

챗GPT 디지털 글쓰기

이광춘, 신종화

2023년 10월 09일

목차

제 1 장

글쓰기

글쓰기는 언어와 문자(Writing System) 언어와 문자 체계를 활용해 생각, 지식, 감정을 표현하는 창의적 과정이다. 글쓰기는 시각적 형태로 기록되어 말하기와 달리 시간과 공간의 제약을 받지 않는다. 과거 물리적 매체에 담겨 후대에 전달됨에 따라 일부 손상과 유실을 피할 수 없었다. 하지만, 현재는 다양한 디지털 문서 형태로 손실 없이 전달할 수 있게 되었다.

글쓰기는 단순한 문장 구성을 넘어서 사회, 문화, 과학, 기술까지 다양한 분야에서 중추적 역할을 수행한다. 법률 문서와 계약서를 작성하여 사회적 규칙과 약속을 명확히 하고, 과학기술 지식을 문서화하여 현대 문명 발전을 촉진하고, 글쓰기는 교육과 학문에서 지식과 정보의 전달 및 공유 수단으로 세대 간 지식 전승했다. 문화와 예술 분야에서는 창의적 글쓰기가 감정과 생각을 표현하는 동시에 사람들을 연결하는 다리 역할을 했고 심리와 정신 건강 분야에서는 자기 반성과 성찰을 통한 개인의 성장과 치유에도 글쓰기가 활용되고 있다.

이러한 다양한 역할을 넘어서, 글쓰기 목적과 형태도 지속적으로 발전하고 있다. 정보 전달, 감정 표현, 이야기 전달 등 기존 목적을 넘어서, 글쓰기는 수학과 과학 저작을 위한 `TeX` / `LATEX` 같은 수식 표현도 포함되고, 더 나아가, 최근에 기계와 의사소통을 위한 프로그래밍 언어도 글쓰기의 범주에 포함되고 있다. 교육용 파이썬이나 통계와 데이터 과학, 그리고 과학기술 저작을 위한 R 언어와 같은 프로그래밍 언어들이 글쓰기의 새로운 영역으로 빠르게 편입되고 있다.

현대 글쓰기의 특징 중 하나로 디지털 전환과 연결성을 꼽을 수 있고 그 중심에 인터넷과 웹의 출현이 있다. 인터넷이 전 세계 컴퓨터를 거대한 하나의 네트워크로 연결시켰다면, 웹은 인터넷 위에서 작동하는 정보 공간을 열었다. 인터넷과 웹은 글쓰기를 통한 커뮤니케이션을 글로벌 수준으로 확장시킴으로써 저자와 독자 사이 시공간적 제약을 크게 줄여, 다양한 문화와 언어 뿐만 아니라 그 이전 세대에서 경험하지 못한 정보와 지식의 융합을 가져왔으며, 실시간으로 정보를 주고받을 수 있어 신속한 글쓰기 측면도 크게 부각됐다.

과거 글쓰기 매체가 주로 종이에 기반했다면, 웹의 출현은 웹사이트, 블로그, 소셜 미디어(SNS) 등 다양한 플랫폼에서 글을 작성하고 공유할 수 있게 되면서 독자가 저자에게 직접 피드백을 줄 수 있는 인터랙티브한 환경도 제공함에 따라 글쓰기는 단방향적인 정보 전달에서 양방향적인 대화로 빠르게 진화했다. 텍스트가 종이 매체에서 주류를 형성했다면, 웹이 바꾼 글쓰기 생산과 소비환경은 이미지, 비디오, 오디오, 데이터, 그래프, 다이어그램, 표, 코드 등 다양한 요소를 통합하여 글을 작성하는 새로운 방식이 가능해졌다.

과거 글쓰기가 주로 물리적 매체, 즉 종이와 펜으로 대표되는 인쇄기술에 크게 의존해서 시간과 공간의 제약을 받았지만, 디지털 글쓰기 등장으로 이러한 제약과 위험은 크게 줄어들었다. 디지털 글쓰기는 웹사이트, 블로그, 소셜 미디어 등 다양한 플랫폼에서 텍스트, 이미지, 비디오, 오디오, 데이터 등 다양한 형태의 정보를 통합적으로 전달할 수 있고, 인터넷을 통해 전 세계 어디서든 접근하고 수정하고 공동저작할 수 있게 되었다.

디지털 글쓰기 발전은 인공지능 기술의 급속한 성장과도 밀접하게 연결되어 있다. 프로그래밍 언어를 통한 코딩은 이제 기계와의 상호 커뮤니케이션을 위한 새로운 형태의 글쓰기로 간주되고 있다. 글쓰기는 단순한 정보 전달이나 감정 표현을 넘어, 고도화된 현대 사회의 기술적, 사회적, 제도적 인프라를 구축하고 유지하는 데에 필수적인 역할을 하고 있기 때문에 글쓰기 없이는 현대 사회의 복잡한 시스템과 그것을 지탱하는 기술이 존재할 수 없으며, 따라서 글쓰기는 현대 사회 생존에 있어서 불가결한 요소라고 할 수 있다.

1.1 문서로 보는 민주주의

글쓰기는 권력분립과 민주주의 체계에서 중요한 역할을 하고 있으며, 인공지능의 부상이 현체계를 더욱 복잡하게 만들지만 동시에 새로운 가능성도 열고 있다. 따라서, 챗

GPT로 촉발된 인공지능 사회에서 글쓰기 중요성은 더욱 강조될 수 밖에 없다.

삼권분립 원리는 국가의 권력을 입법, 사법, 행정으로 나누어 각기 다른 역할을 수행하게 함으로써 권력의 독점과 남용을 방지한다. 이 원리는 국민의 권리와 자유를 보장하는 국가 조직의 핵심 원칙으로, 인류 사회가 발전하면서 다양한 형태로 구현되어 왔다. 삼권분립 원리에서 글쓰기는 중추적인 역할을 차지하고 있다. 입법권은 법률을 '작성'하고, 행정권은 법률을 '실행'하며, 사법권은 법률을 '해석'한다. 이 모든 과정은 문서와 글쓰기에 근간을 두고 있어, 글쓰기는 삼권분립 체계를 원활하게 작동시키는 뼈대라고 할 수 있다.

최근 인공지능(AI)의 급속한 발전은 삼권분립 체계에 깊은 영향을 미치고 있다. 입법 분야에서 인공지능이 복잡한 법률 문서의 작성과 데이터 분석을 통한 새로운 법률의 필요성을 예측하고 있으며, 사법 분야에서는 판례 분석과 법률 문서 해석에 있어 뛰어난 능력을 보여, 판사와 변호사의 의사결정 과정을 적극적으로 지원하고 있다. 행정 분야에서는 인공지능이 정책 결정에 필요한 데이터 분석을 통해 통찰력을 제공하고, 코딩은 기계와 상호 커뮤니케이션 촉진함으로써 행정 서비스 효율성을 높이고, 국민에게 더 나은 서비스를 제공하도록 기여하고 있다.

인공지능의 부상은 삼권분립 체계에서 각 분야의 작동 방식을 혁신적으로 변화시키며, 민주주의와 국가 운영에 새로운 가능성은 제시하고 있으나 항상 기술 발전은 윤리적, 사회적 측면에서도 신중한 검토가 필요하다. 인공지능의 결정 과정이 투명하지 않거나 데이터에 편향이 있을 경우, 민주주의 원칙에 위배되는 결과를 초래할 수 있어 문제점을 신중하게 검토하고 해결책을 마련하고 도입을 추진해야 된다.



그림 1.1: 글쓰기와 삼권분립

1.2 글쓰기 역사

점토판과 쐐기를 이용한 문자 작성에서 시작하여 중세시대에는 붓과 머루를, 서양에서는 잉크와 새깃털펜을 사용해 저작을 했다. 타자기의 발명은 개인 저작을 가능하게 만들었고, 전동타자기의 등장은 문서 작성의 비용을 절감하고 품질을 향상시키며 출판 과정을 신속화했다. 제2차 세계대전 이후에는 군용 애니악(ENIAC) 컴퓨터가 민간에 보급되어 IBM이 1964년에 워드 프로세서를, 1969년에 저장 장치를 시장에 출시했다. 이로써 기계식 저작 방식에서 디지털 저작 방식으로 전환의 기반이 마련되었다. 1980년대 전후로 마이크로소프트 워드 1.0, 워드 퍼펙트, 아래한글과 같은 위치워(WYSIWYG) 방식이 일반인에게 큰 인기를 끌었다면, 문서를 구조적으로 작성하여 컴파일하는 위치윔(WYSIWYM) 방식의 \TeX / \LaTeX 이 비슷한 시기에 과학기술 전문가들 사이에 자리를 잡아갔다.



그림 1.2: 글쓰기 저작도구 진화과정

1.2.1 문서 저작 패러다임

아래한글이나 MS 워드 같은 워드 프로세서는 **위치위그(WYSIWYG: What You See Is What You Get)** 방식을 기반으로 한다. 이 방식은 화면에 보이는 서식이 입력된 텍스트가 최종 출력물과 동일하게 나오는 직관적인 특성을 가진다. 이러한 장점 덕분에 타자기나 컴파일을 필요로 하는 다른 문서 저작 방식에 비해 경쟁력을 보이며 문서 저작의

주류 소프트웨어로 자리잡았다.

반면, **위지웜(WYSIWYM: What You See Is What You Mean)** 방식의 대표적인 예는 **LATEX**(레이텍)이다. 이 방식은 구조화된 형태로 문서를 작성한 후 컴파일을 통해 출판 가능한 PDF 파일을 생성한다. **LATEX**의 주요 장점은 수식, 그래프, 표 등 다양한 구성요소를 미려하게 표현할 수 있다는 것이다. 또한, “문학적 프로그래밍(literate programming)” 패러다임을 통해 텍스트와 코드를 함께 담을 수 있고 위지워그 방식 워드프로세서 보다 문서가 복잡할수록 진가를 보여준다.

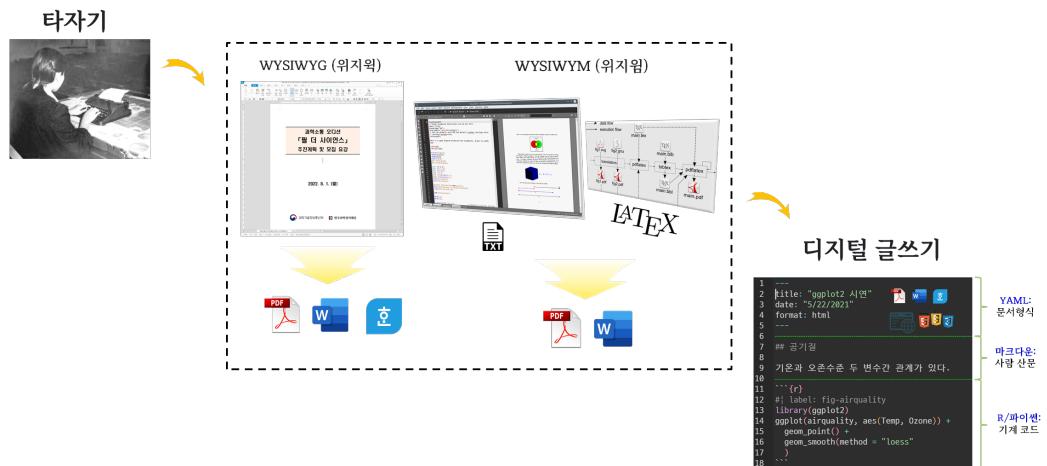


그림 1.3: 문서 저작 패러다임

최근 문서저작 도구의 패러다임은 눈에 띠는 변화를 겪고 있다. 과거에는 일반 사용자를 위한 위지워그(WYSIWYG)와 과학기술 전문가를 위한 위지웜(WYSIWYM)이라는 두 가지 주요 패러다임으로 문서저작 도구가 명확하게 구분되었다. 그러나 최근에는 챗GPT가 촉발한 생성형 AI 기술이 위지워그 패러다임에 본격적으로 도입되어, 워드프로세서부터 엑셀, 파워포인트, 데이터베이스, 프로그래밍, 그리고 멀티미디어 저작까지 다양한 분야에서 발전이 가속화되고 있다. 위지웜 패러다임도 마크다운, 팬독(pandoc), R/Python, **LATEX** 엔진 등 단위기술 발전과 함께 생성형 AI기술이 적극 반영된 RStudio, 주피터, VS코드와 같은 통합개발환경(IDE)의 발전을 통해 두 패러다임 경계가 점차 흐릿해지고 있다.

문서저작의 복잡성이 증가함에 따라 재현성, 추적성, 협업, 코딩, 버전 제어, 자동화, 생산성이 중요한 저작 요소로 부상하고 있다. 이러한 복잡한 문서저작 문제에 대한 해결책으로 문학적 프로그래밍 패러다임이 주목받고 있다.

1.2.2 문서 도구 역사

사람과 의사소통하는 텍스트와 기계와 의사소통하는 프로그래밍 언어를 작성하는 방식은 서로 다르다. 텍스트는 문서를 작성하고, 프로그래밍 언어는 코드를 작성한다는 점에서 차이가 있지만, 모두 문서를 작성한다는 점에서 공통점이 있다. 문서 작성과 코드 통합은 오래 전부터 컴퓨터 과학과 데이터 과학에서 중요한 주제였다. 그 결과 다양한 해법이 제시되고 도구도 개발되었다.

도널드 크누쓰 교수가 1978년 수식이 포함된 과학문서 작성에 특화되어 **TeX**를 개발하였고, 1984년 문학적 프로그래밍 개념을 제안하면서 텍스트와 코드를 섞어 문서와 프로그램을 동시에 작성할 수 있다는 것을 최초로 선보였다. 1985년에 레슬리 램포트가 **TeX**을 기반으로 **L^AT_EX**을 개발하여 수학과 과학 분야에서 널리 사용되는 문서저작 도구로 자리잡았다. [knuth84, spivak1990joy]

1988년에 울프람 박사가 매쓰매티카 노트북(Mathematica Notebooks)을 개발하여 수학과 과학 연구를 위한 고급 작업 환경을 제공했지만 한동안 특별한 도구나 개념의 진전은 없었지만 2000년대 들어서면서 2001년에 페르난도 페레즈가 파이썬 대화하여 IPython 을 개발했고, 2003년에 이맥스(Emacs) 텍스트 편집기에서 동작하는 작업 관리 및 문서 작성 도구인 이맥스 Org-mode를 개발했다. 2004년에 존 그루버가 웹 문서를 쉽게 작성하고 표현할 수 있는 경량 마크업 언어 마크다운(Markdown)을 공개했다. 2006년부터 본격적으로 팬독(Pandoc) 개발이 시작되면서 다양한 마크업 언어 간 자유로운 변환이 가능하게 되었다.

페르난도 페레즈는 2011년 파이썬 기반 iPython Notebook을 개발했고, 곧 이어 2012년 R 기반 knitr 개발이 본격화되었다. 2014년에 페르난도 페레즈가 다시 한번 다양한 프로그래밍 언어를 지원하는 주피터 프로젝트(Project Jupyter)를 주도하고 있으며, 2020년 RStudio로 잘 알려진 포짓(Posit) J.J. 알레어 대표가 과학기술 문서 작성과 데이터 과학에 특화된 쿼토(Quarto)를 개발하여 지속적으로 새로운 기능을 선보이고 있다.

연도	도구	개발자
1978	TeX	Donald Knuth
1984	Literate Programming	Donald Knuth
1988	Mathematica Notebooks	Stephen Wolfram
2001	IPython	Fernando Perez

2003	Emacs org-mode	Carsten Dominik
2004	Markdown	John Gruber
2005	Sage Notebook	William Stein
2006	Pandoc	John MacFarlane
2009	GitHub Flavored Markdown	Tom Preston-Werner
2011	iPython Notebook	Fernando Perez
2012	knitr	Yihui Xie
2014	Project Jupyter	Fernando Perez
2020	쿼토 (Quarto)	J.J. Allaire

서문

끊임없이 진화하는 인간 커뮤니케이션 환경에서 디지털 시대의 도래는 표현과 연결의 새로운 차원을 가져왔습니다. 디지털 불평등 격차해소를 추구하고 디지털 영역의 무한한 잠재력을 포용하는 작가로서, “디지털 글쓰기”를 여러분께 소개합니다. 이 책은 창의성, 기술, 인간 정신이 융합된 공간인 디지털 글쓰기의 광활하고 길들여지지 않은 경계를 탐험하고자 하는 사람들을 위한 안내서이자 동반자 역할을 합니다.

새 시대에 등장한 다양한 플랫폼, 형식, 도구를 살펴보면서 온라인 글쓰기의 세계가 제시하는 독특한 기회와 도전 과제에 대해 자세히 살펴볼 것입니다. 이 책은 단순한 기술 매뉴얼이 아니라 말과 아이디어의 힘, 그리고 디지털 광야에 과감히 도전하는 사람들을 기다리는 무한한 가능성에 대한 찬사입니다.

디지털 시대 글쓰기 진화를 살펴보고, 스토리를 만들고, 공유하고, 소비하는 방식에 대한 기술의 혁신적인 영향을 추적하는 것으로 여정을 시작할 것입니다. 재현 가능한 과학기술 문서작성과 대량생산부터 인터랙티브 서사와 멀티미디어 경험에 이르기까지 새로운 형태의 디지털 스토리텔링의 출현에 대해 논의할 것입니다.

온라인 글쓰기의 세계를 더 깊이 파고들면서 자신만의 목소리와 스타일을 개발하고, 전통적인 글쓰기 기법을 디지털 플랫폼의 고유한 요구에 맞게 조정하며, 소셜 미디어의 힘을 활용하여 독자와 소통하고 작품을 중심으로 커뮤니티를 구축하는 것의 중요성에 대해서도 살펴볼 것입니다.

익명성은 여전히 개인적인 선택이지만, 디지털 영역에서 익명성이 가져다주는 기회와 도전을 인식하는 것은 필수적입니다. 개인정보 보호와 공개성 사이의 미묘한 균형, 그리고 온라인에서 콘텐츠를 제작하고 공유할 때 고려해야 할 윤리적 고려 사항과 책임에 대해 논의할 것입니다.

디지털 글쓰기의 세계를 탐색하는 데 필요한 지식과 도구뿐만 아니라 나만의 목소리를 세상과 공유할 수 있는 영감과 용기를 얻으시기 바랍니다. 광활한 사이버 공간에 여러분의 글이 올려 펴져 경계를 넘어 전 세계 독자들의 마음을 움직일 수 있도록 모험을 떠나보세요.

디지털 오디세이에 오신 것을 환영합니다. 모험을 시작하세요.

제 2 장

챗GPT 글쓰기 맛보기

종이와 펜으로 글을 쓸 때 글쓰기는 주로 생각이나 정보, 감정을 언어의 형태로 표현하는 과정으로 가장 기본적이고 전통적인 글쓰기의 형태로 볼 수 있다. 저자가 머릿속에 있는 아이디어나 메시지를 구조화하여 문장과 단락으로 만드는 과정에서 문법, 어휘, 문장 구조 등 언어의 다양한 요소가 활용된다. 전통적인 글쓰기는 디지털 글쓰기에서 느끼기 어려운 독특한 느낌과 집중도가 있어 펜을 종이에 대고 쓰는 물리적인 행위 자체가 생각을 정리하고 창의성을 불러일으키는 경우가 많고 디지털 장비의 방해 없이 좀 더 몰입하여 글을 쓸 수 있다는 장점도 있다.

전통적인 글쓰기에서 문법, 어휘, 문장 구조 등 언어의 다양한 요소에 집중하는 이유 중 하나는 인쇄출판 비용 문제가 크다. 인쇄출판을 할 경우, 재료와 인력, 시간 등 다양한 비용이 발생하기 때문에, 처음부터 정확하고 명료한 표현을 사용하여 수정이나 재인쇄 필요성을 줄여 비용을 절감해야 하기 때문에 저자가 더 신중하게 글을 쓰도록 만들며, 그 결과로 문법이나 어휘 선택, 문장 구조 등에 더 많은 주의를 기울이게 된다. 인쇄된 글은 디지털 글과 달리 쉽게 수정할 수 없기 때문에, 처음부터 높은 수준의 정확성과 명료성이 요구되기 때문에 작성자로 하여금 더 깊게 생각하고 더 철저하게 검토하는 습관을 들이게 하며, 이러한 과정을 통해 더 높은 품질의 글이 탄생하게 되었다.

따라서, 글쓰기는 모든 사람이 배우는 기초적인 능력이지만, 인쇄출판의 높은 비용 때문에 실제로 글을 발표하고 널리 퍼뜨릴 수 있는 사람은 선택된 소수에 불과하기 때문에 빈익빈, 부익부 문제가 심해지고 있다. 즉, 자본과 기회가 있는 소수의 사람들만이 자신의 아이디어나 의견을 널리 알릴 수 있는 인쇄출판 플랫폼을 갖게 된 것이다. 이로

인해 정치적, 사회적, 경제적, 과학기술 격차가 더욱 심화되며, 다수의 목소리가 제대로 들리지 않는 문제가 발생한다. 이러한 상황은 인터넷과 웹의 대중화로 디지털 글쓰기가 보급됨에 따라 어느 정도 해소되고 있지만, 저작에 해당되는 디지털 글쓰기와 저작물의 인쇄출판과 유통에 대한 접근성 문제는 여전히 해결되어야 할 중요한 과제로 남아 있다.

2.1 챗GPT 등장

챗GPT 등장은 글쓰기 패러다임을 혁신적으로 바꾸고 있다. 전통적인 글쓰기는 문법, 어휘, 문장 구조 등 언어의 다양한 요소를 활용하여 아이디어나 메시지를 구조화하는 데 주력했고, 전통적인 글쓰기에서 저작은 종이와 펜, 출판은 워드 프로세서 도구로 필요한데, 디지털 도구에 익숙하지 않은 사람들에게는 글쓰기 자체가 큰 부담이 되었다.

그러나 챗GPT는 뛰어난 자연어 처리 능력 덕분에 저자는 복잡한 문법이나 어휘 없이도 자신의 생각을 쉽게 표현할 수 있을 뿐만 아니라, 챗GPT는 다양한 형태의 콘텐츠, 예를 들어 수식, 표, 그래프, 다이어그램, 코드 등을 쉽게 생성하고 편집할 수 있어, 글쓰기 범위가 전통적인 텍스트를 넘어 다양한 미디어로 확장되고 있다.

펜과 종이로 저작하여 종이 매체를 중심으로 인쇄출판이 주류를 이뤘던 시대를 워드 프로세서로 저작하여 웹사이트에 디지털 출판하는 시대로 바뀌었던 것에 비견될 만큼 챗GPT는 디지털 글쓰기의 패러다임을 혁신적으로 바꾸고 있다. 챗GPT Text-to-X 기능은 글쓰기 가능성을 더욱 확장시켜 저자는 아이디어나 메시지를 표현하고 전달하는데 단순한 텍스트 입력을 넘어 수식, 그래프, 표, 다이어그램, 코드 등 다양한 형태의 콘텐츠를 쉽게 생성할 수 있게 됨에 따라, 글쓰기 경계를 훨씬 넓히게 되었다. 챗GPT 시대 누구나 복잡한 도구나 전문 지식 없이도 자신의 아이디어나 정보를 풍부하게 표현할 수 있게 되었다.

2.2 글쓰기 진화과정

글쓰기 역사는 인류 문명과 밀접하게 연결되어 있다. 고대에는 점토판과 쌈기를 사용하여 기록을 남겼다. 중세에 이르러 필경사가 등장하고, 붓과 머루, 잉크와 깃털펜이 사용되기 시작했다. 근대에 들어서면 타자기가 등장하고, 1868년에 첫 타자기가, 1902년에 전동타자기가 개발되었다.

컴퓨터 등장은 글쓰기 패러다임을 또 다시 바꾸었다. 1946년에 에니악 컴퓨터가, 1964년에 IBM 워드프로세서 MT/ST가 등장했다. 이후에는 IBM 저장장치 MagCards가 1969년에, TeX가 1978년에, LaTeX이 1984년 등장하면서 디지털 글쓰기 기술적 토대가 탄탄해졌다.

1983년에 마이크로소프트 워드 1.0의 출시로 개인용 컴퓨터에서도 글쓰기가 가능해지면서 디지털 글쓰기의 새로운 시대가 열렸다. 그 후로 프로그래밍 언어와 글쓰기가 통합되기 시작했고, 마크다운, R, 파이썬 등을 활용한 글쓰기가 일반화되면서 문학적 프로그래밍(Literate Programming)이라는 새로운 형태의 글쓰기를 가능하게 했다. 최근에는 챗GPT의 등장으로 문학적 프로그래밍이 한 단계 더 진화하고 있다. 챗GPT는 자연어 처리 능력을 활용하여 코드와 텍스트, 그리고 다양한 미디어 요소를 통합적으로 다룰 수 있어, 디지털 글쓰기의 패러다임을 혁신적으로 바꾸고 있다.



그림 2.1: 챗GPT 디지털 글쓰기 진화과정

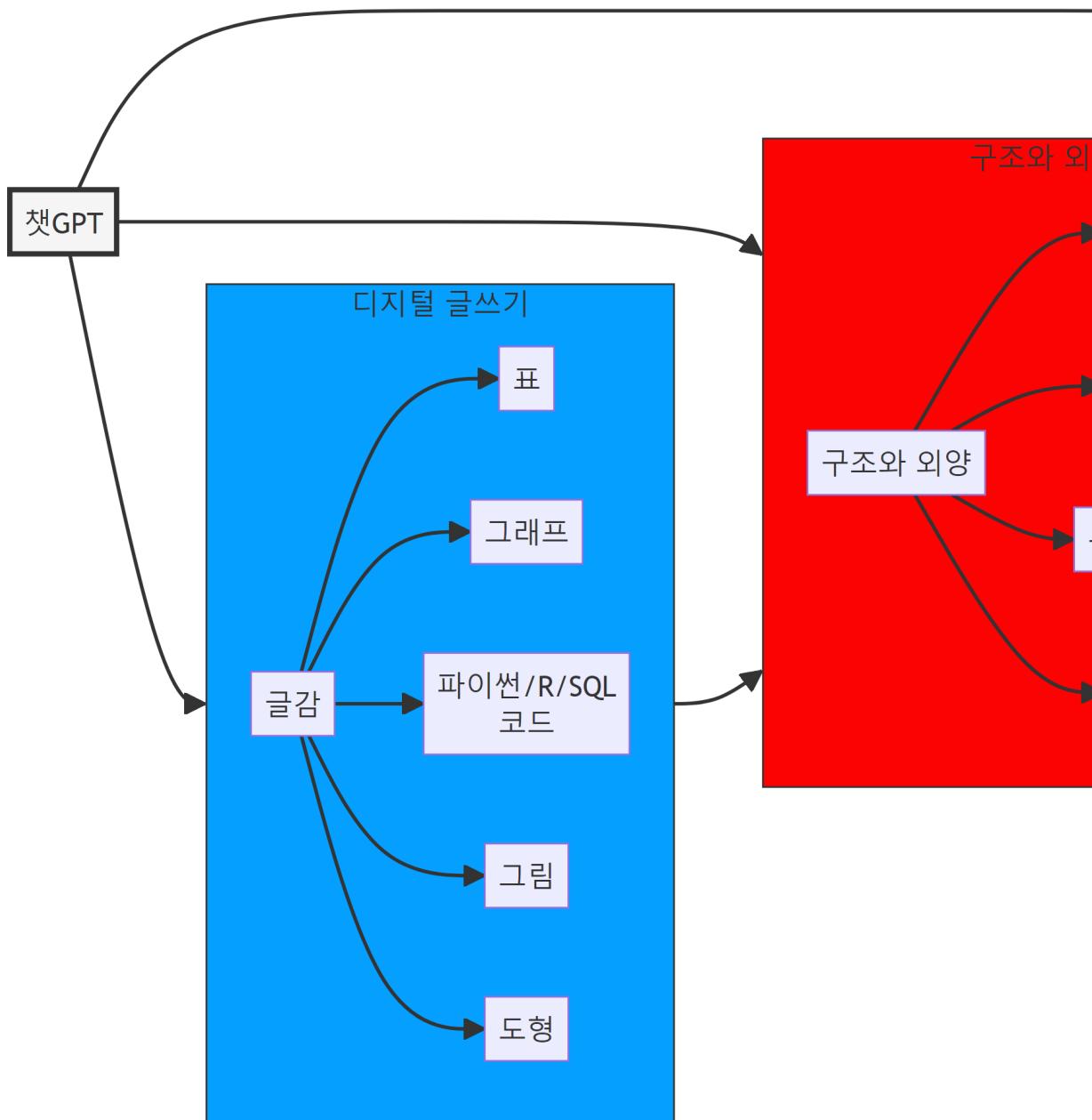
2.3 글쓰기 작업흐름

챗GPT 디지털 글쓰기의 일반적인 작업흐름은 더 세분화할 수 있지만, “디지털 글쓰기”, “구조와 외양”, “배포와 공유”로 구분할 수 있다.

첫 번째 단계인 “디지털 글쓰기”에서는 글감을 시작점으로 하여 다양한 형태의 콘텐츠를 생성하는데 수식, 표, 그래프, 파이썬/R/SQL 코드, 그림, 도형 등이 포함된다. 자연어로 챗GPT를 통해 다양한 저작물 구성요소를 쉽고 빠르게 생성하고 편집할 수 있다.

두 번째 단계인 “구조와 외양”에서 문서의 전반적인 구조와 서식을 결정한다. 문서의 구조, 서식, 레이아웃, 그리고 참조 등이 확립되는데 챗GPT 부조종사(Copilot)이 유령 텍스트(Ghost Text) 기능을 통해 자연어로 저자 작업을 더욱 효율적으로 지원한다.

마지막 단계인 “배포와 공유”에서 작성된 저작물이 독자에게 다양한 매체를 통해 전달 한다. 독자가 반드시 사람일 필요는 없고 독자가 저작자 자신일 수도 있으며, 심지어 기계일 수도 있다. 챗GPT 배포와 공유 과정에서 발생되는 다양한 기술적 문제를 해결해 주는 공동 저작자이자 동료로서 디지털 글쓰기 전체적인 흐름을 완성한다.



2.4 챗GPT 디지털 글쓰기 사례

사전 자료조사를 통해서 한국영화 10선 중 국제시장과 펭귄 데이터를 통해 흥행과 동시에 현재 당면한 ESG 문제를 다룬 뭉클한 감동을 주는 영화 시놉시스 작성 목표로

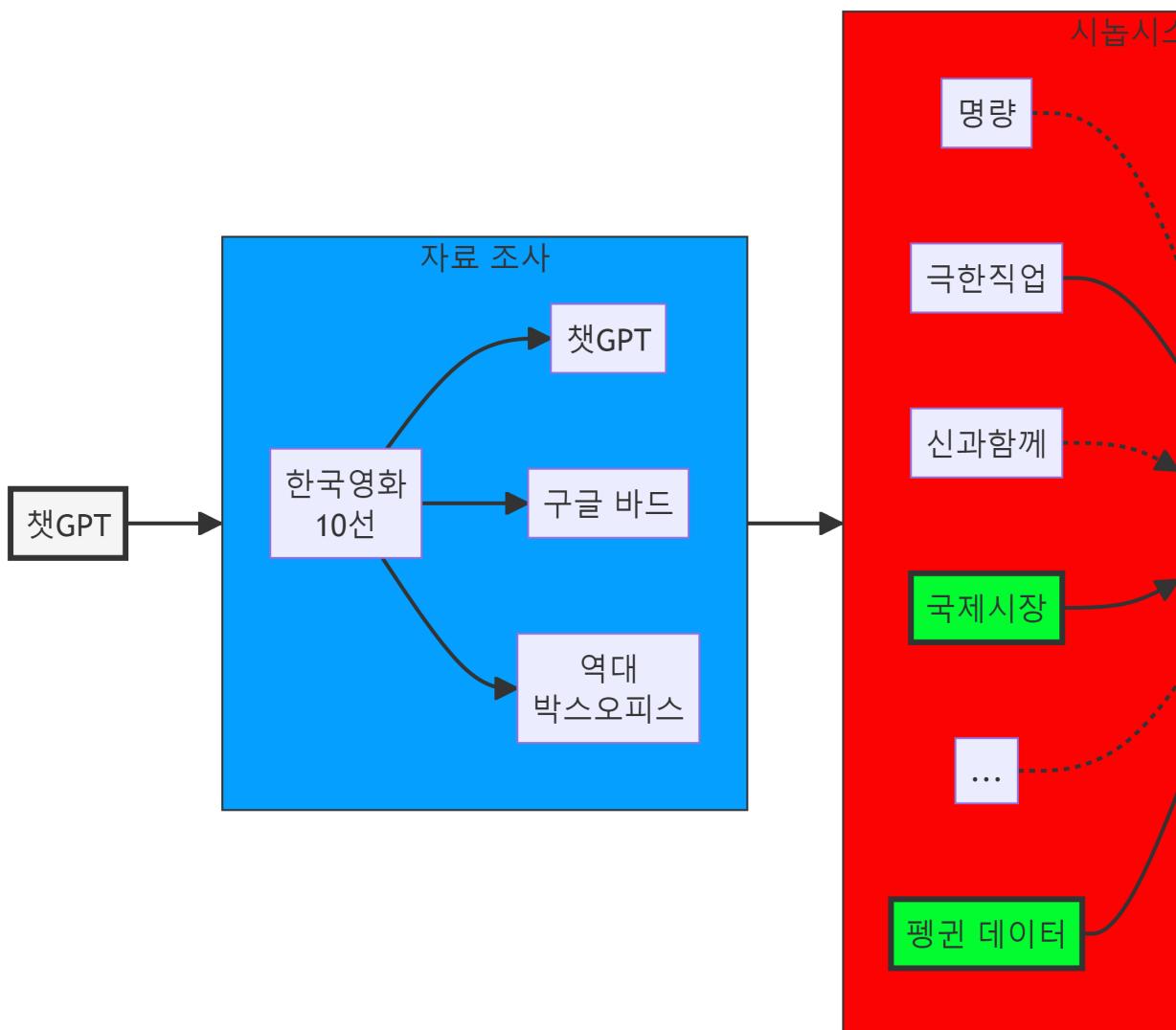
하는 저작물을 작성해보자.

챗GPT 디지털 글쓰기를 활용하여 영화 시놉시스를 작성하는 과정은 다음과 같다.

1. **디지털 글쓰기 단계:** 먼저, “국제시장”과 “펭귄 데이터”를 통해 얻은 글감을 바탕으로 영화의 주제와 구조를 설정한다. 챗GPT를 사용하여 흥행 가능성과 ESG 문제에 대한 데이터를 분석하여 표와 그래프로 시각화한다.
2. **구조와 외양 단계:** 시놉시스 구조를 결정하고, 적절한 서식과 레이아웃을 적용한다. 문서 참조나 인용 등 챗GPT를 활용하여 자연어로 쉽게 추가할 수 있다.
3. **배포와 공유 단계:** 작성된 시놉시스를 다양한 플랫폼과 매체를 통해 공유한다. 디지털 출판 배포과정에서 챗GPT는 기술적 문제를 해결함은 물론 방향도 제시하여 독자가 사람이든 기계든 쉽게 출판저작물에 접근할 수 있도록 지원한다.

목표한 “영화 시놉시스”를 챗GPT 디지털 글쓰기로 작업한 결과물은 다음과 같이 복잡한 데이터 분석부터 감동적인 내러티브까지 한 번에 작업할 수 있다.

영화 제목을 “펭귄의 꿈: ESG로 미래를 바꾸다”로 설정하고, 주인공은 한국에서 ESG 문제에 관심을 가진 젊은 연구원으로 한다. 그는 남극의 펭귄 데이터를 분석하여 지구의 미래를 예측하고, 이를 통해 사람들에게 ESG의 중요성을 알리려고 한다. 영화는 그의 여정과 성장, 그리고 마침내 세계를 바꾸는 끊임없는 결말로 끝난다.



2.4.1 챗GPT 글감 작성

실제 데이터가 필요한 경우 영화진흥위원회에서 매년 발표하는 “한국영화연감”의 영화별 흥행기록 **역대 박스오피스 (공식통계 기준)**에서 정확한 데이터를 확인할 수 있다. 챗GPT는 프롬프트에 대한 답변을 생성하는 언어 모형으로 챗GPT는 데이터베이스나 실시간 정보에 접근하지 않고 기존에 학습된 데이터를 기반으로 답변을 제공한다. 따라서, 100% 사실은 아니지만 빠르게 답변을 제공받아 시놉시스 제작에 활용하는 경우 대략 8~90% 정확도를 보이는 업무에 챗GPT 활용은 가능하다. 예를 들어, “관객 기준

으로 가장 많은 관람객을 갖는 한국 영화 상위 5개 알려주세요”라는 프롬프트에 대한 답변을 챗GPT로 받아 다음과 같이 마크다운 형태로 가공하여 후속 작업에 활용한다.

i 프롬프트

관람객 기준으로 가장 많은 관람객을 갖는 한국 영화 상위 5개 알려주세요 시놉시스는 200자 길이로 작성해주세요. 출력형식은 다음과 같은 형태로 마크다운 소스파일 형태로 출력해 주세요.
|순위|영화명|연도|감독|배급사|관객수|시놉시스|

출력결과

죄송합니다, 제 정보는 2021년 9월까지로 제한되어 있어서 그 이후의 데이터에 대한 정확한 정보를 제공할 수 없습니다. 그러나, 2021년 9월까지 기준으로 가장 많은 관람객을 가진 한국 영화 상위 5개는 다음과 같습니다.

순위	영화		배급			시놉시스
	명	연도	감독	사	관객수	
1	명량	2014	김한민	씨제이이이엔엠	1761만	1592년 임진왜란, 이때 조선은 거대한 전쟁의 중심에 있었다. 여기에서 조선 최고의 장군 이순신이 일어나 대역적인 전투를 벌인다.
2	극한 직업	2019	이병현	씨제이이이엔엠	1626만	각종 범죄 현장에서 물리적인 고난과 위험을 감수하며 사건을 해결하는 특수 대원들의 이야기. 그러나 그들의 직업은 상상을 초월하는 극한의 직업, 치킨집 사장이었다.

3	신과 함께-인과연	2018	김용화	롯데엔터테인먼트	1441만	살아있는 동안 저지른 죄에 대해 사후세계에서 재판을 받는 과정을 그린 영화. 사망 후 49일 동안 7개의 지옥에서 7번의 재판을 받아야 하는데, 이 과정을 수행하는 천사들의 이야기.
4	국제 시장	2014	윤제균	씨제이엔터테인먼트	1426만	한국 전쟁을 배경으로 한 영화로, 한 남자가 가족을 지키고, 살아남기 위해 펼쳐나가는 감동적인 사랑과 희망의 이야기를 담고 있다.
5	어벤져스: 엔드게임	2019	안소니 루소, 조 루소	월트 디즈니 코리아	1393만	타노스에 의해 절반의 생명이 사라진 채 패배한 어벤져스. 남은 멤버들은 어떻게든 가족과 친구들을 다시 데려오기 위한 모든 것을 걸고 마지막 전투에 나선다.

2.4.1.1 실제 데이터

챗GPT는 2021년 9월까지 실제 인터넷에 공개된 데이터를 바탕으로 학습한 언어모형이다. 따라서 최신 데이터는 반영하지 못하고 과거 학습한 데이터도 모든 과거를 대표하는 것도 아니다. 잘 작성된 프롬프트를 챗GPT 언어모형과 결합하여 나온 텍스트는 100% 사실이 아니지만 대략적인 흐름을 파악하는 데는 충분하다. 영화진흥위원회에서 매년 발표하는 “한국영화연감”의 영화별 흥행기록 역대 박스오피스(공식통계 기준)에서 통계적으로 정확한 한국 영화 10개를 뽑아보면 대략 일치함을 확인할 수 있다. 시간이 지나면서 챗GPT에 대한 활용이 늘어나고 언어모형이 고도화되면 이러한 통계적인 정확도는 더욱 높아질 것으로 보인다.

순위	영화명	개봉일	매출액	관객수	국적	배급사
1	명량	1406646000	1,357	1,761	한국	(주)씨제이이엔엠

2	극한직업	1548169200	1,396	1,626	한국	(주)씨제이이엔엠
3	신과함께-죄와 벌	1513695600	1,157	1,441	한국	롯데쇼핑(주)롯데엔터테인먼트
4	국제시장	1418742000	1,109	1,426	한국	(주)씨제이이엔엠
5	어벤져스: 엔드게임	1556031600	1,222	1,393	미국	월트디즈니컴퍼니코리아
6	겨울왕국 2	1574262000	1,148	1,375	미국	월트디즈니컴퍼니코리아
7	아바타	1260975600	1,284	1,362	미국	해리슨앤컴퍼니, 이십세기폭스
8	베테랑	1438700400	1,052	1,341	한국	(주)씨제이이엔엠
9	괴물	1153926000	0	1,302	한국	(주)쇼박스
10	도둑들	1343142000	937	1,298	한국, 홍콩	(주)쇼박스

2.4.1.2 펭귄 데이터셋

미국에서 “George Floyd”가 경찰에 의해 살해되면서 촉발된 “Black Lives Matter” 운동은 아프리카계 미국인을 향한 폭력과 제도적 인종주의에 반대하는 사회운동이다. 한국에서도 소수 정당인 정의당에서 기사로 낼 정도로 적극적으로 나서고 있다.¹

데이터 과학에서 최근 R.A. Fisher의 과거 저술한 “The genetical theory of natural selection” [edwards2000genetical] 우생학(Eugenics) 대한 관점이 논란이 되면서 R 데이터 과학의 첫 데이터셋으로 그가 만들었다고 하는 붓꽃 iris 데이터를 대안적 데이터, 즉 펭귄 데이터로 대체하는 움직임이 활발히 전개되고 있다. palmerpenguins [penguin2020] 데이터셋이 대안으로 많은 호응을 얻고 있다. [AbdulMajedRaja2020], [Levy2019]

2.4.1.3 시놉시스 생성

챗GPT를 활용하면 더욱 정교하고 창의적인 답변 및 콘텐츠를 제작할 수 있다. 예시로 “영화의 시놉시스와 펭귄에 관한 데이터를 결합하여 새로운 영화 시놉시스를 구성해 주세요”라는 프롬프트는 챗GPT에게 서로 다른 주제를 통합해 새로운 작품 시놉시스 제작을 지시내린다. 챗GPT는 제공된 프롬프트를 분석하고 이해한 뒤 새로운 시놉시스를 만들어낸다. 특히 눈에 띄는 점은 챗GPT가 새로운 시놉시스를 대략 10초 이내 빠르게 생성한다는 것이다.

¹ 정환봉 기자 (2020-06-15), “여당 의원 176명 중 누가?…차별금지법 발의할 ‘의인’을 구합니다”, 한겨레 신문

i 프롬프트

영화 시놉시스와 펭귄 데이터가 백틱(“)으로 감싸져 있습니다. 다음 영화 시놉시스와 데이터를 하나로 합하여 전혀 새로운 영화 시놉시스를 작성해주세요.

“첫번째 영화: 국제시장

덕수(황정민)의 가족은 흥남 철수 때 부산 국제시장에 내려온 피난민이다. 전쟁통에 아버지를 잃은 덕수는 졸지에 어머니와 두 동생의 생계를 책임지게 됐다. 동생의 학비와 생활비를 벌기 위해 자신의 꿈을 버린 채 친구 달구(오달수)와 함께 서독의 탄광으로, 베트남전으로 뛰어들어야 했던 덕수. 덕분에 사랑하는 아내 영자(김윤진)를 만날 수 있었고, 토끼 같은 자식들을 대학에 보낼 수 있었지만, 그의 인생에서 ‘그’는 한번도 없었다.

데이터: 펭귄 데이터

이 데이터는 미국 장기 생태 연구 네트워크의 일부인 파머 스테이션 장기 생태 연구 프로그램의 크리스틴 고먼 박사가 2007년부터 2009년 까지 수집한 것입니다. 이 데이터는 환경 데이터 이니셔티브(EDI) 데이터 포털에서 직접 가져온 것으로, 팔머 스테이션 데이터 정책에 따라 CC0 라이선스(“권리 없음”)로 사용할 수 있습니다.”

응답

새로운 영화 시놉시스:

응답

새로운 영화: “펭귄의 꿈”

덕수(황정민)는 부산 국제시장에서 가족을 부양하며 살아가는 평범한 사람이었다. 그러나 그의 가족은 흥남 철수 때 피난민으로 내려온 펭귄들이었다. 덕수는 아버지를 잃은 후, 어머니와 두 동생 펭귄의 생계를 책임지게 되었다.

덕수는 동생 펭귄들의 학비와 생활비를 벌기 위해 자신의 꿈을 포기하고, 친구 달구(오달수)와 함께 미국 장기 생태 연구 네트워크의 일부인 파머 스테이션으로 떠나게 된다. 그곳에서 그는 크리스틴 고먼 박사와 함께 2007년부터 2009년까지

펭귄 데이터를 수집하는 일을 하게 된다.

덕수의 노력 덕분에 사랑하는 아내 영자(김윤진)와 펭귄 동생들을 대학에 보낼 수 있었다. 그러나 그의 인생에서 '그'는 한번도 없었다. 그는 항상 펭귄들을 위해 자신을 희생하며 살아갔다. 그러나 그의 꿈은 언제나 그의 마음속에 살아있었다. 그의 꿈은 펭귄들이 자유롭게 날아다닐 수 있는 세상을 만드는 것이었다. 그의 이야기는 우리에게 희망과 용기, 그리고 사랑에 대한 진정한 의미를 가르쳐준다.

새로운 영화: “펭귄의 꿈”

덕수(황정민)는 부산 국제시장에서 가족을 부양하는 평범한 사람이었다. 그는 전쟁으로 아버지를 잃고, 어머니와 두 동생의 생계를 책임지며, 친구 달구(오달수)와 함께 서독의 탄광과 베트남전으로 뛰어들어야 했다. 그럼에도 불구하고 그는 사랑하는 아내 영자(김윤진)를 만나고, 자식들을 대학에 보낼 수 있었다.

그러던 어느 날, 덕수는 크리스틴 고먼 박사의 펭귄 연구 데이터를 우연히 발견하게 된다. 이 데이터는 2007년부터 2009년까지 미국 장기 생태 연구 네트워크의 일부인 파마 스테이션에서 수집된 것이었다. 덕수는 이 데이터를 통해 펭귄들의 생태와 생활 패턴에 대해 알게 되고, 그들의 삶에 깊은 관심을 갖게 된다.

덕수는 자신의 삶과 펭귄들의 삶을 비교하며, 그들이 겪는 어려움과 도전에 공감하게 된다. 그는 펭귄들의 삶을 이해하고 보호하려는 새로운 꿈을 갖게 되고, 그 꿈을 이루기 위해 자신의 모든 것을 바치기로 결심한다. 이제 덕수의 새로운 도전이 시작되는데…

덕수는 펭귄들의 생태를 연구하며 그들이 겪는 환경적 위협에 대해 알게 된다. 특히, 기후 변화와 인간의 활동이 펭귄들의 서식지를 위협하고 있다는 사실에 충격을 받는다. 그는 이 문제를 해결하기 위해 자신의 모든 것을 바치기로 결심한다. 덕수는 달구와 함께 펭귄 보호 운동을 시작한다. 그들은 펭귄들의 서식지를 보호하고, 인간의 활동이 펭귄들에게 미치는 영향을 줄이기 위한 다양한 활동을 벌인다. 이 과정에서 덕수는 자신의 삶에 새로운 의미와 목표를 찾게 된다.

덕수의 펭귄 보호 운동은 점차 사람들의 주목을 받게 된다. 그의 노력은 많은 사람들에게 영감을 주고, 점차 많은 사람들이 펭귄 보호 운동에 참여하게 된다. 그러나 동시에, 그의 활동은 기업의 이익을 위협하며, 그에게 많은 도전과 위협을 가져다준다.

그러나 덕수는 자신의 꿈을 포기하지 않는다. 그는 펭귄들의 삶을 보호하기 위해 끝까지 싸우기로 결심한다. 그의 노력은 결국 성공하게 되고, 펭귄들의 서식지는

보호되게 된다.

“펭귄의 꿈”은 평범한 사람이 어떻게 자신의 꿈을 이루고, 세상을 바꿀 수 있는지를 보여주는 영화다. 이 영화는 우리에게 꿈을 이루기 위한 도전과 희생, 그리고 그 과정에서 얻는 성취와 만족감에 대해 생각하게 한다.

영화 제목: “펭귄의 시장”

미국 장기 생태 연구 네트워크의 일부인 파머 스테이션에서 크리스틴 고먼 박사(김윤진)는 2007년부터 2009년까지 펭귄에 대한 연구를 수행하였다. 그녀의 연구는 펭귄의 생태계와 그들의 생존 전략에 대한 깊은 이해를 제공하였다.

한편, 부산 국제시장에서 살아가는 덕수(황정민)는 전쟁으로 아버지를 잃고, 어머니와 두 동생의 생계를 책임지게 된다. 그는 동생의 학비와 생활비를 벌기 위해 자신의 꿈을 버리고 친구 달구(오달수)와 함께 미국으로 떠나게 된다. 그의 목적지는 바로 크리스틴 박사가 연구를 수행하는 파머 스테이션이다.

덕수는 파머 스테이션에서 일하면서 크리스틴 박사와 사랑에 빠지게 되고, 그녀의 연구에도 도움을 주게 된다. 그는 펭귄의 생존 전략을 배우며, 그것이 자신의 삶에 어떻게 적용될 수 있는지를 깨닫게 된다. 그의 인생에서 ‘그’는 한번도 없었지만, 펭귄들과의 만남을 통해 그는 새로운 꿈과 희망을 찾게 된다.

“펭귄의 시장”은 생존과 사랑, 그리고 꿈을 찾아가는 이들의 이야기를 그린다.

영화 제목: “펭귄의 꿈”

2007년부터 2009년까지 미국 장기 생태 연구 네트워크의 일부인 파머 스테이션에서 활동한 크리스틴 고먼 박사(김윤진)는 펭귄의 생태계에 대한 깊은 이해를 가지고 있다. 그녀의 연구는 펭귄의 생존 전략과 그들의 환경에 대한 중요한 정보를 제공한다.

한편, 부산 국제시장에서 살아가는 덕수(황정민)는 전쟁으로 아버지를 잃고, 어머니와 두 동생의 생계를 책임지게 된다. 그는 동생들의 학비와 생활비를 벌기 위해 자신의 꿈을 포기하고, 친구 달구(오달수)와 함께 미국으로 떠나게 된다. 그의 목적지는 바로 크리스틴 박사가 연구를 수행하는 파머 스테이션이다.

덕수는 파머 스테이션에서 일하면서 크리스틴 박사와 사랑에 빠지게 되고, 그녀의 연구에도 도움을 주게 된다. 그는 펭귄의 생존 전략을 배우며, 그것이 자신의 삶에 어떻게 적용될 수 있는지를 깨닫게 된다. 그의 인생에서 ‘그’는 한번도 없었지만, 펭귄들과의 만남을 통해 그는 새로운 꿈과 희망을 찾게 된다.

“펭귄의 꿈”은 생존과 사랑, 그리고 꿈을 찾아가는 이들의 이야기를 그린다. 이 영

화는 인간의 삶과 펭귄의 생존 전략 사이의 유사성을 통해 우리 모두가 어떻게 살아가야 하는지에 대한 깊은 메시지를 전달한다.

영화 제목: “데이터의 시장”

덕수(황정민)는 부산 국제시장에서 가족을 부양하며 살아가는 평범한 사람이었다. 그의 가족은 흥남 철수 때 피난민으로 내려온 사람들이었다. 아버지를 잃은 후, 어머니와 두 동생의 생계를 책임지게 된 덕수는 그들의 생활비를 벌기 위해 자신의 꿈을 포기하고, 친구 달구(오달수)와 함께 미국의 데이터 연구소로 떠나게 된다.

데이터 연구소에서 그는 크리스틴 고먼 박사와 함께 2007년부터 2009년까지 펭귄 데이터를 수집하는 일을 하게 된다. 이 과정에서 덕수는 데이터 분석의 중요성과 그것이 어떻게 세상을 바꿀 수 있는지를 깨닫게 된다.

덕수의 노력 덕분에 사랑하는 아내 영자(김윤진)와 동생들을 대학에 보낼 수 있었다. 그러나 그의 인생에서 ‘그’는 한번도 없었다. 그는 항상 가족들을 위해 자신을 희생하며 살아갔다. 그러나 그의 꿈은 언제나 그의 마음속에 살아있었다. 그의 꿈은 데이터를 통해 세상을 바꾸는 것이었다.

“데이터의 시장”은 우리에게 희망과 용기, 그리고 사랑에 대한 진정한 의미를 가르쳐준다. 이 영화는 우리에게 자신을 희생하며 가족을 위해 살아가는 덕수의 이야기를 통해, 가족의 사랑과 헌신, 그리고 데이터의 힘에 대해 생각하게 한다.

영화 제목: “펭귄 데이터의 미스터리”

덕수(황정민)는 부산 국제시장에서 가족을 부양하며 살아가는 평범한 사람이었다. 그의 가족은 흥남 철수 때 피난민으로 내려온 사람들이었다. 아버지를 잃은 후, 어머니와 두 동생의 생계를 책임지게 된 덕수는 그들의 생활비를 벌기 위해 자신의 꿈을 포기하고, 친구 달구(오달수)와 함께 미국의 데이터 연구소로 떠나게 된다.

데이터 연구소에서 그는 크리스틴 고먼 박사와 함께 2007년부터 2009년까지 펭귄 데이터를 수집하는 일을 하게 된다. 이 과정에서 덕수는 펭귄 데이터의 미스터리에 푹 빠지게 되고, 그의 일상은 갑자기 펭귄들의 세상으로 변하게 된다.

덕수의 노력 덕분에 사랑하는 아내 영자(김윤진)와 동생들을 대학에 보낼 수 있었다. 그러나 그의 인생에서 ‘그’는 한번도 없었다. 그는 항상 가족들을 위해 자신을 희생하며 살아갔다. 그러나 그의 꿈은 언제나 그의 마음속에 살아있었다. 그의 꿈은 데이터를 통해 세상을 바꾸는 것이었다.

“펭귄 데이터의 미스터리”는 우리에게 웃음과 재미, 그리고 사랑에 대한 진정한 의미를 가르쳐준다. 이 영화는 우리에게 자신을 희생하며 가족을 위해 살아가는 덕수의 이야기를 통해, 가족의 사랑과 헌신, 그리고 데이터의 힘에 대해 생각하게 한다. 그리고 무엇보다, 펭귄 데이터의 미스터리를 풀어가는 과정에서 발생하는 유머러스한 상황들은 관객들에게 웃음을 선사할 것이다.

챗GPT와 같은 거대 언어 모형(LLM)의 등장은 기존 저작과 창조적 작업에 대한 새로운 관점을 제공하며, 그 과정에서 저작자의 역할도 재평가되고 있다. 일반적으로 사람이 시놉시스나 창작물을 만들기 위해서 상당한 창의적 노력과 시간이 필요했지만 생성형 언어모형 기술에 대한 장벽이 크게 낮아짐에 따라, 빠르고 다양한 방식으로 콘텐츠를 생성할 수 있는 지평이 펼쳐지고 있다.

이러한 변화는 단순히 작업 속도나 효율성을 높이는 것을 넘어, 기존 ‘저작자’ 개념에도 영향을 미치고 있다. 언어 모형을 활용하여 초기 아이디어나 구조를 빠르게 형성한 뒤, 이를 기반으로 사람이 더 깊고 복잡한 내용을 추가하거나 수정한다면, 새로운 형태의 공동 작업이나 협업모델이 될 것이다. 기존 저작자가 단독으로 모든 것을 창조하는 것 이 아니라, 다양한 도구와 협업을 통해 더 풍부하고 다양한 저작물을 만들어낼 수 있는 새로운 길이 열리고 있다.

2.4.2 구조와 외양

챗GPT 디지털 글쓰기에서 “구조와 외양” 단계는 문서 가독성과 효과를 극대화하는 중요한 과정이다. 이 단계에서 다음과 같은 다양한 문서요소와 기능을 활용하여 저작물의 품질을 높힌다.

- 마크다운 기본 글쓰기:** 대중소 제목을 설정하여 문서의 구조를 명확히 한다. 굵은 글씨, 밑줄 등의 텍스트 서식과 단락 구분을 통해 내용을 강조하고 구조화한다.
- 그림 (Figure):** 복잡한 개념이나 데이터를 시각적으로 표현하여 이해를 돋는다.
- 그래프 (Graph):** 데이터 분석 결과를 시각적으로 나타내어 정보의 흐름과 패턴을 쉽게 파악할 수 있게 한다.
- 표 (Table):** 정보를 구조적으로 정리하여 한눈에 볼 수 있게 만든다.
- 도형 (Diagrams):** 프로세스나 시스템의 구조를 시각적으로 설명한다.
- 수식 (LaTeX):** 공학이나 과학 문서에서 복잡한 수학적 표현을 정확하게 나타낸다.
- 호출 블록 (Callout Blocks):** 중요한 정보나 주의사항을 강조하여 독자의 주목을

끈다.

8. **기사 레이아웃 (Article Layout):** 전문적인 기사나 보고서 형식에 맞게 레이아웃을 설정한다.
9. **학술저작 (Scholarly Writing):** 논문이나 학술 문서 작성에 필요한 다양한 서식과 참조 스타일을 적용한다.
10. **다단편집 (Side-by-Side Layout):** 여러 내용을 병렬로 배열하여 비교와 분석을 용이하게 한다.
11. **색상과 시각적 효과:** 다양한 색상과 시각적 효과를 활용하여 문서의 외관을 더욱 풍부하고 독특하게 만든다.

저작물 품질을 높일 수 있는 다양한 문서요소와 기능을 모두 외우고 즉시 활용하는 것은 과거 전문적으로 인쇄출판 종사자들만이 할 수 있었던 작업이지만 챗GPT 자연어 처리 능력을 활용하여 쉽고 빠르게 저작물에 적용할 수 있는 시대가 열렸다.



그림 2.2: 문서 구조와 문학적 프로그래밍 (쿼토)

표, 그래프는 문서 구조와 문학적 프로그래밍의 중요성을 시각적으로 전달하는 중요한 매체다. R 언어 tidyverse, palmerpenguins, gt 패키지를 활용해 펭귄 데이터를 처리하고, 펭귄 종별 암수 개체수를 표로 정리하고 '펭귄종별 암수 개체수'라는 제목과 함께 데이터 출처를 표기하는 각주까지 표에 꼼꼼히 표기했다. 그래프 예시는 R 언어 ggplot2 패키지를 사용해 펭귄 물갈퀴 길이와 체질량 사이의 관계를 그래프로 표현하고 있다. 향후 표와 그래프는 영화 시놉시스 저작물의 중요한 문서 구성요소 중 하나로 활용될 것

표

그

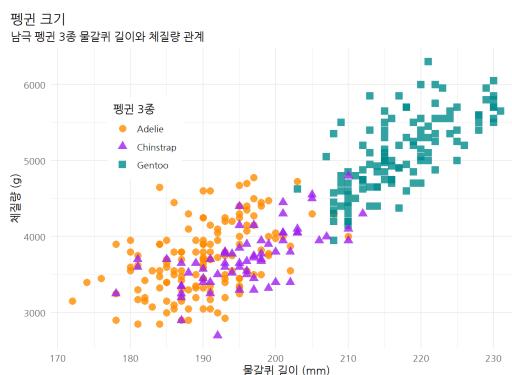
래

프

펭귄종별 암수 개체수¹

펭귄종	암컷	수컷
Adelie	73	73
Chinstrap	34	34
Gentoo	58	61

¹ 데이터 출처: palmerpenguins 패키지



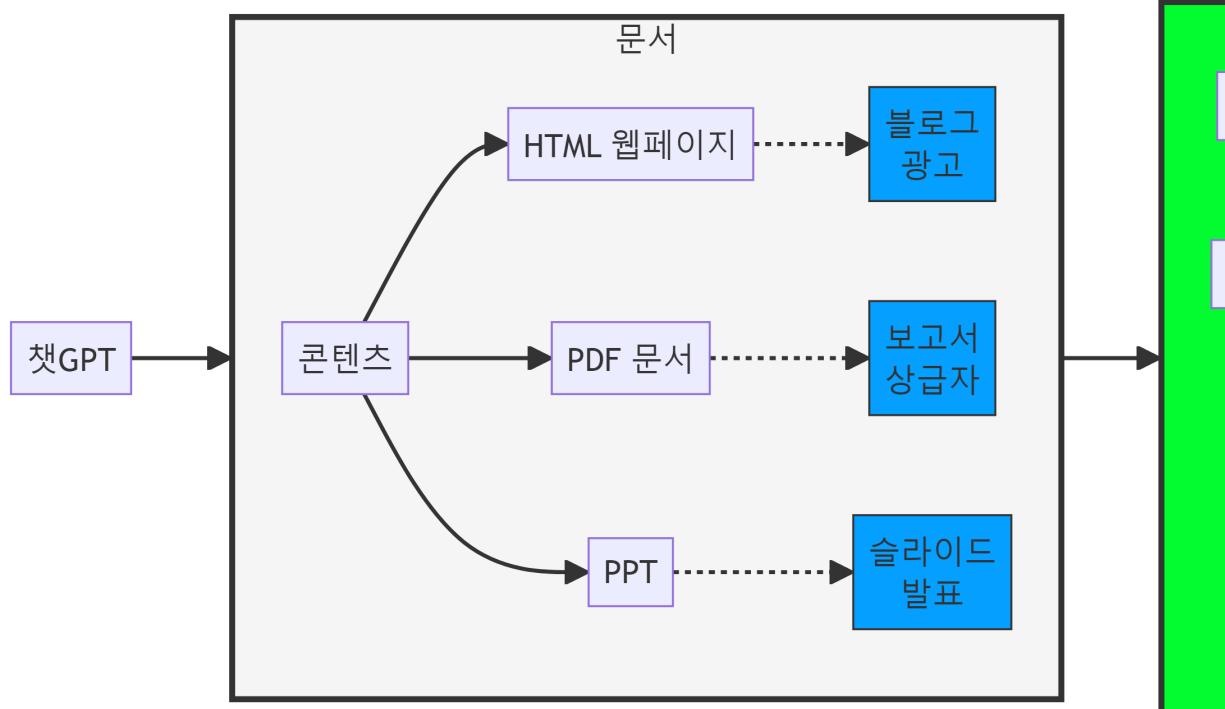
이다.

2.4.3 배포와 공유

디지털 글쓰기 저작물을 완성한 후 다음 단계로 인쇄출판과 배포를 고려해야 한다. 과거와 달리 현재는 물론 앞으로 다양한 문서 저작물을 하나의 프로젝트로 엮어 전자출판을 할 수 있는 플랫폼이 필요하다. Quarto Pub은 이러한 목적으로 개발되어 저자가 작성한 다양한 문서저작물을 원클릭으로 전자출판할 수 있다. 물론 Quarto Pub 외에도 netlify, GitHub Pages, RStudio Connect 등 다른 출판배포 플랫폼도 존재한다. 플랫폼들은 각각의 특성과 장점을 가지고 있어, 저자의 필요와 목적에 따라 적절한 플랫폼을 선택하여 디지털 글쓰기 저작물을 더 넓은 독자에게 전달할 수 있다.

최종 출판저작물 인쇄출판과 배포를 마친 후에는 저작물 공유도 고려할 수 있다. 저작물을 공유함으로써 저자는 자신의 저작물을 다른 사람들과 공유하고, 다른 사람들은 저작물을 활용할 수 있다. 저작물을 공유하는 방법은 다양하지만, 가장 쉬운 방법은 저작물을 공유할 수 있는 플랫폼에 저작물을 공유 저작권 표기를 함께 게시하는 것이다. 소스코드를 공유하면 다른 사람들이 문서 작성에 사용된 소스코드를 쉽게 이해하고, 문

서 작성에 사용된 소스코드를 활용해 자신만의 향상된 저작물을 공유함으로써 글쓰기 문화를 확산시킬 수 있다.



챗GPT 디지털 글쓰기를 통해 저작한 문서는 다양한 형태로 배포될 수 있다. “국제시장과 남극 펭귄”을 모티브로 삼아 제작한 다양한 영화 시놉시스 저작물은 다음과 같이 공개되어 있다.

- **HTML 웹페이지**: HTML 웹페이지 형식으로는 누구나 쉽게 접근하고 읽을 수 있으며, 웹 브라우저만 있으면 어디서든 열람이 가능하다.
- **PDF 파일**: PDF 파일 형식으로는 공식 문서나 보고서로 사용할 수 있으며, 물리적인 출력이 필요할 때도 유용하다.
- **PPT 슬라이드**: PPT 슬라이드 형식으로는 발표나 강의 자료로 활용할 수 있어, 다양한 상황에서 유연하게 사용할 수 있다.

문서 저작을 넘어 다양한 형식으로 문서를 배포하면, 독자나 이용자가 자신에게 가장 편리한 방식으로 문서를 열람할 수 있다. 따라서 챗GPT 디지털 글쓰기는 단순히 텍스트를 생성하는 것을 넘어, 다양한 배포 채널과 형식을 통해 더 넓은 독자층에게 접근할

수 있는 장점도 크다.

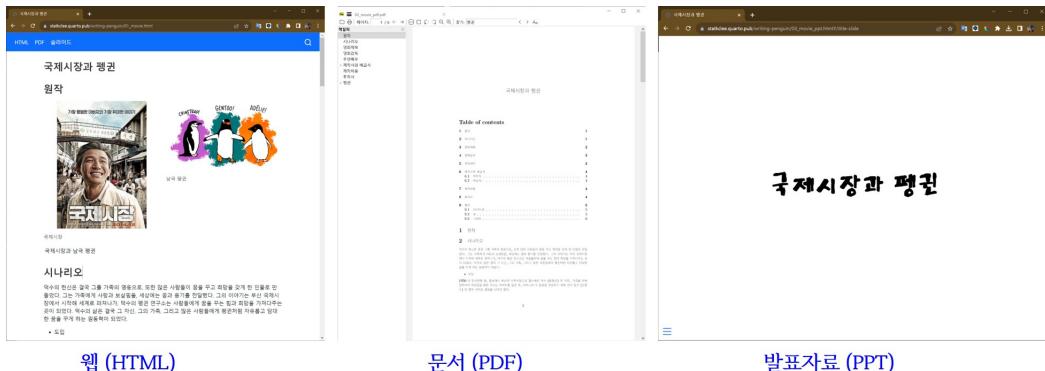


그림 2.3: 웹, 문서, PPT 슬라이드 배포

제 I 편

글쓰기 환경

제 3 장

언어와 문자

글쓰기 역사는 인류 문명과 밀접하게 연결되어 있다. 언어의 발전과 말하기가 수만 년 동안 이루어진 후에, 비로소 문자가 등장했다. 초기 문자 증거는 약 3000 BC경에 이집트 와 메소포타미아에서 발견되었고, 중국에서도 거의 동시기에 독립적으로 발명되었다. 초창기 문자는 주로 상거래나 종교적 행사, 법률 등을 기록하기 위해 사용되었다.

특히 주목할 만한 점은 문자가 등장하기 전에 숫자가 먼저 다양한 형태로 사용되기 시작했다는 점이다. 초기 인류의 생존과 밀접한 관련이 있는 농업분야에서 경작지 면적, 수확량 계산, 별자리나 계절의 변화를 기록하여 더 많은 곡식을 얻기 위한 필요성 때문으로 보인다.

문자는 시간이 흐르면서 단순한 정보의 기록에서 벗어나, 다양한 목적과 사회적 문제 해결을 위한 중요한 도구로 진화해 왔다. 예를 들어, 고대 로마나 그리스에서는 법률과 철학, 역사 등을 글로 기록하였고, 이는 후대에 큰 영향을 미쳤다. 또한, 글쓰기는 문화와 예술, 심지어는 정치와 사회 변혁에도 중요한 역할을 하게 되었다.

3.1 문자 구성요소

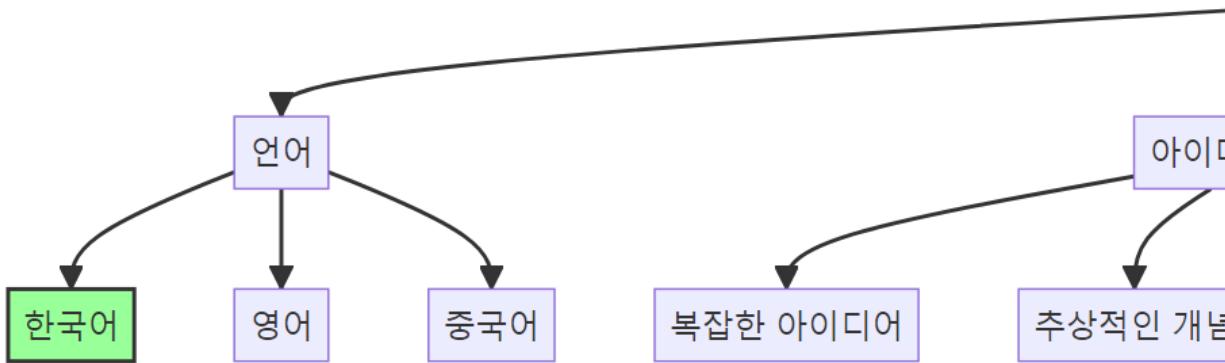
의사소통과 정보 표현과 전달에 있어 중요한 차이를 가지는 언어(Language)와 문자 (Writing System)를 구분하는 것은 중요하다. 언어는 음성, 제스처, 기호 등 다양한 형태로 표현되며, 문법, 어휘, 발음 등을 통해 의미를 전달한다. 이와 달리 문자는 언어를 시각적으로 표현하기 위한 기호체계로, 알파벳, 음절 문자, 로고그램 등 다양한 형태를 가

진다. 언어는 문자 없이도 존재할 수 있으나, 문자는 특정 언어를 표현하고 기록하기 위해 존재한다.

문자가 존재하고 효과적으로 기능하기 위해서는 몇 가지 기본 조건이 필요하다. 언어, 생각, 기호, 매체가 문자가 존재하기 위해 필요한 조건이다.

- 언어: 문자는 특정 언어(한국어, 영어, 중국어 등)의 표현 수단이다. 언어의 복잡성과 다양성을 효과적으로 표현할 수 있는 체계적 문자 체계가 필요하다.
- 생각: 문자는 복잡한 아이디어와 추상적인 개념, 감정 또는 전문 용어를 표현하는데 사용된다. 문자는 아이디어를 구체화하고 전달하는데 필수적인 수단이다.
- 기호: 문자 외에도 숫자, 수식, 교통표지, 음표, 화학식, 프로그래밍 구문 등 다양한 기호와 표식을 사용하여 특정 정보나 의미를 전달한다.
 - 언어 기호: 알파벳, 한글, 한자 등
 - 숫자와 수식: 수학적 표현, 공식 등
 - 교통표지 및 표식: 안전 지시, 경고 표시 등
- 매체: 글쓰기를 기록하고 전송하는 물리적 또는 디지털 매체가 필요하다. 종이와 파피루스와 같은 전통적인 매체에서 현재는 컴퓨터 디스플레이, SSD, 클라우드 디지털 저장소가 그 역할을 대신하고 있다.

언어와 문자를 모두 다루는 것은 지면관계상 불가능하여 한국어를 중심으로 문자에 대해 다루고, 아이디어와 개념에서는 전문 용어를 중심으로 한 과학기술 문서 저작, 기호와 표식에서는 한글을 중심으로 한 언어적 기호와 숫자와 수학적 기호, 마지막으로 매체에서는 PDF 종이출판과 웹 디지털출판을 중심으로 살펴본다.



3.1.1 언어와 문자

위키백과 따르면 전세계 인구는 약 80억명(2030년 기준)에 이르며, 화자들에게 본토어로 성경을 제공할 목적으로 기독교 언어학 봉사단체 에스놀로그 통계에 따르면 7,168개(2023년 기준) 언어가 사용되고 있다. 이 중에서도 영어, 중국어, 힌디어, 스페인어, 프랑스어, 아랍어, 벵골어, 러시아어, 포르투갈어, 인도네시아어가 가장 많은 사용자를 보유하고 있다.¹

순위	언어	구분		
		모국어	제2언어	전체
1	English	380,000,000	1,077,000,000	1,456,000,000
2	Mandarin Chinese	939,000,000	199,000,000	1,138,000,000
3	Hindi	345,000,000	266,000,000	609,000,000
4	Spanish	485,000,000	74,000,000	559,000,000
5	French	81,000,000	229,000,000	310,000,000
6	Modern Standard Arabic	0	274,000,000	274,000,000

¹ 자료출처: https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

7	Bengali	234,000,000	39,000,000	273,000,000
8	Portuguese	236,000,000	27,000,000	264,000,000
9	Russian	147,000,000	108,000,000	255,000,000
10	Urdu	71,000,000	161,000,000	232,000,000
13	Japanese	123,000,000	200,000	123,000,000
21	Vietnamese	85,000,000	1,000,000	86,000,000
24	Korean	82,000,000	29,000,000,000	82,000,000
29	Italian	65,000,000	3,000,000	68,000,000

전세계에는 7,000개가 넘는 언어가 존재하나, 이를 표현하는 문자 체계는 대략 50개 정도에 불과하다. 한국어(제주어 포함)는 한글을 문자 체계로 사용하고 있지만, 이처럼 모든 언어가 고유의 문자(**Writing System**)를 가진 것은 아니다. 실제로 여러 언어가 동일한 문자 체계를 공유하는 경우가 많다. 예를 들어, 알파벳은 라틴어를 기반으로 하며, 영어, 불어, 독어, 이태리어 등 다양한 언어에서 사용되고 있다. 이 외에도 셀틱어, 발틱어, 슬로박어 등도 알파벳이 활용된다. 한자는 중국어와 대만어에서 주로 사용되며, 아랍문자는 아랍어뿐만 아니라 인도-이란 어족(페르시아어, 편잡어 등)에도 쓰인다. 일본어는 가나(□名)라는 문자 체계가 사용된다.

스크립트_명칭	문자유형	인구수
Latin Latin	Alphabet	4,900,000,000
Chinese □字 漢字	Logographic	1,541,000,000
Arabic 𠁻𠁻𠁻𠁻𠁻𠁻	Abjad or Abugida (when diacritics are used)	828,000,000
Devanagari 𠁻𠁻𠁻𠁻𠁻𠁻𠁻	Abugida	480,500,000
Cyrillic Кирилица	Alphabet	289,000,000
Bengali-Assamese 𠁻𠁻𠁻𠁻-𠁻𠁻𠁻𠁻𠁻	Abugida	234,000,000
Kana かな 力ナ	Syllabary	123,000,000
Telugu 𠁻𠁻𠁻𠁻𠁻	Abugida	83,000,000
Hangul 한글 조선글	Alphabet, featural	81,700,000
Tamil 𠁻𠁻𠁻𠁻	Abugida	78,600,000

3.1.2 매체의 역사

글쓰기와 문자의 발전은 인류 역사와 밀접하게 연결되어 있다. 초기에는 상형문자와 같은 시각적 기호를 사용하여 의미를 전달했으며 기원전 16,500년에 라스코 동굴벽화와 반구암각화 등에서 볼 수 있듯이 동굴벽면이나 돌 등을 매체로 사용했다.

수메르 문명의 출현과 함께 사람들이 집단으로 생활하기 시작하면서 도시화가 급속히 진행되었다. 이로 인해 세금 징수, 계약, 무역 등 복잡한 사회적, 정치적, 경제적 문제가 발생했고, 이를 해결하기 위해 더욱 정교한 문자 체계가 필요하게 되었다. 수메르에서는 다양한 정보를 기록하고 보존할 수 있도록 점토를 매체로 사용였다.

이집트 문명도 유사한 문제에 봉착했으며 매체로 점토 대신 파피루스를 사용했다. 파피루스는 더 가볍고 휴대하기 쉬워 글쓰기와 정보 보존 및 전달이 더욱 효율적으로 이루어졌다.



그림 3.1: 문자 매체의 진화

문자를 담을 수 있는 매체로 초기에는 동굴벽이나 돌, 점토 등을 사용했지만, 이러한 매체는 비용이 많이 들고 유지관리가 어려웠다. 또한 상형문자가 담을 수 있는 정보에 한계가 있어, 더 효율적인 방법이 필요했다.

먼저 서양에서는 페니키아인들이 이러한 문제를 해결하기 위해 기원전 1,200년에 페니키아 문자를 창안했다. 이 문자는 소리에 의미를 담을 수 있어, 상형문제에 비해 무한대에 가까운 의미를 표현할 수 있고 같은 지중해 문명권 그리스와 로마 문화를 거쳐 라틴 알파벳이 공표되었고, 이를 담을 수 있는 매체로 양피지가 채택되었다.

한편 중국에서는 106년에는 채륜이 종이를 발명함으로써 글쓰기 문화에 큰 변화가 일어났다. 서기 800년경 목판인쇄술이, 1450년 구텐베르크 인쇄술이 발명되어 출판의 대량 생산이 가능해졌다. 1806년에는 푸어드니어 형제가 종이를 대량으로 생산할 수 있는 초지기를 발명, 저렴한 비용으로 일반 대중도 생각을 문자로 기록하고 널리 자신의 생각을 알릴 수 있게 되었다. 1922년 ISO 216을 통해 종이에 대한 국제 표준이 확립되어 미국을 제외한 대부분의 국가에서는 문자를 담을 수 있는 종이 매체에 대한 표준이 확립되었다.

3.1.3 문자

언어를 기록하기 위한 상징체계로 문자(文字, writing system)가 발명되었다. 문자는 언어 중에서도 음성언어를 기록하기 위해 생겨난 것으로 알려져 있다. 문자의 중요성은 정보의 저장과 전달에 있어 신뢰성이 담보되기 때문에 크다. 문자는 크게 두 가지 유형으로 분류된다.

- 표의문자(Logographic):** 그림이나 상징을 사용하여 단어나 개념을 직접 나타낸다. 중국 한문과 이집트 상형문자가 대표적으로, '木'은 '목'이라는 소리를 내지만, 그 뜻은 '나무'다.
- 표음문자(Phonographic):** 발음이나 소리를 나타내기 위해 설계되었고, 한글, 라틴 알파벳 등 대부분의 언어가 이 유형에 속한다.

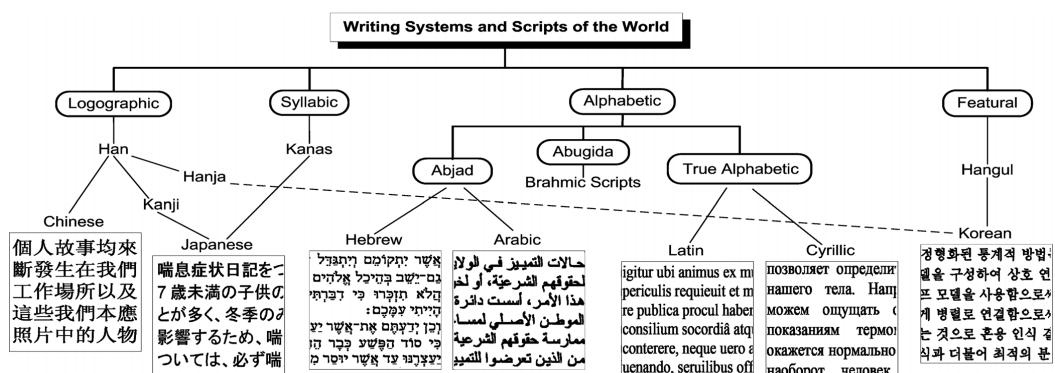


그림 3.2: 문자와 스크립트

표의문자는 상징이나 그림을 통해 의미를 전달하므로, 하나의 문자가 복잡한 개념이나 단어를 나타낼 수 있는 반면에 표음문자는 소리를 기반으로 하므로, 단순하고 규칙적인 구조를 가지며, 다양한 단어와 문장을 구성하기에 유용하다. 표의문자는 의미를 전

달하기 위해 그림이나 상징을 사용하므로, 그림이나 상징을 알지 못하면 의미를 전달할 수 없기 때문에 문자 수가 많아지고, 학습에 많은 시간이 필요하다. 반면에 표음문자는 소리를 기반으로 하므로, 문자를 조합하여 의미를 만들기 때문에 문자 수가 적고, 학습에 상대적으로 적은 시간이 필요하다.

문자체계	문자 수	문자 구분	설명
라틴 알파벳	26	알파벳	영어와 다른 라틴계 언어 사용
한글	14 자음, 10 모음	자음, 모음	한국어 사용
중국어 한자	50,000+	한자	중국어 사용되며, 일부는 일본어, 한국어 사용
아랍 알파벳	28	알파벳	아랍어와 다른 인도-이란 어족 언어 사용
히브리 알파벳	22	알파벳	히브리어 사용
키릴 알파벳	33	알파벳	러시아어와 다른 슬라브 언어 사용
데바나가리	47~52	알파벳	힌디어와 다른 인도 언어 사용
그리스 알파벳	24	알파벳	그리스어 사용
일본어 가나	46 가타카나, 46 히라가나	가타카나, 히라가나	일본어 사용

i 문자와 스크립트

문자는 ‘무엇(what)’을 기록할 것인지에 대한 체계나 원칙이며, 스크립트는 ‘어떻게(how)’ 그것을 기록할 것인지에 대한 구체적인 방법으로 나눠볼 수 있다.

- 문자(Writing System): 언어를 기록하기 위한 전반적인 방법론이나 체계를 의미하며, 언어의 특정 부분(예: 음성, 의미, 문법 등)을 어떻게 표현할 것인지에 대한 규칙과 원칙이 포함된다. 문자는 언어의 복잡성과 뉘앙스를 어떻게 효과적으로 포착할 것인지에 대한 방법을 제공한다. 예를 들어, 알파벳, 로고그램, 음절 문자 등을 들 수 있다.
- 스크립트(Scripts): 특정 글쓰기 시스템을 구현하는 데 사용되는 문자 집합을 의미한다. 스크립트는 문자의 ‘글꼴(폰트)’이나 ‘스타일’과 같은 것으로 생각할 수 있다. 예를 들어, 라틴 스크립트, 한글, 한자 등을 들 수 있다.

앞서 문자를 크게 표음문자와 표의문자로 구분하였다. 이를 좀 더 상세히 세분화하면 전세계 문자는 크게 다음과 같이 4개로 구분된다. 첫 번째는 표어문자(Logographic)로, 하나의 문자는 하나의 단어나 개념을 나타낸다. 대표적으로 중국의 한자가 있으며, 예를 들어 ‘木’ 문자는 ‘나무’라는 개념을 나타낸다. 두 번째는 음절문자(Syllabic)로, 하나의 문자가 하나의 음절을 나타낸다. 일본어의 가나(ひらがな, カタカナ)가 이에 해당하며, ’か’ 문자는 ‘ka’라는 음절을 나타낸다. 세 번째는 음소문자(Alphabetic)로, 하나의 문자가 하나의 음소(최소 음성 단위)를 나타낸다. 로마자로 대표되는 영문자가 이에 해당하며, ’A’ 문자는 ‘에이’라는 음소를 나타낸다. 마지막으로 자질문자(Featural)로 문자의 형태가 그 문자가 나타내는 발음이나 음성 자질과 직접적으로 관련이 있다. 한글이 대표적인 예로, ’ㄱ’ 문자는 뒷니와 혀가 만나는 위치에서 나는 소리를 나타낸다. [gohos2010]

종류 기호 표현 예		
표어 문자	형태소	중국 한자
음절 문자	음절 또는 모라	일본 가나 (문자)
음소 문자	낱소리 (자음 또는 모음)	로마자, 키릴문자, 그리스 문자
아부기다	낱소리 (자음+모음)	데바나가리 문자, 그으즈 문자
아브자드	낱소리 (자음)	아랍 문자, 히브리 문자
자질 문자	음성 자질	한국 한글

3.1.3.1 알파벳

알파벳(Alphabet) 역사는 매우 흥미롭다. 이집트 상형문자에서 시작하여 페니키아 상인을 거쳐 그리스와 로마로 이어지는 과정은 언어와 문자의 진화를 잘 보여준다. 특히, 페니키아 상인이 소리를 기반으로 의미를 전달하는 방식을 채택한 것은 문자의 역사에서 중요한 전환점이다. 이전에는 이집트 상형문자와 같은 표의문자가 주로 사용되었는데, 이러한 문자는 그림이나 상징을 통해 의미를 전달하는 방식은 복잡하고 비효율적이었다. 특히 상업적인 거래나 다양한 문화와 언어 간의 소통에서는 이러한 복잡성이 큰 장애였다.

페니키아 상인들은 문제를 해결하기 위해 소리를 기반으로 한 새로운 문자 체계를 도입했다. 각 문자의 특정 소리나 발음을 조합하여 단순하고 효율적인 의사소통이 가능하게 된다. 표음문자 도입은 상업적인 거래를 더욱 원활하게 하고, 다양한 언어와 문화 간의 소통을 쉽게 만들었으며 알파벳 문자 체계로 발전할 수 있는 기반을 마련했다. 그리

스에서 알파벳 뒤쪽 문자가 추가되고, 로마시대에 비로소 오늘날 알려진 영어 대문자가 완성되었다. 이후 소문자는 손으로 필사하는 과정에서 발명되었고, 다양한 글꼴도 생겨 났다. 구텐베르그 활자인쇄술 발명은 현재와 같은 알파벳 문자 완성을 가능하게 했다.²

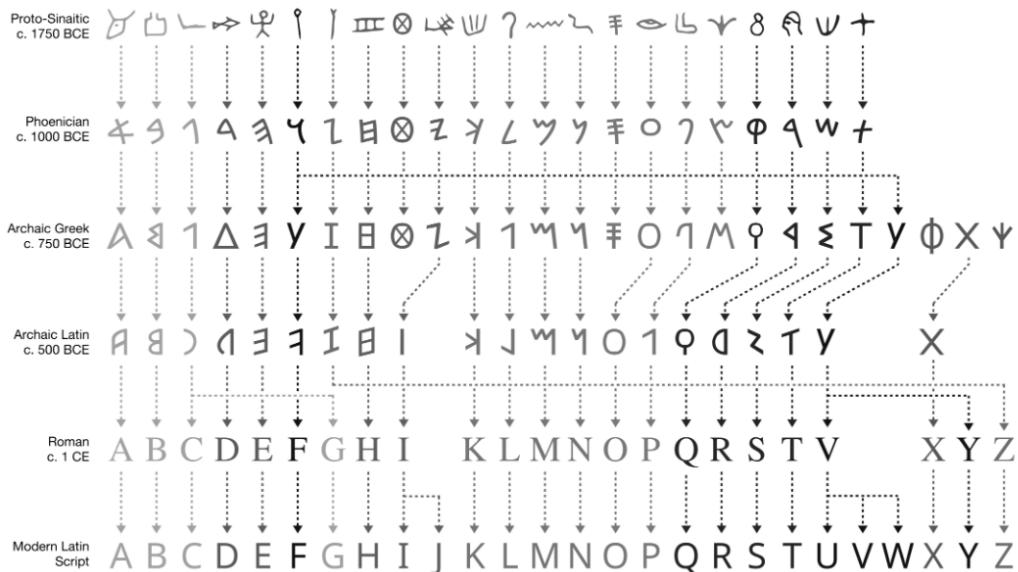


그림 3.3: 알파벳 문자 진화 역사

```
#> 알파벳 대문자: 26 개
#> A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
#> 알파벳 소문자: 26 개
#> a b c d e f g h i j k l m n o p q r s t u v w x y z
```

3.1.3.2 표의 문자

한자와 같은 표의 문자가 여러가지 단점이 있는 것처럼 보이기도 하지만 표의 문자가 갖는 장점으로 인해 그 중요성이 더욱 부각되고 있다. 표의 문자가 갖는 장점은 문자 자체 의미가 부여되기 때문에 전세계 누구도 문자만 보면 그 뜻을 유추할 수 있다는 점이다.

표의 문자의 장점은 의미 자체가 문자에 내재되어 있기 때문에, 언어의 장벽을 뛰어넘을 수 있다. 예를 들어, 한자로 쓰인 '木'는 '나무'라는 의미를 가지고 있어 언어에 상관 없이 일관된다. 따라서 중국어를 모르는 사람도 이 문자를 보고 '나무'라는 개념을 이

²[Evolution of the Alphabet | Earliest Forms to Modern Latin Script [유튜브](#)]

해할 수 있다. 이러한 특성은 다양한 언어와 문화가 교차하는 글로벌한 환경에서 매우 유용하다. 표의문자는 복잡한 개념이나 아이디어를 간결하고 명확하게 전달할 수 있고 시각적인 요소가 강하기 때문에 디자인이나 예술 작업에도 자주 사용된다. 하지만, 의미가 복잡하거나 추상적인 경우, 하나의 문자로 그 의미를 완전히 표현하기 어렵고 표의문자를 익히고 제대로 사용하기 위해서 상대적으로 높은 학습 비용은 단점으로 꼽을 수 있다.

- 그림문자(Pictographic): □, ?, □, …
- 표의문자(Ideographic): □ 금지와 같은 의미를 갖는 문자로 □ □ 교통 표지판, ► 전자제품 사용
- 추상 어표(Abstract Logographic): 한자 사람 인(人), %, &, …

그림문자와 표의문자를 결합하면 매우 효과적인 의사소통 도구가 될 수 있다. 예를 들어, “금연”이라는 표의문자와 금지를 나타내는 원형의 빨간색 선을 그림으로 표현한 그림문자를 함께 사용하면, 언어의 장벽 없이 “흡연 금지”라는 메시지를 명확하게 전달 할 수 있어, 공공장소, 국제공항, 다양한 문화 및 언어가 혼재된 환경에서 특히 유용하다. 단순한 텍스트나 단일한 그림만 사용하는 것보다 사람들이 다양한 언어와 문화 배경을 가지고 있을 때, 그림문자와 표의문자의 조합은 메시지를 빠르고 명확하게 전달 하는 데 큰 도움을 준다.



그림 3.4: 흡연과 금지가 결합된 표의문자

3.1.3.3 자질문자 (한글)

자질문자(featural alphabet)는 문자가 그 문자가 표현하는 소리의 특징(자질)을 시각적으로 나타내는 문자 체계를 의미한다. 즉, 문자의 형태가 그 문자가 나타내는 발음이나 음성 자질과 직접적으로 관련있는 체계로 문자를 배우고 이해하기 쉽다.

자질문자의 대표적인 예로 한글이 꼽힌다. 한글 자음 문자는 발음하는 입모양을 나타내려고 설계되었고, 모음 문자는 혀의 위치와 움직임을 나타낸다. 자질문자는 발음의 물리적, 생리적 특성을 문자에 반영하기 때문에, 언어의 발음 체계에 변화가 생겨도 그에 따라 쉽게 수정이 가능하다는 장점이 있고, 이러한 설계 덕분에 한글은 상대적으로 배우기 쉽고 효율적인 문자 체계로 평가된다.

한글은 세종대왕에 의해 창제되어 한국어를 표기하기 위한 고유문자다. 표음문자와 자질문자의 특성을 모두 가지고 있어, 한자와 달리 음소를 정확하게 표현할 수 있다. 창제 초기에는 훌소리(모음) 11개와 닿소리(자음) 17개, 총 28개의 문자로 구성되었으나, 현재는 훌소리 10개와 닿소리 14개, 총 24개 기본 문자만 사용한다.

한글은 그 구조적 우수성과 표기의 정확성 때문에 지속적인 연구와 보급이 이루어졌고, 음소 단위 표기가 가능하여 매우 높은 표기 정확성을 자랑한다. 20세기에 들어, 다양한 학문 분야에서 한글 구조와 기능에 대한 연구가 활발히 이루어져, 한글만의 고유한 문법과 표기법이 정립되어, 현재 한국어 뿐만 아니라 제주어, 찌아찌아어 등 다양한 언어와 방언의 표기에도 널리 사용되고 있다.

- #> 한글 자음: 14 개
ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
- #> 한글 모음: 10 개
ㅏ ㅑ ㅓ ㅕ ㅗ ㅕ ㅜ ㅕ ㅡ ㅣ

3.2 기호의 역사

문자는 사람의 필요와 유행에 항상 변화하고 발전했다. 초기에는 토큰(token)이라고 불리는 간단한 기호나 물체를 사용하여 정보를 저장하고 전달하는 목적으로 주로 물품 거래나 세금 징수 등을 기록하기 위해 사용되었다. 그 다음 단계로 그림문자(pictogram)가 등장했는데 상징적인 그림을 통해 단순한 정보나 개념을 표현하는 방식이었다. 로고그래피(logography)는 그림문자를 발전시킨 형태로, 특정 단어나 개념을 나타내는 문자를 사용한다는 점에서 시각중심에서 청각중심으로 전환을 의미했다. 알파벳은 로고그래피에서 더욱 발전하여, 언어소리를 분할하여 나타내는 문자로 언어의 복잡성을 더욱 정교하게 표현할 수 있도록 했다. 한글은 조선시대에 이르러 가장 늦게 발명되었는데, 기존 문자를 자체적으로 분석하고 이해하여 전혀 새롭고 효율적인 문자를 만들어 낸 결과물이다.

- 상형문자(기원전 6000~4000년경): 초기 문자 형태인 퍽토그래프(Pictograph)는 사물이나 아이디어를 나타내는 단순한 그림이나 기호였다. 수메르, 이집트, 중국에서 기본적인 정보를 전달하기 위해 사용되었다.
- 설형 문자(기원전 3400~3200년경): 고대 수메르(현대 이라크)에서 개발된 설형문자는 초기 문자 체계 중 하나로 간주된다. 상형문자에서 발전한 문자로 점토판에 짜기 모양의 스타일러스를 놀려 단어와 아이디어를 나타내는 일련의 짜기 모양의 자국을 만들어 표현했다.
- 이집트 상형문자(기원전 3200~3000년경): 고대 이집트인들은 음성 기호와 표의 문자를 조합하여 사용하는 이집트 상형문자를 개발했다. 상형문자는 종교 텍스트, 비문, 공식 문서에 사용되었다.
- 페니키아 알파벳(기원전 1200년경): 페니키아 알파벳은 이집트 상형문자에서 파생된 초기 알파벳 문자다. 각 기호가 고유한 소리를 나타내는 알파벳 개념을 도입하여 문자 발전에 있어 중요한 발전을 이루어냈다.
- 그리스 알파벳(기원전 800년경): 그리스 알파벳은 페니키아 알파벳에서 발전하여 모음을 도입하여 문자 표현력을 크게 향상시켰으며, 그리스 알파벳은 라틴어와 키릴 문자를 비롯한 많은 현대 유럽 문자 체계의 토대를 마련했다.
- 라틴 알파벳(기원전 700년경): 고대 로마인들이 사용한 라틴 알파벳은 그리스 알파벳에서 파생되었다. 시간이 지남에 따라 발전하여 영어, 스페인어, 프랑스어, 독일어 등 많은 현대 문자의 기초를 형성하고 있다.
- 중국어 문자(기원전 1200년경): 중국어 문자(한자)는 상형문자에서 발전하여 각 문자가 단어 또는 형태소(의미 단위)를 나타내는 복잡한 로고그램으로 발전했다. 한자는 일본어와 한국어 등 다른 동아시아 문자 체계의 발전에 영향을 미쳤다.
- 브라흐미 문자(기원전 300년경): 브라흐미 문자는 고대 인도에서 개발되었으며 데바나가리(힌디어와 산스크리트어에 사용), 타밀어, 태국어를 비롯한 많은 남아시아 및 동남아시아 문자의 조상으로 간주된다.
- 아랍 문자(기원전 4세기경): 나바테아 알파벳에서 발전한 아랍 문자는 아랍어를 표기하는 데 사용되며 페르시아어, 우르두어, 파슈토어 등 다른 언어와 함께 사용할 수 있도록 변형되었다.
- 키릴 문자(기원전 9세기경): 러시아어와 다른 여러 슬라브어를 표기하는 데 사용되는 문자 체계이다.

되는 키릴 문자는 선교사 형제인 시릴과 메토디우스가 제1차 불가리아 제국에서 개발했다. 그리스 알파벳을 기반으로 슬라브어 고유의 소리를 표현하기 위한 문자가 추가되었다.

- **한글(기원후 14세기):** 조선전기 제4대 세종대왕이 훈민정음이라는 이름으로 창제 하여 반포한 우리나라 고유의 문자다. 한글은 다른 나라의 기원과 발달 과정에 비하면 전혀 다른 과정을 거쳐 온 문자로, 세종대왕을 중심으로 한 소수의 집현전 학자들에 의해 치밀하게 계획되어 처음부터 완전한 글자꼴과 표현 원리를 정해 1443년에 창제되어 세종28년 1446년에 10월에 반포한 것으로 이러한 독창적인 글자를 만든 일은 세계 역사에 일찍이 찾아볼 수 없다.

제 4 장

편집기와 출판

디지털 전환(Digital Transformation) 전 글쓰기는 착상 후 종이와 연필을 가지고 글을 작성하고 수차례 수정작업을 거쳐 타자기로 탈고를 하고 이를 출판사에 넘겨 책이나 보고서로 고객에게 전달하게 된다. 하지만, 디지털 전환(Digital Transformation) 시대 도래로 글쓰기 방식도 크게 변화하게 된다. 아래한글이나 마이크로소프트 워드, 리브레 Write와 같은 워드 프로세서(Word Processor)를 넘어, Vim, Emacs, VS 코드와 같은 코딩 편집기에서 마크다운을 사용하거나 직접 HTML을 코딩하여 글쓰기를 하게 된다.

중간 단계로 **LATEX**을 사용하여 수식을 포함한 학위 논문이나 학술 저널, 컨퍼런스 발표를 위한 논문을 작성하기도 한다. 이를 위해 TeXStudio와 같은 **LATEX**에 특화된 통합개발환경(IDE)를 사용하여 생산성을 높이기도 한다. RStudio의 Visual 마크다운 편집기가 RStudio v1.4에 도입되면서, 문서 작성 방식에 혁명적 변화가 일어나고 있다. 이러한 변화를 통해 글쓰기의 생산성과 효율성이 대폭 향상되며, 다양한 포맷과 플랫폼에서의 유연한 작업이 가능해진다.

i 통합개발환경

통합개발환경(IDE, Integrated Development Environment)은 소프트웨어 개발을 위해 특별히 설계된 어플리케이션 또는 소프트웨어다. 코드 편집, 디버깅, 컴파일, 실행 등 소프트웨어 개발 전반을 지원한다. IDE는 프로그래밍 언어와 플랫폼에 따라 특화된 도구가 존재하고, 대표적으로 VS 코드, Eclipse, IntelliJ, Xcode, RStudio, 파



그림 4.1: 디지털 글쓰기 통합개발환경(IDE) 편집기

이참 등이 있다.

IDE는 개발자 생산성을 높이고, 작업 효율성을 높이기 위해 자동 완성, 문법 검사, 코드 강조(Highlighting) 등 기능을 제공할 뿐만 아니라, 플러그인 형태로 버전 관리 시스템과 통합, 디버깅 툴, 테스트 자동화 도구 등도 포함되어 있다.

글쓰기 IDE 도구가 필요한 이유는 단순한 텍스트 작성은 넘어서 복잡하고 기능적인 문서제작을 위해서 프로그래밍 언어와 플러그인 지원이 필수적이다. 이렇게 작성된 문서는 단순한 텍스트 제한을 벗어나 더 다양한 정보와 기능을 담게된다. 자동 완성, 문법 검사, 실시간 미리보기 등의 기능을 제공하여 문서작성 속도와 효율성이 비약적으로 높아져 생산성이 크게 향상되는데, 특히 복잡한 문서나 코드가 많이 필요한 문서를 작성할 때 시간을 대폭 줄여준다.

Git과 같은 버전 관리 시스템과 통합이 가능하므로 국내외 협업을 통해 동시에 문서 작성과 수정을 할 수 있고, 이전 버전과의 비교, 병합, 충돌 해결 등 문서 및 코드 공동 작성에서 발생하는 문제를 수월하게 해결할 수 있다. 소스코드 문서를 웹사이트(HTML), PDF, 워드 문서, 발표자료(PPT), 대쉬보드 등 다양한 파일형식으로 쉽게 변환할 수 있어 해 문서 접근성을 높일 수 있고, 다양한 플랫폼에서 사용이 용이하다.

생산성을 높일 수 있는 다양한 기능을 제공하는 글쓰기 IDE 도구로 문서 작성자는 글

쓰기와 고품질 문서 제작 본연의 업무에 더 많은 시간을 투여함으로써 더욱 창의적이고 효과적인 글쓰기가 가능하다.

4.1 저작방식 패러다임

위지위그(WYSIWYG: What You See Is What You Get)는 “보는 대로 얻는다”는 의미로, 사용자가 문서를 편집할 때 화면에 보이는 형태가 최종 출력물과 동일하게 나오는 편집 방식이다. 대다수 현대 워드 프로세서에서 위지위그 방식을 사용하고 있다. 사용자에게 직관적이고 쉽게 접근할 수 있는 인터페이스를 제공하기 때문이다.

그러나 위지위그 방식에도 단점은 있다. 마크다운, TeX 같은 텍스트 기반 편집 방식은 문서 호환성과 범용성을 위해 쓰이는 반면, 위지위그 저작방식은 호환성과 범용성을 다소 희생할 수 밖에 없다. 특히 복잡한 문서나 웹 페이지를 작성하는 경우, 코딩방식으로 전환하여 수작업으로 최적화를 시도하더라도 완벽한 해결이 어렵다. 예를 들어, 위지위그 편집기에서 문서를 작성하면 뒷단에 불필요한 코드나 태그가 자동으로 생성되어 문서 최적화를 방해하며, 시간이 지남에 따라 누적되어 호환성과 재현성에 심각한 문제를 야기한다.^{1 2}

위지웜(WYSIWYM: What You See Is What You Mean)은 “당신이 보는 것은 당신이 뜻하는 것이다”라는 의미로, 위지위그(WYSIWYG) 방식의 한계를 극복하기 위해 나온 대안 편집 방식이다. 위지웜 방식에서는 사용자가 무엇을 의미하는지를 중점으로 두어, 본래의 코드 구조를 더 명확하게 알 수 있다. 코드 의미를 직접적으로 반영하여, 불필요한 요소 없이 효율적으로 문서를 작성할 수 있는 장점이 있다.

RStudio의 Visual 편집 기능은 위지웜 지향점을 잘 반영하고 있다. 사용자는 복잡한 코드나 태그 없이도 의미 있는 문서 구조를 쉽게 생성하고 관리할 수 있어 문서 최적화와 호환성을 높일 수 있으며, 더욱 높은 문서 저작 생산성을 달성할 수 있다.

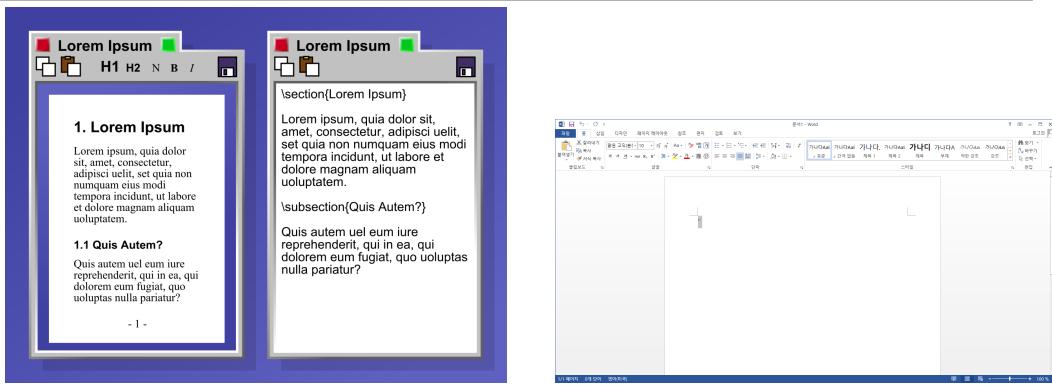
오픈 소스 LaTeX 편집기인 **LyX**는 위지웜 방식을 초기부터 채택하여 사용자에게 코드 본래 구조와 의미를 명확하게 파악할 수 있는 인터페이스를 제공했다. 이러한 접근법은 복잡한 수식이나 과학적인 문서를 작성할 때 특히 유용하며, **L^AT_EX** 복잡성을 낮추면서도 강력한 기능을 최대한 활용할 수 함으로써 위지위그 한계를 극복했다는 평가를 받고 있다.

¹위키백과, “위지위그”

²나무위키, “WYSIWYG”

문서 컴파일

위자위그



4.2 워드 프로세서³

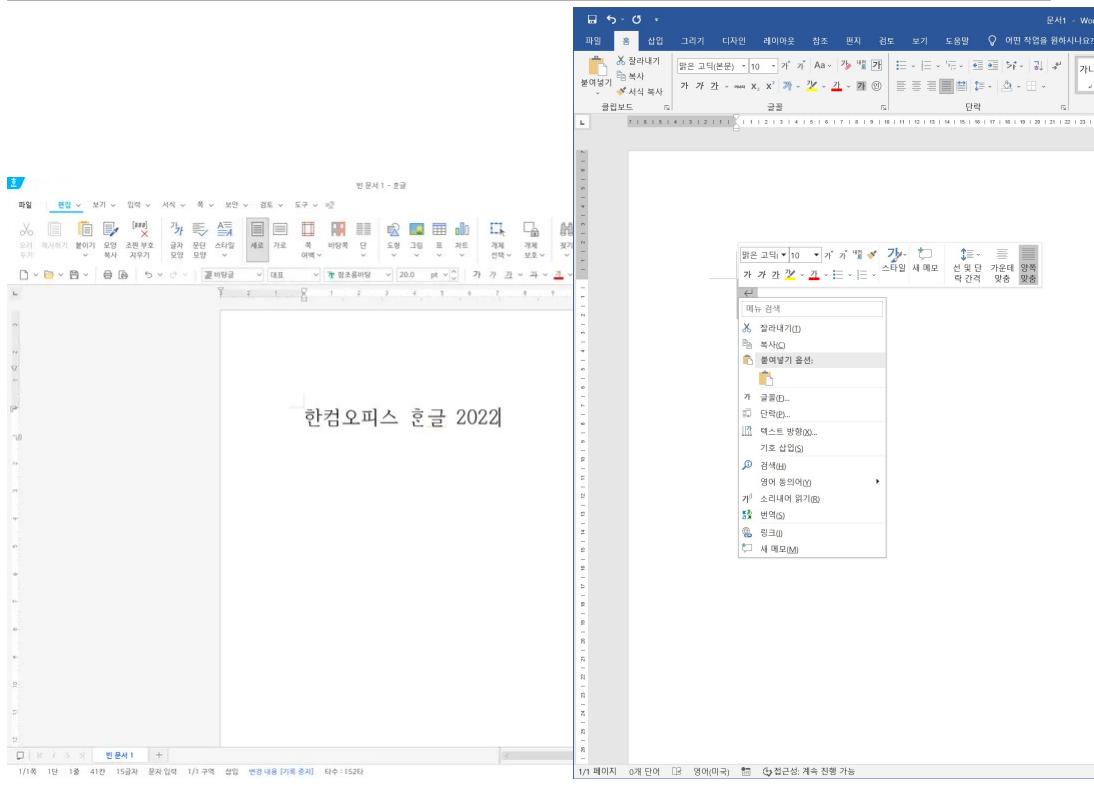
워드 프로세서(Word Processor)는 문서 제작을 위한 소프트웨어로, 시각적으로 잘 구성된 인터페이스와 다양한 편집 기능을 제공한다. 사용자는 이러한 기능을 통해 원하는 형태와 구조를 갖는 문서를 쉽게 저작할 수 있다. 워드 프로세서는 글자 스타일, 문단 구성, 이미지 삽입, 표 만들기 등 다양한 기능이 포괄적으로 제공되며, 윈도우에 기본 제공되는 메모장(NotePad) 텍스트 편집기보다 훨씬 더 풍부한 문서 작성이 가능하다. 2023년 9월 1일부로 윈도우에서 무료로 제공되던 워드패드(WordPad)에 대한 지원도 없어지고 윈도우에서도 제거될 것이라는 발표가 있었다. 따라서, 서식이 필요한 문서를 저작할 경우 MS 워드(Word)나 아래한글과 같은 워드 프로세서를 사용해야 된다.

아래한글은 국내에서 널리 사용되는 워드 프로세서 중 하나로, 한국 문화와 업무 환경에 맞춰 특화된 기능을 제공한다. 정부나 공공기관에서 사용하는 특별한 문서 양식을 미리 저장해 두어, 사용자가 양식을 찾는 데 시간을 쓰지 않고 글쓰기에 바로 집중할 수 있도록 큰 도움을 주었다.

³Jesal Shethna, “Microsoft Word Features”, EDUCBA

국산 워드 프로세서

해외 워드 프로세서



4.3 *LATEX* 편집기

LATEX 편집기는 운영체제나 사용자의 필요에 따라 다양한 선택옵션이 존재한다. 특히, 클라우드 기반 Overleaf, 설치형 TeXstudio, 데이터 과학에 특화된 RStudio IDE가 대표적이다. 편집기들은 *LATEX*에 특화되었거나, 일부 *LATEX* 기능을 분리하여 사용자 요구에 맞춰 활용할 수 있다.

Overleaf는 초기에 ShareLaTeX 서비스로 시작해 현재는 클라우드 기반의 *LATEX* 편집 기능을 제공한다. 클라우드 특성을 살려 문서의 공유와 협업이 용이하다는 점이 큰 장점이다.

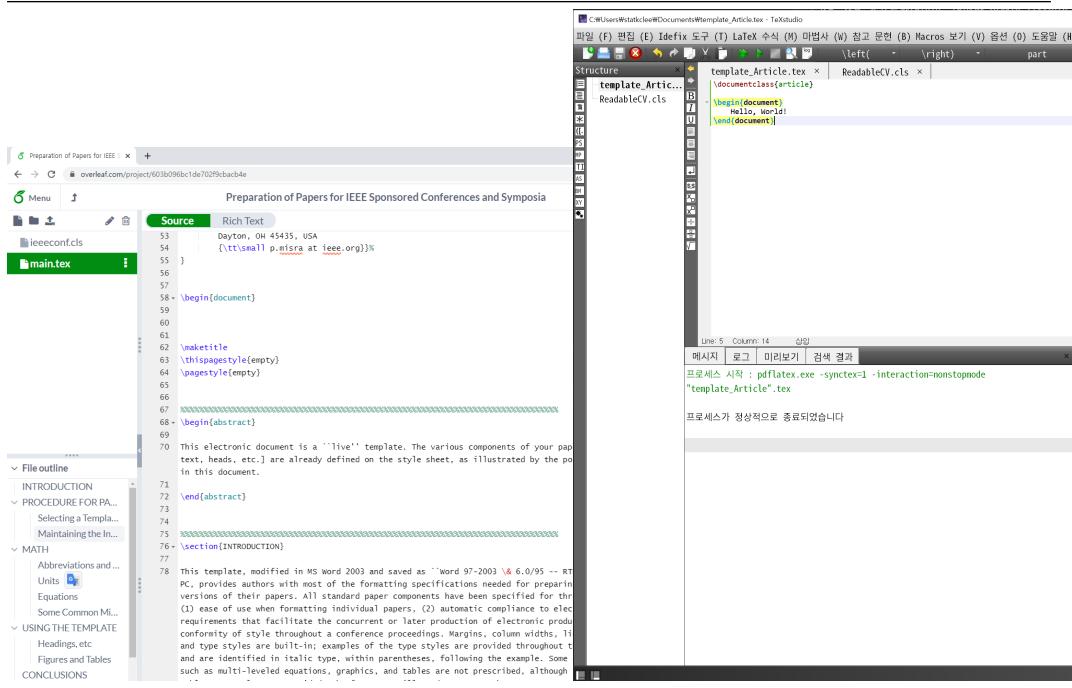
TeXstudio는 Texmaker 후속으로 오픈 소스 *LATEX* 편집기로 제공된다. 사용자는 TeXstudio 웹사이트에서 다운로드 받아 설치할 수 있으며, GitHub [texstudio](#) 저장소를 통해 한국어 현지화 작업에도 참여할 수 있다.



그림 4.2: **TEX** 편집기 다양성

클라우드 편집기

설치형 편집기



4.4 RStudio IDE 편집기

RStudio IDE는 처음 데이터 과학 R 프로그래밍 언어를 위한 통합개발환경으로 시작하였으나 이제는 파이썬을 비롯하여 SQL, Observable JS 등 데이터 과학 전분야를 담당하고 있다. 특히, 데이터 분석, 시각화 및 문서 작업을 간편하게 할 수 있도록 다양한 도구와 기능을 제공할 뿐만 아니라, Shiny 웹 애플리케이션 개발, Plumber를 통한 API 개발, 쿼토(Quarto)/R 마크다운을 활용해 다양한 데이터 과학 산출물을 제작할 수 있다. 또한, Git 버전 관리 시스템과 통합되어 코드 이력을 쉽게 추적하고 관리할 수 있고 GitHub과 연결하여 협업기능도 지원한다.

RStudio 코드 편집기는 자동 완성, 구문 강조, 맞춤법 검사 등 기능을 제공하여 개발자 편의를 향상시켰고, GUI 데이터 뷰어를 통해 데이터프레임과 데이터 객체를 직접적으로 시각적으로 살펴볼 수 있으며, 내장 패키지 관리 시스템을 통해 R 패키지도 손쉽게 설치하고 관리할 수 있다.

웹 앱 Shiny 애플리케이션 개발과 테스트도 가능하고, 문서화 프로그래밍을 차세대 R 마

크다운 큐토(Quarto)를 이용하여 코드, 데이터, 그래프를 하나의 문서로 통합하여 구현할 수 있을 뿐만 아니라 팬독(Pandoc)을 통해 다양한 형태 문서를 자동으로 생성할 수 있다.

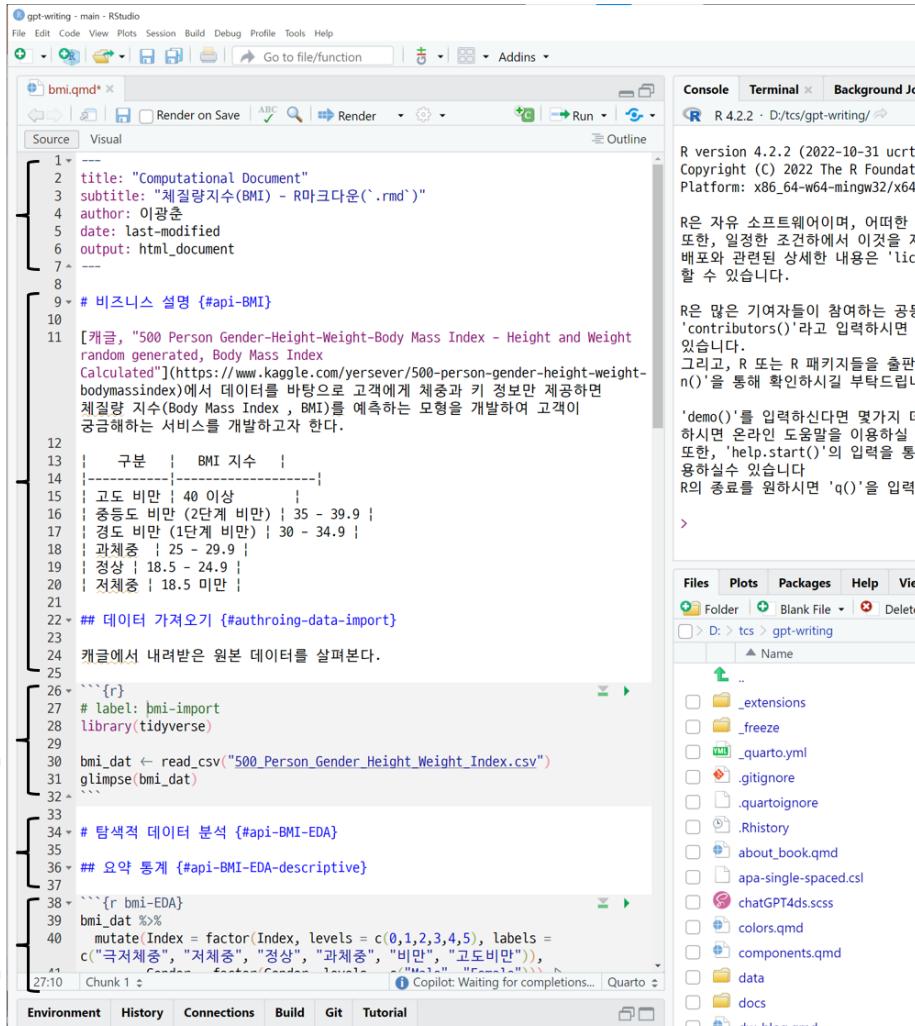
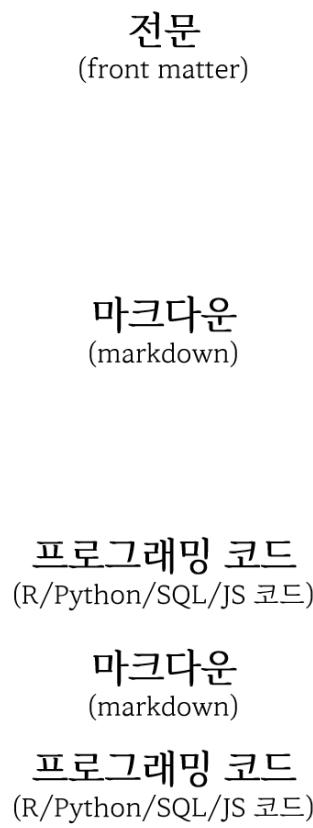


그림 4.3: RStudio 문서화 프로그래밍 사례

4.4.1 Visual 마크다운 편집기

Visual 마크다운 기능을 사용하게 되면 과학/기술 문서 작성의 용이성 뿐만 아니라 인용(Citation), 문학적 프로그래밍(literate programming)을 통한 재현가능한 과학문서 구현, 팬독(Pandoc)을 사용하여 텍스트와 코드로 PDF, HTML, 워드 등 다양한 문서 동시 생

성이 가능하다.

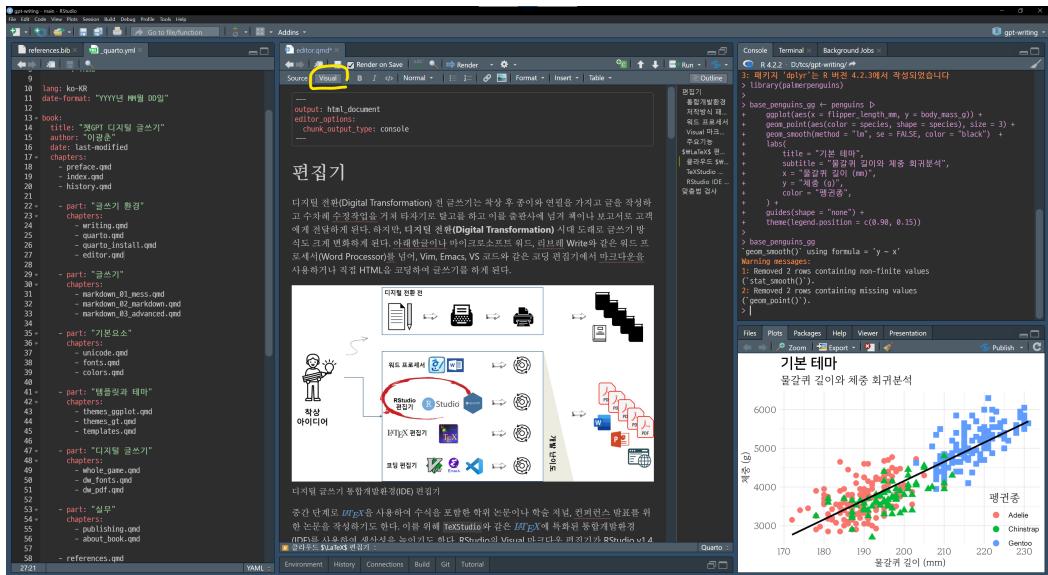


그림 4.4: RStudio Visual 마크다운 편집기능

4.4.2 맞춤법 검사

4.5 쿼토 설치

쿼토(Quarto) 웹사이트에서 Quarto CLI 엔진과 통합개발도구(IDE)를 설치한다. 쿼토 CLI를 지원하는 IDE는 VS Code, RStudio, Jupyter, VIM/Emacs 와 같은 텍스트 편집기가 포함된다. IDE까지 설치를 했다면 문학적 프로그래밍(literate programming)을 통해 사람이 저작하는 마크다운(Markdown)과 기계가 저작하는 프로그래밍 언어(R/Python/SQL/줄리아/자바스크립트)를 결합한 다양한 문서저작을 시작할 수 있다.

4.5.1 윈도우 설치

쿼토(Quarto)를 윈도우 운영체제에 설치하기 위해서는 몇 가지 단계를 거쳐야 한다.

4.5.2 쿼토 다운로드

먼저 Quarto 공식 웹사이트에서 윈도우용 설치 파일을 다운로드한다. 웹사이트 “Download Quarto CLI” 윈도우 버전을 선택하여 다운로드한다.



그림 4.5: 쿼토 도구모음

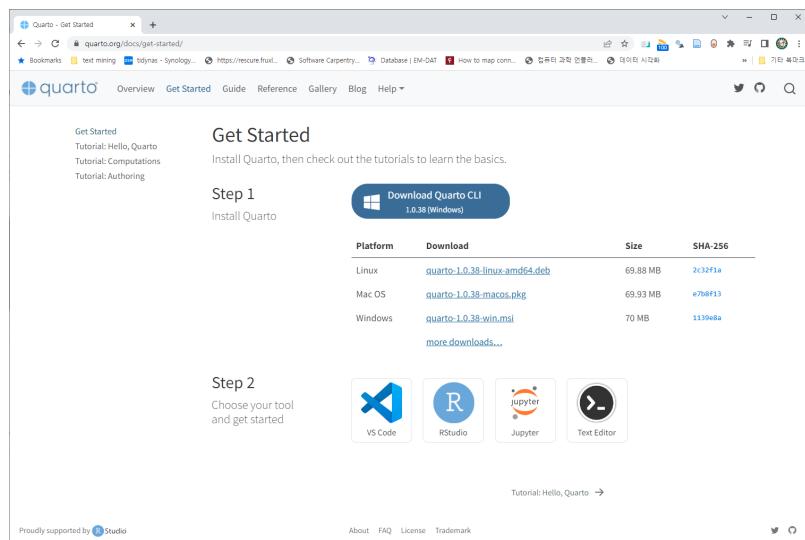


그림 4.6: 쿼토 다운로드 화면

4.5.2.1 쿼토 설치

다운로드한 파일을 더블클릭 실행하여 설치를 진행한다. 설치 마법사가 나타나면 지시에 따라 설치를 완료하면 된다. 설치가 완료되면 환경 변수에 쿼토 설치 경로를 등록해야 한다. ‘제어판’을 열고 ‘시스템과 보안’으로 이동한 다음 ‘시스템’을 선택한다. ‘고급 시스템 설정’을 클릭한 후 ‘환경 변수’ 버튼을 누른다. ‘시스템 변수’에서 ‘Path’ 변수를 찾아 Quarto의 설치 경로를 추가한다. 일반적인 경로는 C:\Users\<사용자계정명>\AppData\Local\Programs\Quarto\bin과 같을 수 있다.



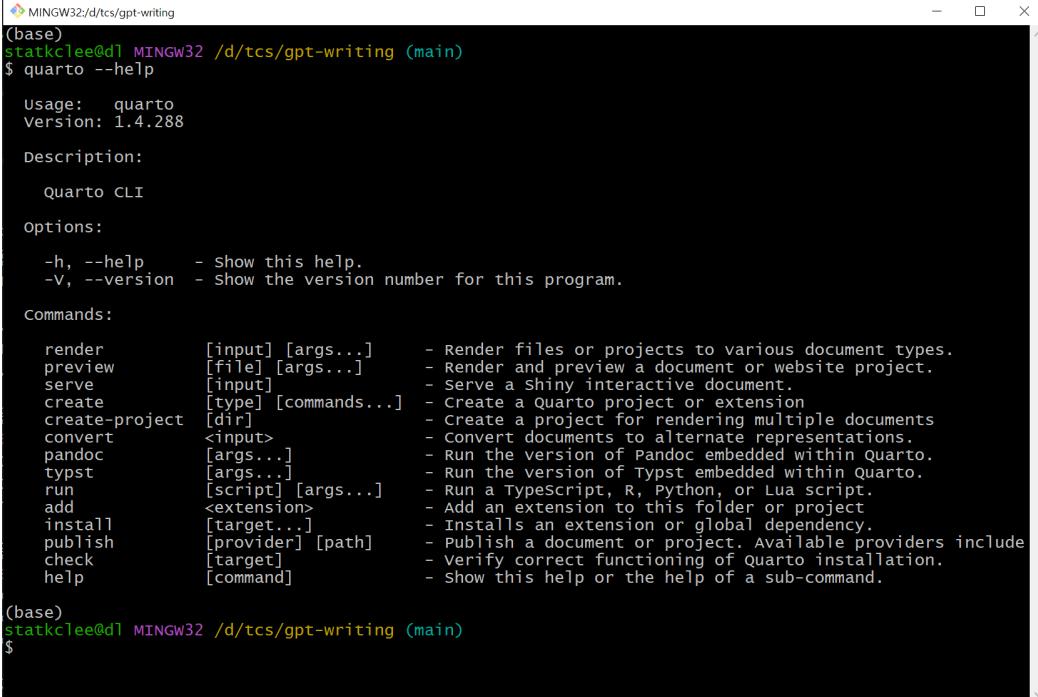
그림 4.7: 쿼토 설치완료

4.5.3 쿼토 CLI

마지막으로 환경 변수 설정이 올바르게 이루어졌는지 확인하기 위해 명령 프롬프트나 터미널을 열어 `quarto --version` 혹은 `quarto --help` 명령을 입력한다. 올바른 버전 번호가 출력되면 설치가 성공적으로 완료된 것이다.

💡 힌트

윈도우 시스템의 경우 초기 윈도우에서 쿼토 실행명령이 `quarto.cmd`이였으나 `quarto.exe`도 지원된다. 즉, 제어판 → 환경 변수 설정 ...에서 ”C:\Users\사용자명\AppData\Local\Programs\Quarto\bin 디렉토리를 등록한 후 `quarto` 명령어를 사용



```
MINGW32/d/tcs/gpt-writing
(base) statkclee@d1 MINGW32 /d/tcs/gpt-writing (main)
$ quarto --help

Usage:   quarto
Version: 1.4.288

Description:
    Quarto CLI

options:
    -h, --help      - Show this help.
    -V, --version   - Show the version number for this program.

commands:
    render          [input] [args...]
    preview         [file] [args...]
    serve           [input]
    create          [type] [commands...]
    create-project  [dir]
    convert         <input>
    pandoc         [args...]
    typst           [args...]
    run             [script] [args...]
    add             <extension>
    install         [target...]
    publish         [provider] [path]
    check           [target]
    help            [command]

    render          [input] [args...]
    preview         [file] [args...]
    serve           [input]
    create          [type] [commands...]
    create-project  [dir]
    convert         <input>
    pandoc         [args...]
    typst           [args...]
    run             [script] [args...]
    add             <extension>
    install         [target...]
    publish         [provider] [path]
    check           [target]
    help            [command]

(base)
statkclee@d1 MINGW32 /d/tcs/gpt-writing (main)
$
```

그림 4.8: 쿼토 CLI 실행화면

한다.

```
Sys.which("quarto")
```

```
quarto
```

```
"C:\\\\Users\\\\사용자명\\\\AppData\\\\Local\\\\Programs\\\\Quarto\\\\bin\\\\quarto.exe"
```

4.6 부조종사

GitHub 부조종사(Copilot)를 Rstudio에서 사용하기 위해서는 특별한 버전 Rstudio, 즉 **일일 빌드 (Daily Builds)**가 필요하여 각자 운영체제에 맞는 RStudio Desktop 버전을 다운로드한 후 설치한다.

Tools → Global Options → Copilot → Enable Github Copilot

설치가 완료되면 Rstudio를 실행하고 상단 메뉴에서 ‘Tools’를 선택한 다음 ‘Global Options’을 클릭하면, 왼쪽 사이드바에서 ‘Copilot’을 선택하고 ‘Enable Github Copilot’ 체크박스를 선택한 다음 ‘Sign In’을 클릭하고 나타나는 링크에서 인증 코드를 입력한다. ‘Authorize Github Copilot Plugin’을 클릭하여 인증 과정을 완료하면 설정이 완료된다.

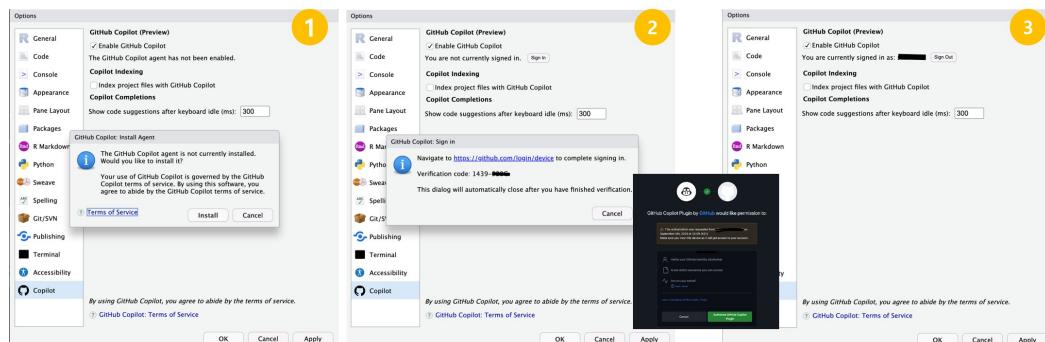


그림 4.9: GitHub 부주종사 설치과정

GitHub Copilot를 Rstudio에서 사용해보면, 작성하려는 코드가 회색으로 예측되어 나타난다. ‘Tab’ 키를 눌러 탭완성(Tab Completion) 기능으로 제시한 코드를 수락하여 개발을 이어간다.

4.7 출판 플랫폼

데이터 과학 분야에서 산출물을 공유하고 출판하는 것은 매우 중요한 단계로 효과적인 출판과 비용적인 면을 고려하여 최적 플랫폼과 도구를 선정한다. 데이터 과학 분야를 개척한 RStudio IDE로 과거 R 마크다운 산출물을 RPubs에 출판한 경험이 있다면, Quarto Pub은 그와 유사한 경험을 제공한다. 다른 대안으로는 netlify, GitHub Pages, RStudio Connect 등이 있다.

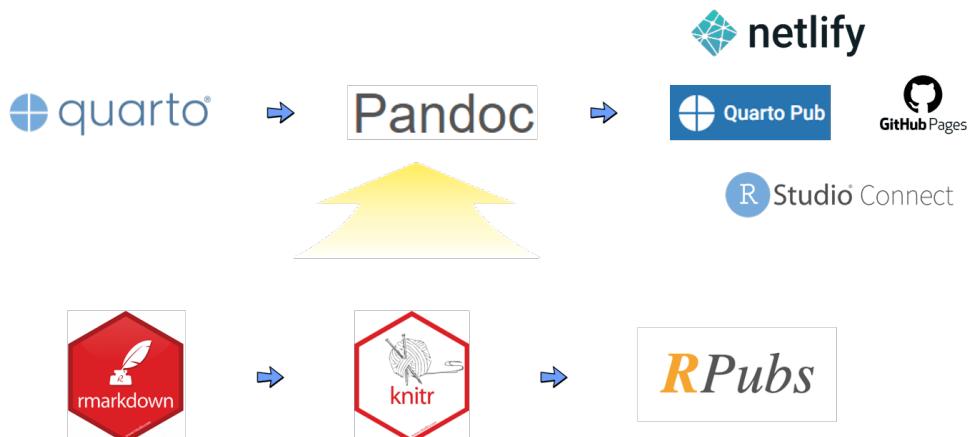


그림 4.10: 쿼토 출판

제 5 장

유니코드와 UTF-8

사람 간의 의사소통은 다양한 기호 체계를 통해 이루어진다. 영어 알파벳, 한글, 한자 등 문자가 의사소통에 사용되는 좋은 예다. 디지털 환경에서 이러한 의사소통을 가능하게 하는 기술적 장치가 바로 문자집합과 문자 인코딩과 디코딩이다.

컴퓨터 시스템은 이진수 바이트를 기본 단위로 사용한다. 바이트는 파일 형태로 묶이거나 네트워크를 통해 전송되어 다른 시스템에 도달한다. 이 데이터가 사람에게 의미 있는 정보로 전달되기 위해서는 인코딩(부호화)과 디코딩(복호화) 과정을 거쳐야 한다.

컴퓨터 시스템은 데이터를 바이트(Byte) 형태로 처리한다. 이 바이트 데이터는 이진수, 즉 010101과 같은 형태로 표현되고, 바이트 데이터를 사람이 읽을 수 있는 문자로 변환하는 최초의 표준이 ASCII(아스키)다. 하지만 ASCII는 256개 문자만을 지원하기 때문에, CJK(중국, 일본, 한국)와 같은 동아시아 문화권에서는 그 한계가 명확하다. 이러한 한계를 해결하기 위해 유니코드(Unicode)가 도입되었다. 유니코드는 영문자는 물론이고 지구상의 거의 모든 문자와 기호를 디지털로 표현할 수 있는 방법을 제공한다.

유니코드(Unicode)는 글자와 코드가 1:1 매핑되어 있는 단순한 코드표에 불과하고 산업표준으로 일종의 국가 당사자간 약속이다. 한글이 표현된 유니코드 영역도 **위키백과**와 **유니코드 영역**에서 찾을 수 있다.

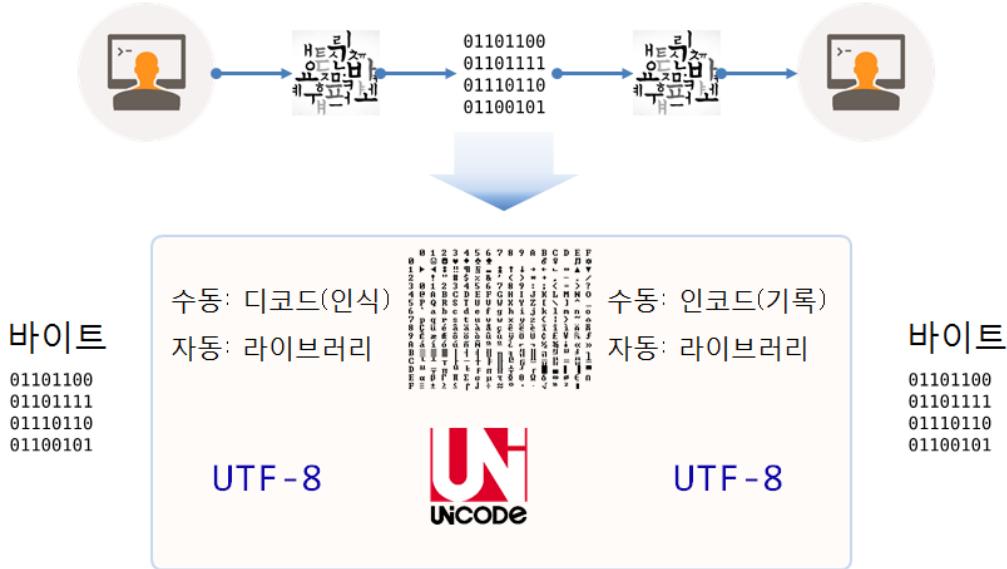


그림 5.1: 유니코드와 UTF-8

i 인코딩 (Encoding)

문자 인코딩(character encoding) 줄여서 인코딩은 사용자가 입력한 문자나 기호들을 컴퓨터가 이용할 수 있는 신호로 만드는 것을 말한다. 넓은 의미의 컴퓨터는 이러한 신호를 입력받고 처리하는 기계를 뜻하며, 신호 처리 시스템을 통해 이렇게 처리된 정보를 사용자가 이해할 수 있게 된다.

All text has a character encoding.

5.1 인코딩 문제

문자 인코딩은 컴퓨터가 텍스트를 바이트로 변환하거나 바이트를 텍스트로 변환하는 방법이다. 인코딩 과정에서는 다양한 문제가 발생할 수 있고, 그 중 세 가지 문제가 많이 알려져 있다. 첫번째는 '두부(Tofu)'라 불리는 상황으로, 컴퓨터가 어떤 문자를 표현해야 할지 알지만, 화면에 어떻게 출력해야 할지 모르기 때문에 빈 사각형 상자로 표시된다. 두번째는 '문자깨짐(Mojibake, 文字化け)'이다. 특히 일본어에서 자주 발생하며, 한 인코딩 방식으로 작성된 텍스트가 다른 인코딩 방식으로 해석될 때 문자가 깨지는 현상을 의미한다. 세번째는 '의문부호(Question Marks)'로, 특정 문자가 다른 문자로 변환될 때 발생된다. 문자집합과 인코딩 궁합이 맞지 않을 때 발생하며, 데이터 손실과 오류

도 야기된다.

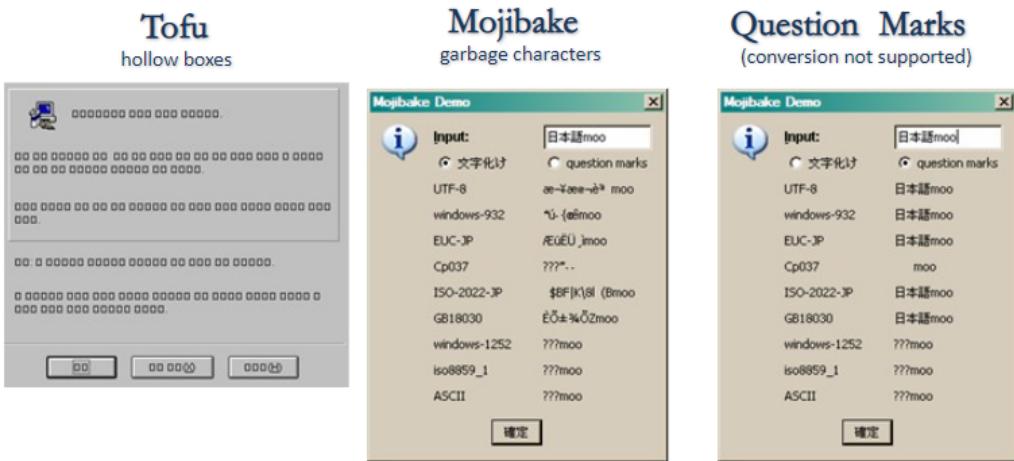


그림 5.2: 세가지 인코딩 문제

5.2 문자 집합

5.2.1 아스키 코드

디지털 글쓰기는 내용과 상관없이 결국 텍스트로 표현되고, 텍스트는 단지 문자다. 하지만, 컴퓨터가 문자 하나를 어떻게 표현할까?

1960년대 미국식 영문자를 컴퓨터로 표현하는 해결책은 간단했다 - 알파벳 26개(대문자, 소문자), 숫자 10, 구두점 몇개, 그리고 전신을 보내던 시절에 제어를 위해 사용된 몇 개 특수 문자("새줄로 이동", "본문 시작", "개행", "경고음", 등). 모두 합쳐도 128개보다 적어서, 아스키(ASCII) 위원회가 문자마다 7비트($2^7 = 128$)를 사용하는 인코딩으로 표준화했다.¹

그림 ??에 아스키 문자표에 제어문자 10개와 출력가능 아스키 문자표 중 영문 대문자 A-I까지 10개를 뽑아 사례로 보여주고 있다. 즉, 문자표는 어떤 문자가 어떤 숫자에 해당하는지를 정의하고 있다.

¹ 미국정보교환표준부호(American Standard Code for Information Interchange, ASCII)는 영문 알파벳을 사용하는 대표적인 문자 인코딩으로 컴퓨터와 통신 장비를 비롯한 문자를 사용하는 많은 장치에서 사용되며, 대부분의 문자 인코딩이 아스키에 기초하고 있다.

이진법	팔진법	십진법	십육진법	약자	한국어_설명	이진법	팔진법	십진법	십육진법	모양
000 0000	0	0	00	NUL	NULL	1000000	100	64	40	@
000 0001	1	1	01	SOH	헤더 시작	1000001	101	65	41	A
000 0010	2	2	02	STX	본문 시작, 헤더 종료	1000010	102	66	42	B
000 0011	3	3	03	ETX	본문 종료	1000011	103	67	43	C
000 0100	4	4	04	EOT	전송 종료	1000100	104	68	44	D
000 0101	5	5	05	ENQ	응답 요구	1000101	105	69	45	E
000 0110	6	6	06	ACK	긍정응답	1000110	106	70	46	F
000 0111	7	7	07	BEL	경고음	1000111	107	71	47	G
000 1000	10	8	08	BS	백스페이스	1001000	110	72	48	H
000 1001	11	9	09	HT	수평 탭	1001001	111	73	49	I

그림 5.3: 제어문자와 출력가능한 아스키 문자표 알파벳 예시

5.2.2 확장 아스키

아스키(ASCII) 방식으로 숫자 2, 문자 q, 혹은 곡절 악센트 ^ 를 표현하는데 충분하다. 하지만, 투르크어족 추바시어 ě, 그리스 문자 β, 러시아 키릴문자 я는 어떻게 저장하고 표현해야 할까? 7-비트를 사용하면 0에서 127까지 숫자를 부여할 수 있지만, 8-비트(즉, 1 바이트)를 사용하게 되면 255까지 표현할 수 있다. 그렇다면, ASCII 표준을 확장해서 추가되는 128개 숫자에 대해 추가로 문자를 표현할 수 있게 된다.

- 아스키: 0…127
- 확장된 아스키: 128…255

불행하게도, 영어문자를 사용하지 않는 세계 곳곳에서 많은 사람들이 시도를 했지만, 방식도 다르고, 호환이 되지 않는 방식으로 작업이 되어, 결과는 엉망진창이 되었다. 예를 들어, 실제 텍스트가 불가리아어로 인코딩되었는데 스페인어 규칙을 사용해서 인코딩한 것으로 프로그램이 간주하고 처리될 경우 결과는 무의미한 횡설수설 값이 출력된다. 이와는 별도로 한중일(CJK) 동아시아 국가들을 비롯한 많은 국가들이 256개 이상 기호를 사용한다. 왜냐하면 8-비트로 특히 동아시아 국가 문자를 표현하는데 부족하기 때문이다.

5.2.3 한글 완성형과 조합형

1980년대부터 컴퓨터를 사용하신 분이면 완성형과 조합형의 표준화 전쟁을 지켜봤을 것이고, 그 이면에는 한글 워드프로세서에 대한 주도권 쟁탈전이 있었던 것을 기억할

것이다. 결국 완성형과 조합형을 모두 포용하는 것으로 마무리 되었지만, 여기서 끝난 게 끝난 것이 아니다. 유닉스 계열에서 KSC5601을 표준으로 받아들인 EUC-KR과 90년대와 2000년대를 호령한 마이크로소프트 CP949 가 있었다. 결국 대한민국 정부에서 주도한 표준화 전쟁은 유닉스/리눅스, 마이크로소프트 모두를 녹여내는 것으로 마무리 되었고, 웹과 모바일 시대는 유니코드로 넘어가서 KSC5601이 유니코드의 원소로 들어가는 것으로 마무리 되었다.

이제 신경쓸 것은 인코딩 … utf-8 만 신경쓰면 된다. 그리고 남은 디지털 레거시 유산을 잘 처리하면 된다.

i 유닉스/리눅스(EUC-KR), 윈도우(CP949)

EUC-KR, CP949 모두 2바이트 한글을 표현하는 방식으로 동일점이 있지만, EUC-KR 방식은 KSC5601-87 완성형을 초기 사용하였으나, KSC5601-92 조합형도 사용할 수 있도록 확장되었다. CP949는 확장 완성형으로도 불리며 EUC-KR에서 표현할 수 없는 한글글자 8,822자를 추가한 것으로 마이크로소프트 코드페이지(Code Page) 949를 사용하면서 일반화되었다.

5.2.4 유니코드

1990년대 나타나기 시작한 해결책을 **유니코드(Unicode)** 라고 부른다. 예를 들어, 영어 A 대문자는 1 바이트, 한글 가는 3 바이트다. 유니코드는 정수값을 서로 다른 수만개 문자와 기호를 표현하는데 정의한다. 'A'는 U+0041, '가'는 U+AC00과 같이 고유한 코드 포인트를 가진다. 하지만, 파일에 혹은 메모리에 문자열로 정수값을 저장하는 방식을 정의하지는 않는다.

각 문자마다 8-비트를 사용하던 방식에서 32-비트 정수를 사용하는 방식으로 전환하면 되지만, 영어, 에스토니아어, 브라질 포르투칼어 같은 알파벳 언어권에는 상당한 공간 낭비가 발생된다. 접근 속도가 중요한 경우 메모리에 문자당 32 비트를 종종 사용한다. 하지만, 파일에 데이터를 저장하거나 인터넷을 통해 전송하는 경우 대부분의 프로그램과 프로그래머는 이와는 다른 방식을 사용한다.

다른 방식은 (거의) 항상 **UTF-8** 으로 불리는 인코딩으로, 문자마다 가변 바이트를 사용한다. 하위 호환성을 위해, 첫 128개 문자(즉, 구 아스키 문자집합)는 바이트 1개에 저장된다. 다음 1920개 문자는 바이트 2개를 사용해서 저장된다. 다음 61,000은 바이트 3

개를 사용해서 저장해 나간다.

궁금하면, 동작 방식이 다음 표에 나타나 있다. “전통적” 문자열은 문자마다 1 바이트를 사용한다. 반대로, “유니코드” 문자열은 문자마다 충분한 메모리를 사용해서 어떤 텍스트 유형이든 저장한다. R, 파이썬 3.x에서 모든 문자열은 유니코드다. 엄청난 바이트를 읽어오거나 저장하여 내보내려고 할 때, 인코딩을 지정하는 것은 엄청난 고통이다.

유니코드 문자열은 여는 인용부호 앞에 소문자 `u`를 붙여 표시한다. 유니코드 문자열을 바이트 문자열로 전환하려면, 인코딩을 명시해야만 된다. 항상 UTF-8을 사용해야만 되고, 그밖의 인코딩을 사용하는 경우 매우, 매우 특별히 좋은 사유가 있어야만 된다. 특별한 인코딩을 사용하는 경우 두번 생각해 보라.

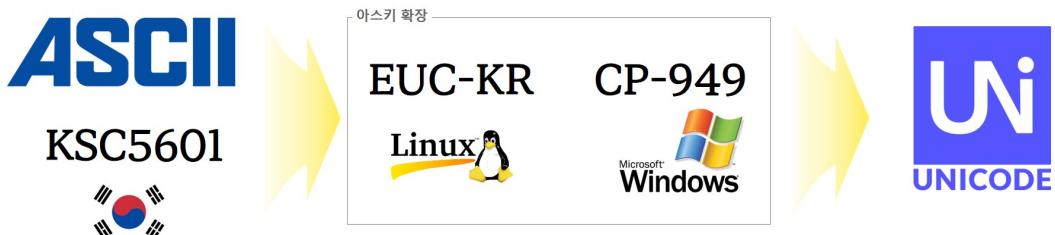


그림 5.4: 아스키에서 유니코드로 진화과정

컴퓨터가 처음 등장할 때 미국 영어권 중심 아스키가 아니고 4바이트 전세계 모든 글자를 표현할 수 있는 유니코드가 사용되었다면 한글을 컴퓨터에 표현하기 위한 지금과 같은 번거로움은 없었을 것이다. 돌이켜보면 초기 컴퓨터가 저장용량 한계로 인해 유니코드가 표준으로 자리를 잡더라도 실용적인 이유로 인해서 한글을 컴퓨터에 표현하기 위한 다른 대안이 제시됐을 것도 분명해 보인다. 초창기 영어권을 중심으로 아스키 표준이 정립되어 현재까지 내려오고, 유니코드와 UTF-8 인코딩이 사실상 표준으로 자리잡았으며, 그 사이 유닉스/리눅스 EUC-KR, 윈도우즈 CP949가 빈틈을 한동안 매우면서 역할을 담당했다.

EUC-KR				
항목	ASCII (1963)	(1980s)	CP949 (1990s)	Unicode (1991)
범위	128개의 문자	2,350개의 한글 문자 등	약 11,172개의 완성형 한글 문 자 등	143,859개의 문 자 (버전 13.0 기준)

EUC-KR				
항목	ASCII (1963)	(1980s)	CP949 (1990s)	Unicode (1991)
비트 수	7비트	8~16비트	8~16비트	다양한 인코딩 방식 (UTF-8, UTF-16, UTF-32 등)
표준	ANSI, ISO/IEC 646	KS X 2901	마이크로소프 트	ISO/IEC 10646
플랫폼	다양한 시스템	UNIX 계열, 일 부 Windows	Windows 계열	다양한 플랫폼
문자 집합	영문 알파벳, 숫자, 특수 문 자	한글, 영문 알 파벳, 숫자, 특 수 문자	한글, 한자, 영 문 알파벳, 숫자 자, 특수 문자	전 세계 언어, 특수 문자, 이 모티콘 등
확장성	확장 불가능	한정적	더 많은 문자 지원	높은 확장성
국제성	영어 중심	한국어 중심	한국어 중심	다국어 지원
유니코드 호환	호환 가능 (U+0000 ~ U+007F)	호환 불가, 변 환 필요	유니코드와 상 호 변환 가능	자체가 표준

5.3 UTF-8

UTF-8(Universal Coded Character Set + Transformation Format – 8-bit의 약자)은 유니코드 중에서 가장 널리 쓰이는 인코딩으로, 유니코드를 위한 가변 길이 문자 인코딩 방식 중 하나로 켄 톰프슨과 롭 파이크가 제작했다.

UTF-8 인코딩의 가장 큰 장점은 아스키(ASCII), 라틴-1(ISO-8859-1)과 호환되어, 문서를 처리하는 경우 아스키, 라틴-1 문서를 변환 없이 그대로 처리할 수 있고 영어를 비롯한 라틴계열 문서로 저장할 때 용량이 매우 작다. 이러한 이유로 많은 오픈소스 소프트웨어와 데이터를 생산하는 미국을 비롯한 유럽언어권에서 UTF-8이 많이 사용되고 있지만, 한글은 한 글자당 3바이트 용량을 차지한다.

5.3.1 웹 표준 인코딩

스마트폰의 대중화에 따라 더이상 윈도우 운영체제에서 사용되는 문자체계가 더이상 표준이 되지 못하고 여러 문제점을 야기함에 따라 유니코드 + UTF-8 체제가 대세로 자리잡고 있는 것이 확연히 나타나고 있다.

2010년 구글에서 발표한 자료에 의하면 2010년 UTF-8 인코딩이 웹에서 주류로 부상하기 시작한 것이 확인되었다. [unicode2010] 웹기반 플롯 디지털 도구를 활용하여 그래프(WebPlotDigitizer)에서 데이터를 추출하여 시각화면 유사한 결과를 시각적으로 표현할 수 있다. 2010년 이후 웹에서 가장 점유율이 높은 인코딩 방식은 UTF-8으로 W3Tech 웹기술조사(Web Technology Surveys)를 통해 확인을 할 수 있다. 여기서 주목할 점은, 프랑스어, 독일어, 스페인어와 같은 서유럽 언어의 문자와 기호 표현하는 ISO-8859-1 인코딩, 종종 “Latin-1”으로 불리는 8비트 문자 인코딩이 현저히 줄고 있다는 점이다.

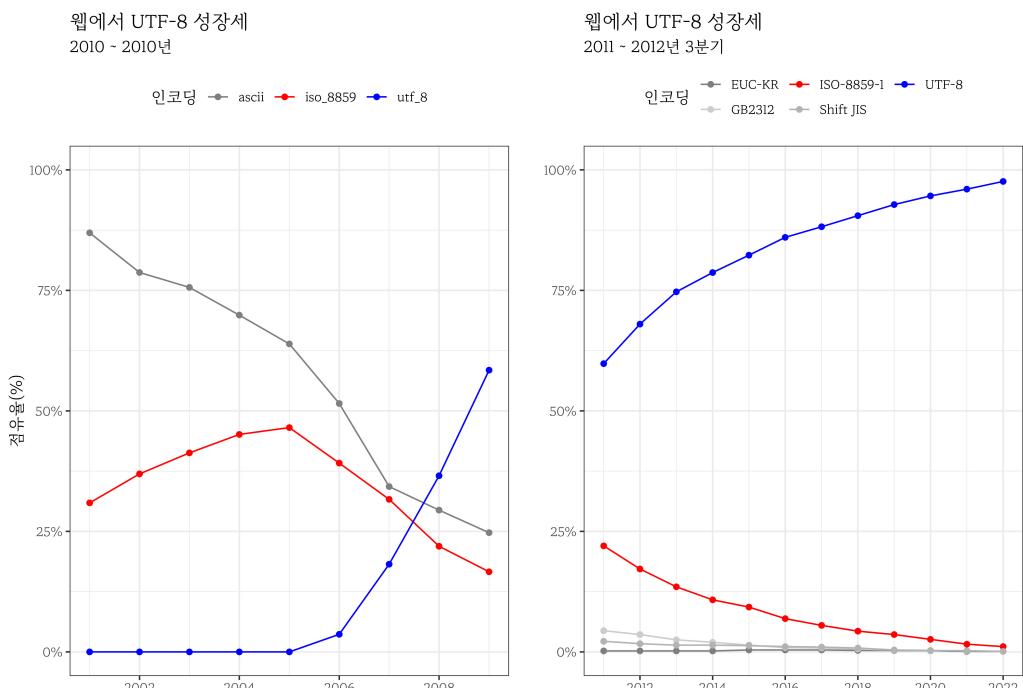


그림 5.5: 2010 ~ 2012 웹에서 UTF-8 성장세

5.4 텍스트 표현

1940년대, 1950년대 편치카드 기술로 정의된 첫번째 방식이 고정폭 레코드(fixed-width records)를 사용하는 것으로, 각 줄마다 동일한 고정길이를 갖는다. 예를 들어, 다음 일본 전통 단시 하이쿠(haiku)를 컴퓨터로 아래와 같이 레코드 3개로 배열했다.(점 문자는 “사용되지 않음”을 의미) 이런 방식은 여전히 데이터베이스에도 사용되고 있다.

일본 전통 단시 하이쿠 예시

```
A crash reduces
your expensive computer
to a simple stone.
```

A		c	r	a	s	h		r	e	d	u	c	e	s	
y	o	u	r		e	x	p	e	n	s	i	v	e		c	o	m	p	u	t	e	r
t	o		a		s	i	m	p	l	e		s	t	o	n	e

그림 5.6: 고정길이 일본 전통 하이쿠 표현

이런 표기법은 N번째 행 앞으로 뒤로 건너뛰기 쉬운데 이유는 각 행이 동일한 크기를 갖기 때문이다. 하지만, 공간을 낭비하는 약점이 있고, 행마다 얼마나 긴 최대 길이를 갖느냐에 관계없이, 궁극적으로 더 긴 길이를 갖는 행을 기준으로 처리해야만 된다.

시간이 흐름에 따라 개발자 대부분은 다른 표현법으로 전환했다. 전환된 표현법에 따르면 텍스트는 단지 연속된 바이트(byte)에 불과하고, 이런 연속된 바이트 일부에 “현재 라인은 여기서 종료”라는 의미가 담겨진다. 이러한 표현법으로 일본 전통 단시 하이ку를 다시 표현하면 다음과 같다.

A		c	r	a	s	h		r	e	d	u	c	e	s	█	y	o	u	r	e	x	p	e	n	s	i	v
e		c	o	m	p	u	t	e	r	█	t	o	a		s	i	m	p	l	e		s	t	o	n	e	.

그림 5.7: 변동길이 일본 전통 하이ку 표현

회색칸이 “행의 끝(end of line)”을 의미한다. 이런 표기법은 더 유연하고, 공간을 덜 낭비하지만, N번째 행 앞으로 뒤로 건너뛰는 것은 어렵게 되었다. 이유는 각각이 다른 길이를 갖기 때문이다. 물론 행 종료를 표시하는데 무엇을 사용할지 결정해야하는 문제

가 남았다. 불행히도, 유닉스에서는 행의 끝으로 개행 문자(newline) 한개, \n으로 정했지만, 윈도우에서는 행의 끝으로 복귀문자(carriage return) 다음에 개행문자, \r\n으로 정했다.

편집기 대부분에서 이런 차이를 탐지하고 처리할 수 있지만, 유닉스와 윈도우를 모두 다뤄야 되는 프로그래머에게는 여전히 성가신 일이다. 윈도우 운영체제 파일에서 데이터를 불러읽어올 경우, 파일에서 \r\n 을 \n 으로 전환하고, 데이터를 써서 저장할 경우 반대 방식으로 전환한다. 그러나 이미지, 소리 또는 기타 이진 파일형식에서 \r 또는 \n에 해당하는 문자가 우연히 포함되어 있다면, 월치 않는 변환이 발생할 수 있다.

5.5 현지화/세계화

현지화(Localization)는 세계화(internationalization)의 동전의 양면과 같다. 세계화를 영어로 internationalization으로 길기 때문에 i18n으로 줄여서 현지화는 영어로 Localization으로 길기 때문에 동일한 로직으로 L10N으로 줄여 표현한다. 현지화에 해당되는 사항은 다음이 포함된다. [oliver2023internationalization]

- 문자 집합
- 통화
- 날씨 온도(°C/°F)
- 길이 (킬로미터, 마일)
- 날짜와 시간
- 키보드 배열
- 좌측에서 우측으로, 위에서 아래, 우측에서 좌측으로 텍스트 작성 방식과 문서양식
- ...

제 6 장

LaTeX과 PDF

LaTeX(“라텍” 혹은 “레이텍”으로 읽음)을 간단히 정의하자면 “논리적인 디자인”이라고 할 수 있다. 작품이 생성되는 과정에서는 작가가 원고를 수기로 작성하거나 타자기로 입력한 뒤 이를 출판사에 제출한다. 그 후, 출판사의 편집 디자이너는 원고를 검토하여 세부적인 출력 형식을 결정하고 이를 인쇄소에 전달한다. 인쇄소에서는 이 정보를 바탕으로 과거에는 식자공이 식자판을 제작했으나, 현재는 컴퓨터가 파일을 생성한다. 그림 ??에 문서 디자인과 문서 논리 구조에 대해 이해를 위해 도식화했다.

미국 스탠포드 대학 크누스(Donald Knuth) 교수가 1978년에 만든 문서조판 프로그램을 TeX(“텍”으로 읽음)이라고 하고, 레슬리 램포트(Leslie Lamport) 교수가 만든 TeX 매크로 팩키지를 라텍이라고 한다. 텍과 라텍의 탄생 역사를 살펴보면 문서 논리 구조와 디자인 분리가 갖는 문서 저작 장점을 잘 나타내고 있다. 저수준(low-level) 언어로, 사용자가 페이지 레이아웃이나 글꼴 설정 등을 직접 제어할 수 있지만, 복잡한 포맷을 만들기 위해 사용자가 직접 매크로를 작성해야 한다는 점에서 일반 저작자가 사용하기에는 한계도 분명히 존재했다.

라텍은 1980년대에 레슬리 램포트에 의해 개발되었고, 텍을 기반으로 개발된다. 텍 복잡성을 단순화하여 더 사용자 친화적인 인터페이스를 제공한다. 따라서, 문서 구조와 내용에 더 집중할 수 있도록 고수준(high-level) 언어를 제공하며, 다양한 템플릿과 패키지가 제공되어 사용자가 쉽게 복잡한 레이아웃과 기능을 구현할 수 있다. 라텍 문서는 내부적으로 텍 엔진을 사용하여 조판된다. [Kim2017]

- 라텍: 편집 디자이너의 역할에 해당하는 작업을 수행
- 텍: 식자공의 역할에 해당하는 작업을 수행

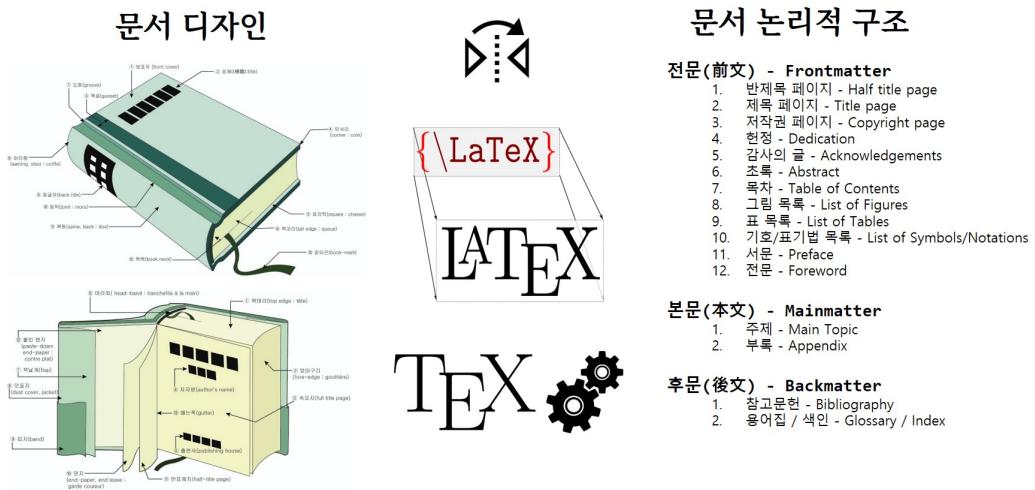


그림 6.1: 문서 디자인과 문서 논리 구조

라텍에서 적용하는 논리적 디자인의 가장 큰 이점은 글쓰기 능력의 향상에 있다. 라텍으로 글을 작성하면서 점차 라텍의 논리적 디자인 철학에 맞춰가게 되며, 이를 통해 글쓰기 실력이 개선된다.

컴퓨터를 이용한 전자조판의 주된 목적은 문서를 읽기 쉽게 구성하여 저자의 의도를 독자에게 명확하게 전달하는 것이다. 이를 위해선 문서의 디자인이 그 내용의 논리적 구조를 잘 반영해야 하며, 이를 달성하기 위해서는 디자이너가 문서의 논리적 구조를 충분히 이해해야 한다. 라텍 명령어 체계는 기본적으로 이러한 문서의 논리적 구조를 명시하는 데 중점을 둔다. 이렇게 설정된 명령어는 텍 엔진을 통해 조판되어 최종적으로 원하는 결과물을 생성한다.

라텍의 가장 큰 장점 중 하나는 확실히 자동화 기능이다. 수식, 참고문헌, 표, 그래프의 번호를 자동으로 관리해주며, 복잡한 문서 구조도 쉽게 다룰 수 있다. 특히, 국내외 학술지에 논문을 제출할 때에는 템플릿만 변경하면 고품질의 논문을 쉽게 생성할 수 있다. 반면, 워드 프로세서를 사용할 경우 이러한 작업은 수동으로 해야 하며, 이로 인해 시간을 낭비하게 되고 결과물이 덜 정교하게 나올 수 있다. 결과적으로, 라텍을 사용하면 글쓰기 본질에 더 집중할 수 있으며, 이를 통해 저자의 생각과 의견을 더욱 솔직하고 명확하게 독자에게 전달할 수 있다.

6.1 라텍 엔진

라텍을 실제 문서 작성에 활용하기 위해서는 적절한 텍 배포판을 선택해야 한다. 이에 대해 KTUG 한국 TeX 사용자 그룹은 *ko.TeX Live*를 추천하고 있으며, 그 이유는 다음과 같다. [Lee2017]

- 라텍 엔진으로 pdfLaTeX, XeLaTeX, LuaLaTeX 세 가지가 있으나, XeLaTeX은 유니코드 텍 엔진으로 동아시아 언어(CJK)를 잘 지원하고, 트루타입 및 오픈타입 폰트를 자유롭게 선택할 수 있어 인기가 높다.
- 운영체제에 따라 차이는 있지만, 실제 문서 작업에 라텍을 안정적으로 설치하고 사용하기 위해서는 TeX Live가 MikTeX보다 안정성과 사용자 지원 측면에서 우수하다.
- 텍 소스파일과 PDF 파일 간의 탐색 기능은 TexStudio IDE를 활용하여 쉽게 이용 할 수 있다.

과거에는 `latex+dvips+pspdf`를 사용한 전통적인 작업 흐름이 있었으며, 이 방식은 `pstricks`를 이용해 PDF 파일을 생성하는데 시간이 오래 걸리고, eps 파일만을 이미지로 사용할 수 있다는 한계가 있었다. 또 다른 방식인 `latex+dvipdfmx` 작업 흐름은 중간 단계로 `.dvi` 파일을 생성한 후 `dvipdfmx`를 이용해 PDF 파일을 만든다. 그러나 현재 가장 인기 있는 작업 흐름은 pdfLaTeX, XeLaTeX, LuaLaTeX를 활용한 방식으로, 이는 `.tex` 소스파일에서 바로 `.pdf` 파일을 생성할 수 있다는 장점이 있다.

논문 외의 문서를 작성할 경우에는 한 단계 더 나아가 `memoir` 또는 `oblivoir` 클래스를 활용하는 것이 효율적이다. 이 방법은 이미 검증된 문서 템플릿을 기반으로 빠르게 PDF 문서를 생성할 수 있는 방법이다.

6.2 라텍 구성요소

수학과 교수였던 도널드 크누스는 문서 조판 시스템을 개발하면서 수식 처리를 간편하게 할 수 있도록 하면서, 메타폰트(METAFONT)라는 글꼴 시스템도 개발하여 고품질 글자체를 구현할 수 있도록 했다. 이러한 기초 위에 레슬리 램포트는 텍을 더욱 사용하기 쉽게 만들기 위해 다양한 매크로를 라텍으로 묶어 공개했다. 그 결과 일반인도 라텍을 이용해 고품질의 문서를 쉽게 작성할 수 있게 되었다. 베클리 대학 마이클 스피액 교수는 수식을 간편하게 문서에 적용할 수 있도록 AMS-TeX라는 매크로를 개발하고, 이

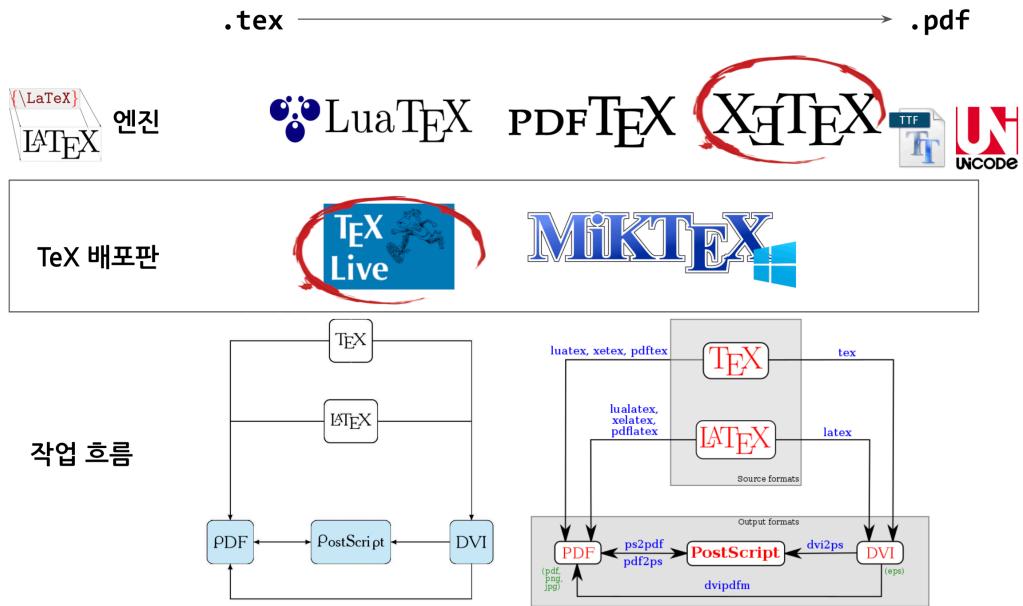


그림 6.2: 한글지원 라텍 엔진과 작업흐름

에 대한 설명을 “The Joy of TeX”이라는 책에서도 공개했다. [spivak1990joy]

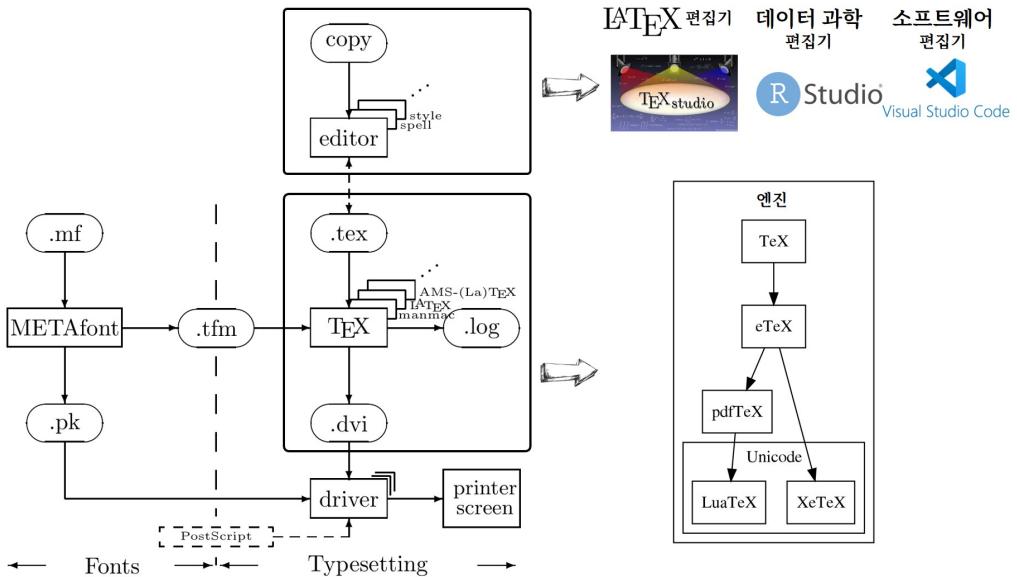
pdfTeX와 e-TeX이 결합하여 사실상 표준 TeX이 되었고, 이를 기반으로 XeTeX과 LuaTeX이 발전하고 있다. 특히, 한글 폰트 처리에 있어서 XeTeX(지텍으로 읽음)이 널리 사용되고 있다.

라텍 동작 원리는 Salomon의 도식화에 영감을 받아 Kees van der Laan이 1994년에 정리한 논문에서 상세하게 설명되어 있다. [van1994tex] 기본적으로 고품질의 전자 출판을 위해 글꼴과 텍이 필요하며, 이 두 요소가 결합하여 고품질의 출력물을 생성한다. 이러한 텍 엔진을 지원하는 다양한 패키지 중에서 AMS-(La)TeX, LATEX, manmac 등이 주목을 받고 있다.

문서의 다양한 구성 요소, 예를 들어 목차, 색인, 참고문헌, 그림과 표 등은 모듈화되어 관리되며, 글꼴 및 구조적인 스타일과 관련된 부분도 별도로 관리되고 있다. TeXWorks와 같은 위지윅(WYSIWYG)을 지원하는 편집기는 구문 강조, 맞춤법 검사 등을 통해 고품질의 라텍 문서 작성을 지원하고 있다.

라텍 문서 작성是为了 기본적인 구성요소는 다음과 같다.

1. **배포판:** 과거에는 다양한 배포판이 윈도우, 리눅스, 맥에서 사용되었지만, 현재는

그림 6.3: 편집기, 엔진, 글꼴을 중심으로 본 \LaTeX 내부

\TeX Live 가 대표적인 \LaTeX 작업 환경이다. Mik\TeX 도 있지만, 큰 차이는 없다.

2. **글꼴:** 초기에는 글꼴로 폰트 메트릭(tfm)과 메타폰트(MetaFont)가 주로 사용되었으나, 현재는 트루타입과 오픈타입이 널리 쓰인다.
 - 폰트 선택은 밥처럼 기본적이면서도 안정적인 것이 좋다. 특별한 상황에서는 그에 맞는 폰트를 선택하는 것이 중요하다.
 - 일반적으로 Serif와 San Serif 폰트를 한 쌍으로 사용하는 것이 균형감이 있다. 예를 들어, 나눔고딕과 나눔명조, KoPub돋움과 KoPub바탕, 힘초롬돋움과 힘초롬바탕 등이 있다.
 - 글자 크기는 10~12 포인트가 적절하며, 이는 장시간 동안의 읽기 피로를 줄이기 위함이다.
3. **그래픽:** 텍/라텍 자체는 그래픽을 주요 영역으로 보지 않는다. 과거에는 dvips와 EPS 형식을 주로 사용했으나, 현재는 .jpg, .png, .pdf 등도 잘 처리한다.
4. **문자와 입력:** 초기에는 ASCII 코드가 주를 이루었으나, 현재는 UTF-8을 권장하고 있다. 특히, \usepackage[utf8x]{inputenc}가 필요한 경우가 많다.
5. **색인과 참고문헌:** makeindex와 bibtex는 색인과 참고문헌 처리의 표준이다. 그러

나 유니코드 처리를 위해 `texindy`와 `biber`도 주목을 받고 있다.

6. 수식: 수학 표현과 관련해서는 미국 수학회가 발전시킨 **amsmath**가 표준이다.
7. 출력: 과거에는 `.dvi` 형식이 기본이었으나, 현재는 `.pdf`가 기본이다. 또한, 웹을 위한 다양한 `.html` 출력도 지원한다. 이에 따라, 웹이 책 형태의 문서보다 더 중요한 위치를 차지하게 될 것으로 예상된다.

```
% A flowchart of a TeX workflow
% Author: Stefan Kottwitz
% https://www.packtpub.com/hardware-and-creative/latex-cookbook
\documentclass[border=10pt]{standalone}
\begin{document}
\smartdiagram[flow diagram:horizontal]{Edit,
  \LaTeX, Bib\TeX/\ biber, make\texttt{-index}, \LaTeX}
\end{document}
```

라텍을 통한 PDF 파일 제작 과정은 몇 가지 주요 단계로 이루어진다. 라텍 전용 TexStudio 같은 통합개발환경(IDE), RStudio 같은 데이터 과학 IDE, 범용 소프트웨어 개발 VS코드 같은 IDE에서 텍스트 편집기를 열어 신규 라텍 파일을 생성하는데 파일 확장자는 `.tex`이다. 파일 시작 부분에 `\documentclass` 명령어를 사용하여 문서의 종류와 옵션을 설정하는데, 학술 논문을 작성한다면 `article` 클래스를, 책을 작성한다면 `book` 클래스 등을 선택한다.

`\usepackage` 명령어를 사용하여 필요한 패키지를 불러와서 특별한 기능이나 스타일을 추가한다. 예를 들어, 수식을 표현하기 위해 `amsmath` 패키지를, 그림을 삽입하기 위해 `graphicx` 패키지 등이 사용된다.

문서의 본문은 `\begin{document}`와 `\end{document}` 사이에 작성되고, 영역 안에서 텍스트, 표, 그림, 수식 등을 자유롭게 배치할 수 있다. 본문 작성 중에 참고문헌 서지관리를 위해 `BibTeX` `texindy`, `biber`를 사용하고, 색인 생성을 위해 `makeindex`를 사용한다.

본문 작성이 완료되면, 라텍 컴파일러를 실행하여 `.tex` 파일을 PDF로 변환한다. 영어는 `pdflatex`, 한글은 `xelatex`을 라텍 엔진으로 사용하고, 참고문헌과 색인이 포함된 경우 컴파일 과정이 수차례 이어진다.

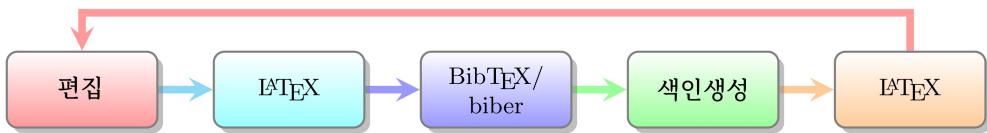


그림 6.4: 라텍 작업흐름

6.3 환경설정

PDF는 기술적, 학술적, 사업적으로 혁신적인 문서 양식이다. PDF가 개발된 것은 오랜 역사를 자랑하지만 아래한글, 워드퍼펙트, MS 워드, 리브레 워드 등에 가려 그 진가를 발휘하지 못했다. 하지만 이제 PDF는 문서 출판의 최고봉이라는데 이의를 제기하는 사람은 없을 것이다. PDF를 통해 아도비(Adobe) 회사는 엄청난 성공을 이어나가고 있고, 과학기술 저작에도 필수적인 파일 형태라 PDF에 대한 기본소양은 인공지능 시대를 살고 있는 시민으로 누구나 갖춰야하는 소양이다.

6.4 라텍 환경설정

라텍 설치와 사용은 처음에는 다소 복잡하게 느껴질 수 있다. 일반적으로 라텍 배포판 (예: TeX Live, MiKTeX)을 설치한 뒤, 텍스트 에디터(예: TeXShop, TeXworks)를 선택하여 작업환경을 구성한다. 설치가 완료되면, .tex 파일을 생성하고 라텍 명령을 통해 PDF를 컴파일한다.

환경변수 설정, 패키지 관리, 글꼴 설정 등 세부적인 환경설정이 필요하다. 특히 한글 문서를 처리할 경우에는 ko.TeX나 CTeX와 같은 한글을 지원하는 패키지를 별도로 설치하고 한글 글꼴도 설정해야 한다.

6.4.1 쿼토 설치

[Install Quarto](#) 웹사이트에 쿼토 설치 방법이 잘 설명되어 있다. 먼저 셸에서 curl로 쿼토 설치파일을 다운로드 받는다. gdebi-core를 설치하고 다운로드 받은 쿼토 파일을 gdebi 명령어로 설치하고 나서 quarto --version 명령어로 설치된 쿼토 버전을 확인한다.

```
$ sudo curl -L0 https://quarto.org/download/latest/quarto-linux-amd64.deb
$ sudo apt-get install gdebi-core
$ sudo gdebi quarto-linux-amd64.deb
$ quarto --version
1.3.450
```

6.4.2 라텍 설치

TinyTeX은 TeX Live를 기반으로 한 경량, 이식성 좋고, 사용하기 쉬운 라텍 배포판으로 라텍 설치와 관련된 일반적인 불편함을 해결하기 위해 만들어졌고, 특히 “이휘 쇄에 (Yihui Xie)”가 개발하여 R과 R 마크다운 통합이 잘 되어 있다.

다른 라텍 배포판들이 수 기가바이트(GB)의 크기를 갖지만, TinyTeX은 설치 크기가 작고 간단한 설치 과정을 제공으로 쉽지만 미려한 문서를 원하는 사용자들에게 인기가 많다. 특히, R과의 통합도 잘 이루어져 있어, 라텍 문서를 컴파일할 때 누락된 라텍 패키지도 자동으로 설치해주는 기능이 많은 사랑을 받고 있다.

보통 TexLive에 기반을 둔 tinytex를 사용하는데 한글을 사용하는 입장에서는 kotex를 설치하는 방법도 모색이 필요하다. 유니코드 문자를 처리하기 위해서 xelatex을 쿼토 LaTeX 기본엔진으로 지정하면 xelatex을 통해 CJK 문자가 포함된 PDF 파일 제작이 가능하다.

쿼토를 설치하면 quarto install tinytex 명령어로 TinyTeX을 쉽게 설치할 수 있고, 설치 정보는 quarto tools info tinytex으로 파악이 가능하다.

```
$ quarto install tinytex
Installing tinytex
[✓] Downloading TinyTeX v2023.09
[✓] Unzipping TinyTeX-v2023.09.tar.gz
[✓] Moving files
[✓] Verifying tlgpg support
[✓] Default Repository: https://mirrors.rit.edu/CTAN/systems/texlive/tlnet/
Installation successful
```

```
$ quarto tools info tinytex
{
  "name": "TinyTeX",
  "installed": true,
  "version": "v2023.09",
  "directory": "/home/statkclee/.TinyTeX",
  "bin-directory": "/home/statkclee/.TinyTeX/bin/x86_64-linux",
  "configuration": {
    "status": "ok"
  }
}
```

kotex 패키지 주요 장점 중 하나는 한글 관련 문제를 해결할 수 있는 다양한 라텍 패키지와 기본 한글 글꼴을 내장하고 있어 한글 문서 작성이 훨씬 편리하다. 하지만, `tinytex` 를 기본 라텍 엔진으로 사용할 경우, 누락된 패키지로 인해 추가 작업이 어려울 수 있다. 이러한 문제를 해결하기 위해 `tinytex::parse_install()` 함수를 사용하면, 오류가 발생한 텍스트를 붙여넣을 때 자동으로 필요한 패키지를 설치해 준다. 설치과정에서 오류 사항이 담긴 `.log` 파일을 `tinytex::parse_install()` 함수에 인자로 넣어주면, 자동으로 필요한 패키지를 설치해 준다.

```
# log 파일이 hello_world.log
tinytex::parse_install("hello_world.log")

tinytex::parse_install(
  text = "! LaTeX Error: File `titlepic.sty' not found."
)
```

i `tinytex` 설치

윈도우 환경에서 쿼토로 PDF 파일 생성할 때 다음과 같은 오류가 발생하여 로그에 기록될 경우 `tinytex` R 패키지를 설치하고 `tinytex::install_tinytex()` 명령어로 `tinytex`를 다시 설치한다.

```
running xelatex - 1

No TeX installation was detected.

Please run 'quarto install tinytex' to install TinyTeX.
If you prefer, you may install TeXLive or another TeX distribution.
```

6.5 헬로월드

영문으로 작성된 PDF 파일은 텍스트, 이미지, 표, 그래프, 다이어그램과 같은 다양한 문서 기본구성요소가 포함된다. 다양한 문서요소로 구성된 PDF 문서는 출력을 했을 때 또렷하고 가독성이 좋다. 하지만, 한글이 추가될 때 언어 호환성 문제가 발생하여 한글 텍스트가 깨지는 문제가 종종 발생된다. 문제해결방법 중 하나가 한글 글꼴을 명시적으로 설정하는 것이다.

한글 글꼴 설정을 통해 문제를 해결한 후에는 쿼토 마크다운(QMD) 형식으로 PDF 파일을 변환한다. 쿼토 마크다운을 사용하는 이유는 라텍보다 마크다운 문서의 작성이 더 생산적이며, 웹 출판을 중점적으로 고려한 결과, 다양한 형태와 아름다운 웹 출판물을 쉽게 제작할 수 있기 때문이다. 쿼토 마크다운을 사용하면 웹 출판을 기본으로 하면서, PDF 출판도 함께 진행할 수 있어 일거양득의 효과를 볼 수 있다.

6.5.1 영문 PDF

기본적인 라텍 문서 구조에 표, 수식, 다이어그램, 주석을 추가한 라텍 기본 문서를 작성한다.

```
\documentclass{article} % 문서 클래스 지정. article은 학술 논문, 보고서 등에 적합
↪ 하다.

\usepackage{tikz} % tikz 패키지 추가. 그래픽을 그리기 위해 필요하다.

\title{fathers of the atomic bomb} % 문서의 제목 설정
```

```
\author{Oppenheimer} % 문서의 저자 설정
\date{\today} % 문서의 날짜 설정. \today를 사용하여 오늘 날짜를 자동 입력한다.
```

```
\begin{document} % 문서의 본문 시작
```

```
\maketitle % 제목, 저자, 날짜를 출력
```

```
\section{Introduction} % "Introduction"이라는 섹션 생성
```

The name "Oppenheimer"\footnote{Oppenheimer is pronounced in English as /'ɒpnəmɪər/. Oppenheimer is a German-origin surname that is most famously associated with J. Robert Oppenheimer, the American physicist known as one of the "fathers of the atomic bomb."} % 본문과 각주

```
\section{Table} % "Table"이라는 섹션 생성
```

Below is a simple table.

```
\begin{center}
\begin{tabular}{|c|c|}
\hline
Header 1 & Header 2 \\
\hline
Row 1, Col 1 & Row 1, Col 2 \\
Row 2, Col 1 & Row 2, Col 2 \\
\hline
\end{tabular}
\end{center}
```

```
\section{Equation} % "Equation"이라는 섹션 생성
```

Here is a simple equation:

```
\begin{equation}
f(x) = x^2
\end{equation}
```

```
\end{equation} % 수식 환경 종료

\section{Diagram} % "Diagram"이라는 섹션 생성
Here is a simple diagram:
\begin{center}
\begin{tikzpicture} % tikz로 그림을 그리기 시작
    \draw[->] (0,0) -- (4,0); % x축 그리기
    \draw[->] (0,0) -- (0,4); % y축 그리기
    \draw (0,0) -- (3,3); % 대각선 그리기
\end{tikzpicture}
\end{center} % 그림 환경 종료

\end{document} % 문서의 본문 종료
```

RStudio로 라텍스를 이용해 PDF 문서를 생성할 때, 먼저 새로운 .tex 문서를 생성하고 표, 수식, 다이어그램, 주석이 포함된 라텍스 코드를 복사-붙여넣는다. RStudio의 Compile PDF 버튼을 클릭하여 컴파일하고, 완료되면 PDF 문서가 자동으로 열린다.

6.5.2 한글 PDF

영문 PDF를 한글로 번역하여 동일한 방식으로 PDF 파일을 만들기 위해 컴파일하게 되면 한글이 깨지거나 아무것도 없는 PDF 문서가 생성된다.

```
\documentclass{article}

\usepackage{tikz}

\title{원자폭탄의 아버지들}
\author{오펜하이默}
\date{\today}

\begin{document}
```

The screenshot shows the RStudio interface with the following components:

- Left Panel:** The code editor displays the LaTeX file `hello_world_english.tex`. The code includes sections for document class, packages, title, author, date, and content. It also contains a footnote about Oppenheimer, a table section, an equation section, and a diagram section.
- Top Bar:** Shows the title "pdf_eng - RStudio" and various menu items like File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Right Panel:**
 - Console:** Shows the command "Running xelatex.exe on hello_world_english.tex... completed" and the creation of a PDF file.
 - Terminal:** Shows the command "Running xelatex.exe on hello_world_english.tex... completed".
 - File Browser:** A file tree showing the project structure: `pdf_eng`, `pdf_eng.Rproj`, `quarto-linux-amd64.deb`, `hello_world_english.tex`, `hello_world_english.pdf`, `hello_world_english.log`, `hello_world_english.synctex.gz`, and `hello_world_english.synctex.gz`. The `hello_world_english.pdf` file is highlighted.

그림 6.5: 라텍 문서 컴파일

fatthers of the atomic bomb

Oppenheimer

September 13, 2023

1 Introduction

The name "Oppenheimer"¹ is a German-origin surname that is most famously associated with J. Robert Oppenheimer, the American physicist known as one of the "fathers of the atomic bomb."

2 Table

Below is a simple table.

Header 1	Header 2
Row 1, Col 1	Row 1, Col 2
Row 2, Col 1	Row 2, Col 2

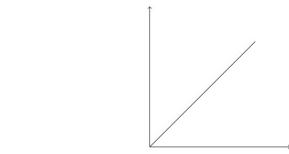
3 Equation

Here is a simple equation:

$$f(x) = x^2 \quad (1)$$

4 Diagram

Here is a simple diagram:



¹Oppenheimer is pronounced in English as /'ɒpənhaɪmər/

1

그림 6.6: 라텍 헬로월드 기본 문서

```
\maketitle
```

```
\section{소개}
```

"오펜하이머"라는 이름\footnote{오펜하이머는 영어로 /'ɒpənhaɪmər/로 발음된다}은 독

- ↪ 일 기원의 성씨로 가장 유명한 사람은 미국 물리학자 J. 로버트 오펜하이머이다. 그
- ↪ 는 "원자폭탄의 아버지"로 알려져 있다.

```
\section{표}
```

아래는 간단한 표이다.

```
\begin{center}
```

```
\begin{tabular}{|c|c|}
```

```
\hline
```

```
헤더 1 & 헤더 2 \\

\hline

행 1, 열 1 & 행 1, 열 2 \\

행 2, 열 1 & 행 2, 열 2 \\

\hline

\end{tabular}

\end{center}
```

\section{수식}

다음은 간단한 수식이다:

```
\begin{equation}

f(x) = x^2

\end{equation}
```

\section{다이어그램}

아래는 간단한 다이어그램이다:

```
\begin{center}

\begin{tikzpicture}

\draw[->] (0,0) -- (4,0);
\draw[->] (0,0) -- (0,4);
\draw (0,0) -- (3,3);

\end{tikzpicture}

\end{center}
```

```
\end{document}
```

한글이 PDF 문서에 표현되지 않는 문제는 글꼴을 .tex 파일에 지정함으로써 해결된다. 먼저 전제조건으로 한글 글꼴(예를 들어, 나눔고딕 글꼴)이 설치되어 있고 이를 라텍에서 컴파일하는 방식으로 작업한다.

```
\documentclass{article}
```

September 13, 2023

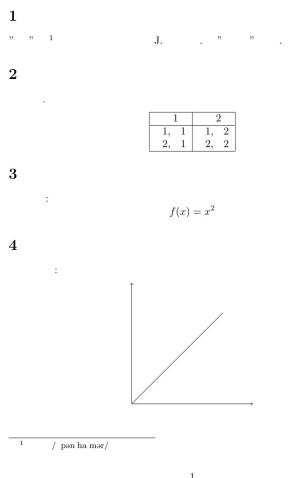


그림 6.7: LaTeX에서 한글이 깨진 PDF 문서

```
\usepackage{tikz}
\usepackage{fontspec}
\setmainfont{NanumGothic} % NanumGothic 글꼴이 설치되어야 함
\usepackage{tipa} % 발음기호(IPA symbols)

\title{원자폭탄의 아버지들}
\author{오픈하이머}
\date{\today}

\begin{document}

\maketitle

\section{소개}
```

"오픈하이머"라는 이름\footnote{오픈하이머는 영어로 \textipa{/ə'pənhaɪmər/} ,
↳ \textipa{/ʌ'pən'pənəl'haɪmər/} 로 발음된다}은 독일 기원의 성씨로 가장 유명한 사람은 미국 물리학자 J. 로버트 오픈하이머이다. 그는 "원자폭탄의 아버지"로 알려져 있다.

\section{표}

아래는 간단한 표이다.

```
\begin{center}
\begin{tabular}{|c|c|}
\hline
헤더 1 & 헤더 2 \\
\hline
행 1, 열 1 & 행 1, 열 2 \\
행 2, 열 1 & 행 2, 열 2 \\
\hline
\end{tabular}
\end{center}
```

\section{수식}

다음은 간단한 수식이다:

```
\begin{equation}
f(x) = x^2
\end{equation}
```

\section{다이어그램}

아래는 간단한 다이어그램이다:

```
\begin{center}
\begin{tikzpicture}
\draw[->] (0,0) -- (4,0);
\draw[->] (0,0) -- (0,4);
\end{tikzpicture}

```

```
\draw (0,0) -- (3,3);  
\end{tikzpicture}  
\end{center}  
  
\end{document}
```

원자폭탄의 아버지들

오펜하이머

September 13, 2023

1 소개

"오펜하이머"라는 이름은 독일 기원의 성씨로 가장 유명한 사람은 미국 물리학자 J. 로버트 오펜하이머이다. 그는 "원자폭탄의 아버지"로 알려져 있다.

2 표

아래는 간단한 표이다.

헤더 1	헤더 2
행 1, 열 1	행 1, 열 2
행 2, 열 1	행 2, 열 2

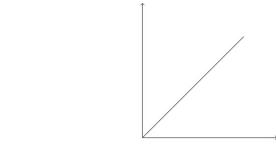
3 수식

다음은 간단한 수식이다:

$$f(x) = x^2 \quad (1)$$

4 다이어그램

아래는 간단한 다이어그램이다:



¹ 오펜하이머는 영어로 /pəhæmər/ , /pənhaɪmər/ 로 발음된다.

1

그림 6.8: L^AT_EX에서 한글문제를 해결한 PDF 문서

6.5.3 쿼토 PDF

PDF 문서 저작을 위해 쿼토 .qmd 파일에서 라텍 편집사항을 기술하려는 경우, 앞서 동일한 문서를 라텍 편집기가 아닌 쿼토 마크다운 파일로 작성한다.

```
---  
title: "원자폭탄의 아버지들"  
author: "오펜하이머"  
date: "2023-10-09"
```

```
format:
  pdf:
    documentclass: article
    geometry:
      - paperheight=257mm
      - paperwidth=188mm
      - top=25mm
      - headsep=10mm
      - bottom=30mm
      - footskip=15mm
      - left=25mm
      - right=25mm
      - centering
    latex_engine: xelatex
    keep_tex: false
    include-in-header:
      text: |
        \usepackage{tikz}
        \usepackage{fontspec}
        \usepackage{tipa}
        \usepackage{setspace}
        \onehalfspacing

    mainfont: "MaruBuri-Regular"
    sansfont: "NanumSquareR"
    monofont: "D2Codingligature"
    editor_options:
      chunk_output_type: console
    ---
# 소개
```

”오펜하이머”라는 이름[¹]은 독일 기원의 성씨로 가장 유명한 사람은 미국 물리학자 J. 로버트 오펜하이머이다. 그는 ”원자폭탄의 아버지”로 알려져 있다.

[¹]: 오펜하이머는 영어로 \textipa{/ˈɒfənhaɪmər/} 로 발음된다.

표

아래는 간단한 표이다.

```
::: {.cell}
::: {.cell-output-display}
```

헤더 1	헤더 2
행 1, 열 1	행 2, 열 1
행 1, 열 2	행 2, 열 2

:::

:::

수식

다음은 간단한 수식이다:

\$\$

```
f(x) = x^2
$$
```

```
# 다이어그램
```

아래는 간단한 다이어그램이다:

```
::: {.cell layout-align="center"}
::: {.cell-output-display}
{fig-align='center'}
:::
:::
```

이 `*.qmd` 파일은 Quarto를 사용하여 PDF로 렌더링할 수 있다. LaTeX 패키지와 명령어는

- ↪ YAML 메타데이터의 `header-includes` 섹션에 추가될 수 있다. 그러나 특별한 LaTeX 명령어나 명령어나 환경이 필요한 경우, 이를 수동으로 `*.qmd` 파일에 적용해야 할 수도 있다.

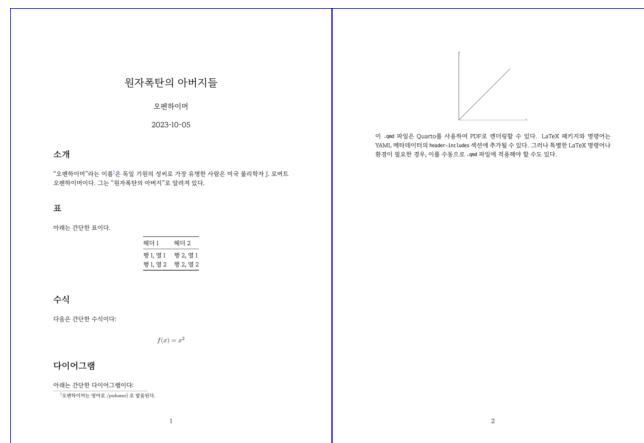
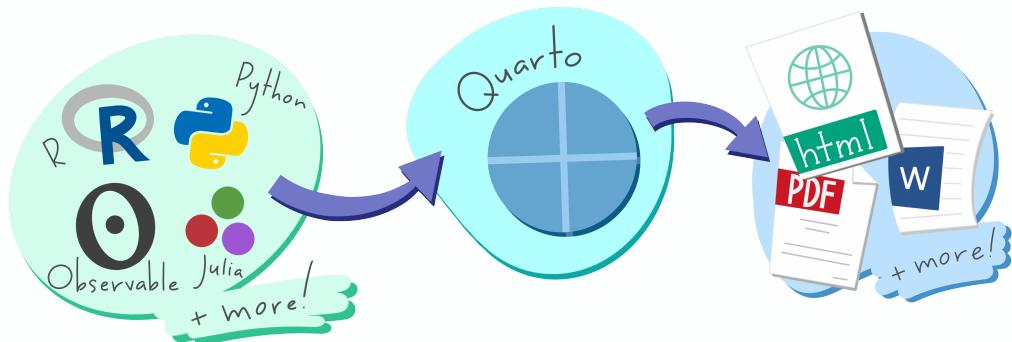


그림 6.9: 쿼토 마크다운으로 PDF 파일 생성

제 7 장

쿼토(Quarto)



쿼토(Quarto)는 다양한 프로그래밍 언어를 지원하는 오픈소스 출판 시스템으로, Pandoc에 기반을 두고 있다. 쿼토는 R, 파이썬, 줄리아, 자바스크립트(Observable JS) 등의 프로그래밍 언어를 지원하며, 사용자는 싱글 소싱(Single Sourcing)을 통해 신속하게 다양한 출판 저작물을 생성할 수 있다. “차세대 R마크다운(Next-Generation R Markdown)”이라는 별명을 가진 쿼토는 약 10년 동안의 R 마크다운의 성공과 실패 경험을 바탕으로 새로운 아키텍처로 재탄생했다. 개발 및 유지보수 팀이 잘 구성되어 있어, 프로젝트의 지속 가능성과 신뢰성이 높고, 무료로 제공되므로 누구나 쉽게 접근할 수 있을 뿐만 아니라 누구나 오픈소스 프로젝트에 기여할 수 있다. [hyde2021single]

디지털 글쓰기와 출판에 대한 고민은 콘텐츠, 디자인, 형식의 세 가지로 나뉜다.

- **콘텐츠(Content):** 저작물의 실질적인 내용으로 글쓰기 핵심이며, 정보 정확성, 완

전성, 통찰력을 제공해야 한다.

- **디자인(Design)**: 저작물의 외양이나 'Look and Feel'을 의미하며 사용자 경험에 큰 영향을 미치며, 정보를 효과적으로 전달하는 데 중요한 역할을 한다.
- **형식(Format)**: 저작물의 최종 형태나 산출물이 담기는 매체(디지털, 인쇄 등)를 의미하며, 어떻게 배포되고 소비될 것인지 결정한다.

쿼토는 세 가지 요소를 충족시키기 위한 다양한 기능을 제공한다. 팬독(Pandoc)과 마크다운을 활용해 수식, 인용, 서지 관리, 콜아웃, 고급 레이아웃 기능 등을 고품질 논문, 보고서, 발표 슬라이드(PPT), 웹사이트, 블로그, 전자책 등 다양한 형태 출판 산출물로 저작하여 인쇄매체 뿐만 아니라 HTML, PDF, MS 워드, ePub 등 다양한 디지털 매체를 통해 독자에게 전달할 수 있다. 특히, 쿼토는 프로그래밍 언어(R, 파이썬 등)와 문서 저작, 출판 산출물 사이 핵심 연결고리로 동작하여 버전 제어, 협업, 재현 가능한 과학기술 문서나 비즈니스에서 자동화된 생산성 높은 문서 저작에 큰 장점을 제공한다.

프로그래밍(계산)	문서 저작	출력물
Python, R, Julia, Observable JS	Pandoc, 마크다운, LaTeX	문서, 웹사이트, PPT, (전자) 책, 블로그 등

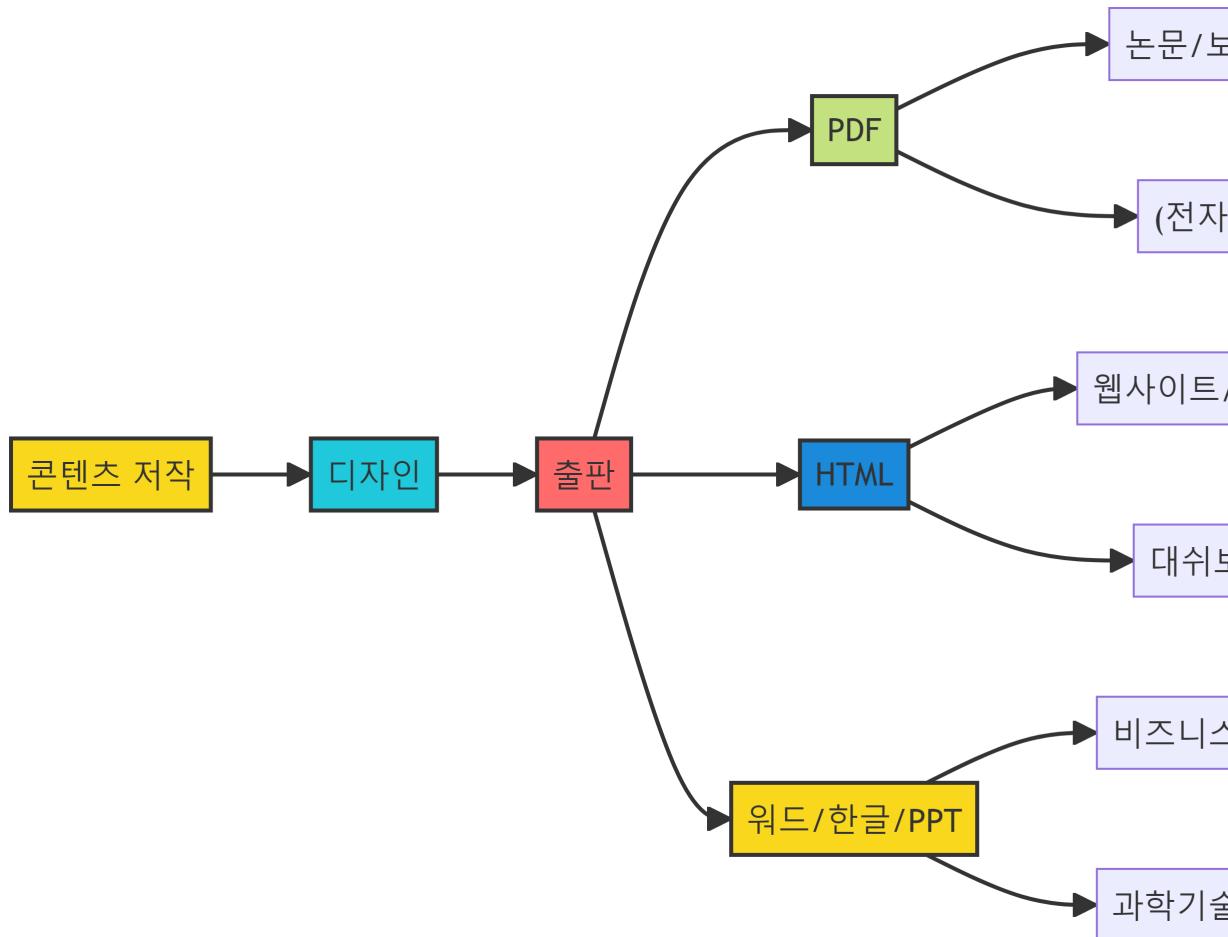
7.1 싱글 소싱 출판저작

데이터 과학을 비롯한 과학기술 분야 출판저작이 다른 분야와 약간 다를 수 있지만, 출판의 기본적인 접근 방식은 대체로 유사하다고 볼 수 있다. 특히, 싱글 소싱(Single Sourcing)¹ 방식을 통해 콘텐츠 저작부터 디자인, 협업, 검수, 최종 출판물 관리까지 일원화하고 자동화한다면, 중복작업과 낭비를 줄이는 동시에 재현 가능성을 높일 수 있어 과학기술 출판저작물에 있어 가장 이상적인 방법으로 평가받고 있다.

싱글 소싱 저작 방식은 데이터, 코드, 수식, 다이어그램, 텍스트 등 다양한 저작물 구성 요소를 하나의 소스에서 관리함으로써, 수정이나 업데이트가 필요할 때 일관성을 유지하면서 효율적으로 저작작업을 수행할 수 있어 연구 결과의 정확성과 신뢰성을 높이기 위한 재현가능한 글쓰기를 중시하는 과학기술 분야에서 특히 강조되고 있으며, 싱

¹“싱글 소싱(Single Sourcing)”은 하나의 원본 콘텐츠를 여러 형식이나 채널에 적용할 수 있도록 하는 문서 작성 및 관리 방법이다. 콘텐츠를 한 번만 작성하고 다양한 출력 형식(예: PDF, 웹 페이지, PPT, 모바일 앱 등)으로 변환하게 되면, 콘텐츠 일관성 유지 및 업데이트와 관리가 훨씬 효율적이다.

글 소싱을 통한 자동화 생산성을 중시하는 비즈니스 글쓰기에서도 점차 글로벌 표준으로 자리를 잡아가고 있다.



7.2 작업흐름

기존 R .Rmd 파일과 파이썬 .ipynb 파일을 .qmd 파일로 통일시킨 것은 쿼토의 주요 특징 중 하나이다. 이러한 통합은 다양한 프로그래밍 언어를 사용하는 복잡한 작업 흐름을 단순화시킬 수 있고 협업을 촉진한다. 저작자는 마크다운을 사용하여 콘텐츠를 작성하고, 프로그래밍 코드는 R, 파이썬, 자바스크립트, 줄리아 등 프로그래밍 언어로 작성한다. 텍스트와 프로그래밍 코드는 팬독(pandoc)을 통해 마크다운 변환이 자동으로 이루어지기 때문에 후속 작업에 대해 걱정할 필요가 없다. 싱글 소싱 개념이 팬독을 통해 자

동화되어 원하는 저작 결과물을 효율적이고 빠르게 얻을 수 있어 저작작업 생산성을 크게 향상시키는 장점이 있다.

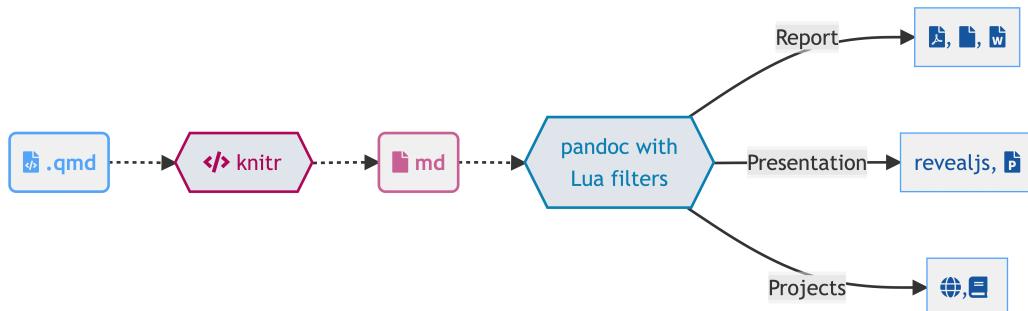


그림 7.1: 쿼토 - R

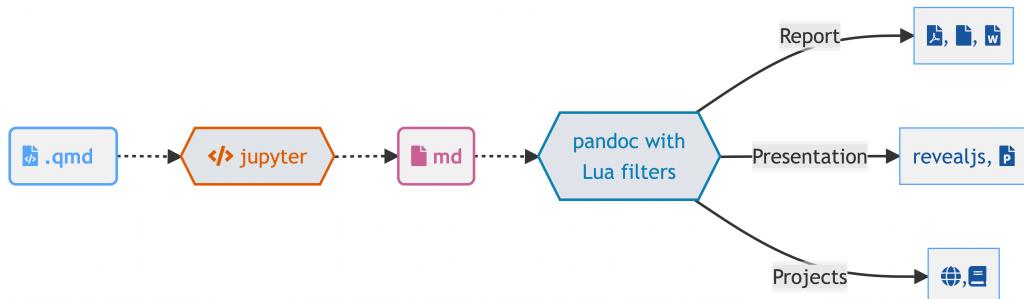


그림 7.2: 쿼토 - 파이썬

그림 7.3: 쿼토 R, 파이썬 싱글소싱

7.3 쿼토 해부

쿼토는 데이터 과학 문서 작성의 새로운 패러다임을 제시하며, R마크다운의 후속임을 자처하고 있다. 10년 이상의 knitr 경험을 바탕으로 개발되었고, 최근에는 RStudio Visual 편집 기능을 통해 위지위그(What You See Is What You Get) 패러다임도 적극 수용함으로써 저작자는 복잡한 코드 작성 없이도 직관적으로 문서를 작성할 수 있게 되었다.

쿼토는 다양한 통합개발환경(IDE)과 호환성을 지원하고 있다. R 사용자는 RStudio IDE를, 파이썬 경험이 풍부한 사용자는 파이참, 범용 소프트웨어 개발 경험을 가지신 분은 VS코드, 이맥스/VIM 등과 통합하여 사용할 수 있고, 쿼토 CLI를 통해 IDE에서 쉽게 저

작물을 다양한 형태로 출판할 수 있다.

쿼토는 다양한 프로그래밍 언어를 지원하여 문서 내에 계산(Computation) 기능을 쉽게 통합할 수 있는 혁신하는 얼마 안되는 문학적 프로그래밍 저작도구다. R, 파이썬, SQL, 자바스크립트 등을 활용해 그래프, 표, 다이어그램, 인터랙티브 산출물을 문서에 반영하여 한층 진화된 디지털 문서 저작을 지원한다.

쿼토는 데이터 과학자, 개발자, 과학기술 연구원, 사무노동자 등 다양한 분야의 종사자 분들로부터 챗GPT 인공지능 시대 자동화를 통한 사무업무 생산성 도구로 자리매김하고 있다.



그림 7.4: 쿼토 작업흐름

쿼토 저작은 메타데이터(전문, front matter), 텍스트, 코드라는 세 가지 주요 구성요소로 이루어진다. 쿼토는 문학적 프로그래밍과 자동화 패러다임을 적극 반영하여 데이터 과학 프로그래밍과 과학기술 문서 작성은 더 효율적이고 일관된 방식으로 저작하고 출판할 수 있도록 지원한다.

- **메타데이터 (YAML):** YAML은 “YAML Ain’t Markup Language”의 약자로, 데이터를 구조화하는 언어이다. 쿼토에서 이를 활용하여 문서 전문(Front Matter)을 구성한다. 문서 상단에 위치하며 문서 제목, 작성자, 날짜, 문서 형식 등을 포함하며 문서의 전반적인 설정과 구성을 담당한다.
- **텍스트 (마크다운):** 마크다운은 2004년에 아론 스워츠와 존 그루버에 의해 개발된

언어로, HTML과 같은 복잡한 마크업 언어 대신 사용되는 간단한 형식의 문서 작성 도구다. 복잡한 디자인보다 콘텐츠 구조와 의미에 중점을 두기 때문에, 문서 저작자는 외양보다 내용에 더 집중할 수 있다.

- **코드 (knitr, jupyter):** knitr는 R 코드를 문서에 통합할 수 있는 패키지, jupyter는 파이썬 사용자에게 인기 있는 패키지이다. 쿼토에서는 이 두 패키지를 활용하여 작성된 코드를 문서에 포함시킬 수 있다. 데이터 분석, 통계값, 그래프, 다이어그램, 지도 등 다양한 코딩 결과물을 문서에 반영시킬 수 있다.

```

1 ---  

2 title: "ggplot2 시연"  

3 date: "5/22/2021"  

4 format: html  

5 ---  

6  

7 ## 공기질  

8  

9 기온과 오존수준 두 변수간 관계가 있다.  

10  

11 ``{r}  

12 #| label: fig-airquality  

13 library(ggplot2)  

14 ggplot(airquality, aes(Temp, Ozone)) +  

15   geom_point() +  

16   geom_smooth(method = "loess")  

17 )  

18 ```

```

그림 7.5: 쿼토 문서 구성요소

7.3.1 YAML

YAML(발음은 야블)은 R 마크다운과 쿼토에서 주로 사용되는 경량언어다. YAML은 문서 전문(front matter) 부분에 위치해 코드와 텍스트로 구성된 문서 본문보다 먼저 위치하며, 문서 메타데이터를 설정하는 역할을 수행한다. 쿼토에서 YAML을 이용해 문서 제목, 작성자, 날짜, 문서 형식, 레이아웃 등을 정의한다. YAML은 키와 값이 콜론(:)으로 구분되는 키값(key-value) 쌍을 사용해 각종 문서 설정정보를 표현하여 전반적인 문서 설정과 구성을 쉽게 관리한다.

키값 쌍

```
---
```

```
key: value
```

```
---
```

문서출력형식 PDF 지정

```
---
```

```
format: pdf
```

```
---
```

YAML을 이용한 문서 전문 작성은 가독성이 뛰어나 읽고 이해하기 쉽다는 장점이 있다. 특히 복잡한 문서 구조나 설정을 지정할 때 유용하며, 중첩된 선택옵션 같은 복잡한 구조도 들여쓰기를 통해 명확하게 표현할 수 있다. YAML은 재사용성이 뛰어나서 같은 설정 값을 여러 곳에서 사용해야 할 경우, 하나의 YAML 파일만 수정하면 모든 곳에 적용될 수 있어 시간과 노력을 크게 절약할 수 있다. 텍스트 기반인 만큼 Git과 같은 버전 제어 시스템과도 잘 통합되어, 여러 저자가 공동으로 작업할 때 진가를 발휘하지만, 들여쓰기에 민감하여 실수로 공백을 잘못 넣었을 경우 오류가 발생할 수 있어 주의가 필요하다.

YAML은 기본적으로 KEY: Value 형태로 구성되어 있지만, 이러한 단순한 구조 덕분에 매우 편리하고 유용하다. YAML을 통한 문서 저작을 경험하게 되면 YAML의 편리성과 유용함을 더 깊게 이해하게 된다.

CLI PDF 지정

```
$ quarto render document.qmd --to pdf
```

YAML 문서형식 PDF

```
---
```

```
format: pdf
```

```
--
```

quarto render 명령을 사용하여 document.qmd 파일을 HTML 형식으로 제작하는 예시에서 CLI 옵션 -M code fold:true을 사용하여 코드 섹션을 접을 수 있게 만들었다. 이러한 설정을 YAML로 바꾸면, 설정 파일 내에 format 섹션을 생성하고 그 안에 html을 정의한다. 그 후, html 포맷 설정에 목차 생성(toc: true)과 코드 접기 기능(code-fold: true)을 추가한다.

CLI 옵션을 직접 입력하는 것과 비교하여 YAML 파일에 설정을 담는 것이 훨씬 깔끔하고 우아하다. CLI에서 많은 선택옵션을 직접 입력하면 명령어가 길어져 가독성이 떨어지고, 나중에 어떤 옵션을 사용했는지 확인하기 어렵다. 하지만 YAML 파일을 사용하면, 모든 설정을 한 곳에서 명확하게 관리할 수 있어 유지보수가 쉽고, 공동 저작자들과 공유와 협업에도 유리하다.

CLI 코드접기

```
$ quarto render document.qmd --to html -M code fold:true
```

YAML 코드접기

```
---
```

```
format:
```

```
  html:
```

```
    toc: true
```

```
    code-fold: true
```

```
--
```

RStudio나 VSCode와 같은 통합개발환경(IDE)에서 제공하는 탭-자동완성(Tab)

Completion) 기능은 YAML 전문코드를 작성할 때 매우 유용하다. 이 기능을 사용하면, 저작자는 첫 글자나 몇몇 글자를 타이핑한 후에 템 키를 눌러 가능한 모든 명령어, 변수, 함수 이름 등을 볼 수 있다. 이로 인해 오타의 가능성이 줄어들고 문서 저작 속도가 향상된다. 특히 RStudio에서는 **Ctrl + space** 단축키를 사용하여 YAML 전문코드를 작성 할 때 가능한 키값을 선택할 수 있는 목록을 제공한다. 이러한 기능은 문서 저작 과정 을 효율적으로 만들어 준다.

The screenshot shows the RStudio Source Editor interface with a dark theme. A file named 'writing_yaml.qmd' is open. The code editor displays the following YAML snippet:

```

1 ---
2 title: "챗GPT 디지털 글쓰기"
3 format: |
4
5
6

```

At the bottom of the code editor, the status bar shows '3:9' and '# 챗GPT 디지털 글쓰기'. The top menu bar includes 'Source', 'Visual', and 'Outline' tabs. The toolbar contains icons for back, forward, render, and run.

그림 7.6: YAML 키 설정

The screenshot shows the RStudio Source Editor interface with a dark theme. A file named 'writing_yaml.qmd' is open. The code editor displays the same YAML snippet as in Figure 7.6. The cursor is at the end of the line 'format: |'. A completion dropdown menu is open, listing two options: 'html' and 'chunkedhtml'. The status bar at the bottom shows '3:13' and '# 챗GPT 디지털 글쓰기'. The top menu bar includes 'Source', 'Visual', and 'Outline' tabs. The toolbar contains icons for back, forward, render, and run.

그림 7.7: 템 자동완성 선택확정

그림 7.8: RStudio YAML 자동완성

7.4 R 마크다운과 쿼토

쿼토는 “차세대 R 마크다운”이라는 별명에 걸맞게 R 마크다운의 다양한 패키지와 기능 을 지원하고 있다. 그러나 쿼토의 개발 기간이 R 마크다운보다 상대적으로 짧기 때문에 아직은 R 마크다운의 모든 기능을 지원하지 않는다. 그럼에도 불구하고, 쿼토는 지속적 으로 지원 범위를 확장하고 있으며, 고유한 기능도 추가하고 있다. 이러한 점들은 쿼토 가 R 마크다운 생태계에 어떤 새로운 가치를 더하고 있는지를 잘 보여준다. 다음 표 ?? 에서 쿼토와 R 마크다운을 다양한 문서 기능별로 비교하고 있다.

기능	R 마크다운	쿼토(Quarto)
파일형식	html_document / pdf_document / word_document	html / pdf / docx
비머(Beamer)	beamer_presentation	beamer
파워포인트(PPT)	powerpoint_presentation	pptx
웹 슬라이드	xaringan / ioslides / revealjs	revealjs
고급 레이아웃	tufte / distill	Quarto Article Layout
상호 참조	html_document2 / pdf_document2 / word_document2	Quarto Crossrefs
웹사이트/블로그	blogdown / distill	Quarto Websites / Quarto Blogs
책	bookdown	Quarto Books
인터랙티브 문서	Shiny Documents	Quarto Interactive Documents
페이지 HTML	pagedown	출시 예정
학술 논문	rticles	Quarto Journal Articles
대쉬보드	flexdashboard	출시 예정
인터랙티브 자습서	learnr	계획 없음

표 7.2: R 마크다운과 쿼토 비교

제 II 편

글쓰기

제 8 장

심각한 현재 상황

초기 현생인류는 손가락, 물감, 동굴 벽화를 통해 자신의 생각과 감정을 표현했다. 이러한 원시적인 도구들을 활용해 사람들은 상상의 세계를 현실로 구현할 수 있었다. 시간이 흐르면서 손가락과 물감은 연필로, 동굴 벽화는 종이로 대체되었고 이렇게 진화된 도구들 덕분에 저자들은 이제 자신의 의도와 생각을 종이 위에 문자와 그림으로 자유롭게 표현할 수 있게 되었다.

처음으로 인쇄기가 등장했을 때 자신의 의도를 문자와 그림으로 자유롭게 표현하는 자유는 크게 변화하지 않았다. 목판을 이용한 인쇄 방식이었지만, 작가는 여전히 원하는 내용을 원하는 위치에 놓을 수 있었고, 1370년경 한국에서 가동활자가 발명되고, 1440년경 유럽에서 구텐베르그가 금속활자가 대중화되면서 인쇄술이 급속히 확산되었다. 이로 인해 수기로 글을 쓰는 필경사에 대한 존경은 점차 감소하기 시작했다.

활자를 사용하면 인쇄업자가 목판을 제작하는 것보다 훨씬 신속하게 인쇄면을 설정할 수 있는 장점이 있었지만 유연성이 떨어졌다. 이유는 조판업자는 일정한 크기 글자를 일렬로 배치해야 했기 때문이다. 필경사도 원하는 곳 어디에나 글씨를 써 넣든 그림을 그려 넣던 자유로웠던 시절은 점차 사라지기 시작했다. 도표는 여전히 가능했지만, 글자 비용이 낮아지면서 상대적으로 몇 배나 더 비쌌다.

1860년대에 발명된 타자기는 중산층 수백만 명에게 '인쇄'를 가능하게 했다. 기계식, 전기식, 전자식 컴퓨터는 모두 타자기 기술을 재활용하여 인쇄 출력물을 생성했다. 1950년

대 등장한 펜 플로터¹는 라인 프린터를 대체하기에는 너무 느리고 비쌌다. 더 큰 문제는 두 기술 모두 완벽하게 작동하지 않았다는 것이다. 아스키 예술로 그림을 표현하거나, 펜플로터로 문자를 작성하는 것은 가능했지만, 어느 것도 특별히 매력적이라고 할 수 없었다.

8.1 세가지 패러다임

문자전용 도구와 그림을 위한 도구 사이 간격을 보여주는 한가지 흔적이 문자와 그림을 제어하기 위한 별도 언어개발에서 찾아볼 수 있다. 플로터는 일반적으로 제도 언어 (*drawing language*)로 제어된다. 다음 예를 보면 이해가 쉽다.

“펜을 위로, (x, y) 위치로 이동, 펜을 아래로, 다시 해당 위치만큼 이동한다”

```
PU;  
PA200,150;  
PD;  
PA250,250;
```

반면에, 라인 프린터는 조판 언어 (*Typesetting language*)를 사용하여 제어된다. 이 언어를 통해 저자는 컴퓨터에게 “두 번째 큰 제목을 설정하라”나 “특정 단어를 이탤릭체로 표시하라”와 같은 지시를 할 수 있다. 이러한 지시를 받은 컴퓨터는 단어 위치와 형식을 자동으로 결정한다. 조판 언어를 통해 문자 배치와 스타일을 정교하게 제어할 수 있게 되었다.

```
.t2 Section Heading  
  
Empty lines separate  
.it paragraphs  
and lines starting with '.' are commands.
```

이 기간 동안, 문서 외양이 아닌 콘텐츠(*content*)에 중점을 둔 세 번째 유형의 언어가 등장했다. 의사나 변호사들은 환자 진료기록이나 판례를 효율적으로 검색하고자 했으나, 1970년 당시 컴퓨터는 자연어 처리 능력이 턱없이 부족했다. 당시 최고의 기술력을 자

¹플로터(plotter): 그래프나 도형, CAD, 도면 등을 출력하기 위한 대형 출력장치다.

랑하던 IBM과 같은 컴퓨터 기업들이 마크업 언어를 개발했다. 마크업 언어를 통해 사람들은 문서 의미, 즉 시맨틱(*semantic*)을 명시적으로 표현할 수 있게 되었다. 마크업 언어는 문서 내용과 구조를 명확하게 기술하게 되면서, 정보 검색과 데이터 관리에 있어 새로운 가능성을 열었다.

```
<person>Derstmann</person> still questions the importance of <chemical>methane</
  ↵ chemical> release
in <event>the Fukuyama disaster</event>.
```

8.2 패러다임의 충돌

이 세 가지 패러다임은 1970년대 레이저 프린터 발명 이후에 충돌했고, 그 긴장감은 1980년대 고해상도 컴퓨터 화면과 1990년대 월드 와이드 웹에 의해 더욱 확대되었다. 한편으로 대부분의 사람들은 단순히 글을 저작하는 것이 목표다. 예를 들어, 이 단어는 여기에, 저 단어는 저기에 놓거나, 일부 단어는 녹색으로, 다른 단어는 이탈릭체로 만들고 싶어하는 것이 전부다. 우리나라에서 아래한글, 전세계적으로 맥사용자는 맥라이트, 윈도우 사용자는 마이크로소프트 워드, 리눅스 사용자는 리브레오피스 라이터(LibreOffice Writer)나 오픈오피스 라이터(OpenOffice Writer)와 같은 위즈윅(WYSIWIG) 편집기가 이러한 저자들의 욕구를 충족시켰지만, 이런 방식으로 만들어진 문서는 두가지 결점을 갖고 있다:

1. 융통성이 없다(*rigid*). 누군가 수작업으로 배치를 바꾸고 나서, 페이지 크기를 변경하면, 수고로운 작업을 다시 해야 한다.
2. 불분명하다. 컴퓨터에 무언가 이택릭으로 표현하도록 지시하면, 해당 문구가 책제 목인지, 혹은 새로운 용어를 정의하는지 분간할 수 없다.

즉, 아래한글과 워드 같은 위치윅 편집기로 생성된 문서는 종종 특정 플랫폼이나 소프트웨어에 종속되어 다른 시스템에서 문서를 편집하거나 여는 것조차 불가능하다. 이유는 위치윅 편집기가 주로 시각적인 표현에 중점을 두다보니 아래한글, 워드 문서파일 자체적으로 복잡한 내부 구조를 가지고 있고 문서 내용과 구조를 명확하게 구분하지 않아서 문서 내용을 다른 형식으로 변환하거나 자동화하는 것이 어렵다.

조판 언어와 마크업 언어는 앞서 언급한 두 가지 문제를 해법을 제시한다. 저자는 텍스트나 그림 외관과 페이지 내 위치에 대한 직접적인 지시를 내리는 대신, 그것들이 어떤

유형인지—예를 들어 제목이나 새로운 용어인지—컴퓨터에게 알려준다. 컴퓨터는 입력을 받은 후에 그것들의 외관과 위치를 결정하여 표현한다. 이러한 방식으로 의미(시맨틱)와 외관을 분리하면, 저자는 “모든 두 번째 제목을 16포인트 나눔고딕체로 왼쪽 정렬하라”와 같이 지시를 내리면, 스타일도 일관되게 쉽고 빠르게 변경시킬 수 있다.

하지만, 이런 접근법도 결점은 있다:

1. 컴퓨터는 텍스트를 이해하지 못하기 때문에 항상 인간과 같은 방식으로 텍스트를 배치하지는 않는다. 따라서 사람들은 나중에 문서 유연성이 떨어지더라도 직접 변경하고 싶어 하는 경우가 많다. 예를 들어, 논문을 작성할 때 컴퓨터가 자동으로 생성한 목차나 참고문헌 위치가 저자 의도와 맞지 않을 수 있다. 이런 경우, 저자가 수동으로 목차를 조정하면, 나중에 문서 다른 부분을 수정할 때 목차도 다시 수동으로 조정해야 하는데 이렇게 되면 원래 자동화를 통해 얻을 수 있던 유연성이 사라지게 된다.
2. 문서의 의미를 지정하는 것은 대부분의 사람들에게 낯선 일이며 제목을 몇 번 확대하는 것보다 훨씬 더 많은 작업이 필요하다. 예를 들어, 연구 논문을 작성하고 제목과 부제목을 강조하고 싶다고 가정해 보자. 아래한글을 사용한다면 제목 텍스트를 선택하고 “굵게”와 “글자 크기 키우기” 버튼을 클릭하기만 하면 된다. 하지만, 라텍이나 마크다운 같은 마크업 언어를 사용한다면, 제목을 특정 코드로 감싸서 그것이 ‘제목’임을 라텍에서 `\title{나의 연구 논문}`과 같이 작성해야 한다. 마크업 언어에 익숙하지 않은 저자는 불필요한 추가 작업처럼 느껴지고 전혀 직관적이지 않다.
3. 저자가 입력한 내용을 해석하고 표시할 내용을 파악하는 데는 컴퓨터 시간이 오래 걸린다. 왜 문서가 의도한 바를 반영하지 못하는지 알아내는데는 몇 배 시간이 듈다. 이것이 정확하게 프로그램을 디버깅하는 것과 같은 상황이고 디버깅은 대체로 쉽지 않은 작업이다. 예를 들어, PPT 슬라이드에 제목과 몇 개의 항목을 넣었을 때 컴퓨터가 자동으로 항목 크기를 줄인다. 이에 불만족해 수동으로 크기를 다시 조절하게 되는데, 원인은 컴퓨터가 사용자 의도를 완벽히 이해하지 못하기 때문에 발생된다.

지금까지 그 누구도 상기 문제를 모두 피하는 무언가를 발명하지는 못했다. 기 때문에 저작을 할 때면 오늘날 과학기술 연구자를 비롯한 많은 저자분들이 다양하고 혼동되는 선택지를 강요받게 되었다:

- 아래한글, 리브레오피스, 마이크로소프트 워드 같은 **데스크톱 위치워 도구**: 편지와 같은 간단한 문서를 만드는 가장 쉬운 방법이지만, 유연성이 떨어지고, 수식이나 버전 관리 시스템과 호환성이 좋지 않다.
- 구글 독스 같은 **웹기반 위치워 도구**: 워드나 한글, 리브레오피스와 비슷한 신속성을 갖추면서 협업도 수월(왜냐하면 모든이가 문서 사본 하나만 공유하기 때문)하지만, 여전히 유연성이 떨어지고 개인 정보를 민간 기업에 의존하는 것에 대한 불안감이 증가하고 있다.
- **데스크톱 라텍(LaTeX)**: 강력한 조판언어로 수식과 참고문헌 관리기능이 뛰어나고, 텍스트 형식으로 문서를 작성하기 때문에 버전 관리와도 잘 호환되지만, 지금 까지 학습하기 가장 복잡하고, 텍스트와 그림을 원하는 곳에 배치시키는 작업에 많은 시간이 소요된다.
- **Authorea, Overleaf** 같은 **웹기반 도구**: 위치워 편집 인터페이스를 저자에게 제공하지만 문서는 라텍으로 저장되고, 변경사항을 타이핑해서 넣을 때마다 실시간으로 화면에 다시 출력해서 보여준다.
- **HTML**: 웹의 기본언어로 라텍보다 훨씬 (훨씬) 더 단순하지만, 훨씬 더 적은 기능을 제공한다: 주석, 참고문헌 관리, 절마다 번호매기기 같은 단순한 기능도 직접적으로 지원되지 않는다. CSS²는 브라우저에 어떻게 표시할지 지시하는 언어로 복잡한 것으로 유명하다.
- **마크다운**: 일반-텍스트 전자우편 관례를 사용하여 HTML 간소화 대안으로 개발되었다. 빈줄은 문단을 구분하고, 이탤릭체로 만드는데 *별표*로 감싸는 등등. HTML보다 더 적은 작업을 수행하지만, 타이핑 양은 훨씬 더 적지만, 불행하게도 거의 모든 마크다운 구현결과물이 자체적인 기능이 추가되어서 “마크다운 표준”은 모순어법으로 볼 수 있다.

마크다운은 라텍과 HTML의 중간 정도의 복잡성을 갖고 있고, 라텍과 마찬가지로 텍스트 파일로 저장되기 때문에 버전 관리 시스템과 잘 호환된다. HTML과 마크다운 모두 직접적으로 수식을 지원하지 않지만, 플러그인 혹은 팩키지가 존재해서 저자가 LaTeX-유형 수식을 문서에 삽입할 수 있다.

²**종속형 시트, 캐스케이딩 스타일 시트(Cascading Style Sheets, CSS)** - 종속형 시트 또는 캐스케이딩 스타일 시트(Cascading Style Sheets, CSS)는 마크업 언어가 실제 표시되는 방법을 기술하는 언어로, W3C의 표준이며, 레이아웃과 스타일을 정의할 때의 자유도가 높다.

저작 도구를 선택할 때 마지막으로 고려할 점은 LaTeX 같은 데스크톱 텍스트 기반 시스템과 재현가능 문서저작을 지원하는 연산기능을 관리하는 다른 도구를 적절히 통합시키는 것이다. 적어도 지금으로는 전형적인 지구물리학 혹은 생물정보학 파이프라인과 구글 독스 혹은 리브레오피스를 통합하는 것이 훨씬 더 복잡하다. 예를 들어, 데이터가 변경될 때 그림이 자동으로 갱신되는 것을 들 수 있다.

8.3 엎친 데 덮친 격

위지윅과 조판/마크업 구분은 실제 파일형식보다 도구로 작업하는 것과 더 연관되어 있다. .docx 파일은 실제로 LaTeX, HTML, 마크다운 파일처럼 조판 명령어와 텍스트가 혼재되어 있다. 차이점은 조판/마크업 언어로 작성된 명령어는 사람이 읽을 수 있는 텍스트로 저장된다는 것이다. 이것이 함의하는 바는 유닉스 명령-라인 유ти리티가 처리할 수 있다는 점이다. ([스택오버플로우](#)에서 지적한 것처럼, 실제로 얼마나 많은 작업 수행할지에 대해 한계가 존재한다) 이와 비교해서, 아래한글, 마이크로소프트 워드, 리브레오피스에 내장된 서식 명령어는 특정한 전용 프로그램을 위해, 특정 프로그램에 의해서 제작되었다. 따라서, grep 같은 일반-텍스트 도구로는 처리가 되지 않는다.

구글 독스도 마찬가지다. 서식 명령어가 문서에 내장되어서 사용자 브라우저 자바스크립트에 의해 실행되어 사용자가 상호 작용하는 렌더링 페이지를 생성한다. 저장형식이 LaTeX이라는 점을 제외하면, Authorea와 Overleaf도 동일하다.

강성 프로그래머는 위지윅 도구와 텍스트가 아닌 형식을 조롱할 수도 있지만, 그들도 완벽하지 않다. 마이크로소프트 워드와 한글과 컴퓨터 아래한글은 수십년 동안 존재해 왔고, 그 기간 동안 문서 형식이 몇 번이나 바뀌었다. 그럼에도 불구하고, 명령줄을 선호하는 개발자는 이를 대체할 도구를 개발하지 못했다. 결과적으로, 버전제어 시스템 대부분은 세계에서 가장 널리 사용되는 문서 형식을 처리할 수 없다. 예를 들어, Git 같은 시스템은 두개 다른 버전의 마이크로소프트 워드 파일 혹은 아래한글 파일을 만났을 때 “차이가 있음” 정도만 제시할 수 있을 뿐이다. 이러한 상황은 미래 더 큰 생산성을 희망하며 수년 동안 효율적으로 사용해온 도구를 포기해야 한다는 것을 의미한다. 결국 버전 관리 도입은 미래의 생산성 향상을 위해 자신과 동료들이 수년간 생산적으로 사용해 온 도구를 버려야 하는 결과를 초래한다.

상기 논의는 저자가 논문과 편지만 작성하다고 가정했지만, 과학연구자는 자주 본인 작업을 시연하는데, 포스터와 슬라이드를 만들어야 하는 경우가 많다. 파워포인트는 말이

필요없는 발표도구의 여왕으로 많은 사람들이 파워포인트 때문에 발표가 엉망이 되어다고 비판하지만, 이것은 마치 시적 표현이 좋지 못한 것을 만연필 평계를 대는 것에 비견된다. 파워포인트와 파워포인트 유사도구는 컴퓨터 화면을 마치 칠판처럼 사람들이 쉽게 사용할 수 있게 만들었다. 글머리표 목록으로 구성된 너무나도 지루한 슬라이드를 쭉 생성할 수도 있지만, 쉽고 자유롭게 이미지, 도표, 텍스트를 섞어 사용할 수도 있다. LaTeX과 HTML로 그런 작업을 수행할 수 있지만, 어느 쪽도 그다지 쉽지는 않다. 사실, LaTeX이나 HTML 모두 어려워서 대부분 사람들은 신경도 쓰지 않는다. 설사 그런 작업을 수행하더라도, 그래픽적 요소는 문서의 중요부분이라기 보다 외부 삽입에 불과하다.

논문과 발표자료를 함께 생각하면 다소 불편한 상황에 있음을 인지하게 된다. 다른 한편으로, 논문과 발표자료는 연구 프로젝트에서 핵심적인 부분으로 코드와 데이터처럼 공유되고 추적관리되어야 한다. 다른 한편으로 [스테픈 터너\(Stephen Turner\)](#)는 다음과 같이 언급했다:

공동작업하는 지친 물리연구원에게 문서를 컴파일하는 개념을 설명하려고 한다고 보자. 그전에 일반 텍스트와 워드 프로세싱 사이 차이점을 설명해야 된다. 그리고 텍스트 편집기도 잊으면 안된다. 그리고 나서 마크다운/LaTeX 컴파일러. 그리고 BiBTeX, Git 그리고 GitHub 등등. 그러는 동안에 연구원은 다른 곳에서 호출 연락을 받을 것이다…

… 달리 설득시킬만큼 노력하지만, 과학컴퓨팅 외부 사람들과 협업할 때, 이러한 열개를 갖고 논문 협업을 하는 장벽이 너무나도 높아서 단순히 극복이 되지 않는다. 좋은 취지는 제쳐두고, 항상 “검토 메뉴에 변경내용 추적을 갖는 워드 문서만 주세요” 혹은 그와 유사하게 끝나게 된다.

가까운 장래에도 저작자 상당수는 순수 텍스트 조판 도구로 바꾸기보다는 계속해서 위치워 편집기를 사용할 것이다. [Authorea](#)와 [Overleaf](#) 같은 하이브리드 시스템이 절벽을 완만한 경사로로 바꿀 것이고, 프로그래머가 궁극적으로 다른 99% 사용자가 선호하는 문서저작에 관심을 가질 것이지만 수년에 걸친 작업량이 될 것이다.

저작자 대부분이 이미 아래한글, 마이크로소프트 워드 같은 데스크톱 위치워 시스템과 구글 독스같은 클라우드 대체 소프트웨어와 친숙하기 때문에 기존 관행이 바뀌기는 어렵다. 하지만, 웹사이트와 블로그를 위한 마크다운, 논문 원고저작을 위한 라텍이 여전히 영향력이 강하다. 웹 문서작업에 마크다운으로 HTML 문서를 저작하는데 큰 어려움이 없지만, 논문 원고저작에 마크다운을 사용하면 학술지 대부분은 받아주지 않고, 이

미 라텍을 사용하고 있는 동료 저작자가 라텍을 버리고 마크다운을 채택할 가능성도 낮고, 저작자들이 문서작업에서 원하는 상당수 기능(예를 들어, 참고문헌 서지관리)을 구현하지 못하고 있다. 반면에 라텍은 PDF 형식으로 컴파일할 수 있고, 그림과 표 레이아웃을 잘 처리하고 버전 제어 시스템과 호환되며, 다양한 문헌 관리 소프트웨어와도 호환되지만 블로그와 웹사이트로 대표되는 웹 출판에 적합하지 않다.

제 9 장

디지털 글쓰기

글쓰기는 인류의 역사와 함께한 중요한 소통 도구로서, 그 방식과 형태는 시대별로 지속적으로 변화해왔다. 최근의 글쓰기 환경은 그 전보다 훨씬 복잡하고 다양한 요소를 요구한다. 과거에는 원고지에 텍스트를 작성하는 것만으로도 충분했다. 한글이나 영어로 이루어진 문장들은 감정을 전달하거나 사실을 나열하며, 이를 통해 독자와의 소통을 이루었다. 그러나 현대의 글쓰기 환경에서는 단순한 텍스트 정보 전달만으로는 부족하다. 이미지, 그래픽, 그리고 인터랙티브한 요소들이 텍스트와 함께 통합되어야 하며, 이를 통해 독자의 관심을 끌고 정보의 흡수를 증진시킨다.

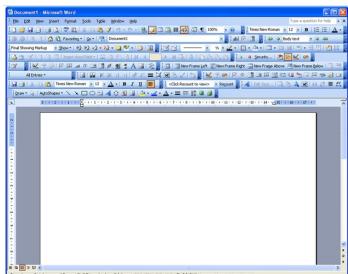
가독성 또한 중요한 고려사항이다. 디자인적인 측면에서 효과적인 글꼴 선택은 텍스트의 내용을 더욱 돋보이게 하며, 독자의 읽기 경험을 향상시킨다. 현대의 글쓰기는 단순한 정보 전달을 넘어서, 그 과정과 결과의 재현성에도 중점을 둔다. 특히 과학연구나 기술문서에서 두드러진다. 자동화 도구와 버전 관리 시스템의 도입을 통해 재현성을 지원하고, 글쓰기 효율성과 투명성을 동시에 높이고 있다.

최근의 눈에 띄는 추세 중 하나는 인공지능 발전과 그에 따른 기계 저작물의 폭증을 들 수 있다. 기계가 작성한 텍스트를 포함한 콘텐츠가 인간 저자와 비슷하거나 오히려 생산 속도나 경제적인 면에서 확실한 우위를 보이면서 이에 대한 연구와 기술적 시도가 활발히 이루어지고 있다.

디지털 글쓰기는 검색 엔진에 의해 쉽게 찾아질 수 있도록 설계되어 있으며, 다양한 미디어가 쉽게 통합될 수 있고, 디지털 글쓰기는 발행 후에도 언제든지 쉽게 조작할 수 있

글쓰기 저작

GUI : 위즈워(WYSWYG)

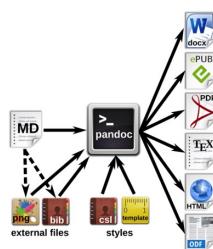


LaTeX, Pandoc: 위즈워(WYSWYM)

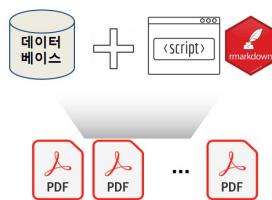


자동화

Pandoc: One Source Multi-Use



Parameterized Documents



기계창작

AI 저작물



Text-to-X



그림 9.1: 현대의 글쓰기: 기술적 도전과 진화의 중심

어 더 동적이고 생동감을 넣을 수도 있고, **깃(Git)**과 같은 버전제어 도구와 연결시켜 글쓰기를 하게 되면 추적도 가능하여 재현가능한 글쓰기가 가능하게 된다. 디지털 글쓰기의 가장 큰 장점 중의 하나는 사람만이 저작을 하는 것이 아니라 기계가 저작한 프로그램도 글쓰기에 담을 수 있어 더욱 풍성한 글쓰기가 가능하다는 점이다.

반면 전통적인 글쓰기는 발행 후 개발이 중단될 뿐만 아니라 한정된 지면(예를 들어 A4, B5)에 다양한 색상을 넣어 강조와 가독성을 높일 수 있으나 인쇄 난이도의 증가로 인해 비용도 급격히 증가하게 되고 무엇보다 정적인 정보만 담을 수 있어 최신 기술 흐름을 반영하는데 한계가 있다.

또한 커뮤니케이션 관점에서 보면 디지털 글쓰기는 온라인 도구를 사용하여 작성 과정의 초기부터 광범위한 청중과 거의 실시간으로 텍스트를 공유할 수 있어 작성 과정이 공개적이고 협업을 기반으로 하고, 저작과정에 별다른 장벽없이 참여할 수 있다는 점에서 전통적인 글쓰기가 소수 저자가 집필을 마무리해야 후속 디자인 작업을 비롯한 후속 출판작업이 진행된다는 면에서 큰 차이가 있다. 특히 전통 글쓰기 결과물이 책과 같은 물리적 매체에 담기게 되면 이를 다시 디지털로 되돌리기 위해 광학문자인식(OCR) 과정을 거쳐야 하며 책속에 담긴 그래프, 표, 이미지 등 데이터는 별도 과정을 거쳐 디지털화 하는 한계도 명확하다.

요약하면, 전통적인 글쓰기가 종이와 펜으로 생각이나 정보를 자유로이 펼칠 수 있어 저자에게 자유로운 표현을 가능했지만 반대급부로 공동저작과 협업이 쉽지 않으며 재사용이 어렵고 증거기반 재현가능한 과학기술 저작물 작성에 뚜렷한 한계가 있고 저작부터 출판까지 시간이 오래소요되고 비용이 많이 들고 품질관리가 쉽지 않은데 최근 등장한 챗GPT와 통합은 불가능에 가깝다는 점이다.

디지털 전환이 가속화되면서 저자의 생각과 전달하는 바를 명확히 하기 위해서는 다양한 디지털 저작 언어와 도구 및 챗GPT와 같은 인공지능도 활용하여 저작해야만 한다는 점에서 디지털 글쓰기는 전통적인 글쓰기와 비교하여 큰 차이가 있다.

9.1 저작 언어

문서 작성은 위한 언어로 제도, 조판, 마크업, 스타일링 언어가 기본적으로 필요하다. **제도 언어(Drawing Language)**는 그래픽 디자인과 도면 생성을 위한 SVG와 AutoCAD가 대표적이고, **조판 언어(Typesetting Language)**는 문서 서식과 인쇄를 목적으로 라텍(LaTeX)과 텍(TeX)이 대표적이다. **마크업 언어(Markup Language)**는 문서를 구조화하는 것을 목적으로 HTML, XML, 마크다운 등이 있고 **스타일링 언어(Styling Language)**는 디자인과 레이아웃을 주로 다루며 CSS, SASS, LESS 등이 해당된다.

개발을 위한 프로그래밍 언어로 먼저, 명령형 프로그래밍 언어는 범용 프로그래밍과 알고리즘 구현을 위한 C/C++, 자바, 파이썬 등이 있고, 선언형 프로그래밍 언어는 데이터 처리와 문서 구조화를 목적으로 SQL, HTML, XML 등이 대표적이다. 함수형 프로그래밍 언어는 복잡한 함수 처리와 데이터 분석을 위해 하스켈, Lisp, R 등이 사용되고, 객체지향 프로그래밍 언어는 소프트웨어 개발과 시스템 설계를 위한 C/C++, 자바, 파이썬 등을 꼽을 수 있고, 스크립팅 언어는 자동화와 웹 개발을 주로 다루며 자바스크립트, 파이썬, 루비 등이 해당된다. 쿼리 언어는 데이터베이스 관리와 데이터 검색을 위해 SQL, GraphQL, 정규 표현식 등이 사용되고 마지막으로 동시성 언어는 병렬 처리를 위한 고(Go), 어랭(Erlang), 클로저(Clojure) 등이 있다.

목적	언어/기술 유형	주요 목적	예시
문서	제도 언어	그래픽 디자인, 도면 생성	SVG, AutoCAD
문서	조판 언어	문서 서식, 인쇄	LaTeX, TeX

목적	언어/기술 유형	주요 목적	예시
문서	마크업 언어	문서 구조화	HTML, XML, 마크다운
문서	스타일링 언어	디자인, 레이아웃	CSS, SASS, LESS
개발	명령형 프로그래밍 언어	범용 프로그래밍, 알고리즘 구현	C/C++, 자바, 파이썬
개발	선언형 프로그래밍 언어	데이터 처리, 문서 구조화	SQL, HTML, XML
개발	함수형 프로그래밍 언어	복잡한 함수 처리, 데이터 분석	하스켈, Lisp, R
개발	객체지향 프로그래밍 언어	소프트웨어 개발, 시스템 설계	C/C++, 자바, 파이썬
개발	스크립팅 언어	자동화, 웹 개발	자바스크립트, 파이썬, R, 루비
개발	쿼리 언어	데이터베이스 관리, 데이터 검색	SQL, GraphQL, 정규 표현식

i 마이크로소프트 워드

마이크로소프트 워드는 1983년에 출시된 워드 프로세서로 마이크로소프트 오피스 일부로 판매되고 있다. 워드 프로세서는 문서 작성을 위한 소프트웨어로 텍스트 편집, 조판, 문서 서식, 인쇄 등을 지원한다. 워드 프로세서는 텍스트 편집과 조판 기능 외에도 문서 작성을 위한 다양한 기능을 제공한다. 아래한글과 기술기반을 공유하고 있는 마이크로소프트 워드는 편집과 조판이 중심이지만 다양한 언어와 기술을 조합하여 제작되었다.

1. **마크업 언어:** 워드 문서는 내부적으로 복잡한 마크업 언어를 사용하여 문서 콘텐츠와 스타일을 분리하고 관리한다.
2. **스타일링 언어:** 워드에는 텍스트 스타일링과 문서 레이아웃을 제어하는 내장 기능이 있어 CSS와 유사한 역할을 하지만 GUI를 통해 제어된다.
3. **스크립팅 언어:** 워드는 VBA(Visual Basic for Applications)와 같은 스크립팅 언어를 내장하여 사용자가 자동화 매크로를 작성할 수 있다.
4. **쿼리 언어:** 워드는 데이터베이스나 다른 외부 데이터 소스와 연동할 수 있고

쿼리 언어가 사용된다. 대표적으로 이름, 주소 등만 다르고 같은 내용이 반복되는 상장, 초대장 등에 메일 머지("Mail Merge") 기능을 사용할 때 데이터 베이스에서 정보를 가져와 개별적인 문서를 자동으로 생성시킨다.

5. **텍스트 처리 엔진**: 워드는 복잡한 텍스트 레이아웃과 조판을 처리하는 자체 엔진을 가지고 있다. 기본적인 텍스트 입력 및 편집 뿐만 아니라, 고급 서식, 스타일링, 그래픽 삽입, 표 작성, 수식 편집, 문법 및 철자 검사 등 다양한 기능을 제공한다.
6. **그래픽 언어**: 워드는 기본 도형 삽입, SmartArt를 통한 다이어그램 생성, 텍스트 상자와 WordArt, 이미지 편집, 선과 테두리 스타일링, 레이어와 그룹 관리, 드래그 앤 드롭 인터페이스, 커스텀 도형 및 디자인 생성 등이 포함된다. 워드 내에서 그래픽과 도면을 쉽게 조작할 수 있고 간단하지만 효과적인 그래픽 표현 언어기능도 내장되어 있다.

9.2 문학적 프로그래밍

최근 저작물은 전통적인 출판 패러다임을 넘어서 급격한 변화를 겪고 있다. 이러한 변화의 중심에는 쟁GPT와 같은 생성형 AI의 부상, 다양한 출판 형식 출현과 자동화, 문학적 프로그래밍(literate programming)이라는 새로운 패러다임의 도입이 큰 역할을 하고 있다.

과학기술 저작물은 이제 다양한 출판 형식을 넘어서, 문학적 프로그래밍의 도입으로 새로운 차원의 변화를 겪고 있다. 이 패러다임은 코드와 텍스트를 결합하여 독자에게는 이해하기 쉽고 읽기 좋은 문서를 생성하는 것이 기본이다. 더 나아가, 저자에게는 이러한 방식이 버전 제어 시스템과 통합되어 협업이 강화되고, 문서의 재사용과 유지보수 가 더욱 수월해진다. 이로 인해 대규모 문서 저작도 가능해지고 있다.

이러한 변화의 배경에는 빅데이터, 데이터 과학, 생성형 AI와 같은 과학기술 정보의 전달과 공유를 더 효율적으로 하기 위한 필요성이 크게 작용하고 있다. 또한, 인터랙티브 상호작용 기능은 문서의 가독성과 커뮤니케이션 능력을 한층 높여주어, 정보의 효율적인 관리와 전달에 있어 새로운 가능성은 제시하고 있다.

R마크다운(R Markdown)은 10년 이상 다양한 형태의 재현가능하고 추적 가능한 문서 저작에 있어 가능성을 책(bookdown), 블로그(blogdown), 패키지 매뉴얼(pkgdown),

PPT(xaringan), 대쉬보드(flexdashboard)와 같은 사례를 통해 증명해왔다. 쿼토는 차세대 R마크다운이라는 별명에 걸맞게 기존의 장점을 유지하면서도, 생성형 AI와 챗GPT가 주도하는 현재의 인공지능 시대에 적합한 새로운 패러다임을 제시하고 있다.



그림 9.2: 쿼토 개요

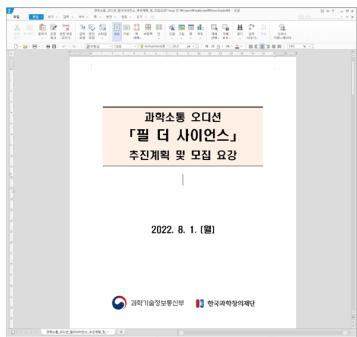
9.2.1 위치윅 vs 위치웜

신속하고 빠르게 누구나 짧은 학습을 통해서 문서를 저작하고 출판할 수 있는 방식은 아래한글 혹은 MS워드 워드프로세서를 사용하는데 이는 **위치위그(WYSIWYG: What You See Is What You Get, “보는 대로 얻는다”)**에 기초한 것으로 문서 편집 과정에서 화면에 포맷된 낱말, 문장이 출력물과 동일하게 나오는 방식을 말한다. 위치윅의 대척점에 있는 것이 **위치웜(WYSIWYM, What You See Is What You Mean)**으로 대표적인 것인 라텍으로 구조화된 방식으로 문서를 작성하면 컴파일을 통해서 최종 문서가 미려한 출판가능한 PDF, PS, DVI 등 확장자를 갖는 출판결과물을 얻을 수 있다.

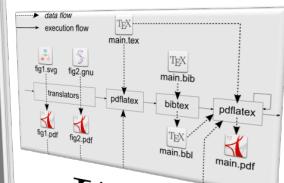
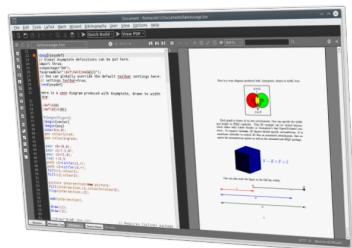
9.2.2 라텍 철학

라텍은 한마디로 정의하게 되면 “**논리적인 디자인**”이라고 볼 수 있다. 저작물이 만들어지는 과정은 저작자가 원고를 손으로 쓰거나 타자기로 쳐서 출판사에 넘겨주면, 출판사의 편집디자이너는 원고를 보고 세부적인 출력형식을 결정하여 인쇄소에 넘긴다. 인

WYSIWYG (위지윅)



WYSIWYM (위지윔)



LATEX



그림 9.3: 위치윅과 위치윔

쇄소는 이를 토대로 과거 식자공이 식자판을 만들었다면 현재는 컴퓨터가 파일을 만든다. 라텍과 텍이 하는 업무가 다소 차이가 난다. [kim2017tex]

- 라텍은 편집디자이너에 해당되는 업무를 수행
- 텍은 식자공에 해당되는 업무를 수행

컴퓨터를 활용하여 전자조판을 넘긴 이유는 문서를 좀 더 읽기 쉽게 만들어 독자에게 저작자의 생각을 잘 이해시키는 것이다. 이러한 목적을 달성하기 위해서는 문서의 디자인이 그 문서의 논리적 구조를 잘 반영시켜야 하고, 반대로 문서의 논리적 구조를 잘 반영시키기 위해서는 문서 디자이너가 문서의 논리적 구조를 잘 이해해야만 된다. 라텍 명령어는 기본적으로 문서의 논리적 구조를 기술하는 것이다. 이렇게 기술된 명령어를 텍으로 전달하여 조판하게 되어 원하는 최종 결과물을 얻게 된다.

라텍에서 채택하는 논리적 디자인의 가장 큰 장점은 글을 더 잘 쓰게 된다는 점에 있다. 라텍으로 글을 작성하게 되면 점점 라텍이 채택하고 있는 논리적 디자인에 맞추게 되고 이를 통해서 글쓰기 실력이 향상된다.

아마도 라텍의 가장 큰 장점은 자동화에 있다. 시각적 디자인(Wysiwig)을 채택하여 작성한 문서를 다른 형식으로 변환하거나 문서의 수식의 일련번호를 로마자에서 아라비

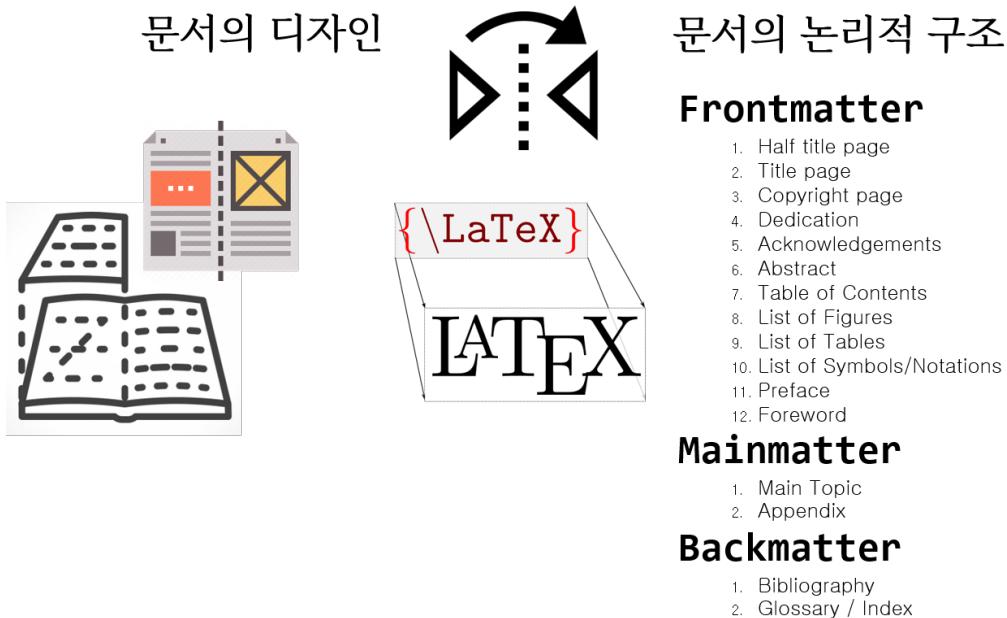


그림 9.4: 문서의 논리적 구조와 디자인

아 숫자로 모두 변경시키거나 표나 그래프 번호를 일괄변경하는 등 이러한 사례는 자주 발견된다. 특히, 국내외 저널에 제출할 논문을 라텍으로 작성한 경우 템플릿만 바꾸면 쉽게 고품질 논문을 만들 수 있는데 워드를 이용하여 작성할 경우 수작업을 하게 되어 비생산적인 시간낭비도 크고 미려한 문서를 얻을 수도 없다.

따라서, 라텍으로 글을 쓰게 되면 글쓰기 본질에 집중할 수 있어 저작자의 생각과 의견을 좀더 진솔하게 가감없이 독자에게 전달시킬 수 있게 된다.

9.2.3 블로그 저작

문서를 논리적으로 디자인하는 반대 개념으로 있는 것이 **위즈윅(WSIWIG, What You See Is What You Get)**으로 대표적인 것이 아래한글, MS 워드와 같은 워드 프로세서다. 시각적 디자인을 하게 되면 논리적이지 못한 애매한 조판을 하기 쉽다. 또한, PDF 파일로도 출력을 할 때 미세하나마 출력물에 일관성이 실종되기도 한다.

개인용 컴퓨터가 보급되면서 아래한글과 같은 워드 프로세서를 사용해서 저작을 하는 것이 일반화되었지만 곧이어 인터넷이 보급되면서 웹에 문서를 저작하는 것이 이제는 더욱 중요하게 되었다. 전문 개발자가 아닌 일반인이 HTML, CSS, JavaScript를 학습

하여 웹에 문서를 제작하고 출판하는 것은 난이도가 있다보니 워드프레스와 티스토리 같은 위치워 패러다임을 채택한 저작도구가 사용되고 있으나 상대적으로 HTML, CSS, JavaScript을 조합한 위치웹 방식과 비교하여 고급스러운 면과 함께 정교함에 있어 아쉬움이 있는 것도 사실이다.

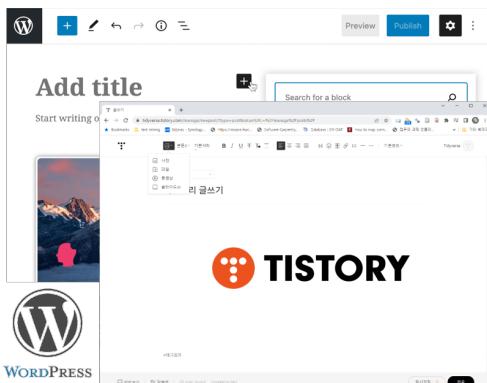


그림 9.5: 워드프레스와 티스토리

웹출판 위치워과 위치웹



그림 9.6: HTML + CSS + 자바스크립트

9.2.4 재현가능 글쓰기

과학의 근본적인 원칙 중 하나는 연구의 결과가 독립적으로 재현가능해야 한다는 것이다. 이 원칙은 연구 타당성을 확인하고, 그 결과가 일반화 가능한지 검증하는 데 필수적이다. 그러나 2010년 전후, 다양한 학문 분야에서 재현할 수 없는 연구 결과가 다수 발표됨에 따라 과학 커뮤니티 내에서 큰 우려가 발생하였고, 이러한 현상을 **재현성 위기 (Reproducibility Crisis)**라고 부르며, 과학연구 신뢰성을 크게 훼손시켰다.

이 위기에 대응하여, 전 세계 연구자들과 전문가들은 재현가능한 연구 저작을 촉진하고 지원하기 위한 다양한 방안을 모색하기 시작했다. 연구방법론 표준화, 데이터 공유, 연구 소프트웨어 투명성 강화 등 다양한 활동이 들불처럼 일어났고, 기술 커뮤니티 형성과 더불어 다양한 도구와 플랫폼 개발을 통해 재현성 문제를 해결하고자 하는 움직임이 확산되고 있다.

데이터 과학에서 재현성은 핵심적인 요소로 간주된다. 연구 결과의 신뢰성과 타당성을 검증하는 데 있어, 다른 연구자나 전문가들이 동일한 결과를 도출할 수 있도록 연구 과정을 투명하게 공유하는 것은 필수적이다. 이러한 맥락에서 R 팩키지는 재현가능한 데

이터 분석 프로젝트를 지원하는 데 중요한 역할을 한다.

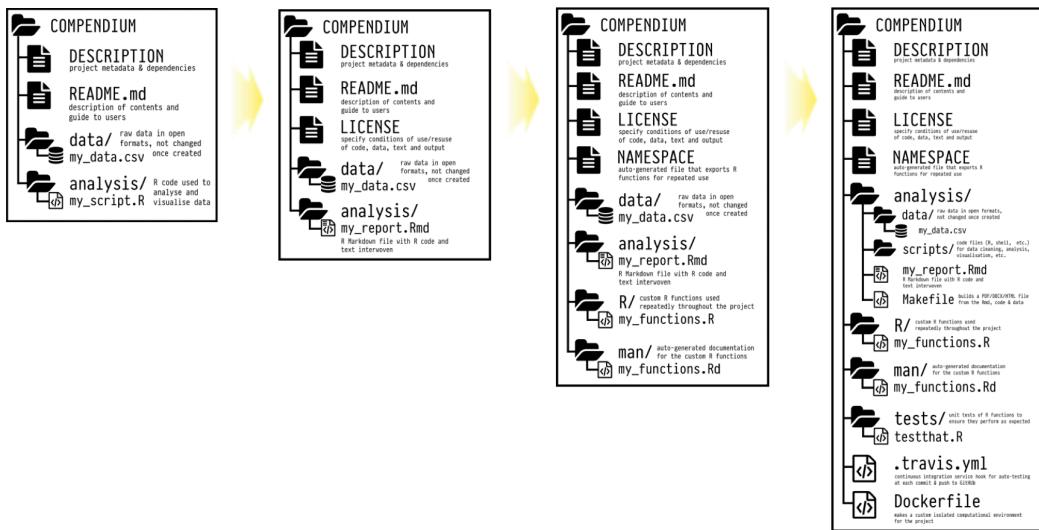


그림 9.7: 재현가능한 과학기술 문서 제작과정

특히, R 팩키지를 활용한 **연구 개요서(research compendia)**는 재현성을 강화하는 훌륭한 도구로 간주된다. 연구 개요서는 연구의 모든 요소(코드, 데이터, 출력 결과물, 환경 등)을 포괄적으로 담고 있어, 연구의 전체 과정을 투명하게 추적하고 검증할 수 있다. 한 사례로 그림 ?? 에 코드부터 단계별로 재현가능한 과학기술 문서 제작과정을 보여주고 있다.

9.3 쿼토와 챗GPT의 등장

쿼토와 챗GPT의 등장은 문서 작성과 관리에 있어 혁신적인 변화를 불러오고 있다. 쿼토는 10년 이상 R마크다운을 통한 검증 과정을 거쳐 실용성이 입증되었다. 쿼토는 위치와 순수 텍스트 조판 장점을 결합하여 더욱 유연하고 효율적인 문서 작성 환경과 경험을 제공한다. 쿼토는 R, 파이썬, 자바스크립트, 줄리아, SQL 등 다양한 프로그래밍 언어를 문학적 프로그래밍(Literate Programming) 패러다임에 따라 통합, 재현 가능한 과학기술 연구를 위한 문서와 코드 작성을 지원한다. 복잡한 레이아웃과 수식도 쉽게 처리할 수 있을 뿐만 아니라, 베전 제어와 같은 고급 기능도 지원하고 다양한 형식(HTML, PDF, PPT, ePub, 웹사이트 등)으로의 내보내기 기능을 통해 과학기술 논문 저작, 도서 출판, 웹 출판 등이 수월하다.

챗GPT는 자연어 처리(NLP) 기술을 이용해 문서 작성, 편집, 그리고 검토 과정을 자동화하거나 저작작업을 보조한다. 복잡한 문서 구조와 내용을 신속하게 파악하여, 저자 의도에 맞는 형식과 스타일로 재구성할 수 있다. 더불어, 문서 의미나 목적에 따라 적합한 템플릿을 추천하고, 특정 주제에 대한 추가 정보나 자료도 제공한다. 이러한 기능들이 쿼토와 같은 문서 저작 도구와 통합되면, 저자들은 문서를 더 효율적이고 직관적으로 작성하고 관리할 수 있을 것이다. 챗GPT와 쿼토의 만남은 문서 저작의 새로운 패러다임을 제시하고 있다.

제 III 편

기본요소

제 10 장

문서 구성요소

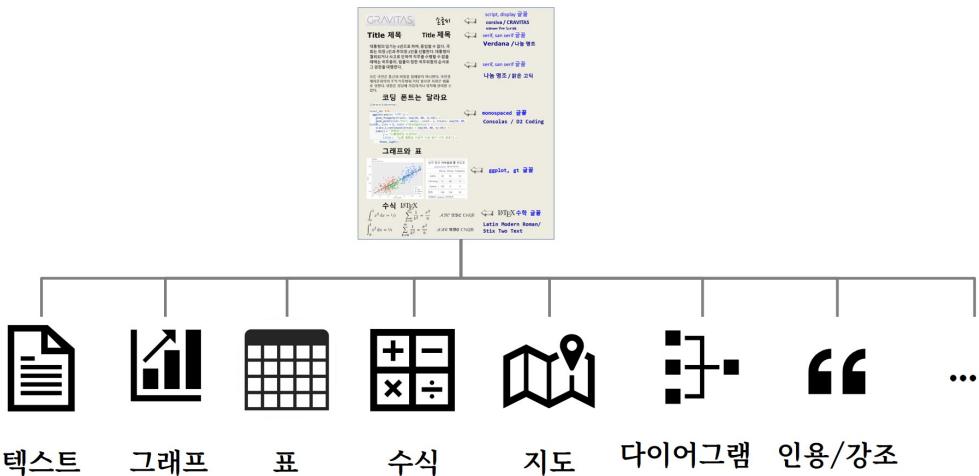


그림 10.1: 디지털 문서 구성요소

10.1 그림

그림은 책 내용을 이해하는 데 도움을 주는 중요한 요소이다. 그림은 과거 실제 사물을 촬영한 사진과 사물이나 생각을 그려서 표현한 것이 전부였다면, 이제는 데이터를 기반으로 다양한 그래프도 만들어낼 수 있고 생성형 AI 기술을 사용해서 시각적 표현을 만들어낼 수도 있다.

그림은 문서에서 텍스트만으로 설명이 어려운 개념이나 데이터를 시각적으로 표현함으로써 이해를 돋고 문서구조를 논리적으로 구성하는 데도 기여한다. 그림 위치는 그림이 설명하려는 내용과 얼마나 밀접하게 연관되어 있는지에 따라 달라지는데 일반적으로 그림은 관련된 텍스트 바로 다음이나 전에 위치하는 것이 일반적이고, 그림에 대한 참조를 본문에서 명확히 하는 것이다.

그림과 같은 시각적 객체가 준비되면 문서와 조화를 이룰 수 있도록 그림 크기, 정렬, 레이아웃, 캡션, 상호참조 등을 고려해야 한다. 특히, 문서의 최종 출력 형태를 고려해야 하는데 많이 사용되는 대표적으로 HTML, PDF, 아래한글에 각각 그림이 문서에 포함되면 다음과 같다.

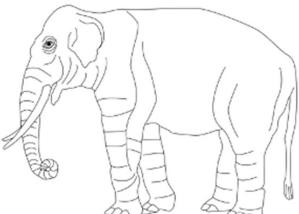
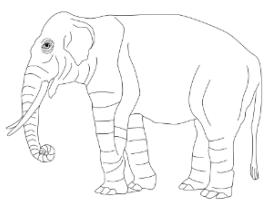
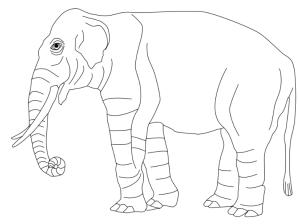
아래한글	HTML	PDF
		

그림 1 코끼리

Figure 1: Elephant

Elephant

그림을 문서에 삽입할 때 그림이 문서의 전체 흐름과 내용에 잘 맞춰 물흐르듯 자연스럽게 구성한다. 그림을 단순히 장식적인 목적으로 넣어서는 안 되며, 본문의 내용을 보충하거나 설명하는 데 도움이 되어야 한다. 그림 크기가 너무 크거나 작으면 읽기 어렵고, 해상도가 낮으면 조약해 보여 글의 품격도 떨어뜨린다. 그림 번호(레이블)는 문서 내에서 그림을 참조할 때 사용되고 그림 설명글(캡션)은 그림 내용을 간략하게 설명하는 기능을 한다. 저작권 관련하여 그림 출처나 저작권 정보도 명기해야 하고, 그림 색상과 스타일이 문서 전체 디자인과 잘 어울리게 조화를 이루어야 한다.

```
library(tidyverse)
library(openai)

extrafont::loadfonts()
```

```
Sys.setenv(OPENAI_API_KEY = Sys.getenv("OPENAI_API_KEY"))

x <- create_image("강원도 설악산 멋진 풍경")

download.file(url = x$data$url, destfile = "images/dalle_mountain.png", mode =
  "wb")
```

특히 생성형 AI 기술의 눈부신 발전으로 인해 사람이 그린 것 같은 정교한 이미지를 문서에 넣는 것도 비용이 많이 들지 않고 쉽게 가능하다. 이제는 텍스트로 작성되는 문단과 문단 사이 적절한 곳에 “프롬프트”를 넣어 이미지를 넣는 것이 큰 어려움 없이 가능하게 되었다. 다음은 최근에 공개된 OpenAI 달리3(DALL·E 3) 모델을 사용해서 ‘강원도 설악산 멋진 풍경’이라는 프롬프트로 생성한 이미지다.



그림 10.2: AI 생성 이미지

10.2 지도

지리학(Geography)은 가장 오래된 과학 중 하나로 꼽히며, 학생때부터 교실 여기저기에 서 볼 수 있는 지도가 아마도 처음 접한 데이터 시각화 산출물일 것이다. 지리학은 지구의 곡률을 정확히 파악하여 지구의 크기와 모양, 중력에 대한 이해를 높이는 데 중요한 역할을 수행했을 뿐만 아니라 대항해 시대 지도가 매우 큰 기여를 했고, 현대에 와서 지리학 데이터 응용 범위는 더욱 확장되어, 민간과 국방 범위를 가리지 않고 상업적으로 입점위치 선정, 우주 탐사, 군사 작전, 인구이동 패턴 분석 등 다방면에서 요긴한 역할을 수행하고 있다.

지도는 글쓰기에서 중요한 구성요소 중 하나로, 특히 학술 논문, 보고서, 여행기, 지리나 역사 분야에서 지도는 필수적이다. 지도는 텍스트로만 설명하기 어려운 공간적 관계나 지리적 배치를 명확하게 시각화함으로써 저자의 주장이나 설명을 더 쉽고 정확하게 이해시킬 수 있고 문서의 전문성과 신뢰성을 높힘으로서 효과적인 커뮤니케이션을 가능하게 한다.

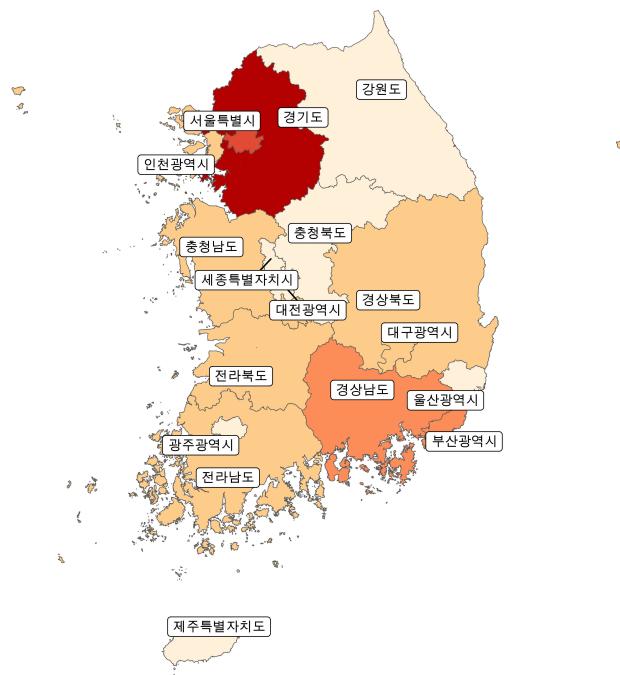
한가지 사례를 들어 보자. 2023년 7월 기준 대한민국 시도별 인구수를 문서의 한 구성요로서 글쓰기를 할 때, 텍스트로 설명하거나 표로 정리하는 방법, 막대그래프로 시각화하는 방법이 경우에 따라 유용할 수 있지만, 시도라는 지리적인 정보를 지도에 녹여 표현하는 것이 가장 효과적이다.

예를 들어, 서울과 부산 인구수가 얼마나 차이가 나는지 텍스트나 표, 그래프로만 보면 그 차이를 알 수는 있지만, 두 도시가 어디에 위치해 있는지, 주변 지역과 어떤 공간적 관계를 가지고 있는지 쉽게 파악하기 어렵다. 반면 지도를 사용하면, 서울과 부산이 어디에 위치해 있는지, 각 지역 인구수가 어떻게 분포되어 있는지를 한눈에 볼 수 있다. 지도를 통해 제공되는 공간적 맥락은 텍스트나 표, 그래프만으로는 얻을 수 없는 근본적인 이해와 통찰력을 제공한다는 점에서 지도를 통한 시각화는 정보 전달과 이해에 있어서 훨씬 더 우위를 차지한다고 할 수 있다.

```
library(sf)
library(tidyverse)
sf_use_s2(FALSE)

## 지도
```

```
korea_map <- read_sf("data/HangJeongDong_ver20230401.geojson")\n\nsido_map <- korea_map |>\n  group_by(sidonm) |>\n  summarise(geometry = sf::st_union(geometry))\n\n## 23년 7월 인구수(KOSIS) 행정구역별, 성별 인구수\npop_tbl <- read_csv("data/행정구역_시군구_별__성별_인구수_20230831223248.csv",\n  locale=locale(encoding="euc-kr"), skip = 1) |>\n  set_names(c("sidonm", "인구수")) |>\n  mutate(sidonm = if_else(sidonm == "강원특별자치도", "강원도", sidonm))\n\nsigo_gg <- sido_map |>\n  left_join(pop_tbl) |>\n  ggplot() +\n    geom_sf(aes(geometry = geometry, fill = cut(인구수, 10)), show.legend = FALSE) +\n    ggrepel::geom_label_repel(aes(label = sidonm, geometry = geometry),\n      size = 20, stat = "sf_coordinates") +\n    theme_void() +\n    scale_fill_brewer(palette = "OrRd")\n\nragg::agg_jpeg("images/GIS_tools.jpeg",\n  width = 10, height = 7, units = "in", res = 600)\nsigo_gg\ndev.off()
```



10.3 그래프

데이터를 시각적으로 표현하는 그래프는 증거기반 문서작성에 중요한 역할을 담당한다. 데이터 시각화를 위한 그래프 생성 프로그래밍 언어는 다양한데 “방법(how)”에 초점을 둔 명령형(imperative) Processing, OpenGL, prefuse 언어 계열과 “무엇(what)”에 초점을 선언형(declarative) D3, ggplot2, Protovis 등으로 나뉜다. R은 통계분석을 위한 프로그래밍 언어로서 데이터 시각화에 강점을 가지고 있다.

그래프는 복잡한 데이터나 패턴을 빠르고 명확하게 전달할 수 있는 수단이기 때문에 데이터를 시각적으로 표현하는 그래프는 증거기반 문서 작성에서 중요한 역할을 한다. 그래프를 생성하기 위한 프로그래밍 언어나 패키지는 크게 두 가지 접근 방식으로 나뉜다. 하나는 “방법(how)”에 초점을 둔 명령형(imperative) 언어로, Processing, OpenGL, prefuse 등으로 그래프 각 요소를 어떻게 표현할 것인지를 명시적으로 지시한다. 다른 하나는 “무엇(what)”에 초점을 둔 선언형(declarative) 언어로, D3, ggplot2, Protovis 등으로 무엇을 그릴 것인지를 중심으로 설계되어 있어, 사용자가 원하는 결과에 더 집중한다.

R 언어는 통계 분석뿐만 아니라 데이터 시각화에도 강점이 있고, 크게 3가지 R 언어기

반 시각화 시스템이 존재한다. Base 시각화 시스템은 R 기본 패키지에 포함되어 있는 기본적인 그래프 생성 도구다. 래티스(lattice) 시각화 시스템은 클리블랜드(Cleveland)의 격자 그래픽(Trellis Graphics)에서 영감을 받아 개발되어 특히, 다변량 데이터를 효과적으로 시각화할 수 있다. ggplot 시각화 시스템은 SPSS 윌킨슨(Wilkinson)의 그래프 문법(Grammar of Graphics)에 기반하여 탄탄한 이론적 토대를 갖고 있으며 파이프 연산자를 +과 ggplot2에 도입되어 의식의 흐름에 맞춰 그래프를 생성할 수 있다는 장점이 있다.

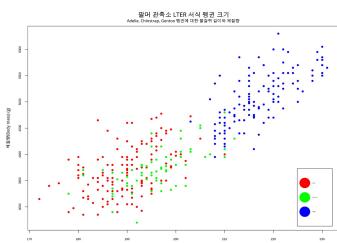


그림 10.3: Base 시스템

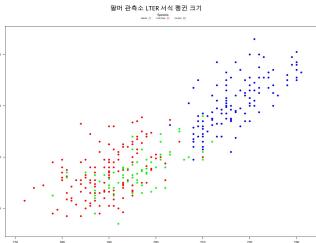


그림 10.4: Trellis 시스템

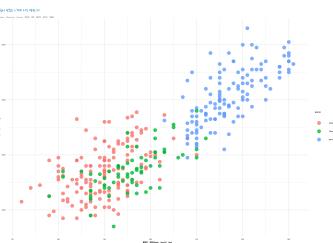


그림 10.5: ggplot2 시스템

그림 10.6: R 그래프 시스템

팔미 관측소 펭귄 데이터를 시각화하는 예제를 통해 일반적인 데이터 시각화 구성 요소와 작업흐름을 일별해보자. 데이터 시각화에 필요한 패키지(tidyverse, showtext, palmerpenguins)를 불러온다.

그래프에 필요한 글꼴("Nanum Pen Script", "Jua" 글꼴)을 showtext 패키지를 사용하여 구글 폰트 사이트에서 가져와서 설치하고, 그래프 제목과 부제목 글꼴, 크기, 색상 등을 theme_quarto 테마로 설정하여 R에서 사용할 수 있도록 준비한다.

palmerpenguins 데이터 패키지에서 팔미 관측소 펭귄 데이터프레임을 불러오고, ggplot 을 사용하여 펭귄 물갈퀴 길이(flipper_length_mm)와 체질량(body_mass_g)을 x축과 y축에 각각 맵핑하고, 펭귄 종(species)에 따라 색상을 다르게 표시하는 산점도를 생성하고, 그래프 제목, 부제목, x축과 y축 제목도 넣고 theme_quarto 테마도 반영하여 시각화 객체를 완성한다.

```
library(tidyverse)
library(showtext)
library(palmerpenguins)

font_add_google("Nanum Pen Script", "nanum_pen_script")
```

```
font_add_google("Jua", "Jua")
showtext_auto()

theme_quarto <- theme(
  text = element_text(family = 'Jua', size = 25),
  plot.title.position = 'plot',
  plot.title = element_text(
    family = 'nanum_pen_script', size = 55,
    face = 'bold',
    colour = thematic::okabe_ito(8)[3],
    margin = margin(t = 2, r = 0, b = 3, l = 0, unit = "mm")
  ),
  plot.subtitle = element_text(
    family = 'Jua', size = 25,
    face = 'bold',
    colour = thematic::okabe_ito(8)[5],
    margin = margin(t = 0, r = 0, b = 0, l = 0, unit = "mm")
  )
)

theme_set(theme_minimal() + theme_quarto)

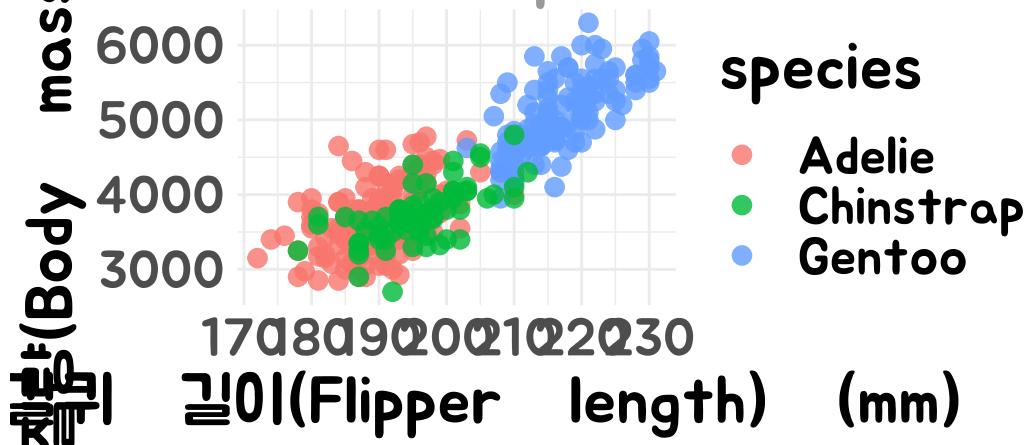
mass_flipper <- ggplot(data = penguins,
  aes(x = flipper_length_mm,
      y = body_mass_g,
      color = species)) +
  geom_point(size = 3,
             alpha = 0.8) +
  labs(title = "팔미 관측소 LTER 서식 펭귄 크기",
       subtitle = "Adelie, Chinstrap, Gentoo 펭귄에 대한 물갈퀴 길이와 체질량",
```

x = "물갈퀴 길이(Flipper length) (mm)",
y = "체질량(Body mass) (g)"

mass_flipper

팔려 관측소 LTER 서식

Adelie, Chinstrap, Gentoo 펭귄



10.4 표

문서에 표를 넣게 되면 정보를 효과적으로 요약할 뿐만 아니라 가독성을 높이는 중요한 커뮤니케이션 도구 중 하나다. 마크다운(Markdown), gt 패키지, 라텍스는 자료를 표 형식으로 표현할 때 각각의 장점이 있다. 마크다운은 기본적이며 간단한 방법으로, 복잡한 설치나 추가 패키지 없이도 빠르게 내용에 충실한 기본표를 작성할 수 있다. gt 패키지는 그래프 문법을 참고하여 표문법을 충실히 준수하며 데이터에 기반한 표제작에 필요한 거의 모든 기능을 제공한다. 라텍스는 가장 전문적인 표를 만들 수 있는 도구로, 논문이나 학술 자료에 적합한 고품질의 표를 제작하는데 유용하다.

마크다운 표

	mpg	cyl	disp
Pontiac Firebird	19.2	8	400.0
Lotus Europa	30.4	4	95.1
Merc 450SL	17.3	8	275.8
Merc 450SLC	15.2	8	275.8
AMC Javelin	15.2	8	304.0



gt 표



mtcars 데이터셋 표		
디자인 글쓰기 gt 패키지		
mpg	cyl	disp
18.7	8	360
13.3	8	350
15.8	8	351
10.4	8	460
21.4	6	258

표제작: 한국 R 사용자회 오픈소스 소프트웨어 개발자

LATEX 표

Car Model	mpg	cyl	disp
Lotus Europa	30.4	4	95.1
Merc 230	22.8	4	140.8
Fiat 128	32.4	4	78.7
Pontiac Firebird	19.2	8	400.0
Merc 450SL	17.3	8	275.8

mtcars 데이터셋

그림 10.7: 표제작 세가지 방식

10.4.1 마크다운 표

mtcars 데이터프레임에서 시연목적으로 관측점 5개와 열 4개를 뽑아내서 kable() 함수에 넣고 출력형식을 markdown으로 지정한다. markdown 외에도 html, pandoc, latex, rst 등으로 출력형식을 지정할 수 있다.

```
library(tidyverse)
library(knitr)

mtcars %>%
  sample_n(5) %>%
  select(mpg, cyl, disp) %>%
  kable("markdown")
```

	mpg	cyl	disp
Volvo 142E	21.4	4	121.0
Lincoln Continental	10.4	8	460.0
Valiant	18.1	6	225.0
Fiat 128	32.4	4	78.7
Merc 450SL	17.3	8	275.8

10.4.2 gt

그래프 문법(grammar of graphics)처럼 표 문법(grammar of table) **gt** 팩키지가 등장했다. 표를 분해하면 다음과 같고 이를 `tibble` 혹은 데이터프레임을 입력받아 GT 객체로 변환시킨 후에 GT 표를 HTML로 출력하는 작업흐름을 갖는다.

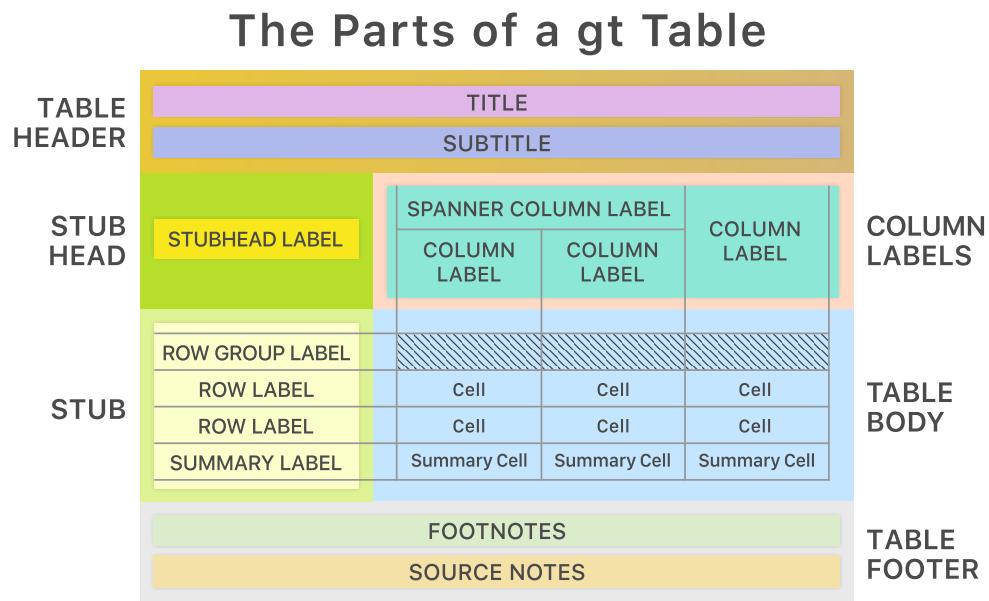


그림 10.8: 표 해부도

gt 팩키지에 내장된 `exibble` 데이터셋은 8×9 구조를 갖는 단순한 데이터셋이지만 표와 관련된 다양한 형태를 개발에 사용할 주요한 정보를 담고 있다.¹

```
library(gt)

mtcars %>%
  sample_n(5) %>%
  select(mpg, cyl, disp) %>%
  gt() |>
  tab_header(
```

¹ Richard Iannone (2020-04-08), “Great Looking Tables: gt (v0.2)”

```

title = md("##mtcars## 데이터셋 표"),
subtitle = md("디지털 글쓰기 `gt` 패키지")
) %>%
tab_source_note("표제작: 한국 R 사용자회 오픈소스 소프트웨어 개발자")

```

mtcars 데이터셋 표 디지털 글쓰기 gt 패키지

mpg	cyl	disp
15.2	8	275.8
14.7	8	440.0
32.4	4	78.7
18.7	8	360.0
13.3	8	350.0

표제작: 한국 R 사용자회 오픈소스 소프트웨어 개발자

10.4.3 라텍 표

라텍은 수식 뿐만 아니라 복잡하고 미려한 표도 코드를 통해 제작할 수 있다. 쿼토에서 라텍 표를 제작하기 위해서는 먼저 표제작을 위한 라텍 표코드를 준비하고, GitHub에서 팬독-필터 `tarleb/parse-latex`를 다음 명령어로 설치한다.

```
$ quarto install extension tarleb/parse-latex
```

쿼토 확장팩이 설치되면 YAML에 다음과 같이 `parse-latex`을 등록하여 사용한다.

```

---
filters:
  - parse-latex
---
```

```
\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|}
\hline
Car Model & mpg & cyl & disp \\
\hline
Lotus Europa & 30.4 & 4 & 95.1 \\
Merc 230 & 22.8 & 4 & 140.8 \\
Fiat 128 & 32.4 & 4 & 78.7 \\
Pontiac Firebird & 19.2 & 8 & 400.0 \\
Merc 450SL & 17.3 & 8 & 275.8 \\
\hline
\end{tabular}
\caption{mtcars 데이터셋}
\end{table}
```

Car Model	mpg	cyl	disp
Lotus Europa	30.4	4	95.1
Merc 230	22.8	4	140.8
Fiat 128	32.4	4	78.7
Pontiac Firebird	19.2	8	400.0
Merc 450SL	17.3	8	275.8

표 10.3: mtcars 데이터셋

10.4.4 표 제작 사례

펭귄 데이터를 기반으로 `gt` 패키지를 사용하여 표 제목, 칼럼명, 셀값에 다양한 글꼴, 크기, 색상, 굵기를 적용한 사례가 표 ??에 코드로 나와 있다. 미국 웨스 앤더슨 영화 색상 (`Darjeeling1`)을 표에 적용하고, 글꼴로 웹글꼴 구글 글꼴을 활용하여 `gt` 패키지로 다양한 글꼴을 적용하고, `fmt_integer()` 함수를 사용해서 숫자 가독성을 높이기 위해 천단위 구분자를 넣고 부가적인 디자인도 표에 적용한다.

```
library(gt)

basic_theme <- function(data, ...){
  data %>%
    tab_options(
      column_labels.background.color = "#F98400",
      table.font.size = px(12),
      column_labels.font.size = px(20),
      row.striping.background_color = "#F2AD00",
      heading.align = "left",
      heading.title.font.size = px(30)
    )
}

palmerpenguins::penguins %>%
  select( 펭귄종=species, 섬=island, bill_length_mm, body_mass_g) %>%
  slice_sample(n = 10) %>%
  gt() %>%
  tab_header(title = md("팔мер 펭귄 표본 **10 마리** 측정 정보")) %>%
  tab_source_note(source_note = "데이터: Kristen Gorman 박사와 남극 팔머 관측소
  ↵ (Palmer Station, Antarctica LTER)에서 수집") %>%
  cols_label(bill_length_mm = "부리 길이 (mm)",
             body_mass_g = "체질량 (g)") %>%
  tab_style(
    style = cell_text(
      font = c(
        google_font(name = "Nanum Pen Script"),
        default_fonts()
      ),
      size = "xx-large",
      color = "gray35",
    )
  )
}
```

```
    weight = "bolder"
),
locations = cells_body(columns = bill_length_mm)
) %>%
# 체질량 칼럼 -----
tab_style(
  style = cell_text(
    font = c(
      google_font(name = "Black Han Sans"),
      default_fonts()
    ),
    size = "large",
    color = "#00A08A",
    weight = "bolder"
),
locations = cells_body(columns = body_mass_g)
) %>%
fmt_integer(body_mass_g) |>
# 표 Header 와 첫번째 칼럼 -----
tab_style(
  style = cell_text(
    font = google_font("Do Hyeon"),
    size = "large"
),
locations = list(
  cells_column_labels(everything()),
  cells_body(columns = 1)
)
) %>%
# 표 제목 -----
tab_style(
```

```

style = cell_text(
  font = google_font("Dokdo"),
  align = "left",
  size = "xx-large",
  color = "#FF0000"
),
locations = cells_title("title")
) %>%
basic_theme()

```

순	용어	대응 영어	
1	앞표지	front cover	책의 맨 곁장. 속장을 보호하기 위해 책 바깥쪽에 부착되는 표지.
2	표제	title	
3	책섶	gusset	책의 양쪽 표지 사이에 있는 공간.
4	모서리	corner	앞표지의 상하 귀퉁이. 양장본에서는 천이나 종이로 둘러싸여 있다.
5	표지턱	square	책 본문 속에 있는 표지.
6	책꼬리	tail edge	
7	도랑	groove	양장본의 두꺼운 앞뒤 표지 사이에 있는 허름한 공간.
8	머리챙	awning/visor	양장본의 책등에 부착되는 투명한 챙.
9	등글자	back title	
10	등턱	joint	양장 제본에서 책의 양쪽 표지 사이에 있는 허름한 공간.
11	책등	spine/back	책 앞쪽 마구리의 반대 뒤쪽 부분.
12	책목	book neck	
13	갈피끈	book-mark/ tassel/ ribbon/spin	책장에 걸어두는 책 표지에 부착되는 장식.

팔며 펭귄 표본 10 마리 측정

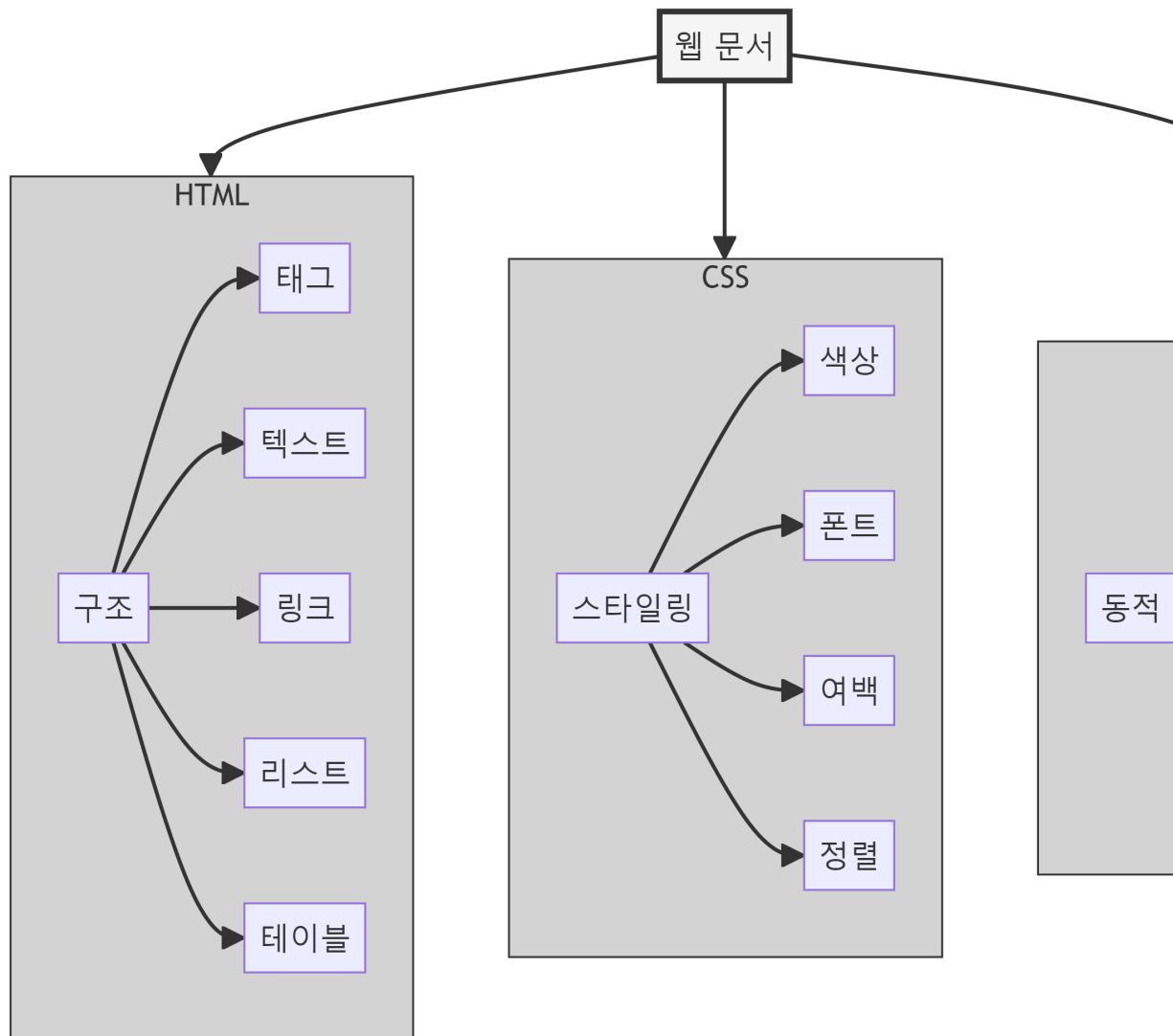
펭귄종	섬	부리 길이 (mm)
Gentoo	Biscoe	43.5
Adelie	Torgersen	36.7
Gentoo	Biscoe	51.1
Gentoo	Biscoe	46.2

Adelie	Dream	37.3
Adelie	Torgersen	35.9
Adelie	Biscoe	40.5
Adelie	Dream	39.6
Gentoo	Biscoe	51.5
Gentoo	Biscoe	47.3

데이터: Kristen Gorman 박사와 남극 팔머 관측소(Palmer Station, Antarctica LTER)에서 수집

10.5 웹문서

웹 문서 구성요소는 크게 HTML, CSS, 자바스크립트로 구분된다. HTML(HyperText Markup Language)은 웹 페이지의 기본 구조와 내용을 정의하고, 다양한 태그를 사용하여 텍스트, 이미지, 링크, 리스트, 테이블 등을 웹 페이지에 배치한다. 주요 HTML 태그로 <head>, <body>, <h1>, <p>, <a>, 등을 들 수 있다. CSS(Cascading Style Sheets)는 웹 페이지 레이아웃과 디자인을 담당한다. 색상, 폰트, 여백, 정렬 등을 지정하여 웹 페이지의 외관을 꾸미는 역할을 한다. CSS는 HTML 문서 내에 <style> 태그를 사용하여 삽입할 수 있고, 외부 저장된 CSS 파일을 연결하여 사용할 수도 있다. 자바스크립트는 웹 페이지에 동적인 기능을 부여한다. 사용자와 상호작용, 데이터 처리, 동적인 요소 변경 등을 담당하고, HTML 문서 내에 <script> 태그를 사용하여 삽입할 수 있다.



웹 문서는 HTML, CSS, 자바스크립트를 사용하여 복잡한 구조와 디자인, 인터랙티브한 요소를 제공하기 위해서 개발이 필요한 반면에 아래한글과 워드 같은 워드 프로세서는 GUI 기반 문서 편집기로 사용자가 메뉴와 버튼을 통해 쉽게 문서를 서식을 지정하여 작성한다. 마크다운은 두가지 문서 저작방식의 장점을 취해 가장 단순한 형태 텍스트 기반 마크업 언어로 특수 문자를 사용하여 서식을 지정한다.

웹 문서			
기능/저작 도구	(HTML/CSS)	아래한글/워드	마크다운
텍스트 입력	HTML 태그를 사용	GUI를 통해 입력	일반 텍스트 입력
서식 지정	CSS 사용	메뉴에서 선택	특수 문자 사용
이미지 삽입	 태그 사용	드래그 앤 드롭	![alt](url) 형식
표 작성	<table> 태그 사용	표 메뉴 사용	파이프(), 하이픈(-) 사용
하이퍼링크	<a> 태그 사용	하이퍼링크 메뉴 사용	[text](url) 형식
문서 구조	HTML 태그 사용	스타일과 목차 사용	# 사용
반응형 디자인	미디어 쿼리 사용	없음	없음
인터랙티브 요소	자바스크립트 사용	매크로 사용	없음
배포	웹 서버 사용	.hwpx 파일 배포	.md 파일 저장
접근성	ARIA 사용	제한적	없음

10.6 다이어그램

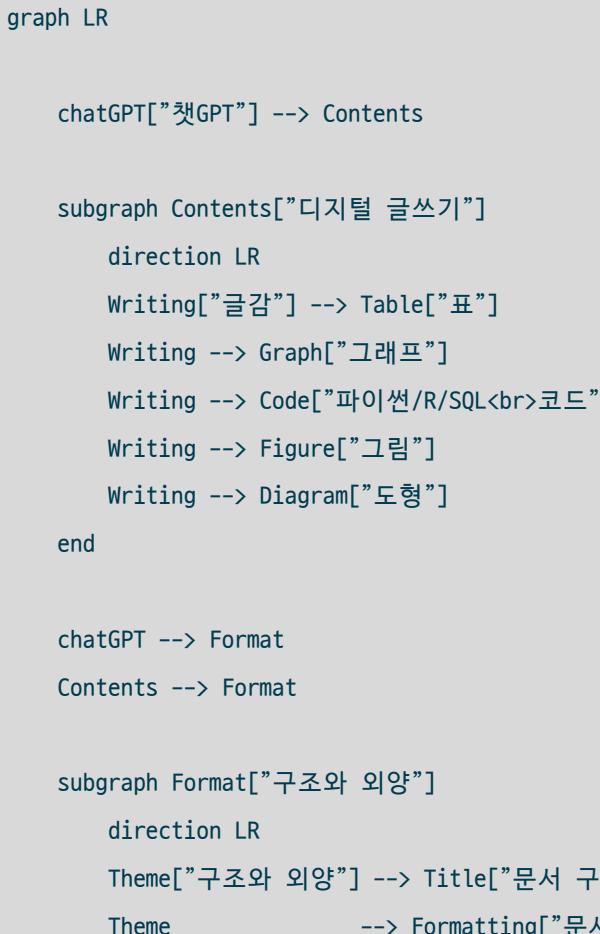
다이어그램(diagram, 도표)는 복잡한 정보나 개념을 단순화하고 시각적으로 표현하여 텍스트만으로 설명하기 어려운 복잡한 데이터나 구조, 관계를 명확히 전달할 수 있어 독자가 문서를 빠르고 정확하게 이해하는 데 도움을 준다. 학술 논문, 보고서, 설명서 등에 포함된 다이어그램은 문서 전문성과 신뢰성을 높혀 효과적인 커뮤니케이션을 가능하게 한다.

다이어그램 저작 방식은 도구를 직접 사용하는 방식과 프로그래밍 언어처럼 다이어그램 전용 언어를 사용하는 방식으로 나뉜다. 잘 알려진 다이어그램 저작 도구로 마이크로소프트 비지오(Microsoft Visio), 루시드차트(Lucidchart), 드로아이오(Draw.io) 등으로 주로 GUI를 통해 다이어그램을 직관적으로 그릴 수 있고 다이어그램 전용 언어로 그래프비즈(Graphviz)와 머메이드(Mermaid)가 대표적이다. Graphviz는 DOT 언어를 사용해 다양한 네트워크 다이어그램, 플로우 차트, 조직도, 의존성 그래프를 제작하는 반면, Mermaid는 마크다운과 유사한 문법으로 웹에서 다이어그램과 차트를 생성하는 자바스크립트 라이브러리로 HTML과 자바스크립트를 사용하는 모든 웹 페이지나 앱과 쉽

게 통합된다는 장점이 있다.

챗GPT를 활용한 디지털 글쓰기 과정을 Mermaid 코드를 이용하여 다이어그램으로 표현하면 다음과 같다. 다이어그램은 글쓰기, 구조와 외양, 배포와 공유 세가지 영역에서 챗GPT가 부기장(Copilot)으로 역할을 수행하는 것을 명확히 보여준다. 디지털 글쓰기가 글감, 표, 그래프, 코드, 그림, 도형 등 다양한 요소로 구성되며, 구조와 외양에서는 문서의 구조, 서식, 레이아웃, 참조 등이 중요하고, 배포와 공유에서는 다양한 대상, 즉 독자, 저자, 기계에게 문서가 전달되는 과정을 빠르고 명확하게 이해할 수 있도록 도와준다.

비지오 같은 유명 GUI 도구보다 다이어그램 전용 언어를 사용하게 되면 높은 수준 사용자 정의와 자동화가 가능할 뿐만 아니라 복잡한 다이어그램도 정확하게 그릴 수 있고, 코드를 재사용함으로써 시간과 비용도 절약할 수 있다. 특히, 버전 관리 시스템에 쉽게 통합할 수 있어 팀원들과 공유와 협업이 용이하다.

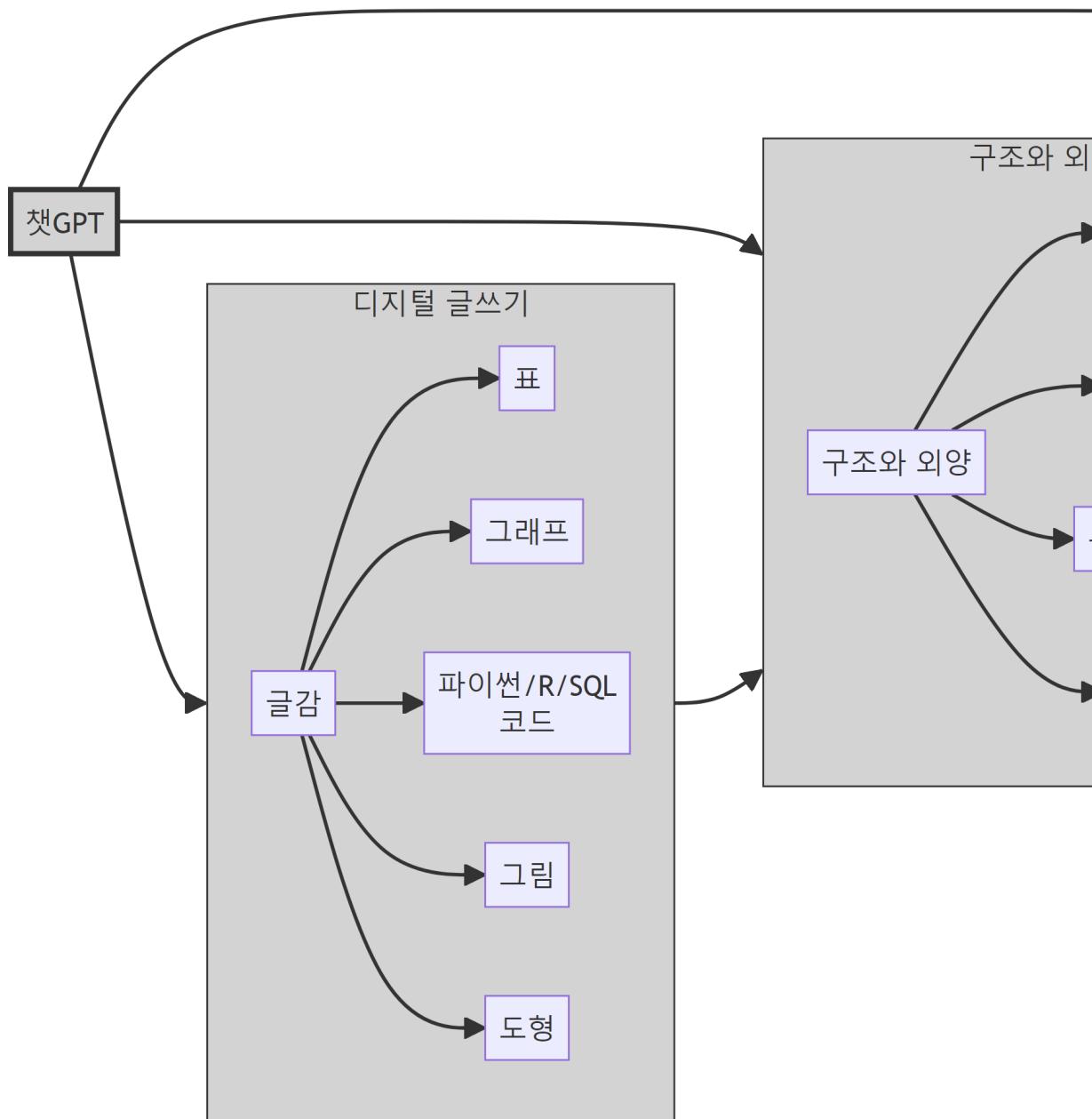


```
Theme          --> Layout[”문서 레이아웃”]
Theme          --> reference[”문서 참조”]
end

chatGPT --> Deployment
Format --> Deployment

subgraph Deployment[”배포와 공유”]
    direction LR
    Deploy --> Reader[”독자(사람)”]
    Deploy --> Author[”독자(저작자)”]
    Deploy --> Machine[”독자(기계)”]
end

style chatGPT fill:#d3d3d3,stroke:#333,stroke-width:3px
style Contents fill:#d3d3d3,stroke:#333,stroke-width:1px
style Format fill:#d3d3d3,stroke:#333,stroke-width:1px
style Deployment fill:#d3d3d3,stroke:#333,stroke-width:1px
```



10.7 수식

수학은 공학, 물리학, 컴퓨터 과학, 경제학 등 다양한 분야에서 중심 역할을 하는 학문으로 정확하고 명료한 수학적 표기는 매우 중요하다. 라텍은 수학적 표현을 깔끔하고