1. Gün 2. Uygulama: BaşlaRken

Önceki uygulamada R ve RStudio ortamının nasıl kurulacağını ve R projesi, klasör yapısı ve ilk R komut dizisiyle ilk iş akışını nasıl oluşturulacağını öğrendiniz.

Bu uygulamaya başlamak için:

- EpiRcourse projenize gidin, bunu açın
- Oluşturduğunuz R komut dizisini açın

Bu uygulama sırasında birkaç satır kod yazmanız istenecektir, bu çalışma sayfasındaki kod parçacıklarından bazılarını kopyalayıp yapıştırabilirsiniz ya da en baştan kod yazmayı deneyebilirsiniz (programlama akışını deneyimlemek için!)

Veri yapıları

Bunlar R'deki en temel yapılardır. Öncelikle aşağıdaki kodu inceleyin. Aynı nesneleri kendi komut dizinizde oluşturun.

```
vector_double <- c(1, 2, 3, 4, 5, 6)

vector_logic <- c(TRUE, FALSE, FALSE, TRUE)

vector_character<- c("A", "B", "C", "D")

vector_integer <- c(1L, 2L, 3L, 4L)
```

Artık komut diziniz, her biri farklı sınıfta veriler içeren dört yeni nesne oluşturmak için gerekli yeni bilgiler içermektedir.
Ancak, komut dizisi yalnızca çalışmanızın günlüğüdür. Bu nesneleri gerçekten oluşturmak için kodu çalıştırmanız gerekir.

R kodunun çalıştırılması

R komut dizisindeki kodu çalıştırmanın farklı yolları vardır:

1. Çalıştırmak istediğiniz komut satırlarını seçin ve ctrl + enter'a basın (veya komut dizisinin sağ üst kısmındaki "Run" (Çalıştır) simgesine tıklayın)
2. Komut dizisinin tüm içeriklerini çalıştırmak istiyorsanız komut dizinizin sağ üst kısmındaki "source" (kaynak) simgesine tıklayın.

Yukarıdaki kodu çalıştırmayı deneyin. Bu satırları çalıştırırken ne oluyor? Oluşturulan yeni nesneleri bulabildiniz mi?

Ders notlarından yola çıkarak yeni vektörün içeriğini konsol penceresinde çıkarmak için ne yaparsınız? (Sonuçları aşağıdaki gibi görebilmelisiniz)

```
## [1] 1 2 3 4 5 6

## [1] TRUE FALSE FALSE  TRUE

## [1] "A" "B" "C" "D"

## [1] 1 2 3 4
```

Matrisler

Matrisler, vektörlerden biraz daha karmaşık yapıdadır ve iki ana özelliği vardır:

- Matris yalnızca bir tür veriden oluşur
- Matris iki boyutludur

Matris oluşturma komutu üç argüman kullanır:

Sözdizimi: matrix_name <- matrix (*data*, *nrow* , *ncol*)

- *data* matriste kullanmak istediğimiz vektörler listesine karşılık gelir
- *nrow* verilerin bölüneceği satır sayısıdır (birinci boyut)
- *ncol* verilerin bölüneceği sütun sayısıdır (ikinci boyut)

Not: Varsayılan olarak, *byrow = TRUE* kullanarak aksini belirtmedikçe matris sütunlara göre doldurulur

data argümanı matrisinizi doldurmak istediğiniz tek bir değer de olabilir (örneğin, 0 veya 10).

Şunu deneyin; iki yeni nesne oluşturun, biri daha önce oluşturduğunuz *vector_double* elemanlarının bulunduğu 2 x 2 matris olacaktır. Diğer nesne 2 X 3 boyutunda 2 değeriyle doldurulan boş bir matris olacaktır. Matrislerinize mat1 ve mat2 adını verin. Matrisinizin içeriğini kontrol edin, böyle bir şey mi elde ettiniz?

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6

##      [,1] [,2] [,3]
## [1,]    2    2    2
## [2,]    2    2    2
```

Matris çarpımı, kodlama sırasında çoğunlukla gerekli bir işlemdir. Ancak, element-wise matris çarpımı (her ögenin birer birer çarpılması) yapmanın koşulu, başlangıçtaki iki matrisin aynı boyutta olmasıdır.

mat1 ve *mat2* matrislerini çarpmanız gerektiğini düşünün. Ne yapabilirsiniz?

- Matrisleri çarpabilmeniz için birini "çevirmenizi" sağlayacak bir yöntem aramak için interneti kullanın.

- *mat1* ve transpoze (devrik) matris *mat2*'nin çarpımı olan üçüncü bir matris (*mat3*) oluşturun.

İPUCU: aradığınız işlemin adı transpozisyondur!

Sonuç şöyle görünmelidir:

```
##      [,1] [,2]
## [1,]    2    8
## [2,]    4   10
## [3,]    6   12
```

Veri çerçeveleri

Veri çerçevesi, bir matrise benzeyen ancak tamamen aynı olmayan heterojen, iki boyutlu bir yapıdır. Matristen farklı olarak birden çok veri türü tek bir veri çerçevesinin bir parçası olabilir.

data.frame komutu argümanları veri çerçevesindeki sütunlardır. (**Önemli**: Bir veri çerçevesine sığdırmak için her sütunda aynı sayıda satır olmalıdır).

Veri çerçeveleri farklı uzunluktaki vektörlere müsaade etmez. Vektörün uzunluğu veri çerçevesinin uzunluğundan az olduğunda veri çerçevesi vektörü kendi uzunluğunda olmaya zorlar.

Aşağıdaki kodu kontrol edin:

```
data_example <- data.frame(vector_character, vector_double, vector_logic,
vector_integer)
```

Ne oluyor? Çalışıyor mu?

Şimdi aynı veri çerçevesini yalnızca ilk dört elemanı vektör çiftine dahil ederek oluşturmaya çalışın:

- Ders notlarını takip ederek bir vektöre nasıl eriştiğinize ve alt küme haline getirdiğinize bakın

- *vector_double*'ın ilk dört elemanı ve tüm vektörlerle bir veri çerçevesi oluşturun

Şu şekilde görünmelidir:

```
##   vector_character vector_double.1.4. vector_logic vector_integer
## 1                A                  1         TRUE              1
## 2                B                  2        FALSE              2
## 3                C                  3        FALSE              3
## 4                D                  4         TRUE              4
```

Artık ikinci satır, üçüncü sütundaki elemana erişebilirsiniz ve bunu *new_vec* yeni nesneye atayabilirsiniz.

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

Listeler

Liste, R'deki en karmaşık yapıdır. Liste herhangi bir boyuttaki her tür nesneden oluşabilir!

```
list_example <- list(vector_character,
                     vector_double,
                     vector_logic,
                     data_example,
                     new_vec)
```

Gördüğünüz gibi listede boyutundan veya sınıfından bağımsız olarak her tür veri bulunabilir.

- Slot adıyla çağırarak yukarıdaki listedeki *vector_logic* elemanına erişebilir misiniz?

Fonksiyonlar

Birkaç tür fonksiyon vardır:

- Temel veya ilkel fonksiyonlar: bunlar temel pakette R'de varsayılan fonksiyonlardır. Örneğin, bazı aritmetik işlemleri ve ayrıca medyan değerler, ortalama değerler ve bir değişkenin özetini çıkarma gibi daha karmaşık işlemleri içerebilirler.

- Paket fonksiyonları: bunlar paket içinde oluşturulan fonksiyonlardır. Örneğin, *stats* paketindeki glm fonksiyonu.

- Kullanıcı tarafından tanımlı fonksiyonlar: Bunlar, herhangi bir kullanıcının özel bir rutin için oluşturabileceği fonksiyonlardır.

Bir fonksiyonun temel bileşenleri şunlardır:

a) Ad: fonksiyona verilen addır (anlamlı ad vermenin önemli olduğunu unutmayın!)

b) Argümanlar: fonksiyonun çalışmasını kontrol eden girdi değeridir (veya değerleridir).

c) Gövde: argümanlarda yapılan işlemler veya değişikliklerdir.

d) Çıktı: argümanları değiştirdikten sonraki sonuçlardır. Bu çıktı bir veri serisine karşılık gelirse geri dön komutunu kullanarak bunu çıkarabiliriz.

e) İç fonksiyon ortamı: bir fonksiyonda bulunan özel kurallar ve nesnelerdir. Bu kurallar ve nesneler fonksiyon dışında çalışmaz.

Kullanıcı tarafından tanımlı fonksiyonlar

Kendi fonksiyonlarınızı oluşturmak R'deki en güçlü araçlardan biridir. Böylece kodlama yaparken zamandan tasarruf edebilir ve veriminizi arttırabilirsiniz.

Öncelikle, aşağıdaki fonksiyonu inceleyin, bileşenlerine bakın ve farklı argüman değerleriyle denemeler yapın. Ayrıca, çıktığı yeni bir nesneye geçirmeye çalışın. *power_function* bir araç olarak kesinlikle gereksizdir, ancak şimdilik iyi bir örnektir!

```
power_function <- function(base,power){ # fonksiyon tanımı ve argümanlar

  #gövde
  x<-base^power

  #çıktı
  return(x)
}
```

Artık bir R fonksiyonunun nasıl yazılacağını daha iyi biliyorsunuz. Sonraki adımları takip edin:

a) Vücut Kitle Endeksini (BMI) hesaplamak için bir fonksiyon oluşturun

b) Bu kişiler için BMI değerini hesaplayın:

- Hasta 1: 1,83 m ve 96 kg
- Hasta 2: 1,65 m ve 78 kg
- Hasta 3: 1,98 m ve 99 kg

Şu sonuçları mı buluyorsunuz?

```
## [1] 28.66613 28.65014 25.25253
```

Paketler

R'de varsayılan olarak tümleşik gelen tüm mükemmel temel fonksiyonlara ek olarak diğer R kullanıcılarının özel amaçlar için oluşturduğu binlerce pakete erişebilirsiniz. Bazı kalite kriterlerini karşıladığı sürece herkes bir paket oluşturabilir. İyi bir paket şeffaf olmalı ve içerdiği tüm fonksiyonlar için yeterli dokümantasyonla birlikte sunulmalıdır.

Onaylanmış R paketlerine Kapsamlı R Arşiv Ağı'ndan (CRAN) https://cran.r-project.org ulaşabilirsiniz ancak aşağıdaki komutları kullanarak Rstudio'dan doğrudan kurmanız gerekir:

- Paket kurun: install.packages("*package-name*"). Bu adım yalnızca bir kez yapılmalıdır.
- Paketi oturumunuza yükleyin: library("*package-name*"). Bu kod satırı genellikle komut dizinizin üst kısmındadır. Bu yüzden bunu her çalıştırdığınızda oturumunuzla ilgili paketleri yükleyecektir.

Kurs sırasında kullanacağımız birkaç kullanışlı paket yüklemeyi deneyin.

```
install.packages("deSolve")   # Diferansiyel denklemleri çözmek için
install.packages("reshape2")  # Veri setlerini kullanmak için
install.packages("tidyverse") # veri yönetimi ve grafik oluşturmak için
paketlerin toplanması
install.packages("here")      # Klasörünüzün kök yolunu elde etmenizi sağlar
```

Bu paketlerin adını google'da aratın, her biri için dokümantasyon bulacaksınız. Sağladıkları fonksiyon setlerini ve nasıl çalıştıklarını keşfedebilirsiniz.