DDFcsv format

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

DDF is a data model used by Gapminder Foundation to structure their data in the open numbers initiative. The data model can be represented in many ways. This document will explain the DDFcsv format, a collection of csv files and one json file.
This document assumes you have read the introduction to the DDF model. Thus, it will only explain how to represent DDF in csv, not what DDF itself is.

File path

What datapoints, entities, concepts and metadata to find in what files is defined in the schema and resources sections of a datapackage.json file. This is the entry point for any machine that wants to explore the data set. This file will always be named: `datapackage.json` and should be placed in the root folder of the dataset.

The file names in this document are merely guidelines. However, we strongly advise you to follow our guidelines. This ensures a consistent, systematic naming of files in every DDF dataset, making them easy to understand and explore by anyone new to your dataset. It also allows you to automatically create a datapackage.json file with one of our tools.
By not enforcing file names we give you freedom to choose custom filenames in situations we did not foresee. Please do not abuse this freedom.

File structure

The file structure described in this document is mandatory to follow.

1 General rules

1.1 File path

All DDFcsv files containing data start with

```
ddf--<collection>
```

Where

`<collection>` is the data collection this file contains. This can be one and only one of: `datapoints, entities, concepts` or `metadata`.

DDFcsv files MUST be placed in the root directory or in a subdirectory of the root of the dataset. However, they MUST NOT be placed in the `/lang/` directory. This directory is reserved for translations.

1.2 File structure

Files are in csv format (wiki) with a mandatory header row on the first line. The encoding MUST be UTF-8. The UTF-8 encoding SHOULD NOT use a BOM. Per convention you MAY use linux-style line endings (LF) instead of the windows-style (CRLF) line endings as defined in the csv standard.
Missing or absent data MUST be represented by an empty field.

All data in DDF is represented by key-value pairs. One file can only contain data with the same key. Different collections never have the same key.

- Different collections can never be in the same file.
  Example: One file can never contain both datapoints and entities.

2 DataPoints

A DataPoint is a key-value pair which stores how indicators change across dimensions. For example, how population changes across years and countries.

2.1 File name

DataPoints in DDFcsv are stored in files named with the following pattern:

```
ddf--datapoints--<indicators>--by--<dimensions>.csv
```

Where

`<indicators>` is a list of indicators in this file, separated by `--`
`<dimensions>` is a list of dimensions in this file, separated by `--`

Each dimension in the list of dimensions has the following pattern:

```
<dimension>[-<values>]
```

Where

`<dimension>` is the concept for this dimension
`<values>` is an exhaustive list of values of this dimension found in this file, separated by `-`

For example

```
ddf--datapoints--population--by--geo--year.csv
ddf--datapoints--gdp--gdp_per_cap--by--geo--year--gender.csv
ddf--datapoints--population--by--geo-usa-swe--year.csv
ddf--datapoints--population--by--geo--year-2000.csv
ddf--datapoints--population--by--geo-ukr--year-2001.csv
```

2.2 File structure

DataPoints are key-value pairs with one or more dimensions (key) and one indicator (value).

The general rule of only the same key in one file also applies to DataPoints. One file can only contain DataPoints with the same key, i.e. the same set of dimensions. There can be multiple indicators for those dimensions in one file because they're not part of the key.

For example, the below table contains

1. DataPoints with dimensions: country and year, indicator: government_type
2. DataPoints with dimensions: country and year, indicator: population

country	year	government_type	population
Germany	1940	dictatorship	3954858
Germany	2015	parliamentary democracy	50495854
Sweden	2015	parliamentary democracy	9504847

ddf--datapoints--government_type--population--by--country--year.csv

Remember that column headers are concept identifiers and that thus they can only contain lowercase alphanumeric characters and underscores.

2.2.1 Splitting files by indicator column

Splitting the above example in two files for each indicator is fine too.

country	year	government_type
Germany	1940	dictatorship
Germany	2015	parliamentary democracy
Sweden	2015	parliamentary democracy

ddf--datapoints--government_type--by--country--year.csv

country	year	population
Germany	1940	3954858
Germany	2015	50495854
Sweden	2015	9504847

ddf--datapoints--population--by--country--year.csv

Splitting the indicators over two separate files has pros and cons:

+ The indicators are separately readable. A machine does not have to load data for both indicators if it's only interested in one.
- There are more files and the total file size is bigger because the keys have to be repeated for each indicator.

2.2.2 Splitting files by dimension value

Another way to split files datapoint files is by the value of the dimensions. For example, from the following table, we could make two files. One containing data for Germany and one for Sweden. Or one for 1940 and one for 2015. Watch how the filenames contain information about what value to expect for a dimension in the file.

country	year	gender	population
germany	1940	male	3954858
germany	2015	female	50495854
sweden	2015	male	9504847

ddf--datapoints--population--by--country--year--gender.csv

By country: One file for germany, one for sweden:

country	year	gender	population
germany	1940	male	3954858
germany	2015	female	50495854

ddf--datapoints--population--by--country-germany--year--gender.csv

country	year	gender	population
sweden	2015	male	9504847

ddf--datapoints--population--by--country-sweden--year--gender.csv

By year: one file for 1940, one for 2015:

country	year	gender	population
germany	2015	female	50495854
sweden	2015	male	9504847

ddf--datapoints--population--by--country--year-2015--gender.csv

country	year	gender	population
germany	1940	male	3954858

ddf--datapoints--population--by--country--year-1940--gender.csv

The filenames follow the naming convention as defined above. Behind each dimension is an exhaustive list of values of that dimension in the file, separated by a single hyphen (-).

The following example shows a more complex case where both country and age values are in the filename. For country, there are two values in the file, so both values are mentioned in the filename.

country	year	age	population
germany	2015	5	50495854
sweden	2015	5	9504847
sweden	2014	5	485858
sweden	2013	5	857474

ddf--datapoints--population--by--geo-sweden-germany--year--age-5.csv

Why?

First of all, it is a way to split large files into smaller files in a structured way.
Second, following this convention allows computers reading the DDFcsv dataset to be more selective in what file contents they have to read to gather certain data.
For example, a population pyramid chart shows the population for only Germany. The dataset contains data for all countries. By splitting files per country and following the naming convention, a computer only has to open the data file containing Germany's population data, instead of reading irrelevant population data for all other countries too.

3. Entities

An entity is a key-value pair which stores properties of entities.
Each entity belongs to one entity domain and can belong to one or more entity sets.

3.1 File name

Entities in DDFcsv are stored in files with the following name:

```
ddf--entities--<entity_domain>[--<entity_set>]*.csv
```

Where

`<entity_domain>` is the entity domain the entities in the file belong to

`[--<entity_set>]*` is included if the file contains only entities belonging to a certain entity set. When one file enumerates multiple entity sets, this part of the filename is repeated for every entity set.
`<entity_set>` is the entity set all entities in the file belong to

For example

```
ddf--entities--geo--country.csv
ddf--entities--geo--country--un_state.csv
ddf--entities--geo.csv
ddf--entities--gender.csv
```

3.2 File format

An entity and one of its properties form a key-value pair. Each entity belongs to one entity domain.
All entities in one entity domain can be enumerated in one file, with their properties. For example the entity domain `geo`, containing all geographic locations, enumerated in `ddf--entities--geo.csv`:

geo	name	is--country	is--city	currency	country
hkg	Hong Kong	1	1	Hong Kong Dollar	hkg
swe	Sweden	1	0	Swedish Krona	
sto	Stockholm	0	1		swe
...	...	...	...	...	...

Once more a reminder that the entity identifiers (here under the `geo` column) can only contain lowercase alphanumeric characters and underscores.

3.2.1 Entity sets

An entity set is a set of entities within one entity domain. The entities in an entity set can be enumerated in one file. The entity-identifier in this file can either be the entity set identifier or the entity domain identifier.
For example, the entity sets `country` and `city` are part of the entity domain `geo`. The city entities can be enumerated in a file `ddf--entities--geo--city.csv`. Within this file, the concept (column header) for the entity-identifier can be either `geo` or `city`, since the entities in the file belong to both. Thus the following two tables are both valid contents of the `ddf--entities--geo--city.csv` file, reflecting exactly the same data.

geo	name	is–city	country
hkg	Hong Kong	1	hkg
got	Gothenburg	1	swe
sto	Stockholm	1	swe
...	...	...	...

ddf--entities--geo--city.csv

city	name	is–city	country
hkg	Hong Kong	1	hkg
got	Gothenburg	1	swe
sto	Stockholm	1	swe
...	...	...	...

ddf--entities--geo--city.csv

3.2.1.1 Multiple definitions of the same entity

One entity has to belong to one entity domain and can belong to one or more entity sets. Entity domains and entity sets can have their members enumerated in separate files. Thus, it is possible to define the same entity in multiple files. An entity can be defined in one of the following:

- A file enumerating the whole entity domain
- A file enumerating one of the entity sets it belongs to.

An entity needs to be defined at least once, in any of the above files. In other words, you can either define it in an entity domain file, an entity set file or both.
In the example below, Hong Kong and Gothenburg are defined in both the enumeration of entity domain `geo` and the enumeration of the entity set `city`.

geo	name
hkg	Hong Kong
got	Gothenburg
swe	Sweden
...	...

ddf--entities--geo.csv

city	is--city	country
hkg	true	hkg
got	true	swe
...	...	...

ddf--entities--geo--city.csv

3.2.1.2 Multiple entity sets

An entity can belong to multiple entity sets. Thus, one entity can be defined in each entity set file it belongs to. For example, the geographic place Hong Kong is in the entity domain `geo`, entity set `country` and entity set `city`. It can be defined in multiple files: a file enumerating all geo's, a file enumerating countries and a file enumerating cities.

When defining the same entity multiple times:

- The files do not have to use the same concept identifier.
  Below you can see the same entity `hkg` be defined with three different concept identifiers: geo, city and country. This is possible because it belongs to all three simultaneously.
- The same entity property can be defined in multiple files. In this case, it must be the same in every file.
  Below, the entity property `name` of `hkg` is *Hong Kong* in every file. If one or more files have a different value for `name`, the data set is inconsistent and thus invalid.

geo	name	latitude	longitude
hkg	Hong Kong	22.2783	114.1747

ddf--entities--geo.csv

city	name	is--city	country
hkg	Hong Kong	true	hkg

ddf--entities--geo--city.csv

country	name	is--country	currency
hkg	Hong Kong	true	Hong Kong Dollar

ddf--entities--geo--country.csv

However, if a data set contains entity sets, they do not have to be in separate files. A data set with an entity domain `geo` and entity sets `country`, `city` and `region`, can:

- have all geo-entities in all sets enumerated in one file: `ddf--entities--geo.csv`
- have one or more entity sets enumerated in separate files

3.2.1.3 Combining domain and sets in one file

It is possible to enumerate all entities in the domain, including their set membership in one file. Below, the entity sets country and city of the domain geo are defined in one file. The entity property `currency` for countries is left empty for Stockholm, which is not a country. The concept-identifier must be `geo`. It cannot be `city` or `country` since not all entities in the file belong to that set. It's also not allowed to have a `city` and a `country` column next to each other.

geo	**name**	**is--country**	**is--city**	**currency**
hkg	Hong Kong	true	true	Hong Kong Dollar
swe	Sweden	true	false	Swedish Krona
sto	Stockholm	false	true	

ddf--entities--geo.csv

3.2.1.4 Combining only certain sets in one file

Two or more entity-sets can share one file. The other entity sets then either have their own files in the same domain or are combined in the entity domain file. Thus, an entity domain file does not necessarily contain all entities in that domain. Some entities can be solely defined in entity set files while not being defined in the entity domain file.
Below, the entity sets country and un_state (United Nations state) have a shared file. Like before, in a shared file the concept-identifier must be the entity domain (`geo`).
The entity set `region` can either have its own file or be part of the entity domain file. In the latter case the key MUST be `geo`, even if it only contains `region` entities.

geo	**name**	**is--country**	**is--un_state**	**currency**	**region**
hkg	Hong Kong	true	false	Hong Kong Dollar	asia
swe	Sweden	true	true	Swedish Krona	europe
asm	American Samoa	true	false		americas

ddf--entities--geo--country--un_state.csv

geo	**name**	**is--region**
asia	Asia	true
africa	Africa	true
europe	Europe	true

ddf--entities--geo--region.csv or *ddf--entities--geo.csv*

4. Concepts

Concepts are the table headers anywhere in a DDFcsv dataset. In one dataset, if two DDFcsv files contain the same table header they are the same concept. Concept properties are concepts which give information about concepts.

The concepts and their properties in a DDFcsv dataset are enumerated in one or more files which have `concept` as their single value key. Typically, one file, called `ddf--concepts.csv` is used.

An example is shown below:

key: concept	*values: concept properties*				
concept	**concept_type**	**domain**	**drill_up**	**unit**	**link**
geo	entity_domain				http://...
region	entity_set	geo			http://...
country	entity_set	geo	region		http://...
city	entity_set	geo	country		http://...
unit	entity_domain				http://...
link	string				
name	string				http://...
government_type	string				http://...
population	measure			people	http://...
area	measure			square_meter	http://...
income_bracket	entity_domain				http://...
latitude	measure			degrees	http://...

5. Metadata

Metadata is data about other data in the dataset. DDFcsv supports a format which enables defining metadata for subsets of the dataset at once. This means you only have to define a certain comment, source or other piece of metadata once or twice instead of copy-pasting it for every relevant record of data.

The relevant subset of the dataset is indicated by a key-value pair and an optional filter. We will first look at metadata without a filter.

All data in DDF is defined by a key and a value. This is the case for datapoints, entities and concepts and even metadata. In the table below we can find some yearly population and income data of countries. The table contains two key-value pairs: `<country,year>-population` and `<country,year>-income`. Country and year form the key for both, population and income are the values.

key		*value*	*value*
country	**year**	**population**	**income**
sweden	1940	9495847	484578
sweden	1950	9948857	387585
sweden	1970	9483538	745847
russia	1960	49548499	485857

For the country entities below there are 3 key-value pairs: `<country>-full_name`, `<country>-latitude` and `<country>-longitude`.

key	*value*	*value*	*value*
country	**full_name**	**latitude**	**longitude**
sweden	Kingdom of Sweden	59.3500	18.0667
norway	Kingdom of Norway	61.0000	8.0000
russia	Russian Federation	60.0000	90.0000

When defining metadata, we will reference the key-value pair the metadata applies to. Below we define the source of each key-value pair above in DDFcsv:

key		*value*
key	**value**	**source**
[“country”, “year”]	population	United Nations Population Statistics
[“country”, “year”]	income	World Bank
[“country”]	full_name	CIA Factbook
[“country”]	[“latitude”, “longitude”]	Socrata Open Data

The key is expressed in one column containing either a string or an array of strings. The value is also expressed as a string or multiple values can be referenced at once using an array of strings. An array for the *key* defines a multi-dimensional key while an array for the *value* defines multiple key-value pairs.
For example: The first row references one key-value pair with a multidimensional key consisting of country and year. The last row references two key-value pairs: country-latitude and country-longitude.

Each column in the metadata, except for key and value is a concept. This means from the metadata table above, the concept `source` needs to be added to the concepts of the dataset. In this case source is a string concept. It can just as well be of any other concept type. For example, sources could be entities with properties like full names, url’s, trustworthiness ratings et cetera.

Metadata key-value pairs are **not unique.**

5.1 Filters for more granular metadata

With the above notation we can define metadata on a key-value pair level. However, sometimes that is not detailed enough. It is possible the population of the country Sweden comes from a different source than that of the country Ukraine. Or that the data from 1800-1950 has a different accuracy than 1950-2015 data.

Filters allow you to reference an even smaller subset of the data for defining metadata, up to the level of one record. You can apply a filter by creating a filter column in the metadata file and adding filters for the relevant rows. The filter column accepts filters in JSON format as described in the DDFQL where documentation.

key	*value*	*value*

key	value	filter	source	accuracy
[“country”, “year”]	population	{ “country”: { “$neq”: “sweden” } }	United Nations Population Stats	

[“country”, “year”]	population	{
		“year”: { “$gte”: 1950 },
		“country”: “sweden”
		}
			Swedish Statistics Bureau	0.95

[“country”, “year”]	population	{
		“year”: { “$gte”: 1950 },
		“country”: { “$neq”: “sweden” }
		}
			FOO	0.90

[“country”, “year”]	population	{ “year”:
		{
		“$gte”: 1800,
		“$lt”: 1950
		}
		}
				0.80

[“country”,”year”]	population	{ “year”:
		{
		“$gte”: 1800,
		“$lt”: 1900
		}
		}
				0.75

[“country”]	full_name	{ “country.latitude”: { “$gt”: 0 } }	CIA Factbook	
[“country”]	[“latitude”, “longitude”]		Socrata Open Data	

```
SELECT
        KEY key, value, filter
        VALUE source
FROM
        metadata
WHERE
        key: { $all: [“geo”, “year”] }
```

```
NO NORMALIZATION
SELECT
        KEY geo, year
        VALUE population, population|source
FROM
        datapoints
WHERE
        geo = “ukraine”
        year > 1900
```

countr	Year	population	source	accuracy
ukraine	1900	438484	UN	
ukraine	2000	49585	UN	0.9

6. DDFcsv Datapackage

Each DDFcsv dataset MUST contain a file named `datapackage.json` which MUST be placed in the root folder of the dataset. It MUST follow the DDFcsv datapackage specification. The `resources` and `ddfSchema` sections of this file are the entry point for any machine that wants to explore the data set.

7. Concept types notation

- Values of boolean concepts types are limited to the following strings: `true`, `TRUE`, `false` or `FALSE`.
- Values of measure concept types are written as strings in decimal notation, with a `.` as the decimal separator and no thousand separators. Scientific notation is not allowed.
- All values of all other concept types are represented by strings

8. Language & Translations

The strings in a DDFcsv dataset SHOULD be written in one language. This language SHOULD be specified in the accompanying `datapackage.json` file and applies to all `string` concepts in the dataset.
A dataset MAY contain translations to additional languages. These translations of strings MUST be stored in one file per language, per original file. An example of a dataset containing translation files can be found here.

8.1 File path

Translations are stored in one folder per language, using the following naming convention

```
/lang/<language_tag>/<original_file_path>
```

Where

- `<language_tag>` MUST be a language identifier as per the Unicode Technical Report #35. This definition is very close to the RFC standard (BCP 47) "Tags for the Identification of Languages". The unicode standard has some additional compatibility built in. Transforming a Unicode language identifier to a BCP47 language tag is simple. This document contains a bit more information about the standards. More friendly overviews like this one will often be accurate enough to help choosing the right language tag.
- `<original_file_path>` MUST be the file path of the file the translations are for, as also defined in `datapackage.json`.

Thus, the file and directory structure in the language folder will mirror the dataset root folder for all files that have translations.

For example

```
/lang/en-US/ddf--datapoints--gov_type--by--geo--year.csv
/lang/nl-nl/ddf--entities--geo--country.csv
/lang/se-SV/ddf--entities-geo--country.csv
/lang/ru/entities/ddf--entities--geo--country.csv
/lang/en/geography_data/ddf--datapoints--name--by--geo--year.csv
```

8.2 File format

A translation file MUST contain all key-concepts and SHOULD contain one or more value-concepts from the original file. The value-concepts in the translation file MUST be string concepts and MUST be present in the original file.

ddf--datapoints--government_type--name--population--by--country--year.csv

country	**year**	**government_type**	**name**	**population**
ger	1940	dictatorship	German Empire	8584
ger	2015	parliamentary democracy	Federal Republic of Germany	9293
swe	2015	parliamentary democracy	Kingdom of Sweden	2874
ned	2015	parliamentary democracy	Kingdom of the Netherlands	3374

/lang/de/ddf--datapoints--government_type--name--population--by--country--year.csv

country	**year**	**government_type**	**name**
ger	1940	Diktatur	Deutsches Reich
ger	2015	parlamentarische Demokratie	Bundesrepublik Deutschland
swe	2015	parlamentarische Demokratie	Königreich Schweden
ned	2015	parlamentarische Demokratie	Königreich der Niederlande

Missing translations

Translation of any single string is OPTIONAL. When a translation of a string is missing, the string in the original language MUST be used. There are four ways in which a translation can be missing, the last three of which are shown in the example below.

1. The translation file for a data file is not present.
2. The string concept is not present in the translation file
   See `government_type` in the example
3. The row is not present in the translation file
   See `country:ned, year:2015` in the example

4. The value is empty in the translation file
   See `name, country:ger, year:2015` in the example

ddf--datapoints--government_type--name--population--by--country--year.csv

country	year	government_type	name	population
ger	1940	dictatorship	German Empire	8584
ger	2015	parliamentary democracy	Federal Republic of Germany	9293
swe	2015	parliamentary democracy	Kingdom of Sweden	2874
ned	2015	parliamentary democracy	Kingdom of the Netherlands	3374

/lang/de/ddf--datapoints--government_type--population--name--by--country--year.csv

country	year	name
ger	1940	Deutsches Reich
ger	2015	
swe	2015	Königreich Schweden

Reasons for choosing this file format

We have chosen this translation format for a number of reasons, these are three important ones:

1. A per file format ensures that a DDFcsv reader only reads the necessary translations for the data it's using. This is opposed to reading a large translation file with all translations to translate only a small subset of the dataset.
2. Including the keys for each string ensures correct translation of all strings. A simple dictionary would not suffice because some strings which are equal in one language are different in others. For example the short name of Rome, Italy and Rome, Georgia, US, are the same in English (Rome and Rome), but different in Italian (Roma and Rome).
3. Having one dataset language and separate translation files ensures that simple applications, which do not support translation functionality, can read the dataset as is. Therefore, it is not possible to replace strings with unique identifiers, even though this could reduce redundancy.

9. Redundant data

Redundant data is data which occurs multiple times in one dataset. Another term for this is duplicate data. This means the same values for the key and value in a key-value pair. For example, in the below tables, the datapoint for the population of Sweden in 2015 is redundant. It exists in two tables. Important is the assumption that `country` is an entity set in the domain `geo` so that `swe` in `geo` and `swe` in `country` represent the same entity. None of the other data is redundant.

ddf--datapoints–population--by--geo–year.csv

geo	**year**	**population**
swe	2015	9799000
ukr	2015	45150000

ddf--datapoints–population--by--country–year.csv

country	**year**	**population**
swe	2015	9799000
ger	2015	81690000

ddf--datapoints–population--by--country–year--gender.csv

country	**year**	**gender**	**population**
ukr	2015	male	22230000
ger	2015	male	40800000

The above example shows **between-file redundancy**. Between-file redundancy is allowed in DDFcsv. It allows for separate csv files to contain full timeseries. For example, time series for population of both cities and countries might contain data for Singapore. Because we allow between-file redundancy, both the ddf--population–by–country--year.csv and ddf--population–by–city--year.csv can contain the same data for Singapore, making both time series complete.
However, **within-file redundancy** is not allowed. Within-file redundancy is when the same data occurs within one file. In other words, the same key is repeated in one file. An example of within-file redundancy is shown below, where Sweden in 2015 is repeated twice in one file.

ddf--datapoints–population--by--geo–year.csv

geo	**year**	**population**
swe	2015	9799000
ukr	2015	45150000
swe	2015	9799000

10. Synonyms

A synonym is a string which can identify a concept or entity in the dataset. The synonyms collection contains entity and concept ids with their synonyms.
E.g.: The country entity `deu` has synonyms "ger", "de" and "Federal Republic of Germany".

10.1 File name

Synonyms are stored in one folder per language, using the following naming convention

```
ddf--synonyms--<synonym_set>
```

Where

- `<synonym_set>` MUST be `concept` when the file defines concept synonyms. `<synonym_set>` MUST be an entity set or entity domain in the dataset when the dictionary defines entity synonyms.

For example

```
ddf--synonyms--concept.csv
ddf--synonyms--country.csv
ddf--synonyms--geo.csv
ddf--synonyms--gender.csv
```

10.2 File format

A synonym file MUST contain a primary key consisting of the concept `synonym` and the concept `<synonym_set>` from the file name. It MAY contain values to the primary key. These values are properties of the synonym entry.

key		*value*
country	**synonym**	**source**
deu	Deutschland	German name
deu	Germany	English name
deu	de	ISO 3166-1 alpha-2
deu	Federal Republic of	Full english name

	Germany	
deu	276	ISO 3166-1 numeric
swe	Sweden	
swe	se	ISO 3166-1 alpha-2
...	...	...

key

concept	**synonym**
population_total	population
population_total	pop
life_expectancy	lex
life_expectancy	Life expectancy at birth
...	...