Bioconductor's `tspair` package

Jeffrey Leek
Department of Oncology
Johns Hopkins University
email: `jtleek@jhu.edu`

October 28, 2009

Contents

1 Overview **1**

2 Simulated Example **2**

3 The `tspcalc` function **2**

4 The `tspplot` function **3**

5 The `tspsig` function **3**

6 Prediction with the `tspair` package **3**

7 Cross-validating the TSP approach **4**

1 Overview

The `tspair` package contains functions for calculating the top scoring pair for classification of high-dimensional data sets [1].
A top scoring pair is a pair of genes whose relative ranks can be used to classify arrays according to a binary phenotype. A top scoring pair classifier has three advantages over standard classifiers: (1) the classifier is based on the relative ranks of genes and is more robust to normalization and preprocessing, (2) the classifier is based on a pair of genes and is likely to be more interpretable than a more complicated classifier, and (3) a classifier based on a small number of genes lends itself to diagnostic tests based on PCR that are both more rapid and cheaper than classifiers based on a large number of genes.
This document provides a tutorial for using the `tsp` package. The package consists of the functions: `tspcalc` for identifying a top scoring pair based on a data matrix or expression set and a group variable, `tspplot` for plotting TSP objects, `tspsig` for calculating significance of TSPs, and `tsp.predict` for predicting the outcomes of new arrays based on a previously calculated TSP. As with any R package, detailed information on functions, their arguments and values, can be obtained from the help files. For instance, to view the help file for the function `tspcalc` within R, type `? tspcalc`. Here we will demonstrate the use of the functions in the `tspair` package to analyze a simulated expression experiment. We will also show how to calculate p-values for significance.

2 Simulated Example

We demonstrate the functionality of this package using simulated gene expression data. The data used in this analysis is included with the `tspair` package as the dataset `tspdata`. This data set consists of simulated data (the variable *dat*) for 1000 genes (in rows) and 50 arrays (in columns). The first 25 arrays correspond to the first group and the second 25 correspond to the second. A second variable gives the group indicator (*grp*) and a third variable is an expression set combining these two elements (*eSet1*).
To load the data set type `data(tspdata)`, and to view a description of this data type `? tspdata`.

```
> library(tspair)
> data(tspdata)
> dim(dat)

[1] 1000   50
```

3 The `tspcalc` function

The `tspcalc` function computes all pairs in the gene expression matrix achieving the top score described in Geman and colleagues (2004) [1], described briefly below. In the expression matrix, genes should be in rows and arrays in columns. Generally, the number of genes is much larger than the number of arrays. First we calculate the top scoring pair using the simulated gene expression matrix and the group indicator:

```
> tsp1 <- tspcalc(dat, grp)
> tsp1

tsp object with: 1 TSPs
Pair:                TSP Score         Tie-Breaker                Indices
TSP 1 :          0.64                  NA                                  5 338
```

The function `tspcalc` returns a tsp object. A tsp object consists of the following elements: *index* - an index giving the rows of the gene expression matrix that define a top scoring pair. If index has more than one row, each row corresponds to a different pair achieving the top score, *score* - the top scoring pair score, defined as: $|\Pr(X_i > Y_i|\text{Class } 1) - \Pr(X_i > Y_i|\text{Class } 2)|$ where X_i is the gene expression measurement for the first gene on array i and Y_i is the gene expression measurement for the second gene of the pair on array i, *grp* - the group indicator variable converted to a binary (0-1) variable, *gene1* - the data for all top scoring pairs concatenated in rows, and *labels* the group labels from the user-defined *grp* variable. If more than one top scoring pair achieves the same maximum score, then the unique TSP is determined by the tie-breaking score described in [2]. Briefly, each expression value is ranked *within its array*, then a rank difference score is calculated for each pair of genes.
It is also possible to calculate the top scoring pair from an expression set object, using either a group indicator variable as with the data matrix, or by indicating a column in the pData of the expression set.

```
> tsp2 <- tspcalc(eSet1, grp)
> tsp3 <- tspcalc(eSet1, 1)
```

4 The tspplot function

The tspplot accepts a tsp object and returns a TSP plot. The figure plots the expression for the first gene in the TSP pair versus the expression for the second gene in the TSP pair across arrays. The user defined groups are plotted in the colors red and blue. The score for the pair is shown across the top of each plot. If there is more than one TSP, hitting return will cycle from one TSP to the next.

```
> tspplot(tsp1)

Number of TSPs:  1
TSP 1
```

5 The tspsig function

The score from tspcalc can be interpreted as the average of sensitivity and specificity of the classifier in the data used to construct the TSP. But to get a legitimate measure of significance, some measure of uncertainty must be calculated using a permutation test, cross-validation, or application of the TSP to a new training set. Here we demonstrate how to use the functions in the tspair to calculate significance of a TSP.
The function tspsig tests the null hypothesis that no TSP exists in the data set by permutation. To calculate the p-value, the group labels are permuted B times and a null TSP score is calculated for each, the p-value is the total number of null TSP scores that exceed the observed TSP score plus one divided by $B + 1$. The function tspsig calculates the significance with a progress bar to indicate the time left in the calculation.

```
> out <- tspsig(dat, grp, B = 50, seed = 12355)

          |2%      |20%      |40%      |60%      |80%      |100%
Progress: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

> out$p

[1] 0.3529412

> out$nullscores

 [1] 0.64 0.60 0.68 0.64 0.60 0.64 0.64 0.60 0.64 0.68 0.60 0.64 0.68 0.68 0.60
[16] 0.64 0.68 0.64 0.68 0.64 0.64 0.72 0.64 0.60 0.64 0.60 0.64 0.60 0.60 0.68
[31] 0.68 0.64 0.64 0.64 0.64 0.68 0.64 0.72 0.68 0.72 0.64 0.64 0.64 0.68 0.64
[46] 0.72 0.64 0.68 0.64 0.72
```

6 Prediction with the tspair package

A major of the advantage of the TSP approach is that predictions are very simple and can be easily calculated either by hand or using the built in functionality of the TSP package. The function summary can be used to tabulate the results from the TSP. Type ?summary.tsp for details.

```
> summary(tsp1, printall = TRUE)
```

```
There are 1 TSPs

Data for TSP: 1
                                Group Labels
1(Gene Gene5  < Gene Gene338 ) diseased healthy
                         FALSE       22       6
                         TRUE         3      19
```

In this example, the expression value for "Gene5" is greater than the expression value for "Gene338" much more often for the diseased patients. So if new data were obtained, when the expression for "Gene5" is greater than the expression for "Gene338" we predict that the patient will be diseased. The `predict` can be used to predict group outcomes for new expression sets or data matrices. Type `?predict` for details. The user can input a TSP object and a new data matrix or expression set. The `predict` searches for the TSP gene names from the original `tspcalc` function call, and based on the row names or featureNames of the new data set identifies the genes to use for prediction. The `predict` function returns a prediction for each new array. If the tsp object includes more than one TSP, the default is to predict from the TSP achieving the highest tie-breaking score from Tan and colleagues [2], but the user may elect to predict from any TSP. Here the variables `dat2` and `eSet2` represent independent data sets that can be used to predict outcomes based on the TSP classifier defined above.

```
> predict(tsp1, eSet2)

 [1] "diseased" "healthy"  "healthy"  "diseased" "healthy"  "diseased"
 [7] "healthy"  "diseased" "diseased" "diseased" "healthy"  "diseased"
[13] "diseased" "healthy"  "healthy"  "diseased" "diseased" "diseased"
[19] "healthy"  "healthy"

> predict(tsp1, dat2)

 [1] "diseased" "healthy"  "healthy"  "diseased" "healthy"  "diseased"
 [7] "healthy"  "diseased" "diseased" "diseased" "healthy"  "diseased"
[13] "diseased" "healthy"  "healthy"  "diseased" "diseased" "diseased"
[19] "healthy"  "healthy"
```

7 Cross-validating the TSP approach

The cross-validation procedure described by Geman and colleagues [1] re-calculates the TSP classifier within each cross-validation loop. This scheme can be intuitively though of as cross-validating the TSP procedure, instead of the specific TSP classifier. To calculate the leave one out cross-validation error, each sample is left out, the TSP classifier is calculated, and the group of the left out sample is predicted. The estimated cross-validation error is the faction of incorrect predictions.

```
> narrays <- ncol(dat)
> correct.prediction <- rep(TRUE, narrays)
> for (i in 1:narrays) {
+     testdat <- dat[, -i]
+     testgrp <- grp[-i]
+     tsptest <- tspcalc(testdat, testgrp)
```

```
+     prediction <- predict(tsptest, dat)[i]
+     correct.prediction[i] <- prediction == grp[i]
+ }
> cv.error <- mean(correct.prediction == FALSE)
> cv.error

[1] 0.44
```

References

[1] D. Geman, C. A'vignon, D. Naiman, and R. Winslow. Classifying gene expression profiles from pairwise mrna comparisons. *Statist. Appl. in Genetics and Molecular Biology*, 2004.

[2] A.C. Tan, D.Q. Naiman, L. Xu, R.L. Winslow, and D. Geman. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics*, 21:3896–3904, 2005.

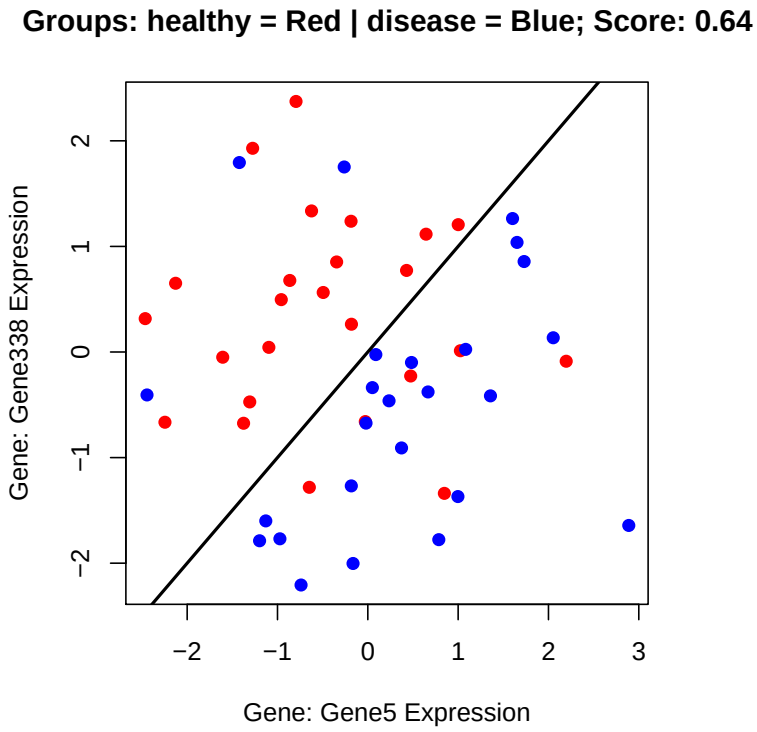

Figure 1: A top scoring pair plot from the simulated example.