data.table

Maja Kuzman

University of Zagreb

2019-10-25

Creating a data.frame

example: pi exercise

```
set.seed(1234)
ddf <- data.frame(x = runif(n = 1000000,-1,1), y=runif(n = 1000000,-1
ddf$insideCircle <- ((ddf$x^2 + ddf$y^2 )<=1)
ddf$group <- cut(1:1000000,10)
ddf$group50 <- cut(1:1000000,50)
head(ddf)
```

```
##            x          y insideCircle        group      group50
## 1 -0.7725932 -0.6618851        FALSE (-999,1e+05] (-999,2e+04]
## 2  0.2445988 -0.2041903         TRUE (-999,1e+05] (-999,2e+04]
## 3  0.2185495  0.6059561         TRUE (-999,1e+05] (-999,2e+04]
## 4  0.2467589 -0.3694550         TRUE (-999,1e+05] (-999,2e+04]
## 5  0.7218308 -0.5909273         TRUE (-999,1e+05] (-999,2e+04]
## 6  0.2806212  0.1858141         TRUE (-999,1e+05] (-999,2e+04]
```

I will show you some examples on this data, and you will do the exercise on data available in R (iris).

Loading the library

If you don't have the package, install it with:
(only once)

```
install.packages(data.table)
```

Once it is installed, you need to load it to R:
(once per session)

```
library(data.table)
```

If you already have a data frame, convert it to data.table by:

```
ddf_dt <- as.data.table(ddf)
```

Packages in R: data.table

All you need to know:

DT[i,j,by]

i: select those rows

j: do this to them

by: do it per groups

data.table: SELECTING ROWS

Selecting rows: What is the number of rows with y cordinate smaller than 0.05 and larger than 0 in ddf:

In data.table:
You select rows similar as you would select elements in vector:

```
nrow(ddf_dt[y<0.05 & y>0])
```

```
## [1] 24649
```

```
ddf_dt[y<0.05 & y>0][1:2]
```

```
##              x          y insideCircle        group      group50
## 1: -0.53554818 0.03680954         TRUE (-999,1e+05] (-999,2e+04]
## 2: -0.01207815 0.01357045         TRUE (-999,1e+05] (-999,2e+04]
```

data.table: SELECTING COLUMNS

Select columns x and y and first 5 rows:

In data.table use . instead of c,
. is short for list()

```
ddf_dt[1:5,.(x,y)]
```

```
##             x          y
## 1: -0.7725932 -0.6618851
## 2:  0.2445988 -0.2041903
## 3:  0.2185495  0.6059561
## 4:  0.2467589 -0.3694550
## 5:  0.7218308 -0.5909273
```

If you want to use column names:

```
cnames <- c("x","y")
ddf_dt[1:5,..cnames]
```

```
##             x          y
## 1: -0.7725932 -0.6618851
## 2:  0.2445988 -0.2041903
```

data.table: SELECTING COLUMNS

note on columns selections: this will also work:

```
ddf_dt[1:3,x:insideCircle]
```

```
##              x          y insideCircle
## 1: -0.7725932 -0.6618851        FALSE
## 2:  0.2445988 -0.2041903         TRUE
## 3:  0.2185495  0.6059561         TRUE
```

```
ddf_dt[1:3,-(x:insideCircle)]
```

```
##           group      group50
## 1: (-999,1e+05] (-999,2e+04]
## 2: (-999,1e+05] (-999,2e+04]
## 3: (-999,1e+05] (-999,2e+04]
```

```
ddf_dt[1:3,!(x:insideCircle)]
```

```
##           group      group50
## 1: (-999,1e+05] (-999,2e+04]
## 2: (-999,1e+05] (-999,2e+04]
```

data.table: exercise iris 1

```
ddf_dt <- as.data.table(ddt)
ddf_dt[1:5,.(x,y)]
cnames <- c("x","y")
ddf_dt[1:5,..cnames]
```

Mini exercise:

- convert iris to data.table - call it iris_dt
- Select all rows in iris_dt with Sepal.Length<6.7
- Select as before, but show only columns Sepal.Length and Species
- Do the previous by using column names!

data.table: exercise iris 1 solved

Mini exercise:

- convert iris to data.table - call it iris_dt
- Select all rows in iris_dt with Sepal.Length<6.7
- Select as before, but show only columns Sepal.Length and Species

```
iris_dt <- as.data.table(iris)
iris_dt[Sepal.Length<6.7]
iris_dt[Sepal.Length<6.7, .(Sepal.Length, Species)]
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
##   1:          5.1         3.5          1.4         0.2    setosa
##   2:          4.9         3.0          1.4         0.2    setosa
##   3:          4.7         3.2          1.3         0.2    setosa
##   4:          4.6         3.1          1.5         0.2    setosa
##   5:          5.0         3.6          1.4         0.2    setosa
##  ---
## 118:          5.8         2.7          5.1         1.9 virginica
## 119:          6.3         2.5          5.0         1.9 virginica
## 120:          6.5         3.0          5.2         2.0 virginica
## 121:          6.2         3.4          5.4         2.3 virginica
## 122:          5.9         3.0          5.1         1.8 virginica
```

data.table: OPERATION ON COLUMNS

Calculations of mean:

```
ddf_dt[,mean(insideCircle)]
```

```
## [1] 0.784404
```

By groups:

```
ddf_dt[,mean(insideCircle), by=group]
```

```
##              group      V1
##  1:  (-999,1e+05] 0.78506
##  2: (1e+05,2e+05] 0.78564
##  3: (2e+05,3e+05] 0.78424
##  4: (3e+05,4e+05] 0.78491
##  5: (4e+05,5e+05] 0.78362
##  6: (5e+05,6e+05] 0.78260
##  7: (6e+05,7e+05] 0.78588
##  8: (7e+05,8e+05] 0.78490
##  9: (8e+05,9e+05] 0.78323
## 10: (9e+05,1e+06] 0.78396
```

data.table: OPERATION ON COLUMNS

Calculations of mean, sd, max x , min x, number of observations per group, get a data frame order it by mean:

```
meanic <- by(ddf$insideCircle,INDICES = ddf$group, mean)
sdic <- by(ddf$insideCircle,INDICES = ddf$group, sd)
maxx <- by(ddf$x,INDICES = ddf$group, max)
minx <- by(ddf$x,INDICES = ddf$group, min)
nr_grp <- by(ddf$insideCircle,INDICES = ddf$group, length)
res_df <- data.frame(as.numeric(meanic),
                     as.numeric(sdic),
                     as.numeric(maxx),
                     as.numeric(minx),
                     as.numeric(nr_grp))
res_df[order(res_df$as.numeric.meanic.),]
```

```
##    as.numeric.meanic. as.numeric.sdic. as.numeric.maxx. as.numeric.minx.
## 6             0.78260        0.4124790        0.9999973       -0.9999503
## 9             0.78323        0.4120467        0.9999842       -0.9999849
## 5             0.78362        0.4117783        0.9999713       -0.9999139
## 10            0.78396        0.4115439        0.9999869       -0.9999748
## 3             0.78424        0.4113506        0.9999322       -0.9999967
## 8             0.78490        0.4108938        0.9999980       -0.9999751
```

data.table: OPERATION ON COLUMNS

Calculations of mean, sd, max x , min x, number of observations per group, get a data frame order it by mean:

```
ddf_dt[ ,
        .(mean=mean(insideCircle),
          sd  =sd(insideCircle),
          min_x= min(x),
          max_x= max(x),
          N= .N),
        by=group][order(mean)]
```

```
##              group    mean        sd      min_x     max_x      N
##  1: (5e+05,6e+05] 0.78260 0.4124790 -0.9999503 0.9999973 100000
##  2: (8e+05,9e+05] 0.78323 0.4120467 -0.9999849 0.9999842 100000
##  3: (4e+05,5e+05] 0.78362 0.4117783 -0.9999139 0.9999713 100000
##  4: (9e+05,1e+06] 0.78396 0.4115439 -0.9999748 0.9999869 100000
##  5: (2e+05,3e+05] 0.78424 0.4113506 -0.9999967 0.9999322 100000
##  6: (7e+05,8e+05] 0.78490 0.4108938 -0.9999751 0.9999980 100000
##  7: (3e+05,4e+05] 0.78491 0.4108868 -0.9999897 0.9999848 100000
##  8:  (-999,1e+05] 0.78506 0.4107828 -0.9999711 0.9999957 100000
##  9: (1e+05,2e+05] 0.78564 0.4103797 -0.9999935 0.9999954 100000
## 10: (6e+05,7e+05] 0.78588 0.4102125 -0.9999978 0.9999886 100000
```

data.table: .N, by

The .N gives number of observations:

```
ddf_dt[,.N]
```

```
## [1] 1000000
```

You can group by multiple variables:

```
ddf_dt[,.N, by=.(group,x>0.5)]
```

```
##              group     x     N
##  1:  (-999,1e+05] FALSE 75047
##  2:  (-999,1e+05]  TRUE 24953
##  3: (1e+05,2e+05] FALSE 75064
##  4: (1e+05,2e+05]  TRUE 24936
##  5: (2e+05,3e+05] FALSE 74994
##  6: (2e+05,3e+05]  TRUE 25006
##  7: (3e+05,4e+05]  TRUE 24913
##  8: (3e+05,4e+05] FALSE 75087
##  9: (4e+05,5e+05] FALSE 74739
## 10: (4e+05,5e+05]  TRUE 25261
## 11: (5e+05,6e+05] FALSE 75097
```

data.table: exercise iris 2

```
ddf_dt[,
       .(N=.N),
       by=.(group,x>0.5)]
```

Exercise on iris_dt:

- Select all rows where Sepal.Length < 6.7 and Species=="virginica"
- For those - use chaining - [][] to calculate mean Petal.Width for all flowers
- Repeat mean Petal.Width but grouped by condition Sepal.Length < 6.7
- How many of those flowers have Sepal.Width>3 and how many less then 3?

data.table: add a new column

Use ':=' to add a new column

```
ddf_dt[,N:=.N]
ddf_dt[1:3]
```

```
##             x          y insideCircle        group      group50       N
## 1: -0.7725932 -0.6618851        FALSE (-999,1e+05] (-999,2e+04] 1000000
## 2:  0.2445988 -0.2041903         TRUE (-999,1e+05] (-999,2e+04] 1000000
## 3:  0.2185495  0.6059561         TRUE (-999,1e+05] (-999,2e+04] 1000000
```

If you want to add multiple columns, use ':='as a function:

```
ddf_dt[ ,
       ':='(N_grp=.N, mean=mean(insideCircle)),
       by=.(group)]
ddf_dt[1:3]
```

```
##             x          y insideCircle        group      group50       N
## 1: -0.7725932 -0.6618851        FALSE (-999,1e+05] (-999,2e+04] 1000000
## 2:  0.2445988 -0.2041903         TRUE (-999,1e+05] (-999,2e+04] 1000000
## 3:  0.2185495  0.6059561         TRUE (-999,1e+05] (-999,2e+04] 1000000
##     N_grp    mean
```

data.table: Exercise add a new column

```
ddf_dt[ ,
       ':='(N_grp=.N, mean=mean(insideCircle)),
       by=.(group)]
```

- Add columns to iris_dt that represent mean and sd of Petal.Width grouped by species.
- use function uniqueN to check how many unique mean petal withs s there are.

```
iris_dt[,
        ':='(meanPW = mean(Petal.Width),
             sdPW = sd(Petal.Width))
        ,Species]
iris_dt[,uniqueN(meanPW)]
```

```
## [1] 3
```

data.table: .I, .GRP

The .I holds row numbers:

```
ddf_dt[,.(row_id=.I,x,y)]
```

```
##           row_id           x           y
##       1:      1 -0.77259318 -0.66188507
##       2:      2  0.24459881 -0.20419032
##       3:      3  0.21854947  0.60595613
##       4:      4  0.24675888 -0.36945498
##       5:      5  0.72183077 -0.59092730
##      ---
##  999996:  999996 -0.35239545 -0.46499966
##  999997:  999997  0.97471153 -0.59863134
##  999998:  999998 -0.35498930 -0.61072369
##  999999:  999999  0.53758948 -0.03446457
## 1000000: 1000000  0.07020645  0.24132790
```

The .GRP holds unique group number:

```
ddf_dt[,.GRP,.(group, insideCircle)][1:10]
```

```
##            group insideCircle GRP
```

data.table Exercise :=, .N, .I, .GRP

```
ddf_dt[ ,
       ':='(N_grp=.N, mean=mean(insideCircle)),
       by=.(group,group50)]
ddf_dt[,.(row_id=.I,x,y)]
ddf_dt[,.GRP,.(group, insideCircle)]
```

- Add columns to iris_dt that represent number of observations, row number and group ID of all rows for which Petal.Length is smaller than 6.5 in iris_dt grouped by species.
- One great benefit of data.table is the ability to sub-assign by reference: Try it: select all rows that have species=="virginica" and rename those Species entries using := to new_virginica

Solution

```
iris_dt[,
        ':='(N=.N, GRP=.GRP, rN=.I),
        by=Species]
iris_dt[Species=="virginica", Species:="new_virginica"]
iris_dt
```

```
##       Sepal.Length Sepal.Width Petal.Length Petal.Width        Species
##    1:          5.1         3.5          1.4         0.2         setosa
##    2:          4.9         3.0          1.4         0.2         setosa
##    3:          4.7         3.2          1.3         0.2         setosa
##    4:          4.6         3.1          1.5         0.2         setosa
##    5:          5.0         3.6          1.4         0.2         setosa
##   ---
##  146:          6.7         3.0          5.2         2.3 new_virginica
##  147:          6.3         2.5          5.0         1.9 new_virginica
##  148:          6.5         3.0          5.2         2.0 new_virginica
##  149:          6.2         3.4          5.4         2.3 new_virginica
##  150:          5.9         3.0          5.1         1.8 new_virginica
##       meanPW      sdPW  N GRP  rN
##    1:  0.246 0.1053856 50   1   1
##    2:  0.246 0.1053856 50   1   2
##    3:  0.246 0.1053856 50   1   3
```

data.table MORE ADVANCED USAGE: keys

You can "set a key" of data.table by using the setkey() function. This will

```
setkey(ddf_dt, insideCircle,x,y)
ddf_dt
```

```
##                    x            y insideCircle         group
##       1: -0.9999978 -0.133110980        FALSE (6e+05,7e+05]
##       2: -0.9999967 -0.805848121        FALSE (2e+05,3e+05]
##       3: -0.9999935 -0.640493895        FALSE (1e+05,2e+05]
##       4: -0.9999897 -0.659440982        FALSE (3e+05,4e+05]
##       5: -0.9999849 -0.019881909        FALSE (8e+05,9e+05]
##      ---
##  999996:  0.9996620 -0.002455097         TRUE (6e+05,7e+05]
##  999997:  0.9997166 -0.011472923         TRUE (2e+05,3e+05]
##  999998:  0.9997216 -0.023574530         TRUE (5e+05,6e+05]
##  999999:  0.9997531  0.002546730         TRUE  (-999,1e+05]
## 1000000:  0.9997804  0.010835514         TRUE (3e+05,4e+05]
##                    group50       N  N_grp    mean
##       1: (6.4e+05,6.6e+05] 1000000 100000 0.78588
##       2:   (2e+05,2.2e+05] 1000000 100000 0.78424
##       3:   (1.8e+05,2e+05] 1000000 100000 0.78564
```

data.table MORE ADVANCED USAGE: .SD, .SDcols

Select all columns with .SD. Select only a subset of all columns by .SDcols:

```
ddf_dt[,.SD, .SDcols=1:2]
```

```
##                    x             y
##       1: -0.9999978 -0.133110980
##       2: -0.9999967 -0.805848121
##       3: -0.9999935 -0.640493895
##       4: -0.9999897 -0.659440982
##       5: -0.9999849 -0.019881909
##      ---
##  999996:  0.9996620 -0.002455097
##  999997:  0.9997166 -0.011472923
##  999998:  0.9997216 -0.023574530
##  999999:  0.9997531  0.002546730
## 1000000:  0.9997804  0.010835514
```

this is especially useful when you want to do the same operation on multiple columns: for example, calculate mean of x and y:

data.table MORE ADVANCED USAGE: .SD, .SDcols

..Or for example select first and last row for each group:

```
ddf_dt[, .SD[c(1, .N)], by=insideCircle]
```

```
##    insideCircle          x           y         group           group50
## 1:        FALSE -0.9999978 -0.13311098 (6e+05,7e+05] (6.4e+05,6.6e+05]
## 2:        FALSE  0.9999980  0.39514274 (7e+05,8e+05] (7.6e+05,7.8e+05]
## 3:         TRUE -0.9997722 -0.02037310 (8e+05,9e+05] (8.4e+05,8.6e+05]
## 4:         TRUE  0.9997804  0.01083551 (3e+05,4e+05]   (3e+05,3.2e+05]
##          N  N_grp    mean
## 1: 1000000 100000 0.78588
## 2: 1000000 100000 0.78490
## 3: 1000000 100000 0.78323
## 4: 1000000 100000 0.78491
```

It is easier if you read it as: SelectedData

data.table exercise MORE ADVANCED USAGE

```
ddf_dt[,lapply(.SD,mean), by=insideCircle,.SDcols=1:2]
ddf_dt[, .SD[c(1, .N)], by=insideCircle]
```

Do the following in a single command:

- order the results by Petal.Width and select first three (smallest) observations .
- Calculate mean of first four columns for iris_dt for those observations

data.table exercise MORE ADVANCED USAGE solved

Do the following in a single command:

- order the results by Petal.Width and select first three (smallest) observations .
- Calculate mean of first four columns for iris_dt for those observations

```
iris_dt[order(Petal.Width),
        .SD[1:3],
        by=Species]
```

```
##          Species Sepal.Length Sepal.Width Petal.Length Petal.Width meanPW
## 1:        setosa          4.9         3.1          1.5         0.1  0.246
## 2:        setosa          4.8         3.0          1.4         0.1  0.246
## 3:        setosa          4.3         3.0          1.1         0.1  0.246
## 4:    versicolor          4.9         2.4          3.3         1.0  1.326
## 5:    versicolor          5.0         2.0          3.5         1.0  1.326
## 6:    versicolor          6.0         2.2          4.0         1.0  1.326
## 7: new_virginica          6.1         2.6          5.6         1.4  2.026
## 8: new_virginica          6.0         2.2          5.0         1.5  2.026
## 9: new_virginica          6.3         2.8          5.1         1.5  2.026
```

data.table exercise MORE ADVANCED USAGE solved

Do the following in a single command:

- order the results by Petal.Width and select first three (smallest) observations .
- Calculate mean of first four columns for iris_dt for those observations

```
iris_dt[order(Petal.Width),
        .SD[1:3],
        by=Species][ ,
                    lapply(.SD,mean),
                    .SDcols=Sepal.Length:Petal.Width]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1:     5.366667    2.588889     3.388889   0.8555556
```

data.table MORE ADVANCED USAGE {}

Suppressing Intermediate Output with {}:

If you want to do multiple things, but dont need to save all steps in separate columns, no problem! Check this out:

Lets calculate x^2 +y^2, save it to xysquared,
check if xysquared is smaller or equal to 1, save it to checkCircle
and finally check if checkCircle is equal to insideCircle -> this is the only comparison we want to save in a variable, call that variable SKOROCEKRAJ

data.table MORE ADVANCED USAGE {} solved

```
ddf_dt[,SKOROCEKRAJ:= {
          xysquared = x^2 + y^2
          checkCircle = xysquared <= 1
          insideCircle == checkCircle
     }]
ddf_dt[,unique(SKOROCEKRAJ)]
```

```
## [1] TRUE
```

data.table exercise MORE ADVANCED USAGE {} solved

```
ddf_dt[,SKOROCEKRAJ:= {
          xysquared = x^2 + y^2
          checkCircle = xysquared <= 1
          insideCircle == checkCircle
     }]
```

Create a new variable sepal_length_diff as the difference from mean value:

data.table exercise MORE ADVANCED USAGE {} solved

Create a new variable sepal_length_diff as the difference from mean value:

```
iris_dt[, sepal_length_diff := {
  mean_sepal_length = mean(Sepal.Length)
  diff_from_avg = Sepal.Length - mean_sepal_length
  round(diff_from_avg, 1)
}]
```

data.table exercise MORE ADVANCED USAGE: merging

Lets define dummy data tables:

```
dt1 <- data.table(x = c("a", "b", "c", "d"), y = c(11.9, 21.4, 5.7, 1
dt2 <- data.table(x= c("a","b","k"),y = c(10, 15, 20), z = c("one", "
dt1
```

```
##    x    y
## 1: a 11.9
## 2: b 21.4
## 3: c  5.7
## 4: d 18.0
```

```
dt2
```

```
##    x  y     z
## 1: a 10   one
## 2: b 15   two
## 3: k 20 three
```

data.table exercise MORE ADVANCED USAGE: merging

Merge those two data tables by variable x

```
merge(dt1,dt2, by="x")
```

```
##    x  y.x y.y   z
## 1: a 11.9  10 one
## 2: b 21.4  15 two
```

```
merge(dt1,dt2, by="x", all.x = T)
merge(dt1,dt2, by="x", all.y = T)
```

data.table exercise MORE ADVANCED USAGE: merging

```
merge(dt1,dt2, by="x", all.y = T)
```

```
dt1[dt2, on = .(x)]
```

```
##    x    y i.y     z
## 1: a 11.9  10   one
## 2: b 21.4  15   two
## 3: k   NA  20 three
```

data.table exercise MORE ADVANCED USAGE: roll

```
dt1
```

```
##    x    y
## 1: a 11.9
## 2: b 21.4
## 3: c  5.7
## 4: d 18.0
```

```
dt2
```

```
##    x  y     z
## 1: a 10   one
## 2: b 15   two
## 3: k 20 three
```

keep rolling! - FORWARD JOIN:

Merge two data frames on CLOSEST SMALLER NUMERICAL VALUE in dt2 - keep all observations from dt1!:

```
dt2[dt1, on = .(y), roll = T]
```

```
##       x    y     z i.x
## 1:    a 11.9   one   a
## 2:    k 21.4 three   b
## 3: <NA>  5.7  <NA>   c
## 4:    b 18.0   two   d
```

keep rolling! - BACKWARD JOIN:

Merge two data frames on CLOSEST LARGER NUMERICAL VALUE in dt2 NOT LARGER THAN 4 - keep all observations from dt1!:

```
dt2[dt1, on = .(y), roll = -4]
```

```
##       x    y     z i.x
## 1:    b 11.9   two   a
## 2: <NA> 21.4  <NA>   b
## 3: <NA>  5.7  <NA>   c
## 4:    k 18.0 three   d
```

roll can also take "nearest".

data.table exercise MORE ADVANCED USAGE: foverlaps

```
dt3 <- data.table(min_y = c(0, 10, 15, 20), max_y = c(10, 15, 20, 30)
setkey(dt3, min_y, max_y)
dt1[, `:=` (dt1_y_end = c(13, 25, 10, 22), dt1_y=y)]
setkey(dt1, dt1_y, dt1_y_end)

foverlaps(dt1, dt3, type = "any")
```

```
##    min_y max_y x    y dt1_y_end dt1_y
## 1:     0    10 c  5.7        10   5.7
## 2:    10    15 c  5.7        10   5.7
## 3:    10    15 a 11.9        13  11.9
## 4:    15    20 d 18.0        22  18.0
## 5:    20    30 d 18.0        22  18.0
## 6:    20    30 b 21.4        25  21.4
```

Each window does not have to be equal

```
dt1
```