

# Package ‘ipumsr’

March 8, 2019

**Title** Read 'IPUMS' Extract Files

**Version** 0.4.0

**Contact** ipums@umn.edu

**URL** <https://www.ipums.org>, <https://github.com/mnpopcenter/ipumsr>

**BugReports** <https://github.com/mnpopcenter/ipumsr/issues>

**Description** An easy way to import census, survey and geographic data provided by 'IPUMS' into R plus tools to help use the associated metadata to make analysis easier. 'IPUMS' data describing 1.4 billion individuals drawn from over 750 censuses and surveys is available free of charge from our website <<https://ipums.org>>.

**License** Mozilla Public License 2.0

**Encoding** UTF-8

**LazyData** true

**Imports** cli, crayon, dplyr (>= 0.7.0), haven (>= 2.0.0), hipread (>= 0.2.0), pillar, purrr, R6, raster, readr, rlang, stringr, tibble, tidyr, xml2, Rcpp, zeallot

**RoxygenNote** 6.1.1

**Suggests** DT, ggplot2, htmltools, knitr, rgdal, rmarkdown, rstudioapi, scales, sf, sp, shiny, testthat, covr, biglm, DBI, RSQLite, dbplyr

**VignetteBuilder** knitr

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Author** Greg Freedman Ellis [aut, cre],  
Derek Burk [ctb],  
Joe Grover [ctb],  
Minnesota Population Center [cph]

**Maintainer** Greg Freedman Ellis <gfellis@umn.edu>

**Repository** CRAN

**Date/Publication** 2019-03-08 22:30:03 UTC

**R topics documented:**

dplyr_select_style . . . . .	2
ipums_bind_rows . . . . .	3
ipums_collect . . . . .	4
ipums_conditions . . . . .	4
ipums_example . . . . .	5
ipums_file_info . . . . .	5
ipums_list_files . . . . .	6
ipums_shape_left_join . . . . .	7
ipums_var_info . . . . .	8
ipums_view . . . . .	9
ipums_website . . . . .	10
join_failures . . . . .	11
lbl . . . . .	12
lbl_add . . . . .	12
lbl_clean . . . . .	13
lbl_collapse . . . . .	14
lbl_na_if . . . . .	15
lbl_relabel . . . . .	16
read_ipums_codebook . . . . .	17
read_ipums_ddi . . . . .	18
read_ipums_micro . . . . .	19
read_ipums_micro_chunked . . . . .	20
read_ipums_micro_yield . . . . .	23
read_ipums_sf . . . . .	25
read_nhgis . . . . .	26
read_terra_area . . . . .	27
read_terra_micro . . . . .	29
read_terra_raster . . . . .	30
set_ipums_var_attributes . . . . .	31
zap_ipums_attributes . . . . .	32
<b>Index</b>	<b>33</b>

---

dplyr\_select\_style      *Select-style helpers from dplyr*

---

**Description**

Several arguments in `ipumsr` allow syntax for selecting variables based on `dplyr`'s `select` function. See details for more information.

## Details

There are 3 broad categories of methods for specifying arguments for these select-style parameters.

- "Character Vector" A character vector of names (such as `c("var1", "var2", "var3")`)
- "'Bare' Vector" A vector of 'bare' names (such as `c(var1, var2, var3)`)
- "Helper Functions" Helper functions from `dplyr::select` such as `starts_with()`, `contains` and others.

## Examples

```
# For microdata, use this syntax to load variables
# Load 3 variables by name
cps_file <- ipums_example("cps_00006.xml")
data <- read_ipums_micro(cps_file, vars = c("YEAR", "MONTH", "PERNUM"))

# Load same 3 variables using bare names
data <- read_ipums_micro(cps_file, vars = c(YEAR, MONTH, PERNUM))

# Use helper functions to load all variables that start with "WT"
data <- read_ipums_micro(cps_file, vars = starts_with("WT"))

# Use bare names and helper function to load YEAR, MONTH and all variables with 'INC' in name
data <- read_ipums_micro(cps_file, vars = c(YEAR, MONTH, contains("INC")))

# For geographic extracts, `data_layer` and `shape_layer` arguments use the same conventions
# to select file names from within zip files.
# (This extract only contains 1 type of file, but some have multiple)
csv_file <- ipums_example("nhgis0008_csv.zip")
data <- read_nhgis(
  csv_file,
  data_layer = contains("pmsa")
)
```

---

ipums\_bind\_rows

*Bind rows together, but preserve labelled class attributes*

---

## Description

Bind rows together, but preserve labelled class attributes

## Usage

```
ipums_bind_rows(..., .id = NULL)
```

## Arguments

<code>...</code>	Either data.frames or list of data.frames
<code>.id</code>	Data frame identifier, when arguments are named (or are named lists of data.frames), will make a new column with this name that has the original names.

**Value**

A data.frame

---

ipums_collect	<i>Collect data into R session with IPUMS attributes</i>
---------------	--

---

**Description**

Convenience wrapper around dplyr [collect](#) and [set\\_ipums\\_var\\_attributes](#).

**Usage**

```
ipums_collect(data, ddi, var_attrs = c("val_labels", "var_label",
  "var_desc"))
```

**Arguments**

data	A dplyr tbl object (generally a tbl_lazy object stored in a database.
ddi	A DDI object, read with <a href="#">read_ipums_ddi</a> .
var_attrs	One or more of val_labels, var_label and var_desc describing what kinds of attributes you want to add. If NULL, will not add any attributes.

**Value**

A local tbl\_df data.frame with IPUMS attributes attached

---

ipums_conditions	<i>Get IPUMS citation and conditions</i>
------------------	--

---

**Description**

Gets information about citation and conditions from a DDI.

**Usage**

```
ipums_conditions(object = NULL)
```

**Arguments**

object	A DDI object (loaded with <a href="#">read_ipums_ddi</a> ). If NULL (the default), will use the conditions from the dataset you loaded most recently.
--------	---

---

ipums_example	<i>Get path to ipums example datasets</i>
---------------	---

---

**Description**

Get access to example extracts.

**Usage**

```
ipums_example(path = NULL)
```

**Arguments**

path                    Name of file. If 'NULL', the example files will be listed.

**Value**

The filepath to an example file, or if path is empty, a vector of all available files.

**Examples**

```
ipums_example() # Lists all available examples
ipums_example("cps_00006.xml") # Gives filepath for a cps DDI
```

---

ipums_file_info	<i>Get IPUMS file information</i>
-----------------	-----------------------------------

---

**Description**

Get IPUMS metadata information about the data file loaded into R from an ipums\_ddi

**Usage**

```
ipums_file_info(object, type = NULL)
```

**Arguments**

object                  An ipums\_ddi object (loaded with [read\\_ipums\\_ddi](#)).

type                    NULL to load all types, or one of "ipums\_project", "extract\_data", "extract\_notes", "conditions" or "citation".

**Value**

If type is NULL, a list with the ipums\_project, extract\_date, extract\_notes, conditions, and citation. Otherwise a string with the type of information requested in type.

**Examples**

```
ddi <- read_ipums_ddi(ipums_example("cps_00006.xml"))
ipums_file_info(ddi)
```

---

ipums_list_files	<i>List files available for analysis in an IPUMS extract</i>
------------------	--

---

**Description**

Find which files can be loaded from an IPUMS extract. On Windows, this is generally a zip file (which you can optionally unzip). On macOS, they are generally unzipped for you, so there will be a directory.

**Usage**

```
ipums_list_files(file, types = NULL, data_layer = NULL,
  shape_layer = NULL, raster_layer = NULL)

ipums_list_data(file, data_layer = NULL)

ipums_list_shape(file, shape_layer = NULL)

ipums_list_raster(file, raster_layer = NULL)
```

**Arguments**

file	An IPUMS extract zip file or directory
types	One or more of "data", "shape", or "raster" indicating what type of files to look for.
data_layer	dplyr <a href="#">select</a> -style notation for the data files to look for
shape_layer	dplyr <a href="#">select</a> -style notation for the shape files to look for
raster_layer	dplyr <a href="#">select</a> -style notation for the raster files to look for

**Value**

A `tbl_df` data.frame containing the files available

**Examples**

```
nhgis_file <- ipums_example("nhgis0008_csv.zip")
ipums_list_files(nhgis_file) # Only one extract available
```

---

ipums\_shape\_left\_join *Join data to geographic boundaries*

---

## Description

Helpers for joining shape files downloaded from the IPUMS website to data from extracts. Because of historical reasons, the attributes of (like variable type) of variables in the shape files does not always match those in the data files.

## Usage

```
ipums_shape_left_join(data, shape_data, by, suffix = c("", "SHAPE"),
  verbose = TRUE)
```

```
ipums_shape_right_join(data, shape_data, by, suffix = c("", "SHAPE"),
  verbose = TRUE)
```

```
ipums_shape_inner_join(data, shape_data, by, suffix = c("", "SHAPE"),
  verbose = TRUE)
```

```
ipums_shape_full_join(data, shape_data, by, suffix = c("", "SHAPE"),
  verbose = TRUE)
```

## Arguments

<code>data</code>	A dataset, usually one that has been aggregated to a geographic level.
<code>shape_data</code>	A shape file (loaded with <code>read_ipums_sf</code> or <code>read_ipums_sp</code> )
<code>by</code>	A vector of variable names to join on. Like the dplyr join functions, named vectors indicate that the names are different between the data and shape file. shape files to load. Accepts a character vector specifying the file name, or <code>dplyr_select_style</code> conventions. Can load multiple shape files, which will be combined.
<code>suffix</code>	For variables that are found in both, but aren't joined on, a suffix to put on the variables. Defaults to nothing for data variables and "_SHAPE" for variables from the shape file.
<code>verbose</code>	If TRUE, will report information about geometries dropped in the merge.

## Value

returns a `sf` or a `SpatialPolygonsDataFrame` depending on what was passed in.

## Examples

```
# Note that these examples use NHGIS data so that they use the example data provided,
# but the functions read_nhgis_sf/read_nhgis_sp perform this merge for you.
```

```

data <- read_nhgis(ipums_example("nhgis0008_csv.zip"))

if (require(sf)) {
  sf <- read_ipums_sf(ipums_example("nhgis0008_shape_small.zip"))
  data_sf <- ipums_shape_inner_join(data, sf, by = "GISJOIN")
}

if (require(sp) && require(rgdal)) {
  sp <- read_ipums_sp(ipums_example("nhgis0008_shape_small.zip"))
  data_sp <- ipums_shape_inner_join(data, sp, by = "GISJOIN")
}

## Not run:
# Sometimes variable names won't match between datasets (for example in IPUMS international)
data <- read_ipums_micro("ipumsi_00004.xml")
shape <- read_ipums_sf("geo2_br1980_2010.zip")
data_sf <- ipums_shape_inner_join(data, shape, by = c("GEO2" = "GEOLEVEL2"))

## End(Not run)

```

---

ipums\_var\_info

*Get IPUMS variable information*


---

## Description

Get IPUMS metadata information about variables loaded into R. Will try to read the metadata from the loaded datasets, but it is more reliable to load the DDI into a separate object and use it instead.

## Usage

```

ipums_var_info(object, vars = NULL)

ipums_var_desc(object, var = NULL)

ipums_var_label(object, var = NULL)

ipums_val_labels(object, var = NULL)

```

## Arguments

object	A DDI object (loaded with <a href="#">read_ipums_ddi</a> ), a data.frame with ipums metadata attached, or a single column from an ipums data.frame.
vars	dplyr <a href="#">select</a> -style notation for the variables to give information about
var	select-style notation for a single variable

ipums\_var\_info() loads all available variable information for one or more variables into a data.frame. If object is a vector, it will include the variable label, variable description and value labels. If object is a data.frame, it will



include it for all variables (or only those specified by vars). If it is a DDI, it will also include information used to read the data from disk, including start/end position in the fixed-width file, implied decimals and variable type.

`ipums_var_desc()` loads the variable description for a single variable.

`ipums_var_label()` loads the short variable label for a single variable.

`ipums_val_labels()` loads the value labels for a single variable.

Note that many R functions drop attributes that provide this information. In order to make sure that they are available, it is best to keep a copy of the separate from the data you are manipulating using `read_ipums_ddi`. Then you can refer to the IPUMS documentation in this object.

### Value

`ipums_var_info` returns a `tbl_df` data frame with variable information, and the other functions return a length 1 character vector.

### Examples

```
ddi <- read_ipums_ddi(ipums_example("cps_00006.xml"))

ipums_var_info(ddi)
ipums_var_desc(ddi, MONTH)
ipums_var_label(ddi, MONTH)
ipums_val_labels(ddi, MONTH)
```

---

ipums\_view

*View a static webpage with variable information from a IPUMS extract*

---

### Description

Requires that `htmltools`, `shiny` and `DT` are installed.

### Usage

```
ipums_view(x, out_file = NULL, launch = TRUE)
```

### Arguments

<code>x</code>	A DDI or other object with <code>ipums</code> attributes (such as data loaded from an extract). Note that the file level information (like extract notes) are only available from the DDI.
<code>out_file</code>	Optionally specify a location to save HTML file. <code>NULL</code> the default makes a temporary file.
<code>launch</code>	Logical indicating whether to launch the website.

**Value**

The filepath to the html (silently if launch is TRUE)

**Examples**

```
ddi <- read_ipums_ddi(ipums_example("cps_00006.xml"))
## Not run:
ipums_view(ddi)
ipums_view(ddi, "codebook.html", launch = FALSE)

## End(Not run)
```

---

ipums\_website

*Launch a browser window to the ipums website*


---

**Description**

Takes a DDI (or you can specify a project directly) and a variable name, and makes a best guess at the URL for the variable's page on the IPUMS website. Note that NHGIS and TerraPop do not have accessible pages for variables.

**Usage**

```
ipums_website(x, var, project = NULL, launch = TRUE, verbose = TRUE,
  var_label = NULL, homepage_if_missing = TRUE)
```

**Arguments**

x	A DDI or empty (if specifying project)
var	A single variable name in a character vector
project	If not using a DDI (or object with a project attribute) A name of an IPUMS project, one of: "IPUMS-USA", "IPUMS-CPS", "IPUMS-International", "IPUMS-DHS", "ATUS-X", "AHTUS-X", "MTUS-X", "NHIS", "Higher Ed", "NHGIS", or "IPUMS Terra"
launch	If TRUE, launch the website.
verbose	If TRUE, message user if no variable specific websites are available
var_label	Sometimes the variable label is useful for finding the correct URL. Only needed if not passing in the ddi object.
homepage_if_missing	If TRUE, Return homepage if project does not provide variable specific web pages.

**Details**

Because some variables are constructed during the extract creation process, the URL may not always work unfortunately.

**Value**

The url to the page on ipums.org (silently if launch is TRUE)

**Examples**

```
ddi <- read_ipums_ddi(ipums_example("cps_00006.xml"))
ipums_website(ddi, "MONTH", launch = FALSE)

## Not run:
# Launches website
ipums_website(ddi, "MONTH")

## End(Not run)

# Can also specify project instead of using DDI
ipums_website(var = "RECTYPE", project = "IPUMS-CPS", launch = FALSE)
```

---

join\_failures

*Report on observations dropped by a join*

---

**Description**

Helper for learning which observations were dropped from a dataset because they were not joined on.

**Usage**

```
join_failures(join_results)
```

**Arguments**

join\_results    A dataset that has just been created by a shape join (like [ipums\\_shape\\_left\\_join](#))

**Value**

returns a list of data.frames, where the first item (shape) is the observations dropped from the shape file and the second (data) is the observations dropped from the data.

---

lbl *Make a label placeholder object*

---

**Description**

Helper to make a placeholder for a label-value pair.

**Usage**

```
lbl(...)
```

**Arguments**

... Either one or two arguments, possibly named `.val` and `.lbl`. If a single unnamed value, represents the label, if 2 unnamed values, the first is the value and the second is the label.

**Value**

A `label_placeholder` object, useful in functions like [lbl\\_add](#)

**See Also**

Other `lbl_helpers`: [lbl\\_add](#), [lbl\\_clean](#), [lbl\\_collapse](#), [lbl\\_na\\_if](#), [lbl\\_relabel](#), [zap\\_ipums\\_attributes](#)

**Examples**

```
x <- haven::labelled(
  c(100, 200, 105, 990, 999, 230),
  c(`Unknown` = 990, NIU = 999)
)

lbl_add(x, lbl(100, "$100"), lbl(105, "$105"), lbl(200, "$200"), lbl(230, "$230"))
```

---

lbl\_add *Add labels for unlabelled values*

---

**Description**

Add labels for values that don't already have them.

**Usage**

```
lbl_add(x, ...)
```

```
lbl_add_vals(x, labeller = as.character, vals = NULL)
```

**Arguments**

x	A <a href="#">labelled</a> vector
...	Labels formed by <a href="#">lbl</a> indicating the value and label to be added.
labeller	A function that takes a single argument of the values and returns the labels. Defaults to <code>as.character</code> . <a href="#">as_function</a> , so also accepts quosure-style lambda functions. See examples for more details.
vals	Vector of values to be labelled. NULL, the default labels all values that are in the data, but aren't already labelled.

**Value**

A `haven::labeled` vector

**See Also**

Other `lbl_helpers`: [lbl\\_clean](#), [lbl\\_collapse](#), [lbl\\_na\\_if](#), [lbl\\_relabel](#), [lbl](#), [zap\\_ipums\\_attributes](#)

**Examples**

```
x <- haven::labelled(
  c(100, 200, 105, 990, 999, 230),
  c(`Unknown` = 990, NIU = 999)
)

lbl_add(x, lbl(100, "$100"), lbl(105, "$105"), lbl(200, "$200"), lbl(230, "$230"))

lbl_add_vals(x)
lbl_add_vals(x, ~paste0("$", .))
lbl_add_vals(x, vals = c(100, 200))
```

---

lbl\_clean

*Clean unused labels*

---

**Description**

Remove labels that do not appear in the data.

**Usage**

```
lbl_clean(x)
```

**Arguments**

x	A <a href="#">labelled</a> vector
---	-----------------------------------

**Value**

A haven::labeled vector

**See Also**

Other lbl\_helpers: [lbl\\_add](#), [lbl\\_collapse](#), [lbl\\_na\\_if](#), [lbl\\_relabel](#), [lbl](#), [zap\\_ipums\\_attributes](#)

**Examples**

```
x <- haven::labelled(
  c(1, 2, 3, 1, 2, 3, 1, 2, 3),
  c(Q1 = 1, Q2 = 2, Q3 = 3, Q4= 4)
)

lbl_clean(x)
```

---

lbl\_collapse

*Collapse labelled values to labels that already exist*

---

**Description**

Converts values to a new value based on their label and value in a [labelled](#) vector. If the newly assigned value does not match an already existing labelled value, the smallest value's label is used. Ignores any value that does not have a label.

**Usage**

```
lbl_collapse(x, .fun)
```

**Arguments**

x	A <a href="#">labelled</a> vector
.fun	A function that takes .val and .lbl (the values and labels) and returns the values of the label you want to change it to. It is passed to a function similar to <a href="#">as_function</a> , so also accepts quosure-style lambda functions (that use values .val and .lbl). See examples for more information.

**Value**

A haven::labeled vector

**See Also**

Other lbl\_helpers: [lbl\\_add](#), [lbl\\_clean](#), [lbl\\_na\\_if](#), [lbl\\_relabel](#), [lbl](#), [zap\\_ipums\\_attributes](#)

**Examples**

```
x <- haven::labelled(
  c(10, 10, 11, 20, 30, 99, 30, 10),
  c(Yes = 10, `Yes - Logically Assigned` = 11, No = 20, Maybe = 30, NIU = 99)
)

lbl_collapse(x, ~(.val %% 10) * 10)
# Notice that 90 get's NIU from 99 even though 90 didn't have a label in original

lbl_collapse(x, ~ifelse(.lbl == 10, 11, .lbl))
# But here 10 is assigned 11's label

# You can also use the more explicit function notation
lbl_collapse(x, function(.val, .lbl) (.val %% 10) * 10)

# Or even the name of a function
collapse_function <- function(.val, .lbl) (.val %% 10) * 10
lbl_collapse(x, "collapse_function")
```

---

**lbl\_na\_if***Set labelled values to missing*

---

**Description**

Convert values to NA based on their label and value in a [labelled](#) vector. Ignores any value that does not have a label.

**Usage**

```
lbl_na_if(x, .predicate)
```

**Arguments**

<code>x</code>	A <a href="#">labelled</a> vector
<code>.predicate</code>	A function that takes <code>.val</code> and <code>.lbl</code> (the values and labels) and returns TRUE or FALSE. It is passed to a function similar to <a href="#">as_function</a> , so also accepts quosure-style lambda functions (that use values <code>.val</code> and <code>.lbl</code> ). See examples for more information.

**Value**

A `haven::labeled` vector

**See Also**

Other `lbl_helpers`: [lbl\\_add](#), [lbl\\_clean](#), [lbl\\_collapse](#), [lbl\\_relabel](#), [lbl](#), [zap\\_ipums\\_attributes](#)

**Examples**

```
x <- haven::labelled(
  c(10, 10, 11, 20, 30, 99, 30, 10),
  c(Yes = 10, `Yes - Logically Assigned` = 11, No = 20, Maybe = 30, NIU = 99)
)

lbl_na_if(x, ~.val >= 90)
lbl_na_if(x, ~.lbl %in% c("Maybe"))
lbl_na_if(x, ~.val >= 90 | .lbl %in% c("Maybe"))

# You can also use the more explicit function notation
lbl_na_if(x, function(.val, .lbl) .val >= 90)

# Or even the name of a function
na_function <- function(.val, .lbl) .val >= 90
lbl_na_if(x, "na_function")
```

---

**lbl\_relabel***Relabel labelled values*

---

**Description**

Converts values to a new value (that may or may not exist) based on their label and value in a [labelled](#) vector. Ignores any value that does not have a label.

**Usage**

```
lbl_relabel(x, ...)
```

**Arguments**

<code>x</code>	A <a href="#">labelled</a> vector
<code>...</code>	Two-sided formulas where the left hand side is a label placeholder (created with the <a href="#">lbl</a> function) or a value that already exists in the data and the right hand side is a function that returns a logical vector that indicates which labels should be relabeled. The right hand side is passed to a function similar to <a href="#">as_function</a> , so also accepts quosure-style lambda functions (that use values <code>.val</code> and <code>.lbl</code> ). See examples for more information.

**Value**

A `haven::labeled` vector

**See Also**

Other `lbl_helpers`: [lbl\\_add](#), [lbl\\_clean](#), [lbl\\_collapse](#), [lbl\\_na\\_if](#), [lbl](#), [zap\\_ipums\\_attributes](#)



**Examples**

```
x <- haven::labelled(
  c(10, 10, 11, 20, 30, 99, 30, 10),
  c(Yes = 10, `Yes - Logically Assigned` = 11, No = 20, Maybe = 30, NIU = 99)
)

lbl_relabel(
  x,
  lbl(10, "Yes/Yes-ish") ~ .val %in% c(10, 11),
  lbl(90, "???) ~ .val == 99 | .lbl == "Maybe"
)

# If relabelling to labels that already exist, don't need to specify both label
# and value:
# If just bare, assumes it is a value:
lbl_relabel(x, 10 ~ .val == 11)
# Use single argument to lbl for the label
lbl_relabel(x, lbl("Yes") ~ .val == 11)
# Or can use named arguments
lbl_relabel(x, lbl(.val = 10) ~ .val == 11)
```

---

read_ipums_codebook	<i>Read metadata from a text codebook in a NHGIS or Terra area-level extract</i>
---------------------	--

---

**Description**

Read text formatted codebooks provided by some IPUMS extract systems such as NHGIS and Terra Area-level extracts in a format analogous to the DDIs available for other projects.

**Usage**

```
read_ipums_codebook(cb_file, data_layer = NULL)
```

**Arguments**

cb_file	Filepath to the codebook (either the .zip file directly downloaded from the website, or the path to the unzipped .txt file).
data_layer	dplyr <a href="#">select</a> -style notation for uniquely identifying the data layer to load. Required for reading from .zip files for extracts with multiple files.

**Value**

A `ipums_ddi` object with information on the variables included in the csv file of a NHGIS extract.

**See Also**

Other `ipums_metadata`: [read\\_ipums\\_ddi](#)

## Examples

```
# Example NHGIS extract
nhgis_file <- ipums_example("nhgis0008_csv.zip")
ddi <- read_ipums_codebook(nhgis_file)
```

---

read_ipums_ddi	<i>Read metadata about an IPUMS extract from a DDI (.xml) file</i>
----------------	--

---

## Description

Reads the metadata about an IPUMS extract from a DDI file into R. Includes information about variable and value labels, terms of usage for the data and positions for the fixed-width file.

## Usage

```
read_ipums_ddi(ddi_file, data_layer = NULL, lower_vars = FALSE)
```

## Arguments

ddi_file	Filepath to DDI xml file
data_layer	If ddi_file is an extract with multiple DDIs, dplyr <a href="#">select</a> -style notation indicating which .xml data layer to load.
lower_vars	Logical indicating whether to convert variable names to lowercase (default is FALSE due to tradition)

## Value

An ipums\_ddi object with metadata information.

## See Also

Other ipums\_metadata: [read\\_ipums\\_codebook](#)

## Examples

```
# Example extract DDI
ddi_file <- ipums_example("cps_00006.xml")
ddi <- read_ipums_ddi(ddi_file)
```

---

read_ipums_micro	<i>Read data from an IPUMS extract</i>
------------------	--

---

## Description

Reads a dataset downloaded from the IPUMS extract system. For IPUMS projects with microdata, it relies on a downloaded DDI codebook and a fixed-width file. Loads the data with value labels (using [labelled](#) format) and variable labels. See 'Details' for more information on how record types are handled by the ipumsr package.

## Usage

```
read_ipums_micro(ddi, vars = NULL, n_max = Inf, data_file = NULL,
  verbose = TRUE, var_attrs = c("val_labels", "var_label", "var_desc"),
  lower_vars = FALSE)
```

```
read_ipums_micro_list(ddi, vars = NULL, n_max = Inf,
  data_file = NULL, verbose = TRUE, var_attrs = c("val_labels",
  "var_label", "var_desc"), lower_vars = FALSE)
```

## Arguments

ddi	Either a filepath to a DDI xml file downloaded from the website, or a ipums_ddi object parsed by <a href="#">read_ipums_ddi</a>
vars	Names of variables to load. Accepts a character vector of names, or <a href="#">dplyr_select_style</a> conventions. For hierarchical data, the rectype id variable will be added even if it is not specified.
n_max	The maximum number of records to load.
data_file	Specify a directory to look for the data file. If left empty, it will look in the same directory as the DDI file.
verbose	Logical, indicating whether to print progress information to console.
var_attrs	Variable attributes to add from the DDI, defaults to adding all (val_labels, var_label and var_desc). See <a href="#">set_ipums_var_attributes</a> for more details.
lower_vars	If reading a DDI from a file, a logical indicating whether to convert variable names to lowercase (default is FALSE due to tradition)

## Details

Some IPUMS projects have data for multiple types of records (eg Household and Person). When downloading data from many of these projects you have the option for the IPUMS extract system to "rectangularize" the data, meaning that the data is transformed so that each row of data represents only one type of record.

There also is the option to download "hierarchical" extracts, which are a single file with record types mixed in the rows. The ipumsr package offers two methods for importing this data.

read\_ipums\_micro loads this data into a "long" format where the record types are mixed in the rows, but the variables are NA for the record types that they do not apply to.

read\_ipums\_micro\_list loads the data into a list of data frames objects, where each data frame contains only one record type. The names of the data frames in the list are the text from the record type labels without 'Record' (often 'HOUSEHOLD' for Household and 'PERSON' for Person).

### Value

read\_ipums\_micro returns a single tbl\_df data frame, and read\_ipums\_micro\_list returns a list of data frames, named by the Record Type. See 'Details' for more information.

### See Also

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

### Examples

```
# Rectangular example file
cps_rect_ddi_file <- ipums_example("cps_00006.xml")

cps <- read_ipums_micro(cps_rect_ddi_file)
# Or load DDI separately to keep the metadata
ddi <- read_ipums_ddi(cps_rect_ddi_file)
cps <- read_ipums_micro(ddi)

# Hierarchical example file
cps_hier_ddi_file <- ipums_example("cps_00010.xml")

# Read in "long" format and you get 1 data frame
cps_long <- read_ipums_micro(cps_hier_ddi_file)
head(cps_long)

# Read in "list" format and you get a list of multiple data frames
cps_list <- read_ipums_micro_list(cps_hier_ddi_file)
head(cps_list$PERSON)
head(cps_list$HOUSEHOLD)

# Or you can use the \code{%<-%} operator from zeallot to unpack
c(household, person) %<-% read_ipums_micro_list(cps_hier_ddi_file)
head(person)
head(household)
```

---

read\_ipums\_micro\_chunked

*Read data from an IPUMS extract (in chunks)*

---

**Description**

Reads a dataset downloaded from the IPUMS extract system, but does so by reading a chunk, then applying your code to that chunk and then continuing, which can allow you to deal with data that is too large to store in your computer's RAM all at once.

**Usage**

```
read_ipums_micro_chunked(ddi, callback, chunk_size = 10000,
  vars = NULL, data_file = NULL, verbose = TRUE,
  var_attrs = c("val_labels", "var_label", "var_desc"))

read_ipums_micro_list_chunked(ddi, callback, chunk_size = 10000,
  vars = NULL, data_file = NULL, verbose = TRUE,
  var_attrs = c("val_labels", "var_label", "var_desc"))
```

**Arguments**

ddi	Either a filepath to a DDI xml file downloaded from the website, or a ipums_ddi object parsed by <a href="#">read_ipums_ddi</a>
callback	An <a href="#">ipums_callback</a> object, or a function that will be converted to an IpumsSideEffectCallback object.
chunk_size	An integer indicating how many observations to read in per chunk (defaults to 10,000). Setting this higher uses more RAM, but will usually be faster.
vars	Names of variables to load. Accepts a character vector of names, or <a href="#">dplyr_select_style</a> conventions. For hierarchical data, the rectype id variable will be added even if it is not specified.
data_file	Specify a directory to look for the data file. If left empty, it will look in the same directory as the DDI file.
verbose	Logical, indicating whether to print progress information to console.
var_attrs	Variable attributes to add from the DDI, defaults to adding all (val_labels, var_label and var_desc). See <a href="#">set_ipums_var_attributes</a> for more details.

**Value**

Depends on the callback object

**See Also**

Other ipums\_read: [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

**Examples**

```
# Select Minnesotan cases from CPS example (Note you can also accomplish
# this and avoid having to even download a huge file using the "Select Cases"
# functionality of the IPUMS extract system)
mn_only <- read_ipums_micro_chunked(
  ipums_example("cps_00006.xml"),
```

```

IpumsDataFrameCallback$new(function(x, pos) {
  x[x$STATEFIP == 27, ]
}),
chunk_size = 1000 # Generally you want this larger, but this example is a small file
)

# Tabulate INCTOT average by state without storing full dataset in memory
library(dplyr)
inc_by_state <- read_ipums_micro_chunked(
  ipums_example("cps_00006.xml"),
  IpumsDataFrameCallback$new(function(x, pos) {
    x %>%
      mutate(
        INCTOT = lbl_na_if(
          INCTOT, ~.lbl %in% c("Missing.", "N.I.U. (Not in Universe).")
        ) %>%
      filter(!is.na(INCTOT)) %>%
      group_by(STATEFIP = as_factor(STATEFIP)) %>%
      summarize(INCTOT_SUM = sum(INCTOT), n = n())
    }),
  chunk_size = 1000 # Generally you want this larger, but this example is a small file
) %>%
group_by(STATEFIP) %>%
summarize(avg_inc = sum(INCTOT_SUM) / sum(n))

# x will be a list when using `read_ipums_micro_list_chunked()`
read_ipums_micro_list_chunked(
  ipums_example("cps_00010.xml"),
  IpumsSideEffectCallback$new(function(x, pos) {
    print(paste0(nrow(x$PERSON), " persons and ", nrow(x$HOUSEHOLD), " households in this chunk."))
  }),
  chunk_size = 1000 # Generally you want this larger, but this example is a small file
)

# Using the biglm package, you can even run a regression without storing
# the full dataset in memory
library(dplyr)
if (require(biglm)) {
  lm_results <- read_ipums_micro_chunked(
    ipums_example("cps_00015.xml"),
    IpumsBiglmCallback$new(
      INCTOT ~ AGE + HEALTH, # Simple regression (may not be very useful)
      function(x, pos) {
        x %>%
          mutate(
            INCTOT = lbl_na_if(
              INCTOT, ~.lbl %in% c("Missing.", "N.I.U. (Not in Universe).")
            ),
            HEALTH = as_factor(HEALTH)
          )
      }
    ),
    chunk_size = 1000 # Generally you want this larger, but this example is a small file
  )
}
)

```

```
summary(lm_results)
}
```

---

```
read_ipums_micro_yield
```

*Read data from an IPUMS extract (in yields)*

---

## Description

Reads a dataset downloaded from the IPUMS extract system, but does so by returning an object that can read a group of lines at a time. This is a more flexible way to read data in chunks than the functions like [read\\_ipums\\_micro\\_chunked](#), allowing you to do things like reading parts of multiple files at the same time and resetting from the beginning more easily than with the chunked functions.

## Usage

```
read_ipums_micro_yield(ddi, vars = NULL, data_file = NULL,
  verbose = TRUE, var_attrs = c("val_labels", "var_label", "var_desc"),
  lower_vars = FALSE)
```

```
read_ipums_micro_list_yield(ddi, vars = NULL, data_file = NULL,
  verbose = TRUE, var_attrs = c("val_labels", "var_label", "var_desc"),
  lower_vars = FALSE)
```

```
IpumsLongYield
```

```
IpumsListYield
```

## Arguments

ddi	Either a filepath to a DDI xml file downloaded from the website, or a <code>ipums_ddi</code> object parsed by <a href="#">read_ipums_ddi</a>
vars	Names of variables to load. Accepts a character vector of names, or <a href="#">dplyr_select_style</a> conventions. For hierarchical data, the rectype id variable will be added even if it is not specified.
data_file	Specify a directory to look for the data file. If left empty, it will look in the same directory as the DDI file.
verbose	Logical, indicating whether to print progress information to console.
var_attrs	Variable attributes to add from the DDI, defaults to adding all ( <code>val_labels</code> , <code>var_label</code> and <code>var_desc</code> ). See <a href="#">set_ipums_var_attributes</a> for more details.
lower_vars	If reading a DDI from a file, a logical indicating whether to convert variable names to lowercase (default is FALSE due to tradition)

**Format**

An object of class R6ClassGenerator of length 24.

**Details**

These functions return an IpumsYield R6 object which have the following methods:

- `yield(n = 10000)` A function to read the next 'yield' from the data, returns a 'tbl\_df' (or list of 'tbl\_df' for 'hipread\_list\_yield()') with up to n rows (it will return NULL if no rows are left, or all available ones if less than n are available).
- `reset()` A function to reset the data so that the next yield will read data from the start.
- `is_done()` A function that returns whether the file has been completely read yet or not.
- `cur_pos` A property that contains the next row number that will be read (1-indexed).

**Value**

A HipYield R6 object (See 'Details' for more information)

**See Also**

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

**Examples**

```
# An example using "long" data
long_yield <- read_ipums_micro_yield(ipums_example("cps_00006.xml"))
# Get first 10 rows
long_yield$yield(10)
# Get 20 more rows now
long_yield$yield(20)
# See what row we're on now
long_yield$cur_pos
# Reset to beginning
long_yield$reset()
# Read the whole thing in chunks and count Minnesotans
total_mn <- 0
while (!long_yield$is_done()) {
  cur_data <- long_yield$yield(1000)
  total_mn <- total_mn + sum(as_factor(cur_data$STATEFIP) == "Minnesota")
}
total_mn

# Can also read hierarchical data as list:
list_yield <- read_ipums_micro_list_yield(ipums_example("cps_00006.xml"))
list_yield$yield(10)
```



---

read_ipums_sf	<i>Read boundary files from an IPUMS extract</i>
---------------	--

---

### Description

Reads the boundary files from an IPUMS extract into R as simple features (sf) objects or SpatialPolygonsDataFrame (sp) objects.

### Usage

```
read_ipums_sf(shape_file, shape_layer = NULL, vars = NULL,
              encoding = NULL, bind_multiple = TRUE, add_layer_var = NULL,
              verbose = TRUE)
```

```
read_ipums_sp(shape_file, shape_layer = NULL, vars = NULL,
              encoding = NULL, bind_multiple = TRUE, add_layer_var = NULL,
              verbose = TRUE)
```

### Arguments

shape_file	Filepath to one or more .shp files, a .zip file from an IPUMS extract or a path to an unzipped folder.
shape_layer	For .zip extracts with multiple datasets, the name of the shape files to load. Accepts a character vector specifying the file name, or <a href="#">dplyr_select_style</a> conventions. Can load multiple shape files, which will be combined.
vars	Which variables in the shape file's data to keep (NULL the default keeps all)
encoding	The text encoding to use when reading the shape file. Typically the defaults should read the data correctly, but for some extracts you may need to set them manually, but if funny characters appear in your data, you may need to. For microdata projects, the default NULL will look for a .cpg file to determine the encoding and if none is available, it will default to latin1. The NHGIS and the IPUMS Terra functions specify the encoding for those projects (latin1 and UTF-8 respectively).
bind_multiple	If TRUE, will combine multiple shape files found into a single object.
add_layer_var	Whether to add a variable named layer that indicates which shape_layer the data came from. NULL, the default, uses TRUE if more than 1 layer is found, and FALSE otherwise.
verbose	If TRUE, will report progress information

### Value

read\_ipums\_sf returns a sf object and read\_ipums\_sp returns a SpatialPolygonsDataFrame.

### See Also

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

## Examples

```
shape_file <- ipums_example("nhgis0008_shape_small.zip")
# If sf package is available, can load as sf object
if (require(sf)) {
  sf_data <- read_ipums_sf(shape_file)
}

# If sp package is available, can load as SpatialPolygonsDataFrame
if (require(sp) && require(rgdal)) {
  sp_data <- read_ipums_sp(shape_file)
}
```

---

read\_nhgis

*Read data from an NHGIS extract*

---

## Description

Reads a dataset downloaded from the NHGIS extract system. Relies on csv files (with or without the extra header row).

## Usage

```
read_nhgis(data_file, data_layer = NULL, verbose = TRUE,
           var_attrs = c("val_labels", "var_label", "var_desc"))

read_nhgis_sf(data_file, shape_file, data_layer = NULL,
              shape_layer = data_layer, shape_encoding = "latin1",
              verbose = TRUE, var_attrs = c("val_labels", "var_label", "var_desc"))

read_nhgis_sp(data_file, shape_file, data_layer = NULL,
              shape_layer = data_layer, shape_encoding = "latin1",
              verbose = TRUE, var_attrs = c("val_labels", "var_label", "var_desc"))
```

## Arguments

data_file	Filepath to the data (either the .zip file directly downloaded from the website, the path to the unzipped folder, or the path to the unzipped .csv file directly).
data_layer	For .zip extracts with multiple datasets, the name of the data to load. Accepts a character vector specifying the file name, or <a href="#">dplyr_select_style</a> conventions. Data layer must uniquely identify a dataset.
verbose	Logical, indicating whether to print progress information to console.
var_attrs	Variable attributes to add from the codebook, defaults to adding all (val_labels, var_label and var_desc). See <a href="#">set_ipums_var_attributes</a> for more details.
shape_file	Filepath to the shape files (either the .zip file directly downloaded from the website, or the path to the unzipped folder, or the unzipped .shp file directly).

- `shape_layer` (Defaults to using the same value as `data_layer`) Specification of which shape files to load using the same semantics as `data_layer`. Can load multiple shape files, which will be combined.
- `shape_encoding` The text encoding to use when reading the shape file. Typically the defaults should read the data correctly, but for some extracts you may need to set them manually, but if funny characters appear in your data, you may need to. Defaults to "latin1" for NHGIS.

### Value

`read_nhgis` returns a `tbl_df` with only the tabular data, `read_nhgis_sf` returns a `sf` object with data and the shapes, and `read_nhgis_sp` returns a `SpatialPolygonsDataFrame` with data and shapes.

### See Also

Other `ipums_read`: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

### Examples

```
csv_file <- ipums_example("nhgis0008_csv.zip")
shape_file <- ipums_example("nhgis0008_shape_small.zip")

data_only <- read_nhgis(csv_file)

# If sf package is available, can load as sf object
if (require(sf)) {
  sf_data <- read_nhgis_sf(csv_file, shape_file)
}

# If sp package is available, can load as SpatialPolygonsDataFrame
if (require(rgdal) && require(sp)) {
  sp_data <- read_nhgis_sp(csv_file, shape_file)
}
```

---

`read_terra_area`      *Read data from an IPUMS Terra area extract*

---

### Description

Reads a area-level dataset downloaded from the IPUMS Terra extract system.

**Usage**

```
read_terra_area(data_file, data_layer = NULL, ddi_file = NULL,
  cb_file = NULL, verbose = TRUE, var_attrs = c("val_labels",
  "var_label", "var_desc"))
```

```
read_terra_area_sf(data_file, shape_file = NULL, data_layer = NULL,
  shape_layer = data_layer, shape_encoding = "UTF-8",
  ddi_file = NULL, cb_file = NULL, verbose = TRUE,
  var_attrs = c("val_labels", "var_label", "var_desc"))
```

```
read_terra_area_sp(data_file, shape_file = NULL, data_layer = NULL,
  shape_layer = data_layer, shape_encoding = "UTF-8",
  ddi_file = NULL, cb_file = NULL, verbose = TRUE,
  var_attrs = c("val_labels", "var_label", "var_desc"))
```

**Arguments**

<code>data_file</code>	Path to the data file, which can either be the .zip file directly downloaded from the IPUMS Terra website, path to the unzipped folder, or to the csv unzipped from the download.
<code>data_layer</code>	For .zip extracts with multiple datasets, the name of the data to load. Accepts a character vector specifying the file name, or <code>dplyr_select_style</code> conventions. Data layer must uniquely identify a dataset.
<code>ddi_file</code>	(Optional) If the download is unzipped, path to the .xml file which provides usage and citation information for extract.
<code>cb_file</code>	(Optional) If the download is unzipped, path to the .txt file which provides usage and citation information for extract.
<code>verbose</code>	Logical, indicating whether to print progress information to console.
<code>var_attrs</code>	Variable attributes to add from the DDI, defaults to adding all ( <code>val_labels</code> , <code>var_label</code> and <code>var_desc</code> ). See <code>set_ipums_var_attributes</code> for more details.
<code>shape_file</code>	(Optional) If the download is unzipped, path to the .zip, folder path or .shp file representing the the shape file. If only the data table is needed, can be set to <code>FALSE</code> to indicate not to load the shape file.
<code>shape_layer</code>	(Defaults to using the same value as <code>data_layer</code> ) Specification of which shape files to load using the same semantics as <code>data_layer</code> . Can load multiple shape files, which will be combined.
<code>shape_encoding</code>	The text encoding to use when reading the shape file. Typically the defaults should read the data correctly, but for some extracts you may need to set them manually, but if funny characters appear in your data, you may need to. Defaults to "UTF-8" for IPUMS Terra.

**Value**

`read_terra_area` returns a `tbl_df` with the tabular data, `read_terra_area_sf` returns a `sf` object with tabular data and shapes, and `read_terra_area_sp` returns a `SpatialPolygonsDataFrame` with data and shapes.

**See Also**

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_micro](#), [read\\_terra\\_raster](#)

**Examples**

```
## Not run:
data <- read_terra_area("2553_bundle.zip")

## End(Not run)
```

---

read_terra_micro	<i>Read data from an IPUMS Terra microdata extract</i>
------------------	--

---

**Description**

Reads a microdata dataset downloaded from the IPUMS Terra extract system.

**Usage**

```
read_terra_micro(data_file, ddi_file = NULL, data_layer = NULL,
  n_max = Inf, verbose = TRUE, var_attrs = c("val_labels",
  "var_label", "var_desc"))
```

**Arguments**

data_file	Path to the data file, which can either be the .zip file directly downloaded from the IPUMS Terra website, a path to the unzipped version of that folder, or to the csv unzipped from the download.
ddi_file	(Optional) If the download is unzipped, path to the .xml file which provides usage and citation information for extract.
data_layer	For .zip extracts with multiple datasets, the name of the data to load. Accepts a character vector specifying the file name, or <a href="#">dplyr_select_style</a> conventions. Data layer must uniquely identify a dataset.
n_max	Maximum number of observations to read from the data
verbose	Logical, indicating whether to print progress information to console.
var_attrs	Variable attributes to add from the DDI, defaults to adding all (val_labels, var_label and var_desc). See <a href="#">set_ipums_var_attributes</a> for more details.

**Value**

read\_terra\_micro returns a tbl\_df with the tabular data. Use [read\\_ipums\\_sf](#) or [read\\_ipums\\_sp](#) to read shape data out of a microdata Terra extract.

## See Also

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_raster](#)

## Examples

```
## Not run:
data <- read_terra_micro("2553_bundle.zip")

## End(Not run)
```

---

read_terra_raster	<i>Read data from an IPUMS Terra raster extract</i>
-------------------	---

---

## Description

Read a single raster datasets downloaded from the IPUMS Terra extract system using `read_terra_raster`, or read multiple into a list using `read_terra_raster_list`.

## Usage

```
read_terra_raster(data_file, data_layer = NULL, verbose = TRUE)

read_terra_raster_list(data_file, data_layer = NULL, verbose = TRUE)
```

## Arguments

<code>data_file</code>	Filepath to the data (either the .zip file directly downloaded from the website, or the path to the unzipped .tiff file(s)).
<code>data_layer</code>	For .zip extracts with multiple raster datasets, the name of the data to load. Accepts a character vector specifying the file name, or <a href="#">dplyr_select_style</a> conventions.
<code>verbose</code>	Logical, indicating whether to print progress information to console.

## Value

For `read_terra_raster` A [raster](#) object, for `read_terra_raster_list` A list of raster objects.

## See Also

Other ipums\_read: [read\\_ipums\\_micro\\_chunked](#), [read\\_ipums\\_micro\\_yield](#), [read\\_ipums\\_micro](#), [read\\_ipums\\_sf](#), [read\\_nhgis](#), [read\\_terra\\_area](#), [read\\_terra\\_micro](#)

## Examples

```
## Not run:
data <- read_terra_raster("2552_bundle.zip", "LCDECIDOPZM2013.tiff")
data <- read_terra_raster_list("2552_bundle.zip", "ZM")

## End(Not run)
```

---

```
set_ipums_var_attributes
```

*Add IPUMS variable attributes to a data.frame*

---

## Description

Add variable attributes from an IPUMS DDI to the variables in a data.frame. This function is usually called automatically for you inside of the read\_\* functions (such as read\_ipums\_micro or read\_nhgis), but they can be useful other times as well. For example, if you store the data in a database, you can store the data without attributes in the database and add them on after loading a subset into a data.frame.

## Usage

```
set_ipums_var_attributes(data, var_info, var_attrs = c("val_labels",
  "var_label", "var_desc"))
```

## Arguments

data	A data.frame
var_info	An ipums_ddi object or a data.frame with the variable information (equivalent to getting ipums_var_info on a DDI).
var_attrs	One or more of val_labels, var_label and var_desc describing what kinds of attributes you want to add. If NULL, will not add any attributes.

## Details

Attribute val\_labels adds the haven::labelled class attributes and the corresponding value labels for variables that have value labels.

Attribute var\_label Adds a short summary of the variable's contents that to the attribute "label". This label is viewable in the RStudio Viewer.

Attribute var\_desc Adds a longer summary of the variable's contents to the attribute "var\_desc" when available.

## Value

A tbl\_df data.frame with data and IPUMS attributes

**Examples**

```

ddi_file <- ipums_example("cps_00006.xml")
ddi <- read_ipums_ddi(ddi_file)
cps <- read_ipums_micro(ddi, var_attrs = NULL) # Don't load with attributes

ipums_var_desc(cps$YEAR) # Not available

# But, we can add on attributes after loading
cps_with_attr <- set_ipums_var_attributes(cps, ddi)
ipums_var_desc(cps_with_attr$YEAR)

```

---

zap\_ipums\_attributes *Remove all IPUMS attributes from a variable (or all variables in a data.frame)*

---

**Description**

Helper to remove ipums attributes (including value labels from the labelled class, the variable label and the variable description). These attributes can sometimes get in the way of functions like the dplyr join functions so you may want to remove them.

**Usage**

```
zap_ipums_attributes(x)
```

**Arguments**

x                    A variable or a whole data.frame to remove attributes from

**Value**

A variable or data.frame

**See Also**

Other lbl\_helpers: [lbl\\_add](#), [lbl\\_clean](#), [lbl\\_collapse](#), [lbl\\_na\\_if](#), [lbl\\_relabel](#), [lbl](#)

**Examples**

```

cps <- read_ipums_micro(ipums_example("cps_00006.xml"))
annual_unemployment <- data.frame(YEAR = c(1962, 1963), unemp = c(5.5, 5.7))

# Avoids warning 'Column `YEAR` has different attributes on LHS and RHS of join'
cps$YEAR <- zap_ipums_attributes(cps$YEAR)
cps <- dplyr::left_join(cps, annual_unemployment, by = "YEAR")

```



# Index

## \*Topic **datasets**

- read\_ipums\_micro\_yield, 23
- as\_function, 13–16
- collect, 4
- dplyr\_select\_style, 2, 7, 19, 21, 23, 25, 26, 28–30
- ipums\_bind\_rows, 3
- ipums\_callback, 21
- ipums\_collect, 4
- ipums\_conditions, 4
- ipums\_example, 5
- ipums\_file\_info, 5
- ipums\_list\_data (ipums\_list\_files), 6
- ipums\_list\_files, 6
- ipums\_list\_raster (ipums\_list\_files), 6
- ipums\_list\_shape (ipums\_list\_files), 6
- ipums\_shape\_full\_join
  - (ipums\_shape\_left\_join), 7
- ipums\_shape\_inner\_join
  - (ipums\_shape\_left\_join), 7
- ipums\_shape\_left\_join, 7, 11
- ipums\_shape\_right\_join
  - (ipums\_shape\_left\_join), 7
- ipums\_val\_labels (ipums\_var\_info), 8
- ipums\_var\_desc (ipums\_var\_info), 8
- ipums\_var\_info, 8
- ipums\_var\_label (ipums\_var\_info), 8
- ipums\_view, 9
- ipums\_website, 10
- IpumsListYield
  - (read\_ipums\_micro\_yield), 23
- IpumsLongYield
  - (read\_ipums\_micro\_yield), 23
- join\_failures, 11
- labelled, 13–16, 19
- lbl, 12, 13–16, 32
- lbl\_add, 12, 12, 14–16, 32
- lbl\_add\_vals (lbl\_add), 12
- lbl\_clean, 12, 13, 13, 14–16, 32
- lbl\_collapse, 12–14, 14, 15, 16, 32
- lbl\_na\_if, 12–14, 15, 16, 32
- lbl\_relabel, 12–15, 16, 32
- raster, 30
- read\_ipums\_codebook, 17, 18
- read\_ipums\_ddi, 4, 5, 8, 9, 17, 18, 19, 21, 23
- read\_ipums\_micro, 19, 21, 24, 25, 27, 29, 30
- read\_ipums\_micro\_chunked, 20, 20, 23–25, 27, 29, 30
- read\_ipums\_micro\_list
  - (read\_ipums\_micro), 19
- read\_ipums\_micro\_list\_chunked
  - (read\_ipums\_micro\_chunked), 20
- read\_ipums\_micro\_list\_yield
  - (read\_ipums\_micro\_yield), 23
- read\_ipums\_micro\_yield, 20, 21, 23, 25, 27, 29, 30
- read\_ipums\_sf, 7, 20, 21, 24, 25, 27, 29, 30
- read\_ipums\_sp, 29
- read\_ipums\_sp (read\_ipums\_sf), 25
- read\_nhgis, 20, 21, 24, 25, 26, 29, 30
- read\_nhgis\_sf (read\_nhgis), 26
- read\_nhgis\_sp (read\_nhgis), 26
- read\_terra\_area, 20, 21, 24, 25, 27, 27, 30
- read\_terra\_area\_sf (read\_terra\_area), 27
- read\_terra\_area\_sp (read\_terra\_area), 27
- read\_terra\_micro, 20, 21, 24, 25, 27, 29, 29, 30
- read\_terra\_raster, 20, 21, 24, 25, 27, 29, 30, 30
- read\_terra\_raster\_list
  - (read\_terra\_raster), 30
- select, 2, 6, 8, 17, 18

set\_ipums\_var\_attributes, [4](#), [19](#), [21](#), [23](#),  
[26](#), [28](#), [29](#), [31](#)

zap\_ipums\_attributes, [12–16](#), [32](#)