Appendix A: Estimation Methods

Let

- M denote a continuous marker or test. By convention, higher values of M are more indicative of the adverse outcome
- T denote the failure time
- C denote the censoring time
- $Z = min(T, C)$ is the follow-up time
- δ denote the event indicator with $\delta = 1$ if $T \leq C$ and $\delta = 0$ if $T > C$
- subscript i denote the variables for a subject i

1. Cumulative (Prevalent) Cases / Dynamic Controls

Let

- s denote the start time of case ascertainment (often s=0 for baseline)
- t denote the stop time of case ascertainment

At any give times s and t and given cut-off value c, we define sensitivity and specificity as:

$$\text{Se}^{\text{C}}(c \mid \text{start} = s, \text{stop} = t) = \text{P}(M > c \mid T \geq s\,,\, T \leq t)$$

$$\text{Sp}^{\text{D}}(c \mid \text{start} = s, \text{stop} = t) = \text{P}(M \leq c \mid T \geq s\,,\, T > t)$$

Using these definitions, the corresponding ROC curve can be defined at any times s, t. Heagerty et al.[1] developed two estimators for sensitivity and specificity where case ascertainment was assumed to begin at baseline, i.e. s = 0. These methods, described below, can be extended to sequential baseline values of s to characterize time-varying performance, as described in the main text.

(a) Kaplan-Meier estimator

Using Bayes' Theorem, the widely used nonparametric Kaplan-Meier estimate of the survival function, and the empirical distribution function of the marker M, Heagerty et al.[1] provided simple estimators for sensitivity and specificity as

$$\widehat{\text{Se}}^{\text{C}}(\text{c}|\text{start} = 0, \text{stop} = t) = \frac{\left\{1 - \hat{S}_{KM}(t|M > c)\right\}\left\{1 - \hat{F}_M(c)\right\}}{1 - \hat{S}_{KM}(t)}$$

$$\widehat{\mathrm{Sp}}^{D}(\mathrm{c}|\mathrm{start}=\ 0,\mathrm{stop}=t)=\frac{\hat{S}_{KM}(t|M\leq c)\ \hat{F}_{M}(c)}{\hat{S}_{KM}(t)}$$

where $\hat{S}_{KM}(t)$ is the Kaplan-Meier estimate of the survival function, $\hat{S}_{KM}(t|M>c)$ is the Kaplan-Meier estimate of the conditional survival function for the subset defined by $M > c$, and $\hat{F}_{M}(c)=\frac{1}{n}\sum 1(M_i\leq c)$ is the empirical distribution function of the marker M.

The Kaplan-Meier estimator is a standard and widely-used nonparametric estimator of the survival function, which uses all the information in the data, including censored observations, for estimation. However, there are two potential drawbacks of this estimation approach: (i) it does not guarantee that sensitivity and specificity are monotone in M and bounded by [0,1], and (ii) the conditional Kaplan-Meier estimator $\hat{S}_{KM}(t|M>c)$ assumes that the censoring mechanism does not depend on M. This assumption may be violated in practice when the intensity of follow-up is influenced by the marker measurements, a common scenario that results in marker-dependent censoring.

(b) Nearest neighbor estimator

An alternative approach proposed by Heagerty et al.[1] to address the above drawbacks is based on a nearest neighbor estimator for the bivariate distribution function of (M, T), $F(c,t) = P(M \leq c, T \leq t)$, or equivalently, $S(c,t) = P(M > c, T > t)$, that was provided by Akritas[2]. The estimator is based on the representation: $S(c,t)=\int_{c}^{\infty} S(t|M=s)dF_{M}(s)$, where $F_M(s)$ is the distribution function of M. This estimator is provided by

$$\hat{S}_{\lambda_n}(c,t)=\frac{1}{n}\sum_i \hat{S}_{\lambda_n}(t|M=M_i)1(M_i>c),$$

where $\hat{S}_{\lambda_n}(t|M=M_i)$ is a suitable estimator of the conditional survival function characterized by a smoothing parameter λ_n. Unless M is discrete and there are sufficient observations at each value of M, some smoothing is required to estimate $S(t \mid M = M_i)$. $K_{\lambda_n}(M_j,M_i)$ is defined as a kernel function that depends on a smoothing parameter λ_n. Using the kernel function, a weighted Kaplan-Meier estimator follows:

$$\hat{S}_{\lambda_n}(t|M=M_i)=\prod_{s\epsilon\mathcal{T}_n,s\leq t}\left\{1-\frac{\sum_j K_{\lambda_n}(M_j,M_i)\,1(Z_j=s)\delta_j}{\sum_j K_{\lambda_n}(M_j,M_i)\,1(Z_j=s)}\right\}$$

where $\mathcal{T}_n$ is the set of unique values of Z_i for observed events, $\delta_i = 1$.

Akritas[2] used a 0/1 nearest neighbor kernel, $K_{\lambda_n}(M_j, M_i) = 1\{-\lambda_n < \hat{F}_M(M_i) - \hat{F}_M(M_j) < \lambda_n\}$, where $2\lambda_n \in (0, 1)$ represents the percentage of individuals that are included in each neighborhood. The resulting estimates of sensitivity and specificity are given by

$$\widehat{\text{Se}}^C(c|\text{start} = 0, \text{stop} = t) = \frac{\{1 - \hat{F}_X(c)\} - \hat{S}_{\lambda_n}(c, t)}{1 - \hat{S}_{\lambda_n}(t)}$$

$$\widehat{\text{Sp}}^D(c|\text{start} = 0, \text{stop} = t) = 1 - \frac{\hat{S}_{\lambda_n}(c, t)}{\hat{S}_{\lambda_n}(t)}$$

where $\hat{S}_{\lambda_n}(t) = \hat{S}_{\lambda_n}(-\infty, t)$. These estimates allow for monotonicity of the sensitivity and specificity. Furthermore, since only local Kaplan-Meier estimators are used in each possible neighborhood of M=*m*, the censoring process is allowed to depend on the marker *M*.

2. Incident Cases / Dynamic Controls

At any give time *t* and given cut-off value *c*, 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):

$$\text{Se}^I(c \mid t) = \text{P}(M > c \mid T = t)$$

$$\text{Sp}^D(c \mid t) = \text{P}(M \leq c \mid T > t)$$

Using these definitions, the corresponding ROC curve can be defined at any time *t*. Below we describe two estimators for sensitivity and specificity based on the incident/dynamic definition.

(a) Semi-parametric Cox model based estimator

Heagerty & Zheng[3] proposed Cox model based methods that use riskset reweighting based on the estimated hazard in order to estimate sensitivity and specificity. The censoring time is assumed to be independent of the failure time and marker. Under proportional hazards, a standard Cox model is fit:

$$\lambda(t \mid M_i) = \lambda_0(t)\, exp(M_i\, \gamma)$$

To estimate sensitivity and specificity, i.e. the marker distribution conditional on survival time, Heagerty & Zheng[3] use Xu and O'Quigley's result that partial likelihood estimation

methods can be exploited to provide model-based estimates of the distribution of covariates conditional on survival time.[3,4] Specifically, letting $R_i(t) = 1(M_i \geq t)$ denote the at-risk indicator, $\pi_i(\gamma, t) = \frac{R_i(t)\exp(M_i\gamma)}{\sum_j R_j(t)\exp(M_j\gamma)}$ can be used to estimate the distribution of marker M, conditional on the event occurring at time t, so that $\hat{P}(M_i \leq m \mid T_i = t) = \sum_k \pi_k(\hat{\gamma}, t)\ 1(M_k \leq m)$. This result and using partial likelihood to estimate γ directly give a semiparametric estimator of sensitivity, which uses a reweighting of the marker distribution observed among the riskset at a time t:

$$\widehat{\mathrm{Se}}^{\mathrm{I}}(c \mid t) = \hat{P}(M > c \mid T = t) = \sum_k 1(M_k > c)\ \pi_k(\hat{\gamma}, t)$$

The methods can also accommodate non-proportional hazards. A varying-coefficient model of the form $\lambda(t \mid M_i) = \lambda_0(t)\exp(M_i\,\gamma(t))$ can be fit to obtain the time-varying coefficient $\hat{\gamma}(t)$ and estimate sensitivity as:

$$\widehat{\mathrm{Se}}^{\mathrm{I}}(c \mid t) = \hat{P}(M_i > c \mid T_i = t) = \sum_k 1(M_k > c)\ \pi_k[\hat{\gamma}(t), t]$$

The time-varying coefficient, $\gamma(t)$, and subsequently $\mathrm{AUC}^{\mathrm{I/D}}(t)$, can be estimated using flexible semiparametric locally weighted partial likelihood methods[5] or local linear smoothing of the scaled Schoenfeld residuals.

An empirical estimator of specificity is given as:

$$\widehat{\mathrm{Sp}}^{D}(c \mid t) = \hat{P}(M \leq c \mid T > t) = \frac{\sum_k 1(M_k > c, T_k > t)}{\sum_k 1(T_k > t)}$$

(b) Non-parametric rank-based estimator

A nonparametric rank-based approach for the estimation of $\mathrm{AUC}^{\mathrm{I/D}}(t)$ was proposed by Saha-Chaudhuri & Heagerty[6]. For a fixed time t, a percentile is calculated for each case in the risk set relative to the controls in the risk set. A perfect marker would have the case marker value greater than 100% of risk set controls. The mean percentile at time t is calculated as the mean of the percentiles for all cases at t, as follows:

$$A(t) = \frac{1}{n_t d_t} \sum_{i \in \mathcal{R}_t^1} \sum_{j \in \mathcal{R}_t^0} 1(M_i > M_j)$$

where $\mathcal{R}_t^1$ and $\mathcal{R}_t^0$ denote the sets of cases and controls in the risk set at time t, respectively.

Unless there are sufficient events at each time *t*, some smoothing is typically required to estimate the AUC. Using a standardized kernel function such that $\sum_j K_{h_n}(t - t_j) = 1$ based on a neighborhood of points defined by parameter h_n, Saha-Chaudhuri & Heagerty[6] defined a smoothed estimator of AUC by:

$$\widehat{AUC}(t) = \sum_j K_{h_n}(t - t_j)\, A(t_j).$$

They used a nearest neighbor kernel, resulting in the following weighted mean rank (WMR) estimator:

$$WMR(t) = \frac{1}{|\mathcal{N}_t(h_n)|} \sum_{t_j \epsilon \mathcal{N}_t(h_n)} A(t_j)$$

where $\mathcal{N}_t(h_n) = (t_j : |t - t_j| < h_n)$ denotes a neighborhood around time *t*. This estimator is used to estimate the summary curve, $\text{AUC}^{\text{I/D}}(t)$, as the local average of mean case percentiles. This nonparametric approach provides a simple description of marker performance within each risk set and, by smoothing individual case percentiles, a final summary curve characterizes accuracy over time.

A smooth curve of sensitivity for a fixed specificity can be estimated in a similar manner as:

$$\widehat{\text{Se}}^{\text{I}}(\text{Sp} \mid t) = \sum_j K_{h_n}(t - t_j)\, 1[A(t) > \text{Sp}]$$

(c) The concordance-index or c-index

The c-index can also be expressed as a weighted average of the area under time-specific ROC curves (AUCs)[3], obtained using the incident/dynamic definition of sensitivity and specificity:

$$\text{c-index} = \int_t \text{AUC}^{\text{I/D}}(t)\, \text{w(t)}\, dt$$

where w(t)=2 f(t) S(t), f(t) represents the distribution of failure times *T* and S(t) represents the survival time. The c-index is straightforward to estimate using the methods described above. Specifically, $\text{AUC}^{\text{I/D}}(t)$ can be estimated using the semiparametric or nonparametric approaches described in subsections (a) and (b) above, respectively, and f(t) and S(t) are derived nonparametrically, using the Kaplan-Meier estimate of the survival function.

3. Competing Risk Outcomes

Here we assume that a single event time T_i may correspond to J mutually exclusive types or causes of failure, $j = 1, 2, \ldots, J$ and we may be interested in one or more specific types. We generalize the definition of the event indicator δ_i, so that $\delta_i = 1,2, \ldots, J$ indicates a specific type or cause of failure, while $\delta_i = 0$ indicates censoring as before.

a) Cumulative (Prevalent) Cases / Dynamic Controls

For the setting of competing risk events, Saha & Heagerty[7] modified the approach of Heagerty et al[1] by using nearest neighbor estimation for the cumulative incidence function (CIF) associated with each type of failure, instead of the bivariate distribution function of the marker and time, (M, T). Estimation of sensitivity is based on the weighted conditional CIF, estimated as follows:

$$\hat{C}_j(t|M = M_i) = \sum_{s<t} \hat{S}_{\in_n}(s|M = M_i)\hat{\lambda}_j(s|M = M_i),$$

where $\hat{\lambda}_j(s|M = M_i)$ is the observed hazard for event type j at time t and $\hat{S}_{\in_n}(s|M = M_i)$ is a locally weighted Kaplan-Meier estimator of the conditional survival function, defined as before using a nearest neighbor kernel $K_{\in_n}(M_j, M_i)$ that depends on a smoothing parameter $\in_n$, with $2\in_n \in (0, 1)$ representing the percentage of individuals that are included in each neighborhood. Using the kernel function, a weighted Kaplan-Meier estimator follows:

$$\hat{S}_{\in_n}(t|M = M_i) = \prod_{s\in\mathcal{T}_n, s\le t} \left\{1 - \frac{\sum_k K_{\in_n}(M_k, M_i)\, 1(Z_k = s)\delta_k}{\sum_k K_{\in_n}(M_k, M_i)\, 1(Z_k \ge s)}\right\}$$

where $\mathcal{T}_n$ is the set of unique observed event times for the event of interest, $\delta = j$. The resulting estimate of sensitivity for event j is given by

$$\widehat{\mathrm{Se}}^{\mathrm{C}}{}_j(\mathrm{c}|\mathrm{start} = 0, \mathrm{stop} = t) = \frac{P(M > c, T \le t, event\ type = j)}{P(T \le t, event\ type = j)} = \frac{\int_c^{\infty} \hat{C}_j(t|M = u)\hat{f}_M(u)du}{\int_{-\infty}^{\infty} \hat{C}_j(t|M = u)\hat{f}_M(u)du}$$

where $f_M(s)$ is the probability density function of marker M.

To estimate specificity, Saha & Heagerty[7] also use the CIF conditional on marker M to get:

$$\widehat{\mathrm{Sp}}^{D}(\mathrm{c}|\mathrm{start} = 0, \mathrm{stop} = t) = \frac{\hat{P}(M > c, T > t)}{\hat{P}(T > t)} = \frac{\int_c^{\infty} \hat{P}(T > t, M = u)du}{\int_{-\infty}^{\infty} \hat{P}(T > t, M = u)du}$$

$$= \frac{\int_c^{\infty} \hat{P}(T > t|M = u)\hat{f}_M(u)du}{\int_{-\infty}^{\infty} \hat{P}(T > t|M = u)\hat{f}_M(u)du}$$

$$= \frac{\int_c^{\infty} [1 - \sum_j \hat{P}(T \le t, \delta = j \mid M = m)]\, \hat{f}_M(u)du}{\int_{-\infty}^{\infty} \left[1 - \sum_j \hat{P}(T \le t, \delta = j \mid M = m)\right] \hat{f}_M(u)du}$$

$$= \frac{\int_c^{\infty} [1 - \sum_j \hat{C}_j(t \mid M = m)]\, \hat{f}_M(u)du}{\int_{-\infty}^{\infty} \left[1 - \sum_j \hat{C}_j(t \mid M = m)\right] \hat{f}_M(u)du}$$

b) Incident Cases / Dynamic Controls

Saha & Heagerty[7] showed that the riskset reweighting used by Heagerty & Zheng[3] to estimate the sensitivity $P(M_i > c \mid T_i = t)$ can also be used with competing risks data. Under proportional hazards, a standard Cox model is fit for event of type *j*:

$$\lambda_j(t \mid M_i) = \lambda_{0,j}(t)\, exp(M_i\, \gamma_j)$$

where γ_j is the cause-specific hazard for event of type *j* associated with the marker. γ_j can be estimated using Maximum Partial Likelihood Estimation by censoring all other types of failure. As before, letting $R_i(t) = 1(M_i \ge t)$ denote the at-risk indicator for time *t*, $\pi_i^j(\gamma_j, t) = \frac{R_i(t)\exp(M_i\gamma_j)}{\sum_k R_k(t)\exp(M_k\gamma_j)}$ for the event of type *j* can be used to estimate the distribution of marker *M*, conditional on event *j* occurring at time *t*, so that the estimates of sensitivity and specificity are analogous to those presented by Heagerty & Zheng[3]. Specifically, we get the following semiparametric estimator of sensitivity for event type *j*:

$$\widehat{\mathrm{Se}}^{\mathrm{I}}{}_j(\mathrm{c} \mid t) = \hat{P}(M > c \mid T = t, \text{event type} = j) = \sum_k 1(M_k > c)\ \pi_k^j(\hat{\gamma}_j, t).$$

The methods can also accommodate non-proportional hazards, by replacing $\hat{\gamma}_j$ with an estimate of the time-varying coefficient, $\gamma_j(t)$, just as before. The time-varying coefficient, $\gamma_j(t)$, and subsequently $\mathrm{AUC}^{\mathrm{I/D}}(\mathrm{t})$, can be estimated using flexible semiparametric locally weighted partial likelihood methods[5] or local linear smoothing of the scaled Schoenfeld residuals.

An empirical estimator of specificity is given as

$$\widehat{\mathrm{Sp}}^{D}(\mathrm{c} \mid t) = \hat{P}(M \le c \mid T > t) = \frac{\sum_k 1(M_k > c, T_k > t)}{\sum_k 1(T_k > t)}$$

References

1. Heagerty PJ, Lumley T, Pepe MS. Time-dependent ROC curves for censored survival data and a diagnostic marker. Biometrics. 2000;56:337–344.

2. Akritas MG. Nearest neighbor estimation of a bivariate distribution under random censoring. Annals of Statistics. 1994;22:1299–1327.

3. Heagerty PJ, Zheng Y. Survival model predictive accuracy and ROC curves. Biometrics. 2005;61:92–105.

4. Xu and O'Quigley. Proportional hazard estimate of the conditional survival function. Journal of the American Statistical Association. 2000; 62:667-680.

5. Cai Z, Sun Y. Local linear estimation for time-dependent coefficients in Cox's regression models. Scand J Stat. 2003;30:93-111.

6. Saha-Chaudhuri P, Heagerty PJ. Non-parametric estimation of a time-dependent predictive accuracy curve. Biostatistics. 2013;14:42–59.

7. Saha P, Heagerty PJ. Time-Dependent Predictive Accuracy in the Presence of Competing Risks. 2010. Biometrics. 66, 999–1011.

Appendix B: Annotated R code

(R file included in Supplementary Materials – download from http://faculty.washington.edu/abansal/software.html)

```
library(survival)

install.packages("survivalROC")
install.packages("risksetROC")
library(survivalROC)
library(risksetROC)

#Download the meanrankROC package from http://faculty.washington.edu/abansal/software.html or from
#github: https://github.com/aasthaa/meanrankROC_package

source("MeanRank.q")
source("NNE-estimate.q")
source("NNE-CrossValidation.q")
source("interpolate.q")

source("dynamicTP.q")
source("NNE-estimate_TPR.q")

source("dynamicIntegrateAUC.R")

#Load in the datasets. Note: The PBC data is freely available in R.
bDat <- pbc[1:312,] #baseline data
bDat$deathEver <- bDat$status
bDat$deathEver[which(bDat$status==1)] <- 0 #censor at transplant
bDat$deathEver[which(bDat$status==2)] <- 1 #death
```

```r
#Build dataset with time-dependent covariates
pbc2 <- tmerge(pbc, pbc, id=id, death = event(time, status)) #set range
pbc2 <- tmerge(pbc2, pbcseq, id=id, ascites = tdc(day, ascites), hepato = tdc(day, hepato),
   spiders = tdc(day, spiders), edema = tdc(day, edema), chol = tdc(day, chol),
   bili = tdc(day, bili), albumin = tdc(day, albumin),
   protime = tdc(day, protime), alk.phos = tdc(day, alk.phos),
   ast = tdc(day, ast), platelet = tdc(day, platelet), stage = tdc(day, stage) )

length(unique(pbc2$id))

dim(pbc2)
pbc2 <- subset(pbc2, id>=1 & id<=312)
dim(pbc2)
length(unique(pbc2$id))
dim(bDat)


#According to documentation, some baseline values for protime and age in pbc were found to be incorrect.
 Correct values in pbcseq
bDat[1:5,]
subset(pbc2, tstart==0)[1:5,]
for(i in 1:312){
   if(pbc2$protime[which(pbc2$id==i & pbc2$tstart==0)] != bDat$protime[i])
      pbc2$protime[which(pbc2$id==i & pbc2$tstart==0)] <- bDat$protime[i]

   if(pbc2$age[which(pbc2$id==i & pbc2$tstart==0)] != bDat$age[i])
      pbc2$age[which(pbc2$id==i & pbc2$tstart==0)] <- bDat$age[i]
}

pbc2$deathEver <- pbc2$status
pbc2$deathEver[which(pbc2$status==1)] <- 0
pbc2$deathEver[which(pbc2$status==2)] <- 1

pbc2$death[which(pbc2$death==1)] <- 0
pbc2$death[which(pbc2$death==2)] <- 1
#Use 10-fold CV to get baseline scores
set.seed(49)

samples <- floor(runif(nrow(bDat), 1,11))
sampSizes <- sapply(seq(1:10), function(s){length(which(samples==s))} )
sampSizes #Check that no subsets with 0 subjects
```

```r
while(min(sampSizes)==0) {
   samples <- floor(runif(nTrain, 1,11))
   sampSizes <- sapply(seq(1:10), function(s){length(which(samples==s))} )
}

###10-fold cross-validation to get predicted baseline scores
score4Baseline_cv <- score5Baseline_cv <- rep(NA,nrow(bDat))

for(s in 1:10) {
   bDat_train <- bDat[-which(samples==s),]
   bDat_test <- bDat[which(samples==s),]

   mod <- coxph(Surv(time=time, event= deathEver) ~ log(bili) + log(protime) + edema + albumin + age,
      data=bDat_train )
   riskVals <- predict(mod, type="risk", newdata= bDat_test)
   score5Baseline_cv[which(samples==s)] <- riskVals

   mod <- coxph(Surv(time=time, event= deathEver) ~ log(protime) + edema + albumin + age, data=bDat_train )
   riskVals <- predict(mod, type="risk", newdata= bDat_test)
   score4Baseline_cv[which(samples==s)] <- riskVals
}
bDat$score4baseline <- score4Baseline_cv
bDat$score5baseline <- score5Baseline_cv


#Fit model on all baseline data, use for prediction of time-varying score
coxMod5baseline <- coxph(Surv(time=time, event= deathEver) ~ log(bili) + log(protime) + edema + albumin +
 age, data= bDat)
pbc2$score5tv <- predict(coxMod5baseline, type="risk", newdata= pbc2)

coxMod4baseline <- coxph(Surv(time=time, event= deathEver) ~ log(protime) + edema + albumin + age, data=
 bDat)
pbc2$score4tv <- predict(coxMod4baseline, type="risk", newdata= pbc2)


#####Table 2
##A. AUC_I/D
tableAUC_ID <- matrix(nrow=2, ncol=length(landmarkTimes))
tableAUC_TV_ID <- matrix(nrow=2, ncol=length(landmarkTimes))
```

```r
#Baseline risk scores
scores <- c("score4baseline","score5baseline")
for(i in 1:length(scores)) {
   currVar <- eval(parse(text=paste("bDat$", scores[i],sep="")))
   mmm <- MeanRank( survival.time= bDat$time, survival.status= bDat$deathEver, marker= currVar )
   bandwidths <- 0.05 + c(1:80)/200
   IMSEs <- vector(length=length(bandwidths))
   for(j in 1:length(bandwidths)) {
      nnnC <- nne.CrossValidate(x=mmm$time, y=mmm$mean.rank, lambda=bandwidths[j]) #CV bandwidth
      IMSEs[j] <- nnnC$IMSE
   }
   currLambdaOS <- mean(bandwidths[which(IMSEs==min(IMSEs, na.rm=T))])
   nnn <- nne(x= mmm$time, y= mmm$mean.rank, lambda=currLambdaOS, nControls=mmm$nControls) #Fixed bandwidth
   tableAUC_ID[i,] <- sapply(landmarkTimes, function(x){ interpolate( x = nnn$x, y=nnn$nne, target=x ) } )
}
rownames(tableAUC_ID) <- scores
colnames(tableAUC_ID) <- landmarkTimes/units
round(tableAUC_ID, 2)

#Updated (time-varying) risk scores
scores <- c("score4tv", "score5tv")
for(i in 1:length(scores)) {
   currVar <- eval(parse(text=paste("pbc2$", scores[i],sep="")))
   mmm <- MeanRank(survival.time=pbc2$tstop, survival.status=pbc2$death, marker=currVar, start=pbc2$tstart)
   bandwidths <- 0.05 + c(1:80)/200
   IMSEs <- vector(length=length(bandwidths))
   for(j in 1:length(bandwidths)) {
      nnnC <- nne.CrossValidate(x=mmm$time, y=mmm$mean.rank, lambda=bandwidths[j]) #CV bandwidth
      IMSEs[j] <- nnnC$IMSE
   }
   currLambdaOS <- mean(bandwidths[which(IMSEs==min(IMSEs, na.rm=T))])
   nnn <- nne(x=mmm$time, y=mmm$mean.rank, lambda=currLambdaOS, nControls=mmm$nControls )  #Fixed bandwidth
   tableAUC_TV_ID[i,] <- sapply(landmarkTimes, function(x){ interpolate( x = nnn$x, y=nnn$nne, target=x ) }
 )
}
rownames(tableAUC_TV_ID) <- scores
colnames(tableAUC_TV_ID) <- landmarkTimes/units
round(tableAUC_TV_ID, 2)

#B. c-index
round(dynamicIntegrateAUC(survival.time=bDat$time, survival.status=bDat$deathEver,
```

```
 marker=bDat$score4baseline, cutoffTime = units*10), 2)
round(dynamicIntegrateAUC(survival.time=bDat$time, survival.status=bDat$deathEver,
 marker=bDat$score5baseline, cutoffTime = units*10), 2)

round(dynamicIntegrateAUC(survival.time= pbc2$tstop, survival.status= pbc2$death, start=pbc2$tstart,
 marker=pbc2$score4tv, cutoffTime = units*10), 2)
round(dynamicIntegrateAUC(survival.time= pbc2$tstop, survival.status= pbc2$death, start=pbc2$tstart,
 marker=pbc2$score5tv, cutoffTime = units*10), 2)

#C. Sequential C/D AUCs on subsetted data at each timepoint and one year ahead to mimic landmark analysis
units <- 365.25

landmarkTimes <- c(1, 4, 6)*units
tableAUC_CD <- matrix(nrow=4, ncol=length(landmarkTimes))

timeWindow <- 1

for(j in 1:length(landmarkTimes)) {
   currData <- subset(bDat, time >= (landmarkTimes[j]))
   currDataTV <- subset(pbc2, tstart <= (landmarkTimes[j]) & tstop > (landmarkTimes[j]))

   nobs <- nrow(currData)
   out1 <- survivalROC( currData$time, currData$deathEver, marker= currData$score4baseline,
              predict.time=(landmarkTimes[j] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
   tableAUC_CD[1,j] <- out1$AUC

   out1 <- survivalROC( currData$time, currData$deathEver, marker= currData$score5baseline,
              predict.time=(landmarkTimes[j] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
   tableAUC_CD[2,j] <- out1$AUC

   nobs <- nrow(currDataTV)
   out1 <- survivalROC( currDataTV$time, currDataTV$deathEver, marker= currDataTV$score4tv,
              predict.time=(landmarkTimes[j] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
   tableAUC_CD[3,j] <- out1$AUC

   out1 <- survivalROC( currDataTV$time, currDataTV$deathEver, marker= currDataTV$score5tv,
              predict.time=(landmarkTimes[j] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
   tableAUC_CD[4,j] <- out1$AUC
}
rownames(tableAUC_CD) <- c("score4baseline","score5baseline", "score4tv", "score5tv")
colnames(tableAUC_CD) <- landmarkTimes/units
```

```
round(tableAUC_CD, 2)


####Bootstrap 95% CIs
nBoot <- 500

##A. Bootstrap CIs - Baseline markers/scores
markers <- c("score4baseline","score5baseline")
set.seed(49)
Cindex_bstrap <- matrix(nrow=nBoot, ncol=length(markers))
bstrapRes <- list()

for(b in 1:nBoot) {
   currData <- bDat[sample(x=seq(1,nrow(bDat)), size=nrow(bDat), replace = T),]
   kmfit <- survfit(Surv(time, deathEver) ~ 1, data= currData)

   currDataLM1 <- currData[which(currData$time >= (landmarkTimes[1])), ]
   currDataLM2 <- currData[which(currData$time >= (landmarkTimes[2])), ]
   currDataLM3 <- currData[which(currData$time >= (landmarkTimes[3])), ]

   aucID_scores <- NULL
   aucCD_scores <- matrix(nrow=length(scores), ncol=length(landmarkTimes))

   for(i in 1:length(markers)) {
      currVar <- eval(parse(text=paste("currData$",markers[i],sep="")))

      ### AUC I/D
      mmm <- MeanRank( survival.time= currData$time, survival.status= currData$deathEver, marker= currVar )
      nnn <- nne( x= mmm$time, y= mmm$mean.rank, lambda=0.3 )  #Fixed bandwidth
      aucID_scores <- rbind(aucID_scores,
         sapply(landmarkTimes, function(t){ interpolate( x = nnn$x, y=nnn$nne, target=t ) } ))

      ### C-index
      Cindex_bstrap[b,i] <- dynamicIntegrateAUC(survival.time=currData$time,
         survival.status= currData$deathEver, marker=currVar, cutoffTime = units*10)

      ### AUC C/D landmark
      if(markers[i]=="score4baseline" | markers[i]=="score5baseline") {
         currDataLM <- currDataLM1
         currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
         out1 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
```

```
                    predict.time=(landmarkTimes[1] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
          currDataLM <- currDataLM2
          currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
          out2 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
                    predict.time=(landmarkTimes[2] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))

          currDataLM <- currDataLM3
          currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
          out3 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
                    predict.time=(landmarkTimes[3] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
          aucCD_scores[i,] <- c(out1$AUC, out2$AUC, out3$AUC)
       }
    }
    bstrapRes[[b]] <- list(aucID_scores=aucID_scores, aucCD_scores=aucCD_scores)
}

#Get CIs for c-indices
Cindex_CIs <- round(apply(Cindex_bstrap, 2, quantile, probs=c(0.025,0.975)),2)
colnames(Cindex_CIs) <- markers
Cindex_CIs

#Get CIs for AUCs
AUC_ID_CIs <- NULL
AUC_CD_CIs <- NULL

for(t in 1:length(landmarkTimes)) {
   AUC_ID <- NULL
   AUC_CD <- NULL
   for(b in 1:nBoot) {
      AUC_ID <- cbind( AUC_ID, bstrapRes[[b]]$aucID_scores[,t] )
      AUC_CD <- cbind( AUC_CD, bstrapRes[[b]]$aucCD_scores[,t] )
   }
   AUC_ID_CI_raw <- round( apply(AUC_ID, 1, quantile, probs=c(0.025,0.975)), 2 )
   AUC_CD_CI_raw <- round( apply(AUC_CD, 1, quantile, probs=c(0.025,0.975)), 2 )

   AUC_ID_CIs <- cbind(AUC_ID_CIs, sapply(seq(2), function(x) paste("(", AUC_ID_CI_raw[1,x], ", ",
 AUC_ID_CI_raw[2,x], ")", sep="" ) ) )
   AUC_CD_CIs <- cbind(AUC_CD_CIs, sapply(seq(2), function(x) paste("(", AUC_CD_CI_raw[1,x], ", ",
 AUC_CD_CI_raw[2,x], ")", sep="" ) ) )
}
rownames(AUC_CD_CIs) <- rownames(AUC_ID_CIs) <- c("4-cov model", "5-cov model")
```

```
colnames(AUC_CD_CIs) <- colnames(AUC_ID_CIs) <- landmarkTimes/units
AUC_ID_CIs
AUC_CD_CIs

##B. Bootstrap CIs - Time-varying scores
markers <- c("score4tv","score5tv")

set.seed(49)
Cindex_bstrapTV <- matrix(nrow=nBoot, ncol=length(markers))
bstrapResTV <- list()

for(b in 1:nBoot) {
   #sample individuals
   subjs <- unique(pbc2$id)
   currSubjs <- sample(x=subjs, size=length(subjs), replace = T)
   currData <- NULL
   for(j in 1:length(currSubjs))
      currData <- rbind(currData, pbc2[which(pbc2$id==currSubjs[j]),])

   kmfit <- survfit(Surv(time=tstart, time2=tstop, event=death) ~ 1, data=currData)

   currDataLM1 <- currData[which(currData$tstart<=(landmarkTimes[1]) & currData$tstop>(landmarkTimes[1])),]
   currDataLM2 <- currData[which(currData$tstart<=(landmarkTimes[2]) & currData$tstop>(landmarkTimes[2])),]
   currDataLM3 <- currData[which(currData$tstart<=(landmarkTimes[3]) & currData$tstop>(landmarkTimes[3])),]

   aucID_scores <- NULL
   aucCD_scores <- matrix(nrow=length(scores), ncol=length(landmarkTimes))

   for(i in 1:length(markers)) {
      currVar <- eval(parse(text=paste("currData$", markers[i],sep="")))

      ### AUC I/D
      mmm <- MeanRank(survival.time=currData$tstop, survival.status=currData$death, start=currData$tstart,
         marker=currVar)
      nnn <- nne( x= mmm$time, y= mmm$mean.rank, lambda=0.3, nControls=mmm$nControls )  #Fixed bandwidth
      aucID_scores <- rbind(aucID_scores,
         sapply(landmarkTimes, function(t){ interpolate( x = nnn$x, y=nnn$nne, target=t ) } ))

      ### C-index
      Cindex_bstrapTV[b,i] <- dynamicIntegrateAUC(survival.time=currData$tstop,
         survival.status=currData$death, start=currData$tstart, marker= currVar, cutoffTime = units*10)
```

```
      ### AUC C/D landmark (for the scores only)
         currDataLM <- currDataLM1
         currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
         out1 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
                   predict.time=(landmarkTimes[1] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
         currDataLM <- currDataLM2
         currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
         out2 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
                   predict.time=(landmarkTimes[2] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
         currDataLM <- currDataLM3
         currVecLM <- eval(parse(text=paste("currDataLM$", markers[i], sep="")))
         out3 <- survivalROC( currDataLM$time, currDataLM$deathEver, marker=currVecLM,
                   predict.time=(landmarkTimes[3] + timeWindow*units), method="NNE", span=0.04*nobs^(-0.2))
         aucCD_scores[i,] <- c(out1$AUC, out2$AUC, out3$AUC)

   }
   bstrapResTV[[b]] <- list(aucID_scores, aucCD_scores)
}

#Get CIs for c-indices
Cindex_CIs <- round(apply(Cindex_bstrapTV, 2, quantile, probs=c(0.025,0.975)), 2)
colnames(Cindex_CIs) <- markers
Cindex_CIs

#Get CIs for AUCs
AUC_CD_CIs <- NULL
AUC_ID_CIs <- NULL

for(t in 1:length(landmarkTimes)) {
   AUC_CD <- NULL
   AUC_ID <- NULL
   for(b in 1:nBoot) {
      AUC_ID <- cbind( AUC_ID, bstrapResTV[[b]][[1]][,t] )
      AUC_CD <- cbind( AUC_CD, bstrapResTV[[b]][[2]][,t] )
   }
   AUC_CD_CI_raw <- round( apply(AUC_CD, 1, quantile, probs=c(0.025,0.975)), 2 )
   AUC_ID_CI_raw <- round( apply(AUC_ID, 1, quantile, probs=c(0.025,0.975)), 2 )

   AUC_CD_CIs <- cbind(AUC_CD_CIs, sapply(seq(2), function(x) paste("(", AUC_CD_CI_raw[1,x], ", ",
      AUC_CD_CI_raw[2,x], ")", sep="" ) ) )
```

```
   AUC_ID_CIs <- cbind(AUC_ID_CIs, sapply(seq(2), function(x) paste("(", AUC_ID_CI_raw[1,x], ", ",
      AUC_ID_CI_raw[2,x], ")", sep="" ) ) )
}
rownames(AUC_CD_CIs) <- rownames(AUC_ID_CIs) <- c("4-cov model", "5-cov model")
colnames(AUC_CD_CIs) <- colnames(AUC_ID_CIs) <- landmarkTimes/units
AUC_ID_CIs
AUC_CD_CIs

#C-index difference
getCindexBstrapCI <- function(nBoot, inData, markerVarName1, markerVarName2, timeVarName,
  eventVarName, cutoffTime) {
   set.seed(49)
   resultStar <- vector(length=nBoot)
   for(i in 1:nBoot) {
      datStar <- inData[sample(seq(nrow(inData)), nrow(inData), replace=T), ]

      markerVar1 <- eval(parse(text=paste("datStar$", markerVarName1, sep="")))
      markerVar2 <- eval(parse(text=paste("datStar$", markerVarName2, sep="")))
      timeVar <- eval(parse(text=paste("datStar$", timeVarName, sep="")))
      eventVar <- eval(parse(text=paste("datStar$", eventVarName, sep="")))

      kmfit <- survfit(Surv(timeVar, eventVar) ~ 1)

      ### Marker 1
      mmm <- MeanRank( survival.time= timeVar, survival.status= eventVar, marker= markerVar1 )

      #Get overlap between survival function and mmm
      meanRanks <-  mmm$mean.rank[which(mmm$time <= cutoffTime)]
      survTimes <- mmm$time[mmm$time <= cutoffTime]
      timeMatch <- match(survTimes, kmfit$time)
      S_t <- kmfit$surv[timeMatch]

      #Calculate weights for c-index
      f_t <- c( 0, (S_t[-length(S_t)] - S_t[-1]) )
      S_tao <- S_t[length(S_t)]
      weights <- (2*f_t*S_t)/(1-S_tao^2)

      Cindex1 <- sum(meanRanks * weights) #C-index

      ### Marker 2
      mmm <- MeanRank( survival.time= timeVar, survival.status= eventVar, marker= markerVar2 )
```

```
      #Get overlap between survival function and mmm
      meanRanks <-  mmm$mean.rank[which(mmm$time <= cutoffTime)]
      survTimes <- mmm$time[mmm$time <= cutoffTime]
      timeMatch <- match(survTimes, kmfit$time)
      S_t <- kmfit$surv[timeMatch]

      #Calculate weights for c-index
      f_t <- c( 0, (S_t[-length(S_t)] - S_t[-1]) )
      S_tao <- S_t[length(S_t)]
      weights <- (2*f_t*S_t)/(1-S_tao^2)

      Cindex2 <- sum(meanRanks * weights) #C-index

      resultStar[i] <- Cindex1 - Cindex2
   }
   return( quantile(resultStar, probs=c(0.025, 0.975)) )
}
getCindexBstrapCI(nBoot=500, inData=bDat, markerVarName1="score5baseline", markerVarName2="score4baseline",
   timeVarName="time", eventVarName="deathEver", cutoffTime=units*10)


###FIGURES
#Figure 2
par(mfrow=c(1,2))
par(ps=10)

#AUC I/D
mmmBaseline <- MeanRank(survival.time=bDat$time, survival.status=bDat$deathEver, marker=bDat$score5baseline)
print(length(mmmBaseline$time))
nnnBaseline <- nne(x=mmmBaseline$time, y=mmmBaseline$mean.rank, lambda=0.2, nControls=mmmBaseline$nControls)
plot( mmmBaseline$time, mmmBaseline$mean.rank, xlab="Time (years)", ylab=expression(AUC^"I/D"*"(t)"),
   ylim=c(0.4,1), col="blue", pch=21, cex=.8, axes=F, xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnnBaseline$x, nnnBaseline$nne, col="blue", lwd=2 )

mmmBaseline <- MeanRank(survival.time=bDat$time, survival.status=bDat$deathEver, marker=bDat$score4baseline)
print(length(mmmBaseline$time))
```

```
nnnBaseline <- nne(x=mmmBaseline$time, y=mmmBaseline$mean.rank, lambda=0.2, nControls=mmmBaseline$nControls)
points( mmmBaseline$time, mmmBaseline$mean.rank, col="orange", pch=21, cex=.8 )
lines( nnnBaseline$x, nnnBaseline$nne, col="orange", lwd=2 )

#ROC I/D (TPR)
fpr <- 0.1

mmmBaseline <- dynamicTP( p=fpr, survival.time= bDat$time, survival.status= bDat$deathEver,
   marker= bDat$score5baseline )
print(length(mmmBaseline$time))
nnnBaseline <- nne_TPR(x=mmmBaseline$time, y=mmmBaseline$mean.rank, lambda=0.3,
   nControls=mmmBaseline$nControls, nCases= mmmBaseline$nCases, p=fpr, survival.time= bDat$time,
   survival.status=bDat$deathEver, marker= bDat$score5baseline )  #Fixed bandwidth
plot(mmmBaseline$time, mmmBaseline$mean.rank, xlab="Time (years)",
   ylab=expression(ROC[t]^"I/D"*"(FPF=10%)"), ylim=c(0,1), col="blue", pch=21, cex=.8, axes=F,
   xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnnBaseline$x, nnnBaseline$nne, col="blue", lwd=2 )

mmmBaseline <- dynamicTP(p=fpr, survival.time=bDat$time, survival.status=bDat$deathEver,
   marker= bDat$score4baseline )
print(length(mmmBaseline$time))
nnnBaseline <- nne_TPR( x= mmmBaseline$time, y= mmmBaseline$mean.rank, lambda=0.3,
   nControls= mmmBaseline$nControls, nCases= mmmBaseline$nCases, p=fpr, survival.time= bDat$time,
   survival.status= bDat$deathEver, marker= bDat$score4baseline )  #Fixed bandwidth
points( mmmBaseline$time, mmmBaseline$mean.rank, col="orange", pch=21, cex=.8 )
lines( nnnBaseline$x, nnnBaseline$nne, col="orange", lwd=2 )
legend(x=1.5*units, y=0.9, legend=c("4 covariates", "5 covariates"), col=c("orange","blue"), lty=1, lwd=2,
   horiz=T)

#Figure 3
mmm <- MeanRank(survival.time=pbc2$tstop, survival.status= pbc2$death, marker= currVar, start=pbc2$tstart)

par(mfrow=c(1,2))
par(ps=10)

#AUC I/D
mmmTV <- MeanRank(survival.time=pbc2$tstop, survival.status=pbc2$death, marker=pbc2$score5tv,
```

```
   start=pbc2$tstart)
print(length(mmmTV$time))
nnn <- nne( x= mmmTV$time, y= mmmTV$mean.rank, lambda=0.2, nControls=mmmTV$nControls )

plot( mmmTV$time, mmmTV$mean.rank, xlab="Time (years)", ylab=expression(AUC^"I/D"*"(t)"), ylim=c(0.4,1),
   col="blue", pch=21, cex=.8, axes=F, xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnn$x, nnn$nne, col="blue", lwd=2, lty=1 )

mmmTV <- MeanRank(survival.time=pbc2$tstop, survival.status=pbc2$death, marker=pbc2$score4tv,
   start=pbc2$tstart)
print(length(mmmTV$time))
nnn <- nne( x= mmmTV$time, y= mmmTV$mean.rank, lambda=0.2, nControls=mmmTV$nControls )  #Fixed bandwidth
points( mmmTV$time, mmmTV$mean.rank, col="orange", pch=21, cex=.8)
lines( nnn$x, nnn$nne, col="orange", lwd=2, lty=1 )

#ROC I/D (TPR)
mmmTV <- dynamicTP( p=fpr, survival.time =pbc2$tstop, survival.status= pbc2$death, marker= pbc2$score5tv,
 start= pbc2$tstart )
print(length(mmmTV$time))
nnn <- nne_TPR(x=mmmTV$time, y=mmmTV$mean.rank, lambda=0.3, nControls=mmmTV$nControls, nCases=mmmTV$nCases,
   p=fpr, survival.time=pbc2$tstop, survival.status= pbc2$death, marker= pbc2$score5tv, start= pbc2$tstart)
plot( mmmTV$time, mmmTV$mean.rank, xlab="Time (years)", ylab=expression(ROC[t]^"I/D"*"(FPF=10%)"),
   ylim=c(0,1), col="blue", pch=21, cex=.8, axes=F, xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnn$x, nnn$nne, col="blue", lwd=2 )

mmmTV <- dynamicTP( p=fpr, survival.time =pbc2$tstop, survival.status= pbc2$death, marker= pbc2$score4tv,
   start=pbc2$tstart )
nnn <- nne_TPR(x=mmmTV$time, y=mmmTV$mean.rank, lambda=0.3, nControls=mmmTV$nControls, nCases=mmmTV$nCases,
   p=fpr, survival.time=pbc2$tstop, survival.status=pbc2$death, marker=pbc2$score4tv, start=pbc2$tstart )
points( mmmTV$time, mmmTV$mean.rank, col="orange", pch=21, cex=.8 )
lines( nnn$x, nnn$nne, col="orange", lwd=2 )
legend(x=2.5*units, y=0.3, legend=c("4 covariates", "5 covariates"), col=c("orange","blue"), lty=1, lwd=2,
   horiz=T)
```

```
##Figure 4: With CIs for baseline and updated risk scores from 5-covariate model using I/D approach
par(mfrow=c(1,2))
par(ps=10)

#ROC I/D
mmmBaseline <- MeanRank(survival.time=bDat$time, survival.status=bDat$deathEver, marker=bDat$score5baseline)
nnnBaseline <- nne(x=mmmBaseline$time, y=mmmBaseline$mean.rank, lambda=0.2, nControls=mmmBaseline$nControls)
plot( mmmBaseline$time, mmmBaseline$mean.rank, xlab="Time (years)", ylab=expression(AUC^"I/D"*"(t)"),
   ylim=c(0.4,1), col="lightblue", pch=21, cex=.8, axes=F, xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnnBaseline$x, nnnBaseline$nne, col="blue", lwd=2 )
lines( nnnBaseline$x, nnnBaseline$nne + 1.96*sqrt(nnnBaseline$var), col="blue", lty=2 )
lines( nnnBaseline$x, nnnBaseline$nne - 1.96*sqrt(nnnBaseline$var), col="blue", lty=2 )

mmmTV <- MeanRank( survival.time= pbc2$tstop, survival.status= pbc2$death, marker= pbc2$score5tv,
   start= pbc2$tstart )
nnn <- nne( x= mmmTV$time, y= mmmTV$mean.rank, lambda=0.2, nControls=mmmTV$nControls )  #Fixed bandwidth
plot( mmmTV$time, mmmTV$mean.rank, xlab="Time (years)", ylab=expression(AUC^"I/D"*"(t)"), ylim=c(0.4,1),
   col="lightblue", pch=21, cex=.8, axes=F, xlim=c(0,11)*units)
axis(1, at=seq(0,10,by=2)*units, labels=seq(0,10,by=2))
axis(2)
box()
abline(h=0.5, lty=2)
lines( nnn$x, nnn$nne, col="blue", lwd=2, lty=1 )
lines( nnn$x, nnn$nne + 1.96*sqrt(nnn$var), col="blue", lty=2 )
lines( nnn$x, nnn$nne - 1.96*sqrt(nnn$var), col="blue", lty=2 )
```