

# Package ‘chunked’

October 12, 2022

**Type** Package

**Title** Chunkwise Text-File Processing for 'dplyr'

**Version** 0.6.0

**Description** Data stored in text file can be processed chunkwise using 'dplyr' commands. These are recorded and executed per data chunk, so large files can be processed with limited memory using the 'LaF' package.

**License** GPL-2

**BugReports** <https://github.com/edwindj/chunked/issues>

**URL** <https://github.com/edwindj/chunked>

**Depends** dplyr (>= 0.7)

**Imports** LaF, utils, rlang, DBI, progress

**Suggests** testthat, RSQLite, dbplyr

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Edwin de Jonge [aut, cre] (<<https://orcid.org/0000-0002-6580-4718>>)

**Maintainer** Edwin de Jonge <edwindjonge@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-03-07 19:10:02 UTC

## R topics documented:

chunked-package . . . . .	2
insert_chunkwise_into . . . . .	3
read_chunkwise . . . . .	3
read_csv_chunkwise . . . . .	4
write_chunkwise . . . . .	6
write_csv_chunkwise . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

---

`chunked-package`*Chunked*

---

### Description

R is a great tool, but processing large text files with data is cumbersome. `chunked` helps you to process large text files with `dplyr` while loading only a part of the data in memory. It builds on the excellent R package `LaF`. Processing commands are writing in `dplyr` syntax, and `chunked` (using `LaF`) will take care that chunk by chunk is processed, taking far less memory than otherwise. `chunked` is useful for selecting columns, mutating columns and filtering rows. It can be used in data pre-processing.

### Implemented `dplyr` verbs

- `filter`
- `select`
- `rename`
- `mutate`
- `transmute`
- `do`
- `left_join`
- `inner_join`
- `anti_join`
- `semi_join`
- `tbl_vars`
- `collect`

`filter`, `select`, `do`, `left_join`, `inner_join`

### Not implemented

The following operators are not implemented because data in `chunked` is processed chunkwise, so these are not available.

- `full_join`
- `right_join`
- `group_by`
- `arrange`
- `tail`

---

insert\_chunkwise\_into *insert data in chunks into a database*

---

### Description

insert\_chunkwise\_into can be used to insert chunks of data into a database. Typically chunked can be used to for preprocessing data before adding it to a database.

### Usage

```
insert_chunkwise_into(x, dest, table, temporary = FALSE, analyze = FALSE)
```

### Arguments

x	tbl_chunk object
dest	database destination, e.g. src_dbi()
table	name of table
temporary	Should the table be removed when the database connection is closed?
analyze	Should the table be analyzed after import?

### Value

a [tbl](#) object pointing to the table in database dest.

---

read\_chunkwise *Read chunkwise from a data source*

---

### Description

Read chunkwise from a data source

### Usage

```
read_chunkwise(src, chunk_size = 10000L, ...)
```

```
## S3 method for class 'character'
read_chunkwise(
  src,
  chunk_size = 10000L,
  format = c("csv", "csv2", "table"),
  stringsAsFactors = FALSE,
  ...
)
```

```
## S3 method for class 'laf'
read_chunkwise(src, chunk_size = 10000L, ...)

## S3 method for class 'tbl_sql'
read_chunkwise(src, chunk_size = 10000L, ...)
```

### Arguments

src	source to read from
chunk_size	size of the chunks
...	parameters used by specific classes
format	used for specifying type of text file
stringsAsFactors	logical should string be read as factors?

### Value

an object of type `tbl_chunk`

---

read_csv_chunkwise	<i>Read chunkwise data from text files</i>
--------------------	--

---

### Description

`read_csv_chunk` will open a connection to a text file. Subsequent dplyr verbs and commands are recorded until `collect`, `write_csv_chunkwise` is called. In that case the recorded commands will be executed chunk by chunk. This

### Usage

```
read_csv_chunkwise(
  file,
  chunk_size = 10000L,
  header = TRUE,
  sep = ",",
  dec = ".",
  stringsAsFactors = FALSE,
  ...
)
```

```
read_csv2_chunkwise(
  file,
  chunk_size = 10000L,
  header = TRUE,
  sep = ";",
  dec = ",",

```

```

    ...
  )

  read_table_chunkwise(
    file,
    chunk_size = 10000L,
    header = TRUE,
    sep = " ",
    dec = ".",
    ...
  )

  read_laf_chunkwise(laf, chunk_size = 10000L)

```

### Arguments

file	path of text file
chunk_size	size of the chunks to be read
header	Does the csv file have a header with column names?
sep	field separator to be used
dec	decimal separator to be used
stringsAsFactors	logical should string be read as factors?
...	not used
	read_laf_chunkwise reads chunkwise from a LaF object created with laf_open. It offers more control over data specification.
laf	laf object created using LaF

### Details

read\_csv\_chunkwise can be best combined with [write\\_csv\\_chunkwise](#) or [insert\\_chunkwise\\_into](#) (see example)

### Examples

```

# create csv file for demo purpose
in_file <- file.path(tempdir(), "in.csv")
write.csv(women, in_file, row.names = FALSE, quote = FALSE)

#
women_chunked <-
  read_chunkwise(in_file) %>% #open chunkwise connection
  mutate(ratio = weight/height) %>%
  filter(ratio > 2) %>%
  select(height, ratio) %>%
  inner_join(data.frame(height=63:66)) # you can join with data.frames!

# no processing done until

```

```

out_file <- file.path(tempdir(), "processed.csv")
women_chunked %>%
  write_chunkwise(file=out_file)

head(women_chunked) # works (without processing all data...)

iris_file <- file.path(tempdir(), "iris.csv")
write.csv(iris, iris_file, row.names = FALSE, quote= FALSE)

iris_chunked <-
  read_chunkwise(iris_file, chunk_size = 49) %>% # 49 for demo purpose
  group_by(Species) %>%
  summarise(sepal_length = mean(Sepal.Length), n=n()) # note that mean is per chunk

```

---

write_chunkwise	<i>Generic function to write chunk by chunk</i>
-----------------	---

---

## Description

Generic function to write chunk by chunk

## Usage

```

write_chunkwise(x, dest, ...)

## S3 method for class 'chunkwise'
write_chunkwise(
  x,
  dest,
  table,
  file = dest,
  format = c("csv", "csv2", "table"),
  ...
)

```

## Arguments

x	chunked input, e.g. created with read_chunkwise or it can be a tbl_sql object.
dest	where should the data be written. May be a character or a src_sql.
...	parameters that will be passed to the specific implementations.
table	table to write to. Only used when dest is a data base(src_sql)
file	File to write to
format	Specifies the text format for written to disk. Only used if x is a character.

---

write\_csv\_chunkwise    *Write chunks to a csv file*

---

## Description

Writes data to a csv file chunk by chunk. This function must be just in conjunction with [read\\_csv\\_chunkwise](#). Chunks of data will be read, processed and written when this function is called. For writing to a database use [insert\\_chunkwise\\_into](#).

## Usage

```
write_csv_chunkwise(  
  x,  
  file = "",  
  sep = ",",  
  dec = ".",  
  col.names = TRUE,  
  row.names = FALSE,  
  ...  
)
```

```
write_csv2_chunkwise(  
  x,  
  file = "",  
  sep = ";",  
  dec = ",",  
  col.names = TRUE,  
  row.names = FALSE,  
  ...  
)
```

```
write_table_chunkwise(  
  x,  
  file = "",  
  sep = "\t",  
  dec = ".",  
  col.names = TRUE,  
  row.names = TRUE,  
  ...  
)
```

## Arguments

x	chunkwise object pointing to a text file
file	file character or connection where the csv file should be written
sep	field separator

dec	decimal separator
col.names	should column names be written?
row.names	should row names be written?
...	passed through to <a href="#">read.table</a>

### Value

chunkwise object (chunkwise), when writing to a file it refers to the newly created file, otherwise to `x`.

### Examples

```
# create csv file for demo purpose
in_file <- file.path(tempdir(), "in.csv")
write.csv(women, in_file, row.names = FALSE, quote = FALSE)

#
women_chunked <-
  read_chunkwise(in_file) %>% #open chunkwise connection
  mutate(ratio = weight/height) %>%
  filter(ratio > 2) %>%
  select(height, ratio) %>%
  inner_join(data.frame(height=63:66)) # you can join with data.frames!

# no processing done until
out_file <- file.path(tempdir(), "processed.csv")
women_chunked %>%
  write_chunkwise(file=out_file)

head(women_chunked) # works (without processing all data...)

iris_file <- file.path(tempdir(), "iris.csv")
write.csv(iris, iris_file, row.names = FALSE, quote= FALSE)

iris_chunked <-
  read_chunkwise(iris_file, chunk_size = 49) %>% # 49 for demo purpose
  group_by(Species) %>%
  summarise(sepal_length = mean(Sepal.Length), n=n()) # note that mean is per chunk
```



# Index

chunked-package, [2](#)

dplyr-verbs (chunked-package), [2](#)

filter (chunked-package), [2](#)

insert\_chunkwise\_into, [3](#), [5](#), [7](#)

mutate (chunked-package), [2](#)

read.table, [8](#)

read\_chunkwise, [3](#)

read\_csv2\_chunkwise  
    (read\_csv\_chunkwise), [4](#)

read\_csv\_chunkwise, [4](#), [7](#)

read\_laf\_chunkwise  
    (read\_csv\_chunkwise), [4](#)

read\_table\_chunkwise  
    (read\_csv\_chunkwise), [4](#)

select (chunked-package), [2](#)

tbl, [3](#)

write\_chunkwise, [6](#)

write\_csv2\_chunkwise  
    (write\_csv\_chunkwise), [7](#)

write\_csv\_chunkwise, [4](#), [5](#), [7](#)

write\_table\_chunkwise  
    (write\_csv\_chunkwise), [7](#)