UNHCR Microdata Curation Handbook - 2023

UNHCR Global Data Service - Statistics and Demographics Section

2/6/23

Table of contents

About the handbook

This handbook is a technical guide for curating personal microdata of persons of concern (PoCs) to UNHCR, and uploading this data internally on the Raw Internal Data Library (RIDL) and publishing it externally on UNHCR's Microdata Library (MDL). It is complementary to the Administrative Instruction, *Curation of Personal Microdata of PoCs to UNHCR, herein referred to as the 'Curation AI'*, and was written with the assumption that any user of the handbook has read and understands the Curation AI.

For guidance on curation and anonymization of specific UNHCR surveys, visit this page instead. Access can be granted by the Curation Team.

Part I

Getting Started

1 Introduction

Key definitions

Microdata is data on the characteristics of statistical units of a population, such as individuals, households, or establishments, collected by a census, survey, or registry.

Data curation is the principled and controlled treatment, organization and integration of data from various sources to systems where it is preserved, maintained, discoverable and accessible for re-use. The end goal of curation is to make data ready for analysis. For the purposes of UNHCR's work, data curation includes uploading data and metadata to one of UNHCR's data libraries and may also include treating data for errors and anonymization.

1.1 Purpose and objective

The purpose of this handbook is to provide detailed technical guidance for Data Curators on the curation of personal microdata of PoCs. The objective is to ensure that the implementation of the Data Curator role is in line with the Curation AI and done in a consistent manner and following best practices.

Note that any mention of microdata or microdataset or, more generally, data or dataset refers to personal microdata of PoCs unless otherwise stated.

1.2 Audience

The primary audience is UNHCR personnel performing the Data Curator role in the curation of microdata of PoCs. The handbook was written with the assumption that anyone using it already has a basic to advanced knowledge of data treatment, statistics, anonymization and statistical applications such as R and Stata.

Secondary audiences of this handbook include any UNHCR personnel involved in data collection, processing and dissemination and any UNHCR personnel involved in the data curation process who are interested in or need to know specific details on how microdata is curated. This includes Personal Data Controllers, Data Protection Focal Points and Data Providers.

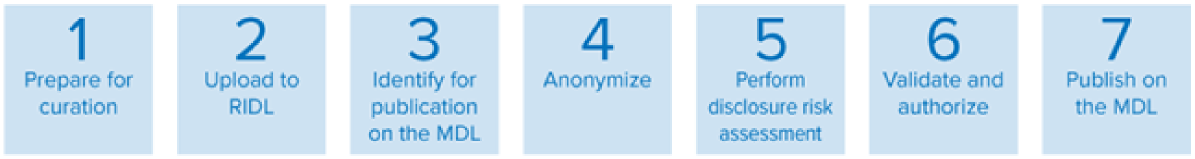
1.3 Scope

This handbook includes technical guidance on the following aspects of data curation within the scope of what is performed by UNHCR under the Curation AI: data checking and preparation, anonymization, disclosure risk assessments and metadata development.

This handbook does not include guidance on data collection, processing and analysis for primary use, even though some of the techniques used may also be transferable to those stages, particularly related to data checking and preparation. Finally, it does not include technical guidance on the use of RIDL or the MDL as this guidance is already available from the developers. They can be found below:

- RIDL: https://im.unhcr.org/ridl/
- MDL: will be added soon because the platform is undergoing an upgrade

In the context of UNHCR's personal microdata of PoCs, the microdata curation process involves seven steps as follows:



This handbook focuses on the technical aspects of steps three to seven, starting after raw or clean data have been downloaded from RIDL to be checked, prepared and anonymized for publication on the MDL.

Different versions of a microdataset are used and/or produced throughout the curation process. They are generally categorized as raw, clean and anonymous. The following table from the Curation AI paragraph 4.3.3 provides a definition of each version and an overview of their purpose.

Version	Definition	Purpose	Location
Raw	Personal or non-personal data in its original format as produced without any edits.	Internal reference and use	RIDL and other systems and tools like proGres v4, KoBo, etc.

Version	Definition	Purpose	Location
Clean	Personal or non-personal data that has been processed, i.e., structured, treated for errors and verified[1]	Internal re-use by UNHCR at the country, regional and/or global levels	RIDL and other systems and tools like proGres v4, KoBo, etc.
Anonymous	Clean personal data that has undergone a technical process of removing or modifying all personal identifiers and codes in such a way that individual data subjects cannot be identified by any means reasonably likely. This process is complemented by other measures to render the risk of re-identification insignificant[2]	Internal re-use by UNHCR at the country, regional and/or global levels Re-use by external actors for programming, analysis, research, advocacy, etc.	RIDL and the MDL

1.4 Contact

If you have questions that are not answered in this handbook, please contact microdata@unhcr.org.

[1]Clean microdata may or may not include direct identifiers. However, clean microdata is still considered personal data until rendered anonymous. Two versions, one with direct identifiers and one without direct identifiers, may need to be catalogued on RIDL. Consider the following example: A country operation needs a version of a clean dataset with direct identifiers for its own use, and it would also like to share a clean version of the dataset without direct identifiers (pseudonymized) for use by other country operations, regional bureaux and/or headquarters. The country operation would catalogue two versions of the clean data on RIDL: one with direct identifiers with more restricted access and one without direct identifiers with less restricted access.

[2]See full definition in the Curation AI

2 Getting started

2.1 File management and naming conventions

To maintain order, structure, consistency and transparency across the organization, standards for naming conventions and file management are important. The following are the standards must be used by Data Curators.

2.1.1 Folder structure

Each dataset that will be curated and published on the MDL must be downloaded from RIDL, checked, prepared, and anonymized on your local computer (i.e., using R as highlighted throughout this Handbook). Not all files associated with the curation will be stored on RIDL including the curation script, the disclosure risk assessment report, the MDL metadata population script and any files that will no longer be needed after the curation is finished. To maintain transparency within the organization, all files associated with this process are stored in a secure repository on the GDS – Data Curation private SharePoint space. The metadata of external datasets that are harvested for the MDL should also be stored in this folder.

Files are organized by region and country where the data was collected. Beyond this, each dataset has a dedicated main folder. The name of this folder is the same as the 'Survey identifier' (see next section). Within this folder, are a set of subfolders organized in a way so that anyone can easily retrace the steps of the curation process. They should be organized into folders using the logic below.

Folder	Content
0_data	All versions of the data. There should be at minimum three subfolders associated with the different versions: raw, clean and anonymous
1_scripts	All scripts used in the curation of the dataset including the script(s) use for data check and preparation, anonymization and disclosure risk assessment script.
2_documentation	The metadata script for the dataset as well as the supporting documents including the questionnaire, report and any other analytical pieces produced (e.g briefs, infographics, etc.).

Folder	Content
3_authorization	The final version of the disclosure risk assessment report and copy of the email from the Personal Data Controller providing or rejecting validation of the anonymization and authorization for release on the MDL[1].

Note that Data Curators are encouraged to use R projects, which facilitate the use of relative paths so that R scripts are easily used by multiple users on different machines. The R project should be stored in the main directory of the dataset establishing it as the root folder. See more details on this in section Software requirements and setup .

2.1.2 Dataset identifier

The dataset identifier (also referred to as a survey identifier by certain DDI compliant catalogues) is a unique code that can be used to identify any file associated with the dataset. For example, it will be the name of the folder on SharePoint where all working files related to the curation are stored. It is also the base for file names (see next section). The dataset identifier must:

- not include any spaces or special characters except for underscores (_);
- only include standard English letters in upper case and numbers;
- separate the different components of the identifier (see below) with underscores[2] (_); and
- not share the same identifiers as another dataset.

The dataset identifier is made up of the following components in the order it is presented below:

- **Source**: The source of the dataset (e.g. UNHCR)
- **Country code**: ISO3 code of the country covered in the field (e.g. AFG for Afghanistan). See list of ISO3 codes here »
- **Year**: year to which data collection started, in the format yyyy (e.g. 2019)
- **Information type**: short description of the information type (e.g. PDM, SENS, etc.).

[1]Note that this practice was changed in September 2022. Prior to that, the authorization emails were stored in an "Authorizations" folder on the GDS-Data Curation SharePoint space and this is where they can be located.

[2]Note that the information in each component should not be separated. For example, UNOCHA should be UNOCHA and not UN_OCHA

- **Other specification (optional)**: if necessary, another field can be added to distinguish different datasets that would have otherwise the same identifier (e.g. a dataset comprised of multiple files, the same survey may be carried out in the same country and year but in different camps or times).

The following is an example identifier for a standardized expanded nutrition survey (SENS) undertaken by **UNHCR in Zimbabwe in 2017** in the camp number 1: `UNHCR_ZWE_2017_SENS_CAMP1.`

An example of file types and their suffixes is shown in the following table:

File type	Suffix
raw data	_data_raw_v0.1
clean data	_data_clean_v1.1
anonymous data	_data_anon_v2.1
questionnaire	_questionnaire
report	_report
infographic	_infographic
map	_map
cleaning script	_script_clean
anonymization script	_script_anon
disclosure risk report	_dra_report

2.1.3 Data file versions

As mentioned under scope, there are three version of a microdataset that are used and/or produced in the curation process. The file version in the RIDL and MDL metadata editor and attached to the file name should be as follows:

File version	Extension
raw	v0.x
clean	v1.x
anonymous[3]	v2.x

So, for example, the first version of the clean data for a socio-economic assessment undertaken by UNHCR in Zimbabwe in 2017 in the camp number 1 would be *`UNHCR_ZWE_2017_SEA_CAMP1_data_clean_v1.1.csv`* and the anonymized version would be *`UNHCR_ZWE_2017_SEA_CAMP1_data_anon_v2.1.csv`*.

[3]Note that following the Curation AI, all anonymous microdata referred to here have been cleaned before anonymized

2.1.4 Dataset title

The dataset title should follow the naming convention used by UNHCR's Operational Data Portal. Survey names in RIDL are preferably in English, however it is acceptable if they are in the language of the report and dataset. In the Microdata Library, the survey name should be in English. Guidance below.

- **Location:** The title of the dataset should always begin with the location at either country or regional level. The location name (followed by a colon) should be added at the beginning of the title. For example: *Uganda: Rhino Settlement Profile - April 2019*, instead of *Settlement Profile - Rhino - April 2019.* In case the dataset refers to multiple countries or a situation, the name of the region or situation could be added, e.g. *Burundi Situation: Population Dashboard - 30 April 2019.*
  Note that the country name should not be included when the dataset is uploaded to the MDL because it is automatically fetched from the metadata and would cause a repetition. In the case however that the dataset on MDL refers to a specific location within a country, the specific location should be included in place of the country name.
- **Title:** The title of the dataset follows the location after a colon and should be descriptive of the type of content (survey, assessment, profiling exercise, etc.). As part of DDI-compliant metadata, and if the dataset has a title in a language different than English, an alternate title can be added in the original language.
- **Date:** The year should be added at the end of the title. The full date should be added when two (or more) surveys, often belonging to a series, have the exact same title. The date, in a [Day] Month Year format should be added at the end of title, e.g. *Lebanon: National Inter-Sector List - May 2019.*
- **Capitalization:** Titles and descriptions should be written in title case. This means only using capital letters for the principal words, e.g. *Zambia: Sector Updates - July 2019.*

Examples:

Zimbabwe: Socio-economic Assessment in Camp 1 – 2017

South Sudan Situation: Population Dashboard - April 2019

2.2 Software requirements and setup

2.2.1 Software requirements

R and R Studio are the software packages used in the examples in this handbook. GDS SDS prefers Data Curators to use R to ensure the curation process is adequately documented, transparent, replicable and to avoid any compatibility issues. For example, data manipulation in MS Excel cannot be tracked because it does not include a script and Stata is a proprietary software not accessible to everyone.

2.2.2 Installing R and R Studio

Install R from https://cloud.r-project.org/ and R Studio from https://www.rstudio.com/products/rstudio/download/#download.

From time to time, you will need to update R. When using Windows, this can be done directly in RStudio using the installr package.

```
#Check R version
sessionInfo() # your current version will display in the first line of the result

# Install and load installr package
install.packages("installr")
library(installr)

# Update R version
updateR()
```

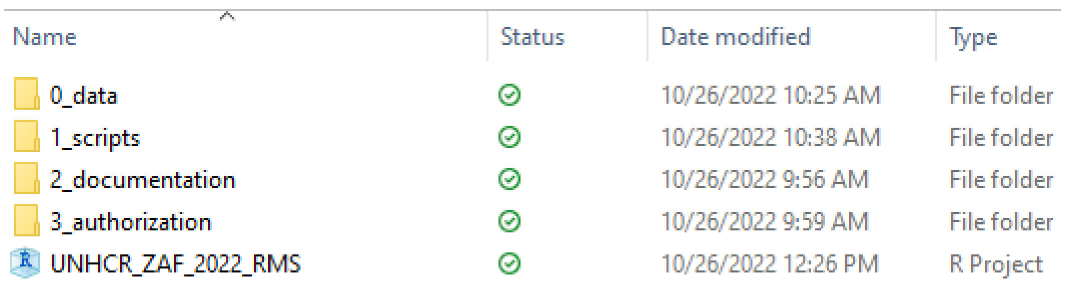
2.2.3 Setting up a R project and creating an R script

The first step is to set up an R project. Using R projects facilitates your work to be easily reproduced by Data Providers and other Data Curators because it facilitates relative paths. See an article about it here.

To set up a project:

a. Firstly, make sure you have created a folder for your dataset that follows the structure and naming conventions outlined in the previous section.
b. Open R Studio and create a new project (File > New Project)
c. Select "Create Project from Existing Directory" and select the main directory of the dataset you created (Note: this will become the root directory from which all your paths will start). See example below:

Now anytime you work on that project, start by opening the project and then create your scripts (New File > R Script) or navigate to them from there.



Name	Status	Date modified	Type
0_data	✓	10/26/2022 10:25 AM	File folder
1_scripts	✓	10/26/2022 10:38 AM	File folder
2_documentation	✓	10/26/2022 9:56 AM	File folder
3_authorization	✓	10/26/2022 9:59 AM	File folder
UNHCR_ZAF_2022_RMS	✓	10/26/2022 12:26 PM	R Project

2.2.4 Installing packages

Once you have installed RStudio, set up your project and started a new script you will also need to install any packages that you will use in your script. For anonymization, you will need the sdcMicro package. You should work directly in the RStudio interface, however the sdcMicro GUI application, which does not require writing R code, may be useful for testing or to those less familiar with R. That said, due to the limited functionality of the GUI application, Data Curators must eventually become familiar with R and interact with the `sdcMicro` package using the R interface.

Installing sdcMicro in R

```
# Install package
install.packages("sdcMicro")

# Load package
library(sdcMicro)

# Load sdcMicro GUI
sdcApp()
```

Several other packages are used in the examples in this handbook for the curation process. They are listed in Annex – R packages. The following provides example code to install and load multiple packages at the same time.

Installing multiple packages in R

```
# Check what packages are installed on your computer
installed.packages()

# List of packages to install and load. Add more to list if needed. This manual assumes th
packages_needed <- c('your_package_name','another_package_name',
                     'and_another_package_name')
```

```
# Run function to install and update packages. If one of the packages shows up as FALSE, t
repos <- c(CRAN = 'https://cran.rstudio.com', CRANextra = 'https://macos.rbind.io')

install.packages(packages_needed[!packages_needed %in% row.names(installed.packages())],
                 dependencies = TRUE,
                 repos = repos)

update.packages(oldPkgs = packages_needed, ask = FALSE)

sapply(packages_needed, require, character.only=TRUE)
```

Note There are many ways to complete a task in R, and a variety of packages that can be used to reach the same objective. The example code provided throughout this handbook may not be / is probably not the only way. For users who prefer different methods, use them. The most important point is that the task is completed, and the result is correct. Suggestions for more efficient methods are most welcome to microdata@unhcr.org.

2.2.5 Importing data

Microdata on RIDL come in a variety of formats but are usually in at least .csv or .xls(x), and sometimes stata (.dta) or spss (.sav).

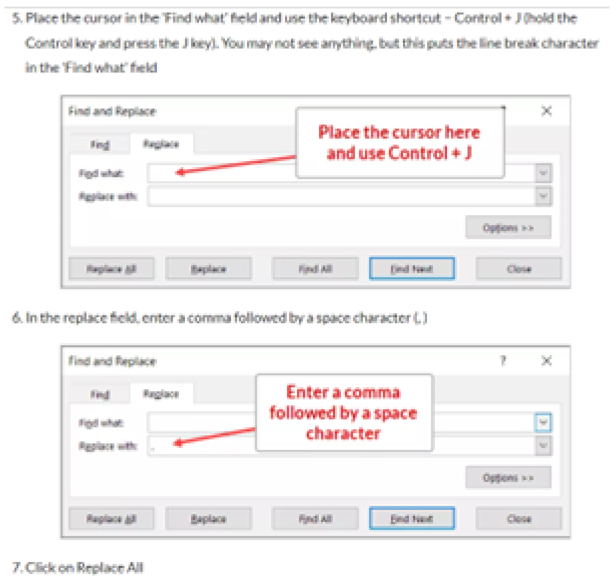
```
dat_raw <- read_csv('0_data/raw/file_name.csv')

# Import .xls or .xlsx using the readxl package
dat_raw <- read_excel('0_data/raw/file_name.xls',sheet='tab_name', na=c('', 'NA', 'missing
dat_raw <- read_excel('0_data/raw/file_name.xlsx',sheet='tab_name')

# Import .dta and .sav files using the haven package
dat_raw <- read_dta('0_data/raw/file_name.dta')
dat_raw <- read_sav('0_data/raw/file_name.sav')
```

Sometimes issues arise when importing the data into R. Below are some common issues and fixes, obviously not an exhaustive list.

Potential issues and solutions when importing data into R

- **A csv file contains accents or special characters such as é, ô, etc.**: Specify the parameter encoding = 'UTF-8' in `read_csv`
- **An Excel file contains records with line breaks**: R will misalign the data, creating new records that are part of another record. Using the readxl package should resolve

this issue, however not always as it depends on the encoding of the source file. The line breaks can be removed directly in Excel using ‘find and replace’ (see below).



5. Place the cursor in the 'Find what' field and use the keyboard shortcut – Control + J (hold the Control key and press the J key). You may not see anything, but this puts the line break character in the 'Find what' field

6. In the replace field, enter a comma followed by a space character (,)

7. Click on Replace All

- **Variable/column types are read incorrectly by R**: By default, R uses the first 1000 observations to ‘guess’ the variable type. If it is an xls(x) or csv file , then `guess_max` option can be used to increase the number of rows used to guess the variable type.

3 Identify data for publication

Any UNHCR personnel who may perform the Data Curator role for a given dataset will need assist in the process of determining whether it is relevant for anonymization and publication on the MDL and ensuring all the information required for the disclosure risk assessment (DRA) is available (see Curation AI section 5.3). This section focuses on these tasks and is slightly different for microdata collected using proGres v4 and outside of proGres v4.

3.1 Data collected using proGres v4

For personal microdata of PoCs collected and managed using proGres v4, the microdata is uploaded to RIDL on an annual basis. The protection team in each operation should meet with the personnel identified to perform the Data Curator role for a given dataset on an annual basis, and address the following:

- Identify if there is any reason not to publish an anonymous version of the data on the MDL (see section 5.3 of the Curation AI).
- Identify if there is any sensitive data or information in the proGres v4 on RIDL.
- Determine the variables and level of detail that need to be preserved in the microdataset.

Note that the initial meeting with a given country operation before the first release of personal microdata of PoCs from proGres v4 will potentially require more time to go over the context, curation objectives and planning, while meetings in the subsequent years may simply aim to determine if the operational context and/or protection risks have changed in a manner that would impact the methods used in microdata curation and the utility of the data.

3.2 Data collected outside of proGres v4

For personal microdata of PoCs collected outside proGres v4, the Data Curator may identify data for curation following several different scenarios, such as:

- After consulting with the Personal Data Controller and/or Data Protection Focal Point, the Data Provider notifies the Data Curator each time they upload personal microdata of PoCs to RIDL and believe it may be suitable for external publication on the MDL.

- UNHCR personnel who regularly perform the Data Curator role, DIMA units and country-level IMOs maintain regular contact and monitor RIDL to identify personal microdata of PoCs that may be suitable for external publication. They may approach the Data Providers at any time.
- A third party may specifically request access to microdata held by UNHCR, and a Data Curator is notified of this request so they can provide support.

A registry of personal microdata of PoCs developed and/or maintained by the Personal Data Controller and Data Providers (see Curation AI paragraph 5.2.3) can serve as a useful tool in identifying microdata collected outside of proGres v4 for curation. The inventory should include some of the following useful details:

- The title of the personal microdata of PoCs collected;
- Location (region and country);
- Year
- If the data were catalogued on RIDL (yes/no);
- link to the dataset on RIDL (if applicable);
- If the data are relevant for anonymization and publication on the MDL (yes/no);
- If they data were curated and published on the MDL (yes/no); and
- Personal Data Controller, Data Provider and Data Curator names.

Below is a very simplified example of a registry. Note that this could be integrated into an existing registry developed by the country operation or regional bureau, simply adding the fields that are helpful to track the storage of microdata on RIDL, anonymization and publication on the MDL.

Table 3.1: Example of registry

Country	Year	Source	Dataset	RIDL	MDL	Data provider	Data Curator
CMR	2016	UNHCR	Minawao: SENS	yes	yes	John Doe	Jane Doe

Datasets that change frequently

For datasets that have frequent updates, such as monitoring exercises repeated on a regular interval (monthly, quarterly, yearly, etc.), panel surveys, etc., each update is treated as a separate (new) dataset. The metadata should specify the coverage of the dataset, allowing Data Users to understand the difference between each version of the dataset.

3.3 Compile information required for the disclosure risk assessment

Once microdata is identified, the Data Curator needs to confirm that all the information required for the disclosure risk assessment (DRA) is available. Most of this information should be in the report (if available) and/or the RIDL metadata, however there are some details that are crucial that may not be available.

The Data Curator can compare what information is available to what is required in the DRA checklist and reach out to the Data Provider to compile any missing or incomplete information as needed. The following is the DRA checklist from Annex D of the Curation AI.

1. What are the potential disclosure risk scenarios – e.g., realistic assumptions about who may be interested in the microdata and for what purpose – and available data and information (both internal and external) covering the same population group that could be linked to this personal microdata (e.g., through the mosaic effect)?
2. Is there any sensitive data or information in the dataset, including sensitive personal data? If yes, what are they? Some commonly found in UNHCR's microdata include, for example:

- geographic location (current and/or former location if displaced population);
- age, gender and diversity considerations such as language, ethnicity, religion;
- occupation, livelihood activity, income level, etc.; and
- variables that are related or can be linked to the sensitive variables.

3. What are the 'key variables' or variables that can be used to indirectly identify a data subject or be linked to another variable that can then be used to identify a data subject? Some commonly found in UNHCR's microdata include, for example:

- geographic location (current and/or former location if displaced population);
- household size, head of household marital status, specific vulnerabilities within the household;
- age, gender and diversity considerations such as language, ethnicity, religion; and
- occupation, livelihood activity, income level, etc.

4. If the microdata is a sample, what is the sample design and weights? Note that if the sample design included stratification, the weights need to be calculated for each stratum.
5. Which cleaning measures were already taken, including the treatment of outliers, grouping/re-grouping of variables, etc.?
6. Was the personal microdata of PoCs collected directly by UNHCR, by a partner on behalf of UNHCR, jointly with a partner, or jointly with another third party?

7. Are additional technical, organizational or legal measures or otherwise binding commitments necessary to be put in place to reduce the risk of disclosure? These measures and commitments are context specific. Some commonly found examples include:

- access control and management
- statistical disclosure control (SDC)
- encryption
- bilateral or multilateral data sharing or collaboration agreements
- Terms of Use
- other contractual or written commitments
- training for personnel
- etc.

3.4 Data from external providers

By default, and when requested by a data provider, the MDL will host metadata on data made available externally by other institutions. Each study will hold a DDI file, but no additional documentation will be hosted in the MDL unless agreed upon by the data owning institution. Data will be hosted on the MDL if it is agreed with the data controlling institution.

Requests to upload metadata from external providers that are hosted externally are evaluated case-by-case, based on the rationale provided in section 3.2 of Administrative Instruction on the Curation of Personal Microdata of Persons of Concern to UNHCR. If data is requested to be made available, the data provider will be requested to meet the conditions outlined in section 5.3.3 of that Administrative Instruction.

When data is not hosted in the MDL, UNHCR's MDL will not handle data requests for external datasets. For institutions that require UNHCR's MDL to do so, the data requests will be handled under the same procedures as UNHCR's datasets.

3.4.1 Procedures for specific platforms

3.4.1.1 OCHA HDX

From UNHCR – MDL to HDX

- HDX has set up an automatic scraper that extracts data from the UNHCR NADA server (external facing platform)

- Custodian of scraper is HDX
- Scraper creates one entry per each dataset to be stored under a UNHCR collection
- Scraper only collects data that is identified in the field 'ID' as beginning with prefix "UNHCR_", as to avoid double harvesting
- In order to maintain versions unified, access to data is handled by redirecting users back to UNHCR request page to download the data.

From HDX to UNHCR – MDL

- UNHCR has written an R script that searches for keywords of HDX datasets using the R client of HDX
- Custodian of script is HCR
- When datasets hit the queries, UNHCR complements the existing metadata available in HDX to make it DDI compliant
- Datasets with created documentation are to be stored under a HDX collection
- In order to maintain versions unified, access to data is handled by redirecting users back to HDX request page to download the data.
- HDX datasets are not stored in UNHCR's internal platform

3.4.1.2 WB Microdata Library

From UNHCR – MDL to WB MDL

- WB uses `NADAR` package to extract metadata at the study and variable level from UNHCR NADA server (external facing platform)
- WB sometimes amends metadata slightly to adhere to internal metadata standards
- Datasets created by UNHCR, i.e. those with the prefix "UNHCR_" in their ID, are to be stored under a UNHCR collection
- In order to maintain versions unified, access to data is handled by redirecting users back to UNHCR request page to download the data.
- Documentation like reports, questionnaires and metadata files are available in both platforms

From WB MDL to UNHCR - MDL

- UNHCR uses keywords to identify suitable datasets for its MDL on WB NADA server

- UNHCR uses MDL package to extract metadata at the study and variable level from WB NADA server (external facing platform)
- In order to maintain versions unified, access to data is handled by redirecting users back to WB request page to download the data.
- Documentation like reports, questionnaires and metadata files are available in both platforms

Part II

Anonymization

4 Anonymize

If the microdata is relevant for publication on the MDL, the Data Curator can start the anonymization process. This firstly involves data checking and preparation followed by the anonymization. This section provides general guidance on the anonymization process, with cross-cutting examples. Additional guidance complementary to this section that covers specific issues for several standard or semi-standard data collection exercises and surveys is covered in Annex – Further guidance on anonymization.

4.1 Data check and preparation

The objective of the data check and preparation is to identify any issues in a dataset (errors and inconsistencies across variables, data incompleteness, etc.) and ensure that:

- the statistics shared in public reports are reproducible; and
- the dataset is ready for re-use by other software packages and users (i.e. variable names and labels are logical, data is in a standard and reusable format, etc.)

A few points to note:

- Ideally any issues in a dataset should have been addressed by the Data Provider as part of the primary data processing and use, and the data shared on RIDL is a clean version. Nonetheless, often the only data available on RIDL is the raw version and/or the clean version may still have some issues that need to be addressed before the data is anonymized and published on the MDL.
- This step may reveal quality issues that would require a reconsideration of the relevance of the publication of the anonymous version of the dataset and/or put into question the results of an already published report or analytical piece. If this is the case, the Data Curator must present their concerns to the Data Provider.
- Data Curators should take a conservative approach and any issue should be fixed only if necessary and only after discussing it with all the Data Providers and receiving their approval.

- If that dataset requires any cleaning, it must be documented, preferably in a R script or other software package, to ensure the replicability of the results. A copy of the raw data (e.g. 0.1) or previous versions of the clean data prior to further cleaning (e.g. 1.1) shared on RIDL should always be preserved.
- This section focuses on the most common issues found in personal microdata of PoCs collected by UNHCR and its partners, however every dataset has its own particularities and may require additional checks.

Data check and preparation checklist

- ☐ Check variable names and labels
- ☐ Check value name and labels
- ☐ Check variable order
- ☐ Check variable types
- ☐ Check data completeness
- ☐ Check for duplicate records
- ☐ Check for missing values
- ☐ Check for excess spacing in string values
- ☐ Check for consistent responses
- ☐ Check for consistent variable relationships
- ☐ Check for outliers
- ☐ Check consistency with primary analysis / report
- ☐ Check consent variable
- ☐ Check and prepare unique ID
- ☐ Save clean version and cleaning script
- ☐ Remove variables that will not be released
- ☐ Prepare sample (if relevant)
- ☐ Prepare weights

4.1.1 Variable names and labels

Variable names (or codes) are the short names used by a software to refer to or call a variable, whereas variable labels are a description of the variable, reflecting the question used in a questionnaire or instructions in a data collection form. At the very least, variable names and labels must be easily interpreted and follow the same style throughout the data. If standards or codebooks are not available from the Data Provider or within UNHCR and/or are not consistent or easily interpreted, the Humanitarian Exchange Language (HXL) should serve as a guide together with the general instructions below.)

The following table highlights how names and variable labels are referred to in R, Stata and Kobo, and any limitations.

Software	Variable name		Variable label	
	How it is called in the software	*Limitations*	*How it is called in the software*	*limitations*
R	name	No spaces Will run into issues if use special characters such as @ or start name with an underscore (_) or number Cannot use words reserved for functions	label	None
Stata	name	No spaces 32 characters First character must be a letter or one of the characters @, #, or $ Cannot use words reserved for functions	label	80-character limit
KoBo	Data Column Name in form creator	No spaces	Question in form creator	No restrictions
	XML values in data download	No limit on # of characters	Label in data download	
		Only letters, numbers, and underscores are allowed		

Software	Variable name		Variable label
		Must start with a letter or an underscore	

4.1.1.1 Variable names

Variable names should be concise, easy-to-read and, where possible, follow standards established or followed by UNHCR and its partners. In general, the following rules should be applied:

- Variable names should be no longer than 32 characters.
- Variable names should only include standard English letters (a,b,c…z), numbers (0,1,…9) and underscore (_).
- Variable names should not include special alphabet letters (æ, ñ, é, etc.), forward/back slashes (/,), periods (.) or any other special characters ($, ' ', *, @, etc.) or spaces.
- Variable names should always start with a letter. Do not start a name with a number or an underscore.
- Variable names should be in lowercase when possible

While there is no style to follow strictly, once you choose a style you should stick with it for all the variables in the dataset. Common styles are underscore (_) separate words (for example: `your_variable_name`) or camel case (for example: `yourVariableName`). Underscore separated lowercase words is the preferred style.

If variable names refer to the questions of a survey, they should reflect the structure of the questionnaire. For example, question 4 of section 7 could be named as `S07_04`. If there are several alternatives in said question and each one is represented in a different column of the dataset, they could be named ordinally as `S07_04_a`.

If the same variable appears in multiple data files, it should have the same name (for example, the household ID may be present in both the household level file and the individual level file). On the other hand, if a variable is named equally across more than one data file but it refers to different information, one of them should be renamed

If the same variable name appears in the same data file more than once, one of the variables should be renamed.

Renaming variables in R

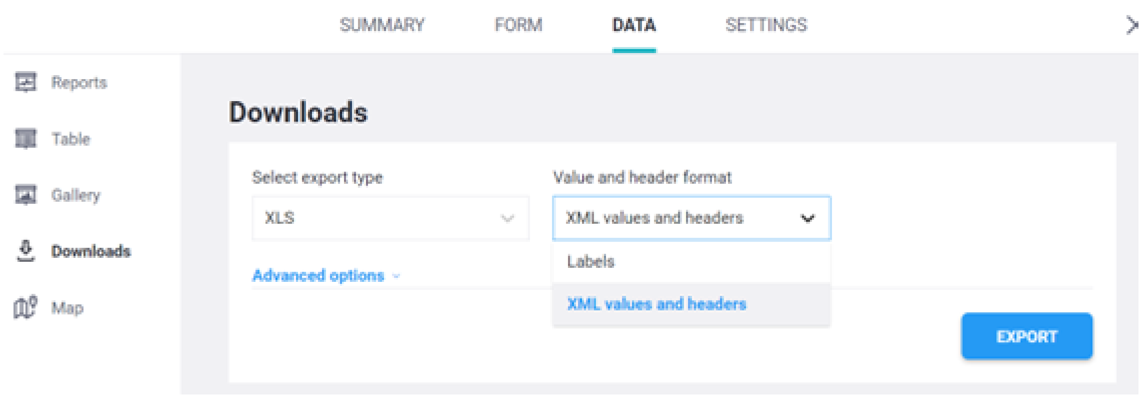
```
#Given the dataset your_dataset - we will use this name in all examples -, you can print t
# print on console all variable names
names(your_dataset)

#You can rename all variable names with the following command (please note that you must p
# Example of converting the names in a data frame with 4 variables
names(your_dataset) <- c("new_var_name1", "new_var_name2", "new_var_name3", "new_var_name4

#If you just need to modify the name of one or few variables, the following code may be us
# Convert the name of one variable
names(your_dataset) [names(your_dataset) == "old_var_name"] <- "new_var_name"

#And alternatively, you can use the dplyr package:
# Rename specific columns
your_dataset <- rename(your_dataset, new_var_name1 = old_var_name1,  new_var_name3 = old_v

#Several other issues in the data can be easily and efficiently managed in R such as conve
# Convert all variable names to lowercase
names(your_dataset) <- tolower(names(your_dataset))

# Change forward slashes in variable names to underscores
names(your_dataset) <- gsub(x = names(your_dataset), pattern = "/", replacement = "_")

# Limit number of characters to <32. Pay attention b/c it may make two or more variables h
names(your_dataset) <- strtrim(names(your_dataset), 31)

#The clean_names function in the janitor package is also a useful tool to clean names, wit
```

Exporting variable names from Kobo

Variables in Kobo have labels (i.e., the question asked in the form) and XML values (i.e., name of the variable). While by default data downloaded from Kobo uses the complete questionnaire questions as column headers, data uploaded to RIDL should have the XML values from Kobo as variable name. To do this:

1. Set the "Value and header format' to 'XML values and headers' when downloading data from Kobo. See screenshot below:

2. Ensure the RIDL record contains the Kobo form, which serves as both the questionnaire and data dictionary with the labels and names (XML values).



4.1.2 Variable order

The order in which variables (columns) appear in a data should be sensible and meaningful. Typically, unique IDs should come first, and other variables should appear in an order that makes it simple to understand the structure of the dataset. If the data was gathered through a survey, the variables order should reflect the questionnaire structure.

Changing variable order in R

```
# using dplyr: specify the variable to move
variable_name <- "your_variable_name"

# Move as first variable of the dataset
your_dataset <- select(your_dataset, variable_name, everything())

# Move as last variable of the dataset
your_dataset <- select(your_dataset, - variable_name, everything())

# Move the variable to the third position
new_variable_position <- 3
your_dataset <- select(your_dataset, 1 : new_variable_position, variable_name, everything(

# Move var_y after var_x using tidyr
your_dataset <- your_dataset %>% relocate("var_y", .after = "var_x")
# Move var_y before var_x
your_dataset <- your_dataset %>% relocate("var_y", .before = "var_x")

#Alternatively, you can specify the exact order of all the variables with the following co
# Write ALL the variables" names in the desired order
# in this simple example, the dataset has only 3 variables
```

```
new_variable_order <- c("var_x", "var_y", "var_z")

# Update the dataset with the new order
your_dataset <- your_dataset[ , new_variable_order]
```

4.1.3 Variable types

4.1.3.1 Variable type

Depending on the software application used to collect and store the data, different types of data may be treated as different variable types. For example, when exporting data to an Excel file from Kobo Toolbox, numbers are often treated as text and as such will be considered characters when brought into R. Depending on the usage of the variable, it may need to be converted before performing a given operation. For example, you cannot use the 'summary' command on a variable of type character, it needs to first be converted to numeric.

The following is a list of the most common variable types or classes in R[1]:

- Numeric – numbers and decimals
- Integer – numeric data without decimals
- Character – text or string
- Factor –Integer values/levels with an associated string which is read as a label
- Logical/boolean – Variable with values TRUE/FALSE

Note If you import a dataset into R, and all the variables are treated as characters (for example), it is not necessary to convert every single variable into their correct format in preparing the clean or anonymized version of the data. Focus on the variables that need to be in the correct format for any manipulation you need to do. For example, if you need to examine a numeric variable, it should be of numeric type. If a variable will be a key variable in `sdcMicro`, it needs to be of type factor (more on that in the next chapter).

Checking and converting variable formats in R

```
# Check the variable format
typeof(your_dataset$your_variable)

#You can convert the format using the as.character, as.factor, as.numeric and as.integer c
```

[1] https://statsandr.com/blog/data-types-in-r/

```
# Convert a single variable to numeric
your_dataset$your_variable <- as.numeric(your_dataset$your_variable)

# Convert a range of variables from your_variable_r to your_variable_y to numeric
dat_clean <- mutate(dat_clean, across(your_variable_r:your_variable_y, as.numeric))
```

4.1.3.2 Multiple choice variables

Some data collection tools, including KoBo Toolbox, include both the short and wide form of a multiple-choice question with multiple select options. For example, the dataset will have one variable with a list of all the responses that the respondent selected, and a binary variable associated with each response option (1 if the response was selected and 0 if it was not). In these cases, both forms of the variable may be preserved in the dataset as each will suit different needs of the users. Special attention must be made in cases where these multiple-choice variables are key variables. Any manipulations to one form of the variable as part of the anonymization process should be reflected on the other form or the other form should be removed.

4.1.3.3 Free text variables

Free text variables are unstructured responses to open-ended questions or requests to specify a response to '`other`' category in multiple choice questions. These variables need to be reviewed because they can be prone to errors or contain sensitive or identifying information.

In general:

- Free text variables with direct identifiers should be removed from clean version of datasets that are not meant to have direct identifiers.
- Free text variables should be removed from anonymous versions of datasets that will be published on the MDL unless the Data Provider requests to keep the free text variables (i.e .because they would be useful to research and do not pose any additional disclosure risk).

4.1.3.4 Free text responses to other in a multiple-choice question

Responses to specify other in a multiple-choice question should be checked to make sure that the response category does not already exist. Consider the example where a respondent is asked how they spent the cash they received from UNHCR and they were provided with the options: food, rent, clothing, utilities, education, livelihood inputs and other. A respondent responded 'other' and when asked to specify what other items they spent they money on, the response was 'rice'. In this example, the response should have been coded as 'food'.

These issues need to be shared with the Data Provider, and the best step for dealing with them determined jointly between the Data Provider and the Data Curator. If it is decided to recode the categories, the following code can be used.

Cleaning up free text responses in other category to multiple choice variables that have existing category

```
# Recode existing category
your_dataset$spent_cash_on_food[your_dataset$spent_cash_on_other_specify=="Rice"] <- "1"
# Replace response to other as 0
your_dataset$spent_cash_on_other[your_dataset$spent_cash_on_other_specify=="Rice"] <- "0"
# Remove free text specifying other
your_dataset$spent_cash_on_other_specify[your_dataset$spent_cash_on_other_specify=="Rice"<

# Recoding several "other" categories to existing category
your_dataset$main_category <- if_else(your_dataset$other_category %in% c("main_category_wi

# For example:
your_dataset$roof_type <- if_else(your_dataset$roof_type_other %in% c("tarpaulin ", "Tarpa
```

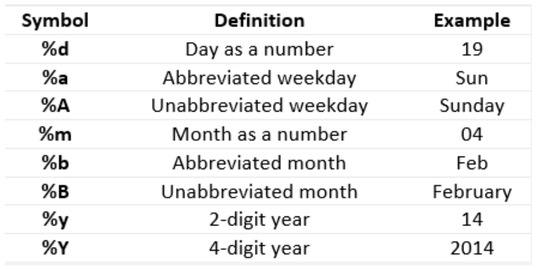
4.1.3.5 Date and time

Date and time formats should be consistent with each other, never use different formats in the same dataset. For maximum compatibility, formats should adhere to the standard ISO 8601[2] or dd-mm-YYYY (i.e. 7th of May of 2021 should be written as 07-05-2021) or YYYY-mm-dd which is the standard for some R packages (see below). In either case, the year should always be four digits and not abbreviated into two.

Formatting dates in R

First, you need to identify the format in which your dates are provided. You can edit the format parameter with the `as.Date` command (in base R) with the codes in the table below:

[2] https://en.wikipedia.org/wiki/ISO_8601



Symbol	Definition	Example
%d	Day as a number	19
%a	Abbreviated weekday	Sun
%A	Unabbreviated weekday	Sunday
%m	Month as a number	04
%b	Abbreviated month	Feb
%B	Unabbreviated month	February
%y	2-digit year	14
%Y	4-digit year	2014

```
# Convert from character to date format using as.Dates
your_dataset$dates <- c("May 27 1984", "July 7 2005")
your_dataset$dates <- as.Date(your_dataset$dates, format = "%B %d %Y")
# Result: [1] "1984-05-27" "2005-07-07"

#Using the lubridate package, dates can be easily converted to date format with the follow
# Convert date from character to date format when the original date is stored as a charact
your_dataset$date <- ymd(your_dataset$date)
# Convert date from character to date format when the original date is stored as a charact
your_dataset$date <- mdy(your_dataset$date)
```

4.1.4 Data completeness

Ensure that there are no gaps or missing information in the data that can be avoided. This will include checking the number of observations against what is reported by the Data Provider and/or in the report as well as comparing the variables in the dataset against the questionnaire or data collection form to make sure that all the expected data is included. Sometimes datasets will include blank records (rows) or variables (columns). These can be removed if they are not meaningful (e.g. system generated variables or kobo metadata, erroneous blank records). They do not need to be removed if they are associated with a real record or variable.

Removing records (rows) and variables (columns) with all missing values in R

```
# remove blank records
your_dataset <- your_dataset[ rowSums(is.na(your_dataset)) < ncol(your_dataset) ,]

#You may also want to remove variables (columns) that have no responses or are blank.
# remove blank variables
your_dataset <- your_dataset[ colSums(is.na(your_dataset)) < nrow(your_dataset) ,]

# remove blank records using the janitor package
```

```
your_dataset %>%
     remove_empty(which="rows")
# remove blank variables
your_dataset %>%
     remove_empty(which="cols")
```

4.1.5 Duplicate records

Datasets should not include any duplicate records. The simplest way to look for duplicates in a dataset is to identify the variable (e.g. unique ID) or combination of variables (e.g. name, age, sex, geographic location) that must be unique to each observation and then verify that there are no observations sharing the same values.

Looking for duplicates in R

```
#Given the dataset your_dataset, you can remove duplicates based of a single variable usin
your_dataset[!duplicated(your_dataset$your_variable), ]

#using dplyr:
your_dataset %>%
   distinct(your_variable,keep_all=TRUE)

#More elaborate step-by-step guide to find duplicates in your_dataset:
# Create data frame with list of IDs and the number of times they occurred
n_occur <- data.frame(table(your_dataset$id))
# Identify which IDs occurred more than once
n_occur[n_occur$Freq > 1,]
# Return the list of IDs that occurred more than once
your_dataset[your_dataset$id %in% n_occur$Var1[n_occur$Freq > 1],]

#To check for duplicates using a combination of variables, first create a unique variable
# Create a new variable (combo_var) that combines multiple variables
your_dataset$combo_var <- paste0(as.character(your_dataset$var_x),"_", as.character(your_d
```

4.1.6 Missing values

Datasets and variables will often contain missing (or blank) values for several different reasons. It should be clear what exactly these values mean. For example, do they indicate that the question is non-applicable, is it a non-response, a true missing value or a value of 0[3]. If it is not clear, the Data Provider should be consulted to clarify the meaning of the missing or blank values.

Working with missing values can be a bit tricky depending on how they appear in the dataset. Often, missing values are left blank, and R treats them as `NA`. Other times they may have specific codes defined during data collection (i.e. -9999, 9999, etc.). This should be recoded to missing as to ensure summary statistics are not being affected by the codification of these values.

Dealing with missing values in R

```
#How data can be imported in R converting blank values to NA and replacing 0 and 9999 valu

# Import data from a csv file encoded in UTF-8. The values specified in na.strings will be
your_dataset <- read.csv("your_dataset_file_name.csv", na.strings=c("","NA"), encoding = "

# Replace all 0 and 9999 values with NA in cases they were incorrectly coded
your_dataset$your_variable[your_dataset$your_variable ==0] <- NA
your_dataset$your_variable[your_dataset$your_variable ==9999] <- NA

#Note that R does not always show NAs in a frequency table. To force R to show the NAs, ad
# Frequency table without NAs
table(your_dataset$your_variable)

# Frequency table with NAs
table(your_dataset$your_variable, useNA = "always")

#Finally, to export data replacing the NA with blank values, the following code can be use

# Save dataset as csv file, setting missing values to blank values
write.csv(your_dataset, "your_dataset_file_name_new.csv", row.names = FALSE, na="", fileEn
```

The proportion of allowed missing values (not to be confused with non-applicable, which is not a missing value) will depend on the impact on data quality and is something that is determined by the Data Curator in consultation with the Data Provider.

Check for missing values in R

[3] In some cases, a 0 is used in place of a blank value. It needs to be determined if this is a true value of 0 or a blank value. 0 should only be used for a 0 value.

```
You can perform common checks on missing values with the following code:
# calculate the total proportion of missing observations in the dataset
mean(is.na(your_dataset))

# calculate the proportions of missing observations for each variable in the dataset and s
sort(colMeans(is.na(your_dataset)), decreasing = TRUE)

# create crosstab of two variables to see how missing values are distributed
table(your_dataset$your_variable1, your_dataset$your_variable2, useNA = "always")

# Delete variables with all NAs
your_dataset =your_dataset[,colSums(is.na(your_dataset)) <nrow(your_dataset)]

# Calculate the proportion of missing observations and create a chart showing most common
summary(aggr(your_dataset, sortVar=TRUE))$combinations

#Finally the naniar package can be useful to visualize missing values. Read more here: htt
```

4.1.7 Excess spacing in string values

String values should never start or end with a space or contain repeated spaces. This issue may be difficult to spot since empty spaces are invisible, but they cause problems during analysis since they are recognized by statistical software. For example, "`your string`" and '`your  string`' may look the same when visually exploring the dataset, but when you create cross tabulations for example, they are different. It may be useful to check this early on because it can assist when checking consistent responses, for example.

The different possible cases are:

- Leading space: '`your string`'
- Trailing space: '`your string`'
- Repeated space between words: '`your    string`'
- Any combination of the above: '`your    string`'

Most softwares have a trimming function to quickly remove trailing, leading and repeated spaces.

```
#The package stringr provides a function called str_squish to trim all spaces in a string.

# trim all the values in a variable
your_dataset$your_variable <- str_squish(your_dataset$your_variable)

#The package janitor provides a function to remove all trailing spaces in variable names:

your_dataset <- clean_names(your_dataset)
```

4.1.8 Consistent responses

Checks should be done on the consistency in responses to key variables, other variables of interest to the Data Users, and/or any variables that the Data Provider advised should be checked. In some cases, inconsistencies may not be discovered until detailed analysis is performed, and the outputs seem 'off'. In any case, the Data Curator should always take a step back and look at the dataset large to see if any obvious issues can at least be spotted. Below are some common examples.

4.1.8.1 Inconsistent data categories

Data category inconsistencies occur when a variable has two or more distinct values representing the same category. This can be caused by synonymous words, typos, formatting, etc. This includes ensuring that, for example, responses to multiple choice questions fall into one of the valid answer options in the questionnaire or data collection form.

The following table highlights an example of inconsistent use of country names, Swaziland and Eswatini refer to the same country as does Côte d'Ivoire and Ivory Coast

id	country_origin
1	Swaziland
2	Côte d'Ivoire
3	Eswatini
4	Ivory Coast

Check for variable inconsistencies in R

```
#You can identify inconsistencies and fix them with the following code:
# Print all existing values for the variable to spot any issue
cat(sort(unique(your_dataset$your_variable)), sep="\n")
# Create a frequency table to see how many records fall in each category
table(your_dataset$your_variable)

#If necessary, replace the values
your_dataset$your_variable[your_dataset$your_variable =="Ivory Coast"] <- "Côte d'Ivoire"
your_dataset$your_variable[your_dataset$your_variable =="Swaziland"] <- "Eswatini"
```

4.1.8.2 Consistent use of capitalization in string values

Most statistical software applications are case sensitive, meaning they consider differently two strings with different capitalization. See the example frequency table below where `Bas Sassandra` is repeated because one of the spellings has lowercase for the second word in the name of the district.

```
# A tibble: 6 x 2
  province      population
  <chr>              <dbl>
1 Abidjan               70
2 Bas sassandra          2
3 Bas Sassandra          7
4 Cavally                4
5 Guemon                 1
6 Tonkpi                 7
```

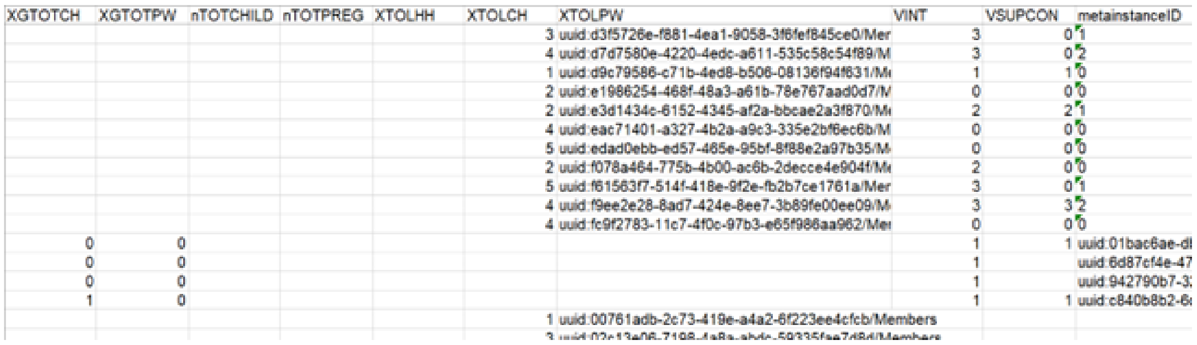
Check and fix inconsistent capitalization in R

```
#Consider the dataset your_dataset and the variable your_variable. A simple way to observe
# Create a frequency table
table(your_dataset$your_variable)

# Replace error with simple find and replace
your_dataset[your_dataset$your_variable=="your response"] <- "Your Response"

#Alternatively, you can capitalize all values of a variable in the same manner. This is pa
# Change all to UPPER CASE
your_dataset$your_variable <- toupper(your_dataset$your_variable)
```

```r
# Change all to lower case
your_dataset$your_variable <- tolower(your_dataset$your_variable)
# Change all to Title Case
your_dataset$your_variable <- tools::toTitleCase(tolower(your_dataset$your_variable))
```

4.1.8.3 Data table alignment

Another cause for inconsistencies is a data entry mistake where variable values are inserted in the wrong column. For example, the values of the variable gender might have been pasted in the column corresponding to the age variable. If gender is a string variable (`female`/`male`) this would be easy to notice, but if the variable is numeric (`1`/`2`) it could be harder to spot. This mistake might affect several variables and multiple observations. Sometimes this can be flagged by simply observing the data frame. See an example below, where it seems at some point in the data manipulation process the data were shifted, as can be seen since the identifier variable (`metainstanceID`) is assigned to another column (`XTOLPW`) .



XGTOTCH	XGTOTPW	nTOTCHILD	nTOTPREG	XTOLHH	XTOLCH	XTOLPW	VINT	VSUPCON	metainstanceID
					3	uuid:d3f5726e-f881-4ea1-9058-3f6fef845ce0/Mer	3	0	1
					4	uuid:d7d7580e-4220-4edc-a611-535c58c54f89/M	3	0	2
					1	uuid:d9c79586-c71b-4ed8-b506-08136f94f631/Me	1	1	0
					2	uuid:e1986254-468f-48a3-a61b-78e767aad0d7/M	0	0	0
					2	uuid:e3d1434c-6152-4345-af2a-bbcae2a3f870/Me	2	2	1
					4	uuid:eac71401-a327-4b2a-a9c3-335e2bf6ec6b/M	0	0	0
					5	uuid:edad0ebb-ed57-465e-95bf-8f88e2a97b35/M	0	0	0
					2	uuid:f078a464-775b-4b00-ac6b-2decce4e904f/Me	2	0	0
					5	uuid:f61563f7-514f-418e-9f2e-fb2b7ce1761a/Mer	3	0	1
					4	uuid:f9ee2e28-8ad7-424e-8ee7-3b89fe00ee09/M	3	3	2
					4	uuid:fc9f2783-11c7-4f0c-97b3-e65f986aa962/Mer	0	0	0
0	0						1	1	uuid:01bac6ae-d
0	0						1		uuid:6d87cf4e-47
0	0						1		uuid:942790b7-3
1	0						1	1	uuid:c840b8b2-6
					1	uuid:00761adb-2c73-419e-a4a2-6f223ee4cfcb/Members			
					3	uuid:02c13e06-7198-4a8a-abdc-59335fae7d8d/Members			

4.1.9 Consistent variable relationships

The Data Curator should identify the main relationship between variables and check for their consistency. Some of the most common ones to look out for include:

- The age of the respondent should be within the age range acceptable to respond to the interview, and what makes sense with the data (i.e. can a 2-year old child be working?).
- The sum of members in a household broken down by age and gender should be equal to a variable on total household size

- The number of pregnant and lactating women should not be larger than the number of women in the household
- Geographic locations should fall under the administrative units where they are located
- Some questions may only be relevant based on the answer to previous questions (i.e. only for recipients of a given programme, only for households with children, etc.)

It is important to check and make sure the patterns are consistent, and if in doubt contact the Data Provider. Consider the following example:

id	camp	status	programme	cash_received	parcels_received
1	**Orange**	**Refugee**	**WFP food parcel**	**250’000**	
2	Orange	Refugee	WFP food parcel		1
3	Pineapple	Refugee	WFP food parcel		1
4	**Lemon**	**IDP**	**CBI**	**250’000**	
5	Pineapple	Refugee	CBI	250’000	
6	Pineapple	Refugee	WFP food parcel		1
7	**Lemon**	**IDP**	**CBI**	**250**	

This table includes a couple of obvious errors and one suspicious issue highlighted in blue:

- Line 1 - The refugee household part of the WFP food parcel programme reported receiving *cash*. This is an obvious error, and the Data Provider should be consulted to see if they can confirm what the correct number of parcels should be or if the data on `parcel_received` will be a loss.
- Line 5 – The response here is suspicious because is the only household in Pineapple camp that was part of the CBI programme and the only refugee in the dataset that received cash (all others were IDPs). The Data Provider should be consulted just to confirm this is correct.
- Line 7 – The data on the amount of cash received appears to be an error when compared with all other data, most probably the Data Collector didn’t type out all the zeros. This should be confirmed with the Data Provider.

It will not be possible for a curator to fix and spot all inconsistencies within a dataset. Therefore, we recommend to select some variables in consultation with the Data Provider, that could have a large impact on utility and focus on those.

Check and fix inconsistencies in variable relationships in R

```
#A simple way to observe variable relationships is to use crosstabs. This can be done very
# create a crosstab
table(your_dataset$programme,your_dataset$cash_received)
table(your_dataset$programme,your_dataset$parcels_received)

# Print the affected records
your_dataset[your_dataset$programme=="WFP food parcel" & !is.na(your_dataset$cash_received

# Suppress incorrect value
your_dataset$cash_received[your_dataset$programme=="WFP food parcel" &
                            !is.na(your_dataset$cash_received)] <- NA

#Say the Data Provider confirmed that record 1 received 1 food parcel
# Fill missing value
your_dataset$parcels_received[your_dataset$id==1] <- 1
# Check work
your_dataset[1,]

#Say the Data Provider confirmed that 250 reported by record 7 is a typo
# Replace incorrect numeric value
your_dataset$cash_received[your_dataset$cash_received==250] <- 250000

#Finally, say the Data Provider confirmed that line 5 is correct. As such, nothing needs t
```

4.1.10 Outliers

Outliers are data points that differ significantly from other observations. They may be due to variability in the observation or may indicate an error. Using charts (histograms, scatter plots and box plots) and crosstabs is a good way to spot outliers. The treatment of outliers should be decided in consultation with the Data Provider.

Note: Outliers may be treated differently in the cleaning and anonymization process. In the cleaning process, they should only be removed or modified if the value is wrong. Another approach may be to flag the outliers instead of fixing them. In the case that the outlier is in a key variable and is not wrong (i.e. a true outlier), then this record will likely increase the risk of disclosure of that particular Data Subject and may need to be treated to lessen the risk. This is addressed in the next chapter.

Consider the simplified example below with three variables, `id`, `income_source`, `income_1month`

```
# A tibble: 15 x 3
      id income_source                 income_1month
   <dbl> <chr>                                 <dbl>
 1     1 Factory employee                      20000
 2     2 Factory employee                       6000
 3     3 Factory employee                       7000
 4     4 Factory employee                       6000
 5     5 Factory employee                       5000
 6     6 Factory employee                       5000
 7     7 Factory employee                       6000
 8     8 Seller / Commercial activity          37000
 9     9 Seller / Commercial activity          39000
10    10 Seller / Commercial activity         400000
11    11 Agriculture / Sale of crops            5000
12    12 Agriculture / Sale of crops            4500
13    13 Agriculture / Sale of crops            4000
14    14 Agriculture / Sale of crops            4700
15    15 Livestock / Sale of animals            7500
```

At first glance, it looks as if there may be a couple outliers, particularly record 1 and 10. The following outlines various ways to explore the outliers in R.

```
#Using the example above, and for the dataset "outliers", simple way to start is to look a

# Summary of variable
summary(outliers$income_1month)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4000    5000    6000   37113   13750  400000
```

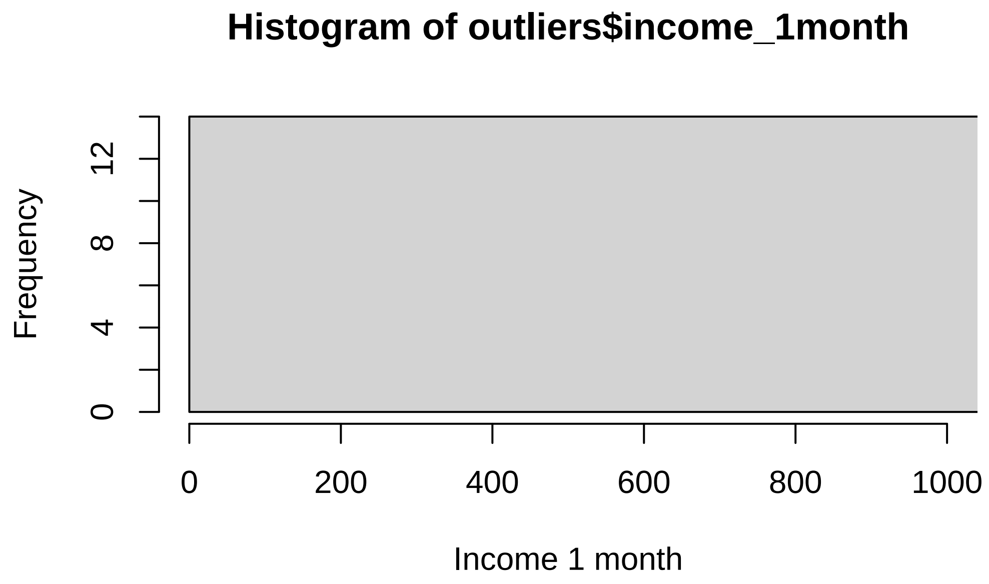
```
# Summary of variable with a condition
summary(outliers$income_1month[outliers$income_source=="Factory employee"])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5000    5500    6000    7857    6500   20000
```

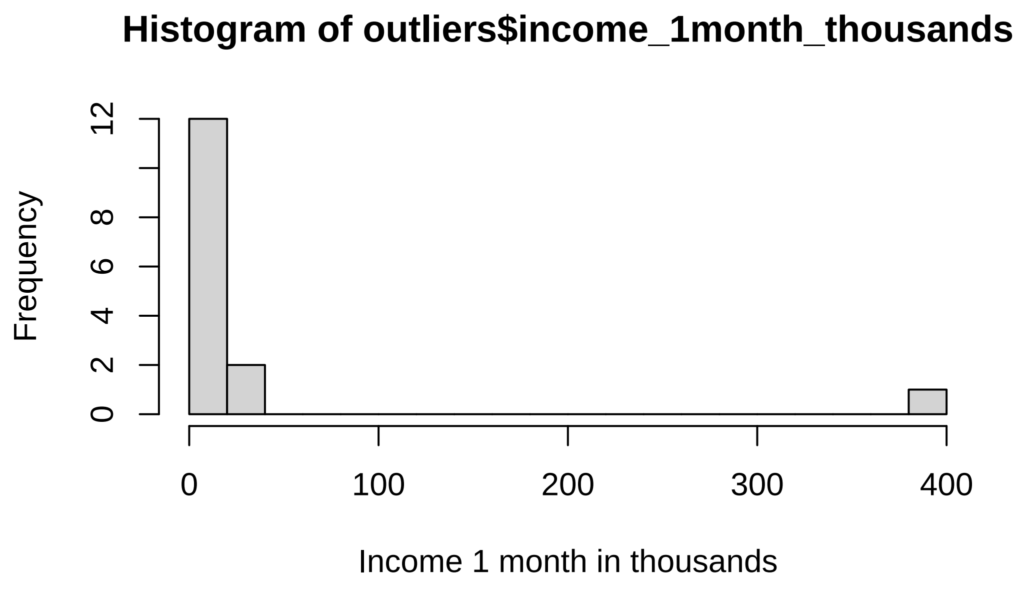
```r
# Calculate outlier cutoff of 1.5x the interquartile range
review_outliers <- 1.5*IQR(outliers$income_1month)
print(review_outliers)
```

```
[1] 13125
```

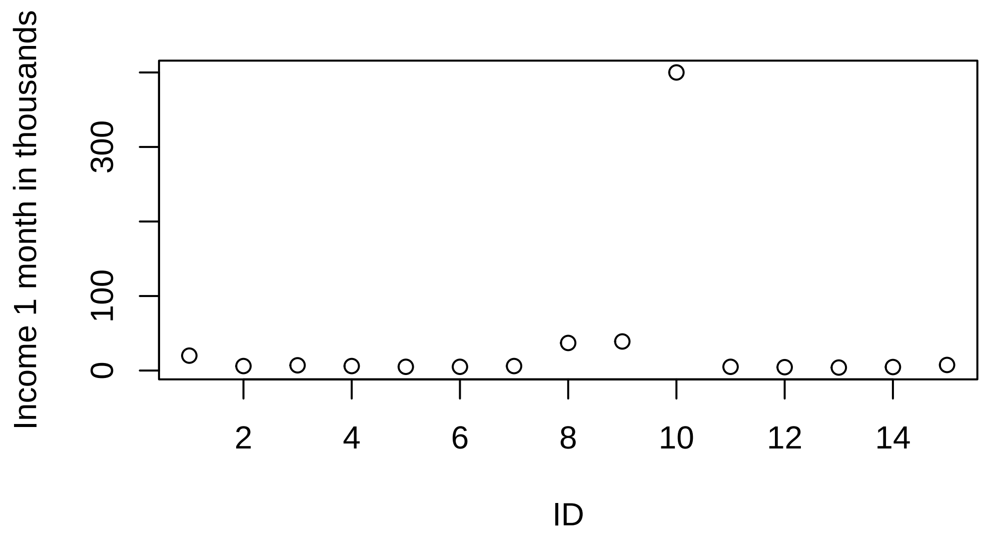
```r
# Create a histogram of income_1month
hist(outliers$income_1month, xlab="Income 1 month", xlim=c(0, 1000))
```



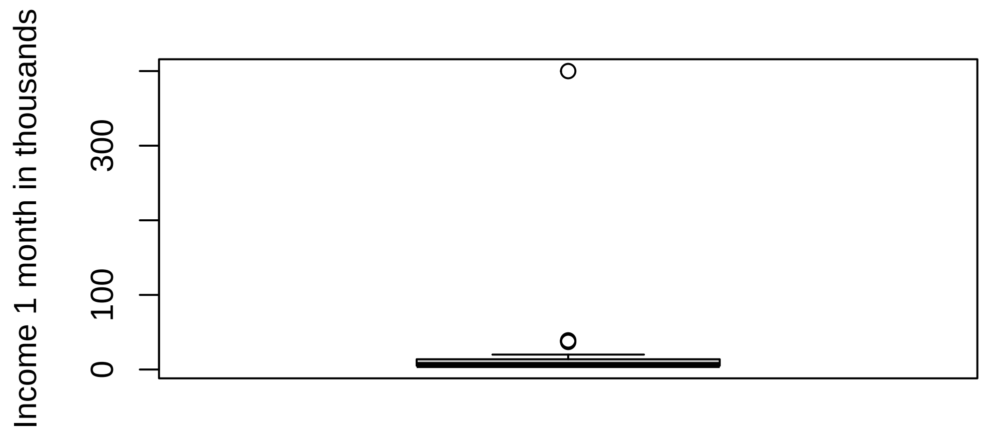
```r
# Create a new variable for months in thousands to make it easier to read on chart and rec
outliers$income_1month_thousands <- outliers$income_1month/1000
hist(outliers$income_1month_thousands,xlab="Income 1 month in thousands", breaks = 25)
```



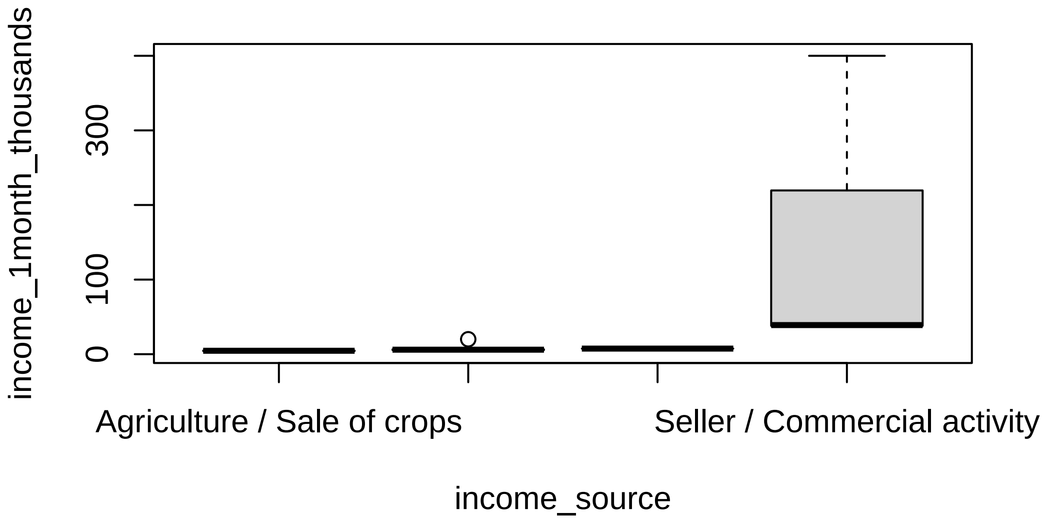
```
# Create a scatter plot of income_1month
plot(outliers$id, outliers$income_1month_thousands, xlab="ID", ylab="Income 1 month in tho
```



```
# Create boxplot of income_1month
boxplot(outliers$income_1month_thousands, ylab="Income 1 month in thousands")
```



```
# Create boxplot of the income_1month variable split by income_source
boxplot(income_1month_thousands~income_source, data = outliers)
```



4.1.11 Consistency primary analysis / report

If an analytical piece or report has been produced with the dataset, consistency between the raw/clean version of the data and the analysis/report should be checked[4]. Sharing a dataset that differs with the report will later result in confusion and requests for clarification from the final users. The following is list of the minimum issues to check before moving on to the next stage of curation.

- The total number of observations in the dataset is the same as reported in the analysis/report.
- The main key indicators presented in the analysis/report can be correctly calculated from the dataset.
- Most of the variables used in the analysis/report or in the data collection form/questionnaire are in the dataset. Note that some variables may be missing if there were issues in their collection, however in general anything reported against

[4]Note that the anonymous version (next section) and the raw/clean versions will certainly have slight differences when manipulations are made as part of the anonymization process. This can be explained in the metadata. The point here is that the data shared by the Data Provider (the base used prior to anonymization) should at least be consistent with any published works).

should at least be present in the clean version of the dataset even if some need to be removed for the anonymous version (discussed in next chapter).

Any discrepancies should be discussed with the Data Provider and documented in the metadata on RIDL.

4.1.12 Consent

In some cases, the data will include a variable associated with a question on the "consent" of the Data Subject. The subject of what this "consent" entails will depend on the data exercise and should be available from the Data Provider. The following provides an overview of some of the typical cases, and how they should be treated AFTER consultation with the Data Provider:

- Normally if the data subject did not provide consent, the data will still include a record for that data subject but all variables after the consent variable will be left blank or `NA`. This can help facilitate calculating the response rate. If the record only includes whether or not they provided consent, and information related to the strata (i.e. camp, province, district, etc.), if applicable, and a unique ID or index, then the variable can be preserved.
- In cases where a respondent did not provide consent and some personal information is still present, all this personal information must be removed.
- In cases where the Data Provider advises that the entire record is removed, the entire record should be removed.

Removing data from records that did not provide consent in R

```
# Remove information from your_variable for records when your_consent_variable is "No"
your_dataset$your_variable[your_dataset$your_consent_variable=="No"]<- NA

#example of removing a record whose consent is "No" using dplyr
your_dataset <- your_dataset %>% filter(!your_consent_variable=="No")

# Drop records with missing consent using dplyr
your_dataset <- your_dataset %>%
  drop_na(your_consent_variable)

# Keep only records where response to consent is 1 or NA using subset function
your_dataset <- subset(your_dataset,your_dataset$your_consent_variable == 1 | is.na(your_d

# Drop records with no consent which was reported as either "No" or "No consent" in this e
```

```
your_dataset <- your_dataset %>%
  filter(!your_consent_variable %in% c("No", "No consent"))
```

4.1.13 Unique ID

Every data table should have a unique ID, either auto-generated by the data collection software or manually generated by the Data Provider. If this is not already present, it should be created for the clean version of the data table so that reference can be made to the data table. A simple $1 : n$ can be created.

Create a unique ID

```
# Assign a unique ID using numbers 1:n and place it at the beginning of the data table
your_dataset$unique_id <- 1:nrow(your_dataset)
your_dataset <- select(your_dataset, unique_id, everything())
```

In addition to the original unique ID, a pseudoID should be created that can be used to re-link the clean version of the data with the eventual anonymous version (because the original unique ID will be removed as part of the anonymization). Before it is necessary to randomize data rows because their original order may be used to guess some anonymized values. For example, household data may have been recorded by a particular geographical order, or the household members could have been interviewed from the oldest to the youngest.

Create pseudo unique ID

```
# randomly shuffle data
set.seed(42)
rows <- sample(nrow(your_dataset))
your_dataset <- your_dataset[rows, ]

# assign new id, ranging 1 to n, and position it before the original id
your_dataset$pseudo_unique_id <- 1:nrow(your_dataset)
your_dataset <- your_dataset %>% relocate("pseudo_unique_id", .before = "unique_id")
# Be mindful that this code will generate a new pseudo ID every time you run it. If you ar
```

4.1.14 Prepare weights

To calculate the risk of re-identification of data subjects, the weight of a given data subject needs to be adequately estimated. If the data are from a sample, for example, the weight of any given record is much larger than that from a full enumeration or census where every single data subject is found within the dataset. Design weights (also referred to as base weights or raw weights) in sample data are the number of units in a population that each unit in the sample represents (i.e. if each unit in a sample represents five units in the population, the design weight is 5).

Note While weights may not be used for analysis of a self-weighted sample as each record equally represents its profile in the population of interest, they still need to be calculated for SDC to property calculate the risk of re-identification of data subjects in a dataset. This is because each record only represents a certain proportion of the overall population, which reduces the risk of re-identification as compared to a full enumeration or census.

For example, the weight of each record in a full enumeration or census would be equal to `1` (meaning the probability of selection is 1/1) whereas the raw weight of each record in a simple random sample of 50 individuals out of a total of 200 individuals would be 4 (meaning the probability of selection is 1/4).

`sdcMicro` uses the universal weight `1` if no weight is provided, equating data to a full enumeration or census. As such, full enumeration or census data do not need weights before going into `sdcMicro`. If the microdata is from probabilistic sample surveys, each observation should have a corresponding weight, calculated by Data Provider based on the sampling methods used to design the study. Unfortunately, weights are not always provided in the data, in which case they should be requested to the Data Provider. In the simplest sampling designs, weights can be added to sample data using the following formulas.

In `sdcMicro`, normalized weights cannot be used.

4.1.14.1 Simple Random Sampling

In the case of simple random sampling, where each respondent has an equal probability of being included in the sample, the design weight is calculated for all records in the dataset using the following formula:

$$\text{survey weight} = \frac{\text{Total number of potential data subjects in sample frame}}{\text{Total number of surveyed data subjects}}$$

Create sample weights in R for a sample frame of 6000 subjects

```
# create design weight variable
your_dataset$weight <- 6000/nrow(your_dataset)
```

4.1.14.2 Stratified Random Sampling

In a stratified random sampling, where the population of interest is divided in distinct strata and samples are drawn separately from each stratum, the design weight needs to be calculated for each stratum using the following formula:

$$\text{survey weight} = \frac{\text{Total number of pot. data subjects in strata 1's sample frame}}{\text{Total number of surveyed data subjects in strata 1}}$$

Adding weights to a dataset that used stratified random sampling in R

```
#The following is an example for creating a design weight variable called weight in your_d

# create design weight variable for each stratum
your_dataset <- your_dataset %>%
  mutate(survey_weight = case_when(camp == "camp1"~ 1000/300,
                                   camp == "camp2"~ 1000/700))
```

4.1.15 Save clean version and cleaning script

If any modifications/cleaning measures were taken in the previous steps, a clean version (or new clean version) of the file can be saved and uploaded to RIDL. If the initial data was the raw version, this should be saved as version 1.1. If the initial version was a clean version (e.g. version 1.1), this one should be stored as a new version (e.g. version 1.2). Additionally, the R script used to create this version should be saved and uploaded to RIDL with the clean version. This is most easily done in the r studio interface by going to File > Save As.

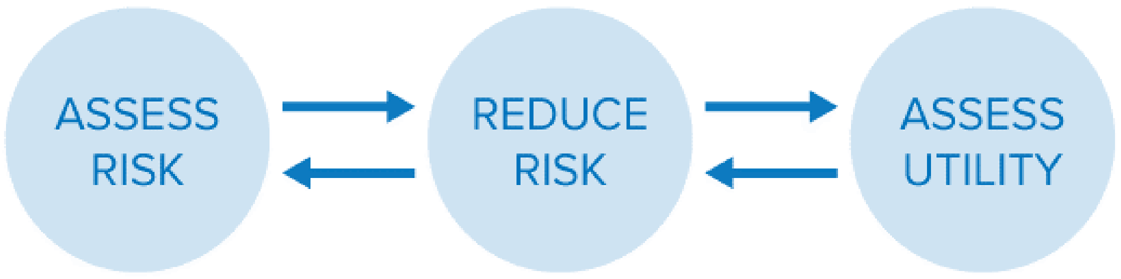
Exporting data in R

```
#Using the haven package, you can export your_clean_data with the following code:
# Export csv
write.csv(your_clean_data,"0_data/clean/your_clean_data.csv",row.names=FALSE)
```

```
#Export dta
write.dta(your_clean_data,"0_data/clean/your_clean_data.dta", version = 11, label = attr(d

#Export RDS, R"s data format
saveRDS(your_clean_data, "0_data/clean/your_clean_data.RDS")
```

4.2 Anonymization

Anonymization is an iterative process involving three stages: **assess the risk of re-identification, reduce the risk of re-identification (i.e., anonymize), and assess the utility of the data (or loss of information)**. While each stage is distinct in its method and purpose and the process always starts with assessing the risk, the process involves moving back and forth between the stages until the data has been effectively rendered anonymous, as shown in the figure below.



UNHCR uses Statistical Disclosure Control (SDC) methods to assess and reduce the risk of re-identification of data subject in each dataset. This handbook does not include theory around anonymization, statistical disclosure control and other methods of minimizing disclosure risk as several available resources from the humanitarian and development field on the topic already exist. For theory, the two that are most often referred to by the Data Curation Team include:

- Statistical Disclosure Control for Microdata: Theory developed by the World Bank
- An Introduction to Disclosure Risk Assessment developed by the UN OCHA Centre for Humanitarian Data

Every dataset has its own characteristics, so this guidance should be considered indicative only. Some cases will require more steps than others.

The main part of this section covers the classic example of a household or individual level microdataset(that is not hierarchical). Annex – Further guidance on anonymization provides more details on the following:

- Household surveys
- Cash-based intervention (CBI) post-distribution monitoring (PDM) surveys
- proGres data
- SENS survey data
- WASH survey data

This section is written under the assumption that the data has already been checked for (see Section 4.1) and all the preliminary information needed for anonymization has been compiled see Compile Info

Anonymization checklist

- ☐ Remove variables that will not be published
- ☐ Prepare sample (if relevant)
- ☐ Assess risk
- ☐ Reduce risk
- ☐ Assess utility
- ☐ Add/modify labels
- ☐ Save anonymous version and script

4.2.1 Create new dataframe

If working continuously in r, create a new data frame labelled anonymous (e.g. `data_anonymous`) that is distinct from the clean version (e.g. `data_clean`) so that comparisons can be made later as part of the utility analysis.

Create a new dataframe in R

```
#Create anonymous version
your_anonymous_data <- your_clean_data
```

4.2.2 Remove variables that will not be published

Any variables that were preserved with the clean version of the data but will not be released with the public version should be removed at this stage.

Variables to be removed from publicly released version of data

- **Direct identifiers** – If some were kept in the clean data, such as unique IDs to link to other datasets, they should now be removed.
- **Sensitive variables** - It may also be necessary to remove variables that are considered too sensitive for a release in a given context, for example, responses to questions about subjects of violence, religion or ethnicity. You should always consult with the Data Provider about the sensitivity of variables, always and especially when in doubt.
- **Key variables that cannot be released** - Sometimes, during the anonymization process, you may realize that it would not be possible to sufficiently anonymize certain key variables. For example, a key variable with a lot of variation such as the name of the camp/village in a dataset that includes over 20 different camps and villages. In this case, you may decide to try to go through the SDC process with this variable as a key variable, but eventually remove it if it is impossible to anonymize without too much utility loss. More about key variables later in this chapter.
- **Free text variables** - Free text variables usually are not useful for the final user analysis and may contain sensitive information. See Free text variables

The following code can be used to remove variables from a dataset.

```r
#This first example is a simple way to remove a single variable.
# Remove a single variable from a dataframe
your_dataset$your_variable <- NULL

#This second example is a simple way to remove multiple variables at the same time
# Remove multiple variables from a dataframe
your_dataset [c("your_variable1", "your_variable2", "your_variable3")] <- NULL

#This third example is useful when developing a template to use for multiple datasets, bec
# Identify direct identifiers
direct_identifiers <- c("enumerator_name", "address_household_number", "gpsCoordinates", "

# Check if direct identifier variable exists and remove them
for (i in direct_identifiers){
 if(i %in% colnames(your_dataset))
{ your_dataset = your_dataset %>% select(-any_of(i))}
}
```

4.2.3 Prepare sample (if relevant)

If the data is census data (or a complete enumeration), it may be necessary to draw a sample of the data for release as opposed to the entire dataset. If the number of observations is small and the re-identification risk low, you can consider releasing the entire dataset. Otherwise, pull a random sample, preferably stratified to increase precision, from the dataset.

Extracting a sample in R

```
#This first example pulls a simple random sample of 20% from your_dataset.

# get sample of identifiers
id_var <- "id"
sample_percentage <- 0.2
sample_ids <- sample(your_dataset[[id_var]], size = round(sample_percentage * nrow(your_da

# create weight and sample
your_dataset$weight <- nrow(your_dataset) / length(sample_ids)
your_dataset <- your_dataset[ your_dataset[[id_var]] %in% sample_ids ,]

#This second example pulls a stratified simple random sample using the "camp" as strata. N

# install the "dplyr" package if not already
install.packages("dplyr")

#Draw a proportionately stratified sample, in this case 20%.
set.seed(1)
your_dataset_sample = your_dataset %>%
  group_by(camp) %>%
  slice_sample(prop = 0.2)

#Draw a sample where the same number of observation is pulled from each strata, in this ca
set.seed(1)
your_dataset_sample = your_dataset %>%
  group_by(camp) %>%
  slice_sample(n = 50)
```

Note that once the sample is drawn, the weights then need to be added. See Prepare weights.

4.2.4 Assess risk

The risk of re-identification is assessed on the key variables[5] and quantitatively analyzed using two main SDC indicators: ***k*-anonymity** and **individual risk**.

k-anonymity

A risk measure that is based on the calculation of the number of observations in a sample sharing the same combination of categorical key variables, where k is the number of data subjects sharing the same combination. An observation violates k - anonymity if the sample frequency count fk is smaller than the specified threshold k , while a dataset satisfies k-anonymity if no observations violate the specified k-anonymity threshold. The risk measure is the number of observations that violates k-anonymity for a certain value of k, which can be calculated using `sdcMicro`.

Individual risk

The probability of correct re-identification of any of the observations in the dataset.

Risk thresholds

UNHCR"s thresholds for SDC for anonymizing the personal microdata of PoCs are presented in the table below.

Indicator	Threshold	Description
k-anonymity	3	Unique combination of key variables is shared by at least three (3) observations in the data
individual risk	<15%	No one observation has greater than 15% chance of re-identification

Certain cases, outlined below, may require the thresholds to be adjusted. The final decision about how to treat these two cases rests with the Personal Data Controller, with support from the Data Provider, Data Protection Focal Point and Data Curator, and in consultation with the Chief DPO as necessary.

- For certain personal microdata of PoCs, it may be determined that these thresholds need to be adjusted to allow for lower risk (i.e., in the case of sensitive microdata). If the risk of disclosure needs to be lower, k-anonymity could be higher and/or the individual risk threshold lower.

[5]Key variables, also called "quasi-identifiers", are a set of variables that, in combination, can be linked to external information to re-identify respondents in a dataset

For personal microdata of PoCs derived from a full enumeration or census (i.e., not a sample), personal microdata that is not fully anonymized according to the thresholds listed above, or personal microdata that may be particularly sensitive or that contains sensitive data or information, UNHCR needs to either consider stricter SDC parameters (as outlined under point a) or a stricter Terms of Use (e.g., release data on the MDL as a Licensed Use File rather than a Public Use File).

4.2.4.1 Define and prepare key variables

SDC methods used in the anonymization process to reduce the risk of re-identifying data subjects focus on the analysis of key variables, namely factor (categorical) key variables or continuous variables that are converted to factor (categorical) as part of re-coding, and manipulations to the data focus on these key variables. Before diving into `sdcMicro`, it"s important to already know which variables will be key variables.

Key variables are context specific, and therefore need to be identified for each dataset based on the possible disclosure scenarios. Annex - Key and sensitive variables provides a more comprehensive list of common key and sensitive variables as well as appropriate anonymization techniques. Annex – includes a list of key variables for specific datasets and surveys.

Only factor key variables are used as part of the risk assessment calculation. It is possible to input continuous (numeric) key variables into `sdcMicro`, however they are not used as part of the analysis of *k*-anonymity, individual or global risk. As such, discrete numeric variables or numeric variables with a finite number of responses such as age, household size, number of children in household, etc. should be treated as factor in `sdcMicro`. The following code provides an example of how to convert key variables to factor.

```
#Convert categorical key variables to factor
your_dataset$your_variable <- as.factor(your_dataset$your_variable)
```

Additionally, only missing values coded as `NA` are read as '`NA`' by `sdcMicro` so it is important that missing values in a key variable are coded as such and not, for example, 0, 99, etc. If this was not already done as part of the data check and preparation process, it should be done now.

```
# Recode missing value code 99 to NA
your_dataset$your_variable[your_dataset$your_variable== 99] <- NA
```

4.2.4.2 Define ghost variables

Variables that might be linked to key variables are considered 'ghost variables' in SDC. Some examples below:

- the name of a refugee camp is linked to the region where it is located;
- variables associated with the disaggregation of total number of household members (e.g. by gender, age, etc.) are ghost variables of total number of household members; and
- income per capita is a ghost variable of household size, etc.

The response to a ghost variable should be removed from the dataset if the response to their associated key variable is removed, and this key variable can be recalculated from the ghost variable.

Defining ghost variables to be used by `sdcMicro` *in R*

```r
# Set up blank list that will be used to define ghost variables
selectedGhostVars <- list()

# Each linkage is a list, with the first element the key variable and the second element t
selectedGhostVars [[1]] <- list()
selectedGhostVars [[1]][[1]] <- "total_hh_size"
selectedGhostVars [[1]][[2]] <- c("hh_mem_0-17_yrs","hh_mem_18_59_yrs","hh_mem_60_yrs")

# If the dataset has more than one set of ghost variables, this can simply be added using
selectedGhostVars [[1]] <- list()
selectedGhostVars [[1]][[1]] <- "total_hh_size"
selectedGhostVars [[1]][[2]] <- c("hh_mem_0-17_yrs", "hh_mem_18_59_yrs",
                                  "hh_mem_60_yrs")
selectedGhostVars [[2]] <- list()
selectedGhostVars [[2]][[1]] <- "region"
selectedGhostVars [[2]][[2]] <- c("camp")
```

4.2.4.3 Create the SDC risk assessment (or SDC problem)

To initialize the SDC risk assessment (or SDC problem in `sdcMicro`), the SDC object (`sdcObject`) needs to first be created. The `sdcObject` encompasses all parameters that feed into the risk assessment. The following is a list of these parameters and sample code to create the `sdcObject`.

Parameter	Description
dat	Name of dataset (table) to go through SDC.
keyVars	Categorical (factor) key variables
numVars	Numeric key variables.
ghostVars	Ghost variables as described in previous section.
weightVar	Weight variable. If this is left blank, SDC assumes it is a census and not a sample dataset.
hhID	Hierarchical identifier, if present. For example, if the table contains individual-level data that is linked to household-level data, then select the variable containing the unique ID for the household.
strataVar	The variable that defines the strata (either strata used for sampling or for analysis) used in perturbative methods like PRAM and Microaggregation. Don't need to specify if it is not relevant.
pramVars	Variables on which you want to apply the PRAM (post randomization method).
excludeVars	Variables to be excluded from the SDC, and as a result dropped from the dataset. It is basically another way to remove variables from the dataset.
seed	The seed makes it possible to reproduce the same results if you apply probabilistic anonymization methods. Please note that the seed should not be shared externally since it may make it possible to reverse the anonymization process. Always choose a random number, never use the default value.
randomizedRecords	If the records should be randomized. Can be TRUE or FALSE.
alpha	Alpha is the weight with which missing values contribute to the computation of key variables frequencies when calculating k-anonymity. Leave at default value of 1.

Setting up an `sdcObj` in `sdcMicro`

```
#The following is an example of setting up a sdcObject with ghost variables selectedGhostV

# Set up sdcMicro object
sdcObj <- createSdcObj(dat = your_dataset,
                       keyVars = c("your_variable1", "your_variable2"),
                       ghostVars = selectedGhostVars,
                       numVar = c("your_variable3"),
                       weightVar = c("weight"),
                       pramVars = NULL,
                       hhId = NULL,
                       strataVar = c("camp"),
                       excludeVars = NULL,
                       seed = 346,
                       randomizeRecords = FALSE,
                       alpha=c(1))
```

Note: In cases where the sdc parameter is not relevant, simply define it as empty or `NULL` as is shown in the above example. In parameters that call for a variable or a number of variables, the codes should be written as a list using the following syntax: `c('variable_name','variable_name')`.

4.2.4.4 Interpret the SDC risk assessment

The SDC risk assessment includes three different groups of quantitative information:

- information on the categorical key variables;
- individual and global risk for categorical key variables; and
- information on k-anonymity.

4.2.4.4.1 General overview and k-anonymity

The first step may be to get a general overview of the risk assessment, and k-anonymity.

```
#Considering your sdcObj, the following code can be used to get a general overview of risk

# Summary of risk assessment results
sdcObj
```

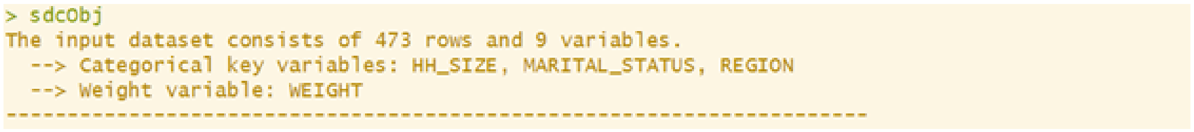
Example SDC risk assessment output

The following is the SDC risk assessment from a household survey for a example dataset built-in the sdcMicro library. Note that this is an example of the problem BEFORE any modifications are made to the data. Reprinting the sdcObj after modifications are made will allow to compare the same information before the modifications and after. Each description is associated with the number in the screenshot of the R output.

```
library(sdcMicro)

data_sdcObj <- readRDS("data_sdcObj.RDS")

###Create object
```

```
sdcObj <- createSdcObj(dat=data_sdcObj,
                          keyVars=c("HH_SIZE", "MARITAL_STATUS", "REGION"),
                          weightVar=c("WEIGHT"),
                          hhId=NULL,
                          strataVar=NULL,
                          pramVars=NULL,
                          excludeVars=NULL,
                          seed=60,
                          randomizeRecords=FALSE,
                          alpha=c(1))
###Risk assessment results
sdcObj
```

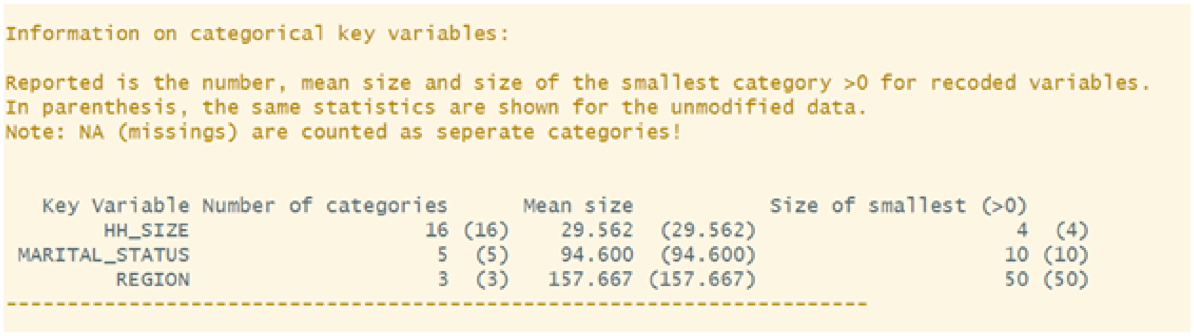
Section 1:



```
> sdcObj
The input dataset consists of 473 rows and 9 variables.
  --> Categorical key variables: HH_SIZE, MARITAL_STATUS, REGION
  --> Weight variable: WEIGHT
---------------------------------------------------------------------------
```

Brief overview of the dataset that is part of the problem including number of records and variables, list of categorical key and numeric key variables and weight variable. If ghost variables were included, they would be listed here.

Section 2:

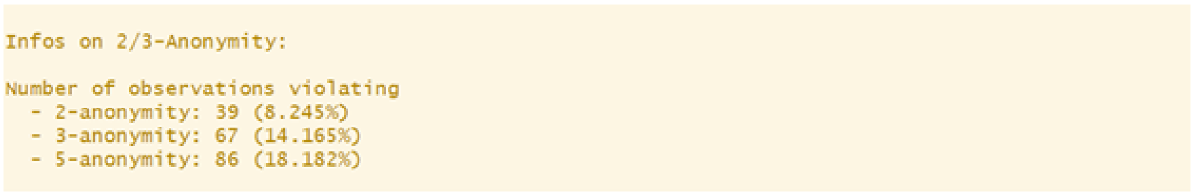


```
Information on categorical key variables:

Reported is the number, mean size and size of the smallest category >0 for recoded variables.
In parenthesis, the same statistics are shown for the unmodified data.
Note: NA (missings) are counted as seperate categories!


   Key Variable Number of categories      Mean size           Size of smallest (>0)
        HH_SIZE                   16 (16)    29.562  (29.562)                      4  (4)
 MARITAL_STATUS                    5  (5)    94.600  (94.600)                     10 (10)
         REGION                    3  (3)   157.667 (157.667)                     50 (50)
---------------------------------------------------------------------------
```

Basic description of the categorical key variables, including the number of categories going into the problem. Note that is example is showing unmodified data. If you run the code after the data is modified.

Section 3:

The number of observations violating 2/3/5-anonymity. To interpret this, you can say that 49.45% of observations in the original dataset do not share the same responses to all categorical



```
Infos on 2/3-Anonymity:

Number of observations violating
  - 2-anonymity: 39 (8.245%)
  - 3-anonymity: 67 (14.165%)
  - 5-anonymity: 86 (18.182%)
```

key variables with at least 2 other records.

4.2.4.4.2 Individual and global risk

Carefully examining global and individual risk can help to identify which observations are risky and further which key variable are causing observations to be more/less risky. You can start by printing out more details on the individual and global risk.

```
#Individual and global risk in R
#Considering your sdcObj, the following code can be used to print out details on individua

# Details on the global and individual risk
sdcObj@risk

#If you need further details, the following codes can help to view more about the specific

# Show the 10 largest individual risk measurements
sdcObj@risk$individual %>% as.data.frame() %>% arrange(desc(risk)) %>% head(10)

# List of observations violating 3-anonymity
kanon3 <- your_dataset[sdcObj@risk$individual[,2] < 3,sdcObj@keyVars]
kanon3 %>% print(n=Inf)

#List of observations with individual risk > 0.15 or 15%
indv15 <- your_dataset[sdcObj@risk$individual[,"risk"] > 0.15,sdcObj@keyVars]
indv15 %>% print(n=Inf)
```

The result first show the measure of global risk followed by a table of individual risk for each observation. In the table on individual risk, there are three columns defined below:

- f_k= Sample frequency of the keys variables for all observations, whereas those with the same keys variables have the same frequency. If equal to 10, for example, there are 10 observations in the dataset with the same combination of key variables. If equal to 1, for example, this observation has a unique combination of key variables and is 'sample unique.' By definition, f_k is the same for all observations with a given key variable combination.

- F_k = Population frequency of a combination of key variables, which is the estimated number of respondents in the population with that number of similar combinations of key variables. If the microdata is a sample and not a census, F_k is the sum of the sample weights of all observations with the same key variable combination. Hence, like f_k is also the same for all observations with a given key variable combination.
- risk or r_k = Individual risk or the probability of disclosure for the observations. As with f_k and F_k, it is the same for all observations sharing the same pattern of values of key variables[6]

As the end goal is to have 0 observations violating 3-anonymity and 0% of records with individual risk >0.15 or 15%, it is helpful to print out a list of risky observation in R to understand if there are any trends around which key variables are increasing the level of risk.

4.2.5 Reduce risk

If the risk assessment shows that the risk is not too high and k-anonymity is three (or the threshold set by the Data Provider and Personal Data Controller if different), then no manipulation needs to be done to the dataset. In this case, the Data Curator can go straight to section Save anonymous version, codebook and anonymization script.

If the risk assessment shows that the risk is too high, **then the data cannot be released without applying SDC methods.**

The goal of SDC is to preserve as much of the data's original content as possible, while reducing the risk of re-identifying data subjects.

Bear in mind that:

1. Highly perturbative methods including PRAM, microaggregation, noise addition, shuffle and rank swapping should be avoided because they can highly distort the data based on the parameters used.

2. Anonymization is an iterative process, which often requires starting over after attempting to anonymize a dataset once, twice, etc. and not being satisfied with the result and loss in utility. Using R with a save script helps to facilitate retracing your steps and also consult other Data Curators for advice.

[6]From the SDC Practice Guide

An overview of anonymization methods is available in the table below. Complementary information can be found in the SDC Practice Guide. The technical part of this section focuses on the most used non-perturbative methods. For further support, reach out to microdata@unhcr.org.

4.2.5.1 Summary of anonymization methods

Key variable type	Method	Classification	Explanation and Example
Categorical	Recoding	non-perturbative, deterministic	Recoding consists of grouping the values of categorical variables into a new variable. This is commonly used to regroup categorical variables with few responses into a larger group. For example, recoding responses with less than XX records to another like category or a group 'other'.
	Local suppression to achieve k-anonymity	non-perturbative, deterministic	Local suppression set the values of the key variables to missing (suppress) until the desired level of k-anonymity is reached (at least a 3-anonymity level).
	Suppression of linked (ghost) variables	non-perturbative, deterministic	Suppressing ghost variables involves suppressing the response in a variable that is linked to the response in a key variable to missing if the response to the key variable is also suppressed (or set to missing). This step is necessary when there is a relationship between variables that could be used to reconstruct a suppressed value.
	Suppression of values with high risk	non-perturbative, deterministic	This method consists of suppressing (or setting to missing) the values of key variables for observations that surpass a certain risk threshold. It is most useful when you are aware of few records with high risk. The result depends on the selected key variable. You may need to repeat the suppression selecting other variables so that you remove the values of more than one key variable, or you may undo it and do it again selecting another key to see if you obtain a better result.

Key variable type	Method	Classification	Explanation and Example
	PRAM (Post RAndomization Method)	perturbative, probabilistic	PRAM consists of randomly changing the values of categorical variables according to an invariant probability transition matrix. This method is especially useful when a dataset contains many key variables, and local suppression and recoding would lead to a significant information loss. One characteristic of the PRAM method implemented by `sdcMicro` is that univariate tabulations remain the same before and after anonymization for all variables that have been subjected to the method. This kind of implementation is called invariant PRAM. If a particular cross tabulation must be maintained, you can specify a strata argument.
Continuous	Rounding	perturbative, deterministic	Rounding is simply rounding a number up so that it is less precise. It is used to prevent exact matching of continuous key variables with external data sources. In addition, rounding can be used to reduce the level of detail in the data. For example, removing decimal figures of income variable or rounding income variable to the nearest thousand.
	Top/bottom recoding	perturbative, deterministic	The purpose of top/bottom coding is to remove outliers (observations that lie at an abnormal distance from other values in a random sample from a population) from numeric key variables. Top and bottom coding are similar to recoding, but instead of recoding all values, only the top and/or bottom values of the distribution or categories are recoded. Top and bottom coding is especially useful if the bulk of the values lies in the center with only few observations outside (outliers). Some numeric values common to UNHCR data are related to age, expenditure and income, and usually you will have to top code high values, as there will often be a few observations above certain thresholds, typically at the tails of the distribution.
	Microaggregation	perturbative, probabilistic	Microaggregation is most suitable for continuous variables but can also be used for categorical. It works by grouping records that are homogenous with respect to the values of certain key variables. The values of selected variables are then replaced with a common value, e.g. the mean of that group.

Key variable type	Method	Classification	Explanation and Example
	Noise addition	perturbative, probabilistic	In noise addition, values are added to or subtracted from the original values of a variable. This prevents exact matching with an external file based on an exact value of a continuous variable.
	Rank swapping	Perturbative, probabilistic	Rank swapping is a type of data swapping used for ordinal continuous variables. Values of the variable are first ordered, and values are swapped with other values that are similar (in the same neighborhood).
	Shuffle	perturbative, probabilistic	Shuffling is similar to swapping but uses an underlying regression model to determine which variables are swapped.

Many of the commonly used SDC methods can be done without using the `sdcMicro` package (i.e. using other R packages before entering into the `sdcObj`) or using specific `sdcMicro` functions inside the `sdcObj`. The Data Curator can choose the method most suitable, as long as utility analysis performed afterwards, and the methods are documented for transparency. The example codes below show examples both inside and outside of `sdcMicro`. Note that in the case that manipulations are performed outside of `sdcMicro`:

a) The risk analysis will only look at the final risk and not the true before and after.

b) The utility analysis in the `sdcMicro` generated report is not relevant because it is comparing a dataset that was already manipulated with the final anonymized version of the dataset. As such, the utility analysis will need to refer to the clean and anonymized data respectively.

4.2.5.2 Categorical key variable methods

4.2.5.2.1 Recoding

Recoding key variables is a great way to reduce the percentage of records violating 3-anonymity without having to suppress too many values. At the same time, the following two points need to always be considered when recoding:

- **Recoding should not cause too much loss of utility**. For example, if indicators need to be calculated for children in school age (elementary and secondary), the recoding of this variable should be according to school year divisions, which would allow for calculations to be made i.e. 0-5, 6-11, 12-17. Instead, grouping in five-year intervals (0-5, 5-10, 10-15, 15-20) would greatly decrease the data utility for these users.

- **If the variable can be recalculated using other variables, this anonymization method is not useful.** A particular case is household size: in cases that there is an individual-level dataset, it can be recalculated simply by counting the number of individuals. Another example is, in the case of stratified sampling, the variable containing the strata (for example camp or camp zone). All observations in the same strata will have the same weight, so it would be possible to guess the strata just by looking at the unique weight value.

Recoding variables using the `sdcMicro` package

```
#Given your sdcObj and the key variables, province_district and total_hh_mem, the followin

# Recode locations with few observations (Location 1, Location 4 and Location 5) into new
sdcObj <- groupAndRename(obj=sdcObj, var="province_district",
                         before=c("Location 1","Location 4","Location 5"),
                         after=c("Other"))
# Recode total number of household members into new distinct groups 1, 2, 3-4, 5-6, >6
sdcObj <- groupAndRename(obj=sdcObj, var="total_hh_mem", before=c("3", "4"),
                         after=c("3-4"), addNA=FALSE)
sdcObj <- groupAndRename(obj=sdcObj, var="total_hh_mem", before=c("5","6"),
                         after=c("5-6"), addNA=FALSE)
sdcObj <- groupAndRename(obj=sdcObj, var="total_hh_mem",
                         before=c("7","8","9","12"), after=c(">6"), addNA=FALSE)

#See risk summary after recoding
sdcObj

#Recoding variables outside sdcMicro

#Given the key variable age, there are a couple different methods to recode the age into g

# Recode age into 7 distinct groups method 1
your_dataset$age <- cut(your_dataset$age,
                         breaks=c(0, 20, 25, 30, 35, 40, 45, Inf),
                         labels=c("15-19", "20-24", "25-29",
                                  "30-34", "35-39", "40-44", "45-49"))

# Recode age into 7 distinct groups method 2 using dplyr
library(dplyr)
your_dataset$age <- case_when(your_dataset$age %in% 0:19 ~ 1,
                        your_dataset$age %in% 20:24 ~ 2,
                        your_dataset$age %in% 25:29 ~ 3,
                        your_dataset$age %in% 30:34 ~ 4,
                        your_dataset$age %in% 35:39 ~ 5,
                        your_dataset$age %in% 40:44 ~ 6,
                        your_dataset$age %in% 45:49 ~ 7)
# add labels to the codes
val_labels(your_dataset$age) <- c("15-19" = 1, "20-24" = 2, "25-29" = 3,
                                  "30-34" = 4, "35-39" = 5, "40-44" = 6,
                                  "45-49" = 7)

#After this, the data can run through sdcMicro.
```

4.2.5.2.2 Local suppression

Local suppression is used to **suppress** observations until *k*-anonymity is reached.

In cooperation with the Data Provider, key variables can be ranked by Data Curators in order of importance, meaning that some key variables will be preserved as much as possible compared to others during the anonymization process. So, for example, the location of a household may be more important than the gender of the head of household which may be considered more important than the age of the head of household. Please note however that using an importance vector will lead to a higher total number of suppressed values in the less important variables. Remember to indicate any ghost variables associated with a key variable before applying local suppression. See section Define ghost variables.

Local suppression in sdcMicro

```
#The following code locally suppresses to reach 3-anonymity. The importance variables are

# Local suppression to obtain 3-anonymity
sdcObj <- kAnon(sdcObj, importance=c(1,2,8,3,4,7,6,5), combs=NULL, k=c(3))

# See risk summary after local suppression
sdcObj

#localSupp function can be used to perform local suppression on a single key variable. Thi
# Local suppression to reach 15% individual risk
sdcObj <- localSupp(sdcObj, threshold = 0.15, your_key_variable)
```

```
# See risk summary after local suppression
sdcObj

# Undo the last local suppression - note this only undoes the last
sdcObj = undolast(sdcObj)
```

#### Numeric key variable methods {#sec-numerickeyvar}

Numeric key variables that cannot be considered factor are not used to calculate *k*-anonymity or individual risk. This increases the importance for direct communication with the Data Provider on the disclosure risk given the level of detail in numeric key variables. Two common methods for decreasing the level of detail in numeric key variables are rounding and top/bottom coding.

4.2.5.2.3 Rounding

```
# example using  sdcMicro
sdcObj@manipKeyVars$your_variable <- round(sdcObj@manipKeyVars$your_variable)

# example not using sdcMicro
your_dataset$your_variable <- round(your_dataset$your_variable)
```

##### Top/bottom coding {#sec-5topcode}

Top/bottom coding variables in R and sdcMicro

```
# example using  sdcMicro
sdcObj <- topBotCoding(sdcObj, value=500000,
                       replacement=1000, column="your_variable")
# example not using sdcMicro
your_dataset$your_variable[your_dataset$your_variable >= 2] <- 2
```

4.2.5.3 Removing records

In some cases, entire records may need to be suppressed either because they are too risky (e.g. the individual risk for a particular record is very high) or they are linked to another variable that can be used to re-calculate a key variable that was manipulated (i.e. suppressed, top coded, etc.). Consider the following two examples:

1. In a hierarchical dataset that includes two tables (one with the household at the subject and the other with the household members as the data subject). If all household members are interviewed, the household size can be calculated from the household member table. In this case, if there are manipulations to a variable that provides the size of the household in the household table, the responses to the variable can be recalculated based on the household member table by simply counting the number of members per household. One option to manage this situation, you could firstly top code the household size in the household data table, then randomly remove individuals from the household member table whose households exceed the threshold.
2. Data were gathered using stratified sampling and the strata correspond with different refugee camps and all observations in the same camp have the same survey weight. If the some of the observations in the camp variable are removed through local suppression, it is still easy to guess the camp through the survey weight. Since the survey weight cannot and should not be removed, the first solution would be to prioritize the camp variable when performing local suppression (see Local suppression) and the second solution would be to remove the records where the camp was suppressed from the anonymous version of the data table (if avoiding suppression is not possible).

Note that suppressing records is not possible using any functions of the `sdcMicro` package.

Supressing records in R

```
#Given key variable camp:

#example of removing records outside of sdcMicro where the variable camp is blank
your_dataset <- your_dataset[!is.na(your_dataset$camp),]

#example of removing row number 134 outside of sdcMicro
your_dataset <- your_dataset[-c(134),]

#example of removing a record whose hh_id is "ABC" using dplyr package
your_dataset <- your_dataset %>% filter(!hh_id=="ABC")

# Example of removing records using subset function with two conditions
your_dataset <- subset(your_dataset,your_dataset$your_variable == "ABC" | your_dataset$you
```

4.2.5.4 Reassess risk

After performing relevant SDC methods, the Data Curator re-evaluates the disclosure risk on the anonymized data using the same methods described in section Interpret the SDC risk assessment. The `sdcObject` summary will display k-anonymity before and after any manipulations that were done inside of the `sdcObject`. As previously stated, it is important to remember that if any manipulations were done outside of the sdcObject, they will not be factored in this analysis.

4.2.5.5 Exporting data from SDC object

Once you are satisfied with an anonymized version of the data, it needs to be exported out of the `sdcObject` to assess utility and for the last fixes before release. This can be done using the following code.

You can export the anonymized version of the data, ***`your_anonymous_data`****, using the following*

```
# export anonymized data file and save to disk (extract sdcObj)
your_anonymous_data <- extractManipData(sdcObj)
```

4.2.6 Assess utility and utility loss

Once the risk thresholds are met, a comparison between the clean and anonymous version needs to be made focusing on re-calculating the main indicators and cross-tabulations of interest for analysis and research. If the difference is statistically significant, the SDC method will need to be repeated until the utility is preserved. If the utility cannot be preserved due to information loss in the anonymization process, then the microdata may not be relevant for external publication on the MDL.

The focus here is on those variables that may have been modified, e.g. the key variables. In datasets where entire records were suppressed, all variables in the dataset were modified and any of potential interest to Data Users need to be reviewed.

Utility checks greatly depend on the kind of dataset and the needs of the final user. Common utility checks are:

1. **Comparing published reports** - If reports based on the dataset have been published, you should recalculate the main statistics on the anonymized dataset and verify that very similar results are obtained.
2. **Categorical key variables and variables of interest -** Compare the distribution of the original dataset against the anonymized one. To establish if any differences are statistically significant, you can compare the confidence intervals on the proportions.

3. **Continuous key variables and variables of interest** – Compare the same descriptive statistics on both the original dataset and the anonymized. For example, compare the means and standard deviations and any other statistics that the end user may need. The aim is to have no statistically significant differences between the dataset before and after the anonymization.
4. **Relationships between variables** - From the report or information shared by the Data Provider, important relationships between variables may have been identified. In this case, it should be established that the relationship between these variables has been maintained.

When assessing utility, it is important to compare the anonymous version of the data against the clean version of the data. In some cases, modifications as part of the anonymization process may have been done before the use of `sdcMicro`. As such, the initial data coming into the `sdcObject` are not the dataset that should be used for comparison but the version before any manipulations were made.

4.2.6.1 Categorical variables

4.2.6.1.1 Basic descriptive statistics

```r
#Descriptive statistics of categorical variables in R
# Compare plots
plot(your_clean_data$your_variable)
mtext("Distribution before Anonymization", side = 3, line = 1, cex = 1.2)
plot(your_anonymous_data$your_variable)
mtext("Distribution after Anonymization", side = 3, line = 1, cex = 1.2)

# Compare table distribution
table_clean <- table(your_clean_data$your_variable_1, your_clean_data$your_variable_2,useN
round(prop.table(table_clean, 1), 2)
table_anon <- table(your_anonymous_data$your_variable_1,
               your_anonymous_data$your_variable_2,useNA= "always")
round(prop.table(table_anon, 1), 2)

# Calculate relationship between two categorical variables using chi-squared test
chisq.test(table(your_clean_data$your_variable), table(your_anonymous_data$your_variable))
prop.test(table(your_clean_data$your_variable), table(your_anonymous_data$your_variable))

#The null hypothesis is that there is no difference between the distribution of the variab
```

```
#In summary, the p-value should be greater than 0.10 or 0.05 to establish that the differe
```

4.2.6.2 Continuous variables

4.2.6.2.1 Basic descriptive statistics

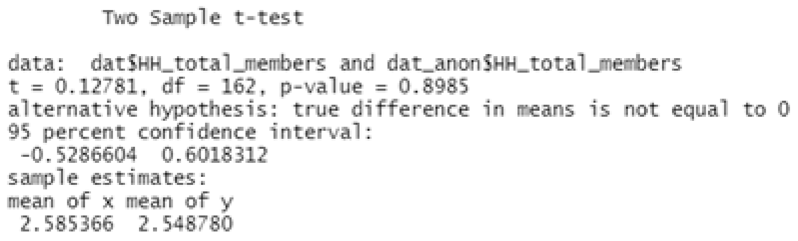
```
# Compare means, covariance and correlations
summary(your_clean_data$your_variable)
summary(your_anonymous_data$your_variable)
mean(your_clean_data$your_variable)
mean(your_anonymous_data$your_variable)
cov(your_clean_data$your_variable)
cor(your_anonymous_data$your_variable)
cov(your_clean_data$your_variable)
cor(your_anonymous_data$your_variable)

# Compare histograms
hist(your_clean_data$your_variable,
     main = "Histogram: Your variable name - original clean data")
hist(your_anonymous_data$your_variable,
     main = "Histogram: Your variable name - anonymous data")

# Compare density distribution
plot(density(your_clean_data$your_variable),
     xlim = c(x, x), ylim = c(x, x),
     main = "Density", xlab = "Your variable")
par (new = TRUE) # adds the second graph on top
plot(density(your_anonymous_data$your_variable),
     xlim = c(x, x), ylim = c(x, x),
     main = "Density", xlab = "Your variable name")
```

4.2.6.2.2 Difference of means test

A difference of means test is useful for comparing the significance of the difference of the mean of two continuous variables.

```
dif_means <- t.test(your_clean_data$your_variable, your_anonymous_data$your_variable, var.
dif_means #print the result

#If the p-value is above the threshold chosen for statistical significance (usually 0.10,
```

The following example compares the total number of household members in the clean version (`dat_clean`) of the dataset and anonymized version (`dat_anon`). The household size was topcoded as part of the anonymization process. The results of the t-test demonstrate that the difference in mean is not statistically significant. In summary, the p-value should be greater than 0.10 or 0.05 to establish that the difference between the clean and anonymous version of the variable is not statistically significant and the utility has been preserved.



```
        Two Sample t-test

data:  dat$HH_total_members and dat_anon$HH_total_members
t = 0.12781, df = 162, p-value = 0.8985
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5286604  0.6018312
sample estimates:
mean of x mean of y
 2.585366  2.548780
```

4.2.6.3 Variable relationships

4.2.6.3.1 Fitted models

Regression parameters of the same regression done on the clean and anonymous versions of the dataset can be done verify if the change was statistically significant. The choice of the regression formula depends on the type of dataset and the relationships between variables. For example, a regression could be used to explain earnings as a function of education and experience. If the new estimated coefficients fall within the original confidence interval, the data can be considered valid for this type of regression after anonymization.

```
#Define the variables to use in the regression
dependent_var <- "your_dependent_variable"
independent_var1 <- "your_independent_variable_1"
independent_var2 <- "your_independent_variable_2"
independent_var3 <- "your_independent_variable_3"

#Define the weight variable
weight_var <- "your_weight_variable"
```

```r
#Define the regression formula
independent_vars_formula <- paste(independent_var1, independent_var2,
                            independent_var3, sep = "+")
regression_formula <- paste(dependent_var, independent_vars_formula, sep = "~")
#Note: the formula will be:
#          your_dependent_variable ~ your_independent_variable_1 +
#          your_independent_variable_2 + your_independent_variable_3

#Calculate the regression of the original data
regression_clean_dataset <- lm(regression_formula,
                               your_clean_data,
                               na.action = na.exclude,
                               weights = your_clean_data[[weight_var]])
#Calculate the regression of the anonymous data
regression_anonymous_dataset <- lm(regression_formula,
                               your_anonymous_data,
                               na.action = na.exclude,
                               weights = your_anonymous_data[[weight_var]])

# Print the 95% CI for the clean dataset
confint(obj = regression_clean_dataset, level = 0.95)

# this will print something like this:
#                              2.5 %    97.5 %
# (Intercept)                 1.0951    1.1437
# your_independent_variable_1      -1.9614   -1.5778
# your_independent_variable_2       3.3923    4.2352
# your_independent_variable_3       2.5120    3.3995

# Print the coefficients for the anonymized dataset
regression_anonymous_dataset$coefficients

# this will print something like this:
#         (Intercept)         your_independent_variable_1
#              1.1074                       -1.7676
#     your_independent_variable_2    your_independent_variable_3
#                    3.6787                     2.8088

#As the coefficients fall within the confidence interval for all three variables (from the
```

4.2.6.4 Applying survey weights to analysis

In cases of disproportionately stratified datasets, weights must be applied to analysis that spans across strata. The **survey** package can be used to apply the weights and run simple statistics.

```
#Once the survey package is installed and loaded, the following can be used to create a ne

# Create survey
your_dataset_survey <- svydesign(ids=~0,
                                 weights=~survey_weight,strata=~your_strat,
                                 survey.lonely.psu="adjust",data=your_dataset)

#Now you can use your_dataset_survey to calculate the statistics needed. Note that it is m
```

4.2.7 Save anonymous version, codebook and anonymization script

Data should be saved at least as a csv for publication on the MDL. If the raw/clean version were provided as a **dta** (stata) or **sav** (spss) that preserves additional details such as variable and value labels it should also be saved in those formats.

```
#The following uses the haven package, so make sure it is installed and loaded before runn

# Export csv
write.csv(your_anonymous_data,
          "0_data/anonymous/your_anonymous_data.xlsx",
          sheetName="data",row.names=FALSE)

# Export xlsx
write_xlsx(your_anonymous_data, "0_data/anonymous/your_anonymous_data.xlsx")


# Export dta
haven::write_dta(your_anonymous_data, "0_data/anonymous/your_anonymous_data.dta")
```

For datasets with encoded values (responses), it can be very helpful for the Data Users to have a codebook that can be shared with the dataset. The example below demonstrates how to create a simple codebook. Note that this assumes that labels have already been created for the variables and values (see previous section).

Creating a codebook in R

```
# variable names and labels
your_dataset_var_codebook <- your_dataset %>%
  var_label() %>%
  enframe(name = "Code",
          value = "Label") %>%
  unnest(cols = Label)

#  value names and labels
your_dataset_val_codebook <- map(select_if(your_dataset, is.labelled), function(col) {
  distinct(tibble(values = as.character(col),
                  labels = as.character(to_factor(col))))}) %>%
  enframe(name = "variable") %>%
  unnest()

# export codebook to excel with a tab for variables and a tab for values
write_xlsx(list(variables=
                  your_dataset_var_codebook,
                values=your_dataset_val_codebook),
           "your_dataset_name_codebook.xlsx")
```

Finally, the anonymization script and workspace (`.R`) should be saved to be stored with the rest of the curation files on the GDS SharePoint space. These files should not be shared on RIDL. The anonymization script can be saved using the R studio interface. The workspace can be saved using the example code below

```
#save the workspace
save.image(file = "1_scripts/your_anonymization_workspace.RData")
```

5 Disclosure risk assessment

Once the microdata has been rendered anonymous or a conclusion has been made that it cannot be rendered anonymous without too much loss of utility, the Data Curator completes the disclosure risk assessment which includes information from the anonymization using SDC as well as complementary information on any other risks associated with disclosure, and writes the disclosure risk assessment (DRA) report. The Data Curator shares this report with the Data Provider, Data Controller and Data Protection Focal Point to inform the validation of the anonymization and authorization for the release of the anonymous version of the dataset on the MDL.

The DRA report must, at minimum, include the following information:

1. Summary of the DRA
2. List of UNHCR personnel, partners and/or other third parties performing each role (see Table 1) in the data curation process
3. Summary of the potential disclosure risk scenarios
4. Anonymization methods

a. Weights used for the anonymization
b. List of variables removed from the dataset
c. List of key variables used for the anonymization
d. Statistical disclosure control (SDC) methods performed and on which variables

5. Assessment of the risk of re-identification of data subjects in the microdata
6. Analysis of the utility of the anonymous microdata
7. Conclusions

a. Can individual data subjects be identified by any means reasonably likely, based on the data alone or in combination with other data? [Yes / No]
b. If necessary, are additional technical, organizational or legal measures or otherwise binding commitments in place to reduce the risk of disclosure? [Yes / No]
c. If the microdataset will be published on the MDL, specify the proposed Terms of Use.

8. Any other relevant comments or explanations.
9. Signatures, titles and dates (*Note: this is necessary for internal accountability for decision-making. As such, an email in lieu of signature is considered sufficient.*)

A R Markdown DRA report template was developed to facilitate the linkage between the anonymization and the report. This is done by combining text and code chunks[1]. It is based off the UNHCR R Markdown report template (`unhcrdown`), which was developed to facilitate an easy-to-read report for both technical and non-technical audience. Before getting started, both need to be installed if not already.

```
packages_needed = c("dplyr","foreign","ggplot2","haven","here",
                    "kableExtra","knitr", "labelled","lubridate",
                    "readr","readxl","rmarkdown","sampling",
                    "sdcMicro","stringr","survey","tidyr","tidyverse",
                    "VIM", "writexl")
                    repos = c(CRAN = 'https://cran.rstudio.com',
                              CRANextra = 'https://macos.rbind.io')
                    install.packages(packages_needed[!packages_needed %in% row.names(insta
                    update.packages(oldPkgs = packages_needed, ask = FALSE)
                    sapply(packages_needed, require, character.only=TRUE)

# UNHCR package for report template
remotes::install_github('vidonne/unhcrdown')
```

5.1 R Markdown text formatting

The following table provides some of the syntax used in R Markdown for formatting text in the DRA report. Note this is only a sample of the various types of formatting that can be done in R Markdown. The following cheat sheet is a useful tool to navigating all additional features:

Syntax	Description
# Header 1	Header 1
## Header 2	Header 2
### Header 3	Header 3

[1]For more information on R Markdown and code chunks: https://rmarkdown.rstudio.com/lesson-3.html and the full code chunk reference guide: https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf?_ga=2.24672569.2078611878.1633074546-1715259657.1633074546

Syntax	Description
### Header 4	Header 4
** bold ** and ___ bold ___	**bold** and **bold**
* italics * and _ italics _	*italics* and *italics*
‘code text‘	`code text`
* unordered list	· unordered list
+ sub-item	sub-item
1. ordered list	1. ordered list
2. ordered list	2. ordered list

5.2 Getting started

The template used below will call on the anonymization workspace and pull any data frames and other objects created. Ensure the workspace was saved prior to starting to write the report (see Prepare weights).

In cases of disproportionately stratified datasets, weights must be applied to analysis that spans across strata. The survey package can be used to apply the weights and run simple statistics.

```
# Create survey
your_dataset_survey <- svydesign(ids=~0, weights=~survey_weight,strata=~your_strata,survey
```

Now you can use your_dataset_survey to calculate the statistics needed. Note that it is more useful to compare proportions produced by the results as opposed to summaries because the summaries will be estimates for the entire population of interest in the sample frame not just within the sample

To start the report, open R studio and create a R Markdown file. Choose the HTML format and select the location to store the document.

Every R Markdown file starts with a YAML header, where the details of the document including title, date, author and template are specified. Details are always between - - -. The title and date will display on the first page of the report as well as in the header. It is defined from the beginning together with the report template.

Creating the title and defining the template in R Markdown

```
---
title: 'UNHCR Microdata Disclosure Risk Assessment Report'
author: 'First and Last Name - Position'
date: '`r Sys.Date()`'
output: unhcrdown::paged_simple
---
```

To be able to “call” on the anonymization script and pull any data frames and other objects created, the anonymization workspace needs to be loaded as well as the packages used. The following code is an example. Note that is it in a code chunk that will be hidden from the report.

About code chunks: Every code chunk should go between `{r}`. If the code output should be displayed but not the code itself, the option `{r, echo = FALSE}` is used. If neither the response nor the code should be displayed, the option `{r, include = FALSE}` should be used. More examples will be provided below.

Setting up the workspace in R Markdown

```
{r files, include=FALSE}
# Load the working space
load( "1_scripts/your_workspace_name.RData"))
```

5.3 The Report

5.3.1 Introduction

The introduction should include the name of the dataset and the link to the dataset on RIDL. This can be written directly in R Markdown using, i.e. does not need to be part of a code chunk. Consider the following example.

```
# Introduction

__Dataset:__ "Côte d'Ivoire: Mapping of persons at risk of statelessness - April 2019"

__RIDL link:__ <https://ridl.unhcr.org/dataset/civ-statelessness-april-2019/>
```

5.3.2 Summary

The objective of the summary is to provide the main conclusions of the DRA. It should be one sentence or a short paragraph that includes the following information:

- if microdata could be rendered anonymous while preserving their utility; and
- if it is recommended to publish on the MDL and under which Terms of Use.

This can be written directly in R Markdown as a text editor:

```
# Summary

__These data could be anonymized while preserving their utility, and it is recommended tha
```

5.3.3 List of roles

The list of roles should include the names of everyone involved in the process including the Personal Data Controller, Data Provider(s), Data Curator and Data Protection Focal Point. This can be written directly in R Markdown using a table format for readability. Consider the following example.

```
Data Curation Roles                     |
--------------------------------------- | -------------
Personal Data Controller                | Name (email)
Data Provider                           | Name (email)
Data Curator                            | Name (email)
```

5.3.4 Summary of disclosure risk scenarios

This section responds to question 1 in the DRA checklist: *What are the potential disclosure risk scenarios – e.g., realistic assumptions about who may be interested in the microdata and for what purpose – and available data and information (both internal and external) covering the same population group that could be linked to this personal microdata (e.g., through the mosaic effect)?*

It should be a short paragraph or two, and can be typed directly using R Markdown as a text editor. Consider the following example.

```
# Potential Disclosure Risk Scenarios

An intruder could be interested in identifying displaced individuals or households living

The likelihood of re-identification of data subjects based on this dataset alone is low du
a) the anonymized dataset only includes a sample of the original full dataset,
b) observable variables were checked for granularity and recoded where the level of detail
```

	N	n	weight
Camp 1	500	140	3.5714
Camp 2	230	50	4.6000
Camp 3	475	125	3.8000

```
c) the data was anonymized until 3-anonymity was reached.

The likelihood of re-identification of data subjects based on combining the anonymized ver
```

5.3.5 Anonymization methods

The anonymization methods section should include at minimum a description of the following:

- weights: an example of a section of weights could look like this:

```
## Weights
#Survey weights were calculated by dividing the total number of data subjects in the sampl

strata <- c("Camp 1", "Camp 2", "Camp 3")
N <- c(500, 230, 475)
n <- c(140, 50, 125)
w <- c(3.5714,4.6000, 3.8000)
weight_table <- data.frame(strata,N,n,w)

colnames(weight_table) <- c("", "N", "n","weight")
library(kableExtra)
```

```
Warning: package 'kableExtra' was built under R version 4.1.3
```

```
kbl(weight_table) %>%
  kable_classic()
```

- variables removed

If any variables were removed from the dataset that will be released on the MDL, they should be listed here. If they were removed using a list (see example under Remove variables that will not be published), then it can simply be called and displayed. Consider the following example.

```
## Variable removed
The following *direct identifiers* were removed from the dataset:

print(direct_identifiers)
```

- key variables

All key variables that were used as part of the SDC risk assessment should be listed here. If other variables were modified, but not used as part of the SDC risk assessment, they can be listed under a separate header (e.g. Variables modified). Consider the following example.

```
## Key variables
The following *key variables* were used for the SDC risk assessment:

  {r, echo=FALSE}
print(names(sdcObj@origData[,sdcObj@keyVars]))
```

- SDC methods performed

Some of this information can be written directly in R Markdown as a text editors; others can be pulled in from the anonymization workspace as code chunks. Consider the following examples broken down by section.

Part III

Publication of data

6 Validation and authorization

Before data can be published on the MDL, there are several steps that need to be taken, including, and most importantly ensuring that the Data Provider agrees with the curation, and validation and authorization is provided by the Personal Data Controller. The steps involved are outlined in section 5.6 of the Curation AI.

Quarterly release Microdata are published on the MDL on a quarterly basis to limit the burden on the DIMAs and Personal Data Controllers to respond to validation and authorization requests particularly if they may have many throughout the year. That said, if requested, microdata can be published outside of the scheduled release.

7 Publication on the MDL and other post-curation tasks

Once validation and authorization is received from the Personal Data Controller, the following tasks need to be done to publish the microdata on the MDL as well as update RIDL and other documentation. They are as follows:

7.1 Create MDL metadata

The metadata needs to be created for the MDL. See [Annex - Metadata guidance] for guidance on creating the metadata.

7.2 Request another Data Curator to review metadata

Before publication, the team of Data Curators share the task of reviewing datasets uploading to MDL from other Data Curators. This is assigned 1-2 weeks before the quarterly publication.

7.3 Publish data

Data are published on a quarterly basis with releases in January, April, July and October.

7.4 Update RIDL

Update RIDL with clean version of data, cleaning script and anonymous version of the data ensuring that each file is labelled correctly. See section Getting started for more details.

7.5 Update data registry

If the operation or region associated with the dataset has a registry, ensure to update this inventory with the relevant information related to the curation. In addition, update any global data inventories that need to be updated.

7.6 Update SharePoint

Save a copy of all curation files on the following SharePoint space GDS – Data Curation > Data repository > region folder > country folder.

A Annex: R Packages

The following is a list of R packages that can be useful to the work of a Data Curator, and a description of how they may be used. It's obviously non-exhaustive.

Package	Description and use
dplyr	Several tools to manipulate and clean data. Part of the larger tidyverse package.
foreign	Read and write data stored by some versions of 'Epi Info', 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', and for reading and writing some 'dBase' files.
ggplot2	Tools to create graphical representations of data. Part of the larger tidyverse package.
haven	Read and write data stored in formats from other statistical pacakages including SAS, SPSS and Stata.
here	Constructs paths to project files, avoiding issues with relative paths.
kableExtra	Generate simple tables
knitr	Generate reports in various formats. Part of the R markdown language
labelled	Manipulate variable and response labels.
lubridate	Part of the tidyverse package, useful for manipulating dates.
readr	Part of the larger tidyverse package, useful for writing R's own data format RDS.
readxl	Read data from .xls or .xlsx format
sampling	Draw samples from a data frame.
sdcMicro	Perform statistical disclosure control on a dataset.
stringr	Part of the larger tidyverse package.
survey	Perform various statistics and models on a survey dataset, applying survey weights where applicable.
tidyr	Part of the larger tidyverse package.
tidyverse	A set of packages used to manipulate, clean and visualize data including (but not limited to) dplyr, ggplot2, lubridate, readr, stringr and tidyr.
VIM	Tools to create graphical representations of data. Can be used to visualize missing and imputed data.
writexl	Export data frame to .xlsx format

B Annex - Metadata guidance

Metadata is designed to make data reusable in a responsible manner and to improve the functionality, search and interoperability of RIDL and the MDL. Metadata are also useful for preservation, institutional knowledge, discovery, understanding the data for analysis, understanding limitations for analysis, and ensuring data's appropriate use.

UNHCR has developed metadata templates based on the needs of RIDL and MDL users and on the most common types of operational data collected by UNHCR and its partners. This section provides general guidance on the metadata fields. Technical guidance on the use of the tools to upload metadata will be added soon as the tools are undergoing re-development.

Metadata standards

The MDL follows internationally accepted metadata standards used across UNHCR as well as operational partners including the World Bank and OCHA. This includes those developed as part of the Data Documentation Initiative (DDI). All metadata are in English.

B.1 Add/modify labels

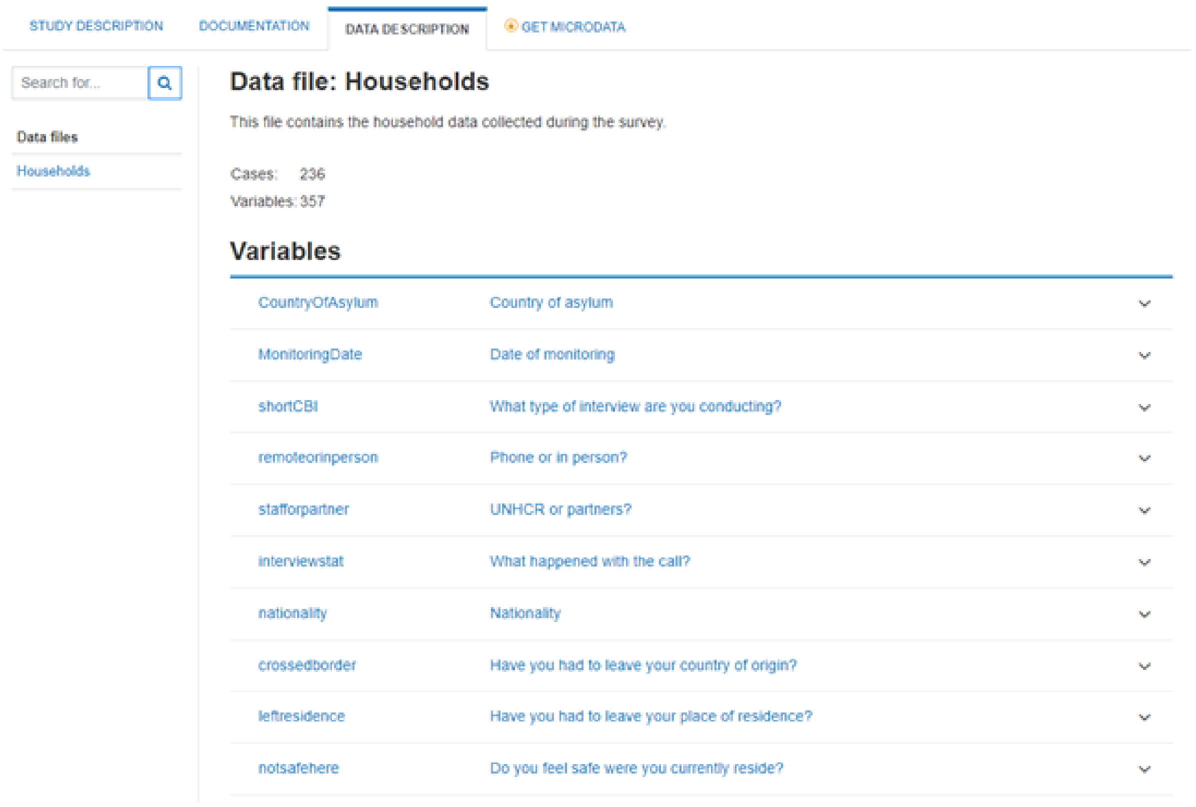
B.1.1 Variable labels

The MDL needs both the variable names and labels for the metadata on the MDL website. This facilitates the detailed search of the contents in each data file. See an example below:

A dataset may or may not have variable labels attached to each variable. Normally they are only attached for stata or spss files. Even if they did have labels, key variables often lose their labels after going through sdcMicro and/or some labels may need to be modified if the variables were modified (e.g. recoded). The following code can be used to add labels to a data frame in R from an Excel file with the list of variable names and variable labels.

Adding labels in R

Given the dataset `your_dataset` and an Excel file `data_labels.xlsx` with two columns that include variable names (`var_names`) and variable labels (`var_labels`), you can use the `labelled` package and the following code to add labels.



Data file: Households

This file contains the household data collected during the survey.

Cases: 236
Variables: 357

Variables

CountryOfAsylum	Country of asylum
MonitoringDate	Date of monitoring
shortCBI	What type of interview are you conducting?
remoteorinperson	Phone or in person?
stafforpartner	UNHCR or partners?
interviewstat	What happened with the call?
nationality	Nationality
crossedborder	Have you had to leave your country of origin?
leftresidence	Have you had to leave your place of residence?
notsafehere	Do you feel safe were you currently reside?

```
# Install package & add library
install.packages('labelled')
library(labelled)

# Import the mapping file. If you use a csv, you can use read.csv. If you use an excel fil
labels_map <- readxl::read_excel('2_documentation/data_labels.xlsx')

# Labeling function
label_from_mapping <- function(a_dataset, var_names, var_labels){

  for(i in 1:length(names(a_dataset))){
    a_name = names(a_dataset)[i]
    if(!a_name %in% var_names){
      print(paste('Label not found for var: ',a_name))
    }else{
      index = match(a_name, var_names)
      a_label = var_labels[index]
      #print(paste('setting to var: ', a_name, ' the label: ', a_label))
      var_label(a_dataset[a_name]) = a_label
    }
  }
  return(a_dataset)
}

# Apply labels
your_dataset <- label_from_mapping(your_dataset, labels_map$var_name, labels_map$var_label
#view the result
View(your_dataset)
```

B.1.2 Value labels

Value labels are descriptions of the values a variable can take. Value labels are particularly important for categorical variables whose value names are a shortened, abbreviated or an encoded version of the response option (e.g. 0 = no, 1 = yes). If the value names are shortened, abbreviated, or encoded AND documentation in a convenient format is available (i.e in a codebook or the questionnaire), then the value names can be shared in their shortened, abbreviated or encoded format. If documentation is not available in a convenient format or cannot be shared, either:

- the value names should be converted to their labels (see example below); or

- a codebook should be created (see example under section Save anonymous version, codebook and anonymization script).

Replacing value names in R

```
# Replace codes with labels
your_dataset$your_variable[your_dataset$your_variable =='1'] <- 'Yes'
your_dataset$your_variable[your_dataset$your_variable =='0'] <- 'No'

# or alternatively
your_dataset %>%
     mutate(your_variable = recode(your_variable, '1' = 'Yes', '0' = 'No'))

#The following is a more efficient way to replace the codes across multiple variables.
# Replace codes with labels for multiple variables
map_values <- function(yesno_var){
     yesno_var = case_when(
     yesno_var == 1 ~ 'yes',
     yesno_var == 0 ~ 'no'
     )
     return(yesno_var)
}

your_dataset <- mutate(your_dataset, across( c(your_var1, your_var2, your_var3), map_value
```

B.2 Metadata fields

Metadata created under the DDI standard has 4 main sections:

- **Dataset (study) level**: General information on the dataset as a whole
- **Data file (resource) level**: Information on the data file itself, including its creation date and version number
- **Variable Level**: Information on the variables in a data file include the variable names, labels and response rate
- **Other files (resource) level**: General information on other resources such as questionnaires, reports, summary analyses, etc.

The following tables provide an overview of the mandatory metadata fields in RIDL and the MDL, and instructions for their use. Some fields are not mandatory in the tools (denoted with an *) however they are still included in the table because they are necessary for the curation process.

Note: While it is not necessary to complete all the fields, the metadata should include all the required fields to facilitate responsible use of the data. When populating the metadata, assume that any users may not have access to the Data Provider to seek more information than is what is available in the metadata.

B.2.1 Mandatory fields at the dataset level

Field RIDL	Field MDL	Description & usage
Title*	Title*	The dataset study title should follow, when possible, the naming convention used by UNHCR's Operational Data Portal which is generally *Country: Subject - Mmm YYYY. Example: Zimbabwe: Socio-economic assessment in Camp 1 — 2017.*

Field RIDL	Field MDL	Description & usage
not included	Study type*	Controlled field for broad category defining the dataset. List of possible responses includes: 1-2-3 Survey, phase 1 [hh/123-1] Administrative Records, Health (ad/hea] Administrative Records, Education (ad/edu] Administrative Records, Other (ad/oth] Agricultural Census [ag/census] Agricultural Survey [ag/oth] Child Labor Survey [hh/cls] Core Welfare Indicators Questionnaire [hh/cwiq] Demographic and Health Survey [hh/dhs] Enterprise Survey [en/oth] Enterprise Census [en/census] Income/Expenditure/Household Survey [hh/ies] Informal Sector Survey [hh/iss] Integrated Survey (non-LSMS) [hh/is] Labor Force Survey [hh/lfs] Living Standards Measurement Study [hh/lsms] Other Household Health Survey [hh/hea] Other Household Survey [hh/oth] Price Survey [hh/prc] Priority Survey (hh/ps] Population and Housing Census [hh/popcen] Sample Frame, Households [sf/hh] Sample Frame, Enterprises [sf/en] Service Provision Assessments [hh/spa] Socio-Economic/Monitoring Survey [hh/sems] Statistical Info. & Monitoring Prog. [hh/simpoc] World Fertility Survey [hh/wfs] World Health Survey [hh/whs]
not included	ID Number	Unique study identifier. Should follow the same structures as outlined in section 2.1.2 Dataset identifier.
URL	*not included*	The URL link to the data on RIDL. This is automatically filled in RIDL, however, should be modified so that it is the same as the dataset identifier (see section 2.1.2 Dataset identifier). For example: https://ridl.unhcr.org/dataset/UNHCR_ZWE_2017_SENS_CAMP1

Field RIDL	Field MDL	Description & usage
available at resource level metadata	Version Description	Should describe the version of the data in the study: v01: Raw data v1.1: Clean data v2.1: Anonymized data
Description	Abstract*	Summary of the dataset including the purpose and objectives of the data collection exercise including any relevant background information, what was collected, from whom was it collected, when was it collected, how was it collected and by whom was it collected as well as any other information pertinent to be known by other Data Users. Note that the subject of the abstract should be the data (not the report or any other product that was developed using the data). Below is an example of good abstract: “Against the recent COVID-19 pandemic and its secondary socio-economic impact, UNHCR in partnership with WFP undertook a Joint Needs Assessment (JNA) in Mantapala settlement to identify livelihood challenges and opportunities and develop socio-economic profiles of the most vulnerable. The JNA was to inform programmatic decisions and suggest the most appropriate and feasible targeting approach for future interventions by UNHCR and WFP in the settlement. An extensive literature review and technical discussions took place to identify the knowledge gap during the assessment design phase. At the time of the assessment, 4,076 households were living in the settlement. Primary data collection took place between 19 and 28 September 2020 by UNHCR and WFP field teams. This included five focus group discussions, two key informant interviews and 1,128 household interviews. The household interviews used a structured questionnaire. Simple random sampling was used, and findings are statistically representative at the settlement level This dataset is the anonymous version of the household interview data collected through the structured questionnaire. It was processed jointly by UNHCR and WFP

Field RIDL	Field MDL	Description & usage
Operational purpose of data	Kind of Data	In the MDL, this field is a broad classification of the data from the following controlled vocabulary: *Sample survey data* *Census/enumeration data* *Administrative records data* *Aggregate data* *Clinical data* *Even/trasaction data* *Observation data/ratings* *Process-produced data* *Time-budget diaries* In RIDL, the field is called 'Operational purpose of data' and allows multiple selection from the following list: *Participatory assessments* *Baseline Household Survey* *Rapid Needs Assessment* *Protection monitoring* *Programme Monitoring* *Population data* *Cartography, infrastructure and GIS* *Results monitoring*
not included	Description of scope	Description of the subject matter covered in the dataset. Note: this is not geographic scope but thematic.
	Country	Name of country(ies) covered by the dataset, as well as its ISO-3 Code. See guidance on country names in the UNHCR Style Companion, March 2019.
Geographies	Geographic Coverage	Describes the geographic units this data has been collected in. In the case of RIDL, the most detailed geographical unit needs to be selected.
-	Universe	Population of interest for the whole study
Data Container	-	Data container on RIDL.
External Access Level	-	Only relevant for microdata to be published on the MDL. Single select on access level on MDL.

Field RIDL	Field MDL	Description & usage
-	Primary investigator	Name of the institution/organization(s) implementing the study. This may be different from the data collector if the institution/organization(s) recruited a third party to collect the data.
Data Collector	Data collector(s)	Name of the institution/organization(s) that collected the data for the study.
Topic classification	Topics	Topic(s) covered by the study. This is a multiple select, with predefined options available in RIDL.
Unit of measurement	Units of analysis*	Unit of observation / subject of the data. For example, if the data subjects were households, the unit of observation is the household, if it was individuals, individuals, etc.
Sampling Procedure*	Sampling procedure*	Multiple select on sampling method if relevant. On MDL, open text field.
Data collection-technique	Data collection mode	Single select on method used to collect the data.
Archived	-	Yes/No if data collection and manipulation is complete and data are to be archived. This should remain 'yes' if data collection/manipulation is still active.

Field RIDL	Field MDL	Description & usage
Admin notes – Sampling procedure*	-	If the data are a sample, this section must include information on the sample frame (including the size of the sample frame), sample size and further details on the sampling methodology.
Admin notes – Access authority*	Access authority	The name and organization of the data controller(s).
		For RIDL, this should be the name of the UNHCR staff member that is considered the data controller. If the data is co-controlled by another organization, this should also include the name of other organizations that have control over the data. For MDL, this should be UNHCR with the contact information as microdata@unhcr.org.
-	Citation Requirement	The citation requirements for MDL Data Users in the following format: Data Controller (YYYY). Study title, YYYY. Accessed from https://microdata.unhcr.org.
		Example: UNHCR (2021). Burkina Faso: Standardized Expanded Nutrition Survey microdata, 2014. Accessed from: https://microdata.unhcr.org. For externally harvested datasets, the citation should remain as provided if there is one or follow the above format if there is not one available in the provided metadata.

B.2.2 Mandatory fields at the resource level

Field RIDL	Field MDL	Description & usage
Name*	Name*	Brief name of the file. This should include if the data are raw, clean or anonymous. Use the following naming conventions to distinguish between the three: data (raw v0.x) data (clean v1.x) data (anonymous v2.x)
File type		Type of file such as .xlsx, .csv, .dta, etc. This is automatically populated based on the file uploaded.
Internal Access Level		Single select on who within UNHCR should have access to the data. Either internally visible (anyone in UNHCR can have access) or private (only certain individuals should have access).
Data collection first date	Data collection dates – Start*	First date of data collection
Data collection last date	Data collection dates – End*	Last date of data collection
Version	Version	Version of the file
File Process Status		The process status of the file: Raw, Cleaned or Cleaned & Anonymized.
Identifiability		The level to which data subjects are identifiable in the data. Select between Personally identifiable (i.e. direct identifiers included), Anonymized 1st level (direct identifiers removed), Anonymized 2nd level (anonymized using UNHCR standards and for release under Licensed Use), Anonymized 3rd level (anonymized using UNHCR standards and for release under Public Use).

C Annex: Key and sensitive variables

The following table is a non-exhaustive list of some of the common key and sensitive variables found in operational microdata collected by UNHCR and partners, and some example actions that could be performed as needed.

Please note that the following variables may be considered key variables in one dataset and not another depending on the context, and the actions may or may not be relevant depending on the dataset and the context.

Variable	Example actions
Unique ID	Replace with pseudonym (see section Unique ID)
Sample weight	In the case of stratified sampling, the weight will have
	If local suppressions are done on the strata variable, t
Household size	Convert to factor and treated as a categorical key vari
	Top code larger households, e.g.: 8+
	Recode into groups useful to analysis, e.g.: 1, 2, 3-4, 5
	In the case of a hierarchical dataset, household size m
Breakdown of household members by age, sex, etc.	Only preserve level of detail useful to analysis.
	For household-level files, replace with ratio, e.g. propo
Location variables	Administrative divisions – In cases where some divisio
	Populated place – In cases where some places have fev
Marital status	Recode to 'married', 'widower', 'separated/divorced', a
Age	Convert to factor and treated as a categorical key vari
	Recode into groups useful to analysis, e.g., <18, 18-59
	In the case of SENS surveys children's module, age in
Gender	In most cases this variable contains only two categorie
Country of origin, country of birth, nationality, etc.	In cases where some categories have few observations,
Variables related to sexual and gender-based violence.	This should be discussed with the Data Provider.
Language	In cases where some categories have few observations,
Income	Round to avoid exact matching.
	Top code outliers.
	Recode into groups useful for analysis.
Open text variables	See section Free text variables.