Introduction to R and Bioconductor
Survival analysis

Benjamin Haibe-Kains[1,2]

[1]Computational Biology and Functional Genomics Laboratory, Dana-Farber Cancer Institute, Harvard School of Public Health
[2]Center for Cancer Computational Biology, Dana-Farber Cancer Institute

December 15, 2011

survival package

The vast majority of the he functions we need to do survival analysis are in the package *survival*. Check if the package *survival* is already loaded into your work space, if it isn't load the library *survival*

```
> search()
> library(survival)
```

We will work with the data set 'leukemia' containing times of death or censoring in patients with Acute Myelogenous Leukemia. The survival data are usually stored in a `Surv` object that is a one-column matrix containing the survival times and events/censoring.

```
> data(leukemia)
> head(leukemia)
```

```
  time status          x
1    9      1 Maintained
2   13      1 Maintained
3   13      0 Maintained
4   18      1 Maintained
5   23      1 Maintained
6   28      0 Maintained
```

```
> help(Surv)
> mysurv <- Surv(leukemia$time, leukemia$status)
> head(mysurv)
```

```
[1]  9  13  13+ 18  23  28+
```

Several methods for survival analysis are implemented in R, mainly in the *survival* package:

Surv - creates a survival object used as a response variable in a model formula, e.g. `Surv(time, status)`

survfit - computes an estimate of a survival curve for censored data using the Kaplan-Meier method, e.g. `survfit(Surv(time, status)  group)`

survdiff - Tests if there is a difference between two or more survival curves,
e.g. `survdiff(Surv(time, status)  group)`

survreg - regression for a parametric survival model with special case, the accelerated failure models that use a log transformation of the response.

coxph - fits a Cox proportional hazards regression model

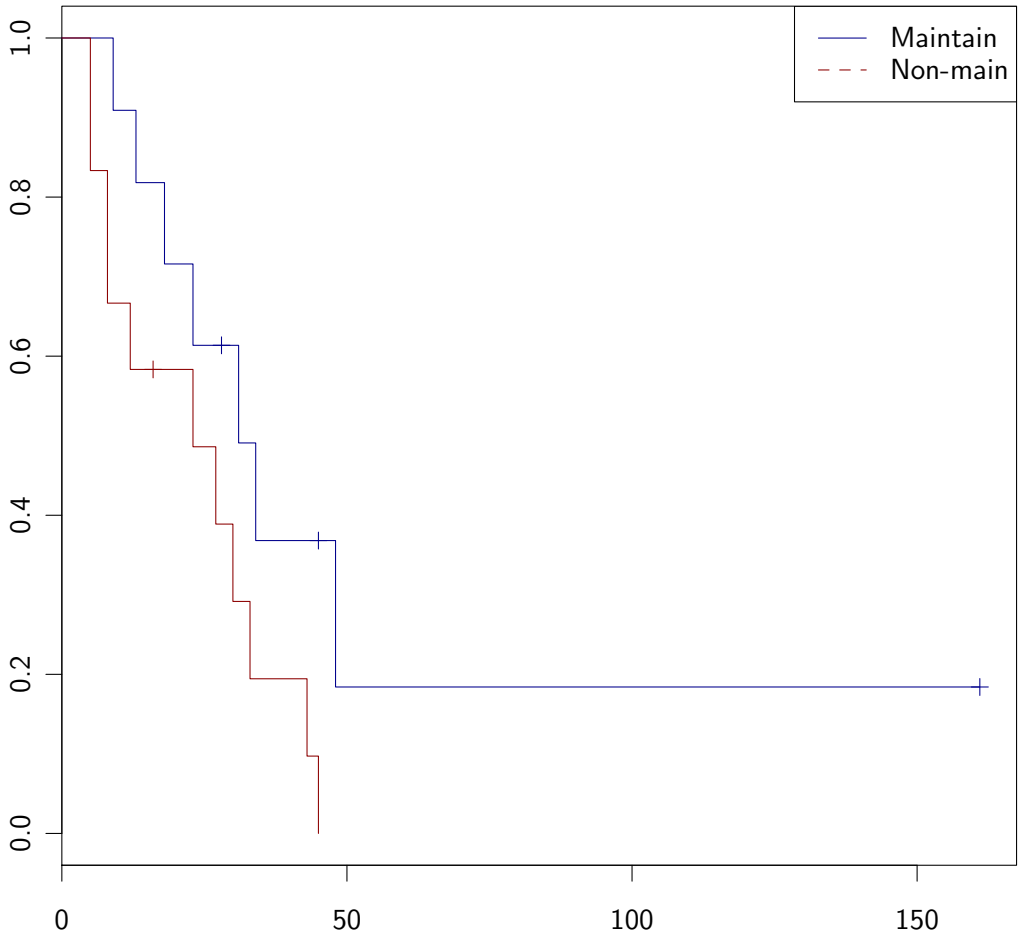
Kaplan-Meier curve estimation

We can easily draw the survival curve of patients representing the proportion of patients who survived over time. We provide the function `survfit` with the following set of arguments:

```
survfit(formula, data, weights, subset, na.action,
       newdata, individual=F, conf.int=.95, se.fit=T,
       type=c("kaplan-meier","fleming-harrington", "fh2"),
       error=c("greenwood","tsiatis"),
       conf.type=c("log","log-log","plain","none"),
       conf.lower=c("usual", "peto", "modified"))

# To see help on this function or a description of these arguments
?survfit
```

```
> leuk.km <- survfit(Surv(time, status) ~ x, data = leukemia)
> plot(leuk.km, lty = 1, col = c("darkblue", "darkred"))
> legend("topright", legend = c("Maintain", "Non-main"),
+     lty = 1:2, col = c("darkblue", "darkred"))
```



Compute confidence intervals and plot them

```
> leuk.km2 <- survfit(Surv(time, status) ~ x, data = leukemia,
+     conf.type = "log-log")
> summary(leuk.km2)
```

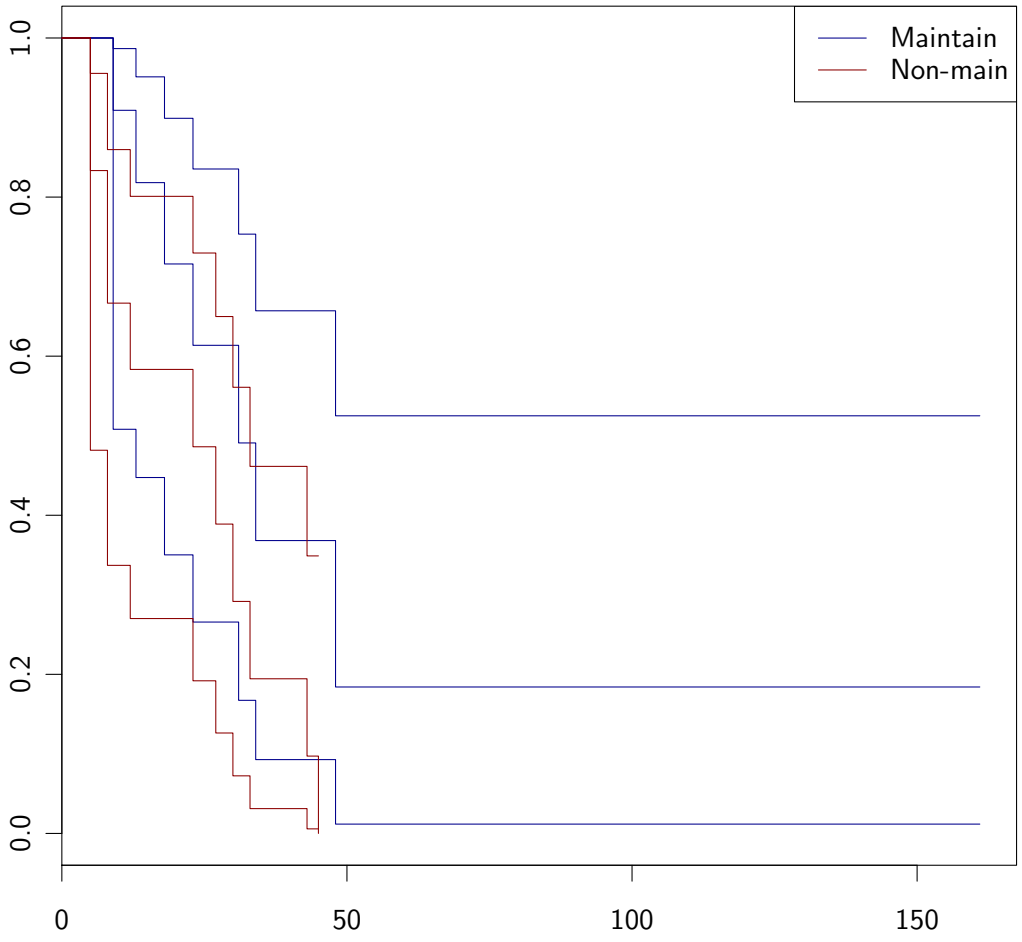
```
Call: survfit(formula = Surv(time, status) ~ x, data = leukemia, conf.type = "log-log")

                x=Maintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    9     11       1    0.909  0.0867       0.5081        0.987
   13     10       1    0.818  0.1163       0.4474        0.951
   18      8       1    0.716  0.1397       0.3502        0.899
   23      7       1    0.614  0.1526       0.2658        0.835
```

```
   31      5       1    0.491  0.1642       0.1673        0.753
   34      4       1    0.368  0.1627       0.0928        0.657
   48      2       1    0.184  0.1535       0.0117        0.525

               x=Nonmaintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    5     12       2   0.8333  0.1076      0.48171        0.956
    8     10       2   0.6667  0.1361      0.33702        0.860
   12      8       1   0.5833  0.1423      0.27014        0.801
   23      6       1   0.4861  0.1481      0.19188        0.730
   27      5       1   0.3889  0.1470      0.12627        0.650
   30      4       1   0.2917  0.1387      0.07240        0.561
   33      3       1   0.1944  0.1219      0.03120        0.461
   43      2       1   0.0972  0.0919      0.00575        0.349
   45      1       1   0.0000     NaN           NA           NA
```

```
> plot(leuk.km2, mark.time = FALSE, conf.int = TRUE,
+     lty = 1, col = c("darkblue", "darkred"))
> legend("topright", legend = c("Maintain", "Non-main"),
+     lty = 1, col = c("darkblue", "darkred"))
```



Test for difference (log-rank test)

```
> survdiff(Surv(time, status) ~ x, data = leukemia)


Call:
survdiff(formula = Surv(time, status) ~ x, data = leukemia)

                N Observed Expected (O-E)^2/E (O-E)^2/V
x=Maintained   11        7    10.69      1.27       3.4
x=Nonmaintained 12       11     7.31      1.86       3.4

 Chisq= 3.4  on 1 degrees of freedom, p= 0.0653
```

Exercise: use the `colon` dataset from the library *survival* to draw the Kaplan-Meier survival curves for the three group of patients encode by 'rx'. Are they significantly different?

Cox regression model Here is an example of Cox regression estimating the benefit of maintaining chemotherapy of with respect to the survival of the patients.

```
> leuk.ph <- coxph(Surv(time, status) ~ x, data = leukemia)
> summary(leuk.ph)


Call:
coxph(formula = Surv(time, status) ~ x, data = leukemia)

  n= 23, number of events= 18

                coef exp(coef) se(coef)     z Pr(>|z|)
xNonmaintained 0.9155    2.4981   0.5119 1.788   0.0737 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

               exp(coef) exp(-coef) lower .95 upper .95
xNonmaintained     2.498     0.4003    0.9159     6.813

Concordance= 0.619  (se = 0.073 )
Rsquare= 0.137   (max possible= 0.976 )
Likelihood ratio test= 3.38  on 1 df,   p=0.06581
Wald test            = 3.2  on 1 df,   p=0.07371
Score (logrank) test = 3.42  on 1 df,   p=0.06454
```

It is not trivial to estimate the relevance of a variable for survival modeling. If this variable is categorical, you can draw the survival curves and statistically compare them. If the variable under interest is continuous you can arbitrarily discretize it (not advisable) or use many existing performance criteria published so far for survival analysis: hazard ratio (see `coxph`), D.index, concordance.index, time-dependent ROC curve, Brier score,... The *survcomp* package contains functions to estimate these criteria.

Exercise: use the `colon` dataset from the library *survival* to fit a Cox regression model using all the variables. Which are the significant variables?

survcomp package

The *survcomp* package implements function to assess and compare the performance of risk/survival prediction models.

Let's first install and load the *survcomp* package

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("survcomp")
```

```
> library(survcomp)
```

Small gene expression datasets are included in the package, especially we will be using the expressionSet objects vdx7g and transbig7g as training and validation sets respectively.

```
> data(breastCancerData)
```

```
> print(vdx7g)
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 7 features, 344 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: VDX_3 VDX_5 ... VDX_2038 (344 total)
  varLabels: samplename dataset ... e.os (21 total)
  varMetadata: labelDescription
featureData
  featureNames: 205225_at 216836_s_at ... 202763_at (7 total)
  fvarLabels: probe Gene.title ... GO.Component.1 (22 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 17420468
Annotation: hgu133a
```

```
> print(transbig7g)
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 7 features, 198 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: VDXGUYU_4002 VDXGUYU_4008 ... VDXRHU_5240 (198 total)
  varLabels: samplename dataset ... e.os (21 total)
  varMetadata: labelDescription
featureData
  featureNames: 205225_at 216836_s_at ... 202763_at (7 total)
  fvarLabels: probe Gene.title ... GO.Component.1 (22 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 17545524
Annotation: hgu133a
```

Now let's fit a multivariate Cox model with the 7 genes included in vdx7g.The survival and gene expression data are stored in the phenotypic information of the expressionSet object.

```
> ## clincal information
> print(head(phenoData(vdx7g)@data))
```

```
      samplename dataset series id        filename size age er grade pgr her2
VDX_3      VDX_3     VDX    VDX  3 GSM36793.CEL.gz   NA  36  0    NA  NA   NA
VDX_5      VDX_5     VDX    VDX  5 GSM36796.CEL.gz   NA  47  1     3  NA   NA
VDX_6      VDX_6     VDX    VDX  6 GSM36797.CEL.gz   NA  44  0     3  NA   NA
VDX_7      VDX_7     VDX    VDX  7 GSM36798.CEL.gz   NA  41  0     3  NA   NA
VDX_8      VDX_8     VDX    VDX  8 GSM36800.CEL.gz   NA  70  0    NA  NA   NA
VDX_9      VDX_9     VDX    VDX  9 GSM36801.CEL.gz   NA  62  1     3  NA   NA
      brca.mutation e.dmfs t.dmfs node t.rfs e.rfs treatment tissue t.os e.os
VDX_3            NA      0   3072    0    NA    NA         0      1   NA   NA
VDX_5            NA      0   3589    0    NA    NA         0      1   NA   NA
VDX_6            NA      1    274    0    NA    NA         0      1   NA   NA
VDX_7            NA      0   3224    0    NA    NA         0      1   NA   NA
VDX_8            NA      1   1125    0    NA    NA         0      1   NA   NA
VDX_9            NA      0   3802    0    NA    NA         0      1   NA   NA
```

```
> dd <- phenoData(vdx7g)@data[, c("t.dmfs", "e.dmfs")]
> colnames(dd) <- c("time", "event")
> ## append gene expressions
> ge <- t(exprs(vdx7g))
> dd <- cbind(dd, ge)
```

Let;s now fit a Cox regression model with the 7 genes.

```
> mm <- coxph(Surv(time, event) ~ ., data = data.frame(dd))
> print(summary(mm))
```

```
Call:
coxph(formula = Surv(time, event) ~ ., data = data.frame(dd))

  n= 344, number of events= 118

                 coef exp(coef)  se(coef)      z Pr(>|z|)
X205225_at    0.118917  1.126276  0.048677  2.443   0.0146 *
X216836_s_at -0.053586  0.947824  0.064541 -0.830   0.4064
X208079_s_at  0.533153  1.704298  0.103690  5.142 2.72e-07 ***
X211668_s_at  0.010908  1.010968  0.091350  0.119   0.9049
X211527_x_at  0.027561  1.027944  0.059753  0.461   0.6446
X209969_s_at -0.192074  0.825246  0.088610 -2.168   0.0302 *
X202763_at    0.008054  1.008087  0.238373  0.034   0.9730
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

             exp(coef) exp(-coef) lower .95 upper .95
X205225_at      1.1263     0.8879    1.0238    1.2390
X216836_s_at    0.9478     1.0550    0.8352    1.0756
X208079_s_at    1.7043     0.5868    1.3909    2.0884
X211668_s_at    1.0110     0.9892    0.8452    1.2092
X211527_x_at    1.0279     0.9728    0.9143    1.1557
X209969_s_at    0.8252     1.2118    0.6937    0.9818
X202763_at      1.0081     0.9920    0.6318    1.6084

Concordance= 0.631  (se = 0.027 )
Rsquare= 0.08   (max possible= 0.979 )
Likelihood ratio test= 28.71  on 7 df,   p=0.0001635
Wald test            = 30.25  on 7 df,   p=8.54e-05
Score (logrank) test = 30.34  on 7 df,   p=8.232e-05
```

```
> predtr <- predict(mm, newdata = data.frame(dd), type = "risk")
```

We can use this Cox model to predict the risk in `transvig7g` independent dataset.

```
> dd2 <- t(exprs(transbig7g))
> predts <- predict(mm, newdata = data.frame(dd2), type = "risk")
```

Using the performance estimate implemented in the *survcomp* package, we can assess the performance of this Cox model in the independent dataset. To learn more about the different performance criteria, we encourage you to consult the references listed in each function.

```
> survdd2 <- phenoData(transbig7g)@data[, c("t.dmfs",
+     "e.dmfs")]
> colnames(survdd2) <- c("time", "event")
```

Hazard ratio

```
> perf <- hazard.ratio(x = predts, surv.time = survdd2[,
+     "time"], surv.event = survdd2[, "event"], na.rm = TRUE)
> print(perf[1:6])


$hazard.ratio
[1] 1.1831

$coef
[1] 0.1681379

$se
[1] 0.1092791
```

```
$lower
[1] 0.9549987

$upper
[1] 1.465683

$p.value
[1] 0.1232623
```

D index

```
> perf <- D.index(x = predts, surv.time = survdd2[,
+     "time"], surv.event = survdd2[, "event"], na.rm = TRUE)
> print(perf[1:6])


$d.index
[1] 1.278633

$coef
[1] 0.2457912

$se
[1] 0.2108593

$lower
[1] 0.8457889

$upper
[1] 1.93299

$p.value
[1] 0.2437993
```

Concordance index

```
> perf <- concordance.index(x = predts, surv.time = survdd2[,
+     "time"], surv.event = survdd2[, "event"], method = "noether",
+     na.rm = TRUE)
> print(perf[1:5])


$c.index
[1] 0.5596659

$se
[1] 0.04160814
```

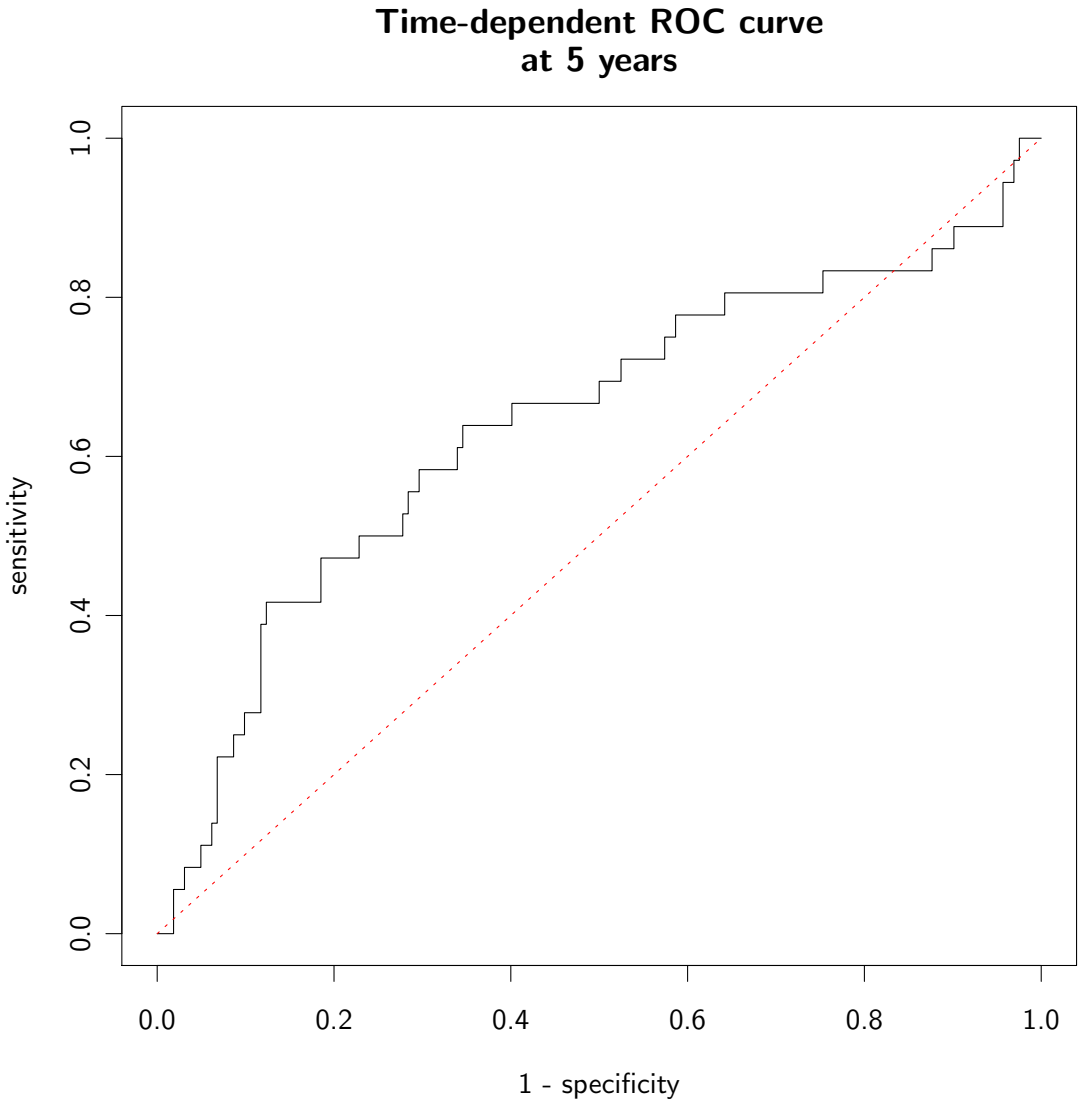
```
$lower
[1] 0.4781154

$upper
[1] 0.6412163

$p.value
[1] 0.07578681
```

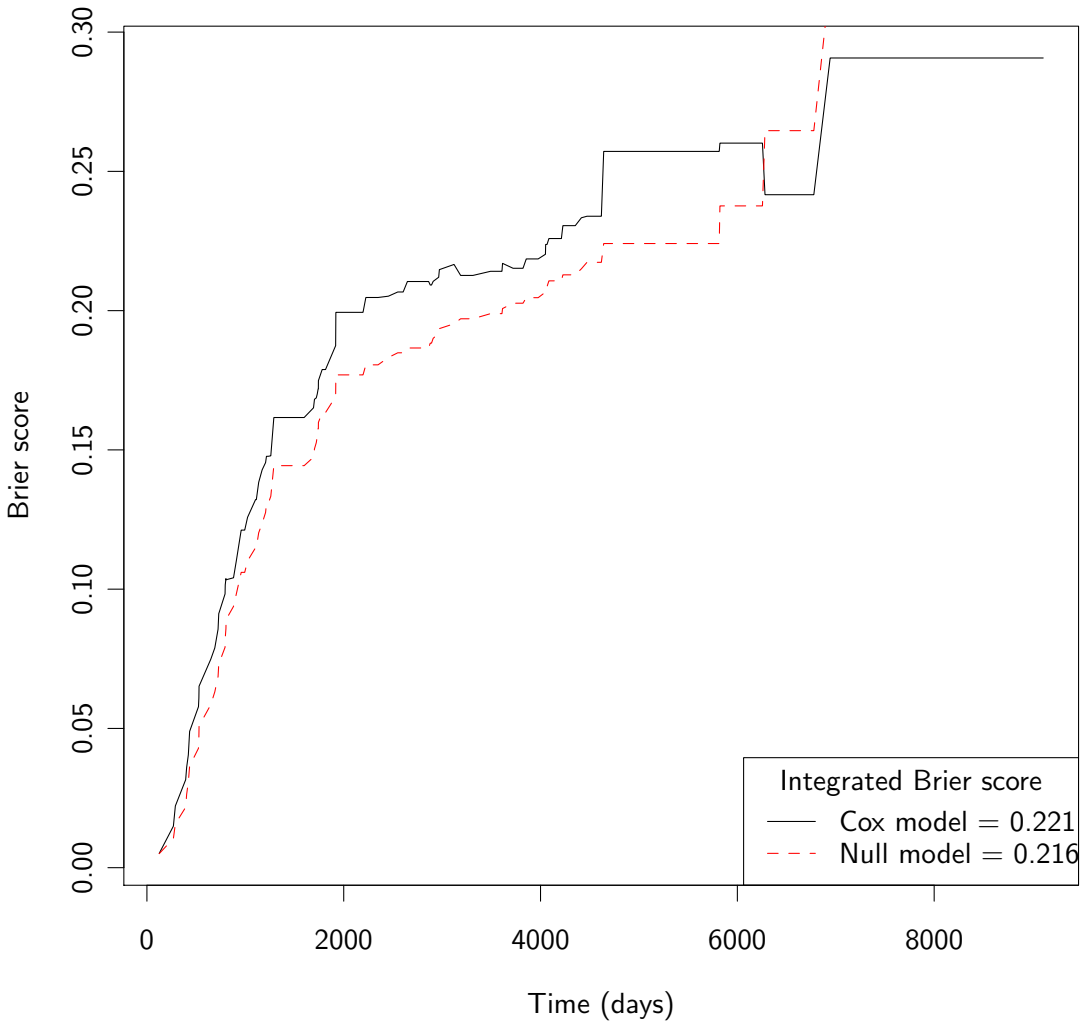
Time-dependent ROC curve

```
> perf <- tdrocc(x = predts, surv.time = survdd2[, "time"],
+     surv.event = survdd2[, "event"], time = 5 * 365, na.rm = TRUE)
> plot(x = 1 - perf$spec, y = perf$sens, type = "l",
+     xlab = "1 - specificity", ylab = "sensitivity", xlim = c(0,
+         1), ylim = c(0, 1), main = "Time-dependent ROC curve\nat 5 years")
> lines(x = c(0, 1), y = c(0, 1), lty = 3, col = "red")
```



Brier score

```
> ddtr <- cbind(time = dd[, "time"], event = dd[, "event"],
+     score = predtr)
> ddts <- cbind(time = survdd2[, "time"], event = survdd2[,
+     "event"], score = predts)
> perf <- sbrier.score2proba(data.tr = data.frame(ddtr),
+     data.ts = data.frame(ddts), method = "cox")
> plot(x = perf$time, y = perf$bsc, xlab = "Time (days)",
+     ylab = "Brier score", type = "l")
> ## null model
> ddtr <- cbind(time = dd[, "time"], event = dd[, "event"],
+     score = 1)
> ddts <- cbind(time = survdd2[, "time"], event = survdd2[,
+     "event"], score = 1)
> perfnull <- sbrier.score2proba(data.tr = data.frame(ddtr),
+     data.ts = data.frame(ddts), method = "cox")
> lines(x = perfnull$time, y = perfnull$bsc, col = "red",
+     lty = 2)
> ## legend
> legend("bottomright", title = "Integrated Brier score",
+     legend = c(sprintf("Cox model = %.3g", perf$bsc.integrated),
+         sprintf("Null model = %.3g", perfnull$bsc.integrated)),
+     col = c("black", "red"), lty = c(1, 2))
```


Brier score
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
2000
4000
6000
8000
Time (days)
Integrated Brier score
Cox model = 0.221
Null model = 0.216