Package 'wefnexus'

November 9, 2026

Type Package

Title Water-Energy-Food-Nutrient-Carbon Nexus Analysis for Agronomic Systems

Version 1.1.0

Description Provides functions for analysing Water-Energy-Food-Nutrient-Carbon (WEFNC) nexus interactions in agricultural production systems. Includes functions for computing water use efficiency (WUE), water productivity (WP), and water footprint (WF) including green, blue, and grey components following the methodology of Hoekstra et al. (2011, ISBN:9781849712798). Includes energy budgeting tools for energy use efficiency (EUE), energy return on investment (EROI), net energy (NE), and energy productivity (EP). Computes nutrient use efficiency (NUE) metrics including agronomic efficiency (AE), physiological efficiency (PE), recovery efficiency (RE), and partial factor productivity (PFP) as defined by Dobermann (2007) <https://digitalcommons.unl.edu/agronomyfacpub/316/> and Congreves et al. (2021) <doi:10.3389/fpls.2021.637108>. Estimates carbon footprint (CF), greenhouse gas (GHG) emissions, soil organic carbon (SOC) stocks, and global warming potential (GWP) using Intergovernmental Panel on Climate Change (IPCC) Sixth Assessment Report (AR6) default values (CH4 = 27, N2O = 273) as reported in Forster et al. (2021) <doi:10.1017/9781009157896.009>. Computes composite Water-Energy-Food-Nutrient-Carbon (WEFNC) nexus indices, trade-off correlation matrices, and generates radar and heatmap visualizations for comparing agricultural treatments. Supports conservation agriculture (CA), irrigated and rain-fed systems, and arid and semi-arid production environments. Methods follow Lal (2004) <doi:10.1016/j.envint.2004.03.005> for carbon emissions from farm operations, and Hoover et al. (2023) <doi:10.1016/j.scitotenv.2022.160992> for water use efficiency indicators. Version 1.1.0 adds a one-call workflow for replicated field trials, analysis of variance with standard error of mean (SEm), critical difference (CD) and coefficient of

variation (CV), objective indicator weights by the entropy,
CRITIC (Diakoulaki et al. (1995)
<doi:10.1016/0305-0548(94)00059-H>) and principal component
methods, TOPSIS ranking of treatments (Hwang and Yoon, 1981), and energy budgeting from physical input
quantities with an editable table of energy equivalents.

License GPL (>= 3)

URL https://github.com/lalitrolaniya/wefnexus,
https://lalitrolaniya.github.io/wefnexus/

BugReports https://github.com/lalitrolaniya/wefnexus/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports stats, graphics, grDevices, rlang

Suggests ggplot2 (>= 3.4.0), knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.1.1

Author Lalit Kumar Rolaniya [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8908-1211>),
Hemant Poonia [aut] (ORCID: <https://orcid.org/0000-0002-0945-8793>,
deceased),
Ram Lal Jat [aut] (ORCID: <https://orcid.org/0009-0003-4339-0555>),
Monika Punia [aut] (ORCID: <https://orcid.org/0009-0002-0294-6767>,
deceased),
Raja Ram Choudhary [aut]

Maintainer Lalit Kumar Rolaniya <lalitrolaniya@gmail.com>

Repository CRAN

Date/Publication 2026-11-09 10:00:02 UTC

Contents

wefnexus-package 4
agronomic_efficiency 5
arid_pulse_nexus 6
arid_pulse_trial 8
caloric_yield 9
carbon_efficiency 10
carbon_footprint 11
carbon_sequestration_rate 13
carbon_sustainability_index 14

crop_water_productivity . . . 15
crop_water_stress_index . . . 16
crop_yield_index . . . 17
depleted_fraction . . . 18
energy_budget . . . 19
energy_coefficients . . . 21
energy_input . . . 22
energy_intensity . . . 23
energy_output . . . 24
energy_productivity . . . 25
energy_profitability . . . 26
energy_use_efficiency . . . 27
eroi . . . 28
food_productivity_index . . . 29
ghg_emission . . . 30
global_warming_potential . . . 32
harvest_index . . . 34
human_energy_profitability . . . 35
internal_utilization_efficiency . . . 36
irrigation_efficiency . . . 37
land_equivalent_ratio . . . 38
net_energy . . . 39
nexus_analysis . . . 40
nexus_anova . . . 42
nexus_heatmap . . . 44
nexus_heatmap_gg . . . 45
nexus_index . . . 46
nexus_radar . . . 47
nexus_radar_gg . . . 48
nexus_rank_compare . . . 49
nexus_sensitivity . . . 51
nexus_summary . . . 52
nexus_sustainability_score . . . 54
nexus_topsis . . . 56
nexus_tradeoff . . . 57
nexus_weights . . . 58
normalize_minmax . . . 60
normalize_zscore . . . 61
nutrient_balance . . . 62
nutrient_harvest_index . . . 63
nutrient_use_efficiency . . . 64
partial_factor_productivity . . . 65
physiological_efficiency . . . 66
plot.cf_result . . . 67
plot.energy_budget . . . 68
plot.nexus_analysis . . . 69
plot.nexus_anova . . . 70
plot.nexus_result . . . 71

plot.rank_compare . . . 71
plot.topsis_result . . . 72
plot.tradeoff_matrix . . . 73
plot_sensitivity . . . 73
plot_tradeoff . . . 74
print.cf_result . . . 75
print.energy_budget . . . 76
print.nexus_analysis . . . 76
print.nexus_anova . . . 77
print.nexus_result . . . 77
print.rank_compare . . . 78
print.sustainability_result . . . 78
print.tradeoff_matrix . . . 79
production_efficiency_index . . . 79
protein_yield . . . 80
recovery_efficiency . . . 81
soil_carbon_stock . . . 82
specific_energy . . . 83
summary.nexus_result . . . 84
system_productivity_index . . . 84
water_footprint . . . 85
water_productivity . . . 87
water_use_efficiency . . . 88

Index **90**

`wefnexus-package` *wefnexus: Water-Energy-Food-Nutrient-Carbon Nexus Analysis for Agronomic Systems*

Description

Provides functions for analysing Water-Energy-Food-Nutrient-Carbon (WEFNC) nexus interactions in agricultural systems. The package provides five core modules and a nexus integration layer:

Water Water use efficiency (WUE), water productivity (WP), water footprint (WF) with green/blue/grey components, irrigation efficiency, crop water stress index (CWSI), and crop water productivity (CWP).

Energy Energy budgeting including input-output analysis, energy use efficiency (EUE), energy return on investment (EROI), energy productivity (EP), net energy (NE), specific energy, and energy profitability.

Food Food productivity indices, harvest index (HI), land equivalent ratio (LER), system productivity index (SPI), caloric and protein yield, and production efficiency index (PEI).

Nutrient Nutrient use efficiency (NUE) metrics including agronomic efficiency (AE), physiological efficiency (PE), recovery efficiency (RE), partial factor productivity (PFP), internal utilization efficiency (IUE), and nutrient harvest index (NHI).

Carbon Carbon footprint (CF) with source-wise breakdown, greenhouse gas (GHG) emission estimation, soil organic carbon (SOC) stocks, carbon sequestration rate (CSR), and global warming potential (GWP) using IPCC AR6 values.

Nexus Normalization, composite nexus index, objective weights (`nexus_weights`: entropy, CRITIC, PCA), TOPSIS ranking (`nexus_topsis`), trade-off matrix, weight sensitivity, radar and heatmap plots.

Workflow `nexus_analysis` runs the full analysis on replicated field data; `nexus_anova` gives treatment means with SEm, CD, CV and letter groups; `energy_budget` converts physical input quantities to energy with `energy_coefficients`.

Set `options(wefnexus.verbose = FALSE)` to silence the informational messages of all functions.

Author(s)

Maintainer: Lalit Kumar Rolaniya <lalitrolaniya@gmail.com> (ORCID)

Authors:

- Lalit Kumar Rolaniya <lalitrolaniya@gmail.com> (ORCID)
- Hemant Poonia <pooniaheman80@gmail.com> (ORCID) (deceased)
- Ram Lal Jat <khokharagro1986@gmail.com> (ORCID)
- Monika Punia <monikapunia02@gmail.com> (ORCID) (deceased)
- Raja Ram Choudhary <rajaramchoudhary26@gmail.com>

See Also

Useful links:

- https://github.com/lalitrolaniya/wefnexus
- https://lalitrolaniya.github.io/wefnexus/
- Report bugs at https://github.com/lalitrolaniya/wefnexus/issues

---

agronomic_efficiency *Agronomic Efficiency (AE)*

Description

Increase in economic yield per unit of nutrient applied.

Usage

```
agronomic_efficiency(
  yield_treated,
  yield_control,
  nutrient_applied,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield_treated`	Numeric vector. Yield with nutrient applied (kg/ha).
`yield_control`	Numeric vector. Yield without nutrient, i.e., from control plot (kg/ha).
`nutrient_applied`	Numeric vector. Amount of nutrient applied (kg/ha).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$AE = \frac{Y_f - Y_0}{F}$$

Value

Numeric vector. Agronomic efficiency (kg grain per kg nutrient).

References

Dobermann, A. (2007). Nutrient use efficiency: measurement and management. In *Fertilizer Best Management Practices*, IFA, Paris, pp. 1-28. https://digitalcommons.unl.edu/agronomyfacpub/316/

Fixen, P. et al. (2015). Nutrient/fertilizer use efficiency: Measurement, current situation and trends. In *Managing Water and Fertilizer for Sustainable Agricultural Intensification*, IFA/IWMI/IPNI/IPI, pp. 8-38.

Congreves, K.A. et al. (2021). Nitrogen use efficiency definitions of today and tomorrow. *Frontiers in Plant Science*, 12, 637108. doi:10.3389/fpls.2021.637108

Examples

```
agronomic_efficiency(yield_treated = 4500, yield_control = 3000,
                     nutrient_applied = 120)
```

`arid_pulse_nexus` *Arid Pulse Nexus Dataset*

Description

A sample dataset with six conservation agriculture treatments in an arid pulse-based cropping system from western Rajasthan, India. Includes water, energy, food, nutrient, and carbon data for WEFNC nexus analysis.

Usage

```
arid_pulse_nexus
```

Format

A data frame with 6 rows and 26 variables:

treatment Character. Treatment combination name

tillage Character. Tillage method: CT (conventional), ZT (zero), PB (permanent beds)

irrigation Character. Irrigation method: Flood, Sprinkler, Drip, or SSDI (sub-surface drip)

residue Logical. Whether crop residue was retained

grain_yield Numeric. Grain yield (kg/ha)

straw_yield Numeric. Straw yield (kg/ha)

irrigation_applied Numeric. Irrigation water applied (mm)

effective_rainfall Numeric. Effective rainfall (mm)

total_water Numeric. Total water consumed (mm)

crop_et Numeric. Crop evapotranspiration (mm)

energy_input Numeric. Total energy input (MJ/ha)

energy_output_grain Numeric. Energy output from grain (MJ/ha)

energy_output_straw Numeric. Energy output from straw (MJ/ha)

n_applied Numeric. Nitrogen applied (kg/ha)

p_applied Numeric. Phosphorus applied (kg P2O5/ha)

k_applied Numeric. Potassium applied (kg K2O/ha)

n_uptake Numeric. Total plant nitrogen uptake (kg/ha)

p_uptake Numeric. Total plant phosphorus uptake (kg/ha)

grain_n_uptake Numeric. Grain nitrogen uptake (kg/ha)

diesel_use Numeric. Diesel consumption (L/ha)

electricity_kwh Numeric. Electricity consumption (kWh/ha)

soc_pct Numeric. Soil organic carbon (percent)

bulk_density Numeric. Soil bulk density (Mg/m3)

ghg_emission Numeric. Total GHG emission (kg CO2-eq/ha)

cost_cultivation Numeric. Cost of cultivation (INR/ha)

gross_return Numeric. Gross return (INR/ha)

Source

Simulated dataset based on typical experimental data from ICAR-Indian Institute of Pulses Research, Regional Centre, Bikaner, Rajasthan, India.

Examples

```
data(arid_pulse_nexus)
str(arid_pulse_nexus)
```

`arid_pulse_trial` *Replicated Arid Pulse Trial Dataset*

Description

A plot-level dataset from a simulated randomized complete block design with the six conservation agriculture treatments of `arid_pulse_nexus` and three replications. Intended for examples of `nexus_analysis` and `nexus_anova`.

Usage

```
arid_pulse_trial
```

Format

A data frame with 18 rows and 11 variables:

treatment Character. Treatment combination name

replication Character. Block (R1 to R3)

grain_yield Numeric. Grain yield (kg/ha)

straw_yield Numeric. Straw yield (kg/ha)

total_water Numeric. Total water consumed (mm)

energy_input Numeric. Total energy input (MJ/ha)

energy_output Numeric. Energy output of grain and straw (MJ/ha), using 14.7 and 12.5 MJ/kg

n_applied Numeric. Nitrogen applied (kg/ha)

n_uptake Numeric. Total plant nitrogen uptake (kg/ha)

grain_n_uptake Numeric. Grain nitrogen uptake (kg/ha)

ghg_emission Numeric. Total GHG emission (kg CO2-eq/ha)

Source

Simulated around the treatment means of `arid_pulse_nexus`; the script is in the package source (`data-raw/make_data.R`).

Examples

```
data(arid_pulse_trial)
aggregate(grain_yield ~ treatment, data = arid_pulse_trial, FUN = mean)
```

`caloric_yield` *Caloric Yield*

Description

Total caloric output per hectare from crop production.

Usage

```
caloric_yield(
  yield,
  caloric_value,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`caloric_value`	Numeric vector. Caloric content (kcal/kg).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$CY = Yield \times CaloricValue$$

Value

Numeric vector. Caloric yield (kcal/ha).

Examples

```
caloric_yield(yield = 4500, caloric_value = 3400)
```

carbon_efficiency *Carbon Efficiency*

Description

Crop yield per unit of greenhouse gas emitted.

Usage

```
carbon_efficiency(
  yield,
  carbon_emission,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
carbon_emission	Numeric vector. Total GHG emission (kg CO2-eq/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$CE = \frac{Yield}{CarbonFootprint}$$

Value

Numeric vector. Carbon efficiency (kg yield per kg CO2-eq). Higher values indicate more yield per unit emission.

Examples

```
carbon_efficiency(yield = 4500, carbon_emission = 2500)
```

carbon_footprint *Carbon Footprint of Crop Production*

Description

Estimates the carbon footprint from major emission sources in crop production, with source-wise breakdown and yield-scaled intensity. Uses IPCC AR6 default GWP values.

Usage

```
carbon_footprint(
  diesel_use = 0,
  electricity_use = 0,
  n_fertilizer = 0,
  p_fertilizer = 0,
  k_fertilizer = 0,
  pesticide_use = 0,
  seed_rate = 0,
  n2o_direct = NULL,
  ch4_emission = 0,
  yield = NULL,
  ef_diesel = 2.68,
  ef_electricity = 0.727,
  ef_n_manufacture = 4.96,
  ef_p_manufacture = 1.61,
  ef_k_manufacture = 0.57,
  ef_pesticide = 10.97,
  ef_seed = 0.58,
  gwp_n2o = 273,
  gwp_ch4 = 27,
  ef_n2o = 0.01,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

diesel_use	Numeric. Diesel consumption (L/ha). Default 0.
electricity_use	Numeric. Electricity consumption (kWh/ha). Default 0.
n_fertilizer	Numeric. Nitrogen fertilizer applied (kg N/ha). Default 0.
p_fertilizer	Numeric. Phosphorus fertilizer applied (kg P2O5/ha). Default 0.
k_fertilizer	Numeric. Potassium fertilizer applied (kg K2O/ha). Default 0.
pesticide_use	Numeric. Pesticide active ingredient used (kg a.i./ha). Default 0.
seed_rate	Numeric. Seed rate (kg/ha). Default 0.

`n2o_direct` Numeric. Direct N2O emissions from soil (kg N2O/ha). If `NULL` (default), estimated as `n_fertilizer * ef_n2o * 44/28` (IPCC Tier 1).

`ch4_emission` Numeric. Methane emission (kg CH4/ha). Default 0. Relevant for rice paddies.

`yield` Numeric. Crop yield (kg/ha) for intensity calculation. Default `NULL`.

`ef_diesel` Numeric. Emission factor for diesel (kg CO2/L). Default 2.68.

`ef_electricity` Numeric. Emission factor for grid electricity (kg CO2/kWh). Default 0.727, the weighted average for the Indian grid in 2023-24 (Central Electricity Authority, CO2 Baseline Database version 20.0). Versions before 1.1.0 used 0.82; set `ef_electricity = 0.82` to reproduce earlier results. Use your own country or year value where appropriate.

`ef_n_manufacture` Numeric. Emission factor for N fertilizer manufacture (kg CO2-eq/kg N). Default 4.96 (urea-based).

`ef_p_manufacture` Numeric. Emission factor for P fertilizer manufacture (kg CO2-eq/kg P2O5). Default 1.61.

`ef_k_manufacture` Numeric. Emission factor for K fertilizer manufacture (kg CO2-eq/kg K2O). Default 0.57.

`ef_pesticide` Numeric. Emission factor for pesticide (kg CO2-eq/kg a.i.). Default 10.97.

`ef_seed` Numeric. Emission factor for seed production (kg CO2-eq/kg). Default 0.58.

`gwp_n2o` Numeric. Global warming potential of N2O relative to CO2. Default 273 (IPCC AR6, 100-year horizon).

`gwp_ch4` Numeric. Global warming potential of CH4 relative to CO2. Default 27 (IPCC AR6, 100-year horizon).

`ef_n2o` Numeric. Direct N2O emission factor (kg N2O-N per kg N applied). Default 0.01, the IPCC aggregated Tier 1 value. The 2019 IPCC Refinement gives 0.005 for dry climates (relevant to arid zones) and 0.016 for synthetic N in wet climates.

`verbose` Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$CF = \sum_i A_i \times EF_i$$

in kg CO2-eq/ha, where A_i is the activity (input use) and EF_i its emission factor; N2O and CH4 are converted with the GWP values. When `yield` is given, the carbon intensity CF/Y (kg CO2-eq/kg) is also returned.

Value

A list with components:

total_cf Numeric. Total carbon footprint (kg CO2-eq/ha)

cf_intensity Numeric. Carbon footprint intensity (kg CO2-eq/kg yield), returned only when `yield` is provided and positive

breakdown Data frame with columns: source, emission_kg_CO2eq, and share_pct showing the contribution of each emission source

References

Lal, R. (2004). Carbon emission from farm operations. *Environment International*, 30(7), 981-990. doi:10.1016/j.envint.2004.03.005

Central Electricity Authority (2024). *CO2 Baseline Database for the Indian Power Sector*, Version 20.0. Ministry of Power, Government of India, New Delhi.

Forster, P. et al. (2021). The Earth's energy budget, climate feedbacks, and climate sensitivity. In *Climate Change 2021: The Physical Science Basis* (IPCC AR6 WGI Chapter 7). doi:10.1017/9781009157896.009

IPCC (2019). *2019 Refinement to the 2006 IPCC Guidelines for National Greenhouse Gas Inventories*. Volume 4: Agriculture, Forestry and Other Land Use. ISBN:978-4-88788-232-4.

Examples

```
cf <- carbon_footprint(
  diesel_use = 60, electricity_use = 200,
  n_fertilizer = 120, p_fertilizer = 60, k_fertilizer = 40,
  pesticide_use = 1.5, seed_rate = 100, yield = 4500
)
cf$total_cf
cf$cf_intensity
cf$breakdown
```

---

carbon_sequestration_rate

Carbon Sequestration Rate

Description

Annual rate of soil carbon change between two measurement time points.

Usage

```
carbon_sequestration_rate(
  soc_initial,
  soc_final,
  years,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`soc_initial`	Numeric vector. Initial SOC stock (Mg C/ha).
`soc_final`	Numeric vector. Final SOC stock (Mg C/ha).
`years`	Numeric. Number of years between measurements.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$CSR = \frac{SOC_{final} - SOC_{initial}}{Years}$$

Value

Numeric vector. Carbon sequestration rate (Mg C/ha/year). Positive values indicate net sequestration; negative values indicate net emission from soil.

References

Minasny, B. et al. (2017). Soil carbon 4 per mille. *Geoderma*, 292, 59-86. doi:10.1016/j.geoderma.2017.01.002

Examples

```
carbon_sequestration_rate(soc_initial = 28.5, soc_final = 31.2,
                          years = 5)
```

---

carbon_sustainability_index
Carbon Sustainability Index (CSI)

Description

Ratio of yield to net carbon emission, accounting for carbon sequestration by the soil.

Usage

```
carbon_sustainability_index(
  yield,
  carbon_emission,
  carbon_sequestered = 0,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`carbon_emission`	Numeric vector. Total GHG emission (kg CO2-eq/ha).
`carbon_sequestered`	Numeric vector. Carbon sequestered in soil (kg CO2-eq/ha). Default 0.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$CSI = \frac{Y}{E - S}$$

where Y is yield (kg/ha), E is GHG emission and S is carbon sequestered (both kg CO2-eq/ha).

Value

Numeric vector. Carbon sustainability index. Higher values indicate better carbon sustainability. Returns `NA` where net emission is zero or negative (net sequestration exceeds emission).

References

Brentrup, F. et al. (2004). Environmental impact assessment of agricultural production systems using the life cycle assessment methodology. *European Journal of Agronomy*, 20(3), 247-264. doi:10.1016/S11610301(03)000248

Examples

```
carbon_sustainability_index(yield = 4500, carbon_emission = 2500,
                            carbon_sequestered = 500)
```

`crop_water_productivity`

Crop Water Productivity (CWP)

Description

Computes crop water productivity as the ratio of yield to total evapotranspiration, analogous to the "more crop per drop" concept.

Usage

```
crop_water_productivity(
  yield,
  et,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`et`	Numeric vector. Seasonal evapotranspiration (mm).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$CWP = \frac{Y}{ET \times 10}$$

The factor 10 converts mm to m3/ha.

Value

Numeric vector. Crop water productivity (kg/m3).

References

Zwart, S.J. & Bastiaanssen, W.G.M. (2004). Review of measured crop water productivity values for irrigated wheat, rice, cotton and maize. *Agricultural Water Management*, 69(2), 115-133. doi:10.1016/j.agwat.2004.04.007

Examples

```
crop_water_productivity(yield = 4500, et = 400)
```

`crop_water_stress_index` *Crop Water Stress Index (CWSI)*

Description

Computes the Crop Water Stress Index from the ratio of actual to potential evapotranspiration (the evapotranspiration form of the index described by Jackson et al., 1981).

Usage

```
crop_water_stress_index(
  actual_et,
  potential_et,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`actual_et`	Numeric vector. Actual evapotranspiration (mm).
`potential_et`	Numeric vector. Potential or reference evapotranspiration (mm).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$CWSI = 1 - \frac{ET_a}{ET_p}$$

Value

Numeric vector. CWSI values between 0 (no stress) and 1 (maximum stress). Values close to 0 indicate well-watered conditions; values close to 1 indicate severe water deficit.

References

Jackson, R.D., Idso, S.B., Reginato, R.J. & Pinter, P.J. (1981). Canopy temperature as a crop water stress indicator. *Water Resources Research*, 17(4), 1133-1138. doi:10.1029/WR017i004p01133

Idso, S.B. et al. (1981). Normalizing the stress-degree-day parameter for environmental variability. *Agricultural Meteorology*, 24, 45-55. doi:10.1016/00021571(81)900327

Examples

```
crop_water_stress_index(c(4.5, 3.8, 3.0, 2.2), c(5.0, 5.0, 5.0, 5.0))
```

`crop_yield_index` *Crop Yield Index (CYI)*

Description

Relative yield performance compared to a check or control treatment.

Usage

```
crop_yield_index(
  yield,
  check_yield,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Treatment yields (kg/ha).
check_yield	Numeric. Control or check yield (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$CYI = \frac{Y}{Y_{check}}$$

Multiply by 100 to express as a percentage of the check.

Value

Numeric vector. Yield index where 1.0 equals the check yield. Values above 1 indicate superiority over check.

Examples

```
crop_yield_index(yield = c(4500, 4200, 3800), check_yield = 4000)
```

---

depleted_fraction *Depleted Fraction (DF)*

Description

Ratio of water beneficially consumed (crop ET) to total water inflow, used to assess basin-level water allocation.

Usage

```
depleted_fraction(
  crop_et,
  total_inflow,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

crop_et	Numeric vector. Beneficial crop evapotranspiration (mm).
total_inflow	Numeric vector. Total water inflow: irrigation plus effective rainfall (mm).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$DF = \frac{ET_{actual}}{Water_{supplied}}$$

Value

Numeric vector. Depleted fraction (proportion, 0 to 1).

References

Perry, C., Steduto, P., Allen, R.G. & Burt, C.M. (2009). Increasing productivity in irrigated agriculture: Agronomic constraints and hydrological realities. *Agricultural Water Management*, 96(11), 1517-1524. doi:10.1016/j.agwat.2009.05.005

Examples

```
depleted_fraction(crop_et = 320, total_inflow = 500)
```

`energy_budget` *Energy Budget from Physical Input Quantities*

Description

Converts physical quantities of inputs (litres of diesel, kg of fertilizer, labour hours, and so on) and outputs (grain and straw) to energy using the equivalents in `energy_coefficients` or a user-supplied table. Returns input energy by source, direct and indirect energy, renewable and non-renewable energy, and, when outputs are given, the main energy efficiency indicators.

Usage

```
energy_budget(
  data,
  coefficients = wefnexus::energy_coefficients,
  yield_col = "grain",
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`data`	A data frame with one row per treatment or plot, or a named numeric vector for a single treatment. Columns (or names) that match the `input` column of `coefficients` are converted; other columns are kept as identifiers (character or factor) or reported as unmatched (numeric).

coefficients — A data frame with columns input, unit, mj_per_unit, type ("input" or "output"), form ("direct" or "indirect") and source ("renewable" or "non-renewable"). Default energy_coefficients. Edit a copy to use values from your own reference.

yield_col — Character. Name of the output column holding economic yield (kg/ha) for energy productivity and intensity. Default "grain".

verbose — Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Energy of each item is quantity multiplied by its energy equivalent (MJ per unit). Shares are percentages of total input energy. When outputs are present the summary also includes energy output, net energy, energy use efficiency (output/input), energy productivity (kg/MJ) and energy intensity (MJ/kg).

Value

An object of class energy_budget: a list with

summary Data frame, one row per treatment: identifier columns, total_input_MJ, direct_MJ, indirect_MJ, renewable_MJ, non_renewable_MJ, their percentage shares, and (if outputs are present) total_output_MJ, net_energy_MJ, EUE, energy_productivity_kg_MJ and energy_intensity_MJ_kg.

detail Long data frame with the energy of every item per treatment and its share of total input energy.

unmatched Character vector of numeric columns not found in coefficients.

See Also

energy_coefficients, energy_input, energy_use_efficiency

Examples

```
inputs <- data.frame(
  treatment = c("CT", "ZT"),
  human_labour_man = c(320, 240),
  diesel = c(65, 38),
  electricity = c(180, 75),
  nitrogen = c(20, 20),
  phosphorus = c(40, 40),
  seed = c(20, 20),
  herbicide = c(0, 0.75),
  grain = c(980, 1280),
  straw = c(1450, 1850)
)
eb <- energy_budget(inputs)
eb
eb$summary

# Use your own energy equivalent for diesel
```

```
my_coef <- energy_coefficients
my_coef$mj_per_unit[my_coef$input == "diesel"] <- 47.8
energy_budget(inputs, coefficients = my_coef, verbose = FALSE)$summary
```

`energy_coefficients` *Energy Equivalents of Agricultural Inputs and Outputs*

Description

A reference table of typical energy equivalents used in energy budgeting of crop production, for use with `energy_budget`.

Usage

```
energy_coefficients
```

Format

A data frame with 20 rows and 7 variables:

input Character. Item name; use these as column names in the data passed to `energy_budget`.

unit Character. Physical unit of the item.

mj_per_unit Numeric. Energy equivalent (MJ per unit).

type Character. `"input"` or `"output"`.

form Character. `"direct"` or `"indirect"` energy.

source Character. `"renewable"` or `"non-renewable"`.

note Character. Short description.

Details

Energy equivalents differ between sources, regions and years. These are commonly used values in Indian and international crop energy studies and are provided as a starting point. Check them against the reference you cite and edit a copy of the table where needed (see the examples in `energy_budget`).

Source

Values compiled from commonly cited energy budgeting literature, including Mittal, J.P. & Dhawan, K.C. (1988). *Research Manual on Energy Requirements in Agricultural Sector*. ICAR, New Delhi; and Pimentel, D. (Ed.) (1980). *Handbook of Energy Utilization in Agriculture*. CRC Press, Boca Raton.

Examples

```
energy_coefficients
subset(energy_coefficients, type == "input" & form == "direct")
```

`energy_input` *Total Energy Input*

Description

Computes total energy input for a cropping system from individual input components with energy equivalents.

Usage

```
energy_input(
  seed = 0,
  fertilizer_n = 0,
  fertilizer_p = 0,
  fertilizer_k = 0,
  fym = 0,
  pesticide = 0,
  diesel = 0,
  electricity = 0,
  human_labour = 0,
  machinery = 0,
  irrigation = 0,
  micronutrient = 0,
  biofertilizer = 0,
  solar_energy = 0,
  other = 0,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`seed`	Numeric. Energy from seed (MJ/ha). Default 0.
`fertilizer_n`	Numeric. Energy from nitrogen fertilizer (MJ/ha). Default 0.
`fertilizer_p`	Numeric. Energy from phosphorus fertilizer (MJ/ha). Default 0.
`fertilizer_k`	Numeric. Energy from potassium fertilizer (MJ/ha). Default 0.
`fym`	Numeric. Energy from farmyard manure (MJ/ha). Default 0.
`pesticide`	Numeric. Energy from pesticides: herbicides, insecticides, fungicides (MJ/ha). Default 0.
`diesel`	Numeric. Energy from diesel fuel (MJ/ha). Default 0.
`electricity`	Numeric. Energy from electricity for irrigation (MJ/ha). Default 0.
`human_labour`	Numeric. Energy from human labour (MJ/ha). Default 0.
`machinery`	Numeric. Energy from machinery depreciation (MJ/ha). Default 0.
`irrigation`	Numeric. Energy for canal/pumped irrigation (MJ/ha). Default 0.
`micronutrient`	Numeric. Energy from micronutrient fertilizers (MJ/ha). Default 0.

biofertilizer	Numeric. Energy from biofertilizers (MJ/ha). Default 0.
solar_energy	Numeric. Energy from solar pumping systems (MJ/ha). Default 0.
other	Numeric. Any additional energy inputs (MJ/ha). Default 0.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$EI = \sum (Input_i \times EE_i)$$

Value

Numeric. Total energy input (MJ/ha), computed as the sum of all individual energy input components.

References

Mittal, V.K. & Dhawan, K.C. (1988). Research manual on energy requirements in agricultural sector. Punjab Agricultural University, Ludhiana, India.

Chaudhary, V.P. et al. (2009). Energy auditing of diversified rice-wheat cropping systems in the Indo-Gangetic Plains. *Energy*, 34(9), 1091-1096. doi:10.1016/j.energy.2009.04.017

Examples

```
energy_input(seed = 250, fertilizer_n = 3600, fertilizer_p = 500,
             diesel = 2800, human_labour = 180, machinery = 1200,
             irrigation = 1500)
```

---

energy_intensity *Energy Intensity*

Description

Energy consumed per unit of crop yield.

Usage

```
energy_intensity(
  energy_in,
  yield,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

energy_in	Numeric vector. Total energy input (MJ/ha).
yield	Numeric vector. Crop yield (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$EI = \frac{EnergyInput}{Yield}$$

Value

Numeric vector. Energy intensity (MJ/kg). Lower values indicate more energy-efficient production.

Examples

```
energy_intensity(energy_in = 12000, yield = 4500)
```

---

energy_output *Total Energy Output*

Description

Computes total energy output from grain and straw/stover yields using crop-specific energy coefficients.

Usage

```
energy_output(
  grain_yield,
  straw_yield = 0,
  grain_energy_coeff = 14.7,
  straw_energy_coeff = 12.5,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

grain_yield	Numeric vector. Grain or economic yield (kg/ha).
straw_yield	Numeric vector. Straw, stover, or by-product yield (kg/ha). Default 0.
grain_energy_coeff	Numeric. Energy coefficient for grain (MJ/kg). Default 14.7 (wheat grain equivalent).

straw_energy_coeff
: Numeric. Energy coefficient for straw (MJ/kg). Default 12.5 (wheat straw equivalent).

verbose
: Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$E_{out} = (Y_g \times C_g) + (Y_s \times C_s)$$

where Y_g and Y_s are grain and straw yields, and C_g and C_s are their respective energy coefficients.

Value

Numeric vector. Total energy output (MJ/ha).

References

Devasenapathy, P., Senthilkumar, G. & Shanmugam, P.M. (2009). Energy management in crop production. Indian Journal of Agronomy, 54(1), 80-90.

Examples

```
energy_output(grain_yield = 4500, straw_yield = 5500)
energy_output(grain_yield = 1500, straw_yield = 2000,
              grain_energy_coeff = 14.3, straw_energy_coeff = 12.5)
```

---

energy_productivity *Energy Productivity*

Description

Crop yield produced per unit of energy input.

Usage

```
energy_productivity(
  yield,
  energy_in,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
energy_in	Numeric vector. Total energy input (MJ/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$EP = \frac{Yield}{EnergyInput}$$

Value

Numeric vector. Energy productivity (kg/MJ). Higher values indicate better yield per unit energy invested.

Examples

```
energy_productivity(yield = 4500, energy_in = 12000)
```

---

energy_profitability *Energy Profitability*

Description

Economic return per unit of energy invested.

Usage

```
energy_profitability(
  gross_return,
  energy_in,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

gross_return	Numeric vector. Gross economic return (currency/ha).
energy_in	Numeric vector. Total energy input (MJ/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$EPr = \frac{GR}{E_{in}}$$

where GR is gross return (currency/ha) and E_{in} is energy input (MJ/ha). The energy-based ratio (output - input) / input is `net_energy(out, in) / in`, or EUE minus 1.

Value

Numeric vector. Energy profitability (currency/MJ).

Examples

```
energy_profitability(gross_return = 112500, energy_in = 12000)
```

`energy_use_efficiency` *Energy Use Efficiency (EUE)*

Description

Ratio of total energy output to total energy input.

Usage

```
energy_use_efficiency(
  energy_out,
  energy_in,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`energy_out`	Numeric vector. Total energy output (MJ/ha).
`energy_in`	Numeric vector. Total energy input (MJ/ha).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$EUE = \frac{E_{out}}{E_{in}}$$

Value

Numeric vector. Energy use efficiency (dimensionless ratio). Values greater than 1 indicate a positive energy balance.

Examples

```
energy_use_efficiency(energy_out = 135000, energy_in = 12000)
```

`eroi` *Energy Return on Investment (EROI)*

Description

Computes the Energy Return on Investment, a key metric for assessing whether an agricultural system produces more energy than it consumes. EROI is mathematically identical to energy use efficiency (EUE) but is the preferred term in energy and sustainability science literature.

Usage

```
eroi(
  energy_out,
  energy_in,
  include_solar = FALSE,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`energy_out`	Numeric vector. Total energy output from the agricultural system (MJ/ha), including grain, straw, and any co-products.
`energy_in`	Numeric vector. Total energy input invested in the agricultural system (MJ/ha), including all direct (diesel, electricity) and indirect (fertilizer manufacture, machinery depreciation) energy inputs.
`include_solar`	Logical. If `TRUE`, includes captured solar energy in the output. Default `FALSE` (standard practice).
`verbose`	Logical. If `TRUE`, prints informational messages and interpretation. Default `TRUE`.

Details

$$EROI = \frac{E_{out}}{E_{in}}$$

EROI is the fundamental metric of net energy analysis. In agricultural contexts, EROI typically ranges from 2 to 15 for conventional cropping systems, with higher values for low-input or conservation agriculture systems.

Value

Numeric vector. EROI values (dimensionless ratio). Interpreted as follows:

EROI > 5 Highly energy-profitable system

1 < EROI < 5 Energy-positive but moderate return

EROI = 1 Break-even: energy output equals input

EROI < 1 Energy sink: system consumes more than it produces

References

Hall, C.A.S., Lambert, J.G. & Balogh, S.B. (2014). EROI of different fuels and the implications for society. *Energy Policy*, 64, 141-152. doi:10.1016/j.enpol.2013.05.049

Murphy, D.J. & Hall, C.A.S. (2010). Year in review - EROI or energy return on (energy) invested. *Annals of the New York Academy of Sciences*, 1185(1), 102-118. doi:10.1111/j.1749-6632.2009.05282.x

Murphy, D.J. et al. (2022). Energy return on investment of major energy carriers: Review and harmonization. *Sustainability*, 14(12), 7098. doi:10.3390/su14127098

Examples

```
# Conservation agriculture with low input
eroi(energy_out = 59800, energy_in = 8500)

# Conventional tillage with high input
eroi(energy_out = 40800, energy_in = 12500)

# Multiple treatments
eroi(energy_out = c(40800, 50500, 59800),
     energy_in = c(12500, 9800, 8500))
```

---

food_productivity_index

Food Productivity Index (FPI)

Description

Computes a composite food productivity index considering yield relative to potential, and optionally nutritional quality.

Usage

```
food_productivity_index(
  yield,
  reference_yield,
  protein_content = NULL,
```

```
  caloric_value = NULL,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
reference_yield	Numeric. Reference or potential yield (kg/ha).
protein_content	Numeric vector. Protein content (percent). Default NULL.
caloric_value	Numeric vector. Caloric value (kcal/100g). Default NULL.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Without protein and caloric values,

$$FPI = \min(Y/Y_{ref}, 1)$$

With both supplied,

$$FPI = 0.5 \min(Y/Y_{ref}, 1) + 0.25 \min(P/40, 1) + 0.25 \min(C/400, 1)$$

where P is protein (percent) and C is caloric value (kcal/100 g); 40 percent protein and 400 kcal/100 g are the upper reference values. The index is on a 0 to 1 scale.

Value

Numeric vector. Food productivity index on a 0 to 1 scale. When protein and caloric values are provided, weights are 0.50 for yield ratio, 0.25 for protein, and 0.25 for caloric content.

Examples

```
food_productivity_index(yield = 4500, reference_yield = 6000,
                        protein_content = 12.5, caloric_value = 340)
```

---

ghg_emission *GHG Emission Estimation from Field Operations*

Description

Quick estimation of greenhouse gas emissions (N2O, CH4, CO2) from agricultural field operations using IPCC Tier 1 default factors.

Usage

```
ghg_emission(
  n_applied,
  residue_burned = FALSE,
  residue_amount = 0,
  paddy_days = 0,
  tillage = "conventional",
  gwp_n2o = 273,
  gwp_ch4 = 27,
  ef_n2o = 0.01,
  ef_ch4_paddy = 1.3,
  combustion_factor = 0.8,
  ef_ch4_burn = 2.7,
  ef_n2o_burn = 0.07,
  include_burn_co2 = FALSE,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

n_applied	Numeric. N fertilizer applied (kg/ha).
residue_burned	Logical. Whether crop residues are burned. Default FALSE.
residue_amount	Numeric. Residue amount (kg dry matter/ha) if burned. Default 0.
paddy_days	Numeric. Days under flooded paddy conditions. Default 0.
tillage	Character. Tillage type: "conventional", "reduced", or "zero". Default "conventional".
gwp_n2o	Numeric. GWP for N2O. Default 273 (IPCC AR6).
gwp_ch4	Numeric. GWP for CH4. Default 27 (IPCC AR6).
ef_n2o	Numeric. Direct N2O emission factor (kg N2O-N per kg N applied). Default 0.01, the IPCC aggregated Tier 1 value. The 2019 IPCC Refinement gives 0.005 for dry climates (relevant to arid zones) and 0.016 for synthetic N in wet climates.
ef_ch4_paddy	Numeric. Baseline CH4 emission factor for continuously flooded rice (kg CH4/ha/day). Default 1.3 (IPCC 2006); the 2019 Refinement gives 1.19.
combustion_factor	Numeric. Fraction of residue actually burnt. Default 0.8 (IPCC 2006 Table 2.6 for maize and rice residues; 0.9 for wheat).
ef_ch4_burn, ef_n2o_burn	Numeric. Emission factors for burning agricultural residues (g per kg dry matter burnt). Defaults 2.7 and 0.07 (IPCC 2006 Table 2.5).
include_burn_co2	Logical. Add biogenic CO2 from residue burning to the total. Default FALSE.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$GHG = \sum (Activity_i \times EF_i)$$

where EF is the emission factor for each activity.

Residue burning follows IPCC (2006, Vol. 4, Ch. 2): mass burnt = residue x combustion factor; CH4 and N2O = mass burnt x emission factor (g/kg dry matter). CO2 from burning crop residue is biogenic and, as in IPCC inventories, is reported but left out of the total unless `include_burn_co2 = TRUE`. Versions before 1.1.0 added that CO2 to the total and omitted burning CH4 and N2O.

The tillage CO2 values (150, 100 and 50 kg CO2/ha for conventional, reduced and zero tillage) are rough screening values. When diesel use is known, `carbon_footprint` gives a better estimate.

Value

For a single observation, a data frame with columns `source`, `gas`, `emission_kg`, `CO2_eq_kg` and `counted` (whether the row enters the total); the total is stored in `attr(, "total_CO2eq")`. For several observations (any input of length > 1), a data frame with one row per observation and a `total_CO2eq` column.

References

IPCC (2006). *2006 IPCC Guidelines for National Greenhouse Gas Inventories*. Volume 4, Chapter 2: Generic methodologies applicable to multiple land-use categories (Tables 2.5 and 2.6). IGES, Japan.

IPCC (2019). *2019 Refinement to the 2006 IPCC Guidelines for National Greenhouse Gas Inventories*. Volume 4, Chapter 11: N2O emissions from managed soils, and CO2 emissions from lime and urea application. ISBN:978-4-88788-232-4.

Examples

```
ghg_emission(n_applied = 120, tillage = "zero")
ghg_emission(n_applied = 150, paddy_days = 90)
ghg_emission(n_applied = 100, residue_burned = TRUE,
             residue_amount = 5000)
ghg_emission(n_applied = c(20, 40, 60), ef_n2o = 0.005)
```

---

global_warming_potential

Global Warming Potential (GWP)

Description

Converts individual greenhouse gas emissions to CO2 equivalents using IPCC AR6 Global Warming Potential values.

Usage

```
global_warming_potential(
  co2 = 0,
  ch4 = 0,
  n2o = 0,
  gwp_ch4 = 27,
  gwp_n2o = 273,
  time_horizon = "100yr",
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

co2	Numeric. CO2 emission (kg/ha). Default 0.
ch4	Numeric. CH4 emission (kg/ha). Default 0.
n2o	Numeric. N2O emission (kg/ha). Default 0.
gwp_ch4	Numeric. GWP of CH4. Default 27 (IPCC AR6, 100-year).
gwp_n2o	Numeric. GWP of N2O. Default 273 (IPCC AR6, 100-year).
time_horizon	Character. "100yr" (default) or "20yr" for different assessment periods.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$GWP = CO_2 + CH_4 \times 27 + N_2O \times 273$$

IPCC AR6 100-year GWP values (with climate-carbon feedbacks): CH4 = 27, N2O = 273.

IPCC AR6 20-year GWP values for non-fossil (biogenic) methane, as emitted by agriculture: CH4 = 79.7, N2O = 273. With time_horizon = "20yr" these replace the defaults, unless you supply gwp_ch4 or gwp_n2o yourself. Versions before 1.1.0 used 81 for CH4.

Note: Previous IPCC AR5 100-year values were CH4 = 34, N2O = 298 (with climate-carbon feedbacks) or CH4 = 28, N2O = 265 (without).

Value

Numeric. Total global warming potential (kg CO2-eq/ha).

References

Forster, P. et al. (2021). The Earth's energy budget, climate feedbacks, and climate sensitivity. In *Climate Change 2021: The Physical Science Basis* (IPCC AR6 WGI Chapter 7), Table 7.15. doi:10.1017/9781009157896.009

Examples

```
global_warming_potential(co2 = 500, ch4 = 50, n2o = 2)
global_warming_potential(co2 = 500, ch4 = 50, n2o = 2,
                         time_horizon = "20yr")
```

`harvest_index` *Harvest Index (HI)*

Description

Proportion of economic yield to total above-ground biological yield.

Usage

```
harvest_index(
  economic_yield,
  biological_yield,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`economic_yield`	Numeric vector. Grain or economic yield (kg/ha).
`biological_yield`	Numeric vector. Total above-ground biomass yield (kg/ha), which is grain plus straw/stover.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$HI = \frac{Y_{econ}}{Y_{biol}}$$

Value

Numeric vector. Harvest index as a proportion (0 to 1). A warning is issued if any value exceeds 1.

References

Hay, R.K.M. (1995). Harvest index: A review of its use in plant breeding and crop physiology. *Annals of Applied Biology*, 126(1), 197-216. doi:10.1111/j.17447348.1995.tb05015.x

Examples

```
harvest_index(economic_yield = 4500, biological_yield = 10000)
```

human_energy_profitability *Human Energy Profitability*

Description

Ratio of energy output to human labour energy input, measuring the degree of human labour amplification by the farming system.

Usage

```
human_energy_profitability(
  energy_out,
  human_energy,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

energy_out	Numeric vector. Total energy output (MJ/ha).
human_energy	Numeric vector. Human labour energy input (MJ/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$HEP = \frac{EnergyOutput}{HumanEnergyInput}$$

Value

Numeric vector. Human energy profitability (dimensionless ratio).

Examples

```
human_energy_profitability(energy_out = 135000, human_energy = 180)
```

`internal_utilization_efficiency` *Internal Utilization Efficiency (IUE)*

Description

Yield produced per unit of total nutrient in the plant.

Usage

```
internal_utilization_efficiency(
  yield,
  total_uptake,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`total_uptake`	Numeric vector. Total plant nutrient uptake (kg/ha).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$IE = \frac{Yield}{TotalNutrientUptake}$$

Value

Numeric vector. Internal utilization efficiency (kg yield per kg nutrient uptake).

Examples

```
internal_utilization_efficiency(yield = 4500, total_uptake = 100)
```

`irrigation_efficiency` *Irrigation Efficiency*

Description

Computes multiple irrigation efficiency metrics: conveyance, application, overall, and consumptive use efficiency.

Usage

```
irrigation_efficiency(
  water_delivered,
  water_diverted,
  water_stored = NULL,
  crop_et = NULL,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`water_delivered`	Numeric vector. Water delivered to field (mm or m3).
`water_diverted`	Numeric vector. Water diverted from source (mm or m3).
`water_stored`	Numeric vector. Water stored in root zone (mm or m3). If `NULL`, application and overall efficiency are not computed.
`crop_et`	Numeric vector. Crop evapotranspiration (mm). If `NULL`, consumptive use efficiency is not computed.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$IE = \frac{Water_{crop}}{Water_{applied}} \times 100$$

Value

A data frame with efficiency metrics expressed as percentages. Columns depend on inputs provided:

conveyance_eff Conveyance efficiency (percent), always computed

application_eff Application efficiency (percent), if `water_stored` is provided

overall_eff Overall efficiency (percent), if `water_stored` is provided

consumptive_eff Consumptive use efficiency (percent), if `crop_et` is provided

References

Grafton, R.Q. et al. (2018). The paradox of irrigation efficiency. *Science*, 361(6404), 748-750. doi:10.1126/science.aat9314

Examples

```
irrigation_efficiency(water_delivered = 400, water_diverted = 500,
                      water_stored = 350, crop_et = 320)
```

---

land_equivalent_ratio *Land Equivalent Ratio (LER)*

Description

Evaluates the productivity advantage of intercropping systems over sole cropping.

Usage

```
land_equivalent_ratio(
  yield_inter_a,
  yield_sole_a,
  yield_inter_b,
  yield_sole_b,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield_inter_a	Numeric vector. Yield of crop A in intercrop (kg/ha).
yield_sole_a	Numeric vector. Yield of crop A in sole crop (kg/ha).
yield_inter_b	Numeric vector. Yield of crop B in intercrop (kg/ha).
yield_sole_b	Numeric vector. Yield of crop B in sole crop (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$LER = \frac{Y_{iA}}{Y_{sA}} + \frac{Y_{iB}}{Y_{sB}}$$

Value

Numeric vector. LER values. Values greater than 1 indicate an intercropping advantage; less than 1 indicates a disadvantage.

References

Mead, R. & Willey, R.W. (1980). The concept of a land equivalent ratio and advantages in yields from intercropping. *Experimental Agriculture*, 16(3), 217-228. doi:10.1017/S0014479700010978

Examples

```
land_equivalent_ratio(yield_inter_a = 3500, yield_sole_a = 4500,
                      yield_inter_b = 800, yield_sole_b = 1200)
```

`net_energy` *Net Energy*

Description

Difference between total energy output and total energy input.

Usage

```
net_energy(
  energy_out,
  energy_in,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`energy_out`	Numeric vector. Total energy output (MJ/ha).
`energy_in`	Numeric vector. Total energy input (MJ/ha).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$NE = EnergyOutput - EnergyInput$$

Value

Numeric vector. Net energy (MJ/ha). Positive values indicate net energy gain; negative values indicate net energy loss.

Examples

```
net_energy(energy_out = 135000, energy_in = 12000)
```

nexus_analysis *Complete Nexus Analysis from Replicated Field Data*

Description

Runs the full WEFNC workflow on plot-level data in one call: computes plot-level indicators, compares treatments by ANOVA (with SEm, CD and CV), builds the composite nexus index from treatment means with equal or objective weights, and ranks treatments by TOPSIS.

Usage

```
nexus_analysis(
  data,
  treatment,
  yield,
  water,
  energy_in,
  energy_out,
  emission,
  n_applied = NULL,
  n_uptake = NULL,
  block = NULL,
  weights = "equal",
  nutrient_metric = NULL,
  carbon_basis = c("intensity", "emission"),
  alpha = 0.05,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

data	A data frame with one row per plot (treatment x replication) or per treatment.
treatment	Character. Name of the treatment column.
yield	Character. Column with economic yield (kg/ha).
water	Character. Column with water consumed (mm).
energy_in	Character. Column with energy input (MJ/ha).
energy_out	Character. Column with energy output (MJ/ha).
emission	Character. Column with GHG emission (kg CO2-eq/ha).
n_applied	Character. Column with N applied (kg/ha), or NULL.
n_uptake	Character. Column with total N uptake (kg/ha), or NULL. At least one of n_applied and n_uptake is required.
block	Character. Column with block (replication) for an RCBD, or NULL for a CRD.
weights	Numeric vector of five weights (Water, Energy, Food, Nutrient, Carbon) or one of "equal" (default), "entropy", "critic", "pca".

`nutrient_metric`
: Character. `"pfp"` or `"iue"`. The default picks `"iue"` when `n_uptake` is given and `"pfp"` otherwise.

`carbon_basis`
: Character. `"intensity"` (default, kg CO2-eq per kg yield) or `"emission"` (kg CO2-eq/ha).

`alpha`
: Numeric. Significance level for ANOVA. Default 0.05.

`verbose`
: Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Plot-level indicators are WUE (kg/ha/mm), EUE, the nutrient indicator (PFP or IUE, kg/kg) and carbon intensity (kg CO2-eq/kg). The nexus index and TOPSIS ranking use treatment means of the raw variables. ANOVA is skipped when treatments are not replicated.

Value

An object of class `nexus_analysis`: a list with

plot_data Input data with plot-level indicator columns `WUE`, `EUE`, `NUE` and `CI` added.

treatment_means Treatment means of the raw variables.

nexus A `nexus_result` from `nexus_summary` on the treatment means.

anova A `nexus_anova` object for yield and the four indicators, or `NULL` without replication.

topsis TOPSIS ranking from `nexus_topsis`.

weights Named weights used for the nexus index.

See Also

`nexus_summary`, `nexus_anova`, `nexus_weights`, `nexus_topsis`

Examples

```
data(arid_pulse_trial)
res <- nexus_analysis(
  arid_pulse_trial, treatment = "treatment", block = "replication",
  yield = "grain_yield", water = "total_water",
  energy_in = "energy_input", energy_out = "energy_output",
  emission = "ghg_emission", n_uptake = "n_uptake",
  weights = "entropy"
)
res
res$anova

plot(res$nexus)
```

nexus_anova *ANOVA Table for Nexus Indicators (CRD or RCBD)*

Description

Runs a one-way analysis of variance for each indicator, in a completely randomized design (CRD) or randomized complete block design (RCBD), and returns the statistics usually reported in agronomy papers: treatment means, standard error of mean (SEm), critical difference (CD, the least significant difference), coefficient of variation (CV) and letter groups.

Usage

```
nexus_anova(
  data,
  response,
  treatment,
  block = NULL,
  alpha = 0.05,
  digits = 2,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

data	A data frame with replication-level (plot-level) data.
response	Character vector of numeric column names to analyse.
treatment	Character. Name of the treatment column.
block	Character. Name of the block (replication) column for an RCBD. NULL (default) fits a CRD.
alpha	Numeric. Significance level for the F test, CD and letters. Default 0.05.
digits	Integer. Decimal places in the formatted table. Default 2.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

For each response the model y ~ block + treatment (RCBD) or y ~ treatment (CRD) is fitted with aov; in an RCBD the treatment F test is adjusted for blocks. With error mean square MSE, r replications and error degrees of freedom df_e:

$$SEm = \sqrt{MSE/r}, \quad SEd = \sqrt{2MSE/r}$$

$$CD = t_{1-\alpha/2,\, df_e} \times SEd, \quad CV = 100\sqrt{MSE}/\bar{y}$$

Treatment means are least-squares means: in an RCBD with missing plots they are adjusted for blocks, and with complete data they equal the raw means. Letters follow Fisher's LSD: every pair

of treatments is compared with its own standard error of the difference, and the compact letter display is built with the insert-and-absorb algorithm of Piepho (2004), so treatments sharing a letter do not differ at level `alpha` and treatments that differ share no letter. With equal replication this is the familiar single CD. With unequal replication or missing plots the single SEm, SEd and CD in the table use the harmonic mean of replications and are approximate, while the letters remain exact. Beyond 26 groups the letters continue with capitals and then numbered letters.

When the treatment F test is not significant, all treatments get the letter "a" and the CD is shown as "NS" in the formatted table. Treatments with no observations for a response are shown as missing.

Value

An object of class `nexus_anova`: a list with

means Data frame of treatment means (treatments in rows, responses in columns).

letters Data frame of letter groups in the same layout.

se Data frame of the standard error of each treatment mean (standard deviation / sqrt(n)), in the same layout.

stats Data frame, one row per response: `F_value`, `p_value`, `significant`, `MSE`, `df_error`, `SEm`, `SEd`, `CD`, `CV_pct`, `grand_mean`, `balanced`.

table Character data frame ready for a manuscript: means with letters, followed by SEm, CD and CV rows.

models List of fitted `aov` objects.

alpha The significance level used.

Use `plot()` on the result for a bar chart with error bars and letters (`plot.nexus_anova`).

References

Gomez, K.A. & Gomez, A.A. (1984). *Statistical Procedures for Agricultural Research*, 2nd ed. John Wiley & Sons, New York.

Piepho, H.-P. (2004). An algorithm for a letter-based representation of all-pairwise comparisons. *Journal of Computational and Graphical Statistics*, 13(2), 456-466. doi:10.1198/1061860043515

Examples

```
data(arid_pulse_trial)
res <- nexus_anova(arid_pulse_trial,
                   response = c("grain_yield", "n_uptake"),
                   treatment = "treatment", block = "replication")
res
res$stats
```

nexus_heatmap *Nexus Heatmap*

Description

Creates a heatmap showing nexus dimension scores across treatments.

Usage

```
nexus_heatmap(
  scores,
  row_labels = NULL,
  col_labels = NULL,
  title = "WEFNC Nexus Heatmap",
  color_palette = NULL
)
```

Arguments

scores	A matrix or data frame where rows represent treatments and columns represent nexus dimensions.
row_labels	Character vector. Treatment names.
col_labels	Character vector. Dimension names.
title	Character. Plot title. Default "WEFNC Nexus Heatmap".
color_palette	Character vector of colors for gradient. Default is a single-hue blue scale that stays readable with colour vision deficiency (version 1.0.0 used red-yellow-green).

Value

Invisibly returns the input scores matrix. Called primarily for the side effect of generating a heatmap plot.

Examples

```
scores <- matrix(c(0.85, 0.70, 0.90, 0.65, 0.80,
                   0.72, 0.80, 0.85, 0.70, 0.60),
                 nrow = 2, byrow = TRUE)
nexus_heatmap(scores,
  row_labels = c("Conservation", "Conventional"),
  col_labels = c("Water", "Energy", "Food", "Nutrient", "Carbon"))
```

nexus_heatmap_gg *Nexus Heatmap (ggplot2)*

Description

Publication-quality heatmap using ggplot2 for WEFNC nexus scores.

Usage

```
nexus_heatmap_gg(
  scores,
  treatment_names = NULL,
  dim_labels = c("Water", "Energy", "Food", "Nutrient", "Carbon"),
  title = "WEFNC Nexus Heatmap",
  colours = NULL
)
```

Arguments

scores	A matrix or data frame (treatments as rows, dimensions as columns).
treatment_names	Character vector. Row labels.
dim_labels	Character vector. Column labels. Default: Water, Energy, Food, Nutrient, Carbon.
title	Character. Plot title.
colours	Character vector of colours for the 0 to 1 gradient. Default is a single-hue blue scale that is readable with colour vision deficiency. The red-yellow-green scale of version 1.0.0 is `c("#D73027", "#FDAE61", "#FEE08B", "#A6D96A", "#1A9850")`.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  scores <- matrix(c(0.85, 0.70, 0.90, 0.65, 0.80,
                     0.72, 0.80, 0.85, 0.70, 0.60),
                   nrow = 2, byrow = TRUE)
  nexus_heatmap_gg(scores, treatment_names = c("CA", "CT"))
}
```

`nexus_index` *WEFNC Nexus Index*

Description

Computes a weighted composite Water-Energy-Food-Nutrient-Carbon nexus sustainability index from normalized dimension scores.

Usage

```
nexus_index(
  water_score,
  energy_score,
  food_score,
  nutrient_score,
  carbon_score,
  weights = rep(0.2, 5),
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`water_score`	Numeric vector. Water performance score (0 to 1).
`energy_score`	Numeric vector. Energy performance score (0 to 1).
`food_score`	Numeric vector. Food productivity score (0 to 1).
`nutrient_score`	Numeric vector. Nutrient use efficiency score (0 to 1).
`carbon_score`	Numeric vector. Carbon performance score (0 to 1), where higher values indicate lower carbon footprint.
`weights`	Numeric vector of length 5. Weights for Water, Energy, Food, Nutrient, and Carbon dimensions. Must sum to 1. Default is equal weighting (0.2 each).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$NI = w_W S_W + w_E S_E + w_F S_F + w_N S_N + w_C S_C$$

Value

Numeric vector. Composite nexus index on a 0 to 1 scale, where higher values indicate better overall sustainability.

Examples

```
nexus_index(water_score = c(0.85, 0.72), energy_score = c(0.70, 0.80),
            food_score = c(0.90, 0.85), nutrient_score = c(0.65, 0.70),
            carbon_score = c(0.80, 0.60))
```

`nexus_radar` *Nexus Radar Plot*

Description

Creates a radar (spider/web) chart showing the WEFNC nexus profile of one or more treatments.

Usage

```
nexus_radar(
  scores,
  labels = c("Water", "Energy", "Food", "Nutrient", "Carbon"),
  treatment_names = NULL,
  colors = NULL,
  title = "WEFNC Nexus Profile",
  fill = TRUE,
  alpha = 0.2
)
```

Arguments

`scores`	A matrix or data frame where rows represent treatments and columns represent nexus dimensions. Values should be on a 0 to 1 scale.
`labels`	Character vector. Names for nexus dimensions. Default: Water, Energy, Food, Nutrient, Carbon.
`treatment_names`	Character vector. Labels for each treatment.
`colors`	Character vector. Colors for each treatment. Default is a colour-blind-checked palette (up to 8 treatments).
`title`	Character. Plot title. Default "WEFNC Nexus Profile".
`fill`	Logical. If `TRUE`, fills the radar polygon area. Default `TRUE`.
`alpha`	Numeric. Fill transparency (0 to 1). Default 0.2.

Value

Invisibly returns the input `scores` matrix. Called primarily for the side effect of generating a radar plot.

Examples

```
scores <- matrix(c(0.85, 0.70, 0.90, 0.65, 0.80,
                   0.72, 0.80, 0.85, 0.70, 0.60),
                 nrow = 2, byrow = TRUE)
nexus_radar(scores,
  treatment_names = c("CA+SSDI", "CT+Flood"))
```

`nexus_radar_gg` *Nexus Radar Plot (ggplot2)*

Description

Publication-quality radar chart using ggplot2 for WEFNC nexus profiles.

Usage

```
nexus_radar_gg(
  scores,
  treatment_names = NULL,
  dim_labels = c("Water", "Energy", "Food", "Nutrient", "Carbon"),
  title = "WEFNC Nexus Profile",
  facet = FALSE,
  reference = TRUE,
  colours = NULL,
  ncol = NULL
)
```

Arguments

`scores`	A matrix or data frame (treatments as rows, dimensions as columns). Values should be on a 0 to 1 scale.
`treatment_names`	Character vector. Labels for treatments.
`dim_labels`	Character vector. Labels for nexus dimensions. Default: Water, Energy, Food, Nutrient, Carbon.
`title`	Character. Plot title.
`facet`	Logical. If `TRUE`, draws one panel per treatment (small multiples), which stays readable with many treatments. Default `FALSE` (all treatments overlaid).
`reference`	Logical. With `facet = TRUE`, draws the mean profile of all treatments as a grey outline in every panel for comparison. Default `TRUE`.
`colours`	Character vector of colours, one per treatment. Default is a colour-blind-checked palette (up to 8 treatments).
`ncol`	Integer. Number of panel columns when `facet = TRUE`.

Details

Overlaid radars become hard to read beyond three or four treatments; use `facet = TRUE` for more.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  scores <- matrix(c(0.85, 0.70, 0.90, 0.65, 0.80,
                     0.72, 0.80, 0.85, 0.70, 0.60),
                   nrow = 2, byrow = TRUE)
  nexus_radar_gg(scores, treatment_names = c("CA", "CT"))
  nexus_radar_gg(scores, treatment_names = c("CA", "CT"), facet = TRUE)
}
```

---

nexus_rank_compare *Compare Treatment Rankings Across Weighting Methods*

Description

Ranks treatments eight ways (composite index and TOPSIS, each with equal, entropy, CRITIC and PCA weights) and measures how far the rankings agree. A ranking that holds across methods is a stronger basis for a recommendation than one that depends on a single choice of weights.

Usage

```
nexus_rank_compare(
  data,
  benefit = NULL,
  treatment_names = NULL,
  methods = c("equal", "entropy", "critic", "pca"),
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`data`	A data frame or matrix with treatments in rows and raw indicators in columns. Only numeric columns are used.
`benefit`	Logical vector, one value per numeric column: `TRUE` if higher is better, `FALSE` if lower is better. Default all `TRUE`.
`treatment_names`	Character vector of treatment labels.
`methods`	Character vector of weighting methods to compare. Default `c("equal", "entropy", "critic", "pca")`.

verbose Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

For each weighting method, the composite index is the weighted sum of min-max normalized indicators (inverted where benefit = FALSE), and TOPSIS is computed with nexus_topsis. Agreement among the m rankings of n treatments is summarised by Kendall's coefficient of concordance with correction for ties,

$$W = \frac{12S}{m^2(n^3 - n) - m\sum T}$$

where S is the sum of squared deviations of the treatment rank sums from their mean and $T = \sum(t^3 - t)$ over tied groups. W runs from 0 (no agreement) to 1 (identical rankings). The chi-square statistic $\chi^2 = m(n-1)W$ with $n-1$ degrees of freedom is reported for reference only: the eight rankings come from the same data and are not independent judges, so its p-value is not a formal test of robustness. Read W itself, and look at which treatments change rank and under which method.

Value

An object of class rank_compare: a list with

ranks Data frame: treatment, one rank column per method, mean_rank and consensus (rank of the mean rank).

scores Data frame of index values and TOPSIS closeness.

weights Matrix of weights (methods x indicators).

kendall_w Kendall's W.

chisq, df, p_value Chi-square test of W.

spearman Matrix of Spearman correlations between rankings.

References

Kendall, M.G. & Babington Smith, B. (1939). The problem of m rankings. *The Annals of Mathematical Statistics*, 10(3), 275-287. doi:10.1214/aoms/1177732186

See Also

nexus_weights, nexus_topsis, nexus_sensitivity

Examples

```
data(arid_pulse_nexus)
d <- arid_pulse_nexus
ind <- data.frame(
  WUE = d$grain_yield / d$total_water,
  EUE = (d$energy_output_grain + d$energy_output_straw) / d$energy_input,
  Yield = d$grain_yield,
  IUE = d$grain_yield / d$n_uptake,
  CF = d$ghg_emission
```

```
)
rc <- nexus_rank_compare(ind, benefit = c(TRUE, TRUE, TRUE, TRUE, FALSE),
                         treatment_names = d$treatment)
rc
if (requireNamespace("ggplot2", quietly = TRUE)) plot(rc)
```

`nexus_sensitivity` *Nexus Weight Sensitivity Analysis*

Description

Evaluates how the composite nexus index changes as the weight of one dimension varies from 0 to its maximum feasible value, with remaining weight distributed equally among the other four dimensions.

Usage

```
nexus_sensitivity(
  water_score,
  energy_score,
  food_score,
  nutrient_score,
  carbon_score,
  treatment_names = NULL,
  steps = 20,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`water_score`	Numeric vector. Water scores (0 to 1).
`energy_score`	Numeric vector. Energy scores (0 to 1).
`food_score`	Numeric vector. Food scores (0 to 1).
`nutrient_score`	Numeric vector. Nutrient scores (0 to 1).
`carbon_score`	Numeric vector. Carbon scores (0 to 1).
`treatment_names`	Character vector. Treatment labels.
`steps`	Integer. Number of weight steps to evaluate. Default 20.
`verbose`	Logical. If `TRUE`, prints a message. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Value

A data frame with columns: dimension, weight, treatment, and nexus_index showing how the index changes as each dimension's weight varies.

Examples

```
sa <- nexus_sensitivity(
  water_score = c(0.9, 0.5), energy_score = c(0.6, 0.8),
  food_score = c(0.8, 0.7), nutrient_score = c(0.7, 0.6),
  carbon_score = c(0.5, 0.9),
  treatment_names = c("CA", "CT"), steps = 10
)
head(sa)
```

`nexus_summary` *Nexus Summary*

Description

Generates a comprehensive one-call nexus analysis from raw agronomic field data, computing all dimension metrics, normalizing, and producing the composite index.

Usage

```
nexus_summary(
  yield,
  water_consumed,
  energy_input,
  energy_output,
  n_applied = NULL,
  n_uptake = NULL,
  carbon_emission,
  treatment_names = NULL,
  weights = rep(0.2, 5),
  nutrient_metric = c("pfp", "iue"),
  carbon_basis = c("emission", "intensity"),
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`water_consumed`	Numeric vector. Total water consumed (mm).
`energy_input`	Numeric vector. Total energy input (MJ/ha).
`energy_output`	Numeric vector. Total energy output (MJ/ha).
`n_applied`	Numeric vector. Nitrogen applied (kg/ha). Needed when `nutrient_metric = "pfp"`. May be `NULL` when `nutrient_metric = "iue"`.
`n_uptake`	Numeric vector. Total plant nitrogen uptake (kg/ha). Needed when `nutrient_metric = "iue"`; when supplied, internal utilization efficiency is also reported.

`carbon_emission`
: Numeric vector. Total GHG emission (kg CO2-eq/ha).

`treatment_names`
: Character vector. Treatment labels. Default `NULL`.

`weights`
: Numeric vector of length 5 (Water, Energy, Food, Nutrient, Carbon) summing to 1, or one of `"equal"`, `"entropy"`, `"critic"` or `"pca"` to derive weights with `nexus_weights`. Default equal weights.

`nutrient_metric`
: Character. Indicator for the nutrient dimension: `"pfp"` (partial factor productivity of applied N, default) or `"iue"` (internal utilization efficiency, yield per kg N uptake). Use `"iue"` for legumes grown without fertilizer N, where PFP cannot be computed.

`carbon_basis`
: Character. Indicator for the carbon dimension: `"emission"` (total emission per ha, default, as in version 1.0.0) or `"intensity"` (emission per kg yield). Intensity avoids penalising high-yielding treatments for larger absolute emissions.

`verbose`
: Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Each dimension indicator is min-max normalized across the treatments supplied (carbon is inverted, so lower emission scores higher). Scores are therefore relative to the set of treatments being compared; with only two treatments every score is 0 or 1. Min-max scaling stretches any spread to the full 0 to 1 range, so a small absolute difference (for example 2 percent in IUE) can look as large as a big one; check the raw indicators before interpreting. When every treatment receives the same N, PFP is proportional to yield, so the function warns and `nutrient_metric = "iue"` is the better choice.

Value

A data frame of class `nexus_result` with one row per treatment and columns:

treatment Character. Treatment name

yield_kg_ha Numeric. Crop yield

WUE_kg_mm Numeric. Water use efficiency

EUE_ratio Numeric. Energy use efficiency ratio

EROI Numeric. Energy return on investment

net_energy_MJ Numeric. Net energy balance

PFP_kg_kgN Numeric. Partial factor productivity of N (`NA` when N applied is missing or zero)

carbon_eff Numeric. Carbon efficiency

carbon_intensity Numeric. Carbon intensity

W_score Numeric. Normalized water score

E_score Numeric. Normalized energy score

F_score Numeric. Normalized food score

N_score Numeric. Normalized nutrient score

C_score Numeric. Normalized carbon score

nexus_index Numeric. Composite nexus index

IUE_kg_kgN Numeric. Internal utilization efficiency of N (NA when n_uptake is not supplied)

rank Integer. Rank by nexus index (1 = best)

The weights used are stored in `attr(result, "weights")`.

Examples

```
nexus_summary(
  yield = c(4500, 4200, 3800),
  water_consumed = c(450, 400, 350),
  energy_input = c(12000, 11000, 9500),
  energy_output = c(135000, 125000, 112000),
  n_applied = c(150, 120, 90),
  n_uptake = c(100, 90, 80),
  carbon_emission = c(2500, 2200, 1800),
  treatment_names = c("CA+Drip", "CT+Sprinkler", "ZT+Rainfed")
)

# Legume without fertilizer N, carbon scored per kg grain
nexus_summary(
  yield = c(950, 870, 780),
  water_consumed = c(260, 240, 220),
  energy_input = c(5200, 4800, 4100),
  energy_output = c(36000, 33000, 29500),
  n_applied = c(0, 0, 0),
  n_uptake = c(62, 57, 50),
  carbon_emission = c(610, 560, 470),
  nutrient_metric = "iue", carbon_basis = "intensity"
)
```

---

nexus_sustainability_score

Nexus Sustainability Score

Description

Comprehensive sustainability score with categorization based on user-defined thresholds.

Usage

```
nexus_sustainability_score(
  water_score,
  energy_score,
  food_score,
  nutrient_score,
```

```
  carbon_score,
  weights = rep(0.2, 5),
  thresholds = c(0.4, 0.6, 0.8),
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

water_score	Numeric vector. Water dimension score (0 to 1).
energy_score	Numeric vector. Energy dimension score (0 to 1).
food_score	Numeric vector. Food dimension score (0 to 1).
nutrient_score	Numeric vector. Nutrient dimension score (0 to 1).
carbon_score	Numeric vector. Carbon dimension score (0 to 1).
weights	Numeric vector of length 5. Default equal weights.
thresholds	Numeric vector of length 3. Breakpoints for sustainability categories: c(low, medium, high). Default c(0.4, 0.6, 0.8).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$NSS = \sum_{i=1}^{5} w_i S_i$$

where S_i are the dimension scores and w_i the weights (equal weights give the mean).

Value

A data frame with columns:

nexus_score Numeric. Composite sustainability score (0 to 1)

category Character. Sustainability category: Unsustainable, Low Sustainability, Moderately Sustainable, or Highly Sustainable

water Numeric. Water dimension input score

energy Numeric. Energy dimension input score

food Numeric. Food dimension input score

nutrient Numeric. Nutrient dimension input score

carbon Numeric. Carbon dimension input score

Examples

```
nexus_sustainability_score(
  water_score = c(0.85, 0.60, 0.45),
  energy_score = c(0.70, 0.55, 0.35),
  food_score = c(0.90, 0.80, 0.50),
  nutrient_score = c(0.65, 0.70, 0.40),
  carbon_score = c(0.80, 0.50, 0.30)
)
```

---

nexus_topsis *TOPSIS Ranking of Treatments*

Description

Ranks treatments by the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) using several nexus indicators at once. Unlike nexus_index, TOPSIS works on raw indicators in their own units and handles "lower is better" indicators directly.

Usage

```
nexus_topsis(
  data,
  weights = "equal",
  benefit = NULL,
  treatment_names = NULL,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

data	A data frame or matrix with treatments in rows and indicators in columns. Only numeric columns are used.
weights	Numeric vector of weights, one per numeric column, or a method name passed to nexus_weights ("equal", "entropy", "critic", "pca"). Default "equal".
benefit	Logical vector, one value per numeric column. TRUE for higher-is-better indicators, FALSE for lower-is-better indicators such as carbon footprint or energy intensity. Default all TRUE.
treatment_names	Character vector of treatment labels. Default uses row names or "T 1", "T 2", and so on.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Each column is vector-normalized, $r_{ij} = x_{ij} / \sqrt{\sum_i x_{ij}^2}$, and multiplied by its weight. The ideal solution takes the best value of each column and the anti-ideal the worst. The closeness coefficient is $C_i = D_i^- / (D_i^+ + D_i^-)$, where D^+ and D^- are Euclidean distances to the ideal and anti-ideal. C_i lies between 0 and 1; higher is better.

Value

A data frame of class `topsis_result` with columns `treatment`, `d_ideal`, `d_anti_ideal`, `closeness` and `rank` (1 = best). The weights used are stored in `attr(, "weights")`. Use `plot()` for a ranked bar chart.

References

Hwang, C.L. & Yoon, K. (1981). *Multiple Attribute Decision Making: Methods and Applications*. Springer, Berlin.

See Also

`nexus_weights`

Examples

```
data(arid_pulse_nexus)
d <- arid_pulse_nexus
ind <- data.frame(
  WUE = d$grain_yield / d$total_water,
  EUE = (d$energy_output_grain + d$energy_output_straw) / d$energy_input,
  Yield = d$grain_yield,
  IUE = d$grain_yield / d$n_uptake,
  CF = d$ghg_emission
)
nexus_topsis(ind, weights = "entropy",
             benefit = c(TRUE, TRUE, TRUE, TRUE, FALSE),
             treatment_names = d$treatment)
```

---

nexus_tradeoff *Nexus Trade-off Matrix*

Description

Computes pairwise correlation matrix among nexus dimensions to identify synergies (positive) and trade-offs (negative).

Usage

```
nexus_tradeoff(
  data,
  method = "pearson",
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

data	A data frame with numeric columns for each nexus dimension (e.g., WUE, EUE, yield, NUE, carbon efficiency).
method	Character. Correlation method: "pearson" (default), "spearman", or "kendall".
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Value

A numeric correlation matrix with pairwise correlations. When all indicators are higher-is-better, positive values suggest synergies and negative values trade-offs; for a lower-is-better indicator (such as carbon footprint) the reading reverses. The p-values of the pairwise correlation tests (cor.test) and the number of complete pairs are attached as attr(, "p_values") and attr(, "n"). The result has class tradeoff_matrix and behaves as an ordinary matrix; plot() or plot_tradeoff draws it.

Examples

```
df <- data.frame(WUE = c(8.5, 7.2, 6.5, 9.0, 7.8),
                 EUE = c(11.2, 12.5, 9.8, 10.5, 11.0),
                 Yield = c(1500, 1380, 1250, 1650, 1580),
                 NUE = c(25, 23, 20.8, 27.5, 26.3),
                 CF = c(2500, 3000, 1800, 2800, 2200))
nexus_tradeoff(df)
```

---

nexus_weights *Objective Weights for Nexus Indicators*

Description

Derives dimension weights from the data instead of assigning them by judgement. Supports equal weights, the Shannon entropy weight method, the CRITIC method and PCA-based weights. The result can be passed directly to the weights argument of nexus_index, nexus_sustainability_score, nexus_summary or nexus_topsis.

Usage

```
nexus_weights(
  data,
  method = c("entropy", "critic", "pca", "equal"),
  benefit = NULL,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

data	A data frame or matrix with treatments in rows and indicators (for example the five WEFNC dimension scores) in columns. Only numeric columns are used.
method	Character. One of "entropy" (default), "critic", "pca" or "equal".
benefit	Logical vector, one value per numeric column. TRUE means higher values are better, FALSE means lower values are better (for example carbon footprint). Default all TRUE. Direction changes entropy and CRITIC weights, because it changes the normalized values and the signs of correlations; PCA weights do not depend on it.
verbose	Logical. If TRUE, prints the weights. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Indicators are first min-max normalized to 0 to 1 (inverted where benefit = FALSE). With m treatments and n indicators:

Entropy. $p_{ij} = x_{ij} / \sum_i x_{ij}$, $e_j = -\frac{1}{\ln m} \sum_i p_{ij} \ln p_{ij}$ (with $0 \ln 0 = 0$), and $w_j = (1 - e_j) / \sum_j (1 - e_j)$. Indicators that vary more between treatments get more weight.

CRITIC. $C_j = \sigma_j \sum_k (1 - r_{jk})$ and $w_j = C_j / \sum_j C_j$, where σ_j is the standard deviation of the normalized indicator and r_{jk} the Pearson correlation between indicators. Indicators that vary more and overlap less with others get more weight.

PCA. Principal components of the standardized indicators are computed and components with eigenvalue of at least 1 are kept (at least one). The weight of indicator j is proportional to $\sum_k \lambda_k |v_{jk}|$, where λ_k is the eigenvalue and v_{jk} the loading.

Constant indicators receive zero weight under entropy, CRITIC and PCA. With very few treatments (fewer than about five) data-driven weights are unstable and should be reported with a sensitivity analysis (nexus_sensitivity).

Value

A named numeric vector of weights summing to 1, with the method stored in attr(, "method").

References

Hwang, C.L. & Yoon, K. (1981). *Multiple Attribute Decision Making: Methods and Applications*. Springer, Berlin.

Diakoulaki, D., Mavrotas, G. & Papayannakis, L. (1995). Determining objective weights in multiple criteria problems: The CRITIC method. *Computers & Operations Research*, 22(7), 763-770. doi:10.1016/03050548(94)00059H

See Also

nexus_topsis, nexus_sensitivity

Examples

```
data(arid_pulse_nexus)
d <- arid_pulse_nexus
ind <- data.frame(
  WUE = d$grain_yield / d$total_water,
  EUE = (d$energy_output_grain + d$energy_output_straw) / d$energy_input,
  Yield = d$grain_yield,
  IUE = d$grain_yield / d$n_uptake,
  CF = d$ghg_emission
)
nexus_weights(ind, method = "entropy",
              benefit = c(TRUE, TRUE, TRUE, TRUE, FALSE))
nexus_weights(ind, method = "critic",
              benefit = c(TRUE, TRUE, TRUE, TRUE, FALSE))
```

---

normalize_minmax *Normalize Data (Min-Max)*

Description

Min-max normalization to scale values between 0 and 1.

Usage

```
normalize_minmax(x, inverse = FALSE)
```

Arguments

x	Numeric vector.
inverse	Logical. If TRUE, higher original values receive lower normalized scores. Use for metrics where lower is better, such as carbon footprint or energy intensity. Default FALSE.

Details

Formula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Value

Numeric vector normalized to the 0 to 1 range. Returns 0.5 for all elements if the input has zero range.

Examples

```
normalize_minmax(c(10, 20, 30, 40, 50))
normalize_minmax(c(10, 20, 30, 40, 50), inverse = TRUE)
```

`normalize_zscore` *Normalize Data (Z-score)*

Description

Standardization to z-scores (mean = 0, standard deviation = 1).

Usage

```
normalize_zscore(x)
```

Arguments

`x`	Numeric vector.

Details

Formula:

$$Z = \frac{X - \mu}{\sigma}$$

Value

Numeric vector of z-scores. Returns all zeros if standard deviation is zero.

Examples

```
normalize_zscore(c(10, 20, 30, 40, 50))
```

`nutrient_balance` *Nutrient Balance Sheet*

Description

Input-output nutrient balance for major nutrients (N, P, K).

Usage

```
nutrient_balance(
  n_input,
  p_input,
  k_input,
  n_output,
  p_output,
  k_output,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`n_input`	Numeric. Total N input from all sources (kg/ha).
`p_input`	Numeric. Total P input (kg/ha).
`k_input`	Numeric. Total K input (kg/ha).
`n_output`	Numeric. Total N removal or uptake (kg/ha).
`p_output`	Numeric. Total P removal or uptake (kg/ha).
`k_output`	Numeric. Total K removal or uptake (kg/ha).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$NB = NutrientApplied - NutrientRemoved$$

Zero inputs are allowed (for example no K applied), giving a negative balance that shows nutrient mining.

Value

A data frame with five columns:

nutrient Character. Nutrient name: N, P, or K

input_kg_ha Numeric. Total nutrient input (kg/ha)

output_kg_ha Numeric. Total nutrient output (kg/ha)

balance_kg_ha Numeric. Balance: input minus output (kg/ha). Positive = surplus; negative = depletion

output_input_ratio Numeric. Output to input ratio (`NA` when the input is zero)

Examples

```
nutrient_balance(n_input = 120, p_input = 60, k_input = 40,
                 n_output = 95, p_output = 25, k_output = 80)
# No potassium applied: balance shows K mining
nutrient_balance(n_input = 20, p_input = 40, k_input = 0,
                 n_output = 56.8, p_output = 8.2, k_output = 45)
```

---

nutrient_harvest_index

Nutrient Harvest Index (NHI)

Description

Proportion of total plant nutrient partitioned to grain.

Usage

```
nutrient_harvest_index(
  grain_nutrient_uptake,
  total_nutrient_uptake,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

grain_nutrient_uptake	Numeric vector. Nutrient uptake in grain (kg/ha).
total_nutrient_uptake	Numeric vector. Total plant nutrient uptake (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$NHI = \frac{U_{grain}}{U_{total}}$$

where U_{grain} is nutrient uptake in grain and U_{total} is total plant nutrient uptake (kg/ha).

Value

Numeric vector. Nutrient harvest index (proportion, 0 to 1).

Examples

```
nutrient_harvest_index(grain_nutrient_uptake = 75,
                       total_nutrient_uptake = 100)
```

`nutrient_use_efficiency` *Comprehensive Nutrient Use Efficiency (NUE)*

Description

Computes all major nutrient use efficiency metrics for a given nutrient in a single call.

Usage

```
nutrient_use_efficiency(
  yield_treated,
  yield_control,
  nutrient_applied,
  uptake_treated,
  uptake_control,
  nutrient_name = "N",
  grain_uptake = NULL,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield_treated`	Numeric. Yield with nutrient applied (kg/ha).
`yield_control`	Numeric. Yield without nutrient (kg/ha).
`nutrient_applied`	Numeric. Nutrient applied (kg/ha).
`uptake_treated`	Numeric. Total nutrient uptake with fertilizer (kg/ha).
`uptake_control`	Numeric. Total nutrient uptake without fertilizer (kg/ha).
`nutrient_name`	Character. Name of the nutrient for labeling (e.g., "N", "P", "K"). Default "N".
`grain_uptake`	Numeric. Nutrient uptake in grain of the treated crop (kg/ha). Used to compute the nutrient harvest index. If `NULL` (default), the nutrient harvest index is returned as `NA`.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Returns agronomic efficiency (AE), physiological efficiency (PE), recovery efficiency (RE), partial factor productivity (PFP), internal utilization efficiency (IUE) and nutrient harvest index (NHI). See the individual functions for formulas. Versions before 1.1.0 returned a fixed NHI of 0.75; NHI is now computed from `grain_uptake`.

Value

A data frame with one row and seven columns:

nutrient Character. Nutrient name

agronomic_eff Numeric. Agronomic efficiency (kg/kg)

physiological_eff Numeric. Physiological efficiency (kg/kg)

recovery_eff Numeric. Recovery efficiency (proportion)

partial_factor_prod Numeric. Partial factor productivity (kg/kg)

internal_util_eff Numeric. Internal utilization efficiency (kg/kg)

nutrient_harvest_idx Numeric. Nutrient harvest index (grain uptake / total uptake). `NA` when `grain_uptake` is not supplied.

References

Dobermann, A. (2007). Nutrient use efficiency: measurement and management. In *Fertilizer Best Management Practices*, IFA, Paris, pp. 1-28. https://digitalcommons.unl.edu/agronomyfacpub/316/

Congreves, K.A. et al. (2021). Nitrogen use efficiency definitions of today and tomorrow. *Frontiers in Plant Science*, 12, 637108. doi:10.3389/fpls.2021.637108

Examples

```
nutrient_use_efficiency(
  yield_treated = 4500, yield_control = 3000,
  nutrient_applied = 120, uptake_treated = 100,
  uptake_control = 60, nutrient_name = "N", grain_uptake = 72
)
```

---

partial_factor_productivity

Partial Factor Productivity (PFP)

Description

Total yield per unit of nutrient applied, without deduction of control yield.

Usage

```
partial_factor_productivity(
  yield,
  nutrient_applied,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
nutrient_applied	Numeric vector. Nutrient applied (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$PFP = \frac{Y}{F}$$

Value

Numeric vector. Partial factor productivity (kg yield per kg nutrient applied).

Examples

```
partial_factor_productivity(yield = 4500, nutrient_applied = 120)
```

---

physiological_efficiency

Physiological Efficiency (PE)

Description

Yield increase per unit of nutrient uptake increase.

Usage

```
physiological_efficiency(
  yield_treated,
  yield_control,
  uptake_treated,
  uptake_control,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield_treated	Numeric vector. Yield with nutrient applied (kg/ha).
yield_control	Numeric vector. Yield without nutrient (kg/ha).
uptake_treated	Numeric vector. Total nutrient uptake with fertilizer (kg/ha).
uptake_control	Numeric vector. Total nutrient uptake without fertilizer (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$PE = \frac{Y_f - Y_0}{U_f - U_0}$$

Value

Numeric vector. Physiological efficiency (kg grain per kg nutrient uptake). Returns `NA` where uptake difference is zero.

Examples

```
physiological_efficiency(yield_treated = 4500, yield_control = 3000,
                         uptake_treated = 100, uptake_control = 60)
```

---

plot.cf_result *Plot method for cf_result*

Description

Plot method for cf_result

Usage

```
## S3 method for class 'cf_result'
plot(x, ...)
```

Arguments

`x`	An object of class `cf_result`.
`...`	Additional arguments passed to `barplot`.

Value

No return value, called for the side effect of generating a horizontal bar plot showing carbon footprint by emission source.

`plot.energy_budget` *Plot method for energy_budget*

Description

Stacked horizontal bars of input energy for each treatment, split by input item, by direct and indirect energy, or by renewable and non-renewable energy.

Usage

```
## S3 method for class 'energy_budget'
plot(
  x,
  type = c("items", "form", "source"),
  percent = FALSE,
  top = 7,
  title = NULL,
  ...
)
```

Arguments

`x`	An object of class `energy_budget`.
`type`	Character. `"items"` (default), `"form"` (direct/indirect) or `"source"` (renewable/non-renewable).
`percent`	Logical. If `TRUE`, bars show percentage shares instead of MJ/ha. Default `FALSE`.
`top`	Integer. For `type = "items"`, the number of largest items shown individually; the rest are combined as "Other". Default 7 (maximum 7).
`title`	Character. Plot title. Default `NULL`.
`...`	Unused.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  inputs <- data.frame(
    treatment = c("CT", "ZT"),
    human_labour_man = c(320, 240), diesel = c(65, 38),
    electricity = c(180, 75), nitrogen = c(20, 20),
    phosphorus = c(40, 40), seed = c(20, 20), herbicide = c(0, 0.75)
  )
  eb <- energy_budget(inputs, verbose = FALSE)
  plot(eb)
  plot(eb, type = "source", percent = TRUE)
```

```
}
```

`plot.nexus_analysis` *Plot method for nexus_analysis*

Description

Quick access to the main figures of a `nexus_analysis`.

Usage

```
## S3 method for class 'nexus_analysis'
plot(x, which = c("radar", "anova", "topsis", "heatmap"), ...)
```

Arguments

`x`	An object of class `nexus_analysis`.
`which`	Character. `"radar"` (default, one panel per treatment), `"anova"` (means, error bars and letters), `"topsis"` (ranking) or `"heatmap"` (dimension scores).
`...`	Passed to the underlying plot function.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(arid_pulse_trial)
  res <- nexus_analysis(
    arid_pulse_trial, treatment = "treatment", block = "replication",
    yield = "grain_yield", water = "total_water",
    energy_in = "energy_input", energy_out = "energy_output",
    emission = "ghg_emission", n_uptake = "n_uptake", verbose = FALSE)
  plot(res)
  plot(res, which = "anova")
}
```

`plot.nexus_anova` *Plot method for nexus_anova*

Description

Draws treatment means as horizontal bars with error bars and the letter groups from `nexus_anova`, one panel per response.

Usage

```
## S3 method for class 'nexus_anova'
plot(
  x,
  response = NULL,
  error = c("se", "sem", "none"),
  labels = NULL,
  ncol = NULL,
  title = NULL,
  ...
)
```

Arguments

`x`	An object of class `nexus_anova`.
`response`	Character vector of responses to plot. Default all.
`error`	Character. Error bars: `"se"` (standard error of each treatment mean, default), `"sem"` (pooled SEm from the ANOVA) or `"none"`.
`labels`	Optional named character vector to relabel panels, for example `c(grain_yield = "Grain yield (kg/ha)")`.
`ncol`	Integer. Number of panel columns. Default chosen automatically.
`title`	Character. Plot title. Default `NULL` (none).
`...`	Unused.

Details

Panels for responses whose treatment F test is not significant are marked "(ns)" and show no letters.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(arid_pulse_trial)
  res <- nexus_anova(arid_pulse_trial,
                     response = c("grain_yield", "n_uptake"),
                     treatment = "treatment", block = "replication",
                     verbose = FALSE)
  plot(res, labels = c(grain_yield = "Grain yield (kg/ha)",
                       n_uptake = "N uptake (kg/ha)"))
}
```

---

plot.nexus_result *Plot method for nexus_result*

Description

Plot method for nexus_result

Usage

```
## S3 method for class 'nexus_result'
plot(x, type = c("radar", "heatmap"), ...)
```

Arguments

x	An object of class nexus_result.
type	Character. Plot type: "radar" (default) or "heatmap".
...	Additional arguments passed to nexus_radar or nexus_heatmap.

Value

Invisibly returns the scores matrix. Called for the side effect of generating a radar or heatmap plot.

---

plot.rank_compare *Plot method for rank_compare*

Description

Rank chart: each line follows one treatment across the eight rankings, so stable and unstable treatments are easy to see.

Usage

```
## S3 method for class 'rank_compare'
plot(x, title = NULL, ...)
```

Arguments

x	An object of class rank_compare.
title	Character. Plot title. Default NULL.
...	Unused.

Value

A ggplot2 object.

---

plot.topsis_result *Plot method for topsis_result*

Description

Ranked horizontal bars of the TOPSIS closeness coefficient, with the best treatment highlighted.

Usage

```
## S3 method for class 'topsis_result'
plot(x, highlight = 1, title = NULL, ...)
```

Arguments

x	An object of class topsis_result from nexus_topsis.
highlight	Integer. Number of top-ranked treatments to highlight. Default 1.
title	Character. Plot title. Default NULL.
...	Unused.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(arid_pulse_nexus)
  d <- arid_pulse_nexus
  ind <- data.frame(WUE = d$grain_yield / d$total_water,
                    Yield = d$grain_yield, CF = d$ghg_emission)
  ts <- nexus_topsis(ind, benefit = c(TRUE, TRUE, FALSE),
                     treatment_names = d$treatment, verbose = FALSE)
  plot(ts)
}
```

plot.tradeoff_matrix *Plot method for tradeoff_matrix*

Description

Plot method for tradeoff_matrix

Usage

```
## S3 method for class 'tradeoff_matrix'
plot(x, ...)
```

Arguments

x	An object of class tradeoff_matrix.
...	Passed to plot_tradeoff, for example benefit.

Value

A ggplot2 object.

plot_sensitivity *Plot Nexus Sensitivity Analysis (ggplot2)*

Description

Visualises the output of nexus_sensitivity as faceted line plots showing how the nexus index changes with dimension weights.

Usage

```
plot_sensitivity(sa, title = "Nexus Weight Sensitivity")
```

Arguments

sa	Data frame returned by nexus_sensitivity.
title	Character. Plot title.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  sa <- nexus_sensitivity(
    water_score = c(0.9, 0.5), energy_score = c(0.6, 0.8),
    food_score = c(0.8, 0.7), nutrient_score = c(0.7, 0.6),
    carbon_score = c(0.5, 0.9),
    treatment_names = c("CA", "CT"), steps = 10
  )
  plot_sensitivity(sa)
}
```

`plot_tradeoff` *Plot a Nexus Trade-off Matrix*

Description

Draws the correlation matrix from `nexus_tradeoff` as a heatmap with a diverging scale (blue positive, red negative, grey near zero). Cells show the coefficient and significance stars. Supply `benefit` to orient the indicators so that blue always means a synergy and red a trade-off.

Usage

```
plot_tradeoff(
  cor_mat,
  p_values = attr(cor_mat, "p_values"),
  benefit = NULL,
  triangle = c("lower", "full"),
  digits = 2,
  title = NULL
)
```

Arguments

`cor_mat`	A correlation matrix, normally from `nexus_tradeoff`.
`p_values`	Matrix of p-values. Default taken from `attr(cor_mat, "p_values")`; if absent, no stars are shown.
`benefit`	Optional logical vector, one value per indicator: `TRUE` if higher is better, `FALSE` if lower is better (for example GHG emission). When given, each coefficient is multiplied by the product of the two directions, so a positive value always means both indicators improve together (synergy). When `NULL` (default), the raw correlations are shown.
`triangle`	Character. `"lower"` (default) shows each pair once; `"full"` shows the whole matrix.
`digits`	Integer. Decimal places for the coefficients. Default 2.
`title`	Character. Plot title. Default `NULL`.

Details

Stars: * p < 0.05, ** p < 0.01, *** p < 0.001 (orienting changes only the sign, not the p-value). Without `benefit`, a negative correlation is not necessarily a trade-off: lower GHG emission with higher yield is a synergy. With few treatments, correlations are imprecise; interpret them with care.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(arid_pulse_trial)
  d <- arid_pulse_trial
  ind <- data.frame(Yield = d$grain_yield,
                    WUE = d$grain_yield / d$total_water,
                    EUE = d$energy_output / d$energy_input,
                    IUE = d$grain_yield / d$n_uptake,
                    GHG = d$ghg_emission)
  tm <- nexus_tradeoff(ind, verbose = FALSE)
  plot_tradeoff(tm)
  # GHG is lower-is-better: orient so blue = synergy, red = trade-off
  plot_tradeoff(tm, benefit = c(TRUE, TRUE, TRUE, TRUE, FALSE))
}
```

`print.cf_result` *Print method for cf_result*

Description

Print method for cf_result

Usage

```
## S3 method for class 'cf_result'
print(x, ...)
```

Arguments

`x`	An object of class `cf_result`.
`...`	Additional arguments passed to `print`.

Value

Invisibly returns the input object `x`. Called for the side effect of printing a formatted carbon footprint summary to the console.

print.energy_budget *Print method for energy_budget*

Description

Print method for energy_budget

Usage

```
## S3 method for class 'energy_budget'
print(x, ...)
```

Arguments

x	An object of class energy_budget.
...	Additional arguments passed to print.data.frame.

Value

Invisibly returns x. Called for the side effect of printing the energy budget summary.

print.nexus_analysis *Print method for nexus_analysis*

Description

Print method for nexus_analysis

Usage

```
## S3 method for class 'nexus_analysis'
print(x, ...)
```

Arguments

x	An object of class nexus_analysis.
...	Additional arguments (currently unused).

Value

Invisibly returns x. Called for the side effect of printing the ranking of treatments.

print.nexus_anova *Print method for nexus_anova*

Description

Print method for nexus_anova

Usage

```
## S3 method for class 'nexus_anova'
print(x, ...)
```

Arguments

x	An object of class nexus_anova.
...	Additional arguments passed to print.data.frame.

Value

Invisibly returns x. Called for the side effect of printing the formatted comparison table.

print.nexus_result *Print method for nexus_result*

Description

Print method for nexus_result

Usage

```
## S3 method for class 'nexus_result'
print(x, ...)
```

Arguments

x	An object of class nexus_result.
...	Additional arguments passed to print.data.frame.

Value

Invisibly returns the input object x. Called for the side effect of printing a formatted nexus analysis summary to the console.

print.rank_compare *Print method for rank_compare*

Description

Print method for rank_compare

Usage

```
## S3 method for class 'rank_compare'
print(x, ...)
```

Arguments

x	An object of class rank_compare.
...	Additional arguments passed to print.data.frame.

Value

Invisibly returns x. Called for the side effect of printing the ranks and the concordance statistics.

print.sustainability_result *Print method for sustainability_result*

Description

Print method for sustainability_result

Usage

```
## S3 method for class 'sustainability_result'
print(x, ...)
```

Arguments

x	An object of class sustainability_result.
...	Additional arguments passed to print.data.frame.

Value

Invisibly returns the input object x. Called for the side effect of printing sustainability scores and categories to the console.

print.tradeoff_matrix *Print method for tradeoff_matrix*

Description

Print method for tradeoff_matrix

Usage

```
## S3 method for class 'tradeoff_matrix'
print(x, ...)
```

Arguments

x	An object of class tradeoff_matrix from nexus_tradeoff.
...	Additional arguments passed to print.

Value

Invisibly returns x. Called for the side effect of printing the correlation matrix.

production_efficiency_index *Production Efficiency Index (PEI)*

Description

Yield produced per unit of production cost, measuring economic efficiency of crop production.

Usage

```
production_efficiency_index(
  yield,
  cost,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
cost	Numeric vector. Total cost of production (currency/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$PEI = \frac{Y}{C}$$

where Y is yield (kg/ha) and C is the total cost of production (currency/ha).

Value

Numeric vector. Production efficiency index (kg/currency unit).

Examples

```
production_efficiency_index(yield = 4500, cost = 28500)
```

---

protein_yield *Protein Yield*

Description

Total protein output per hectare from crop production.

Usage

```
protein_yield(
  yield,
  protein_content,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yield	Numeric vector. Crop yield (kg/ha).
protein_content	Numeric vector. Protein content (percent).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

Formula:

$$PY = Yield \times ProteinContent/100$$

Value

Numeric vector. Protein yield (kg/ha).

Examples

```
protein_yield(yield = 4500, protein_content = 12.5)
```

---

recovery_efficiency *Recovery Efficiency (RE)*

Description

Proportion of applied nutrient that is recovered in crop biomass.

Usage

```
recovery_efficiency(
  uptake_treated,
  uptake_control,
  nutrient_applied,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

uptake_treated	Numeric vector. Total nutrient uptake with fertilizer (kg/ha).
uptake_control	Numeric vector. Total nutrient uptake without fertilizer (kg/ha).
nutrient_applied	Numeric vector. Nutrient applied (kg/ha).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$RE = \frac{U_f - U_0}{F}$$

Value

Numeric vector. Recovery efficiency as a proportion (0 to 1, but can exceed 1 due to priming effects or added benefits).

Examples

```
recovery_efficiency(uptake_treated = 100, uptake_control = 60,
                    nutrient_applied = 120)
```

`soil_carbon_stock` *Soil Organic Carbon Stock (SOC)*

Description

Estimates soil organic carbon stock for a given soil depth.

Usage

```
soil_carbon_stock(
  soc_pct,
  bulk_density,
  depth = 30,
  coarse_fraction = 0,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`soc_pct`	Numeric vector. Soil organic carbon content (percent).
`bulk_density`	Numeric vector. Soil bulk density (Mg/m3 or g/cm3).
`depth`	Numeric. Soil sampling depth (cm). Default 30.
`coarse_fraction`	Numeric vector. Volume fraction of coarse fragments greater than 2 mm (0 to 1). Default 0.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

$$SOC = SOC_{pct} \times BD \times D \times (1 - CF)$$

where SOC_pct is organic carbon (percent), BD is bulk density (Mg/m3), D is depth (cm) and CF is the coarse fragment fraction. The product of percent, Mg/m3 and cm is already Mg C/ha: (SOC/100) x BD x (D/100 m) x 10000 m2/ha.

Versions before 1.1.0 multiplied by an extra factor of 0.1, which gave kg C/m2 while labelling the result Mg C/ha, so stocks were 10 times too low.

Value

Numeric vector. SOC stock (Mg C/ha, equivalent to t C/ha).

References

Batjes, N.H. (1996). Total carbon and nitrogen in the soils of the world. *European Journal of Soil Science*, 47(2), 151-163. doi:10.1111/j.13652389.1996.tb01386.x

Examples

```
soil_carbon_stock(soc_pct = 0.65, bulk_density = 1.45, depth = 30)
```

`specific_energy` *Specific Energy*

Description

Energy input per unit area per unit time, useful for comparing cropping systems of different durations.

Usage

```
specific_energy(
  energy_in,
  duration,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`energy_in`	Numeric vector. Total energy input (MJ/ha).
`duration`	Numeric vector. Crop duration (days).
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Formula:

$$SE = \frac{E_{in}}{d}$$

where E_{in} is energy input (MJ/ha) and d is crop duration (days).

Note that much of the energy budgeting literature uses "specific energy" for energy input per kg of produce (MJ/kg). That quantity is returned by `energy_intensity`.

Value

Numeric vector. Specific energy (MJ/ha/day).

Examples

```
specific_energy(energy_in = 12000, duration = 150)
```

summary.nexus_result *Summary method for nexus_result*

Description

Summary method for nexus_result

Usage

```
## S3 method for class 'nexus_result'
summary(object, ...)
```

Arguments

object	An object of class nexus_result.
...	Additional arguments (currently unused).

Value

Invisibly returns the input object object. Called for the side effect of printing a detailed dimension-wise summary to the console.

system_productivity_index *System Productivity Index (SPI)*

Description

Converts yields from a cropping system to a common crop-equivalent yield based on economic value.

Usage

```
system_productivity_index(
  yields,
  prices,
  base_price,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

yields	Numeric vector. Yields of individual crops in the system (kg/ha).
prices	Numeric vector. Market prices of individual crops (currency/kg). Must be the same length as yields.
base_price	Numeric. Price of the base or reference crop (currency/kg).
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

$$SPI = \frac{\sum(Y_i \times P_i)}{P_{base}}$$

Value

Numeric. System productivity expressed as base crop equivalent yield (kg/ha).

Examples

```
system_productivity_index(yields = c(5000, 4500),
                          prices = c(22, 25), base_price = 25)
```

---

water_footprint *Water Footprint (WF)*

Description

Computes the water footprint of crop production decomposed into green, blue, and grey components following the Water Footprint Assessment framework.

Usage

```
water_footprint(
  green_water,
  blue_water,
  grey_water = 0,
  yield,
  area = 1,
  per_ton = FALSE,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

green_water	Numeric vector. Green water use, i.e., effective rainfall consumed by the crop during growth (mm).
blue_water	Numeric vector. Blue water use, i.e., irrigation water applied and consumed (mm).
grey_water	Numeric vector. Grey water, i.e., freshwater volume required to dilute pollutant load to acceptable standards (mm). Default is 0.
yield	Numeric vector. Crop yield (kg/ha).
area	Deprecated and ignored. Water use (mm) and yield (kg/ha) are both per hectare, so the area cancels out. Versions before 1.1.0 multiplied the footprint by area, which was incorrect.
per_ton	Logical. If TRUE, returns water footprint in m3/ton. Default FALSE, returning m3/kg.
verbose	Logical. If TRUE, prints informational messages. Default getOption("wefnexus.verbose", TRUE), so options(wefnexus.verbose = FALSE) silences all functions.

Details

The water footprint per unit mass of product is:

$$WF = \frac{CWU \times 10}{Y}$$

where CWU is crop water use (mm) and Y is yield (kg/ha). The factor 10 converts mm over 1 ha to m3.

Value

A data frame with four numeric columns:

green_wf Green water footprint (m3/kg or m3/ton)

blue_wf Blue water footprint (m3/kg or m3/ton)

grey_wf Grey water footprint (m3/kg or m3/ton)

total_wf Total water footprint (m3/kg or m3/ton)

References

Hoekstra, A.Y., Chapagain, A.K., Aldaya, M.M. & Mekonnen, M.M. (2011). *The Water Footprint Assessment Manual: Setting the Global Standard*. Earthscan, London. ISBN:9781849712798.

Mialyk, O. et al. (2024). Water footprints and crop water use of 175 individual crops for 1990-2019 simulated with a global crop model. *Scientific Data*, 11, 200. doi:10.1038/s41597024030513

Examples

```
water_footprint(green_water = 300, blue_water = 200,
                grey_water = 50, yield = 4000)
```

`water_productivity` *Water Productivity (WP)*

Description

Computes physical or economic water productivity per unit of water used.

Usage

```
water_productivity(
  yield,
  water_applied,
  price = NULL,
  unit = "mm",
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`water_applied`	Numeric vector. Total water applied (mm or m3/ha).
`price`	Numeric. Market price per kg of produce (currency/kg). If `NULL` (default), returns physical water productivity.
`unit`	Character. Unit of water input: `"mm"` (default) or `"m3"`.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Physical water productivity:

$$WP = \frac{Y}{W_{m3}}$$

Economic water productivity:

$$WP_{econ} = \frac{Y \times P}{W_{m3}}$$

When `unit = "mm"`, water is converted to m3/ha using 1 mm = 10 m3/ha.

Value

Numeric vector. Physical water productivity in kg/m3 when `price` is `NULL`, or economic water productivity in currency/m3 when `price` is specified.

References

Molden, D. et al. (2010). Improving agricultural water productivity: Between optimism and caution. *Agricultural Water Management*, 97(4), 528-535. doi:10.1016/j.agwat.2009.03.023

Examples

```
water_productivity(4500, 500, unit = "mm")
water_productivity(4500, 500, price = 25, unit = "mm")
```

`water_use_efficiency` *Water Use Efficiency (WUE)*

Description

Computes water use efficiency as the ratio of economic yield to total water consumed (evapotranspiration or irrigation plus rainfall).

Usage

```
water_use_efficiency(
  yield,
  water_consumed,
  verbose = getOption("wefnexus.verbose", TRUE)
)
```

Arguments

`yield`	Numeric vector. Crop yield (kg/ha).
`water_consumed`	Numeric vector. Total water consumed (mm), typically evapotranspiration (ET), or irrigation applied plus effective rainfall.
`verbose`	Logical. If `TRUE`, prints informational messages. Default `getOption("wefnexus.verbose", TRUE)`, so `options(wefnexus.verbose = FALSE)` silences all functions.

Details

Water use efficiency is defined as:

$$WUE = \frac{Y}{W}$$

where Y is economic yield (kg/ha) and W is total water consumed (mm).

Value

Numeric vector. Water use efficiency in kg/ha/mm. Higher values indicate more efficient use of water for crop production.

References

Hoover, D.L. et al. (2023). Indicators of water use efficiency across diverse agroecosystems and scales. *Science of the Total Environment*, 864, 160992. doi:10.1016/j.scitotenv.2022.160992

Examples

```
# Wheat yield with different irrigation levels
yield <- c(4500, 4200, 3800, 3500)
water <- c(450, 400, 350, 300)
water_use_efficiency(yield, water)
```

Index

∗ **datasets**
  arid_pulse_nexus, 6
  arid_pulse_trial, 8
  energy_coefficients, 21

agronomic_efficiency, 5
aov, *42*
arid_pulse_nexus, 6, *8*
arid_pulse_trial, 8

barplot, *67*

caloric_yield, 9
carbon_efficiency, 10
carbon_footprint, 11, *32*
carbon_sequestration_rate, 13
carbon_sustainability_index, 14
cor.test, *58*
crop_water_productivity, 15
crop_water_stress_index, 16
crop_yield_index, 17

depleted_fraction, 18

energy_budget, *5*, 19, *21*
energy_coefficients, *5*, *19, 20*, 21
energy_input, *20*, 22
energy_intensity, 23, *83*
energy_output, 24
energy_productivity, 25
energy_profitability, 26
energy_use_efficiency, *20*, 27
eroi, 28

food_productivity_index, 29

ghg_emission, 30
global_warming_potential, 32

harvest_index, 34
human_energy_profitability, 35

internal_utilization_efficiency, 36
irrigation_efficiency, 37

land_equivalent_ratio, 38

net_energy, 39
nexus_analysis, *5*, *8*, 40, *69*
nexus_anova, *5*, *8*, *41*, 42, *70*
nexus_heatmap, 44, *71*
nexus_heatmap_gg, 45
nexus_index, 46, *56*, *58*
nexus_radar, 47, *71*
nexus_radar_gg, 48
nexus_rank_compare, 49
nexus_sensitivity, *50*, 51, *59, 60*, *73*
nexus_summary, *41*, 52, *58*
nexus_sustainability_score, 54, *58*
nexus_topsis, *5*, *41*, *50*, 56, *58*, *60*, *72*
nexus_tradeoff, 57, *74*, *79*
nexus_weights, *5*, *41*, *50*, *53*, *56, 57*, 58
normalize_minmax, 60
normalize_zscore, 61
nutrient_balance, 62
nutrient_harvest_index, 63
nutrient_use_efficiency, 64

partial_factor_productivity, 65
physiological_efficiency, 66
plot.cf_result, 67
plot.energy_budget, 68
plot.nexus_analysis, 69
plot.nexus_anova, *43*, 70
plot.nexus_result, 71
plot.rank_compare, 71
plot.topsis_result, 72
plot.tradeoff_matrix, 73
plot_sensitivity, 73
plot_tradeoff, *58*, *73*, 74
print.cf_result, 75
print.energy_budget, 76

print.nexus_analysis, 76
print.nexus_anova, 77
print.nexus_result, 77
print.rank_compare, 78
print.sustainability_result, 78
print.tradeoff_matrix, 79
production_efficiency_index, 79
protein_yield, 80

recovery_efficiency, 81

soil_carbon_stock, 82
specific_energy, 83
summary.nexus_result, 84
system_productivity_index, 84

water_footprint, 85
water_productivity, 87
water_use_efficiency, 88
wefnexus *(*wefnexus-package*)*, 4
wefnexus-package, 4