

# Package ‘prt’

May 16, 2021

**Title** Tabular Data Backed by Partitioned 'fst' Files

**Version** 0.1.4

**Description** Intended for larger-than-memory tabular data, 'prt' objects provide an interface to read row and/or column subsets into memory as data.table objects. Data queries, constructed as 'R' expressions, are evaluated using the non-standard evaluation framework provided by 'rlang' and file-backing is powered by the fast and efficient 'fst' package.

**URL** <https://nbenn.github.io/prt/>

**BugReports** <https://github.com/nbenn/prt/issues>

**License** GPL-3

**Imports** assertthat, fst, data.table, utils, vctrs, tibble, cli,  
pillar, crayon, fansi, backports, rlang,

**Suggests** testthat, xml2, covr, withr, nycflights13, datasets,  
rmarkdown, knitr, bench

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Nicolas Bennett [aut, cre],  
Drago Plecko [ctb]

**Maintainer** Nicolas Bennett <nicolas.bennett@stat.math.ethz.ch>

**Repository** CRAN

**Date/Publication** 2021-05-16 08:00:06 UTC

## R topics documented:

glimpse.prt . . . . .	2
new_prt . . . . .	3
nse . . . . .	5
print.prt . . . . .	7
subsetting . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

## Description

The tibble S3 generic function `pillar::glimpse()` is implemented for `prt` objects as well. Inspired by the output of `str()` when applied to `data.frames`, this function is intended to display the structure of the data in terms of columns, irrespective of how the data is organized in terms of R objects. Similarly to `trunc_dt()`, the function providing the bulk of functionality, `glimpse_dt()`, is exported such that implementing a class specific `pillar::glimpse()` function for other classes that representing tabular data is straightforward.

## Usage

```
## S3 method for class 'prt'  
glimpse(x, width = NULL, ...)  
  
glimpse_dt(x, width = NULL)  
  
str_sum(x)  
  
## S3 method for class 'prt'  
str(object, ...)  
  
str_dt(x, ...)
```

## Arguments

<code>x</code>	An object to glimpse at.
<code>width</code>	Width of output: defaults to the setting of the option <code>tibble.width</code> (if finite) or the width of the console.
<code>...</code>	Unused, for extensibility.
<code>object</code>	any R object about which you want to have some information.

## Details

Alongside a `prt`-specific `pillar::glimpse()` method, a `str()` method is provided as well for `prt` objects. However, breaking with base R expectations, it is not the structure of the object in terms of R objects that is shown, but in the same spirit as `pillar::glimpse()` it is the structure of the data that is printed. How this data is represents with respect to R objects is abstracted away as to show output as would be expected if the data were represented by a `data.frame`.

In similar spirit as `trunc_dt()` and `glimpse_dt()`, a `str_dt()` function is exported which provides the core functionality driving the `prt` implementation of `str()`. This function requires availability of a `head()` function for any object that is passed and output can be customized by implementing an optional `str_sum()` function.

**Examples**

```
cars <- as_prt(mtcars)

pillar::glimpse(cars)
pillar::glimpse(cars, width = 30)

str(cars)
str(cars, vec.len = 1)

str(unclass(cars))

str_sum(cars)
```

---

new\_prt

*Methods for creating and inspecting prt objects*

---

**Description**

The constructor `new_prt()` creates a `prt` object from one or several `fst` files, making sure that each table consist of identically named, ordered and typed columns. In order to create a `prt` object from an in-memory table, `as_prt()` coerces objects inheriting from `data.frame` to `prt` by first splitting rows into `n_chunks`, writing `fst` files to the directory `dir` and calling `new_prt()` on the resulting `fst` files. If this default splitting of rows (which might impact efficiency of subsequent queries on the data) is not optimal, a list of objects inheriting from `data.frame` is a valid `x` argument as well.

**Usage**

```
new_prt(files)

as_prt(x, n_chunks = NULL, dir = tempfile())

is_prt(x)

n_part(x)

part_nrow(x)

## S3 method for class 'prt'
head(x, n = 6L, ...)

## S3 method for class 'prt'
tail(x, n = 6L, ...)

## S3 method for class 'prt'
as.data.table(x, ...)
```

```
## S3 method for class 'prt'
as.list(x, ...)

## S3 method for class 'prt'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'prt'
as.matrix(x, ...)
```

## Arguments

<code>files</code>	Character vector of file name(s).
<code>x</code>	A prt object.
<code>n_chunks</code>	Count variable specifying the number of chunks <code>x</code> is split into.
<code>dir</code>	Directory where the chunked <code>fst::fst()</code> objects reside in.
<code>n</code>	Count variable indicating the number of rows to return.
<code>...</code>	Generic consistency: additional arguments are ignored and a warning is issued.
<code>row.names, optional</code>	Generic consistency: passing anything other than the default value issues a warning.

## Details

To check whether an object inherits from `prt`, the function `is_prt()` is exported, the number of partitions can be queried by calling `n_part()` and the number of rows per partition is available as `part_nrow()`.

The base R S3 generic functions `dim()`, `length()`, `dimnames()` and `names()`, have `prt`-specific implementations, where `dim()` returns the overall table dimensions, `length()` is synonymous for `ncol()`, `dimnames()` returns a length 2 list containing NULL column names as character vector and `names()` is synonymous for `colnames()`. Both setting and getting row names on `prt` objects is not supported and more generally, calling replacement functions such as `names<-()` or `dimnames<-()` leads to an error, as `prt` objects are immutable. The base R S3 generic functions `head()` and `tail()` are available as well and are used internally to provide an extensible mechanism for printing (see `trunc_dt()`).

Coercion to other base R objects is possible via `as.list()`, `as.data.frame()` and `as.matrix()` and for coercion to `data.table`, its generic function `data.table::as.data.table()` is available to `prt` objects. All coercion involves reading the full data into memory at once which might be problematic in cases of large data sets.

## Examples

```
cars <- as_prt(mtcars, n_chunks = 2L)

is_prt(cars)
n_part(cars)
part_nrow(cars)
```

```
nrow(cars)
ncol(cars)

colnames(cars)
names(cars)

head(cars)
tail(cars, n = 2)

str(as.list(cars))
str(as.data.frame(cars))
```

---

nse

*NSE subsetting*


---

### Description

A cornerstone feature of `prt` is the ability to load a (small) subset of rows (or columns) from a much larger tabular dataset. In order to specify such a subset, an implementation of the base R S3 generic function `subset()` is provided, driving the non-standard evaluation (NSE) of an expression within the context of the data (with similar semantics as the base R implementation for `data.frames`).

### Usage

```
## S3 method for class 'prt'
subset(x, subset, select, part_safe = FALSE, drop = FALSE, ...)

subset_quo(
  x,
  subset = NULL,
  select = NULL,
  part_safe = FALSE,
  env = parent.frame()
)
```

### Arguments

<code>x</code>	object to be subsetted.
<code>subset</code>	logical expression indicating elements or rows to keep: missing values are taken as false.
<code>select</code>	expression, indicating columns to select from a data frame.
<code>part_safe</code>	Logical flag indicating whether the subset expression can be safely be applied to individual partitions.
<code>drop</code>	passed on to <code>[]</code> indexing operator.
<code>...</code>	further arguments to be passed to or from other methods.
<code>env</code>	The environment in which <code>subset</code> and <code>select</code> are evaluated in. This environment is not applicable for quosures because they have their own environments.

## Details

The functions powering NSE are `rlang::enquo()` which quote the subset and select arguments and `rlang::eval_tidy()` which evaluates the expressions. This allows for some `rlang`-specific features to be used, such as the `.data/.env` pronouns, or the double-curly brace forwarding operator. For some example code, please refer to `vignette("prt", package = "prt")`.

While the function `subset()` quotes the arguments passed as `subset` and `select`, the function `subset_quo()` can be used to operate on already quoted expressions. A final noteworthy departure from the base R interface is the `part_safe` argument: this logical flag indicates whether it is safe to evaluate the expression on partitions individually or whether dependencies between partitions prevent this from yielding correct results. As it is not straightforward to determine if dependencies might exist from the expression alone, the default is `FALSE`, which in many cases will result in a less efficient resolution of the row-selection and it is up to the user to enable this optimization.

## Examples

```
dat <- as_prt(mtcars, n_chunks = 2L)

subset(dat, cyl == 6)
subset(dat, cyl == 6 & hp > 110)

colnames(subset(dat, select = mpg:hp))
colnames(subset(dat, select = -c(vs, am)))

sub_6 <- subset(dat, cyl == 6)

thresh <- 6
identical(subset(dat, cyl == thresh), sub_6)
identical(subset(dat, cyl == .env$thresh), sub_6)

cyl <- 6
identical(subset(dat, cyl == cyl), data.table::as.data.table(dat))
identical(subset(dat, cyl == !!cyl), sub_6)
identical(subset(dat, .data$cyl == .env$cyl), sub_6)

expr <- quote(cyl == 6)
# passing a quoted expression to subset() will yield an error
## Not run:
  subset(dat, expr)

## End(Not run)
identical(subset_quo(dat, expr), sub_6)

identical(
  subset(dat, qsec > mean(qsec), part_safe = TRUE),
  subset(dat, qsec > mean(qsec), part_safe = FALSE)
)
```

print.prt

*Printing prt*

## Description

Printing of `prt` objects combines the concise yet informative design of only showing as many columns as the terminal width allows for, introduced by `tibble`, with the `data.table` approach of showing both the first and last few rows of a table. Implementation wise, the interface is designed to mimic that of `tibble` printing as closely as possible, offering the same function arguments and using the same option settings (and default values) as introduced by `tibble`.

## Usage

```
## S3 method for class 'prt'
print(x, ..., n = NULL, width = NULL, n_extra = NULL)

## S3 method for class 'prt'
format(x, ..., n = NULL, width = NULL, n_extra = NULL)

trunc_dt(x, n = NULL, width = NULL, n_extra = NULL)
```

## Arguments

<code>x</code>	Object to format or print.
<code>...</code>	Other arguments passed on to individual methods.
<code>n</code>	Number of rows to show. If <code>NULL</code> , the default, will print all rows if less than option <code>tibble.print_max</code> . Otherwise, will print <code>tibble.print_min</code> rows.
<code>width</code>	Width of text output to generate. This defaults to <code>NULL</code> , which means use <code>getOption("tibble.width")</code> or (if also <code>NULL</code> ) <code>getOption("width")</code> ; the latter displays only the columns that fit on one screen. You can also set <code>options(tibble.width = Inf)</code> to override this default and always print all columns, this may be slow for very wide tibbles.
<code>n_extra</code>	Number of extra columns to print abbreviated information for, if the width is too small for the entire tibble. If <code>NULL</code> , the default, will print information about at most <code>tibble.max_extra_cols</code> extra columns.

## Details

While the function `tibble::trunc_mat()` does most of the heavy lifting for formatting `tibble` printing output, `prt` exports the function `trunc_dt()`, which drives analogous functionality while adding the top/bottom `n` row concept. This function can be used for creating `print()` methods for other classes which represent tabular data, given that this class implements `dim()`, `head()` and `tail()` (and optionally `tibble::tbl_sum()`) methods. For an example of this, see `vignette("prt", package = "prt")`.

The following session options are set by `tibble` and are respected by `prt`, as well as any other package that were to call `trunc_dt()`:

- `tibble.print_max`: Row number threshold: Maximum number of rows printed. Set to `Inf` to always print all rows. Default: 20.
- `tibble.print_min`: Number of rows printed if row number threshold is exceeded. Default: 10.
- `tibble.width`: Output width. Default: `NULL` (use width option).
- `tibble.max_extra_cols`: Number of extra columns printed in reduced form. Default: 100.

Both `tibble` and `prt` rely on `pillar` for formatting columns and therefore, the following options set by `pillar` are applicable to `prt` printing as well.

### Package options

- `pillar.bold`: Use bold font, e.g. for column headers? This currently defaults to `FALSE`, because many terminal fonts have poor support for bold fonts.
- `pillar.subtle`: Use subtle style, e.g. for row numbers and data types? Default: `TRUE`.
- `pillar.subtle_num`: Use subtle style for insignificant digits? Default: `FALSE`, is also affected by the `pillar.subtle` option.
- `pillar.neg`: Highlight negative numbers? Default: `TRUE`.
- `pillar.sigfig`: The number of significant digits that will be printed and highlighted, default: 3. Set the `pillar.subtle` option to `FALSE` to turn off highlighting of significant digits.
- `pillar.min_title_chars`: The minimum number of characters for the column title, default: 15. Column titles may be truncated up to that width to save horizontal space. Set to `Inf` to turn off truncation of column titles.
- `pillar.min_chars`: The minimum number of characters wide to display character columns, default: 0. Character columns may be truncated up to that width to save horizontal space. Set to `Inf` to turn off truncation of character columns.
- `pillar.max_dec_width`: The maximum allowed width for decimal notation, default 13.

### Examples

```
cars <- as_prt(mtcars)

print(cars)
print(cars, n = 2)
print(cars, width = 30)
print(cars, width = 30, n_extra = 2)
```



**Description**

Both single element subsetting via `[[` and `$`, as well as multi-element subsetting via `[` are available for `prt` objects. Subsetting semantics are modeled after those of the `tibble` class with the main difference being that there `tibble` returns `tibble` objects, `prt` returns `data.tables`. Differences to base R include that partial column name matching for `$` is not allowed and coercion to lower dimensions for `[` is always disabled by default. As `prt` objects are immutable, all subset-replace functions (`[[<-`, `$<-` and `[<-`) yield an error when passed a `prt` object.

**Usage**

```
## S3 method for class 'prt'
x[[i, j, ..., exact = TRUE]]

## S3 method for class 'prt'
x$name

## S3 method for class 'prt'
x[i, j, drop = FALSE]
```

**Arguments**

<code>x</code>	A <code>prt</code> object.
<code>i, j</code>	Row/column indexes. If <code>j</code> is omitted, <code>i</code> is used as column index.
<code>...</code>	Generic compatibility: any further arguments are ignored.
<code>exact</code>	Generic compatibility: only the default value of <code>TRUE</code> is supported.
<code>name</code>	A literal character string or a <a href="#">name</a> (possibly <a href="#">backtick</a> quoted).
<code>drop</code>	Coerce to a vector if fetching one column via <code>tbl[, j]</code> . Default <code>FALSE</code> , ignored when accessing a column via <code>tbl[j]</code> .

**Examples**

```
dat <- as_prt(mtcars)

identical(dat$mpg, dat[["mpg"]])

dat$mp
mtcars$mp

identical(dim(dat["mpg"]), dim(mtcars["mpg"]))
identical(dim(dat[, "mpg"]), dim(mtcars[, "mpg"]))
identical(dim(dat[1L, ]), dim(mtcars[1L, ]))
```

# Index

[.prt (subsetting), 8  
[[.prt (subsetting), 8  
\$.prt (subsetting), 8

as.data.frame(), 4  
as.data.frame.prt (new\_prt), 3  
as.data.table.prt (new\_prt), 3  
as.list(), 4  
as.list.prt (new\_prt), 3  
as.matrix(), 4  
as.matrix.prt (new\_prt), 3  
as\_prt (new\_prt), 3

backtick, 9

colnames(), 4

data.table::as.data.table(), 4  
dim(), 4, 7  
dimnames(), 4

format.prt (print.prt), 7  
fst::fst(), 4

glimpse.prt, 2  
glimpse\_dt (glimpse.prt), 2

head(), 2, 4, 7  
head.prt (new\_prt), 3

is\_prt (new\_prt), 3

length(), 4

n\_part (new\_prt), 3  
name, 9  
names(), 4  
ncol(), 4  
new\_prt, 3  
nse, 5

part\_nrow (new\_prt), 3

pillar::glimpse(), 2  
print(), 7  
print.prt, 7

str(), 2  
str.prt (glimpse.prt), 2  
str\_dt (glimpse.prt), 2  
str\_sum (glimpse.prt), 2  
subset.prt (nse), 5  
subset\_quo (nse), 5  
subsetting, 8

tail(), 4, 7  
tail.prt (new\_prt), 3  
tibble::tbl\_sum(), 7  
tibble::trunc\_mat(), 7  
trunc\_dt (print.prt), 7  
trunc\_dt(), 2, 4