

A laptop screen displays the RStudio software interface. The main window shows a plot titled 'Per Kooling' with a line graph and a bar chart. A large blue R logo is overlaid on the plot. The Persian text 'کارگاه مقدماتی تحلیل آماری با نرم افزار R' is centered over the screen. Below it, the text 'ارائه دهنده: منصور گلزاری | مهرماه ۱۴۰۴' is displayed. The laptop is on a desk with a small potted plant in the background.

کارگاه مقدماتی تحلیل آماری با نرم افزار R

ارائه دهنده: منصور گلزاری | مهرماه ۱۴۰۴

معرفی سه‌گانه کارگاه‌ها



کارگاه ۳: آمار استنباطی و
رگرسیون خطی
آزمون‌ها و رگرسیون خطی



کارگاه ۲: مصورسازی داده‌ها
گراف‌های اکتشافی



کارگاه ۱: مبانی R، از صفر تا
ساختار داده‌ها
نصب، syntax، متغیرها و
ساختارهای داده‌ای



R

یک زبان برنامه‌نویسی متن‌باز و رایگان
است که توسط متخصصان آمار
طراحی شده.

این زبان در پزشکی، زیست‌شناسی،
اقتصاد و داده‌کاوی کاربرد گسترده‌ای
دارد و با بیش از ۲ میلیون کاربر،
استاندارد طلایی تحلیل داده‌های
علمی است.

چرا R؟

جامعه فعال 

انجمن‌های پرسش و پاسخ، همایش‌های سالانه و تالارهای گفتگوی ۲۴ ساعته

۱۸۰۰۰+ بسته 

از تحلیل ژنوم تا یادگیری عمیق؛ هر روز بسته‌های جدیدی اضافه می‌شود

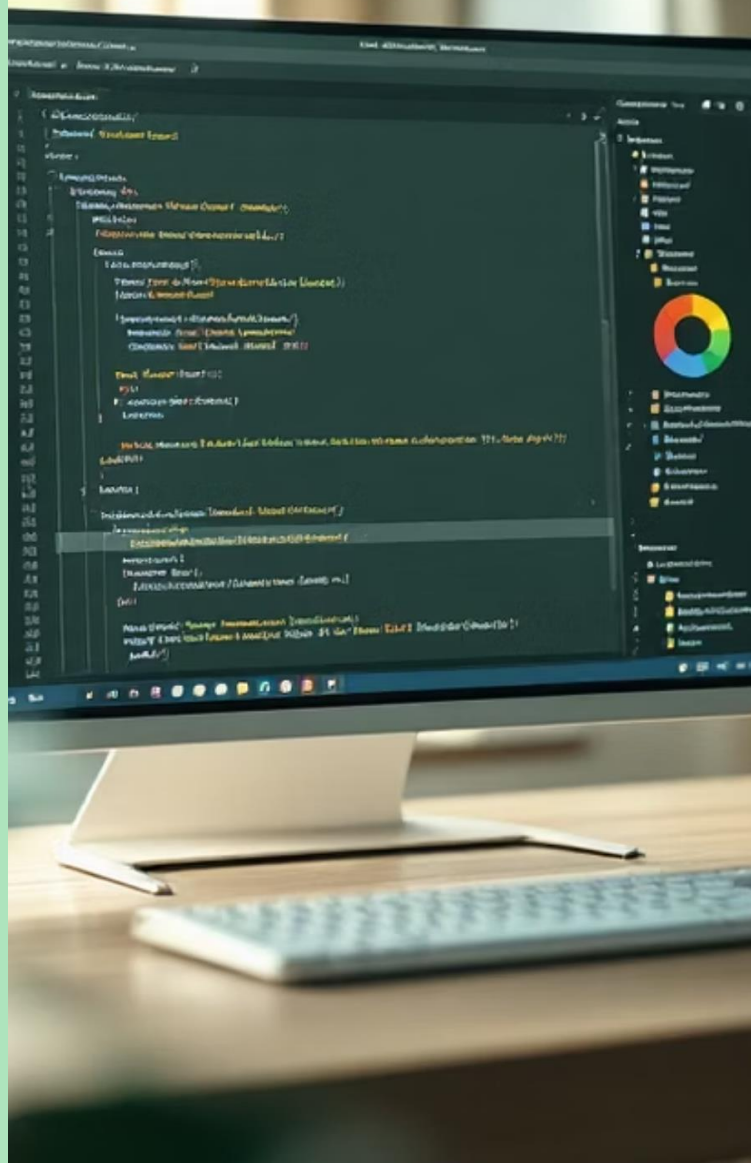
پردازش حجیم 

فایل‌های چندگیگابایتی را در ثانیه بارگذاری و تحلیل می‌کند

گراف حرفه‌ای 

نمودارهای قابل چاپ در مجلات ISI با یک خط کد





نصب R و RStudio

به آدرس cran.r-project.org مراجعه کنید.

نسخه متناسب با سیستم عامل خود (ویندوز/مک/لینوکس) را انتخاب کنید.

پس از نصب R، نرم افزار RStudio را از rstudio.com دریافت کنید.

نرم افزار RStudio محیط توسعه یکپارچه ای است که کار با R را لذت بخش می کند.

توجه: همیشه ابتدا R را نصب کنید سپس RStudio را اضافه کنید.





دوست برنامه‌نویس RStudio:

- تکمیل خودکار: با فشار Tab نام توابع و متغیرها کامل می‌شود 📝
- نمایش گرافیکی: نمودارها در پنجره اختصاصی با امکان زوم و ذخیره سازی 📊
- مدیریت پروژه: فایل‌ها، تاریخچه و بسته‌ها در یک نمای جامع 📁

مسیر کار خود را تعیین کنید

قبل از هر کاری باید بدانید در کدام پوشه هستید:

مسیر فعلی را نشان می‌دهد # `getwd()`

برای تغییر مسیر:

ویندوز # `setwd("C:/mydata")`

مک # `setwd("/Users/doc/data")`

نکته: از اسلش / یا دابل بک اسلش \\ استفاده کنید 



متغیرها: گنجینه داده‌ها

1

عملگر انتساب

برای ذخیره مقدار از $>$ یا $=$ استفاده می‌کنیم

2

مثال عملی

```
x <- 10  
y = 5  
z <- x + y # z برابر ۱۵
```

همه چیز یک شیء است

زبان R، شیء‌گراست؛ یعنی اعداد، متن‌ها، حتی توابع هم نوعی شیء محسوب می‌شوند.



انواع داده در R

نرم افزار R برای انواع مختلف داده‌ها، قالب‌های متنوعی ارائه می‌دهد.

1

Numeric 

اعداد اعشاری و صحیح برای محاسبات ریاضی



Character 

متن و رشته‌های حروفی برای اطلاعات کیفی



Logical  

مقادیر TRUE/FALSE برای تصمیم‌گیری‌های منطقی





بردارها: ستون‌های داده

بردار ساده‌ترین ساختار داده‌ای است که عناصر هم‌نوع را کنار هم قرار می‌دهد.

سه تابع اصلی برای ساخت بردار:

`c(1,3,5,7)`: ترکیب مقادیر `c()`

`seq(1,10,2)`: دنباله عددی `seq()`

`rep("A",5)`: تکرار مقدار `rep()`

توابع ریاضی در R

با استفاده از توابع داخلی R می‌توانید عملیات‌های ریاضی را به‌سادگی انجام دهید.

sqrt()

ریشه دوم عدد

log10()

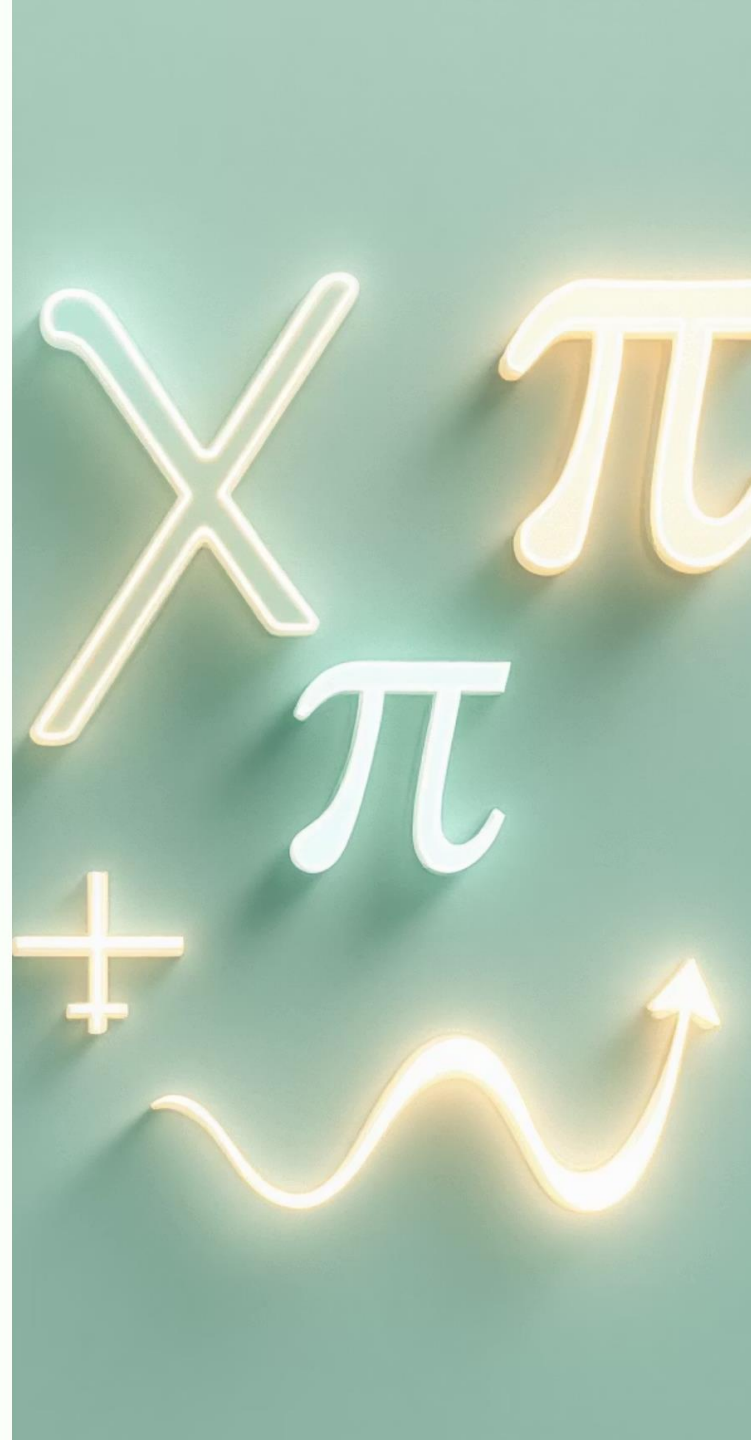
لگاریتم دهدهی

round()

گرد کردن اعداد

sin()

سینوس زاویه





توابع آماری کاربردی

برای خلاصه‌سازی و تحلیل داده‌ها، R توابع آماری قدرتمندی ارائه می‌دهد.

انحراف و واریانس

var(): واریانس نمونه

sd(): انحراف معیار

میانگین و میانه

mean(): میانگین حسابی

median(): میانه داده‌ها

عملگرهای منطقی و مقایسه‌ای

برای تصمیم‌گیری در کدنویسی، از عملگرهای منطقی استفاده می‌کنیم.



برابر

$a == b \rightarrow \text{TRUE}$



بزرگ‌تر

$a > b \rightarrow \text{FALSE}$



کوچک‌تر

$a < b \rightarrow \text{TRUE}$



منطقی OR

$a | b \rightarrow \text{TRUE}$



منطقی AND

$a \& b \rightarrow \text{TRUE}$

ماتریس‌ها: ساختار دوبعدی داده‌ها

ماتریس‌ها برای ذخیره داده‌های عددی در قالب جدولی بسیار مناسب‌اند.

ایجاد ماتریس

با تابع **matrix()** می‌توانید ماتریس بسازید.

مثال: `matrix(1:9, nrow=3)`

توابع کاربردی

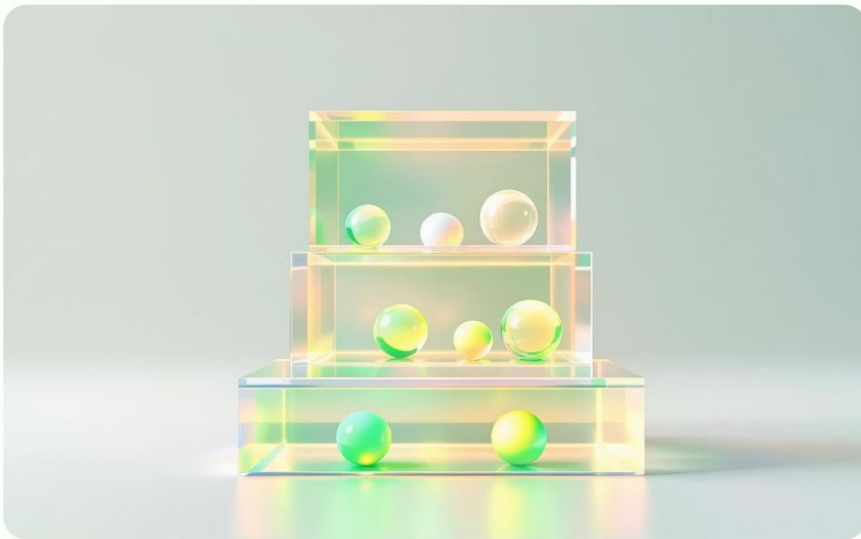
rowSums(): جمع هر سطر

colMeans(): میانگین هر ستون

۶	۳	۲	۱
۱۵	۶	۵	۴
۲۴	۹	۸	۷
۴۵	۱۸	۱۵	۱۲

لیست‌ها: مجموعه‌ای از اشیای مختلف

لیست‌ها امکان ترکیب انواع مختلف داده‌ها را در یک ساختار واحد می‌دهند.



ساختار تو در تو

لیست‌ها می‌توانند حاوی لیست‌های دیگر باشند و ساختارهای پیچیده ایجاد کنند.



ترکیب انواع داده

در لیست می‌توانید عدد، متن، بردار و حتی ماتریس را کنار هم قرار دهید.

مثال عملی: 

```
list(name="Ali", age=25, scores=c(15,18,20))
```

دیتافریم چیست؟

دیتافریم یکی از مهم‌ترین و پرکاربردترین ساختارهای داده در زبان R است که برای کار با داده‌های واقعی طراحی شده است.



انعطاف‌پذیری نوع داده

هر ستون می‌تواند نوع داده متفاوتی داشته باشد - اعداد، متن، منطقی یا تاریخ. این ویژگی باعث می‌شود دیتافریم برای داده‌های واقعی مناسب باشد.



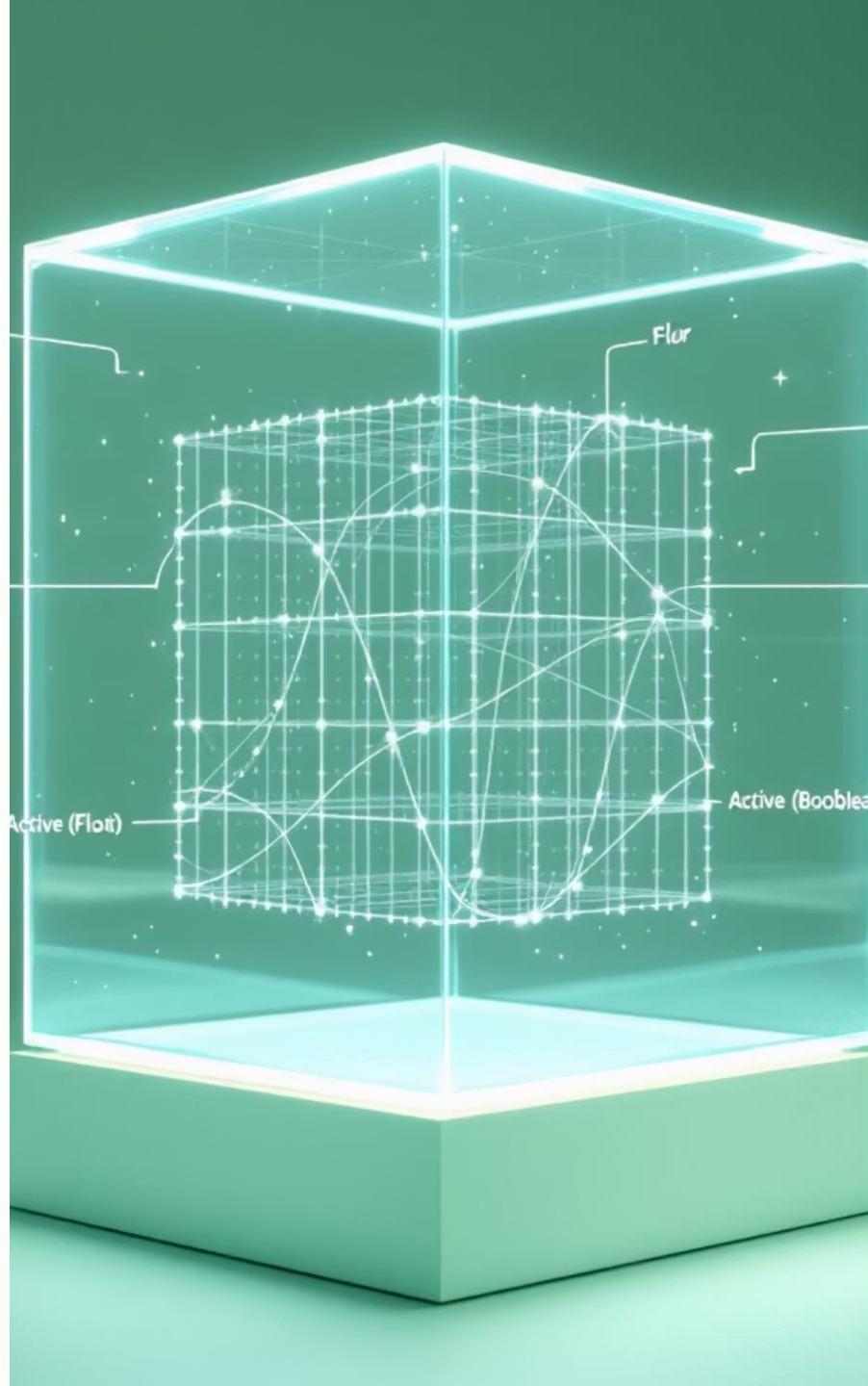
ساختار جدولی

دیتافریم یک ساختار داده‌ای دوبعدی است که شبیه به جداول اکسل یا جداول پایگاه داده عمل می‌کند و داده‌ها را به صورت سطر و ستون نگهداری می‌کند.



طول یکسان ستون‌ها

همه ستون‌ها باید دارای تعداد سطرهای یکسان باشند تا یکپارچگی و سازگاری داده‌ها حفظ شود.



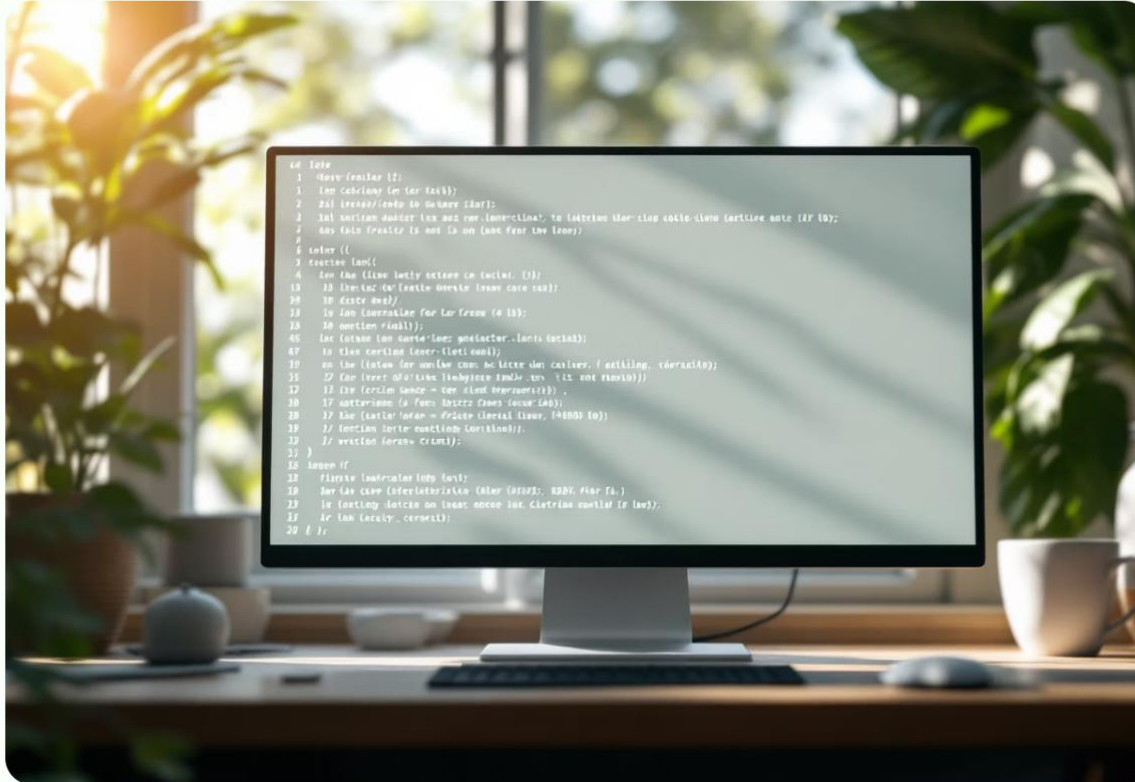
ایجاد دیتافریم

ساخت دیتافریم با `data.frame()`

برای ایجاد دیتافریم از تابع `data.frame()` استفاده می‌کنیم. هر ستون را با نام و مقادیرش تعریف می‌کنیم:

```
df <- data.frame(  
  name = c("علی", "سارا", "رضا"),  
  age = c(23, 21, 25),  
  score = c(18.5, 19, 17.5)  
)  
  
print(df)
```

این کد یک دیتافریم با سه ستون و سه سطر ایجاد می‌کند که اطلاعات دانشجویان را نگهداری می‌کند.



نکات مهم:

- تمام ستون‌ها باید طول یکسان داشته باشند
- نام ستون‌ها به صورت خودکار از نام متغیرها گرفته می‌شود
- می‌توانید انواع داده مختلف را در ستون‌های مختلف ترکیب کنید

دسترسی به داده‌ها

روش‌های مختلفی برای دسترسی به ستون‌ها، سطرها و عناصر خاص در دیتافریم وجود دارد:

1

دسترسی به ستون

با استفاده از علامت \$ می‌توانید به یک ستون دسترسی پیدا کنید:

```
df$name  
# خروجی: "علی" "سارا" "رضا"
```

2

دسترسی به سطر

با استفاده از ایندکس‌گذاری می‌توانید یک سطر کامل را انتخاب کنید:

```
df[2, ]  
# خروجی: سطر دوم با تمام ستون‌ها
```

3

دسترسی به ستون با ایندکس

می‌توانید با شماره ستون نیز به آن دسترسی داشته باشید:

```
df[, 3]  
# خروجی: ستون سوم (score)
```

4

دسترسی به عنصر خاص

برای دسترسی به یک سلول مشخص از ایندکس سطر و ستون استفاده کنید:

```
df[2, 3]  
# خروجی: نمره سارا (19)
```

کار با دیتافریم‌ها

توابع پرکاربرد



mean(df\$score)

محاسبه میانگین نمرات دانشجویان



summary(df)

نمایش خلاصه آماری کامل از تمام ستون‌ها



nrow(df)

شمارش تعداد سطرهای دیتافریم



ncol(df)

شمارش تعداد ستون‌های دیتافریم

افزودن داده‌های جدید

می‌توانید به راحتی ستون‌ها و سطرهای جدید به دیتافریم اضافه کنید:

افزودن ستون جدید

```
df$passed <- df$score >= 18
```

این کد یک ستون منطقی جدید ایجاد می‌کند که نشان می‌دهد آیا دانشجو قبول شده است یا خیر.

افزودن سطر جدید

```
new_row <- data.frame(  
  name = "مینا",  
  age = 22,  
  score = 20,  
  passed = TRUE  
)
```

```
df <- rbind(df, new_row)
```

با استفاده از `rbind()` می‌توانید سطرهای جدید را به دیتافریم اضافه کنید.

دیتافریم vs لیست در R

دو ساختار داده پرکاربرد با تفاوت‌های کلیدی



لیست (List)

ساختار: یک بعدی و بسیار انعطاف‌پذیر

طول: عناصر می‌توانند طول‌های مختلف داشته باشند

نوع داده: هر عنصر می‌تواند هر نوع داده‌ای باشد - عدد، متن، بردار، ماتریس، حتی دیتافریم!

دسترسی: از طریق `[[ ]]` یا `$`

کاربرد اصلی: ذخیره داده‌های ترکیبی و پیچیده، خروجی توابع آماری



دیتافریم (Data Frame)

ساختار: دو بعدی، شبیه جدول اکسل

طول: همه ستون‌ها باید طول یکسان داشته باشند

نوع داده: هر ستون می‌تواند نوع متفاوتی داشته باشد (عددی، متنی، منطقی)

دسترسی: از طریق `[row, col]` یا `$`

کاربرد اصلی: تحلیل داده‌های جدولی، محاسبات آماری، گزارش‌گیری

مثال لیست

```
my_list <- list(  
  name = "محمد",  
  age = 28,  
  scores = c(85, 90, 78)  
)
```

مجموعه‌ای انعطاف‌پذیر از عناصر مختلف

مثال دیتافریم

```
df <- data.frame(  
  name = c("علی", "سارا"),  
  age = c(25, 30)  
)
```

جدول منظم با ردیف‌ها و ستون‌های هم‌اندازه

فاکتورها (Factors): داده‌های رسته‌ای

فاکتورها برای متغیرهای کیفی که مقادیر مشخصی دارند، بسیار مناسب‌اند.

ساخت فاکتور

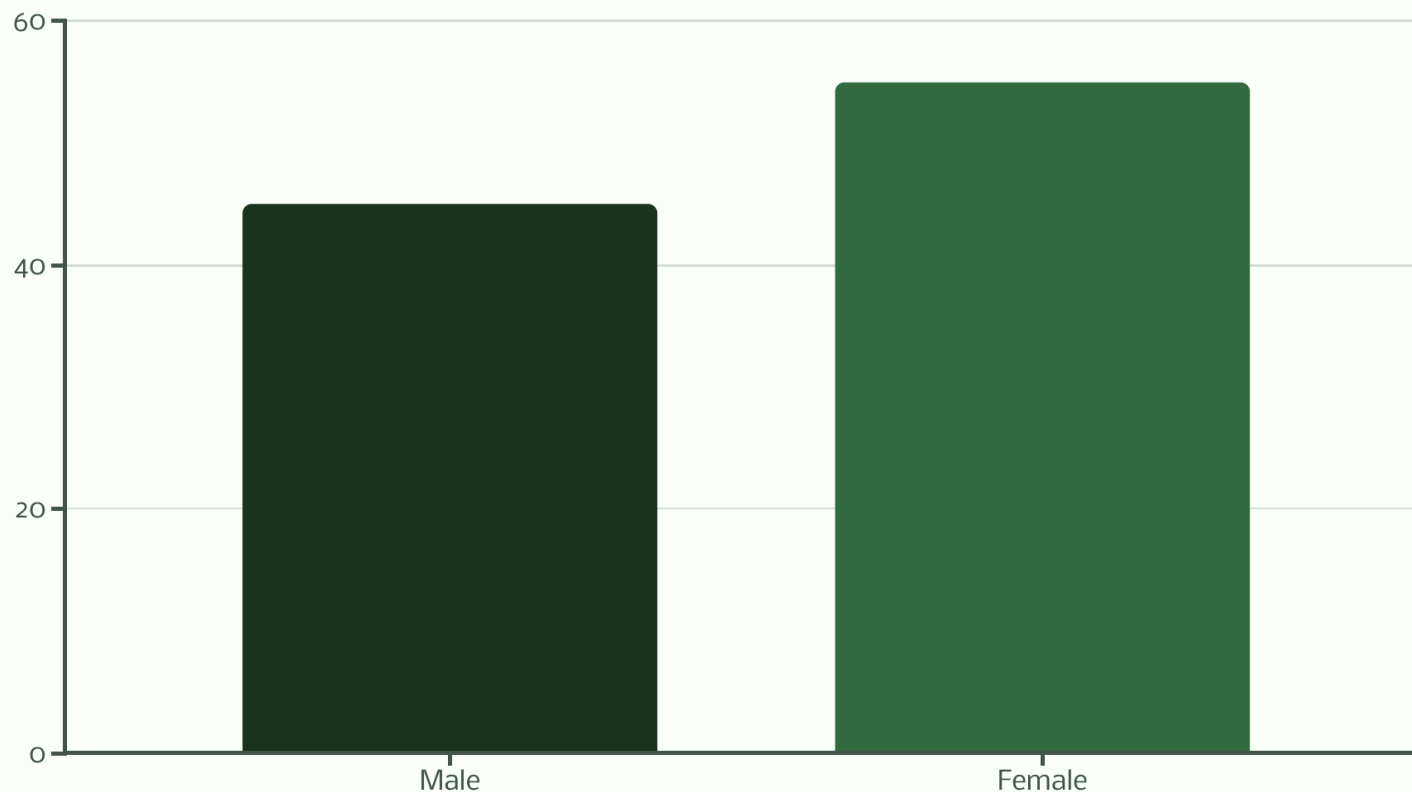
با **factor()** می‌توانید متغیرهای دسته‌ای بسازید.

مثال: `gender <- factor(c("Male", "Female", "Male"))`

توابع کاربردی

levels(): سطوح فاکتور را نشان می‌دهد

summary(): فراوانی هر سطح را گزارش می‌کند





ورود و خروج داده‌ها در R



پردازش و تحلیل

اعمال عملیات آماری، ایجاد مدل‌ها و تجزیه و تحلیل داده‌های بارگذاری شده

ورود داده‌ها

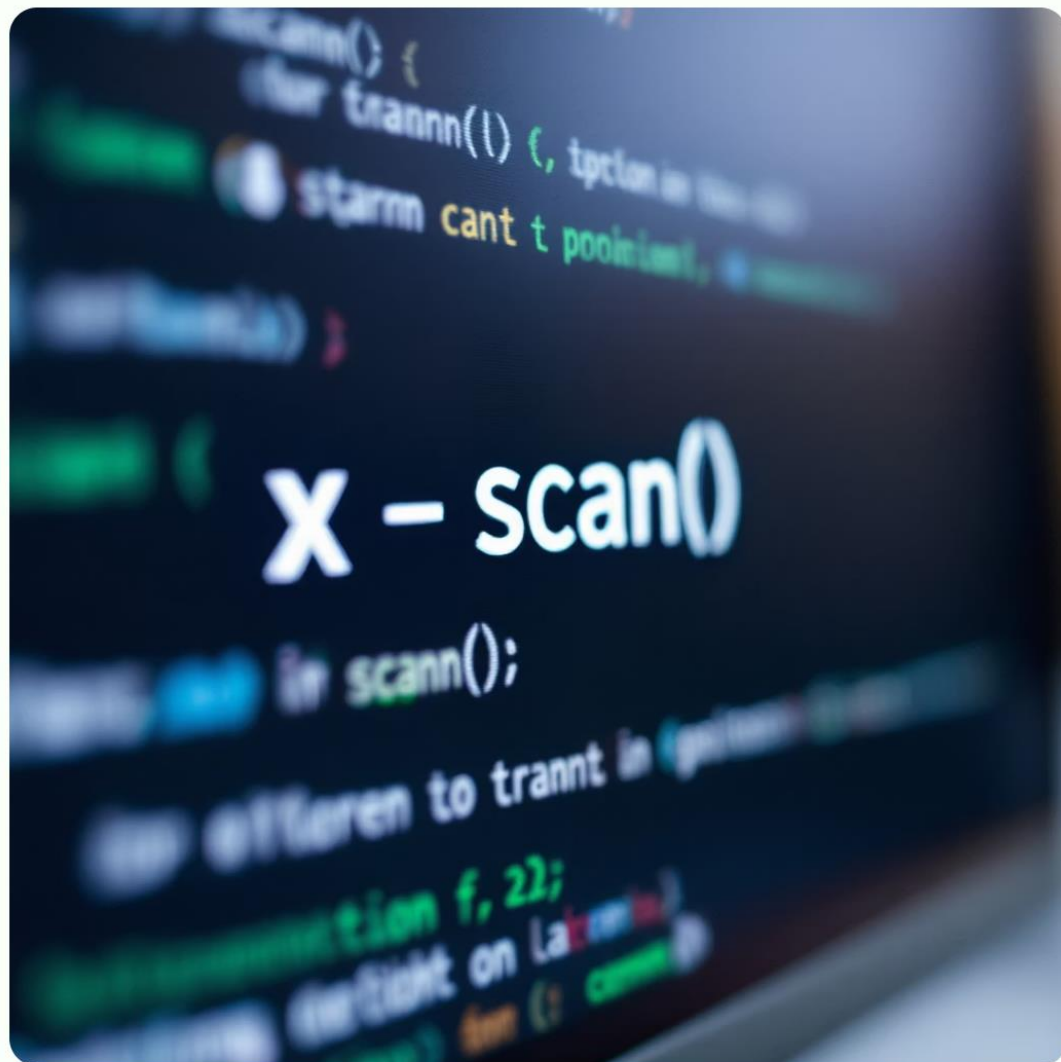
اولین قدم در هر تحلیل آماری، بارگذاری و آماده‌سازی داده‌های خام است



ذخیره نتایج

آخرین مرحله شامل صادرات نتایج تحلیل به فرمت‌های قابل استفاده

خواندن داده‌ها از کاربر



تابع scan() برای ورودی تعاملی

امکان دریافت داده‌ها مستقیماً از کاربر در زمان اجرا

x <- scan() ورود اعداد:

- ورود متن: `y <- scan(what = character())`
- محدودیت تعداد: `z <- scan(n = 3)`

این روش برای پروژه‌های کوچک و آموزشی مناسب است.

خواندن فایل‌های داده

فایل‌های CSV

استاندارد ترین فرمت انتقال داده

```
read.csv("data.csv")
```

مناسب برای داده‌های جدولی ساده

فایل‌های متنی (TXT)

برای داده‌های جداسازی شده با کاراکتر خاص

```
read.table("grades.txt", header=TRUE, sep="\t")
```

قابل تنظیم برای فرمت‌های مختلف



CSV

نوشتن و ذخیره داده‌ها

ذخیره به فرمت CSV

```
write.csv(df, "scores.csv", row.names=FALSE)
```

محبوب‌ترین فرمت برای تبادل داده

1

ذخیره متنی با جداکننده تب

```
write.table(df, "scores.txt", sep="\t")
```

مناسب برای داده‌های پیچیده‌تر

2

مدیریت داده‌های گمشده



محاسبه با نادیده گیری NA

```
mean(x, na.rm=TRUE)
```

انجام عملیات آماری بدون در نظر گیری مقادیر گمشده



حذف ردیف‌های ناقص

```
na.omit(x)
```

حذف کامل ردیف‌هایی که حاوی مقدار گمشده هستند



شناسایی مقادیر گمشده

```
which(is.na(x))
```

یافتن موقعیت دقیق داده‌های گمشده در مجموعه داده

بسته‌ها (Packages) در R



گسترش قابلیت‌های R

بسته‌ها امکانات اضافی و تخصصی به R اضافه می‌کنند

- نصب: `install.packages("readxl")`
- بارگذاری: `library(readxl)`
- خواندن (Excel): `read_excel("students.xlsx", sheet=2)`

دریافت داده از اینترنت



خواندن متن خط به خط

```
readLines("https://example.com/info.txt")
```

مناسب برای فایل‌های متنی و لاگ‌های آنلاین



خواندن مستقیم CSV

```
read.csv("https://example.com/data.csv")
```

بارگذاری مستقیم فایل‌های آنلاین بدون ذخیره محلی

حلقه‌ها و شرطها در R

شرطها (Conditionals)

اجرای شرطی کد بر اساس شرایط مشخص

- دستور `if`
- دستور `else`

اتوماسیون تحلیل

انجام خودکار تحلیل‌های پیچیده و تکراری



حلقه‌ها (Loops)

تکرار خودکار عملیات روی مجموعه داده‌ها

- `for loops`
- `while loops`

دستور if و else در R

ساختارهای شرطی ابزارهای قدرتمندی برای کنترل جریان اجرای برنامه هستند. با استفاده از دستورات if و else می‌توانید تصمیم‌گیری‌های هوشمند در کد خود پیاده‌سازی کنید.

ساختار پایه

```
{ (شرط) if
# کدی که اجرا می‌شود
# اگر شرط درست باشد
} else {
# کدی که اجرا می‌شود
# اگر شرط نادرست باشد
}
```

مثال کاربردی

```
x <- 5
if (x > 3) {
  print("بزرگتر از ۳")
} else {
  print("کوچکتر یا مساوی ۳")
}
```

شرط درست

اگر شرط TRUE باشد، بلوک کد داخل if اجرا می‌شود

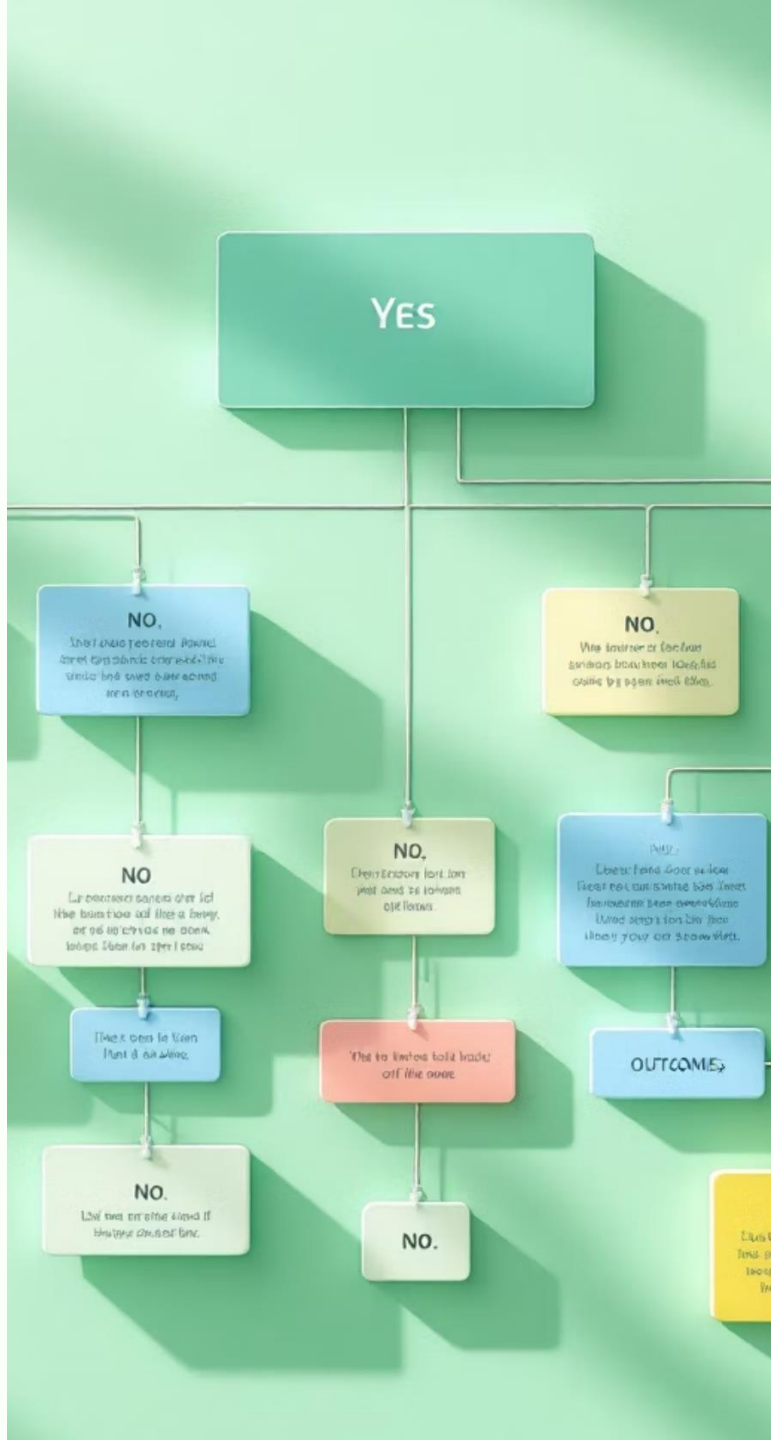
شرط نادرست

اگر شرط FALSE باشد، بلوک کد داخل else اجرا می‌شود

انعطاف‌پذیری

می‌توانید else را حذف کنید یا از if else برای شرایط چندگانه استفاده کنید

نکته مهم: شرط باید مقدار منطقی (TRUE یا FALSE) برگرداند تا دستور if بتواند تصمیم‌گیری کند.



حلقه for: گام به گام از ابتدا تا انتها

حلقه for یکی از پرکاربردترین ساختارهای تکراری است که به شما امکان می‌دهد یک بلوک کد را برای تمام عناصر یک مجموعه اجرا کنید.

ساختار کلی:

```
for (مجموعه in متغیر) {  
    دستورات  
}
```

مثال کاربردی:

```
for (i in 1:10) {  
    print(i)  
}
```



```
while =1 (loop,  
repeated = actions)
```

```
while = repioop)  
while + or exit:
```

حلقه **while**: تکرار با شرط



نکته مهم

حتماً شرط توقف را درست تعریف کنید تا حلقه بی‌نهایت نشود و برنامه متوقف نماند!



مثال عملی

```
x <- 1  
while (x <= 3) {  
  print(x)  
  x <- x + 1  
}
```

خروجی: 1، 2، 3



مفهوم اصلی

حلقه **while** تا زمانی که شرط مشخصی برقرار است، کد را اجرا می‌کند. این حلقه برای مواقعی مناسب است که تعداد تکرارها از قبل مشخص نیست.

حلقه repeat: تکرار بدون محدودیت



چه زمانی از **repeat** استفاده کنیم؟

زمانی که می‌خواهید کدی را به صورت نامحدود اجرا کنید تا شرط خاصی برقرار شود، از حلقه **repeat** استفاده می‌کنیم.

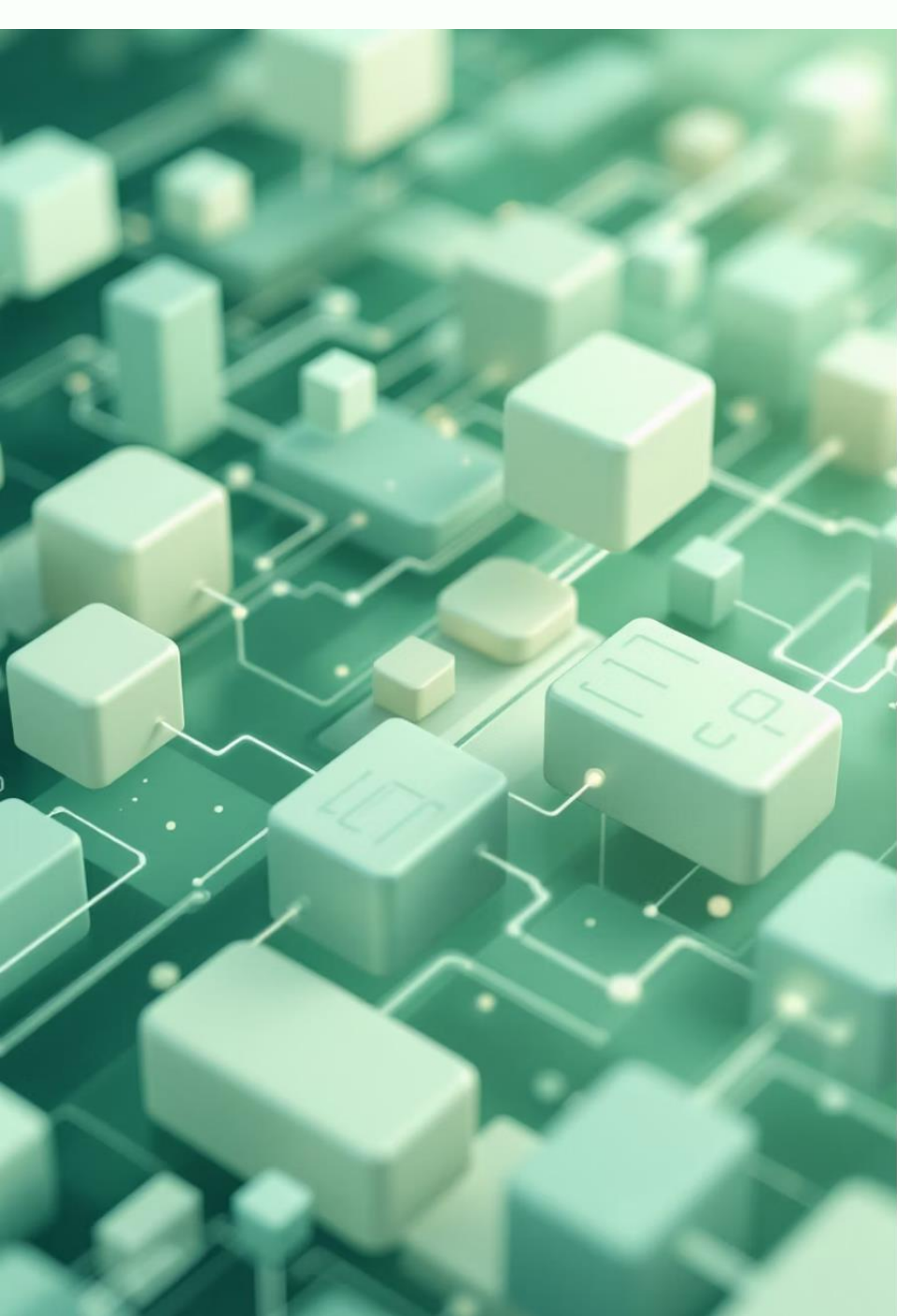
ساختار کلی:

```
repeat {  
  دستورات  
  if (شرط_توقف) break  
}
```

مثال کاربردی:

```
x <- 1  
repeat {  
  print(x)  
  x <- x + 1  
  if (x > 4) break  
}
```

این حلقه اعداد 1 تا 4 را چاپ می‌کند و سپس متوقف می‌شود.



مقدمه بر توابع در R

توابع = کدهای قابل استفاده مجدد

توابع به شما امکان می‌دهند بلوک‌های کد را یک بار بنویسید و در جاهای مختلف استفاده کنید، بدون نیاز به تکرار کد.

سازماندهی و کاهش خطا

با استفاده از توابع، کد شما منظم‌تر، خواناتر و کمتر مستعد خطا می‌شود. تغییرات را فقط یک‌بار در تابع اعمال کنید.

ورودی و خروجی

توابع می‌توانند داده‌ها را به عنوان ورودی دریافت کنند، روی آن‌ها پردازش انجام دهند و نتیجه را به عنوان خروجی برگردانند.

ساختار تابع در R



خروجی (return)

نتیجه نهایی که برمی‌گردد

25



پردازش (بدنه تابع)

کدهایی که روی ورودی اجرا می‌شوند

```
function(x) {  
  x ^ 2  
}
```



ورودی (آرگومان‌ها)

داده‌هایی که به تابع می‌دهیم

x = 5

مثال کامل:

```
myfun <- function(x) {  
  result <- x ^ 2  
  return(result)  
}
```

استفاده از تابع
myfun(5) # خروجی: 25

انواع توابع: با و بدون ورودی

تابع بدون ورودی

این نوع توابع هیچ پارامتری دریافت نمی‌کنند و معمولاً یک عمل ثابت انجام می‌دهند.

مثال:

```
hello <- function() {  
  print("Hello R!")  
  print("خوش آمدید")  
}
```

اجرای تابع #
hello()

خروجی: دو پیام چاپ می‌شود.

تابع با ورودی

این توابع یک یا چند پارامتر دریافت می‌کنند و بر اساس آن‌ها عمل می‌کنند.

مثال:

```
square <- function(x) {  
  return(x^2)  
}
```

اجرای تابع #
square(5) # خروجی: 25
square(10) # خروجی: 100

انعطاف‌پذیری: با ورودی‌های مختلف،
خروجی‌های متفاوت می‌گیرید.



با تشکر از حضور شما



آرزوی موفقیت

امیدواریم مطالب و دانش ارائه شده در این جلسه، راهگشای مسیر علمی و پژوهشی شما باشد و در دستیابی به اهداف تحصیلی‌تان یاری‌رسان باشد.



سپاسگزاری

از حضور گرم و ارزشمند شما در این کارگاه دانشگاهی صمیمانه سپاسگزاریم. توجه و مشارکت فعال شما، این نشست را به تجربه‌ای ماندگار تبدیل کرد.



پیشرفت مستمر

برای تمامی شما دانشجویان و همکاران عزیز، آرزوی موفقیت روزافزون، پیشرفت علمی پایدار و تحقق آرزوهای دانشگاهی داریم.

با سپاس فراوان | Thank You

