6
REFERÊNCIAS BIBLIOGRÁFICAS

CHATFIELD, Christopher. **The Analysis of Time Series: an introduction**. 6ª edição. Flórida: Editora Chapman & Hall/CRC, 2004.

COSTA, Antonio Fernando B. / EPPRECHT, Eugenio K. / CARPINETTI, Luiz Cesar R. **Controle Estatístico de Qualidade**. 2ª edição. São Paulo: Editora Atlas S.A., 2005.

OLIVEIRA JUNIOR, Paulo M. **Impacto da Imprecisão da Previsão de Demanda na Cadeia Logística: Um Estudo de Caso na Indústria de Bebidas**. Dissertação de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

CHOPRA, Sunil / MEINDL, Peter. **Gerenciamento da Cadeia de Suprimentos – Estratégia, Planejamento e Operação**. 2ª edição. São Paulo: Editora Pearson, 2004.

LAW, Averill M. **Simulation Modeling and Analysis**. 4ª edição. Nova Iorque: Editora McGraw-Hill, 2007.

MONTGOMERY, Douglas C. **Introdução ao Controle Estatístico da Qualidade**. 4ª edição. Rio de Janeiro: Editora LTC editora, 2004.

SILVER, Eduard A. / PYKE, David F. / PETERSON, Rein. **Inventory Management and Production Planning and Scheduling**. 3ª edição. Nova Iorque: Editora John Wiley & Sons, 1998.

WINTERS, Peter R. Forecasting Sales by Exponentially Weighted Moving Averages. **Management Science**. Volume 6, p. 324-342, 1960.

APÊNDICE 1 – Fórmulas de Regressão Linear para Dados Históricos

A regressão linear, nas séries históricas, não foi feita automaticamente por *software* estatístico e sim, programada e calculada pelas fórmulas aqui apresentadas:

- **Modelo Constante:**

  O cálculo da previsão da demanda inicial é:

$$\hat{X}_0 = \frac{\sum_{t=-n+1}^{0} X_t}{n} \tag{A1.1}$$

Onde X_t é a demanda do período histórico t e n é o número de períodos históricos.

O erro quadrático médio inicial é:

$$EQM_0 = \frac{1}{n-1}\sum_{t=-n+1}^{0} (\varepsilon_t)^2 \tag{A1.2}$$

Onde ε_t é:

$$\varepsilon_t = \hat{X}_0 - X_t \tag{A1.3}$$

Onde X_t é a demanda do período histórico t, n é o número de períodos históricos e $\hat{X}_0$ é a previsão da demanda inicial. Sendo n-1 graus de liberdade no denominador, pois estimamos apenas uma variável ($\hat{X}_0$).

- **Modelo Crescente:**

  No modelo crescente, temos que calcular, também, os valores iniciais do nível da demanda ($\hat{a}_0$) e da taxa de crescimento ($\hat{b}_0$), dados por:

$$\hat{a}_0 = \bar{X} + \hat{b}_0\bar{t} \qquad \textbf{(A1.4)}$$

$$\hat{b}_0 = \frac{\sum_{t=-n+1}^{0}((t-\bar{t})X_t)}{\sum_{t=-n+1}^{0}(t-\bar{t})^2} \qquad \textbf{(A1.5)}$$

onde:

$\bar{t}$ é a média dos períodos históricos:

$$\bar{t} = \frac{1}{2}(n+1) \qquad \textbf{(A1.6)}$$

$\bar{X}$ é a média da demanda para os períodos históricos:

$$\bar{X} = \frac{1}{n}\sum_{t=-n+1}^{0} X_t \qquad \textbf{(A1.7)}$$

Já o cálculo da previsão da demanda inicial é dado por:

$$\hat{X}_1 = \hat{a}_0 + \hat{b}_0 \qquad \textbf{(A1.8)}$$

Por último, o erro quadrático médio inicial é:

$$EQM_0 = \frac{1}{n-2}\sum_{t=-n+1}^{0}(\varepsilon_t)^2 \qquad \textbf{(A1.9)}$$

onde **n** é o número de períodos históricos. São n-2 graus de liberdade no denominador, pois estimamos duas variáveis ($\hat{a}_0$ **e** $\hat{b}_0$) e ε_t é dado por:

$$\varepsilon_t = \hat{X}_t - X_t \quad , t = \{-n+1, -n+2, \dots, -1, 0\} \qquad \textbf{(A1.10)}$$

sendo:

$$\hat{X}_t = \bar{X} + \hat{b}_0(t - \bar{t}) \quad , t = \{-n+1, -n+2, \ldots , -1, 0\} \qquad \textbf{(A1.11)}$$

APÊNDICE 2 – Programa em C utilizado para obtenção dos valores de NMA nos métodos analisados

- **Parte comum aos dois modelos: Biblioteca da programação em C e geração de números aleatórios ~N(0,1):**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>  // Para usar relógio interno na geração de aleatório do C
#include <math.h>
#define norm   2.328306549295728e-10  /* 1.0/(m1+1) */
#define norm2  2.328318825240738e-10  /* 1.0/(m2+1) */
#define m1     4294967087.0
#define m2     4294944443.0
#define TAM 400  // Tamanho do comprimento da amostra; usamos 400 e 800
#define PER 36     // Tamanho da série histórica
#define ROD 10000  // Número de rodadas da simulação

static double drng[][6] =         // Matriz para geração de aleatórios do algoritmo do
LAW
{ Matriz de números muito extensa. Para obter a matriz, consultar Law (2007,
pg. 419).
    };

    /* Habilita matriz de sementes para a variável "stream". */
    void mrandgt(double* seed, int stream)
    {
    int i;
       for (i = 0; i <= 5; ++i) seed[i] = drng[stream][i];
    }
```

/* Gera o próximo número aleatório pelo algoritmo de LAW */

```
double mrand(int stream)
{
   long k;
   double p,
         s10 = rand(), s11 = drng[stream][1], s12 = drng[stream][2],
         s20 = drng[stream][3], s21 = drng[stream][4], s22 = drng[stream][5];

   p = 1403580.0 * s11 - 810728.0 * s10;
   k = p / m1;  p -= k*m1;  if (p < 0.0) p += m1;
   s10 = s11;   s11 = s12;  s12 = p;

   p = 527612.0 * s22 - 1370589.0 * s20;
   k = p / m2;  p -= k*m2;  if (p < 0.0) p += m2;
   s20 = s21;   s21 = s22;  s22 = p;

   drng[stream][0] = s10;  drng[stream][1] = s11;  drng[stream][2] = s12;
   drng[stream][3] = s20;  drng[stream][4] = s21;  drng[stream][5] = s22;

   if (s12 <= s22) return ((s12 - s22 + m1) * norm);
   else return ((s12 - s22) * norm);
}
```

/* Habilita o vetor de sementes para a variável "stream". */

```
void mrandst(double* seed, int stream)
{
int i;
   for (i = 0; i <= 5; ++i) drng[stream][i] = seed[i];
}
```

- **Modelo Constante:**

```
int main(void) {
   srand(time(NULL)); // Semente (relógio do sistema) para gerador de aleatórios
   float  erro, demand, ts, eqm, at, vies, dam, u1, u2;
   float  erro_ant, erro_prev, epad, eqm_ant, ewma, ewma_ant, lim, xhat, xhat_ant, alfa, beta, omega, lambda;
   float a, a_ant, a_xh, b, b_ant, b_xh;
   int t;
   int graf_ts, graf_cusum, graf_ewma;
   float smais = 0., smenos = 0., d, ke, kc;
   int i; float dd[PER+1],resid[PER+1], dbarra, tbarra, tbarra2; //para gerar serie histórica
   int kont = 1, kont_ts = 0, kont_ewma = 0, kont_cusum = 0;

   printf("VALORES DAS VARIÁVEIS DE CONTROLE: \n");
   printf("DEMANDA (a + b.t + erro):\n");
   printf("Trata-se do MODELO CONSTANTE !\n");
   printf("Digite o valor de b : "); scanf("%f",&b);
   printf("Digite o valor de a : "); scanf("%f",&a);
   printf("Digite o valor de alfa-> "); scanf("%f",&alfa);
   printf("Digite o valor de omega(w)-> "); scanf("%f",&omega);
   printf("Digite o valor de K para o EWMA-> "); scanf("%f",&ke);
   printf("Digite o valor de lambda (parâmetro do EWMA)-> "); scanf("%f",&lambda);
   printf("Digite o valor de K para o CUSUM-> "); scanf("%f",&kc);
   printf("Digite o valor de d (parâmetro do CUSUM)-> "); scanf("%f",&d);

   while (kont <= ROD)  {
   graf_ts = 1, graf_cusum = 1, graf_ewma = 1; smais = 0., smenos = 0.;
```

```
at = vies = 0.0;
ewma_ant = 0.0;
```

// Cálculos para inicializar a previsão da demanda com 36 dados da série histórica

```
for(t=1; t<(PER+1); t++){
    u1 = mrand(t);
    u2 = mrand(t+1);
    erro = (cos(6.283 * u2) * sqrt(-2. * log(u1)));
    dd[t]= a + erro;
}
dd[0]=0.;
for(i=1; i<(PER+1); i++)
      dd[0] += dd[i];
dbarra = dd[0]/PER; tbarra= (PER+1)/2.;
xhat = dbarra;
dd[0]=0.;
for(i=1; i<(PER+1); i++)
     dd[0] += pow((dd[i]-dbarra),2);
eqm = dd[0]/(PER-1);
```

//Fim do cálculo da série histórica

// Cálculos da previsão da demanda

```
for(t=(PER+1);  ;t++){
  u1 = mrand(t);
  u2 = mrand(t + 1);
  erro = (cos(6.283 * u2) * sqrt(-2. * log(u1)));
  xhat_ant = xhat;
  demand = a + (b * (t-PER)) + erro;
  xhat = (alfa * demand) + ((1 - alfa) * xhat_ant);
  erro_prev = xhat_ant - demand;
```

```
xhat_ant = xhat;
eqm_ant = eqm;

// GRÁFICO TS
if (graf_ts){
      vies += erro_prev;
      at += fabs(erro_prev);
      dam = at/(t-PER);
      ts = vies/dam;
      if(fabs(ts) > 6 ){
              graf_ts = 0;  kont_ts += (t-PER);
      }
}
 // FIM DO GRÁFICO TS

 // GRÁFICOS EWMA E CUSUM
 if (graf_ewma || graf_cusum){
    eqm = (omega * erro_prev * erro_prev) + ((1 - omega) * eqm_ant);
    epad = erro_prev / sqrt(eqm_ant);
    eqm_ant = eqm;

    // GRÁFICO EWMA
    if (graf_ewma){
       ewma = (lambda * epad) + ((1 - lambda) * ewma_ant);
       ewma_ant = ewma;
       lim = fabs(ke * sqrt(lambda/(2 - lambda)* (1 - pow((1 - lambda),2.*
(t-PER)))));
       if (t==(TAM - 1)) {kont_ewma += TAM; break;}

       if (fabs(ewma) > lim)  {
          graf_ewma = 0; kont_ewma += (t-PER);
```

```
      }
    }
    // FIM DO GRÁFICO EWMA

    // GRÁFICO CUSUM
    if (graf_cusum){
     smais = epad - d + smais;
     smais = (smais > 0) ? smais:0.;
     smenos = -d - epad + smenos;
     smenos = (smenos > 0) ? smenos : 0.;
     if (t==(TAM - 2)) {kont_cusum += TAM ;}

     if ((smais > kc) || (smenos > kc)) {
        graf_cusum = 0;  kont_cusum += (t-PER);
     }
    }
    //FIM DO GRÁFICO CUSUM
 }

   if (!(graf_ts || graf_ewma || graf_cusum))
        break;  // Os três gráficos atingiram os limites
 }
kont++;
}

printf("\nVALORES MÉDIOS :");
printf("\n     E[R] TS = %5.2f",kont_ts/(float)ROD);
printf("\n     E[R] EWMA = %5.2f",kont_ewma/(float)ROD);
printf("\n     E[R] CUSUM = %5.2f\n",kont_cusum/(float)ROD);
return 0;
}
```

- **Modelo Crescente:**

```
int main(void) {
   srand(time(NULL)); // Semente (relógio do sistema) para gerador de aleatórios
   float  erro, demand, ts, eqm, at, vies, dam, u1, u2;
   float  erro_ant, erro_prev, epad, eqm_ant, ewma, ewma_ant, lim, xhat, xhat_ant, alfa, beta, omega, lambda;
   float a, a_ant, a_xh, b, b_ant, b_xh;
   int t;
   int graf_ts, graf_cusum, graf_ewma;
   float smais, smenos, d, ke, kc;
   int i; float dd[PER+1],resid[PER+1], dbarra, tbarra, tbarra2; // Para gerar série histórica
   int kont = 1, kont_ts = 0, kont_ewma = 0, kont_cusum = 0;

   printf("VALORES DAS VARIÁVEIS DE CONTROLE: \n");
   printf("DEMANDA (a + b.t + erro):\n");
   printf("Trata-se do MODELO CRESCENTE !\n");
   printf("Digite o valor de b : "); scanf("%f",&b);
   printf ("Digite o valor de beta :"); scanf("%f",&beta);
   printf("Digite o valor de a : "); scanf("%f",&a);
   printf("Digite o valor de alfa-> "); scanf("%f",&alfa);
   printf("Digite o valor de omega(w)-> "); scanf("%f",&omega);
   printf("Digite o valor de K para o EWMA-> "); scanf("%f",&ke);
   printf("Digite o valor de lambda (parâmetro do EWMA)-> "); scanf("%f",&lambda);
   printf("Digite o valor de K para o CUSUM-> "); scanf("%f",&kc);
   printf("Digite o valor de d (parâmetro do CUSUM)-> "); scanf("%f",&d);

   while (kont <= ROD)  {
```

```
graf_ts = 1, graf_cusum = 1, graf_ewma = 1; smais = 0., smenos = 0.;
at = vies = 0.0;
ewma_ant = 0.0;

// Cálculos para inicializar a previsão da demanda com 36 dados da série histórica
for(t=1; t<(PER+1); t++){
    u1 = mrand(t);
    u2 = mrand(t+1);
    erro = (cos(6.283 * u2) * sqrt(-2. * log(u1)));
    dd[t]= a + (0.05 * t) + erro;
}
dd[0]=0.;
for(i=1; i<(PER+1); i++)
      dd[0] += dd[i];
dbarra=dd[0]/PER; tbarra=(PER+1)/2.;
dd[0]=0.; tbarra2=0.;
for(i=1; i<(PER+1); i++){
      dd[0] += ((i-tbarra) * dd[i]);
      tbarra2 += pow((i-tbarra),2);
}
b_xh = dd[0]/tbarra2;
a_xh = dbarra - (b_xh * tbarra);
xhat = a_xh + (b_xh * PER);
for(i=1; i<(PER+1); i++)
     resid[i] = (a_xh + (b_xh * i))-dd[i];
resid[0]=0.;
for(i=1; i<(PER+1); i++)
     resid[0] += pow(resid[i],2);
eqm = resid[0] / (PER-2);
// Término dos cálculos para série histórica
```

```
// Cálculos da previsão da demanda
for(t=(PER+1);  ;t++){
   u1 = mrand(t);
   u2 = mrand(t + 1);
   erro = (cos(6.283 * u2) * sqrt(-2. * log(u1)));
   xhat_ant = xhat;
   demand = a + (b * (t-PER)) + erro;
   a_ant = a_xh; b_ant = b_xh;
   xhat = a_ant + b_ant;
   a_xh = (alfa * demand) + ((1 - alfa) * xhat);
   b_xh = (beta * (a_xh - a_ant)) + ((1 - beta) * b_ant);
   erro_prev = xhat_ant - demand;
   eqm_ant = eqm;

 // GRÁFICO TS
 if (graf_ts){
     vies += erro_prev;
     at += fabs(erro_prev);
     dam = at/(t-PER);
     ts = vies/dam;
     if(fabs(ts) > 6 ){
         graf_ts = 0;  kont_ts += (t-PER);
     }
 }
 // FIM DO GRÁFICO TS

 // GRÁFICO EWMA E CUSUM
 if (graf_ewma || graf_cusum){
   eqm = (omega * erro_prev * erro_prev) + ((1 - omega) * eqm_ant);
   epad = erro_prev / sqrt(eqm_ant);
   eqm_ant = eqm;
```

```
// GRÁFICO EWMA
if (graf_ewma){
   ewma = (lambda * epad) + ((1 - lambda) * ewma_ant);
   ewma_ant = ewma;
   lim = fabs(ke * sqrt(lambda/(2 - lambda)* (1 - pow((1 - lambda),2.*
(t-PER)))));

   if (t==(TAM - 1)) {kont_ewma += TAM; break;}
   if (fabs(ewma) > lim)  {
      graf_ewma = 0; kont_ewma += (t-PER);
   }
}
// FIM DO GRÁFICO EWMA

// GRÁFICO CUSUM
if (graf_cusum){
   smais = epad - d + smais;
   smais = (smais > 0) ? smais:0.;
   smenos = -d - epad + smenos;
   smenos = (smenos > 0) ? smenos : 0.;

   if (t==(TAM - 2)) {kont_cusum += TAM;}
   if ((smais > kc) || (smenos > kc)) {
   graf_cusum = 0;  kont_cusum += (t-PER);
   }
}
// FIM DO GRÁFICO CUSUM
}

if (!(graf_ts || graf_ewma || graf_cusum))
   break; //termina quando os três gráficos atingirem o limite
```

```
 }
 kont++;
 }

 printf("\nVALORES MÉDIOS :");
 printf("\n     E[R] TS = %5.2f",kont_ts/(float)ROD);
 printf("\n     E[R] EWMA = %5.2f",kont_ewma/(float)ROD);
 printf("\n     E[R] CUSUM = %5.2f\n",kont_cusum/(float)ROD);
return 0;
}
```