

*범용 AI-AutoReporting 개발 매뉴얼

목 차

1. 언어 및 실행 환경 설치
2. 프로그램 폴더 구조
3. 자동화 프로그램 활용 환경(Rstudio) 실행
4. 자동화 프로그램 실행
5. Code

1. 언어 및 실행 환경 설치

▣ 프래그램 언어

- R
- 오픈소스
- 다운로드 사이트 : <https://cran.r-project.org/bin/windows/base/>

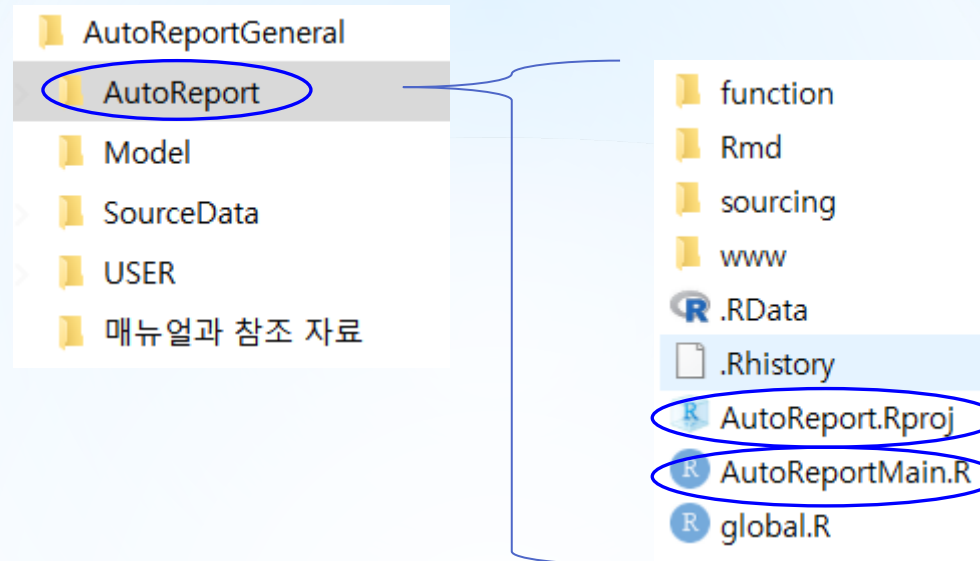
▣ 통합 개발 및 실행 환경

- RStudio
- 오픈소스
- 다운로드 사이트 : <https://www.rstudio.com/products/rstudio/download/>

▣ 배포 USB의 “download” 폴더에 윈도우용 상기 2개의 설치 파일 들어 있음

2. 자동화 프로그램 폴더 구조

▣ 폴더 구조



- **AutoReportGeneral** : 최상위 폴더로 다섯 개의 하위 폴더 보유
 - **AutoReport** : 자동화 프로그램 코드 등을 저장하는 폴더
 - **AutoReport.Rproj** 파일 : Rstudio 개발 및 실행 환경 구축용 파일
 - **AutoReportMain.R** 파일: 자동화 프로그램의 메인 파일
 - **Model**: 예측 모델이 저장되는 폴더
 - **SourceData**: 맞춤용 데이터 파일이 저장되는 폴더
 - **USER**: 범용 데이터 파일이나, 리포트 파일이 저장되는 폴더
 - **매뉴얼과 참조자료** : 매뉴얼과 참조자료가 저장되는 폴더

3. 자동화 프로그램 활용 환경(Rstudio) 실행

▣ 활용 환경 실행

● AutoReport.Rproj 파일을 더블클릭 → 활용 환경 RStudio

The screenshot shows the RStudio interface with the 'AutoReport - RStudio' window. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and running the application. The source editor displays the 'AutoReportMain.R' script, which defines a Shiny application with a navbar and several panels. The console shows the R startup message and an error: 'Error in loadNamespace(name) : there is no package called 'unknown''. The file explorer on the right shows the project structure, including files like .RData, .Rhistory, AutoReport.Rproj, AutoReportMain.R, function, global.R, Rmd, sourcing, and www.

```
1 # options(error=browser)
2 options(error=NULL)
3 source("global.R", encoding="UTF-8")
4
5 ui <- navbarPage( "Auto-Report", theme=shinytheme("cerulean"), # United
6   useShinyjs(), # Set up shinyjs
7   tabPanel("Sourcing",
8     sourcingMainUI()),
9   tabPanel(paste0("Sampling"),
10     samplingMainUI()
11   )
12   # |
13   # tabPanel(paste0("Explore"),
14   #   ExploreFuncMainUI()
15   # ),
16   # # tabPanel("Modify"),
17
18   # tabPanel("Model",
19   #   NewModelFuncMainUI()
20   # )
21
22   # tabPanel("Verify",
23   #   tabsetPanel(type="tabs",
24   #     tabPanel("Top2A", h3("this is Top2A panel")),
25   #     tabPanel("Top2B", h3("this is Top2B panel")),
26   #     tabPanel("Top2C", h3("this is Top2C panel"))
27   #   )
28   # )
29 }
```

Environment: Global Environment

Environment is empty

Console:

```
C:/R/AutoReportEnv/AutoReport - RStudio
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Error in loadNamespace(name) : there is no package called 'unknown'
>
```

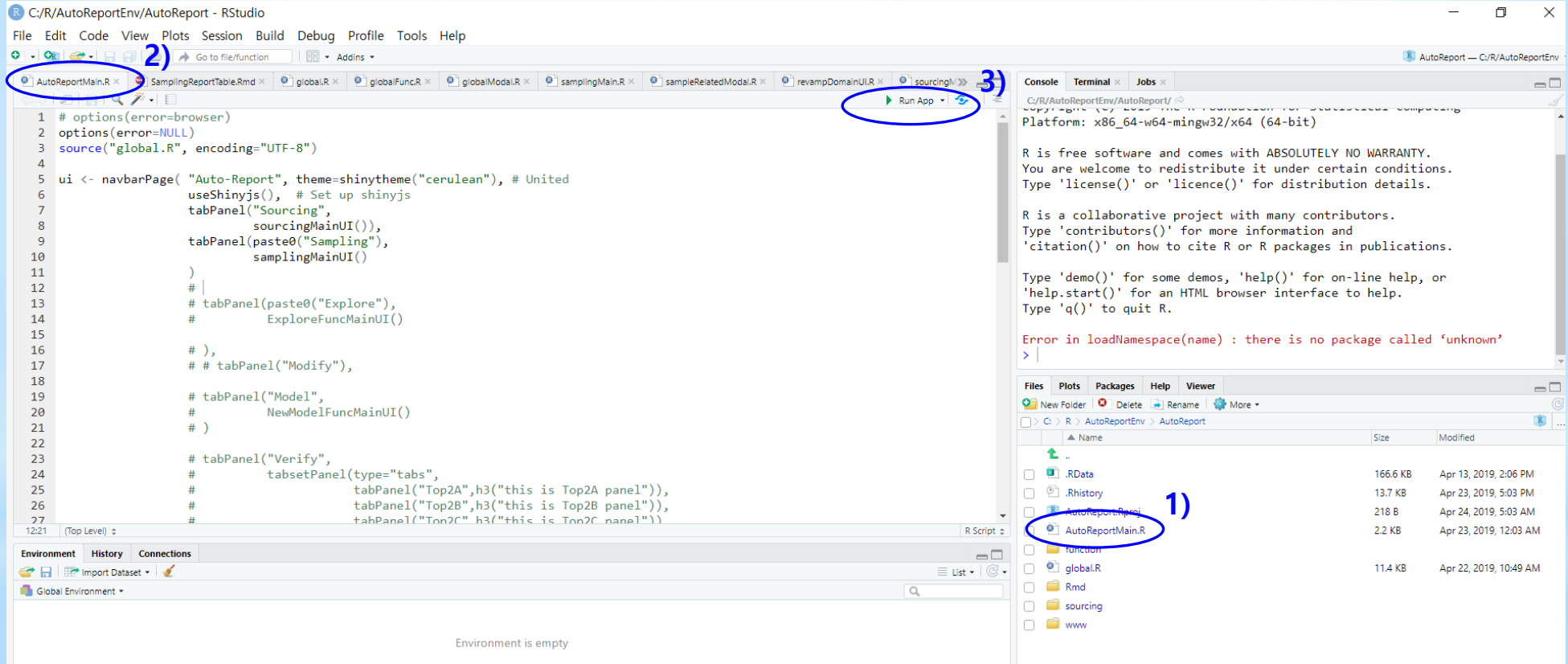
Files:

Name	Size	Modified
..		
.RData	166.6 KB	Apr 13, 2019, 2:06 PM
.Rhistory	13.7 KB	Apr 23, 2019, 5:03 PM
AutoReport.Rproj	218 B	Apr 24, 2019, 5:03 AM
AutoReportMain.R	2.2 KB	Apr 23, 2019, 12:03 AM
function		
global.R	11.4 KB	Apr 22, 2019, 10:49 AM
Rmd		
sourcing		
www		

4. 자동화 프로그램 실행

1) 4사분면의 AutoReportMain.R을 클릭

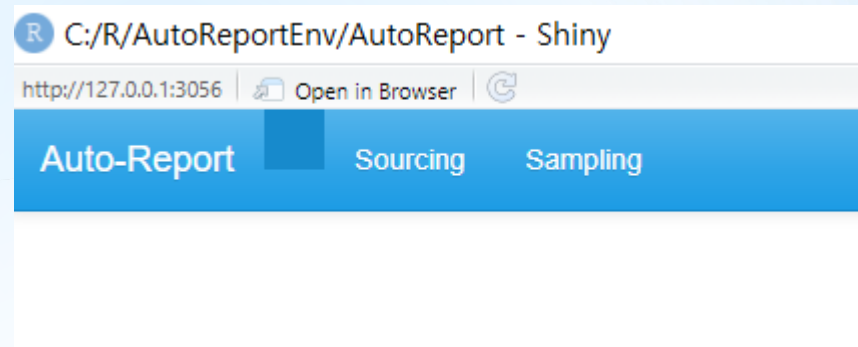
2) 2사분면의 코드 입력창에 AutoReportMain.R 파일이 보임



3) 코드 입력창의 Run App 버튼을 클릭하면 자동화 프로그램 실행 : 뒷페이지 프로그램 화면 참조

4. 자동화 프로그램 실행

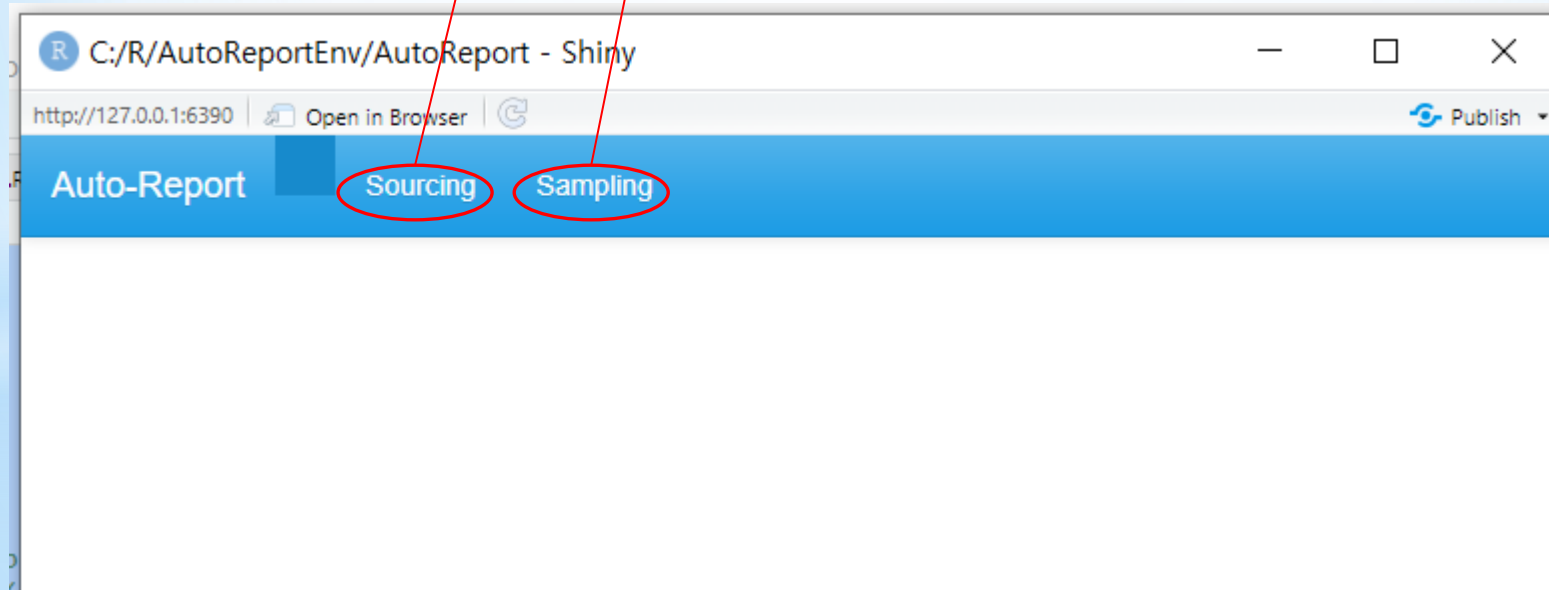
4) 실행된 자동화 프로그램 화면



5-1. 메인 메뉴용 Code

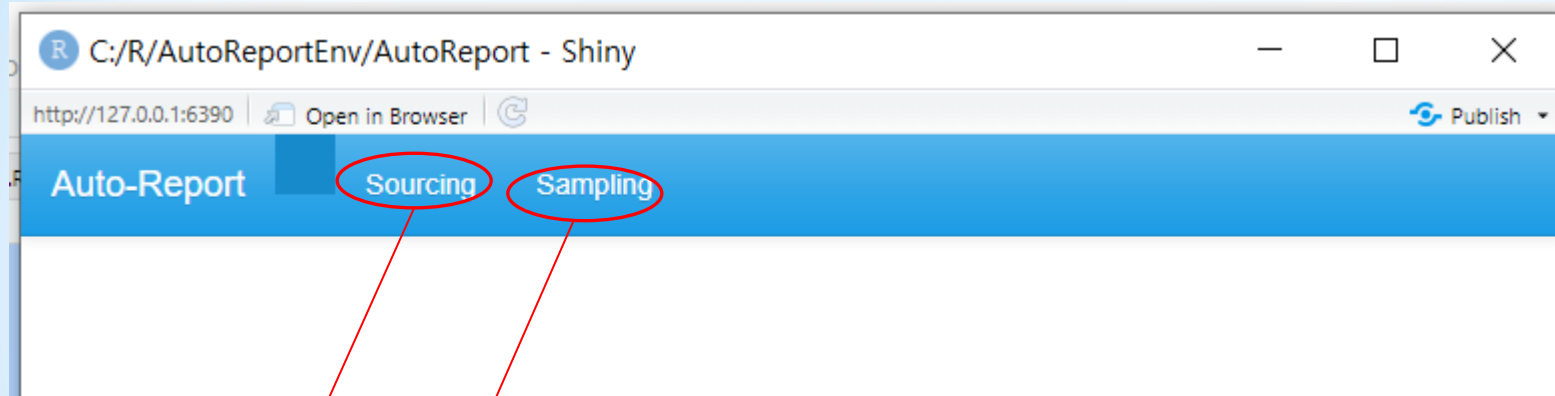
▣ 메인 메뉴 생성 Code

```
ui <- navbarPage( "Auto-Report", theme=shinytheme("cerulean"), # United
  useShinyjs(), # Set up shinyjs
  tabPanel("Sourcing",
    sourcingMainUI()),
  tabPanel(paste0("Sampling"),
    samplingMainUI()
  )
)
```



5-1. 메인 메뉴용 Code

▣ 메인 메뉴 선택시 실행 Code



```
server <- function(input, output, session) {  
  output$sum1 <- renderPrint({  
    summary(cars)  
  })  
  sourcingMain(input, output, session)  
  samplingMain(input, output, session)  
}
```


5-2. 서브 메뉴용 Code

■ Sourcing용 서브메뉴 생성 Code

```
sourcingMainUI <- function() {  
  tabsetPanel( type="tabs",  
    tabPanel("SourcingSub1",  
      fluidPage(  
        fluidRow(  
          column(2, actionButton("renderReportSourcing", "리포트" ) ),  
          includeCSS("www/html/globalModal.css"),  
          column(2, actionButton("renderReportCommonSource", "범용 리포트" ) ),  
          column(6, tags$h4(":::")) ),  
          column(2, actionButton("treatVar", "변수 전처리" ) )  
        ),  
        tags$hr(),  
        fluidRow(  
          column(3,  
            switch(sourcingCat,  
              RExample = {radioButtons("source", "원천 데이터",  
                c("empty","수입세 환급"="importTaxRefund","mtcars", "diamonds","범용"="EXCEL", "clipboard" )))}  
            ),  
          column(9, erbatimTextOutput("strDFsource"))  
        )  
      )  
    )  
  )  
}
```



5-2. 서브 메뉴용 Code

■ Sourcing용 서브메뉴 생성 Code

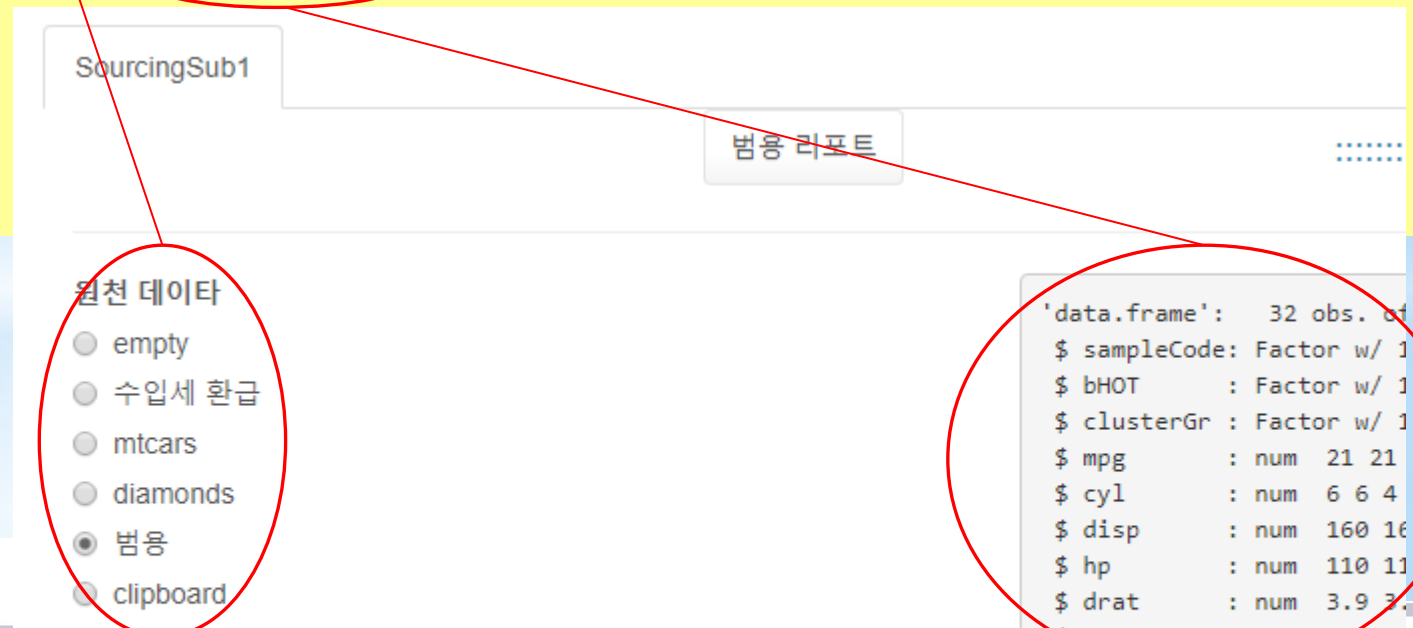
```
sourcingMainUI <- function() {  
  tabsetPanel( type="tabs",  
    tabPanel("SourcingSub1",  
      fluidPage(  
        fluidRow(  
          column(2, actionButton("renderReportSourcing", "리포트" )),  
          includeCSS("www/html/globalModal.css"),  
          column(2, actionButton("renderReportCommonSource", "범용 리포트" )),  
          column(6, tags$h4(":::")) ,  
          column(2, actionButton("treatVar", "변수 전처리") )  
        ),  
        ~~~~~  
      )  
    )  
  }
```



5-2. 서브 메뉴용 Code

■ Sourcing용 서브메뉴 생성 Code

```
sourcingMainUI <- function() {  
  tabsetPanel( type="tabs",  
    tabPanel("SourcingSub1",  
      fluidPage(  
        ~~~~~  
        fluidRow(  
          column(3,  
            switch(sourcingCat,  
              RExample = {radioButtons("source", "원천 데이터",  
                c("empty", "수입세 환급"="importTaxRefund", "mtcars", "diamonds", "범용"="EXCEL", "clipboard" )})  
            },  
          column(9, verbatimTextOutput("strDFsource"))  
        )  
      )  
    )  
  )  
}
```



5-3. 원천데이터 Sourcing 대화창용 Code

■ 원천데이터 라디오버튼 클릭 대응 Code

```
output$strDFsource <- renderPrint({
  chosenDFSourceFile <- ""
  show("renderReportSourcing")
  hide("treatVar")

  switch(input$source,
    importTaxRefund = {
      source("sourcing/importTaxRefund/loadSource.R", encoding="UTF-8")
      loadSource()
    },
    mtcars = {
      source("sourcing/mtcars/loadSource.R", encoding="UTF-8")
      loadSource()
    },
    diamonds = {
      source("sourcing/diamonds/loadSource.R", encoding="UTF-8")
      loadSource()
    },
    EXCEL = {
      show("treatVar")
      source("sourcing/EXCEL/loadSource.R", encoding="UTF-8")
      loadSource()
    },
    ~~~~~
```

5-3. 원천데이터 Sourcing 대화창용 Code

■ 원천데이터 라디오버튼 클릭 대응 Code

~~~~~

```
clipboard = {  
  DFSsource <- read.delim("clipboard", stringsAsFactors = T)  
  # 빈 InsertReqInputUI 함수 정의  
  source("sourcing/clipboard/function/InsertReqInputUI.R", encoding="UTF-8")  
},  
empty = {  
  return()  
},  
{  
  
  )  
  bHOT <- rep("Normal", NROW(DFSsource))  
  clusterGr <- rep("1", NROW(DFSsource))  
  sampleCode <- rep("remains", NROW(DFSsource))  
  DFSsource <- cbind(sampleCode, bHOT, clusterGr, DFSsource )  
  reactDFSsource(DFSsource)  
  curSampleExplore <- DFSsource  
  str(DFSsource)  
  
})
```

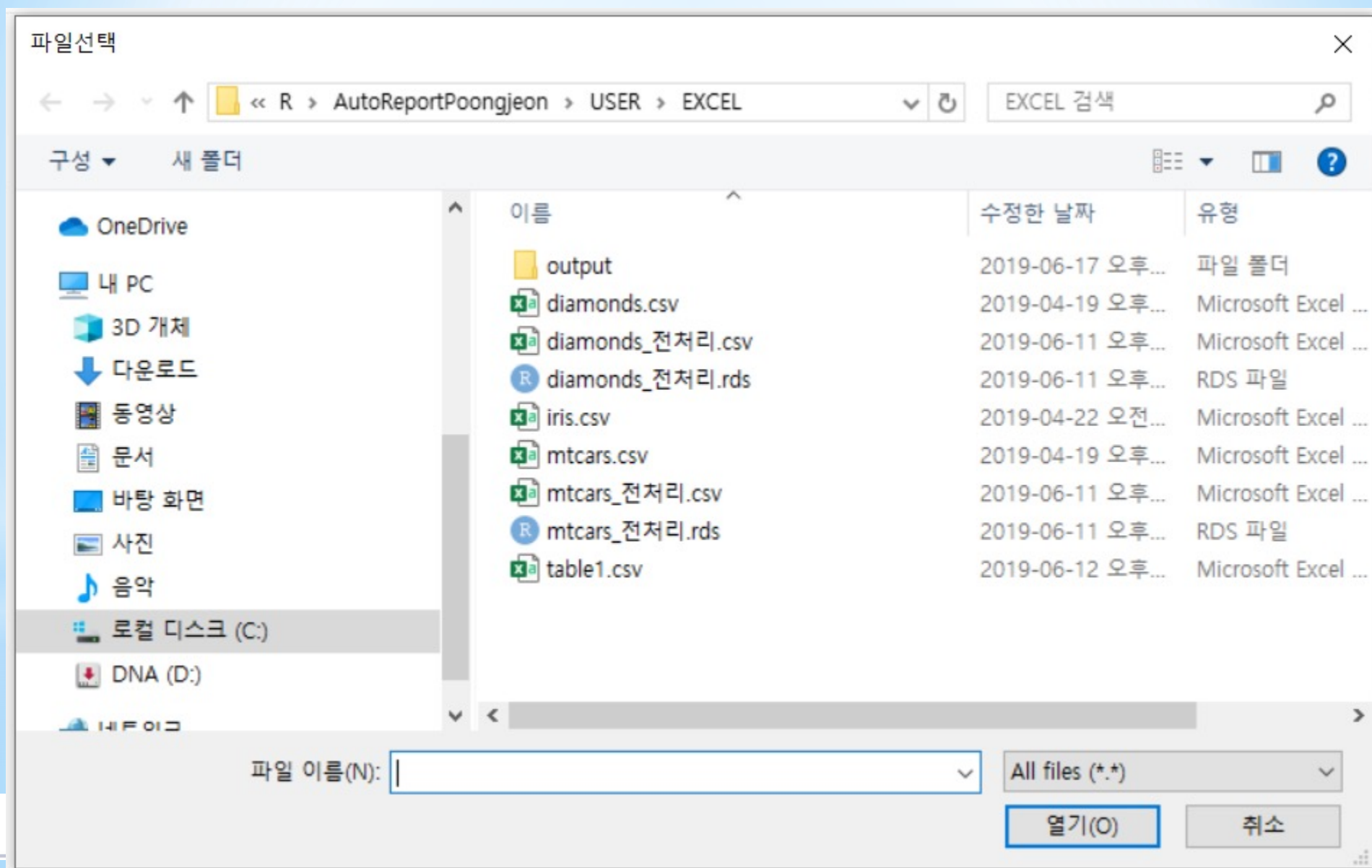
## 5-3. 원천데이터 Sourcing 대화창용 Code

### ■ 범용데이터 선정 대화창 생성 Code

```
loadSource <- function() {  
  projectWD <- getwd()  
  setwd("../")  
  dirPath <- paste0(getwd(), "/USER/EXCEL")  
  setwd(dirPath)  
  # Tell R to sleep until the current directory matches the expected directory  
  while(getwd() != normalizePath(dirPath, winslash="/")) {  
    Sys.sleep(0.02)  
  }  
  filePath <- file.choose()  
  
  chosenDFSsourceFileExt <- str_split(filePath, "WWW")[[1]][length( str_split(filePath, "WWW")[[1]])]  
  chosenDFSsourceFile <- str_split(chosenDFSsourceFileExt, "WW.")[[1]][1]  
  chosenDFSsourceExt <- str_split(chosenDFSsourceFileExt, "WW.")[[1]][2]  
  
  setwd(".././AutoReport")  
  switch(chosenDFSsourceExt,  
    csv = {  
      DFSsource <- read_csv(filePath)  
    },  
    rds = {  
      DFSsource <- read_rds(filePath)  
    }  
  )  
  ~~~~~
```

## 5-3. 원천데이터 Sourcing 대화창용 Code

### ■ 범용데이터 선정 대화창 생성 Code







# (첨부 3) Auto-Reporting 주요 Code

## ▣ Radio Button 대화창 생성 Code

```
showModal(ModalRadioButtons(numVar, numVar, "okMCP", "Y 변수", failed = FALSE))
```

The screenshot shows a dialog box titled "Y 변수" (Y variable). It contains a horizontal row of radio buttons for selecting a variable: mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, and carb. The "mpg" option is selected. A red oval highlights the entire row of radio buttons. A blue oval highlights the title "Y 변수". A red line connects the "numVar" parameter in the code above to the "Y 변수" title. A "Next" button is located at the bottom right of the dialog box.

# (첨부 3) Auto-Reporting 주요 Code

## ■ Checkbox 대화창 생성 Code

```
showModal(ModalCheckboxGroup(title=curSelCatVar,
modalCheckboxID="ModalSelCatVarExplore", label="범주 선정",
choiceNames=choiceNames, choiceValues=choiceValues,
modalOKButtonID="okModalSelCatVarExplore"))
```

범주형 변수

cut

color

clarity

cut

범주 선정

☐ ALL ☐ Fair ☐ Good ☐ Ideal ☐ Premium ☐ Very Good

취소

OK

clarity

범주 선정

☐ ALL ☐ I1 ☐ IF ☐ SI1 ☐ SI2 ☐ VS1 ☐ VS2 ☐ VVS1 ☐ VVS2

취소

OK

# (첨부 3) Auto-Reporting 주요 Code

## ▣ 리포트 생성 R Code

```
params <- list(df=dfReportCommon)
outputFileName <- paste0("commonDescriptiveReport", fromReportCommon, chosenDFSsourceFile, ".html")
rmarkdown::render("Rmd/commonDescriptiveReport.Rmd", output_file = outputFileName,
 output_dir = pathHTMLReport,
 params = params,
 envir = new.env(parent = globalenv()))
```

리포트를 생성하는 Rmarkdown Code

생성된 리포트 파일명

(예) commonDescriptiveReport\_source\_mtcars.html  
리포트 종류\_분석 단계\_데이터파일명

# (첨부 3) Auto-Reporting 주요 Code

## ▣ 리포트 생성 Rmarkdown Code

```
Scatter1 {data-navmenu="Scatter Plot"}
=====

Row {data-height=500}

###
```{r, scatter_1}
renderScatterPlot <- function(x, y, color=NULL, size=NULL, shape=NULL) {
  ~~~~~
}
if(!is.na(x=aesList[["x"]][1]))
  renderScatterPlot(x=aesList[["x"]][1], y=aesList[["y"]], color=aesList[["color"]],
                    size=aesList[["size"]], shape=aesList[["shape"]])
...

###
```{r, scatter_2}
if(!is.na(x=aesList[["x"]][2]))
 renderScatterPlot(x=aesList[["x"]][2], y=aesList[["y"]], color=aesList[["color"]],
 size=aesList[["size"]], shape=aesList[["shape"]])
...

```

# (첨부 3) Auto-Reporting 주요 Code

## ▣ 리포트 생성 Rmarkdown Code

```
Row {data-height=500}
```

```

```

```
###
```

```
```${r, scatter_3}
```

```
if(!is.na(x=aesList[["x"]][3]))
```

```
  renderScatterPlot(x=aesList[["x"]][3], y=aesList[["y"]], color=aesList[["color"]],  
                    size=aesList[["size"]], shape=aesList[["shape"]])
```

```
```\n
```

```
###
```

```
```${r, scatter_4}
```

```
if(!is.na(x=aesList[["x"]][4]))
```

```
  renderScatterPlot(x=aesList[["x"]][4], y=aesList[["y"]], color=aesList[["color"]],  
                    size=aesList[["size"]], shape=aesList[["shape"]])
```

```
```\n
```

# (첨부 3) Auto-Reporting 주요 Code

## ■ 리포트 페이지

