Profile Analysis in **R**

version 1.0.1

Christopher David Desjardins

July 11, 2013

The purpose of this manual is to document the `criterion.pattern()` and `profile.cv()` functions presented in [1] and implemented in **R**. This manual begins with a brief introduction to reading data into **R**, then describes the two principal functions and how to use them, and concludes with examples of both functions that replicate the results presented in [1]. This manual assumes that the reader is familiar with this article. To more thoroughly understand these functions and their output, the reader should refer to that article.

Functions and arguments that can be sent to **R** are written in `this format` and libraries and references to objects and output are written in *this format*.

1 Reading data into R

There are myriad ways that an applied researcher in education might want to read data into **R**. For the sake of simplicity, I will assume that a researcher is managing their data in **SPSS** and wants to read their data into **R**. There are two principal ways that this could be done. The first is directly through the *foreign* library and the `read.spss()` function. The second, indirectly, is by exporting ones' data from SPSS to a .csv file and then using the `read.csv()` function. I will demonstrate both ways.

1.1 Reading SPSS data directly into R via the *foreign* library.

The *foreign* library must be loaded initially and then the `read.spss()` function needs to be called.This library should be installed by default by **R**. If it is not, then it can be installed by running `install.packages("foreign")`.

```
> library(foreign)
> sack <- read.spss("sackett5.sav",to.data.frame=T,use.value.labels=T)
```

This assigns the sackett5.sav file to *sack*, you are then able to directly manipulate the data file without needing to rerun the `read.spss()` function again. The `head()` command shows the first few records in your data file so that you can verify that the data was imported correctly.

1.2 Reading SPSS data indirectly into R via the `read.csv()` function.

To use this method it is assumed that you exported your data as a csv file in SPSS. This should be done in SPSS and not in **R**.

```
> sack.csv <- read.csv(file = "sackett5.csv")
```

This assigns the sackett5.csv file to *sack.csv* for later manipulation. After you have successfully read your data into **R** it is useful to save your data file in the native **R** file format, Rdata. To do this run the following:

```
> save(sack,file="sack.Rdata")
> load("sack.Rdata")
```

2 Functions

This section documents how to use the two principal functions, `criterion.pattern()` and `profile.cv()`. In order to parallel nomenclature already presented in Davison and Davenport (2002), object names are assigned the same names as those used by Davison and Davenport (2002) with the exception of a few objects.

2.1 The criterion.pattern() function

```
criterion.pattern(x,y,k)
```

The `criterion.pattern()` function requires three arguments: `x`, `y`, and `k`. The argument `x` corresponds to the predictor variables. This matrix of covariates may be in their own object or in the same object as the dependent variable. The `y` argument corresponds to the dependent variable. This vector may be in a separate object or in the same object as `x`. Finally `k` corresponds to the scalar constant. It is necessary to supply the `criterion.pattern()` function with these arguments.

As this function implements all of the algorithms described in Davison and Davenport (2002) with the exception of the cross-validation (provided in `profile.cv()`) the output is contained within the function is quite exhaustive and the user should refer to that article for more information about how each specific component is calculated.

The default output from `criterion.pattern()` consists of three parts: The regression weights, the vector of regression weight deviations around the mean *bstar*, the R^2 for the full model, for the level effect, and for the pattern effect. The default output also includes five F-test on these components: H_o: $R^2_{full} = 0$; H_o: $R^2_{pat} = 0$; H_o: $R^2_{lvl} = 0$; H_o: $R^2_{full} = R^2_{lvl}$; and H_o: $R^2_{full} = R^2_{pat}$.

Additionally output may be called from the `criterion.pattern()`. This includes the level component (*lvl.comp*) and the pattern component (*pat.comp*). *lvl.comp* provides a vector of scores corresponding to the level component. This is labeled as $\mathbf{X}_p$ in [1]. *pat.comp* provides the matrix of pattern component scores. This is labeled as $\mathtt{x}_p$ in Davison and Davenport (2002). *b* is a vector that contains the regression weights, this is not the deviation vector around the regression weight mean. The vector containing the regression weight deviations around the regression weight mean corresponds to *bstar*. This is labeled b* in [1] but it is programmed as *bstar* because **R** does not allow the use of the * key. *xc* corresponds to the criterion pattern based on *k* and can be calculated by multiplying *bstar* and *k*. The covariance between the actual predictor scores of person *p* and their scores in criterion-pattern vector (*Covpc)* is also available as is the vector of predicted scores, *Ypred*.

The examples below show how to run the functions, however, I recommend saving the output from the functions to an object so that you can easily retrieve them and in the case of `profile.cv()` so that results can be replicated. It is also necessary to store the output so that you can call the additional information available in the function.

2.2 The profilecv() function

```
profilecv(x,y)
```

The `profilecv()` function implements the cross-validation technique described in [1]. It requires two arguments: `x` and `y`. The argument `x` corresponds to the predictor variables. This matrix of covariates may be in their own object or in the same object as the dependent variable. The `y` argument corresponds to the dependent variable. This vector may be in a separate object or in the same object as `x`. There is no `k` argument for this function as `k` is irrelevant to this calculation.

There are five tables provided by this function: A F-table examining the full model, H_o:$R^2_{full} = 0$; a F-table examining the pattern effect, H_o: $R^2_{pat} = 0$; a F-table examining the level effect, H_o: $R^2_{lvl} = 0$; a F-table examining H_o: $R^2_{full} = R^2_{lvl}$; and a F-table examining H_o: $R^2_{full} = R^2_{pat}$. Within each of these table are estimates of the proportion of variance accounted for by the model/effect, the F-statistic, the degrees of freedom, and the p-values reported for each random sample. It should be noted that this function will never have the same output twice as each time `profile.cv()` is called, a new random split is made. If you want to get the same results it is necessary to use the `set.seed()` function.

Again, it is recommended that you save the output of this function to an object.

3 Examples

To demonstrate the use of the `criterion.pattern()` and the `profilecv()` functions, two data sets are provided. The first data set, labeled dd2002.Rdata, can be used to recreate table 1 in [1]. The second data set, sack.Rdata, can be used to recreate example 1 using the Sackett data set. The Sackett data set is not in the public domain and can not be shared.

3.1 criterion.pattern()

3.1.1 criterion.pattern() IPMM example

```
> ## Load the package
> library(profanal)
```

```
> ## Results presented in Table 1 load("dd2002.Rdata")
> IPMMc <- matrix(c(75, 60, 50, 50, 60, 75, 45, 55, 60, 60, 55, 45, 50,
           50, 75, 60, 45, 55, 60, 75, 55, 45, 60, 60),ncol=4,byrow=T)
> NP <- c(1,1,1,0,0,0)
> colnames(IPMMc) <- c("A","H","S","B")
> m0 <- criterion.pattern(IPMMc,NP,1)
> m0
```

```
(Intercept)           A           H           S           B
 0.50000000  0.00923077  0.02307692 -0.00923077 -0.02307692
```

```
         A           H           S           B 
 0.00923077  0.02307692 -0.00923077 -0.02307692 
                 R2
Full Model 0.969231
Pattern    0.969231
Level      0.000000
               F.statistic df.1 df.2   pvalue
R2full = 0           7.875    4    1 0.260419
R2pat = 0           10.500    3    1 0.222190
R2lvl = 0            0.000    1    1 1.000000
R2full = R2lvl      10.500    3    1 0.222190
R2full = R2pat       0.000    1    1 1.000000
```

```
> m1 <- criterion.pattern(IPMMc,NP,500)
```

Output from these functions are stored in *m0* and *m1*, respectively. Note that only the output from *m0* is shown. The output from *m1* can be easily called by typing *m1* in the R-console.

To evaluate what is in the *m1* and examine the pattern component (*pat.comp*).

```
> str(m1)
```

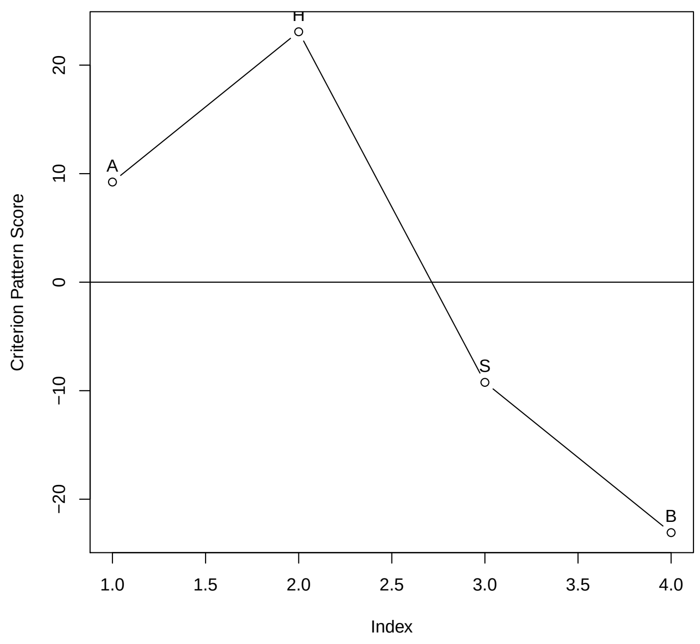
```
List of 10
 $ lvl.comp: num [1:6] 58.8 58.8 55 58.8 58.8 ...
 $ pat.comp: num [1:6, 1:4] 16.25 1.25 5 -8.75 -13.75 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "A" "H" "S" "B"
 $ b       : Named num [1:5] 0.5 0.00923 0.02308 -0.00923 -0.02308
  ..- attr(*, "names")= chr [1:5] "(Intercept)" "A" "H" "S" ...
 $ bstar   : Named num [1:4] 0.00923 0.02308 -0.00923 -0.02308
  ..- attr(*, "names")= chr [1:4] "A" "H" "S" "B"
 $ xc      : Named num [1:4] 4.62 11.54 -4.62 -11.54
  ..- attr(*, "names")= chr [1:4] "A" "H" "S" "B"
 $ k       : num 500
 $ Covpc   : num [1:6, 1] 57.7 75 49 -57.7 -75 ...
 $ Ypred   : Named num [1:6] 0.9615 1.1 0.8923 0.0385 -0.1 ...
  ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
 $ r2      : num [1:3, 1] 0.97 0.97 0
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "Full Model" "Pattern" "Level"
  .. ..$ : chr "R2"
 $ F.table :'data.frame':       5 obs. of  3 variables:
  ..$ F.statistic: num [1:5] 7.88 10.5 0 10.5 0
  ..$ df         : num [1:5, 1:2] 4 3 1 3 1 1 1 1 1 1
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:5] "full.df" "pat.df" "lvl.df" "pat.df" ...
  .. .. ..$ : NULL
  ..$ pvalue     : num [1:5, 1] 0.26 0.222 1 0.222 1
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:5] "p.value.F.R2.full" "p.value.F.R2.pat.only" "p.value.F.R2.lvl.only"
  .. .. ..$ : NULL
 - attr(*, "class")= chr "critpat"
```

```
> m1$pat.comp
```

```
          A      H      S      B
[1,]  16.25   1.25  -8.75  -8.75
[2,]   1.25  16.25 -13.75  -3.75
[3,]   5.00   5.00   0.00 -10.00
[4,]  -8.75  -8.75  16.25   1.25
[5,] -13.75  -3.75   1.25  16.25
[6,]   0.00 -10.00   5.00   5.00
```

To replicate the Figure 1 in [1]

```
> m2 <- criterion.pattern(IPMMc,NP,1000)
> plot(m2$xc, type="b",ylab="Criterion Pattern Score")
> abline(a=0,b=0)
> text(m2$xc,labels(m2$xc), cex=1, pos=3)
```



3.1.2 criterion.pattern() Sackett example

```
## Unfortunately not available to share but include to reproduce Davison & Davenport (2002)
> load("sack.Rdata")
> sack.m <- na.omit(sack)
> criterion.pattern(x = as.matrix(sack.m[, 2:7]), y = as.matrix(sack.m[,
+     1]), k = 100)


 (Intercept)       rtheme       itheme       atheme       stheme       etheme
41.449243157 -0.211675068  0.009353870 -0.005280127  0.308281646  0.103993332
      ctheme
-0.083369241
     rtheme      itheme      atheme      stheme      etheme      ctheme
-0.23189247 -0.01086353 -0.02549753  0.28806424  0.08377593 -0.10358664
                 R2
Full Model 0.133327
Pattern    0.129174
Level      0.006036
               F.statistic df.1 df.2   pvalue
R2full = 0        8.230315    6  321 0.000000
```

```
R2pat = 0          9.523090    5  321 0.000000
R2lvl = 0          1.949263    1  321 0.163631
R2full = R2lvl     9.429267    5  321 0.000000
R2full = R2pat     1.538263    1  321 0.215782
```

Th e `as.matrix()` command is necessary as `X` and `Y` are in the same object. Therefore, I am assigning column vectors 2 through 7 to `X` and column vector 1 to `Y`. In the first example, *m0*, you can see that the predictor variables and the dependent variable are stored in different objects. In both examples, `k` is assigned on the fly.

As shown in the first example, `criterion.pattern()` will assume that the first argument you give is for `X`, the next for `Y`, and the final argument for `k`. You do not have to specify these arguments in this order but if you do not specify them in this order then you need to use `X=` and so on.

3.2 criterion.pattern()

```
> source("criterion_pattern-0.1.R")
> c0 <-profilecv(x=as.matrix(sack.m[,2:7]),y=as.matrix(sack.m[,1]))
> c0


$R2.full
                       R2 F.statistic df1 df2   pvalue
Random Sample 1 0.129565    23.96501   1 161 0.000002
Random Sample 2 0.077396    13.50603   1 161 0.000323

$R2.pat
                       R2 F.statistic df1 df2   pvalue
Random Sample 1 0.127511    23.67578   1 162 0.000003
Random Sample 2 0.073950    12.93650   1 162 0.000428

$R2.lvl
                       R2 F.statistic df1 df2   pvalue
Random Sample 1 0.004617    0.751455   1 162 0.387298
Random Sample 2 0.004861    0.791322   1 162 0.375020

$R2.full.lvl
                       R2 F.statistic df1 df2   pvalue
Random Sample 1 0.127511    23.25453   1 162 0.000003
Random Sample 2 0.073950    12.73638   1 162 0.000472

$R2.full.pat
                       R2 F.statistic df1 df2   pvalue
```

```
Random Sample 1 0.004617    0.382216   1 162 0.537288
Random Sample 2 0.004861    0.605098   1 162 0.437774
```

4 Additional output from `criterion.pattern()`

```
> m1$lvl.comp

[1] 58.75 58.75 55.00 58.75 58.75 55.00

> m1$pat.comp

          A      H      S      B
[1,]  16.25   1.25  -8.75  -8.75
[2,]   1.25  16.25 -13.75  -3.75
[3,]   5.00   5.00   0.00 -10.00
[4,]  -8.75  -8.75  16.25   1.25
[5,] -13.75  -3.75   1.25  16.25
[6,]   0.00 -10.00   5.00   5.00

> m1$b

(Intercept)           A           H           S           B
 0.50000000  0.00923077  0.02307692 -0.00923077 -0.02307692

> m1$bstar

          A           H           S           B
 0.00923077  0.02307692 -0.00923077 -0.02307692

> m1$xc

        A          H          S          B
  4.615385  11.538462  -4.615385 -11.538462

> m1$k

[1] 500

> m1$Covpc
```

```
          [,1]
[1,]  57.69231
[2,]  75.00000
[3,]  49.03846
[4,] -57.69231
[5,] -75.00000
[6,] -49.03846
```

```
> m1$Ypred
```

```
          1           2           3           4           5           6
 0.96153846  1.10000000  0.89230769  0.03846154 -0.10000000  0.10769231
```

```
> m1$r2
```

```
                 R2
Full Model 0.969231
Pattern    0.969231
Level      0.000000
```

```
> m1$F.table
```

```
               F.statistic df.1 df.2   pvalue
R2full = 0           7.875    4    1 0.260419
R2pat = 0           10.500    3    1 0.222190
R2lvl = 0            0.000    1    1 1.000000
R2full = R2lvl      10.500    3    1 0.222190
R2full = R2pat       0.000    1    1 1.000000
```

References

[1] M.L. Davison and E.C. Davenport. Identifying criterion-related patterns of predictor scores using multiple regression. *Psychological Methods*, 7(4):468–484, 2002.