Interdyme Report #3: A Note on "Right-Direction Scaling"

Douglas Meade - February, 1996

Introduction

Have you ever had a situation where you needed to scale a set of numbers to a control total, but those numbers were both positive and negative? Perhaps your first naive solution was to calculate:

$$S = \frac{C}{\sum X_i} \qquad (1)$$

where: S is the necessary scaling factor.
C is the control total.
X_i are the elements to be scaled.

However, you may have encountered a difficulty using this naive technique. To illustrate this, let's say you had the set of 4 numbers:

{ 99, -100, 50, -50}

which sum to -1. Assume the desired control total is +1. Then, the calculated scaling factor would be 1/-1 = -1, and the resulting set of elements would be

{-99, 100, -50, 50}.

Certainly these do sum to the control total, but great violence has been perpetrated on their relative positions on the number line!

Right-Direction Scaling to the Rescue!

The goal behind right-direction scaling is that all components of the set should preserve their relative ordering, and this means that all elements should be scaled in the same direction: the *right* direction. How is this achieved?

To illustrate, assume we have the following set of 5 observations:

{5, -2, 3, -4, 6} which sum to 8.

Assume that we are trying to scale these numbers up to that the total equals 10. The idea behind right-direction scaling is that we don't want to scale all numbers up in absolute value, but rather to scale all numbers in the same direction. The scaling algorithm multiplies the positive numbers by the desired scaling factor, and *divides* the negative numbers by the scaling factor.

Assume that

$A = \sum (positive\ elements)$
$B = \sum (negative\ elements)$
$C = Control\ Total$

We are trying to find the scaling factor S such that

$$AS + B/S = C \qquad (2)$$

This is essentially a quadratic equation in S, as we can see if we rewrite the equation as

$$AS^2 - CS + B = 0 \qquad (3)$$

To solve for S, we can use the general quadratic formula to obtain:

$$S = \frac{C + \sqrt{C^2 - 4AB}}{2A} \qquad (4)$$

If we use this value of S in our original problem, to multiply the positives and divide the negatives, we obtain conformance to our desired control total, while still preserving the original rank ordering of the elements to be scaled.

Using our example:

$C = 10$ (control total)
$A = 14$
$B = -6$

The formula for S yields

$$S = \frac{10 + \sqrt{100 - 4(14)(-6)}}{2(14)} = \frac{10 + 20.881}{28} = 1.1029$$

Multiplying the positives by this number, and dividing the negatives, yields the set: {5.5144, -1.8134, 3.3086, -3.6269, 6.6173} which can be verified to sum to 10.

In Interdyme, right-directional scaling is performed by the function `rdscale()`. The calling signature of this function is

```
rdscale(Vector v, int n, int group[], float control);
```

The vector `v` contains the elements to be scaled. A subset or group of the elements can be chosen, if you don't want to scale the whole vector. This group is indicated by the integer vector `group`, which contains a list of sectors or categories. The integer `n` is the number of elements to be scaled, and the variable `control` is the control total desired.

This `rdscale()` function is used by the vector fixing function of Interdyme, as you may imagine. However, it can be used generally anywhere that you need to perform scaling of a set of elements of a vector to a control total.

The C++ code for `rdscale()` follows on the next page.

The *rdscale()* Function

```
/* rdscale() -- right direction scaling ****************************
  v is a Vector of floats such that v[1] is the first element.
  n is the number of elements in the group
  group is a pointer to n integers, the sector numbers entering
    into the scaling
  control is what they are being scaled to equal.
  This routine does no range checking of the sector numbers.
  */

double rdscale(Vector v, short n, short *group, float control)
{
  float psumma=0.0; // sum of positive elements
  float nsumma=0.0; // sum of negative elements
  float fac=0.0;                      // scaling factor
  float x=0.0;
  short i, j, allzeroes=TRUE;

  psumma = nsumma = 0.;
  for(i = 0; i < n; i++){
    x = v1[group[i]];
    if(x > 0) psumma += x;
    else nsumma += x;
    if(fabs(x)>=.0000001) allzeroes=FALSE;
    }
  if(allzeroes && fabs(control)>.000001)
    return(0);  // do nothing if we have no data to scale.
  // Quadratic formula
  if(psumma > 0.){
    fac = control; // To get into double
    fac = fac*fac - psumma*nsumma*4.; // discriminant
    fac = sqrt(fac);
    fac = (fac + control)/(psumma*2);
    }
  else if(fabs(control) >= .000001)
    fac = nsumma/control;
  else fac = 0.;

  for(i = 0; i < n; i++){
    j = group[i];
    if(v1[j ] > 0.) v1[j] *= fac;
    else {
      // Next line is necessary when both control and sum are 0.
      if(fabs(fac)<=.000001) v1[j]=0.0; // prevent divide by 0.
      else v1[j] /= fac;
      }
    arith(" in rdscale.",i+1);
    }
  return(fac);
  }
```