6.3 フィルター　移動平均　　ver.0.1

ここでは株価のテクニカル分析などで使われる様々な移動平均について取り上げて、その特色や限界を再考してみましょう。また、6.1 で取り上げたフィルターとの関連を調べてみよう。算術移動平均と幾何移動平均についてはすでに３節でプログラムをしたので省略する。その際には、例えば 1,2,3,4,5 の期間について 1,2,3,4,5 のデータがあるとき、その５期移動平均を計算するには、その真ん中の第３期を中心に移動平均をとるものであった。それを算術平均でする方法と、データが非負であるならば幾何平均で計算する方法が存在する。それを動かしていき、上の場合、前後２期ずつを欠損値にすれば移動平均は完成する。前後に原系列を入れて補正する方法もある。一般に、幾何平均で計算するほうが散らばりが小さいことが知られている。ただし、上の方法では現在が第５期であるときは第７期に経過するまでリアルタイムで計算できないわけで、株式のテクニカル分析などでは、便宜上、現在から遡って移動平均をとって、その現在の値にする。しかしながら、これは、実質的には第３期の移動平均値を第５期にずらしたものにすぎないので、単純移動平均を用いた分析ではその分だけ遅れた分析をしていることになる。ジュニア版では、現在にずらした株価テクニカル分析の手法を扱っている。したがって、そこでは５期移動平均の場合、特に処理を行なわなければ、最初の４期が欠損値になる。

単純移動平均（算術、幾何）と最後に取り上げる移動メディアンに言える長所と短所を述べよう。長所は計算が極めて簡単で、データのトレンドを読み解くには簡便であることである。短所は、逆に人工的な周期を発生させてしまうことになり、見せかけの関係が生まれる原因になりやすい。また、中央、非中央どちらの方法をとろうとも、項数ｋとした時（ｋ－１）／２個だけの欠損値を両端に生み出す。このことは、特に単純移動平均では将来を予測をすることは技術的に困難であることを意味する。移動平均系はすべて異常値に敏感に反応して、それを滑らかにするようにデータを平均する。一方、移動メディアンはそもそも異常値をカットしてしまうのでその影響は全く受けない。単純移動平均や単純メディアンにはその背景にモデルは存在しない。また、スペクトラルによる遮断特性の観点から単純移動平均は、非常に大雑把でその帯域が制限されたデータを生み出すことが知られている。

単純移動平均の以上による短所を考慮に入れて、古くからいくつかの試みがなされている。１つには、２つのずれた、または、異なる項数の単純移動平均を用いるやり方。もう１つには、対称加重移動平均にすることによって、背景にサイクルや３次モデルを考えるもの。そして、それを改良して、データが欠落する端点にも非対称なウエイトを与えてすべてのデータをフィルタリングするもの。もう１つには、漸化式を用いて、そのウエイトが指数的に減少していくようなフィルターを考えるもの。そして、そこにトレンドを計算する漸化式をもう１つ導入するもの。これらの試みにより、たとえ遮断特性が劣っていても、２つ組み合わせて解決する流れ、そして、端点の欠落値をなくすために（このことは遅行

性を解消することを意味する）非対称ウエイトまたは指数平均を考える流れ、そして、平均の概念からそもそも離れてメディアンに行く流れがある。

現実には、マクロの世界では、モデルやサイクルを背景に考えた非対称ウエイトによる加重移動平均の方向に進んでいる。また、それらを一般化し整理する方向に進んでいる。それらの先には、状態空間表現を用いた世界がある。

中央化移動平均（四半期のケース）

再掲になるが、四半期データの場合に、原系列から季節成分と不規則変動部分の２つを取り除いて、トレンド＋サイクルを求めるのに使われる。４期の移動平均と１つずらした４期の移動平均の値の２つをさらに平均することによって、その真ん中の期の値にするものである。

```
proc cma(x);
    local n,k,m,i,m1,m2;
    n=rows(x); k=cols(x);
    m=zeros(n-4,k);
    i=1;
    do while i<=n-4;
        m1=(x[i,.]+x[i+1,.]+x[i+2,.]+x[i+3,.])/4;
        m2=(x[i+1,.]+x[i+2,.]+x[i+3,.]+x[i+4,.])/4;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    retp(m);
endp;
```

中央化移動平均（12 項のケース）

同様に、12 期のケースにも２つをずらして平均することによってできる。下では、x[i,.]+x[i+1,.]+ … +x[i+11,.]を計算する代わりに sumc(x[i:i+11])を、x[i+1,.]+x[i+2,.]+ … +x[i+12,.] を計算する代わりに sumc(x[i+1:i+12])を用いるものとする。

```
proc cma12(x);
    local n,k,m,i,m1,m2;
    n=rows(x); k=cols(x);
    m=zeros(n-12,k);
    i=1;
    do while i<=n-12;
        m1=sumc(x[i:i+11])/12;
```

```
        m2=sumc(x[i+1:i+12])/12;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    retp(m);
endp;
```

ただし、これらは不規則変動をもカットしてしまう。季節成分だけをカットするには、さらに 6.1 でやったような季節それぞれの成分を計算するような操作を加える必要がある。なお、上の２つのプログラムはすぐ使えるようにデータの端の両側に生じる欠落値自体を考えずに、データを最終的にその分短くしている。上では算術平均で計算しているが幾何平均にしても同様な計算に簡単に変更できる。

Bollinger Band（非中央）

単純移動平均のまわりのσを求めて、その±２σの幅を求めることを逐次的に繰り返すものである。

```
proc bollinger(x,k);
    local m,i,j,sig;
/* MA(k) calculation */
    m=zeros(rows(x),cols(x));
    i=k;
    do while i<=rows(x);
        j=1;
        do while j<=k;
            m[i,.]=m[i,.]+x[i-k+j,.];
            j=j+1;
        endo;
        i=i+1;
    endo;
    m=m/k;
    m[1:k-1,.]=miss(m[1:k-1,.],0);
/* sigma calculation */
    sig=zeros(rows(x),cols(x));
    i=k;
    do while i<=rows(x);
        sig[i]=sd(x[i-k+1:i,.]);
        i=i+1;
```

```
    endo;
    sig;
    retp(m~(m+2*sig)~(m-2*sig));
endp;

proc sd(x);
    local n,sum,i,xbar,s2,s;
    n=rows(x);
    sum=0;
    i=1;
    do while i<=n;
        sum=sum+x[i,.];
        i=i+1;
    endo;
    xbar=sum/n;
    s2=sumc((x-xbar)^2)/n;     @ Divided by n. @
    s=sqrt(s2);
    retp(s);
endp;
```

後半のｎで割るバージョンの標準偏差計算の sd を作成する代わりに、前半の procedure の中で sd を呼び出しているところを組込み関数 stdc に変更して、n-1 をかけてｎで割れば同じことができます。上では、(m+2*sig)と(m-2*sig)の２つをリターンとして返しているが、さらにこれを１倍したものや３倍したものを～の印で水平方向にマージすると、２重３重のバンドが描ける。ここでは２倍を使っているが、95%や 90%の有意水準の値を計算してそれを倍数に用いることも考えられるであろう。

加重移動平均（非中央）

直近の値ほどウエイトを置くものである。例えば、５期の加重移動平均をする場合、当期が５、前期が４、その前が３、その次が２、最後に１のウエイトをそれぞれ与えて、合計が１になるように調整したものを原系列にかけ合わせて、それを逐次的に動かして求めるものである。以下では、項数ｋを設定すれば自動的にk,k-1,k-2,k-3,k-4,…のウエイトがその項数幅だけ形成される。

```
proc wma(x,k);
    local w,m,i;
    w=seqa(1,1,k)/sumc(seqa(1,1,k));
    m=zeros(rows(x),cols(x));
```

```
    i=k;
    do while i<=rows(x);
        m[i,.]=w'x[i-k+1:i,.];
        i=i+1;
    endo;
    m[1:k-1,.]=miss(m[1:k-1,.],0);
    retp(m);
endp;
```

一般加重移動平均（非中央）

ウエイトの数値を列ベクトルwで与えるものが加重平均の一般形である。ウエイトは、例えば、w={0.1,0.2,0.3,0.4};などとして与えられるが、その和は必ずしも１である必要はない。また、ウエイト列の要素の数が加重移動平均の期間となる。

```
proc gwma(x,w);
    local k,m,i;
    k=rows(w);
    m=zeros(rows(x),cols(x));
    i=k;
    do while i<=rows(x);
        m[i,.]=w'x[i-k+1:i,.];
        i=i+1;
    endo;
    m[1:k-1,.]=miss(m[1:k-1,.],0);
    retp(m);
endp;
```

なお、1 つ前の加重平均は、この関数において、{1/15,2/15,3/15,4/15,5/15}を計算して小数にしたものを列でwとして与えたものと同一である。すなわち、

```
    w=(1/15)|(2/15)|(3/15)|(4/15)|(5/15);
```

をウエイトとして与えたものに等しい。ただし、GAUSS では｛　｝の中に分数を書くことはできないことに注意のこと。

対称加重移動平均

同じくwのウエイト列ベクトルを与えるのであるが、その最初の要素を中心にして

$$w=\{w_1,w_2,w_3,\dots\} \quad \rightarrow \quad b=\{\dots,w_3,w_2,w_1,w_2,w_3,\dots\};$$

という具合にウエイトを左右対称にした加重平均である。プログラムは一見複雑ではあるが、単に上のwからｂへの変換を行なったものを一般加重移動平均と同じように逐次的にかけ合せただけのものである。

```
proc symwma(x,w);
    local t,k,b,i,m;
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
    m[1:k,.]=miss(m[1:k,.],0);
    m[t-k+1:t,.]=miss(m[t-k+1:t,.],0);
    retp(m);
endp;
```

このフィルターは、非中央ではない。ｋ＋１番目から始まって、ｔ－ｋに終わる。移動平均の値は中央に格納される。このウエイトベクトルに下限と上限を設定した Band Pass の定数を計算して代入したものが、6.1 の Baxter-King フィルターそのものになる。

Spencer15 期移動平均

対称加重移動平均のウエイトを３次多項式を満たすものにしたものが、Spencerの15項移動平均である。以下のプログラムは、対称加重移動平均のプログラムのウエイト列を内部で定数で与えたプログラムにすぎない。一般に、$m_t = c_0 + c_1 t + \ldots + c_k t^k$について

$$m_t = \sum_j a_j m_{t-j}$$

となる必要十分条件は

$$\sum_j a_j = 1$$

$$\sum_j j^m a_j = 0 \qquad for \quad m = 1,2,..,k$$

となることが知られている。実際に、$\{a_{-7},a_{-6},a_{-5},a_{-4},a_{-3},a_{-2},a_{-1},a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7\}$＝１／320[-3,-6,-5,3,21,46,67,74,67,46,21,3,-5,-6,-3]の下のケースでは、そのウエイトの合計は１で、ｋを３とした時の$(-7)^3(-3)+(-6)^3(-6)+(-5)^3(-5)+(-4)^3(3)+(-3)^3(21)+(-2)^3(46)+(-1)^3(67)+0^3(74)+1^3(67)+2^3(46)+3^3(21)+4^3(3)+5^3(-5)+6^3(-6)+7^3(-3)=0$となり、320で割っても０となる。これを２乗のケースと１乗のケースにあてはめても同様に０となるので、２つの必要十分条件は満たされていることがわかる。

```
proc spencer15(x);
    local w,t,k,b,i,m;
    w={74,67,46,21,3,-5,-6,-3}; w=w/320;
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
    m[1:k,.]=miss(m[1:k,.],0);
    m[t-k+1:t,.]=miss(m[t-k+1:t,.],0);
    retp(m);
endp;
```

上の冒頭のウエイトベクトルをw={3,4,-1}; w=w/9;に変更すれば３次式のSpencer５が簡単に得られる。その場合も、1／９[-1,4,3,4,-1]の合計は１であり、またk=3のとき、$(-2)^3(-1)+(-1)^3(4)+0^3(3)+1^3(4)+2^3(-1)=0$であり９で割っても０となる。また、k=２のとき、$(-2)^2(-1)+(-1)^2(4)+0^2(3)+1^2(4)+2^2(-1)=0$ またk=1のとき(-2)(-1)＋(-1)(4)＋0(3)＋1(4)＋

2(-1)＝０であるから同様に９で割ってもすべて０になり、３次式を通す必要十分条件は満たされている。おもしろいことに、このSpencerの高次式のｋ＝１にしたとき、すなわち、$m_t=c_0+c_1 t$であるときは、線形そのものであることがわかる。上で述べた必要十分条件も、単純移動平均の場合、例えば、５項の場合、{1/5,1/5,1/5,1/5,1/5}という対称ウエイトになり、その和は明らかに１であり、$(-2)^1(1/5)+(-1)^1(1/5)+0^1(1/5)+1^1(1/5)+2^1(1/5)$の計算は対称式なので０であることがわかる。これは５項に限らずすべての奇数項数の単純移動平均にあてはまることは容易に推測できる。したがって単純移動平均は、実はSpencer型の対称加重移動平均のｋ＝１の場合の特殊例であって、それに包含されることがわかる。

Henderson13 **期移動平均**

上では、リターンの前の２行で最初のｋ期と最後のｋ期を除外している。そうではなくて、データが対称加重平均では足りない部分もウエイトを与えて計算させるような、補正された対称加重移動平均がこの Henderson13 である。実際には、その係数は３次式を満たすような、移動平均の３階差分の平方和を最小化するように定められている。すなわち、

$$\min \sum_{i=-k}^{k} (\Delta^3 a(i))^2$$

$$where \Delta^3 a(i) = a(i) - 3a(i-1) + 3a(i-2) - a(i-3)$$

$$a(i) = c(k)[(k+1)^2 - i^2][(k+2)^2 - i^2][(k+3)^3 - i^2][3(k+2)^2 - 16 - 11i^2]$$

というc(k)というｋに依存する定数に対してウエイトa(i)の３階の階差の平方和を最小にするようなウエイトを用いる。この場合も Spencer の移動平均の３次の場合（k=3 のケース）の必要十分条件を満たす。以下では、もうすでに計算されているウエイトを用いる。

```
proc henderson13(x);
    local w,t,k,b,i,m,w1,w2,w3,w4,w5,w6;
/* Symmetric part in the middle */
    w={0.24006,0.21434,0.14736,0.06549,0,-0.02786,-0.01935};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
```

```
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
w1={-0.09186,-0.05811,0.01202,0.11977,0.24390,0.35315,0.42113};
w2={-0.04271,-0.03863,0.00182,0.07990,0.17436,0.25392,0.29223,0.27910};
w3={-0.01603,-0.02487,0.00267,0.06784,0.14939,0.21605,0.24144,0.21540,0.14810};
w4={-0.00813,-0.02019,0.00413,0.06608,0.14441,0.20784,0.23002,0.20076,0.13024,0.04483};
w5={-0.01099,-0.02204,0.00330,0.06626,0.14559,0.21004,0.23324,0.20498,0.13547,0.05108,-0.01694};
w6={-0.01643,-0.02577,0.00127,0.06594,0.14698,0.21314,0.23803,0.21149,0.14368,0.06099,-0.00532,-0.03401};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    m[4,.]=rev(w4)'x[1:k+4,.]; m[t-3,.]=w4'x[t-k-3:t];
    m[5,.]=rev(w5)'x[1:k+5,.]; m[t-4,.]=w5'x[t-k-4:t];
    m[6,.]=rev(w6)'x[1:k+6,.]; m[t-5,.]=w6'x[t-k-5:t];
    retp(m);
endp;
```

Henderson **移動平均**(5,7,9,15,17,23)

全く同様にして、その他の項数の Henderson 移動平均の計算ができる。

```
proc henderson5(x);
    local w,t,k,b,i,m,w1,w2;
/* Symmetric part in the middle */
    w={0.558,0.294,-0.073};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
```

```
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.073,0.403,0.670};
    w2={-0.073,0.294,0.522,0.257};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    retp(m);
endp;

proc henderson7(x);
    local w,t,k,b,i,m,w1,w2,w3;
/* Symmetric part in the middle */
    w={0.412,0.294,0.059,-0.059};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.034,0.116,0.383,0.535};
    w2={-0.054,0.061,0.294,0.41,0.289};
```

```
    w3={-0.053,0.058,0.287,0.399,0.275,0.034};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    retp(m);
endp;

proc henderson9(x);
    local w,t,k,b,i,m,w1,w2,w3,w4;
/* Symmetric part in the middle */
    w={0.33,0.267,0.119,-0.010,-0.041};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.156,-0.034,0.185,0.424,0.581};
    w2={-0.049,-0.011,0.126,0.282,0.354,0.298};
    w3={-0.022,0,0.12,0.259,0.315,0.242,0.086};
    w4={-0.031,-0.004,0.12,0.263,0.324,0.255,0.102,-0.029};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    m[4,.]=rev(w4)'x[1:k+4,.]; m[t-3,.]=w4'x[t-k-3:t];
    retp(m);
```

```
endp;

proc henderson15(x);
    local w,t,k,b,i,m,w1,w2,w3,w4,w5,w6,w7;
/* Symmetric part in the middle */
    w={0.212,0.194,0.146,0.083,0.024,-0.014,-0.024,-0.014};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.079,-0.057,-0.014,0.057,0.149,0.244,0.325,0.375};
    w2={-0.04,-0.039,-0.016,0.034,0.105,0.18,0.24,0.27,0.265};
    w3={-0.016,-0.025,-0.013,0.027,0.088,0.152,0.202,0.221,0.205,0.159};
    w4={-0.005,-0.018,-0.01,0.026,0.083,0.143,0.189,0.205,0.185,0.135,0.069};
    w5={-0.005,-0.018,-0.01,0.026,0.082,0.143,0.188,0.203,0.183,0.133,0.067,0.006};
    w6={-0.008,-0.02,-0.011,0.025,0.083,0.144,0.191,0.207,0.188,0.139,0.074,0.014,-
0.026};
    w7={-0.012,-0.023,-0.013,0.025,0.083,0.146,0.193,0.210,0.192,0.144,0.080,0.021,-
0.017,-0.028};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    m[4,.]=rev(w4)'x[1:k+4,.]; m[t-3,.]=w4'x[t-k-3:t];
    m[5,.]=rev(w5)'x[1:k+5,.]; m[t-4,.]=w5'x[t-k-4:t];
```

```
    m[6,.]=rev(w6)'x[1:k+6,.]; m[t-5,.]=w6'x[t-k-5:t];
    m[7,.]=rev(w7)'x[1:k+7,.]; m[t-6,.]=w7'x[t-k-6:t];
    retp(m);
endp;

proc henderson17(x);
    local w,t,k,b,i,m,w1,w2,w3,w4,w5,w6,w7,w8;
/* Symmetric part in the middle */
    w={0.190,0.176,0.141,0.092,0.042,0.002,-0.019,-0.02,-0.009};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.081,-0.062,-0.032,0.018,0.087,0.166,0.244,0.309,0.351};
    w2={-0.042,-0.04,-0.026,0.007,0.059,0.121,0.182,0.23,0.255,0.254};
    w3={-0.017,-0.025,-0.02,0.004,0.047,0.101,0.152,0.191,0.206,0.197,0.164};
    w4={-0.005,-0.016,-0.015,0.005,0.043,0.092,0.14,0.174,0.186,0.172,0.136,0.086};
    w5={-0.001,-0.013,-0.014,0.005,0.043,0.091,0.138,0.171,0.181,0.167,0.129,0.078,
0.026};
    w6={-0.003,-0.015,-0.015,0.005,0.043,0.091,0.138,0.172,0.183,0.169,0.132,0.081,
0.029,-0.012};
    w7={-0.006,-0.017,-0.016,0.004,0.043,0.092,0.14,0.174,0.186,0.173,0.137,0.087,
0.036,-0.005,-0.027};
    w8={-0.009,-0.019,-0.018,0.003,0.042,0.092,0.141,0.176,0.188,0.175,0.14,0.091,
```

```
0.04,0.001,-0.021,-0.023};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    m[4,.]=rev(w4)'x[1:k+4,.]; m[t-3,.]=w4'x[t-k-3:t];
    m[5,.]=rev(w5)'x[1:k+5,.]; m[t-4,.]=w5'x[t-k-4:t];
    m[6,.]=rev(w6)'x[1:k+6,.]; m[t-5,.]=w6'x[t-k-5:t];
    m[7,.]=rev(w7)'x[1:k+7,.]; m[t-6,.]=w7'x[t-k-6:t];
    m[8,.]=rev(w8)'x[1:k+8,.]; m[t-7,.]=w8'x[t-k-7:t];
    retp(m);
endp;

proc henderson23(x);
    local w,t,k,b,i,m,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11;
/* Symmetric part in the middle */
    w={0.148,0.138,0.122,0.097,0.068,0.039,0.013,-0.005,-0.015,-0.016,-0.011,-0.004};
    t=rows(x);
    k=rows(w)-1;
    b=zeros(2*k+1,1);
    b[k+1]=w[1];
    i=1;
    do while i<=k;
        b[k+1-i]=w[i+1]; b[k+1+i]=w[i+1];
        i=i+1;
    endo;
    m=zeros(rows(x),cols(x));
    i=k+1;
    do while i<=t-k;
        m[i,.]=b'x[i-k:i+k,.];
        i=i+1;
    endo;
/* Asymmetric part at end points */
    w1={-0.077,-0.064,-0.049,-0.028,0.002,0.039,0.084,0.133,0.1819999999999999,0.227,
0.2629999999999999,0.2879999999999999};
    w2={-0.04599999999999999,-0.041,-0.035,-0.02399999999999999,-0.004,0.025,
0.06099999999999999,0.101,0.1409999999999999,0.1759999999999999,0.203,0.218999
```

```
9999999999,0.224};
    w3={-0.02199999999999999,-0.025,-0.025,-0.019,-0.005,0.01799999999999999,0.049,
0.08200000000000001,0.116,0.1459999999999999,0.166,0.177,0.1759999999999999,
0.166};
    w4={-0.008,-0.014,-0.01799999999999999,-0.015,0.004,0.015,0.042,
0.07299999999999999,0.103,0.129,0.1469999999999999,0.154,0.15,0.134,0.112};
    w5={-0.001,-0.008,-0.013,-0.01199999999999999,-0.003,0.015,
0.04099999999999999,0.06899999999999999,0.0981,0.1211,0.1370999999999999,
0.1421,0.1350999999999999,0.1191,0.09510000000000001,0.0661};
    w6={0.003,-0.006,-0.01099999999999999,-0.01099999999999999,-0.002,0.015,0.039,
0.067,0.095,0.1189999999999999,0.134,0.139,0.131,0.114,0.08799999999999998,
0.05899999999999999,0.027};
    w7={0.002,-0.006,-0.01199999999999999,-0.01099999999999999,-0.003,0.015,0.039,
0.068,0.09599999999999999,0.1179999999999999,0.134,0.138,0.132,0.114,
0.08899999999999998,0.05899999999999999,0.027,0.001};
    w8={0.001,-0.007,-0.013,-0.01099999999999999,-0.003,0.015,0.039,0.068,
0.09599999999999999,0.1199999999999999,0.135,0.14,0.133,0.116,0.09,0.06,0.031,
0.005,-0.015};
    w9={-0.002,-0.007,-0.013,-0.013,-0.003,0.014,0.039,0.068,0.09699999999999999,
0.1199999999999999,0.137,0.14,0.136,0.1179999999999999,0.09399999999999998,
0.064,0.034,0.008,-0.01,-0.021};
    w10={-0.003,-0.01,-0.015,-0.014,-0.005,0.014,0.04,0.069,0.09699999999999999,
0.1219999999999999,0.138,0.1429999999999999,0.137,0.1199999999999999,0.095,
0.067,0.037,0.01099999999999999,-0.007,-0.017,-0.019};
    w11={-0.004,-0.01099999999999999,-0.016,-0.015,-0.005,0.013,0.039,0.068,
0.09699999999999999,0.1219999999999999,0.138,0.1439999999999999,0.138,
0.1219999999999999,0.09699999999999999,0.068,0.039,0.013,-0.005,-0.015,-0.016,
-0.01099999999999999};
    m[1,.]=rev(w1)'x[1:k+1,.]; m[t,.]=w1'x[t-k:t];
    m[2,.]=rev(w2)'x[1:k+2,.]; m[t-1,.]=w2'x[t-k-1:t];
    m[3,.]=rev(w3)'x[1:k+3,.]; m[t-2,.]=w3'x[t-k-2:t];
    m[4,.]=rev(w4)'x[1:k+4,.]; m[t-3,.]=w4'x[t-k-3:t];
    m[5,.]=rev(w5)'x[1:k+5,.]; m[t-4,.]=w5'x[t-k-4:t];
    m[6,.]=rev(w6)'x[1:k+6,.]; m[t-5,.]=w6'x[t-k-5:t];
    m[7,.]=rev(w7)'x[1:k+7,.]; m[t-6,.]=w7'x[t-k-6:t];
    m[8,.]=rev(w8)'x[1:k+8,.]; m[t-7,.]=w8'x[t-k-7:t];
```

```
    m[9,.]=rev(w9)'x[1:k+9,.]; m[t-8,.]=w9'x[t-k-8:t];
    m[10,.]=rev(w10)'x[1:k+10,.]; m[t-9,.]=w10'x[t-k-9:t];
    m[11,.]=rev(w11)'x[1:k+11,.]; m[t-10,.]=w11'x[t-k-10:t];
    retp(m);
endp;
```

以上はすべて、ノイズとトレンドの割合を通常の 3.5 に固定してあらかじめ計算された係数の値をもとに移動加重平均させたものである。これらの考え方は、Census X11 や X12 のフィルターのトレンド推定のベースとなるものである。

平滑移動平均（非中央）

```
proc smoothma(x,k);
    local m,sum,i;
    m=zeros(rows(x),cols(x));
    sum=sumc(x[1:k,.]);
    m[k,.]=sum/k;
    i=k+1;
    do while i<=rows(x);
        m[i,.]=(sum-m[i-1,.]+x[i,.])/k;
        sum=sum-m[i-1,.]+x[i,.];
        i=i+1;
    endo;
    m[1:k-1,.]=miss(m[1:k-1,.],0);
    retp(m);
endp;
```

EMA **指数加重移動平均**

任意の０と１の間の値である平滑制約αに対して

$$m_1 = x_1$$

$$m_t = \alpha x_t + (1-\alpha) m_{t-1} \qquad , t = 2,3,4,\ldots,n$$

という漸化式をプログラムするものである。αが１に近ければ原系列に近くなり、反対に０に近くなれば伸ばされた形になる。

```
proc ewma(x,alpha);
    local m,i;
    m=zeros(rows(x),cols(x));
    m[1,.]=x[1,.];
    i=2;
```

```
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
    endo;
    retp(m);
endp;
```

これは、実際には、ｔ≧2 に対して

$$m_t = \sum_{j=0}^{t-2} \alpha(1-\alpha)^j x_{t-j} + (1-\alpha)^{t-1} x_1$$

となるが、これは現在から遠くへ行くほどにそのウエイトが指数的に減少していくことを示しているので、指数平滑または EMA と呼ばれている。現在から遠くなれば微々たるウエイトになるのであるが、すべてのデータのヒストリーを使うところにこのフィルターの特色がある。一般に遅行性はないとされるが、この背景にモデルがあるわけではない。

Holt's Method

EWMA のトレンド項を考えた特殊な形として Holt の方法がある。パラメータαとβに対して、通常の EMA を考える Lt 式とトレンド項を考える Tt 式の２つからなる計算を、

$L_1 = x_1$

$T_1 = x_2 - x_1$

を初期条件として、t＝2,3,4,…,n に対して

$L_t = \alpha x_t + (1-\alpha) L_{t-1}$

$T_t = \beta (L_t - L_{t-1}) + (1-\beta) T_{t-1}$

$m_t = L_{t-1} + T_{t-1}$

という漸化式を計算して t=2,3,4,…,n に対して、m_t を求めるものである。

```
proc holt(x,alpha,beta);
    local L,T,i,m;
    n=rows(x);
    L=zeros(rows(x),cols(x)); T=zeros(rows(x),cols(x)); m=zeros(rows(x),cols(x));
    L[1,.]=x[1,.];
    T[1,.]=x[2,.]-x[1,.];
    i=2;
    do while i<=rows(x);
        L[i,.]=alpha*x[i,.]+(1-alpha)*L[i-1,.];
        T[i,.]=beta*(L[i,.]-L[i-1,.])+(1-beta)*T[i-1,.];
        m[i,.]=L[i-1,.]+T[i-1,.];
        i=i+1;
```

```
    endo;
    m[1,.]=miss(m[1,.],0);
    retp(m);
endp;
```

Double EMA, Triple EMA

　α を固定して EWMA をさらにもう一回したダブル、さらにもう一回したトリプルも簡単にプログラムできる。

```
proc ewma2(x,alpha);
    local m,i;
    m=zeros(rows(x),cols(x));
    m[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
    endo;
/* 2nd ewma */
    x=m;
    m[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
    endo;
    retp(m);
endp;

proc ewma3(x,alpha);
    local m,i;
    m=zeros(rows(x),cols(x));
    m[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
```

```
    endo;
/* 2nd ewma */
    x=m;
    m[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
    endo;
/* 3rd ewma */
    x=m;
    m[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m[i,.]=alpha*x[i,.]+(1-alpha)*m[i-1,.];
        i=i+1;
    endo;
    retp(m);
endp;
```

上の計算は、非効率ではあるが、EMA の計算をデータ x を計算された EMA の m にスイッチして X=m とすることで、EMA の計算を２回、さらに３回行なっただけのものである。なお、EMA 系の係数αには、期間をｎとした上でα＝２／（ｎ＋１）とすることも多い。EMA 系のフィルターにはこれ以外にも数多くのバリエーションが存在する。

MACD

上のα＝２／（ｎ＋１）の関係式を用いて、与えられた短期と長期の EMA の値の差をとって MACD とし、これをその移動平均とともに示してやりゼロ基準線との関係を探るのがテクニカル分析における MACD の考え方である。

```
proc macd(x);
    local short,long,ma,m1,m2,alpha,i,macdi,signali,j;
/* Parameters to be changed */
    short=12;
    long =26;
    ma     = 9;
/* Short EWMA1 */
    m1=zeros(rows(x),cols(x));
```

```
    alpha=2/(short+1);
    m1[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m1[i,.]=alpha*x[i,.]+(1-alpha)*m1[i-1,.];
        i=i+1;
    endo;
/* Long EWMA2 */
    m2=zeros(rows(x),cols(x));
    alpha=2/(long+1);
    m2[1,.]=x[1,.];
    i=2;
    do while i<=rows(x);
        m2[i,.]=alpha*x[i,.]+(1-alpha)*m2[i-1,.];
        i=i+1;
    endo;
/* MACD and its Moving Average */
    macdi=m1-m2;
    signali=zeros(rows(x),cols(x));
    i=ma;
    do while i<=rows(x);
        j=1;
        do while j<=ma;
            signali[i,.]=signali[i,.]+macdi[i-ma+j,.];
            j=j+1;
        endo;
        i=i+1;
    endo;
    signali=signali/ma;
    signali[1:ma-1,.]=miss(signali[1:ma-1,.],0);
    retp(zeros(rows(x),1)~macdi~signali);
endp;
```

上では、短期 EMA１を 12 期のものとし、長期 EMA2 を 26 期のものとした上で

MACD＝EMA１－EMA2
Signal=MACD の移動平均

の２つを計算したものである。移動平均には非中央の９期移動平均を用いている。これらの値（Short,Long,MA の３つ）は適宜変更してもらいたい。結果は、原系列を入れると３列（ゼロ基準線、MACD、Signal）が出てくるので、例えば、y が原系列だとすると

```
new; cls;
（　y の呼び出し部分　）
library pgraph;
graphset;
xy(seqa(1,1,rows(y)),macd(y));
```

として、以下にmacdのprocedureを置いてやれば計算される。なお、上のMACDの計算にはEWMA１－EWMA2をさらに現在の値で割った割合が用いられることもある。

移動平均乖離率

同じような考え方で、もっと単純に移動平均を用いる方法もある。非中央の移動平均に戻って、短期と長期の移動平均の商をとって、それを１から引いたものを 100 倍したものを移動平均の乖離率とする。

```
proc madiff(x);
    local short,long,m1,m2,i,j,diff;
/* Parameters to be changed */
    short=5;
    long =25;
/* Short MA */
    m1=zeros(rows(x),cols(x));
    i=short;
    do while i<=rows(x);
        j=1;
        do while j<=short;
            m1[i,.]=m1[i,.]+x[i-short+j,.];
            j=j+1;
        endo;
        i=i+1;
    endo;
    m1=m1/short;
    m1[1:short-1,.]=miss(m1[1:short-1,.],0);
/* Long MA */
    m2=zeros(rows(x),cols(x));
```

```
    i=long;
    do while i<=rows(x);
        j=1;
        do while j<=long;
            m2[i,.]=m2[i,.]+x[i-long+j,.];
            j=j+1;
        endo;
        i=i+1;
    endo;
    m2=m2/long;
    m2[1:long-1,.]=miss(m1[1:long-1,.],0);
/* Ratio of MAs */
    diff=(m1./m2-1)*100;
    diff[1:long-1,.]=miss(diff[1:long-1,.],0);
    retp(zeros(rows(x),1)~diff);
endp;
```

上では短期に５期移動平均 MA1、長期に 25 期移動平均 MA2 をとって、それらの乖離率を

移動平均乖離率＝（MA1／MA2－１）×100

として計算をし、ゼロ基準線とともにアウトプットとして出している。期間は適宜変更してもらいたい。上の計算では通常株価グラフでプロットされるような５日と 25 日の２つの移動平均線の背景にあるものを説明することができる。

単純移動メディアン（非中央）

アウトライヤーを除去するのには、平均ではなくてメディアンを取っていく方法もある。一番単純なものは、一回だけｋ期間内のメディアンを取ることを逐次的に行なう。

```
proc mmedian(x,k);
    local m,i;
    m=zeros(rows(x),cols(x));
    i=k;
    do while i<=rows(x);
        m[i,.]=median(x[i-k+1:i,.]);
        i=i+1;
    endo;
    m[1:k-1,.]=miss(m[1:k-1,.],0);
```

```
    retp(m);
endp;
```

繰り返し移動メディアン

上の単純メディアンを中央が中心になるように変更して、欠損値になる両側(k-1)/2 個に原系列を補充した上で、移動メディアンを繰り返し行なって動かなくなるまで行ったものが次のものである。ただし、項数は奇数である必要がある。

```
proc repmmedian(x,k);
    local m,i,j;
    if k/2==round(k/2);
        errorlog "ERROR: Parameter k must be odd number.";
        retp;
    endif;
    m=zeros(rows(x),cols(x));
    n=rows(x);
    j=1;
    do until x==m;
        if j>=2;
            x=m;
        endif;
        i=k;
        do while i<=n;
            m[i-(k-1)/2,.]=median(x[i-k+1:i,.]);
            i=i+1;
        endo;
        m[1:(k-1)/2,.]=x[1:(k-1)/2,.];
        m[n-(k-1)/2+1:n,.]=x[n-(k-1)/2+1:n,.];
        print/rz "iteration" j;
        j=j+1;
    endo;
    retp(m);
endp;
```

移動メディアンは、メディアンの概念そのものがアウトライヤーを除外するので、異常値や極端な上昇や下降の影響を受けない。また、構造変化を表すのに適したフィルターであることも知られている。ただし、期間ｋの選択には３または５が用いられる。それ以上に

なると特に処理を行なわない限り、平坦な部分が多くなる。ｋが３または５の場合でも、既存の値の中からメディアンが選択されるので、その軌跡は見た目上、平滑されたようには見えないことが多い。

以上、「移動平均」という大きな枠組みで捉えれば、背景に高次式があろうと線形であろうと、Baxter-King のフィルターであろうとすべてが１つであることがわかる。さらに、その計算空間を複素領域（実現領域は実数である）までに拡大すれば、Hodrick-Prescott フィルターは

$$y_t^T = \frac{\theta_1\theta_2}{\lambda}[\sum_{j=0}^{\infty}(A_1\theta_1^{\,j} + A_2\theta_2^{\,j})y_{t-j} + \sum_{j=0}^{\infty}(A_1\theta_1^{\,j} + A_2\theta_2^{\,j})y_{t+j}$$

というような、λと A_1 と A_2 という定数に対して、θ_1 と θ_2 の２つの複素共役数の関数で表せることがわかっている。これも無限期間を考慮に入れた「移動平均」と言える。その動きを有限のデータ区間に近似させて考えて発展させたものが、Baxter-King のフィルターであると言える。この Baxter-King も６期および 32 期に高バンドと低バンドを固定した上でｋ＝８にすれば、$\{ w_{-8}, w_{-7}, w_{-6}, w_{-5}, w_{-4}, w_{-3}, w_{-2}, w_{-1}, w_0, w_1, w_2, w_3, s_4, w_5, w_6, w_7, w_8\}$=
{-0.014053754, -0.013941531, -0.057736405, -0.11678896, -0.13390896, -0.067670906, 0.068203190, 0.20484220, 0.26211026, 0.20484220, 0.068203190, -0.067670906, -0.13390896, -0.11678896, -0.057736405, -0.013941531, -0.014053754 }
という対称ウエイト表現ができることがわかっている。なお、この場合のウエイトの総和は１ではなくて０である。その他のバンドの範囲やｋの選択によってこれらの数値は変化するが、依然としてそれぞれに固定した対称ウエイトである。これらすべては、「移動平均」という大きな枠組みでつながっているのである。