Supplement: switchBox: An R package for k-Top Scoring Pairs (kTSP) classifier development.

Bahman Afsari[1], Donald Geman[2], Elana J. Fertig[1] and Luigi Marchionni[1]

[1] The Sidney Kimmel Comprehensive Cancer Center,
Johns Hopkins University School of Medicine
[2] Department of Applied Mathematics,
Johns Hopkins University School of Medicine

Modified: July 29, 2014. Compiled: August 11, 2014

Contents

1 Introduction

The switchBox R Bioconductor package allows to train and validate a K-Top-Scoring-Pair (KTSP) classifier, as used by Marchionni et al to predict breast cancer recurrence [1]. KTSP is an extension of the Top-Scoring Pair (*TSP*) algorithm described by Geman and colleagues [2, 3, 4]. A TSP is a simple binary classifier based on the ordering of two measurements (*e.g.* the expressions of two genes), in which each of the two possible orderings votes for one class or the other. The KTSP algorithm aggregates the votes from multiple disjoint TSPs based on a specific decision rule, usually the "majority wins" rule.

This supplement to the manuscript *"switchBox: An R package for k-Top Scoring Pairs (kTSP) classifier development"*, exemplifies how to use the switchBox package and shows its advanges over the ktspair package [5]. Additional examples are also available in the package vignette.

2 Installing the package

To download and install the package switchBox from Bioconductor :

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

To download and install the package switchBox from GitHub:

```
> require(devtools)
> install_git("https://github.com/marchion/git.switchBox", subdir="switchBox")
```

To load the library:

```
> require(switchBox)
> require(gplots)
```

3 Data structure

3.1 Training set

Load the example training data contained in the switchBox package:

```
> ### Load the example data for the TRAINING set
> data(trainingData)
```

The object matTraining is a numeric matrix containing gene expression data for the 78 breast cancer patients and the 70 genes used to implement the MammaPrint assay [6]. This data was obtained from from the MammaPrintData package, as described in [1]. Samples are stored by column and genes by row. Gene annotation is stored as rownames(matTraining).

```
> class(matTraining)
[1] "matrix"
> dim(matTraining)
[1] 70 78
```

```
> str(matTraining)

 num [1:70, 1:78] -0.0564 0.0347 -0.0451 -0.1556 0.1394 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:70] "AA555029_RC_Hs.370457" "AF257175_Hs.15250" "AK000745_Hs.377155" "AKAP2_Hs.516834" ...
  ..$ : chr [1:78] "Training1.Bad" "Training2.Bad" "Training3.Good" "Training4.Good" ...
```

The factor `trainingGroup` contains the prognostic information:

```
> ### Show group variable for the TRAINING set
> table(trainingGroup)

trainingGroup
 Bad Good
  34   44
```

3.2 Testing set

Load the example testing data contained in the `switchBox` package:

```
> ### Load the example data for the TEST set
> data(testingData)
```

The object `matTesting` is a numeric matrix containing gene expression data for the 307 breast cancer patients and the 70 genes used to validate the MammaPrint assay [7]. This data was obtained from from the `MammaPrintData` package, as described in [1]. Also in this case samples are stored by column and genes by row. Gene annotation is stored as `rownames(matTraining)`.

```
> class(matTesting)

[1] "matrix"

> dim(matTesting)

[1]  70 307

> str(matTesting)

 num [1:70, 1:307] 0.0035 -0.0599 -0.0678 0.1139 -0.094 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:70] "AA555029_RC_Hs.370457" "AF257175_Hs.15250" "AK000745_Hs.377155" "AKAP2_Hs.516834" ...
  ..$ : chr [1:307] "Test1.Good" "Test2.Good" "Test3.Good" "Test4.Good" ...
```

The factor `testingGroup` contains the prognostic information:

```
> ### Show group variable for the TEST set
> table(testingGroup)

testingGroup
 Bad Good
  47  260
```

4 Producing Figure 1

Here we show how to generate a heatmap displaying individual classification for each pair contained in a KTSP classifier (see Figure 1 of the main paper and Figure 1 in the supplement). The code is identical to the main paper except for the plotting commands, which we did not include in the paper due to space limitations. The chunk of code below is used to train the classifier and compute individual KTSP votes in the training set.

```
> ### Training kTSP classifier
> classifier <- SWAP.KTSP.Train(matTraining, trainingGroup)

Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.

> ### Compute the KTSP statistics
> kappa <- SWAP.KTSP.Statistics(matTraining, classifier)
```

The chunk of code below is used to generate the heatmap depicting the individual votes for all the TSP contained in the classifier.

```
> mat <- t(1*kappa$comparisons)
> rownames(mat) <- gsub(">", "\n more express than\n", rownames(mat))
> heatmap.2(mat,
            scale="none", Rowv=TRUE, Colv=TRUE,  dendrogram="none",
            trace="none", key=FALSE,
            col=c("lightsteelblue2", "pink3"),
            labCol=toupper(paste(trainingGroup, "Prognosis")),
            sepwidth=c(0.075,0.075), sepcolor="black",
            rowsep=1:ncol(kappa$comparisons),
            colsep=1:nrow(kappa$comparisons),
            lmat=rbind( c(0, 3), c(2, 1), c(0, 4) ), lhei=c(0.1, 5, 0.5), lwid=c(0.15, 5),
            mar=c(7.5, 12), cexRow=0.85, cexCol=0.9)
```

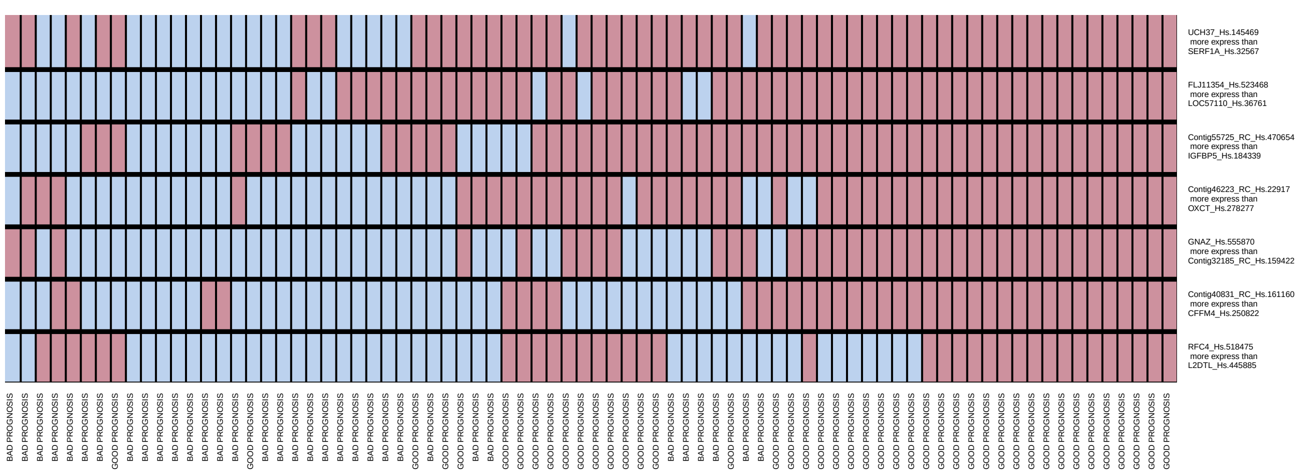

Figure 1: Heatmap showing the individual TSP votes.

5 Comparison to the `ktspair` package

In this section, we compare `switchBox` with the previously proposed package `ktspair` [5]. The `switchBox` package adds additional features to `ktspair`:

1. `switchBox` allows for more flexibility in defining the range of candidates for selecting k, (*e.g.*, defining the number of gene pairs to chose from when building the classifier) (see Section 5.1);
2. The function `SWAP.KTSP.Statistics` in the `switchBox` package enables flexible rules to aggregate the votes from each TSP contained in the classfier (see Section 5.2);
3. The function `SWAP.CalculateSignedScore` in `switchBox` allows to calculate the pairwise scores for applications other than classification (see Section 5.3).

In addition, we run numerical comparisons between classifiers of five year breast cancer recurrence from gene expression data in [1]. These experiments show that `switchBox` is:

1. Faster (Section 5.4);
2. More robust since ktspair may yield different classifiers and predictions from the identical training and test set (Section 5.5).

5.1 Flexible candidate for k

`switchBox` allows for any arbitrary range of candidates for the number of pairs used in the classifiers. This is controlled using the argument `krange`, which represents the candidate pair numbers. In some cases, where top scores are not very significant, a wider range of k may help to implement a better predictor. In the example below we force the kTSP to choose between 11 or 13 pairs

```
> ### Show the arguments of the training function
> args(SWAP.KTSP.Train)

function (inputMat, phenoGroup, krange = c(3, 5, 7:10), FilterFunc = SWAP.Filter.Wilcoxon,
    RestrictedPairs, ...)
NULL

> ### Train the classifier
> classifier <- SWAP.KTSP.Train(inputMat = matTraining,
                                phenoGroup = trainingGroup, krange = c(11, 13))

Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
11 TSP will be used to build the final classifier.

> ### Apply the classifier to the test set of samples
> swapk_11_13_pred <- SWAP.KTSP.Classify(matTesting, classifier)
> ### Show the contingency table
> table(swapk_11_13_pred, testingGroup)

                testingGroup
swapk_11_13_pred Bad Good
            Bad   35  104
            Good  12  156

> ### Calculating classification accuracy
> TRUEPred <- (swapk_11_13_pred == testingGroup)
> Acc11_13 <- (mean(TRUEPred[which(testingGroup==levels(testingGroup)[1])]) +
              mean(TRUEPred[which(testingGroup==levels(testingGroup)[2])]))/2
> Acc11_13
```

```
[1] 0.6723404
```

The accuracy of this classifier is 0.672, which is better than the accuracies achieved by default ranges for both packages (Section 5.5). This shows that in some cases, a greater flexibility for the range of k can help improve the biomarker's prediction accuracy. On the other hand, in ktspair, the user can only set a single even pair number or let the program choose from a fixed set $\{3, 5, 7, 9\}$ using cross-validation.

5.2 Vote aggregation

In the classification phase, switchBox allows to aggregate individual pair votes according to any user defined voting scheme instead of the standard thresholding majority vote, the only option in ktspair and the default in switchBox. In switchBox, this choice can be implemented by setting the argument DecisionFunc of the SWAP.KTSP.Classify function. The code below provides several different examples of the way individual TSP votes can be aggregated in the final decision rule.

The arguments to the (SWAP.KTSP.Classify) function are shown below:

```
> args(SWAP.KTSP.Classify)

function (inputMat, classifier, DecisionFunc)
NULL
```

The following code chunk uses the default parameters to set the outcome to be the count of the signed TSP votes:

```
> ### Train a classifier
> classifier <- SWAP.KTSP.Train(matTraining, trainingGroup,
                           FilterFunc = NULL, krange=8)

No feature filtering procedure will be used...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
8 TSP will be used to build the final classifier.

> ### Compute the statistics using the default parameters:
> ### counting the signed TSP votes
> ktspStatDefault <- SWAP.KTSP.Statistics(inputMat = matTraining,
     classifier = classifier)
> ### Show the components in the output
> names(ktspStatDefault)

[1] "statistics"  "comparisons"

> ### Show some of the votes
> head(ktspStatDefault$comparisons[ , 1:2])

               GNAZ_Hs.555870>Contig32185_RC_Hs.159422
Training1.Bad                                    FALSE
Training2.Bad                                    FALSE
Training3.Good                                    TRUE
Training4.Good                                    TRUE
Training5.Bad                                    FALSE
Training6.Bad                                    FALSE
               Contig46223_RC_Hs.22917>OXCT_Hs.278277
Training1.Bad                                   FALSE
Training2.Bad                                   FALSE
Training3.Good                                   TRUE
Training4.Good                                   TRUE
Training5.Bad                                    TRUE
Training6.Bad                                   FALSE
```

```
> ### Show default statistics
> head(ktspStatDefault$statistics)

 Training1.Bad  Training2.Bad Training3.Good Training4.Good  Training5.Bad
            -6             -6              6              6              0
 Training6.Bad
            -2
```

The following code chunk uses the sum to aggregate the TSP votes:

```
> ### Compute the classifier statistics
> ktspStatSum <- SWAP.KTSP.Statistics(inputMat = matTraining,
     classifier = classifier, CombineFunc=sum)
> ### Show statistics obtained using the sum
> head(ktspStatSum$statistics)

 Training1.Bad  Training2.Bad Training3.Good Training4.Good  Training5.Bad
             1              1              7              7              4
 Training6.Bad
             3
```

The following code chunk sets the outcome to binary, having a value of TRUE when more than two TSP pairs are TRUE:

```
> ### Define an aggregation function
> sumGreaterThan2 <- function(x) {
         sum(x) > 2
         }
> ### Compute the classifier statistics
> ktspStatThreshold <- SWAP.KTSP.Statistics(inputMat = matTraining,
     classifier = classifier,  CombineFunc = sumGreaterThan2)
> ### Show statistics obtained using the threshold
> head(ktspStatThreshold$statistics)

 Training1.Bad  Training2.Bad Training3.Good Training4.Good  Training5.Bad
         FALSE          FALSE           TRUE           TRUE           TRUE
 Training6.Bad
          TRUE
```

5.3 Compute the signed TSP scores

The `switchBox` allows also to compute the scores for each gene pair of interest. This can be achieved by using the `SWAP.CalculateSignedScore` function as shown below.

Compute the scores using all features for all possible pairs:

```
> ### Compute the scores using all features for all possible pairs
> scores <- SWAP.CalculateSignedScore(matTraining,  trainingGroup, FilterFunc=NULL)

No feature filtering procedure will be used...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.

> ### Show scores
> class(scores)

[1] "list"

> ### computed statistics
> names(scores)

[1] "score"  "P"      "Q"      "labels"

> ### The score matrix dimentions
> dim(scores$score)

[1] 70 70
```

Figure 2 shows the score distribution for all possible gene pairs in the training data. For more examples please see the package vignette.

```
> hist(scores$score[upper.tri(scores$score)], main="Histograms of TSP scores",
      xlab="TSP score", col="salmon")
```

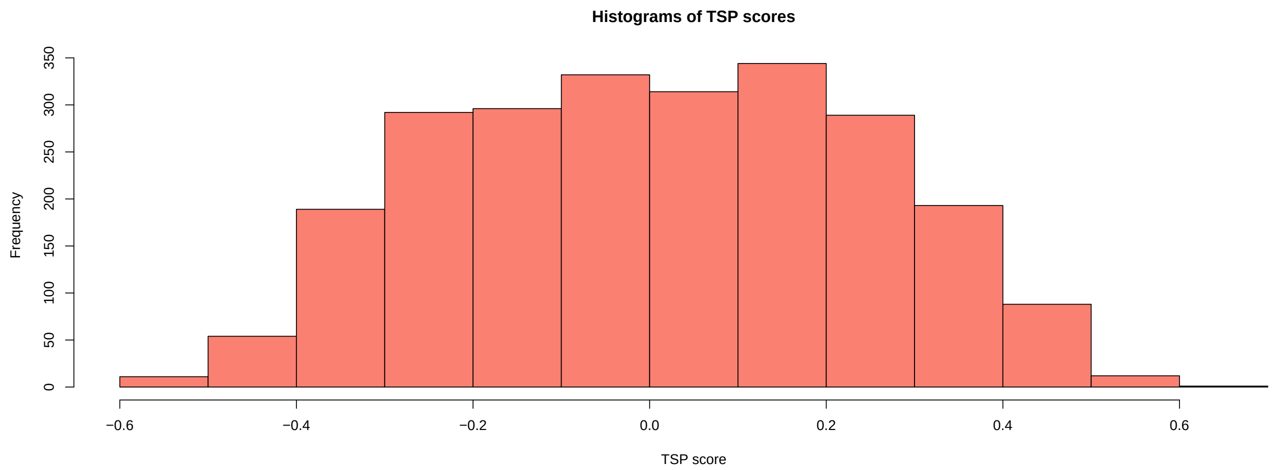

Figure 2: TSP score histogram for all possible gene pair in the training data.

5.4 Computational time comparison

Here we compare computation time running `ktspair` and `switchBox` with default options 10 times.

```
> ######################################################################################
> ### Using ktspair
> require(ktspair)
> ### Training KTSP ten times with ktspair
> ktspairTime <- system.time(
         for (i in 1:10) {
                 ktsp1 <- ktspcalc(matTraining, trainingGroup)
         })

The k-TSP cannot be computed for k =  7
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
Crossvalidation with k =  9  worked only for  2  partitions
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
Crossvalidation with k =  9  worked only for  2  partitions
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
```

```
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
Crossvalidation with k =  9  worked only for  2  partitions
The k-TSP cannot be computed for k =  7
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
Crossvalidation with k =  9  worked only for  1  partitions

> ### Training KTSP ten times with ktspair  takes the following amount of time
> ktspairTime

   user  system elapsed
   4.53    0.02    4.55

> ####################################################################################
> ### Training KTSP ten times with switchBox
> switchBoxTime <- system.time(
         for(i in 1:10) {
                 classifier <- SWAP.KTSP.Train(matTraining, trainingGroup, krange=c(3:10))
         })

Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
```

```
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.
Applying filtering function to inputMat...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.

> ### Training KTSP ten times with switchBox takes the following amount of time
> switchBoxTime

   user  system elapsed
   0.26    0.00    0.27
```

The time needed for training a KTSP classifier ten times using `ktspair` was 4.55 seconds, while the time needed with `switchBox` to accomplish the same task was 0.27 seconds on T530 Lenovo Laptop with Intell CORE i7 CPU.

5.5 Classifiers robustness

Below we show that the strategy for chosing the k pairs implemented in `switchBox` is more robust. In fact, since the `ktspair` package uses an inner loop of cross-validation for pair selection, threre is an inherent randomness in choosing the pairs. The following example shows how the identical training and testing data leads to different classifiers.

```
> ####################################################################################
> ### ktspair method using seed = 1
> ktspSeed1 <- ktspcalc(matTraining, trainingGroup, seed=1)

The k-TSP cannot be computed for k =  7
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  5
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7
Crossvalidation with k =  9  worked only for  2  partitions

> ### ktspair method using seed = 100
> ktspSeed100 <- ktspcalc(matTraining, trainingGroup, seed=100)

The k-TSP cannot be computed for k =  9
The value of k has been reduced to k =  7

> ### Printing ktspair classifiers for seed = 1
> summary(ktspSeed1)

There are  5 TSPs

Data for the k-TSP
                 Group Labels
Prediction Labels Bad Good
             Bad   31    5
             Good   3   39
```

```
> ktspSeed1

k-TSP object with: 5 TSPs
Pair:                TSP Score               Indices
TSP 1 :          0.6                         19 42
TSP 2 :          0.55                         24 58
TSP 3 :          0.53                         50 63
TSP 4 :          0.53                         13 22
TSP 5 :          0.53                         37 52

> ### Printing ktspair classifiers for seed = 100
> summary(ktspSeed100)

There are  7 TSPs

Data for the k-TSP
                 Group Labels
Prediction Labels Bad Good
             Bad   29    4
             Good   5   40

> ktspSeed100

k-TSP object with: 7 TSPs
Pair:                TSP Score               Indices
TSP 1 :          0.6                         19 42
TSP 2 :          0.55                         24 58
TSP 3 :          0.53                         50 63
TSP 4 :          0.53                         13 22
TSP 5 :          0.53                         37 52
TSP 6 :          0.52                         27 46
TSP 7 :          0.51                         64 69

> ### The classifiers trained with ktspair are not identical
> identical(ktspSeed1, ktspSeed100)

[1] FALSE

> ####################################################################################
> ### switchBox method using seed = 1
> set.seed(1)
> swapktspSeed1 <- SWAP.KTSP.Train(matTraining, trainingGroup, FilterFunc = NULL)

No feature filtering procedure will be used...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.

> swapktspSeed1$TSPs

     [,1]                        [,2]
[1,] "GNAZ_Hs.555870"            "Contig32185_RC_Hs.159422"
[2,] "Contig46223_RC_Hs.22917"  "OXCT_Hs.278277"
[3,] "RFC4_Hs.518475"            "L2DTL_Hs.445885"
[4,] "Contig40831_RC_Hs.161160" "CFFM4_Hs.250822"
[5,] "FLJ11354_Hs.523468"        "LOC57110_Hs.36761"
[6,] "Contig55725_RC_Hs.470654" "IGFBP5_Hs.184339"
[7,] "UCH37_Hs.145469"           "SERF1A_Hs.32567"

> ### switchBox method using seed = 100
> set.seed(100)
> swapktspSeed100 <- SWAP.KTSP.Train(matTraining, trainingGroup, FilterFunc = NULL)

No feature filtering procedure will be used...
Computing scores for 70 features.
This will require enough memory for 2415 pairs.
Selecting K...
7 TSP will be used to build the final classifier.

> swapktspSeed100$TSPs
```

```
     [,1]                         [,2]
[1,] "GNAZ_Hs.555870"             "Contig32185_RC_Hs.159422"
[2,] "Contig46223_RC_Hs.22917"    "OXCT_Hs.278277"
[3,] "RFC4_Hs.518475"             "L2DTL_Hs.445885"
[4,] "Contig40831_RC_Hs.161160"   "CFFM4_Hs.250822"
[5,] "FLJ11354_Hs.523468"         "LOC57110_Hs.36761"
[6,] "Contig55725_RC_Hs.470654"   "IGFBP5_Hs.184339"
[7,] "UCH37_Hs.145469"            "SERF1A_Hs.32567"
```

```
> ### The classifiers trained with switchBox are identical
> identical(swapktspSeed1, swapktspSeed100)
```

```
[1] TRUE
```

From the example above it can be seen that different seeds (*i.e.* 1 and 100) for `ktspair` may lead to different pair numbers. Because the algorithm for pair selection in `switchBox` is deterministic, it does not depend on the seed. Note that in our example above with seed = 100, `ktspair` finds the same classifier as the `switchBox` function `SWAP.KTSP.Train`. So, we will now only focus on the two different KTSP classfiers – *i.e.* `ktspSeed1` and `swapktspSeed1` – and assess their performance in the test set.

```
> ##################################################################################
> ### Test set prediction using ktspair derived classifier
> ktspPred1 <- predict(ktspSeed1, dat=matTesting)
> ### Performance in the test set for ktspair
> table(ktspPred1 , testingGroup)
```

```
         testingGroup
ktspPred1 Bad Good
     Bad   26   94
     Good  21  166
```

```
> TRUEPred <- (ktspPred1 == testingGroup)
> ktspAcc1 <- (mean(TRUEPred[which(testingGroup==levels(testingGroup)[1])]) +
              mean(TRUEPred[which(testingGroup==levels(testingGroup)[2])]))/2
> ### Overall accuracy for ktspair in the test set
> ktspAcc1
```

```
[1] 0.5958265
```

```
> ##################################################################################
> ### Test set prediction using switchBox derived classifier
> swapktspPred1 <- SWAP.KTSP.Classify(matTesting, swapktspSeed1)
> table(swapktspPred1, testingGroup)
```

```
             testingGroup
swapktspPred1 Bad Good
         Bad   27   81
         Good  20  179
```

```
> ### Performance in the test set for switchBox
> TRUEPred <- (swapktspPred1 == testingGroup)
> swapktspAcc1 <- (mean(TRUEPred[which(testingGroup==levels(testingGroup)[1])]) +
                 mean(TRUEPred[which(testingGroup==levels(testingGroup)[2])]))/2
> ### Overall accuracy for switchBox in the test set
> swapktspAcc1
```

```
[1] 0.6314648
```

Now, we calculate the distribution of accuracies resulting from the stochastic algorithm in `ktspair` resulting from 100 random trials.

```
> ### Set the initial seed
> set.seed(1)
> ### Compute the classifier for 100 random inizialization
> ktspList <- lapply(1:100, function(...) ktspcalc(matTraining, trainingGroup) )
> ### Compute the accuracy on the test set for all classifiers
> ktspAccuracy <- sapply(ktspList, function(ktsptrain, ...) {
        ktspPred <- predict(ktsptrain, dat=matTesting)
        TRUEPred <- (ktspPred == testingGroup)
        acc <- (mean(TRUEPred[which(testingGroup==levels(testingGroup)[1])]) +
                mean(TRUEPred[which(testingGroup==levels(testingGroup)[2])]))/2
        return(acc)
 })
> ### Add K
> ktspAccuracy <- data.frame(Accuracy=ktspAccuracy,
                           K=sapply(ktspList, function(x, ...) { paste("K =", length(x$ktspscore)) }),
                           stringsAsFactors=FALSE)
```

In this example, the `switchBox` method results in a slightly better prediction performance compared to `ktspair` in the test set (accuracy of 0.631 for `switchBox` *versus* an average accuracy of 0.616 for `ktspair` respectively). In Table 1 summarizes the consistency of KTSP classifier using the `ktspair` package.

K	Frequency	Accuracy
K = 1	0.04	0.677
K = 3	0.13	0.615
K = 5	0.42	0.596
K = 7	0.41	0.631

Table 1: Consistency of KTSP classifier using the `ktspair` package. Classifiers were trained on the same dataset using a hundred different seeds. The table summarizes the frequency of occurrence of each K resulting from the training procedure along with the accuracy in the test set.

6 System Information

Session information:

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

- R version 3.0.1 (2013-05-16), x86_64-w64-mingw32
- Locale: LC_COLLATE=English_United States.1252, LC_CTYPE=English_United States.1252, LC_MONETARY=English_United States.1252, LC_NUMERIC=C, LC_TIME=English_United States.1252
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: Biobase 2.20.1, BiocGenerics 0.6.0, gplots 2.14.0, ktspair 1.0, switchBox 0.99.9, xtable 1.7-3
- Loaded via a namespace (and not attached): bitops 1.0-6, caTools 1.17, gdata 2.13.3, gtools 3.4.1, KernSmooth 2.23-10, tools 3.0.1

7 Literature Cited

References

[1] Luigi Marchionni, Bahman Afsari, Donald Geman, and Jeffrey T Leek. A simple and reproducible breast cancer prognostic test. *BMC Genomics*, 14:336, 2013.

[2] Donald Geman, Christian d'Avignon, Daniel Q Naiman, and Raimond L Winslow. Classifying gene expression profiles from pairwise mrna comparisons. *Stat Appl Genet Mol Biol*, 3:Article19, 2004.

[3] Aik Choon Tan, Daniel Q Naiman, Lei Xu, Raimond L Winslow, and Donald Geman. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics*, 21(20):3896–904, Oct 2005.

[4] Lei Xu, Aik Choon Tan, Daniel Q Naiman, Donald Geman, and Raimond L Winslow. Robust prostate cancer marker genes emerge from direct integration of inter-study microarray data. *Bioinformatics*, 21(20):3905–11, Oct 2005.

[5] Julien Damond. ktspair: k-top scoring pairs for microarray classification. *R package version*, 1(0), 2014.

[6] Annuska M Glas, Arno Floore, Leonie J M J Delahaye, Anke T Witteveen, Rob C F Pover, Niels Bakx, Jaana S T Lahti-Domenici, Tako J Bruinsma, Marc O Warmoes, René Bernards, Lodewyk F A Wessels, and Laura J Van't Veer. Converting a breast cancer microarray signature into a high-throughput diagnostic test. *BMC Genomics*, 7:278, 2006.

[7] Marc Buyse, Sherene Loi, Laura van't Veer, Giuseppe Viale, Mauro Delorenzi, Annuska M Glas, Mahasti Saghatchian d'Assignies, Jonas Bergh, Rosette Lidereau, Paul Ellis, Adrian Harris, Jan Bogaerts, Patrick Therasse, Arno Floore, Mohamed Amakrane, Fanny Piette, Emiel Rutgers, Christos Sotiriou, Fatima Cardoso, Martine J Piccart, and TRANSBIG Consortium. Validation and clinical utility of a 70-gene prognostic signature for women with node-negative breast cancer. *J Natl Cancer Inst*, 98(17):1183–92, Sep 2006.