Tutorial for the opair package

Rune Haubo B Christensen

October 6, 2013

file: `opair_tutorial.Rnw`

1 Introduction

The opair package contains functions that facilitate analysis of ordinal paired comparisons.

The `opair` function fits the Thurstonian model and estimates d'. It also provides confidence intervals for d' and p-values for difference and similarity tests. The `plot` method for `opair` objects provide dot plots or bar plots of the estimated d' values, and the `save.opair` function writes the coefficient table including d' values, confidence limits and p-values to a `csv` file. This allows the user to import d' values into, for example, Microsoft Excel and make custom plots here. The package also includes two sample data sets; `NV` and `BG`.

This tutorial explains the functionality in the opair package using examples.

All the functions have help pages or manual pages available with, for example `help(opair)`. This document is meant as a supplement — not a substitute to these pages.

The opair package is loaded with the following command:

```
R> library(opair)
```

This makes all the functions in the opair package available to us. Note that the package automatically loads the ordinal package as well as a couple of additional packages. The ordinal package is used internally to compute the d' values. If `library(opair)` gives an error message, you probably need to install one or more packages on your computer. The opair package, for example, can be installed with

```
R> install.packages("opair")
```

which requires that you are one the internet. Note: This will only work when the package has been published on CRAN (`www.cran.r-project.org`).

Information about version of R and various packages used in this document is provided in Appendix˜A.

2 Estimation of d' with the `opair` function

The `opair` function estimates the Thurstonian model for ordinal paired comparisons. It takes the following arguments:

```
R> args(opair)

function (descriptors, products, d.equiv = 0.5, conf.level = 0.95,
    abbreviate.names = FALSE, ...)
NULL
```

where `descriptors` is a `data.frame` of descriptor variables, `products` is a factor indicating different products with the reference product being the first level, `d.equiv` is the point of equivalence used in computing the p-value of the equivalence test, `conf.level` specifies the confidence level (0.95 is the default), and `abbreviate.names` controls if names of the `descriptors` should be abbreviated to compact the printed output table.

As an example we will consider the `NV` data set (see `help(NV)` for details) that is included in the package. The data set has the following format (first six lines of the data set are shown):

```
R> head(NV)

  Taster Samples Red color Thickness Fruit quantity Smooth Compact Sweetness
1      1     432         0         0              0      0       0         0
2      1     568         0         0              0     -1       0         0
3      1     841        -1         0              0      0       0         0
4      2     432        -1         0              1      0       0         0
5      2     568        -1         1              1     -1       0         1
6      2     841        -1         1              1     -1       0         0
  Acidity Fruit intensity Global Dairy Off notes Lasting fruit Lasting sweet
1       0               0            0         0             0             0
2       0               0            0         0             0             0
3       0               1            0         0            -1            -1
4       1               0            0         0             0             1
5       1               0            0         1             0             1
6       1               1            0         0             1             0
```

The data is organized such that each row corresponds to a single trial or sample being evaluated on a number of attributes. Here the `Taster` column indicates which of 13 tasters provided the ratings, the `Samples` column indicates the three products involved in the experiment (432 is the reference product), and the remaining 12 columns hold the ratings for each of the 12 attributes or descriptors. The ratings take on the values (–2, –1, 0, 1, 2) corresponding to (*much less*, *less*, *same*, *more*, and *much more*).

We can now use the `str` function to reveal the internal structure of the data set:

```
R> str(NV)

'data.frame':        33 obs. of  14 variables:
 $ Taster         : Factor w/ 11 levels "1","10","11",..: 1 1 1 4 4 4 5 5 5 6 ...
 $ Samples        : Factor w/ 3 levels "432","568","841": 1 2 3 1 2 3 1 2 3 1 ...
 $ Red color      : int  0 0 -1 -1 -1 -1 -1 0 -1 1 ...
 $ Thickness      : int  0 0 0 0 1 1 -1 -1 2 0 ...
 $ Fruit quantity : int  0 0 0 1 1 1 -1 0 1 1 ...
 $ Smooth         : int  0 -1 0 0 -1 -1 0 -1 0 0 ...
 $ Compact        : int  0 0 0 0 0 0 -1 0 1 0 ...
 $ Sweetness      : int  0 0 0 0 1 0 0 -1 -1 0 ...
 $ Acidity        : int  0 0 0 1 1 1 0 1 0 0 ...
 $ Fruit intensity: int  0 0 1 0 0 1 -1 0 0 0 ...
 $ Global Dairy   : int  0 0 0 0 0 0 1 -1 -1 0 ...
 $ Off notes      : int  0 0 0 0 1 0 0 0 0 0 ...
```

```
 $ Lasting fruit  : int  0 0 -1 0 0 1 -1 1 0 0 ...
 $ Lasting sweet  : int  0 0 -1 1 1 0 0 1 0 0 ...
```

Here we see that the data set is stored as a `data.frame`. Note that `Samples` is stored as a factor and that the reference product 432 is the first level as required. Also note that the descriptor ratings are stored as integers and *not* as factors. The `opair` function will convert the variables into factors internally.

2.1 Estimation of d' for a single descriptor

We can use the `opair` function to estimate d' for a single descriptor of interest. Suppose we want to estimate d' for the `Thickness` descriptor, then we can extract a `data.frame` for `Thickness` with:

```
R> thick <- NV["Thickness"]
```

and check that it looks ok by displaying the first six rows with

```
R> head(thick)
```

```
  Thickness
1         0
2         0
3         0
4         0
5         1
6         1
```

We are now ready to fit the Thurstonian model and estimate d':

```
R> opair(descriptors=thick, products=NV$Samples, d.equiv=0.5,
+        conf.level=0.95)

Thurstonian model for ordinal paired comparisons


d-prime estimates:
 Descriptor Product d.prime   lower upper   p.diff p.equiv
  Thickness     568  0.2745 -0.9851 1.534 0.669261  0.3629
  Thickness     841  1.8927  0.5453 3.240 0.005903  0.9786
```

The output contains two d' values — one for each of the test products. Also provided is the two-sided 95% confidence limits for d', the two-sided p-value for the difference test (if `p.diff` is small, e.g.˜less than 0.05, there is evidence that d' is different from zero), and the two-sided p-value for the equivalence test (if `p.equiv` is small there is evidence that d' is larger than -`d.equiv` and smaller than `d.equiv`, where `d.equiv` = 0.5) based on the TOST method.

The equivalence region (-`d.equiv`, `d.equiv`), and the confidence level can changed as one may find appropriate.

Since `d.equiv=0.5` and `conf.level=0.95` are the defaults and because `descriptors` and `products` are the first two arguments, we would get the same result with

```
R> opair(thick, NV$Samples)
```

We could also extract the `Thickness` `data.frame` within the `opair` function:

```
R> opair(NV["Thickness"], NV$Samples)
```

2.2 Estimation of d' for multiple descriptors

The opair function can estimate d' for multiple descriptors at once. For example, the results for `Thickness` and `Sweetness` are provided by

```
R> opair(NV[c("Thickness", "Sweetness")], NV$Samples)

Thurstonian model for ordinal paired comparisons

d-prime estimates:
 Descriptor Product d.prime   lower upper   p.diff p.equiv
  Thickness     568  0.2745 -0.9851 1.534 0.669261  0.3629
  Thickness     841  1.8927  0.5453 3.240 0.005903  0.9786
  Sweetness     568  0.5860 -0.6736 1.846 0.361857  0.5532
  Sweetness     841  0.4332 -0.8195 1.686 0.497862  0.4584
```

We can also get the results for all the descriptors in the `NV` data set. First extract a vector of descriptor names and then fit the `opair` model to all 12 descriptors:

```
R> (desc.names <- names(NV)[4:ncol(NV)])

 [1] "Thickness"       "Fruit quantity"  "Smooth"          "Compact"
 [5] "Sweetness"       "Acidity"         "Fruit intensity" "Global Dairy"
 [9] "Off notes"       "Lasting fruit"   "Lasting sweet"

R> opair(NV[desc.names], NV$Samples)

Thurstonian model for ordinal paired comparisons

d-prime estimates:
      Descriptor Product d.prime    lower  upper   p.diff p.equiv
       Thickness     568  0.2745 -0.98508 1.5341 0.669261  0.3629
       Thickness     841  1.8927  0.54527 3.2401 0.005903  0.9786
  Fruit quantity     568  0.9738 -0.32046 2.2681 0.140299  0.7635
  Fruit quantity     841  1.5933  0.25225 2.9343 0.019878  0.9450
          Smooth     568 -0.8294 -2.16235 0.5036 0.222671  0.6859
          Smooth     841 -0.8590 -2.20003 0.4820 0.209295  0.7001
         Compact     568 -0.4825 -1.84022 0.8753 0.486130  0.4899
         Compact     841  1.4652  0.03723 2.8931 0.044319  0.9074
       Sweetness     568  0.5860 -0.67361 1.8456 0.361857  0.5532
       Sweetness     841  0.4332 -0.81945 1.6859 0.497862  0.4584
         Acidity     568  1.5632  0.09869 3.0277 0.036436  0.9226
         Acidity     841  1.5632  0.09869 3.0277 0.036436  0.9226
 Fruit intensity     568  1.1423 -0.20063 2.4853 0.095483  0.8257
 Fruit intensity     841  0.9718 -0.34627 2.2900 0.148437  0.7585
    Global Dairy     568 -0.4054 -1.72922 0.9185 0.548408  0.4443
    Global Dairy     841 -0.2059 -1.52493 1.1131 0.759619  0.3311
       Off notes     568  0.6035 -1.30613 2.5131 0.535662  0.5423
       Off notes     841  0.0000 -2.07676 2.0768 1.000000  0.3185
   Lasting fruit     568  1.1147 -0.22798 2.4573 0.103701  0.8152
   Lasting fruit     841  0.5592 -0.73910 1.8575 0.398574  0.5356
   Lasting sweet     568  1.1879 -0.13711 2.5130 0.078892  0.8456
   Lasting sweet     841 -0.1850 -1.45942 1.0895 0.776070  0.3140
```

If the `descriptors` or `products` have long labels, it can be difficult to view the table in its entire width. For that reason it is possible to abbreviate the labels of `descriptors` and `products` by setting the argument `abbreviate.names` to `TRUE`. For the `'Fruit quantity impression' descriptor` this looks like

```
R> opair(NV[desc.names[2]], NV$Samples, abbreviate.names=TRUE)

Thurstonian model for ordinal paired comparisons

d-prime estimates:
 Descriptor Product d.prime   lower upper  p.diff p.equiv
 Fruitqntty     568  0.9738 -0.3205 2.268 0.14030  0.7635
 Fruitqntty     841  1.5933  0.2523 2.934 0.01988  0.9450
```

Partial matching can be used, so

```
R> opair(NV[desc.names[2]], NV$Samples, abbrev=TRUE)
```

will give the same results.

3 Plotting d' values from the `opair` function

Two types of plots are directly available for illustrating d' values produced by the `opair` function; a dot plot and a bar plot.

To produce the plots, we first save the `opair` model in the object `fit`:

```
R> fit <- opair(NV[desc.names], NV$Samples)
```

Note that this does not print anything. Evaluating

```
R> fit
```

will however product the printed output shown above.

We can produce the dot plot with the following command:

```
R> plot(fit, type=1)
```

The result is shown in Figure˜1. Here labels and dots are blue for negative d' values and red for positive d' values.

The `plot` method for `opair` objects can also produce a bar plot using `type = 2`:

```
R> plot(fit, type=2)
```

The result is shown in Figure˜2. Here, also, bars are blue for negative d' values and red for positive d' values. Bars are labeled by consecutive numbers.

4 Saving d' values to a `csv` file

The function `save.opair` will save the coefficient table including d' values, confidence limits and p-values to a `csv` file.

The results from above can be saved with

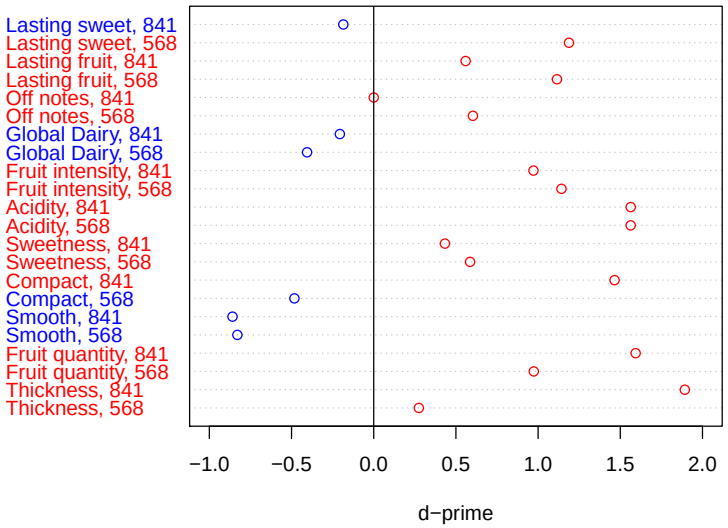

Figure 1: Illustration of d' values in a dot plot.

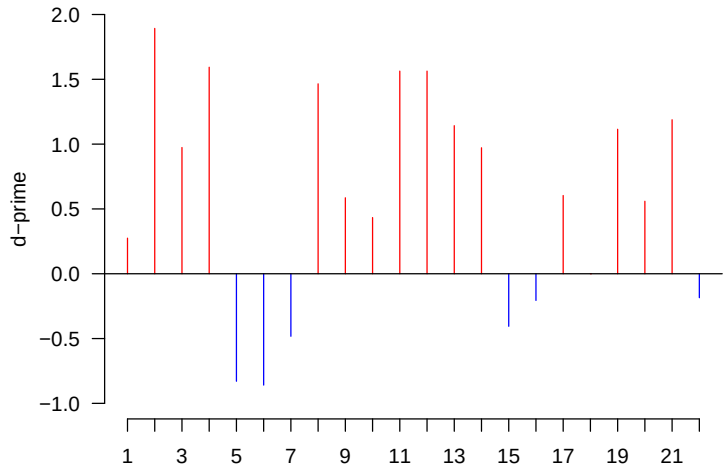

Figure 2: Illustration of d' values in a bar plot.

```
R> save.opair(fit, "myDprimes")
```

This will produce the file `myDprimes.csv`, and it will put the file in the working directory. If you do not remember where that is, you can find out by running

```
R> getwd()
```

By default the columns of the file will be separated by a comma, and the decimal operator will be a dot. This can be changed, though, for instance, the following will produce a semi-colon separated file where a comma is used as decimal separator.

```
R> save.opair(fit, "myDprimes", sep=";", dec=".")
```

Microsoft Excel will open csv files correctly if the column and decimal separators are selected appropriately. The right setting depends on the language or regional settings in Microsoft Excel (for example, `sep=";", dec="."` works in the Excel settings for Denmark).

If you want to save the file somewhere else, you can supply the path to the folder where you want to save the file. For example, to save the file in the root of the C-drive, use

```
R> save.opair(fit, "C:/myDprimes")
```

5 Reading in data from spreadsheets to use with the `opair` function

To analyze data stored in Microsoft Excel spreadsheet, there are two main obstacles:

1. Getting the data into R
2. Making sure data are in the right format.

The are many ways to get data from a spreadsheet into R. I find that the easiest way is to first save the data as a comma separated csv file using the *save as* menu and then reading the data into R using the R function `read.table` with something like

```
R> myData <- read.table("myData", header=TRUE, sep=",")
```

If in doubt about how the data should be structured in the spreadsheet before loading into R, take a look at the data set produced by

```
R> write.table(NV, file="NV.csv", row.names=FALSE)
```

A SessionInfo

```
R> sessionInfo()
```

```
R version 3.0.2 Patched (2013-10-04 r64027)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

```
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] opair_0.2-0

loaded via a namespace (and not attached):
[1] MASS_7.3-29       Matrix_1.0-14     grid_3.0.2        lattice_0.20-23
[5] ordinal_2013.9-30 tools_3.0.2       ucminf_1.1-3
```