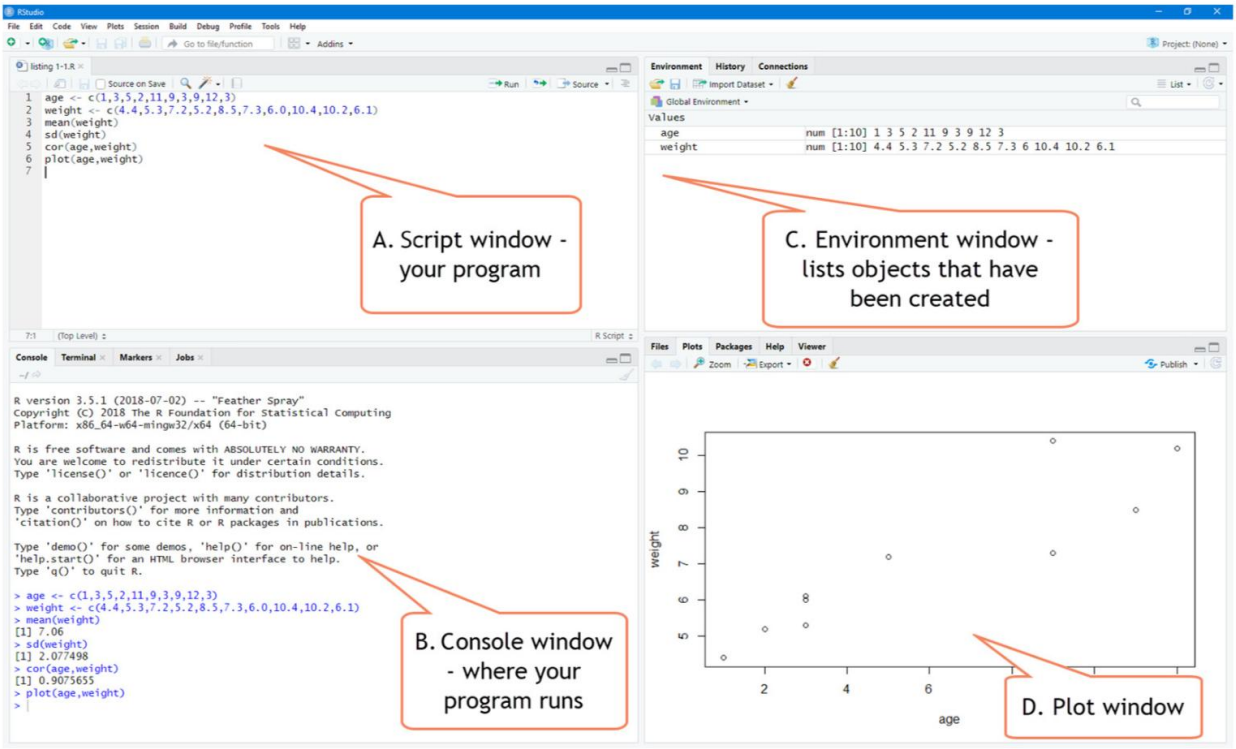
Day 1: Introduction to R programming
Session 1: Introduction to Stata and Data Import
1.1. Installing R and RStudio

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

https://www.r-project.org/

https://posit.co/download/rstudio-desktop/

1.2. Starting to work with R and RStudio



Setting working environment

E001-setting_working_directory.R

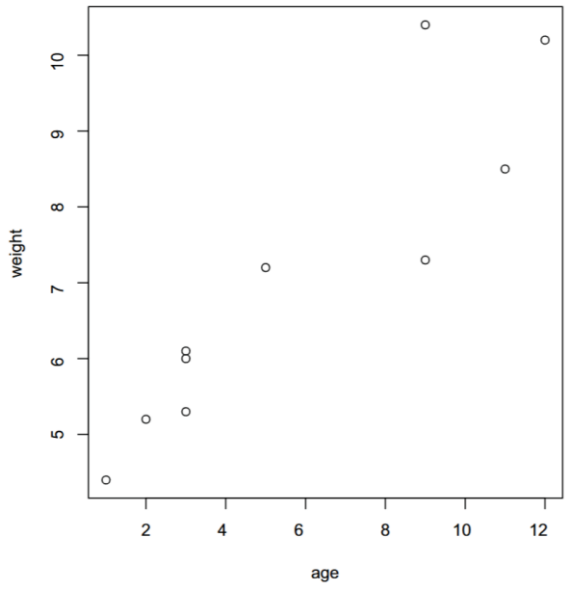
```
#checking the current working directory
getwd()

#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)
```

```
> 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)
> mean(weight)
[1] 7.06
> sd(weight)
[1] 2.077498
> cor(age,weight)
[1] 0.9075655
> plot(age,weight)
```



Getting help

E003-getting_help.R

```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

```r
#installing a package
install.packages('plotly')
```

```r
#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')
```

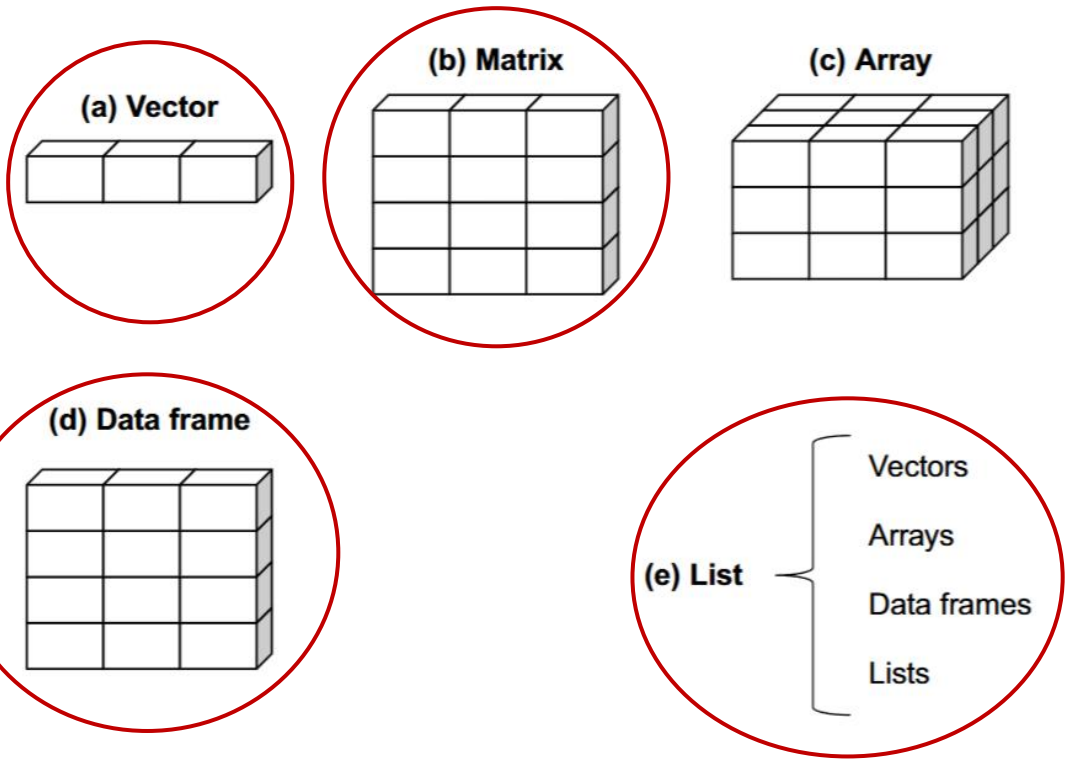
Task 1:

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.

```r
#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

2.1. 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
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

#creating a matrix with byrow = TRUE
m2 <- matrix(x, nrow = 2, ncol = 4, byrow = TRUE)
m2
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8

#-------------------------
#accessing elements of a matrix
#-------------------------
m2[1,] #selecting the first row
[1] 1 2 3 4

m2[,3] #selecting the third column
 [1] 3 7

m2[2,c(3,4)] #selecting the 3rd and 4th element of the 2nd row
[1] 7 8

#-------------------------
#basic matrix operations
#-------------------------
m1 + m2 #addition
     [,1] [,2] [,3] [,4]
[1,]    2    5    8   11
[2,]    7   10   13   16

m1 - m2 #subtraction
     [,1] [,2] [,3] [,4]
[1,]    0    1    2    3
[2,]   -3   -2   -1    0

t(m1) #transpose
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8

t(m1) %*% m2 #matrix dot product
     [,1] [,2] [,3] [,4]
[1,]   11   14   17   20
[2,]   23   30   37   44
[3,]   35   46   57   68
[4,]   47   62   77   92

mm <- matrix(c(1,2,3,4), nrow = 2)
mm
     [,1] [,2]
[1,]    1    3
[2,]    2    4

det(mm) #determinant of a matrix
[1] -2
solve(mm) #inverse of a matrix
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

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
  id age sex diabetes    status
1  1  25   0    Type1      Poor
2  2  34   1    Type2  Improved
3  3  28   1    Type2 Excellent
4  4  52   0    Type1      Poor

#------------------------------
#accessing a column
#------------------------------
df[,2] #accessing the column values by index
 [1] 25 34 28 52
df$age #accessing the column values by name
 [1] 25 34 28 52
df["age"] #accessing the column by name
  age
1  25
2  34
3  28
4  52

#------------------------------
#accessing a multiple columns
#------------------------------
df[,c(1,3,4)] #accessing the columns by index
  id sex diabetes
1  1   0    Type1
2  2   1    Type2
3  3   1    Type2
4  4   0    Type1
df[c("id","status")] #accessing the columns by name
  id    status
1  1      Poor
2  2  Improved
3  3 Excellent
4  4      Poor

#------------------------------
#creating a frequency twoway table
#------------------------------
table(df$diabetes, df$status)
       Excellent Improved Poor
Type1          0        0    2
Type2          1        1    0

#------------------------------
#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
  id age    sex diabetes    status
1  1  25 Female    Type1      Poor
2  2  34   Male    Type2  Improved
3  3  28   Male    Type2 Excellent
4  4  52 Female    Type1      Poor

df$sex
 > df$sex
 [1] Female Male   Male   Female
 Levels: Male Female

df$diabetes
[1] Type1 Type2 Type2 Type1
Levels: Type1 Type2

#------------------------------
# Variable labeling
#------------------------------
attr(df$id, "label") <- "Patient ID"
attr(df$age, "label") <- "Patient Age in years"
attr(df$sex, "label") <- "Patient sex"
View(df)
```

	id Patient ID	age Patient Age in years	sex Patient sex	diabetes	status
1	1	25	Female	Type1	Poor
2	2	34	Male	Type2	Improved
3	3	28	Male	Type2	Excellent
4	4	52	Female	Type1	Poor

```
#removing a label
attr(df$sex, "label") <- NULL
View(df)
```

	id Patient ID	age Patient Age in years	sex	diabetes	status
1	1	25	Female	Type1	Poor
2	2	34	Male	Type2	Improved
3	3	28	Male	Type2	Excellent
4	4	52	Female	Type1	Poor

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
```

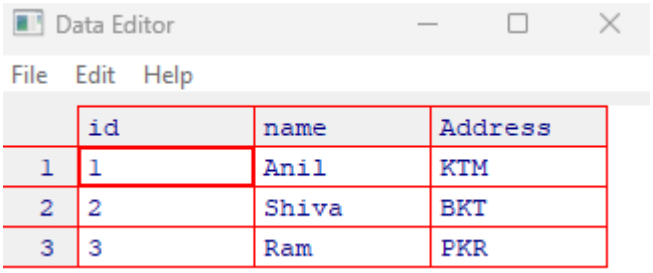
```
$title
[1] "My First List"

$ages
[1] 25 26 18 39

[[3]]
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

[[4]]
[1] "one"   "two"   "three"

#------------------------------
#accessing an object
#------------------------------
mylist$title #by object name in the list
[1] "My First List"

mylist[['ages']] #by object name in the list
[1] 25 26 18 39

mylist[[1]] #by index number
[1] "My First List"
```

2.2. Data input

Data input using keyboard (manual)

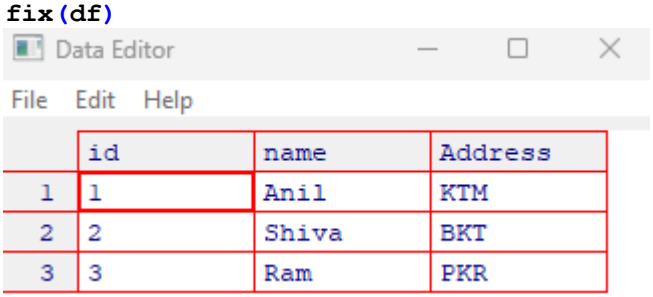
E009-data_input_keyboard.R

```
#---------------------------------------------------------------------
# Creating a new dataframe with data input using keyboard
#---------------------------------------------------------------------
df <- edit(data.frame())
```



	id	name	Address
1	1	Anil	KTM
2	2	Shiva	BKT
3	3	Ram	PKR

```
#---------------------------------------------------------------------
# Editing an existing dataframe
#---------------------------------------------------------------------
fix(df)
```



	id	name	Address
1	1	Anil	KTM
2	2	Shiva	BKT
3	3	Ram	PKR

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)
'data.frame':   84 obs. of  5 variables:
 $ ID       : int  57 46 77 17 36 23 75 39 33 55 ...
 $ Treatment: chr  "Treated" "Treated" "Treated" "Treated" ...
 $ Sex      : chr  "Male" "Male" "Male" "Male" ...
 $ Age      : int  27 29 30 32 46 58 59 59 63 63 ...
 $ Improved : chr  "Some" "None" "None" "Marked" ...

#---------------------------------------------------------------------
# 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

3.1. 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
   year country ccode ele_rural   ele_total ele_urban    en_int ren_ele  ren_con tot_ele     tfec
1  1990   Nepal   NPL        NA  0.01000000  75.22305 10.791280     877 229285.7     878 241049.6
2  1991   Nepal   NPL        NA  0.06506019  76.11861 10.473814     900 234348.3     932 248807.4
3  1992   Nepal   NPL        NA  0.45856440  77.01398 10.305795     839 239342.5     886 254184.2
4  1993   Nepal   NPL        NA  1.72627795  77.90834 10.185771     871 244530.4     933 261402.5
5  1994   Nepal   NPL        NA  3.53385210  78.80063  9.746334     929 250040.5    1010 270664.5
6  1995   Nepal   NPL        NA  7.57501268  79.68983  9.716170    1159 255819.2    1196 278875.5
7  1996   Nepal   NPL 12.100000 17.90000000  78.40000  9.446465    1181 261564.0    1221 285814.0
8  1997   Nepal   NPL  6.762828 15.61756420  81.45483  9.283220    1062 267281.6    1169 294659.4
9  1998   Nepal   NPL 10.767130 19.61283875  82.32856  9.245502    1131 273283.7    1250 302000.4
```

```
filter(energy, country == 'Nepal' & year > 1999)
   year country ccode ele_rural ele_total ele_urban   en_int ren_ele  ren_con tot_ele     tfec
1  2000   Nepal   NPL  18.76920  27.53761  84.05401 9.286283    1632 296792.2    1659 336176.8
2  2001   Nepal   NPL  17.40000  24.60000  85.70000 9.152637    1849 304942.8    1867 347487.4
3  2002   Nepal   NPL  27.01114  35.39025  85.75838 9.243013    2119 315976.7    2123 351306.6
4  2003   Nepal   NPL  31.24014  39.30412  86.60945 9.143760    2263 323323.3    2267 361485.8
5  2004   Nepal   NPL  27.30000  37.20000  87.40000 8.860626    2404 330933.7    2418 366494.6
6  2005   Nepal   NPL  39.76455  47.13726  88.32349 8.852740    2517 338400.5    2533 378027.7
7  2006   Nepal   NPL  43.20000  51.20000  90.10000 8.563978    2735 344938.9    2748 378001.7
8  2007   Nepal   NPL  48.42928  55.01801  90.07214 8.442045    2783 352430.4    2792 385961.7
9  2008   Nepal   NPL  52.83117  58.98989  90.96571 8.234679    2803 359566.3    2812 397447.3
10 2009   Nepal   NPL  57.28125  62.98281  91.87177 8.110633    3102 364858.3    3115 410289.3
11 2010   Nepal   NPL  61.77698  66.99374  92.78856 7.965682    3205 368994.9    3208 422711.1
12 2011   Nepal   NPL  72.90000  76.30000  97.00000 7.972062    3490 380337.4    3492 437443.4
13 2012   Nepal   NPL  70.89714  75.05766  94.64713 7.272666    3533 354167.1    3552 418156.1
14 2013   Nepal   NPL  75.51703  79.10464  95.58533 7.765713    3507 401076.6    3517 464999.7
15 2014   Nepal   NPL  81.70000  84.90000  97.70000 7.603696    3796 407097.6    3797 482487.3
16 2015   Nepal   NPL  84.86755  87.21362  97.47068 7.423829    3503 412327.7    3503 483589.7
17 2016   Nepal   NPL  85.20000  90.70000  94.50000       NA      NA       NA      NA       NA

# *** arrange ***
arrange(energy, desc(country), tot_ele)
   year     country ccode ele_rural ele_total ele_urban   en_int ren_ele    ren_con tot_ele      tfec
1  1998    Zimbabwe   ZWE  7.319880  33.07075  85.49733 12.680697    1926 235472.600    6583 347106.07
2  2000    Zimbabwe   ZWE  7.179767  33.59742  85.43575 13.295705    3194 250429.600    6995 361585.36
3  1999    Zimbabwe   ZWE  8.300000  38.40000  87.40000 13.634836    2908 244743.200    7091 379962.06
4  2009    Zimbabwe   ZWE 17.452830  43.36908  90.86194 22.460707    5515 288035.200    7291 346388.87
5  1997    Zimbabwe   ZWE  7.340585  32.77453  85.51710 12.749890    2123 231314.200    7298 343268.36
6  1996    Zimbabwe   ZWE  7.333691  32.45995  85.53069 13.223212    2163 228153.300    7323 341862.48
7  1993    Zimbabwe   ZWE  7.195429  31.43668  85.54465 15.862088    2062 216519.000    7468 336116.97
8  1994    Zimbabwe   ZWE  3.400000  28.10000  80.40000 14.416903    2095 220858.100    7535 322292.58
9  2007    Zimbabwe   ZWE  9.713195  35.18448  85.19933 20.511152    5446 282020.900    7609 361194.15
10 2008    Zimbabwe   ZWE 10.331723  35.45731  85.10939 23.358132    5787 283868.200    7625 346045.80

# *** na.omit ***
na.omit(energy)

# *** slice *** : used to choose rows using their position
slice(energy,3:7)
  year       country ccode  ele_rural ele_total ele_urban   en_int ren_ele     ren_con tot_ele       tfec
1 1990       Algeria   DZA  96.392315  98.27138 100.00000 3.500935     135    811.7773 16104.0 458040.442
2 1990 American Samoa   ASM         NA        NA        NA       NA       0      0.0000   100.0    306.000
3 1990       Andorra   AND 100.000000 100.00000 100.00000       NA     120    952.1450   120.0   6670.695
4 1990        Angola   AGO   7.518615  11.39781  22.68237 4.605300     725 135443.7000   841.0 187451.703
5 1990      Anguilla   AIA         NA  89.19866  89.19866       NA       0      1.8270    16.7    615.397

slice_head(energy, n = 3)
#OR
head(energy, n = 3)
  year     country ccode ele_rural ele_total ele_urban   en_int ren_ele    ren_con tot_ele      tfec
1 1990 Afghanistan   AFG        NA   0.01000  52.03698 1.884113     764  6312.3920    1128  39639.42
2 1990     Albania   ALB 100.00000 100.00000 100.00000 7.912243    2848 20429.1800    3296  80057.64
3 1990     Algeria   DZA  96.39231  98.27138 100.00000 3.500935     135   811.7773   16104 458040.44

slice_tail(energy, n = 3)
#OR
tail(energy, n = 3)
     year     country ccode ele_rural ele_total ele_urban en_int ren_ele ren_con tot_ele tfec
6991 2016 Yemen, Rep.   YEM 57.691162  71.64235  97.33986      NA      NA      NA      NA   NA
6992 2016      Zambia   ZMB  2.657746  27.21934  62.01537      NA      NA      NA      NA   NA
6993 2016    Zimbabwe   ZWE 15.575584  38.14514  85.50016      NA      NA      NA      NA   NA

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
```

```
    year ccode      tot_ele
1   1990   AFG    1128.0000
2   1990   ALB    3296.0000
3   1990   DZA   16104.0000
4   1990   ASM     100.0000
5   1990   AND     120.0000
6   1990   AGO     841.0000
7   1990   AIA      16.7000
8   1990   ATG      95.0000
9   1990   ARG   50740.0000
10  1990   ARM   10362.0000
11  1990   ABW     238.0000
```

```
select(energy, year:ccode) #selects columns from year to ccode
```

```
    year                                    country ccode
1   1990                                Afghanistan   AFG
2   1990                                    Albania   ALB
3   1990                                    Algeria   DZA
4   1990                             American Samoa   ASM
5   1990                                    Andorra   AND
6   1990                                     Angola   AGO
7   1990                                   Anguilla   AIA
8   1990                        Antigua and Barbuda   ATG
9   1990                                  Argentina   ARG
10  1990                                    Armenia   ARM
11  1990                                      Aruba   ABW
```

```
select(energy, !(year:ccode)) #selects columns other than from year to ccode
```

```
      ele_rural   ele_total  ele_urban    en_int     ren_ele      ren_con     tot_ele        tfec
1            NA   0.0100000  52.036976  1.884113    764.0000 6.312392e+03   1128.0000 3.963942e+04
2   100.0000000 100.0000000 100.000000  7.912243   2848.0000 2.042918e+04   3296.0000 8.005764e+04
3    96.3923147  98.2713776 100.000000  3.500935    135.0000 8.117773e+02  16104.0000 4.580404e+05
4            NA          NA         NA        NA      0.0000 0.000000e+00    100.0000 3.060000e+02
5   100.0000000 100.0000000 100.000000        NA    120.0000 9.521450e+02    120.0000 6.670695e+03
6     7.5186151  11.3978081  22.682375  4.605300    725.0000 1.354437e+05    841.0000 1.874517e+05
7            NA  89.1986618  89.198662        NA      0.0000 1.827000e+00     16.7000 6.153970e+02
8    76.9614775  85.1231995 100.000000  3.953882      0.0000 0.000000e+00     95.0000 2.551900e+03
9    90.6408234  90.6408234  90.640823  5.439097  17983.0000 1.057142e+05  50740.0000 1.184751e+06
10   95.7897045  97.6803742  98.593979 24.372197   1555.0000 5.502305e+03  10362.0000 2.596720e+05
11   76.7569078  88.4453506  88.905413        NA      0.0000 2.366578e+00    238.0000 3.975967e+03
```

```
head(select(energy, contains('tot'))) #selects columns with names that
contains tot
```

```
  ele_total tot_ele
1   0.01000    1128
2 100.00000    3296
3  98.27138   16104
4        NA     100
5 100.00000     120
6  11.39781     841
```

```
head(select(energy, starts_with('ele'))) #selects columns with names that
starts with ele
```

```
    ele_rural ele_total ele_urban
1          NA   0.01000  52.03698
2 100.000000 100.00000 100.00000
3  96.392315  98.27138 100.00000
4          NA        NA        NA
5 100.000000 100.00000 100.00000
6   7.518615  11.39781  22.68237
```

```
head(select(energy, ends_with('ele'))) #selects columns with names that ends
with ele
```

```
  ren_ele tot_ele
1     764    1128
2    2848    3296
3     135   16104
4       0     100
5     120     120
6     725     841
```

```
# *** rename ***
head(rename(energy, year_AD = year))
```

```
  year_AD        country ccode  ele_rural ele_total ele_urban    en_int ren_ele     ren_con tot_ele
1    1990    Afghanistan   AFG         NA   0.01000  52.03698 1.884113     764   6312.3920    1128  3
2    1990        Albania   ALB 100.000000 100.00000 100.00000 7.912243    2848  20429.1800    3296  8
3    1990        Algeria   DZA  96.392315  98.27138 100.00000 3.500935     135    811.7773   16104 45
4    1990 American Samoa   ASM         NA        NA        NA       NA       0      0.0000     100
5    1990        Andorra   AND 100.000000 100.00000 100.00000       NA     120    952.1450     120
6    1990         Angola   AGO   7.518615  11.39781  22.68237 4.605300     725 135443.7000     841 18

# *** mutate ***
head(mutate(energy, ren_ele_share = ren_ele/tot_ele * 100))
_con tot_ele       tfec ren_ele_share
3920    1128  39639.420     67.730496
1800    3296  80057.645     86.407767
7773   16104 458040.442      0.838301
0000     100    306.000      0.000000
1450     120   6670.695    100.000000
7000     841 187451.703     86.206897
```

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.

```r
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)

4.1. Merging datasets

E015-joins.R

```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)
  id colA
1  1    A
2  2    B
3  3    C

print(df2)
  id colB
1  1    X
2  3    Y
3  5    Z

inner_join(df1, df2, by = 'id') #Return rows with matching keys in both data
frames
  id colA colB
1  1    A    X
2  3    C    Y

left_join(df1, df2, by = 'id') #Return all rows from first data frame,
matching rows from second
  id colA colB
1  1    A    X
2  2    B <NA>
3  3    C    Y

right_join(df1, df2, by = 'id') #Return all rows from second data frame,
matching rows from first
  id colA colB
1  1    A    X
2  3    C    Y
3  5 <NA>    Z

full_join(df1, df2, by = 'id') #Return all rows from both data frames,
matching by keys
  id colA colB
1  1    A    X
2  2    B <NA>
3  3    C    Y
4  5 <NA>    Z
```

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)
```

4.2. 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')))
```

4.3. 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)

#---------------------------------------------------------------------
```

```r
# 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')
```