# Creating a data.table

**data.table():** creates a data table from vectors of the same length.

**as.data.table(**DF**,** keep.rownames = “V1”**):** convert a data frame or a list of vectors to a data.table.

**fread():** read csv files very fast.

*fread(str, skip = 2 , nrows = 1)*

*fread(str, skip = ”a,b” , nrows = 1)*

*fread(str, select = c("a" , "c")) or*

*fread(str, select = c(1 , 3))*

*fread(str, drop = "b") or*

*fread(str, drop = 2)*

*fread(str, colClasses = c(x5 = "factor"))*

*fread(str,*

*colClasses = list(numeric = 1:4,*

*factor = c("x5" , "x6")))*

*fread(str, fill = TRUE)*

*fread(str, na.strings = c("###" , "#N/A"))*

# Transformation Structure

**cols** <- names(DT)

**DT[ i , j , by ]**

**.N** <- nrow(DT)

**.SD** <- DT

**i: row**

**DT[**1:5**]:** first 5 rows.

**DT**[-(1:5)**]** or **DT**[!(1:5)**]:** Not first 5 row.

**DT[**.N**]:** Returns the last row.

**DT[**V1 == “A” & V2 == “B”**]:** V1 is A and V2 is B.

**DT[**V1 %chin% c(“A”,”B”)**]:** V1 is A or B.

**DT[**V1 %like% “^A”**]:** V1 that starts with A.

**DT[**V1 %between% c(1,5)**]:** V1 between 1 and 5.

**DT[**order(V1)**]:** Ascender order of V1.

**DT[**order(-V1)**]:** Descender order of V1.

**j: columns**

**DT[,** ”V1”**]:** Select V1 in a data.table.

**DT[,** V1**]:** Select V1 as vector.

**DT[,** c("V1", "V2")**]:** Select V1 and V2.

**DT[,** !c("V1", "V2")**]:** Not Select V1 and V2.

**DT[,** .(V3 = V1, V2 = as.character(V2)**]:** Select V1 and V2 by renaming V1 to V3 and transforming V2 to character.

**DT[,**.(V1 = mean(V1))**]:** Summarise the mean of V1.

**DT[,** `:=`(V3 = V1 + V2)**]:** Mutate V3.

**DT[,** c("V1", "V2") := lapply(.SD, sqrt),

.SDcols = c("V1", "V2")**] :** Mutate across V1 and V2.

**by: groups**

**DT[,** .N **,** “V1”**]** or **DT[,** .N **,** .(V2 =V1)**]:** Number of rows for any level of V1.

**DT[,** uniqueN(V1)**,** V2**]:** Number of unique values for any level of V2.

# Chaining expressions

**DT[**...**][**...**][**...**]**

# Joining tables

**DT1[ i , on , j , by]**

i: DT2

on: common groups

j: columns of DT1

by: groups of DT1

**BUT**

**.EACHI** groups **j** by

each **on group** from DT2

keys <- c(“V1” = ”V2”)

setkey(DT, V1, V2) or setkey(DT, keys)

Text, chat or text message

Description automatically generated

## Right join

**DT1[**DT2**,** on = .(V1 = V2) ,

allow.cartesian = FALSE,

mult = c("first", “last”)**]**

merge(x = DT1, y = DT2,

by.x = "V1", by.y= "V2",

all.y = TRUE)

## Inner join

**DT1[**DT2**,** on = .(V1= V1x) **,**

nomatch = 0**]**

merge(x = DT1, y = DT2,

by.x = "V1", by.y = "V1x")

## Anti-joins

**DT1[** !DT2 **,** on = .(V1)**]:** We just see rows of DT1 aren’t in DT2.

## Semi-join

**DT1[**DT2[,”V1”]**,** on = "V1"**,**

nomatch = 0**][,** unique(.SD)**]**

## Full join

merge(x = DT1, y = DT2,

by = "V1", all = TRUE)

## Left join

merge(x = DT1, y = DT2,

by = "V1", all.x = TRUE)

# Functions

**uniqueN(**V1**):** number of unique values in a vector or DT.

**duplicated(**, by = “V1”, fromLast = TRUE**) :** what rows are duplicates?

**unique(**, by = “V1”, fromLast = TRUE**)**

**which.max(**vector**)**

**get(**string**)** : evaluate a string as a column reference.

**(**string**):** used to set a new column name.

**tables():** shows you all DTs loaded in your R session.

**str():** shows you the type of each column in a single DT

**haskey(**DT**):** checks whether you have set keys.

**key(**DT**)**: returns the key columns you have set

**fwrite(**DT, "fwrite.csv")

**grep(**pattern, vector, value = FALSE**)** returns indexes of strings matching a pattern.

**shift(**sales, type = "lag", n = 15**)**

**separate\_rows**

**DT[**, .(`V1`= unlist(tstrsplit(`V1", ")) ),

by = setdiff( names(DT) , "V1" )**]**

**tidyr::complete**

**DT[**CJ(V1 = V1, V2 = V2, unique=TRUE),

on=.(V1, V2)**]**

**Fill**

**DT[**,V1 := V1[nafill(replace(seq\_along(V1),

is.na(V1),

NA\_integer\_),

"locf")] **]**

nafill(value, type = "locf")

**nest\_data**

**DT[**, .(data = list(.SD)),films**]**

**unnest\_data**

**DT[**, .(films = films[[1]]), name**]**

**swn[**, counts := map(data, ~count(.x, spices))**]**

stargazer::stargazer(list(mod1, mod2), type = "text")

Reduce( f = function(x, y){merge(x, y, by = "obstime x = list(someDT, otherDT) )

Graphical user interface

Description automatically generated with low confidence

copy(`\_DT1`)[, `:=`(c("A", "B"), tstrsplit(x, split = "\\."))]

copy(`\_DT2`)[, `:=`(x = nafill(x, "locf"))]

copy(`\_DT2`)[, `:=`(x = fcoalesce(x, 99))]

`\_DT7`[, .(data = .(.SD)), by = .(x)]

**setnames**(copy(`\_DT1`), toupper)

**rbind**("2015" = sales\_2015, "2016" = sales\_2016, idcol = "year", fill = TRUE, use.names = TRUE)

**rbindlist(**list(DT1, DT2, DT3), fill = TRUE**)**

**fintersect()** : what rows do these two data.tables share in common?

**funion()** : what is the unique set of rows across these two data.tables ?

**fsetdiff**(all = TRUE) : what rows are unique to this data.table ?

MyVarb <- data.table(Y=rnorm(100),

V1=rnorm(100),

V2=rnorm(100))

eq=c("Y ~ V1", "Y ~ V2", "Y ~ V1 + V2")

DT<-rbindlist(lapply(eq, function(mod) {

reg<-lm(mod, data=MyVarb)

dt<-data.table(summary(reg)$coefficients)

dt[,coef:=row.names(summary(reg)$coefficients)]

dt[,aic:=AIC(reg)]

dt[,model:=mod]

}))