Package ‘kmcut’ v.2.0 tutorial

Igor B. Kuznetsov and Javed Khan

July 15, 2020

1. Overview

This document provides a brief tutorial for the R package ‘kmcut’. The main purpose of the package is to identify potential prognostic gene signatures and an optimal numeric cutoff for each signature that can be used to stratify a group of test subjects (samples) into two sub-groups with significantly different survival (better *vs*. worse). Originally, the package was intended to be used with variables that describe gene expression, such as microarray/RNAseq expression levels of individual genes or gene expression-derived signatures, such as single-sample Gene Set Enrichment Analysis (ssGSEA) signatures. However, it can be used with any numeric feature that has a sufficiently large proportion of unique values. The main requirement of the package is that for a group of test subjects (samples) two types of data are available: (i) right-censored survival time data and (ii) at least one gene expression-like feature with a large proportion of unique numeric values describing each test subject.

2. Installing the package

The package can be installed directly from github.com by utilizing the function install_github() from the package “devtools”. The code is below:

```
# Install package “devtools”
install.packages("devtools")
# Install package “kmcut”
library(devtools)
install_github("ibkstore/kmcut2")
```

Another way is to download archive “kmcut2.tar.gz” from https://github.com/ibkstore/kmcut2 to a local folder and then use it to install the package. The code below uses “kmcut2.tar.gz” downloaded to a local folder “c:/test”:

```
install.packages("C:/test/kmcut2.tar.gz", repos = NULL, type = "source")
```

You may also need to install a few other packages used by “kmcut” (if these packages are not already installed on your computer):

```
install.packages(c("survival", "stringr", "data.table","pracma"))
```

3. Preparing the input data

All core functions in the package require two input tab-delimited text files: (i) a file with right-censored survival data and (ii) a file with gene expression-like feature(s). The package contains two built-in example files with survival data and gene expression data that describe 295 neuroblastoma tumor samples [1] (see below).

i) The file with survival data must contain at least three columns labeled ‘sample_id’, ‘stime’, and ‘scens’. Column ‘sample_id’ contains a unique identifier of each test subject (sample) and must be the first column in the file. Column ‘stime’ contains the survival time for each test subject. Column ‘scens’ contains the censoring variable for each test subject (0 or 1). If other columns are present in the file, they will be ignored. ‘stime’ and ‘scens’ can be in any column in the file, except the first. An example file with survival data provided with the package can be loaded and viewed as follows:

```
library(kmcut)
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
writeLines(readLines(con=sdat))
```

Output:

```
>sample_id      stime      scens
>SEQC_NB001     1362       1
>SEQC_NB002     2836       1
>SEQC_NB003     1191       1
>SEQC_NB005     220        1
>SEQC_NB006     2217       0
...
```

ii) The file with gene expression-like feature(s) must contain samples in columns and features in rows. The first column and first row must contain test subject (sample) and gene identifiers, respectively. An example file with gene expression features provided with the package can be loaded and viewed as follows:

```
library(kmcut)
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
writeLines(readLines(con=fdat))
```

Output:

```
>tracking_id    SEQC_NB001     SEQC_NB002     SEQC_NB003     SEQC_NB005     SEQC_NB006
>MYCN           4.16347458     3.464994927    8.494631614    8.438018327    5.509974474
>MYH2           0.006539622    0.009077256    0              4.111214977    0.008735951
...
```

The sample identifiers in both files must be exactly the same.

4. Running the package and interpreting the output

4.1 Function 'kmoptpermcut'

This function uses each distinct value of every feature from the dataset as a stratification cutoff to select a cutoff that results in the maximum separation of the Kaplan-Meier survival curves, and then estimates the statistical significance of this optimal cutoff by means of the permutation test. A detailed description of the steps implemented in this function is provided below.

First, an ordered list C of all distinct values of a given feature observed in the group of test subjects (samples) is created, all values in the list being sorted from smallest to largest. Then, each value from the list is used as a stratification cutoff: samples with the feature below or equal to the cutoff are labeled as 'low' and above the cutoff as 'high'. To avoid edge effects, if for a particular cutoff the size of low or high sub-group is smaller than 'min_fraction' of the total number of samples, this cutoff is discarded from the list. The log-rank test is applied to compare survival distributions between the low and high groups for each stratification cutoff, and the value of the test statistic is recorded. After all stratification cutoffs from the list are tested, the vector of the observed values of the test statistic is created, $O=(o_1, o_2,\ldots, o_n)$, where o_i is the observed value of the test statistic for the cutoff c_i in the list $C=(c_1, c_2,\ldots, c_n)$. The cutoff that results in the largest value of the test statistic, O_{MAX}, is selected as the optimal cutoff, C_{OPT}. If two or more cutoffs result in the same observed value of the test statistic, the cutoff closest to the median is selected.

Additionally, the shape of the plot of the observed values of the test statistic is compared to the expected "ideal" plot. The empirical assumption behind this comparison is that the "ideal" optimization should produce a plot of the observed values of the test statistic with exactly one peak, meaning that the values monotonically increase before the peak and monotonically decrease after the peak. Initially, three possible expected plots are defined: E_1, E_2, and E_3. Each plot consists of cutoff points from C, has a single peak, with the height of this peak being equal to O_{MAX}. In the plot E_1 the peak is located at the 25^{th} percentile of all distinct values of the feature, in the plot E_2 at the 50^{th} percentile, and in the plot E_3 at the 75^{th} percentile. The final expected plot, E, is selected from (E_1,E_2,E_3) to minimize the distance from the location of its peak to C_{OPT}. The similarity of is quantified by the Spearman rank correlation between the vectors of the observed and expected values, $r(O, E)$.

The statistical significance of the stratification cutoff C_{OPT} obtained from the optimization procedure for a given feature is estimated by means of a random permutation test run for 'n_iter' iterations. On each iteration i, the sample labels in the survival data and in the feature data are randomly shuffled, the same optimization procedure is applied to the randomized data, and the largest value of the observed test statistic $O_{RAND}(i)$ and the Spearman rank correlation between the observed and expected plot $r_{RAND}(i)$ are recorded. After all 'n_iter' iterations are completed, the p-value is calculated as follows:

$$\boldsymbol{p} = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n\_iter}} \boldsymbol{d}[\boldsymbol{O_{RAND}(i)} \geq \boldsymbol{O_{MAX}\ AND\ r_{RAND}(i)} \geq \boldsymbol{r(O,E)}]}{n\_iter}$$

Where $d[O_{RAND}(i) \geq O_{MAX}$ AND $r_{RAND}(i) \geq r(O, E)] = 1$ if on random iteration *i* the largest value of the observed test statistic is equal to or greater than O_{MAX} *and* the Spearman rank correlation between the observed and the expected plot is equal to or greater than $r(O, E)$. Otherwise, $d[O_{RAND}(i) \geq O_{MAX}$ AND $r_{RAND}(i) \geq r(O, E)] = 0$.

Example of how to use 'kmoptpermcut' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Run the permutation test for 100 iterations for each gene, with fixed seed=1234
kmoptpermcut(fname=fdat, sfname=sdat, wdir="c:/test", n_iter=100, seed=1234)
```

This run will create three output files in directory "c:/test":

a) PDF file with plots:
"example_genes_295_KMoptp_minf_0.10_iter_100.pdf"

This file contains two plots created for each of the two genes: MYCN (Figures 1 & 2) and MYH2 (Figures 3 & 4).

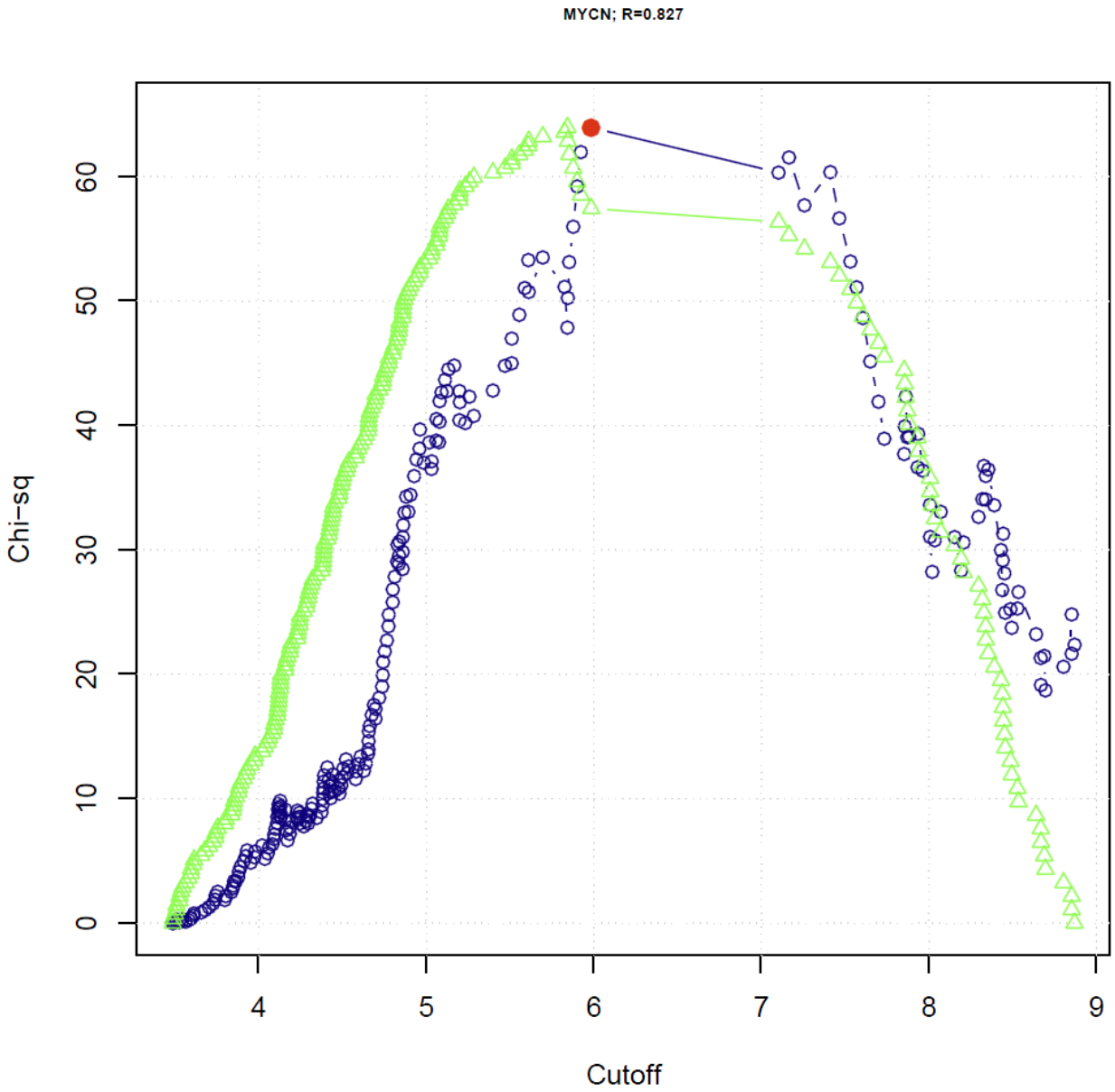

Figure 1. The observed optimization plot (blue circles) and the expected optimization plot (green triangles) for the MYCN gene. The optimal stratification cutoff is highlighted by the red circle. The Spearman rank correlation between the observed and the expected plots is high, R=0.827, indicating consistent optimization.

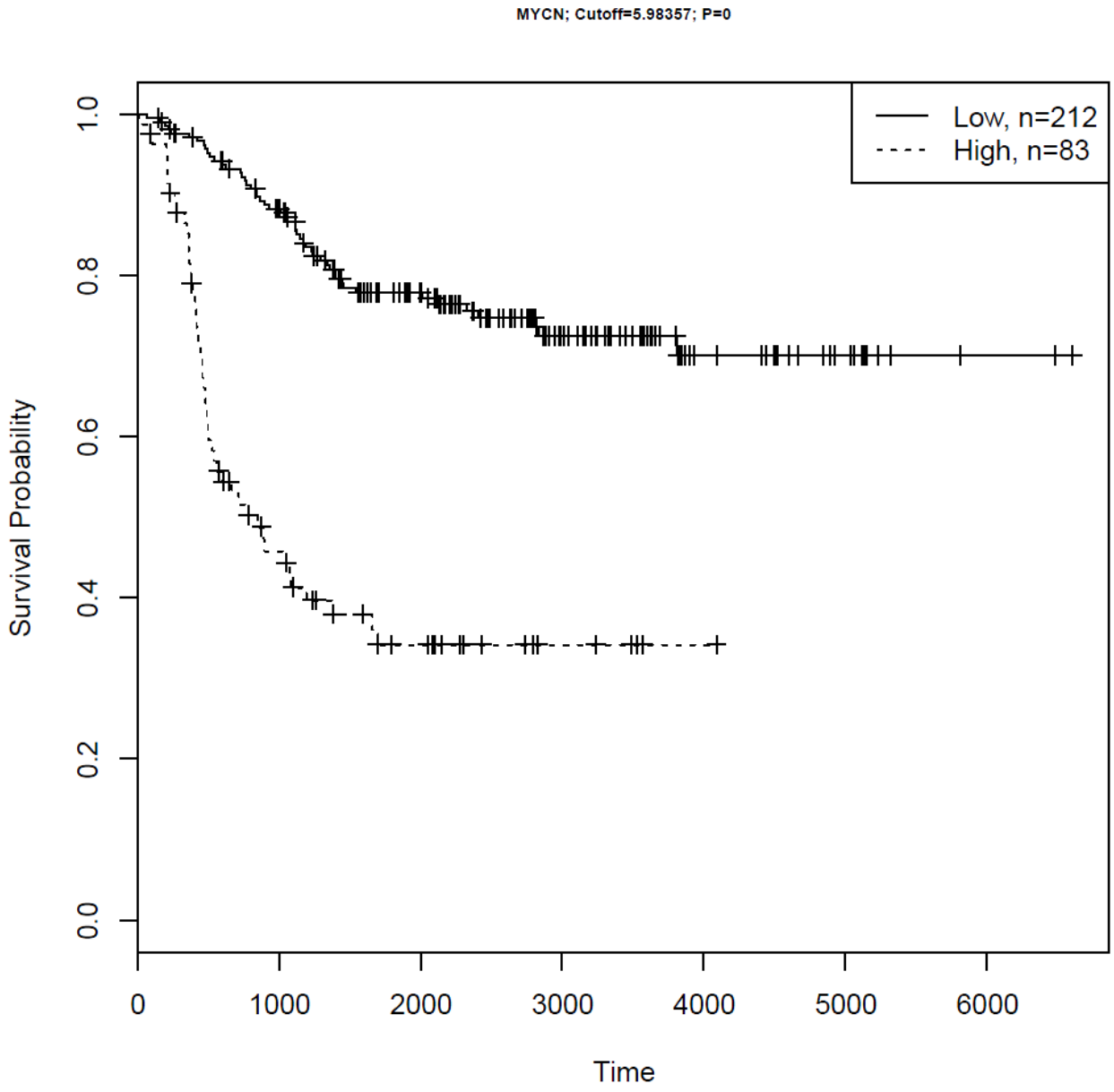

Figure 2. The Kaplan-Meier survival curves for MYCN low and high groups stratified using the optimal cutoff=5.98357. The permutation test p-value is significant, P=0, (note that the p-value is exactly 0 because only 100 iterations of the permutation test were used in this example).

The MYCN oncogene expression is known to be a strong predictor of survival outcome for neuroblastoma patients – low expression levels correspond to better survival, high expression levels correspond to poor survival [2]. This is exactly what is observed based on the significant results of the optimization for MYCN.

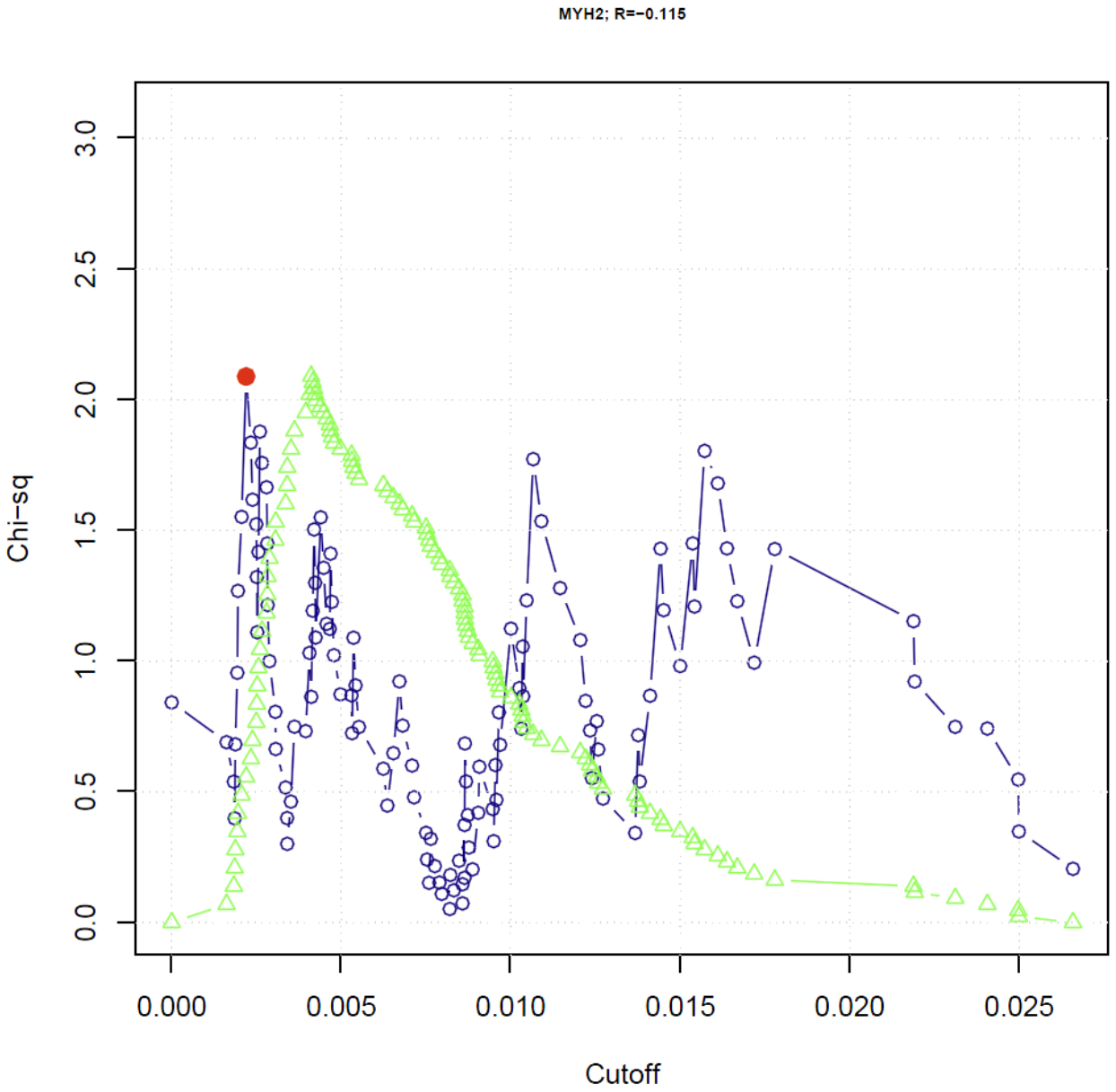

Figure 3. The observed optimization plot (blue circles) and the expected optimization plot (green triangles) for the MYH2 gene. The optimal stratification cutoff is highlighted by the red circle. The Spearman rank correlation between the observed and the expected plots is low, R=-0.115, indicating inconsistent optimization.

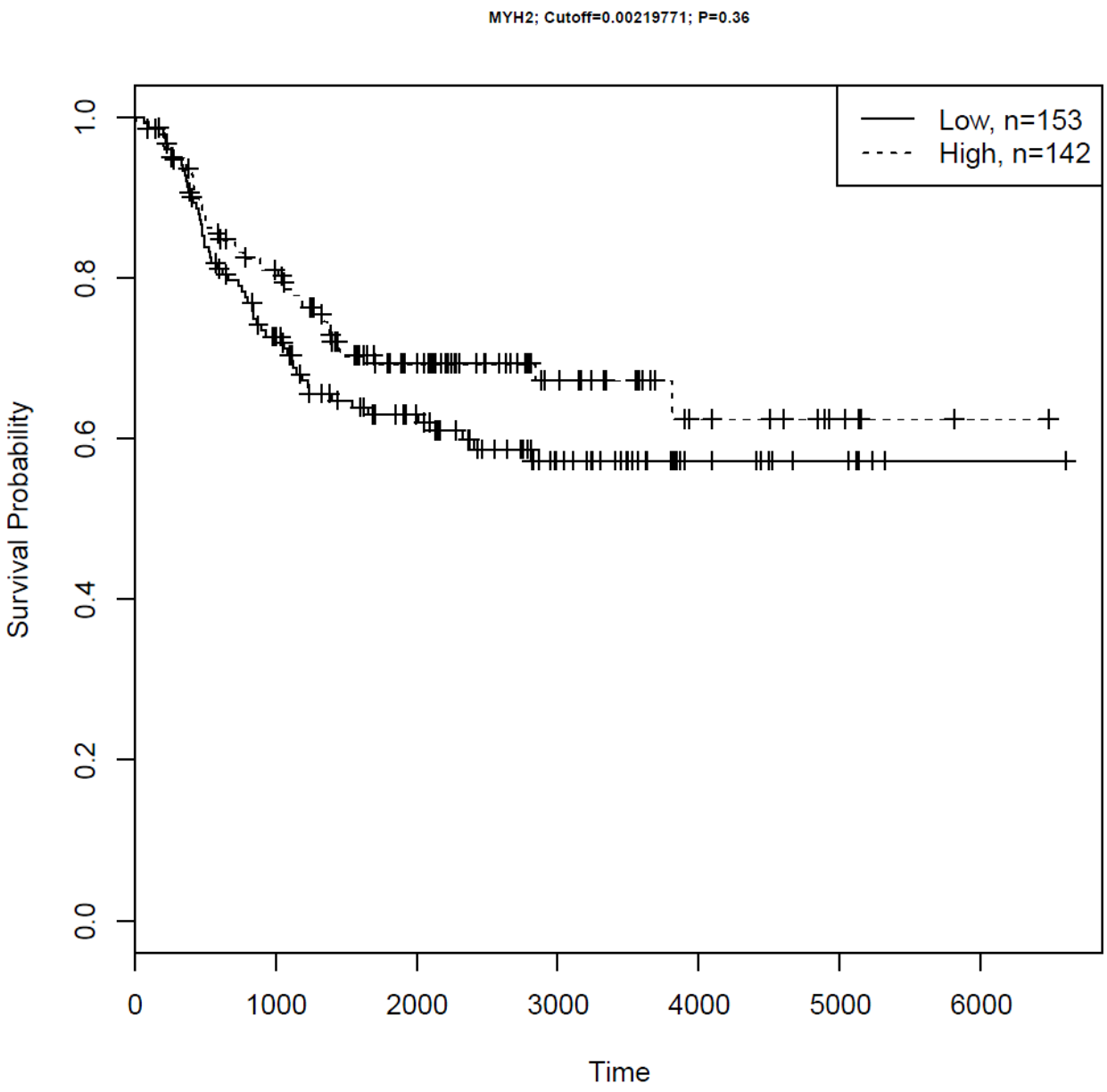

Figure 4. The Kaplan-Meier survival curves for MYH2 low and high groups stratified using the optimal cutoff=0.00219771. The permutation test p-value is not significant (P=0.36).

The MYH2 gene encodes the myosin heavy chain 2, which is a protein found in the muscle tissue and the level of its expression has nothing to do with neuroblastoma survival [3]. This is exactly what is observed based on statistically not significant results of the optimization for MYH2.

b) Tab-delimited text file with the results:
"example_genes_295_KMoptp_minf_0.10_iter_100.txt"

tracking_id	CUTOFF	CHI_SQ	LOW_N	HIGH_N	R	P	FDR_P
MYCN	5.98357	63.94	212	83	0.827	0	0
MYH2	0.00219771	2.09	153	142	-0.115	0.36	0.36

1st column – gene id, 2nd – optimal stratification cutoff, 3rd – test statistic calculated for the optimal cutoff, 4th – number of samples in low-expression sub-group, 5th - number of samples in high-expression sub-group, 6th – permutation test p-value, 7th – FDR-adjusted p-value.

c) CSV file with low/high sample labels:
"example_genes_295_KMoptp_minf_0.10_iter_100_labels.csv"

```
sample_id,MYCN,MYH2
SEQC_NB001,1,2
SEQC_NB002,1,2
SEQC_NB003,2,1
SEQC_NB005,2,2
SEQC_NB006,1,2
SEQC_NB007,1,2
…
```

1st column – sample ids, all subsequent columns contain low/high labels for each gene (feature), where 1 and 2 correspond to low- and high-expression sub-groups, respectively ("low' means below the cutoff and 'high' means above the cutoff).

4.2 Function 'kmoptscut'

This function uses each distinct value of a given feature observed in the dataset as a stratification cutoff to select a cutoff that results in the maximum separation of the Kaplan-Meier survival curves, but does not use the permutation test to estimate the statistical significance of this optimal cutoff. This function produces output files virtually identical to the output of 'kmoptpermcut' (except for the permutation test p-value) and is meant to be used as a fast exploratory alternative to 'kmoptpermcut'. Example of how to use 'kmoptscut' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
```

```
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Run the function
kmoptscut(fname=fdat, sfname=sdat, wdir="c:/test")
```

This will create three output files in directory "c:/test":

a) PDF file with plots:
"example_genes_295_KMopt_minf_0.10.pdf"
b) Tab-delimited text file with the results:
"example_genes_295_KMopt_minf_0.10.txt"
c) CSV file with low/high sample labels:
"example_genes_295_KMopt_minf_0.10_labels.csv"

The format of these three files is identical to the output files described in 3.1 for function 'kmoptpermcut'. The only difference is that the tab-delimited file with the results in column 'P' and PDF file with Kaplan-Meier curves contain the p-value obtained from the log-rank test performed with the optimal cutoff. This value is provided only for information purposes and should not be treated as an actual p-vale because it does not reflect multiple tests involved in the optimization procedure applied to select the optimal cutoff.

4.3 Function 'kmqcut'

This function for each feature uses the cutoff supplied as quantile (from 0 to 100) to stratify samples into 2 groups (below/above this quantile calculated for each feature), plots Kaplan-Meier survival curves for these two groups, and calculates the log-rank test p-value. Since for each feature only one cutoff corresponding to the specified quantile is used, it is equivalent to performing one log-rank test per feature (no optimization is performed).

Example of how to use 'kmqcut' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Use the 50th quantile (the median) to stratify the samples
kmqcut(fname=fdat, sfname=sdat, wdir="c:/test", quant=50)
```

This will create three output files in directory "c:/test":

a) PDF file with Kaplan-Meier curves:
"example_genes_295_KM_quant_50.pdf"
b) Tab-delimited text file with the results:
"example_genes_295_KM_quant_50.txt"
c) CSV file with low/high sample labels:
"example_genes_295_KM_quant_50_labels.csv"

The format of these three files is identical to the output files described in 3.1 for function 'kmoptpermcut'. The only difference is that the PDF file does not contain observed vs. expected optimization plots.

4.4 Function 'kmucut'

This function for each feature uses the user-supplied fixed value as a cutoff to stratify samples into 2 groups (below/above this cutoff), plots Kaplan-Meier survival curves for these two groups, and calculates the log-rank test p-value. Since for each feature the same fixed cutoff is used, it is equivalent to performing one log-rank test per feature (no optimization is performed).

Example of how to use 'kmucut' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Use the cutoff=5 to stratify the samples and remove features that have less than 90% unique
# values (this eliminates MYH2 gene)
kmucut(fname=fdat, sfname=sdat, wdir="c:/test", cutoff=5, min_uval=90)
```

This will create three output files in directory "c:/test":

a) PDF file with plots:
"example_genes_295_KM_ucut_5.pdf"
b) Tab-delimited text file with the results:
"example_genes_295_KM_ucut_5.txt"
c) CSV file with low/high sample labels:
"example_genes_295_KM_ucut_5_labels.csv"

The format of these three files is identical to the output files described in 3.1 for function ‘kmoptpermcut’. The only difference is that the PDF file does not contain observed vs. expected optimization plots.

4.5 Function ‘kmvalcut’

This function creates Kaplan-Meier survival curves for each feature from a validation data file by using a file with previously determined stratification thresholds (one threshold per feature), and calculates the log-rank test p-value. Since for each feature only one previously determined cutoff is used, it is equivalent to performing one log-rank test per feature (no optimization is performed). The file with previously determined stratification thresholds is a tab-delimited file with the table that contains one or more feature and a stratification threshold for each feature (the table is generated by functions ‘kmoptscut’, ‘kmoptpermcut’, ‘kmqcut’ or ‘kmucut’). It must have first two columns named as 'tracking_id' and 'CUTOFF'. The 'tracking_id' column contains feature names, the 'CUTOFF' column contains stratification threshold for each feature.

Example of how to use ‘kmvalcut’ with the data files included in the package to validate stratification thresholds previously determined by ‘kmqcut’. Note that in this example the file used for validation is the same file ‘example_genes_295.txt’ originally used to determine the thresholds as described in 3.3. For a real validation a file with independent test data should be used.

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# "example_genes_295_KM_quant_50.txt" is a file with cutoffs created by 'kmqcut'
# and must exist in directory "c:/test"
kmvalcut(input1="example_genes_295_KM_quant_50.txt", input2=fdat, sfname=sdat,
wdir="c:/test")
```

This will create three output files in directory "c:/test":

a) PDF file with plots:
"example_genes_295_KM_val.pdf"
b) Tab-delimited text file with the results:
"example_genes_295_KM_val.txt"

c) CSV file with low/high sample labels:
"example_genes_295_KM_val_labels.csv"

The format of these three files is identical to the output files described in 3.1 for function 'kmoptpermcut'. The only difference is that the PDF file does not contain observed vs. expected optimization plots.

4.6 Function 'ucoxbatch'

This function performs a univariate Cox proportional hazard regression for each feature in the dataset in a batch mode. It calculates the hazard ratio, concordance, and the likelihood ratio test p-value. Example of how to use 'ucoxbatch' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Run the function
ucoxbatch(fname = fdat, sfname = sdat, wdir = "c:/test")
```

This will create a tab-delimited text file "example_genes_295_ucoxpbatch.txt" in directory "c:/test":

tracking_id	CC	HR	P	FDR_P
MYCN	0.67456	1.35531	1.23567857778923e-10	2.47135715557845e-10
MYH2	0.47145	1.04611	0.869693551308325	0.869693551308325

1st column – gene id, 2nd – concordance, 3rd – hazard ratio, 4th – likelihood ratio test p-value, 5th – FDR-adjusted p-value.

4.7 Function 'ucoxpred'

This function fits a univariate Cox proportional hazard regression model for each feature in a training dataset and then uses the models to predict risk scores for the same features in a test dataset. Example of how to use 'ucoxpred' with the data files included in the package:

```
library(stats)
library(survival)
library(stringr)
```

```
library(data.table)
library(tools)
library(pracma)
library(kmcut)
# Load example gene expression data and survival data for 2 genes and 295 samples
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
sdat = system.file("extdata", "survival_data_295.txt", package="kmcut")
# Use the example dataset as both a training dataset and as a test dataset (that is, perform a
# resubstitution). Note that in a real world application training and test datasets will be
# different from one another.
ucoxpred(fname1=fdat, sfname1=sdat, fname2=fdat, sfname2=sdat, wdir="c:/test")
```

This will create three output files in directory "c:/test":

a) Tab-delimited text file with Cox regression summary for the training data: "example_genes_295_cox_train_sum.txt"

tracking_id	CC	HR	P	FDR_P
MYCN	0.67456	1.35531	1.23567857778923e-10	2.47135715557845e-10
MYH2	0.47145	1.04611	0.869693551308325	0.869693551308325

1st column – gene id, 2nd – concordance, 3rd – hazard ratio, 4th – likelihood ratio test p-value, 5th – FDR-adjusted p-value.

b) Tab-delimited text file with the risk scores for the training data: "example_genes_295_train_score.txt"

In this file, rows are features (genes) and columns are samples.

c) Tab-delimited text file with the predicted risk scores for the test data: "example_genes_295_test_score.txt"

In this file, rows are features (genes) and columns are samples.

4.8 Functions for manipulating data tables

The package also contains the following functions that can be used to manipulate data tables:

1) Function ‘extractrows’ extracts a sub-set of rows (such as a group of gene ids) from a data table. All columns will be preserved. Names of the rows to be extracted must be in a text file, one name per line (the exact names, case-sensitive, no extra symbols are allowed). Example of how to use the function with the data files included in the package:

```
library(data.table)
library(kmcut)
```

```
# Load example gene expression data table for 2 genes (2 rows)
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
# Load a list that contains one gene id (MYCN)
idlist = system.file("extdata", "rowids.txt", package="kmcut")
# Run the function
extractrows(fnamein=fdat, fids=idlist, fnameout="example_genes_subset.txt", wdir="c:/test")
```

This will create a tab-delimited text file "example_genes_subset.txt" with one row "MYCN" in directory "c:/test".

2) Function ‘extractcolumns’ extracts a sub-set of columns (such as a group of samples) from a data table. All rows will be preserved. Names of the columns to be extracted must be in a text file, one name per line (the exact names, case-sensitive, no extra symbols are allowed). Example of how to use the function with the data files included in the package:

```
library(data.table)
library(kmcut)
# Load example gene expression data table for 2 genes
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
# Load a list that contains column (sample) ids
idlist = system.file("extdata", "columnids.txt", package="kmcut")
# Run the function
extractcolumns(fnamein=fdat, fids=idlist, fnameout="example_samples_subset.txt",
wdir="c:/test")
```

This will create a tab-delimited text file "example_samples_subset.txt" in directory "c:/test".

3) Function ‘transposetable’ transposes a data table (that is, converts rows to columns and columns to rows). Row names will become column names, and column names will become row names. Example of how to use the function with the data file included in the package:

```
library(data.table)
library(kmcut)
# Load example gene expression data table for 2 genes. In this file, genes are rows and samples
are columns.
fdat = system.file("extdata", "example_genes_295.txt", package="kmcut")
# Run the function
transposetable(fnamein=fdat, fnameout="example_genes_295_transposed.txt", wdir="c:/test")
```

This will create a tab-delimited text file "example_genes_295_transposed.txt" with the transposed table in directory "c:/test". In this file, genes are columns and samples are rows.

5. References

[1] Zhang W, Yu Y, Hertwig F, et al: Comparison of RNA-seq and microarray-based models for clinical endpoint prediction. Genome Biol. 16:133, 2015

[2] Norris MD, Bordow SB, Haber PS, et al: Evidence that the MYCN oncogene regulates MRP gene expression in neuroblastoma. Eur. J. Cancer. 33(12):1911-1916, 1997

[3] Smerdu V, Karsch-Mizrachi I, Campione M, et al: Type IIx myosin heavy chain transcripts are expressed in type IIb fibers of human skeletal muscle**.** Am. J. Physiol. 267(6 Pt 1): C1723-1728, 1994.