User Guide: SRinference (ver. 2.01)

Torben Schubert

Fraunhofer Institute for Systems and Innovation Research
Breslauer Str. 48, 76139 Karlsruhe (Germany)
Email: torben.schubert@isi.fhg.de, Phone: 0049 721 6809 357, Fax: 0049 721 6809 357

also: Karlsruhe University, Institute for Economic Policy Research (IWW)

1 Introduction

1.1 Using SRinference

SRinference is an add-on package for the open-access statistical environment R. SRinference contains functions for calculating tests and confidence intervals for technological and scientometric specialisation ratios. The statistical details are described and developed in Schubert and Grupp (2009) and will not be reconsidered here. Therefore the reader is referred to the literature.

1.2 Changes to ver. 1.1

Version 2.01 provides several improvements to the preceding version 1.1.[1] First, it provides several wrapper functions which shall facilitate the use of the SRinference functionality for large tables. Second, the

2 Installing SRinference

In order to use SRinference, the user must install the R main program, which is available under http://www.r-project.org. R is free of charge.

1) Download the zip-file from http://www.isi.fhg.de/p/mitarbeiter/tos.html and save it to your hard drive.

2) Start R by double-clicking on the desktop icon.

3) Choose `Packages` from the main menu and then select the option `Install from local zip file`. Browse to the location of the zipped SRinference library and select this file. Installing the package might take a while depending on your hardware configuration. R should eventually indicate that SRinference has been installed. If R displays an error message repeat the steps from above.

[1] Version 2.0 was an internal version, which is largely identical to version 2.01.

3 Using SRinference

SRinference comprises functionality for the inference for specialisation ratios provided in Schubert and Grupp (2009). In this article both simulation and analytical inference techniques are discussed, where all of them are implemented in SRinference.

For large samples the analytical tests will be preferred, while the simulation tests are better convergence rates, which should increase small sample performance. What is large depends on the skewness of the underlying counting distribution and the test performed.

However, as rule of thumb, we propose using the analytical when the sample size for a certain class is at least 200 and the subject under consideration should have at least 20 objects. For instance calculating the specialisation measure of Germany's patents in biotechnology would require that total biotech patents are above 200 and that Germany has at least 20 biotechnology patents. If the sample is small, SRinference provides bootstrapping procedures, which are more accurate but also are more time-consuming. In the next section the analytical tests are discussed. In Section 3.2 the bootstrap function is described.

3.1 The analytical tests

The first step to work with SRinference is to load it to the memory. This can be done by entering the following command in the main window of R:[2]

```
require(SRinference)
```

Now any function contained in SRinference may be accessed by R. The main function for performing the analytical tests is `SRtest`. Let's first have a look on the help for `SRtest`. Type:

```
?SRtest
```

A help window should open, which displays the syntax and the features. The section `Usage` shows the following entry:

---

[2] Note that R is case sensitive. So the command `require(srinference)` will produce and error.

```
SRtest(units, unitsrefperiod = Null, obs = 1, refunit =
  c(sum(units[, 1]), sum(units[, 2])), refunitrefperiod = Null,
  testtype = "OneSample", null = 0)
```

Basically this summarises the arguments and options of the function `SRtest`, where an important is `testtype`, indicating which test shall be performed. `"OneSample"` is selected by default.

3.1.1 One sample test

Turning to this easiest test first, `units` is the most important argument, because it contains the data to perform inference on. `refunit` is another important argument. It contains the values of the reference unit. If not specified, then `refunit` is assumed to be the sum of what has been passed to the unit argument. `obs` is a scalar indicating for which unit the calculation shall be done. E.g. if `obs=1`, then the specialisation ratio and the statistical inference will be done for the first unit, whose values are in the first row of the matrix passed to `SRtest` by the `units` argument. `Null` is the value under the Null hypothesis, which will often be 0 (no specialisation).

Let's have a look at an example.

The following lines, which should be typed into the R main window, give exemplary patent data for developing countries in the form of a matrix (`un`) and a vector (`ref`). Other names can of course be chosen.

```
un<-matrix(c(1625,6342,8609,26440,    4702,16018,
   11019,26183,    26227,58423,   2190,5945,    7112,16495,
   935,2433, 2449,5652,    822,2301, 17168,29576,   2809,7674,
   5206,19115),ncol=2, byrow=T)

ref<-c(343196,1211400)
```

To check that everything is correct, type:

```
un
```

R should print the following statement to the screen:

```
       [,1]  [,2]
 [1,]  1625  6342
 [2,]  8609 26440
 [3,]  4702 16018
 [4,] 11019 26183
 [5,] 26227 58423
 [6,]  2190  5945
 [7,]  7112 16495
 [8,]   935  2433
 [9,]  2449  5652
[10,]   822  2301
[11,] 17168 29576
[12,]  2809  7674
[13,]  5206 19115
```

Typing

```
ref
```

should result in the following statement:

```
[1]  343196 1211400
```

Now, use the following command:

```
SRtest(units=un,obs=3,refunit=ref, null=0, testtype="OneSample")
```

This commands R to calculate the specialisation ratio including inference for the one sample test for the Null of no specialisation, where observation 3 is used. The output should look as follows:

```
Specialisation Ratio for Subject 3    0.0355

Teststat. Value H0                    2.9193

p-Value (two-sided)                   0.0036

Std.deviation                         0.0122
```

This indicates that observation 3 is slightly overspecialised, but the result is still significant at the 0.36%-level. This results from the large numbers of observations.

3.1.2 Two sample test

The two sample test serves to compare specialisation ratios of two units. This will again be accomplished by `SRtest`, where we must set `testtype="TwoSample"`. Additionally we need to specify the observations to use. This will be done by `obs`, which is now a vector of length 2.

Type:

```
SRtest(units=un,obs=c(6,12),refunit=ref, null=0, testtype
  ="TwoSample")
```

The output looks as follows.

```
Specialisation Ratio for Subject 6     0.2567
Specialisation Ratio for Subject 12    0.2508
Difference                             0.0060
Teststat. Value H0                     0.2806
p-Value (two-sided)                    0.7790
Std.deviation                          0.0212
```

The results show that the specialisation of observations 6 and 12 (both about 0.25) are not significantly different.

3.1.3 Time Change Test

When testing for time change, we want to know, whether perceived change in specialisation over time is statistically significant. Therefore information for two periods has to be passed to `SRtest`. This will be done by the arguments `unitsrefperiod` and `refunitrefperiod`. First enter the needed data as follows:

```
unt<-matrix(c(1243,4599, 5533,17431 , 3695, 12202 ,    8642,
   20517,  16947, 35241,   2174,5557,    5674, 12414,    885,
   2222 , 2219, 4762,    685,  1954 , 18200,  31236 ,  2258 ,
   5973,   4371 ,14954),  ncol=2, byrow=T)

reft<-c(306825,1104084)
```

Then perform a time change test for observation 3 by the following command:

```
SRtest(units=un,unitsrefperiod=unt,obs=3,refunit=ref,refunitrefp
   eriod=reft, null=0, testtype="TimeChange")
```

The R output is:

```
Specialisation Ratio for Subject 3 in Second Period  0.0355
Specialisation Ratio for Subject 3 in First Period   0.0857
Difference                                          -0.0502
Teststat. Value H0                                  -2.7598
p-Value (one-sided)                                  0.0058
Std.deviation                                        0.0182
```

which indicates the decline from 0.0857 to 0.0355 is indeed a statistically significant change at the 0.58%-level.

3.1.4 Using the Wrapper Functions

Often the results are needed for many observations, and using the function `SRtest` would be cumbersome and tedious. Therefore, in SRinference 2.01 several wrapper functions are contained which shall make routine work more comfortable. These are `SRsummary`, `SRsummarytable` and `SRexttable`. Note that up to now, the short-cut wrapper functions only work for the one-sample test.

3.1.4.i SRsummary

Taking for instance the case from above, a complicated way of running one-sample tests for all subjects contained in the matrix `un`, would to run a test for each observation separately by using `SRtest` and cycling through the `obs`-object. Instead, we can use SRsummary, which only requires the following command

```
SRsummary(units=un, refunit=ref, null=0)
```

to obtain the desired outcome. It results in an output like this, where only the first four of 13 columns are depicted:

```
                          1      2      3       4
Specialisation Ratio -0.1001  0.1383 0.0355  0.3763
Teststat. Value H0   -4.7391 16.1229 2.9193 61.2522
p-Value (two-sided)   0.0000  0.0000 0.0036  0.0000
Std.deviation         0.0211  0.0086 0.0122  0.0061
```

3.1.4.ii SRsummarytable

SRsumary works well for this situation, but sometimes analyses for very large tables might be of interest. SRinference provides functionality to work on that as well. However, the data must have a very specific format.

Suppose we have the following table of patent data, where the countries (subjects) are on the columns correspond to countries (more generally subjects) and the rows correspond to the classes. In the last column we should have the reference unit and in the last row the country totals.

This should be imported R by one of the regular methods. Note, that the resulting R-object should be a matrix where the class and country names are the row and column names rather than data entries itself.

	AT	BE	BG	BR	CA	CH	CN	CY	CZ	Total
Electrical machinery, apparatus, energy	92	76	1	22	65	275	24		6	10831
Electronic components	46	47		3	40	124	14		2	10364
Telecommunications	85	92		8	229	236	17	1	1	17961
Audio-visual electronics	21	80	1	2	35	61	7		2	7251
Computers, office ma-chinery	78	170	1	19	136	246	30		2	22185
Measurement, control	112	89	4	14	90	462	17	1	8	14823
Medical equipment	80	103	2	22	92	394	25	1	6	14149
Optics	36	169	1	11	37	143	17		4	6651
Basic chemicals, paints, soaps, petroleum prod	143	627	1	34	150	591	54	1	14	22897
Polymers, rubber, man-made fibres	222	356	2	17	140	572	29		6	20047
Non-polymer materials	263	195	2	21	98	382	30	1	9	13819
Pharmaceuticals	176	308	11	24	229	720	59	1	21	24238
Energy machinery	76	39	3	14	30	197	22		4	7743
General machinery	149	97	4	28	71	372	18		4	10082
Machine-tools	93	26		3	23	218	8		6	5473
Special machinery	236	268	3	31	95	659	19		17	16840
Transport	185	88	9	21	88	277	27		8	15826
Metal products	155	66	1	7	29	296	14		7	6607
Textiles, wearing, leather, wood, paper, domesti	233	302	3	26	122	592	56		14	17093
Sum	2481	3198	49	327	1799	6817	487	6	141	264880

Suppose, the name of the matrix is datamat, then use the following command:

```
SRsummarytable(datamat)
```

to produce a table of containing the inference results for each matrix cell.

3.1.4.iii SRexttable

SRexttable is useful when the results are re-used in R. But most of the times, we simply want the results of SRsummarytable in an output file readable e.g. by Excel. In that case, it might be even more easy to call a command that imports a table and exports the results automatically.

A slightly different format is needed, because the class names have to be provided in a separate file.

Assuming that tablesr.txt is a tabstop-seperated text-file containing all the information of the previous table except for the class names and rnames.txt is a tabstop-seperated text-file that contains a single column of class names, then this can be done by using the following command:

```
SRexttable(input="C:/tablesr.txt",output="C:/rpa0405.txt",rnames
   =T, path="C:/rnames.txt")
```

The `output` argument contains the path, where the results are written.

IMPORTANT: In the exemplifying data table there are many empty cells. These are interpreted as 0s. If you want to indicate that there are missings, the entries should be stated explicitly as `NA`.

3.2 Simulation tests

When the sample size is small (see above), then the bootstrap simulations are recommended, since they have higher order precision. The function that performs any of the tests described previously in their respective bootstrap versions is SRboottest. More specifically it simulates confidence intervals. However, since a test that rejects whenever the Null-value is not part of a confidence interval with level $1-\alpha$ is itself a valid level-α test, handling tests explicitly is unnecessary.

In order to open the help window, type

```
?SRboottest
```

In the usage section we see that the only arguments not familiar are `iter` and `alpha`. The first denotes the number of bootstrap replications. Technically the precision rises with the value of iter. However, choosing large values will also increase computation time. Recommended are values between 1000 and 5000. Note, that `iter` has no default. `alpha` is the confidence level and may be chosen as required (default: 0.05). Everything else remains unchanged and will not be discussed here. If further examples are required type:

```
example(SRboottest)
```

which will run the help file examples.

References

Schubert, T., Grupp, H. (2009): Tests and Confidence Intervals for a Class of Scientometric, Technological and Economic Specialisation Ratios, Applied Economics, forthcoming