

راهنمای جامع و گام به گام نصب و کار با R و RStudio

مفاهیم کلیدی این راهنما:

- **R**: یک زبان برنامه‌نویسی و محیط نرم‌افزاری رایگان برای محاسبات آماری و بیوانفورماتیک. مثل یک جعبه ابزار قدرتمند برای کار با داده‌هاست.
- **RStudio**: یک محیط توسعه یکپارچه (IDE) که کار با R رو خیلی راحت‌تر می‌کنه. مثل یک میز کار منظم و مجهز برای استفاده از جعبه ابزار R.
- **بیوانفورماتیک**: استفاده از ابزارهای محاسباتی و آماری برای تجزیه و تحلیل داده‌های زیستی، مثل توالی ژنوم‌ها، بیان ژن‌ها و ساختار پروتئین‌ها. R یک ابزار اساسی در این زمینه است.
- **کتابخانه‌ها (Packages)**: مجموعه‌ای از توابع و داده‌ها که قابلیت‌های R را افزایش می‌دهند. stringr یک نمونه از این کتابخانه‌هاست که برای کار با متن (رشته‌ها) استفاده می‌شود.
- **Splitting Strings**: عملیات تقسیم یک رشته متنی به بخش‌های کوچک‌تر بر اساس یک جداکننده خاص.

حالا بریم سراغ جزئیات:

بخش اول: نصب R و RStudio

نصب R و RStudio مثل چیدن میز کار و آوردن ابزارهاست قبل از شروع یک پروژه جذاب! نیاز هست که اول R (جعبه ابزار) رو نصب کنی و بعد RStudio (میز کار) رو.

گام اول: نصب R

R پایه و اساس کار ماست. برای نصبش مراحل زیر رو دنبال کن:

۱. ورود به سایت رسمی CRAN:

- وارد این آدرس بشو: <https://cran.r-project.org>
- CRAN مخفف Comprehensive R Archive Network هست و مخزن اصلی R محسوب میشه.

۲. انتخاب سیستم عامل:

- بسته به اینکه از چه سیستم عاملی استفاده می‌کنی (Windows, macOS, Linux)، روی لینک مربوطه کلیک کن.
- مثلاً اگه ویندوز داری، روی "Download R for Windows" کلیک کن. ۳. دانلود نسخه پایه (base):

- در صفحه مربوط به سیستم عاملت، معمولاً گزینه‌ای به اسم "base" وجود دارد. این همون نسخه اولیه و پایه R هست که نیاز داری. روش کلیک کن.
- بعد روی لینک "Download R x.y.z for Windows" (یا نسخه مربوط به سیستم عاملت) کلیک کن. (X.y.Z شماره نسخه هست و ممکنه فرق کنه).

۳. اجرای فایل نصب:

- فایل دانلود شده رو اجرا کن. مراحل نصبش مثل نصب نرم‌افزارهای عادی.
- میتونی تنظیمات پیش‌فرض رو قبول کنی، مگر اینکه نیاز خاصی داشته باشی.

نکات تکمیلی برای نصب R:

- اجازه مدیر (Administrator rights): ممکنه برای نصب نیاز به اجازه مدیر سیستم داشته باشی.
- انتخاب مسیر نصب: معمولاً بهتره در مسیر پیش‌فرض نصب بشه، مگر اینکه دلیل خاصی برای تغییرش داشته باشی.
- ۳۲ بیتی یا ۶۴ بیتی: نصاب‌های جدید R معمولاً هر دو نسخه رو شامل میشن و خودشون تشخیص میدن. اگه انتخاب دستی داشتی، مطمئن شو که نسخه مناسب سیستم عاملت رو انتخاب می‌کنی.

گام دوم: نصب RStudio

حالا که جعبه ابزار[®] رو داری، وقتشه میز کار (RStudio) رو بچینی!

۱. ورود به سایت رسمی RStudio:

- وارد این آدرس بشو: <https://posit.co/downloads> (اسم شرکت سازنده RStudio الان Posit هست).

۲. انتخاب نسخه RStudio Desktop:

- در بخش "RStudio Desktop"، گزینه "FREE" رو پیدا کن و روی "Download" کلیک کن. این نسخه رایگان و برای استفاده شخصی یا دانشگاهی کافیه.

۳. دانلود نصاب مناسب سیستم عامل:

- سایت RStudio به صورت خودکار سیستم عامل تو رو تشخیص میده و لینک دانلود مناسب رو پیشنهاد میکنه. روی اون لینک کلیک کن تا دانلود شروع بشه.

۴. اجرای فایل نصب:

- فایل دانلود شده رو اجرا کن. مراحل نصب شبیه نصب R هست و معمولاً با پذیرش تنظیمات پیش‌فرض مشکلی نخواهی داشت.

نکات تکمیلی برای نصب RStudio:

- نصب R قبل از RStudio: حتماً مطمئن شو که R رو قبل از نصب RStudio نصب کردی. RStudio برای کار کردن به R نیاز داره.

- **اولین اجرا:** بعد از نصب، RStudio رو باز کن. محیطش معمولاً شامل ۴ پنل اصلی: Source (برای نوشتن کد)، Console (برای اجرای فوری کد و دیدن خروجی)، Environment/History (برای دیدن متغیرها و تاریخچه دستورات)، و Files/Plots/Packages/Help/Viewer (برای مدیریت فایل‌ها، دیدن نمودارها، پکیج‌ها و راهنما).

بخش دوم: تاریخچه و اهمیت R در علوم زیستی و بیوانفورماتیک

R فقط یک زبان برنامه‌نویسی نیست، بلکه یک جامعه بزرگ و یک اکوسیستم قدرتمند برای تحلیل داده‌هاست. نقش اون در علوم زیستی و بیوانفورماتیک بسیار حیاتی شده.

تاریخچه مختصری از R

- R در اوایل دهه ۹۰ میلادی توسط راس ایهاکا (Ross Ihaka) و رابرت جنتلمن (Robert Gentleman) در دانشگاه آکلند نیوزیلند به عنوان یک زبان آماری توسعه پیدا کرد.
- این زبان بر پایه زبان S که در آزمایشگاه‌های بل (Bell Labs) توسعه یافته بود، ساخته شد.
- هدف اصلی ایجاد زبانی بود که محیطی انعطاف‌پذیر و تعاملی برای تجزیه و تحلیل داده‌ها و گرافیک ارائه دهد.
- در سال ۱۹۹۷، پروژه R به صورت متن‌باز (Open Source) تحت گواهینامه GNU منتشر شد و از آن زمان به سرعت رشد کرده است.

اهمیت R در علوم زیستی و بیوانفورماتیک

چرا R اینقدر در زمینه ما محبوب و مهم شده؟

- **ابزار قدرتمند آماری:** علوم زیستی و بیوانفورماتیک به شدت مبتنی بر آمار هستند. R مجموعه‌ای بی‌نظیر از ابزارهای آماری را فراهم می‌کند، از آزمون‌های ساده t و ANOVA گرفته تا مدل‌های پیچیده رگرسیون و تحلیل‌های چندمتغیره.
- **جامعه Bioconductor:** این شاید مهمترین دلیل باشد! Bioconductor یک پروژه بزرگ و متن‌باز است که پکیج‌های تخصصی R را برای تجزیه و تحلیل داده‌های ژنومیک، پروتئومیک، متابولومیک و سایر داده‌های "Omics" توسعه می‌دهد. هزاران پکیج در Bioconductor وجود دارد که ابزارهایی برای:
 - تجزیه و تحلیل داده‌های RNA-Seq (مثل پکیج‌های DESeq2, edgeR)
 - تحلیل داده‌های میکروآرای (Microarray)
 - پردازش داده‌های توالی‌یابی نسل جدید (NGS)
 - بیان افتراقی ژن‌ها (Differential Gene Expression)
 - تحلیل شبکه‌های زیستی
 - ویژوال‌سازی داده‌های حجیم زیستی
 - و بسیاری کاربردهای دیگر فراهم می‌کنند.

- **مصورسازی داده‌ها:** R قابلیت‌های گرافیکی فوق‌العاده‌ای دارد (مخصوصاً با پکیج‌هایی مثل ggplot2) که امکان ایجاد نمودارهای پیچیده و گویا از داده‌های زیستی را فراهم می‌کند.
- **متن‌باز و رایگان:** این ویژگی باعث شده که R به صورت گسترده در دانشگاه‌ها و مراکز تحقیقاتی در سراسر جهان پذیرفته شود. همچنین، ماهیت متن‌باز آن به این معنی است که هر کسی می‌تواند کد را بررسی کند، بهبود بخشد و پکیج‌های جدید بنویسد.
- **انعطاف‌پذیری و قابلیت اسکرپت‌نویسی:** R به تو اجازه می‌دهد تحلیل‌های خودت رو به صورت اسکرپت (فایل کد) بنویسی. این کار باعث میشه تحلیل‌هایت قابل تکرار (Reproducible) باشند؛ یعنی هر زمان و توسط هر کسی قابل اجرا با همان نتایج اولیه باشند. این برای صحت نتایج علمی بسیار مهمه.
- **جامعه کاربری بزرگ:** به دلیل استفاده گسترده، جامعه کاربری R بسیار فعال است. این یعنی اگر به مشکلی برخوردی، احتمالاً کسی قبلاً آن مشکل را داشته و راه‌حل آن آنلاین (مثلاً در سایت Stack Overflow یا انجمن‌های Bioconductor) پیدا می‌شود.

خلاصه: R به دلیل قدرت آماری، اکوسیستم غنی پکیج‌های تخصصی (مخصوصاً Bioconductor)، قابلیت‌های ویژوال‌سازی عالی، رایگان بودن و قابلیت تکرارپذیری تحلیل‌ها، به ابزاری ضروری برای دانشجویان و محققان در علوم زیستی و بیوانفورماتیک تبدیل شده است.

ویژگی	R	RStudio
نقش	زبان برنامه‌نویسی و موتور محاسباتی	محیط توسعه یکپارچه (IDE)
تشبیه	موتور ماشین	داشبورد، فرمان و کابین ماشین
عملکرد	اجرای کدها و محاسبات آماری	آسان‌سازی کدنویسی، مدیریت پروژه و نمایش خروجی‌ها
وابستگی	مستقل است	برای کار کردن به R نیاز دارد

بخش سوم: تحلیل کد R و کارکرد stringr (جدا کردن ژن‌ها)

حالا بریم سراغ قطعه کد استفاده شده در این دوره. این کد یک مثال عالی از نحوه استفاده از R برای پردازش داده‌های زیستی (نتایج یک تحلیل بیان ژن) و پاکسازی داده‌های متنی (لیست ژن‌ها) است.

مفاهیم کلیدی این بخش:

- **LogFC (Log Fold Change):** مقداری که نشان‌دهنده تغییر در بیان یک ژن بین دو شرط (مثلاً بیمار در مقابل سالم) است. $\text{LogFC} < 0$ یعنی افزایش بیان، $\text{LogFC} > 0$ یعنی کاهش بیان. $\text{LogFC} 2$ یا -2 معمولاً به عنوان آستانه برای تغییر معنی‌دار در نظر گرفته می‌شود.
- **adj.P.Val (Adjusted P-value):** مقدار P تعدیل شده که نشان‌دهنده معنی‌داری آماری تغییر بیان است، با در نظر گرفتن تست‌های متعدد (Multiple Testing Correction). مقدار کوچک (معمولاً کمتر از ۰.۰۵) نشان‌دهنده معنی‌داری است.

- **Subset**: انتخاب زیرمجموعه‌ای از داده‌ها بر اساس شرایط خاص.
- **stringr**: پکیجی در R برای آسان‌تر کردن کار با رشته‌های متنی.
- **str_split()**: تابعی از پکیج stringr که یک رشته را بر اساس یک الگو (Pattern) به بخش‌های کوچک‌تر تقسیم می‌کند.
- **unique()**: تابعی در R که مقادیر تکراری را از یک بردار یا لیست حذف کرده و فقط مقادیر منحصر به فرد را نگه می‌دارد. R``

Split Genes by stringr Step by Step

```
library(stringr)
```

این خط پکیج stringr را بارگذاری می‌کند تا بتوانیم از توابع آن استفاده کنیم.

Up# این یک کامنت است و نشان می‌دهد بخش بعدی مربوط به ژن‌های Up-regulated (افزایش بیان یافته) است.

```
up = subset(GSE37364, logFC > 2 & adj.P.Val<0.05)
```

این خط داده‌های dataset به نام GSE37364 را فیلتر می‌کند.

فقط ردیف‌هایی انتخاب می‌شوند که:

۱. ستون 'logFC' آن‌ها بزرگتر از ۲ باشد (یعنی حداقل ۴ برابر افزایش بیان)
۲. ستون 'adj.P.Val' آن‌ها کوچکتر از ۰.۰۵ باشد (یعنی تغییر بیان از نظر آماری معنی‌دار باشد).

نتیجه فیلتر در متغیری به نام 'up' ذخیره می‌شود.

```
up.genes <- unique(str_split(up$Gene.symbol, "///", simplify = TRUE)[, 1])
```

این خط اصلی کار ماست که از stringr استفاده می‌کند! بیایید آن را بخش به بخش بررسی کنیم:

۱. up\$Gene.symbol: این قسمت ستون 'Gene.symbol' را از داده‌های فیلتر شده 'up' انتخاب می‌کند.

فرض بر این است که در این ستون، اسامی ژن‌ها وجود دارد و گاهی اوقات ممکن است چندین نام ژن با جداکننده “///” کنار هم نوشته شده باشند (مثلاً “GeneA///GeneB”).

۲. `str_split(..., “///”, simplify = TRUE)`: تابع `str_split` فراخوانی می‌شود.

- آرگومان اول (`up$Gene.symbol`): رشته‌هایی هستند که باید تقسیم شوند.
- آرگومان دوم (“///”): الگوی جداکننده است. تابع هر جا که این الگو را ببیند، رشته را از آنجا تقسیم می‌کند.
- آرگومان سوم (`simplify = TRUE`): باعث می‌شود خروجی `str_split` به جای یک لیست پیچیده، یک ماتریس (`matrix`) باشد. هر ردیف این ماتریس متناظر با یک ردیف اصلی در `up$Gene.symbol` است و ستون‌های آن شامل بخش‌های جدا شده رشته اصلی هستند.

۳. `[...]`, `[۱]`: این قسمت از نتیجه ماتریس تولید شده توسط `str_split`، فقط ستون اول را انتخاب می‌کند.

این یعنی اگر یک ردیف شامل چندین ژن (مثل “GeneA///GeneB”) باشد، ما فقط ژن اول (“GeneA”) را انتخاب می‌کنیم. این نکته مهمی است و به این معنی است که کد فقط اولین ژن را از لیست ژن‌های چندگانه استخراج می‌کند.

۴. `unique(...)`: در نهایت، تابع `unique` روی لیستی از ژن‌های انتخاب شده (فقط اولین ژن‌ها) اعمال می‌شود.

این باعث می‌شود هر نام ژنی فقط یک بار در لیست نهایی ‘`up.genes`’ وجود داشته باشد و تکراری‌ها حذف شوند.

نتیجه نهایی اینکه (لیست منحصر به فرد ژن‌های افزایش بیان یافته) در متغیر ‘`up.genes`’ ذخیره می‌شود.

```
write.table(up.genes, "GSE37364_up.txt", col.names = F, row.names=F, sep="\t", quote=F)
```

این خط لیست ژن‌های افزایش بیان یافته (‘`up.genes`’) را در یک فایل متنی ذخیره می‌کند.

- “GSE37364_up.txt”: نام فایل خروجی است.
- `col.names = F`: یعنی نام ستون‌ها در فایل نوشته نشود.
- `row.names=F`: یعنی شماره ردیف‌ها در فایل نوشته نشود.
- `sep=“\t”`: یعنی ستون‌ها (اگر بیش از یکی بودند) با کاراکتر `Tab` از هم جدا شوند (برای فایل‌های متنی استاندارد).
- `quote=F`: یعنی اطراف مقادیر متنی علامت نقل قول (“”) گذاشته نشود.

بخش بعدی مربوط به ژن‌های Down-regulated (کاهش بیان یافته) است.

```
down = subset(GSE37364, logFC < -2 & adj.P.Val<0.05)
```

این خط مشابه بخش بالا، داده‌ها را برای ژن‌های کاهش بیان یافته فیلتر می‌کند.

فقط ردیف‌هایی که 'logFC' آن‌ها کمتر از منفی ۲ (یعنی حداقل ۴ برابر کاهش بیان) و 'adj.P.Val' آن‌ها کمتر از ۰.۰۵ باشد، انتخاب می‌شوند.

```
down.genes <- unique(str_split(down$Gene.symbol, "///", simplify = TRUE)[, 1])
```

این خط نیز مشابه بخش بالا، ستون 'Gene.symbol' را از داده‌های 'down' انتخاب کرده، بر اساس "///" تقسیم می‌کند، فقط اولین ژن را از هر ردیف برمی‌دارد، و در نهایت مقادیر منحصر به فرد را در متغیر 'down.genes' ذخیره می‌کند.

```
write.table(down.genes, "GSE37364_down.txt", col.names = F, row.names=F, sep="\t", quote=F)
```

این خط لیست ژن‌های کاهش بیان یافته ('down.genes') را در یک فایل متنی به نام "GSE37364_down.txt" ذخیره می‌کند، با همان فرمت‌بندی (بدون نام ستون/ردیف، با جداکننده Tab، بدون نقل قول).

جمع‌بندی و گام بعدی

شما با اجرای این کد، دو فایل متنی بسیار ارزشمند تولید کرده‌اید:

۱. GSE37364_up.txt: لیستی از ژن‌هایی که بیانشان به شدت افزایش یافته.

۲. GSE37364_down.txt: لیستی از ژن‌هایی که بیانشان به شدت کاهش یافته.

گام بعدی چیست؟ شما می‌توانید این لیست‌های ژنی را علاوه بر استفاده در Drug Repurposing و Gene-Drug Interaction که موضوع دوره ماست! به ابزارهای آنلاین تحلیل مسیر و غنی‌سازی (Pathway and Enrichment Analysis) مثل DAVID یا Metascope بدهید تا بفهمید این ژن‌ها در کدام مسیرهای بیولوژیکی، عملکرد مولکولی یا فرایندهای سلولی (مثلاً پاسخ ایمنی، آپوپتوز و...) نقش دارند. اینجاست که تحلیل داده به کشف بیولوژیکی منجر می‌شود!