opendataformat

Contents

Introducing the opendataformat package

Tired of struggling to convert open data formats into R dataframes? Look no further and welcome the opendataformat package.

With just a few lines of code, you can convert a data package specified as opendataformat into an R dataframe (`read_opendf()`) or convert an R dataframe into a data package specified in the opendataformat (`write_opendf()`).

But wait, there's more! Our package goes beyond parsing. Accessing metadata has never been easier. Dive into the treasure trove of information stored in your R dataframes. Explore dataset labels, descriptions, and valuable details about variables such as labels and value labels with the `docu_opendf()` function and the `labels_opendf()` function.

This vignette will guide you through a series of examples that demonstrate the possibilities of these functions. Let's get started!

Installation

You can install and update the package from gitlab using the `install_git()`-function from the devtools-library.

```r
library(devtools)
#> Lade nötiges Paket: usethis
devtools::install_git("https://git.soep.de/opendata/r-package-opendataformat.git",
                      quiet = TRUE)
```

```
library(opendataformat)
#> Lade nötiges Paket: crayon
#> Lade nötiges Paket: magrittr
#> Lade nötiges Paket: xml2
#> Lade nötiges Paket: data.table
```

Functions in the openmetadata Package

Read the Open Data Format: `read_opendf()`

The opendataformat package provides example data that is specified as 'Open Data Format'. Data in the open data format is a ZIP file containing a csv-file and a xml-file. The example data contains the data-csv (`data.csv`) with 20 rows and 7 columns and the metadata-XML (`metadata.xml`). The metadata file describes the dataset and its variables. If you are interested in what these files look like, you can find an example in the Open Data Format git repository: https://git.soep.de/opendata/specification/-/tree/main/external/example To load the data, we need to specify the path to the zip-file. Here the example data is loaded using `read_opendf()` Alternatively, you can set `file` to a zip-file in the working directory.

```
path <- system.file("extdata", "data.zip", package="opendataformat")
df <- read_opendf(file = path)
#df <- read_opendf(file = "../my_data.zip")
```

The output of the `read_opendf()` function is an R-data.frame object, that has the additional class `opendf`. It has additional metadata stored in the attributes of the dataframe and the variables/columns. These include the languages of the metadata, labels, descriptions, urls, variable types, and value labels.

```
df
   bap87 bap9201 bap9001 bap9002 bap9003 bap96      name
1      4      -2       1      -1       2 -2.00     Jakob
2      3       5      -2       1       4  1.57      Luca
3     NA      -1      -1       2      -1  1.92    Emilia
4      1       9      -2       2       4  1.85        -1
5     -1       4       2       3       1  1.91   Johanna
6      3       4      -1       4      -2  1.80      Paul
7      1       9       2      -1      -1  1.80
8      5       6       1      -1       1  1.96       Mia
9      5       5       5       3       1  1.64       Ben
10    -2       4       4      -1      -2  1.93     Jakob
11    -1       4       2       1       5  1.93     Anton
12    -2       5       3      -2       4    NA Charlotte
13     3      -1       2       1       2  1.74      Luca
14     2      -2      -2       4      -1  1.65     Maria
15     5      -1      -2      -1      -1  1.80   Johanna
16     4       5       1       3      -1  1.58      Emma
17     3       7       1       2      -2  1.95     Felix
18     3      NA       5       3      -2  1.98     David
19    -2       8       1       4       5  1.61        -2
20     2       8       3       1       2  1.83     Anton
```

If you load the haven package, you see the variable labels in the active language .

```r
#library(haven)
#View(df)
```

If you want to import a dataset with metadata only in one or several languages. You can use the `languages`-argument. To load the example data only with english labels and descriptions, set `languages="en"`:

```r
df_en <- read_opendf(file = path, languages="en")
```

By default `languages="all"`:

```r
df_en <- read_opendf(file = path, languages="all")
```

You can also give a list of languages::

```r
df_en <- read_opendf(file = path, languages=c("en", "de"))
```

Explore Dataset Information: `docu_opendf()`

Display Medatata

You can explore dataset information using two methods. Firstly, you can browse metadata at the record level, providing an overview of the dataset. Alternatively, you have the option to examine specific variable details, allowing you to gain insights into selected data attributes.

By default, when using the `docu_opendf()` function, dataset-level information is presented through the console and an HTML page. If you're utilizing RStudio, this html-page will be displayed within the RStudio viewer.

```r
docu_opendf(df)
```

To display the metadata only in the console, utilize the `style` argument with the value set to `print` (or `console`). This ensures that the information is conveniently displayed on the R console, serving our specific demonstration purposes. To display metadata information only in the viewer, set `style="viewer"` or `style="html"`. By default `style="both"`.

```r
docu_opendf(df, style = "print")
#> Dataset:  bap
#> Label:
#> [en]Data from individual questionnaires 2010
#> Description:
#> [en]The data were collected as part of the SOEP-Core study using the questionnaire "Living in Germany
#> languages:
#>     en de (active: en)
#> url:
#>     https://paneldata.org/soep-core/data/bap
```

To obtain a comprehensive overview of all variables within the dataset, simply set the agrument `variables="yes"`.

```
docu_opendf(df, variables="yes", style="print")
Dataset:  bap
Label:
[en]Data from individual questionnaires 2010
Description:
[en]The data were collected as part of the SOEP-Core study using the questionnaire "Living in Germany -
languages:
    en de (active: en)
url:
    https://paneldata.org/soep-core/data/bap
Variables:
  Variable                           Label en
1    bap87                    Current Health
2  bap9201    hours of sleep, normal workday
3  bap9001     Pressed For Time Last 4 Weeks
4  bap9002 Run-down, Melancholy Last 4 Weeks
5  bap9003        Well-balanced Last 4 Weeks
6    bap96                            Height
7     name                         Firstname
```

If you are interested in just one specific variable, you can do this:

```
docu_opendf(df$bap9001, style = "print")
#> Variable:  bap9001
#> Label:
#> [en]Pressed For Time Last 4 Weeks
#> Description:
#> [en]Frequency of feeling time pressure in the past 4 weeks
#> type:
#>     numeric
#> url:
#>     https://paneldata.org/soep-core/data/bap/bap9001
#> Value Labels:
#>                 Value             en
#> Does not apply    -2 Does not apply
#> No Answer         -1      No Answer
#> Always             1         Always
#> Often              2          Often
#> Sometimes          3      Sometimes
#> Almost Never       4   Almost Never
#> Never              5          Never
```

Display Metadata in a Specific Language

Certain datasets offer metadata such as labels, descriptions, or value labels in multiple languages. To display the metadata in all languages supported by your dataset, you can simply set the `languages` argument to `all`. This setting enables you to identify the range of languages available for accessing the relevant metadata within your dataset.

```
docu_opendf(df$bap9001, style = "print", variables="yes", languages = "all")
```

If you have a specific language of interest, you can easily display it by utilizing the corresponding language code. Simply specify the desired language code to retrieve the metadata in the language of your choice.

This enables you to access the specific language variant of variable labels, value labels. In this example, we display the German version:

```
docu_opendf(df$bap9001, style = "print", languages = "de")
#> Variable:  bap9001
#> Label:
#> [de]Eile, Zeitdruck letzten 4 Wochen
#> Description:
#> [de]Häufigkeit des Gefühls von Zeitdruck in den letzten 4 Wochen
#> type:
#>     numeric
#> url:
#>     https://paneldata.org/soep-core/data/bap/bap9001
#> Value Labels:
#>                  Value              de
#> trifft nicht zu    -2 trifft nicht zu
#> keine Angabe       -1    keine Angabe
#> Immer               1           Immer
#> Oft                 2             Oft
#> Manchmal            3        Manchmal
#> Fast nie            4        Fast nie
#> Nie                 5             Nie
```

You can apply this function to the entire dataset, allowing you to access the desired information across all variables.

```
docu_opendf(df, style = "print", variables="yes", languages = "de")
```

If you prefer another display style, you can use the datasets' metadata directly from the attributes and write your own code:

```
for (i in names(df)) {
  cat(
    paste0(attributes(df[[i]])$name, ": ", attributes(df[[i]])$label_de, "\n")
  )
}
bap87: Gesundheitszustand gegenwärtig
bap9201: Stunden Schlaf, normaler Werktag
bap9001: Eile, Zeitdruck letzten 4 Wochen
bap9002: Niedergeschlagen letzten 4 Wochen
bap9003: Ausgeglichen letzten 4 Wochen
bap96: Körpergröße
name: Vorname
```

You can also use the labels_opendf() function to retrieve labels and other metadata for the variables:

```
labels_opendf(df)
                                bap87                              bap9201
                     "Current Health"    "hours of sleep, normal workday"
                              bap9001                              bap9002
    "Pressed For Time Last 4 Weeks" "Run-down, Melancholy Last 4 Weeks"
                              bap9003                                bap96
       "Well-balanced Last 4 Weeks"                             "Height"
```

```
                                 name
                          "Firstname"
```

or the value labels:

```
labels_opendf(df$bap87, valuelabels=T)
Does not apply      No Answer      Very good           Good   Satisfactory
            -2             -1              1              2              3
          Poor            Bad
             4              5
```

Set the Active Language of a Data Frame: `setLanguage_opendf()`

Alternatively, you can set the current (active) language for a dataset-object. (This function tries to copy the label language function from Stata.)

```
df<-setLanguage_opendf(df, language="de")

docu_opendf(df$bap9001, style = "print")
```

To display which languages are available for the dataset metadata, display the `languages` attribute:

```
attributes(df)$languages
#> [1] "en" "de"
attr(df, "languages")
#> [1] "en" "de"
```

Access Metadata: `labels_opendf()` and `attributes()`

Browsing through datasets' metadata provides a valuable initial overview. However, when it comes time to dive into the analysis work, questions arise regarding the storage location of the metadata and the process of accessing and utilizing it. Let's explore how and where the metadata is stored, and how we can effectively access and leverage it for analysis purposes.

A easy way to retrieve metadata is to use the `labels_opendf()` function to get labels and other metadata.

Retrieve the Attributes with `attributes()` and `attr()`

Another way is to retrieve metadata directly from the attributes. The metadata imported from the Open Data Format file into an R dataframe is stored as R attributes. By using the base R functions `attributes()` and `attr()`, you can easily access this metadata. When providing the entire dataset to the function, R will display all the metadata describing the dataset as a whole in your console.

```
attributes(df)
$names
[1] "bap87"   "bap9201" "bap9001" "bap9002" "bap9003" "bap96"   "name"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
$name
[1] "bap"

$description_en
[1] "The data were collected as part of the SOEP-Core study using the questionnaire \"Living in Germany

$description_de
[1] "Die Daten wurden im Rahmen der Studie SOEP-Core mittels des Fragebogens „Leben in Deutschland - Be

$label_en
[1] "Data from individual questionnaires 2010"

$label_de
[1] "Daten vom Personenfragebogen 2010"

$url
[1] "https://paneldata.org/soep-core/data/bap"

$languages
[1] "en" "de"

$lang
[1] "en"

$label
[1] "Data from individual questionnaires 2010"

$class
[1] "data.frame" "opendf"
```

If you provide a specific variable to the function, only the corresponding metadata for that variable will be printed.

```
attributes(df$bap87)
$name
[1] "bap87"

$label_en
[1] "Current Health"

$label_de
[1] "Gesundheitszustand gegenwärtig"

$description_en
[1] "Question: How would you describe your current health?"

$description_de
[1] "Frage: Wie würden Sie Ihren gegenwärtigen Gesundheitszustand beschreiben?"

$type
[1] "numeric"

$url
```

```
[1] "https://paneldata.org/soep-core/data/bap/bap87"

$labels_en
Does not apply      No Answer      Very good           Good   Satisfactory
            -2             -1              1              2              3
          Poor            Bad
             4              5

$labels_de
  trifft nicht zu      keine Angabe          Sehr gut                Gut
               -2                -1                 1                  2
Zufriedenstellend       Weniger gut          Schlecht
                3                 4                 5

$languages
[1] "en" "de"

$lang
[1] "en"

$label
[1] "Current Health"
```

If you're interested in a particular attribute, you can access it using the dollar sign followed by the attribute name. For instance, let's consider accessing a variable label in German (language code: de) as an example.

```
attributes(df$bap87)$label_de
[1] "Gesundheitszustand gegenwärtig"
```

Alternatively, you can use the `attr()` function to get the same result:

```
attr(df$bap87, "label_de")
```

Moreover, you have the flexibility to copy, remove, and modify these attributes to suit your needs.

```
attributes(df$bap87)$description_de<-NULL
attributes(df$bap87)$description_de
NULL
```

Retrieve Metadata with `labels_opendf()`

You can also use the `labels_opendf()` function to retrieve labels and other metadata for the variables. By default, the function will return the variable labels for a dataset:

```
labels_opendf(df)
                               bap87                              bap9201
                    "Current Health"    "hours of sleep, normal workday"
                             bap9001                              bap9002
    "Pressed For Time Last 4 Weeks" "Run-down, Melancholy Last 4 Weeks"
                             bap9003                                bap96
       "Well-balanced Last 4 Weeks"                             "Height"
                                name
                         "Firstname"
```

or for a specific variable::

```
labels_opendf(df$bap96)
   bap96
"Height"
```

To retrieve labels in a specific language, use the language parameter:

```
labels_opendf(df, language="en")
```

Or set the active language of the dataset using the `setLanguage_opendf()` function:

```
df<-setLanguage_opendf(df, language="en")
labels_opendf(df)
```

You can also use the `labels_opendf()` function to retrieve value labels for a specific variable by setting the argument `valuelabels=TRUE`:

```
labels_opendf(df$bap9001, valuelabels=T)
Does not apply      No Answer         Always          Often      Sometimes
            -2             -1              1              2              3
  Almost Never          Never
             4              5
```

Alternatively, you can set the argument `retrieve="valuelabels"`:

```
labels_opendf(df$bap9001, retrieve="valuelabels")
```

The value labels for each value are stored in the namespace:

```
names(labels_opendf(df$bap9001, valuelabels=T))
[1] "Does not apply" "No Answer"      "Always"         "Often"
[5] "Sometimes"      "Almost Never"   "Never"
```

You can use the `labels_opendf()` function to return descriptions, urls, variable types and metadata languages as well:

To retrieve variable description(s), set the argument `retrieve="description"`:

```
labels_opendf(df, retrieve="description")
                                                     bap87
 "Question: How would you describe your current health?"
                                                   bap9201
                                   "Sleep hours per weekday"
                                                   bap9001
"Frequency of feeling time pressure in the past 4 weeks"
                                                   bap9002
        "Frequency of feeling a sad and depressed state"
                                                   bap9003
                          "Frequency of feeling balance"
                                                     bap96
                                               "Body size"
                                                      name
                                               "Firstname"
```

To retrieve variable url(s), set the argument `retrieve="url"`:

```
labels_opendf(df, retrieve="url")
```

To retrieve variable type(s), set the argument `retrieve="type"`:

```
labels_opendf(df, retrieve="type")
```

Export to the Open Data Format: `write_opendf()`

To save a dataset as opendf-file, we can use the `write_opendf()` function. Let's assume we want to save the first four columns of our dataset as a new opendf-file. We use the `write_opendf()` function and indicate the r-dataframe and the file name (and location if it).

```
write_opendf(
  x = df[,1:4],
  file = "../df_1_4.zip"
)

#or :
df_14<-df[,1:4]
write_opendf(
  x = df[,1:4],
  file = "df_1_4.zip"
)
```

The XML file metadata.xml and the CSV file data.csv are saved within the directory 'data_rec', as well as within the ZIP file 'data_rec.zip. The dataset looks the same as before, just with fewer variables:

```
<?xml version='1.0' encoding='utf-8'?>
<codeBook xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ddi:codebook:2_5 ht
    <fileDscr>
        <fileTxt>
            <fileName>bap</fileName>
            <fileCont xml:lang="en">The data were collected as part of the SOEP-Core study using the qu
            <fileCont xml:lang="de">Die Daten wurden im Rahmen der Studie SOEP-Core mittels des Fragebo
            <fileCitation>
                <titlStmt>
                    <titl xml:lang="en">Data from individual questionnaires 2010</titl>
                    <titl xml:lang="de">Daten vom Personenfragebogen 2010</titl>
                </titlStmt>
            </fileCitation>
        </fileTxt>
        <notes>
            <ExtLink URI="https://paneldata.org/soep-core/data/bap" />
        </notes>
    </fileDscr>
    <dataDscr>
        <var name="bap87">
            <labl xml:lang="en">Current Health</labl>
            <labl xml:lang="de">Gesundheitszustand gegenwärtig</labl>
            <txt xml:lang="en">Question: How would you describe your current health?</txt>
            <txt xml:lang="de">Frage: Wie würden Sie Ihren gegenwärtigen Gesundheitszustand beschreiben
            <notes>
                <ExtLink URI="https://paneldata.org/soep-core/data/bap/bap87" />
            </notes>
            <varFormat type="numeric" />
            <catgry>
                <catValu>-2</catValu>
                <labl xml:lang="en">Does not apply</labl>
                <labl xml:lang="de">trifft nicht zu</labl>
            </catgry>
            <catgry>
                <catValu>-1</catValu>
                <labl xml:lang="en">No Answer</labl>
                <labl xml:lang="de">keine Angabe</labl>
            </catgry>
            <catgry>
                <catValu>1</catValu>
                <labl xml:lang="en">Very good</labl>
                <labl xml:lang="de">Sehr gut</labl>
...
```

The data.csv file now includes just four columns:

```
"bap87","bap9201","bap9001","bap9002"
4,-2,1,-1
3,5,-2,1
,-1,-1,2
1,9,-2,2
-1,4,2,3
3,4,-1,4
1,9,2,-1
...
```

If you wish to export only the metadata for documentation or archiving purposes, you can achieve this by setting the argument `export_data="no"`. By doing so, the resulting directory or zip file will solely contain the metadata XML file, excluding the data CSV file. This allows you to specifically capture and preserve the metadata without including the actual data, providing a solution for documentation or archiving needs.

```
write_opendf(
  x=df,
  file = "../df_metadata.zip",
  export_data = FALSE
)
```

If you wish to export the dataset with the metadata only in one or some languages, set the languages argument to `languages=c("en")`. Default: `languages="all"`

```
write_opendf(
  x=df,
  file = "../df_en.zip",
  languages = "en"
)
```

By default, languages is set to`languages="all"`. You can also define a list of languages to be exported:

```
write_opendf(
  x=df,
  file = "../df_en_de.zip",
  languages = c("en","de")
)
```

Let's Analyse and Make Use of the Metadata

Now let's see how we can use the metadata to better understand the data and make more informative plots.

Frequency Table

```
table(df$bap87, useNA = "ifany")

  -2   -1    1    2    3    4    5 <NA> 
   3    2    2    2    5    2    3    1 
```

As expected, the frequency table displays the occurrence count of each variable value. Now, let's enhance the convenience of the frequency table by utilizing the value labels associated with the variables. To access the value labels, as explained in the preceding section, you can utilize the base R function `attributes()`. Let's proceed to examine them now:

```
attributes(df$bap87)$labels_en
Does not apply      No Answer      Very good           Good   Satisfactory 
            -2             -1              1              2              3 
          Poor            Bad 
             4              5 
```

```
attributes(df$bap87)$labels_de
  trifft nicht zu      keine Angabe          Sehr gut               Gut 
               -2                -1                 1                 2 
Zufriedenstellend       Weniger gut          Schlecht 
                3                 4                 5 
```

```
table(
  factor(
    df$bap87, labels = names(attributes(df$bap87)$labels_en
                             )
    )
  )

Does not apply      No Answer      Very good           Good   Satisfactory 
             3              2              2              2              5 
          Poor            Bad 
             2              3 
```

Alternatively you can use the labels_opendf()-function to get the value labels:

```
table(
  factor(
    df$bap87, labels = names(labels_opendf(df$bap87, valuelabels=T))
    )
  )
```

To display the data in a language other than the default one, let's try German by appending the respective language code to the attribute name. For example, you can use $labels_de to access the German language labels and present the information accordingly.

```
table(
  factor(
    df$bap87,
      labels=names(attributes(df$bap87)$labels_de)
    )
  )

  trifft nicht zu      keine Angabe          Sehr gut                Gut
                3                 2                 2                  2
Zufriedenstellend       Weniger gut          Schlecht
                5                 2                 3
```

Or using labels_opendf()-function:

```
table(
  factor(
    df$bap87,
      labels=names(labels_opendf(df$bap87, valuelabels=T, language="de"))
    )
  )

  trifft nicht zu      keine Angabe          Sehr gut                Gut
                3                 2                 2                  2
Zufriedenstellend       Weniger gut          Schlecht
                5                 2                 3
```

Recoding a Variable

We want to display the table with only valid answers. Therefore, we set the values `-2` and `-1` to `NA`. Because we do not want to overwrite the original variable, we generate a new one:

```
bap87_rec <- df$bap87
```

We check the attributes of the metadata and notice they are also copied from the original variable to the new one:

```
attributes(bap87_rec)
$name
[1] "bap87"

$label_en
```

```
[1] "Current Health"

$label_de
[1] "Gesundheitszustand gegenwärtig"

$description_en
[1] "Question: How would you describe your current health?"

$type
[1] "numeric"

$url
[1] "https://paneldata.org/soep-core/data/bap/bap87"

$labels_en
Does not apply      No Answer      Very good           Good   Satisfactory
            -2             -1              1              2              3
          Poor            Bad
             4              5

$labels_de
  trifft nicht zu      keine Angabe          Sehr gut                Gut
               -2                -1                 1                  2
Zufriedenstellend       Weniger gut          Schlecht
                3                 4                 5

$languages
[1] "en" "de"

$lang
[1] "en"

$label
[1] "Current Health"
```

Now we can set the negative values to NA:

```
for (row in 1:length(bap87_rec)) {
  if (!is.na(bap87_rec[row]) && bap87_rec[row] <= -1) {
    bap87_rec[row] <- NA
  }
}

table(bap87_rec, useNA = "ifany")
bap87_rec
   1    2    3    4    5 <NA>
   2    2    5    2    3    6
```

We notice that the copied values and value labels do not fit anymore:

```
attributes(bap87_rec)$labels_en
Does not apply      No Answer      Very good           Good   Satisfactory
            -2             -1              1              2              3
```

```
          Poor              Bad
             4                5
```

To change that, we'll copy positions 3 to 7, retaining the desired range of values and their respective value labels.

```
attributes(bap87_rec)$labels_en <- unname(attributes(df$bap87)$labels_en)[3:7] # values
names(attributes(bap87_rec)$labels_en) <- names(attributes(df$bap87)$labels_en)[3:7] # labels

attributes(bap87_rec)$labels_en
   Very good         Good Satisfactory          Poor           Bad
           1            2            3            4             5
```

Do the same for the other language versions of the new recoded variable:

```
attributes(bap87_rec)$labels_de <- unname(attributes(df$bap87)$labels_de)[3:7] # values
names(attributes(bap87_rec)$labels_de) <- names(attributes(df$bap87)$labels_de)[3:7] # labels
```

We do also notice that the variable name is not adequate. We replace the name copied from the original variable with the new name bap87_rec.

```
attributes(bap87_rec)$name <- "bap87_rec"
attributes(bap87_rec)$name
[1] "bap87_rec"
```

Now we generate the frequency table by using the variable as a factor variable.

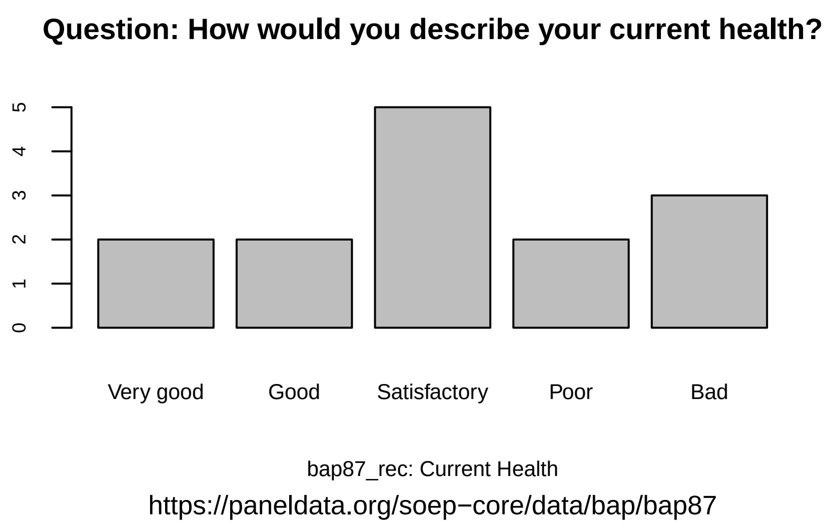
```
table(
  factor(
    bap87_rec,
      labels=names(attributes(bap87_rec)$labels_en)
    )
  )

   Very good         Good Satisfactory          Poor           Bad
           2            2            5            2             3
```

Barplot

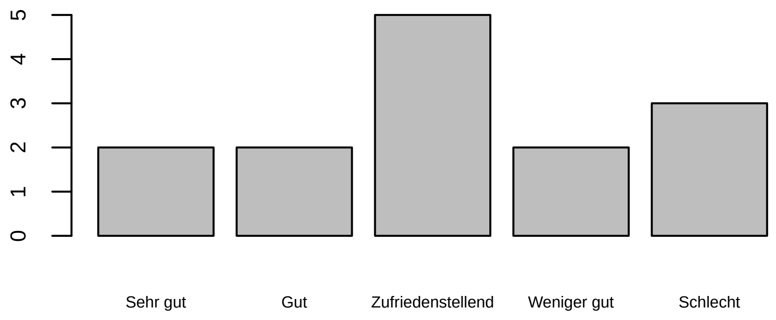
To create a barplot, we will utilize the recoded variable from the previous section. This example will demonstrate how to leverage metadata to create a more convenient and informative graph. By incorporating the metadata into the visualization, we can enhance the graph's interpretability and provide a clearer understanding of the data.

```
barplot(
  table(
  factor(
    bap87_rec,
      labels=names(attributes(bap87_rec)$labels_en)
    )
  ),
  main = attributes(bap87_rec)$description_en, # title
  xlab = paste0(
    attributes(bap87_rec)$name,": ", attributes(bap87_rec)$label), # label
  sub = attributes(bap87_rec)$url, # subtitle
  cex.main = 0.9, cex.names = 0.7, cex.sub = 0.8, cex.axis = 0.6, cex.lab = 0.7 # font sizes
)
```



Drawing a barplot with the German description becomes effortless when dealing with dates that have multiple language versions of labels and descriptions. Simply append the language code to the end of the label attributes, and you'll be able to generate the desired barplot with the German description:

```
barplot(
  table(
  factor(
    bap87_rec,
      labels=names(attributes(bap87_rec)$labels_de)
    )
  ),
  main = attributes(bap87_rec)$description_de, # title
  xlab = paste0(
    attributes(bap87_rec)$name,": ", attributes(bap87_rec)$label_de), # label
  sub = attributes(bap87_rec)$url, # subtitle
  cex.main = 0.7, cex.names = 0.5, cex.sub = 0.8, cex.axis = 0.7, cex.lab = 0.7 # font sizes
)
```



bap87_rec: Gesundheitszustand gegenwärtig

https://paneldata.org/soep−core/data/bap/bap87