



# 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` and call it `DT`.

```
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)
```

```
> DT
```

	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

SUBSETTING ROWS USING `i`

What?	Example	Notes	Output																																								
Subsetting rows by numbers.	<code>DT[3:5,]</code> #or <code>DT[3:5]</code>	Selects third to fifth row.	<table><thead><tr><th>V1</th><th>V2</th><th>V3</th><th>V4</th></tr></thead><tbody><tr><td>1:</td><td>C</td><td>-1.0604</td><td>3</td></tr><tr><td>2:</td><td>A</td><td>0.6651</td><td>4</td></tr><tr><td>3:</td><td>B</td><td>-1.1727</td><td>5</td></tr></tbody></table>	V1	V2	V3	V4	1:	C	-1.0604	3	2:	A	0.6651	4	3:	B	-1.1727	5																								
V1	V2	V3	V4																																								
1:	C	-1.0604	3																																								
2:	A	0.6651	4																																								
3:	B	-1.1727	5																																								
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 <b>value1</b> or <b>value2</b> in column.	<code>DT[ V2 == "A"]</code>  <code>DT[ V2 %in% c("A", "C")]</code>	Selects all rows that have value <b>A</b> in column <b>V2</b> .  Select all rows that have the value <b>A</b> or <b>C</b> in column <b>V2</b> .	<table><thead><tr><th>V1</th><th>V2</th><th>V3</th><th>V4</th></tr></thead><tbody><tr><td>1:</td><td>A</td><td>-1.1727</td><td>1</td></tr><tr><td>2:</td><td>A</td><td>0.6651</td><td>4</td></tr><tr><td>3:</td><td>A</td><td>-1.0604</td><td>7</td></tr><tr><td>4:</td><td>A</td><td>-0.3825</td><td>10</td></tr></tbody></table> <table><thead><tr><th>V1</th><th>V2</th><th>V3</th><th>V4</th></tr></thead><tbody><tr><td>1:</td><td>A</td><td>-1.1727</td><td>1</td></tr><tr><td>2:</td><td>C</td><td>-1.0604</td><td>3</td></tr><tr><td>7:</td><td>A</td><td>-0.3825</td><td>10</td></tr><tr><td>8:</td><td>C</td><td>0.6651</td><td>12</td></tr></tbody></table>	V1	V2	V3	V4	1:	A	-1.1727	1	2:	A	0.6651	4	3:	A	-1.0604	7	4:	A	-0.3825	10	V1	V2	V3	V4	1:	A	-1.1727	1	2:	C	-1.0604	3	7:	A	-0.3825	10	8:	C	0.6651	12
V1	V2	V3	V4																																								
1:	A	-1.1727	1																																								
2:	A	0.6651	4																																								
3:	A	-1.0604	7																																								
4:	A	-0.3825	10																																								
V1	V2	V3	V4																																								
1:	A	-1.1727	1																																								
2:	C	-1.0604	3																																								
7:	A	-0.3825	10																																								
8:	C	0.6651	12																																								

MANIPULATING ON COLUMNS IN `j`

What?	Example	Notes	Output
Select 1 column in <code>j</code> .	<code>DT[, V2]</code>	Column <b>V2</b> is returned as a vector.	<pre>[1] "A" "B" "C" "A"</pre>

MANIPULATING ON COLUMNS IN <code>j</code>																
What?	Example	Notes	Output													
Select 1 column in <code>j</code> .	<code>DT[, V2]</code>	Column <b>V2</b> is returned as a vector.	<code>[1]</code> "A" "B" "C" ...	<table><tr><th>V2</th><th>V3</th></tr><tr><td>A</td><td>-1.1727</td></tr><tr><td>B</td><td>-0.3825</td></tr><tr><td>C</td><td>-1.0604</td></tr><tr><td>...</td><td>...</td></tr></table>	V2	V3	A	-1.1727	B	-0.3825	C	-1.0604	...	...		
V2	V3															
A	-1.1727															
B	-0.3825															
C	-1.0604															
...	...															
Select several columns in <code>j</code> .	<code>DT[, .(V2, V3)]</code>	Columns <b>V2</b> and <b>V3</b> are returned as a <code>data.table</code> .		<table><tr><th>V2</th><th>V3</th></tr><tr><td>A</td><td>-1.1727</td></tr><tr><td>B</td><td>-0.3825</td></tr><tr><td>C</td><td>-1.0604</td></tr><tr><td>...</td><td>...</td></tr></table>	V2	V3	A	-1.1727	B	-0.3825	C	-1.0604	...	...		
V2	V3															
A	-1.1727															
B	-0.3825															
C	-1.0604															
...	...															
<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 <b>V1</b> 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 <b>V1</b> and the standard deviation of <b>V3</b> in a <code>data.table</code> .		<table><tr><th>V1</th><th>V2</th></tr><tr><td>1</td><td>18 0.7634655</td></tr></table>	V1	V2	1	18 0.7634655								
V1	V2															
1	18 0.7634655															
Assigning column names to computed columns.	<code>DT[, .(Aggregate = sum(V1), Sd.V3 = sd(V3))]</code>	The same as above, but with new names.	Aggregate 1:	<table><tr><th>Sd.V3</th></tr><tr><td>18 0.7634655</td></tr></table>	Sd.V3	18 0.7634655										
Sd.V3																
18 0.7634655																
Columns get recycled if different length.	<code>DT[, .(V1, Sd.V3 = sd(V3))]</code>	Selects column <b>V1</b> , and compute std. dev. of <b>V3</b> , which returns a single value and gets recycled.	<table><tr><th>V1</th><th>Sd.V3</th></tr><tr><td>1</td><td>1 0.7634655</td></tr><tr><td>2</td><td>2 0.7634655</td></tr><tr><td>...</td><td>...</td></tr><tr><td>11</td><td>1 0.7634655</td></tr><tr><td>12</td><td>2 0.7634655</td></tr></table>	V1	Sd.V3	1	1 0.7634655	2	2 0.7634655	...	...	11	1 0.7634655	12	2 0.7634655	
V1	Sd.V3															
1	1 0.7634655															
2	2 0.7634655															
...	...															
11	1 0.7634655															
12	2 0.7634655															
Multiple expressions can be wrapped in curly braces.	<code>DT[, {print(V2) plot(V3) NULL}]</code>	Print column <b>V2</b> and plot <b>V3</b> .	<code>[1]</code> "A" "B" "C" "A" "B" "C" ... #And a plot													

DOING <code>j</code> <b>BY</b> GROUP					
What?	Example	Notes	Output		
Doing <code>j</code> <b>by</b> group.	<code>DT[, . (V4.Sum = sum(V4)) ,by=V1]</code>	Calculates the sum of <b>V4</b> , for every group in <b>V1</b> .	V1	V4.Sum	
			1:	1 36	
Doing <code>j</code> <b>by</b> 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 <b>V1</b> and <b>V2</b> .	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
Call functions in <b>by</b> .	<code>DT[, . (V4.Sum = sum(V4)) ,by=sign(V1-1)]</code>	Calculates the sum of <b>V4</b> , for every group in <b>sign(V1-1)</b> .	sign	V4.Sum	
			1:	0 36	
			2:	1 42	
Assigning new column names in <b>by</b> .	<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.	V1.01	V4.Sum	
			1:	0 36	
			2:	1 42	
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 <b>V4</b> , for every group in <b>V1</b> , after subsetting on the first five rows.	V1	V4.Sum	
			1:	1 9	
			2:	2 6	
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 <b>V1</b> .	V1	N	
			1:	1 6	
			2:	2 6	

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 <b>V1</b> is updated by what is after <code>:=</code> .	Returns the result invisibly. Column <b>V1</b> went from: <code>[1] 1 2 1 2 ... to [1] 2.72 7.39 2.72 7.39 ...</code>	
Adding/updating several columns by reference using <code>:=</code> .	<code>DT[, c("V1", "V2") := list(round(exp(V1), 2), LETTERS[4:6])]</code>	Column <b>V1</b> and <b>V2</b> are updated by what is after <code>:=</code> .	Returns the result invisibly. Column <b>V1</b> changed as above. Column <b>V2</b> went from: <code>[1] "A" "B" "C" "A" "B" "C" ... to: [1] "D" "E" "F" "D" "E" "F" ...</code>	
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 <b>V1</b> .	Returns the result invisibly. Column <b>V1</b> became <b>NULL</b> .	
Remove several columns instantly using <code>:=</code> .	<code>DT[, c("V1", "V2") := NULL]</code>	Removes columns <b>V1</b> and <b>V2</b> .	Returns the result invisibly. Column <b>V1</b> and <b>V2</b> became <b>NULL</b> .	
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 <b>NULL</b> .	
	<code>DT[, (Cols.chosen) := NULL]</code>	Deletes the columns specified in the variable <code>Cols.chosen</code> ( <b>V1</b> and <b>V2</b> ).	Returns the result invisibly. Columns <b>V1</b> and <b>V2</b> became <b>NULL</b> .	

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 <b>V2</b> .	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 <b>V2</b> in the line above) has the value <b>A</b> .  Returns all the rows where the key column ( <b>V2</b> ) has the value <b>A</b> or <b>C</b> .	V1 V2 1: 1 A 2: 2 A 3: 1 A 4: 2 A	V3 V4 -1.1727 1 0.6651 4 -1.0604 7 -0.3825 10
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 <b>A</b> in the key column ( <b>V2</b> ).  Returns last row of all rows that match the value <b>A</b> in the key column ( <b>V2</b> ).	V1 V2 1: 1 A 8: 2 A	V3 V4 -1.1727 1 0.6651 12
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 <b>NA</b> , but can be changed to 0. <b>0</b> 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 ( <b>V2</b> ) has the value <b>A</b> or <b>D</b> . <b>A</b> is found, <b>D</b> is not so <b>NA</b> is returned for <b>D</b> .  Returns all the rows where the key column ( <b>V2</b> ) has the value <b>A</b> or <b>D</b> . Value <b>D</b> is not found and not returned because of the <code>nomatch</code> argument.	V1 V2 1: 1 A 2: 2 A 3: 1 A 4: 2 A 5: NA	V3 V4 -1.1727 1 0.6651 4 -1.0604 7 -0.3825 10 D NA NA
<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 <b>V4</b> , for the rows of the key column ( <b>V2</b> ) that have values <b>A</b> or <b>C</b> .  Returns one sum of column <b>V4</b> for the rows of column <b>V2</b> that have value <b>A</b> , and another sum for the rows of column <b>V2</b> that have value <b>C</b> .	[1] 52  V2 V1 1: A 22 2: C 30	
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 <b>V1</b> and then by column <b>V2</b> within each group of column <b>V1</b> .  Selects the rows that have the value <b>2</b> for the first key (column <b>V1</b> ) and the value <b>C</b> for the second key (column <b>V2</b> ).  Selects the rows that have the value 2 for the first key (column <b>V1</b> ) and within those rows the value <b>A</b> or <b>C</b> for the second key (column <b>V2</b> ).	V1 V2 1: 2 C 2: 2 C	V3 V4 -0.3825 6 0.6651 12
			V1 V2 1: 2 A 2: 2 A 3: 2 C 4: 2 C	V3 V4 0.6651 4 -0.3825 10 -0.3825 6 0.6651 12

ADVANCED DATA TABLE OPERATIONS					
What?	Example	Notes	Output		
.N contains the number of rows or the last row.	Usable in i: DT[.N-1]	Returns the penultimate row of the data.table.	V1	V2	V3 V4
	Usable in j: DT[, .N]	Returns the number of rows.	1: 1	B	-1.0604 11
.( ) is an alias to list() and means the same. The . ( ) notation is not needed when there is only one item in by or j.	Usable in j: DT[, .(V2,V3)] #or DT[, list(V2,V3)]	Columns V2 and V3 are returned as a data.table.	V2	V3	
	Usable in by: DT[, mean(V3), by=.(V1,V2)]	Returns the result of j, grouped by all possible combinations of groups specified in by.	1: 1	A	-1.1727
			2: 2	B	-0.3825
			3: 1	C	-1.11655
			4: 2	A	0.14130
			5: 1	B	-1.11655
			6: 2	C	0.14130
					...
.SD is a data.table and holds all the values of all columns, except the one specified in by. It reduces programming time but keeps readability. .SD is only accessible in j.	DT[, print(.SD), by=V2]	To look at what .SD contains.	#All of .SD (output too long to display here)		
	DT[, .SD[c(1, .N)], by=V2]	Selects the first and last row grouped by column V2.	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
	DT[, lapply(.SD, sum), by=V2]	Calculates the sum of all columns in .SD grouped by V2.	V2	V1	V3 V4
			1: A	6	-1.9505 22
			2: B	6	-1.9505 26
			3: C	6	-1.9505 30
.SDcols is used together with .SD, to specify a subset of the columns of .SD to be used in j.	DT[, lapply(.SD, sum), by=V2, .SDcols = c("V3", "V4")]	Same as above, but only for columns V3 and V4 of .SD.	V2	V3 V4	
			1: A	-1.9505 22	
.SDcols can be the result of a function call.	DT[, lapply(.SD, sum), by=V2, .SDcols = paste0("V", 3:4)]	Same result as the line above.	2: B	-1.9505 26	
			3: C	-1.9505 30	
CHAINING HELPS TACK EXPRESSIONS 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.	DT<-DT[, . (V4.Sum = sum(V4)),by=V1]	First calculates sum of <b>V4</b> , grouped by <b>V1</b> . Then selects that group of which the sum is > 40 without chaining.	V1	V4.Sum
	DT[V4.Sum > 40] #no chaining		1: 1	36
			2: 2	42
	DT[, . (V4.Sum = sum(V4)),by=V1][V4.Sum > 40 ]	Same as above, but with chaining.	V1	V4.Sum
			1: 2	42
Order the results by chaining.	DT[, . (V4.Sum = sum(V4)),by=V1][order(~V1)]	Calculates sum of <b>V4</b> , grouped by <b>V1</b> , and then orders the result on <b>V1</b> .	V1	V4.Sum
			1: 2	42
			2: 1	36
USING THE set()-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))</code> <code>{ set(DT,</code> <code>  i=rows[i],</code> <code>  j = cols[i],</code> <code>  value = NA) }</code>	Sequence along the values of rows, and for the values of cols, set the values of those elements equal to <b>NA</b> .	Returns the result invisibly.  <code>&gt; DT</code>  <table><thead><tr><th></th><th>V1</th><th>V2</th><th></th><th>V3</th><th>V4</th></tr></thead><tbody><tr><td>1:</td><td>1</td><td>A</td><td>-1.1727</td><td>1</td><td></td></tr><tr><td>2:</td><td>2</td><td>B</td><td>-0.3825</td><td>2</td><td></td></tr><tr><td>3:</td><td>NA</td><td>C</td><td>-1.0604</td><td>3</td><td></td></tr><tr><td>4:</td><td>NA</td><td>A</td><td>0.6651</td><td>4</td><td></td></tr><tr><td>5:</td><td>1</td><td>NA</td><td>-1.1727</td><td>5</td><td></td></tr><tr><td>6:</td><td>2</td><td>NA</td><td>-0.3825</td><td>6</td><td></td></tr><tr><td>7:</td><td>1</td><td>A</td><td>-1.0604</td><td>7</td><td></td></tr><tr><td>8:</td><td>2</td><td>B</td><td>0.6651</td><td>8</td><td></td></tr></tbody></table>		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	
	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																																																					
<code>setnames()</code> is used to create or update column names by reference.	Syntax of <code>setnames()</code> : <code>setnames(DT, "old", "new") [1]</code>  <code>setnames(DT, "V2", "Rating")</code>	Changes (set) the name of column <b>old</b> to <b>new</b> . 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 <b>V2</b> to <b>Rating</b> . Returns the result invisibly.																																																							