Splitting Long Sequences of Letters (DNA, RNA, Proteins, Etc.)*

Boris Veytsman

2006/08/07, v0.1

Abstract

Sometimes one needs to typeset long sentences of letters, which should not have spaces between them (like letters in words), but could be split between lines at any point, and without a hyphenation character. This package provides a command for such sequences.

Contents

1 Introduction 2
2 User Interface 2
  2.1 Main Command ........................................... 2
  2.2 Customization ............................................. 3
  2.3 Grouping and Commands ................................. 4
3 Implementation 6
  3.1 Declarations ............................................. 6
  3.2 Inserted Text ............................................. 6
  3.3 Scanner .................................................. 6
  3.4 The Last Words .......................................... 6

*©Boris Veytsman, 2006
1 Introduction

At a recent Practical\TeX conference (Practical\TeX-2006, Rutgers, New Jersey, USA, \url{http://www.tug.org/practicaltex2006}) Klaus Höppner asked, how one typesets long sequences like the ones related to DNA code. Usually there is no space between letters, but a sequence could be split at any point and continued on the next line. The audience suggested several solutions to this problem. One solution, for example, was to define a new language, where hyphenation is possible at any point, and hyphenation character is empty. However, this would require regeneration of all \TeX formats, which might be not practical or even not possible. Another solution, suggested, if my memory is right, by Peter Flynn, was to scan the sequence and insert a breaking point after each letter. This later approach is implemented in this package.

2 User Interface

2.1 Main Command

The main (and actually the only) command in this package is \texttt{\seqsplit}. Its usage is very simple, for example to typeset the gene HBB, related to sickle cell anaemia (actually, the corresponding mRNA Reference Sequence), we use the following:

\begin{verbatim}
\seqsplit{acatttgtctctgacacaactgtgttcactagcaacctcaaacagacaccatggtgcatctgactc
ttgaggagaagtctgccgttactgccctgtggggcaaggtgaacgtggatgaagttggtggtgag
ccctgggcaggctgctggtggtctacccttggacccagaggttctttgagtcctttggggatctg
tccactcctgatgctgttatgggcaaccctaaggtgaaggctcatggcaagaaagtgctcggtgc
ctagtgatggcctggctcacctggacaacctcaaggtccttctgctgtctcttctcttgcccac
tccatcactaactaaactgpggatattatgagggccttgagcatctggattctgccc
taatataaaaacacttttttcccgc}.
\end{verbatim}

which produces

\begin{verbatim}
acatttgtctctgacacaactgtgttcactagcaacctcaaacagacaccatggtgcatctgactc
ttgaggagaagtctgccgttactgccctgtggggcaaggtgaacgtggatgaagttggtggtgag
ccctgggcaggctgctggtggtctacccttggacccagaggttctttgagtcctttggggatctg
tccactcctgatgctgttatgggcaaccctaaggtgaaggctcatggcaagaaagtgctcggtgc
cattcactaactaaactgpggatattatgagggccttgagcatctggattctgccc
\end{verbatim}

Note that the breaking points in the code (commented out by \%) have nothing to do with the breaking points in the typeset sequence and are introduced only for readability of the code.
The corresponding protein sequence (β-globulin) is shorter:

\seqsplit{
mvhlteksavtalwkg
\}nvdevggealgrll

vypwtfqreffsgd1stpdavmpkpzk

vkahgkkvlgafsdqlahdnlkgtfatlsehelckdhvdp

frilgtnvvlcvlahhfkg

keftppvqaaykvvvag

analahykhj

mvhlteksavtalwkg

\}nvdevggealgrll

vypwtfqreffsgd1stpdavmpkpvkva

hkgkkvlgafsdqlahdnlkgtfatlsehelckdhvdp

frilgtnvvlcvlahhfkeftppvqaaykvvvag

analahykhj

The command works in math mode as well:

$\pi = \\seqsplit{\pi = 3.1415926535897932384626433832795028841971693993751058209
749445923078164062862089862803482534211767082114808651328230
664709384460560582231725359408128481117450284102701938521105
559646229485493083196442881097565933416287456482337867831
65271201909146548692364634610454326648213393607260249141273
372458700660315588174881520902962829254097021798
9064307027705392171729317675284674818467669405132
0005681271452635606877857713427577896091736717872
14684409012249534301465495853710507922796892589235\ldots}$

\\seqinsert

2.2 Customization

The command \seqsplit can be customized by redefining the command \seqinsert, which is the macro that is inserted between the letters of the sequence. By default it is defined as \allowbreak in math mode and \hspace{0pt plus 0.02em} in text mode: a slightly stretchable glue of zero length. This definition gives \TeX a chance to justify the lines. However, there might be other definitions. For example, if we want hyphenations at the breakpoints in text mode, we can use:
which produces for the $\beta$-globulin protein from the previous section the following:

mvhltpeeksavtalwgvkvnvedvgealglllvyypwtqrfessfdlsptdavmgnpkvkahgkkvlgafsdlahldnlkgtfatselfslhcddklhvdpenfrllgnvlvcvlahhfgkeftppvqaayqkvvagvanalhkyh.

Another redefinition,

mvhltpeeksavtalwgvkvnvedvgealglllvyypwtqrfessfdlsptdavmgnpkvkahgkkvlgafsdlahldnlkgtfatselfslhcddklhvdpenfrllgnvlvcvlahhfgkeftppvqaayqkvvagvanalhkyh.

2.3 Grouping and Commands

The command \texttt{\seqsplit} does not insert breakpoints between the letters inside braces \{\ldots\}. Compare the typesetting of $\beta$-globulin in Section 2.1 and the following example:

mvhltpeeksavtalwgvkvnvedvgealglllvyypwtqrfessfdlsptdavmgnpkvkahgkkvlgafsdlahldnlkgtfatselfslhcddklhvdpenfrllgnvlvcvlahhfgkeftppvqaayqkvvagvanalhkyh.

The braces around \{kahg\} prevented a splitting of this group. This effect can be used for typesetting special substrings inside sequences.

The way \texttt{\seqsplit} works interferes with formatting commands like \texttt{\textit{}}. Therefore the sequence \{kahg\} is not italicized in the following example:

mvhltpeeksavtalwgvkvnvedvgealglllvyypwtqrfessfdlsptdavmgnpkv\textit{kahg}kkvlgafsdlahldnlkgtfatselfslhcddklhvdpenfrllgnvlvcvlahhfgkeftppvqaayqkvvagvanalhkyh.

Using grouping \{\texttt{\textit{kahg}}\} we can save the situation:

mvhltpeeksavtalwgvkvnvedvgealglllvyypwtqrfessfdlsptdavmgnpkv\texttt{\textit{kahg}}kkvlgafsdlahldnlkgtfatselfslhcddklhvdpenfrllgnvlvcvlahhfgkeftppvqaayqkvvagvanalhkyh.
If we want the italicized sequence to be splittable as well, we can use \textit{nested} \seqsplit:

\seqsplit{mvhltpeeksavtalwgkvnvdevgealgrllvvypwtqrfesfgdlstpdavmgnpkv kahgkkvlga\textit{sd}glahdlntkgtfatlshcdklhvfdpenfrllgnvlcvlahhfgeftppvqaayqkvvagvanalahkyh}.

These tricks allow one to produce splittable sequences with a rather complex formatting.
3 Implementation

3.1 Declarations

We start with declaration, who we are:

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{seqsplit} \[2006/08/07 v0.1 Splitting long sequences (DNA, RNA, proteins, etc.) \]

3.2 Inserted Text

\seqinsert This is the macro we insert between letters:

\def\seqinsert{\ifmmode\allowbreak\else\hspace{0pt plus 0.02em}\fi}

3.3 Scanner

The scanner code is not too trivial. Here we describe it in detail.

\seqsplit The main (actually, the only) user-space macro just starts the scanner.

\def\seqsplit#1{\SQSPL@scan#1\SQSPL@end}

The macro \SQSPL@end is never expanded, it is just a marker.

\SQSPL@scan The macro \SQSPL@scan saves the next token in the special register \SQSPL@next, so we can decide what to do with it:

\def\SQSPL@scan\futurelet\SQSPL@next\SQSPL@scani

Now since we know the next token, we can decide to either stop the expansion if we met the end, or continue it if we did not.

\def\SQSPL@scani#1{\ifx\SQSPL@end\SQSPL@next\def\SQSPL@process{\@gobble}\else\def\SQSPL@process{\SQSPL@doprocess}\fi\SQSPL@process{#1}}

\SQSPL@doprocess The processing of a letter depends on what is the next letter. If the sequence is finished, we should not insert anything after the last letter: we do not want to break the line between the sequence and, say, a comma. Therefore we insert a special smart macro:

\def\SQSPL@doprocess#1{#1\SQSPL@insert}

\SQSPL@insert The macro \SQSPL@insert uses \futurelet to check whether the processed letter is the last one in the sentence:

\def\SQSPL@insert\futurelet\SQSPL@next\SQSPL@doinset

\SQSPL@doinset And this is the macro that inserts \seqinsert and continues scanning:

\def\SQSPL@doinset{\ifx\SQSPL@end\SQSPL@next\relax\else\seqinsert\fi\SQSPL@scan}

3.4 The Last Words

\[//style\]
Change History

v0.1
  General: The first released version  1
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.

<table>
<thead>
<tr>
<th>S</th>
<th>\SQSPL@doprocess</th>
<th>\SQSPL@process</th>
<th>\SQSPL@end</th>
<th>\SQSPL@insert</th>
<th>\SQSPL@process</th>
<th>\SQSPL@scani</th>
<th>\SQSPL@scan</th>
<th>\SQSPL@doinsert</th>
<th>\SQSPL@doinsert</th>
<th>\SQSPL@process</th>
<th>\SQSPL@doinsert</th>
</tr>
</thead>
<tbody>
<tr>
<td>\seqinsert</td>
<td>3, 5, 16</td>
<td>\SQSPL@process</td>
<td>6, 9, 15</td>
<td>\SQSPL@insert</td>
<td>12, 13</td>
<td>\SQSPL@end</td>
<td>\SQSPL@scani</td>
<td>6, 7, 17</td>
<td>\SQSPL@doinsert</td>
<td>13, 14</td>
<td>\SQSPL@next</td>
</tr>
<tr>
<td>\seqsplit</td>
<td>2, 6</td>
<td>\SQSPL@doinsert</td>
<td>12, 13</td>
<td>\SQSPL@doinsert</td>
<td>13, 14</td>
<td>\SQSPL@next</td>
<td>7, 9, 13, 15</td>
<td>\SQSPL@scani</td>
<td>7, 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>