Knitr 소개

전희원
http://freesearch.pe.kr

January 14, 2013

Contents

Abstract

이 문서는knitr(Xie, 2012)를 사용해 어떻게 아름다운 리서치 문서를 만들 수 있는지 설명하기 위한 문서이다. 대부분의 한글 윈도우 사용자를 고려해 한글 윈도우 기반에서 RStudio환경을 기준으로 설명을 하고자 한다.

1 Envirement Setting

일반적으로 knitr를 활용한 분석 레포팅 문서 작업에는 TEX이라는 환경이 필요하며, 문서 편집을 위해서는 윈도우용 RStduio[1]를 사용한다. RStudio설치 방법은 홈페이지에 잘 설명이 되어 있으니 따로 설명하지 않겠다.

TexLive 2012를 윈도우 환경에서 설치하는 방법은 http://faq.ktug.or.kr/faq/Windows%BF%A1%BC%ADTeXLive2012%BF%CDkoTeX%BC%B3%C4%A1%C7%CF%B1%E2를 참고한다.

위 두가지 설치를 완료했다면, 필요한 모든 프로그램은 구비가 된 것이다. 단 한가지 설정을 확인할 부분이 있는데, RStduio를 실행한뒤 Tools-Options-Sweave에서 *'Weave Rnw file using'* 을 'knitr'로 설정한 부분을 확인 한다.

[1] 윈도우용 RStudio는 http://www.rstudio.com/에서 다운받을 수 있는데, 서버용이 아닌 윈도우용으로 다운받도록 한다.

2 knitr

knitr는 R코드를 포함한 TEX포맷 형식의 Rnw파일을 컴파일해 `.tex`파일을 만드는 역할을 한다. 또한 HTML을 비롯해 마크다운(markdown)[2]등의 다양한 문서포맷으로의 변환을 지원하며 R코드의 실행 결과를 임베딩해 해당 포맷으로 리턴해 주는 역할을 수행한다.
사실 knitr는 Sweave를 보강한 패키지로서 Sweave의 많은 단점을 보강하고 새로운 기능을 추가한 패키지이다. Sweave와 함께 R을 기반으로 Literate Programming(문학적 프로그래밍)[3]을 가능하게 한다. 현재 이 패키지는 R에서 시각화 툴과 더불어 상당히 많은 각광을 받고 있는 패키지이며 많은 논문저자나 연구원들이 이 패키지를 사용해 연구문서작업을 수행하고 있다.
필자의 경우 이 패키지를 사용하여 마크다운기반의 분석 리포트를 공유해 왔으며 간단한 명령어 하나로 문서와 코드를 분리하여 필요시 코드만 사용할 수 있게끔 관리하곤 했다.
문학적 프로그래밍의 묘미는 데이터 분석시 타인을 설득하는 문서를 작성하기 위함와 더불어 본인 스스로 분석 논리를 정리해 나가는 과정을 수행하며 분석의 타당성을 스스로 입증해 내게 하는 힘이 있는 기법이다.

3 한글 윈도우 문제

한글 윈도우에서 R기반 업무를 하다보면 항상 문제가 생기는 부분이 있는데, 바로 한글 인코딩이 R의 인코딩 체계와 맞지 않아서 생기는 문제가 대부분이다. 이는 R이 태생적으로 유니코드 기반의 인코딩에서 최적으로 작동하게끔 설계된 이유도 있으며, 다른 한편으로는 한글 윈도우가 유니코드기반이 아니기 때문이기도 한다. 많은 분들이 알고 계시는 한글 윈도우의 한글 인코딩 기법은 'cp949' 혹은 'euc-kr'로 알려져 있다.
문제는 R에서 파일을 오픈할때 인코딩을 입력받게 되어 있는데, 기본 옵션으로 getOption('encoding')에서 출력되는 인코딩을 사용하는데, 이 함수의 리턴값은 한글 윈도우에서는 "na-

[2] 마크다운(markdown)은 간단한 마크업 언어로, 이메일 상에서 일반 텍스트로 문장 구조를 표기하던 관례를 규칙으로 만든 문법이다.

[3] 문학적 프로그래밍(literate programming)은 프로그래밍 방법론의 한 가지로, 프로그래밍을 할 때 컴퓨터로 컴파일 가능한 코드를 만드는 것보다 사람이 이해하기 쉬운 코드를 만드는 것에 중점을 두는 방법이다. 다시 말해, 사람이 보고 이해할 수 있도록 문서를 만들듯이 프로그래밍을 하는 것이 목적이다. '마치 문학작품을 읽는 것처럼 프로그래밍을 읽을 수 있도록 만드는 것'이 목표이므로 '문학적 프로그래밍'이라는 이름을 지었다.
문학적 프로그래밍에서는 프로그램에 대한 문서와 소스 코드가 하나의 소스 파일에 들어 있으며, 특별한 프로그램을 사용하여 문서와 소스 코드를 각각 따로 뽑아낸다. 소스 코드는 흔히 '덩어리'라고 부르는 단위로 구분하고, 각각의 덩어리가 어떤 원리로 동작하는지 등을 소스 파일에서 자세히 설명한다. 여기서 덩어리는 실제 소스 코드의 순서와 다르게 배열되어도 상관 없으며, 문서로 설명하고자 하는 순서대로 배열한다. 컴파일러나 인터프리터가 처리할 수 있는 형태로 만드는 것은 별도의 프로그램을 이용한다.(출처: 위키피디아)

tive.enc" 이다. 이 때문에 대부분 UTF-8 인코딩으로 작성하는 knitr 문서의 경우 잘못한 인코딩을 기반으로 문서를 컴파일하게 되며 에러를 내고 만다.
따라서 한글 문서 작성을 위해 지켜야 될 아래와 같은 룰이 있다.

- knitr 문서는 반드시 UTF-8로 작성한다.
  - 이는 RStudio에서 Tools-Options-General에서의 제일 하단에서 설정할 수 있다.
- `.Rnw`파일 편집화면에서 *Compile PDF* 명령어를 사용하지 않고, 아래와 같은 명시적인 코드로 실행한다.

```
# 만일 'foo.Rnw' 파일을 편집하고 있다고 가정한다면.
options(encoding = "UTF-8")
library(knitr)
knit("foo.Rnw")
system("pdflatex foo.tex")
```

필자의 경우 위 코드와 같은 방법으로 매번 실행을 하다가 아래와 같은 함수를 만들어 함수를 호출하는 방법으로 pdf문서로 컴파일 하였다. 아래 방법의 경우 encoding을 다시 원상태로 복귀를 시켜주는 장점이 있어서 약간 편리하다.

```
compile_pdf <- function(filename, typeset = "pdflatex") {
    require(knitr)
    prev_enc <- getOption("encoding")
    options(encoding = "UTF-8")
    rnw <- knit(filename, envir = new.env())
    system(paste(typeset, rnw))
    options(encoding = prev_enc)
}

compile_pdf("foo.Rnw")
```

이런 함수도 귀찮다 싶은 독자분은 아래와 같은 명령어만 저장해둬서 사용하는 것도 좋다.

```
require(devtools)
source_gist("https://gist.github.com/4485043")
compile_pdf("foo.Rnw")
```

물론 위 함수가 동작하기 위해서는 1 절에서의 모든 설정이 정확하게 적용되어야 된다. 참고문헌을 만들기 위해서는 여러번 latex 명령어를 수행해야 되는데, 이 때에는 위 함수가 적합하지 않으므로 위 gist 에서 제공하는 `knit_ko_win()` 이라는 함수를 사용하는 것도 좋다. 아마도 R 인코딩 체계에 대해서 이해를 하고 있다면 위와 같은 코드를 사용하지 않고 여러분 나름대로의 방법으로 실행하고 컴파일 하면 될 것이다. 만일 R 인코딩 체계에 대해서 알고 싶은 독자분이 계시다면 필자가 공개해둔 'R에서의 한글표현 및 인코딩'(`http://www.youtube.com/watch?v=araCWRa5Nxg`) 이라는 동영상을 봐두는 것도 도움이 될 것이다.

많은 R사용자들은 R을 좀더 쾌적하게 사용하기 위해서 리눅스를 사용하라고 이야기 한다. 이는 바로 이런 인코딩문제와 더불어 R이 리눅스를 기반으로 개발되고 있기 때문이다. 하지만 이 문서가 왜 윈도우를 기반으로 하고 있느냐고 묻는다면 국내 R 사용자 대부분이 윈도우를 사용하고 있기 때문이라고 말씀드리고 싶다.

References

Xie Y (2012). *knitr: A general-purpose package for dynamic report generation in R*. R package version 0.9, URL `http://CRAN.R-project.org/package=knitr`.