

# EC969: Application for Data Analysis

## R Basics

Dr Angus Holford and Dr Sania Wadud

Session 2022-2023

In this session, we will learn some R basics.

### Key Terms

RStudio - RStudio is a Graphical User Interface (GUI) for easier use of R.

Objects - Everything you store in R.

Functions - A function is a code operation that accept inputs and returns a transformed output.

Packages - An R package is a shareable bundle of functions.

Scripts - A script is the document file that hold your commands.

### Resources for learning

#### Resources within RStudio

- Help documentation
- Interactive tutorials
  - learnr R package
  - swirl

## Cheatsheets

There are many PDF “cheatsheets” available on the RStudio website, for example:

- Factors with *forcats* package
- Dates and times with *lubridate* package
- Strings with *stringr* package
- iterative operations with *purrr* package
- Data import
- Data transformation cheatsheet with *dplyr* package
- R Markdown (to create documents like PDF, Word, Powerpoint...)
- Shiny (to build interactive web apps)
- Data visualization with *ggplot2* package
- Cartography (GIS)
- *leaflet* package (interactive maps)
- Python with R (*reticulate* package)

## Installation

We have shown that in other class.

### How to update R and

Version- *sessionInfo()*

update- *installr :: updateR()* or “Help” -> “Check for Updates”

### Other software you may need to install

- TinyTeX (for compiling an RMarkdown document to PDF)

- Pandoc (for compiling RMarkdown documents)
- RTools (for building packages for R)
- phantomjs (for saving still images of animated networks, such as transmission chains)

## RStudio orientation

- The Source Pane
- The R Console Pane
- The Environment Pane
- Plots, Viewer, Packages, and Help Pane

## Basic commands in RStudio

### Run a command

Commands in R can be entered directly into the console. Entering 3 or more characters of a command into the console or a script will open the suggested command menu. The command menu suggests commands or the names of variables you have intended to type, alongside a description and suggested use of the command.

After completing a command and pressing *Enter*, R will immediately run the code, print the output and move to a new line. Using the  $\uparrow$  key will repeat the last command entered into the console.

### Using R Script

Multiple commands can be entered into a script, one after the other across multiple lines. R Script allows you to edit and reuse previous commands and to create more complicated lists of commands.

To run the script one line at a time, navigate the cursor to the appropriate line and press *CTRL + Enter*. To run all commands from the start, press *CTRL + Shift + Enter*.

## Saving script in R

To save the script click File > Save and then enter the file name ‘rbasics’ in the project folder. This will save your script as ‘rbasics.R’.

## Objects

Everything in R is an object, and R is an “object-oriented” language. These sections will explain:

- How to create objects (<-)
- Types of objects (e.g. data frames, vectors..)
- How to access subparts of objects (e.g. variables in a dataset)
- Classes of objects (e.g. numeric, logical, integer, double, character, factor)

## Defining Objects

Defining objects (<-) “is defined as”

*object\_name* <- *value*

Equals signs =

You will also see equals signs in R code:

- A double equals sign == between two objects or values asks a logical question: “is this equal to that?”.
- You can use a single equals sign = in place of <- to create and define objects, but this is discouraged.

A quick note on naming of objects

- Object names must not contain spaces, but you should use underscore (\_) or a period (.) instead of a space.
- Object names are case-sensitive (meaning that Dataset\_A is different from dataset\_A).
- Object names must begin with a letter (cannot begin with a number like 1, 2 or 3)

## Object structure

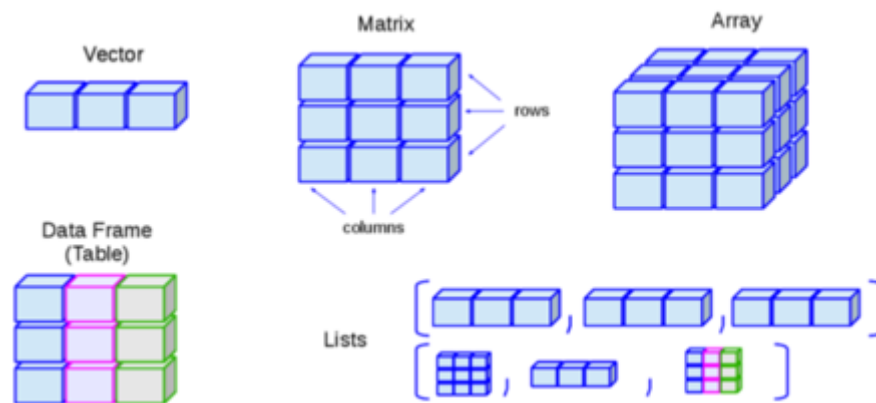


Figure 1: data structure

## Object Classes

Class	Examples
Character	"Character objects are in quotation marks"
Integer	-6, 12, or 2020
Numeric	23.5 or 16
Factor	An variable of economic status with ordered values
Date	2018-06-12 or 15/3/1956 or Wed 6 Jan 1988
Logical	TRUE or FALSE
data.frame	The example AJS dataset

Class	Examples
tibble	Any data frame, list, or matrix can be converted to a tibble with <code>as_tibble()</code>
list	A list could hold a single number, and a dataframe, and a vector, and even another list within it!

## Operators

### Mathematical Operators

Purpose	Example in R
addition	$2 + 3$
subtraction	$2 - 3$
multiplication	$2 * 3$
division	$30 / 5$
exponent	$2^3$
order of operations	$( )$

### Mathematical functions

Purpose	Function
rounding	<code>round(x, digits = n)</code>
rounding	<code>janitor::round_half_up(x, digits = n)</code>
ceiling (round up)	<code>ceiling(x)</code>
floor (round down)	<code>floor(x)</code>
absolute value	<code>abs(x)</code>
square root	<code>sqrt(x)</code>

Purpose	Function
exponent	exponent(x)
natural logarithm	log(x)
log base 10	log10(x)
log base 2	log2(x)

Using %in%

```
vector <- c("a", "b", "c", "d","e")

"a" %in% vector

"b" %in% vector

# to negate, put an exclamation in front

!"a" %in% vector
```

## Start a new project in R

When starting to work with a new dataset, a new project should be created.

## Working directory

Getting working directory

```
getwd()
```

Setting working directory *setwd()*

## Installing packages

To run commands other than basic functions, you will need to install some plugins or packages to bring extra functionality. To install packages use the command *install.packages()*. You can install the following packages in R.

```
install.packages(c("readxl", "psych", "FSA", "car", "ggplot2", "stargazer"))
```

## Loading packages

```
library(readxl)
library(psych)
library(FSA)
library(car)
library(ggplot2)
library(stargazer)
```

## Importing an excel file into R

To import the data from an excel file, use the command *read\_excel()*.

```
# you can use package wooldridge to get data as well.
datadf <- read_excel("M:/EC969/R/EC969/wages.xlsx", sheet = "WAGE2")

#install.packages('AER')
## install devtools and then
devtools::install_git("https://github.com/ccolonescu/PoEdata")
## install.packages(POEdata)
library(AER)
data(Fatalities)
library(PoEdata)
data(food, package='PoEdata')
```



The data is now imported into R.

## Viewing imported data

To view all the data, you can use `View()` command to view data/variables.

```
View(datadf)
```

```
View(Fatalities)
```

To view a particular part of the data set, for example, you can use command

```
# to view wage column
```

```
View(datadf$wage)
```

```
# to view wage to IQ
```

```
View(datadf[,1:3])
```

```
# before comma represents row number
```

```
# after comma presents column number
```

```
# to view first 10 rows of datadf
```

```
View(datadf[1:10,])
```

To see the variables in console, you can use `print()`

```
# to view first 10 rows of wage to IQ in console
```

```
print(datadf[1:10,1:3])
```

```
## # A tibble: 10 x 3
```

```
##   wage hours   IQ
```

```
##   <dbl> <dbl> <dbl>
```

```
## 1   769   40   93
```

```
## 2   808   50  119
```

```
## 3 825 40 108
## 4 650 40 96
## 5 562 40 74
## 6 1400 40 116
## 7 600 40 91
## 8 1081 40 114
## 9 1154 45 111
## 10 1000 40 95
```

## R variables and data type

The common variable types and data types that you will be working with in R is as follows:

```
str(datadf)
str(Fatalities)
str(food)
```

## Summary/Descriptive statistics

It is important to know the summary and descriptive statistics of the data. You can do this using the following commands. *summary()* command provide details of minimum, 1st quantile, median, mean, 3rd quantile and maximum of variables. *psych::describe()* provide

```
# to view summary of the data
summary(datadf)
```

```
##      wage      hours      IQ      educ
##  Min.   : 115.0  Min.   :20.00  Min.   : 50.0  Min.   : 9.00
##  1st Qu.: 669.0  1st Qu.:40.00  1st Qu.: 92.0  1st Qu.:12.00
##  Median : 905.0  Median :40.00  Median :102.0  Median :12.00
##  Mean   : 957.9  Mean   :43.93  Mean   :101.3  Mean   :13.47
##  3rd Qu.:1160.0  3rd Qu.:48.00  3rd Qu.:112.0  3rd Qu.:16.00
```

```
## Max. :3078.0 Max. :80.00 Max. :145.0 Max. :18.00
## exper tenure age married
## Min. : 1.00 Min. : 0.000 Min. :28.00 Min. :0.000
## 1st Qu.: 8.00 1st Qu.: 3.000 1st Qu.:30.00 1st Qu.:1.000
## Median :11.00 Median : 7.000 Median :33.00 Median :1.000
## Mean :11.56 Mean : 7.234 Mean :33.08 Mean :0.893
## 3rd Qu.:15.00 3rd Qu.:11.000 3rd Qu.:36.00 3rd Qu.:1.000
## Max. :23.00 Max. :22.000 Max. :38.00 Max. :1.000
## south urban
## Min. :0.0000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000
## Median :0.0000 Median :1.0000
## Mean :0.3412 Mean :0.7176
## 3rd Qu.:1.0000 3rd Qu.:1.0000
## Max. :1.0000 Max. :1.0000
```

*# to view descriptive statistics of data*

```
psych::describe(datadf)
```

```
## vars n mean sd median trimmed mad min max range skew
## wage 1 935 957.95 404.36 905 919.69 369.17 115 3078 2963 1.20
## hours 2 935 43.93 7.22 40 42.80 0.00 20 80 60 1.59
## IQ 3 935 101.28 15.05 102 101.87 14.83 50 145 95 -0.34
## educ 4 935 13.47 2.20 12 13.33 1.48 9 18 9 0.55
## exper 5 935 11.56 4.37 11 11.52 4.45 1 23 22 0.08
## tenure 6 935 7.23 5.08 7 6.89 5.93 0 22 22 0.43
## age 7 935 33.08 3.11 33 33.04 4.45 28 38 10 0.12
## married 8 935 0.89 0.31 1 0.99 0.00 0 1 1 -2.54
## south 9 935 0.34 0.47 0 0.30 0.00 0 1 1 0.67
## urban 10 935 0.72 0.45 1 0.77 0.00 0 1 1 -0.97
## kurtosis se
```

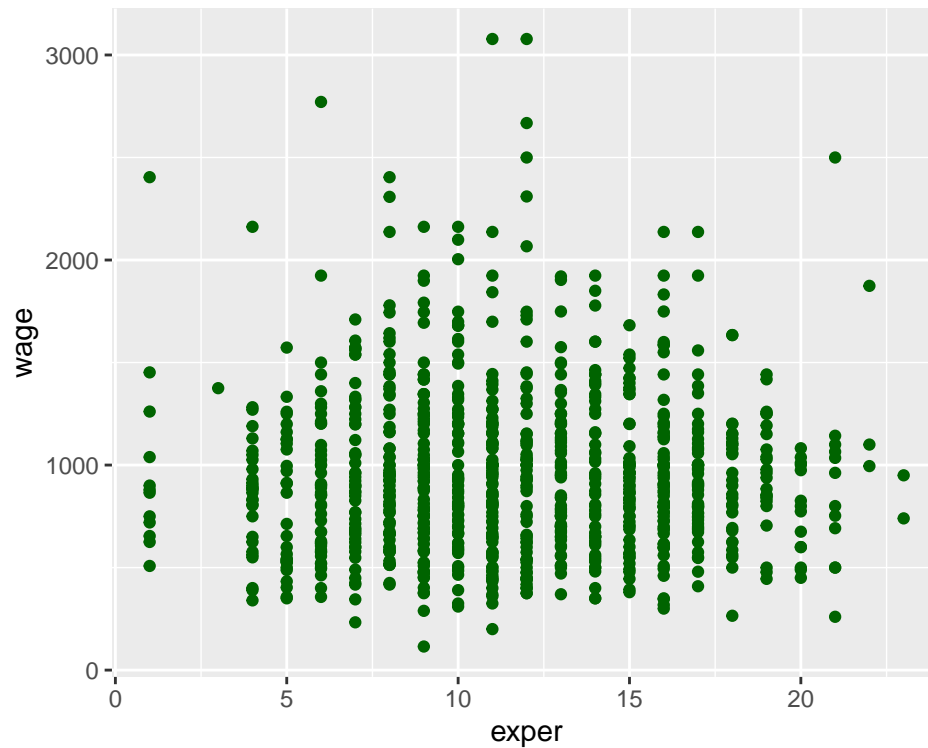
```
## wage      2.68 13.22
## hours      4.14  0.24
## IQ        -0.03  0.49
## educ      -0.74  0.07
## exper     -0.57  0.14
## tenure    -0.81  0.17
## age       -1.26  0.10
## married    4.45  0.01
## south     -1.55  0.02
## urban     -1.07  0.01
```

## Plotting data

### Scatter plot

Looking at the data is a good practice. To do a scatter plot, for instance *wage* versus *exper*, you can use the command

```
# you can change shape, color, size of geom point
ggplot(datadf,aes(x=exper,y=wage))+ geom_point(color="darkgreen")
```



```
#from PoEdata using plot function
plot(food$income, food$food_exp,
     ylim=c(0, max(food$food_exp)),
     xlim=c(0, max(food$income)),
     xlab="weekly income in $100",
     ylab="weekly food expenditure ($)",
     type = "p")
```



## Removing variables from your data

Sometimes you might want to remove variables from your dataset. There are various ways to do that. For example, to remove the variables *south* and *urban* we could use,

1. Delete column by name

The most easiest way to drop columns is by using `subset()` function. In the code below, we are telling R to drop variables *south* and *urban*. The `-` sign indicates dropping variables. Make sure the variable names would NOT be specified in quotes when using `subset()` function.

```
dfsthur <- subset(datadf, select = -c(south,urban))
```

2. Drop columns by column index numbers

It's easier to remove variables by their position number. In the following code, we want to drop variables that are positioned at ninth column and tenth columns. The `-` sign is to drop variables.

```
dfsthur <- datadf[,-c(9,10)]
```

Alternatively, to remove all variables other than *wage* we could use any of the following commands

```
dfwage <- datadf$wage # using $
```

```
dfwage <- subset(datadf,select = wage) # using function subset()
```

```
dfwage <- datadf[,1] # column index
```

Also, if a variable already exists and you want to redefine its content, you can change the right side of the function to change the content.

### 3. Deleting object

If you want to delete a object you have created, you can use the function *rm()*. For removing all the object use *rm(list = ls())*.

## Estimating a Linear Regression

$$food\_exp = \beta_1 + \beta_2 income + u$$

```
# Running linear regression
```

```
mod1 <- lm(food_exp ~ income, data = food)
```

```
# beta1
```

```
b1 <- coef(mod1)[[1]]
```

```
#beta2
```

```
b2 <- coef(mod1)[[2]]
```

```
# summary results from linear regression
```

```

smod1<- summary(mod1)

smod1

##
## Call:
## lm(formula = food_exp ~ income, data = food)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -223.025  -50.816   -6.324   67.879  212.044
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   83.416     43.410   1.922  0.0622 .
## income       10.210      2.093   4.877 1.95e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 89.52 on 38 degrees of freedom
## Multiple R-squared:  0.385, Adjusted R-squared:  0.3688
## F-statistic: 23.79 on 1 and 38 DF, p-value: 1.946e-05

```

Using a fancy table

```

# Using fancy table style
stargazer::stargazer(mod1,type='latex')

```



Table 4:

	Dependent variable:
	food_exp
income	10.210*** (2.093)
Constant	83.416* (43.410)
Observations	40
R <sup>2</sup>	0.385
Adjusted R <sup>2</sup>	0.369
Residual Std. Error	89.517 (df = 38)
F Statistic	23.789*** (df = 1; 38)
Note:	*p<0.1; **p<0.05; ***p<0.01

### Plotting with coefficients

```
plot(food$income, food$food_exp,
      xlab="weekly income in $100",
      ylab="weekly food expenditure in $",
      type = "p")
abline(b1,b2)
```



Getting name/coefficients from estimated model

```
# list of names
```

```
names(mod1)
```

```
## [1] "coefficients" "residuals"    "effects"      "rank"
## [5] "fitted.values" "assign"        "qr"           "df.residual"
## [9] "xlevels"       "call"          "terms"        "model"
```

```
names(smod1)
```

```
## [1] "call"          "terms"          "residuals"      "coefficients"
## [5] "aliased"        "sigma"          "df"             "r.squared"
## [9] "adj.r.squared" "fstatistic"     "cov.unscaled"
```

```
# coefficient
```

```
mod1$coefficients
```

```
## (Intercept)      income
```

```
##      83.41600      10.20964
```

```
smod1$coefficients
```

```
##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 83.41600   43.410163  1.921578 6.218242e-02
## income      10.20964    2.093264  4.877381 1.945862e-05
```

## Prediction with the Linear Regression Model

$$\widehat{food\_exp} = 83.416002 + 10.209643 * 20 = 287.608861$$

```
mod1 <- lm(food_exp~income, data=food)
newx <- data.frame(income = c(20, 25, 27))
y_hat <- predict(mod1, newx)
names(y_hat) <- c("income=$2000", "$2500", "$2700")
y_hat
```

```
## income=$2000      $2500      $2700
##      287.6089      338.6571      359.0764
```

## Estimated Variances and Covariance of Regression Coefficients

```
varb1 <- vcov(mod1)[1, 1]; varb1
```

```
## [1] 1884.442
```

```
varb2 <- vcov(mod1)[2, 2]; varb2
```

```
## [1] 4.381752
```

```
covb1b2 <- vcov(mod1)[1,2]; covb1b2
```

```
## [1] -85.90316
```