Introduction to `risksetROC`: An R Package for Riskset-ROC (Incident/Dynamic) Calculation

P. Saha,[1] P. J. Heagerty,[1,2] and Yingye Zheng[2]
[1] Department of Biostatistics, University of Washington
Seattle, Washington 98195, USA
[2] Division of Public Health Sciences, Fred Hutchison Cancer Research Center
Seattle, Washington 98109, USA

1 Introduction

ROC curves are a popular method for displaying sensitivity and specificity of a continuous diagnostic marker, say, M, for a binary disease variable, say, D. But many disease outcomes are time dependent, $D(t)$ rather than D, and ROC curves that vary as a function of time may be more appropriate. A common example of this time-dependent disease status would be a patient's vital status, where $D(t) = 1$ if a patient has died prior to time t and zero otherwise. A method was proposed by Heagerty et al. (2000) to summarize the discrimination potential of a marker X, measured at baseline ($t = 0$) by calculating ROC curves for cumulative disease or death incidence by time t based on *cumulative/dynamic* definition of time-varying sensitivity and specificity. Another definition, namely, the *incident/dynamic* definition of time-varying sensitivity and specificity was defined in Heagerty and Zheng (2005). This definition is based on dividing the riskset at each time t into cases and controls and hence is a natural companion of hazard models. Based on the incident sensitivity and dynamic specificity, ROC curve can be defined, AUC (Area Under Curve) can be obtained and a global concordance summary can also be estimated. This version of time-dependent accuracy measures have some advantages which we will discuss later. We here introduce an R/S-plus package `risksetROC` which is able to handle the computations as discussed in the paper and produce an ROC curve at a specified time or AUC up to a given time and obtain global concordance measure. To that end, we start by briefly revisiting some necessary concepts and exhibit the utility of the functions in this package. The paper is organized as follows: in the next section we discuss some basic concepts and notations, in the Methods section, we talk briefly about the underlying model and estimation along with a detailed discussion of various functions of the `risksetROC` package and an illustration of how to apply these methods to the real data using two well known data sets: Mayo PBC data and VA lung cancer data.

2 Notation and Basic concepts

2.1 Notation

We introduce the following notations:

M denotes the diagnostic test or marker. By convention, higher values of M are more indicative of the disease.

T denotes the failure time.

C denotes the censoring time.

$X = min(T, C)$ is the follow-up time.

Δ is a censoring indicator with $\Delta = 1$ if $T \leq C$ and $\Delta = 0$ if $T > C$.

We use the counting process $D(t) = 1$ if $T \leq t$ and $D(t) = 0$ if $T > t$ to denote failure (disease) status at any time t with $D(t) = 1$ indicating that event occurred prior to time t.

The survival time T can also be represented through the counting process, $N^*(t) = \mathbb{1}(T_i \leq t)$, or the corresponding increment, $dN^*(t) = N^*(t) - N^*(t-)$. Note that we focus on the counting process $N^*(t)$ which is defined solely in terms of the survival time T rather than the more common notation $N(t) = \mathbb{1}(X \leq t, \Delta = 1)$, which depends on the censoring time. Let $R(t) = \mathbb{1}(X \geq t)$ denote the at-risk indicator. We use subscript i to denote the variable(s) for a subject i and also assume that for each subject we have a collection of time-invariant covariates, $Z_i = (Z_{i1}, Z_{i2}, \ldots, Z_{ip})$.

We now define the *incident/dynamic* version of sensitivity and specificity. At each time-point t, the riskset or the patients still at risk of an event or failure is divided into two mutually exclusive groups: cases and controls. Among the riskset patients, the cases are those who experienced an event at time t while controls are those who did not. Thus, at any given time t and a given cut-off value c, we define:

$$\begin{aligned} \text{sensitivity}^{\mathbb{I}}(c,t) &: \Pr\{M > c \mid T = t\} = \Pr\{M > c \mid dN^*(t) = 1\} \\ \text{specificity}^{\mathbb{D}}(c,t) &: \Pr\{M \leq c \mid T \geq t\} = \Pr\{M \leq c \mid N^*(t) = 0\} \end{aligned}$$

Using the above definitions, we can define the corresponding ROC curve $ROC(t)$ at any time t.

Using this approach a subject, say i, can play the role of a control for an early time, $t < T_i$, but then play the role of case when $t = T_i$. This dynamic status parallels the multiple contributions that a subject can make to the partial likelihood function. Here sensitivity measures the expected fraction of subjects with a marker greater than c among the subpopulation of individuals who die at time t, while specificity measures the fraction of subjects with a marker less than or equal to c among those who survive beyond time t. Incident sensitivity and dynamic specificity are defined by dichotomizing the risk set at time t into those observed to die (cases) and those observed to survive (controls).

2.2 Time-Dependent ROC Curves

After defining the incident sensitivity and dynamic specificity, ROC curves can be computed. In this article we focus on incident/dynamic (I/D) ROC curves defined as the function $\text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p)$, where p denotes the dynamic false-positive rate (1-specificity), and $\text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p)$ denotes the corresponding incident true-positive rate. Specifically, let c^p be defined as the threshold that yields a false-positive rate of $p : P(M_i > c^p | T_i > t) = 1 - \text{specificity}^{\mathbb{D}}(c^p, t) = p$. The true-positive rate, $\text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p)$, is the sensitivity that is obtained using this threshold, or $\text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p) = \text{sensitivity}^{\mathbb{I}}(c^p, t) = P(M_i > c^p | T_i = t)$. Using the true and false-positive rate functions $TP_t^{\mathbb{I}}(c) = \text{sensitivity}^{\mathbb{I}}(c,t)$ and $FP_t^{\mathbb{D}}(c) = 1 - \text{specificity}^{\mathbb{D}}(c,t)$ allows the ROC curve to be written as the composition of $TP_t^{\mathbb{I}}(c)$ and the inverse function $[FP_t^{\mathbb{D}}]^{-1}(p) = c^p$:

$$\text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p) \quad = \quad \text{TP}_t^{\mathbb{I}}\{[\text{FP}_t^{\mathbb{D}}]^{-1}(p)\}$$

for $p \in [0, 1]$. We use the notation $\text{AUC}(t) = \int_0^1 \text{ROC}_t^{\mathbb{I}/\mathbb{D}}(p)dp$ to denote the area under the I/D ROC curve for time t.

2.3 Time-Dependent AUC and Concordance

In the previous subsection we discussed how ROC methods can be used to characterize the ability of a marker to distinguish cases at time t from controls at time t. However, in many applications no a priori time t is identified, and a global accuracy summary is desired. In this subsection we discuss how time-dependent ROC curves are related to a standard "concordance" summary. The global summary we adopt is

$$C = P[M_j > M_k \mid T_j < T_k],$$

which indicates the probability that the subject who died at the earlier time has a larger value of the marker. This is not the usual form (i.e., $P[M_j > M_k \mid T_j > T_k]$), but reflects the conventions for ROC analysis.

In order to understand the relationship between this discrimination summary and ROC curves we assume independence of observations (M_j, T_j) and (M_k, T_k), and assume that T_j is continuous such that $P(T_k = T_j) = 0$. We use $P(x)$ to denote probability or density depending on the context. These assumptions imply that the concordance summary C is a weighted average of the area under time-specific ROC curves,

$$\begin{aligned}
& P[M_j > M_k \mid T_j < T_k] \\
= \; & 2\int_t P[\{M_j > M_k\} \mid \{T_j = t\} \cap \{t < T_k\}] \\
& \times P[\{T_j = t\} \cap \{t < T_k\}]dt \\
= \; & \int_t AUC(t) \times w(t)dt \\
= \; & E_T[AUC(T) \times 2 \times S(T)]
\end{aligned}$$

with $w(t) = 2 \times f(t) \times S(t)$. In this notation AUC$(t)$ is based on the I/D definition of sensitivity and specificity, AUC$(t) = P(M_j > M_k \mid T_j = t, T_k > t)$.

In practice we would typically restrict attention to a fixed follow-up period $(0, \tau)$. The concordance summary can be modified to account for finite follow-up:

$$C^\tau = \int_0^\tau \text{AUC}(t) \times w^\tau(t)dt \qquad (1)$$

where $w^\tau(t) = 2 \times f(t) \times S(t)/W^\tau$, $W^\tau = \int_0^\tau 2 \times f(t) \times S(t)dt = 1 - S^2(\tau)$. The restricted concordance summary remains a weighted average of the time-specific AUCs with the weights rescaled such that they integrate to 1.0 over the range $(0, \tau)$. The interpretation of C^τ is a slight modification of the original concordance, where $C^\tau = P[M_j > M_k \mid T_j < T_k, T_j < \tau]$. Thus C^τ is the probability that the predictions for a random pair of subjects are concordant with their outcomes, given that the smaller event time occurs in $(0, \tau)$.

3 Methods

In this section, we will discuss about the package `risksetROC` and the functions therein. The function to create ROC curve at a given time point is also called `risksetROC()` while `risksetAUC()` creates AUC up to a specified time-point and integrates the AUC values to estimate concordance index. These two functions are build upon the basic functions: `SchoenSmooth()`, `llCoxReg()`, `CoxWeights()` and `IntegrateAUC()`. The first two functions estimates the time-varying coefficients of a Cox model using different methods. The time-varying or time-invariant hazard estimates are used by `CoxWeights()` function to estimates ROC at a specified time-point while the last one integrates the AUC values based on different time-points to obtain an estimate of concordance index. The set of utility functions also includes `riskset()`, `weightedKM()` and `KM.plot()`. Among these functions, `KM.plot()` is a easier version of the generic `plot()` function for plotting Kaplan-Meier survival curve and `weightedKM()` estimates survival $S(t)$ when sampling weights are permitted. The `riskset()` function creates riskset at unique failure times. The details are provided later. We note here that the functions can accommodate both right censored and left truncated data.

3.1 The `risksetROC()` function

This function creates ROC curve at a given time from a survival dataset.

```
> library(risksetROC)
> args(risksetROC)
function (Stime, entry = NULL, status, marker, predict.time, method =
     "Cox", span = NULL, order = 1, window = "asymmetric", prop = 0.5,
     plot = TRUE, type = "l", xlab = "FP", ylab = "TP", ...)
NULL
```

The arguments and their details are as follows:

- `Stime`: For right censored data, this is the follow up time. For left truncated data, this is the ending time for the interval.
- `entry`: For left truncated data, this is the entry time of the interval. The default is set to *NULL* for right censored data.
- `status`: survival status, 1 if had an event and 0 otherwise.
- `marker`: marker value. We want to see how good the marker is to distinguish between cases (incident cases) and controls. This can either be a linear predictor from a Cox model or a raw marker vector.
- `predict.time`: time point of interest. For example, if survival time is given in days and if we want to see how well the marker predicts those who would die at the end of first month, i.e., 30 days, then `predict.time` would be 30. Note that `Stime` and `predict.time` has the same unit.
- `method`: either of `"Cox"`, `"LocalCox"`, `"Schoenfeld"`, default is `"Cox"`. `method = "Cox"` is based on the assumption that the survival model has a proportional hazard in marker. The other two methods assumes that the proportional hazard does not hold. For estimation of time-varying coefficient, either of two methods (`"LocalCox"` or `"Schoenfeld"`) can be used. These two variations of non-proportional hazard differs in that how the smoothing is handled. In case of `method = "Schoenfeld"`, Schoenfeld residuals are used to estimate the time varying hazard coefficient. `method = "LocalCox"` uses the method of Cai and Sun (2003) to estimate the time-varying coefficients. Note that `method = "LocalCox"` uses the function `llCoxReg()` and the `method = "Schoenfeld"` uses the function `SchoenSmooth()`. We will discuss these functions in detail later in this section.
- `span`: bandwidth parameter that controls the size of a local neighborhood, needed for `method = "LocalCox"` or `method = "Schoenfeld"`.
- `order`: 0 or 1, if `order = 0`, time-varying coefficient is estimated by a local mean within a window and is approximated by a local linear function if `order = 1`. This argument is needed for `method = "Schoenfeld"` and the default value is 1.
- `window`: either of `"asymmetric"` or `"symmetric"`, default is `"asymmetric`, needed for `method = "LocalCox"`. This argument handles how the observations near a given event time are considered. If `window = "symmetric"` then a symmetric window is considered around a given event time. Otherwise an asymmetric window is considered.
- `prop`: This denotes what proportion of the length of time-interval to cover when doing a local Cox fitting at `predict.time`, needed for `method="LocalCox"`, and the default is 0.5. If `prop = 1.0`, then time-varying hazard is estimated at all the event times and the hazard estimate at or close to `predict.time` is used for further estimation of (FP, TP) values. Otherwise, a subset of event times covering `predict.time` is selected to estimate the time-varying hazard at `predict.time`. Thus, specifying `prop` to be less than 1 allows to run the subroutine (`llCoxReg()` call) on a subset of the data and hence is much faster.

- `plot`: `TRUE` or `FALSE`, default is `TRUE`. If `plot = TRUE` then ROC curve is plotted.
- `type`: This and the remaining options are for plotting the ROC curve. This is the same `type` option from `plot()` function. Default is "l", can be either of "p" for points, "l" for line, "b" for both.
- `xlab`: label for x-axis. Default is `"FP"`.
- `ylab`: label for y-axis. Default is `"TP"`.
- `...`: additional plot arguments.

This function accepts a marker based on any model. Then if a PH model is assumed, then `method = "Cox"` can be used, which takes the marker values, estimate the hazard ratio (in case the marker is already optimized using a Cox model the estimated HR would be 1) and uses Equation (1) and Equation (2) of Heagerty and Zheng (2005) to find true positive (TP) and false positive (FP) values corresponding to different threshold values of the marker. In case, PH assumption is violated, `method = "LocalCox"` or `method = "Schoenfeld"` can be used. These two methods allow for a time-varying hazard in marker and differs in how the estimation is handled. In case of `method = "Schoenfeld"` options, function `SchoenSmooth()` is used which in turn uses the residuals from a `coxph()` function (from R-package `survival`) to estimate the time-varying hazard. On the other hand, `method = "LocalCox"` uses locally weighted partial likelihood methods (function `llCoxReg()`) of Cai and Sun (2003) to estimate the time-varying coefficient. The original or the modified markers are then used to estimate the (FP, TP) for different thresholds.

The function returns a list with the following items: `marker` denoting unique marker values for calculation of true and false positives, `TP` denoting the True Positive values corresponding to unique marker values, `FP` the False Positive values and `AUC` denoting the Area Under (ROC) Curve at `predict.time`.

3.2 The `risksetAUC()` function

This function estimates AUC at unique failure times and integrates AUC with weights to find concordance index up to a given time point.

```
> args(risksetAUC)
function (Stime, entry = NULL, status, marker, method = "Cox", span = NULL,
    order = 1, window = "asymmetric", tmax, weight = "rescale",
    plot = TRUE, type = "l", xlab = "Time", ylab = "AUC", ...)
NULL
```

The arguments are similar to the argument of `risksetROC()`. The additional arguments are discussed below.

- `tmax`: maximum time to be considered for estimation of concordance index. Concordance index is defined as the probability that the markers values from two patients are ordered given that the their event times are ordered. It is assumed that smaller of the two event time is less than `tmax`. Note that `tmax` is τ in the Equation 1.
- `weight`: This indicates how the weights for estimation of concordance index the weights should be handled. If `weight = “rescale”`, then the weights are rescaled such that their sum is unity. The rescaled weight is really $w^{\tau}(t)$ in Equation 1. If `weight = “conditional”`, concordance index is estimated conditional on the fact that both the event times are less than `tmax`.

This function is similar to the `risksetROC()` function. It creates ROC curve at each unique failure times (till `tmax`) and uses the AUC values to estimate the concordance index (Equation 1). In particular it uses `weightedKM()` to estimate the survival probabilities at unique failure times. Modified or original marker values (depending on the `method` choice) are then used to find (FP, TP) values at unique failure times. AUC values at these failure times is also estimated. Finally it uses function `IntegrateAUC()` to find an estimate of concordance index. The return list therefore includes `utimes` denoting ordered unique failure times, `St` denoting Kaplan-Meier survival probabilities at `utimes`, `AUC` at `utimes` and `Cindex` denoting an estimate of concordance index.

3.3 Utility Functions

3.3.1 The `riskset()` function

The `riskset()` function creates riskset at each unique failure time from a right censored or both right censored and left truncated survival dataset.

```
> args(riskset)
function (dat, entry = FALSE)
NULL
```

The argument `dat` is a survival dataset with at least three variables: survival times, survival status and marker x, in that order. Note that the name of these columns are not important, but their order is. The survival data set may have additional variables. The additional argument `entry` denotes whether the data is right censored or left truncated. The default is `entry = FALSE` indicating right censored data. If `entry = TRUE`, then at least four variables are expected. These are: entry time, exit time (survival time or censoring time), status at exit time and marker.

This function first identifies the unique failure times from the dataset. If an event happened at baseline i.e., at time $t = 0$, then the event time is reassigned to time $t = 0.0001$. In case of left truncated data if entry time and exit time are the same and the subject experienced an event at the exit time, then also 0.0001 is added to the exit time. The entire time-length is then subdivided into smaller intervals. Thus, if $0 < t_1 < t_2 < \ldots < t_k$ denotes the unique failure times after reassignment of any event time that may have happened at time $t = 0$, then the smaller time-intervals to be considered are $(t_0, t_1], (t_1, t_2], \ldots, (t_{k-1}, t_k]$, where t_0 is 0 if right censored data is considered and is minimum of the entry times when left truncated data is considered. Note that in case of left truncated data, if a subject enter into the study in between two consecutive event times, say t_i and t_{i+1}, he is considered to be present in the risk set at time t_{i+1}, and hence considered to be present in the riskset $(t_i, t_{i+1}]$. For the first interval $(t_0, t_1]$, all the patients are at risk of an event. This set of patients is then subdivided into two groups: cases and controls. Cases are those who experienced an event at t_1 and all the rest are controls. The cases are assigned a new status value of 1 to indicate that they have an event at time t_1. The new status value for controls in this time interval is 0. We then drop the cases from the previous interval from consideration as they are no longer at risk of an event later on and move over to the next time interval. The subjects who had an event at time t_2 are now cases and are assigned a new status of 1 to indicate that they have an event at time t_2. The rest of the patients who did not have an event till time t_2 are considered as controls and moved on to the next riskset. We repeat this procedure till the event time t_k. The new dataset that is created after `riskset()` call have one more column than the original dataset if `entry = FALSE` and have the same number of columns if `entry = TRUE`. The first two columns of the new dataset are called `start` and `finish` indicating the endpoints of the time intervals considered. The third column is the `newStatus` column which takes value 1 if the subject had an event between $(\text{start}, \text{finish}]$ and 0 otherwise. The rest of the columns are the remaining columns of the original dataset (marker and other variables).

3.3.2 The `llCoxReg()` function

This function estimates time-varying hazard using the method of Cai and Sun (2003).

```
> args(llCoxReg)
function (Stime, entry = NULL, status, marker, span = 0.4, p = 1, window = "asymmetric")
NULL
```

Here `p` is 1 if the time-varying coefficient is of interest and 2 if the derivative of time-varying coefficient is also of interest, default is 1. Both `risksetROC()` and `risksetAUC()` uses `p = 1` when `method = "LocalCox"` is used. The rest of the arguments are the same as the arguments of `risksetROC()`. This function returns a list of the following items:

- `time`: unique failure times.
- `beta`: estimate of time-varying hazard at the unique failure times.

3.3.3 The `SchoenSmooth()` function

Just like the `llCoxReg()` function, this function also estimate time-varying hazard using Schoenfeld residuals from a `coxph()` model fit.

```
> args(SchoenSmooth)
function (fit, Stime, status, span = 0.4, order = 0, entry = NULL)
NULL
```

Other than the usual arguments, this function uses `fit` which is the result of fitting a Cox regression model, using the `coxph()` function. When left truncated data is considered, `Stime` denotes the survival times and `entry` now denotes the entry times for subjects. The return items are the same as `llCoxReg()`, except, it returns (multiple) failure times and time-varying hazard at those times. So if three subjects have an event time of 100 days, then it will return the hazard at $t = 100$ three times.

3.3.4 The `weightedKM()` function

This function returns estimated Kaplan-Meier survival probability at unique failure times and also allows for weights to be introduced in the estimation.

```
> args(weightedKM)
function (Stime, status, wt = NULL, entry = NULL)
```

Here `wt` denotes the weights. By default, all observations are given equal weight and all the subjects are assumed to enter at the same time (right censored data). In that case, `Stime` denotes the survival time for subjects. When left truncated data is considered, `Stime` again denotes the survival times as before and `entry` now denotes the entry times for subjects. The return list includes `time` denoting the unique event times and `survival` denoting the estimated survival probabilities.

3.3.5 The `KM.plot()` function

As stated earlier, this is an easier version of the `plot()` function to create Kaplan-Meier survival plot.

```
> args(KM.plot)
function (Stime, survival, max.T = NULL, lty = NULL, all = TRUE,
    ...)
```

The arguments have the usual meaning and `...` denotes the additional plotting arguments to be passed on. In case `all = TRUE`, a new plot would be created with time and corresponding survival probability, otherwise, the KM survival probability would be added to an existing plot.

3.3.6 The `CoxWeights()` function

This function estimates of TP and FP based on a Cox model as discussed in Heagerty and Zheng (2005), for incident/dynamic ROC curve. TP is estimated as Equation (1) and FP is estimated as Equation (2) of the paper. The arguments are as follows:

```
> args(CoxWeights)
function (marker, Stime, status, predict.time, entry = NULL)
NULL
```

Here `marker` denotes the marker (modified to reflect time-varying hazard or used as it is to reflect PH) while `predict.time` is the time point of interest. Note that, `predict.time`, `Stime` and `entry` should have the same units. The return list includes `marker` denoting the ordered marker values of the subjects in the riskset at `predict.time`, `TP` denoting true positives and `FP` denoting false positives at these marker values and `AUC` denoting the AUC at `predict.time`.

3.3.7 The `IntegrateAUC()` function

This function accepts AUC at unique failure times and Kaplan-Meier survival probability to find an estimate of C in Equation 1.

```
> args(IntegrateAUC)
function (AUC, utimes, St, tmax, weight = "rescale")
NULL
```

The `tmax` and `weight` arguments have the same interpretation of the `tmax` and `weight` argument from `risksetAUC()` function and other arguments have the usual meaning.

3.4 Illustration: Mayo PBC data and Veterans' Administration Lung Cancer data

We demonstrate the methods discussed above with two well-known datasets. Note that Veterans' Administration Lung Cancer data (or `VA` from here onwards) is used as it is from the library `MASS` while Mayo PBC data is obtained from `http://lib.stat.cmu.edu/datasets/pbc`. This has the same set of variables and observations as the `pbc` data from library `survival`, but the variables are arranged differently.

```
> data(pbc)
> str(pbc)
'data.frame':   418 obs. of  20 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ fudays  : int  400 4500 1012 1925 1504 2503 1832 2466 2400 51 ...
 $ status  : int  2 0 2 2 1 2 0 2 2 2 ...
 $ drug    : int  1 1 1 1 2 2 2 2 1 2 ...
 $ age     : int  21464 20617 25594 19994 13918 24201 20284 19379 15526 25772 ...
 $ sex     : int  1 1 0 1 1 1 1 1 1 1 ...
 $ ascites : int  1 0 0 0 0 0 0 0 0 1 ...
 $ hepatom : int  1 1 0 1 1 1 1 0 0 0 ...
 $ spiders : int  1 1 0 1 1 0 0 0 1 1 ...
 $ edema   : num  1 0 0.5 0.5 0 0 0 0 0 1 ...
 $ bili    : num  14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
 $ chol    : int  261 302 176 244 279 248 322 280 562 200 ...
 $ albumin : num  2.6 4.14 3.48 2.54 3.53 3.98 4.09 4 3.08 2.74 ...
 $ copper  : int  156 54 210 64 143 50 52 52 79 140 ...
 $ alkphos : num  1718 7395  516 6122  671 ...
 $ sgot    : num  137.9 113.5  96.1  60.6 113.2 ...
 $ trig    : int  172 88 55 92 72 63 213 189 88 143 ...
 $ platelet: int  190 221 151 183 136 NA 204 373 251 302 ...
 $ protime : num  12.2 10.6 12 10.3 10.9 11 9.7 11 11 11.5 ...
 $ stage   : int  4 3 4 4 3 3 3 3 2 4 ...
```

Note that the first 312 are the randomized patients. We start by using a marker based on five covariates: `log(bili)`, `log(protime)`, `edema`, `albumin` and `age` based on the randomized patients.

```
## considering only randomized patients
pbc1 <- pbc[1:312,]
## create new censoring variable combine 0,1 as 0, 2 as 1
survival.status <- ifelse( pbc1$status==2, 1, 0)
survival.time <- pbc1$fudays
pbc1$status1 <- survival.status
fit <- coxph( Surv(fudays,status1) ~ log(bili) +
                                     log(protime) +
                                     edema +
                                     albumin +
                                     age,
              data=pbc1 )
eta <- fit$linear.predictors
```

We will now plot the ROCs at $t = 90$ using the three method.

```
nobs <- length(survival.time[survival.status==1])
span <- 1.0*(nobs^(-0.2))

ROC.CC90=risksetROC(Stime=survival.time, status=survival.status,
                    marker=eta, predict.time=90, method="Cox", main=
                    "PBC Data: ROC Curve", lty=2, col="red")
ROC.SS90=risksetROC(Stime=survival.time, status=survival.status,
                    marker=eta, predict.time=90, method="Schoenfeld",
                    plot=FALSE, span=span)
ROC.LL90=risksetROC(Stime=survival.time, status=survival.status,
                    marker=eta, predict.time=90, method="LocalCox",
                    plot=FALSE, span=span, prop=1.0)

lines(ROC.SS90$FP, ROC.SS90$TP, lty=3, col="darkblue")
lines(ROC.LL90$FP, ROC.LL90$TP, lty=4, col="green")
legend(.6, .25, lty=c(2,3,4), col=c("red", "darkblue", "green"),
legend=c("Cox","Schoenfeld","LocalCox"), bty="n")
```

Given the (FP, TP), we can plot the ROC curves $ROC(t)$ at time $t = 90$ days (Figure 1). Note that the AUCs from these three methods are: 0.88 (`Cox`), 0.94 (`Schoenfeld`) and 0.91 (`LocalCox`).

To show the usage of the `risksetAUC` function, we will use the `VA` data.

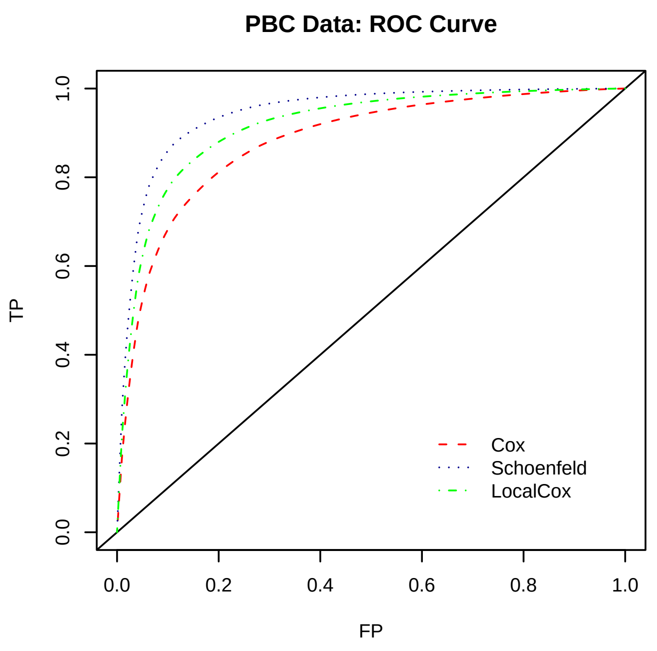

Figure 1: Estimated ROC curves $ROC(t)$ at $t = 90$ days

```
library(MASS)
data(VA)
survival.time=VA$stime
survival.status=VA$status
score <- VA$Karn
cell.type <- factor(VA$cell)
tx <- as.integer( VA$treat==1 )
age <- VA$age
survival.status[survival.time>500 ] <- 0
survival.time[survival.time>500 ] <- 500
fit0 <- coxph( Surv(survival.time,survival.status)
        ~ score + cell.type + tx + age, na.action=na.omit)
eta <- fit0$linear.predictor
tmax=300

AUC.CC=risksetAUC(Stime=survival.time, status=survival.status, marker=eta,
                  method="Cox", tmax=tmax, col="red", lty=2, main="VA
                  Lung Cancer Data: AUC Plot");
AUC.SS=risksetAUC(Stime=survival.time, status=survival.status, marker=eta,
                  method="Schoenfeld", span=0.4, tmax=tmax, plot=FALSE);
AUC.LL=risksetAUC(Stime=survival.time, status=survival.status, marker=eta,
                  method="LocalCox", span=0.4, tmax=tmax, plot=FALSE);

lines(AUC.SS$utimes, AUC.SS$AUC, lty=3, col="darkblue")
lines(AUC.LL$utimes, AUC.LL$AUC, lty=4, col="green")
```

```
legend(250, .5, lty=c(2,3,4), col=c("red", "darkblue", "green"),
legend=c("Cox","Schoenfeld","LocalCox"), bty="n")
```

Note that the estimated concordance index from the three methods are: 0.703 (`Cox`), 0.731 (`Schoenfeld`) and 0.718 (`LocalCox`).

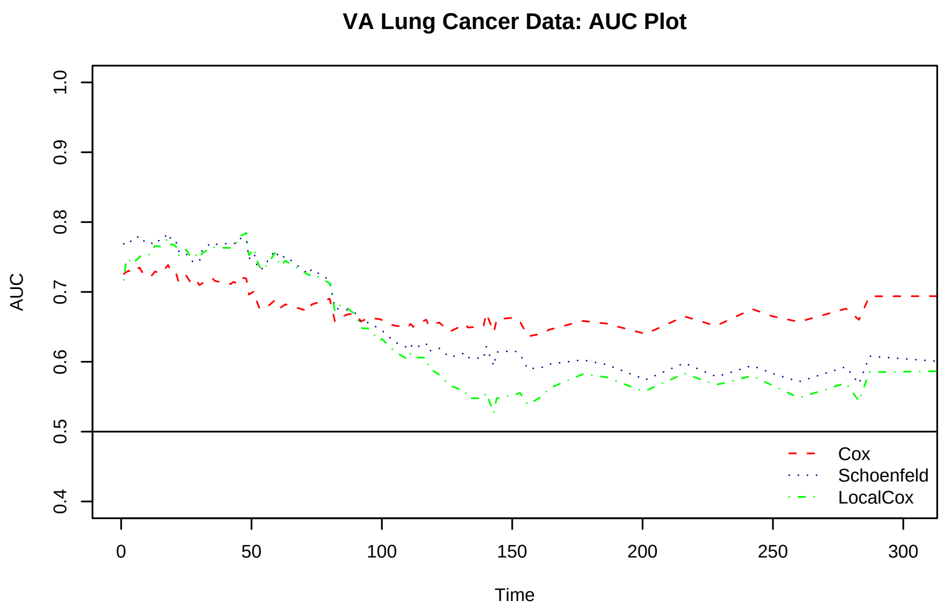

Figure 2: AUC based on VA lung cancer data up to 300 days

We close this section by noting that a comparison of the three methods of estimation can be found in the original paper (Heagerty and Zheng (2005)). From the point of view of use of the functions, we note that in case of time-varying coefficient, `method = "LocalCox"` takes (about 5 times) longer (system CPU time) to run then the comparable `method = "Schoenfeld"`, for example, `method = "Schoenfeld"` takes 0.007 units of system CPU time whereas `method = "LocalCox"` takes 0.033 units of system CPU time.

```
> system.time(risksetROC(Stime=survival.time, status=survival.status,
marker=eta, predict.time=90, method="Schoenfeld", plot=FALSE,
span=span))
[1] 0.739 0.007 0.746 0.000 0.000
> system.time(risksetROC(Stime=survival.time, status=survival.status,
marker=eta, predict.time=90, method="LocalCox", plot=FALSE,
span=span))
[1] 16.542  0.033 16.581  0.000  0.000
```

4 Conclusion

ROC curves are useful tool for showing diagnostic potential of continuous marker in the setting of survival data. Based on the paper by Heagerty and Zheng (2005), we here introduce and discuss the usage of a new R/S-plus package `risksetROC` and the functions therein. These functions are able to estimate the discriminatory potential of a continuous marker by time-dependent ROC curve $ROC(t)$ based on *incident* sensitivity and *dynamic* specificity. We note here that for inference and variance estimation, we now suggest bootstrapping, however we are in a process of developing the theory and later, software, to estimate the variance of the $AUC(t)$ and C. We here show the different uses of the package and hope, it will be helpful to the scientific community.

References

Cai, Z. and Sun, Y. (2003). Local linear estimation for time-dependent coefficient in Cox's regression model. *Scandinavian Journal of Statistics*, 30: 93 – 111.

Heagerty, P. J., Lumley, T., and Pepe, M. S. (2000). Time-dependent ROC curves for censored survival data and a diagnostic marker. *Biometrics*, 56 : 337–344.

Heagerty, P. J. and Zheng, Y. (2005). Survival model predictive accuracy and ROC curves. *Biometrics*, 61 : 92–105.