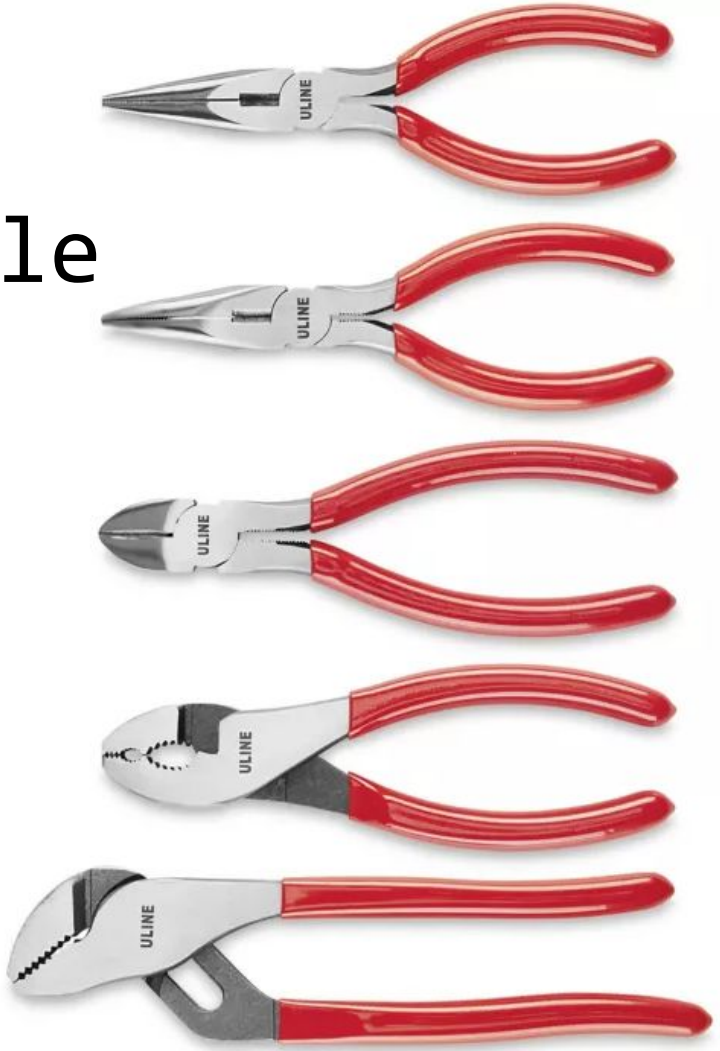
# Some Font Tools…

```
$ varLib.interpolatable
$ halfkern
$ varLib.avarPlanner
$ ttLib.scaleUpem
$ pens.svgPathPen
```

TypeLab @ Typographics 2024
June 16, 2024. NYC
Marianna Paszkowska & Behdad Esfahbod



*Give me six hours to chop down a tree and I will spend the first four sharpening the axe.*

Abraham Lincoln

# How we can use tools in our workflows

Tools can be used in any phase of the font development project:

- Deciding & Estimating
- Planning
- Development (Design and Engineering)
- Testing and Quality Assurance
- Distribution

How:

- On the sources and binaries
- Continuous integration on the repository
- Batch processing
- Locally

# varLib.interpolatable

# Testing interpolation

What are we looking for:

- Finding outline bugs, example: kinks
- Finding errors in glyph topology
- Finding inconsistent component positioning
- Finding issues with harmonisation
- Finding issues in feature variation / interpolation jump glitches
- Easy glyph identification
- Visual preview
- Easy to locate the problem in the design space
- It should be shareable

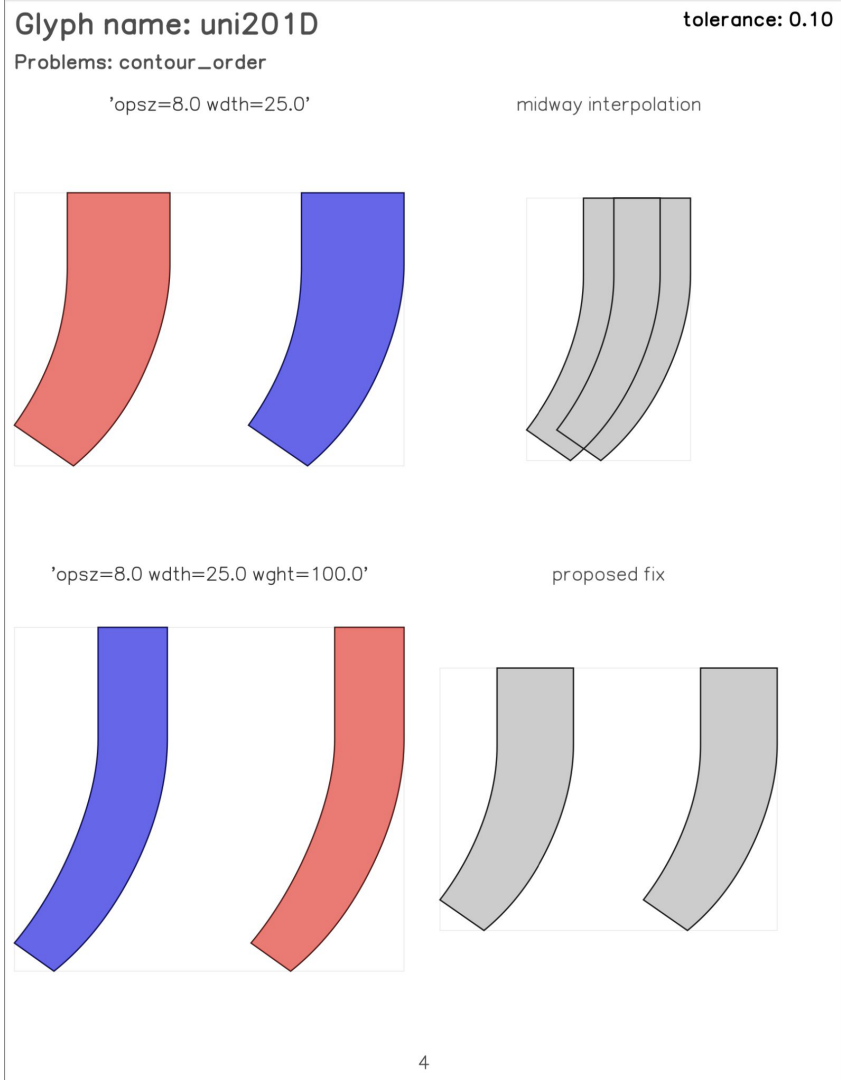
# `$ fonttools varLib.interpolatable`

Works on:

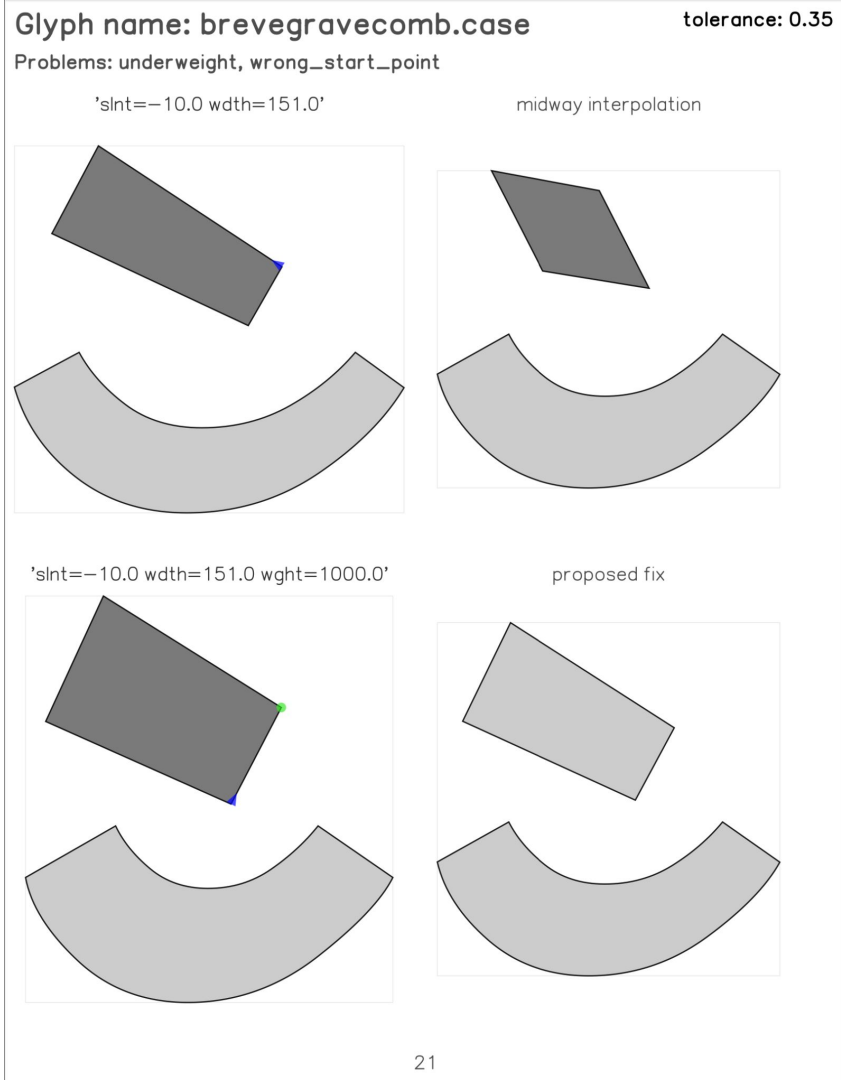
- `.glyphs` Glyphs source
- `.designspace` UFO+DesignSpace source
- `.ttf/.otf` Variable font

```
$ fonttools varLib.interpolatable \
            RobotoFlex.ttf \
            --pdf out.pdf
```

# Wrong contour order



# Wrong start point



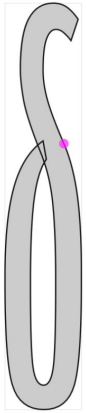
# Kink

Glyph name: uni03B4

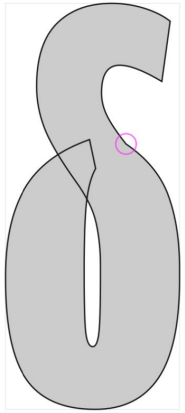
tolerance: 0.67

Problems: kink

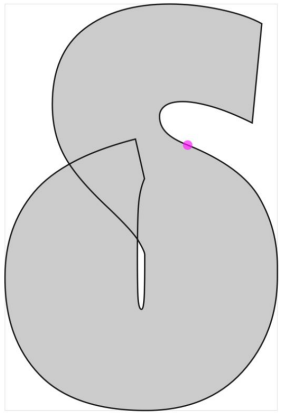
'opsz=144.0 wdth=25.0'



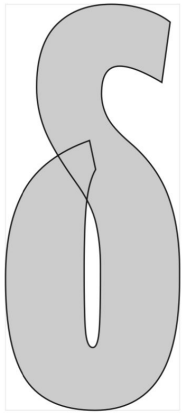
midway interpolation



'opsz=144.0 wdth=25.0 wght=1000.0'



proposed fix



# Underweight

Glyph name: diagonalbaro

tolerance: 0.60

Problems: underweight

'GRAD=−200.0 opsz=144.0 wght=100.0'

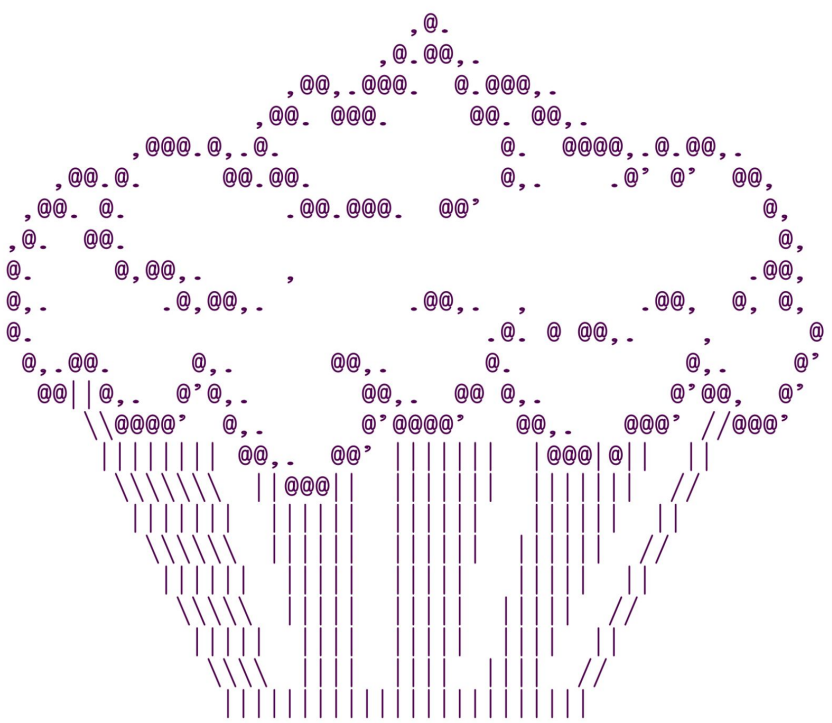
midway interpolation

'GRAD=−200.0 opsz=144.0 wdth=25.0 wght=100.0'

```
$ fonttools varLib.interpolatable \
            RobotoFlex.ttf \
            --pdf out.pdf \
            --glyphs uni0031
```

# All good

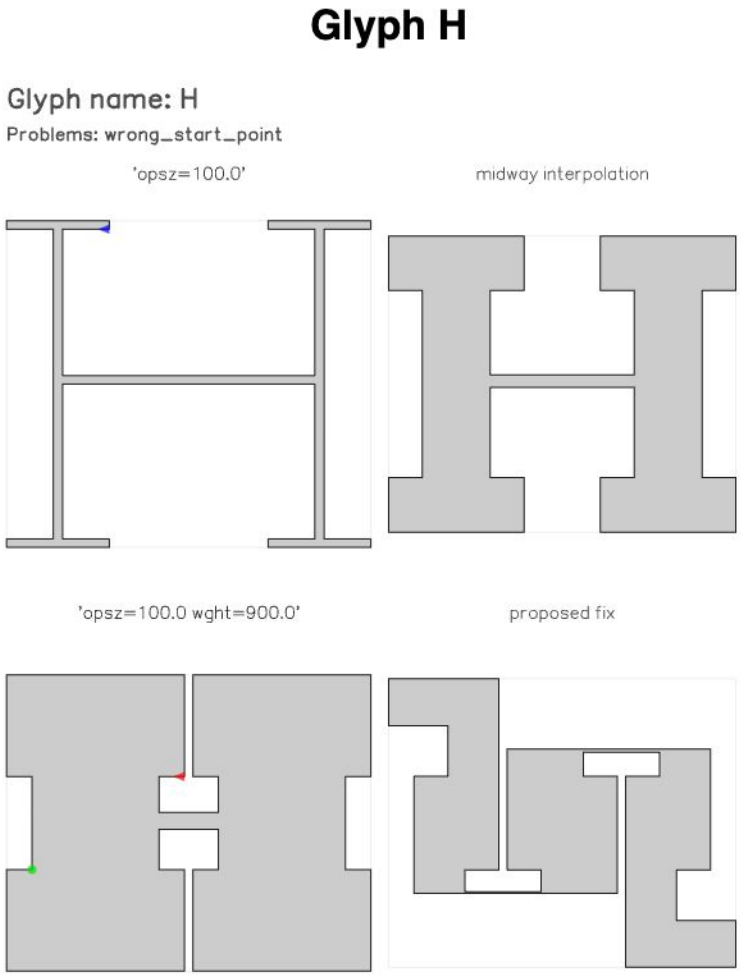
Your font's good! Have a cupcake...



# false pos·i·tive

*noun*

a test result which incorrectly indicates that a particular condition or attribute is present.



# false neg·a·tive

*noun*

a test result which incorrectly indicates that a particular condition or attribute is absent.

# Testing interpolation

How was I reviewing the interpolation until the varlib.interpolatable update:

- In sources
  - scripts and plugins for font editors
- In a compiled font:
  - Manually reviews (TypeRoof https://videoproof.graphicore.de/ for animating)
  - varLib.interpolatable as plain terminal output

*Go away or I will replace you with a very small shell script.*


ThinkGeek

halfkern

# halfkern

Project management tool for defining kerning target:

- How do I define my minimum viable kerning considering the specific design of this typeface?
- How do I define my minimum viable kerning for the target language?
- How do I test if I really have a good kerning support for my target languages?

# halfkern

Existing open source materials / tools that I had known about at the time

- Scripts for font editors: https://github.com/mekkablue/Glyphs-Scripts/tree/master/Kerning
- BubbleKern: https://github.com/Tosche/BubbleKern
- collidoscope: https://github.com/googlefonts/collidoscope
- many more...

# `halfkern`

Initial ideas for the tests:

- Kerning strings: https://github.com/kosbarts/Basic_Latin_minimum_kerning_pairs

Issues:

- Wide language support is lacking

# halfkern

**Rainer Erich Scheichelbauer** Developer Nov 2020

I have this line in *Preferences* > *Sample Strings:*

```
nď, ď. ď: ď; ď! ď? (ď) [ď] {ď} ‘ď’ „ď“ ‚ď‘ ď*\nnďa ďá ďh ďk ďl ďm ď
```

I derived it from a text analysis of a bunch of Czech and Slovak books I found online.

2

7 / 8
Nov 2020

Nov 2020

# `halfkern`

Initial ideas:

- Kerning strings
- Commonly kerned pairs: kerning frequencies
- Machine learning

# halfkern

https://typedrawers.com/discussion/2707/most-common-kern-pairs

## Most common kern pairs

Simon Cozens Posts: 727
May 2018

@Wei Huang asked me on Twitter whether, in the course of my messing about with automated kerning, I had compiled a table of the most common kern pairs. I replied that I hadn't but that it wouldn't be hard to do. So I did it. Based on a set of 514 fonts, here is a file with the top ten-thousand most common kern pairs and how many of those 514 fonts implement them.

The top fifty are:

| | | |
|---|---|---|
| 504 | L | T |
| 499 | L | V |
| 495 | L | Y |

# halfkern

```
116 Uslash x
116 Ohungarumlaut quotesinglbase
116 Ohungarumlaut quotedblbase
116 Lslash Gbreve
116 Ldot wgrave
116 Ldot wdieresis
116 Ldot wcircumflex
116 Ldot wacute
116 Ldot w
116 Ldot uni021A
116 Ldot Ygrave
116 Ldot Ydieresis
116 Ldot Yacute
116 Ldot Y
116 Ldot V
116 Ldot Tcaron
116 Ldot T
116 Lcommaaccent ycircumflex
116 Lcommaaccent quoteleft
116 Lcommaaccent quotedblleft
116 K fl
```

```
123 O ae
123 N A
123 M c
123 Lcaron Wcir
```

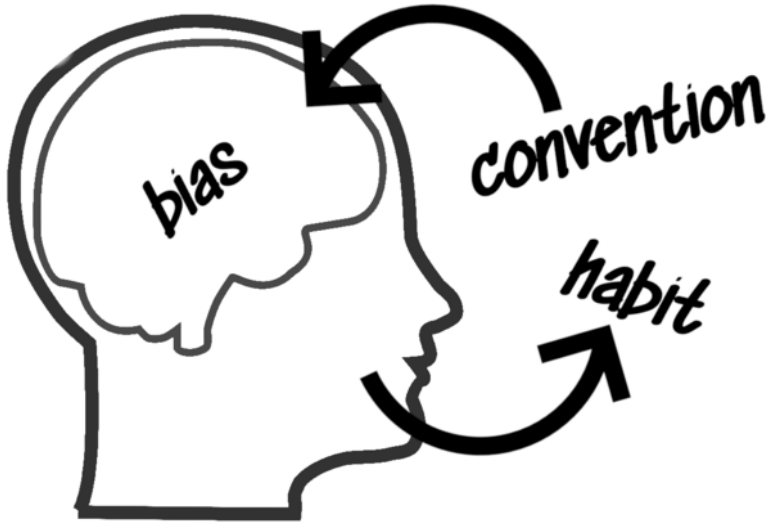
```
145 Ohungarumlau
145 O Tcaron
145 M Y
145 Lcaron y
145 L comma
```

```
96 O M
96 M d
96 M Omacron
96 M Ohungarumlaut
```

# data bias

*noun*

systematic error of a statistical result due to distortion during sampling, data collection, or data analysis.



# halfkern

Initial ideas:

- ~~Kerning strings~~ (coverage issues / coverage bias)
- ~~Commonly kerned pairs: kerning frequencies~~ (selection bias)
- ~~Machine learning~~ (bias risk)

# halfkern

Make it design specific and think like a type designer:

- “*Badness is a measure of how bad a line looks.*” – Donald Knuth's book "TEX and METAFONT: New Directions in Typesetting"

- Knuth–Plass line-breaking algorithm and badness score

  *The algorithm works by dividing the text into a stream of three kinds of objects: boxes, which are non-resizable chunks of content, glue, which are flexible, resizeable elements, and penalties, which represent places where breaking is undesirable (or, if negative, desirable).[2] The loss function, known as "badness", is defined in terms of the deformation of the glue elements, and any extra penalties incurred through line breaking.* – from Wikipedia

# `halfkern`

Make it design specific and think like a type designer:

- Our badness is a deviation of the spacing from the spacing of control character strings.
- Control characters are those which the type designer would use for spacing.
- We assume that the spacing is already done and correct.

# `halfkern`

Text sample goal

- Ideally, a corpus of at least 1 million words is suggested to achieve a reasonably representative set of bigram frequencies. For example that was the basis for Brown Corpus (500 samples of English from various genres).
- But… we need to be practical.

# halfkern

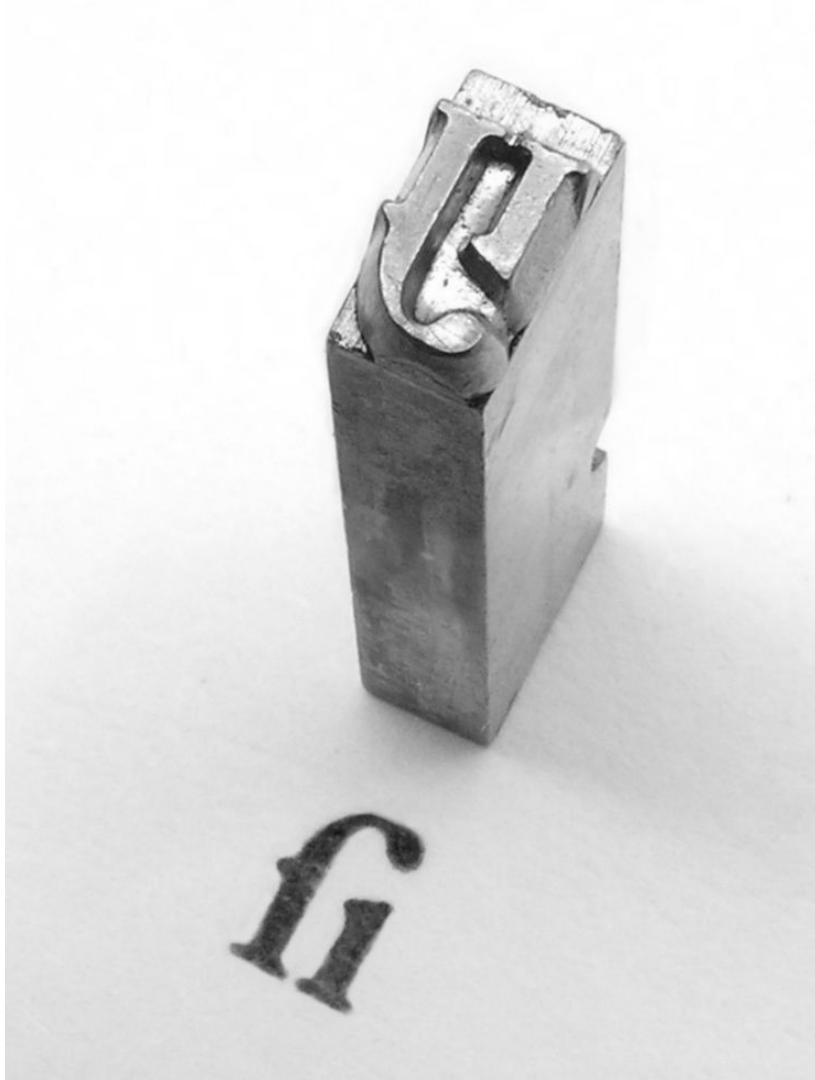
Implementation

- New half-assed autokerner in town (use with caution)
- Useful for kern auditing, to flag pairs that might need attention
- Using a language dictionary to find kern pairs
- Geometric spacing, different algorithms
- Compares to existing font kern and report discrepancies

# `halfkern`

Algorithm:

- Two envelope profiles:
  - Signed Distance Field (`sdf`)
  - Gaussian Blur (`gaussian`)
- Two reduction methods:
  - Summation (`sum`)
  - Maximum (`max`)
- Four combinations
- Defaults to `sdf sum`
- Calibrates based on `ll`, `nn`, `oo`
- Negative kern to *half* of the measurement



```
$ python3 halfkern/kern_pair.py \
          NotoSans-VF.ttf \
          en_US.dic \
          --letters-only \
          --pdf out.pdf
```

github.com/behdad/halfkern

# keming

*noun*

the result of improper kerning.

# Kern River

California
United States

# halfkern

ToDo:

- Add support for more scripts than Latin
- Improve testing for punctuation (overkerns)
- Suggest kerning groups
- Add support for connecting scripts
- Find better language data sets
  - We used LibreOffice dictionaries. They contain around 50.000 up to 300.000 words, depending on the language.
  - Data is of a low quality. Why? Data is noisy. There are words that in the natural language would be outliers, but are frequently present in our data.

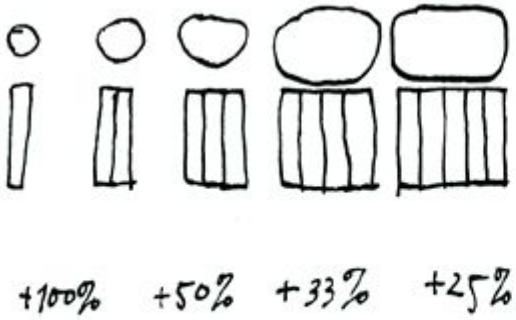
varLib.avarPlanner

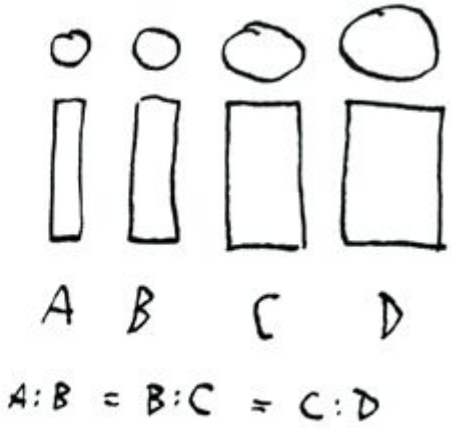
# `$ fonttools varLib.avarPlanner`

Plan your standard axes:

- **Weight** Calculate curve based on ink color *geometric* progression.
- **Width** Calculate curve based on true width ratios.
- **Slant** Correctly implement “degrees” unit.
- **Optical Size** Calculate smooth curve based on ink color progression.

# Weight

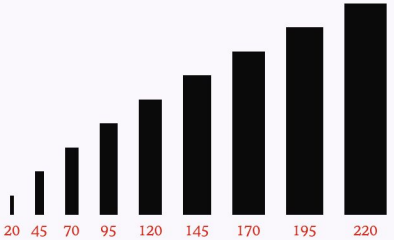


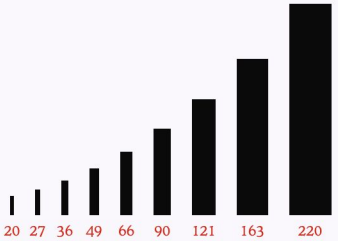

lucasfonts.com/learn/interpolation-theory

diacritics.club/family-steps

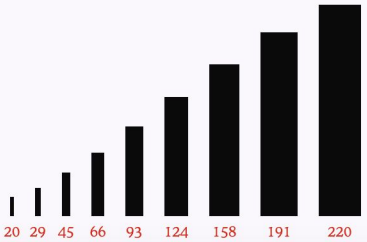
LINEAR PROGRESSION




LUC(AS) PROGRESSION




IMPALLARI PROGRESSION




Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

你好世界 你好世界

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

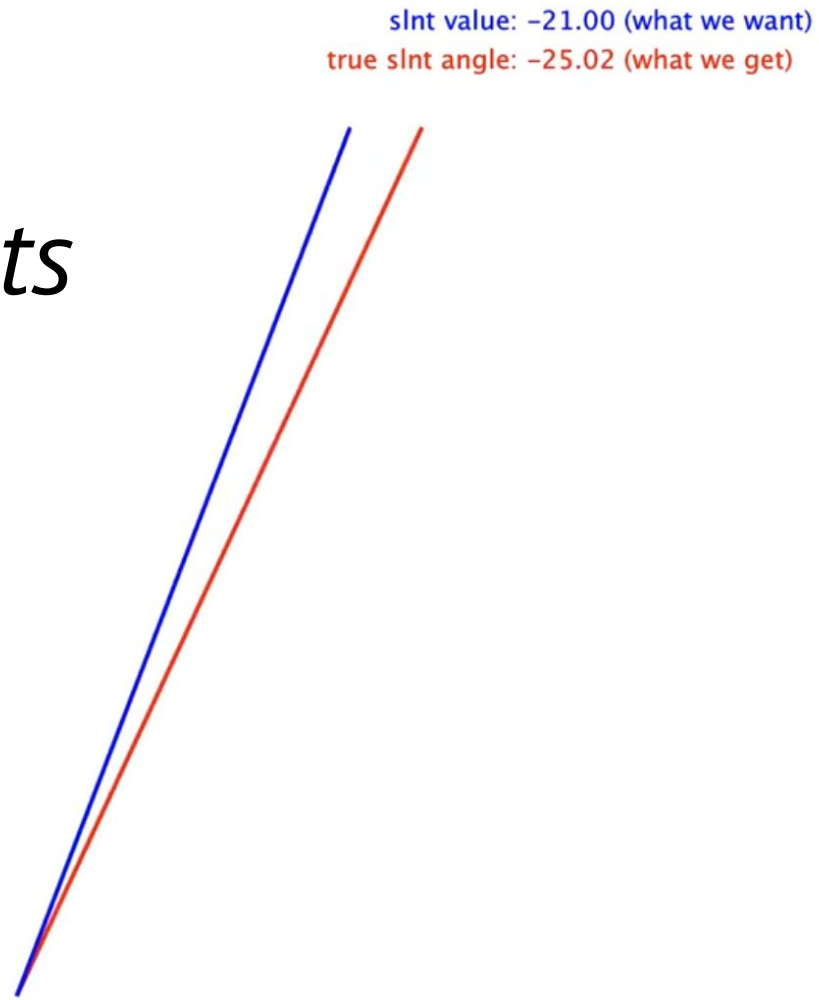
السلام عليكم السلام عليكم

السلام عليكم السلام عليكم

# Slant

theartnewspaper.com

# *Hey 'slnt' axis, trigonometry wants a chat*




twitter.com/axis_praxis/status/1083183293368029184

# True Italics Café

Cape Town
South Africa

# Optical Size

therussianstore.com

Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World
Hello World Hello World

Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

ttLib.scaleUpem

# `$ fonttools ttLib.scaleUpem`

Useful for:

- Testing different Units-per-EM values
- Merging fonts using fonttools merge
- File-size optimization mechanism

# pens.svgPathPen

```
$ fonttools pens.svgPathPen \
            RobotoFlex.ttf S
```

```
<?xml version="1.0" encoding="UTF-8"?>
<svg width="1220" height="2400" xmlns="http://www.w3.org/2000/svg">
<g transform="translate(0 1900) scale(1 -1)"><path d="M90 398Q90 200 235.0
85.0Q380 -30 620 -30Q860 -30 1000.0 85.0Q1140 200 1140 380Q1140 560 1034.0
659.0Q928 758 694 824Q460 890 385.0 945.0Q310 1000 310 1095Q310 1200 383.0
1261.0Q456 1322 610 1322Q764 1322 842.0 1251.0Q920 1180 920
1064V1052H1112V1064Q1112 1248 976.0 1367.0Q840 1486 610 1486Q380 1486
250.0 1373.0Q120 1260 120 1090Q120 920 219.0 827.5Q318 735 556 666Q785 600
867.5 540.0Q950 480 950 380Q950 270 865.0 202.0Q780 134 620 134Q460 134
371.0 200.0Q282 266 282 398V410H90Z"/></g>
</svg>
```

cffLib.CFFToCFF2
cffLib.CFF2ToCFF

```
$ fonttools cffLib.CFFToCFF2
$ fonttools cffLib.CFF2ToCFF
```

- New tools
- `CFF2ToCFF` used for instantiating a CFF2 variable font to a CFF
- Still buggy
- Help needed

*Life is a series of building, testing, changing and iterating.*

Lauren Mosenthal

*Image source: laurenmosenthal.com*

qq

# Also, btw...

*Girls should be encouraged to pursue their dreams and break the stereotypes.*

Margaret Hamilton

*We need to understand that if we all work on inclusion together, it's going to be faster, broader, better, and more thorough than anything we can do on our own.*

Ellen Pao

*Thank you!*