R을 활용한 보고서 작성 및 자동화

https://mrchypark.github.io/dabrp_classnote2/class7

박찬엽

2017년 8월 2일

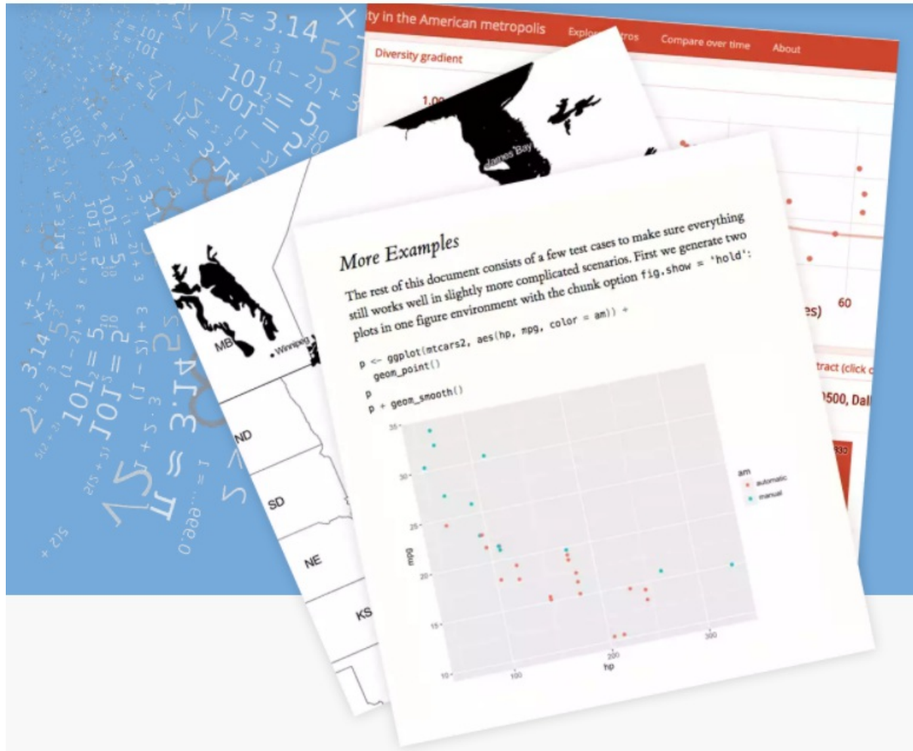
목차

과제 확인

과제 확인

R Markdown from RStudio

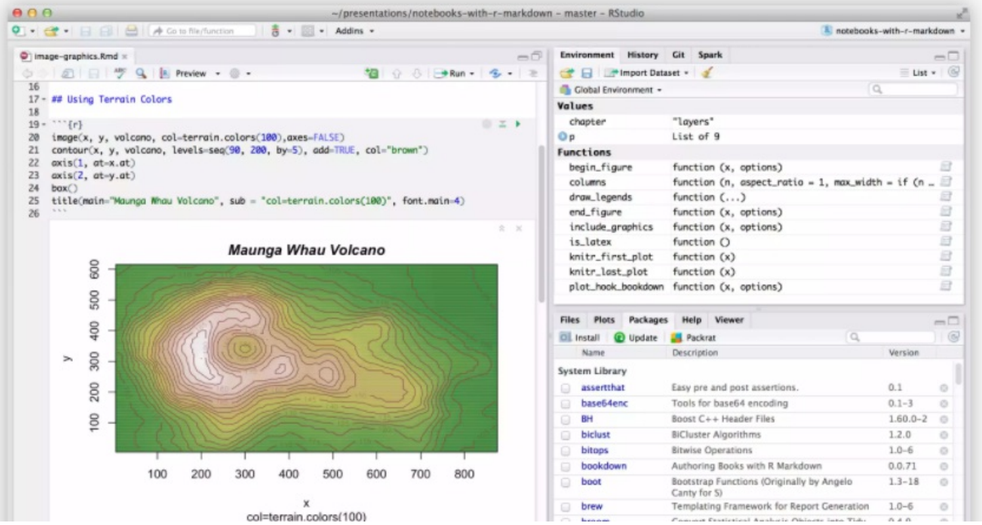
Get Started Gallery Formats Articles



Analyze. Share. Reproduce.

Your data tells a story. Tell it with R Markdown. Turn your analyses into high quality documents, reports, presentations and dashboards.

R Markdown documents are fully reproducible. Use a productive notebook interface to weave together narrative text and code to produce elegantly formatted output. Use multiple languages including R, Python, and SQL.



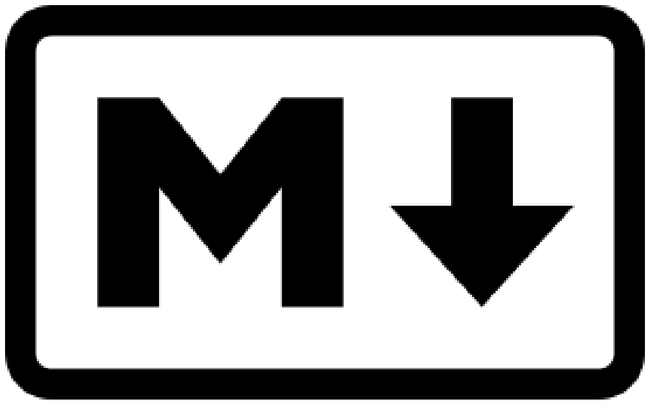
rmarkdown 이란

rmarkdown 패키지는 Rmd 양식의 문서를 knitr과 pandoc으로 다양한 문서양식으로 생성해주는 패키지입니다. knitr 역시 Rmd양식의 문서를 md양식의 문서로 바꿔주는 패키지이며, pandoc 생태계에 의지해 md양식을 다양한 양식 문서로 만들 수 있게 되었습니다.

knitr은 rmd양식의 문서 중 R엔진의 처리가 필요한 계산과 이미지 생성등의 작업을 자동화하여 md의 문법과 위치에 맞게 md문서를 생성해 줍니다. 그리고 pandoc이 YAML양식의 해더 정보를 바탕으로 다른 포멧의 저작물로 변환해주는 것입니다. 변환 가능한 저작물로는 notebook, html, pdf, word, odt등의 문서와 ioslides, reveal.js, Slidy, Beamer의 슬라이드, 대쉬보드, 웹페이지, 책 등 입니다. 지금 자료는 xaringan(remark.js 기반 html 프레젠테이션)으로 만들었습니다.

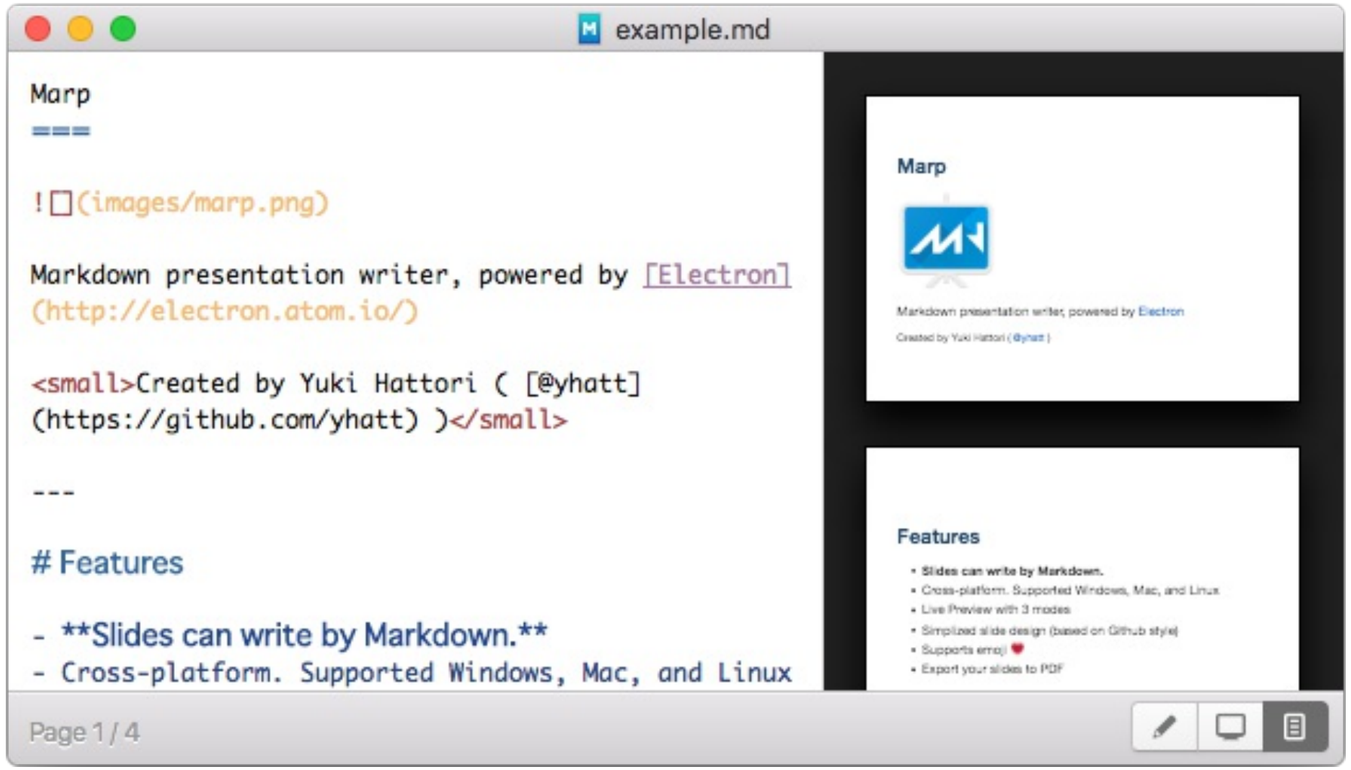
md와 rmd

markdown은 마크다운이란 plain text를 쉽게 일정 양식의 문서로 바꿀 수 있게 간단한 규칙을 정해 놓은 것을 말합니다. 그 중 문서에 필요한 최소한의 markdown 규칙을 commonmark에서 지정을 했구요. 너무 단순한 규칙 때문에 매우 많은 확장 문법들이 각자의 방식으로 발전했습니다. 특히 오픈소스의 발달로 소스에 대한 설명 등이 plain text 양식으로, README 라는 이름의 파일로 남기는 것이 관례가 되었습니다. 소스 관리 저장소 서비스를 운영하는 github 같은 곳에서는 웹으로 서비스를 제공하고 있고, 이 plain text를 예쁘게 보여주기 위해서 markdown문법을 채택했으며 각 서비스 들이 각자 추가적인 문법을 제공하고 있는 중입니다. 현재 0.27버전이 정리가 되었고 이곳에서 확인하실 수 있습니다.



marp

마크다운을 실시간으로 반영해서 보여주는 에디터입니다. 함께 사용하면서 실습해보겠습니다.



문법 소개

markdown이 down인 만큼 html에서 구조화를 위해 사용하는 요소들을 사용할 수 있습니다.

문법 소개

markdown이 down인 만큼 html에서 구조화를 위해 사용하는 요소들을 사용할 수 있습니다.

```
제목 : # , =====
인용 : >
강조 : * , _
링크 : [텍스트](주소 "설명 생략가능")
이미지 : ![텍스트](이미지주소 "설명 생략가능")
리스트 : 1 , * , - , +
코드표시 : <code>코드</code> , 한줄 띄우고 스페이스 4칸 , ``코드`` (` 가 3개여야 함)
줄바꿈 : 엔터 2번 , 강제 줄바꿈은 문장끝에 스페이스바 2칸
가로선 : ----- , ***** , +++++
```

제목

#이 표시가 html에서는 <h1> 태그와 같습니다. <h>는 총 6까지 있으며 #표시는 숫자를 쓰는게 아니라 # 갯수로 수를 표현합니다.

```
# 제목1 : HTML의 <h1> 태그
## 제목2 : HTML의 <h2> 태그
### 제목3 : HTML의 <h3> 태그
#### 제목4 : HTML의 <h4> 태그
##### 제목5 : HTML의 <h5> 태그
###### 제목6 : HTML의 <h6> 태그
```

인용

인용은 인용 속에 인용, 인용 속에 인용이 가능한 구조입니다. 보통 왼쪽에 세로줄이 있는 형태로 외부에 보여줍니다.

```
> ## 제일 바깥의 인용
>> ** 인용의 인용 **
```

> ## 제일 바깥의 인용
>
> > **인용의 인용**

blod & italic

강조하는 방식으로는 * 이나 _의 갯수로 결정납니다.

```
이것은 **굵은 글씨** 입니다.
이것은 __굵은 글씨__ 입니다.

이것은 *기울여 쓰기* 입니다.
이것은 _기울여 쓰기_ 입니다.

이것은 ***강조하며 기울여 쓰기*** 입니다.
이것은 ___강조하며 기울여 쓰기___ 입니다.
```

이것은 **굵은 글씨** 입니다.
이것은 **굵은 글씨** 입니다.

이것은 *기울여 쓰기* 입니다.
이것은 *기울여 쓰기* 입니다.

이것은 ***강조하며 기울여 쓰기*** 입니다.
이것은 ***강조하며 기울여 쓰기*** 입니다.

links

바로가기 링크를 페이지상에 바로 구현할 수 있습니다. 링크를 바로 옆에 작성하는 방식과 주석처럼 다는 2가지 방식이 있습니다. 앞에서 사용한 방식을 인라인 방식이라고 하고, 뒤에 방식을 레퍼런스 방식이라고 합니다.

```
이것은 [링크](http://www.example.com) 입니다.
이것은 [링크](http://www.example.com "설명문구 생략가능") 입니다.
이것은 [구글 링크][1] 입니다.
이것은 [마이크로소프트 링크][b] 입니다.
이것은 [애플 링크][three] 입니다.

[1]: https://www.google.co.kr "구글"
[b]: https://www.microsoft.com/ko-kr/ "마이크로소프트"
[three]: http://www.apple.com/kr/ "애플"
```

이것은 링크 입니다.

이것은 링크 입니다.

이것은 구글 링크 입니다.

이것은 마이크로소프트 링크 입니다.

이것은 애플 링크 입니다.

image

이미지를 첨부하는 방법은 마우스를 사용하는 방식이 아니라서 좀 불편할 수 있습니다. 앞서 바로가기 링크를 만드는 것과 같은 방식인데 앞에 ! 를 추가해서 작성합니다. 이미지는 실제로 문서에서 해당 링크의 이미지를 가져와서 문서에서 보여주는 방식입니다. 이미지의 위치나 크기 같은 것을 조정하기 위해서는 기본 markdown 문법에서는 제공하지 않습니다. 그래서 관련 문법을 제공하는 서비스를 사용하거나 `html` 문법이나 `css` 를 사용해야 합니다.

```
1. ![깃헙](https://upload.wikimedia.org/wikip
2. ![깃허브][imgGithub]

[imgGithub]: https://upload.wikimedia.org/wi

3. <img src="https://upload.wikimedia.org/wi
```

1. 

2. 

3. 

number

앞에 숫자를 붙여서 작업하고 싶을 때가 있습니다. `1.` 를 앞에 입력하시면 들여쓰기가 같은 줄에 따라 숫자를 붙여줍니다. 숫자를 자유롭게 사용하셔도 순서대로 작업해주니 보통 1로만 작성하는 것 같습니다. 들여쓰기는 4칸 띄어쓰기를 의미합니다.

```
1. 번호1
1. 번호2
    1. 번호1
    1. 번호2
    1. 번호3
1. 번호3
  1. 번호3
9. 번호9
```

1. 번호1
2. 번호2
    1. 번호1
    2. 번호2
    3. 번호3
3. 번호3
    1. 번호3
4. 번호 9

list

숫자 붙이는 것과 같이 앞에 동그라미 표시 등을 통해서 간결한 문체로 작성할 때가 있습니다. 들여쓰기로 3단까지 다른 모양을 자동으로 보여줍니다.

```
  리스트 *
    * 리스트 *
    - 리스트 -
  리스트 *
  - 리스트 -

  리스트 *
    - 리스트 -
        - 리스트 -
            - 리스트 -
```

- 리스트 *
  - 리스트 *
  - 리스트 -
- 리스트 *
  - 리스트 -
- 리스트 *
  - 리스트 -
    - 리스트 -
      - 리스트 -

코드

코드는 복사하기 좋고, 텍스트 그대로를 문서에 남길때 사용합니다. 위에 복사를 위한 글자들 모두 코드를 작성하는 양식으로 작성하였습니다. 아래에는 코드 안에 내용을 보이게 하기위해서 앞에 `를 2개만 작성했습니다. souce를 직접 확인해 보시면 좋습니다.

``(`가 3개 여야 함)
코드
``(`가 3개 여야 함)

```
코드
```

수식

LaTex는 수식을 입력하는 규칙이 규정되어 있습니다. 이를 활용해서 $$ 수식문법 $$ 의 규칙으로 복잡한 수식을 작성할 수 있습니다.
수식문법에 대해서는 좋은 포스트를 공유하니 추가적으로 필요하신 분들은 공부해보시면 좋을 것 같습니다.

```
$$
\left|\sum_{i=1}^n a_ib_i\right|
\le
\left(\sum_{i=1}^n a_i^2\right)^{1/2}
\left(\sum_{i=1}^n b_i^2\right)^{1/2}
$$
```

$$\left|\sum_{i=1}^n a_ib_i\right| \le \left(\sum_{i=1}^n a_i^2\right)^{1/2} \left(\sum_{i=1}^n b_i^2\right)^{1/2}$$

Rmd 양식으로 문서 작성

rmarkdown는 위에서 설명한 markdown문법을 활용하여 R의 명령문과 조합해서 문서를 작성하기 위한 하나의 양식입니다. 공식 홈페이지는 이곳이고 소스는 이 곳을 확인하시면 됩니다. 한글 cheatsheet도 있고, rstudio > help > cheatsheet 에도 cheatsheet과 reference가 있습니다.

R code 실행 결과

Rmd의 가장 강력한 점은 코드의 결과를 옮겨적거나 따로 저장해서 고쳐내지 않고, 최종 결과물을 코드에서 바로 작성할 수 있다는 점입니다. Rmd의 R 코드 실행시 변환해주는 내용은 아래 5가지 입니다.

```
{R, results="asis",echo=F}
if(!require(knitr)){
  install.packages("knitr")}
library(knitr)

obj<-c("소스코드","텍스트",
"플롯","메세지","경고")
role<-c("코드청크에 들어있는 R 코드",
    "summary(iris)와 같은 글자 출력 결과물",
    "plot(iris)와 같은 그림 출력 결과물",
    "메세지","경고")

dat<-data.frame("객체"=obj,"목적"=role)

kable(dat,align="cl", "html")
```

객체	목적
소스코드	코드청크에 들어있는 R 코드
텍스트	summary(iris)와 같은 글자 출력 결과물
플롯	plot(iris)와 같은 그림 출력 결과물
메세지	메세지
경고	경고

line R code

글자들 사이에 만약 항상 문서를 생성하는 시점의 날짜로 지정하고 싶다면 아래와 작성하면 됩니다.
본 문서는 2017-08-02 에 작성되었습니다.
소스를 확인해서 어떻게 작성했는지 확인해보세요.

code chunk

chunk는 덩어리라는 뜻 답게 코드가 실행되는 하나의 단위입니다. Rmd은 컴퓨터에 설치만 되어 있다면 다른 언어들도 chunk내의 실행 결과를 변환해서 md파일로 바꿔줍니다. R과 다른 점은 각 chunk별로 독립적인 프로세스를 실행시키기 때문에 변수들이 chunk 사이에서 함께 사용할 수 없습니다. 원래 코드를 작성하는 방식에 code를 작성하면 되는데 윗줄의 표시 뒤에 중괄호 R을 작성하면 R 엔진으로 실행시킬 코드라는 사실을 knitr의 렌더링 함수가 이해하게 됩니다.

출력 옵션들

위에 5개 내용에 대해서 fig.cap을 빼고 모두 TRUE/FALSE로 출력할지 말지 조정할 수 있습니다.

```
echo    : 코드
include : 코드와 글자 출력물
fig.cap : 그림 출력물
message : 메세지
warning : 경고
```

각 옵션은 chuck에게 R로 처리함을 알려주는 {R} 안에 작성합니다. 앞에 "는 제거해야 합니다.

```
{r echo=T}
{r echo=T, message=F}
{r echo=T, message=F,warning=F}
```

fig.cap은 그림 출력물의 가로 세로 길이나 위치 등의 옵션을 조정할 수 있습니다.

실습 1

1. rstudio > File > New File > R MarkDown... 으로 rmd 파일을 만드세요.
2. code chunk를 만들고 수업 6에 과제 1.3의 코드를 넣어주세요.
3. 결과를 확인하고, 그림만 출력, 코드만 출력을 진행해주세요.

YAML 해더

Rmd 초기 설명에 잠깐 언급한 YAML 해더에 대해서 설명하겠습니다. 이것에 따라서 html, word 등 출력 결과물을 조절할 수 있습니다.

```
---
title: 테스트 해더
author: 박찬엽
date: 2017년 5월
output: html_document
---
```

output 옵션

첫 양식인 html을 하기 위해서는 위와 같이 output 옵션을 html_document로 지정하면 됩니다. 그러면 해더 정보를 바탕으로 html 문서로 변환을 해줍니다. html 문서로 변환하는 것의 가장 큰 장점은 역시 동적 문서 형식을 유지할 수 있다는 점입니다 PDF나 word는 정적 문서이기 때문에 제한 된 기능으로만 출력물을 저장할 수 있습니다. Rmd에 동적 출력 결과물이 code상에 실행하도록 되어 있다면 PDF나 word로 변환시 에러를 일으킵니다. webshot은 본래 웹페이지를 기계적으로 스크린샷하기 위해 작성된 패키지인데, 문서중 동적인 요소들을 스크린샷해서 반영해주는 동작을 해줍니다.

테마 설정

특별히 해더에 정보를 추가해서 디자인이나 테마를 변경할 수 있습니다. html은 이미 많은 style을 미리 지정해두고 이름만으로 사용할 수 있게 되어있습니다. style에서 지정해볼 수 있는 다른 테마들을 찾아보세요.

```
---
title: 테스트 해더
author: 박찬엽
date: 2017년 5월
output:
  html_document:
    theme: flatly
---
```

실습 2

1. 실습 1에서 만든 파일을 불러오세요.
    - 테마를 `Yeti`로 설정해주세요.
    - 결과를 확인하고 다른 테마도 시도해주세요.

특별한 디자인 변경

디자인은 내부적으로 css 파일에 작성된 내용으로 결정됩니다. 세부적인 변경을 작성한 css를 추가하려면 아래와 같이 작성합니다.

```
---
title: 테스트 해더
author: 박찬엽
date: 2017년 5월
output:
  html_document:
    css: css/customStyle.css
---
```

위와 같이 작성하고 css 오른쪽에 디자인에 해당하는 css 파일의 경로를 지정해주면 됩니다.

YAML params

YAML이 데이터 양식이라고 설명을 드렸습니다. 그래서 Rmd내부에서 접근할 수 있는 데이터를 params라는 이름으로 미리 작성해 둘 수 있습니다. 지금은 inline 코드를 양쪽에 -로 처리 했습니다.

```
---
title: -r params$chapter- 테스트 해더
params:
  chapter: 5
author: 박찬엽
date: 2017년 5월
output:
  html_document:
    css: css/customStyle.css
---
```

해더 내 양식으로 보여드리려고 위와 같이 작성했지만 Rmd 문서 모든 code 작성 공간에서 같은 양식으로 값에 접근할 수 있습니다.

word 문서로 만들기

Rmd는 word 문서로도 변환할 수 있습니다. 위에서 설명한 html_document 위치에 word_document 라고 입력하면 됩니다.
word에서는 스타일을 word문서에서 참조하는 방법이 있어 소개하려고 합니다.

reference docx 사용하기

1. 우선 `Rmd`를 `word` 출력물로 설정해서 새로 만듭니다. 그리고 knit 버튼을 이용해 우선 `docx` 확장자의 `word` 문서를 만듭니다. 그냥 새 파일을 만들수도 있지만, 호환이 잘 안되는 경우도 있어서 안전한 방법을 설명했습니다.

reference docx 사용하기

1. 우선 `Rmd`를 `word` 출력물로 설정해서 새로 만듭니다. 그리고 knit 버튼을 이용해 우선 `docx` 확장자의 `word` 문서를 만듭니다. 그냥 새 파일을 만들수도 있지만, 호환이 잘 안되는 경우도 있어서 안전한 방법을 설명했습니다.

2. 만들어진 워드 파일을 새로운 이름으로 바꿔 저장합니다. 이번 예시에서는 `style.docx`로 저장했습니다.

reference docx 사용하기

1. 우선 `Rmd`를 `word` 출력물로 설정해서 새로 만듭니다. 그리고 knit 버튼을 이용해 우선 `docx` 확장자의 `word` 문서를 만듭니다. 그냥 새 파일을 만들수도 있지만, 호환이 잘 안되는 경우도 있어서 안전한 방법을 설명했습니다.

2. 만들어진 워드 파일을 새로운 이름으로 바꿔 저장합니다. 이번 예시에서는 `style.docx`로 저장했습니다.

3. 저장된 `style.docx` 파일을 열어서 스타일 정보를 수정합니다. 확인하셔야 할 부분은 수정해야될 정보가 스타일이지, 각 텍스트 들의 디자인이 아니라는 점입니다.

reference docx 사용하기

1. 우선 Rmd를 word 출력물로 설정해서 새로 만듭니다. 그리고 knit 버튼을 이용해 우선 docx 확장자의 word 문서를 만듭니다. 그냥 새 파일을 만들수도 있지만, 호환이 잘 안되는 경우도 있어서 안전한 방법을 설명했습니다.

2. 만들어진 워드 파일을 새로운 이름으로 바꿔 저장합니다. 이번 예시에서는 style.docx로 저장했습니다.

3. 저장된 style.docx 파일을 열어서 스타일 정보를 수정합니다. 확인하셔야 할 부분은 수정해야될 정보가 스타일이지, 각 텍스트 들의 디자인이 아니라는 점입니다.

4. 스타일 정보를 YAML 해더에 아래와 같이 입력해 줍니다.

```
---
title: 워드 스타일 테스트 해더
author: 박찬엽
date: 2017년 5월
output:
  word_document:
    reference_docx: style.docx
---
```

파일의 경로는 Rmd 파일이 있는 폴더의 위치가 working diractory라고 인식하므로 같은 곳에 있으면 추가적인 경로 작성 없이 위와 같이 파일 이름만 작성해 주면 됩니다. word_style.Rmd에 가능한 한 보이는 양식들을 넣은 기본 문서로 작성했으니 이 것으로 word문서를 생성후 스타일 파일로 사용하시면 좋을 것 같습니다.

PDF 문서로 만들기

PDF 문서로 만들 때는 위에 잠시 나왔던 LaTex이 설치되어 있어야 합니다. 이곳에서 각자의 운영체제에 맞게 다운 받아 설치하시면 됩니다. `linux`는 기본 설치 되어 있을 텐데 콘솔에서 `latex`을 입력해보시면 됩니다. 없으면 debian 계열 `sudo apt-get install texlive`, Centos 계열 `sudo yum install tetex`로 설치할 수 있습니다.

PDF는 `word_document` 부분을 `pdf_document`라고 하면 됩니다. `?pdf_document`를 입력해서 설정할 수 있는 것이 무엇이 있는지 확인해 보세요.

한글 때문에 폰트 및 추가로 설정해야 하는 부분이 있습니다. 여기와 여기를 참고하세요.

스케줄러

운영체제에 포함되어 있는 스케줄러란 일정한 시간이나 조건에 반복적으로 작업을 수행하게 해주는 프로그램입니다. 어떤 언어에서든 계속적인 작업을 위해서 스케줄러를 사용하는데, 아래는 대표적으로 사용하는 스케줄러를 소개하고 있습니다.

특히 환경변수라고 하는 부분에 대해서 `rstudio`의 작업환경과 조금 다르고, 분리해서 생각해야 할 부분이 있어서 조금 골치 아프기도 합니다. 운영체제에 특화되서 사용하기도 합니다. 윈도우는 내장되어 있는 작업 스케줄러, 리눅스 계열과 mac은 전통적으로 `cron`을 사용하고 있습니다. 서버에 일을 시키기 위해서 `cron`을 사용하는 방법을 익히는 것은 매우 중요한 것 같습니다.

windows 작업 스케줄러

윈도우는 실행 파일을 정해진 시간에 자동으로 실행시켜주는 `task scheduler`가 있습니다. 이것을 컨트롤하는 패키지인 taskscheduleR가 있어서 함께 사용해 보겠습니다.

linux 작업 스케줄러

cron 또한 위와 같은 cronR이 있습니다. 사용하는 모양이 같으니 한번 사용해 보겠습니다. 두 가지 모두 rstudio를 관리자 권한으로 실행시켜줘야 합니다.

과제 1

1. 각 github에 `class7` 이라는 저장소를 만들어주세요.
    - class7 저장소로 프로젝트를 만들어주세요.
    - 수업 6의 과제를 모두 index.rmd 파일로 작성해 주세요.
    - 취향 것 예쁘게 만들어서 html 파일로 변환해주세요.
    - 프로젝트내에 docs라는 폴더를 만들어주세요.
    - index.rmd로 변환된 index.html 파일을 docs 폴더에 넣어주세요.
    - github setting에서 github pages에 폴더를 docs로 설정해주세요.
    - `https://`각자계정`.github.io/class7` 위치를 확인해주세요.