The Typo Aid Package (v.0.4.7)

Daniele Ratti

May 7, 2017

Abstract

The typoaid package provides some useful tools in order to provide data regarding the used fonts, and some hints about typesetting them. This manual is divided in two parts, the first mainly concerned with the end-user commands and usage, the second is a collection of notes regarding future development and code

Contents

Part I
User manual

1 Introduction

Since `typoaid` comes as a set of diagnostic tools to decide on how to set the type area, in a manner that can hopefully be typographically pleasant and equilibrate, and that takes in consideration the typographic tradition.

The set of macros, though, are not meant to give a single *directive* on how big should be the measure of the text, or something like that: the tools only give tips and give common used guidelines to decide upon the correct measure for a given font.

Each one of the package commands comes with:

- an *unstarred* version (e.g.) `\tychperwidth{}`, which types out the calculations to the page
- a *starred* vesion (e.g.) `\tychperwidth*{}`, which outputs the calculation *to the terminal*
- a copy of the output data *to the log file* (both starred and unstarred commands do log)
- the possibility to accept, a series of one or more*font switches*, such as `\bfseries\itshape`, in order to provide calculation for the specific switch or combination. Please note that *also font-family switches from* `fontspec` *are supported.*

1.1 Compatibility

The package is compatible with pdfLaTeX, LuaLaTeXand XeLaTeX, and will accept a font family switch defined via the `fontspec` package.

2 Simple commands

The `\typrintalph` is used to calculate the alphabet length, given the font switches. The length is computed *without kerning,* that is: using the pure letter widths. `\typrintalph`

The `\typrintex` command is used to calculate the ex-height of the font, given the switches. `\typrintex`

The `\typrintem` command, computes the em-width of the current font. Namely this is the same as the font body size, but may yield different results for different font switches, especially with `T1` `\typrintem`

fonts, where each family could have its em-width; this is proven not to be the case for OpenType fonts.

The `\tyallsimple` command simply calls internally the aforementioned commands, and returns their outputs. Its starred version calls the starred version of the same commands. `\tyallsimple`

3 Char per width commands

The package provides two commands that calculate the *char per width* and *width given a number of characters* desired in a line. The calculations are based on the algorithm in section 6 on page 7. Please note that the calculations are bound to be approximated estimates only, and do not imply that the result will produce *exactly* what it's asked of it (being it the char per width, or the width for a given number of chars); again, refer to section 6 on page 7 to have further information.

3.1 Characters fitting a given width

The `\tychperwidth` command syntax extends the *standard* `\typoaid` syntax, since this commands accepts an optional parameter which should be a dimension. If given the calculation will be performed on that length, otherwise they will be performed on the `\columnwidth`. `\tychperwidth`

Note: the number of char is given as a rounded integer, so it's bound to be an approximate number.

Examples here are two examples on how to use the command:

```
\tychperwidth*[17pc]{}
```

Will produce:

- an output to the terminal (since the starred version is used) and the log
- a calculation on the current font with no extra switches (since the mandatory argument is empty)
- a calculation on how many characters will fit into `17pc`

```
\tychperwidth{\itshape}
```

Will instead:

- an output to the page and the log (unstarred version)
- a calculation on the italic alphabet of the current font family
- a calculation on how many characters of the aforementioned alphabet will fit into the current `\columnwidth`

3.2 Width for a given number of characters

Conversely on what is discussed in section 3.1 on the preceding page, it may be desirable to obtain the length of a column that will accommodate a specific number of characters. This is done using the`\tywidthgivchar` command. \tywidthgivchar

The command usage is somewhat different from the other commands of the package, since the font switches are accepted *as an optional argument,* while the mandatory argument is the number of characters for the calculations.

Examples here are two examples of the command usages:

```
\tywidthgivchar*{68}
```

will output in terminal (starred version) and log the width to typeset 68 characters with the current font.

```
\tywidthgivchar[\bfseries]{35}
```

Will instead output – in the page and the log – the width of the column that will accommodate 35 characters of the bold version of the current font.

4 Height commands

To determine the height of a given text, usually [1] one can determine a *form factor* to which its text will comply, and then will tweak it in order to accommodate an *integer number of lines.*

The `\tyheight` command gives an hint to the user, outputting the number of lines. The command comes with the two usual `typoaid` starred scheme, but its parameters are a bit more complex since: `\tyheight`

- it accepts a first optional parameter (in square brackets) which indicates a font switch
- it accepts two "mandatory" (i.e. curly braces-delimited) arguments, indicating resp:
  1. the height to be used (if no parameter is given, the current value of `textheight` is used)
  2. the baseline skip to be used (if no parameter is given, the current value of `baselineskip` is used
- accepts also a trailing optional parameter (square brackets) which has the optional

Examples The command syntax may appear complex, but it's actually modeled around what I think it would be its more common usage. Anyway here are some examples:

```
\tyheight{}{}
```

gives the height and number of lines for current data.

```
\tyheight{538pt}{13pt}
```

gives the number of rows in 538pt using a 13pt baselineskip.

```
\tyheight[\bfseries]{}{13pt}
```

will output the number of lines in `bfseries` with 13 pt baselineskip and current height

`\tyheight[\itshape]{540pt}{13.6pt}[11pt]`

will add the condition of the 11 pt size. Of course one can use other values and data.

Note: The other width command currently included are tabular ones, thus given in section 5.3 on the next page.

5 Tabular data

The package provides two diagnostic commands, which will output tables when used in the unstarred version – they will output a list of things in the terminal for the starred version; and also for the log in any "version".

5.1 Font tables

The `\tyfonttable` typesets a table containing \tyfonttable

- alphabet length
- ex-height
- em-width

for the following font-shapes:

- roman
- bold
- italic
- small case
- slanted
- sans-serif

The command, nevertheless accepts a switch, which is conceived to printout the data for a font family defined by the `fontspec` package, via `\newfontfamily`; but any font-family changing command that behaves the same way is just as suitable.

5.2 Width tables

The main purpose of the package is the use of the `\tywidthtable` command, which will output some suggested widths, which are commonly used [1, 2] in the professional typesetting world. The starred version, as usual, prints a long list into the terminal. \tywidthtable

There are a few remarks:

- where no name is given in parenthesis, the calculation are either provided for reference, or are the author's personal estimates of useful lengths;
- the name in parenthesis refers to methods or calculations given in [1, 2];
- all of the calculations are done on the *em width,* instead of the font *body size,* since they're almost always the same (see section 2 on page 2), and when they're not, I think it would lead to better results
- *multicol* means that a setting is most suited for multiple columns
- the number of characters for width is the result of a rounding (as most quantities here reported are) so all of the numbers are to be taken as approximates.

5.3 Height tables

The `\tyformfactorheight` is the main *vertical* command and is used to have a guideline on how tall the text should be. \tyformfactorheight

In common practice (at least the author's), the design tends to have a specific *form factor,* i.e., a given ratio between text height and width. This will have to be matched with the need to have an integer number of lines in the text height.

The command, then, will come in two flavours (starred and unstarred) and will always accept the following parameters, given in order:

- an optional font family switch (in square brackets)
- a mandatory dimension for the text height, in curly braces; if not given the package will assume `columnwidth`
- a mandatory *form factor* parameter which *cannot be empty*
- an optional square bracketed parameter, to specify the baseline skip to be used, defaulting to the current one.

Upon executing, the command will output (either in the page or the terminal, plus log) three rows containing the height, the number of lines and the form factor; these will be:

1. the *strict* result obtained multiplying the width by the form factor. Could result in non-integer number of lines per height
2. the *truncated number of lines* resulted from the previous calculation, with the updated form factor and height.
3. the *up rounded number of lines* resulted from the first calculation, with the updated form factor and height

This way the user can specify some form-factor guidelines, and see which form factors are the nearest which accommodate an integer number of lines.

Part II
Notes, details, licensing

6 Calculations

The calculations performed by this package are only indicative. The calculations are based on two concepts:

Alphabet length: this is calculated by measuring the sum of the whole letters without considering specific kernings.

Mean character width: this calculation is tricky. In fact using the typefitting table in [1], I messed around with the data and found out that the mean char width used there is $\approx 1/(26.5 + 1\% \cdot \alpha)$ where α is the alphabet length.

Use of em: as it is said before, every calculation uses the *em width* (where applicable) instead of the *character size.*

Regarding the mean char width, I don't know whether the calculation are correct for all languages or just for English. I just found out that since the calculations cannot be other than approximations, these give consistent results, even in Italian.

7 Known issues

The code works hopefully, and should do fine. Despite that it needs to be worked on, both from the functionality and code prettyness. I will go on and work on these issues. Please mark that I already know them, so don't be scared by the code

Let's go global Also I struggled a bit with the sense of *local* and *global* in `expl3` language. The code has all the variables set to be global. This is highly undesirable and an urgent step will be to refactor these in order to make them global only if needed. Unfortunately the problem became more and more apparent from v.0.4.0, but I'm not capable of doing it any other way

Required packages I don't like depending upon too many packages. Currently I depend on table typesetting packages such as `array, booktabs`. Since the package could possibly be used without any typeset output, I will consider to introduce a package-wide option to enable the aforementioned packages and functionalities.

8 Future implementations and roadmap

As seen in the previous section, the package is far from complete. I set up a little roadmap:

1. Rework and make good outputs in log and terminal
2. Introduce LuaTeX specific commands and functionalities

9 Changelog

This is a stub and will be until version 1.0 will be released, since the package is still in continuous development. The minor revisions indicate a refactoring or a new functionalities. Subminor revision track tests and tunings

v0.0.7 First release

v0.1.0 Completed the *simple commands*

v0.2.0 Introduced the *fonts table*

v0.3.0 Introduced the *width table*

v0.3.8 Fixed various issues, second release

v.0.4.7 Fixed and prettied all the code, introduced vertical commands

10 Acknowledgments

Special thanks to Enrico Gregorio, which, apart from helping me – especially for the LuaTEX safe no-kerning code – also tried to teach me `expl3` and some of its nuisances. I'm a bit ashamed to be such an awful student. I would also like to thank very much: TEXnician, Joseph Wright, David Carlisle and Barbara Beeton for their support and kind remarks. **Please mark that none of the people mentioned above have any idea on how I chose to implement (and typeset) the code inside the package, but they nevertheless provided many useful tips and tools for me to develop this. They cannot be blamed for anything that can be wrong in this package.**

11 License and contacts

The package is maintained by Daniele Ratti.
Email: `ilfuria+tya at gmail dot com`
Repository: `https://github.com/ilFuria/typoaid/tree/master`

References

[1] R. Bringhurst, *The elements of typographic style*, version 4.1, 2015.

[2] J. Felici, *The complete manual of typography*, second edition, 2012.

Index