



DATA ANALYSIS THE DATA.TABLE WAY

The official Cheat Sheet for the [DataCamp](#) course



General form: `DT[i, j, by]` → “Take `DT`, subset rows using `i`, then calculate `j` grouped by `by`”

CREATE A DATA TABLE			
Create a <code>data.table</code> and call it <code>DT</code> .	<pre>library(data.table) set.seed(45L) DT <- data.table(V1=c(1L,2L), V2=LETTERS[1:3], V3=round(rnorm(4),4), V4=1:12)</pre>	<pre>> DT</pre>	<pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 B -0.3825 2 3: 1 C -1.0604 3 4: 2 A 0.6651 4 5: 1 B -1.1727 5 6: 2 C -0.3825 6 7: 1 A -1.0604 7 8: 2 B 0.6651 8 9: 1 C -1.1727 9 10: 2 A -0.3825 10 11: 1 B -1.0604 11 12: 2 C 0.6651 12</pre>

SUBSETTING ROWS USING <code>i</code>			
What?	Example	Notes	Output
Subsetting rows by numbers.	<code>DT[3:5,]</code> #or <code>DT[3:5]</code>	Selects third to fifth row.	<pre> V1 V2 V3 V4 1: 1 C -1.0604 3 2: 2 A 0.6651 4 3: 1 B -1.1727 5</pre>
Use column names to select rows in <code>i</code> based on a condition using fast automatic indexing. Or for selecting on multiple values: <code>DT[column %in% c("value1", "value2")]</code> , which selects all rows that have value1 or value2 in column.	<code>DT[V2 == "A"]</code> <code>DT[V2 %in% c("A", "C")]</code>	Selects all rows that have value A in column V2 . Select all rows that have the value A or C in column V2 .	<pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 A 0.6651 4 3: 1 A -1.0604 7 4: 2 A -0.3825 10</pre> <pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 1 C -1.0604 3 ... 7: 2 A -0.3825 10 8: 2 C 0.6651 12</pre>

MANIPULATING ON COLUMNS IN <code>j</code>			
What?	Example	Notes	Output
Select 1 column in <code>j</code> .	<code>DT[, V2]</code>	Column V2 is returned as a vector.	<pre>[1] "A" "B" "C" "A" "B" "C" ...</pre>
Select several columns in <code>j</code> .	<code>DT[, .(V2, V3)]</code>	Columns V2 and V3 are returned as a <code>data.table</code> .	<pre> V2 V3 1: A -1.1727 2: B -0.3825 3: C -1.0604 ...</pre>
<code>.</code> (<code>()</code>) is an alias to <code>list()</code> . If <code>.</code> (<code>()</code>) is used, the returned value is a <code>data.table</code> . If <code>.</code> (<code>()</code>) is not used, the result is a vector.			
Call functions in <code>j</code> .	<code>DT[, sum(V1)]</code>	Returns the sum of all elements of column V1 in a vector.	<pre>[1] 18</pre>
Computing on several columns.	<code>DT[, .(sum(V1), sd(V3))]</code>	Returns the sum of all elements of column V1 and the standard deviation of V3 in a <code>data.table</code> .	<pre> V1 V2 1: 18 0.7634655</pre>
Assigning column names to computed columns.	<code>DT[, .(Aggregate = sum(V1), Sd.V3 = sd(V3))]</code>	The same as above, but with new names.	<pre>Aggregate Sd.V3 1: 18 0.7634655</pre>
Columns get recycled if different length.	<code>DT[, .(V1, Sd.V3 = sd(V3))]</code>	Selects column V1 , and compute std. dev. of V3 , which returns a single value and gets recycled.	<pre> V1 Sd.V3 1: 1 0.7634655 2: 2 0.7634655 ... 11: 1 0.7634655 12: 2 0.7634655</pre>
Multiple expressions can be wrapped in curly braces.	<code>DT[, {print(V2) plot(V3) NULL}]</code>	Print column V2 and plot V3 .	<pre>[1] "A" "B" "C" "A" "B" "C" ... #And a plot</pre>

DOING <code>j</code> BY GROUP			
What?	Example	Notes	Output
Doing <code>j</code> by group.	<code>DT[, .(V4.Sum = sum(V4)), by=V1]</code>	Calculates the sum of V4 , for every group in V1 .	<pre> V1 V4.Sum 1: 1 36</pre>
Doing <code>j</code> by several groups using <code>.</code> (<code>()</code>).	<code>DT[, .(V4.Sum = sum(V4)), by=.(V1, V2)]</code>	The same as above, but for every group in V1 and V2 .	<pre> V1 V2 V4.Sum 1: 1 A 8 2: 2 B 10 3: 1 C 12 4: 2 A 14 5: 1 B 16 6: 2 C 18</pre>
Call functions in by .	<code>DT[, .(V4.Sum = sum(V4)), by=sign(V1-1)]</code>	Calculates the sum of V4 , for every group in sign(V1-1) .	<pre> sign V4.Sum 1: 0 36 2: 1 42</pre>
Assigning new column names in by .	<code>DT[, .(V4.Sum = sum(V4)), by=.(V1.01 = sign(V1-1))]</code>	Same as above, but with a new name for the variable we are grouping by.	<pre> V1.01 V4.Sum 2: 1 42</pre>
Grouping only on a subset by specifying <code>i</code> .	<code>DT[1:5, .(V4.Sum = sum(V4)), by=V1]</code>	Calculates the sum of V4 , for every group in V1 , after subsetting on the first five rows.	<pre> V1 V4.Sum 1: 1 9 2: 2 6</pre>
Using <code>.N</code> to get the total number of observations of each group.	<code>DT[, .N, by=V1]</code>	Count the number of rows for every group in V1 .	<pre> V1 N 2: 1 6 2: 2 6</pre>

ADDING/UPDATING COLUMNS BY REFERENCE IN <code>j</code> USING <code>:=</code>			
What?	Example	Notes	Output
Adding/updating a column by reference using <code>:=</code> in one line. Watch out: extra assignment (<code>DT <- DT[...]</code>) is redundant.	<code>DT[, V1 := round(exp(V1), 2)]</code>	Column V1 is updated by what is after <code>:=</code> .	Returns the result invisibly. Column V1 went from: <pre>[1] 1 2 1 2 ... to [1] 2.72 7.39 2.72 7.39 ...</pre>
Adding/updating several columns by reference using <code>:=</code> .	<code>DT[, c("V1", "V2") := list(round(exp(V1), 2), LETTERS[4:6])]</code>	Column V1 and V2 are updated by what is after <code>:=</code> .	Returns the result invisibly. Column V1 changed as above. Column V2 went from: <pre>[1] "A" "B" "C" "A" "B" "C" ...to: [1] "D" "E" "E" "D" "E" "E" ...</pre>
Using functional <code>:=</code> .	<code>DT[, ':= ' (V1 = round(exp(V1), 2), V2 = LETTERS[4:6])][1]</code>	Another way to write the same line as above this one, but easier to write comments side-by-side. Also, when <code>[1]</code> is added the result is printed to the screen.	Same changes as line above this one, but the result is printed to the screen because of the <code>[1]</code> at the end of the statement.
Remove a column instantly using <code>:=</code> .	<code>DT[, V1 := NULL]</code>	Removes column V1 .	Returns the result invisibly. Column V1 became NULL .
Remove several columns instantly using <code>:=</code> .	<code>DT[, c("V1", "V2") := NULL]</code>	Removes columns V1 and V2 .	Returns the result invisibly. Column V1 and V2 became NULL .
Wrap the name of a variable which contains column names in parenthesis to pass the contents of that variable to be deleted.	<code>Cols.chosen = c("A", "B")</code> <code>DT[, Cols.chosen := NULL]</code>	Watch out: this deletes the column with column name <code>Cols.chosen</code> .	Returns the result invisibly. Column with name <code>Cols.chosen</code> became NULL .
	<code>DT[, (Cols.chosen) := NULL]</code>	Deletes the columns specified in the variable <code>Cols.chosen</code> (V1 and V2).	Returns the result invisibly. Columns V1 and V2 became NULL .

INDEXING AND KEYS			
What?	Example	Notes	Output
Use <code>setkey()</code> to set a key on a <code>DT</code> . The data is sorted on the column we specified by reference.	<code>setkey(DT, V2)</code>	A key is set on column V2 .	Returns results invisibly.
Use keys like supercharged rownames to select rows.	<code>DT["A"]</code> <code>DT[c("A", "C")]</code>	Returns all the rows where the key column (set to column V2 in the line above) has the value A . Returns all the rows where the key column (V2) has the value A or C .	<pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 A 0.6651 4 3: 1 A -1.0604 7 4: 2 A -0.3825 10</pre> <pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 A 0.6651 4 ... 7: 1 C -1.1727 9 8: 2 C 0.6651 12</pre>
The <code>mult</code> argument is used to control which row that <code>i</code> matches to is returned, default is all.	<code>DT["A", mult = "first"]</code> <code>DT["A", mult = "last"]</code>	Returns first row of all rows that match the value A in the key column (V2). Returns last row of all rows that match the value A in the key column (V2).	<pre> V1 V2 V3 V4 1: 1 A -1.1727 1</pre> <pre> V1 V2 V3 V4 1: 2 A -0.3825 10</pre>
The <code>nomatch</code> argument is used to control what happens when a value specified in <code>i</code> has no match in the rows of the <code>DT</code> . Default is NA , but can be changed to 0. 0 means no rows will be returned for that non-matched row of <code>i</code> .	<code>DT[c("A", "D")]</code> <code>DT[c("A", "D"), nomatch = 0]</code>	Returns all the rows where the key column (V2) has the value A or D . A is found, D is not so NA is returned for D . Returns all the rows where the key column (V2) has the value A or D . Value D is not found and not returned because of the <code>nomatch</code> argument.	<pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 A 0.6651 4 3: 1 A -1.0604 7 4: 2 A -0.3825 10 5: NA D NA NA</pre> <pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 A 0.6651 4 3: 1 A -1.0604 7 4: 2 A -0.3825 10</pre>
<code>by=.EACHI</code> allows to group by each subset of known groups in <code>i</code> . A key needs to be set to use <code>by=.EACHI</code> .	<code>DT[c("A", "C"), sum(V4)]</code> <code>DT[c("A", "C"), sum(V4), by=.EACHI]</code>	Returns one total sum of column V4 , for the rows of the key column (V2) that have values A or C . Returns one sum of column V4 for the rows of column V2 that have value A , and another sum for the rows of column V2 that have value C .	<pre>[1] 52</pre> <pre> V2 V1 1: A 22 2: C 30</pre>
Any number of columns can be set as key using <code>setkey()</code> . This way rows can be selected on 2 keys which is an equijoin.	<code>setkey(DT, V1, V2)</code> <code>DT[, (2, "C")]</code> <code>DT[, (2, c("A", "C"))]</code>	Sorts by column V1 and then by column V2 within each group of column V1 . Selects the rows that have the value 2 for the first key (column V1) and the value C for the second key (column V2). Selects the rows that have the value 2 for the first key (column V1) and within those rows the value A or C for the second key (column V2).	Returns results invisibly. <pre> V1 V2 V3 V4 1: 2 C -0.3825 6 2: 2 C 0.6651 12</pre> <pre> V1 V2 V3 V4 1: 2 A 0.6651 4 2: 2 A -0.3825 10 3: 2 C -0.3825 6 4: 2 C 0.6651 12</pre>

ADVANCED DATA TABLE OPERATIONS			
What?	Example	Notes	Output
<code>.N</code> contains the number of rows or the last row.	Usable in <code>i</code> : <code>DT[.N-1]</code> Usable in <code>j</code> : <code>DT[, .N]</code>	Returns the penultimate row of the <code>data.table</code> . Returns the number of rows.	<pre> V1 V2 V3 V4 1: 1 B -1.0604 11</pre> <pre>[1] 12</pre>
<code>.</code> (<code>()</code>) is an alias to <code>list()</code> and means the same. The <code>.</code> (<code>()</code>) notation is not needed when there is only one item in by or <code>j</code> .	Usable in <code>j</code> : <code>DT[, .(V2, V3)]</code> #or <code>DT[, list(V2, V3)]</code> Usable in by : <code>DT[, mean(V3), by=.(V1, V2)]</code>	Columns V2 and V3 are returned as a <code>data.table</code> . Returns the result of <code>j</code> , grouped by all possible combinations of groups specified in by .	<pre> V2 V3 1: A -1.1727 2: B 0.14130 3: C -1.11655 4: A 0.14130 5: B -1.11655 6: C 0.14130</pre> <pre> V1 V2 V3 1: 1 A -1.1727 1 2: 2 B -0.3825 2 3: 1 C -1.0604 3 4: 2 A -1.1727 4 5: 1 B -1.1727 5 6: 2 NA -0.3825 6 7: 1 A -1.0604 7 8: 2 B 0.6651 8</pre>
<code>.SD</code> is a <code>data.table</code> and holds all the values of all columns, except the one specified in by . It reduces programming time but keeps readability. <code>.SD</code> is only accessible in <code>j</code> .	<code>DT[, print(.SD), by=V2]</code> <code>DT[, .SD[c(1, .N)], by=V2]</code> <code>DT[, lapply(.SD, sum), by=V2]</code>	To look at what <code>.SD</code> contains. Selects the first and last row grouped by column V2 . Calculates the sum of all columns in <code>.SD</code> grouped by V2 .	<pre>#All of .SD (output too long to display here)</pre> <pre> V2 V1 V3 V4 1: A 1 -1.1727 1 2: A 2 -0.3825 10 3: B 2 -0.3825 2 4: B 1 -1.0604 11 5: C 1 -1.0604 3 6: C 2 0.6651 12</pre> <pre> V2 V1 V3 V4 1: A 6 -1.9505 22 2: B 6 -1.9505 26 3: C 6 -1.9505 30</pre>
<code>.SDcols</code> is used together with <code>.SD</code> , to specify a subset of the columns of <code>.SD</code> to be used in <code>j</code> .	<code>DT[, lapply(.SD, sum), by=V2, .SDcols = c("V3", "V4")]</code>	Same as above, but only for columns V3 and V4 of <code>.SD</code> .	<pre> V2 V3 V4 1: A -1.9505 22 2: B -1.9505 26 3: C -1.9505 30</pre>
<code>.SDcols</code> can be the result of a function call.	<code>DT[, lapply(.SD, sum), by=V2, .SDcols = paste0("V", 3:4)]</code>	Same result as the line above.	<pre> V2 V3 V4 3: C -1.9505 30</pre>

CHAINING HELPS TACK INTERMEDIATE TOGETHER AND AVOID (UNNECESSARY) INTERMEDIATE ASSIGNMENTS			
What?	Example	Notes	Output
Do 2 (or more) sets of statements at once by chaining them in one statement. This corresponds to <i>having</i> in SQL.	<code>DT<-DT[, .(V4.Sum = sum(V4)), by=V1]</code> <code>DT[V4.Sum > 40] #no chaining</code> <code>DT[, .(V4.Sum = sum(V4)), by=V1][V4.Sum > 40]</code>	First calculates sum of V4 , grouped by V1 . Then selects that group of which the sum is > 40 without chaining. Same as above, but with chaining.	<pre> V1 V4.Sum 1: 1 36 2: 2 42</pre> <pre> V1 V4.Sum 1: 2 42</pre>
Order the results by chaining.	<code>DT[, .(V4.Sum = sum(V4)), by=V1][order(-V1)]</code>	Calculates sum of V4 , grouped by V1 , and then orders the result on V1 .	<pre> V1 V4.Sum 1: 2 42 2: 1 36</pre>

USING THE <code>set()</code> -FAMILY			
What?	Example	Notes	Output
<code>set()</code> is used to repeatedly update rows and columns by reference. <code>Set()</code> is a loopable low overhead version of <code>:=</code> . Watch out: It can not handle grouping operations.	Syntax of <code>set()</code> : <code>for (i in from:to) set(DT, row, column, new value)</code> . <code>rows = list(3:4, 5:6)</code> <code>cols = 1:2</code> <code>for (i in seq_along(rows)) { set(DT, i=rows[i], j = cols[i], value = NA) }</code>	Sequence along the values of rows, and for the values of cols, set the values of those elements equal to NA .	Returns the result invisibly. <pre>> DT</pre> <pre> V1 V2 V3 V4 1: 1 A -1.1727 1 2: 2 B -0.3825 2 3: NA C -1.0604 3 4: NA A 0.6651 4 5: 1 NA -1.1727 5 6: 2 NA -0.3825 6 7: 1 A -1.0604 7 8: 2 B 0.6651 8</pre>
<code>setnames()</code> is used to create or update column names by reference.	<code>setnames(DT, "old", "new") [1]</code> <code>setnames(DT, "V2", "Rating")</code> <code>setnames(DT, c("V2", "V3"), c("V2.rating", "V3.DataCamp"))</code>	Changes (set) the name of column old to new . Also, when <code>[1]</code> is added at the end of any <code>set()</code> function the result is printed to the screen. Sets the name of column V2 to Rating . Changes two column names.	Returns the result invisibly. Returns the result invisibly.
<code>setcolorder()</code> is used to reorder columns by reference.	<code>setcolorder(DT, "neworder")</code> <code>setcolorder(DT, c("V2", "V1", "V4", "V3"))</code>	neworder is a character vector of the new column name ordering. Changes the column ordering to the contents of the vector.	Returns the result invisibly. The new column order is now <pre>[1] "V2" "V1" "V4" "V3"</pre>