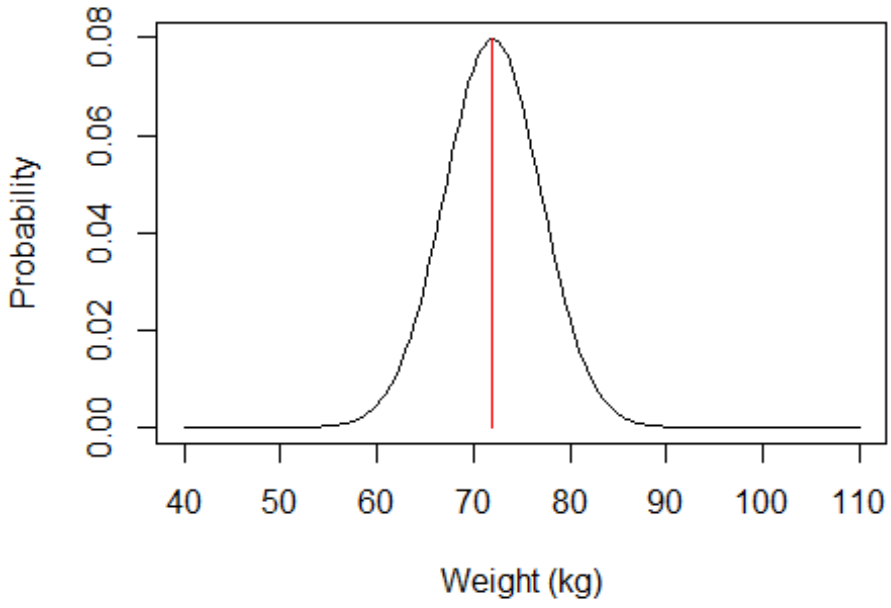
4. Gün 1. Uygulama: Stokastik modeller

Bu uygulamada stokastiklik, salgın kalıcılığı ve azalarak bitme olasılığı ile ilgili kavramlardan bazılarını uygulayacağız. Bu amaç doğrultusunda binom dağılımı ve R'de bir dağılımdan nasıl örnek alabileceğimizi inceleyerek başlayacağız.

Binom dağılıma neden ihtiyaç duyarız?

İstatistik ve olasılık teorisinde belirli özellikleri istatistiksel dağılımla ilişkilendirme eğilimindeyiz. Örneğin, popülasyon vücut ağırlığının normal bir dağılımı takip ettiğini güvenle söyleyebiliriz. Normal dağılım süreklidir (kilonuz 74 kg veya 74,37 kg olabilir ve dağılımın bir parçası olabilir) ve verinin yayılmasını ve merkezi ölçütü yansıtan ortalama değer ve standart sapmaya göre tanımlanır. Örneğin, şunu söyleriz:

$$Kilo \sim Normal(72.5,5)$$



Binom dağılım, bir dizi deneydeki başarılı deney sayısının ayrık olasılıklı dağılımıdır. Bir bozuk parayla birkaç kez yazı tura attığınızı ve kaç kez tura geleceğini saydığınızı düşünün. Binom dağılım *n* (deneme sayısı) ve *p* (başarı olasılığı) parametrelerini alır. Yazı tura atma örneğimize devam edersek bozuk para düzgünse başarı olasılığının

(paranın tura gelmesi) %50 olduğunu söyleyebiliriz. Bu, bozuk parayla yeterince sayıda yazı tura atarsak başarılı sonuç sayısının %50'ye yaklaşması gerektiği anlamına gelir.

Daha da önemlisi, 1 denemelik (yalnızca bir kez yazı tura atmak) binom deneylerden bahsettiğimizde *Bernoulli* denemelerinden bahsetmiş oluruz. > 1 denemeler için binom dağılımdan bahsederiz.

Bu R'de kullanmaya çalışalım. R'de *rbinom* temel fonksiyonu bu tür deneyleri hesaplamak için kullanılır. *rbinom* gözlem sayısı için *n*, deneme sayısı için *boyut*, başarı olasılığı için ise *prob* parametrelerini kullanır. Aşağıdaki kodu çalıştırın:

```
# Bir kez yazı tura atalım (Bernoulli)
rbinom(n=1,size=1,prob=0.5)
```

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

```
# Şimdi 100 deneme yapalım
rbinom(n=1,size=100,prob=0.5)
```

```
## [1] 54
```

1. 100 kez yazı tura attığınızda kaç kez başarılı oldunuz?
2. Deneme sayısını artırırsanız ne olur?

Bu dersle daha ilgili bir durumu inceleyelim. Binom dağılımı artık biliyorsunuz, binom olasılığın enfeksiyonun bulaşma süreciyle nasıl ilişkili olduğunu artık görebilirsiniz. Şimdiye kadar önceki derslerimizde bulaşmanın en az üç faktöre bağlı olduğunu öğrendik:

Bulaşma olasılığı

$$\beta$$

, enfeksiyon prevalansı

$$\frac{I}{N}$$

ve bulaşıcı döneminin süresi

$$D$$

.

Daha önce incelediğimiz gibi herhangi bir zaman noktasındaki yeni enfeksiyon sayısı (*Y*) şu şekilde tanımlanabilir:

$$Y = S\frac{\beta I}{N}$$

Burada S duyarlı birey sayısıdır. Deterministik açıdan, değişkenlerimiz ve parametrelerimiz aynı olduğu sürece bu her zaman aynı rakamı verecektir. Ancak, daha önce öğrendiğimiz gibi enfeksiyon süreci rastgelelik içermektedir.

Binomial denemelerini hatırlamak için enfeksiyon sürecimizin bu çerçeveye nasıl sığabildiğini görebiliriz. Belirli bir noktada enfeksiyon prevalansı %10 ve *beta* = 0,2 olsun dersek enfeksiyon olasılığı 0,1 x 0,2 olur. Bu dağılımdaki deneme sayısı duyarlı bireylerin sayısıdır. Stokastik olayları hesaba katan enfeksiyon sayısını tahmin etmek için aşağıdaki kodu uyarlamaya çalışın:

```
# Parametrelerimizi tanımlayın
R0 <- 2     # Temel üreme numarası
gamma<- 0.1 # iyileşme oranı
beta<-  ?? # <- R0 formülünden beta değerini çıkarabilir misiniz?
prevalence <- 0.1 # Enfeksiyon prevalansı (I/N)
S <- 1000 # t zamanındaki duyarlı bireyler

n_trials <- ?
probability_of_infection<- ?

# Yeni enfeksiyon sayını Y tahmin etmek için ikiterimli dağılımdan çekin
Y<-rbinom(n=1,size= n_trials ,prob=probability_of_infection)
```

Enfeksiyon sürecinde stokastikliğin basit R komutlarıyla nasıl ifade edilebileceğini artık daha iyi anladık, stokastik bir modeli formüle etmeye çalışalım.

Teknik Parantez:

Stokastik model R'de farklı yaklaşımlarla kodlanabilir. *Odin* adı verilen bir paket kullanacağız. Bu amaç için farklı paketler hatta temel R bile kullanılabilir ancak *Odin*'in ayrık stokastik sürecin oluşturulmasını daha sade bir şekilde gösterdiğini düşünüyorum. *Odin*'i kurmak için aşağıdaki kodu kopyalayıp komut dizinize yapıştırın. Kurduktan sonra bu kodu silebilirsiniz veya değerlendirebilirsiniz (yalnızca bir kez kurulum yapmak için ihtiyacınız vardır).

```
if (!require("drat")) install.packages("drat")
drat:::add("mrc-ide")
install.packages("dde")
install.packages("odin")
```

İşlem birkaç dakika sürebilir, kurulumdan sonra *odin* paketini normal biçimde arayabilirsiniz:

```
library(odin)
```

Stokastik modeli formüle etmek

Topluluğunuzda yeni bir viral hastalık ("X" hastalığı) tespit edilmiştir. Şimdiye kadar toplanan gözetim verilerinden birkaç parametre tahmin edilmiştir. Başlangıç R0 değeri ~ 2 olarak tahmin edilmiştir ve semptomların başlangıcından iyileşmeye kadar geçen süre ortalama 8 gündür. Ayrıca, "X" hastalığının önceki salgınlarında enfeksiyon yoluyla kazanılan bağışıklığın ortalama 3 aylık bir ömre sahip olduğu kaydedilmiştir.

Aşağıdaki kod SIR stokastik modeli tanımlamaktadır. 1. Boşlukları (*??* sembolüyle işaretlenmiş) doldurmaya çalışın ve stokastik olayların enfeksiyon sürecine nasıl dahil edildiğine dikkat edin:

```
library(odin)
library(ggplot2)
library(reshape2)

sir_generator <- odin::odin({
  ## Bölmeler arasında geçişler için temel denklemler:
  update(S) <- S - n_SI + n_RS # Duyarlı
  update(I) <- I + n_SI - n_IR # Enfekte
  update(R) <- R + n_IR - n_RS # İyileşmiş

  ## Bireysel geçiş olasılıkları:
  p_SI <- 1 - exp(-beta * I / N) # S'den I'ya
  p_IR <- 1 - exp(-gamma)        # I'dan R'ye
  p_RS <- 1 - exp(-delta)        # R'den S'ye

  ## Bölmeler arasında değişen sayıları tanımlamak için ikiterimli
dağılımlardan çekin:
  n_SI <- rbinom(S, p_SI) # Yeni enfeksiyonlar
  n_IR <- rbinom(I, p_IR) # Yeni iyileşmiş
  n_RS <- rbinom(R, p_RS) # Yeni bağışıklık kaybı

  ## Toplam popülasyon boyutu
  N <- S + I + R

  # Beta'yı R0'a göre tanımlayın
  beta <- ?? # Beta'yı R0'a göre tanımlayabilir misiniz?

  ## Başlangıç durumları:
  initial(S) <- S_ini
  initial(I) <- I_ini
  initial(R) <- 0

  ## Kullanıcı tanımlı parametreler - parantez içindekiler varsayılan:
  S_ini <- user()
```

```
  I_ini <- user()
  R0    <- user()
  gamma <- user()
  delta <- user()

}, verbose = FALSE)
```

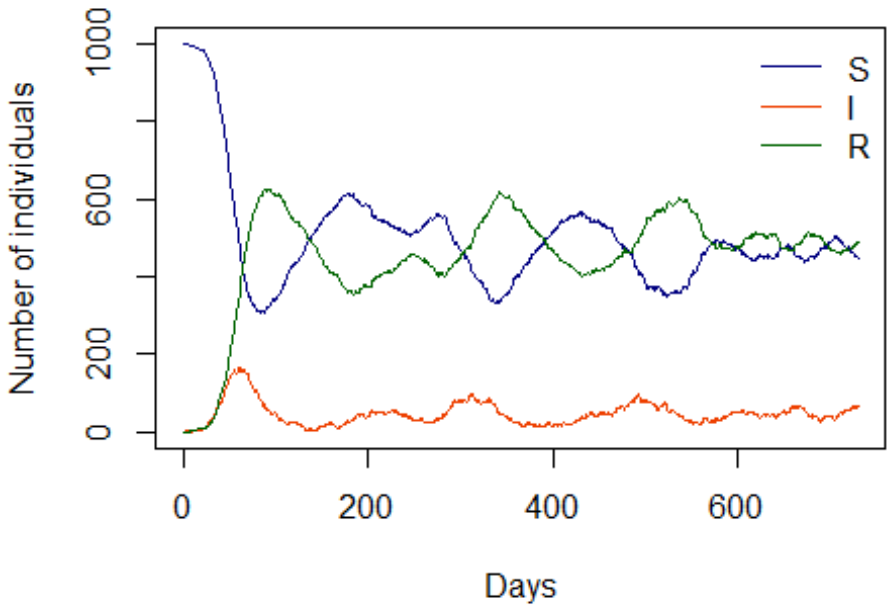
Soru işaretli boşlukları doldurduysanız şimdi de

- Model parametrelerini tanımlayın ve sonuçları görmek için modelinizi çalıştırın.

```
# Parametreleri tanımlayın
R0   <- ?? # R0
gamma<- ?? # iyileşme oranı
delta<- ?? # Bağışıklık kaybı oranı
S0   <- 1000 # 0 zamanında duyarlı popülasyon
I0   <- 1    # 0 zamanında enfekte popülasyon


# Tanımlanan parametrelerle "sir" model nesnesi oluşturun
sir <- sir_generator$new(
  S_ini = S0,
  I_ini = I0,
  R0    = R0,
  gamma = gamma,
  delta = delta)

# Rastgele sayı üretimi (R standart) tohum oluşturun
set.seed(1)

# SIR modelimizi iki yıllık süre boyunca çalıştırın (gün adımlarında)
t_end<- 365 * 2
sir_res <- sir$run(0:t_end)

# Model çıktısına göz atın

head(sir_res)


#Modelimizin grafiğini çizin

sir_col <- c("Navyblue", "orangered2", "darkgreen") # Renklerin grafiğini
çizin

days <- sir_res[, 1] # grafiğimiz için zaman vektörü tanımlayın

matplot(days, sir_res[, -1], xlab = "Days", ylab = "Number of individuals",
        type = "l", col = sir_col, lty = 1)
```

```
legend("topright", lwd = 1, col = sir_col, legend = c("S", "I", "R"), bty =
"n")
```

Modelinizi çalıştırdıktan sonra aşağıdaki gibi bir grafik görmelisiniz:



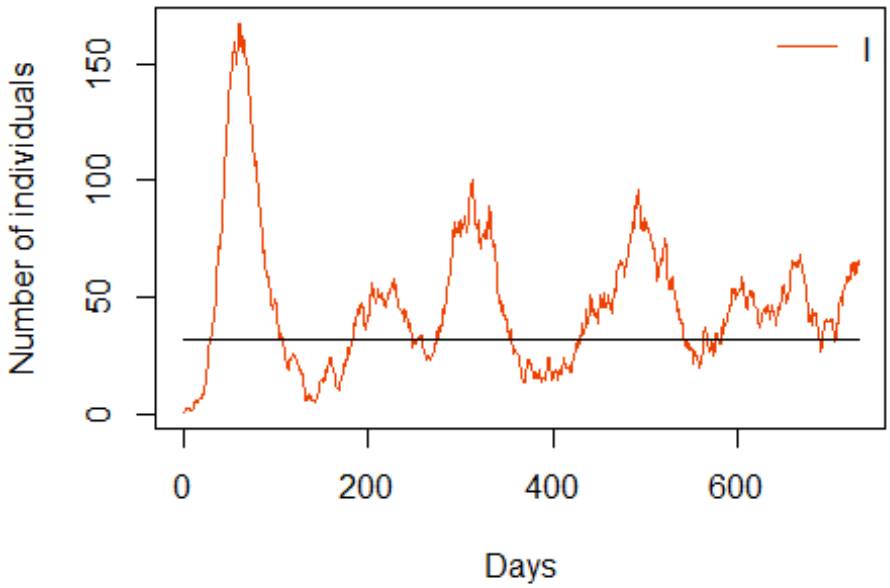
Stokastik bir süreç başlattığımız göz önüne alındığında her modelin çalışması farklı olacaktır. Sonuçların nasıl değiştiğini görmek için aynı kodu birkaç kez çalıştırmayı deneyin.

Salgının kalıcılığı konusundaki dersimizden salgının sona erme olasılığının daha muhtemel olduğu bir eşik değerini kabaca tahmin edebileceğimizi hatırlayabilirsiniz.

c) Aşağıdaki kodda bu eşiği tanımlamaya çalışın ve bu eşiğe göre grafik haline getirilen enfeksiyonları görmek için modelinizin grafiğini tekrar çizin.

```
# Bir eşik tanımlayın

Y_limit <- ??
  # İpucu: bunun popülasyon boyutuna göre nasıl tanımlandığını görmek için
ders slaytlarına bakın

#Modelimizin grafiğini çizin

sir_col <- c("Navyblue", "orangered2", "darkgreen") # Renklerin grafiğini
çizin
```

```
days <- sir_res[, 1] # grafiğimiz için zaman vektörü tanımlayın

matplot(days, sir_res[, 3], xlab = "Days", ylab = "Number of individuals",
        type = "l", col = "orangered2", lty = 1)
lines(c(0,365*2),c(Y_limit,Y_limit), col="black")
legend("topright", lwd = 1, col = "Orangered2", legend = c("I"), bty = "n")
```



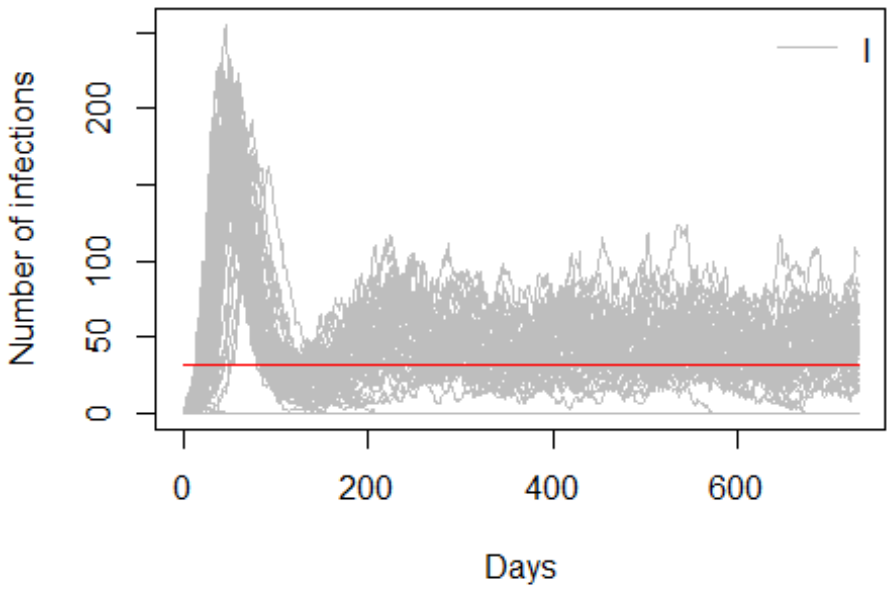
4. Enfeksiyonların zaman içindeki trendi hakkında ne söyleyebilirsiniz?
5. Bu analizi R0 = 1,1 değeri ve R0 = 4 için tekrarlamayı deneyin. Ne gözlemliyorsunuz?
6. R0=2 eşitliğini kullanarak ortalama bağışıklık kaybını 1 yıl kabul etmek için delta değerini değiştirebilir misiniz? Ne gözlemliyorsunuz ve bu davranışın nedenini açıklayabilir misiniz?

Modelimizin birçok eş zamanlı gerçekleştirmesini aynı anda çalıştırmak için *odin* paketinin başka bir özelliğini kullanabiliriz.

7. Aşağıdaki kodda modelimizi 100 kez çalıştıracağız ve sonuçlarını grafik haline getireceğiz

```
# Modelimizi 100 kez çalıştırmak için kopyalamayı kullanın
sir_100 <- sir$run(0:t_end, replicate = 100)
# res_200 <- sir$transform_variables(res_200)
# res_200 <- cbind.data.frame(t = res_200[[1]], res_200[-1])
```

```
matplot(sir_100[, 1,],sir_100[, 3,], xlab = "Days", ylab = "Number of
infections",
        type = "l", lty = 1, col="grey")
lines(c(0,t_end),c(Y_limit,Y_limit),type="l", col="red")
legend("topright", lwd = 1, col = "grey", legend = c("I"), bty = "n")
```



Aynı model için çok sayıda kopya veya simülasyona sahip olduğumuzu göz önünde bulundurarak mevcut model parametreleri için salgının sona erme olasılığını tahmin edebiliriz. Bunu, simülasyon süresi sonunda kaç enfeksiyon yörüngesinin sıfıra eşit olduğuna bakarak yapabiliriz.

8. Sona erme olasılığını tahmin etmek için aşağıdaki kodu kullanın.
9. Model parametrelerinizi R0=4,5 ve R0 = 1 olarak değiştirin ve bu değeri yeniden tahmin edin

```
# Sona erme olasılığını bulmak için kullanıcı tarafından tanımlı bir
fonksiyon oluşturun
prob_extinct<-function(results,t_end){

  n_extinct<-length(which(results[t_end,3,]==0)) # sıfırda sona eren
simülasyonları bulun
  n_runs   <-length(results[1,1,])

  return(prob_extinction=n_extinct/n_runs)
}
```

```
# model sonuçları nesnemizi ve zaman uzunluğunu girerek yeni tanımlanan
fonksiyonu çağırın
prob_extinct(sir_100,t_end)

## [1] 0,49
```