SurvComp: a package for performance assessment and comparison for survival analysis

Benjamin Haibe-Kains[1,2,3], Markus Schröder[1,5], John Quackenbush[1,2], Christos Sotiriou[4], and Gianluca Bontempi[3]

[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
[3]Machine Learning Group, Université Libre de Bruxelles
[4]Breast Cancer Translational Research Laboratory, Institut Jules Bordet, Université Libre de Bruxelles
[5]Bioinformatics Resource Facility, Center for Biotechnology, Bielefeld University, Germany

March 3, 2011

Contents

1 Introduction

The *SurvComp* package is providing functions to assess and to statistically compare the performance of risk prediction (survival) models. It includes (i) implementation of state-of-the-art statistics developed to measure the performance of risk prediction models and (ii) to combine these statistics estimated from multiple datasets using a meta-analytical framework, functions (iii) to visualize those measurements in a clear and compact way, and (iv) to statistically compare the performance of competitive models.

1.1 Installation

SurvComp requires that *survival*, *ipred*, *prodlim*, *survivalROC*, *SuppDists*, *bootstrap* and *R* ($>=$ 2.3.0) are installed. These should be installed automatically when you install *SurvComp*. To install *SurvComp*, source biocLite from bioconductor:

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

Load the *SurvComp*, into your current workspace:

```
> library(survcomp)
```

1.2 Further help

To view the *SurvComp* description and a summary of all the functions within *SurvComp*, type the following:

```
> library(help = survcomp)
```

1.3 Citing

We are delighted if you use this package. Please do email us if you find a bug or have a suggestion. We would be very grateful if you could cite:

B. Haibe-Kains, C. Desmedt, C. Sotiriou and G. Bontempi (2008) A comparative study of survival models for breast cancer prognostication based on microarray data: does a single gene beat them all? *Bioinformatics* **24(19):**2200-2208.

2 A use case: from expression data to survival analysis

We will very briefly demonstrate some of the functions in *SurvComp*. We use the `breastCancerData` datafile for demonstration purposes, it includes subsets of the datasets *breast-*

CancerMAINZ, *breastCancerTRANSBIG*, *breastCancerUPP*, *breastCancerUNT*, *breastCancerVDX* and *breastCancerNKI*, available as experimental datapackages on Bioconductor. The six datasets in `breastCancerData` contain the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as uPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008 [1]. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

2.1 Overview

To use the `ExpressionSet` object we have to load the *Biobase* package. We also make use of the package *xtable* in order to visualize some of the results as tables in this Vignette.

```
> library(Biobase)
> library(xtable)
```

Loading the `breastCancerData` object will results in 6 new objects. If you execute `ls()` you will see `mainz7g`,`transbig7g`, `upp7g`, `unt7g`, `vdx7g` and `nki7g`. More details about these datasets are available in the `breastCancerData` manpage (`?breastCancerData`).

```
> data(breastCancerData)
> mainz7g
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 7 features, 200 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: MAINZ_BC6001 MAINZ_BC6002 ... MAINZ_BC6232 (200 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: 18593943
Annotation: hgu133a
```

Before we can start the analysis, we have to define the annotation for the mentioned seven genes, the datasets we use and a few help-variables. We define the gene symbol list (`gsList`), the entrez-gene ID list (`gidList`), the probe names for the Agilent microarray (`probesNKI`),

	Gene Symbol	Gene ID	Probes Agilent	Probes Affy
1	esr1	2099	NM_000125	205225_at
2	erbb2	2064	NM_004448	216836_s_at
3	aurka	6790	NM_003600	208079_s_at
4	plau	5328	NM_002658	211668_s_at
5	vegfa	7422	NM_003376	211527_x_at
6	stat1	6772	NM_007315	209969_s_at
7	casp3	836	NM_004346	202763_at

Table 1: Overview of the annotation of the seven genes.

the probe names for the Affymetrix microarray (`probesAffy`), a list containing the dataset names (`datasetList`), spaces for displaying the text in the forestplot at the right place (`myspace` and `mybigspace`) and `tc` for setting the censored time to 10 years. We converted the gene symbols for each gene to lowercase for better separation from the datasets. Table 1 gives an overview of the gene annotation.

```
> gsList <- tolower(fData(mainz7g)[, "Gene.symbol"])
> gidList <- fData(mainz7g)[, "Gene.ID"]
> probesNKI <- as.character(fData(nki7g)[, "probe"])
> probesAffy <- fData(mainz7g)[, "probe"]
> datasetList <- c("MAINZ", "TRANSBIG", "UPP", "UNT", "VDX", "NKI",
+     "", "Overall")
> myspace <- " "
> mybigspace <- "    "
> tc <- 10 * 365
```

2.2 Computing concordance index, D index and hazard ratio

To compute the concordance index [2, 3] for each gene in each dataset, we have to call the `concordance.index()`) function for each dataset. See '`?concordance.index`' for details. The following command shows the computation of the concordance index for each gene in the `mainz7g` dataset.

```
> cindexall.mainz.small <- t(apply(X = exprs(mainz7g), MARGIN = 1,
+     function(x, y, z) {
+         tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+             method = "noether", na.rm = TRUE)
+         return(c(cindex = tt$c.index, cindex.se = tt$se, lower = tt$lower,
+             upper = tt$upper))
```

	Min	Max
MAINZ	4.05	14.57
TRANSBIG	4.87	15.18
UPP	4.13	11.22
UNT	-5.04	3.77
VDX	2.77	15.61
NKI	-1.62	0.93

Table 2: Overview of the gene expression ranges in the six datasets.

```
+     }, y = pData(mainz7g)[, "t.dmfs"], z = pData(mainz7g)[, "e.dmfs"]))
```

To compute the D index [4] for each gene in each dataset, we have to call the `D.index()` function. See '`?D.index`' for details. The following command shows the computation of the D index for each gene in the `mainz7g` dataset.

```
> dindexall.mainz.small <- t(apply(X = exprs(mainz7g), MARGIN = 1,
+     function(x, y, z) {
+         tt <- D.index(x = x, surv.time = y, surv.event = z, na.rm = TRUE)
+         return(c(dindex = tt$d.index, dindex.se = tt$se, lower = tt$lower,
+             upper = tt$upper))
+     }, y = pData(mainz7g)[, "t.dmfs"], z = pData(mainz7g)[, "e.dmfs"]))
```

To compute the hazard ratio [5] for each gene in each dataset, we have to call the `hazard.ratio()` function. See `?hazard.ratio` for details. Before we compute the hazard ratio, we have to rescale the gene expression data for each dataset to a comparable scale, since the Affymetrix and Agilent microarrays have a different range of their gene expression, which would affect the hazard ratio computation. Table 2 gives an overview of the gene expression ranges in the six datasets that are included in `breastCancerData`.

Therefore we use the following function to rescale the gene expression values to lie approximately in [-1,1], robust to extreme values (possibly outliers).

```
> rescale <- function(x, na.rm = FALSE, q = 0.05) {
+     ma <- quantile(x, probs = 1 - (q/2), na.rm = na.rm)
+     mi <- quantile(x, probs = q/2, na.rm = na.rm)
+     x <- (x - mi)/(ma - mi)
+     return((x - 0.5) * 2)
+ }
```

The following command shows the rescaling and the computation of the hazard ratio for each gene in the mainz7g dataset.

```
> hratio.mainz.small <- t(apply(X = rescale(exprs(mainz7g), q = 0.05,
+     na.rm = TRUE), MARGIN = 1, function(x, y, z) {
+     tt <- hazard.ratio(x = x, surv.time = y, surv.event = z,
+         na.rm = TRUE)
+     return(c(hratio = tt$hazard.ratio, hratio.se = tt$se, lower = tt$lower,
+         upper = tt$upper))
+ }, y = pData(mainz7g)[, "t.dmfs"], z = pData(mainz7g)[, "e.dmfs"]))
```

To get an overall estimate over all datasets for the concordance index from each gene, we iterate over all the concordance indices of all datasets and combine them with the combine.est() function [6] and recalculate the lower- and upper border accordingly. We do that for the D indices and the hazard ratios in the same way.

2.3 Combining estimations across datasets

```
> tt <- as.data.frame(NULL)
> for (i in 1:7) {
+     tt <- rbind(tt, combine.est(x = cbind(cindexall.mainz.small[i,
+         "cindex"], cindexall.transbig.small[i, "cindex"], cindexall.upp.small[i,
+         "cindex"], cindexall.unt.small[i, "cindex"], cindexall.vdx.small[i,
+         "cindex"], cindexall.nki.small[i, "cindex"]), x.se = cbind(cindexall.mainz.sm
+         "cindex.se"], cindexall.transbig.small[i, "cindex.se"],
+         cindexall.upp.small[i, "cindex.se"], cindexall.unt.small[i,
+             "cindex.se"], cindexall.vdx.small[i, "cindex.se"],
+         cindexall.nki.small[i, "cindex.se"]), ))
+ }
> tt$lower <- tt$estimate + qnorm(0.025, lower.tail = TRUE) * tt$se
> tt$upper <- tt$estimate + qnorm(0.025, lower.tail = FALSE) *
+     tt$se
> rownames(tt) <- gsList
> colnames(tt) <- c("cindex", "cindex.se", "lower", "upper")
> ccindex <- tt
```

The combined concordance indices for the six datasets are shown in table 3.
The combined log2 D indices for the six datasets are shown in table 4.
The combined log2 hazard ratios for the six datasets are shown in table 5.

	cindex	cindex.se	lower	upper
esr1	0.46	0.02	0.43	0.49
erbb2	0.50	0.02	0.47	0.53
aurka	0.64	0.01	0.62	0.67
plau	0.52	0.01	0.49	0.55
vegfa	0.56	0.01	0.53	0.59
stat1	0.53	0.01	0.51	0.56
casp3	0.52	0.01	0.50	0.55

Table 3: Combined concordance indices of each gene for the six datasets.

	dindex	dindex.se	lower	upper
esr1	-0.17	-3.64	-0.45	0.07
erbb2	0.09	-3.62	-0.14	0.29
aurka	0.96	-3.66	0.84	1.07
plau	0.24	-3.63	0.03	0.42
vegfa	0.45	-3.65	0.28	0.61
stat1	0.19	-3.69	-0.01	0.37
casp3	0.19	-3.66	-0.02	0.37

Table 4: Combined log2 D indices of each gene for the six datasets.

	hratio	hratio.se	lower	upper
esr1	-0.27	-3.63	-0.57	-0.01
erbb2	0.38	-3.11	0.10	0.61
aurka	2.07	-2.59	1.95	2.18
plau	0.84	-2.34	0.50	1.13
vegfa	0.93	-2.79	0.69	1.13
stat1	0.48	-2.68	0.12	0.76
casp3	3.24	-1.23	3.11	3.36

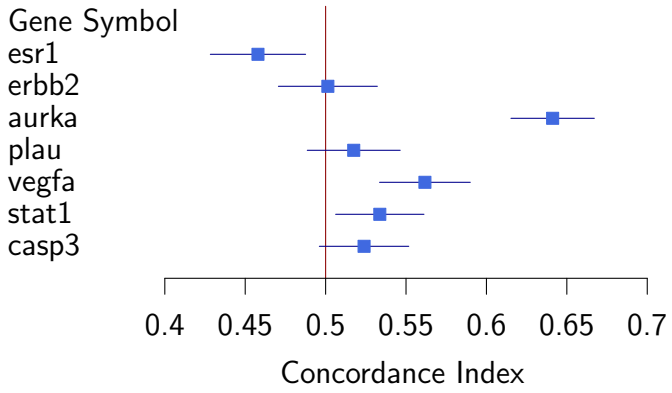
Table 5: Combined log2 hazard ratios of each gene for the six datasets.

2.4 The forestplot.surv

To display the combined concordance indices of each genes over all datasets, we use the `forestplot.surv()` function [7]. The resulting forestplot for all concordance indices is:

```
> labeltext <- cbind(c("Gene Symbol", gsList), c(rep(myspace, 8)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, ccindex$cindex)
```

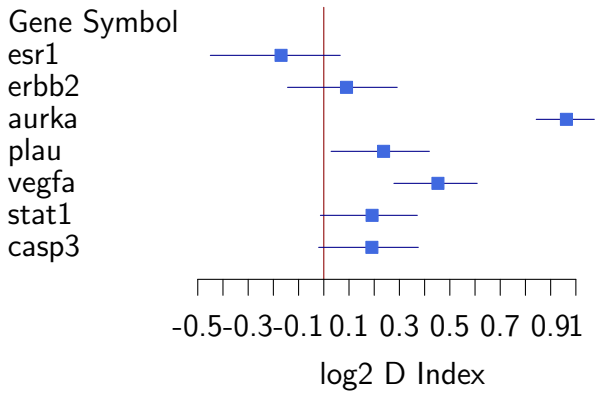
```
> r.lower <- c(NA, ccindex$lower)
> r.upper <- c(NA, ccindex$upper)
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0.5, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(0.4, 0.7, 0.05), xlab = paste("Concordance
Index",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(0.4, 1))
```



The resulting forestplot for all D indices is:

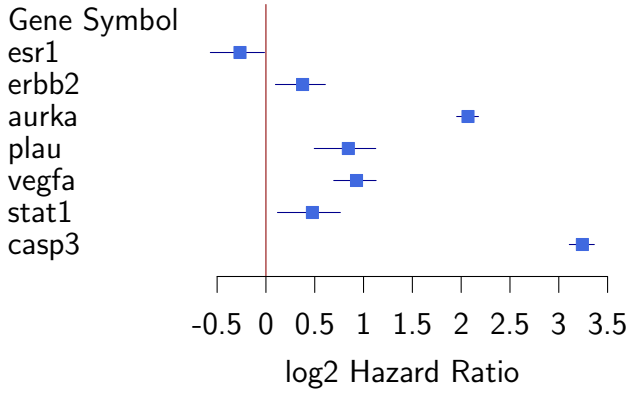
```
> labeltext <- cbind(c("Gene Symbol", gsList), c(rep(myspace, 8)))
```

```
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, log2(cdindex$dindex))
> r.lower <- c(NA, log2(cdindex$lower))
> r.upper <- c(NA, log2(cdindex$upper))
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(-0.5, 1, 0.1), xlab = paste("log2 D In-
dex",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(-0.5, 1.25))
```



The resulting forestplot for all hazard ratios is:

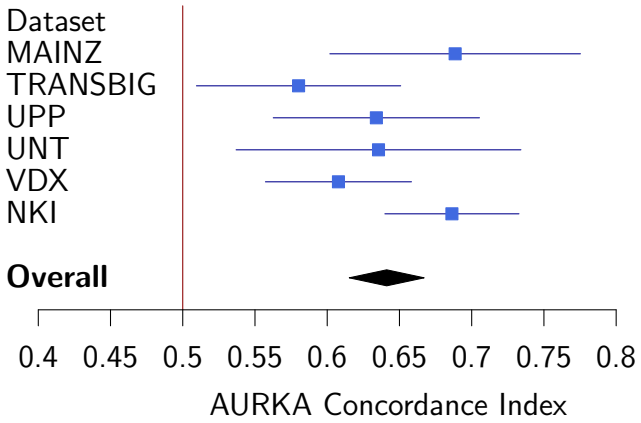
```
> labeltext <- cbind(c("Gene Symbol", gsList), c(rep(mybigspace,
+     8)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, log2(chratio$hratio))
> r.lower <- c(NA, log2(chratio$lower))
> r.upper <- c(NA, log2(chratio$upper))
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(-0.5, 3.5, 0.5), xlab = paste("log2 Haz-
ard Ratio",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(-0.75, 3.5))
```



Taking a more specific look, e.g. at the genes AURKA and VEGF, we create the forestplot the same way as before, showing the concordance indices for both genes in each dataset and the combined estimation over all datasets.

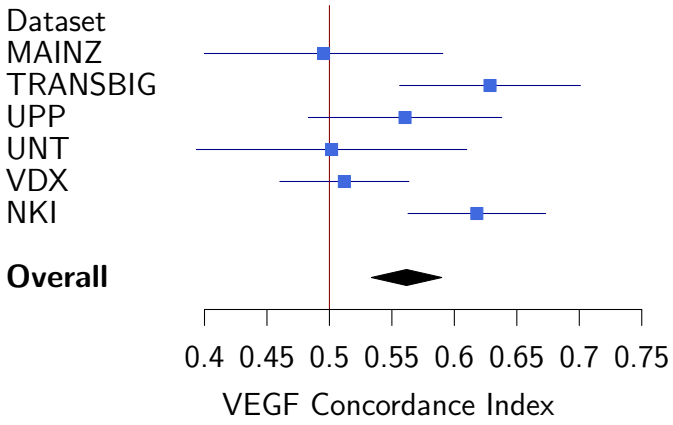
```
> tt <- rbind(cindexall.mainz.small[3, ], cindexall.transbig.small[3,
+     ], cindexall.upp.small[3, ], cindexall.unt.small[3, ], cindexall.vdx.small[3,
+     ], cindexall.nki.small[3, ], NA, as.numeric(ccindex[3, ]))
> rownames(tt) <- datasetList
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", datasetList), c(rep(mybigspace,
+     length(datasetList) + 1)))
> bs <- rep(0.5, nrow(labeltext))
```

```
> r.mean <- c(NA, tt$cindex)
> r.lower <- c(NA, tt$lower)
> r.upper <- c(NA, tt$upper)
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0.5, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(0.4, 0.8, 0.05), xlab = paste("AURKA Con-
cordance Index",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(0.5, 1), is.summary = (c(rep(FALSE, 8), TRUE)))
```




```
> tt <- rbind(cindexall.mainz.small[5, ], cindexall.transbig.small[5,
```

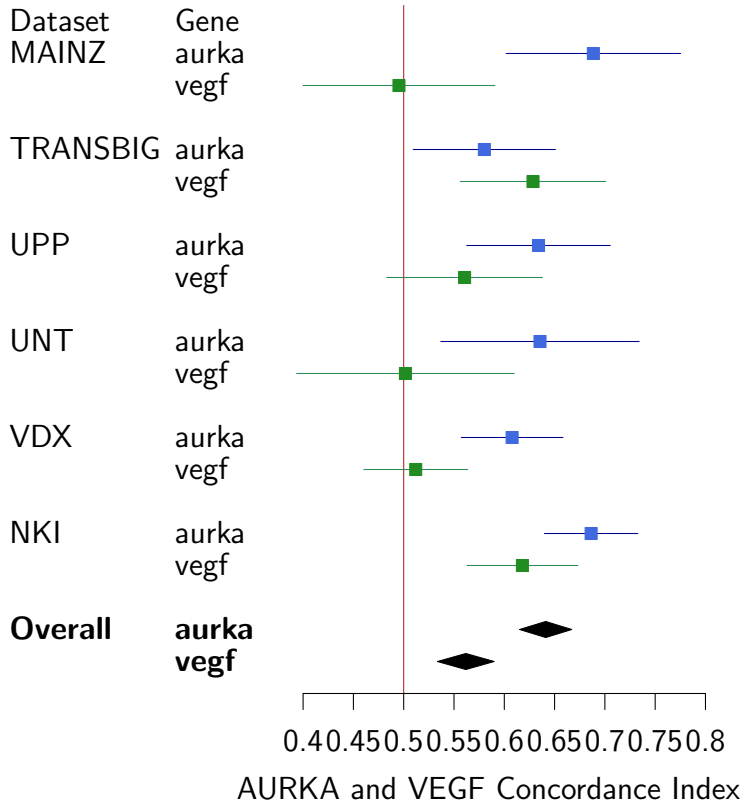
```
+     ], cindexall.upp.small[5, ], cindexall.unt.small[5, ], cindexall.vdx.small[5,
+     ], cindexall.nki.small[5, ], NA, as.numeric(ccindex[5, ]))
> rownames(tt) <- datasetList
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", datasetList), c(rep(mybigspace,
+     length(datasetList) + 1)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, tt$cindex)
> r.lower <- c(NA, tt$lower)
> r.upper <- c(NA, tt$upper)
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0.5, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(0.4, 0.75, 0.05), xlab = paste("VEGF Con-
cordance Index",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(0.3, 0.75), is.summary = (c(rep(FALSE, 8), TRUE)))
```



More advanced displaying of the genes AURKA and VEGF in a single forestplot with different colors and labels is possible:

```
> tt <- rbind(cindexall.mainz.small[3, ], cindexall.mainz.small[5,
+     ], NA, cindexall.transbig.small[3, ], cindexall.transbig.small[5,
+     ], NA, cindexall.upp.small[3, ], cindexall.upp.small[5, ],
+     NA, cindexall.unt.small[3, ], cindexall.unt.small[5, ], NA,
+     cindexall.vdx.small[3, ], cindexall.vdx.small[5, ], NA, cindexall.nki.small[3,
+         ], cindexall.nki.small[5, ], NA, as.numeric(ccindex[3,
+         ]), as.numeric(ccindex[5, ]))
> rownames(tt) <- c("MAINZa", "MAINZv", "a", "TRANSBIGa", "TRANSBIGv",
+     "b", "UPPa", "UPPv", "c", "UNTa", "UNTv", "d", "VDXa", "VDXv",
```

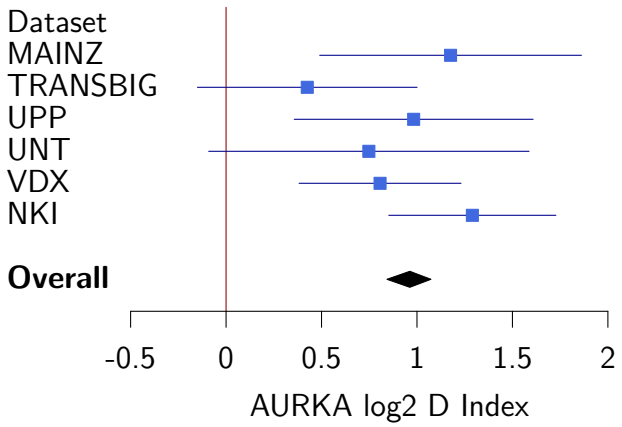
```
+     "e", "NKIa", "NKIv", "f", "ALLa", "ALLv")
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", "MAINZ", NA, NA, "TRANSBIG",
+     NA, NA, "UPP", NA, NA, "UNT", NA, NA, "VDX", NA, NA, "NKI",
+     NA, NA, "Overall", NA), c("Gene", rep(c("aurka", "vegf",
+     NA), length(datasetList) - 2), c("aurka", "vegf")), c(rep(mybigspace,
+     21)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, tt$cindex)
> r.lower <- c(NA, tt$lower)
> r.upper <- c(NA, tt$upper)
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0.5, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(0.4, 0.8, 0.05), xlab = paste("AURKA and
VEGF Concordance Index",
+         myspace, sep = ""), col = meta.colors(line = c(rep(c(NA,
+         "darkblue", "seagreen"), 7)), zero = "firebrick", box = c(rep(c(NA,
+         " royalblue", "forestgreen"), 7))), box.size = bs, clip = c(0.3,
+         1), is.summary = (c(rep(FALSE, 19), TRUE, TRUE)))
```



We display the D indices for both genes in each dataset and the combined estimation over all datasets in the same way.

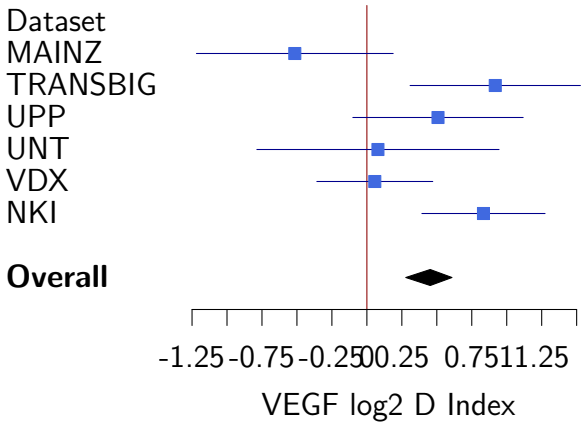
```
> tt <- rbind(dindexall.mainz.small[3, ], dindexall.transbig.small[3,
+     ], dindexall.upp.small[3, ], dindexall.unt.small[3, ], dindexall.vdx.small[3,
+     ], dindexall.nki.small[3, ], NA, as.numeric(cdindex[3, ]))
> rownames(tt) <- datasetList
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", datasetList), c(rep(mybigspace,
+     length(datasetList) + 1)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, log2(tt$dindex))
```

```
> r.lower <- c(NA, log2(tt$lower))
> r.upper <- c(NA, log2(tt$upper))
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(-0.5, 2, 0.5), xlab = paste("AURKA log2
D Index",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(-0.25, 2), is.summary = (c(rep(FALSE, 8), TRUE)))
```




```
> tt <- rbind(dindexall.mainz.small[5, ], dindexall.transbig.small[5,
+     ], dindexall.upp.small[5, ], dindexall.unt.small[5, ], dindexall.vdx.small[5,
```

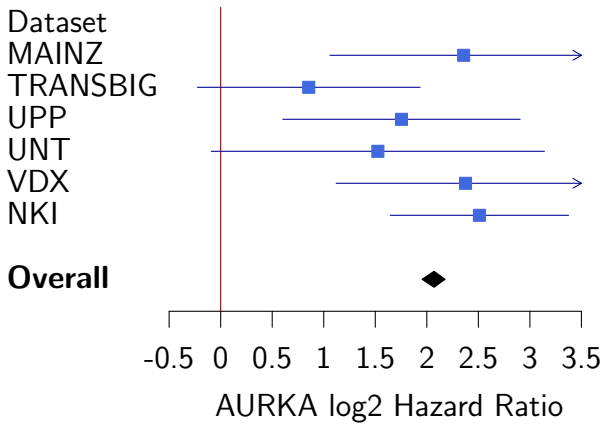
```
+     ], dindexall.nki.small[5, ], NA, as.numeric(cdindex[5, ]))
> rownames(tt) <- datasetList
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", datasetList), c(rep(mybigspace,
+     length(datasetList) + 1)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, log2(tt$dindex))
> r.lower <- c(NA, log2(tt$lower))
> r.upper <- c(NA, log2(tt$upper))
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(-1.25, 1.5, 0.25), xlab = paste("VEGF log2
D Index",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(-1.5, 1.75), is.summary = (c(rep(FALSE, 8), TRUE)))
```



And at last the hazard ratio for the gene AURKA in each dataset and the combined estimation over all datasets.

```
> tt <- rbind(hratio.mainz.small[3, ], hratio.transbig.small[3,
+     ], hratio.upp.small[3, ], hratio.unt.small[3, ], hratio.vdx.small[3,
+     ], hratio.nki.small[3, ], NA, as.numeric(chratio[3, ]))
> rownames(tt) <- datasetList
> tt <- as.data.frame(tt)
> labeltext <- cbind(c("Dataset", datasetList), c(rep(myspace,
+     length(datasetList) + 1)))
> bs <- rep(0.5, nrow(labeltext))
> r.mean <- c(NA, log2(tt$hratio))
```

```
> r.lower <- c(NA, log2(tt$lower))
> r.upper <- c(NA, log2(tt$upper))
> forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+     upper = r.upper, zero = 0, align = c("l"), graphwidth = unit(2,
+         "inches"), x.ticks = seq(-0.5, 3.5, 0.5), xlab = paste("AURKA log2
Hazard Ratio",
+         myspace, sep = ""), col = meta.colors(box = "royalblue",
+         line = "darkblue", zero = "darkred"), box.size = bs,
+     clip = c(-0.5, 3.5), is.summary = (c(rep(FALSE, 8), TRUE)))
```



The following small loop shows an easy way for creating several forestplots showing the concordance indices for a single gene for all datasets and the combined estimation over all datasets. The same can be done for the D indices and hazard ratios. Since it is not yet

possible to combine several forestplots in one figure (e.g. with `par(mfrow=c(2,2))`), we don't display the results of the following loop.

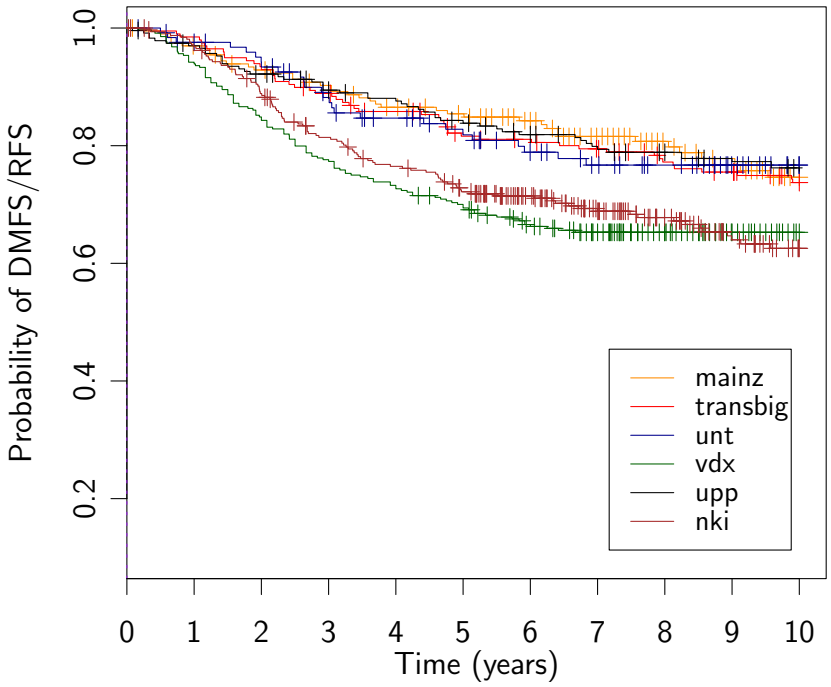
```
> for (i in 1:length(gsList)) {
+     tt <- rbind(cindexall.mainz.small[i, ], cindexall.transbig.small[i,
+         ], cindexall.upp.small[i, ], cindexall.unt.small[i, ],
+         cindexall.vdx.small[i, ], cindexall.nki.small[i, ], NA,
+         as.numeric(ccindex[i, ]))
+     rownames(tt) <- datasetList
+     tt <- as.data.frame(tt)
+     labeltext <- cbind(c("Dataset", datasetList), c(rep(myspace,
+         length(datasetList) + 1)))
+     bs <- rep(0.5, nrow(labeltext))
+     r.mean <- c(NA, tt$cindex)
+     r.lower <- c(NA, tt$lower)
+     r.upper <- c(NA, tt$upper)
+     x.ticks.lower <- (floor((min(r.mean, na.rm = TRUE) - 0.1) *
+         10)/10)
+     x.ticks.upper <- (floor((max(r.mean, na.rm = TRUE) + 0.2) *
+         10)/10)
+     forestplot.surv(labeltext = labeltext, mean = r.mean, lower = r.lower,
+         upper = r.upper, zero = 0.5, align = c("l"), graphwidth = unit(2,
+             "inches"), x.ticks = seq(x.ticks.lower, x.ticks.upper,
+             0.05), xlab = paste(gsList[i], " Concordance Index",
+             myspace, sep = ""), col = meta.colors(box = "royalblue",
+             line = "darkblue", zero = "darkred"), box.size = bs,
+         clip = c(0.3, 0.8), is.summary = (c(rep(FALSE, 8), TRUE)))
+ }
```

2.5 Kaplan Meier survival curves

To display a Kaplan Meier curve [8] for all datasets you can use:

```
> surv.data <- censor.time(surv.time = c(pData(mainz7g)[, "t.dmfs"],
+     pData(transbig7g)[, "t.dmfs"], pData(unt7g)[, "t.dmfs"],
+     pData(vdx7g)[, "t.dmfs"], pData(upp7g)[, "t.rfs"], pData(nki7g)[,
+         "t.dmfs"])/365, surv.event = c(pData(mainz7g)[, "e.dmfs"],
+     pData(transbig7g)[, "e.dmfs"], pData(unt7g)[, "e.dmfs"],
+     pData(vdx7g)[, "e.dmfs"], pData(upp7g)[, "e.rfs"], pData(nki7g)[,
+         "e.dmfs"]), time.cens = tc/365)
> gg <- factor(c(rep("mainz", nrow(pData(mainz7g))), rep("transbig",
```

```
+     nrow(pData(transbig7g))), rep("unt", nrow(pData(unt7g))),
+     rep("vdx", nrow(pData(vdx7g))), rep("upp", nrow(pData(upp7g))),
+     rep("nki", nrow(pData(nki7g)))), levels = c("mainz", "transbig",
+     "unt", "vdx", "upp", "nki"))
> dd <- data.frame(time = surv.data[[1]], event = surv.data[[2]],
+     group = gg)
> km.coxph.plot(formula.s = formula(Surv(time, event) ~ group),
+     data.s = dd, sub.s = "all", x.label = "Time (years)", y.label = "Prob-
ability of DMFS/RFS",
+     main.title = "", sub.title = "", leg.pos = "bottomright",
+     leg.inset = 0.05, o.text = FALSE, v.line = FALSE, h.line = FALSE,
+     .lty = rep(1, length(levels(gg))), show.n.risk = TRUE, n.risk.step =
1,
+     n.risk.cex = 0.85, .col = c("darkorange", "red", "darkblue",
+         "darkgreen", "black", "brown"), leg.text = paste(levels(gg),
+         myspace, sep = ""), verbose = FALSE, ylim = c(0.1, 1))
```



No. At Risk

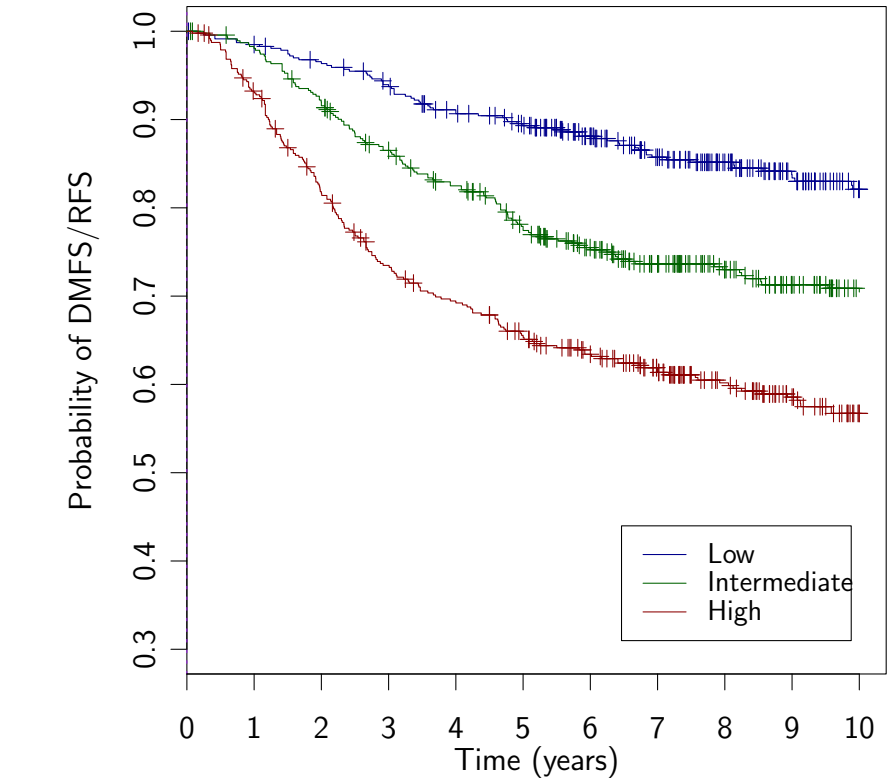
mainz	200	192	180	172	160	154	130	109	87	77	60
transbig	198	196	185	176	167	155	152	147	137	133	123
unt	125	119	112	102	94	87	78	69	62	49	38
vdx	344	324	292	267	252	231	207	184	136	89	53
upp	234	225	209	195	188	177	167	159	152	146	134
nki	319	306	279	250	233	215	180	148	119	95	77

If you want do display the survival curve for a single gene using the data of all six datasets, we have to concatenate the survival and expression data of all datasets (see `surv.time.all`, `surv.event.all` and `aurka.exprs` below). After that we split the patients in each dataset into three parts acording to their gene expression. We use the function `quantile()` for that. In the end we have three groups, representing the low gene expression group (lowest 33% of the gene expression), intermediate gene expression group (gene expression between 33% and 66%) and high gene expression group (over 66%).

```
> aurkaGs <- "AURKA"
> aurkaGid <- 6790
> aurkaPaf <- "208079_s_at"
> aurkaPagi <- "NM_003600"
```

```
> surv.time.all <- c(pData(mainz7g)[, "t.dmfs"], pData(transbig7g)[,
+     "t.dmfs"], pData(unt7g)[, "t.dmfs"], pData(upp7g)[, "t.rfs"],
+     pData(vdx7g)[, "t.dmfs"], pData(nki7g)[, "t.dmfs"])
> surv.event.all <- c(pData(mainz7g)[, "e.dmfs"], pData(transbig7g)[,
+     "e.dmfs"], pData(unt7g)[, "e.dmfs"], pData(upp7g)[, "e.rfs"],
+     pData(vdx7g)[, "e.dmfs"], pData(nki7g)[, "e.dmfs"])
> aurka.exprs <- c(exprs(mainz7g)[aurkaPaf, ], exprs(transbig7g)[aurkaPaf,
+     ], exprs(unt7g)[aurkaPaf, ], exprs(upp7g)[aurkaPaf, ], exprs(vdx7g)[aurkaPaf,
+     ], exprs(nki7g)[aurkaPagi, ])
> aurka.exprs.length <- c(length(exprs(mainz7g)[aurkaPaf, ]), length(exprs(transbig7g)[
+     ]), length(exprs(unt7g)[aurkaPaf, ]), length(exprs(upp7g)[aurkaPaf,
+     ]), length(exprs(vdx7g)[aurkaPaf, ]), length(exprs(nki7g)[aurkaPagi,
+     ]))
> pos <- 0
> mygroup <- NULL
> for (i in aurka.exprs.length) {
+     qq <- aurka.exprs[(pos + 1):(pos + i)]
+     myq <- quantile(qq, probs = c(0.33, 0.66), na.rm = TRUE)
+     qq[aurka.exprs[(pos + 1):(pos + i)] < myq[1]] <- 1
+     qq[aurka.exprs[(pos + 1):(pos + i)] >= myq[1] & aurka.exprs[(pos +
+         1):(pos + i)] < myq[2]] <- 2
+     qq[aurka.exprs[(pos + 1):(pos + i)] > myq[2]] <- 3
+     qq <- factor(x = qq, levels = 1:3)
+     mygroup <- c(mygroup, qq)
+     pos <- pos + i
+ }
> surv.data <- censor.time(surv.time = surv.time.all/365, surv.event = surv.event.all,
+     time.cens = tc/365)
> dd <- data.frame(time = surv.data[[1]], event = surv.data[[2]],
+     gg = mygroup)
> gg <- factor(c(rep("mainz", nrow(pData(mainz7g))), rep("transbig",
+     nrow(pData(transbig7g))), rep("unt", nrow(pData(unt7g))),
+     rep("upp", nrow(pData(upp7g))), rep("vdx", nrow(pData(vdx7g))),
+     rep("nki", nrow(pData(nki7g)))), levels = c("mainz", "transbig",
+     "unt", "upp", "vdx", "nki"))
> km.coxph.plot(formula.s = formula(Surv(time, event) ~ gg), data.s = dd,
+     sub.s = "all", x.label = "Time (years)", y.label = "Probability of DMFS/RFS",
+     main.title = "", sub.title = "", leg.text = c("Low    ", "Intermedi-
ate    ",
+         "High    "), leg.pos = "bottomright", leg.inset = 0.05,
+     o.text = FALSE, v.line = FALSE, h.line = FALSE, .col = c("darkblue",
+         "darkgreen", "darkred"), .lty = 1, show.n.risk = TRUE,
```

```
+     n.risk.step = 1, n.risk.cex = 0.85, verbose = FALSE, ylim = c(0.3,
+         1))
```



No. At Risk

Low	467	459	446	431	411	395	351	314	268	220	178
Intermediate	468	456	424	391	369	333	297	263	222	193	166
High	478	437	378	331	307	282	257	230	195	167	137

2.6 Meta analysis of estimation values

The *SurvComp* package integrates functions for meta-analysis of risk-prediction models, e.g. for the concordance index or the D index. The following example shows the `cindex.comp.meta()` function [9]. Table 6 shows the p-values representing the difference between the cindices of two genes using the cindices of all six datasets. For example, the cindex of the gene AURKA is with a p-value of 0.00001 significantly different from the cindex of the gene VEGF using the six datasets.

```
> cindexMetaMainz <- t(apply(X = exprs(mainz7g), MARGIN = 1, function(x,
+     y, z) {
+     tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+         method = "noether", na.rm = TRUE)
+     return(tt)
+ }, y = pData(mainz7g)[, "t.dmfs"], z = pData(mainz7g)[, "e.dmfs"]))
> cindexMetaTransbig <- t(apply(X = exprs(transbig7g), MARGIN = 1,
+     function(x, y, z) {
+         tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+             method = "noether", na.rm = TRUE)
+         return(tt)
+     }, y = pData(transbig7g)[, "t.dmfs"], z = pData(transbig7g)[,
+         "e.dmfs"]))
> cindexMetaUpp <- t(apply(X = exprs(upp7g), MARGIN = 1, function(x,
+     y, z) {
+     tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+         method = "noether", na.rm = TRUE)
+     return(tt)
+ }, y = pData(upp7g)[, "t.rfs"], z = pData(upp7g)[, "e.rfs"]))
> cindexMetaUnt <- t(apply(X = exprs(unt7g), MARGIN = 1, function(x,
+     y, z) {
+     tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+         method = "noether", na.rm = TRUE)
+     return(tt)
+ }, y = pData(unt7g)[, "t.dmfs"], z = pData(unt7g)[, "e.dmfs"]))
> cindexMetaVdx <- t(apply(X = exprs(vdx7g), MARGIN = 1, function(x,
+     y, z) {
+     tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+         method = "noether", na.rm = TRUE)
+     return(tt)
+ }, y = pData(vdx7g)[, "t.dmfs"], z = pData(vdx7g)[, "e.dmfs"]))
> ccNki <- complete.cases(exprs(nki7g)[1, ], exprs(nki7g)[2, ],
+     exprs(nki7g)[3, ], exprs(nki7g)[4, ], exprs(nki7g)[5, ],
+     exprs(nki7g)[6, ], exprs(nki7g)[7, ], pData(nki7g)[, "e.dmfs"],
+     pData(nki7g)[, "e.dmfs"])
> cindexMetaNki <- t(apply(X = exprs(nki7g)[, ccNki], MARGIN = 1,
+     function(x, y, z) {
+         tt <- concordance.index(x = x, surv.time = y, surv.event = z,
+             method = "noether", na.rm = TRUE)
+         return(tt)
+     }, y = pData(nki7g)[ccNki, "t.dmfs"], z = pData(nki7g)[ccNki,
+         "e.dmfs"]))
```

	esr1	erbb2	aurka	plau	vegfa	stat1	casp3
esr1	1.00000	0.96698	1.00000	0.99489	1.00000	0.99967	0.99860
erbb2	0.03302	1.00000	1.00000	0.79944	0.99855	0.94848	0.87718
aurka	0.00000	0.00000	1.00000	0.00000	0.00001	0.00000	0.00000
plau	0.00511	0.20056	1.00000	1.00000	0.98988	0.80287	0.62407
vegfa	0.00000	0.00145	0.99999	0.01012	1.00000	0.07944	0.02743
stat1	0.00033	0.05152	1.00000	0.19713	0.92056	1.00000	0.28393
casp3	0.00140	0.12282	1.00000	0.37593	0.97257	0.71607	1.00000

Table 6: `cindex.comp.meta()` results showing the significance of the difference between concordance indices.

```
> ccmData <- tt <- rr <- NULL
> for (i in 1:7) {
+     tt <- NULL
+     listOne <- list(mainz = cindexMetaMainz[[i]], transbig = cindexMeta-
Transbig[[i]],
+         upp = cindexMetaUpp[[i]], unt = cindexMetaUnt[[i]], vdx = cindexMetaVdx[[i]],
+         nki = cindexMetaNki[[i]])
+     for (j in 1:7) {
+         listTwo <- list(mainz = cindexMetaMainz[[j]], transbig = cindexMeta-
Transbig[[j]],
+             upp = cindexMetaUpp[[j]], unt = cindexMetaUnt[[j]],
+             vdx = cindexMetaVdx[[j]], nki = cindexMetaNki[[j]])
+         rr <- cindex.comp.meta(list.cindex1 = listOne, list.cindex2 = listTwo)
+         tt <- cbind(tt, rr$p.value)
+     }
+     ccmData <- rbind(ccmData, tt)
+ }
> ccmData <- as.data.frame(ccmData)
> colnames(ccmData) <- gsList
> rownames(ccmData) <- gsList
```

3 Session Info

- R version 2.12.1 (2010-12-16), `i686-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.utf8`, `LC_NUMERIC=C`, `LC_TIME=en_US.utf8`, `LC_COLLATE=en_US.utf8`, `LC_MONETARY=C`, `LC_MESSAGES=en_US.utf8`, `LC_PAPER=en_US.utf8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.utf8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, tools, utils
- Other packages: Biobase 2.10.0, bootstrap 1.0-22, cacheSweave 0.4-5, class 7.3-2, codetools 0.2-8, filehash 2.1-1, formatR 0.2-0, getopt 1.15, highlight 0.2-5, ipred 0.8-11, KernSmooth 2.23-4, MASS 7.3-7, mlbench 2.1-0, nnet 7.3-1, parser 0.0-13, pgfSweave 1.1.3, prodlim 1.1.3, Rcpp 0.9.2, rmeta 2.16, rpart 3.1-48, stashR 0.3-3, SuppDists 1.1-8, survcomp 1.1.8, survival 2.36-5, survivalROC 1.0.0, tikzDevice 0.5.3, xtable 1.5-6
- Loaded via a namespace (and not attached): digest 0.4.2

4 Functions within SurvComp

For references to the following functions, please see [2]-[21].

Function	Description
D.index	Function to compute the D index
breastCancerData	Sample data containing six datasets for gene expression, annotations and clinical data
censor.time	Function to artificially censor survival data
cindex.comp	Function to compare two concordance indices
cindex.comp.meta	Function to compare two concordance indices
combine.est	Function to combine estimates
combine.test	Function to combine probabilities
concordance.index	Function to compute the concordance index for survival or binary class prediction
cvpl	Function to compute the CVPL
dindex.comp	Function to compare two D indices
dindex.comp.meta	Function to compare two D indices
fisherz	Function to compute Fisher z transformation
forestplot.surv	Forest plots enables to display performance estimates of survival models
getsurv2	Function to retrieve the survival probabilities at a specific point in time
hazard.ratio	Function to estimate the hazard ratio through Cox regression
hr.comp	Function to statistically compare two hazard ratios
hr.comp.meta	Function to compare two concordance indices
hr.comp2	Function to statistically compare two hazard ratios (alternative interface)
iauc.comp	Function to compare two IAUCs through time-dependent ROC curves
ibsc.comp	Function to compare two IBSCs
km.coxph.plot	Function to plot several Kaplan-Meier survival curves
logpl	Function to compute the log partial likelihood of a Cox model
no.at.risk	Function to compute the number of individuals at risk
sbrier.score2proba	Function to compute the BSCs from a risk score, for all the times of event occurrence
score2proba	Function to compute the survival probabilities from a risk score
survcomp-package	Performance Assessment and Comparison for Survival Analysis
td.sens.spec	Function to compute sensitivity and specificity for a binary classification of survival data
tdrocc	Function to compute time-dependent ROC curves
test.hetero.est	Function to test the heterogeneity of set of probabilities
test.hetero.test	Function to test the heterogeneity of set of probabilities

References

[1] Desmedt, C., Haibe-Kains, B., Wirapati, P., Buyse, M., Larsimont, D., Bontempi, G., Delorenzi, M., Piccart, M. and Sotiriou, C.: Biological Processes Associated with Breast Cancer Clinical Outcome Depend on the Molecular Subtypes. *Clinical Cancer Research*, **16**, 5158-5165. 2008

[2] Harrel Jr, F. E. and Lee, K. L. and Mark, D. B.: Tutorial in biostatistics: multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, **15**, 361-387. 1996.

[3] Pencina, M. J. and D'Agostino, R. B.: Overall C as a measure of discrimination in survival analysis: model specific population value and confidence interval estimation. *Statistics in Medicine*, **23**, 2109-2123. 2004.

[4] Royston, P. and Sauerbrei, W.: A new measure of prognostic separation in survival data. *Statistics in Medicine*, **23**, 723-748. 2004.

[5] Cox, D. R.: Regression Models and Life Tables. *Journal of the Royal Statistical Society Series B*, **34**, 187-220. 1972.

[6] Cochrane, W. G.: The combination of estimates from different experiments. *Biometrics*, **10**, 101-129. 1954.

[7] Lewis, Steff, and Mike Clarke: Forest plots: trying to see the wood and the trees. *BMJ*, **322**, 1479-1480. 2001

[8] Kaplan, E. L., and Paul Meier: Nonparametric Estimation from Incomplete Observations. *Journal of the American Statistical Association*, **53**, 457-481. 1958

[9] Haibe-Kains, B. and Desmedt, C. and Sotiriou, C. and Bontempi, G.: A comparative study of survival models for breast cancer prognostication based on microarray data: does a single gene beat them all? *Bioinformatics*, **24**:19, 2200-2208. 2008.

[10] Whitlock, M. C.: Combining probability from independent tests: the weighted Z-method is superior to Fisher's approach. *J. Evol. Biol.*, **18**, 1368-1373. 2005.

[11] Heagerty, P. J. and Lumley, T. L. and Pepe, M. S.: Time-Dependent ROC Curves for Censored Survival Data and a Diagnostic Marker. *Biometrics*, **56**, 337-344. 2000.

[12] Efron, B. and Tibshirani, R.: The Bootstrap Method for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, **1**, 1-35. 1986.

[13] Becker, R. A., Chambers, J. M. and Wilks, A. R.: The New S Language. *Wadsworth & Brooks/Cole*, 1988.

[14] Andersen, P. K. and Borgan, O. and Gill, R. D. and Keiding, N.: Statistical Models Based on Counting Processes *Springer*, 1993.

[15] Brier, G. W.: Verification of forecasts expressed in terms of probabilities. *Monthly Weather Review*, **78**, 1-3. 1950.

[16] Graf, E. and Schmoor, C. and Sauerbrei, W. and Schumacher, M.: Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, **18**, 2529-2545. 1999.

[17] Wilcoxon, F.: Individual comparisons by ranking methods. *Biometrics Bulletin*, **1**, 80-83. 1945.

[18] Student: The Probable Error of a Mean. *Biometrika*, **6**, 1-25. 1908.

[19] R. A. Fisher: Frequency distribution of the values of the correlation coefficient in samples of an indefinitely large population. *Biometrika*, **10**, 507-521. 1915.

[20] Verweij PJM. and van Houwelingen H.: Cross-validation in survival analysis. *Statistics in Medicine*, **12**, 2305-2314. 1993.

[21] van Houwelingen H, Bruinsma T, Hart AA, van't Veer LJ and Wessels LFA: Cross-validated Cox regression on microarray gene expression data. *tatistics in Medicine*, **25**, 3201-3216. 2006.