The **skmath** package[*][†]

Simon Sigurdhsson ⌈sigurdhsson@gmail.com⌋

Version 0.3a

Abstract The skmath package provides improved and new math commands for superior typesetting with less effort.

1 Introduction

This package intends to provide helpful (re-)definitions of commands related to typesetting mathematics, and specifically typesetting them in a more intuitive, less verbose and more beautiful way. It was originally not intended for use by the public, and as such there may be incompatibilities with other packages of which I am not aware, but I figured it could be useful to other people as well.

2 Usage

2.1 Options

As of version v0.3a, there is only one option: `commonsets`. By default, it is disabled but if the option is given the package will define `\N`, `\Z`, `\Q`, `\R` and `\C` as blackboard variants of the respective letters, to represent the common sets of numbers.

[*] Available on `http://www.ctan.org/pkg/skmath`.

[†] Development version available on `https://github.com/urdh/skmath`.

2.2 New commands

The package defines a number of new commands that aid in typesetting certain mathematical formulae.

```
\N
\Z
\Q
\R
\C
```

These commands are only available if the `commonsets` option is given. They typeset the set of natural, integer, rational, real and complex numbers respectively.

Example:

$$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}.$$

```
\begin{equation*}
  \N, \Z, \Q, \R, \C.
\end{equation*}
```

```
\norm  {⟨expression⟩}
 \abs  {⟨expression⟩}
```

The commands `\norm` and `\abs`, quite expectedly, typeset the norm ans absolute value of an expression, respectively. They have one mandatory argument (the expression), and different norms can be achieved by appending a subscript after the argument of `\norm`.

Example:

$$\left\|x\right\|_p = \left(\sum_{i=1}^{n} \left|x_i\right|^p\right)^{1/p}$$

```
\begin{equation*}
  \norm{\vec{x}}_p =
    \left(\sum_{i=1}^n \abs{x_i}^p\right)%
      ^{\sfrac{1}{p}}
\end{equation*}
```

`\d` {⟨*variable*⟩}

There is also a command `\d`, with one mandatory argument, that typesets the differential part of an integral.

Example:

$$\int_{\mathbb{R}}\! \frac{\sin\left(x\right)}{x} \,\mathrm{d}x$$

```
\begin{equation*}
  \int_{\R}\! \frac{\sin{x}}{x} \d{x}
\end{equation*}
```

`\pd` *{⟨*function*⟩}{⟨*var*⟩,⟨*var*⟩,...}

This macro typesets a partial derivative. The starred variant typesets derivatives as subscripts, i.e. f_{xxy}, while the unstarred variant typesets full fractions:

Example:

$$\frac{\partial^{m+n} f}{\partial x^m \partial y^n}$$

```
\begin{equation*}
  \pd{f}{x^m,y^n}
\end{equation*}
```

As the example shows, the comma-separated list of variables also supports superscripts to denote the number of derivatives, and the sum of the variables is automatically calculated.

`\E` {⟨*expression*⟩}

The command `\E` typesets the expectation of a random variable.

Example:

$$\mathrm{E}\left[\hat{\mu}\right] = \mu$$

```
\begin{equation*}
  \E{\hat{\mu}} = \mu
\end{equation*}
```

`\P` {⟨*expression*⟩`\given`⟨*expression*⟩}

The `\P` command typesets a probability. The `\given` command can be used to typeset conditional probabilities, within `\P`.

Example:

$$\mathrm{P}\left(A \mid B\right) = \frac{\mathrm{P}\left(B \mid A\right)\mathrm{P}\left(A\right)}{\mathrm{P}\left(B\right)}$$

```
\begin{equation*}
  \P{A\given B} =
    \frac{\P{B\given A}\P{A}}{\P{B}}
\end{equation*}
```

`\var` {⟨*expression*⟩}
`\cov` {⟨*expression*⟩}{⟨*expression*⟩}

The commands `\var` and `\cov` typeset the variance and covariance of an expression.

Example:

$$\mathrm{Var}\left(X\right) = \mathrm{E}\left[(X-\mu)^2\right]$$
$$\mathrm{Cov}\left(X, Y\right) = \mathrm{E}\left[XY\right] - \mathrm{E}\left[X\right]\mathrm{E}\left[Y\right]$$

```
\begin{gather*}
  \var{X} = \E{(X-\mu)^2}\\
  \cov{X}{Y} = \E{XY}-\E{X}\E{Y}
\end{gather*}
```

2.3 Improved commands

In addition to adding new commands, this package also redefines already existing commands in a mostly backwards-compatible way to improve their usefulness.

`\sin` [⟨*power*⟩]{⟨*expression*⟩}

`\arcsin` {⟨*expression*⟩}

`\cos` [⟨*power*⟩]{⟨*expression*⟩}

`\arccos` {⟨*expression*⟩}

`\tan` [⟨*power*⟩]{⟨*expression*⟩}

`\arctan` {⟨*expression*⟩}

`\cot` [⟨*power*⟩]{⟨*expression*⟩}

`\sinh` [⟨*power*⟩]{⟨*expression*⟩}

`\cosh` [⟨*power*⟩]{⟨*expression*⟩}

`\tanh` [⟨*power*⟩]{⟨*expression*⟩}

The trigonometric functions have been redefined to typeset more easily. They typeset ⟨*expression*⟩ as an argument of the expression, and (if applicable) ⟨*power*⟩ as a superscript between the function and its argument, *e.g.* $\sin^2(\phi)$.

`\ln` {⟨*expression*⟩}

The natural logarithm macro `\ln` has also been redefined to require an argument which is typeset as the argument of the logarithm.

`\log` [⟨*base*⟩]{⟨*expression*⟩}

The related macro `\log` is redefined in a similar way, but also accepts an optional argument denoting the base of the logarithm: $\log_2(x)$.

`\exp` *{⟨*expression*⟩}

The exponential, `\exp`, is redefined to typeset its argument as a superscript of e in some display styles, and as an argument of $\exp$ otherwise:

$$e^{\sqrt{2}\exp(x)}$$

Additionally, it is possible to force the $\exp$ mode by using the starred variant.

```
\min    *[⟨domain⟩]{⟨expression⟩}
\argmin *[⟨domain⟩]{⟨expression⟩}
\max    *[⟨domain⟩]{⟨expression⟩}
\argmax *[⟨domain⟩]{⟨expression⟩}
\sup    *[⟨domain⟩]{⟨expression⟩}
\inf    *[⟨domain⟩]{⟨expression⟩}
```

The maximum/minimum macros have been redefined in a manner similar to the trigonometric functions. They typeset ⟨*expression*⟩ inside curly brackets (the starred version omits the brackets), with the optional ⟨*domain*⟩ typeset in a subscript after the operator (*e.g.* $\min_{x\in\mathbb{R}_+} f(x)$). Argument variants are also provided, and the ⟨*expression*⟩ is centered underneath the operator if possible:

$$\operatorname*{arg\,min}_{x\in\mathbb{R}_+} f(x)$$

2.4 Stylistic changes

Some commands have been redefined in a completely backwards-compatible way to improve the end result of their typesetting.

```
\frac {⟨numerator⟩}{⟨denominator⟩}
```

The `\frac` command has been changed to improve typesetting, allowing displaystyle math in some settings.

```
\bar {⟨expression⟩}
\vec {⟨expression⟩}
```

The `\bar` command has been changed to cover the entire ⟨*expression*⟩ (*i.e.* $\overline{uv}$), and `\vec` has been changed to match the `\vectorsym` command provided by isomath.

3 Known issues

A list of current issues is available in the Github repository of this package[1], but as of the release of v0.3a, there are no known issues.

If you discover any bugs in this package, please report them to the issue tracker in the skmath Github repository.

4 Implementation

The package implementation is very simple. First, we do the standard LaTeX 2_ε preamble thing, then we require some dependencies.

⟨package⟩
```
\RequirePackage{expl3,l3keys2e,xparse}
\ProvidesExplPackage{skmath}
    {2013/06/24}{0.3a}{improved math commands}
\PassOptionsToPackage{intlimits,sumlimits,namelimits}{amsmath}
\RequirePackage{amssymb,mathtools,xfrac,isomath}
```

We optionally provide commands to typeset common sets.

`h_define_common_sets:` (no arguments)

⟨package⟩
```
\cs_new:Nn\__skmath_define_common_sets:{
```

`\N`(no arguments)

⟨package⟩
```
    \NewDocumentCommand\N{}{\ensuremath{\mathbb{N}}}
```

`\Z`(no arguments)

⟨package⟩
```
    \NewDocumentCommand\Z{}{\ensuremath{\mathbb{Z}}}
```

`\Q`(no arguments)

[1] https://github.com/urdh/skmath/issues

⟨package⟩

```
  \NewDocumentCommand\Q{}{\ensuremath{\mathbb{Q}}}
```

\R(no arguments)

⟨package⟩

```
  \NewDocumentCommand\R{}{\ensuremath{\mathbb{R}}}
```

\C(no arguments)

⟨package⟩

```
  \NewDocumentCommand\C{}{\ensuremath{\mathbb{C}}}
```

⟨package⟩

```
}
```

This is followed by commands to typeset the norm and absolute value.

\abs(no arguments)

⟨package⟩

```
\DeclarePairedDelimiter\abs{\lvert}{\rvert}
```

\norm(no arguments)

⟨package⟩

```
\DeclarePairedDelimiter\norm{\lVert}{\rVert}
```

Next come the statistical commands.

\E(no arguments)

Here, we define \E after the preamble since it may break otherwise.

⟨package⟩

```
\AtBeginDocument{
  \DeclareDocumentCommand\E{m}{%
    \ensuremath{\operatorname{E}\left[#1\right]}%
  }
}
```

The \P command saves any old \given command, replacing it locally with the new \given command provided by the package.

\P(no arguments)

⟨package⟩

```
\DeclareDocumentCommand\P{m}{%
  \ensuremath{\operatorname{P}%
    \mkern-1.5mu\left(%
    \cs_set_eq:NN\__skmath_saved_given:\given%
```

`\given` (no arguments)

⟨package⟩

```
    \DeclareDocumentCommand\given{}{\mid}%
```

⟨package⟩

```
    #1%
    \cs_set_eq:NN\given\__skmath_saved_given:%
    \right)%
  }%
}
```

`\var`(no arguments)

⟨package⟩

```
  \DeclareDocumentCommand\var{m}{%
    \ensuremath{\operatorname{Var}\left(#1\right)}%
  }
```

`\cov`(no arguments)

⟨package⟩

```
  \DeclareDocumentCommand\cov{mm}{%
    \ensuremath{\operatorname{Cov}\left(#1,#2\right)}%
  }
```

We replace all trigonometric functions and some other common functions with alternatives that take an argument (or optionally, several arguments).

⟨package⟩

```
\cs_new_eq:NN\__skmath_sin:\sin
\cs_new_eq:NN\__skmath_cos:\cos
\cs_new_eq:NN\__skmath_tan:\tan
\cs_new_eq:NN\__skmath_cot:\cot
\cs_new_eq:NN\__skmath_arcsin:\arcsin
\cs_new_eq:NN\__skmath_arccos:\arccos
\cs_new_eq:NN\__skmath_arctan:\arctan
```

```
\cs_new_eq:NN\__skmath_sinh:\sinh
\cs_new_eq:NN\__skmath_cosh:\cosh
\cs_new_eq:NN\__skmath_tanh:\tanh
\cs_new_eq:NN\__skmath_ln:\log
\cs_new_eq:NN\__skmath_log:\log
\cs_new_eq:NN\__skmath_exp:\exp
\cs_new_eq:NN\__skmath_min:\min
\cs_new_eq:NN\__skmath_max:\max
\cs_new_eq:NN\__skmath_sup:\sup
\cs_new_eq:NN\__skmath_inf:\inf
```

`\sin`(no arguments)

⟨package⟩

```
\RenewDocumentCommand\sin{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_sin:\left(#2\right)}}
    {\ensuremath{\__skmath_sin:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

`\cos`(no arguments)

⟨package⟩

```
\RenewDocumentCommand\cos{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_cos:\left(#2\right)}}
    {\ensuremath{\__skmath_cos:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

`\tan`(no arguments)

⟨package⟩

```
\RenewDocumentCommand\tan{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_tan:\left(#2\right)}}
    {\ensuremath{\__skmath_tan:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

`\cot`(no arguments)

⟨package⟩

```
\RenewDocumentCommand\cot{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_cot:\left(#2\right)}}
    {\ensuremath{\__skmath_cot:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

\arcsin(no arguments)

⟨package⟩

```
\RenewDocumentCommand\arcsin{m}{%
  \ensuremath{\__skmath_arcsin:\left(#1\right)}%
}
```

\arccos(no arguments)

⟨package⟩

```
\RenewDocumentCommand\arccos{m}{%
  \ensuremath{\__skmath_arccos:\left(#1\right)}%
}
```

\arctan(no arguments)

⟨package⟩

```
\RenewDocumentCommand\arctan{m}{%
  \ensuremath{\__skmath_arctan:\left(#1\right)}%
}
```

\sinh(no arguments)

⟨package⟩

```
\RenewDocumentCommand\sinh{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_sinh:\left(#2\right)}}
    {\ensuremath{\__skmath_sinh:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

\cosh(no arguments)

⟨package⟩

```
\RenewDocumentCommand\cosh{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_cosh:\left(#2\right)}}
    {\ensuremath{\__skmath_cosh:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

\tanh(no arguments)

⟨package⟩

```
\RenewDocumentCommand\tanh{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_tanh:\left(#2\right)}}
    {\ensuremath{\__skmath_tanh:\c_math_superscript_token{#1}\left(#2\right)}}%
}
```

\ln(no arguments)

⟨package⟩

```
\RenewDocumentCommand\ln{m}{%
  \ensuremath{\__skmath_ln:\left(#1\right)}%
}
```

\log(no arguments)

⟨package⟩

```
\RenewDocumentCommand\log{om}{%
  \IfNoValueTF{#1}
    {\ensuremath{\__skmath_log:\left(#2\right)}}
    {\ensuremath{\__skmath_log:\c_math_subscript_token{#1}\left(#2\right)}}%
}
```

\exp(no arguments)

⟨package⟩

```
\RenewDocumentCommand\exp{sm}{\ensuremath{
  \IfBooleanTF{#1}{
    \__skmath_exp:\left(#2\right)
  }{
    \mathchoice
      {e\c_math_superscript_token{#2}}
      {\__skmath_exp:\left(#2\right)}
      {\__skmath_exp:\left(#2\right)}
      {\__skmath_exp:\left(#2\right)}
  }
}}
```

⟨package⟩

```
\cs_new_nopar:Npn\__skmath_minmax_backend:nnnn#1#2#3#4{%
  \use:c{__skmath_#1:}
  \IfNoValueF{#3}{
    \c_math_subscript_token{
      \mathchoice{\mathclap{#3}}{#3}{#3}{#3}
    }
  }
  \IfBooleanTF{#2}{#4}{\left\{#4\right\}}
}
```

\min(no arguments)

⟨package⟩

```
\RenewDocumentCommand\min{som}{%
  \ensuremath{\__skmath_minmax_backend:nnnn{min}{#1}{#2}{#3}}
}
```

\argmin(no arguments)

⟨package⟩

```
\NewDocumentCommand\argmin{som}{%
  \ensuremath{\arg\__skmath_minmax_backend:nnnn{min}{#1}{#2}{#3}}
}
```

\max(no arguments)

⟨package⟩

```
\RenewDocumentCommand\max{som}{%
  \ensuremath{\__skmath_max_minbackend:nnnn{max}{#1}{#2}{#3}}
}
```

\argmax(no arguments)

⟨package⟩

```
\NewDocumentCommand\argmax{som}{%
  \ensuremath{\arg\__skmath_minmax_backend:nnnn{max}{#1}{#2}{#3}}
}
```

\sup(no arguments)

⟨package⟩

```
\RenewDocumentCommand\sup{som}{%
  \ensuremath{\__skmath_minmax_backend:nnnn{sup}{#1}{#2}{#3}}
}
```

\inf(no arguments)

⟨package⟩

```
\RenewDocumentCommand\inf{som}{%
  \ensuremath{\__skmath_minmax_backend:nnnn{inf}{#1}{#2}{#3}}
}
```

The fraction command is modified to improve typesetting.

\frac(no arguments)

⟨package⟩

```
\RenewDocumentCommand\frac{mm}{\genfrac{}{}{}{}%
             {\displaystyle #1}{\displaystyle #2}}
```

We introduce a couple of helper macros for the \pd macro backend.

⟨package⟩

```
\prg_new_conditional:Npnn\__skmath_if_numerical:n#1{p,T,F,TF}{
  \str_case_x:nnn{#1}{
    {0}{\prg_return_true:}
    {1}{\prg_return_true:}
    {2}{\prg_return_true:}
    {3}{\prg_return_true:}
    {4}{\prg_return_true:}
    {5}{\prg_return_true:}
    {6}{\prg_return_true:}
    {7}{\prg_return_true:}
    {8}{\prg_return_true:}
    {9}{\prg_return_true:}
  }{\prg_return_false:}
```

```
}
⟨package⟩
\cs_new_nopar:Npn\__skmath_pd_vars_sum:n#1{
  \tl_clear:N\l_tmpa_tl
  \int_zero:N\l_tmpa_int
  \bool_set_true:N\l_tmpa_bool
  \clist_set:Nn\l_tmpa_clist{#1}
  \clist_map_inline:Nn\l_tmpa_clist{
    \seq_set_split:Nnn\l_tmpa_seq{^}{##1}
    \int_compare:nT{\seq_count:N\l_tmpa_seq<\c_two}{
      \seq_put_right:Nn\l_tmpa_seq{1}
    }
    \seq_get_right:NN\l_tmpa_seq\l_tmpb_tl
    \__skmath_if_numerical:nTF{\tl_use:N\l_tmpb_tl}{
      \int_add:Nn\l_tmpa_int{\tl_use:N\l_tmpb_tl}
    }{
      \bool_set_false:N\l_tmpa_bool
      \tl_if_empty:NF\l_tmpa_tl{\tl_put_right:Nn\l_tmpa_tl{+}}
      \tl_put_right:Nx\l_tmpa_tl{\tl_use:N\l_tmpb_tl}
    }
  }
  \int_compare:nT{\l_tmpa_int>\c_zero}{\int_use:N\l_tmpa_int}
  \bool_if:NF\l_tmpa_bool{
    \int_compare:nT{\l_tmpa_int>\c_zero}{+}
    \tl_use:N\l_tmpa_tl
  }
}
⟨package⟩
\prg_new_conditional:Npnn\__skmath_pd_if_vars_sum_above_one:n#1{p,T,F,TF}{
  \clist_set:Nn\l_tmpa_clist{#1}
  \int_compare:nTF{\clist_count:N\l_tmpa_clist>\c_one}{
    \prg_return_true:
  }{
    \clist_pop:NN\l_tmpa_clist\l_tmpa_tl
    \seq_set_split:NnV\l_tmpa_seq{^}{\l_tmpa_tl}
    \int_compare:nTF{\seq_count:N\l_tmpa_seq<\c_two}{
      \prg_return_false:
    }{
      \prg_return_true:
    }
  }
}
```

⟨package⟩

```
\cs_new_nopar:Npn\__skmath_pd_superscript_vars:n#1{
  \clist_set:Nn\l_tmpa_clist{#1}
  \clist_map_inline:Nn\l_tmpa_clist{
    \partial
    \seq_set_split:Nnn\l_tmpa_seq{^}{##1}
    \seq_pop:NN\l_tmpa_seq\l_tmpb_tl
    \tl_use:N\l_tmpb_tl
    \int_compare:nT{\seq_count:N\l_tmpa_seq>0}{
      \seq_pop:NN\l_tmpa_seq\l_tmpb_tl
      \c_math_superscript_token
      \tl_use:N\l_tmpb_tl
    }
  }
}
```

⟨package⟩

```
\cs_new_nopar:Npn\__skmath_pd_subscript_vars:n#1{
  \clist_set:Nn\l_tmpa_clist{#1}
  \clist_map_inline:Nn\l_tmpa_clist{
    \seq_set_split:Nnn\l_tmpa_seq{^}{##1}
    \seq_pop:NN\l_tmpa_seq\l_tmpa_tl
    \int_set:Nn\l_tmpa_int{\c_one}
    \int_compare:nT{\seq_count:N\l_tmpa_seq>\c_zero}{
      \seq_pop:NN\l_tmpa_seq\l_tmpb_tl
      \int_set:Nn\l_tmpa_int{\tl_use:N\l_tmpb_tl}
    }
    \prg_replicate:nn{\l_tmpa_int}{\tl_use:N\l_tmpa_tl}
  }
}
```

⟨package⟩

```
\cs_new_nopar:Npn\__skmath_pd_fraction:nn#1#2{
  \frac{
    \partial
    \__skmath_pd_if_vars_sum_above_one:nT{#2}{
      \c_math_superscript_token{\__skmath_pd_vars_sum:n{#2}}
    }
    {#1}
  }{
    \__skmath_pd_superscript_vars:n{#2}
  }
}
```

⟨package⟩

```
\cs_new_nopar:Npn\__skmath_pd_subscript:nn#1#2{
  {#1}\c_math_subscript_token{
    \__skmath_pd_subscript_vars:n{#2}
  }
}
```

Definition of `\bar`, `\pd` and `\d` is deferred until after all packages are loaded to avoid collisions with other packages.

⟨package⟩

```
\AtBeginDocument{%
```

The `\bar` command is modified to impove typesetting.

`\bar`(no arguments)

⟨package⟩

```
\DeclareDocumentCommand\bar{m}{%
    \ensuremath{\mkern 1.5mu\overline{\mkern-1.5mu{#1}\mkern-1.5mu}\mkern 1.5mu}}
```

This is the partial derivative macro, but most of the functionality was defined as private macros earlier.

`\pd` `#1`: Boolean distinguishing between starred and unstarred variant
`#2`: Tokens representing a mathematical function
`#3`: Comma-separated list of variables, possibly with superscripts

⟨package⟩

```
\DeclareDocumentCommand\pd{smm}{
  \ensuremath{
    \IfBooleanTF{#1}
```

```
      {\__skmath_pd_subscript:nn{#2}{#3}}
      {\__skmath_pd_fraction:nn{#2}{#3}}
  }
}
```

We introduce a command to typeset the differential part of integrals, shamefully stolen from an answer on TEX.SE.

\d(no arguments)

```
⟨package⟩ \DeclareDocumentCommand\d{m}{\ensuremath{\,\mathrm{d}#1%
                                  \peek_meaning_ignore_spaces:NT\d{\!}}}
```

```
⟨package⟩ }
```

Finally, we define a nicer way to denote vectors.

\vec(no arguments)

```
⟨package⟩ \cs_set_eq:NN\vec\vectorsym
```

We end by declaring an option.

```
⟨package⟩ \keys_define:nn{skmath}{
            commonsets .code:n =
              {\__skmath_define_common_sets:}
          }
          \ProcessKeysOptions{skmath}
```

```
⟨package⟩ \endinput
```

5 Installation

The easiest way to install this package is using the package manager provided by your LATEX installation if such a program is available. Failing that, provided you have obtained the package source (`skmath.tex` and `Makefile`) from either CTAN or Github, running `make install` inside the source directory works well. This will extract the documentation and code from `skmath.tex`, install all files into the TDS tree at `TEXMFHOME` and run `mktexlsr`.

If you want to extract code and documentation without installing the package, run `make all` instead. If you insist on not using `make`, remember that packages distributed using skdoc must be extracted using `pdflatex`, *not* `tex` or `latex`.

6 Changes

v0.1

General: Initial version.

v0.1a

`\d`: Fixed obtuse errors.

v0.1b

General: Load amsmath with `intlimits` option.

`\bar`: Added `\bar` replacement.

`\C`: Moved to xparse command definition.

`\d`: Moved to xparse command definition.

`\exp`: Moved to xparse command definition.

`\frac`: Moved to xparse command definition.

`\N`: Moved to xparse command definition.

`\Q`: Moved to xparse command definition.

`\R`: Moved to xparse command definition.

`\Z`: Moved to xparse command definition.

v0.1c

General: Moved package from docstrip to skdoc.

v0.1d

General: Fixed fatal documentation and package errors.

v0.1e

General: Added statistics commands.

`\cov`: Added `\cov` command.

`\E`: Added `\E` command.

`\given`: Added `\given` command.

`\P`: Added `\P` command.

`\var`: Added `\var` command.

v0.1f

\E: Fixed 'Command \E already defined!' error.

v0.1g

General: Documentation fixes.

v0.1h

\bar: Wrap in \AtBeginDocument.

v0.2

General: Use expl3 functionality throughout the package.

\cov: Use \operatorname.

\d: Use \peek_meaning_ignore_spaces:NT instead of \@ifnextchar.

\E: Use \operatorname.

\P: Use \operatorname, use \cs_new_eq:NN instead of \let.

\var: Use \operatorname.

\vec: Use \cs_new_eq:NN instead of \let.

v0.2a

General: Load amsmath with more limits options.

v0.3

General: Added \min/\max and friends. Added \pd.

v0.3a

General: Added \sinh, \cosh and \tanh.

\exp: Added starred variant.

7 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the page were the implementation of the corresponding entry is discussed. Numbers in roman refer to other mentions of the entry.