Variance in the survival package

Terry Therneau

30 Sept 2024

1 Infinitesimal jackknife

1.1 Definition

If θ is a statistic that depends on a set of observations, then

$$\begin{aligned} J_{(i)} &= (\theta - \theta_{-i}) \\ V_J &= \frac{n-1}{n} \sum_i (J_{(i)} - \overline{J})^2 \end{aligned}$$

are the jackknife values and the jackknife estimate of variance, respectively, where $-i$ denotes the estimate computed without observation i.

The infinitesimal jackknife (IJ) values and IJ estimate of variance are defined as

$$\begin{aligned} \mathcal{J}_{(i)} &= \left.\frac{\partial \beta_j}{\partial w_i}\right|_{w=1} \\ V_{\mathcal{J}} &= \sum \mathcal{J}^2_{(i)} \end{aligned}$$

An underlying idea is that the IJ is a first order Taylor series approximation to the jackknife. When $n > 20$ and there are no large leverage points, the approximation is often very good. Within the survival package, we can often compute the IJ in $O(n)$ time, versus $O(n^2)$ for the jackknife. One ancillary but useful property of the IJ values is that they sum to zero. In terms of the sign, the survival package takes the view that if adding observation i increase the statistic, then the IJ leverage is positive.

If there are multiple observations per subject then one can use the grouped jackknife or grouped IJ, simply sum the values with group before adding up the squares. Say subject Smith has observations 8–10 in the data set, perhaps due to a time-dependent covariate. Then for the code

```
> coxfit <- coxph(Surv(time1, time2, status) ~ x1 + x2 + x3, mydata, id=clinic)
> D1 = resid(coxfit, type="dfbeta")
> D2 = resid(coxfit, type="dfbeta", collapse=TRUE)
```

D1 will have one row per observation, D2 will have one row per clinic number = sum of rows for that subject. D2'D2 is the grouped IJ variance estimate. This is the default variance when a survfit or coxph call contains an `id` or `cluster` statement.

1.2 Formal

The detailed formulas can be found in formula.pdf. Let W be a diagonal matrix of per observation sampling weights and B be the grouping matrix ($B =$ identity if there in one obs per subject)

```
> B <- model.matrix( ~ factor(id) -1, data=mydata)
```

then

$$V = DWB'BWD \tag{1}$$

For more complex sampling designs B will be more complex, in that case use the survey package.

1.3 Replication weights versus sampling weights

A replication weight of 3 means that there are 3 observations that are identical, but we have compressed the data to save computer memory. (This was still common when I began my career.) The weights statement in coxph, survreg, and survfit implements replication weights, by default.

A sampling weight represents differential weighting and is often fractional. Examples are from survey samples, or IPW used to balance a study. An approximate variance can be obtained by rescaling so that $\sum w_i = n$ and treating these as replication weights. The more proper approach is the IJ variance of formula (1) above.

2 History

Like many good statistical ideas the IJ has been rediscovered multiple times in the statistical literature and has appeared under multplie names. Examples are White's standard error for the linear model (or Huber-White, or Eicker-Huber-White), the working independence variance estimate for a generalized estimating equation (GEE) model, and the Horvitz-Thompson in survey sampling. Is is also called a sandwich estimate.

The White estimate for a linear model is

$$V = (X'X)^{-1}X'R^2X(X'X)^{-1}$$

where R is a diagonal matrix of residuals.

3 Example

For the simple KM, let $e(s)$ and $n(s)$ be the number of events and the number at risk at time s, respectively.

$$\begin{aligned}
\lambda(s) &= \frac{e(s)}{n(s)} \\
&= \frac{\sum w_i dN_i(s)}{\sum w_i Y_i(s)} \qquad (2)\\
S(t) &= \prod_{s \le t} [1-\lambda(s)] \\
\frac{\partial \lambda(s)}{\partial w_i} &= \frac{dN_i(s) - Y_i(s)\lambda(s)}{n(s)} \qquad (3)\\
\frac{\partial S(t)}{\partial w_i} &= \frac{\partial \left(\prod_{s\le t}[1-\lambda(s)]\right)}{\partial w_i} \\
&= \frac{\partial \exp(\log(\prod_{s\le t}[1-\lambda(s)]))}{\partial w_i} \\
&= S(t) \sum_{s\le t} -\frac{dN_i(s) - Y_i(s)\lambda(s)}{n(s)[1-\lambda(s)]} \qquad (4)\\
V(t) &= \sum_{i=1}^{n} \left(\frac{\partial S(t)}{\partial w_i}\right)^2 \qquad (5)\\
&= S^2(t) \sum_{s\le t} \frac{e(t)}{n(t)[n(t)-e(t)]} \qquad (6)
\end{aligned}$$

The exp(log(trick of equation (4) makes the derivative simpler; as importantly it makes the computer code simpler since the term can be accumulated over time. Equation (5) is the IJ variance for the Kaplan-Meier, and equation (6) is the well known Greenwood estimator of variance. If all the case weights are 1 and there is no delayed entry, it turns out that these are identical. This was discovered emprically, and later proven mathematically by Anne Eaton.

3.1 Future

In the survival package, the `resid` function is slowly expanding to deal with all the cases.

- Current:
  - coxph: coefficients
  - survreg: coefficients
  - KM and AJ: probability in state, cumulative hazard, sojourn
- soon: survival, hazard, sojourn from coxph fit
- someday: predicted curves from survreg

An optional collapse argument will return one row per id.

The full IJ residual matrix for an Aalen-Johansen curve would have dimension (number of observations) x (number of event times) x (number of states). That from a post MSH curve will also have x (number of rows in newdata). This can get very large, very fast, so choose your time points carefully.

Most estimands from a MSH can be based on the three primary outputs of hazard, probability in state, or sojourn. This means that, at least in principle, a variance for an estimand θ can be based on the IJ values. That is, first compute the influence of each observation on θ to get an appropriate D matrix, than apply the standard formula above. How well this works in practice for various θ has not yet been investigated.

Computations to fit a MSH model are $O(2np) + O(dp^2)$ where n is the number of observations, d the number of events and p the number of coefficients. Anything that is $O(n^2)$ or $O(nd)$ can be *much* slower, so take care. A growing 'methods' vignette in the package has many pages devoted to the strategies I use to avoid this within the standard computations. IJ residuals have been a challenge. (SAS phreg has a leading term of $O(ndp)$ for (time, time2) data, BTW).

Notes:

1. For an ordinary KM, the IJ variance exactly equals the Greenwood variance. This was a big surprise. Ann Eaton was able to prove it by induction.

2. The AJ and post MSH curves are each a step function. The sojourn time in state k = area under the $p_k(t)$ curve

$$\mathcal{J}_i(AUC_k(t)) = \sum_{s \le t} \mathcal{J}_i(p_k(s))\Delta(s)$$

The naive algorithm that walks forward in time, udating each observation's IJ at each event time, will be $O(nd)$ and thus may be very slow. How can we do this efficiently?

3. One way to think of the D matrix for a Cox model is to consider a Newton-Raphson update step near the solution. At the solution the NR step is 0, per below, where U is the vector of first derivatives and $\mathcal{I}$ the information matrix. Modifiy the weight for observation i slightly and compute the change in $\hat\beta$, this is line 2. Others have pointed out that the derivative of $\mathcal{I}$ is of smaller order than the dU term so we ignore it. Finally dU/dw is a matrix with n rows, each containing the derivative of U wrt weight i.

$$\begin{aligned}
0 &= U'\mathcal{I}^{-1} \\
\frac{\partial \beta}{\partial w_i} &= \frac{\partial U}{\partial w_i}\mathcal{I}^{-1} + U\frac{\partial \mathcal{I}^{-1}}{\partial w_i} \\
D &\approx \frac{dU}{dw}\mathcal{I}^{-1} \\
D'D &\approx \mathcal{I}^{-1}\left[\left(\frac{dU}{dw}\right)'\left(\frac{dU}{dw}\right)\right]\mathcal{I}^{-1}
\end{aligned}$$

The final line has the form of a sandwich estimate.

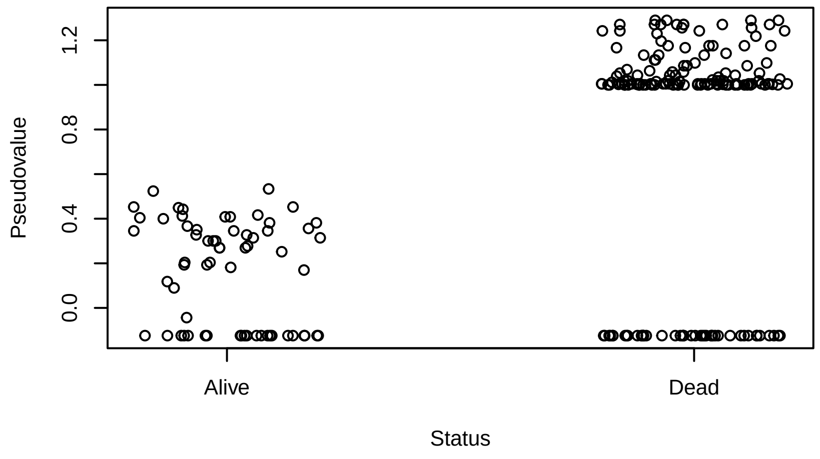
4 Psuedovalues

The IJ pseudovalue is

$$n\mathcal{J}_i + \hat\theta$$

If there is no censoring, the IJ pseudovalue from a KM is t_i, the original survival time. For censored data the pseudovalue computed at time s can act as an *uncensored* representation of $\min(t_i, s)$ for each subject. For instance

```
> sfit <- survfit(Surv(time, status) ~1, data=lung)
> ps <- 1- pseudo(sfit, type="surv", time=365)  # 1 year survival
> # we usually want to model y=death not h=alive in the logit, hence the 1-
> plot(ps ~ jitter(status-1),lung, xaxt='n', xlab="Status", ylab="Pseudovalue")
> axis(1, 0:1, c("Alive", "Dead"))
```



```
>
> lung2 <-cbind(lung, id=1:nrow(lung), pseudo= ps)
> pfit <- glm(pseudo ~ ph.ecog + sex, family= gaussian(link= blogit()), lung2)
> summary(pfit)

Call:
glm(formula = pseudo ~ ph.ecog + sex, family = gaussian(link = blogit()),
    data = lung2)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8458     0.4968   1.703  0.09002
ph.ecog       0.7243     0.2379   3.045  0.00261
sex          -0.8225     0.3195  -2.574  0.01069

(Dispersion parameter for gaussian family taken to be 0.2749687)
```

```
    Null deviance: 66.545  on 226  degrees of freedom
Residual deviance: 61.590  on 224  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 356.09

Number of Fisher Scoring iterations: 5
>
> sdata <- svydesign(id=~id, variables= ~ pseudo + ph.ecog + sex + age,
                    weights=NULL, data= lung2)
> pfit2 <- svyglm(pseudo ~ ph.ecog + sex, design=sdata,
                 family= gaussian(link= blogit()))
> all.equal(coef(pfit2),coef(pfit))
[1] TRUE
> rbind("glm se"= sqrt(diag(vcov(pfit))), "svyglm" = sqrt(diag(vcov(pfit2))))
       (Intercept)   ph.ecog       sex
glm se   0.4967507 0.2378795 0.3195179
svyglm   0.5172857 0.2321103 0.3191577
```

See the `psuedo` command in the survival package. Formally, the variance of a psuedo-value fit needs to use a sandwich estimate, such as is provided by svyglm. When there is a single endpoint, the ordinary glm std will often be very close. When multiple time values are used in a joint model, then the robust sandwich is essential. See the psuedovalues vignette in the survivalVignettes package for more examples.

Notes:

- The pseudo library in R uses the standard jackknife, and is much slower.
- Psueudovalue regression appears be incorrect when there is delayed entry.