# R ile Veri Görselleştirme

Dinçer GÖKSÜLÜK, Phd.

13 Mayıs 2022

## Contents



Kurs Öncesi Gereksinimler:

- **R 4.2.0** sürümü.
- **RStudio Desktop 2022.02.2+485** veya üzeri sürümler.
- MAC Kullanıcılar için **XQuartz** güncel sürümü.

## Giriş

R programının güçlü yönlerinden birisi verilerin görselleştirilmesi için sunmuş olduğu imkanlardır. R yazılımı grafikler konusunda oldukça gelişmiş ve etkin özelliklere sahiptir. Bu özelliklerin ek paketler ile (kütüphaneler) sunuluyor olması sayesinde grafik kütüphaneleri her geçen gün daha fazla gelişmekte ve mevcut kütüphanelere yeni özellikler eklenmektedir.

Veri görselleştirmede neden **R** tercih edilmelidir?

- Çok fazla sayıda elektronik/basılı kaynak, internet sayfaları ve tartışma forumları
- Baskı kalitesi yüksek grafikler
- Bilimsel kabul edilebilirliği yüksek
- Kolay programlanabilir grafik kodları
- Tamamen açık kaynak kodlar
- LaTeX & Sweave gibi bilimsel yazım araçları ile tam entegrasyon
- Çok sayıda grafik kütüphaneleri ve web tabanlı yazılımlar ile uyumluluk
- **Ücretsiz** yazılım
- Grafikleri bir çok farklı formatta kaydedebilme imkanı (png, jpeg, bmp, tiff, pdf, svg, postscript, . . . )
- . . .

R yazılımında onlarca grafik kütüphanesi mevcuttur. Temel kurulum ile gelen kütüphaneler basit grafikler oluşturmak için (histogram, scatter, boxplot, . . . ) yeterli imkanlar sunmaktadır. Ancak, daha gelişmiş ve kişiselleştirilmiş grafikler oluşturabilmek için alternatif kütüphanelerden yararlanmak gerekmektedir.

- Temel grafik kütüphaneleri (R Base Graphics, Low-level infrastructure)
  - graphics
  - grid (Manual, Book)
- İleri düzey grafik kütüphaneleri (High-level infrastructure)
  - lattice

  - ggplot2
  - gridSVG
  - scatterplot3d
  - RColorBrewer
  - ...
- İnteraktif grafikler
  - **highcharter**: (JavaScript HighCharts kütüphanesini kullanır. **Ticari kullanımda ücretlidir.**). Oldukça geniş bir kullanım kılavuzuna sahiptir. Çok fazla sayıda örnek grafik kodlarını internet ortamında bulmak mümkündür. (http://www.highcharts.com)
  - **plotly**: JavaScript kütüphaneleri kullanan bir diğer interaktif grafik kütüphanesidir. **Ticari kullanımda ücretlidir.** Diğer kütüphanelere göre halen gelişim aşamasındadır (https://plot.ly).
  - **googleVis**: Google tarafından sağlanan interaktif grafikleri kullanan bir R kütüphanesidir (https://cran.r-project.org/web/packages/googleVis/index.html).
  - **igraph**
  - **rgl**
  - **GGobi**
  - ...
- **Veri Seti**: MPV
- **Veri Ortamı**: CSV (virgülle ayrılmış)

```
# Gerekli R kütüphanelerinin yüklenmesi
library(readr)
library(ggplot2)
library(dplyr)
library(magrittr)
```

```
## MPV verisinin yüklenmesi
# mpv <- read.csv(file = "data/MPV.csv", header = TRUE, sep = ",", dec = ".", stringsAsFactors = TRUE)

mpv <- mpv_yedek <- read_csv(file = "data/MPV.csv", col_names = TRUE)
#mpv <- read_csv(file = file.choose(), col_names = TRUE)
```

```
# Değişken kodlamaları.
# cinsiyet -- 0: Kadın,  1: Erkek
# grup -- 1: Tedavi,  0: Kontrol
# Exitus: 1: Ex,  0: Sağ
# Nuks: 1: Evet,  0: Hayır

# Kategorik değişkenler için etiket tanımlamaları
mpv <- mpv %>%
  mutate(
    cinsiyet = factor(cinsiyet, levels = c(0, 1), labels = c("Kadın", "Erkek")),
    grup = factor(grup, levels = c(0, 1), labels = c("Kontrol", "Tedavi")),
    Exitus = factor(Exitus, levels = c(0, 1), labels = c("Sağ", "Ex")),
    Nuks = factor(Nuks, levels = c(0, 1), labels = c("Hayır", "Evet")),
  )
```

# Statik Grafikler

## Temel Grafik Kütüphaneleri (`graphics`)

- Kullanımı basittir ancak temel grafikler ile sınırlıdır.
- R kurulumu ile birlikte gelmektedir. Ayrıca bir kütüphane kurulumuna gerek yoktur.

| Fonksiyon | İşlev |
|---|---|
| `plot(...)` | İki değişkenli (x-y plot) grafik çizme işlemi için **generic S3 fonksiyon**dur. |
| `barplot(...)` | Çubuk grafik |
| `boxplot(...)` | Kutu-çizgi grafiği |
| `hist(...)` | Histogram |
| `pie(...)` | Pasta dilimleri grafiği |
| `dotchart(...)` | Nokta grafiği |
| `qqnorm`, `qqline`, `qqplot` | Q-Q grafiği |
| `image`, `heatmap`, `contour`, `persp` | Diğer grafikler |
| ... | ... |

### Bazı temel grafikler (histogram, boxplot, ...)

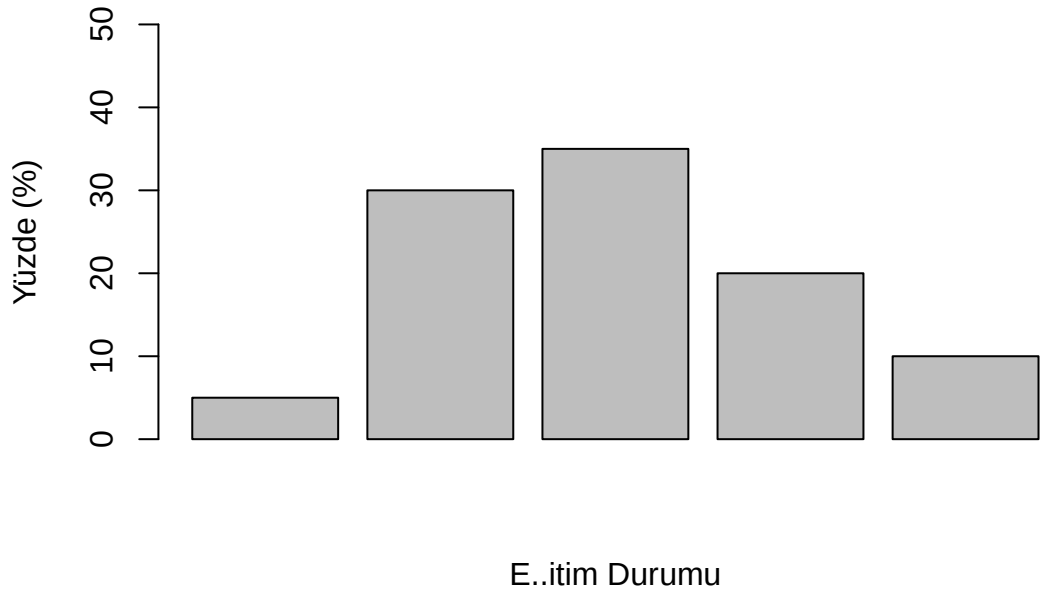
R programı açılışta `graphics` kütüphanesini otomatik olarak yüklemektedir. Bu nedenle tekrar `library(graphics)` yazarak kütüphanenin aktif edilmesine gerek yoktur.

```
## Grafiklerde kullanılacak bir diğer veri seti: Diamond verisi.
data(diamonds)
?ggplot2:::diamonds      ## Diamonds  verisi hakkında detaylı bilgi için
```

**a) Çubuk grafik (barplot)**:

```
barplot(diamonds[ ,"color"])  ## Hata: 'height' must be a vector or a matrix

# Barplot çizilecek olan verinin her bir çubuğa karşılık gelen sayı veya yüzdeleri
# verecek şekilde düzenlenmesi gerekmektedir.

barplot(c(5, 30, 35, 20, 10), xlab = "Eğitim Durumu", ylab = "Yüzde (%)")
```

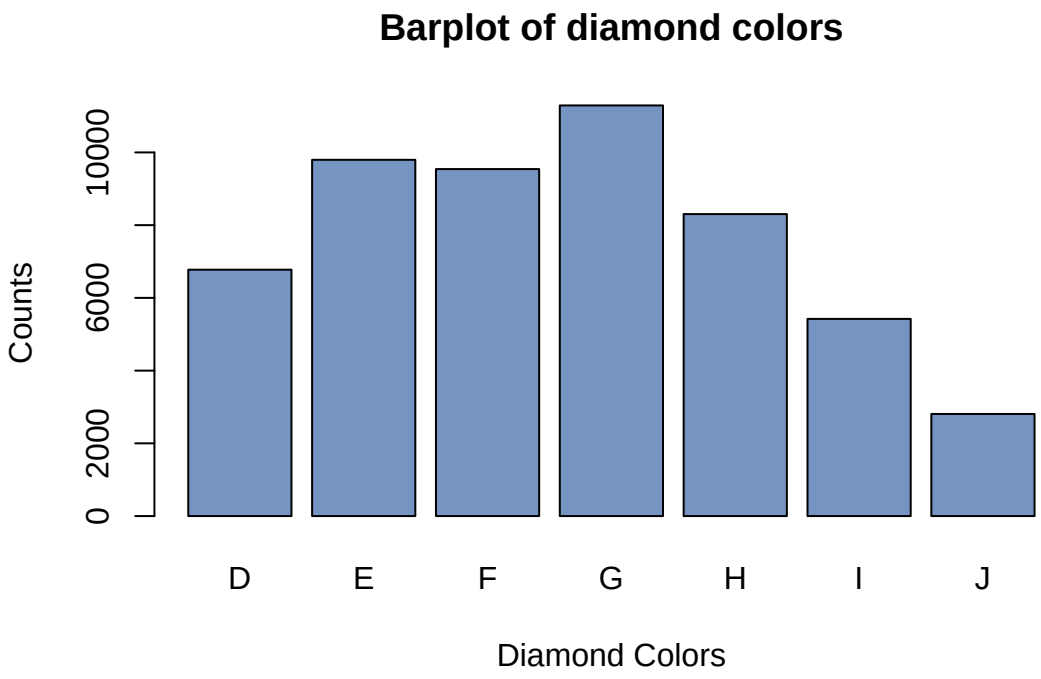



- Diamonds verisinde **color** değişkeni `barplot(...)` için uygun formata getirilmelidir.

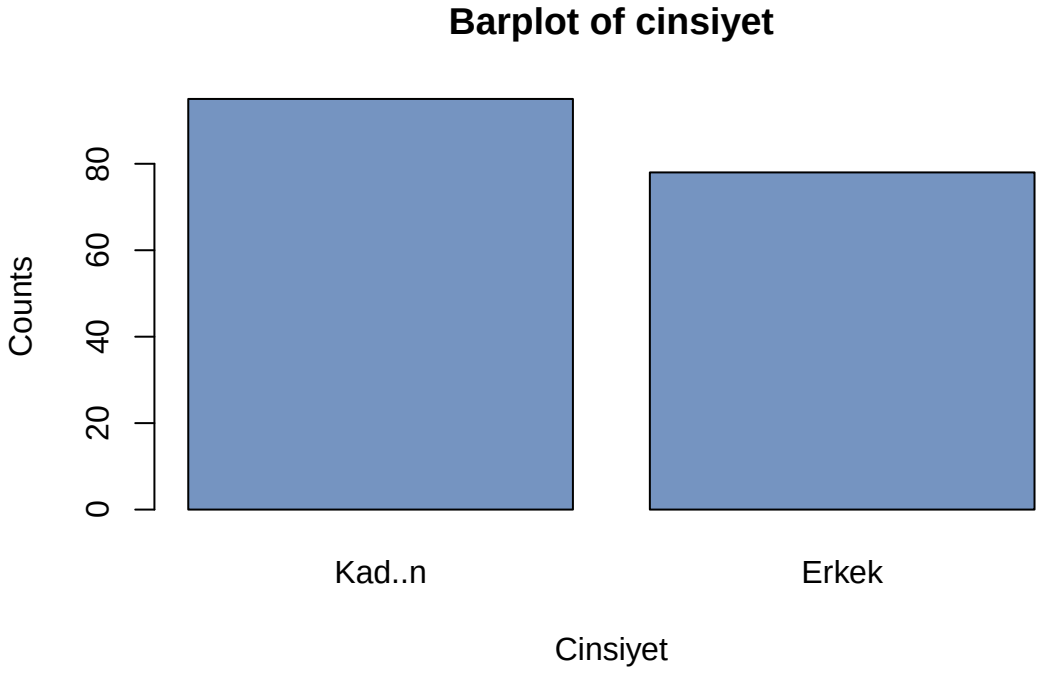
```
bardata <- diamonds %$%
  table(color)

print(bardata)
```

```
## color
##     D     E     F     G     H     I     J
##  6775  9797  9542 11292  8304  5422  2808
```

```
barplot(bardata, main = "Barplot of diamond colors", ylab = "Counts", xlab = "Diamond Colors",
        col = rgb(0.1, 0.3, 0.6, alpha = 0.6))
```




```
tbl <- table(mpv$cinsiyet)

barplot(tbl, main = "Barplot of cinsiyet", ylab = "Counts", xlab = "Cinsiyet",
        col = rgb(0.1, 0.3, 0.6, alpha = 0.6))
```



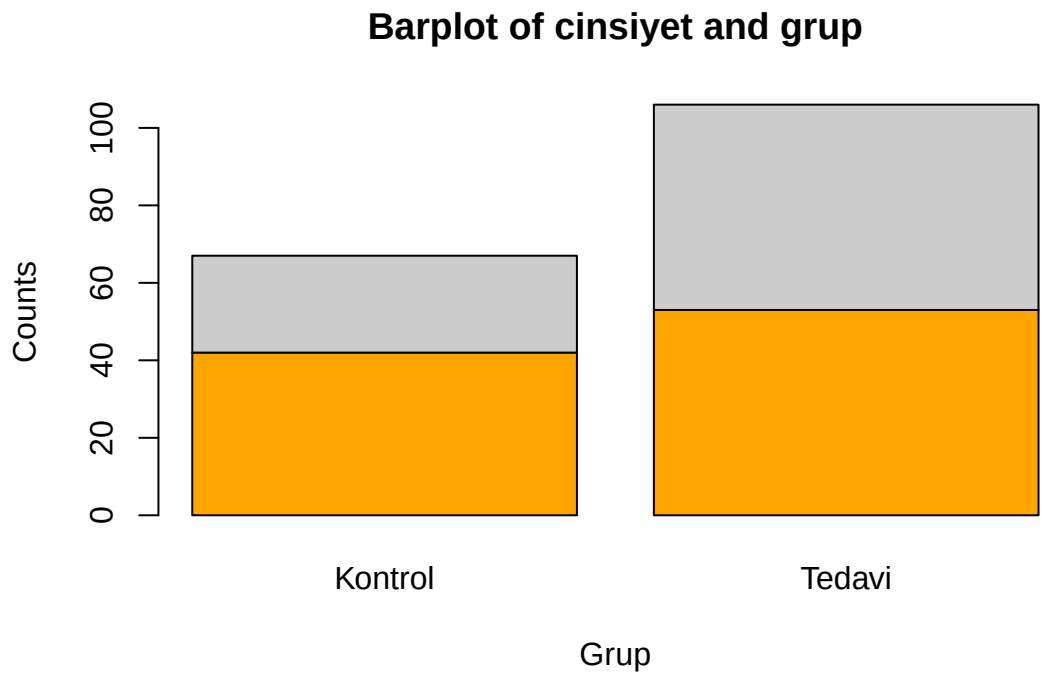
```
# Cinsiyet için çizilen grafiği y ekseninde yüzdeler olacak şekilde düzenleyiniz.
```

- Gruplandırılmış/Bindirilmiş Çubuk Grafikleri

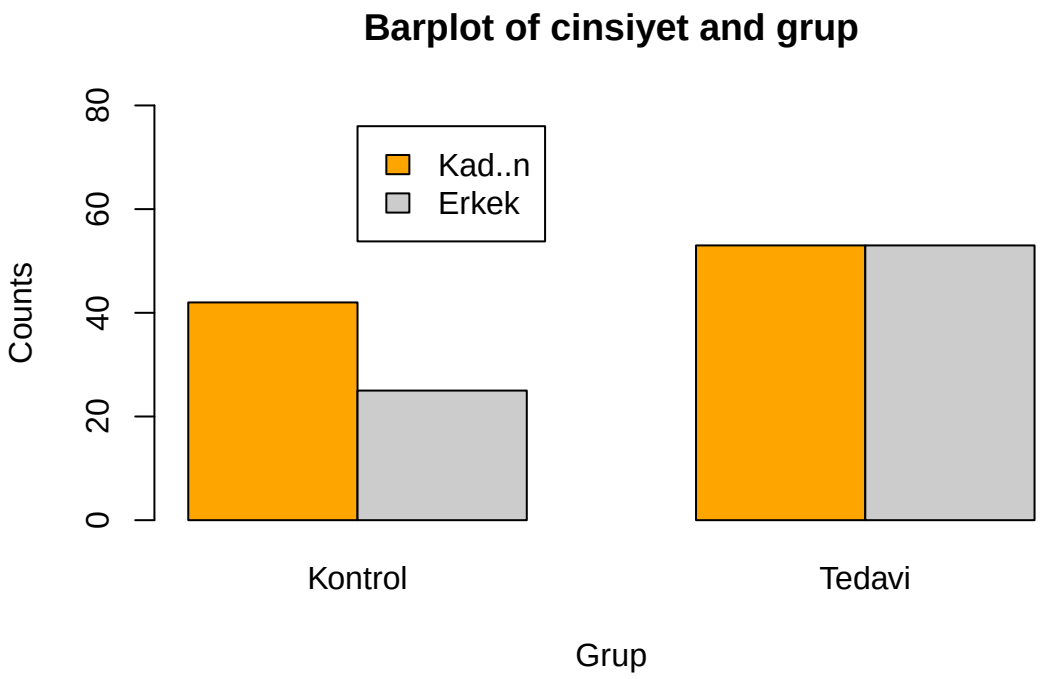
```
tbl <- with(mpv, table(cinsiyet, grup))
tbl
```

```
##         grup
## cinsiyet Kontrol Tedavi
##    Kadın      42     53
##    Erkek      25     53
```

```
## Bindirmeli Çubuk Grafik (Stacked)
barplot(tbl, main = "Barplot of cinsiyet and grup", ylab = "Counts",
        xlab = "Grup", col = c("orange", "gray80"))
```



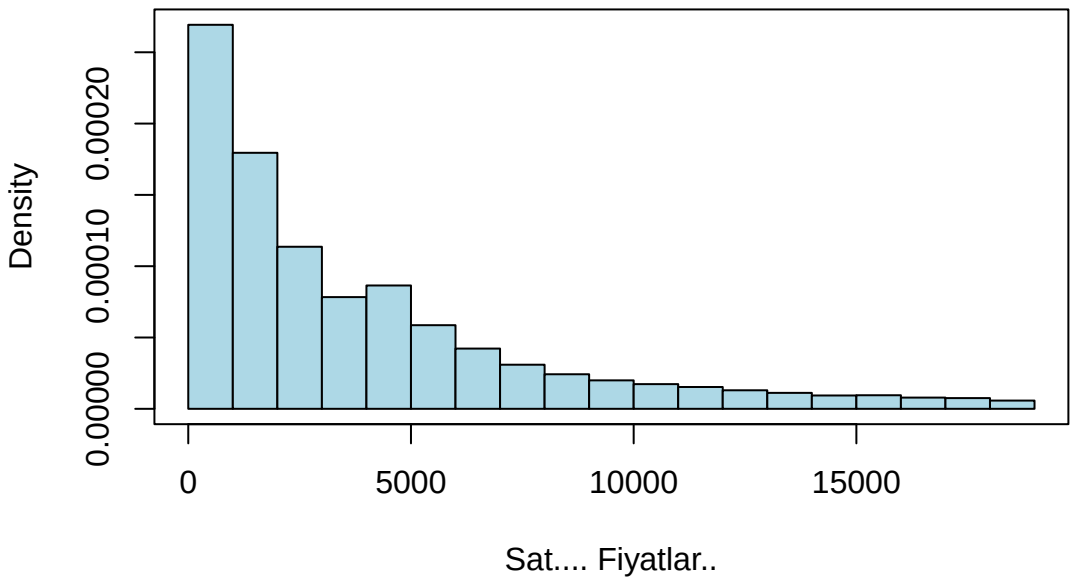

```
# Gruplandırılmış Çubuk Grafik
barplot(tbl, main = "Barplot of cinsiyet and grup", ylab = "Counts",
        xlab = "Grup", beside = TRUE, col = c("orange", "gray80"),
        ylim = c(0, 80))

#
legend(x = 2, y = 76, legend = c("Kadın", "Erkek"), fill = c("orange", "gray80"))
```




```
## Yukarıdaki iki grafiği yüzdeler ile çiziniz.
```

**b) Histogram (`hist(...)`):**

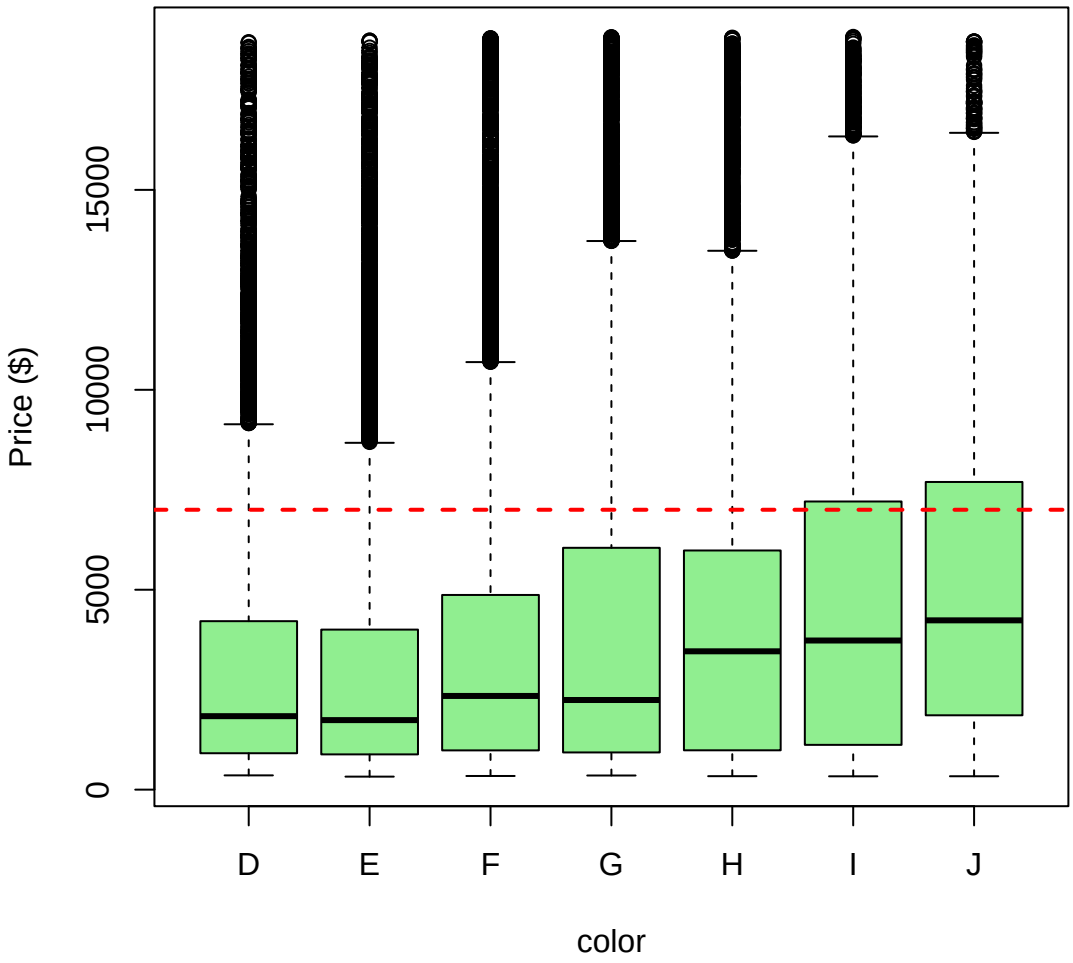
```
hist(diamonds$price, freq = FALSE, col = "lightblue",
     main = NULL, xlab = "Satış Fiyatları")
box()
```




**c) Boxplot (`boxplot(...)`):**

- Boxplot grafikleri tek değişkenli olarak veya bir grup değişkenine göre gruplandırılmış olarak çizilebilir.

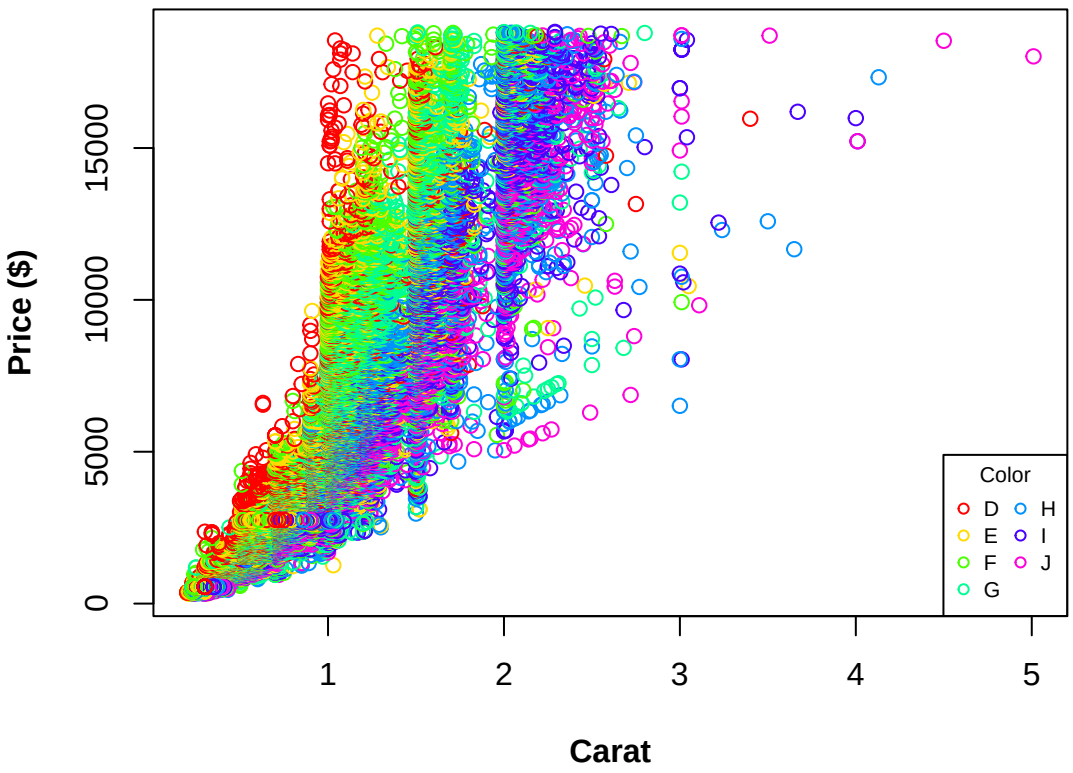
```
boxplot(price ~ color, data = diamonds, col = "lightgreen",
        ylab = "Price ($)", outline = TRUE)    ## outline = FALSE ise outlier değerler grafikte gösteri

## 7000$ noktasından yatay bir referans çizgisi ekleyelim:
abline(h = 7000, lty = 2, col = "red", lwd = 2)
```




**d) Saçılım Grafiği / Saçılım Matris Grafiği (`plot(...)`, `pairs(...)`):**

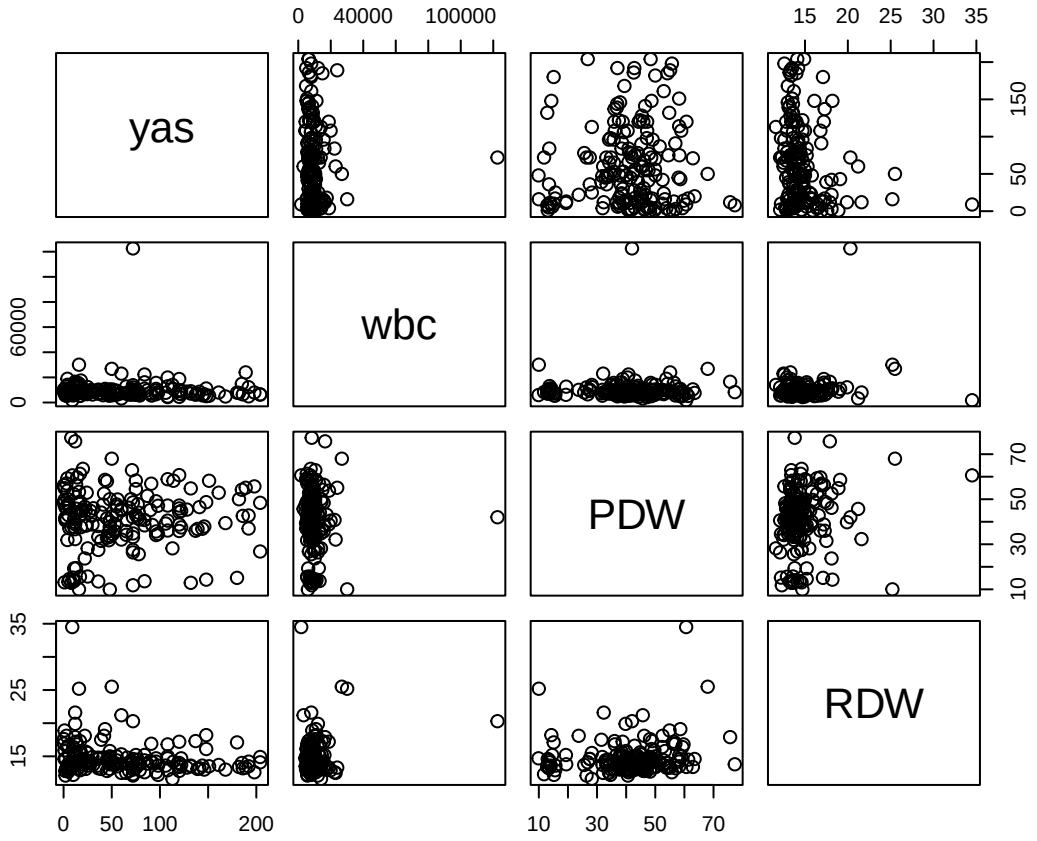
- İki değişken için saçılım grafiği `plot(...)` fonksiyonu kullanılarak çizilebilir.
- İkiden fazla değişken için matris saçılım graifği ise `plot(...)` veya `pairs(...)` fonksiyonları ile çizilebilir.

```
point.colors <- rainbow(7)

with(diamonds, {
  plot(carat, price, xlab = "Carat", ylab = "Price ($)", font.lab = 2,
       col = point.colors[as.numeric(diamonds$color)])
})
legend("bottomright", legend = levels(diamonds$color), col = point.colors, pch = 21,
       cex = 0.7, title = "Color", ncol = 2)
```




```
## Sayısal değişkenler için matrix saçılım grafiği:
## İşlem süresinin fazla olmasından dolayı "diamonds" verisi yerine "MPV" verisi kullanılmıştır.

# pairs(mpv[ ,c("yas", "wbc", "PDW", "RDW")])
plot(mpv[ ,c("yas", "wbc", "PDW", "RDW")], pch = 21, cex = 1.3)
```

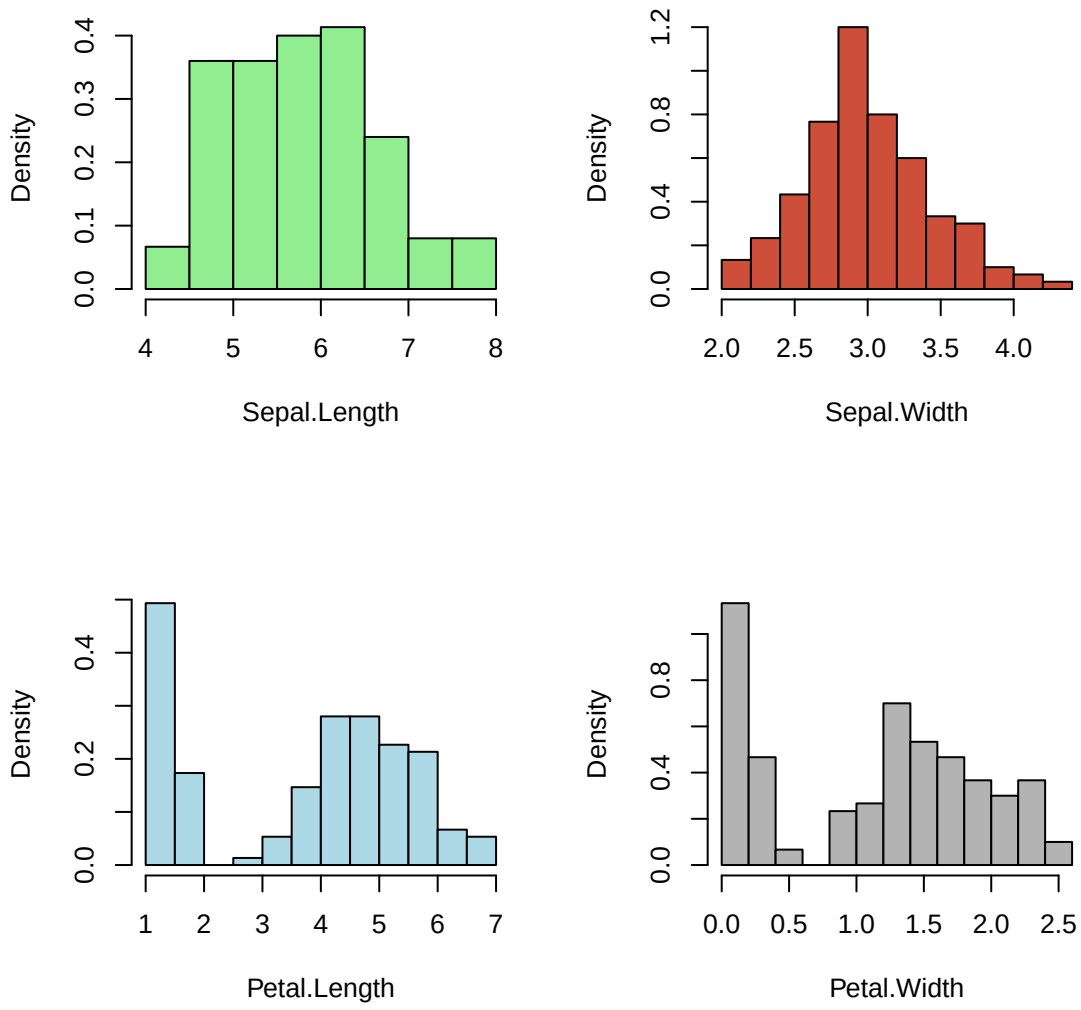
 




**Çoklu Grafikler ve Grafik Özelliklerinin Düzenlenmesi (`par()` fonksiyonu)**

Birden fazla grafiğin birlikte çizilebilmesi için **mfrow** grafik parametresi kullanılır. Bu işlem grafik çizilmeden önce `par(...)` fonksiyonu ile tanımlanır. Temel grafik kütüphanesinde yapılabilecek diğer opsiyonlar için `?par` içeriğine bakınız.

```
data(iris)

# iris verisinin sayısal değişkenlerini 2x2'lik bir yapıda oluşturalım.
par(mfrow = c(2, 2))    # mfrow = c(nrow, ncol)

vNames <- colnames(iris)[-5]  # "Species"" değişkeni çıkartılıyor.
                              # Kalan değişkenlerin isimleri vNames adı ile saklanıyor.

histColors <- c("lightgreen", "tomato3", "lightblue", "gray70")

for (i in 1:length(vNames)){
  hist(iris[ ,vNames[i]], col = histColors[i], main = NULL, xlab = vNames[i], freq = FALSE)
}
```

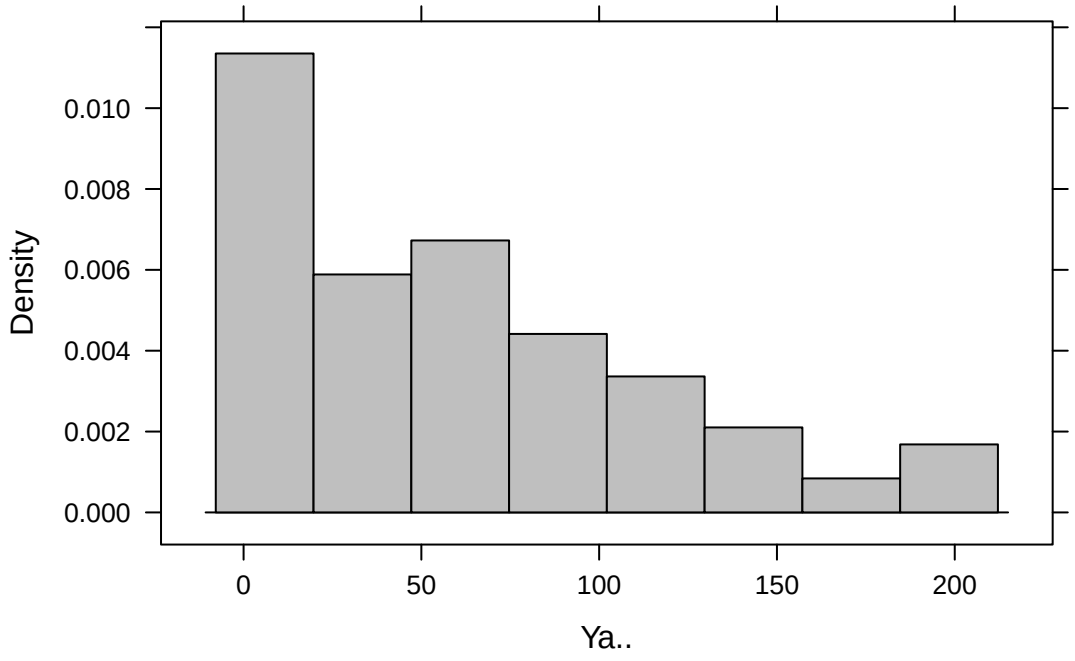



## lattice kütüphanesi

- Kod yapısı `graphics` kütüphanesine benzerdir.
- İleri düzey grafikler çizmeye imkan tanıyan güçlü bir kütüphanedir.
- İnternet ortamında veya basılı olarak çok sayıda doküman ve örnek bulmak mümkündür
- Kullanım kılavuzu

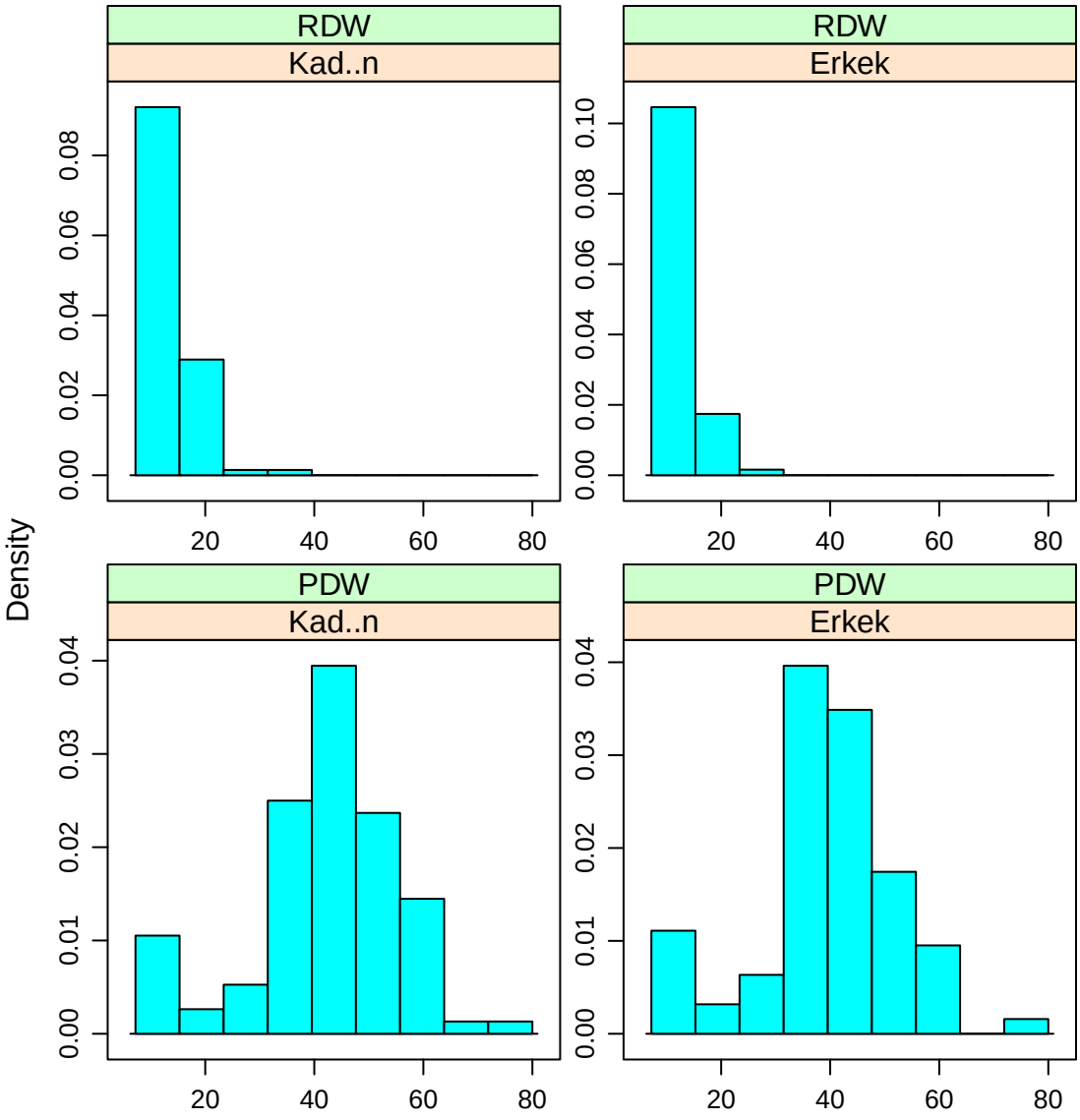
### a) Histogram

```
library(lattice)

histogram(mpv[["yas"]], xlab = "Yaş", col = "gray75", type = "density")
```



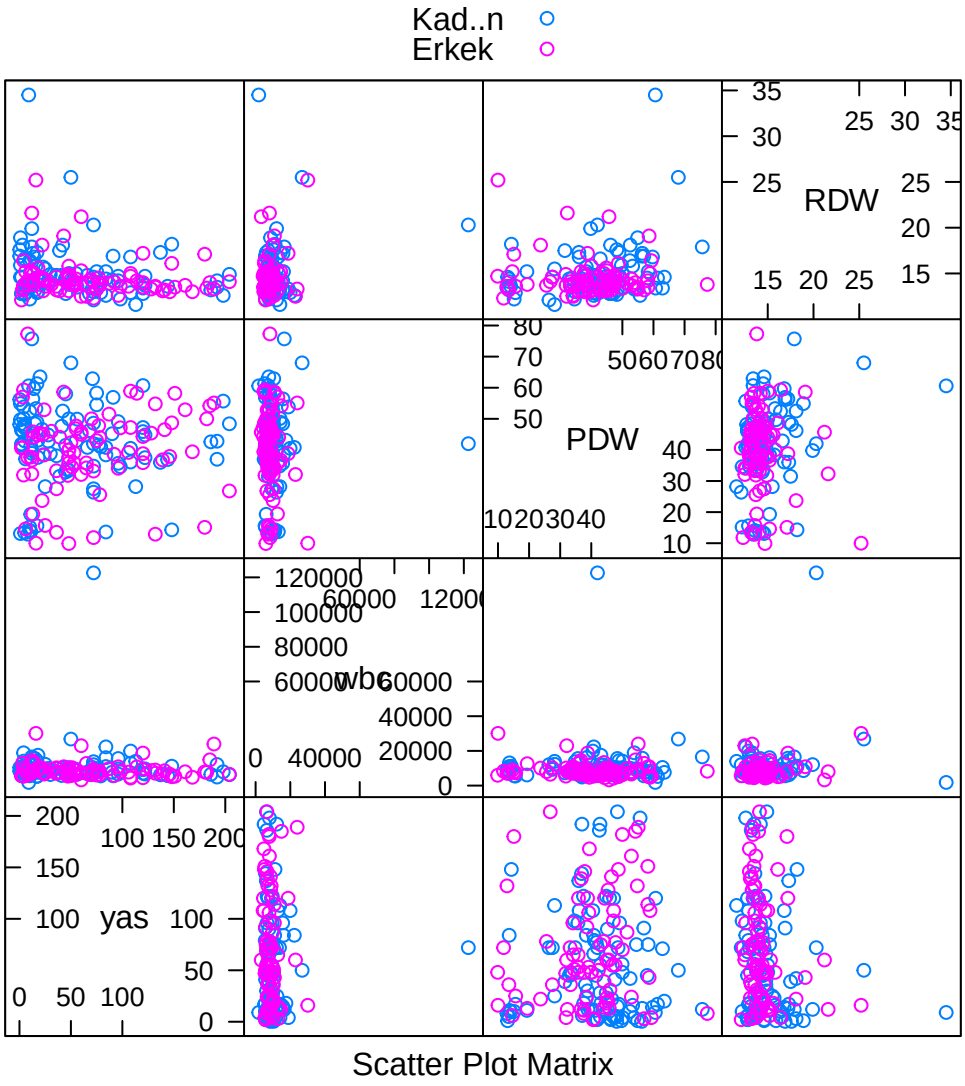

```r
# Formula yapısı kullanmak:
# formula: y ~ x | group
histogram(~ PDW + RDW | cinsiyet, data = mpv, type = "density", xlab = "",
          scales = list(relation = "free"))
```



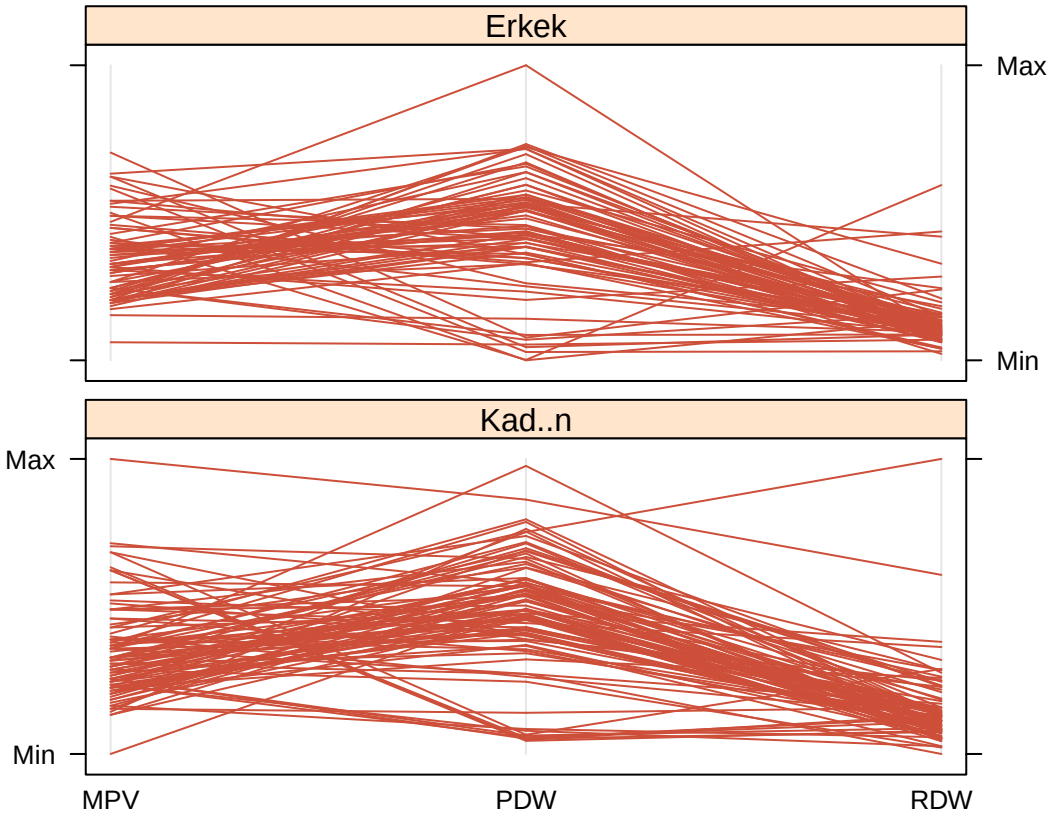

**b) Scatterplot**

```
splom(~ mpv[ ,c("yas", "wbc", "PDW", "RDW")], groups = cinsiyet, data = mpv,
      auto.key = TRUE)
```




**c) Parallel Plot**

```
library(lattice)
p2 <- parallelplot(~ mpv[ ,c("MPV", "PDW", "RDW")] | cinsiyet, data = mpv,
                   horizontal.axis = FALSE, layout = c(1,2,1), col = "tomato3")
plot(p2)
```

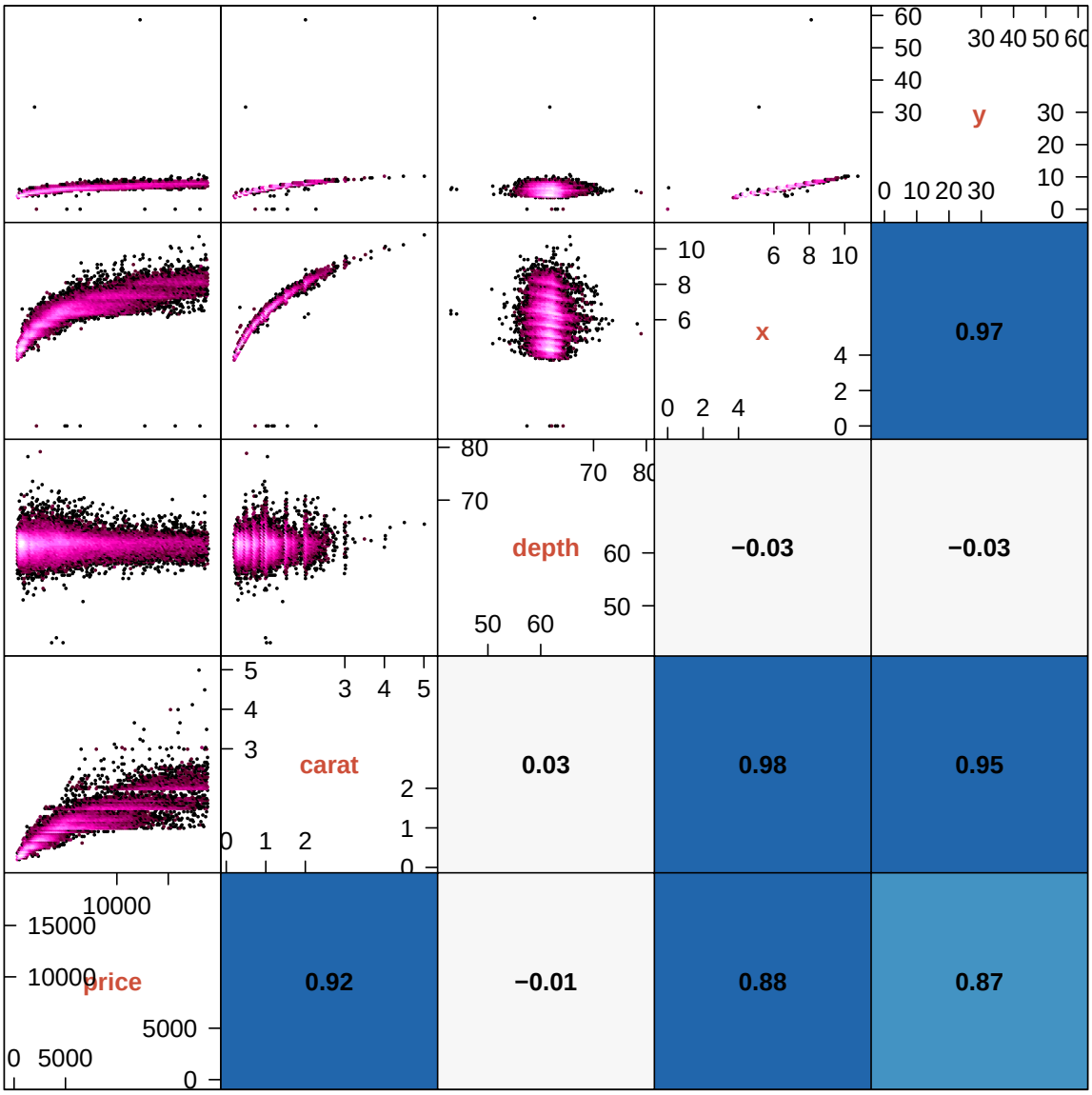



**d) Daha fazlası ...**

```
library(lattice)
library(latticeExtra)
library(hexbin)
library(RColorBrewer)

## Kaynak: http://wresch.github.io/2012/11/30/modified-splom.html
compRepl <- function(df){
# function to compare replicates (each variable of df)
# as hexbin plot matrix
# Args:
#   df      data frame
# Results:
#   lattice plot
  ct <- custom.theme(
            symbol = c("black", brewer.pal(n = 8, name = "Dark2")),
            fill = brewer.pal(n = 12, name = "Set3"),
            region = brewer.pal(n = 11, name = "Spectral"),
            reference = "#e8e8e8",
            bg = "transparent", fg = "black",
            lwd=2, pch=16)
  ct$axis.text$cex = 1
  ct$par.xlab.text$cex = 1
```

```
  ct$par.ylab.text$cex = 1

  splom(~df,
        #pscales = 0, #don't show axes,
        par.settings = ct,
        upper.panel = panel.hexbinplot,  # use hexbinplot
        xbins = 100,                     # number of bins
        trans = log10, inv=function(x) 10^x, # density color scale transformation
        colramp = magent,                    # with this color scheme
        #colramp = LinGray,

        # show correlation coefficient in lower panel
        lower.panel = function(x,  y, ...) {
          panel.fill(col = brewer.pal(9, "RdBu")[round(cor(x, y) *  4 + 5)])
          panel.text(sum(range(x))/2, sum(range(y))/2, round(cor(x, y), 2), font = 2)
        },
        varname.cex = 1, #smaller name in diagonal
        varname.font = 2,
        varname.col = "tomato3"
  )
}

compRepl(diamonds[ ,c("price", "carat", "depth", "x", "y")])
```




Scatter Plot Matrix

- Lattice kütüphanesi ile grafiklerin bir çok özelliğine müdahele edilebilir ve kişiselleştirilmiş grafik modülleri oluşturulabilir. Ancak, **lattice** kütüphanesine tam anlamıyla hakim olmak biraz zaman alacaktır.
- Lattice kütüphanesine benzer şekilde bir çok özelliğine müdahele edilebilen bir diğer kütüphane ise **ggplot2** kütüphanesidir.
- **ggplot2** kütüphanesi **lattice** kütüphanesine kıyasla daha kolay öğrenilebilecek bir grafik kütüphanesidir.

## ggplot2 kütüphanesi

- İleri düzey ve kullanışlı bir grafik kütüphanesidir.
- Grafik ile ilgili her şey bir akış içerisinde parçalar halinde yazılır.
- Kodlamanın parçalara ayrılması sebebiyle anlaşılması kolaydır.
- Dokümantasyonu geniş bir kütüphanedir.
  - Kullanım Kılavuzu: https://ggplot2.tidyverse.org/index.html

  - Kitap: https://ggplot2-book.org/
  - Cookbook: https://r-graphics.org/index.html
  - Cheatsheets: https://www.rstudio.com/resources/cheatsheets/
- Grafik derleyicisi `ggplot(...)` veya `qplot(...)` fonksiyonu ile aktif hale getirilir.
- Grafiğe eklenecek her özellik `ggplot(...)` fonksiyonunu takip edecek şekilde yazılır ve `+` ile katmanlar olarak eklenir.

`ggplot(...)` fonksiyonu iki temel parametreye ihtiyaç duyar:

- `data`: grafik çiziminde dikkate alınacak olan veriler
- `mapping`: grafik çizimi için gerekli olan bilgiler. `aes(...)` fonksiyonu içerisinde belirlenir.

**ggplot2** kütüphanesinde grafikler katmanlar halinde birbiri ardına eklenerek çizilir. Her eklenen yeni katman `+` ile bir önceki katmana eklenerek grafiğin son hali elde edilir.

```
ggplot(data, mapping = aes(...), ...) +
  katman1 +
  katman2 +
  ...

ggplot()
```

Katmanlara ek olarak grafik ile ilgili bütün düzenlemeler ve grafik özellikleri benzer şekilde `ggplot(...)` fonksiyonuna `+` kullanılarak eklenir:

```
ggplot(data = mpv, mapping = aes(x = yas, y = notrofil)) +
  geom_point() +     ## Scatter Plot
  xlab(...) +    ## x-axis label
  geom_smooth(...) +    ## Smooth curve
  theme(...)       ## Tema özellikleri
```

**ggplot2** kütüphanesinde kullanılan temel bileşenler:

| Fonksiyon | İşlev |
|---|---|
| `ggplot(...)`, `qplot(...)` | Grafik derleyicisini aktif eder. `qplot(...)` fonksiyonu `ggplot(...)` fonksiyonun basite indirgenmiş bir alternatifidir. |
| `geom_*(...)` | Geometrik nesneleri belirlemek için kullanılır. Çizilecek olan grafiğin türüne göre `*` bölümünde geometrik nesnenin adı kullanılır. `*` yerine grafik türüne göre `line`, `point`, `bar`, `qq`, ... gibi gösterimler kullanılır. **ggplot2**'de kullanılabilecek geometrik nesnelerin listesine https://ggplot2.tidyverse.org/index.html üzerinden erişilebilir. |
| `stat_*` | Grafik çizilmeden önce veri üzerinde dönüşümler ve düzenlemeler yapılmak isteniyor ise `stat_*` fonksiyonundan yararlanılır. |
| `theme(...)` | Grafik tema özelliklerini değiştirir. |
| `scale_*`, `coord_*`, `position_*`, `aes(...)` | Grafik eksenleri, paneller ve yerleşimleri gibi diğer özellikleri değiştirmek için kullanılır. |

**ggplot2**'de kullanılabilen diğer fonksiyonlar için https://ggplot2.tidyverse.org/index.html adresinden yararlanabilirsiniz.

**`qplot(...)` fonksiyonu:**

```
qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE,
  geom = "auto", xlim = c(NA, NA), ylim = c(NA, NA), log = "",
  main = NULL, xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)), asp = NA, stat = NULL, position = NULL)
```

```
quickplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE,
  geom = "auto", xlim = c(NA, NA), ylim = c(NA, NA), log = "",
  main = NULL, xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)), asp = NA, stat = NULL, position = NULL)
```

**Scatter plot**:

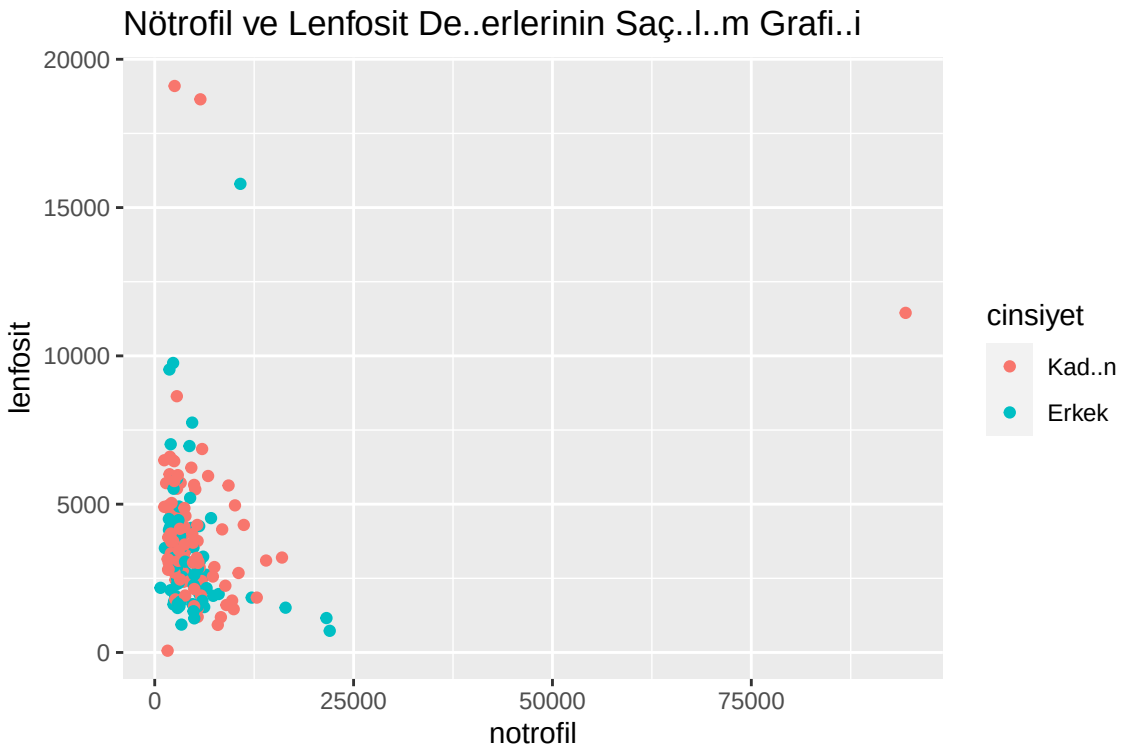
```
# Scatter plot
# Grouped by Species (different colors for each)
# Loess curve without confidence interval around.
qplot(x = carat, y = price, geom = "point", data = diamonds, colour = color,
      main = "Scatter plot of diamond prices and carats") +
  geom_smooth(method = "loess", se = FALSE, lty = 1, lwd = 1)  ## add regression line.

# "geom" içerisinde birden fazla geometrik obje tanımlanabilir.
# Bu durumda bazı objeleri "+" kullanarak eklemeye gerek kalmaz.
qplot(x = carat, y = price, geom = c("point", "smooth"), data = diamonds,
      colour = color, main = "Scatter plot of diamond prices and carats")

# İkiden fazla "geom" tanımlandığı durumda objelerden herhangi birisine ait
# parametre qplot(...) içerisinde tanımlanamaz. Örneğin; "smooth" objesine
# ait "se" değeri FALSE yapılır ise fonksiyon uyarı verecektir. Burada haricen
# girilen parametreler 'geom' içerisinde tanımlanan bütün katmanlara aktarılır.
# Herhangi bir katmanda bu parametre kullanılmıyor ise o katman için uyarı
# verir.
qplot(x = carat, y = price, geom = c("point", "smooth"), data = diamonds,
      colour = color, main = "Scatter plot of diamond prices and carats",
      se = TRUE)   # 'Ignoring unknown parameters: se'
```

`qplot(...)` için dikkat edilmesi gerekenler:

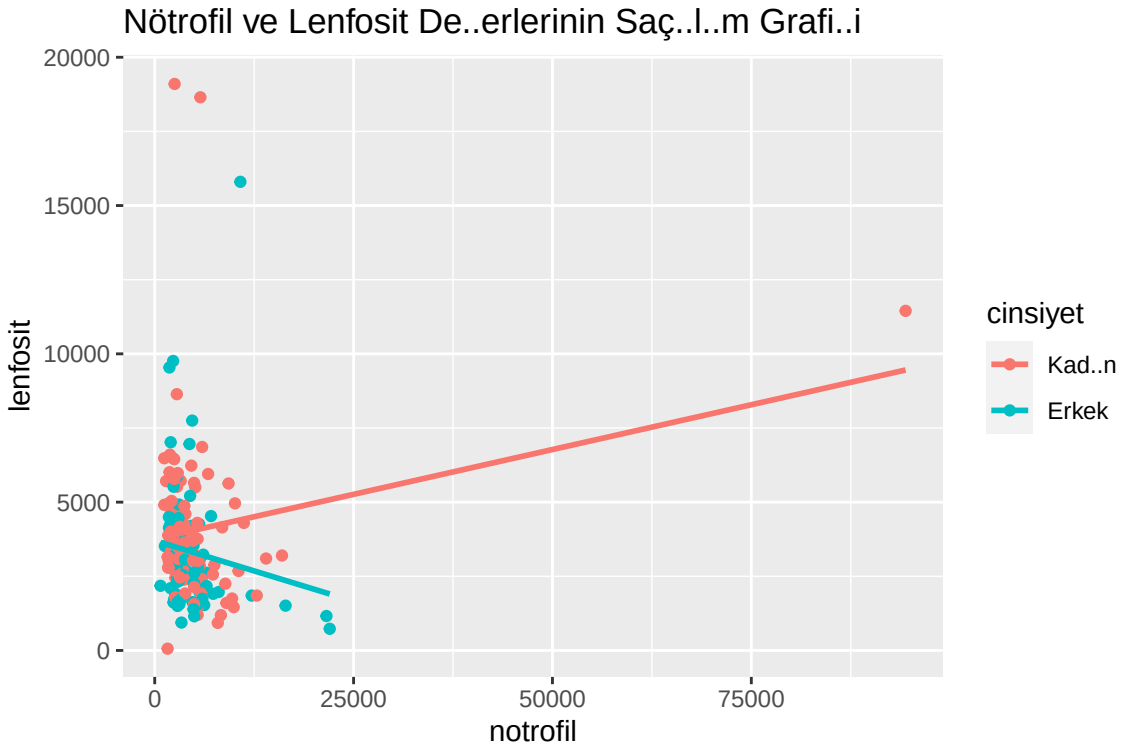
- `qplot()` fonksiyonu ile yalnızca bir `geom_` objesi tanımlanır ise bu objeye ait parametreler `qplot(...)` içerisinde girilebilir.
- `geom_*` ile tanımlanan objeler aksi belirtilmediği sürece varsayılan parametre değerleri ile çizilir.
- Birden fazla `geom_` objesi tanımlanır ise bu katmanlara ait parametreler `qplot(...)` fonksiyonunu takip edecek şekilde `+` ile eklenmelidir.

```
myplot <- qplot(x = notrofil, y = lenfosit, geom = c("point"),
                main = "Nötrofil ve Lenfosit Değerlerinin Saçılım Grafiği", data = mpv, colour = cinsiy

print(myplot)
```

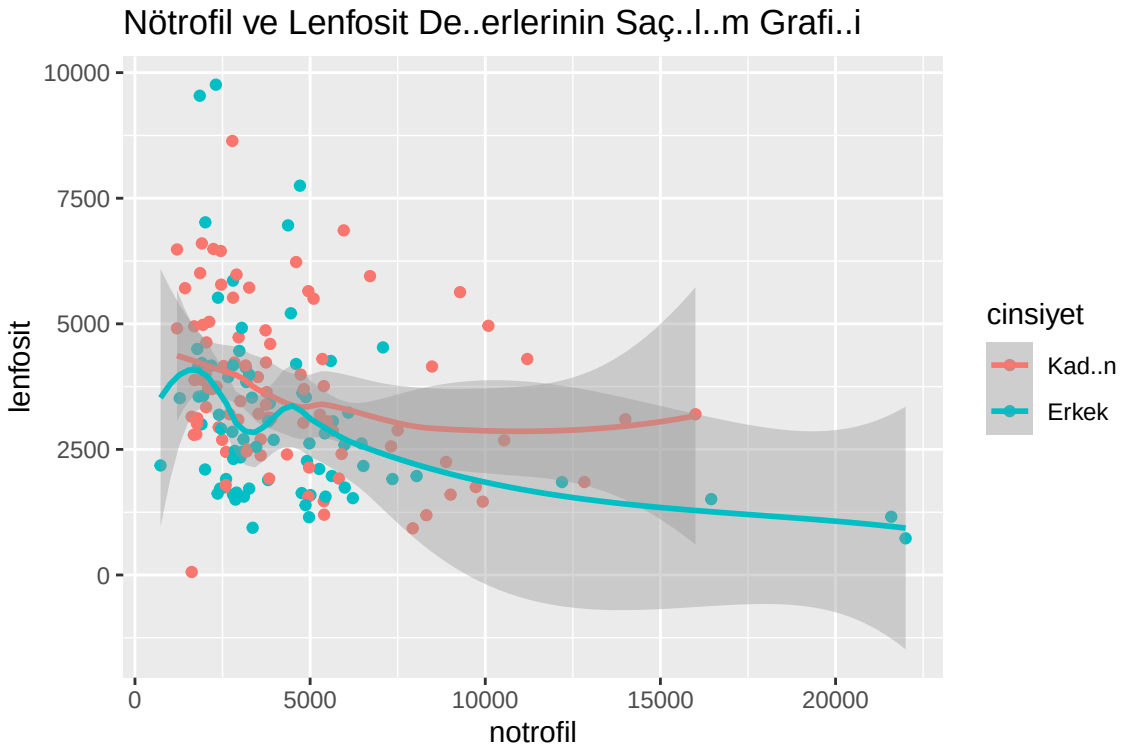



```
# Güven aralığını kaldıralım.
# Tema olarak siyah - beyaz tema kullanalım.
myplot <- myplot + geom_smooth(se = FALSE, method = "lm")
print(myplot)
```



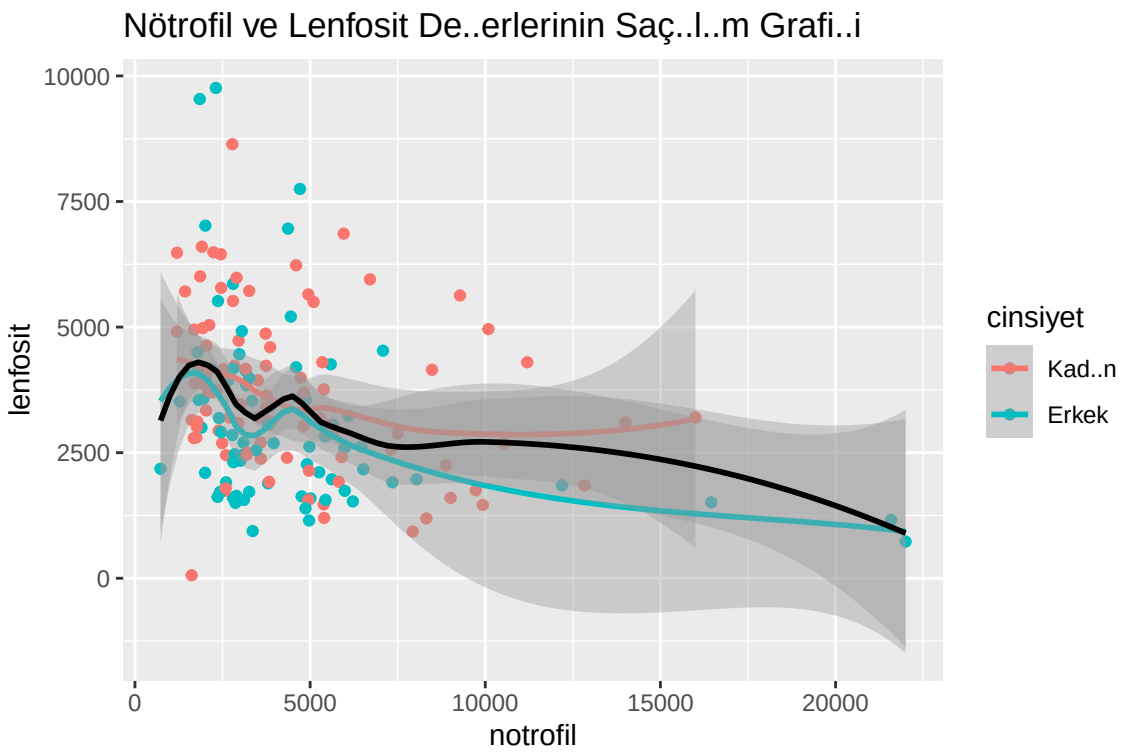
**Dikkat:** `qplot(...)` ile tanımlanan `geom` katmanı fonksiyon dışarısında tekrar `+ geom_*(...)` olarak eklenir ise bu katman yeni bir grafik olarak mevcut grafiğin üzerine eklenir. Bu durumda aynı katman iki defa çizilmiş olacağı için istenilen sonuçlar elde edilemeyebilir.

```
library(dplyr)
library(ggplot2)

## notrofil değerleri 25bin altında, lenfosit değerleri 10bin altında olan
## gözlemler seçilir.
mpv2 <- filter(.data = mpv, notrofil < 25000 & lenfosit < 10000)


myplot <- qplot(x = notrofil, y = lenfosit, geom = c("point"),
                main = "Nötrofil ve Lenfosit Değerlerinin Saçılım Grafiği", data = mpv2, colour = cinsi

myplot <- myplot + geom_smooth(method = "loess", se = TRUE)
print(myplot)
```




**Bütün gözlemleri dikkate alan genel "loess" eğrisini güven aralıkları ile birlikte mevcut grafiğe ekleyelim:**

```
# geom_* objeleri kendi içinde qplot(..)'dan bağımsız olarak oluşturulabilir.
# Oluşturulan nesne bir katman olarak kullanılacağı için qplot() fonksiyonuna '+' ile eklenebilir.
overallLoessCurve <- geom_smooth(data = mpv2, mapping = aes(x = notrofil, y = lenfosit),
                                 se = TRUE, col = "black", span = 0.5)
myplot <- myplot +
  overallLoessCurve

print(myplot)
```




**`ggplot(...)` fonksiyonu:**

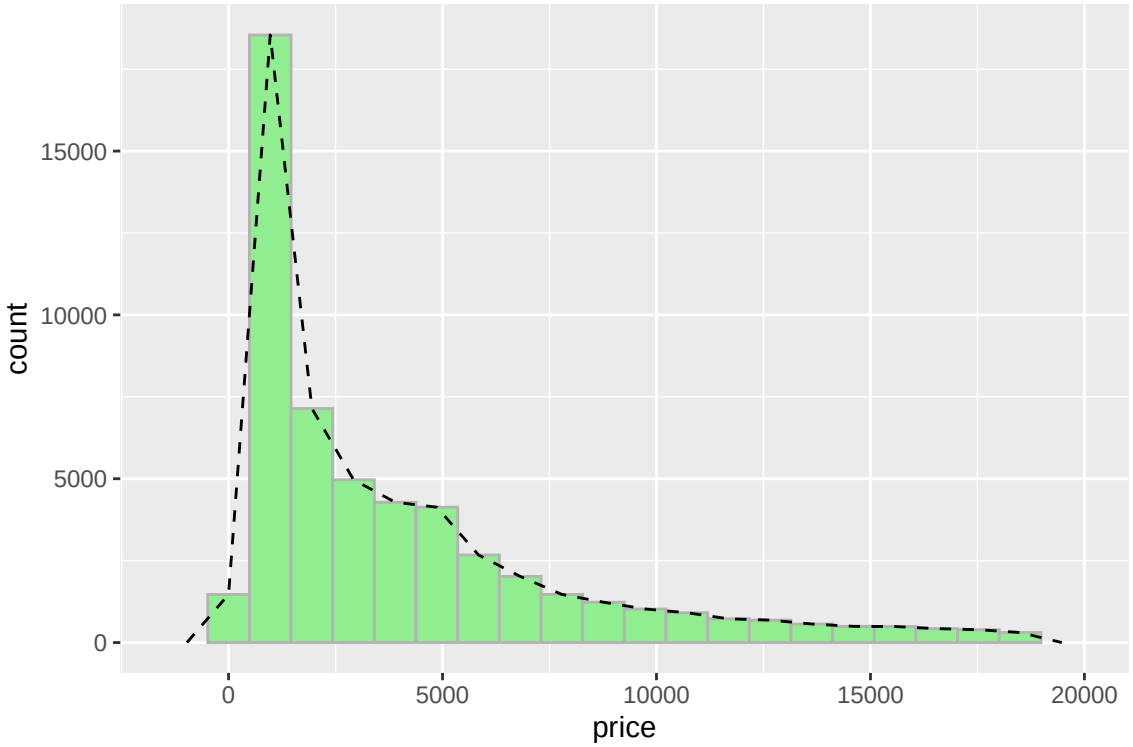
- `ggplot(...)` fonksiyonu grafik çizimi yapılmak üzere grafik derleyicisini çalıştırır.
- Grafik çizim alanı üzerine eklemek istediğiniz bütün geometrik nesneler ve nesne özellikleri katmanlar halinde eklenir.
- `qplot(...)` fonksiyonuna göre daha basit kullanıma sahiptir. Ancak, her katman için ayrıca işlem yapılmalıdır.
- `ggplot(...)` içerisinde belirlenen değişkenler ve ver seti aksi belirtilmediği sürece bütün geometrik nesnelerde (`geom_*(...)`) aynı şekilde kullanılır.

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

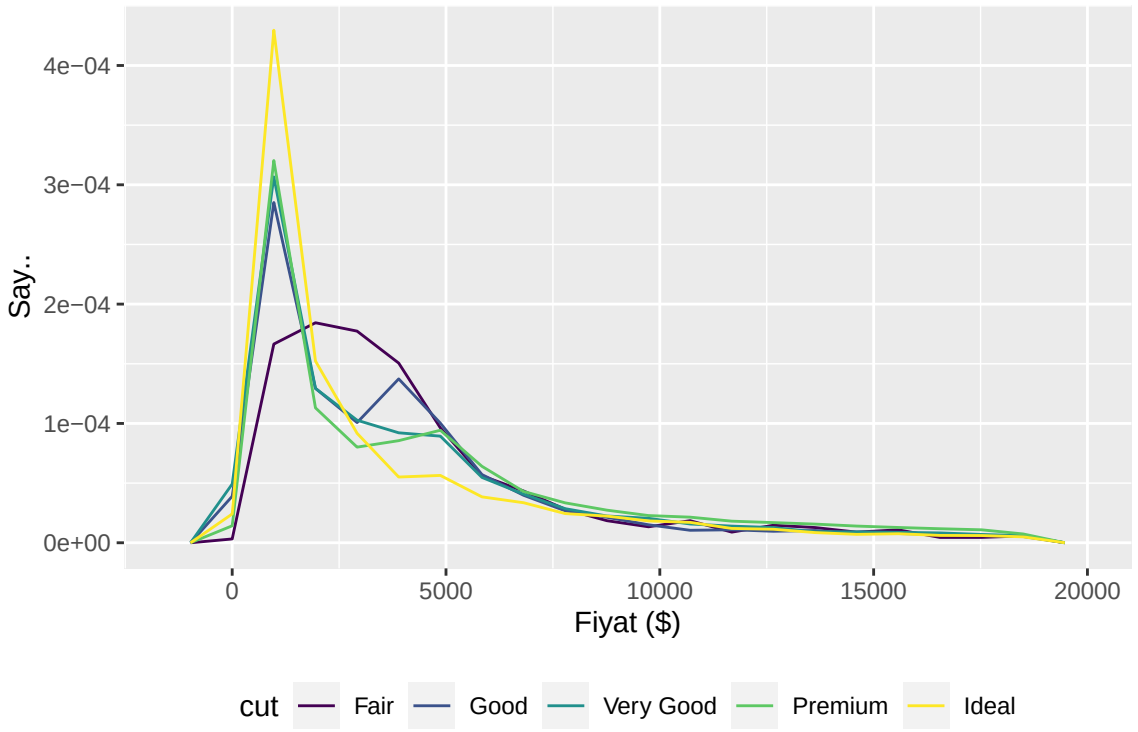
**Histogram, Frekans Poligonu** (`geom_histogram(...)`, `geom_freqpoly(...)`):

```
data(diamonds)

## use "..count.." or "..density" to switch between counts and densities on y-axis
myPlot <- ggplot(data = diamonds, mapping = aes(x = price, y = ..count..)) +
  geom_histogram(bins = 20, fill = "lightgreen", color = "gray70") +
  geom_freqpoly(bins = 20, color = "black", lty = 2)

print(myPlot)
```

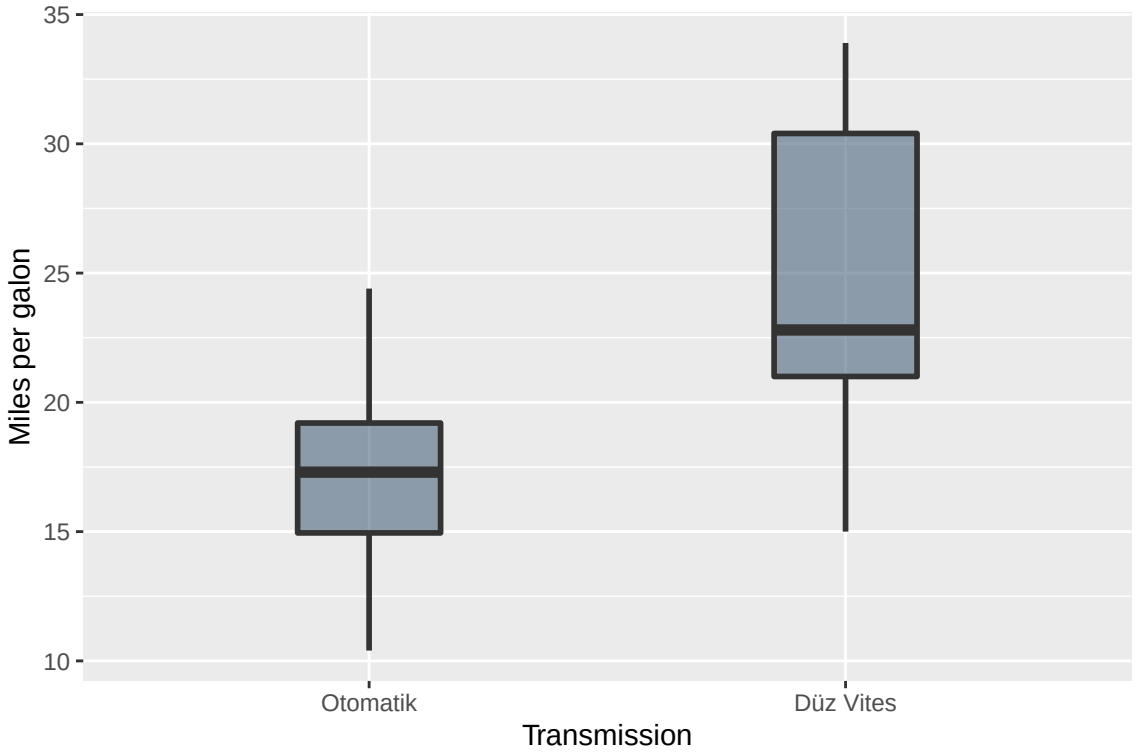



```
## y-ekseninde density olacak şekilde "gruplandırılmış frekans poligonu"
myPlot <- ggplot(data = diamonds, mapping = aes(x = price, y = ..density.., colour = cut)) +
  geom_freqpoly(bins = 20) +
  theme(legend.position = "bottom") +     ## legends at the bottom
  ylab("Sayı") +
  xlab("Fiyat ($)")

print(myPlot)
```




**Box Plot** (`geom_boxplot(...)`):

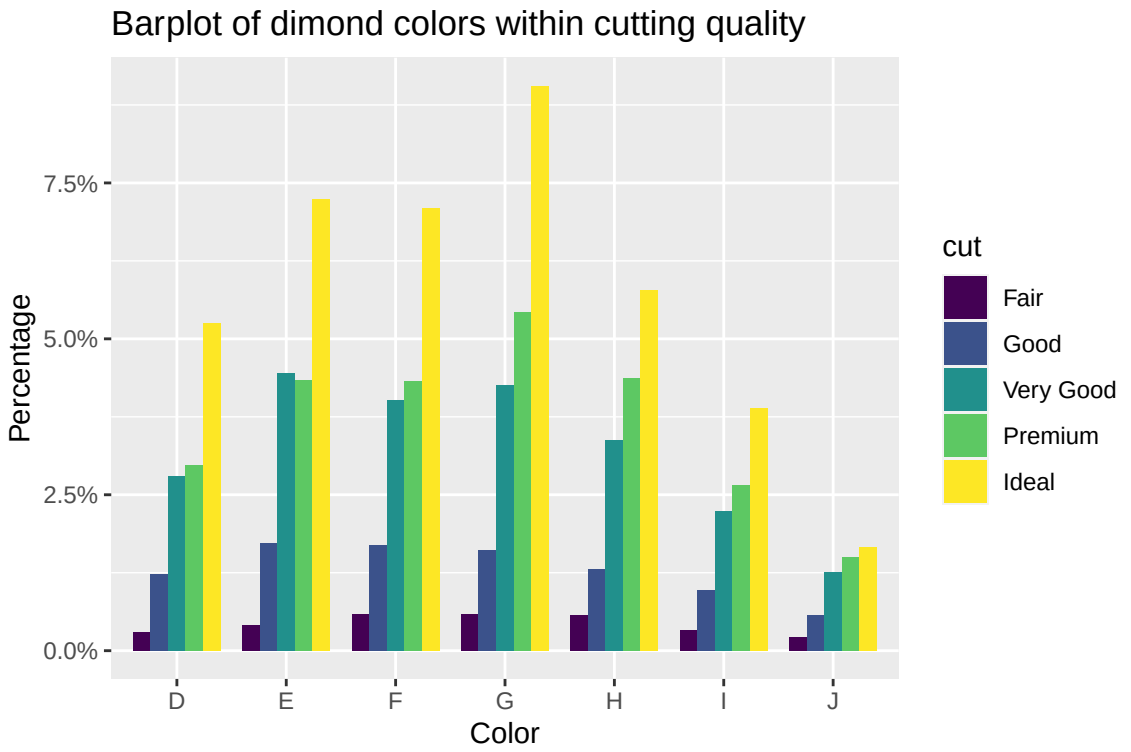
```
library(dplyr)
data(mtcars)

## Araçların vites bilgisini gösteren "am" değişkenini "Otomatik", "Düz Vites" isimleri ile
## yeniden kodlayıp vites türüne göre araçların yakıt tüketimlerinin kutu çizgi grafiğini oluşturalım.

mtcars <- mtcars %>%
  mutate(Transmission = recode_factor(am, '0' = "Otomatik", '1' = "Düz Vites"))

ggplot(data = mtcars, mapping = aes(x = Transmission, y = mpg)) +
  geom_boxplot(fill = rgb(0.3, 0.4, 0.5, 0.6), width = 0.3, lwd = 1) +
  ylab("Miles per galon")
```




```
  # geom_point(colour = "gray40")
  # geom_jitter(width = 0.25, color = "tomato3")   # jittering
```

**Bar Graph** (`geom_bar(...)`):

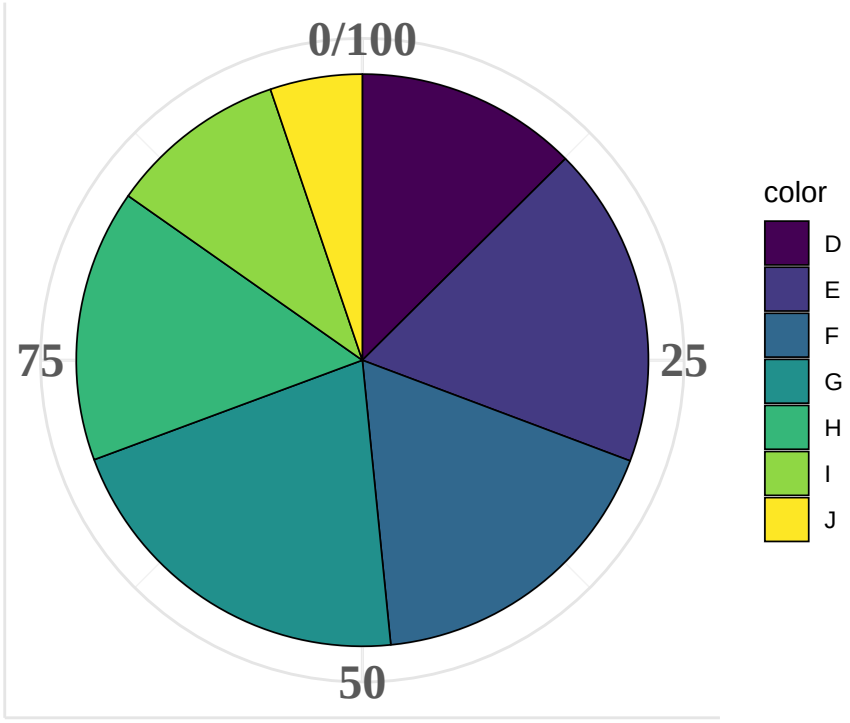
```
data(diamonds)

# Stacked
ggplot(diamonds, aes(x = color, fill = cut)) +
  geom_bar(position = "stack")

# Unstacked
ggplot(diamonds, aes(x = color, fill = cut)) +
  geom_bar(position = "dodge")

# Stacked, percentage on the y-axis
# Yüzdeler genel toplama bölünerek elde edilir.
ggplot(diamonds, aes(x = color, y = 100 * (..count..) / sum(..count..), fill = cut)) +
  geom_bar(position = "stack") +
  ylab("Percentage (%)")

# Stacked, percentage on the y-axis
# Her x kategorisi kendi içinde değerlendirilir.
ggplot(diamonds, aes(x = color, y = 100 * (..count..) / sum(..count..), fill = cut)) +
  geom_bar(position = "fill") +
  ylab("Percentage") +
  xlab("Color") +
  scale_y_continuous(labels = scales:::percent_format())
```




**Pie Chart** (`geom_bar(...) + coord_polar(...)`):

- Pie Chart grafiği çizmek için `geom_rect(...)` fonksiyonu kullanılır.
- Bar Graph'den elde edilen bilgiler `coord_polar(...)` fonksiyonu ile dairesel eksene dönüştürülür.

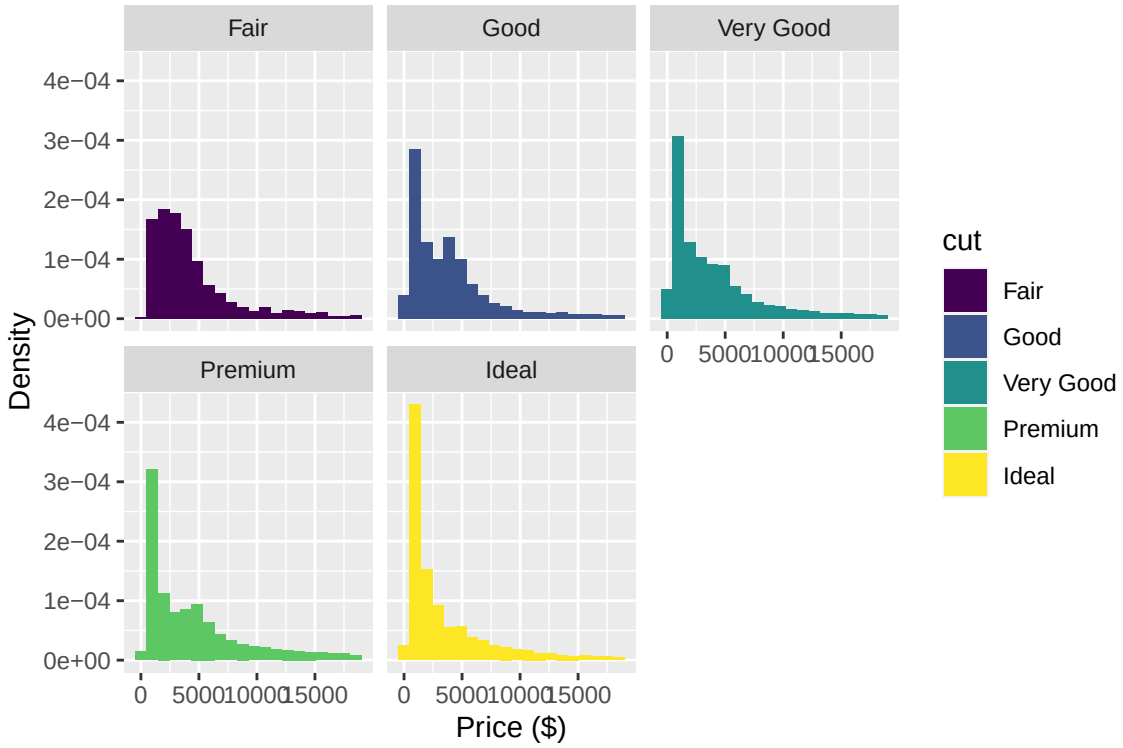
```
library(dplyr)
data(diamonds)

diamonds_grp <- group_by(diamonds, color)    # data is grouped by "color"
pieData <- summarise(diamonds_grp, Count = n())
pieData <- mutate(pieData, Percentage = Count / sum(Count),
                  end = 100*cumsum(Percentage),
                  start = c(0, end[-length(end)]))    ## Start and End points are calculated for each s

ggplot(pieData, aes(fill = color, ymax = start, ymin = end, xmax = 6, xmin = 0)) +
  theme(panel.background = element_rect(fill = "white", linetype = "solid", colour = "white"),
                           panel.grid.minor = element_line(color = "gray95"),
                           panel.grid.major = element_line(color = "gray90"),
                           axis.ticks=element_blank(),
                           axis.title.x = element_text(),
                           axis.text.x = element_text(family = "Times", size = 18,
                                                      colour = "gray35", face = "bold"),
                           axis.title.y = element_text(),
                           axis.text.y = element_text(colour = "white"),
                           axis.line = element_line(colour = "gray90")) +
  geom_rect(stat = "identity", color = "black", lwd = 0.3) +
  xlim(c(0, 6)) +
  coord_polar(theta = "y")
```



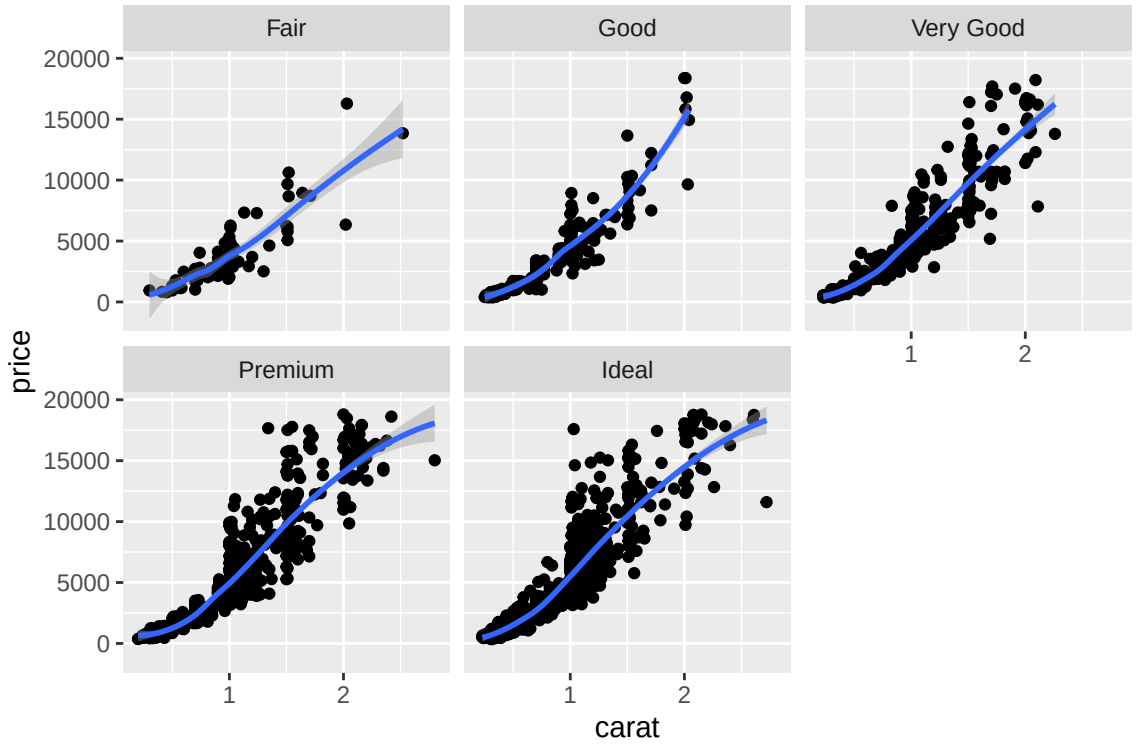

**Çoklu grafiklerin oluşturulması (`facet_wrap(...)`, `facet_grid(...)`)**

- `facet_wrap(...)` fonksiyonu ile oluşturulan panellerin satır ve sütun sayıları belirlenebilir.
- `facet_grid(...)` fonksiyonu satır ve sütun yerleşimlerini otomatik yapmaktadır.
- `facet_*(...)` fonksiyonları temel grafiklerde kullanılan `par(mfrow = c(...))` ile aynı görevi görmez. Farklı türden grafiklerin ayrı panellerde gösterilebilmesi için **grid** kütüphanesinde yer alan bazı fonksiyonlardan yararlanılır.

```
ggplot(diamonds, aes(x = price, y = ..density.., fill = cut)) +
  geom_histogram(bins = 20) +
  facet_wrap(~ cut, nrow = 2) +
  xlab("Price ($)") +
  ylab("Density")
```

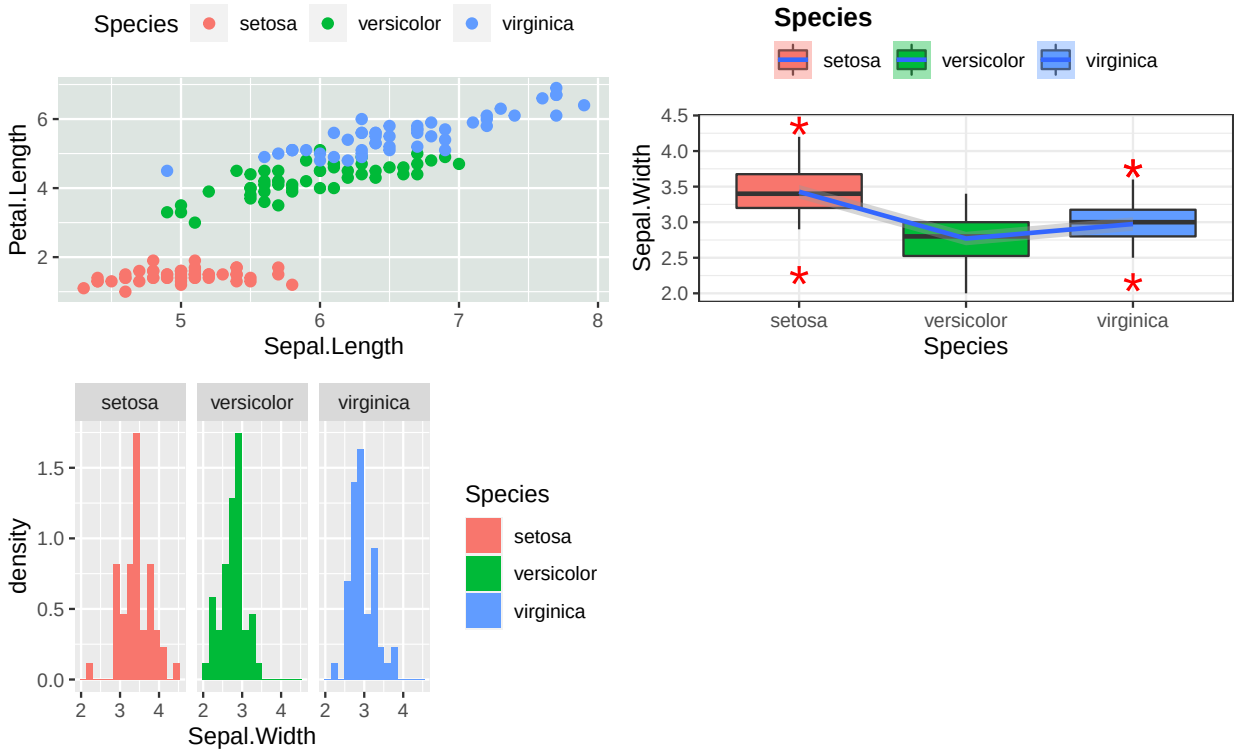



```r
n <- nrow(diamonds)
sampleData <- diamonds[sample(n, 0.05*n, FALSE), ]

# Diamonds verisinin %5'i rasgele olarak seçilmiştir.
# Grafik çizimini hızlandırmak için.
ggplot(sampleData, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth(method = "loess") +
  facet_wrap(~ cut, nrow = 2)
```




```
library(gridExtra)
library(grid)
data(iris)

plotScatter <- ggplot2:::ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, colour = Species)) +
  theme(panel.background = element_rect(fill = rgb(0.1, 0.3, 0.2, 0.15))) +
  geom_point(size = 2) +
  theme(legend.position = "top")

bwTheme <- theme_bw()   ## Black and White theme.
bwTheme$legend.position <- "top"

plotBox <- ggplot(iris, aes(x = Species, y = Sepal.Width, fill = Species)) +
  bwTheme +
  geom_boxplot(outlier.colour = "red", outlier.shape = "*", outlier.size = 8) +
  geom_smooth(aes(group = 1), method = "loess") +
  scale_fill_discrete(guide = guide_legend(title.theme = element_text(face = "bold", angle = 0),
                                           title.position = "top"))

plotHistogram <- ggplot(iris, aes(x = Sepal.Width, y = ..density.., fill = Species)) +
  geom_histogram(bins = 15) +
  facet_grid(~ Species)

grid.arrange(plotScatter, plotBox, plotHistogram, nrow = 2, ncol = 2)
```

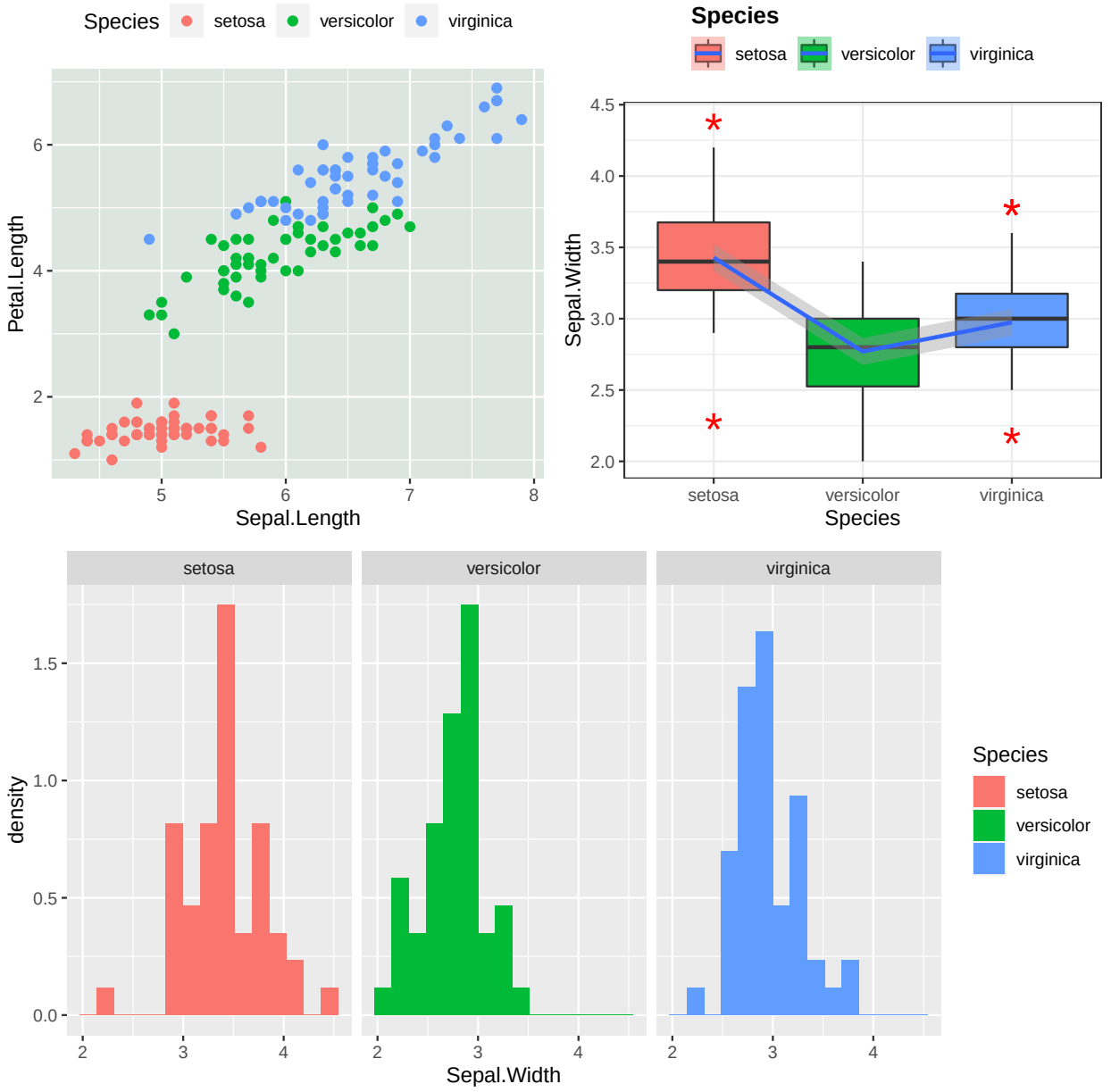



Grafiklerin çizim alanındaki yerleşimlerini ve grafiğin çizim alanında kapladığı bölgeyi keyfi olarak belirlemek mümkündür. Bunun için aşağıda verilmiş olan "multiplot(...)" fonksiyonundan yararlanabilirsiniz.

```
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
# Source: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
#

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
```

```
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

```
# Histogram grafikleri 2. satırın tamamını kaplayacaktır.
multiplot(plotScatter, plotBox, plotHistogram, layout = matrix(c(1,2,3,3), ncol = 2, byrow = TRUE))
```




```
# Grafik alanının kroki görünümü için aşağıdaki kodu çalıştırabilirsiniz.
plotarea <- layout(matrix(c(1,2,3,3), ncol = 2 byrow = TRUE))
layout.show(plotarea)
```

**theme(...): ggplot grafiklerinin kişiselleştirilmesi**

```
plotData <- mpv %>%
  mutate(NLR = 100 * notrofil / lenfosit)

## Scatter plot
ggplot(plotData, aes(x = RDW, y = NLR, colour = yas)) +
  geom_point(size = 3) +
  theme_bw() +
  theme(legend.title = element_text(face = "bold"), legend.margin = margin(0,10,0,20)) +
```

```
  guides(colour = guide_colourbar(title = "Yaş"))

# Önceden tanımlı temalar
theme_bw()
theme_classic()
...
```

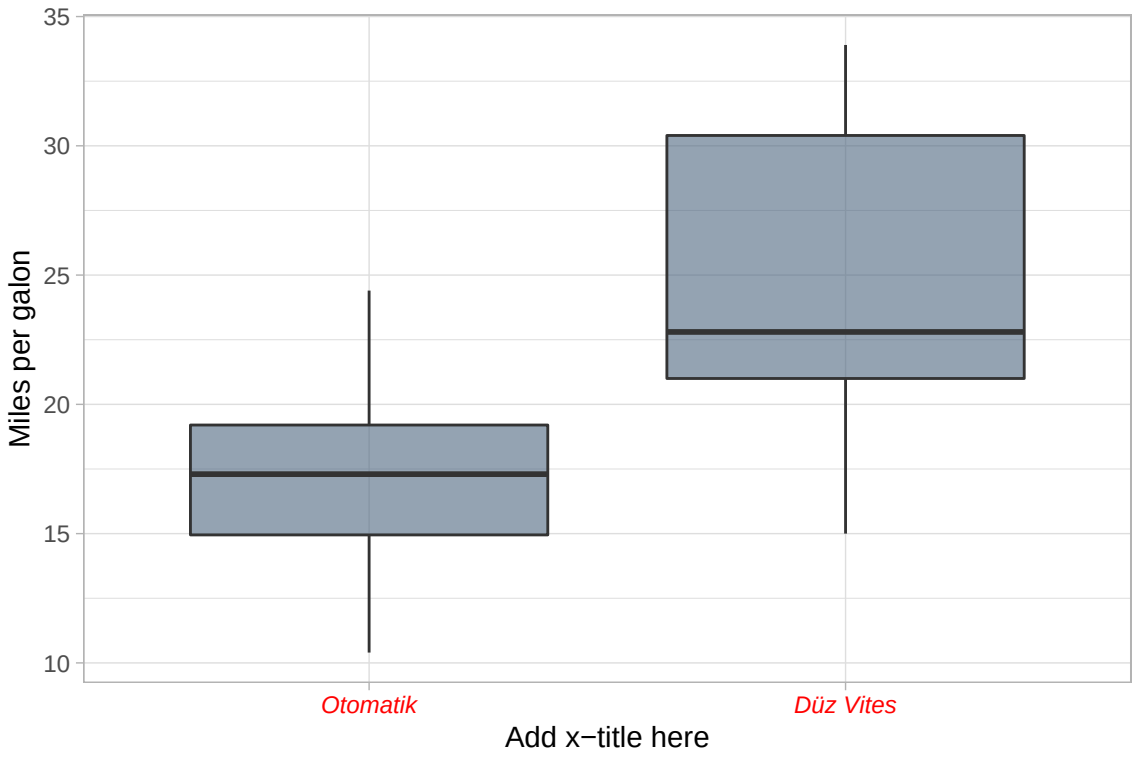
**ggplot_build(...): grafik bilgilerinin dışarıya aktarılması**

- **ggplot2** kütüphanesinde çizilen grafiklere ait bütün bilgiler `ggplot_build(...)` fonksiyonu yardımı ile dışarıya aktarılabilir.
- Grafiğe ait bu bilgiler ilerleyen aşamalarda interaktif grafikler çizilirken kullanılabilir.
- Grafik bilgilerinden bir veya bir kaçı değiştirilerek grafiğin yeniden düzenlenmiş hali ile çizilmesi sağlanabilir.

```
mtcars <- mtcars %>%
  mutate(Transmission = dplyr:::recode_factor(am, '0' = "Otomatik", '1' = "Düz Vites"))

myPlot <- ggplot(data = mtcars, mapping = aes(x = Transmission, y = mpg)) +
  geom_boxplot(fill = rgb(0.3, 0.4, 0.5, 0.6)) +
  ylab("Miles per galon") +
  theme_light()

plotInfo <- ggplot_build(myPlot)
str(myPlot)    ## Grafiğe ait bütün bilgiler.
```

```
# Grafik x-eksen etiketini değiştirip yeniden çizelim:
plotInfo$plot$labels$x <- "Add x-title here"

plotInfo$plot$theme$axis.text.x$face <- "italic"
plotInfo$plot$theme$axis.text.x$colour <- "red"

## Değiştirilmiş halleri ile grafik tekrar çizilir ve grafiğe ait tüm bilgiler ekrana basılır.
print(plotInfo)
```




## Grafiklerin kaydedilmesi

- R ile çizilen grafiklerin kaydedilmesi için grafikler çizilmeden önce kaydedilmek istenen türe ait fonksiyonun aktif edilmesi gerekmektedir.
- Grafikler `png(...)`, `pdf(...)`, `png(...)`, `bmp(...)`, `svg(...)`, ... gibi fonksiyonlar kullanılarak kaydedilir.
- Grafik sürücüsünün kapatılması ve sonuçların kaydedilmesi için son satıra `dev.off()` komutu eklenir.

```
pdf(file = "myplot.pdf", width = 10, height = 8)   ## Çalışma dizinine "myplot.pdf" adı ile kaydeder.
plot(iris[ ,-5])
dev.off()
```

Kaydetme işlemlerinde kullanılacak opsiyonlar için ilgili fonksiyonun yardım dosyalarına bakınız.