Tidy Verbs for Fast Data Manipulation: : CHEAT SHEET

Basics

Combining the merits of syntax elegance from **dplyr** and computing performance from **data.table**, **tidyfst** intends to provide users with state-of-the-art data manipulation tools with least pain. The data.table syntax (**dt[i, j, by]**) could be applied in tidyfst, while the tidy data principal is implemented everywhere.

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Data I/O

CSV(considered to be general on various platforms):
fread("file.csv"**)** – read data from a flat file such as .csv or .tsv into R.
fwrite(dt, "file.csv"**)** – write data to a flat file from R.

FST(considered to be fast and memory efficient):
import_fst("file.fst"**)** – read data from a flat file of .fst into R.
export_fst(dt, "file.fst"**)** – write data to a .fst file from R.

Dealing with NAs

drop_na_dt – drop entries with NA in the column(s)
replace_na_dt – replace NA with other value
delete_na_cols – delete column(s) when NA volume or proportion is larger than a threshold
delete_na_rows– delete row(s) when NA volume or proportion is larger than a threshold
fill_na_dt – fill NA with previous or next observations

tidyft

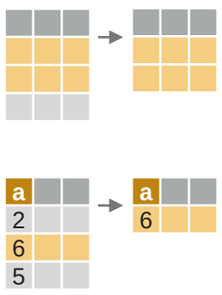
In data.table, there is an important feature: modification by reference. This could be really useful in high performance computation. This feature is implemented in tidyft, the mirror package of tidyfst.

Subset data

BY COLUMN

select_dt(dt, 2) – get column(s) as data.table
pull_dt(dt, 2) – get column as a vector
select_mix – select columns flexibly

BY ROW



Subset row based on:

- **slice_dt** – Position
- **slice_sample** - Randomly
- **slice_max_dt/slice_min_dt** – values
- **slice_top_dt/slice_tail_dt** - Position from top or bottom
- **distinct_dt** – unique values
- **filter_dt** - condition

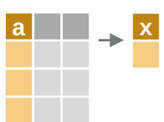
LOGICAL OPERATORS TO USE IN filter_dt

<	<=	is.na()	%in%	|	**%like%**
>	>=	!is.na()	!	&	**%between%**

Update data

mutate_dt – add or mutate column(s)
mutate_vars – update multiple columns
mutate_when – update entries that meet certain condition

Aggregation

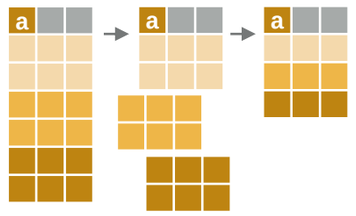


summarise_dt – aggregate one column to one value
count_dt/add_count_dt – get unique counts
summarise_vars – aggregate multiple columns
summarise_when – conditional aggregation

Reorder data

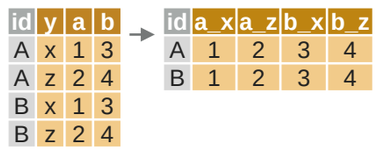
arrange_dt – reorder data by row according to column value
relocate_dt – reorder data by column(s)

Group computation

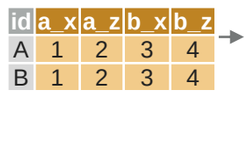


While tidyfst provides **group_dt/group_by_dt/group_by_exe** to implement computation in groups. If you are handling big data, it is recommended to use the ***by*** parameter in the function when available, e.g. *summarise_dt(iris,avg = mean(Sepal.Length),by = Species)*

Reshape data



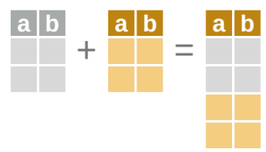
wider_dt – Transform data from long to wide



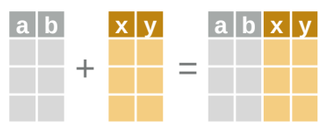
longer_dt – Transform data from wide to long

Merge data

BIND

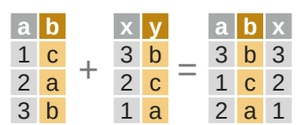


rbind



cbind

JOIN



left_join_dt / right_join_dt
inner_join_dt / full_join_dt
anti_join_dt / semi_join_dt

SET OPERATIONS

intersect_dt
union_dt
setdiff_dt
setequal_dt