commalists-tools

Macros for manipulating numeral comma separated lists: sorting, adding, removing, etc

Version 0.20a – 20/02/2026

Cédric Pierquet
c pierquet – at – outlook . fr
https://github.com/cpierquet/latex-packages/tree/main/commalists-tools

---

```
\def\mytestlist{14,10,15,11,9,10}
\lenoflist*{\mytestlist}
\minoflist*{\mytestlist}
\maxoflist*{14,10,15,11,9,10}
\sortasclist*{\mytestlist}
\meanoflist*{\mytestlist}
\removevalinlist*{10}{\mytestlist}
\testifvalinlist{15}{\mytestlist}{true}{false}
\getvaluefromlist*{\mytestlist}{3}
```

We consider the list: 14,10,15,11,9,10

There’s 6 values in the list

The minimum value is 9 and the maximum is 15

Ascending sorted list is 9,10,10,11,14,15

Meaning value of the list is 11.5

If we remove the value 10, then the list is 14,15,11,9

We test if 15 is in the list: true

The third value of the list is 15

Contents

1 Loading, useful packages

In order to load randintlist, simply use:

```
\usepackage{commalists-tools}
```

Loaded packages are listofitems, xintexpr and xstring.

The idea is to propose "reusage" of lists with listofitems or other commands, for example.

2 The individual macros

2.1 Global usage

Package commalists-tools supports (basic) manipulations on numeral comma separated lists:

- sorting;
- adding values;
- removing values;
- counting values;
- mean, sum, product;
- get value, get length;
- etc.

Starred versions only print the result, whereas non starred versions store the result into a macro.

All engines TEX are compatible with this package.

2.2 The macros for calculating

```
%getting length of list, only printing
\lenoflist*{list}
%storing length of list into macro (\resmylen by default)
\lenoflist{list}
%getting min of list, only printing
\minoflist*{list}
%storing min of list into \macro (\resmin by default)
\minoflist{list}[\macro]
%getting max of list, only printing
\maxoflist*{list}
%storing max of list into \macro (\resmax by default)
\maxoflist{list}[\macro]
```

```
%only printing
\lenoflist*{14,10,15,11,9,10}\\
%only printing
\minoflist*{14,10,15,11,9,10} and \maxoflist*{14,10,15,11,9,10}
```

6
9 and 15

```
%storing len/min/max of list
\def\mytestlist{10,14.5,20,12,8.5}
\lenoflist{\mytestlist}[\mylen]Length of list is \mylen\ \&
\minoflist{\mytestlist}[\mymin]Min of list is \mymin\ \&
\maxoflist{\mytestlist}[\mymax]Max of list is \mymax
```

Length of list is 5 & Min of list is 8.5 & Max of list is 20

```
%getting mean of list, only printing
\meanoflist*{list}
%storing mean of list into \macro (\resmean by default)
\meanoflist{list}[\macro]
%getting sum of list, only printing
\sumoflist*{list}
%storing max of list into \macro (\ressum by default)
\sumoflist{list}[\macro]
%getting prod of list, only printing
\prodoflist*{list}
%storing max of list into \macro (\resprod by default)
\prodoflist{list}[\macro]
```

```
%only printing
\meanoflist*{14,10,15,11,9,10} \\
%storing
\meanoflist{14,10,15,11,9,10}[\mymean]\mymean \\
%only printing
\sumoflist*{14,10,15,11,9,10} and \prodoflist*{14,10,15,11,9,10} \\
%storing
\sumoflist{14,10,15,11,9,10}[\mysum]\prodoflist{14,10,15,11,9,10}[\myprod]
The sum is \mysum\ and the prod is \myprod
```

11.5
11.5
69 and 2079000
The sum is 69 and the prod is 2079000

2.3 Macros for manipulating

```
%sorting (asc), only printing
\sortasclist*{list}
%sorting (asc) and storing (overwrite)
\sortasclist{list}
%sorting (des), only printing
\sortdeslist*{list}
%sorting (des) and storing (overwrite)
\sortdeslist{list}
```

```
%sorting (asc), only printing
\sortasclist*{14,10,15,11.5,9,10}\\
%sorting (asc) and storing (overwrite)
\def\tmpsortlist{14,10,15,11.5,9,10}
\sortasclist{\tmpsortlist}\tmpsortlist\\
%analysing
\readlist*\tmpSORTlist{\tmpsortlist}
\showitems{\tmpSORTlist}
```

9,10,10,11.5,14,15
9,10,10,11.5,14,15
9 10 10 11.5 14 15

```
%extract value, only printing
\getvaluefromlist*{list}{index}
%extract value and storing into macro (\myelt by default)
\getvaluefromlist{list}{index}[\macro]
```

```
%extract value, only printing
\def\listtmp{1,2,3,6,3,1,5,7,3}%
\getvaluefromlist*{\listtmp}{4}\par\smallskip
%storing
\getvaluefromlist{\listtmp}{-1}[\mylastelt]The last element is \mylastelt
```

6

The last element is 3

```
%sorting (des), only printing
\sortdeslist*{14,10,15,11.5,9,10}\\
%sorting (asc) and storing (overwrite)
\def\tmpsortlist{14,10,15,11.5,9,10}
\sortdeslist{\tmpsortlist}\tmpsortlist\\
%analysing
\readlist*\tmpSORTlist{\tmpsortlist}
\showitems{\tmpSORTlist}
```

15,14,11.5,10,10,9
15,14,11.5,10,10,9
15 14 11.5 10 10 9

```
%adding, only printing
\addvalinlist*{values}{list}
%adding and storing (overwrite)
\addvalinlist{values}{list}
%removing, only printing
\removevalinlist*{value}{list}
%removing and storing (overwrite)
\removevalinlist{value}{list}
```

```
%only printing
\addvalinlist*{3}{1,2,5,6}\\
%defining and adding
\def\tmpaddlist{1,2,4,5,6}
\addvalinlist{3}{\tmpaddlist}\tmpaddlist\\
%analysing
\readlist*\tmpADDlist{\tmpaddlist}
\showitems{\tmpADDlist}
```

1,2,5,6,3
1,2,4,5,6,3
1 2 4 5 6 3

```
%only printing
\removevalinlist*{3}{1,2,3,6,3,1,5,7,3}\\
%defining and removing
\def\tmpremlist{1,2,3,6,3,1,5,7,3}
\removevalinlist{3}{\tmpremlist}\tmpremlist\\
%analysing
\readlist*\tmpREMlist{\tmpremlist}
\showitems{\tmpREMlist}
```

1,2,6,1,5,7
1,2,6,1,5,7
1 2 6 1 5 7

```
%reversing, only printing
\reverselist*{list}
%reversing and storing (overwrite)
\reverselist{list}
```

```
%only printing
\reverselist*{14,10,15,11,9,10}\\
%reversing and storing
%storing
\reverselist{14,10,15,11,9,10}[\myreverse]\myreverse\\
%analysing
\readlist*\tmpREVERSElist{\myreverse}
\showitems{\tmpREVERSElist}
```

10,9,11,15,10,14
10,9,11,15,10,14
10 9 11 15 10 14

2.4 Macros with testing

```
%testing if value is in list, with boolean result in \macro (\resisinlist by default)
\boolvalinlist{value}{list}[\macro]
%conditionnal test if value is in list, according to xint syntax
\testifvalinlist{3}{0,1,2,3}{true}{false}
```

```
%test with xint syntax
\testifvalinlist{-1}{0,1,2,3}{true}{false}\\
%test with xint syntax
\testifvalinlist{3}{0,1,2,3}{true}{false}\\
%boolean macro
\def\myteslist{0,5,10,5,6,9,7,8}
\boolvalinlist{5}{\myteslist}[\resisinlist]\resisinlist
```

false
true
1

```
%counting value, only printing
\countvalinlist*{value}{list}
%counting value, with result in \macro (\rescount by default)
\countvalinlist{value}{list}[\macro]
```

```
%only printing
\countvalinlist*{8}{1,2,3,4,5,6,6,7,8,8,8,9}\\
%storing
\def\tmpcountlist{1,2,3,4,5,6,6,7,8,8,8,9}
\countvalinlist{6}{\tmpcountlist}[\rescountsix]\rescountsix\\
\countvalinlist{10}{\tmpcountlist}[\rescountten]\rescountten
```

3
2
0

3 Macros for already processed lists

In order to speed up the process, it's possible to "define" a list (processed with `listofitems`) and the to call specific macros.

```
%define a list, to be processed by listofitems
\definemylist{rawlist}{processed list}

%Specific macros (same specs as above)
\lenofdeflist(*){processed list, without \ !}[\macro]
\countvalindeflist(*){val}{processed list, without \ !}[\macro]
\meanofdeflist(*){processed list, without \ !}[\macro]
\sumofdeflist(*){processed list, without \ !}[\macro]
\prodofdeflist(*){processed list, without \ !}[\macro]
\getvaluefromdeflist(*){processed list, without \ !}{index}[\macro]
```

```
%define a list, to be processed by listofitems
\definemylist{1,4,5,6,7,8,9,19,9,9,10}{\mydeflist}\showitems{\mydeflist}

%Specific macros
\lenofdeflist*{mydeflist}\\
\getvaluefromdeflist*{mydeflist}{5}\\
\countvalindeflist*{9}{mydeflist}\\
\meanofdeflist*{mydeflist}\\
\sumofdeflist*{mydeflist}\\
\prodofdeflist*{mydeflist}
```

1 4 5 6 7 8 9 19 9 9 10
11
7
3
7.909090909090909
87
930787200

If the list needs to be "modified" (add, remove, reverse) the best is to use individual macros, and after use *define* and specific macros :-)

4 History

0.20a: Improvements in L3 version of package
0.1.7: Improvements in L3 version of package
0.1.6: L3 version of package
0.1.5: Bugfix
0.1.4: Changing name of macro
0.1.3: Duplicate macro with ProfMaquette (so new name)
0.1.2: Alternative macros for already treated lists
0.1.1: Compatibility with decimals, len of list, get value
0.1.0: Initial version

5 The code

```
% Author     : C. Pierquet
% licence    : Released under the LaTeX Project Public License v1.3c or later, see
     http://www.latex-project.org/lppl.txt

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{commalists-tools}[2026/02/20 0.20a Basic operations for numeral comma separated lists]

%------History
% 0.20a same version number for both (legacy & l3) versions
% 0.1.7 Improvements with l3 package
% 0.1.6 L3 version of package
% 0.1.5 Bugfix
% 0.1.4 Change name of macro + bugfix
% 0.1.3 Duplicate macro with ProfMaquette (so new name)
% 0.1.2 New versions with already traited lists
% 0.1.1 Sorting with decimals (!)
% 0.1.0 Initial version

%------Packages
\RequirePackage{listofitems}
\RequirePackage{xintexpr}
\RequirePackage{xstring}

%-----Macros (LaTeX3...) for sorting
\ExplSyntaxOn
\NewDocumentCommand\sortasclist{ s m }
{
    \clist_set:Nx \l_foo_clist {#2}
    \clist_sort:Nn \l_foo_clist{
        \fp_compare:nNnTF { ##1 } > { ##2 }
            { \sort_return_swapped: }
            { \sort_return_same: }
    }
    \IfBooleanTF{#1}%if star := just printing // if not, storing
        {%
            {\l_foo_clist}%
        }%
        {%
            \xdef#2{\l_foo_clist}%
        }%
}

\NewDocumentCommand\sortdeslist{ s m }
{
    \clist_set:Nx \l_foo_clist {#2}
    \clist_sort:Nn \l_foo_clist{
        \fp_compare:nNnTF { ##1 } < { ##2 }
            { \sort_return_swapped: }
            { \sort_return_same: }
    }
    \IfBooleanTF{#1}%if star := just printing // if not, storing
        {%
```

```
            {\l_foo_clist}%
        }%
        {%
            \xdef#2{\l_foo_clist}%
        }%
}
\ExplSyntaxOff

%----Macros (LaTeX2)
\NewDocumentCommand\addvalinlist{ s m m }{%
    \IfBooleanTF{#1}%if star := just printing // if not, storing
        {%
            #3,#2%
        }%
        {%
            \xdef#3{#3,#2}%
        }%
}

\NewDocumentCommand\definemylist{ m m }{%
    %1 = raw list
    %2 = traited list with listofitems
    \setsepchar{,}%
    \readlist*#2{#1}%
}

\NewDocumentCommand\removevalinlist{ s m m }{%
    \setsepchar{,}%
    \readlist*\tmplistread{#3}%
    \xdef\tmpresremovelist{}%
    \xintFor* ##1 in {\xintSeq{1}{\tmplistreadlen}}\do{%
        \xintifboolexpr{ \tmplistread[##1] == #2}%
            {}%
            {%
                \xintifboolexpr{ ##1 == 1 }%
                    {%
                        \xdef\tmpresremovelist{\tmplistread[##1]}%
                    }%
                    {%
                        \xdef\tmpresremovelist{\tmpresremovelist,\tmplistread[##1]}%
                    }%
            }%
    }%
    \IfBooleanTF{#1}%if star := just printing // if not, storing
        {%
            \tmpresremovelist%
        }%
        {%
            \xdef#3{\tmpresremovelist}%
        }%
}

\NewDocumentCommand\boolvalinlist{ m m O{\resisinlist} }{%
    \IfSubStr{,#2,}{,#1,}{\def#3{1}}{\def#3{0}}%
}

\NewDocumentCommand\testifvalinlist{ m m m m }{%
    \IfSubStr{,#2,}{,#1,}{\xdef\RESTMPVALUE{1}}{\xdef\RESTMPVALUE{0}}%
    \xintifboolexpr{ \RESTMPVALUE == 1}{#3}{#4}%
}

\NewDocumentCommand\countvalinlist{ s m m O{\rescount} }{%
    \setsepchar{,}%
    \readlist*\tmplistread{#3}%
    \xdef#4{0}%
    \xintFor* ##1 in {\xintSeq{1}{\tmplistreadlen}}\do{%
        \xintifboolexpr{ \tmplistread[##1] == #2}%
        {%
            \xdef#4{\fpeval{#4+1}}%
        }%
        {}%
    }%
```

```
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #4%
        }%
}

\NewDocumentCommand\countvalindeflist{ s m m O{\rescount} }{%
    %\readlist*\tmplistread{#3}%
    \xdef#4{0}%
    \xintFor* ##1 in {\xintSeq{1}{\csname#3len\endcsname}}\do{%
        \xintifboolexpr{ \csname#3\endcsname[##1] == #2}%
        {%
            \xdef#4{\fpeval{#4+1}}%
        }%
        {}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #4%
        }%
}

\NewDocumentCommand\minoflist{ s m O{\resmin} }{%
    \IfBooleanTF{#1}%
        {%
            \fpeval{min(#2)}%
        }%
        {%
            \xdef#3{\fpeval{min(#2)}}%
        }%
}

\NewDocumentCommand\maxoflist{ s m O{\resmax} }{%
    \IfBooleanTF{#1}%
        {%
            \fpeval{max(#2)}%
        }%
        {%
            \xdef#3{\fpeval{max(#2)}}%
        }%
}

\NewDocumentCommand\meanoflist{ s m O{\resmean} }{%
    \xdef#3{0}%
    \setsepchar{,}%
    \readlist*\tmpmeanlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\tmpmeanlistlen}}\do{%
        \xdef#3{\fpeval{#3+\tmpmeanlist[##1]}}%
    }%
    \xdef#3{\fpeval{(#3)/\tmpmeanlistlen}}%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\meanofdeflist{ s m O{\resmean} }{%
    \xdef#3{0}%
    %\readlist*\tmpmeanlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\csname#2len\endcsname}}\do{%
        \xdef#3{\fpeval{#3+\csname#2\endcsname[##1]}}%
    }%
    \xdef#3{\fpeval{(#3)/\csname#2len\endcsname}}%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\sumoflist{ s m O{\ressum} }{%
    \xdef#3{0}%
    \setsepchar{,}%
```

```
    \readlist*\tmpsumlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\tmpsumlistlen}}\do{%
        \xdef#3{\fpeval{#3+\tmpsumlist[##1]}}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\sumofdeflist{ s m O{\ressum} }{%
    \xdef#3{0}%
    %\readlist*\tmpsumlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\csname#2len\endcsname}}\do{%
        \xdef#3{\fpeval{#3+\csname#2\endcsname[##1]}}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\prodoflist{ s m O{\resprod} }{%
    \xdef#3{1}%
    \setsepchar{,}%
    \readlist*\tmpprodlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\tmpprodlistlen}}\do{%
        \xdef#3{\fpeval{(#3)*(\tmpprodlist[##1])}}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\prodofdeflist{ s m O{\resprod} }{%
    \xdef#3{1}%
    \setsepchar{,}%
    \readlist*\tmpprodlist{#2}%
    \xintFor* ##1 in {\xintSeq{1}{\csname#2len\endcsname}}\do{%
        \xdef#3{\fpeval{(#3)*(\csname#2\endcsname[##1])}}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\reverselist{ s m O{\resrevlist} }{%
    \setsepchar{,}%
    \readlist*\tmpreverselist{#2}%
    \xdef#3{\tmpreverselist[-1]}%
    \xintFor* ##1 in {\xintSeq{2}{\tmpreverselistlen}}\do{%
        \xdef#3{#3,\tmpreverselist[-##1]}%
    }%
    \IfBooleanT{#1}%if star := just printing // if not, storing
        {%
            #3%
        }%
}

\NewDocumentCommand\getvaluefromlist{ s m m O{\resmyelt} }{%
    \setsepchar{,}%
    \readlist*\tmpreadlist{#2}%
    \xdef#4{\tmpreadlist[#3]}%
    \IfBooleanT{#1}{#4}%
}

\NewDocumentCommand\getvaluefromdeflist{ s m m O{\resmyelt} }{%
    %\readlist*\tmpreadlist{#2}%
    \xdef#4{\csname#2\endcsname[#3]}%
    \IfBooleanT{#1}{#4}%
```

```
}

\NewDocumentCommand\lenoflist{ s m O{\resmylen} }{%
    \setsepchar{,}%
    \readlist*\tmpreadlist{#2}%
    \xdef#3{\tmpreadlistlen}%
    \IfBooleanT{#1}{#3}%
}

\NewDocumentCommand\lenofdeflist{ s m O{\resmylen} }{%
    %\readlist*\tmpreadlist{#2}%
    \xdef#3{\csname#2len\endcsname}%
    \IfBooleanT{#1}{#3}%
}

\endinput
```