The relsize package*

by Donald Arseneau†
originally based on smaller.sty by Bernie Cosell
and combined with code by Matt Swift

Sept 21, 2011

It is frequently the case that something should be typeset somewhat larger or smaller than the surrounding text, whatever that size happens to be. Specifying such sizing commands explicitly makes it difficult to modify the font sizes of a document at a later time, and makes it hard to write macros that work at arbitrary sizes. To fill this need, relsize.sty defines several commands for LaTeX to set font sizes relative to the current size.

Command	Function
`\relsize{`i`}`	Change font size by i steps.
`\larger[`i`]`	Increase size by (optional) i steps (default 1).
`\smaller[`i`]`	Reduce font size by i steps (default 1).
`\relscale{`f`}`	Change font size by scale factor f.
`\textlarger[`i`]{`$\langle text \rangle$`}`	Text size enlarged by (optional) i steps.
`\textsmaller[`i`]{`$\langle text \rangle$`}`	Text size reduced by (optional) i steps.
`\textscale{`f`}{`$\langle text \rangle$`}`	Text size scaled by factor f.

To refresh your memory, the font sizing commands in LaTeX are, in order: `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge` (package moresize adds `\ssmall` and `\HUGE`). The main new command provided by relsize.sty is `\relsize`, which takes one (positive or negative) number as its argument; the number specifies how many "steps" by which to change the font size, where each step is a scaling factor of 1.2, corresponding to the usual difference between the size commands. For example, if `{\relsize{-2} smaller}` appears in normal sized text, the word "smaller" is printed in script size type. If the same command appears in a `\Large` section title, then "smaller" is printed in normal size.

*This manual corresponds to relsize.sty v4.0, dated Sept 21, 2011.

†asnd@triumf.ca, Vancouver, Canada

There are also the commands `\larger` and `\smaller`, which normally change the font size by one step in the obvious direction; `\larger` is an abbreviation for `\relsize{+1}`, and `\smaller` is an abbreviation for `\relsize{-1}`. For example, `{\large... \larger{WOW!}}` prints "WOW!" in `\Large` type. You can also specify bigger steps as an optional argument for `\larger` and `\smaller`: `\larger[3]` is equivalent to `\relsize{3}` and `\smaller[2]` is `\relsize{-2}`. (Both `\larger` and `\smaller` accept negative arguments, but please don't make things so obscure!) If you want to change size by several steps it is much better to give an increment than to string several `\larger` commands together; i.e., `\relsize{3}` or `\larger[3]`, but not `\larger\larger\larger`. Half-steps are possible, as in `\relsize{-0.5}` to change from 10 pt `\normalsize` to 9 pt `\small`, but other numbers are rounded to the nearest half-integer.

All of the `\relsize`, `\larger`, and `\smaller` commands are "switches" just like the regular sizing commands. That is, they change the size for all following text until the scope is ended by a closing brace (or tabular cell, or environment ...). There are alternate versions called `\textlarger` and `\textsmaller` that take some text as an argument and apply the size change to only that text: `\textlarger{big}`.

Using the number of "magnification steps" to indicate font size can be confusing to some people, and limiting in certain uses. There are commands with syntax `\textscale{`⟨*factor*⟩`}{`⟨*text*⟩`}` and `\relscale{`⟨*factor*⟩`}` to select the size based on a scale factor, like `\relscale{0.75}`.

If the size requested is too small or too large, a warning is given, and the size will only change as far as appropriate, typically `\tiny` or `\Huge`. These limits are controlled by the commands `\RSsmallest` and `\RSlargest`, which get set automatically when relsize.sty is loaded, but you can redefine them to other length values: `\renewcommand\RSlargest{50pt}` (do not use `\setlength`).

Fine point: The combination `\relsize{`n`}\relsize{`$-n$`}` is not guaranteed to restore the current font size! That is because the increment n may be enough to overflow the range of sizes, depending on `\RSsmallest`, `\RSlargest`, and `\RSpercentTolerance` (below). You should use grouping to undo relative size changes because it is unsafe to counteract one change with an "equal" change in the opposite direction. Or just use the commands that take the text as arguments, like `\textsmaller{this}`.

Typically, the font-size commands do not select fonts at precise regular size ratios (and some commands give half-intervals). `\relsize` and the others will select, and execute, the command for the size closest to the desired size. Then, if the relative difference from the target size is more than `\RSpercentTolerance` a further font-size selection is made. By default, `\RSpercentTolerance` is an empty macro, which causes automatic selection: "30" (30 %) when the current "fontshape" definition is composed of only discrete sizes, and "5" when the fontshape definition covers ranges of sizes. The higher setting for discrete fonts ensures only the pre-defined sizes get used. (By default LaTeX uses Computer Modern fonts at discrete sizes; you get full coverage of sizes by using

`\usepackage{type1cm}` or various other font package.) For special uses, or when the font shape definitions are not parsed properly, you can redefine the percent tolerance: `\renewcommand\RSpercentTolerance{10}`. Define it as "0" (zero) to ensure the scaling is exactly as specified, regardless of the document's standard font sizes.

$$- \times \bigtimes \times -$$

All of the commands described above are text commands; they cannot be used in math mode. There are special `\mathsmaller` and `\mathlarger` commands provided, but these do not use the same sizes that the text versions use. Instead, they step between the usual math "styles" which you can explicitly set using the commands `\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle`. However, the `\mathlarger` command will also increase the size beyond regular `\displaystyle` by selecting a larger regular font size (using `\larger`). (Yes, this is a kludge, and doesn't work very well, but it can still be useful.) If you want to use this to create big integral signs, then you must also load the package exscale so that math symbols can change size. The sizes selected are:

Current style	`\mathsmaller` gives	`\mathlarger` gives
`\displaystyle`	`\textstyle` (similar)	`\displaystyle` in `\larger` font
`\textstyle`	`\scriptstyle`	`\displaystyle` (similar)
`\scriptstyle`	`\scriptscriptstyle`	`\textstyle`
`\scriptscriptstyle`	`\scriptscriptstyle`	`\scriptstyle`

For example, try `$\frac{\mathlarger{E}}{E}$` $= \frac{E}{E}$. Note that, for most symbols, `\displaystyle` and `\textstyle` give the same size, so `$N,\mathlarger{N}$` $= NN$, showing two identical N's, but `\sum` and `\int` do get bigger in display style, and fractions are treated differently too:

`$\int\frac{1}{2}dN - \mathlarger{\int\frac{1}{2}dN}$` $= \int \frac{1}{2}dN - \int \frac{1}{2}dN.$

As you might have guessed, `\mathlarger` and `\mathsmaller` should only be used in math mode.

These commands will attempt to attach any superscripts and subscripts directly to the symbol within the braces, rather than how they would attach to a math sub-formula. On the other hand, math accents and the math spacing do behave as if the symbol is enclosed in braces (which it is). Operators should be explicitly declared to use the right operator type (`\mathrel`, `\mathbin`, `\mathop`) to get the correct spacing, e.g., `\mathrel{\mathsmaller{=}}`.

Due to their oddities, the math larger/smaller commands should not be trusted very far, and they will not be useful in every instance.