

genefu: a package for breast cancer gene expression analysis

Deena M.A. Gendoo^{1,2}, Natchar Ratanasirigulchai¹, Markus Schröder³, and Benjamin Haibe-Kains^{*1,2}

¹Bioinformatics and Computational Genomics Laboratory, Princess Margaret Cancer Center, University Health Network, Toronto, Ontario, Canada

²Department of Medical Biophysics, University of Toronto, Toronto, Canada

³UCD School of Biomolecular and Biomedical Science, Conway Institute, University College Dublin, Belfield, Dublin, Ireland

⁴Machine Learning Group, Université Libre de Bruxelles

July 28, 2015

Contents

1	Introduction	1
2	Loading package for case studies	1
3	Load Datasets and Packages for Case Studies	2
4	Case Study : Compare Molecular Subtype Classifications	4
5	Case Study : Comparing risk prediction models	15
6	References	26
7	Session Info	27

1 Introduction

The *genefu* package is providing relevant functions for gene expression analysis, especially in breast cancer. This package includes a number of algorithms for molecular subtype classification. The package also includes implementations of prognostic prediction algorithms, along with lists of prognostic gene signatures on which these algorithms were based.

Please refer to the manuscript URL and Lab website: <http://www.pmgenomics.ca/bhklab/software/genefu>

Please also refer to the References section below, for additional information on publications that have cited Version 1 of *genefu*.

2 Loading package for case studies

First we load the *genefu* into the workspace. The package is publicly available and can be installed from Bioconductor version 2.8 or higher in R version 2.13.0 or higher.

To install the *genefu* package:

*benjamin.haibe.kains@utoronto.ca

```
knitr::opts_chunk$set(eval=TRUE,cache=TRUE)
source("http://bioconductor.org/biocLite.R")
biocLite("genefu")
```

For computing the risk scores, estimates of the performance of the risk scores, combining the estimates and comparing the estimates we have to load the genefu and survcomp packages into the workspace. We also load all the packages we need to conduct the case studies.

```
library(genefu)
library(xtable)
library(rmeta)
library(Biobase)
library(caret)
library(e1071)
```

3 Load Datasets and Packages for Case Studies

The following case study compares risk prediction models. This includes computing risk scores, computing estimates of the performance of the risk scores, as well as combining the estimates and comparing them.

The five data sets that we use in the case study are publicly available as experimental data packages on Bioconductor.org. In particular we used:

breastCancerMAINZ: bioconductor.org/packages/release/data/experiment/html/breastCancerMAINZ.html

breastCancerUPP: bioconductor.org/packages/release/data/experiment/html/breastCancerUPP.html

breastCancerUNT: bioconductor.org/packages/release/data/experiment/html/breastCancerUNT.html

breastCancerNKI: bioconductor.org/packages/release/data/experiment/html/breastCancerNKI.html

breastCancerTRANSBIG: bioconductor.org/packages/release/data/experiment/html/breastCancerTRANSBIG.html

Please Note: We don't use the breastCancerVDX experimental package in this case study since it has been used as training data set for GENIUS. Please refer to Haibe-Kains et al, 2010. The breastCancerVDX is found at the following link:

breastCancerVDX: <http://www.bioconductor.org/packages/release/data/experiment/html/breastCancerVDX.html>

These experimental data packages can be installed from Bioconductor version 2.8 or higher in R version 2.13.0 or higher. For the experimental data packages the commands for installing the data sets are:

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("breastCancerMAINZ")
biocLite("breastCancerTRANSBIG")
biocLite("breastCancerUPP")
biocLite("breastCancerUNT")
biocLite("breastCancerNKI")
```

And to load the packages into R, please use the following commands:

```
library(breastCancerMAINZ)
library(breastCancerTRANSBIG)
library(breastCancerUPP)
library(breastCancerUNT)
library(breastCancerNKI)
```

Table 1: Detailed overview for the data sets used in the case study

Dataset	Patients [#]	ER+ [#]	HER2+ [#]	Age [years]	Grade [1/2/3]	Platform
MAINZ	200	155	23	25-90	29/136/35	HGU133A
TRANSBIG	198	123	35	24-60	30/83/83	HGU133A
UPP	251	175	46	28-93	67/128/54	HGU133AB
UNT	137	94	21	24-73	32/51/29	HGU133AB
NKI	337	212	53	26-62	79/109/149	Agilent
Overall	1123	759	178	25-73	237/507/350	Affy/Agilent

Table1 shows an overview of the data sets and the patients. From those 1123 breast cancer patients we selected only the patients that are node negative and didn't receive any treatment (except local radiotherapy), which results in 713 patients.

Since there are duplicated patients in the five data sets, we have to identify the duplicated patients and we subsequently store them in a vector.

```
data(breastCancerData)
cinfo <- colnames(pData(mainz7g))
data.all <- c("transbig7g"=transbig7g, "unt7g"=unt7g, "upp7g"=upp7g, "mainz7g"=mainz7g, "nki7g"=nki7g)

idtoremove.all <- NULL
duplres <- NULL

## No overlaps in the MainZ and NKI datasets.

## Focus on UNT vs UPP vs TRANSBIG
demo.all <- rbind(pData(transbig7g), pData(unt7g), pData(upp7g))
dn2 <- c("TRANSBIG", "UNT", "UPP")

## Karolinska
## Search for the VDXKIU, KIU, UPPU series
ds2 <- c("VDXKIU", "KIU", "UPPU")
demot <- demo.all[complete.cases(demo.all[, c("series")]) & is.element(demo.all[, "series"], ds2), ]

# Find the duplicated patients in that series
duplid <- sort(unique(demot[duplicated(demot[, "id"]), "id"]))
duplrest <- NULL
for(i in 1:length(duplid)) {
  tt <- NULL
  for(k in 1:length(dn2)) {
    myx <- sort(row.names(demot)[complete.cases(demot[, c("id", "dataset")]) &
      demot[, "id"] == duplid[i] & demot[, "dataset"] == dn2[k]])
    if(length(myx) > 0) { tt <- c(tt, myx) }
  }
  duplrest <- c(duplrest, list(tt))
}
names(duplrest) <- duplid
duplres <- c(duplres, duplrest)

## Oxford
## Search for the VVDXOXFU, OXFU series
ds2 <- c("VVDXOXFU", "OXFU")
demot <- demo.all[complete.cases(demo.all[, c("series")]) & is.element(demo.all[, "series"], ds2), ]
```

```

# Find the duplicated patients in that series
duplid <- sort(unique(demot[duplicated(demot[, "id"]), "id"]))
duplrest <- NULL
for(i in 1:length(duplid)) {
  tt <- NULL
  for(k in 1:length(dn2)) {
    myx <- sort(row.names(demot)[complete.cases(demot[, c("id", "dataset")]) &
                                                         demot[, "id"] == duplid[i] & demot[, "dataset"] == dn2[k]])
    if(length(myx) > 0) { tt <- c(tt, myx) }
  }
  duplrest <- c(duplrest, list(tt))
}
names(duplrest) <- duplid
duplres <- c(duplres, duplrest)

## Full set duplicated patients
duPL <- sort(unlist(lapply(duplres, function(x) { return(x[-1]) } )))

```

4 Case Study : Compare Molecular Subtype Classifications

For this case study, we first created a merged expression set that combines all the samples across the 5 datasets. To do this we first select for the most representative probes across the esets and get the corresponding gene ids. We then perform quantile normalization on the gene expression datasets to merge them.

```

library(WGCNA)

## Loading required package: dynamicTreeCut
## Loading required package: fastcluster
##
## Attaching package: 'fastcluster'
##
## The following object is masked from 'package:stats':
##
##   hclust
##
## Loading required package: DBI

## =====
## *
## * Package WGCNA 1.47 loaded.
## *
## * Important note: It appears that your system supports multi-threading,
## * but it is not enabled within WGCNA in R.
## * To allow multi-threading within WGCNA with all available cores, use
## *
## *     allowWGCNAThreads()
## *
## * within R. Use disableWGCNAThreads() to disable threading if necessary.
## * Alternatively, set the following environment variable on your system:
## *
## *     ALLOW_WGCNA_THREADS=<number_of_processors>
## *

```

```

## *      for example
## *
## *      ALLOW_WGCNA_THREADS=4
## *
## *      To set the environment variable in linux bash shell, type
## *
## *      export ALLOW_WGCNA_THREADS=4
## *
## *      before running R. Other operating systems or shells will
## *      have a similar command to achieve the same aim.
## *
## =====
##
## Attaching package: 'WGCNA'
##
## The following object is masked from 'package:stats':
##
## cor

##-----
## Call the datasetMerging function
##-----

datasetMerging<-function (esets, nthread=1) {

  ## all unique Entrez gene ids
  ugid <- lapply(esets, function(x) {
    return(Biobase::fData(x)[, is.element(colnames(fData(x)),
                                           c("probe", "EntrezGene.ID"))]) })
  ugid$transbig$EntrezGene.ID<-as.character(ugid$transbig$EntrezGene.ID)
  ugid$unt$EntrezGene.ID<-as.character(ugid$unt$EntrezGene.ID)
  ugid$supp$EntrezGene.ID<-as.character(ugid$supp$EntrezGene.ID)
  ugid$mainz$EntrezGene.ID<-as.character(ugid$mainz$EntrezGene.ID)
  ugid$nki$EntrezGene.ID<-as.character(ugid$nki$EntrezGene.ID)

  ugid <- do.call(rbind, ugid)
  ugid <- ugid[!is.na(ugid[, "EntrezGene.ID"]) &
              !duplicated(ugid[, "EntrezGene.ID"]), , drop=FALSE]
  rownames(ugid) <- gsub(sprintf("(%s).", paste(names(esets), collapse="|")), "", rownames(ugid))
  feature.merged <- ugid

  ## expression data
  exprs.merged <- lapply(esets, function (x, y)
  {
    ee <- Biobase::exprs(x)[is.element(rownames(exprs(x)), rownames(feature.merged)),]
    print(dim(ee))
    eem <- matrix(NA, nrow=length(y), ncol=ncol(ee), dimnames=list(y, colnames(ee)))
    print(dim(eem))
    print(length(intersect(rownames(ee), rownames(eem))))
    eem[rownames(ee), colnames(ee)] <- ee
    return (eem)
  }, y=rownames(feature.merged))

  exprs.merged <- do.call(cbind, exprs.merged)

```

```

## clinical info
ucid <- lapply(esets, function(x) { return(colnames(Biobase::pData(x))) })
ucid <- table(unlist(ucid))
ucid <- names(ucid)[ucid == length(esets)]
clinicinfo.merged <- lapply(esets, function (x , y) {
  ee <- Biobase::pData(x)[ , y, drop=FALSE]
}, y=ucid)
clinicinfo.merged <- do.call(rbind, clinicinfo.merged)
rownames(clinicinfo.merged) <- colnames(exprs.merged)

## create a merged expressionSet object
eset.merged <- ExpressionSet(assayData=exprs.merged,
                             phenoData=AnnotatedDataFrame(data=clinicinfo.merged),
                             featureData=AnnotatedDataFrame(data=feature.merged))
experimentData(eset.merged)@preprocessing <- list("normalization"="mixed",
                                                  package="unspecified", version="0")

annotation(eset.merged) <- "mixed"
## standardization

## robust scaling followed by quantile normalization
ee <- exprs(eset.merged)

# ee <- apply(ee, 2, genefu::rescale)
splitix <- parallel::splitIndices(nx=ncol(ee), ncl=nthread)
mcres <- parallel::mclapply(splitix, function(x, data)
{
  res <- apply(data[ , x, drop=FALSE], 2, function (dx) {
    return ((rescale(dx, q=0.05, na.rm=TRUE) - 0.5) * 2)
  })
  return (res)
}, data=ee, mc.cores=nthread)
ee <- do.call(cbind, mcres)

## quantile normalization
ee <- limma::normalizeBetweenArrays(object=ee, method="quantile")
exprs(eset.merged) <- ee

return (eset.merged)
}

##-----
## Probe-gene mapping
##-----

nthread <- 1
availcore <- parallel::detectCores()
if (nthread > availcore) { nthread <- availcore }
options("mc.cores"=nthread)

strEsets <- c("transbig", "unt", "upp", "mainz", "nki")
esets <- list()

for(i in 1:length(strEsets))

```

```

{
  ## load dataset
  esets[i] <- get(data(list=strEsets[i]))
}
names(esets) <- strEsets
newEsets <- list()

for (counter in esets)
{
  fData(counter)$EntrezGene.ID[fData(counter)$EntrezGene.ID == ""
                                |fData(counter)$EntrezGene.ID == " " ] <- NA
  ## take out NA from fData, select those rows from expr
  counter.fdata <- fData(counter)[!is.na(fData(counter)$EntrezGene.ID),]
  counter.expr <- exprs(counter)[rownames(counter.fdata),]

  x <- cbind(counter.fdata$EntrezGene.ID, counter.expr)
  collapsed.wgcna.mrv <- WGCNA::collapseRows(x[,-1], rowID=rownames(x),
                                             rowGroup=x[,1], method="maxRowVariance")

  best_probe <- collapsed.wgcna.mrv[[3]]
  Biobase::exprs(counter) <- counter.expr[best_probe, ]
  Biobase::fData(counter) <- counter.fdata[best_probe,]
  rownames(Biobase::exprs(counter)) <- Biobase::fData(counter)$EntrezGene.ID
  rownames(Biobase::fData(counter)) <- Biobase::fData(counter)$EntrezGene.ID
  newEsets <- c(newEsets, counter)
}
names(newEsets) <- strEsets

##-----
## Merge the datasets to eset.merged & remove duplicates
##-----
eset.merged <- datasetMerging(newEsets)

## [1] 13091    198
## [1] 18742    198
## [1] 13091
## [1] 18528    137
## [1] 18742    137
## [1] 18528
## [1] 18528    251
## [1] 18742    251
## [1] 18528
## [1] 13091    200
## [1] 18742    200
## [1] 13091
## [1] 13116    337
## [1] 18742    337
## [1] 13116

pData(eset.merged) <- cbind(Biobase::pData(eset.merged), "duplicates"=NA)
dups<-duPL
names(dups)<-dups

exprs(eset.merged)<-exprs(eset.merged[ , -which(colnames(eset.merged) %in% c(dups))])
pData(eset.merged)<-pData(eset.merged)[-which(rownames(pData(eset.merged)) %in% c(dups)),]

```

We now perform molecular subtyping on the merged dataset. Here, we perform subtyping using the PAM50 as well as the SCMGene subtyping algorithms.

```
#Extract expression set, pData, fData for each dataset
ddata <- t(exprs(eset.merged))
ddemo <- phenoData(eset.merged)@data
dannot <- featureData(eset.merged)@data

#MOLECULAR SUBTYPING
#Perform subtyping using scmod2.robust
# scmod2.robust: List of parameters defining the subtype clustering model
# (as defined by Wirapati et al)
# SubtypePredictions<-subtype.cluster.predict(sbt.model=scmod2.robust,
#                                             data=ddata,annot=dannot,do.mapping=TRUE,verbose=TRUE)

SubtypePredictions<-molecular.subtyping(sbt.model = "scmod2",data = ddata,
                                       annot = dannot,do.mapping = TRUE)

#Get sample counts pertaining to each subtype
table(SubtypePredictions$subtype)

##
##          ER-/HER2- ER+/HER2- High Prolif ER+/HER2- Low Prolif
##          89          626          154
##          HER2+
##          150

#Select samples pertaining to Basal Subtype
Basals<-names(which(SubtypePredictions$subtype == "ER-/HER2-"))
#Select samples pertaining to HER2 Subtype
HER2s<-names(which(SubtypePredictions$subtype == "HER2+"))
#Select samples pertaining to Luminal Subtypes
LuminalB<-names(which(SubtypePredictions$subtype == "ER+/HER2- High Prolif"))
LuminalA<-names(which(SubtypePredictions$subtype == "ER+/HER2- Low Prolif"))
#ASSIGN SUBTYPES TO EVERY SAMPLE, ADD TO THE EXISTING PHENODATA
ddemo$SCMGENE<-SubtypePredictions$subtype
ddemo[LuminalB,]$SCMGENE<-"LumB"
ddemo[LuminalA,]$SCMGENE<-"LumA"
ddemo[Basals,]$SCMGENE<-"Basal"
ddemo[HER2s,]$SCMGENE<-"Her2"

# Perform subtyping using PAM50
# Matrix should have samples as ROWS, genes as COLUMNS
rownames(dannot)<-dannot$probe<-dannot$EntrezGene.ID

PAM50Preds<-molecular.subtyping(sbt.model = "pam50",data=ddata,
                               annot=dannot,do.mapping=TRUE)
# PAM50Preds<-intrinsic.cluster.predict(sbt.model=pam50,data=ddata,
#                                       annot=dannot,do.mapping=TRUE,verbose=TRUE)
table(PAM50Preds$subtype)

##
## Basal Her2 LumB LumA Normal
## 146 84 320 422 47
```



```

ddemo$PAM50<-PAM50Preds$subtype
LumA<-names(PAM50Preds$subtype)[which(PAM50Preds$subtype == "LumA")]
LumB<-names(PAM50Preds$subtype)[which(PAM50Preds$subtype == "LumB")]
Normals<-names(PAM50Preds$subtype)[which(PAM50Preds$subtype == "Normal")]
ddemo[LumA,]$PAM50<-"LumA"
ddemo[LumB,]$PAM50<-"LumB"

```

We can compare the performance of both molecular subtyping methods and determine how concordant subtype predictions are across the global population. We first generate a confusion matrix of the subtype predictions.

```

# Obtain the subtype prediction counts for PAM50
table(ddemo$PAM50)

##
## Basal Her2 LumB LumA Normal
## 146 84 320 422 47

# Obtain the subtype prediction counts for SCMGENE
table(ddemo$SCMGENE)

##
## Basal Her2 LumA LumB
## 89 150 154 626

ddemo$PAM50<-as.character(ddemo$PAM50)
# We compare the samples that are predicted as pertaining to a molecular subtyp
# We ignore for now the samples that predict as 'Normal' by PAM50
confusionMatrix(ddemo[-which(rownames(ddemo) %in% Normals),]$SCMGENE,
                ddemo[-which(rownames(ddemo) %in% Normals),]$PAM50)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction Basal Her2 LumA LumB
## Basal      70 6 1 1
## Her2       8 65 11 59
## LumA       0 1 126 8
## LumB      68 12 284 252
##
## Overall Statistics
##
##          Accuracy : 0.5278
##          95% CI : (0.4958, 0.5596)
## No Information Rate : 0.4342
## P-Value [Acc > NIR] : 2.833e-09
##
##          Kappa : 0.3314
## Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##          Class: Basal Class: Her2 Class: LumA Class: LumB
## Sensitivity          0.47945      0.77381      0.2986      0.7875
## Specificity          0.99031      0.91216      0.9836      0.4417

```

## Pos Pred Value	0.89744	0.45455	0.9333	0.4091
## Neg Pred Value	0.91499	0.97708	0.6464	0.8090
## Prevalence	0.15021	0.08642	0.4342	0.3292
## Detection Rate	0.07202	0.06687	0.1296	0.2593
## Detection Prevalence	0.08025	0.14712	0.1389	0.6337
## Balanced Accuracy	0.73488	0.84299	0.6411	0.6146

From these results, the concordance of the predictions between these models is around 83 percent.

We can also compare the survival of patients for each subtype. We plot the survival curves of patients by subtype, based on each molecular classification algorithm

```
# http://www.inside-r.org/r-doc/survival/survfit.coxph
library(survival)

data.for.survival.SCMGENE <- ddemo[,c("e.os", "t.os", "SCMGENE", "age")]
data.for.survival.PAM50 <- ddemo[,c("e.os", "t.os", "PAM50", "age")]
# Remove patients with missing survival information
data.for.survival.SCMGENE <- data.for.survival.SCMGENE[complete.cases(data.for.survival.SCMGENE),]
data.for.survival.PAM50 <- data.for.survival.PAM50[complete.cases(data.for.survival.PAM50),]

days.per.month <- 30.4368
days.per.year <- 365.242

data.for.survival.PAM50$months_to_death <- data.for.survival.PAM50$t.os / days.per.month
data.for.survival.PAM50$vital_status <- data.for.survival.PAM50$e.os == "1"
surv.obj.PAM50 <- survfit(Surv(data.for.survival.PAM50$months_to_death,
                              data.for.survival.PAM50$vital_status) ~ data.for.survival.PAM50$PAM50)

data.for.survival.SCMGENE$months_to_death <- data.for.survival.SCMGENE$t.os / days.per.month
data.for.survival.SCMGENE$vital_status <- data.for.survival.SCMGENE$e.os == "1"
surv.obj.SCMGENE <- survfit(Surv(
  data.for.survival.SCMGENE$months_to_death,
  data.for.survival.SCMGENE$vital_status) ~ data.for.survival.SCMGENE$SCMGENE)

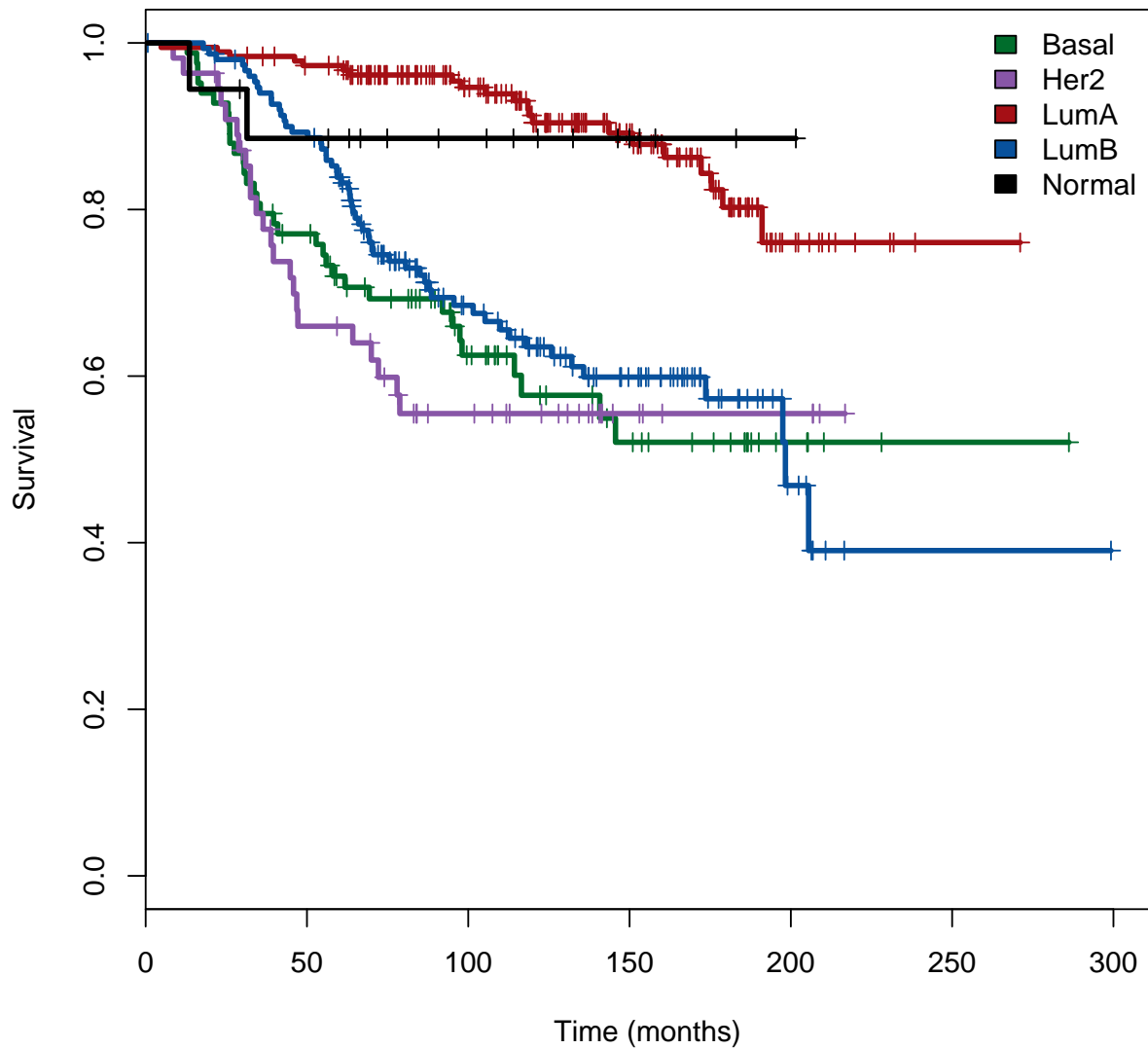
message("KAPLAN-MEIR CURVE - USING PAM50")

## KAPLAN-MEIR CURVE - USING PAM50

# survMisc::autoplot(surv.obj.PAM50, title="Survival curves PAM50", censSize=0)$plot +
#   scale_colour_manual(name="Strata", values=c("black", "green", "blue", "red"))

plot(main = "Survival Curves PAM50", surv.obj.PAM50,
     col = c("#006d2c", "#8856a7", "#a50f15", "#08519c", "#000000"), lty = 1, lwd = 3,
     xlab = "Time (months)", ylab = "Survival")
legend("topright",
     fill = c("#006d2c", "#8856a7", "#a50f15", "#08519c", "#000000"),
     legend = c("Basal", "Her2", "LumA", "LumB", "Normal"), bty = "n")
```

Survival Curves PAM50



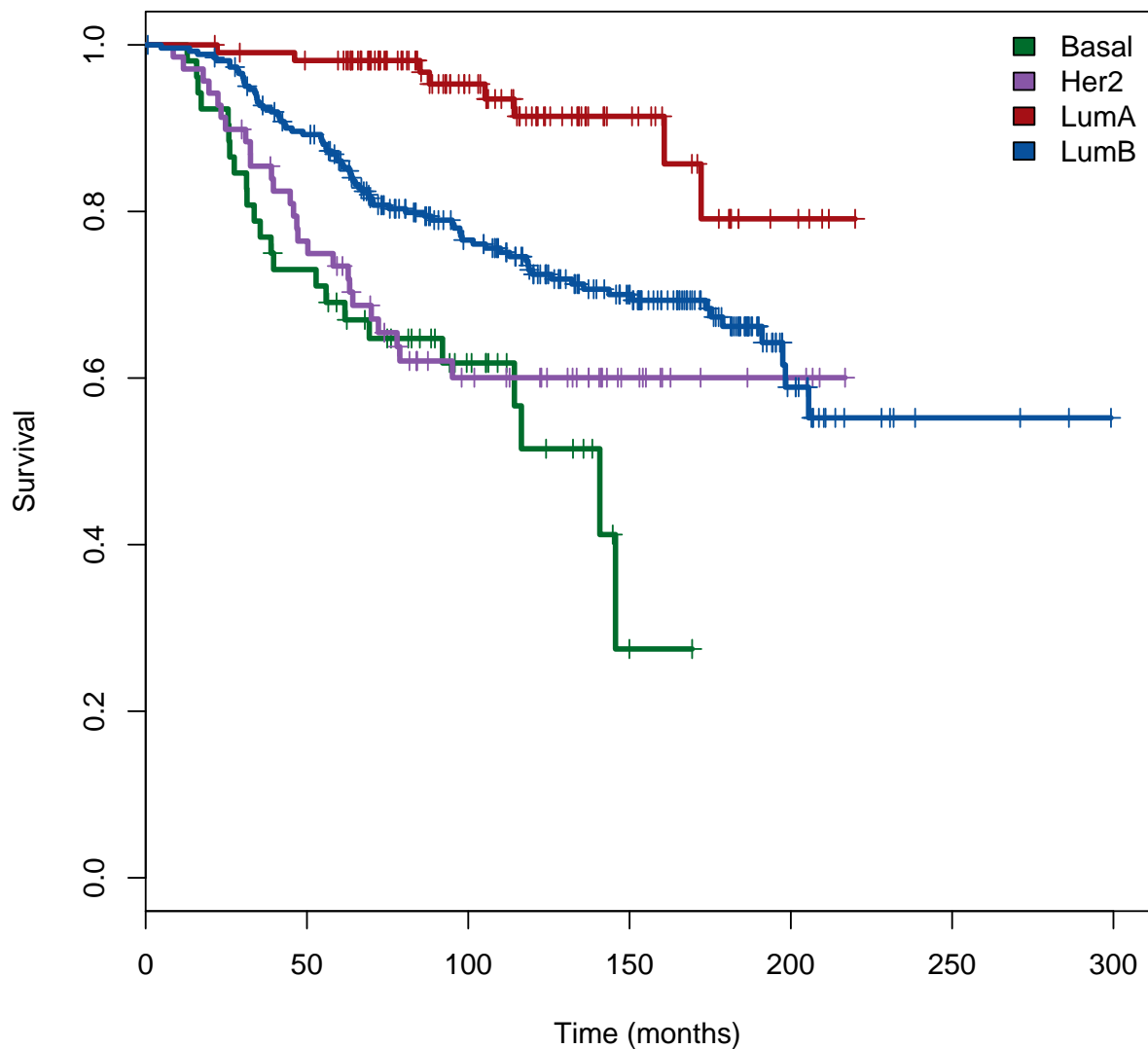
```
message("KAPLAN-MEIR CURVE - USING SCMGENE")

## KAPLAN-MEIR CURVE - USING SCMGENE

# survMisc::autoplot(surv.obj.SCMGENE, title="Survival curves SCMGENE", censSize=0)$plot +
#   scale_colour_manual(name="Strata", values=c("black", "green", "blue"))

plot(main = "Survival Curves SCMGENE", surv.obj.SCMGENE,
     col = c("#006d2c", "#8856a7", "#a50f15", "#08519c"), lty = 1, lwd = 3,
     xlab = "Time (months)", ylab = "Survival")
legend("topright",
     fill = c("#006d2c", "#8856a7", "#a50f15", "#08519c"),
     legend = c("Basal", "Her2", "LumA", "LumB"), bty = "n")
```

Survival Curves SCMGENE



```
## GENERATE A OVERLAYED PLOT OF SURVIVAL CURVES
message("Overlaid Survival Plots based on PAM50 and SCMGENE")

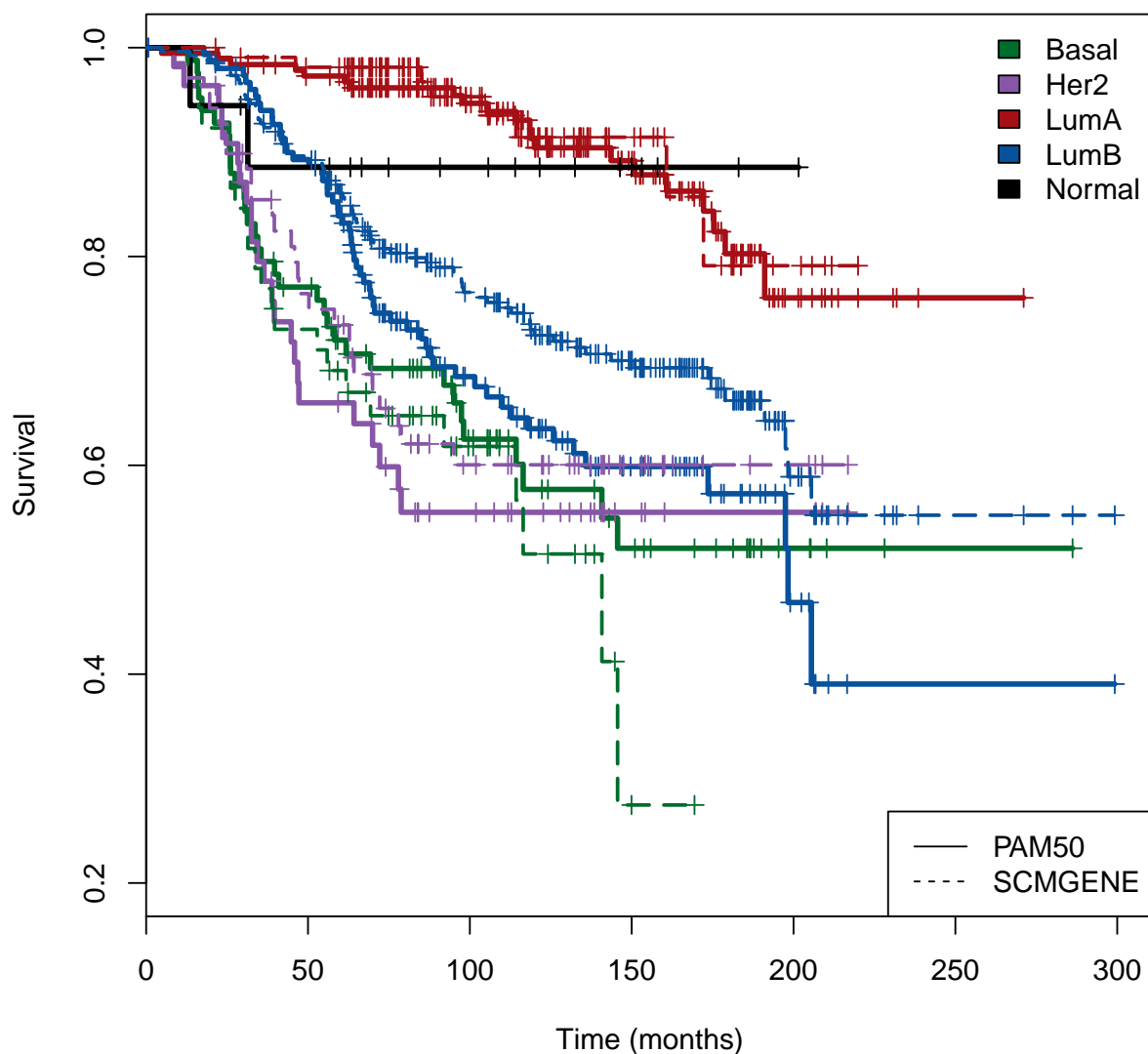
## Overlaid Survival Plots based on PAM50 and SCMGENE

## Basal      Her2      LuminalA  LuminalB  Normal
plot(surv.obj.PAM50,col =c("#006d2c", "#8856a7", "#a50f15", "#08519c", "#000000"),lty = 1,lwd = 3,
     xlab = "Time (months)",ylab = "Survival",ymin = 0.2)
legend("topright",
     fill = c("#006d2c", "#8856a7", "#a50f15", "#08519c", "#000000"),
     legend = c("Basal", "Her2", "LumA", "LumB", "Normal"),bty = "n")

par(new=TRUE)

## Basal      Her2      LuminalA  LuminalB
```

```
lines(surv.obj.SCMGENE,col =c("#006d2c", "#8856a7", "#a50f15", "#08519c"),lwd=2,lty=5)
legend("bottomright",c("PAM50", "SCMGENE"),lty=c("solid", "dashed"))
```



We can now compare which of the molecular subtyping algorithms is more prognostic. To do this we use a Cross-validated Partial Likelihood (cvpl) calculation from survcomp. This returns the mean cross-validated partial likelihood, for each algorithm, using molecular subtypes for stratification

```
set.seed(12345)
PatientMatrix<-ddata[rownames(data.for.survival.PAM50),]

PAM5_CVPL<-cvpl(x=data.for.survival.PAM50$age,
  surv.time=data.for.survival.PAM50$months_to_death,
  surv.event=data.for.survival.PAM50$vital_status,
  strata=as.integer(factor(data.for.survival.PAM50$PAM50)),
```

```

nfold=10, setseed=54321)$cvpl
SCMEGENE_CVPL<-cvpl(x=data.for.survival.SCMGENE$age,
  surv.time=data.for.survival.SCMGENE$months_to_death,
  surv.event=data.for.survival.SCMGENE$vital_status,
  strata=as.integer(factor(data.for.survival.SCMGENE$SCMGENE)),
  nfold=10, setseed=54321)$cvpl

print.data.frame(data.frame(cbind(PAM5_CVPL,SCMEGENE_CVPL)))

##          PAM5_CVPL SCMEGENE_CVPL
## logpl  1.419845      1.502288

```

5 Case Study : Comparing risk prediction models

We compute the risk scores using the following list of algorithms (and corresponding gene functions):

Subtype Clustering Model using just the AURKA gene: `scmgene.robust()`

Subtype Clustering Model using just the ESR1 gene: `scmgene.robust()`

Subtype Clustering Model using just the ERBB2 gene: `scmgene.robust()`

NPI: `npi()`

GGI: `ggi()`

GENIUS: `genius()`

EndoPredict: `endoPredict()`

OncotypeDx: `oncotypedx()`

TamR: `tamr()`

GENE70: `gene70()`

PIK3CA: `pik3cags()`

rorS: `rorS()`

```
dn <- c("transbig", "unt", "upp", "mainz", "nki")
dn.platform <- c("affy", "affy", "affy", "affy", "agilent")

res <- ddemo.all <- ddemo.coln <- NULL
for(i in 1:length(dn)) {

  ## load dataset
  dd <- get(data(list=dn[i]))

  #Extract expression set, pData, fData for each dataset
  ddata <- t(exprs(dd))
  ddemo <- phenoData(dd)@data
  dannot <- featureData(dd)@data
  ddemo.all <- c(ddemo.all, list(ddemo))
  if(is.null(ddemo.coln))
  { ddemo.coln <- colnames(ddemo) } else
  { ddemo.coln <- intersect(ddemo.coln, colnames(ddemo)) }
  rest <- NULL

  ## AURKA
  ## if affy platform consider the probe published in Desmedt et al., CCR, 2008
  if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
  modt <- scmgene.robust$mod$AURKA
  ## if agilent platform consider the probe published in Desmedt et al., CCR, 2008
  if(dn.platform[i] == "agilent") {
    domap <- FALSE
    modt[ , "probe"] <- "NM_003600"
  }
  rest <- cbind(rest, "AURKA"=sig.score(x=modt, data=ddata, annot=dannot, do.mapping=domap)$score)
```

```

## ESR1
## if affy platform consider the probe published in Desmedt et al., CCR, 2008
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
modt <- scmgene.robust$mod$ESR1
## if agilent platform consider the probe published in Desmedt et al., CCR, 2008
if(dn.platform[i] == "agilent") {
  domap <- FALSE
  modt[ , "probe"] <- "NM_000125"
}
rest <- cbind(rest, "ESR1"=sig.score(x=modt, data=ddata, annot=dannot, do.mapping=domap)$score)

## ERBB2
## if affy platform consider the probe published in Desmedt et al., CCR, 2008
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
modt <- scmgene.robust$mod$ERBB2
## if agilent platform consider the probe published in Desmedt et al., CCR, 2008
if(dn.platform[i] == "agilent") {
  domap <- FALSE
  modt[ , "probe"] <- "NM_004448"
}
rest <- cbind(rest, "ERBB2"=sig.score(x=modt, data=ddata, annot=dannot, do.mapping=domap)$score)

## NPI
ss <- ddemo[ , "size"]
gg <- ddemo[ , "grade"]
nn <- rep(NA, nrow(ddemo))
nn[complete.cases(ddemo[ , "node"]) & ddemo[ , "node"] == 0] <- 1
nn[complete.cases(ddemo[ , "node"]) & ddemo[ , "node"] == 1] <- 3
names(ss) <- names(gg) <- names(nn) <- rownames(ddemo)
rest <- cbind(rest, "NPI"=npi(size=ss, grade=gg, node=nn, na.rm=TRUE)$score)

## GGI
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "GGI"=ggi(data=ddata, annot=dannot, do.mapping=domap)$score)

## GENIUS
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "GENIUS"=genius(data=ddata, annot=dannot, do.mapping=domap)$score)

## ENDOPREDICT
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "EndoPredict"=endoPredict(data=ddata, annot=dannot, do.mapping=domap)$score)

# OncotypeDx
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "OncotypeDx"=oncotypedx(data=ddata, annot=dannot, do.mapping=domap)$score)

## TamR
# Note: risk is not implemented, the function will return NA values
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "TAMR13"=tamr13(data=ddata, annot=dannot, do.mapping=domap)$score)

## GENE70

```



```

# Need to do mapping for Affy platforms because this is based on Agilent.
# Hence the mapping rule is reversed here!
if(dn.platform[i] == "affy") { domap <- TRUE } else { domap <- FALSE }
rest <- cbind(rest, "GENE70"=gene70(data=ddata, annot=dannot, std="none",do.mapping=domap)$score)

## Pik3cags
if(dn.platform[i] == "affy") { domap <- FALSE } else { domap <- TRUE }
rest <- cbind(rest, "PIK3CA"=pik3cags(data=ddata, annot=dannot, do.mapping=domap))

## rorS
# Uses the pam50 algorithm. Need to do mapping for both Affy and Agilent
rest <- cbind(rest, "rorS"=rorS(data=ddata, annot=dannot, do.mapping=TRUE)$score)

## GENE76
# Mainly designed for Affy platforms. Has been excluded here

# BIND ALL TOGETHER
res <- rbind(res, rest)
}
names(ddemo.all) <- dn

```

For further analysis and handling of the data we store all information in one object. We also remove the duplicated patients from the analysis and take only those patients into account, that have complete information for nodal, survival and treatment status.

```

ddemot <- NULL
for(i in 1:length(ddemo.all)) {
  ddemot <- rbind(ddemot, ddemo.all[[i]][ , ddemo.coln, drop=FALSE])
}
res[complete.cases(ddemot[ , "dataset"]) & ddemot[ , "dataset"] == "VDX", "GENIUS"] <- NA

## select only untreated node-negative patients with all risk predictions
## ie(incomplete cases (where risk prediction may be missing for a sample) are subsequently removed)
# Note that increasing the number of risk prediction analyses
# may increase the number of incomplete cases
# In the pconcordance indexevious vignette for genefu version1, we were only testing 4 risk predictors,
# so we had a total of 722 complete cases remaining
# Here, we are now testing 12 risk predictors, so we only have 713 complete cases remaining.
# The difference of 9 cases between the two versions are all from the NKI dataset.
myx <- complete.cases(res, ddemot[ , c("node", "treatment")]) &
  ddemot[ , "treatment"] == 0 & ddemot[ , "node"] == 0 & !is.element(rownames(ddemot), duPL)

res <- res[myx, , drop=FALSE]
ddemot <- ddemot[myx, , drop=FALSE]

```

To compare the risk score performances, we compute the concordance index¹, which is the probability that, for a pair of randomly chosen comparable samples, the sample with the higher risk prediction will experience an event before the other sample or belongs to a higher binary class.

```

cc.res <- complete.cases(res)
datasetList <- c("MAINZ", "TRANSBIG", "UPP", "UNT", "NKI")
riskPList <- c("AURKA", "ESR1", "ERBB2", "NPI", "GGI", "GENIUS",

```

¹The same analysis could be performed with D index and hazard ratio by using the functions `D.index` and `hazard.ratio` from the *survcomp* package respectively

```

        "EndoPredict", "OncotypeDx", "TAMR13", "GENE70", "PIK3CA", "rorS")
setT <- setE <- NULL
resMatrix <- as.list(NULL)

for(i in datasetList)
{
  dataset.only <- ddemot[, "dataset"] == i
  patientsAll <- cc.res & dataset.only

  ## set type of available survival data
  if(i == "UPP") {
    setT <- "t.rfs"
    setE <- "e.rfs"
  } else {
    setT <- "t.dmfs"
    setE <- "e.dmfs"
  }

  # Calculate cindex computation for each predictor
  for (Dat in riskPList)
  {
    cindex <- t(apply(X=t(res[patientsAll,Dat]), 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)); },
      y=ddemot[patientsAll,setT], z=ddemot[patientsAll, setE]))

    resMatrix[[Dat]] <- rbind(resMatrix[[Dat]], cindex)
  }
}

```

Using a random-effects model we combine the dataset-specific performance estimated into overall estimates for each risk prediction model:

```

for(i in names(resMatrix)){
  #Get a meta-estimate
  ceData <- combine.est(x=resMatrix[[i]][, "cindex"], x.se=resMatrix[[i]][, "cindex.se"], hetero=TRUE)
  cLower <- ceData$estimate + qnorm(0.025, lower.tail=TRUE) * ceData$se
  cUpper <- ceData$estimate + qnorm(0.025, lower.tail=FALSE) * ceData$se

  cindex0 <- cbind("cindex"=ceData$estimate, "cindex.se"=ceData$se, "lower"=cLower, "upper"=cUpper)
  resMatrix[[i]] <- rbind(resMatrix[[i]], cindex0)
  rownames(resMatrix[[i]]) <- c(datasetList, "Overall")
}

```

In order to compare the different risk prediction models we compute one-sided p-values of the meta-estimates:

```

pv <- sapply(resMatrix, function(x) { return(x["Overall", c("cindex", "cindex.se")]) })
pv <- apply(pv, 2, function(x) { return(pnorm((x[1] - 0.5) / x[2], lower.tail=x[1] < 0.5)) })
printPV <- matrix(pv, ncol=length(names(resMatrix)))
rownames(printPV) <- "P-value"
colnames(printPV) <- names(pv)
printPV<-t(printPV)

```

And print the table of P-values:

```
xtable(printPV, digits=c(0, -1))
```

	P-value
AURKA	4.5E-08
ESR1	6.5E-03
ERBB2	4.6E-01
NPI	1.8E-15
GGI	2.8E-14
GENIUS	6.1E-23
EndoPredict	7.7E-13
OncotypeDx	9.3E-14
TAMR13	2.5E-07
GENE70	1.8E-10
PIK3CA	2.3E-03
rorS	9.5E-11

The following figures represent the risk score performances measured by the concordance index each of the prognostic predictors.

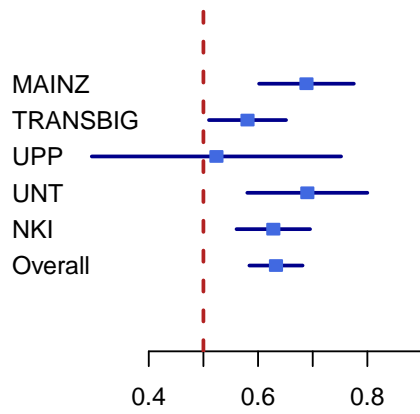
```
RiskPList <- c("AURKA","ESR1","ERBB2","NPI", "GGI", "GENIUS",
               "EndoPredict","OncotypeDx","TAMR13","GENE70","PIK3CA","rorS")
datasetListF <- c("MAINZ","TRANSBIG","UPP","UNT","NKI", "Overall")
myspace <- " "
par(mfrow=c(2,2))
for (RP in RiskPList)
{
  #<<forestplotDat,fig=TRUE>>=
  ## Forestplot
  tt <- rbind(resMatrix[[RP]][1:5,],
              "Overall"=resMatrix[[RP]][6,])

  tt <- as.data.frame(tt)
  labeltext <- (datasetListF)

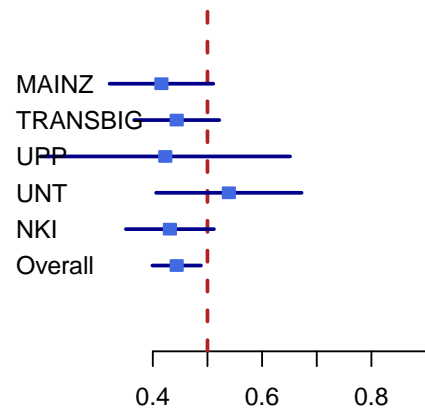
  r.mean <- c(tt$cindex)
  r.lower <- c(tt$lower)
  r.upper <- c(tt$upper)

  metaplot.surv(mn=r.mean, lower=r.lower, upper=r.upper, labels=labeltext, xlim=c(0.4,0.9),
                boxsize=0.5, zero=0.5,
                col=meta.colors(box="royalblue",line="darkblue",zero="firebrick"),
                main=paste(RP))
}
```

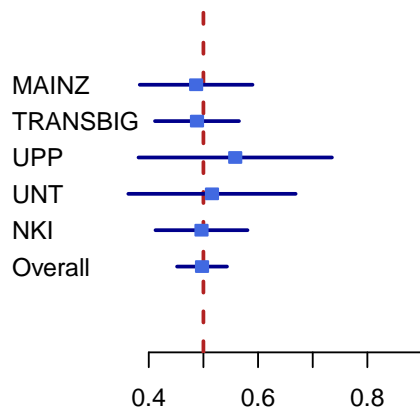
AURKA



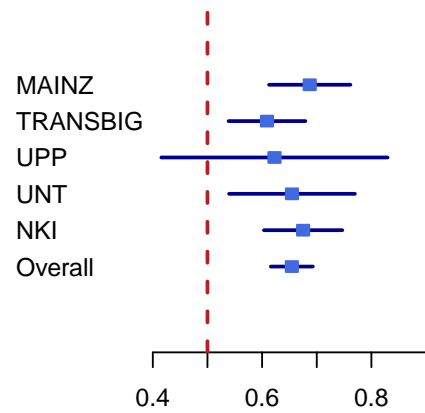
ESR1



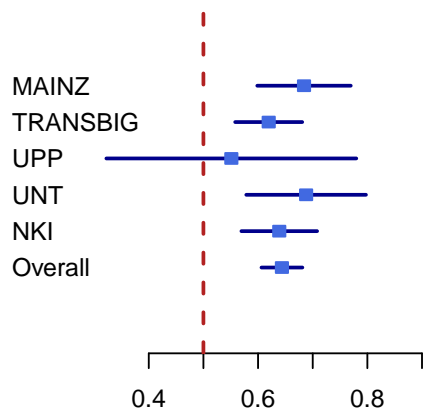
ERBB2



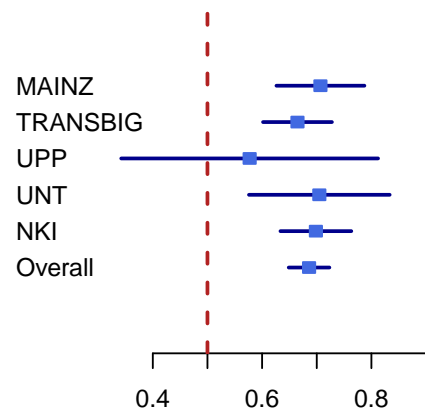
NPI



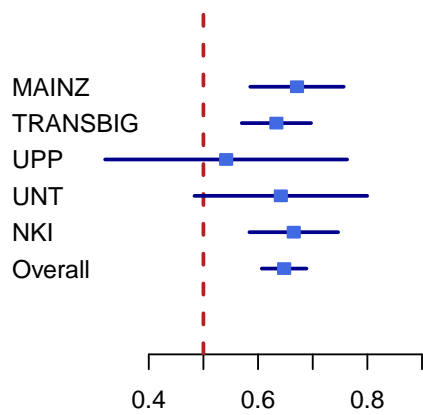
GGI



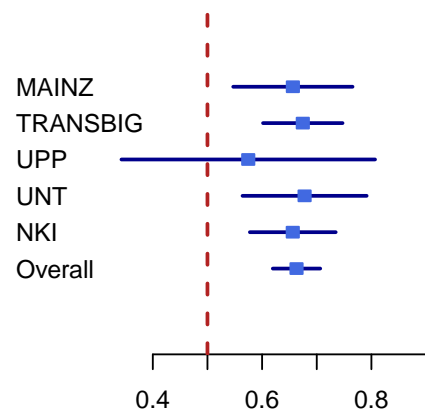
GENIUS

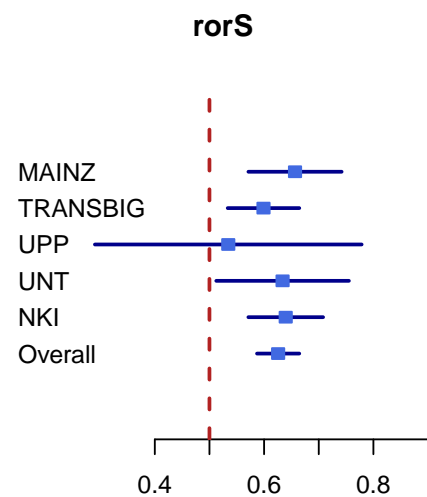
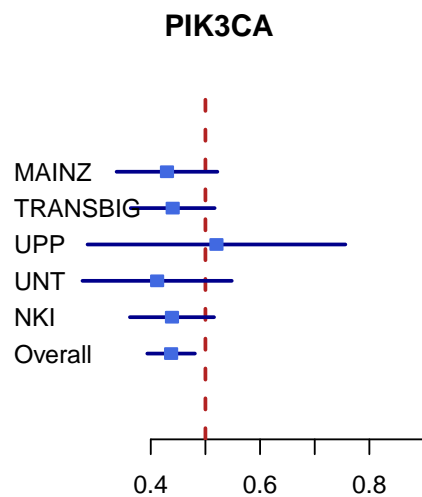
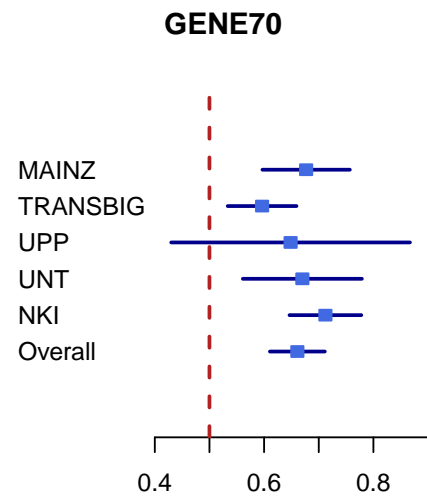
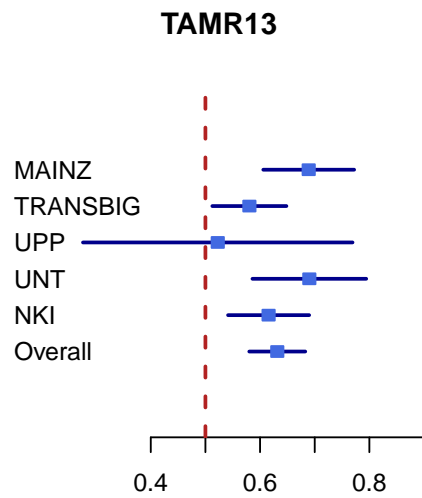


EndoPredict



OncotypeDx





```
##@
#
```

We can also represent the overall estimates across all prognostic predictors, across all the datasets.

```
## Overall Forestplot
mybigspace <- " "
tt <- rbind("OverallA"=resMatrix[["AURKA"]][6,],
            "OverallE1"=resMatrix[["ESR1"]][6,],
            "OverallE2"=resMatrix[["ERBB2"]][6,],
            "OverallN"=resMatrix[["NPI"]][6,],
            "OverallM"=resMatrix[["GGI"]][6,],
            "OverallG"=resMatrix[["GENIUS"]][6,],
            "OverallE3"=resMatrix[["EndoPredict"]][6,],
```

```

    "OverallOD"=resMatrix[["OncotypeDx"]][6,],
    "OverallT"=resMatrix[["TAMR13"]][6,],
    "OverallG70"=resMatrix[["GENE70"]][6,],
    "OverallP"=resMatrix[["PIK3CA"]][6,],
    "OverallR"=resMatrix[["rorS"]][6,]
  )

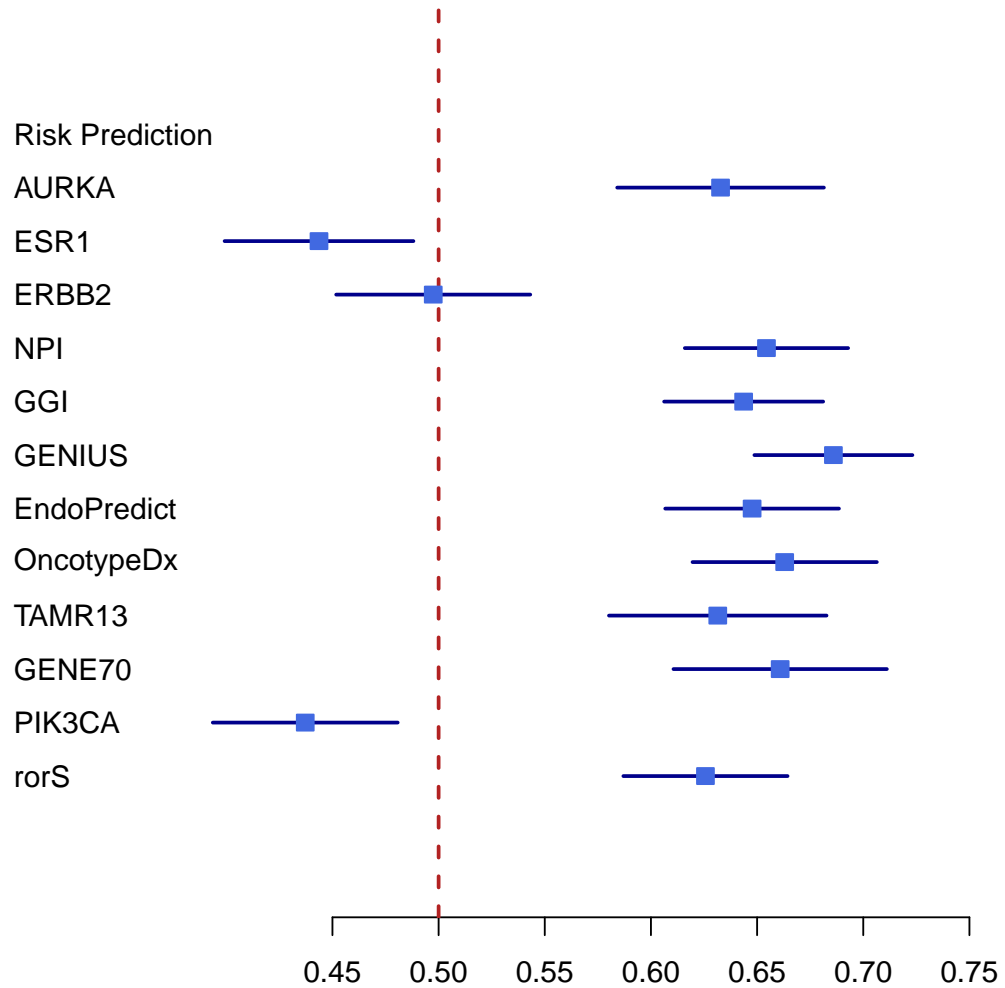
tt <- as.data.frame(tt)
labeltext <- cbind(c("Risk Prediction","AURKA","ESR1","ERBB2","NPI",
                    "GGI","GENIUS","EndoPredict","OncotypeDx","TAMR13","GENE70","PIK3CA","rorS"))

r.mean <- c(NA,tt$cindec)
r.lower <- c(NA,tt$lower)
r.upper <- c(NA,tt$upper)

metaplot.surv(mn=r.mean, lower=r.lower, upper=r.upper, labels=labeltext, xlim=c(0.45,0.75),
              boxsize=0.5, zero=0.5,
              col=meta.colors(box="royalblue",line="darkblue",zero="firebrick"),
              main="Overall Concordance Index")

```

Overall Concordance Index



In order to assess the difference between the risk scores, we compute the concordance indices with their p-values and compare the estimates with the `cindex.comp.meta` with a paired student t test.

```
cc.res <- complete.cases(res)
datasetList <- c("MAINZ", "TRANSBIG", "UPP", "UNT", "NKI")
riskPList <- c("AURKA", "ESR1", "ERBB2", "NPI", "GGI", "GENIUS",
               "EndoPredict", "OncotypeDx", "TAMR13", "GENE70", "PIK3CA", "rorS")
setT <- setE <- NULL
resMatrixFull <- as.list(NULL)

for(i in datasetList)
{
  dataset.only <- ddemot[, "dataset"] == i
  patientsAll <- cc.res & dataset.only
}
```



```

## set type of available survival data
if(i == "UPP") {
  setT <- "t.rfs"
  setE <- "e.rfs"
} else {
  setT <- "t.dmfs"
  setE <- "e.dmfs"
}

## cindex and p-value computation per algorithm
for (Dat in riskPList)
{
  cindex <- t(apply(X=t(res[patientsAll,Dat]), 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=ddemot[patientsAll,setT], z=ddemot[patientsAll, setE]))

  resMatrixFull[[Dat]] <- rbind(resMatrixFull[[Dat]], cindex)
}

for(i in names(resMatrixFull)){
  rownames(resMatrixFull[[i]]) <- datasetList
}

ccmData <- tt <- rr <- NULL
for(i in 1:length(resMatrixFull)){
  tt <- NULL
  for(j in 1:length(resMatrixFull)){
    if(i != j) { rr <- cindex.comp.meta(list.cindex1=resMatrixFull[[i]],
                                         list.cindex2=resMatrixFull[[j]], hetero=TRUE)$p.value }
    else { rr <- 1 }
    tt <- cbind(tt, rr)
  }
  ccmData <- rbind(ccmData, tt)
}
ccmData <- as.data.frame(ccmData)
colnames(ccmData) <- riskPList
rownames(ccmData) <- riskPList

```

Table 2 displays the for multiple testing uncorrected p-values for the comparison of the different methods:

```

xtable(ccmData, digits=c(0, rep(-1,ncol(ccmData))), size="footnotesize")

```

We correct the p-value with Holms method:

```

ccmDataPval <- matrix(p.adjust(data.matrix(ccmData), method="holm"),ncol=length(riskPList),
                      dimnames=list(rownames(ccmData),colnames(ccmData)))

```

Table 3 displays the corrected p-values:

```

xtable(ccmDataPval, digits=c(0, rep(-1,ncol(ccmDataPval))), size="footnotesize")

```

	AURKA	ESR1	ERBB2	NPI	GGI	GENIUS	EndoPredict	OncotypeDx	TAMR13
AURKA	1.0E+00	1.0E-07	5.0E-05	7.8E-01	6.9E-01	9.8E-01	7.3E-01	8.8E-01	4.8E-01
ESR1	1.0E+00	1.0E+00	9.5E-01	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
ERBB2	1.0E+00	4.6E-02	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
NPI	2.2E-01	3.5E-11	2.8E-07	1.0E+00	3.3E-01	9.0E-01	3.9E-01	6.3E-01	2.1E-01
GGI	3.1E-01	5.0E-10	1.3E-06	6.7E-01	1.0E+00	9.7E-01	5.8E-01	8.2E-01	3.1E-01
GENIUS	2.3E-02	2.2E-15	4.8E-10	1.0E-01	2.9E-02	1.0E+00	5.6E-02	1.8E-01	2.1E-02
EndoPredict	2.7E-01	2.3E-09	1.1E-06	6.1E-01	4.2E-01	9.4E-01	1.0E+00	7.6E-01	2.7E-01
OncotypeDx	1.2E-01	9.7E-10	2.4E-07	3.7E-01	1.8E-01	8.2E-01	2.4E-01	1.0E+00	1.4E-01
TAMR13	5.2E-01	1.6E-07	7.7E-05	7.9E-01	6.9E-01	9.8E-01	7.3E-01	8.6E-01	1.0E+00
GENE70	1.5E-01	9.1E-09	3.1E-06	4.1E-01	2.3E-01	8.2E-01	3.0E-01	5.3E-01	1.6E-01
PIK3CA	1.0E+00	5.9E-01	9.8E-01	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
rorS	6.2E-01	1.2E-08	1.8E-05	8.8E-01	8.3E-01	1.0E+00	8.5E-01	9.5E-01	5.9E-01

	AURKA	ESR1	ERBB2	NPI	GGI	GENIUS	EndoPredict	OncotypeDx	TAMR13
AURKA	1.0E+00	1.3E-05	6.0E-03	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
ESR1	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
ERBB2	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
NPI	1.0E+00	4.9E-09	3.5E-05	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
GGI	1.0E+00	6.9E-08	1.5E-04	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
GENIUS	1.0E+00	3.1E-13	6.5E-08	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
EndoPredict	1.0E+00	3.1E-07	1.4E-04	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
OncotypeDx	1.0E+00	1.3E-07	3.0E-05	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
TAMR13	1.0E+00	2.0E-05	9.1E-03	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
GENE70	1.0E+00	1.2E-06	3.7E-04	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
PIK3CA	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00
rorS	1.0E+00	1.5E-06	2.2E-03	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00

6 References

The following is a list of publications that have cited *genefu* (Version 1) in the past.

Where *genefu* was used in subtyping

- Larsen, M.J. et al., 2014. Microarray-Based RNA Profiling of Breast Cancer: Batch Effect Removal Improves Cross-Platform Consistency. *BioMed Research International*, 2014, pp.1-11.
- Miller, T.W. et al., 2011. A gene expression signature from human breast cancer cells with acquired hormone independence identifies MYC as a mediator of antiestrogen resistance. *Clinical cancer research : an official journal of the American Association for Cancer Research*, 17(7), pp.2024-2034.
- Karn, T. et al., 2011. Homogeneous Datasets of Triple Negative Breast Cancers Enable the Identification of Novel Prognostic and Predictive Signatures S. Ranganathan, ed. *PloS one*, 6(12), p.e28403.

Where *genefu* was used in Comparing Subtyping Schemes

- Haibe-Kains, B. et al., 2012. A three-gene model to robustly identify breast cancer molecular subtypes. *Journal of the National Cancer Institute*, 104(4), pp.311-325.
- Curtis, C. et al., 2012. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*.
- Balko, J.M. et al., 2012. Profiling of residual breast cancers after neoadjuvant chemotherapy identifies DUSP4 deficiency as a mechanism of drug resistance. *Nature medicine*, 18(7), pp.1052-1059.

Paquet, E.R. and Hallett, M.T., 2015. Absolute Assignment of Breast Cancer Intrinsic Molecular Subtype. Journal of the National Cancer Institute, 107(1), pp.dju357-dju357.

Patil, P. et al., 2015. Test set bias affects reproducibility of gene signatures. Bioinformatics, p.btv157.

Where genefu was used to Compute Prognostic gene signature scores

Haibe-Kains, B. et al., 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), pp.2200-2208.

Haibe-Kains, B. et al., 2010. A fuzzy gene expression-based computational approach improves breast cancer prognostication. Genome biology, 11(2), p.R18.

Madden, S.F. et al., 2013. BreastMark: An Integrated Approach to Mining Publicly Available Transcriptomic Datasets Relating to Breast Cancer Outcome. Breast Cancer Research, 15(4), p.R52.

Fumagalli, D. et al., 2014. Transfer of clinically relevant gene expression signatures in breast cancer: from Affymetrix microarray to Illumina RNA-Sequencing technology. BMC genomics, 15(1), p.1008.

Beck A.H. et al., 2013. Significance Analysis of Prognostic Signatures. PLoS Computational Biology, 9(1), e1002875.

As well as other publications

APOBEC3B expression in breast cancer reflects cellular proliferation, while a deletion polymorphism is associated with immune activation. Cescon DW, Haibe-Kains B, Mak TW. Proc Natl Acad Sci U S A. 2015 Mar 3;112(9):2841-6. doi: 10.1073/pnas.1424869112. Epub 2015 Feb 17. PMID: 25730878

Radovich M. et al., 2014. Characterizing the heterogeneity of triple-negative breast cancers using microdissected normal ductal epithelium and RNA-sequencing. Breast cancer research and treatment, 143(1), pp.57-68.

Tramm T. et al., 2014. Relationship between the prognostic and predictive value of the intrinsic subtypes and a validated gene profile predictive of loco-regional control and benefit from post-mastectomy radiotherapy in patients with high-risk breast cancer. Acta Oncologica 53(10), pp.1337-1346.

Doan, T.B. et al., 2014. Breast cancer prognosis predicted by nuclear receptor-coregulator networks. Molecular oncology 8(5), pp.998-1013.

7 Session Info

```
toLatex(sessionInfo())
```

- R version 3.2.0 Patched (2015-05-20 r68389), x86_64-apple-darwin10.8.0
- Locale: en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8
- Base packages: base, datasets, graphics, grDevices, grid, methods, parallel, stats, utils
- Other packages: AIMS 1.0.0, Biobase 2.28.0, BiocGenerics 0.14.0, biomaRt 2.24.0, breastCancerMAINZ 1.6.0, breastCancerNKI 1.6.0, breastCancerTRANSBIG 1.6.0, breastCancerUNT 1.6.0, breastCancerUPP 1.6.0, caret 6.0-47, cluster 2.0.1, DBI 0.3.1, dynamicTreeCut 1.62, e1071 1.6-4, fastcluster 1.1.16, genefu 2.0.1, ggplot2 1.0.1, iC10 1.1.2, iC10TrainingData 1.0.1, knitr 1.10.5, lattice 0.20-31, mclust 5.0.1, pamr 1.55, prodlm 1.5.1, rmeta 2.16, RSQLite 1.0.0, survcomp 1.18.0, survival 2.38-1, WGCNA 1.47, xtable 1.7-4

- Loaded via a namespace (and not attached): acepack 1.3-3.3, amap 0.8-14, AnnotationDbi 1.30.1, bitops 1.0-6, bootstrap 2015.2, BradleyTerry2 1.0-6, brglm 0.5-9, car 2.0-25, class 7.3-12, codetools 0.2-11, colorspace 1.2-6, digest 0.6.8, doParallel 1.0.8, evaluate 0.7, foreach 1.4.2, foreign 0.8-63, formatR 1.2, Formula 1.2-1, GenomeInfoDb 1.4.0, GO.db 3.1.2, gridExtra 0.9.1, gtable 0.1.2, gtools 3.5.0, highr 0.5, Hmisc 3.16-0, impute 1.42.0, IRanges 2.2.2, iterators 1.0.7, KernSmooth 2.23-14, latticeExtra 0.6-26, lava 1.4.0, limma 3.24.5, lme4 1.1-8, magrittr 1.5, MASS 7.3-40, Matrix 1.2-0, matrixStats 0.14.2, mgcv 1.8-6, minqa 1.2.4, munsell 0.4.2, nlme 3.1-120, nloptr 1.0.4, nnet 7.3-9, pbkrtest 0.4-2, plyr 1.8.3, preprocessCore 1.30.0, proto 0.3-10, quantreg 5.11, RColorBrewer 1.1-2, Rcpp 0.11.6, RCurl 1.95-4.6, reshape 0.8.5, reshape2 1.4.1, rpart 4.1-9, S4Vectors 0.6.0, scales 0.2.5, SparseM 1.6, splines 3.2.0, stats4 3.2.0, stringi 0.4-1, stringr 1.0.0, SuppDists 1.1-9.1, survivalROC 1.0.3, tools 3.2.0, XML 3.98-1.1