataframe.

## foodGroupData *Single-function food group dataframe creation*

### Description

Creates food groups dataframe, with constraints data, in a single function.

### Usage

```
foodGroupData(filepath, df_foods, diets)
```

### Arguments

| | |
|---|---|
| filepath | Path in which the dataset, in .xlsx format, is stored. |
| df_foods | Foods dataframe. |
| diets | Chosen diets. Constraint sheets in foods dataset must be of format 'constraints_DIETNAME_diet_food_g then the parameter passed will be DIETNAME. Can be a vector of diets in format c('DIETNAME1','DIETNAME2',...,'DIETNAMEN'). |

### Value

Food groups dataframe.

## foods *Foods dataset example*

### Description

A set of data containing commonly available foods based on a Brazilian typical diet.

### Usage

```
foods
```

### Format

A dataframe with 99 rows and 45 columns:

**food_group** Food group, i.e. 'Fruit' or 'Vegetable'

**food_group_id** Numerical code for dood group

**food_name** Food name, i.e. LEMON

**food_id** Unique numerical food id

**variety** Variety. Must be 1, 2 or 3

**redmeat** Boolean redmeat identifier

**CF_gCO2eq** Carbon footprint
**WF_l** Water footprint
**EF_g_m2** Ecological footprint
**serve_size_C** Serve size for current diet, in grams
**man_min_C** Minimal current diet intake for males, in grams
**woman_min_C** Minimal current diet intake for females, in grams
**boy_min_C** Minimal current diet intake for boys, in grams
**girl_min_C** Minimal current diet intake for girls, in grams
**man_max_C** Maximal current diet intake for males, in grams
**woman_max_C** Maximal current diet intake for females, in grams
**boy_max_C** Maximal current diet intake for boys, in grams
**girl_max_C** Maximal current diet intake for girls, in grams
**serve_size_PF** Serve size for EAT-Lancet diet, in grams
**man_min_PF** Minimal EAT-Lancet diet intake for males, in grams
**woman_min_PF** Minimal EAT-Lancet diet intake for females, in grams
**boy_min_PF** Minimal EAT-Lancet diet intake for boys, in grams
**girl_min_PF** Minimal EAT-Lancet diet intake for girls, in grams
**man_max_PF** Maximal EAT-Lancet diet intake for males, in grams
**woman_max_PF** Maximal EAT-Lancet diet intake for females, in grams
**boy_max_PF** Maximal EAT-Lancet diet intake for boys, in grams
**girl_max_PF** Maximal EAT-Lancet diet intake for girls, in grams
**serve_size_H** Serve size for healthy diet, in grams
**man_min_H** Minimal healthy diet intake for males, in grams
**woman_min_H** Minimal healthy diet intake for females, in grams
**boy_min_H** Minimal healthy diet intake for boys, in grams
**girl_min_H** Minimal healthy diet intake for girls, in grams
**man_max_H** Maximal healthy diet intake for males, in grams
**woman_max_H** Maximal healthy diet intake for females, in grams
**boy_max_H** Maximal healthy diet intake for boys, in grams
**girl_max_H** Maximal healthy diet intake for girls, in grams
**energy_kj_g** Energy content of food, in kJ/g
**fat_g** Fat content of food per grams
**sat_fat_g** Saturated fat content of food per grams
**CHO_g** Carbohydrates content of food per grams
**sugars_g** Sugars content of food per grams
**fibre_g** Fibre content of food per grams
**protein_g** Protein content of food per grams
**sodium_mg** Sodium content of food per miligrams
**price** Price of food per 100g

### Source

Elaborated by authors.

---

`getDifference` *General difference calculation*

---

### Description

Applies difference calculation to entire dataset.

### Usage

```
getDifference(df_target, df_nutrients, merge_col)
```

### Arguments

| | |
|---|---|
| `df_target` | Constraints dataframe. |
| `df_nutrients` | Nutrients/serves from random meal plan dataframe. |
| `merge_col` | Column to join both dataframes. |

### Value

Differences dataframe.

---

`getFoodGroupServes` *Food group serves calculator*

---

### Description

Calculates total food group serves of random meal plan.

### Usage

```
getFoodGroupServes(df)
```

### Arguments

| | |
|---|---|
| `df` | Random meal plan. |

### Value

Food group serves dataframe.

## getNutrients *Nutrients values calculator*

### Description

Calculates nutritional value of meal plan.

### Usage

```
getNutrients(df, nutrient_cols = NULL)
```

### Arguments

| | |
|---|---|
| df | Random meal plan. |
| nutrient_cols | Optional parameter. Vector of nutrients column names to be used if nutrients are different from standard dataset. |

### Value

Nutrients dataframe.

## getPerc *Percentage values calculator*

### Description

Calculates percentage nutrient values.

### Usage

```
getPerc(df_nutri, df_meal)
```

### Arguments

| | |
|---|---|
| df_nutri | Nutrient constraints dataframe. |
| df_meal | Random meal plan |

### Value

Percentage dataframe.

## join_function *Join function*

### Description

Safely performs a left join between two dataframes.

### Usage

```
join_function(df1, df2, condition)
```

### Arguments

| | |
|---|---|
| df1 | First dataframe. |
| df2 | Second dataframe. |
| condition | Column in which the two datframes will be joined. Can be a single string or a vector. |

### Value

Dataframe.

## monteCarlo *Monte Carlo simulation*

### Description

Creates a Monte Carlo simulation to a given number of iterations. A hit meal consists of one that returnz zero difference between nutrient targets and random meal plan, food groups serves and respects lower linked foods serves lower or equal to higher linked foods serves, if existent.

### Usage

```
monteCarlo(
  dir_path,
  iterations,
  foods_df,
  nutrient_targets_df,
  food_group_targets_df,
  person,
  diet,
  allowed_varieties,
  min_serve_size_difference,
  allow_discretionary = TRUE,
  allow_alcohol = TRUE,
```

```
  allow_takeaway = TRUE,
  emission_cols = NULL,
  nutrient_cols = NULL,
  nutrient_constraints = NULL,
  linked_low_1 = NULL,
  linked_high_1 = NULL,
  linked_low_2 = NULL,
  linked_high_2 = NULL
)
```

## Arguments

| | |
|---|---|
| dir_path | A string containing the path where a directory will be created. |
| iterations | Number of iterations. Integer. |
| foods_df | Foods dataframe. |
| nutrient_targets_df | Nutrient constraints dataframe. |
| food_group_targets_df | Food group serves dataframe. |
| person | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |
| diet | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_foods' sheet in dataset. |
| allowed_varieties | Permitted food varieties. Can be a vector of the following: 1,2 and/or 3. |
| min_serve_size_difference | Multiplier to serve difference. A float between 0 and 1. |
| allow_discretionary | Boolean variable checking if discretionary foods are permitted. Default TRUE. |
| allow_alcohol | Boolean variable checking if alcohol is permitted. Default TRUE. |
| allow_takeaway | Boolean variable checking if takeaway is permitted. Default TRUE. |
| emission_cols | Optional parameter. Emission column names if standard dataset isn't used. |
| nutrient_cols | Optional parameter. Nutrients column names if standard dataset isn't used. |
| nutrient_constraints | Optional parameter. Vector of nutrients column names to be used if not all nutrients are to be used as constraints. |
| linked_low_1 | Optional parameter. Vector of lower bound food IDs. |
| linked_high_1 | Optional parameter. Vector of higher bound food IDs. |
| linked_low_2 | Optional parameter. Vector of lower bound food IDs. |
| linked_high_2 | Optional parameter. Vector of higher bound food IDs. |

## Value

List of dataframes, containing results of simulation.

`monteCarloSimulation` *Single-function Monte Carlo simulation and results export.*

## Description

Runs Monte Carlo Simulation and prints results, in .xlsx format, in a single funtion.

## Usage

```
monteCarloSimulation(
  dir_path,
  iterations,
  foods_df,
  nutrient_targets_df,
  food_group_targets_df,
  person,
  diet,
  allowed_varieties,
  min_serve_size_difference,
  allow_discretionary = TRUE,
  allow_alcohol = TRUE,
  allow_takeaway = TRUE,
  emission_cols = NULL,
  nutrient_cols = NULL,
  nutrient_constraints = NULL,
  linked_low_1 = NULL,
  linked_high_1 = NULL,
  linked_low_2 = NULL,
  linked_high_2 = NULL
)
```

## Arguments

| | |
|---|---|
| `dir_path` | A string containing the path where a directory will be created. This same path will hold the reports Excel workbook. |
| `iterations` | Number of iterations. Integer. |
| `foods_df` | Foods dataframe. |
| `nutrient_targets_df` | Nutrient constraints dataframe. |
| `food_group_targets_df` | Food group serves dataframe. |
| `person` | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |
| `diet` | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_foods' sheet in dataset. |

`allowed_varieties`
: Permitted food varieties. Can be a vector of the following: 1,2 and/or 3.

`min_serve_size_difference`
: Multiplier to serve difference. A float between 0 and 1.

`allow_discretionary`
: Boolean variable checking if discretionary foods are permitted. Default TRUE.

`allow_alcohol` Boolean variable checking if alcohol is permitted. Default TRUE.

`allow_takeaway` Boolean variable checking if takeaway is permitted. Default TRUE.

`emission_cols` Optional parameter. Emission column names if standard dataset isn't used.

`nutrient_cols` Optional parameter. Nutrients column names if standard dataset isn't used.

`nutrient_constraints`
: Optional parameter. Vector of nutrients column names to be used if not all nutrients are to be used as constraints.

`linked_low_1` Optional parameter. Vector of lower bound food IDs.

`linked_high_1` Optional parameter. Vector of higher bound food IDs.

`linked_low_2` Optional parameter. Vector of lower bound food IDs.

`linked_high_2` Optional parameter. Vector of higher bound food IDs.

## Value

No R object return. Prints an Excel workbook.

---

# `nutrient_targets` *Nutrients dataset example*

---

## Description

A set of data containing nutrient weekly targets based on a Brazilian typical diet.

## Usage

```
nutrient_targets
```

## Format

A dataframe with 12 rows and 48 columns:

**individual** Person whose nutrient targets will be provided: man, woman, boy or girl

**diet** Diet whose nutrient targets will be provided: current (C), EAT-Lancet (PF) or healthy (H)

**energy_kj_min** Minimal weekly intake of energy, in kJ/g

**energy_kj_max** Maximal weekly intake of energy, in kJ/g

**fat_grams_min** Minimal weekly intake of fat in grams

**fat_grams_max** Maximal weekly intake of fat in grams

**sat_fat_grams_min** Minimal weekly intake of saturated fat in grams
**sat_fat_grams_max** Maximal weekly intake of saturated fat in grams
**CHO_gram_mins** Minimal weekly intake of carbohydrates in grams
**CHO_gram_max** Maximal weekly intake of carbohydrates in grams
**sugars_grams_min** Minimal weekly intake of sugars in grams
**sugars_grams_max** Maximal weekly intake of sugars in grams
**fibre_grams_min** Minimal weekly intake of fibre in grams
**fibre_grams_max** Maximal weekly intake of fibre in grams
**protein_grams_min** Minimal weekly intake of protein in grams
**protein_grams_max** Maximal weekly intake of protein in grams
**sodium_mgrams_min** Minimal weekly intake of sodium in grams
**sodium_mgrams_max** Maximal weekly intake of sodium in grams
**protein_perc_min** Minimal weekly intake of protein in percentage
**protein_perc_max** Maximal weekly intake of protein in percentage
**sat_fat_perc_min** Minimal weekly intake of saturated fat in percentage
**sat_fat_perc_max** Maximal weekly intake of saturated fat in percentage
**fat_perc_min** Minimal weekly intake of fat in percentage
**fat_perc_max** Maximal weekly intake of fat in percentage
**CHO_perc_mins** Minimal weekly intake of carbohydrates in percentage
**CHO_perc_max** Maximal weekly intake of carbohydrates in percentage
**redmeat_grams_min** Minimal weekly intake of red meat in grams
**redmeat_grams_max** Maximal weekly intake of red meat in grams
**fruit_serves_min** Minimal weekly intake of Fruit in serves
**fruit_serves_max** Maximal weekly intake of Fruit in serves
**starchy_veg_serves_min** Minimal weekly intake of Starchy vegetables in serves
**starchy_veg_serves_max** Maximal weekly intake of Starchy vegetables in serves
**veg_serves_min** Minimal weekly intake of Vegetables in serves
**veg_serves_max** Maximal weekly intake of Vegetables in serves
**dairy_serves_min** Minimal weekly intake of Dairy in serves
**dairy_serves_max** Maximal weekly intake of Dairy in serves
**grain_serves_min** Minimal weekly intake of Grains in serves
**grain_serves_max** Maximal weekly intake of Grains in serves
**protein_serves_min** Minimal weekly intake of Protein in serves
**protein_serves_max** Maximal weekly intake of Protein in serves
**sugars_perc_mins** Minimal weekly intake of sugars in percentage
**sugars_perc_max** Maximal weekly intake of sugars in percentage
**alcohol_perc_mins** Minimal weekly intake of Alcohol in percentage
**alcohol_perc_max** Maximal weekly intake of Alcohol in percentage
**discretionary_perc_mins** Minimal weekly intake of Discretionary foods in percentage
**discretionary_perc_max** Maximal weekly intake of Discretionary foods in percentage
**takeaway_perc_mins** Minimal weekly intake of Takeaway in percentage
**takeaway_perc_max** Maximal weekly intake of Takeaway foods in percentage

## Source

Elaborated by authors.

---

`nutrientDataCalculation` *Nutrient data application to random meal plan created*

---

## Description

Applies nutrient data calculation to random meal plan generated.

## Usage

```
nutrientDataCalculation(df, nutrient_cols = NULL)
```

## Arguments

| | |
|---|---|
| `df` | Random meal plan. |
| `nutrient_cols` | Optional parameter. Nutrient column names if standard dataset isn't used. |

## Value

Random meal plan with nutrients calculated.

---

`permitted_individuals` *Permitted individuals check*

---

## Description

Checks if logged individuals are one or all of the following: man, woman, boy or girl.

## Usage

```
permitted_individuals(df)
```

## Arguments

| | |
|---|---|
| `df` | Variable. |

## Value

No R object return, performs only a check.

## priceEmissionData *Price/emission data application to random meal plan created*

### Description

Applies price and emission data calculation to random meal plan generated.

### Usage

```
priceEmissionData(df, emission_cols = NULL)
```

### Arguments

| | |
|---|---|
| df | Random meal plan. |
| emission_cols | Optional parameter. Emission column names if standard dataset isn't used. |

### Value

Random meal plan with price and emissions calculated.

## printResults *Exportation of Monte Carlo results*

### Description

Exports, in .xlsx format, the results of Monte Carlo simulation.

### Usage

```
printResults(file_path, results, person, diet, allowed_varieties, iterations)
```

### Arguments

| | |
|---|---|
| file_path | A string containing the path where the file will be saved. |
| results | List of results |
| person | Individual whose random meal plan will be created to. Can be one of man, woman, boy or girl. |
| diet | Chosen diet. Must be DIETNAME from 'constraints_DIETNAME_diet_foods' sheet in dataset. |
| allowed_varieties | Permitted food varieties. Can be a vector of the following: 1,2 and/or 3. |
| iterations | Number of iterations. Integer. |

### Value

No R object return, prints a Excel workbook.

## random_plan *Random deletion*

### Description

Randomly deletes a food.

### Usage

```
random_plan(df, column, condition)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| column | Column from which decision about removal of values will be made. |
| condition | Condition that, if is true, will enable radom removal. |

### Value

Random meal plan

## redmeat_check *Redmeat flag*

### Description

Sets a boolean redmeat flag column in dataset.

### Usage

```
redmeat_check(id, redmeat_ids)
```

### Arguments

| | |
|---|---|
| id | Food group id column in dataframe. |
| redmeat_ids | Vector of unique food IDs that are redmeat. |

### Value

No R object return, performs only a check.

## remove_suffix *Suffix removal*

### Description

Removes one of two suffixes from column names

### Usage

```
remove_suffix(vector, suffix_1, suffix_2)
```

### Arguments

| | |
|---|---|
| vector | Vector of column names |
| suffix_1 | First suffix to be removed. |
| suffix_2 | Second suffix to be removed. |

### Value

Vector of column names without suffixes.

## sample_safe *Safe sampling*

### Description

Safely extracts a random unitary sample from a vector.

### Usage

```
sample_safe(x)
```

### Arguments

| | |
|---|---|
| x | Vector. |

### Value

Random sample.

### Examples

```
intake <- DIETCOST::sample_safe(c(10,25,37,52,100));
```

## sauces_protein_discretionary_change *Sauces, protein and discretionary food groups treatment*

### Description

Treats above said food name groups to the format used in the package.

### Usage

```
sauces_protein_discretionary_change(group)
```

### Arguments

| | |
|---|---|
| group | Food group column in dataframe. |

### Value

Treated dataframe.

## standard_name_check *Standard name check*

### Description

Checks if variable names are the standard defined into DIETCOST R standard table.

### Usage

```
standard_name_check(df, ...)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| ... | Any number of strings. |

### Value

No R object return, performs only a check.

### Examples

```
standard_name_check(DIETCOST::foods, 'food_id', 'food_name')
```

## starchy_fill *Starchy vegetables serves addition*

### Description

Adds minimum and maximum serves of starchy vegetables.

### Usage

```
starchy_fill(df, starchy_name, serve_identifier, max_identifier)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| starchy_name | Starchy vegetables food group name. Default 'Starchy vegetables'. |
| serve_identifier | Serve column identifier. Default 'serve'. |
| max_identifier | Max column identifier. Default 'max'. |

### Value

Food group dataframe with starchy vegetable minimum and maximum serves columns added.

## treat_df *Pre-treatment of constraint data*

### Description

Pre-treatment of constraints dataframe.

### Usage

```
treat_df(df, min_identifier, max_identifier, suffix, max_scale, override_min)
```

### Arguments

| | |
|---|---|
| df | Dataframe to be treated. |
| min_identifier | Minimum value column identifier. 'Min' in standard dataset. |
| max_identifier | Maximum value column identifier. 'Max' in standard dataset. |
| suffix | Suffix to be added to column name. |
| max_scale | Maximum scale. Default is two. |
| override_min | If is not null, overrides all minimum values. |

### Value

Treated dataframe.

## treat_groups_df *Treatment of food group constraints dataframe*

### Description

Converts weekly food group serves to daily and adds diet suffix to column names.

### Usage

```
treat_groups_df(df, suffix)
```

### Arguments

| | |
|---|---|
| df | Dataframe. |
| suffix | Suffix to be added to column. |

### Value

Treated food group dataframe.

## unique_values *Unique value check*

### Description

Checks if there are non-unique values in dataset.

### Usage

```
unique_values(value, df, value_col, value_name)
```

### Arguments

| | |
|---|---|
| value | Column from which an unique vector will be formed. |
| df | Dataframe in which lies the column to be checked. |
| value_col | Name of the column to be checked, in string format. |
| value_name | Name of the variable tested. |

### Value

No R object return, performs only a check.

upload_data *Data upload*

## Description

Safely uploads data to be processed in DIETCOST software.

## Usage

```
upload_data(filepath, sheet)
```

## Arguments

| | |
|---|---|
| filepath | The filepath in which the dataset, in .xlsx format, is saved. |
| sheet | The sheet of the .xlsx to be read. |

## Value

The dataframe generated by the file which was read.

# Index


∗ **dataset**
    food_groups, 19
    foods, 23
    nutrient_targets, 30

add_float_range, 3
add_range, 4
addConstraintData, 4
addEmissionData, 5
addFoodGroupsConstraintData, 5
addNutrientData, 6
addPriceData, 6

calculateGroupedResults, 7
calculateResults, 7
check_function, 8
check_id_defined, 8
check_match_food_price, 9
check_match_individual_diet, 9
check_min_exists, 10
check_nom_num_df, 10
check_non_num, 11
check_spelling, 11
check_variety, 12
checkLinkedFoods, 12
checks_optional_food_groups, 13
checkZeroDiff, 13
converts_dataframe, 14
convertWeeklyFoodGroups, 14
convertWeeklyNutrientTargets, 15
createFoodData, 15
createFoodGroupData, 16
createNutrientTargets, 16
createRandomMeal, 17

diff_calc, 18

energy_conversor, 19

food_groups, 19
foodData, 22
foodGroupData, 23
foods, 23

getDifference, 25
getFoodGroupServes, 25
getNutrients, 26
getPerc, 26

join_function, 27

monteCarlo, 27
monteCarloSimulation, 29

nutrient_targets, 30
nutrientDataCalculation, 32

permitted_individuals, 32
priceEmissionData, 33
printResults, 33

random_plan, 34
redmeat_check, 34
remove_suffix, 35

sample_safe, 35
sauces_protein_discretionary_change, 36
standard_name_check, 36
starchy_fill, 37

treat_df, 37
treat_groups_df, 38

unique_values, 38
upload_data, 39