siunitx: A comprehensive (SI) units package

Joseph Wright

Abstract

The siunitx package provides a powerful toolkit for typesetting numbers and units in LaTeX. By incorporating detail about the agreed rules for presenting scientific data, siunitx enables authors to concentrate on the meaning of their input and leave the package to deal with the formatting. Version 2 of siunitx increases the scope of the package and is intended to act as a solid base for further development.

1 Introduction

Physical quantities are important mathematical data, which appear in print in many scientific publications. These physical quantities are made up of a number and an associated unit: the two parts together make up a single mathematical entity. A series of international agreements have led to the *Système International d'Unités* (SI units), administered by the *Bureau International des Poids et Mesures* (Bureau International des Poids et Mesures, 2010). This system lays down units such as the kilogram, metre and kelvin, and provides a self-consistent approach to measuring all physical quantities. At the same time, there are clearly defined standards which describe how the data should be presented. The US National Institute for Standards and Technology (NIST) have described 'best practices' for presenting numbers and units in scientific contexts (National Institute for Standards and Technology, 2010).

LaTeX's approach of providing logical structure is ideally suited to helping authors follow these rules without needing to study them in detail. However, this does not mean that it has been easy to write a comprehensive package to deal with both numbers and units. This article provides an overview of the siunitx package (Wright, 2010), which aims to be *A comprehensive (SI) units package.*

2 A little history

2.1 Before siunitx

siunitx is the latest in a series of different LaTeX packages for handling units, and it is therefore useful to know a little about the earlier implementations.

The package units (Reichert, 1998) provides perhaps the most basic units support: the macro `\unit` to mark up input as a unit (with an optional value).

`\unit[`⟨*value*⟩`]{`⟨*unit*⟩`}`

Building on this, the unitsdef package (Happel, 2005) provides a number of pre-defined unit macros, which expand to the appropriate symbol(s) for the unit. Unfortunately, these definitions require additional macros so that the package can format them correctly:

```
\newunit{\newtonmeter}
  {\newton\unittimes\meter}
\newunit{\newtonmeterpersec}
  {\per{\newton\unittimes\meter}{\second}}
```

As we will see, siunitx is able to define similar unit macros but without needing support functions such as `\unittimes`.

An alternative approach to defining unit macros was provided by SIunits (Heldoorn and Wright, 2007). SIunits provides a larger set of pre-defined units than unitsdef, but again requires support functions in these definitions. These support functions define the appearance of units, so altering how a unit is displayed requires a new definition. For example, `\amperepersquaremetre` prints $\mathrm{A/m^2}$ while `\amperepersquaremetrenp` is used for $\mathrm{A\,m^{-2}}$.

The SIstyle package (Els, 2008) tackles the need for units and values to be typeset using the correct fonts. As such, it focusses on the appearance of the output, rather than logical markup of input. This can have advantages, as input such as

```
\SI{10}{m/s}
```

is certainly easy to read.

Finally, while not focussed on units, the numprint package (Harders, 2008) has provided the complementary ability to format numbers, for example separating large numbers so that the digits are grouped.

2.2 A new approach to units

Before the development of siunitx, the best approach to typesetting units was to use the combination of SIunits and SIstyle, to get logical input and controlled output.

Development of siunitx began with a simple bug report for SIunits on `comp.text.tex`. I naïvely volunteered to take a look at it, and contacted Marcel Heldoorn with a solution to the issue at hand. It turned out that he no longer had time for supporting SIunits, and said that I was free to take over. Having fixed the bug at hand, I even more naïvely asked on the newsgroup if there were any improvements to be made. I soon had quite a list!

I took a step back, and looked at the combination of SIunits and SIstyle and the feature request list I'd built up. It was clear that I needed to do more than simply patch SIunits. I also took a careful look at biblatex (Lehman, 2010), which shows how a user interface should be done. My conclusion was that I needed to write a package from the ground up, combining the features of SIunits and SIstyle with

a key–value interface rather than a complex mix of different control macros.

This effort led to the first version of siunitx, which inherited a great deal of code from its predecessors. The feature requests kept coming, and some of these were rather 'bolted on' to the first version of siunitx. Partly as a result of this, and partly as I'm now involved in the LaTeX3 Project, I decided to rewrite the package using the expl3 approach (LaTeX3 Project, 2010). This has allowed the internal code of the package to be made much more structured, which will hopefully enable me to continue to add new features without compromising the existing features of the package.

3 Units

The core function of the siunitx package is typesetting units in a flexible way and with a natural input syntax. The macro which does this is `\si` (think 'a bit like "SI"'). The `\si` macro can typeset both literal input such as `\si{m.s^{-1}}` and the semantic version `\si{\metre\per\second}` to give the same output: $\mathrm{m\,s^{-1}}$. Allowing two forms of input means that users get to make a choice on how semantic they want their input to be.

There are lots of things going on when something like `\si{m.s^{-1}}` is typeset. The first thing to notice is that the superscript will work equally-happily in math and text mode (the same is true for subscripts). What is also true is that you get *exactly the same* output in both cases: the fonts and spacing used are determined by siunitx. The standard settings use the document text font for units, but the document math font for numbers. Numbers as handled by siunitx are essentially mathematical, and so they should (probably) match any other mathematics. Both numbers and units are typeset ignoring any local font changes, such as bold or italics.

Now, some of those choices will not suit everyone: a classic case is units in section headings, where bold seems a more 'natural' choice than the usual mid-weight font. To handle the need to be flexible, siunitx provides the `\sisetup` macro, which takes a list of key–value options (there are currently about 140!). Settings can also be given as an optional argument to `\si`, which allows them to be applied to individual cases: `\sisetup` applies to everything that follows, subject to the usual TeX grouping. So in a heading, rather than `\si{m.s^{-1}}` we might have `\si[detect-weight]{m.s^{-1}}`.

What about the unit macros: are they flexible? One of the key aims of siunitx is to use semantic markup with units so that different output appearances don't need different input syntaxes. Sticking with the example `\si{\metre\per\second}`, there are a number of possible variations. As we've already seen, the standard settings give '$\mathrm{m\,s^{-1}}$', with the `\per` macro converted into a superscripted power. Another common choice is 'm/s', using a slash to show the division. That's done by setting the option `per-mode = symbol`. Then again, you might want to show things as a fraction, '$\frac{\mathrm{m}}{\mathrm{s}}$', achieved by setting `per-mode = fraction`.

That is fine for a relatively simple unit, but what about a more complex case such as

```
\si{\joule\per\mole\squared
  \metre\per\cubic\candela}
```

(*i.e.* $\mathrm{J\,mol^{-2}\,m\,cd^{-3}}$)? When given as a fraction or using a slash, there need to be some brackets or rearrangement of the order. The package knows about this, and can automatically produce the appropriate output, which might be '$\mathrm{J\,m/(mol^2\,cd^3)}$' or '$\frac{\mathrm{J\,m}}{\mathrm{mol^2\,cd^3}}$'. It can even produce mathematically-invalid output like '$\mathrm{J/mol^2\,m/cd^3}$' if you want.

As already indicated, there are a *lot* of options available, and I don't want to repeat the manual here. However, I hope that the concept of 'one clear input, many forms of output' comes through.

One last idea to highlight is that new units can be defined using the two macros `\DeclareSIUnit` and `\DeclareSIUnitWithOptions`. These are used to set up new units, perhaps with a special appearance. So if I want to give '$\frac{\mathrm{m}}{\mathrm{s}}$' with a slash but everything else as powers, I might define

```
\DeclareSIUnitWithOptions{\mpers}
  {\metre\per\second}{per-mode = fraction}
```

and then use `\mpers` as the unit. Name clashes are not an issue: siunitx only defines the unit macros within the argument of its own macros.

4 Numbers

Most of the time, units in scientific writing go with numbers. So siunitx needs to be able to deal with those as well. This is handled by the imaginatively-named `\num` macro. This takes the number itself as the one mandatory argument, with a first optional argument for any settings that apply.

Probably the most common function this performs is grouping digits. So `\num{12345}` will give '12 345' rather than '12345'. The latter is of course available as an option: `group-digits = false`.

There are two other common formatting changes. First, it is useful to format `\num{12e3}` as '12×10^3', which is done automatically. Secondly, depending on where in the world you are you might want `\num{123.45}` to display as '123,45'. The package uses settings such as `input-exponent-markers` and

`output-decimal-marker` to decide on the format of the input and how it should look as output for these cases.

Another common requirement with numbers is to round them, fixing either decimal places or significant figures. Here, the two options `round-mode` and `round-precision` are used. The standard settings do not do any rounding at all, so `\num{123.456}` gives '123.456'. This can easily be converted into '123.46' by setting `round-mode = places`, or '120' by setting `round-mode = figures`. As you might work out, the standard setting is `round-precision = 2`, and this applies to whatever rounding is being done. As we'll see, rounding is particularly useful in tables.

5 Numbers with units

We've seen both numbers and units on their own, but obviously the two need to be combined. For that, the `\SI` macro is available, and takes one number and one mandatory unit argument to print the combination of the two. As with `\num` and `\si`, the first argument is optional and applies local settings.

All of the options for units and numbers alone apply to combinations too, but there are some special options which only make sense for combinations. The most obvious is a choice of the separator between a number and the associated unit. The standard setting is thin space: '10 m'. This is controlled by the `number-unit-separator` option, which expects an argument in math mode. So to use a full text-mode space you'd set `number-unit-separator = \text{ }`, with the result '10 m'.

Closely related to the *size* of the space between number and unit is how it behaves at a line break. The standard settings do not allow a break here, but particularly in narrow columns (such as in this document) it is more realistic to allow breaks to occur. The option to do control is called `allow-number-unit-breaks`, which will allow a break: '10 m'. (As you might guess, the text in this paragraph is finely balanced to give a break in the right place!).

6 Tables

Placing numbers in tables so that the decimal markers are aligned is very important so that scientific data are clear. To support this, siunitx defined the `S` column type. At its most basic, all you do is use this in place of a `c` column and let the package do the work. So with the input

```
\begin{tabular}{S}
  \toprule
  {Some numbers} \\
  \midrule
    1.234e2 \\
  567.8e0   \\
    4.3543e1 \\
  \bottomrule
\end{tabular}
```

Table 1: Simple number alignment using the `S` column

Some numbers
1.234×10^2
567.8
4.3543×10^1

you can get the output in Table 1. Notice that the table heading is wrapped in braces: this tells siunitx to treat this as text and not to look for a number.

Now, Table 1 is not a very good table, as the data are not really comparable. It's usually best to avoid exponents in the body of a table, and to put them into the header instead. It's also common to round tabular data to some sensible number of significant figures. Table 2 is a better version of the same table, with input that reads

```
\begin{tabular}{S[
    table-auto-round,
    table-omit-exponent,
    table-format   = 1.2,
    fixed-exponent = 2
  ]}
  \toprule
  {Some numbers/\num{e2}} \\
  \midrule
    1.234e2 \\
  567.8e0   \\
    4.3543e1 \\
  \bottomrule
\end{tabular}
```

Table 2: Exploiting the power of the `S` column

Some numbers/10^2
1.23
5.68
0.44

This illustrates several table-related functions in one go. First, the `S` column accepts an optional argument, letting each column have its own behaviour. The option `table-format` is used to define how much space siunitx will need for the output: here there is one integer and two decimal digits, with no signs or exponents. The `table-auto-round` and `table-omit-exponent` options have self-explanatory

names, while `fixed-exponent = 2` is used to 'shift' the decimal place in the input. This combination of options means that the input does not require any manipulation: an advantage if it's a long list copied from some automated source!

7 Extras

We've seen the main macros that siunitx provides, but there are a few more specialist ones which deal with more unusual cases. These 'extras' all take the usual optional first argument, and have their own dedicated options.

The `\ang` macro takes angle input, either as a decimal or in degrees, minutes and seconds. The latter is necessary for things like '1°2′3″', which is given as `\ang{1;2;3}`. One particularly notable option here is `angle-symbol-over-decimal`, which can give output such as '1°2′3.″4' from the input

```
\ang[angle-symbol-over-decimal]{1;2;3.4}
```

I'm told that this is useful for astronomy: that is far from my area of expertise, but as always the aim is to give users what they want with the minimum of fuss.

There are two specialist macros for cases where the same unit applies to multiple numbers: `\SIrange` and `\SIlist`. These let you type

```
\SIrange{10}{20}{\metre}
```

and get '10 m to 20 m', or to type

```
\SIlist{1;2;3;4;5}{\metre}
```

and get '1 m, 2 m, 3 m, 4 m and 5 m'. You'll notice that the standard settings repeat the unit for each number. Not everyone will like that, so you can use

```
\SIlist[list-units = brackets]
  {1;2;3;4;5}{\metre}
```

and get '(1, 2, 3, 4 and 5) m', or even

```
\SIlist[list-units = single]
  {1;2;3;4;5}{\metre}
```

to give the (mathematically incorrect) '1, 2, 3, 4 and 5 m'.

8 Summary

The siunitx package aims to be 'a comprehensive (SI) units package' while remaining accessible to users. It supplies a small number of flexible macros along with a large set of key–value options to control output either globally or for individual cases.

Here, I've tried to highlight how siunitx works, showing off some of the powerful features it supplies. The manual contains examples for almost all of the options, and if you can't see how to do something with siunitx you can always submit a feature request!

9 Acknowledgements

Thanks to Danie Els and Marcel Heldoorn for the SIstyle and SIunits packages: siunitx would not exist without them. Thanks to Stefan Pinnow for his careful testing of siunitx: his contribution to the package has been invaluable.

References

LaTeX3 Project. "The expl3 package and LaTeX3 programming". Available from CTAN, `macros/latex/contrib/expl3`, 2010.

Bureau International des Poids et Mesures. "The International System of Units (SI)". `http://www.bipm.org/en/si/`, 2010.

Els, D. N. J. "The SIstyle package". Available from CTAN, `macros/latex/contrib/SIstyle`, 2008.

Happel, Patrick. "unitsdef – Typesetting units with LaTeX 2ε". Available from CTAN, `macros/latex/contrib/unitsdef`, 2005.

Harders, Harald. "The numprint package". Available from CTAN, `macros/latex/contrib/numprint`, 2008.

Heldoorn, Marcel, and J. A. Wright. "The SIunits package: Consistent application of SI units". Available from CTAN, `macros/latex/contrib/SIunits`, 2007.

Lehman, Philipp. "The biblatex package: Programmable Bibliographies and Citations". Available from CTAN, `macros/latex/contrib/biblatex`, 2010.

National Institute for Standards and Technology. "International System of Units from NIST". `http://physics.nist.gov/cuu/Units/index.html`, 2010.

Reichert, Axel. "units.sty – nicefrac.sty". Available from CTAN, `macros/latex/contrib/units`, 1998.

Wright, Joseph A. "siunitx – A comprehensive (SI) units package". Available from CTAN, `macros/latex/contrib/siunitx`, 2010.

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2 dot co dot uk