Ekonometri ve İstatistik Sayı:1 2005 15-05

İSTANBUL ÜNİVERSİTESİ
İKTİSAT FAKÜLTESİ
EKONOMETRİ VE İSTATİSTİK
DERGİSİ

Rastlantısal Şeritler ile En Küçük Medyan Kareler Doğrusunun Bulunması

Yard. Doç. Dr. Enis Sınıksaran* Dr. Araş. Gör. Aylin Aktükün**

Bu makale 15.12.2004 tarihinde alınmış, hakem kontrolü sonrasında yayını uygun bulunmuştur.

Absract :

The purpose of this paper is to present an algorithm to calculate the least median squares LMS line. In the algorithm, random strips which would catch at least fifty percent of observations are thrown on the convex hull of the data in the variable space. The LMS line lies exactly at the middle of the narrowest strip.

Keywords : The least median squares; Strip; Algorithm; Convex hull; Mathematica.

Özet :

Bu çalışmada amaç En Küçük Medyan Kareler (Least Median Squares) ya da kısaca EKMK doğrusunun hesaplanması için bir algoritma sunmaktır. Algoritmada değişken uzayında (X-Y koordinat sistemi) verinin konveks kabuğu üzerine atılan ve gözlemlerin en az yüzde ellisini içeren rastlantısal şeritler ile EKMK doğrusu aranmaktadır. Bu şeritlerden en dar olanının tam ortasından geçen doğru EKMK doğrusudur.

Anahtar Kelimeler : En küçük medyan kareler; Şerit; Algoritma; Konveks kabuk; Mathematica.

* **Adres :** İstanbul Üniversitesi, İktisat Fakültesi, Ekonometri Bölümü, Tel: (0212) 440 00 00-11659.
E-Mail : saran@istanbul.edu.tr
** **Adres :** İstanbul Üniversitesi, İktisat Fakültesi, Ekonometri Bölümü, Tel: (0212) 440 00 00-11659.
E-Mail : aylin@istanbul.edu.tr

Rastlantısal Şeritler ile En Küçük Medyan Kareler Doğrusunun Bulunması

1. En Küçük Medyan Kareler

Basit doğrusal regresyon modelini ele alalım :

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad , \quad i = 1 , 2 , ... , n$$

Modelde temel amaç $\boldsymbol{\beta} = (\beta_0, \beta_1)$ parametre vektörünün tahmincisi $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1)$ vektörünü elde etmektir. Bu vektörü bulmada EKKY kriteri

$$\underset{\hat{\beta}}{\text{Minimum}} \sum [y_i - (\beta_0 + \beta_1 x_i)]^2 \qquad (1)$$

şeklinde ifade edilebilir. EKKY en yaygın kullanılan ve cebrik işlemlere en uygun tahmin yöntemi olsa da, özellikle son bir kaç 10 yıldır varsayımların ihlaline ve uç değerlere karşı olan dayanıksızlığı nedeniyle eleştirilmekte ve alternatif olarak dayanıklı (robust) yöntemler önerilmektedir. Bu çalışmada amaç dayanıklı yöntemlerden En Küçük Medyan Kareler (Least Median Squares) ya da kısaca EKMK tahminlerinin hesabı için bir algoritma sunmaktır.

Rousseeuw (1984) EKMK tahminini, (1) 'deki amaç fonksiyonunda "Σ" yerine "Medyan" koymak olarak tanımlar :

$$\underset{\hat{\beta}}{\text{Minimum}} \text{ Medyan } [y_i - (\beta_0 + \beta_1 x_i)]^2 \qquad (2)$$

Ancak (1) 'deki amaç fonksiyonundan küçük bir değişiklikle elde edilen bu yeni amaç fonksiyonunda analitik bir çözüm elde etmek olanaksız olup, amaca ulaşmayı sağlayacak β_0 ve β_1 tahminlerinin sayısal değerleri bilgisayar iterasyonları ile bulunabilir. Rousseeuw (1984), Rousseeuw ve Leroy (1987), Edelsbrunner ve Souvaine (1990), Olson (1997) ve Mount ve Ark.(1997) bu iterasyonlar için çeşitli algoritmalar önermişlerdir. Yaygın olarak kullanılan Rousseeuw (1984) algoritmasında, verinin tüm mümkün alt kümelerine EKKY uygulanır ve her biri için kalıntıların medyanı hesaplanır. Bu medyanlar içinde minimuma sahip olan alt kümenin EKKY tahminleri EKMK tahmini olarak kabul edilir.

Küçük veriler için EKMK tahminlerinin kesin değerlerini hesaplamak olanaklı ise de büyük verilerde tüm mümkün alt kümelerin taranması büyük işlem yükü getireceğinden genellikle olanaksızdır. Bu durumda bazı alt kümelerin rastlantısal olarak çekilmesi ve amaç fonksiyonuna bu alt kümelerin uygulanması düşünülebilir. Rousseeuw ve Leroy (1987) belirli kısıtlar altında veriden rastlantısal olarak çekilen en az bir alt kümenin istenilen sonucu verme olasılığının bire yakın olduğunu ispatlamıştır. Buna göre n birimlik bir veriden p elemanlı m tane altküme seçtiğimizde p tane aşırı olmayan değer içeren en az bir alt kümeye rastlama olasılığı n/p 'nin çok büyük değerleri için aşağıdaki ifadeye eşittir:

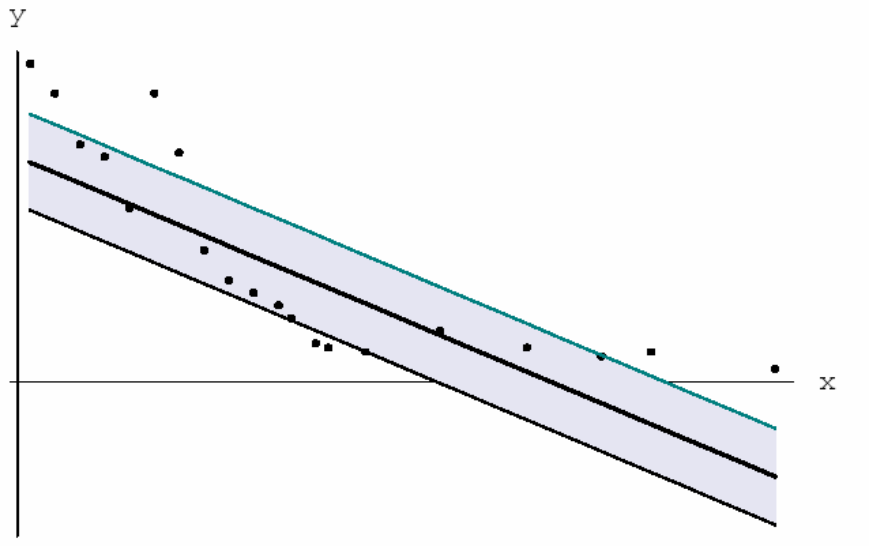
$$1 - [1 - (1 - \varepsilon)^p]^m$$

Burada ε, verideki kirlenme oranını göstermektedir. Öyleyse, örneğin kirlenme oranının % 30 olduğu bir veriden 5 birimlik alt kümeler çektiğimizde bunlardan en az birinin tamamen iyi (aşırı olmayan) gözlemlerden oluşma olasılığının 0,95 olmasını istiyorsak, çekmemiz gereken 5 birimlik alt kümelerin sayısı (m):

$$0{,}95 = 1 - [1 - (1 - \varepsilon)^5]^m$$

eşitliğinden yaklaşık m = 17 olarak bulunabilir. Rousseeuw kendi yazdığı ve bir Fortran programı olan PROGRESS 'de yukarıda betimlenen rastlantısal algoritmayı kullanmıştır. Algoritma zaman bakımından O(nlogn) büyüklük mertebesi ile literatürdeki en iyi algoritmadır.

Öte yandan Rousseeuw (1984), EKMK tahminlerinin $n^{-1/3}$ yakınsama hızı nedeniyle EKKY tahminlerinden daha etkin (efficient) olmadığını da belirtmiştir. Yine de % 50 kirlenmeye karşı dayanabilmesi nedeniyle EKMK, dayanıklı yöntemler içinde yaygın olarak kullanılmaktadır. Rousseeuw ve Zomeren (1990), ayrıca, EKMK yöntemini uçdeğerlerin teşhisinde kullanmayı önermiştir. Buna göre standartlaştırılmış EKMK kalıntı değerleri ile MVE (Minimum Volume Ellpsoid) 'den hesaplanan dayanıklı uzaklıkları birlikte gösteren bir grafik yardımıyla, Mahalanobis Uzaklıkları gibi klasik teşhisçilerin kaçırdığı uçdeğerler tespit edilebilmektedir. Burada standartlaştırılmış EKMK kalıntı değeri (-2.5 , 2.5) aralığına düşen gözlemler potansiyel uçdeğer olarak düşünülmektedir.



Şekil 1: Gözlemlerin % 50 'den fazlasını içeren bir rastlantısal şerit

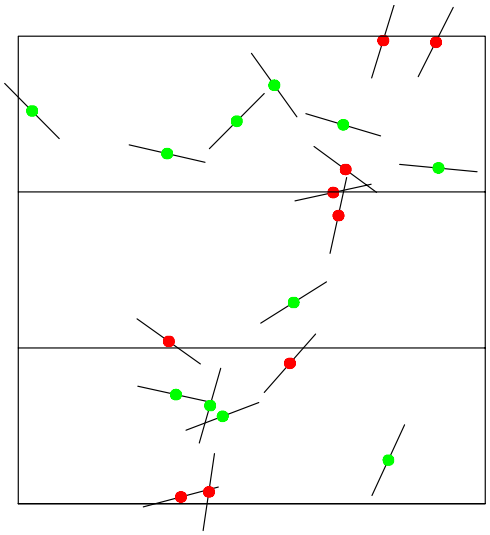
Bu çalışmada sunulan algoritma, (2) 'de tanımlanan amaç fonksiyonunu sağlayan β_0 ve β_1 tahminlerinin belirlediği doğrunun Rousseeuw (1984) tarafından yapılan

geometrik tanımına dayanmaktadır. Buna göre X-Y değişken uzayında yani bağımsız ve bağımlı değişkenlerin eksen olarak seçildiği ve yaygın olarak serpilme diyagramı olarak adlandırılan 2 boyutlu koordinat sisteminde, gözlemlerin en az yüzde ellisini içeren şeritlerden en dar olanının (y yönünde ölçüldüğünde) tam ortasından geçen doğru EKMK doğrusudur. Şekil 1 'de 20 gözlemden oluşan bir verinin serpilme diyagramını ve verinin 11 gözlemini yani % 55 'ini içeren bir şerit ile şeridin tam ortasından geçen muhtemel EKMK doğrusu görülmektedir. Şüphesiz verinin en az % 50 'sini içeren böyle bir çok şerit söz konusu olacaktır. Amaç, bu şeritlerden en dar olanını bulmaktır. Steele ve Steiger (1986) bu şeridin belirlediği iki doğrudan birisinin gözlemlerden en az ikisinden, diğer doğrunun ise en az birinden geçeceğini ispatlamıştır. Souvaine ve Steele (1987) bu olguya dayanarak dual uzayda iki farklı EKMK algoritması önermiş, Edelsbrunner ve Souvaine (1990) ise güdümlü topolojik süpürme (guided topological sweep) yöntemi ile yine dual uzayda etkin bir algoritma önermiş, ayrıca minimum şerit kalınlığının dual uzayda hesaplanabileceğini kanıtlamışlardır.

2. Algoritma ve Uygulama

EKMK doğrusunun geometrik tanımına dayanarak X-Y uzayında bir çözüm arıyorsak akla gelebilecek en basit algoritma bu uzaya rastlantısal şeritler atmak, bu şeritlerden gözlemlerin % 50 'sini içerenleri saptayıp, en dar olanını seçmek olacaktır. Ancak X-Y uzayı nın her bölgesinde bu taramayı yapmak zaman ve işlem açısından oldukça hantal bir süreç olup taranacak bölgede bir sınırlama yapmak anlamlı olacaktır.

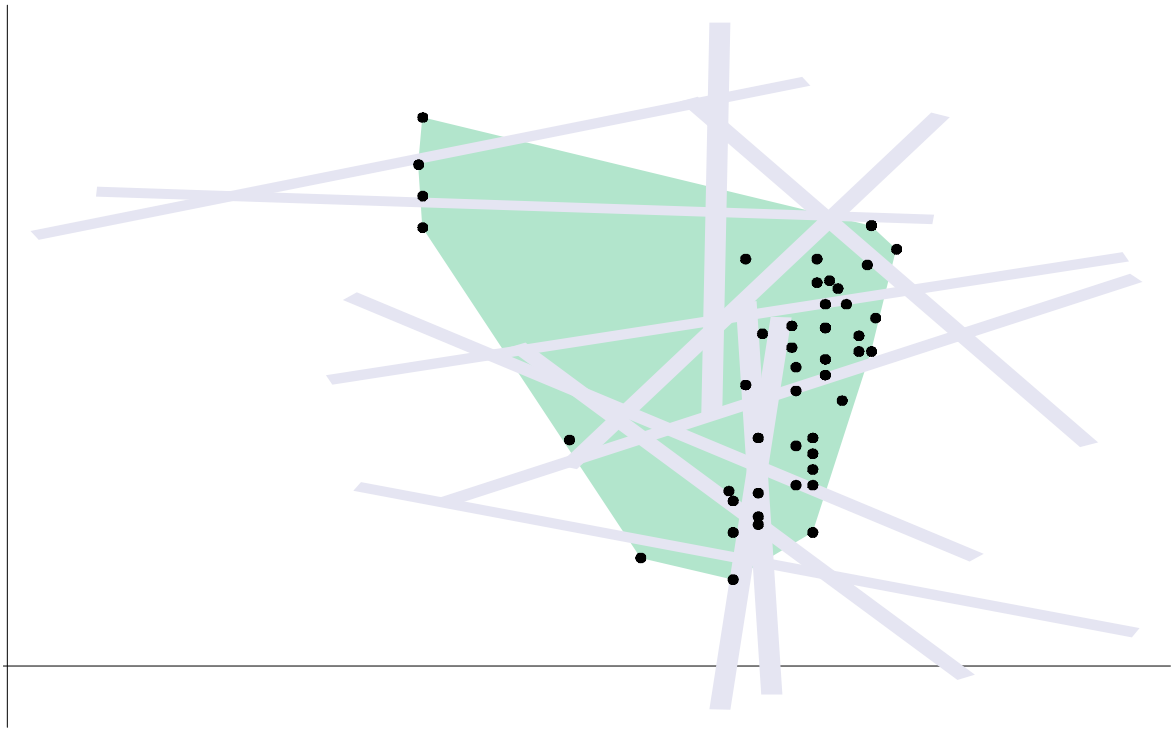
Muhtemel EKMK doğrusunu içerecek şeridin, verinin konveks kabuğunu[1] (Convex Hull) kesmek zorunda olacağı açıktır. Öyleyse X-Y uzayının her bölgesinde tarama yapmak yerine rastlantısal şeritleri konveks kabuk üzerine atmak işlem ve zaman tasarrufu sağlayarak, algoritmayı etkinleştirecektir.



Şekil 2: Buffon 'un iğne deneyi

[1] Bir verinin konveks kabuğu, gözlemleri sınırlayan, bir anlamda bir çit çeken konveks poligondur.

Böyle bir süreç, Buffon 'un iğne problemi olarak bilinen deneye benzerlik içermektedir (Bu deneyin çeşitli versiyonları ve istatistik bağlamlarına ilişkin tartışmalar için bkz. Perlman ve Wichura (1975), Solomon (1978) ve Wood ve Robertson (1997)). Buffon deneyinde iğneler yatay paralel çizgiler çizilmiş düzgün bir yüzey üzerine atıldığında, iğnenin yatay paralel çizgileri kesip kesmemesi ile ilgilenilir. İğnenin çizgileri kesmesi, iki değişkene bağlıdır: İğnenin düşme açısı ve çizgilere yakınlığı. Dolayısıyla böyle bir deneyin simülasyonunu yaparken belirli bir bölgede rastlantısal bir nokta (iğnenin orta noktası olarak düşünülebilir) ve rastlantısal bir açı seçmek yeterli olacaktır. Şekil 2, *Mathematica 4.0* 'da yaptığımız Buffon deneyinin sonuçlarını göstermektedir. Deneyde iğnelerin orta noktaları için dikdörtgen bölgelerin sınırlarına girmek üzere uniform dağılımdan bağımsız olarak yirmişer rastlantısal sayı seçilmiştir. İğnelerin düşme açıları için de yine Uniform dağılımdan $(0, \pi)$ aralığından yirmi rastlantısal sayı seçilerek iğnelerin konumları belirlenmiştir. Şekil 2 'de yirmi iğneden dokuzunun yatay çizgileri kestiği görülmektedir.



Şekil 3 : Konveks kabuk üzerinde rastlantısal şeritler

EKMK doğrusunu bulmada Buffon deneyinden öykündük. Şeritleri iğneler, şeritlerin üzerine atılacağı yüzeyi ise konveks kabuk olarak düşündük. Algoritmayı aşağıdaki gibi özetleyebiliriz:
1) X-Y uzayında şeritlerin konumlarını Buffon deneyinde olduğu gibi iki değişken belirlemiştir:

- Şeritlerin ağırlık merkezi
- Şeritlerin yatayla yaptığı açı

Her iki değişken de uniform dağılan bağımsız rastlantı değişkenleri olarak varsayılmıştır. Ağırlık merkezleri için rastlantısal sayıların seçileceği aralık, diğer bir deyişle uniform dağılımın parametreleri bağımsız değişkene ilişkin gözlemlerin en küçük ve en büyük değerleri olarak belirlenmiştir. Açılar içinse doğallıkla $(0,\pi)$ aralığı seçilmiştir.

2) Verini konveks kabuğunun sınırladığı bölge belirlenir ve çizilir.

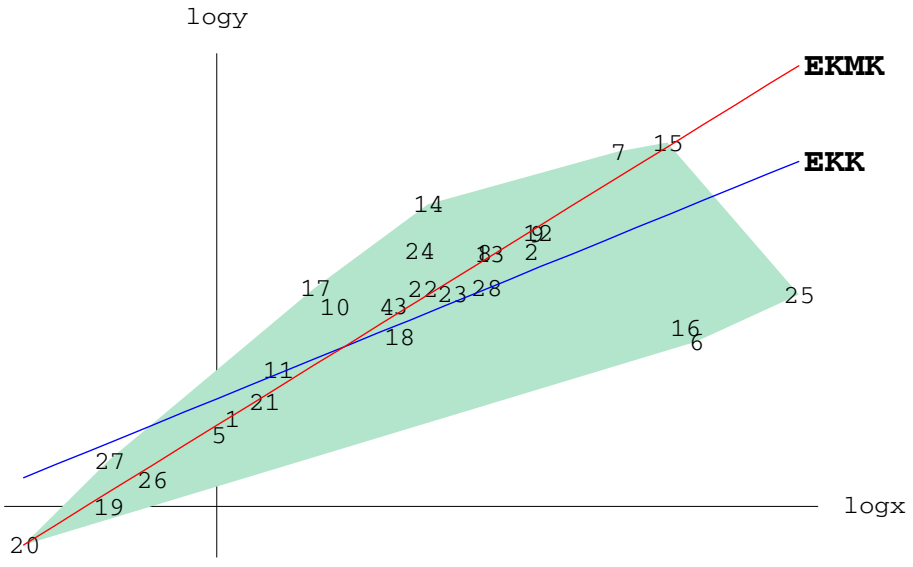
3) Rastlantısal sayılar çekilerek rastlantısal şeritler belirlenir. Şerit boyu konveks kabuğun en geniş kısmına eşit olarak seçilir. Şeridin genişliği için ise önce alabileceği en küçük değer seçilir. Bu değer (tanım gereği dikey eksen yönünde ölçüldüğü için) bağımlı değişkene ilişkin gözlemlerin en büyük ve en küçük değeri arasındaki farkın 0,0001' i olarak belirlendi.

4) Şeritlerden konveks kabuk üzerine düşenler ve bunların içinden de verinin en az % 50 sini içerenler saptanır. Örnek olarak Şekil 3 'de bir verinin konveks kabuğu ve üzerinde rastlantısal şeritler görülmektedir.

5) Şeritlerin kalınlıkları bir önceki kalınlığın 0,1 'i kadar artırılır. Konveks kabuğun en geniş kısmına eşit olunca durdurulur.

6) Verinin % 50 'sini içeren şeritlerden en dar olanı belirlenir. Bu şeridin ortasından geçen doğru EKMK doğrusu olarak hesaplanır.

Algoritmanın programı *Mathematica 4.0* dilinde yazılmış olup Strip[veri,n] fonksiyonu kullanıcıya konveks kabuk üzerine atılacak şerit sayısını (n) belirleme olanağı sunmaktadır. Program verinin serpilme diyagramı ile hesapladığı en dar şeridin ve bu şeridin ortasından geçen EKMK doğrusunun grafiğinin yanı sıra, hesaplanan en dar şeridin genişliği ve EKMK parametre tahminleri de verilmektedir Yaptığımız Monte Carlo simülasyonlar n 'nin 1000 seçilmesinin tatminkar sonuçlar verdiğini göstermiştir.



Şekil 4: 28 Memelinin vücut ve beyin ağırlıklarının serpilme diyagramı ve konveks kabuk

Algoritmayı, Rousseeuw ve Leroy 'un (1987) çok bilinen bir verisine uyguladık. Veri 28 memelinin vücut ağırlıkları (x) ile beyin ağırlıklarından (y) oluşmaktadır. Rousseeuw ve Leroy söz konusu değişkenlere

$$\log y = \beta_0 + \beta_1 \log x + \varepsilon$$

modelini uygulamıştır. Veriye ilişkin EKK ve EKMK tahmin denklemleri aşağıda verilmiştir:

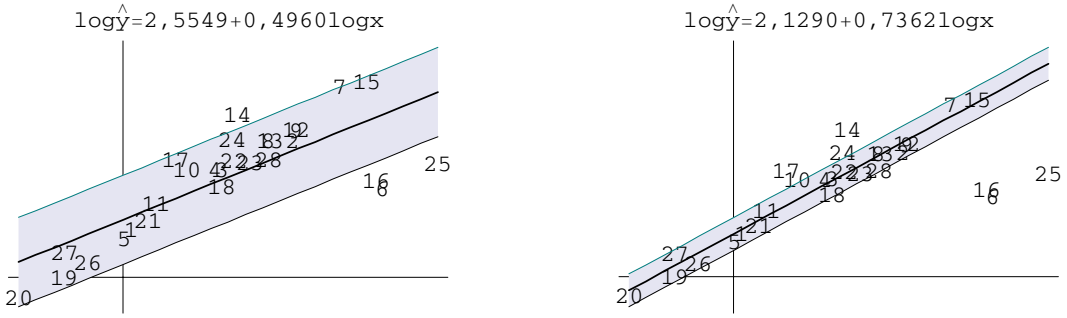
$$EKK : \log\hat{y} = 2{,}5549 + 0{,}49600 \log x$$

$$EKMK : \log\hat{y} = 1{,}92148 + 0{,}7518 \log x$$

Veri Tablo 1 'de verilmiş olup, verinin serpilme diyagramı EKK ve EKMK doğruları ile konveks kabuk Şekil 4 'de görülmektedir. Verideki 6, 16 ve 25 numaralı gözlemler uç değerler olup, EKK doğrusuna kaldıraç etkisi yaptıkları açıkça görülmektedir. Dayanıklı bir yöntem olan EKMK ise bu uçdeğerlerden etkilenmemiştir.

	Log(y)	Log[x]
1	0.300	2.092
2	6.142	6.047
3	3.593	4.783
4	3.320	4.745
5	0.039	1.705
6	9.367	3.912
7	7.843	8.434
8	5.232	6.038
9	6.256	6.485
10	2.303	4.745
11	1.194	3.243
12	6.271	6.522
13	5.333	6.006
14	4.127	7.185
15	8.803	8.650
16	9.148	4.248
17	1.917	5.187
18	3.555	4.025
19	-2.120	0.000
20	-3.772	-0.916
21	0.916	2.493
22	4.016	5.165
23	4.605	5.056
24	3.954	6.087
25	11.370	5.040
26	-1.273	0.641
27	-2.104	1.099
28	5.257	5.193

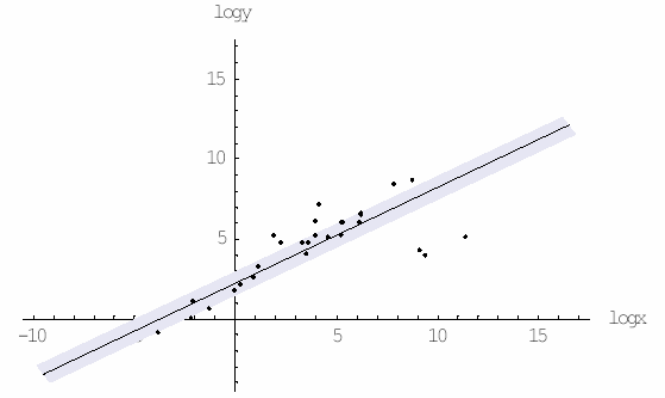
Tablo 1: 28 Memelinin Vücut (Kg.) ve Beyin Ağırlıklarının (Gr.) Logaritmaları



Şekil 5 : İki Farklı Şerit ve EKMK Doğrusu Adayları

Şekil 5 'de yukarıda betimlediğimiz süreçte elde edilen 2 farklı şerit ve bu şeritlerin ortalarının belirlediği EKMK doğrusu adayları görülmektedir. Kalınlıkları ve pozisyonları farklı olan her iki şeritte verinin % 50 'den fazlasını içermektedir. Kalın şeridin EKMK tahmininin EKK tahmini ile aynı değerleri verdiğine dikkat ediniz. Daha ince olan şeridin bekleneceği gibi gerçek EKMK tahmin doğrusuna yakın olduğu görülmektedir.

Aşağıda programın çıktısı görülmektedir. Program en dar şeridi hesaplamış; verinin serpilme diyagramı ve şeridin ortasından geçen EKMK doğrusunun grafiğini vermiştir. Çıktıda ayrıca şeridin genişliğini ve doğruyu belirleyen EKMK parametre tahminlerini görmekteyiz.




```
**Parametre Tahminleri Ve Şerit Kalınlığı**
β̂0                  1,92148
β̂1                  0,7518
Şerit Kalınlığı     1.21677
```

3. Sonuç

Bu çalışmada EKMK doğrusunun elde edilmesine ilişkin geometrik bir algoritma sunduk. Sunulan algoritmanın zaman ve işlem hacmi açısından literatürdeki diğer

algoritmalardan daha etkin olduğunu söyleyemeyiz. Ancak EKMK yönteminin geometrik mantığının kolay anlaşılması ve iterasyon sonuçlarının görsel izlenebilmesi açısından yararlı bir süreç olarak düşünülebilir.

Kaynakça

- Edelsbrunner, H., Souvaine, D.L. (1990), "Computing Least Median of Squares Regression Lines and Guided Topological Sweep", **Journal of the American Statistical Association**, 85, 115-119.

- Mount, D.M., Netanyahu, N.S., Romanik, K., Silverman, R., Wu, A.Y. (1997), "A Practical Approximation Algorithm for the LMS Line Estimator", **8th Ann. ACMSIAM Symposium on Discrete Algorithms**, 473-482.

- Olson, C.F. (1997), "An Approximation Algorithm for Least Median of Squares Regression", **Information Processing Letters**, 63, 237-241.

- Pearlman, M.D, Wichura, M.J (1975), "Sharpening Buffon's Needle", **American Statistician**, 29,157-163

- Rousseeuw, P.J. (1984), "Least Median of Squares Regression", **Journal of the American Statistical Association**, 79, 871-880.

- Rousseeuw, P.J., Leroy, A.M. (1987), **Robust Regression and Outlier Detection**, John Wiley&Sons, Canada, 26.

- Solomon, H. (1978), **GeometricProbability**, SIAM, Philadelphia.

- Souvaine, D.L., Steele, J.M. (1987), "Time- and Space- Efficient Algorithms for Least Median of Squares Regression", **Journal of the American Statistical Association**, 82, 794-801.

- Steele, J.M, Steiger, W.L. (1986), "Algorithms and Complexity for Least Median of Squares Regression", **Discrete Applied Mathematics**, 14, 93-100.

- Wood, G.R., Robertson, J.M. (1998), "Buffon Got It Straight", **Statistics&Probability Letters**, 37,415-421.

Ek : Algoritmanın Mathematica kodları aşağıda verilmiştir :

```
Needs[Statistics`LinearRegression`]
Needs[Statistics`MultiDescriptiveStatistics`]
Needs[DiscreteMath`Combinatorica`]
Needs[Statistics`DescriptiveStatistics`]
Needs[Graphics`Graphics`]
Needs[DiscreteMath`ComputationalGeometry`]
Strip[data_, n_] :=
 Module[{alar, bler, w, l, beta, middleofneddlesabss, middleofneddlesordd,
   tt, ff, CHkesenler, middleofneddlesabs, middleofneddlesord, z, angles, abysis1, abysis2,
   ordinates1, ordinates2, k, pols, prot, convekshull, cvd, convexd, tr, kesenler, post
   , pos, kesenstripler, x1, x2, y1, y2, intercepts, slopes, lmsline, t, tp},
  alar = Transpose[data][[1]];
  bler = Transpose[data][[2]];
w = Max[Transpose[data][[2]]] - Min[Transpose[data][[2]]];
w = .0001 * w;
l = 2 (Max[Transpose[data][[1]]] - Min[Transpose[data][[1]]]);
(Label[begin];
w = w + .1 * w;
beta = N[180 / Pi * ArcTan[l, w]];
middleofneddlesabss = Table[Random[Real, {Min[alar], Max[alar]}], {n}];
middleofneddlesordd = Table[Random[Real, {Min[bler], Max[bler]}], {n}];
cvd = ConvexHull[data];
convexd = Table[data[[cvd[[i]]]], {i, 1, Length[cvd]}];
mids = Transpose[{middleofneddlesabss, middleofneddlesordd}];
tt = Table[PLCPolygon[convexd, mids[[i]]], {i, 1, n}];
ff = Flatten[Position[tt, 1]];
CHkesenler = Table[mids[[ff[[i]]]], {i, 1, Length[ff]}];
middleofneddlesabs = Transpose[CHkesenler][[1]];
middleofneddlesord = Transpose[CHkesenler][[2]];
z = Length[CHkesenler];
angles = Table[Random[Real, {0, 180}], {z}];
abysis1 = Table[middleofneddlesabs[[i]] + (l / 2) * Cos[angles[[i]] Degree], {i, 1, z}];
abysis2 = Table[middleofneddlesabs[[i]] - (l / 2) * Cos[angles[[i]] Degree], {i, 1, z}];
ordinates1 = Table[middleofneddlesord[[i]] + (l / 2) * Sin[angles[[i]] Degree], {i, 1, z}];
ordinates2 = Table[middleofneddlesord[[i]] - (l / 2) * Sin[angles[[i]] Degree], {i, 1, z}];
k = IntegerPart[Length[data] / 2] + 1;
pols = Table[{{middleofneddlesabs[[i]] + (l / 2) * Sqrt[(l^2 + w^2)] * Cos[(angles[[i]] - beta) Degree],
        middleofneddlesord[[i]] + (l / 2) * Sqrt[(l^2 + w^2)] * Sin[(angles[[i]] - beta) Degree]},
       {middleofneddlesabs[[i]] + (l / 2) * Sqrt[(l^2 + w^2)] * Cos[(angles[[i]] + beta) Degree],
        middleofneddlesord[[i]] + (l / 2) * Sqrt[(l^2 + w^2)] * Sin[(angles[[i]] + beta) Degree]},
       {middleofneddlesabs[[i]] - (l / 2) * Sqrt[(l^2 + w^2)] * Cos[(angles[[i]] - beta) Degree],
        middleofneddlesord[[i]] - (l / 2) * Sqrt[(l^2 + w^2)] * Sin[(angles[[i]] - beta) Degree]},
       {middleofneddlesabs[[i]] - (l / 2) * Sqrt[(l^2 + w^2)] * Cos[(angles[[i]] + beta) Degree],
        middleofneddlesord[[i]] - (l / 2) * Sqrt[(l^2 + w^2)] * Sin[(angles[[i]] + beta) Degree]}
      }, {i, 1, z}];
prot = Table[Polygon[pols[[i]]], {i, 1, z}];
convekshull = Show[Graphics[{{RGBColor[1, 0, 0], prot}}], DisplayFunction → Identity];
cvd = ConvexHull[data];
convexd = Table[data[[cvd[[i]]]], {i, 1, Length[cvd]}];
```

```
kes = Table[PLCPolygon[pols[[j]], data[[i]]], {i, 1, Length[data]}, {j, 1, z}];
tr = Table[Cases[Transpose[kes][[i]], 1], {i, 1, z}];
kesenler = Table[Length[tr[[i]]], {i, 1, z}];
post = Select[kesenler, # ≥ k &];
   If[Length[post] == 0, Goto[begin]];
pos = Flatten[Table[Flatten[Position[kesenler, Union[post][[i]]]], {i, 1, Length[Union[post]]}]];
kesenstripler = Table[pols[[pos[[i]]]], {i, 1, Length[pos]}];
xl = Table[Table[abysis1[[pos[[i]]]], {i, 1, Length[pos]}][[i]], {i, 1, Length[pos]}];
x2 = Table[Table[abysis2[[pos[[i]]]], {i, 1, Length[pos]}][[i]], {i, 1, Length[pos]}];
yl = Table[Table[ordinates1[[pos[[i]]]], {i, 1, Length[pos]}][[i]], {i, 1, Length[pos]}];
y2 = Table[Table[ordinates2[[pos[[i]]]], {i, 1, Length[pos]}][[i]], {i, 1, Length[pos]}];
slopes = Table[(y2[[i]] - yl[[i]]) / (x2[[i]] - xl[[i]]), {i, 1, Length[pos]}];
intercepts = Table[yl[[i]] - ((y2[[i]] - yl[[i]]) / (x2[[i]] - xl[[i]])) * xl[[i]], {i, 1, Length[pos]}]
ssnew = Show[Graphics[{Thickness[.007], Table[Line[{{xl[[i]], yl[[i]]}, {x2[[i]], y2[[i]]}}],
       {i, 1, Length[pos]}]}], PlotRange → All, Axes → None, AspectRatio → 1, ImageSize → {400, 400},
      DisplayFunction → Identity];
farklar = Table[Max[Transpose[kesenstripler[[i]]][[2]]] - Min[Transpose[kesenstripler[[i]]][[2]]]
          {i, 1, Length[kesenstripler]}];
minw = Min[Table[Max[Transpose[kesenstripler[[i]]][[2]]] - Min[Transpose[kesenstripler[[i]]][[2]]]
         {i, 1, Length[kesenstripler]}]];
ninwstrip = kesenstripler[[Flatten[Position[farklar, minw]][[1]]]];
ninwstripgraph = Show[Graphics[{{RGBColor[.90, .90, .95], Polygon[ninwstrip]}}],
                      DisplayFunction → Identity];
xl = (ninwstrip[[2]][[1]] + ninwstrip[[1]][[1]]) / 2;
yl = (ninwstrip[[2]][[2]] + ninwstrip[[1]][[2]]) / 2;
x2 = (ninwstrip[[4]][[1]] + ninwstrip[[3]][[1]]) / 2;
y2 = (ninwstrip[[4]][[2]] + ninwstrip[[3]][[2]]) / 2;
slope = SetPrecision[(y2 - yl) / (x2 - xl), 2];
intercept = SetPrecision[yl - slope * xl, 4];
Clear[x];
f[t_] := intercept + slope * t;
t = .875 xl;
lmsline = Show[Graphics[{{Thickness[.005], Line[{{xl, yl}, {x2, y2}}]},
       Text[FontForm[, {Courier, 10}], {.85 xl, 1.4 yl}, {0, -1}]}],
       DisplayFunction → Identity];
tp = Show[ListPlot[data, PlotStyle → {RGBColor[0, 0, 0], PointSize[1 / 85]},
          DisplayFunction → Identity]];
Show[ninwstripgraph, lmsline, tp, Axes → True, AxesLabel → {x, y},
       DisplayFunction → $DisplayFunction];
 Print[**Parametre Tahminleri Ve Şerit Kalınlığı**];
Print[TableForm[{{β̂0, intercept}, {β̂1, slope}, {Şerit Kalınlığı, w}}]];
```