



# 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 data.table and call it DT.	<pre>library(data.table) set.seed(45L) DT &lt;- data.table(V1=c(1L,2L),                  V2=LETTERS[1:3],                  V3=round(rnorm(4),4),                  V4=1:12)</pre>	> DT	<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 <code>value1</code> or <code>value2</code> in column.	<code>DT[V2 == "A"]</code>  <code>DT[V2 %in% c("A", "C")]</code>	Selects all rows that have value <code>A</code> in column <code>V2</code> .  Select all rows that have the value <code>A</code> or <code>C</code> in column <code>V2</code> .	<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 <code>V2</code> is returned as a vector.	<code>[1]</code> "A" "B" "C" "A" "B" "C" ...
Select several columns in <code>j</code> .	<code>DT[, .(V2, V3)]</code>	Columns <code>V2</code> and <code>V3</code> are returned as a data.table.	<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 data.table. 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 <code>V1</code> in a vector.	<code>[1]</code> 18
Computing on several columns.	<code>DT[, .(sum(V1), sd(V3))]</code>	Returns the sum of all elements of column <code>V1</code> and the standard deviation of <code>V3</code> in a data.table.	<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 <code>V1</code> , and compute std. dev. of <code>V3</code> , 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 <code>V2</code> and plot <code>V3</code> .	<code>[1]</code> "A" "B" "C" "A" "B" "C" ... #And a plot

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 <code>V4</code> , for every group in <code>V1</code> .	<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 <code>V1</code> and <code>V2</code> .	<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 <code>by</code> .	<code>DT[, .(V4.Sum = sum(V4)), by=sign(V1-1)]</code>	Calculates the sum of <code>V4</code> , for every group in <code>sign(V1-1)</code> .	<pre>sign V4.Sum 1: 0 36 2: 1 42</pre>
Assigning new column names in <code>by</code> .	<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 1: 0 36 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 <code>V4</code> , for every group in <code>V1</code> , 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 <code>V1</code> .	<pre>V1 N 1: 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. <b>Watch out:</b> extra assignment ( <code>DT &lt;- DT[...]</code> ) is redundant.	<code>DT[, V1 := round(exp(V1), 2)]</code>	Column <code>V1</code> is updated by what is after <code>:=</code> .	Returns the result invisibly. Column <code>V1</code> went from: <code>[1]</code> 1 2 1 2 ... to <code>[1]</code> 2.72 7.39 2.72 7.39 ...
Adding/updating several columns by reference using <code>:=</code> .	<code>DT[, c("V1", "V2") := list(round(exp(V1), 2), LETTERS[4:6])]</code>	Column <code>V1</code> and <code>V2</code> are updated by what is after <code>:=</code> .	Returns the result invisibly. Column <code>V1</code> changed as above. Column <code>V2</code> went from: <code>[1]</code> "A" "B" "C" "A" "B" "C" ...to: <code>[1]</code> "D" "E" "E" "D" "E" "E" ...
Using functional <code>:=</code> .	<code>DT[, `:=`(V1 = round(exp(V1), 2), V2 = LETTERS[4:6])][, ]</code>	Another way to write the same line as above this one, but easier to write comments side-by-side. Also, when <code>[]</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>[]</code> at the end of the statement.
Remove a column instantly using <code>:=</code> .	<code>DT[, V1 := NULL]</code>	Removes column <code>V1</code> .	Returns the result invisibly. Column <code>V1</code> became <code>NULL</code> .
Remove several columns instantly using <code>:=</code> .	<code>DT[, c("V1", "V2") := NULL]</code>	Removes columns <code>V1</code> and <code>V2</code> .	Returns the result invisibly. Column <code>V1</code> and <code>V2</code> became <code>NULL</code> .
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>	<b>Watch out:</b> this deletes the column with column name <code>Cols.chosen</code> .	Returns the result invisibly. Column with name <code>Cols.chosen</code> became <code>NULL</code> .
	<code>DT[, (Cols.chosen) := NULL]</code>	Deletes the columns specified in the variable <code>Cols.chosen</code> ( <code>V1</code> and <code>V2</code> ).	Returns the result invisibly. Columns <code>V1</code> and <code>V2</code> became <code>NULL</code> .

INDEXING AND KEYS			
What?	Example	Notes	Output
Use <code>setkey()</code> to set a key on a DT. The data is sorted on the column we specified by reference.	<code>setkey(DT, V2)</code>	A key is set on column <code>V2</code> .	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 <code>V2</code> in the line above) has the value <code>A</code> .  Returns all the rows where the key column ( <code>V2</code> ) has the value <code>A</code> or <code>C</code> .	<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 <code>A</code> in the key column ( <code>V2</code> ).  Returns last row of all rows that match the value <code>A</code> in the key column ( <code>V2</code> ).	<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 DT. Default is <code>NA</code> , but can be changed to <code>0</code> . <code>0</code> 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 ( <code>V2</code> ) has the value <code>A</code> or <code>D</code> . <code>A</code> is found, <code>D</code> is not so <code>NA</code> is returned for <code>D</code> .  Returns all the rows where the key column ( <code>V2</code> ) has the value <code>A</code> or <code>D</code> . Value <code>D</code> 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</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 <code>V4</code> , for the rows of the key column ( <code>V2</code> ) that have values <code>A</code> or <code>C</code> .  Returns one sum of column <code>V4</code> for the rows of column <code>V2</code> that have value <code>A</code> , and another sum for the rows of column <code>V2</code> that have value <code>C</code> .	<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 <code>V1</code> and then by column <code>V2</code> within each group of column <code>V1</code> .  Selects the rows that have the value <code>2</code> for the first key (column <code>V1</code> ) and the value <code>C</code> for the second key (column <code>V2</code> ).  Selects the rows that have the value <code>2</code> for the first key (column <code>V1</code> ) and within those rows the value <code>A</code> or <code>C</code> for the second key (column <code>V2</code> ).	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 data.table.  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 <code>by</code> or <code>j</code> .	Usable in <code>j</code> : <code>DT[, .(V2, V3)]</code> #or <code>DT[, list(V2, V3)]</code>	Columns <code>V2</code> and <code>V3</code> are returned as a data.table.	<pre>V2 V3 1: A -1.1727 2: B -0.3825 3: C -1.0604 ... </pre>
<code>.SD</code> is a data.table and holds all the values of all columns, except the one specified in <code>by</code> . 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>	To look at what <code>.SD</code> contains.  Selects the first and last row grouped by column <code>V2</code> .	#All of .SD (output too long to display here)  <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>
<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 <code>V3</code> and <code>V4</code> of <code>.SD</code> .	<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> 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 V1 V3 V4 1: A 6 -1.9505 22 2: B 6 -1.9505 26 3: C 6 -1.9505 30</pre>

CHAINING HELPS TACKLE INTERMEDIATE ASSIGNMENTS 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 &lt;- DT[, .(V4.Sum = sum(V4)), by=V1]</code> <code>DT[V4.Sum &gt; 40] #no chaining</code>  <code>DT[, .(V4.Sum = sum(V4)), by=V1][V4.Sum &gt; 40]</code>	First calculates sum of <code>V4</code> , grouped by <code>V1</code> . 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 <code>V4</code> , grouped by <code>V1</code> , and then orders the result on <code>V1</code> .	<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> . <b>Watch out:</b> 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 <code>NA</code> .	Returns the result invisibly.  <pre>&gt; DT 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 <code>old</code> to <code>new</code> . 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 <code>V2</code> to <code>Rating</code> . Returns the result invisibly.  Changes two column names. 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>	<code>neworder</code> 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 <code>[1]</code> "V2" "V1" "V4" "V3"