Improving the speed and accuracy when fitting flexible parametric survival models on the log hazard scale.

Paul C Lambert[1,2]

[1]Cancer Registry of Norway, Norwegian Institute of Public Health, Norway
[2]Medical Epidemiology and Biostatistics, Karolinska Institutet, Stockholm, Sweden

2024 Northern European Stata Conference, Oslo, Norway
10th September 2024

Introduction

- Yesterday, I taught a pre-conference course on flexible parametric survival models using `stpm3`.
- I concentrated on models on the log cumulative hazard scale.
- Sometimes, there is a need to fit models on the log hazard scale.
- However, there are many more computational challenges for models on the log hazard scale.
- This talk will describe how I dealt with these computational challenges.

Flexible parametric survival models (FPSMs)

- FPSMs use spline functions to model the effect of time.
- Models can be fitted on different scales, but the most common is the log cumulative hazard scale.

$$\ln[H(t|\mathbf{x}_i)] = s\left(\ln(t)|\gamma, \mathbf{k}_0\right) + \mathbf{x}_i\boldsymbol{\beta}$$

- On this scale we can derive the hazard and cumulative hazard functions analytically, which are fed into the log-likelihood.
- Fitting these models is very quick.

Cancer Registry of Norway
Karolinska Institutet

FPSMs on the log hazard scale

- We change from $H(t)$ to $h(t)$.

$$\ln\left[h(t)\right] = s(\ln(t)|\gamma) + \mathbf{x}\beta$$

- We are now on the log hazard scale. This is useful for,
  - Modelling SMRs/SIRs.
  - Multiple time-scales.
  - Sometimes with multiple time-dependent effects.
  - When apply constraints (e.g. constrain HRs to be proportional after certain time.)

FPSMs on the log hazard scale

- The change from uppercase $H(t)$ to lowercase $h(t)$ complicates things.

$$\ln\left[h(t)\right] = s(\ln(t)|\gamma) + \mathbf{x}\boldsymbol{\beta}$$

- The log-Likelihood is

$$\ell_i = d_i \ln\left[h(t_i)\right] + \int_{t_{0i}}^{t_i} h(u)du$$

$$\ell_i = d_i \ln\left[s(\ln(t)|\gamma) + \mathbf{x}\boldsymbol{\beta}\right] + \int_{t_{0i}}^{t_i} \exp(s(\ln(u)|\gamma) + \mathbf{x}\boldsymbol{\beta})du$$

- Not analytically tractable, so need to use numerical integration.

Splines on the log hazard scale in Stata - history

`stgenreg` General software could write out any hazard function. Integration used Gauss-Legendre Quadrature

`strcs` Models on log hazard function using restricted cubic splines. Integration using Gauss-Legendre Quadrature, but used analytic integrals before first and after last knot.

`merlin` Integration used Gauss-Legendre Quadrature.

Numerical intergration

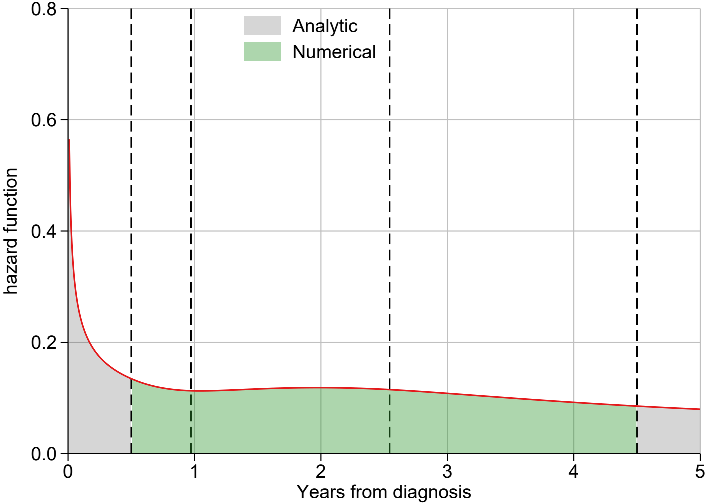
$$\ell_i = d_i \ln\left[h(t_i)\right] + \int_{t_{0i}}^{t_i} h(u)du$$

- Need to numerically integrate for all N individuals
- This will be every time likelihood function called
  - and for gradient function
  - and for Hessian matrix
- Can use Gaussian quadrature (with K nodes).

$$\int_{t_{0i}}^{t_i} h(u)du \approx \frac{t_i - t_{0i}}{2} \sum_{k=1}^{K} w_k h\left(\frac{t_i - t_{0i}}{2}\epsilon_i + \frac{t_i + t_{0i}}{2}\right)$$

Three part integration (initially used in `strcs`)

Boundary knots moved away from boundaries for illustration



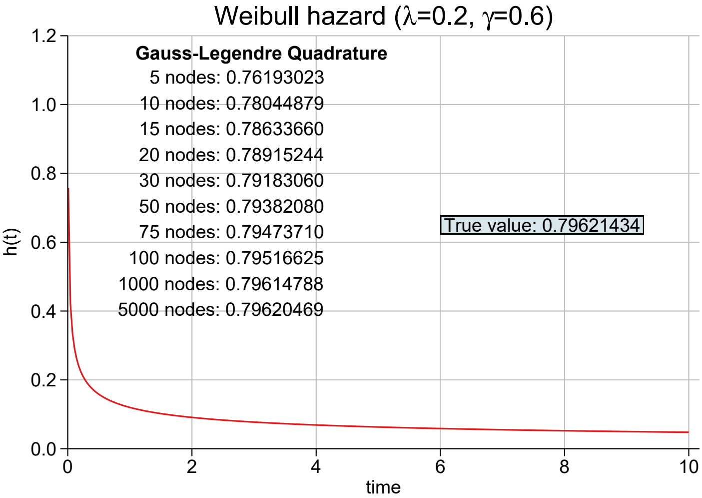
tanh-sinh quadrature

- Gauss legendre quadrature performs poorly when there is a singularity at $t = 0$.
- This means that you need many nodes, which becomes much more computationally intensive.
- I recently came accross the tanh-sinh quadrature method[**?**].
- It stated on Wikepedia that,

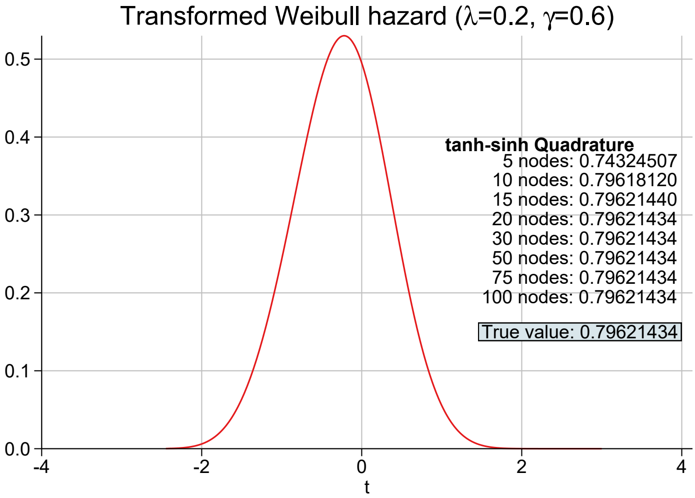
It is especially applied where singularities or infinite derivatives exist at one or both endpoints

- So, I decided to explore.

Numerical integration of Weibull 1



Numerical integration of Weibull 2



tanh-sinh quadrature is the default method in stpm3

```
. stpm3 , scale(lnhazard) df(4) nolog
                                                   Number of obs =  6,242
                                                   Wald chi2(4)  = 822.75
Log likelihood = -9345.3029                        Prob > chi2   = 0.0000

------------------------------------------------------------------------------
             | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
time         |
        _ns1 |   8.521712   .4113931    20.71   0.000     7.715397    9.328028
        _ns2 |  -2.390542   .1994054   -11.99   0.000    -2.781369   -1.999714
        _ns3 |    .674227   .1012383     6.66   0.000     .4758036    .8726504
        _ns4 |   .7512846   .2063079     3.64   0.000     .3469285    1.155641
       _cons |  -2.890098    .102914   -28.08   0.000    -3.091805    -2.68839
------------------------------------------------------------------------------
Quadrature method: tanh-sinh with 30 nodes.
Analytical integration before first and after last knot.
```

- You can explore combinations of 3-part integration vs all numerical integration and Gauss Legendre vs tanh-sinh integration by specifying suboptions for `integoptions()` in `stpm3`.

Cancer Registry of Norway   Karolinska Institutet

Comparing methods.

Gauss Legendre: all numerical integration

```
. est tab n10 n20 n25 n30 n50 n1000, stats(ll) b(%6.5f) stfmt(%9.3f)
```

Variable	n10	n20	n25	n30	n50	n1000
_ns1	15.77737	12.55816	11.83925	11.41981	10.76840	10.24086
_ns2	-5.31281	-3.34123	-2.85658	-2.56926	-2.11758	-1.74656
_ns3	1.39246	1.22665	1.15463	1.10849	1.03205	0.96639
_cons	-3.39790	-3.35931	-3.34217	-3.33122	-3.31315	-3.29773
ll	-1203.823	-1218.122	-1219.871	-1220.776	-1222.052	-1222.947

tanh-sinh: 3-part integration

```
. est tab n10_tanh3  n20_tanh3 n25_tanh3 n30_tanh3 n50_tanh3 n1000_tanh3, stats(ll) b(%6.5f) stfmt(%9
```

Variable	n10_tanh3	n20_tanh3	n25_tanh3	n30_tanh3	n50_tanh3	n1000_t~3
_ns1	10.26746	10.22919	10.23109	10.22917	10.22974	10.22975
_ns2	-1.76273	-1.73814	-1.73938	-1.73815	-1.73851	-1.73852
_ns3	1.00457	0.96459	0.96569	0.96449	0.96484	0.96484
_cons	-3.31129	-3.29728	-3.29766	-3.29725	-3.29737	-3.29737
ll	-1223.109	-1222.958	-1222.959	-1222.959	-1222.959	-1222.959

Using python and automatic differentation

- For large datasets numerical integration is slow.
- I put some effort into speeding up computation times.
  - Likelihood evaluator in Mata
  - Derive gradient and Hessians
  - Some other computational improvements.
- However, could I make it faster by calling Python to do the heavy computation?

Faster models with large data sets

- For large datasets can send heavy computations to Python.
- Just add `python` option.
- The `mlad` program is used to maximize the likelihood.
- Calls `mlad`
  - Maximum Likelihood using Automatic Differentiation.
  - Calls Python Jax module.
  - Scores and Hessian automatically created
  - Just-In-Time (JIT) compilation
  - Efficient use of multiple processors.

```
. stpm3 i.dep, scale(lnhazard) df(5)
. stpm3 i.dep, scale(lnhazard) df(5) python
```

See `mlad` talk at Stata Conference

https://www.stata.com/meeting/us21/slides/US21_Lambert.pdf

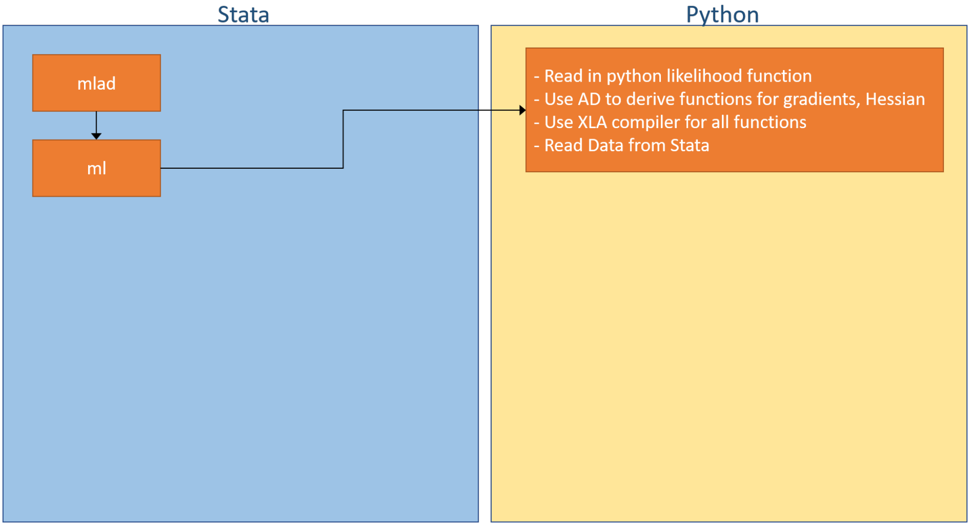
A alternative optimizer, `mlad`

- Rather than a Stata program to define the likelihood the user needs to write a Python function.
- Automatic differentiation is used so the gradient and Hessian functions are calculated automatically using Jax.
- Likelihood, gradient and Hessian functions are compiled so fast and can make use of multiple processors.
- Makes use of Stata's `ml` command for setup, updating parameters and assessing convergence.
- All results are returned in Stata in standard `ml` format, so standard post-estimation tools are available.

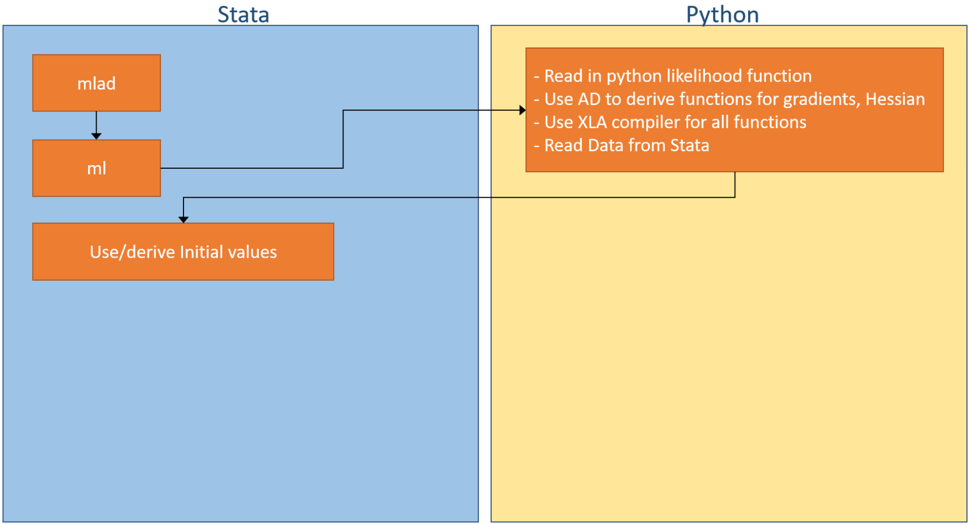
A alternative optimizer, `mlad`

- Rather than a Stata program to define the likelihood the user needs to write a Python function.
- Automatic differentiation is used so the gradient and Hessian functions are calculated automatically using Jax.
- Likelihood, gradient and Hessian functions are compiled so fast and can make use of multiple processors.
- Makes use of Stata's `ml` command for setup, updating parameters and assessing convergence.
- All results are returned in Stata in standard `ml` format, so standard post-estimation tools are available.

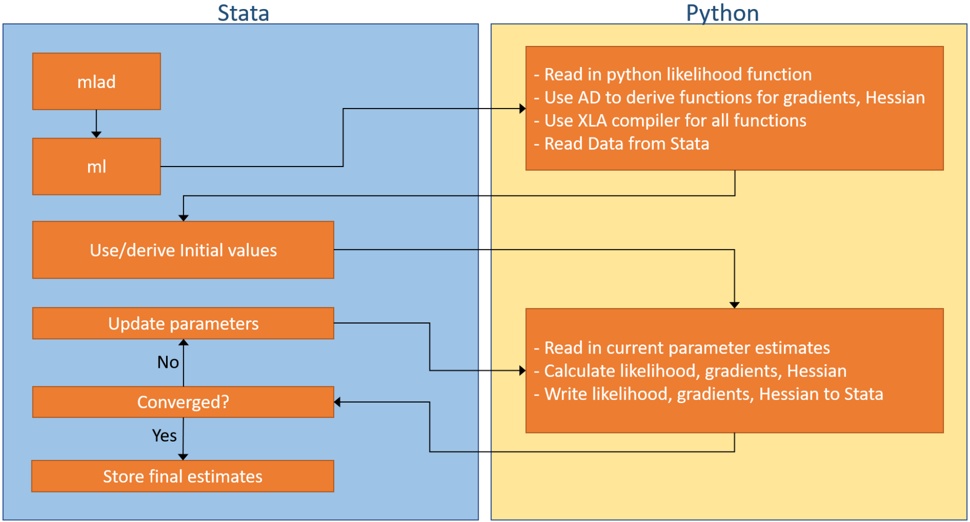
What is mlad doing?



What is mlad doing?



What is `mlad` doing?



Cancer Registry of Norway
Karolinska Institutet

Times in seconds

	Sample Size	
	500,000	1,000,000
3 part integration		
`strcs`	2930	4807

Times in seconds

		Sample Size			
		500,000		1,000,000	
3 part integration					
	`strcs`	2930		4807	
	`stpm3`	493	(83.1%)	981	(79.6%)

Times in seconds

	Sample Size			
	500,000		1,000,000	
3 part integration				
`strcs`	2930		4807	
`stpm3`	493	(83.1%)	981	(79.6%)
`stpm3 (python option)`	46	(98.4%)	83	(98.3%)

Cancer Registry of Norway Karolinska Institutet

Times in seconds

	Sample Size			
	500,000		1,000,000	
3 part integration				
`strcs`	2930		4807	
`stpm3`	493	(83.1%)	981	(79.6%)
`stpm3 (python option)`	46	(98.4%)	83	(98.3%)
All numerical integration				

Times in seconds

	Sample Size			
	500,000		1,000,000	
3 part integration				
`strcs`	2930		4807	
`stpm3`	493	(83.1%)	981	(79.6%)
`stpm3 (python option)`	46	(98.4%)	83	(98.3%)
All numerical integration				
`stmerlin`	1950		3996	

Times in seconds

	Sample Size			
	500,000		1,000,000	
3 part integration				
`strcs`	2930		4807	
`stpm3`	493	(83.1%)	981	(79.6%)
`stpm3 (python option)`	46	(98.4%)	83	(98.3%)
All numerical integration				
`stmerlin`	1950		3996	
`stpm3`	464	(76.2%)	917	(77.0%)

Cancer Registry of Norway Karolinska Institutet

Times in seconds

	Sample Size 500,000		1,000,000	
3 part integration				
`strcs`	2930		4807	
`stpm3`	493	(83.1%)	981	(79.6%)
`stpm3 (python option)`	46	(98.4%)	83	(98.3%)
All numerical integration				
`stmerlin`	1950		3996	
`stpm3`	464	(76.2%)	917	(77.0%)
`stpm3 (python option)`	34	(98.3%)	69	(98.3%)

Note: Using State BE (only using 1 core) on machine with 16 cores.

Summary

- It is now much faster to fit FPSMs on the log hazard scale.
- This makes their use feasible in large datasets.
- Choice of quadrature method can lead to important improvements in accuracy of results.
  - tanh-sinh quadrature and 3-part integration lead to big improvements.
- Use of `python` option is simple for the user.
  - Requires installation of various python packages.