15. ÖNGÖRÜSEL ANALİTİK YÖNTEMLER: SİMÜLASYON

H. Kemal İLTER

Bölüme Genel Bakış

Simülasyon çalışması, gerçek veya fiziksel bir sistemin çıktılarını veya davranışını tahmin etmek için tasarlanmış ve bilgisayarla yapılan bir matematiksel modelleme süreci olarak ele alınır. Seçilen matematiksel modelin güvenilirliğini kontrol etmeye de imkan sağladığı için, simülasyon bir çok doğal (fizik, kimya, biyoloji) ve sosyal (ekonomi, psikoloji, sosyal bilimler, sağlık, mühendislik) sistemin modellendiği çalışma alanları için önemli bir araçtır. Amaç, gerçek sistemin davranışlarının ve sistem çıktılarının benzerlerinin oluşturulması ve sistemin tanımlanmasıdır.

Anahtar Kavramlar

İstatistiksel analiz
Modelleme
Monte Carlo yöntemi
Olasılık dağılımları
Rastsallık
Simülasyon yaklaşımları
Stokastik süreçler

Giriş

Öngörüsel analitik, büyük veriye ilişkin olarak, geçmişte olan olayların özelliklerinin tanımlanması durumunda gelecekle ilişkili tahmin yapmayı kolaylaştıran bir analiz şeklidir ve iki temel yaklaşımla, örüntü tanıma (pattern recognition) ve simülasyon (simulation) ile uygulanmaktadır. Farklı dallarda çeşitli kullanımları olan öngörüsel analitik çalışmalarının sağlık, telekomünikasyon, üretim, turizm gibi bazı endüstrilerde daha yaygın kullanıldığı gözlenmektedir.

Yapay zekanın ve makine öğrenmesinin öngörüsel analitiğe katkısının yanısıra simülasyon, problemlerin tanımlanması, gelecekle ilgili tahminler geliştirilmesi ve optimum kararları desteklemesi açısından güçlü bir araç olarak görülebilir.

Simülasyon, bir sisteme ilişkin operasyonun belirli bir zaman süresince çalışan yaklaşık bir taklidi olarak düşünülebilir. Diğer yandan, bir süreci ifade edecek şekilde kullanımı da yaygındır. Bu açıdan bakıldığında, simülasyon, bir sistemin davranışlarının belirlenebilmesi için o sistemin laboratuvar ortamında oluşturulması anlamına gelebilmektedir. Simülasyon kavramı, eğitim, mühendislik, oyun, test gibi farklı bağlamlarda kullanılıyor olsa da, bu bölümde *bilgisayarla oluşturulan deneylerin bilimsel modellerinin simülasyonu* olarak ele alınmaktadır. Türkçe literatürde bazı yazarlar *simülasyon* kelimesi yerine *benzetim* kelimesini kullanmayı tercih etmektedir.

Gerçek sistemin özelliklerinin ancak sistem çalıştırıldığı durumda ortaya çıkacağı düşünüldüğünde, bir simülasyon çalışması belirli bir sistemin çalıştırılmasının tehlikeli, maliyetli veya karmaşık olması durumunda sisteme ilişkin modelin geliştirilmesini ve gerçek sistem sonuçlarına istatistiki olarak benzeyen sonuçlar üretilmesini sağlar. Ek olarak, simülasyon çalışmalarındaki bazı durumlarda henüz üretilmemiş bir sistemin (Örn. henüz yatırım aşamasındaki bir fabrikanın üretim süreçleri) veya üretilmesi imkansız bir sistemin (Örn. Jüpiter'e gönderilecek bir uzay aracına ilişkin süreçler) özellikleri araştırılıyor olabilir. Simülasyon, bir sistemin belirli bir zaman içinde sayısal olarak değişimini veya evrimleşmesini modellemek için de kullanılabilmektedir.

Simülasyon çalışması, gerçek veya fiziksel bir sistemin çıktılarını veya davranışını tahmin etmek için tasarlanmış ve bilgisayarla yapılan bir matematiksel modelleme süreci olarak ele

alınır. Seçilen matematiksel modelin güvenilirliğini kontrol etmeye de imkan sağladığı için, simülasyon bir çok doğal (fizik, kimya, biyoloji) ve sosyal (ekonomi, psikoloji, sosyal bilimler, sağlık, mühendislik) sistemin modellendiği çalışma alanları için önemli bir araçtır. Amaç, gerçek sistemin davranışlarının ve sistem çıktılarının benzerlerinin oluşturulması ve sistemin tanımlanmasıdır.

Çeşitli durumlarda bir analitik modelin simülasyonuna ihtiyaç duyulabilir. Örneğin,

- analitik modele ilişkin tüm varsayımların anlamlı ve geçerli olmadığı durumlarda,
- istenen sonuçları elde etmenin matematiksel karmaşıklıktan dolayı zor olduğu durumlarda,
- iyi (optimal olması gerekmeyen veya alt-optimal) çözümlerin yeterli olduğu durumlarda.

Bir çok pratik problemin simülasyon modeli bilgisayarların kullanımını gerektirmektedir. Bir simülasyon modelinin oluşturulmasında, problemin karmaşıklığı, simülasyon maliyeti, kullanıcı deneyimi, arzu edilen sonuçların detayları gibi nedenlerle kullanılan farklı bilişim tercihleri olabilmektedir:

- Bir tablolama yazılımının kullanılması (MS Excel, Apple Numbers, Google Sheets, vb.; @Risk, Crystal Ball gibi eklentilerle birlikte),
- Programlama dillerinin kullanılması (Python, FORTRAN, PL/1, C, Pascal, Basic, vb.),
- Simülasyon dillerinin kullanılması (GPSS, SIMAN, SLAM, vb.),
- Bir simülasyon yazılımının kullanılması (ARENA, PROMODEL, vb.).

Simülasyon modellerini etkin kullanabilmek için, modelleme ve programlama bilgisinin yanısıra, istatistik bilgisinin de önemli olduğu görülmektedir.

Karmaşık sistemlerin simülasyonuna ilişkin bazı örnekler

1997, 66.239 aracın (tank, kamyon ve diğer askeri taşıt araçları) belirli bir bölgeyi ele geçirmesine ilişkin çöl savaşı simülasyonu (NASA)
2005, İnsan beyninin moleküler düzeydeki simülasyonu (Blue Brain, EPFL)
2005, Tüm canlılarda kompleks protein üretiminden sorumlu ribozomun 2,64 milyon atomlu modelinin simülasyonu (LANL)
2012, *Mycoplasma genitalium* bakterisinin tüm yaşam döngüsününün simülasyonu (STANFORD)
2013, Bir milyar atomdan oluşan malzeme deformasyonu modelinin simülasyonu (IBM)

Simülasyon Türleri

Simülasyon çalışmalarının amacına, incelenen sistemin özelliklerine, bileşenleri arasındaki ilişkilere, simülasyon zamanının akışına veya çalıştırılma şekline göre farklı şekillerde sınıflandırmalar söz konusudur.

Sınıflandırma

Simülasyona ilişkin, çalışmanın özelliklerine göre farklı sınıflandırmalar yapılabilir. Fiziksel özelliklerine, çalıştırma zamanına, değişkenlerin belirsizlik durumlarına veya simülasyonun modellendiği teknik platforma göre farklı sınıflar kullanılabilir.

Sınıflandırma, simülasyonda kullanılan bileşenlerin fiziksel özelliklerine göre yapılabilir. *Fiziksel simülasyon*, fiziksel objelerin gerçek sistem bileşenlerini temsilen kullanılarak bir model oluşturulmasıdır. Bu objelerin boyutları ve maliyetleri gerçek sistemdekine oranla daha

düşük olduğu için tercih edilebilir. *Etkileşimli simülasyon*, insanın simülasyon çalışmasının bir parçası olarak bulunması ve sistem davranışını bireyin davranışıyla değiştirebildiği modelleme çalışmaları olarak düşünülebilir. *Bilgisayarlı simülasyon*, bu bölümün ana temasını oluşturan ve gerçek sistemlerin bilgisayarda modellenebilmesine izin veren çalışmalar olarak tanımlanabilir.

Simülasyonlar, çalıştırılırken kullanılan zaman akışının sürekliliğine göre, sürekli ve kesikli simülasyon olarak sınıflandırılabilir. *Sürekli simülasyon*, zamanın sürekliliği dikkate alınarak yapılan çalışmaları ifade ederken diferansiyel denklemlerin kullanımını gerektirmektedir. *Kesikli simülasyon* ise, sistemi belirli zaman dilimlerinde takip edilebilmesiyle ve sistemin belirli zamanlarda değişen durumlarına ilişkin değerleriyle modelleme çalışmalarına imkan verir.

Başka bir sınıflandırma değişkenlerin belirli olmasına (deterministik) veya belirsizlik içermesine (stokastik) bağlı olarak yapılabilir. *Deterministik simülasyon*, model içinde kullanılan değişkenlerin deterministik algoritmalarla hesaplanabildiği ve *stokastik simülasyon* değişkenlerin rastsal değişimlerin bir sonucu olarak çıktığı bir çalışmayı anlatmaktadır. Stokastik simülasyonların çoğunda, yapay-rastsal (sözde-rastsal) sayıların kullanıldığı Monte Carlo yöntemleri tercih edilmektedir.

Zamana dayalı diğer bir sınıflandırma, sistemlerin zamanla değişip değişmediğine bağlı olarak ortaya çıkar. *Statik simülasyon modelleri*, sistemdeki anlık durum değişikliğinin takip edilebileceği, zamanla değişmeyen ve bu nedenle zaman akışını dikkate almayan modellerdir. *Dinamik simülasyon modelleri* zaman içinde değişen, gelişen ve evrilen sistemlere ilişkin modellerdir.

Stokastik modeller, deterministik modellerin aksine, rastsal ya da olasılığa dayalı değişkenler içerir. Diğer yandan, dinamik modeller, statik modellerin aksine, zamanı özel bir bağımsız değişken olarak ele alır. Bazı sistemleri bu iki sınıfın birlikte tanımlaması olağandır:

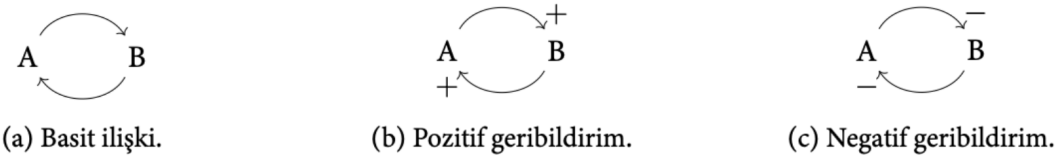
- deterministik ve statik modeller (Örn. x'in bağımsız ve y'nin bağımlı değişken olduğu birinci dereceden bir eşitlik),
- stokastik ve statik modeller (Örn. bir para atışında yazı ya da tura gelme olasılığı),
- deterministik ve dinamik modeller (Örn. zamanın, t, bağımsız değişken olduğu bir diferansiyel eşitlik),
- stokastik ve dinamik modeller (Örn. tek sunuculu bir kuyruk sistemindeki bekleme sürelerine ilişkin model).

Simülasyonun belirli bir bilgisayarda, bir bilgisayarın çok sayıdaki işlemcilerinde aynı anda veya farklı bilgisayar sistemlerinin birleşiminden oluşan bir ağda çalıştırılması mümkündür. Bu durum diğer bir sınıflandırmayı sunmaktadır. Genelde olduğu gibi, modelleme ve simülasyon çalışmasının tek bir bilgisayarda oluşturulması *özerk simülasyon* (stand-alone simulation) olarak adlandırılabilir. Simülasyonun hızlandırılması ve yüksek performans istenen çalışmalardaki hesaplama işlemlerinin yükünün çok sayıda işlemciye dağıtılarak yapılması *paralel simülasyon*u tanımlamaktadır. Dağıtık veritabanlarına, farklı işletim sistemlerine veya kaynaklara erişimin gerektirdiği durumlarda simülasyonun birden fazla sayıda bilgisayar sisteminde çalıştırılması *dağıtık simülasyon* için bir tanımlama olabilir.

Simülasyon Yaklaşımları

Simülasyon yaklaşımları, sistemleri farklı bakış açısıyla incelemek için kullanılan *yapısal simülasyon kuramları* olarak da adlandırılabilir. Temel özelliklerine bakıldığında bunların, sistemin bileşenleri arasındaki ilişkilerin sistem davranışına etkisinin ele alınmasını sağlayan yaklaşımlar olduğu söylenebilir.

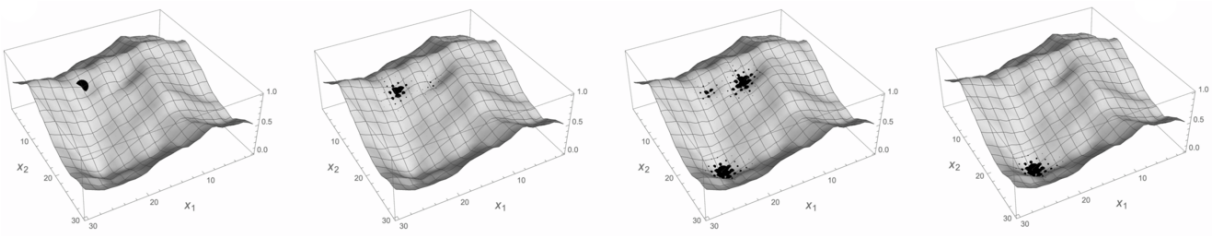
Sistem dinamiği yaklaşımı, alt-sistemlerin nedensel ilişkilerinin bir sistemin davranışını nasıl etkilediğine odaklanır. Bu yaklaşım, belirli bir sistemin (Örn. örgüt) döngüsel nedensellik içeren bir dizi basit süreç (Örn. a değişkeni, yine kendisini etkileyen b değişkenini etkiler) olarak modellenmesini sağlar (Şekil 1a). Bu döngüsel nedensellik, güçlendiren ve artıran pozitif geribildirim veya zayıflatan ve azaltan negatif geribildirim şeklinde olabilir (Şekil 1b ve Şekil 1c). Sistem dinamiği yaklaşımı, karmaşık nedensellik ve zamanlama içeren sistemlerin davranışlarının anlaşılmasında kullanım alanı bulmuştur.



(a) Basit ilişki. (b) Pozitif geribildirim. (c) Negatif geribildirim.

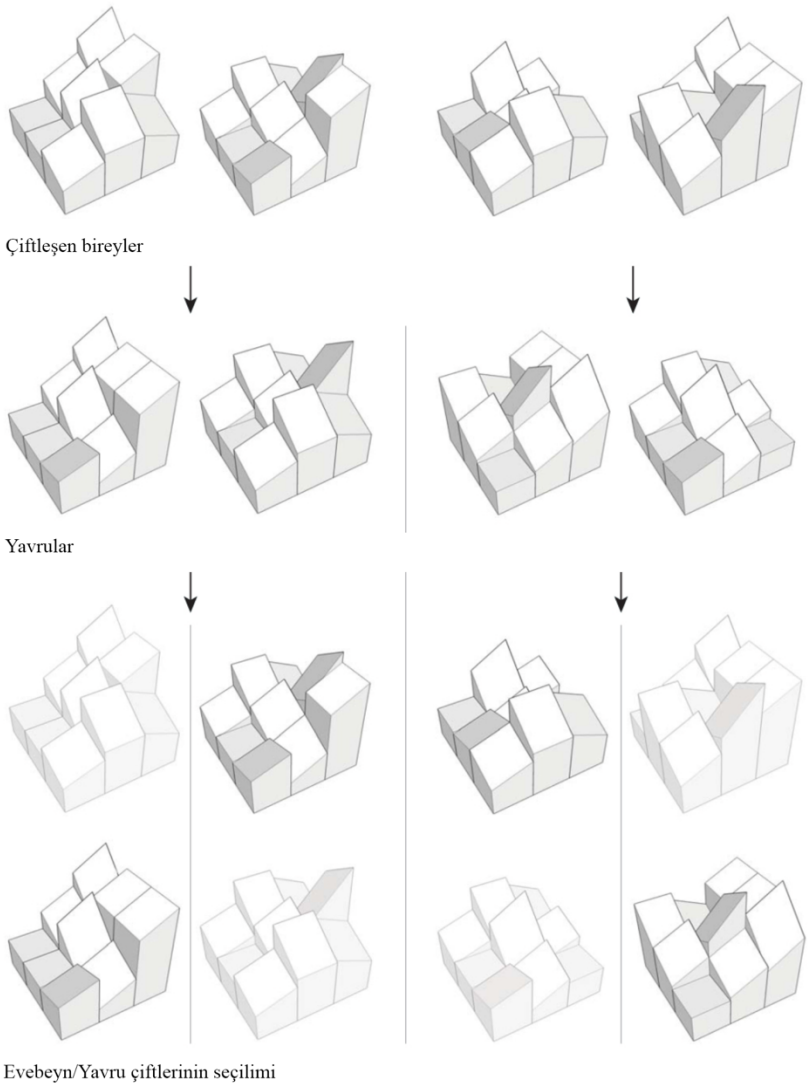
Şekil 1. Döngüsel nedensellik.

NK uyum yüzeyleri yaklaşımı, sistem elemanları arasındaki ilişkilerin önemli olduğu durumlarda modüler bir sistemin, optimal noktaya ulaşmak için, ne kadar hızlı ve etkin uyum gösterdiğine odaklanır. Bu yaklaşımda sistem, N sayıdaki düğümden oluşan bir küme ve kümedeki düğümler arasındaki K sayıdaki etkileşim yoluyla kavramsallaştırılır. Sistemin, optimal noktayı bulmak için, süregelen adımlar ve sıçramalar gibi uyum (adaptation) veya arama (search) stratejilerini kullandığı varsayılır. NK uyum yüzeyleri yaklaşımı, düğümler arasındaki güçlü veya zayıf ilişkilerden etkilenen modüler bir sistemde uyumun ne kadar hızlı ve etkin optimum bir noktaya ulaştığının anlaşılması amacıyla kullanılır (Şekil 2).



Şekil 2. NK uyum yüzeyinde hareket eden bir popülasyon.

Genetik algoritmalar, genlerden oluşan heterojen bir popülasyonun ne kadar hızlı ve etkin öğrendiğine odaklanır. Adaptasyon, birikmiş deneyim yoluyla kademeli gelişimi destekleyen stokastik bir evrim süreci (çeşitlilik, seçilim, üreme) aracılığıyla gerçekleşir. Çeşitlilik (variation) iki şekilde oluşur; mutasyon (hatalara yol açan bir ya da daha fazla gendeki rastsal değişim, İng. mutation) ve genetik değişim (varolan genlerin bileşiminden sorumlu olan ajanlardaki gen kümelerinin rastsal değişimi, çaprazlama—crossover). Seçilim, ajanların (genlerin) performanslarına (fitness) göre gerçekleşir. Üreme, seçilen ajanların bir nesilden diğerine kopyalanmasıdır (Şekil 3). Zamanla, başarılı çeşitliliğin korunma olasılığı daha yüksek olacak ve gelecekteki çeşitliliğin temelini oluşturacaktır. Sonunda popülasyonda yalnızca yüksek performanslı ajanlar kalacaktır ve genellikle tek bir ajan formu hayatta kalır. Bir popülasyonun evrimsel adaptasyonu, daha önce alınan kararların bir devamı olarak (path dependent), tümleşik (combinatorial) optimal bir forma doğru olacaktır. Genetik algoritmalar, sistem bileşenlerinin veya popülasyon bireylerinin deneylerle iyileştirilmiş çözümleri nasıl öğrendiğini (adaptive learning) açıklamak için kullanılmaktadır.



Şekil 3. Genetik seçilim.

Hücresel otomata, uzamsal olarak ilişkili yarı-zeki ajanlar arasındaki mikro düzeydeki etkileşimlerden makro düzey sistem modellerinin ortaya çıkmasına odaklanır. Uzamsal ilişki, bileşenlerin birbirlerini etkileme derecesinin aralarındaki uzaklığa bağlı olduğunu anlatır. Bileşenler, etkileşimin nasıl olduğuna ilişkin basit kurallara göre davranır. Kurallar, yakındaki bileşenlerin uzak olanlardan daha fazla etkilendiği uzamsal süreçlerle ilgili olmakla birlikte, genellikle tüm bileşenler için aynı ve deterministiktir. Hücresel otomata, makro düzeydeki örüntülerin, mikro-düzeydeki uzamsal süreçlerden (dağılma, yayılma, ayrışma, rekabet, vb.) nasıl ortaya çıktığını araştırmak için kullanılmaktadır.

Stokastik süreçler, simülasyonların rastsal süreçlerle özel olarak tasarlanmalarını sağlayan esnek bir yaklaşımdır. Bu yaklaşım, üzerinde çalışılan sistem veya onun kuramsal dayanağı hakkında bir varsayımda bulunmaz ve yukarıda bahsedilen yapısal simülasyon kuramlarına veya varsayımlarına uymayan durumlar için kullanılır. Dinamik değişkenliğe ya da zamansal geçişlere sahip sistemlerin farklı süreçlerinin bir araya getirilmesinde, rastsallığın çeşitli kaynaklardan (sistem çevresindeki olaylardan, süreçlerin zamansal ilişkilerinden veya sistemin kendi içinden) elde edildiği ve stokastik —basit ya da karmaşık— istatistiki dağılımlar olarak kullanıldığı görülmektedir. Simülasyonun oluşturulmasında genellikle bilinen süreçler (Örn. Markov zincirleri) ve rastsallık kaynakları (Örn. varış zamanları için Poisson dağılımı) kullanılır.

Rastsal Sayı, Rastsal Değişken ve Stokastik Süreç Üretmek

Rastsallık, düzensizlik durumunu anlatır. Rastsal süreç ise, sonuçları deterministik olarak açıklanamayan ancak bir olasılık dağılımını takip eden olaylar bütünüdür. İstatistiğin konuları içinde sıklıkla karşılaşılan rastsallık, belirli bir korelasyondan (ilişkiden) yoksun olayları istatistiksel olarak açıklamak için kullanılır.

Rastsallık, çeşitli bilim dallarında son yüzyılda önemini artırmış görünmektedir. Örneğin fizikte, termodimanik alanında, istatistiksel mekaniğin gelişmesine ilişkin olarak moleküllerin rastsal hareketleri bir fikir olarak ortaya çıkmıştır ve kuantum mekaniğinin temelini oluşturacak çeşitli konularda da kullanılmaktadır. Biyolojide, evrim teorisinin temelinde bulunan, genlerin çevre koşullarından etkilenerek mutasyona uğraması ve canlının hayatta kalma başarısının belirlenmesi gibi konularda kullanılmaktadır. Matematikte, olasılık teorisinin içinde yeralan rastsallık, olayların oluşma olasılıklarını matematiksel olarak açıklamak için kullanılmaktadır ve kökeni bahis oyunlarına dayanmaktadır. İstatistik, gözlem yoluyla belirlenen verilerin olasılık dağılımlarını belirlemek için kullanılmaktayken, simülasyon, gözlem yaparak elde edilemeyecek kadar çok miktarda rastsal sayı oluşturmak için üzerinde çalışılan bir alandır.

Rastsallık, matematiksel bir algoritmayla ortaya çıkarıldıysa yapay-rastsallık (pseudo-random) olarak adlandırılır. Rastsallık gerçekte öngörülemez olmasına rağmen yapay-rastsallığa sahip bir olayın, aslında o olayın saf (gerçek) rastsal olmadığını, olayın olasılık veya beklenen değer gibi kavramlarla açıklanabilen bir karakteristiğe sahip olduğu görülür. Örneğin bir madeni para atışının sonucu öngörülemez olmasına rağmen, sonuçlarının (yazı veya tura) ortaya çıkma olasılıklarından bahsedilebilir ve rastsaldır. Belirli bir sayıdaki atışın yarısında tura gelmesinin beklenmesi işte bu olasılığa dayalı bir beklenti olacaktır. Gerçekleşen değerler farklı olabilir ancak yeterince çok sayıda atışın sonunda yazı ve tura sayılarının birbirine çok yaklaştığının görülmesi bu olayın olasılığının bir sonucudur.

Rastsallık, diğer yandan, tümüyle öngörülemezlik değildir. Örneğin, kriptoloji söz konusu olduğunda, aynı mesajı alan iki kişiden biri mesajı tamamen rastsal olarak algılarken, diğeri için, *mesajın anahtarına* sahip olduğu için, tamamen anlamlı olacaktır. Diğer yandan, deterministik kozmolojide uzayın rastsal olmadığı ancak öngörülemez olduğuna dair hipotezler kullanılmaktadır.

Rastsallık temelde üç durum sonucunda ortaya çıkabilir:

- Rastsallık doğadaki olaylardan kaynaklanıyor olabilir (Örn. Brown hareketi ve Wiener metodu).
- Rastsallık başlangıç durumlarından, diğer bir deyişle başlangıçtaki mikro farklılıkların bir sonucu olarak, oluşuyor olabilir (Örn. Kaos teorisi).
- Rastsallık sistem tarafından üretiliyor olabilir (yapay-rastsallık, Örn. algoritmalar).

Bu bölümde düzgün (uniform) dağılımla rastsal sayı üretme yöntemleri ele alınmaktadır.

Yapay-rastsal sayı kavramı $[0,1]$ aralığındaki düzgün dağılıma ilişkin rastsal sayıları belirtir. $[0,1]$ aralığı dışındaki düzgün dağılımına sahip diğer rastsal sayılar *rastsal değişkenler* veya *stokastik değişkenler* olarak adlandırılır.

Rastsal Sayı Üretmek

Genel olarak, rastsal sayılar oluşturmak için kabul edilebilir bir yöntem, aşağıdaki özelliklere sahip sayı dizileri ortaya çıkarmalıdır:

- düzgün dağılıma sahip,
- istatistiksel olarak bağımsız,

- tekrar üretilebilir,
- dizinin istenen herhangi bir uzunluğu (periyod) için tekrarlanmayan.

Rastsal sayı üretmeye ilişkin geçmişe bakıldığında, ilk yöntemin John von Neuman'ın 1940'lı yıllarda ortaya koyduğu orta kare yöntemi olduğu düşünülebilir. Neuman'ın fikri, önceki rastsal sayının karesini almak ve oluşan sayı dizisinin ortasındaki rakamları diziden çıkarmaktı. Örneğin, 10 basamaklı sayılar üretildiğini ve önceki değerin 5772156649 olduğunu varsayıldığında, bu sayının karesi 33317792380594909291'dir ve yönteme göre seçilen sonraki rastsal sayı 7923805949 olacaktır. Buradaki sorun, yöntemin bir rastsal sayı dizisi ortaya çıkarıp çıkarmadığıyla ilgiliydi. Orta kare yöntemi nispeten yavaştı ve istatistiksel olarak tatmin edici değildi ve daha başarılı diğer algoritmaların geliştirilmesiyle kullanımına son verildi.

Simülasyon çalışmalarının önemli bileşenlerinden olan rastsal sayı ve rastsal değişken üreteçleri farklı yazılımlara gömülmüş algoritmalar olarak kullanılır. Çok sayıda uygulama bulunmaktadır ve eşleşik üreteçler, Tausworthe üreteci, gecikmeli Fibonacci üreteci veya Mersenne burgusu yaygın kullanılan üreteçlere örnek olabilir. Bir not olarak belirtilmesi gereken, bu üreteçlerdeki modül (mod—modulus) için 2^{32} ile 2^{64} arasında değerler kullanıldığıdır, bunun aksine, aşağıdaki örneklerde basit sayılar kullanılmıştır.

Yapay-rastsallık üreteçlerinin kullanımıyla ilgili vurgulanması gereken bir diğer konu üreteçlerin bir başlangıç değeri (seed) ile başlatıldığıdır. Bu başlangıç değeri aynı kaldığı sürece algoritmanın her çalışmasında aynı rastsal sayı dizisi üretilebilir.

R'da rastsal sayı üretmek için aşağıdaki örnek kullanılabilir:

```
# Ortalamasi 0 ve standart sapmasi 1 olan
# standart normal rastsal sayilar
x <- rnorm(10)
x
# Ortalamasi 20 ve standart sapmasi 2 olan
# standart normal rastsal sayilar
x <- rnorm(10, 20, 2)
x
# Temel istatistik
summary(x)
#
pnorm(2)
```

Eşleşik üreteç (congruential method), yapay-rastsal sayıların üretilmesinde kullanılan popüler bir yöntemdir. Temelde *karma* ve *çarpımsal* olarak iki türde kullanımı yaygın olsa da iki türün birlikte kullanımı da söz konusudur. Eşleşik üreteç 0 ve $m-1$ arasında rastsallık üretir ve 0 ile 1 arasındaki düzgün dağılıma sahip yapay-rastsal sayılar x_i'nin m'e bölümüyle bulunabilir.

Doğrusal eşleşik üreteç,

$$x_{i+1} = f(x_i, x_{i-1}, \dots) \ (mod\ m)$$

olarak yazılabilir ve çeşitli özel uygulamaları olan yöntemin en sık kullanılan şekli *karma eşleşik yöntem* olarak adlandırılır,

$$x_{i+1} = ax_i + c \ (mod\ m) \quad x_i, a, c, m \geq 0$$

Örneğin;

$$x_{i+1} = 2x_i + 7(mod 9) \quad \text{Başlangıç değeri, } x_0 = 1234$$

ile şu rastsal dizi oluşturulur: $\overline{073461}073461073461$ ...

Örneğe ilişkin R kodu aşağıda görülmektedir:

```
# Eslesik uretec fonksiyonu
eslesik_uretec <- function(a,c,m,adet,seed) {
    x <- rep(0,adet)
    x[1] <- seed
    for (i in 1:(adet-1)) {
      x[i+1] <- (a * x[i] + c) %% m
    }
    return(list(x=x))
}
# Fonksiyonun ornekteki parametreler ile calistirilmasi
z <- eslesik_uretec(2,7,9,20,1234)
z
```

Tausworthe üreteci, $m = 2$ olduğunda elde edilebilen ve x_i'nin 0 ya da 1 değerlerini alabildiği eşleşik bir üreteç olarak düşünülebilir. a_i'nin değerleri genellikle ikili sayı sistemi ile (0 ve 1) belirlenir.

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \cdots + a_n x_{i-n}) \; (mod\ 2)$$

Örneğin;

$$x_i = (1x_{i-1} + 1x_{i-2} + \cdots + 1x_{i-10}) \; (mod\ 2)$$

$$\text{Başlangıç değeri, } x_0 = 1234, a_i = 1 \text{ ve } n = 10$$

ile şu rastsal dizi oluşturulur: $\overline{0111111010}0111111010101111110101$ ...

Örneğe ilişkin R kodu aşağıda görülmektedir. (Örnek R kütüphanesi için bkz. SyncRNG)

```
# https://cran.r-project.org/web/packages/SyncRNG/
library(SyncRNG)
# Tausworthe ureteci
tausworthe_ureteci <- SyncRNG(seed=1234)
for (i in 1:10) {
  cat(tausworthe_ureteci$randbelow(2), '\n')
}
```

Gecikmeli Fibonacci üreteci (lagged Fibonacci generator, LFG), Fibonacci dizisine ilişkin temel üzerine kurulmuştur. Fibonacci dizisi, dizinin her bileşeninin kendinden önceki iki bileşenin toplamı olduğu bir yapıdadır ve başlangıç değerleri 1,1,2,3,5,8,13,21, ... gibidir.

$$x_n = x_{n-1} + x_{n-2} \quad x_0 = 0, x_1 = 1$$

Örneğin;

$$x_n = x_{n-3} + x_{n-5} (mod\ 8)$$

$$\text{Başlangıç değeri, } x_i = \{1,2,3,4,5\}, i = 5, m = 8 \text{ ve } 0 < j = 3 < k = 5$$

ile şu rastsal dizi oluşturulur: 12345460034634172433357602370 31 ... (Örnek R kütüphanesi için bkz. rTRNG)

Mersenne burgusu (Mersenne twister, MT), birçok üreteçten yüksek olan maksimum periyodu ile yüksek performansa sahip bir yapay-rastsal sayı üretecidir. Mersenne burgusu, üretilen sayıların tekrar etmeye başladığı periyod kadar büyük bir dizi üretir. Bu dizi 32 bitlik bloklar

halinde gruplanır. Bir örnek olarak, Mersenne burgusunun 32-bit çıktısının $2^{19937}-1$ sahip olması ile 00101110 00101110 00101110 00101110 ... gibi 1-bit'lik çıktısı olan herhangi bir üretecin 8'lik periyoda sahip olması karşılaştırılabilir. (Örnek R kütüphanesi için bkz. pseudoRNG)

Mersenne burgusu aşağıdaki gibi gösterilir:

$$x_{k+n} = x_{k+m} \oplus \left(x_k^u \mid x_{k+1}^k \right) A, k \geq 0$$

x ile ifade edilen her blok w bit'lik büyüklüğe sahiptir. Diğer parametrelerin temel açıklamaları aşağıda görülmektedir:

x_k^u	x_k'nın üst $w - r$ bit'i, $0 \leq r \leq w$.
x_{k+1}^l	x_{k+1}'in alt r bit'i.
$\oplus$	dışlamalı VEYA.
$\mid$	iki bit dizisinin ardaşık olarak birleştirilmesi.
n	yinelenme derecesi ilişkisi.
m	$1 \leq m \leq n$ aralığındaki tam sayı.
A	Yukarıdaki yinelemede çarpma işleminin hızlı yapılabilmesi için tanımlanmış sabit bir $w \times w$ matrisi.

Rastsal Sayı Üreteçlerine İlişkin İstatistik Testleri

Programlama dilleri ve diğer yazılım paketleri aracılığıyla kullanılabilen tüm rastsal sayı üreteçleri kabul edilebilir istatistiksel davranışa sahip olmadığından herhangi bir yapay-rastsal sayı üretecinin çıktısını kullanmadan önce istatistiksel olarak kontrol etmek gerekebilir.

Aşağıdaki ilk üç istatistiksel test (frekans testi, seri test ve otokorelasyon testi) bir bit (0 veya 1) dizisinin rastsallığını kontrol ederken, sonraki iki istatistiksel test (çalışma testi ve uyum iyiliği için ki-kare testi) $[0,1]$ aralığındaki yapay-rastsal sayıların rastsallığını kontrol eder. İstatistiksel test, hipotez olarak bilinen belirli bir varsayımın doğruluğunun kontrol edilmesini içerir.

Bir yapay-rastsal sayı üretecinin rastsallığının kontrol edilmesi durumunda, rastsal sayılardan oluşan bir diziye ihtiyaç duyulacaktır ve bu veriye dayanarak H_0'ın kabul edilip edilmeyeceğine karar vermek için hipotez testi kullanılır. Testin sonucu deterministik değil olasılığa dayalıdır ve H_0'ın belirli bir olasılıkla kabul ya da red edildiğini söyleyecektir.

Frekans testi (Monobit testi), gerçek bir ikili rastsal dizideki 1'lerin ve 0'ların sayısının yaklaşık olarak aynı olacağı varsayımına dayanır. Bunu kontrol etmek için frekans testi *tamamlayıcı hata fonksiyonunu* (*erfc*) kullanır. *erfc*, ortalaması 0 ve standart sapması $\frac{1}{\sqrt{2}}$ ile normal dağılıma sahip rastsal bir Y değişkeni için, $erf(x)$, Y'nin $[-x, x]$ aralığında olma olasılığıdır. *erfc* fonksiyonu şu şekilde yazılabilir:

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-u^2} du$$

Frekans testine ilişkin adımlar aşağıda örnekte gösterilmektedir, %99 anlamlılık düzeyinde $P \geq 0.01$ ise bit dizisi rastsal olarak kabul edilir:

Adım 1. Yapay-rastsal sayılar üretilerek uzunluğu 10 olan bir bit dizisi olarak birleştirilir. Dizideki 0'lar -1'lere dönüştürülerek dizideki sayılar toplanır.

$$\varepsilon = 1011010101 \text{ ve } \mathrm{n} = 10 \text{ ise}$$

$$\mathrm{S_n} = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2.$$

Adım 2. Test istatistiği hesaplanır.

$$\mathrm{s} = \frac{|2|}{\sqrt{10}} = 0.632455532$$

Adım 3. P değeri hesaplanır.

$$\mathrm{P} = \frac{0.632455532}{\sqrt{2}} = 0.52708926$$

Örnekte hesaplanan $P = 0.52708926 \geq 0.01$ olduğundan dizi rastsaldır.

k sayıda bit'i birleştirmenin 2^k farklı yolu vardır. k bit dizisi rastsal ise, bu kombinasyonların her birinin gerçekleşme şansı aynıdır. Seri test, bu kombinasyonların her birinin meydana gelme sayısının düzgün dağılıma uyumunu belirler.

Seri test, e bir yapay-rastsal sayı üreteci tarafından oluşturulan n bit'lik dizi olduğunda (n için önerilen minimum değer 100'dür) e'de bulunan $\mathrm{k} < \lfloor \log_2 \mathrm{n} \rfloor - 2$ durumundaki $\mathrm{k}, \mathrm{k}-1, \mathrm{k}-2$ bitlik çakışan blokların rastsallığını kontrol eder.

Otokorelasyon testinin odak noktası, e'nin bir üreteç tarafından oluşturulan n bit'lik dizi olduğu kabul edildiğinde, e'nin, bit'lerin konumu $\mathrm{d_i}$'ye kaydırılarak elde edilen diğer bir diziden farklı olacağıdır. Örneğin, $\mathrm{e} = 1001101$ ile $\mathrm{d_1} = 0011011, \mathrm{d_2} = 0110110, \mathrm{d_3} = 1101100$ arasındaki farklılıklar rastsallığın test edildiği hipotez testi için kullanılacaktır.

Tekrarlar testi, ikili bir rastsal dizideki 1'lerin ve 0'ların çeşitli uzunluktaki ardışık tekrarlarının salınımını kontrol etmek için kullanılır. Dizi *frekans testi*ni geçtiyse, *tekrarlar testi*ne gerek olmayacaktır.

Dizideki 1'lerin test-öncesi π değeri hesaplanır.

$$\pi = \frac{\sum_j \varepsilon_j}{n}$$

Örneğin;

$$\varepsilon = 1001101011 \text{ ve } n = 10 \text{ ise}$$

$$\pi = \frac{6}{10}$$

Uyum iyiliği için Ki-kare testi, $[0,1]$ aralığındaki yapay-rastsal bir sayı dizisinin düzgün dağılıma sahip olup olmadığını kontrol eder. Test, genelde ampirik bir dağılımın belirli bir teorik dağılıma uyup uymadığını kontrol etmek için kullanılır.

$k > 100$ olacak şekilde eşit uzunlukta ve k sayıda alt-aralığa bölünmüş $[0,1]$ aralığındaki yapay-rastsal bir sayı dizisinde i'ninci alt aralığa düşen yapay-rastsal sayı miktarı, $\mathrm{f_i}$, *gözlenen değerler* olarak adlandırılır. Bu alt-aralıkların yeterli sayıda gözlem içerdiği ($\mathrm{f_i} > 5$) ve n'in örneklem büyüklüğü olduğu kabul edildiğinde, her alt-aralığa düşen rastsal sayı miktarının

ortalamasının $\frac{n}{k}$ (teorik değer) olması üretilen rastsal sayı dizisinin *gerçek* düzgün dağılıma sahip olduğu anlamına gelmektedir.

Ki-kare testi, gözlenen ve teorik değerler arasındaki farkın rastsal dalgalanmalardan mı yoksa ampirik dağılımın belirli teorik dağılımı takip etmemesinden mi kaynaklandığını ölçer.

Teorik dağılımın düzgün dağılım olduğu durumda ve $k-1$ serbestlik derecesinde Ki-kare istatistiği şu şekildedir:

$$\chi^2 = \frac{k}{n}\sum_{i=1}^{k}\left(f_i - \frac{n}{k}\right)^2$$

Üretilen rastsal sayıların $[0,1]$ aralığında düzgün dağılıma sahip olduğuna ilişkin hipotez, H_0, $k-1$ serbestlik derecesi ve α anlamlılık düzeyindeki χ^2'nin değeri, Ki-kare tablolarından elde edilen değerden büyükse reddedilir.

Olasılık Dağılımlarından Örnek Almak

Belirli bir olasılık dağılımına uyan yapay-rastsal sayılardan örnek almak, sayılar üretildikten sonra bir ya da birden fazla rastsal değişken (r) kullanılarak bu olasılık dağılımından seçildiği varsayılan yeni bir rastsallık (x) üretmekle ilgilidir.

Ters Dönüşüm Yöntemi

Ters dönüşüm yöntemi, belirli bir olasılık yoğunluk fonksiyonundan stokastik değişkenler elde etmek için kullanılmaktadır ve kümülatif yoğunluk fonksiyonunun analitik olarak ters dönüşüme uygun olduğu durumlar için geçerlidir. $f(x)$'in $[0,1]$ aralığında tanımlı kümülatif yoğunluk fonksiyonu olan $F(x)$ stokastik değişkenler üretmek için kullanılır.

$F(x)$'in rastsal olduğu kabul edildiğinde, $F(x) = r$ eşitliği, rastsallık ile bu rastsallık sonucunda oluşan olaylar arasındaki ilişkiyi tanımlamak için kullanılır. Belirli bir rastsal süreç içeren olayın araştırılması, diğer bir deyişle r ile x arasındaki ilişkinin tanımlanması, $F(x)$ fonksiyonunun ters dönüşüm yöntemiyle işlenmesi ile mümkün olmaktadır ve $x = F^{-1}(r)$ eşitliği ile gösterilir.

Örneğin, olasılık yoğunluk fonksiyonu,

$$f(x) = 2x, \quad 0 \le x \le 1$$

olan bir olaylar zincirinin kümülatif yoğunluk fonksiyonu,

$$F(x) = \int_0^x 2t\,dt = x^2, \quad 0 \le x \le 1$$

şeklinde tanımlanabilir ve r ile x arasındaki ilişkiyi

$$r = x^2 \quad \text{veya} \quad x = \sqrt{r}$$

olarak gösterir.

Ters dönüşüm yöntemi, sürekli ya da kesikli olasılık dağılımlarından örnekler almak ve simülasyon modelindeki sistemlerin çıktılarının rastsallığını ortaya çıkarmak için kullanılır.

Sürekli Olasılık Dağılımları

Bu bölümde düzgün dağılımdan, üstel dağılımdan, Erlang dağılımından ve normal dağılımdan örnek almakla ilgili bilgi verilmektedir.

Düzgün dağılımın olasılık yoğunluk fonksiyonu şu şekilde tanımlanabilir:

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & a \le x \le b \\ 0, & x < a \; veya \; x > b \end{cases}$$

Kümülatif yoğunluk fonksiyonu,

$$F(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & x > b \end{cases}$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$r = F(x) = \frac{x-a}{b-a}$$

veya

$$x = a + (b-a)r$$

olacaktır.

Üstel dağılımın olasılık yoğunluk fonksiyonu şu şekilde tanımlanabilir,

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0 \\ 0, & x < 0 \end{cases}$$

Kümülatif yoğunluk fonksiyonu:

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x}, & x \ge 0 \\ 0, & x < 0 \end{cases}$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$r = F(x;\lambda) = 1 - e^{-\lambda x}$$

veya

$$x = \frac{-\ln r}{\lambda}$$

olacaktır.

Erlang dağılımının olasılık yoğunluk fonksiyonu şu şekilde tanımlanır,

$$f(x;k,\lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} \qquad x \ge 0, \lambda \ge 0$$

Kümülatif yoğunluk fonksiyonu,

$$F(x;k,\lambda) = 1 - \sum_{n=1}^{k-1} \frac{1}{n!} e^{-\lambda x} (\lambda x)^n$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$x = E(k,\lambda) = -\frac{1}{\lambda} \ln \prod_{i=1}^{k} r_i$$

olacaktır.

Normal dağılımın olasılık yoğunluk fonksiyonu şu şekilde tanımlanır,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Kümülatif yoğunluk fonksiyonu,

$$F(x) = \Phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$r = \frac{x-\mu}{\sigma}$$

veya

$$x = \mu + r\,\sigma$$

olacaktır.

Kesikli Olasılık Dağılımları

Bu bölümde geometrik dağılım, Binom dağılımı ve Poisson dağılımı ile ilgili bilgi verilmektedir.

Geometrik dağılımın olasılık kütle fonksiyonu, ilk başarılı deneyden önce x sayıda başarısızlık olasılığı olarak, şu şekilde tanımlanır:

$$P(Y = x) = pq^x$$

p başarı olasılığı,

q başarısızlık olasılığı.

Kümülatif dağılım fonksiyonu,

$$F(x) = \sum_{s=0}^{x} p\,q^s = p\,\frac{1-q^{x+1}}{1-q}$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$r = q^x$$

veya

$$x = \frac{\log r}{\log q}$$

olacaktır.

Binom dağılımının olasılık kütle fonksiyonu şu şekilde tanımlanır,

$$f(k,n,p) = P(k;n,p) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Kümülatif dağılım fonksiyonu,

$$F(k;n,p) = \Pr(X \le k) = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i}$$

ise rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$k_i = \begin{cases} k_{i-1} + 1, & r_i < p \\ k_{i-1}, & r_i > p \end{cases}$$

olacaktır.

Poisson dağılımı, belirli bir zaman aralığında oluşan ortalama olay sayısının λ olduğu kabul edildiğinde, bir birim zaman aralığında k sayıda olayın oluşmasının modellenmesi için kullanılır ve aşağıdaki olasılık kütle fonksiyonuna sahiptir.

$$f(k;\lambda) = \Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

k olay sayısı ($k = 0,1,2, \ldots$)

e Euler sayısı ($e = 2.71828 \ldots$)

Rastsal süreçler üretmek için ters dönüşüm yöntemi ile,

$$t_i = \frac{-1}{\lambda} \log r_i$$

olacaktır.

Reddetme Yöntemi

Kabul-red yöntemi veya *kabul-red algoritması* olarak da adlandırılan **reddetme yöntemi** (rejection method), belirli bir dağılımdan örneklem almak ve gözlemler oluşturmak için kullanılan temel bir simülasyon yöntemidir. Yöntem, belirli bir yoğunluğa sahip $\mathbb{R}^m$'deki herhangi bir dağılım için kullanılabilir. Reddetme yöntemi, olasılık fonksiyonunun, $f(x)$, sınırlı ve sonlu olduğu ($a \leq x \leq b$) varsayıldığında rastsal değişkenler üretmek için kullanılabilir.

Reddetme yöntemi, iki-boyutlu Kartezyen koordinat sisteminde düzgün dağılıma sahip bir tek-boyutlu rastsal değişken için yapılan gözleme ve örneklemin yoğunluk fonksiyonunun grafiğinin altındaki alanda oluşacağı varsayımına dayanır. Yöntem çeşitli örnekler için N-boyutlu olarak genişletilebilir.

Yöntem şu şekilde çalışmaktadır:

- $f(x)$ belirli bir c parametresiyle normalize edilir, $cf(x) \leq 1$.
- x, r'nin rastsal sayı olduğu bir doğrusal bir fonksiyon olarak tanımlanır, Örn. $x = a + (b - a)r$.
- Rastsal sayı çiftleri üretilir, (r_1, r_2).
- Bu çiftler $r_2 \leq cf(a + (b - a)r_1)$ ilişkisini sağlıyorsa, Örn. $x = a + (b - a)r_1$ eşitliği rastsal değişken olarak kullanılır.

Monte Carlo Simülasyonu

Monte Carlo yöntemi veya *çoklu olasılık simülasyonu* olarak da bilinen **Monte Carlo simülasyonu**, sonucu belirsiz bir olayın olası sonuçlarını tahmin etmek için kullanılan matematiksel bir tekniktir. Monte Carlo simülasyonları yapay zeka, hisse senedi fiyatları, satış tahmini, proje yönetimi ve fiyatlandırma gibi birçok gerçek hayat senaryosunda riskin etkisinin değerlendirilmesine imkan tanır. Ayrıca, karar vericinin girdinin belirli bir çıktıya etkisini değerlendirebileceği duyarlılık analizi veya değişkenlerin ilişkisinin anlaşılabileceği girdi korelasyonunun hesaplanması ile sabit girdili tahmine dayalı modellere göre avantaj sağlar.

Monte Carlo simülasyonu, tipik bir tahmin modelinden farklı olarak, bir dizi sabit girdiye bağlı tahmini bir değer aralığına dayalı bir dizi sonucun tahmin edilmesini sağlar. Diğer bir deyişle, Monte Carlo simülasyonu doğası gereği belirsizliğe sahip herhangi bir değişken için olasılık dağılımlarından yararlanarak olası sonuçların bir modelini oluşturur ve minimum ile maksimum değerler arasında farklı bir rastsal sayı kümeleri kullanarak sonuçların tekrar hesaplanmasına izin verir. Tipik bir Monte Carlo deneyinde bu süreç çok sayıda olası sonuç üretmek için tekrarlanabilir. Monte Carlo simülasyonu, tahmindeki doğruluk başarısı nedeniyle uzun vadeli tahminler için de kullanılmaktadır. Girdi sayısı arttıkça, tahminlerin sayısının artması sonuçların daha doğru bir şekilde ileriye doğru yansıtılmasına olanak tanır.

Monte Carlo simülasyonunun basit bir örneği olarak, atılan iki standart zarın sonuçlarına ilişkin olasılığın hesaplaması düşünülebilir ve olası 36 sonuç kombinasyonu vardır. Buna dayanarak, belirli bir sonucun olasılığı hesaplanabilir ve tekrarlanan her deney bu sonucun teorik değere yaklaşmasına neden olacaktır.

Monte Carlo simülasyonuna ilişkin süreç şu adımlardan oluşur:

- Tahmin edilecek bağımlı değişken ile tahmini yönlendirecek bağımsız değişkenlerin (başlangıç değişkenleri, risk değişkenleri veya tahmin değişkenleri olarak da adlandırılır) belirlenerek tahmin modelinin oluşturulması.
- Geçmiş veriler veya analistin öznel yargısı ile bağımsız değişkenlerin olasılık dağılımlarının belirlenmesi.
- Bağımsız değişkenlere ilişkin rastsal değerlerin üretilerek simülasyonun çalıştırılması.
- Simülasyonun istatistiki anlamlılık ifade eden yeterli sonuç toplanana kadar tekrarlanması.

Diğer yandan, **Monte Carlo yöntemi**, tekrarlanan rastsal değerlerden belirli sonuçlar çıkarabilmek için kullanılan sayısal bir yöntemdir ve hem fiziksel hem de matematiksel sistemlerin simülasyonu için kullanılmaktadır. Kesin sonuç verecek bir deterministik algoritma ile hesaplama yapılamayacak durumlarda tercih edilmektedir. Monte Carlo yöntemi sistemlerin simülasyonları sırasında rastsal değerlerin özelliklerinin belirlenmesi sırasında kullanılmaktadır.

Monte Carlo yöntemi temel olarak şu aşamalardan oluşur:

- Olası girdilerin tanımlanması,
- Olası girdilerden rastsal olarak değerlerin seçilmesi ve deterministik hesaplamanın yapılması,
- Bağımsız hesaplamaların sonuçlarının birleştirilerek sonuca ulaşılması.

Monte Carlo'nun nasıl tanımlanması gerektiği konusunda fikir birliği yoktur. Bazı araştırmacılar stokastik simülasyon olarak tanımlarken diğerleri matematiksel veya istatistiksel bir problemi çözmek için kullanılabilecek bir yöntem olarak tanımlamaktadır. Örneğin, atomlardan radyasyon emisyonu doğal bir stokastik süreçtir ve doğrudan simüle edilebilir veya sürecin ortalama davranışı Monte Carlo yöntemleri kullanılarak çözülebilen stokastik denklemlerle tanımlanabilir. (Örnek R kütüphanesi için bkz. MonteCarlo)

Kesikli Simülasyon

Kesikli simülasyon (ayrık olay simülasyonu), bir sistemin işleyişinin zaman akışı boyunca birbirinden ayrı (ayrık) olaylar dizisi olarak modellemek için kullanılır. Her olay belirli bir zamanda meydana gelir ve sistemdeki bir durum değişikliğini gösterir. Birbirini izleyen olaylar arasında, sistemde hiçbir değişikliğin meydana gelmediği varsayılır ve bu nedenle simülasyon

zamanı, *sonraki-olay zaman ilerlemesi* olarak tanımlandığı gibi, bir sonraki olayın meydana gelme zamanına doğrudan atlayabilir.

Sonraki-olay zaman ilerlemesine ek olarak, *sabit-artışlı zaman ilerlemesi* adı verilen, zamanın küçük zaman dilimlerine bölündüğü ve sistem durumunun bu dilimlerde meydana gelen olaylar kümesine göre güncellendiği alternatif bir yaklaşım da vardır. Sonraki-olay zaman simülasyonunda her bir zaman diliminin simüle edilmesi gerekmediğinden, sabit-artışlı-zaman simülasyonundan çok daha hızlı çalışır.

Kesikli simülasyonunun yukarıda belirtilen her iki türü de, sistem durumunun, durum değişkenlerinin değişim oranlarını tanımlayan diferansiyel denklemlerin akış zamanı boyunca sürekli olarak değiştiği *sürekli simülasyon* ile tezat oluşturur.

Belirli bir önceliğe sahip olayların gerçekleştiği kesikli simülasyona ilişkin bileşenler rastsal sayı üreteçlerinin ve sistemin istatistiki özelliklerinin yanında durum, zaman, olay ve sonuçlandırma kriteri olarak düşünülebilir.

Durum

Sistem durumu, incelenecek sistemin önemli özelliklerini tanımlayan bir dizi değişkendir. Belirli bir zamandaki durum, $S(t)$, bir olay meydana geldiğinde değeri değişebilen bir basamak fonksiyonu (step function) ile tanımlanır.

Zaman

Simülasyon, modellenen sistemdeki ölçü birimleri ne olursa olsun mevcut simülasyon zamanını takip etmelidir. Kesikli simülasyonlarda, sürekli simülasyonların aksine, olaylar anlık olduğu için simülasyon ilerledikçe saat bir sonraki olayın başlangıç zamanına atlar. Simülasyon ilerledikçe, belirli koşullarda zaman değişkenlerinin kesinliğini kaybettiği durumlar ortaya çıkabilmekte ve tekrar-normalleştirilmeleri (re-normalization) gerekmektedir.

Olay

Simülasyon, en az bir *simülasyon olayları listesi* içerir. Bu, daha önce simüle edilmiş herhangi bir olayın sonucunu bekleyen ancak henüz simüle edilmemiş olayları listelediği için *bekleyen olay kümesi* olarak adlandırılır. Bir olay, meydana geldiği zamanla ve o olayın simülasyonu için kullanılacak parametrik bir kodla tanımlanır. Olaylar, anlık olduğundan, zamana yayılan olay dizileri olarak modellenir ancak bazı simülasyonlarda belirli bir olayın gerçekleşmesine ilişkin başlangıç ve bitiş zamanları kullanılarak olay anının bir zaman aralığı olarak belirlenmesi mümkündür.

Aynı anda tek bir olayın gerçekleştiği varsayımına dayalı *tek-çevrimli simülasyon motorları*nın aksine, *çok-çevrimli simülasyon motorları* aynı anda birden fazla sayıda olayın gerçekleşebileceğini varsayar. Bekleyen olay kümesi, olayların gerçekleşme zamanlarına göre sıralanmış bir *öncelik sırası*na sahiptir.

Sonuçlandırma kriteri

Olayların gerçekleşme özellikleri belirli olduğundan kesikli simülasyon —teorik olarak— sonsuza kadar çalışabilir. Bu nedenle, simülasyonun ne zaman sona erdirileceğine ilişkin karar, t anında, n sayıda olay gerçekleştikten sonra veya X istatiksel ölçütü x değerine ulaştığında gibi çeşitli kriterlere bağlı olarak verilebilir.

Simülasyonlar, karmaşık sistemlerdeki sorunların teşhis edilmesine yardımcı olmak için etkin araçlardır ve kesikli simülasyona ilişkin uygulamalara bakıldığında yaygın bir kullanım alanı olduğu görülebilir.

Süreç yönetimindeki önemli konulardan biri olarak *kısıtlar teorisi* (theory of constraints), bir sistemdeki darboğazların nedenlerinin anlaşılabilmesi ve bunları engellenebilmesi için, süreçlerdeki sorunların teşhis edilmesi gerektiğinden bahseder. Darboğazların belirlenmesi ve ortadan kaldırılması, süreçlerin ve bu süreçlerden oluşan genel sistemin iyileştirilmesine olanak tanır. Örneğin, üretim işletmelerindeki darboğazlar, fazla stok, aşırı üretim, süreçlerdeki veya rotalamadaki değişkenlik nedeniyle oluşabilir. Kapasite kullanımı, zamanında teslimat oranı, hurda ya da atık oranı, nakit döngüsü gibi performans göstergelerini içeren sistem, bir simülasyon modeliyle tanımlandıktan sonra detaylı analizi yapılarak karar vericinin sistemin performans göstergelerini anlamasını ve iyileştirmesini sağlayabilir.

Hastane uygulamalarına örnek olarak bir ameliyathanenin birkaç cerrahi disiplin arasında paylaşıldığı durum verilebilir. Bu sürecin doğasını daha iyi anlayarak verimi artırmak simülasyon ile mümkün olabilir. Çoğu sistem iyileştirme fikri, yalın süreçler, altı sigma veya toplam kalite yönetimi gibi sağlam ilkeler ve kanıtlanmış metodolojiler üzerine kuruludur ancak belirli durumlarda sistemi iyileştirmede başarısız olabilir. Bir simülasyon modeli, kullanıcının genel sistem bağlamında bir performans iyileştirme fikrini anlamasını ve test etmesini sağlar. Sermaye yatırım kararlarının değerlendirilmesinde kullanılan simülasyon potansiyel yatırımları modellemek için yaygın olarak kullanılır. Kesikli simülasyon, gerçek dağıtımdan önce yeni protokolleri veya farklı sistem mimarilerini (dağıtılmış, hiyerarşik, merkezi, P2P) simüle etmek için bilgisayar ağlarında kullanılır. Bu tür simülasyonlarda hizmet süresi, bant genişliği, transfer edilen bilgi paketleri, kaynak tüketimi vb. gibi farklı değerlendirme ölçütleri tanımlamak mümkündür.

Kesikli Simülasyonun İstatiksel Analizi

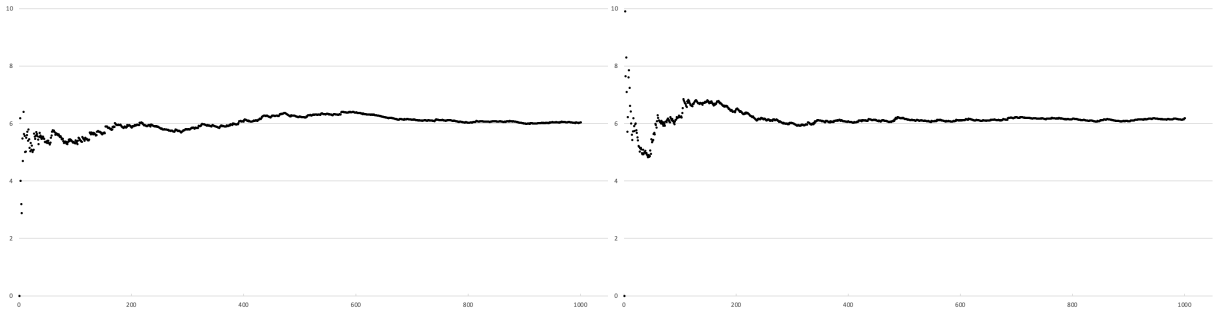
Bir simülasyon çalışmasının önemli bir bileşeni, çıktının, diğer bir deyişle simülasyon modelinden elde edilen verinin, istatistiksel analizidir. Bu analiz, bir simülasyon modeli geliştirmenin amacı olan, modele ilişkin çeşitli performans göstergelerinin tahmin edilmesi olarak görülebilir. Simülasyon modelinin belirli bir zamandaki durumu sisteme ilişkin tahmini etkileyecektir.

Bir simülasyon, modele ilişkin sistemin $t = 0$'da *başlangıç durumu* (initial condition) olarak bilinen belirli bir durumda olduğunu varsayarak çalışmaya başlar. Simülasyonun istatiksel davranışı bu başlangıç durumundan $t = T$'ye kadar devam eden *geçici durum* (transient state, transient period) olarak adlandırılabilecek belirli bir süre için etkilenecektir. Simülasyon zamanı ilerledikçe ($t > T$) veya simülasyonun tekrar sayısı arttıkça simülasyonun istatistiki davranışının başlangıç durumundan bağımsız hale geldiği ve *durağan (sabit, kararlı) durum* (steady state) olarak bilinen ve kararlı istatistiki davranışa sahip bir durumda olduğu görülecektir.

Bir simülasyonun başlangıç durumunun belirlenmesi için temelde iki strateji kullanılabilir. İlk strateji simülasyona boş bir sistemle diğer bir deyişle başlangıç durumunda sistemde devam eden bir işlemin olmadığını varsayarak başlamaktır. İkinci strateji ise başlangıç durumunun sistemin durağan durumda oluşturduğu istatistiği temsil etmesini sağlamaktır. Bu strateji simülasyonun geçici dönemini kısaltabilse de analist açısından sisteme ilişkin önceden bilgi sahibi olmayı gerektirir.

Durağan durumdaki bir sistem modelinin simülasyonunun istatistiki analizinin yapılması bu sisteme ilişkin performans göstergelerinin belirli bir olasılıkla tahmin edilmesini sağlar. Diğer yandan, bu analiz, sistemin istatistiki davranışı başlangıç durumuyla yakından ilişkili olduğundan geçici durumdaki bir sistem modelinin simülasyonu için, belirli başlangıç durumu değişikliklerinin —geçici durumdaki— sistemin çalışmasını nasıl etkileyeceğinin incelenmesi dışında, anlamlı görünmemektedir.

Simülasyon çalışmalarında belirli bir sistemin —doğal olarak— durağan durumdaki istatistiki davranışı incelenir. Bu inceleme, simülasyon modelinin yeterince uzun bir zaman çalıştıktan sonra geçici durumdan ayrılmış olmasını gerektirir. Diğer yandan geçici durumla durağan durum arasındaki bu sınır belirli ya da kesin değildir (bkz. Şekil 5).



a. Birinci iterasyon. b. İkinci iterasyon.

Şekil 5. Bir mağazaya, geliş oranı, $\lambda = 10/saat$ ile gelen müşterilerin simülasyon modelinde hesaplanan ortalama gelişlerarası zamana ilişkin iterasyonlara ait grafikler.

Belirli bir simülasyon modeli sisteminin geçici durumdan ne zaman ayrıldığının ya da durağan duruma ne zaman eriştiğinin belirlenmesi için çeşitli yöntemler kullanılır. İlk yöntem simülasyonun çok uzun bir süre çalıştırılarak çıktının istatistiki özelliklerinin belirlenmesidir. Bu çalışmada toplanması gereken büyük miktarda veri olacaktır. İkinci yöntem simülasyonun durağan duruma kadar veri toplanmadan çalıştırılması ve belirlenen sınırdan sonraki veriyi istatiksel analiz için kullanılmasıdır. İkinci yöntemde, simülasyona ilişkin birbirinden farklı $T_1, T_2, \dots, T_k$ geçici dönemler istatiksel olarak test edilmekte ve $T_1 < T_2 < \cdots < T_i < \cdots < T_k$ ve şeklinde sıralanmaktadır. T_i'den sonraki aralıklar için durağan durum istatistikleri önemli ölçüde değişmeyecektir. Üçüncü yöntem olarak, simülasyonun çıktısına ilişkin hareketli ortalamanın zamanla önemli ölçüde değişmediği bir sınırın tespit edilmesidir.

Simülasyon Modelinin Geçerliliğinin Doğrulanması

Bir simülasyon modelinin doğrulanması (validation) genellikle ihmal edilen önemli bir konudur ve "Bir simülasyon modeli gerçek sistemin operasyonlarını ne kadar doğru şekilde yansıtır?" veya "Elde edilen simülasyon sonuçlarının doğru ve anlamlı olduğundan ne kadar emin olunabilir?" gibi sorulara yanıt verilebilmesini sağlar. Bir simülasyon modeli, sonuçları araştırılan sistemden elde edilen gerçek verilerle karşılaştırılarak doğrulanabilir.

Bir simülasyon modelini doğrulamak için simülasyona ilişkin bazı kontroller yapılabilir. Bu kontrol çalışmaları aşağıdakileri içerecek şekilde planlanır:

- *Yapay-rastsal sayı üreteçleri*nin kontrol edilmesi, sayıların $(0,1)$ aralığında eşit olarak dağılmış olmasını ve istatistiksel bağımsızlık kriterlerini karşılamasını sağlar.
- *Stokastik değişken üreteçleri*nin kontrol edilmesi, simülasyon modelinde kullanılan stokastik değişken üreteçlerinin benzer istatistiksel testlerden geçirilmesini sağlar.
- *Simülasyonun çalışma mantığı*nın kontrol edilmesi, olası mantıksal hataları belirleyebilmek için simülasyonda her bir olay gerçekleştiğinde durum değişkenlerinin, simülasyon olay listesinin ve diğer ilgili verinin uygun şekilde güncellenip güncellenmediğinin kontrol edilmesini sağlar.
- *İlişkilerin geçerliliği*nin kontrolü modele ve bileşenlerine ilişkin varsayımların güvenilir olup olmadığını kontrol edilmesini sağlar.

- *Çıktının geçerliliği*nin kontrolünde, —eğer mevcutsa— sistemle ilgili gerçek veriler simülasyon modelinden elde edilen çıktılarla karşılaştırılır.

Uygulama

Monte Carlo simülasyonu, kullanılan istatistiksel yöntemlerin performansını incelemek için de kullanılmaktadır. Araştırmacılar, bazı sorulara yanıt verebilmek için simülasyonlardan yararlanır: (1) hatalar heteroskedastik ise sıradan en küçük kareler regresyonu (ordinary least squares, OLS) nasıl performans gösterir? (2) Eksik verilerin varlığı, bir eğilim skoru (propensity score analysis) analizinden elde edilen tahminleri nasıl etkiler? (3) Kümeler küçük olduğunda varyans tahmini nasıl çalışır? Bu tür soruları yanıtlamak için, yapay-rastsal sayılardan örneklemeye dayalı binlerce veri kümesi simüle edilebilir. Simülasyon çalışmalarından elde edilen performans ölçümleri, sonlu sayıda tekrarlanan örneğe dayanan tahminlerdir ve bu nedenle bazı rastsal hatalar içerir. Performans ölçülerini yorumlarken, tahmin hatasının ne kadar büyük olduğunu dikkate almak önemlidir. Bu tipik olarak Monte Carlo standart hatası (Monte Carlo standard error, MCSE) olarak ölçülür.

Bu uygulamada çeşitli performans ölçülerini ve ilişkili fonksiyonlar kullanarak bir veri kümesindeki benzer örneklem büyüklüğü olan standart sapma değerleri farklı iki ayrı veri grubuna ait *p-değeri* ve *MCSE değeri* hesaplanmaktadır.

Bu tür simülasyon çalışmaları genel olarak aşağıdaki adımları içerir:

1. Adım Deneysel tasarımın oluşturulması
2. Adım Veri setinin oluşturulması
3. Adım Veri seti üzerinde istatistiksel yöntemlerin kullanılması.
4. Adım 2. ve 3. adımların defalarca çalıştırılarak kullanılan istatistiki yöntemlerin performansının özetlenmesi.
5. Adım Deneysel tasarımda belirtilen tüm durumlar için 4. adımın tekrar edilmesi.
6. Adım Genel sonuçların ortaya çıkarılması.

Deneysel tasarım

Simülasyon çalışmasına başlamadan önce, gözlemi yapılacak model ile örneklem büyüklüğü, kayıp ya da hatalı verinin tüm veri kümesi içindeki oranı ve varyans gibi çeşitli tasarım parametrelerinin belirlenmesi gerekir.

Veri seti

Simülasyon verisinin oluşturulmasına ilişkin aşağıdaki fonksiyon model parametrelerini içerir. Böylece farklı parametreler ile veriler üretilebilir.

```
generate_dat <- function(model_params) {

  return(dat)
}
```

İki veri grubu için rastsal normal verilerin oluşturulduğu aşağıdaki örnekte ikinci veri grubuna ilişkin standart sapma değeri ilk grubun standart sapma değerinin iki katıdır. Fonksiyon, üç bağımsız değişkeni parametre olarak kullanır: *n1*, birinci veri grubu ve *n2,* ikinci veri grubu

için örneklem büyüklüğünü ifade ederken *mean_diff* grup ortalamaları arasındaki farkı göstermektedir.

```
generate_dat <- function(n1, n2, mean_diff){

  dat <- tibble(
    y = c(rnorm(n = n1, mean_diff, 1), # ortalama mean_diff,
st.sap. 1
          rnorm(n = n2, 0, 2)), # ortalama 0, st.sap. 2
    group = c(rep("Group 1", n1), rep("Group 2", n2))
  )

  return(dat)

}
```

Fonksiyon oluşturduktan sonra test edilerek doğru çalışıp çalışmadığını kontrol edilir. Aşağıda, her grup için örneklem büyüklüğünün *10.000* kişi olduğu ve *mean_diff* değeri *1* olarak belirlenmiş bir veri kümesi oluşturulmaktadır.

```
set.seed(1234) # baslangic degeri
example_dat <- generate_dat(n1 = 10000, n2= 10000, mean_diff
= 1)

example_dat %>%
  head()
```

```
## # A tibble: 6 × 2
##        y group
##    <dbl> <chr>
## 1 -0.207 Group 1
## 2  1.28  Group 1
## 3  2.08  Group 1
## 4 -1.35  Group 1
## 5  1.43  Group 1
## 6  1.51  Group 1
```

Şekil 6. Veri kümesinin ilk satırlarını gösteren ekran görüntüsü.

Aşağıda oluşturulan özet tablosunda ortalama değer birinci grup için *1*'e ve ikinci grup için *0*'a yakındır. Standart sapma değeri ise birinci grup için 1'e ve ikinci grup için 2'ye yakındır. Bu değerler daha önceki modele ilişkin fonksiyonda tanımlandığı gibi oluşmuştur.

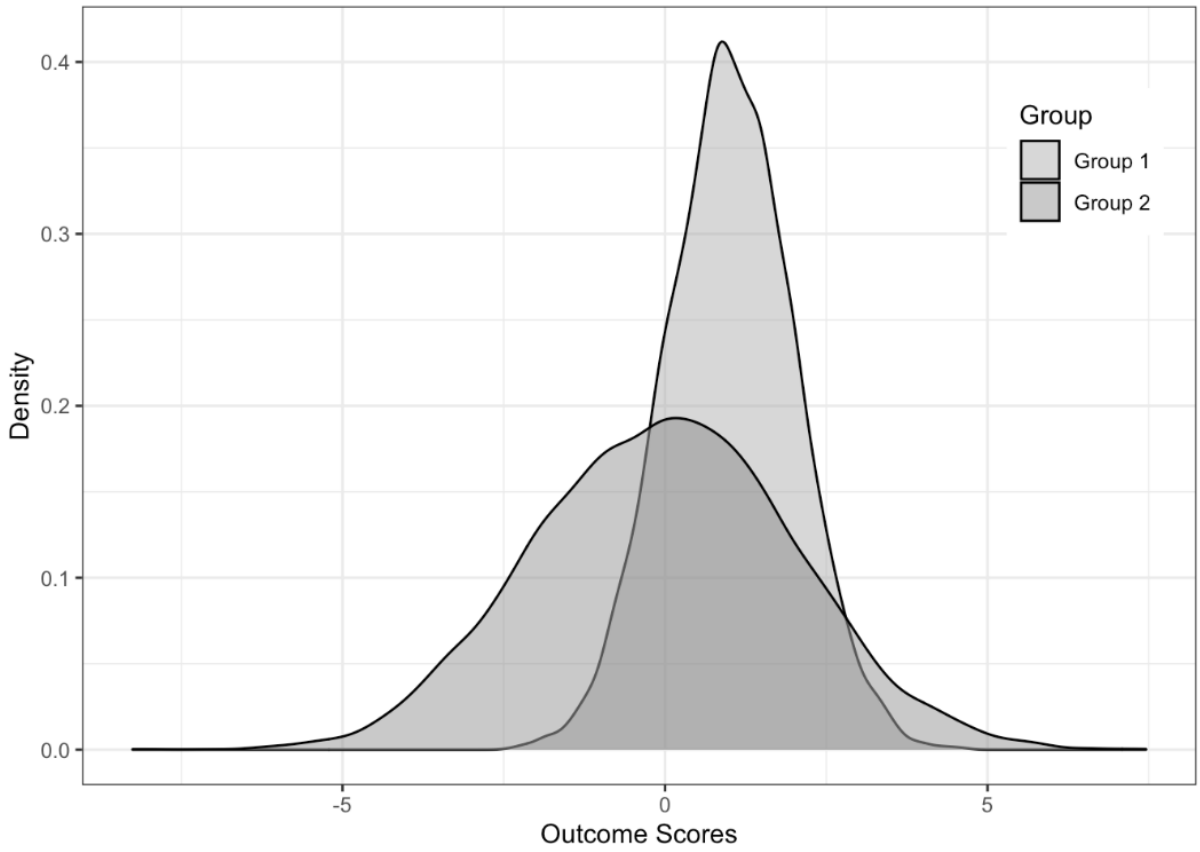
```
example_dat %>%
  group_by(group) %>%
  summarize(n = n(),
            M = mean(y),
            SD = sd(y)) %>%
  kable(digits = 3)
```

group	n	M	SD
Group 1	10000	1.006	0.988
Group 2	10000	-0.015	2.012

Şekil 7. Veri kümesindeki iki gruba ait örneklem sayısı (*n*) ile ortalama (*M*) ve standart sapma (*SD*) değerlerini gösteren ekran görüntüsü.

Aşağıda her grup için oluşturulan değerlere ilişkin yoğunluk grafiğini görülmektedir. Her iki gruba ilişkin dağılımlar normal dağılım ve zirveleri arasındaki fark *1*'e yakınken varyansların daha önce belirtildiği gibi farklı olduğu görülmektedir.

```
ggplot(example_dat, aes(x = y, fill = group)) +
  geom_density(alpha = .5) +
  labs(x = "Outcome Scores", y = "Density", fill = "Group")
+
  theme_bw() +
  theme(legend.position = c(0.9, 0.8))
```



Şekil 8. Gruplar için oluşturulan değerlere (outcome scores) ilişkin yoğunluk (density) grafiği.

İstatistik

Bu adımda, test istatistikleri, regresyon katsayıları, *p*-değerleri veya güven aralıkları hesaplanmaktadır. Fonksiyon, veriyi ve tahmine ilişkin tasarım parametrelerini içermektedir.

```
estimate <- function(dat, design_params) {

  return(results)
}
```

Aşağıda, t-testini simüle edilmiş bir veri kümesinde çalıştıran örnek fonksiyon bulunmaktadır. Fonksiyon, varyansın homojen olduğunu varsayan geleneksel bir t-testi ve iki veri grubu için popülasyon varyanslarının eşit olmadığını varsayan bir Welch t-testini çalıştırmaktadır.

```
# t and p value
calc_t <- function(est, vd, df, method){

  se <- sqrt(vd)  # standard error
  t <- est / se # t-test
  p_val <-  2 * pt(-abs(t), df = df) # p value
  ci <- est + c(-1, 1) * qt(.975, df = df) * se # confidence
interval

  res <- tibble(method = method, est = est, p_val = p_val,
                lower_bound = ci[1], upper_bound = ci[2])

  return(res)
}


estimate <- function(dat, n1, n2){

  # calculate summary stats
  means <- tapply(dat$y, dat$group, mean)
  vars <- tapply(dat$y, dat$group, var)

  # calculate summary stats
  est <- means[1] - means[2] # mean diff
  var_1 <- vars[1] # var for group 1
  var_2 <- vars[2] # var for group 2

  # conventional t-test
  dft <- n1 + n2 - 2  # degrees of freedom
  sp_sq <- ((n1 - 1) * var_1 + (n2 - 1) * var_2) / dft  #
pooled var
  vdt <- sp_sq * (1 / n1 + 1 / n2) # variance of estimate

  # welch t-test
  dfw <- (var_1 / n1 + var_2 / n2)^2 / (((1 / (n1 - 1)) *
(var_1 / n1)^2) + ((1 / (n2 - 1)) * (var_2 / n2)^2))  #
degrees of freedom
  vdw <- var_1 / n1 + var_2 / n2 # variance of estimate

  results <- bind_rows(calc_t(est = est, vd = vdt, df = dft,
method = "t-test"),
                       calc_t(est = est, vd = vdw, df = dfw,
method = "Welch t-test"))


  return(results)

}
```

Fonksiyonun olması gerektiği gibi çalışıp çalışmadığını kontrol etmek yerinde olacaktır. Aşağıdaki örnekte veri kümesi için estimate() fonksiyonunu çalıştırılmaktadır:

```
est_res <-
  estimate(example_dat, n1 = 10000, n2 = 10000) %>%
  mutate_if(is.numeric, round, 5)

est_res
```

```
## # A tibble: 2 × 5
##   method         est p_val lower_bound upper_bound
##   <chr>        <dbl> <dbl>       <dbl>       <dbl>
## 1 t-test        1.02     0       0.978        1.07
## 2 Welch t-test  1.02     0       0.978        1.07
```

Şekil 9. *est_res* fonksiyonunun çıktısına ilişkin ekran görüntüsü.

Sonuçların karşılaştırması *t.test()* fonksiyonu ile yapılabilir:

```
t_res <-
  bind_rows(
    tidy(t.test(y ~ group, data = example_dat, var.equal =
TRUE)),
    tidy(t.test(y ~ group, data = example_dat))
  ) %>%
  mutate(
    estimate = estimate1 - estimate2,
    method = c("t-test", "Welch t-test")
  ) %>%
  select(method, est = estimate, p_val = p.value,
lower_bound = conf.low, upper_bound = conf.high) %>%
  mutate_if(is.numeric, round, 5)

t_res
```

```
## # A tibble: 2 × 5
##   method         est p_val lower_bound upper_bound
##   <chr>        <dbl> <dbl>       <dbl>       <dbl>
## 1 t-test        1.02     0       0.978        1.07
## 2 Welch t-test  1.02     0       0.978        1.07
```

Şekil 10. *t_res* fonksiyonunun çıktısına ilişkin ekran görüntüsü.

İstatistiki yöntemlerin performansı

Bu adımda, kullanılan istatistiki yöntemlerin performansının ölçülmesi için bir fonksiyon oluşturulmaktadır.

```
calc_performance <- function(results, model_params) {

  return(performance_measures)
}
```

Reddedilme oranlarının hesaplanması için aşağıdaki fonksiyon kullanılabilir.

```
calc_performance <- function(results) {

  performance_measures <- results %>%
    group_by(method) %>%
    group_modify(~ calc_rejection(.x, p_values = p_val))

  return(performance_measures)
}
```

Simülatör (simülasyon motoru)

Aşağıdaki belirli bir simülasyon motoru oluşturabilmek için kullanılan bir kod iskeleti bulunmaktadır. Burada bağımsız değişkenler, simülasyon için tekrarlama sayısı ile verinin oluşturulması ve tahmine ilişkin fonksiyonu çalıştırmak için gereken parametreler görülmektedir.

```
run_sim <- function(iterations, model_params, design_params,
seed = NULL) {
  if (!is.null(seed)) set.seed(seed)

  results <-
    rerun(iterations, {
      dat <- generate_dat(model_params)
      estimate(dat, design_params)
    }) %>%
    bind_rows()

  calc_performance(results, model_params)
}
```

Aşağıdaki simülasyon motoru üzerinde çalışılan örneğe ilişkindir:

```
run_sim <- function(iterations, n1, n2, mean_diff, seed =
NULL) {
  if (!is.null(seed)) set.seed(seed)

  results <-
    rerun(iterations, {
      dat <- generate_dat(n1 = n1, n2 = n2, mean_diff =
mean_diff)
      estimate(dat = dat, n1 = n1, n2 = n2)
    }) %>%
    bind_rows()

  calc_performance(results)

}
```

Deneysel tasarım

Örneğe ilişkin tüm fonksiyonlar oluşturulmuş olduğundan simülasyon çalışmasında değiştirilmek istenen faktörler belirtilebilir. Aşağıdaki kod, tasarım faktörlerinin bir listesi ile faktör düzeylerine ilişkin kombinasyonlar oluşturulmaktadır.

```
set.seed(1234)

design_factors <- list(factor1 = , factor2 = , ...) #
combine into a design set

params <-
  cross_df(design_factors) %>%
  mutate(
    iterations = 1000
    seed = round(runif(1) * 2^30) + 1:n()
  )

lengths(design_factors)
nrow(params)
```

Birinci veri grubunun örneklem boyutunu 50 ve ikinci veri grubunun örneklem boyutu 50 ve 70 olarak belirlenmiştir. İki veri grubunun ortalamalarının farkları 0 ile 2 arasında 0.5 aralıkla değişmektedir ve varyansları eşit değildir. Simüle edilen her veri grubuyla, varyansın eşit olduğunu varsayan t-testi ve varyansın eşit olmadığını varsayan Welch t-testini çalıştırılmaktadır.

```
set.seed(1234)

design_factors <- list(
  n1 = 50,
  n2 = c(50, 70),
  mean_diff = c(0, .5, 1, 2)
)
params <-
  cross_df(design_factors) %>%
  mutate(
    iterations = 1000,
    seed = round(runif(1) * 2^30) + 1:n()
  )


# All look right?
lengths(design_factors)
nrow(params)
head(params)
```

```
##        n1        n2 mean_diff
##         1         2         4
```

```
## [1] 8
```

```
## # A tibble: 6 × 5
##      n1    n2 mean_diff iterations      seed
##   <dbl> <dbl>     <dbl>      <dbl>     <dbl>
## 1    50    50         0       1000 122088109
## 2    50    70         0       1000 122088110
## 3    50    50       0.5       1000 122088111
## 4    50    70       0.5       1000 122088112
## 5    50    50         1       1000 122088113
## 6    50    70         1       1000 122088114
```

Şekil 10. Tasarım faktörlerine ilişkin *lengths*, *nrow* ve *head* özelliklerine ilişkin ekran görüntüsü.

Simülasyonun çalıştırılması

Bu adımda simülasyon çalıştırılarak çeşitli sayıdaki tekrarlar için sonuçlar oluşturulmaktadır.

```
system.time(
  results <-
    params %>%
    mutate(
      res = pmap(., .f = run_sim)
    ) %>%
    unnest(cols = res)
)
```

```
results %>%
  kable()
```

n1	n2	mean_diff	iterations	seed	method	K	rej_rate	rej_rate_mcse
50	50	0.0	1000	122088109	t-test	1000	0.052	0.0070211
50	50	0.0	1000	122088109	Welch t-test	1000	0.052	0.0070211
50	70	0.0	1000	122088110	t-test	1000	0.021	0.0045342
50	70	0.0	1000	122088110	Welch t-test	1000	0.044	0.0064857
50	50	0.5	1000	122088111	t-test	1000	0.376	0.0153174
50	50	0.5	1000	122088111	Welch t-test	1000	0.372	0.0152845
50	70	0.5	1000	122088112	t-test	1000	0.339	0.0149693
50	70	0.5	1000	122088112	Welch t-test	1000	0.419	0.0156025
50	50	1.0	1000	122088113	t-test	1000	0.896	0.0096532
50	50	1.0	1000	122088113	Welch t-test	1000	0.894	0.0097347
50	70	1.0	1000	122088114	t-test	1000	0.924	0.0083800
50	70	1.0	1000	122088114	Welch t-test	1000	0.953	0.0066926
50	50	2.0	1000	122088115	t-test	1000	1.000	0.0000000
50	50	2.0	1000	122088115	Welch t-test	1000	1.000	0.0000000
50	70	2.0	1000	122088116	t-test	1000	1.000	0.0000000
50	70	2.0	1000	122088116	Welch t-test	1000	1.000	0.0000000

Şekil 11. Simülasyon sonuçlarına ilişkin ekran görüntüsü.

Yukarıda (Şekil 11), t-testi ve Welch t-testi kullanılarak hipotez testleri için reddetme oranları hesaplanmıştır. H_0, iki grubun aynı ortalamalara sahip olmasıdır. (Örnek R kütüphaneleri için bkz. simhelpers, dplyr, tibble, purrr, tidyr, knitr, kableExtra, broom, ggplot2. Örnekle ilgili daha fazla bilgi için bkz. Morris, TP, White, IR, Crowther, MJ. (2019). Using simulation studies to evaluate statistical methods. Statistics in Medicine. 38: 2074– 2102.)

Sonuç

Simülasyon çalışmalarındaki rastsal bir deneydeki bilginin çeşitli ölçütlerinden bahsetmek yararlı olabilir. Rastsal bir deneyin ölçütlerinin pdf (olasılık yoğunluğu fonksiyonu) f olan rastsal bir vektörle açıklandığı durumda deneye ilişkin tüm olasılıksal bilgi pdf f'de

bulunmaktadır. Diğer yandan, deneylerle ilgili bilgi, ölçütlerin ortalaması ve değişkenliği ile ilgili bilgiyi sağlayan, sırasıyla, X'e ilişkin beklenti ve kovaryans matrisi gibi birkaç anahtar sayıyla ifade edilebilir.

Bilgiye ilişkin diğer bir ölçüt, *Shannon entropisi*nin X mesajının ikili bir iletişim kanalında transfer edilebilmesi için ihtiyaç duyulan ortalama bit sayısıyla karakterize ettiği kodlama ve telekomünikasyon teorisiyle ilgilidir.

Başka bir bilgi yaklaşımı istatistik içinde bulunabilir. Noktasal kestirim teorisinde, pdf f, θ parametre vektörüne dayanır. Bu problem, bir X çıktısı yolu ile θ'nın ne kadar iyi kestirimle bulunabileceği, diğer bir deyişle, bir veri olan X'in içinde θ'yla ilgili ne kadar bilgi bulunduğuyla ilgilidir. Bu türdeki çeşitli ölçütler *maksimum olabilirlik* (maximum likelihood) ve *Fisher bilgi matrisi* ile ilişkilendirilir. Son olarak, rastsal bir deneydeki bilgi miktarı, *Kullback-Liebler uzaklığı* ya da *çapraz-entropi* gibi uzaklık kavramıyla ölçülür.

Öngörüsel (öngörüye dayalı) analitik, veriden bilgi çıkarmak ve çalışılan modele ilişkin eğilimleri ve davranış kalıplarını tahmin etmek için kullanılan bir istatistik alanıdır. Öngörüsel analitiğin istatistiksel teknikleri, açıklayıcı değişkenler ile geçmiş olaylardan tahmin edilen değişkenler arasındaki ilişkileri yakalamaya ve bilinmeyen bir sonucu öngörmek için bunlardan yararlanmaya dayanır. Öngörüsel analitiği, daha iyi kararlar almak üzere sistemlerin gelecekteki davranışlarını tahmin etmek için deneyimden (verilerden) öğrenen bir teknoloji olarak tanımlamak yanlış olmayacaktır. Bu teknolojinin arka planında bulunan simülasyon çalışmaları istatistiksel yöntemlerin stokastik süreçlere uygulanabileceği platformları sunduğu için bu bölümde bulunmaktadır.

Bununla birlikte, sonuçların doğruluğu ve kullanılabilirliğinin büyük ölçüde veri analizinin düzeyine ve bu analizde kullanılan varsayımların kalitesine bağlıdır. Bilgisayar simülasyonlarında bazen göz ardı edilse de, sonuçların doğruluğunun test edilmesi ve modelin doğru bir şekilde anlaşılmasını sağlamak için duyarlılık analizi yapmak önemlidir.

Kaynaklar

Banks, J., Carson, J., NelsonB, Nicol D. (2001). *Discrete-event system simulation*. Prentice Hall.

Bassham, L. E. (2010) *A statistical testing for pseudorandom number generators for cryptographic applications*. NIST special publication 800-22.

Bengtsson, H. (2020). *Future: Unified parallel and distributed processing in r for everyone*. https://CRAN.R-project.org/package=future

Devroye, L. (1986). *Non-uniform random variate generation*. Springer-Verlag, New York. http://www.nrbook.com/devroye/

EPFL (2005). *Mission to build a simulated brain begins*. École Polytechnique Fédérale de Lausanne. https://www.newscientist.com/article/dn7470-mission-to-build-a-simulated-brain-begins/

Forrester, J. W. (1961). *Industrial dynamics*. MIT Press.

Iannone, R. (2020). *DiagrammeR: Graph/network visualization*. https://CRAN.R-project.org/package=DiagrammeR

IBM (2013). *Molecular simulation of macroscopic phenomena*. https://web.archive.org/web/20130522082737/http://www.almaden.ibm.com/st/past_projects/fractures/

Karr, J. R. vd. (2012). *A Whole-cell computational model predicts phenotype from genotype*. Cell. https://doi.org/10.1016/j.cell.2012.05.044

Knuth D. E. (1997), *The Art of computer programming V2 seminumerical algorithms*. Addison-Wesley.

Knuth, D. E. (2015). *The Art of Programming*. Addison-Wesley Professional.

LANL (2005). *Largest computational biology simulation mimics life's most essential nanomachine*. Los Alamos National Laboratory. https://web.archive.org/web/20070704061957/http://www.lanl.gov/news/index.php/fuseaction/home.story/story_id/7428

Matsumoto M., Saito M. (2008). *SIMD-oriented fast Mersenne twister: a 128-bit Pseudorandom number generator.*

Matsumoto, M. and Nishimura, T. (1998). *Dynamic creation of pseudorandom number generators, Monte Carlo and Quasi-Monte Carlo methods*. Springer, pp 56-69.

Matsumoto, M. and Nishimura, T. (1998). *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*. ACM Trans. Model. Comput. Simul. 8, 1, 3-30.

Menezes, A. J., van Oorschot, P. C., Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC Press.

Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074–2102.

NASA (1997). *Researchers stage largest military simulation ever*. California Institute of Technology, Jet Propulsion Laboratory. https://web.archive.org/web/20080122123958/http://www.jpl.nasa.gov/releases/97/military.html

NIST/SEMATECH. *e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/.

Paneton F., L'Ecuyer P. and Matsumoto M. (2006). *Improved long-period generators based on linear recurrences modulo 2*. ACM Transactions on Mathematical Software.

Park S. K., Miller K. W. (1988). *Random number generators: good ones are hard to find.* Association for Computing Machinery, vol. 31, 10, pp 1192-2001.

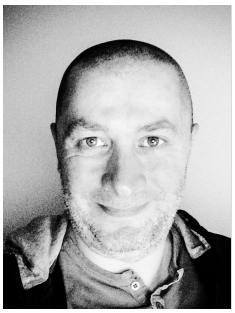
Schweppe, F. C. (1973). *Uncertain dynamic systems*. Prentice-Hall.

Sokolowski, J. A., Banks, C. M. (2009). *Principles of modeling and simulation*. John Wiley & Son.

Sterman, J. (2010). *Business dynamics: Systems thinking and modeling for the complex world*. McGraw-Hill Education.

Vaughan, D., & Dancho, M. (2018). *Furrr: Apply mapping functions in parallel using futures*. https://CRAN.R-project.org/package=furrr

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., … Yutani, H. (2019). *Welcome to the tidyverse*. Journal of Open Source Software, 4(43), 1686. https://doi.org/10.21105/joss.01686



Dr. H. Kemal İlter 1994 Orta Doğu Teknik Üniversitesi mühendislik lisans mezunudur ve 1998'de sayısal yöntemler alanında yüksek lisans, 2004'de e-iş ve ERP sistemleri alanında doktora çalışmalarını tamamlamıştır. Bilişim sistemleri alanında profesyonel olarak çalıştıktan sonra 1996-2000 arasında Gazi Üniversitesi İşletme Bölümü, 2000-2011 arasında Başkent Üniversitesi İşletme Bölümü, 2011-2021 arasında Yıldırım Beyazıt Üniversitesi Yönetim Bilişim Sistemleri Bölümü'nde tam zamanlı ve TOBB ETÜ'de 2006-2021 arasında yarı-zamanlı olarak bulunan Dr. İlter, 2021'den bu yana Bakırçay Üniversitesi Yönetim Bilişim Sistemleri Bölümü'nde profesör olarak tam zamanlı olarak çalışmaktadır. Kanada, Montreal, Concordia University'de 2012-2015 arasında doktora sonrası araştırmacı ve 2019-2020 arasında misafir öğretim üyesi olarak bulunan Dr. İlter, acil ve afet durumlarının simülasyonu (simulation of states of emergency and disasters), tedarik zinciri haritalaması (supply chain mapping), uyum yüzeyleri (fitness landscapes) ve üretim sistemlerinde karmaşıklık (operations complexity) konularında çalışmaktadır.