Package: statAfrikR (via r-universe)

October 25, 2026

Title Statistical Tools for African National Statistics Institutes

Version 0.2.1

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://statafrikr.org/, https://github.com/damoko2004/statAfrikR, https://damoko2004.github.io/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 shiny (>= 1.7.0), 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

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 2026-09-25 09:30:44 UTC

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

RemoteRef HEAD

RemoteSha 407a25cbab473fd0e69e3f9a62b8f2e881f96f57

Contents

accouchements_assistes . . . 5
analyse_regression . . . 6
analyse_spatiale . . . 7
analyser_non_reponse . . . 8
anemie . . . 9
anonymiser_donnees . . . 10
appliquer_ponderations . . . 11
autonomisation_femmes . . . 12
bonheur_declare . . . 14
calcul_deff . . . 15
calcul_fgt . . . 15
calcul_gini . . . 17
calcul_idh . . . 18
calcul_ipm . . . 19
calcul_ipm_national . . . 21
calculer_deflateur . . . 22
calibrer_poids . . . 23
carte_choroplethe . . . 24
carte_exporter . . . 26
carte_import . . . 27
carte_ipm . . . 27
carte_joindre . . . 28
carte_pauvrete . . . 29
carte_thematique . . . 30
carte_zones . . . 32
catalogue_indicateurs . . . 33
check_na . . . 34
check_types . . . 34
coherence_demo . . . 35

comparer_ipm . . . 36
comparer_pib . . . 37
compresser_package_diffusion . . . 38
confiance_institutions . . . 39
courbe_lorenz . . . 41
creer_design . . . 42
decomposer_fgt . . . 43
decomposer_inegalite . . . 44
decomposer_ipm . . . 45
decomposer_theil . . . 46
detecter_ecarts_geo . . . 46
emaciation . . . 47
emploi_informel . . . 48
emploi_vulnerable . . . 49
exporter_code_dashboard . . . 50
exporter_excel_ins . . . 51
exporter_graphique . . . 52
exporter_sdmx . . . 53
fusion_datasets . . . 55
generer_bulletin . . . 56
generer_metadonnees_ddi . . . 57
generer_rapport . . . 58
generer_rapport_enquete . . . 59
generer_rapport_odd . . . 60
graphique_barres . . . 62
graphique_fgt . . . 63
graphique_ipm . . . 64
graphique_tendance . . . 65
handicap_prevalence . . . 66
harmoniser_regions . . . 67
harmoniser_zones . . . 68
import_cspro . . . 69
import_csv . . . 70
import_excel . . . 71
import_kobo . . . 72
import_odk . . . 73
import_sas . . . 74
import_spss . . . 75
import_stata . . . 76
imputer_valeurs . . . 77
indice_atkinson . . . 78
indice_palma . . . 78
insuffisance_ponderale . . . 79
lancer_dashboard . . . 80
lister_templates . . . 81
mariage_precoce . . . 82
migrer_nomenclature . . . 83
mortalite_5ans . . . 84

myers . . . 85
nettoyer_libelles . . . 86
odd_catalogue . . . 87
odd_indicateur . . . 88
palette_ins . . . 90
parite_education . . . 90
part_quintile . . . 91
pauvrete_travail . . . 92
perception_economique . . . 93
privations_ressenties . . . 94
protection_sociale . . . 95
pyramide_age . . . 96
pyramide_ages . . . 97
rapport_harmonisation . . . 98
rapport_qualite_demo . . . 99
rapport_qualite_sondage . . . 100
ratio_masculinite . . . 101
recoder_variable . . . 102
retard_croissance . . . 103
saf_cameroun_departements . . . 104
saf_cedeao . . . 105
saf_cemac . . . 106
saf_eau . . . 106
saf_pays_afrique . . . 107
saf_rca_prefectures . . . 107
saf_sadc . . . 108
saf_senegal_regions . . . 108
saf_subdivisions_afrique . . . 109
sante_mentale . . . 109
satisfaction_vie . . . 110
sentiment_securite . . . 111
sous_emploi_temps . . . 112
standardiser_ages . . . 113
stat_descr . . . 114
suivre_revisions_pib . . . 115
supprimer_doublons . . . 116
tab_croisee . . . 117
table_concordance . . . 118
tableau_bien_etre_subjectif . . . 119
tableau_bord_pib . . . 120
tableau_croise_ins . . . 121
tableau_descriptif . . . 122
tableau_fgt . . . 123
tableau_genre . . . 124
tableau_inegalites . . . 125
tableau_ipm . . . 126
tableau_marche_travail . . . 127
tableau_odd . . . 128

tableau_sante . . . 129
taux_activite . . . 130
taux_croissance . . . 131
taux_emploi . . . 132
taux_sous_utilisation . . . 133
theme_ins . . . 134
tracer_flux_traitement . . . 135
travail_enfants . . . 136
vaccination . . . 137
valider_coherence_geo . . . 138
valider_dictionnaire . . . 139
valider_poids . . . 140
valider_qualite_donnees . . . 140
valider_statistique_ins . . . 141
violence_basee_genre . . . 143
whipple . . . 144

Index **145**

`accouchements_assistes` *Calculer le taux d'accouchements assistes*

Description

Proportion des naissances assistees par un personnel de sante qualifie (medecin, sage-femme, infirmier). ODD 3.1.2.

Usage

```
accouchements_assistes(donnees, var_assiste, poids = NULL, sous_groupes = NULL)
```

Arguments

`donnees`	data.frame – Donnees naissances recentes
`var_assiste`	character – Variable 0/1 indiquant un accouchement assiste par personnel qualifie
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec taux global et par sous-groupe

Examples

```
set.seed(42)
n <- 400
naissances <- data.frame(
  assiste = rbinom(n, 1, 0.65),
  milieu  = sample(c("urbain","rural"), n, TRUE),
  region  = sample(paste0("R", 1:5), n, TRUE),
  poids   = runif(n, 0.8, 1.3)
)
accouchements_assistes(naissances, "assiste",
                       poids = "poids",
                       sous_groupes = c("milieu","region"))
```

---

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)
```

`analyser_non_reponse` *Analyser la non-reponse*

Description

Calcule les taux de non-reponse par strate et teste les biais de non-reponse potentiels par comparaison des repondants et non-repondants sur les variables disponibles.

Usage

```
analyser_non_reponse(
  donnees,
  var_reponse,
  var_strate = NULL,
  vars_biais = NULL
)
```

Arguments

`donnees`	data.frame – Donnees avec variable reponse
`var_reponse`	character – Variable 0/1 : a repondu (1) ou non-repondant (0)
`var_strate`	character ou NULL – Variable de stratification. Defaut : NULL
`vars_biais`	character ou NULL – Variables pour test de biais (disponibles pour tous : repondants + non-repondants). Defaut : NULL

Value

Un tibble avec taux de reponse et statistiques de biais

Examples

```
set.seed(42)
n <- 400
donnees <- data.frame(
  repondu = rbinom(n, 1, 0.88),
  strate  = sample(c("Urbain","Rural"), n, TRUE),
  age_cm  = sample(25:65, n, TRUE),
  taille_men = sample(3:10, n, TRUE)
)
analyser_non_reponse(donnees, "repondu",
                     var_strate = "strate",
                     vars_biais = c("age_cm","taille_men"))
```

`anemie` *Calculer la prevalence de l'anemie*

Description

Calcule la prevalence de l'anemie selon les seuils OMS (hemoglobine en g/dL). Distingue les niveaux legere, moderee, severe.

Usage

```
anemie(
  donnees,
  var_hemoglobine,
  var_groupe_cible = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees menages ou individus
`var_hemoglobine`	character – Taux d'hemoglobine en g/dL
`var_groupe_cible`	character ou NULL – Groupe cible : "femmes_15_49", "enfants_6_59", "enceintes", "tous". Defaut : NULL (tous)
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec prevalence par niveau d'anemie

Examples

```
set.seed(42)
n <- 400
femmes <- data.frame(
  hemoglobine = rnorm(n, 11.5, 2.1),
  milieu      = sample(c("urbain","rural"), n, TRUE),
  poids       = runif(n, 0.8, 1.3)
)
anemie(femmes, var_hemoglobine = "hemoglobine", poids = "poids")
```

`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,
  seuil_cellule = 5L,
  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.
seuil_cellule	integer – Seuil de confidentialite : les cellules avec effectif < seuil_cellule declenchent un avertissement. Defaut : 5L. Configurable selon la politique de l'INS.
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")
}
```

---

autonomisation_femmes *Calculer l'indice d'autonomisation des femmes*

Description

Calcule un indice composite d'autonomisation des femmes base sur trois dimensions : prises de decision, acces aux ressources et mobilite. Conforme a la methodologie DHS/MICS.

Usage

```
autonomisation_femmes(
  donnees,
  vars_decision = NULL,
  vars_ressources = NULL,
  vars_mobilite = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees femmes
vars_decision	character ou NULL – Variables 0/1 de prise de decision (ex: decisions achat menage, sante, visites). Defaut : NULL
vars_ressources	character ou NULL – Variables 0/1 d'acces aux ressources (ex: compte bancaire, revenu propre, terre). Defaut : NULL
vars_mobilite	character ou NULL – Variables 0/1 de mobilite (ex: libre de se deplacer seule, rendre visite, marche). Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_genre`

Examples

```
set.seed(42)
n <- 400
femmes <- data.frame(
  dec_achat   = rbinom(n, 1, 0.55),
  dec_sante   = rbinom(n, 1, 0.48),
  compte_ban  = rbinom(n, 1, 0.32),
  rev_propre  = rbinom(n, 1, 0.41),
  libre_dep   = rbinom(n, 1, 0.62),
  milieu      = sample(c("urbain","rural"), n, TRUE),
  poids       = runif(n, 0.8, 1.3)
)
autonomisation_femmes(femmes,
  vars_decision   = c("dec_achat","dec_sante"),
  vars_ressources = c("compte_ban","rev_propre"),
  vars_mobilite   = c("libre_dep"),
  poids = "poids", sous_groupes = "milieu")
```

`bonheur_declare` *Calculer la part de population se declarant heureuse*

Description

Proportion d'individus se declarant tres heureux ou heureux. Variable categorielle recodee en indicateur 0/1.

Usage

```
bonheur_declare(
  donnees,
  var_bonheur,
  codes_heureux = c(1, 2),
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees individuelles
`var_bonheur`	character – Variable de bonheur. Valeurs attendues : 1="Tres heureux", 2="Heureux", 3="Pas tres heureux", 4="Pas du tout heureux" OU variable 0/1 deja binaire.
`codes_heureux`	numeric ou character – Codes consideres comme "heureux" (les autres = "pas heureux"). Defaut : c(1, 2)
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_subjectif`

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  bonheur  = sample(1:4, n, TRUE, prob=c(0.20,0.45,0.25,0.10)),
  milieu   = sample(c("urbain","rural"), n, TRUE),
  poids    = runif(n, 0.8, 1.3)
)
bonheur_declare(individus, "bonheur", codes_heureux=c(1,2),
                poids="poids", sous_groupes="milieu")
```

calcul_deff *Calculer l'effet de plan (DEFF)*

Description

Calcule le Design Effect (DEFF) pour les variables d'interet. DEFF = Variance estimee sous le plan complexe / Variance sous SRS. Permet de calculer la taille effective de l'echantillon.

Usage

```
calcul_deff(saf_design, variables, type = c("proportion", "moyenne"))
```

Arguments

saf_design	saf_design – Objet cree par creer_design()
variables	character – Variables pour lesquelles calculer le DEFF
type	character – "proportion" ou "moyenne". Defaut : "proportion"

Value

Un tibble avec DEFF et taille effective par variable

Examples

```
set.seed(42)
n <- 300
donnees <- data.frame(
  poids_sond = runif(n, 1800, 3500),
  strate     = sample(c("Urbain","Rural"), n, TRUE),
  grappe     = sample(1:20, n, TRUE),
  pauvre     = rbinom(n, 1, 0.45),
  revenu     = pmax(0, rnorm(n, 180000, 90000))
)
design <- suppressMessages(
  creer_design(donnees,"poids_sond",var_strate="strate",var_grappe="grappe"))
calcul_deff(design, variables=c("pauvre","revenu"))
```

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 IC 95%. Defaut : TRUE. Methode : Wilson (echantillon simple) ou Taylor via `creer_design()` pour plans complexes. Pour publications officielles INS, privilegier la variance de Taylor : `calcul_fgt(creer_design(donnees, ...), ...)`. 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_gini *Calculer le coefficient de Gini*

Description

Calcule le coefficient de Gini avec intervalle de confiance par bootstrap. Le Gini mesure l'inegalite de distribution d'une variable de revenu ou de consommation (0 = egalite parfaite, 1 = inegalite maximale).

Usage

```
calcul_gini(
  donnees,
  var_revenu,
  poids = NULL,
  ic = FALSE,
  n_bootstrap = 200L,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu ou de consommation (valeurs strictement positives)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
ic	logical – Calculer l'IC 95% par bootstrap. Defaut : FALSE
n_bootstrap	integer – Replications bootstrap. Defaut : 200L
sous_groupes	character ou NULL – Variables de sous-groupe pour decomposition. Defaut : NULL

Value

Un objet de classe `saf_gini`

Examples

```
set.seed(42)
n <- 300
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  milieu  = sample(c("urbain","rural"), n, TRUE),
  poids   = runif(n, 0.8, 1.3)
)
res <- calcul_gini(menages, "depense", poids = "poids")
print(res)
```

`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 Pauvrete Multidimensionnelle (IPM)*

Description

Calcule l'IPM standard PNUD/OPHI selon la methode Alkire-Foster (2011). L'IPM = H x A, ou H est l'incidence (proportion de menages multidimensionnellement pauvres) et A l'intensite moyenne de privation parmi les pauvres.

Usage

```
calcul_ipm(
  donnees,
  var_nutrition = NULL,
  var_mortalite_inf = NULL,
  var_annees_scol = NULL,
  var_scolarisation = NULL,
  var_combustible = NULL,
  var_assainissement = NULL,
  var_eau = NULL,
  var_electricite = NULL,
  var_logement = NULL,
  var_actifs = NULL,
  poids = NULL,
  seuil_k = 1/3,
  ic = FALSE,
  n_bootstrap = 200L
)
```

Arguments

donnees	data.frame – Donnees menages
var_nutrition	character ou NULL – Variable nutrition (1 = prive, 0 = non prive). Defaut : NULL

`var_mortalite_inf`
: character ou NULL – Deces d'enfant dans le menage (1 = oui). Defaut : NULL

`var_annees_scol`
: character ou NULL – Aucun membre >= 6 ans ayant complete 6 ans de scolarite (1 = prive). Defaut : NULL

`var_scolarisation`
: character ou NULL – Enfant non scolarise dans le menage (1 = prive). Defaut : NULL

`var_combustible`
: character ou NULL – Combustible solide pour la cuisine (1 = prive). Defaut : NULL

`var_assainissement`
: character ou NULL – Assainissement non ameliore (1 = prive). Defaut : NULL

`var_eau`
: character ou NULL – Eau non potable (1 = prive). Defaut : NULL

`var_electricite`
: character ou NULL – Sans electricite (1 = prive). Defaut : NULL

`var_logement`
: character ou NULL – Logement inadequat (1 = prive). Defaut : NULL

`var_actifs`
: character ou NULL – Pas d'actifs de base (1 = prive). Defaut : NULL

`poids`
: character ou NULL – Variable de ponderation. Defaut : NULL (poids egaux)

`seuil_k`
: numeric – Seuil de pauvrete multidimensionnelle (proportion de privations). Defaut : 1/3

`ic`
: logical – Calculer les intervalles de confiance (bootstrap). Defaut : FALSE

`n_bootstrap`
: integer – Nombre de replications bootstrap. Defaut : 200L

Value

Un objet de classe `saf_ipm` contenant : `IPM`, `H`, `A`, `contributions`, `n_obs`, `seuil_k`, `methode`

References

Alkire, S. & Foster, J. (2011). Counting and multidimensional poverty measurement. *Journal of Public Economics*, 95(7-8), 476-487. doi:10.1016/j.jpubeco.2010.11.006

Examples

```
set.seed(42)
n <- 200
menages <- data.frame(
  nutrition      = rbinom(n, 1, 0.35),
  mortalite_inf  = rbinom(n, 1, 0.12),
  annees_scol    = rbinom(n, 1, 0.40),
  scolarisation  = rbinom(n, 1, 0.28),
  combustible    = rbinom(n, 1, 0.55),
  assainissement = rbinom(n, 1, 0.48),
  eau            = rbinom(n, 1, 0.38),
  electricite    = rbinom(n, 1, 0.62),
  logement       = rbinom(n, 1, 0.42),
  actifs         = rbinom(n, 1, 0.30),
```

```
  poids          = runif(n, 0.8, 1.3)
)
res <- calcul_ipm(menages,
  var_nutrition      = "nutrition",
  var_mortalite_inf  = "mortalite_inf",
  var_annees_scol    = "annees_scol",
  var_scolarisation  = "scolarisation",
  var_combustible    = "combustible",
  var_assainissement = "assainissement",
  var_eau            = "eau",
  var_electricite    = "electricite",
  var_logement       = "logement",
  var_actifs         = "actifs",
  poids              = "poids"
)
print(res)
```

`calcul_ipm_national` *Calculer un IPM national personnalise*

Description

Calcule un IPM national adapte aux realites locales (ODD 1.2.2). L'INS peut choisir ses indicateurs, leurs poids et le seuil k.

Usage

```
calcul_ipm_national(
  donnees,
  indicateurs,
  poids = NULL,
  seuil_k = 1/3,
  ic = FALSE,
  n_bootstrap = 200L
)
```

Arguments

`donnees`	data.frame – Donnees menages
`indicateurs`	list – Liste nommee definissant les indicateurs : chaque element contient `var` (nom de variable), `poids` (poids numerique) et `dimension` (nom de la dimension).
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`seuil_k`	numeric – Seuil de pauvrete. Defaut : 1/3
`ic`	logical – Intervalles de confiance bootstrap. Defaut : FALSE
`n_bootstrap`	integer – Replications bootstrap. Defaut : 200L

Value

Un objet `saf_ipm`

Examples

```
set.seed(42)
n <- 150
menages <- data.frame(
  eau        = rbinom(n, 1, 0.45),
  electricite = rbinom(n, 1, 0.60),
  scol       = rbinom(n, 1, 0.35),
  sante      = rbinom(n, 1, 0.25),
  poids      = runif(n, 0.8, 1.3)
)
indics <- list(
  acces_eau   = list(var = "eau",        poids = 0.25,
                     dimension = "Conditions de vie"),
  acces_elec  = list(var = "electricite", poids = 0.25,
                     dimension = "Conditions de vie"),
  education   = list(var = "scol",       poids = 0.30,
                     dimension = "Education"),
  sante_base  = list(var = "sante",      poids = 0.20,
                     dimension = "Sante")
)
res <- calcul_ipm_national(menages, indicateurs = indics,
                           poids = "poids", seuil_k = 1/3)
print(res)
```

`calculer_deflateur` *Calculer et appliquer le deflateur du PIB*

Description

Convertit les valeurs en prix courants en prix constants (volume) en utilisant le deflateur implicite du PIB ou un indice de prix fourni. Base annee referencee configurable.

Usage

```
calculer_deflateur(
  donnees,
  var_pib_courant,
  var_deflateur = NULL,
  var_ipc = NULL,
  annee_base = 2015L,
  var_annee = "annee"
)
```

Arguments

donnees	data.frame – Donnees avec PIB courant et deflateur
var_pib_courant	character – Variable PIB en prix courants
var_deflateur	character ou NULL – Variable deflateur (base 100). Si NULL, calcule depuis var_ipc. Defaut : NULL
var_ipc	character ou NULL – Indice des prix a la consommation (si deflateur non fourni). Defaut : NULL
annee_base	integer – Annee de base du deflateur. Defaut : 2015L
var_annee	character – Variable annee. Defaut : "annee"

Value

Un tibble avec PIB courant, deflateur et PIB constant

Examples

```
comptes <- data.frame(
  annee        = 2015:2022,
  pib_courant  = c(2100,2180,2250,2310,2195,2280,2380,2450),
  deflateur    = c(100,103.2,106.8,110.5,109.2,113.5,118.2,122.8)
)
calculer_deflateur(comptes, "pib_courant",
                   var_deflateur="deflateur",
                   annee_base=2015L)
```

---

calibrer_poids *Calibrer les poids de sondage*

Description

Ajuste (cale) les poids de sondage pour que les estimations correspondent aux totaux de population connus (rake / calage generalise GREG). Methode post-stratification generalisee.

Usage

```
calibrer_poids(
  saf_design,
  marges,
  var_calibrage,
  methode = c("raking", "post-stratification"),
  tolerance = 1e-06,
  max_iter = 50L
)
```

Arguments

saf_design	saf_design – Objet design a calibrer
marges	list – Liste nommee des totaux de population par variable. Exemple : list(sexe=c(H=1200000, F=1350000), milieu=c(urbain=1500000, rural=1050000))
var_calibrage	character – Variables de calage (doivent etre dans les donnees). Ex : c("sexe", "milieu")
methode	character – "raking" ou "post-stratification". Defaut : "raking"
tolerance	numeric – Tolerance de convergence. Defaut : 1e-6
max_iter	integer – Nombre maximum d'iterations. Defaut : 50L

Value

Un objet saf_design avec poids calibres

Examples

```
set.seed(42)
n <- 300
donnees <- data.frame(
  poids_sond = runif(n, 1800, 3500),
  strate     = sample(c("Urbain","Rural"), n, TRUE),
  grappe     = sample(1:20, n, TRUE),
  sexe       = sample(c("H","F"), n, TRUE),
  milieu     = sample(c("urbain","rural"), n, TRUE)
)
design <- suppressMessages(
  creer_design(donnees,"poids_sond",var_strate="strate",var_grappe="grappe"))
marges <- list(
  sexe   = c(H=500000, F=550000),
  milieu = c(urbain=600000, rural=450000)
)
calibrer_poids(design, marges, var_calibrage=c("sexe","milieu"))
```

---

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_ipm *Carte des privations IPM par zone*

Description

Cartographie l'IPM ou une composante (H, A, ou un indicateur de privation) par zone geographique. Necessite un objet sf et les resultats de decomposer_ipm().

Usage

```
carte_ipm(
  decomp_ipm,
  sf_obj,
  cle_sf,
  cle_decomp,
  var = c("IPM", "H", "A"),
  titre = NULL,
  source = NULL
)
```

Arguments

decomp_ipm	tibble – Resultat de decomposer_ipm()
sf_obj	sf – Fond de carte (depuis carte_zones())
cle_sf	character – Variable cle dans sf_obj
cle_decomp	character – Variable cle dans decomp_ipm
var	character – Variable a cartographier : "IPM", "H" ou "A". Defaut : "IPM"
titre	character ou NULL – Titre. Defaut : NULL
source	character ou NULL – Source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
## Not run:
  rca <- carte_zones("rca")
  res <- calcul_ipm(menages_rca, ...)
  decomp <- decomposer_ipm(res, menages_rca, "prefecture")
  carte_ipm(decomp, rca, cle_sf="prefecture",
            cle_decomp="prefecture")

## 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"))
```

`catalogue_indicateurs` *Consulter le catalogue des indicateurs statistiques*

Description

Retourne la fiche methodologique complete d'un indicateur : definition, formule, standard de reference, ODD, methode de variance, seuils de qualite et statut INS-ready. Conforme GSBPM 5.2 / UN NQAF.

Usage

```
catalogue_indicateurs(
  code = NULL,
  domaine = NULL,
  format = c("tableau", "liste", "complet")
)
```

Arguments

code	character ou NULL – Code de l'indicateur. Si NULL, retourne la liste de tous les indicateurs disponibles. Codes disponibles : "FGT0", "FGT1", "FGT2", "IPM", "GINI", "STUNTING", "TAUX_ACTIVITE", "CHOMAGE", "INFORMEL", "MARIAGE_PRECOCE", "SATISFACTION_VIE", "PIB_CROISSANCE", "WHIPPLE"
domaine	character ou NULL – Filtrer par domaine. Ex : "Pauvrete monetaire", "Emploi", "Sante et nutrition". Defaut : NULL (tous les domaines)
format	character – Format de sortie : "liste" (objet R complet), "tableau" (tibble synthetique), "complet" (impression detaillee). Defaut : "tableau"

Value

Un tibble, une liste ou un affichage selon le format choisi

Examples

```
# Lister tous les indicateurs
catalogue_indicateurs()

# Fiche methodologique d'un indicateur
catalogue_indicateurs("FGT0")
catalogue_indicateurs("IPM")
catalogue_indicateurs("GINI")

# Filtrer par domaine
catalogue_indicateurs(domaine = "Emploi")

# Format complet avec impression
catalogue_indicateurs("FGT0", format = "complet")
```

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)
```

---

coherence_demo *Verifier la coherence demographique des donnees*

Description

Detecte les incoherences courantes dans les donnees demographiques : age mere < age enfant, dates d'evenements incoherentes, ages impossibles, sex-ratio aberrants, etc.

Usage

```
coherence_demo(
  donnees,
  var_age = NULL,
  var_age_mere = NULL,
  var_age_enfant = NULL,
  var_age_deces = NULL,
  var_sexe = NULL,
  age_max_plausible = 120L
)
```

Arguments

donnees	data.frame – Donnees individuelles ou menages
var_age	character ou NULL – Variable age. Defaut : NULL
var_age_mere	character ou NULL – Variable age de la mere. Defaut : NULL
var_age_enfant	character ou NULL – Variable age de l'enfant. Defaut : NULL
var_age_deces	character ou NULL – Variable age au deces. Defaut : NULL
var_sexe	character ou NULL – Variable sexe. Defaut : NULL
age_max_plausible	integer – Age maximum plausible. Defaut : 120L

Value

Un tibble des anomalies detectees avec leur nombre

Examples

```
set.seed(42)
n <- 500
menages <- data.frame(
  age         = c(sample(0:100, n-5, TRUE), rep(-1, 3), rep(130, 2)),
  age_mere    = sample(15:60, n, TRUE),
  age_enfant  = sample(0:30,  n, TRUE),
  age_deces   = c(sample(1:90, n-10, TRUE), rep(NA, 10)),
  sexe        = sample(c("H","F","X"), n, TRUE, prob=c(0.49,0.49,0.02))
)
coherence_demo(menages,
  var_age="age", var_age_mere="age_mere",
  var_age_enfant="age_enfant",
  var_sexe="sexe")
```

---

comparer_ipm *Comparer l'IPM global et l'IPM national*

Description

Compare les resultats du MPI global PNUD/OPHI et d'un IPM national. Produit un tableau des ecarts et un graphique.

Usage

```
comparer_ipm(
  ipm_global,
  ipm_national,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y"))
)
```

Arguments

ipm_global	saf_ipm – Resultat IPM global (calcul_ipm())
ipm_national	saf_ipm – Resultat IPM national (calcul_ipm_national())
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante

Value

Un tibble comparatif + graphique (invisible)

Examples

```
set.seed(42)
n <- 200
men <- data.frame(
  nutr = rbinom(n,1,0.35), elec = rbinom(n,1,0.6),
  eau  = rbinom(n,1,0.4),  scol = rbinom(n,1,0.3),
  mort = rbinom(n,1,0.12), comb = rbinom(n,1,0.5),
  poids = runif(n,0.8,1.3)
)
ipm_g <- calcul_ipm(men, var_nutrition="nutr",
  var_electricite="elec", var_eau="eau",
  var_scolarisation="scol", var_mortalite_inf="mort",
  var_combustible="comb", poids="poids")
ipm_n <- calcul_ipm_national(men,
  indicateurs=list(
    eau  = list(var="eau",  poids=0.4, dimension="Vie"),
    elec = list(var="elec", poids=0.35, dimension="Vie"),
    scol = list(var="scol", poids=0.25, dimension="Education")
  ), poids="poids")
comparer_ipm(ipm_g, ipm_n, pays="Centrafrique", annee=2026L)
```

---

comparer_pib *Comparer le PIB entre sources (INS vs UNSD vs BM vs FMI)*

Description

Compare les estimations du PIB publiees par l'INS et les institutions internationales. Identifie les ecarts, leurs causes probables et produit un tableau de diagnostic. Outil cle pour renforcer la maitrise du narratif statistique national.

Usage

```
comparer_pib(
  pib_ins,
  pib_intl,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  unite = "Mds FCFA",
  methode_ins = NULL,
  seuil_alerte = 3
)
```

Arguments

pib_ins	numeric – PIB publie par l'INS (en milliards FCFA ou USD selon l'unite choisie)
pib_intl	named numeric – PIB des institutions internationales. Vecteur nomme : ex : c(UNSD=245.2, BM=241.8, FMI=248.5)

pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee de reference
unite	character – Unite monetaire. Defaut : "Mds FCFA"
methode_ins	character ou NULL – Methode INS : "production", "depenses", "revenus". Defaut : NULL
seuil_alerte	numeric – Seuil d'ecart en % declenchant une alerte. Defaut : 3.0

Value

Un objet de classe saf_pib

Examples

```
res <- comparer_pib(
  pib_ins  = 2450.5,
  pib_intl = c(UNSD=2380.2, BM=2412.8, FMI=2398.5),
  pays     = "Cameroun",
  annee    = 2023L,
  unite    = "Mds FCFA"
)
print(res)
```

---

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)
```

---

confiance_institutions

Calculer la confiance dans les institutions

Description

Proportion de la population faisant confiance aux institutions publiques (gouvernement, justice, police, parlement). Indicateur de gouvernance percu. Methodologie Afrobarometer.

Usage

```
confiance_institutions(
  donnees,
  vars_institutions,
  codes_confiance = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles
vars_institutions	named character – Vecteur nomme : institution -> variable 0/1 (ou codes_confiance si categorielle). Ex : c(Gouvernement="conf_gouv", Justice="conf_just")
codes_confiance	numeric ou NULL – Codes indiquant confiance. Defaut : NULL (variables deja binaires)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec taux de confiance par institution

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  conf_gouv  = rbinom(n, 1, 0.35),
  conf_just  = rbinom(n, 1, 0.42),
  conf_pol   = rbinom(n, 1, 0.38),
  milieu     = sample(c("urbain","rural"), n, TRUE),
  poids      = runif(n, 0.8, 1.3)
)
confiance_institutions(individus,
  vars_institutions = c(Gouvernement="conf_gouv",
                        Justice="conf_just",
                        Police="conf_pol"),
  poids="poids", sous_groupes="milieu")
```

courbe_lorenz *Tracer la courbe de Lorenz*

Description

Produit la courbe de Lorenz institutionnelle montrant la distribution cumulee des revenus. L'aire entre la diagonale et la courbe est egale a Gini/2.

Usage

```
courbe_lorenz(
  donnees,
  var_revenu,
  poids = NULL,
  var_groupe = NULL,
  titre = NULL,
  source = NULL
)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
poids	character ou NULL – Variable de ponderation. Defaut : NULL
var_groupe	character ou NULL – Variable de groupe pour comparaison. Defaut : NULL
titre	character ou NULL – Titre. Defaut : NULL
source	character ou NULL – Source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(150, 1/150000), rexp(50, 1/500000)),
  milieu  = sample(c("urbain","rural"), 200, TRUE),
  poids   = runif(200, 0.8, 1.3)
)
courbe_lorenz(menages, "depense", poids = "poids")
```

`creer_design` *Creer un objet design d'enquete complexe*

Description

Cree un objet de design d'enquete en precisant les strates, grappes, poids de sondage et la correction de population finie (FPC). Compatible avec les enquetes EHCVM, DHS, MICS, EFT, RGPH.

Usage

```
creer_design(
  donnees,
  var_poids,
  var_strate = NULL,
  var_grappe = NULL,
  var_fpc = NULL,
  type = c("stratifie_grappes", "stratifie", "en_grappes", "simple")
)
```

Arguments

`donnees`	data.frame – Donnees de l'enquete
`var_poids`	character – Variable de poids de sondage
`var_strate`	character ou NULL – Variable de stratification. Defaut : NULL
`var_grappe`	character ou NULL – Variable d'unites primaires de sondage (grappes / UPS). Defaut : NULL
`var_fpc`	character ou NULL – Variable de correction de population finie (taille de la strate dans la population). Defaut : NULL
`type`	character – Type de design : "stratifie", "en_grappes", "stratifie_grappes", "simple". Defaut : "stratifie_grappes"

Value

Un objet de classe `saf_design` encapsulant un objet `survey::svydesign`

Examples

```
set.seed(42)
n <- 300
donnees <- data.frame(
  poids_sond = runif(n, 1800, 3500),
  strate     = sample(c("Urbain","Rural","Semi-urbain"), n, TRUE),
  grappe     = sample(1:30, n, TRUE),
  revenu     = pmax(0, rnorm(n, 180000, 90000)),
  pauvre     = rbinom(n, 1, 0.45)
)
design <- creer_design(donnees, "poids_sond",
```

```
                          var_strate="strate", var_grappe="grappe")
print(design)
```

`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)
```

`decomposer_ipm` *Decomposer l'IPM par sous-groupe*

Description

Decompose l'IPM par region, milieu ou sexe du chef de menage. Retourne H, A et IPM par groupe, ainsi que la contribution de chaque groupe a l'IPM national.

Usage

```
decomposer_ipm(res_ipm, donnees, var_groupe, poids = NULL)
```

Arguments

`res_ipm`	saf_ipm – Resultat de `calcul_ipm()` ou `calcul_ipm_national()`
`donnees`	data.frame – Donnees originales (meme ordre que lors du calcul de l'IPM)
`var_groupe`	character – Variable de sous-groupe (ex : "region", "milieu", "sexe_cm")
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un tibble avec H, A, IPM, n_obs et contribution par groupe

Examples

```
set.seed(42)
n <- 300
menages <- data.frame(
  nutrition      = rbinom(n, 1, 0.35),
  electricite    = rbinom(n, 1, 0.60),
  eau            = rbinom(n, 1, 0.40),
  milieu         = sample(c("urbain","rural"), n, TRUE),
  poids          = runif(n, 0.8, 1.3)
)
res <- calcul_ipm(menages,
  var_nutrition = "nutrition", var_electricite = "electricite",
  var_eau = "eau", poids = "poids")
decomposer_ipm(res, menages, var_groupe = "milieu", poids = "poids")
```

decomposer_theil *Decomposer les inegalites selon l'indice de Theil*

Description

Decompose l'inegalite totale en composantes inter-groupe et intra-groupe selon l'indice de Theil T (entropie generalisee GE(1)). Indispensable pour identifier les sources spatiales ou sociales de l'inegalite.

Usage

```
decomposer_theil(donnees, var_revenu, var_groupe, poids = NULL)

theil(donnees, var_revenu, var_groupe, poids = NULL)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
var_groupe	character – Variable de groupe (region, milieu, sexe_cm)
poids	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un tibble avec Theil total, inter-groupe, intra-groupe et contribution de chaque groupe

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  milieu  = sample(c("urbain","rural"), 300, TRUE),
  poids   = runif(300, 0.8, 1.3)
)
decomposer_theil(menages, "depense", "milieu", poids = "poids")
```

detecter_ecarts_geo *Detecter les zones non appariees entre deux sources*

Description

Identifie les zones presentes dans les donnees mais absentes de la reference geographique. Propose des suggestions d'appariement par score de similarite (Jaro-Winkler). Essentiel avant toute jointure cartographique.

Usage

```
detecter_ecarts_geo(
  donnees,
  var_zone,
  reference,
  var_ref = "nom_local",
  n_suggestions = 3L
)
```

Arguments

donnees	data.frame – Donnees avec zones a verifier
var_zone	character – Variable zone dans les donnees
reference	data.frame ou saf_concordance – Reference geographique
var_ref	character – Variable zone dans la reference. Defaut : "nom_local"
n_suggestions	integer – Nombre de suggestions par zone. Defaut : 3L

Value

Un tibble avec zones non appariees et suggestions

Examples

```
donnees <- data.frame(
  region = c("Bangui","Ombella-Mpoko","Kemo","Zone inconnue"),
  valeur = c(100, 200, 150, 80),
  stringsAsFactors = FALSE
)
reference <- data.frame(
  nom_local = c("Bangui","Ombella-M'Poko","Kemo",
                "Bamingui-Bangoran","Mbomou"),
  stringsAsFactors = FALSE
)
detecter_ecarts_geo(donnees, "region", reference, "nom_local")
```

---

emaciation *Calculer le taux d'emaciation (wasting)*

Description

Prevalence de l'emaciation : poids-pour-taille < -2 ecarts-types (standards OMS 2006). Indicateur de malnutrition aigue.

Usage

```
emaciation(
  donnees,
  var_poids_taille_z,
  poids = NULL,
  sous_groupes = NULL,
  seuil = -2
)
```

Arguments

donnees	data.frame – Donnees enfants < 5 ans
var_poids_taille_z	character – Score Z poids-pour-taille (WHZ)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL
seuil	numeric – Seuil score Z. Defaut : -2

Value

Un objet de classe `saf_anthropo`

Examples

```
set.seed(42)
n <- 400
enfants <- data.frame(
  whz  = rnorm(n, -0.8, 1.1),
  poids = runif(n, 0.8, 1.3)
)
emaciation(enfants, var_poids_taille_z = "whz", poids = "poids")
```

---

emploi_informel *Calculer le taux d'emploi informel*

Description

Calcule la part de l'emploi informel selon la definition OIT 2013. L'emploi informel comprend les travailleurs sans contrat ecrit, sans protection sociale et sans conges payes.

Usage

```
emploi_informel(
  donnees,
  var_informel,
  var_statut = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees employes
var_informel	character – Variable 0/1 : emploi informel
var_statut	character ou NULL – Variable de statut d'emploi (pour decomposition). Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
n <- 400
employes <- data.frame(
  informel = rbinom(n, 1, 0.72),
  secteur  = sample(c("agriculture","services","industrie"), n, TRUE),
  sexe     = sample(c("H","F"), n, TRUE),
  poids    = runif(n, 0.8, 1.3)
)
emploi_informel(employes, "informel", poids="poids",
                sous_groupes=c("secteur","sexe"))
```

---

emploi_vulnerable *Calculer le taux d'emploi vulnerable*

Description

Part des actifs occupes en situation d'emploi vulnerable : travailleurs familiaux non remuneres et travailleurs independants precaires. ODD 8.3.1.

Usage

```
emploi_vulnerable(donnees, var_vulnerable, poids = NULL, sous_groupes = NULL)
```

Arguments

donnees	data.frame – Donnees actifs occupes
var_vulnerable	character – Variable 0/1 : emploi vulnerable
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
n <- 400
employes <- data.frame(
  vulnerable = rbinom(n, 1, 0.55),
  sexe       = sample(c("H","F"), n, TRUE),
  poids      = runif(n, 0.8, 1.3)
)
emploi_vulnerable(employes, "vulnerable", poids="poids",
                  sous_groupes="sexe")
```

---

exporter_code_dashboard

Exporter le code source complet du dashboard

Description

Genere un fichier R autonome et entierement commente que l'agent INS peut ouvrir, modifier et relancer librement. Contient l'integralite du code du dashboard avec des sections clairement identifiees pour la personnalisation : titres, couleurs, indicateurs, filtres, sections.

Usage

```
exporter_code_dashboard(
  chemin = "dashboard_statAfrikR.R",
  pays = "Pays",
  titre = NULL,
  sous_titre = NULL,
  var_poids = NULL,
  var_region = NULL,
  var_milieu = NULL
)
```

Arguments

chemin	character – Chemin du fichier R a generer. Defaut : "dashboard_statAfrikR.R"
pays	character – Nom du pays pre-rempli. Defaut : "Pays"
titre	character ou NULL – Titre pre-rempli. Defaut : NULL
sous_titre	character ou NULL – Sous-titre pre-rempli. Defaut : NULL
var_poids	character ou NULL – Variable poids pre-remplie. Defaut : NULL
var_region	character ou NULL – Variable region pre-remplie. Defaut : NULL
var_milieu	character ou NULL – Variable milieu pre-remplie. Defaut : NULL

Value

Invisible : chemin du fichier genere

Examples

```
## Not run:
  # Generer le code source personnalisable
  exporter_code_dashboard(
    chemin     = "mon_dashboard_rca.R",
    pays       = "Republique Centrafricaine",
    titre      = "Tableau de bord - Enquete EHCVM 2022",
    sous_titre = "Indicateurs bien-etre - INS RCA",
    var_poids  = "poids_sondage",
    var_region = "prefecture",
    var_milieu = "milieu"
  )
  # Ouvrir le fichier genere dans RStudio
  file.edit("mon_dashboard_rca.R")

## 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_ipm *Graphique des contributions IPM*

Description

Produit un graphique en barres des contributions de chaque indicateur et chaque dimension a l'IPM. Retourne un objet ggplot2.

Usage

```
graphique_ipm(
  res_ipm,
  type = c("indicateurs", "dimensions"),
  titre = NULL,
  source = NULL
)
```

Arguments

res_ipm	saf_ipm – Resultat de calcul_ipm() ou calcul_ipm_national()
type	character – "indicateurs" ou "dimensions". Defaut : "indicateurs"
titre	character ou NULL – Titre du graphique. Defaut : NULL
source	character ou NULL – Note de source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
set.seed(42)
n <- 200
menages <- data.frame(
  nutrition   = rbinom(n,1,0.35), electricite = rbinom(n,1,0.60),
  eau         = rbinom(n,1,0.40), scolarisation = rbinom(n,1,0.30),
  poids       = runif(n,0.8,1.3)
)
res <- calcul_ipm(menages, var_nutrition="nutrition",
  var_electricite="electricite", var_eau="eau",
  var_scolarisation="scolarisation", poids="poids")
graphique_ipm(res)
```

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")
}
```

`handicap_prevalence` *Calculer la prevalence du handicap*

Description

Prevalence du handicap par type (visuel, auditif, moteur, cognitif) selon la methodologie Washington Group / recensement.

Usage

```
handicap_prevalence(donnees, vars_handicap, poids = NULL, sous_groupes = NULL)
```

Arguments

`donnees`	data.frame – Donnees individuelles
`vars_handicap`	named character – Vecteur nomme : type de handicap -> variable 0/1. Ex: c(Visuel="hand_vis", Moteur="hand_mot")
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec prevalence par type de handicap

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  hand_vis = rbinom(n, 1, 0.04),
  hand_aud = rbinom(n, 1, 0.03),
  hand_mot = rbinom(n, 1, 0.05),
  sexe     = sample(c("H","F"), n, TRUE),
  poids    = runif(n, 0.8, 1.3)
)
handicap_prevalence(individus,
  vars_handicap = c(Visuel="hand_vis", Auditif="hand_aud",
                    Moteur="hand_mot"),
  poids = "poids")
```

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)
```

`harmoniser_zones` *Harmoniser les noms de zones geographiques*

Description

Normalise automatiquement les noms de zones : suppression des accents, standardisation de la casse, suppression des caracteres speciaux. Permet de faire des jointures geographiques robustes meme quand les orthographes different entre sources.

Usage

```
harmoniser_zones(
  donnees,
  var_zone,
  reference = NULL,
  var_ref = "nom_local",
  seuil_similarite = 0.85
)
```

Arguments

`donnees`	data.frame – Donnees a harmoniser
`var_zone`	character – Variable contenant les noms de zones
`reference`	saf_concordance ou data.frame – Table de reference. Si NULL, harmonise seulement les noms (sans correspondance). Defaut : NULL
`var_ref`	character – Variable de la reference correspondant. Defaut : "nom_local"
`seuil_similarite`	numeric – Seuil de score Jaro-Winkler pour l'appariement automatique (0-1). Defaut : 0.85

Value

Un data.frame avec colonne supplementaire de nom harmonise

Examples

```
donnees_enquete <- data.frame(
  region    = c("Bangui","BANGUI","Ombella Mpoko","Bamingui Bangoran"),
  valeur    = c(120, 85, 200, 150),
  stringsAsFactors = FALSE
)
harmoniser_zones(donnees_enquete, "region")
```

`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)
```

indice_atkinson *Calculer l'indice d'Atkinson*

Description

L'indice d'Atkinson est une mesure d'inegalite parametrique dont le parametre epsilon reflete l aversion a l'inegalite de la societe. Epsilon = 0 : indifference ; Epsilon = 1 : forte aversion.

Usage

```
indice_atkinson(donnees, var_revenu, poids = NULL, epsilon = 1)

atkinson(donnees, var_revenu, poids = NULL, epsilon = 1)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
poids	character ou NULL – Variable de ponderation. Defaut : NULL
epsilon	numeric – Parametre d'aversion a l'inegalite (>0). Defaut : 1.0

Value

Un scalaire numerique (indice d'Atkinson)

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  poids   = runif(300, 0.8, 1.3)
)
indice_atkinson(menages, "depense", poids = "poids", epsilon = 1)
```

indice_palma *Calculer l'indice de Palma*

Description

L'indice de Palma est le rapport entre la part de revenu des 10% les plus riches et celle des 40% les plus pauvres. Il est plus sensible aux extremes que le coefficient de Gini.

Usage

```
indice_palma(donnees, var_revenu, poids = NULL)

palma(donnees, var_revenu, poids = NULL)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
poids	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un scalaire numerique (indice de Palma)

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  poids   = runif(300, 0.8, 1.3)
)
indice_palma(menages, "depense", poids = "poids")
```

---

insuffisance_ponderale

Calculer le taux d'insuffisance ponderale

Description

Prevalence de l'insuffisance ponderale : poids-pour-age < -2 ecarts-types chez les enfants de moins de 5 ans.

Usage

```
insuffisance_ponderale(
  donnees,
  var_poids_age_z,
  poids = NULL,
  sous_groupes = NULL,
  seuil = -2
)
```

Arguments

donnees	data.frame – Donnees enfants < 5 ans
var_poids_age_z	character – Score Z poids-pour-age (WAZ)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL
seuil	numeric – Seuil score Z. Defaut : -2

Value

Un objet de classe saf_anthropo

Examples

```
set.seed(42)
n <- 400
enfants <- data.frame(
  waz  = rnorm(n, -1.0, 1.2),
  poids = runif(n, 0.8, 1.3)
)
insuffisance_ponderale(enfants, var_poids_age_z = "waz", poids = "poids")
```

---

lancer_dashboard *Lancer le tableau de bord interactif statAfrikR*

Description

Lance une application Shiny pre-configuree presentant tous les indicateurs de bien-etre calcules depuis les donnees fournies. Zero code Shiny requis. Fonctionne entierement hors ligne. Compatible EHCVM, DHS, MICS, EFT.

Usage

```
lancer_dashboard(
  donnees = NULL,
  var_poids = NULL,
  var_region = NULL,
  var_milieu = NULL,
  var_annee = NULL,
  pays = "Pays",
  titre = NULL,
  sous_titre = NULL,
  port = 3838L,
  lancer = TRUE,
  export_html = NULL
)
```

Arguments

donnees	data.frame ou NULL – Donnees menages. Si NULL, utilise des donnees de demonstration. Defaut : NULL
var_poids	character ou NULL – Variable de ponderation. Defaut : NULL
var_region	character ou NULL – Variable region/prefecture. Defaut : NULL
var_milieu	character ou NULL – Variable milieu (urbain/rural). Defaut : NULL
var_annee	character ou NULL – Variable annee. Defaut : NULL
pays	character – Nom du pays pour les titres. Defaut : "Pays"
titre	character ou NULL – Titre principal du dashboard. Defaut : "Tableau de bord - NOM_DU_PAYS" (nom du pays en parametre)
sous_titre	character ou NULL – Sous-titre du dashboard. Defaut : "Indicateurs de bien-etre - NOM_DU_PAYS - statAfrikR v0.2.0"
port	integer – Port Shiny. Defaut : 3838L
lancer	logical – Lancer l'app (TRUE) ou retourner l'objet shinyApp (FALSE). Defaut : TRUE
export_html	character ou NULL – Chemin pour exporter un rapport HTML statique sans lancer Shiny. Defaut : NULL

Value

Invisible : objet shinyApp si lancer=FALSE, NULL sinon

Examples

```
## Not run:
  lancer_dashboard()
  lancer_dashboard(
    donnees    = mon_enquete,
    var_poids  = "poids_sondage",
    var_region = "prefecture",
    pays       = "Centrafrique"
  )

## 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()
```

---

mariage_precoce *Calculer le taux de mariage precoce*

Description

Proportion de femmes/hommes maries avant 18 ans (ODD 5.3.1) avec desagregation et analyse des tendances par cohorte d'age.

Usage

```
mariage_precoce(
  donnees,
  var_marie_avant_18,
  var_cohorte = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees femmes (15-49 ans)
var_marie_avant_18	character – Variable 0/1 : marie(e) avant 18 ans
var_cohorte	character ou NULL – Variable de cohorte ou groupe d'age pour analyser les tendances. Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe saf_genre

Examples

```
set.seed(42)
n <- 500
femmes <- data.frame(
  marie_18  = rbinom(n, 1, 0.42),
  cohorte   = sample(c("15-19","20-24","25-29","30-34","35-49"),
                     n, TRUE),
  milieu    = sample(c("urbain","rural"), n, TRUE),
  poids     = runif(n, 0.8, 1.3)
)
mariage_precoce(femmes, "marie_18",
                var_cohorte = "cohorte",
                poids = "poids", sous_groupes = "milieu")
```

`migrer_nomenclature` *Migrer d'une nomenclature geographique vers une autre*

Description

Convertit les codes ou noms d'une nomenclature vers une autre en utilisant une table de concordance. Gere les fusions et scissions de zones administratives dans le temps.

Usage

```
migrer_nomenclature(
  donnees,
  var_zone_source,
  concordance,
  var_source = "nom_local",
  var_cible = "code_iso",
  var_resultat = "zone_harmonisee",
  conserver_non_apparies = TRUE
)
```

Arguments

`donnees`	data.frame – Donnees a convertir
`var_zone_source`	character – Variable zone source
`concordance`	saf_concordance ou data.frame – Table de concordance
`var_source`	character – Variable source dans la concordance. Defaut : "nom_local"
`var_cible`	character – Variable cible dans la concordance. Defaut : "code_iso"
`var_resultat`	character – Nom de la nouvelle variable. Defaut : "zone_harmonisee"
`conserver_non_apparies`	logical – Conserver les zones non appariees (NA dans var_cible). Defaut : TRUE

Value

Un data.frame avec la nouvelle variable de zone

Examples

```
concordance_df <- data.frame(
  nom_local  = c("Bangui","Ombella-M'Poko","Bamingui-Bangoran"),
  code_iso   = c("CF-BGF","CF-MP","CF-BB"),
  code_ins   = c("01","02","03"),
  stringsAsFactors = FALSE
)
concordance <- table_concordance(concordance_df, "nom_local",
                                 var_code_iso="code_iso",
                                 var_code_ins="code_ins")
donnees_enquete <- data.frame(
  zone   = c("Bangui","Ombella-M'Poko","Zone inconnue"),
  valeur = c(100, 200, 150),
  stringsAsFactors = FALSE
)
migrer_nomenclature(donnees_enquete, "zone", concordance,
                    var_source="nom_local", var_cible="code_iso")
```

---

mortalite_5ans *Calculer le taux de mortalite des enfants de moins de 5 ans*

Description

Calcule le taux de mortalite infanto-juvenile (U5MR) en pour 1 000 naissances vivantes. ODD 3.2.1.

Usage

```
mortalite_5ans(
  donnees,
  var_naissances_vivantes,
  var_deces_enfants,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees femmes en age de procrer (15-49 ans) ou historique des naissances
var_naissances_vivantes	character – Nombre total de naissances vivantes par femme

var_deces_enfants	character – Nombre de deces d'enfants < 5 ans par femme
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe saf_mortalite

Examples

```
set.seed(42)
n <- 600
femmes <- data.frame(
  naissances_vivantes = rpois(n, 3.5),
  deces_enfants       = rbinom(n, 4, 0.08),
  milieu              = sample(c("urbain","rural"), n, TRUE),
  poids               = runif(n, 0.8, 1.3)
)
mortalite_5ans(femmes,
  var_naissances_vivantes = "naissances_vivantes",
  var_deces_enfants       = "deces_enfants",
  poids                   = "poids")
```

---

myers *Calculer l'indice de Myers (attraction sur tous les chiffres)*

Description

Mesure l'attraction sur les 10 chiffres terminaux (0-9). L'indice de Myers varie de 0 (aucune attraction) a 90 (attraction maximale). Un indice < 10 indique une bonne qualite des donnees d'age.

Usage

```
myers(donnees, var_age, age_min = 10L, age_max = 89L, poids = NULL)
```

Arguments

donnees	data.frame – Donnees individuelles
var_age	character – Variable age en annees revolues
age_min	integer – Age minimum. Defaut : 10L
age_max	integer – Age maximum. Defaut : 89L
poids	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un objet de classe saf_qualite_demo

Examples

```
set.seed(42)
n <- 1000
ages_base <- sample(10:89, n, TRUE)
# Attraction sur les ages ronds
ages <- ifelse(runif(n) < 0.20,
               ages_base - (ages_base %% 10),
               ages_base)
individus <- data.frame(age = pmax(10, pmin(89, ages)))
myers(individus, "age")
```

`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)
```

parite_education *Calculer l'indice de parite filles/garcons en education*

Description

Calcule l'Indice de Parite entre les Sexes (ISP) pour les taux de scolarisation bruts et nets. ODD 4.5.1. ISP = valeur filles / valeur garcons. ISP = 1 = parite parfaite.

Usage

```
parite_education(
  donnees,
  var_scolarise,
  var_sexe,
  poids = NULL,
  var_niveau = NULL
)
```

Arguments

donnees	data.frame – Donnees menages ou enfants
var_scolarise	character – Variable 0/1 : enfant scolarise
var_sexe	character – Variable sexe (H/F, M/F, 1/2, homme/femme)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
var_niveau	character ou NULL – Niveau scolaire pour ISP par niveau. Defaut : NULL

Value

Un objet de classe saf_genre

Examples

```
set.seed(42)
n <- 600
enfants <- data.frame(
  scolarise = rbinom(n, 1, 0.72),
  sexe      = sample(c("H","F"), n, TRUE),
  niveau    = sample(c("primaire","secondaire"), n, TRUE),
  poids     = runif(n, 0.8, 1.3)
)
parite_education(enfants, "scolarise", "sexe",
                 poids="poids", var_niveau="niveau")
```

---

part_quintile *Calculer les parts de revenu par quintile ou decile*

Description

Calcule la part de revenu ou de consommation detenue par chaque quintile (ou decile) de la population. Inclut les ratios Palma et Q5/Q1.

Usage

```
part_quintile(donnees, var_revenu, poids = NULL, n_groupes = 5L)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
poids	character ou NULL – Variable de ponderation. Defaut : NULL
n_groupes	integer – 5 (quintiles) ou 10 (deciles). Defaut : 5L

Value

Un tibble avec parts par groupe + ratios Palma et Q5/Q1

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  poids   = runif(300, 0.8, 1.3)
)
part_quintile(menages, "depense", poids = "poids")
```

`pauvrete_travail` *Calculer le taux de pauvrete au travail (working poor)*

Description

Proportion des actifs occupes vivant sous le seuil de pauvrete national. Indicateur ODD 8.1.1 et lien emploi-pauvrete.

Usage

```
pauvrete_travail(
  donnees,
  var_employe,
  var_revenu,
  seuil,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees individuelles (actifs occupes)
`var_employe`	character – Variable 0/1 : a un emploi
`var_revenu`	character – Variable de revenu ou consommation par tete du menage
`seuil`	numeric – Seuil de pauvrete national
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  employe = rbinom(n, 1, 0.60),
  revenu  = pmax(10000, rnorm(n, 180000, 90000)),
  milieu  = sample(c("urbain","rural"), n, TRUE),
  poids   = runif(n, 0.8, 1.3)
)
pauvrete_travail(individus, "employe", "revenu",
                  seuil = 144800, poids = "poids")
```

---

perception_economique *Calculer la perception de la situation economique*

Description

Proportion de menages pouvant joindre les deux bouts ou percevant leur situation economique comme bonne ou tres bonne.

Usage

```
perception_economique(
  donnees,
  var_perception,
  codes_positifs = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees menages
var_perception	character – Variable de perception economique. 0/1 (1=situation positive) OU categorielle avec codes_positifs.
codes_positifs	numeric ou NULL – Codes positifs si variable categorielle. Defaut : NULL (variable deja binaire)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_subjectif`

Examples

```
set.seed(42)
n <- 400
menages <- data.frame(
  econ_ok = rbinom(n, 1, 0.38),
  quintile = sample(paste0("Q", 1:5), n, TRUE),
  poids   = runif(n, 0.8, 1.3)
)
perception_economique(menages, "econ_ok", poids="poids",
                      sous_groupes="quintile")
```

`privations_ressenties` *Calculer le score de privations ressenties*

Description

Calcule la proportion de menages ayant declare souffrir de privations subjectives (manque de nourriture, eau, medicaments, argent) au cours des 12 derniers mois. Complement a l'IPM.

Usage

```
privations_ressenties(
  donnees,
  vars_privations,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees menages
`vars_privations`	named character – Vecteur nomme : type de privation -> variable 0/1. Ex : c(Nourriture="manque_nour", Eau="manque_eau", Medicaments="manque_med")
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec prevalence par type de privation

Examples

```
set.seed(42)
n <- 400
menages <- data.frame(
  manque_nour = rbinom(n, 1, 0.48),
  manque_eau  = rbinom(n, 1, 0.35),
  manque_med  = rbinom(n, 1, 0.58),
  manque_arg  = rbinom(n, 1, 0.62),
  milieu      = sample(c("urbain","rural"), n, TRUE),
  poids       = runif(n, 0.8, 1.3)
)
privations_ressenties(menages,
  vars_privations = c(Nourriture="manque_nour",
                      Eau="manque_eau",
                      Medicaments="manque_med",
                      Argent="manque_arg"),
  poids="poids", sous_groupes="milieu")
```

---

protection_sociale *Calculer la couverture de protection sociale*

Description

Calcule la proportion de la population couverte par au moins un regime de protection sociale (retraite, assurance maladie, allocations). ODD 1.3.1.

Usage

```
protection_sociale(donnees, vars_protection, poids = NULL, sous_groupes = NULL)
```

Arguments

donnees	data.frame – Donnees menages ou individus
vars_protection	named character – Vecteur nomme : type de protection -> variable 0/1. Ex: c(Retraite="retraite", Assurance_maladie="assur_maladie")
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec couverture par type de protection

Examples

```
set.seed(42)
n <- 400
individus <- data.frame(
  retraite      = rbinom(n, 1, 0.18),
  assur_maladie = rbinom(n, 1, 0.22),
  allocations   = rbinom(n, 1, 0.12),
  poids         = runif(n, 0.8, 1.3)
)
protection_sociale(individus,
  vars_protection = c(Retraite="retraite",
                      Assurance_maladie="assur_maladie",
                      Allocations="allocations"),
  poids = "poids")
```

---

pyramide_age *Construire et tracer la pyramide des ages*

Description

Produit une pyramide des ages institutionnelle au format ggplot2. Permet la superposition de deux annees pour l'analyse des tendances demographiques.

Usage

```
pyramide_age(
  donnees,
  var_age,
  var_sexe,
  code_homme = "H",
  code_femme = "F",
  poids = NULL,
  groupes_age = 5L,
  titre = NULL,
  source = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_age	character – Variable age en annees
var_sexe	character – Variable sexe
code_homme	character ou numeric – Code hommes. Defaut : "H"
code_femme	character ou numeric – Code femmes. Defaut : "F"
poids	character ou NULL – Variable de ponderation. Defaut : NULL

groupes_age	integer – Largeur des groupes. Defaut : 5L
titre	character ou NULL – Titre. Defaut : NULL
source	character ou NULL – Source. Defaut : NULL

Value

Un objet ggplot2

Examples

```
set.seed(42)
n <- 3000
individus <- data.frame(
  age   = round(rexp(n, 1/30)),
  sexe  = sample(c("H","F"), n, TRUE),
  poids = runif(n, 0.8, 1.3)
)
individus$age <- pmin(individus$age, 85)
pyramide_age(individus, "age", "sexe", poids="poids",
             titre="Pyramide des ages", source="Recensement 2024")
```

---

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")
}
```

---

rapport_harmonisation *Produire un rapport d'harmonisation geographique*

Description

Genere un rapport complet sur l'harmonisation geographique : zones appariees, non appariees, ambigues. Format institutionnel conforme aux standards IHSN/PARIS21.

Usage

```
rapport_harmonisation(
  donnees,
  var_zone,
  reference,
  var_ref = "nom_local",
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y"))
)
```

Arguments

donnees	data.frame – Donnees source
var_zone	character – Variable zone
reference	data.frame ou saf_concordance – Reference geographique
var_ref	character – Variable zone de reference. Defaut : "nom_local"
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante

Value

Un tibble de rapport d'harmonisation

Examples

```
donnees <- data.frame(
  region = c("Bangui","Ombella-Mpoko","Kemo","Zone inconnue"),
  stringsAsFactors = FALSE
)
ref <- data.frame(
  nom_local = c("Bangui","Ombella-M'Poko","Kemo","Bamingui-Bangoran"),
  stringsAsFactors = FALSE
)
rapport_harmonisation(donnees, "region", ref, pays="RCA", annee=2026L)
```

---

rapport_qualite_demo *Produire un rapport de qualite des donnees demographiques*

Description

Synthetise les indicateurs de qualite des donnees demographiques (Whipple, Myers, rapport de masculinite, coherence) en un tableau institutionnel conforme aux standards Nations Unies.

Usage

```
rapport_qualite_demo(
  donnees,
  var_age,
  var_sexe = NULL,
  poids = NULL,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  source_donnees = "Recensement"
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_age	character – Variable age
var_sexe	character ou NULL – Variable sexe. Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante
source_donnees	character – Source des donnees. Defaut : "Recensement"

Value

Un tibble de rapport qualite

Examples

```
set.seed(42)
n <- 2000
ages_base <- sample(10:79, n, TRUE)
ages <- ifelse(runif(n) < 0.15, ages_base - ages_base%%5, ages_base)
individus <- data.frame(
  age   = ages,
  sexe  = sample(c("H","F"), n, TRUE),
  poids = runif(n, 0.8, 1.3)
)
rapport_qualite_demo(individus, "age", var_sexe="sexe",
                     poids="poids", pays="RCA", annee=2024L)
```

---

rapport_qualite_sondage

Produire un rapport de qualite du plan de sondage

Description

Genere un rapport de synthese sur la qualite du plan de sondage : taux de reponse, effets de plan, CV, validite des poids. Format institutionnel IHSN/DHS.

Usage

```
rapport_qualite_sondage(
  saf_design,
  variables = NULL,
  var_reponse = NULL,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y"))
)
```

Arguments

saf_design	saf_design – Objet design
variables	character ou NULL – Variables pour le calcul du DEFF. Defaut : NULL
var_reponse	character ou NULL – Variable de non-reponse. Defaut : NULL
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante

Value

Un tibble de rapport qualite

Examples

```
set.seed(42)
n <- 300
donnees <- data.frame(
  poids_sond = runif(n, 1800, 3500),
  strate     = sample(c("Urbain","Rural"), n, TRUE),
  grappe     = sample(1:20, n, TRUE),
  repondu    = rbinom(n, 1, 0.88),
  pauvre     = rbinom(n, 1, 0.45)
)
design <- suppressMessages(
  creer_design(donnees,"poids_sond",var_strate="strate",var_grappe="grappe"))
rapport_qualite_sondage(design, variables="pauvre",
                        var_reponse="repondu",
                        pays="RCA", annee=2026L)
```

---

ratio_masculinite *Calculer le rapport de masculinite par groupe d'age*

Description

Calcule le rapport de masculinite (nombre d'hommes pour 100 femmes) par groupe d'age quinquennal. Permet de detecter des anomalies liees aux guerres, migrations ou erreurs de collecte.

Usage

```
ratio_masculinite(
  donnees,
  var_age,
  var_sexe,
  code_homme = "H",
  code_femme = "F",
  poids = NULL,
  groupes_age = 5L
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_age	character – Variable age en annees
var_sexe	character – Variable sexe
code_homme	character ou numeric – Code pour les hommes. Defaut : "H"
code_femme	character ou numeric – Code pour les femmes. Defaut : "F"
poids	character ou NULL – Variable de ponderation. Defaut : NULL
groupes_age	integer – Largeur des groupes d'age. Defaut : 5L

Value

Un tibble avec rapport de masculinite par groupe d'age

Examples

```
set.seed(42)
n <- 2000
individus <- data.frame(
  age   = sample(0:79, n, TRUE),
  sexe  = sample(c("H","F"), n, TRUE, prob=c(0.49,0.51)),
  poids = runif(n, 0.8, 1.3)
)
ratio_masculinite(individus, "age", "sexe", poids="poids")
```

---

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)
```

---

retard_croissance *Calculer le taux de retard de croissance (stunting)*

Description

Calcule la prevalence du retard de croissance chez les enfants de moins de 5 ans (taille-pour-age < -2 ecarts-types selon les standards OMS 2006). ODD 2.2.1.

Usage

```
retard_croissance(
  donnees,
  var_taille_age_z = NULL,
  var_taille = NULL,
  var_age_mois = NULL,
  var_sexe = NULL,
  poids = NULL,
```

```
  sous_groupes = NULL,
  seuil = -2
)
```

Arguments

donnees	data.frame – Donnees enfants < 5 ans
var_taille_age_z	character – Score Z taille-pour-age (HAZ). Si NULL, calcule depuis var_taille, var_age et var_sexe. Defaut : NULL
var_taille	character ou NULL – Taille en cm. Defaut : NULL
var_age_mois	character ou NULL – Age en mois. Defaut : NULL
var_sexe	character ou NULL – Sexe (1=garcon, 2=fille ou "M"/"F"). Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL
seuil	numeric – Seuil (score Z). Defaut : -2

Value

Un objet de classe `saf_anthropo`

Examples

```
set.seed(42)
n <- 500
enfants <- data.frame(
  haz   = rnorm(n, -1.2, 1.3),
  milieu = sample(c("urbain","rural"), n, TRUE),
  poids  = runif(n, 0.7, 1.4)
)
retard_croissance(enfants, var_taille_age_z = "haz", poids = "poids")
```

---

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')
```

sante_mentale

Calculer les indicateurs de sante mentale

Description

Prevalence des symptomes de stress, anxiete et isolement percu. Compatible avec les modules Kessler K6, PHQ-4 et GAD-2 adaptes aux enquetes menages africaines.

Usage

```
sante_mentale(
  donnees,
  var_stress = NULL,
  var_anxiete = NULL,
  var_isolement = NULL,
  type = c("binaire", "score"),
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_stress	character ou NULL – Variable stress (0/1 ou score). Defaut : NULL
var_anxiete	character ou NULL – Variable anxiete (0/1). Defaut : NULL
var_isolement	character ou NULL – Variable isolement social (0/1). Defaut : NULL
type	character – "binaire" (variables 0/1) ou "score" (variables continues). Defaut : "binaire"
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec prevalence par indicateur

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  stress    = rbinom(n, 1, 0.38),
  anxiete   = rbinom(n, 1, 0.28),
  isolement = rbinom(n, 1, 0.22),
  sexe      = sample(c("H","F"), n, TRUE),
  poids     = runif(n, 0.8, 1.3)
)
sante_mentale(individus, var_stress="stress",
              var_anxiete="anxiete", var_isolement="isolement",
              poids="poids", sous_groupes="sexe")
```

---

satisfaction_vie *Calculer le score de satisfaction dans la vie*

Description

Calcule le score moyen de satisfaction dans la vie sur une echelle de 0 a 10 (echelle de Cantril). Indicateur de bien-etre subjectif global conforme a la methodologie OCDE / Gallup World Poll.

Usage

```
satisfaction_vie(donnees, var_satisfaction, poids = NULL, sous_groupes = NULL)
```

Arguments

donnees	data.frame – Donnees individuelles
var_satisfaction	character – Variable de satisfaction (0-10)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_subjectif`

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  satisfaction = pmin(10, pmax(0, round(rnorm(n, 5.8, 2.1)))),
  milieu       = sample(c("urbain","rural"), n, TRUE),
  quintile     = sample(paste0("Q", 1:5), n, TRUE),
  poids        = runif(n, 0.8, 1.3)
)
satisfaction_vie(individus, "satisfaction", poids="poids",
                 sous_groupes=c("milieu","quintile"))
```

---

sentiment_securite *Calculer le sentiment de securite*

Description

Proportion de la population se sentant en securite (le soir dans le quartier, au travail, dans les transports). Methodologie Afrobarometer / Gallup.

Usage

```
sentiment_securite(
  donnees,
  var_securite,
  codes_securise = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_securite	character – Variable de securite percue (0/1 ou codes_securise)
codes_securise	numeric ou NULL – Codes indiquant "securise". Defaut : NULL (variable deja binaire)
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_subjectif`

Examples

```
set.seed(42)
n <- 400
individus <- data.frame(
  securise = rbinom(n, 1, 0.52),
  sexe     = sample(c("H","F"), n, TRUE),
  milieu   = sample(c("urbain","rural"), n, TRUE),
  poids    = runif(n, 0.8, 1.3)
)
sentiment_securite(individus, "securise", poids="poids",
                   sous_groupes=c("sexe","milieu"))
```

---

sous_emploi_temps *Calculer le taux de sous-emploi en temps de travail*

Description

Calcule la proportion de personnes en emploi travaillant moins d'heures qu'elles ne le souhaitent (sous-emploi visible). Composante de LU2 dans la mesure de la sous-utilisation de la main-d'oeuvre OIT.

Usage

```
sous_emploi_temps(
  donnees,
  var_heures_travaillees,
  var_heures_souhaitees = NULL,
  seuil_heures = 35L,
  var_disponible = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individus en emploi
var_heures_travaillees	character – Heures effectivement travaillees par semaine
var_heures_souhaitees	character ou NULL – Heures souhaitees. Si NULL, utilise le seuil. Defaut : NULL
seuil_heures	integer – Seuil d'heures hebdomadaires en dessous duquel il y a sous-emploi (defaut : 35L)
var_disponible	character ou NULL – Variable 0/1 : disponible pour travailler plus. Defaut : NULL
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
n <- 400
employes <- data.frame(
  heures_trav = pmax(0, rnorm(n, 38, 15)),
  disponible  = rbinom(n, 1, 0.3),
  poids       = runif(n, 0.8, 1.3)
)
sous_emploi_temps(employes, "heures_trav",
                  seuil_heures = 35L, poids = "poids")
```

`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)
```

---

suivre_revisions_pib *Suivre les revisions du PIB*

Description

Analyse les revisions entre les estimations preliminaires, provisoires et definitives du PIB. Produit un tableau et un graphique de suivi conforme aux bonnes pratiques de communication des INS.

Usage

```
suivre_revisions_pib(
  donnees,
  var_annee = "annee",
  var_version = "version",
  var_valeur = "valeur",
  pays = "Pays",
  unite = "Mds FCFA"
)
```

Arguments

donnees	data.frame – Donnees avec colonnes annee, version, valeur. Versions attendues : "preliminaire", "provisoire", "definitif"
var_annee	character – Variable annee. Defaut : "annee"
var_version	character – Variable version du PIB. Defaut : "version"

var_valeur	character – Variable valeur du PIB. Defaut : "valeur"
pays	character – Nom du pays. Defaut : "Pays"
unite	character – Unite. Defaut : "Mds FCFA"

Value

Un objet de classe saf_revision_pib

Examples

```
revisions <- data.frame(
  annee   = rep(2019:2022, each=3),
  version = rep(c("preliminaire","provisoire","definitif"), 4),
  valeur  = c(2210,2225,2240, 2320,2335,2348,
              2415,2430,NA,   2510,NA,NA),
  stringsAsFactors = FALSE
)
suivre_revisions_pib(revisions, pays="Cameroun", unite="Mds FCFA")
```

---

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)
```

`table_concordance` *Creer une table de concordance entre nomenclatures geographiques*

Description

Cree une table de correspondance entre les differentes nomenclatures geographiques utilisees en Afrique : nomenclature locale des INS, codes ISO 3166-2, codes GADM, codes EHCVM, et noms officiels. Essentielle pour harmoniser les donnees entre plusieurs sources.

Usage

```
table_concordance(
  donnees,
  var_nom_local = "nom_zone",
  var_code_ins = NULL,
  var_code_iso = NULL,
  var_code_gadm = NULL,
  var_niveau = NULL,
  var_pays = NULL
)
```

Arguments

`donnees`	data.frame – Donnees de reference avec les noms/codes des zones administratives
`var_nom_local`	character – Variable nom local (utilise par l'INS). Defaut : "nom_zone"
`var_code_ins`	character ou NULL – Variable code INS. Defaut : NULL
`var_code_iso`	character ou NULL – Variable code ISO 3166-2. Defaut : NULL
`var_code_gadm`	character ou NULL – Variable code GADM. Defaut : NULL
`var_niveau`	character ou NULL – Variable niveau administratif (1 = region, 2 = departement, etc.). Defaut : NULL
`var_pays`	character ou NULL – Variable pays (code ISO 3). Defaut : NULL

Value

Un objet de classe `saf_concordance`

Examples

```
zones_rca <- data.frame(
  nom_zone   = c("Bangui","Ombella-M'Poko","Bamingui-Bangoran",
                 "Mbomou","Haute-Kotto"),
  code_ins   = c("01","02","03","04","05"),
  code_iso   = c("CF-BGF","CF-MP","CF-BB","CF-MB","CF-HK"),
  code_gadm  = c("GADMa001","GADMa002","GADMa003","GADMa004","GADMa005"),
  niveau     = rep(1L, 5),
  pays       = rep("CAF", 5),
  stringsAsFactors = FALSE
)
table_concordance(zones_rca, var_nom_local="nom_zone",
                  var_code_ins="code_ins", var_code_iso="code_iso",
                  var_code_gadm="code_gadm", var_niveau="niveau",
                  var_pays="pays")
```

---

tableau_bien_etre_subjectif

Tableau de bord du bien-etre subjectif

Description

Synthetise tous les indicateurs de bien-etre subjectif en un tableau institutionnel unique. Format adapte aux rapports nationaux sur le bien-etre et la qualite de vie.

Usage

```
tableau_bien_etre_subjectif(
  indicateurs,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  source = NULL
)
```

Arguments

indicateurs	list – Liste nommee d'objets saf_subjectif ou tibbles
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante
source	character ou NULL – Source. Defaut : NULL

Value

Un tibble institutionnel

Examples

```
set.seed(42)
n <- 400
individus <- data.frame(
  satisf   = pmin(10,pmax(0,round(rnorm(n,5.8,2.1)))),
  bonheur  = rbinom(n,1,0.62),
  securise = rbinom(n,1,0.52),
  poids    = runif(n,0.8,1.3)
)
r1 <- suppressMessages(
  satisfaction_vie(individus,"satisf",poids="poids"))
r2 <- suppressMessages(
  sentiment_securite(individus,"securise",poids="poids"))
tableau_bien_etre_subjectif(list(satisfaction=r1, securite=r2),
                            pays="RCA", annee=2026L)
```

---

tableau_bord_pib *Tableau de bord des comptes nationaux*

Description

Produit un tableau de bord institutionnel du PIB combinant croissance, deflateur, comparaison de sources et revisions. Format comparable aux publications des INS africains.

Usage

```
tableau_bord_pib(
  donnees,
  var_pib_courant,
  var_pib_constant = NULL,
  var_annee = "annee",
  vars_secteurs = NULL,
  pays = "Pays",
  unite = "Mds FCFA"
)
```

Arguments

donnees	data.frame – Donnees de comptes nationaux
var_pib_courant	character – PIB en prix courants
var_pib_constant	character ou NULL – PIB en prix constants. Defaut : NULL
var_annee	character – Variable annee. Defaut : "annee"
vars_secteurs	named character ou NULL – Secteurs economiques. Defaut : NULL
pays	character – Nom du pays. Defaut : "Pays"
unite	character – Unite. Defaut : "Mds FCFA"

Value

Un tibble du tableau de bord PIB

Examples

```
comptes <- data.frame(
  annee       = 2018:2022,
  pib_courant = c(2100,2180,2250,2195,2380),
  pib_cst     = c(2000,2060,2110,2060,2210)
)
tableau_bord_pib(comptes, "pib_courant",
                 var_pib_constant="pib_cst",
                 pays="Cameroun", unite="Mds FCFA")
```

---

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_genre *Tableau de bord des indicateurs de genre et d'inclusion*

Description

Synthetise tous les indicateurs genre en un tableau institutionnel comparable aux rapports ODD 5 et DHS.

Usage

```
tableau_genre(
  indicateurs,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  source = NULL
)
```

Arguments

indicateurs	list – Liste nommee d'objets saf_genre ou tibbles
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante
source	character ou NULL – Source. Defaut : NULL

Value

Un tibble institutionnel

Examples

```
set.seed(42)
n <- 400
femmes <- data.frame(
  marie_18 = rbinom(n, 1, 0.42),
  vbg_phys = rbinom(n, 1, 0.28),
  poids    = runif(n, 0.8, 1.3)
)
rm <- suppressMessages(
  mariage_precoce(femmes, "marie_18", poids="poids"))
rvbg <- violence_basee_genre(femmes,
  var_violence_physique="vbg_phys", poids="poids")
tableau_genre(list(mariage=rm, vbg=rvbg), pays="RCA", annee=2026L)
```

---

tableau_inegalites *Tableau de bord des inegalites*

Description

Calcule toutes les mesures d'inegalite en une seule fonction et les presente dans un tableau institutionnel. Inclut Gini, Palma, Theil, Atkinson et les parts par quintile.

Usage

```
tableau_inegalites(
  donnees,
  var_revenu,
  poids = NULL,
  var_groupe = NULL,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y"))
)
```

Arguments

donnees	data.frame – Donnees menages
var_revenu	character – Variable de revenu
poids	character ou NULL – Variable de ponderation. Defaut : NULL
var_groupe	character ou NULL – Variable de groupe pour decomposition. Defaut : NULL
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante

Value

Un tibble avec toutes les mesures d'inegalite

Examples

```
set.seed(42)
menages <- data.frame(
  depense = c(rexp(200, 1/150000), rexp(100, 1/500000)),
  milieu  = sample(c("urbain","rural"), 300, TRUE),
  poids   = runif(300, 0.8, 1.3)
)
tableau_inegalites(menages, "depense", poids="poids",
                   var_groupe="milieu", pays="Centrafrique",
                   annee=2026L)
```

`tableau_ipm` *Tableau institutionnel IPM*

Description

Genere un tableau flextable institutionnel (style INS/PARIS21) des resultats IPM, exportable en Word ou Excel.

Usage

```
tableau_ipm(
  res_ipm,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  chemin_word = NULL
)
```

Arguments

`res_ipm`	saf_ipm – Resultat de calcul_ipm() ou calcul_ipm_national()
`pays`	character – Nom du pays. Defaut : "Pays"
`annee`	integer – Annee de reference
`chemin_word`	character ou NULL – Chemin Word (.docx). Si NULL, retourne l'objet flextable. Defaut : NULL

Value

Un objet `flextable` (invisible si chemin_word fourni)

Examples

```
set.seed(42)
n <- 200
menages <- data.frame(
  nutrition   = rbinom(n,1,0.35), electricite = rbinom(n,1,0.60),
  eau         = rbinom(n,1,0.40), poids = runif(n,0.8,1.3)
)
res <- calcul_ipm(menages, var_nutrition="nutrition",
  var_electricite="electricite", var_eau="eau", poids="poids")
tableau_ipm(res, pays="Centrafrique", annee=2026L)
```

---

tableau_marche_travail

Tableau de bord du marche du travail

Description

Synthetise tous les indicateurs du marche du travail en un tableau institutionnel unique, comparable aux publications OIT/ILOSTAT et aux rapports nationaux sur l'emploi.

Usage

```
tableau_marche_travail(
  indicateurs,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  source = NULL
)
```

Arguments

indicateurs	list – Liste nommee d'objets saf_emploi ou tibbles (protection_sociale, taux_sous_utilisation)
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante
source	character ou NULL – Source. Defaut : NULL

Value

Un tibble institutionnel

Examples

```
set.seed(42)
n <- 500
ind <- data.frame(
  actif      = rbinom(n, 1, 0.65),
  employe    = rbinom(n, 1, 0.55),
  informel   = rbinom(n, 1, 0.72),
  poids      = runif(n, 0.8, 1.3),
  stringsAsFactors = FALSE
)
r1 <- suppressMessages(taux_activite(ind, "actif", poids="poids"))
r2 <- suppressMessages(taux_emploi(ind, "employe", poids="poids"))
r3 <- suppressMessages(emploi_informel(ind, "informel", poids="poids"))
tableau_marche_travail(list(activite=r1, emploi=r2, informel=r3),
                       pays="Centrafrique", annee=2026L)
```

---

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)
```

---

tableau_sante *Tableau de bord des indicateurs de sante*

Description

Synthetise tous les indicateurs de sante disponibles en un seul tableau institutionnel. Produit un rapport comparable aux tableaux de bord DHS/MICS.

Usage

```
tableau_sante(
  indicateurs,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y")),
  source = NULL
)
```

Arguments

indicateurs	list – Liste nommee d'objets saf_anthropo ou saf_mortalite, ou tibbles de vaccination/accouchements
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee. Defaut : annee courante
source	character ou NULL – Source des donnees. Defaut : NULL

Value

Un tibble institutionnel

Examples

```
set.seed(42)
n <- 400
enfants <- data.frame(
  haz  = rnorm(n,-1.2,1.3), waz = rnorm(n,-1.0,1.2),
  poids = runif(n,0.8,1.3)
)
r1 <- suppressMessages(
  retard_croissance(enfants, var_taille_age_z="haz", poids="poids"))
r2 <- suppressMessages(
  insuffisance_ponderale(enfants, var_poids_age_z="waz", poids="poids"))
tableau_sante(list(stunting=r1, underweight=r2),
              pays="Centrafrique", annee=2026L)
```

---

taux_activite *Calculer le taux d'activite*

Description

Calcule le taux d'activite selon la definition du Bureau International du Travail (BIT) : proportion de la population en age de travailler qui est active (employes + chomeurs). ODD 8.

Usage

```
taux_activite(
  donnees,
  var_actif,
  var_age = NULL,
  age_min = 15L,
  age_max = 64L,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles ou menages
var_actif	character – Variable 0/1 : actif (employe ou chomeur)
var_age	character ou NULL – Variable age pour filtrer la population en age de travailler. Defaut : NULL
age_min	integer – Age minimum. Defaut : 15L
age_max	integer – Age maximum. Defaut : 64L
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation (sexe, milieu, region). Defaut : NULL

Value

Un objet de classe saf_emploi

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  actif  = rbinom(n, 1, 0.65),
  age    = sample(15:64, n, TRUE),
  sexe   = sample(c("H","F"), n, TRUE),
  milieu = sample(c("urbain","rural"), n, TRUE),
  poids  = runif(n, 0.8, 1.3)
)
taux_activite(individus, "actif", var_age="age", poids="poids",
              sous_groupes=c("sexe","milieu"))
```

---

taux_croissance *Calculer les taux de croissance du PIB et contributions*

Description

Calcule les taux de croissance reels et nominaux du PIB et la contribution de chaque secteur a la croissance. Conforme aux methodes SCN 2008.

Usage

```
taux_croissance(
  donnees,
  var_pib,
  vars_secteurs = NULL,
  var_annee = "annee",
  var_trimestre = NULL
)
```

Arguments

donnees	data.frame – Donnees de comptes nationaux avec secteurs
var_pib	character – Variable PIB total (prix constants)
vars_secteurs	named character ou NULL – Vecteur nomme des variables sectorielles. Ex : c(Agriculture="agri", Industrie="indus")
var_annee	character – Variable annee. Defaut : "annee"
var_trimestre	character ou NULL – Variable trimestre (si donnees trimestrielles). Defaut : NULL

Value

Un tibble avec taux de croissance et contributions sectorielles

Examples

```
comptes <- data.frame(
  annee = 2018:2022,
  pib   = c(2100, 2180, 2250, 2195, 2380),
  agri  = c(420,  440,  460,  445,  490),
  indus = c(630,  660,  680,  660,  715),
  serv  = c(1050, 1080, 1110, 1090, 1175)
)
taux_croissance(comptes, "pib",
  vars_secteurs = c(Agriculture="agri",
                    Industrie="indus",
                    Services="serv"),
  var_annee = "annee")
```

---

taux_emploi *Calculer le taux d'emploi*

Description

Proportion de la population en age de travailler ayant un emploi (salarie ou independant). Indicateur cle ODD 8.5.

Usage

```
taux_emploi(
  donnees,
  var_employe,
  var_age = NULL,
  age_min = 15L,
  age_max = 64L,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

donnees	data.frame – Donnees individuelles
var_employe	character – Variable 0/1 : a un emploi
var_age	character ou NULL – Variable age. Defaut : NULL
age_min	integer – Age minimum. Defaut : 15L
age_max	integer – Age maximum. Defaut : 64L
poids	character ou NULL – Variable de ponderation. Defaut : NULL
sous_groupes	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
individus <- data.frame(
  employe = rbinom(500, 1, 0.55),
  age     = sample(15:64, 500, TRUE),
  sexe    = sample(c("H","F"), 500, TRUE),
  poids   = runif(500, 0.8, 1.3)
)
taux_emploi(individus, "employe", var_age="age",
            poids="poids", sous_groupes="sexe")
```

---

taux_sous_utilisation *Calculer le taux de sous-utilisation composite de la main-d'oeuvre*

Description

Calcule le taux composite OIT LU3 ou LU4 qui agrege le chomage, le sous-emploi en temps de travail et les actifs potentiels non recherchant activement. Depasse le simple taux de chomage pour mesurer la sous-utilisation reelle.

Usage

```
taux_sous_utilisation(
  donnees,
  var_chomeur,
  var_sous_emploi_t = NULL,
  var_actif_potentiel = NULL,
  poids = NULL
)
```

Arguments

`donnees`	data.frame – Donnees individuelles
`var_chomeur`	character – Variable 0/1 : chomeur (BIT)
`var_sous_emploi_t`	character ou NULL – Variable 0/1 : sous-emploi temps. Defaut : NULL
`var_actif_potentiel`	character ou NULL – Variable 0/1 : actif potentiel (souhaite travailler mais ne cherche pas). Defaut : NULL
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un tibble avec les composantes et le taux composite

Examples

```
set.seed(42)
n <- 500
individus <- data.frame(
  chomeur         = rbinom(n, 1, 0.14),
  sous_emploi_t   = rbinom(n, 1, 0.08),
  actif_potentiel = rbinom(n, 1, 0.05),
  poids           = runif(n, 0.8, 1.3)
)
taux_sous_utilisation(individus, "chomeur",
                      var_sous_emploi_t = "sous_emploi_t",
                      var_actif_potentiel = "actif_potentiel",
                      poids = "poids")
```

---

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)
```

`travail_enfants` *Calculer la prevalence du travail des enfants*

Description

Prevalence du travail des enfants (5-17 ans) selon la definition OIT/MICS. ODD 8.7.1.

Usage

```
travail_enfants(
  donnees,
  var_travail_enfant,
  var_age = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees enfants 5-17 ans
`var_travail_enfant`	character – Variable 0/1 : enfant au travail
`var_age`	character ou NULL – Variable age pour filtrer 5-17 ans. Defaut : NULL
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un objet de classe `saf_emploi`

Examples

```
set.seed(42)
n <- 600
enfants <- data.frame(
  travail = rbinom(n, 1, 0.22),
  age     = sample(5:17, n, TRUE),
  sexe    = sample(c("H","F"), n, TRUE),
  milieu  = sample(c("urbain","rural"), n, TRUE),
  poids   = runif(n, 0.8, 1.3)
)
travail_enfants(enfants, "travail", var_age="age",
                poids="poids", sous_groupes=c("sexe","milieu"))
```

`vaccination` *Calculer la couverture vaccinale*

Description

Calcule la couverture vaccinale par antigene selon la methodologie DHS/MICS. Indicateurs ODD 3.b.1.

Usage

```
vaccination(
  donnees,
  vars_vaccins,
  poids = NULL,
  var_carte_sante = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees enfants
`vars_vaccins`	named character – Vecteur nomme : nom du vaccin -> nom de la variable (0/1). Ex: c(DTC3="dtc3", Rougeole="rougeole")
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`var_carte_sante`	character ou NULL – Variable indiquant si l'enfant a une carte de sante (pour ajustement). Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec couverture par antigene

Examples

```
set.seed(42)
n <- 500
enfants <- data.frame(
  dtc3      = rbinom(n, 1, 0.72),
  rougeole  = rbinom(n, 1, 0.68),
  polio3    = rbinom(n, 1, 0.75),
  bcg       = rbinom(n, 1, 0.90),
  poids     = runif(n, 0.8, 1.3),
  milieu    = sample(c("urbain","rural"), n, TRUE)
)
vaccination(enfants,
  vars_vaccins = c(DTC3="dtc3", Rougeole="rougeole",
```

```
                  Polio3="polio3", BCG="bcg"),
  poids = "poids")
```

`valider_coherence_geo` *Valider la coherence de la couverture geographique*

Description

Verifie que les donnees couvrent l'ensemble des zones du referentiel officiel. Identifie les zones manquantes, les doublons et les zones surnumeraires (non dans le referentiel).

Usage

```
valider_coherence_geo(donnees, var_zone, referentiel, var_ref = "nom_local")
```

Arguments

`donnees`	data.frame – Donnees a verifier
`var_zone`	character – Variable zone dans les donnees
`referentiel`	data.frame ou saf_concordance – Referentiel officiel
`var_ref`	character – Variable zone dans le referentiel. Defaut : "nom_local"

Value

Un tibble de rapport de coherence

Examples

```
donnees <- data.frame(
  region = c("Bangui","Ombella-M'Poko","Bangui"),
  valeur = c(100, 200, 50),
  stringsAsFactors = FALSE
)
ref <- data.frame(
  nom_local = c("Bangui","Ombella-M'Poko","Kemo","Mbomou"),
  stringsAsFactors = FALSE
)
valider_coherence_geo(donnees, "region", ref, "nom_local")
```

`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_poids *Valider les poids de sondage*

Description

Verifie la coherence des poids de sondage : somme par strate, detection des poids aberrants (outliers), DEFF eleves, comparaison avec la population cible. Etape indispensable avant toute analyse.

Usage

```
valider_poids(saf_design, population_cible = NULL, seuil_outlier = 3)
```

Arguments

saf_design	saf_design – Objet cree par creer_design()
population_cible	numeric ou NULL – Population totale cible (pour verifier que sum(poids) est coherent). Defaut : NULL
seuil_outlier	numeric – Seuil de detection des outliers en ecarts-types (score Z). Defaut : 3.0

Value

Un tibble avec les statistiques de validation par strate

Examples

```
set.seed(42)
n <- 300
donnees <- data.frame(
  poids_sond = runif(n, 1800, 3500),
  strate     = sample(c("Urbain","Rural"), n, TRUE),
  grappe     = sample(1:20, n, TRUE)
)
design <- suppressMessages(
  creer_design(donnees,"poids_sond",var_strate="strate",var_grappe="grappe"))
valider_poids(design)
```

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)
```

---

valider_statistique_ins

Valider la qualite statistique d'une enquete (INS Validation Suite)

Description

Produit un rapport de validation complet couvrant 10 criteres de qualite statistique institutionnelle : donnees, poids, plan de sondage, valeurs manquantes, geographie, precision, confidentialite, metadonnees, reproductibilite et indicateurs cles. Conforme aux standards IHSN / PARIS21 / ONU.

Usage

```
valider_statistique_ins(
  donnees,
  var_poids = NULL,
  var_region = NULL,
  var_milieu = NULL,
  var_depense = NULL,
  seuil_pauvrete = NULL,
  dictionnaire = NULL,
  seuil_na_alerte = 0.05,
  seuil_cv_alerte = 33,
```

```
  seuil_cellule = 5L,
  pays = "Pays",
  annee = as.integer(format(Sys.Date(), "%Y"))
)
```

Arguments

donnees	data.frame – Donnees de l'enquete
var_poids	character ou NULL – Variable de ponderation. Defaut : NULL
var_region	character ou NULL – Variable region/prefecture. Defaut : NULL
var_milieu	character ou NULL – Variable milieu. Defaut : NULL
var_depense	character ou NULL – Variable de consommation/depense pour le calcul FGT. Defaut : NULL
seuil_pauvrete	numeric ou NULL – Seuil de pauvrete. Defaut : NULL
dictionnaire	data.frame ou NULL – Dictionnaire des variables avec colonnes : variable, label, type_attendu. Defaut : NULL
seuil_na_alerte	numeric – Seuil de % valeurs manquantes declenchant une alerte. Defaut : 0.05 (5%)
seuil_cv_alerte	numeric – Seuil de CV declenchant une alerte. Defaut : 33.0
seuil_cellule	integer – Seuil de confidentialite. Defaut : 5L
pays	character – Nom du pays. Defaut : "Pays"
annee	integer – Annee de l'enquete. Defaut : annee courante

Value

Un objet de classe `saf_validation_ins` avec le rapport complet et le statut global

Examples

```
set.seed(42)
n <- 500
donnees <- data.frame(
  region  = sample(c("Nord","Sud","Est","Ouest"), n, TRUE),
  milieu  = sample(c("urbain","rural"), n, TRUE),
  poids   = runif(n, 800, 3500),
  depense = pmax(10000, rnorm(n, 165000, 90000)),
  sexe    = sample(c("H","F"), n, TRUE),
  age     = sample(15:80, n, TRUE),
  stringsAsFactors = FALSE
)
valider_statistique_ins(donnees,
  var_poids  = "poids",
  var_region = "region",
  var_milieu = "milieu",
  pays       = "Centrafrique",
  annee      = 2024L)
```

`violence_basee_genre` *Calculer la prevalence des violences basees sur le genre*

Description

Prevalence des violences physiques, sexuelles et/ou psychologiques exercees par un partenaire intime ou un non-partenaire. Methodologie DHS/OMS. ODD 5.2.1.

Usage

```
violence_basee_genre(
  donnees,
  var_violence_physique = NULL,
  var_violence_sexuelle = NULL,
  var_violence_psycho = NULL,
  poids = NULL,
  sous_groupes = NULL
)
```

Arguments

`donnees`	data.frame – Donnees femmes (15-49 ans)
`var_violence_physique`	character ou NULL – VBG physique (0/1)
`var_violence_sexuelle`	character ou NULL – VBG sexuelle (0/1)
`var_violence_psycho`	character ou NULL – VBG psychologique (0/1)
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL
`sous_groupes`	character ou NULL – Variables de desagregation. Defaut : NULL

Value

Un tibble avec prevalence par type de violence

Examples

```
set.seed(42)
n <- 500
femmes <- data.frame(
  vbg_phys   = rbinom(n, 1, 0.28),
  vbg_sex    = rbinom(n, 1, 0.18),
  vbg_psycho = rbinom(n, 1, 0.35),
  milieu     = sample(c("urbain","rural"), n, TRUE),
  poids      = runif(n, 0.8, 1.3)
)
violence_basee_genre(femmes,
```

```
  var_violence_physique  = "vbg_phys",
  var_violence_sexuelle  = "vbg_sex",
  var_violence_psycho    = "vbg_psycho",
  poids = "poids", sous_groupes = "milieu")
```

`whipple` *Calculer l'indice de Whipple*

Description

Mesure l'attraction des ages se terminant par 0 et 5 dans les declarations d'age. Un indice de Whipple < 105 indique des donnees de tres bonne qualite ; > 175 indique une qualite tres mauvaise. Methodologie Nations Unies / IUSSP.

Usage

```
whipple(donnees, var_age, age_min = 23L, age_max = 62L, poids = NULL)
```

Arguments

`donnees`	data.frame – Donnees individuelles
`var_age`	character – Variable age en annees revolues
`age_min`	integer – Age minimum pour le calcul. Defaut : 23L
`age_max`	integer – Age maximum pour le calcul. Defaut : 62L
`poids`	character ou NULL – Variable de ponderation. Defaut : NULL

Value

Un objet de classe `saf_qualite_demo`

Examples

```
set.seed(42)
n <- 1000
# Simulation d'attraction sur les multiples de 5
ages_base <- sample(23:62, n, TRUE)
attire    <- ages_base - (ages_base %% 5) # attire vers 0 et 5
ages      <- ifelse(runif(n) < 0.25, attire, ages_base)
individus <- data.frame(age = ages, poids = runif(n, 0.8, 1.3))
whipple(individus, "age")
```

Index

∗ **datasets**
  saf_cameroun_departements, 104
  saf_cedeao, 105
  saf_cemac, 106
  saf_eau, 106
  saf_pays_afrique, 107
  saf_rca_prefectures, 107
  saf_sadc, 108
  saf_senegal_regions, 108
  saf_subdivisions_afrique, 109

accouchements_assistes, 5
analyse_regression, 6
analyse_spatiale, 7
analyser_non_reponse, 8
anemie, 9
anonymiser_donnees, 10
appliquer_ponderations, 11
atkinson *(*indice_atkinson*)*, 78
autonomisation_femmes, 12

bonheur_declare, 14

calcul_deff, 15
calcul_fgt, 15
calcul_gini, 17
calcul_idh, 18
calcul_ipm, 19
calcul_ipm_national, 21
calculer_deflateur, 22
calibrer_poids, 23
carte_choroplethe, 24
carte_exporter, 26
carte_import, 27
carte_ipm, 27
carte_joindre, 28
carte_pauvrete, 29
carte_thematique, 30
carte_zones, 32
catalogue_indicateurs, 33
check_na, 34
check_types, 34
coherence_demo, 35
comparer_ipm, 36
comparer_pib, 37
compresser_package_diffusion, 38
confiance_institutions, 39
courbe_lorenz, 41
creer_design, 42

decomposer_fgt, 43
decomposer_inegalite, 44
decomposer_ipm, 45
decomposer_theil, 46
detecter_ecarts_geo, 46

emaciation, 47
emploi_informel, 48
emploi_vulnerable, 49
exporter_code_dashboard, 50
exporter_excel_ins, 51
exporter_graphique, 52
exporter_sdmx, 53

fusion_datasets, 55

generer_bulletin, 56
generer_metadonnees_ddi, 57
generer_rapport, 58
generer_rapport_enquete, 59
generer_rapport_odd, 60
graphique_barres, 62
graphique_fgt, 63
graphique_ipm, 64
graphique_tendance, 65

handicap_prevalence, 66
harmoniser_regions, 67
harmoniser_zones, 68

import_cspro, 69

import_csv, 70, *71*
import_excel, 71
import_kobo, 72
import_odk, 73
import_sas, 74
import_spss, 75
import_stata, 76
imputer_valeurs, 77
indice_atkinson, 78
indice_palma, 78
insuffisance_ponderale, 79

lancer_dashboard, 80
lister_templates, 81

mariage_precoce, 82
migrer_nomenclature, 83
mortalite_5ans, 84
myers, 85

nettoyer_libelles, 86

odd_catalogue, 87
odd_indicateur, 88

palette_ins, 90
palma *(*indice_palma*)*, 78
parite_education, 90
part_quintile, 91
pauvrete_travail, 92
perception_economique, 93
privations_ressenties, 94
protection_sociale, 95
pyramide_age, 96
pyramide_ages, 97

rapport_harmonisation, 98
rapport_qualite_demo, 99
rapport_qualite_sondage, 100
ratio_masculinite, 101
recoder_variable, 102
retard_croissance, 103

saf_cameroun_departements, 104
saf_cedeao, 105
saf_cemac, 106
saf_eau, 106
saf_pays_afrique, 107
saf_rca_prefectures, 107
saf_sadc, 108
saf_senegal_regions, 108
saf_subdivisions_afrique, 109
sante_mentale, 109
satisfaction_vie, 110
sentiment_securite, 111
sous_emploi_temps, 112
standardiser_ages, 113
stat_descr, *12*, 114
suivre_revisions_pib, 115
supprimer_doublons, 116

tab_croisee, *12*, 117
table_concordance, 118
tableau_bien_etre_subjectif, 119
tableau_bord_pib, 120
tableau_croise_ins, 121
tableau_descriptif, 122
tableau_fgt, 123
tableau_genre, 124
tableau_inegalites, 125
tableau_ipm, 126
tableau_marche_travail, 127
tableau_odd, 128
tableau_sante, 129
taux_activite, 130
taux_croissance, 131
taux_emploi, 132
taux_sous_utilisation, 133
theil *(*decomposer_theil*)*, 46
theme_ins, 134
tracer_flux_traitement, 135
travail_enfants, 136

vaccination, 137
valider_coherence_geo, 138
valider_dictionnaire, *71*, 139
valider_poids, 140
valider_qualite_donnees, 140
valider_statistique_ins, 141
violence_basee_genre, 143

whipple, 144