**Day 1: Introduction to R programming**

**Session 1: Introduction to Stata and Data Import**

* 1. **Installing R and RStudio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | R | Python | STATA | SPSS |
| License | Free and open-source | Free and open-source | Proprietary | Proprietary |
| User Interface | Command-line, IDEs (RStudio) | Command-line, IDEs (Jupyter, PyCharm) | Graphical user interface (GUI) | Graphical user interface (GUI) |
| Visualization | Extensive libraries (ggplot2) | Extensive libraries (matplotlib, seaborn) | Basic | Basic |
| Type | Statistical Analysis | General Purpose Language | Statistical Analysis | Statistical Analysis |
| Community Support | Large, active community | Large, active community | Smaller, specialized | Moderate |
| Cost | Free | Free | Paid | Paid |
| Learning Curve | Steep | Moderate | Relatively easy | Relatively easy |
| Flexibility | High | High | Moderate | Moderate |

|  |  |
| --- | --- |
| R (programming language) - Wikipedia | [**https://www.r-project.org/**](https://www.r-project.org/) |
| RStudio products - Rtask | [**https://posit.co/download/rstudio-desktop/**](https://posit.co/download/rstudio-desktop/) |

* 1. **Starting to work with R and RStudio**

**A screenshot of a computer program

Description automatically generated**

**Setting working environment**

|  |
| --- |
| E001-setting\_working\_directory.R |
| #checking the current working directory  getwd()  #setting the new working directory  setwd('G:/.shortcut-targets-by-id/1ASYiYxEAcgJNzEL19Ex5oWLpC3ekSV7j/ROOT/TSM/to be delivered/training delivered/R training NSO') |

**Starting to work with R**

|  |
| --- |
| E002-starting\_with\_R.R |
| #creating dummy dataset  age <- c(1,3,5,2,11,9,3,9,12,3)  weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)  #calculating mean weight  mean(weight)  #calculating standard deviation of weight  sd(weight)  #calculating correlation between age and weight  cor(age, weight)  #plotting age and weight  plot(age, weight) |

**A white background with black numbers

Description automatically generated**

**A graph with numbers and a number of objects

Description automatically generated with medium confidence**

**Getting help**

|  |
| --- |
| E003-getting\_help.R |
| #general help  help.start()  #help on a function  help('lm') #OR  ?lm  #help on a package  help(package = 'stats')  #Searches the help system for instances of the given string  help.search('correlation') #OR  ??correlation  #Examples of given function  example('lm')  #Lists all available example datasets contained in currently loaded packages  data()  #listing and exploring a particular vignette (detailed help file with tutorial and examples)  vignette()  vignette('ggplot2') |

**Package management**

|  |
| --- |
| E004-package\_management.R |
| #installing a package  install.packages('plotly')  #loading a package  library(plotly)  #printing package information  packageDescription('plotly')  help(package = 'plotly')  #using the loaded package  plot\_ly(x = 2001:2020, y = (1:20)^2, type = 'bar')  #removing a package  remove.packages('plotly') |

**Task 1:**

1. Open the general help and look at the Introduction to R section.
2. Install the vcd package.
3. Load the package and read the description of the dataset Arthritis.
4. List the available dataset.
5. Print out the Arthritis dataset.
6. Run the example that comes with the Arthritis dataset.

**#1. Open the general help and look at the Introduction to R section.**

**help.start()**

**#2. Install the vcd package.**

**install.packages("vcd")**

**#3. List the available dataset.**

**data(package = 'vcd')**

**#4. Load the package and read the description of the dataset Arthritis.**

**library(vcd)**

**help(Arthritis)**

**#5. Print out the Arthritis dataset.**

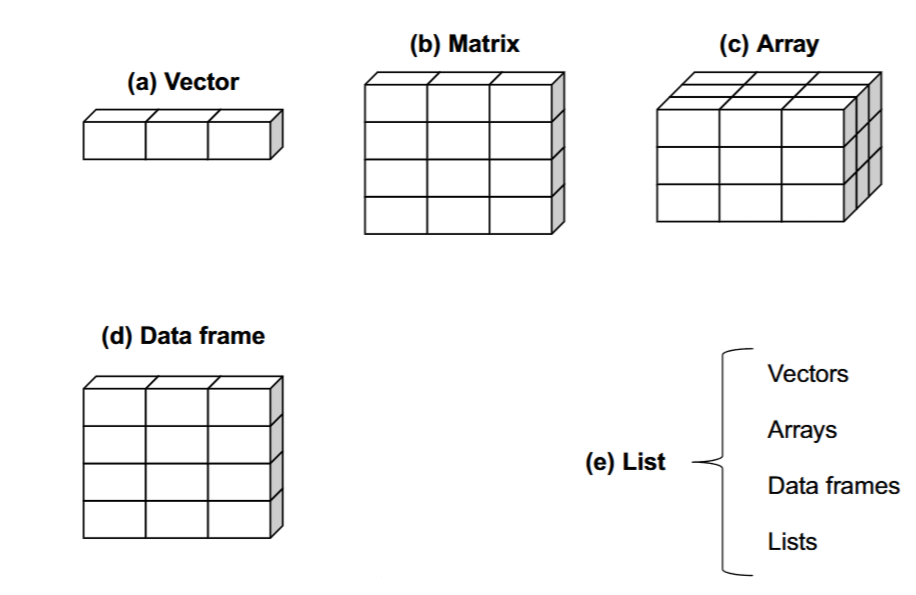
**Arthritis**

**#6. Run the example that comes with the Arthritis dataset.**

**example(Arthritis)**

**Session 2: Data objects**

* 1. **Data structures in R**

****

**Vector**

Vectors are one-dimensional arrays that can hold numeric data, character data, or logical data.

|  |
| --- |
| E005-vector.R |
| a <- c(11,22,33,44,55)  b <- c('one','two','three')  c <- c(TRUE, FALSE, FALSE)  #accessing vector elements  a[4]  b[c(1,3)]  a[3:5]  #modifying an element  a[1] <- 99  a  #removing an object  rm(a) |

**Matrix**

|  |
| --- |
| E006-matrix.R |
| x <- c(1,2,3,4,5,6,7,8)  #creating a matrix (by default byrow == FALSE)  m1 <- matrix(x, nrow = 2, ncol = 4)  m1    #creating a matrix with byrow = TRUE  m2 <- matrix(x, nrow = 2, ncol = 4, byrow = TRUE)  m2    #--------------------------  #accessing elements of a matrix  #--------------------------  m2[1,] #selecting the first row    m2[,3] #selecting the third column    m2[2,c(3,4)] #selecting the 3rd and 4th element of the 2nd row    #--------------------------  #basic matrix operations  #--------------------------  m1 + m2 #addition    m1 - m2 #subtraction    t(m1) #transpose    t(m1) %\*% m2 #matrix dot product    mm <- matrix(c(1,2,3,4), nrow = 2)  mm    det(mm) #determinant of a matrix    solve(mm) #inverse of a matrix |

**Dataframe**

A data frame is more general than a matrix in that different columns can contain different modes of data (numeric, character, and so on). It’s similar to the dataset you’d typically see in Stat.

|  |
| --- |
| E007-dataframe.R |
| #-------------------------------  #creating a dataframe  #-------------------------------  id <- c(1,2,3,4)  age <- c(25,34,28,52)  sex <- c(0,1,1,0) #0 - female, 1 - male  diabetes <- c("Type1", "Type2", "Type2", "Type1")  status <- c("Poor", "Improved", "Excellent", "Poor")  df <- data.frame(id, age, sex, diabetes, status)  df    #-------------------------------  #accessing a column  #-------------------------------  df[,2] #accessing the column values by index    df$age #accessing the column values by name    df["age"] #accessing the column by name    #-------------------------------  #accessing a multiple columns  #-------------------------------  df[,c(1,3,4)] #accessing the columns by index    df[c("id","status")] #accessing the columns by name    #-------------------------------  #creating a frequency twoway table  #-------------------------------  table(df$diabetes, df$status)    #-------------------------------  #Converting sex column to a factor (categorical) type  #-------------------------------  df$sex <- factor(df$sex, levels = c(1,0), labels = c("Male", "Female"))  df$diabetes <- factor(df$diabetes)  df    df$sex    df$diabetes    #-------------------------------  # Variable labeling  #-------------------------------  attr(df$id, "label") <- "Patient ID"  attr(df$age, "label") <- "Patient Age in years"  attr(df$sex, "label") <- "Patient sex"  View(df)    #removing a label  attr(df$sex, "label") <- NULL  View(df) |

**List**

A list is an ordered collection of R objects

|  |
| --- |
| E008-list.R |
| g <- "My First List"  h <- c(25, 26, 18, 39)  j <- matrix(1:10, nrow=5)  k <- c("one", "two", "three")  mylist <- list(title=g, ages=h, j, k)  mylist    #-------------------------------  #accessing an object  #-------------------------------  mylist$title #by object name in the list    mylist[['ages']] #by object name in the list    mylist[[1]] #by index number |

* 1. **Data input**

**Data input using keyboard (manual)**

|  |
| --- |
| E009-data\_input\_keyboard.R |
| #---------------------------------------------------------------------  # Creating a new dataframe with data input using keyboard  #---------------------------------------------------------------------  df <- edit(data.frame())    #---------------------------------------------------------------------  # Editing an existing dataframe  #---------------------------------------------------------------------  fix(df) |

**Importing data from various sources**

|  |
| --- |
| E010-data\_import.R |
| #---------------------------------------------------------------------  # Importing data from a delimited text file (e.g. csv)  #---------------------------------------------------------------------  df1 <- read.table('data/001-Arthritis.csv', header = T, sep = ',')  # OR  df2 <- read.csv('data/001-Arthritis.csv')  # Checking data structure (variable types)  str(df1)    #---------------------------------------------------------------------  # Importing data from excel file  #---------------------------------------------------------------------  library(readxl) #install the package if not installed  df3 <- read\_xlsx('data/002-excel\_data.xlsx', sheet = 'Orange')  df4 <- read\_xlsx('data/002-excel\_data.xlsx', sheet = 'infert')  #---------------------------------------------------------------------  # Importing data from SPSS and Stata  #---------------------------------------------------------------------  library(haven)  df5 <- read\_spss('data/003-mn.sav')  df6 <- read\_stata('data/004-campus.dta')  #---------------------------------------------------------------------  # Importing files directly from the web  #---------------------------------------------------------------------  df7 <- read.csv('https://people.sc.fsu.edu/~jburkardt/data/csv/biostats.csv') |

**Session 3: Data management**

* 1. **Data management using dplyr package**

**Rows operations**

|  |
| --- |
| E011-dplyr\_operation\_rows.R |
| #---------------------------------------------------------------------  # Import datasets  #---------------------------------------------------------------------  classf <- read.csv('data/005-wb\_class.csv')  energy <- read.csv('data/006-wb\_energy.csv')  var\_def <- read.csv('data/007-wb\_energy\_var\_def.csv') #variable definition  #---------------------------------------------------------------------  # Rows operation  #---------------------------------------------------------------------  # \*\*\* filter \*\*\*  library(dplyr)  data\_nepal <- filter(energy, country == 'Nepal')  data\_nepal    filter(energy, country == 'Nepal' & year > 1999)    # \*\*\* arrange \*\*\*  arrange(energy, desc(country), tot\_ele)    # \*\*\* na.omit \*\*\*  na.omit(energy)  # \*\*\* slice \*\*\* : used to choose rows using their position  slice(energy,3:7)    slice\_head(energy, n = 3)  #OR  head(energy, n = 3)    slice\_tail(energy, n = 3)  #OR  tail(energy, n = 3)    slice\_sample(energy, n = 5) #randomly selects 5 observations (rows)  slice\_sample(energy, prop = 0.01) #selects 1% sample randomly  slice\_sample(energy, prop = 0.01, replace = T) #selects 1% sample randomly with replacement |

**Columns operations**

|  |
| --- |
| E012-dplyr\_operation\_columns.R |
| #---------------------------------------------------------------------  # Columns operation  #---------------------------------------------------------------------  # \*\*\* select \*\*\*  select(energy, year, ccode, tot\_ele) #selects year, ccode, and tot\_ele    select(energy, year:ccode) #selects columns from year to ccode    select(energy, !(year:ccode)) #selects columns other than from year to ccode    head(select(energy, contains('tot'))) #selects columns with names that contains tot    head(select(energy, starts\_with('ele'))) #selects columns with names that starts with ele    head(select(energy, ends\_with('ele'))) #selects columns with names that ends with ele    # \*\*\* rename \*\*\*  head(rename(energy, year\_AD = year))    # \*\*\* mutate \*\*\*  head(mutate(energy, ren\_ele\_share = ren\_ele/tot\_ele \* 100)) |

**Piping (%>%)** : Piping is used for chaining multiple operations together in a clean way.

*Example:* Suppose you are interested in renewable electricity output data in Nepal and India. Now, you want to perform the following operations with the help of piping (%>%).

* Select columns **year, country, ren\_ele, tot\_ele** from **energy** dataframe.
* Keep data of Nepal and India only.
* Sort the dataframe according to **country** and **year** columns.
* Create a new column **ren\_ele\_share** by calculating share of renewable electricity output in total output (i.e. **ren\_ele/tot\_ele\*100**).
* Save the new dataframe as **energy\_np\_in**

|  |
| --- |
| E013-dplyr\_piping.R |
| energy\_np\_in <- energy %>%  select(year, country, ren\_ele, tot\_ele) %>% # Select columns year, country, ren\_ele, tot\_ele  filter(country == 'Nepal' | country =='India') %>% # Keep data of Nepal and India only  arrange(country, year) %>% # Sort the dataframe according to country and year columns  mutate(ren\_ele\_share = ren\_ele/tot\_ele\*100) # Create a new column ren\_ele\_share  View(energy\_np\_in) |

Summarizing by categories using **group\_by()** and **summarize()** functions.

*Example:* Suppose now you want to summarize the dataframe **energy\_np\_in** by calculating max, min, and average values of **ren\_ele\_share** in Nepal and India and save summarized dataframe as **energy\_summary**.

|  |
| --- |
| E014-dplyr\_summarize.R |
| energy\_summary <- energy\_np\_in %>%  na.omit() %>%  group\_by(country) %>%  summarize(max = max(ren\_ele\_share),  min = min(ren\_ele\_share),  mean = mean(ren\_ele\_share))  View(energy\_summary) |

**Task 2:**

Let’s summarize the dataframe **energy** by calculating max, min, and average values of **ele\_total** [Access to electricity (% of total population)] in each year.

**df\_summary <- energy %>%**

**na.omit() %>%**

**group\_by(year) %>%**

**summarize(max = max(ele\_total),**

**min = min(ele\_total),**

**mean = mean(ele\_total))**

**df\_summary**

**Session 4: Data management (Continued)**

* 1. **Merging datasets**

|  |
| --- |
| E015-joins.R |
| df1 <- data.frame(id = c(1, 2, 3), colA = c("A", "B", "C"))  df2 <- data.frame(id = c(1, 3, 5), colB = c("X", "Y", "Z"))  print(df1)    print(df2)    inner\_join(df1, df2, by = 'id') #Return rows with matching keys in both data frames    left\_join(df1, df2, by = 'id') #Return all rows from first data frame, matching rows from second    right\_join(df1, df2, by = 'id') #Return all rows from second data frame, matching rows from first    full\_join(df1, df2, by = 'id') #Return all rows from both data frames, matching by keys |

**Task 3:**

* Let’s left\_join dataframes **energy** and **classf** by common column **ccode**.
* Summarize by calculating average values of **ele\_total** [Access to electricity (% of total population)] for each **year** and **country** **group** (i.e., H, UM, LM, L).
* Save the summarized dataframe as **wb\_energy**.

|  |
| --- |
| wb\_energy <- left\_join(energy, classf, by = 'ccode') %>%  na.omit() %>%  group\_by(year, wb\_class) %>%  summarize(average = mean(ele\_total))  View(wb\_energy) |

**Task 4:**

* Load **"data/008-nlfs2.dta"** dataset and store it in **nlfs2** dataframe.
* From **nlfs2,** create a dataframe of family size named **fsize.**
* Merge dataframes **nlfs2** and **fsize**. Then replace the **nlfs2** dataframe with the merged dataframe.
* Keep columns **psu, hhid, family\_size, q10, q09.**
* Rename **q10** to **age**, **q09** to **gender**.

**library(haven)**

**library(dplyr)**

**nlfs2 <- read\_stata('data/008-nlfs2.dta') #nlfs2 data with 50 percent data**

**fsize <- nlfs2 %>%**

**count(psu, hhid) %>%**

**rename(family\_size = n)**

**# OR**

**fsize <- nlfs2 %>%**

**group\_by(psu, hhid) %>%**

**summarize(family\_size = n())**

**nlfs2 <- full\_join(nlfs2, fsize, by = c('psu','hhid'))**

**nlfs2 <- nlfs2 %>%**

**select(psu, hhid, family\_size, q10, q09) %>%**

**rename(age = q10, gender = q09)**

* 1. **Conditional data generation**

Based on the above **nlfs2** dataframe, lets create the following columns with the conditions.

* Generate a column **family\_size\_group** ( small <= 2, medium <=4, large >= 5)
* **age\_group** (teen <= 20, adult <=60, old >= 61)
* **male** (1 if male, 0 if female). Then convert the **male** to a factor variable with appropriate labels (1 – Male, 0 -- Female).

|  |
| --- |
| E016-case\_when.R |
| nlfs2 <- nlfs2 %>%  mutate(family\_size\_group = case\_when(family\_size <= 2 ~ 'Small',  family\_size <= 4 ~ 'Medium',  family\_size >= 5 ~ 'Large')) %>%  mutate(age\_group = case\_when(age <= 20 ~ 'Teen',  age <= 60 ~ 'Adult',  TRUE ~ 'Old')) %>%  mutate(male = case\_when(gender == 2 ~ 0,  gender == 1 ~ 1)) %>%  mutate(male = factor(male, levels = c(1,0), labels = c('Male', 'Female'))) |

* 1. **Exporting a dataframe to csv, excel, RData, dta files.**

|  |
| --- |
| E017-exporting\_dataframe.R |
| #----------------------------------------------------------------------  # Exporting dataframe to a csv file  #----------------------------------------------------------------------  write.csv(fsize, file = 'fsize.csv', row.names = F)  #----------------------------------------------------------------------  # Exporting dataframe to a excel file  #----------------------------------------------------------------------  library(openxlsx)  write.xlsx(fsize, file = "fsize.xlsx")  #----------------------------------------------------------------------  # Exporting dataframe to a RData file  #----------------------------------------------------------------------  save(fsize, file = 'fsize.RData')  load('fsize.RData')  #----------------------------------------------------------------------  # Exporting dataframe to a dta (Stata) file  #----------------------------------------------------------------------  library(haven)  write\_dta(fsize, path = 'fsize.dta') |