Assessing artificial censoring weights for clone censor weighting studies

A simulation-based approach

Author: David Pritchard[1], Ph.D. [1]Target RWE

Motivations for simulating data

- Can assess the correctness of software implementations
- Measure bias incurred by using misspecified models
- Assess variance inflation incurred by censoring and inverse probability weighting methods. Useful for power and precision calculations

Example application: assessing bias due to model misspecification

We wanted to assess the accuracy of estimating the probability of remaining on protocol using two different estimation techniques.

- Protocol: undergo surgery within the first six months of followup. Patients who did not undergo surgery were artificially censored at six months
- Generated simulated data where surgery time was based on a time-varying biomarker
- Estimated the probability of surgery by six months using the following models. Both included the biomarker data as inputs.
  - Cox proportional hazards method to calculate the complement of the survival probability of undergoing the surgery by six months
  - Logistic regression with surgery status by six months as the outcome
- Result: logistic regression had a nearly three times higher mean absolute error than estimation based on a Cox proportional hazards model (MAE of 0.0620 compared to 0.0212)
- Conclusion: Logistic regression was unable to capture the time-varying data-generating relationship

Simulating baseline data: resampling

One approach to generating dataset of realistic baseline covariates is to identify a population of interest in a RWE dataset and then sample with replacement from those observed values.

- Sometimes called plasmode simulation
- Very easy and realistic
- The use of your simulated data may be governed by a data use agreement which could prevent you from using it in this manner or sharing the generated data

Simulating baseline data: prescriptive modeling

Suppose that we want to generate the variables *Age*, *Gender*, and *BMI*. We could posit the following models.

- Let *Age* be normally distributed with mean 62 and standard deviation 20
- Let *Gender* follow a logistic regression distribution conditional on *Age*
- Let *BMI* follow a linear regression model conditional on *Age* and *Gender* and some chosen standard deviation

If we have a real data set then we can sequentially estimate the mean and standard deviation for *Age*, the logistic regression coefficients for *Gender*, and the linear regression coefficients and standard deviation for *BMI*.

When prescriptive modeling becomes difficult

Suppose we have several correlated variables such as the statuses for the following comorbidities:

- Coronary artery disease
- Peripheral artery disease
- Cerebrovascular disease
- Type 2 diabetes

It's hard to posit an explicit causal mechanism that describes the relationship between these variables; it might be nice to simply describe the probability of each status being true as well as the pairwise correlations between each.

Simulating baseline data: moments-based approach

A common approach to generating correlated random variables is to generate a multivariate normal random variable and transform the various components to a target set of values. For example, suppose that we want to generate the variables *Age*, *Gender*, and *BMI*.

Then you might specify means of say 62 for age, 0 for gender, and 28 for BMI and corresponding standard deviations of 20, 1, and 6 along with the following correlation matrix.

	Age	BMI	Gender
Age	1.00	−0.01	0.01
Gender	−0.01	1.00	0.04
BMI	0.01	0.04	1.00

Note that the mean and variance chosen for *Gender* is up to this point arbitrary. With this information we have everything you need to specify and draw samples from a multivariate normal distribution.

Finally, to map *Gender* to a binary variable with say probability of 0.5 and noting that gender was drawn from a standard normal distribution, we can convert all values less than 0 to *male* and all values no less than 0 to *female*.

Selecting parameters for baseline variables

- If we have access to a dataset then we can readily calculate the means and covariance matrix for a set of variables to use when generating data
- One complication is that the correlations that we specify for the multivariate normal distribution don't correspond exactly with the correlations between transformed variables; this can be rectified by adjusting the correlations appropriately (see Fialkowski (2018) for a comprehensive treatment)

Simulating time-varying covariates example: weight loss data

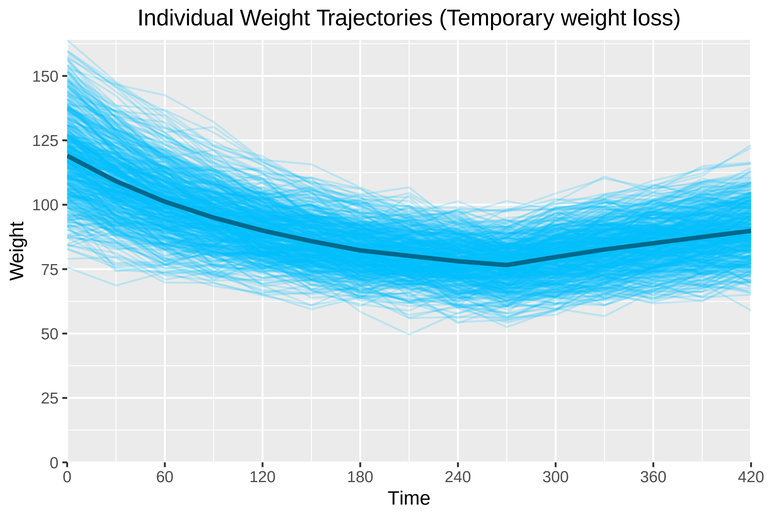
Autoregressive models are a type of time series model where the value of the series at a give time is a linear combination of the previous values plus an additive random term. A simple type of autoregressive model is an AR(1) model which can be expressed as follows.

$$y_t = \mu + \phi(y_{t-1} - \mu) + \epsilon_t.$$

When $|\phi| < 1$ then this process will achieve a steady state centered at μ. Thus, μ and ϕ parameterize how much weight loss a subject will achieve and how long it will take them to achieve it.

Additional extensions:

- We can additionally generate μ and ϕ on a per-subject basis; for example we could model them based on a subject's baseline covariates.
- We can chain multiple AR(1) processes together to create more complex weight-loss patterns such as temporary weight loss where some portion of the weight is regained after an initial loss.



Simulating time to event data: overview

Want to simulate events such as the following:

- Loss to follow up
- Treatment discontinuation
- Related comorbidities
- Outcome of interest

KM and Cox PH assumptions

When we analyze data we'll typically use the Kaplan-Meier or Cox proportional hazards methods to estimate the survival distribution for time to event processes; thus we'll want to generate data from distributions that are consistent with the assumptions made by these models.

The Kaplan-Meier estimator is nonparametric and requires only that survival is a function only of time.

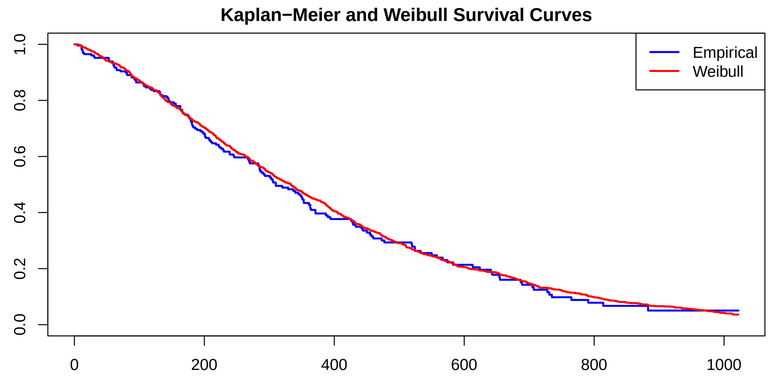
The Cox proportional hazards model assumes that

$$\lambda(t|x(t)) = \lambda_o(t) \exp\left\{\beta^T x(t)\right\},$$

where λ is the hazard function, λ_0 is the baseline hazard function, and $\exp\{\cdot\}$ is the proportional hazards term. The baseline cumulative hazard function is estimated by the Breslow estimator which is nonparametric and requires only that the hazard is a function only of time.

Using the Weibull distribution for the baseline hazard

A commonly used distribution to model the baseline hazard is the Weibull distribution. We can use the Kaplan-Meier estimator on a real dataset to generate a pseudo target baseline cumulative hazard and then choose the optimal parameters for the Weibull distribution to emulate the empirical distribution.



Generating time-to-event values

The proportional hazards function can be estimated directly in the data using the Cox proportional hazards model and used for the coefficients in the simulated data distributions.

Once you have the baseline hazard and proportional hazards in hand then you can use the inverse transform sampling method to generate time-to-event values. See Austin (2012) for a presentation of this approach.

Calculating true probabilities using Monte Carlo sampling

One of the goals of generating simulated data is to assess the properties of various statistical estimators for dynamic treatment regimes.

- In principle, since we know all of the distributions that were used to simulate the data we could (via analytical methods) calculate the integral corresponding to the probability of survival up to time t
- However, this quickly becomes infeasible with longitudinal data so we rely on Monte Carlo sampling to generate large numbers of samples
- Since we don't have to contend with censoring in the simulated data we can simply calculate the proportion of events that occur before any given time t
- To generate conditional distributions under constraints (e.g. the time until surgery for a protocol that requires surgery to occur within the first six months) we can use the accept-reject algorithm