# ihsMW User Guide

Cleaning and harmonising Malawi Integrated Household Survey data in R

Vitumbiko Kayuni

29 July 2026

## Contents

# About this guide

This is the complete manual for **ihsMW**, an R package for cleaning, harmonising and analysing the Malawi Integrated Household Survey (IHS).

It is written to be read start to finish by someone who has never used the package, and then kept nearby as a reference. Every code block that does not require microdata runs as written — the guide is generated by executing its own examples, so what you see is what the package actually produces.

**Three ways to read it.** Online at https://vituk123.github.io/ihsMW/articles/ihsMW-user-guide.html, as a PDF for printing and offline use, or as R Markdown source if you want to re-run and adapt every example yourself.

**What this guide assumes.** Basic R: you can install a package, read a data file, and subset a data frame. It does not assume you know anything about the IHS, about survey weights, or about `dplyr`.

---

# 1. Installation

## From CRAN

```r
install.packages("ihsMW")
```

## Development version

```r
# install.packages("pak")
pak::pak("vituk123/ihsMW")

# or
# install.packages("remotes")
remotes::install_github("vituk123/ihsMW")
```

### Confirm the installation

```
library(ihsMW)
packageVersion("ihsMW")
#> [1] '1.1.1'
```

`ihsMW` requires **R 4.1.0 or later**, because it uses the native pipe (`|>`) internally.

---

## 2. Package dependencies

`ihsMW` is deliberately light. Everything needed for the core cleaning and harmonisation workflow is a hard dependency; anything optional is suggested, so a minimal install stays small.

### Required (installed automatically)

| Package | Minimum | Used for |
|---|---|---|
| `dplyr` | 1.1.0 | joins, grouping, aggregation |
| `readr` | 2.1.0 | reading the bundled crosswalk and reference CSVs |
| `rlang` | 1.1.0 | argument matching and tidy evaluation |
| `cli` | 3.6.0 | the informative console messages and errors |

### Suggested (install if you need the feature)

| Package | Needed for |
|---|---|
| `haven` | reading Stata `.dta` microdata — **you will almost certainly want this** |
| `survey` | `ihs_svydesign()` and design-based standard errors |
| `srvyr` | `dplyr`-style syntax over survey designs |
| `testthat`, `withr`, `jsonlite` | running the package test suite |
| `knitr`, `rmarkdown`, `pkgdown` | rebuilding this guide and the website |

Install the ones most people want in one go:

```
install.packages(c("haven", "survey", "srvyr"))
```

### No internet required

Once installed, `ihsMW` never makes a network request. The crosswalk, conversion factors and CPI series are all bundled inside the package. This is deliberate: survey work often happens on airgapped or restricted machines, and an analysis that silently depends on a live API is not reproducible.

---

## 3. Package overview

### The problem

The IHS is Malawi's flagship household survey, run roughly every five years by the National Statistical Office with World Bank LSMS support. It is the evidence base for poverty measurement, food security monitoring and agricultural policy in Malawi.

Working with it raw is painful in specific, repetitive ways:

1. **Variable names drift between rounds.** The same concept is `hhweight` in IHS2 and IHS3 but `hh_wgt` from IHS4 onward. The design stratum is `stratum` through IHS5 and `strata` in IHS6. Multiply this across thousands of variables.
2. **Missing values are coded as numbers.** `-99` means "refused", `-98` means "don't know". Take a mean without recoding and your average household is younger than zero.
3. **Harvest quantities are in whatever unit the household used.** Pails, ox-carts, heaps, 50 kg bags — all in the same column. Summing it is meaningless without crop-, unit- and region-specific conversion factors.
4. **Money is nominal.** Malawi's price level rose roughly 24-fold between 2004 and 2025. Comparing kwacha across rounds without deflating produces nonsense.
5. **The sample is not simple random.** It is a stratified two-stage cluster design. Ignore the weights and your estimates are biased; ignore the clustering and your standard errors are far too small.

Every research team solves these five problems again, in their own scripts, with their own undocumented judgement calls. `ihsMW` solves them once.

## What the package does

Thirteen exported functions covering the pipeline from raw file to publication table:

```
#> Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
#> Use 'xfun::attr2()' instead.
#> See help("Deprecated")
#> Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
#> Use 'xfun::attr2()' instead.
#> See help("Deprecated")
```

| Stage | Function | Purpose |
|---|---|---|
| Discover | ihs_search() | Find a variable by keyword across every round |
| Discover | ihs_crosswalk_check() | See how much of the crosswalk is comparable across rounds |
| Discover | ihs_panel_ids() | Look up the ID, weight and stratum columns for a round |
| Discover | ihs_harmonise() | Rename raw columns to consistent harmonised names |
| Clean | ihs_clean() | Recode missing codes and winsorize, with an audit trail |
| Clean | ihs_standardize_missing() | Turn -99/-98/999 into NA |
| Clean | ihs_winsorize() | Cap outliers at percentiles, optionally within strata |
| Transform | ihs_convert_units() | Convert harvest units to kilograms with NSO factors |
| Transform | ihs_aggregate() | Roll individual or plot records up to the household |
| Transform | ihs_merge() | Join modules on auto-detected ID columns |
| Analyse | ihs_deflate() | Convert nominal kwacha to real, constant-price kwacha |
| Analyse | ihs_svydesign() | Build a survey design object for correct standard errors |
| Analyse | ihs_report() | Produce a weighted summary statistics table |

## What the package does *not* do

Being explicit about the boundaries saves you time:

- **It does not distribute microdata.** World Bank licensing forbids it. You download the files yourself (see §5).
- **It does not construct a poverty line or a consumption aggregate.** Those are published with each round; the package helps you clean and compare them, not rebuild them.
- **It does not adjust for spatial price differences.** `ihs_deflate()` applies a national CPI over time. If you need a Lilongwe-vs-Mzuzu price adjustment, use the spatial price index published with the round's consumption aggregate.
- **It does not link households into a panel across rounds.** The IHS panel component has its own ID structure and attrition rules; `ihs_panel_ids()` tells you which columns to use, but the linking logic is yours.
- **It does not make judgement calls for you.** When a variable cannot be mapped or a unit cannot be converted, you get `NA` and a message — never a quietly invented number.

### Supported rounds

```
ihs_panel_ids("all")
#> Panel ID columns across all supported IHS rounds.
#>       role     IHS2     IHS3    IHS4    IHS5    IHS6
#> 1    hh_id  case_id  case_id case_id case_id case_id
#> 2 indiv_id      PID      PID     PID     PID     PID
#> 3    ea_id    ea_id    ea_id   ea_id   ea_id   ea_id
#> 4   strata  stratum  stratum stratum stratum  strata
#> 5   weight hhweight hhweight  hh_wgt  hh_wgt  hh_wgt
```

Note the two renames that catch people out: the weight column changes at IHS4, and the stratum column changes at IHS6. `ihs_svydesign()` handles both automatically.

IHS1 (1997/98) is not supported — it is not available in a form the crosswalk can address.

---

## 4. Basic workflow

Almost every analysis follows the same seven steps.

```
+-------------------------+
| 1. Read raw file        |   haven::read_dta() / readr::read_csv()
+-----------+-------------+
            v
+-------------------------+
| 2. Harmonise names      |   ihs_harmonise(round = "IHS6")
+-----------+-------------+
            v
+-------------------------+
| 3. Merge modules        |   ihs_merge()
+-----------+-------------+
            v
+-------------------------+
| 4. Clean                |   ihs_clean()
+-----------+-------------+
            v
+-------------------------+
| 5. Convert & deflate    |   ihs_convert_units() / ihs_deflate()
+-----------+-------------+
            v
+-------------------------+
| 6. Survey design        |   ihs_svydesign()
+-----------+-------------+
            v
+-------------------------+
| 7. Report               |   ihs_report() / survey::svymean()
+-------------------------+
```

The order matters in two places:

- **Harmonise before merging.** `ihs_merge()` detects join keys by name, and those names only line up after harmonisation.
- **Build the design before subsetting.** Filtering rows out of a data frame destroys the stratum and cluster structure. Create the design on the full data, then use `survey::subset()` on the design object.

---

## 5. Getting the data

The microdata are free but require registration.

1. Go to the World Bank Microdata Library.

2. Search for the round you want:

| Round | Years | Catalogue title |
|---|---|---|
| IHS2 | 2004/05 | Malawi Second Integrated Household Survey |
| IHS3 | 2010/11 | Malawi Third Integrated Household Survey |
| IHS4 | 2016/17 | Malawi Fourth Integrated Household Survey |
| IHS5 | 2019/20 | Malawi Fifth Integrated Household Survey |
| IHS6 | 2024/25 | Malawi Sixth Integrated Household Survey |

3. Request access, stating your research purpose. Approval is usually quick.
4. Download the **Stata (`.dta`)** distribution if you have a choice — see the warning below.
5. Unzip into a folder per round.

### Prefer the Stata distribution

Some IHS CSV exports substitute *value labels* for numeric codes. In the IHS5 CSV release, for instance, `crop_code` contains `"MAIZE LOCAL"` rather than `1`, and the harvest unit contains `"50 KG BAG"` rather than `2`. The conversion factor table is keyed on the numeric codes, so unit conversion cannot work on those files.

`ihs_convert_units()` detects this and stops with an explanatory error rather than handing you a column of `NA`. The fix is to read the `.dta` version:

```
library(haven)
ag <- haven::read_dta("IHS5/ag_mod_i.dta")    # numeric codes, with labels attached
```

`haven` also preserves variable labels, which `ihs_harmonise()` carries through the rename.

---

## 6. Input data requirements

### Format

Any `data.frame` works, including a `tibble` or the labelled data frame that `haven::read_dta()` returns. `ihsMW` returns the same class it was given.

### What each function needs

| Function | Requires |
|---|---|
| `ihs_harmonise()` | Any data frame; column names matching the round's crosswalk entries |
| `ihs_merge()` | Two or more data frames sharing at least one ID column |
| `ihs_standardize_missing()` | Any data frame; only numeric columns are touched |
| `ihs_winsorize()` | Numeric target columns; the `by` column must exist |
| `ihs_convert_units()` | Numeric `crop`, `unit` and quantity columns |
| `ihs_aggregate()` | A grouping column present in the data |
| `ihs_deflate()` | Numeric value columns plus a `round` argument or an `ihs_round` column |
| `ihs_svydesign()` | A weight column; strata and PSU columns if you want them used |
| `ihs_report()` | Numeric variables; grouping and weight columns if used |

### Column naming

After `ihs_harmonise()`, columns carry the crosswalk's `harmonised_name`. Downstream functions detect standard columns case-insensitively, so `HHID` and `hhid` are equivalent. The names each function looks for:

- **Weights:** `hh_wgt`, `hhweight`, `weight`, `panelweight`, `hh_wgt_adj`, `hhwght`
- **Strata:** `stratum`, `strata`, `strataid`, `strat`
- **PSU / cluster:** `ea_id`, `psu`, `cluster`, `clusterid`
- **Join keys:** `case_id`, `hhid`, `hh_id`, `HHID`, `ea_id`, `PID`

If your columns are named differently, pass them explicitly — every auto-detecting function accepts an override.

---

## 7. Output structure

### New columns, never overwritten values

`ihsMW` never modifies a value in place. Transformations add a suffixed column and leave the original intact, so you can always audit or revert:

| Function | Suffix | Example |
|---|---|---|
| `ihs_winsorize()` | `_w` | `food_exp` → `food_exp_w` |
| `ihs_convert_units()` | `_kg` | `quantity` → `quantity_kg` |
| `ihs_deflate()` | `_real` | `food_exp` → `food_exp_real` |

Suffixes compose, which is how you tell at a glance what has been done: `rexp_cat011_w_real` is a winsorized, then deflated, food consumption variable.

`ihs_harmonise()` is the exception — renaming columns is its entire purpose. It also adds an `ihs_round` column, which `ihs_deflate()` later reads.

### Audit attributes

Several functions attach a machine-readable record of what they did:

```
survey_data <- data.frame(
  region   = rep(c("North", "South"), each = 50),
  food_exp = c(rnorm(50, 100, 10), rnorm(50, 500, 50))
)
survey_data$food_exp[1]  <- -99      # a refusal code
survey_data$food_exp[60] <- 99999    # an implausible outlier

cleaned <- ihs_clean(survey_data,
                     winsorize_vars = "food_exp",
                     winsorize_by   = "region")

str(attr(cleaned, "ihs_audit"))
#> List of 6
#>  $ initial_rows       : int 100
#>  $ initial_cols       : int 2
#>  $ missing_conversions:List of 1
#>   ..$ food_exp: int 1
#>  $ winsorized_vars    :List of 1
#>   ..$ food_exp:List of 2
#>   .. ..$ capped_lower: num 2
#>   .. ..$ capped_upper: num 2
#>  $ final_rows         : int 100
#>  $ final_cols         : int 3
```

The audit tells you 1 value was recoded to `NA` and how many observations were capped at each bound. Put this in your appendix and reviewers stop asking how you handled outliers.

Attributes available:

| Attribute | Set by | Contains |
|---|---|---|
| `ihs_audit` | `ihs_clean()` | Row/column counts before and after, plus both logs below |
| `ihs_missing_conversions` | `ihs_standardize_missing()` | Count of values recoded per column |
| `ihs_winsorized_vars` | `ihs_winsorize()` | Count capped at lower and upper bound per variable |
| `ihs_merge_log` | `ihs_merge()` | Row counts at each merge step |

---

# 8. Function reference

Every exported function, in workflow order.

## 8.1 `ihs_search()` — find a variable

```
ihs_search(keyword, round = NULL, fields = c("name", "label", "module"))
```

Searches the bundled crosswalk. Use it before writing any code, to find out what your variable is called in each round.

```
res <- ihs_search("household size")
#> Found 1 variable matching "household size".
res[, c("harmonised_name", "label", "ihs2_name", "ihs5_name", "ihs6_name", "n_rounds")]
#> # A tibble: 1 x 6
#>   harmonised_name label          ihs2_name ihs5_name ihs6_name n_rounds
#>   <chr>           <chr>          <chr>     <chr>     <chr>        <dbl>
#> 1 hhsize          Household size hhsize    hhsize    hhsize           5
```

Restrict to a round to see only variables that exist in it:

```
nrow(ihs_search("livestock", round = "IHS6"))
#> Found 44 variables matching "livestock".
#> [1] 44
```

Search a single field to cut noise:

```
nrow(ihs_search("expenditure", fields = "label"))
#> Found 5 variables matching "expenditure".
#> [1] 5
```

**Returns** a tibble with one row per matching variable, one `ihs*_name` column per round, `n_rounds` (how many rounds it appears in) and `needs_review`.

**Watch out:** `keyword` is a regular expression, so `.` and `(` are special. Search for `"exp"`, not `"exp("`.

---

## 8.2 `ihs_crosswalk_check()` — audit cross-round comparability

```
ihs_crosswalk_check(verbose = TRUE)
```

Prints a coverage report and returns the full crosswalk. Run it before committing to a pooled multi-round analysis.

```
cw <- ihs_crosswalk_check(verbose = FALSE)

nrow(cw)                    # total harmonised variables
#> [1] 6482
```

```
table(cw$n_rounds_avail)    # how many rounds each appears in
#>
#>    1    2    3    4    5
#> 3293  664 1165 1311   49
```

Read that table carefully. Coverage is **not** uniform — only a small core of variables appears in all five rounds, and most appear in one or two. Any claim of a "consistent 20-year series" needs to survive this table first.

```
# Variables available in every round
head(cw$harmonised_name[cw$n_rounds_avail == 5], 15)
#>  [1] "case_id"     "hhid"        "hhsize"      "plotid"      "poor"
#>  [6] "region"      "reside"      "rexp_cat01"  "rexp_cat011" "rexp_cat012"
#> [11] "rexp_cat02"  "rexp_cat021" "rexp_cat022" "rexp_cat03"  "rexp_cat031"
```

### Why full-series coverage is so thin

Two things drive it, and it is worth knowing which is which:

1. **IHS2 is a much smaller instrument.** It has roughly 1,276 variables against IHS6's 3,214. A large share of what later rounds ask simply was not asked in 2004.
2. **The crosswalk is built mostly by matching variable names.** About 98% of its rows link rounds that happen to use the *same* name. IHS2 used an entirely different naming scheme (`ca15`, `cd37`, `sec_a`), so only 85 of its 1,272 mapped variables currently link to a later round.

Point 2 means the all-five-rounds count is a **lower bound**, not a ceiling. Label-based matching would recover more IHS2 links, and contributions on that front are welcome — see §15. Coverage from IHS3 onwards is much better:

```
c(
  `IHS3+IHS4+IHS5`      = sum(!is.na(cw$ihs3_name) & !is.na(cw$ihs4_name) &
                              !is.na(cw$ihs5_name)),
  `IHS4+IHS5+IHS6`      = sum(!is.na(cw$ihs4_name) & !is.na(cw$ihs5_name) &
                              !is.na(cw$ihs6_name)),
  `all five rounds`     = sum(cw$n_rounds_avail == 5)
)
#>  IHS3+IHS4+IHS5  IHS4+IHS5+IHS6 all five rounds
#>            1637            2200              49
```

If your question can be answered from IHS3 onwards, you have far more to work with than the all-five figure suggests.

### `needs_review` and `ihs6_expansion_of`

Two columns describe how much to trust an IHS6 mapping:

- `ihs6_expansion_of` names the parent question when IHS6 split a "select all that apply" item into one binary column per option. IHS5's single `ag_e07` became `ag_e07__1`, `ag_e07__4` and so on. These are **not** new concepts, and there are 342 of them.
- `needs_review` marks a variable that is new in IHS6 and whose concept has not been cross-checked against an earlier instrument. Confirm what it measures before pooling it across rounds.

```
# Multi-select expansions, with their parent question
exp <- cw[!is.na(cw$ihs6_expansion_of),
          c("harmonised_name", "ihs6_expansion_of")]
head(exp, 5)
#> # A tibble: 5 x 2
#>   harmonised_name ihs6_expansion_of
#>   <chr>           <chr>
#> 1 ag_d53__0       ag_d53
#> 2 ag_d53__1       ag_d53
#> 3 ag_d53__2       ag_d53
#> 4 ag_d55__0       ag_d55
#> 5 ag_d55__1       ag_d55
```

```r
nrow(exp)
#> [1] 342
```

---

## 8.3 ihs_panel_ids() — look up design columns

```r
ihs_panel_ids(round = "IHS5")
```

```r
ihs_panel_ids("IHS6")
#> Standard ID columns for "IHS6":
#>     hh_id  indiv_id     ea_id    strata    weight
#> "case_id"     "PID"   "ea_id"  "strata"  "hh_wgt"
```

Use `"all"` for the cross-round comparison shown in §3.

---

## 8.4 ihs_harmonise() — rename to standard names

```r
ihs_harmonise(data, round = "IHS5", extra = FALSE)
```

The foundation of everything else. Renames raw columns to harmonised names using the crosswalk, preserving Stata variable labels, and adds an `ihs_round` column.

```r
raw <- data.frame(
  case_id = c("a", "b"),
  hhsize  = c(4, 6),
  junk    = c(1, 2)
)

ihs_harmonise(raw, round = "IHS6")
#> Harmonised 2 columns for IHS6.
#>   case_id hhsize ihs_round
#> 1       a      4      IHS6
#> 2       b      6      IHS6
```

By default, columns absent from the crosswalk are dropped. Keep everything with `extra = TRUE`:

```r
ihs_harmonise(raw, round = "IHS6", extra = TRUE)
#> Harmonised 2 columns for IHS6.
#>   case_id hhsize junk ihs_round
#> 1       a      4    1      IHS6
#> 2       b      6    2      IHS6
```

**Watch out:**

- Matching is case-insensitive, and each raw column is claimed by at most one crosswalk entry.
- A warning that *no* columns mapped almost always means the wrong `round`.
- `extra = FALSE` is destructive. If you need a variable that is not in the crosswalk, use `extra = TRUE` and open an issue so it can be added.

---

## 8.5 ihs_merge() — join modules

```r
ihs_merge(..., by = NULL, type = "left")
```

```r
hh <- data.frame(case_id = c("A", "B", "C"), hhsize = c(4, 6, 3))
ag <- data.frame(case_id = c("A", "B", "D"), harvest_kg = c(120, 340, 90))

merged <- ihs_merge(hh, ag)
```

```
#> Auto-detected join columns: "case_id"
#> Merged 2 data.frames: 3 rows, 3 columns.
merged
#>   case_id hhsize harvest_kg
#> 1       A      4        120
#> 2       B      6        340
#> 3       C      3         NA
```

`type` may be `"left"` (default), `"inner"` or `"full"`. Row counts at each step are recorded:

```
attr(merged, "ihs_merge_log")
#> [[1]]
#> [[1]]$step
#> [1] 0
#>
#> [[1]]$rows
#> [1] 3
#>
#>
#> [[2]]
#> [[2]]$step
#> [1] 1
#>
#> [[2]]$rows_before
#> [1] 3
#>
#> [[2]]$rows_after
#> [1] 3
#>
#> [[2]]$rows_right
#> [1] 3
```

**Watch out:** columns present in more than one input but not used as join keys get dplyr's `.x`/`.y` suffixes. `ihs_merge()` warns when this happens, because it silently breaks downstream auto-detection — once `hh_wgt` becomes `hh_wgt.x`, `ihs_svydesign()` can no longer find the survey weight:

```
a <- data.frame(case_id = c("A", "B"), region = 1:2, x = 1:2)
b <- data.frame(case_id = c("A", "B"), region = 1:2, y = 3:4)

names(suppressWarnings(ihs_merge(a, b)))
#> Auto-detected join columns: "case_id"
#> Merged 2 data.frames: 2 rows, 5 columns.
#> [1] "case_id"  "region.x" "x"        "region.y" "y"
```

Fix it by dropping the duplicate before merging, or by adding the shared column to `by`:

```
names(ihs_merge(a, b, by = c("case_id", "region")))
#> Merged 2 data.frames: 2 rows, 4 columns.
#> [1] "case_id" "region"  "x"       "y"
```

---

## 8.6 `ihs_standardize_missing()` — recode missing codes

```
ihs_standardize_missing(data)
```

Recodes `-99`, `-98`, `-97`, `999`, `998` and `997` to `NA` in numeric columns.

```
df <- data.frame(age = c(34, -99, 41, 998), village = c("A", "B", "C", "D"))
clean <- ihs_standardize_missing(df)
clean$age
#> [1] 34 NA 41 NA
attr(clean, "ihs_missing_conversions")
#> $age
```

```
#> [1] 2
```

**Watch out:** `999` is a legitimate value for some variables — a plot area in square metres, a price in kwacha. This is a blanket recode. Check your variables first, and if in doubt pass only the affected columns.

---

## 8.7 `ihs_winsorize()` — cap outliers

```
ihs_winsorize(data, vars, by = NULL, probs = c(0.01, 0.99))
```

Caps values below the lower and above the upper percentile, writing the result to a new `_w` column.

```
set.seed(42)
df <- data.frame(
  region = rep(c("North", "South"), each = 50),
  cons   = c(rnorm(50, 100, 10), rnorm(50, 500, 50))
)

global    <- ihs_winsorize(df, vars = "cons", probs = c(0.05, 0.95))
by_region <- ihs_winsorize(df, vars = "cons", by = "region",
                           probs = c(0.05, 0.95))

# The raw column is untouched either way
identical(global$cons, df$cons)
#> [1] TRUE
```

**Why stratify.** A single national 99th percentile treats the richest rural households as outliers, because the urban distribution sits far above them. Compare the maximum retained in the North under each approach:

```
c(global    = max(global$cons_w[global$region == "North"]),
  stratified = max(by_region$cons_w[by_region$region == "North"]))
#>     global stratified
#>   122.8665   117.2254
```

Stratifying preserves the shape of both distributions. This matters enormously for inequality and poverty measures.

---

## 8.8 `ihs_clean()` — the wrapper

```
ihs_clean(data, winsorize_vars = NULL, winsorize_by = NULL,
          probs = c(0.01, 0.99))
```

Runs `ihs_standardize_missing()` and then `ihs_winsorize()`, collecting both audit logs into a single `ihs_audit` attribute. Use it unless you need finer control.

```
result <- ihs_clean(df, winsorize_vars = "cons", winsorize_by = "region")
names(attr(result, "ihs_audit"))
#> [1] "initial_rows"        "initial_cols"        "missing_conversions"
#> [4] "winsorized_vars"     "final_rows"          "final_cols"
```

**Watch out:** the argument is `winsorize_vars`, not `winsorize_cols`, and the grouping argument is `winsorize_by`, not `strata_col`. There is no `missing_cols` argument — missing-code recoding always applies to every numeric column.

---

## 8.9 `ihs_convert_units()` — harvest units to kilograms

```
ihs_convert_units(data, qty_col, unit_col, crop_col, unmapped = "warn")
```

Converts reported harvest quantities to kilograms using 1,608 official NSO conversion factors, keyed on crop, unit, region and shelling condition.

```
harvest <- data.frame(
  crop_code = c(1, 1, 1),
  unit_code = c(1, 2, 3),
  quantity  = c(100, 2, 5),
  region    = c(1, 2, 2)
)

ihs_convert_units(harvest, qty_col = "quantity",
                  unit_col = "unit_code", crop_col = "crop_code")
#>   crop_code unit_code quantity region quantity_kg
#> 1         1         1      100      1         100
#> 2         1         2        2      2         100
#> 3         1         3        5      2         450
```

Two pails of maize is 100 kg; five of the next unit up is 450 kg. Summing the raw `quantity` column would have given 107 of nothing in particular.

**Matching rules,** applied in order:

1. exact match on crop, unit, region and condition;
2. same crop, unit and region, falling back through the condition preference order shelled (1) → not applicable (3) → unshelled (2);
3. the same two steps against the Central region, used as the national fallback when a crop-unit pair has no factor for the row's own region.

A `region` column is detected automatically. Without one, every row is priced at Central-region factors and the function says so — applying Central factors to Northern or Southern harvests biases every converted quantity.

There is one exception to the crop-specific rule. Unit code 1 is KILOGRAM, and a kilogram is a kilogram whatever the crop, so quantities already reported in kilograms convert even for crops the factor table does not list — including those introduced in IHS6:

```
kg <- data.frame(crop_code = c(1, 48), unit_code = c(1, 1), quantity = c(7, 40))
ihs_convert_units(kg, "quantity", "unit_code", "crop_code")
#> No region column found; using Central-region conversion factors for all rows.
#> i Add a `region` column to use region-specific factors.
#>   crop_code unit_code quantity quantity_kg
#> 1         1         1        7           7
#> 2        48         1       40          40
```

The package verifies this against the bundled table rather than assuming it: if a future NSO release gave the kilogram unit a crop-specific factor, the rule switches itself off.

**Unmappable combinations return `NA`,** never a guess, and the function names them:

```
odd <- data.frame(crop_code = 999, unit_code = 999, quantity = 10)
suppressWarnings(
  ihs_convert_units(odd, "quantity", "unit_code", "crop_code")
)
#> No region column found; using Central-region conversion factors for all rows.
#> i Add a `region` column to use region-specific factors.
#>   crop_code unit_code quantity quantity_kg
#> 1       999       999       10          NA
```

Set `unmapped = "error"` to make failures fatal in a production pipeline, or `"ignore"` to silence the warning once you have understood it.

**Known gaps in the factor table**

On IHS6 crop sales, about 10% of otherwise-convertible rows still return `NA`. The causes, in order of size:

- **Units with no published factor at all** — code 6 (No. 10 plate), 7 (No. 12 plate) and 10 (bale) are used in IHS6 but absent from the NSO table.
- **Crops introduced in IHS6** — codes 42, 48, 49 and 50 have no factors for non-kilogram units.
- **Unit 13, "OTHER (SPECIFY)"** — unmappable by design. The household named its own unit, so no general factor can exist. Roughly 113 rows in `ag_mod_i`.

Do not paper over these. `NA` is the correct answer, and a total computed over the rows that *did* convert is honest as long as you report the coverage. If you have an official NSO factor for any of these, please contribute it — the repository has an issue template for exactly this.

---

## 8.10 `ihs_aggregate()` — roll up to the household

```
ihs_aggregate(data, group_col = "case_id")
```

```
members <- data.frame(
  case_id   = c("A", "A", "B", "B", "B"),
  income    = c(1000, 500, 200, 300, 250),
  has_radio = c(0, 1, 0, 0, 0),
  district  = c("Lilongwe", "Lilongwe", "Blantyre", "Blantyre", "Blantyre")
)

ihs_aggregate(members, group_col = "case_id")
#> i Aggregating data by `case_id`...
#> # A tibble: 2 x 4
#>   case_id income has_radio district
#>   <chr>    <dbl>     <dbl> <chr>
#> 1 A         1500         1 Lilongwe
#> 2 B          750         0 Blantyre
```

**Rules.** Continuous numeric columns are summed; 0/1 columns become a logical OR; logical columns become `any()`; character and factor columns take their single distinct value, or the distinct values joined with `" | "`; anything else is dropped with a warning.

**Watch out:** summing is right for quantities and expenditures and wrong for ages, prices and rates. Select your columns before calling, or aggregate those yourself.

---

## 8.11 `ihs_deflate()` — nominal to real

```
ihs_deflate(data, value_cols, round = NULL, base_year = 2019)
```

Converts nominal kwacha to constant-price kwacha so figures from different rounds can be compared.

```
pooled <- data.frame(
  food_exp  = c(1000, 1000, 1000, 1000, 1000),
  ihs_round = c("IHS2", "IHS3", "IHS4", "IHS5", "IHS6")
)

ihs_deflate(pooled, value_cols = "food_exp")
#> Auto-detected round(s) from `ihs_round`: "IHS2", "IHS3", "IHS4", "IHS5", and
#> "IHS6"
#> ! CPI for 2024 are provisional and may be revised.
#> i Bundled series retrieved 2026-07-29. Rebuild with 'data-raw/02_build_cpi.R'
#>   to refresh.
#> Deflated `food_exp` across 5 round(s) to base year 2019.
#>   food_exp ihs_round food_exp_real
#> 1     1000      IHS2     7521.0590
```

```
#> 2     1000      IHS3     4183.4003
#> 3     1000      IHS4     1371.4787
#> 4     1000      IHS5     1000.0000
#> 5     1000      IHS6      408.9428
```

1,000 kwacha in 2004 bought what roughly 7,500 bought in 2019; 1,000 kwacha in 2024 bought about 409-worth. Comparing the nominal column across rounds would have reversed the story entirely.

Because `ihs_harmonise()` adds `ihs_round`, pooled data deflates in one call with no arguments. For a single-round frame, name the round:

```
ihs_deflate(data.frame(v = 1000), value_cols = "v", round = "IHS6")
#> ! CPI for 2024 are provisional and may be revised.
#> i Bundled series retrieved 2026-07-29. Rebuild with 'data-raw/02_build_cpi.R'
#>   to refresh.
#> `v`: deflation factor = 0.4089 (IHS6 -> 2019)
#>      v   v_real
#> 1 1000 408.9428
```

**The CPI series.** World Bank WDI `FP.CPI.TOTL` for Malawi, rebased to 2019 = 100, covering 2004–2025. Rounds map to their survey midpoint year: IHS2 → 2004, IHS3 → 2010, IHS4 → 2016, IHS5 → 2019, IHS6 → 2024.

The table records where each figure came from and whether it is still liable to revision:

```
cpi <- read.csv(system.file("extdata", "mw_cpi_annual.csv", package = "ihsMW"))
tail(cpi[, c("year", "cpi_index", "retrieved", "provisional")], 3)
#>    year cpi_index  retrieved provisional
#> 20 2023   185.001 2026-07-29       FALSE
#> 21 2024   244.533 2026-07-29        TRUE
#> 22 2025   313.908 2026-07-29        TRUE
```

The World Bank revises recent observations as national statistics are finalised, so the two most recent years are marked `provisional`. `ihs_deflate()` says so whenever a round you are deflating depends on one — as IHS6 (2024) currently does. These are the best figures available; just be aware they can move, and record which vintage you used. Refresh with `data-raw/02_build_cpi.R`.

**Watch out:** this is a **national** deflator over time. It does not adjust for price differences between districts or between urban and rural areas. If your analysis is sensitive to those, each round's consumption aggregate publishes its own spatial index — IHS6 ships `spatial_indexL` (regional price level) and `price_indexL` (combined spatial-temporal) — and you should divide by that rather than relying on this function alone.

---

## 8.12 `ihs_svydesign()` — build the survey design

```
ihs_svydesign(data, weight_col = NULL, strata_col = NULL, psu_col = NULL)
```

Wraps `survey::svydesign()`, detecting the weight, stratum and PSU columns.

```
hh <- data.frame(
  case_id  = 1:20,
  ea_id    = rep(1:5, each = 4),
  strata   = rep(c("urban", "rural"), each = 10),
  hh_wgt   = runif(20, 0.5, 3),
  food_exp = rnorm(20, 5000, 1000)
)

dsgn <- ihs_svydesign(hh)
#> Auto-detected weight column: `hh_wgt`
#> Auto-detected strata column: `strata`
#> Auto-detected PSU column: `ea_id`
#> Creating survey design:
```

```
#> * Weights: `hh_wgt`
#> * Strata: `strata`
#> * PSU: `ea_id`
survey::svymean(~food_exp, dsgn, na.rm = TRUE)
#>              mean     SE
#> food_exp 4570 166.78
```

The same call works on every round: case-insensitive matching handles IHS2/IHS3 `hhweight`, IHS4+ `hh_wgt`, IHS2–IHS5 `stratum` and IHS6 `strata`.

If strata or PSU are missing the function degrades gracefully — to a clustered design, then to a simple weighted one — and tells you which it built. Read that message: a simple weighted design gives correct point estimates but standard errors that are too small.

---

### 8.13 `ihs_report()` — summary table

```
ihs_report(data, vars = NULL, by = NULL, weights = NULL)
```

```
hh <- data.frame(
  hhsize   = c(4, 6, 3, 5, 7, 2),
  food_exp = c(1200, 3400, 900, 2100, 4500, 700),
  region   = c("North", "North", "Central", "Central", "South", "South"),
  hh_wgt   = c(1.2, 0.8, 1.5, 1.1, 0.9, 1.3)
)

ihs_report(hh, vars = c("hhsize", "food_exp"))
#> Summary statistics for 2 variables.
#>   variable n     mean        sd median min  max pct_missing
#> 1   hhsize 6    4.500    1.8708    4.5   2    7           0
#> 2 food_exp 6 2133.333 1526.6521 1650.0 700 4500           0
```

Group and weight it:

```
ihs_report(hh, vars = "food_exp", by = "region", weights = "hh_wgt")
#> Summary statistics for 1 variable across 3 groups.
#>    region variable n     mean        sd median  min  max pct_missing
#> 1   North food_exp 2 2080.000 1077.7755   2300 1200 3400           0
#> 2 Central food_exp 2 1407.692  592.8569   1500  900 2100           0
#> 3   South food_exp 2 2254.546 1868.3311   2600  700 4500           0
```

With `vars = NULL` every numeric column is summarised, with weight-like columns excluded automatically.

**Watch out:** only the mean and SD are weighted. `n`, `median`, `min`, `max` and `pct_missing` describe the sample, not the population. For weighted quantiles or design-correct standard errors, use `ihs_svydesign()` and the `survey` package.

---

## 9. Complete worked workflows

These use real file names. Substitute your own download paths.

### Workflow A — one round, weighted poverty statistics

```
library(ihsMW)
library(haven)
library(survey)

# 1. Read
demog <- haven::read_dta("IHS6/hh_mod_a_filt.dta")
```

```r
cons  <- haven::read_dta("IHS6/ihs6_consumptionaggregates.dta")

# 2. Harmonise (adds `ihs_round`)
demog_h <- ihs_harmonise(demog, round = "IHS6")
cons_h  <- ihs_harmonise(cons,  round = "IHS6")

# 3. Merge, dropping columns that appear in both to avoid .x/.y suffixes
dupes  <- c("region", "district", "hhsize", "hh_wgt", "ihs_round")
merged <- ihs_merge(demog_h, cons_h[, setdiff(names(cons_h), dupes)])

# 4. Clean: recode missing codes, winsorize within region
clean <- ihs_clean(merged,
                   winsorize_vars = "rexp_cat011",
                   winsorize_by   = "region")

# 5. Deflate to 2019 prices
real <- ihs_deflate(clean, value_cols = "rexp_cat011_w")

# 6. Survey design
dsgn <- ihs_svydesign(real)

# 7. Report
ihs_report(real,
           vars    = c("hhsize", "rexp_cat011_w_real"),
           by      = "region",
           weights = "hh_wgt")

# Design-correct national mean with a standard error
survey::svymean(~rexp_cat011_w_real, dsgn, na.rm = TRUE)

# Document what cleaning did, for your appendix
str(attr(clean, "ihs_audit"))
```

## Workflow B — pooling IHS4, IHS5 and IHS6

```r
library(ihsMW)
library(haven)
library(dplyr)

read_round <- function(path, round) {
  haven::read_dta(path) |> ihs_harmonise(round = round)
}

ihs4 <- read_round("IHS4/ihs4_consumption_aggregate.dta", "IHS4")
ihs5 <- read_round("IHS5/ihs5_consumption_aggregate.dta", "IHS5")
ihs6 <- read_round("IHS6/ihs6_consumptionaggregates.dta", "IHS6")

# Check comparability BEFORE pooling
cw <- ihs_crosswalk_check(verbose = FALSE)
comparable <- cw$harmonised_name[
  !is.na(cw$ihs4_name) & !is.na(cw$ihs5_name) & !is.na(cw$ihs6_name)
]

keep <- intersect(comparable, Reduce(intersect, list(names(ihs4), names(ihs5), names(ihs6))))
keep <- union(keep, "ihs_round")

pooled <- bind_rows(ihs4[, keep], ihs5[, keep], ihs6[, keep])

# One call deflates every round correctly, using `ihs_round`
pooled <- ihs_deflate(pooled, value_cols = "rexp_cat011")

# Real food consumption by round, on a common 2019 basis
ihs_report(pooled, vars = "rexp_cat011_real", by = "ihs_round",
```

```
            weights = "hh_wgt")
```

The `comparable` step is the important one. Pooling variables that do not actually exist in all three rounds produces a series with silent structural breaks.

### Workflow C — agricultural production per household

```
library(ihsMW)
library(haven)

# Plot-level harvest, and the household roster for region
harvest <- haven::read_dta("IHS6/ag_mod_g.dta")
roster  <- haven::read_dta("IHS6/hh_mod_a_filt.dta")

# Attach region so region-specific conversion factors are used
harvest <- ihs_merge(harvest, roster[, c("case_id", "region")], by = "case_id")

# Convert mixed units to kilograms
harvest_kg <- ihs_convert_units(
  harvest,
  qty_col  = "ag_g13a",
  unit_col = "ag_g13b",
  crop_col = "crop_code",
  unmapped = "warn"
)

# Sum to household level
hh_production <- ihs_aggregate(
  harvest_kg[, c("case_id", "ag_g13a_kg")],
  group_col = "case_id"
)

head(hh_production)
```

Note the merge in step two: without a `region` column, every row is converted at Central-region factors.

---

## 10. Common mistakes

**1. Using the wrong argument names.** The three that trip people up most:

```
# WRONG
ihs_clean(df, winsorize_cols = "x", strata_col = "region")
ihs_winsorize(df, value_col = "x", strata_col = "region")
ihs_aggregate(df, id_cols = "case_id", val_cols = "x")

# RIGHT
ihs_clean(df, winsorize_vars = "x", winsorize_by = "region")
ihs_winsorize(df, vars = "x", by = "region")
ihs_aggregate(df, group_col = "case_id")
```

**2. Merging before harmonising.** Join keys are detected by name, and the names only line up after `ihs_harmonise()`.

**3. Analysing the raw column after winsorizing.** `ihs_winsorize()` writes to `x_w` and leaves `x` alone. If your results look unchanged, you are probably still using `x`.

**4. Subsetting before building the survey design.** This breaks the stratum and cluster structure and gives wrong standard errors:

```
# WRONG
rural <- data[data$urban == 0, ]
dsgn  <- ihs_svydesign(rural)
```

```r
# RIGHT
dsgn  <- ihs_svydesign(data)
rural <- subset(dsgn, urban == 0)
```

**5. Comparing nominal kwacha across rounds.** Always `ihs_deflate()` first. Malawi's price level rose roughly 24-fold between 2004 and 2025; nominal comparisons across rounds are not merely imprecise, they are usually backwards.

**6. Assuming every variable exists in every round.** Check `ihs_crosswalk_check()` first. Only a small core spans all five rounds.

**7. Ignoring the `.x/.y` warning from `ihs_merge()`.** It is telling you that a column you rely on — often the survey weight — has been renamed.

**8. Summing quantities before converting units.** A column mixing pails, ox-carts and 50 kg bags cannot be summed. Convert first.

**9. Using CSV files for unit conversion.** Some IHS CSV exports contain value labels instead of numeric codes. Use the `.dta` distribution.

**10. Treating `-99` as a real number.** Run `ihs_standardize_missing()` — or `ihs_clean()`, which includes it — before any arithmetic.

---

# 11. Troubleshooting

| Message | Cause | Fix |
|---|---|---|
| `Invalid round(s) specified: "IHS7"` | Unsupported round | Use IHS2–IHS6. IHS1 is not available. |
| `No columns were mapped to harmonised names` | Wrong `round`, or a file whose variables are not in the crosswalk | Check the `round` argument first; then `ihs_search()` for one of your column names |
| `No common ID columns detected across all data.frames` | Inputs were not harmonised, or genuinely share no key | Harmonise first, or pass `by` explicitly |
| `N columns appeared in more than one input and were suffixed` | Shared non-key columns got `.x/.y` | Drop the duplicates, or add them to `by` |
| `Merge step 1 expanded rows from X to Y` | Many-to-many join | Aggregate one side to household level first, or add a finer key |
| `Column 'crop_code' holds value labels, not numeric codes` | CSV export substituted labels for codes | Read the `.dta` version with `haven::read_dta()` |
| `Failed to map N crop-unit combinations` | No published NSO factor for that pair | Expected for new IHS6 codes; those rows get `NA`. Report the combination as an issue |
| `Could not detect a survey weight column` | Weight renamed by a merge, or absent | Check for `hh_wgt.x`; pass `weight_col` explicitly |
| `Grouping variable 'region' not found in data` | Column dropped by `ihs_harmonise(extra = FALSE)` or suffixed by a merge | Check `names(data)` |
| `Base year 2019 not found in CPI table` | `base_year` outside 2004–2025 | Choose a year within range |
| `Cannot determine IHS round` | No `round` argument and no `ihs_round` column | Pass `round`, or harmonise first |

### Diagnosing anything else

```
# What do I actually have?
names(data)
str(data[, 1:10])

# Did harmonisation do anything?
sum(names(data) %in% ihs_crosswalk_check(verbose = FALSE)$harmonised_name)

# Where did my rows go?
attr(merged, "ihs_merge_log")

# What did cleaning change?
str(attr(cleaned, "ihs_audit"))
```

---

## 12. Frequently asked questions

**Does the package include the survey data?** No. World Bank licensing prohibits redistribution. Download it yourself (§5). Everything else — crosswalk, conversion factors, CPI — is bundled.

**Does it need an internet connection?** No. `ihsMW` never makes a network request after installation.

**Which rounds are supported?** IHS2 (2004/05), IHS3 (2010/11), IHS4 (2016/17), IHS5 (2019/20) and IHS6 (2024/25). IHS1 (1997/98) is not available in a usable form.

**Can I compare a variable across all five rounds?** Only if it exists in all five, and only 49 do. Check with `ihs_crosswalk_check()` before designing your analysis. Coverage from IHS3 onwards is far better — see §8.2 for why, and for the numbers.

**Why did `ihs_convert_units()` return `NA` for a quantity in kilograms?** It should not, and no longer does: kilogram quantities convert even for crops the factor table does not list. If you still see this, you are on a version before 1.1.0.

**What changed in IHS6?** Three things matter. The design stratum was renamed `stratum` → `strata` (handled automatically). `adulteq` (adult equivalents) is absent from the consumption aggregate, so per-adult-equivalent measures must be constructed yourself. And consumption aggregates are published in July 2024 prices — use `ihs_deflate()` to put every round on a common basis.

**A variable I need is missing from the crosswalk.** Use `ihs_harmonise(extra = TRUE)` to keep it, then open a crosswalk mapping issue so it is available to everyone next release.

**Why did `ihs_convert_units()` return all `NA`?** Either your crop/unit columns hold value labels rather than numeric codes (the function now errors rather than returning `NA` for this), or the crop-unit combinations have no published factor. The warning names them.

**Should I winsorize at 1% or 5%?** There is no universal answer. `c(0.01, 0.99)` is the common default for consumption. Report whichever you used, and check that your conclusions survive the alternative.

**Should I winsorize before or after deflating?** Before. Winsorizing bounds the nominal distribution within its own round; deflating afterwards is a monotone rescaling that preserves those bounds.

**Can I use `ihsMW` with `srvyr`?** Yes. `ihs_svydesign()` returns a standard `survey.design2`, so `srvyr::as_survey(dsgn)` works directly.

**Is the CPI series the official NSO one?** It is the World Bank WDI series `FP.CPI.TOTL` for Malawi, rebased to 2019 = 100. WDI is compiled from NSO returns but the two can differ slightly in revision timing. If your work requires the NSO series exactly, substitute it — the file is a plain CSV at `system.file("extdata", "mw_cpi_annual.csv", package = "ihsMW")`.

**How do I cite this?** See §14.

---

## 12a. Known limitations

Stated plainly, so you can judge whether they affect your work. None of these are hidden by the package — each one surfaces as a message, an `NA`, or a column you can inspect.

| Limitation | What it means for you | Status |
|---|---|---|
| **Thin full-series coverage.** Only 49 variables span all five rounds | IHS2 is a genuinely smaller instrument, but the crosswalk also under-links it because it matches mostly on variable *names* and IHS2 used a different scheme | Improvable — label-based matching would recover more. Contributions welcome |
| **~270 IHS6 variables flagged `needs_review`** | New in IHS6 and not cross-checked against an earlier instrument. Confirm the concept before pooling across rounds | Inspect via the `needs_review` column |
| **~10% of IHS6 crop sales rows do not convert** | Units 6, 7 and 10 and crops 42, 48–50 have no published NSO factor; unit 13 is "OTHER (SPECIFY)" and never will | Partly fixable with official NSO data |
| **No spatial price adjustment** | `ihs_deflate()` is a national deflator over time only | Use the round's own `spatial_indexL` if you need it |
| **The two most recent CPI years are provisional** | Real values for IHS6 can shift when the World Bank revises 2024 | `ihs_deflate()` tells you; refresh with `data-raw/02_build_cpi.R` |
| **No cross-round household panel linking** | `ihs_panel_ids()` gives you the ID columns, but the linking and attrition logic is yours | Out of scope by design |

The general principle: where `ihsMW` cannot give you a defensible answer it gives you `NA` and says why. It never invents a number to fill a gap.

---

## 13. Best practices

**Script everything; click nothing.** The whole point of `ihsMW` is that your cleaning decisions live in version-controlled code rather than in a colleague's memory.

**Keep the audit trail.** Save `attr(x, "ihs_audit")` alongside your results. It answers the outlier and missing-data questions reviewers always ask.

**Check comparability before pooling.** Run `ihs_crosswalk_check()` and filter to variables that genuinely exist in every round you use.

**Deflate before comparing money across rounds.** Every time.

**Winsorize within strata.** A national threshold systematically over-trims the top of poorer strata. Pass `by`.

**Build the design once, on the full data.** Subset the design object, never the data frame.

**Read the console messages.** `ihsMW` is deliberately chatty about what it auto-detected. The messages tell you which weight column it found and which join keys it used — the two things most likely to be silently wrong.

**Report your parameters.** Winsorization percentiles, base year, which rounds and which variables. A reader should be able to reproduce your table from your methods section.

**Pin your versions.** Record `packageVersion("ihsMW")` in your output, or use `renv`. The crosswalk grows between releases.

**Prefer `.dta` over `.csv`.** Numeric codes and preserved variable labels.

# 14. Citation

To cite the package:

```
citation("ihsMW")
```

```
@Manual{ihsMW,
  title  = {ihsMW: Clean and Harmonise Malawi Integrated Household Survey Data},
  author = {Vitumbiko Kayuni},
  year   = {2026},
  note   = {R package version 1.1.1},
  url    = {https://github.com/vituk123/ihsMW},
}
```

**Also cite the data.** `ihsMW` is a tool, not a source. Any publication using IHS microdata must cite the National Statistical Office of Malawi and the World Bank LSMS programme. Each round's Basic Information Document gives the exact required form — use it, because the requirement is a condition of your data access agreement.

**Cite the reference data if you rely on it.** The crop conversion factors are NSO's; the CPI series is World Bank WDI `FP.CPI.TOTL`.

# 15. Getting help and contributing

- **Documentation:** https://vituk123.github.io/ihsMW/
- **Bug reports:** https://github.com/vituk123/ihsMW/issues
- **Crosswalk mappings and conversion factors:** the repository has dedicated issue templates. A mapping you add helps everyone working on the IHS.

When reporting a bug, include the output of `sessionInfo()`, the version from `packageVersion("ihsMW")`, and a minimal reproducible example using synthetic data rather than microdata you are not licensed to share.

```
#> Generated with ihsMW 1.1.1 on 29 July 2026
```