Python package applying survival tree for left truncated and right censored data (LTRC)

Tomer Weiss and Tamirat Etana
Affiliation

Tree methods are a popular class of non parametric methods for analyzing data. One extension of the basic tree methodology is the survival tree,which applies recursive partitioning to censored survival data. There are several existing survival tree methods in the literature,which are mainly designed for right-censored data in python. We propose a new survival trees package for left-truncated and right-censored (LTRC) data based on the existing sklearn implementation of regression trees, as did in R. We'll compare our implementation to **(author?)** (1) implementation in R via the same simulations they preformed and on a real world data they used, and discuss the similarities and dissimilarities. For an introduction of the matter at hand and theoretical details we refer again to **(author?)** (1)

LTRCART

We tried to implement an equivalent to the "LTRCART" function from the "LTRCtrees" from R as much as possible. As said by **(author?)**, three steps are needed to implement this method. The first is estimation of the cumulative function $\Lambda_0(t)$ based on all of the LTRC data, $(L_i, R_i, \delta_i, x_i)$ where L_i is left truncation time, $R_i$is the observed survival time/censoring time, δ_i is the event indicator and x_i is the matrix of covariates for individual i. The second step is computing the "exposure time" for observation i by $\hat{\Lambda}_o(R_i) - \hat{\Lambda}_o(L_i)$ based on $\hat{\Lambda}_0(t)$ estimated from the previous step. The final and third step is to fit a "Poisson regression tree" by treating the $\hat{\Lambda}_o(R_i) - \hat{\Lambda}_o(L_i)$ as the new observed times t_i and δ_i as the new c_i and thus extending the proposed survival tree algorithm by **(author?)** (2), for LTRC data. We followed those steps in python while looking at the R code of **(author?)** for sanity check. A major difference in our implementation is that we did not fit the tree using the pairs of $(\hat{\Lambda}_o(R_i) - \hat{\Lambda}_o(L_i), \delta_i)$, we supplied only $\hat{\Lambda}_o(R_i) - \hat{\Lambda}_o(L_i)$, due to some technical difficulties we didn't figure out yet in the "sklearn" implementation of the cart "tree". This obviously will hamper our results, so take that in mind when proceeding.

Data and Simulations structure

We tried to reproduce the results from **(author?)**. Specifically we used our python implementation of LTRCART to try and reproduce the upper panel of Figure 10 (assay of serum free light chain data for 7874 subjects in the R package survival **(author?)** (3)) and Table 1 (recovery rates of the a simulated tree structure under different time, right censoring and left truncation distributions).

Results

Here the results we got:

We can see we did not get the same exactly the same results as in **(author?)**, but it is a really good considering the model does not see that event indicator.

For table 1 we'll show the percentage of times the actual tree is the first sub-tree (i.e., catching the real important features), as opposed to calculating the relative frequency we got the actual tree (like Fu & Simonoff). We did this because the tree implementation of "rpart" and "sklearn" are quite different (we've set the possible parameters R's defaults, but there are some that does not exist in "sklearn").

Discussion

As we mentioned above, a major difference in our implementation compared to Fu & Simonoff is the absence of the event indicator in the tree fitting process (as it should be), which obviously hampered the results presented in their paper. Despite that, we do see a relatively good results, which do show that we are in the correct route for the full implementation of LTRCART, which is good news.

Future Work

An obvious direction for future work would be to modify sklearn implementation to accept the event indication and do apply the wanted process to apply LTRCART. Another direction we already started to pursue is the implementation of LTRCIT method describes in **(author?)** (1). We tried to find (with no success) an official package implementing the conditional inference tree (CIT), and to use that as base for CIT modified to LTRC data. Nevertheless, we wrote a code (github repo for our work: RSF-Repo) as if there is an official package implementing CIT and used that accordingly.

Figure 1. fitted LTRCART on creatinine data

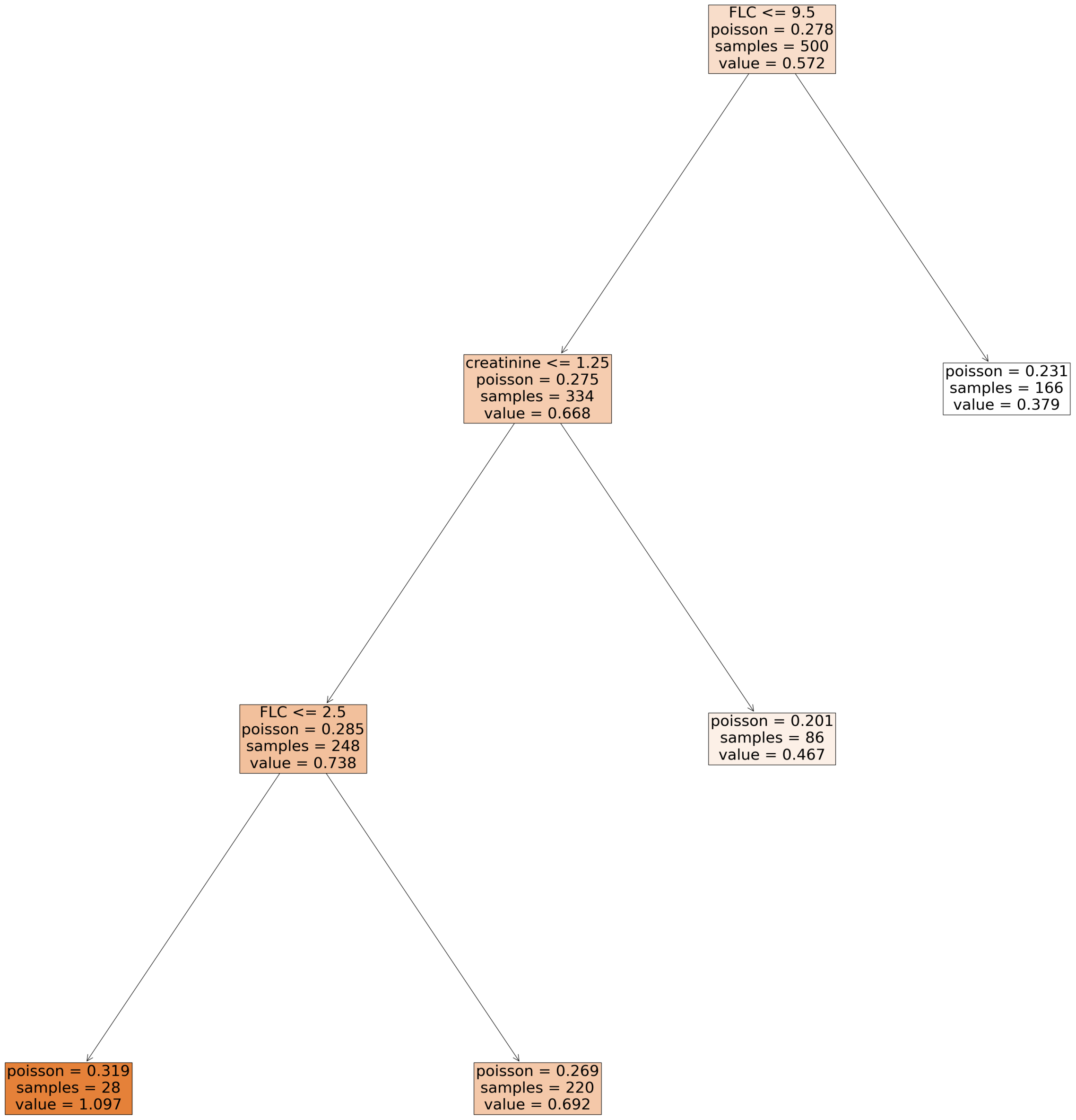

Table 1
Simulations results as in table 1 in Fu and Simonoff

N = 100						
Censor.rate	Truncation	Exponential	Weibull-I	Weibull-D	Log-normal	Bathtub
Light	$U(0,1)$	14	7	47	38	
Heavy	$U(0,1)$	19	14	51	33	
Light	$U(0,2)$	19	10	58	47	
Heavy	$U(0,2)$	16	3	56	48	
Light	$U(0,3)$	10	12	36	43	
Heavy	$U(0,3)$	17	11	43	36	
N = 300						
Censor.rate	Truncation	Exponential	Weibull-I	Weibull-D	Log-normal	Bathtub
Light	$U(0,1)$	48	45	77	77	
Heavy	$U(0,1)$	45	66	74	69	
Light	$U(0,2)$	50	49	74	78	
Heavy	$U(0,2)$	41	49	62	74	
Light	$U(0,3)$	44	44	67	75	
Heavy	$U(0,3)$	53	54	68	78	
N = 500						
Censor.rate	Truncation	Exponential	Weibull-I	Weibull-D	Log-normal	Bathtub
Light	$U(0,1)$	68	62	79	89	
Heavy	$U(0,1)$	71	72	71	82	
Light	$U(0,2)$	69	84	79	89	
Heavy	$U(0,2)$	79	77	72	85	
Light	$U(0,3)$	60	84	62	81	
Heavy	$U(0,3)$	68	73	65	85	

References

Wei Fu and Jeffrey S Simonoff. Survival trees for left-truncated and right-censored data, with application to time-varying covariate data. *Biostatistics*, 18(2):352–369, 2017.

Michael LeBlanc and John Crowley. Relative risk trees for censored survival data. *Biometrics*, pages 411–425, 1992.

Terry M Therneau and Thomas Lumley. Package âsurvivalâ. *R Top Doc*, 128(10):28–33, 2015.