Chapter 1

환경설정 및 유틸리티

이번 챕터에서는 R을 사용하는데 있어 알아두면 도움이 되는 유틸리티에 대해서 알아봅니다.

1.1 파일과 스크립트 관리

R에서 콘솔로 작업을 하는데 프로그램의 크기가 늘어난다면 프로그램의 작성에 매우 어려움을 느끼게 될 것입니다. 따라서, R에서 제공하는 스크립트 에디터가 아닌 외부 텍스트에디터를 이용하여 스크립트를 작성 뒤 실행하면 좋을 것입니다.

초기 메시지 없이 R 실행하기 먼저 아래와 같이 R을 실행시킵니다. -quiet라는 옵션을 주게 되면 처음에 출력되는 시작 메시지를 표시하지 않습니다.

```
gnustats@CHL072:~/Desktop$ R --quiet
>
```

현재 위치 확인하기: 일단은 내가 현재 어디에 있는지 알아야 할 것입니다.

```
> getwd()
[1] "/home/gnustats/Desktop"
>
```

작업디렉토리 만들기: 현재 나의 바탕화면에서 재밌는 것들이 많이 놓여 있기 때문에 작업에 혼돈을 주지 않기 위해 testR이라는 작업디렉토리를 생성할 것입니다.

```
> dir.create("testR")
```

작업디렉토리 지정하기 그리고 난 다음에 실제로 작업을 위해서 작업디렉토리로 들어갑니다. 그리고 나서 제대로 들어왔는지 확인해 봅니다.

```
> setwd("testR")
> getwd()
[1] "/home/gnustats/Desktop/testR"
>
```

작업디렉토리 내 파일확인하기: 방금 막 만들어 놓은 작업디렉토리이기 때문에 아무 파일도 존재하지 않습니다. 이를 list.files()라는 함수를 이용하여 확인합니다.

```
> list.files()
character(0)
```

스크립트 생성하기 이제 스크립트를 작성합니다.

```
> file.create("myRcode.R")
[1] TRUE
> list.files()
[1] "myRcode.R"
```

스크립트 확인하기 이렇게 myRcode.R 이라는 파일만 생성을 했지 프로그램에는 아무런 내용이 없을 것입니다. 이를 확인하기 위해서는 file.show() 함수를 사용합니다.

```
> file.show("myRcode.R")
>
```

아무런 내용이 없으므로 아무것도 없이 END 라는 메시지만 보일 것입니다.

에디터 설정하기: 그런데, 내가 파일을 무슨 에디터를 이용하여 작성하는지를 모릅니다. 그래서 확인해 보도록 하겠습니다.

```
> options()$editor
[1] "vi"
```

그런데 나는 vi 라는 에디터를 싫어합니다. 그 이유는 나는 emacs 또는 nano 를 사용하기 때문입니다. 그래픽 에디터로는 geany 또는 gedit 을 사용합니다. 그래서 바꾸고 싶습니다.

```
> options(editor="nano")
> options()$editor
[1] "nano"
>
```

이렇게 바뀜을 확인할 수 있었습니다.

파일 편집 후 콘솔 내에서 실행하기 그래서 이제 아까 생성한 myRcode.R을 편집하도록 하겠습니다.

```
> file.edit("myRcode.R")
```

이렇게 했더니 커서가 깜빡이면서 nano가 열리면서 스크립트를 작성할 준비가 되었습니다. 그래서 아래와 같이 입력하고 저장하고 파일에디터를 닫았습니다.

```
print("Hello R!  This is my first R program")
```

이제 저정한 스크립트를 실행하기 위해서 아래와 같이 source() 함수를 사용합니다.

```
> source("myRcode.R")
[1] "Hello R!  This is my first R program"
>
```

파일 복사하기: 이제 이 스크립트를 보관하기 위해서 복사를 해 두겠습니다.

```
> list.files()
[1] "myRcode.R"
> file.copy("myRcode.R", "myRcode.copied.R")
[1] TRUE
> list.files()
[1] "myRcode.copied.R" "myRcode.R"
>
```

아하! 잘 복사가 되었음을 확인하였습니다.

파일 삭제하기: 그런데, 생각해 보니 이 파일을 대체 왜 복사했는지 모르겠습니다. 그래서 지워보고 싶습니다.

```
> file.remove("myRcode.copied.R")
[1] TRUE
> list.files()
[1] "myRcode.R"
>
```

이제 다시 파일을 아래와 같이 수정해 봅니다.

```
print("Hello R!  This is my first R program")
print("안녕 난 R 이야!")
x <- 3
y <- 2
cat(x, "+", y, "=", x+y, "\n")
```

스크립트를 터미널에서 실행하기: 먼저 R을 종료하고 testR이라는 작업디렉토리에 들어갑니다.

```
> q()
Save workspace image? [y/n/c]: n
gnustats@CHL072:~/Desktop$ cd testR
gnustats@CHL072:~/Desktop/testR$ ls
myRcode.R
```

myRcode.R 파일이 작업디렉토리에 있는 것을 확인한 뒤에, 아래와 같이 명령어를 입력합니다.

```
gnustats@CHL072:~/Desktop/testR$ R CMD BATCH --quiet myRcode.R
gnustats@CHL072:~/Desktop/testR$ ls
myRcode.R  myRcode.Rout
```

아하! 새로운 myRcode.Rout 이라는 파일이 새로이 생긴 것을 알 수 있으며 내용을 확인해 보니 아래와 같습니다.

```
gnustats@CHL072:~/Desktop/testR$ cat myRcode.Rout
> print("Hello R!  This is my first R program")
[1] "Hello R!  This is my first R program"
> print("안녕 난 R 이야!")
[1] "안녕 난 R 이야!"
> x <- 3
> y <- 2
> cat(x, "+", y, "=", x+y, "\n")
3 + 2 = 5
>
> proc.time()
   user  system elapsed
  0.208   0.016   0.206
gnustats@CHL072:~/Desktop/testR$
```

프로그램의 결과가 프로그램이 수행된 시간과 함께 myRcode.Rout 이라는 파일에 저장되었음을 알 수 있습니다. -slave라는 옵션을 아래와 같이 주게 사용한다면 저장되는 결과물은 아래와 같습니다.

```
gnustats@CHL072:~/Desktop/testR$ R CMD BATCH --quiet --slave myRcode.R
gnustats@CHL072:~/Desktop/testR$ cat myRcode.Rout
[1] "Hello R!  This is my first R program"
[1] "안녕 난 R 이야!"
3 + 2 = 5
> proc.time()
   user  system elapsed
  0.224   0.012   0.224
gnustats@CHL072:~/Desktop/testR$
```

쉘 프로그램으로 실행하기: 쉘프로그램을 작성하시는 분들은 다음과 같이 R 스크립트를 작성하고 수행할 수 있습니다. myRcode.R 프로그램을 아래와 같이 변경합니다.

```
gnustats@CHL072:~/Desktop/testR$ cat myRcode.R
#!/usr/bin/Rscript --vanilla

print("Hello R!  This is my first R program")
print("안녕 난 R 이야!")
x <- 3
y <- 2
cat(x, "+", y, "=", x+y, "\n")
gnustats@CHL072:~/Desktop/testR$
```

이제 쉘 프로그램과 같이 실행할 수 있습니다.

```
gnustats@CHL072:~/Desktop/testR$ chmod u+x myRcode.R
gnustats@CHL072:~/Desktop/testR$ ./myRcode.R
```

```
[1] "Hello R!  This is my first R program"
[1] "안녕 난 R 이야!"
3 + 2 = 5
```

디렉토리 변경: 이제 다른 작업을 해야하므로 기존 작업과 혼돈하지 않기 위해서 또하나의 디렉토리를 생성해 봅니다. 그렇고 보니, 이런식으로 작업을 하게 된다면 디렉토리를 이동하는것이 중요할 것입니다. 그러나 이것은 위에서 언급한 setwd()이면 충분합니다. 아래의 예제를 살펴보세요.

```
> # 현재 디렉토리 확인
> getwd()
[1] "/home/gnustats/Desktop/testR"
>
> # tmp 디렉토리 생성
> dir.create("tmp")
>
> # tmp 디렉토리로 이동
> setwd("tmp")
>
> # 현재 디렉토리 다시 확인
> getwd()
[1] "/home/gnustats/Desktop/testR/tmp"
>
> # 다시 상위 디렉토리 testR로 이동
> setwd("..")
>
> # 현재 디렉토리 다시 확인
> getwd()
[1] "/home/gnustats/Desktop/testR"
```

1.2 객체관리

객체확인: R에서 생성되고 다루어지는 그 모든 것들을 객체라고 합니다. 객체라는 개념을 보다 정확하게 이해하기 위해서는 클래스라는 개념을 알아야 하는데, 본 파트에서는 R의 사용에 익숙해지는 것이 주목적이므로 다루지 않습니다. 그러나, 객체지향적 프로그래밍을 위한 클래스와 객체, 그리고 메소드라는 개념에 대해서는 "패키지 작성"이라는 챕터에서 다루도록 하겠습니다.

현재 세션내에서 사용되고 있는 모든 객체들의 목록을 확인해 보고 싶을 때는 ls()함수를 이용합니다. R을 시작하면 어떠한 작업도 하지 않았기 때문에 생성된 객체가 없을 것입니다.

```
gnustats@CHL072:~/Desktop/testR$ R --quiet
> ls()
character(0)
```

따라서, ls() 함수를 입력했을 때 아무것도 보여줄 것이 없다는 의미의 character(0) 이라는 것이 메시지가 나옵니다. 이러한 표현은 ls()의 결과값이 문자형이기 때문입니다. (ls는 list의 약어입니다).

이번에는 이전에 작성한 myRcode.R 프로그램을 수행해봅니다. 그리고 나서 어떤 객체가 있는지 살펴봅니다.

```
> source("myRcode.R")
[1] "Hello R!  This is my first R program"
[1] "안녕 난 R 이야!"
3 + 2 = 5
> ls()
[1] "x" "y"
```

아하! 이전 프로그램에서 x와 y라는 변수를 생성하여 값을 대입했었기 때문에 객체가 존재합니다. 따라서 "x"와 "y"라는 객체가 나열됨을 확인할 수 있습니다. 이번에는 z 라는 객체를 생성해 봅니다. 그리고 다시 어떤 객체들이 존재하는지 확인해 봅니다.

```
> z <- 8
> ls()
[1] "x" "y" "z"
```

한개의 객체를 더 생성해 봅니다.

```
> a <- "너도 R이니?  나도 R이야"
> ls()
[1] "a" "x" "y" "z"
>
```

객체삭제: 만약, 작업을 하면서 너무 많은 양의 객체들로 인하여 R의 메모리 사용에 영향이 생길때 객체를 삭제하고 싶을 경우가 있습니다. 이럴때는 rm()이라는 함수를 사용합니다. (rm 은 remove의 약자입니다).

```
> # 현재 세션의 객체들을 확인합니다.
> ls()
[1] "a" "x" "y" "z"
>
> # 객체 x를 삭제해봅니다
> rm("x")
> ls()
[1] "a" "y" "z"
>
> # 객체 x를 삭제한 뒤에 다시 x를 사용하고자 하면 어떻게 될까요?
> x
Error: object 'x' not found
>
```

객체를 확인할때는 ls()를 사용하지만 "객체"라는 말 그대로 objects()라는 함수도 동일한 역할을 수행합니다.

```
> # 현재 세션의 객체들 확인하기
> objects()
[1] "a" "y" "z"
>
> # 이번엔 "y"와 "z"를 한 번에 삭제하기
> rm(list=c("y", "z"))
> objects()
[1] "a"
>
```

현 세션내의 작업공간 (workspace)의 모든 객체들을 삭제하기 위해서는 아래와 같이 할 수 있습니다.

```
> rm(list=ls())
> objects()
character(0)
>
```

위에서 언급한 바와 같이 객체라는 것은 R 세션에서 다루는 모든 것을 의미하기 때문에, 벡터, 행렬, 배열, 데이터프레임, 리스트, 함수 등 어떠한 것이 될 수 있습니다. 그런데, 데이터 프레임은 특히 많이 쓰입니다. 그 이유는 우리가 가장 많이 접하는 엑셀과 같은 스프레드 형식의 데이터 형식이기 때문입니다.

스프레드시트 형식의 뷰어: 예를들어, 아래와 같이 x 라는 데이터 프레임이 있다고 가정합니다.

```
> x <- data.frame(v1=rnorm(5), v2=rnorm(5))
> x
         v1         v2
1 -0.2266056  1.2148930
2 -1.9334847  0.1454115
3 -1.5104792 -0.7163344
4  0.5226286  0.1827389
5 -0.8681059 -0.8193394
```

그런데, 데이터를 보면서 작업을 하는 것이 보다 수월하기 때문에 난 데이터 x를 계속 띄워놓고 싶습니다. 그래서 View()함수를 이용합니다.

```
> View(x)
```

(스크린 캡쳐가 좋을듯 한데...)

View(x)라고 하면 스프레드시트 형식의 데이터 뷰어가 나타나게 됩니다. 이 함수를 이용하여 데이터를 띄워놓은 상태에서 작업하면 참 편리한 경우가 많이 있습니다.

스프레드시트 형식의 데이터 편집기: 그런데 View() 함수는 데이터를 보여주기만 할 뿐, 수정할 수는 없습니다. 그래서 이번에는 수정을 하고 싶습니다. 만약, 3행 2열의 값 -0.7163344 을 0.272 로 변경해 보고자 한다면 edit(x) 함수를 아래와 같이 이용합니다.

```
> x1 <- edit(x)
> x1
          v1         v2
1 -0.2266056  1.2148930
2 -1.9334847  0.1454115
3 -1.5104792  0.2720000
4  0.5226286  0.1827389
5 -0.8681059 -0.8193394
>
```

edit()함수와 동일한 기능을 가진 함수는 fix()입니다. 아래와 같이 해 보세요. 동일한 결과를 얻을 수 있습니다.

```
> x2 <- fix(x)
> x2
          v1         v2
1 -0.2266056  1.2148930
2 -1.9334847  0.1454115
3 -1.5104792  0.2720000
4  0.5226286  0.1827389
5 -0.8681059 -0.8193394
>
```

콘솔 상에서 데이터를 직접 입력하기: 그런데, 아주 작은 크기의 데이터라면 바로 손으로 입력할 수도 있습니다. 이런 경우에는 scan()함수를 이용합니다.

```
> d1 <- scan()
1: 1 2 3
4: 4 5
6: 0 0 0 0 1 3 4 2
14:
Read 13 items
> d1
 [1] 1 2 3 4 5 0 0 0 0 1 3 4 2
>
```

작업공간과 작업내역을 저장하고 불러오기: 가끔 급한 용무때문에 급히 자리를 떠야 할 경우가 있습니다. 이럴 때 작업을 하고 있던 그 모든 것들을 저장하고 나중에 불러올 수 있다면 매우 좋을 것입니다. R은 이런 경우를 대비하여 save()와 load()라는 기능을 제공합니다.

이를 이해하기 위해서 아래와 같이 해봅니다.

```
gnustats@CHL072:~/Desktop/testR$ R --quiet
> ls()
character(0)
```

```
> x <- 3
> y <- 4
> x + y
[1] 7
> ls()
[1] "x" "y"
```

이제 자리를 떠야 해서 아래와 같이 추가로 입력합니다.

```
> save("x", file="하다만거-2013-05-09.RData")
> savehistory(file="하다말고저장-2013-05-09.Rhistory")
> q()
```

그리고, 작업디렉토리를 탐색기를 열어 직접 확인해보세요. "하다만거-2013-05-09.RData"라는 파일과 "하다말고저장-2013-05-09.Rhiostry"라는 파일 두개가 생성되었음을 알 수 있습니다. 이렇게 확인이 되었으면 R을 종료합니다.

이제 다시 R을 실행한 뒤 저장된 것이 어떤 것인지 확인해 봅니다.

```
gnustats@CHL072:~/Desktop/testR$ R --quiet
>
> # 시작했으므로 아무 객체도 없을 것입니다.
> ls()
character(0)
>
> # 과거의 작업기록을 불러옵니다.
> loadhistory(file="하다말고저장-2013-05-09.Rhistory")
>
> # 내용을 확인해 봅니다
>
> history()

## R에 의해 열린 외부 에디터로부터 보여지는 이전 사용의 기록
ls()
rm(list=ls())
q()
ls()
x <- 3
y <- 4
x + y
ls()
save("x", file="하다만거-2013-05-09.RData")
savehistory(file="하다말고저장-2013-05-09.Rhistory")
history()
```

```
(END)
## 편집기를 닫고 원래 R 콘솔로 복귀
>
> # 이전에 작업하고 저장했던 명령어의 기록들이 그대로 보여집니다.
>
> ls()
character(0)
>
> # 그러나, 이것은 명령어만 불러온 것이지 작업하던 객체들을 불러온 것은 아닙니다.
> # 따라서, 작업하던 객체들을 불러옵니다.
>
> load(file="하다만거-2013-05-09.RData")
> ls()
[1] "x"
> x
[1] 3
>
```

1.3 환경설정 및 관리

프로그래밍을 작성할 때 R이 설치되어 있는 환경에 대한 정보를 가지고 있는 것은 R을 보다 안전한 방향으로 쓸 수 있게 해줍니다.

시스템 요약정보: 먼저 설치된 R은 어떻게 환경설정이 되어 있는지에 대한 요약정보를 sessionInfo()함수를 이용하여 확인할 수 있습니다. 즉, R의 버전과 플랫폼, 그리고 지역설정 및 현재 로딩되어 사용이 가능한 패키지의 목록을 볼 수 있습니다. 그러나, sessionInfo()를 통해 얻어지는 내용은 .Platform이라는 리스트, search() 그리고 options()라는 환경설정 관련함수의 값들로부터 단순히 요약된 정보입니다.

```
> sessionInfo()
R version 2.15.1 (2012-06-22)
Platform: i686-pc-linux-gnu (32-bit)

locale:
 [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
 [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

콘솔 출력을 횡방향으로 확장하기 이런것을 굳이 확인할 필요가 있나? 라고 의문을 가지시는 경우가 있으나, 알고 계시면 사용에 많은 편리함을 얻을 수 있습니다. 예를들면 다음과 같습니다.

간혹, 여러개의 변수를 가지고 있는 데이터 프레임을 살펴볼때 출력되는 콘솔에서의 창이 좁아서 아래와 같이 데이터 확인을 불편하게 할 때는 경험하는 적이 많을 것입니다.

```
gnustats@CHL072:~$ R --quiet
> var.names <- paste("VAR", 1:10, sep="")
> mydata <- as.data.frame(t(LETTERS[1:10]))
> names(mydata) <- paste("변수명", 1:10, sep="")
> mydata
  변수명1 변수명2 변수명3 변수명4 변수명5 변수명6 변수명7 변수명8 변수명9
1       A       B       C       D       E       F       G       H       I
  변수명10
1        J
```

그 이유는 아래와 같이 환경설정에서 출력에 관계된 부분의 출력너비가 80으로 제한되어 있기 때문입니다. 여기에서 왜 기본값이 80이라고 물어보지 마시길 부탁드립니다. 여기에서 80은 단순히 GNU 소프트웨어들이 사용하는 텍스트 프로그램의 기본값일뿐입니다.

```
> options()$width
[1] 80
```

이 값을 찾기 위해서 getOption()이라는 함수를 사용할 수 도 있습니다.

```
> getOption("width")
[1] 80
```

그래서 이 기본값을 아래와 같이 좀 더 넓게 변경해보고 출력을 해 봅니다.

```
> options(width="120")
> mydata
  변수명1 변수명2 변수명3 변수명4 변수명5 변수명6 변수명7 변수명8 변수명9 변수명10
1       A       B       C       D       E       F       G       H       I        J
>
```

오! 좀 더 확실히 볼 수 있게 되었습니다.

프롬프트의 모양 변경하기: 이러한 환경설정과 관계된 부분은 또 있습니다. R을 사용하면서 프롬프트의 모양이 단순한 화살표인 것은 아래와 같은 설정 때문입니다.

```
> getOption("prompt")
[1] "> "
>
```

이건 R 프로그램 프롬프트라는 것을 확실히 표기하기 위해서는 아래와 같이 합니다.

```
> options(prompt="R> ")
R>
```

프롬프트가 바뀌어서 이젠 내가 R 콘솔에서 작업을 하고 있구나라는 생각이 확실해집니다.

유효숫자의 개수 변경하기: 이제 아래와 같이 나눗셈을 생각해 봅니다. 각종 수치연산의 결과는 저렇게 소수점 7자리까지 표기가 될 것입니다.

```
R> 5/3
[1] 1.666667
```

그 이유는 설정에 유효숫자 자리표기 (digits)가 7 로 설정되어 있기 때문입니다.

```
> getOption("digits")
[1] 7
```

만약, 독자가 결과를 요약하여 표에 사용될 숫자를 소수점 3자리까지만 하고 싶다면 아래와 같이 할 수 있습니다.

```
R> options(digits="4")
R> 5/3
[1] 1.667
```

만약, 정교한 숫자가 필요한 경우에는 유효숫자의 자리 (최대 22까지)를 늘릴 수 있습니다.

```
R> options(digits="22")
R> 5/3
[1] 1.666666666666666740682
R>
```

기본적으로 사용가능한 패키지 확인: R을 시작하자마자 기본적으로 사용할 수 있는 패키지를 확인하는 것은 아래와 같이 할 수 있습니다. 즉, 어떠한 함수이던지 간에 아래의 패키지들 내에 포함이 되어 있다면 패키지 설치 및 불러오기 과정 없이 사용이 가능합니다.

```
> getOption("defaultPackages")
[1] "datasets"  "utils"     "grDevices" "graphics"  "stats"     "methods"
>
```

패키지내의 함수목록 살펴보기: 방금전에 어떠한 함수이던지 패키지 목록안에만 있으면 사용이 가능하다고 했는데, 독자의 가장 큰 궁금함은 아마도 "대체 해당 패키지 안에 어떤 함수가 머가 있는지???" 라는 것일 것입니다. 패키지안의 모든 함수들의 목록을 살펴보고 싶다면 아래와 같이 하세요. 이러한 탐색기능은 탐색경로를 통해 이루어지므로 아래와 같이 search() 라는 함수를 사용하도록 하세요.

```
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
>
```

여기에서 독자는 아마도 위에서 보여준 결과랑 엇비슷하다는 것을 느끼지만, 다른 점이 패키지명앞에 모두 package: 와 같은 형식을 가지고 있음을 알게 될 거입니다. 이것을 네임스페이스(namespace)를 이용한 표기라고 합니다. (저기있는 .GlbalEnv와 Autoloads는 나중에 패키지 개발 부분에서 설명합니다).

그럼, 이제 내가 stats 라는 패키지에 있는 모든 함수들의 목록을 알고 싶다고 합니다.

```
> ls("package:stats")
  [1] "acf"                   "acf2AR"                "add1"
  [4] "addmargins"            "add.scope"             "aggregate"
  [7] "aggregate.data.frame" "aggregate.default"    "aggregate.ts"
 [10] "AIC"                   "alias"                 "anova"
 [13] "anova.glm"             "anova.glmlist"         "anovalist.lm"
 [16] "anova.lm"              "anova.lmlist"          "anova.mlm"
 [19] "ansari.test"           "aov"                   "approx"
 [22] "approxfun"             "ar"                    "ar.burg"
 [25] "arima"                 "arima0"                "arima0.diag"
 [28] "arima.sim"             "ARMAacf"               "ARMAtoMA"
 [31] "ar.mle"                "ar.ols"                "ar.yw"
 [34] "as.dendrogram"         "as.dist"               "as.formula"
 [37] "as.hclust"             "asOneSidedFormula"     "as.stepfun"
 [40] "as.ts"                 "ave"                   "bandwidth.kernel"
 [43] "bartlett.test"         "BIC"                   "binomial"
 [46] "binom.test"            "biplot"                "Box.test"
 [49] "bw.bcv"                "bw.nrd"                "bw.nrd0"
 [52] "bw.SJ"                 "bw.ucv"                "C"
 [55] "cancor"                "case.names"            "ccf"
 [58] "chisq.test"            "clearNames"            "cmdscale"
 [61] "coef"                  "coefficients"          "complete.cases"
 [64] "confint"               "confint.default"       "constrOptim"
 [67] "contrasts"             "contrasts<-"           "contr.helmert"
 [70] "contr.poly"            "contr.SAS"             "contr.sum"
 [73] "contr.treatment"       "convolve"              "cooks.distance"
 [76] "cophenetic"            "cor"                   "cor.test"
 [79] "cov"                   "cov2cor"               "covratio"
 [82] "cov.wt"                "cpgram"                "cutree"
 [85] "cycle"                 "D"                     "dbeta"
 [88] "dbinom"                "dcauchy"               "dchisq"
 [91] "decompose"             "delete.response"       "deltat"
```

```
 [94] "dendrapply"          "density"           "density.default"
 [97] "deriv"               "deriv3"            "deriv3.default"
[100] "deriv3.formula"      "deriv.default"     "deriv.formula"
[103] "deviance"            "dexp"              "df"
[106] "dfbeta"              "dfbetas"           "dffits"
[109] "df.kernel"           "df.residual"       "dgamma"
[112] "dgeom"               "dhyper"            "diffinv"
[115] "diff.ts"             "dist"              "dlnorm"
[118] "dlogis"              "dmultinom"         "dnbinom"
[121] "dnorm"               "dpois"             "drop1"
[124] "drop.scope"          "drop.terms"        "dsignrank"
[127] "dt"                  "dummy.coef"        "dunif"
[130] "dweibull"            "dwilcox"           "ecdf"
[133] "eff.aovlist"         "effects"           "embed"
[136] "end"                 "estVar"            "expand.model.frame"
[139] "extractAIC"          "factanal"          "factor.scope"
[142] "family"              "fft"               "filter"
[145] "fisher.test"         "fitted"            "fitted.values"
[148] "fivenum"             "fligner.test"      "formula"
[151] "frequency"           "friedman.test"     "ftable"
[154] "Gamma"               "gaussian"          "get_all_vars"
[157] "getCall"             "getInitial"        "glm"
[160] "glm.control"         "glm.fit"           "glm.fit.null"
[163] "hasTsp"              "hat"               "hatvalues"
[166] "hatvalues.lm"        "hclust"            "heatmap"
[169] "HoltWinters"         "influence"         "influence.measures"
[172] "integrate"           "interaction.plot"  "inverse.gaussian"
[175] "IQR"                 "is.empty.model"    "is.leaf"
[178] "is.mts"              "isoreg"            "is.stepfun"
[181] "is.ts"               "is.tskernel"       "KalmanForecast"
[184] "KalmanLike"          "KalmanRun"         "KalmanSmooth"
[187] "kernapply"           "kernel"            "kmeans"
[190] "knots"               "kruskal.test"      "ksmooth"
[193] "ks.test"             "lag"               "lag.plot"
[196] "line"                "lines.ts"          "lm"
[199] "lm.fit"              "lm.fit.null"       "lm.influence"
[202] "lm.wfit"             "lm.wfit.null"      "loadings"
[205] "loess"               "loess.control"     "loess.smooth"
[208] "logLik"              "loglin"            "lowess"
[211] "ls.diag"             "lsfit"             "ls.print"
[214] "mad"                 "mahalanobis"       "makeARIMA"
[217] "make.link"           "makepredictcall"   "manova"
```

```
[220] "mantelhaen.test"       "mauchley.test"         "mauchly.test"
[223] "mcnemar.test"          "median"                "median.default"
[226] "medpolish"             "model.extract"         "model.frame"
[229] "model.frame.aovlist"   "model.frame.default"   "model.frame.glm"
[232] "model.frame.lm"        "model.matrix"          "model.matrix.default"
[235] "model.matrix.lm"       "model.offset"          "model.response"
[238] "model.tables"          "model.weights"         "monthplot"
[241] "mood.test"             "mvfft"                 "na.action"
[244] "na.contiguous"         "na.exclude"            "na.fail"
[247] "na.omit"               "na.pass"               "napredict"
[250] "naprint"               "naresid"               "nextn"
[253] "nlm"                   "nlminb"                "nls"
[256] "nls.control"           "NLSstAsymptotic"       "NLSstClosestX"
[259] "NLSstLfAsymptote"      "NLSstRtAsymptote"      "nobs"
[262] "numericDeriv"          "offset"                "oneway.test"
[265] "optim"                 "optimHess"             "optimise"
[268] "optimize"              "order.dendrogram"      "pacf"
[271] "p.adjust"              "p.adjust.methods"      "pairwise.prop.test"
[274] "pairwise.table"        "pairwise.t.test"       "pairwise.wilcox.test"
[277] "pbeta"                 "pbinom"                "pbirthday"
[280] "pcauchy"               "pchisq"                "pexp"
[283] "pf"                    "pgamma"                "pgeom"
[286] "phyper"                "plclust"               "plnorm"
[289] "plogis"                "plot.density"          "plot.ecdf"
[292] "plot.lm"               "plot.mlm"              "plot.spec"
[295] "plot.spec.coherency"   "plot.spec.phase"      "plot.stepfun"
[298] "plot.ts"               "plot.TukeyHSD"         "pnbinom"
[301] "pnorm"                 "poisson"               "poisson.test"
[304] "poly"                  "polym"                 "power"
[307] "power.anova.test"      "power.prop.test"       "power.t.test"
[310] "ppoints"               "ppois"                 "ppr"
[313] "PP.test"               "prcomp"                "predict"
[316] "predict.glm"           "predict.lm"            "predict.mlm"
[319] "predict.poly"          "preplot"               "princomp"
[322] "print.anova"           "printCoefmat"          "print.coefmat"
[325] "print.density"         "print.family"          "print.formula"
[328] "print.ftable"          "print.glm"             "print.infl"
[331] "print.integrate"       "print.lm"              "print.logLik"
[334] "print.terms"           "print.ts"              "profile"
[337] "proj"                  "promax"                "prop.test"
[340] "prop.trend.test"       "psignrank"             "pt"
[343] "ptukey"                "punif"                 "pweibull"
```

```
[346] "pwilcox"             "qbeta"              "qbinom"
[349] "qbirthday"           "qcauchy"            "qchisq"
[352] "qexp"                "qf"                 "qgamma"
[355] "qgeom"               "qhyper"             "qlnorm"
[358] "qlogis"              "qnbinom"            "qnorm"
[361] "qpois"               "qqline"             "qqnorm"
[364] "qqnorm.default"      "qqplot"             "qsignrank"
[367] "qt"                  "qtukey"             "quade.test"
[370] "quantile"            "quantile.default"   "quasi"
[373] "quasibinomial"       "quasipoisson"       "qunif"
[376] "qweibull"            "qwilcox"            "r2dtable"
[379] "rbeta"               "rbinom"             "rcauchy"
[382] "rchisq"              "read.ftable"        "rect.hclust"
[385] "reformulate"         "relevel"            "reorder"
[388] "replications"        "reshape"            "reshapeLong"
[391] "reshapeWide"         "resid"              "residuals"
[394] "residuals.default"   "residuals.glm"      "residuals.lm"
[397] "rexp"                "rf"                 "rgamma"
[400] "rgeom"               "rhyper"             "rlnorm"
[403] "rlogis"              "rmultinom"          "rnbinom"
[406] "rnorm"               "rpois"              "rsignrank"
[409] "rstandard"           "rstandard.glm"      "rstandard.lm"
[412] "rstudent"            "rstudent.glm"       "rstudent.lm"
[415] "rt"                  "runif"              "runmed"
[418] "rweibull"            "rwilcox"            "rWishart"
[421] "scatter.smooth"      "screeplot"          "sd"
[424] "se.contrast"         "selfStart"          "setNames"
[427] "shapiro.test"        "simulate"           "smooth"
[430] "smoothEnds"          "smooth.spline"      "sortedXyData"
[433] "spec.ar"             "spec.pgram"         "spec.taper"
[436] "spectrum"            "spline"             "splinefun"
[439] "splinefunH"          "SSasymp"            "SSasympOff"
[442] "SSasympOrig"         "SSbiexp"            "SSD"
[445] "SSfol"               "SSfpl"              "SSgompertz"
[448] "SSlogis"             "SSmicmen"           "SSweibull"
[451] "start"               "stat.anova"         "step"
[454] "stepfun"             "stl"                "StructTS"
[457] "summary.aov"         "summary.aovlist"    "summary.glm"
[460] "summary.infl"        "summary.lm"         "summary.manova"
[463] "summary.mlm"         "summary.stepfun"    "supsmu"
[466] "symnum"              "termplot"           "terms"
[469] "terms.aovlist"       "terms.default"      "terms.formula"
```

```
[472] "terms.terms"          "time"              "toeplitz"
[475] "ts"                   "tsdiag"            "ts.intersect"
[478] "tsp"                  "tsp<-"             "ts.plot"
[481] "tsSmooth"             "ts.union"          "t.test"
[484] "TukeyHSD"             "TukeyHSD.aov"      "uniroot"
[487] "update"               "update.default"    "update.formula"
[490] "var"                  "variable.names"    "varimax"
[493] "var.test"             "vcov"              "weighted.mean"
[496] "weighted.residuals"   "weights"           "wilcox.test"
[499] "window"               "window<-"          "write.ftable"
[502] "xtabs"
>
```

이제 사용하는데 있어 점점 익숙해지셨을 것이라고 생각합니다. (이 문서의 지은이는 "통계분석" 챕터에서 저기 나와 있는 모든 함수들이 어떤 역할을 하는지 설명할 것입니다).

파일경로 표시와 운영체제의 정보확인: 많은 윈도우즈 사용자들이 데이터의 위치를 지정하기 위해서 파일경로를 작성시 유닉스에서 사용하는 단일 슬래쉬를 사용한 뒤에 파일을 읽을 수 없다고 불만을 제기합니다. 보다 정확히 말하면 윈도우즈라고 국한하기 보다는 운영체제 별로 파일경로를 나타내는 구분자가 플랫폼마다 다르기 때문입니다. 이는 .Platform 이라는 리스트를 통해 확인이 가능합니다. 아래와 같이 입력해 보세요.

```
> do.call(rbind, .Platform)
           [,1]
OS.type    "unix"
file.sep   "/"
dynlib.ext ".so"
GUI        "X11"
endian     "little"
pkgType    "source"
path.sep   ":"
r_arch     ""
>
```

정보를 읽어보면 내가 사용하고 있는 운영체제는 unix와 같은 환경이고, 파일경로를 표시하는 구분자는 단일 슬래쉬(/)를 사용하는 것입니다. 아하! 파일경로를 표시하는 구분자가 슬래쉬였기 때문에 내 운영체제에 단일 슬래쉬를 사용하여 파일경로를 써도 무방한 것입니다. 그러면 윈도우즈 사용자의 경우에는 아마도 단일 또는 더블 역슬래쉬가 나올 것입니다. 추가적으로 또한 다이나믹라이브러리 확장자는 .so 이며 그래픽 환경은 X11 을 이용하는 것입니다.

1.4 한글과 UTF-8 인코딩:

한글과 인코딩의 간략한 내용을 챕터 1. 시작하면서 에서 한글 표현과 인코딩에 관련하여 라는 부분에서 언급했었습니다. (다시 읽어보시면 도움이 될 것입니다. 한글인터페이스의 사용은 단순히 사용자의

선택에 의한 자유입니다. The Korean R Translation Team 에서 진행하는 여러가지 태스크 중에서 한가지는 이러한 선택적 자유에 대하여 여러분의 편의를 제공하는 것입니다).

인코딩이란 쉽게 말해서 어떤 언어를 표시하기 위한 부호들의 집합이라고 할 수 있습니다. R에서 기본적인 문자표현은 UTF-8이라는 인코딩 방식을 사용합니다. 다음의 위키페이지를 읽어보시면 더욱 도움이 될 것입니다. (http://ko.wikipedia.org/wiki/Utf-8). 이에 대해서 조금 알아보겠습니다.

먼저 x에 단순히 영문 문자열 Hi를 입력한뒤 Encoding()함수를 이용하여 인코딩이 무엇인지 알아봅니다.

```
gnustats@CHL072:~$ R --quiet
> x <- "Hi"
> Encoding(x)
[1] "unknown"
>
```

앗... unknown (알수없음) 이라고 합니다. 그 이유는 아직 x 라는 문자열에 특정히 주어진 인코딩 방식이 없기 때문입니다. 따라서, R은 이런 문자열의 처리를 기본적인 인코딩 (즉, native.enc 또는 "")을 따르게 될 것입니다.

```
> options()$encoding
[1] "native.enc"
>
```

여기에서 native.enc 이라는 것은 위에서 l10n_info()라는 함수를 통해서 확인이 가능합니다.

```
> l10n_info()
$MBCS
[1] TRUE

$`UTF-8`
[1] TRUE

$`Latin-1`
[1] FALSE
```

기본 로케일 (즉, 운영체제가 사용되는 지역에 있는 사용자를 위한 설정)과 관계되어 있습니다. (본래 이 로케일은 또한 시간과 날짜를 표기하는 것과 밀접한 관련이 있으므로, 로케일에 대한 기본내용, 로케일이 정의된 위치, 캐릭터맵, 한국어 관련 로케일 추가방법, 그리고 한국어 출력 방법에 대해서는 "문자열과 날짜 및 시간"이라는 챕터에서 "날짜 및 시간"이라는 섹션에 기록해 두었었으므로, 이에 대해서 알고 싶으신 분들은 관련 섹션을 참고해주시길 바랍니다)

보통 해당 로케일에서 사용되는 인코딩 방식 native.enc이라고 하는 것은 영어권에서는 latin-1 또는 ISO-8559-1입니다. 그리고, 한국어, 중국어, 일본어와 같은 동아시아는 UTF-8 이라는 인코딩방식을 통하여 문자를 표기하게 됩니다. 그런데, 언어별로 알파벳은 latin-1, 한자권 언어는 UTF-8 이라고 하면 사용의 통일성에 문제가 생기게 됩니다. 따라서, R의 기본적인 언어처리는 l10n_info()의 함수에서 보여주는 것과 같이 UTF-8 로 통일하여 문자를 처리하게 하되, 일반적인 문자열 x 가 입력되었을때 이를 따로 지정하도록 할 수 있게 해두었습니다.

이를 확인하기 위해 아래처럼 입력해봅니다.

```
> x1 <- "난 한글이다!"
> Encoding(x1)
[1] "UTF-8"
>
```

만약, 한글이 latin1 이라는 인코딩이 사용하게 되면 어떻게 될까요? 아래처럼 해봅니다.

```
> x1
[1] "난 한글이다!"
> Encoding(x1) <- "latin1"
> x1
[1] "ë\u0082\u009c í\u0095\u009cê¸\u0080i\u009d´ë\u008b¤!"
>
```

알 수 없는 이상야릇한(?) 문자가 찍혀나옵니다. 그리고, 여러분은 금방 알아차릴 것입니다. 바로 저 문자들이 여러분이 데이터를 불러올 때 가장 많이 보게 되는 에러메시지라는 것을요.

저 문자들을 제대로 처리하기 위해서는 아래와 같이 합니다.

```
> x1
[1] "ë\u0082\u009c í\u0095\u009cê¸\u0080i\u009d´ë\u008b¤!"
> iconv(x1, to="UTF-8")
[1] "난 한글이다!"
>
```

여기에서 사용된 iconv()라는 함수는 이렇게 인코딩을 변경해주는 역할을 합니다.

그런데 일반적으로 입력되어지는 x라는 객체의 인코딩을 알수는 없습니다. 또한, R은 그러한 기능이 현재 (R-3.0.0 기준)는 존재하지 않습니다. 따라서, 위에서 사용한 iconv(x1, from="latin1", to="UTF-8")라고 하시면 에러를 경험하게 될 것입니다. 한글을 보기 위해서는 to 라는 인자만 이용하시면 됩니다.

그러나, 일반적으로 데이터를 불러올때는 이는 더 큰문제가 될 것입니다. 이런 경우에는 iconvlist()라는 함수를 이용하여 현재 R이 지원하는 모든 인코딩 종류와 매칭하여 한글이 잘 나오는가를 확인해 보는 길 밖에는 없습니다. 이에 대한 실례는 다음의 두 가지 메일링 답변을 통해서 확인해 보실 수 있습니다.

- read.table()에서 입력인코딩과 출력인코딩 옵션을 설정하여 알수없는 한글 인코딩으로 되어 있는 엑셀파일을 읽어오고자 할때 성공적으로 읽어오는 방법 [Ihelp-urquestion] Fwd: 페북에서 댓글 보고 맥에서 r사용시 한글 폰트 깨짐현상 질문드립니다.
- iconvlist()를 이용하여 알수없는 인코딩 파일 중에서 한글을 표현해줄 수 있는 인코딩을 찾는 일반적인 방법 [Ihelp-urquestion] [KRUG] Windows XP Home Edition , Windows 7 Home Premium, ...
- 윈도우즈에서 R을 한글을 사용하지 않고 영문으로만 사용하도록 한글번역 자체를 오프해버리는 방법 [Ihelp-urquestion] R-3.0.0 한글/영어 전환법

1.5 패키지 관리