

Packages in R

Learning objectives

- Introduce packages in R studio
- Installing packages
- Using packages in R studio

Packages in R

➤ What are packages in R?

- ❑ Packages are bundles of reusable R code, functions, data, tests, and documentation.
- ❑ They are designed to be shared and used by others, promoting collaboration and efficiency.

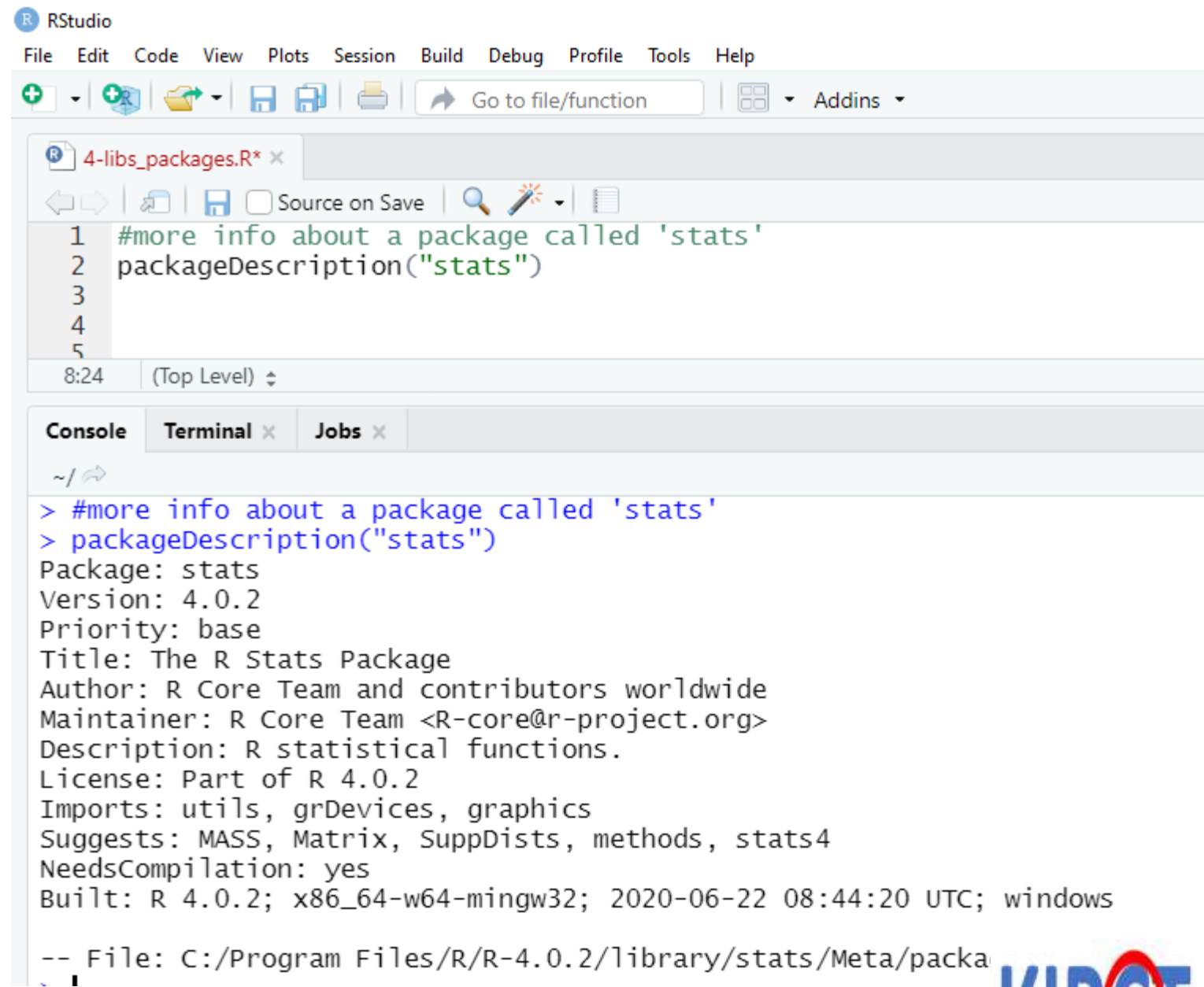
- ❑ Packages add new capabilities to R, allowing you to perform specific tasks or analyses that aren't part of the base R language.
- ❑ They cover a vast range of domains, from statistics and data manipulation to graphics, machine learning, and more.

-
- A package can contain multiple packages (eg., tidyverse package containing dplyr, ggplot2, lubridate, readr, purrr, tibble, tidyr etc.).
 - Packages can also make coding more efficient and help organize your work (eg., dplyr package)
 - You can find information about packages in the description file (shown on next slide)
 - ❖ The description file describes what the package does, name of authors, etc.
 - A standard set of packages are automatically available when you download and install R.

Suppose we heard about a cool package called “stats”

If we type the command: “packageDescription” and place the package name “stats” in quotations and then run the code, we will obtain more information about the package in the console package.

*note you cannot obtain information about packages that are not already installed



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor window shows a script named '4-libs_packages.R*' with the following code:

```
1 #more info about a package called 'stats'  
2 packageDescription("stats")  
3  
4  
5
```

The console window below the editor shows the output of the command:

```
> #more info about a package called 'stats'  
> packageDescription("stats")  
Package: stats  
Version: 4.0.2  
Priority: base  
Title: The R Stats Package  
Author: R Core Team and contributors worldwide  
Maintainer: R Core Team <R-core@r-project.org>  
Description: R statistical functions.  
License: Part of R 4.0.2  
Imports: utils, grDevices, graphics  
Suggests: MASS, Matrix, SuppDists, methods, stats4  
NeedsCompilation: yes  
Built: R 4.0.2; x86_64-w64-mingw32; 2020-06-22 08:44:20 UTC; windows  
-- File: C:/Program Files/R/R-4.0.2/library/stats/Meta/packa
```

By typing `help(package="stats")`, the R documentation will pop up on your lower right window where you will be able to access more information about the package.

```
4-libs_packagesR*
1 #more info about a package called 'stats'
2 packageDescription("stats")
3 help(package="stats")
4
5
6
7 (Top Level)
R Script

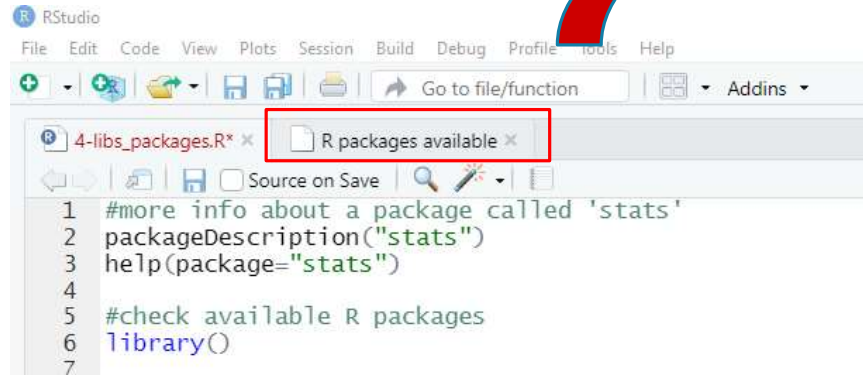
Console Terminal Jobs
> #more info about a package called 'stats'
> packageDescription("stats")
Package: stats
Version: 4.0.2
Priority: base
Title: The R Stats Package
Author: R Core Team and contributors worldwide
Maintainer: R Core Team <R-core@r-project.org>
Description: R statistical functions.
License: Part of R 4.0.2
Imports: utils, grDevices, graphics
Suggests: MASS, Matrix, SuppDists, methods, stats4
NeedsCompilation: yes
Built: R 4.0.2; x86_64-w64-mingw32; 2020-06-22 08:44:20 UTC; windows

-- File: C:/Program Files/R/R-4.0.2/library/stats/Meta/package.rds
> help(package="stats")
> |
```

The screenshot shows the R help window for the 'stats' package. The title bar reads 'R: The R Stats Package'. The main content area displays the package title 'The R Stats Package' with the R logo. Below this, it states 'Documentation for package 'stats' version 4.0.2'. There are two bullet points: 'DESCRIPTION file' and 'Code demos. Use `demo()` to run them.' Under the heading 'Help Pages', there is an alphabetical index 'A B C D E F G H I K L M N O P Q R S T U V W X misc'. The letter 'A' is expanded to show a list of functions: `acf` (Auto- and Cross- Covariance and -Correlation Function Estimation), `acf2AR` (Compute an AR Process Exactly Fitting an ACF), `add.scope` (Compute Allowed Changes in Adding to or Dropping from a Formula), `add1` (Add or Drop All Possible Single Terms to a Model), and `addmargins` (Puts Arbitrary Margins on Multidimensional Tables or Arrays).

Where can you find packages?

- Packages are stored under a directory called a “library” located in the R environment
- By default, R will install a number of packages during the initial installation.
- However, as your programming needs increase so will your repertoire of packages 😊
 - We will show you how to install new packages not already installed by default

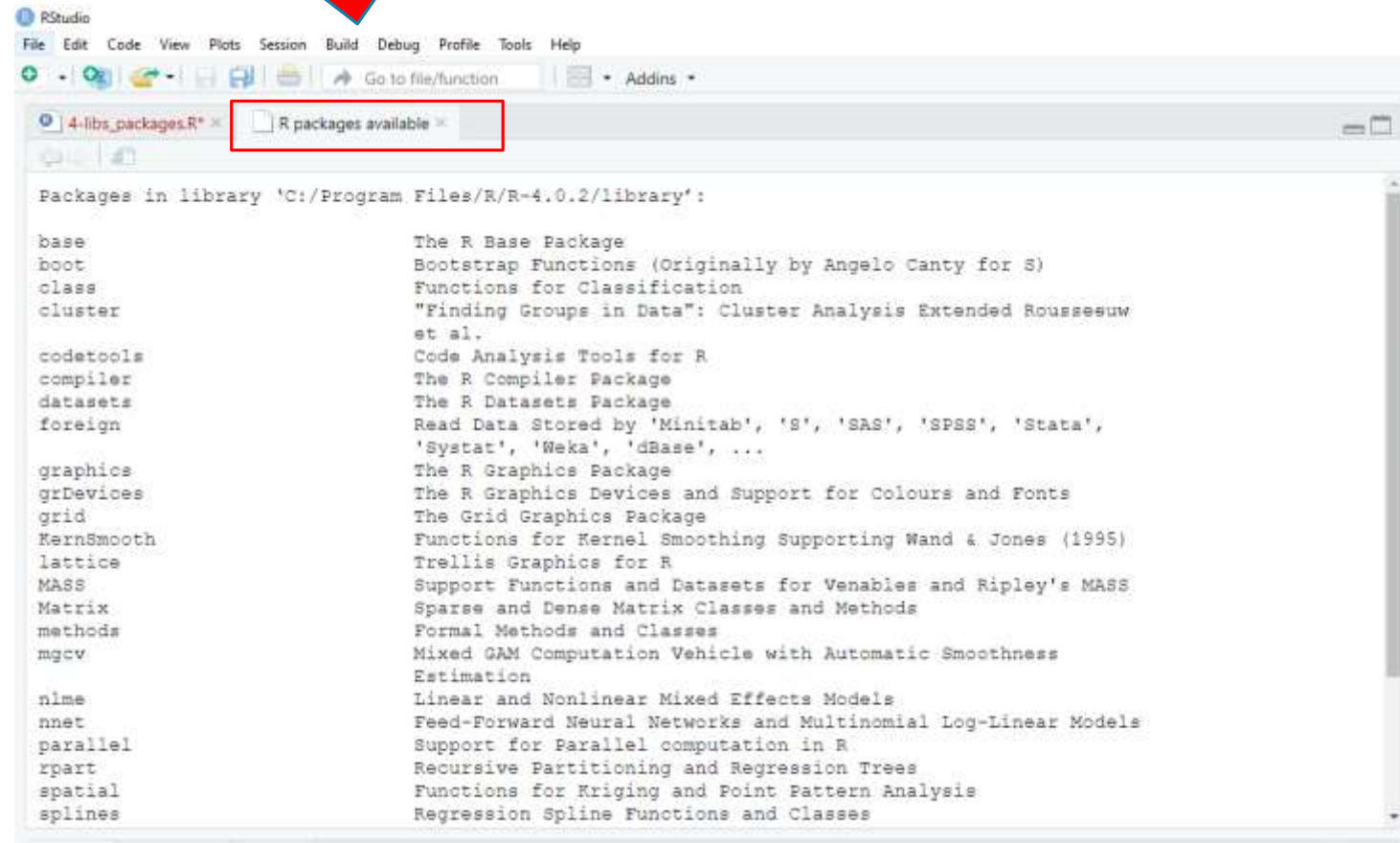


RStudio interface showing a script editor with the following code:

```
1 #more info about a package called 'stats'  
2 packageDescription("stats")  
3 help(package="stats")  
4  
5 #check available R packages  
6 library()  
7
```

A new tab titled "R packages available" is highlighted with a red box and a red arrow points to the right-hand screenshot.

When you type 'library()'
a new tab will
automatically open and
list the packages
available in your library



RStudio interface showing the output of the `library()` command. A new tab titled "R packages available" is highlighted with a red box. The output lists the following packages in the library 'C:/Program Files/R/R-4.0.2/library':

Package Name	Description
base	The R Base Package
boot	Bootstrap Functions (Originally by Angelo Canty for S)
class	Functions for Classification
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
compiler	The R Compiler Package
datasets	The R Datasets Package
foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...
graphics	The R Graphics Package
grDevices	The R Graphics Devices and Support for Colours and Fonts
grid	The Grid Graphics Package
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)
lattice	Trellis Graphics for R
MASS	Support Functions and Datasets for Venables and Ripley's MASS
Matrix	Sparse and Dense Matrix Classes and Methods
methods	Formal Methods and Classes
mgcv	Mixed GAM Computation Vehicle with Automatic Smoothness Estimation
nime	Linear and Nonlinear Mixed Effects Models
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models
parallel	Support for Parallel computation in R
rpart	Recursive Partitioning and Regression Trees
spatial	Functions for Kriging and Point Pattern Analysis
splines	Regression Spline Functions and Classes

Repositories

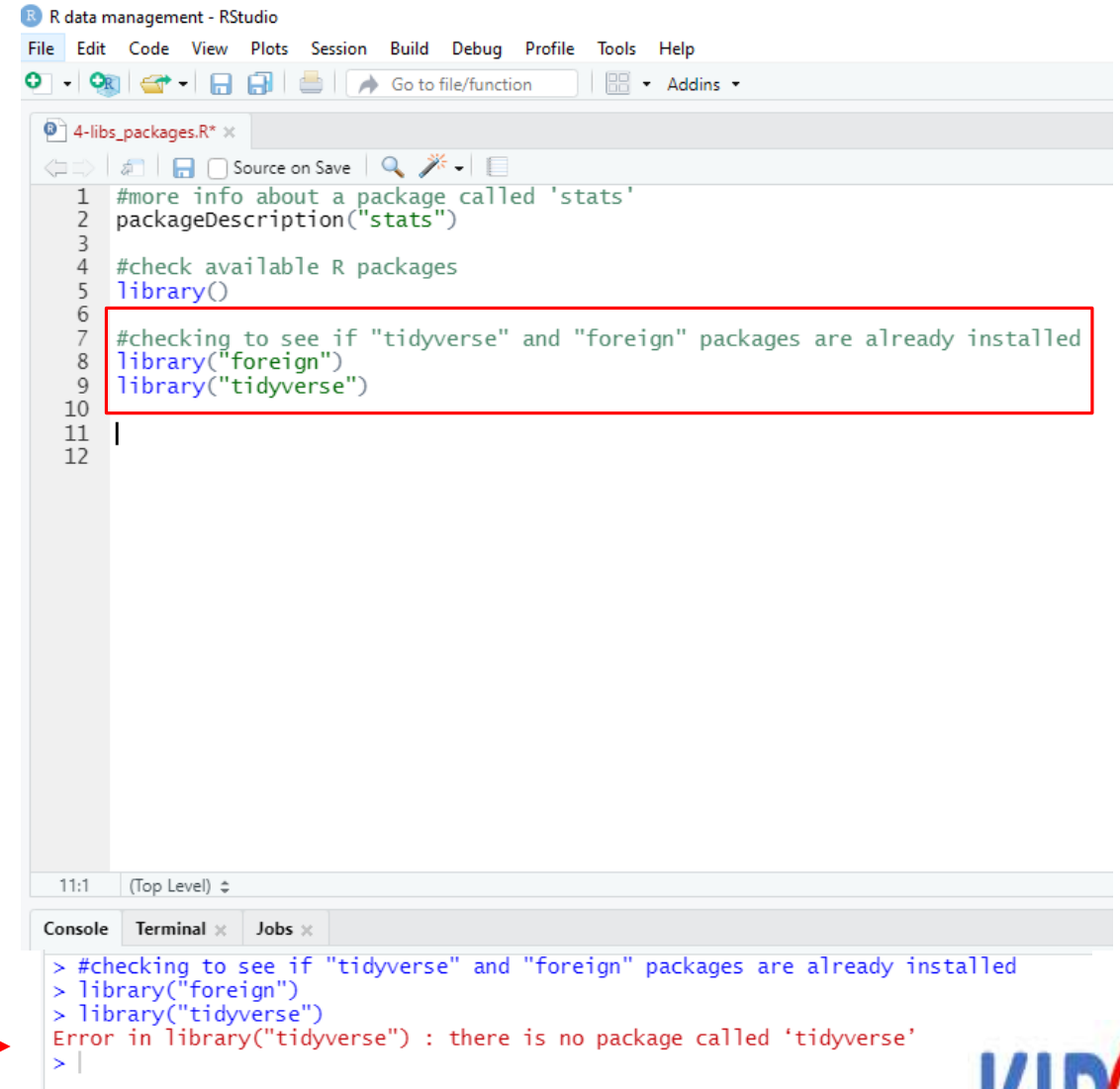
- What if you want to install a new package?
- There are three main “repositories” where you can obtain packages.
 - A repository is a location where you can find packages and install them. The most common repositories are the following:
 - ❖ CRAN: <https://cran.r-project.org/> this is the official repository and is maintained by the R community worldwide
 - ❖ Bioconductor: <https://www.bioconductor.org/> open source software for bioinformatics
 - ❖ Github: <https://github.com/> not R specific but most popular repository for open source projects

How to check if package is already installed?

- Packages are constantly being developed.
- Perhaps you may have already downloaded a package and forgot if you have it or not.
- You've heard of two cool packages called "*tidyverse*" and "*foreign*" but aren't sure if you already have them or not.
- Let's check and see.

Type "library"
and then the
name of the
package

Notice you do not
receive an error
message when you
type foreign but when
you search for tidy
verse, you receive an
error message



The screenshot shows the RStudio interface with a script editor and a console. The script editor contains the following R code:

```
1 #more info about a package called 'stats'
2 packageDescription("stats")
3
4 #check available R packages
5 library()
6
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 |
12
```

The code on lines 8 and 9 is highlighted with a red box. The console at the bottom shows the output of these commands:

```
> #checking to see if "tidyverse" and "foreign" packages are already installed
> library("foreign")
> library("tidyverse")
Error in library("tidyverse") : there is no package called 'tidyverse'
> |
```

A red arrow points from the text "you receive an error message" to the error message in the console.

How to install a new package?

➤ We will show you how to install a package in R by downloading a commonly used package known as “tidyverse”

- Please note that tidyverse is a collection of R packages
- Tidyverse is a powerful collection of R packages for data management



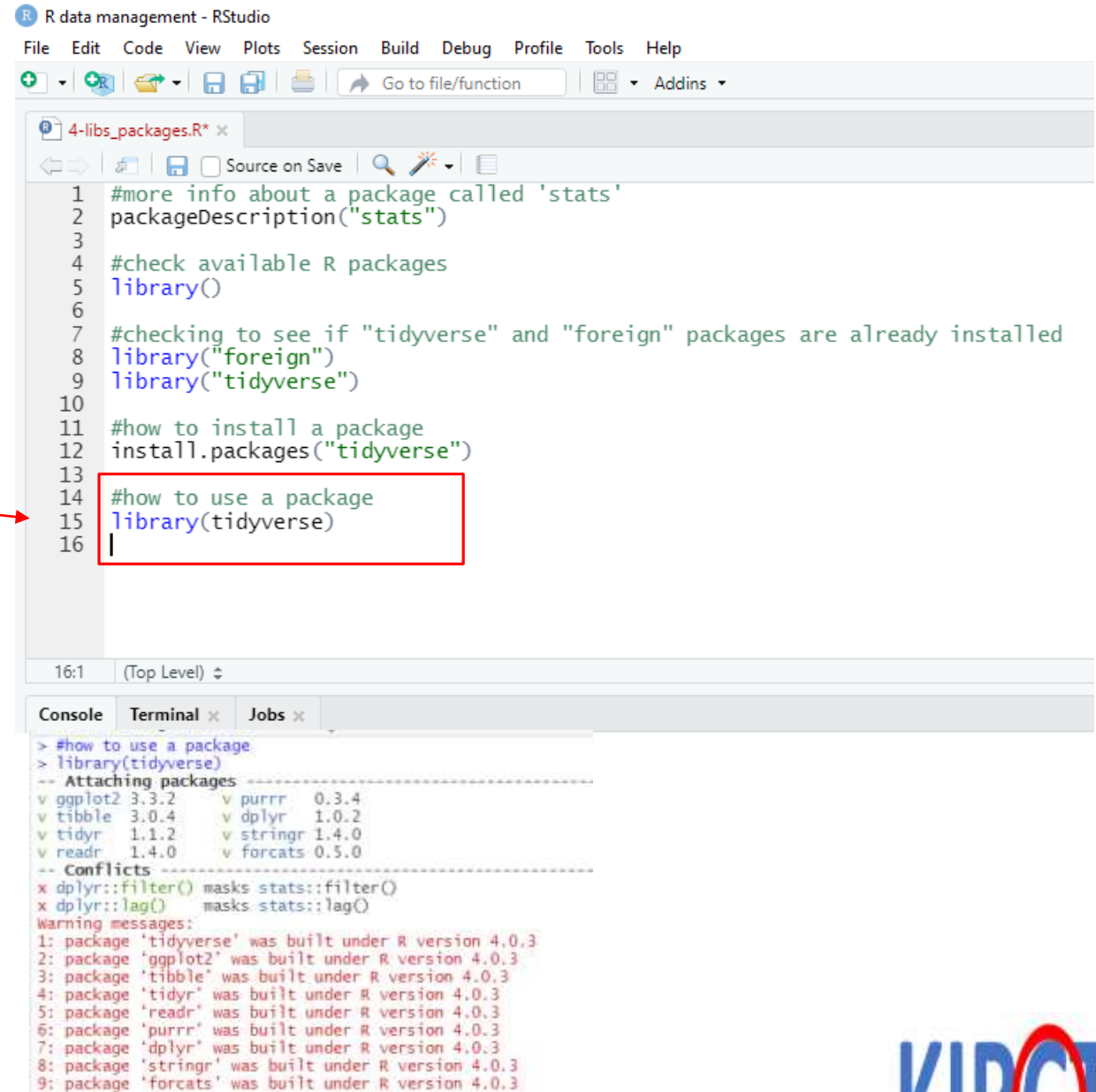
Using a package in R

You will not be able to use the package until you load it with the `library()` function.

Each time you want to use the package in your Rscript, you have to use the `library()` function.

This is telling you that the tidyverse package has also attached these packages (eg., `ggplot2`)

The warning messages will not affect the use of the package.



The screenshot shows the RStudio interface. The script editor displays the following code:

```
1 #more info about a package called 'stats'
2 packageDescription("stats")
3
4 #check available R packages
5 library()
6
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
```

The console output shows the execution of `library(tidyverse)`:

```
> #how to use a package
> library(tidyverse)
-- Attaching packages -----
v ggplot2 3.3.2      v purrr  0.3.4
v tibble  3.0.4      v dplyr  1.0.2
v tidyr   1.1.2      v stringr 1.4.0
v readr   1.4.0      v forcats 0.5.0
-- Conflicts -----
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Warning messages:
1: package 'tidyverse' was built under R version 4.0.3
2: package 'ggplot2' was built under R version 4.0.3
3: package 'tibble' was built under R version 4.0.3
4: package 'tidyr' was built under R version 4.0.3
5: package 'readr' was built under R version 4.0.3
6: package 'purrr' was built under R version 4.0.3
7: package 'dplyr' was built under R version 4.0.3
8: package 'stringr' was built under R version 4.0.3
9: package 'forcats' was built under R version 4.0.3
```

Red arrows point from the text on the left to the `library(tidyverse)` line in the script and the console output.

➤ What is the primary purpose of an R package?

- Download additional software to RStudio.

-  Add new functions and data to R for specific tasks.

- Change the appearance of the RStudio interface.

- Store personal R scripts and settings.

-
- Which of the following statements is false about R packages?
- ❑ Packages can be installed from online repositories like CRAN.
 - ❑ Base R functions and package functions can coexist in the same R session.
 - ❑ All packages are created and maintained by the R core team.
 - ❑ You can use the `library()` function to load a package into your R session.

➤ Which package is the most widely used for data manipulation and analysis in R?

a) ggplot2

 b) dplyr

c) readr

d) stats

➤ What are the advantages of using nested packages compared to individual packages?

a) Reduced total number of package installs.

 b) Simplified organization and sharing of related functions.

c) Increased compatibility with older versions of R.

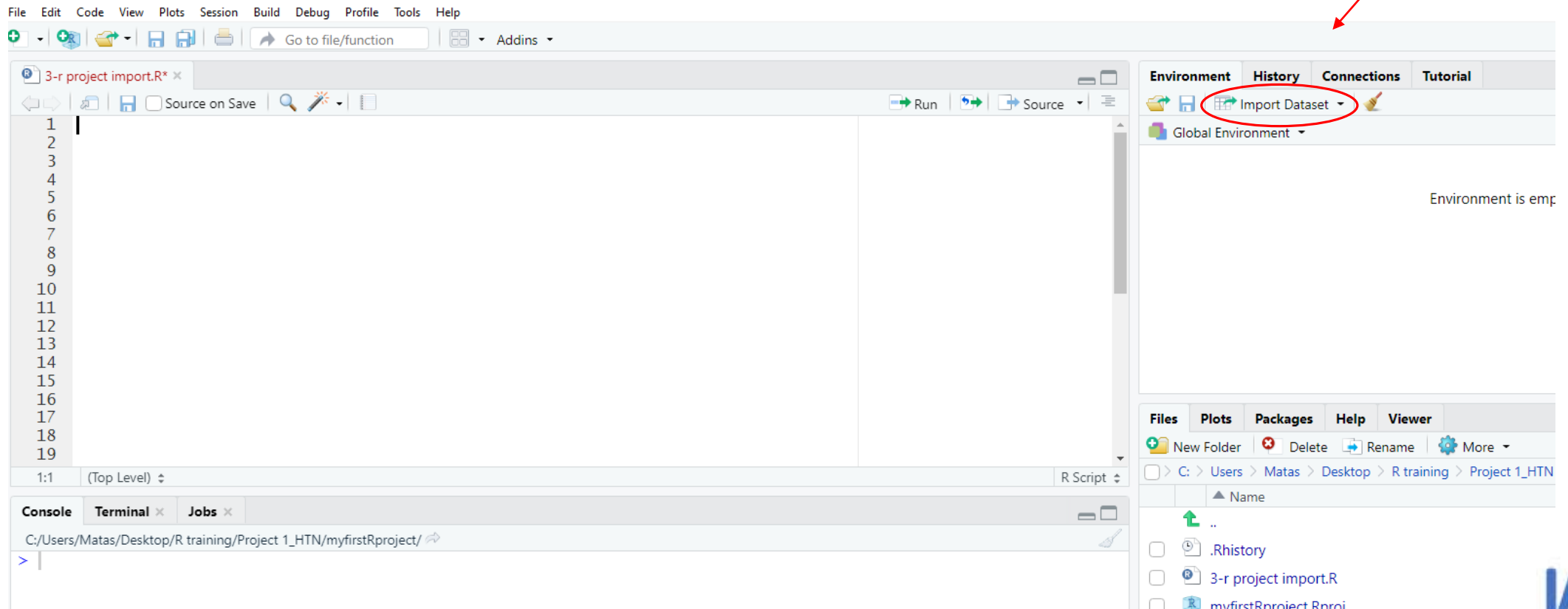
d) Easier access to functions from multiple independent packages.

Load the previous pat_info dataset

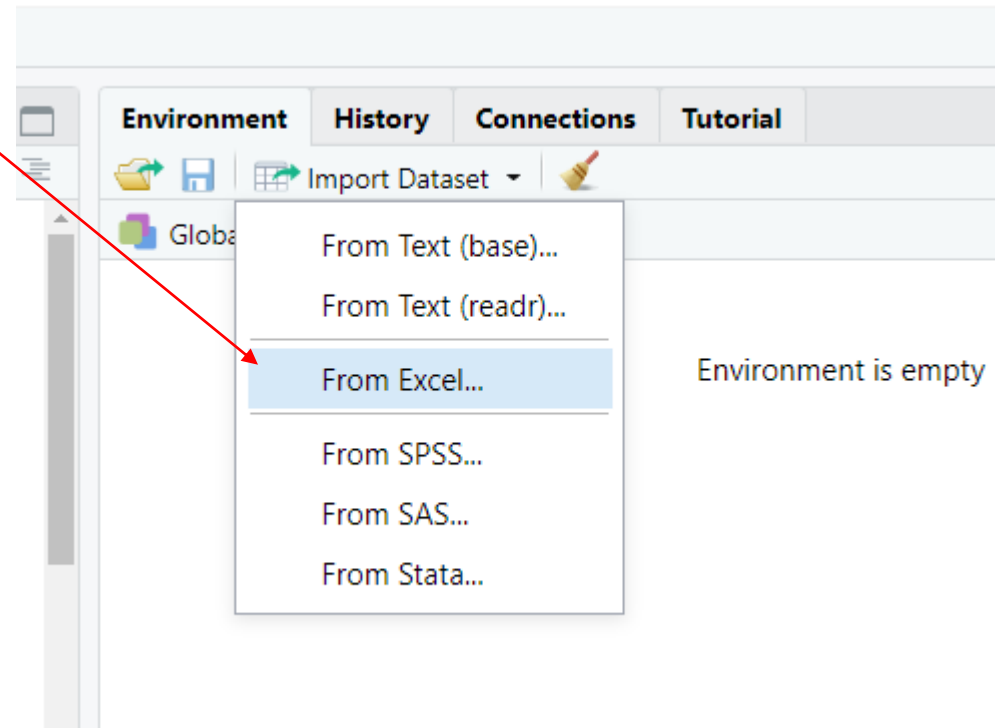
THE NEXT SLIDES ARE A REVIEW OF HOW TO IMPORT AN EXCEL SPREADSHEET (SAME SLIDES FROM MODULE 03)

How to upload an excel file in R

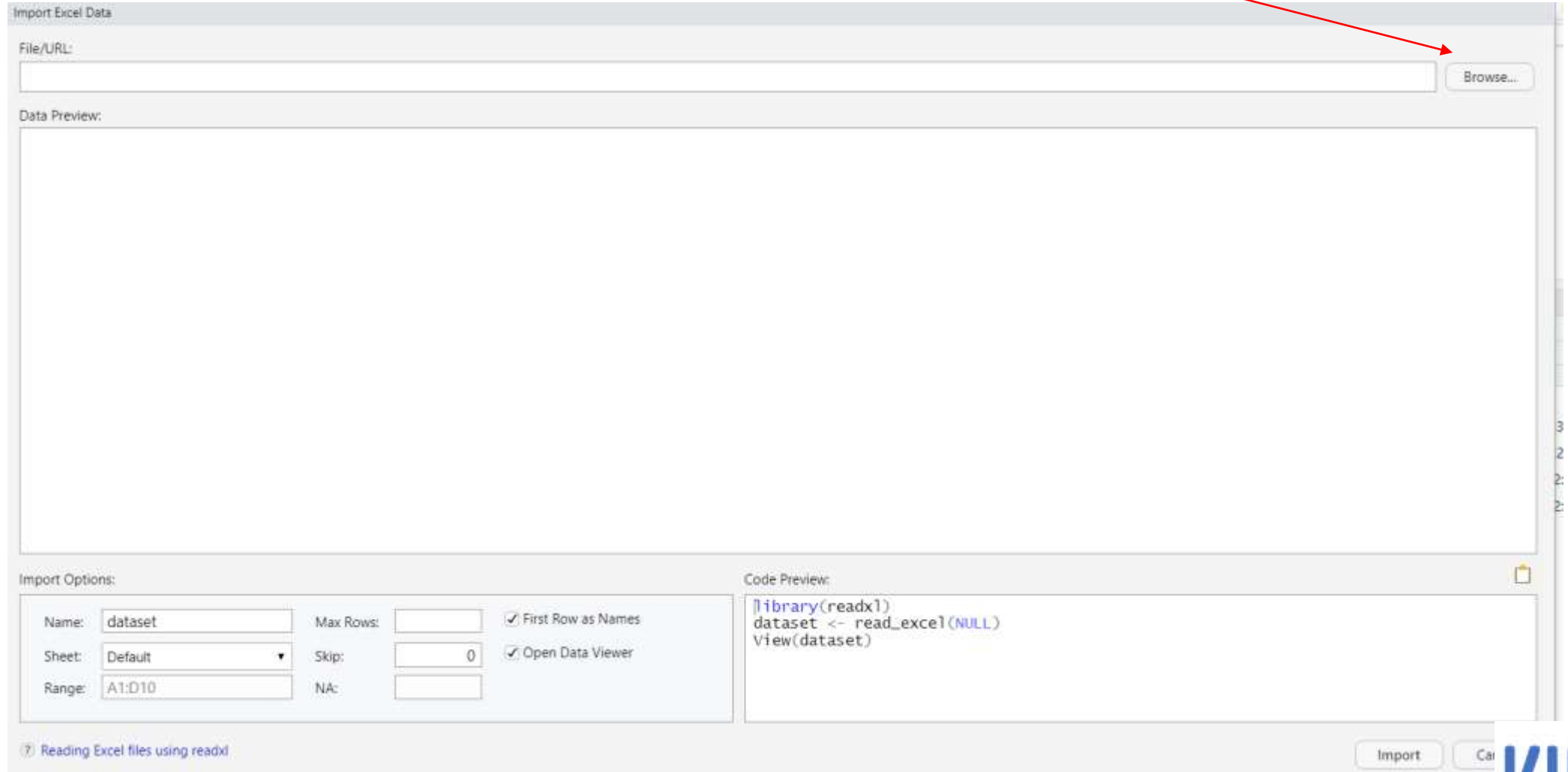
The easiest way is to click “import dataset”



Select "from Excel"



Select “browse” and search for your excel file. Make sure it’s saved on your computer!



Import Excel Data

File/URL:

Data Preview:

Import Options:

Name: <input type="text" value="dataset"/>	Max Rows: <input type="text"/>	<input checked="" type="checkbox"/> First Row as Names
Sheet: <input type="text" value="Default"/>	Skip: <input type="text" value="0"/>	<input checked="" type="checkbox"/> Open Data Viewer
Range: <input type="text" value="A1:D10"/>	NA: <input type="text"/>	

Code Preview:

```
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
```

[\(?\) Reading Excel files using readxl](#)

Once you select your excel file, you will see the preview of the data. Next, select “import”.

Import Excel Data

File/URL:
C:/Users/User/Box/_Courses/R Class/R data management/Data sets/pat_info.xlsx Update

Data Preview:

ID <small>(double)</small>	Age <small>(double)</small>	Sex <small>(character)</small>	HTN_Med <small>(double)</small>	Race <small>(double)</small>
1	50	M	0	1
2	67	M	1	3
3	75	F	1	2
4	31	F	0	1
5	29	F	0	1
6	74	M	0	4
7	58	F	1	2
8	41	M	0	3
9	86	M	1	4
10	22	M	1	1

Previewing first 50 entries.

Import Options:

Name: Max Rows:
Sheet: Skip:
Range: NA:

First Row as Names
 Open Data Viewer

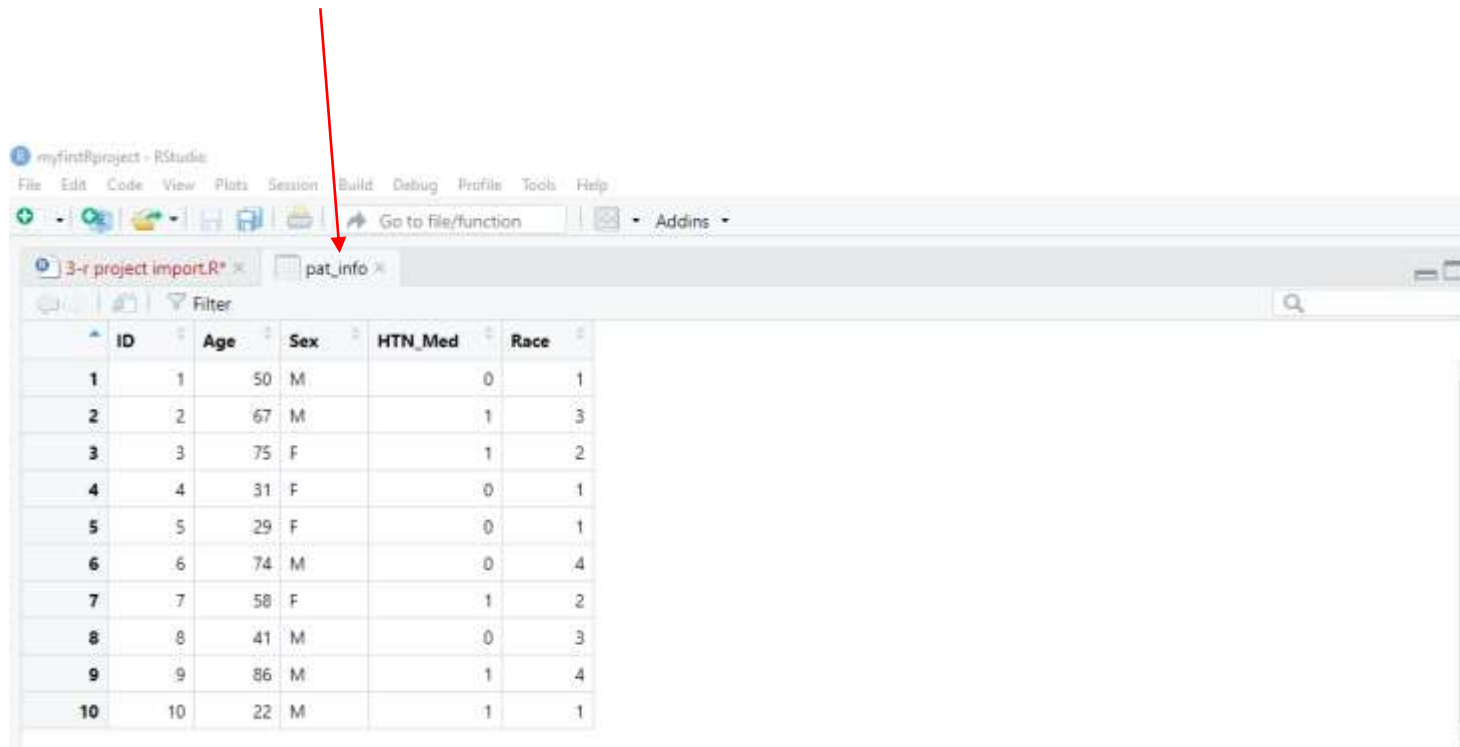
Code Preview:

```
library(readxl)
pat_info <- read_excel("C:/Users/Matas/Box/_Courses/R Class/R data management/Data
sets/pat_info.xlsx")
View(pat_info)
```

Import

Reading Excel files using readxl

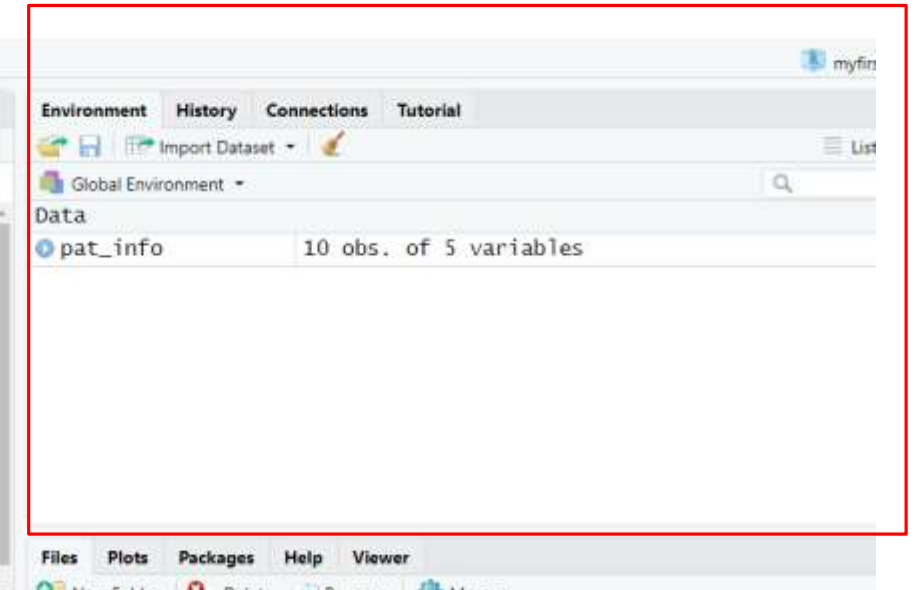
You see a new tab that has opened your dataset in R.



The screenshot shows the RStudio interface with a new tab titled 'pat_info' open. A red arrow points to this tab. The main window displays a data table with the following columns: ID, Age, Sex, HTN_Med, and Race. The data is as follows:

ID	Age	Sex	HTN_Med	Race
1	50	M	0	1
2	67	M	1	3
3	75	F	1	2
4	31	F	0	1
5	29	F	0	1
6	74	M	0	4
7	58	F	1	2
8	41	M	0	3
9	86	M	1	4
10	22	M	1	1

The dataset is officially in the R environment as shown here. Name of dataset is "pat_info"



The screenshot shows the RStudio Environment pane. A red arrow points to the 'pat_info' dataset listed under the 'Data' section. The dataset is described as having 10 observations and 5 variables.

Environment	History	Connections	Tutorial
Global Environment			
Data			
pat_info			

Load the previous df_example dataset

Using dplyr package under tidyverse

Dplyr package

Dplyr: overview

- The dplyr package helps with data manipulation by providing a set of verbs that help with the most common data manipulation tasks:
 - filter(): picks observations based on certain values
 - arrange(): arranges the ordering of the rows
 - select(): picks variables
 - mutate(): adds new variables
 - summarise(): reduces values to a single summary

- Before teaching you how to use these functions, we have to discuss “piping” in R.

“Piping”

- The `dyplr` package offers the “piping operator” which looks like this: `%>%`
- You can think of the piping operator to mean “and then”. For instance, the result from one step is then “piped” into the next step.

- For example

```
filter(dataset, values=males) %>%  
  group_by(clinic) %>%  
  summarise (newvar=mean(continuous_variable))
```

In plain English, this would translate to, please look at my dataset called “dataset”, filter by male values *AND THEN* group by clinic *AND THEN* summarize a new variable I will create based on the mean of a continuous variable. It may not make intuitive sense initially but we will practice with some examples.

filter() function

Let's start simple. Let's filter observations to patients above the age of 60.

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
4-libs_packages.R* x
Source on Save
1 #more info about a package called 'stats'
2 packageDescription("stats")
3
4 #check available R packages
5 library()
6
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19
20
25:1 (Top Level) ↓
```

Option 1:
Filter(name of dataset,
condition)

You can quickly see the results down here. Easy if you have a small dataset. What if you want to save it by making a new subset?

```
Console Terminal x Jobs x
C:/Users/Matas/Box/_Courses/R Class/R data management/
> #using filter() function
> filter(pat_info, Age>60)
# A tibble: 4 x 5
  ID Age Sex HTN_Med Race
  <dbl> <dbl> <chr> <dbl> <dbl>
1 2 67 M 1 3
2 3 75 F 1 2
3 6 74 M 0 4
4 9 86 M 1 4
```

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
4-libs_packages.R* x above60 x
Source on Save
1 #more info about a package called 'stats'
2 packageDescription("stats")
3
4 #check available R packages
5 library()
6
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19 above60 <-filter(pat_info, Age>60)
20 view(above60)
```

Option 2:
Make a dataset, we are going to call it "above60" and then apply the filter function

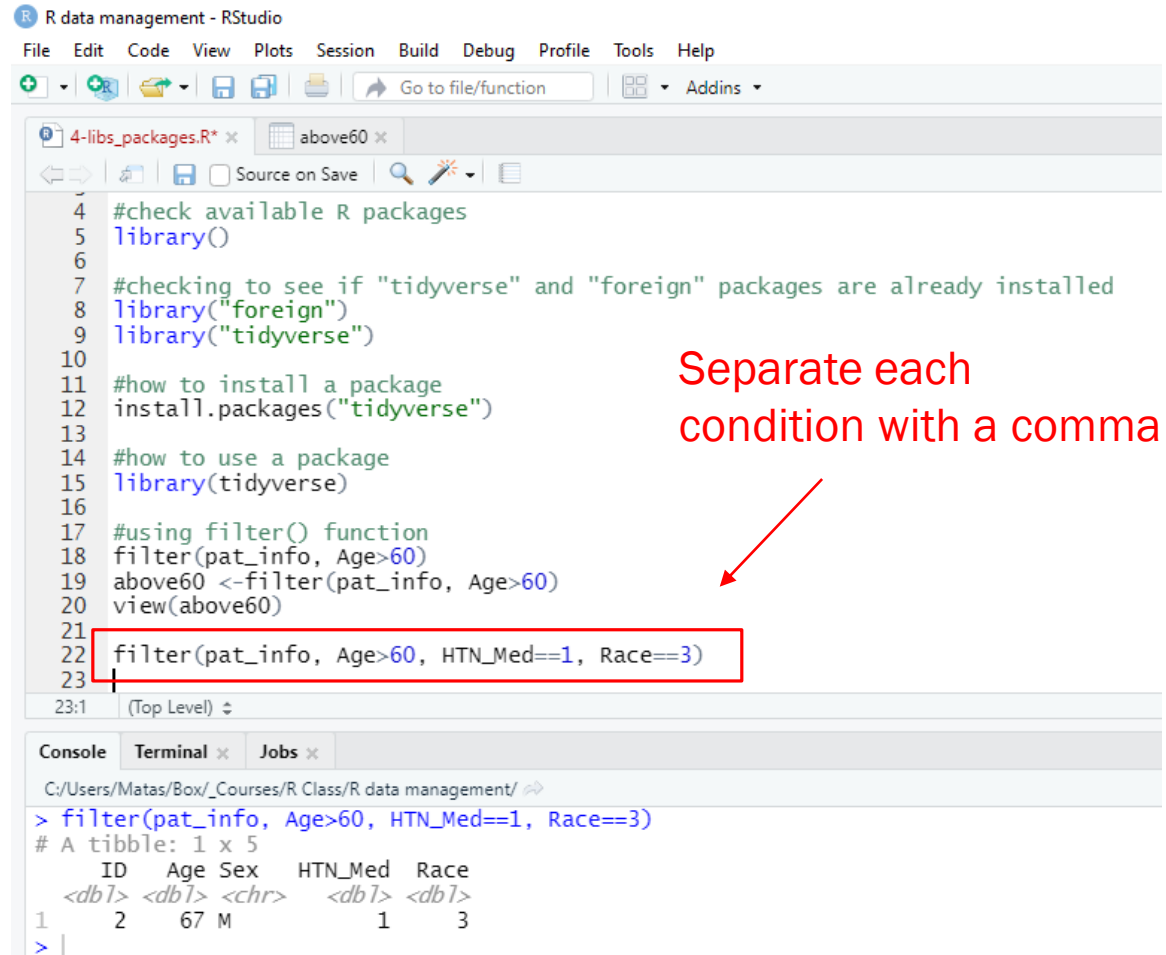
```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
4-libs_packages.R* x above60 x
Filter
ID Age Sex HTN_Med Race
1 2 67 M 1 3
2 3 75 F 1 2
3 6 74 M 0 4
4 9 86 M 1 4
```



filter() function using the dplyr package under the tidyverse package

➤ Suppose you are interested in identifying patients under the following conditions:

- Age>60
- Indicator for hypertension medication
- Race=3 (Hispanic)



```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
4-libs_packages.R* x above60 x
Source on Save
4 #check available R packages
5 library()
6
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19 above60 <-filter(pat_info, Age>60)
20 view(above60)
21
22 filter(pat_info, Age>60, HTN_Med==1, Race==3)
23
23:1 (Top Level) ↕
Console Terminal x Jobs x
C:/Users/Matas/Box/_Courses/R Class/R data management/ ↗
> filter(pat_info, Age>60, HTN_Med==1, Race==3)
# A tibble: 1 x 5
  ID Age Sex HTN_Med Race
<dbl> <dbl> <chr> <dbl> <dbl>
1 2 67 M 1 3
> |
```

Separate each condition with a comma



Filter variables

- The `filter()` function chooses rows that meet a specific criteria.
- Suppose that we want to filter the data table to only show the entries for which `ANC_HF` is “Yes”
- To do this we use the `filter()`
- The filter function take the data frame as the first argument and the conditional statement as the next.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

4. Examples.R x Untitled1* x

```
1 # Date: october, 2023
2 # Title:
3 # Programmer:
4
5
6
7
8 filter(df_EX, ANC_HF == "Yes")
9 |
```

9:1 (Top Level)

Console Terminal x Background Jobs x

R 4.3.1 · c:/users/user/desktop/mpd/

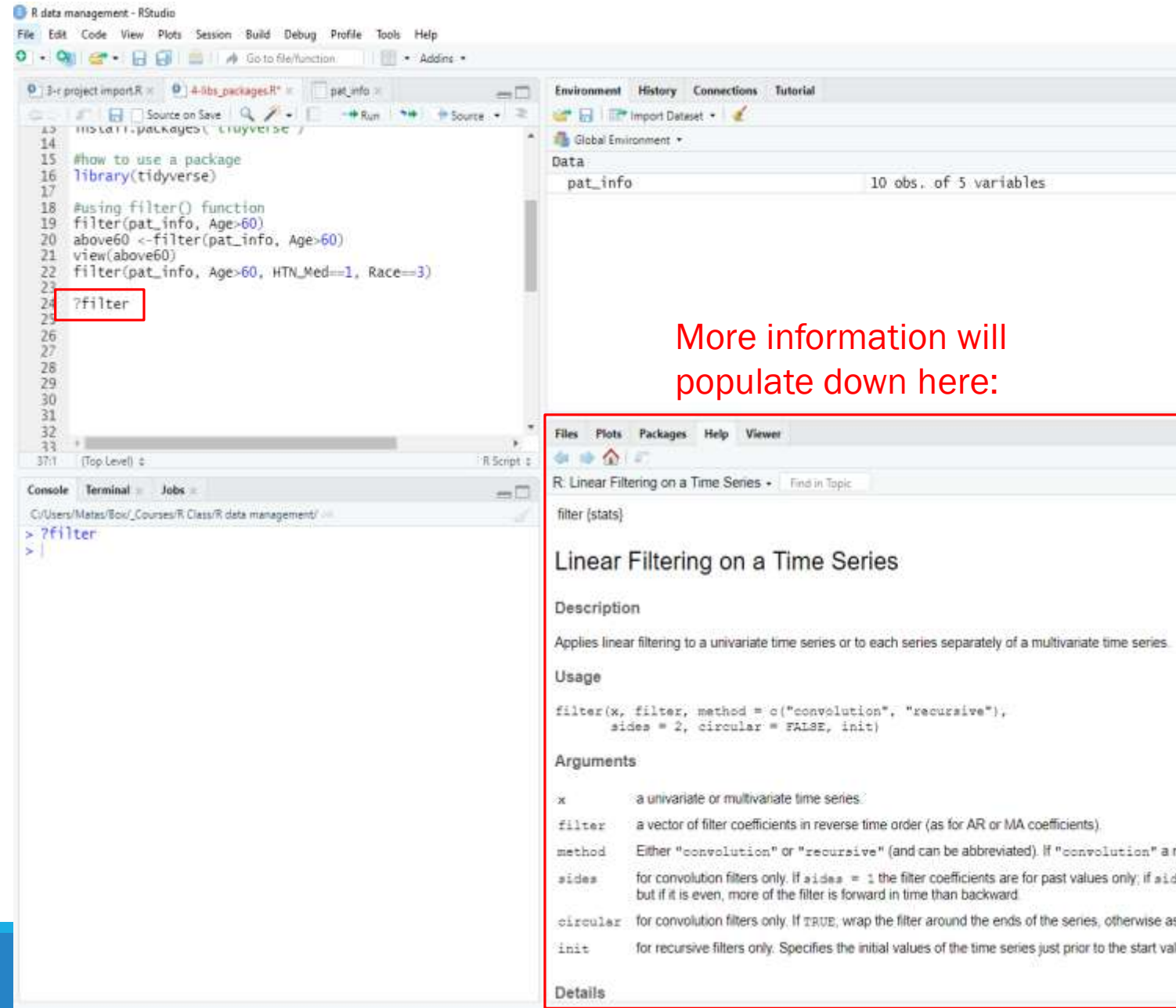
```
> # Date: october, 2023
> # Title:
> # Programmer:
>
>
>
>
> filter(df_EX, ANC_HF == "Yes")
  New_HF_ID New_ID child_ID order_child EmergencyCS ANC_HF Age Age_cat Educ_cat Parity
1         2   5886  5886-1         1           No    Yes   25 20-35 yrs      <NA>      0
2         2   5887  5887-1         1           No    Yes   32 20-35 yrs      <NA>      3
3         2   5914  5914-1         1           Yes    Yes   31 20-35 yrs      <NA>      2
4         2   5968  5968-1         1           No    Yes   24 20-35 yrs No education  1
5         2   5970  5970-1         1           Yes    Yes   33 20-35 yrs      <NA>      0
6         2   5976  5976-1         1           No    Yes   31 20-35 yrs      <NA>      2
7         2   5977  5977-1         1           No    Yes   30 20-35 yrs  Secondary  1
8         2   5979  5979-1         1           No    Yes   35 20-35 yrs Post-secondary 0
9         2   5980  5980-1         1           No    Yes   23 20-35 yrs      <NA>      0
10        2   5981  5981-1         1           No    Yes   30 20-35 yrs      <NA>      5
```

From the data frame "df_Ex" we use the filter () to select on ANC observations that are "Yes"

Output with only ANC_HF that are "Yes"

If you don't know what a function does...

You can type
?**nameoffunction**
eg.,
?**filter**
and you will receive
more information
regarding the function
in the bottom right
window.



The screenshot shows the RStudio interface. The main editor window contains the following R code:

```
13 #install.packages("tidyverse")
14
15 #how to use a package
16 library(tidyverse)
17
18 #using filter() function
19 filter(pat_info, Age>60)
20 above60 <- filter(pat_info, Age>60)
21 view(above60)
22 filter(pat_info, Age>60, HTN_Med==1, Race==3)
23
24 ?filter
25
26
27
28
29
30
31
32
33
```

The Environment pane on the right shows a data object named 'pat_info' with 10 observations and 5 variables.

The Console window shows the command `> ?filter` being entered.

The help window on the bottom right displays the documentation for the `filter` function from the `stats` package, titled "Linear Filtering on a Time Series".

filter (stats)

Linear Filtering on a Time Series

Description

Applies linear filtering to a univariate time series or to each series separately of a multivariate time series.

Usage

```
filter(x, filter, method = c("convolution", "recursive"),
       sides = 2, circular = FALSE, init)
```

Arguments

- `x`: a univariate or multivariate time series.
- `filter`: a vector of filter coefficients in reverse time order (as for AR or MA coefficients).
- `method`: Either "convolution" or "recursive" (and can be abbreviated). If "convolution" a
- `sides`: for convolution filters only. If `sides = 1` the filter coefficients are for past values only; if `sides = 2` but if it is even, more of the filter is forward in time than backward.
- `circular`: for convolution filters only. If `TRUE`, wrap the filter around the ends of the series, otherwise as
- `init`: for recursive filters only. Specifies the initial values of the time series just prior to the start val

Details

More information will
populate down here:

A large, light blue, semi-transparent R logo is positioned on the left side of the slide, extending from the top to the bottom. It features a green circle at the top left and a white 'R' inside a blue oval.

Arrange function

- **Arrange ()** orders the rows of a data frame by the values of selected columns.

The **arrange()** function takes two arguments:

- The data frame to be reordered.
- A column names, in the order that the rows should be reordered by

The **arrange()** function will sort the rows of the data frame in ascending order by default, but you can specify descending order by using the **desc()** function.

arrange() function

Suppose you want to arrange your data in ascending and then descending order by age.

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
4-libs_packages.R
7 #checking to see if "tidyverse" and "foreign" packages are already installed
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19 above60 <-filter(pat_info, Age>60)
20 view(above60)
21 filter(pat_info, Age>60, HTN_Med==1, Race==3)
22
23 #arrange
24 arrange(pat_info, Age)
25
26
28:1 (Top Level)
Console Terminal Jobs
C:/Users/Matas/Box/_Courses/R Class/R data management/
> #arrange
> arrange(pat_info, Age)
# A tibble: 10 x 5
  ID Age Sex HTN_Med Race
  <dbl> <dbl> <chr> <dbl> <dbl>
1 10 22 M 1 1
2 5 29 F 0 1
3 4 31 F 0 1
4 8 41 M 0 3
5 1 50 M 0 1
6 7 58 F 1 2
7 2 67 M 1 3
8 6 74 M 0 4
9 3 75 F 1 2
10 9 86 M 1 4
```

You can re-arrange by age in ascending order (default is ascending)

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
4-libs_packages.R*
8 library("foreign")
9 library("tidyverse")
10
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19 above60 <-filter(pat_info, Age>60)
20 view(above60)
21 filter(pat_info, Age>60, HTN_Med==1, Race==3)
22
23 #arrange
24 arrange(pat_info, Age)
25 arrange(pat_info, desc(Age))
26
27
26:1 (Top Level)
Console Terminal Jobs
C:/Users/Matas/Box/_Courses/R Class/R data management/
> arrange(pat_info, desc(Age))
# A tibble: 10 x 5
  ID Age Sex HTN_Med Race
  <dbl> <dbl> <chr> <dbl> <dbl>
1 9 86 M 1 4
2 3 75 F 1 2
3 6 74 M 0 4
4 2 67 M 1 3
5 7 58 F 1 2
6 1 50 M 0 1
7 8 41 M 0 3
8 4 31 F 0 1
9 5 29 F 0 1
10 10 22 M 1 1
```

If you include the "desc" option you can arrange by descending order by age





Classwork;

Use the `df_example` dataset

- Arrange the variable `Age` in ascending order
- Arrange the variable `Age` in descending order

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ + + + + Go to file/function Addins
4. Examples.R* x df_EX_Child x df_EX x Untitled1* x
Source on Save Run
1 # Date: october, 2023
2 # Title:
3 # Programmer
4
5
6
7 arrange(df_EX, Age)
8
9 arrange(df_EX, desc(Age))
10
11
12 |
13
```

Arrange the variable Age in ascending order



Arrange the variable Age in ascending order



Age arranged in ascending order



Console Terminal x Background Jobs x

R 4.3.1 · c/users/user/desktop/mpd/

```
> arrange(df_EX, Age)
```

	New_HF_ID	New_ID	Child_ID	Order_Child	EmergencyCS	ANC_HF	Age	Age_cat
1	5	15506	15506-1	1	No	Yes	9	9-19 yrs
2	25	105037	105037-1	1	Yes	No	14	9-19 yrs
3	2	7060	7060-1	1	Yes	Yes	15	9-19 yrs
4	3	11673	11673-1	1	No	No	15	9-19 yrs
5	5	16016	16016-1	1	Yes	Yes	15	9-19 yrs
6	5	16126	16126-1	1	No	No	15	9-19 yrs
7	5	18569	18569-1	1	Yes	No	15	9-19 yrs
8	2	5892	5892-1	1	No	<NA>	16	9-19 yrs
9	2	7614	7614-1	1	No	<NA>	16	9-19 yrs
10	2	7621	7621-1	1	No	Yes	16	9-19 yrs
11	3	9893	9893-1	1	No	No	16	9-19 yrs
12	3	10376	10376-1	1	Yes	No	16	9-19 yrs
13	3	11360	11360-2	2	No	Yes	16	9-19 yrs
14	4	13712	13712-1	1	No	Yes	16	9-19 yrs
15	4	13929	13929-1	1	No	Yes	16	9-19 yrs
16	5	16424	16424-2	2	Yes	No	16	9-19 yrs



Select variables

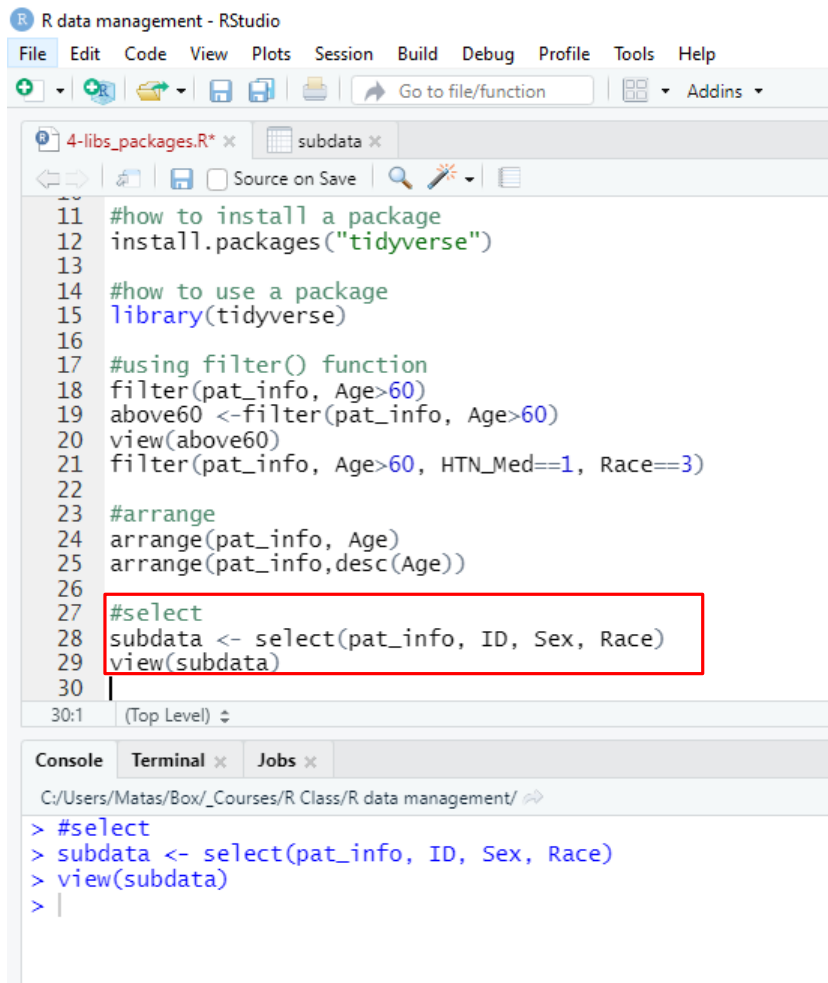
The `select()` function in R is used to select specific columns from a data frame that we want to work with.

The `select()` function takes two main arguments:

- `select()` takes a data frame as its first argument.
- The subsequent arguments specify the columns to keep:
 - You can use column names directly.
 - You can use numerical indices to refer to column positions (e.g. 4, 5, 6).
- The result is stored in a new data frame (`selected_df`) containing the chosen columns.

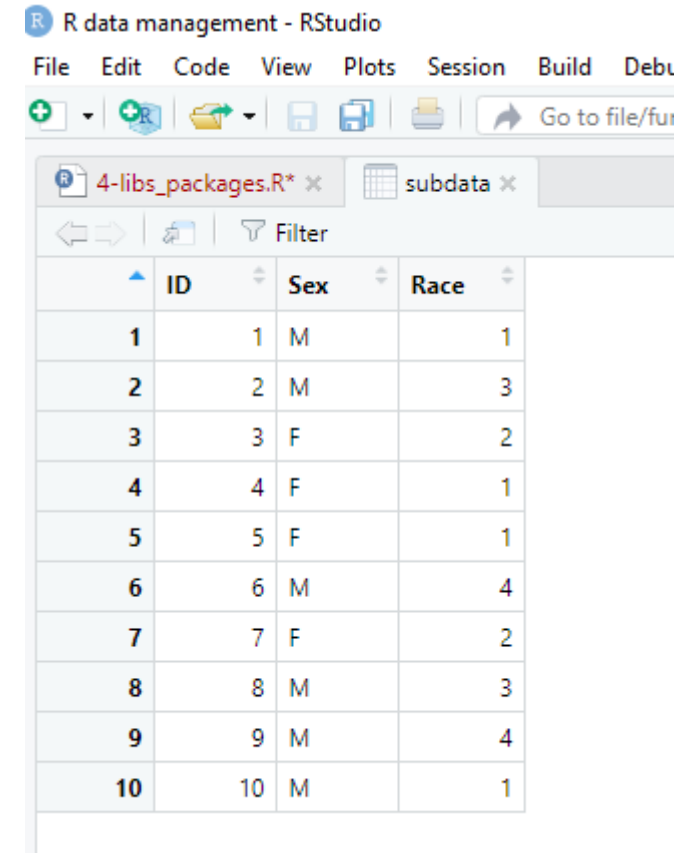
select() function

You may work with large datasets with 100+ variables. However, you may be interested in only 20 variables. You can use the select function to “select” the variables/columns you want.



```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
4-libs_packages.R* x subdata x
Source on Save
11 #how to install a package
12 install.packages("tidyverse")
13
14 #how to use a package
15 library(tidyverse)
16
17 #using filter() function
18 filter(pat_info, Age>60)
19 above60 <-filter(pat_info, Age>60)
20 view(above60)
21 filter(pat_info, Age>60, HTN_Med==1, Race==3)
22
23 #arrange
24 arrange(pat_info, Age)
25 arrange(pat_info,desc(Age))
26
27 #select
28 subdata <- select(pat_info, ID, Sex, Race)
29 view(subdata)
30
30:1 (Top Level)
Console Terminal x Jobs x
C:/Users/Matas/Box/_Courses/R Class/R data management/
> #select
> subdata <- select(pat_info, ID, Sex, Race)
> view(subdata)
> |
```

Here, we are creating a smaller dataset called “subdata” and selecting only the following variables: ID, Sex, and Race



	ID	Sex	Race
1	1	M	1
2	2	M	3
3	3	F	2
4	4	F	1
5	5	F	1
6	6	M	4
7	7	F	2
8	8	M	3
9	9	M	4
10	10	M	1



Select variables

- Here's a question leading to the answer on using `select()` in R:
- You have a data frame named `df_example` containing several variables.
- However, you're only interested in working with four specific variables: "4", "5", "6", and "Parity".
- How can you efficiently select these four columns from `df_example` and create a new data frame containing only these variables for further analysis?

-
- `select()` takes the data frame (`df_example`) as its first argument.
 - The subsequent arguments specify the columns to keep:
 - You can use column names directly (Parity).
 - You can use numerical indices to refer to column positions (4, 5, 6).
 - The result is stored in a new data frame (`selected_df`) containing only the chosen columns.

We use the `select()` to select the variables 4,5,6 and parity

`print()` will print the output in the console

Console showing the selected variables

The screenshot shows the RStudio interface. The editor window contains the following R code:

```
1 # Date: october, 2023
2 # Title:
3 # Programmer:
4 #
5 #
6 #
7 #
8 exampl_1 <- df_EX %>%
9   select(4:6, Parity) %>%
10  print()
11 #
12 #
```

The console window shows the execution of the code, with the output of the `print()` function:

```
> # Date: october, 2023
> # Title:
> # Programmer:
> #
> #
> #
> #
> exampl_1 <- df_EX %>%
+   select(4:6, Parity) %>%
+   print()
  Order_Child EmergencyCS ANC_HF Parity
1             1          NO  <NA>     2
2             1          NO  <NA>     1
3             1          NO  <NA>     1
4             1          NO  <NA>     0
5             1          NO   Yes     0
6             1          NO   Yes     3
```

-
- You have a data frame named `df_example` containing several variables.
 - However, you're only interested in working with four specific variables: "4", "5", "6", "Parity", `Age_cat`, and `ANC_HF`.
 - How can you efficiently select these four columns from `df_example` and create a new data frame containing only these variables for Participants that respond YES to the variable `ANC_HF` (**attended ANC at the health facility**)

```
1 # Date: october, 2023
2 # Title:
3 # Programmer:
4 #
5 #
6 #
7 #
8 df_EX %>%
9   select(4:6, Parity, ANC_HF, Age_cat) %>%
10  filter(ANC_HF=="Yes") %>%
11  print()
12 #
13 #
```

```
R 4.3.1 . c:/users/user/desktop/mpd/
> # Date: october, 2023
> # Title:
> # Programmer:
> #
> #
> #
> #
> df_EX %>%
+   select(4:6, Parity, ANC_HF, Age_cat) %>%
+   filter(ANC_HF=="Yes") %>%
+   print()
  Order_Child EmergencyCS ANC_HF Parity Age_cat
1           1           No   Yes     0 20-35 yrs
2           1           No   Yes     3 20-35 yrs
3           1           Yes   Yes     2 20-35 yrs
4           1           No   Yes     1 20-35 yrs
5           1           Yes   Yes     0 20-35 yrs
6           1           No   Yes     2 20-35 yrs
7           1           No   Yes     1 20-35 yrs
```

We use the `select ()` to select the variables 4,5,6 and parity

The `filter ()` will give an output for observations only of patients that satisfy the answer YES to the variable ANC_HF

The output shows data only for patients who attended ANC at the health facility



Mutate function

Imagine you're decorating a room:

- `mutate()` is like adding new furniture or accessories to enhance the space.
- It lets you create new columns (furnishings) in a data frame (room), based on existing data (the room's foundation).

How it works:

- 1. Specify the data frame:** Tell `mutate()` which data frame you want to modify, like choosing the room for decorating.
- 2. Define new columns:** Name the new columns you want to add, like picking out specific furniture pieces.
- 3. Provide values:** Describe how to calculate or create the values for these new columns, using existing data from other columns in the data frame. This is like arranging the furniture and accessories in the room

mutate() function

Select () is used to select specific columns from the **df** dataset.

```
Untitled1* x
Source on Save
25
26
27
28
29 result <- df %>%
30   select(New_ID, Age, Parity) %>%
31   mutate(Age_norm = Age / mean(Age, na.rm = TRUE))
32
33 # Print the result
34 print(result)
35
36
37
38
39
40
41
42
43
44
```

```
Console Terminal x Background Jobs x
R 4.3.2 · c:/users/LENOVO/adrenals/
> result <- df %>%
+   select(New_ID, Age, Parity) %>%
+   mutate(Age_norm = Age / mean(Age, na.rm = TRUE))
> # Print the result
> print(result)
  New_ID Age Parity Age_norm
1   5882  28     2  0.9631689
2   5883  33     1  1.1351633
3   5884  34     1  1.1695622
4   5885  23     0  0.7911744
5   5886  25     0  0.8599722
6   5887  32     3  1.1007644
7   5888  37     2  1.2727589
8   5889  39     3  1.3415567
```

Mutate() is used to create a new column called "Age_norm" by dividing the "Age" column by the mean of the "Age" column (excluding missing values).

A large, faint R logo is visible in the background on the left side of the slide. It consists of a white letter 'R' inside a light blue oval, with a green circle above it.

Group_by function

- A common operation in data exploration is to first split data into groups and then compute the summaries for each group.
- the `group_by()` function (from the dplyr package) help us do this.
- The `group_by()` function takes a data frame and organize it using a specified primary variable

Code syntax

➤ `group_by (data, variables)`

Where

➤ `data`: is the data frame to be grouped

➤ `variables`: are the variables to group by



We can then summarise the data after grouping,

- **Summarise ()** applies a summarization to each group of observations separately



Summarise function

- The summarise() function in R is used to summarize a data frame into a new data frame with fewer rows and columns. (It is part of the tidyverse package)

summarise() takes two arguments:

- data: The data frame that you want to summarize.
- expr: A list of expressions that you want to use to summarize the data. Each expression should return a single value.

summarise() function

The summarise function collapse data depending on what you request

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
4-libs_packages.R x
Source on Save
21 filter(pat_info, Age>60, HTN_Med==1, Race==3)
22
23 #arrange
24 arrange(pat_info, Age)
25 arrange(pat_info, desc(Age))
26
27 #select
28 subdata <- select(pat_info, ID, Sex, Race)
29 view(subdata)
30
31 subdata2 <- select(pat_info, !Race)
32 view(subdata2)
33
34 #mutate
35 addcol <- mutate(pat_info, senior= ifelse(Age>=65, 1, 0))
36 view(addcol)
37
38 #summarise
39 summarise(pat_info, meanage = mean(Age))
40
40:1 (Top Level)
Console Terminal x Jobs x
C:/Users/Matas/Box/_Courses/R Class/R data management/
> #summarise
> summarise(pat_info, meanage = mean(Age))
# A tibble: 1 x 1
  meanage
  <dbl>
1 53.3
```

Here we are instructing R to provide a new row calculating the mean of Age which we are calling “meanage”)


Results are presented below. The average age of the sample is 53.3

summarise() function

The summarise function collapse data depending on what you request

```
R data management - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
4-libs_packages.R x
Source on Save
26
27 #select
28 subdata <- select(pat_info, ID, Sex, Race)
29 view(subdata)
30
31 subdata2 <- select(pat_info, !Race)
32 view(subdata2)
33
34 #mutate
35 addcol <- mutate(pat_info, senior= ifelse(Age>=65, 1, 0))
36 view(addcol)
37
38 #summarise
39 summarise(pat_info, meanage = mean(Age))
40
41 #combining functions with the piping operator
42 pat_info %>%
43   group_by(Race) %>%
44   summarise(meanAge= mean(Age))
45
45:1 (Top Level)
Console Terminal x Jobs x
C:/Users/Matas/Box/_Courses/R Class/R data management/
> pat_info %>%
+   group_by(Race) %>%
+   summarise(meanAge= mean(Age))
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 4 x 2
  Race meanAge
  <dbl> <dbl>
1     1     33
2     2    66.5
3     3     54
4     4     80
```

Here we are instructing R to group by the variable Race AND THEN give us the mean age of each group.

- 
-
- Scenario: suppose we want to compare the statistics for those who had Emergency CS versus those did not.
 - This can be done in R using the **group_by** function, which works together with functions like summarize to re-organise a data frame by one of the given variables
 - Functions such as **Sum** and **Mean** can be used in combination with group_by in R to calculate data as organized by variable.
 - Class work; show the Emergency CS and mean statistics of Age, from the df_example dataset using group_by and summarise function

References/helpful resources

- <https://www.datacamp.com/community/tutorials/r-packages-guide>
- https://www.tutorialspoint.com/r/r_packages.htm#:~:text=R%20packages%20are%20a%20collection,needed%20for%20some%20specific%20purpose.
- <https://www.analyticsvidhya.com/blog/2019/05/beginner-guide-tidyverse-most-powerful-collection-r-packages-data-science/>

using `group_by` and `summarise` function to get the mean age

```
Untitled1* x
Source on Save
35
36
37
38
39
40 em_cs_grp <- df %>%
41   group_by(EmergencyCS) %>%
42   summarise(mean_age = mean(Age, na.rm = TRUE)) %>%
43   print()
44
45
46
47
48
49
50
51
52
53
54
55
```

Output of the code

```
Console Terminal Background Jobs x
R 4.3.2 · c:/users/LENOVO/adrenals/
> #
> #
> em_cs_grp <- df %>%
+   group_by(EmergencyCS) %>%
+   summarise(mean_age = mean(Age, na.rm = TRUE)) %>%
+   print()
# A tibble: 3 × 2
  EmergencyCS mean_age
  <int>      <dbl>
1         0      29.0
2         1      29.4
3        NA      29.8
> |
```

