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

At a recent PracticalTEX conference (PracticalTEX-2006, Rutgers, New Jersey, USA, `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

`\seqsplit` The main (and actually the only) command in this package is `\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:

```
\seqsplit{%
acatttgcttctgacacaactgtgttcactagcaacctcaaacagacaccatggtgcatc%
tgactcctgaggagaagtctgccgttactgccctgtggggcaaggtgaacgtggatgaag%
ttggtggtgaggccctgggcaggctgctggtggtctacccttggacccagaggttctttg%
agtcctttggggatctgtccactcctgatgctgttatgggcaaccctaaggtgaaggctc%
atggcaagaaagtgctcggtgcctttagtgatggcctggctcacctggacaacctcaagg%
gcacctttgccacactgagtgagctgcactgtgacaagctgcacgtggatcctgagaact%
tcaggctcctgggcaacgtgctggtctgtgtgctggcccatcactttggcaaagaattca%
ccccaccagtgcaggctgcctatcagaaagtggtggctggtgtggctaatgccctggccc%
acaagtatcactaagctcgctttcttgctgtccaatttctattaaaggttcctttgttcc%
ctaagtccaactactaaactgggggatattatgaagggccttgagcatctggattctgcc%
taataaaaaacatttatttcattgc}.
```

which produces

acatttgcttctgacacaactgtgttcactagcaacctcaaacagacaccatggtgcatctgactcc
tgaggagaagtctgccgttactgccctgtggggcaaggtgaacgtggatgaagttggtggtgagg
ccctgggcaggctgctggtggtctacccttggacccagaggttctttgagtcctttggggatctgtc
cactcctgatgctgttatgggcaaccctaaggtgaaggctcatggcaagaaagtgctcggtgcctt
tagtgatggcctggctcacctggacaacctcaagggcacctttgccacactgagtgagctgcactg
tgacaagctgcacgtggatcctgagaacttcaggctcctgggcaacgtgctggtctgtgtgctggc
ccatcactttggcaaagaattcaccccaccagtgcaggctgcctatcagaaagtggtggctggtgt
ggctaatgccctggcccacaagtatcactaagctcgctttcttgctgtccaatttctattaaaggtt
cctttgttccctaagtccaactactaaactgggggatattatgaagggccttgagcatctggattct
gcctaataaaaaacatttatttcattgc.

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{%
mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpk%
vkahgkkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfg%
keftppvqaayqkvvagvanalahkyh}.
```

mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkvka
hgkkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvqaa
yqkvvagvanalahkyh.

The command works in math mode as well:

```
$\pi = \seqsplit{%
3.
1415926535 8979323846 2643383279 5028841971 6939937510
5820974944 5923078164 0628620899 8628034825 3421170679
8214808651 3282306647 0938446095 5058223172 5359408128
4811174502 8410270193 8521105559 6446229489 5493038196
4428810975 6659334461 2847564823 3786783165 2712019091
4564856692 3460348610 4543266482 1339360726 0249141273
7245870066 0631558817 4881520920 9628292540 9171536436
7892590360 0113305305 4882046652 1384146951 9415116094
3305727036 5759591953 0921861173 8193261179 3105118548
0744623799 6274956735 1885752724 8912279381 8301194912
9833673362 4406566430 8602139494 6395224737 1907021798
6094370277 0539217176 2931767523 8467481846 7669405132
0005681271 4526356082 7785771342 7577896091 7363717872
1468440901 2249534301 4654958537 1050792279 6892589235}
\ldots$
```

$\pi = 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892350 . . .

2.2 Customization

`\seqinsert` 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 hyphens at the breakpoints in text mode, we can use:

```
\renewcommand{\seqinsert}{\ifmmode\allowbreak\else\-\fi}
```

which produces for the β-globulin protein from the previous section the following:

> mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkvka-
> hgkkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvqaa-
> yqkvvagvanalahkyh.

Another redefinition,

```
\renewcommand{\seqinsert}{\ifmmode\allowbreak\else{} \fi},
```

produces an output with spaces between letters. Note that there is no space between the last letter and the dot: the package takes care of this:

> m v h l t p e e k s a v t a l w g k v n v d e v g g e a l g r l l v v y p w
> t q r f f e s f g d l s t p d a v m g n p k v k a h g k k v l g a f s d g l a
> h l d n l k g t f a t l s e l h c d k l h v d p e n f r l l g n v l v c v l a h
> h f g k e f t p p v q a a y q k v v a g v a n a l a h k y h.

2.3 Grouping and Commands

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

```
\seqsplit{%
mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpk%
v{kahg}kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfg%
keftppvqaayqkvvagvanalahkyh}.
```

> mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkv
> kahgkkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvq
> aayqkvvagvanalahkyh.

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

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

```
\seqsplit{%
mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpk%
v\textit{kahg}kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvl%
vcvlahhfgkeftppvqaayqkvvagvanalahkyh}.
```

> mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkv
> kahgkkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvq
> aayqkvvagvanalahkyh.

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

```
\seqsplit{%
mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpk%
v{\textit{kahg}}kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvl%
vcvlahhfgkeftppvqaayqkvvagvanalahkyh}.
```

mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkv*kahg*kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvq aayqkvvagvanalahkyh.

If we want the italicized sequence to be splittable as well, we can use *nested* `\seqsplit`:

```
\seqsplit{%
mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpk%
v{\textit{\seqsplit{kahg}}}kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvl%
vcvlahhfgkeftppvqaayqkvvagvanalahkyh}.
```

mvhltpeeksavtalwgkvnvdevggealgrllvvypwtqrffesfgdlstpdavmgnpkv*ka hg*kkvlgafsdglahldnlkgtfatlselhcdklhvdpenfrllgnvlvcvlahhfgkeftppvqaa yqkvvagvanalahkyh.

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:

```
⟨*style⟩
\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}
```

`\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@doinsert}
```

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

```
\def\SQSPL@doinsert{%
  \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.