R Scripts and Functions – Survival Short Course

```
MISCELLANEOUS SCRIPTS

(1) Survival Curves

Define specific dataset for practice:
Hodgkin (Site=33011), Young (Age<=40), Primary (SqNo <=1),
Male (Sex=M), Detroit registry (Reg=20), Black (RacB=2)

> Dtmp1 <- LYMleu[Age <= 40 & Site==33011 & SqNo <= 1 & Sex=="M" &
       Reg=="20" & RacB=="2", c(4,5,11,13,17)]  ### 212 x 5

> SrF1 <- survfit(Surv(Tim,Dth), data=Dtmp1)
> SrF1B <- survfit(Surv(Tim,Dth), data=Dtmp1, conf.type=plain)

-----------------------------------------------------------------

(2) Utility function for parallelized KM

### Now can use parallelized format to calculate many KM
###   or Nelson-Aalen curves with standard errors and/or CIs
###   for the case of cross-classified data, e.g. all with Site=33011,
###  cross-classified by 10*((Age+9) %/% 10), Reg, Race (RacB=1,2,11),Sex
### In this case use values 0:347 for distinct times and create new
###   array e.g. of AgeInt, Reg, Race, Sex, Tim, EvtCt as starting point

## Start by coding and testing a utility function:

> KMarray <- function(Dfram, range=0:347) {
## Here we assume that Dfram is a data-fram with "Tim" and "Dth"
##  columns as the first two, and the others as categorical factors.
##  We create an array based on distinct values of the other columns
##  to contain number at risk and number of events on a single common
##  grid of points given by "range". Output array has the same
##  categorical columns, followed by "Tim" (with values in "range")
##  and also the computed values "KM", "NlsAal", "SErr" (for NlsAal)
   nfac <- ncol(Dfram)-2
   for(i in 3:(nfac+2)) Dfram[i] <- factor(Dfram[,i])
   auxtab <- table(Dfram[,c(3:(nfac+2),1)])
   ntim <- length(range)
   arrtmp <- array(0, dim=c(prod(dim(auxtab)[1:nfac])*ntim,5))
   ## Now enter the "Tim" and "Dth" values for the
   ##  nonempty combinations of factors and tim-variable.
   arrtmp[auxtab > 0, 1] <- auxtab[auxtab > 0]
   arrtmp[auxtab > 0, 2] <- aggregate(Dfram[,2], Dfram[,
```

```
      c(3:(nfac+2),1)], sum)[,nfac+2]
   ## Next make cumulative numbers at risk, then Nelson-Aalen & KM
   arrtmp <- array(arrtmp, dim=c(prod(dim(auxtab)[1:nfac]),ntim,5))
   for(i in 1:prod(dim(auxtab)[1:nfac])) {
      arrtmp[i,,1] <- rev(cumsum(rev(arrtmp[i,,1])))
      arrtmp[i,,3] <- cumprod(1-arrtmp[i,,2]/arrtmp[i,,1])
      arrtmp[i,,4] <- cumsum(arrtmp[i,,2]/arrtmp[i,,1])
      arrtmp[i,,5] <- sqrt(cumsum(arrtmp[i,,2]/(arrtmp[i,,1]*
         (arrtmp[i,,1]-arrtmp[i,,2]))))
      if(arrtmp[i,ntim,1]==0) {
         nlst <- min((1:ntim)[arrtmp[i,,1] == 0])
         arrtmp[i,nlst:ntim,3:5] <- outer(rep(1,ntim-nlst+1),
                arrtmp[i,nlst-1,3:5]) }
   ### Correction inserted 2/17/05: if the table ends with rows
   ###   where nrsk=0, then just replicate the final legitimate
   ###   KM, NlsA, and SErr entries in those rows.
   }
   array(arrtmp, dim=c(dim(auxtab)[1:nfac],ntim,5),
     dimnames= c(dimnames(auxtab)[1:nfac], list(Tim=range,
     Curv=c("nrsk","ndth","KM","NlsAal", "SErr"))))
   }

> attach(LYMleu)
> Dtmp2 <- LYMleu[Age <= 40 & Site==33011 & SqNo <= 1 & Sex=="M",
      c(17,13,5,11)]
> dim(Dtmp2)
[1] 6443    5
> detach()
> Dtmp2$Age <- (Dtmp2$Age %/% 6)*6
> Dtmp2$Rad <- as.numeric(Dtmp2$Rad != 0)
> table(Dtmp2[,3:4])
    Rad
Age  0   1
  0   19  27
  6   70 154
  12 246 424
  18 636 862
  24 796 942
  30 650 717
  36 464 436
> tmpout <- KMarray(Dtmp2)

> WeibMLE(0.5+as.numeric(dimnames(tmpout)[[3]]) -> xt,
   tmpout[5,1,,2], rsk=tmpout[5,1,,1])
[1] 0.01134801 0.72591004
```

```
------------------------------------------------------------

(3) Visual display of KM vs Weibull Estimated Cum Hazards

## Next check visually in graph:
> plot(xt, tmpout[5,1,,4], xlab="Time (months)", ylab="Cum Hazard",
     main=paste("Plots of KM & Weibull Est'd Cum Hazards","\n",
     "Age 24-29, Hodgkin, Male, No Radiation, 1st Cancer",sep=""))
  lines(xt, 0.01134801*xt^0.72591004, lty=1)
### OK, this is a really nice picture, which shows both how the
###  Weibull hazard tracks survival effectively while not
###  giving a really good fit !
  lines(xt, tmpout[5,1,,4]+tmpout[5,1,,5]*1.96, lty=4)
  lines(xt, tmpout[5,1,,4]-tmpout[5,1,,5]*1.96, lty=4)
### Note the characteristics of this plotted dataset:
###  Age <= 40 & Site==33011 (Hodgkin Lymphoma) & SqNo <= 1 & Sex=="M"
###  Further subsetted: Age 24-29 at diagnosis & No Radiation (Rad=0)
  text(locator(), "Plotted points are Nelson-Aalen est.")
  text(locator(), "Confidence intervals computed pointwise.")
> legend(locator(), legend=c("Weibull MLE cum-hazard",
     "Nelson-Aalen 95% Conf. Int."), lty=c(1,4))

------------------------------------------------------------------

(4) Illustration of Srho function for Independent-censoring
Sensitivity analysis

## Example: (using same data pictured in WeibPic1)

> plot(xt, tmpout[5,1,,3], xlab="Time (months)", ylab="Survival Prob",
   main=paste("Kaplan-Meier Curve, CI from Nelson-Aalen, &", "\n",
   "Srho  curves based on dep-cens with given  rho", sep=""))
  lines(xt, Srho(xt, 2, tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=3)
  lines(xt, Srho(xt, 1/2, tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=3)
### OK, this seems to work and to provide a good example, because
###   there was heavy censoring: can also exhibit confidence intervals
###   based on back-transforming CI for Nelson-Aalen:
  lines(xt, exp(-tmpout[5,1,,4]+1.96*tmpout[5,1,,5]), lty=1)
  lines(xt, exp(-tmpout[5,1,,4]-1.96*tmpout[5,1,,5]), lty=1)
  lines(xt, Srho(xt, 5, tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=5)
  lines(xt, Srho(xt, 1/5, tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=5)
  lines(xt, Srho(xt,100,tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=8)
  lines(xt, Srho(xt,.01,tmpout[5,1,,1], tmpout[5,1,,2])[[1]], lty=8)
> legend(locator(), legend=c("KM Conf Int","Srho, rho=.5,2",
    "Srho, rho=.2,5", "Srho, rho=0, Inf"), bty="n", lty=c(1,3,5,8))
```

```
-----------------------------------------------------------------

(5) Graphical displays, combined vs stratified survival curves

> skmFM <- array(0, dim=c(348,2,2), dimnames=list((0:347)+0.5,
        c("F","M"), c("Strat","Comb")))
  for(k in 1:2) skmFM[,k,2] <- cumprod(1 - apply(KMout3[2:6,k,
     2,3:4,,2],3,sum)/apply(KMout3[2:6,k,2,3:4,,1],3,sum))
  wts <- cbind(c(KMout3[2:6,1,2,3:4,1,1]), c(KMout3[2:6,2,2,3:4,1,1]))
  wts <- apply(wts,2, function(wcol) wcol/sum(wcol)) ## cols now sum to 1
  skmFM[,,1] <- t(matrix(KMout3[2:6,k,2,3:4,,3], nrow=10)) %*% wts

> matplot(xt,matrix(skmFM, ncol=4), typ="l", xlab="Time (Months)",
     ylab="Survival Probability", main=paste("Kaplan-Meier Curves",
     " SEER Hodgkins", "\n", "By Sex, Combined vs Stratified over Age,Stag",
     sep=""))
  legend(locator(), legend=c("FStrat","MStrat","FComb","MComb"), lty=1:4)
  text(locator(), "Total 1358 F, 1762 M")
  text(locator(), "Stratified logrank chisq = 31.6")
  text(locator(), "Combined logrank chisq = 44.7")

## Examine F vs M differences via logrank:
> survdiff(Surv(Tim,Dth) ~ Sex, data=Dtmp3,
      WBO==1 & (Stag==4 | Stag==9) & Rad==1 & AgeCat!=7 & AgeCat!=1)$chisq
[1] 44.71284  #### Highly significant logrank F vs M in combined setting

> survdiff(Surv(Tim,Dth) ~ Sex + strata(Stag,AgeCat), data=Dtmp3,
      WBO==1 & (Stag==4 | Stag==9) & Rad==1 & AgeCat!=7 & AgeCat!=1)$chisq
[1] 31.64084  ### Still quite significant but less so !!
### But this is still much more significant than the difference "looks"
###   in the picture !! But the differences we calculated
###    group-by-group and aggregated!

-----------------------------------------------------------------------

(6) Coding to create data structure for time-dependent covariates
Cox-model fitting

Time-dependent covariate model coding
### Create data frame from Dcox4 with Reg==27
> Dcoxtmp <- cbind.data.frame(Dcox4[Dcox4$Reg==27,c(3:5,7)],
      logtim = Dcox4$Tim[Dcox4$Reg==27]+.5,
      start=Dcox4$Tim[Dcox4$Reg==27]+.5,
      stop=Dcox4$Tim[Dcox4$Reg==27]+.5, event=Dcox4$Dth[Dcox4$Reg==27])
  DcoxTD <- NULL
  for(k in 1:7) {
```

```
    ### intervals of length 50, code log at midpt.
    tmin <- (k-1)*50
    tmax <- k*50
    inds <- (1:nrow(Dcoxtmp))[Dcoxtmp$logtim > tmin]
    Daux <- Dcoxtmp
    Daux$logtim <- log(tmin+.5)
    Daux$start <- tmin
    Daux$event <- Dcoxtmp$event*(Dcoxtmp$stop <= tmax)
    Daux$stop <- pmin(tmax, Dcoxtmp$stop)
    DcoxTD <- rbind.data.frame(DcoxTD, Daux[inds,])
    }
```

==========

LISTINGS OF A FEW CUSTOMIZED R FUNCTIONS USED IN THE COURSE

```
> WeibMLE <- function(tim, dth, rsk=NULL, lower=.2, upper=4) {
   ## This function implements ML estimation using "uniroot"
  ## applied to ML equations. Lower limit in search for gamma
  ## is lower =.2 by default; upper=4 by default.
  ## Note that we use rsk and dth to account for times at
  ## which 0 or multiple deaths occur. Default for rsk is NULL
  ## which we use to mean rev(1:length(tim)).
   if(sum(abs(tim-sort(tim)))>0) {
         neword <- order(tim)
         tim <- tim[neword]
         dth <- dth[neword]
         if(!is.null(rsk)) rsk <- rsk[neword] }
   if(is.null(rsk)) rsk <- rev(1:length(tim))
   tmult <- -diff(c(rsk,0))
   Ndth <- sum(dth)
   Logdth <- sum(dth*log(tim))
  ## NB. No death-times exactly 0 are allowed (eg input
  ##    should have .5 added for integer-grouped data).
   gamhat <- uniroot(function(gam, .Ndth, .Logsum, .tim, .tmult)
     .Logsum-.Ndth*sum(.tmult*.tim^gam*(log(.tim)-1/gam))/sum(
        .tmult*.tim^gam), lower=lower,upper=upper, .Ndth=Ndth,
        .Logsum=Logdth, .tim=tim[tmult>0],
        .tmult=tmult[tmult>0])$root
   lamhat <- sum(dth)/sum(tmult*tim^gamhat)
   list(parm=c(lamhat, gamhat), tmult=tmult, Ndth=Ndth) }

## Note: the outputs of this fitting function consist
##   of the fitted parameters (list-component $parm, a 2-dim vector)
##   along with vector of multiplicities of events at distinct times
##   and of numbers of observed deaths at those times.
```

```
> Srho
function (tim, rho, rsk, dth)
{
## Function to implement the Srho estimator (6) of Slud &
##   Rubinstein 1983 Biometrika, along with SE estimator.
## INPUTS: tim = sorted distinct event times,
##         rho  = rho-function values at  dtim  times.
##     rsk, dth = at-risk and observed-death counts at dtim times.
## OUTPUT list components: Srho =  vector of Srho values at dtim;
##     and (later) Srho.SE = standard-error estimator.
  ntim <- length(tim)
  Srho <- numeric(ntim)
  nmax <- if(rsk[length(rsk)]==0) match(0,rsk) - 1
       else ntim
  if(length(rho)==1) rho <- rep(rho,ntim)
  if(sum(abs(rho[1:nmax]-1) > 1.e-4)) {
       aux <- cumprod(((rsk-dth)/(rsk+dth*(rho-1)))[1:nmax])
       Srho[1:nmax] <- aux*(1+cumsum(dth[1:nmax]*(rho[1:nmax]-1)/
           c(1, aux[-nmax]))/rsk[1])
  }
  else {
     Srho[1:nmax] <- cumprod(1 - dth[1:nmax]/rsk[1:nmax])
     ##  Srho.SE <- Srho*sqrt(cumsum(dth/(rsk*(rsk-dth))
  }
  if(nmax<ntim) Srho[(nmax+1):ntim] <- Srho[nmax]
  list(Srho=Srho)
}

## Output is function Srho used in sensitivity analyses of
##   independent competing risks assumption.
```