Literate Programming과 출판: LaTeX의 관점

서울 R 미트업

2023년 8월 10일

김강수

Korean TeX Users Group

O, what a tangled web we weave
When first we practice to deceive

— Sir Walter Scott

D. Knuth, 1984

Literate Programming

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on *explaining to human beings* what we want a computer to do.

The WEB System

because I think that a complex piece of software is, indeed, best regarded as a **web** that has been delicately pieced together from simple materials. We understand a complicated system by understanding its simple parts, and by understanding the simple relations between those parts and their immediate neighbors.

DUAL Usage concept: Weaving and Tangling

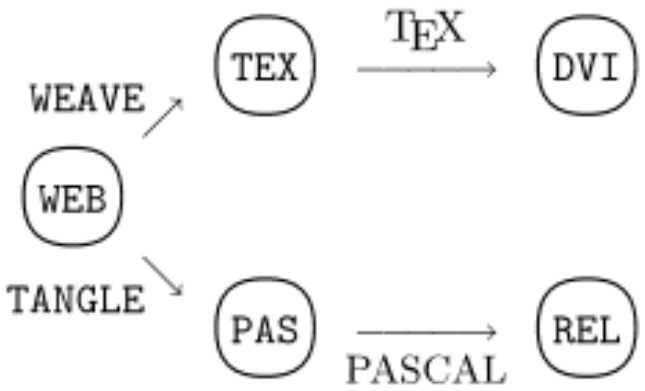


Figure 1. Dual usage of a WEB file.

TeX: The Program

216. When TeX's work on one level is interrupted, the state is saved by calling *push_nest*. This routine changes *head* and *tail* so that a new (empty) list is begun; it does not change *mode* or *aux*.

```
procedure push_nest;  { enter a new semantic level, save the old }
  begin if nest_ptr > max_nest_stack then
    begin max_nest_stack ← nest_ptr;
    if nest_ptr = nest_size then overflow("semantic␣nest␣size", nest_size);
    end;
  nest[nest_ptr] ← cur_list;  { stack the record }
  incr(nest_ptr); head ← get_avail; tail ← head; prev_graf ← 0; mode_line ← line;
  end;
```

217. Conversely, when TeX is finished on the current level, the former state is restored by calling *pop_nest*. This routine will never be called at the lowest semantic level, nor will it be called unless *head* is a node that should be returned to free memory.

```
procedure pop_nest;  { leave a semantic level, re-enter the old }
  begin free_avail(head); decr(nest_ptr); cur_list ← nest[nest_ptr];
  end;
```

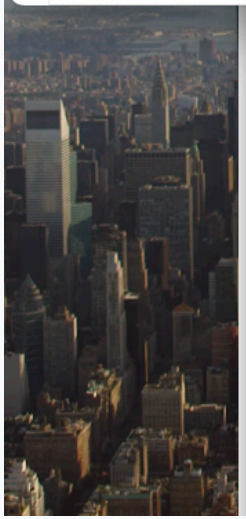
텍스트-코드 혼합 코딩: LaTeX

srcandtext.sty

a.tex

조판 clxelatex Macros 태그 Label 템플릿

```
%% 주의할 것은 \stm{myfn_undersix:NN}의 첫 인자를 ``남아 있는 clist''로,
%% 두번째 인자를 $2$를 기준으로 남아 있는 clist에 대하여 올림나눗셈한 결과로
%% 넘겨주어야 하는 것이다.
%% \numpar 다음 코드는 이것을 구현하였다.
%<Code>
\cs_new:Npn \myfn_oversix:NN #1 #2
{
%</Code>
%% clist의 현재 상태를 임시 clist로 복사하고 두 번째 인자를 \stm{l_catch_int}로 복사한다.
%% 6이하일 때와 동일.
%<Code>
    \clist_set_eq:NN \l_tmpa_clist #1
    \int_set_eq:NN \l_catch_int #2
%</Code>
%% 주어진 clist에서 \stm{l_catch_int}개까지 솎아내고 각각 임시 clist에 put
%<Code>
```



{1,2,3,4,5,6,7}

이 주어진 경우라면, 일단 ceil(7/3) = 3을 기준으로 최초 3개를 추려낸다. 그러면 a={1}, b={2}, c={3}이 되고 4, 5, 6, 7이 남는데, 이 4개에 대해서는 앞서 정의한 \myfn_undersix:NN을 적용하면 (단 a, b, c의 내용의 현재 상태를 바꾸지 않고) 4와 6이 a에, 5와 7이 b에 추가될 것이다. c에는 더이상 추가될 것이 없다. 주의할 것은 \myfn_undersix:NN의 첫 인자를 "남아 있는 clist"로, 두번째 인자를 2를 기준으로 남아 있는 clist에 대하여 올림나눗셈한 결과로 넘겨주어야 하는 것이다.

2. 다음 코드는 이것을 구현하였다.

```
\cs_new:Npn \myfn_oversix:NN #1 #2
{
```

clist의 현재 상태를 임시 clist로 복사하고 두 번째 인자를 \l_catch_int로 복사한다. 6이하일 때와 동일.

```
    \clist_set_eq:NN \l_tmpa_clist #1
    \int_set_eq:NN \l_catch_int #2
```

주어진 clist에서 \l_catch_int개까지 솎아내고 각각 임시 clist에 put

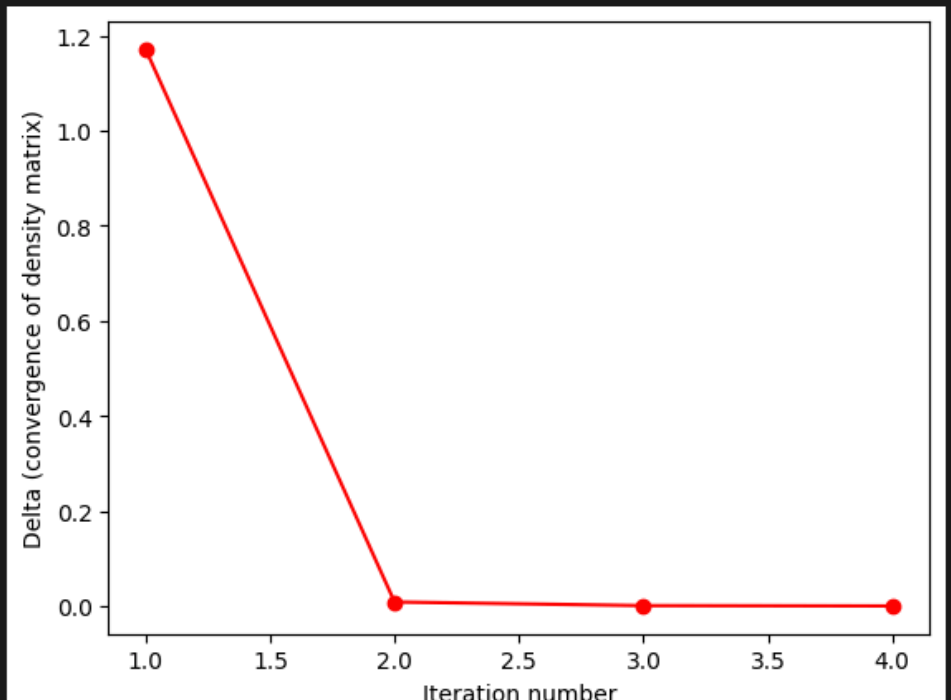
텍스트-코드 혼합 코딩: JupyterLab

Plotting delta

The variable delta holds the change in the magnitude of the density matrix between iterations. As this is the quantity we are using to check for convergence, let's go ahead and plot that too.

```
[47]
fig, axes = plt.subplots()
axes.plot(range(1,curriter+1),deltas, 'ro-')
axes.set_xlabel("Iteration number")
axes.set_ylabel("Delta (convergence of density matrix)")
plt.show()
```

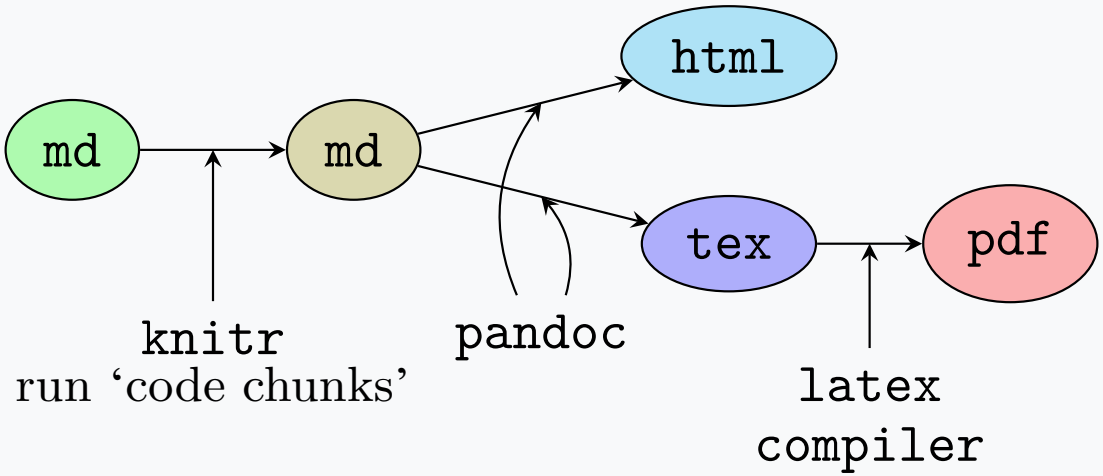
Python



Sweave Document: knitr, Rnw

```
206 % ==making bodies============================
207 \begin{document}
208 \SweaveOpts{concordance=TRUE}
209 \title{중심극한정리(Central Limit Theorem)의 예시}
210 \maketitle
211
212 <<random, cache=TRUE, echo=FALSE>>=
213 a<-1:999
214 n<-30
215 i<-50
216 var_a<-var(a)*((n-1)/n)
217 sd_a<-sqrt(var_a)
218 @
219 상자 안에 \Sexpr{min(a)}에서 \Sexpr{max(a)}까지 숫자가 표시된 \Sexpr{length(a
    )}개의 상태가 균질한 공($X$)을 넣고 이를 특정 모집단(population)이라고 가정하자
    . 이 모집단의 평균 $\mu(X)$은 \Sexpr{mean(a)}이다. 모집단의 분산은
```

code chunk를 이용하는 Markdown 어프로치



Comments

- Literate Programming은 결국, “실행되는 코드”와 그 코드와 관련된 “문서”를 동시에 생성하는 일이다. (즉, tangling과 weaving이라는 두 가지가 갖추어지면 Literate Programming이라고 할 수 있다.)

Comments

- Literate Programming은 결국, “실행되는 코드”와 그 코드와 관련된 “문서”를 동시에 생성하는 일이다. (즉, tangling과 weaving이라는 두 가지가 갖추어지면 Literate Programming이라고 할 수 있다.)
- Literate Programming은 프로그래밍(또는 코딩)의 관점을 코드 그 자체로부터 코드에 대한 ‘문해’로 옮길 것을 요구한다.

Comments

- Literate Programming은 결국, "실행되는 코드"와 그 코드와 관련된 "문서"를 동시에 생성하는 일이다. (즉, tangling과 weaving이라는 두 가지가 갖추어지면 Literate Programming이라고 할 수 있다.)
- Literate Programming은 프로그래밍(또는 코딩)의 관점을 코드 그 자체로부터 코드에 대한 '문해'로 옮길 것을 요구한다.
- Pascal(`WEB`), C(`CWEB`), Fortran(`FWEB`) 등 각 언어별로 문학적 프로그래밍 도구가 발달하였으며, 현재는 markdown에 기반하여 python을 활용하는 jupyter notebook, R을 활용하는 rmd, 이로부터 발전한 Quarto 등이 있으나, 핵심은 rendering (=weaving)과 exporting (=tangling)이라는 두 가지 기능이다.

LaTeX: The Backend Printing Engine

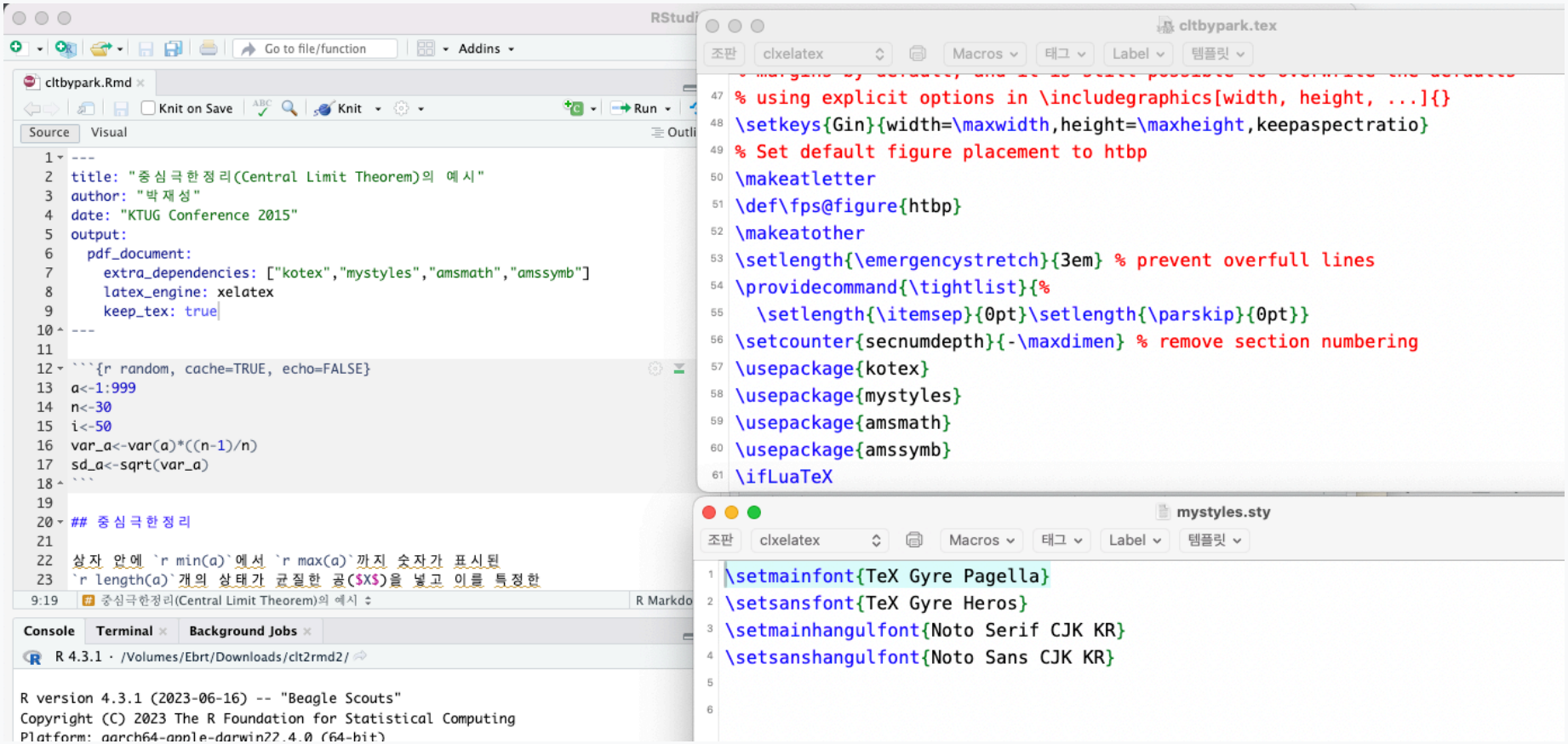
Markdown to LaTeX

- 출판을 문제삼는다면 최종 출력 포맷은 pdf이다.
- `pandoc`이라는 강력한 변환 툴이 이 역할을 하고 있다. `pandoc`은 다양한 문서 포맷 간의 상호 변환 도구이다.
- `multimarkdown`이라는 마크다운 유틸리티도 비슷한 일을 한다.
- 어떤 방법을 사용하든지, **컴파일 가능한 tex 파일**을 생성하였다면 pdf를 얻는 것은 LaTeX의 역할이 된다.

한국어 문서에서 만나는 문제

- LaTeX에서 다국어 문제는 대부분 babel 패키지에 의한다.
- 그러나 현재 한국어를 babel은 완전히 지원하지 않고 있어서(LuaLaTeX에서 부분적으로 지원한다) 다른 유럽어처럼 자연스럽게 LANGUAGE SPECIFIER 지정으로 언어 설정을 할 수 없다. (이 때문에 HTML을 생성할 때와 PDF를 생성할 때 언어 설정에 혼선이 생길 가능성이 없지 않다.)
- XeTeX과 ko.TeX으로 대부분의 한국어 문서 관련 문제를 해결할 수 있으나, 일부 "자가 해결"에 의존해야 하는 경우도 있다. 특히 참고문헌 목록 등.

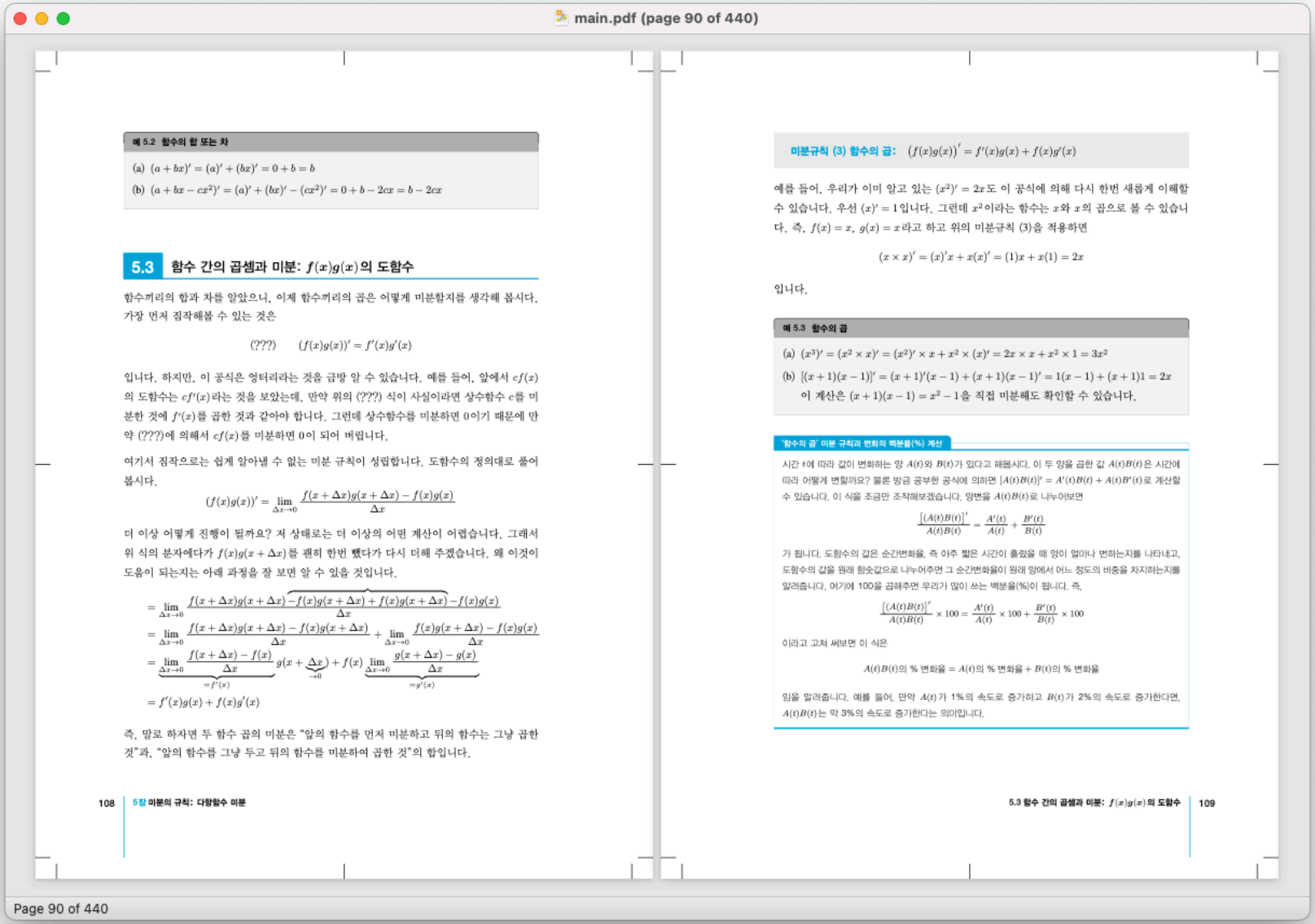
LaTeX 파일의 구조


cltbypark.Rmd
Knit on Save
Knit
Run
Source
Visual
---
title: "중심극한정리(Central Limit Theorem)의 예시"
author: "박재성"
date: "KTUG Conference 2015"
output:
pdf_document:
extra_dependencies: ["kotex","mystyles","amsmath","amssymb"]
latex_engine: xelatex
keep_tex: true

```{r random, cache=TRUE, echo=FALSE}
a<-1:999
n<-30
i<-50
var_a<-var(a)*((n-1)/n)
sd_a<-sqrt(var_a)
```
중심극한정리
상자 안에 `r min(a)`에서 `r max(a)`까지 숫자가 표시된
`r length(a)`개의 상태가 균질한 공(X)을 넣고 이를 특정한
9:19
중심극한정리(Central Limit Theorem)의 예시
R Markdo
Console
Terminal
Background Jobs
R 4.3.1 · /Volumes/Ebrt/Downloads/clt2rmd2/
R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
cltbypark.tex
조판
clxelatex
Macros
태그
Label
템플릿
% using explicit options in \includegraphics[width, height, ...]{}
\setkeys{Gin}{width=\maxwidth,height=\maxheight,keepaspectratio}
% Set default figure placement to htbp
\makeatletter
\def\fps@figure{htbp}
\makeatother
\setlength{\emergencystretch}{3em} % prevent overfull lines
\providecommand{\tightlist}{%
\setlength{\itemsep}{0pt}\setlength{\parskip}{0pt}}
\setcounter{secnumdepth}{-\maxdimen} % remove section numbering
\usepackage{kotex}
\usepackage{mystyles}
\usepackage{amsmath}
\usepackage{amssymb}
\ifLuaTeX
mystyles.sty
조판
clxelatex
Macros
태그
Label
템플릿
\setmainfont{TeX Gyre Pagella}
\setsansfont{TeX Gyre Heros}
\setmainhangulfont{Noto Serif CJK KR}
\setsanshangulfont{Noto Sans CJK KR}

문학적으로 작성된 소스로부터의 TeX 출판

TEX 출판의 가능성


main.pdf (page 90 of 440)
예 5.2 함수의 합 또는 차
(a) $(a+bx)' = (a)' + (bx)' = 0 + b = b$
(b) $(a+bx-cx^2)' = (a)' + (bx)' - (cx^2)' = 0 + b - 2cx = b - 2cx$
5.3 함수 간의 곱셈과 미분: $f(x)g(x)$의 도함수
함수끼리의 합과 차를 알았으니, 이제 함수끼리의 곱은 어떻게 미분할지를 생각해 봅시다. 가장 먼저 짐작해볼 수 있는 것은
(???) $(f(x)g(x))' = f'(x)g'(x)$
입니다. 하지만, 이 공식은 엉터리라는 것을 금방 알 수 있습니다. 예를 들어, 앞에서 $cf(x)$의 도함수는 $cf'(x)$라는 것을 보았는데, 만약 위의 (???) 식이 사실이라면 상수함수 c를 미분한 것에 $f'(x)$를 곱한 것과 같아야 합니다. 그런데 상수함수를 미분하면 0이기 때문에 만약 (???)에 의해서 $cf(x)$를 미분하면 0이 되어 버립니다.
여기서 짐작으로는 쉽게 알아낼 수 없는 미분 규칙이 성립합니다. 도함수의 정의대로 풀어 봅시다.
$$(f(x)g(x))' = \lim_{\Delta x \to 0} \frac{f(x+\Delta x)g(x+\Delta x) - f(x)g(x)}{\Delta x}$$
더 이상 어떻게 진행이 될까요? 저 상태로는 더 이상의 어떤 계산이 어렵습니다. 그래서 위 식의 분자에다가 $f(x)g(x+\Delta x)$를 괜히 한번 뺐다가 다시 더해 주겠습니다. 왜 이것이 도움이 되는지는 아래 과정을 잘 보면 알 수 있을 것입니다.
$$= \lim_{\Delta x \to 0} \frac{f(x+\Delta x)g(x+\Delta x) - f(x)g(x+\Delta x) + f(x)g(x+\Delta x) - f(x)g(x)}{\Delta x}$$
$$= \lim_{\Delta x \to 0} \frac{f(x+\Delta x)g(x+\Delta x) - f(x)g(x+\Delta x)}{\Delta x} + \lim_{\Delta x \to 0} \frac{f(x)g(x+\Delta x) - f(x)g(x)}{\Delta x}$$
$$= \underbrace{\lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}}_{=f'(x)} \underbrace{g(x+\underbrace{\Delta x}_{\to 0})}_{\to g(x)} + f(x) \underbrace{\lim_{\Delta x \to 0} \frac{g(x+\Delta x) - g(x)}{\Delta x}}_{=g'(x)}$$
$$= f'(x)g(x) + f(x)g'(x)$$
즉, 말로 하자면 두 함수 곱의 미분은 "앞의 함수를 먼저 미분하고 뒤의 함수는 그냥 곱한 것"과, "앞의 함수를 그냥 두고 뒤의 함수를 미분하여 곱한 것"의 합입니다.
108 5장 미분의 규칙: 다항함수 미분
미분규칙 (3) 함수의 곱: $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$
예를 들어, 우리가 이미 알고 있는 $(x^2)' = 2x$도 이 공식에 의해 다시 한번 새롭게 이해할 수 있습니다. 우선 $(x)' = 1$입니다. 그런데 x^2이라는 함수는 x와 x의 곱으로 볼 수 있습니다. 즉, $f(x) = x$, $g(x) = x$라고 하고 위의 미분규칙 (3)을 적용하면
$$(x \times x)' = (x)'x + x(x)' = (1)x + x(1) = 2x$$
입니다.
예 5.3 함수의 곱
(a) $(x^3)' = (x^2 \times x)' = (x^2)' \times x + x^2 \times (x)' = 2x \times x + x^2 \times 1 = 3x^2$
(b) $[(x+1)(x-1)]' = (x+1)'(x-1) + (x+1)(x-1)' = 1(x-1) + (x+1)1 = 2x$
이 계산은 $(x+1)(x-1) = x^2 - 1$을 직접 미분해도 확인할 수 있습니다.
'함수의 곱' 미분 규칙과 변화의 백분율(%) 계산
시간 t에 따라 값이 변화하는 양 $A(t)$와 $B(t)$가 있다고 해봅시다. 이 두 양을 곱한 값 $A(t)B(t)$은 시간에 따라 어떻게 변할까요? 물론 방금 공부한 공식에 의하면 $[A(t)B(t)]' = A'(t)B(t) + A(t)B'(t)$로 계산할 수 있습니다. 이 식을 조금만 조작해보겠습니다. 양변을 $A(t)B(t)$로 나누어보면
$$\frac{[A(t)B(t)]'}{A(t)B(t)} = \frac{A'(t)}{A(t)} + \frac{B'(t)}{B(t)}$$
가 됩니다. 도함수의 값은 순간변화율, 즉 아주 짧은 시간이 흘렀을 때 양이 얼마나 변하는지를 나타내고, 도함수의 값을 원래 함숫값으로 나누어주면 그 순간변화율이 원래 양에서 어느 정도의 비중을 차지하는지를 알려줍니다. 여기에 100을 곱해주면 우리가 많이 쓰는 백분율(%)이 됩니다. 즉,
$$\frac{[A(t)B(t)]'}{A(t)B(t)} \times 100 = \frac{A'(t)}{A(t)} \times 100 + \frac{B'(t)}{B(t)} \times 100$$
이라고 고쳐 써보면 이 식은
$A(t)B(t)$의 % 변화율 = $A(t)$의 % 변화율 + $B(t)$의 % 변화율
임을 알려줍니다. 예를 들어, 만약 $A(t)$가 1%의 속도로 증가하고 $B(t)$가 2%의 속도로 증가한다면, $A(t)B(t)$는 약 3%의 속도로 증가한다는 의미입니다.
5.3 함수 간의 곱셈과 미분: $f(x)g(x)$의 도함수 109
Page 90 of 440

출판 관행과 표준 LaTeX 원고

- 오리지널 소스(md)로부터 얻은 tex 파일을 원본 그대로 출판에 이용할 수 있는가?
- 출판사와 편집자의 조판 관행에 대한 고려가 사전에 필요한가?
- 출판 가능한 원고를 작성하기 위하여 오리지널 소스를 작성할 때 어느 정도 고려하여야 하는가?
- 내용(contents)이 결손없이 작성되어 있다면 디자인 요소는 대부분의 경우 최소한의 노력으로 보정할 수 있는가?
- 소위 '싱글 소스 솔루션'은 어느 정도나 가능할까? 그러기 위해서 갖추어야 할 요건은?

준비하여야 할 것(1)

클래스 현재 한국어 서적의 출판에는 oblivoir 클래스가 일반적으로 이용된다.

컴파일러 XeTeX이 가장 많이 사용된다. 주요한 이유는 폰트 때문이다.

폰트 한글 폰트 중에는 품위가 높은 것이 많지 않고, 쓸 만한 자유 글꼴이 드물다. 그래도 KoPub 폰트나 Noto 폰트는 충분히 인쇄에 활용할 만하며, 필요하다면 상업용 글꼴을 써야 한다.

그림 문서가 생성해내는 그림, 외부 그림은 PDF 또는 PNG 포맷으로 제작하여 별도로 준비한다. 만약 색도 인쇄를 고려한다면 분색(color separation)을 충분히 고려하여야 한다.

표 문서가 생성하는 표는 대부분의 경우 출판에 적합하지 않다. 새로 그리거나 보정하여야 한다.

소스 코드 리스팅

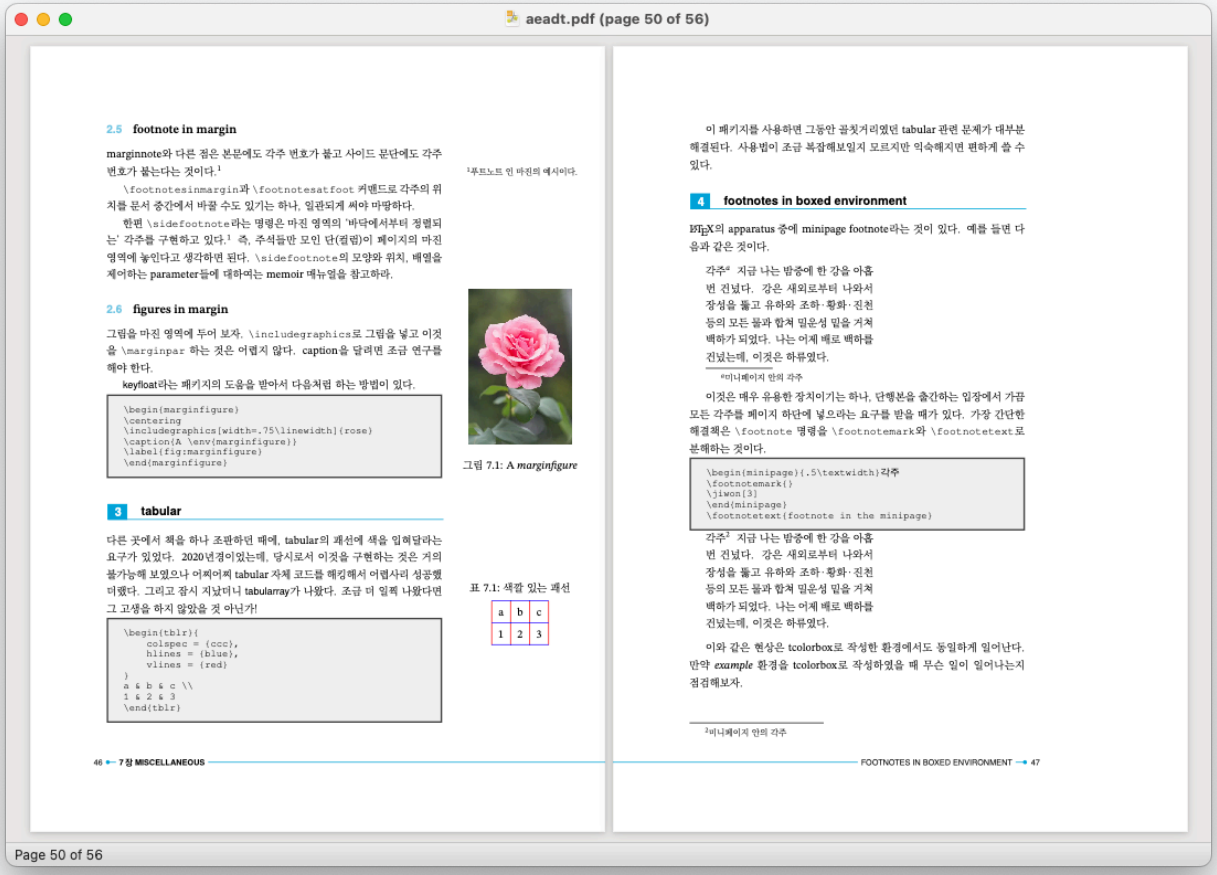
- 오리지널 문서의 code chunks는 '코드 블럭' 즉 verbatim으로 식자한다.
- 단순히 verbatim으로 충분한지 listings 패키지를 사용하도록 하고 있지는 않은지 검토해야 한다.
- listings는 유니코드 문자를 잘 처리하지 못하므로, 혹시 minted를 고려해야 하는지도 생각하자.
- 소스 코드 리스팅에서 가장 유의해야 할 사항은 판면과 행의 길이이다. 한 행이 판면을 넘어가는 경우가 빈발하므로 처음부터 주의하는 것이 필요하다. 행갈이를 자동화할 수도 있으나 의도하지 않은 결과를 얻을 가능성도 많다.

문서의 요소

- 섹셔닝. 장절 편제. 단행본이므로 적어도 \chapter 명령이 존재하도록 오리지널 소스를 작성하여야 할 것이다.
- 교차참조와 자동조사. 한국어 문서에서 자동조사의 자동화는 매우 중요한 일이다. 이를 자동화하는 패키지(예: ksjosaref)를 활용하는 것도 생각해보아야 한다.
- 떠다니는 개체. 표, 소스 코드 리스팅, 그림 등은 텍스트의 흐름과 안배되어 배치된다.
- 인용과 문헌목록. 현재 한국어 문서에서 가장 취약한 주제 중의 하나가 문헌목록이다. 이것은 biblatex을 "개인화"하여 어느 정도 해결할 수 있으나 그 표준 형태에 대하여 이해가 있어야 한다.
- 색인. 한국어 색인은 `komkindex`라는 유틸리티를 활용하는 것이 현실적이다. `texindy`나 `xindex`와 같은 떠오르는 도구가 있으나 출판에 활용하기에는 아직 준비가 부족하다.

샘플 단행본 예제

필자가 표준 단행본 샘플로 공개해둔 문서 aeadt(*Almost Everything About Document Styling*) 양식으로 qmd 문서를 출판에 적합하게 작성하는 예시.



결론과 제안

- 문서를 작성할 때, 목적의식을 명확히하여야 한다. weaving에 필요한 요소들을 사전에 점검한다.
- 비표준 명령은 최소한으로 줄인다. markdown으로 표현할 수 있는 것만을 활용하는 것이 최선이지만, 후에 최종 조정을 위하여 필요한 markup을 빠뜨리지 말하야 한다. 즉 code에 대해서만이 아니라 문서에 대해서도 comment를 충분히.
- 디자인 요소는 최후의 문제이다. code의 정합성과 내용의 완결성, 그리고 그 아름다움이 문석 작성의 핵심이다. 아이디어로 독자를 설득할 수 없다면 디자인이 화려한들 무슨 소용인가?
- 어떤 언어를 사용하든지 프로그래밍은 항상 이 코드를 남에게 이해시키려면 어떤 문장이 필요한가를 의식하면서 써야 한다.
- 내용이 충실하다면 그 훌륭한 내용을 멋진 출판물로 제작할 방법은 얼마든지 마련되어 있다.

Thank you

감사합니다.