

Guide to Using Parquet Files with Stata

BPLIM

2025-01-23

Contents

1.	What is Parquet?	1
2.	Key Features of Parquet	2
2.1.	Store Layout and Predicate Pushdown	2
2.2.	Efficient Data Storage	2
2.3.	Partitioning	2
3.	Advantages of Parquet	2
4.	How to handle Parquet files in Stata?	3
4.1	Describe Parquet Files in Stata - <code>pq describe</code>	3
4.2	Read Parquet Files in Stata - <code>pq use</code>	5
4.3.	Save Parquet Files in Stata - <code>pq save</code>	10
4.4.	Merging Parquet Files - <code>pq merge</code>	14
4.5.	Appending Parquet Files - <code>pq append</code>	22
5.	Conclusion	26

1. What is Parquet?

Parquet is an open-source file format designed for efficient data storage and retrieval. Parquet is particularly useful when working with large datasets because it offers faster read times and significantly reduced file sizes compared to `.dta` files.¹ The **stata-parquet-io** package, developed by John Rothbaum, allows you to efficiently handle Parquet files directly in Stata.

For installation instructions, visit the **stata-parquet-io** package [GitHub](#) page and consult the help files with `help pq` for more information. Make sure you are using the latest version of the package (version 1.9.1 as of this date). This package is already available in BPLIM containers.

This guide includes example usage of `pq` and does not replace the `pq` documentation. Please refer to the **stata-parquet-io** [GitHub](#) page for full details.

¹All examples in this guide used Stata version 19.5.

2. Key Features of Parquet

2.1. Store Layout and Predicate Pushdown

In Stata, the system first retrieves all data from the data source and then applies the filtering conditions during processing, even when using the `if()` option in the `use` command. This means that all the data, including irrelevant rows and columns, must be transferred and processed, which can be inefficient for large datasets.

However, Parquet uses columnar storage with row groups, meaning that data is stored in column chunks. This structure allows filtering to occur at the data source, transferring and processing only the rows that satisfy the condition and the selected columns. This results in **faster reads and more efficient queries**.

2.2. Efficient Data Storage

Parquet uses two main encoding types that help compress data, reducing the memory footprint of the file. The first is **dictionary encoding**, in which Parquet creates a dictionary of unique column values and replaces them with compact integer references from the dictionary. The second is **run-length encoding (RLE)**. When a column contains many repeated values, Parquet stores each run as a pair of value-count, further reducing storage needs.

Additionally, Parquet also supports multiple **compression algorithms** that significantly reduce data size.

2.3. Partitioning

Parquet allows large files to be divided into smaller chunks, which involves **dividing data into subdirectories** based on the values of one or more columns.

The main advantage is that if queries involve the partitioning columns, Parquet can skip reading entire directories that don't match the filter conditions, leading to faster query performance. This minimizes the amount of data read from disk, reduces I/O operations, and keeps data well organized.

3. Advantages of Parquet

Considering the key features of Parquet explained above, the **main advantages** of Parquet are:

- Parquet is an open-source format;
- Thanks to its hybrid storage layout and partitioning, Parquet supports fast read operations;
- Parquet provides efficient data compression, reducing storage requirements both in memory and on disk;

- Parquet is compatible with multiple languages and tools, including Python, R, Julia, Stata and many processing frameworks.

4. How to handle Parquet files in Stata?

The Stata user-written command `pq` (`stata-parquet-io` package) allows to describe, use, save, merge and append Parquet files. There is also an official Stata command `import parquet`, but it is less flexible.

4.1 Describe Parquet Files in Stata - `pq describe`

To describe Parquet files in Stata use the subcommand `pq describe`:

```
pq describe using filename
```

This subcommand describes the content of a Parquet file and lists the available variables, even before using the dataset.

Before reading a Parquet file, it is a good practice to use this subcommand to know the structure of the file and define the variables to be used next. However, be aware that a major limitation of Parquet files is that they do not preserve the metadata (value labels, variable labels, variable formats, etc.) meaning that the `pq describe` subcommand only shows the variable name and type. BPLIM always makes complete metadata files available (*metaxl* files) along with the Parquet data files.

Example 1

The dataset `panel_GR_2010_2018.dta` is used as an illustrative example to compare the use and performance of `.dta` and `.parquet` files. In this example, the associated metadata file is named “`META_PANEL_GR.xlsx`”. This comparison depends on the complexity, size, and structure of the data, meaning that performance may vary. Each observation is uniquely identified by the combination of worker (variable `niss_ps_encrip`), firm (variable `tina_ee`), and month (variable `cod_mes`). This dataset contains more than 775 million observations between January 2010 and December 2018, and the corresponding `.dta` file size is 30.3 GB.

For the `.dta` dataset:

```
describe using "panel_GR_2010_2018.dta"
```

Contains data

Observations: 775,773,431

27 Nov 2025 10:16

Variables: 10

Variable	Storage	Display	Value
----------	---------	---------	-------

name	type	format	label	Variable label
year	int	%8.0g		Ano de referência dos dados
cod_mes	int	%tm		Mês de referencia da remuneração
tina_ee	long	%10.0f		Número de identificação fiscal anonimizado da entidade empregadora
cod_estabelecimento	int	%8.0g		Código do estabelecimento do trabalhador
niss_ps_encrip	double	%15.0f		Número de Identificação da Segurança Social da Pessoa Singular - encriptado
qtd_numero_dias	double	%10.0g		Total de dias da remuneração
val_remun	double	%10.0g		Valor total da remuneração
cod_regime	int	%8.0g	COD_REGIME	Regime
tipo_qualificacao	int	%91.0g	COD_TIPO_QUALIFICACAO	Tipo de qualificação
nat_remuneracao	int	%73.0g	COD_NAT_Remuneracao	Natureza da remuneração

Sorted by:

For the same dataset in *.parquet*:

```
pq describe using "panel_GR_2010_2018.parquet"
```

Variable Name	Polars Type	Stata Typ
> e		
> -----		
year	Int16	long
cod_mes	Date	date
tina_ee	Int32	long
cod_estabelecimento	Int16	long
niss_ps_encrip	Float64	double
qtd_numero_dias	Float64	double
val_remun	Float64	double
cod_regime	Int16	long
tipo_qualificacao	Int16	long
nat_remuneracao	Int16	long

```
n columns = 10
n rows = 775773431
```

4.2 Read Parquet Files in Stata - pq use

To import Parquet files into Stata use the subcommand **pq use**:

```
 pq use using filename
```

The subcommand also allows to:

- Select the variables that you want to import into Stata;
- Specify a subset of rows to read (*in* option);
- Import only the rows that satisfy a specified condition (*if* option);
- Sort the data by specified variables during the read operation (*sort* option);
- Read data concurrently and improve efficiency (*parallelize* option). For tall datasets (many rows) use *parallelize(rows)* and for wide datasets (many columns) use *parallelize(columns)*.

As mentioned before, Parquet files do not preserve the metadata. To overcome this limitation, BPLIM **metaxl** tool can be used after importing the data into Stata. The **metaxl** package provides multiple solutions to help users handle metadata and is available on [BPLIM's GitHub](#) and all BPLIM containers. The subcommand **metaxl apply** applies metadata saved in an Excel file to the data in memory. For more information, please consult the help files.

Examples

Example 2: Reading a complete dataset

For the *.dta* dataset:

```
use "panel_GR_2010_2018.dta", clear  
count
```

775,773,431

For the same dataset in *.parquet*:

```
pq use using "panel_GR_2010_2018.parquet", clear  
count
```

775,773,431

In this example, reading either the *.dta* or *.parquet* dataset takes roughly the same amount of time (approximately 2 minutes).

Example 3: Reading a set of variables

In most cases, we do not need to use the entire dataset; reading only a small subset of variables is sufficient.

For the *.dta* dataset:

```
use cod_mes niss_ps_encrip tina_ee val_remun using "panel_GR_2010_2018.dta", clear  
count
```

775,773,431

For the same dataset in *.parquet*:

```
pq use cod_mes niss_ps_encrip tina_ee val_remun using "panel_GR_2010_2018.parquet", clear  
count
```

775,773,431

In this example, reading a *.parquet* file takes less than 1 minute, while a *.dta* file takes 2 minutes.

Example 4: Reading a partitioned dataset

As mentioned above, Parquet files can be partitioned according to one or more variables. To read a complete partitioned *.parquet* dataset, use `pq use` as usual and specify the main Parquet file name without the need to identify any partition:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear  
tab year, miss
```

year	Freq.	Percent	Cum.
2010	59,562,885	7.68	7.68
2011	67,904,889	8.75	16.43
2012	73,110,643	9.42	25.86
2013	88,307,366	11.38	37.24
2014	89,624,277	11.55	48.79
2015	93,068,353	12.00	60.79
2016	98,479,849	12.69	73.48
2017	104,361,789	13.45	86.94
2018	101,353,380	13.06	100.00
Total	775,773,431	100.00	

In this example, reading a partitioned or a non-partitioned parquet dataset takes approximately the same time.

Example 5: Reading data for a specific year

For the *.dta* dataset, Stata must load the entire data into memory before applying any filtering conditions, even when using the *if* option:

```
use if year == 2015 using "panel_GR_2010_2018.dta", clear  
tab year, miss
```

Ano de referência dos dados	Freq.	Percent	Cum.
2015 93,068,353	100.00	100.00	
Total 93,068,353	100.00		

When using a *.parquet* file partitioned by year, it is possible to read only the required partition:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015)  
tab year
```

year	Freq.	Percent	Cum.
2015 93,068,353	100.00	100.00	
Total 93,068,353	100.00		

In this example, loading one year of data takes over 3 minutes when using *.dta* files, whereas using partitioned *.parquet* files only takes a few seconds.

Example 6: Combining filters

You can combine filters using partition variables along with other conditions. For instance, it is possible to load only the 2015 partition while also selecting workers with *tipo_qualificacao* equal to 1071:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015 & tipo_qualificacao == 1071)  
tab year, miss  
tab tipo_qualificacao, miss
```

year	Freq.	Percent	Cum.
2015	88,244,244	100.00	100.00
Total	88,244,244	100.00	

tipo_qualif	icacao	Freq.	Percent	Cum.
1071		88,244,244	100.00	100.00
Total		88,244,244	100.00	

Tip: Filtering dates

The `if` option filters observations during data import by translating Stata syntax into SQL for Parquet filtering.

While most Stata expressions behave as expected, datetime functions are not supported in this context (as of version 1.9.1). For **example**, the following command will result in an error:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015 & cod_mes >= td(01nov2015))
```

In addition, Stata dates are measure as the number of days since 1 January 1960, whereas Polars uses the Unix epoch (1 January 1970) as its reference. Consequently, numeric Stata dates cannot be used directly in Parquet filters. For **example**, 1 November 2015 corresponds to 20393 days in Stata, but the following command will import no observations:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015 & cod_mes >= 20393)
```

Therefore, when filtering on dates, users must use Polars syntax. In the previous **example**, the correct filter condition to import observations with dates on or after 1 November 2015 is as follows:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015 & cod_mes >= date('01nov2015', '%d%b%Y'))
```

Example 7: Using `metaxl` to apply metadata

As mentioned before, `.parquet` files do not preserve the metadata such as variable and value labels or storage types:

```
pq use using "panel_GR_2010_2018_byear.parquet", clear if(year == 2015)
describe
```

```

Contains data
Observations: 93,068,353
Variables: 10
-----
Variable      Storage   Display    Value
      name        type     format   label      Variable label
-----
year          long      %12.0g
cod_mes       long      %td
tina_ee       long      %12.0g
cod_estabelec~o long      %12.0g
niss_ps_encrip double    %10.0g
qtd_numero_dias double    %10.0g
val_remun     double    %10.0g
cod_regime    long      %12.0g
tipo_qualific~o long      %12.0g
nat_remuneracao long      %12.0g

```

Sorted by:

Note: Dataset has changed since last saved.

To apply the metadata to a *.parquet* file use the `metaxl apply` subcommand, specifying the name of the metafile in the option *meta*:

```
metaxl apply, meta("META_PANEL_GR") nobackup
```

```
describe
```

```

Contains data
Observations: 93,068,353
Variables: 10
-----
Variable      Storage   Display    Value
      name        type     format   label      Variable label
-----
year          int       %9.0g           Ano de referência dos dados
cod_mes       int       %tm            Mês de referencia da remuneração
tina_ee       long      %10.0f          Número de identificação fiscal anonimizado
cod_estabelec~o int      %8.0g          Código do estabelecimento do trabalhador
niss_ps_encrip double    %15.0f          Número de Identificação da Segurança Social
qtd_numero_dias double    %10.0g          Total de dias da remuneração
val_remun     double    %10.0g          Valor total da remuneração
cod_regime    int       %8.0g           COD_REGIME

```

		Regime
tipo_qualific~o int	%91.0g	COD_TIPO_QUALIFICACAO Tipo de qualificação
nat_remuneracao int	%73.0g	COD_NAT_Remuneracao Natureza da remuneração

Sorted by:

Note: Dataset has changed since last saved.

Tip: Extract Metadata

If you don't have a metadata file, or you modified the existing metadata, you can use the `metaxl extract` subcommand to save a new Excel metadata file for later use. For example, to replace the existing metadata file ("META_PANEL_GR.xlsx") use the following command:

```
metaxl extract, metafile("META_PANEL_GR") replace nobackup
```

4.3. Save Parquet Files in Stata - pq save

To save data as a Parquet file use the `pq save` subcommand:

```
pq save using filename
```

The subcommand allows to:

- Select the variables to save;
- Overwrite an existing Parquet file (*replace* option);
- Save only rows that satisfy a specified condition (*if* option);
- Create a partitioned Parquet dataset (*partition_by* option);
- Add information without overwriting existing partitions (*nopartitionoverwrite* option). All partitions must have the same variable names and types; otherwise, the data will not be saved correctly and reading the dataset will fail;
- Specify the compression algorithm to use in the saved Parquet file, with *zstd* used by default (*compression* option);
- Compress the data during the read operation to reduce memory usage (*compress* option). When working with Stata variables that require high numeric precision, this option should be used with caution, as it can change storage types and result in a loss of precision.

Examples

Example 8: Saving a complete dataset

To save as *.dta*:

```
sort niss_ps_encrip tina_ee cod_mes  
save "panel_GR_2010_2018_v1.dta", replace
```

To save the same data as *.parquet*:

```
sort niss_ps_encrip tina_ee cod_mes  
pq save using "panel_GR_2010_2018_v1.parquet", replace
```

In this example, saving the *.dta* takes 5 minutes, while saving the *.parquet* takes approximately 3 minutes. Additionally, the *.dta* file is 30.3 GB, whereas the corresponding *.parquet* file is only 2.4 GB — a reduction in file size of over **90%**.

Tip: Sort

The way you sort the data before saving it as a *.parquet* affects the size of the Parquet file.

As a general rule, the data should be sorted by the combined key, with the variable that has the most unique values first. However, depending on the structure of the data, this may not be the most efficient option.

In the previous **example** the data are sorted by *niss_ps_encrip* (5 577 919 unique values), *tina_ee* (765 590 unique values) and *cod_mes* (108 unique values). This sorting order reduces file size significantly: the dataset is **2.4 GB**, compared with **5.4 GB** when unsorted.

Example 9: Saving a set of variables

To save as *.dta*:

```
keep cod_mes niss_ps_encrip tina_ee val_remun  
sort niss_ps_encrip tina_ee cod_mes  
save "panel_GR_2010_2018_v2.dta", replace
```

To save the same data as *.parquet* there is no need to modify the data in memory:

```
sort niss_ps_encrip tina_ee cod_mes  
pq save cod_mes niss_ps_encrip tina_ee val_remun using "panel_GR_2010_2018_v2.parquet", re
```

In this example, saving a *.dta* file or a *.parquet* file takes 5 minutes. However, the *.dta* file is 15.9 GB, whereas the *.parquet* is 557 MB.

Example 10: Saving data for a specific year

To save as *.dta*:

```
keep if year == 2015  
sort niss_ps_encrip tina_ee cod_mes  
save "panel_GR_2015.dta", replace
```

Again, to save the same data as *.parquet* there is no need to modify the data in memory:

```
sort niss_ps_encrip tina_ee cod_mes  
pq save using "panel_GR_2015.parquet", replace if(year == 2015)
```

In this example, the *.dta* file is 3.6 GB, whereas the *.parquet* is 310 MB, and saving both takes approximately the same time (less than 2 minutes).

Example 11: Saving a partitioned dataset by year

To save a *.parquet* file with partitions by year:

```
sort niss_ps_encrip tina_ee cod_mes  
pq save using "panel_GR_2010_2018_byyear.parquet", replace partition_by(year)
```

The result is a folder **panel_GR_2010_2018_byyear.parquet** divided into subfolders, one for each year.

panel_GR_2010_2018_byear.parquet	
Name	
 year=2010	
 year=2011	
 year=2012	
 year=2013	
 year=2014	
 year=2015	
 year=2016	
 year=2017	
 year=2018	

In this example, saving a `.parquet` file with partitions takes roughly 5 minutes and each folder is 360 MB maximum.

Tip: Date type variables (e.g. monthly, quartely, ...)

The `pq save` subcommand correctly saves date variables in daily format, allowing them to be read properly by other software such as R, Python, and Julia. This is not the case for monthly, quarterly, or other non-daily formats. Therefore, we recommend conversion of all date variables to a daily format before saving the data in Parquet format.

To convert monthly dates to daily dates (using the first day of the month as default):

```
gen cod_mes2 = dofm(cod_mes)
format cod_mes2 %td
```

To convert quarterly dates to daily dates (using the first day and month of the quarter as default):

```
gen cod_mes3 = dofq(cod_mes)
format cod_mes3 %td
```

4.4. Merging Parquet Files - pq merge

To merge a Parquet file with data in memory, use the **pq merge** subcommand:

```
pq merge merge_type using filename
```

The subcommand allows to:

- Specify which observations to keep after merging (*keep* option);
- Specify which variables to keep from the using dataset (*keepusing* option);
- Replace the missing values in the master dataset with values from the using dataset (*update* option).

Examples

Consider that the dataset **panel_GR_2010_2018.dta** is merged with the dataset **QLF_2023_2025.dta**, which contains information on worker-firm relationship (variables *niss_ps_encrip* and *tina_ee*, respectively):

```
describe
```

```
Contains data from QLF_2023_2025.dta
Observations:      32,620,722
Variables:          10
Name             Storage   Display    Value
           type     format   label
-----Variable label
tina_ee        long      %10.0f      Número de identificação fiscal anonimizado d
niss_ps_encrip double    %15.0f      Número de Identificação da Segurança Social
                                      encriptado
codigo_qualif~o long      %12.0g      Número identificativo do enquadramento da Pe
cod_regime      int       %8.0g       COD_REGIME
tipos_qualific~o int      %91.0g      COD_TIPO_QUALIFICACAO
motivo_fim_qu~o int      %123.0g     COD_MOTIVO_FIM_QUALIFICACAO
data_inicio_q~o int       %tm        Data de início de qualificação
data_fim_qual~o int       %tm        Data de fim de qualificação (se qualificaç~
data_inicio_a~e int       %tm        Data de início de atividade de um trabalhado
datacessacao     int       %tm        Data de cessação da atividade de um trabalha
```

Sorted by: niss_ps_encrip tina_ee

Example 12: Merging two datasets

For the *.dta* datasets:

```
use "panel_GR_2010_2018.dta", clear
merge m:1 niss_ps_encrip tina_ee using "QLF_2023_2025.dta", keepusing(data_inicio_qualificacion)

(label COD_REGIME already defined)
(label COD_TIPO_QUALIFICACAO already defined)

Result                               Number of obs
-----
Not matched                         20,163,445
    from master                      55,682  (_merge==1)
    from using                       20,107,763  (_merge==2)

Matched                             775,717,749  (_merge==3)
-----
```

For the same datasets in *.parquet*:

```
pq use using "panel_GR_2010_2018.parquet", clear
pq merge m:1 niss_ps_encrip tina_ee using "QLF_2023_2025.parquet", keepusing(data_inicio_qualificacion)
```

Loading parquet file and saving to temporary dta

Variable	Obs	Mean	Std. dev.	Min	Max
niss_ps_en~p	32,620,722	1.49e+10	2.89e+09	1.00e+10	1.99e+10
tina_ee	23,989,747	5.31e+08	9.65e+07	1.00e+08	1.00e+09
data_inicio	32,329,358	657.5949	108.7434	-980	787
data_fim_q~o	22,927,664	694.2574	54.25436	600	1822

Merging to data

```
Result                               Number of obs
-----
Not matched                         20,163,445
    from master                      55,682  (_merge==1)
    from using                       20,107,763  (_merge==2)

Matched                             775,717,749  (_merge==3)
-----
```

In this example, merging *.dta* files takes 7 minutes, whereas merging *.parquet* files takes 8 minutes.

Tip: Sort

The merge varlist should follow the same order as the sorting of the master dataset to optimize performance.

In this example, the data are first sorted by *niss_ps_encrip* and then by *tina_ee*. The merge varlist should be defined in this same order. Reversing the order increases the merge time from 8 minutes to 12.

Example 13: Applying metadata to merged datasets

When merging two datasets, it is necessary to apply the metadata from both the master and the using dataset. There are two ways to do this using the BPLIM `metaxl` package:

- The first option is to apply two separate metadata files — one for each dataset. Apply the first metadata file and then, when applying the second with `metaxl apply`, use the `skip` option to preserve the metadata already applied.
- The second option is to merge the two metadata files into a single one using the `metaxl combine` subcommand, and then apply the combined metadata file using the `metaxl apply` as usual.

The best solution depends on the datasets. If both datasets share common variables but differ in storage types, variable labels or value label categories, using `metaxl combine` is recommended to avoid inconsistencies. Regardless of the method, ensure you are using version 0.7 or later of the `metaxl` package, when applying metadata to a merged dataset.

Example 13.1: `metaxl apply` with `skip` option

First, import the `panel_GR_2010_2018.parquet` dataset and apply the corresponding metadata file - “`META_PANEL_GR.xlsx`” - with `metaxl apply`:

```
 pq use using "panel_GR_2010_2018.parquet", clear  
 metaxl apply, meta("META_PANEL_GR") nobackup
```

Next, add the variables `data_inicio_qualificacao` and `data_fim_qualificacao` from the `QLF_2023_2025.parquet` dataset, using `pq merge` as shown earlier:

```
 pq merge m:1 niss_ps_encrip tina_ee using "QLF_2023_2025.parquet", keepusing(data_inicio_q  
 drop _merge  
 describe
```

Loading parquet file and saving to temporary dta

Variable	Obs	Mean	Std. dev.	Min	Max
niss_ps_en~p	32,620,722	1.49e+10	2.89e+09	1.00e+10	1.99e+10
tina_ee	23,989,747	5.31e+08	9.65e+07	1.00e+08	1.00e+09

```

data_inicio | 32,329,358      657.5949     108.7434      -980       787
data_fim_q~o | 22,927,664      694.2574     54.25436      600      1822
Merging to data

```

Result	Number of obs
Not matched	20,163,445
from master	55,682 (_merge==1)
from using	20,107,763 (_merge==2)
Matched	775,717,749 (_merge==3)

Contains data

Observations: 795,881,194
Variables: 12

Variable	Storage	Display	Value	
name	type	format	label	Variable label
year	int	%9.0g		Ano de referência dos dados
cod_mes	int	%tm		Mês de referencia da remuneração
tina_ee	long	%10.0f		Número de identificação fiscal anonimizado
cod_estabelec~o	int	%8.0g		Código do estabelecimento do trabalhador
niss_ps_encrip	double	%15.0f		Número de Identificação da Segurança Social
qtd_numero_dias	double	%10.0g		Total de dias da remuneração
val_remun	double	%10.0g		Valor total da remuneração
cod_regime	int	%8.0g	COD_REGIME	
				Regime
tipo_qualific~o	int	%91.0g	COD_TIPO_QUALIFICACAO	
				Tipo de qualificação
nat_remuneracao	int	%73.0g	COD_NAT_Remuneracao	
				Natureza da remuneração
data_inicio_q~o	long	%td		
data_fim_qual~o	long	%td		

Sorted by:

Note: Dataset has changed since last saved.

All variables from the master dataset have metadata, unlike the variables merged from the using dataset: *data_inicio_qualificacao* and *data_fim_qualificacao*. We now want to add metadata only for these two variables, skipping those that already include metadata.

To do this, we apply the “META_QLF.xlsx” metadata file with the `metaxl apply` command, specifying the variables already with metadata in the `skip` option:

```
label language default, delete  
metaxl apply, meta("META_QLF") skip(year cod_mes tina_ee cod_estabelecimento niss_ps_encri
```

```
Data characteristics not available in metadata file  
Skipping variable cod_mes  
Skipping variable cod_regime  
variable "codigo_qualificacao" only available in "META_QLF.xlsx"  
variable "data_cessacao" only available in "META_QLF.xlsx"  
variable "data_inicio_actividade" only available in "META_QLF.xlsx"  
variable "motivo_fim_qualificacao" only available in "META_QLF.xlsx"  
Skipping variable niss_ps_encrip  
Skipping variable tina_ee  
Skipping variable tipo_qualificacao  
Skipping variable cod_estabelecimento  
Skipping variable nat_remuneracao  
Skipping variable qtd_numero_dias  
Skipping variable val_remun  
Skipping variable year
```

```
describe
```

Contains data

Observations: 795,881,194
Variables: 12

Variable name	Storage type	Display format	Value label	Variable label
year	int	%9.0g		Ano de referência dos dados
cod_mes	int	%tm		Mês de referencia da remuneração
tina_ee	long	%10.0f		Número de identificação fiscal anonimizado
cod_estabelec~o	int	%8.0g		Código do estabelecimento do trabalhador
niss_ps_encrip	double	%15.0f		Número de Identificação da Segurança Social
qtd_numero_dias	double	%10.0g		Total de dias da remuneração
val_remun	double	%10.0g		Valor total da remuneração
cod_regime	int	%8.0g	COD_REGIME	Regime
tipo_qualific~o	int	%91.0g	COD_TIPO_QUALIFICACAO	Tipo de qualificação
nat_remuneracao	int	%73.0g	COD_NAT_Remuneracao	Natureza da remuneração
data_inicio_q~o	int	%tm		Data de início de qualificação

```

data_fim_qual~o int      %tm          Data de fim de qualificação (se qualificaç~ao
-----
Sorted by:
Note: Dataset has changed since last saved.

```

Tip: Language

In Stata, variable and value labels may exist in multiple languages — for example, Portuguese (pt) and English (en). You can use the `label language` command to check which languages are defined in the data. By default, Stata creates a label language called “default”.

To successfully use the `skip` option in `metaxl apply`, one condition must be met: the label language in the data must match at least one of the languages of the metadata being applied.

In the previous **example**, the data include two languages: “pt” and “default”:

The metadata to be applied use the “pt” language:

Features	Content
File	panel_GR_2010_2018.dta
Data Label	
Sorted by	
Label languages	pt

< > **data_features_gen** data_features_spec |

Therefore, the “default” language must be removed with the command `label language default, delete`.

Example 13.2: metaxl combine

First, use the `metaxl combine` subcommand to combine the metadata files “META_PANEL_GR.xlsx” and “META_QLF.xlsx”, specifying which file should be prioritized in case of inconsistent metadata with the option `keep`.

In this example, the option `keep(f1)` indicates that, in the case of inconsistencies between both metadata files (e.g., duplicate variable or value labels), the ones specified in the first metadata file (f1) will be kept:

```
metaxl combine, f1("META_PANEL_GR") f2("META_QLF") keep(f1) meta("META_ALL") replace
```

Worksheets found only in META_PANEL_GR.xlsx: vl_COD_NAT_REMUNERACAO char_dta

Worksheets found only in META_QLF.xlsx: vl_COD_MOTIVO_FIM_QUALIFICACAO

Combining meta files META_PANEL_GR.xlsx and META_QLF.xlsx

[data_features_gen] Removed 3 duplicates based on Features. Kept observations from file f1
[variables] Removed 2 duplicates based on variable. Kept observations from file f1

File META_ALL.xlsx saved

Then merge the .parquet files as shown before:

```
 pq use using "panel_GR_2010_2018.parquet", clear
 pq merge m:1 niss_ps_encrip tina_ee using "QLF_2023_2025.parquet", keepusing(data_inicio_q
 drop _merge
 describe
```

Loading parquet file and saving to temporary dta

Variable	Obs	Mean	Std. dev.	Min	Max
niss_ps_en~p	32,620,722	1.49e+10	2.89e+09	1.00e+10	1.99e+10
tina_ee	23,989,747	5.31e+08	9.65e+07	1.00e+08	1.00e+09
data_inicio	32,329,358	657.5949	108.7434	-980	787
data_fim_q~o	22,927,664	694.2574	54.25436	600	1822

Merging to data

Result	Number of obs
Not matched	20,163,445
from master	55,682 (_merge==1)
from using	20,107,763 (_merge==2)
Matched	775,717,749 (_merge==3)

Contains data

Observations: 795,881,194

Variables: 12

Variable	Storage	Display	Value
name	type	format	label
year	long	%12.0g	
cod_mes	long	%td	
tina_ee	long	%12.0g	
cod_estabelec~o	long	%12.0g	
niss_ps_encrip	double	%10.0g	

```

qtd_numero_dias double %10.0g
val_remun double %10.0g
cod_regime long %12.0g
tipo_qualific~o long %12.0g
nat_remuneracao long %12.0g
data_inicio_q~o long %td
data_fim_qual~o long %td

```

Sorted by:

Note: Dataset has changed since last saved.

Finally, apply the combined metadata file (“META_ALL”) to the merged dataset using `metaxl apply`. This option will replace all the metadata in memory with the one contained in the combined metadata file:

```
metaxl apply, meta("META_ALL") nobackup
```

```
describe
```

Contains data

Observations: 795,881,194
Variables: 12

Variable name	Storage type	Display format	Value label	Variable label
year	int	%9.0g		Ano de referência dos dados
cod_mes	int	%tm		Mês de referencia da remuneração
tina_ee	long	%10.0f		Número de identificação fiscal anonimizado
cod_estabelec~o	int	%8.0g		Código do estabelecimento do trabalhador
niss_ps_encrip	double	%15.0f		Número de Identificação da Segurança Social encriptado
qtd_numero_dias	double	%10.0g		Total de dias da remuneração
val_remun	double	%10.0g		Valor total da remuneração
cod_regime	int	%8.0g	COD_REGIME	Regime
tipo_qualific~o	int	%91.0g	COD_TIPO_QUALIFICACAO	Tipo de qualificação
nat_remuneracao	int	%73.0g	COD_NAT_REMUNERACAO	Natureza da remuneração
data_inicio_q~o	int	%tm		Data de início de qualificação
data_fim_qual~o	int	%tm		Data de fim de qualificação (se qualificaç~o)

Sorted by:

Note: Dataset has changed since last saved.

In this example, using `metaxl apply` with the `skip` option or using `metaxl combine` takes roughly the same amount of time. Therefore, the better choice depends on the specifics of each case, such as whether the datasets share common variables or whether value labels categories differ.

4.5. Appending Parquet Files - `pq append`

To append a Parquet file to data in memory, use the `pq append` subcommand:

```
pq append using filename
```

All `pq use` options are also available for `pq append`.

Examples

Consider that the dataset `panel_GR_2010_2018.dta`, containing data between January 2010 and December 2018, is appended to a similar dataset `panel_GR_2019_2025.dta`, with data from January 2019 to August 2025.

Example 14: Appending two datasets

For the `.dta` datasets:

```
use "panel_GR_2010_2018.dta", clear
tab year, miss
```

Ano de referência dos dados	Freq.	Percent	Cum.
2010	59,562,885	7.68	7.68
2011	67,904,889	8.75	16.43
2012	73,110,643	9.42	25.86
2013	88,307,366	11.38	37.24
2014	89,624,277	11.55	48.79
2015	93,068,353	12.00	60.79
2016	98,479,849	12.69	73.48
2017	104,361,789	13.45	86.94
2018	101,353,380	13.06	100.00
Total	775,773,431		100.00

```

append using "panel_GR_2019_2025.dta"
tab year, miss

(label COD_TIPO_QUALIFICACAO already defined)
(label COD_NAT_REMUNERACAO already defined)
(label COD_REGIME already defined)

```

Ano de referência dos dados	Freq.	Percent	Cum.
2010	59562885	3.84	3.84
2011	67904889	4.37	8.21
2012	73110643	4.71	12.92
2013	88307366	5.69	18.61
2014	89624277	5.77	24.38
2015	93068353	5.99	30.37
2016	98479849	6.34	36.71
2017	104361789	6.72	43.44
2018	101353380	6.53	49.96
2019	106638447	6.87	56.83
2020	105939646	6.82	63.65
2021	110393257	7.11	70.76
2022	119353366	7.69	78.45
2023	123472821	7.95	86.40
2024	126249498	8.13	94.53
2025	84881661	5.47	100.00
Total	1552702127	100.00	

For the same datasets in *.parquet*:

```

pq use using "panel_GR_2010_2018.parquet", clear
tab year, miss

```

year	Freq.	Percent	Cum.
2010	59,562,885	7.68	7.68
2011	67,904,889	8.75	16.43
2012	73,110,643	9.42	25.86
2013	88,307,366	11.38	37.24
2014	89,624,277	11.55	48.79
2015	93,068,353	12.00	60.79

2016 98,479,849	12.69	73.48
2017 104,361,789	13.45	86.94
2018 101,353,380	13.06	100.00
Total 775,773,431		100.00

```
 pq append using "panel_GR_2019_2025.parquet"
 tab year, miss
```

year	Freq.	Percent	Cum.
2010 59562885	3.84	3.84	
2011 67904889	4.37	8.21	
2012 73110643	4.71	12.92	
2013 88307366	5.69	18.61	
2014 89624277	5.77	24.38	
2015 93068353	5.99	30.37	
2016 98479849	6.34	36.71	
2017 104361789	6.72	43.44	
2018 101353380	6.53	49.96	
2019 106638447	6.87	56.83	
2020 105939646	6.82	63.65	
2021 110393257	7.11	70.76	
2022 119353366	7.69	78.45	
2023 123472821	7.95	86.40	
2024 126249498	8.13	94.53	
2025 84881661	5.47	100.00	
Total 1552702127		100.00	

In this example, the append takes 4 minutes with *.dta* files and 2 minutes with *.parquet* files.

Tip: Metadata

As the dataset gets larger, the `metaxl apply` subcommand will take more time to run. When the metadata is valid for both datasets, apply it to the first dataset in memory and then execute `pq append` to achieve better performance. For example:

```
 pq use using "panel_GR_2010_2018.parquet", clear
 metaxl apply, meta("META_PANEL_GR") nobackup
 pq append using "panel_GR_2019_2025.parquet"
```

However, if different metadata files are required, use one of the options described above: `metaxl apply` with the *skip* option, or `metaxl combine` followed by `metaxl apply`.

Example 15: Appending a set of variables

```
 pq use using "panel_GR_2010_2018.parquet", clear  
 tab year, miss
```

year	Freq.	Percent	Cum.
2010	59,562,885	7.68	7.68
2011	67,904,889	8.75	16.43
2012	73,110,643	9.42	25.86
2013	88,307,366	11.38	37.24
2014	89,624,277	11.55	48.79
2015	93,068,353	12.00	60.79
2016	98,479,849	12.69	73.48
2017	104,361,789	13.45	86.94
2018	101,353,380	13.06	100.00
Total	775,773,431	100.00	

```
 pq append year cod_mes niss_ps_encrip val_remun using "panel_GR_2019_2025.parquet"  
 tab year, miss
```

year	Freq.	Percent	Cum.
2010	59562885	3.84	3.84
2011	67904889	4.37	8.21
2012	73110643	4.71	12.92
2013	88307366	5.69	18.61
2014	89624277	5.77	24.38
2015	93068353	5.99	30.37
2016	98479849	6.34	36.71
2017	104361789	6.72	43.44
2018	101353380	6.53	49.96
2019	106638447	6.87	56.83
2020	105939646	6.82	63.65
2021	110393257	7.11	70.76
2022	119353366	7.69	78.45
2023	123472821	7.95	86.40
2024	126249498	8.13	94.53
2025	84881661	5.47	100.00
Total	1552702127	100.00	

In this example, appending all the variables takes 2 minutes, while appending only a set takes only a few seconds.

Tip: Package version

Only use the `pq append` subcommand with version 1.9.0 or later. Previous versions will not work properly.

5. Conclusion

In conclusion, Parquet files offer significant advantages in terms of storage efficiency. Moreover, when used properly, they can outperform `.dta` files in processing speed.

Below is a summary of the key recommendations discussed above for an efficient use of Parquet files:

1. In reading Parquet files, always use the `parallelize` option;
2. Read only the necessary variables;
3. Use `metaxl` to apply the metadata to Parquet files;
4. Save Parquet files with partitions that will be used to filter the data;
5. Before saving a Parquet file, ensure that monthly and quarterly dates are saved in a daily format;
6. Sort the dataset by the key variables before saving it in a Parquet file to more efficient storage;
7. In merging Parquet files, ensure that the order of the keys in the subcommand corresponds to the sorting order of the data.