ISyE 6404 CP.1 Part II
Bootstrap Method: PH Regression & Large p Small n Problems

Yuan Gao, Kevin Lee, Akshay Govindaraj,
Yijun (Emma) Wan, Peter Williams, Ruixuan Zhang

Date: 2018-11-05

Contents

Workload Distribution 2
1. Bootstrap Method 3
2. Large-p-Small-n Problems in PH-Regression (30%) 6
Publication 1: Proportional Hazards Mixed Models 6
Publication 2: Penalized Cox Regression Analysis 6
Publication 3: Machine Learning for Survival Analysis 7
Code Appendix 9
Routine: CPU Parallelized Bootstrap for PH Regression, Hand Calculations 9
Routine: HPD Interval Computations 10

Workload Distribution

Below is a description of tasks completed by each team member for this project:

Team Member	Task Description
Yuan Gao	Publication 1 Summary
Kevin Lee	Contributed to Bootstrap CI (1.645, norm, bca)
Akshay Govindaraj	Analysis of Results & Findings, Proof reading
Yijun (Emma) Wan	Publication 3 Summary
Peter Williams	Bootstrap & HPD Computation, Latex Formatting, Document Editing
Ruixuan Zhang	Publication 2 Summary

1. Bootstrap Method

Use the bootstrap method to construct a 90% pointwise confidence interval (CI) of betacoefficient(s) based on the PH-regression parameter-estimate(s). With the 10000 bootstrap samples you have the entire distribution of beta-estimate(s). Thus, you can use the methods below to construct CIs.

i) Calculate standard error (denoted as s.e.(beta-est)) of a beta-estimate from the 1000 bootstrapped beta-estimates. Use [beta-est. +- 1.645 · s.e.(beta-est.)] as the largesample CI.

Use the bootstrap standard errors for our confidence interval for the "age" effect estimate and the bootstrap mean given alongside the standard error estimates. This leads us to the following confidence intervals through the above equation [beta-est. +/- 1.645*s.e.(beta-est.):

Table 1: PH Regression: Bootstrap Results (B = 10,000)

Variable	Estimate (Mean)	Std Error (1.645 * Mean SE)
age	0.001	[-0.014,0.017]
sex	-1.482	[-2.069,-0.895]
frail	1.322	[0.968,1.677]

The Proportional Hazards regression model works best for covariates that are not a function of time because the exponential distribution is 'memoryless'. As seen from the above results, age doesn't make a good variable in the proportional hazards model since age changes with time. We would expect the coefficient to be negative for infants and positive for aging populations, but since the sample population includes a diverse age group, the coefficient is nearly zero.

ii) If the R-program provides 90% confidence interval from either normal or chi-square approximation based on the large-sample theory, it would be worthwhile to compare the results against the ones given above to examine what R-program is doing.

The R program *boot.ci()* does indeed have a function that allows us to create a 90% confidence interval from normal approximation:

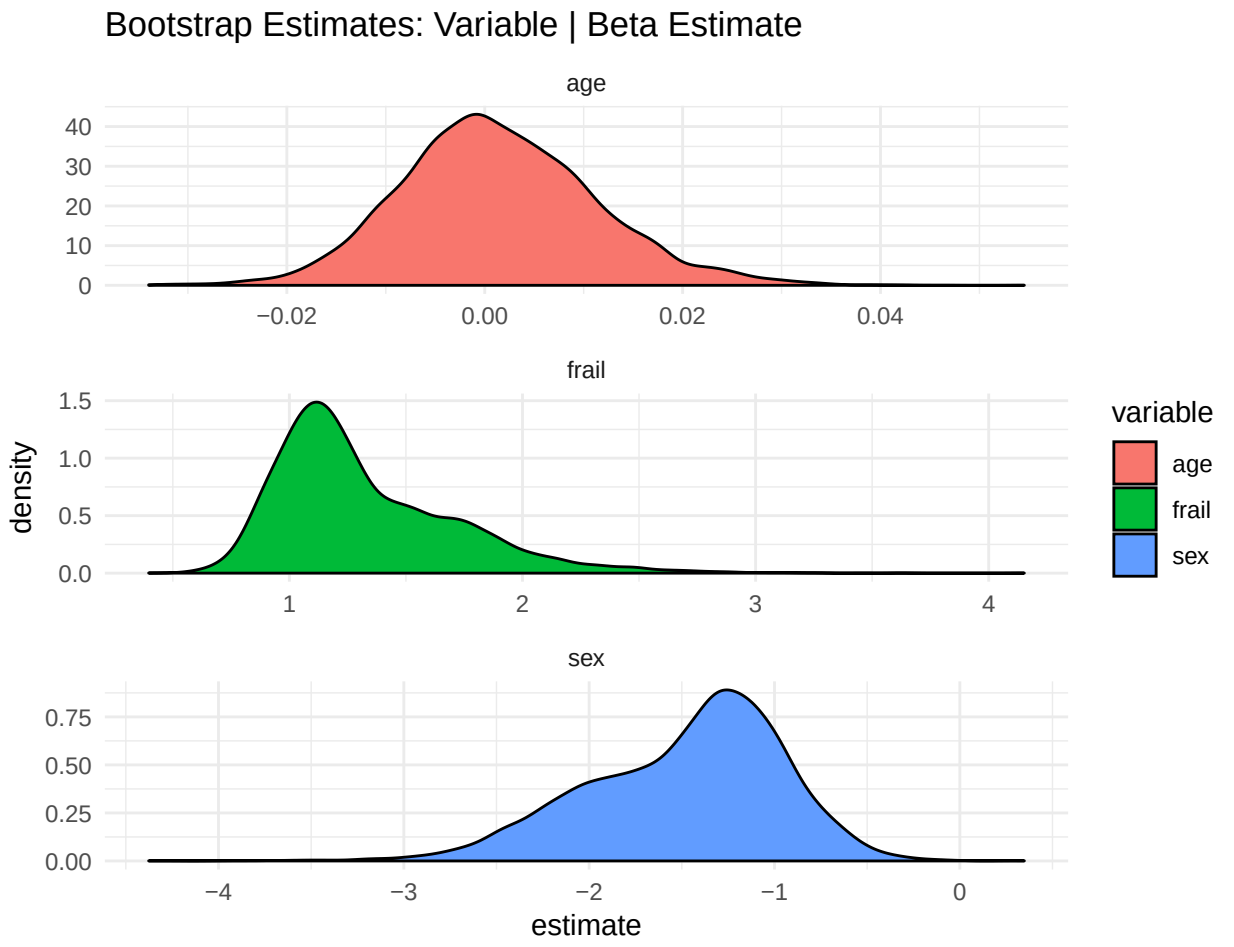
Table 2: Bootstrap Results Comparison (B = 10,000)

Variable	Interval: boot Package	Interval: Hand Calculation
age	[-0.021,0.012]	[-0.014,0.017]
sex	[-1.955,-0.235]	[-2.069,-0.895]
frail	[0.368,1.61]	[0.968,1.677]

This gives us a confidence interval that is quite similar to the confidence interval calculated from part (i), though the "normal approximation" interval is slightly narrow compared to the confidence interval in (i). This leads one to believe there is another factor involved in the R program's interval calculation, as the 1.645 used in (i) is based off of normal approximations using z.

iii) If the distribution of the bootstrap samples of beta-estimate is not symmetric likenormal, use 10000 · 0.05 = 500th lower and upper percentiles as the second large sample CI. This is the so-called Percentile-Bootstrap-CI.

A quick visualization of the bootstrap estimates, indicates approximate symmetry for the age effect estimates, but the sex and frail effect estimates are clearly not symmetric:



The percentile and "normal" bootstrap confidence intervals are formulated differently. The percentile bootstrap is dependent only upon the quantiles of the bootstrap samples, with no real "mean estimate" used or idea of the shape of the distribution without further exploration through graphing/tabulation. With the normal assumption, one used the estimate mean as a center and the spread of the bootstrap estimates through standard error. In accounting for tails, the normal approximation interval will include a bias correction component, which may be why there is a difference between the calculations of confidence interval through R package implementation (boot.ci()) and hand calculation.

iv) Apply the Bias Correction (BC) Method to obtain BC-Bootstrap-CI. See Section 15.3 textbook for the details of the BC-method.

The Bias Correction bootstrap method is essentially an extension of the above percentile bootstrap method. A prominent issue in utilizing the percentile bootstrap method for confidence intervals is the fact that the estimate from the original data is not used (only the quantiles of the bootstrap) and skewness can often be unaccounted for when dealing with smaller samples of data. The BC method calculates the confidence interval bounds through a formula that involves two factors: the bias factor and the acceleration factor (which measures the rate of change of the standard error of the bootstrap sample). The bias factor is dependent on the proportion of estimates with values lower than the overall bootstrap estimate. Thus, when the calculated bias factor has a lower magnitude

(closer to zero), our confidence interval is closer to simply becoming a percentile-method bootstrap as in (iii). When adjusted for these factors, we can calculate the BCa confidence interval, and it is a simple option to implement in R.

The following table summarizes results from our hand calculations, and both the 'Normal' approximation, along with the 'BCa' interval estimates:

Table 3: Bootstrap Results Comparison (B = 10,000)

Variable	boot (BCa)	boot (Normal)	Hand calc (Normal)	Hand calc (Percentile)
age	[-0.019,0.012]	[-0.021,0.012]	[-0.014,0.017]	[-0.014,0.018]
sex	[-2.051,-0.395]	[-1.955,-0.235]	[-2.069,-0.895]	[-2.426,-0.742]
frail	[0.691,1.711]	[0.368,1.61]	[0.968,1.677]	[0.857,2.058]

As we can see with the BCa confidence interval, it is narrower compared to the calculation of part (i). It is slightly wider compared to the interval calculated using the "norm" type in R for normal approximation. This may be attributed to the ability for bias correction methods to account for tails more effectively, widening the confidence interval slightly.
We must also keep in mind that despite these differing numbers, they are all indeed very similar to each other regardless. Each variable's confidence interval shows the same "effect" (e.g. variables with confidence intervals that were non-inclusive of 0 had the same quality for the other calculated confidence intervals).

v) A better way to construct the large-sample CI for situation in (iii) is to adjust the percentiles in two tails for having 100 bootstrap samples in total. That is, use a trialand-error to locate these two tail-percentiles (there are better methods to find them) for a non-symmetric distribution. For example, one tail might have 700 bootstrap samples, and the other tail has 300 bootstrap samples. The locations of these two tails form the CI. Students can look into the so-called Highest-Probability-Density Confidence Interval (HPD-CI) for a better version of this idea.

To get a sense of how the HPD confidence interval would differ, from the results above, we wrote a simple search function that scans possible 90% point intervals along our boostrap estimates for each variable, and searches for the minimal range among bootstrap estimate intervals. The numerical details can be found in the appendix. The results are show here, and are consistent with other confidence intervals computed in this exercise:

Table 4: Variable Effect Estimate HPD Intervals (90%)

Variable	HPD Interval	Point Index Range (Bootstrap Results, B = 10,000)
age	[-0.014,0.018]	[481,9480]
sex	[-2.349,-0.68]	[642,9641]
frail	[0.792,1.918]	[204,9203]

2. Large-p-Small-n Problems in PH-Regression (30%)

Task: Search the Internet using the key words "Large-p-Small-n Problems", locate three publications with a section on real-life data analysis/example, and briefly summarize (in an half-page report for each publication):
1) the goal of the study,
2) key ideas in solving the problem, and
3) what do you learn from their real-life study

Publication 1: Proportional Hazards Mixed Models

Title: Proportional Hazards Mixed Models: A Review with Applications to Twin Models
Authors: Xu, R
Publication: Metodoloski Zevski (January 2004)
Link: http://www.stat-d.si/mz/mz1.1/xu.pdf

Goal: used the twin pairs consisting of monozygotic and dizygotic twins to implement Proportional Hazards Mixed Models (PHMM) for mixed effects models with right-censored data. Key ideas:
1. added random effects (Normal distribution) to traditional PH model which enable covariate by cluster interactions.
2. took log, the model is written in the equivalent form of a linear transformation model.
3. decomposed the variation in the transformed event time, into contributions from the fixed covariate effects, the random effects, and a fixed error distribution.
4. used EM algorithm to find the estimators where the M-step implementation was similar to the standard Cox model, and the E-step involved Markov Chain Monte Carlo (MCMC) to approximate the conditional distribution of the random effects given data.

Study: The standard cox model is just a beginning. Faced with the real world problem and data, there is more complex relationships. We need to capture mixed, or cluster specific, covariate effects. So, when we study the statistical methods we need to understanding the assumptions, mathematical procedure, results and applications. As a result, we can easily combine methods to analysis the real world problem.

Publication 2: Penalized Cox Regression Analysis

Title: Penalized Cox regression analysis in the high- dimensional and low-sample size settings, with application to microarray gene expression data
Authors: J Gui, H Li
Publication: Bioinformatics, Volume 21, Issue 13, 1 July 2005
Link: https://academic.oup.com/bioinformatics/article/21/13/3001/196819

Goal: Find a model with good prediction accuracy and parsimony property to predict the classified outcome of different types of cancer using gene expression profiles of the cancerous cells as predictors under restriction of high-dimensionality and low-sample-size.

Key ideas in solving the problem:

The article use statistical method mainly including Cox proportional hazard regression model and Lasso estimation to conduct a LARS-Cox procedure which is based on LARS algorithm.

1. Collect data about micro array gene expression and categorical outcome.
2. Use Cox PH regression model to build up log-partial-likelihood function with respect to coefficients β of gene expression signatures.
3. Use Lasso estimate for estimating β subject to the summation of parameters β's is less than a tuning parameter s that determine how many covariates with coefficients are zero.
4. Minimize the Cross-validated partial likelihood to choose the best value of tuning parameter s.
5. Minimize a specific expansion of PH likelihood function (constructed through one-term Taylor Series) using iterative method and find the estimate of β with smallest variance.
6. Select gene expression profiles using the LARS-Cox produce and compare the prediction results with other methods. It turns out that the LARS-Cox procedure is very useful in identifying genes expressions, building a parsimonious predictive model and classifying patients into relevantly higher-risk and lower-risk groups.

Study: Based on standard Cox proportional hazard regression model, researchers often use different statistical method combined with the standard model to optimize the procedure and outcome of regression, which is based on practice of real-life data analysis and adequate statistical knowledges. In this way, it becomes possible to optimize a technical problem such as high-dimensionality and small-sample-size in real life application.

Publication 3: Machine Learning for Survival Analysis

Title: Machine Learning for Survival Analysis: A Survey.
Authors: Ping Wang, Yan Li, Chandan K. Reddy
Publication: ACM Comput. Surv. 1, 1, Article 1 (March 2017)
Link: http://dmkd.cs.vt.edu/papers/CSUR17.pdf

Goal: The primary goal of survival analysis is to predict the occurrence of specific events of interest at future time points. In this survey, the authors have discussed several topics that are closely related to survival analysis and illustrate several successful applications in various real-world application domains. This paper provided a more thorough understanding of the recent advances in survival analysis and offer some guidelines on applying these approaches to solve new problems that arise in applications with censored data.

Key ideas:
1. This article provided a comprehensive review of the conventional survival analysis methods and various machine learning methods for survival analysis, and described other related topics along with the evaluation metric.
2. Researches introduced the basic notations and concepts in survival analysis, including the structure of survival data and the common functions used in survival analysis.
3. Then, they introduced the well-studied statistical survival methods and the representative machine learning based survival methods.
4. Furthermore, they discussed the related topics in survival analysis, including data transformation, early prediction and complex events.
5. They also provided the implementation details of these survival methods and described the commonly used performance evaluation metrics for these models.

Study: Due to the widespread availability of longitudinal data from various real-world domains combined with the recent developments in various machine learning methods, there is an increasing demand for understanding and improving methods for effectively handling survival data.
Traditionally, statistical approaches have been widely developed in the literature to overcome this censoring issue. In addition, many machine learning algorithms are adapted to effectively handle survival data and tackle other challenging problems that arise in real-world data.
Besides the traditional applications in healthcare and biomedicine, survival analysis was also successfully applied in various real-world problems, such as reliability, student retention and user behavior modeling.

Code Appendix

Routine: CPU Parallelized Bootstrap for PH Regression, Hand Calculations

Details on CPU distributed boostrap routine relying on *parallel* and *data.table* libraries:

```
suppressPackageStartupMessages( library(parallel))
suppressPackageStartupMessages( library(data.table))
data("kidney")
kidney <- data.table(kidney)
#parallelize bootstrap to speed up compute
B <- 1e04

bs_res <- rbindlist(mclapply(1:B, function(b) {
  pick_indices <- sample(1:nrow(kidney), size = nrow(kidney), replace = T)
  bs_data <- kidney[pick_indices]
  bs_mod <- coxph(Surv(time, status) ~ age + sex + disease + frail ,
                  data = bs_data)
  df_coef <- data.table(variable = names(bs_mod$coefficients) ,
                        estimate = as.numeric(bs_mod$coefficients),
                        estimate_type = 'Beta',  stringsAsFactors = F)
  df_se <- data.table(variable = names(bs_mod$coefficients) ,
                        estimate = sqrt(diag(bs_mod$var)),
                        estimate_type = 'Standard Error',
                      stringsAsFactors = F)
  return(rbindlist(list(df_coef, df_se)))
}, mc.cores = 10))

#summarize bootstrap results
sum_res <- bs_res[ ,.(bs_mean = mean(estimate, na.rm=T)),.(variable,estimate_type)]
se <- sum_res[estimate_type == 'Standard Error']$bs_mean
sum_res <- sum_res[estimate_type == 'Beta']
sum_res <- transform(sum_res,
             std_err = paste0('[',round(bs_mean + 1.645*se,digits = 3),',',
                                  round(bs_mean - 1.645*se,digits = 3),']'),
             bs_mean = round(bs_mean, digits = 3),
             estimate_type = NULL)
setnames(sum_res, names(sum_res),
         c('Variable','Estimate (Mean)','Std Error (1.645 * Mean SE)'))
```

Routine: HPD Interval Computations

Details on a CPU distributed HPD interval search utilizing the *parallel* and *data.table* libraries:

```
#brute force HPD implementation
#example implementation below based on randomly generated normal variates
N <- 1e04
x <- rnorm(N)
interval <- 0.9

#brute force
find_hpd <- function(x, interval){
  n <- length(x)
  x <- sort(x, decreasing = F)
  point_int <- round(n * interval, digits = 0)
  start_point <- 1:(n-point_int + 1)
  end_point <- start_point + point_int - 1

  hpd_res <- rbindlist(mclapply(1:(n-point_int+1),
    function(a){
      data.frame(
        min_x = x[start_point[a]],
        max_x = x[end_point[a]],
        range_x = x[end_point[a]] - x[start_point[a]],
        index_x_range = paste0('[',start_point[a],',',end_point[a],']'),
        stringsAsFactors = F
      )
    }, mc.cores = 4))

    return(hpd_res[which.min(hpd_res$range_x)])
}

#find_hpd(x = x, interval = 0.9)
```

Questions?
Contact: ygao390, kylee20, ywan40, agovindaraj6, pwilliams60, rzhang438 | @gatech.edu