Generation of a Predictive Melphalan Resistance Index by Drug Screen of B-Cell Cancer Cell Lines

Martin Boegsted, Johanne M. Holst, Kirsten Fogd, Steffen Falgreen, Suzette Sorensen, Alexander Schmitz, Anne Bukh, Hans E. Johnsen, Mette Nyegaard, Karen Dybkaer

April 6, 2011

Steffen Falgreen and Martin Boegsted
Contents

1 Data Acquisition and R settings 4
   1.1 Data Acquisition 4
   1.2 R Settings 5
   1.3 Various Parameter Settings 6
   1.4 Loading Functions 6
   1.5 R-session Information 6

2 Loading the Retrospective Clinical Data 8
   2.1 Loading the Arkansas Data 8
      2.1.1 Reading the Metadata 8
      2.1.2 Creating Survival Objects 8
      2.1.3 Loading and Normalising the .CEL Files 8
   2.2 Loading the Hummel Data 9
      2.2.1 Reading the Metadata 9
      2.2.2 Creating the Survival Object 10
      2.2.3 Loading and Normalising the .CEL Files 10

3 Establishment of the NCI60 Gene Expression Data 11

4 Establishment of the NCI60 Dose Response Data 13
   4.1 The NCI60 Cell Line Metadata 14
   4.2 Converting .csv to .Rdata 14

5 Establishment of the NCI60 LDA Classifier 18
   5.1 Definition of the NCI60 Resistance Classes 18
   5.2 Cross-Validation 19
   5.3 Establishing the LDA Classifier 24

6 Testing the NCI60 LDA Classifier on the Arkansas Data 30
   6.1 Predicting Melphalan Resistance Classes 30
   6.2 Association between the Resistance Classes and OS 30
   6.3 Association between the Resistance Classes and EFS 31

7 Establishment of the NCI60 SPLS Resistance Index 33
### 15.1 Predicting Melphalan Resistance in the Arkansas data

- **15.1.1 Association between the Melphalan Resistance Index and OS**  
  - 108
- **15.1.2 Association between the Melphalan Resistance Index and EFS**  
  - 110

### 16 Auxiliary Functions

- **16.1 Dose Response Functions**  
  - 115
- **16.2 Model Building Functions**  
  - 127
- **16.3 Functions used for the SPLS Analysis**  
  - 127
- **16.4 Functions used in the Simple Analysis**  
  - 136
- **16.5 Annotation and Plotting Tools**  
  - 136
1 Data Acquisition and R settings

1.1 Data Acquisition

The *NCI60* Cancer Cell Line Panel

The *NCI60* panel consists of 59 cancer cell lines. Gene expression data, dose response and metadata are publicly available.

Affymetrix GeneChip Human Genome U133A (HG-U133A) arrays with gene expression data are available as .CEL files via


under GEO accession number GSE5720.

The DTP human tumor cell line drug screen data (August 2008 release) were downloaded from the website


The metadata were downloaded May 22, 2009, from


The *BCell* Cancer Cell Line Panel

The *BCell* panel consists of 18 cancer cell lines all originating from patients suffering from a B-cell malignancy. Affymetrix GeneChip Human Genome U133 Plus 2.0 arrays (HG-U133 Plus 2.0) have been used to obtain the gene expression data of the cell lines.

The gene expression data have been deposited at


under GEO accession number GSE22759.

Retrospective Clinical Data Sets

The *Arkansas* Data

The *Arkansas* data consist of data from 565 patients with multiple myeloma (MM). The data include gene expressions from a HG-U133Plus 2.0 array platform and information about overall survival (OS) and event free survival (EFS). The metadata and .CEL files are available at


under GEO accession number GSE24080.
The hummel Data

The Hummel data consist of data from patients suffering from diffuse large B-cell lymphoma (DLBCL). The data include gene expressions obtained through the HG-U133A microarray platform and OS. The metadata and .CEL files are available at


1.2 R Settings

```r
> options(width = 70)
> options(continue = " ")
> options(digits = 22)
> windowsFonts(Times = windowsFont("TT Times New Roman"))
> pdf.options(family = "Times")
> par(family = "Times")
```

The libraries which are required for the analysis are loaded below

```r
> require(affy)
> require(annaffy)
> require(annotate)
> require(annotationTools)
> require(arrayQualityMetrics)
> require(Biobase)
> require(caTools)
> require(Design)
> require(foreign)
> require(genefilter)
> require(geneplotter)
> require(gdata)
> require(glmnet)
> require(gplots)
> require(hgu133a.db)
> require(hgu133a2.db)
> require(hgu133aprobe)
> require(hgu133plus2.db)
> require(limma)
> require(nlme)
> require(outliers)
> require(pls)
> require(prada)
> require(quantreg)
```
1.3 Various Parameter Settings

In the following code chunk the color scheme for the plots is determined, the cut.off values used for the KM plots are set, and the working catalogue is specified.

```r
> our.colscheme <- c("black", "darkgrey", "red")
> names(our.colscheme) <- c("Sensitive", "Intermediate", "Resistant")
> cut.points <- c(0.25, 0.75)
> setwd(working.directory)
```

1.4 Loading Functions

Auxilary functions used throughout are printed out in Section 16.

1.5 R-session Information

Information regarding the R-session is printed below

```r
> toLatex(sessionInfo())
```

- R version 2.12.1 (2010-12-16), x86_64-pc-mingw32
- Locale: LC_COLLATE=Danish_Denmark.1252, LC_CTYPE=Danish_Denmark.1252,
  LC_MONETARY=Danish_Denmark.1252, LC_NUMERIC=C,
  LC_TIME=Danish_Denmark.1252
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, utils
- Other packages: affy 1.28.0, affyPLM 1.26.0, annaffy 1.22.0, annotate 1.28.0,
  AnnotationDbi 1.12.0, annotationTools 1.22.0, arrayQualityMetrics 3.2.4,
  Biobase 2.10.0, bitops 1.0-4.1, cacheSweave 0.4-5, caTools 1.11, corpcor 1.5.7,
  DBI 0.2-5, Design 2.3-0, entropy 1.1.5, fdrtool 1.2.6, filehash 2.1-1, foreign 0.8-41,
  gcrma 2.22.0, gdata 2.8.1, genefilter 1.32.0, geneplotter 1.28.0, glmnet 1.5.1,
  GO.db 2.4.5, gplots 2.8.0, gtools 2.6.2, hgu133a.db 2.4.5, hgu133a2.db 2.4.5,
  hgu133aprobe 2.7.0, hgu133plus2.db 2.4.5, Hmisc 3.8-3, KEGG.db 2.4.5,
lattice 0.19-13, limma 3.6.9, MASS 7.3-9, Matrix 0.999375-46, mvtnorm 0.9-95, nlme 3.1-97, nnet 7.3-1, org.Hs.eg.db 2.4.6, outliers 0.13-3, pcaPP 1.8-3, pls 2.1-0, prada 1.26.0, preprocessCore 1.12.0, quantreg 4.53, RColorBrewer 1.0-2, robustbase 0.5-0-1, rrcov 1.1-00, RSQLite 0.9-4, sda 1.1.0, SparseM 0.86, spls 2.1-0, stashR 0.3-3, statmod 1.4.8, survival 2.36-2, TTR 0.20-2, vsn 3.18.0, XML 3.2-0.2, xtable 1.5-6, xts 0.7-5, zoo 1.6-4

- Loaded via a namespace (and not attached): affyio 1.18.0, beadarray 2.0.2, Biostrings 2.18.2, cluster 1.13.2, digest 0.4.2, hwriter 1.3, IRanges 1.8.8, latticeExtra 0.6-14, marray 1.28.0, simpleaffy 2.26.1, stats4 2.12.1, SVGAnnotation 0.7-2, tools 2.12.1
2 Loading the Retrospective Clinical Data

2.1 Loading the Arkansas Data

A directory which includes the Arkansas data and a directory to store the generated data are created.

```r
> Arkansas.ext.dir <- file.path(getwd(), "External data/Arkansas")
> Arkansas.gen.dir <- file.path(getwd(), "Generated data/Arkansas")
```

The directory storing generated data is created.

```r
> dir.create(path = file.path(Arkansas.gen.dir),
              showWarnings = FALSE, recursive = TRUE, mode = "0777")
```

2.1.1 Reading the Metadata

The metadata was transformed manually from XLS to the semicolon-separated file Arkansasmetadata.csv.

```r
> metadataArkansas <- read.csv2(file.path(Arkansas.ext.dir, "Metadata", "Arkansasmetadata.csv"))
> n.Arkansas.all <- nrow(metadataArkansas)
```

2.1.2 Creating Survival Objects

The OS and EFS survival objects are created using the `Surv` function from the `survival` package.

```r
> metadataArkansas$OS <- Surv(as.numeric(metadataArkansas$OS.months.),
                              metadataArkansas$OS.censor == 1)
> metadataArkansas$EFS <- Surv(as.numeric(metadataArkansas$EFS.months.),
                              metadataArkansas$EFS.censor == 1)
```

2.1.3 Loading and Normalising the .CEL Files

The .CEL files are loaded into R. The metadata are restricted to patients with an associated .CEL file.

```r
> row.names(metadataArkansas) <- metadataArkansas$CELfilename
> files <- dir(file.path(Arkansas.ext.dir, "Celfiles"))
> metadataArkansas <- metadataArkansas[substring(files, 11, 100), ]
> n.Arkansas <- nrow(metadataArkansas)
> row.names(metadataArkansas) <- metadataArkansas$PATID
```
This resulted in the removal of 6 patients. Finally, the .CEL files are read and pre-processed using the RMA approach.

```r
> GEPArkansas.file <- file.path(Arkansas.gen.dir, "GEPArkansas.Rdata")
> if(file.exists(GEPArkansas.file)){
  load(GEPArkansas.file)
} else{
  phenoData <- new("AnnotatedDataFrame", data = metadataArkansas)
  cel.files <- file.path(Arkansas.ext.dir, "Celfiles", files)

  GEPArkansas <- just.rma(filenames = cel.files,
                         phenoData = phenoData)

  # The data is only saved if the rownames in the phenoData match
  # the colnames of the gene expression data.
  if(all(rownames(pData(GEPArkansas)) ==
        colnames(exprs(GEPArkansas)))){
    save(GEPArkansas, file=GEPArkansas.file)
  }
}
```

### 2.2 Loading the Hummel Data

Directories which include the Hummel and the generated data are created.

```r
> Hummel.ext.dir <- file.path(getwd(), "External data/Hummel")
> Hummel.gen.dir <- file.path(getwd(), "Generated data/Hummel")

> dir.create(path = file.path(Hummel.gen.dir),
             showWarnings = FALSE, recursive = TRUE, mode = "0777")
```

#### 2.2.1 Reading the Metadata

```r
> metadataHummel <- read.csv(file.path(Hummel.ext.dir,
                                        "Metadata", "GSE4475_Clinical.csv"))
```

Attention is restricted to patients for whom the follow up time is observed and is greater than 0 and information regarding the given chemotherapy is registered.

```r
> n.na.FU <- dim(metadataHummel[is.na(metadataHummel$FollowupTime..months), ])[1]
> metadataHummel <- metadataHummel[!is.na(metadataHummel$FollowupTime..months), ]
> n.zero <-
```
dim(metadataHummel[metadataHummel$FollowupTime..months == 0, ])[1]
> metadataHummel <- metadataHummel[metadataHummel$FollowupTime..months > 0, ]
> n.na.chemo <- dim(metadataHummel[is.na(metadataHummel$Chemotherapy), ])[1]
> metadataHummel <- metadataHummel[!is.na(metadataHummel$Chemotherapy), ]

Only patients who have received "CHOP-like" chemotherapy are used.

> chosen <- as.character(metadataHummel$Chemotherapy) == "CHOP-like"
> n.not.CHOP <- dim(metadataHummel[!chosen, ])[1]
> metadataHummel <- metadataHummel[chosen, ]

This results in a dataset with 87 patients. Table 2.1 shows the number of patients removed in each step.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Patients excluded</th>
</tr>
</thead>
<tbody>
<tr>
<td>No follow up information</td>
<td>62</td>
</tr>
<tr>
<td>Follow up equal to 0</td>
<td>4</td>
</tr>
<tr>
<td>No chemotherapy information</td>
<td>16</td>
</tr>
<tr>
<td>Chemotherapy other than CHOP</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 2.1: Number of excluded patients in each step of the Hummel data set.

### 2.2.2 Creating the Survival Object

> metadataHummel$OS <- Surv(as.numeric(metadataHummel$FollowupTime..months),
   metadataHummel$Survival.Status == "dead")

### 2.2.3 Loading and Normalising the .CEL Files

> file <- file.path("Generated data", "Hummel", "HummelHGU133A.Rdata")
> if(file.exists(file)){
    load(file)
  }else{
    GEPHummel<-
    justRMA(filenames = file.path("External data/Hummel/Celfiles",
        paste(metadataHummel$GEO.ID,.CEL", sep = ")
    dir.create(path=paste("Generated data/Hummel/", sep = ")
    save(GEPHummel, file = file)
  }

10
3 Establishment of the NCI60 Gene Expression Data

The directories which are used with the NCI60 cell line panel are defined

> NCI60.ext.dir <- file.path(getwd(), "External data", "NCI60")
> NCI60.gen.dir <- file.path(getwd(), "Generated data", "NCI60")

and created

> dir.create(NCI60.gen.dir,
showWarnings = FALSE, recursive = TRUE, mode = "0777")

The NCI60 .CEL files are read into R in the following code chunk. For pre-processing of the arrays the RMA approach is used.

> GEPNCI60.file <- file.path(NCI60.gen.dir, "GEPNCI60.RMA.Rdata")
> if(file.exists(GEPNCI60.file)){
  load(GEPNCI60.file)
}else{
  ## The metadata for the .cel files are stored in the file
  NCI60.cel.names <-
  read.table(file.path(NCI60.ext.dir, "Celfiles", "u133a.fnames.txt"),
  sep = " ",
  header = TRUE,
  check.names = TRUE,
  row.names = 1,
  col.names = c("cell.line.name", "array.type"))

  ## The NCI60 data includes both HGU133A and HGU133B arrays. However, only the HGU133A arrays are used in the analysis
  NCI60.cel.names <-
  NCI60.cel.names[!grep("hg133a", NCI60.cel.names$array.type),]

  ## The cell line IGROV1 is duplicated and data from experiment a21 are chosen
  NCI60.cel.names <-
  NCI60.cel.names[NCI60.cel.names$cell.line.name !="IGROV1" | NCI60.cel.names$array.type != "hg133a31",]
## Some of the cell line names are extended with /ATCC and
## the following code removes the extension

```r
NCI60.cel.names$cell.line.name <-
gsub("/ATCC", ",", as.character(NCI60.cel.names$cell.line.name))
```

## Finally, the .CEL files are imported and RMA normalised

```r
GEPNCI60 <-
  justRMA(filenames = paste(rownames(NCI60.cel.names),
    ",.CEL", sep = ","),
    celfile.path = file.path(NCI60.ext.dir,"Celfiles"))
```

## The sample names of the exprs object are
## set equal to sample names from the metadata

```r
sampleNames(GEPNCI60) <-
  as.character(NCI60.cel.names$cell.line.name)
sampleNames(GEPNCI60) <- gsub("-", "\", sampleNames(GEPNCI60))
save(GEPNCI60, file = GEPNCI60.file)
```

```r
c > n.NCI60 <- dim(GEPNCI60)[2]
c > p.NCI60 <- dim(GEPNCI60)[1]
c > GEPNCI60133a <- GEPNCI60
```
4 Establishment of the NCI60 Dose Response Data

Parts of the code used in the present chapter have been developed by Kevin Coombes and Keith Baggerley and can be downloaded from the website

- [http://bioinformatics.mdanderson.org/Supplements/ReproRsCh-Chemo/](http://bioinformatics.mdanderson.org/Supplements/ReproRsCh-Chemo/)

The two output catalogues are created by the code chunk below.

```r
> dir.create(path=figure.output,
  showWarnings = FALSE, recursive = TRUE, mode = "0777")
> dir.create(path=table.output,
  showWarnings = FALSE, recursive = TRUE, mode = "0777")
```

The DTP human tumor cell line drug screen data (August 2008 release) were downloaded from the website


This contained the three files:

1. cancer60gi50.lis
2. cancer60lc50.lis
3. cancer60tg.i lis.

Dose response data regarding melphalan induced GI$_{50}$ values are extracted. The data are stored in the directory below.

```r
> setwd("External data/NCI60/Doseresponse/Processed/")
```

A Perl script which extracts the melphalan data from the GI$_{50}$ data is invoked.

```r
> file <- paste("CANCER60", "GI50", ".LIS", sep="")
> sourceFile <- paste("../Raw/", file, sep = ")
> targetFile <- paste(file, "MMproject", ",csv", sep = ")
> command <- paste("perl strip4MMproject.pl", sourceFile, targetFile)
> if(file.exists(targetFile)){
  system(command)
}
> setwd("../../../../")
```
4.1 The NCI60 Cell Line Metadata

The Cell Miner metadata were downloaded May 22, 2009, from

• http://discover.nci.nih.gov/cellminer/celllineinfo.do

into the excel document cellLine_metadata_783789569.xls. The file was transformed manually from XLS to the semicolon-separated file NCI60metadata.csv.

```r
> metadataNCI60 <-
   read.csv2(file.path(NCI60.ext.dir, "Metadata", "NCI60metadata.csv"))

Change the problematic names.

> metadataNCI60$Cell.Name <- gsub("_", ".", metadataNCI60$Cell.Name)

Save dataset in the data folder.

> save(metadataNCI60,
   file = file.path(NCI60.gen.dir, "metadataNCI60.Rdata"))
```

4.2 Converting .csv to .Rdata

```r
> nsc <- "8806"
> names(nsc) <- "Melphalan"

The function getNSC is able to read the data from the comma-separated source files.

> nci60Dir <- file.path(NCI60.ext.dir, "Doseresponse", "Processed")
> gi50 <- getNSC("cancer60gi50.lis.MMproject.csv", "NLOGGI50", nci60Dir)

Drug response data of the cell lines which are present in the NCI60 metadata are chosen.

The cell line name stored in the metadata includes an abbreviation describing which tissue group it originates from. This information is not included in the dose response data. The code chunk below removes the information ensuring that the two data sets are comparable.

```r
> xx1 <- metadataNCI60$Cell.Name
> xx2 <- unlist(strsplit(xx1,"\:"))[2 * (0:(length(xx1) - 1)) + 2]
> xx <- sort(xx2)

The rownames of the dose response data are stored

> yy <- rownames(gi50)

and corrected.
A matrix consisting of the GI50 value is established.

```r
> yy[yy == "HL-60(TB)"] <- "HL-60"
> yy[yy == "RXF 393"] <- "RXF.393"
> yy[yy == "T-47D"] <- "T47D"
> yy[yy == "NCI/ADR-RES"] <- "NCI.ADR.RES"
> yy[yy == "A549/ATCC"] <- "A549"
> yy[yy == "MDA-MB-231/ATCC"] <- "MDA.MB.231"
> yy <- gsub(" ", ",", gsub(c("-")), c("."), yy))
> novi <- yy %in% xx

The dose response data regarding melphalan are stored in a .Rdata object.

```r
> save(nsc, GI50, file = file.path(NCI60.gen.dir, "nsndata.Rdata"))
```

The resistance index which is used in the analysis is established

```r
> NCI60Resistanceindex <- -1 * GI50[, 1]

When the NCI60 resistance index is used to establish a gene expression signature the gene expressions stored in the expression set GEPNCI60 are used as input variables. It is therefore necessary to align these objects.

```r
> NCI60.names <- sampleNames(GEPNCI60)
> excluded.cellline <- setdiff(names(NCI60Resistanceindex), NCI60.names)
> NCI60Resistanceindex <- cbind(NCI60Resistanceindex[NCI60.names])
> save(NCI60Resistanceindex, 
file = file.path(NCI60.gen.dir, "NCI60Resistanceindex.Rdata"))
```

The cell line names are stored in a vector for later use.

```r
> NCI60.names.sorted <- names(NCI60Resistanceindex[order(NCI60Resistanceindex),])
```

The resistance index which is used in the analysis is established and plotted with the following code chunk.

```r
> mel <- NCI60Resistanceindex[order(NCI60Resistanceindex),]
> mean <- mean(mel)
> xx <- mel - mean
> pdf(paste(figure.output,"/BarplotNCI60MEL.pdf", sep = ""), 
width = 4, height = 6)
> barplot2(xx, names.arg = names(mel),
horiz = TRUE, las = 2, offset = mean,
```
plot.grid = TRUE, grid.inc = 10,
space = 0.4,
col = "black",
cex.names = 0.4,
cex.main = 1,
cex.axis = 0.6,
xlab = "Log base 10 of the molar concentration",
main = "The NCI60 Melphalan Resistance Index"
)
> dev.off()

The NCI60 Melphalan Resistance Index

Log base 10 of the molar concentration

Figure 4.1: Barplot of the GI_{50}-value for melphalan treatment of the 59 cell lines in the NCI60 panel.

Finally, a clean up is performed.

> rm(novi, gi50, xx, xx1, xx2, yy)
5 Establishment of the *NCI60* LDA Classifier

The seed is set to ensure repeatability of the results.

```r
> set.seed(1000)
```

The LDA analysis includes all gene expressions which are expressed significantly different between the sensitive, intermediate and resistant cell lines at a significance level of 0.05.

```r
> pval <- 0.05
```

5.1 Definition of the *NCI60* Resistance Classes

The number of sensitive, intermediate and resistant cell lines are defined.

```r
> sens <- 14
> resi <- 14
> inte <- n.NCI60 - sens - resi
> class.list <- c(rep("Sensitive", sens),
                  rep("Intermediate", inte),
                  rep("Resistant", resi))
```

A data frame is constructed containing the GI\(_{50}\) values and the class of each cell line. The data frame is shown in Table 5.1.

```r
> sort.NCI60.class <-
    data.frame(NCI60Resistanceindex = sort(NCI60Resistanceindex),
                class = class.list)
> row.names(sort.NCI60.class) <- NCI60.names.sorted
```

The data set is sorted according to the NCI60 melphalan resistance index.

```r
> NCI60.class <- sort.NCI60.class[NCI60.names,]
> NCI60.class$class <- as.character(NCI60.class$class)
```

The following code chunk constructs a box plot of the GI\(_{50}\) values grouped into resistant, intermediate and sensitive cell lines. The plot is shown in Figure 5.1.

```r
> pdf(file.path(figure.output, "NCI60resvssensBoxplot.pdf"))
> sort.NCI60.class$class <- relevel(sort.NCI60.class$class,
                                    names(our.colscheme)[1])
> boxplot(sort.NCI60.class$class$NCI60Resistanceindex ~
         sort.NCI60.class$class,
         border = our.colscheme)
> dev.off()
```
Figure 5.1: Box plot of the GI50 values grouped into resistant, intermediate and sensitive cell lines.

5.2 Cross-Validation

In order to choose optimal parameters for the LDA analysis cross-validation (CV) is used. In the code chunk below the cutoff value for the unspecific filtering is likewise determined through CV. In the method the parameters of the LDA analysis are determined through CV for a wide range of values for the parameter var.cutoff in the function nsFilter. The combination of parameters in LDA and the var.cutoff which results in the maximum accuracy is chosen for further analysis. We want to investigate whether the function nsFilter chooses gene expressions which perform better than random noise. The analysis is performed as described above, however, instead of selecting the gene expression profiles through nsFilter they are chosen at random.

The various fractions to be filtered out are determined

```r
> cut offs <- seq(0.01, 0.98, by = 0.01)
```

A data frame containing the CV accuracy for LDA based on the gene expression profiles obtained through nsFilter and random selection is defined.

```r
> NCI60LDAcrossval.res <- data.frame(cut offs = cut offs,
                                     accuracy = 0,
                                     acc.rand = 0)

> NCI60LDAcrossval.res.file <-
    file.path(NCI60.gen.dir, "NCI60LDAcrossval.res.Rdata")
```
if(file.exists("NCI60LDAcrossval.res.file")) {
  load("NCI60LDAcrossval.res.file")
} else {
  for(cut.off in cut.offs) {
    cell.res <-
    data.frame(nsFilter = rep(0, n.NCI60),
               random = rep(0, n.NCI60))

    ## nsFilter expressions

    NCI60.filtered <- nsFilter(GEPNCI60[, rownames(NCI60.class)],
                               var.cutoff = cut.off)$eset

    exprs(NCI60.filtered) <- t(scale(t(exprs(NCI60.filtered))))

    ## Random expressions

    p.NCI60.filt.LDA <- length(featureNames(NCI60.filtered))
    rand.probes <- sample(p.NCI60, p.NCI60.filt.LDA)

    NCI60.random <- GEPNCI60[rand.probes, rownames(NCI60.class)]

    exprs(NCI60.random) <- t(scale(t(exprs(NCI60.random))))

    ## Analysis with random expressions

    for(cell in 1:n.NCI60) {
      cat("cell line =", cell)

      ## Leave-one-out from arrays

      looNCI60.filtered <- NCI60.random[, - cell]

      ## Leave-one-out from factor levels

      factor.NCI60 <- factor(as.character(NCI60.class$class[ - cell]))
      design.NCI60 <- model.matrix(~ 0 + factor.NCI60)
      colnames(design.NCI60) <- levels(factor.NCI60)

      contrast.matrixNCI60 <-
        makeContrasts(Resistant - Sensitive,
                      Intermediate - Sensitive,
                      levels = design.NCI60)
## Linear model fit

```r
fitNCI60 <- lmFit(looNCI60.filtered,
    design = design.NCI60)
fit2NCI60 <- contrasts.fit(fitNCI60, contrast.matrixNCI60)
fit2NCI60 <- eBayes(fit2NCI60)
NCI60lmFit <- topTable(fit2NCI60,
    coef = 1:(length(unique(NCI60.class$class)) - 1),
    adjust = "BH", number = Inf)
```

## In the case where more than 400 genes are significantly differentially expressed at 0.05 level only the top 400 are used

```r
probesets.NCI60 <- NCI60lmFit$ID[NCI60lmFit$P.Val <= pval]
probesets.NCI60 <- probesets.NCI60[1:min(length(probesets.NCI60), 400)]
NCI601.filtered <-
    looNCI60.filtered[probesets.NCI60,]
fitLDANCI60 <-
    sda(t(exprs(NCI601.filtered)), NCI60.class$class[ - cell])
```

## Internal prediction of the cell lines

```r
predi <-
    predict(fitLDANCI60,
        t(exprs(NCI60.random[probesets.NCI60, cell])))
```

## Correctly classified

```r
cell.res$random[cell] <-
    as.numeric(predi$class == NCI60.class$class[cell])
```

```
} 
cat("nsFilter analysis")
for(cell in 1:n.NCI60) {
    cat(cell)
    ## Leave-one-out from arrays
    looNCI60.filtered <- NCI60.filtered[, -cell]
    ## Leave-one-out from factor levels
```
factor.NCI60 <- factor(as.character(NCI60.class$class[ - cell]))
design.NCI60 <- model.matrix(~ 0 + factor.NCI60)
colnames(design.NCI60) <- levels(factor.NCI60)

contrast.matrixNCI60 <-
    makeContrasts(Resistant - Sensitive,
                  Intermediate - Sensitive,
                  levels = design.NCI60)

## Linear model fit
fitNCI60 <- lmFit(looNCI60.filtered,
                  design = design.NCI60)
fit2NCI60 <- contrasts.fit(fitNCI60, contrast.matrixNCI60)
fit2NCI60 <- eBayes(fit2NCI60)

NCI60lmFit <- topTable(fit2NCI60,
                        coef = 1:(length(
                                     unique(NCI60.class$class)) - 1),
                        adjust = "BH", number = Inf)

## In the case where more than 400 genes are significantly differentially expressed at 0.05 level only the top 400 are used
probesets.NCI60 <- NCI60lmFit$ID[NCI60lmFit$P.Val <= pval]
probesets.NCI60 <- probesets.NCI60[1:min(length(probesets.NCI60), 400)]

NCI601.filtered <-
    looNCI60.filtered[probesets.NCI60,]

fitLDANCI60 <-
    sda(t(exprs(NCI601.filtered)), NCI60.class$class[ - cell])

## Internal prediction of the cell lines
predi <-
    predict(fitLDANCI60,
            t(exprs(NCI601.filtered[probesets.NCI60, cell])))

## Correctly classified

22
cell.res$nsFilter[cell] <-
   as.numeric(predi$class == NCI60.class$class[cell])
}
cat("var.cutoff =", cut.off)
cat("mean accuracy =", mean(cell.res$nsFilter))
NCI60LDAcrossval.res[NCI60LDAcrossval.res$cut-offs ==
cut.off, ]$accuracy <- mean(cell.res$nsFilter)
NCI60LDAcrossval.res[NCI60LDAcrossval.res$cut-offs ==
cut.off, ]$acc.rand <- mean(cell.res$rand)
}
save(NCI60LDAcrossval.res, file = NCI60LDAcrossval.res.file)
}

The combination of parameters resulting in the maximum accuracy is stored in the object
accuracy.max,
> accuracy.max <-
   NCI60LDAcrossval.res[NCI60LDAcrossval.res$accuracy ==
   max(NCI60LDAcrossval.res$accuracy), ]
and plotted in Figure 5.2.

> pdf(file.path(figure.output,"NCI60LDAcomparison.pdf"))
> plot(NCI60LDAcrossval.res$cut-offs,
   NCI60LDAcrossval.res$accuracy,
   ylim = c(min(NCI60LDAcrossval.res$accuracy,
   NCI60LDAcrossval.res$acc.rand),
   max(NCI60LDAcrossval.res$accuracy,
   NCI60LDAcrossval.res$acc.rand)),
   ylab = "Predicted Accuracy",
   xlab = "Fraction filtered out",
   type = "n",
   main = "NCI60 Predicted Accuracy")
> lines(NCI60LDAcrossval.res$cut-offs,
   NCI60LDAcrossval.res$accuracy,
   lty = 1, col = "black")
> lines(NCI60LDAcrossval.res$cut-offs,
   NCI60LDAcrossval.res$acc.rand,
   lty = 2, col = "black")
> legend("bottomleft", lty = c(1,2,1), lwd = c(1,1,2),
   legend = c("nsFiltered probes",
   "Random probes"),
   col = "black", bty = "n")
> dev.off()
Figure 5.2: CV accuracy for the LDA analysis at various values of the parameter var.cutoff in nsFilter. The maximum accuracy 0.542 is achieved when var.cutoff is equal to 0.38.

5.3 Establishing the LDA Classifier

The value of the parameter var.cutoff resulting in the maximum accuracy is used on the entire data set in the function nsFilter

```r
> cut.off <- accuracy.max$max$cut.offs[length(accuracy.max$max$cut.offs)]
> NCI60.filtered <- nsFilter(GEPNCI60[, rownames(NCI60.class)], var.cutoff = cut.off)$eset
```

resulting in an expression set with 59 samples and 7861 features. The gene expressions are scaled to have unit variance and zero mean.

```r
> exprs(NCI60.filtered) <- t(scale(t(exprs(NCI60.filtered))))
```

The design matrix used in lmFit to extract the gene expression profiles which are significantly differentially expressed between the three groups is established.

```r
> factor.NCI60 <- factor(NCI60.class$class)
> design.NCI60 <- model.matrix(~ 0 + factor.NCI60)
> colnames(design.NCI60) <- levels(factor.NCI60)
```
A matrix defining the set of contrasts is established through the function `makeContrasts` from the `limma` package.

```r
> contrast.matrixNCI60 <-
makeContrasts(Resistant - Sensitive, 
             Intermediate - Sensitive, 
             levels = design.NCI60)
```

Finally, a linear model is fitted.

```r
> fitNCI60 <- lmFit(NCI60.filtered,
                 design = design.NCI60)
```

The coefficients and standard errors are estimated according to the constructed set of contrasts.

```r
> fit2NCI60 <- contrasts.fit(fitNCI60, contrast.matrixNCI60)
```

Using the functions `eBayes` and `topTable` the gene expression profiles are ranked according to the FDR for differential expressions between the three categories of resistance to melphalan.

```r
> fit2NCI60 <- eBayes(fit2NCI60)
> NCI60lmFit <- topTable(fit2NCI60,
                    coef = 1:(length(unique(NCI60.class$class)) - 1),
                    adjust = "BH",
                    number = Inf)
```

The gene expressions which are expressed significantly different with a significance level of 0.05 are selected for further analysis. If the number exceeds 400 only the top 400 are used.

```r
> probesets.NCI60 <- NCI60lmFit$ID[NCI60lmFit$P.Val <= pval]
> probesets.NCI60 <- probesets.NCI60[1:min(length(probesets.NCI60), 400)]
> p.NCI60.LDA <- length(probesets.NCI60)
```

This results in a 400 gene expression profile. The gene symbols are looked up and stored together with the `topTable` result.

```r
> NCI60LDAgenes <-
        as.vector(unlist(lookUp(probesets.NCI60, "hgu133plus2", "SYMBOL")))
> probesets.LDA.NCI60 <- matrix(NA, nrow = p.NCI60.LDA, ncol = 5)
> row.names(probesets.LDA.NCI60) <- probesets.NCI60
> probesets.LDA.NCI60[,2:5] <-
        signif(as.matrix(NCI60lmFit[1:p.NCI60.LDA,4:7]), 4)
> colnames(probesets.LDA.NCI60) <- c("Symbol", colnames(NCI60lmFit[,4:7]))
> probesets.LDA.NCI60[,1] <- NCI60LDAgenes
```

The first 40 genes are shown in Table 5.2. Finally, the LDA classifier is built using the function `sda` from the package `sda`. 

25
> fitLDANCI60 <-
    sda(t(exprs(NCI60.filtered[probesets.NCI60,])), NCI60.class$class)
Table 5.1: A summary of the resistance levels and defined resistance classes for each cell line.

<table>
<thead>
<tr>
<th>Cell line</th>
<th>NCI60 Resistance index</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>-5.77</td>
<td>Sensitive</td>
</tr>
<tr>
<td>HL.60</td>
<td>-5.61</td>
<td>Sensitive</td>
</tr>
<tr>
<td>MOLT.4</td>
<td>-5.56</td>
<td>Sensitive</td>
</tr>
<tr>
<td>CCRF.CEM</td>
<td>-5.55</td>
<td>Sensitive</td>
</tr>
<tr>
<td>NCL.H322M</td>
<td>-5.13</td>
<td>Sensitive</td>
</tr>
<tr>
<td>ACHN</td>
<td>-5.05</td>
<td>Sensitive</td>
</tr>
<tr>
<td>CAKI.1</td>
<td>-4.97</td>
<td>Sensitive</td>
</tr>
<tr>
<td>MCF7</td>
<td>-4.94</td>
<td>Sensitive</td>
</tr>
<tr>
<td>UACC.62</td>
<td>-4.94</td>
<td>Sensitive</td>
</tr>
<tr>
<td>T47D</td>
<td>-4.85</td>
<td>Sensitive</td>
</tr>
<tr>
<td>SN12C</td>
<td>-4.83</td>
<td>Sensitive</td>
</tr>
<tr>
<td>HOP.62</td>
<td>-4.79</td>
<td>Sensitive</td>
</tr>
<tr>
<td>SF.539</td>
<td>-4.77</td>
<td>Sensitive</td>
</tr>
<tr>
<td>786.0</td>
<td>-4.76</td>
<td>Sensitive</td>
</tr>
<tr>
<td>SF.268</td>
<td>-4.70</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SF.295</td>
<td>-4.70</td>
<td>Intermediate</td>
</tr>
<tr>
<td>LOXIMVI</td>
<td>-4.70</td>
<td>Intermediate</td>
</tr>
<tr>
<td>NCL.H226</td>
<td>-4.64</td>
<td>Intermediate</td>
</tr>
<tr>
<td>MALME.3M</td>
<td>-4.63</td>
<td>Intermediate</td>
</tr>
<tr>
<td>M14</td>
<td>-4.61</td>
<td>Intermediate</td>
</tr>
<tr>
<td>NCL.H460</td>
<td>-4.60</td>
<td>Intermediate</td>
</tr>
<tr>
<td>U251</td>
<td>-4.57</td>
<td>Intermediate</td>
</tr>
<tr>
<td>RXF.393</td>
<td>-4.54</td>
<td>Intermediate</td>
</tr>
<tr>
<td>A549</td>
<td>-4.54</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SK.OV.3</td>
<td>-4.49</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SW.620</td>
<td>-4.47</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SNB.75</td>
<td>-4.46</td>
<td>Intermediate</td>
</tr>
<tr>
<td>OVCAR.8</td>
<td>-4.45</td>
<td>Intermediate</td>
</tr>
<tr>
<td>HOP.92</td>
<td>-4.44</td>
<td>Intermediate</td>
</tr>
<tr>
<td>PC.3</td>
<td>-4.43</td>
<td>Intermediate</td>
</tr>
<tr>
<td>BT.549</td>
<td>-4.42</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SK.MEL.5</td>
<td>-4.40</td>
<td>Intermediate</td>
</tr>
<tr>
<td>HCT.15</td>
<td>-4.40</td>
<td>Intermediate</td>
</tr>
<tr>
<td>OVCAR.4</td>
<td>-4.39</td>
<td>Intermediate</td>
</tr>
<tr>
<td>IGROV1</td>
<td>-4.38</td>
<td>Intermediate</td>
</tr>
<tr>
<td>NCL.H522</td>
<td>-4.38</td>
<td>Intermediate</td>
</tr>
<tr>
<td>MDA.MB.435</td>
<td>-4.38</td>
<td>Intermediate</td>
</tr>
<tr>
<td>OVCAR.3</td>
<td>-4.38</td>
<td>Intermediate</td>
</tr>
<tr>
<td>HCT.116</td>
<td>-4.38</td>
<td>Intermediate</td>
</tr>
<tr>
<td>HCC.2998</td>
<td>-4.37</td>
<td>Intermediate</td>
</tr>
</tbody>
</table>
Table 5.1: (continued)

<table>
<thead>
<tr>
<th>Cell line</th>
<th>NCI60Resistance index</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>UACC.257</td>
<td>−4.37</td>
<td>Intermediate</td>
</tr>
<tr>
<td>COLO205</td>
<td>−4.37</td>
<td>Intermediate</td>
</tr>
<tr>
<td>EKVX</td>
<td>−4.36</td>
<td>Intermediate</td>
</tr>
<tr>
<td>RPMI.8226</td>
<td>−4.36</td>
<td>Intermediate</td>
</tr>
<tr>
<td>MDA.N</td>
<td>−4.33</td>
<td>Intermediate</td>
</tr>
<tr>
<td>DU.145</td>
<td>−4.33</td>
<td>Resistant</td>
</tr>
<tr>
<td>MDA.MB.231</td>
<td>−4.31</td>
<td>Resistant</td>
</tr>
<tr>
<td>UO.31</td>
<td>−4.31</td>
<td>Resistant</td>
</tr>
<tr>
<td>OVCAR.5</td>
<td>−4.31</td>
<td>Resistant</td>
</tr>
<tr>
<td>NCI.ADR.RES</td>
<td>−4.31</td>
<td>Resistant</td>
</tr>
<tr>
<td>K.562</td>
<td>−4.30</td>
<td>Resistant</td>
</tr>
<tr>
<td>HS578T</td>
<td>−4.29</td>
<td>Resistant</td>
</tr>
<tr>
<td>SNB.19</td>
<td>−4.29</td>
<td>Resistant</td>
</tr>
<tr>
<td>SK.MEL.28</td>
<td>−4.24</td>
<td>Resistant</td>
</tr>
<tr>
<td>SK.MEL.2</td>
<td>−4.24</td>
<td>Resistant</td>
</tr>
<tr>
<td>TK.10</td>
<td>−4.17</td>
<td>Resistant</td>
</tr>
<tr>
<td>KM12</td>
<td>−4.14</td>
<td>Resistant</td>
</tr>
<tr>
<td>HT29</td>
<td>−4.12</td>
<td>Resistant</td>
</tr>
<tr>
<td>A498</td>
<td>−3.99</td>
<td>Resistant</td>
</tr>
</tbody>
</table>

Table 5.2: A top table for the first 40 gene expressions used in the LDA classification.

<table>
<thead>
<tr>
<th>Probesets</th>
<th>Symbol</th>
<th>F</th>
<th>P.Value</th>
<th>adj.P.Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>214736_s_at</td>
<td>ADD1</td>
<td>7.888</td>
<td>0.000375</td>
<td>0.8467</td>
</tr>
<tr>
<td>213160_at</td>
<td>DOCK2</td>
<td>7.693</td>
<td>0.0004561</td>
<td>0.8467</td>
</tr>
<tr>
<td>209604_s_at</td>
<td>GATA3</td>
<td>7.65</td>
<td>0.000476</td>
<td>0.8467</td>
</tr>
<tr>
<td>209215_at</td>
<td>MFSD10</td>
<td>7.618</td>
<td>0.0004914</td>
<td>0.8467</td>
</tr>
<tr>
<td>218308_at</td>
<td>TACC3</td>
<td>7.421</td>
<td>0.0005986</td>
<td>0.8467</td>
</tr>
<tr>
<td>208132_s_at</td>
<td>BAT2</td>
<td>7.298</td>
<td>0.0006766</td>
<td>0.8467</td>
</tr>
<tr>
<td>201443_s_at</td>
<td>ATP6AP2</td>
<td>6.823</td>
<td>0.001089</td>
<td>0.8467</td>
</tr>
<tr>
<td>209052_s_at</td>
<td>WHSC1</td>
<td>6.786</td>
<td>0.001129</td>
<td>0.8467</td>
</tr>
<tr>
<td>203600_s_at</td>
<td>FAM193A</td>
<td>6.602</td>
<td>0.001358</td>
<td>0.8467</td>
</tr>
<tr>
<td>203054_s_at</td>
<td>TCTA</td>
<td>6.454</td>
<td>0.001574</td>
<td>0.8467</td>
</tr>
<tr>
<td>219401_at</td>
<td>XYLT2</td>
<td>6.409</td>
<td>0.001646</td>
<td>0.8467</td>
</tr>
<tr>
<td>43544_at</td>
<td>MED16</td>
<td>6.377</td>
<td>0.0017</td>
<td>0.8467</td>
</tr>
<tr>
<td>212696_s_at</td>
<td>RNF4</td>
<td>6.331</td>
<td>0.00178</td>
<td>0.8467</td>
</tr>
<tr>
<td>211594_s_at</td>
<td>MRPL9</td>
<td>6.273</td>
<td>0.001887</td>
<td>0.8467</td>
</tr>
<tr>
<td>208764_s_at</td>
<td>ATP5G2</td>
<td>6.266</td>
<td>0.001901</td>
<td>0.8467</td>
</tr>
<tr>
<td>220081_s_at</td>
<td>HSD17B7</td>
<td>6.211</td>
<td>0.002007</td>
<td>0.8467</td>
</tr>
</tbody>
</table>

28
Table 5.2: (continued)

<table>
<thead>
<tr>
<th>Probesets</th>
<th>Symbol</th>
<th>F</th>
<th>P.Value</th>
<th>adj.P.Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>213416_at</td>
<td>ITGA4</td>
<td>6.138</td>
<td>0.002158</td>
<td>0.8467</td>
</tr>
<tr>
<td>202764_at</td>
<td>STIM1</td>
<td>6.097</td>
<td>0.00225</td>
<td>0.8467</td>
</tr>
<tr>
<td>212546_s_at</td>
<td>FRYL</td>
<td>5.937</td>
<td>0.002639</td>
<td>0.8467</td>
</tr>
<tr>
<td>203997_at</td>
<td>PTPN3</td>
<td>5.932</td>
<td>0.002654</td>
<td>0.8467</td>
</tr>
<tr>
<td>202771_at</td>
<td>FAM38A</td>
<td>5.858</td>
<td>0.002857</td>
<td>0.8467</td>
</tr>
<tr>
<td>34260_at</td>
<td>TELO2</td>
<td>5.831</td>
<td>0.002935</td>
<td>0.8467</td>
</tr>
<tr>
<td>204960_at</td>
<td>PTPRCAP</td>
<td>5.81</td>
<td>0.002996</td>
<td>0.8467</td>
</tr>
<tr>
<td>201923_at</td>
<td>PRDX4</td>
<td>5.72</td>
<td>0.003278</td>
<td>0.8467</td>
</tr>
<tr>
<td>218884_s_at</td>
<td>GUF1</td>
<td>5.631</td>
<td>0.003586</td>
<td>0.8467</td>
</tr>
<tr>
<td>205349_at</td>
<td>GNA15</td>
<td>5.589</td>
<td>0.003739</td>
<td>0.8467</td>
</tr>
<tr>
<td>220597_s_at</td>
<td>ARL6IP4</td>
<td>5.573</td>
<td>0.003799</td>
<td>0.8467</td>
</tr>
<tr>
<td>209234_at</td>
<td>KIF1B</td>
<td>5.552</td>
<td>0.003879</td>
<td>0.8467</td>
</tr>
<tr>
<td>202110_at</td>
<td>COX7B</td>
<td>5.493</td>
<td>0.004114</td>
<td>0.8467</td>
</tr>
<tr>
<td>204374_s_at</td>
<td>GALK1</td>
<td>5.491</td>
<td>0.004123</td>
<td>0.8467</td>
</tr>
<tr>
<td>213468_at</td>
<td>ERCC2</td>
<td>5.49</td>
<td>0.004129</td>
<td>0.8467</td>
</tr>
<tr>
<td>202014_at</td>
<td>PPP1R15A</td>
<td>5.482</td>
<td>0.004116</td>
<td>0.8467</td>
</tr>
<tr>
<td>201000_at</td>
<td>AARS</td>
<td>5.467</td>
<td>0.004224</td>
<td>0.8467</td>
</tr>
<tr>
<td>202496_at</td>
<td>EDC4</td>
<td>5.426</td>
<td>0.004403</td>
<td>0.8467</td>
</tr>
<tr>
<td>221620_s_at</td>
<td>APOO</td>
<td>5.367</td>
<td>0.004667</td>
<td>0.8467</td>
</tr>
<tr>
<td>222062_at</td>
<td>IL27RA</td>
<td>5.354</td>
<td>0.004729</td>
<td>0.8467</td>
</tr>
<tr>
<td>208658_at</td>
<td>PDIA4</td>
<td>5.345</td>
<td>0.00477</td>
<td>0.8467</td>
</tr>
<tr>
<td>212114_at</td>
<td>ATXN7L3B</td>
<td>5.332</td>
<td>0.004835</td>
<td>0.8467</td>
</tr>
<tr>
<td>218438_s_at</td>
<td>MED28</td>
<td>5.213</td>
<td>0.005443</td>
<td>0.8467</td>
</tr>
<tr>
<td>218600_at</td>
<td>LIMD2</td>
<td>5.18</td>
<td>0.005631</td>
<td>0.8467</td>
</tr>
</tbody>
</table>
6 Testing the NCI60 LDA Classifier on the Arkansas Data

6.1 Predicting Melphalan Resistance Classes

First, the expression data are extracted from the expression set.

```r
> Arkansas.matrix <- exprs(GEPArkansas)
```

Next, the expression set are ordered according to the LDA classifier,

```r
> sortNCI60Arkansas <- Arkansas.matrix[probesets.NCI60, ]
```

and scaled so that each gene expression profile across the patients have unit variance and zero mean.

```r
> sortNCI60Arkansas <- t(scale(t(sortNCI60Arkansas)))
```

Finally, the resistance classes are predicted using the function `predict.sda` from the package `sda`

```r
> predictDLDAArkansas <- predict(fitLDANCI60, t(sortNCI60Arkansas))
```

The predicted classes are summarized in Table 6.1.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate</td>
<td>309</td>
</tr>
<tr>
<td>Resistant</td>
<td>129</td>
</tr>
<tr>
<td>Sensitive</td>
<td>121</td>
</tr>
</tbody>
</table>

**Table 6.1:** Summary of the class predictions made by the NCI60 LDA classifier.

6.2 Association between the Resistance Classes and OS

In the following code chunk Kaplan-Meier survival curves are created for the resistance classes. The result is shown in Figure 6.1.

```r
> pdf(file.path(figure.output, "LDANCI60ArkansasKMplotOS.pdf"))
> par(mfrow = c(1, 1))
> metadataArkansas$DLDAThreshold <- predictDLDAArkansas$class
> p.NCI60.LDA.OS <- PlotKM.sda(predictDLDAArkansas$class, metadataArkansas$OS, ylab = "Overall Survival")
```
Figure 6.1: Kaplan-Meier survival curves based on the predicted NCI60 LDA classes. The logrank test comparing the survival curves results in a P-value of 0.05.

6.3 Association between the Resistance Classes and EFS

In the following code chunk Kaplan-Meier survival curves are created for the predicted resistance classes. The result is shown in Figure 6.2.

```r
> pdf(file.path(figure.output,"LDANCI60ArkansasKMplot.pdf"))
> par(mfrow = c(1, 1))
> metadataArkansas$DLDAThreshold <- predictDLDAArkansas$class
> p.NCI60.LDA.EFS <- PlotKM.sda(predictDLDAArkansas$class, metadataArkansas$EFS,
  ylab = "Event Free Survival",
  col = our.colscheme[order(names(our.colscheme))],
  xlab = "Time (months)",
  legend = names(our.colscheme),
  col.leg = our.colscheme,
  main = paste("Arkansas", "\n",
  "NCI60 Kaplan-Meier OS Curves",sep=""))
> dev.off()
```
Figure 6.2: Kaplan-Meier survival curves based on the predicted NCI60 LDA classes. The logrank test comparing the survival curves results in a P-value of 0.3.
7 Establishment of the \textit{NCI60} SPLS Resistance Index

7.1 Cross-Validation

When the output variable is one dimensional, SPLS has two tuning parameters: the number of hidden components $K$ and the shrinkage parameter $\eta$. Below, the two tuning parameters are determined through CV for a wide range of values for the parameter \texttt{var.cutoff} in the function \texttt{nsFilter}. The combination of $K$, $\eta$ and \texttt{var.cutoff} which results in the minimum Mean Square Prediction Error (MSPE) is chosen for further analysis.

By selecting the expression profiles at random instead of using \texttt{nsFilter} it is possible to investigate whether the function choose expression profiles that perform better than random noise.

In order to avoid gene expression profiles which are not marginally related to melphalan resistance sure independence screening is used. Only gene expressions which are significantly correlated at the significance level specified below are used in the analysis.

\begin{verbatim}
> pval <- 0.05
The P-values are not adjusted.
> adjust <- "none"
> set.seed(1000)
> NCI60.SPLS.crossval.res.file <-
   file.path(NCI60.gen.dir, "NCI60SPLScrossval.res.Rdata")
> if(file.exists(NCI60.SPLS.crossval.res.file)){
   load(NCI60.SPLS.crossval.res.file)
} else {
  cutoffs <- seq(0.1, 0.98, by = 0.01)
  NCI60.SPLS.crossval.res <-
     data.frame(cutoffs = cutoffs,
                eta = 0, K=0,
                mspe = 0, rand = 0)
  for(cutoff in cutoffs){
    NCI60.filtered <- nsFilter(GEPNCI60133a,
                var.cutoff = cutoff)$eset
    n.feat <- length(featureNames(NCI60.filtered))
  }
\end{verbatim}
rand.probes <- sample(length(featureNames(GEPNCI60133a)), n.feat)

NCI60.random <- exprs(GEPNCI60133a[rand.probes, ])

eta <- seq(0.2, 0.99, length.out = 200)

fit.spls.cv <-
  cv.spls(x = t(NCI60.random),
           y = NCI60Resistanceindex,
           fold = length(NCI60Resistanceindex),
           eta = eta,
           K = c(1, 2, 3),
           plot.it = FALSE,
           do.SIS = TRUE,
           pval = pval,
           adjust = adjust)

NCI60.SPLS.crossval.res[NCI60.SPLS.crossval.res$cutoffs ==
    cutoff, ]$rand <- min(fit.spls.cv$mspe)

NCI60GEP <- exprs(NCI60.filtered)

fit.spls.cv <-
  cv.spls(x = t(NCI60GEP),
           y = NCI60Resistanceindex,
           fold = length(NCI60Resistanceindex),
           eta = eta,
           K = c(1, 2, 3),
           plot.it = FALSE,
           do.SIS = TRUE,
           pval = pval,
           adjust = adjust)

NCI60.SPLS.crossval.res[NCI60.SPLS.crossval.res$cutoffs ==
    cutoff, ]$eta <- fit.spls.cv$eta.opt
NCI60.SPLS.crossval.res[NCI60.SPLS.crossval.res$cutoffs ==
    cutoff, ]$K <- fit.spls.cv$K.opt
NCI60.SPLS.crossval.res[NCI60.SPLS.crossval.res$cutoffs ==
    cutoff, ]$mspe <- min(fit.spls.cv$mspe)

} save(NCI60.SPLS.crossval.res, file = NCI60.SPLS.crossval.res.file)

The result of the CV routine is extracted by the code chunk below and the optimal values
for these three parameters are shown in Table 7.1.

```r
mspe.min <- NCI60.SPLS.crossval.res[NCI60.SPLS.crossval.res$mspe ==
  min(NCI60.SPLS.crossval.res$mspe), ]
n.mspe <- dim(mspe.min)[1]
cutoff <- mspe.min$cutoffs[n.mspe]
K <- mspe.min$K[n.mspe]
etan <- mspe.min$eta[n.mspe]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optimal value by CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>cutoff</td>
<td>0.6600</td>
</tr>
<tr>
<td>η</td>
<td>0.9265</td>
</tr>
<tr>
<td>K</td>
<td>3.0000</td>
</tr>
</tbody>
</table>

Table 7.1: Values minimizing the MSPE.

The code chunk below constructs a plot of the minimum MSPE achieved through each of the tested values of `var.cutoff` together with the minimum MSPE achieved through the randomly selected expression profiles.

```r
pdf(file.path(figure.output, "NCI60MSPEcomparison.pdf"))
plot(NCI60.SPLS.crossval.res$cutoffs,
     NCI60.SPLS.crossval.res$mspe,
     ylim = c(min(NCI60.SPLS.crossval.res$mspe,
                 NCI60.SPLS.crossval.res$rand),
              max(NCI60.SPLS.crossval.res$mspe,
                  NCI60.SPLS.crossval.res$rand)),
     ylab = "MSPE", xlab = "Fraction filtered out", type = "n",
     main = "NCI60 MSPE")
lines(NCI60.SPLS.crossval.res$cutoffs,
      NCI60.SPLS.crossval.res$mspe, lty = 1)
lines(NCI60.SPLS.crossval.res$cutoffs,
      NCI60.SPLS.crossval.res$rand, lty = 2)
legend("topleft", lty = c(1, 2),
       legend = c("nsFilter", "Random"),
       bty = "n")
dev.off()
```

The plot is shown in Figure 7.1. Setting `var.cutoff` equal to 0.66 results in the smallest minimum MSPE.
Figure 7.1: The minimum MSPE achieved through CV on K and η in SPLS for a variety of values for the var.cutoff. The smallest minimum MSPE is obtained with var.cutoff equal to 0.66

When setting var.cutoff equal to 0.66 the resulting CV on K and η is shown in Figure 7.2. In order to establish the plot filtering of the expression data with the parameter var.cutoff set equal to 0.66 is performed.

```r
> NCI60.filtered <- nsFilter(GEPNCI60133a, var.cutoff = cutoff)$eset
> GEPNCI60 <- exprs(NCI60.filtered)
```

When using the settings in nsFilter the 4311 most varying mRNA expression profiles are used for further analysis.

```r
> NCI60MSPE.file <- file.path(figure.output, "NCI60MSPE.pdf")
> if(!file.exists(NCI60MSPE.file)){
    fit.spls.cv <-
        cv.spls(x = t(GEPNCI60),
        y = NCI60Resistanceindex,
```
fold = n.NCI60,
etta = seq(0.5, 0.99, length.out = 100),
K = c(1, 2, 3),
plot.it = FALSE,
do.SIS = TRUE,
pval = pval,
adjust = adjust)

pdf(file.path(figure.output, "NCI60MSPE.pdf"),
width = 7, height = 7)

trace.mspe(fit.spls.cv, header = "NCI60 MSPE")

dev.off()

Figure 7.2: MSPE for the SPLS regression with var.cutoff in nsFilter set equal to 0.66. This leads to the selection of K = 3 hidden components and a sparsity parameter \( \eta = 0.9265 \).

In order to investigate the accuracy of the SPLS model with the specific settings the CV predicted resistance index is calculated once more and stored in the object `pred.1`.
> pred.1 <- matrix(NA, nrow = n.NCI60, ncol = 2)
> rownames(pred.1) <- NCI60.names.sorted
> colnames(pred.1) <- c("Measured", "predicted")
> pred.1[, 1] <- NCI60Resistanceindex[NCI60.names.sorted, ]
> for(i in NCI60.names.sorted){
  GEPNCI60.cv <- exprs(NCI60.filtered[, sampleNames(GEPNCI60133a) %w/o% i])
  y <- NCI60Resistanceindex[sampleNames(GEPNCI60133a) %w/o% i, ]
  NCI60.fit.spls.cv <- spls(x = t(GEPNCI60.cv),
                            y = y,
                            K = K,
                            eta = eta,
                            do.SIS = TRUE,
                            pval = pval,
                            adjust = adjust)
  newx <- exprs(GEPNCI60133a)[row.names(GEPNCI60.cv), i]
  pred.1[i, 2] <- as.vector(predict(NCI60.fit.spls.cv,
                                    newx = t(as.matrix(newx))))
}

In Figure 7.3A the predicted melphalan resistance index is plotted against the GI$_{50}$ values. When the parameter var.cutoff in the function nsFilter is determined in this manner the resulting MSPE obtained through CV with SPLS is overoptimistic. In order to determine how well the model performs CV over the chosen parameters including var.cutoff is performed.

> pred.2 <- matrix(NA, nrow = n.NCI60, ncol = 3)
> rownames(pred.2) <- NCI60.names.sorted
> colnames(pred.2) <- c("Measured", "Predicted Naive", "Predicted")
> pred.2[, 1] <- NCI60Resistanceindex[NCI60.names.sorted, ]
> pred.2[, 2] <- pred.1[, 2]
> for(i in NCI60.names){
  NCI60.filtered.cv <-
    nsFilter(GEPNCI60133a[, sampleNames(GEPNCI60133a) %w/o% i],
             var.cutoff = cutoff)$eset
  GEPNCI60.cv <- exprs(NCI60.filtered.cv)
  y <- NCI60Resistanceindex[sampleNames(GEPNCI60133a) %w/o% i, ]
}
\[ NCI60.fit.spls.cv \leftarrow \text{spls}(x = \text{t(GEPNCI60.cv)}, \]
\[ y = y, \]
\[ K = K, \]
\[ \text{eta} = \text{eta}, \]
\[ \text{do.SIS} = \text{TRUE}, \]
\[ \text{pval} = \text{pval}, \]
\[ \text{adjust} = \text{adjust} \]
\]
\[ \text{newx} \leftarrow \text{exprs(GEPNCI60133a)[row.names(GEPNCI60.cv), i]} \]
\[ \text{pred.2}[i, 3] \leftarrow \text{as.vector(predict(NCI60.fit.spls.cv,} \]
\[ \text{newx} = \text{t(as.matrix(newx)))} \]
\}

In Figure 7.3B the predicted resistance index is plotted against the measured resistance index. As assumed the result is slightly worse than when the parameter \text{var.cutoff} is not included in the CV step.

\[ \text{Figure 7.3: Predicted resistance index vs. the measured resistance index. The left panel shows CV after filtering. The right panel shows predictions where filtering is performed each time a cell line is left out.} \]

### 7.2 Establishment of the Gene Expression Signature

In the previous section various values were determined through CV. In this section these values are used for fitting a gene expression signature. For this purpose the function \text{spls}
from the package `spls` is used.

The gene expression signature is established and saved for later resistance index predictions.

```r
> NCI60.fit.spls <- spls(x = t(GEPNCI60),
                      y = NCI60Resistanceindex,
                      K = K,
                      eta = eta,
                      do.SIS = TRUE,
                      pval = pval,
                      adjust = adjust)
> save(NCI60.fit.spls,
       file = file.path("Generated data", "NCI60", "NCI60.fit.spls.Rdata"))
```

### 7.3 Investigation of the chosen Gene Expression Profiles

Only 4 mRNA expression profiles are used in the signature. In the code chunk below the gene symbol for these 4 mRNA expression profiles is looked up. Furthermore, in order to see if these mRNA profiles have been observed in other papers studying chemoresistance the following search is conducted in PubMed.

```r
> resTable <- lookUpPubmed(NCI60.fit.spls, n.coef = 3)
```

The results of the search are shown in Table 7.2.

In order to investigate how the identified probes relate to chemoresistance marginally correlation coefficients are calculated and the association between gene expression and resistance index is plotted in Section 7.4.

```r
> NCI60Coef <- coef(NCI60.fit.spls)
> NCI60Coef <- NCI60Coef[NCI60Coef != 0,]
> NCI60Coef <- round(NCI60Coef[order(NCI60Coef)], 4)
> if(length(NCI60Coef) == 1) {
  corm <- t(as.matrix(GEPNCI60[names(NCI60Coef), ]))
  row.names(corm) <- names(NCI60Coef)
```
7.4 Plots of Marginal Associations
8 Testing the NCI60 SPLS Resistance Index on the Arkansas Data Set

8.1 Predicting the Resistance Index in the Arkansas data

Firstly, the expression data are extracted from the expression set.

```r
> arkansas.matrix <- exprs(GEPArkansas)
```

Next, the probes which were removed by nsFilter with var.cutoff set equal to 0.66 are discarded from the matrix.

```r
> NCI60ArkansasTest <- arkansas.matrix[featureNames(NCI60.filtered), ]
```

Finally, the signature is used to predict each patient’s resistance index.

```r
> NCI60Arkansasindex <- as.vector(predict(NCI60.fit.spls, newx = t(NCI60ArkansasTest)))
```

8.2 Association between the Resistance Index and OS

Kaplan-Meier Survival Curves

Kaplan-Meier survival curves are constructed by categorizing the patients as being either sensitive, intermediate or resistant. The 25% with the lowest predicted melphalan resistance index are categorised as sensitive and the 25% with the highest predicted melphalan resistance index are categorised as resistant. The remaining subjects are characterized as having intermediate resistance.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 8.1.

```r
> pdf(file.path(figure.output,"NCI60arkansasOSKMplot.pdf"))
> NCI60arkansasOSKM.P <- PlotKM(NCI60Arkansasindex,metadataArkansas$OS, cut.points = cut.points, xlab = "Time (months)", ylab = "OS ratio", xmax = 110, main = "Arkansas \n NCI60 Kaplan-Meier OS curves"
)
> dev.off()
```
Figure 8.1: Kaplan-Meier survival curves based on the NCI60 resistance index. The logrank test comparing the survival curves results in a P-value of 0.1.

8.2.1 Cox Proportional Hazards

A Cox proportional hazards model is fitted where the association between log relative hazard and the resistance index is modelled by a spline with four knots.

\[
\text{n.knots} <- 4 \\
\text{coxfit.os} <- \text{coxph(metdataArkansas$OS ~ rcs(NCI60Arkansasindex, n.knots))}
\]

Table 8.1 summarizes three tests for no association between the log relative hazard and the resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>3.72</td>
<td>3</td>
</tr>
<tr>
<td>Wald test</td>
<td>3.20</td>
<td>3</td>
</tr>
<tr>
<td>Score (log rank) test</td>
<td>3.23</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 8.1: A summary of three tests for no association between the log relative hazard and the resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the NCI60 resistance index. The result is depicted in Figure 8.2.
Figure 8.2: The log relative hazard as a function of the NCI60 resistance index. The P-value is the likelihood ratio test for no RCS-association between the log relative hazard and the resistance index. The dashed lines represent 95% confidence intervals.

### 8.2.2 Association between the Resistance Index and EFS

**Kaplan-Meier Survival Curves**

Similar to the previous section Kaplan-Meier survival curves are made with the patients grouped to be sensitive, intermediate or resistant, however, EFS is used as endpoint.
The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 8.3.

```r
> pdf(file.path(figure.output,"NCI60arkansasEFSKMplot.pdf"))
> NCI60arkansasEFSKM.P <-
  PlotKM(NCI60Arkansasindex,metadataArkansas$EFS,
  cut.points = cut.points,
  xlab = "Time (months)",
  ylab = "EFS ratio",
  xmax = 110,
  main = "Arkansas \n NCI60 Kaplan-Meier EFS curves")
> dev.off()
```

![Kaplan-Meier survival curves](attachment:NCI60arkansasEFSKMplot.pdf)

**Figure 8.3:** Kaplan-Meier survival curves based on the NCI60 resistance index. The logrank test comparing the survival curves results in a P-value of 0.2.

**Cox Proportional Hazards**

A Cox proportional hazards model is fitted were the association between log relative hazard and the resistance index is modelled by a spline with four knots.

```r
> n.knots <- 4
> coxfit.os <- coxph(metadataArkansas$EFS ~
  rcs(NCI60Arkansasindex, n.knots))
```
Table 8.2 summarises three tests for no association between the log relative hazard and the resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>0.263</td>
<td>3</td>
</tr>
<tr>
<td>Wald test</td>
<td>0.260</td>
<td>3</td>
</tr>
<tr>
<td>Score (log rank) test</td>
<td>0.259</td>
<td>3</td>
</tr>
</tbody>
</table>

The code chunk below produces a plot of the log relative hazard as a function of the NCI60 resistance index. The result is depicted in Figure 8.4.

```r
> pdf(file.path(figure.output, "NCI60arkansasEFS Predictors.pdf"))
> par(mfrow = c(1, 1))
> d <- datadist(NCI60Arkansasindex)
> options(datadist = "d", width = 150)
> NCI60f <- cph(metadataArkansas$EFS ~ rcs(NCI60Arkansasindex, n.knots))
> plot(NCI60f, xlab = "Fitted NCI60 resistance index")
> title("Arkansas \n NCI60 - EFS Cox Proportional Hazards")
> legend("bottomright",
    bty = "n",
    legend = paste("P-value = ",
                 as.character(signif(unlist(NCI60f)$stats.P, 1)), sep = ""))
> dev.off()
```
Figure 8.4: The log relative hazard as a function of the NCI60 resistance index. The P-value is the likelihood ratio test for no RCS-association between the log relative hazard and the resistance index. The dashed lines represent 95% confidence intervals.
9 Establishment of the \textit{BCell} Gene Expression Data

Firstly, various directories are specified.

\begin{verbatim}
> BCell.ext.dir <- paste(getwd(), "/External data/Bcell", sep = "")
> BCell.gen.dir <- paste(getwd(), "/Generated data/BCell/", sep = "")
> figure.output <- paste(getwd(), "/Output/Figures", sep = "")
> table.output <- paste(getwd(), "/Output/Tables", sep = "")
\end{verbatim}

The three output directories are created by the code chunk below.

\begin{verbatim}
> dir.create(path=BCell.gen.dir,
> showWarnings = FALSE, recursive = TRUE, mode = "0777")
> dir.create(path=figure.output,
> showWarnings = FALSE, recursive = TRUE, mode = "0777")
> dir.create(path=table.output,
> showWarnings = FALSE, recursive = TRUE, mode = "0777")
\end{verbatim}

The gene expression arrays used for the analysis are specified in the metadata. This is used to construct a phenoData object.

\begin{verbatim}
> file <- paste(BCell.ext.dir, "/Metadata/metadata_MM.csv", sep = "")
> metadata <- read.csv2(file)
> rownames(metadata) <- metadata$Name
> phenoData <- new("AnnotatedDataFrame", data = metadata)
\end{verbatim}

9.1 RMA Normalisation of the HG-U133 Plus 2.0 Arrays

\begin{verbatim}
> file <- paste(BCell.gen.dir, "GEPBCell.RMA.Rdata", sep = "")
> if(file.exists(file)){
>    load(file)
> }else{
>    fns <-
>        paste(getwd(), "/External data/BCell/Celfiles/",
>               metadata$Filename, ".CEL", sep = "")
>    GEPBCell.RMA <- just.rma(filenames = fns,
>                            phenoData = phenoData)
>    save(GEPBCell.RMA, file=file)
> }
> n.BCell <- dim(GEPBCell.RMA)[2]
> p.BCell <- dim(GEPBCell.RMA)[1]
\end{verbatim}
Write GEPBCell.rma to a space separated .txt file, used for GEO upload.

```r
> write.table(exprs(GEPBCell.RMA),
  file = paste(table.output,"/GEPBCell.RMA", ".txt", sep = ""),
  sep = "\t")
```

This results in an expression set with 54675 features and 18 samples.

**Unsupervised clustering**

The microarrays of the BCell panel were prepared in 5 batches numbered subsequently from 1 to 5. The cell line gene expression profiles were compared using hierarchical cluster analysis with average linkage as the agglomeration method and the Euclidean norm as distance measure. The result is shown in Figure 9.1.

![Cluster Analysis](image)

**Figure 9.1:** Hierarchical clustering of the 18 cell lines. The numbers annotated below each branch represents batch runs of the cell lines.

The four cell lines *OCI-Ly7, DB, HT,* and *SU-DHL-4* originate from patients with DLBCL. The two cell lines *RPMI-8226* and *RPMI-8226 LR5* originate from the same person as do the two cell lines *KMS-12-BM* and *KMS-12-PE*. The cell lines clustered together showing the existence of an intra person correlation. We conclude that no severe batch effects can be detected from the present analysis.
9.2 Restriction to the Affymetrix HG-U133A Subset

```r
> GEPBCell133a <- GEPBCell.RMA[intersect(hgu133aprobe$Probe.Set.Name,
    featureNames(GEPBCell.RMA)), ]

> write.table(exprs(GEPBCell133a),
  file = paste(table.output,"/GEPBCell133a", ".txt", sep = ""),
  sep = "\t")
```
10 Establishment of the BCell Dose Response Data

When a cell line is exposed to melphalan the majority of the cells will be resistant to the drug until the dose reaches a certain threshold. A dose of the drug beyond this threshold causes an abrupt decline in the number of living cells. In this chapter the dose response data for the BCell panel is loaded into R and the GI\textsubscript{50} values are calculated. The plates which have been used for the analysis are specified in the metadata. This is used to construct a phenoData object. All information regarding the 96 well microtiter plates used for the dose response experiments are stored in the following metadata

\begin{verbatim}
> file <- paste(BCell.ext.dir,"/Metadata/",
   "Cell_lines_18_doses_Melphalan_Metadata.csv", sep = "")
> metadata <- read.csv2(file)
\end{verbatim}

A pilot study was conducted in order to give an idea of the necessary span in the doses to ensure that the threshold is included. The 18 doses, denoted C1,...,C18, used to cover this span are shown in Table 10.1. The strongest dose is C18 and C17 is made by halving this dose, similarly C16 is made by halving C17 and so on, until reaching the weakest dose C1. Every time such a halving is made an error is introduced, and to minimise this source of error the doses are split into three series: C18,...,C13, C12,...,C7, and C6,...,C1 where the doses C12 and C6 are made directly from C18. These three series are represented by the three sets of columns in Table 10.1.

<table>
<thead>
<tr>
<th>Label</th>
<th>Dose</th>
<th>Label</th>
<th>Dose</th>
<th>Label</th>
<th>Dose</th>
</tr>
</thead>
<tbody>
<tr>
<td>C18</td>
<td>60.00</td>
<td>C12</td>
<td>0.937500</td>
<td>C6</td>
<td>0.014648</td>
</tr>
<tr>
<td>C17</td>
<td>30.00</td>
<td>C11</td>
<td>0.468750</td>
<td>C5</td>
<td>0.007324</td>
</tr>
<tr>
<td>C16</td>
<td>15.00</td>
<td>C10</td>
<td>0.234375</td>
<td>C4</td>
<td>0.003662</td>
</tr>
<tr>
<td>C15</td>
<td>7.500</td>
<td>C9</td>
<td>0.117188</td>
<td>C3</td>
<td>0.001831</td>
</tr>
<tr>
<td>C14</td>
<td>3.750</td>
<td>C8</td>
<td>0.058594</td>
<td>C2</td>
<td>0.000916</td>
</tr>
<tr>
<td>C13</td>
<td>1.875</td>
<td>C7</td>
<td>0.029297</td>
<td>C1</td>
<td>0.000458</td>
</tr>
</tbody>
</table>

**Table 10.1**: The 18 different doses (in µg/ml) used for each dose response experiment. The strongest and weakest doses are denoted C18 and C1, respectively.

The Plate Setup

In order to measure the response of a certain cell line, a culture plate with 96 wells is used. The standard setup for such a culture plate is depicted in Table 10.2. The wells marked C0,...,C18 contain cell suspension from the examined cell line and the 18 different doses of melphalan. At time t\textsubscript{0} cell suspension is added to wells labelled C0,...,C18 and
medium is added to the first and last column of the plate, labelled M and the wells labelled B.

The different dilutions of melphalan depicted in Table 10.1 are added to the respectable wells 24 hours later, at time $t_1$. At this point salt water is added to wells labelled C0 and B, while nothing is added to wells labelled M.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>M</td>
<td>C0</td>
<td>C2</td>
<td>C4</td>
<td>C6</td>
<td>C8</td>
<td>C10</td>
<td>C12</td>
<td>C14</td>
<td>C16</td>
<td>C18</td>
<td>M</td>
</tr>
<tr>
<td>B</td>
<td>M</td>
<td>C0</td>
<td>C2</td>
<td>C4</td>
<td>C6</td>
<td>C8</td>
<td>C10</td>
<td>C12</td>
<td>C14</td>
<td>C16</td>
<td>C18</td>
<td>M</td>
</tr>
<tr>
<td>C</td>
<td>M</td>
<td>C0</td>
<td>C2</td>
<td>C4</td>
<td>C6</td>
<td>C8</td>
<td>C10</td>
<td>C12</td>
<td>C14</td>
<td>C16</td>
<td>C18</td>
<td>M</td>
</tr>
<tr>
<td>D</td>
<td>M</td>
<td>C0</td>
<td>C2</td>
<td>C4</td>
<td>C6</td>
<td>C8</td>
<td>C10</td>
<td>C12</td>
<td>C14</td>
<td>C16</td>
<td>C18</td>
<td>M</td>
</tr>
<tr>
<td>E</td>
<td>M</td>
<td>C1</td>
<td>C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
<td>C17</td>
<td>B</td>
<td>M</td>
</tr>
<tr>
<td>F</td>
<td>M</td>
<td>C1</td>
<td>C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
<td>C17</td>
<td>B</td>
<td>M</td>
</tr>
<tr>
<td>G</td>
<td>M</td>
<td>C1</td>
<td>C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
<td>C17</td>
<td>B</td>
<td>M</td>
</tr>
<tr>
<td>H</td>
<td>M</td>
<td>C1</td>
<td>C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
<td>C17</td>
<td>B</td>
<td>M</td>
</tr>
</tbody>
</table>

Table 10.2: The standard set-up for the culture plates. The labelling of the wells is interpreted as: Wells labelled M contain medium alone, wells labelled C0 contain cell culture and salt water is added at time $t_1$, wells labelled C1,...,C18 contain cell culture and drug dilutions corresponding to Table 10.1 are added at time $t_1$, and finally wells labelled B contain medium alone and added salt water at time $t_1$.

The plate is harvested 48 hours later, at time $t_2$. When a plate is harvested the measured absorbance for each well containing cell culture is denoted $A_{c,k}$, where $c = 0,\ldots,18$ and $k = 1,\ldots,4$ denotes the concentration of melphalan and replica, respectively. The absorbance measured for the background is denoted $B_k$.

The result of such an experiment is stored in a .DBF file and the following code chunk reads all the .DBF files associated with the analysis.

```r
> chemo.index <- readDBFMeta(file.path(BCell.ext.dir, "Doseresponse"),
               metadata = metadata)
> excluded <- createGIData(chemo.index)[[2]]
```

The measurements in row A and B were removed to avoid any potential border effect. A Grubbs test was applied on every remaining triplicates and outliers were expunged from the dataset. A significance level of 0.01 leads to the exclusion of 34/7200 wells.

### 10.1 Calculation of the Percentage Growth

In order to calculate the GI$_{50}$ values the percentage growth (PG) needs to be calculated first. When calculating the percentage growth the absorbance of the wells C0 is compared with the absorbance of the wells C1,...,C18.

The four wells labelled B in Table 10.2 do not contain any cellular material, and instead of adding melphalan at time $t_1$ salt water is added. Thus, the four wells indicate the plate...
specific background absorbance. We assume that the background absorbance is additive.

Let

\[ \bar{A}_c = \frac{1}{3} \sum_{k=1}^{3} A_{c,k} \quad \text{and} \quad \bar{B} = \frac{1}{3} \sum_{k=1}^{3} B_k \]

denote the average of the measured absorbance for the wells containing cell suspension and background wells, respectively.

We have chosen to calculate the percentage growth at concentration \( c \) as

\[
PG_c = \frac{\bar{A}_c - \bar{B}}{\bar{A}_0 - \bar{B}} \times 100.
\] (10.1)

This is done by the following function.

> GIdata <- createGIData(chemo.index)[[1]]

The results are shown in Figure 10.1 where the replicated runs of the 18 cell lines are plotted in separate panels.

> key.variety <-
list(space = "top",
     text = list(levels(GIdata$platerep)),
     points = list(pch = c(17, 18, 15),
                   col = "black"),
     title = "Replica",
     columns = 3)

> plot <- (xyplot((GI * 100) ~ t|name,
                   data = GIdata,
                   groups = platerep,
                   key = key.variety,
                   lty = 1,
                   pch = c(17, 18, 15),
                   col.line = "black",
                   col.symbol = "black",
                   col.grid = "black",
                   scales = list(x = list(log = 2)),
                   layout = c(3, 6),
                   height = 10,
                   width = 20,
                   aspect = 0.65,
                   type = c("b", "g"),
                   ylab = "Per cent of Control",
                   labels = list(c("Control", "Treatment A", "Treatment B")))

54
xlab = expression(paste("Concentration (", mu,"g/ml")

main = paste("Growth Inhibition for Cell Lines",
"Treated with Melphalan"))

> pdf(file.path(figure.output, "complete_data.pdf"),
       width = 6, height = 8.5)
> print(plot)
> dev.off()
Growth Inhibition for Cell Lines Treated with Melphalan

Figure 10.1: The result of replicated dose response runs of the cell lines are plotted in separate panels.
It seems that the following five points are outliers and are removed prior to the calculation of GI\(_{50}\) values.

```r
> GIdata <- GIdata[!(GIdata$name == "OPM-2" &
>                      GIdata$platerep == 1 &
>                      GIdata$GIname == 16 ), ]
> GIdata <- GIdata[!(GIdata$name == "MM1S" &
>                      GIdata$platerep == 1 &
>                      GIdata$GIname == 7), ]
> GIdata <- GIdata[!(GIdata$name == "MM1S" &
>                      GIdata$platerep == 1 &
>                      GIdata$GIname == 9 ), ]
> GIdata <- GIdata[!(GIdata$name == "MOLP-2" &
>                      GIdata$platerep == 1 &
>                      GIdata$GIname == 7 ), ]
> GIdata <- GIdata[!(GIdata$name == "KMS-12-PE" &
>                      GIdata$platerep == 1 &
>                      GIdata$GIname == 9 ), ]
> GIdata <- GIdata[GIdata$t != 0, ]
> GIdata$name <- as.factor(as.character(GIdata$name))
```

10.2 Calculation of the GI\(_{50}\) values

In order to calculate the GI\(_{50}\) value for each cell line linear interpolation between the 18 PG values are chosen. The first dose at which the interpolation is equal to 50% is used as the GI\(_{50}\) value.

When applying this method the PG values of the replicated dose response experiments are summarized as the mean response at each concentration. Let \(\text{PG}_{i,j,c}\) be the calculated percentage growth for the \(c\)th drug concentration of the \(j\)th replica within the \(i\)th cell line, and

\[
\text{PG}_{i,.c} = \frac{1}{k} \sum_{j=1}^{k} \text{PG}_{i,j,c},
\]

where \(k\) represents the number of replicates for the \(i\)th cell line. In the code chunk below a dataset consisting only of these mean values is created.

```r
> GImean <- aggregate(GIdata$GI,
>                      list(GIdata$name, GIdata$t),
>                      FUN = mean)
> names(GImean) <- c("name", "t", "GI")
> GInames <- levels(GImean$name)
```
In order to compare the results obtained through the BCell panel with those obtained with the NCI60 panel the GI50 values are transformed to the log10 mM scale. This is done by the following calculation:

\[
\log_{10}(\text{mmol/ml}) = \log_{10}\left(\frac{1000 \cdot (\mu g/ml)}{305.2}/1000000\right)
\]

\[\text{(10.2)}\]

```r
> GImean$t.m <- molaer(GImean$t)
```

The averaged transformed percentage growth values are shown in Figure 10.2A for each cell line along with the linear interpolation. The figure is produced by the code chunk below.

```r
> pdf(paste(figure.output, "/BCellDosecurves.pdf", sep = ""),
    width = 10, height = 7)

The settings for the plot are determined.

```r
> matplot(c(min(GImean$t.m), max(GImean$t.m)),
    c(min(GImean$GI * 100), max(GImean$GI) * 100),
    type = "n",
    xlab = expression(paste("Concentration ",
        \log_{10}(\text{m mol/ml})", sep = "")),
    ylab = "Percent of Control", las = 1,
    main = paste("Melphalan Induced Growth Inhibition", sep = ""),
    cex.axis = 0.7)

> # The five most sensitive are coloured black,
> # the eight intermediate are coloured dark grey
> # and the five most resistant are coloured red
> col <- c(rep(our.colscheme[1], 5),
    rep(our.colscheme[2], c(n.BCell - 5 * 2)),
    rep(our.colscheme[3], 5))

> # Six different line types are used
> lty <- c(1:5, 1:c(n.BCell - 5 * 2), 1:5)

> names <- levels(GImean$name)

> # The lines are drawn
> h <- 1

```r
> for(i in names){
    ```r
    lines(GImean$t.m[GImean$name == i],
        GImean$GI [GImean$name == i] * 100,
        col = col[h], lty = lty[h], lwd = 1.4)
    ```r
    h <- h + 1
}

> # A legend, ordered according to sensitivity, is inserted
> legend("bottomleft", names, col = col,
    lty = lty, cex = 0.7, bty = "n")
```
Linear interpolation between the PG values is used to extract the GI\textsubscript{50} values. In the following the first point at which the dose response curve drops below the 50\% per cent level is used as the GI\textsubscript{50} value. The results are summarized in Table 11.1.

```r
> GInames <- levels(GImean$name)
> GIvalues <- rep(0, length(GInames))
> names(GIvalues) <- GInames
> for(j in GInames){
  GIvalues[j] <- findGI(GImean$t.m[GImean$name == j],
                      GImean$GI [GImean$name == j],
                      GI = 0.5)
}
> BCellResistanceindex <- GIvalues
```

When the BCell resistance index is used to establish a gene expression signature the mRNA profiles stored in the expression set GEPBCell1133a are used as input variables. It is therefore necessary to align the objects.

```r
> BCell.names <- sampleNames(GEPBCell1133a)
> BCellResistanceindex <- cbind(BCellResistanceindex[BCell.names])
> save(BCellResistanceindex, file=paste(BCell.gen.dir,
                               "/BCellResistanceindex.Rdata", sep = ""))
> BCell.names.sorted <- names(BCellResistanceindex[order(BCellResistanceindex),])
```

### 10.3 Subsampling with Replacement

In order to assess the uncertainty of the estimates we have chosen to use subsampling with replacement of the replicated wells and recalculation of the GI\textsubscript{50} values in each dataset 200 times is used.

```r
> file <- c(paste(BCell.gen.dir,"/bs.Rdata",sep=""))
> if(file.exists(file)){
    load(file)
}else{

    set.seed(1000)
    bs <- matrix(NA, ncol = n.BCell, nrow = n.subsamples)
    colnames(bs) <- levels(GImean$name)
    for(i in 1:n.subsamples) {
        GIdata <- createGIData(chemo.index, Bootstrap = TRUE)[[1]]
        # The outliers which were removed from the original data are
        # also removed here
```
GIdata <- GIdata[!(GIdata$name == "OPM-2" 
      & GIdata$platerep == 1 
      & GIdata$GIname == 16 ), ]

GIdata <- GIdata[!(GIdata$name == "MM1S" 
      & GIdata$platerep == 1 
      & GIdata$GIname == 7), ]

GIdata <- GIdata[!(GIdata$name == "MM1S" 
      & GIdata$platerep == 1 
      & GIdata$GIname == 9 ), ]

GIdata <- GIdata[!(GIdata$name == "MOLP-2" 
      & GIdata$platerep == 1 
      & GIdata$GIname == 7 ), ]

GIdata <- GIdata[!(GIdata$name == "KMS-12-PE" 
      & GIdata$platerep == 1 
      & GIdata$GIname == 9 ), ]

#==============================================================

GImean.bs <- aggregate(GIdata$GI,
    list(GIdata$name, GIdata$t),
    FUN=mean)

names(GImean.bs) <- c("name","t","GI")
GImean.bs$t.m <- molaer(GImean.bs$t)
GIvalues <- rep(0,length(GInames))
names(GIvalues) <- GInames

for(j in GInames){
    GIvalues[j] <- findGI(GImean.bs$t.m[GImean.bs$name == j],
        GImean.bs$GI [GImean.bs$name == j],
        GI = 0.5)
}

bs[i,] <- GIvalues
print(i)
}

save(bs, file = file)

Figure 10.2B summarizes the result of the 200 GI50 values for each cell line in a box plot.
Figure 10.2: The average PG values are shown in plot A for each cell line in the BCell panel along with linear interpolation. Plot B shows a box plot of the GI$_{50}$ values obtained for each cell line from each of the 200 bootstrapped datasets through linear interpolation.
11 Establishment of the BCell LDA Classifier

Setting the seed to ensure repeatability of the results.

> set.seed(1000)

The LDA analysis includes all gene expressions which are expressed significantly different between the sensitive, intermediate and resistant cell lines at an significance level of 0.05.

> pval <- 0.05

11.1 Definition of the BCell Resistance Classes

Defining and annotating the sensitive, intermediate and resistant cell lines.

> sens <- 5
> resi <- 5
> inte <- n.BCell - sens - resi
> class.list <- c(rep("Sensitive", sens),
                 rep("Intermediate", inte),
                 rep("Resistant", resi))

A data frame is constructed containing the GI\textsubscript{50} values and the class of each cell line. The data frame is shown in Table 11.1.

> sort.BCell.class <-
  data.frame(BCellResistanceIndex = sort(BCellResistanceIndex),
             class = class.list)
<table>
<thead>
<tr>
<th>Cell line</th>
<th>BCellResistanceindex</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOLP-2</td>
<td>-6.0210</td>
<td>Sensitive</td>
</tr>
<tr>
<td>MOLP-8</td>
<td>-5.5889</td>
<td>Sensitive</td>
</tr>
<tr>
<td>MM1S</td>
<td>-5.2443</td>
<td>Sensitive</td>
</tr>
<tr>
<td>NCI-H929</td>
<td>-5.1836</td>
<td>Sensitive</td>
</tr>
<tr>
<td>OPM-2</td>
<td>-4.9366</td>
<td>Sensitive</td>
</tr>
<tr>
<td>AMO-1</td>
<td>-4.8391</td>
<td>Intermediate</td>
</tr>
<tr>
<td>SU-DHL-4</td>
<td>-4.8111</td>
<td>Intermediate</td>
</tr>
<tr>
<td>U-266</td>
<td>-4.8088</td>
<td>Intermediate</td>
</tr>
<tr>
<td>DB</td>
<td>-4.7802</td>
<td>Intermediate</td>
</tr>
<tr>
<td>KMS-12-BM</td>
<td>-4.7596</td>
<td>Intermediate</td>
</tr>
<tr>
<td>KMM-1</td>
<td>-4.7198</td>
<td>Intermediate</td>
</tr>
<tr>
<td>OCI-Ly7</td>
<td>-4.7053</td>
<td>Intermediate</td>
</tr>
<tr>
<td>HT</td>
<td>-4.6502</td>
<td>Intermediate</td>
</tr>
<tr>
<td>LP-1</td>
<td>-4.5975</td>
<td>Resistant</td>
</tr>
<tr>
<td>KMS-11</td>
<td>-4.5848</td>
<td>Resistant</td>
</tr>
<tr>
<td>KMS-12-PE</td>
<td>-4.5612</td>
<td>Resistant</td>
</tr>
<tr>
<td>RPMI-8226</td>
<td>-4.4932</td>
<td>Resistant</td>
</tr>
<tr>
<td>RPMI-8226 LR5</td>
<td>-4.1343</td>
<td>Resistant</td>
</tr>
</tbody>
</table>

Table 11.1: A summary of the level of resistance for each cell line.

The data are sorted according to the observed GI<sub>50</sub> value.

```r
> BCell.class <- sort.BCell.class[rownames(BCellResistanceindex),]
> BCell.class$class <- as.character(BCell.class$class)
```

The following code chunk constructs a box plot of the GI<sub>50</sub> values grouped into resistant, intermediate and sensitive cell lines.

```r
> pdf(file.path(figure.output, "BCellresvssensBoxplot.pdf"))
> sort.BCell.class$class <- relevel(sort.BCell.class$class, names(our.colscheme)[1])
> boxplot(sort.BCell.class$BCellResistanceindex ~ sort.BCell.class$class, border = our.colscheme)
> dev.off()
```

The plot is shown in Figure 11.1.
11.2 Cross-Validation

In order to choose optimal parameters for the LDA analysis CV is used. In the code chunk below the cutoff value of the unspecific filtering is likewise determined through CV. In this method the parameters of the LDA analysis are determined through CV for a wide range of values of the parameter \texttt{var.cutoff} in the function \texttt{nsFilter}. The combination of parameters in LDA and the \texttt{var.cutoff} that results in the maximum accuracy is chosen for further analysis.

Investigating whether the function \texttt{nsFilter} choose expression profiles that perform better than random noise, randomly selected expression profiles is used instead of \texttt{nsFilter} filter.

The various fractions to be filtered out is determined

```
> cut-offs <- seq(0.01, 0.98, by = 0.01)
```

A data frame containing the CV accuracy for LDA based on the gene expression profiles obtained through \texttt{nsFilter} and random selection is established.

```
> BCellLDAcrossval.res <- data.frame(cut-offs = cut-offs,
```

\textbf{Figure 11.1}: Boxplots of the GI\textsubscript{50} values grouped into the resistant, intermediate and sensitive cell lines.
accuracy = 0,
acc.rand = 0)

> BCellLDAcrossval.res.file <-
    file.path(BCell.gen.dir, "BCellLDAcrossval.res.Rdata")
> if(file.exists(BCellLDAcrossval.res.file)){
    load(BCellLDAcrossval.res.file)
} else{
    for(cut.off in cut.offs){
        print("Cut off:"))
        print(cut.off)

        cell.res <-
            data.frame(nsFilter = rep(0, length(BCell.class$BCellResistanceindex),
                                random = rep(0, length(BCell.class$BCellResistanceindex))))

        ## nsFilter expression
        BCell.filtered <- nsFilter(GEPBCell133a[, rownames(BCell.class)],
                                   var.cutoff = cut.off)$eset

        exprs(BCell.filtered) <-
            t(scale(t(exprs(BCell.filtered))))

        ## Random expressions
        n.feat <- length(featureNames(BCell.filtered))
        rand.probes <- sample(length(featureNames(GEPBCell133a)), n.feat)
        BCell.random <- GEPBCell133a[rand.probes, rownames(BCell.class)]

        exprs(BCell.random) <- t(scale(t(exprs(BCell.random))))

        ## Analysis with random expressions
        for(cell in 1:length(BCell.class$BCellResistanceindex)){
            print(rownames(BCell.class)[cell])
            print(cell)

            ## Leave-one-out from arrays
            looBCell.filtered <-
                BCell.random[, -cell]

            65
## Leave-one-out from factor levels

```r
fBCell <- factor(as.character(BCell.class$class[-cell]))
designBCell <- model.matrix(~ 0 + fBCell)

colnames(designBCell) <- levels(fBCell)

contrast.matrixBCell <-
  makeContrasts(Resistant - Sensitive,
                Intermediate - Sensitive,
                levels = designBCell)

## Linear model fit

fitBCell <- lmFit(looBCell.filtered, design = designBCell)

fit2BCell <- contrasts.fit(fitBCell, contrast.matrixBCell)
fit2BCell <- eBayes(fit2BCell)

BCelllmFit <- topTable(fit2BCell, 
  coef = 1:(length(unique(BCell.class$class)) - 1),
  adjust = "BH", number = Inf)

probesets.BCell <- BCelllmFit$ID[BCelllmFit$P.Val <= pval]

BCell1.filtered <-
  looBCell.filtered[probesets.BCell,]

fitLDABCell <-
  sda(t(exprs(BCell1.filtered)), BCell.class$class[-cell])

## Internal prediction of the cell lines

predi <-
  predict(fitLDABCell, 
    t(exprs(BCell.random[probesets.BCell, cell])))

## Correctly classified

cell.res$random[cell] <-
  as.numeric(predi$class == BCell.class$class[cell])
```

66
## nsFilter analysis

for(cell in 1:length(BCell.class$BCellResistanceindex)){
  print(rownames(BCell.class)[cell])
  print(cell)

  ## Leave-one-out from arrays
  looBCell.filtered <-
    BCell.filtered[, -cell]

  ## Leave-one-out from factor levels
  fBCell <- factor(as.character(BCell.class$class[-cell]))
  designBCell <- model.matrix(~ 0 + fBCell)
  colnames(designBCell) <- levels(fBCell)
  contrast.matrixBCell <-
    makeContrasts(Resistant - Sensitive,
                 Intermediate - Sensitive,
                 levels = designBCell)

  ## Linear model fit
  fitBCell <- lmFit(looBCell.filtered,
                    design = designBCell)
  fit2BCell <- contrasts.fit(fitBCell, contrast.matrixBCell)
  fit2BCell <- eBayes(fit2BCell)
  BCelllmFit <- topTable(fit2BCell,
                         coef = 1:(length(
                                unique(BCell.class$class))-1),
                         adjust = "BH", number = Inf)
  probesets.BCell <- BCelllmFit$ID[BCelllmFit$P.Val <= pval]
  BCell1.filtered <-
    looBCell.filtered[probesets.BCell,]
  fitLDABCell <-

  67
\texttt{sda(t(exprs(BCell1.filtered)), BCell.class$\texttt{class[-cell]})}

\texttt{## Internal prediction of the cell lines}
\texttt{predi <- predict(fitLDABCell,}
\texttt{\hspace{1cm} t(exprs(BCell.filtered[probesets.BCell, cell])))}
\texttt{## Correctly classified}
\texttt{cell.res$\texttt{nsFilter}[cell] <- as.numeric(predi$\texttt{class} == BCell.class$\texttt{class}[cell])}

\texttt{BCellLDAcrossval.res[BCellLDAcrossval.res$cut.off ==}
\texttt{\hspace{1cm} cut.off, ]$\texttt{accuracy} <- mean(cell.res$\texttt{nsFilter})}
\texttt{BCellLDAcrossval.res[BCellLDAcrossval.res$cut.off ==}
\texttt{\hspace{1cm} cut.off, ]$\texttt{acc.rand} <- mean(cell.res$\texttt{rand})}

\texttt{save(BCellLDAcrossval.res, file = BCellLDAcrossval.res.file)}

The combination of the parameters resulting in the maximum accuracy in the CV routine
is stored in the object \texttt{accuracy.max},

\texttt{\textgreater{} accuracy.max <- BCellLDAcrossval.res[BCellLDAcrossval.res$\texttt{accuracy} ==}
\texttt{\hspace{1cm} max(BCellLDAcrossval.res$\texttt{accuracy}), ]}

and plotted

\texttt{\textgreater{} pdf(file.path(figure.output, "BCellLDAcomparison.pdf"))}
\texttt{\textgreater{} plot(BCellLDAcrossval.res$cut.off, BCellLDAcrossval.res$\texttt{accuracy},}
\texttt{\hspace{1cm} ylim = c(min(BCellLDAcrossval.res$\texttt{accuracy},}
\texttt{\hspace{2cm} BCellLDAcrossval.res$\texttt{acc.rand}),}
\texttt{\hspace{2cm} max(BCellLDAcrossval.res$\texttt{accuracy},}
\texttt{\hspace{2cm} BCellLDAcrossval.res$\texttt{acc.rand})),}
\texttt{\hspace{1cm} ylab = "Accuracy",}
\texttt{\hspace{1cm} xlab = "Fraction filtered out", type="n")}
\texttt{\textgreater{} lines(BCellLDAcrossval.res$cut.off, BCellLDAcrossval.res$\texttt{accuracy},}
\texttt{\hspace{1cm} 68}
Figure 11.2: The CV accuracy for the LDA analysis at various values of the parameter var.cutoff in nsFilter. The maximum accuracy 0.611 is achieved when var.cutoff is set equal to 0.95.

11.3 Establishing the LDA Classifier

The value of the parameter var.cutoff resulting in the maximum accuracy is used in the function nsFilter on the entire dataset,

> cut.off <- accuracy.max$cut.offs[length(accuracy.max$cut.offs)]
> BCell.filtered <- nsFilter(GEPBCell133a[, rownames(BCell.class)],
var.cutoff = cut.off)$eset
and the gene expression profiles are scaled to have unit variance and zero mean.

```r
> exprs(BCell.filtered) <-
    t(scale(t(exprs(BCell.filtered))))
```

The design matrix which is used in the function `lmFit` to extract the gene expression profiles which are significantly differentially expressed between the three groups is established.

```r
> fBCell <- factor(BCell.class$class)
> designBCell <- model.matrix(~ 0 + fBCell)
> colnames(designBCell) <- levels(fBCell)
```

A matrix defining the set of contrasts is established through the function `makeContrasts` from the `limma` package.

```r
> contrast.matrixBCell <-
    makeContrasts(Resistant - Sensitive,
        Intermediate - Sensitive,
        levels = designBCell)
```

Finally, a linear model is fitted.

```r
> fitBCell <- lmFit(BCell.filtered, design = designBCell)
```

The coefficients and standard errors are estimated according to the constructed set of contrasts.

```r
> fit2BCell <- contrasts.fit(fitBCell, contrast.matrixBCell)
```

Using the functions `eBayes` and `topTable` the gene expression profiles are ranked according to the adjusted P-values for differential expressions between the three categories of resistance to melphalan.

```r
> fit2BCell <- eBayes(fit2BCell)
> BCelllmFit <- topTable(fit2BCell,
    coef = 1:(length(unique(BCell.class$class)) - 1),
    adjust = "BH",
    number = Inf)
```

The gene expressions expressed significantly different at a significance level of 0.05 are selected for further analysis.

```r
> probesets.BCell <- BCelllmFit$ID[BCelllmFit$P.Val <= pval]
> n.probes <- length(probesets.BCell)
```

This results in 159 gene expressions. The gene symbols are looked up and stored together with the `topTable` result.
> BCellLDAgenes <- as.vector(unlist(lookUp(probesets.BCell, "hgu133plus2", "SYMBOL")))
> probesets.LDA.BCell <- matrix(NA, nrow = n.probes, ncol = 5)
> row.names(probesets.LDA.BCell) <- probesets.BCell
> probesets.LDA.BCell[,2:5] <- signif(as.matrix(BCelllmFit[1:n.probes, 4:7]), 4)
> colnames(probesets.LDA.BCell) <- c("Symbol",colnames(BCelllmFit[, 4:7]))
> probesets.LDA.BCell[,1] <- BCellLDAgenes

The matrix is shown in Table 11.2

Finally, the LDA analysis is conducted using the function sda from the package sda and saved for later predictions.

> fitLDABCell <- sda(t(exprs(BCell.filtered[probesets.BCell,])), BCell.class$class)
> save(fitLDABCell, file = file.path(BCell.gen.dir, "fitLDABCell.Rdata"))

Table 11.2: A summary of the toptable for the 159 gene expressions used in the LDA classifier.

<table>
<thead>
<tr>
<th>Probesets</th>
<th>Symbol</th>
<th>AveExpr</th>
<th>F</th>
<th>P.Value</th>
<th>adj.P.Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>209310_at</td>
<td>CASP4</td>
<td>2.282e-16</td>
<td>6.343</td>
<td>0.001759</td>
<td>0.1572</td>
</tr>
<tr>
<td>221004_at</td>
<td>ITM2C</td>
<td>-2.722e-16</td>
<td>6.245</td>
<td>0.00194</td>
<td>0.1572</td>
</tr>
<tr>
<td>221912_at</td>
<td>CCDC28B</td>
<td>-1.258e-16</td>
<td>6.139</td>
<td>0.002158</td>
<td>0.1572</td>
</tr>
<tr>
<td>209735_at</td>
<td>ABCG2</td>
<td>-8.481e-17</td>
<td>5.973</td>
<td>0.002545</td>
<td>0.1572</td>
</tr>
<tr>
<td>221297_at</td>
<td>GPRC5D</td>
<td>2.564e-17</td>
<td>5.863</td>
<td>0.002843</td>
<td>0.1572</td>
</tr>
<tr>
<td>206060_at</td>
<td>PTPN22</td>
<td>1.627e-16</td>
<td>5.853</td>
<td>0.002872</td>
<td>0.1572</td>
</tr>
<tr>
<td>210145_at</td>
<td>PLA2G4A</td>
<td>-1.388e-17</td>
<td>5.676</td>
<td>0.003427</td>
<td>0.1572</td>
</tr>
<tr>
<td>218330_at</td>
<td>NAV2</td>
<td>-1.496e-16</td>
<td>5.673</td>
<td>0.003438</td>
<td>0.1572</td>
</tr>
<tr>
<td>209568_at</td>
<td>RGL1</td>
<td>-2.619e-16</td>
<td>5.652</td>
<td>0.003512</td>
<td>0.1572</td>
</tr>
<tr>
<td>212543_at</td>
<td>AIM1</td>
<td>1.103e-16</td>
<td>5.51</td>
<td>0.004046</td>
<td>0.1572</td>
</tr>
<tr>
<td>200602_at</td>
<td>APP</td>
<td>5.089e-17</td>
<td>5.246</td>
<td>0.005266</td>
<td>0.1572</td>
</tr>
<tr>
<td>221268_at</td>
<td>SGPP1</td>
<td>1.588e-16</td>
<td>5.236</td>
<td>0.00532</td>
<td>0.1572</td>
</tr>
<tr>
<td>205807_at</td>
<td>TUFT1</td>
<td>-1.82e-16</td>
<td>5.179</td>
<td>0.005635</td>
<td>0.1572</td>
</tr>
<tr>
<td>203758_at</td>
<td>CTSO</td>
<td>6.014e-17</td>
<td>5.131</td>
<td>0.005908</td>
<td>0.1572</td>
</tr>
<tr>
<td>204552_at</td>
<td>INPP4A</td>
<td>2.806e-16</td>
<td>5.117</td>
<td>0.005996</td>
<td>0.1572</td>
</tr>
<tr>
<td>201125_at</td>
<td>ITGB5</td>
<td>-4.715e-16</td>
<td>5.089</td>
<td>0.006163</td>
<td>0.1572</td>
</tr>
<tr>
<td>210942_at</td>
<td>ST3GAL6</td>
<td>-5.204e-17</td>
<td>5.081</td>
<td>0.006216</td>
<td>0.1572</td>
</tr>
<tr>
<td>206121_at</td>
<td>AMPD1</td>
<td>-7.859e-17</td>
<td>5.053</td>
<td>0.006393</td>
<td>0.1572</td>
</tr>
<tr>
<td>222258_at</td>
<td>SH3BP4</td>
<td>-1.218e-16</td>
<td>5.032</td>
<td>0.006529</td>
<td>0.1572</td>
</tr>
<tr>
<td>219572_at</td>
<td>CADPS2</td>
<td>-3.547e-17</td>
<td>4.992</td>
<td>0.006795</td>
<td>0.1572</td>
</tr>
<tr>
<td>211368_at</td>
<td>CASP1</td>
<td>4.163e-17</td>
<td>4.929</td>
<td>0.007237</td>
<td>0.1572</td>
</tr>
<tr>
<td>34210_at</td>
<td>CD52</td>
<td>-5.551e-17</td>
<td>4.916</td>
<td>0.007325</td>
<td>0.1572</td>
</tr>
<tr>
<td>202551_at</td>
<td>CRIM1</td>
<td>-1.349e-16</td>
<td>4.91</td>
<td>0.007374</td>
<td>0.1572</td>
</tr>
<tr>
<td>Probesets</td>
<td>Symbol</td>
<td>AveExpr</td>
<td>F</td>
<td>P.Value</td>
<td>adj.P.Val</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>202947_s_at</td>
<td>GYPC</td>
<td>-1.588e-16</td>
<td>4.866</td>
<td>0.007705</td>
<td>0.1572</td>
</tr>
<tr>
<td>212886_at</td>
<td>CCDC69</td>
<td>3.915e-16</td>
<td>4.857</td>
<td>0.007771</td>
<td>0.1572</td>
</tr>
<tr>
<td>205016_at</td>
<td>TGFA</td>
<td>-6.818e-17</td>
<td>4.85</td>
<td>0.007827</td>
<td>0.1572</td>
</tr>
<tr>
<td>209163_at</td>
<td>CYB561</td>
<td>4.795e-18</td>
<td>4.827</td>
<td>0.008014</td>
<td>0.1572</td>
</tr>
<tr>
<td>219191_s_at</td>
<td>BIN2</td>
<td>1.11e-16</td>
<td>4.81</td>
<td>0.008145</td>
<td>0.1572</td>
</tr>
<tr>
<td>210279_at</td>
<td>GPR18</td>
<td>-1.388e-16</td>
<td>4.801</td>
<td>0.008222</td>
<td>0.1572</td>
</tr>
<tr>
<td>218793_s_at</td>
<td>SCML1</td>
<td>1.295e-16</td>
<td>4.77</td>
<td>0.008479</td>
<td>0.1572</td>
</tr>
<tr>
<td>205110_s_at</td>
<td>FGFI3</td>
<td>2.467e-17</td>
<td>4.755</td>
<td>0.008606</td>
<td>0.1572</td>
</tr>
<tr>
<td>214023_x_at</td>
<td>TUBB2B</td>
<td>-1.068e-16</td>
<td>4.646</td>
<td>0.009601</td>
<td>0.1572</td>
</tr>
<tr>
<td>202609_at</td>
<td>EPS8</td>
<td>-2.66e-17</td>
<td>4.643</td>
<td>0.009632</td>
<td>0.1572</td>
</tr>
<tr>
<td>221526_x_at</td>
<td>PARD3</td>
<td>1.665e-16</td>
<td>4.634</td>
<td>0.009717</td>
<td>0.1572</td>
</tr>
<tr>
<td>221645_s_at</td>
<td>ZNF83</td>
<td>7.922e-17</td>
<td>4.612</td>
<td>0.009929</td>
<td>0.1572</td>
</tr>
<tr>
<td>221704_s_at</td>
<td>VPS37B</td>
<td>-2.321e-16</td>
<td>4.585</td>
<td>0.0102</td>
<td>0.1572</td>
</tr>
<tr>
<td>219221_at</td>
<td>ZBTB38</td>
<td>3.296e-16</td>
<td>4.567</td>
<td>0.01039</td>
<td>0.1572</td>
</tr>
<tr>
<td>212715_s_at</td>
<td>MICAL3</td>
<td>1.759e-16</td>
<td>4.555</td>
<td>0.01051</td>
<td>0.1572</td>
</tr>
<tr>
<td>213415_at</td>
<td>CLIC2</td>
<td>7.71e-17</td>
<td>4.534</td>
<td>0.01073</td>
<td>0.1572</td>
</tr>
<tr>
<td>222317_at</td>
<td>PDE3B</td>
<td>5.474e-17</td>
<td>4.52</td>
<td>0.01088</td>
<td>0.1572</td>
</tr>
<tr>
<td>206698_at</td>
<td>XK</td>
<td>-1.48e-16</td>
<td>4.508</td>
<td>0.01102</td>
<td>0.1572</td>
</tr>
<tr>
<td>202732_at</td>
<td>PKIG</td>
<td>-2.313e-18</td>
<td>4.485</td>
<td>0.01128</td>
<td>0.1572</td>
</tr>
<tr>
<td>200660_at</td>
<td>S100A11</td>
<td>2.012e-16</td>
<td>4.463</td>
<td>0.01153</td>
<td>0.1572</td>
</tr>
<tr>
<td>218723_s_at</td>
<td>C13orf15</td>
<td>-7.556e-17</td>
<td>4.444</td>
<td>0.01175</td>
<td>0.1572</td>
</tr>
<tr>
<td>204730_at</td>
<td>RIMS3</td>
<td>-2.506e-16</td>
<td>4.44</td>
<td>0.0118</td>
<td>0.1572</td>
</tr>
<tr>
<td>212588_at</td>
<td>PTPRC</td>
<td>-9.252e-17</td>
<td>4.381</td>
<td>0.01251</td>
<td>0.1572</td>
</tr>
<tr>
<td>204589_at</td>
<td>NUAK1</td>
<td>-1.82e-16</td>
<td>4.379</td>
<td>0.01254</td>
<td>0.1572</td>
</tr>
<tr>
<td>207826_s_at</td>
<td>ID3</td>
<td>-3.77e-16</td>
<td>4.369</td>
<td>0.01266</td>
<td>0.1572</td>
</tr>
<tr>
<td>212195_at</td>
<td>IL6ST</td>
<td>1.261e-16</td>
<td>4.364</td>
<td>0.01273</td>
<td>0.1572</td>
</tr>
<tr>
<td>222150_s_at</td>
<td>PION</td>
<td>-9.714e-17</td>
<td>4.363</td>
<td>0.01273</td>
<td>0.1572</td>
</tr>
<tr>
<td>218080_x_at</td>
<td>FAF1</td>
<td>5.158e-16</td>
<td>4.352</td>
<td>0.01288</td>
<td>0.1572</td>
</tr>
<tr>
<td>212724_at</td>
<td>RND3</td>
<td>-1.542e-16</td>
<td>4.34</td>
<td>0.01303</td>
<td>0.1572</td>
</tr>
<tr>
<td>203932_at</td>
<td>HLA-DMB</td>
<td>-2.105e-16</td>
<td>4.332</td>
<td>0.01314</td>
<td>0.1572</td>
</tr>
<tr>
<td>214890_s_at</td>
<td>FAM149A</td>
<td>-2.393e-16</td>
<td>4.276</td>
<td>0.0139</td>
<td>0.1612</td>
</tr>
<tr>
<td>201301_s_at</td>
<td>ANXA4</td>
<td>-3.847e-16</td>
<td>4.256</td>
<td>0.01418</td>
<td>0.1612</td>
</tr>
<tr>
<td>213325_at</td>
<td>PVRL3</td>
<td>-2.409e-16</td>
<td>4.228</td>
<td>0.01458</td>
<td>0.1612</td>
</tr>
<tr>
<td>219159_s_at</td>
<td>SLAMF7</td>
<td>-8.635e-17</td>
<td>4.22</td>
<td>0.0147</td>
<td>0.1612</td>
</tr>
<tr>
<td>221727_at</td>
<td>SUB1</td>
<td>2.799e-16</td>
<td>4.204</td>
<td>0.01494</td>
<td>0.1612</td>
</tr>
<tr>
<td>209829_at</td>
<td>FAM65B</td>
<td>9.56e-17</td>
<td>4.179</td>
<td>0.01531</td>
<td>0.1612</td>
</tr>
<tr>
<td>202371_at</td>
<td>TCEAL4</td>
<td>-2.066e-16</td>
<td>4.177</td>
<td>0.01535</td>
<td>0.1612</td>
</tr>
<tr>
<td>221942_s_at</td>
<td>GUCY1A3</td>
<td>1.581e-16</td>
<td>4.165</td>
<td>0.01553</td>
<td>0.1612</td>
</tr>
<tr>
<td>201998_at</td>
<td>ST6GAL1</td>
<td>-2.891e-17</td>
<td>4.14</td>
<td>0.01592</td>
<td>0.1612</td>
</tr>
<tr>
<td>202746_at</td>
<td>ITM2A</td>
<td>3.084e-18</td>
<td>4.134</td>
<td>0.01602</td>
<td>0.1612</td>
</tr>
<tr>
<td>Probesets</td>
<td>Symbol</td>
<td>AveExpr</td>
<td>F</td>
<td>P.Value</td>
<td>adj.P.Val</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>207307_at</td>
<td>HTR2C</td>
<td>1.673e-16</td>
<td>4.104</td>
<td>0.01651</td>
<td>0.1635</td>
</tr>
<tr>
<td>201063_at</td>
<td>RCN1</td>
<td>5.86e-17</td>
<td>4.041</td>
<td>0.01758</td>
<td>0.1677</td>
</tr>
<tr>
<td>204613_at</td>
<td>PLCG2</td>
<td>4.24e-18</td>
<td>4.024</td>
<td>0.01789</td>
<td>0.1677</td>
</tr>
<tr>
<td>214608_s_at</td>
<td>EYA1</td>
<td>-3.392e-17</td>
<td>4.019</td>
<td>0.01798</td>
<td>0.1677</td>
</tr>
<tr>
<td>202906_s_at</td>
<td>TSPO</td>
<td>-4.572e-16</td>
<td>3.979</td>
<td>0.01871</td>
<td>0.1677</td>
</tr>
<tr>
<td>203795_s_at</td>
<td>BCL7A</td>
<td>-1.542e-16</td>
<td>3.955</td>
<td>0.01916</td>
<td>0.1677</td>
</tr>
<tr>
<td>212192_at</td>
<td>KCTD12</td>
<td>-1.295e-16</td>
<td>3.948</td>
<td>0.01929</td>
<td>0.1677</td>
</tr>
<tr>
<td>213502_s_at</td>
<td>LOC91316</td>
<td>1.465e-16</td>
<td>3.938</td>
<td>0.01948</td>
<td>0.1677</td>
</tr>
<tr>
<td>202177_at</td>
<td>GA6</td>
<td>-2.853e-17</td>
<td>3.938</td>
<td>0.01949</td>
<td>0.1685</td>
</tr>
<tr>
<td>209348_s_at</td>
<td>MAF</td>
<td>-4.626e-17</td>
<td>3.935</td>
<td>0.01955</td>
<td>0.1685</td>
</tr>
<tr>
<td>218409_s_at</td>
<td>DNAJC1</td>
<td>2.467e-17</td>
<td>3.929</td>
<td>0.01966</td>
<td>0.1677</td>
</tr>
<tr>
<td>205945_at</td>
<td>IL6R</td>
<td>-2.082e-16</td>
<td>3.92</td>
<td>0.01984</td>
<td>0.1677</td>
</tr>
<tr>
<td>200706_s_at</td>
<td>LITAF</td>
<td>-4.179e-16</td>
<td>3.883</td>
<td>0.02058</td>
<td>0.1685</td>
</tr>
<tr>
<td>212442_s_at</td>
<td>LASS6</td>
<td>1.727e-16</td>
<td>3.882</td>
<td>0.02061</td>
<td>0.1685</td>
</tr>
<tr>
<td>213245_at</td>
<td>ADCY1</td>
<td>-1.295e-16</td>
<td>3.876</td>
<td>0.02073</td>
<td>0.1685</td>
</tr>
<tr>
<td>201681_s_at</td>
<td>DLG5</td>
<td>1.798e-16</td>
<td>3.828</td>
<td>0.02176</td>
<td>0.1709</td>
</tr>
<tr>
<td>201212_at</td>
<td>LGMN</td>
<td>-1.557e-16</td>
<td>3.785</td>
<td>0.02271</td>
<td>0.1709</td>
</tr>
<tr>
<td>201841_s_at</td>
<td>HSPB1</td>
<td>-4.125e-16</td>
<td>3.765</td>
<td>0.02316</td>
<td>0.1709</td>
</tr>
<tr>
<td>202933_s_at</td>
<td>YES1</td>
<td>1.773e-16</td>
<td>3.751</td>
<td>0.0235</td>
<td>0.1709</td>
</tr>
<tr>
<td>206641_at</td>
<td>TNFRSF17</td>
<td>2.603e-16</td>
<td>3.745</td>
<td>0.02364</td>
<td>0.1709</td>
</tr>
<tr>
<td>203986_at</td>
<td>STBD1</td>
<td>-1.226e-16</td>
<td>3.741</td>
<td>0.02373</td>
<td>0.1709</td>
</tr>
<tr>
<td>208892_s_at</td>
<td>DUSP6</td>
<td>-7.864e-17</td>
<td>3.731</td>
<td>0.02396</td>
<td>0.1709</td>
</tr>
<tr>
<td>202824_at</td>
<td>GSTP1</td>
<td>-1.11e-16</td>
<td>3.712</td>
<td>0.02443</td>
<td>0.1709</td>
</tr>
<tr>
<td>205229_s_at</td>
<td>COCH</td>
<td>1.382e-16</td>
<td>3.685</td>
<td>0.02509</td>
<td>0.1709</td>
</tr>
<tr>
<td>219696_at</td>
<td>DENND1B</td>
<td>-3.369e-16</td>
<td>3.681</td>
<td>0.0252</td>
<td>0.1709</td>
</tr>
<tr>
<td>217995_at</td>
<td>SQRL</td>
<td>-1.661e-16</td>
<td>3.678</td>
<td>0.02526</td>
<td>0.1709</td>
</tr>
<tr>
<td>209198_s_at</td>
<td>SYT11</td>
<td>1.885e-16</td>
<td>3.671</td>
<td>0.02546</td>
<td>0.1709</td>
</tr>
<tr>
<td>205718_at</td>
<td>ITGB7</td>
<td>-3.115e-16</td>
<td>3.661</td>
<td>0.02571</td>
<td>0.1709</td>
</tr>
<tr>
<td>203476_at</td>
<td>TPBG</td>
<td>5.86e-17</td>
<td>3.659</td>
<td>0.02575</td>
<td>0.1709</td>
</tr>
<tr>
<td>204960_at</td>
<td>PTPRCAP</td>
<td>2.205e-16</td>
<td>3.657</td>
<td>0.0258</td>
<td>0.1709</td>
</tr>
<tr>
<td>203411_s_at</td>
<td>LMNA</td>
<td>2.209e-16</td>
<td>3.649</td>
<td>0.02601</td>
<td>0.1709</td>
</tr>
<tr>
<td>200839_s_at</td>
<td>CTSB</td>
<td>1.577e-16</td>
<td>3.625</td>
<td>0.02665</td>
<td>0.1709</td>
</tr>
<tr>
<td>209340_at</td>
<td>UAP1</td>
<td>-3.3e-16</td>
<td>3.624</td>
<td>0.02666</td>
<td>0.1709</td>
</tr>
<tr>
<td>218404_at</td>
<td>SNX10</td>
<td>-1.847e-16</td>
<td>3.621</td>
<td>0.02677</td>
<td>0.1709</td>
</tr>
<tr>
<td>219010_at</td>
<td>C1orf106</td>
<td>-3.48e-16</td>
<td>3.615</td>
<td>0.02691</td>
<td>0.1709</td>
</tr>
<tr>
<td>219551_at</td>
<td>EAF2</td>
<td>5.196e-16</td>
<td>3.612</td>
<td>0.027</td>
<td>0.1709</td>
</tr>
<tr>
<td>205943_at</td>
<td>TDO2</td>
<td>-1.279e-16</td>
<td>3.61</td>
<td>0.02705</td>
<td>0.1709</td>
</tr>
<tr>
<td>212096_s_at</td>
<td>MTUS1</td>
<td>1.789e-16</td>
<td>3.594</td>
<td>0.02748</td>
<td>0.1709</td>
</tr>
<tr>
<td>207039_at</td>
<td>CDKN2A</td>
<td>-2.101e-17</td>
<td>3.594</td>
<td>0.02749</td>
<td>0.1709</td>
</tr>
<tr>
<td>204254_s_at</td>
<td>VDR</td>
<td>-2.39e-17</td>
<td>3.565</td>
<td>0.02829</td>
<td>0.1741</td>
</tr>
<tr>
<td>Probesets</td>
<td>Symbol</td>
<td>AveExpr</td>
<td>F</td>
<td>P.Value</td>
<td>adj.P.Val</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>202242_at</td>
<td>TSPAN7</td>
<td>2.136e-16</td>
<td>3.543</td>
<td>0.02892</td>
<td>0.1763</td>
</tr>
<tr>
<td>202136_at</td>
<td>ZMYND11</td>
<td>-2.375e-16</td>
<td>3.522</td>
<td>0.02955</td>
<td>0.1764</td>
</tr>
<tr>
<td>218847_at</td>
<td>IGF2BP2</td>
<td>-3.422e-16</td>
<td>3.52</td>
<td>0.02961</td>
<td>0.1764</td>
</tr>
<tr>
<td>212843_at</td>
<td>NCAM1</td>
<td>-3.286e-17</td>
<td>3.499</td>
<td>0.03021</td>
<td>0.1764</td>
</tr>
<tr>
<td>218718_at</td>
<td>PDGFC</td>
<td>-9.252e-18</td>
<td>3.495</td>
<td>0.03035</td>
<td>0.1764</td>
</tr>
<tr>
<td>201647_s_at</td>
<td>SCARB2</td>
<td>1.519e-16</td>
<td>3.48</td>
<td>0.03079</td>
<td>0.1764</td>
</tr>
<tr>
<td>200697_at</td>
<td>HK1</td>
<td>3.84e-16</td>
<td>3.463</td>
<td>0.03134</td>
<td>0.1764</td>
</tr>
<tr>
<td>209619_at</td>
<td>CD74</td>
<td>-1.058e-16</td>
<td>3.458</td>
<td>0.0315</td>
<td>0.1764</td>
</tr>
<tr>
<td>203397_s_at</td>
<td>GALNT3</td>
<td>-3.481e-17</td>
<td>3.457</td>
<td>0.03152</td>
<td>0.1764</td>
</tr>
<tr>
<td>201828_s_at</td>
<td>FAM127A</td>
<td>-5.089e-16</td>
<td>3.451</td>
<td>0.03172</td>
<td>0.1764</td>
</tr>
<tr>
<td>60474_s_at</td>
<td>FERMT1</td>
<td>-1.164e-16</td>
<td>3.441</td>
<td>0.03202</td>
<td>0.1764</td>
</tr>
<tr>
<td>221122_at</td>
<td>HRASLS2</td>
<td>2.567e-16</td>
<td>3.43</td>
<td>0.03229</td>
<td>0.1764</td>
</tr>
<tr>
<td>204688_at</td>
<td>SGCE</td>
<td>-3.701e-16</td>
<td>3.431</td>
<td>0.03236</td>
<td>0.1764</td>
</tr>
<tr>
<td>212097_at</td>
<td>CAV1</td>
<td>-7.71e-18</td>
<td>3.41</td>
<td>0.03304</td>
<td>0.1764</td>
</tr>
<tr>
<td>217967_s_at</td>
<td>FAM129A</td>
<td>1.664e-16</td>
<td>3.41</td>
<td>0.03306</td>
<td>0.1764</td>
</tr>
<tr>
<td>202946_s_at</td>
<td>BTBD3</td>
<td>-7.633e-17</td>
<td>3.407</td>
<td>0.03314</td>
<td>0.1764</td>
</tr>
<tr>
<td>206632_s_at</td>
<td>APOBEC3B</td>
<td>-2.018e-16</td>
<td>3.393</td>
<td>0.03362</td>
<td>0.1764</td>
</tr>
<tr>
<td>219003_s_at</td>
<td>MANEA</td>
<td>-3.192e-16</td>
<td>3.367</td>
<td>0.03448</td>
<td>0.1764</td>
</tr>
<tr>
<td>205903_s_at</td>
<td>KCNN3</td>
<td>-3.3e-16</td>
<td>3.366</td>
<td>0.03452</td>
<td>0.1764</td>
</tr>
<tr>
<td>206700_s_at</td>
<td>KDM5D</td>
<td>1.419e-16</td>
<td>3.363</td>
<td>0.03464</td>
<td>0.1764</td>
</tr>
<tr>
<td>218974_at</td>
<td>SOBP</td>
<td>-2.213e-16</td>
<td>3.359</td>
<td>0.03478</td>
<td>0.1764</td>
</tr>
<tr>
<td>206609_at</td>
<td>MAGEC1</td>
<td>1.288e-16</td>
<td>3.359</td>
<td>0.03479</td>
<td>0.1764</td>
</tr>
<tr>
<td>205297_s_at</td>
<td>CD79B</td>
<td>-1.766e-16</td>
<td>3.317</td>
<td>0.03625</td>
<td>0.1802</td>
</tr>
<tr>
<td>205933_at</td>
<td>SETBP1</td>
<td>1.195e-17</td>
<td>3.314</td>
<td>0.03635</td>
<td>0.1802</td>
</tr>
<tr>
<td>202388_at</td>
<td>RGS2</td>
<td>-2.22e-16</td>
<td>3.31</td>
<td>0.03652</td>
<td>0.1802</td>
</tr>
<tr>
<td>201462_at</td>
<td>SCRNI</td>
<td>-4.163e-17</td>
<td>3.293</td>
<td>0.03716</td>
<td>0.1802</td>
</tr>
<tr>
<td>203167_at</td>
<td>TIMP2</td>
<td>4.318e-17</td>
<td>3.284</td>
<td>0.03747</td>
<td>0.1802</td>
</tr>
<tr>
<td>205098_at</td>
<td>CCR1</td>
<td>-9.252e-17</td>
<td>3.277</td>
<td>0.03772</td>
<td>0.1802</td>
</tr>
<tr>
<td>219014_at</td>
<td>PLAC8</td>
<td>1.48e-16</td>
<td>3.272</td>
<td>0.03792</td>
<td>0.1802</td>
</tr>
<tr>
<td>214452_at</td>
<td>BCAT1</td>
<td>-1.593e-16</td>
<td>3.271</td>
<td>0.03795</td>
<td>0.1802</td>
</tr>
<tr>
<td>202017_at</td>
<td>EPHX1</td>
<td>-1.218e-16</td>
<td>3.268</td>
<td>0.03808</td>
<td>0.1802</td>
</tr>
<tr>
<td>210473_s_at</td>
<td>GPR125</td>
<td>-1.511e-16</td>
<td>3.247</td>
<td>0.0389</td>
<td>0.1827</td>
</tr>
<tr>
<td>204409_s_at</td>
<td>EIF1AY</td>
<td>1.234e-17</td>
<td>3.233</td>
<td>0.03944</td>
<td>0.183</td>
</tr>
<tr>
<td>220603_s_at</td>
<td>MCTP2</td>
<td>6.63e-17</td>
<td>3.225</td>
<td>0.03975</td>
<td>0.183</td>
</tr>
<tr>
<td>212345_s_at</td>
<td>CREB3L2</td>
<td>-2.831e-16</td>
<td>3.208</td>
<td>0.04044</td>
<td>0.183</td>
</tr>
<tr>
<td>203710_at</td>
<td>ITPR1</td>
<td>-5.551e-17</td>
<td>3.208</td>
<td>0.04045</td>
<td>0.183</td>
</tr>
<tr>
<td>200999_s_at</td>
<td>CKAP4</td>
<td>-1.82e-16</td>
<td>3.203</td>
<td>0.04066</td>
<td>0.183</td>
</tr>
<tr>
<td>204923_at</td>
<td>SASH3</td>
<td>-1.542e-17</td>
<td>3.202</td>
<td>0.04069</td>
<td>0.183</td>
</tr>
<tr>
<td>204364_s_at</td>
<td>REEP1</td>
<td>-1.739e-16</td>
<td>3.155</td>
<td>0.04263</td>
<td>0.1894</td>
</tr>
<tr>
<td>218618_s_at</td>
<td>FNDC3B</td>
<td>3.523e-16</td>
<td>3.153</td>
<td>0.04272</td>
<td>0.1894</td>
</tr>
</tbody>
</table>
Table 11.2: (continued)

<table>
<thead>
<tr>
<th>Probesets</th>
<th>Symbol</th>
<th>AveExpr</th>
<th>F</th>
<th>P.Value</th>
<th>adj.P.Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>203324_s_at</td>
<td>CAV2</td>
<td>-5.86e-17</td>
<td>3.143</td>
<td>0.04316</td>
<td>0.1895</td>
</tr>
<tr>
<td>211105_s_at</td>
<td>NFATC1</td>
<td>-5.86e-17</td>
<td>3.136</td>
<td>0.04346</td>
<td>0.1895</td>
</tr>
<tr>
<td>220306_at</td>
<td>FAM46C</td>
<td>-4.042e-16</td>
<td>3.132</td>
<td>0.04364</td>
<td>0.1895</td>
</tr>
<tr>
<td>213135_at</td>
<td>TIAM1</td>
<td>9.213e-17</td>
<td>3.118</td>
<td>0.04424</td>
<td>0.1896</td>
</tr>
<tr>
<td>206624_at</td>
<td>USP9Y</td>
<td>2.56e-16</td>
<td>3.116</td>
<td>0.04432</td>
<td>0.1896</td>
</tr>
<tr>
<td>211373_s_at</td>
<td>PSEN2</td>
<td>9.059e-17</td>
<td>3.106</td>
<td>0.04478</td>
<td>0.1896</td>
</tr>
<tr>
<td>206167_s_at</td>
<td>ARHGAP6</td>
<td>-3.547e-17</td>
<td>3.096</td>
<td>0.04522</td>
<td>0.1896</td>
</tr>
<tr>
<td>201858_s_at</td>
<td>SRGN</td>
<td>6.091e-17</td>
<td>3.095</td>
<td>0.04529</td>
<td>0.1896</td>
</tr>
<tr>
<td>201540_at</td>
<td>FHL1</td>
<td>-1.735e-16</td>
<td>3.088</td>
<td>0.04557</td>
<td>0.1896</td>
</tr>
<tr>
<td>218450_at</td>
<td>HEBP1</td>
<td>-1.156e-16</td>
<td>3.085</td>
<td>0.04575</td>
<td>0.1896</td>
</tr>
<tr>
<td>213913_s_at</td>
<td>TBC1D30</td>
<td>-1.372e-16</td>
<td>3.067</td>
<td>0.04654</td>
<td>0.1916</td>
</tr>
<tr>
<td>217551_at</td>
<td>OR7E14P</td>
<td>-1.11e-16</td>
<td>3.039</td>
<td>0.04789</td>
<td>0.1946</td>
</tr>
<tr>
<td>221666_s_at</td>
<td>PYCARD</td>
<td>5.705e-17</td>
<td>3.039</td>
<td>0.04789</td>
<td>0.1946</td>
</tr>
<tr>
<td>214131_at</td>
<td>CYorf15B</td>
<td>-6.168e-17</td>
<td>3.032</td>
<td>0.04822</td>
<td>0.1947</td>
</tr>
<tr>
<td>219812_at</td>
<td>PVRIG</td>
<td>1.372e-16</td>
<td>3.005</td>
<td>0.04954</td>
<td>0.1983</td>
</tr>
<tr>
<td>213478_at</td>
<td>KAZ</td>
<td>2.282e-16</td>
<td>2.999</td>
<td>0.04985</td>
<td>0.1983</td>
</tr>
</tbody>
</table>
12 Testing the \textit{BCell} LDA Classifier on the Arkansas Data

12.1 Predicting the Melphalan Resistance Classes

Firstly, the expression data are extracted from the expression set.

\begin{verbatim}
> Arkansas.matrix <- exprs(GEPArkansas)
\end{verbatim}

Next, the expression set is ordered according to the LDA classifier

\begin{verbatim}
> sortBCellArkansas <- Arkansas.matrix[probesets.BCell, ]
\end{verbatim}

and scaled so that each gene expression profile has unit variance and zero mean.

\begin{verbatim}
> sortBCellArkansas <- t(scale(t(sortBCellArkansas)))
\end{verbatim}

Finally, the resistance classes are predicted.

\begin{verbatim}
> predictDLDAArkansas <- predict(fitLDABCell, t(sortBCellArkansas))
\end{verbatim}

The predictions are summarized in Table \ref{table:resistance_classes}.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate</td>
<td>336</td>
</tr>
<tr>
<td>Resistant</td>
<td>119</td>
</tr>
<tr>
<td>Sensitive</td>
<td>104</td>
</tr>
</tbody>
</table>

\textit{Table 12.1}: A summary of the predicted resistance classes made by the \textit{BCell} LDA classifier.

12.2 Association between the Resistance Classes and OS

In the following code chunk Kaplan-Meier survival curves are created for the resistance classes. The result is shown in Figure \ref{figure:os_survival}.

\begin{verbatim}
> pdf(file.path(figure.output, "LDABCellArkansasKMplotOS.pdf"))
> par(mfrow = c(1, 1))
> metadataArkansas$DLDAThreshold <- predictDLDAArkansas$class
> p.BCell.LDA.OS <- PlotKM.sda(predictDLDAArkansas$class, metadataArkansas$OS,
  ylab = "Overall Survival",
  col = our.colscheme[order(names(our.colscheme))],
  xlab = "Time (months)"
)
\end{verbatim}
12.3 Association between the Resistance Classes and EFS

In the following code chunk Kaplan-Meier survival curves are created for the resistance classes. The result is shown in Figure 12.2.

```r
> pdf(file.path(figure.output,"LDABCellArkansasKMplot.pdf"))
> par(mfrow = c(1, 1))
> metadataArkansas$DLDAThreshold <- predictDLDAArkansas$class
> p.BCell.LDA.EFS <- PlotKM.sda(predictDLDAArkansas$class, metadataArkansas$EFS,
  ylab = "Event Free Survival",
  col  = our.colscheme[order(names(our.colscheme))],
  xlab = "Time (months)",
  legend = names(our.colscheme),
  col.leg = our.colscheme,
  main = paste("Arkansas", 
    "BCell Kaplan-Meier OS Curves",sep=""))
> dev.off()
```

Figure 12.1: Kaplan-Meier survival curves based on the BCell LDA classifier. The logrank test comparing the survival curves results in a P-value of 0.006.
Figure 12.2: Kaplan-Meier survival curves based on the BCell LDA classifier. The logrank test comparing the survival curves results in a P-value of 2e-05.
13 Establishment of the *BCell* SPLS Resistance Index

13.1 Cross-Validation

When the output variable is one dimensional, SPLS has two tuning parameters: the number of hidden components $K$ and the shrinkage parameter $\eta$. Below, the two tuning parameters are determined through CV of a wide range of values for the parameter `var.cutoff` in the function `nsFilter`. The combination of $K$, $\eta$ and `var.cutoff` which results in the minimum Mean Square Prediction Error (MSPE) is chosen for further analysis.

By selecting the expression profiles at random instead of using `nsFilter` it is possible to investigate whether the function `nsFilter` choose expression profiles that perform better than random noise.

In order to avoid gene expression profiles which are not marginally related to melphalan resistance sure independence screening is used. Only gene expression profiles which are significantly correlated at the significance level specified below are used in the analysis.

```r
> pval <- 0.05

The P-values are not adjusted.

> adjust <- "none"

> set.seed(1000)
> BCell.SPLS.crossval.res.file <-
  file.path(BCell.gen.dir, "BCellSPLScrossval.res.Rdata")
> if(file.exists(BCell.SPLS.crossval.res.file)){
  load(BCell.SPLS.crossval.res.file)
} else {
  cutoffs <- seq(0.1, 0.98, by = 0.01)

  BCell.SPLS.crossval.res <-
    data.frame(cutoffs = cutoffs,
                eta = 0, K=0,
                mspe = 0, rand = 0)

  for(cutoff in cutoffs){
    BCell.filtered <- nsFilter(GEPBCell133a,
                                var.cutoff = cutoff)$eset
    n.feat <- length(featureNames(BCell.filtered))
  } 
```
```r
rand.probes <- sample(length(featureNames(GEPBCell133a)), n.feat)

BCell.random <- exprs(GEPBCell133a[rand.probes, ])

cutoff <- seq(0.2, 0.99, length.out = 200)

fit.spls.cv <-
  cv.spls(x = t(BCell.random),
  y = BCellResistanceindex,
  fold = length(BCellResistanceindex),
  eta = laeta,
  K = c(1, 2, 3),
  plot.it = FALSE,
  do.SIS = TRUE,
  pval = pval,
  adjust = adjust)

BCell.SPLS.crossval.res[BCell.SPLS.crossval.res$cutoffs ==
cutoff, ]$rand <- min(fit.spls.cv$mspe)

BCellGEP <- exprs(BCell.filtered)

sis.cv <- SISCV(BCellGEP, BCellResistanceindex, adjust = "fdr")

fit.spls.cv <-
  cv.spls(x = t(BCellGEP),
  y = BCellResistanceindex,
  fold = length(BCellResistanceindex),
  eta = eta,
  K = c(1, 2, 3),
  plot.it = FALSE,
  do.SIS = TRUE,
  pval = pval,
  adjust = adjust)

BCell.SPLS.crossval.res[BCell.SPLS.crossval.res$cutoffs ==
cutoff, ]$delta <- fit.spls.cv$eta.opt
BCell.SPLS.crossval.res[BCell.SPLS.crossval.res$cutoffs ==
cutoff, ]$K <- fit.spls.cv$K.opt
BCell.SPLS.crossval.res[BCell.SPLS.crossval.res$cutoffs ==
cutoff, ]$mspe <- min(fit.spls.cv$mspe)
}
save(BCell.SPLS.crossval.res, file = BCell.SPLS.crossval.res.file)
```
The result of the CV routine is extracted by the code chunk below and the optimal values for the three parameters are shown in Table 13.1.

```r
> mspe.min <- BCell.SPLS.crossval.res[BCell.SPLS.crossval.res$mspe ==
min(BCell.SPLS.crossval.res$mspe), ]
> n.mspe <- dim(mspe.min)[1]
> cutoff <- mspe.min$cutoffs[n.mspe]
> K <- mspe.min$K[n.mspe]
> eta <- mspe.min$eta[n.mspe]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optimal value by CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>cutoff</td>
<td>0.7400</td>
</tr>
<tr>
<td>η</td>
<td>0.8193</td>
</tr>
<tr>
<td>K</td>
<td>2.0000</td>
</tr>
</tbody>
</table>

Table 13.1

The code chunk below constructs a plot of the minimum MSPE achieved through each of the tested values for the parameter `var.cutoff` together with minimum MSPE achieved through the randomly selected gene expressions.

```r
> pdf(file.path(figure.output, "BCellMSPEcomparison.pdf"))
> plot(BCell.SPLS.crossval.res$cutoffs,
BCell.SPLS.crossval.res$mspe,
ylim = c(min(BCell.SPLS.crossval.res$mspe,
BCell.SPLS.crossval.res$rand),
max(BCell.SPLS.crossval.res$mspe,
BCell.SPLS.crossval.res$rand)),
ylab = "MSPE", xlab = "Fraction filtered out", type = "n",
main = "BCell MSPE")
> lines(BCell.SPLS.crossval.res$cutoffs,
BCell.SPLS.crossval.res$mspe, lty = 1)
> lines(BCell.SPLS.crossval.res$cutoffs,
BCell.SPLS.crossval.res$rand, lty = 2)
> legend("topleft", lty = c(1, 2),
legend = c("nsFilter", "Random"),
bty = "n")
> dev.off()
```

The plot is shown in Figure 13.1. Setting `var.cutoff` equal to 0.74 results in the minimal MSPE.
Figure 13.1: The minimum MSPE achieved through CV on $K$ and $\eta$ in SPLS for a variety of values for the parameter var.cutoff. The dashed line represents the minimum MSPE achieved through CV on $K$ and $\eta$ in SPLS for randomly selected gene expressions. The smallest minimum MSPE is obtained with a value of var.cutoff equal to 0.74.

When setting var.cutoff equal to 0.74 the resulting CV on $K$ and $\eta$ is shown in Figure 13.2. In order to establish the plot filtering of the expression data with the parameter var.cutoff set equal to 0.74 is performed.

```R
> BCell.filtered <- nsFilter(GEPBCell133a, var.cutoff = cutoff)$eset
> GEPBCell <- exprs(BCell.filtered)
```

When using these settings in nsFilter only the 3297 most varying gene expression profiles are used for further analysis.

```R
> BCellMSPE.file <- file.path(figure.output, "BCellMSPE.pdf")
> if(!file.exists(BCellMSPE.file)){
    fit.spls.cv <-
```
cv.spls(x = t(GEPBCell),
y = BCellResistanceindex,
fold = n.BCell,
eta = seq(0.5, 0.99, length.out = 1000),
K = c(1, 2, 3),
plot.it = FALSE,
do.SIS = TRUE,
pval = pval,
adjust = adjust)

pdf(file.path(figure.output, "BCellMSPE.pdf"),
    width = 7, height = 7)

trace.mspe(fit.spls.cv, header = "BCell MSPE", xlim = c(0.6, 1))

dev.off()

Figure 13.2: The MSPE for the SPLS regression with var.cutoff in nsFilter set equal to 0.74. This leads to the selection of K=2 hidden components and a sparsity parameter $\eta=0.8193$. 

83
In order to investigate the accuracy of the SPLS model with the specific settings the CV predicted resistance index is calculated once more and stored in the object \texttt{pred.1}.

\begin{verbatim}
> pred.1 <- matrix(NA, nrow = n.BCell, ncol = 2)
> rownames(pred.1) <- BCell.names.sorted
> colnames(pred.1) <- c("Measured", "predicted")
> pred.1[, 1] <- BCellResistanceindex[BCell.names.sorted, ]
> for(i in BCell.names.sorted){
  GEPBCell.cv <- exprs(BCell.filtered[, sampleNames(GEPBCell133a) %w/o% i])
  y <- BCellResistanceindex[sampleNames(GEPBCell133a) %w/o% i, ]
  BCell.fit.spls.cv <- spls(x = t(GEPBCell.cv), y = y, K = K, eta = eta, do.SIS = TRUE, pval = pval, adjust = adjust)
  newx <- exprs(GEPBCell133a)[row.names(GEPBCell.cv), i]
  pred.1[i, 2] <- as.vector(predict(BCell.fit.spls.cv, newx = t(as.matrix(newx))))
}
\end{verbatim}

In Figure 13.3A the CV predicted melphalan resistance is plotted against the measured GI$_{50}$ values.

When the parameter \texttt{var.cutoff} in the function \texttt{nsFilter} is determined in this manner the resulting MSPE obtained through CV with SPLS is overoptimistic. In order to determine how well the model performs, CV over the chosen parameters including \texttt{var.cutoff} is performed.

\begin{verbatim}
> pred.2 <- matrix(NA, nrow = n.BCell, ncol = 3)
> rownames(pred.2) <- BCell.names.sorted
> colnames(pred.2) <- c("Measured", "Predicted Naive", "Predicted")
> pred.2[, 1] <- BCellResistanceindex[BCell.names.sorted, ]
> pred.2[, 2] <- pred.1[, 2]
> for(i in BCell.names.sorted){
  BCell.filtered.cv <- nsFilter(GEPBCell133a[, sampleNames(GEPBCell133a) %w/o% i],
\end{verbatim}
$$\text{var.cutoff} = \text{cutoff}$$

\[
\text{GEPBCell.cv} \leftarrow \text{exprs(BCell.filtered.cv)}
\]

\[
y \leftarrow \text{BCellResistanceindex[sampleNames(GEPBCell133a) \%o\% i, ]}
\]

\[
\text{BCell.fit.spls.cv} \leftarrow \text{spls}(x = \text{t(GEPBCell.cv)},
\quad \text{y} = \text{y},
\quad \text{K} = \text{K},
\quad \text{eta} = \text{eta},
\quad \text{do.SIS} = \text{TRUE},
\quad \text{pval} = \text{pval},
\quad \text{adjust} = \text{adjust})
\]

\[
\text{newx} \leftarrow \text{exprs(GEPBCell133a)[row.names(GEPBCell.cv), i]}
\]

\[
\text{pred.2[i, 3]} \leftarrow \text{as.vector(predict(BCell.fit.spls.cv,}
\quad \text{newx = t(as.matrix(newx)))}
\]

\}

In Figure 13.3 the predicted resistance index is plotted against the measured resistance index. As assumed the result is slightly worse than when the parameter \text{var.cutoff} is not included in the CV step.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig13.3}
\caption{Predicted resistance index vs. the measured resistance index. The left panel shows CV after filtering. The right panel shows predictions where filtering is performed each time a cell line is left out.}
\end{figure}
13.2 Establishment of the Gene Expression Signature

In the previous section various values were determined through CV. In this section these values are used for fitting a gene expression signature. For this purpose the function `spls` from the package `spls` is used.

The gene expression signature is established and saved for later predictions.

```r
> BCell.fit.spls <- spls(x = t(GEPBCell),
                        y = BCellResistanceindex,
                        K = K,
                        eta = eta,
                        do.SIS = TRUE,
                        pval = pval,
                        adjust = adjust)
> save(BCell.fit.spls,
      file = file.path("Generated data", "BCell", "BCell.fit.spls.Rdata"))
```

13.3 Investigation of the chosen Gene Expression Profile

Only 19 gene expression profiles are used in the signature. In the code chunk below the gene symbols for the 19 gene expression profiles are looked up. Furthermore, in order to see if these gene profiles have been observed in other papers studying chemoresistance the following search is conducted on PubMed.

```r
> resTable <- lookUpPubmed(BCell.fit.spls, n.coef = 3)
```

The result of the search is shown in Table 13.2. In order to investigate how the identified probes relate to chemoresistance marginally correlations are plotted in Section 13.4.

```r
> BCellCoef <- coef(BCell.fit.spls)
> BCellCoef <- BCellCoef[BCellCoef != 0, ]
> BCellCoef <- round(BCellCoef[order(BCellCoef)], 4)
> corm <- GEPBCell[.names(BCellCoef), ]
> for(i in 1:length(BCellCoef)){
    pdf(paste(figure.output, "/", i, "BCellcorrelations.pdf", sep = ""),
        width = 7, height = 7)
    plot(BCellResistanceindex ~ corm[, i],
         ylab = "Resistance index",
         xlab = "Gene expression",
         type = "n")
    title(unlist(lookUp(rownames(corm)[i],
                      "hgu133plus2", "SYMBOL")))
}
<table>
<thead>
<tr>
<th>U133 ID</th>
<th>Gene Symbol</th>
<th>Mean</th>
<th>SD</th>
<th>Location</th>
<th>Weight</th>
<th>PMID</th>
</tr>
</thead>
<tbody>
<tr>
<td>205990_s_at</td>
<td>WNT5A</td>
<td>10.719</td>
<td>0.719</td>
<td>3p21-p14</td>
<td>-0.065</td>
<td>3</td>
</tr>
<tr>
<td>203708_at</td>
<td>PDE4B</td>
<td>6.272</td>
<td>1.110</td>
<td>1p31</td>
<td>-0.053</td>
<td>2</td>
</tr>
<tr>
<td>201990_s_at</td>
<td>CREBL2</td>
<td>5.980</td>
<td>0.667</td>
<td>12p13</td>
<td>-0.046</td>
<td>0</td>
</tr>
<tr>
<td>218751_s_at</td>
<td>FBXW7</td>
<td>7.539</td>
<td>1.022</td>
<td>4q31.3</td>
<td>-0.044</td>
<td>3</td>
</tr>
<tr>
<td>201889_at</td>
<td>FAM3C</td>
<td>6.752</td>
<td>1.808</td>
<td>7q31</td>
<td>-0.039</td>
<td>0</td>
</tr>
<tr>
<td>206405_s_at</td>
<td>USP6</td>
<td>9.859</td>
<td>0.935</td>
<td>17p13</td>
<td>0.038</td>
<td>0</td>
</tr>
<tr>
<td>219049_at</td>
<td>CSGALNACT1</td>
<td>8.712</td>
<td>0.867</td>
<td>8p21.3</td>
<td>-0.037</td>
<td>0</td>
</tr>
<tr>
<td>205862_at</td>
<td>GREB1</td>
<td>5.326</td>
<td>0.953</td>
<td>2p25.1</td>
<td>-0.034</td>
<td>2</td>
</tr>
<tr>
<td>219748_at</td>
<td>TREML2</td>
<td>6.556</td>
<td>2.162</td>
<td>6p21.1</td>
<td>-0.033</td>
<td>0</td>
</tr>
<tr>
<td>204786_s_at</td>
<td>IFNAR2</td>
<td>10.851</td>
<td>0.905</td>
<td>21q22.1, 21q22.11</td>
<td>-0.033</td>
<td>4</td>
</tr>
<tr>
<td>204204_at</td>
<td>SLC31A2</td>
<td>4.451</td>
<td>1.450</td>
<td>9q31-q32</td>
<td>-0.025</td>
<td>0</td>
</tr>
<tr>
<td>217825_s_at</td>
<td>UBE2J1</td>
<td>8.450</td>
<td>0.877</td>
<td>6q15</td>
<td>-0.020</td>
<td>0</td>
</tr>
<tr>
<td>213555_at</td>
<td>RWDD2A</td>
<td>5.133</td>
<td>1.933</td>
<td>6q14.2</td>
<td>-0.019</td>
<td>0</td>
</tr>
<tr>
<td>212122_at</td>
<td>RHOQ</td>
<td>9.410</td>
<td>0.813</td>
<td>2p21</td>
<td>-0.016</td>
<td>0</td>
</tr>
<tr>
<td>203895_at</td>
<td>PLCB4</td>
<td>7.099</td>
<td>2.446</td>
<td>20p12</td>
<td>-0.015</td>
<td>0</td>
</tr>
<tr>
<td>202043_s_at</td>
<td>SMS</td>
<td>7.162</td>
<td>1.033</td>
<td>Xp22.1</td>
<td>0.011</td>
<td>48</td>
</tr>
<tr>
<td>217104_at</td>
<td>ST20</td>
<td>6.569</td>
<td>1.862</td>
<td>15q25.1</td>
<td>0.012</td>
<td>0</td>
</tr>
<tr>
<td>212055_at</td>
<td>C18orf10</td>
<td>7.653</td>
<td>0.776</td>
<td>18q12.2</td>
<td>0.025</td>
<td>0</td>
</tr>
<tr>
<td>221210_s_at</td>
<td>NPL</td>
<td>4.953</td>
<td>1.076</td>
<td>1q25</td>
<td>0.032</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 13.2:** Location and weight for the 19 probes identified by use of the BCell panel in combination with SPLS regression.
13.4 Plots of Marginal Associations

**WNT5A**

**PDE4B**

**CREBL2**

**FBXW7**
Gene expression vs. Resistance index for different cell lines:

- **ST20**
- **C18orf10**
- **NPL**

Correlation coefficients:  
- Cor = 0.67  
- Cor = 0.78  
- Cor = 0.76
14 Testing the \textit{BCell} SPLS Resistance Index

14.1 Predicting the Melphalan Resistance in the \textit{Arkansas} data

Firstly, the expression data are extracted from the expression set.

\begin{verbatim}
> arkansas.matrix <- exprs(GEPArkansas)
\end{verbatim}

Next, the probes which were removed by \texttt{nsFilter} with \texttt{var.cutoff} set equal to 0.74 are discarded from the matrix.

\begin{verbatim}
> BCellArkansasTest <-
    arkansas.matrix[featureNames(BCell.filtered),]
\end{verbatim}

Finally, the signature is used to predict each patient’s resistance index.

\begin{verbatim}
> BCellArkansasindex <- as.vector(predict(BCell.fit.spls,
    newx = t(BCellArkansasTest)))
\end{verbatim}

14.1.1 Association between the Resistance Index and OS

\textbf{Kaplan-Meier Survival Curves}

Kaplan-Meier survival curves are constructed for each of the groups sensitive, intermediate and resistant. The 25\% with the lowest predicted melphalan resistance index are categorized as sensitive and the 25\% with the highest predicted melphalan resistance index are categorized as resistant. The remaining subjects are characterised as having intermediate resistance.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 14.1.

\begin{verbatim}
> pdf(file.path(figure.output,"BCellarkansasOSKMplot.pdf"))
> BCellarkansasOSKM.P <-
    PlotKM(BCellArkansasindex, metadataArkansas$OS,
    cut.points = cut.points,
    xlab = "Time (months)",
    ylab = "OS ratio",
    xmax = 110,
    main = "Arkansas \n BCell Kaplan-Meier OS curves")
> dev.off()
\end{verbatim}
Figure 14.1: Kaplan-Meier survival curves based on the BCell resistance index. The logrank test comparing the survival curves results in a P-value of 1e-06.

Cox Proportional Hazards

The HR and its 95% confidence interval are calculated between the sensitive and resistant classes.

\[
\text{HR} = 2.9 \ (2.41, 3.35)
\]

A Cox proportional hazards model is fitted where the association between log relative hazard and the resistance index is modelled by a spline with four knots.

\[
\text{coxfit.os} = \text{coxph}(\text{metadataArkansas$OS} \sim \text{rcs(BCellArkansasindex, n.knots)})
\]

Table 14.1 summarizes three tests for no association between the log relative hazard and the resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the BCell resistance index. The result is depicted in Figure 14.2.
Table 14.1: A summary of three tests for no association between the log relative hazard and the resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test</th>
<th>D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>29.7</td>
<td>3</td>
<td>1.62e−06</td>
</tr>
<tr>
<td>Wald test</td>
<td>27.9</td>
<td>3</td>
<td>3.79e−06</td>
</tr>
<tr>
<td>Score (log rank) test</td>
<td>29.4</td>
<td>3</td>
<td>1.87e−06</td>
</tr>
</tbody>
</table>

```R
> pdf(file.path(figure.output,"BCellarkanassOSPredictors.pdf"))
> par(mfrow = c(1, 1))
> d <- datadist(BCellArkansasindex)
> options(datadist = "d", width = 150)
> BCellf <- cph(metadataArkansas$OS ~ rcs(BCellArkansasindex, n.knots))
> plot(BCellf, xlab = "Fitted BCell resistance index")
> title("Arkansas \n BCell - OS Cox Proportional Hazards")
> legend("bottomright", ",", bty = "n",
       title = paste("P-value = ",
                     as.character(signif(unlist(BCellf)$stats.P, 1)),
                     sep = ""))
> dev.off()
```
14.1.2 Association between the Resistance Index and EFS

Kaplan-Meier Survival Curves

Similarly to the previous section Kaplan-Meier survival curves are made with the patients grouped to be sensitive, intermediate or resistant, however, EFS is used as endpoint instead of OS.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 14.3.

```
> pdf(file.path(figure.output,"BCellarkansasEFSKMplot.pdf"))
> BCellarkansasEFSKM.P <-
>   PlotKM(BCellArkansasindex,metadataArkansas$EFS,
>          cut.points = cut.points,
>          xlab = "Time (months)",
>          ylab = "EFS ratio",
>          xmax = 110,
>          main = "Arkansas \n BCell Kaplan-Meier EFS curves"
>
> dev.off()
```
Figure 14.3: Kaplan-Meier survival curves based on the *BCell* resistance index. The logrank test comparing the survival curves results in a *P*-value of 5e-04.

**Cox Proportional Hazards**

The HR and its 95% confidence interval are calculated between the sensitive and resistant classes.

```r
> HR <- CalcHR(BCellArkansasindex, metadataArkansas$EFS.months,
>                  metadataArkansas$EFS.censor == 1)
> HR
```

```
[1] "HR = 2.2 (1.75,2.67)"
```

A Cox proportional hazards model is fitted where the association between log relative hazard and the resistance index is modelled by a spline with four knots.

```r
> n.knots <- 4
> coxfit.os <- coxph(metadataArkansas$EFS ~ rcs(BCellArkansasindex, n.knots))
```

Table 14.2 summarizes three tests for no association between the log relative hazard and the resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the *BCell* resistance index. The result is depicted in Figure 14.4.

```r
> pdf(file.path(figure.output,"BCellarkansasEFSPredictors.pdf"))
> par(mfrow=c(1,1))
```
## Table 14.2: A summary of three tests for no association between log relative hazard and the predicted resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test</th>
<th>D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>17.9</td>
<td>3</td>
<td>0.000453</td>
</tr>
<tr>
<td>Wald test</td>
<td>17.0</td>
<td>3</td>
<td>0.000719</td>
</tr>
<tr>
<td>Score (logrank) test</td>
<td>17.7</td>
<td>3</td>
<td>0.000517</td>
</tr>
</tbody>
</table>

```r
> d <- datadist(BCellArkansasindex)
> options(datadist="d", width=150)
> BCellif <- cph(metadataArkansas$EFS ~ rcs(BCellArkansasindex, n.knots))
> plot(BCellif, xlab="Fitted BCell resistance index")
> title("Arkansas \n BCell - EFS Cox Proportional Hazards")
> legend("bottomright", ",", bty = "n",
>   title = paste("P-value = ",
>     as.character(signif(unlist(BCellif)$stats.P, 1)),
>     sep=""))
> dev.off()
```

**Figure 14.4:** The log relative hazard as a function of the BCell resistance index. The P-value is the likelihood ratio test for no RCS-association between log relative hazard and resistance. The dashed lines represent 95% confidence intervals.
14.2 Predicting Melphalan Resistance in the *Hummel* Data

Firstly, the expression data are extracted from the expression set.

```r
> hummel.matrix <- exprs(GEPHummel)
```

Next, the probes which were removed by `nsFilter` with `var.cutoff` set equal to 0.74 are discarded for the matrix.

```r
> BCellHummelTest <- hummel.matrix[featureNames(BCell.filtered),]
```

Finally, the signature is used to predict the patients resistance index.

```r
> BCellHummelindex <- as.vector(predict(BCell.fit.spls,
                        newx = t(BCellHummelTest)))
```

14.2.1 Association between the Resistance Index and OS

Kaplan-Meier Survival Curves

Kaplan-Meier survival curves are made with the patients grouped into sensitive, intermediate and resistant. The 25% with the lowest predicted melphalan resistance index are categorized as sensitive and the 25% with the highest predicted melphalan resistance index are categorized as resistant. The remaining subjects are characterised as having intermediate resistance.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 14.5.

```r
> pdf(file.path(figure.output, "BCellhummelOSKMplot.pdf"))
> BCellhummelOSKM.P <-
  PlotKM(BCellHummelindex, metadataHummel$OS,
       cut.points = cut.points,
       xlab = "Time (months)",
       ylab = "OS ratio",
       xmax = 110,
       main = "Hummel \n BCell Kaplan-Meier OS curves"
  )
> dev.off()
```
Figure 14.5: Kaplan-Meier survival curves based on the BCell resistance index. The logrank test comparing the survival curves results in a P-value of 0.08.

Cox Proportional Hazards

A Cox proportional hazards model is fitted where the association between log relative hazard and the resistance index is modelled by a spline with four knots.

```r
> n.knots <- 4
> coxfit.os <- coxph(metadataHummel$OS ~ rcs(BCellHummelindex, n.knots))
```

Table 14.3 summarizes three tests for no association between log relative hazard and the predicted resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>2.45</td>
<td>0.484</td>
</tr>
<tr>
<td>Wald test</td>
<td>2.81</td>
<td>0.423</td>
</tr>
<tr>
<td>Score (log rank) test</td>
<td>2.89</td>
<td>0.409</td>
</tr>
</tbody>
</table>

Table 14.3: A summary of three tests for no association between log relative hazard and the predicted resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the BCell resistance index. The result is depicted in Figure 14.6.
> pdf(file.path(figure.output,"BCellhummelOSPredictors.pdf"))
> par(mfrow=c(1,1))
> d <- datadist(BCellHummelindex)
> options(datadist="d", width=150)
> BCellf <- cph(metadataHummel$OS ~ rcs(BCellHummelindex,4))
> plot(BCellf,
    xlab="Fitted BCell resistance index")
> title("Hummel \n BCell - OS Cox Proportional Hazards")
> legend("bottomright", "", bty = "n",
    title = paste("P-value = ",
        as.character(signif(unlist(BCellf)$stats.P, 1)),
        sep = "))
> dev.off()

![Graph showing log relative hazard as a function of BCell resistance index. The P-value is the likelihood ratio test for no RCS-association between log Relative Hazard and resistance. The dashed lines represent 95% confidence intervals.]

**Figure 14.6:** This figure shows the log relative hazard as a function of the BCell resistance index. The P-value is the likelihood ratio test for no RCS-association between log Relative Hazard and resistance. The dashed lines represent 95% confidence intervals.

### 14.3 Regularisation Paths for the BCell Panel

Set seed for repeatability

> set.seed(1000)

Settings for the repeated datasets
> cutoff <- cutoff
> newGEP.filtered <- nsFilter(GEPBCCell133a,
    var.cutoff = cutoff)$eset
> newGEP.matrix <- t(scale(t(exprs(newGEP.filtered))))
> top.correlated <- 100
> n.chosen <- 20

> genchem.list <- corFilter(exprs.matrix = newGEP.matrix,
    chemoindex = BCellResistanceindex,
    alpha = 0.85)

> n.repeats <- 100

> BCell.reg.path.res <- data.frame(sens = rep(0, n.repeats),
    spec = rep(0, n.repeats),
    fdr = rep(0, n.repeats),
    eta = rep(0, n.repeats),
    K = rep(0, n.repeats))

Perform the repeated analyses.

> BCell.reg.path.res.files <- file.path(BCell.gen.dir, "BCell.reg.path.res.Rdata")
> if(file.exists(BCell.reg.path.res.files)){
    load(BCell.reg.path.res.files)
} else{
    for(reg.n in 1:n.repeats){
        print(reg.n)

        fixed.list <- sample(rownames(genchem.list[1 : top.correlated, ]),
            n.chosen, replace=FALSE)
        name.list <- rownames(newGEP.matrix)
        perm.list <- setdiff(name.list, fixed.list)

        sample.list <- sample(1:dim(newGEP.matrix)[2], replace=FALSE)
        perm.matrix <- rbind(newGEP.matrix[fixed.list, ],
            newGEP.matrix[perm.list, sample.list])

        eta <- seq(0.2, 0.99, length.out = 200)
        K <- c(1, 2, 3)
        fit.spls.cv <-
            cv.spls(x = t(perm.matrix),
                y = BCellResistanceindex,
                fold = n.BCell, eta = eta, K = K,
                ...)
plot.it = FALSE,
do.SIS = TRUE,
pval = pval,
adjust = adjust)

\( \eta_{\text{opt}} \) <- \text{fit.spls.cv$\eta$}_{\text{opt}}
\( K_{\text{opt}} \) <- \text{fit.spls.cv$K$}_{\text{opt}}

\( B\text{Cell.reg.path.res$\eta$}[\text{reg.n}] \) <- \( \eta_{\text{opt}} \)
\( B\text{Cell.reg.path.res$K$}[\text{reg.n}] \) <- \( K_{\text{opt}} \)

c\text{coef.container} \leftarrow \text{matrix}(0, ncol=\text{length}(\eta), nrow = 
\text{length(perm.matrix[, i]))}
\text{colnames(coef.container)} \leftarrow \text{rownames(fit.spls.cv$mspemat)}
\text{rownames(coef.container)} \leftarrow \text{rownames(perm.matrix)}

colidx <- 1
\text{for}(i in \eta)\{ 
  \text{print}(i) 

  \text{curr.fit.spls} \leftarrow \text{spls(x = t(perm.matrix), 
    y = BCellResistanceindex, 
    K = K_{\text{opt}}, 
    \eta = i, 
    do.SIS = TRUE, 
    pval = pval, 
    adjust = adjust)}

  \text{coefi} \leftarrow \text{coef(curr.fit.spls)}
  \text{coef.container[row.names(coefi), colidx]} \leftarrow \text{coefi}

  \text{colidx} \leftarrow \text{colidx} + 1 
\}

\## \text{Sensitivity and specificity}
\text{all.coef} \leftarrow \text{as.vector(coef.container[, \eta == \eta_{\text{opt}}])}
\text{all.pos} \leftarrow \text{as.vector(coef.container[fixed.list, \eta == \eta_{\text{opt}}])}

\text{n.truepos} \leftarrow \text{length(all.pos[all.pos != 0])}
\text{n.falsepos} \leftarrow \text{length(all.coef[all.coef != 0])} - \text{n.truepos}

\## \text{Sensitivity}
BCell.reg.path.res$sens[reg.n] <- n.truepos / n.chosen

## Specificity

BCell.reg.path.res$spec[reg.n] <-
  (length(coef.container[, 1]) - n.falsepos) / length(coef.container[, 1])

## False discovery rate

BCell.reg.path.res$fdr[reg.n] <- 1 - n.truepos / length(all.coef[all.coef != 0])

print(BCell.reg.path.res)

}
save(BCell.reg.path.res, coefi, fixed.list, coef.container,
  file = BCell.reg.path.res.files)

The resulting regularisation paths of the last simulation are shown in Figure 14.7.

> ts.col <- rep("darkgrey", dim(coef.container)[1])
> names(ts.col) <- rownames(coef.container)
> ts.type <- rep(2, dim(coef.container)[1])
> names(ts.type) <- rownames(coef.container)
> ts.col[fixed.list] <- "black"
> ts.type[fixed.list] <- 1
> sparse.idx <- apply(coef.container, 1, sum) != 0
> sparse.container <- coef.container[sparse.idx, ]
> ts.col <- ts.col[sparse.idx]
> ts.type <- ts.type[sparse.idx]

> all.coef <- as.vector(coef.container[ , eta ==
  BCell.reg.path.res$eta[nrow(BCell.reg.path.res)]]))
> all.pos <- as.vector(coef.container[fixed.list, eta ==
  BCell.reg.path.res$eta[nrow(BCell.reg.path.res)]]))
> n.truepos <- length(all.pos [all.pos != 0])
> n.falsepos <- length(all.coef[all.coef != 0]) - n.truepos

> pdf(file.path(figure.output,"regularizationpath.pdf"))
> eta <- seq(0.2, 0.99, length.out = 200)
> plot(eta, sparse.container[1, ],
>       type = "n",
>       xlab = expression(eta),

104
ylab = "Weight",
main = "Regularization Path",
xlim = c(0.65, 1),
ylim = c(min(sparse.container), max(sparse.container)))

> for(i in 1:length(sparse.container[, 1])) {
  if (ts.col[i] == "darkgrey")
    lines(eta, sparse.container[i, ],
           col = ts.col[i],
           lty = ts.type,
           lwd = 2)
}

> for(i in 1:length(sparse.container[, 1])) {
  if (ts.col[i] == "black")
    lines(eta, sparse.container[i, ],
           col = ts.col[i],
           lty = ts.type,
           lwd = 2)
}

> legend("topleft",
       legend = c("True", "False"),
       title = "Probesets",
       bty = "n",
       lty = c(1, 1),
       col = c( "black", "darkgrey"),
       lwd = 2)

> dev.off()

Summary of the simulations are calculated and summarised in Table 14.4

> accuracy <- matrix(NA, nrow = 2, ncol = 5)
> accuracy[1, ] <- round(apply(BCell.reg.path.res, 2, mean), 3)
> accuracy[2, ] <- round(apply(BCell.reg.path.res, 2, sd) / 
                         sqrt(n.repeats - 1), 3)
<table>
<thead>
<tr>
<th>Accuracy</th>
<th>sens</th>
<th>spec</th>
<th>fdr</th>
<th>eta</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.536</td>
<td>0.987</td>
<td>0.673</td>
<td>0.800</td>
<td>2.370</td>
</tr>
<tr>
<td>SD</td>
<td>0.039</td>
<td>0.002</td>
<td>0.024</td>
<td>0.012</td>
<td>0.071</td>
</tr>
</tbody>
</table>

Table 14.4: Summary of the repeated simulations.

Figure 14.7: Regularization paths for the optimal number of partial least squares component $K = 3$ and regularization parameter $\eta = 0.827$ are shown. The true probesets have black lines and the false probesets are illustrated with grey lines. The sensitivity is 0.55, the specificity is 0.994 and the false discovery rate is 0.633.
15 Simple Analysis

In this section a resistance index based on the two cell lines MOLP-2 and RPMI-8226 LR5 is developed. Firstly, the duplicated Entrez Ids and control probes are removed using the nsFilter function var.cutoff set to 0.001.

> influential.data <- nsFilter(GEPBCell1133a, var.cutoff = 0.001)$eset

The gene expressions of the two cell lines are extracted and stored in the object influential.matrix.

> influential.data <- influential.data[, c("MOLP-2", "RPMI-8226 LR5")]
> influential.matrix <- exprs(influential.data)

The data are sorted according to differences in the gene expressions between the two cell lines.

> diff.list <- apply(influential.matrix, 1, diff)
> sort.diff.list <- sort(diff.list)
> n.greatest <- 50

and the 100 most differentially expressed genes are used to establish the gene expression signature.

> len.list <- length(sort.diff.list)
> most.diff.exprs <-
  sort.diff.list[c(1:n.greatest, (len.list - n.greatest + 1):len.list)]
> names.most <- names(most.diff.exprs)
> save(most.diff.exprs,
  file=file.path(BCell.gen.dir, "BCell.fit.simple.Rdata"))

The gene expressions are summarized in Table 15.3.

15.1 Predicting Melphalan Resistance in the Arkansas data

A melphalan resistance index is constructed by means of the gene expressions established in the previous section and used to predict the resistance index of the patients in the Arkansas data. Notice, the absolute difference is used as weight.

> BCellArkansasTest <-
  arkansas.matrix[names.most,]
> BCellArkansasindex <- my.predict(most.diff.exprs, BCellArkansasTest)[1,]
15.1.1 Association between the Melphalan Resistance Index and OS

Kaplan-Meier Survival Curves

Kaplan-Meier survival curves are constructed for the sensitive, intermediate and resistant groups. The 25% with the lowest predicted resistance index are categorized as sensitive and the 25% with the highest predicted melphalan resistance index are categorized as resistant. The remaining subjects are characterised as having intermediate resistance.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 15.1.

```r
> pdf(file.path(figure.output, "BCellarkansasOSKMplot_simple.pdf"))
> BCellarkansasOSKM_simple.P <-
  PlotKM(BCellArkansasindex, metadataArkansas$OS,
  cut.points = cut.points,
  xlab = "Time (months)",
  ylab = "OS ratio",
  xmax = 110,
  main = "Arkansas \n Kaplan–Meier OS curves")
> dev.off()
```

Figure 15.1: Kaplan-Meier survival curves based on the two cell lines resistance index. The logrank test comparing the survival curves results in a P-value of 0.004.
Cox Proportional Hazards

A Cox proportional hazards model is fitted were the association between log relative hazard and the two cell line resistance index is modelled by a spline with four knots.

```r
> n.knots <- 4
> coxfit.os <- coxph(metadataArkansas$OS ~ rcs(BCellArkansasindex, n.knots))
```

Table 15.1 summarizes three tests for no association between log relative hazard and the predicted resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>7.41</td>
<td>0.0599</td>
</tr>
<tr>
<td>Wald test</td>
<td>8.06</td>
<td>0.0447</td>
</tr>
<tr>
<td>Score (logrank) test</td>
<td>8.28</td>
<td>0.0406</td>
</tr>
</tbody>
</table>

Table 15.1: A summary of three tests for no association between log relative hazard and the predicted resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the two cell line resistance index. The result is depicted in Figure 15.2.

```r
> pdf(file.path(figure.output, "BCellarkansasOSPredictors_simple.pdf"))
> par(mfrow=c(1,1))
> d <- datadist(BCellArkansasindex)
> options(datadist="d", width=150)
> BCellf <- cph(metadataArkansas$OS ~ rcs(BCellArkansasindex, n.knots))
> plot(BCellf,
>      xlab="Fitted resistance index")
> title("Arkansas - OS Cox Proportional Hazards")
> legend("bottomright","",bty="n",
>        title=paste("P-value = ",
>                      as.character(signif(unlist(BCellf)$stats.P,1))," ",sep=""))
> dev.off()
```
Figure 15.2: This figure shows the log relative hazard as a function of the two cell line resistance index. The P-value is the likelihood ratio test for no RCS-association between log relative hazard and resistance. The dashed lines represent 95% confidence intervals.

15.1.2 Association between the Melphalan Resistance Index and EFS

Kaplan-Meier Survival Curves

Similarly to the previous section Kaplan-Meier survival curves are made with the data grouped into sensitive, intermediate and resistant, however, EFS is used as endpoint instead of OS.

The Kaplan-Meier survival curves are plotted with the code chunk below and the result is shown in Figure 15.3.

```r
> pdf(file.path(figure.output, "BCellarkansasEFSKMPplot_simple.pdf"))
> BCellarkansasEFSKMP_plot_simple.P <-
  PlotKM(BCellArkansasindex,metadataArkansas$EFS,
    cut.points = cut.points,
    xlab = "Time (months)",
    ylab = "EFS ratio",
    xmax = 110,
    main = "Arkansas \n Kaplan-Meier EFS curves")
> dev.off()
```
Figure 15.3: Kaplan-Meier survival curves based on the two cell lines resistance index. The logrank test comparing the survival curves results in a P-value of 6e-04.

Cox Proportional Hazards

A Cox proportional hazards model is fitted with the two cell line resistance index modelled using a spline with four knots.

\[
\begin{aligned}
&> n.knots <- 4 \\
&> coxfit.os <- coxph(metadataArkansas$EFS ~ rcs(BCellArkansasindex, 4))
\end{aligned}
\]

Table 15.2 summarizes three tests for no association between log relative hazard and the predicted resistance index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test D.o.F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood ratio test</td>
<td>12.4</td>
<td>0.00620</td>
</tr>
<tr>
<td>Wald test</td>
<td>12.7</td>
<td>0.00533</td>
</tr>
<tr>
<td>Score (log rank) test</td>
<td>13.1</td>
<td>0.00449</td>
</tr>
</tbody>
</table>

Table 15.2: A summary of three tests for no association between log Relative Hazard and the predicted resistance index.

The code chunk below produces a plot of the log relative hazard as a function of the two cell line resistance index. The result is depicted in Figure 15.4.
> pdf(file.path(figure.output, "BCellArkansasEFSPredictors_simple.pdf"))
> par(mfrow=c(1,1))
> d <- datadist(BCellArkansasindex)
> options(datadist="d", width=150)
> BCellf <- cph(metadataArkansas$EFS ~ rcs(BCellArkansasindex, n.knots))
> plot(BCellf,
  xlab="Fitted resistance index")
> title("Arkansas BCell - EFS Cox Proportional Hazards")
> legend("bottomright","",bty="n",
  title=paste("P-value = ",
               as.character(signif(unlist(BCellf)$stats.P,1))," ",sep=""))
> dev.off()

Figure 15.4: This figure shows the log relative hazard as a function of the two cell line resistance index. The P-value is the likelihood ratio test for no RCS-association between log relative hazard and resistance. The dashed lines represent 95% confidence intervals.
Finally, a table is created with the most differentially expressed genes, containing gene symbols and weights.

```r
> genes.most <- as.vector(unlist(lookUp(names.most, "hgu133plus2", "SYMBOL")))
> xx <- matrix(NaN, nrow = n.greatest, ncol = 6)
> xx[, c(1, 4)] <- "as.vector<"(xx[, c(1, 4)],
    gsub("_", "\\textunderscore ", names.most))
> xx[, c(2, 5)] <- "as.vector<"(xx[, c(2, 5)], genes.most)
> xx[, c(3, 6)] <- "as.vector<"(xx[, c(3, 6)], round(most.diff.exprs, 3))
> colnames(xx) <- rep(c("U133a ID", "Gene Symbol", "Weight"), 2)
> latex.default(xx, title = "Cell line",
    file = "",
    booktabs = TRUE,
    ctable = FALSE,
    longtable = TRUE,
    lines.page = Inf,
    table.env = TRUE,
    digits = 3,
    label = "tab:influentialnames",
    caption = paste("Table of the gene expressions used in the two cell line resistance index."))
```

### Table 15.3: Table of the gene expressions used in the two cell line resistance index.

<table>
<thead>
<tr>
<th>U133a ID</th>
<th>Gene Symbol</th>
<th>Weight</th>
<th>U133a ID</th>
<th>Gene Symbol</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>205174_s_at</td>
<td>QPCT</td>
<td>-7.893</td>
<td>210942_s_at</td>
<td>ST3GAL6</td>
<td>5.596</td>
</tr>
<tr>
<td>205590_at</td>
<td>RASGRP1</td>
<td>-7.89</td>
<td>215189_at</td>
<td>KRT86</td>
<td>5.599</td>
</tr>
<tr>
<td>213831_at</td>
<td>HLA-DQA1</td>
<td>-7.591</td>
<td>212473_s_at</td>
<td>MICAL2</td>
<td>5.687</td>
</tr>
<tr>
<td>212122_at</td>
<td>RHOQ</td>
<td>-7.303</td>
<td>209735_at</td>
<td>ABCG2</td>
<td>5.69</td>
</tr>
<tr>
<td>210587_at</td>
<td>INHBE</td>
<td>-7.032</td>
<td>218502_s_at</td>
<td>TRPS1</td>
<td>5.717</td>
</tr>
<tr>
<td>39248_at</td>
<td>AQP3</td>
<td>-6.988</td>
<td>214023_s_at</td>
<td>TUBB2B</td>
<td>5.721</td>
</tr>
<tr>
<td>205990_s_at</td>
<td>WNT5A</td>
<td>-6.854</td>
<td>212724_at</td>
<td>RND3</td>
<td>5.752</td>
</tr>
<tr>
<td>202609_at</td>
<td>EPS8</td>
<td>-6.797</td>
<td>203474_at</td>
<td>IQGAP2</td>
<td>5.767</td>
</tr>
<tr>
<td>219159_s_at</td>
<td>SLAMF7</td>
<td>-6.793</td>
<td>204014_at</td>
<td>DUSP4</td>
<td>5.772</td>
</tr>
<tr>
<td>213938_at</td>
<td>ERC2</td>
<td>-6.724</td>
<td>213524_s_at</td>
<td>G0S2</td>
<td>5.774</td>
</tr>
<tr>
<td>204118_at</td>
<td>CD48</td>
<td>-6.706</td>
<td>217906_at</td>
<td>PHLDA1</td>
<td>5.797</td>
</tr>
<tr>
<td>207191_s_at</td>
<td>ISLR</td>
<td>-6.422</td>
<td>206897_at</td>
<td>PAGE1</td>
<td>5.824</td>
</tr>
<tr>
<td>205801_s_at</td>
<td>RASGRP3</td>
<td>-6.387</td>
<td>218736_s_at</td>
<td>PALMD</td>
<td>5.835</td>
</tr>
<tr>
<td>212998_s_at</td>
<td>HLA-DQB1</td>
<td>-6.351</td>
<td>204734_at</td>
<td>KRT15</td>
<td>5.978</td>
</tr>
<tr>
<td>212063_at</td>
<td>CD44</td>
<td>-6.321</td>
<td>214039_s_at</td>
<td>LAPTMM4B</td>
<td>6.004</td>
</tr>
<tr>
<td>220132_s_at</td>
<td>CLEC2D</td>
<td>-6.251</td>
<td>204533_at</td>
<td>CXCL10</td>
<td>6.053</td>
</tr>
<tr>
<td>221558_s_at</td>
<td>LEF1</td>
<td>-6.186</td>
<td>205098_at</td>
<td>CCR1</td>
<td>6.107</td>
</tr>
<tr>
<td>219073_s_at</td>
<td>OSBPPL10</td>
<td>-6.165</td>
<td>221698_s_at</td>
<td>CLEC7A</td>
<td>6.122</td>
</tr>
</tbody>
</table>
Table 15.3: (continued)

<table>
<thead>
<tr>
<th>U133a ID</th>
<th>Gene Symbol</th>
<th>Weight</th>
<th>U133a ID</th>
<th>Gene Symbol</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>219667_s_at</td>
<td>BANK1</td>
<td>-6.147</td>
<td>202551_s_at</td>
<td>CRIM1</td>
<td>6.174</td>
</tr>
<tr>
<td>204971_at</td>
<td>CSTA</td>
<td>-6.137</td>
<td>203666_s_at</td>
<td>CXCL12</td>
<td>6.201</td>
</tr>
<tr>
<td>219049_at</td>
<td>CSGALNACT1</td>
<td>-6.13</td>
<td>204006_s_at</td>
<td>AGAP1</td>
<td>6.228</td>
</tr>
<tr>
<td>201998_at</td>
<td>ST6GAL1</td>
<td>-5.816</td>
<td>44790_s_at</td>
<td>CI3orf18</td>
<td>6.246</td>
</tr>
<tr>
<td>206150_at</td>
<td>CD27</td>
<td>-5.799</td>
<td>204105_s_at</td>
<td>NRCAM</td>
<td>6.259</td>
</tr>
<tr>
<td>211990_at</td>
<td>HLA-DPA1</td>
<td>-5.789</td>
<td>220415_s_at</td>
<td>TNN13K</td>
<td>6.366</td>
</tr>
<tr>
<td>219926_at</td>
<td>POPDC3</td>
<td>-5.752</td>
<td>209803_s_at</td>
<td>PHLD1A2</td>
<td>6.427</td>
</tr>
<tr>
<td>208683_at</td>
<td>CAPN2</td>
<td>-5.736</td>
<td>202350_s_at</td>
<td>MATN2</td>
<td>6.463</td>
</tr>
<tr>
<td>207734_at</td>
<td>LAX1</td>
<td>-5.673</td>
<td>203917_s_at</td>
<td>CXADR</td>
<td>6.479</td>
</tr>
<tr>
<td>220129_at</td>
<td>SOHLH2</td>
<td>-5.534</td>
<td>207979_s_at</td>
<td>CD8B</td>
<td>6.512</td>
</tr>
<tr>
<td>202252_at</td>
<td>RAB13</td>
<td>-5.527</td>
<td>200606_s_at</td>
<td>DSP</td>
<td>6.739</td>
</tr>
<tr>
<td>205671_s_at</td>
<td>HLA-DOB</td>
<td>-5.512</td>
<td>212599_at</td>
<td>AUTS2</td>
<td>6.791</td>
</tr>
<tr>
<td>200999_s_at</td>
<td>CKAP4</td>
<td>-5.435</td>
<td>202677_at</td>
<td>RASA1</td>
<td>6.806</td>
</tr>
<tr>
<td>213638_at</td>
<td>PHACTR1</td>
<td>-5.405</td>
<td>201445_at</td>
<td>CNN3</td>
<td>6.812</td>
</tr>
<tr>
<td>202746_at</td>
<td>ITM2A</td>
<td>-5.381</td>
<td>204151_x_at</td>
<td>AKR1C1</td>
<td>6.908</td>
</tr>
<tr>
<td>209942_x_at</td>
<td>MAGEA3</td>
<td>-5.351</td>
<td>204485_s_at</td>
<td>TOMIL1</td>
<td>6.909</td>
</tr>
<tr>
<td>203708_at</td>
<td>PDE4B</td>
<td>-5.323</td>
<td>219181_s_at</td>
<td>LIPG</td>
<td>6.931</td>
</tr>
<tr>
<td>209366_x_at</td>
<td>CYB5A</td>
<td>-5.277</td>
<td>212192_at</td>
<td>KCTD12</td>
<td>7.011</td>
</tr>
<tr>
<td>203895_at</td>
<td>PLCB4</td>
<td>-5.25</td>
<td>221210_s_at</td>
<td>NPL</td>
<td>7.063</td>
</tr>
<tr>
<td>213170_at</td>
<td>GPX7</td>
<td>-5.208</td>
<td>205549_at</td>
<td>PCP4</td>
<td>7.135</td>
</tr>
<tr>
<td>221297_at</td>
<td>GPRC5D</td>
<td>-5.158</td>
<td>209699_x_at</td>
<td>AKR1C2</td>
<td>7.169</td>
</tr>
<tr>
<td>201925_s_at</td>
<td>CD55</td>
<td>-5.155</td>
<td>201667_at</td>
<td>GJA1</td>
<td>7.485</td>
</tr>
<tr>
<td>221571_at</td>
<td>TRAF3</td>
<td>-5.155</td>
<td>210095_s_at</td>
<td>IGFBP3</td>
<td>7.581</td>
</tr>
<tr>
<td>219371_s_at</td>
<td>KLF2</td>
<td>-5.132</td>
<td>206336_at</td>
<td>CXCL6</td>
<td>7.688</td>
</tr>
<tr>
<td>205884_at</td>
<td>ITGA4</td>
<td>-5.09</td>
<td>205898_at</td>
<td>CX3CR1</td>
<td>7.769</td>
</tr>
<tr>
<td>201952_at</td>
<td>ALCAM</td>
<td>-5.018</td>
<td>212094_at</td>
<td>PEG10</td>
<td>7.778</td>
</tr>
<tr>
<td>206609_at</td>
<td>MAGEC1</td>
<td>-5.014</td>
<td>201427_s_at</td>
<td>SEPP1</td>
<td>8.534</td>
</tr>
<tr>
<td>203397_s_at</td>
<td>GALNT3</td>
<td>-4.979</td>
<td>209348_s_at</td>
<td>MAF</td>
<td>8.855</td>
</tr>
<tr>
<td>202947_s_at</td>
<td>GYPC</td>
<td>-4.966</td>
<td>209395_s_at</td>
<td>CHI3L1</td>
<td>9.056</td>
</tr>
<tr>
<td>205632_s_at</td>
<td>PIP5K1B</td>
<td>-4.953</td>
<td>209160_at</td>
<td>AKR1C3</td>
<td>9.105</td>
</tr>
<tr>
<td>210889_s_at</td>
<td>FCGR2B</td>
<td>-4.935</td>
<td>205336_at</td>
<td>PVALB</td>
<td>9.142</td>
</tr>
<tr>
<td>209619_at</td>
<td>CD74</td>
<td>-4.926</td>
<td>211719_x_at</td>
<td>FN1</td>
<td>9.776</td>
</tr>
</tbody>
</table>
16 Auxiliary Functions

16.1 Dose Response Functions

Function for extracting members of x which are not present in y.

```r
> "%w/o%" <- function(x, y) x[!x %in% y]
>
> "as.vector<-" <- function(xx, y){
  d <- nrow(xx) * ncol(xx) - length(y)
  if(d != 0){y <- c(y, rep("", d))}
  h <- 0:(ncol(xx) - 1) * nrow(xx) + 1
  h2 <- c(h[ -1] - 1, length(y))
  for(i in 1:ncol(xx)){
    xx[,i] <- y[h[i]:h2[i]]
  }
  return(xx)
}
```

NCI60 dose extraction.

```r
> getNSC <- function(filename, target, path = NULL) {
## Eat the extra final spaces
  fixup <- function(x) {
    x <- as.character(x)
    x <- sub(" $", "", x)
    factor(x)
  }
  ## Load the data and clean it up
  if(!is.null(path)) {
    filename <- file.path(path, filename)
  }
  rt <- read.table(filename,
                   sep=',', header = TRUE, row.names = NULL,
                   quote = '', comment.char = '')
  rt$NSC <- factor(rt$NSC)
  rt$LCONC <- factor(rt$LCONC)
  temp <- as.character(rt$STDDEV)
  temp <- gsub(" ", "", temp)
  temp[temp=="."] <- NA
  rt$STDDEV <- as.numeric(temp)
  rt$PANEL <- fixup(rt$PANEL)
  rt$CELL <- fixup(rt$CELL)
  rt[, "Key"] <- factor(paste(as.character(rt$NSC),
```
as.character(rt$LCONC), sep='L'))

len <- table(rt$CELL, rt$Key)

# Use groupedData and gsummary to average over meaningless replicates
form <- dummy ~ 1|NSC/LCONC/CELL
form[[2]] <- as.name(target)
gd <- groupedData(form, data = rt)
simple <- gsummary(gd, mean, na.rm = TRUE)
result <- tapply(simple[, target], list(simple$CELL, simple$Key),
mean, na.rm = TRUE)
result

}  

Loading .DBF files into R

> readDBFMeta <- function(path = getwd(), metadata, remove.edge = TRUE){

metadata$h2 <- paste(metadata$Name, metadata$Drug.s., sep=":")
h4 <- (aggregate(rep(1, length(metadata$Drug.s.)),
metadata[, c("Name", "Drug.s.")], sum))
for(i in metadata$h2){
metadata$platerep[metadata$h2==i] <-
1:h4$x[paste(h4$Name, h4$Drug.s., sep=":"==i]
}

chemoIndex <- vector()
celllines <- vector()
drug <- vector()
rep <- vector()
content <- vector()
colnum <- vector()
wellnum <- vector()
plate <- vector()
plateset <- vector()
time <- vector()
platerep <- vector()
disease <- vector()
for(i in 1:length(metadata[metadata$Drug.s.])){

    file0 <- metadata$Data.file.day.1[i]
    file48 <- metadata$Data.file.day.3[i]

    ## Plate 1 is read into R

116
cell0 <- read.dbf(paste(path, "/", file0, ".dbf", sep=""))
cell0 <- cell0[-(1:2),]
cell0$COLNUM <- factor(substr(cell0$WELLNUM, 2, 3))
cell0$ROW <- factor(substr(cell0$WELLNUM, 1, 1))
cell0 <- cell0[cell0$COLNUM != "01" &
                  cell0$COLNUM != "12",]
if(remove.edge == TRUE){
    cell0 <- cell0[cell0$ROW != "A" &
                   cell0$ROW != "H",]
}
cell0 <- cell0[order(cell0$CONTENT),]
if(remove.edge == TRUE){
    cell0$REP <- rep(1:3, 20)
}else{
    cell0$REP <- rep(1:4, 20)
}

## Plate 2 is read into R

cell48 <- read.dbf(paste(path, "/", file48, ".dbf", sep=""))
cell48 <- cell48[-(1:2),]
cell48$COLNUM <- factor(substr(cell48$WELLNUM, 2, 3))
cell48$ROW <- factor(substr(cell48$WELLNUM, 1, 1))
cell48 <- cell48[cell48$COLNUM != "01" &
                  cell48$COLNUM != "12",]
if(remove.edge == TRUE){
    cell48 <- cell48[cell48$ROW != "A" &
                   cell48$ROW != "H",]
}
cell48 <- cell48[order(cell48$CONTENT),]
if(remove.edge == TRUE){
    cell48$REP <- rep(1:3, 20)
}else{
    cell48$REP <- rep(1:4, 20)
}

n <- length(cell0 [, "M1"])
m <- length(cell48[, "M1"])

chemoIndex <- c(chemoIndex, cell0[, "M1"], cell48[, "M1"], celllines <- c(celllines, rep(paste(metadata$Name[i]), n + m))
drug <- c(drug, rep(paste(metadata$Drug.s.[i]), n + m))
disease <- c(disease, rep(paste(metadata$Disease[i]), n + m))
rep <- c(rep, cell0[, "REP"], cell48[, "REP"])
content <- c(content, as.character(cell0[, "CONTENT"], colnum <- c(colnum, as.character(cell0[, "COLNUM"]),
            as.character(cell48[, "CONTENT"]))
time <- c(time, rep(1, n), rep(3, m))
wellnum <- c(wellnum, as.character(cell0[, "WELLNUM"], as.character(cell48[, "WELLNUM"])))
plate <- c(plate, rep(paste(i * 2 - 1), n), rep(i * 2, m))
plateset <- c(plateset, rep(i, (n + m)))
platerep <- c(platerep, rep(metadata$platerep[i], (n + m)))

}

chemoIndex <- data.frame(index = chemoIndex, drug = drug,
            name = celllines, rep = rep,
            wellnum = wellnum, content = content,
            colnum = colnum, time = time,
            plate = plate, plateset = plateset,
            platerep = platerep, disease = disease)

>
Expressing dose in molaer

> molaer <- function(x){
    log10(1000*x/305.2/1000000)
}

Calculate means using the bootstrap.

> meanBootstrap <- function(x){
    n <- length(x)
    b <- sample(x,n,replace=TRUE)
    mean(b)
}
> logistic <- function(x, a, b) {
    2*(10^(a+b*x)/(1+10^(a+b*x))) - 1
    2*exp(-a*x^2) - 1
}

Determine whether or not there are outliers.

> grubbsTestPValue <- function(X) {
    test <- grubbs.test(X)
    Grubbs.P.value <- test$p.value
    return(Grubbs.P.value)
}

Extract the replicates which need to be excluded.

> grubbsTestHL <- function(X) {
    test <- grubbs.test(X)
    x <- strsplit(test$alternative, " ")
    y <- as.numeric(x[[1]][3])
    return(y)
}

Perform Grubbs test on the data.

> addGrubbs <- function(X, p = 0.05, grubbs = 1) {
    grubbs.p.value <- aggregate(X$index,
                                 X[, c("plate", "content")],
                                 grubbsTestPValue)

    names(grubbs.p.value)[3] <- "grubbs.p.value"

    grubbs.hl <- aggregate(X$index,
                            X[, c("plate", "content")],
                            grubbsTestHL)

    names(grubbs.hl)[3] <- "grubbs.hl"

    X <- merge(X, grubbs.p.value, by = c("plate", "content"))
    X <- merge(X, grubbs.hl, by = c("plate", "content"))

    X$outlier <- ifelse(X$index == X$grubbs.hl &
                         X$grubbs.p.value < p, 1, 0)
}

119
excluded <- dim(X[X$outlier == 1, ])[1]
X2 <- X
X <- X[X$outlier == 0, ]

if(grubbs == 2){
  grubbs.p.value <- aggregate(X$index,
    X[, c("plate", "content")], grubbsTestPValue )
  names(grubbs.p.value)[3] <- "grubbs.p.value2"
  grubbs.hl <- aggregate(X$index,
    X[, c("plate", "content")], grubbsTestHL)
  names(grubbs.hl)[3] <- "grubbs.hl2"
  X <- merge(X, grubbs.p.value, by=c("plate", "content"))
  X <- merge(X, grubbs.hl, by=c("plate", "content"))
  X$outlier2 <- ifelse(X$index == X$grubbs.hl2 &
    X$grubbs.p.value2 < p, 1, 0)
  excluded <- c(excluded, dim(X[X$outlier2 == 1, ])[1])
}
if(grubbs == 1){X3 <- X}
return(list(X, excluded, X2, X3))

Function for background correction of the plates.

> backgroundCorrected <- function(X, Bootstrap = FALSE, bcormethod = "difference"){

  y <- vector()
  mean.all.b <- mean(X$index[X$outlier == 0 &
    X$content == "B"])

  if(Bootstrap == FALSE){
    for(i in unique(X$plate)){
      mean.b <- mean(X$index[X$outlier == 0 &
        X$content == "B" &
        X$plate == i])
      y <- c(y, mean.b)
    }
  }

  return(list(mean.all.b, y))
}
if(bcormethod == "difference"){
    y <- c(y, (X$index[X$plate == i] - mean.b))
} else{
    y <- c(y, (X$index[X$plate == i] / mean.b))
}
}

else{
    for(i in unique(X$plate)){
        mean.b <- meanBootstrap(X$index[X$outlier == 0 &
            X$content == "B" &
            X$plate == i])

        if(bcormethod == "difference"){
            y <- c(y, (X$index[X$plate == i] - mean.b))
        } else{
            y <- c(y, (X$index[X$plate == i] / mean.b))
        }
    }
}

X$backgroundCorrected <- y
X$backgroundCorrected[X$content == "B"] <- X$index[X$content == "B"]

return(X)
}

Ensure that the absorbance values lie within a certain interval.

> diffsetCorrection <- function(X, int.cor=c(0.2, 1.2), model = "R2"){
    X <- X[X$backgroundCorrected >= int.cor[1] & X$backgroundCorrected <= int.cor[2], ]

    if(model %in% c("D1", "R1")){
        for(i in unique(X$plateset)){
            diffset <- setdiff(X$content[X$plateset == i & X$time == 1],
                                X$content[X$plateset == i & X$time == 3])

            diffset2 <- setdiff(X$content[X$plateset == i & X$time == 3],
                                 X$content[X$plateset == i & X$time == 1])

            diffset <- c(diffset, diffset2)
        }
    }

    return(X)
}
X <- X[!(X$content %in% diffset & X$plateset == i), ]
}
}
return(X)
}

> D1 <- function(x, y, z) {
  res <- ifelse(x >= 0, x / (y-z[1]), x / z[1])
  return(res)
}

> GI <- function(X, model = "R2", MA = FALSE, Bootstrap = FALSE) {

  X <- X[X$content != "B", ]

  ## Calculate the mean or the replicated wells
  if(Bootstrap == FALSE) {
    X2 <- aggregate(X$backgroundCorrected, 
      X[, c("plate", "content")], mean)
  }else{
    X2 <- aggregate(X$backgroundCorrected, 
      X[, c("plate", "content")], meanBootstrap)
  }

  ## Merge the datasets
  X <- merge(X2, X, by=c("plate", "content"), all=T)

  ## Use only use one of the replicates
  X2 <- aggregate(X$rep, 
    X[, c("plate", "content")], min)

  X2 <- rename.vars(X2, "x", "keep")
  X <- merge(X2, X, by = c("plate", "content"), all = T)
  X <- X[X$keep == X$rep, ]

This function calculates the growth inhibition through various models. It sets the model to be used for calculating GI: MA uses moving average to smooth the curves giving a more stable result. If the Bootstrap options is TRUE the function performs bootstrap at the well level.
X$plate <- as.numeric(X$plate)

GI <- vector()
GIname <- vector()
drug <- vector()
plateset <- vector()
platerp <- vector()
disease <- vector()
t <- vector()
name <- vector()

X <- X[order(X$plate),]

for(i in unique(X$plateset)) {
    drug.i <- paste(X$drug[X$plateset == i][1])
    if(model == "D1"){
        X0 <- X$x[X$plateset == i & X$time == 1]
        names(X0) <- X$content[X$plateset == i & X$time == 1]
        X1 <- X$x[X$plateset == i & X$time == 3]
        names(X1) <- X$content[X$plateset == i & X$time == 3]
        diff <- X1 - X0[c("X1")]
        G1res <- D1(diff, X1[c("X1")], X0)
    }
    if(model == "D3"){
        X0 <- X$x[X$plateset == i & X$time == 1]
        names(X0) <- X$content[X$plateset == i & X$time == 1]
        X1 <- X$x[X$plateset == i & X$time == 3]
        names(X1) <- X$content[X$plateset == i & X$time == 3]
        x0 <- mean(X0[c(paste("X",1:6,sep = ""))])
        x1 <- mean(X1[c(paste("X",1:4,sep = ""))])
        diff <- X1 - x0
        G1res <- ifelse(diff >= 0, diff / (x1 - x0), diff / x0)
    }
    if(model == "D4"){
        sets <- unique(X$plateset[X$name == X$name[X$plateset == i]])
        X0 <- X$x[X$plateset == i & X$time == 1]
        names(X0) <- X$content[X$plateset == i & X$time == 1]
    }

}
X1 <- X$x[X$plateset == i & X$time == 3]
names(X1) <- X$content[X$plateset == i & X$time == 3]

# The first six are used as reference on plate 1
x0 <- mean(X0[c(paste("X",1:6,sep = ""))])
x1 <- mean(X1[c(paste("X",1:4,sep = ""))])
diff <- X1 - x0

# The first four are used as reference on plate 2
GIres <- ifelse(diff >= 0, diff / (x1 - x0), diff / x0)

if(model == "D2"){
X0 <- X$x[X$plateset == i & X$time == 1]
names(X0) <- X$content[X$plateset == i & X$time == 1]
X1 <- X$x[X$plateset == i & X$time == 3]
names(X1) <- X$content[X$plateset == i & X$time == 3]

diff <- X1 - X0[1]
GIres <- D1(diff, X1[c("X1")], X0)
}

if(model == "R1"){
X0 <- X$x[X$plateset == i & X$time == 1]
names(X0) <- X$content[X$plateset == i & X$time == 1]
X1 <- X$x[X$plateset == i & X$time == 3]
names(X1) <- X$content[X$plateset == i & X$time == 3]

diff <- X1/X0
GIres <- diff/diff[c("X1")]
}

if(model == "R2"){
X1 <- X$x[X$plateset == i & X$time == 3]
names(X1) <- X$content[X$plateset == i & X$time == 3]

diff <- X1
GIres <- X1/X1[c("X1")]
}

if(model == "R3"){
X1 <- X$x[X$plateset == i & X$time == 3]
names(X1) <- X$content[X$plateset == i & X$time == 3]

124
```r
diff <- X1
GIres <- X1/mean(X1[c(paste("X", 1:4, sep=""))])
}

GI <- c(GI, GIres)
h1 <- substr(names(diff[names(diff) != "B"]), 2, 10)
h1 <- as.numeric(h1) -1
GIname <- c(GIname, h1)
plateset <- c(plateset, rep(i, length(diff[names(diff) != "B"])))
platerep <- c(platerep, rep(paste(X$platerep[X$plateset == i][1]),
 length(diff[names(diff) != "B"))))
drug <- c(drug, rep(drug.i, length(diff[names(diff) != "B"))))
disease <- c(disease, rep(paste(X$disease[X$plateset == i][1]),
 length(diff[names(diff) != "B"))))
name <- c(name, rep(paste(X$name[X$plateset == i][1]),
 length(diff[names(diff) != "B"))))
t <- c(t, as.numeric(metadata[i, paste("C", h1, sep=""))])

X <- data.frame(name = name, drug = drug,
 plateset = plateset, platerep = platerep,
 disease = disease, GIname = GIname,
 GI = GI, t=t, t2 = log10(t))

X <- X[order(X$t), ]
X <- X[order(X$plateset), ]
X <- X[order(X$GIname != "O"), ]
X <- X[order(X$name), ]
X <- X[order(X$disease), ]
X <- X[order(X$drug), ]

X$name <- as.factor(as.character(X$name))
X$disease <- as.factor(as.character(X$disease))
X$drug <- as.factor(as.character(X$drug))

if(MA!=FALSE){
 X$trial <- paste(X$drug, X$name, X$platerep, sep=":")
ta <- tapply(X$GI, c(X$trial), pSMA)
names <- dimnames(ta)
MAGI <- vector()
for(i in (names[[1]])){
 MAGI <- c(MAGI, eval(parse(text=paste("ta", "", i,
 "", sep="")))))
}
```

This function is used for the establishment of the GI50 values by use of functions previously defined.

```r
> createGIData <- function(X = chemo.index, grubbs = 1,
  b.cor = TRUE, int.cor = FALSE,
  model = "R2", MA = FALSE,
  p = 0.01, Bootstrap = FALSE,
  log = FALSE, bcormethod = "difference"){

  pSMA <- function(x)
  {
    SMA(x, n=MA)
  }

  if(grubbs == 1){
    x <- addGrubbs(X, p = p, grubbs = 1)
    excluded <- x[[2]]
    X <- x[[1]]
  }

  if(grubbs == 2){
    x <- addGrubbs(X, p = p, grubbs = 2)
    excluded <- x[[2]]
    X <- x[[1]]
  }

  if(grubbs == FALSE){
    X$outlier <- 0
    excluded <- 0
  }

  #if(log == TRUE){X$index <- log(X$index)}
  if(b.cor == TRUE){
    X <- backgroundCorrected(X, Bootstrap = Bootstrap,
                            bcormethod = bcormethod)
  }else{
    X <- X[X$content!="B", ]
  }

  X$MAGI <- MAGI

  return(X)
}
```


if(int.cor != FALSE){
  X <- diffsetCorrection(X, int.cor = int.cor)
}

if(log == TRUE){X$index <- log(X$index)}

X <- GI(X, model = model, MA = MA, Bootstrap = Bootstrap)

return(list(X,excluded))

Function for extracting the GI\textsubscript{50} values

\begin{verbatim}
> findGI <- function(x, y, GI = 0.5){
  if(y[1] <= GI){
    res <- x[1]
  }else{
    res <- Inf
  }
  n <- length(y)
  i <- 1
  repeat{
    if(y[i + 1] < GI & i < (n-1)){
      a <- (y[i + 1] - y[i]) /(x[i + 1] - x[i])
      b <- - 1 * a * x[i] + y[i]
      res <- (GI - b) / a
      break()
    }
    i <- i + 1
  }
  res
}
\end{verbatim}

16.2 Model Building Functions

16.3 Functions used for the SPLS Analysis

The function SIS defined below performs Sure Independence Screening. The output of the function is the P-value for the null-hypothesis of no correlation. The P-values may be adjusted through the following methods:
holm, hochberg, hommel, bonferroni, BH, BY, fdr, none.

```r
> SIS <- function(exprs, index, adjust = p.adjust.methods){
    n <- dim(exprs)[2]
    p <- dim(exprs)[1]

    if(class(index) == "matrix"四周
        names <- row.names(index)
    }else{
        names <- names(index)
    }

    if(class(exprs) == "ExpressionSet"四周
        exprs <- exprs(exprs)
    }

    x <- t(scale(t(exprs)))
    y <- scale(index)

    sis.score <- (x %*% y)[, 1] / (n - 1)
    sis.cor <- sis.score / sqrt((1 - sis.score^2) / (n - 2))
    sis.pval <- 2 * pt(-1 * abs(sis.cor), n - 2)
    p.adjust <- p.adjust(sis.pval, method = adjust)

    return(p.adjust)
}
```

The function SISCV performs SIS in a way suitable for CV.

```r
> SISCV <- function(exprs, index, adjust = p.adjust.methods, fold = length(index), foldi = "gen"){
    n <- dim(exprs)[2]
    p <- dim(exprs)[1]

    if(class(index) == "matrix"){
        names <- row.names(index)
    }else{
        names <- names(index)
    }

    if(class(exprs) == "ExpressionSet"四周
        exprs <- exprs(exprs)
    }

    128
if(foldi[1] == "gen") { 
  foldi <- split(sample(1:n), rep(1:fold, length = n))
}

sis.cv <- matrix(0, ncol = fold, nrow = p)
sis.cv.adjust <- matrix(0, ncol = fold, nrow = p)

rownames(sis.cv) <- row.names(exprs)
rownames(sis.cv.adjust) <- row.names(exprs)

cname <- vector()

for (j in 1:fold) {
  cname[j] <- paste(names[foldi[[j]]], collapse = " ", "")
  omit <- foldi[[j]]
  x <- t(scale(t(exprs[, - omit])))
  y <- scale(index[ - omit])
  sis.cv[,j] <- SIS(exprs = x, index = y, adjust = "none")
  sis.cv.adjust[,j] <- SIS(exprs = x, index = y, adjust = adjust)
}

colnames(sis.cv) <- cname
colnames(sis.cv.adjust) <- cname
class(sis.cv) <- "SISCV"
class(sis.cv.adjust) <- "SISCV"
return(list(sis.cv, sis.cv.adjust))

> summary.SISCV <- function(SISCV, p.value = 0.05){
  nSignificant <- function(x){
    length(x[x <= p.value])
  }
  return(apply(SISCV, 2, nSignificant))
}

> View.SISCV <- function(SISCV){
  class(SISCV) <- "matrix"
  View(SISCV)
}

Construction of a filter function which only finds the correlation coefficient between genes and chemosensitivity, P-value, and FDR for the hypothesis of no correlation

> corFilter <- function(exprs.matrix, chemoindex, alpha=0.01){
  cor.filter <- function(x){
    fit <- cor.test(x, chemoindex)
    return(c(fit$estimate, fit$p.value))
  }

129
# FDR is per default set to 0.01
probelist <- genefilter(exprs.matrix, cor.filter)
probelistLength <- length(probelist[1,])
probelistNumber <- 1:probelistLength
probelist <- rbind(probelist, 0)
row.names(probelist) <- c("cor", "pValue", "reject")
probelist <- t(probelist)
probelist <- probelist[order(probelist[,2]),]
bool <- probelist[,2] - alpha * probelistNumber/probelistLength
probelist[,3] <- bool<=0
a <- 1:length(probelist[,3])
if(length(a[bool<=0])>0){
  b <- max(a[bool<=0])
  probelist[,3][1:b] <- 1
}
return(probelist)
}

In order to use the SPLS functions together with SIS the three functions `spls`, `cv.spls` and `predict.spls` from the package `spls` are recoded.

```r
> spls <- function (x, y, K, eta, kappa = 0.5, select = "pls2", fit = "simpls",
                   scale.x = TRUE, scale.y = FALSE, eps = 1e-04, maxstep = 100,
                   trace = FALSE, do.SIS = FALSE, pval = 0.05, adjust = p.adjust.methods)
{
  if(do.SIS){
    sis <- SIS(t(x), y, adjust = adjust)
    sis.names <- names(sis[sis <= pval])
    if(length(sis.names) < max(K)){
      sis.names <- names(sort(sis)[1:max(K)])
    }
    x <- x[, sis.names]
  }
  x <- as.matrix(x)
  n <- nrow(x)
  p <- ncol(x)
  ip <- c(1:p)
  y <- as.matrix(y)
  q <- ncol(y)
  one <- matrix(1, 1, n)
  mu <- one %% y/n
  y <- scale(y, drop(mu), FALSE)
  meanx <- drop(one %% x)/n
  x <- scale(x, meanx, FALSE)
```
if (scale.x) {
    normx <- sqrt(drop(one %*% (x^2))/(n - 1))
    if (any(normx < .Machine$double.eps)) {
        stop("Some of the columns of the predictor matrix have zero variance.")
    }
    x <- scale(x, FALSE, normx)
} else {
    normx <- rep(1, p)
}

if (scale.y) {
    normy <- sqrt(drop(one %*% (y^2))/(n - 1))
    if (any(normy < .Machine$double.eps)) {
        stop("Some of the columns of the response matrix have zero variance.")
    }
    y <- scale(y, FALSE, normy)
} else {
    normy <- rep(1, q)
}

betahat <- matrix(0, p, q)
betamat <- list()
x1 <- x
y1 <- y
type <- correctp(x, y, eta, K, kappa, select, fit)
eta <- type$eta
K <- type$K
kappa <- type$kappa
select <- type$select
fit <- type$fit
if (is.null(colnames(x))) {
    xnames <- c(1:p)
} else {
    xnames <- colnames(x)
}

new2As <- list()
if (trace) {
    cat("The variables that join the set of selected variables at each step:
")
}
for (k in 1:K) {
    Z <- t(x1) %*% y1
    what <- spls.dv(Z, eta, kappa, eps, maxstep)
    A <- unique(ip[what != 0 & betahat[, 1] != 0])
    new2A <- ip[what != 0 & betahat[, 1] == 0]
    xA <- x[, A, drop = FALSE]
    plsfit <- plsr(y ~ xA, ncomp = min(k, length(A)), method = fit,
betahat <- matrix(0, p, q)
betahat[A, ] <- matrix(coef(plsfit), length(A), q)
betamat[[k]] <- betahat
pj <- plsfit$projection
if (select == "pls2") {
  y1 <- y - x %*% betahat
}
if (select == "simpls") {
pw <- pj %*% solve(t(pj) %*% pj) %*% t(pj)
x1 <- x
x1[, A] <- x[, A, drop = FALSE] - x[, A, drop = FALSE] %*% pw
}
new2As[[k]] <- new2A
if (trace) {
  if (length(new2A) <= 10) {
    cat(paste("- ", k, "th step (K = ", k, "):\n", sep = ""))
    cat(xnames[new2A])
    cat("\n")
  } else {
    cat(paste("- ", k, "th step (K = ", k, "):\n", sep = ""))
    nlines <- ceiling(length(new2A)/10)
    for (i in 0:(nlines - 2)) {
      cat(xnames[new2A[(10 * i + 1):(10 * (i + 1))]])
      cat("\n")
    }
    cat(xnames[new2A[(10 * (nlines - 1) + 1):length(new2A)]])
    cat("\n")
  }
}
if (!is.null(colnames(x))) {
  rownames(betahat) <- colnames(x)
}
if (q > 1 & !is.null(colnames(y))) {
  colnames(betahat) <- colnames(y)
}
object <- list(x = x, y = y, betahat = betahat, A = A, betamat = betamat, 
  new2As = new2As, mu = mu, meanx = meanx, normx = normx, 
  normy = normy, eta = eta, K = K, kappa = kappa, select = select,
fit = fit, projection = pj)
class(object) <- "spls"
object
}

> cv.spls <- function (x, y, fold = 10, K, eta, kappa = 0.5, select = "pls2",
  fit = "simpls", scale.x = TRUE, scale.y = FALSE,
  plot.it = TRUE, do.SIS = FALSE, pval = 0.05,
  adjust = p.adjust.methods) {

  x <- as.matrix(x)
n <- nrow(x)
p <- ncol(x)
ip <- c(1:p)
y <- as.matrix(y)
q <- ncol(y)
type <- correctp(x, y, eta, K, kappa, select, fit)
etta <- type$eta
K <- type$K
kappa <- type$kappa
select <- type$select
fit <- type$fit
foldi <- split(sample(1:n), rep(1:fold, length = n))
mspemat <- matrix(0, length(eta), length(K))
if(do.SIS == TRUE){
  cat(paste("SIS", "\n"))
sis.cv <- SISCV(exprs = t(x),
               index = y,
               adjust = adjust,
               fold = fold,
               foldi = foldi)[[2]]
}

  for (i in 1:length(eta)) {
    cat(paste("eta = ", eta[i], "\n"))
    mspemat[i, ] <- matrix(0, fold, length(K))
    for (j in 1:fold) {
      omit <- foldi[j]

      if(do.SIS == TRUE){
        sis <- sis.cv[, j]
        sis.names <- names(sis[sis <= pval])
        if(length(sis.names) < max(K)){
          sis.names <- names(sort(sis)[1:max(K)])
        }
    }

133
object <- spls(x[-omit, sis.names, drop = FALSE], y[-omit, drop = FALSE], eta = eta[i], kappa = kappa, K = max(K), select = select, fit = fit, scale.x = scale.x, scale.y = scale.y, trace = FALSE, do.SIS = FALSE, pval = pval, adjust = adjust)

newx <- x[omit, colnames(object$x), drop = FALSE]
newx <- scale(newx, object$meanx, object$normx)
betamat <- object$betamat
for (k in K) {
pred <- newx %*% betamat[[k]] + matrix(1, nrow(newx), 1) %*% object$mu
mspemati[j, (k - min(K) + 1)] <- mean(apply((y[omit, ] - pred)^2, 2, mean))
}
mspemat[i, ] <- apply(mspemati, 2, mean)

minpmse <- min(mspemat)
rownames(mspemat) <- eta
colnames(mspemat) <- K
mspecol <- apply(mspemat, 2, min)
msperow <- apply(mspemat, 1, min)
K.opt <- min(K[mspecol == minpmse])
eta.opt <- max(eta[msperow == minpmse])
cat(paste("Optimal parameters: eta = ", eta.opt, ", ", sep = "\n"))
cat(paste("K = ", K.opt, "\n", sep = ""))
if (plot.it) {
  heatmap.spls(t(mspemat), xlab = "K", ylab = "eta", main = "CV MSPE Plot",
              coln = 16, as = "n")
}
rownames(mspemat) <- paste("eta = ", eta)
colnames(mspemat) <- paste("K = ", K)
cv <- list(mspemat = mspemat, eta.opt = eta.opt, K.opt = K.opt)
invisible(cv)

> predict.spls <- function (object, newx, type = c("fit", "coefficient"), ...)
```r
{ 
  type <- match.arg(type)
  betahat <- object$betahat
  x <- object$x
  A <- object$A
  cols <- colnames(x)
p <- ncol(x)
if (type == "fit") {
  if (missing(newx)) {
    pred <- x %*% betahat + matrix(1, nrow(x), 1) %*% object$mu
  } else {
    newx <- newx[, cols, drop = FALSE]
    if (ncol(newx) != p & ncol(newx) != length(A)) {
      stop("The dimension of test dataset is inappropriate!")
    }
    if (ncol(newx) == p) {
      newx <- newx[, A, drop = FALSE]
    }
    newx <- scale(newx, object$meanx[A], object$normx[A])
pred <- newx %*% betahat[, , drop = FALSE] + matrix(1, nrow(newx), 1) %*% object$mu
  }
} else {
  pred <- betahat
}
invisible(pred)
}

> trace.mspe <- function(fit.cv, header = "", col = our.colscheme,
                        ylim = "gen", xlim = "gen",
                        lty = rep(1, dim(fit.cv$mspemat)[2])){
  colMap <- colors()[rep(c(272, 177, 297, 201, 321, 225, 418, 355), 2)]
  X <- fit.cv$mspemat
  eta <- as.numeric(substring(rownames(X), first = 6))
  dimX <- dim(X)
  if(ylim[1] == "gen"){ylim <- c(min(X), max(X))}
  if(xlim[1] == "gen"){xlim <- c(min(eta), max(eta))}
  plot(eta, X[, 1], ylim = ylim, xlim = xlim,
       xlab = expression(eta),
       ylab = "CV MSPE", type = "n", main = header)

135
}
for(i in 1:dimX[2]){
    lines(eta, X[, i], lty = lty[i], col = col[i], lwd = 1.5)
}

legend("bottomright", bty = "n",
    title = "k",
    legend = as.character(1:dimX[2]),
    lty = lty,
    col = col)

16.4 Functions used in the Simple Analysis

Function to make simple sensitivity predictions

```r
> my.predict <- function(x,y){
  x %*% y
}
```

16.5 Annotation and Plotting Tools

```r
> lookUpPubmed <- function(spls.fit, n.coef = 3,
    query.1 = "std"){
  Coef <- coef(spls.fit)
  Coef <- Coef[Coef != 0, ]
  Coef <- round(Coef[order(Coef)], n.coef)
  resTable <- data.frame(geneSymbol = as.character(
                        unlist(lookUp(names(Coef),
                          "hgu133plus2","SYMBOL"))))
  rownames(resTable) <- gsub("_","\\textunderscore ", names(Coef))
  anTable <- aafTableAnn(names(Coef), "hgu133a2.db", aaf.handler())
  band <- rep("", length(resTable[, 1]))
  for(i in 1:length(resTable[, 1])){
      band[i] <- paste(anTable[[8]][[i]]@band, collapse = ", ")
  }
  resTable$mean <- round(spls.fit$meanx[spls.fit$A], n.coef)
  resTable$sd <- round(spls.fit$normx[spls.fit$A], n.coef)
  resTable$Locus <- band
  resTable$weight <- Coef
```
resTable$pmid <- 0
for(i in 1:length(resTable[, 1])){
  if(query.1 == "std"){
    query <-
    paste(resTable[i,]$geneSymbol,
      ' AND chemotherapy AND (resistance OR sensitivity)',
      sep = ""
  }else{
    query <- query.1
  }
  query <- gsub('\s+','+', query)
  url <-
  url <- paste(url, "&db=pubmed&term=" , query, sep = ""
  datafile <- tempfile(pattern = "pub")
  try(download.file(url, destfile = datafile, method = "internal",
    mode = "wb", quiet = TRUE), silent = TRUE)
  xml <- xmlTreeParse(datafile, asTree = TRUE)
  nid <- xmlValue(xmlElementsByTagName(xmlRoot(xml), "Count")[[1]])
  #lid <- xmlElementsByTagName(xmlRoot(xml), "IdList", recursive = TRUE)[[1]]
  #pmidlist <- unlist(lapply(xmlElementsByTagName(lid, "Id"), xmlValue))
  resTable[i,]$pmid <- as.numeric(nid)#length(pmidlist)
  colnames(resTable) <- c("Gene Symbol", "Mean", "SD", "Location", "Weight", "PMID"
  return(resTable)
}

> PlotKM.sda <- function(index,surv.object,
  col = c("black", "darkgrey", "red"),
  legend = c("Sensitive", "Intermediate", "Resistant"),
  col.leg = col,
  lty = c(1, 1, 1),
  main = "",
  ylab = "",
  xlab = "",

137
levels <- levels(index)

plot(survfit(surv.object ~ index),
      col = col,
      lwd = 2,
      xlab = xlab,
      ylab = ylab,
      xmax = xmax,
      main = main)

logRankTest <- survdiff(surv.object ~ index)

nchar <- max(nchar(legend)) - nchar(legend) + 1
spaces <- vector()
for(i in 1:length(legend)){
    spaces[i] <- paste(rep(" ", nchar[i]), sep = "", collapse = "")
}
xx <- as.matrix(table(predictDLDArkanas$class))
xx <- xx[legend,]

nchar <- max(nchar(xx)) - nchar(xx) + 1
spaces2 <- vector()
for(i in 1:length(legend)){
    spaces2[i] <- paste(rep(" ", nchar[i]), sep = "", collapse = "")
}

old.par <- par(no.readonly = TRUE)
on.exit(par(old.par))

par(family = 'mono')

legend("bottomleft",
       legend = paste(legend, ",", spaces, "n = ", spaces2,
                    xx, sep = ""),
       bty = "n",
       col = col.leg,
       lty = lty,
       lwd = 2)
legend("bottomright",
    bty = "n",
    legend = paste("P-value = ",
    as.character(signif(1-pchisq(logRankTest$chisq, 1), 1)), sep = ""))

return(signif(1-pchisq(logRankTest$chisq, 1), 1))
}

> CalcHR <- function(index,surv.data,surv.ind){
  threshold <- cut(index,
      c(min(index) - 1,
      quantile(index, cut.points),
      max(index) + 1 ))
  uthres <- levels(threshold)[c(1,3)]
  thres <- as.character(threshold[threshold == uthres[1] |
      threshold == uthres[2]])
  new.index <- index[threshold == uthres[1] |
      threshold == uthres[2]]
  new.surv <- surv.data[threshold == uthres[1] |
      threshold == uthres[2]]
  new.ind <- surv.ind[threshold == uthres[1] |
      threshold == uthres[2]]
  new.surv.object <- Surv(as.numeric(new.surv),new.ind)
  fit <- coxph(new.surv.object ~ as.factor(thres))
  coef.m <- -1*fit$coef
  std.m <- sqrt(fit$var)
  return(
    paste("HR = ",signif(exp(coef.m),2), " (",
    signif(exp(coef.m)-1.96*std.m,3),",",
    signif(exp(coef.m)+1.96*std.m,3),")", 
    sep="")
  )
}

>
PlotKM <- function(index, surv.object, 
cut.points = c(1/3, 2/3),
our.colscheme = c("black", "darkgrey", "red"),
legend = c("Sensitive", "Intermediate", "Resistant"),
lty = c(1, 1, 1),
main = "",
ylab = "",
xlab = "",
...) {

# Function to perform KM plot

threshold <- cut(index,
  c(min(index) - 1,
    quantile(index, cut.points),
    max(index) + 1 ))

levels <- levels(threshold)

plot(survfit(surv.object ~ threshold),
  col = our.colscheme,
  lwd = 2,
  xlab = xlab,
  ylab = ylab,
  main = main, ...)

logRankTest <- survdiff(surv.object ~ threshold)

nchar <- max(nchar(legend)) - nchar(legend) + 1
spaces <- vector()
for(i in 1:length(legend)){
  spaces[i] <- paste(rep(" ", nchar[i]), sep = "", collapse = "")
}

nchar <- max(nchar(summary(threshold))) - nchar(summary(threshold)) + 1
spaces2 <- vector()
for(i in 1:length(legend)){
  spaces2[i] <- paste(rep(" ", nchar[i]), sep = "", collapse = "")
}

old.par <- par(no.readonly = TRUE)
on.exit(par(old.par))

par(family = 'mono')
legend("bottomleft",
      legend = paste(legend, ",", spaces, "n = ", spaces2,
                    summary(threshold), sep = ""),
      bty = "n",
      col = our.colscheme,
      lty = lty,
      lwd = 2)

legend("bottomright",
       bty = "n",
       legend = paste("P-value = ",
                      as.character(signif(1-pchisq(logRankTest$chisq, 1), 1)), sep = ""))

return(signif(1-pchisq(logRankTest$chisq, 1), 1))
}

141