

Package ‘tikatuwq’

October 22, 2025

Type Package

Title Water Quality Assessment and Environmental Compliance in Brazil

Version 0.7.2

Maintainer Vinicius Saraiva Santos <vinisaraiva@gmail.com>

Description Tools to import, clean, validate, and analyze freshwater quality data in Brazil. Implements water quality indices including the Water Quality Index (WQI/IQA), the Trophic State Index (TSI/IET) after Carlson (1977) <[doi:10.4319/lo.1977.22.2.0361](https://doi.org/10.4319/lo.1977.22.2.0361)> and Lamparelli (2004) <<https://www.teses.usp.br/teses/disponiveis/41/41134/tde-20032006-075813/publico/TeseLamparelli2004.pdf>>, and the National Sanitation Foundation Water Quality Index (NSF WQI) <[doi:10.1007/s11157-023-09650-7](https://doi.org/10.1007/s11157-023-09650-7)>. The package also checks compliance with Brazilian standard CONAMA Resolution 357/2005 <https://conama.mma.gov.br/?id=450&option=com_sisconama&task=arquivo.download> and generates reproducible reports for routine monitoring workflows.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

Depends R (>= 4.1)

Imports dplyr, readr, tibble, rlang, stats, utils, ggplot2, tidyr, lubridate, stringr, glue, scales, broom, purrr, leaflet

Suggests testthat (>= 3.0.0), spelling, rmarkdown, knitr, pkgdown

VignetteBuilder knitr

URL <https://github.com/tikatuwq/tikatuwq>,
<https://tikatuwq.github.io/tikatuwq/>

BugReports <https://github.com/tikatuwq/tikatuwq/issues>

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Vinicius Saraiva Santos [aut, cre] (ORCID:
<<https://orcid.org/0009-0007-1387-7927>>)

Repository CRAN

Date/Publication 2025-10-22 13:40:02 UTC

Contents

clean_units	3
conama_check	3
conama_limits	4
conama_report	5
conama_summary	6
conama_text	7
generate_analysis	8
iet_carlson	9
iet_lamparelli	10
iqa	12
nsfwqi	13
param_plot	14
param_plot_multi	15
param_summary	16
param_summary_multi	18
param_trend	19
param_trend_multi	20
plot_box	21
plot_heatmap	21
plot_iqa	22
plot_map	23
plot_series	24
plot_trend	24
read_wq	26
render_report	27
resume_wq	28
trend_param	29
validate_wq	30
wq_demo	31

Index	33
--------------	-----------

clean_units	<i>Normalize/standardize units (placeholder)</i>
-------------	--

Description

Extension point to normalize units (e.g., mg/L, uS/cm). Currently returns df unchanged.

Usage

```
clean_units(df, units_map = NULL)
```

Arguments

df	Input data frame / tibble.
units_map	Optional mapping of units.

Value

The input df unchanged (placeholder).

See Also

[read_wq\(\)](#)

Examples

```
clean_units(data.frame(ph = c(7, 7.2), od = c(6.5, 7.0)))
```

conama_check	<i>CONAMA conformity check (detailed; default class = "2")</i>
--------------	--

Description

For each parameter present in df, adds columns:

- *_ok (logical),
- *_status one of "ok", "acima_do_maximo", "abaixo_do_minimo",
- *__lim_min and *__lim_max (thresholds used),
- *__delta (difference to the relevant limit; >0 above max, <0 below min, 0 if ok).

If multiple limit rows exist for the same parameter, *_ok is TRUE if any row is satisfied; for status/lim_min/lim_max/delta, the first satisfied row is chosen; if none satisfy, the row with the smallest absolute violation (min |delta|) is used.

Usage

```
conama_check(df, classe = "2")
```

Arguments

df A tibble/data.frame with parameter columns (e.g., ph, turbidez, od, dbo).
classe Character class label (e.g., "especial", "1", "2", "3", "4").

Value

The input df with additional columns per parameter as described.

See Also

[conama_limits\(\)](#), [conama_summary\(\)](#), [conama_report\(\)](#), [conama_text\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
head(conama_check(wq_demo, classe = "2"))

## End(Not run)
```

conama_limits

Limits for Brazilian CONAMA 357/2005

Description

Returns the parameter limits defined by CONAMA Resolution 357/2005 for a given water-use class.

Usage

```
conama_limits(class)
```

Arguments

class Integer or character. Target class (e.g., 1, 2, 3, 4 or "special"), according to CONAMA 357/2005.

Value

A tibble/data frame with one row per parameter and regulatory thresholds. Typical columns:

- parametro: parameter name (character, normalized to snake_case)
- classe: class label (character)
- min/max (or equivalents): numeric thresholds (may be NA)
- other metadata columns if present (e.g., unit, criterion)

Examples

```
# Class 2 thresholds (first rows)
head(conama_limits(2))
```

conama_report	CONAMA conformity report (table)
---------------	----------------------------------

Description

CONAMA conformity report (table)

Usage

```
conama_report(
  df,
  classe = "2",
  only_violations = TRUE,
  pretty = FALSE,
  decimal_mark = ",",
  big_mark = "."
)
```

Arguments

- df Input data
- classe CONAMA class label (e.g., "2")
- only_violations If TRUE, returns only rows with status != "ok"
- pretty If TRUE, returns formatted numeric columns for display
- decimal_mark Decimal separator (default ",")
- big_mark Thousands separator (default ".")

Value

A tibble. When pretty = FALSE: parametro, valor, lim_min, lim_max, status, delta. When pretty = TRUE, numeric columns are formatted as character with "natural" decimals.

See Also

```
conama\_summary\(\), conama\_text\(\)
```

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
conama_report(wq_demo, classe = "2", only_violations = TRUE)
conama_report(wq_demo, classe = "2", only_violations = TRUE, pretty = TRUE)

## End(Not run)
```

conama_summary	<i>CONAMA conformity summary (long format)</i>
----------------	--

Description

CONAMA conformity summary (long format)

Usage

```
conama_summary(df, classe = "2")
```

Arguments

df	Input data
classe	CONAMA class label

Value

A tibble with columns: parametro, valor, lim_min, lim_max, status, ok, delta.

See Also

[conama_check\(\)](#), [conama_report\(\)](#), [conama_text\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
head(conama_summary(wq_demo, classe = "2"))

## End(Not run)
```

conama_text	<i>Text summary of conformity (bulleted, formatted)</i>
-------------	---

Description

Text summary of conformity (bulleted, formatted)

Usage

```
conama_text(  
  df,  
  classe = "2",  
  only_violations = FALSE,  
  decimal_mark = ",",  
  big_mark = "."  
)
```

Arguments

df	Input data
classe	CONAMA class label
only_violations	If TRUE, list only parameters with violation
decimal_mark	Decimal separator (default ",")
big_mark	Thousands separator (default ".")

Value

Character vector of lines (first line is a header, the rest are bullets).

See Also

[conama_summary\(\)](#), [conama_report\(\)](#)

Examples

```
## Not run:  
data("wq_demo", package = "tikatuwq")  
cat(conama_text(wq_demo, classe = "2"), sep = "\n")  
  
## End(Not run)
```

generate_analysis	<i>Generate analytical paragraphs (rule-based)</i>
-------------------	--

Description

Produce 3–5 paragraphs short, legible by humans, summarizing the quality of the water from the IQA/WQI, conformity with the CONAMA 357/2005 and (optionally) simple temporal trends. It is **rule-based** (does not use IA) and accepts optional metadata to compose the text.

Usage

```
generate_analysis(
  df,
  classe_conama = "2",
  incluir_tendencia = TRUE,
  parametros_tendencia = c("turbidez", "od", "pH"),
  contexto = list(rio = NA, periodo = NA, cidade = NA)
)
```

Arguments

df	Data frame containing at least the point column. It is recommended to also include the necessary columns for CONAMA checks and for the IQA calculation.
classe_conama	Character (ex. "2"). Target class for the CONAMA 357/2005 resolution check.
incluir_tendencia	Logical; if TRUE, calculates simple linear trends over time.
parametros_tendencia	Character vector; names of the parameters to test for temporal trend.
contexto	List of optional metadata (PT/EN), for example <code>list(rio = "Rio Pardo", periodo = "jan-jun/2025", cidade = "Lencois")</code> . The accepted keys are <code>rio/river</code> , <code>periodo/period</code> , <code>cidade</code> .

Value

Character vector with 3 to 5 analytical paragraphs ready for the report.

See Also

[iqa\(\)](#), [conama_check\(\)](#)

Examples

```
## Not run:
library(tikatuwq)
data("wq_demo")
txt <- generate_analysis(
```



```

df = wq_demo,
classe_conama = "2",
incluir_tendencia = TRUE,
parametros_tendencia = c("turbidez", "od", "pH"),
contexto = list(rio = "Rio Azul", periodo = "jan-jun/2025")
)
cat(paste(txt, collapse = "\n\n"))

## End(Not run)

```

iet_carlson

Trophic State Index (Carlson)

Description

Computes Carlson's Trophic State Index (TSI/IET) from Secchi depth, chlorophyll-a, and total phosphorus. Returns component scores and the overall IET as the row-wise mean of available components.

Usage

```
iet_carlson(secchi = NULL, clorofila = NULL, tp = NULL)
```

Arguments

secchi	Numeric vector with Secchi depth (m).
clorofila	Numeric vector with chlorophyll-a (ug/L).
tp	Numeric vector with total phosphorus (ug/L).

Details

Implemented component formulas (Carlson 1977):

- $TSI_Secchi = 60 - 14.41 * \log_{10}(secchi)$
- $TSI_Chla = 9.81 * \log_{10}(clorofila) + 30.6$
- $TSI_TP = 14.42 * \log_{10}(tp) + 4.15$

Inputs may contain NA and are recycled according to R rules. The overall index IET is the row mean of the available components (`na.rm = TRUE`).

Value

A data frame with columns (when applicable):

- `TSI_Secchi` — component from Secchi depth.
- `TSI_Chla` — component from chlorophyll-a.
- `TSI_TP` — component from total phosphorus.
- `IET` — overall Carlson index (row mean).

References

Carlson, R. E. (1977). *A trophic state index for lakes*. Limnology and Oceanography, 22(2), 361–369. doi:10.4319/lo.1977.22.2.0361

See Also

[iet_lamparelli\(\)](#), [iqa\(\)](#), [conama_check\(\)](#)

Examples

```
# Example data
secchi <- c(1.2, 0.8, 0.4)      # m
clorofila <- c(5, 12, 30)      # ug/L
tp <- c(20, 40, 70)           # ug/L

iet_carlson(secchi = secchi, clorofila = clorofila, tp = tp)

# With only one component
iet_carlson(secchi = secchi)
```

iet_lamparelli	<i>Trophic State Index (Lamparelli)</i>
----------------	---

Description

Computes components of the Lamparelli trophic state index (TSI/IET) from total phosphorus, chlorophyll-a, and Secchi depth, and returns the overall Lamparelli index as the row-wise mean of available components.

Usage

```
iet_lamparelli(
  tp = NULL,
  chla = NULL,
  sd = NULL,
  ambiente = c("rio", "reservatorio")
)
```

Arguments

tp	Numeric total phosphorus (mg/L).
chla	Numeric chlorophyll-a (ug/L).
sd	Numeric Secchi disk depth (m).
ambiente	Character, environment type: 'rio' or 'reservatorio'.

Details

Implemented component formulas (simple skeleton):

- $IET_TP = 10 + 10 * \log_{10}(\max(tp, 0.001))$
- $IET_Chla = 10 + 10 * \log_{10}(\max(chla, 0.001))$
- $IET_Secchi = 60 - 14.41 * \log_{10}(\max(sd, 0.001))$

The overall IET_Lamp is the row mean of available components (`na.rm = TRUE`). Inputs are recycled by standard R vector recycling rules and can contain NA.

This is a minimal, pragmatic implementation intended for quick summaries; practitioners should confirm the most appropriate equations/coefficients for the specific waterbody type and region before regulatory use.

Value

A data frame with columns (when applicable):

- IET_TP — component from total phosphorus.
- IET_Chla — component from chlorophyll-a.
- IET_Secchi — component from Secchi depth.
- IET_Lamp — overall Lamparelli index (row mean).
- ambiente — the informed environment label.

References

Carlson, R. E. (1977). *A trophic state index for lakes*. Limnology and Oceanography, 22(2), 361–369. doi:10.4319/lo.1977.22.2.0361

Lamparelli, M. C. (2004). *Graus de trofia em corpos d'água do Estado de São Paulo*. (Tese de Doutorado). Instituto de Biociências, USP.

See Also

[iet_carlson\(\)](#), [iqa\(\)](#), [conama_check\(\)](#)

Examples

```
# Vectors (can include NA)
tp  <- c(0.02, 0.05, 0.10)      # mg/L
chla <- c(5, 12, 30)            # ug/L
sd  <- c(1.2, 0.8, 0.4)         # m

iet_lamparelli(tp = tp, chla = chla, sd = sd, ambiente = "reservatorio")

# With a single component:
iet_lamparelli(tp = tp, ambiente = "rio")
```

iqa	<i>Water Quality Index (WQI / IQA)</i>
-----	--

Description

Computes IQA/WQI by combining parameter-specific sub-scores (Qi) via a **weighted mean**. Sub-scores are obtained by piecewise-linear interpolation over approximate curves (CETESB/NSF-like).

Usage

```
iqa(  
  df,  
  pesos = c(od = 0.17, coliformes = 0.15, dbo = 0.1, nt_total = 0.1, p_total = 0.1,  
            turbidez = 0.08, tds = 0.08, pH = 0.12, temperatura = 0.1),  
  method = c("CETESB_approx"),  
  na_rm = FALSE  
)
```

Arguments

df	Data frame (or tibble) with required parameter columns. Expected defaults (Portuguese names): od, coliformes, dbo, nt_total, p_total, turbidez, tds, ph (or pH), temperatura.
pesos	Named numeric weights for each parameter (sum not required). Defaults follow CETESB/NSF practice: • od = 0.17 • coliformes = 0.15 • dbo = 0.10 • nt_total = 0.10 • p_total = 0.10 • turbidez = 0.08 • tds = 0.08 • pH = 0.12 (mapped to column ph if needed) • temperatura = 0.10
method	Character scalar; interpolation table set. Currently only "CETESB_approx".
na_rm	Logical; if FALSE (default), rows containing missing Qi values will trigger an error. If TRUE, the IQA is computed using only available parameters, with the denominator adjusted to the sum of the weights of present parameters.

Details

- Column name compatibility:
- The interpolation table uses the key "pH". If your data uses a ph column (lowercase), it is automatically mapped to the "pH" curve.

- All other parameter names are used as-is.

Values are clipped into $[0, 100]$ after aggregation.

Value

The input df with an added numeric column IQA. The attribute "iqa_method" is set on the returned data.frame/tibble.

Examples

```
# Minimal example using the demo data:
d <- wq_demo
d2 <- iqa(d, na_rm = TRUE)
head(d2$IQA)
```

nsfwqi

NSF Water Quality Index (NSF WQI, prototype)

Description

Computes a **prototype** NSF WQI as a weighted arithmetic mean of parameter sub-scores (Q_i) using simple piecewise rules. This is intended for quick demonstrations and is **not** a full replication of the original NSF curves.

Usage

```
nsfwqi(
  df,
  pesos = c(do = 0.17, fc = 0.16, ph = 0.11, bod = 0.11, temp_change = 0.1, po4 = 0.1,
    no3 = 0.1, turbidez = 0.08, sst = 0.07),
  na_rm = FALSE
)
```

Arguments

df	Data frame containing columns compatible with the mapping above.
pesos	Named numeric vector with parameter weights. Defaults follow a common NSF WQI variant: do=.17, fc=.16, ph=.11, bod=.11, temp_change=.10, po4=.10, no3=.10, turbidez=.08, sst=.07.
na_rm	Logical; allow NA per row and rescale weights to available parameters (TRUE) or error on missing inputs (FALSE).

Details

The function accepts both NSF-style column names and common Brazilian aliases. The mapping tried (if present) is:

- do <- od
- fc <- coliformes
- ph <- pH or ph
- bod <- dbo
- turbidez stays turbidez
- sst <- solidos_suspensos
- po4 <- po4 or p_ortofosfato
- no3 <- no3 or n_nitrato
- temp_change must be supplied as-is (delta T to reference)

If na_rm = TRUE, weights are rescaled **per row** to the parameters available in that row. If na_rm = FALSE (default), any missing required input leads to an error.

Value

The input df with an added numeric column NSFWQI.

Examples

```
d <- wq_demo
# create minimal aliases so the prototype can run
d$do <- d$od
d$fc <- d$coliformes
d$ph <- d$ph
d$bod <- d$dbod
# others are missing; use na_rm = TRUE to rescale weights by row
out <- nsfwqi(d, na_rm = TRUE)
head(out$NSFWQI)
```

param_plot

Plot temporal de um parametro (com filtro por rio e/ou ponto)

Description

Gera grafico temporal para **um parametro**, com opcoes de filtro por rios e/ou pontos. Se houver mais de um ponto, a cor diferencia pontos; opcional facet = TRUE para facetar por ponto. Pode adicionar reta de tendencia com add_trend = TRUE (lm).

Usage

```
param_plot(
  df,
  parametro,
  rios = NULL,
  pontos = NULL,
  add_trend = TRUE,
  facet = FALSE
)
```

Arguments

df	Data frame com data e a coluna do parametro. Idealmente contem ponto, e opcionalmente rio.
parametro	Character; nome do parametro.
rios	Vetor de nomes de rio a filtrar (opcional; usa coluna rio se existir).
pontos	Vetor de pontos a filtrar (opcional; usa coluna ponto se existir).
add_trend	Logical; se TRUE, adiciona geom_smooth(method = "lm", se = FALSE).
facet	Logical; se TRUE e houver ponto, aplica facet_wrap(~ponto).

Value

Objeto ggplot.

See Also

Other parameter-tools: [param_plot_multi\(\)](#), [param_summary\(\)](#), [param_summary_multi\(\)](#), [param_trend\(\)](#), [param_trend_multi\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
param_plot(wq_demo, "turbidez", pontos = c("P1","P2"), add_trend = TRUE, facet = TRUE)

## End(Not run)
```

param_plot_multi	<i>Plot temporal para varios parametros (filtro por rio/ponto)</i>
------------------	--

Description

Combina varios parametros em um unico grafico. Por padrao:

- cor = ponto (se existir);
- facet = "parametro" cria paineis por parametro;
- facet = "grid" usa grade ponto ~ parametro quando ha mais de um ponto.

Usage

```
param_plot_multi(
  df,
  parametros,
  rios = NULL,
  pontos = NULL,
  add_trend = TRUE,
  facet = c("parametro", "none", "grid")
)
```

Arguments

df	Data frame com data e colunas dos parametros.
parametros	Vetor de nomes de parametros.
rios	Vetor de rios para filtrar (opcional; usa coluna rio se existir).
pontos	Vetor de pontos para filtrar (opcional; usa coluna ponto se existir).
add_trend	Logical; se TRUE, adiciona reta lm em cada painel.
facet	"parametro", "none" ou "grid".

Value

Objeto ggplot.

See Also

Other parameter-tools: [param_plot\(\)](#), [param_summary\(\)](#), [param_summary_multi\(\)](#), [param_trend\(\)](#), [param_trend_multi\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
param_plot_multi(wq_demo, c("turbidez","od"), pontos = c("P1","P2"),
  add_trend = TRUE, facet = "grid")

## End(Not run)
```

param_summary

Resumo estatístico por parametro (com filtro por rio e/ou ponto)

Description

Produz resumo estatístico para **um parametro**, com opções de filtro por rios e/ou pontos e agregação opcional por período (mes/trimestre/ano), quando houver coluna data.

Usage

```
param_summary(  
  df,  
  parametro,  
  rios = NULL,  
  pontos = NULL,  
  period = c("none", "month", "quarter", "year"),  
  na_rm = TRUE  
)
```

Arguments

df	Data frame com ao menos a coluna do parametro. Idealmente contem ponto, e opcionalmente rio e data.
parametro	Character; nome do parametro (ex.: "turbidez", "od", "pH").
rios	Vetor de nomes de rio a filtrar (opcional; usa coluna rio se existir).
pontos	Vetor de pontos a filtrar (opcional; usa coluna ponto se existir).
period	"none", "month", "quarter" ou "year" para agregar por periodo (requer coluna data).
na_rm	Remover NA dos calculos? (default TRUE)

Value

Tibble com colunas de agrupamento disponiveis (rio, ponto, periodo) e metricas: n, mean, sd, min, median, max.

See Also

Other parameter-tools: [param_plot\(\)](#), [param_plot_multi\(\)](#), [param_summary_multi\(\)](#), [param_trend\(\)](#), [param_trend_multi\(\)](#)

Examples

```
## Not run:  
data("wq_demo", package = "tikatuwq")  
param_summary(wq_demo, "turbidez", pontos = "P1")  
param_summary(wq_demo, "od", rios = "Rio Azul", period = "month")  
  
## End(Not run)
```

param_summary_multi	<i>Resumo para varios parametros (filtro por rio/ponto)</i>
---------------------	---

Description

Itera sobre um vetor de parametros, chamando `param_summary()` para cada um, e combina as saidas em uma unica tabela, acrescentando a coluna parametro.

Usage

```
param_summary_multi(
  df,
  parametros,
  rios = NULL,
  pontos = NULL,
  period = c("none", "month", "quarter", "year"),
  na_rm = TRUE
)
```

Arguments

df	Data frame com colunas necessarias (ver <code>param_summary()</code>).
parametros	Vetor de nomes de parametros (ex.: <code>c("turbidez", "od", "pH")</code>).
rios	Vetor de rios para filtrar (opcional; usa coluna rio se existir).
pontos	Vetor de pontos para filtrar (opcional; usa coluna ponto se existir).
period	"none", "month", "quarter", "year" (igual a <code>param_summary()</code>).
na_rm	Logical; repassado para <code>param_summary()</code> .

Value

Tibble combinando os resumos de todos os parametros, com coluna parametro indicando a origem.

See Also

Other parameter-tools: [param_plot\(\)](#), [param_plot_multi\(\)](#), [param_summary\(\)](#), [param_trend\(\)](#), [param_trend_multi\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
param_summary_multi(wq_demo, c("turbidez", "od"), pontos = c("P1", "P2"))

## End(Not run)
```

param_trend	<i>Tendencia temporal por parametro (por rio/ponto se existentes)</i>
-------------	---

Description

Ajusta um modelo **lm(valor ~ tempo)** para **um parametro**, retornando slope, p_value, r2 e n. Se existirem colunas rio e/ou ponto, calcula por grupo; caso contrario, calcula geral.

Usage

```
param_trend(df, parametro, rios = NULL, pontos = NULL, na_rm = TRUE)
```

Arguments

df	Data frame com data e a coluna do parametro. Idealmente contem ponto, e opcionalmente rio.
parametro	Character; nome do parametro.
rios	Vetor de nomes de rio a filtrar (opcional; usa coluna rio se existir).
pontos	Vetor de pontos a filtrar (opcional; usa coluna ponto se existir).
na_rm	Remover NA antes do ajuste? (default TRUE)

Value

Tibble com colunas de agrupamento (quando existirem) + slope (por dia), p_value, r2, n.

See Also

Other parameter-tools: [param_plot\(\)](#), [param_plot_multi\(\)](#), [param_summary\(\)](#), [param_summary_multi\(\)](#), [param_trend_multi\(\)](#)

Examples

```
## Not run:  
data("wq_demo", package = "tikatuwq")  
param_trend(wq_demo, "turbidez", pontos = c("P1", "P2"))  
  
## End(Not run)
```

param_trend_multi	<i>Tendencia para varios parametros (filtro por rio/ponto)</i>
-------------------	--

Description

Itera sobre um vetor de parametros, chamando `param_trend()` para cada um, e combina as saidas em uma unica tabela, acrescentando a coluna `parametro`.

Usage

```
param_trend_multi(df, parametros, rios = NULL, pontos = NULL, na_rm = TRUE)
```

Arguments

<code>df</code>	Data frame com data e colunas dos parametros.
<code>parametros</code>	Vetor de nomes de parametros.
<code>rios</code>	Vetor de rios (opcional; usa coluna rio se existir).
<code>pontos</code>	Vetor de pontos (opcional; usa coluna ponto se existir).
<code>na_rm</code>	Logical; repassado para <code>param_trend()</code> .

Value

Tibble combinando as tendencias de todos os parametros, com coluna `parametro`.

See Also

Other parameter-tools: [param_plot\(\)](#), [param_plot_multi\(\)](#), [param_summary\(\)](#), [param_summary_multi\(\)](#), [param_trend\(\)](#)

Examples

```
## Not run:
data("wq_demo", package = "tikatuwq")
param_trend_multi(wq_demo, c("turbidez","od"), pontos = "P1")

## End(Not run)
```

plot_box	<i>Boxplots by site/parameter</i>
----------	-----------------------------------

Description

Boxplots of one numeric parameter grouped by a categorical column.

Usage

```
plot_box(df, parametro, by = "ponto")
```

Arguments

df	Data frame with water quality data.
parametro	Character; name of the numeric parameter column.
by	Character; grouping column (e.g., "ponto").

Value

A ggplot object.

See Also

[plot_series\(\)](#), [plot_heatmap\(\)](#), [iqa\(\)](#)

Examples

```
data(wq_demo)
plot_box(wq_demo, "turbidez", by = "ponto")
```

plot_heatmap	<i>Heatmap of parameters vs. sites</i>
--------------	--

Description

Heatmap for long-format data (date x parameter).

Usage

```
plot_heatmap(df_long)
```

Arguments

df_long	Long-format data frame with columns data, parametro, valor.
---------	---

Value

A ggplot object.

Examples

```
# Example: reshape wq_demo to long and plot
data(wq_demo)
library(tidyr)
df_long <- tidyr::pivot_longer(
  wq_demo,
  cols = c("ph", "od", "turbidez", "dbo"),
  names_to = "parametro",
  values_to = "valor"
)
plot_heatmap(df_long)
```

plot_iqa

Plot IQA by site/date

Description

Bar plot of IQA values per site/date. Requires an IQA column.

Usage

```
plot_iqa(df)
```

Arguments

df Data frame returned by iqa() (or with equivalent columns).

Value

A ggplot object.

See Also

[iqa\(\)](#), [plot_series\(\)](#), [plot_box\(\)](#)

Examples

```
data(wq_demo)
d <- iqa(wq_demo, na_rm = TRUE)
plot_iqa(d)
```

plot_map

Mapa interativo de pontos de coleta

Description

Gera um mapa Leaflet com os pontos de coleta que possuam coordenadas de latitude e longitude validas. Mostra informacoes adicionais no popup.

Usage

```
plot_map(  
  data,  
  popup_cols = NULL,  
  cluster = TRUE,  
  color_by = NULL,  
  tiles = "OpenStreetMap"  
)
```

Arguments

data	Um data.frame contendo as colunas de coordenadas. Sao aceitos nomes "latitude"/"longitude" ou "lat"/"lon".
popup_cols	Vetor de colunas a exibir no popup (ex.: c("rio","ponto","data","iqa")). Se NULL, usa colunas comuns se existirem.
cluster	Agrupar marcadores proximos (TRUE/FALSE). Default = TRUE.
color_by	Nome de coluna para colorir os pontos (opcional). Se for "iqa", aplica classes de qualidade da agua.
tiles	Provedor de tiles (default = "OpenStreetMap").

Value

Objeto htmlwidget (mapa Leaflet).

Examples

```
df <- data.frame(  
  rio = c("Buranhem", "Chamagunga"),  
  ponto = c("P1", "P2"),  
  data = as.Date(c("2025-09-20", "2025-09-21")),  
  latitude = c(-16.435, -16.498),  
  longitude = c(-39.062, -39.080),  
  iqa = c(72, 58)  
)  
plot_map(df, popup_cols = c("rio", "ponto", "data", "iqa"), color_by = "iqa")
```

plot_series	<i>Time series by parameter</i>
-------------	---------------------------------

Description

Plot a time series for one numeric parameter, optionally colored/faceted by a grouping column.

Usage

```
plot_series(df, parametro, facet = NULL)
```

Arguments

df	Data frame with a data column (Date/POSIXct) and the parameter column.
parametro	Character; name of the numeric column to plot on Y.
facet	Character or NULL; optional grouping column name to color/facet.

Value

A ggplot object.

See Also

[plot_box\(\)](#), [plot_heatmap\(\)](#), [iqa\(\)](#)

Examples

```
data(wq_demo)
# Basic: time series of turbidity
p <- plot_series(wq_demo, "turbidez")
# With color/facet by sampling point
p2 <- plot_series(wq_demo, "turbidez", facet = "ponto")
```

plot_trend	<i>Linha de tendencia temporal para parametros de qualidade da agua</i>
------------	---

Description

Gera um grafico de series temporais com pontos observados e linhas de tendencia ajustadas. Suporta metodos robustos (Theil-Sen), lineares (OLS) ou suavizados (LOESS). Util para verificar tendencias de parametros ambientais por ponto e/ou rio.

Usage

```
plot_trend(  
  data,  
  param,  
  date_col = "data",  
  group_cols = c("rio", "ponto"),  
  method = c("theilsen", "ols", "loess"),  
  show_points = TRUE,  
  min_n = 6  
)
```

Arguments

data	data.frame. Deve conter ao menos uma coluna de datas e a coluna do parametro a ser analisado.
param	character. Nome da coluna do parametro (ex.: "turbidez", "iqa").
date_col	character. Nome da coluna de datas. Default = "data".
group_cols	character. Vetor com colunas para agrupamento (ex.: c("rio","ponto")). Use "none" para nao facetar. Default = c("rio","ponto").
method	character. Metodo de ajuste da tendencia: • "theilsen" (padrao): regressao Theil-Sen (robusta a outliers). • "ols": regressao linear simples (minimos quadrados). • "loess": curva suavizada, sem inclinacao unica.
show_points	logical. Mostrar pontos observados? Default = TRUE.
min_n	integer. Numero minimo de observacoes por grupo para calcular tendencia. Default = 6.

Details

- A funcao desenha pontos e linhas conectando as observacoes, alem da linha de tendencia calculada pelo metodo escolhido.
- Quando group_cols possui mais de uma categoria, os grupos sao facetados.
- "theilsen" e mais robusto a valores atipicos do que "ols".
- "loess" e util quando nao se espera relacao linear no tempo.

Value

Objeto ggplot2, que pode ser plotado diretamente.

See Also

[plot_series\(\)](#), [iqa\(\)](#)

Examples

```
# Exemplo simples: turbidez com tendencia Theil-Sen
set.seed(1)
df <- data.frame(
  data = as.Date("2024-01-01") + 0:11*30,
  rio = "Demo", ponto = "P1",
  turbidez = 20 + (-0.3)*(0:11) + rnorm(12, 0, 1)
)
plot_trend(df, param = "turbidez", method = "theilsen")

# Exemplo com multiplos grupos e facetamento (OLS)
df2 <- data.frame(
  data = rep(seq(as.Date("2024-01-01"), by = "30 days", length.out = 12), 2),
  rio = rep(c("Rio A", "Rio B"), each = 12),
  ponto = rep(c("P1", "P2"), each = 12),
  od = c(7 + rnorm(12, 0, 0.5), 6 + rnorm(12, 0, 0.5))
)
plot_trend(df2, param = "od", method = "ols")
```

read_wq

Read water-quality CSV (robust parsing)

Description

Reads a CSV file with **comma or semicolon delimiter** and **comma or dot** as decimal mark, ignoring unit suffixes (e.g., "0,04 mg/L"). Everything is read as text first, column names are normalized, and likely numeric columns are parsed robustly. A conservative safeguard adjusts obviously out-of-range pH values (e.g., 72 -> 7.2).

Usage

```
read_wq(path, tz = "America/Bahia")
```

Arguments

path	Path to the CSV file.
tz	Time zone for dates (kept for compatibility; dates are Date).

Value

A tibble with:

- normalized, lowercase column names (spaces to _, non-alnum removed);
- numeric columns parsed ignoring unit strings;
- data parsed to Date (tries ymd then dmy);
- ponto coerced to character (if present).

Parsed numeric candidates

```
c("ph","od","turbidez","dbo","coliiformes","p_total","ptotal","fosforo_total","temperatura","ec","co
```

See Also

```
clean_units(), validate_wq(), conama_check(), iqa()
```

Examples

```
## Not run:
# Minimal example (write a small CSV and read it):
tmp <- tempfile(fileext = ".csv")
writeLines(
  c("ponto;data;ph;od;turbidez",
    "R1_01;2025-01-20;7,2;6,8;5,1",
    "R1_01;21/01/2025;7.1;7.0;4.8 mg/L"),
  tmp
)
x <- read_wq(tmp)
str(x)

## End(Not run)
```

render_report

*Render a water-quality report from the internal R Markdown template***Description**

Renders an HTML report using the package's internal R Markdown template. By default, the output is written to a **temporary directory** to comply with CRAN policies. The function returns (invisibly) the full path to the generated file.

Usage

```
render_report(
  df,
  meta = list(river = NA, period = NA),
  output_file = "wq_report.html",
  output_dir = tempdir(),
  template = system.file("templates", "report_rmd.Rmd", package = "tikatuwq")
)
```

Arguments

df	Data frame with the input data used by the template.
meta	Named list with contextual metadata (e.g., river, period).
output_file	File name for the report (default "wq_report.html").

output_dir	Directory where the file will be written (default tempdir()). It will be created if it does not exist.
template	Path to the internal template file. Defaults to the package Rmd template shipped under inst/templates/report_rmd.Rmd.

Details

The template expects a data frame with columns compatible with the package (e.g., ponto, data, parameters used by IQA/CONAMA checks). You can pass optional metadata via meta, such as river and period.

This function relies on **rmarkdown** (listed in Suggests). If the package is not available, an informative error is thrown.

Value

Invisible character string: the absolute path to the generated report.

Notes

- The default output directory is tempdir() to avoid writing into the user's project folder during examples or automated checks.
- The template is an **Rmd** (R Markdown). If you prefer Quarto, provide a custom template path to a .qmd and ensure your environment supports it.

See Also

rmarkdown::render()

Examples

```
# Minimal example (writes to a temporary directory)
d <- wq_demo
path <- render_report(d, meta = list(river = "Example River", period = "Jan-Feb"))
file.exists(path)
```

resume_wq

Descriptive summaries by group

Description

Computes basic descriptive statistics (mean, median, sd) for all **numeric** columns in df, grouped by one or more keys.

Usage

```
resume_wq(df, by = c("ponto", "mes"), funs = c("mean", "median", "sd"))
```

Arguments

df	A data frame or tibble.
by	Character vector with grouping column names (default <code>c("ponto", "mes")</code>). Any names not present in df are ignored.
funcs	Deprecated (kept for compatibility; ignored). The function always computes mean, median and sd with <code>na.rm = TRUE</code> .

Details

- Grouping columns not found in df are silently dropped.
- If no grouping columns remain, an error is thrown.
- Only numeric columns are summarized; if none exist, an error is thrown.
- Missing values are ignored (`na.rm = TRUE`).

Value

A tibble with the grouping keys and one column per statistic/variable, named as `{var}_{stat}` (e.g., `od_mean`, `od_median`, `od_sd`).

See Also

`dplyr::summarise()`, `dplyr::across()`

Examples

```
# Using the demo dataset shipped with the package
d <- wq_demo
# Example: group by point (ponto)
s1 <- resume_wq(d, by = "ponto")
head(s1)

# Example: group by point and month (if 'mes' exists in your data)
# s2 <- resume_wq(d, by = c("ponto", "mes"))
```

trend_param

Tendencia monotona por parametro e ponto (Theil-Sen + Spearman)

Description

Calcula a inclinacao de Theil-Sen (robusta) e o p-valor do teste de correlacao de Spearman entre tempo e o valor do parametro. Retorna estatisticas por grupo (ex.: rio, ponto).

Usage

```
trend_param(  
  data,  
  param,  
  date_col = "data",  
  group_cols = c("rio", "ponto"),  
  min_n = 6,  
  alpha = 0.05  
)
```

Arguments

data	data.frame com pelo menos uma coluna de data e a coluna do parametro.
param	nome do parametro (string), por exemplo "turbidez" ou "iqa".
date_col	nome da coluna de datas. Default: "data".
group_cols	vetor de nomes para agrupar. Default: c("rio","ponto").
min_n	amostra minima por grupo. Default: 6.
alpha	nivel de significancia para classificar tendencia. Default: 0.05.

Value

data.frame com colunas por grupo e: n, date_min, date_max, days_span, slope_per_year, intercept, rho_spearman, p_value, trend ("aumento" / "queda" / "estavel"), pct_change_period (aprox. % no periodo observado).

Examples

```
set.seed(1)  
df <- data.frame(  
  data = as.Date("2024-01-01") + 0:11*30,  
  rio = "Demo", ponto = "P1",  
  turbidez = 20 + (-0.3)*(0:11) + rnorm(12, 0, 1)  
)  
trend_param(df, param = "turbidez")
```

validate_wq	<i>Validate presence of required columns</i>
-------------	--

Description

Ensures a minimal set of columns exists in the dataset; otherwise throws an error listing the missing names.

Usage

```
validate_wq(  
  df,  
  required = c("ph", "turbidez", "od", "dbo", "nt_total", "p_total", "tds",  
               "temperatura", "coliiformes")  
)
```

Arguments

df	Input data frame / tibble.
required	Character vector of required column names.

Value

The input df if valid; otherwise, an error is thrown.

See Also

[read_wq\(\)](#), [conama_check\(\)](#)

Examples

```
df_ex <- data.frame(  
  ph = 7, turbidez = 2, od = 7, dbo = 3,  
  nt_total = 0.8, p_total = 0.05, tds = 150,  
  temperatura = 24, coliiformes = 200  
)  
validate_wq(df_ex)
```

wq_demo

Demo water quality dataset

Description

A tiny example dataset used in examples and vignettes. Column names follow the package's Portuguese conventions (e.g., ponto, data, turbidez).

Usage

```
data(wq_demo)
```

Format

A data frame (tibble) with 20 rows and 11 columns:

ponto character, monitoring point id
data Date, sampling date
ph numeric, pH
od numeric, dissolved oxygen (mg/L)
turbidez numeric, NTU
dbo numeric, mg/L
coliformes integer, MPN/100 mL
p_total numeric, total phosphorus (mg/L)
nt_total numeric, total nitrogen (mg/L)
temperatura numeric, degrees Celsius
tds numeric, total dissolved solids (mg/L)

Details

The dataset is simulated for illustrative purposes and is suitable for quick examples of `iqa()`, `conama_check()`, and plotting helpers.

Source

Simulated for package examples.

See Also

[iqa\(\)](#), [conama_check\(\)](#), [plot_series\(\)](#), [plot_box\(\)](#), [plot_iqa\(\)](#), [plot_heatmap\(\)](#)

Examples

```
data("wq_demo", package = "tikatuwq")
head(wq_demo)
# quick IQA example:
# iqa(wq_demo, na_rm = TRUE)
```


Index

- * **datasets**
 - wq_demo, [31](#)
- * **parameter-tools**
 - param_plot, [14](#)
 - param_plot_multi, [15](#)
 - param_summary, [16](#)
 - param_summary_multi, [18](#)
 - param_trend, [19](#)
 - param_trend_multi, [20](#)
- * **reporting-tools**
 - generate_analysis, [8](#)
- * **reporting**
 - render_report, [27](#)
- clean_units, [3](#)
- clean_units(), [27](#)
- conama_check, [3](#)
- conama_check(), [6](#), [8](#), [10](#), [11](#), [27](#), [31](#), [32](#)
- conama_limits, [4](#)
- conama_limits(), [4](#)
- conama_report, [5](#)
- conama_report(), [4](#), [6](#), [7](#)
- conama_summary, [6](#)
- conama_summary(), [4](#), [5](#), [7](#)
- conama_text, [7](#)
- conama_text(), [4–6](#)
- dplyr::across(), [29](#)
- dplyr::summarise(), [29](#)
- generate_analysis, [8](#)
- iet_carlson, [9](#)
- iet_carlson(), [11](#)
- iet_lamparelli, [10](#)
- iet_lamparelli(), [10](#)
- iqa, [12](#)
- iqa(), [8](#), [10](#), [11](#), [21](#), [22](#), [24](#), [25](#), [27](#), [32](#)
- nsfwqi, [13](#)
- param_plot, [14](#), [16–20](#)
- param_plot_multi, [15](#), [15](#), [17–20](#)
- param_summary, [15](#), [16](#), [16](#), [18–20](#)
- param_summary_multi, [15–17](#), [18](#), [19](#), [20](#)
- param_trend, [15–18](#), [19](#), [20](#)
- param_trend_multi, [15–19](#), [20](#)
- plot_box, [21](#)
- plot_box(), [22](#), [24](#), [32](#)
- plot_heatmap, [21](#)
- plot_heatmap(), [21](#), [24](#), [32](#)
- plot_iqa, [22](#)
- plot_iqa(), [32](#)
- plot_map, [23](#)
- plot_series, [24](#)
- plot_series(), [21](#), [22](#), [25](#), [32](#)
- plot_trend, [24](#)
- read_wq, [26](#)
- read_wq(), [3](#), [31](#)
- render_report, [27](#)
- resume_wq, [28](#)
- trend_param, [29](#)
- validate_wq, [30](#)
- validate_wq(), [27](#)
- wq_demo, [31](#)