In search of the number of clusters

Foreword. Some of the current clustering algorithms need the preliminary knowledge of the expected number of clusters. Often, the solution is reached after a sequence of tentative cluster numbers.
Because even the language about clustering, like center, minimum deviance, etc. refer to a gravitational mindset, it may be interesting to frame the problem explicitly as a gravitational one trying to apply the Newton law to data.

A parallel algorithm.
Given the number of clusters, the first step some packages start with is the search of the so called “initial seeds”. I miss any info about the sensitivity of the solution to the set of seeds by package. I suspect, anyway, that the solution may depend on the set of initial seeds, because initial seeds are iteratively updated after the aggregation of a new point to the growing clusters around them. This family of algorithms is thus sequential and we know that going on sequentially may lead to different solutions when you start from different starting points.
Ideally, the algorithm *should be parallel* where at any step all points simultaneously contribute to the solution.
Well, the gravitational algorithm is parallel. Without knowing the number of clusters, you can imagine that each weighted point follows the Newton gravitational law

$$F = g\frac{Mm}{r^2}$$

Given a point having a weight M, you compute the gravitational force exerted on it by all the other points and let the Newton’s law to operate. For your target point, in the time interval dt, you have the acceleration a, the infinitesimal velocity dv and the infinitesimal displacement ds

$$a = g\frac{m}{r^2};\ \ dv = g\frac{m}{r^2}dt;\ \ ds = g\frac{m}{r^2}dt^2$$

To avoid going too fast, you need to state a maximum ds. A reasonable choice would be a fraction of the range of the variables that concur to clustering. Because g and dt are both constant, you can disregard g and act only on dt to reach your objective. Implicitly, you set $g = 1$.
The gravitational algorithm is parallel by its very nature, because all points are under the effect of all the other points. Any sequential shortcoming is

thus avoided and the points' cloud clusters around barycenter's that are created by the gravitational law in a manner much similar to what happened with the planetary system where asteroids converged because of the Newton's law and created planets.

A comparison. The natural case for a comparison is the Fisher's Irish problem. I'm currently using SAS so my comparison is with the SAS FastClus procedure. Because the species are known, the test is in the fit between the clusters and the species.

SAS. The case is taken from the SAS library of examples.

```
proc fastclus data=iris maxc=3 maxiter=10 out=clus;
   var SepalLength SepalWidth PetalLength PetalWidth;
run;

proc freq;
   tables cluster*Species;
run;
```

CLUSTER	Species	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	Versicolor	2	1.33	2	1.33
1	Virginica	36	24.00	38	25.33
2	Setosa	50	33.33	88	58.67
3	Versicolor	48	32.00	136	90.67
3	Virginica	14	9.33	150	100.00

SAS does a pretty good job. Setosa is 100% in the cluster 2, Versicolor is 96% in the cluster 3, except a couple of points that belong to cluster 1, while Virginica is split between the clusters 1 (72%) and 3 (28%).

Gravitational method.

First you need to compute the pairwise distances between all points. A self-join of the table *pmeno*, that keeps centered data, is enough. Centering is useful, though not needed. Newton laws have a Universal validity, you should translate nothing. Centering eases interpretation.

```
proc sql;
   create table grav.paira as
   select
a.seq as seqa, a.seplc as spl1, a.sepwc as spw1, a.petlc as ptl1, a.petwc
as ptw1,
```

```
b.seq as seqb, b.seplc as spl2, b.sepwc as spw2, b.petlc as ptl2, b.petwc
as ptw2
   from grav.pmeno as a, grav.pmeno as b
   where a.seq ne b.seq
   order by a.seq, b.seq
   ;
```

Variables names have been shortened and the termination *c* was added to mean *centered*. In addition, in the resulting table they are again shortened and distinguished by the parent table.
Now you delete points with a zero distance (you would get a *black hole*!)

```
data grav.notequal;
    set grav.paira;
if (spl1-spl2) eq 0 and spw1-spw2) eq 0 and ptl1-ptl2) eq 0 and
   (ptw1-ptw2) eq 0  then delete;
     run;
```

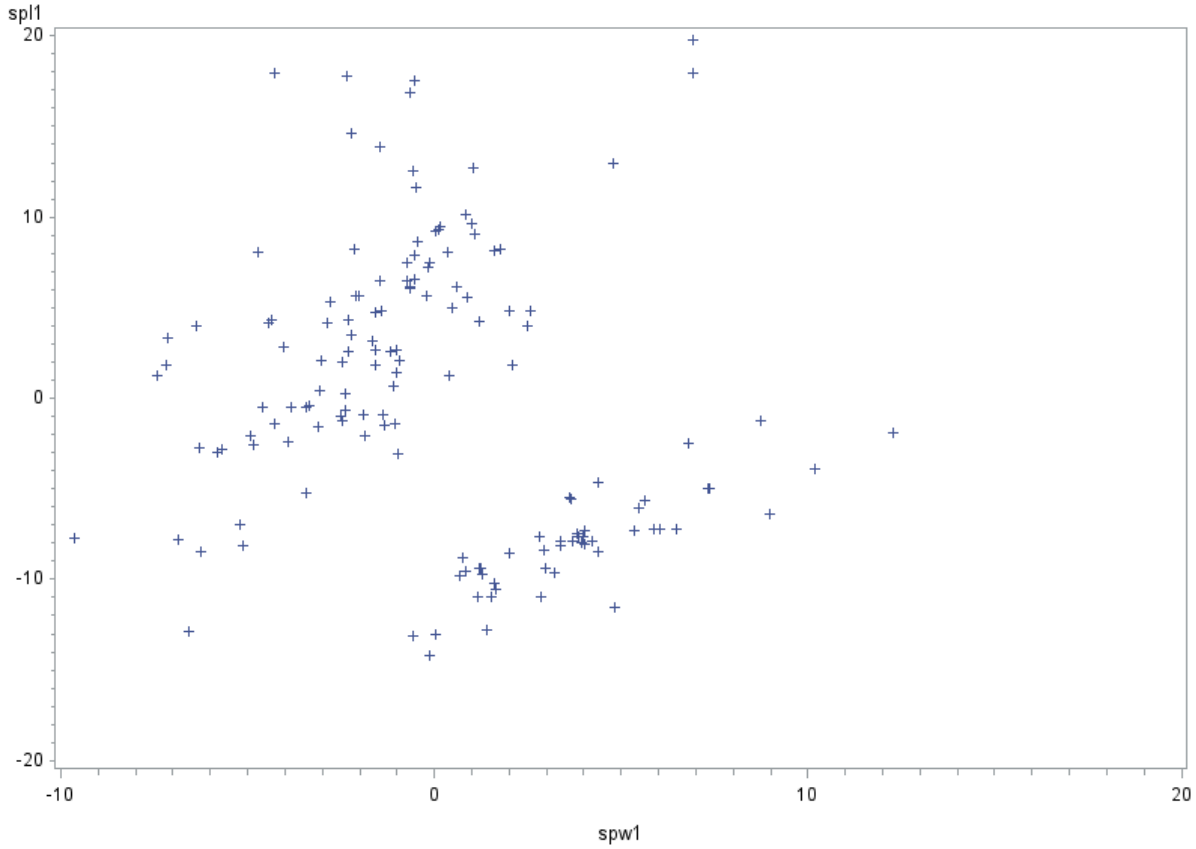
Now you compute the pairwise distances and displacements

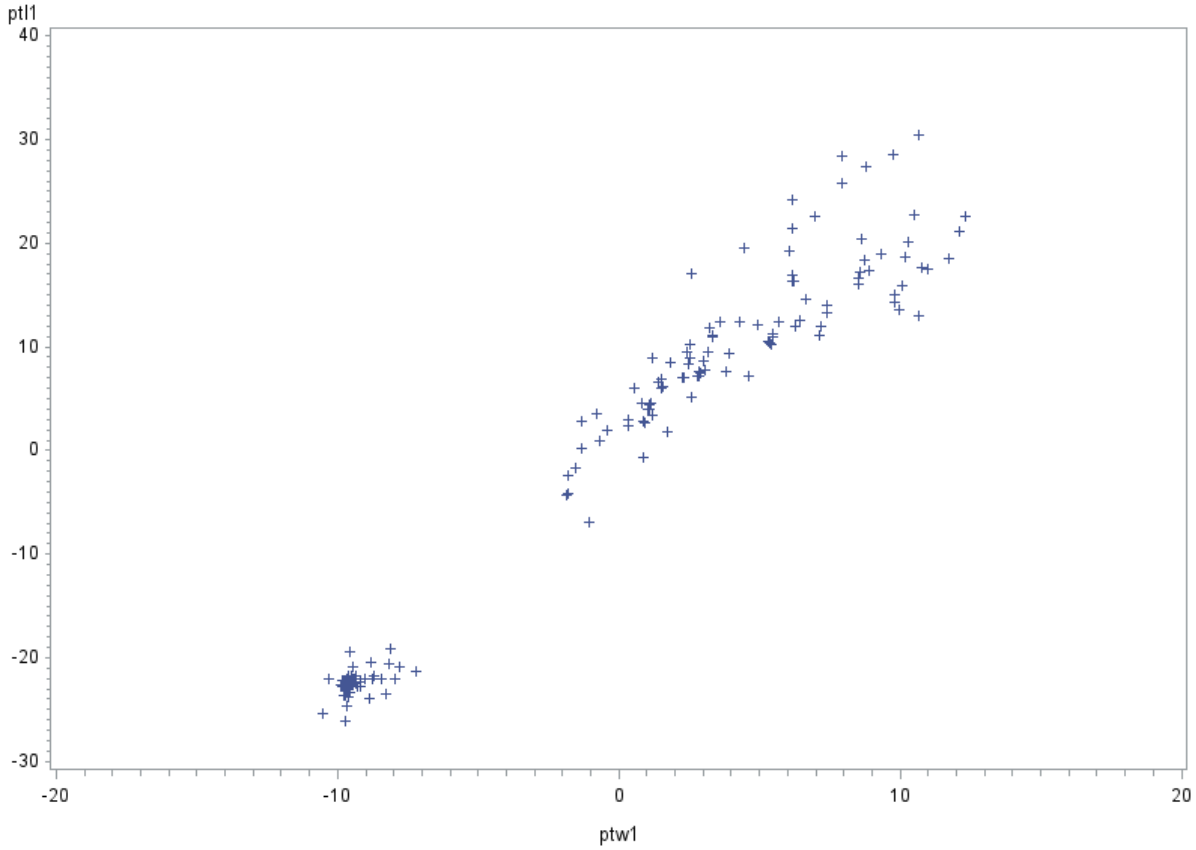
```
data grav.sqdist;
 set grav.notequal;
 dspl = spl1-spl2;
 dspw = spw1-spw2;
 dptl = ptl1-ptl2;
 dptw = ptw1-ptw2;
 sqdist = dspl**2 + dspw**2 + dptl**2 + dptw**2;
 dt = 0.5;
 M = 1;
 ds1 = -dspl*(M/sqdist)*dt**2;
 ds2 = -dspw*(M/sqdist)*dt**2;
 ds3 = -dptl*(M/sqdist)*dt**2;
 ds4 = -dptw*(M/sqdist)*dt**2;
 run;
```

Each point is subject to a force given by the sum of all forces exerted on it by the other points. You then sum the displacements by point

```
proc means data = grav.sqdist noprint;
by seqa;
var ds1 ds2 ds3 ds4;
output out = grav.step0a sum=;
run;
```

Gravitational forces have to be recalculated after each iteration because points shifts are not proportional to the original distances.
After just 3 iterations we come to the following point arrangement for sepals and petals


spl1
20
10
0
-10
-20
-10
0
10
20
spw1


ptl1
40
30
20
10
0
-10
-20
-30
-20
-10
0
10
20
ptw1

The clusters are found to be three without specifying them in advance and the cross tabulation is

Species	clust	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Setosa	**2**	50	33.33	50	33.33
Versicolor	**2**	4	2.67	54	36.00
Versicolor	**3**	46	30.67	100	66.67
Virginica	**3**	50	33.33	150	100.00

An almost perfect job.

The method can be easily applied to the disentanglement of normal components in a normal mixture.