FIXMAHAL
A software for
Fixed Point Cluster analysis
by means of the Mahalanobis distance

Christian Hennig
Institut für Mathematische Stochastik
Universität Hamburg
Bundesstr. 55
D-20146 Hamburg, Germany
hennig@math.uni-hamburg.de

Abstract

FIXMAHAL is a software that performs Fixed Point Cluster (FPC) analysis by means of the Mahalanobis distance. FPC analysis is a technique for cluster analysis based on the identification of outliers. In general it is introduced in Hennig (1998). An FPC is a subset of a data set that does not contain any outlier, and all other points of the data set have to be outliers w.r.t. the FPC. FPCs can overlap and need not to be exhaustive. Outliers need not to be included in any FPC and FPCs remain stable under addition or deletion of outliers. FPC analysis does not require a specification of the number of clusters.

This paper should serve as a user's manual for the software FIXMAHAL. The software FIXREG for finding linear regression FPCs can be handled in a similar way.

1 Mahalanobis distance Fixed Point Clusters

FPC analysis is a tool for finding subsets (*clusters*) of a data set which are

- homogenous in the sense that they can be adequately described by some homogenous parametrical distribution which I call *cluster reference distribution*, and that they contain no outlier from this distribution,
- separated from the rest of the data in the sense that all other data points are outliers w.r.t. to the FPC.

The concept is explained in more detail in Hennig (1997, 1998).

The program FIXMAHAL tries to find clusters of the shape of some multivariate Gaussian distribution. That is, they should be adequatly described by a cluster reference distribution of the form $\mathcal{N}(\mu, \mathbf{\Sigma})$, $\mu \in I\!R^p$, $\mathbf{\Sigma} \in I\!R^{p \times p}$ positive semidefinite and they should be separated from the remaining data points.

Let $g \in \{0,1\}^n$ be some indicator vector and $\mathbf{X}(g)$ be the matrix of the data points $\mathbf{x}_i$ indicated by $g_i = 1$. Then the data subset $\mathbf{X}(g)$ is called *Fixed Point Cluster* (FPC) w.r.t. $\mathbf{X}$, if g is a fixed point of

$$f: \{0,1\}^n \mapsto \{0,1\}^n,$$
$$f_i(g) = 1\left[(\mathbf{x}_i - \hat{\mu}(\mathbf{X}(g)))'\hat{\mathbf{\Sigma}}(\mathbf{X}(g))^{-1}(\mathbf{x}_i - \hat{\mu}(\mathbf{X}(g))) \leq m\right]$$

with some prechosen constant c, called the *Mahalanobis cutoff* (e.g. $c = \chi^2_{p;1-\alpha}$, $\alpha = 0.01$, where α is called the *level* of the outlier identification). $\hat{\mu}(\mathbf{X}(g))$ and $\hat{\mathbf{\Sigma}}(\mathbf{X}(g))$ are the mean and UMVU covariance matrix estimators[1] based on the data subset $\mathbf{X}(g)$.

The function f is an outlier identifier (0 for outliers): A point is considered as an outlier w.r.t. a Gaussian distribution with parameters μ and $\mathbf{\Sigma}$ if it falls into the *outlier region* $\{(\mathbf{x}_i - \hat{\mu}(\mathbf{X}(g)))'\hat{\mathbf{\Sigma}}(\mathbf{X}(g))^{-1}(\mathbf{x}_i - \hat{\mu}(\mathbf{X}(g))) > c\}$. That is, a point is considered as outlier from some data subset $\mathbf{X}(g)$, if its Mahalanobis distance is larger than c (see Rousseeuw and Leroy (1987) for a discussion of this approach to outlier identification).

Therefore an FPC $\mathbf{X}(g)$ has the property of being exactly the set of non-outliers in $\mathbf{X}$ w.r.t. itself. In this sense, an FPC is homogenuous and separated from the rest and

[1]The Moore-Penrose inverse is used in case of singularity of $\hat{\mathbf{\Sigma}}(\mathbf{X}(g))$.

can therefore be interpreted as a cluster. Note that the FPC itself needs not to have a "‘Gaussian shape"’. The Gaussian reference distribution only defines what "‘outlier"’ and "‘separatedness"’ mean.

The constant c defines the tolerance level of the outlier identification. It is related to the probability α that a regular point from a homogenous $p-$dimensional Gaussian distribution is identified as an outlier. The smaller the level α, the larger the Mahalanobis cutoff c, and the larger c, the stronger separation is needed for some data subset $\mathbf{X}(g)$ to be an FPC.

Fixed point clusters are found by an algorithm that starts with some random or prechosen point configuration and converges towards an FPC. The algorithm must run many times to find all relevant FPCs of a data set.

In difference to other methods for cluster analysis, FPC analysis does not require a specification of the usually unknown number of clusters and it is able to treat outliers from the whole data adequately. Clusters may overlap and the data set may not be exhausted by clusters.

The program FIXMAHAL finds FPCs in data sets. It shows the points that form FPCs and the corresponding indicator vectors (that can be used e.g. by some visualization software), means and covariance matrix estimators. There are default settings for m and the number of algorithm runs that can be modified by the user. If there is some a-priori grouping of the data and the interest lies in the question if these groups correspond to well separated clusters in the data, the algorithm can be started with these groups. The same strategy is appropriate if the data set has a large dimension or very many points since random search needs so many iterations in these cases that the computing time can get excessive.

You can also compute the outliers w.r.t. a given data subset with FIXREG, see section 4.4 o).

2 Loading FIXMAHAL

FIXMAHAL can be downloaded from my web page:

`http://www.math.uni-hamburg.de/home/hennig/`

You can get the following files:

fixmahal.exe - MS-DOS executable file

fixmahal.tar.gz - source code, makefile, data example, `manmahal.tex`, unzip with

```
gzip -d fixmahal.tar.gz
tar -xvf fixmahal.tar
```

on UNIX and LINUX systems.

unzip.exe - unzip-program for MS-DOS

fixmahal.zip - source code, makefile, data example, `manual.tex`, unzip with

```
unzip fixmahal.zip
```

on systems where `unzip.exe` is avialable.

A `fixmahal`-directory will be created where the software is unzipped.

3 Before running FIXMAHAL

FIXMAHAL is written in C, i.e. it can be used on every system that contains a C-compiler. Under MS-DOS you can use the `FIXMAHAL.EXE`-File and start it with the command `fixmahal` without any compilation work.

Under UNIX/LINUX you have to compile an executable file first. This also holds if the transfer of the binary file `FIXMAHAL.EXE` did not work or if you want to change the limitation of the data set size of $n = 500$ and $p = 5$ that is implemented in `FIXMAHAL.EXE` due to the limited stack space of the Borland C-compiler.

For the compilation you need one of the following C-compilers on your system:

cc (Sun standard C-compiler:) Compilation by `make -f fixmahcc.mak`. Afterwards the program can be started by `fixmahcc`.

gcc (Gnu-compiler, installed on almost all UNIX and LINUX-systems:) Compilation by `make -f fixmahgc.mak`. Afterwards the program can be started by `fixmahal`.

bcc (Borland-compiler under MS-DOS; the older version `tcc` should also work well:) Compilation by `make -f fixmahbc.mak`. Afterwards the program can be started by `fixmahbc`.

Other C-compilers may work as well, but then you have to create your own `make`-file by plugging in your compiler/linker and the appropriate options into the lines

```
CC=
LN=

CC_OPTS=
LN_OPTS=
```

of one of the `.mak`-files from the WWW-installation.

The compilation needs the following files in the same directory (this will be done automatically if you download the software from the WWW):

```
fixdefs.h           lokglo.h
vfm.h               vfm.c
loklesen.h          loklesen.c
lokstan.h           lokstan.c
meancov.h           meancov.c
standard.h          standard.c
matrix.h            matrix.c
```

and the suitable `.mak`-file. You also will find the test data described below and the LaTeX-file of this manual.

You can modify the sizes of some variables - maximum size of the data set, maximum filename length for in- and output etc. - using the file `fixdefs.h` that looks like follows:

```
#define NKOR [number] /* Maximum n */
#define PKOR [number]  /* Maximum p; for regression maximum (p + 1) */
#define NLEN 50 /* Maximum filename length  */
#define VLEN 10 /* Maximum variable name length */

#ifndef _vecdef_
#define _vecdef_

typedef double vektor[PKOR];
typedef double kvektor[PKOR-1];

#endif
```

Take care of any memory restrictions caused by your compiler or your system. In principle there is no limitation of the data set size. I tried examples up to $n = 187000$, but this needs lots of memory even for the output, since casewise information is given for every cluster.

4 Running FIXMAHAL

4.1 In case of trouble...

The software FIXMAHAL was tested with many data sets on various systems. However, one can never be sure to avoid any remaining errors and system incompatibilities. Also the protection of the software against inadmissible input is weak. If you have any trouble, please check first if it could be a consequence of some wrong or inadmissible input. Surprising results and even program crashs can also stem from problems with the data set (e.g. if there are commata instead of decimal points). You can take option i) from section 4.4 to examine if your data was read properly. Too large values for `NKOR` and `PKOR` in `fixdefs.h` (see section 3 to modify) can result in stack overflow and memory errors. FIXMAHAL needs memory to store alle found FPCs. If there are too much of them, the output closes with

```
WARNING! Heap memory full! Program terminated.
Enlarge minimum cluster size or Mahalanobis cutoff to get less clusters.
```

In that case consider e) and h) in section 4.4.

In case of other problems, please contact me under `hennig@math.uni-hamburg.de`.

4.2 Data format

After having started the program, the first thing you have to specify is the format of your data. Your data need to be an ASCII-file. The values have to be separated by arbitrarily many spaces, carriage returns or TAB-characters. The program reads files casewise, i.e. the file has to contain all variable values of one case before giving the values of the next case. Usually it will have the form "lines are cases, columns are variables". Decimal points have to be points. Scientific notation (`1.2256e-8`) is possible. You have three options for the beginning of the file:

```
 1: Data starts with 1st line, manual input of n, p
 2: 1st line head,
    2nd line number of data points n, number of indep. variables p
 3: 1st line head, manual input of n, p
```

The following data set (`europe.dat`) contains a headline and a line indicating $n = 26$ and $p = 2$ (option 2). It is an excerpt from data set No. 363 of Hand et al. (1994) on percentages employed in different industries in Europe. The first variable is Agriculture, the second variable is Finance, the third variable indicates the country and the fourth is a grouping variable that was added for illustratory purposes.

```
Europe data
26 2
3.3         6.2     Bel    1
9.2         6.5     Den    1
10.8        6.0     Fra    1
6.7         5.0     WGer   1
23.2        2.8     Irl    1
15.9        1.6     Ita    1
7.7         4.6     Lux    1
6.3         6.8     Net    1
2.7         5.7     UK     1
12.7        4.9     Aus    0
13.0        5.5     Fin    0
41.4        2.4     Gre    0
9.0         4.7     Nor    0
27.8        2.7     Por    0
22.9        8.5     Spa    0
6.1         6.0     Swe    0
7.7         5.3     Swi    0
66.8        1.1     Tur    0
23.6        0.7     Bul    2
16.5        0.9     Cze    2
4.2         1.2     EGer   2
21.7        0.9     Hun    2
31.1        0.9     Pol    2
34.7        1.3     Rum    2
23.7        0.5     USSR   2
48.7        11.3    Yug    0
```

The data file may contain more cases and variables than those needed for the analysis. Missing values are possible. They must be indicated by a single character that is not allowed to be a number, e.g. "*" or ".".

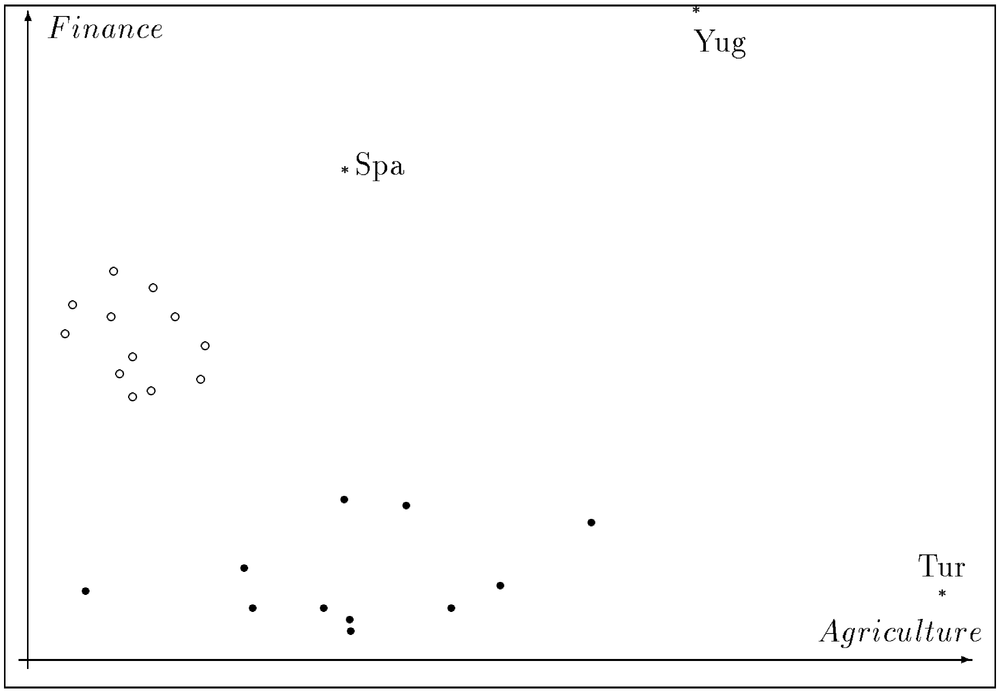

Figure 1: Europe data with FPCs 3 and 4, see section 5

4.3 File specifications

Now you have to enter the filename. The next questions of the program are:

```
Read an identificator variable (value length max. 10) ? (y/n)
Use a grouping variable for iteration starts ? (y/n)
```

An identificator variable is a variable that is used to indicate the cases in the output file. If no identificator variable is given, the cases are indicated by their position number in the data file. The values of the identificator variable have to be strings without spaces. In the `europe`-data set, the third variable shows abbreviations of the countries which identify the cases. A variable used for other purposes can also serve as identificator variable.

If you want to start the FPC algorithm from some prechosen point configuration, you can define a "grouping variable". The grouping variable must be non-negative integer-valued and can be contained in the data file or may be generated by use of the case numbers (see below). In the `europe`-example, the fourth variable will serve as a grouping variable. That is, there will be three iteration runs, started from the points indicated by the values 0,1,2, respectively, of the grouping variable additional to a number of iteration runs from random starting configurations.

Now you have to specify the name of the output file. It is important at least under MS-DOS that you do not give it an ending. The endings are added to the name by the program. FIXMAHAL creates a *main output file* that gets the ending `.fix` and for each FPC an additional *case information file* getting a number as ending. For the `europe`-data I entered the filename `europe` and got output files `europe.fix, europe.1, europe.2` and so on. In a particular situation you might get also files with the ending `.nnumber`, i.e. `europe.n1` and so on, see section 4.4 o).

If you made any wrong inputs up to now, break the program and start again. The following steps can be repeated.

Next the program asks you to give the total numer of variables in the input file (3 in the example). Afterwards you can enter the variable names and positions (variables, not characters are counted). For the `europe`-example:

```
Name of identificator variable    country
Position from the left ? 3
Name of 1th variable  agr
Position from the left of 1th independent variable:  1
Name of 2th variable  fin
Position from the left of 2th independent variable:  2
Name of grouping variable (values must be integer >= 0!) group
Generation of grouping variable by case number? (y/n) n
Position from the left of grouping variable:  4
```

If you decide to generate the grouping variable by case numbers, the program asks

```
Group [number] from case [previous input] to case [input number]
```

until the input number is greater or equal than n.

Now you have to enter the character that indicates a missing value. Enter an arbitrary character if the data does not contain missing values. All cases with missing values will be excluded from the analysis.

Answering "n" to the question `Everything correct ? (y/n)` gives rise to a repetition of the variable definition procedure.

4.4 Iteration specifications

Now the program shows the settings for the iterations (default settings are given in square brackets):

```
 Settings for cluster search:
 (*) indicates test functions for curious users; possibly unstable or useless.
     a) [100*3^(p-1)] runs with random start,
     b) (*) Increase factor of search probability for data not included in
              clusters: [1] (integer !),
     c) (*) Two levels for cluster search: [no]
     e) Standard level [0.002] (=>Mahalanobis cutoff c [$\chi^2_{p;1-level}$])
     f) (*) Number of data points for search start: [p+1]
     g) (*) Do you want to get a question for break all 20 runs: [no]
     h) Minimum size of clusters: [2*(p+1)]
     i) Output of standardized values (median and MAD): [no]
     j) (*) Do you want to get a question if a cluster is relevant: [no]
     k) (*) Do you want to enter points with decreased search probability: [no]
     m) Input of Mahalanobis cutoff for standard level
     n) Suppress standardization: [no]
     o) Iteration step limit: [min(n,5000)]
     q) Start of calculations
Enter a letter, if you want to change the default settings
```

During the development of the program I used some test functions that are presumably useless for the standard user. In order to that, they are not tested as good as the rest of the program. Nervertheless I decided to keep them in the program for curious users. They are not fully documentated. If you want to be as sure as possible that everything will work, feel discouraged to change the settings indicated by (*).

You can start the program immediately by entering “q”. The default settings can be changed by chosing one of the other letters. If you decide to change some number, you will be asked for the new value after having entered the letter. To change a “yes-no-variable”, you only have to enter the letter. The order of your changes is free. You may repeat and therefore cancel all your changes.

a) The default setting for the number of iterations with randomly chosen starting configuration is chosen so that the probability for finding an FPC of $\frac{1}{3}$ of the points of the data set stays approximatively the same for each value of p. If the cluster would be very well separated, i.e. it would be a clear pattern of the data set, this probability would be very high. The probability decreases with the size of the cluster relatively to n, but with the default choice you have already good chances to find clear separated FPCs down to $\frac{1}{5}$ of the data set.

There could be two main reasons for changing the default: If n is large, the interest in finding smaller clusters could be greater so that the value should be enlarged. If on the other hand p is too large, the default setting leads to too large - with $p > 4$ presumably unacceptable - computation times. The only feasible way to overcome this problem lies in the use of deterministic starting configurations from some reasonable grouping variable based on a-priori information about the data. The number of iterations with random starting configurations can then be set to 0; a small number of random iterations can only lead by chance to reasonable findings.

b) (*) If you want to find parts of a partition of the data set, it can be reasonable to exclude the points from the random search, which are already included in found FPCs. You can use option b) to increase the probability of points, that are not already parts of found FPCs, compared to the others. In my experience this does not lead to better results.

c) (*) If you change c), the program will always first iterate an FPC of a lower level[2] and the iteration with the *standard level* will always be started with the FPC of lower level. For the lower level clusters you will get a reduced output and so I suggest to run the program twice if you really are interested in the results using two distinct levels.

d), l) only appear if you have changed c) and correspond to e), m).

e) If you want to change the level[3] and the corresponding Mahalanobis cutoff c, you can enter the new level directly by “e”. Choose “m” to change the level via c. The respective corresponding other value will be calculated by the program. Experience

[2]The term “level” is explained in section 1.

[3]The term “level” is explained in section 1, the term “standard level” is chosen as opposed to the lower level, see c).

shows that in a small data set often more FPCs occur. This is because there are more clear gaps between points in that case. Note that a Gaussian distribution with unknown covariance matrix has $p + \frac{p(p+1)}{2}$ free parameters. That is, under $p = 2$ there are 5 free parameters to fit each data subset. If a data subset appears to be connected (i.e. there are no other points in its convex hull) and is fitted very good by the parameters (which happens often if the subset size is not large compared to the number of free parameters), the subset has a good chance to come out an an FPC.

Though a level of 0.002 often appears to be a good choice, I suggest to try smaller values (up to 0.1) for very large data sets, if too few FPCs were found, and smaller values to prevent a too large number of FPCs in small data sets. For the `europe`-data I chose a level of 0.0001, since a size of e.g. $26/2 = 13$ points for clusters of interest is very small to estimate 5 free parameters.

f) (*) By default, the random starting configuration has the least possible size. This leads to the best chance to find many clusters. If you change f), do not choose a smaller value. If you enter a larger value, you decrease the chance to find clusters with remarkably fewer points as n.

g) (*) If you change g), you will get a question all 20 runs if the iteration should be stopped.

h) The probability is high that some small point configurations can be fitted almost perfectly by the free parameters, especially if they lie almost exactly on some lower dimensional hyperplane. If you allow all these constellations to come out as an FPC, you can get a very large output. This holds especially for higher p. If the data set is small, you could be interested in smaller clusters nevertheless. If n or p are large, it could be reasonable to increase the value to see less FPCs. Note that in my experience the restrictions to some point group to come out as an FPC are remarkably smaller than using some usual partitioning method (see also the discussion in e)).

i) For decreasing the probability of numerical problems with very small or large values, FIXMAHAL standardizes all the variables to median 0 and median absolute deviation ("MAD") 1 (see Rousseeuw and Leroy (1987) for explanations). This does not affect the results because all parameter estimators will be re-standardized and FPC analysis satisfy the necessary equivariance properties. Choose "i" if you want to see the standardized values, the variable medians and MADs[4]. This could also be a way to test if your data was read properly.

j) (*) If an FPC is found, you can get a question if it is "relevant" in your opinion. If it is not, it will not appear in the output.

k) (*) You can choose "k" if you want to decrease the probability of some data points to get into the starting configuration. This means that the points are treated as

[4]If some variable has MAD$= 0$, i.e. more than half of the values are equal, the standardization will be omitted.

points of already found FPCs in b); change b) also to get a real decrease. The input procedure is not comfortable and this option is not tested very good.

m) see e).

n) see i). If n is large, the standardization takes a lot of time and option n) becomes reasonable. Also changing n) is a way to get unstandardized residuals into the case information output files (see i)). Note that this does not prevent the word "standardized" from appearing in the output. It must be ignored if the standardization has been suppressed.

o) The algorithm is proven to converge in a finite number of steps in the case $p = 1$ and it converges presumably also in higher dimensions. Nevertheless the convergence can fail because of rounding errors. Here is the number of iteration steps after which the program terminates the iteration. It is large enough that I expect the iteration to find an FPC certainly even with $n > 100000$ if it does not get trapped in some loop. If this happens, you will get the information in the output. If n is very large and your computer is slow or you do not have enough time, a smaller number could be reasonable.

Since the algorithm exceeds the step limit usually only when convergence to some FPC has finished and some single points get out of and into the configuration, FIXMAHAL produces a case information file (see section 5.2) also in the case that the step limit was exceeded. These files get the ending `.n1`, `.n2` and so on. This enables a further attractive feature of the program: If you want to compute the outliers w.r.t. to some given data subset, you can use some grouping variable for iteration starts, where the subset of interest appears as a group, and set the iteration step limit to 1. Then you get the outlier indicator vector (0 for outliers) in the corresponding case information file.

After having entered "q", the program starts the first iteration with the whole data set as starting configuration. This leads usually to some FPC that contains almost the whole data set (*WD-cluster*) and that also appears very often during the iterations with random starting configuration. If there are enough points in the complement of the WD-cluster, an iteration follows that starts with these points. Then the iterations with random starting configurations are carried out and at last the iterations with starting configurations determined by the grouping variable. The monitor shows you the state of the program.

5 Interpreting the FIXMAHAL-output

5.1 The main output file

The output consists of a main output file with the suffix `.fix` and case information files for each found FPC that have the endings `.[FPC-number]`. First it is advisable to consider the main output file. It is called `europe.fix` for the `europe`-example. The file starts with some informations concerning the data set and the iteration settings. You get the medians, MADs and standardized data values if you have changed setting i).

The file also contains the mean value and covariance matrix estimator of the whole data set. After that, the found FPCs, i.e. their points and parameters - sometimes very many - are shown in order of appearance.

I recommend to start the inspection of the output by taking a look at the end of the file for the `cluster similarity table`. This can help you considerably to understand your output. The table for the `europe`-data looks as follows[5]:

```
* Cluster similarity table *
Total number of standard level clusters: 9
Number of points in the intersection of standard level clusters no.

No.  Total    1    2    3    4    5    6    7    8    9
Times found  34  143   22   19   48    2    9    3    2

  1     26        24   12   11   25    9   14    8   12
  2     24             12   11   24    8   13    8   11
  3     12                   0   12    0   12    0    0
  4     11                        11    8    0    8   11
  5     25                              9   13    8   12
  6      9                                   0    8    9
  7     14                                        0    0
  8      8                                             8
  9     12

0 iterations did not finish fast enough.

22 iterations led to too small clusters.
```

The program found 9 FPCs. This seems to be a lot and not all are interesting; even more FPCs would be found if the level would not have been lowered to 0.0001. You can use the table to get some overview. The line `Times found` gives the number of times that the algorithm yielded the corresponding FPC. The column `Total` gives the number of points that belong to the FPC. The rest of the table gives the number of points in the intersection of the FPCs. The number of "Times found" is a measure for the stability of an FPC. If there are too many clusters, one should start to interpret the most stable FPCs. FPCs that occured only once are usually not of interest. Note, however, that it is more difficult to find a small cluster. The numbers of "Times found" are only adequate to compare FPCs of roughly the same size.

We see that cluster 1 that contains 26 points (i.e. all) came out 34 times of 304 runs (300 random + 1 whole data set + 3 groups as starting configurations). Cluster 1 is always the WD-cluster. The first guideline to the interpretion is that almost always one or more clusters are found most often which contain almost the whole data set. This happens since if the iteration starts with points that cannot be fitted adequately by the same cluster reference distribution, it usually converges to the WD-cluster or some very similar constellation. The corresponding FPCs are only interesting if one wants to find

[5]Recall that this is the result of iterations with random starting configurations. It will not necessarily be reproduced with the same data set and the same iteration settings.

outliers from the tendency of the whole data set. These will not be included at least in some of the FPCs that are similar to the whole data set.

The most stable other FPCs are the clusters 2, 3, 4, and 5. The table shows that the FPCs 2 and 5 contain the whole data set except one point, two points respectively. These constellations are clusters only in the sense that one can interprete the excluded points (`Tur` and `Yug`) as outliers from the whole data set.

FPC 8 is a subset of FPC 4 that is a subset of FPC 9. That is, we have three variants of the same data pattern and if we are only interested in a rough description of the data set, we can concentrate on FPC 4 that came out most often (filled circles in Fig. 1). If we want a more sophisticated interpretation, we can explain the points that belong to some but not all of these FPCs as lying "in the periphery" of the pattern. The next interesting cluster is FPC 3 (empty circles in Fig. 1). FPC 7 contains only 2 points more and can be interpreted as another variant of the same constellation. There Between the FPCs 3 and 4 there is no intersection, and so FPC 3 clearly describes a distinct pattern of the data. In fact, FPC 3 contains the western european countries with the advanced reduction of agricultural structures. FPC 4 consists mainly of eastern european countries, but also of more agricultural oriented western countries like Ireland and Portugal. Spain as well as Turkey and Yugoslawia do not appear in one of these FPCs.

The single clusters appear in the main output file as follows:

```
Successful run.


Output of standard level cluster no. 4:


Mean vector:
          agr          fin
   23.9818182     1.4454545

Covariance matrix :
          agr          fin
  100.2176364     2.4399091
    2.4399091     0.6767273



Cluster contains points
        Irl          Ita          Gre          Por          Bul          Cze
       EGer          Hun          Pol          Rum         USSR
Cluster contains 11 points
```

First the program shows the FPC mean vector and the UMVU-covariance matrix estimator for Gaussian distributions. Note that if the data would consist of only one cluster or there would be no overlap between clusters, the estimators for the single clusters would be biased because sometimes data points would be cut off by the outlier identification.

If you used a grouping variable, you get also the information what FPCs came out when the iterations were started from the groups above the cluster similarity table, e.g.

```
* Iteration start with group 1: *
* Iteration led to standard level-cluster no. 2. *
```

5.2 The case information files

The case information files, e.g. `europe.3`, look like this:

```
Case information for cluster no. 3

country - group - Mahalanobis distance from cluster - cluster indicator
         Bel       1         2.229423  1
         Den       1         2.183647  1
         Fra       1         1.501099  1
        WGer       1          1.11772  1
         Irl       1         29.07671  0
         Ita       1         31.64697  0
         Lux       1          2.17699  1
         Net       1         2.779329  1
          UK       1         2.718902  1
         Aus       0         2.482121  1
         Fin       0         2.545417  1
         Gre       0         108.6212  0
         Nor       0         1.564477  1
         Por       0          42.9965  0
         Spa       0         52.75312  0
         Swe       0        0.4864902  1
         Swi       0        0.2143849  1
         Tur       0         330.0299  0
         Bul       2         55.19835  0
         Cze       2         43.23071  0
        EGer       2         46.71948  0
         Hun       2         48.63078  0
         Pol       2         72.62398  0
         Rum       2         81.98747  0
        USSR       2          58.7499  0
         Yug       0         303.9825  0

group - number of cases from group in cluster
   0   5 (of 10)
   1   7 (of 9)
   2   0 (of 7)
```

If you want to visualize the findings of the FPC analysis, I recommend to add the values as new columns[6] to your data file and to use some software that is able to show e.g. scatterplots - possibly with more than two dimensions. The cluster indicator can be used to give the points of the FPC a particular color. The Mahalanobis distances can be used to assess the separatedness of the FPC from the rest of the data. Of course,

[6]Unfortunately most of the text editors that I know are not able to handle columns adequately. Under UNIX you can use the direct shell commands `cut` and `paste`, under MS-DOS I recommend the *q-editor*. With some effort, you can also do it with *EXCEL*.

identificators and groups appear only if you used an identificator or a group variable. The list `group - number of cases from group` helps you to evaluate the relation of your groups to the FPCs. You can see to what extent the groups correspond to clusters in the data.

6 References

Hand, D. J., Daly, F., Lunn, A. D., McConway, K. J., and Ostrowski, E. (Eds.) (1994). *A Handbook of Small Data Sets*, Chapman and Hall.

Hennig, C. (1997). Fixed Point Clusters and their Relation to Stochastic Models, in: *Classification and Knowledge Organization*, Klar, R. and Opitz, O. (Eds.), Springer, 20-28.

Hennig, C. (1998). Clustering and Outlier Identification: Fixed Point Cluster Analysis, in: *Advances in Data Science and Classification*, Rizzi, A., Vichi, M. and Bock, H.-H. (Eds.), Springer, 37-42.

Rousseeuw, P. J. and. Leroy, A. M. (1987). *Robust Regression and Outlier Detection*, Wiley.

All my papers are avialable from `http://www.math.uni-hamburg.de/home/hennig/`.