

# Shiny ile Dinamik Web Sitesi Oluřturma El Kitabı

Baran Doğru

Son Güncelleme: 06.02.2020

## Contents

<b>1 Giriř</b>	<b>1</b>
1.1 Amaç . . . . .	1
1.2 Motivasyon . . . . .	2
<b>2 Hazırlıklar ve İlk Shiny Uygulaması</b>	<b>2</b>
2.1 Kurulum . . . . .	2
2.2 İlk Shiny Uygulaması . . . . .	3
<b>3 Shiny Uygulamasına Genel Bakıř</b>	<b>7</b>
3.1 Shiny Uygulamasının En Basit Hali . . . . .	7
3.2 Shiny Uygulamasının Genel Hatları . . . . .	8
<b>Ekler</b>	<b>13</b>
Yazar Hakkında . . . . .	13
RStudio Cloud’da Çalışma . . . . .	14
Örnek Projenin Kodu ve Çalıştırma Linki . . . . .	16

## 1 Giriř

### 1.1 Amaç

Bu döküman, Shiny aracılığıyla dinamik ve interaktif web sitelerinin nasıl oluşturulabileceğine dair bir giriş yapılması amacıyla kaleme alınmıştır. Shiny, verilerin daha açık bir şekilde anlaşılabilmesi ve anlatılabilmesi adına direkt olarak R kodu yazarak dinamik ve interaktif web siteleri oluşturmaya yarayan bir R paketidir.

Dökümanda öncelikli olarak elektrik piyasasında rol alan bireyler hedef kitle olarak alınmış olsa da Shiny kullanarak bir web sitesi oluşturmak isteyen herkes için oldukça faydalı bir döküman olduğuna inanıyoruz.

## 1.2 Motivasyon

Bu dökümanda yer alan bütün kodlara Github hesabından ulaşabilirsiniz.

Bu dökümanı başarıyla tamamladığınızda sizlerin de aşağıda gösterilen uygulamaya benzer uygulamalar yaratabileceğinize inanıyor, kolaylıklar diliyoruz.

(Bu kitabı pdf veya epub formatında görüntülüyorsanız uygulamayı göremeyeceksiniz.)

Bu örnek uygulamanın kodu ve çalıştırma linkine Örnek Projenin Kodu ve Çalıştırma Linki kısmından ulaşabilirsiniz.

## 2 Hazırlıklar ve İlk Shiny Uygulaması

Bu bölümde gerekli hazırlıkların tamamlanması ve ilk Shiny uygulamasının nasıl çalıştırılacağı anlatılacaktır.

### 2.1 Kurulum

#### 2.1.1 R ve RStudio'nun İndirilmesi / RStudio Cloud

Öncelikle ücretsiz bir istatistiksel programlama dili olan R'nin yüklenmesi gerekiyor. R'ı indirmek için <https://cran.rstudio.com/> sitesine ilerleyiniz ve işletim sisteminize uygun olan versiyonuna tıklayarak indirmeyi başlatınız.

Daha sonra güçlü bir kullanıcı arayüzü olan RStudio'nun indirilmesi gerekiyor. Onun için de <https://rstudio.com/products/rstudio/download/> linkini kullanabilirsiniz. İndirme tamamlandıktan sonra direktifler takip edilerek RStudio kurulumu da tamamlanmış olacaktır.

Eğer bütün bu indirmeleri yapmak istemezseniz ise, yine bir RStudio ürünü olan tamamen ücretsiz ve herhangi bir indirmeye ihtiyaç duymayan çevrimiçi platform RStudio Cloud uygulamasını ziyaret edebilirsiniz. Bu işlemlerin nasıl yapılacağının anlatıldığı RStudio Cloud'da Çalışma bölümüne göz atabilirsiniz.

#### 2.1.2 Kurulumun Test Edilmesi

Kurulumlar tamamlandıktan sonra sisteminizin çalıştığından emin olmak adına RStudio'yu açın ve Konsol veya "Console" yazan yere tıklayarak basit bir kod yazın. Örneğin  $x = 3 + 4$  yazın ve ENTER'a tıklayın. Burada  $x$  nesnesine bir toplama işleminin sonucu atanmış oldu. Bu nesnenin **Environment** (Ortam) penceresi altında 7 değeriyle belirdiğine dikkat ediniz.

Herhangi bir hata almadığınızdan emin olduğunuzda bir sonraki bölüme geçebilirsiniz.

#### 2.1.3 İlk R Script Dosyasının Oluşturulması

Yazılan kodları ileride inceleyebilmek, üzerinde değişiklikler yapabilmek için R Script olarak adlandırılan belgeler üzerinde çalışmanız daha mantıklı olacaktır. Dikkat edeceğiniz üzere yukarıda konsola yazdığınız `x <- 3 + 4` komutu üzerinde bir değişiklik yapamayacaksınız. Yeni bir R Script belgesi yaratmak için ise RStudio'nun üstünde göreceğiniz panelden "File", "New File" ve "R Script" sırasıyla seçin. İlk R belgenizin açıldığını göreceksiniz.

R Script'lerde her bir satır kodu ayrı ayrı çalıştırmanız gerekiyor. Bunun için de üzerinde olduğunuz satırı çalıştırmak için Windows kullanıcısı iseniz **CTRL+ENTER**, MacOS kullanıyorsanız **CMD+ENTER** kombinasyonlarını kullanmalısınız.

### 2.1.4 R’da Paketler Hakkında

R paketleri fonksiyonlar, hazır kodlar veya veri setleri içerebilen farklı amaçlar göz önünde bulundurularak oluşturulmuş paketlerdir. R programlama dilini ilk indirdiğinizde birçok paket beraberinde indirilmiş şekildedir ve eğer kullanmak istediğiniz paket bu paketler arasında değilse onu da indirmek ve yüklemek oldukça basittir.

### 2.1.5 Gerekli Paketlerin İndirilip Yüklenmesi

Bu bölümde kitabın ilerleyen aşamalarında kullanılacak paketlerin indirilmesi ve yüklenmesi tamamlanacaktır.

- Web uygulaması oluşturmak için **shiny**,
- Daha güzel bir tablo görünümüne ulaşmak için **DT**,
- Veri manipülasyonu için **dplyr**,
- Veri görselleştirmesi için **ggplot2**,
- Tarih - zaman verilerinin manipülasyonları için ise **lubridate**,
- Daha ileri seviye veri manipülasyonları için **tidyr**,

paketleri kullanılacaktır. Paketleri indirmek için aşağıdaki kodu ilk R Script belgenizde çalıştırabilirsiniz. (İndirme ile alakalı kodları bir kere çalıştırmanız yetecektir.)

R’da her bir satırı ayrı ayrı çalıştırmanız gerektiğine dikkat ediniz. (Üzerinde olduğunuz satırı çalıştırmak için Windows kullanıcısı iseniz **CTRL+ENTER**, MacOS kullanıyorsanız **CMD+ENTER** kombinasyonlarını kullanmalısınız.) İndirmeler internet bağlantınızın durumuna göre 1-5 dakika arası sürebilir.

```
install.packages("shiny")
install.packages("DT")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("lubridate")
install.packages("tidyr")
```

İndirilen paketlerin yüklenmesi için ise kullanılması gereken kod aşağıda bulunabilir. (Bu kodları ise programı her açtığınızda tekrar uygulamanız gerekmektedir.)

```
library(shiny)
library(DT)
library(dplyr)
library(ggplot2)
library(lubridate)
library(tidyr)
```

Yüklemeler de tamamlandığında ilk Shiny uygulamanızı nasıl oluşturabileceğinizi anlatan kısma geçmeye hazırsınız.

## 2.2 İlk Shiny Uygulaması

### 2.2.1 Çevrimiçi Çalıştırma

Bu kitapta kullanılacak olan bütün örnek uygulamalar açık bir Github hesabında paylaşılacak ve her bir örneğin ardından o uygulamaya kendi bilgisayarlarınızdan göz atabilmek için gereken tek satırlık R kodu da altında paylaşılacaktır.

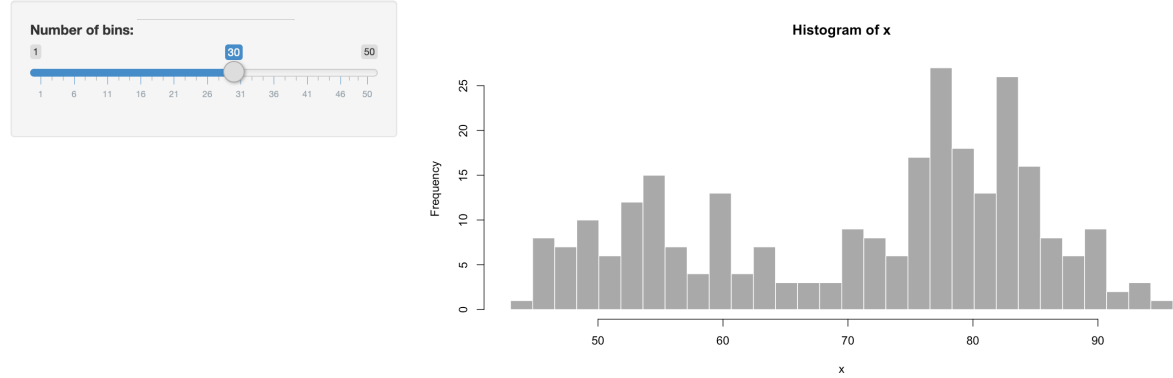
Örneğin birazdan oluşturacağımız ilk Shiny uygulamasına göz atmak için aşağıdaki kodu çalıştırabilirsiniz.

```
runGitHub("R-shiny", "acikenerji", subdir = "docs/barandogru_shinyBookExamples/first_shiny")
```

Burada kullanılan `runGitHub()` fonksiyonu `shiny` paketinin içinde gelen bir fonksiyondur. O nedenle çalışmaya başlamadan önce `library(shiny)` komutunu çalıştırdığınızdan emin olunuz.

Bu aşamaya kadar her şeyi doğru yaptıysanız karşınıza şu şekilde bir uygulama çıkacaktır.

### Old Faithful Geyser Data

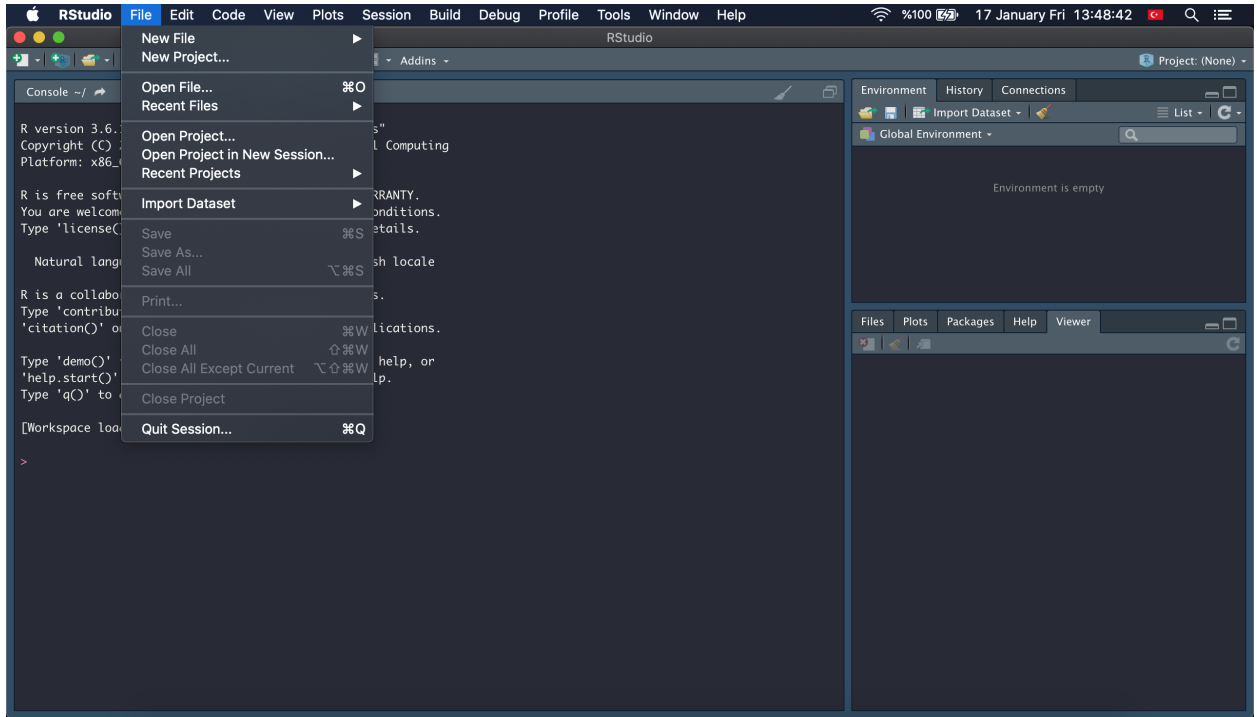


Şimdi ise bu uygulamayı sizin nasıl yaratabileceğinizi anlatalım.

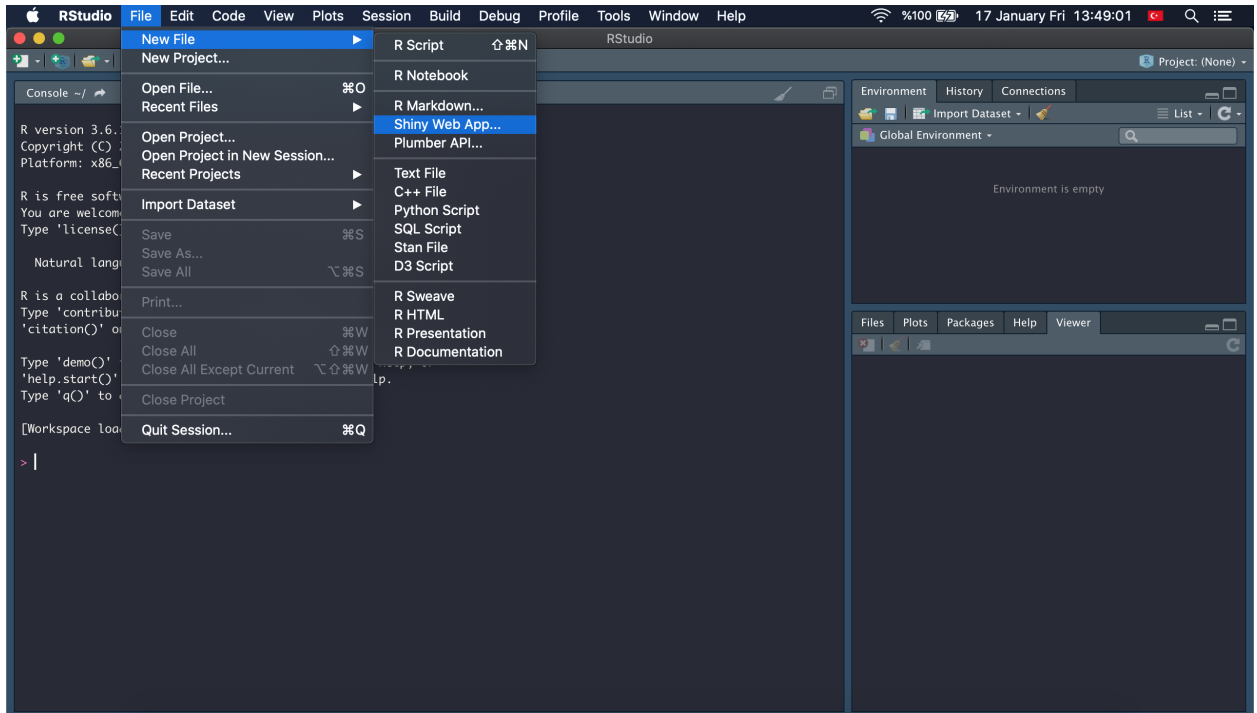
### 2.2.2 İlk Shiny Uygulamasının Oluşturulması

Yukarıda çalıştırdığımız uygulama, ilk R Shiny belgenizi oluşturduğunuzda karşınıza çıkan örnek bir şablon ve bu kısımda bu şablonu nasıl yaratabileceğinizi adım adım anlatacağız. Sonraki bölümlerde ise sıfırdan bir Shiny uygulamasını nasıl oluşturabileceğinizi ve uygulamanın genel hatlarını örnekler üzerinden göstereceğiz.

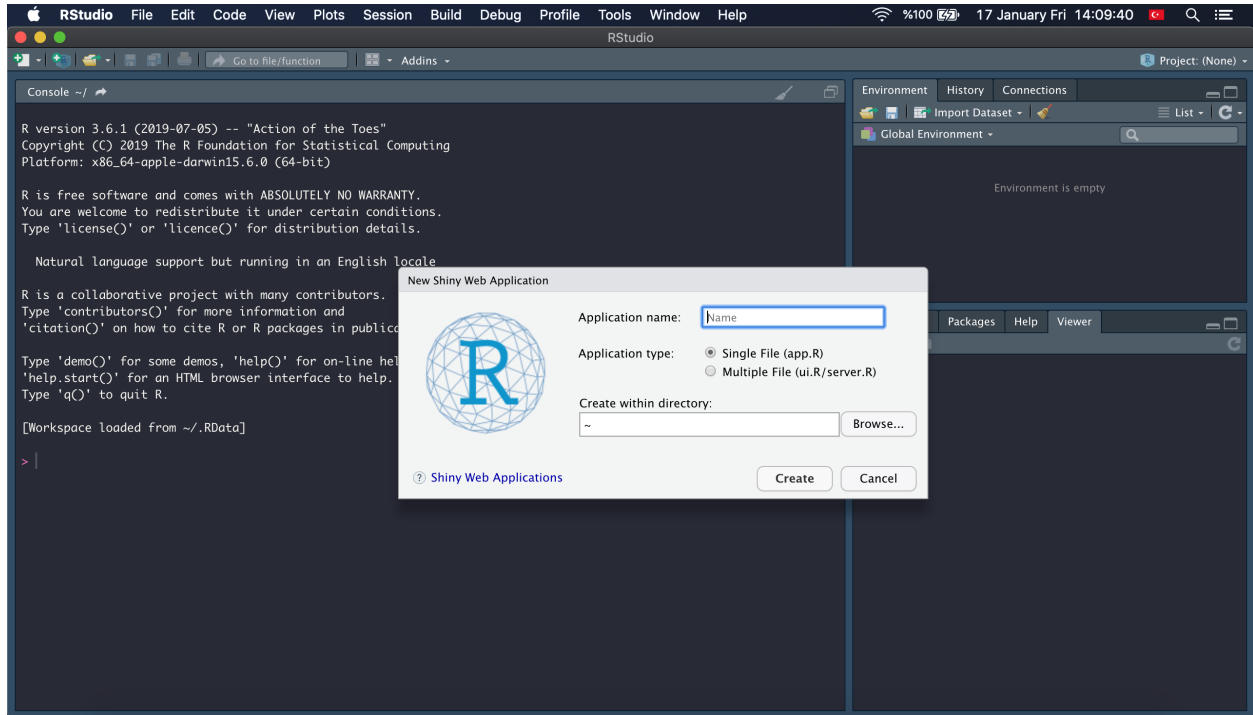
1. Bilgisayarınızda RStudio'yu açın.



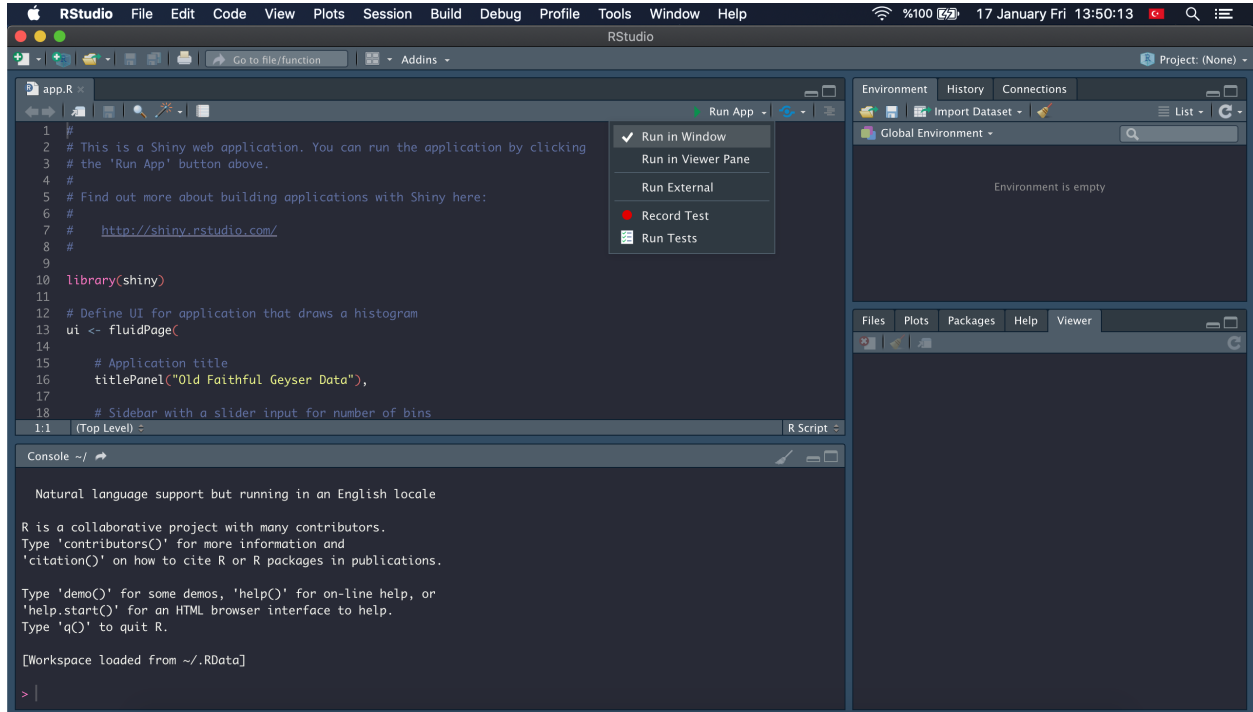
2. “File” -> “New File” -> “Shiny Web App” adımlarını uygulayın.



3. Karşınıza çıkan ekranda oluşturmak istediğiniz belgenin ismini ve çalışma dizinini belirleyin. (Burada belge seçeneklerinden “Single File (app.r)” seçeneğini seçmelisiniz.)



4. Oluşturulan belgenin sağ üst kısmında yeşil renkli okla belirtilen “Run App” butonuna tıklayın. İlk uygulamanızı çalıştırmış oldunuz :)



Bir sonraki bölümde bir Shiny uygulamasının genel hatları ve kullanışlı fonksiyonları adım adım örneklerle anlatılacaktır.

## 3 Shiny Uygulamasına Genel Bakış

Bu bölümde bir Shiny uygulamasının genel hatları ve önemli fonksiyonları detaylı şekilde anlatılacaktır. Her bir adımda gösterilen uygulamanın kodları paylaşılacak ve kullanılan yeni fonksiyonlar açıklanacaktır. Aynı zamanda uygulamaları çevrimiçi görüntüleyebilmemiz için de tek satırlık kodlar paylaşılacaktır.

### 3.1 Shiny Uygulamasının En Basit Hali

Uygulamanın açıklanmasına geçmeden önce uygulamaya göz atmanızı tavsiye ediyoruz. Uygulamayı çalıştırmak için aşağıdaki kodu çalıştırabilirsiniz. (`library(shiny)` komutunu bu aşamadan önce çalıştırdığınızdan emin olun.)

```
runGitHub("R-shiny", "acikenerji", subdir = "docs/barandogru_shinyBookExamples/almost_empty")
```

Karşınıza şu şekilde bir uygulama çıkmalı.

#### İlk Uygulama

İlk kenar çubuğu

İlk ana pano

Şimdi ise daha önce İlk Shiny Uygulamasının Oluşturulması bölümünde anlattığımız şekilde yeni bir `app.R` belgesi oluşturun ve belgenin içeriğini boşaltıp aşağıdaki kodu yapıştırdıktan sonra sağ üst kısımdan “Run App” butonuna tıklayın.

```
library(shiny)
# Kullanıcı arayüzünün tanımlanması
ui <- fluidPage(
  # Uygulama başlığının girilmesi
  titlePanel("İlk Uygulama"),
  # Kenar çubuğunun tanımlanması
  sidebarLayout(
    sidebarPanel(("İlk kenar çubuğu")),
    # Ana panonun tanımlanması
    mainPanel(("İlk ana pano"))
  )
)
```

```

)
)
# Arka planda işlemleri uygulayan "server"ın tanımlanması
server <- function(input, output) {
}
# Uygulamanın çalıştırılması
shinyApp(ui = ui, server = server)

```

Aynı uygulamayı çalıştırdığınızı göreceksiniz. Bu uygulama bir Shiny uygulamasının alabileceği en basit hal. Burada yapılan işlemleri açıklayacak olursak,

- Öncelikle `library(shiny)` komutu ile Shiny paketi yükleniyor.
- Sonra uygulama içerisinde `ui` ve `server` adında iki objenin içinde belirli özellikler tanımlanıyor. (Bu objeler ve tanımlanan özellikler ileride daha detaylı anlatılacaktır.)
- En sonda da “Run App” butonuna tıkladığınızda uygulamanın çalışmasını sağlayan `shinyApp(ui = ui, server = server)` komutu çalıştırılıyor.

Şimdi bir Shiny uygulamasının genel hatlarını inceleyelim.

## 3.2 Shiny Uygulamasının Genel Hatları

Shiny uygulaması temel olarak `ui` yani kullanıcının karşısına çıkan kullanıcı arayüzü (“User Interface”) ve `server` yani programın arka planında işlemleri gerçekleştiren kısımlarından oluşur.

`ui` kısmı kullanıcının uygulamayı açtığında karşısına çıkan her şey, `server` kısmı ise arka planda yapılan bütün operasyonların merkezi olarak adlandırılabilir. Bir Shiny uygulaması temel olarak 3 kısımdan oluşur. Bu üç kısmın iletişimleri ise temel olarak şu şekildedir.

“Yan Panel” kullanıcı girdilerinin (`input`) alındığı kısımdır. Oluşturulacak interaktif web sitesi için kullanıcıların seçimler yaptığı ya da metinler / sayılar girdiği bölümdür. Alınan bu inputlar sonrasında ise “Ana Panel”de gösterilecek olan çıktıyı (`output`) oluşturmak üzere “Server” fonksiyonuna gönderilirler.

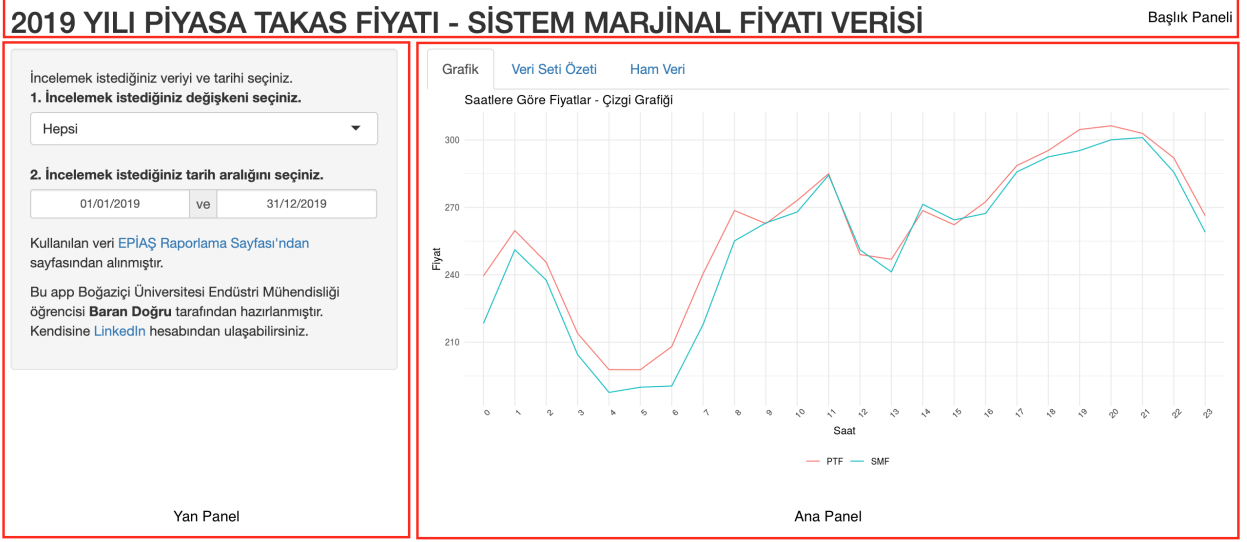
Yani özetle iletişimde aşağıdaki sıra izlenir.

$$YanPanel(SidebarPanel) \rightarrow Server \rightarrow AnaPanel(MainPanel)$$

Kullanıcı karşısına çıkan bir Shiny uygulaması ise “Yan Panel”, “Ana Panel” ve “Başlık Panel”inden oluşur. Bu düzen aşağıdaki ekran görüntüsünde daha rahat görülebilir.

“Server” kısmının kullanıcı arayüzünde direkt bir karşılığı yoktur.





Şimdi ise bu iletişimlerin sürdürülmesini sağlayan kısımları ve kodları inceleyelim.

### 3.2.1 Yan Panel

Yan Panel'de tanımlanabilecek çok sayıda girdi (input) çeşidi vardır. Bu input çeşitlerinden bir kısmını kullanılan fonksiyon isimleri ile birlikte sıralayacak olursak,

- Onay kutucukları koymak için `checkboxGroupInput()` fonksiyonu,
- Tarih girdisi için `dateInput()` ve tarih aralığı girdisi için `dateRangeInput()` fonksiyonu,
- Belirli opsiyonların açılan bir listede sıralanması için `selectInput()` fonksiyonu,
- Kaydırma çubuğuyla girdi almak için `sliderInput()`,
- Bir yazı girdisi almak için ise `textInput()` fonksiyonları kullanılabilir.

Daha fazla input fonksiyonu için shiny Cheatsheet dökümanına göz atabilirsiniz.

Şimdi bu fonksiyonların örneklendirildiği uygulamayı çalıştırmak için aşağıdaki kodu çalıştırabilirsiniz.

```
runGitHub("R-shiny", "acikenerji", subdir = "docs/barandogru_shinyBookExamples/just_input")
```

Karşınıza şu şekilde bir uygulama çıkacak,

## İlk Uygulama

İlk Yan Panel

İstediğiniz kutucuğu seçiniz.

☐ Seçenek1  
☐ Seçenek2

İstediğiniz tarihi seçiniz.

01/01/2019

İstediğiniz seçimi yapınız.

Seçenek1

Aşağıdaki çubuğu kullanarak seçiminizi yapınız.

01530

036912151821242730

Metin giriniz.

Buraya Metin

İlk Ana Panel

Bu uygulamanın R kodu ise aşağıdaki gibi,

```
library(shiny)
# Kullanıcı arayüzünün tanımlanması
ui <- fluidPage(
  # Uygulama başlığının girilmesi
  titlePanel("İlk Uygulama"),
  # Kenar çubuğunun tanımlanması
  sidebarLayout(
    sidebarPanel(("İlk Yan Panel"),
      checkboxGroupInput(inputId = "kutucuk", label = "İstediğiniz kutucuğu seçiniz.",
        choices = c("Seçenek1", "Seçenek2")),
      dateInput(inputId = "tarih", label = "İstediğiniz tarihi seçiniz.",
        min = "2019-01-01", max = "2019-12-31", value = "2019-01-01",
        format="dd/mm/yyyy", language="tr", weekstart=1),
      selectInput(inputId = "menü", label = "İstediğiniz seçimi yapınız.",
        choices = c("Seçenek1", "Seçenek2")),
      sliderInput(inputId = "kaydırmaCubugu",
        label = "Aşağıdaki çubuğu kullanarak seçiminizi yapınız.",
        min = 0, max = 30, value = 15),
      textInput(inputId = "yazi", label = "Metin giriniz.", placeholder = "Buraya Metin")
    ),
    # Ana panonun tanımlanması
    mainPanel(("İlk Ana Panel"))
  )
)
# Arka planda işlemleri uygulayan "server"ın tanımlanması
server <- function(input, output) {
}
```

```
# Uygulamanın çalıştırılması
shinyApp(ui = ui, server = server)
```

Şimdi bu kodun üzerinden beraber geçelim. Fark edilebileceği üzere Shiny Uygulamasının En Basit Hali bölümünden farklı olan tek kısım `sidebarPanel()` fonksiyonu, bu nedenle de uygulamanın herhangi bir çıktısı olmaması oldukça normal.

- Bu input fonksiyonlarının tamamının ortak iki parametresi var, `inputId` ve `label`. `inputId` parametresi alınan inputun daha sonra “Server” kısmında kullanılabilmesi için o inputa yalnızca kodun bildiği bir isim vermek, `label` parametresi ise kullanıcının karşısına çıkan mesajın oluşturulmasından sorumludur.
- Bu iki parametre dışında, `checkboxGroupInput()` ve `selectInput()` input fonksiyonlarının `choices` adlı ortak bir parametreleri vardır. Bu parametre kullanıcının karşısına çıkacak seçeneklerin belirlenmesini sağlar.
- `dateInput()` fonksiyonunun ise `min`, `max`, `value`, `format`, `language`, `format`, `weekstart` gibi birçok parametresini görüyoruz. Bu parametreler sırasıyla minimum değerin belirlenmesi, maksimum değerin belirlenmesi, başlangıç değerinin belirlenmesi, gösterilecek tarihin dilinin seçilmesi, gösterilecek tarihin formatının belirlenmesi ve gösterilecek tarihte haftanın ilk gününün hangi gün olacağının belirlenmesinden sorumludur.
- `sliderInput()` fonksiyonunun `min`, `max` ve `value` olmak üzere 3 daha parametresini görüyoruz. Bu parametreler sırasıyla kaydırma çubuğunun minimum, maksimum ve başlangıç değerlerini belirlememizi sağlar.
- `textInput()` fonksiyonunun ise ek olarak `placeholder` adında bir parametresini görüyoruz, bu parametre kullanıcı herhangi bir metin girmeden önce metin kutusunda yer alan “Yer Tutucu” metin olarak tanımlanabilir.

Bu fonksiyonların yukarıda bahsedilen parametreler dışında da pek çok parametresi olduğuna dikkat edilmelidir. Bu parametreler için shiny Cheatsheet dökümanına göz atabilirsiniz.

Şimdi ise bu inputların “Server” fonksiyonunda gerekli işlemlerden geçerek “Ana Panel”de nasıl yansıtıldığını inceleyelim. “Server” ve “Ana Panel” kısımları birbiriyle oldukça bağlantılı olduğundan bu kısımlar birlikte ele alınacaktır.

### 3.2.2 Server ve Ana Panel

Ana panelde birçok farklı çıktı formatı sergilenebilir. Bu farklı opsiyonlardan bahsetmeden önce “input”larla bu iki kısım arasındaki bağlantının nasıl çalıştığını incelemek adına örnek uygulamayı aşağıdaki kod yardımıyla çalıştırın.

```
runGitHub("R-shiny", "acikenerji", subdir = "docs/barandogru_shinyBookExamples/input_output")
```

Karşınıza şu şekilde bir uygulama çıkacak.

## Merhabalar,

Lütfen bilgilerinizi giriniz.

**Adınızı giriniz.**

**Doğum tarihinizi seçiniz.**

**Cinsiyetinizi seçiniz.**

Erkek

**Boyunuzu seçiniz. (cm)**

100 115 130 145 160 175 188 205 220 235 250

Bilgileriniz şu şekilde:  
Selam Baran Doğru :)  
Doğum Tarihiniz: 1998-06-19  
Cinsiyetiniz: Erkek  
Boyunuz: 188 cm

Siz de kendi bilgilerinizi girerek uygulamanın farklı çıktıları verdiğini gözlemleyebilirsiniz.

Şimdi ise bu uygulamanın koduna bakalım.

```
library(shiny)
# Kullanıcı arayüzünün tanımlanması
ui <- fluidPage(
  # Uygulama başlığının girilmesi
  titlePanel("Merhabalar,"),
  # Kenar çubuğunun tanımlanması
  sidebarLayout(
    sidebarPanel(("Lütfen bilgilerinizi giriniz. "),
      textInput(inputId = "yazi", label = "Adınızı giriniz.", placeholder = "Adınız"),
      dateInput(inputId = "tarih", label = "Doğum tarihinizi seçiniz.",
        min = "1975-01-01", max = "2019-12-31", value = "1975-01-01",
        format="dd/mm/yyyy", language="tr", weekstart=1),
      selectInput(inputId = "kutucuk", label = "Cinsiyetinizi seçiniz.",
        choices = c("Kadın", "Erkek", "Diğer"), selected = NULL),
      sliderInput(inputId = "kaydırmaCubugu",
        label = "Boyunuzu seçiniz. (cm)",
        min = 100, max = 250, value = 150)
    ),
    # Ana panonun tanımlanması
    mainPanel(("Bilgileriniz şu şekilde: "),
      textOutput("text1"),
      textOutput("text2"),
      textOutput("text3"),
      textOutput("text4")
    )
  )
)
```

```

)
# Arka planda işlemleri uygulayan "server"ın tanımlanması
server <- function(input, output) {
  output$text1 <- renderText(paste("Selam ", as.character(input$yazı), " :"))
  output$text2 <- renderText(paste("Doğum Tarihiniz: ", as.character(input$tarih)))
  output$text3 <- renderText(paste("Cinsiyetiniz: ", as.character(input$kutucuk)))
  output$text4 <- renderText(paste("Boyunuz: ", as.character(input$kaydırmaCubugu), " cm"))
}
# Uygulamanın çalıştırılması
shinyApp(ui = ui, server = server)

```

Burada bir daha “sidebarPanel” kısmındaki input alma süreci anlatılmayacaktır. “Input”lar alındıktan sonra “Server” içinde farklı **render** fonksiyonları ile (Bu örnekte yalnızca **renderText** fonksiyonu kullanılmıştır.) çağırılırlar. Bu çağırılma sırasında \$ operatörü ve daha önce “Input” kısmında **inputId** parametresine verilen isim kullanılır. Daha sonra bunlar kullanıcının seçeceği farklı isimdeki “output” değişkenlerine eşitlenirler.

En sonda ise, bu “output” değişkenleri farklı **output** fonksiyonları (Bu örnekte yalnızca **textOutput** fonksiyonu kullanılmıştır.) içinde çağırılarak “Ana Panel”de gösterilmeleri sağlanır. Buradaki çağırma sırasında ise **output** fonksiyonu ile kullanılan \$ operatörü sonrasında belirlenen isim kullanılır.

Temel olarak bir Shiny uygulamasının akışı şu şekilde ifade edilebilir.

*Input – > Render – > Output*

Şimdi ise farklı render fonksiyonlarını ve onlarla bağlantılı olan output fonksiyonlarını aşağıda inceleyelim.

render fonksiyonları	output fonksiyonları
renderText()	textOutput()
renderPrint()	verbatimTextOutput()
renderPlot()	plotOutput()
renderImage()	imageOutput()
renderTable()	tableOutput()

Bu fonksiyonlar ve daha fazlası hakkında daha detaylı bilgi için shiny Cheatsheet dökümanına göz atabilirsiniz.

Shiny uygulamaları kullanıcının hayal gücü ve istekleri doğrultusunda şekillenmektedir ve çok kompleks uygulamalar oluşturmak da mümkündür. Bu dökümanda temel bir giriş yapılması hedeflenmiş, bundan sonrası kullanıcıya bırakılmıştır. Herhangi bir sorunuz olması durumunda kitabın yazarına Yazar Hakkında kısmında belirtilen iletişim bilgisinden ulaşabilirsiniz.

## Ekler

### Yazar Hakkında

- *Baran Doğru*

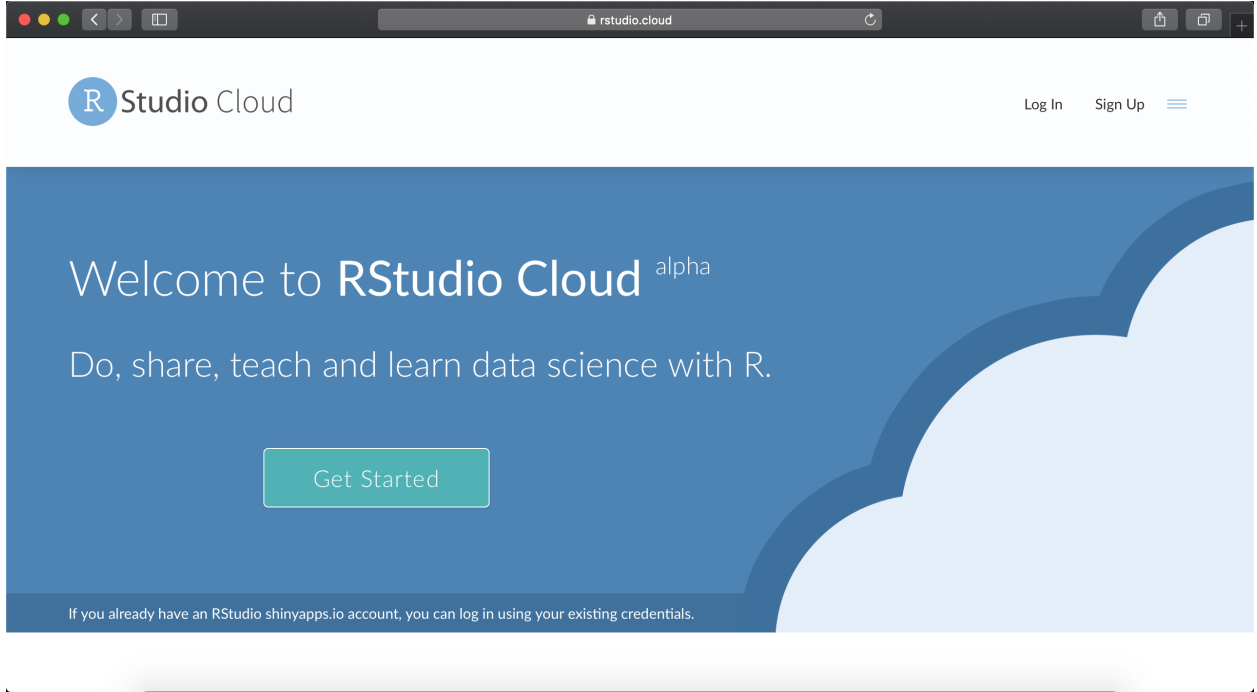
Baran Doğru, Boğaziçi Üniversitesi Endüstri Mühendisliği öğrencisidir. Veri bilimine büyük ilgi duyan Baran Doğru Algotopy’de staj yaptığı dönemde bu dökümanın oluşturulmasında büyük pay sahibi olmuştur. Kendisine ulaşmak için LinkedIn hesabını ziyaret edebilirsiniz.

## RStudio Cloud'da Çalışma

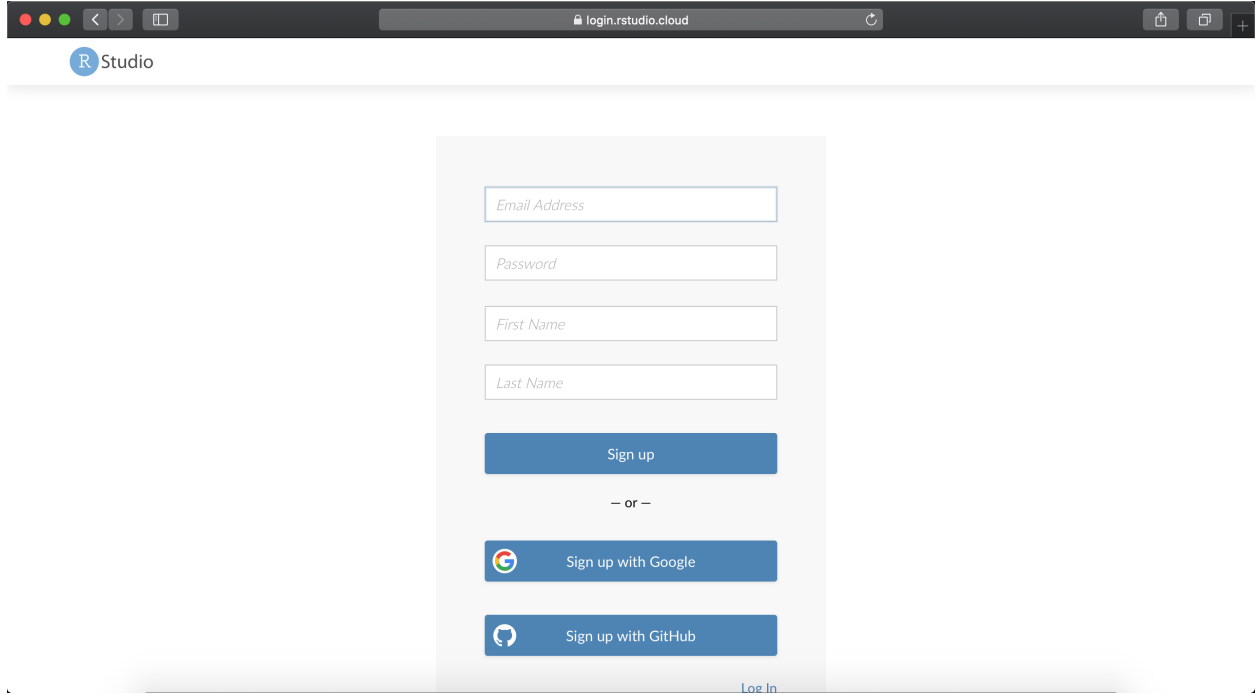
RStudio Cloud, R ve RStudio'yu bilgisayarınıza indirmeden çevrimiçi olarak kodlarınızı yazabileceğiniz, çalışmalarınızı is arkadaşlarınızla rahatça paylaşabileceğiniz, gerekli paketleri bilgisayarınıza yükleyip yüklemediğinizi dert etmeyeceğiniz tamamen ücretsiz bir platformdur.

RStudio Cloud'a erişmek ve platformu kullanmak için aşağıdaki adımları izleyebilirsiniz.

1. Tercih ettiğiniz web tarayıcıda “RStudio Cloud” yazarak aratılınca karşınıza çıkan ilk linke tıkladığınızda bu ekranla karşılaşacaksınız. Ana ekranın sağ üst köşesindeki “Sign Up” butonuna tıklayarak kayıt ekranına ulaşın.

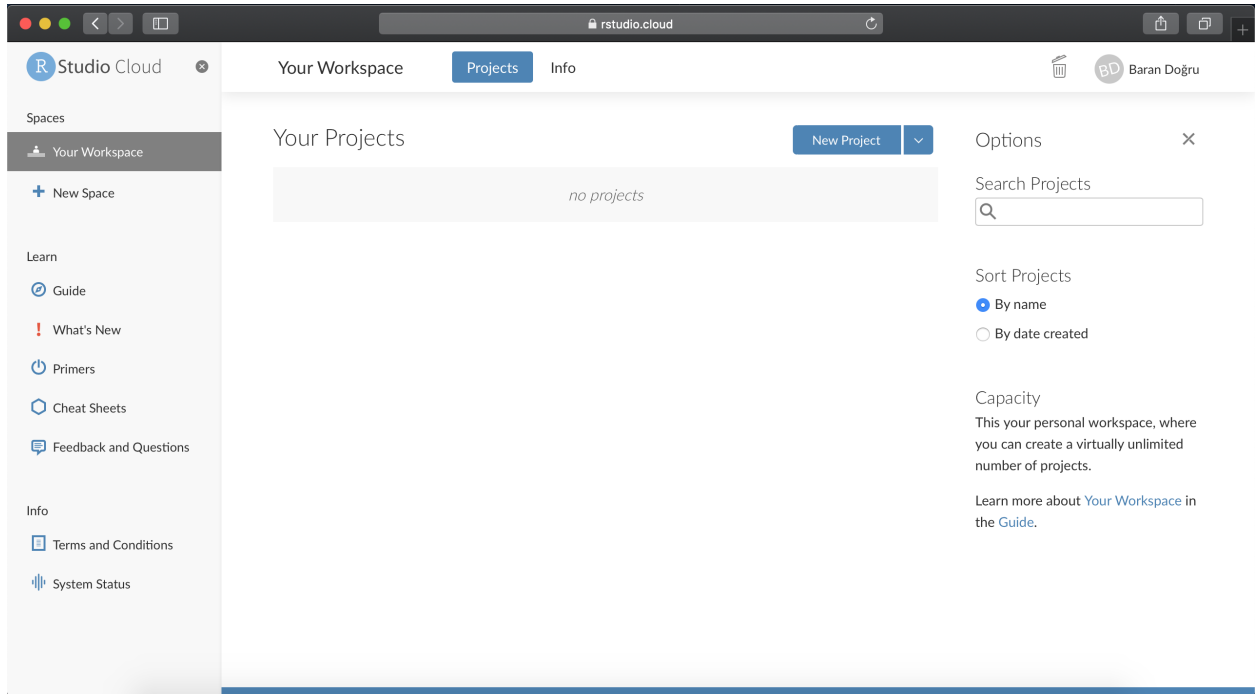


2. Karşınıza çıkan kayıt formunu doldurun.



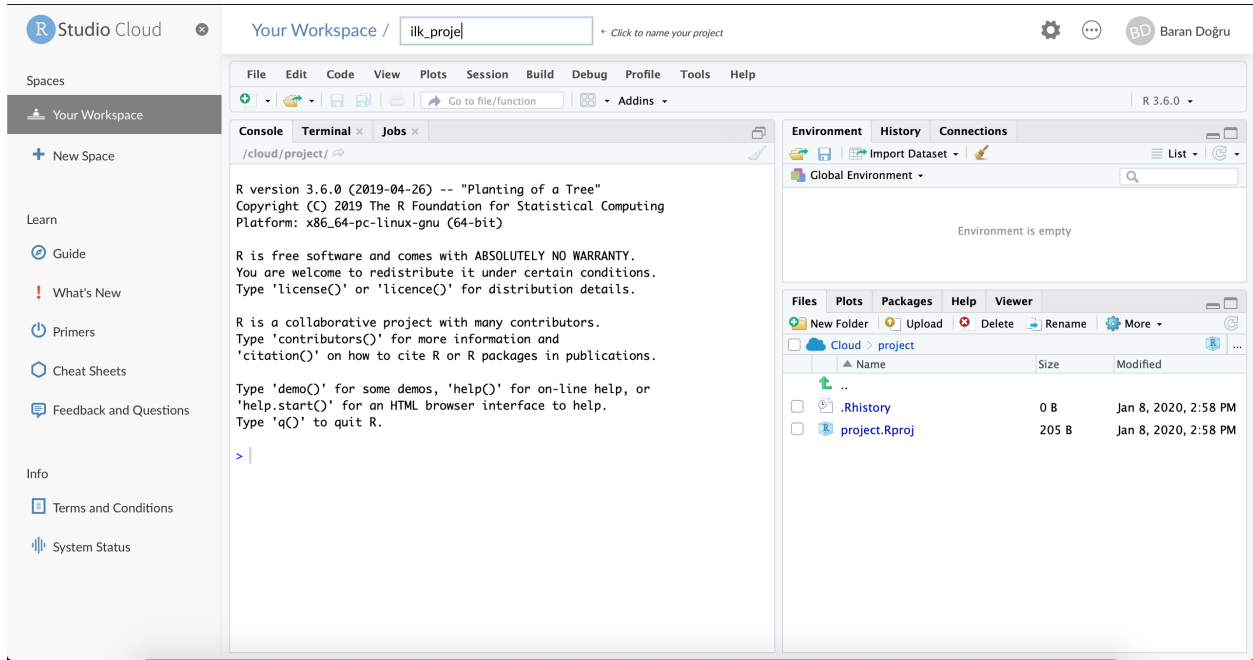
The image shows the RStudio Cloud login page. At the top, there's a browser window with the address bar showing 'login.rstudio.cloud'. Below the browser window, the RStudio logo is visible. The main content area is a light gray box containing a sign-up form. The form has four input fields: 'Email Address', 'Password', 'First Name', and 'Last Name'. Below these fields is a blue 'Sign up' button. Underneath the button is the text '-- or --'. There are two more buttons: 'Sign up with Google' (with the Google logo) and 'Sign up with GitHub' (with the GitHub logo). At the bottom right of the form, there is a 'Log In' link.

3. Kaydınızı tamamlayın ve hesabınıza giriş yapın. Artık RStudio Cloud'u kullanmaya hazırsınız.

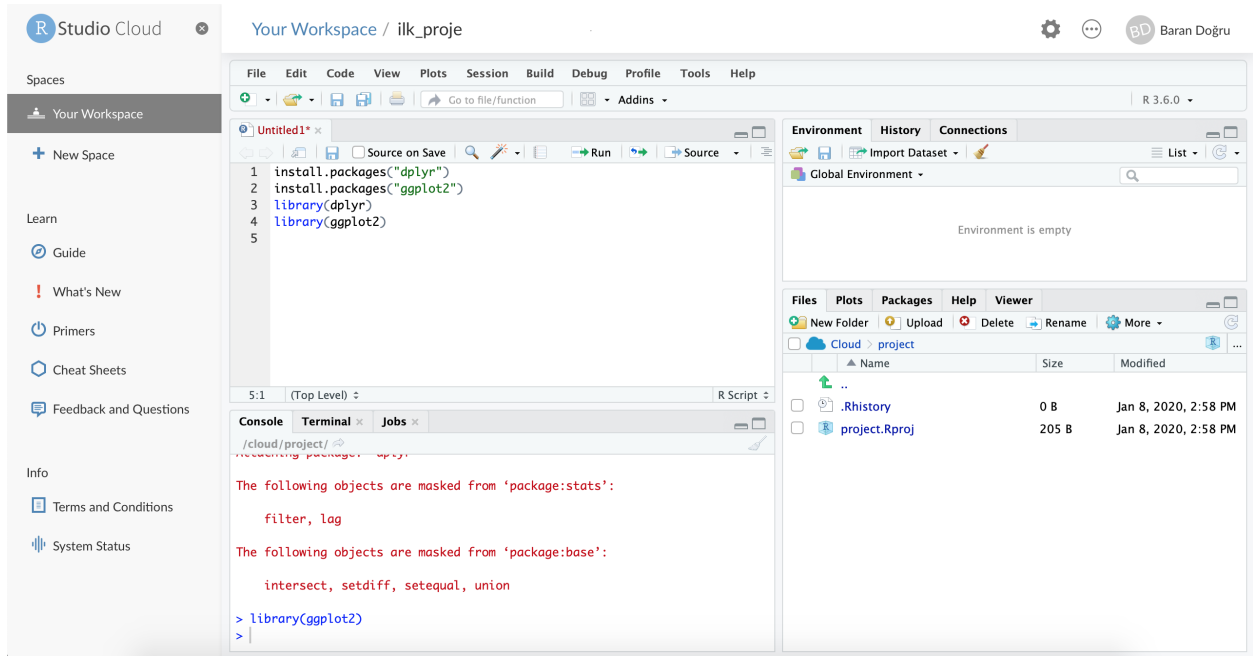


The image shows the RStudio Cloud dashboard. At the top, there's a browser window with the address bar showing 'rstudio.cloud'. Below the browser window, the RStudio Cloud logo is visible. The dashboard has a sidebar on the left with the following sections: 'Spaces' (with 'Your Workspace' selected), 'Learn' (with links to 'Guide', 'What's New', 'Primers', 'Cheat Sheets', and 'Feedback and Questions'), and 'Info' (with links to 'Terms and Conditions' and 'System Status'). The main content area is titled 'Your Projects' and has a 'New Project' button. Below the title, there's a message 'no projects'. On the right side, there's an 'Options' panel with a search bar for 'Search Projects', a 'Sort Projects' section with radio buttons for 'By name' (selected) and 'By date created', and a 'Capacity' section with text about the personal workspace and a link to the 'Guide'.

4. İlk projenizi oluşturmak için “New Project” butonuna tıklayın.



5. Açılan ilk projenize R Script dosyası eklemek için ise üstteki bardan “File”, “New File” ve “R Script” sırasıyla seçin. Artık R’da kod yazmaya hazırsınız.



## Örnek Projenin Kodu ve Çalıştırma Linki

Bu aşamada Motivasyon kısmında gösterilen uygulamanın kodu ve çalıştırma linki paylaşılacaktır. Bu örnek bu kitapta anlatılan seviyenin biraz üstünde olabilir, herhangi bir soru durumunda yazara ulaşmak için Yazar Hakkında bölümünde verilen iletişim bilgisi kullanılabilir.

- Uygulamayı bilgisayarınızda çalıştırmak için aşağıdaki kodu çalıştırabilirsiniz,



```
runGitHub("R-shiny", "acikenerji", subdir = "docs/barandogru_shinyBookExamples/final_project")
```

- Uygulamanın kodu ise aşağıdaki gibidir,

```
# gerekli paketlerin yüklenmesi
library(shiny)
library(dplyr)
library(tidyr)
library(ggplot2)
library(lubridate)
library(DT)

# verinin indirilmesi
ptfsmf <- readRDS(url("https://github.com/acikenerji/verianalizi101/blob/master/duzenlenmis_ptfsmf.rds?"))

# Kullanıcı arayüzünün tanımlanması
ui <- fluidPage(

  # Uygulama başlığının girilmesi
  titlePanel("2019 YILI PİYASA TAKAS FİYATI - SİSTEM MARJİNAL FİYATI VERİSİ"),

  # Kenar çubuğunun tanımlanması
  sidebarLayout(
    sidebarPanel(
      (title = "İncelemek istediğiniz veriyi ve tarihi seçiniz."),
      selectInput("variableIn", "1. İncelemek istediğiniz değişkeni seçiniz." ,
        choices=c("Hepsi", "PTF", "SMF"), selected="Hepsi"),
      dateRangeInput("dateIn",
        "2. İncelemek istediğiniz tarih aralığını seçiniz.",
        start=min(ptfsmf$Tarih),
        end=max(ptfsmf$Tarih),
        min=min(ptfsmf$Tarih),
        max=max(ptfsmf$Tarih),
        format="dd/mm/yyyy",
        separator="ve",
        language="tr",
        weekstart=1),
      p("Kullanılan veri ", tags$a(href = "https://rapor.epias.com.tr/rapor/xhtml/ptfSmfListeleme",
        "EPİAŞ Raporlama Sayfası'ndan",
        target = "_blank"),
        " sayfasından alınmıştır."),
      p("Bu app Boğaziçi Üniversitesi Endüstri Mühendisliği öğrencisi",
        strong("Baran Doğru"), "tarafından hazırlanmıştır.",
        "Kendisine", tags$a(href = "https://www.linkedin.com/in/barandogru", "LinkedIn",
        target = "_blank"),
        "hesabından ulaşabilirsiniz.")
    ),

    # Ana panonun tanımlanması
    mainPanel(
      tabsetPanel(type = "tab",
        tabPanel("Grafik", plotOutput("plot1")),
        tabPanel("Veri Seti Özeti", verbatimTextOutput("summary")),
        tabPanel(title = "Ham Veri", dataTableOutput("data", height = "500px"))
      )
    )
  )
)
```

```

    ),
  )
)

# Arka planda işlemleri uygulayan "server"ın tanımlanması
server <- function(input, output) {

  # grafik çıktısının tanımlanması
  output$plot1 <- renderPlot({

    # verinin grafik için düzenlenmesi
    ptfsmf <- ptfsmf %>% mutate(Saat = hour(Tarih), Tarih = as.Date(Tarih)) %>%
      pivot_longer(cols = PTF:SMF, names_to = "Degisken", values_to = "Fiyat")

    pl_df <- ptfsmf %>%
      filter(Tarih >= input$dateIn[1] & Tarih <= input$dateIn[2]) %>%
      group_by(Degisken, Saat) %>%
      summarise(Fiyat = mean(Fiyat))

    if(input$variableIn != "Hepsi"){
      pl_df <- pl_df %>% filter(Degisken == input$variableIn)
      ggplot(pl_df, aes(x=Saat, y=Fiyat, color=Degisken)) +
        geom_line() +
        theme_minimal() +
        scale_x_discrete(limits=c(0:23)) +
        labs(title = "Saatlere Göre Fiyatlar - Çizgi Grafiği") +
        theme(axis.text.x = element_text(angle = 45), legend.position = "bottom",
              legend.title = element_blank())
    }
    else{
      ggplot(pl_df, aes(x=Saat, y=Fiyat, color=Degisken)) +
        geom_line() +
        theme_minimal() +
        scale_x_discrete(limits=c(0:23)) +
        labs(title = "Saatlere Göre Fiyatlar - Çizgi Grafiği") +
        theme(axis.text.x = element_text(angle = 45), legend.position = "bottom",
              legend.title = element_blank())
    }

  })

  # tablo çıktısının tanımlanması
  output$data <- renderDataTable({
    ptfsmf$Tarih <- as.character(ptfsmf$Tarih)

    ptfsmf <- ptfsmf %>% filter(Tarih >= input$dateIn[1] & Tarih <= input$dateIn[2]) %>%
      select(Tarih, PTF, SMF)

    if(input$variableIn != "Hepsi"){
      ptfsmf <- ptfsmf %>% select(Tarih, input$variableIn)
      datatable(ptfsmf)
    }
  })
}

```

```

    }
    else
      datatable(ptfsmf)
  })

  # özet çıktısının tanımlanması
  output$summary <- renderPrint({

    ptfsmf <- ptfsmf %>%
      mutate(Tarih_temp = as.Date(Tarih)) %>%
      filter(Tarih_temp >= input$dateIn[1] & Tarih_temp <= input$dateIn[2]) %>%
      select(-c(Tarih_temp, NDF, PDF))

    if(input$variableIn != "Hepsi"){
      ptfsmf <- ptfsmf %>% select(Tarih, input$variableIn)
      summary(ptfsmf)
    }
    else
      summary(ptfsmf)

  })

}

# Uygulamanın çalıştırılması
shinyApp(ui = ui, server = server)

```