KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN FAKÜLTESİ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ BÖLÜMÜ

PARAMETRİK OLMAYAN İSTATİSTİKSEL YÖNTEMLER İÇİN JAVASCRIPT KÜTÜPHANESİ

LİSANS TEZİ

REGAİP ERGENEKON YİĞİT

TRABZON
2018

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN FAKÜLTESİ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ BÖLÜMÜ

PARAMETRİK OLMAYAN İSTATİSTİKSEL YÖNTEMLER İÇİN JAVASCRIPT KÜTÜPHANESİ

REGAİP ERGENEKON YİĞİT

Tez Danışmanı : Dr. Öğr. Üyesi TOLGA BERBER

Jüri Üyeleri

Danışman : Dr. Öğr. Üyesi Tolga BERBER …...……

Üye : Dr. Öğr. Üyesi Halil İbrahim ŞAHİN …...……

Üye : Doç. Dr. Orhan KESEMEN ……...……

Yedek : Prof. Dr. Zafer KÜÇÜK ……...……

2018

ÖNSÖZ

"Parametrik Olmayan İstatistiksel Yöntemler için JavaScript Kütüphanesi" isimli bu tez Karadeniz Teknik Üniversitesi Fen Fakültesi İstatistik ve Bilgisayar Bilimleri Bölümü, Lisans Programı'nda hazırlanmıştır.

Başta tez çalışması süresince değerli yardım ve katkılarıyla beni yönlendiren danışman hocam sayın Dr. Öğr. Üyesi Tolga BERBER'e, lisans eğitimi yapma fırsatı bulduğum İstatistik ve Bilgisayar Bilimleri Bölümündeki tüm hocalarıma teşekkürü borç bilirim.

Son olarak, tüm hayatım boyunca maddi ve manevi her zaman beni destekleyen, her adımımda arkamda duran anneme sonsuz teşekkürlerimi sunarım.

Bu tezin, bundan sonraki çalışmalara katkı sağlamasını temenni ederim.

REGAİP ERGENEKON YİĞİT

Trabzon 2018

İÇİNDEKİLER

Sayfa No

Lisans Tezi

ÖZET

PARAMETRİK OLMAYAN İSTATİSTİKSEL YÖNTEMLER İÇİN JAVASCRIPT KÜTÜPHANESİ

REGAİP ERGENEKON YİĞİT

Karadeniz Teknik Üniversitesi
Fen Fakültesi
İstatistik ve Bilgisayar Bilimleri Bölümü
Danışman: Dr. Öğr. Üyesi Tolga BERBER
2018, 61 Sayfa

Araştırma, bir gerçeği ortaya çıkarmak, bir sorunu çözümlemek ve eldeki verileri arttırmak için bilimsel yöntem ve tekniklerden yararlanılarak yapılan düzenli çalışmadır. Araştırma sonucunda elde edilen verilerin doğru yorumlanabilmesi, araştırmanın amaca uygun şekilde yapılması ve analiz sırasında kullanılan istatistiksel testlerin doğru seçilmesine bağlıdır. Test seçimi, yapılan araştırmanın amacı ve elde edilen verinin; ölçme düzeyi, değişken türü, dağılım biçimi ile varyans homojenliği gibi özellikleri göz önünde bulundurarak yapılır. Araştırmalarda birbirinden farklı ölçme düzeyine sahip birden çok değişken olduğu durumlarda, araştırmacılar için uygun test seçimi karmaşık bir problem haline gelmektedir. Özellikle tıp alanında yapılan bazı çalışmalarda doğru istatistiksel yöntemlerin kullanılmadığını gösteren birçok araştırma mevcuttur. Literatürde test seçimi konusunda araştırmacılara yardımcı olması amacıyla yapılmış çeşitli çalışmalar mevcuttur. Bu çalışmalarda test seçimi için cevaplanması gereken bazı sorunlar ile karar ağaçları ve tabloların kullanılması önerilmiştir. Bunun yanında bazı istatistiksel paket programlarının veri girişi sonrasında yapılacak testi önerme özelliği bulunmaktadır. Bu çalışmada, parametrik olmayan istatistiksel testlerin istemci ve sunucu tarafında hesaplatılabilmesini sağlayan bir JavaScript kütüphanesi hazırlanmak amaçlanmıştır.

Anahtar Kelimeler: Parametrik Olmayan İstatistiksel Yöntemler, JavaScript Kütüphanesi

Thesis

SUMMARY

JAVASCRIPT LIBRARY FOR NONPARAMETRIC STATISTICAL METHODS

REGAİP ERGENEKON YİĞİT

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Statistical and Computer Science Undergraduate Program
Supervisor: Assist. Prof. Dr. Tolga BERBER
2018, 61 Pages

Research is the systematic work of using scientific methods and techniques to reveal a truth, to solve a problem and to increase the amount of data available. The exact clarification of the data obtained from the results of research depends on the suitability of the research methods and the correct selection of the statistical tests used during the analysis. The choice of statistical tests should be based on the purpose of the study conducted and the properties of acquired data such as measurement level, variable type, distribution form and homogeneity of variance. The choice of the proper test becomes a complex problem for researchers, when there are multiple variables with different levels of measurement in study. Especially, there are some studies in literature proving that some statistical methods are used inappropriately in medical studies. There are various studies in the literature to help researchers in selecting the proper statistical test. In these studies, it has been suggested to use some questions with decision trees and tables for statistical test selection process. In addition, some statistical package programs can suggest tests to be performed after data entry. In this study, it is aimed to prepare a JavaScript library that allows non-parametric statistical tests to be calculated by the client or server.

Key Words: Nonparametric Statistical Methods, JavaScript Library

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

H_0 : Yokluk hipotezi

H_1 : Karşıt hipotez

G_j : j. sınıftaki gözlenen frekans

B_j : j. sınıftaki beklenen frekans

χ^2 : Ki-kare test istatistiği

$F(x)$: Birikimli dağılım fonksiyonu

X : Rastgele değişken

$S(x)$: Gözlenen birikimli dağılım fonksiyonu

$\overline{X}$: Örneklem ortalaması

S^2 : Örneklem varyansının tahmin edicisi

N : Örneklem hacmi

σ^2 : Örneklem varyansı

$GAKT$: Gruplar arası kareler toplamı

$GIKT$: Gruplar içi kareler toplamı

$GAOK$: Gruplar arası ortalama kare

$GIOK$: Gruplar içi ortalama kare

$F_{\alpha,\beta}$: α, β serbestlik dereceli F dağılımı test istatistiği

z : Normal dağılım test istatistiği

t : Student t dağılımı test istatistiği

$R(X_i)$: X örneğinin i. sıra sayısı

$\overline{R}_{.j}$: j. örnek için sıra sayıları ortalaması

1. TEMEL KAVRAMLAR

1.1. İstatistik

İstatistik örneği betimlemek için hesaplanan değerlerdir. İstatistik hesaplamanın amaçlarından biri parametreyi tahmin etmektir. Örnekten hesaplanan $\overline{X}, P, S^2$ ve r istatistikleri sırasıyla, μ, Π, σ^2 ve p parametreleri tahmin edilir.

1.1.1. Yığın (Kitle)

Yığın, amaca uygun olarak belirli özelliğe sahip olan birimlerin topluluğu veya araştırmacının ilgi alanına giren birimler topluluğu olarak tanımlanır. Yığının kapsamı da yine araştırmacı tarafından belirlenir. Bir eğitim uzmanı okullardaki eğitim kalitesi konusunda bir araştırma yapıyorsa bu araştırmacı için belirlediği okullardaki öğrencilerin notları yığındır.

1.1.2. Örnek (Örneklem)

Yığının birim sayısı çok fazla ise yığındaki birimlerden bilgi toplamak çok zor veya imkânsız olabilir. Böyle durumlarda yığının bazı birimlerinden bilgi derleme yoluna gidilir. Yığındaki birimler içinden seçilen ve yığının özelliklerini taşıdığı kabul edilen az sayıda birimden oluşan kümeye örnek (örneklem) denir.

1.2. JavaScript

JavaScript web sitelerinde etkileşimler gerçekleştirilebilmesi için çıkmış dinamik bir programlama dilidir. 1995 yılında Netscape şirketinin aynı isimdeki internet tarayıcısı olan Netscape, web sitelerinin etkileşimini arttırmak amacıyla mühendislerinden Brendan Eich tarafından 10 gün içerisinde ilk versiyonu ortaya çıkmıştır. Yayınlandığı zaman adı LiveScript olan dil 1 yıl geçmeden JavaScript ismini almıştır. Microsoft bu dili destekleyen ilk büyük teknoloji şirket olmuştur. JavaScript, ilk versiyonunun yayınlanmasından 1 yıl sonra versiyonlama konusunda standart bir tarifname oluşturmak amacıyla ECMA

komitesine dahil olmuştur ve dili geliştiren topluluğa da TC-39 ismi verilir. 1997 yılında bu standardın ilk versiyonu olan ECMAScript 1 (ES1) yayınlanmıştır. Böylece diğer tarayıcı geliştiricileri bu standarda uygun şekilde kendi tarayıcılarında gerekli iyileştirmeleri gerçekleştirmeleri amaçlanmıştır. Standartlaştırma süreci bu döngüde bugüne kadar devam etmiştir ve etmeye devam etmektedir. 1998'de ECMAScript 2 (ES2) ve 1999'da ECMAScript 3 (ES3) yayınlandı. ECMAScript 4'ün (ES4) 2008'de yayınlanmasına kadar web alanında birçok gelişme yaşandı. Örneğin Firefox web tarayıcısı ve JSON veri tipinin yayınlanması önemli gelişmeler arasında gösterilebilir. 2009'da ECMAScript 5 (ES5) yayınlandıktan sonra yeniden dilin gelişmesi süreci yavaşlamıştır. Ancak 2015 yılında ECMAScript 6'nın (ES6) yayınlanmasıyla JavaScript, gelmiş geçmiş en fazla yeniliğin olduğu ECMAScript versiyonu olmuştur. TC-39 grubu versiyon isimlendirme konusunda artan rakamlar yerine yayınlandığı yılı versiyon ismi olarak kullanmaya başlamıştır. Yani dilin versiyonu ECMAScript 2015 olarak değişmiştir. Dilin modernleşmesi adına en büyük yenilikler bu versiyonda yaşanmıştır. 2015'ten itibaren tarayıcıların büyük güncellemeler sonrasında adaptasyon sorunu yaşamaması için her yıl ve ufak yenilikler yapılması kararı alınmıştır. Artık her yıl düzenli olarak JavaScript dilinin yeni versiyonu yayınlanmaktadır. 2018 itibariyle en güncel versiyon ECMAScript 2018'dir. Her geçen yıl ufak yeniliklerle JavaScript gelişmeye devam ediyor. 2018 itibariyle modern tarayıcılar tüm ECMAScript özelliklerini desteklemektedir.

1.2.1. Babel

ECMAScript 2015'in yayınlanmasından sonra gelen yenilikler çok büyük ve fazla olmasından dolayı tarayıcılarda birçok kırılım yaşandı. Bu kırılımların önüne geçebilmek için CommonJS adında ES5 versiyonun özelliklerini kapsayan bir ECMAScript türevi oluşturulmuştur. ES2015 özelliklerini eski tarayıcılarda kullanmak isteyen bir grup geliştirici Babel projesini başlattı. Proje amacı yeni nesil JavaScript kodlarını CommonJS yani ES5 uyumlu hale getiren bir çeviri aracıdır. Babel yeni nesil JavaScript'i bugünden tarayıcılarda kullanmamızı sağlamaktadır.

1.2.2. NodeJS

JavaScript ortaya çıktığı günden itibaren sadece tarayıcılar içerisinde çalışan bir dil iken 2009 yılında Ryan Dahl tarafından dilin tarayıcı bağımlılığını ortadan kaldırılıp sunucu tarafında da çalışabilmesi sağlanmıştır.

1.3. Kütüphane

Tez kapsamında hazırlanan projede parametrik olmayan testlerin hem tarayıcılarda hem de NodeJS yani sunucu tarafında çalışabilen bir kütüphane olması hedeflenmiştir. Kod yazım stili olarak modern JavaScript özellikleri kullanılmıştır. Her iki platformda kullanılabilmesi için Babel ile CommonJS dönüşümü gerçekleşmiştir.

2. UYUM İYİLİĞİ TESTLERİ

Parametrik testlerin varsayımlarından biri örneklerin seçildiği yığınların dağılımlarının biçiminin normal olduğudur. Parametrik testleri kullanırken yığınların dağılımları ile ilgili bu varsayımın sağlandığı konusunda emin olmak gerekir. Bu nedenle parametrik testler yapılırken yığınlarla ilgili normallik varsayımı konusunda şüphe varsa, bu varsayımın sağlanıp sağlanmadığını ortaya çıkarmak için bir test yapılmalıdır. Bu testlerin amacı örnek verisinin öngörülen dağılıma uyup uymadığına karar vermektir. Bu tip testlere uyum iyiliği testleri denir.

2.1. Ki-Kare Uyum İyiliği Testi

Bu testin temel özelliği, gözlenen frekanslar ile yokluk hipotezi doğru iken beklenen frekanslar arasındaki farkların büyüklüklerine dayanmaktadır.

Tablo 1. Ki-Kare Gözlenen Frekans Tablosu

Sınıflar	**1**	**2**	**3**	**…**	**j**	**…**	**c**	**Toplam**
Gözlenen frekanslar	G_1	G_2	G_3	…	G_j	…	G_c	n

c, sınıf sayısı olmak üzere, ki-kare uyum iyiliği testinde yokluk ve karşıt hipotezler genel olarak aşağıdaki gibidir.

H_0 : Örnek belirli bir dağılıma sahip yığından seçilmiştir.
H_1 : Örnek yokluk hipotezinde belirtilen dağılımdan seçilmiştir.

Yokluk hipotezi doğru ise yığından seçilen örneğin yığının karakteristiklerini taşıması beklenir. Örnek söz edilen dağılımdan seçilmiş ise sınıfların her birinde beklenen frekanslar ile gözlenen frekansların birbirine eşit ya da yakın olması beklenir. Test istatistiği,

$$\sum_{j=1}^{c} \frac{\left(G_j - B_j\right)^2}{B_j} \sim \chi^2 \qquad (1)$$

Olarak tanımlanır.

Yokluk hipotezi doğru ve öngörülen dağılım için parametre tahmini yapılmadıysa yukarıda tanımlanan istatistik yaklaşık olarak $c-1$ serbestlik dereceli ki-kare dağılımına sahiptir. Test işleminde karar şu şekilde verilir. χ^2_{c-1} istatistiği için hesaplanan değer χ^2_h ve α anlamlılık düzeyinde tablo değeri $\chi^2_{c-1,\alpha}$ olmak üzere aşağıdaki gibi tanımlanır.

$\chi^2_h > \chi^2_{c-1,\alpha}\ ise\ H_0\ reddedilir.$
$\chi^2_h \leq \chi^2_{c-1,\alpha}\ ise\ H_0\ reddedilemez.$

2.2. Kolmogorov-Smirnov Uyum İyiliği Testi

Kolmogorov-Smirnov uyum iyiliği testi oranlama ya da eşit aralıklı düzeyde ölçülen değişkenler için kullanılır. Birikimli dağılım fonksiyonu $F(x)$, X rastgele değişkeninin değerinin x değerine eşit ya da daha küçük olması olasılığıdır. Bu matematiksel olarak aşağıdaki biçimde ifade edilebilir.

$$F(x) = P(X \leq x) \tag{2}$$

$F_0(x)$ hipotezde belirtilen dağılımın birikimli dağılım fonksiyonu iken, yokluk hipotezi ve karşıt hipotezler aşağıdaki gibidir.

$$H_0 : F(x) = F_0(x),\ tüm\ x\ değerleri\ için$$
$$H_1 :\ F(x) \neq F_0(x),\ en\ az\ bir\ x\ değeri\ için$$

Gözlenen birikimli dağılım fonksiyonu $S(x)$ olmak üzere: $S(x), x$ değerine eşit ya da daha küçük değerli örnek birimlerinin sayısının örnek hacmine oranıdır.

$$D = \sup_{x} |S(x) - F_0(x)| \tag{3}$$

D istatistiğinin örnekleme dağılımından elde edilen kritik değerler tablosu Kolmogorov (1933) tarafından yayınlanmıştır. Bu tablodan bulunan değerler D_k olsun. D istatistiği için örnekten hesaplanan değer D_h olmak üzere, karar kuralı aşağıdaki gibidir.

$$D_h \geq D_k\ ise\ H_0\ reddedilir.$$
$$D_h < D_k\ ise\ H_0\ reddedilemez.$$

$F_0(x)$ sürekli iken $S(x)$ kesiklidir. Bu nedenle seçilen bir örnek için X'in değer almadığı herhangi bir aralığın iki ucunda $|S(x) - F_0(x)|$ için iki farklı sonuç vardır.

$$D\ =\ \sup\{|S(x_j) - F_0(x_j)|, |S(x_{j-1}) - F_0(x_j)|\} \tag{4}$$

2.3. Normallik İçin Lilliefors Testi

Lilliefors testi temelde Kolmogorov-Smirnov testine dayanmaktadır. Normal dağılıma uygunluk testinde öngörülen normal dağılımın parametreleri parametreleri bilinmiyorsa ve bunlar örnek verisinden tahmin ediliyorsa Lilliefors testinin kullanımı önerilmektedir. Bu test için hipotezler aşağıdaki gibi ifade edilir.

H_0 : n hacimli örnek ortalaması ve varyansı bilinmeyen normal dağılımdan gelmiştir.
H_1 : n hacimli örnek bir normal dağılımdan gelmemiştir.

n hacimli örnek $X_1, X_2, \dots, X_n$ olsun. H_0 hipotezinde öngörülen normal dağılımın bilinmeyen parametreleri μ ve σ^2 parametrelerinin tahmin edicileri, yani $\overline{X}$ ve S^2 kullanılır. Buna göre dağılımın normalliği bilgisi ile,

$$F_0(x) = P(X < x) = P\left(Z < \frac{Z-\overline{X}}{S}\right) \tag{5}$$

fonksiyonu elde edilir. Lilliefors testinde $S(x)$ fonksiyonu Z değerlerine dayanır. Bu nedenle,

$$S(x) = \frac{z \text{ değerine eşit ya da daha küçük değerli örnek birimlerin sayısı}}{n}$$

$F_0(x)$ fonksiyonu sürekli ve $S(x)$ fonksiyonu da kesikli olduğundan normallik için Lilliefors istatistiği aşağıdaki biçimde elde edilmiştir.

$$D_h = \sup\{|S(x_j) - F_0(x_j)|, |S(x_{j-1}) - F_0(x_j)|\} \tag{6}$$

2.4. Normallik İçin Shapiro-Wilk Testi

Shapiro-Wilk normallik testi örnek verisinin, parametreleri bilinmeyen bir normal dağılımdan geldiği durumlarda kullanılır. Bu test için hipotezler,

H_0: $F(x)$, ortalaması ve varyansı bilinmeyen bir normal dağılım fonksiyonudur.
H_1: $F(x)$, ortalaması ve varyansı bilinmeyen bir normal dağılım fonksiyonu değildir.

n hacimli rastgele bir örnek $X_1, X_2, \dots, X_n$ olsun ve bu rastgele örnek için sıralı istatistikler de $X_{(1)}, X_{(2)}, \dots, X_{(n)}$ olsun. Ölçüm değerleri ile aritmetik ortalamanın farklarının kareleri toplamı,

$$D = \sum_{i=1}^{n} \left(\mathrm{X_i} - \overline{X}\right)^2 \tag{7}$$

ile ifade edilsin.

$k = \frac{n}{2}$ olmak üzere $a_1, a_2, \dots, a_n$ sabitleri S. S. Shapiro ve M. Wilk (1965) tarafından yayınlanmış olan tablodan bulunur ve W ile gösterilen Shapiro-Wilk istatistiği Eşitlik (8)'deki gibi tanımlanır.

$$W = \frac{1}{D}\left[\sum_{i=1}^{k} a_i\left(X_{(n-i+1)} - X_{(i)}\right)\right]^2 \tag{8}$$

Normallik için Shapiro-Wilk testinde, W istatistiği için hesaplanan değer W_h olmak üzere, $P(W < W_h)$ değerine göre karar verilir.

$$P(W < W_h) < \alpha \tag{9}$$

3. VARYANSLARIN HOMOJENLİĞİ İÇİN TESTLER

Varyansların homojenliği varsayımı veri analizinde uygun test istatistiğinin belirlenmesi için önemlidir. Örneğin varyans analizi problemlerinde F testinin (Anova) kullanımı varjans homojenliği varsayımına dayanır. Varyans homojenliği için yapılan testlerde hipotezler,

$H_0: \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2 = \sigma^2$

$H_1: \sigma^2$'lerin en az biri farklıdır

biçiminde ifade edilir.

H_0 hipotezinin testi için k sayıda yığından birer rastgele örnek seçilsin. j. yığından seçilen örneğin hacmi $n_j, j = 1, 2, \ldots, k$ ve $i = 1, 2, \ldots, n_j$ olmak üzere bu örnekler X_{ij} ile gösterilsin. k sayıda örnek için gösterim Tablo 2'de verilmiştir.

Tablo 2. Verinin Örneklere Göre Dağılımı

Örnek No			
1	2	…	k
X_{11}	X_{12}	…	X_{1k}
X_{21}	X_{22}	…	X_{2k}
…	…	…	…
$X_{n_1 1}$	$X_{n_2 2}$	…	$X_{n_k k}$

3.1. Levene Testi

Bu test, j. örnekteki i. Birimin değeri X_{ij} ve j. örneğin ortalaması da $\overline{X}_j$ olmak üzere,

$$Z_{ij} = \left|X_{ij} - \overline{X}_j\right|, \quad \overline{X}_j = \frac{1}{n_j}\sum_{i=1}^{n_j} X_{ij} \tag{10}$$

olarak tanımlanan gözlem değerlerinden ortalamanın sapmalarının mutlak değerleri ile tek faktörlü varyans analizi yönteminin kullanılmasına dayanır. Buna göre, j. örnekteki mutlak sapmaların ortalaması

$$\overline{Z}_j = \frac{1}{n_j}\sum_{i=1}^{n_j} Z_{ij} \tag{11}$$

ve mutlak sapmaların genel ortalaması da,

$$\overline{Z} = \frac{1}{n_j}\sum_{j=1}^{k} \sum_{i=1}^{n_j} Z_{ij} \tag{12}$$

olmak üzere, gruplar arası kareler toplamı (GAKT) ve gruplar içi kareler toplamı (GIKT) sırasıyla aşağıdaki şekilde hesaplanır;

$$GAKT = \sum_{j=1}^{k} n_j\left(\overline{Z}_j - \overline{Z}\right)^2, \quad GIKT = \sum_{j=1}^{k} \sum_{i=1}^{n_j}\left(Z_{ij} - \overline{Z}_j\right)^2 \tag{13}$$

Gruplar arası ortalama kare (GAOK) ve gruplar içi ortalama kare (GIKT),

$$GAOK = \frac{GAKT}{k-1}, \quad GIOK = \frac{GIKT}{n-k} \tag{14}$$

olmak üzere, varyansların homojenliği için Levene testi istatistiği aşağıdaki gibidir.

$$F_{k-1,n-k} = \frac{GAOK}{GIOK} \tag{15}$$

3.2. Brown-Forsythe Testi

Varyansların homojenliği için Brown-Forsythe testi Levene testine benzer. Levene testi ölçüm değerleri aritmetik ortalama arasındaki mutlak farkları kullanılırken Brown-Forsythe testi ölçüm değeri ile örnek meydanı (ortancası) arasındaki mutlak farklara dayanır.

j. örnekteki i. Birimin ölçüm değeri X_{ij} ve j. örneğin meydanı da m_j olsun.

$$m_j = Medyan\left(X_{1j}, X_{2j}, \dots, X_{nj}\right) \quad j = 1,2, \dots, k \tag{16}$$

Bilindiği gibi medyan için aşağıdaki tanımlar da yapılabilir.

1) n_j tek $X_{1j}, X_{2j}, \dots, X_{n_j j}$ değerleri küçükten büyüğe doğru sıralandığında ortadaki değer medyandır.
2) n_j çift $X_{1j}, X_{2j}, \dots, X_{n_j j}$ değerleri küçükten büyüğe doğru sıralandığında ortadaki iki değerin ortalaması medyandır.

Buna göre,

$$Z_{ij} = \left|X_{ij} - m_j\right| \tag{17}$$

olarak tanımlansın. Brown-Forysthe testi eşitlik (19)'daki gibi tanımlanan Z_{ij} değerleri tek faktörlü varyans analizi yönteminin kullanımına dayanır. Levene testinde verilen GAKT, GIKT, GAOK ve GIOK formülleri bu test için de geçerlidir. Brown-Forsythe test istatistiği aşağıdaki gibi tanımlanır.

$$BF = \frac{GAOK}{GIOK} \tag{18}$$

BF test istatistiği $k-1$ ve $n-k$ serbestlik dereceli F dağılımına sahiptir ve $BF \sim F_{k-1,n-k}$ biçiminde ifade edilir.

Bu sonuca göre,

$F_h > F_{k-1,n-k,\alpha}\ ise\ H_0\ reddedilir.$

$F_h \leq F_{k-1,n-k,\alpha}\ ise\ H_0\ reddedilemez.$

3.3. Layard Ki-Kare Testi

Varyansların homojenliği için Layard ki-kare testi, Levene testi ve Brown-Forsythe testine göre farklı yapıdadır.

j. örnekteki i. birim değerleri X_{ij}, j. örnek hacmi n_j, j. örnek ortalaması $\overline{X}_j$ ve $n = \sum n_j$ olsun. Buna göre,

$$\hat{\gamma} = \frac{n\sum_{j=1}^{k}\ \sum_{i=1}^{n_j}\left(X_{ij}-X_j\right)^4}{\left(\sum_{j=1}^{k}\ \sum_{i=1}^{n_j}\left(X_{ij}-\overline{X}_j\right)^2\right)^2} - 3 \tag{19}$$

ve

$$\hat{T}^2 = 2 + \left(1 - \left(\frac{k}{n}\right)\right)\hat{\gamma} \tag{20}$$

olmak üzere,

$$L = \frac{\sum_{j=1}^{k}(n_j-1)\left[logS_j^2 - \frac{1}{\sum_{j=1}^{k}(n_j-1)}\sum_{j=1}^{k}(n_j-1)logS_j^2\right]^2}{\hat{T}^2} \tag{21}$$

olarak tanımlanır. L istatistiği $k-1$ serbestlik dereceli ki-kare dağılımına sahiptir.

$$L \sim \chi^2_{k-1} \tag{22}$$

Bu nedenle istatistiğin hesaplanan değeri χ^2_h olmak üzere,

$\chi^2_h > \chi^2_{k-1,\alpha}\ ise\ H_0\ reddedilir.$
$\chi^2_h \leq \chi^2_{k-1,\alpha}\ ise\ H_0\ reddedilemez.$

4. BİR ÖRNEK İÇİN TESTLER

Bir örnek için testler, örneklerinin belirli bir yığından çekilip çekilmediğinin ya da başka bir deyimle belirli bir yığına ait olup olmadığının incelenmesi amacı ile kullanılır. Yığına ilişkin aritmetik ortalama ile ilgili hipotezler için yığın varyansının bilinip bilinmemesine göre parametrik testlerden z testi ya da t testi kullanılabilir. Ancak bu iki testin kullanılabilmesi için örnek birimlerinin seçildiği yığının ilgili değişken bakımından dağılımının normal olması gerekir. Bu varsayım sağlanmadığında, özellikle örnek hacmi küçükse, parametrik testleri kullanmak doğru olmaz. Böyle durumlar için uygun analiz yöntemi parametrik olmayan testlerdir.

4.1. Binom Testi

Bazı araştırmalarda bir karakteristik bakımından yığına ilişkin oran ilgi konusu olabilir. Örneğin bir hekim uygulandığı tedaviye olumlu tepki veren hastaların oranının 0.80'den büyük olup olmadığını, bir işyeri müşterilerinde kadınların oranının 0.50'den farklı olup olmadığını bilmek isteyebilir. Bu tür problemlerde yığın ya soyut özellikte ya da çok sayıda birim içerdiğinden ilgi duyulan bu parametreyi (oran) hesaplamak yerine yığından seçilen n hacimli bir örnekteki bilgi kullanılarak bu parametreye ilişkin bir hipotez testi ile karar verilmeye çalışılır. Bu amaçla kullanılan parametrik olmayan test binom testi olarak bilinir.

Binom testinde yığına ilişkin bilinmeyen oran π ve 0 ile 1 arasındaki herhangi bir sayı da π_0 olmak üzere, hipotezler;

$H_0: \pi = \pi_0$
$H_1: \pi \neq \pi_0$

biçiminde oluşturulur. Test istatistiği b olmak üzere, δ_i değişkeni aşağıdaki gibi tanımlanır.

$$b = \sum_{i=1}^{n} \delta_i \sim Binom \tag{23}$$

$$\delta_i = \begin{cases} 1, & i.\text{ölçüm değeri ilgilenilen özellikte ise} \\ 0, & i.\text{ölçüm değeri ilgilenilen özellikte değilse} \end{cases} \tag{24}$$

4.2. Bir Örnek İşaret Testi

1710 yılında John Artbuthnot tarafından önerilen bir örnek işaret testi parametrik olmayan testler içerisinde en eskisidir. Ayrıca bu test binom testinin özel bir hali olarak da düşünülebilir.

İşaret testinde, M yığınına ilişkin bilinmeyen medyan ve M_0 herhangi bir reel sayı olmak üzere, hipotezler;

$H_0: M = M_0$
$H_1: M \neq M_0$

biçiminde oluşturulur. Test istatistiği k olmak üzere,

$$k = \sum_{i=1}^{n} \delta_i \tag{25}$$

biçiminde elde edilir. Burada δ_i değişkeni aşağıdaki gibi tanımlanır. İlgili yığından rastgele seçilen n hacimli örnek $X_i : X_1, X_2, \dots, X_n$ olsun.

$$D_i = X_i - M_0, \quad i = 1,2, \dots, n \tag{26}$$

$$\delta_i = \begin{cases} 1, & D_i > 0 \\ 0, & D_i < 0 \end{cases} \tag{27}$$

4.3. Wilcoxon İşaretli Sıra Sayıları Testi

İşaret testi gözlem değerleri ile hipotezde belirtilen medyan değeri arasındaki farkların sadece işaretlerini kullanır. Wilcoxon işaretli sıra sayıları testi ise yokluk hipotezinin test edilmesinde farkların büyüklüklerini de hesaba katan parametrik olmayan bir testtir. Wilcoxon işaretli sıra sayıları testi D_i farklarının büyüklüklerini de hesaba kattığı için işaret testine göre daha çok bilgi kullanır ve bu nedenle işaret testine göre daha güçlüdür.

Wilcoxon işaretli sıra sayıları testinde, M yığına ilişkin bilinmeyen medyan ve M_0 herhangi bir reel sayı olmak üzere, hipotezler;

$H_0: M = M_0$
$H_1: M \neq M_0$

biçiminde oluşturulur. Test istatistiği T^+ olmak üzere,

$$T^+ = \sum_{i=1}^{n} r(|D_i|)\delta_i \tag{28}$$

biçiminde elde edilir. Ve D_i ile δ_i istatistikleri aşağıdaki biçimde elde edilir.

$$D_i = X_i - M_0, \quad i = 1,2,\dots,n \tag{29}$$

$$\delta_i = \begin{cases} 1, & D_i > 0 \\ 0, & D_i < 0 \end{cases} \tag{30}$$

$|D_i|$ için sayı-sıralı istatistik de $r(|D_i|)$ olsun.

$$r(|D_i|) = \sum_{i=1}^{n} S\left[|D_i| - |D_j|\right] \tag{31}$$

$$S(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \tag{32}$$

biçiminde tanımlanır.

4.4. Rastgelelik için Dizi Parçaları (Run) Testi

Rastgele istatistikte önemli bir kavramdır ve veri analizinde sıkça karşılaşılır. Örneğin istatistiksel testlerin çoğu ilgili yığından n hacimli rastgele örnek seçilmesine dayanır. Benzer biçimde istatistiksel kalite kontrolü süreci üretim sürecinden sonra n hacimli ürünün rastgele seçildiği varsayımına dayanır. Regresyonda da hata terimlerinin 0 ortalamalı ve σ^2 varyanslı bir rastgele bir değişken olduğu varsayılır.

Bu çalışmada dizi parçaları testi n hacimli örnekten derlenen ölçüm (gözlem) değerlerinin iki tür simge ile gösterildiği durum için açıklanacaktır. Bu test için ölçüm değerlerinin önce iki tür simgeden oluşan bir dizi olarak gösterilmesi gerekir. Bir kuyrukta yer alan 8 bireyin cinsiyetlerine göre sıralanışı bu duruma uygun bir örnektir ve Tablo 3'te yer almaktadır.

Tablo 3. Bir Kuyrukta Yer Alan Bireylerin Cinsiyet ve Boylarına Göre Sıralanışı

Birey No	1	2	3	4	5	6	7	8
Cinsiyet	K	K	E	E	K	E	E	K
$\boldsymbol{X_i}$	1.70	1.75	1.80	1.82	1.69	1.90	1.74	1.65

Aritmetik ortalamadan sapmaların rastgelelik gösterip göstermediği araştırılmak istenirse önce bu veridek89i ölçüm değerlerinin aritmetik ortalamadan sapmalarına dayalı simgeler oluşturulur. Bu veriler için $\overline{X} = 1.756$'dır.

Aritmetik ortalamanın altında olanlar A, Aritmetik ortalamanın üstünde olanlar B ile ifade edildiğinde bireylere ilişkin simgeler Tablo 4'te verildiği gibidir.

Tablo 4. Run Testi için Sembolize Edilmiş Değerler

Birey No	1	2	3	4	5	6	7	8
Simge	A	A	Ü	Ü	A	Ü	A	A

Rastgelelik için Run testi n birimlik örnekten derlenen verinin ortalamadan, medyandan ya da herhangi bir değerden sapmalarının rastgele olup olmadığının kontrol edilmesinde kullanılır.

5. İKİ BAĞIMSIZ ÖRNEK İÇİN TESTLER

İki bağımsız örnek için testler, iki örnek ortalamasının eşitliği hipotezini test etmek amacıyla kullanılır. Bu amaç doğrultusunda parametrik testlerden t testi kullanılabilir. Ancak bu testin kullanımı örneklerin seçildikleri yığınların dağılım biçimlerinin normal olduğu varsayımına dayanır. Eğer iki bağımsız örnekten biri bile normal dağılımdan gelmediyse ve küçük hacimli örnekler kullanılıyorsa, bu parametrik testin kullanımı doğru olmaz. Böyle durumlarda parametrik olmayan testler tercih edilmelidir.

5.1. KONUM İLE İLGİLİ TESTLER

5.1.1. Medyan Testi

İki bağımsız örnek için parametrik olmayan testlerden biri medyan testidir. Bu test örneklerin geldikleri yığınların konum parametrelerinin eşitliği hipotezini test etmek amacıyla kullanılır.

$H_0: M_1 = M_2$
$H_1: M_1 \neq M_2$

Tablo 5. Medyan Testi Frekans Tablosu

	Örnek No		Toplam
	1	2	
Birleştirilmiş örnek medyanından büyük değerli olanların sayısı	A	B	A + B
Birleştirilmiş örnek medyanından küçük değerli olanların sayısı	C	D	C + D
Toplam	A + C	B + D	n

Yokluk hipotezini test etmek için A ve B istatistiklerinin örnekleme dağılımının bilinmesi gerekir. A ve B istatistiklerinin ortak olasılık fonksiyonu bilinen hipergeometrik olasılık fonksiyonudur.

$$f(b,a) = \frac{\binom{n_1}{a}\binom{n_2}{b}}{\binom{n_1+n_2}{a+b}} \tag{33}$$

Test işlemini sonuçlandırmak hipergeometrik olasılık fonksiyonu ile olasılıkların hesaplanması gerekir. Bu işlemleri sonuçlandırmak örnek hacimleri çok küçük olmadıkça, son derece sıkıcıdır. Örnek hacimleri yeterince büyükse alternatif bir yol kullanılabilir. Yığındaki birim sayısı örnek hacminin en az 10 katı ise hipergeometrik dağılım Binom dağılımına yeterince yaklaşır. Bu durumda her örnek için birer Binom dağılım tanımlanabilir. Binom dağılımının yeterince büyük örnek hacimleri için normal dağılıma yaklaşımı da kullanılırsa test istatistiği elde edilebilir.

$$P_1 = \frac{A}{n_1}, \qquad P_2 = \frac{B}{n_2}, \qquad P = \frac{A+B}{n} \tag{34}$$

olmak üzere, aşağıdaki gibi tanımlanan istatistik

$$Z = \frac{P_1 - P_2}{\sqrt{P(1-P)\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}} \tag{35}$$

standart normal dağılıma yaklaşır. Bu nedenle büyük örnek hacimleri durumunda yokluk hipotizinin testinde standart normal dağılımdan yararlanılır.

5.1.2. Mann-Whitney Testi

İki bağımsız örnek için örneklerin geldikleri yığınların konum parametrelerinin (medyan) eşitliği hipotezini test etmek amacıyla kullanılır. Bu amaçla kullanılan Mann-Whitney testinin temel özelliği gözlem değerlerine atanan büyüklük sıra sayılarına dayanmasıdır.

Örneklerin seçildikleri yığınların bilinmeyen medyanları sırasıyla M_1 ve M_2 olmak üzere, Mann-Whitney testinde hipotezler aşağıdaki biçimde tanımlanır;

$H_0: M_1 = M_2$
$H_1: M_1 \neq M_2$

Mann-Whitney test istatistiği elde edilirken önce n_1 ve n_2 hacimli iki bağımsız örnek birleştirilir ve $n_1 + n_2 = n$ hacimli birleştirilmiş örnek oluşturulur. Sonra bu birleştirilmiş örnekteki birimlere küçükten büyüğe doğru sıra sayıları atanır. Bu sıra sayıları atama işleminde aynı değerli olanlar varsa bunlara ortalama sıra sayısı atanır. X gözlemlerine atanan sıra sayılarının toplamı S olmak üzere, Mann-Whitney test istatistiği,

$$T = S - \frac{n_1(n_1+1)}{1} \tag{36}$$

biçiminde tanımlanır. Test istatistiğinin alabileceği değerler $T = 0,1, \dots, n_1, n_2$ aralığındadır.

5.1.3. Wald-Wolfowitz Dizi Parçaları Testi

Bu test iki bağımsız örnekten derlenen verinin Rastgelelik için Dizi Parçaları (Run) testinde kullanılabilecek biçimde düzenlenmesine dayanır. Run testinde olduğu gibi Wald-Wolfowitz testinde de ölçüm değerleri simgelerle temsil edilir. Wald-Wolfowitz dizi parçaları testinde hipotezler aşağıdaki gibi tanımlanır.

H_0 : X ve Y gözlemleri benzer dağılımlı yığınlardan gelmiştir.
H_1 : X ve Y gözlemleri benzer dağılımlı yığınlardan gelmemiştir.

5.1.4. Fisher Tam Olasılık Testi

İki bağımsız örnekten derlenen verideki her bir gözlem değerinin iki (ayrık) sınıftan birine sınıflandırılması problemi ile uygulamada sıkça karşılaştırılır. Örneğin, A ve B ilaçlarının uygulandığı hastaların tedaviye olumlu tepki verenler ve vermeyenler olarak sınıflandırılmaları gibi. Bu tür sınıflandırma ile 2x2 boyutlu bir frekans tablosu elde edilir. Fisher tam olasılık testinde hipotezler, yığınları için ilgilenilen özellik bakımından oranlar sırasıyla π_1 ve π_2 olmak üzere Tablo 6'daki gibi ifade edilir.

$H_0: \pi_1 = \pi_2$

$H_1: \pi_1 \neq \pi_2$

Tablo 6. Fisher Tam Olasılık Testi Frekans Tablosu

Örnek No	İlgilenilen özellikte olanlar	İlgilenilen özellikte olmayanlar	Toplam
1	a	$n_1 - a$	n_1
2	b	$n_2 - b$	n_2
Toplam	$a + b$	$n - (a + b)$	n

Bu testte "ikinci örnekte ilgilenilen özellikte olanlar sınıfına düşen birim sayısı" test istatistiği olarak kullanılır. Yukarıda verilen tabloya göre test istatistiği b ile gösterilir ve bu test istatistiği hipergeometrik dağılıma sahiptir.

$$f(b, a) = \frac{\binom{n_1}{a}\binom{n_2}{b}}{\binom{n_1+n_2}{a+b}} \tag{37}$$

Bu testteki temel amaç iki ayrık sınıfa düşen birimlerin oranı bakımından iki grubun farklı olup olmadığının belirlenmesidir. Fisher testi bu amaçla kullanılan testlerden biridir.

5.1.5. Kolmogorov-Smirnov Testi

Kolmogorov-Smirnov testinde amaç n_1 ve n_2 hacimli iki örneğin aynı yığından mı yoksa farklı yığınlardan mı geldiğine karar vermektir. Test için hipotezler;

$H_0 : F_1(x) = F_2(x)$, tüm x değerleri için
$H_1 : F_1(x) \neq F_2(x)$, en az bir x değeri için

biçiminde tanımlanır. Test istatistiği D'yi elde etmek için,

$S_1(x)$: 1. örneğe ilişkin gözlenen dağılım fonksiyonu
$S_2(x)$: 1. örneğe ilişkin gözlenen dağılım fonksiyonu

olmak üzere, iki yanlı test için test istatistiği Eşitlik (38)'deki gibi tanımlanır.

$$\mathrm{D} = \text{En büyük}|\mathrm{S}_1(\mathrm{x}) - \mathrm{S}_2(\mathrm{x})| \tag{38}$$

5.2. DAĞILIŞ İLE İLGİLİ TESTLER

5.2.1. Mood Testi

Bazen dağılış (saçılım, yayılım) ölçüsü olan iki parametrenin eşit olup olmadığının bilinmesine ihtiyaç duyulur. İki ortalama farkına ilişkin test yaparken yığın varyansları bilinmiyorsa, bu bilgiye özellikle küçük örnek hacimlerinde ihtiyaç vardır. Çünkü t dağılımının serbestlik derecesinin belirlenebilmesi bu bilgiyi gerektirir. İki yığının dağılış parametrelerinin eşitliğinin testi için kullanılan parametrik olmayan testlerden biri Mood testidir. Mood testinin kullanılabilmesi için yığınların medyanlarının aynı olması gerekir.

Mood testi için hipotezler;

$H_0 : \sigma_1 = \sigma_2$
$H_1 : \sigma_1 \neq \sigma_2$

biçiminde kurulur. Test istatistiği M olmak üzere,

$$M = \sum_{i=1}^{n_1} \left(r_i - \frac{n+1}{2} \right)^2 \quad (39)$$

Burada n birleştirilmiş örnek hacmi $n = (n_1 + n_2)$ ve r_i, birleştirilmiş örnekteki ölçüm değerlerine atanan sıra sayılarındaki X gözlemlerine karşı gelenler şeklinde tanımlanır.

5.2.2. Siegel-Tukey Testi

İki dağılış parametresinin eşitliği hipotezi için kullanılabilecek testlerden bir diğeri de Siegel-Tukey testidir. Varsayımları Mood testi ile aynıdır. Siegel-Tukey testi için hipotezler;

$H_0: \sigma_1 = \sigma_2$
$H_1: \sigma_1 \neq \sigma_2$

biçiminde tanımlanır. Test istatistiği ST olmak üzere,

$$ST = \sum_{i=1}^{n} a_i \delta_i \quad (40)$$

olarak tanımlanır. Burada $n = (n_1 + n_2)$ olmak üzere,

$$\delta_i = \begin{cases} 1, & i.sıradaki\ değer\ X\ gözlemi\ ise \\ 0, & i.sıradaki\ değer\ Y\ gözlemi\ ise \end{cases} \quad (41)$$

$$a_i = \begin{cases} 2i, & i.çift\ ve\ 1 < i \leq \left(\frac{n}{2}\right)\ ise \\ 2i-1, & i.tek\ ve\ 1 \leq i \leq \left(\frac{n}{2}\right)\ ise \\ 2(n-i)+2, & i.çift\ ve\ \left(\frac{n}{2}\right) < i \leq n\ ise \\ 2(n-i)+1, & i.tek\ ve\ \left(\frac{n}{2}\right) < i < n\ ise \end{cases} \quad (42)$$

şeklinde elde edilir.

6. İKİ BAĞIMLI ÖRNEK İÇİN TESTLER

İki bağımlı örneğe ilişkin veri genel olarak iki şekilde elde edilir. Bunlardan birincisi "işlem öncesi – işlem sonrası" tasarımıdır. Bu tasarımda önce örneğe seçilen birimlerin ilgili bağımlı değişken bakımında aldıkları değerler ölçülür. Daha sonra bu birimlere ilgili işlem uygulanır ve işlem uygulaması bittikten sonra aynı bağımlı değişken bakımından bu birimlerin işlemden sonraki aldıkları değerler tekrar ölçülür. Örneğin rastgele seçilen 10 kadının ağırlıklarının ölçülmüş olduğu varsayılan. Bu kadınlara bir zayıflama rejimi uygulanmış olsun. Aynı kadınların zayıflama rejiminden 20 gün sonraki ağırlıkları da saptanabilir. Bu 10 kadının zayıflama rejiminden önceki ve sonraki ağırlıkları işlem öncesi – işlem sonrası tasarımı için bir örnektir.

6.1. İşaret Testi

İki bağımsız örnek için işaret testi, bir örnek işaret testine dayanır. Test hipotezleri aşağıdaki gibi tanımlanır.

H_0 : D_i farklı yığınının medyanı sıfırdır
H_1 : D_i farklı yığınının medyanı sıfırdan farklıdır

İki bağımlı örnek için işaret testi X ve Y değişkenlerinden yeni bir değişken tanımlamayı gerektirir. Bu yeni değişken D olsun. D değişkeni Eşitlik (43)'deki gibidir.

$$D_i = X_i - Y_i,\ i = 1,2,\dots,n \tag{43}$$

İşaret testinde olduğu gibi işaret testi istatistiği aşağıdaki gibi tanımlanır.

$$\delta_i = \begin{cases} 1, & D_i > 0 \\ 0, & D_i < 0 \end{cases} \tag{44}$$

$$k = \sum_{i=1}^{n} \delta_i \tag{45}$$

6.2. Wilcoxon İşaretli Sıra Sayıları Testi

İki bağımlı örnek problemlerinde bağımlı değişken için ölçme düzeyi eşit aralıklı ya da oranlama ise işaret testinin kullanımı bilgi kaybına neden olur. Böyle durumlarda daha güçlü olan Wilcoxon işaretli sıra sayıları testi seçilmelidir.

Test hipotezleri aşağıdaki gibi tanımlanır.

$H_0 : D_i$ farklı yığınının medyanı sıfırdır

$H_1 : D_i$ farklı yığınının medyanı sıfırdan farklıdır

Test istatistiği T^+ olmak üzere,

$$T^+ = \sum_{i=1}^{n} r(|D_i|)\delta_i \tag{46}$$

biçiminde elde edilir. Burada test istatistiği aşağıdaki biçimde elde edilir.

$$D_i = X_i - Y_i,\ i = 1,2,\dots,n \tag{47}$$

$$\delta_i = \begin{cases} 1, & D_i > 0 \\ 0, & D_i < 0 \end{cases} \tag{48}$$

6.3. McNemar Testi

Bu test iki bağımlı örnek problemlerinde ölçüm (işlem) sonucuna göre eşlerin kategorilere (sınıflara ayrıldığı) durumlarda kullanılır.

Tablo 7. McNemar Testi Frekans Tablosu

		İşlem Grubu		
		Negatif	**Pozitif**	**Toplam**
Kontrol Grubu	**Negatif**	A	C	A + C
	Pozitif	B	D	B + D
	Toplam	A + B	C + D	n

Bir işlemin (koşul) altında ilgilenilen özellikte olanların oranı π_1 ve diğer işlem altında ilgilenilen özellikte olanların oranı da π_2 olmak üzere, McNemar testinde hipotezler,

$H_0: \pi_1 = \pi_2$
$H_1: \pi_1 \neq \pi_2$

biçiminde ifade edilir. İlgilenilen özellik bakımından örnek oranlarını P_1 ve P_2 ile gösterelim. Buradan,

$$P_1 = \frac{A+C}{n} \qquad \text{ve} \qquad P_2 = \frac{A+B}{n} \tag{49}$$

olarak belirlenir. Bu örnek oranları arasındaki fark,

$$P_1 - P_2 = \frac{C-B}{n} \tag{50}$$

olur. Yokluk hipotezi doğru iken

$$E = \left(\frac{C-B}{n}\right) = 0 \tag{51}$$

dır. H_0 hipotezinin testi için McNemar test istatistiği

$$Z = \frac{C-B}{\sqrt{C+B}} \tag{52}$$

biçimindedir.

7. BAĞIMSIZLIK VE HOMOJENLİK İÇİN Kİ-KARE TESTLERİ

Ki-kare testleri istatistikte önemli yer tutar. Bu testler, istatistik biliminin kurucuları arasında kabul edilen Karl Pearson tarafından geliştirilmiştir. Ki-kare istatistiğine dayanan uyum iyiliği testleri ile bu testler içinde yer alan bağımsızlık ve homojenlik için ki-kare testleri gözlenen frekanslar ile yokluk hipotezi doğru iken beklenen frekansların karşılaştırılması temeline dayanır. Bu tür testlerde gözlenen frekansların beklenen frekanslara uyumunun ölçüsü hesaplanır.

7.1. Bağımsızlık İçin Ki-Kare Testi

Bağımsızlık için ki-kare testinde biri sınıflama diğeri sıralama veya ikisi de sınıflama düzeyinde ölçülen iki değişken arasında ilişki olup olmadığı veya bu değişkenlerin bağımsız olup olmadığını test etmek için kullanılır.

H_0 : Değişkenler bağımsızdır.
H_1 : Değişkenler bağımsız değildir.

veya

H_0 : Değişkenler bağımsızdır.
H_1 : Değişkenler bağımsız değildir.

Test istatistiği aşağıdaki gibi ifade edilir.

$$\sum_{j=1}^{c} \sum_{i=1}^{r} \frac{\left(G_{ij}-B_{ij}\right)^2}{B_{ij}} \sim \chi^2_{(r-1)(c-1)} \tag{53}$$

Buradaki B_{ij} eşitliği aşağıda verilmiştir.

$$P_1 = \frac{(n_{.j})(n_{i.})}{n} \tag{54}$$

c, örneklem sayısı ve r, örneklem hacmini ifade eder.

7.2. Homojenlik İçin Ki-Kare Testi

Bu tür problemlerde, iki örneğin aynı yığından gelip gelmediğini belirlenmeye çalışılır. Bu nedenle yokluk ve alternatif hipotezler

H_0 : Örnekler aynı yığından gelmiştir.
H_1 : Örnekler aynı yığından gelmemiştir.

olarak ifade edilir. Yukarıdaki H_0 hipotezinin testi için bağımsızlık için ki-kare testindeki test istatistiği kullanılır. Bu istatistik daha önce,

$$\sum_{j=1}^{c} \sum_{i=1}^{r} \frac{(G_{ij}-B_{ij})^2}{B_{ij}} \sim \chi^2_{(r-1)(c-1)} \tag{55}$$

olarak verilmiştir. Homojenlik için ki-kare testinde G_{ij} ve B_{ij} değerlerinin bulunması bağımsızlık için ki-kare testindeki gibidir. Bu istatistik için hesaplanan değer χ^2_h olmak üzere,

$$\chi^2_h > \chi^2_{(r-1)(c-1),\alpha} \tag{56}$$

ise H_0 hipotezi reddedilir.

8. İKİDEN FAZLA BAĞIMSIZ ÖRNEK İÇİN TESTLER

İkiden fazla bağımsız örneğin seçildiği yığınların konum parametrelerinin eşitliği hipotezinin testi için bilinen parametrik yöntem "F testi"'dir. Tek faktörlü varyans analizi olarak da bilinen bu yöntemde yokluk ve alternatif hipotezi, yığın sayısı k, j. yığının ortalaması μ_j ve genel ortalama da μ olmak üzere,

$H_0: \mu_1 = \mu_2 = \cdots = \mu_k = \mu$
$H_1: \mu_j$'lerin en az biri farklıdır

biçiminde ifade edilir.

8.1. Medyan Testi

İkiden fazla bağımsız örneğin medyanları aynı olan yığınlardan gelip gelmediğinin araştırılmasında Medyan testi kullanılır. Medyan testi için hipotezler;

H_0 : Örneklerin seçildikleri yığınlar aynı medyana sahiptir.
H_1 : Örneklerin seçildikleri yığınların en az birinin medyanı farklıdır.

biçiminde tanımlanır. m, birleştirilmiş örnek medyanı ve $B_{ij} = \frac{n_j}{2} olmak\ üzere\ ve$

G_{1i} : j. örnekte m değerinden büyük değerli olanların sayısı
G_{2i} : j. örnekte m değerinden eşit veya küçük değerli olanların sayısı

olmak üzere test istatistiği aşağıdaki şekilde elde edilir.

$$\sum_{j=1}^{k} \sum_{i=1}^{2} \frac{\left(G_{ij}-B_{ij}\right)^2}{B_{ij}} \sim \chi^2_{(k-1)(2-1)} \tag{57}$$

8.2. Kruskal-Wallis H Testi

Varyans çözümlemesinde amaç, her biri n_j hacimli k sayıda bağımsız örneğin aynı yığından gelip gelmediğine karar vermektedir. Kruskal-Wallis H testi tek yönlü varyans analizinin parametrik olamayan karşılığıdır. Hipotezler aşağıdaki gibidir.

H_0 : Örneklerin seçildiği k sayıda yığının dağılım fonksiyonları aynıdır.
H_1 : Örneklerin seçildiği k sayıda yığının hepsi aynı medyana sahip değildir.

Test istatistiği H olmak üzere aşağıdaki gibi tanımlanmıştır.

$$H = \frac{12}{n(n+1)} \sum_{j=1}^{k} n_j \left(\overline{R_{.j}} - \frac{n+1}{2}\right)^2 \tag{58}$$

9. İKİDEN FAZLA BAĞIMLI ÖRNEK İÇİN TESTLER

Aynı örnek birimlerinde bağımlı değişken bakımından ölçüm (gözlem) değerleri çok fazla değişkenlik gösterirse Kruskall-Wallis H testi ile gruplar arasındaki farkı ortaya çıkarmak kolay olmayabilir. Bazen ilgili bağımlı değişken bakımından gruplar arasındaki farklılık, aynı grup içindeki birimler arasındaki değişkenlik biçiminde gizlenebilir. Aynı grup içindeki birimler arasındaki değişkenlik etkisi "blok etkisi" olarak adlandırılır.

Örnek birimlerini "bloklar" olarak adlandırılan homojen alt gruplara ayırdıktan sonra ilgili bağımlı değişken bakımından gruplar (faktör düzeyleri) arasında farklılık olup olmadığının araştırılması problemi ile birçok alanda karşılaşılabilinir. Örneğin hastaları yaş gruplarına ayırdıktan sonra ilaçların etkilerinin farklı olup olmadığının araştırılması gibi. Burada yaş grupları bloklar olarak adlandırılır ve her bloktaki hastaların yaşları aynıdır. Bu yöntem istatistiksel deney tasarımında "Rastgele Tamamlanmış Blok Tasarımı" olarak da bilinir.

9.1. Friedman'ın S Testi

Friedman'ın S testi parametrik iki yönlü (faktörlü) varyans analizi yönteminin parametrik olmayan alternatifidir. Bu testin temel özelliği normallik ve homojenlik varsayımı gerektirmesi ve ölçüm değerlerine atanan büyüklük sıra sayılarına dayanmasıdır.

9.2. Cochran'ın Q Testi

Bağımlı değişken ölçüm değeri, iki değer alabilen bir değer değişken özelliği taşıyorsa işlemlerin etki bakımından aynı olup olmadığını belirlemek için kullanılabilecek testlerden biri de Cocran'ın Q testidir. Örneğin A, B, C, D ve E ilaçları farklı yaş gruplarında birer hastaya uygulandıktan sonra, hastaların "tedaviye olumlu tepki vermedi" veya "tedaviye olumlu tepki verdi" biçiminde değerlendirilmesi gibi. Burada "tedaviye olumlu tepki vermedi" için "0" ve "tedaviye olumlu tepki verdi" için de "1" verilebilir.

10. İLİŞKİ KATSAYILARI

Eğer aralarında ilişki olup olmadığı araştırılan iki değişken de oranlama ve/veya eşit aralıklı ölçme düzeyinde ölçülmüşse, yani değişkenlerin ikisi de sayısal değişkenler ise, bu amaçla kullanılabilen istatistiklerden biri örnek korelasyon katsayısıdır. Hesaplama sonucu elde edilen katsayı $[-1, +1]$ aralığında değer alır.

10.1. Spearman'ın Sıra Korelasyon Katsayısı

Spearman sıra korelasyon katsayısı iki değişken arasında ilişki olup olmadığının kontrolü için kullanılır. Veriye atanan büyüklük sıra sayılarına dayanır ve r_s ile elde edilir. Burada d_i aşağıdaki gibidir.

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \qquad \text{ve} \qquad d_i = R(X_i) - R(Y_i) \tag{59}$$

10.2. Kendall'ın τ İlişki Katsayısı

Kendall'ın τ ilişki katsayısı iki değişken arasında ilişki olup olmadığının kontrolü için kullanılır. X_i ve Y_i aralarında ilişki olup olmadığı araştırılan iki örnek ve,

p_i : Her Y_i değeri için doğal sıranın sağlandığı durum sayısı
q_i : Her Y_i değeri için doğal sıranın sağlanmadığı durum sayısı

olarak üzere, τ ilişki katsayısı;

$$\hat{\tau} = \frac{P-Q}{n(n-1)/2} \tag{60}$$

ile de elde edilir. Burada P ve Q aşağıdaki gibidir.

$$P = \sum p_i \qquad \text{ve} \qquad Q = \sum q_i \tag{61}$$

11. EKLER

11.1. Util

```
const frequency = arr =>
  arr.reduce((acc, val) => acc.set(val, 1 + (acc.get(val) || 0)), new Map());

const reduceDigit = (arr, digit) =>
  typeof arr === 'object' ?
    arr.map(_ => parseFloat(_.toFixed(digit))) :
    parseFloat(arr.toFixed(digit));

const flatten = (arr, depth = 1) =>
  depth !== 1 ?
    arr.reduce((a, v) => a.concat(Array.isArray(v) ? flatten(v, depth - 1) : v),
[]) :
    arr.reduce((a, v) => a.concat(v), []);

const sortArr = f => arr => [...arr].sort(f);

const factorial = n =>
  n < 0 ?
    (() => {
      throw new TypeError('Negative numbers are not allowed!');
    })() :
    n <= 1 ? 1 : n * factorial(n - 1);

const combinations = (n, r) =>
  factorial(n) / (factorial(r) * factorial(n - r));

const clone = x => [...x];

const head = arr => arr[0];

const shift = x => x.slice(1);

const sum = (...arr) => [...arr].reduce((acc, val) => acc + val, 0);

const min = (arr, n = 1) => [...arr].sort((a, b) => a - b).slice(0, n);

const max = (arr, n = 1) => [...arr].sort((a, b) => b - a).slice(0, n);

const mean = (...nums) => [...nums].reduce((acc, val) => acc + val, 0) /
nums.length;

const median = arr => {
  const mid = Math.floor(arr.length / 2),
    nums = [...arr].sort((a, b) => a - b);

  return arr.length % 2 !== 0 ? nums[mid] : (nums[mid - 1] + nums[mid]) / 2;
};
```

11.2. Ki-Kare Uyum İyiliği Testi

```
function ChiSquareGoodnessOfFit({ observed, alpha = 0.05, way = 'one-way' }) {
  const c = observed.length;
  const pj = 1 / c;
  const df = c - 1;
  const avg = mean(...observed);
  const sumObserved = sum(...observed);
  const expected = Array.from({ length: c }, () => sumObserved * pj);
  const chiSqrCalc = sum(...expected.map((v, i) => (observed[i] - v) ** 2 / v));
  const pValue = chiSqrCdf(chiSqrCalc, df);
}
```

11.3. Kolmogorov-Smirnov Uyum İyiliği Testi

```
function KolmogorovSmirnovGoodnessFit({
observed, mean = undefined, standardDeviation = undefined, alpha = 0.05 }) {
  const c = observed.length;
  const frequencyMap = frequency(observed);
  const sortedFrequencyMap = new Map([...frequencyMap.entries()].sort());
  const X = [...sortedFrequencyMap.keys()];
  const frequencyX = [...sortedFrequencyMap.values()];
  const Sx = [];
  const cumulativeFrequency = [];

  frequencyX.reduce((prev, curr, i) => (cumulativeFrequency[i] = prev + curr), 0);
  cumulativeFrequency.reduce((prev, curr, i) => (Sx[i] = curr / c), 0);
  let Xf = [];
  let XSubMeanX = [];
  let XSubMeanXFrequencyX = [];

  if (typeof mean === 'undefined' && typeof standardDeviation === 'undefined') {
    Xf = X.map((v, i) => v * frequencyX[i]);
    mean = sum(...Xf) / c;
    XSubMeanX = X.map(v => v - mean);
    XSubMeanXFrequencyX = XSubMeanX.map((v, i) => (v ** 2) * frequencyX[i]);
    standardDeviation = sum(...XSubMeanXFrequencyX) / (c - 1);
  }
  const variance = Math.sqrt(standardDeviation);
  const Z = X.map(v => (v - mean) / variance);
  const fX = X.map(v => normalCdf(v, mean, variance));
  const SxSubfX = Sx.map((v, i) => i === 0 ? fX[i] : Math.abs(Sx[i - 1] - fX[i]));
  const Dc = max(SxSubfX);
}
```

11.4. Normallik İçin Lilliefors Testi

```
function Lilliefors({
observed, mean = undefined, standardDeviation = undefined, alpha = 0.05 }) {
  const c = observed.length;
  const frequencyMap = frequency(observed);
  const sortedFrequencyMap = new Map([...frequencyMap.entries()].sort());
  const X = [...sortedFrequencyMap.keys()];
  const frequencyX = [...sortedFrequencyMap.values()];
  const Sx = [];
  const cumulativeFrequency = [];

  frequencyX.reduce((prev, curr, i) => (cumulativeFrequency[i] = prev + curr), 0);
  cumulativeFrequency.reduce((prev, curr, i) => (Sx[i] = curr / c), 0);
  let Xf = [];
  let XSubMeanX = [];
  let XSubMeanXFrequencyX = [];

  if (typeof mean === 'undefined' && typeof standardDeviation === 'undefined') {
    Xf = X.map((v, i) => v * frequencyX[i]);
    mean = sum(...Xf) / c;
    XSubMeanX = X.map(v => v - mean);
    XSubMeanXFrequencyX = XSubMeanX.map((v, i) => (v ** 2) * frequencyX[i]);
    standardDeviation = sum(...XSubMeanXFrequencyX) / (c - 1);
  }
  const variance = Math.sqrt(standardDeviation);
  const Z = X.map(v => (v - mean) / variance);
  const fX = X.map(v => normalCdf(v, mean, variance));
  const SxSubfX = Sx.map((v, i) => i === 0 ? fX[i] : Math.abs(Sx[i - 1] - fX[i]));
  const Dc = max(SxSubfX);
}
```

11.5. Normallik İçin Shapiro-Wilk Testi

```
function ShapiroWilk({
observed, mean = undefined, standardDeviation = undefined, alpha = 0.05 }) {
  const c = observed.length;
  const frequencyMap = frequency(observed);
  const sortedFrequencyMap = new Map([...frequencyMap.entries()].sort());
  const X = [...sortedFrequencyMap.keys()];
  const frequencyX = [...sortedFrequencyMap.values()];
  const Sx = [];
  const cumulativeFrequency = [];

  frequencyX.reduce((prev, curr, i) => (cumulativeFrequency[i] = prev + curr), 0);
  cumulativeFrequency.reduce((prev, curr, i) => (Sx[i] = curr / c), 0);
  let Xf = [];
  let XSubMeanX = [];
  let XSubMeanXFrequencyX = [];

  if (typeof mean === 'undefined' && typeof standardDeviation === 'undefined') {
    Xf = X.map((v, i) => v * frequencyX[i]);
    mean = sum(...Xf) / c;
    XSubMeanX = X.map(v => v - mean);
    XSubMeanXFrequencyX = XSubMeanX.map((v, i) => (v ** 2) * frequencyX[i]);
    standardDeviation = sum(...XSubMeanXFrequencyX) / (c - 1);
  }
  const variance = Math.sqrt(standardDeviation);
  const Z = X.map(v => (v - mean) / variance);
}
```

11.6. Levene Testi

```
function Levene({ observed, alpha = 0.05, way = 'one-way' }) {
  const c = observed.length;
  const flatObserved = flatten(observed);
  const n = flatObserved.length;
  const ni = observed.map(v => v.length);
  const meanObserved = observed.map(v => mean(...v));
  const Z = observed.map((v, i) => v.map(val => Math.abs(val - meanObserved[i])));
  const sumZi = Z.map((_, i) => sum(...Z[i]));
  const meanZi = sumZi.map((v, i) => v / ni[i]);
  const meanZ = sum(...sumZi) / n;
  const SSB = sum(...ni.map((v, i) => v * ((meanZi[i] - meanZ) ** 2)));
  const SSW = sum(...Z.map((v, i) => sum(...v.map(val => (val - meanZi[i]) **
2))));
  const MSB = SSB / (c - 1);
  const MSW = SSW / (n - c);
  const fCalc = MSB / MSW;
  const pValue = 1 - fCdf(fCalc, c - 1, n - c);
}
```

11.7. Brown-Forsythe Testi

```
function BrownForsythe({ observed, alpha = 0.05, way = 'one-way' }) {

  const c = observed.length;

  const flatObserved = flatten(observed);

  const n = flatObserved.length;

  const ni = observed.map(v => v.length);

  const medianObserved = observed.map(v => median(v));

  const Z = observed.map((v, i) => v.map(val => Math.abs(val -
medianObserved[i])));

  const sumZi = Z.map((_, i) => sum(...Z[i]));

  const meanZi = sumZi.map((v, i) => v / ni[i]);

  const meanZ = sum(...sumZi) / n;

  const SSB = sum(...ni.map((v, i) => v * ((meanZi[i] - meanZ) ** 2)));

  const SSW = sum(...Z.map((v, i) => sum(...v.map(val => (val - meanZi[i]) **
2))));

  const MSB = SSB / (c - 1);

  const MSW = SSW / (n - c);

  const fCalc = MSB / MSW;

  const pValue = 1 - fCdf(fCalc, c - 1, n - c);

}
```

11.8. Layard Ki-Kare Testi

```
function LayardChiSquare({ observed, alpha = 0.05, way = 'one-way' }) {
  const c = observed.length;
  const df = c - 1;
  const flatObserved = flatten(observed);
  const n = flatObserved.length;
  const ni = observed.map(v => v.length);
  const meanObserved = observed.map(v => mean(...v));
  const Z = observed.map((v, i) => v.map(val => val - meanObserved[i]));
  const SSW = Z.map((v, i) => sum(...v.map(val => val ** 2)));
  const SSSW = Z.map((v, i) => sum(...v.map(val => val ** 4)));
  const gamma = ((n * sum(...SSSW)) / (sum(...SSW) ** 2)) - 3;
  const T = 2 + (1 - (c / n)) * gamma;
  const sSqr = SSW.map(v => v / c);
  const logsSqr = SSW.map(v => Math.log10(v / c));
  const calc1 = sum(...ni.map((v, i) =>
    (v - 1) * logsSqr[i])) / sum(...ni.map(v => v - 1));
  const calc2 = ni.map((v, i) => (v - 1) * ((logsSqr[i] - calc1) ** 2));
  const L = sum(...calc2) / T;
  const pValue = 1 - chiSqrCdf(L, df);
}
```

11.9. Binom Testi

```
function Binomial({ observed, piZero, alpha = 0.05, way = 'one-way' }) {
  const n = observed.length;
  const bCalc = sum(...observed);
  const w = way === 'one-way' ? 1 : 2;
  const Pb = Array.from({ length: n + 1 }, (_, i) =>
    combinations(n, i) * (piZero ** i) * ((1 - piZero) ** (n - i)));
  const Pbi = Pb.reduce((acc, cur, i) => {
    return acc.value > alpha / w ?
      { value: acc.value, index: acc.index } :
      { value: acc.value + cur, index: i };
  }, { value: 0, index: 0 });
}
```

11.10. Bir Örnek İşaret Testi

```
function OneSampleSign({ observed, M0, alpha = 0.05, way = 'one-way' }) {
  const n = observed.length;
  const Di = observed.map(v => v - M0);
  const delta = Di.map(v => v > 0 ? 1 : 0);
  const k = sum(...delta);
  const Pk = Array.from({ length: n + 1 }, (_, i) =>
    combinations(n, i) * ((1 / 2) ** i) * ((1 / 2) ** (n - i)));
}
```

11.11. Wilcoxon İşaretli Sıra Sayıları Testi

```
function WilcoxonSignedRanks({ observed, M0, alpha = 0.05, way = 'one-way' }) {
  const n = observed.length;
  const Di = observed.map(v => v - M0);
  const delta = Di.filter(v => v !== 0).map((v, i) => v > 0 ? 1 : 0);
  const absDi = Di.filter(v => v !== 0).map(v => Math.abs(v));
  const indexedAbsDi = absDi.map((v, i) => ({ value: v, index: i }));
  const indexSortedAbsDi = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...indexedAbsDi]);
  const values = [...new Set(indexSortedAbsDi.map(v => v.value))];
  let meanIndice = [];

  values.forEach((item, i) => {
    const indexes = indexSortedAbsDi.map((val, j) => {
      val.index = j;
      return val;
    })
      .filter(a => a.value === item)
      .map(a => a.index + 1);

    meanIndice[i] = indexes.map(_ => mean(...indexes));
  });
  const flatMeanIndice = flatten(meanIndice);

  indexSortedAbsDi.map((v, i) => (v.index = flatMeanIndice[i]));
  const rAbsDi = indexedAbsDi.map(v => v.index);
  const TPlus = sum(...rAbsDi.map((v, i) => v * delta[i]));
}
```

11.12. Rastgelelik için Dizi Parçaları (Run) Testi

```
function Runs({ observed, alpha = 0.05, way = 'one-way' }) {
  let n, n1, n2, ind, meanX;
  let upDown;

  if (observed[0][0]) {
    meanX = mean(...observed[1]);
    upDown = observed[1].map(x => x > meanX ? 'U' : 'D');
    n1 = observed[0].filter(x => x === 'M').length;
    n2 = observed[0].filter(x => x === 'W').length;
    n = n1 + n2;
  } else {
    n = observed.length;
    meanX = mean(...observed);
    upDown = observed.map(x => x > meanX ? 'U' : 'D');
    ind = upDown.reduce((acc, cur) => {
      acc[cur] = (acc[cur] || 0) + 1;
      return acc;
    }, {});
  }
}
```

11.13. Medyan Testi

```
function LocationMedian({ observed, alpha = 0.05 }) {
  const x = observed[0];
  const y = observed[1];
  const n1 = x.length;
  const n2 = y.length;
  const n = n1 + n2;
  const sortedX = sortArr((a, b) => a - b)(x);
  const sortedY = sortArr((a, b) => a - b)(y);
  const sortedXY = sortArr((a, b) => a - b)([...x, ...y]);
  const medianXY = median(sortedXY);

  const xMedian = [[], []];

  x.map((v, i) => v > medianXY ? xMedian[0].push(v) : xMedian[1].push(v));
  const xMedianLength = xMedian.map(v => v.length);

  const yMedian = [[], []];

  y.map((v, i) => v > medianXY ? yMedian[0].push(v) : yMedian[1].push(v));
  const yMedianLength = yMedian.map(v => v.length);

  const P1 = xMedianLength[0] / n1;
  const P2 = yMedianLength[0] / n2;
  const P = (xMedianLength[0] + yMedianLength[0]) / n;
  const Z = (P1 - P2) / Math.sqrt((P * (1 - P) * ((1 / n1) + (1 / n2))));
  const pValue = 1 - zScore(Math.abs(Z));
}
```

11.14. Mann-Whitney Testi

```
function MannWhitney({ observed, alpha = 0.05 }) {
  const x = observed[0];
  const y = observed[1];
  const flattenObserved = flatten(observed);
  const n1 = x.length;
  const n2 = y.length;
  const n = n1 + n2;
  const indexedObs = flattenObserved.map((v, i) => ({ value: v, index: i }));
  const indexSortedObs = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...indexedObs]);
  const values = [...new Set(indexSortedObs.map(v => v.value))];
  const meanIndice = [];
  values.forEach((item, i) => {
    const indexes = indexSortedObs.map((itm, j) => {
      itm.index = j;
      return itm;
    })
      .filter(v => v.value === item)
      .map(v => v.index + 1);
    meanIndice[i] = indexes.map(_ => mean(...indexes));
  });
  const flatMeanIndice = meanIndice.flatten();
  indexSortedObs.map((v, i) => (v.index = flatMeanIndice[i]));
  const rObs = indexedObs.map(v => v.index);
  const rX = Array.from({ length: n1 }, (_, i) => rObs[i]);
  const rY = Array.from({ length: n2 }, (_, i) => rObs[n1 + i]);
  const S = sum(...rX);
  const T = S - ((n1 * (n1 + 1)) / 2);
}
```

11.15. Wald-Wolfowitz Dizi Parçaları Testi

```
function WaldWolfowitz({ observed, M0, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0];
  const xObj = x.map(v => ({ value: v, index: 'X' }));
  const y = observed[1];
  const yObj = y.map(v => ({ value: v, index: 'Y' }));
  const n1 = x.length;
  const n2 = y.length;
  const n = n1 + n2;
  const xy = [...xObj, ...yObj];
  const sortedXY = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...xy]);
  let rCalc = 1;

  Array.from({ length: sortedXY.length - 1 }, (_, i) => {
    if (sortedXY[i].index !== sortedXY[i + 1].index) rCalc++;
    return rCalc;
  });
}
```

11.16. Kolmogorov-Smirnov Testi

```
function KolmogorovSmirnov({ observed, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0];
  const xObj = x.map(v => ({ value: v, index: 'X' }));
  const y = observed[1];
  const yObj = y.map(v => ({ value: v, index: 'Y' }));
  const n1 = x.length;
  const n2 = y.length;
  const n = n1 + n2;
  const xy = [...xObj, ...yObj];
  const sortedXY = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...xy]);
  const Xi = [], Yi = [];
  sortedXY.map(v => {
    v.index === 'X' ? Xi.push(v.value) : Xi.push(null);
    v.index === 'Y' ? Yi.push(v.value) : Yi.push(null);
  });
  const cumulativeFrequencyX = [];
  const cumulativeFrequencyY = [];
  const S1x = [], S2x = [];
  Xi.map(v => typeof v === 'number' ? 1 : null)
    .reduce((prev, curr, i) => (cumulativeFrequencyX[i] = prev + curr), 0);
  cumulativeFrequencyX.reduce((prev, curr, i) => (S1x[i] = curr / n1), 0);
  Yi.map(v => typeof v === 'number' ? 1 : null)
    .reduce((prev, curr, i) => (cumulativeFrequencyY[i] = prev + curr), 0);
  cumulativeFrequencyY.reduce((prev, curr, i) => (S2x[i] = curr / n2), 0);
  const Sx = Array.from({ length: n }, (_, i) =>
    way === 'one-way' ? S2x[i] - S1x[i] : S1x[i] - S2x[i]);
  const D = max(Sx);
}
```

11.17. Fisher Tam Olasılık Testi

```
function FishersExact({ observed, alpha = 0.05, way = 'one-way' }) {
  const a = observed[0][0];
  const b = observed[1][0];
  const n1 = sum(...observed[0]);
  const n2 = sum(...observed[1]);
  const n = n1 + n2;
  const w = way === 'one-way' ? 1 : 2;

  const Pb = Array.from({ length: (n2 + 1) }, (_, i) => {
    return (
      combinations(n2, i) * combinations(n1, (a + b - i))) /
        combinations(n, (a + b)
    );
  });

  const Pbi = Pb.reduce((acc, cur, i) => {
    return acc.value > alpha / w ?
      { value: acc.value, index: acc.index } :
      { value: acc.value + cur, index: i };
  }, { value: 0, index: 0 });
}
```

11.18. Mood Testi

```
function Mood({ observed, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0], y = observed[1];
  const flattenObserved = flatten(observed);
  const n1 = x.length, n2 = y.length;
  const n = n1 + n2;
  const indexedObs = flattenObserved.map((v, i) => ({ value: v, index: i }));
  const indexSortedObs = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...indexedObs]);
  const values = [...new Set(indexSortedObs.map(v => v.value))];
  const meanIndice = [];
  values.forEach((item, i) => {
    const indexes = indexSortedObs.map((itm, j) => {
      itm.index = j;
      return itm;
    })
      .filter(v => v.value === item)
      .map(v => v.index + 1);
    meanIndice[i] = indexes.map(_ => mean(...indexes));
  });
  const flatMeanIndice = meanIndice.flatten();
  indexSortedObs.map((v, i) => (v.index = flatMeanIndice[i]));
  const rObs = indexedObs.map(v => v.index);
  const rX = Array.from({ length: n1 }, (_, i) => rObs[i]);
  const rY = Array.from({ length: n2 }, (_, i) => rObs[n1 + i]);
  const ri = rX;
  const riMinus = ri.map(v => v - ((n + 1) / 2));
  const riMinusSqr = riMinus.map(v => v ** 2);
  const M = sum(...riMinusSqr);
}
```

11.19. Siegel-Tukey Testi

```
function SiegelTukey({ observed, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0];
  const xObj = x.map(v => ({ value: v, index: 'X' }));
  const y = observed[1];
  const yObj = y.map(v => ({ value: v, index: 'Y' }));
  const flattenObserved = flatten(observed);
  const idx = flattenObserved.map((_, i) => i + 1);
  const n1 = x.length;
  const n2 = y.length;
  const n = n1 + n2;
  const xy = [...xObj, ...yObj];
  const sortedXY = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...xy]);
  const sortedX = sortedXY.map(v => v.value);
  const sortedY = sortedXY.map(v => v.index);
  const delta = sortedXY.map(v => v.index === 'X' ? 1 : 0);
  const ai = idx.map((v, i) => {
    if (v % 2 === 0 && v > 1 && v <= (n / 2)) return 2 * v;
    if ((v % 2 !== 0) && (v >= 1) && (v <= (n / 2))) return 2 * v - 1;
    if (v % 2 === 0 && v > (n / 2) && v <= n) return 2 * (n - v) + 2;
    if (v % 2 !== 0 && v > (n / 2) && v < n) return 2 * (n - v) + 1;
  });
  const ST = sum(...ai.map((v, i) => v * delta[i]));
}
```

11.20. İşaret Testi

```
function Sign({ observed, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0];
  const y = observed[1];
  const Di = x.map((v, i) => v - y[i]);
  const delta = Di.filter(v => v !== 0).map((v, i) => v > 0 ? 1 : 0);
  const kCalc = sum(...delta);
  const n = delta.length;
  const w = way === 'one-way' ? 1 : 2;

  const Pk = Array.from({ length: n }, (_, i) =>
    combinations(n, i) * Math.pow(1 / 2, n));

  const Pki = Pk.reduce((acc, cur, i) => {
    return acc.value > alpha / w ?
      { value: acc.value, index: acc.index } :
      { value: acc.value + cur, index: i };
  }, { value: 0, index: 0 });
}
```

11.21. Wilcoxon İşaretli Sıra Sayıları Testi

```
function WilcoxonRanksum({ observed, alpha = 0.05, way = 'one-way' }) {
  const x = observed[0];
  const y = observed[1];
  const Di = x.map((v, i) => v - y[i]);
  const delta = Di.filter(v => v !== 0).map((v, i) => v > 0 ? 1 : 0);
  const absDi = Di.filter(v => v !== 0).map(v => Math.abs(v));
  const indexedAbsDi = absDi.map((v, i) => ({ value: v, index: i }));
  const indexSortedAbsDi = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...indexedAbsDi]);
  const values = [...new Set(indexSortedAbsDi.map(v => v.value))];
  const meanIndice = [];
  values.forEach((item, i) => {
    const indexes = indexSortedAbsDi.map((itm, j) => {
      itm.index = j;
      return itm;
    })
      .filter(v => v.value === item)
      .map(v => v.index + 1);
    meanIndice[i] = indexes.map(_ => mean(...indexes));
  });
  const flatMeanIndice = meanIndice.flatten();
  indexSortedAbsDi.map((v, i) => (v.index = flatMeanIndice[i]));
  const rAbsDi = indexedAbsDi.map(v => v.index);
  const n = rAbsDi.length;
  const TPlus = sum(...rAbsDi.map((v, i) => v * delta[i]));
}
```

11.22. McNemar Testi

```
function McNemar({ observed, alpha = 0.05, way = 'one-way' }) {
  const A = observed[0][0];
  const C = observed[0][1];
  const B = observed[1][0];
  const D = observed[1][1];
  const AB = A + B;
  const AC = A + C;
  const BD = B + D;
  const CD = C + D;
  const n = AB + CD;
  const Z = (C - B) / (Math.sqrt(C + B));
  const pValue = 1 - zScore(Math.abs(Z));
}
```

11.23. Bağımsızlık İçin Ki-Kare Testi

```
function ChiSquareForIndependence({ observed, alpha = 0.05, way = 'one-way' }) {
  const r = observed.length;
  const c = observed[0].length;
  const sumI = Array.from({ length: c }, (_, i) => sum(...observed.map(v =>
v[i])));
  const sumJ = observed.map(v => sum(...v));
  const n = sum(...(sumI || sumJ));
  const expected = Array.from({ length: r }, (_, i) =>
    Array.from({ length: c }, (_, j) => {
      return (sumI[j] * sumJ[i]) / n;
    })
  );
  const chiSqr = expected.map((v, i) =>
    v.map((val, j) => (observed[i][j] - val) ** 2 / val));
  const chiSqrCalc = sum(...chiSqr.map(v => sum(...v)));
  const df = (c - 1) * (r - 1);
  const v = Math.sqrt(chiSqrCalc / (n * min([(c - 1), (r - 1)])));
  const pValue = 1 - chiSqrCdf(chiSqrCalc, df);
}
```

11.24. Homojenlik İçin Ki-Kare Testi

```
function ChiSquareForHomogeneity({ observed, alpha = 0.05, way = 'one-way' }) {
  const r = observed.length;
  const c = observed[0].length;
  const sumI = Array.from({ length: c }, (_, i) => sum(...observed.map(v =>
v[i])));
  const sumJ = observed.map(v => sum(...v));
  const n = sum(...(sumI || sumJ));
  const expected = Array.from({ length: r }, (_, i) =>
    Array.from({ length: c }, (_, j) => {
      return (sumI[j] * sumJ[i]) / n;
    })
  );
  const chiSqr = expected.map((v, i) =>
    v.map((val, j) => (observed[i][j] - val) ** 2 / val));
  const chiSqrCalc = sum(...chiSqr.map(v => sum(...v)));
  const df = (c - 1) * (r - 1);
  const pValue = 1 - chiSqrCdf(chiSqrCalc, df);
}
```

11.25. Medyan Testi

```
function Median({ observed, alpha = 0.05 }) {
  const ni = observed.map(v => v.length);
  const n = sum(...ni);
  const flattenObserved = flatten(observed);
  const m = median(flattenObserved);
  const sortedObserved = observed.map(v => sortArr((a, b) => a - b)(v));
  const nMedian = Array.from({ length: ni.length }, () => [[], []]);
  observed.map((v, i) => v.map((val, j) =>
    val > m ? nMedian[i][0].push(val) : nMedian[i][1].push(val)));
  const nMedianLength = nMedian.map(v => v.map(val => val.length));
  const sumI = Array.from({ length: nMedianLength[0].length }, (_, i) =>
    sum(...nMedianLength.map(v => v[i])));
  const sumJ = Array.from({ length: nMedianLength.length }, (_, i) =>
    sum(...nMedianLength[i]));
  const expected = Array.from({ length: sumJ.length }, (_, i) =>
    Array.from({ length: sumI.length }, (_, j) => sumJ[i] / 2));
  const chiSqr = expected.map((v, i) => v.map((val, j) =>
    (nMedianLength[i][j] - val) ** 2 / val));
  const chiSqrCalc = sum(...chiSqr.map(v => sum(...v)));
  const df = ni.length - 1;
  const pValue = 1 - chiSqrCdf(chiSqrCalc, df);
}
```

11.26. Kruskal-Wallis H Testi

```
function KruskalWallis({ observed, alpha = 0.05 }) {
  const ni = observed.map(v => v.length);
  const n = sum(...ni);
  const flattenObserved = flatten(observed);
  const indexedObs = flattenObserved.map((v, i) => ({ value: v, index: i }));
  const indexSortedObs = sortArr((_x, _y) =>
    _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...indexedObs]);
  const values = [...new Set(indexSortedObs.map(v => v.value))];
  const meanIndice = [];

  values.forEach((item, i) => {
    const indexes = indexSortedObs.map((itm, j) => {
      itm.index = j;
      return itm;
    })
      .filter(v => v.value === item)
      .map(v => v.index + 1);
    meanIndice[i] = indexes.map(_ => mean(...indexes));
  });
```

```
  const flatMeanIndice = flatten(meanIndice);
  indexSortedObs.map((v, i) => (v.index = flatMeanIndice[i]));
  let rr = indexedObs.map(v => v.index);
  const clonedObs = clone(observed);

  const rObs = clonedObs.map(v => v.map(val => {
    val = head(rr);
    rr = shift(rr);
    return val;
  }));
  const Rij = rObs.map(v => sum(...v));
  const H = (12 / (n * (n + 1))) * sum(...Rij.map((v, i) =>
    (v ** 2) / ni[i])) - (3 * (n + 1));
  const frequencyRank = frequency(flatMeanIndice);
  const frequencyArr = [...frequencyRank.values()];
  const Ti = frequencyArr.map(v => (v - 1) * v * (v + 1));
  const HStar = H / (1 - (sum(...Ti) / ((n ** 3) - n)));
}
```

11.27. Spearman'ın Sıra Korelasyon Katsayısı

```
function SpearmanRankCorrelation({ observed, alpha = 0.05 }) {
  const x = observed[0], y = observed[1];
  const n = (x.length || y.length);
  const indexedObs = observed.map(v => v.map((v, i) => ({ value: v, index: i })));
  const indexSortedObs = indexedObs
    .map(v => sortArr((_x, _y) =>
      _x.value > _y.value ? 1 : _x.value === _y.value ? 0 : -1)([...v]));
  const values = indexSortedObs.map(v => [...new Set(v.map(_ => _.value))]);
  const meanIndice = [[], []];
  values.map((v, i) => v.forEach((item, j) => {
    const indexes = indexSortedObs[i].map((val, k) => {
      val.index = k;
      return val;
    })
     .filter(_ => _.value === item)
      .map(_ => _.index + 1);
    meanIndice[i][j] = indexes.map(_ => mean(...indexes));
  }));
  const flatMeanIndice = meanIndice.map(v => flatten(v));
  indexSortedObs.map((v, i) => v.map((val, j) =>
    (val.index = flatMeanIndice[i][j])));
  const rObs = indexedObs.map(v => v.map(val => val.index));
  const rX = rObs[0];
  const rY = rObs[1];
  const di = rX.map((v, i) => v - rY[i]);
  const diSqr = di.map(v => v ** 2);
  const sumDiSqr = sum(...diSqr);
  const rs = 1 - ((6 * sumDiSqr) / (n * ((n ** 2) - 1)));
}
```

11.28. Kendall’ın τ İlişki Katsayısı

```
function Kendall({ observed, alpha = 0.05 }) {
  const x = observed[0];
  const y = observed[1];
  const n = (x.length || y.length);
  const xyObs = x.map((v, i) => ({ key: v, value: y[i] }));
  const keySortedObs = sortArr((_x, _y) =>
    _x.key > _y.key ? 1 : _x.key === _y.key ? 0 : -1)([...xyObs]);
  const pq = [[], []];
  const temp = keySortedObs.map(v => v.value);
  keySortedObs.map(v => temp.map(val =>
    v.value > val ? pq[0].push(val) : pq[1].push(val)));
  // TODO
}
```

12. KAYNAKLAR

1. Baldick, C., Oxford Dictionary of Literal Terms, OUP Oxford, 2015.
2. Gangam, H. and Altunkaynak, B., Parametrik Olmayan İstatistiksel Yöntemler, 2012.
3. Not, D. O. R., How to Select Appropriate Statistical Test?, Journal of Pharmaceutical Negative Result | October, 1,2 (2010) 61.
4. Massey Jr, F.J., The Kolmogorov-Smirnov Test for Goodness of Fit, Journal of the American Statistical Association, 46,253 (1951) 68-78.
5. Stephens, M.A., EDF Statistics for Goodness of Fit and Some Comparisons, Journal of the American Statistical Association, 69,347 (1974) 730-737.
6. Lilliefors, H.W., On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown, Journal of the American Statistical Association, 62,318 (1967) 399-402.
7. Shapiro, S.S. and Wilk, M.B., An Analysis of Variance Test for Normality (Complete Samples), Biometrika, 52,3-4 (1965) 591-611.
8. Glass, G. V, Testing Homogeneity of Variances, American Educational Research Journal, 3,3 (1966) 187-190.
9. Brown, M.B. and Forsythe, A.B., The Small Sample Behavior of Some Statistics Which Test the Equality of Several Means, Technometrics, 16,1 (1974) 129-132.
10. Güriş, S. and Astar, M., Bilimsel Araştırmalarda SPSS ile İstatistik, 2014.
11. Wadsworth, G.P. and Bryan, J.G., Introduction to Probability and Random Variables, 7, McGraw-Hill New York:, 1960.
12. Lehmann, E.L. and D' Abrera, H.J.M., Nonparametrics: Statistical Methods Based on Ranks, Springer New York, 2006.
13. Wilcoxon, F. and Wilcox, R.A., Some Rapid Approximate Statistical Procedures, Lederle Laboratories, 1964.
14. Upton, G.J.G., Fisher's Exact Test, Journal of the Royal Statistical Society. Series A (Statistics in Society) (1992) 395-402.
15. Birnbaum, Z.W., On a Use of the Mann-Whitney Statistic, 1956, Proceedings of the third Berkeley symposium on mathematical statistics and probability, 13-17.
16. Gibbons, J.D. and Chakraborti, S., Nonparametric Statistical Inference, Springer, 2011.

17. Slud, E. and Wei, L.J., Two-Sample Repeated Significance Tests Based on the Modified Wilcoxon Statistic, Journal of the American Statistical Association, 77,380 (1982) 862-868.
18. Christensen, R., Plane Answers to Complex Questions, Springer New York, New York, NY, 2011.
19. Kazım, Ö., Paket Programlar İle İstatistiksel Veri Analizi, Kaan Kitabevi, Eskişehir, 2004.
20. Siegel, S., Nonparametric Statistics for the Behavioral Sciences., 1956.
21. Kruskal, W.H., A Nonparametric Test for the Several Sample Problem, The Annals of Mathematical Statistics (1952) 525-540.
22. Toutenburg, H., Lehmann, EL, Nonparametrics:Statistical Methods Based on Ranks, San Francisco. Holden-Day, Inc., 1975. 480 S., Zamm-Journal of the Applied Mathematics and Mechanics/Zeitschrift Für Angewandte Mathematik Und Mechanik, 57,9 (1977) 562.
23. Ananda, M.M.A. and Weerahandi, S., Two-Way Anova with Unequal Cell Frequencies and Unequal Variances, Statistica Sinica (1997) 631-646.
24. Kum, S., Guideline For Suitable Statistical Test Selection, Plevra Bulteni, 8,2, 2014, 26-29.
25. Lang, T., Twenty Statistical Errors Even You Can Find in Biomedical Research Articles, 2004.
26. Akdeniz, F., Olasılık ve İstatistik, Nobel Kitapevi, Adana, 1995.
27. Sedgwich, P., Pearson's Correlation Coefficient, Bmj, 345,7 (2012).
28. Myers, L.and Sirois, M.J., Spearman Correlation Coefficients, Differences between, Wiley StatsRef: Statistics Reference Online, 2006.
29. Sen, P.K., Estimates of the Regression Coefficient Based on Kendall's Tau, Journal of the American Statistical Association, 63,324 (1968) 1379-1389.
30. Field, A. and Hole, G., How to Design and Report Experiments, Sage, 2002.
31. Leech, N.L., Barrett, K.C. and Morgan, G.A., IBM SPSS for Intermediate Statistics: Use and Interpolation, Routledge, 2014.
32. Mertler, C.A. and Reinhart, R.V., Advanced and Multivariate Statistical Methods: Practical Application and Interpretation, Routledge, 2016.
33. Code Team Work, Veri Analizi Platformu, Yayın No: 1.0., Trabzon 2016.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı: Regaip Ergenekon Yiğit
Doğum Tarihi ve Yeri: 23.11.1995 - ANTALYA
Telefon Numarası: +90 545 895 9420
Mail Adresi: yigitergenekon@gmail.com

Yaptığı Çalışmalar

- Okabe Stok Kontrol Yazılımı
  (C++ ile Nesne Yönemlimli Programlama)
  Danışman: Dr. Öğr. Üyesi TOLGA BERBER

- Mikroskop Görüntülerinin İstatistiksel Yazılımlar ile Analizi
  (13. Uluslararası İstatistik Öğrenci Kolokyumu)
  Danışman: Doç. Dr. ORHAN KESEMEN

- Karadeniz Teknik Üniversitesi Hastanesi Otopakı Hakkında Anket
  Danışman: Dr. Öğr. Üyesi UĞUR ŞEVİK