R data.table : : **CHEAT SHEET**

Basics

`data.table` provides a high-performance version of base R's **`data.frame`** with syntax and feature enhancements for ease of use, convenience and programming speed

General Form

```
DT[i,    j,    by]
```

- `i`: On which rows
- `j`: What to do?
- `by`: Grouped by what?

CREATE `data.table`

```
library('data.table')
DT <- data.table(V1=c(1:10),
                 V2= rep(c("red", "blue"),5))
```

CONVERT TO/FROM `data.table`

```
irisDT <- as.data.table(iris)
iris   <- as.data.frame(irisDT)
```

SET COLUMN NAMES

V1	V2

↓

weight	color

```
colnames(DT) <- c("weight",
"color")

setnames(DT,
         c("V1","V2"),
         c("weight","color"))
```

IMPORT AND EXPORT

Importing File

`fread("file.csv")`- fast and parallelized delimited file reader

Exporting File

`fwrite(file, "file.csv")`- fast and parallelized file writer

Using `i` - Filtering

Filtering functions return a data.table with a subset of records. Filtering answers the question: "on which rows"?

```
DT[i,    j,    by]
```

- `i`: On which rows

Example data.table

V1	V2
1	red
2	blue
3	red
4	blue
5	red

Slicing Rows

```
DT[1:2, ]
DT[1:2]
```

→

V1	V2
1	red
2	blue

Filtering with Condition

```
DT[V1 > 3]
```

→

V1	V2
4	blue
5	red

```
DT[V2 %in% c("blue")]
```

Using `j` – Manipulating

Manipulating functions performs a task on columns in the **`data.table`**. Columns can be selected, aggregated, etc. Manipulating answers the question: "what to do"?

```
DT[i,    j,    by]
```

- `j`: What to do?

Select Columns

```
DT[, V2]
DT[, .(V2)]
```

→

V2
red
blue

Summarizing

```
DT[, .(sd(V1),mean(V1))]
```

→

V1	V2
3.03	5.5

```
DT[, .(SDev = sd(V1),
       Mean = mean(V1))]
```

→

SDev	Mean
3.03	5.5

Adding/Updating Columns

Columns are updated by what is after the :=

```
DT[,c("V1","V2"):=
     list(round(log(V1)),
     substring(V2,0,1))]
```

→

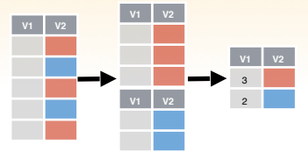
V1	V2
0	r
1	b
1	r
1	b
2	r

Using by - Grouping

Grouping functions performs a manipulation, j, on each group. Grouping answers the question: "grouped by what"?

```
DT[i,    j,    by]
```

- `by`: Grouped by what?



```
DT[, .N, by=V1]
```

```
DT[1:3,.(SDev=sd(V1),Mean=mean(V1)),by=V2]
```

Calculate the standard deviation and mean of V1 for every group in V2 after filtering to the first 3 rows.

SUBSET OF DATA

```
DT[, lapply(.SD, mean), by = V3,
     .SDcols  = c("V1", "V2")]
```

Calculate mean of all columns specified in .SDcols in the dataset for every group in V3

LAGS AND LEADS

```
lag_cols <- c("lag_V1", "lag_V2")
DT[order(year),
   lag_cols := shift(.SD,1,type="lag"),
                     V3,
                     .SDcols=c("V1", "V2")]
```

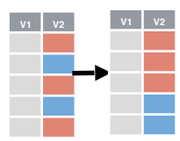
Order by year

Return last lag of .SDcols for every group in V3

Using Keys

Keys = supercharged rownames
Incredibly fast filtering

```
setkey(DT, V2)
```



`setkey(data.table, column)`
1. Orders the data.table by column
2. Marks the column as the key

The key column has been set to V2
To filter on the key column, simply provide the value(s)

```
DT[.("blue")]
```

→

V1	V2
2	blue
4	blue

Python datatable : : **CHEAT SHEET**

datatable

Basics

datatable provides a high-performance version of Python's pandas or SFrame. The package is closely related to R's data.table and attempts to mimic its core algorithms and API

General Form

```
DT[i,    j,    by]
```

On which rows | What to do? | Grouped by what?

CREATE datatable

```
import datatable as dt
from datatable import f, mean, sd
DT = dt.Frame({"V1": range(1,11),
"V2": ["red","blue"]*5})
```

CONVERT TO/FROM datatable

```
Frame = pandas.DataFrame({"A": [2, 5, 8],
                          "B": ["e", "r", "qq"]})
frame_dt = dt.Frame(frame) #convert to datatable
```

```
frame_pd = frame_DT.topandas() #convert to pandas
frame_py = frame_DT.topython() #convert to python
frame_np = frame_DT.tonumpy()  #convert to numpy
```

SET COLUMN NAMES

V1	V2

↓

weight	color

```
DT.names = ["weight", "color"]

DT.rename({"V1":"weight",
           "V2":"color"})
```

IMPORT AND EXPORT

Importing and Exporting Files

DT = dt.fread("file.csv"**)**- fast and parallelized delimited file reader
DT.to_csv("file.csv"**)**- fast and parallelized CSV writer

DT = dt.open("file.nff"**)**- instantaneous opening of binary files saved from datatable
DT.save("file.nff"**)**- fast binary file writer

Using i - Filtering

Filtering functions return a Frame with a subset of records. Filtering answers the question: "on which rows"?

```
DT[i,    j,    by]
```

On which rows

Example Frame

V1	V2
1	red
2	blue
3	red
4	blue
5	red

Slicing Rows

```
DT[0:2,:]
DT[:2, :]
```

→

V1	V2
1	red
2	blue

Filtering with Condition

```
DT[f.V1 > 3, :]
```

→

V1	V2
4	blue
5	red

Filtering by string not yet implemented

Using j – Manipulating

Manipulating functions performs a task on columns in the **Frame**. Columns can be selected, aggregated, etc. Manipulating answers the question: "what to do"?

```
DT[i,    j,    by]
```

What to do?

Select Columns

```
DT[:, ["V2"]]
DT(select = ["V2"])
```

→

V2
red
blue

Summarizing

```
DT[:,[sd(f.V1), mean(f.V1)]]
```

→

V1	V2
3.03	5.5

```
DT[:,{"SDev": sd(f.V1),
      "Mean": mean(f.V1)}]
```

→

SDev	Mean
3.03	5.5

Adding/Updating Columns

Columns are updated by what is after the =

```
DT[["V1", "V2"]] =
    DT[[(f.V1+1), (f.V1/2)]]
```

→

V1	V2
2	0.5
3	1
4	1.5
5	2
6	2.5

Using by - Grouping

Grouping functions performs a manipulation, j, on each group. Grouping answers the question: "grouped by what"?

```
DT[i,    j,    by]
```

Grouped by what?

```
DT[0:3,
   {'SDev':sd(f.V1), 'Mean':mean(f.V1)},
   "V2"]
```

Calculate the standard deviation and mean of V1 for every group in V2 after filtering to the first 3 rows.

SUBSET OF DATA

```
DT(select=[mean(f[n]) for n in ["V1", "V2"]],
groupby="V3")
```

Calculate mean of the columns specified in the list ["V1", "V2"] in the dataset for every group in V3

Current Limitations

Currently **datatable** is in the Alpha stage and undergoing active development. Some of the features are incomplete or missing.

See datatable.h2o.ai for viewing the current progress

FUTURE FEATURES

- ❖ Joins
- ❖ Lag and Leads
- ❖ Use of Keys
- ❖ Categorical and Datetime Types
- ❖ Additional Aggregation Capabilities

H2O.ai