

Package: humind (via r-universe)

September 22, 2026

Title Compose MSNA Indicators

Version 2026.4.1

Description Compose usual humanitarian composite indicators related to IMPACT Multi-Sector Needs Assessments (MSNAs) such as food sec indicators (rCSI, LCSi, HHS, FCS), Washington Group Short Set of questions, JMP ladders, as well as specific indicators related to the MSNI framework.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports dplyr (>= 1.2.0), glue, purrr, rlang (>= 0.4.11), stats, tidyr, Hmisc, stringr, cli, checkmate

URL <https://impact-initiatives-hppu.github.io/humind/>,
<https://github.com/impact-initiatives-hppu/humind>

BugReports <https://github.com/impact-initiatives-hppu/humind/issues>

Suggests knitr, rmarkdown, roxygen2, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

LazyData true

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository <https://impact-initiatives-hppu.r-universe.dev>

Date/Publication 2026-09-22 11:38:54 UTC

RemoteUrl <https://github.com/impact-initiatives-hppu/humind>

RemoteRef HEAD

RemoteSha 07e30e9ca2b2ac0be977c675ee7b37415b44b821

Contents

add_access_to_phone_best	3
add_age_cat	5
add_barriers_assistance_any	6
add_comp_edu	7
add_comp_foodsec	8
add_comp_health	9
add_comp_prot	10
add_comp_snfi	11
add_comp_wash	14
add_drinking_water_source_cat	17
add_expenditure_type_freq_rank	20
add_expenditure_type_infreq_rank	21
add_expenditure_type_prop_freq	22
add_expenditure_type_prop_infreq	23
add_expenditure_type_zero_freq	24
add_expenditure_type_zero_infreq	25
add_fclcm_phase	26
add_fcm_phase	28
add_fcs	30
add_fds_cannot_cat	31
add_handwashing_facility_cat	33
add_hhs	36
add_hoh_final	38
add_hwise	39
add_income_source_prop	41
add_income_source_rank	42
add_income_source_zero_to_sl	43
add_lcsi	44
add_loop_age_dummy	46
add_loop_edu_access_d	48
add_loop_edu_barrier_protection_d	49
add_loop_edu_disrupted_d	51
add_loop_edu_ind_age_corrected	52
add_loop_healthcare_needed_cat	54
add_loop_wgq_ss	56
add_msni	58
add_occupancy_cat	60
add_prot_score_movement	61
add_prot_score_practices	63
add_prot_score_rights	65
add_quantile_interval	67
add_rcsi	68
add_received_assistance	69
add_sanitation_facility_cat	70
add_shelter_damage_cat	73
add_shelter_issue_cat	74

add_shelter_type_cat	75
are_cols_numeric	76
are_values_in_range	77
are_values_in_set	77
count_if_above_zero	78
drop_shared_loop_cols	79
humind_edu_ind	79
humind_health_ind	80
humind_main	80
if_not_in_stop	81
impute_value	82
is_in_need	83
num_cat	83
rank_top3_vars	84
subvec_in	85
subvec_not_in	86
sum_vars	86
value_to_sl	87

Index 88

add_access_to_phone_best

Add Best Phone Access and Coverage Information

Description

This function recodes the best type of phone accessible in the household and the coverage network type. It provides two main functions:

1. `add_access_to_phone_best()`: Recodes the best type of phone accessible in the household.
2. `add_access_to_phone_coverage()`: Recodes the coverage network type and the best type of phone owned in the household.

Usage

```
add_access_to_phone_best(
  df,
  access_to_phone = "etc_access_to_phone",
  none = "none",
  smartphone = "smartphone",
  feature_phone = "feature_phone",
  basic_phone = "basic_phone",
  dnk = "dnk",
  pnta = "pnta",
  sep = "/"
)
```

```

add_access_to_phone_coverage(
  df,
  coverage_network_type = "etc_coverage_network_type",
  coverage_none = "no_coverage",
  coverage_no_internet = c("only_sms", "voice_sms", "only_voice"),
  coverage_yes_internet = c("voice_sms_internet", "only_internet"),
  coverage_undefined = c("dnk", "pnta"),
  access_to_phone_best = "etc_access_to_phone_best",
  access_to_phone_none = "none",
  access_to_basic_phone = "basic_phone",
  access_to_feature_phone = "feature_phone",
  access_to_smartphone = "smartphone",
  access_to_undefined = "undefined"
)

```

Arguments

<code>df</code>	The input data frame
<code>access_to_phone</code>	The name of the column that contains the access to phone information.
<code>none</code>	The value for no phone.
<code>smartphone</code>	The value for smartphone.
<code>feature_phone</code>	The value for feature phone.
<code>basic_phone</code>	The value for basic phone.
<code>dnk</code>	The value for do not know.
<code>pnta</code>	The value for prefer not to answer.
<code>sep</code>	Separator for the binary columns.
<code>coverage_network_type</code>	Network coverage column.
<code>coverage_none</code>	Value for no coverage.
<code>coverage_no_internet</code>	Vector of values for no internet.
<code>coverage_yes_internet</code>	Value for yes internet.
<code>coverage_undefined</code>	Vector of values for do not know.
<code>access_to_phone_best</code>	The name of the column that contains the access to phone information.
<code>access_to_phone_none</code>	Value for no phone.
<code>access_to_basic_phone</code>	Value for basic phone.
<code>access_to_feature_phone</code>	Value for feature phone.

access_to_smartphone
Value for smartphone.

access_to_undefined
Value for undefined.

Value

A data frame with an additional column 'etc_access_to_phone_best' containing the recoded best type of phone accessible.

A data frame with an additional column 'etc_access_to_phone_coverage' containing the recoded coverage and phone access information.

add_age_cat	<i>Categorize Age and Create Age-Related Variables</i>
-------------	--

Description

This function provides two main functionalities:

1. `add_age_cat()`: Adds age categories to a dataframe based on specified breaks.
2. `add_age_18_cat()`: Adds age categories and a dummy variable for below and above 18 years old.

Usage

```
add_age_cat(
  df,
  age_col,
  breaks = c(0, 18, 60, 120),
  labels = NULL,
  int_undefined = c(-999, 999),
  char_undefined = "undefined",
  new_colname = NULL
)

add_age_18_cat(
  df,
  age_col,
  int_undefined = c(-999, 999),
  char_undefined = "undefined",
  new_colname = NULL
)
```

Arguments

df	A dataframe.
age_col	The column name to recategorize.
breaks	A vector of cut points.
labels	A vector of labels. If NULL, the labels will be the breaks.
int_undefined	A vector of values undefined (such as -999, 999) to replace by char_undefined.
char_undefined	A character to replace int_undefined values, often values corresponding to Don't know or Prefer not to answer.
new_colname	The name of the new column. If NULL, it adds "_cat" to the age_col (or "_18_cat" for add_age_18_cat()).

Value

- For add_age_cat(): A dataframe with an additional column containing the age categories.
- For add_age_18_cat(): A dataframe with two additional columns: one with categories (below_18, above_18) and one with a dummy variable (0 for below 18, 1 for above 18).

add_barriers_assistance_any

Any barriers to accessing humanitarian assistance encountered

Description

This function recodes any barriers to accessing humanitarian assistance encountered from the barriers to accessing humanitarian assistance by type indicator.

Usage

```
add_barriers_assistance_any(
  df,
  barriers_assistance = "aap_barriers_assistance",
  none = "none",
  pnta = "pnta",
  dnk = "dnk"
)
```

Arguments

df	The input data frame
barriers_assistance	The name of the column that contains the information about the barriers to accessing humanitarian assistance.
none	The choice value that indicates that no barriers were encountered.
pnta	The value value that indicates that the person prefers not to answer.
dnk	The value value that indicates that the person does not know.

Value

The data frame with the new variable aap_barriers_assistance_any_barrier

add_comp_edu	<i>Calculate Education Sectoral Composite Score and Need Indicators</i>
--------------	---

Description

This function calculates an education sectoral composite score based on various education-related indicators. It computes scores for disrupted education and attendance/barriers, combines them into a total score, and determines if a household is in need or in severe need of educational assistance.

Prerequisite functions:

- add_loop_age_dummy.R
- add_loop_edu_access_d.R
- add_loop_edu_barrier_protection_d.R
- add_loop_edu_disrupted_d.R
- add_loop_edu_ind_age_corrected.R

Usage

```
add_comp_edu(
  df,
  schooling_age_n = "edu_schooling_age_n",
  no_access_n = "edu_no_access_n",
  barrier_protection_n = "edu_barrier_protection_n",
  attack_n = "edu_disrupted_attack_n",
  hazards_n = "edu_disrupted_hazards_n",
  displaced_n = "edu_disrupted_displaced_n",
  teacher_n = "edu_disrupted_teacher_n"
)
```

Arguments

df	A data frame.
schooling_age_n	Column name for the number of children of schooling age.
no_access_n	Column name for the number of children with no access to education.
barrier_protection_n	Column name for the number of children with barriers to protection.
attack_n	Column name for the number of children with disrupted education due to the school being occupied by armed groups
hazards_n	Column name for the number of children with disrupted education due to hazards.

displaced_n	Column name for the number of children with disrupted education due to a recent displacement.
teacher_n	Column name for the number of children with disrupted education due to teachers' absence.

Value

A data frame with additional columns:

- comp_edu_score_disrupted: Score for disrupted education (1-4).
- comp_edu_score_attendance: Score for attendance and barriers (1-4)
- comp_edu_score: Total education composite score (max of disrupted and attendance scores).
- comp_edu_in_need: Binary indicator for being in need of educational assistance.
- comp_edu_in_severe_need: Binary indicator for being in severe need of educational assistance.

add_comp_foodsec	<i>Calculate Food Security Sectoral Composite Score and Need Indicators</i>
------------------	---

Description

This function calculates a food security sectoral composite score based on the FCLCM phase. It assigns a score from 1 to 5 corresponding to the FCLCM phase, and determines if a household is in need or in severe need of food security assistance.

Prerequisite food security functions must be run before this function:

- add_fcs()
- add_hhs()
- add_rcsi()
- add_lcsi()
- add_fclcm_phase()

Usage

```
add_comp_foodsec(
  df,
  fclcm_phase = "fclcm_phase",
  phase1 = "Phase 1 FCLC",
  phase2 = "Phase 2 FCLC",
  phase3 = "Phase 3 FCLC",
  phase4 = "Phase 4 FCLC",
  phase5 = "Phase 5 FCLC"
)
```

Arguments

df	A data frame.
fclcm_phase	Column name for the FCLCM phase.
phase1	Label for Phase 1 FCLC.
phase2	Label for Phase 2 FCLC.
phase3	Label for Phase 3 FCLC.
phase4	Label for Phase 4 FCLC.
phase5	Label for Phase 5 FCLC.

Value

A data frame with additional columns:

- comp_foodsec_score: Food security composite score (1-5)
- comp_foodsec_in_need: Binary indicator for being in need of food security assistance
- comp_foodsec_in_severe_need: Binary indicator for being in severe need of food security assistance

add_comp_health	<i>Calculate Health Composite Score and Need Indicators</i>
-----------------	---

Description

This function calculates a health composite score based on healthcare access and needs. It considers whether individuals needed healthcare, whether they accessed it if needed, and optionally takes into account disability status. The function then determines if a household is in need or in severe need of health assistance based on the calculated score.

Prerequisite functions:

- add_loop_healthcare_needed_cat.R

Usage

```
add_comp_health(
  df,
  ind_healthcare_needed_no_n = "health_ind_healthcare_needed_no_n",
  ind_healthcare_needed_yes_unmet_n = "health_ind_healthcare_needed_yes_unmet_n",
  ind_healthcare_needed_yes_met_n = "health_ind_healthcare_needed_yes_met_n"
)
```

Arguments

`df` A data frame.

`ind_healthcare_needed_no_n` Column name for the number of individuals who did not need to access healthcare.

`ind_healthcare_needed_yes_unmet_n` Column name for the number of individuals who needed to access healthcare but did not.

`ind_healthcare_needed_yes_met_n` Column name for the number of individuals who needed to access healthcare and did.

Value

A data frame with additional columns:

- `comp_health_score`: Health composite score (1-4)
- `comp_health_in_need`: Binary indicator for being in need of health assistance
- `comp_health_in_severe_need`: Binary indicator for being in severe need of health assistance

<code>add_comp_prot</code>	<i>Add overall Protection composite score and need flags</i>
----------------------------	--

Description

Assumes that per-dimension severity scores have already been computed and added to `df` by earlier functions: `add_prot_score_movement()`, `add_prot_score_practices()`, and `add_prot_score_rights()`. This function performs only **Step 4** of the 2025 Protection Composite workflow: it takes the maximum of those three existing columns to create `comp_prot_score`, then generates binary “in need” indicators.

Usage

```
add_comp_prot(df)
```

Arguments

`df` A `data.frame` or `tibble` containing numeric columns:

- `comp_prot_score_movement` – severity for movement dimension
- `comp_prot_score_practices` – severity for practices dimension
- `comp_prot_score_rights` – severity for rights & services dimension

If any of these three columns are missing, the function will abort, reminding you to run the corresponding prep functions first.

Details

- **Column checks** via `purrr::iwalk()` ensure the three dimension scores exist.
- **Computation** uses `pmax()` (with `na.rm = FALSE`)
- **Thresholds** (3 for “in need”, 4 for “severe need”) are currently hard-coded.

Value

The input `df`, with three new columns:

- `comp_prot_score` – overall protection severity (maximum of the three dimensions)
- `comp_prot_in_need` – binary (0/1): 1 if `comp_prot_score` $\geq$ 3, else 0
- `comp_prot_in_severe_need` – binary (0/1): 1 if `comp_prot_score` $\geq$ 4, else 0

`add_comp_snfi`

Add SNFI Sectoral Composite Score and Need Indicators

Description

This function calculates the Shelter, NFI and HLP (SNFI) sectoral composite score based on shelter type, shelter issues, security of tenure, shelter damage, and functional disability scale (FDS) indicators. It also determines if a household is in need or in severe need based on the calculated score.

Prerequisite functions:

- `add_shelter_issue_cat.R`
- `add_shelter_type_cat.R`
- `add_occupancy_cat.R`
- `add_fds_cannot_cat.R`
- `add_shelter_damage_cat.R`

Usage

```
add_comp_snfi(
  df,
  shelter_type_cat = "snfi_shelter_type_cat",
  shelter_type_cat_none = "none",
  shelter_type_cat_inadequate = "inadequate",
  shelter_type_cat_adequate = "adequate",
  shelter_type_cat_undefined = "undefined",
  shelter_issue_cat = "snfi_shelter_issue_cat",
  shelter_issue_cat_1_to_3 = "1_to_3",
  shelter_issue_cat_4_to_7 = "4_to_7",
  shelter_issue_cat_8_to_11 = "8_to_11",
  shelter_issue_cat_none = "none",
```

```

shelter_issue_cat_undefined = "undefined",
shelter_issue_cat_other = "other",
tenure_security_cat = "hlp_tenure_security",
tenure_security_cat_high_risk = "high_risk",
tenure_security_cat_medium_risk = "medium_risk",
tenure_security_cat_low_risk = "low_risk",
tenure_security_cat_undefined = "undefined",
fds_cannot_cat = "snfi_fds_cannot_cat",
fds_cannot_cat_4 = "4_tasks",
fds_cannot_cat_2_to_3 = "2_to_3_tasks",
fds_cannot_cat_1 = "1_task",
fds_cannot_cat_none = "none",
fds_cannot_cat_undefined = "undefined",
shelter_damage_cat = "snfi_shelter_damage_cat",
shelter_damage_cat_none = "none",
shelter_damage_cat_damaged = "damaged",
shelter_damage_cat_part = "part",
shelter_damage_cat_total = "total",
shelter_damage_cat_undefined = "undefined"
)

```

Arguments

df A data frame containing the required SNFI indicators.

shelter_type_cat Column name for shelter type.

shelter_type_cat_none Level for no shelter.

shelter_type_cat_inadequate Level for inadequate shelter.

shelter_type_cat_adequate Level for adequate shelter.

shelter_type_cat_undefined Level for undefined shelter.

shelter_issue_cat Column name for shelter issue.

shelter_issue_cat_1_to_3 Level for 1 to 3 shelter issues.

shelter_issue_cat_4_to_7 Level for 4 to 7 shelter issues.

shelter_issue_cat_8_to_11 Level for 8 to 11 shelter issues.

shelter_issue_cat_none Level for no shelter issues.

shelter_issue_cat_undefined Level for undefined shelter issues.

shelter_issue_cat_other Level for other shelter issues.

tenure_security_cat
Column name for security of tenure.

tenure_security_cat_high_risk
Level for high risk with security of tenure.

tenure_security_cat_medium_risk
Level for medium risk with security of tenure.

tenure_security_cat_low_risk
Level for low risk with security of tenure.

tenure_security_cat_undefined
Level for undefined with security of tenure.

fds_cannot_cat Column name for fds cannot.

fds_cannot_cat_4
Level for 4 tasks that cannot be done.

fds_cannot_cat_2_to_3
Level for 2 to 3 tasks that cannot be done.

fds_cannot_cat_1
Level for 1 task that cannot be done.

fds_cannot_cat_none
Level for no tasks that cannot be done.

fds_cannot_cat_undefined
Level for undefined fds cannot.

shelter_damage_cat
Column for shelter damage category.

shelter_damage_cat_none
Level name for no shelter damage.

shelter_damage_cat_damaged
Level name for minor damages.

shelter_damage_cat_part
Level name for roof with risk of collapse.

shelter_damage_cat_total
Level name for total collapse or destruction.

shelter_damage_cat_undefined
Level name for undefined or unknown status for shelter damage.

Value

A data frame with added columns:

- comp_snfi_score_shelter_type_cat Score based on shelter type
- comp_snfi_score_shelter_issue_cat Score based on shelter issues
- comp_snfi_score_tenure_security_cat Score based on security of tenure status
- comp_snfi_score_fds_cannot_cat Score based on FDS
- comp_snfi_score_shelter_damage_cat Score based on shelter damage
- comp_snfi_score: Overall SNFI composite score
- comp_snfi_in_need: Indicator for being in need
- comp_snfi_in_severe_need: Indicator for being in severe need

add_comp_wash

*Add WASH Sectoral Composite Score and Need Indicator***Description**

Calculates a comprehensive WASH (Water, Sanitation, and Hygiene) composite score for households across different settings (camp, urban, rural). The function generates sub-scores for water quantity, water quality, sanitation, and hygiene, and creates a dummy variable to indicate households in need of WASH assistance.

The function performs detailed scoring based on multiple WASH-related variables, applying different scoring logic for camp, urban, and rural settings.

Prerequisite functions:

- `add_hwise()`
- `add_sanitation_facility_cat()`
- `add_handwashing_facility_cat()`
- `add_drinking_water_source_cat()`

Usage

```
add_comp_wash(
  df,
  setting = "setting",
  setting_camp = c("camp_formal", "camp_informal"),
  setting_urban = "urban",
  setting_rural = "rural",
  comp_wash_hwise_score = "comp_wash_score_water_quantity",
  drinking_water_quality_jmp_cat = "wash_drinking_water_quality_jmp_cat",
  drinking_water_quality_jmp_cat_surface_water = "surface_water",
  drinking_water_quality_jmp_cat_unimproved = "unimproved",
  drinking_water_quality_jmp_cat_limited = "limited",
  drinking_water_quality_jmp_cat_basic = "basic",
  drinking_water_quality_jmp_cat_safely_managed = "safely_managed",
  drinking_water_quality_jmp_cat_undefined = "undefined",
  sanitation_facility_jmp_cat = "wash_sanitation_facility_jmp_cat",
  sanitation_facility_jmp_cat_open_defecation = "open_defecation",
  sanitation_facility_jmp_cat_unimproved = "unimproved",
  sanitation_facility_jmp_cat_limited = "limited",
  sanitation_facility_jmp_cat_basic = "basic",
  sanitation_facility_jmp_cat_undefined = "undefined",
  sanitation_facility_cat = "wash_sanitation_facility_cat",
  sanitation_facility_cat_none = "none",
  sanitation_facility_cat_unimproved = "unimproved",
  sanitation_facility_cat_improved = "improved",
  sanitation_facility_cat_undefined = "undefined",
```

```

sanitation_facility_n_ind = "wash_sharing_sanitation_facility_n_ind",
sanitation_facility_n_ind_50_and_above = "50_and_above",
sanitation_facility_n_ind_20_to_49 = "20_to_49",
sanitation_facility_n_ind_19_and_below = "19_and_below",
sharing_sanitation_facility_cat = "wash_sharing_sanitation_facility_cat",
sharing_sanitation_facility_cat_shared = "shared",
sharing_sanitation_facility_cat_not_shared = "not_shared",
sharing_sanitation_facility_cat_not_applicable = "not_applicable",
sharing_sanitation_facility_cat_undefined = "undefined",
handwashing_facility_jmp_cat = "wash_handwashing_facility_jmp_cat",
handwashing_facility_jmp_cat_no_facility = "no_facility",
handwashing_facility_jmp_cat_limited = "limited",
handwashing_facility_jmp_cat_basic = "basic",
handwashing_facility_jmp_cat_undefined = "undefined"
)

```

Arguments

df	A data frame containing the required WASH-related variables.
setting	Column name for the setting (camp, urban, or rural).
setting_camp	Setting value(s) for camp.
setting_urban	Setting value for urban.
setting_rural	Setting value for rural.
comp_wash_hwise_score	Column name for the pre-computed HWISE water quantity severity score (1–5), as produced by add_hwise() .
drinking_water_quality_jmp_cat	Column name for drinking water quality JMP category.
drinking_water_quality_jmp_cat_surface_water	Value for "surface water" in drinking water quality JMP category.
drinking_water_quality_jmp_cat_unimproved	Value for "unimproved" in drinking water quality JMP category.
drinking_water_quality_jmp_cat_limited	Value for "limited" in drinking water quality JMP category.
drinking_water_quality_jmp_cat_basic	Value for "basic" in drinking water quality JMP category.
drinking_water_quality_jmp_cat_safely_managed	Value for "safely managed" in drinking water quality JMP category.
drinking_water_quality_jmp_cat_undefined	Value for "undefined" in drinking water quality JMP category.
sanitation_facility_jmp_cat	Column name for sanitation facility JMP category.
sanitation_facility_jmp_cat_open_defecation	Value for "open defecation" in sanitation facility JMP category.
sanitation_facility_jmp_cat_unimproved	Value for "unimproved" in sanitation facility JMP category.

sanitation_facility_jmp_cat_limited
 Value for "limited" in sanitation facility JMP category.

sanitation_facility_jmp_cat_basic
 Value for "basic" in sanitation facility JMP category.

sanitation_facility_jmp_cat_undefined
 Value for "undefined" in sanitation facility JMP category.

sanitation_facility_cat
 Column name for sanitation facility category.

sanitation_facility_cat_none
 Value for "none" in sanitation facility category.

sanitation_facility_cat_unimproved
 Value for "unimproved" in sanitation facility category.

sanitation_facility_cat_improved
 Value for "improved" in sanitation facility category.

sanitation_facility_cat_undefined
 Value for "undefined" in sanitation facility category.

sanitation_facility_n_ind
 Column name for number of individuals using the sanitation facility.

sanitation_facility_n_ind_50_and_above
 Value for "50 and above" in number of individuals using the sanitation facility.

sanitation_facility_n_ind_20_to_49
 Value for "20 to 49" in number of individuals using the sanitation facility.

sanitation_facility_n_ind_19_and_below
 Value for "19 and below" in number of individuals using the sanitation facility.

sharing_sanitation_facility_cat
 Column name for sharing a sanitation facility.

sharing_sanitation_facility_cat_shared
 Value for "shared" in sharing a sanitation facility.

sharing_sanitation_facility_cat_not_shared
 Value for "not shared" in sharing a sanitation facility.

sharing_sanitation_facility_cat_not_applicable
 Value for "not applicable" in sharing a sanitation facility.

sharing_sanitation_facility_cat_undefined
 Value for "undefined" in sharing a sanitation facility.

handwashing_facility_jmp_cat
 Column name for handwashing facility JMP category.

handwashing_facility_jmp_cat_no_facility
 Value for "no facility" in handwashing facility JMP category.

handwashing_facility_jmp_cat_limited
 Value for "limited" in handwashing facility JMP category.

handwashing_facility_jmp_cat_basic
 Value for "basic" in handwashing facility JMP category.

handwashing_facility_jmp_cat_undefined
 Value for "undefined" in handwashing facility JMP category.

Value

A data frame with new columns:

- comp_wash_score_water_quantity: Numeric score for water quantity (1-5).
- comp_wash_score_water_quality: Numeric score for water quality (1-5).
- comp_wash_score_sanitation: Numeric score for sanitation (1-5).
- comp_wash_score_hygiene: Numeric score for hygiene (1-5).
- comp_wash_score: Overall WASH composite score (0-20).
- comp_wash_in_need: Binary indicator (0 or 1) for households in need of WASH assistance.

add_drinking_water_source_cat

Add Drinking Water Quality JMP Category

Description

This function recodes the water source and time to fetch water into a joint JMP (Joint Monitoring Programme) category.

Usage

```
add_drinking_water_source_cat(
  df,
  drinking_water_source = "wash_drinking_water_source",
  drinking_water_source_cat_improved = c("piped_dwelling", "piped_compound",
    "piped_neighbour", "tap", "borehole", "protected_well", "well_spring",
    "rainwater_collection", "tank_truck", "cart_tank", "kiosk", "bottled_water",
    "sachet_water"),
  drinking_water_source_cat_unimproved = c("unprotected_well", "unprotected_spring"),
  drinking_water_source_cat_surface_water = "surface_water",
  drinking_water_source_cat_undefined = c("dnk", "pnta", "other")
)

add_drinking_water_time_cat(
  df,
  drinking_water_time_yn = "wash_drinking_water_time_yn",
  water_on_premises = c("water_in_dwelling", "water_in_plot"),
  number_minutes = "number_minutes",
  dnk = "dnk",
  undefined = "pnta",
  drinking_water_time_int = "wash_drinking_water_time_int",
  max_minutes = 600,
  drinking_water_time_sl = "wash_drinking_water_time_sl",
  sl_under_30_min = c("5min_or_less", "5min_15min", "15min_30min"),
  sl_30min_1hr = "30min_1hr",
```

```

sl_more_than_1hr = "more_than_1hr",
sl_undefined = c("dnk", "pnta"),
drinking_water_source = "wash_drinking_water_source",
skipped_drinking_water_source_premises = "piped_dwelling",
skipped_drinking_water_source_undefined = c("dnk", "pnta")
)

add_drinking_water_time_threshold_cat(
  df,
  drinking_water_time_30min_cat = "wash_drinking_water_time_cat",
  drinking_water_time_30min_cat_premises = "premises",
  drinking_water_time_30min_cat_under_30min = c("under_30_min"),
  drinking_water_time_30min_cat_above_30min = c("30min_1hr", "more_than_1hr"),
  drinking_water_time_30min_cat_undefined = "undefined"
)

add_drinking_water_quality_jmp_cat(
  df,
  drinking_water_source_cat = "wash_drinking_water_source_cat",
  drinking_water_source_cat_improved = "improved",
  drinking_water_source_cat_unimproved = "unimproved",
  drinking_water_source_cat_surface_water = "surface_water",
  drinking_water_source_cat_undefined = "undefined",
  drinking_water_time_30min_cat = "wash_drinking_water_time_30min_cat",
  drinking_water_time_30min_cat_premises = "premises",
  drinking_water_time_30min_cat_under_30min = "under_30min",
  drinking_water_time_30min_cat_above_30min = "above_30min",
  drinking_water_time_30min_cat_undefined = "undefined"
)

```

Arguments

df A data frame.

drinking_water_source Component column: Water source types.

drinking_water_source_cat_improved Response code for improved water source.

drinking_water_source_cat_unimproved Response code for unimproved water source.

drinking_water_source_cat_surface_water Response code for surface water source.

drinking_water_source_cat_undefined Response code for undefined water source.

drinking_water_time_yn Component column: Time to fetch water, scoping question.

water_on_premises Character vector of responses codes for water on premises.

number_minutes	Character vector of responses codes for number of minutes.
dnk	Character vector of responses codes for "Don't know".
undefined	Character vector of responses codes for undefined information, e.g. "Prefer not to answer".
drinking_water_time_int	Component column: Time to fetch water, integer.
max_minutes	Integer, the maximum value for the time to fetch water.
drinking_water_time_sl	Component column: Time to fetch water, simple choice.
sl_under_30_min	Character vector of response codes for under 30 minutes, e.g. c("5min_or_less", "5min_15min", "15min_30min").
sl_30min_1hr	Response code for 30 minutes to 1 hour.
sl_more_than_1hr	Response code for more than 1 hour.
sl_undefined	Character vector of responses codes for undefined information, e.g. "Don't know" or "Prefer not to answer".
skipped_drinking_water_source_premises	Character vector of responses codes for skipped water source on premises, e.g. "Piped into dwelling".
skipped_drinking_water_source_undefined	Character vector of responses codes for skipped water source undefined, e.g. "Don't know" or "Prefer not to answer".
drinking_water_time_30min_cat	Component column: Time to fetch water, recoded categories.
drinking_water_time_30min_cat_premises	Response code for water on premises.
drinking_water_time_30min_cat_under_30min	Response code for under 30 minutes.
drinking_water_time_30min_cat_above_30min	Response code for above 30 minutes.
drinking_water_time_30min_cat_undefined	Response code for undefined time.
drinking_water_source_cat	Component column: Water source categories.

Value

A data frame with a new column 'wash_drinking_water_quality_jmp_cat' containing:

- limited: Response indicating limited access to safe drinking water.
- basic: Response indicating basic access to safe drinking water.
- unimproved: Response indicating unimproved water sources.
- surface_water: Response indicating surface water sources.
- undefined: Response for undefined categories.

`add_expenditure_type_freq_rank`*Rank Top 3 Frequent Expenditure Types*

Description

This function ranks the top 3 most frequent expenditure types for each household based on the amount spent on various categories. It adds new columns to the input data frame indicating the top 3 expenditure types.

Prerequisite function:

- `add_expenditure_type_zero_freq.R`

Usage

```
add_expenditure_type_freq_rank(  
  df,  
  expenditure_freq_types = c("cm_expenditure_frequent_food",  
    "cm_expenditure_frequent_rent", "cm_expenditure_frequent_water",  
    "cm_expenditure_frequent_nfi", "cm_expenditure_frequent_utilities",  
    "cm_expenditure_frequent_fuel", "cm_expenditure_frequent_transportation",  
    "cm_expenditure_frequent_communication", "cm_expenditure_frequent_other"),  
  id_col = "uuid"  
)
```

Arguments

<code>df</code>	A data frame containing expenditure data for households.
<code>expenditure_freq_types</code>	A character vector. The names of the columns that contain the amount of frequent expenditures types.
<code>id_col</code>	The name of the column that contains the unique identifier.

Value

A data frame with additional columns:

- `cm_freq_expenditure_top1`: The most frequent expenditure type.
- `cm_freq_expenditure_top2`: The second most frequent expenditure type.
- `cm_freq_expenditure_top3`: The third most frequent expenditure type.

`add_expenditure_type_infreq_rank`*Rank Top 3 Infrequent Expenditure Types*

Description

This function ranks the top 3 most infrequent expenditure types for each household based on the amount spent on various categories. It adds new columns to the input data frame indicating the top 3 infrequent expenditure types.

Prerequisite function:

- `add_expenditure_type_zero_infreq.R`

Usage

```
add_expenditure_type_infreq_rank(  
  df,  
  expenditure_infreq_types = c("cm_expenditure_infrequent_shelter",  
    "cm_expenditure_infrequent_nfi", "cm_expenditure_infrequent_health",  
    "cm_expenditure_infrequent_education", "cm_expenditure_infrequent_debt",  
    "cm_expenditure_infrequent_clothing", "cm_expenditure_infrequent_other"),  
  id_col = "uuid"  
)
```

Arguments

<code>df</code>	A data frame containing infrequent expenditure data for households.
<code>expenditure_infreq_types</code>	A character vector. The names of the columns that contain the amount of infrequent expenditures types.
<code>id_col</code>	The name of the column that contains the unique identifier.

Details

Rank Top 3 Infrequent Expenditure Types

Value

A data frame with additional columns:

- `cm_infreq_expenditure_top1`: The most infrequent expenditure type.
- `cm_infreq_expenditure_top2`: The second most infrequent expenditure type.
- `cm_infreq_expenditure_top3`: The third most infrequent expenditure type.

add_expenditure_type_prop_freq

Add Frequent Expenditure Type Amount as Proportions of Total Frequent Expenditure

Description

This function calculates the proportion of each frequent expenditure type relative to the total frequent expenditure. It adds new columns to the input data frame representing these proportions.

Prerequisite function:

- add_expenditure_type_zero_freq.R

Usage

```
add_expenditure_type_prop_freq(
  df,
  cm_expenditure_frequent_food = "cm_expenditure_frequent_food",
  cm_expenditure_frequent_rent = "cm_expenditure_frequent_rent",
  cm_expenditure_frequent_water = "cm_expenditure_frequent_water",
  cm_expenditure_frequent_nfi = "cm_expenditure_frequent_nfi",
  cm_expenditure_frequent_utilities = "cm_expenditure_frequent_utilities",
  cm_expenditure_frequent_fuel = "cm_expenditure_frequent_fuel",
  cm_expenditure_frequent_transportation = "cm_expenditure_frequent_transportation",
  cm_expenditure_frequent_communication = "cm_expenditure_frequent_communication",
  cm_expenditure_frequent_other = "cm_expenditure_frequent_other"
)
```

Arguments

df	A data frame containing frequent expenditure data.
cm_expenditure_frequent_food	Column name for food items expenditure amount.
cm_expenditure_frequent_rent	Column name for rent or shelter and/or land expenditure amount.
cm_expenditure_frequent_water	Column name for water expenditure amount from all sources combined.
cm_expenditure_frequent_nfi	Column name for non-food household items expenditure amount for regular purchase, such as hygiene items, lightbulbs, etc.
cm_expenditure_frequent_utilities	Column name for utilities expenditure amount, such as electricity or gas connections, etc.
cm_expenditure_frequent_fuel	Column name for fuel expenditure amount for cooking, for vehicles, etc.

cm_expenditure_frequent_transportation
 Column name for transportation expenditure amount, not including vehicle fuel.

cm_expenditure_frequent_communication
 Column name for communications expenditure amount, such as phone airtime, Internet costs, etc.

cm_expenditure_frequent_other
 Column name for all other frequent expenditures.

Value

A data frame with additional columns:

- cm_expenditure_frequent_total: The total frequent expenditure amount.
- cm_expenditure_frequent_food_prop: Proportion of food expenditure.
- cm_expenditure_frequent_rent_prop: Proportion of rent expenditure.
- cm_expenditure_frequent_water_prop: Proportion of water expenditure.
- cm_expenditure_frequent_nfi_prop: Proportion of non-food items expenditure.
- cm_expenditure_frequent_utilities_prop: Proportion of utilities expenditure.
- cm_expenditure_frequent_fuel_prop: Proportion of fuel expenditure.
- cm_expenditure_frequent_transportation_prop: Proportion of transportation expenditure.
- cm_expenditure_frequent_communication_prop: Proportion of communication expenditure.
- cm_expenditure_frequent_other_prop: Proportion of other frequent expenditures.

add_expenditure_type_prop_infreq

Add Infrequent Expenditure Type Proportions

Description

This function calculates the proportion of each infrequent expenditure type relative to the total infrequent expenditure. It adds new columns to the input data frame representing these proportions.

Prerequisite function:

- add_expenditure_type_zero_infreq.R

Usage

```
add_expenditure_type_prop_infreq(
  df,
  cm_expenditure_infrequent_shelter = "cm_expenditure_infrequent_shelter",
  cm_expenditure_infrequent_nfi = "cm_expenditure_infrequent_nfi",
  cm_expenditure_infrequent_health = "cm_expenditure_infrequent_health",
  cm_expenditure_infrequent_education = "cm_expenditure_infrequent_education",
  cm_expenditure_infrequent_debt = "cm_expenditure_infrequent_debt",
  cm_expenditure_infrequent_clothing = "cm_expenditure_infrequent_clothing",
  cm_expenditure_infrequent_other = "cm_expenditure_infrequent_other"
)
```

Arguments

df	A data frame containing infrequent expenditure data.
cm_expenditure_infrequent_shelter	Column name for shelter maintenance or repair expenditure amount.
cm_expenditure_infrequent_nfi	Column containing expenditure amount for non-food household items for infrequent purchase, such as blankets, cooking pots, clothing, etc.
cm_expenditure_infrequent_health	Column name for health-related expenditures, such as healthcare, medicine, etc.
cm_expenditure_infrequent_education	Column name for education-related expenditures, such school fees, supplies, uniforms, etc.
cm_expenditure_infrequent_debt	Column containing expenditure amount for debt repayment.
cm_expenditure_infrequent_clothing	Column containing expenditure amount for clothing.
cm_expenditure_infrequent_other	Column name for all other infrequent expenditures.

Value

A data frame with additional columns:

- cm_expenditure_infrequent_total: The total infrequent expenditure amount.
- cm_expenditure_infrequent_shelter_prop: Proportion of shelter maintenance or repair expenditure.
- cm_expenditure_infrequent_nfi_prop: Proportion of non-food household items expenditure.
- cm_expenditure_infrequent_health_prop: Proportion of health-related expenditures.
- cm_expenditure_infrequent_education_prop: Proportion of education-related expenditures.
- cm_expenditure_infrequent_debt_prop: Proportion of debt repayment expenditure.
- cm_expenditure_infrequent_clothing_prop: Proportion of clothing-related expenditure.
- cm_expenditure_infrequent_other_prop: Proportion of other infrequent expenditures.

add_expenditure_type_zero_freq

Add Zero for Skipped Frequent Expenditure Types

Description

This function adds zero values to frequent expenditure types that were skipped in the survey. Frequent expenditures are defined as those with a 30-day recall period. The function also ensures that when no expenditure is reported, all expenditure types are set to zero.

Usage

```
add_expenditure_type_zero_freq(
  df,
  expenditure_freq = "cm_expenditure_frequent",
  none = "none",
  undefined = c("dnk", "pnta"),
  expenditure_freq_types = c("cm_expenditure_frequent_food",
    "cm_expenditure_frequent_rent", "cm_expenditure_frequent_water",
    "cm_expenditure_frequent_nfi", "cm_expenditure_frequent_utilities",
    "cm_expenditure_frequent_fuel", "cm_expenditure_frequent_transportation",
    "cm_expenditure_frequent_communication", "cm_expenditure_frequent_other")
)
```

Arguments

<code>df</code>	A data frame containing expenditure data.
<code>expenditure_freq</code>	A character string. The name of the column that contains the frequent expenditures.
<code>none</code>	The value for no expenditure.
<code>undefined</code>	A character vector. The values that indicate that the frequent expenditures type was skipped.
<code>expenditure_freq_types</code>	A character vector. The names of the columns that contain the amount of frequent expenditures types.

Value

A data frame with updated expenditure columns: columns specified in `expenditure_freq_types` are updated with zeros for skipped entries.

```
add_expenditure_type_zero_infreq
```

Add Zero for Skipped Infrequent Expenditure Types

Description

This function adds zero values to infrequent expenditure types that were skipped in the survey. Infrequent expenditures are defined as those with a 6-month recall period. The function also ensures that when no expenditure is reported, all expenditure types are set to zero.

Usage

```
add_expenditure_type_zero_infreq(
  df,
  expenditure_infreq = "cm_expenditure_frequent",
  none = "none",
  undefined = c("dnk", "pnta"),
  expenditure_infreq_types = c("cm_expenditure_infrequent_shelter",
    "cm_expenditure_infrequent_nfi", "cm_expenditure_infrequent_health",
    "cm_expenditure_infrequent_education", "cm_expenditure_infrequent_debt",
    "cm_expenditure_infrequent_clothing", "cm_expenditure_infrequent_other")
)
```

Arguments

df A data frame containing expenditure data.

expenditure_infreq A character string. The name of the column that contains the infrequent expenditures.

none The value for no expenditure.

undefined A character vector. The values that indicate that the infrequent expenditures type was skipped.

expenditure_infreq_types A character vector. The names of the columns that contain the amount of infrequent expenditures types.

Value

A data frame with updated expenditure columns: columns specified in `expenditure_infreq_types` are updated with zeros for skipped entries.

add_fclcm_phase

Calculating the Food Consumption-Livelihood Coping Matrix

Description

Calculating the Food Consumption-Livelihood Coping Matrix

Usage

```
add_fclcm_phase(
  .dataset,
  fc_phase_var = "fsl_fc_phase",
  fc_phase_1 = "Phase 1 FC",
  fc_phase_2 = "Phase 2 FC",
  fc_phase_3 = "Phase 3 FC",
  fc_phase_4 = "Phase 4 FC",
```

```

    fc_phase_5 = "Phase 5 FC",
    lcs_cat_var = "fsl_lcsi_cat",
    lcs_cat_none = "None",
    lcs_cat_stress = "Stress",
    lcs_cat_crisis = "Crisis",
    lcs_cat_emergency = "Emergency"
  )

```

Arguments

.dataset	Main Dataset
fc_phase_var	Column name containing food consumption phase.
fc_phase_1	The name of the value "Phase 1 FC" (by default) in the food consumption phase.
fc_phase_2	The name of the value "Phase 2 FC" (by default) in the food consumption phase.
fc_phase_3	The name of the value "Phase 3 FC" (by default) in the food consumption phase.
fc_phase_4	The name of the value "Phase 4 FC" (by default) in the food consumption phase.
fc_phase_5	The name of the value "Phase 5 FC" (by default) in the food consumption phase.
lcs_cat_var	Column name containing livelihood coping category.
lcs_cat_none	The name of the value "None" (by default) in the livelihood coping category.
lcs_cat_stress	The name of the value "Stress" (by default) in the livelihood coping category.
lcs_cat_crisis	The name of the value "Crisis" (by default) in the livelihood coping category.
lcs_cat_emergency	The name of the value "Emergency" (by default) in the livelihood coping category.

Value

Returns a dataframe with a additional column for FCLC phase.

Examples

```

test_df <- data.frame(
  fsl_lcsi_cat = c("None", "Stress"),
  fsl_fc_phase = c("Phase 1 FC", "Phase 2 FC")
)
test_df |> add_fclcm_phase()

```

add_fcm_phase	<i>Add the food consumption matrix to the dataset</i>
---------------	---

Description

Add the food consumption matrix to the dataset

Usage

```
add_fcm_phase(
  .dataset,
  fcs_column_name = "fsl_fcs_cat",
  rcsi_column_name = "fsl_rcsi_cat",
  hhs_column_name = "fsl_hhs_cat_ipc",
  hdds_column_name = "fsl_hdds_cat",
  fcs_categories_acceptable = "Acceptable",
  fcs_categories_poor = "Poor",
  fcs_categories_borderline = "Borderline",
  rcsi_categories_low = "No to Low",
  rcsi_categories_medium = "Medium",
  rcsi_categories_high = "High",
  hhs_categories_none = "None",
  hhs_categories_little = "Little",
  hhs_categories_moderate = "Moderate",
  hhs_categories_severe = "Severe",
  hhs_categories_very_severe = "Very Severe",
  hdds_categories_low = "Low",
  hdds_categories_medium = "Medium",
  hdds_categories_high = "High"
)
```

Arguments

.dataset	Main Dataset
fcs_column_name	A string specifying the column name of the food consumption score in the dataset
rcsi_column_name	A string specifying the column name of the reduced coping strategy index in the dataset
hhs_column_name	A string specifying the column name of the household hunger scale in the dataset
hdds_column_name	A string specifying the column name of the household dietary diversity score in the dataset
fcs_categories_acceptable	The name of the value "Acceptable" (by default) in the fcs categories

fcs_categories_poor
The name of the value "Poor" (by default) in the fcs categories

fcs_categories_borderline
The name of the value "Borderline" (by default) in the fcs categories

rcsi_categories_low
The name of the value "No to Low" (by default) in the rcsi categories

rcsi_categories_medium
The name of the value "Medium" (by default) in the rcsi categories

rcsi_categories_high
The name of the value "High" (by default) in the rcsi categories

hhs_categories_none
The name of the value "None" (by default) in the hhs categories

hhs_categories_little
The name of the value "Little" (by default) in the hhs categories

hhs_categories_moderate
The name of the value "Moderate" (by default) in the hhs categories

hhs_categories_severe
The name of the value "Severe" (by default) in the hhs categories

hhs_categories_very_severe
The name of the value "Very Severe" (by default) in the hhs categories

hdds_categories_low
The name of the value "Low" (by default) in the hdds categories

hdds_categories_medium
The name of the value "Medium" (by default) in the hdds categories

hdds_categories_high
The name of the value "High" (by default) in the hdds categories

Value

this function returns a dataframe with a column called `fc_cell` that includes values from 1 to 45 representing the Food Consumption Score Matrix and the `fc_phase` column that includes the different 5 phases of food consumption

Examples

```
test_data <- data.frame(
  fsl_fcs_cat = c("Acceptable", "Poor", "Borderline", "Acceptable"),
  fsl_rcsi_cat = c("No to Low", "Medium", "No to Low", "High"),
  fsl_hhs_cat = c("None", "Little", "Severe", "Very Severe"),
  fsl_hdds_cat = c("Low", "High", "Medium", "High")
)
add_fcm_phase(test_data,
  fcs_column_name = "fsl_fcs_cat",
  rcsi_column_name = "fsl_rcsi_cat",
  hhs_column_name = "fsl_hhs_cat",
  hdds_column_name = "fsl_hdds_cat",
  fcs_categories_acceptable = "Acceptable",
  fcs_categories_poor = "Poor",
```

```
fcs_categories_borderline = "Borderline",
rcsi_categories_low = "No to Low",
rcsi_categories_medium = "Medium",
rcsi_categories_high = "High",
hhs_categories_none = "None",
hhs_categories_little = "Little",
hhs_categories_moderate = "Moderate",
hhs_categories_severe = "Severe",
hhs_categories_very_severe = "Very Severe",
hdds_categories_low = "Low",
hdds_categories_medium = "Medium",
hdds_categories_high = "High"
)
```

add_fcs	<i>add_fcs</i>
---------	----------------

Description

add_fcs

Usage

```
add_fcs(
  .dataset,
  cutoffs = c("normal", "alternative"),
  fsl_fcs_cereal = "fsl_fcs_cereal",
  fsl_fcs_legumes = "fsl_fcs_legumes",
  fsl_fcs_veg = "fsl_fcs_veg",
  fsl_fcs_fruit = "fsl_fcs_fruit",
  fsl_fcs_meat = "fsl_fcs_meat",
  fsl_fcs_dairy = "fsl_fcs_dairy",
  fsl_fcs_sugar = "fsl_fcs_sugar",
  fsl_fcs_oil = "fsl_fcs_oil"
)
```

Arguments

.dataset	Main Dataset
cutoffs	either "normal", or "alternative". The default is set to normal
fsl_fcs_cereal	the name of the variable that indicates the number of days cereals were consumed
fsl_fcs_legumes	the name of the variable that indicates the number of days legumes were consumed
fsl_fcs_veg	the name of the variable that indicates the number of days vegetables were consumed

fsl_fcs_fruit	the name of the variable that indicates the number of days fruits were consumed
fsl_fcs_meat	the name of the variable that indicates the number of days meat/fish were consumed
fsl_fcs_dairy	the name of the variable that indicates the number of days dairy were consumed
fsl_fcs_sugar	the name of the variable that indicates the number of days sugar was consumed
fsl_fcs_oil	the name of the variable that indicates the number of days oil were consumed

Value

the dataset with fsl_fcs_score and fsl_fcs_cat computed, as well as the 8 weighted food groups

Examples

```
df1 <- data.frame(
  fsl_fcs_cereal = c(1, 2, 3, 2, 5, 6, 7),
  fsl_fcs_legumes = c(3, 4, 5, 6, 1, 6, 5),
  fsl_fcs_veg = c(3, 2, 1, 6, 5, 4, 3),
  fsl_fcs_fruit = c(1, 4, 6, 2, 2, 2, 4),
  fsl_fcs_meat = c(5, 4, 3, 2, 7, 4, 5),
  fsl_fcs_dairy = c(1, 2, 6, 7, 3, 4, 2),
  fsl_fcs_sugar = c(1, 7, 6, 5, 2, 3, 4),
  fsl_fcs_oil = c(2, 3, 6, 5, 1, 7, 4)
)
add_fcs(.dataset = df1,
  cutoffs = "normal"
)
```

add_fds_cannot_cat	<i>Add Functional Domestic Space Tasks Categories</i>
--------------------	---

Description

This function adds categories for functional domestic space tasks based on various input parameters. It processes cooking, sleeping, storing, and lighting tasks, creating new columns with standardized categories and a summary of tasks that cannot be performed.

Usage

```
add_fds_cannot_cat(
  df,
  fds_cooking = "snfi_fds_cooking",
  fds_cooking_can = "yes",
  fds_cooking_cannot = "no",
  fds_cooking_no_need = "no_need",
  fds_cooking_undefined = "pnta",
  fds_sleeping = "snfi_fds_sleeping",
```

```

    fds_sleeping_can = "yes",
    fds_sleeping_cannot = "no",
    fds_sleeping_undefined = "pnta",
    fds_storing = "snfi_fds_storing",
    fds_storing_cannot = "no",
    fds_storing_can = "yes",
    fds_storing_undefined = "pnta",
    lighting_source = "energy_lighting_source",
    lighting_source_none = "none",
    lighting_source_undefined = c("pnta", "dnk")
)

```

Arguments

<code>df</code>	A data frame containing the input columns
<code>fds_cooking</code>	Column name for cooking tasks
<code>fds_cooking_can</code>	Value for can perform cooking tasks without issues
<code>fds_cooking_cannot</code>	Value for facing issues when cooking
<code>fds_cooking_no_need</code>	Value for no need to perform cooking tasks
<code>fds_cooking_undefined</code>	Vector of undefined responses for cooking tasks
<code>fds_sleeping</code>	Column name for sleeping tasks
<code>fds_sleeping_can</code>	Value for can perform sleeping tasks
<code>fds_sleeping_cannot</code>	Value for cannot perform sleeping tasks
<code>fds_sleeping_undefined</code>	Vector of undefined responses for sleeping tasks
<code>fds_storing</code>	Column name for storing tasks
<code>fds_storing_cannot</code>	Value for cannot perform storing tasks
<code>fds_storing_can</code>	Value for can perform storing tasks
<code>fds_storing_undefined</code>	Vector of undefined responses for storing tasks
<code>lighting_source</code>	Column name for lighting source
<code>lighting_source_none</code>	Value for no lighting source
<code>lighting_source_undefined</code>	Vector of undefined responses for lighting source

Value

A data frame with additional columns:

- snfi_fds_cooking: Standardized categories for cooking tasks
- snfi_fds_sleeping: Standardized categories for sleeping tasks
- snfi_fds_storing: Standardized categories for storing tasks
- energy_lighting_source: Standardized categories for lighting source
- snfi_fds_cannot_n: Number of tasks that cannot be performed
- snfi_fds_cannot_cat: Categorized number of tasks that cannot be performed

add_handwashing_facility_cat

Add Handwashing Facility Category

Description

This function categorizes handwashing facilities into "Basic," "Limited," or "No facility" based on survey data. It evaluates various parameters related to the availability of handwashing facilities and the presence of soap and water.

Usage

```
add_handwashing_facility_cat(
  df,
  survey_modality = "survey_modality",
  survey_modality_in_person = c("in_person"),
  survey_modality_remote = c("remote"),
  facility = "wash_handwashing_facility",
  facility_yes = c("available_fixed_in_dwelling", "available_fixed_in_plot",
    "available_mobile"),
  facility_no = "none",
  facility_no_permission = "no_permission",
  facility_undefined = "other",
  facility_observed_water = "wash_handwashing_facility_observed_water_yn",
  facility_observed_water_yes = "water_available",
  facility_observed_water_no = c("water_not_available"),
  facility_observed_soap = "wash_soap_observed_yn",
  facility_observed_soap_yes = "soap_available",
  facility_observed_soap_no = "soap_not_available",
  facility_observed_soap_undefined = c("other", "pnta"),
  facility_reported = "wash_handwashing_facility_reported",
  facility_reported_yes = c("fixed_dwelling", "fixed_yard", "mobile"),
  facility_reported_no = c("none"),
  facility_reported_undefined = c("other", "dnk", "pnta"),
  facility_reported_water = "wash_handwashing_facility_water_reported_yn",
```

```

facility_reported_water_yes = "yes",
facility_reported_water_no = c("no"),
facility_reported_water_undefined = c("dnk", "pnta"),
facility_reported_soap = "wash_soap_reported_yn",
facility_reported_soap_yes = "yes",
facility_reported_soap_no = c("no"),
facility_reported_soap_undefined = c("dnk", "pnta"),
soap_type_observed = "wash_soap_observed_type",
soap_type_observed_yes = c("soap", "detergent"),
soap_type_observed_no = c("ash_mud_sand"),
soap_type_observed_undefined = c("dnk", "pnta", "other"),
soap_type_reported = "wash_soap_reported_type",
soap_type_reported_yes = c("soap", "detergent"),
soap_type_reported_no = c("ash_mud_sand"),
soap_type_reported_undefined = c("dnk", "pnta", "other")
)

```

Arguments

df	A data frame containing the survey data.
survey_modality	Column name for the survey modality (e.g., "survey_modality"). If the source form does not collect this column, the caller must add it before calling this function, set to "in_person" for a fully in-person data collection.
survey_modality_in_person	Options for the in-person survey modality.
survey_modality_remote	Options for the remote survey modality.
facility	Column name for the observed handwashing facility (e.g., "wash_handwashing_facility").
facility_yes	Response code(s) indicating availability (e.g., "available").
facility_no	Response code indicating unavailability (e.g., "none").
facility_no_permission	Response code for cases with no permission to observe.
facility_undefined	Response code(s) for undefined situations.
facility_observed_water	Column name for observed water availability (e.g., "wash_handwashing_facility_observed_water_yn").
facility_observed_water_yes	Response code indicating water is available.
facility_observed_water_no	Response code(s) indicating water is not available.
facility_observed_soap	Column name for observed soap availability.
facility_observed_soap_yes	Response code indicating soap is available (observed).

facility_observed_soap_no
Response code(s) indicating soap is not available (observed).

facility_observed_soap_undefined
Response code(s) for undefined observed soap situations.

facility_reported
Column name for reported handwashing facility (used in remote/no-permission cases).

facility_reported_yes
Response code(s) indicating facility is available (reported).

facility_reported_no
Response code(s) indicating no facility is available (reported).

facility_reported_undefined
Response code(s) for undefined reported facility situations.

facility_reported_water
Column name for reported water availability under no permission or remote conditions.

facility_reported_water_yes
Response code(s) indicating water is available under no permission or remote conditions.

facility_reported_water_no
Response code(s) indicating water is not available under no permission or remote conditions.

facility_reported_water_undefined
Response code(s) for undefined water situations under no permission or remote conditions.

facility_reported_soap
Column name for reported soap availability under no permission or remote conditions.

facility_reported_soap_yes
Response code(s) indicating soap is available under no permission or remote conditions.

facility_reported_soap_no
Response code(s) indicating soap is not available under no permission or remote conditions.

facility_reported_soap_undefined
Response code(s) for undefined soap situations under no permission or remote conditions.

soap_type_observed
Column name for the type of soap (observed).

soap_type_observed_yes
Response code(s) indicating soap is observed

soap_type_observed_no
Response code(s) indicating soap is NOT observed

soap_type_observed_undefined
Response code(s) for undefined situations.

soap_type_reported
Column name for the type of soap (reported).

soap_type_reported_yes
Response code(s) indicating soap is available.

soap_type_reported_no
Response code(s) indicating soap is NOT available.

soap_type_reported_undefined
Response code(s) for undefined situations.

Details

The function checks for the presence and validity of all required columns and values in the input data frame. It ensures that all parameters are documented and used in the logic. The categorization follows JMP (Joint Monitoring Programme) standards for handwashing facilities.

Value

A data frame with an additional column 'wash_handwashing_facility_jmp_cat': Categorized hand-washing facilities: "Basic," "Limited," or "No Facility."

add_hhs	<i>Add the household hunger scale to the dataset</i>
---------	--

Description

Add the household hunger scale to the dataset

Usage

```
add_hhs(  
  .dataset,  
  fsl_hhs_nofoodhh = "fsl_hhs_nofoodhh",  
  fsl_hhs_nofoodhh_freq = "fsl_hhs_nofoodhh_freq",  
  fsl_hhs_sleephungry = "fsl_hhs_sleephungry",  
  fsl_hhs_sleephungry_freq = "fsl_hhs_sleephungry_freq",  
  fsl_hhs_alldaynight = "fsl_hhs_alldaynight",  
  fsl_hhs_alldaynight_freq = "fsl_hhs_alldaynight_freq",  
  yes_answer = "yes",  
  no_answer = "no",  
  rarely_answer = "rarely",  
  sometimes_answer = "sometimes",  
  often_answer = "often"  
)
```

Arguments

.dataset Main Dataset

fsl_hhs_nofoodhh	The name of the column "In the past 4 weeks (30 days), was there ever no food to eat of any kind in your house because of lack of resources to get food?". It has to be a string.
fsl_hhs_nofoodhh_freq	The name of the column "How often did this happen in the past (4 weeks/30 days)?" . It has to be a string.
fsl_hhs_sleephungry	The name of the column "In the past 4 weeks (30 days), did you or any household member go to sleep at night hungry because there was not enough food?". It has to be a string.
fsl_hhs_sleephungry_freq	The name of the column "How often did this happen in the past (4 weeks/30 days)?" . It has to be a string.
fsl_hhs_alldaynight	The name of the column "In the past 4 weeks (30 days), did you or any household member go a whole day and night without eating anything at all because there was not enough food?". It has to be a string.
fsl_hhs_alldaynight_freq	The name of the column "How often did this happen in the past (4 weeks/30 days)?" . It has to be a string.
yes_answer	Value used for "yes"
no_answer	Value used for the "no"
rarely_answer	Value used for "rarely"
sometimes_answer	Value used for "sometimes"
often_answer	Value used for "often"

Value

It returns the dataframe with 12 extras columns: recoded hhs questions, score for the 3 sets of questions (from 0 to 2), the HHS score (from 0 to 6), the HHS category and the HHS IPC category

Examples

```
{
  input_data <- data.frame(
    fsl_hhs_nofoodhh = c("no", "yes", "no", "no", "no"),
    fsl_hhs_nofoodhh_freq = c(NA_character_, "rarely",
      NA_character_, NA_character_, NA_character_),
    fsl_hhs_sleephungry = c("no", "no", "yes", "no", "no"),
    fsl_hhs_sleephungry_freq = c(NA_character_, NA_character_,
      "often", NA_character_, NA_character_),
    fsl_hhs_alldaynight = c("no", "no", "yes", "yes", "yes"),
    fsl_hhs_alldaynight_freq = c(NA_character_, NA_character_,
      "often", "rarely", "sometimes")
  )
}
```

```

add_hhs(
  .dataset = input_data,
  fsl_hhs_nofoodhh = "fsl_hhs_nofoodhh",
  fsl_hhs_nofoodhh_freq = "fsl_hhs_nofoodhh_freq",
  fsl_hhs_sleephungry = "fsl_hhs_sleephungry",
  fsl_hhs_sleephungry_freq = "fsl_hhs_sleephungry_freq",
  fsl_hhs_alldaynight = "fsl_hhs_alldaynight",
  fsl_hhs_alldaynight_freq = "fsl_hhs_alldaynight_freq",
  yes_answer = "yes",
  no_answer = "no",
  rarely_answer = "rarely",
  sometimes_answer = "sometimes",
  often_answer = "often"
)
}

```

add_hoh_final

Update Head of Household Information

Description

This function updates the head of household (HoH) gender and age information based on whether the respondent is the HoH or not. If the respondent is the HoH, their gender and age are used; otherwise, the existing HoH information is retained.

Usage

```

add_hoh_final(
  df,
  resp_hoh_yn = "resp_hoh_yn",
  yes = "yes",
  no = "no",
  hoh_gender = "hoh_gender",
  hoh_age = "hoh_age",
  resp_gender = "resp_gender",
  resp_age = "resp_age"
)

```

Arguments

df	The input dataframe.
resp_hoh_yn	The column name for whether the respondent is the head of household (HoH) yes/no indicator.
yes	The value for 'yes' in the 'resp_hoh_yn' column.
no	The value for 'no' in the 'resp_hoh_yn' column.
hoh_gender	The column name for the household (HoH) gender.
hoh_age	The column name for the household (HoH) age.

resp_gender	The column name for the respondent (resp) gender.
resp_age	The column name for the respondent (resp) age.

Value

A modified dataframe with updated columns:

- hoh_gender: Updated head of household gender.
- hoh_age: Updated head of household age.

add_hwise	<i>Add Household Water Insecurity Experiences (HWISE) Score</i>
-----------	---

Description

Computes the total hwise4_score (0–12) from the four HWISE survey items (water quantity), and maps it to a WASH component score for water quantity (comp_wash_score_water_quantity, 1–5). The four items are internally recoded to numeric scores but are only exposed in the output (as _score-suffixed columns) when .keep_recoded = TRUE.

Usage

```
add_hwise(
  .df,
  wash_hwise_drink = "wash_hwise_drink",
  wash_hwise_hands = "wash_hwise_hands",
  wash_hwise_plans = "wash_hwise_plans",
  wash_hwise_worry = "wash_hwise_worry",
  hwise_never = "never",
  hwise_rarely = "rarely",
  hwise_sometimes = "sometimes",
  hwise_often = "often",
  hwise_always = "always",
  hwise_dnk = "dnk",
  hwise_pnta = "pnta",
  .keep_recoded = FALSE
)
```

Arguments

.df	A data frame containing the four HWISE columns.
wash_hwise_drink	Column name for: <i>"In the last 4 weeks, how frequently has there NOT been as much water to drink as you would like for you or anyone in your household?"</i>

wash_hwise_hands	Column name for: <i>"In the last 4 weeks, how frequently have you or anyone in your household had to go without washing hands after dirty activities (e.g., defecating or changing diapers, cleaning animal dung) because of problems with water?"</i>
wash_hwise_plans	Column name for: <i>"In the last 4 weeks how often did you have to change schedules or plans because of problems with water?"</i>
wash_hwise_worry	Column name for: <i>"In the last 4 weeks how often did you or anyone in your household worry that you would not have enough water for all of your needs?"</i>
hwise_never	Response code for "Never (0 days)" (scored 0).
hwise_rarely	Response code for "Rarely (1–2 days)" (scored 1).
hwise_sometimes	Response code for "Some days (3–10 days)" (scored 2).
hwise_often	Response code for "Often (11–20 days)" (scored 3).
hwise_always	Response code for "Always (more than 20 days)" (scored 3).
hwise_dnk	Response code for "Don't know" (scored NA).
hwise_pnta	Response code for "Prefer not to answer" (scored NA).
.keep_recoded	Logical, whether to keep the recoded numeric HWISE columns (ending in _score) in the output data frame. Default is FALSE.

Value

.df with new columns:

- The four HWISE columns recoded to numeric scores (0–3, NA for dnk/pnta), suffixed with _score (only kept in the output if .keep_recoded = TRUE).
- hwise4_score: row sum of the four recoded columns (0–12).
- comp_wash_score_water_quantity: WASH severity score (1–5): 1 = score 0–3, 2 = 4–6, 3 = 7–8, 4 = 9–10, 5 = 11+.

See Also

[add_comp_wash\(\)](#)

Examples

```
df <- dplyr::tibble(
  wash_hwise_drink = c("never",      "sometimes", "often"),
  wash_hwise_hands = c("rarely",    "often",      "always"),
  wash_hwise_plans = c("sometimes", "often",      "always"),
  wash_hwise_worry = c("never",     "sometimes",  "always")
)

result <- add_hwise(df)
dplyr::select(result, hwise4_score, comp_wash_score_water_quantity)
```

```
# keep the per-item recoded scores (ending in `_score`)  
result_recoded <- add_hwise(df, .keep_recoded = TRUE)  
dplyr::select(result_recoded, dplyr::matches("_score$"))
```

add_income_source_prop

Add Income Source Amounts as Proportions of Total Income

Description

This function calculates the proportion of each income source relative to the total income. It also computes the total income from all sources.

Prerequisite function:

- add_income_source_zero_to_sl.R

Usage

```
add_income_source_prop(  
  df,  
  income_souce_salaried_n = "cm_income_source_salaried_n",  
  income_source_casual_n = "cm_income_source_casual_n",  
  income_source_own_business_n = "cm_income_source_own_business_n",  
  income_source_own_production_n = "cm_income_source_own_production_n",  
  income_source_social_benefits_n = "cm_income_source_social_benefits_n",  
  income_source_rent_n = "cm_income_source_rent_n",  
  income_source_remittances_n = "cm_income_source_remittances_n",  
  income_source_assistance_n = "cm_income_source_assistance_n",  
  income_source_support_friends_n = NULL,  
  income_source_donation_n = NULL,  
  income_source_other_n = "cm_income_source_other_n"  
)
```

Arguments

df	A data frame containing income source columns
income_souce_salaried_n	Column name for salaried income amount
income_source_casual_n	Column name for casual income amount
income_source_own_business_n	Column name for own business income amount
income_source_own_production_n	Column name for own production income amount
income_source_social_benefits_n	Column name for social benefits income amount

income_source_rent_n
 Column name for rent income amount
 income_source_remittances_n
 Column name for remittances income amount
 income_source_assistance_n
 Column name for assistance income amount
 income_source_support_friends_n
 Column name for support from friends income amount
 income_source_donation_n
 Column name for donation income amount
 income_source_other_n
 Column name for other income amount

Value

A data frame with additional columns:

- cm_income_total: Total income from all sources.
- *_prop: Proportion of each income source relative to total income (e.g., cm_income_source_salaried_n_prop).

Details on loans

Loans (income_source_support_friends_n and income_source_donation_n) are considered to be a cash influx. Yet they do not count as a formal income source. While it is good practice to collect these figures as part of this module, they should not be included in the total income calculation. This is why their default value here is NULL.

add_income_source_rank

Add Income Source Categories, Count, and Ranking Top 3

Description

This function categorizes income sources, counts them by type, and ranks the top 3 income sources. It also adds categories for emergency, unstable, stable, and other income sources.

Prerequisite function:

- add_income_source_zero_to_sl.R

Usage

```
add_income_source_rank(
  df,
  emergency = c("cm_income_source_assistance_n", "cm_income_source_support_friends_n",
    "cm_income_source_donation_n"),
  unstable = c("cm_income_source_casual_n", "cm_income_source_social_benefits_n",
    "cm_income_source_rent_n", "cm_income_source_remittances_n"),
```

```

    stable = c("cm_income_source_salaried_n", "cm_income_source_own_business_n",
               "cm_income_source_own_production_n"),
    other = "cm_income_source_other_n",
    id_col = "uuid"
  )

```

Arguments

df	A data frame containing income source information.
emergency	A vector of column names containing emergency income sources.
unstable	A vector of column names containing unstable income sources.
stable	A vector of column names containing stable income sources.
other	The name of the column containing other income sources.
id_col	The name of the column containing the unique identifier.

Value

A data frame with additional columns:

- cm_income_source_emergency_n: Count of emergency income sources.
- cm_income_source_unstable_n: Count of unstable income sources.
- cm_income_source_stable_n: Count of stable income sources.
- cm_income_source_other_n: Count of other income sources.
- cm_income_source_top1: Top income source.
- cm_income_source_top2: Second top income source.
- cm_income_source_top3: Third top income source.

add_income_source_zero_to_sl

Add Zero to Income Sources When Skipped

Description

This function adds zero to income source columns when the main income source was skipped or undefined. It also ensures that all income sources are zero when the main income source is "none".

Usage

```

add_income_source_zero_to_sl(
  df,
  income_source = "cm_income_source",
  none = "none",
  undefined = c("dnk", "pnta"),
  income_sources = c("cm_income_source_salaried_n", "cm_income_source_casual_n",

```

```
    "cm_income_source_own_business_n", "cm_income_source_own_production_n",  
    "cm_income_source_social_benefits_n", "cm_income_source_rent_n",  
    "cm_income_source_remittances_n", "cm_income_source_assistance_n",  
    "cm_income_source_support_friends_n", "cm_income_source_donation_n",  
    "cm_income_source_other_n")  
  )
```

Arguments

df	A data frame containing income source information.
income_source	A character string. The name of the column that contains the main income source.
none	A character string. The value that indicates that no income source was selected.
undefined	A character vector. The values that indicate that the income source was undefined or skipped.
income_sources	A character vector. The names of the columns that contain the amount of income from various sources.

Value

- A data frame with updated income source columns:
- income_sources: All specified income source columns are updated to zero when the main income source is skipped, undefined, or none.

add_lcsi	<i>Add LCSi</i>
----------	-----------------

Description

Function to calculate Livelihood Coping Strategy Index (LCSI)

Usage

```
add_lcsi(  
  .dataset,  
  fsl_lcsi_stress1 = "fsl_lcsi_stress1",  
  fsl_lcsi_stress2 = "fsl_lcsi_stress2",  
  fsl_lcsi_stress3 = "fsl_lcsi_stress3",  
  fsl_lcsi_stress4 = "fsl_lcsi_stress4",  
  fsl_lcsi_crisis1 = "fsl_lcsi_crisis1",  
  fsl_lcsi_crisis2 = "fsl_lcsi_crisis2",  
  fsl_lcsi_crisis3 = "fsl_lcsi_crisis3",  
  fsl_lcsi_emergency1 = "fsl_lcsi_emergency1",  
  fsl_lcsi_emergency2 = "fsl_lcsi_emergency2",  
  fsl_lcsi_emergency3 = "fsl_lcsi_emergency3",  
  yes_val = "yes",
```

```

    no_val = "no_had_no_need",
    exhausted_val = "no_exhausted",
    not_applicable_val = "not_applicable"
)

```

Arguments

<code>.dataset</code>	Main Dataset
<code>fsl_lcsi_stress1</code>	the name of the variable that indicates the first stress LCSI strategy
<code>fsl_lcsi_stress2</code>	the name of the variable that indicates the second stress LCSI strategy
<code>fsl_lcsi_stress3</code>	the name of the variable that indicates the third stress LCSI strategy
<code>fsl_lcsi_stress4</code>	the name of the variable that indicates the fourth stress LCSI strategy
<code>fsl_lcsi_crisis1</code>	the name of the variable that indicates the first crisis LCSI strategy
<code>fsl_lcsi_crisis2</code>	the name of the variable that indicates the second crisis LCSI strategy
<code>fsl_lcsi_crisis3</code>	the name of the variable that indicates the third crisis LCSI strategy
<code>fsl_lcsi_emergency1</code>	the name of the variable that indicates the first emergency LCSI strategy
<code>fsl_lcsi_emergency2</code>	the name of the variable that indicates the second emergency LCSI strategy
<code>fsl_lcsi_emergency3</code>	the name of the variable that indicates the third emergency LCSI strategy
<code>yes_val</code>	A character value in the dataset associated with "Yes, used this coping strategy in the last 30 days."
<code>no_val</code>	A character value in the dataset associated with "No, have not used this coping strategy in the last 30 days."
<code>exhausted_val</code>	A character value in the dataset associated with "No, haven't used in the last 30 days because I've exhausted this coping strategy in the last 6 or 12 months."
<code>not_applicable_val</code>	A character value in the dataset associated with "This coping strategy is not applicable for the household."

Value

Returns a dataframe with added columns for LCSI indicators.

- `fsl_lcsi_x_yes` : 1 means one of the of the x strategies was used (*yes_val*)
- `fsl_lcsi_x_exhaust`: 1 means one of the x strategies was exhausted and could not be used (*exhausted_val*)

- fsl_lcsi_x: 1 means one of the x strategies was if either used (*yes_val*) or exhausted (*exhausted_val*) Where x is stress, crisis or emergency
- fsl_lcsi_cat_yes : the highest category between the fsl_lcsi_x_yes
- fsl_lcsi_cat_exhaust: the highest category between the fsl_lcsi_x_exhaust
- fsl_lcsi_cat: the highest category between the fsl_lcsi_x

Examples

```
{
  input_data1 <- data.frame(fsl_lcsi_stress1 = c("No", "No", "Exhausted", "Not Applicable", "No"),
    fsl_lcsi_stress2 = c("No", "Yes", "Not Applicable", "No", "No"),
    fsl_lcsi_stress3 = c("Not Applicable", "Not Applicable", "Yes", "No", "No"),
    fsl_lcsi_stress4 = c("Not Applicable", "No", "Yes", "Yes", "No"),
    fsl_lcsi_crisis1 = c("No", "Not Applicable", "Yes", "Exhausted", "No"),
    fsl_lcsi_crisis2 = c("No", "No", "No", "No", "No"),
    fsl_lcsi_crisis3 = c("No", "No", "Yes", "Not Applicable", "No"),
    fsl_lcsi_emergency1 = c("No", "Not Applicable", "Not Applicable", "No", "No"),
    fsl_lcsi_emergency2 = c("No", "Not Applicable", "Yes", "Not Applicable", "No"),
    fsl_lcsi_emergency3 = c("Not Applicable", "No", "Not Applicable", "No", "Exhausted"))

  add_lcsi(.dataset = input_data1,
    yes_val = "Yes",
    no_val = "No",
    exhausted_val = "Exhausted",
    not_applicable_val = "Not Applicable")
}
```

add_loop_age_dummy	<i>Add Age and Gender Dummy Variables for Demographics Loop</i>
--------------------	---

Description

This set of functions adds dummy variables for age classes and gender in the demographics loop, and optionally sums these dummies to the main dataset. It includes four main functions: [add_loop_age_dummy\(\)](#), [add_loop_age_dummy_to_main\(\)](#), [add_loop_age_gender_dummy\(\)](#), and [add_loop_age_gender_dummy_to_main\(\)](#).

Usage

```
add_loop_age_dummy(
  loop,
  ind_age = "ind_age",
  lb = 5,
  ub = 18,
  new_colname = NULL
)

add_loop_age_dummy_to_main(
```

```

    main,
    loop,
    ind_age_dummy = "ind_age_5_18",
    id_col_main = "uuid",
    id_col_loop = "uuid",
    new_colname = NULL
)

add_loop_age_gender_dummy(
  loop,
  ind_age = "ind_age",
  lb = 5,
  ub = 18,
  ind_gender = "ind_gender",
  gender = "female",
  new_colname = NULL
)

add_loop_age_gender_dummy_to_main(
  main,
  loop,
  ind_age_gender_dummy = "ind_age_female_5_18",
  id_col_main = "uuid",
  id_col_loop = "uuid",
  new_colname = NULL
)

```

Arguments

loop	A data frame of individual-level data.
ind_age	Column name for individual age.
lb	Lower bound for the age class.
ub	Upper bound for the age class.
new_colname	New column name for the dummy variable. If NULL, then the default is "ind_age_lb_ub" or "ind_dummy_n" or "ind_age_gender_lb_ub" or "ind_age_gender_dummy_n" depending on the function used.
main	A data frame of household-level data.
ind_age_dummy	Column name for the dummy variable of the age class.
id_col_main	Column name for the unique identifier in the main dataset.
id_col_loop	Column name for the unique identifier in the loop dataset.
ind_gender	Column name for individual gender.
gender	Response options of interest, e.g. "Female".
ind_age_gender_dummy	Column name for the dummy variable of the age-gender class.

Value

Depending on the function used:

- `add_loop_age_dummy`: Returns the loop data frame with an additional column for the age dummy variable.
- `add_loop_age_dummy_to_main`: Returns the main data frame with an additional column for the sum of age dummy variables.
- `add_loop_age_gender_dummy`: Returns the loop data frame with an additional column for the age-gender dummy variable.
- `add_loop_age_gender_dummy_to_main`: Returns the main data frame with an additional column for the sum of age-gender dummy variables.

`add_loop_edu_access_d` *Add Education Access Dummy Variables*

Description

This function adds dummy variables to an individual-level dataframe for all school-aged children. It creates a variable (`edu_ind_access_d`) that indicates whether the child accessed school (1 if accessed, 0 if explicitly not accessed) and another variable (`edu_ind_no_access_d`) that indicates no access to school.

Prerequisite function:

- `add_loop_edu_ind_age_corrected.R`

Usage

```
add_loop_edu_access_d(
  loop,
  ind_access = "edu_access",
  yes = "yes",
  no = "no",
  pnta = "pnta",
  dnk = "dnk",
  ind_schooling_age_d = "edu_ind_age_schooling"
)

add_loop_edu_access_d_to_main(
  main,
  loop,
  ind_access_d = "edu_ind_access_d",
  ind_no_access_d = "edu_ind_no_access_d",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)
```

Arguments

loop	A data frame of individual-level data for the loop.
ind_access	Column name for education access.
yes	Value indicating access to education (e.g., "yes").
no	Value indicating no access to education (e.g., "no").
pnta	Value indicating prefer not to answer (e.g., "pnta").
dnk	Value indicating don't know (e.g., "dnk").
ind_schooling_age_d	Column name for the dummy variable indicating schooling age.
main	A data frame of household-level data.
ind_access_d	Column name for education access (binary).
ind_no_access_d	Column name for education no access (binary).
id_col_main	Column name for the unique identifier in the main dataset.
id_col_loop	Column name for the unique identifier in the loop dataset.

Details

pnta (prefer not to answer) and dnk (don't know) responses are coded as NA, not as "not accessed"

Value

A data frame with additional columns:

- edu_ind_access_d: Dummy variable indicating access to education (1 if accessed, 0 if explicitly not accessed, NA if not school age or pnta/dnk).
- edu_ind_no_access_d: Dummy variable indicating no access to education (1 if explicitly not accessed, 0 if accessed, NA if not school age or pnta/dnk).

add_loop_edu_barrier_protection_d

Add Child Protection Barriers to Education Variable

Description

This function adds a dummy variable for child protection barriers to education in individual-level data and aggregates it to household-level data.

Prerequisite function:

- add_loop_edu_ind_age_corrected.R

Usage

```

add_loop_edu_barrier_protection_d(
  loop,
  barriers = "edu_barrier",
  protection_issues = c("protection_at_school", "protection_travel_school",
    "child_work_home", "child_work_outside", "child_armed_group", "child_marriage",
    "child_pregnancy", "ban", "enroll_lack_documentation", "discrimination"),
  ind_schooling_age_d = "edu_ind_age_schooling",
  barriers_undefined = c("dnk", "pnta", "other"),
  non_protection_issues = c("costs", "lack_interest", "not_priority",
    "lack_accessible_school", "lack_classrooms", "lack_wash_facilities", "school_closed",
    "lack_teacher", "curriculum_not_useful", "child_health", "language",
    "enroll_displacement", "child_too_young", "child_graduated")
)

add_loop_edu_barrier_protection_d_to_main(
  main,
  loop,
  ind_barrier_protection_d = "edu_ind_barrier_protection_d",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)

```

Arguments

loop	A data frame of individual-level data for the loop.
barriers	Column name for the child protection barrier category.
protection_issues	Vector of protection issues RESPONSE CODES. Values in this set are flagged as protection barriers. Must not overlap barriers_undefined or non_protection_issues.
ind_schooling_age_d	Column name for the dummy variable of schooling age.
barriers_undefined	Vector of undefined/non-response values. These are accepted but treated as non-barriers. Defaults to common survey non-responses: "dnk", "pnta", "other". Must not overlap protection_issues or non_protection_issues.
non_protection_issues	Vector of valid non-protection barrier RESPONSE CODES (access barriers and flag codes). Values in this set are accepted as valid, but do not affect the edu_ind_barrier_protection_d dummy. Defaults to the non-protection response codes of the standard edu_barrier question. Must not overlap protection_issues or barriers_undefined.
main	A data frame of household-level data.
ind_barrier_protection_d	Column name for the dummy variable of the child protection barrier category.
id_col_main	Column name for the unique identifier in the main dataset.
id_col_loop	Column name for the unique identifier in the loop dataset.

Value

A data frame with an additional column:

- `edu_ind_barrier_protection_d`: Dummy variable indicating if a school-aged child faces a protection barrier (1) or not (0).

A data frame with an additional column:

`edu_barrier_protection_n`

Count of school-aged children facing protection barriers in each household.

`add_loop_edu_disrupted_d`

Add Education Disruption Categories

Description

This function adds education disruption binaries to individual-level data and summarizes them at the household level. It includes two main functions: `add_loop_edu_disrupted_d()` for individual-level data and `add_loop_edu_disrupted_d_to_main()` for household-level data.

Prerequisite function:

- `add_loop_edu_ind_age_corrected.R`

Usage

```
add_loop_edu_disrupted_d(
  df,
  attack = "edu_disrupted_attack",
  hazards = "edu_disrupted_hazards",
  displaced = "edu_disrupted_displaced",
  teacher = "edu_disrupted_teacher",
  levels = c("yes", "no", "dnk", "pnta"),
  ind_schooling_age_d = "edu_ind_age_schooling"
)

add_loop_edu_disrupted_d_to_main(
  main,
  loop,
  attack_d = "edu_disrupted_attack_d",
  hazards_d = "edu_disrupted_hazards_d",
  displaced_d = "edu_disrupted_displaced_d",
  teacher_d = "edu_disrupted_teacher_d",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)
```

Arguments

df	A data frame containing individual-level education data.
attack	Column name for attack disruption. NULL if dimension is not present in the data frame.
hazards	Column name for hazards disruption.
displaced	Column name for displaced disruption.
teacher	Column name for teacher disruption.
levels	Vector of levels for the disruption variables.
ind_schooling_age_d	Column name for the dummy variable of the schooling age class.
main	A data frame of household-level data.
loop	A data frame of individual-level data.
attack_d	Column name for the dummy variable of the attack dimension. NULL if dimension is not present in the data frame.
hazards_d	Column name for the dummy variable of the hazards dimension.
displaced_d	Column name for the dummy variable of the displaced dimension.
teacher_d	Column name for the dummy variable of the teacher dimension.
id_col_main	Column name for the unique identifier in the main dataset.
id_col_loop	Column name for the unique identifier in the loop dataset.

Value

A data frame with additional columns:

- *_d: Binary columns for each disruption type (e.g., edu_disrupted_attack_d).

A data frame with additional columns:

- edu_disrupted_*_n: Count of individuals experiencing each type of education disruption (e.g., edu_disrupted_hazards_n).

add_loop_edu_ind_age_corrected

Add a correct schooling age to the loop

Description

Add a correct schooling age to the loop

Usage

```

add_loop_edu_ind_age_corrected(
  loop,
  main,
  id_col_loop = "uuid",
  id_col_main = "uuid",
  survey_start_date = "start",
  school_year_start_month = 9,
  ind_age = "ind_age",
  month = NULL,
  schooling_start_age = 5,
  schooling_end_age = 17
)

add_loop_edu_ind_schooling_age_d_to_main(
  main,
  loop,
  ind_schooling_age_d = "edu_ind_age_schooling",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)

```

Arguments

loop	A data frame of individual-level data.
main	A data frame of household-level data.
id_col_loop	Column name for the unique identifier in the loop dataset.
id_col_main	Column name for the unique identifier in the main dataset.
survey_start_date	Survey start date column name in main.
school_year_start_month	The month when the school year has started.
ind_age	The individual age column.
month	If not NULL, an integer between 1 and 12 which will be used as the month of data collection for all households.
schooling_start_age	The age at which we assign the value 1 to edu_ind_age_schooling. Default is 5.
schooling_end_age	The maximum age considered within the schooling-age population. Default is 17.
ind_schooling_age_d	Column name for the dummy variable of the schooling age class.

Value

2 new columns: "edu_ind_age_corrected" with the corrected individual age, and a dummy variable edu_ind_age_schooling

add_loop_healthcare_needed_cat

Add Healthcare Needed Category to Individual Data

Description

This function adds healthcare need categories to individual-level data, including disability information if provided. It creates dummy variables for different healthcare need scenarios and optionally includes disability-specific indicators.

This function aggregates individual-level healthcare need data to the household level, including disability-specific information if provided.

Usage

```
add_loop_healthcare_needed_cat(
  loop,
  ind_healthcare_needed = "health_ind_healthcare_needed",
  ind_healthcare_needed_no = "no",
  ind_healthcare_needed_yes = "yes",
  ind_healthcare_needed_dnk = "dnk",
  ind_healthcare_needed_pnta = "pnta",
  ind_healthcare_received = "health_ind_healthcare_received",
  ind_healthcare_received_no = "no",
  ind_healthcare_received_yes = "yes",
  ind_healthcare_received_dnk = "dnk",
  ind_healthcare_received_pnta = "pnta",
  ind_age = "ind_age"
)

add_loop_healthcare_needed_cat_to_main(
  main,
  loop,
  ind_healthcare_needed_no = "health_ind_healthcare_needed_no",
  ind_healthcare_needed_yes_unmet = "health_ind_healthcare_needed_yes_unmet",
  ind_healthcare_needed_yes_met = "health_ind_healthcare_needed_yes_met",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)
```

Arguments

loop	A data frame of individual-level data.
ind_healthcare_needed	The name of the variable that indicates if healthcare is needed.
ind_healthcare_needed_no	The binary variable that indicates if healthcare is not needed.

ind_healthcare_needed_yes	Level for "yes" in ind_healthcare_needed.
ind_healthcare_needed_dnk	Level for "don't know" in ind_healthcare_needed.
ind_healthcare_needed_pnta	Level for "prefer not to answer" in ind_healthcare_needed.
ind_healthcare_received	The name of the variable that indicates if healthcare is received.
ind_healthcare_received_no	Level for "no" in ind_healthcare_received.
ind_healthcare_received_yes	Level for "yes" in ind_healthcare_received.
ind_healthcare_received_dnk	Level for "don't know" in ind_healthcare_received.
ind_healthcare_received_pnta	Level for "prefer not to answer" in ind_healthcare_received.
ind_age	The name of the variable that indicates the age of the individual.
main	A data frame of household-level data.
ind_healthcare_needed_yes_unmet	The binary variable that indicates if healthcare is needed but unmet.
ind_healthcare_needed_yes_met	The binary variable that indicates if healthcare is needed and met.
id_col_main	The column name for the unique identifier in the main data frame.
id_col_loop	The column name for the unique identifier in the loop data frame.

Value

A data frame with additional columns:

- health_ind_healthcare_needed_d: Dummy variable for healthcare needed.
- health_ind_healthcare_received_d: Dummy variable for healthcare received.
- health_ind_healthcare_needed_cat: Categorized healthcare need: "no_need", "yes_unmet_need", or "yes_met_need".
- health_ind_healthcare_needed_no: Dummy variable for no healthcare need.
- health_ind_healthcare_needed_yes_unmet: Dummy variable for unmet healthcare need.
- health_ind_healthcare_needed_yes_met: Dummy variable for met healthcare need.

A data frame with additional columns:

- health_ind_healthcare_needed_no_n: Count of individuals not needing healthcare.
- health_ind_healthcare_needed_yes_unmet_n: Count of individuals with unmet healthcare needs.
- health_ind_healthcare_needed_yes_met_n: Count of individuals with met healthcare needs.

add_loop_wgq_ss

Prepare Washington Group Short Set (WG-SS) Disability Indicators

Description

This function prepares dummy variables for each WG-SS component (vision, hearing, mobility, cognition, self-care, communication) and their levels (no difficulty, some difficulty, a lot of difficulty, cannot do at all). It combines them into sum scores for each level and creates disability binary cut-offs variables.

Usage

```
add_loop_wgq_ss(
  loop,
  ind_age = "ind_age",
  vision = "wgq_vision",
  hearing = "wgq_hearing",
  mobility = "wgq_mobility",
  cognition = "wgq_cognition",
  self_care = "wgq_self_care",
  communication = "wgq_communication",
  no_difficulty = "no_difficulty",
  some_difficulty = "some_difficulty",
  lot_of_difficulty = "a_lot_of_difficulty",
  cannot_do = "cannot_do_at_all",
  undefined = c("dnk", "pnta")
)

add_loop_wgq_ss_to_main(
  main,
  loop,
  wgq_dis_4 = "wgq_dis_4",
  wgq_dis_3 = "wgq_dis_3",
  wgq_dis_2 = "wgq_dis_2",
  wgq_dis_1 = "wgq_dis_1",
  ind_age_above_5 = "ind_age_above_5",
  id_col_main = "uuid",
  id_col_loop = "uuid"
)
```

Arguments

loop	A data frame of individual-level data.
ind_age	The individual age column.
vision	Vision component column.
hearing	Hearing component column.

mobility	Mobility component column.
cognition	Cognition component column.
self_care	Self-care component column.
communication	Communication component column.
no_difficulty	Level for no difficulty.
some_difficulty	Level for some difficulty.
lot_of_difficulty	Level for a lot of difficulty.
cannot_do	Level for cannot do at all.
undefined	Vector of undefined responses, such as Prefer not to answer and Don't know.
main	A data frame of household-level data.
wgq_dis_4	Column name for the disability 4 cut-offs binary.
wgq_dis_3	Column name for the disability 3 cut-offs binary.
wgq_dis_2	Column name for the disability 2 cut-offs binary.
wgq_dis_1	Column name for the disability 1 cut-offs binary.
ind_age_above_5	Column name for the age above 5 dummy in the individual-level dataset.
id_col_main	Column name for the unique identifier in the main dataset.
id_col_loop	Column name for the unique identifier in the loop dataset.

Value

A data frame with additional columns:

- *_cannot_do_d: Binary indicators for "cannot do" level for each component.
- *_lot_of_difficulty_d: Binary indicators for "a lot of difficulty" level for each component.
- *_some_difficulty_d: Binary indicators for "some difficulty" level for each component.
- *_no_difficulty_d: Binary indicators for "no difficulty" level for each component.
- wgq_cannot_do_n: Sum of "cannot do" indicators across all components.
- wgq_lot_of_difficulty_n: Sum of "a lot of difficulty" indicators across all components.
- wgq_some_difficulty_n: Sum of "some difficulty" indicators across all components.
- wgq_no_difficulty_n: Sum of "no difficulty" indicators across all components.
- wgq_cannot_do_d: Binary indicator for any "cannot do" across all components.
- wgq_lot_of_difficulty_d: Binary indicator for any "a lot of difficulty" across all components.
- wgq_some_difficulty_d: Binary indicator for any "some difficulty" across all components.
- wgq_no_difficulty_d: Binary indicator for any "no difficulty" across all components.
- wgq_dis_4: Disability cut-off 4: any domain coded as "cannot do at all".
- wgq_dis_3: Disability cut-off 3: any domain coded as "a lot of difficulty" or "cannot do at all".
- wgq_dis_2: Disability cut-off 2: at least 2 domains coded as "some difficulty" or any domain coded as "a lot of difficulty" or "cannot do at all".
- wgq_dis_1: Disability cut-off 1: at least one domain coded as "some difficulty", "a lot of difficulty", or "cannot do at all".

add_msni

*Add Multi-Sectoral Needs Index (MSNI) Score and Related Indicators***Description**

This function calculates the MSNI score, determines if households are in need or severe need, counts the number of sectoral needs, and creates a sectoral needs profile.

Prerequisite functions:

- add_comp_edu.R
- add_comp_foodsec.R
- add_comp_health.R
- add_comp_prot.R
- add_comp_snfi.R
- add_comp_wash.R

Usage

```
add_msni(
  df,
  comp_foodsec_score = "comp_foodsec_score",
  comp_snfi_score = "comp_snfi_score",
  comp_wash_score = "comp_wash_score",
  comp_prot_score = "comp_prot_score",
  comp_health_score = "comp_health_score",
  comp_edu_score = "comp_edu_score",
  comp_foodsec_in_need = "comp_foodsec_in_need",
  comp_snfi_in_need = "comp_snfi_in_need",
  comp_wash_in_need = "comp_wash_in_need",
  comp_prot_in_need = "comp_prot_in_need",
  comp_health_in_need = "comp_health_in_need",
  comp_edu_in_need = "comp_edu_in_need",
  comp_foodsec_in_severe_need = "comp_foodsec_in_severe_need",
  comp_snfi_in_severe_need = "comp_snfi_in_severe_need",
  comp_wash_in_severe_need = "comp_wash_in_severe_need",
  comp_prot_in_severe_need = "comp_prot_in_severe_need",
  comp_edu_in_severe_need = "comp_edu_in_severe_need"
)
```

Arguments

df A data frame containing sectoral composite scores and in-need indicators.

comp_foodsec_score Column name for the food security composite score.

comp_snfi_score	Column name for the SNFI composite score.
comp_wash_score	Column name for the WASH composite score.
comp_prot_score	Column name for the protection composite score.
comp_health_score	Column name for the health composite score.
comp_edu_score	Column name for the education composite score.
comp_foodsec_in_need	Column name for food security in need.
comp_snfi_in_need	Column name for SNFI in need.
comp_wash_in_need	Column name for WASH in need.
comp_prot_in_need	Column name for protection in need.
comp_health_in_need	Column name for health in need.
comp_edu_in_need	Column name for education in need.
comp_foodsec_in_severe_need	Column name for food security in severe need.
comp_snfi_in_severe_need	Column name for SNFI in severe need.
comp_wash_in_severe_need	Column name for WASH in severe need.
comp_prot_in_severe_need	Column name for protection in severe need.
comp_edu_in_severe_need	Column name for education in severe need.

Value

A data frame with 7 new columns:

- `msni_score`: The Multi-Sectoral Needs Index score.
- `msni_in_need`: Binary indicator for households in need.
- `msni_in_severe_need`: Binary indicator for households in severe need.
- `sector_in_need_n`: Number of sectoral needs identified.
- `sector_in_severe_need_n`: Number of sectoral severe needs identified.
- `sector_needs_profile`: Profile of sectoral needs identified (NA if no sectoral need is identified).
- `sector_severe_needs_profile`: Profile of sectoral severe needs identified (NA if no sectoral severe need is identified).

add_occupancy_cat	<i>Add Category of Occupancy Arrangement and Tenure Security</i>
-------------------	--

Description

This function categorizes occupancy arrangements and eviction risk into high, medium, or low risk, and then creates a new column for overall tenure security, taking the maximum risk from both.

Usage

```
add_occupancy_cat(
  df,
  occupancy = "hlp_occupancy",
  occupancy_high_risk = c("no_agreement"),
  occupancy_medium_risk = c("rented", "hosted_free"),
  occupancy_low_risk = c("ownership"),
  occupancy_undefined = c("dnk", "pnta", "other"),
  eviction = "hlp_risk_eviction",
  eviction_high_risk = "yes",
  eviction_low_risk = "no",
  eviction_undefined = c("dnk", "pnta")
)
```

Arguments

df	A data frame containing occupancy arrangement data.
occupancy	Component column: Occupancy arrangement.
occupancy_high_risk	Character vector of high risk occupancy arrangements.
occupancy_medium_risk	Character vector of medium risk occupancy arrangements.
occupancy_low_risk	Character vector of low risk occupancy arrangements.
occupancy_undefined	Character vector of undefined response codes (e.g. "Prefer not to answer").
eviction	Component column: Eviction risk.
eviction_high_risk	Character vector of high risk eviction responses.
eviction_low_risk	Character vector of low risk eviction responses.
eviction_undefined	Character vector of undefined eviction responses.

Value

A data frame with additional columns:

- hlp_occupancy_cat: Categorized occupancy arrangement: "high_risk", "medium_risk", "low_risk", or "undefined".
- hlp_eviction_cat: Categorized eviction risk: "high_risk", "low_risk", or "undefined".
- hlp_tenure_security: Maximum risk between hlp_occupancy_cat and hlp_eviction_cat.

```
add_prot_score_movement
      add_prot_score_movement
```

Description

add_prot_score_movement

Usage

```
add_prot_score_movement(
  df,
  sep = "/",
  prot_needs_3_movement = "prot_needs_3_movement",
  no_changes_feel_unsafe = "no_changes_feel_unsafe",
  no_safety_concerns = "no_safety_concerns",
  women_girls_avoid_places = "women_girls_avoid_places",
  men_avoid_places = "men_avoid_places",
  boys_avoid_places = "boys_avoid_places",
  women_girls_avoid_night = "women_girls_avoid_night",
  men_avoid_night = "men_avoid_night",
  boys_avoid_night = "boys_avoid_night",
  girls_boys_avoid_school = "girls_boys_avoid_school",
  different_routes = "different_routes",
  avoid_markets = "avoid_markets",
  avoid_public_offices = "avoid_public_offices",
  avoid_fields = "avoid_fields",
  women_girls_boys_avoid_firewood = "women_girls_boys_avoid_firewood",
  women_girls_boys_avoid_places = "women_girls_boys_avoid_places",
  other_safety_measures = "other_safety_measures",
  dnk = "dnk",
  pnta = "pnta",
  men_avoid_places_weight = 1,
  men_avoid_night_weight = 1,
  .keep_weighted = FALSE
)
```

Arguments

df Data frame containing the survey data.
 sep Separator for the binary columns, default is "/".
 prot_needs_3_movement Column name
 no_changes_feel_unsafe answer option
 no_safety_concerns answer option
 women_girls_avoid_places answer option
 men_avoid_places answer option
 boys_avoid_places answer option
 women_girls_avoid_night answer option
 men_avoid_night answer option
 boys_avoid_night answer option
 girls_boys_avoid_school answer option
 different_routes answer option
 avoid_markets answer option
 avoid_public_offices answer option
 avoid_fields answer option
 women_girls_boys_avoid_firewood answer option
 women_girls_boys_avoid_places answer option
 other_safety_measures answer option
 dnk answer option
 pnta answer option
 men_avoid_places_weight Numeric weight for men_avoid_places (0–2). Default is 1.
 men_avoid_night_weight Numeric weight for men_avoid_night (0–2). Default is 1.
 .keep_weighted Logical, whether to keep the weighted columns in the output data frame. Default is FALSE.

Value

data frame with additional columns:

- comp_prot_score_prot_needs_3
- comp_prot_score_movement

add_prot_score_practices

Add composite score for Ability to Participate in Safe Practices and Activities

Description

Creates weighted scores for two multi-choice questions about threats affecting household members' ability to engage in activities and social interactions. Then computes an overall severity category (1–4) from the combined scores.

Usage

```
add_prot_score_practices(
  df,
  sep = "/",
  prot_needs_2_activities = "prot_needs_2_activities",
  yes_work = "yes_work",
  yes_livelihood = "yes_livelihood",
  yes_safety = "yes_safety",
  yes_farm = "yes_farm",
  yes_water = "yes_water",
  yes_other_activities = "yes_other_activities",
  yes_free_choices = "yes_free_choices",
  prot_needs_2_social = "prot_needs_2_social",
  yes_visiting_family = "yes_visiting_family",
  yes_visiting_friends = "yes_visiting_friends",
  yes_community_events = "yes_community_events",
  yes_joining_groups = "yes_joining_groups",
  yes_other_social = "yes_other_social",
  yes_child_recreation = "yes_child_recreation",
  yes_decision_making = "yes_decision_making",
  no = "no",
  dnk = "dnk",
  pnta = "pnta",
  .keep_weighted = FALSE
)
```

Arguments

<code>df</code>	A data frame containing binary indicator columns. Each question has multiple "option" columns named using the pattern <code>question\(<code>sep</code>\)option_suffix</code> .
<code>sep</code>	Separator between question names and suffixes in the column names. Defaults to <code>"/"</code> .
<code>prot_needs_2_activities</code>	Name of the first question (activities). Default: <code>"prot_needs_2_activities"</code> .
<code>yes_work</code>	Answer option name for "Yes, it affected the ability to work" option. Default: <code>"yes_work"</code> .
<code>yes_livelihood</code>	Answer option name for "Yes, it affected the general access to livelihood" column. Default: <code>"yes_livelihood"</code> . "Yes, it affected the general access to livelihood" option. Default: <code>"yes_livelihood"</code> .
<code>yes_safety</code>	Answer option name for "Yes, it affected the ability to seek safety" column. Default: <code>"yes_safety"</code> . "Yes, it affected the ability to seek safety" option. Default: <code>"yes_safety"</code> .
<code>yes_farm</code>	Answer option name for "Yes, it affected the ability to farm" column. Default: <code>"yes_farm"</code> . "Yes, it affected the ability to farm" option. Default: <code>"yes_farm"</code> .
<code>yes_water</code>	Answer option name for "Yes, it affected the ability to collect water" column. Default: <code>"yes_water"</code> . "Yes, it affected the ability to collect water" option. Default: <code>"yes_water"</code> .
<code>yes_other_activities</code>	Answer option name for "Yes, it affected other activities needed to meet needs" column. Default: <code>"yes_other_activities"</code> . "Yes, it affected other activities needed to meet needs" option. Default: <code>"yes_other_activities"</code> .
<code>yes_free_choices</code>	Answer option name for "Yes, it affected the ability to make free choices" column. Default: <code>"yes_free_choices"</code> . "Yes, it affected the ability to make free choices" option. Default: <code>"yes_free_choices"</code> .
<code>prot_needs_2_social</code>	Name of the second question (social interactions). Default: <code>"prot_needs_2_social"</code> .
<code>yes_visiting_family</code>	Answer option name for "Yes, visiting family members" column. Default: <code>"yes_visiting_family"</code> . "Yes, visiting family members" option. Default: <code>"yes_visiting_family"</code> .
<code>yes_visiting_friends</code>	Answer option name for "Yes, visiting friends" column. Default: <code>"yes_visiting_friends"</code> . "Yes, visiting friends" option. Default: <code>"yes_visiting_friends"</code> .
<code>yes_community_events</code>	Answer option name for "Yes, attending community events" column. Default: <code>"yes_community_events"</code> . "Yes, attending community events" option. Default: <code>"yes_community_events"</code> .
<code>yes_joining_groups</code>	Answer option name for "Yes, joining groups or public gatherings" column. Default: <code>"yes_joining_groups"</code> . "Yes, joining groups or public gatherings" option. Default: <code>"yes_joining_groups"</code> .

yes_other_social	Answer option name for "Yes, participating in other social activities" column. Default: "yes_other_social". "Yes, participating in other social activities" option. Default: "yes_other_social".
yes_child_recreation	Answer option name for "Yes, children's recreational activities" column. Default: "yes_child_recreation". "Yes, children's recreational activities" option. Default: "yes_child_recreation".
yes_decision_making	Answer option name for "Yes, participating in decision making bodies" column. Default: "yes_decision_making". "Yes, participating in decision making bodies" option. Default: "yes_decision_making".
no	Answer option name for "No" column. Default: "no". "No" option. Default: "no".
dnk	Answer option name for "Don't know" column (treated as NA). Default: "dnk". "Don't know" option (treated as NA). Default: "dnk".
pnta	Answer option name for "Prefer not to answer" column (treated as NA). Default: "pnta". "Prefer not to answer" option (treated as NA). Default: "pnta".
.keep_weighted	Logical; if TRUE, retains the intermediate weighted columns (suffix "_w"). Default: FALSE.

Value

The input data frame with three new composite-score columns:

- comp_prot_score_prot_needs_2_activities: weighted sum of activity-related options.
- comp_prot_score_prot_needs_2_social: weighted sum of social-related options.
- comp_prot_score_practices: overall severity (1–4) based on combined score.

Plus optional weighted columns (suffix "_w") if .keep_weighted = TRUE.

add_prot_score_rights *Add composite score for Ability to Access Rights and Services*

Description

Computes weighted sums for two questions about household difficulties accessing services and justice/legal resources, then derives an overall severity category (1–4).

Usage

```
add_prot_score_rights(
  df,
  sep = "/",
  prot_needs_1_services = "prot_needs_1_services",
  yes_healthcare = "yes_healthcare",
```

```

yes_schools = "yes_schools",
yes_therapeutic_services = "yes_therapeutic_services",
yes_edu_facilities = "yes_edu_facilities",
yes_social_services = "yes_social_services",
yes_gov_services = "yes_gov_services",
yes_other_services = "yes_other_services",
prot_needs_1_justice = "prot_needs_1_justice",
yes_identity_documents = "yes_identity_documents",
yes_counselling_legal = "yes_counselling_legal",
yes_property_docs = "yes_property_docs",
yes_birth_certificates = "yes_birth_certificates",
yes_gov_services_justice = "yes_gov_services",
yes_other_services_justice = "yes_other_services",
none = "none",
no = "no",
dnk = "dnk",
pnta = "pnta",
.keep_weighted = FALSE
)

```

Arguments

df	A data frame containing binary indicator columns for both questions.
sep	Separator between question code and answer option in column names. Default: "/".
prot_needs_1_services	Base name of services question. Default: "prot_needs_1_services".
yes_healthcare	Answer option name for "Yes, accessing healthcare". Default: "yes_healthcare".
yes_schools	Answer option name for "Yes, accessing schools". Default: "yes_schools".
yes_therapeutic_services	Answer option name for "Yes, therapeutic centers". Default: "yes_therapeutic_services".
yes_edu_facilities	Answer option name for "Yes, education facilities". Default: "yes_edu_facilities".
yes_social_services	Answer option name for "Yes, social services including referral services, and safe spaces for children". Default: "yes_social_services".
yes_gov_services	Answer option name for "Yes, accessing governmental services". Default: "yes_gov_services".
yes_other_services	Answer option name for "Yes, accessing other services". Default: "yes_other_services".
prot_needs_1_justice	Base name of justice question. Default: "prot_needs_1_justice".
yes_identity_documents	Answer option name for "Yes, accessing identity and civil documents services". Default: "yes_identity_documents".

yes_counselling_legal	Answer option name for "Yes, individual counselling or legal assistance". Default: "yes_counselling_legal".
yes_property_docs	Answer option name for "Yes, accessing house, land and property documentation". Default: "yes_property_docs".
yes_birth_certificates	Answer option name for "Yes, accessing birth certificates". Default: "yes_birth_certificates".
yes_gov_services_justice	Answer option name for "Yes, accessing governmental services". Default: "yes_gov_services".
yes_other_services_justice	Answer option name for "Yes, accessing other services". Default: "yes_other_services".
none	Answer option name for "None" (services' no-barrier option). Default: "none".
no	Answer option name for "No" (justice's no-barrier option). Default: "no".
dnk	Answer option name for "Don't know" Default: "dnk".
pnta	Answer option name for "Prefer not to answer" Default: "pnta".
.keep_weighted	Logical; if TRUE, retain intermediate weighted columns with suffix _w. Default: FALSE.

Value

Input data frame with three new composite-score columns:

- comp_prot_score_prot_needs_1_services: weighted sum of services options.
- comp_prot_score_prot_needs_1_justice: weighted sum of justice options.
- comp_prot_score_rights: overall severity (1–4) based on combined score. Plus optional _w columns if .keep_weighted = TRUE.

add_quantile_interval *Add Quantile Intervals to Income Variable*

Description

This function calculates weighted income quantiles and classifies households according to these. It can handle multiple income variables and applies weighted or unweighted quantiles based on the input.

Usage

```
add_quantile_interval(
  df,
  vars,
  cut_offs = c(0, 0.2, 0.4, 0.6, 0.8, 1),
  weight = NULL
)
```

Arguments

df	A dataframe containing the income variables.
vars	The variable or variables with income integer values.
cut_offs	The cut-offs for computation of quantiles. The default corresponds to the quantiles of the distribution, based on the weighted median.
weight	The sampling weights column name. If NULL, unweighted quantiles are calculated.

Value

- A dataframe with additional columns:
- *_qtl: For each input variable, a new column is added with the quantile interval classification. The naming format is 'original_variable_name'_qtl.

add_rcsi	<i>Add indicator for reduced Household CSI Score(rcsi)</i>
----------	--

Description

Add indicator for reduced Household CSI Score(rcsi)

Usage

```
add_rcsi(  
  .dataset,  
  fsl_rcsi_lessquality = "fsl_rcsi_lessquality",  
  fsl_rcsi_borrow = "fsl_rcsi_borrow",  
  fsl_rcsi_mealsize = "fsl_rcsi_mealsize",  
  fsl_rcsi_mealadult = "fsl_rcsi_mealadult",  
  fsl_rcsi_mealnb = "fsl_rcsi_mealnb"  
)
```

Arguments

.dataset	Main Dataset
fsl_rcsi_lessquality	Column representing question- During the last 7 days, were there days (and, if so, how many) when your household had to rely on less preferred and less expensive food to cope with a lack of food or money to buy it?
fsl_rcsi_borrow	Column representing question- During the last 7 days, were there days (and, if so, how many) when your household had to borrow food or rely on help from a relative or friend to cope with a lack of food or money to buy it?

fsl_rcsi_mealsize

Column representing question- During the last 7 days, were there days (and, if so, how many) when your household had to limit portion size of meals at meal times to cope with a lack of food or money to buy it?

fsl_rcsi_mealadult

Column representing question- During the last 7 days, were there days (and, if so, how many) when your household had to restrict consumption by adults in order for small children to eat to cope with a lack of food or money to buy it?

fsl_rcsi_mealnb

Column representing question - During the last 7 days, were there days (and, if so, how many) when your household had to reduce number of meals eaten in a day to cope with a lack of food or money to buy it?

Value

A dataset with 7 additional columns: 5 weighted strategy columns (rcsi*_weighted), fsl_rcsi_score, and fsl_rcsi_cat.

Examples

```
test_data <- data.frame(
  fsl_rcsi_lessquality = c(1, 2, 3, 1),
  fsl_rcsi_borrow = c(0, 0, 3, 0),
  fsl_rcsi_mealsize = c(4, 2, 6, 1),
  fsl_rcsi_mealadult = c(4, 3, 5, 0),
  fsl_rcsi_mealnb = c(2, 5, NA_integer_, 1)
)
add_rcsi(test_data)
```

add_received_assistance

Add Received Assistance (Combined Calculation)

Description

This function calculates and adds a new column for received assistance based on two input columns: assistance received in the last 12 months and the recall period of the assistance.

Usage

```
add_received_assistance(
  df,
  received_assistance_12m = "aap_received_assistance_12m",
  yes = "yes",
  no = "no",
  undefined = c("dnk", "pnta"),
  received_assistance_date = "aap_received_assistance_date",
  date_past_30d = "past_30d",
```

```

date_1_3_months = "1_3_months",
date_4_6_months = "4_6_months",
date_7_12_months = "7_12_months",
date_undefined = c("dnk", "pnta")
)

```

Arguments

<code>df</code>	A data frame containing assistance information.
<code>received_assistance_12m</code>	Column name for received assistance in the last 12 months.
<code>yes</code>	Value for yes.
<code>no</code>	Value for no.
<code>undefined</code>	Vector of undefined responses.
<code>received_assistance_date</code>	Column name for the recall period when the assistance was received.
<code>date_past_30d</code>	Value for received assistance in the past 30 days.
<code>date_1_3_months</code>	Value for received assistance in the last 1-3 months.
<code>date_4_6_months</code>	Value for received assistance in the last 4-6 months.
<code>date_7_12_months</code>	Value for received assistance in the last 7-12 months.
<code>date_undefined</code>	Vector of undefined responses for the date of received assistance.

Value

A data frame with an additional column:

- `aap_received_assistance`: Categorized received assistance: "undefined", "no", "past_30d", "1_3_months", "4_6_months", or "7_12_months".

`add_sanitation_facility_cat`
Sanitation Facility Classification

Description

This set of functions classifies sanitation facilities according to various criteria. It includes functions to categorize sanitation facility types, sharing status, number of individuals sharing, and JMP (Joint Monitoring Programme) classification.

`add_sharing_sanitation_facility_cat`: Recode sharing status of sanitation facilities. Maps user responses to `shared`, `not_shared`, or `not_applicable` when the sanitation facility was skipped.

add_sharing_sanitation_facility_n_ind: Calculate the number of individuals sharing a sanitation facility per household. Uses household size and survey weights to estimate and then categorizes counts into predefined bands.

add_sanitation_facility_jmp_cat: Combine sanitation facility type and sharing status to derive a JMP classification (e.g., basic, limited, unimproved, open_defecation, undefined).

Usage

```
add_sanitation_facility_cat(
  df,
  sanitation_facility = "wash_sanitation_facility",
  improved = c("flush_piped_sewer", "flush_septic_tank", "flush_pit_latrine",
    "flush_dnk_where", "pit_latrine_slab", "twin_pit_latrine_slab",
    "ventilated_pit_latrine_slab", "container", "compost"),
  unimproved = c("flush_open_drain", "pit_latrine_wo_slab", "bucket", "hanging_toilet",
    "twin_pit_latrine_wo_slab", "other_container"),
  none = "none",
  undefined = c("other", "dnk", "pnta")
)

add_sharing_sanitation_facility_cat(
  df,
  sharing_sanitation_facility = "wash_sanitation_facility_sharing_yn",
  yes = "yes",
  no = "no",
  undefined = c("dnk", "pnta"),
  sanitation_facility = "wash_sanitation_facility",
  skipped_sanitation_facility = c("none")
)

add_sharing_sanitation_facility_n_ind(
  df,
  sharing_sanitation_facility_cat = "wash_sharing_sanitation_facility_cat",
  sharing_sanitation_facility_cat_shared = "shared",
  sharing_sanitation_facility_cat_not_shared = "not_shared",
  sharing_sanitation_facility_cat_not_applicable = "not_applicable",
  sharing_sanitation_facility_cat_undefined = "undefined",
  sanitation_facility_sharing_n = "wash_sanitation_facility_sharing_n",
  hh_size = "hh_size",
  weight = "weight"
)

add_sanitation_facility_jmp_cat(
  df,
  sanitation_facility_cat = "wash_sanitation_facility_cat",
  sanitation_facility_cat_improved = "improved",
  sanitation_facility_cat_unimproved = "unimproved",
  sanitation_facility_cat_none = "none",
```

```

sanitation_facility_cat_undefined = "undefined",
sharing_sanitation_facility_cat = "wash_sharing_sanitation_facility_cat",
sharing_sanitation_facility_cat_shared = "shared",
sharing_sanitation_facility_cat_not_shared = "not_shared",
sharing_sanitation_facility_cat_not_applicable = "not_applicable",
sharing_sanitation_facility_cat_undefined = "undefined"
)

```

Arguments

<code>df</code>	A data frame containing both sanitation facility types and sharing status information.
<code>sanitation_facility</code>	Column name for sanitation facility types.
<code>improved</code>	Character vector of response codes for Improved facilities.
<code>unimproved</code>	Character vector of response codes for Unimproved facilities.
<code>none</code>	Character vector of response codes for No sanitation facility/Open defecation.
<code>undefined</code>	Character vector of response codes indicating undefined responses (e.g., c("dnk", "pnta")).
<code>sharing_sanitation_facility</code>	Component column: Number of people with whom the facility is shared.
<code>yes</code>	Character vector of response codes for Yes.
<code>no</code>	Character vector of response codes for No.
<code>skipped_sanitation_facility</code>	Character vector of response codes for skipped sanitation facility.
<code>sharing_sanitation_facility_cat</code>	Component column: Sharing status of sanitation facility recoded.
<code>sharing_sanitation_facility_cat_shared</code>	Level: Shared sanitation facility.
<code>sharing_sanitation_facility_cat_not_shared</code>	Level: Not shared sanitation facility.
<code>sharing_sanitation_facility_cat_not_applicable</code>	Level: Not applicable sharing status.
<code>sharing_sanitation_facility_cat_undefined</code>	Level: Undefined sharing status.
<code>sanitation_facility_sharing_n</code>	Component column: Number of households sharing the sanitation facility.
<code>hh_size</code>	Column name for household size.
<code>weight</code>	Column name for survey weights. For unweighted analysis, this column must still be provided, with its value set to 1.
<code>sanitation_facility_cat</code>	Component column: Sanitation facility types recoded.
<code>sanitation_facility_cat_improved</code>	Level: Improved sanitation facility.

```

sanitation_facility_cat_unimproved
    Level: Unimproved sanitation facility.
sanitation_facility_cat_none
    Level: No sanitation facility.
sanitation_facility_cat_undefined
    Level: Undefined sanitation facility.

```

Value

A data frame with an additional column:

- wash_sanitation_facility_cat: Categorized sanitation facility: "none", "unimproved", "improved", or "undefined".

A data frame with an additional column:

- wash_sharing_sanitation_facility_cat: Categorized sharing status: "shared", "not_shared", or "not_applicable".

A data frame with additional columns:

- wash_sanitation_facility_sharing_n_calc: Estimated number of individuals sharing the sanitation facility.
- wash_sharing_sanitation_n_ind: Categorized number of individuals sharing a sanitation facility.

A data frame with an additional column:

- wash_sanitization_jmp_cat: Categorized JMP classification based on both type and sharing status.

add_shelter_damage_cat

Add Category of Shelter damage (SNFI dimension)

Description

This function categorizes the shelter damage category based on provided criteria.

Usage

```

add_shelter_damage_cat(
  df,
  sep = "/",
  snfi_shelter_damage = "snfi_shelter_damage",
  none = "none",
  minor = "minor_roof",
  major = "major_roof",
  damage_windows_doors = "windows_doors",

```

```
    damage_floors = "floors",
    damage_walls = "walls",
    total_collapse = "total_collapse",
    other = "other",
    dnk = "dnk",
    pnta = "pnta"
)
```

Arguments

df	Data frame containing the survey data.
sep	Separator for the binary columns, default is "/".
snfi_shelter_damage	Column name
none	answer option
minor	answer option
major	answer option
damage_windows_doors	answer option
damage_floors	answer option
damage_walls	answer option
total_collapse	answer option
other	answer option
dnk	answer option
pnta	answer option

Value

A data frame with an additional column:

- snfi_shelter_damage_cat: Categorized shelter damages: "none", "damaged", "part", "total", or "undefined".

add_shelter_issue_cat	<i>Add Number of Shelter Issues and Related Category</i>
-----------------------	--

Description

This function calculates the number of shelter issues and categorizes them based on predefined thresholds. It also handles undefined and other responses.

Usage

```
add_shelter_issue_cat(
  df,
  shelter_issue = "snfi_shelter_issue",
  none = "none",
  issues = c("lack_privacy", "lack_space", "temperature", "ventilation", "vectors",
    "no_natural_light", "leak", "lock", "lack_lighting", "difficulty_move",
    "lack_space_laundry"),
  undefined = c("dnk", "pnta"),
  other = c("other"),
  sep = "/"
)
```

Arguments

df	A data frame containing shelter issue data.
shelter_issue	Component column: Shelter issues.
none	Response code for no issue.
issues	Character vector of issues.
undefined	Character vector of undefined responses codes (e.g. "Prefer not to answer").
other	Character vector of other responses codes (e.g. "Other").
sep	Separator for the binary columns.

Value

A data frame with additional columns:

- snfi_shelter_issue_n: Count of shelter issues.
- snfi_shelter_issue_cat: Categorized shelter issues: "none", "undefined", "other", "1_to_3", "4_to_7", or "8_to_11".

add_shelter_type_cat *Combine and Recode Shelter Types*

Description

This function combines both shelter types questions and recodes the type of shelter into categories such as "none", "inadequate", "adequate", or "undefined".

Usage

```
add_shelter_type_cat(  
  df,  
  shelter_type = "snfi_shelter_type",  
  sl_none = "none",  
  sl_collective_center = "collective_center",  
  sl_undefined = "pnta",  
  shelter_type_individual = "snfi_shelter_type_individual",  
  adequate = c("house", "apartment"),  
  inadequate = c("makeshift", "unfinished_building", "tent"),  
  undefined = c("pnta", "other", "dnk")  
)
```

Arguments

df	A data frame containing shelter type information.
shelter_type	Component column: Shelter type categories.
sl_none	Character vector of responses codes for none/sleeping in the open that are skipped.
sl_collective_center	Character vector of responses codes for collective center that are skipped.
sl_undefined	Character vector of undefined responses codes (e.g. "Prefer not to answer") that are skipped.
shelter_type_individual	Component column: Individual shelter types.
adequate	Character vector of responses codes for adequate shelter types.
inadequate	Character vector of responses codes for inadequate shelter types.
undefined	Character vector of responses codes for undefined shelter types.

Value

- A data frame with an additional column:
- snfi_shelter_type_cat: Categorized shelter type: "none", "inadequate", "adequate", or "undefined".

are_cols_numeric	<i>Stop statement values are not numeric</i>
------------------	--

Description

Stop statement values are not numeric

Usage

```
are_cols_numeric(df, cols)
```

Arguments

df	A data frame
cols	A vector of column names (quoted)

Value

A stop statement

are_values_in_range	<i>Stop statement values are not in range</i>
---------------------	---

Description

Stop statement values are not in range

Usage

```
are_values_in_range(df, cols, lower = 0, upper = 7)
```

Arguments

df	A data frame
cols	A vector of column names (quoted)
lower	Lower bound
upper	Upper bound

Value

A stop statement

are_values_in_set	<i>Stop statement values are not in set</i>
-------------------	---

Description

Stop statement values are not in set

Usage

```
are_values_in_set(  
  df,  
  cols,  
  set,  
  main_message = "All values must be in the following set: "  
)
```

Arguments

df	A data frame
cols	A vector of column names (quoted)
set	A vector of values
main_message	A main message

Value

A stop statement

count_if_above_zero	<i>Create Dummy Variables and Count Across if Above Zero</i>
---------------------	--

Description

This function creates dummy variables for each input column, setting 1 if the value is above zero, 0 if it's zero, and NA if it's below zero. It then sums these dummy variables to create a count.

Usage

```
count_if_above_zero(df, vars, new_colname)
```

Arguments

df	A data frame containing the columns to be processed.
vars	Vector of column names to be processed.
new_colname	The name of the new column that will contain the count.

Value

A data frame with additional columns:

- `"*_d"`: Dummy variables for each input column, where 'star' is the original column name.
- `new_colname`: A new column containing the count of values above zero across the specified variables.

drop_shared_loop_cols	<i>Drop columns from main that are also computed in loop</i>
-----------------------	--

Description

Before joining loop-derived columns into main, drop any column names main already shares with loop (e.g. from a prior run), except the two id columns used for the join.

Usage

drop_shared_loop_cols(main, loop, id_col_main, id_col_loop)

Arguments

- main A data frame (household-level)
- loop A data frame (individual-level, already summarized)
- id_col_main Column name for the unique identifier in main
- id_col_loop Column name for the unique identifier in loop

Value

main, with shared non-id columns dropped

humind_edu_ind	<i>Dummy demo MSNA education loop data</i>
----------------	--

Description

Dummy individual-level education data modeled on the 2026 MSNA guidance, used to illustrate the humind sector-by-sector MSNI pipeline in vignette("msni-workflow").

Usage

humind_edu_ind

Format

- A data frame with 26,630 rows and 8 variables:
- _submission__uuid** Join key back to [humind_main](#)'s _uuid.
 - edu_ind_age** Individual's age, in years.
 - edu_access** Whether the individual is currently attending education.
 - edu_barrier** Barriers to education access.
 - edu_disrupted_attack, edu_disrupted_hazards, edu_disrupted_displaced, edu_disrupted_teacher** Reasons education was disrupted.

Source

Dummy data modeled on the 2026 MSNA guidance, for demonstration purposes only.

humind_health_ind	<i>Dummy MSNA health loop data</i>
-------------------	------------------------------------

Description

Dummy individual-level health data modeled on the 2026 MSNA guidance, used to illustrate the humind sector-by-sector MSNI pipeline in vignette("msni-workflow").

Usage

humind_health_ind

Format

A data frame with 30,300 rows and 3 variables:

- _submission__uuid** Join key back to [humind_main](#)'s _uuid.
- health_ind_healthcare_needed** Whether the individual needed healthcare.
- health_ind_healthcare_received** Whether the individual received the healthcare they needed.

Source

Dummy data modeled on the 2026 MSNA guidance, for demonstration purposes only.

humind_main	<i>Dummy MSNA household (main) data</i>
-------------	---

Description

Dummy household-level data modeled on the 2026 MSNA guidance, used to illustrate the humind sector-by-sector MSNI pipeline in vignette("msni-workflow").

Usage

humind_main

Format

A data frame with 5,408 rows and 148 variables, grouped by sector and prefix:

- `_uuid`: household unique identifier; join key for `humind_health_ind` and `humind_edu_ind` (via `_submission__uuid`).
- `fsl_*`: food security items (FCS, HHS, rCSI, LCSi incl. `_host/_camp` variants) – see `add_fcs()`, `add_hhs()`, `add_rcsi()`, `add_lcsi()`.
- `wash_*`: WASH items (HWISE, drinking water, sanitation, handwashing) – see `add_hwise()`, `add_sanitation_facility_cat()`.
- `snfi_*`, `hlp_*`: shelter, damage, occupancy – see `add_shelter_type_cat()`, `add_occupancy_cat()`.
- `prot_needs_*`: protection movement/practices/rights items – see `add_prot_score_movement()`.
- `admin1`: geography, for context only; unused by any `add_*`().

Source

Dummy data modeled on the 2026 MSNA guidance, for demonstration purposes only.

<code>if_not_in_stop</code>	<i>Stop statement "If not in colnames" with colnames</i>
-----------------------------	--

Description

Stop statement "If not in colnames" with colnames

Usage

```
if_not_in_stop(df, cols, df_name, arg = NULL)
```

Arguments

<code>df</code>	A data frame
<code>cols</code>	A vector of column names (quoted)
<code>df_name</code>	Provide the tibble name as a character string
<code>arg</code>	Default to NULL.

Value

A stop statement

impute_value	<i>Impute Missing Values</i>
--------------	------------------------------

Description

This set of functions replaces missing values in specified variables.

- `impute_value()` replaces all missing values with a specific value, while
- `impute_median()` replaces missing values with the (weighted) median of the variable.

Usage

```
impute_value(df, vars, value)
```

```
impute_median(df, vars, group = NULL, weighted = FALSE, weight = NULL)
```

Arguments

<code>df</code>	A dataframe containing the variables to be imputed.
<code>vars</code>	A character vector of the variables to replace NA with.
<code>value</code>	The value to replace NA with.
<code>group</code>	A character vector of the grouping variables.
<code>weighted</code>	A boolean indicating whether to use the weighted median or not.
<code>weight</code>	The weight variable.

Value

A dataframe with imputed values:

- `vars`: The specified variables with missing values replaced.

A dataframe with imputed values:

- `vars`: The specified variables with missing values replaced by their respective (weighted) medians.

is_in_need

*Add Dummy Variable for 'In Need' Status***Description**

This set of functions adds new binary variables indicating if a score is above certain thresholds. `is_in_need()` adds a variable for scores above 3 (indicating "in need"), while `is_in_severe_need()` adds a variable for scores above 4 (indicating "in severe need").

Usage

```
is_in_need(df, score, new_colname = NULL)
```

```
is_in_severe_need(df, score, new_colname = NULL)
```

Arguments

<code>df</code>	A dataframe containing the score variable.
<code>score</code>	The variable containing the score on a 1-5 scale.
<code>new_colname</code>	The name of the new column. If NULL, the function will create a new column with the name 'score_in_need' or 'score_in_severe_need'.

Value

A dataframe with an additional column:

- `new_colname`: A binary variable indicating if the score meets the threshold for being "in need" (1) or not (0).

num_cat

*Add Categories for a Numeric Variable***Description**

This function categorizes a numeric column in a dataframe based on specified breaks and labels. It can handle undefined values and provides options for customizing the categorization process.

Usage

```
num_cat(
  df,
  num_col,
  breaks,
  labels = NULL,
  int_undefined = c(-999, 999),
```

```
char_undefined = "Unknown",
new_colname = NULL,
plus_last = FALSE,
above_last = FALSE
)
```

Arguments

df	A dataframe containing the numeric column to be categorized.
num_col	The column name to recategorize.
breaks	A vector of cut points for categorization.
labels	A vector of labels for the categories. If NULL, labels are automatically generated.
int_undefined	A vector of numeric values to be replaced by char_undefined.
char_undefined	A character string to replace int_undefined values.
new_colname	The name of the new column. If NULL, it's automatically generated.
plus_last	Logical, whether to add a "+" to the last category.
above_last	Logical, whether to add a category for values above the last break.

Value

- A dataframe with an additional column:
- new_colname: A new column containing the categorized values of the original numeric column.

rank_top3_vars	<i>Rank Top 3 Variables</i>
----------------	-----------------------------

Description

This function ranks the top 3 numeric variables for each row in a data frame. In case of equality, it takes precedence in the order of 'vars' as factors.

Usage

```
rank_top3_vars(
  df,
  vars,
  new_colname_top1,
  new_colname_top2,
  new_colname_top3,
  id_col = "uuid"
)
```

Arguments

- df A data frame containing the variables to be ranked.
- vars A character vector of numeric variable names to be ranked.
- new_colname_top1 The new column name for the top variable.
- new_colname_top2 The new column name for the 2nd top variable.
- new_colname_top3 The new column name for the 3rd top variable.
- id_col The column name for the unique identifier.

Value

- A data frame with additional columns:
- new_colname_top1: The name of the top-ranked variable for each row.
 - new_colname_top2: The name of the second-ranked variable for each row.
 - new_colname_top3: The name of the third-ranked variable for each row.

subvec_in	<i>Subvec in</i>
-----------	------------------

Description

Subvec in

Usage

subvec_in(vector, set)

Arguments

- vector A vector to subset
- set A set-vector

Value

A subset of a list or a vector

subvec_not_in	<i>Subvec not in</i>
---------------	----------------------

Description

Subvec not in

Usage

```
subvec_not_in(vector, set)
```

Arguments

- | | |
|--------|--------------------|
| vector | A vector to subset |
| set | A set-vector |

Value

A subset of vector not in set

sum_vars	<i>Sum Columns Row-wise</i>
----------	-----------------------------

Description

This function sums up specified columns row-wise in a dataframe, with options for imputation of missing values and weighted calculations.

Usage

```
sum_vars(  
  df,  
  vars,  
  new_colname,  
  imputation = "none",  
  na_rm = FALSE,  
  weight = "weight",  
  value = 0,  
  group = NULL  
)
```

Arguments

df	A dataframe containing the columns to be summed.
vars	A character vector of the columns to sum.
new_colname	A character vector of the new column name.
imputation	The imputation method, either none (default), value, median or weighted median.
na_rm	A boolean indicating whether to remove missing values when summing across columns.
weight	The weight variable to calculate weighted means or medians.
value	The value to replace missing values with if imputation is "value".
group	A character vector of the grouping variables.

Value

A dataframe with an additional column:

- new_colname: the sum of the specified variables, with imputation applied if specified.

value_to_sl

Add a specific value to variables that were skipped

Description

Add a specific value to variables that were skipped

Usage

```
value_to_sl(df, var, undefined = NULL, sl_vars, sl_value, suffix = "")
```

Arguments

df	A data frame.
var	The name of the variable that is used for the skip logic.
undefined	A character vector of values that are considered as "undefined" and should not be replaced in follow-up variables.
sl_vars	A character vector of variable names which were skipped and should be replaced with a value.
sl_value	The value that should be added to the skipped variables.
suffix	A suffix to add to the new variable names. Default to no suffix (empty string).

Index

* datasets

- humind_edu_ind, [79](#)
- humind_health_ind, [80](#)
- humind_main, [80](#)

add_access_to_phone_best, [3](#)
add_access_to_phone_coverage
 (add_access_to_phone_best), [3](#)
add_age_18_cat (add_age_cat), [5](#)
add_age_cat, [5](#)
add_barriers_assistance_any, [6](#)
add_comp_edu, [7](#)
add_comp_foodsec, [8](#)
add_comp_health, [9](#)
add_comp_prot, [10](#)
add_comp_snfi, [11](#)
add_comp_wash, [14](#)
add_comp_wash(), [40](#)
add_drinking_water_quality_jmp_cat
 (add_drinking_water_source_cat),
 [17](#)
add_drinking_water_source_cat, [17](#)
add_drinking_water_source_cat(), [14](#)
add_drinking_water_time_cat
 (add_drinking_water_source_cat),
 [17](#)
add_drinking_water_time_threshold_cat
 (add_drinking_water_source_cat),
 [17](#)
add_expenditure_type_freq_rank, [20](#)
add_expenditure_type_infreq_rank, [21](#)
add_expenditure_type_prop_freq, [22](#)
add_expenditure_type_prop_infreq, [23](#)
add_expenditure_type_zero_freq, [24](#)
add_expenditure_type_zero_infreq, [25](#)
add_fclcm_phase, [26](#)
add_fcm_phase, [28](#)
add_fcs, [30](#)
add_fcs(), [81](#)
add_fds_cannot_cat, [31](#)
add_handwashing_facility_cat, [33](#)
add_handwashing_facility_cat(), [14](#)
add_hhs, [36](#)
add_hhs(), [81](#)
add_hoh_final, [38](#)
add_hwise, [39](#)
add_hwise(), [14](#), [15](#), [81](#)
add_income_source_prop, [41](#)
add_income_source_rank, [42](#)
add_income_source_zero_to_sl, [43](#)
add_lcsi, [44](#)
add_lcsi(), [81](#)
add_loop_age_dummy, [46](#)
add_loop_age_dummy(), [46](#)
add_loop_age_dummy_to_main
 (add_loop_age_dummy), [46](#)
add_loop_age_dummy_to_main(), [46](#)
add_loop_age_gender_dummy
 (add_loop_age_dummy), [46](#)
add_loop_age_gender_dummy(), [46](#)
add_loop_age_gender_dummy_to_main
 (add_loop_age_dummy), [46](#)
add_loop_age_gender_dummy_to_main(),
 [46](#)
add_loop_edu_access_d, [48](#)
add_loop_edu_access_d_to_main
 (add_loop_edu_access_d), [48](#)
add_loop_edu_barrier_protection_d, [49](#)
add_loop_edu_barrier_protection_d_to_main
 (add_loop_edu_barrier_protection_d),
 [49](#)
add_loop_edu_disrupted_d, [51](#)
add_loop_edu_disrupted_d(), [51](#)
add_loop_edu_disrupted_d_to_main
 (add_loop_edu_disrupted_d), [51](#)
add_loop_edu_disrupted_d_to_main(), [51](#)
add_loop_edu_ind_age_corrected, [52](#)
add_loop_edu_ind_schooling_age_d_to_main
 (add_loop_edu_ind_age_corrected),

[52](#)
 add_loop_healthcare_needed_cat, [54](#)
 add_loop_healthcare_needed_cat_to_main
 (add_loop_healthcare_needed_cat),
 [54](#)
 add_loop_wgq_ss, [56](#)
 add_loop_wgq_ss_to_main
 (add_loop_wgq_ss), [56](#)
 add_msni, [58](#)
 add_occupancy_cat, [60](#)
 add_occupancy_cat(), [81](#)
 add_prot_score_movement, [61](#)
 add_prot_score_movement(), [10](#), [81](#)
 add_prot_score_practices, [63](#)
 add_prot_score_practices(), [10](#)
 add_prot_score_rights, [65](#)
 add_prot_score_rights(), [10](#)
 add_quantile_interval, [67](#)
 add_rcsi, [68](#)
 add_rcsi(), [81](#)
 add_received_assistance, [69](#)
 add_sanitation_facility_cat, [70](#)
 add_sanitation_facility_cat(), [14](#), [81](#)
 add_sanitation_facility_jmp_cat
 (add_sanitation_facility_cat),
 [70](#)
 add_sharing_sanitation_facility_cat
 (add_sanitation_facility_cat),
 [70](#)
 add_sharing_sanitation_facility_n_ind
 (add_sanitation_facility_cat),
 [70](#)
 add_shelter_damage_cat, [73](#)
 add_shelter_issue_cat, [74](#)
 add_shelter_type_cat, [75](#)
 add_shelter_type_cat(), [81](#)
 are_cols_numeric, [76](#)
 are_values_in_range, [77](#)
 are_values_in_set, [77](#)

 count_if_above_zero, [78](#)

 drop_shared_loop_cols, [79](#)

 humind_edu_ind, [79](#), [81](#)
 humind_health_ind, [80](#), [81](#)
 humind_main, [79](#), [80](#), [80](#)

 if_not_in_stop, [81](#)

 impute_median(impute_value), [82](#)
 impute_median(), [82](#)
 impute_value, [82](#)
 impute_value(), [82](#)
 is_in_need, [83](#)
 is_in_need(), [83](#)
 is_in_severe_need(is_in_need), [83](#)
 is_in_severe_need(), [83](#)

 num_cat, [83](#)

 pmax(), [11](#)
 purrr::iwalk(), [11](#)

 rank_top3_vars, [84](#)

 subvec_in, [85](#)
 subvec_not_in, [86](#)
 sum_vars, [86](#)

 value_to_sl, [87](#)