Package: statAfrikR (via r-universe)

May 30, 2026

Title Statistical Tools for African National Statistics Institutes

Version 0.2.0

Description A comprehensive statistical toolbox for National Statistics Institutes (INS) in Africa. Provides functions for survey data import ('KoboToolbox', 'ODK', 'CSPro', 'Excel', 'Stata', 'SPSS'), data processing and validation, weighted statistical analysis (descriptive statistics, cross-tabulations, regression, Human Development Index (HDI), Multidimensional Poverty Index (MPI) following Alkire and Foster (2011) <doi:10.1093/oep/gpr051>, inequalities), visualization (age pyramids, thematic maps, official charts) and dissemination ('SDMX' export, 'DDI' metadata, anonymization, Word/PDF reports). Designed to work in resource-constrained environments, offline and in French.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Language fr

URL https://github.com/damoko2004/statAfrikR

BugReports https://github.com/damoko2004/statAfrikR/issues

Depends R (>= 4.2.0)

Imports dplyr (>= 1.1.0), forcats (>= 1.0.0), ggplot2 (>= 3.4.0), ggrepel (>= 0.9.0), haven (>= 2.5.0), readr (>= 2.1.0), readxl (>= 1.4.0), rlang (>= 1.1.0), scales (>= 1.2.0), sf (>= 1.0.0), stringr (>= 1.5.0), survey (>= 4.1.0), tibble (>= 3.1.0), tidyr (>= 1.3.0)

Suggests flextable (>= 0.9.0), httr2 (>= 0.2.0), jsonlite (>= 1.8.0), officer (>= 0.6.0), openxlsx2 (>= 0.8.0), srvyr (>= 1.1.0), testthat (>= 3.0.0), rmarkdown (>= 2.20), withr (>= 2.5.0), knitr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Dikers Amoko [aut, cre]

Maintainer Dikers Amoko <diamoko@gmail.com>

LazyData true

LazyDataCompression xz

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://damoko2004.r-universe.dev

Date/Publication 2026-05-30 22:50:06 UTC

RemoteUrl https://github.com/damoko2004/statafrikr

RemoteRef HEAD

RemoteSha f9144c1b302fdbc6b7df1ea674ccba30c1636929

Contents

`analyse_regression` *Analyse de régression*

Description

Ajuste un modèle de régression linéaire, logistique ou de Poisson avec prise en compte optionnelle du plan de sondage complexe. Produit un tableau de résultats formaté avec OR/RR si approprié.

Usage

```
analyse_regression(
  formule,
  data,
  type = c("lineaire", "logistique", "poisson"),
  niveau_confiance = 0.95,
  format_sortie = c("tibble", "liste", "flextable")
)
```

Arguments

`formule`	formula — Formule du modèle (ex: `revenu ~ age + sexe`)
`data`	data.frame, tibble ou objet `svydesign` — Données
`type`	character — Type de modèle : `"lineaire"`, `"logistique"`, `"poisson"`. Défaut : "lineaire".
`niveau_confiance`	numeric — Niveau de confiance pour les IC. Défaut : 0.95.
`format_sortie`	character — `"liste"`, `"tibble"` ou `"flextable"`. Défaut : "tibble".

Value

Selon format_sortie : liste complète, tibble ou flextable des coefficients avec IC et p-valeurs.

Examples

```
## Not run:
  donnees <- data.frame(
    revenu = rnorm(100, 200000, 50000),
    age    = sample(20:65, 100, replace=TRUE),
    sexe   = sample(c("H","F"), 100, replace=TRUE)
  )
  analyse_regression(revenu ~ age + sexe, donnees)

## End(Not run)
```

`analyse_spatiale` *Analyse spatiale — jointure et indicateurs par zone*

Description

Joint un jeu de données statistiques avec un shapefile géographique et calcule des indicateurs par aire géographique. Produit un objet sf enrichi prêt pour la cartographie.

Usage

```
analyse_spatiale(
  data,
  shapefile,
  var_geo_data,
  var_geo_shape,
  indicateurs = NULL,
  fonctions = list(moyenne = function(x) mean(x, na.rm = TRUE), n = function(x)
    sum(!is.na(x)))
)
```

Arguments

`data`	data.frame ou tibble — Données avec variable géographique
`shapefile`	sf ou character — Objet sf ou chemin vers un fichier shapefile (.shp, .gpkg, .geojson)
`var_geo_data`	character — Variable géographique dans `data`
`var_geo_shape`	character — Variable géographique dans le shapefile
`indicateurs`	character ou NULL — Variables à agréger par zone. Si NULL, toutes les variables numériques. Défaut : NULL.
`fonctions`	list — Fonctions d'agrégation nommées. Défaut : `list(moyenne = mean, n = length)`.

Value

Un objet `sf` avec les indicateurs calculés par zone.

Examples

```
## Not run:
  carte <- analyse_spatiale(
    data          = donnees_enquete,
    shapefile     = "data/shapefiles/regions.shp",
    var_geo_data  = "region",
    var_geo_shape = "NOM_REGION",
    indicateurs   = c("taux_pauvrete", "revenu_moyen")
  )

## End(Not run)
```

`anonymiser_donnees` *Anonymiser un jeu de données*

Description

Applique les techniques d'anonymisation conformes aux standards IHSN/PARIS21 : suppression, masquage, generalisation, perturbation et pseudonymisation des variables sensibles.

Usage

```
anonymiser_donnees(
  data,
  vars_supprimer = NULL,
  vars_masquer = NULL,
  vars_perturber = NULL,
  vars_generaliser = NULL,
  niveau_bruit = 0.05,
  graine = 42L,
  rapport = TRUE
)
```

Arguments

`data`	data.frame ou tibble — Données à anonymiser
`vars_supprimer`	character ou NULL — Variables à supprimer entièrement. Défaut : NULL.
`vars_masquer`	character ou NULL — Variables à remplacer par des codes anonymes. Défaut : NULL.
`vars_perturber`	character ou NULL — Variables numériques à perturber par bruit aléatoire. Défaut : NULL.
`vars_generaliser`	list ou NULL — Liste nommée de variables à généraliser avec les bornes : `list(age = 5, revenu = 10000)`. Défaut : NULL.
`niveau_bruit`	numeric — Niveau de bruit pour la perturbation (proportion de l'écart-type). Défaut : 0.05.
`graine`	integer — Graine aléatoire. Défaut : 42.
`rapport`	logical — Produire un rapport d'anonymisation. Défaut : TRUE.

Value

Si `rapport = FALSE` : tibble anonymisé. Si `rapport = TRUE` : liste avec `$donnees` et `$rapport`.

Examples

```
## Not run:
  resultat <- anonymiser_donnees(
    donnees_enquete,
    vars_supprimer  = c("nom", "prenom", "telephone"),
    vars_masquer    = c("id_menage", "id_individu"),
    vars_perturber  = c("revenu_mensuel"),
    vars_generaliser = list(age = 5)
  )
  donnees_anon <- resultat$donnees

## End(Not run)
```

---

appliquer_ponderations

Appliquer les pondérations d'enquête

Description

Crée un objet de plan de sondage complexe à partir d'un tibble et des variables de pondération, strates et grappes. Enveloppe ergonomique autour de `survey::svydesign()` avec validation complète des poids et messages d'erreur en français.

Usage

```
appliquer_ponderations(
  data,
  var_poids,
  var_strate = NULL,
  var_grappe = NULL,
  var_fpc = NULL,
  normaliser = FALSE
)
```

Arguments

`data`	data.frame ou tibble — Données de l'enquête
`var_poids`	character — Nom de la variable de pondération finale
`var_strate`	character ou NULL — Variable de stratification. Défaut : NULL.
`var_grappe`	character ou NULL — Variable d'unité primaire de sondage (UPS/cluster). Défaut : NULL.
`var_fpc`	character ou NULL — Variable de correction pour population finie (FPC). Défaut : NULL.
`normaliser`	logical — Normaliser les poids pour que leur somme soit égale à l'effectif de l'échantillon. Défaut : FALSE.

Value

Un objet svydesign du package survey.

See Also

tab_croisee, stat_descr

Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  donnees <- data.frame(
    revenu      = rnorm(50, 200000, 50000),
    poids_final = runif(50, 0.5, 3),
    strate      = sample(1:4, 50, replace = TRUE),
    grappe_id   = sample(1:10, 50, replace = TRUE)
  )
  appliquer_ponderations(donnees,
    var_poids = "poids_final",
    var_strate = "strate",
    var_grappe = "grappe_id")
}
```

---

calcul_fgt *Calcul des indices de pauvrete FGT*

Description

Calcule les indices Foster-Greer-Thorbecke (FGT0, FGT1, FGT2) avec prise en compte optionnelle du plan de sondage complexe. Les trois indices mesurent respectivement l'incidence, la profondeur et la severite de la pauvrete monetaire.

Usage

```
calcul_fgt(
  data,
  var_depense,
  seuil_pauvrete,
  poids = NULL,
  strate = NULL,
  grappe = NULL,
  sous_groupes = NULL,
  alpha = c(0, 1, 2),
  ic = TRUE,
  na.rm = TRUE
)
```

Arguments

`data`	data.frame, tibble ou objet `svydesign` – Donnees source
`var_depense`	character – Nom de la variable de depenses ou revenus par tete (en monnaie locale, strictement positive)
`seuil_pauvrete`	numeric – Seuil de pauvrete en monnaie locale. Meme unite que `var_depense`
`poids`	character ou NULL – Nom de la variable de ponderation. Ignore si `data` est un `svydesign`. Defaut : NULL
`strate`	character ou NULL – Variable de stratification. Defaut : NULL
`grappe`	character ou NULL – Variable d'identifiant de grappe. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de decomposition (region, milieu, sexe). Defaut : NULL
`alpha`	numeric – Parametre de sensibilite : 0, 1, 2 ou vecteur. Defaut : `c(0, 1, 2)`
`ic`	logical – Calculer les intervalles de confiance a 95%. Defaut : TRUE
`na.rm`	logical – Exclure les valeurs manquantes. Defaut : TRUE

Value

Un objet de classe `saf_fgt`

References

Foster, J., Greer, J., & Thorbecke, E. (1984). A class of decomposable poverty measures. Econometrica, 52(3), 761-766. doi:10.2307/1913475

Examples

```
set.seed(42)
menages <- data.frame(
  depense_pc = c(rexp(70, rate = 1/150000), rexp(30, rate = 1/400000)),
  poids      = runif(100, 0.8, 1.2),
  region     = sample(c("Bangui", "Ombella", "Lobaye"), 100, TRUE),
  milieu     = sample(c("urbain", "rural"), 100, TRUE, prob = c(0.4, 0.6))
)
fgt <- calcul_fgt(menages, var_depense = "depense_pc",
                  seuil_pauvrete = 220000, poids = "poids")
print(fgt)
```

`calcul_idh` *Calculer l'Indice de Développement Humain (IDH)*

Description

Calcule l'IDH et ses trois dimensions (santé, éducation, revenu) selon la méthodologie officielle PNUD post-2010. Applicable au niveau national ou infranational.

Usage

```
calcul_idh(
  esperance_vie,
  annees_scol_moy,
  annees_scol_att,
  rnb_habitant,
  niveau = c("national", "infranational"),
  annee = NULL
)
```

Arguments

`esperance_vie`	numeric — Espérance de vie à la naissance (années)
`annees_scol_moy`	numeric — Durée moyenne de scolarisation (années)
`annees_scol_att`	numeric — Durée attendue de scolarisation (années)
`rnb_habitant`	numeric — RNB par habitant en PPA (USD constants 2017)
`niveau`	character — `"national"` ou `"infranational"`. Défaut : "national".
`annee`	integer ou NULL — Année de référence. Défaut : NULL.

Value

Une liste avec : `idh`, `indice_sante`, `indice_education`, `indice_revenu`, `categorie`.

References

PNUD (2023). Technical Notes: Calculating the Human Development Indices.

Examples

```
idh <- calcul_idh(
  esperance_vie   = 61.2,
  annees_scol_moy = 5.4,
  annees_scol_att = 9.8,
  rnb_habitant    = 2350,
  annee           = 2023
)
cat("IDH :", idh$idh)
```

`calcul_ipm` *Calculer l'Indice de Pauvreté Multidimensionnelle (IPM)*

Description

Calcule l'IPM selon la méthodologie OPHI/PNUD (Alkire-Foster). Supporte les dimensions standard (santé, éducation, niveau de vie) et des dimensions personnalisées.

Usage

```
calcul_ipm(
  data,
  indicateurs,
  poids_dimensions = NULL,
  seuil_pauvrete = 1/3,
  var_poids = NULL
)
```

Arguments

`data`	data.frame ou tibble — Données individuelles ou ménages
`indicateurs`	list — Liste nommée des indicateurs par dimension. Chaque élément est un vecteur de noms de variables (0/1 : 1 = privation). Ex: `list(sante = c("malnutrition", "mortalite_enfant"), ...)`
`poids_dimensions`	numeric ou NULL — Poids de chaque dimension (doit sommer à 1). Si NULL, poids égaux. Défaut : NULL.
`seuil_pauvrete`	numeric — Seuil de privation pour être considéré multidimensionnellement pauvre (entre 0 et 1). Défaut : 1/3.
`var_poids`	character ou NULL — Variable de pondération. Défaut : NULL.

Value

Une liste avec : `ipm`, `H` (incidence), `A` (intensité), `contributions` par dimension, `donnees_enrichies`.

References

Alkire, S. & Foster, J. (2011). Counting and multidimensional poverty measurement. Journal of Public Economics, 95(7-8), 476-487.

Examples

```
## Not run:
  donnees <- data.frame(
    malnutrition         = sample(0:1, 50, replace = TRUE),
    mortalite_enfant     = sample(0:1, 50, replace = TRUE),
    annees_scolarisation = sample(0:1, 50, replace = TRUE),
```

```
  enfants_scolarises   = sample(0:1, 50, replace = TRUE),
  electricite          = sample(0:1, 50, replace = TRUE),
  eau_potable          = sample(0:1, 50, replace = TRUE)
)
indicateurs <- list(
  sante      = c("malnutrition", "mortalite_enfant"),
  education  = c("annees_scolarisation", "enfants_scolarises"),
  niveau_vie = c("electricite", "eau_potable")
)
calcul_ipm(donnees, indicateurs)

## End(Not run)
```

`carte_choroplethe` *Carte choroplethe statistique institutionnelle*

Description

Produit une carte choroplethe professionnelle. Utilise uniquement ggplot2 (deja installe avec statAfrikR) et sf. Gestion automatique des labels pour les petits pays.

Usage

```
carte_choroplethe(
  sf_obj,
  var,
  titre = NULL,
  sous_titre = NULL,
  legende = NULL,
  palette = c("pauvrete", "developpement", "eau", "neutre"),
  n_classes = 5L,
  methode = c("quantile", "jenks", "egal", "sd"),
  col_label = NULL,
  inverser = FALSE,
  source = NULL
)
```

Arguments

`sf_obj`	sf – Objet sf enrichi (depuis `carte_joindre()`)
`var`	character – Variable numerique a cartographier
`titre`	character ou NULL – Titre. Defaut : NULL
`sous_titre`	character ou NULL – Sous-titre. Defaut : NULL
`legende`	character ou NULL – Titre de la legende. Defaut : NULL
`palette`	character – Palette : `"pauvrete"` (jaune-rouge), `"developpement"` (rouge-vert), `"eau"` (bleu clair-fonce), `"neutre"` (gris-bleu). Defaut : "pauvrete"

n_classes	integer – Nombre de classes (2-9). Defaut : 5L
methode	character – Discretisation : "quantile", "jenks", "egal", "sd". Defaut : "quantile"
col_label	character ou NULL – Variable a afficher comme label sur chaque zone. Defaut : NULL
inverser	logical – Inverser la palette. Defaut : FALSE
source	character ou NULL – Note de source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
rca <- carte_zones("rca")
n <- nrow(rca)
stats <- data.frame(
  prefecture    = rca$prefecture,
  taux_pauvrete = c(74.2, 71.8, 68.5, 65.3, 72.1,
                    55.4, 48.7, 51.2, 62.3, 58.9,
                    28.4, 42.1, 52.8, 63.7, 59.4,
                    44.6, 70.5)[seq_len(n)]
)
sf_enr <- carte_joindre(rca, stats, "prefecture", "prefecture")
carte_choroplethe(sf_enr, "taux_pauvrete",
                  titre = "Pauvrete en RCA",
                  source = "Donnees simulees")
```

---

carte_exporter *Exporter une carte en fichier image*

Description

Exporte un objet ggplot2 en PNG, PDF ou SVG.

Usage

```
carte_exporter(carte, chemin, largeur = 20, hauteur = 15, resolution = 300L)
```

Arguments

carte	ggplot – Objet ggplot2
chemin	character – Chemin de sortie (.png, .pdf, .svg)
largeur	numeric – Largeur en cm. Defaut : 20
hauteur	numeric – Hauteur en cm. Defaut : 15
resolution	integer – Resolution DPI (PNG). Defaut : 300L

Value

Chemin du fichier (invisible)

Examples

```
## Not run:
  g <- carte_choroplethe(sf_enrichi, "taux_pauvrete")
  carte_exporter(g, file.path(tempdir(), "carte.png"))

## End(Not run)
```

`carte_import` *Importer un fichier geographique*

Description

Importe un fichier geographique externe (shapefile, GeoJSON, GeoPackage) fourni par l'utilisateur ou l'INS.

Usage

```
carte_import(chemin, crs = 4326L, couche = NULL, simplifier = FALSE)
```

Arguments

`chemin`	character – Chemin vers le fichier (.shp, .geojson, .gpkg)
`crs`	integer – Code EPSG cible. Defaut : 4326 (WGS 84)
`couche`	character ou NULL – Couche GeoPackage. Defaut : NULL
`simplifier`	logical – Simplifier la geometrie. Defaut : FALSE

Value

Un objet `sf`

Examples

```
## Not run:
  regions <- carte_import("data/regions_enquete.shp")

## End(Not run)
```

carte_joindre *Joindre des donnees statistiques a un fond de carte*

Description

Joint un objet sf avec un data.frame statistique. Normalise automatiquement les cles pour eviter les erreurs de correspondance dues aux accents, casses et espaces.

Usage

```
carte_joindre(
  sf_obj,
  data,
  cle_geo,
  cle_data = NULL,
  type = c("gauche", "interne"),
  normaliser = TRUE
)
```

Arguments

sf_obj	sf – Objet geographique (depuis carte_zones() ou carte_import())
data	data.frame – Donnees statistiques
cle_geo	character – Variable cle dans sf_obj
cle_data	character – Variable cle dans data. Si NULL, utilise cle_geo. Defaut : NULL
type	character – "gauche" (toutes zones conservees) ou "interne" (zones appariees uniquement). Defaut : "gauche"
normaliser	logical – Normaliser les cles (recommande). Defaut : TRUE

Value

Un objet sf enrichi

Examples

```
# Avec les fonds de cartes integres
rca <- carte_zones("rca")
stats <- data.frame(
  prefecture    = rca$prefecture,
  taux_pauvrete = c(74.2, 71.8, 68.5, 65.3, 72.1,
                    55.4, 48.7, 51.2, 62.3, 58.9,
                    28.4, 42.1, 52.8, 63.7, 59.4,
                    44.6, 70.5)[seq_len(nrow(rca))]
)
sf_enrichi <- carte_joindre(rca, stats,
                            cle_geo  = "prefecture",
                            cle_data = "prefecture")
```

carte_pauvrete *Carte thematique de la pauvrete (FGT0)*

Description

Specialisation de carte_choroplethe() pour les indices de pauvrete. Symbologie standardisee AFRISTAT/Banque mondiale avec seuil d'alerte.

Usage

```
carte_pauvrete(
  sf_obj,
  var_fgt0,
  seuil_alerte = 0.5,
  col_label = NULL,
  titre = NULL,
  source = NULL
)
```

Arguments

sf_obj	sf – Objet sf avec taux de pauvrete
var_fgt0	character – Variable de taux de pauvrete
seuil_alerte	numeric – Seuil d'alerte. Defaut : 0.5
col_label	character ou NULL – Variable de label. Defaut : NULL
titre	character ou NULL – Titre. Defaut : titre automatique
source	character ou NULL – Source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
rca <- carte_zones("rca")
n <- nrow(rca)
stats <- data.frame(
  prefecture    = rca$prefecture,
  taux_pauvrete = c(74.2, 71.8, 68.5, 65.3, 72.1,
                    55.4, 48.7, 51.2, 62.3, 58.9,
                    28.4, 42.1, 52.8, 63.7, 59.4,
                    44.6, 70.5)[seq_len(n)]
)
sf_enr <- carte_joindre(rca, stats, "prefecture", "prefecture")
sf_enr$taux_prop <- sf_enr$taux_pauvrete / 100
carte_pauvrete(sf_enr, var_fgt0 = "taux_prop",
               source = "Donnees simulees")
```

carte_thematique *Carte thématique choroplèthe*

Description

Génère une carte choroplèthe à partir d'un objet sf enrichi ou de la jointure d'un shapefile avec des données statistiques.

Usage

```
carte_thematique(
  data_sf = NULL,
  shapefile = NULL,
  data = NULL,
  var_geo_shape = NULL,
  var_geo_data = NULL,
  var_couleur,
  titre = NULL,
  sous_titre = NULL,
  source = NULL,
  palette = c("sequentiel", "divergent"),
  n_classes = 5L,
  na_couleur = "#cccccc"
)
```

Arguments

data_sf	sf ou NULL — Objet sf avec données. Si NULL, utilise shapefile + data. Défaut : NULL.
shapefile	sf ou character ou NULL — Shapefile si data_sf est NULL. Défaut : NULL.
data	data.frame ou NULL — Données à joindre si data_sf est NULL. Défaut : NULL.
var_geo_shape	character ou NULL — Variable clé dans le shapefile. Défaut : NULL.
var_geo_data	character ou NULL — Variable clé dans les données. Défaut : NULL.
var_couleur	character — Variable à représenter par la couleur
titre	character ou NULL — Titre de la carte. Défaut : NULL.
sous_titre	character ou NULL — Sous-titre. Défaut : NULL.
source	character ou NULL — Source des données. Défaut : NULL.
palette	character — Palette de couleurs : "sequentiel", "divergent". Défaut : "sequentiel".
n_classes	integer — Nombre de classes. Défaut : 5.
na_couleur	character — Couleur pour les NA. Défaut : "#cccccc".

Value

Un objet ggplot.

Examples

```
## Not run:
  carte_thematique(
    data_sf      = regions_sf_enrichi,
    var_couleur  = "taux_pauvrete_moyenne",
    titre        = "Taux de pauvrete par region"
  )

## End(Not run)
```

`carte_zones` *Charger un fond de carte africain integre*

Description

Charge un fond de carte geographique directement integre dans statAfrikR. Aucun package supplementaire requis.

Usage

```
carte_zones(
  zone = c("afrique", "cemac", "cedeao", "eau", "sadc", "rca", "subdivisions"),
  pays = NULL
)
```

Arguments

zone	character – Zone geographique : "afrique" (54 pays), "cemac" (6 pays), "cedeao" (15 pays), "eau" (Afrique de l'Est), "sadc" (Afrique Australe), "rca" (17 prefectures RCA). Defaut : "afrique"
pays	character ou NULL – Filtrer par noms de pays (colonne pays) ou codes ISO3 (colonne iso3). Defaut : NULL

Value

Un objet sf pret a l'emploi

Examples

```
# Tous les pays africains
afrique <- carte_zones("afrique")

# Zone CEMAC uniquement
cemac <- carte_zones("cemac")
```

```
# Prefectures de la RCA
rca <- carte_zones("rca")

# Filtrer : Cameroun + Centrafrique uniquement
sous_zone <- carte_zones("afrique", pays = c("CMR", "CAF"))
```

---

check_na *Détecter les valeurs manquantes*

Description

Calcule le taux de valeurs manquantes par variable et produit un rapport de complétude. Alerte sur les variables dépassant le seuil.

Usage

```
check_na(data, seuil = 0.1, vars = NULL, alerter = TRUE)
```

Arguments

data	data.frame ou tibble — Données à analyser
seuil	numeric — Taux de NA à partir duquel une alerte est émise (entre 0 et 1). Défaut : 0.1 (10%).
vars	character ou NULL — Variables à analyser. Si NULL, toutes les variables. Défaut : NULL.
alerter	logical — Émettre des avertissements pour les variables dépassant le seuil. Défaut : TRUE.

Value

Un tibble avec les colonnes : variable, n_total, n_manquant, taux_na, statut.

Examples

```
## Not run:
  rapport_na <- check_na(donnees_enquete)
  rapport_na <- check_na(donnees_enquete, seuil = 0.05, vars = c("age", "revenu"))

## End(Not run)
```

check_types *Vérifier les types de variables*

Description

Vérifie la cohérence des types de variables par rapport aux types attendus. Détecte les problèmes courants : dates stockées en caractères, nombres stockés en texte, variables binaires incohérentes.

Usage

```
check_types(data, dictionnaire = NULL)
```

Arguments

data	data.frame ou tibble — Données à vérifier
dictionnaire	data.frame ou NULL — Dictionnaire avec colonnes nom_variable et type_attendu. Si NULL, détection automatique des problèmes courants. Défaut : NULL.

Value

Un tibble avec les anomalies détectées : variable, type_actuel, type_attendu, probleme.

Examples

```
## Not run:
  anomalies <- check_types(donnees_enquete)

## End(Not run)
```

compresser_package_diffusion *Compresser un package de diffusion*

Description

Crée une archive ZIP structurée prête à diffuser, incluant les données, la documentation, les métadonnées et les scripts de traitement. Conforme aux standards IHSN de documentation des enquêtes.

Usage

```
compresser_package_diffusion(
  donnees,
  repertoire_sortie,
  nom_package,
  inclure_csv = TRUE,
  inclure_rds = TRUE,
  fichiers_supplementaires = NULL,
  metadonnees = NULL
)
```

Arguments

donnees	data.frame ou tibble — Données à archiver
repertoire_sortie	character — Répertoire de destination de l'archive
nom_package	character — Nom de base de l'archive (sans extension)
inclure_csv	logical — Inclure les données en CSV. Défaut : TRUE.
inclure_rds	logical — Inclure les données en RDS. Défaut : TRUE.
fichiers_supplementaires	character ou NULL — Chemins vers des fichiers additionnels à inclure (rapports, scripts, etc.). Défaut : NULL.
metadonnees	list ou NULL — Métadonnées à inclure dans un fichier README automatique. Défaut : NULL.

Value

Chemin de l'archive ZIP (invisible).

Examples

```
## Not run:
  compresser_package_diffusion(
    donnees              = donnees_emop_anon,
    repertoire_sortie    = "diffusion/",
    nom_package          = "EMOP_BEN_2023_v1",
    fichiers_supplementaires = c(file.path(tempdir(), "rapport.docx"),
                                 file.path(tempdir(), "emop_ddi.xml")),
    metadonnees = list(
      titre       = "EMOP Bénin 2023",
      institution = "INSAE",
      version     = "1.0"
    )
  )

## End(Not run)
```

decomposer_fgt *Decomposer les indices FGT par sous-groupe*

Description

Decompose un indice FGT en contributions relatives de chaque sous-groupe a la pauvrete nationale.

Usage

```
decomposer_fgt(fgt_obj, variable, alpha_cible = 0)
```

Arguments

fgt_obj	objet saf_fgt – Resultat de calcul_fgt() avec sous_groupes renseigne
variable	character – Sous-groupe a decomposer
alpha_cible	numeric – Indice a decomposer : 0, 1 ou 2. Defaut : 0

Value

Tibble avec contributions absolues et relatives

Examples

```
set.seed(42)
menages <- data.frame(
  depense_pc = c(rexp(70, rate = 1/150000), rexp(30, rate = 1/400000)),
  poids  = runif(100, 0.8, 1.2),
  milieu = sample(c("urbain", "rural"), 100, TRUE, prob = c(0.4, 0.6))
)
fgt <- calcul_fgt(menages, "depense_pc", 220000,
                  poids = "poids", sous_groupes = "milieu")
decomposer_fgt(fgt, "milieu", alpha_cible = 0)
```

decomposer_inegalite *Décomposer les inégalités*

Description

Calcule les mesures d'inégalité (Gini, Theil, Atkinson) et leur décomposition inter/intra-groupe pour une variable de revenu ou de dépense.

Usage

```
decomposer_inegalite(
  data,
  var_revenu,
  var_groupe = NULL,
  var_poids = NULL,
  mesures = c("all", "gini", "theil", "atkinson")
)
```

Arguments

data	data.frame ou tibble — Données
var_revenu	character — Variable de revenu/dépense (strictement positive)
var_groupe	character ou NULL — Variable de groupe pour la décomposition. Défaut : NULL.
var_poids	character ou NULL — Variable de pondération. Défaut : NULL.
mesures	character — Mesures à calculer : "gini", "theil", "atkinson", "all". Défaut : "all".

Value

Une liste avec les mesures d'inégalité et leur décomposition.

Examples

```
## Not run:
  donnees <- data.frame(
    depense_totale = rnorm(100, 250000, 80000),
    milieu = sample(c("urbain", "rural"), 100, replace = TRUE)
  )
  decomposer_inegalite(donnees, var_revenu="depense_totale", var_groupe="milieu")

## End(Not run)
```

---

exporter_excel_ins *Export Excel institutionnel multi-feuilles*

Description

Exporte une liste de tableaux dans un classeur Excel avec feuille de sommaire.

Usage

```
exporter_excel_ins(
  tableaux,
  chemin,
  titre_classeur = "Statistiques INS",
  pays = NULL,
  annee = NULL,
  style = c("ins_standard", "minimal")
)
```

Arguments

tableaux	list – Liste nommee de data.frames a exporter
chemin	character – Chemin du fichier Excel
titre_classeur	character – Titre general. Defaut : "Statistiques INS"
pays	character ou NULL – Pays/organisation. Defaut : NULL
annee	integer ou NULL – Annee de reference. Defaut : NULL
style	character – Style : "ins_standard" ou "minimal". Defaut : "ins_standard"

Value

Chemin du fichier cree (invisible)

Examples

```
## Not run:
  tableaux <- list(
    "Descriptif" = data.frame(
      variable = c("age", "revenu"),
      n        = c(200L, 200L),
      moyenne  = c(38.2, 245000)
    ),
    "Croisement" = data.frame(
      region = c("Nord", "Sud"),
      urbain = c(45.2, 32.1),
      rural  = c(54.8, 67.9)
    )
  )
  chemin <- file.path(tempdir(), "stats_ins.xlsx")
  exporter_excel_ins(tableaux, chemin,
                     titre_classeur = "Enquete menages 2026",
                     pays = "Centrafrique", annee = 2026L)

## End(Not run)
```

exporter_graphique *Exporter un graphique en haute résolution*

Description

Exporte un objet ggplot en PNG, PDF ou SVG avec les paramètres optimaux pour publication officielle.

Usage

```
exporter_graphique(
  graphique,
  chemin,
  largeur = 20,
  hauteur = 14,
  dpi = 300L,
  fond = "white"
)
```

Arguments

graphique	ggplot — Objet graphique à exporter
chemin	character — Chemin de sortie avec extension (.png, .pdf, .svg)
largeur	numeric — Largeur en cm. Défaut : 20.
hauteur	numeric — Hauteur en cm. Défaut : 14.
dpi	integer — Résolution pour PNG (ignoré pour PDF/SVG). Défaut : 300.
fond	character — Couleur de fond. Défaut : "white".

Value

Chemin du fichier exporté (invisible).

Examples

```
## Not run:
 donnees <- data.frame(age=sample(0:80,100,replace=TRUE), sexe=sample(c("H","F"),100,replace=TRUE))
  p <- pyramide_ages(donnees, "age", "sexe")
  exporter_graphique(p, file.path(tempdir(), "pyramide.png"))

## End(Not run)
```

`exporter_sdmx` *Exporter des données au format SDMX*

Description

Génère un fichier SDMX-CSV ou SDMX-ML (Structure Data Message) conforme aux standards SDMX 2.1 pour l'échange de données statistiques avec les organisations internationales (FMI, BM, OCDE, etc.).

Usage

```
exporter_sdmx(
  data,
  flux_donnees,
  agence = "INS",
  vars_dimensions,
  vars_mesures,
  vars_attributs = NULL,
  fichier_sortie,
  version = "2.1"
)
```

Arguments

`data`	data.frame ou tibble — Données à exporter
`flux_donnees`	character — Identifiant du flux de données (DataFlow). Ex: "BEN_EMOP_2023"
`agence`	character — Identifiant de l'agence productrice. Ex: "INSAE", "INSD". Défaut : "INS".
`vars_dimensions`	character — Variables identifiant les dimensions (axes d'analyse). Ex: c("PAYS", "ANNEE", "REGION").
`vars_mesures`	character — Variables contenant les valeurs mesurées.
`vars_attributs`	character ou NULL — Variables d'attributs (métadonnées). Défaut : NULL.
`fichier_sortie`	character — Chemin du fichier de sortie (.csv).
`version`	character — Version SDMX. Défaut : "2.1".

Value

Chemin du fichier exporté (invisible).

Examples

```
## Not run:
  exporter_sdmx(
    data           = indicateurs_regionaux,
    flux_donnees   = "BEN_IDH_2023",
```

```
    agence          = "INSAE",
    vars_dimensions = c("region", "annee"),
    vars_mesures    = c("idh", "taux_pauvrete"),
    fichier_sortie  = file.path(tempdir(), "indicateurs_sdmx.csv")
  )

## End(Not run)
```

`fusion_datasets` *Fusionner plusieurs jeux de données*

Description

Fusionne plusieurs datasets horizontalement (jointure) ou verticalement (empilement). Gère les conflits de noms de variables et produit un rapport de fusion.

Usage

```
fusion_datasets(
  liste_data,
  type = c("vertical", "horizontal"),
  cle = NULL,
  jointure = c("gauche", "interne", "droite", "complete"),
  suffixes = c("_1", "_2")
)
```

Arguments

`liste_data`	list — Liste nommée de data.frames/tibbles à fusionner
`type`	character — Type de fusion : `"horizontal"` (jointure par clé), `"vertical"` (empilement / append). Défaut : "vertical".
`cle`	character ou NULL — Variable(s) clé(s) pour la fusion horizontale. Obligatoire si `type = "horizontal"`. Défaut : NULL.
`jointure`	character — Type de jointure horizontale : `"interne"`, `"gauche"`, `"droite"`, `"complete"`. Défaut : "gauche".
`suffixes`	character — Suffixes pour les variables en conflit lors d'une fusion horizontale. Défaut : c("_1", "_2").

Value

Un tibble fusionné.

Examples

```
## Not run:
  # Empilement de deux vagues d'enquête
  donnees_total <- fusion_datasets(
    liste_data = list(vague1 = emop_2022, vague2 = emop_2023),
    type       = "vertical"
  )
  # Jointure ménages + individus
  donnees_merged <- fusion_datasets(
    liste_data = list(menages = df_menages, individus = df_individus),
    type       = "horizontal",
    cle        = "id_menage"
  )

## End(Not run)
```

`generer_bulletin` *Generer un bulletin statistique periodique*

Description

Genere un bulletin statistique mensuel ou trimestriel avec indicateurs cles, graphiques de tendance et tableau de bord.

Usage

```
generer_bulletin(
  indicateurs,
  periode,
  pays = "Pays",
  sortie = c("html", "word"),
  chemin_sortie = tempdir(),
  ouvrir = FALSE
)
```

Arguments

`indicateurs`	data.frame – Tableau des indicateurs avec colonnes : `indicateur`, `valeur`, `unite`, `periode`
`periode`	character – Periode de reference (ex : "T1 2026", "Janvier 2026")
`pays`	character – Nom du pays. Defaut : "Pays"
`sortie`	character – Format : `"html"` ou `"word"`. Defaut : "html"
`chemin_sortie`	character – Repertoire de destination. Defaut : `tempdir()`
`ouvrir`	logical – Ouvrir apres generation. Defaut : FALSE

Value

Chemin du fichier genere (invisible)

Examples

```
## Not run:
  indicateurs <- data.frame(
    indicateur = c("Taux de pauvrete", "Taux de chomage",
                   "Acces eau potable", "Taux alphabetisation"),
    valeur     = c(71.8, 14.5, 42.3, 56.7),
    unite      = c("%", "%", "%", "%"),
    periode    = rep("2026", 4)
  )
  generer_bulletin(indicateurs, periode = "T1 2026",
                   pays = "Republique Centrafricaine")

## End(Not run)
```

`generer_metadonnees_ddi` *Générer une fiche de métadonnées DDI*

Description

Produit une fiche de métadonnées au format DDI (Data Documentation Initiative) Codebook 2.5, standard international pour l'archivage des enquêtes statistiques (IHSN, NADA, NESSTAR).

Usage

```
generer_metadonnees_ddi(
  data,
  titre = NULL,
  pays = NULL,
  annee = NULL,
  institution = NULL,
  auteurs = NULL,
  description = NULL,
  fichier_sortie = NULL,
  langue = "fr"
)
```

Arguments

`data`	data.frame ou tibble — Données de l'enquête
`titre`	character — Titre de l'enquête
`pays`	character — Pays concerné

annee	integer ou character — Année de l'enquête
institution	character — Institution productrice
auteurs	character ou NULL — Auteurs. Défaut : NULL.
description	character ou NULL — Description de l'enquête. Défaut : NULL.
fichier_sortie	character — Chemin du fichier XML de sortie
langue	character — Langue principale. Défaut : "fr".

Value

Chemin du fichier généré (invisible).

Examples

```
## Not run:
  generer_metadonnees_ddi(
    data        = donnees_emop,
    titre       = "Enquête Modulaire sur les Conditions de Vie - 2023",
    pays        = "Bénin",
    annee       = 2023,
    institution = "INSAE",
    fichier_sortie = file.path(tempdir(), "emop_2023_ddi.xml")
  )

## End(Not run)
```

---

generer_rapport *Générer un rapport statistique officiel*

Description

Produit un rapport Word (.docx) ou PDF à partir d'un template R Markdown. Intègre automatiquement les tableaux, graphiques et métadonnées. Compatible avec les templates AFRISTAT et PARIS21.

Usage

```
generer_rapport(
  donnees,
  template = "bulletin_mensuel",
  format_sortie = c("word", "pdf"),
  fichier_sortie = NULL,
  metadonnees = NULL,
  ouvrir = FALSE
)
```

Arguments

donnees	data.frame ou tibble — Données à inclure dans le rapport
template	character — Chemin vers le template .Rmd ou nom d'un template intégré : "bulletin_mensuel", "rapport_annuel", "fiche_pays". Défaut : "bulletin_mensuel".
format_sortie	character — "word" ou "pdf". Défaut : "word".
fichier_sortie	character ou NULL — Chemin du fichier de sortie. Si NULL, génère un nom automatique. Défaut : NULL.
metadonnees	list ou NULL — Liste de métadonnées à injecter : titre, auteur, pays, annee, institution. Défaut : NULL.
ouvrir	logical — Ouvrir le rapport après génération. Défaut : FALSE.

Value

Chemin du fichier généré (invisible).

Examples

```
## Not run:
  generer_rapport(
    donnees        = resultats_enquete,
    template       = "bulletin_mensuel",
    format_sortie  = "word",
    metadonnees    = list(
      titre       = "Bulletin mensuel - Mars 2024",
      pays        = "Bénin",
      institution = "INSAE",
      annee       = 2024
    )
  )

## End(Not run)
```

---

generer_rapport_enquete

Generer un rapport d'enquete menage parametre

Description

Genere un rapport statistique complet a partir de donnees d'enquete menage, en utilisant un template R Markdown pre-construit. Le rapport inclut une page de garde, des statistiques descriptives, une pyramide demographique et les indicateurs cles.

Usage

```
generer_rapport_enquete(
  donnees,
  meta = list(),
  template = c("enquete_menage", "bulletin"),
  sortie = c("html", "pdf", "word"),
  chemin_sortie = tempdir(),
  vars_analyse = NULL,
  var_poids = NULL,
  ouvrir = FALSE
)
```

Arguments

donnees	data.frame – Donnees d'enquete menage
meta	list – Metadonnees du rapport. Elements attendus : pays, titre, annee, source, auteur (tous optionnels)
template	character – Nom du template : "enquete_menage" ou "bulletin". Defaut : "enquete_menage"
sortie	character – Format de sortie : "html", "pdf" ou "word". Defaut : "html"
chemin_sortie	character – Repertoire de destination. Defaut : tempdir()
vars_analyse	character ou NULL – Variables a analyser. Si NULL, detecte automatiquement les variables numeriques. Defaut : NULL
var_poids	character ou NULL – Variable de ponderation. Defaut : NULL
ouvrir	logical – Ouvrir le rapport apres generation. Defaut : FALSE

Value

Chemin du fichier genere (invisible)

Examples

```
## Not run:
  set.seed(42)
  donnees <- data.frame(
    age    = sample(18:70, 200, replace = TRUE),
    revenu = rexp(200, rate = 1/250000),
    sexe   = sample(c("H","F"), 200, TRUE),
    milieu = sample(c("urbain","rural"), 200, TRUE),
    poids  = runif(200, 0.8, 1.3)
  )
  meta <- list(
    pays   = "Republique Centrafricaine",
    titre  = "Enquete sur les conditions de vie des menages",
    annee  = 2026L,
    source = "ICASEES",
    auteur = "Direction des Statistiques"
  )
```

```
  generer_rapport_enquete(donnees, meta, sortie = "html")

## End(Not run)
```

`generer_rapport_odd` *Generer un rapport de suivi des ODD parametre*

Description

Genere un rapport de suivi des Objectifs de Developpement Durable au format standard PARIS21/Nations Unies.

Usage

```
generer_rapport_odd(
  resultats_odd,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  cibles = NULL,
  sortie = c("html", "pdf", "word"),
  chemin_sortie = tempdir(),
  ouvrir = FALSE
)
```

Arguments

`resultats_odd`	list – Liste d'objets `saf_odd` (resultats de `odd_indicateur()`)
`pays`	character – Nom du pays. Defaut : "Pays"
`annee`	integer – Annee de reference. Defaut : annee courante
`cibles`	list ou NULL – Cibles nationales par code ODD. Defaut : NULL
`sortie`	character – Format : `"html"`, `"pdf"` ou `"word"`. Defaut : "html"
`chemin_sortie`	character – Repertoire. Defaut : `tempdir()`
`ouvrir`	logical – Ouvrir apres generation. Defaut : FALSE

Value

Chemin du fichier genere (invisible)

Examples

```
## Not run:
  set.seed(42)
  menages <- data.frame(
    depense_pc  = c(rexp(70, 1/150000), rexp(30, 1/500000)),
    electricite = sample(c(0L,1L), 100, TRUE, c(0.45, 0.55)),
```

```
  internet    = sample(c(0L,1L), 100, TRUE, c(0.72, 0.28))
)
r1 <- odd_indicateur(menages, "1.2.1",
                     var_depense = "depense_pc", seuil = 200000)
r2 <- odd_indicateur(menages, "7.1.1",
                     var_indicateur = "electricite")
generer_rapport_odd(
  list(r1, r2),
  pays  = "Republique Centrafricaine",
  annee = 2026L,
  cibles = list("1.2.1" = 50, "7.1.1" = 80)
)

## End(Not run)
```

`graphique_barres` *Graphique en barres pondéré*

Description

Génère un graphique en barres avec intervalles de confiance optionnels, adapté aux résultats d'enquêtes pondérées.

Usage

```
graphique_barres(
  data,
  var_x,
  var_y,
  var_groupe = NULL,
  var_ic_bas = NULL,
  var_ic_haut = NULL,
  position = c("dodge", "stack"),
  titre = NULL,
  label_x = NULL,
  label_y = NULL,
  pourcentage = FALSE,
  trier = FALSE
)
```

Arguments

`data`	data.frame, tibble ou résultat de `tab_croisee()` — Données source
`var_x`	character — Variable en abscisse (catégories)
`var_y`	character — Variable en ordonnée (valeurs)
`var_groupe`	character ou NULL — Variable de regroupement (barres groupées). Défaut : NULL.

var_ic_bas	character ou NULL — Variable borne inférieure IC. Défaut : NULL.
var_ic_haut	character ou NULL — Variable borne supérieure IC. Défaut : NULL.
position	character — "dodge" (groupé) ou "stack" (empilé). Défaut : "dodge".
titre	character ou NULL — Titre. Défaut : NULL.
label_x	character ou NULL — Label axe X. Défaut : NULL.
label_y	character ou NULL — Label axe Y. Défaut : NULL.
pourcentage	logical — Formater l'axe Y en pourcentage. Défaut : FALSE.
trier	logical — Trier les barres par valeur décroissante. Défaut : FALSE.

Value

Un objet ggplot.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(region=c("Nord","Sud","Est","Ouest"), valeur=c(35.2,28.7,41.1,22.5))
  graphique_barres(df, var_x="region", var_y="valeur", titre="Indicateur")
}
```

---

graphique_fgt *Graphique des indices FGT*

Description

Visualise les indices FGT nationaux et/ou par sous-groupe.

Usage

```
graphique_fgt(
  fgt_obj,
  type = c("barres", "indices"),
  variable = NULL,
  couleur = "#1B4965"
)
```

Arguments

fgt_obj	objet saf_fgt – Resultat de calcul_fgt()
type	character – "barres" ou "indices". Defaut : "barres"
variable	character ou NULL – Variable de sous-groupe a representer
couleur	character – Couleur principale. Defaut : "#1B4965"

Value

Objet ggplot2

Examples

```
## Not run:
  set.seed(42)
  menages <- data.frame(
    depense_pc = c(rexp(70, 1/150000), rexp(30, 1/400000)),
    poids  = runif(100, 0.8, 1.2),
    milieu = sample(c("urbain", "rural"), 100, TRUE, c(0.4, 0.6))
  )
  fgt <- calcul_fgt(menages, "depense_pc", 220000,
                    poids = "poids", sous_groupes = "milieu")
  graphique_fgt(fgt, type = "barres", variable = "milieu")

## End(Not run)
```

---

graphique_tendance *Graphique de tendance temporelle*

Description

Génère un graphique de tendance pour un ou plusieurs indicateurs sur une période temporelle. Adapté au suivi des indicateurs ODD et des indicateurs macroéconomiques.

Usage

```
graphique_tendance(
  data,
  var_temps,
  var_valeur = NULL,
  var_indicateur = NULL,
  vars_indicateurs = NULL,
  titre = NULL,
  label_y = "Valeur",
  afficher_points = TRUE,
  afficher_valeurs = FALSE,
  lisser = FALSE
)
```

Arguments

data	data.frame ou tibble — Données en format long ou large
var_temps	character — Variable temporelle (année, trimestre...)
var_valeur	character — Variable de valeur (format long) ou NULL si format large. Défaut : NULL.
var_indicateur	character ou NULL — Variable d'indicateur (format long). Défaut : NULL.
vars_indicateurs	character ou NULL — Vecteur de colonnes à tracer (format large). Défaut : NULL.

titre	character ou NULL — Titre. Défaut : NULL.
label_y	character — Label axe Y. Défaut : "Valeur".
afficher_points	logical — Afficher les points. Défaut : TRUE.
afficher_valeurs	logical — Annoter les valeurs. Défaut : FALSE.
lisser	logical — Ajouter une courbe lissée (loess). Défaut : FALSE.

Value

Un objet ggplot.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(annee=2015:2023, pib=c(45,48,51,49,53,55,52,58,61))
  graphique_tendance(df, var_temps="annee", vars_indicateurs="pib", titre="PIB")
}
```

---

harmoniser_regions *Harmoniser les noms de régions/provinces*

Description

Standardise les noms de régions géographiques selon un référentiel national ou africain. Corrige les variantes orthographiques, les abréviations et les noms en langues locales.

Usage

```
harmoniser_regions(
  data,
  var_region,
  pays = NULL,
  table_correspondance = NULL,
  var_sortie = "region_std",
  signaler_non_trouves = TRUE
)
```

Arguments

data	data.frame ou tibble — Données à harmoniser
var_region	character — Nom de la variable contenant les régions
pays	character ou NULL — Code pays ISO2 pour utiliser le référentiel intégré (ex: "BJ", "BF", "SN", "CI", "ML", "NE", "TG", "CM", "GN"). Si NULL, utilise table_correspondance. Défaut : NULL.

table_correspondance
: data.frame ou NULL — Table de correspondance avec colonnes `original` et `standardise`. Si NULL et pays est NULL, tente une correspondance floue automatique. Défaut : NULL.

var_sortie
: character — Nom de la nouvelle colonne standardisée. Défaut : "region_std".

signaler_non_trouves
: logical — Afficher les valeurs non reconnues. Défaut : TRUE.

Value

Le tibble avec une colonne `var_sortie` ajoutée contenant les régions standardisées.

Examples

```
## Not run:
  donnees <- data.frame(region = c("Littoral","Atlantique","Borgou"))
  harmoniser_regions(donnees, var_region = "region", pays = "BJ")

## End(Not run)
```

---

import_cspro *Importer des données CSPro*

Description

Lit les fichiers de données produits par CSPro (format .dat avec dictionnaire .dcf associé). Retourne un tibble avec les labels issus du dictionnaire CSPro. Fonction prioritaire pour les RGPH africains. Compatible avec CSPro 4.x à 8.x.

Usage

```
import_cspro(
  fichier_dat,
  fichier_dcf = NULL,
  niveau = NULL,
  encoding = "UTF-8",
  max_lignes = NULL,
  verbose = TRUE
)
```

Arguments

fichier_dat
: character — Chemin vers le fichier de données (.dat)

fichier_dcf
: character ou NULL — Chemin vers le dictionnaire (.dcf). Si NULL, recherche automatiquement un .dcf de même nom dans le même répertoire. Défaut : NULL.

niveau	character ou NULL — Niveau d'enregistrement à lire (ex: "MENAGE", "INDIVIDU", "LOGEMENT"). Si NULL, lit le premier niveau. Défaut : NULL.
encoding	character — Encodage du fichier source. Défaut : "UTF-8".
max_lignes	integer ou NULL — Nombre maximum de lignes à lire (utile pour les tests sur grands fichiers RGPH). Défaut : NULL (tout lire).
verbose	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble avec une colonne par variable CSPro. Les labels de valeurs sont stockés dans les attributs du tibble (`attr(., "labels_cspro")`).

Examples

```
## Not run:
  # Import du niveau ménage d'un RGPH
  menages <- import_cspro(
    fichier_dat = "data/rgph_2024.dat",
    fichier_dcf = "data/rgph_2024.dcf",
    niveau      = "MENAGE"
  )

  # Test sur les 1000 premières lignes
  test <- import_cspro("data/rgph_2024.dat", max_lignes = 1000)

## End(Not run)
```

---

import_csv *Importer un fichier CSV*

Description

Importe un fichier CSV avec détection automatique de l'encodage, du séparateur et du séparateur décimal. Gère les formats courants des INS africains (séparateurs point-virgule, virgule, tabulation).

Usage

```
import_csv(
  chemin,
  separateur = NULL,
  encodage = "UTF-8",
  decimal = ".",
  na = c("", "NA", "N/A", "n/a", ".", " "),
  verbose = TRUE
)
```

Arguments

chemin	character — Chemin vers le fichier CSV
separateur	character ou NULL — Séparateur de colonnes. Si NULL, détection automatique. Défaut : NULL.
encodage	character — Encodage du fichier. Défaut : "UTF-8" (essaie aussi "latin1" si UTF-8 échoue).
decimal	character — Séparateur décimal ("." ou ","). Défaut : ".".
na	character — Valeurs à interpréter comme NA. Défaut : c("", "NA", "N/A", "n/a", ".", " ").
verbose	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble.

Examples

```
## Not run:
  donnees <- import_csv("data/prix_marches.csv")
  donnees_fr <- import_csv("data/donnees_fr.csv", decimal = ",")

## End(Not run)
```

---

import_excel *Importer un fichier Excel*

Description

Importe un fichier Excel (.xlsx, .xls) avec détection automatique des feuilles, gestion des en-têtes multiples et conversion intelligente des types de colonnes. Optimisé pour les formats courants des INS africains.

Usage

```
import_excel(
  chemin,
  feuille = 1,
  skip = 0,
  col_types = NULL,
  na = c("", "NA", "N/A", "n/a", ".", " "),
  verbose = TRUE
)
```

Arguments

chemin	character — Chemin vers le fichier Excel (.xlsx ou .xls)
feuille	character ou integer ou NULL — Nom ou numéro de la feuille à importer. Si NULL, importe toutes les feuilles sous forme de liste. Défaut : 1 (première feuille).
skip	integer — Nombre de lignes à ignorer avant l'en-tête. Défaut : 0.
col_types	character ou NULL — Types des colonnes (voir readxl). Si NULL, détection automatique. Défaut : NULL.
na	character — Valeurs à interpréter comme NA. Défaut : c("", "NA", "N/A", "n/a", ".", " ").
verbose	logical — Afficher les messages de progression. Défaut : TRUE.

Value

Un tibble si une seule feuille, une liste de tibbles si `feuille = NULL`.

See Also

`import_csv`, `valider_dictionnaire`

Examples

```
## Not run:
  # Import de la première feuille
  donnees <- import_excel("data/enquete_menage.xlsx")

  # Import d'une feuille spécifique
  menages <- import_excel("data/emop_2024.xlsx", feuille = "Menages")

  # Import de toutes les feuilles
  toutes <- import_excel("data/rapport.xlsx", feuille = NULL)

## End(Not run)
```

---

import_kobo *Importer des données depuis KoboToolbox*

Description

Importe un formulaire KoboToolbox via fichier local (XLS/JSON) ou via l'API REST KoboToolbox. Retourne un tibble annoté avec les métadonnées du formulaire. Compatible avec KoboToolbox et KoBoCAT.

Usage

```
import_kobo(
  source,
  uid = NULL,
  token = Sys.getenv("KOBO_TOKEN"),
  format = c("xls", "json"),
  langue = "French (fr)",
  verbose = TRUE
)
```

Arguments

source	character — Chemin vers un fichier XLS/JSON local, ou URL de base de l'API (ex: "https://kf.kobotoolbox.org").
uid	character ou NULL — Identifiant unique du formulaire (requis si source = URL API). Défaut : NULL.
token	character — Jeton d'authentification API. Peut aussi être défini via la variable d'environnement KOBO_TOKEN. Défaut : Sys.getenv("KOBO_TOKEN").
format	character — Format du fichier local : "xls" ou "json". Ignoré si source est une URL. Défaut : "xls".
langue	character — Code langue pour les labels multilingues (ex: "French (fr)", "English (en)"). Défaut : "French (fr)".
verbose	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble avec les colonnes du formulaire. L'attribut attr(., "metadonnees_kobo") contient le dictionnaire des variables.

Examples

```
## Not run:
  # Import depuis fichier XLS local
  donnees <- import_kobo(source = "data/enquete_2024.xls")

  # Import depuis API KoboToolbox
  Sys.setenv(KOBO_TOKEN = "mon_token_secret")
  donnees <- import_kobo(
    source = "https://kf.kobotoolbox.org",
    uid    = "aXmNk7pQrS",
    langue = "French (fr)"
  )

## End(Not run)
```

`import_odk` *Importer des données ODK Central*

Description

Importe les soumissions d'un formulaire ODK Central via l'API REST ou depuis un fichier d'export local (ZIP ou CSV). Compatible avec ODK Central 1.x et 2.x.

Usage

```
import_odk(
  source,
  projet_id = NULL,
  formulaire_id = NULL,
  email = Sys.getenv("ODK_EMAIL"),
  mot_de_passe = Sys.getenv("ODK_PASSWORD"),
  inclure_metadonnees = FALSE,
  verbose = TRUE
)
```

Arguments

`source`	character — URL du serveur ODK Central (ex: "https://odk.monins.org") ou chemin vers un fichier d'export .zip/.csv.
`projet_id`	integer ou NULL — ID du projet ODK. Requis si source est une URL. Défaut : NULL.
`formulaire_id`	character ou NULL — ID du formulaire ODK. Requis si source est une URL. Défaut : NULL.
`email`	character ou NULL — Email de connexion ODK Central. Peut aussi être défini via `ODK_EMAIL`. Défaut : NULL.
`mot_de_passe`	character — Mot de passe ODK Central. Peut aussi être défini via `ODK_PASSWORD`. Défaut : `Sys.getenv("ODK_PASSWORD")`.
`inclure_metadonnees`	logical — Inclure les colonnes de métadonnées ODK (timestamps, deviceid, etc.). Défaut : FALSE.
`verbose`	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble contenant les soumissions du formulaire.

Examples

```
## Not run:
  # Import depuis export ZIP local
  donnees <- import_odk(source = "data/odk_export_2024.zip")

  # Import depuis API ODK Central
  Sys.setenv(ODK_EMAIL = "admin@ins.org", ODK_PASSWORD = "motdepasse")
  donnees <- import_odk(
    source        = "https://odk.monins.org",
    projet_id     = 1,
    formulaire_id = "enquete_menage_2024"
  )

## End(Not run)
```

`import_sas` *Importer un fichier SAS*

Description

Importe un fichier SAS (.sas7bdat) ou un fichier de formats SAS (.sas7bcat) avec préservation des labels.

Usage

```
import_sas(chemin, chemin_formats = NULL, encoding = "UTF-8", verbose = TRUE)
```

Arguments

`chemin`	character — Chemin vers le fichier .sas7bdat
`chemin_formats`	character ou NULL — Chemin vers le fichier de formats .sas7bcat (optionnel). Défaut : NULL.
`encoding`	character — Encodage. Défaut : "UTF-8".
`verbose`	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble.

Examples

```
## Not run:
  donnees <- import_sas("data/enquete_emploi.sas7bdat")

## End(Not run)
```

`import_spss` *Importer un fichier SPSS*

Description

Importe un fichier SPSS (.sav ou .zsav) avec préservation des labels de variables et de valeurs SPSS.

Usage

```
import_spss(
  chemin,
  garder_labels = TRUE,
  convertir_labels = FALSE,
  encoding = NULL,
  verbose = TRUE
)
```

Arguments

`chemin`	character — Chemin vers le fichier .sav ou .zsav
`garder_labels`	logical — Conserver les labels. Défaut : TRUE.
`convertir_labels`	logical — Convertir en facteurs. Défaut : FALSE.
`encoding`	character ou NULL — Encodage. Si NULL, détection auto. Défaut : NULL.
`verbose`	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble.

Examples

```
## Not run:
  mics <- import_spss("data/mics6_enfants.sav")

## End(Not run)
```

`import_stata` *Importer un fichier Stata*

Description

Importe un fichier Stata (.dta) avec préservation des labels de variables et de valeurs. Compatible avec toutes les versions Stata (Stata 8 à Stata 18).

Usage

```
import_stata(
  chemin,
  encoding = "UTF-8",
  garder_labels = TRUE,
  convertir_labels = FALSE,
  verbose = TRUE
)
```

Arguments

`chemin`	character — Chemin vers le fichier .dta
`encoding`	character — Encodage pour les labels. Défaut : "UTF-8".
`garder_labels`	logical — Conserver les labels comme attributs. Défaut : TRUE.
`convertir_labels`	logical — Convertir les variables labellisées en facteurs. Défaut : FALSE.
`verbose`	logical — Afficher les messages. Défaut : TRUE.

Value

Un tibble avec attributs de labels si `garder_labels = TRUE`.

Examples

```
## Not run:
  eds <- import_stata("data/eds_2021.dta")
  eds_facteurs <- import_stata("data/eds_2021.dta", convertir_labels = TRUE)

## End(Not run)
```

`imputer_valeurs` *Imputer les valeurs manquantes*

Description

Impute les valeurs manquantes d'un dataset selon la méthode spécifiée. Supporte l'imputation simple (statistiques descriptives), hot-deck et par régression. Produit un rapport de traçabilité.

Usage

```
imputer_valeurs(
  data,
  vars = NULL,
  methode = c("mediane", "moyenne", "mode", "hot_deck", "regression"),
  vars_auxiliaires = NULL,
  graine = 42L,
  rapport = TRUE
)
```

Arguments

`data`	data.frame ou tibble — Données avec valeurs manquantes
`vars`	character ou NULL — Variables à imputer. Si NULL, toutes les variables avec valeurs manquantes. Défaut : NULL.
`methode`	character — Méthode d'imputation : `"mediane"`, `"moyenne"`, `"mode"`, `"hot_deck"`, `"regression"`. Défaut : "mediane".
`vars_auxiliaires`	character ou NULL — Variables auxiliaires pour l'imputation par régression ou hot-deck. Défaut : NULL.
`graine`	integer — Graine aléatoire pour la reproductibilité. Défaut : 42.
`rapport`	logical — Retourner un rapport d'imputation. Défaut : TRUE.

Value

Si `rapport = FALSE` : tibble imputé. Si `rapport = TRUE` : liste avec `$donnees` et `$rapport`.

Examples

```
## Not run:
  donnees <- data.frame(revenu_mensuel=c(150000,NA,200000), age=c(25,34,NA))
  imputer_valeurs(donnees, vars=c("revenu_mensuel","age"), methode="mediane")

## End(Not run)
```

lister_templates *Lister les templates de rapport disponibles*

Description

Retourne la liste des templates R Markdown disponibles dans statAfrikR avec leur description et parametres.

Usage

```
lister_templates()
```

Value

Un tibble avec : nom, description, parametres, format_sortie

Examples

```
lister_templates()
```

nettoyer_libelles *Nettoyer les libellés de variables textuelles*

Description

Normalise les chaînes de caractères : suppression des espaces superflus, normalisation de la casse, gestion des caractères spéciaux, correction des encodages. Préserve les caractères accentués africains et francophones.

Usage

```
nettoyer_libelles(
  data,
  vars = NULL,
  casse = c("titre", "majuscule", "minuscule", "aucune"),
  supprimer_espaces = TRUE,
  supprimer_ponctuation = FALSE,
  encodage = "UTF-8"
)
```

Arguments

data	data.frame ou tibble — Données à nettoyer
vars	character ou NULL — Variables à nettoyer. Si NULL, toutes les variables de type character. Défaut : NULL.
casse	character — Normalisation de la casse : "titre" (Première Lettre Majuscule), "majuscule", "minuscule", "aucune". Défaut : "titre".
supprimer_espaces	logical — Supprimer les espaces multiples et les espaces en début/fin. Défaut : TRUE.
supprimer_ponctuation	logical — Supprimer la ponctuation superflue. Défaut : FALSE.
encodage	character — Encodage cible. Défaut : "UTF-8".

Value

Un tibble avec les variables textuelles nettoyées.

Examples

```
## Not run:
  donnees <- data.frame(region = c(" nord ","SUD"), commune = c("Cotonou "," parakou"))
  nettoyer_libelles(donnees, vars = c("region", "commune"))

## End(Not run)
```

---

odd_catalogue *Catalogue des indicateurs ODD pour les INS africains*

Description

Retourne le catalogue des indicateurs ODD implementes dans statAfrikR, selectionnes selon leur pertinence et disponibilite pour les INS africains (reference PARIS21, AFRISTAT, Nations Unies).

Usage

```
odd_catalogue(
  objectif = NULL,
  disponibilite = NULL,
  statut = NULL,
  format = c("tibble", "flextable")
)
```

Arguments

objectif	integer ou NULL – Filtrer par numero d'ODD (1 a 17). NULL retourne tous les indicateurs. Defaut : NULL
disponibilite	character ou NULL – Filtrer par disponibilite des donnees : "haute", "moyenne" ou "faible". Defaut : NULL
statut	character ou NULL – Filtrer par statut d'implementation : "implementee" ou "documentation". Defaut : NULL
format	character – "tibble" ou "flextable". Defaut : "tibble"

Value

Tibble ou flextable du catalogue filtre

Examples

```
# Tous les indicateurs
odd_catalogue()

# Indicateurs ODD 1 (pauvrete) implementes
odd_catalogue(objectif = 1, statut = "implementee")

# Indicateurs a haute disponibilite
odd_catalogue(disponibilite = "haute")
```

---

odd_indicateur *Calculer un indicateur ODD*

Description

Calcule un indicateur ODD specifique a partir des donnees d'enquete. Retourne un objet saf_odd avec la valeur, les metadonnees et les composantes du calcul.

Usage

```
odd_indicateur(data, code_odd, ..., poids = NULL, na.rm = TRUE)
```

Arguments

data	data.frame ou svydesign – Donnees d'enquete
code_odd	character – Code ODD au format "X.Y.Z" (ex : "1.1.1", "7.1.1"). Voir odd_catalogue() pour la liste
...	Arguments specifiques a chaque indicateur (voir Details)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
na.rm	logical – Exclure les NA. Defaut : TRUE

Details

Arguments specifiques par indicateur :

1.1.1 `var_depense` : variable de depenses par tete, `seuil` : seuil en monnaie locale (defaut : equivalence 2.15 USD PPA)

1.2.1 `var_depense` : variable de depenses par tete, `seuil` : seuil national de pauvrete

2.2.1 `var_taille` : taille de l'enfant (cm), `var_age_mois` : age en mois, `var_sexe` : sexe (1=M, 2=F ou "M"/"F")

3.2.1 `var_deces_enfant` : indicateur deces enfant (0/1), `var_naissances` : nombre de naissances vivantes ou variable indicatrice

5.3.1 `var_age_mariage` : age au premier mariage, `var_age_actuel` : age actuel, `seuil_age` : seuil (defaut : 18)

5.5.2, 8.5.2, 8.6.1, 6.1.1, 7.1.1, 10.2.1, 11.1.1, 16.9.1, 17.8.1 `var_indicateur` : variable binaire (1 = condition remplie, 0 = non)

10.1.1 `var_depense` : variable de depenses, `var_periode` : variable de periode (annees), `seuil_pct` : percentile inferieur (defaut : 40)

Value

Un objet de classe `saf_odd` avec :

valeur Valeur de l'indicateur

unite Unite de mesure

code_odd Code ODD

titre_court Libelle court

numerateur Numerateur du calcul

denominateur Denominateur du calcul

n_obs Observations utilisees

na_count Valeurs manquantes exclues

meta Metadonnees completes du catalogue

Examples

```
set.seed(42)
menages <- data.frame(
  depense_pc = c(rexp(70, rate = 1/150000), rexp(30, rate = 1/500000)),
  poids      = runif(100, 0.8, 1.2),
  electricite = sample(c(0L, 1L), 100, TRUE, prob = c(0.45, 0.55)),
  eau_potable = sample(c(0L, 1L), 100, TRUE, prob = c(0.35, 0.65)),
  internet    = sample(c(0L, 1L), 100, TRUE, prob = c(0.7, 0.3))
)

# ODD 1.2.1 - Pauvrete nationale
odd_indicateur(menages, "1.2.1",
               var_depense = "depense_pc", seuil = 200000,
               poids = "poids")
```

```
# ODD 7.1.1 - Acces a l'electricite
odd_indicateur(menages, "7.1.1", var_indicateur = "electricite",
               poids = "poids")
```

`palette_ins` *Palette de couleurs INS*

Description

Retourne une palette de couleurs officielle pour les graphiques INS, compatible daltonisme.

Usage

```
palette_ins(n = 6, type = c("categoriel", "sequentiel", "divergent"))
```

Arguments

`n`	integer — Nombre de couleurs souhaité. Défaut : 6.
`type`	character — `"categoriel"`, `"sequentiel"`, `"divergent"`. Défaut : "categoriel".

Value

Vecteur de codes couleurs hexadécimaux.

Examples

```
palette_ins(4)
```

`pyramide_ages` *Pyramide des âges*

Description

Génère une pyramide des âges pondérée à partir de données individuelles ou agrégées. Adapté aux recensements et enquêtes démographiques.

Usage

```
pyramide_ages(
  data,
  var_age,
  var_sexe,
  var_poids = NULL,
  modalite_homme = "Masculin",
  modalite_femme = "Féminin",
  largeur_classe = 5L,
  age_max = 80L,
  titre = NULL,
  pourcentage = TRUE,
  couleur_homme = "#1B6CA8",
  couleur_femme = "#E8872A"
)
```

Arguments

data	data.frame ou tibble — Données individuelles
var_age	character — Variable d'âge
var_sexe	character — Variable de sexe/genre
var_poids	character ou NULL — Variable de pondération. Défaut : NULL.
modalite_homme	character — Modalité masculine. Défaut : "Masculin".
modalite_femme	character — Modalité féminine. Défaut : "Féminin".
largeur_classe	integer — Largeur des classes d'âge en années. Défaut : 5.
age_max	integer — Âge maximum affiché. Défaut : 80.
titre	character ou NULL — Titre du graphique. Défaut : NULL.
pourcentage	logical — Afficher en pourcentage (TRUE) ou effectifs (FALSE). Défaut : TRUE.
couleur_homme	character — Couleur hommes. Défaut : "#1B6CA8".
couleur_femme	character — Couleur femmes. Défaut : "#E8872A".

Value

Un objet ggplot.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
 donnees <- data.frame(age=sample(0:80,100,replace=TRUE), sexe=sample(c("H","F"),100,replace=TRUE))
  pyramide_ages(donnees, var_age="age", var_sexe="sexe")
}
```

recoder_variable *Recoder une variable*

Description

Recode une variable selon une table de correspondance explicite. Supporte le recodage numérique (classes d'âge, quintiles) et textuel (modalités). Traçabilité complète des transformations.

Usage

```
recoder_variable(
  data,
  var,
  table_recodage,
  var_sortie = NULL,
  na_si_absent = TRUE
)
```

Arguments

data	data.frame ou tibble — Données
var	character — Nom de la variable à recoder
table_recodage	data.frame — Table avec colonnes avant et apres, ou vecteur nommé.
var_sortie	character ou NULL — Nom de la variable recodée. Si NULL, remplace la variable originale. Défaut : NULL.
na_si_absent	logical — Mettre NA si la valeur n'est pas dans la table de recodage. Défaut : TRUE.

Value

Le tibble avec la variable recodée.

Examples

```
## Not run:
  # Recodage des classes d'âge
  table_age <- data.frame(
    avant = c("15-24", "25-34", "35-49", "50+"),
    apres = c("Jeune", "Adulte", "Adulte", "Senior")
  )
  donnees <- recoder_variable(donnees, "classe_age", table_age)
  # Recodage avec vecteur nommé
  donnees <- recoder_variable(
    donnees, "sexe",
    table_recodage = c("1" = "Masculin", "2" = "Féminin")
  )

## End(Not run)
```

saf_cameroun_departements

Departements du Cameroun

Description

10 regions du Cameroun.

Usage

```
saf_cameroun_departements
```

Format

Objet sf avec colonnes : departement, code, geometry.

Source

Natural Earth – statAfrikR Foundation

Examples

```
cam <- carte_zones('subdivisions', pays = 'CMR')
```

saf_cedeao

Zone CEDEAO – 15 pays membres

Description

Communaute Economique des Etats de l'Afrique de l'Ouest.

Usage

```
saf_cedeao
```

Format

Objet sf (sous-ensemble de saf_pays_afrique)

Source

Natural Earth – statAfrikR Foundation

Examples

```
cedeao <- carte_zones('cedeao')
```

saf_cemac *Zone CEMAC – 6 pays membres*

Description

Cameroun, Centrafrique, Congo, Gabon, Guinee Equatoriale, Tchad.

Usage

```
saf_cemac
```

Format

Objet sf (sous-ensemble de saf_pays_afrique)

Source

Natural Earth – statAfrikR Foundation

Examples

```
cemac <- carte_zones('cemac')
```

saf_eau *Zone EAC – Afrique de l'Est*

Description

Communaute d'Afrique de l'Est.

Usage

```
saf_eau
```

Format

Objet sf (sous-ensemble de saf_pays_afrique)

Source

Natural Earth – statAfrikR Foundation

Examples

```
eau <- carte_zones('eau')
```

saf_pays_afrique *Pays africains – fond de carte integre*

Description

Fond de carte des 54 pays africains.

Usage

```
saf_pays_afrique
```

Format

Objet sf avec colonnes : pays, pays_fr, iso2, iso3, sous_region, geometry.

Source

Natural Earth via rnaturalearth – statAfrikR Foundation

Examples

```
afrique <- carte_zones('afrique')
```

saf_rca_prefectures *Prefectures de la Republique Centrafricaine*

Description

17 prefectures de la RCA avec codes administratifs.

Usage

```
saf_rca_prefectures
```

Format

Objet sf avec colonnes : pays, iso3, prefecture, code, geometry.

Source

Natural Earth – statAfrikR Foundation

Examples

```
rca <- carte_zones('rca')
```

saf_sadc *Zone SADC – Afrique Australe*

Description

Communaute de Developpement de l'Afrique Australe.

Usage

```
saf_sadc
```

Format

Objet sf (sous-ensemble de saf_pays_afrique)

Source

Natural Earth – statAfrikR Foundation

Examples

```
sadc <- carte_zones('sadc')
```

saf_senegal_regions *Regions du Senegal*

Description

14 regions du Senegal.

Usage

```
saf_senegal_regions
```

Format

Objet sf avec colonnes : region, code, geometry.

Source

Natural Earth – statAfrikR Foundation

Examples

```
sen <- carte_zones('subdivisions', pays = 'SEN')
```

saf_subdivisions_afrique

Subdivisions de niveau 1 – tous les pays africains

Description

Prefectures, regions et provinces de 43 pays africains (744 subdivisions + 17 prefectures RCA = 761 total).

Usage

```
saf_subdivisions_afrique
```

Format

Objet sf avec colonnes : pays, iso3, subdivision, code_admin, geometry.

Source

Natural Earth – statAfrikR Foundation

Examples

```
cam <- carte_zones('subdivisions', pays = 'CMR')
```

standardiser_ages

Standardiser les âges déclarés

Description

Détecte et corrige le "heap effect" (attraction vers les âges ronds) fréquent dans les enquêtes africaines où les âges sont déclarés. Calcule l'indice de Whipple et l'indice de Myers pour évaluer la qualité.

Usage

```
standardiser_ages(
  data,
  var_age = "age",
  methode = c("aucune", "interpolation", "united_nations"),
  age_min = 0L,
  age_max = 120L
)
```

Arguments

data	data.frame ou tibble — Données
var_age	character — Nom de la variable d'âge
methode	character — Méthode de correction : "aucune" (diagnostic uniquement), "interpolation" (répartition uniforme autour des âges ronds), "united_nations" (méthode Nations Unies). Défaut : "aucune".
age_min	integer — Âge minimum valide. Défaut : 0.
age_max	integer — Âge maximum valide. Défaut : 120.

Value

Une liste avec :

donnees	tibble avec âges corrigés si methode != "aucune"
indice_whipple	numeric — Indice de Whipple (1 = parfait, > 1.05 = problème)
indice_myers	numeric — Indice de Myers (0 = parfait)
diagnostic	character — Évaluation de la qualité

Examples

```
## Not run:
  donnees <- data.frame(age = sample(0:80, 200, replace=TRUE))
  standardiser_ages(donnees, var_age="age")

## End(Not run)
```

---

stat_descr *Statistiques descriptives pondérées*

Description

Calcule les statistiques descriptives complètes pour une ou plusieurs variables numériques, avec prise en compte optionnelle du plan de sondage complexe. Produit un tableau formaté prêt à publier.

Usage

```
stat_descr(
  data,
  vars,
  groupe = NULL,
  ponderee = TRUE,
  ic = TRUE,
  format_sortie = c("tibble", "flextable")
)
```

Arguments

data	data.frame, tibble ou objet svydesign — Données source
vars	character — Noms des variables à analyser
groupe	character ou NULL — Variable de regroupement. Défaut : NULL.
ponderee	logical — Utiliser les pondérations si data est un svydesign. Défaut : TRUE.
ic	logical — Calculer les intervalles de confiance à 95%. Défaut : TRUE.
format_sortie	character — Format : "tibble" ou "flextable". Défaut : "tibble".

Value

Tibble ou flextable avec : n, moyenne, médiane, écart-type, min, max, IC95.

Examples

```
## Not run:
  donnees <- data.frame(age=c(25,34,45), revenu=c(150000,200000,180000))
  stat_descr(donnees, vars=c("age","revenu"))

## End(Not run)
```

---

supprimer_doublons *Détecter et supprimer les doublons*

Description

Identifie et supprime les enregistrements dupliqués selon une ou plusieurs clés d'identification. Produit un rapport des doublons détectés.

Usage

```
supprimer_doublons(
  data,
  cles = NULL,
  garder = c("premier", "dernier", "aucun"),
  rapport = TRUE
)
```

Arguments

data	data.frame ou tibble — Données à dédupliquer
cles	character ou NULL — Variables clés pour la détection. Si NULL, utilise toutes les colonnes. Défaut : NULL.
garder	character — Quel doublon conserver : "premier" (première occurrence), "dernier" (dernière occurrence), "aucun" (supprimer tous les doublons). Défaut : "premier".
rapport	logical — Retourner un rapport des doublons. Défaut : TRUE.

Value

Si rapport = FALSE : tibble dédupliqué. Si rapport = TRUE : liste avec $donnees et $rapport.

Examples

```
## Not run:
  donnees <- data.frame(id=c(1,2,2,3), val=c(10,20,20,30))
  supprimer_doublons(donnees, cles="id")

## End(Not run)
```

---

tab_croisee *Tableau croisé pondéré avec intervalles de confiance*

Description

Produit un tableau croisé (ou tableau de fréquences simple) avec prise en compte optionnelle de la pondération et du plan de sondage complexe. Résultat formaté pour publication directe.

Usage

```
tab_croisee(
  data,
  var_ligne,
  var_col = NULL,
  var_poids = NULL,
  ic = TRUE,
  pourcentage = c("colonne", "ligne", "total"),
  format_sortie = c("flextable", "tibble")
)
```

Arguments

data	data.frame, tibble ou objet svydesign — Données source
var_ligne	character — Variable en ligne
var_col	character ou NULL — Variable en colonne. Si NULL, tableau de fréquences simple. Défaut : NULL.
var_poids	character ou NULL — Variable de pondération (ignorée si data est un svydesign). Défaut : NULL.
ic	logical — Calculer les IC à 95%. Défaut : TRUE.
pourcentage	character — Type : "colonne", "ligne", "total". Défaut : "colonne".
format_sortie	character — "tibble" ou "flextable". Défaut : "flextable".

Value

Tibble ou flextable du tableau croisé.

Examples

```
## Not run:
  donnees <- data.frame(
    region = sample(c("Nord", "Sud", "Est"), 50, replace = TRUE),
    sexe   = sample(c("H", "F"), 50, replace = TRUE)
  )
  tab_croisee(donnees, "region", "sexe")

## End(Not run)
```

`tableau_croise_ins` *Tableau croise pondere - format institutionnel INS*

Description

Produit un tableau de contingence pondere avec marges, pourcentages et test du chi-deux.

Usage

```
tableau_croise_ins(
  data,
  ligne,
  colonne,
  poids = NULL,
  type_pct = c("colonne", "ligne", "total"),
  marges = TRUE,
  chi2 = TRUE,
  format = c("tibble", "flextable", "excel"),
  chemin = NULL,
  titre = NULL
)
```

Arguments

`data`	data.frame, tibble ou `svydesign` – Donnees
`ligne`	character – Variable en ligne
`colonne`	character – Variable en colonne
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`type_pct`	character – `"colonne"`, `"ligne"` ou `"total"`. Defaut : "colonne"
`marges`	logical – Afficher les totaux. Defaut : TRUE
`chi2`	logical – Calculer le test du chi-deux. Defaut : TRUE
`format`	character – `"tibble"`, `"flextable"` ou `"excel"`. Defaut : "tibble"
`chemin`	character ou NULL – Chemin Excel. Defaut : NULL
`titre`	character ou NULL – Titre. Defaut : NULL

Value

Tibble, flextable ou chemin Excel

Examples

```
set.seed(42)
donnees <- data.frame(
  region = sample(c("Nord", "Sud", "Est", "Ouest"), 300, TRUE),
  milieu = sample(c("urbain", "rural"), 300, TRUE, prob = c(0.4, 0.6)),
  poids  = runif(300, 0.7, 1.4)
)
tableau_croise_ins(donnees, "region", "milieu", poids = "poids")
```

---

tableau_descriptif *Tableau de statistiques descriptives institutionnel*

Description

Produit un tableau de statistiques descriptives pondere, formate selon les conventions des INS africains. Exportable en Word (flextable) ou Excel (openxlsx2).

Usage

```
tableau_descriptif(
  data,
  vars,
  poids = NULL,
  par = NULL,
  stats = c("n", "moyenne", "mediane", "ecart_type", "min", "max", "ic"),
  format = c("tibble", "flextable", "excel"),
  chemin = NULL,
  titre = NULL,
  source = NULL
)
```

Arguments

data	data.frame, tibble ou objet svydesign – Donnees
vars	character – Variables numeriques a analyser
poids	character ou NULL – Variable de ponderation. Defaut : NULL
par	character ou NULL – Variable de ventilation (groupe). Defaut : NULL
stats	character – Statistiques parmi : "n", "moyenne", "mediane", "ecart_type", "min", "max", "ic". Defaut : toutes
format	character – "tibble", "flextable" ou "excel". Defaut : "tibble"
chemin	character ou NULL – Chemin Excel. Defaut : NULL
titre	character ou NULL – Titre du tableau. Defaut : NULL
source	character ou NULL – Note source. Defaut : NULL

Value

Tibble, flextable ou chemin fichier Excel

Examples

```
set.seed(42)
donnees <- data.frame(
  age    = sample(18:65, 200, replace = TRUE),
  revenu = rexp(200, rate = 1/250000),
  poids  = runif(200, 0.8, 1.3),
  milieu = sample(c("urbain", "rural"), 200, TRUE)
)
tableau_descriptif(donnees, vars = c("age", "revenu"), poids = "poids")
```

---

tableau_fgt *Tableau institutionnel des indices FGT*

Description

Genere un tableau des indices FGT formate selon les conventions des INS africains et de la Banque mondiale (EHCVM).

Usage

```
tableau_fgt(
  fgt_obj,
  format = c("tibble", "flextable", "excel"),
  chemin = NULL,
  titre = NULL,
  inclure_sous_groupes = TRUE
)
```

Arguments

`fgt_obj`	objet `saf_fgt` – Resultat de `calcul_fgt()`
`format`	character – Format de sortie : `"tibble"`, `"flextable"` ou `"excel"`. Defaut : "tibble"
`chemin`	character ou NULL – Chemin du fichier Excel. Si NULL, utilise `tempdir()`. Defaut : NULL
`titre`	character – Titre du tableau
`inclure_sous_groupes`	logical – Inclure les tableaux par sous-groupe. Defaut : TRUE

Value

Tibble, flextable ou chemin du fichier Excel

Examples

```
set.seed(42)
menages <- data.frame(
  depense_pc = c(rexp(70, rate = 1/150000), rexp(30, rate = 1/400000)),
  poids      = runif(100, 0.8, 1.2)
)
fgt <- calcul_fgt(menages, "depense_pc", 220000, poids = "poids")
tableau_fgt(fgt)
```

`tableau_odd` *Tableau de suivi des indicateurs ODD*

Description

Calcule et presente plusieurs indicateurs ODD dans un tableau de suivi formate selon les standards de reporting PARIS21/Nations Unies.

Usage

```
tableau_odd(
  resultats,
  pays = NULL,
  annee = NULL,
  cibles = NULL,
  format = c("tibble", "flextable", "excel"),
  chemin = NULL
)
```

Arguments

`resultats`	list – Liste nommee d'objets `saf_odd` (resultats de `odd_indicateur()`)
`pays`	character ou NULL – Nom du pays. Defaut : NULL
`annee`	integer ou NULL – Annee de reference. Defaut : NULL
`cibles`	list ou NULL – Liste nommee des cibles nationales par code ODD. Ex : `list("1.2.1" = 25, "7.1.1" = 80)`. Defaut : NULL
`format`	character – `"tibble"`, `"flextable"` ou `"excel"`. Defaut : "tibble"
`chemin`	character ou NULL – Chemin Excel. Defaut : NULL

Value

Tibble, flextable ou chemin Excel

Examples

```
set.seed(42)
menages <- data.frame(
  depense_pc  = c(rexp(70, 1/150000), rexp(30, 1/500000)),
  poids       = runif(100, 0.8, 1.2),
  electricite = sample(c(0L, 1L), 100, TRUE, c(0.45, 0.55)),
  eau_potable = sample(c(0L, 1L), 100, TRUE, c(0.35, 0.65))
)
r1 <- odd_indicateur(menages, "1.2.1",
                     var_depense = "depense_pc", seuil = 200000)
r2 <- odd_indicateur(menages, "7.1.1",
                     var_indicateur = "electricite")
tableau_odd(list(r1, r2), pays = "Centrafrique", annee = 2026L)
```

---

theme_ins *Thème ggplot2 officiel INS*

Description

Applique un thème graphique professionnel adapté aux publications officielles des Instituts Nationaux de Statistique africains. Inspiré des chartes graphiques AFRISTAT/PARIS21.

Usage

```
theme_ins(
  base_size = 11,
  base_family = "sans",
  couleur_fond = "white",
  grille = TRUE,
  grille_mineure = FALSE
)
```

Arguments

base_size	numeric — Taille de base de la police. Défaut : 11.
base_family	character — Famille de police. Défaut : "sans".
couleur_fond	character — Couleur de fond du panneau. Défaut : "white".
grille	logical — Afficher les lignes de grille. Défaut : TRUE.
grille_mineure	logical — Afficher la grille mineure. Défaut : FALSE.

Value

Un objet theme ggplot2.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
    ggplot2::geom_point() + theme_ins()
}
```

`tracer_flux_traitement` *Tracer le flux de traitement*

Description

Crée et maintient un journal horodaté des transformations appliquées à un dataset. Permet l'auditabilité complète du pipeline de traitement des données.

Usage

```
tracer_flux_traitement(data, action, journal = NULL, details = NULL)
```

Arguments

`data`	data.frame ou tibble — Données traitées
`action`	character — Description de l'action effectuée
`journal`	list ou NULL — Journal existant à compléter. Si NULL, crée un nouveau journal. Défaut : NULL.
`details`	list ou NULL — Détails supplémentaires à enregistrer (ex: paramètres utilisés). Défaut : NULL.

Value

Une liste mise à jour avec `$donnees` et `$journal`.

Examples

```
## Not run:
  donnees <- data.frame(id=1:3, val=c(10,20,30))
  e1 <- tracer_flux_traitement(donnees, action="Import")
  e2 <- tracer_flux_traitement(e1$donnees, action="Nettoyage", journal=e1$journal)
  print(e2$journal)

## End(Not run)
```

`valider_dictionnaire` *Valider la cohérence données / dictionnaire*

Description

Vérifie que les variables d'un dataset correspondent au dictionnaire fourni : présence des variables, types, plages de valeurs, modalités attendues. Produit un rapport de validation structuré avec un score de qualité global.

Usage

```
valider_dictionnaire(data, dictionnaire, stopper_si_critique = FALSE)
```

Arguments

`data`	data.frame ou tibble — Données à valider
`dictionnaire`	data.frame — Dictionnaire avec colonnes obligatoires : `nom_variable`, `type`. Colonnes optionnelles : `valeurs_valides`, `min`, `max`, `obligatoire`.
`stopper_si_critique`	logical — Arrêter l'exécution si des erreurs critiques sont détectées. Défaut : FALSE.

Value

Une liste avec :

`valide`	logical — TRUE si aucune erreur critique
`rapport`	data.frame — Détail des anomalies
`score_qualite`	numeric — Score de 0 à 100

Examples

```
## Not run:
  dico <- data.frame(
    nom_variable = c("age", "sexe", "region"),
    type         = c("numeric", "character", "character"),
    obligatoire  = c(TRUE, TRUE, FALSE)
  )
  resultat <- valider_dictionnaire(donnees, dico)
  if (!resultat$valide) print(resultat$rapport)

## End(Not run)
```

`valider_qualite_donnees` *Valider la qualité globale d'un jeu de données*

Description

Calcule un score de qualité composite (0-100) en évaluant la complétude, la cohérence, l'unicité et la plausibilité des données.

Usage

```
valider_qualite_donnees(data, seuil_na = 0.1, vars_cles = NULL)
```

Arguments

`data`	data.frame ou tibble — Données à évaluer
`seuil_na`	numeric — Seuil acceptable de valeurs manquantes. Défaut : 0.1.
`vars_cles`	character ou NULL — Variables clés pour le test d'unicité. Défaut : NULL.

Value

Une liste avec `score_global` et le détail par dimension.

Examples

```
## Not run:
  donnees <- data.frame(
    id_menage = 1:50,
    age       = c(sample(20:70, 45, replace = TRUE), rep(NA, 5)),
    revenu    = c(rnorm(48, 200000, 50000), NA, NA)
  )
  valider_qualite_donnees(donnees, vars_cles="id_menage")

## End(Not run)
```

Index