The \texttt{ran\_toks} Package

D. P. Story
Email: dpstory@uakron.edu
processed December 30, 2019

Contents

1 Alternate package name: \texttt{ran\_toks} \hfill 3
2 Commands for controlling the process \hfill 3
3 Utility commands \hfill 4
4 The main commands \hfill 7
   4.1 Additional user access commands \hfill 9
5 Commands that support a DB application \hfill 10
6 Index \hfill 12
7 Change History \hfill 14

1 \texttt{(*package)}

\textbf{Description.} This short package randomizes a list of tokens. The command, \texttt{\\ranToks}, takes one argument, which is a list of tokens:

\begin{verbatim}
\ranToks{\langle name \rangle}
{
  {\langle tok_1 \rangle}{\langle tok_2 \rangle}...{\langle tok_n \rangle}
}
\end{verbatim}

The command defines a series of $n$ internal commands, one for each of the tokens. The definitions are essentially randomized. The randomized tokens are accessed through the command \texttt{\useRanTok}. For example

\begin{verbatim}
\useRanTok{1}, \useRanTok{2},...\useRanTok{n}
\end{verbatim}

gives a random listing of the $n$ tokens. These can be arranged on the page as desired.

There is a second construct, designed for more elaborate randomization.
The contents of each of the \texttt{rtVW} environments are written to the computer's hard drive, then input back in random order, using \texttt{\useRanTok}, eg,

\begin{verbatim}
\useRanTok{1}, \useRanTok{2},..., \useRanTok{n}
\end{verbatim}

Other details are left to the readers' imagination.

\textbf{Requirements.} As of this writing, we require only the \texttt{verbatim} package and \texttt{random.tex}, the package was written by Donald Arseneau.

\begin{verbatim}
2 \RequirePackage{verbatim}
\end{verbatim}

\textbf{Input} \texttt{random.tex}. Input \texttt{random.tex} if not already input.

\begin{verbatim}
3 \@ifundefined{nextrandom}{\input{random.tex}}{}
\end{verbatim}

We redefine \texttt{nextrandom} from \texttt{random.tex} to save the initializing seed.

\begin{verbatim}
4 \def\nextrandom{\begingroup
5 \ifnum\randomi<\@ne % then initialize with time
6 \global\randomi\time
7 \global\multiply\randomi388 \global\advance\randomi\year
8 \global\multiply\randomi31 \global\advance\randomi\day
9 \global\multiply\randomi97 \global\advance\randomi\month
10 \message{Randomizer initialized to \the\randomi.}\
11 \nextrandom \nextrandom \nextrandom
12 \xdef\InitSeedValue{\the\randomi}%
13 \fi
14 \count@ii\randomi
15 \divide\count@ii 127773 % modulus = multiplier * 127773 + 2836
16 \count@\count@ii
17 \multiply\count@ii 127773
18 \global\advance\randomi-\count@ii % random mod 127773
19 \global\multiply\randomi 16807
20 \multiply\count@ 2836
21 \global\advance\randomi-\count@
22 \ifnum\randomi<\z@ \global\advance\randomi 2147483647\relax\fi
23 \endgroup
24 }
\end{verbatim}
The code for this package was taken from the dps package, and modified suitably. We use several token registers and count registers. This can probably be optimized.

We use several token registers and count registers. This can probably be optimized.

25 \newtoks\rt@listIn \rt@listIn={}
26 \newtoks\rt@newListIn \rt@newListIn={}
27 \newtoks\rt@listOut \rt@listOut={}
28 \newcount\rt@nMax
29 \newcount\rt@nCnt
30 \newcount\rt@getRanNum
31 \newif\ifrtdebug \rtdebugfalse
32 \newif\ifwerandomize \werandomizetrue
33 \newif\ifsaveseed \saveseedtrue
34 \newwrite\rt@Verb@write
35 \def\rt@nameedef#1{\expandafter\edef\csname #1\endcsname}
36 ⟨/package⟩
37 ⟨∗altpkgname⟩

1 Alternate package name: ran-toks

CTAN lists this package (ran_toks) as ran-toks, so we’ll create a dummy package by that name.

38 \NeedsTeXFormat{LaTeX2e}
39 \ProvidesPackage{ran-toks}[2019/12/28 v1.0 ran-toks Alt-name (dps)]
40 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{ran_toks}}
41 \ProcessOptions
42 \RequirePackage{ran_toks}[2019/12/28]
43 ⟨/package⟩
44 ⟨∗altpkgname⟩

2 Commands for controlling the process

\ranToksOn These two turn on and turn off randomization.
\ranToksOff
\def\ranToksOn{\werandomizetrue}
\def\ranToksOff{\werandomizefalse}
\useThisSeed initializes the random number generator. Use this to reproduce the same sequence
of pseudo-random numbers from an earlier run. We also set \saveseedfalse so we do not write the initial seed to the disk.
\useLastAsSeed initializes the random number generator using the last random seed. If the file
\jobname\_rt.sav does not exist, the generator will be initialized using time and date data.

\def\useThisSeed#1{\saveseedfalse \randomi=#1}
\@onlypreamble\useThisSeed
\def\useLastAsSeed{\rt@useLastAsSeed}
\@onlypreamble\useLastAsSeed

3
When \useLastAsSeed, the last becomes the first.

\def\InitSeedValue{\the\randomi}

A standard \verbatim write used in exerquiz and other package in the AeB family.

\verbatimwrite
\verbatim@start
We write only if \ifsaveseed is true.

\verbatim\verbatim@msgi{initializing seed value}\%
\verbatim\verbatim@msgii{last random number used}\%
\verbatim\verbatim@IWVO{\InitSeedValue \verbatim@msgi}
\verbatim\verbatim@IWVO{\randomi \verbatim@msgii}\immediate
\closeout\saveseedinfo

Save the initial seed value to hard drive.
\AtEndDocument{\verbatimwrite}
\rt@populateList\{n\}\ is a utility command, its argument \(n\) is a positive integer, and it generates a list of the form \(\{1\}\{2\}\ldots\{n\}\) and is held in the token register \rt@listIn. This listing is later randomly permuted by \rt@RandomizeList.

\begin{verbatim}
def \rt@populateList\#1{\rt@listIn={}\rt@nCnt\z@ 
  \@whilenum\rt@nCnt<#1\do{\advance\rt@nCnt\@ne
  \edef\rt@listInHold{\the\rt@listIn\noexpand\{\the\rt@nCnt\}}%
  \rt@listIn=\expandafter{\rt@listInHold}}}
\end{verbatim}

\rt@RandomizeList\{n\}\ is the command that gets the process of randomizing the input list going. The argument is the number \(n\) of tokens. If \werandomize is false, it just returns the input list; otherwise, it calls \rt@randomizeList to actually do the work.

\begin{verbatim}
def \rt@RandomizeList\#1{\global \rt@listIn={}
  \global \rt@newListIn={}
  \global \rt@listOut={}
  \rt@nMax=#1\relax
  \rt@populateList\{\the\rt@nMax\}
  \ifwerandomize
    \expandafter\rt@randomizeList
  \else
    \global \rt@listOut=\expandafter{\the\rt@listIn}\fi
  \global \rt@nameedef{\rt@BaseName-List}{\the\rt@listOut}}
\end{verbatim}

\rt@RandomizeList randomizes the list of consecutive integers, and leaves the results, \(\{k_1\}\{k_2\}\ldots\{k_n\}\) in the token register \rt@listOut. \rt@RandomizeList is a loop, looping between itself and \rt@loopTest.

\begin{verbatim}
def \rt@RandomizeList{\let\=\rt@processi 
  \setrannum{\rt@getRanNum}{1}{\the\rt@nMax}\%
  \ifrtdebug\typeout{string\rt@getRanNum=\the\rt@getRanNum}\fi
  \rt@nCnt\z@
  \ifrtdebug\typeout{LISTING:\ \the\rt@listIn}\fi
  \the\rt@listIn
  \rt@loopTest
  \rt@loopTest}
\end{verbatim}

In \rt@RandomizeList, we \let\=\rt@processi before dumping the contents of \rt@listIn. We then go into a loop \rt@loopTest. \rt@getRanNum is the random integer between 1 and \rt@nMax.
We perform modular arithmetic when the index of \texttt{\useRanTok} is too large. \texttt{\rt@modarith} performs modular arithmetic on its arguments \((\#1 \mod \#2)\) and returns the result in the macro \texttt{\rt@mod}.

This macro uses \texttt{\dimen0} and \texttt{\dimen2}, so it should be called within a group.

\texttt{\rt@modarith} performs modular arithmetic on its arguments \((\#1 \mod \#2)\) and returns the result in the macro \texttt{\rt@mod}.

Warning messages, these are \texttt{\rt@badIndex} and \texttt{\rt@badTokName}.

\texttt{\rt@modarith} performs modular arithmetic on its arguments \((\#1 \mod \#2)\) and returns the result in the macro \texttt{\rt@mod}.

\texttt{\rt@badIndex} Warning messages, these are \texttt{\rt@badIndex} and \texttt{\rt@badTokName}.
4 The main commands

\ranToks\{\langle token-list\rangle\} takes one argument, \{\langle token-list\rangle\}, a list of tokens. It randomizes them. The randomized listing can be accessed using \useRanTok.

\begin{verbatim}
\def\ranToks#1{\begingroup\useRTName{#1}\\r@nToks\long\def\r@nToks#1{\rt@nMax\z@\r@ndToks#1\rt@NIL}\\def\rt@NIL{\@nil}\useRTName{\langle name\rangle} sets the base name (use prior to the use of \useRanTok).\def\useRTName#1[1]{\gdef\rt@BaseName{#1}}\let\rt@BaseName\@empty\bRTVToks{\langle name\rangle} \bRTVToks and \eRTVToks enclose a series of rtVW environments. The single argument is the name of this set of verbatim write “tokens”.\def\eRTVToks{\global\rt@nameedef{\rt@BaseName Cnt}{\the\rt@nCnt}\\expandafter\r@nVToks\expandafter{\rt@BaseName}}\rtVW \rtVW is a verbatim write environment. It saves its contents to the file \jobname_.\rt@BaseName\the\rt@nCnt.cut. The file is later input back into the source file in a random way.\def\reVerbEnd{\ifhmode\unskip\fi}Insert the hook \rtVWHook prior to writing the verbatim content. The default is \relax.\def\rtVWHook#1{\def\@rgi{#1}\ifx\@rgi\@empty\let\RTVWHook\relax\else\def\RTVWHook{#1}\fi}\newenvironment{rtVW}{\global\advance\rt@nCnt\@ne\\immediate\openout\rt@Verb@write\jobname_.\rt@BaseName\the\rt@nCnt.cut\\let\verbatim@out\rt@Verb@write\\rt@IWVO{\string\RTVWHook}\\verbatimwrite}{% \endverbatimwrite\immediate\write\rt@Verb@write{\string\reVerbEnd}\\immediate\closeout\rt@Verb@write}}\end{verbatim}

rtVW
\@nVToks randomizes the contents of the rt\textbackslash W environment.

\@endToks is main looping command for \texttt{\textbackslash ranToks} and \texttt{\textbackslash eRTVToks} (through \texttt{\textbackslash @nVToks}). If the ending token \texttt{\textbackslash rt@NIL} is detected, we break off and go to \texttt{\textbackslash @endToks}.

\texttt{\textbackslash @performRanDefns\{\textbackslash name\}} The \texttt{\textbackslash @performRanDefns} performs code that is repeated in several other macros: \texttt{\textbackslash @endToks}, \texttt{\textbackslash reorderRanToks}, and \texttt{\textbackslash copyRanToks}. It randomizes the list \texttt{\textbackslash rt@RandomizeList}, then assigns the randomized list to the definitions.

\texttt{\textbackslash @performRanDefns\{\textbackslash name\}} The \texttt{\textbackslash performRanDefns} performs code that is repeated in several other macros: \texttt{\textbackslash @endToks}, \texttt{\textbackslash reorderRanToks}, and \texttt{\textbackslash copyRanToks}. It randomizes the list \texttt{\textbackslash rt@RandomizeList}, then assigns the randomized list to the definitions.
Good to go. We reorder this list.

\begin{verbatim}
\rt@performRanDefs{\@nameuse{nMax4#1}}\fi
\endgroup}
\copyRanToks{(name1)}{(name2)} Use this command to copy \(name1\) to \(name2\). This gives a randomization of the same list, without affecting the original order of \(name1\).

\newcommand{\copyRanToks}{\begingroup\expandafter
\ifx\csname nMax4#1\endcsname\relax
Source list is not defined
\rt@badTokName{#1}\
\else\expandafter
Source list is defined
\fi}
\iffalse\csname nMax4#1\endcsname\relax
\else\expandafter\fi\endgroup
\useRTName{#2}\global\rt@nameedef{nMax4#2}{\@nameuse{nMax4#1}}\rt@nCnt=\csname nMax4#2\endcsname\relax
\@whilenum\rt@nCnt>0\do{\global
\rt@nameedef{rtTok\the\rt@nCnt#2}{{\noexpand\@nameuse{rtTok\the\rt@nCnt#1}}}\advance\rt@nCnt\m@ne}
\rt@performRanDefs{\@nameuse{nMax4#2}}\else
Destination list is defined already, warn the user.
\rt@warnTokName{#2}\fi
\fi}
\iffalse\csname nMax4#2\endcsname\relax
\else\expandafter\fi\endgroup
\useRanTok{#2}\global\rt@nameedef{rtRanTok\the\rt@nCnt\rt@BaseName}{\noexpand\@nameuse{rtTok#1\rt@BaseName}}}

4.1 Additional user access commands
\nToksFor{(name)} expands the number of tokens whose name is \(name\) (#1).
\newcommand{\nToksFor}[1]{\expandafter
\ifx\csname nMax4#1\endcsname\relax
\textbf{??}\rt@badTokName{#1}\
\else\expandafter\fi}
\iffalse\csname nMax4#1\endcsname\relax
\else\expandafter\fi\endgroup
\def{\rt@ssign{#1}}{\advance\rt@nCnt\m@ne\global
\rt@nameedef{rtRanTok\the\rt@Cnt\rt@BaseName}{\noexpand\@nameuse{rtTok\the\rt@Cnt\rt@BaseName}}}

\rtTokByNum{(name)}{(num)} is an internal macro, but it can be used publicly. The argument of it is an integer, eg, \rtTokByNum{3} is the third token, as listed in the order given in the argument of \ranToks.
\useRanTok\langle name\rangle\{\langle num\rangle\} After \texttt{\runToks} has been executed, the user has access to the randomized tokens through \useRanTok. The argument \langle num\rangle is an integer 1 through max.

\useRanTok\langle prior\rangle\{\langle post\rangle\}\{\langle name\rangle\} lists all items in the list as passed by the required argument. For expanding in a list environment, use \item as the optional argument. Designed for listing all question in an exam document in random order.

\displayListRandomly\langle prior\rangle\{\langle post\rangle\}\{\langle name\rangle\} lists all items in the list as passed by the required argument. For expanding in a list environment, use \item as the optional argument. Designed for listing all question in an exam document in random order.

5 Commands that support a DB application

We begin with some utility commands to help parse the argument of \useProbDBs.

\def\rt@gettonil#1\@nil{\def\to@nilarg{#1}}
\def\rt@ifspc{\ifx\@let@token\@sptoken\let\rt@next\rt@xifspc\else\let\rt@next\rt@gettonil\fi\rt@next}
\begingroup
\def\:{\rt@xifspc}

\begin{enumerate}
\item Within the optional arguments, we define \i, \first, \last, and \lesson to do some logic on the arguments. These four macro are defined locally and not available outside the command \displayListRandomly.
\item \displayListRandomly\langle prior\rangle\{\langle post\rangle\}\{\langle name\rangle\} lists all items in the list as passed by the required argument. For expanding in a list environment, use \item as the optional argument. Designed for listing all question in an exam document in random order.
\end{enumerate}
\useTheseDBs\{\textit{list}\} Inputs any files included in the comma-delimited list. The base names need only be listed, as the extension is assumed to be .tex. The command \useProbDBs can only be used in the preamble. Refer to the demo file mc_db.tex for an illustration of its intended use.

\begin{verbatim}
273 \expandafter\gdef\csname useTheseDBs\endcsname: {\futurelet\@let@token\rt@ifspc}
274 \endgroup
275 \def\rt@strppcs{\futurelet\@let@token\rt@ifspc}

\useTheseDBs\{\textit{list}\}
\end{verbatim}
6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

| Symbols | |
|---------|---------|---------|---------|---------|
| \%      | 87      | \i      | 10      |
| :       | 272, 273| \IfFileExists | 53          |
| @bsphack| 71      | \ifhmode | 176       |
| @}let@token | 268, 273, 275 | \ifrtdebug | 31, 105, 107, 119, 127, 132, 198 |
| @}makeother | 72      | \ifsaveseed | 33, 81    |
| @}onlypreamble | 49, 51  | \ifverandomize | 32, 99     |
| @}Orgi | 177     | \InitSeedValue | 12, 57, 61, 80, 88 |
| @}sptoken | 268     | \input | 3, 201    |
| ^       | 73      | \InputIfFileExists | 286      |
| \active | 73      | \inputlineno | 142, 151, 160 |
| @AtEndDocument | 91  | \inputRandomSeed | 69      |
| \bRTVToks | 154, 172| \begin{verbatimwrite} | 78, 187 |
| \copyRanToks | 220     | \endgroup | 252, 266 |
| \CurrentOption | 41      | \endverbatimwrite | 78, 187 |
| \day | 8       | \endverbatimwrite | 78, 187 |
| \DeclareOption | 41      | \message | 70, 71, 221 |
| \displayListRandomly | 254, 255| \message | 70, 71, 221 |
| \displayListRandomly | 10, 253| \message | 70, 71, 221 |
| \divide | 15, 136| \message | 70, 71, 221 |
| \dospecials | 72      | \message | 70, 71, 221 |
| \egroup | 252, 266| \message | 70, 71, 221 |
| \endverbatimwrite | 78, 187| \message | 70, 71, 221 |
| \message | rtVW | 176 |
| \message | 155, 173| \message | 70, 71, 221 |
| \filename@area | 278     | \message | 70, 71, 221 |
| \filename@base | 278     | \message | 70, 71, 221 |
| \filename@ext | 278, 279| \message | 70, 71, 221 |
| \filename@parse | 276     | \message | 70, 71, 221 |
| \message | first | 10, 260 |
| \message | 273, 275| \message | 70, 71, 221 |

| I          | |
|------------|---------|---------|---------|---------|
| \IfFileExists | 53      | \input | 3, 201  |
| \inputRandomSeed | 69      | \inputlineno | 142, 151, 160 |
| \last | 10, 260, 261, 263 |
| \lastRandomNum | 58, 60  |
| \lessone | 10, 262 |
| \message | 111, 136, 231, 261 |
| \message | 9 |
| \message | 7–9, 17, 19, 20, 137 |
| \message | 38 |
| \message | 34, 82 |
| \message | 4, 11 |
| \message | 9, 143, 239, 250 |
| \message | 56 |
| \message | 83, 181 |
| \message | 8, 168, 203, 204, 208 |
| \message | 54, 63 |
| \message | 277 |
| \message | 140, 150, 159 |
| \message | 41 |
| \message | 276, 287 |
| \message | 42 |
| \message | 30 |
| \message | 8, 175, 191, 199 |

12
7 Change History

v1.0b (2013/07/29)
General: Added \displayListRandomly ........ 10

v1.0c (2013/08/03)
General: Save the initial seed value to \rtInitSeedValue .................. 2

v1.0d (2013/08/03)
General: Added conditional input of random.tex . 2

v1.0e (2016/02/06)
General: Added optional argument to \displayListRandomly ............... 10

v1.1 (2017/05/04)
General: Fixed a bug, when the first two tokens #1 are the same, we get an incorrect decision ... 8

v1.2 (2019/12/28)
General: Added dummy package ran-toks ....... 3
rtVW: Defined \rtVWHook ................. 7