Variable fonts in LuaTeX, with an introduction to the Junicode VF and Elstob fonts

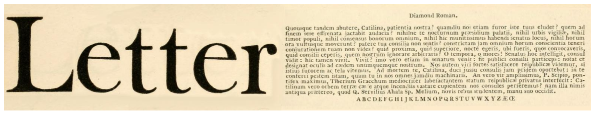
Peter S. Baker

Abstract

This paper introduces variable fonts, now supported by LuaTeX, and explains the benefits this new font technology offers to LuaLaTeX users — chief among these being the restoration of some of the typographical capabilities of metal type, nearly lost with the advent of digital fonts. It then describes the capabilities of two variable fonts developed by the author — Junicode VF (where "V" stands for "variable", not "virtual") and Elstob — and briefly introduces the packages that provide access to them, especially the options and commands for controlling their axes.

1 Background

In the days of metal type, long before fonts were scalable, type had to be produced in a variety of sizes. For example, forty-seven pages of the 1798 Caslon *Specimen of Printing Types* [3, pp. 13–105] are devoted to the display of roman and italic type in sizes ranging from "Six Lines Pica" (about 72pt) to "Diamond" (about 4.5pt) — see fig. 1. Type within



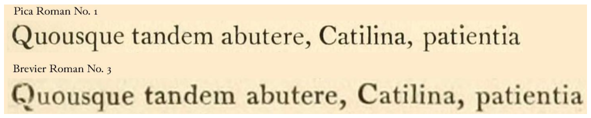
Diamond Roman.

Quousque tandem abutere, Catilina, patientia nostra? quamdiu nos etiam furor iste tuus eludet? quem ad finem sese effrenata jactabit audacia? nihilne te nocturnum præsidium palatii, nihil urbis vigiliæ, nihil timor populi, nihil consensus bonorum omnium, nihil hic munitissimus habendi senatus locus, nihil horum ora vultusque moverunt? patere tua consilia non sentis? constrictam jam omnium horum conscientia teneri conjurationem tuam non vides? quid proxima, quid superiore, nocte egeris, ubi fueris, quos convocaveris, quid consilii ceperis, quem nostrum ignorare arbitraris? O tempora, o mores! Senatus hoc intelligit, consul vidit: hic tamen vivit. Vivit? imo vero etiam in senatum venit: fit publici consilii particeps: notat et designat oculis ad cædem unumquemque nostrum. Nos autem viri fortes satisfacere reipublicæ videmur, si istius furorem ac tela vitemus. Ad mortem te, Catilina, duci jussu consulis jam pridem oportebat: in te conferri pestem istam, quam tu in nos omnes jamdiu machinaris. An vero vir amplissimus, P. Scipio, pontifex maximus, Tiberium Gracchum mediocriter labefactantem statum reipublicæ privatus interfecit: Catilinam vero orbem terræ cæde atque incendiis vastare cupientem nos consules perferemus? nam illa nimis antiqua prætereo, quod Q. Servilius Ahala Sp. Melium, novis rebus studentem, manu suo occidit.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Æ Œ

Figure 1: Six Lines Pica and Diamond type, from *A Specimen of Printing Types, by Wm Caslon* [3, pp. [27], [101]].

many font families varied not only in size, but also in shape, with smaller sizes cut proportionally wider, heavier, and with a higher x-height than larger sizes — see fig. 2, where two Caslon specimens, about 12pt and 8pt, have been scaled to the same size. Well into



Pica Roman No. 1

Quousque tandem abutere, Catilina, patientia

Brevier Roman No. 3

Quousque tandem abutere, Catilina, patientia

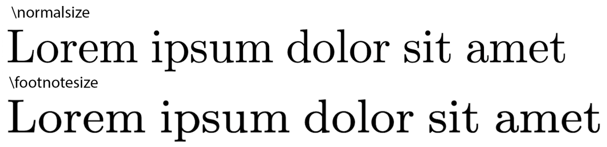
Figure 2: Two sizes of type, from *A Specimen of Printing Types, by Wm Caslon* [3, pp. [77], [97]].

the twentieth century, foundries offered their major typefaces in a wide variety of styles and sizes. For example, the American Type Founders catalogue of 1923 devotes sixty-one pages to a dizzying number of Caslon types — "Caslon Bold Condensed" (6pt to 120pt), "Heavy Caslon" (6pt to 84pt), and "Caslon Openface" (8pt to 48pt), to mention only a few [2, pp. 130–191]. In all cases, smaller and larger types varied in shape as well as in size.

The advent of phototypesetting in the 1950s brought with it the ability to scale type by means of differently powered lenses — but type scaled in this way could vary only in size, not in shape. Although foundries continued to provide a wide variety of styles within type families, typographers increasingly thought scaling alone a good enough solution to the problem posed by the need for differently sized text in printed works. If the type of the footnotes looked anemic compared to that of the body text, that seemed an acceptable price to pay for the convenience and economy of working with a single typeface.

Digital typesetting brought at least the possibility of a return to the practices of early typefounders, and indeed some modern font families feature the kind of variation by size found in the Caslon type catalogue. For example, a number of Adobe "Pro" font families have styles labeled "Caption", "Subhead", and "Display" in addition to the default. Such styles are called **optical sizes** because they are designed to be optically correct within certain size ranges. However, fonts with optical sizes are uncommon and often costly. Most digital fonts belong to so-called RIBBI families, consisting of just four styles — Regular, Italic, Bold, and Bold Italic. Users generally think these styles quite sufficient, and only the most sophisticated typographers bother with such stylistic niceties as those offered by the Adobe "Pro" fonts — especially when they have to be manually selected.[1]

TeX can boast some of the most sophisticated typographical capabilities of any digital publishing system. Donald Knuth's Computer Modern family of fonts (along with several derivatives) has always featured optical sizes similar to those typical of metal type (fig. 3), and TeX automatically selects the correct optical size for any run of text set in Computer Modern: users rarely have to think about it.



\normalsize

Lorem ipsum dolor sit amet

\footnotesize

Lorem ipsum dolor sit amet

Figure 3: Computer Modern at `\normalsize` and `\footnotesize`, scaled to the same size for comparison.

The introduction of Jonathan Kew's XeTeX in 2004, followed soon afterwards by Will Robertson's fontspec, was a momentous development for TeX.

[1] For a brief history of optical sizes, see Ahrens and Mugikura [1, pp. 17–25].

doi.org/10.47397/tb/45-1/tb139baker-junicodevf

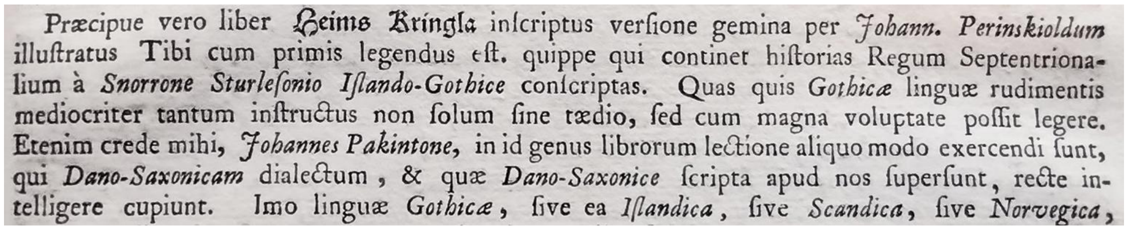

Præcipue vero liber Heims Kringla inſcriptus verſione gemina per *Johann. Perinskioldum*
illuſtratus Tibi cum primis legendus eſt. quippe qui continet hiſtorias Regum Septentriona-
lium à *Snorrone Sturleſonio Iſlando-Gothice* conſcriptas. Quas quis *Gothicæ* linguæ rudimentis
mediocriter tantum inſtructus non ſolum ſine tædio, ſed cum magna voluptate poſſit legere.
Etenim crede mihi, *Johannes Pakintone*, in id genus librorum lectione aliquo modo exercendi ſunt,
qui *Dano-Saxonicam* dialectum , & quæ *Dano-Saxonice* ſcripta apud nos ſuperſunt, recte in-
telligere cupiunt. Imo linguæ *Gothicæ*, ſive ea *Iſlandica*, ſive *Scandica*, ſive *Norvegica*,

Figure 4: A sample of Fell's Pica roman and italic type, from Hickes [5].

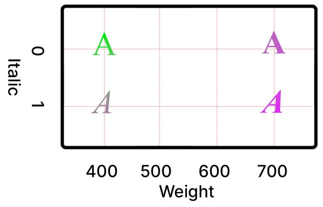


Figure 5: Design space for the four-member Times New Roman font family.

Quite suddenly, users gained the ability to access all the fonts installed on their systems, and most of these were in the popular OpenType format.[2] For the typographically ambitious, new vistas opened up.

But with its embrace of OpenType fonts, the TEX community not only benefited from their convenience, but also inherited their limitations. The vast majority of font families installed in any system, and indeed in the CTAN repository, offered only the traditional four styles, while most extended font families — those with more than the four RIBBI styles — were nevertheless stylistically impoverished in comparison with Computer Modern or the immense Caslon family of the 1798 *Specimen*.[3] Twenty years later, font technology has advanced, but we are still struggling to make up the ground we lost in the transition to digital typography.

In font families of the kind I have been discussing, both RIBBI and extended, the collection of styles is organized around one or more **axes** — that is, aspects of a font's design that can change in a systematic way. A font's axes can be pictured as being like the axes in a graph — though in font terminology the graph is called the **design space**. The most common axis is **Weight** — that is, the proportion of black to white in a font's glyphs. Other standard axes are **Width**, **Slant**, and **Optical Size**. **Italic**

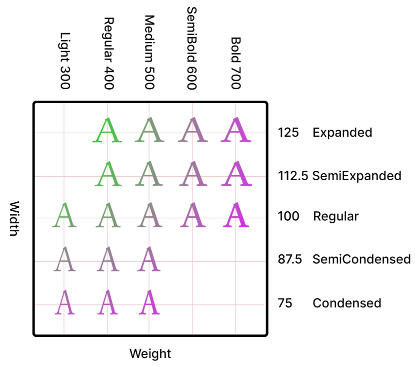


Figure 6: Design space for the nineteen-member Junicode roman subfamily.

is also an axis, though one with only two values, since a font either is or is not italic. A RIBBI family positions its four styles at the extremes of two axes, Weight and Italic (see fig. 5), but an extended family may fill in some of the empty spaces in and around the RIBBI design space and add its own axes as well.

The lost ground I mentioned above has been much on my mind as I've worked on the two fonts I am going to discuss in this article. I first developed Junicode in the mid-1990s as a tool for students and scholars of medieval Europe, but it has grown over the decades to support scholars in numerous disciplines (mostly linguistic, literary, and historical). The font is based on types commissioned by John Fell (1625–1686), Bishop of Oxford and a key figure in the early history of the Oxford University Press (see fig. 4).

Fell bequeathed these types to the university, and they were used in many books issued by the Press in the seventeenth and eighteenth centuries. Version 1 of Junicode was a RIBBI family, but in 2019 I began work on a more capable version, released in August 2023. Junicode version 2 is an extended family with three major axes (Italic, Weight, and Width), which are combined in various ways to make thirty-eight styles or **instances** — that is, locations in a font's design space selected and named by the designer. Fig. 6 shows the design space for Junicode's roman face, with its nineteen instances.

[2] See Kew [7] and Robertson [8]. OpenType is a font format, first developed by Microsoft in the 1990s, that enables such features as ligatures, the setting of complex scripts like Arabic and Devangari, and much more.

[3] XƎTEX automatically uses the optical sizes of Adobe's "Pro" fonts, but these cannot be included in CTAN — and in any case they offer fewer styles than the larger metal type families.

Junicode is well known to TeX users, having been in the CTAN repository since 2009. With more than 5,000 glyphs, it is an unwieldy thing, and as many of these glyphs have non-standard encodings, it can raise accessibility issues in digital texts of all kinds. Version 2 offers solutions to the accessibility problems of version 1, but the non-standard encodings remain — each of them a potential trap for the unwary user.[4]

With both accessibility and typographical issues in mind, I began work on the Elstob font (`github.com/psb1558/Elstob-font`) in 2018. The first alpha version was released the following year, and the current version is 2.104. Based on another of Bishop Fell's typefaces, Elstob was meant to be lightweight and entirely standards-compliant: while it would not have Junicode's vast character set, its curated selection of glyphs would meet the needs of most medievalists and linguists, and using it would all but guarantee an accessible end product. Further, Elstob would have both Weight and Optical Size axes so that different sizes of type could coexist more comfortably on a page.

Version 1 of Junicode was a static font family — that is, one in which each instance is packaged in its own file. Version 2 also comes in a static version consisting of thirty-eight font files, while the static version of Elstob consists of no fewer than forty-eight files. But each of these fonts has a much more capable and compact variable version as well.

2 Variable fonts

Even extended font families impose significant limitations on users. Because every instance of a family adds a file (in the case of Junicode, a rather large one), many locations in the design space are unavailable to users. Often a large area of the design space may have no instances in it at all. To most users the poverty of the typical static font family will not seem a hardship — after all, it's what we're used to. But what if you could set your chosen typeface in *any* weight and *any* width? If you think you might make use of such an ability, you will like variable fonts.

The specification for variable fonts (the official name is "OpenType Font Variations") first appeared in 2016,[5] and these fonts have been steadily gaining ground ever since. A variable font is one that replaces the several files of a font face (usually roman or italic) with a single file containing a set of glyph outlines accompanied by deltas governing their transformation. With appropriate software support, outlines can be transformed *continuously* along the font's axes by supplying numerical values. So Junicode VF, the variable version of Junicode, replaces the thirty-eight files of the static font with just two — one for roman and one for italic. Both the static and the variable fonts are based on the same design space, but in the variable font *every possible location* is occupied — not only the blank areas of fig. 6, but also the interstices. If you think the Regular weight of 400 a little too light and the Medium weight of 500 too heavy, you can choose a weight of 420, or 457, or 443.25. See fig. 7, where the difference between 457 and the surrounding weights is subtle, but would make a visible difference in the darkness of a text block.

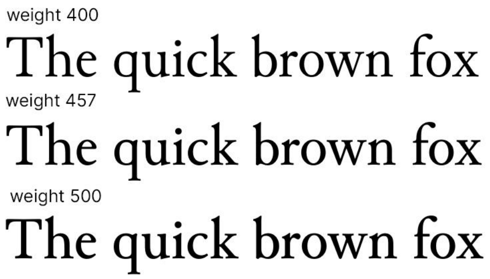


Figure 7: Junicode VF roman at weights of 400 ("Regular"), 457 (custom), and 500 ("Medium").

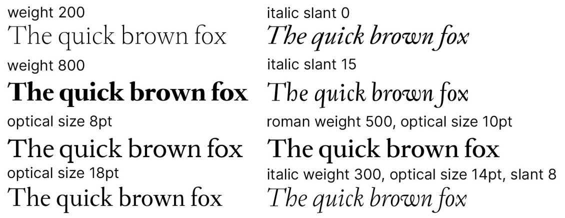


Figure 8: A sampler of styles available with the Elstob font.

Like the static Junicode font, Junicode VF is in the CTAN repository, along with documentation and a package for loading the font and accessing its various features. Elstob is not in CTAN, but can be downloaded for free;[6] the latest releases come with a package for TeX users like the one that accompanies Junicode VF.

In addition to Elstob's Weight and Optical Size axes, the italic face includes a Slant axis (see fig. 8).[7]

[4] For more about the accessibility problems raised by Junicode's extensive use of Unicode Private Use Area encodings for specialist medieval glyphs, see the *Junicode Manual* (`github.com/psb1558/Junicode-font/blob/master/docs/JunicodeManual.pdf`), §4.1.

[5] See the OpenType specification [4], especially the section "OpenType Font Variations overview". For an accessible introduction to variable fonts, see Hudson [6].

[6] `github.com/psb1558/Elstob-font`. Like most open source fonts, Elstob can be downloaded from numerous commercial sites, but to obtain the latest version, users should download only from the GitHub repository.

[7] Junicode and Elstob also have specialized axes which are available to TeX users. For Junicode, "Enlarge" lets users set

The current release of Elstob comes with a package like the one for Junicode VF.

3 Support for variable fonts in TEX

Experimental support for variable fonts in luaotfload, the font loader for LuaTEX, first appeared in December 2020.[8] Since that time, support for these fonts has developed rapidly, so that it can now be called mature: if the program has any significant shortcomings or bugs, I have been unable to discover them, though I have been using it almost daily for the better part of a year.

Users should make sure they are running version 3.26 (included in TEX Live 2023) or later of luaotfload, as variable font support is incomplete in earlier versions.

As of version 2.9a (released 2024-02-13), fontspec includes explicit support for variable fonts: for details, see section III.7 of its documentation. Variable fonts can also be managed via fontspec's RawFeature command, present in older versions. fontspec users should always select the HarfBuzz renderer when using variable fonts, as node mode may sometimes fail to load these fonts or apply OpenType features incorrectly (the packages discussed in the next section invoke the HarfBuzz renderer).

XƎTEX and other flavors of TEX do not support variable fonts; only LuaTEX.

4 The `junicodevf` and `elstob` packages

The packages for Junicode VF and Elstob are designed to resemble many of the font packages in CTAN — for example, those for `ebgaramond`, `sourceserifpro`, and `roboto` (see Voß [9, p. 299] for a partial list of packages and a link to a complete list). They are loaded in the usual way, with `\usepackage{junicodevf}` or `\usepackage{elstob}`, and they accept a more or less standard set of options, including the following:

extralight (Elstob only) The weight of the main font (that is, the four-style collection selected by fontspec's `\setmainfont` command) is ≈200.

light The weight of the main font is ≈300.

medium The weight of the main font is ≈500.

semibold The weight of the Bold style of the main font is ≈600.

extrabold (Elstob only) The weight of Bold style of the main font is ≈800.

the enlarged lowercase letters that often begin sentences in medieval manuscripts; for Elstob, "Grade" changes the weight of text without changing its width (a more useful feature for web designers than for TEX users), and "Spacing" approximates the word- and sentence-spacing of early metal type.

[8] See the `NEWS` file in the luaotfload repository (`github.com/latex3/luaotfload`, accessed 2024-3-18).

condensed (Junicode VF only) The width of the main font (≈75) is about 85% of Regular (100).

semicondensed (Junicode VF only) The main font is wider than Condensed but narrower than Regular (≈87.5).

expanded (Junicode VF only) The width of the main font is about 115% (≈125) of Regular.

semiexpanded (Junicode VF only) The main font is wider than Regular, but narrower than expanded (≈112.5).

Although these options resemble those for CTAN's static fonts, they produce very different effects, in that they do not produce text in a fixed style, but rather in a range of styles that vary with text size — that is to say, optical sizes, which are supported natively by Elstob and emulated in Junicode by making fine adjustments to the Weight and Width axes (thus the approximation signs in the option list above).

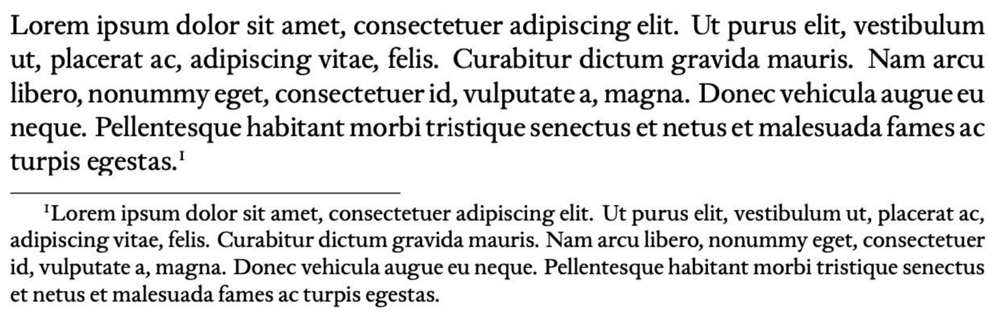
Figs. 9 and 10 illustrate the contrast between Elstob set as body text (11pt) and as footnote text (about 8pt). In fig. 9 the difference in glyph shapes is scarcely visible, but of course that is the point: as early typefounders understood, small type appears to match larger type when the shape is properly adjusted. Fig. 10, which enlarges body text and footnote text to the same size, shows the difference more clearly, the footnote text being heavier and with a higher x-height and shorter descenders than the body text.

For users dissatisfied with the `junicodevf` and `elstob` defaults and the options listed above, two sets of options allow even finer control over these fonts' optical sizing. Here is the first set, which enables adjustments to design choices made via the standard options:

weightadjustment Adjusts the weight of the type by adding this number. For example, if you choose `medium` for the main font (weight ≈500) and `bold` (the default, with weight ≈700), and also include the option `weightadjustment=-25`, then the weights of Medium and Bold text will be lightened by 25 (to ≈475 and ≈675).

widthadjustment (Junicode only) Adjusts the width of the type by adding this number. For example, if you choose `semicondensed` for your document (width ≈87.5), and you also include the option `widthadjustment=5`, then the width will be ≈92.5, between `semicondensed` and `regular`.

opticalsizeadjustment (Elstob only) Adjusts the optical size. By default, the value of this axis is 8 for 8pt text, 12 for 12pt, etc. But if you pass the option `opticalsizeadjustment=-1.5`, the


Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.[1]

[1]Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Figure 9: Body text and footnote set in Elstob.

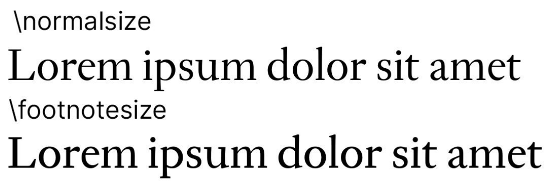

\normalsize
Lorem ipsum dolor sit amet
\footnotesize
Lorem ipsum dolor sit amet

Figure 10: Body text and footnote text enlarged to the same size.

optical size axis will be 6.5 for 8pt type, 10.5 for 12pt, etc. (always staying in the range 6–18).

The second set, to be used instead of the options listed above, gives the user complete control over axis values for every text size in a document. To illustrate, this is the relevant part of the command that loads the `junicodevf` package for the *Junicode Manual*:[9]

```
usepackage[
  MainRegularSizeFeatures={
    {size=8.6,wght=550,wdth=120},
    {size=10.99,wght=475,wdth=115},
    {size=21.59,wght=400,wdth=112.5},
    {size=21.59,wght=351,wdth=100}
  },
  MainItalicSizeFeatures={
    {size=8.6,wght=550,wdth=118},
    {size=10.99,wght=475,wdth=114},
    {size=21.59,wght=450,wdth=111},
    {size=21.59,wght=372,wdth=98}
  },
  MainBoldSizeFeatures={
    {size=8.6,wght=700,wdth=120},
    {size=10.99,wght=700,wdth=115},
    {size=21.59,wght=650,wdth=112.5},
    {size=21.59,wght=600,wdth=100}
  },
  MainBoldItalicSizeFeatures={
    {size=8.6,wght=700,wdth=118},
    {size=10.99,wght=700,wdth=114},
    {size=21.59,wght=650,wdth=111},
    {size=21.59,wght=600,wdth=98}
  },
]{junicodevf}
```

[9] `github.com/psb1558/Junicode-font/blob/master/docs/JunicodeManual.sty`.

For each of the four RIBBI styles, this command defines a list of associative arrays, each prescribing axis coordinates for a range of sizes. In these arrays, a `size` key is mandatory: any array without one is ignored. The arrays should be ordered by point size. The first array prescribes axis coordinates for all sizes up to `size`, the last array for all sizes greater than `size`, and any intermediate arrays a range from the previous to the current `size`. So the ranges covered in each list above are `-8.6pt`, `8.6-10.99pt`, `10.99-21.59pt`, and `21.59pt-`.

Keys other than `size` are the four-letter tags for the font's axes: `wght` (Weight) and `wdth` (Width).[10] When a key is omitted, the default value for that axis is used. When SizeFeatures are given in this way, they override any other options that set or change axis coordinates (e.g. `weightadjustment`).

These lists define only four size ranges because the *Junicode Manual* needs only four; but you can define as many as you need. (The `junicodevf` package, if invoked without options, defines eleven.)

Both the `junicodevf` and `elstob` packages define commands for invoking font styles that match the instances of the corresponding static fonts, plus a few more. These are listed in the fonts' documentation, but as an example, when using Junicode one can switch temporarily from the main font to Condensed Light as follows:

```
{\jCondLight The quick brown fox.}
```

These alternate styles can be customized just as the main styles can. For example, to darken and widen the `\jCondLight` style a little, include this option when loading the `junicodevf` package:

```
CondLightSizeFeatures={
    {size=4,wght=325,wdth=80},
}
```

[10] In OpenType programming, axes are identified by four-letter tags rather than their longer names. By convention, tags for axes defined in the OpenType standard are lowercase, while custom axes are uppercase. The default value of Junicode's ENLA (Enlarge) custom axis is used.

The junicodevf and elstob packages load fontspec (no need to load it again) and depend on that package for all their functionality. The options that manipulate axes generate fontspec commands, while other options are passed through to fontspec. For example, the MainFeatures option allows users to turn on features for all styles of the main font:

```
\usepackage[
    MainFeatures={
        Language=English,
        StylisticSet=9
    }
]{junicodevf}
```

Here, fontspec options set the language of the main font to English and turn on Stylistic Set 9, which modernizes some number-shapes. The same can be done for individual styles of the main font with options like MainBoldItalicFeatures, and for alternate styles with options like CondLightFeatures (features for the Condensed Light style).

Both Junicode and Elstob offer large numbers of OpenType features, and their packages also provide commands for convenient access to a selection of them. For example, Stylistic Set 12 enables transliteration of English text to early English runes for both Junicode and Elstob. One can turn it on either with a fontspec command or with a mnemonic command from the junicodevf or elstob package (with the result shown in fig. 11):

```
fontspec: \addfontfeature{StylisticSet=12}
elstob:   \EarlyEnglishFuthorc
```

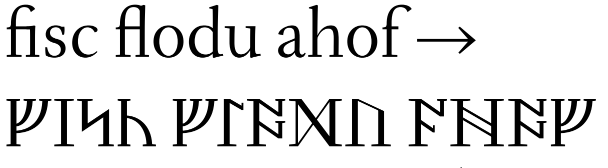


Figure 11: The effect of Stylistic Set 12 (Early English Futhorc) in Elstob.

All of these commands, listed in the Junicode and Elstob documentation, have text variants that work like \textit and \textbf:

```
\textEarlyEnglishFuthorc{fisc flodu ahof}
```

Both Junicode and Elstob, but especially Junicode, also have a number of Character Variant (cvNN) features, which afford access to one or more variants for individual characters. These can be accessed either with fontspec commands or with more compact alternates: (\jcv for Junicode, \ecv for Elstob, \textcv for both). Mnemonics can be used to select from the collection of Character Variant features in either fontspec or junicodevf/elstob commands:

```
fontspec: \addfontfeature{%
              CharacterVariant=\ecvg:1}g
elstob:   \ecv[1]{\ecvg}g
```

Here \ecvg is a mnemonic for 14, identifying Elstob's Character Variant feature for the letter **g** (cv14). The 1 that appears in both commands is an index that selects the second variant shape of that letter.

5 Conclusion

Junicode VF is, to my knowledge, the first variable font to appear in the CTAN repository, and junicodevf and elstob are first attempts at packages for loading variable fonts in LuaLaTeX. I welcome critiques of these packages, and especially their options for managing the axes of these fonts. As more variable fonts appear in CTAN, it would be useful to standardize the interfaces of any dedicated packages that accompany them.

References

[1] T. Ahrens, S. Mugikura. *Size-Specific Adjustments to Type Designs.* Just Another Foundry, Munich, 2014.

[2] American Type Founders Company. *Specimen Book & Catalogue.* [Jersey City], 1923. archive.org/details/specimenbookcata00amer

[3] W. Caslon. *A Specimen of Printing Types, by Wm Caslon, Letter-Founder to the King.* C. Whittingham, 1798. archive.org/details/specimenofprinti00casl

[4] P. Constable, K. Turetzky, et al. OpenType specification version 1.9, 2022. learn.microsoft.com/en-us/typography/opentype/spec/

[5] G. Hickes. *Linguarum vett. septentrionalium thesaurus grammatico-criticus et archæologicus.* [Oxford University Press], Oxford, 1703-1705.

[6] J. Hudson. Introducing OpenType variable fonts, 2016. medium.com/variable-fonts/https-medium-com-tiro-introducing-opentype-variable-fonts-12ba6cd2369

[7] J. Kew. XeTeX, the Multilingual Lion: TeX meets Unicode and smart font technologies. *TUGboat* 26(2):115–124, 2005. tug.org/TUGboat/tb26-2/kew.pdf

[8] W. Robertson. Advanced font features with XeTeX — the fontspec package. *TUGboat* 26(3):215–223, 2005. tug.org/TUGboat/tb26-3/tb84robertson.pdf

[9] H. Voß. Using OpenType and TrueType fonts with XeLaTeX and LuaLaTeX. *TUGboat* 43(3):295–299, 2022. tug.org/TUGboat/tb43-3/tb135voss-unifont.pdf

◇ Peter S. Baker
b.tarde (at) gmail dot com
https://github.com/psb1558/