Data wrangling : I/O (Part-1)

Data wrangling is a task of great importance in data analysis. Data wrangling, is the process of importing, cleaning and transforming raw data into actionable information for analysis. It is a time-consuming process which is estimated to take about 60-80% of analyst's time. In this series we will go through this process. It will be a brief series with goal to craft the reader's skills on the data wrangling task. This is the first part of this series and it aims to cover the importing and exporting of data locally.
Please download the data from here

Before proceeding, it might be helpful to look over the help pages for the read.csv, read.table, read_excel, fromJSON, as.data.frame, xmlParse, xmlToList, write.csv, write.table, write.xlsx, toJSON, write, write.xml.

Moreover please load the following libraries.

```
install.packages("readxl")
library(readxl)
install.packages("xlsx")
library(xlsx)
install.packages("rjson")
library(rjson)
install.packages("plyr")
library(plyr)
install.packages("XML")
library(XML)
install.packages("kulife")
library(kulife)
```

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1

Import the data.csv file to the csv_file object.

Exercise 2

Import the data.txt file to the txt_file object.

Exercise 3

Import the data.xls file to the xls_file object.

Exercise 4

Import the data.xlsx file to the xlsx_file object.

Learn more about Data Pre-Processing in the online course R Data Pre-Processing & Data Management – Shape your Data!. In this course you will learn how to:

- import data into R in several ways while also beeing able to identify a suitable import tool
- use SQL code within R
- And much more

Exercise 5

Import the data.json file to the json_file object.

Exercise 6

Import the data.xml file to the xml_file object.

Exercise 7

Export the csv_file as “data.csv” and txt_file as “data.txt”.

Exercise 8

Export the xls_file as “data.xls” and xlsx_file as “data.xlsx”.

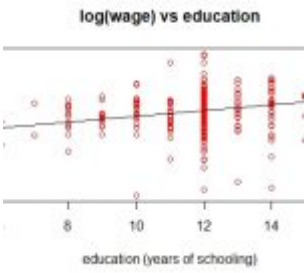
Exercise 9

Export the json_file as “data.json”.

Exercise 10

Export the xml_file as “data.xml”.

Instrumental Variables in R exercises (Part-2)



This is the second part of the series on Instrumental Variables. For other parts of the series follow the tag instrumental variables.

In this exercise set we will build on the example from part-1. We will now consider an over-identified case i.e. we have multiple IVs for an endogenous variable. We will also look at tests for endogeneity and over-identifying restrictions.

Answers to the exercises are available here.

Exercise 1

Load the AER package (package description: here). Next, load PSID1976 dataset provided with the AER package. This has data regarding labor force participation of married women.
Define a new data-frame that has data for all married women that were employed. This data-frame will be used for the following exercises.

Exercise 2
We will use a more elaborate model as compared to the previous set.
Regress log(wage) on education, experience and experience^2.

Exercise 3
Previously, we identified feducation and meducation as possible IVs for education.
Regress education on experience, experience^2, feducation and meducation. Comment on your results.

Exercise 4
Test the hypothesis that feducation and meducation are jointly equal to zero.

Exercise 5
Use 2SLS to estimate the IV based return to education.

Exercise 6
Use the ivreg command to get the same results as in Exercise-5. Note that standard errors would be different.

Exercise 7
There is a short form for the ivreg command which saves time when we are dealing with numerous variables.
Try the short format and check that you get the same results as in Exercise-6.

Exercise 8
Regress log(wage) on education, experience, experience^2 and residuals from the model estimated in Exercise-3.
Use your result to test for the endogeneity of education.

Exercise 9
Regress the residuals from exercise-7 on experience, experience^2, feducation and meducation.
Use your result to test for over-identifying restrictions.

Exercise 10
The two tests we did in exercises 8 and 9 can be conveniently obtained using the summary command with diagnostics turned on. Verify that you get the same test results with the summary command.

Manipulate Biological Data Using Biostrings Package Exercises (Part 1)

Bioinformatics is an amalgamation Biology and Computer science.Biological Data is manipulated using Computers and Computer software's in Bioinformatics. Biological Data includes DNA; RNA & Protiens. DNA & RNA is made of Nucleotides which are our genetic material in which we are coded.Our Structure and Functions are done by protein, which are build of Amino acids

In the exercises below we cover how we can Manipulate Biological Data using Biostrings package in Bioconductor.

Install Packages
Biostrings

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1

Print out the standard Genetic Code table using Biostrings Package

Exercise 2

Print the first codon in the standard genetic code

Exercise 3

Print out the Standard RNA Genetic Code table using Biostrings package

Exercise 4

Print out the Standard RNA Genetic Code of Stop codon using Biostrings package

Exercise 5

Print out the standard Amino acid codon table using Biostrings

Exercise 6

Print the code of the start codon from the standard genetic code

Exercise 7

Print the three letter code of Amino acid Methionine using Biostrings

Exercise 8

Create a DNA string and print the length and dinucleotide frequency of the string

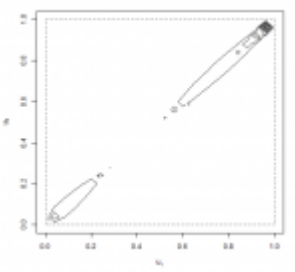
Exercise 9

Create RNA string and print the length and dinucleotide frequency of the string

Exercise 10

Create a Protein string and print the length of the protein

Shiny Applications Layouts Exercises (Part-9)



Shiny Application Layouts – Update Input

In the ninth part of our series we will see the updated input

scenario. This is different from the Dynamic UI example we met in part 8, where the UI component is generated on the server and sent to the UI, where it replaces an existing UI component.

This part can be useful for you in two ways.

First of all, you can see different ways to enhance the appearance and the utility of your shiny app.

Secondly you can make a revision on what you learnt in "Building Shiny App" series as we will build basic shiny staff in order to present it in the proper way.

Read the examples below to understand the logic of what we are going to do and then test yous skills with the exercise set we prepared for you. Lets begin!

Answers to the exercises are available here.

UI Context

As always let us create the skeleton of the app.

```
#ui.r
fluidPage(
titlePanel("Title"),
fluidRow(
column(3,
wellPanel()),
column(3,
wellPanel()),
column(3,
wellPanel())
)
)
#server.R
function(input, output, clientData, session) {}
```

Exercise 1

Build the initial UI of your app. Give it a title, use 4 columns and one row for each one of the three well Panel that we are going to use.

Applying reactivity

To create a reactive context for our app we will use the observe({}) function in our server side.

Exercise 2

Create reactive context. **HINT**: Use observe({}).

"Master" Inputs.

Now we are going to create the two inputs that control the rest of the inputs of the app. Look at the example:

```
#ui.r
textInput("text_input",
"labels:",
"Some Text"),
sliderInput("slider_input",
"values:",
min = 1, max = 100, value = 50)
#server.r
t_input <- input$text_input
s_input <- input$slider_input
```

Exercise 3

Create a text Input in the ui side that controls labels and then pass it to a new variable on the server side.

Exercise 4

Create a slider Input in the ui side that controls values and then pass it to a new variable on the server side.

Dependent Inputs

Firstly let’s create a text input that changes both the label and the text.

```
#ui.r
textInput("Text", "Text input:", value = "text")
#server.r
updateTextInput(session, "Text",
label = paste("Sth", t_input),
value = paste("Sth", s_input)
)
```

Exercise 5

Create a text input, in the second well Panel,that changes its label and its value according to the two “Master” Inputs you created before. **HINT**: Use updateTextInput().

Exercise 6

Create a slider input,in the second well Panel,that changes its label and its value according to the two “Master” Inputs you created before. **HINT**: Use updateSliderInput().

Exercise 7

Create a numeric input,in the second well Panel,that changes its label and its value according to the two “Master” Inputs you created before. **HINT**: Use updateNumericInput().

Exercise 8

Create a date input,in the second well Panel,that changes its label and its value according to the two “Master” Inputs you created before. **HINT**: Use updateDateInput().

In order to create a Checkbox group with the same conditions

as the rest of the inputs we just created we should first create a list of options like in the example below:

```
#server.r
options <- list()
options[[paste("option", s_input, "A")]] <-
paste0("option-", s_input, "-A")
options[[paste("option", s_input, "B")]] <-
paste0("option-", s_input, "-B")
```

Exercise 9

Create a list with three choices for your Checkbox Group. **HINT**: Use list().

Exercise 10

Create a checkboxgroup input, in the third well Panel,that changes its label and its value according to the two "Master" Inputs you created before. **HINT**: Use updateCheckboxGroupInput().

A Primer in Functional Programming in R Exercises (Part – 1)

In the exercises below we cover the basics of functional programming in R(part 1 of a two series exercises on functional programming) . We consider recursion with R , apply family of functions , higher order functions such as Map ,Reduce,Filter in R .
Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Create a function which calculates factorial of a number with help of Reduce ,

Exercise 2
Create the same function which calculates factorial but with recursion and memoization. :

Exercise 3
Create a function cum_add which makes cumulative summation for e.g if x <- 1:3 cum_add(x) will result in 1 3 6 .Don't use cumsum .

Exercise 4
Create a function which takes a dataframe and returns mean , minimum and maximum of all numeric columns . Your function should take a dataframe as input .for e.g your_func(iris)
.Try to avoid loops ,its possible to make it in one line .swag your functional skills .

Learn more about Functions in the online course R Programming: Advanced Analytics In R For Data Science. In this course you will learn how to prepare your data quicker and more efficiently, leaving you more time to focus on analyzing and visualizing your results!

Exercise 5
Create a function centerColumnAroundMean which takes a numeric

dataframe and manipulates the dataframe such a way that all column’s values are centered with mean value of the column . For example if my data frame is like this
df <- data.frame(x= 1:5,y=6:10,c=11:15)

then centerColumnAroundMean(df) will result in

```
##    x  y  c
## 1 -2 -2 -2
## 2 -1 -1 -1
## 3  0  0  0
## 4  1  1  1
## 5  2  2  2
```

It is possible to achieve this by single line of R , Please dont use loops .

Exercise 6

I have list of movies ,which have two movie franchieses as the elements ,Starwars and LOTR.You can create the movie list by
my_movie_list<- list(STARWARS= list("A NEW HOPE","The Last Jedi","The Force Awakens"),LOTR=list("THE FELLOWSHIP OF THE RING","THE Two Towers","The RETURN of the KING","Hobbit" = list("An unexpected Journey","The Battle of the FIVE ARMY","The Desolation of Smaug")))
now the problem I am facing is some of the texts are in caps and some of them are in small ,without any particular order . I would like the list to have a format like
“The Last Jedi” , Help me in writing a function which will do the same . Please keep in mind that the list is a nested list .

Exercise 7

Load the diamonds data set from ggplot 2 package , I want to buy a diamond of each color but don’t want to pay the top price , I believe that the second highest price is good enough for me . Help me in finding the second highest price for each color from the dataset.

Exercise 8

Use the already loaded diamonds dataset from previous exercise , I want to know the average price for each combination of cut and color .Your output should be similar to following.Don't use table.

```
##                D       E       F       G       H       I       J
## Fair      4291.061 3682.312 3827.003 4239.255 5135.683 4685.446 4975.655
## Good      3405.382 3423.644 3495.750 4123.482 4276.255 5078.533 4574.173
## Very Good 3470.467 3214.652 3778.820 3872.754 4535.390 5255.880 5103.513
## Premium   3631.293 3538.914 4324.890 4500.742 5216.707 5946.181 6294.592
## Ideal     2629.095 2597.550 3374.939 3720.706 3889.335 4451.970 4918.186
```

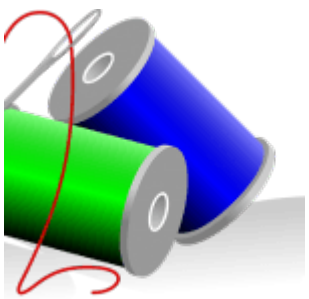
Exercise 9

Load the iris dataset , I want to get the third row for each group of species . Help me to write a short function for me to achieve that .

Exercise 10

Create a new environment with new.env() command and create 3 vector variables under that environment as a=1:10;b=100:500;c=1000:1500
without knowing or manually calling mean for all the vector variables can you print the mean of all variables of the new environment .

Character Functions (Advanced)



This set of exercises will help you to help you improve your skills with character functions in R. Most of the exercises are related with text mining, a statistical technique that analyses text using statistics. If you find them interesting I would suggest checking the library tm, this includes functions designed for this task. There are many applications

of text mining, a pretty popular one is the ability to associate a text with his or her author, this was how J.K.Rowling (Harry potter author) was caught publishing a new novel series under an alias. Before proceeding, it might be helpful to look over the help pages for the nchar, tolower, toupper, grep, sub and strsplit. Take at the library stringr and the functions it includes such as str_sub.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Before starting the set of exercises run the following code lines :

```
if (!'tm' %in% installed.packages()) install.packages('tm')
library(tm)
txt = system.file("texts", "txt", package = "tm")
ovid = VCorpus(DirSource(txt, encoding = "UTF-8"),
readerControl = list(language = "lat"))
OVID = c(data.frame(text=unlist(TEXT), stringsAsFactors = F))
TEXT = lapply(ovid[1:5], as.character)
TEXT1 = TEXT[[4]]
```

Exercise 1

Delete all the punctuation marks from TEXT1

Exercise 2

How many letters does TEXT1 contains?

Exercise 3

How many words does TEXT1 contains?

Exercise 4

What is the most common word in TEXT1?

Learn more about Text analysis in the online course Text Analytics/Text Mining Using R. In this course you will learn how create, analyse and finally visualize your text based data source. Having all the steps easily outlined will be a great reference source for future work.

Exercise 5

Get an object that contains all the words with at least one capital letter (Make sure the object contains each word only once)

Exercise 6

Which are the 5 most common letter in the object OVID?

Exercise 7

Which letters from the alphabet are not in the object OVID

Exercise 8

On the OVID object, there is a character from the popular sitcom 'FRIENDS' , Who is he/she? There were six main characters (Chandler, Phoebe, Ross, Monica, Joey, Rachel)

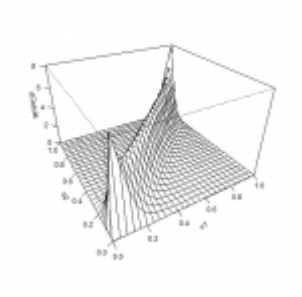
Exercise 9

Find the line where this character is mentioned

Exercise 10

How many words finish with a vowel, how many with a consonant?

Introduction to copulas Exercises (Part-2)



Copulas are a powerful statistical tool commonly used in the finance sector to generate samples from a given multivariate joint distribution.
The principal advantage of using those types of function over other methods is that copulas describe the multivariate joint distribution as his margin and the dependence structure between them,
which give the user the power to fine tune his model component by component.
For example, if you have two independent variables of known distribution X_1 and X_2 which interact to create a dependant variable Y you can set X_1 and X_2 as the margin of the distribution of Y and find the appropriate copula to simulate the interaction between the margins and fit the data.

In the previous post we've seen how to create a copula object and how to generate sample with the most commonly used copula.
In this post we'll learn how to use choose a copula that fit your data and how to make a rough estimate of the probability of a given event.

To be able to do those exercises, you must have installed the packages ggplot2,fitdistrplus, VineCopula and copula. Also, you can find the dataset we'll use
for this set of exercises here. It's a clean dataset of the daily return of the Apple and Microsoft from
May 2000 to May 2017.

Answers to the exercises are available here.

Exercise 1

We'll start by fitting the margin. First, do a histogram of both Apple and Microsoft returns to see the shape of both distributions.

Exercise 2
Both distributions seems symmetric and have a domain which contain positive and negative values.
Knowing those facts, use the fitdist() function to see how the normal, logistic and Cauchy distribution fit the Apple returns dataset.
Which of those three distributions is best suited to simulate the Apple return dataset and what are the parameter of this distribution?

Exercise 3
Repeat exercise 2 with the Microsoft return.

Exercise 4
Plot the joint distribution of the Apple and Microsoft daily returns. Add the regression line to the plot and compute the correlation of both variables.

Exercise 5
Use the pobs() from the VineCopula() package to compute the pseudo-observations for both returns values, then use the BiCopSelect() function to select the copula
which minimise the AIC on the dataset. Which copula is selected and what are his parameters.

Exercise 6
Use the appropriate function from the VineCopula() package to create a copula object with the parameter computed in the last exercise. Then, do a three dimensional plot and a contour plot

of the copula.

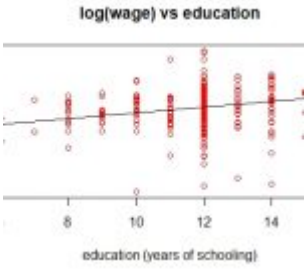
Exercise 7
Set the seed to 42 and generate a sample of 1000 points from this copula. Plot the sample and calculate the correlation of this sample.
Does the correlation of the sample is similar to the correlation between the Apple and Microsoft returns?

Exercise 8
Create a distribution from the copula you selected and the margins you fitted in the exercise 2 and 3.

Exercise 9
Generate 1000 points from the distribution of exercise 8 and plot those points, with the Apple and Microsoft returns, in the same plot.

Exercise 10
Having made a model, let’s make some crude estimation with it! Imagine that this model has been proven to be effective to describe the relation between the apple return and
the Microsoft return for a considerable amound of time and there’s a spike in the price of Apple stock. Suppose you have another model who describe the Apple stock and who
lead you to believe that the daily return on this stock has a 90% chance to be between 0.038 and 0.045. Using only this information, compute the range containing the possible
daily return of the Microsoft stock at the end of the day and the mean of the possible values.

Instrumental Variables in R exercises (Part-1)



One of the most frequently encountered issues in econometrics is endogeneity.
Consider the simple Ordinary Least Squares (OLS) regression setting in which we model wages as a function of years of schooling (education):

$$log(wage_i) = \beta_0 + \beta_1 education_i + \epsilon_i$$

One of the main assumption of OLS is that the independent variables are not correlated with the error term. However, this is seldom the case in most practical applications. The violation of this assumption causes the beta estimates to be biased. In the above example, years of schooling (education) is likely to be correlated with 'individual ability' which is part of the error term. Therefore, estimating this model would give a biased estimate of the beta parameter of education as it is considered endogenous. One option is to obtain a measure of ability such as IQ and have it as an independent variable in the model. Availability of such data is not always possible.

A frequently used approach is the method of instrumental variables: intuitively, we need to identify a variable (called instrumental variable) that is not part of the model, has high correlation with the endogenous variable and is uncorrelated with the error term. In our example, one possible candidate for instrumental variable (IV) is father's education. The

appropriateness of the IV is always debatable and it would be a good thought experiment to see why father's education can fulfill the criteria for an IV we just defined and why it might fail.

The following exercises provide a basic introduction to using instrumental variable technique in R.

Answers to the exercises are available here.

Exercise 1
Load the AER package (package description: here). Next, load PSID1976 dataset provided with the AER package. This has data regarding labor force participation of married women.
The data is sourced from: Mroz, T. A. (1987) The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions. Econometrica 55, 765–799.

Exercise 2
Get summary statistics for the data and identify possible candidates as instrumental variables for education.

Exercise 3
Regress log(wage) on education. Why do you get an error message? (Hint: focus on the participation variable)

Exercise 4
Regress log(wage) on education and state the estimated return to another year of education for women that participated in the labor force. (Hint: use the subset command)

Note: the following exercises are based on the subsample of 428 working women used on exercise-4.

Exercise 5
Plot log(wage) against education along with the fitted regression line from exercise-4.

Exercise 6

Regress education on feducation (father’s years of education) and comment on the estimated correlation.

Exercise 7
The IV estimate of the return to education can be found using the two-stage least squares (2SLS) procedure.
Regress log(wage) on the fitted values of the model estimated in exercise-6.
The beta parameter for the fitted values is the IV estimate for return to education.
Note that the associated standard errors reported are incorrect as these are not based on the usual OLS formula.

Exercise 8
The ivreg function provides a convenient way to estimate IV models and get correct standard errors.
Using the ivreg function, regress log(wage) on education and specify feducation as the IV.
Do you get the same beta estimates as in exercise-6?

Exercise 9
Get 90% confidence intervals for the return to education from exercises 4 and 8.

Exercise 10
Repeat exercises 6 and 8 using mother’s education (meducation) as the IV and comment on the results.

Call Center Productivity Boosting with ML Exercises

The telephone had rung when Jean was watching her favorite TV Show. It was a call center selling newspaper, so she got really upset. This situation is not unpleasant just for Jean. The call center is losing too! By calling a person the will never buy whatever is been sold, the call center is wasting money. Modern machine learning algorithms can help predicting who will be a buyer before the agent pick up the phone. This exercise will teach how to do this task using R.

Answers to the exercises are available here.

Exercise 1
Load libraries randomForest, ggplot2, and caret.

Exercise 2
Download bank-full.csv from This website.

Exercise 3
Take a look at this data set using head function. Read the data dictionary to understand all variables.

Exercise 4
Compare age, housing and loan using ggplot boxplots to find out any relations with y variable.

Exercise 5
Compare day, marital and loan using ggplot boxplots to find out any relations with y variable.

Exercise 6
Make a data partition in order to separate training and testing sets. Reserve 30% of all data for testing procedures.

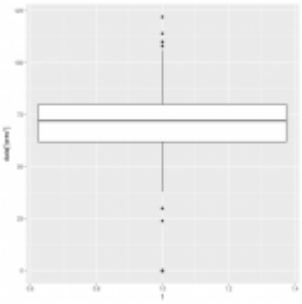
Exercise 7
Create a prediction model using random forest algorithm. To make this experiment reproducible set seed equals to 1234.

Exercise 8
Predict values for the testing set, and take a look at those values using head function.

Exercise 9
Figure out how many trees were create using this algorithm and the estimate error rate. Create a confusion matrix using the testing versus predicted data using the table function. Why this is different from the Confusion matrix stated in the model description?

Exercise 10
Consider that you are making 100 calls to make a single sale. How many calls you will need now using this machine learning algorithm?

Shiny Application Layouts Exercises (Part-8)



Shiny Application Layouts – Dynamic UI

In the eightth part of our series we will see how we can build a user interface with dynamically generated components.

The UI components are generated on the server side with the use of renderUI() and are displayed with uiOutput() on the ui side. Every time a new command is sent by the user, it replaces the previous command.

This part can be useful for you in two ways.

First of all, you can see different ways to enhance the appearance and the utility of your shiny app.

Secondly you can make a revision on what you learnt in "Building Shiny App" series as we will build basic shiny staff in order to present it in the proper way.

Read the examples below to understand the logic of what we are going to do and then test yous skills with the exercise set we prepared for you. Lets begin!

Answers to the exercises are available here.

Application Contextt

First of all let's build the skeleton of our app like the example below.

```
#ui.R
fluidPage(
titlePanel("Title"),
fluidRow(

column(4,wellPanel()),
```

```
column(4,wellPanel()),
column(4,wellPanel())))
#server.R
function(input, output) {}
```

Exercise 1

Create the initial fluid Page with a title. **HINT**: Use fluidPage() and titlePanel().

Exercise 2

Create a row, separate it with columns of size = 3 and add a well Panel in each one of them. **HINT**: Use fluidRow(), column() and wellPanel().

Below is an example of a selectInput() from which the user is going to choose the dynamic component he wants.

```
#ui.r
selectInput("input_type", "Input type",
c("slider", "text", "numeric", "checkbox",
"checkboxGroup", "radioButtons", "selectInput",
"selectInput (multi)", "date", "daterange"
)
)
```

Exercise 3

Add a selectInput() in the first well Panel with: slider input, text input, numeric input, check box, radio buttons and date.

The uiOutput() generates the dynamic UI component.

```
#ui.R
uiOutput("ui")
```

Exercise 4

In the second well Panel add the function that is going to generate your ui output.

In the third well Panel we are going to create two “boxes”. The first one will display the type of the input that the user chooses every time while the second the value of this input. Both of them will be displayed as text. So we have to use verbatimTextOutput() and use tags for the titles. Look at the ui side example below:

```
#ui.R
tags$p("Title 1:"),
verbatimTextOutput("input_title_1"),
tags$p("Title 2:"),
verbatimTextOutput("input_title_2")
```

Exercise 5

In the third well panel should automatically be typed the type of the input of your choice as long as the value of this input. Create only the ui side. **HINT**: Use verbatimTextOutput().

Exercise 6

Name the two components that you just created. **HINT**: Use tags.

Reactivity

Now let’s go to the server side to make things reactive like this:

```
#server.R
output$ui <- renderUI({})
```

Exercise 7

Put your app in a reactive context. **HINT**: Use renderUI().

Now we have to connect the dynamic components we created in **Exercise 3** with their server side. Below is an example:

```
#server.R
"slider" = sliderInput("dynamic", "Dynamic",
min = 1, max = 100, value = 50),
"text" = textInput("dynamic", "Dynamic",
value = "EXAMPLE"),
"numeric" = numericInput("dynamic", "Dynamic",
value = 10),
"checkbox" = checkboxInput("dynamic", "Dynamic",
value = TRUE),
"radioButtons" = radioButtons("dynamic", "Dynamic",
choices = c("Option 1" = "option1",
"Option 2" = "option2"),
selected = "option1"
),
"date" = dateInput("dynamic", "Dynamic"))
```

Exercise 8

Create the server side of the dynamic components you created in **Exercise 3**. Put values of your choise but make sure they are connected with the ui side.

Exercise 9

Display the name of the dynamic component of your choice. **HINT**: Use renderText().

Exercise 10

Display the value of the dynamic component that you chose. **HINT**: Use renderPrint().