R 과 Quarto 를 이용한 데이터 테이블 작성

서울시립대학교 빅데이터 혁신융합대학 사업단

서울시립대학교 통계학과 이용희, 정영두

2025-01-29

목차

그림 목록

표 목록

개요

목표 및 개요

내용 요약 및 범위

- 결측값에 대한 기초 분석과 기본적인 대체방법에 대하여 소개

- R 언어로 보고서용 데이터 테이블을 자동으로 작성하는 방법 소개
- `flextable` 패키지를 이용하여 고급 품질의 테이블 작성 방법 소개
- 테이블을 포함한 보고서를 다양한 형식(MS Word, PDF, MS Excel)으로 생성하는 방법 소개

활용범위 및 기대효과

분석에 필요한 것들

> **!** 필요한 선행 지식
>
> 이 교과서를 공부하기 위해서는 다음과 같은 선행지식이 필요합니다.
>
> 1. R 언어의 기초: 기본적인 프로그래밍의 문법, 패키지 사용, `data.frame` 에 대한 정의 및 기초,
>
> 2. 기초 통계학: 연속형 자료와 범주형 자료에 대한 요약 통계량(평균, 표준편차, 비율 등), 회귀분석의 기초

프로그램

- R: 데이터 사이언스를 위한 컴퓨터 언어
- `Rstudio` : R 언어를 사용하기 위한 GUI
- `Quarto` 는 오픈소스로 공개된 과학기술 분야의 출판 시스템니다 (Allaire et al. (2024) 참고)

R 패키지

이 책애서 사용하는 R 패키지는 다음과 같다.

```
library(here)
library(tidyverse)
library(knitr)
library(DescTools)
library(mice)
library(ggmice)
library(naniar)
library(flextable)
library(broom)
library(sjtable2df)
```

예제 자료

국민체력100 측정자료

- 출처: 문화빅데이터 프랫폼 - 서울올림픽기념국민체육진흥공단
- 데이터 소개
  - 서울올림픽기념 국민체육진흥공단에서 관리하고 있는 국민체력측정데이터의 항목별 측정 정보를 제공하는 데이터
  - 체력측정 센터명, 연령대, 신장, 체중, 윗몸일으키기, BMI, 제자리 멀리뛰기 등의 체력측정 항목별 결과를 조회 가능

2019년 서울특별시 부동산 실거래가 자료

- 출처: 서울 열린데이터 광장
- 데이터 소개
  - 서울 열린 데이터 광장에서 2019년 서울특별시 부동산 실거래가 정보를 가진 자료
  - 2019년 거래하여 신고한 주택들의 위치와 정보 그리고 실거래 가격이 포함된 총 67238건의 거래자료 중 일부 자료와 함성자료

1 효율적인 통계의 생산

1.1 재현 가능한 통계 업무

과학적 연구에서 중요한 원칙 중 하나는 연구 결과의 **재현 가능성(reproducibility)**이다. 이는 동일한 환경과 동일한 방법을 사용하여 연구를 수행할 때, 항상 같은 결과를 얻을 수 있는 특성을 의미한다. 즉, 연구자가 동일한 실험이나 분석을 반복할 경우뿐만 아니라, 다른 연구자가 동일한 절차를 따라 연구를 수행했을 때도 같은 결론에 도달할 수 있어야 한다. 이는 연구의 신뢰성을 확보하는 핵심 요소로 작용한다.

재현 가능성은 특히 통계 업무에서 필수적인 요소이다. 데이터를 분석하고 해석하는 과정에서 일관된 결과를 얻는 것이 중요하며, 이는 연구뿐만 아니라 실무에서도 중요한 가치를 지닌다. 통계 업무에서 재현 가능성이 중요한 이유는 크게 **지속성**과 **효율성**의 두 가지 측면에서 설명할 수 있다.

1.1.1 지속성: 연구와 업무의 연속성 유지

재현 가능성이 확보되면, 시간이 지나거나 담당자가 변경되더라도 유사한 업무를 쉽게 반복적으로 수행할 수 있다. 즉, 특정 분석 작업이 연구자의 개인적인 역량에 의존하지 않고, 체계적인 시스템을 기반으로 수행될 수 있도록 한다. 이를 통해 조직 내에서 안정적으로 데이터를 활용하고 연구를 지속할 수 있으며, 동일한 연구를 반복할 때에도 일관된 결과를 얻을 수 있다.

1.1.2 효율성: 분석 과정의 표준화 및 확장 가능성

재현 가능한 연구 환경에서는 분석 과정이 표준화된 코드로 연결되고 구성되므로, 동일한 절차를 반복하는 데 드는 시간과 노력을 줄일 수 있다. 또한, 연구나 업무의 요구사항이 변화하더라도 기존의 분석 코드나 방법을 쉽게 수정하고 확장할 수 있다. 이는 새로운 데이터가 추가되거나 분석의 방향이 수정될 때도 빠르고 유연하게 대응할 수 있도록 해준다.

더 나아가, 협업 환경에서도 재현 가능성은 큰 장점이 된다. 동일한 코드와 데이터를 공유함으로써 팀원 간의 원활한 협업이 가능하며, 연구나 업무의 일관성을 유지할 수 있다. 이는 조직 내에서 분석 결과의 신뢰도를 높이고, 효율적인 의사 결정을 지원하는 중요한 요소로 작용한다.

재현 가능성은 연구의 신뢰성과 투명성을 보장하는 필수적인 요소이며, 통계 업무에서도 그 중요성이 크다. 지속적으로 업무를 수행할 수 있도록 하며, 효율적인 분석을 가능하게 함으로써 연구자와 실무자 모두에게 유익한 환경을 제공한다. 따라서 연구 및 데이터 분석을 수행하는 모든 과정에서 재현 가능성을 높이는 것이 바람직하다.

1.2 자료의 관리와 통계의 생산

통계를 신뢰할 수 있도록 유지하고 활용하기 위해서는 **원천 자료(source data; master data)**와 **메타정보(meta data)**의 체계적인 관리가 필수적이다. 원천 자료는 모든 통계 생산의 기초가 되는 데이터이며, 메타정보는 해당 자료에 대한 설명과 구조를 정의하는 정보를 의미한다. 이러한 자료를 체계적으로 관리해야 일관된 통계를 생산할 수 있으며, 변화하는 데이터 환경에서도 효율적으로 대응할 수 있다.

1.2.1 체계적인 원천 자료 및 메타정보 관리

통계의 신뢰성과 재현 가능성을 보장하기 위해서는 원천 자료와 이를 설명하는 메타정보가 지속적으로 관리될 필요가 있다. 특히, 원천 자료가 변경될 경우 이를 바탕으로 생성된 중간 자료들도 자동으로 업데이트되는 체계를 갖추는 것이 중요하다. 이를 통해 데이터의 일관성을 유지하고, 새로운 데이터가 추가되더라도 기존의 분석 결과를 손쉽게 갱신할 수 있다. 또한, 메타정보 역시 원천 자료와 동일한 위치에서 지속적으로 관리되어야 한다. 이렇게 하면 데이터가 변하더라도 그 의미와 구조를 일관되게 유지할 수 있어, 분석 과정에서 혼란을 줄이고 효율성을 높일 수 있다.

1.2.2 재현 가능한 통계 생산 체계 구축

통계를 생산하는 과정은 체계적으로 프로그램화되고 연결되어야 하며, 필요에 따라 쉽게 수정할 수 있어야 한다. 즉, 단순히 데이터를 분석하는 것이 아니라, **재현 가능한 통계 생산 체계**를 갖추는 것이 중요하다. 이를 위해 모든 분석 과정이 자동화되고, 변경 사항이 발생했을 때도 별도의 수작업 없이 업데이트될 수 있어야 한다.

그러나 현실에서는 수작업을 기반으로 한 통계 생산 방식이 여전히 많이 사용되고 있다. 예를 들어, 엑셀을 이용한 통계 생산 방식은 타인이 동일한 결과를 재현하기 어려운 경우가 많다. 엑셀 파일 내에서 수식이 숨겨져 있거나, 데이터가 수동으로 수정되는 경우가 발생하기 때문이다. 따라서, 신뢰할 수 있는 통계 시스템을 구축하기 위해서는 엑셀과 같은 개별적인 도구에 의존하는 방식에서 벗어나, **자동화된 분석 코드와 체계적인 데이터 관리 시스템**을 활용하는 것이 필요하다.

통계의 신뢰성과 재현 가능성을 높이기 위해서는 원천 자료와 메타정보를 체계적으로 관리하고, 통계 생산 과정 전체를 자동화된 시스템으로 구축하는 것이 중요하다. 이를 통해 데이터의 변경에도 신속하게 대응할 수 있으며, 분석 과정에서의 오류를 최소화할 수 있다. 궁극적으로 이러한 체계적인 접근 방식은 보다 정확하고 신뢰할 수 있는 통계 생산을 가능하게 한다.

2 결측자료의 발생원인과 유형

이장에서는 결측자료의 발생원인과 유형에 대하여 살펴보고 싱제 자료를 이용한 결측값에 대한 처리를 연습해 본다.에 대한 전반적인 이해를 해보자.

2.1 서론

데이터 분석에서 신뢰할 수 있는 결과를 도출하기 위해서는 정확하고 완전한 데이터를 확보하는 것이 필수적이다. 그러나 현실에서는 다양한 이유로 인해 데이터의 일부가 누락되는 **결측값(missing data)** 문제가 발생한다. 결측값은 데이터 수집, 입력, 저장, 관리 등의 다양한 과정에서 생길 수 있으며, 연구 설계 및 방법론의 한계로 인해 발생하기도 한다. 이러한 결측값은 단순한 누락된 정보 이상의 의미를 가지며, 분석의 신뢰성과 해석 가능성에 큰 영향을 미칠 수 있다.

결측값이 포함된 데이터는 표본의 대표성을 저하시킬 수 있으며, 분석 결과를 왜곡시킬 가능성이 크다. 또한, 결측값을 처리하는 방식에 따라 분석 결과가 달라질 수 있어 연구자가 신중하게 접근해야 한다. 특히, 통계적 검정과 추정의 정확도가 낮아지거나, 데이터 활용의 효율성이 감소하는 등의 문제가 발생할 수 있다. 따라서 결측값이 발생하는 원인을 파악하고, 이에 대한 적절한 처리 방법을 적용하는 것은 데이터 분석에서 필수적인 과정이다.

이 장에서는 결측값이 발생하는 주요 원인을 분석하고, 결측값이 통계 분석에 미치는 영향을 체계적으로 살펴본다. 이를 통해 데이터의 완전성을 유지하고 분석 결과의 신뢰성을 높이는 전략을 모색할 수 있을 것이다.

2.2 결측값 발생 원인과 영향

데이터 분석에서 결측값은 연구의 신뢰성을 저하시킬 수 있는 중요한 요소이다. 결측값이 발생하는 원인은 다양하며, 이는 데이터 수집, 입력 및 처리, 저장 및 관리, 연구 설계 등의 과정에서 나타날 수 있다. 또한, 결측값은 통계 분석의 정확성과 신뢰성에도 영향을 미칠 수 있어 이에 대한 적절한 이해와 처리가 필수적이다.

2.2.1 결측값 발생 원인

결측값은 데이터가 정상적으로 수집·입력·저장되지 못한 경우 발생하며, 그 원인은 다음과 같이 네 가지 주요 과정에서 찾을 수 있다.

데이터 수집 과정의 문제

데이터가 처음 수집되는 단계에서 결측값이 발생할 수 있다.

- 설문조사 응답 누락: 조사 대상자가 질문에 답변하지 않는 경우 발생하며, 이는 의도적인 응답 회피, 무지, 무관심 등의 요인으로 인해 발생할 수 있다.

- 기술적 문제: 장비의 고장이나 네트워크 장애로 인해 데이터가 정상적으로 기록되지 않는 경우도 있다.

데이터 입력 및 처리 과정의 문제

데이터가 입력되거나 변환되는 과정에서 실수나 오류로 인해 결측값이 발생할 수 있다.

- 데이터 입력 오류: 수작업으로 데이터를 입력할 때 실수가 발생하거나, 자동 입력 시스템에서 오류가 발생할 수 있다.
- 데이터 처리 및 변환 오류: 데이터 변환 과정에서 적절한 값이 입력되지 않거나, 오류로 인해 일부 값이 손실될 수 있다.

데이터 저장 및 관리 과정의 문제

데이터가 저장되고 관리되는 과정에서도 결측값이 발생할 가능성이 있다.

- 데이터베이스 관리 문제: 저장 공간의 제한으로 인해 일부 데이터가 삭제되거나, 데이터 정리 과정에서 중요한 값이 누락될 수 있다.
- 백업 및 복원 문제: 백업 과정에서 일부 데이터가 저장되지 않거나, 복원 시 손실되는 경우도 결측값의 원인이 된다.

연구 설계 및 방법론의 문제

연구 설계 자체의 한계로 인해 데이터를 충분히 확보하지 못하는 경우도 있다.

- 연구 설계의 불완전성: 중요한 변수가 설계에서 누락되었거나, 질문이 부적절하게 구성되었을 경우 특정 값이 수집되지 못할 수 있다.
- 데이터 수집 방법의 한계: 표본 설계에서 오류가 발생하면 특정 그룹의 데이터가 충분히 확보되지 못하여 결측값이 발생할 가능성이 있다.

2.2.2 결측값이 통계 분석에 미치는 영향

결측값은 통계 분석의 신뢰성을 저하시킬 수 있으며, 분석 결과의 해석과 활용에 여러 가지 문제를 야기할 수 있다. 결측값이 미치는 주요 영향은 다음과 같다.

통계적 추정의 정확성 저하

- 표본 크기의 감소: 결측값을 포함한 데이터를 제거하면 전체 표본 크기가 줄어들어 통계적 정확도가 낮아진다. 이는 분석 결과의 신뢰도를 저하시킬 수 있으며, 표본이 모집단을 대표하는 특성을 잃을 가능성이 있다.
- 왜곡된 추정: 결측값이 특정한 패턴을 가지고 나타날 경우 분석 결과가 편향될 수 있다. 예를 들어, 특정 그룹에서 결측값이 더 많이 발생한다면, 분석 결과가 해당 그룹의 특성을 반영하지 못할 수 있다.

분석 결과의 해석 어려움

- 해석의 모호성: 결측값을 어떻게 처리하느냐에 따라 분석 결과가 달라질 수 있으며, 이는 연구자가 분석 결과를 해석하는 데 어려움을 줄 수 있다.
- 일반화의 제한: 특정 그룹에서 결측값이 집중적으로 발생했다면, 연구 결과를 전체 모집단에 적용하기 어려울 수 있다.

데이터 분석의 효율성 감소

- 데이터 활용 제한: 결측값이 많으면 사용할 수 있는 데이터의 양이 줄어들고, 특히 머신러닝 모델과 같은 고급 분석 기법을 적용하는 것이 어려워질 수 있다.
- 분석 과정의 복잡성 증가: 결측값을 처리하기 위해 보간법(imputation), 대체 값 적용 등의 추가적인 작업이 필요하며, 이로 인해 분석 과정이 복잡해지고 시간이 더 소요될 수 있다.

통계적 검정의 신뢰성 저하

- 유효성 저하: 결측값이 많을 경우 수행하는 통계적 검정의 결과가 신뢰할 수 없는 수준이 될 수 있다. 이는 잘못된 결론을 초래할 위험이 있다.
- 추정량의 분산 증가: 결측값을 대체하는 방식에 따라 데이터의 분산이 증가할 수 있으며, 이는 통계적 검정의 정확도를 낮추고 결과의 신뢰성을 저하시킬 수 있다.

결측값은 데이터 분석 과정에서 피할 수 없는 문제이며, 이를 적절히 처리하지 않으면 연구의 신뢰성과 분석 결과의 정확성이 크게 저하될 수 있다. 따라서 데이터 수집, 입력, 저장, 관리 및 분석 과정에서 결측값을 최소화하기 위한 전략을 마련하는 것이 필요하다. 또한, 결측값이 발생했을 때 이를 올바르게 처리하는 방법을 적용하여 통계 분석의 신뢰성과 활용도를 높이는 것이 중요하다.

2.3 결측값의 분포와 패턴

데이터 분석에서 결측값을 효과적으로 처리하기 위해서는 먼저 **결측값의 분포와 패턴**을 정확히 파악하는 과정이 필요하다. 단순히 결측값이 존재하는지 여부만 확인하는 것이 아니라, 결측값이 어떻게 분포되어 있으며 특정 변수 간에 어떤 관계를 형성하는지 분석하는 것이 중요하다. 이를 통해 결측값이 데이터에 미치는 영향을 평가하고, 적절한 보완 및 대체 방법을 결정할 수 있다.

2.3.1 결측값 분석의 중요성

데이터에 결측값이 포함되어 있을 경우, 이를 무작정 삭제하거나 단순한 대체 값으로 채우는 것은 적절하지 않을 수 있다. 왜냐하면 결측값의 발생 원인과 분포 형태에 따라 분석 결과에 미치는 영향이 다를 수 있기 때문이다. 따라서 결측값을 처리하기 전에 반드시 결측값의 분포와 구조를 면밀히 분석해야 한다.

결측값이 특정 변수에 집중적으로 발생하는지, 여러 변수에서 동시에 나타나는지, 결측값이 일정한 패턴을 가지고 있는지 등의 요소를 파악하는 것은 이후 분석의 정확성을 높이는 데 중요한 역할을 한다.

2.3.2 결측값의 분포와 패턴 분석 방법

결측값의 구조를 이해하기 위해서는 여러 가지 분석 기법을 활용할 수 있다. 대표적인 방법은 다음과 같다.

변수별 결측값 빈도 및 비율 분석

먼저, 데이터의 각 변수에서 결측값이 얼마나 발생했는지를 파악해야 한다. 이를 위해 변수별로 **결측값의 개수와 비율**을 계산하는 것이 기본적인 방법이다. 특정 변수에서 결측값이 과도하게 많다면 해당 변수를 분석에 포함할 것인지, 대체 값을 사용할 것인지 등을 고려해야 한다.

여러 변수에서의 결측값 위치 및 패턴 분석

데이터에 존재하는 여러 변수들 간의 결측값 분포를 확인하는 것도 중요하다. 특정 변수에서 결측값이 발생할 때 다른 변수에서도 동일한 패턴으로 결측값이 발생하는지 확인하는 과정이 필요하다. 이를 통해 **결측값이 무작위(random missing)**인지, 아니면 특정한 규칙을 가지고 있는지 판단할 수 있다.

변수 간 결측값의 관계 분석

일부 변수에서 발생한 결측값이 다른 변수의 결측값과 관련되어 있을 수 있다. 예를 들어, 특정 연령대에서 설문 응답이 누락되는 패턴이 있다면, 연령 변수와 설문 응답 변수 간의 연관성을 고려해야 한다. 이를 파악하기 위해 변수 간의 결측값 관계를 분석하는 것이 필요하다.

2.4 결측값 대체

데이터 분석에서 결측값을 적절히 처리하는 것은 연구의 신뢰성을 보장하는 중요한 요소이다. 결측값을 방치하면 분석 결과의 왜곡을 초래할 수 있으며, 잘못된 결론으로 이어질 위험이 크다. 하지만 모든 결측값이 동일한 방식으로 처리될 수 있는 것은 아니다. 데이터의 특성과 결측값 발생 패턴에 따라 적절한 처리 방법을 신중히 선택해야 한다.

결측값을 처리하는 다양한 방법이 존재하지만, 특정한 상황에서 반드시 사용해야 하는 고정된 방법은 없다. 그렇기 때문에 연구자는 전반적인 결측값 처리 기법을 숙지한 후, 데이터의 특성과 분석 목적에 맞는 방법을 직접 선택하는 것이 바람직하다.

> **!** 결측값을 대체하는 것보다 중요한 것은 예방이다
>
> 데이터 분석에서 결측값을 처리하는 다양한 방법이 존재하지만, 보다 근본적으로 중요한 것은 애초에 결측값이 발생하지 않도록 하는 것이다. 결측값이 발생한 후 이를 보완하는 작업은 필연적으로 분석의 신뢰성을 저하시킬 위험이 있으며, 데이터의 원래 정보를 온전히 복원하기 어렵다. 따라서 결측값을 줄이기 위한 사전 예방 조치가 무엇보다 중요하며, 이는 시간과 비용을 투자할 가치가 있는 과정이다.
> 많은 연구와 실무에서 결측값이 발생하면 이를 삭제하거나 평균값, 회귀 모델 등을 활용하여 대체하는 방식으로 해결하려 한다. 하지만 어떤 방법을 사용하든, 결측값을 대체한 데이터는 원본 데이터와 동일한 정보량을 갖지 못한다는 점을 간과해서는 안 된다. 결측값 대체 과정에서 데이터의 변동성이 줄어들거나, 특정 패턴이 왜곡될 수 있으며, 결과적으로 분석의 신뢰성과 예측력도 낮아질 가능성이 크다.

결측값을 최소화하는 것이 중요하다. 하지만 현실적으로 결측값이 발생하는 경우, 다양한 보완 방법을 활용하여 데이터를 최대한 보존하는 방향으로 처리해야 한다.

- 삭제법 (Deletion Method)
  - 가장 직관적인 방법은 결측값이 포함된 데이터를 제거하는 것이다. 여기에는 두 가지 방식이 있다.
    - 행 삭제 (Listwise Deletion): 결측값이 포함된 레코드(행)를 제거하는 방식
    - 열 삭제 (Variable Deletion): 특정 변수(열)에서 결측값이 많을 경우, 해당 변수를 분석에서 제외하는 방식
    - 이 방법은 데이터가 완전 무작위로 결측되는 경우(MCAR; Missing Completely At Random) 유용할 수 있다. 그러나 데이터 손실이 크며, 표본 크기가 급격히 줄어들 수 있다는 단점이 있다.
- 단순 대체법 (Simple Imputation)
  - 결측값을 특정한 값으로 대체하는 방식으로, 비교적 간단하고 빠르게 적용할 수 있다.

  - Hot Deck 대체법: 결측된 레코드를 동일한 특성을 가진 다른 레코드의 값으로 대체
  - 평균(중앙값) 대체: 연속형 변수의 결측값을 해당 변수의 평균(mean) 또는 중앙값(median)으로 대체
  - 최빈값(Mode) 대체: 범주형 변수의 경우, 가장 많이 나타난 값(최빈값, mode)으로 대체
  - 이 방법은 계산이 쉽고 빠르지만, 데이터의 분산을 감소시키고 편향을 초래할 가능성이 있다는 한계가 있다.
- 예측 모델 기반 대체 (Model-Based Imputation)
  - 보다 정교한 접근법으로, 회귀 분석이나 거리 기반 알고리즘을 활용하여 결측값을 예측하는 방법이다.
  - 회귀 대체법 (Regression Imputation): 다른 변수들을 활용하여 회귀 모델을 생성하고, 이 모델을 사용해 결측값을 예측하여 대체
  - 거리 대체법 (Nearest Neighbor Imputation): 결측값이 없는 데이터 포인트와 가장 유사한 레코드를 찾아 해당 값을 대체
  - 이 방법은 단순 대체법보다 데이터의 패턴을 더 잘 반영할 수 있지만, 모델이 올바르게 설정되지 않으면 추정값이 실제 데이터와 다르게 왜곡될 위험이 있다.
- 다중 대체법 (Multiple Imputation, MI)
  - 다중 대체법은 결측값을 여러 번 대체하여 데이터의 불확실성을 반영하는 방법이다.
  - 동일한 결측값에 대해 여러 번 대체 수행 (예: 5회, 10회)
  - 각 대체본(imputed dataset)은 결측값이 서로 다른 값으로 대체된 별도의 데이터셋
  - 최종 분석 시 여러 개의 대체본을 통합하여 결측값 처리의 불확실성을 반영
  - 이 방법은 결측값을 처리하는 가장 발전된 기법 중 하나로, 통계적 추론에서 보다 신뢰할 수 있는 결과를 제공한다. 다만, 계산 과정이 복잡하고 추가적인 연산이 필요하다는 점이 단점이다.

결측값 처리 방법은 데이터의 특성과 분석 목적에 따라 신중하게 선택해야 한다. 무조건적인 삭제는 데이터 손실을 초래할 수 있으며, 단순 대체법은 분석 결과를 왜곡할 가능성이 있다. 보다 정교한 예측 모델 기반 대체법이나 다중 대체법을 활용하면 데이터의 패턴을 보존하면서도 결측값 문제를 효과적으로 해결할 수 있다.

궁극적으로 중요한 것은 결측값이 분석 결과에 미치는 영향을 최소화하는 방향으로 처리 전략을 설정하는 것이다. 연구자는 다양한 방법을 이해하고, 데이터의 특성을 고려하여 최적의 접근법을 선택하는 것이 필요하다.

2.5 실습: 결측값의 검토와 패턴 분석

다양한 라이브러리를 활용하면 결측값의 위치와 패턴을 직관적으로 확인할 수 있으며, 이를 통해 결측값이 무작위로 발생한 것인지, 특정 그룹이나 변수에 집중적으로 나타나는지를 분석할 수 있다. 이러한 분석은 결측값 처리 방법을 결정하는 중요한 기초 자료가 되며, 분석 결과의 신뢰성을 높이는 데 기여한다.

결측값을 효과적으로 처리하기 위해서는 먼저 데이터 내에서 결측값이 어떻게 분포하고 있는지를 분석하는 과정이 필수적이다. 변수별 결측값의 빈도와 비율을 분석하고, 여러 변수 간 결측값의 위치와 패턴을 살펴봄으로써 결측값이 연구 결과에 미칠 영향을 최소화할 수

있다. 이를 위해 적절한 통계 프로그램과 시각화 기법을 활용하면 보다 정확한 결측값 분석이 가능하며, 이후의 데이터 처리 및 분석 과정에서 신뢰성을 확보할 수 있다.

2.5.1 결측값 패턴 분석을 위한 도구

결측값의 분포와 패턴을 보다 효과적으로 분석하기 위해서는 통계 프로그램 및 시각화 도구를 활용하는 것이 유용하다. 이를 통해 결측값이 무작위로 발생한 것인지, 특정 패턴을 가지고 있는지 확인할 수 있으며, 이후 결측값을 보완하는 전략을 수립하는 데 도움을 줄 수 있다.

R에서는 결측값의 분석에 유용한 다음과 같은 패키지들이 있다.

- `DescTools` 패키지
  - 다양한 기초 통계 분석의 방법을 제공하는 패키지
  - `PlotMiss` 함수를 사용하여 결측값의 위치와 비율을 시각화할 수 있다.
- `ggmice` 패키지 +`plot_pattern` 함수를 사용하여 결측값의 패턴을 시각화할 수 있다.
- `mice` 패키지
  - `mice` 함수를 사용하여 결측값을 대체한 자료를 생성
- `naniar` 패키지
  - 체계적이며 간결한 방법으로 결측값에 대한 요약, 시각화, 대체를 할 수 있는 다양한 기능을 제공한다.

Python에서는 `pandas` 라이브러리를 이용하여 결측값의 빈도와 비율을 쉽게 확인할 수 있으며, `missingno` 라이브러리를 활용하면 결측값의 분포를 시각적으로 분석할 수 있다.

2.5.2 예제 자료

서울 열린 데이터 광장에서 제공하는 2019년 서울특별시 부동산 실거래가 정보를 가진 자료를 어번 예제에 사용하려고 한다. 2019년 거래하여 신고한 주택들의 위치와 정보 그리고 실거래 가격이 포함된 총 67238건의 실제 거래자료는 거래된 주택에 대한 실거래가 등 다양한 정보를 포함하고 있다.

본 예제에서는 실습을 위하여 원래 자료의 일부를 분석할 것이며 교육을 위하여 결측값의 유형이 서로 다른 합성 자료(synthetic data)도 사용할 것이다. 다음은 이 절에서 사용할 예제 자료의 특성은 다음과 같다.

- 예제 자료에서는 이미 지어진 아파트의 거래만 선택(분양 아파트 제외)
- 아파트의 거래에서 1000건의 자료만 임의로 선택
- 거래가격의 단위는 백만원으로 변경
- 결측값이 없는 원자료와 결측값의 유형이 다른 3개의 합성 자료로 구성
- 예제에서 사용할 자료는 다음과 같은 6개의 컬럼을 가지고 있다.
  - 실거래가아이디
  - 건축년도 (주택이 지어진 년도)
  - 자치구명
  - 건물면적 (제곱미터)
  - 층정보

- 물건금액 (거래 가격, 백만원)

이제 예제 자료를 포함한 R 데이터를 불러오자.

```
load(here::here("data", "missing_data.RData"))
ls()
```

```
[1] "df_full"   "df_miss_1" "df_miss_2" "df_miss_3"
```

결측값이 없는 예제 자료 df_full 의 일부는 아래와 같다. 결측값이 존재하는 다른 3개의 자료(df_miss_1, df_miss_2, df_miss_3)는 결측값이 없는 예제 자료 df_full 에서 결측값이 특정한 패턴을 가지고 발생하도록 만든 합성자료이다.

```
knitr::kable(df_full %>% head(10))
```

표 2.1: 결측값이 없는 예제 자료

실거래가아이디	자치구명	건축년도	건물면적	층정보	물건금액
1	송파구	2007	59.88	12	1330
2	용산구	2004	126.19	21	1560
3	성동구	2000	59.96	12	750
4	서대문구	2004	84.88	5	400
5	성동구	2016	84.64	6	1300
6	중랑구	1997	84.08	16	450
7	광진구	2001	84.91	12	1025
8	송파구	2006	135.82	30	2332
9	성북구	2008	84.83	4	620
10	성북구	1994	84.90	9	505

결측값이 있는 자료 df_miss_1 의 일부는 다음과 같으며 결측값은 NA 로 표시된다.

```
knitr::kable(df_miss_1 %>% head(10))
```

표 2.2: 결측값이 있는 예제 자료

실거래가아이디	자치구명	건축년도	건물면적	층정보	물건금액
1	송파구	2007	59.88	NA	NA
2	용산구	2004	126.19	21	1560
3	성동구	2000	NA	12	750
4	서대문구	2004	84.88	5	400
5	성동구	2016	84.64	6	1300
6	중랑구	1997	84.08	16	450
7	광진구	2001	NA	12	1025
8	송파구	NA	135.82	30	2332
9	성북구	2008	84.83	4	620
10	성북구	1994	84.90	9	505

2.5.3 결측값의 검토

먼저 결측값이 있는 자료 df_miss_1 에서 결측값의 위치를 시각적으로 살펴보기 위하여 다음과 같이 PlotMiss 함수를 사용해 보자.

```
PlotMiss(df_miss_1, main="결측값의 위치와 비율")
```

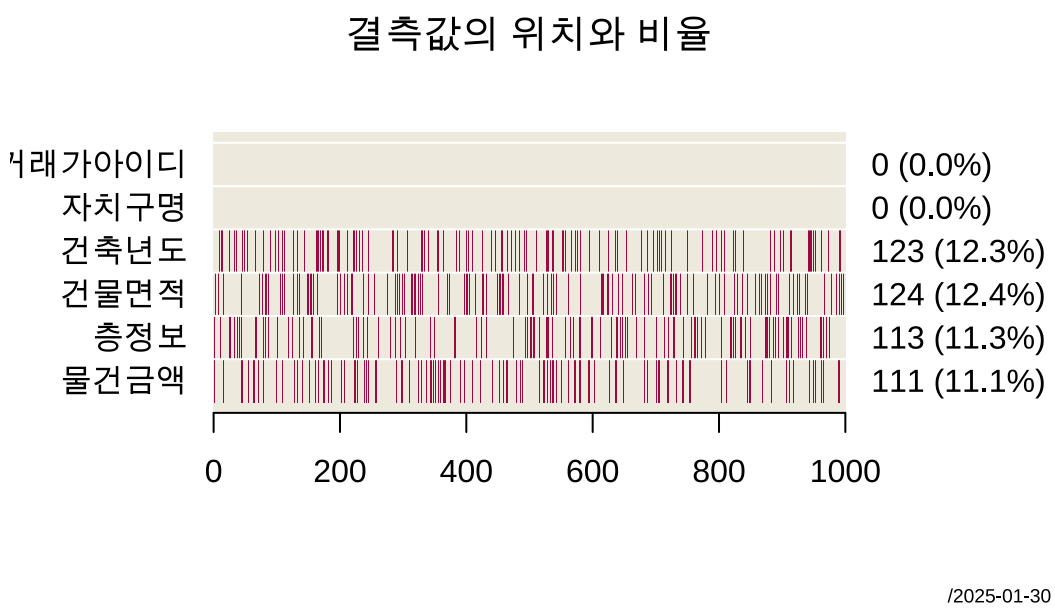


그림 2.1: 결측값의 위치와 비율 - df_miss_1

그림 2.1 을 보면 4개의 변수에 결측값이 존재하는 자료이며 그림 오른쪽에 각 변수에 존재하는 결측값의 개수외 비율이 나타난다.

이제 plot_pattern 함수를 이용하여 결측가 발생하는 패턴에 대해서 살펴보자

```
plot_pattern(df_miss_1, square= FALSE)
```

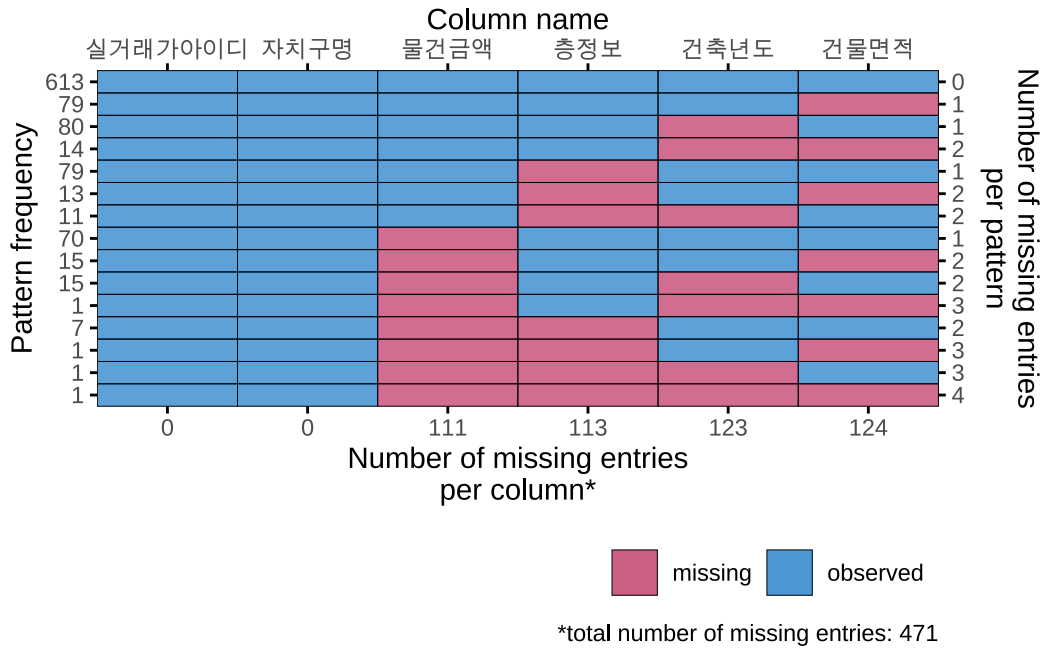


그림 2.2: 무작위로 발생한 결측값

그림 2.2 의 각 행은 결측값이 나타난 유형을 알려주며, 오른쪽의 숫자는 주어진 패턴으로 나타난 결측된 컬럼의 개수이다. 왼쪽은 결측값이 나타난 유형에 속하는 자료의 개수(pattern frequency)이다. 아래 쪽에 나타난 숫자는 각 컬럼에 나타나는 결측값의 개수이다.

그림 2.2 에 나타나 결측값의 발생 형태를 보면 특별한 경향이 없이 4개의 변수에서 무작위로 발생한 것으로 판단된다.

다음으로 결측값이 있는 자료 `df_miss_2` 에서 결측값의 위치를 시각적으로 살펴보자.

```
PlotMiss(df_miss_2, main="결측값의 위치와 비율")
```

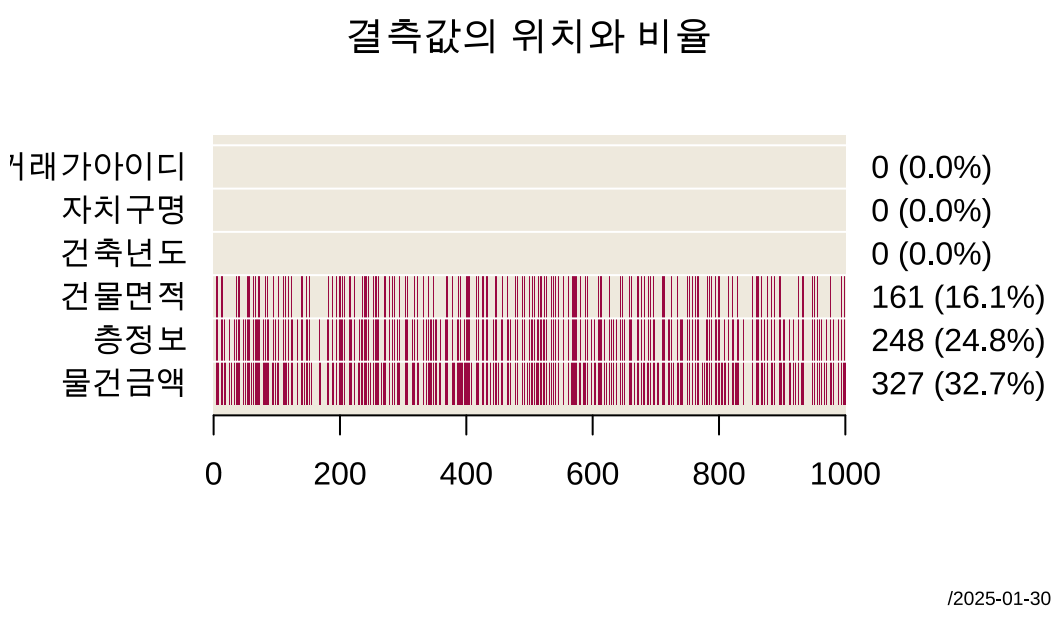


그림 2.3: 결측값의 위치와 비율 - df_miss_2

plot_pattern 함수를 이용하여 결측가 발생하는 패턴에 대해서 살펴보자

```
plot_pattern(df_miss_2, square= FALSE)
```

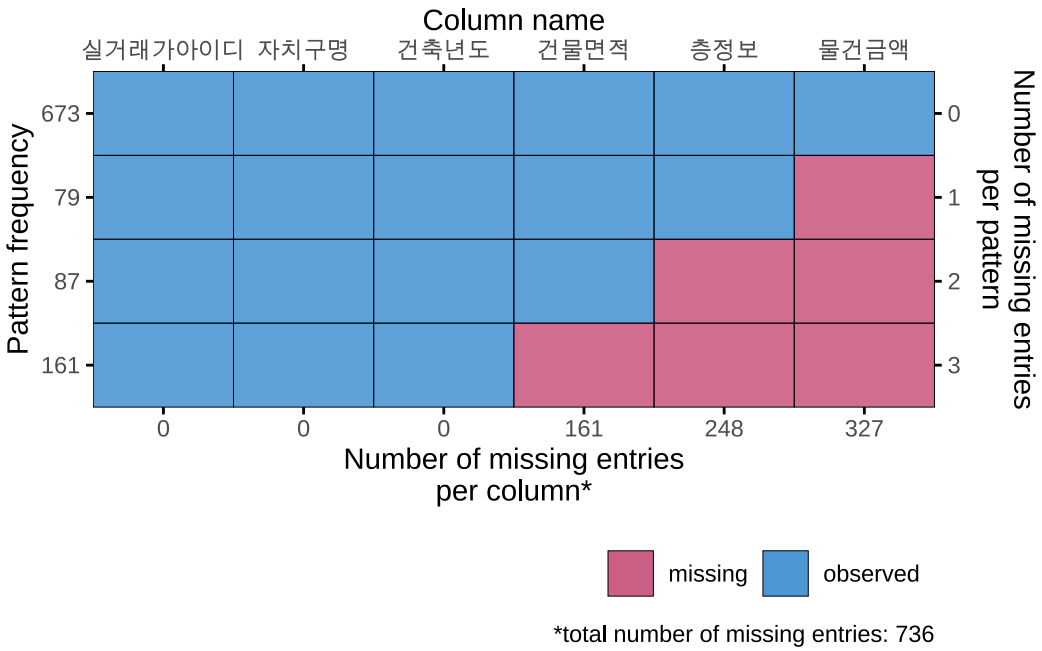


그림 2.4: 결측값의 계층적인 패턴

그림 2.4 에 나타나 결측값의 발생 형태를 보면 특별한 경향이 있는 것으로 보인다.

df_miss_2 자료에서는 세 개의 변수에서 결측값이 존재하며, 이 결측값들은 단순한 무작위적 발생이 아닌 계층적인 패턴을 따르는 것으로 나타났다. 층정보컬럼는건물면적' 값이 결측된 경우, 반드시 결측되는 구조를 보였다. 즉, 건물면적에 대한 정보가 없는 경우 층수 역시 제공되지 않는 패턴이 관찰된 것이다. 이러한 상관관계는 데이터의 누락이 단순한 실수나 오류에서 비롯된 것이 아니라, 특정 컬럼 간 논리적인 관계 속에서 발생하고 있음을 보여준다.

또한, 물건금액 컬럼의 경우, 층정보가 결측되면 자동으로 결측값이 되는 구조를 보였다. 이는 물건금액이 층정보에 의존적이라는 점을 시사하며, 데이터의 구조적 특성을 이해하는 데 중요한 단서가 된다.

다음은 naniar 패키지에서 제공하는 결측값의 위치를 제공하는 함수 geom_miss_point() 를 사용하여 건축년도 와 건물면적 의 관계를 나타내는 산점도(scatter plot)에 건물면적이 없는 결측값을 나타낸 그림이다.

```
ggplot(df_miss_2,
       aes(x = `건축년도`,
           y = `건물면적`)) +
  geom_miss_point()
```

그림 2.5: 두 변수와의 관계와 결측값 패턴

건물면적 컬럼의 경우, 건축연도가 최근일수록 결측값이 발생할 가능성이 높은 것으로 확인된다. 이는 건축연도가 최신일수록 특정 정보가 기록되지 않는 경향이 있음을 시사한다.

마지막으로 결측값이 있는 자료 `df_miss_3` 에서 결측값의 위치를 시각적으로 살펴보자.

```
PlotMiss(df_miss_2, main="결측값의 위치와 비율")
```

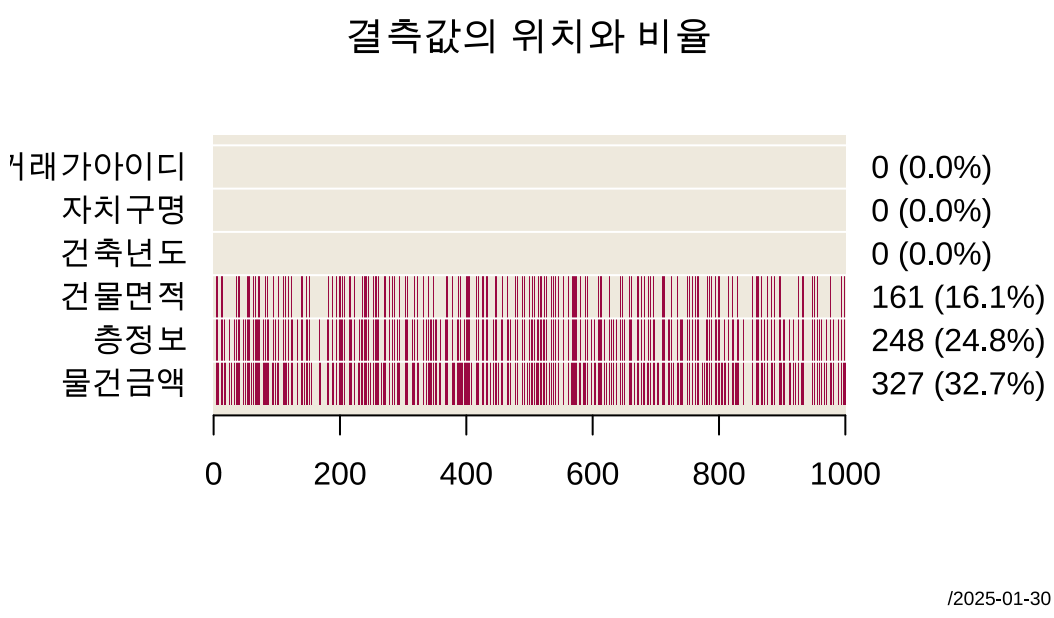


그림 2.6: 결측값의 위치와 비율 - df_miss_3

`plot_pattern` 함수를 이용하여 자료 `df_miss_3` 에서 결측가 발생하는 패턴에 대해서 살펴보자

```
plot_pattern(df_miss_3, square= FALSE)
```

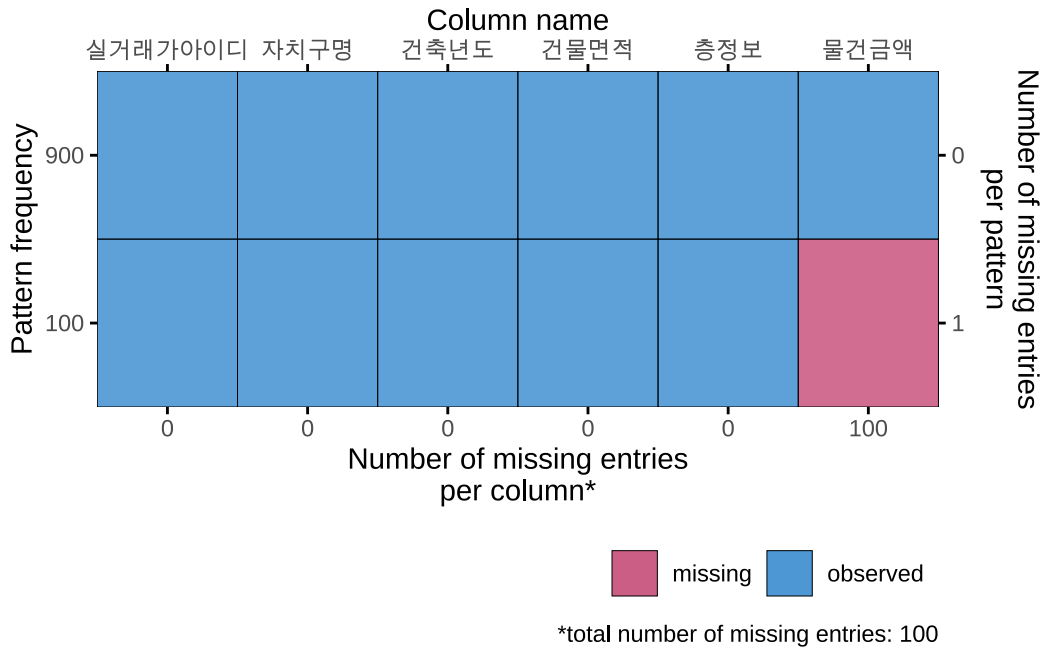


그림 2.7: 중요한 컬럼에서 결측값의 발생

그림 2.7 에 나타나 결측값의 발생 형태를 보면 자료레서 가장 관심있는 컬럼인 물건금액 에서만 10% 결측이 발생하였다. 물건금액 이 중요한 변수이기 때문에 결측값이 발생하는 패컨이 무작위인지 아니면 특별한 패턴을 가지고 있는지가 중요하다.

이제 건물면적 에 따라서 변하는 물건금액 에 대한 관계를 나타내는 산점도(scatter plot)에 건물면적이 없는 결측값을 나타낸 그림이다.

```
ggplot(df_miss_3,
       aes(x = `건물면적`,
           y = `물건금액`)) +
  geom_miss_point()
```

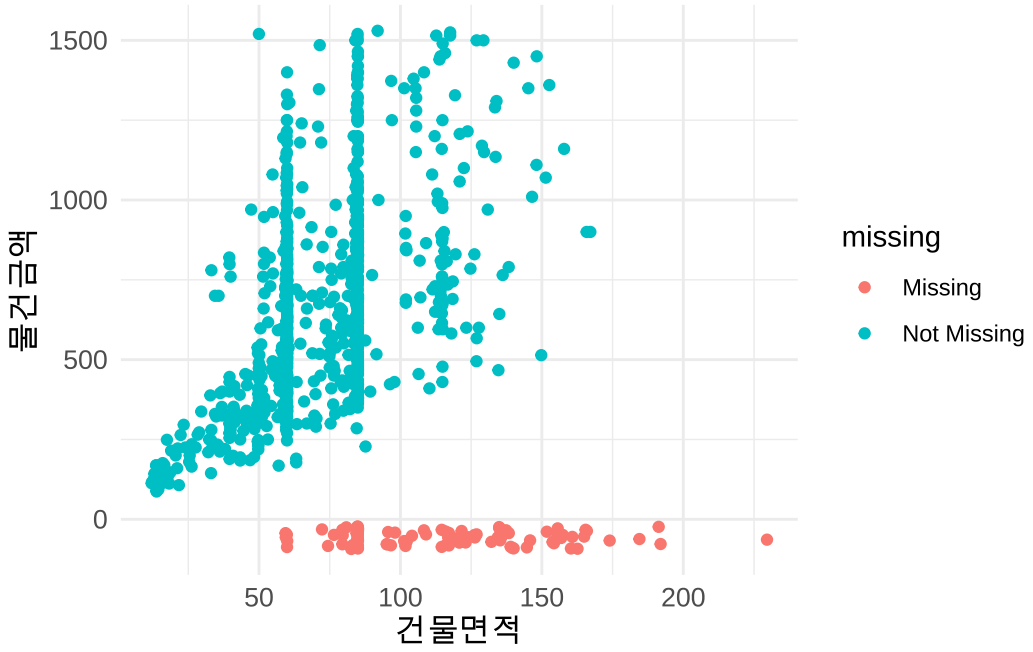


그림 2.8: 두 변수와의 관계와 결측값 패턴

그림 2.8 를 보면 자료에 건물면적이 커질수록 물건금액 이 결측값이 나타날 경향이 큰 것으로 보인다. 특히, 건물면적이 매우 큰 자료에서는 물건금액 이 결측인 경우가 매우 많다. 이렇게 가장 관심있는 변수에서 나타나는 결측값의 패턴이 무작위가 아니라 자신의 값에 영향을 받는 경우, 결측값을 가진 단위를 제거한 자료를 분석한 결과는 실제 결과와 매우 다를 수 있다.

결측값의 발생 패턴을 무시한 채 단순히 보정하거나 제거할 경우, 데이터의 본질적인 특성을 왜곡할 위험이 있다. 따라서 결측값을 처리하기 전, 그 발생 구조를 면밀히 분석하고 적절한 보완 전략을 수립하는 것이 무엇보다 중요하다.

데이터 분석에서 결측값은 단순한 누락 이상의 의미를 지닌다. 특히, 결측값이 일정한 패턴을 보이며 발생할 경우, 이를 무작정 보완하거나 제거하는 것이 아니라 그 구조를 이해하는 것이 중요하다.

3 데이터 요약표 작성의 기초

3.0.1 서론

데이터 분석 과정에서 가장 중요한 단계 중 하나는 자료를 요약하고, 통계적으로 의미 있는 형태로 정리하는 것이다. 원시 데이터(raw data)는 보통 복잡하고 정리가 필요한 경우가 많으며, 이를 효율적으로 정리하지 않으면 분석의 방향을 설정하는 것이 어려워질 수 있다. 이러한 과정에서 R의 대표적인 데이터 처리 패키지인 `dplyr`과 `tidyr`이 강력한 도구로 활용된다.

`dplyr`은 데이터 프레임을 다룰 때 가장 널리 사용되는 패키지로, 직관적인 문법과 강력한 기능을 제공한다. 특히, 데이터를 요약하고 통계 테이블을 생성하는 과정에서 코드의 가독성(tidy evaluation)을 높이고 실행 속도를 최적화할 수 있다는 점에서 유용하다. 특히, 연속적인 데이터 변환에서 %>%(pipe 연산자)를 활용하면 여러 연산을 논리적으로 연결하여 가독성을 높일 수 있다.

통계 테이블을 만들 때 데이터는 종종 정리되지 않은 형태(untidy data)로 존재하며, 분석을 위해 올바른 형태로 변환해야 하는 경우가 많다. `tidyr` 패키지는 이러한 데이터 정리 과정에서 필수적인 역할을 한다. `tidyr`을 사용하면 넓은 형식(Wide format)과 긴 형식(Long format)의 자료로 상호 변환이 가능하다. 또한 결측값 및 불완전한 데이터 정리하는데 유용하게 사용된다.

대부분의 데이터 정리는 **요약(summarization)**과 **변환(transformation)**이 함께 이루어진다. `dplyr`과 `tidyr`은 각각의 역할을 수행하며, 함께 사용하면 더욱 강력한 데이터 처리 능력을 발휘할 수 있다.

표 3.1: `dplyr`과 `tidyr` 의 주요함수와 그 기능

단계	기능	주요 함수
데이터 요약	그룹별 통계 계산	`group_by()`, `summarise()`
데이터 변환	넓은 형식 ↔ 긴 형식 변환	`pivot_wider()`, `pivot_longer()`
결측값 처리	누락된 값 제거 또는 대체	`drop_na()`, `replace_na()`
변수 정리	새로운 변수 생성 또는 선택	`mutate()`, `select()`

이처럼 `dplyr`은 데이터를 요약하고 통계를 생성하는 역할을 하며, `tidyr`은 정리되지 않은 데이터를 보다 분석하기 쉬운 형태로 변환하는 역할을 한다. 따라서, 통계 테이블을 만들거나 데이터를 정리하는 모든 과정에서 `dplyr`과 `tidyr`은 필수적인 도구**이며, 이를 활용하면 데이터 분석의 효율성과 신뢰성을 크게 향상시킬 수 있다.

3.1 데이터의 요약

이 절에서는 요약통계량을 도출하거나 보고서로 올리기위해 자료를 요약하여 정리하는 방법을 보이고자 한다.

먼저, 함수들을 설명하기 위한 간단한 데이터프레임을 만들어본다.

```r
df <- data.frame(
  이름 = c("홍길동", "김영희", "박찬호", "이소라", "최민식"),
  도시 = c("서울", "부산", "서울", "인천", "부산"),
  나이 = c(25, 30, 35, 40, 28),
  점수 = c(80, 90, 75, 82, 95)
)

knitr::kable(df)
```

표 3.2: 예제 데이터프레임

이름	도시	나이	점수
홍길동	서울	25	80
김영희	부산	30	90
박찬호	서울	35	75
이소라	인천	40	82
최민식	부산	28	95

3.1.1 그룹의 지정: group_by()

- 특정 열(또는 여러 열)을 기준으로 행들을 묶어 그룹한다.
- 그룹화 후에는 요약summarize(), 변환mutate(), 혹은 재구조화reframe() 등의 작업을 그룹 단위로 수행할 수 있다.

```r
df_grouped <- df %>%
  group_by(도시)

df_grouped
```

```
# A tibble: 5 x 4
# Groups:   도시 [3]
  이름  도시  나이 점수
 <chr>  <chr> <dbl> <dbl>
1 홍길동 서울   25    80
2 김영희 부산   30    90
3 박찬호 서울   35    75
4 이소라 인천   40    82
5 최민식 부산   28    95
```

- 결과물을 보면 눈에 보이는 데이터는 바뀌지 않지만, 내부적으로 도시를 기준으로 데이터가 '그룹화'된 상태가 된다. 이후에 summarize(), reframe() 등을 하면 "도시별"로 계산한다.

3.1.2 자료의 요약: summarize() 또는 summarise()

- 평균, 합계, 개수 등의 통계를 구할 때 자주 사용한다. 한 번에 여러 요약 통계를 낼 수도 있다.
- 도시별로 평균 나이와 평균 점수를 구해보자.
  - mean(나이): 각 도시 그룹 내 나이의 평균
  - mean(점수): 각 도시 그룹 내 점수의 평균
  - n(): 그룹 내 행(row)의 개수(인원수)를 세어줌

```
df %>%
  group_by(도시) %>%
  summarize(
    `평균나이` = mean(`나이`),
    `평균점수` = mean(`점수`),
    `인원수`   = n()
  )
```

```
# A tibble: 3 x 4
  도시  평균나이 평균점수 인원수
 <chr>     <dbl>    <dbl>  <int>
1 부산     29      92.5       2
2 서울     30      77.5       2
3 인천     40      82         1
```

3.1.3 제이터프레임의 컬럼명

dplyr과 tidyr 제공되는 함수들은 간결한 연산(tidy evaluation)의 방식이 적용되어 데이터프레임의 컬럼 이름을 나타낼 때 영문과 한글 상관없이 따옴표를 붙이지 않고 없이 사용할 수 있다. 하지만 컬럼명을 한글로 사용하는 경우 간결한 연산이 적용되지 않는 다른 패키지의 함수와 같이 사용하는 경우 문제가 발생할 수 있다.

따라서 한글 컬럼 이름은 언제나 역따옴표 “ 로 묶어주는 것이 안전하다.

3.1.4 컬럼의 위치 이동: relocate()

- 열(column)의 순서를 재배치한다.
  - .before: 어떤 열의 앞에 지정한 열들을 옮길지 설정 (또는 .after도 있음)
  - .before = 이름: 기존의 이름 열 앞에 도시 열을 위치시키라는 의미
- 데이터를 보기 편하게 만들거나, 특정 열을 맨 앞으로 가져올 때 유용하다.
- 예제 자료에서 도시 열을 맨 앞으로 가져오기

```
df_relocated <- df %>%
  relocate(도시, .before = 이름)

df_relocated
```

```
  도시  이름 나이 점수
1 서울 홍길동   25   80
2 부산 김영희   30   90
3 서울 박찬호   35   75
4 인천 이소라   40   82
5 부산 최민식   28   95
```

3.1.5 컬럼의 선택: `select()`

- 원하는 열만 선택하거나, 열의 순서를 지정하는 데 사용한다.
- 데이터에서 특정 열만 추출하거나, 불필요한 열을 제거하고 싶을 때 사용한다.
- 다음 예시는 이름과 점수 열만 선택

```
df_selected <- df %>%
  select(이름, 점수)

df_selected
```

```
  이름 점수
1 홍길동   80
2 김영희   90
3 박찬호   75
4 이소라   82
5 최민식   95
```

3.1.6 재구조화: `reframe()`

- `summarize()`와 비슷하게 그룹화된 데이터를 재구조화하되, 결과를 그룹별 여러 행으로 반환할 수도 있다.
- dplyr 1.1.0부터 추가된 기능으로, 기존 `summarize()`가 그룹당 "요약된 한 행"을 반환했던 것과 달리, `reframe()`은 그룹별로 여러 행을 만들어낼 수 있다.
- 다음 예시는 도시별로 데이터를 모은 뒤, 그 안에서 '이름'과 '점수'만 추려서 재구조화

```
df_reframed <- df %>%
  group_by(도시) %>%
  reframe(
    이름,
    점수,
    평균도시점수 = mean(점수)
  )

df_reframed
```

```
# A tibble: 5 x 4
  도시 이름   점수 평균도시점수
 <chr> <chr>  <dbl>        <dbl>
1 부산 김영희   90         92.5
2 부산 최민식   95         92.5
3 서울 홍길동   80         77.5
4 서울 박찬호   75         77.5
5 인천 이소라   82         82
```

- 도시별로 그룹화한 뒤 각 그룹 내부에서 모든 이름과 점수를 그대로 나열하면서, 같은 그룹 내 평균 점수(평균도시점수)를 함께 표시한다.
- 결과는 그룹당 여러 행(그룹에 속한 사람 수만큼)이 나올 수 있으며, 요약 통계는 반복되어 표시한다.

3.1.7 컬럼명 변경 `rename()` 또는 `rename_with`

- 특정 규칙에 따라 여러 열의 이름을 한꺼번에 변경할 때 사용한다.
- 예를 들어, 모든 열 이름에 접두사(prefix)를 붙이거나, 대소문자 변환 등을 할 수 있다.

```
df_renamed1 <- df %>%
  rename(city = 도시, name = 이름)

df_renamed1
```

```
    name city 나이 점수
1 홍길동 서울   25   80
2 김영희 부산   30   90
3 박찬호 서울   35   75
4 이소라 인천   40   82
5 최민식 부산   28   95
```

- 다음 예시는 모든 열 이름에 “new_”라는 접두사를 붙인다.
  - `.fn`: 열 이름을 어떻게 변환할지 정의한 함수
  - `.cols`: 변환할 열의 범위 지정(예: 특정 열, `everything()`)

```
df_renamed2 <- df %>%
  rename_with(.fn = ~ paste0("new_", .), .cols = everything())

df_renamed2
```

```
  new_이름 new_도시 new_나이 new_점수
1   홍길동    서울      25        80
2   김영희    부산      30        90
3   박찬호    서울      35        75
4   이소라    인천      40        82
5   최민식    부산      28        95
```

3.1.8 정열: arrange()

- 원하는 열을 기준으로 행을 정렬한다. 오름차순/내림차순 모두 가능.
- 내림차순 정렬 시에는 desc()를 사용.
- 다음 예시는 점수 순서대로 오름차순 정렬

```
df_arranged <- df %>%
  arrange(점수)

df_arranged
```

```
     이름 도시 나이 점수
1 박찬호 서울   35   75
2 홍길동 서울   25   80
3 이소라 인천   40   82
4 김영희 부산   30   90
5 최민식 부산   28   95
```

- arrange(점수): 점수가 낮은 순서에서 높은 순서로 정렬
- 만약 높은 순서부터 정렬을 원한다면 arrange(desc(점수))

3.1.9 컬럼 묶기 across()

- 여러 열에 대해 동일한 함수를 일괄 적용할 때 사용한다.
- mutate(), summarize() 등에서, 선택한 여러 열에 대해 한꺼번에 연산을 적용할 수 있다.
- 다음 예시는 나이와 점수를 각각 2배로 만들고, 새로운 열(나이2배, 점수2배)을 생성

```
df_across <- df %>%
  mutate(
    across(
      .cols = c(나이, 점수),        # 어떤 열에 적용할지
      .fns  = ~ . * 2,              # 어떤 함수를 적용할지
      .names = "{.col}2배"          # 새로운 열 이름 형식
    )
  )

df_across
```

```
     이름 도시 나이 점수 나이2배 점수2배
1 홍길동 서울   25   80      50     160
2 김영희 부산   30   90      60     180
3 박찬호 서울   35   75      70     150
4 이소라 인천   40   82      80     164
5 최민식 부산   28   95      56     190
```

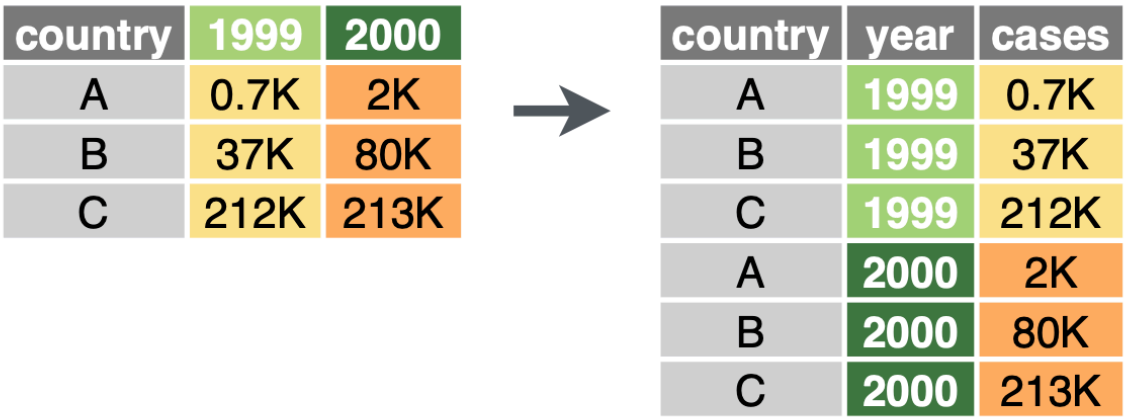
- `.cols`: 변환 적용 대상 열 지정
- `.fns`: 적용할 함수
- `.names`: 새 열 이름 패턴 정의 (`"{.col}"`, `"{.col}_새이름"` 등)

3.2 데이터의 변환

데이터프레임의 형태를 바꾸는 중요한 함수 2개를 소개한다.

3.2.1 긴 형식: `pivot_longer()`

- 가로 방향으로 나열된 데이터를 세로 방향(긴 형태, Long format)로 변환한다.



긴 형식으로 변환(그림출처: `tidyr` 홈페이지)

- 보통 측정 항목(예: 나이, 점수)이 열로 되어 있을 때, 이를 "변수 이름"과 "값"의 두 열로 합쳐서 "길게"(long) 만들 때 쓰인다.
- 다음 예시는 나이와 점수 열을 세로 방향으로 길게 변환

```
df_long <- df %>%
  pivot_longer(
    cols      = c(나이, 점수),
    names_to  = "측정항목",
    values_to = "값"
  )

df_long
```

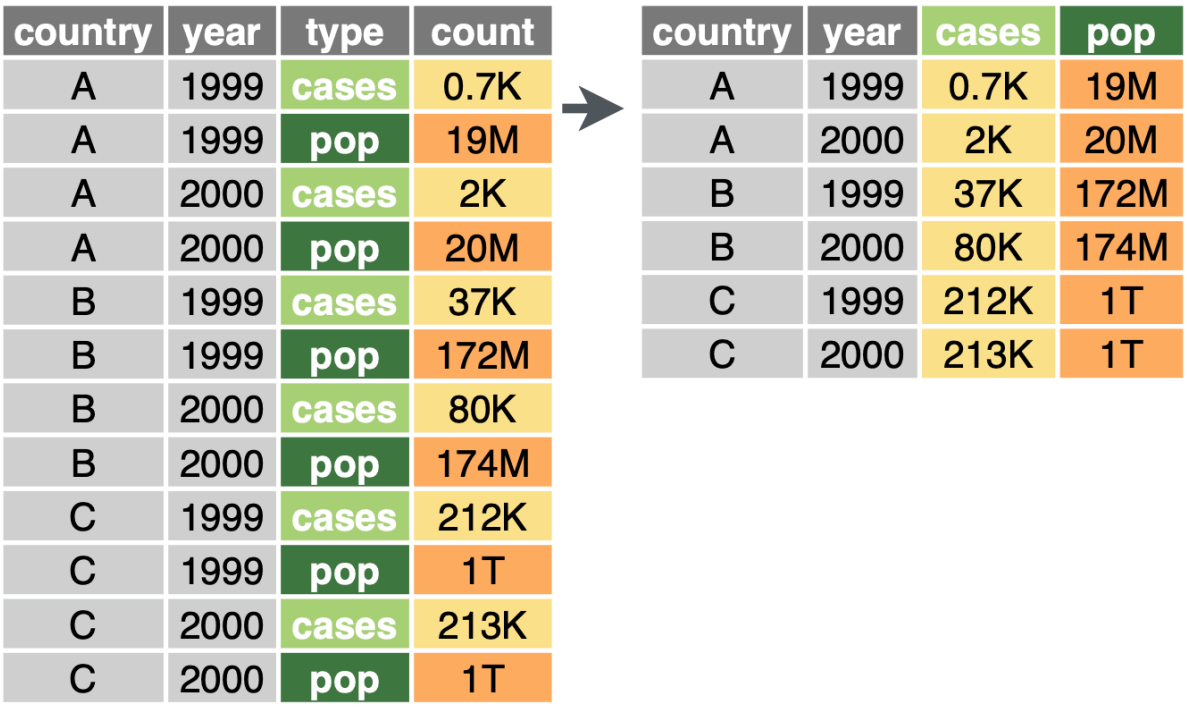
```
# A tibble: 10 x 4
   이름  도시 측정항목  값
  <chr>  <chr> <chr>    <dbl>
 1 홍길동 서울 나이      25
 2 홍길동 서울 점수      80
```

```
 3 김영희 부산 나이     30
 4 김영희 부산 점수     90
 5 박찬호 서울 나이     35
 6 박찬호 서울 점수     75
 7 이소라 인천 나이     40
 8 이소라 인천 점수     82
 9 최민식 부산 나이     28
10 최민식 부산 점수     95
```

- `cols`: 세로로 변환할(길게 만들) 대상 열을 지정
- `names_to`: 기존 열 이름이 어떤 새 열 이름으로 저장될지 지정
- `values_to`: 기존 열 값이 어떤 새 열 이름으로 저장될지 지정

3.2.2 넓은 형식: `pivot_wider()`

- 세로 방향(긴 형태)로 나열된 데이터를 가로 방향(넓은 형태)으로 변환한다.



country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

넓 은 형식으로 변환(그림출처: `tidyr` 홈페이지)

- 일반적으로 `pivot_longer()`로 만들었던 긴 데이터 프레임을 다시 열별로 펼칠 때 사용한다.
- 다음 예시는 `pivot_longer()` 결과물(df_long)을 다시 넓은 형식으로 변환

```
df_wide <- df_long %>%
  pivot_wider(
    names_from  = "측정항목",
    values_from = "값"
  )
```

```
df_wide
```

```
# A tibble: 5 x 4
  이름  도시  나이 점수
 <chr>  <chr> <dbl> <dbl>
1 홍길동 서울   25    80
2 김영희 부산   30    90
3 박찬호 서울   35    75
4 이소라 인천   40    82
5 최민식 부산   28    95
```

- `names_from`: 어떤 열의 값을 "열 이름"으로 펼칠지 지정
- `values_from`: 어떤 열의 값을 "열 값"으로 사용할지 지정
- 다시 이름, 도시를 그대로 두면서, 나이와 점수가 각각 열로 복원된 형태가 된다.

3.3 실습: 데이터요약표 만들기

이제 실제 데이터를 이용하여 분할표와 요약통계표를 구해보자.

```
load(here::here("data", "physical100_data.RData"))
ls()
```

```
 [1] "df"           "df_across"    "df_arranged"  "df_grouped"   "df_long"
 [6] "df_reframed"  "df_relocated" "df_renamed1"  "df_renamed2"  "df_selected"
[11] "df_wide"      "df1_youth"
```

3.3.1 소개

- 2018-2023년 국민체력100사업에서 측정한 초등학교 체력측정 자료
- 실습을 위하여 1000명 임의로 추출한 자료
- 이 실습에서 통계표를 작성하는 원천 자료(raw data; source data)로 이용

```
head(df1_youth, 10)
```

```
       ID CENTER_NM TEST_AGE   TEST_YMD TEST_YEAR TEST_SEX ITEM_F001 ITEM_F002
    <int>    <char>    <int>     <Date>     <int>   <char>     <num>     <num>
 1:     1      삼척    13 2018-10-12      2018     남성   172.5      62.7
 2:     2      동작    18 2018-05-17      2018     남성   180.8      79.2
 3:     3      마포    16 2018-05-02      2018     남성   169.0      63.0
 4:     4      강릉    17 2019-04-01      2019     여성   151.9      52.5
 5:     5      남원    14 2020-09-22      2020     여성   150.1      61.2
 6:     6      안동    17 2018-10-22      2018     남성   173.6      53.5
```

```
 7:     7      원주    15 2023-05-08      2023     여성   163.8      66.2
 8:     8      태백    14 2022-05-17      2022     남성   162.9      53.4
 9:     9      성남    16 2018-09-19      2018     남성   163.1      63.9
10:    10      서초    13 2019-05-21      2019     남성   169.3      66.7
    ITEM_F009 ITEM_F012 ITEM_F018 ITEM_F020 ITEM_F022 ITEM_F028
        <num>     <num>     <num>     <num>     <num>     <num>
 1:        NA      -8.2      21.1        25        NA      42.2
 2:        NA       8.5      24.2        64       245      78.1
 3:        27      -7.0      22.1        36       165      52.5
 4:        NA      10.9      22.8        17        NA      36.7
 5:        NA      27.5      27.2        18       137      37.2
 6:        27       9.0      17.8        29       220      71.7
 7:        NA      25.2      24.7        15       133      45.2
 8:        NA       9.6      20.1        31       210      45.7
 9:        NA      17.6      24.0        50        NA      58.3
10:        NA       1.2      23.3        20       150      42.1
```

3.3.2 요약통계표

- df1_youth에서 성별과 각 년도별로 그룹화 해서 도수,평균 등을 계산.

코드에 대해 설명하면,

1. 원천자료 df1_youth 를 지정하고
2. TEST_SEX, TEST_YEAR, ITEM_F001 컬럼을 선택하고
3. TEST_SEX, TEST_YEAR 컬럼으로 그룹화한 후
4. 신장(ITEM_F001) 에 대한 그룹별 돗수, 평균, 표준편차를 계산

```
basic_stat_table <- df1_youth %>%
  select(TEST_SEX, TEST_YEAR, ITEM_F001) %>%
  group_by(TEST_SEX, TEST_YEAR) %>%
  summarize(n = n(), MEAN =mean(ITEM_F001, na.rm= TRUE ),
            SD =sd(ITEM_F001, na.rm= TRUE ),
            .groups = "drop"  # 그룹을 모두 해제
            )
basic_stat_table
```

```
# A tibble: 14 x 5
   TEST_SEX TEST_YEAR     n  MEAN     SD
   <chr>         <int> <int> <dbl>  <dbl>
 1 남성       2017    85  188. 173.
 2 남성       2018   131  171.   6.38
 3 남성       2019   140  171.   7.49
 4 남성       2020    20  172.   5.26
 5 남성       2021    29  173.   6.61
 6 남성       2022    66  174.   6.28
 7 남성       2023    89  171.   7.79
```

```
 8 여성     2017    83  160.   6.27
 9 여성     2018    90  160.   5.38
10 여성     2019    96  160.   5.33
11 여성     2020    13  160.   4.28
12 여성     2021    22  159.   4.11
13 여성     2022    52  161.   5.59
14 여성     2023    84  161.   6.05
```

3.3.3 분할표

남여별로 각 년도에 돗수를 분할표로 구해본다.

1. 원천자료 `df1_youth` 를 지정하고
2. `TEST_SEX`, `TEST_YEAR` 컬럼을 선택하고
3. `TEST_SEX`, `TEST_YEAR` 컬럼으로 그룹화한 후
4. 각 그룹의 조합의 돗수를 n 으로 계산한다.
5. 측정연도 `TEST_YEAR` 의 숫자 앞에 Y 를 붙여서 새로운 열이름을 만들고, 성별과 연도에 해당하는 돗수로 교차표를 만든다.

```
cross_stat_table <- df1_youth %>%
  select(TEST_SEX, TEST_YEAR) %>%
  group_by(TEST_SEX, TEST_YEAR) %>%
  summarize(n = n()) %>%
  pivot_wider(names_from = TEST_YEAR,
              values_from = n, names_prefix = "Y" )
```

```
`summarise()` has grouped output by 'TEST_SEX'. You can override using the
`.groups` argument.
```

```
cross_stat_table
```

```
# A tibble: 2 x 8
# Groups:   TEST_SEX [2]
  TEST_SEX Y2017 Y2018 Y2019 Y2020 Y2021 Y2022 Y2023
  <chr>    <int> <int> <int> <int> <int> <int> <int>
1 남성        85   131   140    20    29    66    89
2 여성        83    90    96    13    22    52    84
```

3.3.4 통계표 만들기

통계표를 만드는 방법을 여러가지 보일 것이다.

우선 통계량을 구하는 함수를 코드에 일일히 입력하기 번거로우니 여러 개의 지정된 함수와 이름을 `list`로 묶어서 `my_summ_func`에 저장하는 방법을 사용한다.

각 측정항목에 적용할 통계 함수와 출력값의 이름을 지정한다.

```
my_summ_func <- list(
  개수 = ~sum(!is.na(.x)),
  결측개수 = ~sum(is.na(.x)),
  평균 = ~mean(.x, na.rm = TRUE),
  표준편차 = ~sd(.x, na.rm = TRUE),
  최소값 = ~min(.x, na.rm = TRUE),
  백분위25 = ~quantile(.x, probs = 0.25, na.rm = TRUE),
  중앙값 = ~median(.x, na.rm = TRUE),
  백분위75 = ~quantile(.x, probs = 0.75, na.rm = TRUE),
  최대값 = ~max(.x, na.rm = TRUE)
)
```

- ~sum,~mean, 등등: 함수를 쓸때 간단하게 작성하는 방법
- .x: 자리표시자로써 열 또는 원소를 지칭하는 의미
- na.rm: 결측치(NA)를 제거하고 계산할지 여부를 물어보는 것

1. 1단계 통계표

간단한 통계표부터 보여본다.

- 원천자료 df1_youth를 지정하고
- TEST_SEX, TEST_YEAR 컬럼을 선택하고
- my_summ_func 에서 정의한 함수를 열이름이 ITEM을 포함한 모든열에 적용한다.

```
aa <- df1_youth %>%
  group_by(TEST_SEX, TEST_YEAR) %>%
  summarise(across(contains("ITEM"), my_summ_func,
                   .names = "{.col}-{.fn}"))
```

```
`summarise()` has grouped output by 'TEST_SEX'. You can override using the
`.groups` argument.
```

```
aa
```

```
# A tibble: 14 x 74
# Groups:   TEST_SEX [2]
   TEST_SEX TEST_YEAR `ITEM_F001-개수` `ITEM_F001-결측개수` `ITEM_F001-평균`
  <chr>        <int>            <int>                <int>             <dbl>
 1 남성      2017               85                    0              188.
 2 남성      2018              131                    0              171.
 3 남성      2019              140                    0              171.
 4 남성      2020               20                    0              172.
 5 남성      2021               29                    0              173.
 6 남성      2022               66                    0              174.
 7 남성      2023               89                    0              171.
 8 여성      2017               83                    0              160.
 9 여성      2018               90                    0              160.
```

```
10 여성     2019               96                    0             160.
11 여성     2020               13                    0             160.
12 여성     2021               22                    0             159.
13 여성     2022               52                    0             161.
14 여성     2023               84                    0             161.
# i 69 more variables: `ITEM_F001-표준편차` <dbl>, `ITEM_F001-최소값` <dbl>,
#   `ITEM_F001-백분위25` <dbl>, `ITEM_F001-중앙값` <dbl>,
#   `ITEM_F001-백분위75` <dbl>, `ITEM_F001-최대값` <dbl>,
#   `ITEM_F002-개수` <int>, `ITEM_F002-결측개수` <int>, `ITEM_F002-평균` <dbl>,
#   `ITEM_F002-표준편차` <dbl>, `ITEM_F002-최소값` <dbl>,
#   `ITEM_F002-백분위25` <dbl>, `ITEM_F002-중앙값` <dbl>,
#   `ITEM_F002-백분위75` <dbl>, `ITEM_F002-최대값` <dbl>, ...
```

2. 2단계 통계표

1단계 통계표에서 pivot_longer을 사용해 추가적으로 진행을 해본다.

- 항목이름과 통계이름으로 구성된 엷은 열을 가진 자료를 다시 긴 행을 가진 자료로 변환
- 변환시 열이름을 두 열로 나누어 저장(항목이름 item과 통계량 stat)
- 항목이름 item열을 앞으로 배치

```
bb <- aa %>% pivot_longer(!c(TEST_SEX, TEST_YEAR), names_to = c("item", "stat"), names_sep="-" , values
bb
```

```
# A tibble: 1,008 x 5
# Groups:   TEST_SEX [2]
   TEST_SEX TEST_YEAR item      stat     value
   <chr>         <int> <chr>     <chr>    <dbl>
 1 남성       2017 ITEM_F001 개수    85
 2 남성       2017 ITEM_F001 결측개수  0
 3 남성       2017 ITEM_F001 평균    188.
 4 남성       2017 ITEM_F001 표준편차 173.
 5 남성       2017 ITEM_F001 최소값  145.
 6 남성       2017 ITEM_F001 백분위25  165.
 7 남성       2017 ITEM_F001 중앙값  170
 8 남성       2017 ITEM_F001 백분위75  175.
 9 남성       2017 ITEM_F001 최대값  1761
10 남성       2017 ITEM_F002 개수    85
# i 998 more rows
```

3. 3단계 통계표

2단계 통계표에서 pivot_wider을 진행해 추가적으로 진행을 해본다.

- 긴 행을 가진 자료를 측정 년도를 열로 바꾸어 긴 열을 가진 자료로 변환
- 항목이름과 성별순으로 자료를 정렬

```
cc <- bb %>% dplyr::relocate(item) %>% pivot_wider(names_from = TEST_YEAR, values_from = value)
cc
```

```
# A tibble: 144 x 10
# Groups:   TEST_SEX [2]
   item      TEST_SEX stat     `2017` `2018` `2019` `2020` `2021` `2022` `2023`
   <chr>     <chr>    <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1 ITEM_F001 남성   개수     85   131    140     20     29     66     89
 2 ITEM_F001 남성   결측개수  0     0      0      0      0      0      0
 3 ITEM_F001 남성   평균    188. 171.    171.   172.   173.   174.   171.
 4 ITEM_F001 남성   표준편차 173.   6.38   7.49   5.26   6.61   6.28   7.79
 5 ITEM_F001 남성   최소값   145. 146.    144.   161.   155    159    143.
 6 ITEM_F001 남성   백분위25   165. 167.    166.   169.   170    171.   166.
 7 ITEM_F001 남성   중앙값   170  172.    171.   173.   174.   174.   172.
 8 ITEM_F001 남성   백분위75   175. 175.    176.   174.   177.   177.   176.
 9 ITEM_F001 남성   최대값  1761  184.    189    183.   185.   189    187.
10 ITEM_F002 남성   개수     85   131    140     20     29     66     89
# i 134 more rows
```

3.3.5 함수만들기

실제로 통계표를 만들고 출력하는 데까지 한번에 하는 것이 아닌 여러번의 과정을 거쳐야 한다는 것을 알고 있다.

그렇기에 여러번 돌릴 때마다 코드를 수정하면서 돌리는 것은 비효율적이므로 함수를 만들어서 필요할 때 마다 간편하게 통계표를 만드는 과정을 보려고 한다.

우선, summarise에 쓰일 통계량이 많기에 리스트 형태로 미리 지정하는 것이 편하다.

```
my_summ_func <- list(
  `개수` = ~sum(!is.na(.x)),
  `결측개수` = ~sum(is.na(.x)),
  `평균` = ~mean(.x, na.rm = TRUE),
  `표준편차` = ~sd(.x, na.rm = TRUE),
  `최소값` = ~min(.x, na.rm = TRUE),
  `백분위25` = ~quantile(.x, probs = 0.25, na.rm = TRUE),
  `중앙값` = ~median(.x, na.rm = TRUE),
  `백분위75` = ~quantile(.x, probs = 0.75, na.rm = TRUE),
  `최대값` = ~max(.x, na.rm = TRUE)
)

my_summ_func_2 <- list(
  `개수` = ~n(),
  `평균` = ~mean(.x, na.rm = TRUE),
  `표준편차` = ~sd(.x, na.rm = TRUE)
)
```

이제 직접 함수를 만들고 이해해보자.

```
summ_function_general <- function(df, cols_to_row, cols_to_col, cols_to_summ, stat_fun, stat_to_column

  col1 <- syms(cols_to_row)
  col2 <- syms(cols_to_col)
  col3 <- syms(cols_to_summ)

  tab1 <- df %>%
    group_by(!!!col1, !!!col2) %>%
    summarise(across(all_of(cols_to_summ), stat_fun,
                     .names = "{.col}-{.fn}")) %>%
    pivot_longer(cols = -c(!!!col1, !!!col2),
                 names_to = c("ITEM", "STAT"),
                 names_sep="-" , values_to = "value")  %>%
    relocate(ITEM) %>%
    dplyr::arrange(ITEM) %>%
    dplyr::ungroup()

  if (stat_to_column) {
    tab1 <- tab1  %>% pivot_wider(names_from = all_of(c(cols_to_col,"STAT")), values_from = value)
  }
  else {
    tab1 <- tab1 %>% pivot_wider(names_from = all_of(cols_to_col), values_from = value)
  }

   return(tab1)
}
```

우선, 입력할 인자들이 무엇인지 확인하자.

```
function(df, #  데이터프레임 입력
         cols_to_row,  # "행 그룹"으로 사용할 열 이름(들).
         cols_to_col,  # "열 그룹"으로 사용할 열 이름(들).
         cols_to_summ, # 요약 함수(stat_fun)를 적용할 대상 열 이름(들).
         stat_fun,     # 요약에 사용할 함수(예: mean, sum, sd 등).
         stat_to_column = FALSE # 결과를 펼칠 때(pivot_wider) 통계값(STAT)을 열 이름으로 포함할지 여부
         )
```

1. `syms`

- 문자열 벡터를 symbol 리스트로 바꿔준다.
- 예를 들어 `cols_to_row = c("SEX","YEAR")`라면 `col1`은 `list(quote(SEX), quote(YEAR))`가 된다.
- 이후 `!!!`(splicing operator)와 함께 `group_by()` 등에 활용하기 위함이다.

```
  col1 <- syms(cols_to_row)
  col2 <- syms(cols_to_col)
  col3 <- syms(cols_to_summ)
```

2. `!!!`

- `!!!` 연산자를 쓰는 이유는, 직접함수를 만들 때 필요한 것으로 입력한 인자를 잘 인식시키기 위한 과정이라 생각하면 된다.
- `syms()`로 만들어진 다수의 symbol 리스트를 `group_by()`나 `pivot_longer()` 등의 인자에 '개별 인자'처럼 인식해준다.

```
tab1 <- df %>% group_by(!!!col1, !!!col2)
```

즉, `!!!col1` 은 "col1이 가진 모든 symbol을 풀어서 인자로 넣어달라"라는 뜻이다.

예를 들어 `col1`이 `list(quote(SEX), quote(YEAR))`라면, `group_by(!!!col1)`는 `group_by(SEX, YEAR)`로 동작하게 된다.

3. `summarise(across...`

- 각 그룹별로 `cols_to_summ`에 지정된 열에 대해, `stat_fun`을 적용해 요약한다.
- 결과 열 이름은 `"{.col}-{.fn}"` 형태로 설정된다.

```
summarise(across(all_of(cols_to_summ), stat_fun, .names = "{.col}-{.fn}"))
```

4. `pivot_longer(cols = ...`

- 요약 후 만들어진 여러 통계량 열(`ITEM_F001-mean`, `ITEM_F001-sd` 등)을 세로 형태(long format)로 펼친다.
- `names_sep = "-"`를 기준으로 ITEM과 STAT 두 부분으로 분리.
- 예: `ITEM_F001-mean` → ITEM = `ITEM_F001`, STAT = `mean`.

```
pivot_longer(cols = -c(!!!col1, !!!col2), names_to = c("ITEM", "STAT"), names_sep = "-", values_to = "v
```

`cols = -c(!!!col1, !!!col2)`는 그룹 열을 제외한 모든 열을 `pivot_longer` 대상으로 삼는 것.

5. `dplyr::arrange(ITEM)`

`dplyr`패키지 안에 `arrange`함수를 쓴다는 의미이다.

- `ITEM`열 기준으로 정렬(오름차순)한다.

6. `dplyr::ungroup()`

- 그룹화를 해제한다(추가 연산에서 혼동을 막기 위해).

7. `if (stat_to_column) { ... } else { ... }`

- `stat_to_column`이 `TRUE`라면, `cols_to_col`과 `STAT` 모두를 열 이름으로 펼친다.
- 예: `[ SEX, YEAR, STAT ]` 조합이 열이 됨.

```
if (stat_to_column) { tab1 <- tab1 %>% pivot_wider(names_from = all_of(c(cols_to_col, "STAT")), values_
```

- `stat_to_column = FALSE`이면, `cols_to_col`만 열 이름으로 사용해서 펼친다.
- 결과적으로 STAT은 데이터 행으로 남는다.

ITEM	TEST_SEX	TEST_AGE	STAT	2017	2018	2019	2020	2021
ITEM_F001	남성	13	개수	16.000000	20.000000	26.000000	1.000000	2.000000
ITEM_F001	남성	13	평균	162.350000	164.715000	164.892308	182.800000	159.400000
ITEM_F001	남성	13	표준편차	7.448937	6.895328	8.639024	NA	6.222540
ITEM_F001	남성	14	개수	16.000000	17.000000	32.000000	3.000000	2.000000
ITEM_F001	남성	14	평균	165.456250	168.505882	167.603125	169.433333	170.600000
ITEM_F001	남성	14	표준편차	5.399873	5.318067	6.567539	4.309679	11.455130
ITEM_F001	남성	15	개수	16.000000	22.000000	24.000000	3.000000	2.000000
ITEM_F001	남성	15	평균	269.781250	170.977273	173.600000	170.900000	169.700000
ITEM_F001	남성	15	표준편차	397.692213	6.552058	5.362592	4.622770	6.646804
ITEM_F001	남성	16	개수	15.000000	32.000000	23.000000	4.000000	7.000000
ITEM_F001	남성	16	평균	173.200000	173.031250	173.539130	170.825000	175.342857
ITEM_F001	남성	16	표준편차	7.156915	4.476421	6.486751	6.439138	2.869296
ITEM_F001	남성	17	개수	11.000000	26.000000	30.000000	5.000000	10.000000
ITEM_F001	남성	17	평균	175.000000	173.450000	173.573333	174.260000	173.140000
ITEM_F001	남성	17	표준편차	4.850979	4.888374	6.267262	4.496443	6.764154
ITEM_F001	남성	18	개수	11.000000	14.000000	5.000000	4.000000	6.000000
ITEM_F001	남성	18	평균	174.527273	174.628571	171.580000	169.325000	176.533333
ITEM_F001	남성	18	표준편차	5.833196	5.844054	3.834319	3.718759	2.445949
ITEM_F001	여성	13	개수	17.000000	11.000000	16.000000	2.000000	5.000000
ITEM_F001	여성	13	평균	157.288235	156.290909	156.493750	158.900000	157.360000

```
else { tab1 <- tab1 %>% pivot_wider(names_from = all_of(cols_to_col), values_from = value) }
```

설명만 보았을 때는 어려우니 뒤에 결과물을 보고 이해해보자.

3.3.6 직접만든 함수 적용

첫번 째 통계표의 경우는

- 행(row) 그룹핑 기준: TEST_SEX(성별), TEST_AGE(나이)
- 열(column) 그룹핑 기준: TEST_YEAR(검사연도)
- 통계 함수를 적용할 열들: ITEM_F001, ITEM_F002
- 계산할 함수: my_summ_func_2

로 표를 만들어준다.

```
tab1 <- summ_function_general(df1_youth,
                              c("TEST_SEX", "TEST_AGE"),
                              c("TEST_YEAR"),
                              c("ITEM_F001", "ITEM_F002"),
                              stat_fun = my_summ_func_2)
```

```
`summarise()` has grouped output by 'TEST_SEX', 'TEST_AGE'. You can override
using the `.groups` argument.
```

ITEM	TEST_AGE	STAT	2017_남성	2017_여성	2018_남성	2018_여성	2019_남성	2019_여성
ITEM_F001	13	개수	16.000000	17.000000	20.000000	11.000000	26.000000	16.000000
ITEM_F001	13	평균	162.350000	157.288235	164.715000	156.290909	164.892308	156.493750
ITEM_F001	13	표준편차	7.448937	7.663214	6.895328	5.826054	8.639024	3.423150
ITEM_F001	14	개수	16.000000	13.000000	17.000000	16.000000	32.000000	15.000000
ITEM_F001	14	평균	165.456250	159.900000	168.505882	159.350000	167.603125	161.040000
ITEM_F001	14	표준편차	5.399873	7.355723	5.318067	4.665047	6.567539	4.529869
ITEM_F001	15	개수	16.000000	18.000000	22.000000	13.000000	24.000000	22.000000
ITEM_F001	15	평균	269.781250	162.316667	170.977273	162.400000	173.600000	160.622727
ITEM_F001	15	표준편차	397.692213	5.818353	6.552058	6.858450	5.362592	5.114678
ITEM_F001	16	개수	15.000000	14.000000	32.000000	24.000000	23.000000	21.000000
ITEM_F001	16	평균	173.200000	159.314286	173.031250	160.920833	173.539130	159.680952
ITEM_F001	16	표준편차	7.156915	5.439154	4.476421	5.050095	6.486751	5.956309
ITEM_F001	17	개수	11.000000	15.000000	26.000000	19.000000	30.000000	17.000000
ITEM_F001	17	평균	175.000000	160.960000	173.450000	161.984210	173.573333	159.347059
ITEM_F001	17	표준편차	4.850979	4.036760	4.888374	3.481102	6.267262	6.197491
ITEM_F001	18	개수	11.000000	6.000000	14.000000	7.000000	5.000000	5.000000
ITEM_F001	18	평균	174.527273	161.150000	174.628571	160.471429	171.580000	163.080000
ITEM_F001	18	표준편차	5.833196	6.338060	5.844054	6.273945	3.834319	4.930213
ITEM_F002	13	개수	16.000000	17.000000	20.000000	11.000000	26.000000	16.000000
ITEM_F002	13	평균	56.943750	52.735294	64.265000	51.027273	63.730769	49.350000

```
head(tab1,20) %>% kbl() %>%kable_styling()
```

이후 tab2,tab3도 동일하게 알부분만 변경해서 통계표를 만든 결과이다.

```
tab2 <- summ_function_general(df1_youth,
                              c("TEST_AGE"),
                              c("TEST_YEAR", "TEST_SEX"),
                              c("ITEM_F001", "ITEM_F002"),
                              stat_fun = my_summ_func_2)
```

```
`summarise()` has grouped output by 'TEST_AGE', 'TEST_YEAR'. You can override
using the `.groups` argument.
```

```
head(tab2,20) %>% kbl() %>%kable_styling()
```

```
tab3 <- summ_function_general(df1_youth,
                              c("TEST_SEX", "TEST_AGE"),
                              c("TEST_YEAR"),
                              c("ITEM_F001", "ITEM_F002"),
                              stat_fun = my_summ_func_2,
                              stat_to_column = TRUE)
```

```
`summarise()` has grouped output by 'TEST_SEX', 'TEST_AGE'. You can override
using the `.groups` argument.
```

ITEM	TEST_SEX	TEST_AGE	2017_개수	2017_평균	2017_표준편차	2018_개수	2018_평균	2018_표
ITEM_F001	남성	13	16	162.35000	7.448937	20	164.71500	6.8
ITEM_F001	남성	14	16	165.45625	5.399873	17	168.50588	5.3
ITEM_F001	남성	15	16	269.78125	397.692213	22	170.97727	6.5
ITEM_F001	남성	16	15	173.20000	7.156915	32	173.03125	4.4
ITEM_F001	남성	17	11	175.00000	4.850979	26	173.45000	4.8
ITEM_F001	남성	18	11	174.52727	5.833196	14	174.62857	5.8
ITEM_F001	여성	13	17	157.28824	7.663214	11	156.29091	5.8
ITEM_F001	여성	14	13	159.90000	7.355723	16	159.35000	4.6
ITEM_F001	여성	15	18	162.31667	5.818353	13	162.40000	6.8
ITEM_F001	여성	16	14	159.31429	5.439154	24	160.92083	5.6
ITEM_F001	여성	17	15	160.96000	4.036760	19	161.98421	3.4
ITEM_F001	여성	18	6	161.15000	6.338060	7	160.47143	6.1
ITEM_F002	남성	13	16	56.94375	15.884289	20	64.26500	15.3
ITEM_F002	남성	14	16	59.13125	11.659086	17	63.07647	14.8
ITEM_F002	남성	15	16	104.26250	173.489050	22	66.11364	15.8
ITEM_F002	남성	16	15	66.78667	9.549336	32	73.22187	13.5
ITEM_F002	남성	17	11	77.20909	9.333215	26	67.40000	12.1
ITEM_F002	남성	18	11	67.73636	14.405573	14	73.68571	16.8
ITEM_F002	여성	13	17	52.73529	13.056413	11	51.02727	12.5
ITEM_F002	여성	14	13	53.63846	16.585965	16	52.63750	5.8

```
head(tab3,20) %>% kbl() %>%kable_styling()
```

4 통계표 출판

4.1 소개

통계표를 출력하기 위해 대표적으로 2가지 패키지가 존재한다.

1. `Quarto`

- 여러가지 방법으로 생산된 문서를 다양한 형식(html, pdf, docx, ···)으로 출판할 수 있도록 하는 저작도구.
- 마크다운 언어와 프로그래밍 언어를 같이 사용할 수 있는 기능.
- 지금 강의노트제작도 `Quarto`로 제작했다.

2. `flextable`

- 데이터프레임을 표(table)로 변환하여 다양한 형식(html, pdf, docx,···)으로 출판할 수 패키지.
- 표에 대한 다양한 형식과 여러가지 기능을 제공.
- 보고서 수준의 고품질 통계표를 출력 가능.

4.1.1 `flextable`로 출력방법설명

`flextable`을 이용해 아래한글(`HWP`)까지 출력하는 방법을 소개해보고자 한다.

결과표 또는 통계표를 갖고 `flextable`을 이용해 출력하는 방법은 다음과 같다.

1. 프로그램 R을 이용해 원하는 통계결과표를 작성한다.
2. `flextable`을 이용해 워드(`docx`)로 출력한다.
3. 워드(`docx`)로 출력한 표를 아래한글(`HWP`)로 출력한다.

방법을 알았으니 `flextable`을 쓰는 방법을 다음 섹션에서 확인해보자.

4.1.2 간단한 `flextable` 예시

이전 챕터 `dplyr`,`tidyr`패키지의 함수소개에서 쓰인 데이터프레임을 갖고 `flextable`을 보이는 함수를 소개한다.

```
library(flextable)

df <- data.frame(
  이름 = c("홍길동", "김영희", "박찬호", "이소라", "최민식"),
  나이 = c(25, 30, 35, 40, 28),
  도시 = c("서울", "부산", "서울", "인천", "부산"),
  점수 = c(80, 90, 75, 82, 95)
)
```

1. 기본 flextable 생성

데이터프레임 df를 가장 기본적인 flextable 형태로 보여준다.

```
ft1 <- flextable(df)
ft1
```

이름	나이	도시	점수
홍길동	25	서울	80
김영희	30	부산	90
박찬호	35	서울	75
이소라	40	인천	82
최민식	28	부산	95

2. 헤더(header) 스타일 및 테마 적용

테이블에서 헤더부분의 행이름부분의 스타일을 설정하는 방법은 다음과 같다.

```
ft2 <- flextable(df)
# 헤더 부분을 굵게(bold) 설정
ft2 <- bold(ft2, part = "header")

# 헤더 배경 색 지정
ft2 <- bg(ft2, bg = "#F5F5F5", part = "header")

# 표 전체 너비를 자동으로 맞추기
ft2 <- autofit(ft2)

# 박스형 테마 적용(선 굵기가 약간 강조됨)
ft2 <- theme_box(ft2)

ft2
```

이름	나이	도시	점수
홍길동	25	서울	80
김영희	30	부산	90
박찬호	35	서울	75
이소라	40	인천	82
최민식	28	부산	95

- `bold(ft2, part = "header")`: 헤더를 볼드체로 만듦
- `bg(ft2, bg = "#F5F5F5", part = "header")`: 헤더에 배경색을 입힘
- `autofit()`: 열 너비를 데이터에 맞게 자동 조정
- `theme_box()`: 테이블에 살짝 박스형 테마 적용

3. 조건부 색상 적용

점수가 80 미만인 행 강조하는 코드는 다음과 같다.

```
ft3 <- flextable(df)

# 조건: 점수가 80 미만인 경우, 해당 행의 "점수" 셀 글씨 색을 빨간색으로
ft3 <- color(ft3, i = ~ 점수 < 80, j = "점수", color = "red")

# 조건: 점수가 80 미만인 경우, 해당 행의 배경을 옅은 분홍색으로
ft3 <- bg(ft3, i = ~ 점수 < 80, bg = "#FFF0F0")

# 조금 더 보기 좋게 자동 너비 조정
ft3 <- autofit(ft3)

ft3
```

이름	나이	도시	점수
홍길동	25	서울	80
김영희	30	부산	90
박찬호	35	서울	75
이소라	40	인천	82
최민식	28	부산	95

- `color(ft3, i = ~ 점수 < 80, j = "점수", color = "red")` 조건(점수 < 80)을 만족하는 행의 점수 열을 빨간색 글씨로.
- `bg(ft3, i = ~ 점수 < 80, bg = "#FFF0F0")` 동일한 조건을 만족하는 행의 전체 셀 배경을 옅은 분홍색으로.

4. 일부 열 정렬 및 숫자 포맷 지정

일부 열에 숫자를 우측으로 정렬하고 싶거나, 표에 제목을 삽입하고 싶은 경우는 다음과 같다.

```
ft4 <- flextable(df)

# 나이, 점수 열을 우측 정렬
ft4 <- align(ft4, j = c("나이", "점수"), align = "right", part = "body")

# 표의 제목 삽입
ft4 <- add_header_row(
  ft4,
  values = c("성적표 예시"),
  colwidths = 4
)

# 표 너비 자동 설정
ft4 <- autofit(ft4)

ft4
```

성적표 예시			
이름	나이	도시	점수
홍길동	25	서울	80
김영희	30	부산	90
박찬호	35	서울	75
이소라	40	인천	82
최민식	28	부산	95

- `align(..., align = "right")`: 숫자 열을 우측 정렬
- `add_header_row()`: 전체 표 위에 큰 헤더(타이틀) 한 줄 추가
- `colwidths = 4`는 한 줄을 4개의 컬럼(columns) 폭만큼 병합해서 하나로 만든다는 의미

4.2 통계표의 출력

이전 챕터에서 쓴초등학교 체력측정 자료의 통계표를 활용해보자.

3가지 스타일의 통계표를 보인 결과를 `flextable`로 출력해보자.

```
load(here::here("data", "physical100_data.RData"))
ls()
```

```
[1] "df"         "df1_youth" "ft1"        "ft2"        "ft3"        "ft4"
```

```
tab1 <- summ_function_general(df1_youth, c("TEST_SEX", "TEST_AGE"), c("TEST_YEAR"), c("ITEM_F001", "ITE

tab2 <- summ_function_general(df1_youth, c("TEST_AGE"), c("TEST_YEAR", "TEST_SEX"), c("ITEM_F001", "ITE

tab3 <- summ_function_general(df1_youth, c("TEST_SEX", "TEST_AGE"), c("TEST_YEAR"), c("ITEM_F001", "ITE
```

4.2.1 예제

tab1의 통계표를 다음과 같이 적용한다.

1. `tab1_cols <- colnames(tab1)`

tab1의 모든 열 이름을 추출해 `tab1_cols`에 저장한다.

이 값은 곧 `flextable(col_keys = tab1_cols)`에서 사용된다.

2. `tab1 %>% dplyr::group_by(ITEM, TEST_SEX, TEST_AGE)`

`tab1`를 `ITEM`, `TEST_SEX`, `TEST_AGE` 조합으로 그룹화한다.

이후 `mutate()` 등에서 그룹 내 계산을 수행할 수 있다.

3. `dplyr::mutate(is_last_val_in_group = row_number() == max(row_number()))`

각 그룹별(`ITEM`, `TEST_SEX`, `TEST_AGE`)로 **행 번호(row_number())**를 매기고, 그 번호가 그룹 내 최대값이면 `TRUE`를, 아니면 `FALSE`를 `is_last_val_in_group` 컬럼에 저장한다.

4. `flextable(col_keys = tab1_cols )`

`flextable` 패키지로 `tab1`를 테이블 형태로 변환한다.

`col_keys = tab1_cols`로 지정해, 모든 열(이름은 `tab1_cols`)을 테이블에 표시한다.

5. `merge_v(j = 1:3)`

`j = 1:3`는 테이블의 첫 번째 부터 세 번째 열에 대해 세로 병합을 적용하겠다는 의미이다.

6. `autofit()`

셀 너비와 높이를 자동으로 맞추도록 설정한다.

7. `theme_booktabs(bold_header = TRUE)`

`flextable`의 테마를 "booktabs" 스타일로 지정하고, 헤더(header)를 볼드체로 만든다.

8. `align(align = "left", part = "header", j =1:3)`

테이블의 헤더 부분(`part = "header"`) 중, 첫 3개 열(`j =1:3`)을 왼쪽 정렬(`"left"`)한다.

9. `colformat_double(i = ~(STAT=="개수" | STAT=="결측개수"), digits = 0, big.mark = get_flextable_default$big.mark)`

숫자 서식 지정 함수로써,

`i = ~(STAT=="개수" | STAT=="결측개수")`: 조건식으로, STAT 열이 "개수" 또는 "결측개수"인 행을 골라낸다.

해당 셀들의 표시 소수점 자릿수(digits)를 0으로 설정 → 정수로 표현.

10. `colformat_double(i = ~!(STAT=="개수" | STAT=="결측개수"), digits = 2)`

바로 위와 비슷한 숫자 서식 지정이지만, 조건이 STAT가 "개수" 또는 "결측개수"가 아닌 경우(`!()`)이다.

소수점 자릿수를 2로 설정 → 소수 둘째 자리까지 표시한다.

11. `hline(i = ~is_last_val_in_group == TRUE, border = fp_border())`

가로줄(hline)을 추가한다.

`i = ~is_last_val_in_group == TRUE` → 각 그룹의 마지막 행에 가로줄을 그린다.

`fp_border()`는 flextable에서 제공하는 테두리(border) 설정 함수로, 기본 스타일의 선을 만든다.

12. `fix_border_issues()`

테이블 테두리 관련된 미세한 문제를 자동으로 조정(수정)해 주는 함수이다.

```
tab1_cols <- colnames(tab1)
AA <-  tab1 %>%
    dplyr::group_by(ITEM, TEST_SEX, TEST_AGE) %>%
    dplyr::mutate(is_last_val_in_group = row_number() == max(row_number()))  %>%
    flextable(col_keys = tab1_cols ) %>%
    merge_v( j = 1:3 )  %>%
    autofit()  %>%
    theme_booktabs(bold_header = TRUE) %>%
    align(align = "left", part = "header", j =1:3) %>%
    colformat_double(i = ~(`STAT`=="개수" | `STAT`=="결측개수" ) , digits = 0,
                     big.mark =     get_flextable_defaults()$big.mark) %>%
    colformat_double(i = ~!(`STAT`=="개수" | `STAT`=="결측개수" ), digits = 2) %>%
    hline(i = ~is_last_val_in_group == TRUE, border = fp_border()) %>%
    fix_border_issues()

AA
```

ITEM	TEST_SEX	TEST_AGE	STAT	2017	2018	2019
			개수	16	20	2
		13	평균	162.35	164.72	164.8
			표준편차	7.45	6.90	8.6
			개수	16	17	3
		14	평균	165.46	168.51	167.6
			표준편차	5.40	5.32	6.5

ITEM	TEST_SEX	TEST_AGE	STAT	2017	2018	2019
ITEM_F001	남성	15	개수	16	22	24
			평균	269.78	170.98	173.60
			표준편차	397.69	6.55	5.30
		16	개수	15	32	23
			평균	173.20	173.03	173.54
			표준편차	7.16	4.48	6.49
		17	개수	11	26	30
			평균	175.00	173.45	173.57
			표준편차	4.85	4.89	6.27
		18	개수	11	14	5
			평균	174.53	174.63	171.58
			표준편차	5.83	5.84	3.83
	여성	13	개수	17	11	10
			평균	157.29	156.29	156.49
			표준편차	7.66	5.83	3.41
		14	개수	13	16	15
			평균	159.90	159.35	161.04
			표준편차	7.36	4.67	4.53
		15	개수	18	13	22
			평균	162.32	162.40	160.62
			표준편차	5.82	6.86	5.11
		16	개수	14	24	21
			평균	159.31	160.92	159.68
			표준편차	5.44	5.05	5.90
		17	개수	15	19	17
			평균	160.96	161.98	159.33
			표준편차	4.04	3.48	6.20
		18	개수	6	7	5
			평균	161.15	160.47	163.08
			표준편차	6.34	6.27	4.93

ITEM	TEST_SEX	TEST_AGE	STAT	2017	2018	2019
			개수	16	20	2
		13	평균	56.94	64.27	63.7
			표준편차	15.88	15.55	16.5
			개수	16	17	3
		14	평균	59.13	63.08	62.9
			표준편차	11.66	14.85	13.3
			개수	16	22	2
		15	평균	104.26	66.11	68.0
			표준편차	173.49	15.83	14.0
	남성		개수	15	32	2
		16	평균	66.79	73.22	65.8
			표준편차	9.55	13.52	9.8
			개수	11	26	3
		17	평균	77.21	67.40	71.2
			표준편차	9.33	12.21	21.5
			개수	11	14	
		18	평균	67.74	73.69	76.0
			표준편차	14.41	16.88	22.1
			개수	17	11	1
		13	평균	52.74	51.03	49.3
			표준편차	13.06	12.55	7.3
			개수	13	16	1
		14	평균	53.64	52.64	58.5
			표준편차	16.59	5.89	14.1
			개수	18	13	2
		15	평균	58.88	61.78	56.9
			표준편차	11.45	11.75	7.0
			개수	14	24	2
		16	평균	56.21	57.67	58.6
			표준편차	8.17	7.25	12.1

여성

ITEM	TEST_SEX	TEST_AGE	STAT	2017	2018	2019
			개수	15	19	1
		17	평균	62.97	55.25	61.2
			표준편차	13.19	13.89	16.3
			개수	6	7	
		18	평균	61.92	58.01	63.8
			표준편차	16.52	9.31	10.7

이러한 형태로 다음 통계표도 비슷하게 결과를 낼 수 있다.

하지만 매번 통계표를 구할 때 복잡한 코드를 입력하기 어려우니 이 경우에도 함수를 만들어서 간편하게 돌려보자.

```
tab2_cols <- colnames(tab2)
BB <-  tab2 %>%
    dplyr::group_by(ITEM, TEST_AGE) %>%
    dplyr::mutate(is_last_val_in_group = row_number() == max(row_number()))  %>%
    flextable(col_keys = tab2_cols ) %>%
    merge_v( j = 1:2 )  %>%
    valign(j = 1:2, valign = "top") %>%
    separate_header() %>%
    autofit()  %>%
    theme_booktabs(bold_header = TRUE) %>%
    align(align = "left", part = "header", j =1:3) %>%
    colformat_double(i = ~(`STAT`=="개수" | `STAT`=="결측개수" ) , digits = 0,
                     big.mark =     get_flextable_defaults()$big.mark) %>%
    colformat_double(i = ~!(`STAT`=="개수" | `STAT`=="결측개수" ), digits = 2) %>%
    hline(i = ~is_last_val_in_group == TRUE, border = fp_border()) %>%
    fix_border_issues()

BB
```

	TEST		2017		2018	
ITEM	AGE	STAT	남성	여성	남성	여성
	13	개수	16	17	20	11
		평균	162.35	157.29	164.72	156.29
		표준편차	7.45	7.66	6.90	5.83
	14	개수	16	13	17	16
		평균	165.46	159.90	168.51	159.35
		표준편차	5.40	7.36	5.32	4.67

ITEM	TEST AGE	STAT	2017 남성	2017 여성	2018 남성	2018 여성
	15	개수	16	18	22	13
		평균	269.78	162.32	170.98	162.40
		표준편차	397.69	5.82	6.55	6.86
	16	개수	15	14	32	24
		평균	173.20	159.31	173.03	160.92
		표준편차	7.16	5.44	4.48	5.05
	17	개수	11	15	26	19
		평균	175.00	160.96	173.45	161.98
		표준편차	4.85	4.04	4.89	3.48
	18	개수	11	6	14	7
		평균	174.53	161.15	174.63	160.47
		표준편차	5.83	6.34	5.84	6.27
	13	개수	16	17	20	11
		평균	56.94	52.74	64.27	51.03
		표준편차	15.88	13.06	15.55	12.55
	14	개수	16	13	17	16
		평균	59.13	53.64	63.08	52.64
		표준편차	11.66	16.59	14.85	5.89
	15	개수	16	18	22	13
		평균	104.26	58.88	66.11	61.78
		표준편차	173.49	11.45	15.83	11.75
	16	개수	15	14	32	24
		평균	66.79	56.21	73.22	57.67
		표준편차	9.55	8.17	13.52	7.25
	17	개수	11	15	26	19
		평균	77.21	62.97	67.40	55.25
		표준편차	9.33	13.19	12.21	13.89
		개수	11	6	14	7

ITEM	TEST AGE	18 STAT	2017 남성	2017 여성	2018 남성	2018 여성
		평균	67.74	61.92	73.69	58.01
		표준편차	14.41	16.52	16.88	9.31

```
tab3_cols <- colnames(tab3)
CC <-  tab3 %>%
    dplyr::group_by(ITEM, TEST_SEX) %>%
    dplyr::mutate(is_last_val_in_group = row_number() == max(row_number()))  %>%
    flextable(col_keys = tab3_cols ) %>%
    merge_v( j = 1:2 )  %>%
    valign(j = 1:2, valign = "top") %>%
    separate_header() %>%
    autofit()  %>%
    theme_booktabs(bold_header = TRUE) %>%
    align(align = "left", part = "header", j =1:3) %>%
    #colformat_double(i = ~(`STAT`=="개수" | `STAT`=="결측개수" ) , digits = 0,
    #                 big.mark =     get_flextable_defaults()$big.mark) %>%
    #colformat_double(i = ~!(`STAT`=="개수" | `STAT`=="결측개수" ), digits = 2) %>%
    hline(i = ~is_last_val_in_group == TRUE, border = fp_border()) %>%
    fix_border_issues()

CC
```

ITEM	TEST SEX	AGE	2017 개수	2017 평균	2017 표준편차	개수	평균	표
	남성	13	16	162.35000	7.448937	20	164.71500	6.89
		14	16	165.45625	5.399873	17	168.50588	5.31
		15	16	269.78125	397.692213	22	170.97727	6.55
		16	15	173.20000	7.156915	32	173.03125	4.47
		17	11	175.00000	4.850979	26	173.45000	4.88
		18	11	174.52727	5.833197	14	174.62857	5.84
		13	17	157.28824	7.663214	11	156.29091	5.82
		14	13	159.90000	7.355723	16	159.35000	4.66
		15	18	162.31667	5.818353	13	162.40000	6.85
		16	14	159.31429	5.439154	24	160.92083	5.05
		17	15	160.96000	4.036760	19	161.98421	3.48

여성

	TEST							
						2017		
ITEM	SEX	AGE	개수	평균	표준편차	개수	평균	표
		18	6	161.15000	6.338060	7	160.47143	6.2
ITEM_F002	남성	13	16	56.94375	15.884289	20	64.26500	15.5
		14	16	59.13125	11.659086	17	63.07647	14.8
		15	16	104.26250	173.489050	22	66.11364	15.8
		16	15	66.78667	9.549336	32	73.22187	13.5
		17	11	77.20909	9.333215	26	67.40000	12.2
		18	11	67.73636	14.405573	14	73.68571	16.8
	여성	13	17	52.73529	13.056413	11	51.02727	12.5
		14	13	53.63846	16.585965	16	52.63750	5.8
		15	18	58.88333	11.447849	13	61.77692	11.7
		16	14	56.20714	8.167804	24	57.67500	7.2
		17	15	62.96667	13.193487	19	55.25263	13.8
		18	6	61.91667	16.521794	7	58.01429	9.3

4.2.2 함수만들기 (함수보완수정추후 예정)

`flextable`도 `dplyr`,`tidyr`와 마찬가지로 여러가지 세부조정을 할수 있기에 코드가 많이 복잡할 수 있다.

또한, 통계표를 1개만 출력하는 것이 아니기에 여러가지 통계표를 출력하는 데 비효율적이므로 함수를 만들어서 간편하게 통계표를 출력해보자.

`table_function_general`은 예제에 나온 코드와의 차이점은

- "이 데이터프레임의 상위 몇 개 열을 '행 그룹'으로 지정할 것인지"를 `num_group_rows`로 조절한다.
- 직접함수를 만들 때 필요한 것으로 입력한 인자를 잘 인식시키기 위해 `!!!`와 `syms`를 적용했다.

```
table_function_general <- function(df, num_group_rows) {

    table_cols <- colnames(df)
    col1 <- syms(table_cols[1:num_group_rows])

ft1 <-  df %>%

    dplyr::group_by(!!!col1) %>%
    dplyr::mutate(is_last_val_in_group = row_number() == max(row_number()))  %>%
    flextable(col_keys = table_cols ) %>%
    merge_v( j = 1:num_group_rows )  %>%
```

```
    valign(j = 1:(num_group_rows), valign = "top") %>%
    separate_header() %>%
    autofit()  %>%
    theme_booktabs(bold_header = TRUE) %>%
    align(align = "left", part = "header", j =1:num_group_rows) %>%
    #colformat_double(i = ~(`STAT`=="개수" | `STAT`=="결측개수" ) , digits = 0,
    #                 big.mark =     get_flextable_defaults()$big.mark) %>%
    #colformat_double(i = ~!(`STAT`=="개수" | `STAT`=="결측개수" ), digits = 2) %>%
    hline(i = ~is_last_val_in_group == TRUE, border = fp_border()) %>%
    fix_border_issues()

    return(ft1)
}
```

직접 만든 함수를 사용하여 다음과 같이 통계표를 쉽게 구할 수 있다.

```
mytable1 <- table_function_general(tab1, 3)
mytable1
```

ITEM	TEST SEX	AGE	STAT	2017	2018	2019	
		13	개수	16.000000	20.000000	26.000000	1.000
			평균	162.350000	164.715000	164.892308	182.800
			표준편차	7.448937	6.895328	8.639024	
		14	개수	16.000000	17.000000	32.000000	3.000
			평균	165.456250	168.505882	167.603125	169.433
			표준편차	5.399873	5.318067	6.567539	4.309
		15	개수	16.000000	22.000000	24.000000	3.000
			평균	269.781250	170.977273	173.600000	170.900
			표준편차	397.692213	6.552058	5.362592	4.622
		16	개수	15.000000	32.000000	23.000000	4.000
			평균	173.200000	173.031250	173.539130	170.825
			표준편차	7.156915	4.476421	6.486751	6.439
		17	개수	11.000000	26.000000	30.000000	5.000
			평균	175.000000	173.450000	173.573333	174.260
			표준편차	4.850979	4.888374	6.267262	4.496

ITEM	TEST SEX	AGE	STAT	2017	2018	2019	
		18	개수	11.000000	14.000000	5.000000	4.000
			평균	174.527273	174.628571	171.580000	169.325
			표준편차	5.833197	5.844054	3.834319	3.718
	여성	13	개수	17.000000	11.000000	16.000000	2.000
			평균	157.288235	156.290909	156.493750	158.900
			표준편차	7.663214	5.826054	3.423150	4.949
		14	개수	13.000000	16.000000	15.000000	2.000
			평균	159.900000	159.350000	161.040000	155.600
			표준편차	7.355723	4.665047	4.529869	7.778
		15	개수	18.000000	13.000000	22.000000	2.000
			평균	162.316667	162.400000	160.622727	159.950
			표준편차	5.818353	6.858450	5.114678	3.181
		16	개수	14.000000	24.000000	21.000000	2.000
			평균	159.314286	160.920833	159.680952	158.650
			표준편차	5.439154	5.050095	5.956309	0.494
		17	개수	15.000000	19.000000	17.000000	3.000
			평균	160.960000	161.984211	159.347059	161.000
			표준편차	4.036760	3.481102	6.197491	1.562
		18	개수	6.000000	7.000000	5.000000	2.000
			평균	161.150000	160.471429	163.080000	164.500
			표준편차	6.338060	6.273945	4.930213	5.656
		13	개수	16.000000	20.000000	26.000000	1.000
			평균	56.943750	64.265000	63.730769	97.500
			표준편차	15.884289	15.545562	16.570716	
		14	개수	16.000000	17.000000	32.000000	3.000
			평균	59.131250	63.076471	62.968750	63.066
			표준편차	11.659086	14.848086	13.394256	8.277
			개수	16.000000	22.000000	24.000000	3.000

ITEM	TEST SEX	AGE		STAT	2017	2018	2019	
				평균	104.262500	66.113636	68.091667	72.566
				표준편차	173.489050	15.825425	14.052446	8.393
			16	개수	15.000000	32.000000	23.000000	4.000
				평균	66.786667	73.221875	65.821739	60.875
				표준편차	9.549336	13.522806	9.863153	4.680
			17	개수	11.000000	26.000000	30.000000	5.000
				평균	77.209091	67.400000	71.206667	71.560
				표준편차	9.333215	12.206883	21.510125	10.260
			18	개수	11.000000	14.000000	5.000000	4.000
				평균	67.736364	73.685714	76.060000	76.500
				표준편차	14.405573	16.877015	22.125845	8.869
			13	개수	17.000000	11.000000	16.000000	2.000
				평균	52.735294	51.027273	49.350000	53.000
				표준편차	13.056413	12.546481	7.319381	4.242
			14	개수	13.000000	16.000000	15.000000	2.000
				평균	53.638462	52.637500	58.500000	62.650
				표준편차	16.585965	5.889298	14.159600	2.050
			15	개수	18.000000	13.000000	22.000000	2.000
				평균	58.883333	61.776923	56.900000	61.400
				표준편차	11.447849	11.747919	7.043065	19.657
			16	개수	14.000000	24.000000	21.000000	2.000
				평균	56.207143	57.675000	58.652381	56.750
				표준편차	8.167804	7.253979	12.105933	5.161
			17	개수	15.000000	19.000000	17.000000	3.000
				평균	62.966667	55.252632	61.223529	56.700
				표준편차	13.193487	13.886499	16.329671	13.346
				개수	6.000000	7.000000	5.000000	2.000
				평균	61.916667	58.014286	63.880000	54.300

	TEST	18				
ITEM	SEX	AGE	STAT	2017	2018	2019
			표준편차	16.521794	9.313866	10.742998

```
mytable2 <- table_function_general(tab2, 2)
mytable2
```

	TEST			2017		2018	
ITEM	AGE		STAT	남성	여성	남성	여성
ITEM_F001	13		개수	16.000000	17.000000	20.000000	11.000000
			평균	162.350000	157.288235	164.715000	156.290909
			표준편차	7.448937	7.663214	6.895328	5.826054
	14		개수	16.000000	13.000000	17.000000	16.000000
			평균	165.456250	159.900000	168.505882	159.350000
			표준편차	5.399873	7.355723	5.318067	4.665047
	15		개수	16.000000	18.000000	22.000000	13.000000
			평균	269.781250	162.316667	170.977273	162.400000
			표준편차	397.692213	5.818353	6.552058	6.858450
	16		개수	15.000000	14.000000	32.000000	24.000000
			평균	173.200000	159.314286	173.031250	160.920833
			표준편차	7.156915	5.439154	4.476421	5.050095
	17		개수	11.000000	15.000000	26.000000	19.000000
			평균	175.000000	160.960000	173.450000	161.984211
			표준편차	4.850979	4.036760	4.888374	3.481102
	18		개수	11.000000	6.000000	14.000000	7.000000
			평균	174.527273	161.150000	174.628571	160.471429
			표준편차	5.833197	6.338060	5.844054	6.273945
	13		개수	16.000000	17.000000	20.000000	11.000000
			평균	56.943750	52.735294	64.265000	51.027273
			표준편차	15.884289	13.056413	15.545562	12.546481
			개수	16.000000	13.000000	17.000000	16.000000

ITEM	TEST AGE	14 STAT	2017 남성	2017 여성	2018 남성	2018 여성
		평균	59.131250	53.638462	63.076471	52.637500
		표준편차	11.659086	16.585965	14.848086	5.889298
	15	개수	16.000000	18.000000	22.000000	13.000000
		평균	104.262500	58.883333	66.113636	61.776923
		표준편차	173.489050	11.447849	15.825425	11.747919
	16	개수	15.000000	14.000000	32.000000	24.000000
		평균	66.786667	56.207143	73.221875	57.675000
		표준편차	9.549336	8.167804	13.522806	7.253979
	17	개수	11.000000	15.000000	26.000000	19.000000
		평균	77.209091	62.966667	67.400000	55.252632
		표준편차	9.333215	13.193487	12.206883	13.886499
	18	개수	11.000000	6.000000	14.000000	7.000000
		평균	67.736364	61.916667	73.685714	58.014286
		표준편차	14.405573	16.521794	16.877015	9.313866

```
mytable3 <- table_function_general(tab3, 2)
mytable3
```

ITEM	TEST SEX	AGE	2017 개수	2017 평균	2017 표준편차	2017 개수	2017 평균	표
	남성	13	16	162.35000	7.448937	20	164.71500	6.89
		14	16	165.45625	5.399873	17	168.50588	5.31
		15	16	269.78125	397.692213	22	170.97727	6.55
		16	15	173.20000	7.156915	32	173.03125	4.47
		17	11	175.00000	4.850979	26	173.45000	4.88
		18	11	174.52727	5.833197	14	174.62857	5.84
		13	17	157.28824	7.663214	11	156.29091	5.82
		14	13	159.90000	7.355723	16	159.35000	4.66
		15	18	162.31667	5.818353	13	162.40000	6.85

여성

	TEST					2017		
ITEM	SEX	AGE	개수	평균	표준편차	개수	평균	표
		16	14	159.31429	5.439154	24	160.92083	5.0
		17	15	160.96000	4.036760	19	161.98421	3.4
		18	6	161.15000	6.338060	7	160.47143	6.2
ITEM_F002	남성	13	16	56.94375	15.884289	20	64.26500	15.5
		14	16	59.13125	11.659086	17	63.07647	14.8
		15	16	104.26250	173.489050	22	66.11364	15.8
		16	15	66.78667	9.549336	32	73.22187	13.5
		17	11	77.20909	9.333215	26	67.40000	12.2
		18	11	67.73636	14.405573	14	73.68571	16.8
	여성	13	17	52.73529	13.056413	11	51.02727	12.5
		14	13	53.63846	16.585965	16	52.63750	5.8
		15	18	58.88333	11.447849	13	61.77692	11.7
		16	14	56.20714	8.167804	24	57.67500	7.2
		17	15	62.96667	13.193487	19	55.25263	13.8
		18	6	61.91667	16.521794	7	58.01429	9.3

4.3 아래한글(HWP)로 내보내기

이제 `flextable`로 출력한 표를 아래한글로 내보내는 과정을 확인해보자.

아래한글은 우리나라에서만 쓰이는 문서프로그램이기에 `flextable`에서 직접 아래한글로 표를 내보내는 것은 없다.

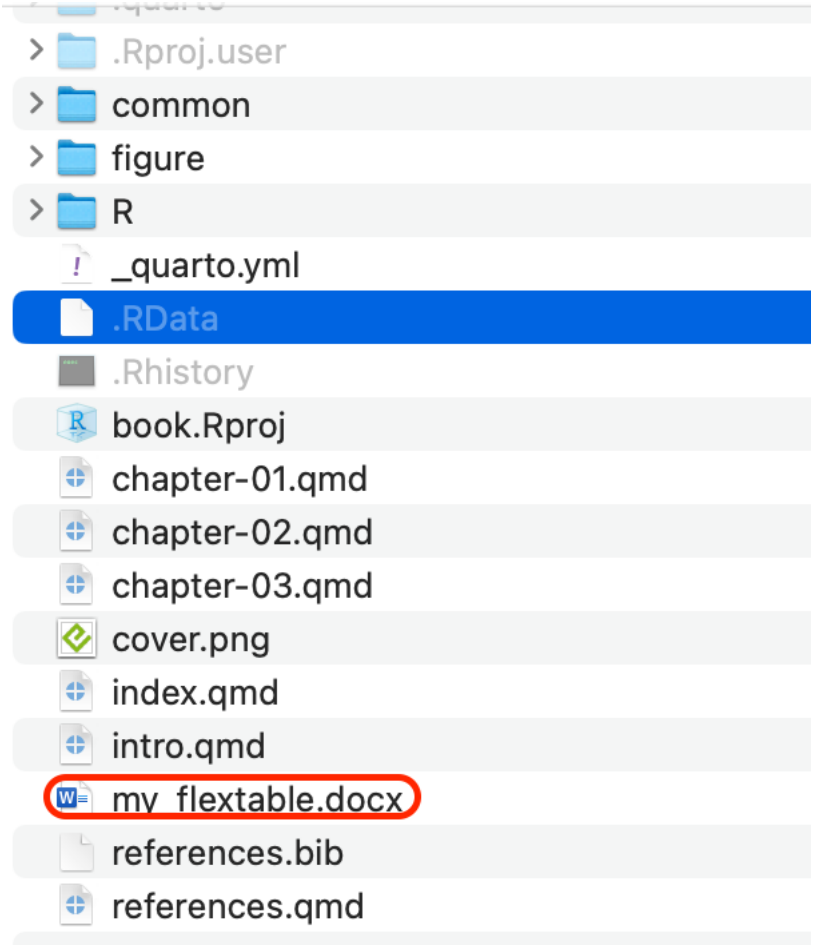
그렇기에 워드(`WORD`)로 내보낸 표를 아래한글로 가져오면 된다.

가장 간단한 방법으로 워드로 내보낸 표를 복사 붙여넣기를 통해 아래한글로 내보낼 수 있다.

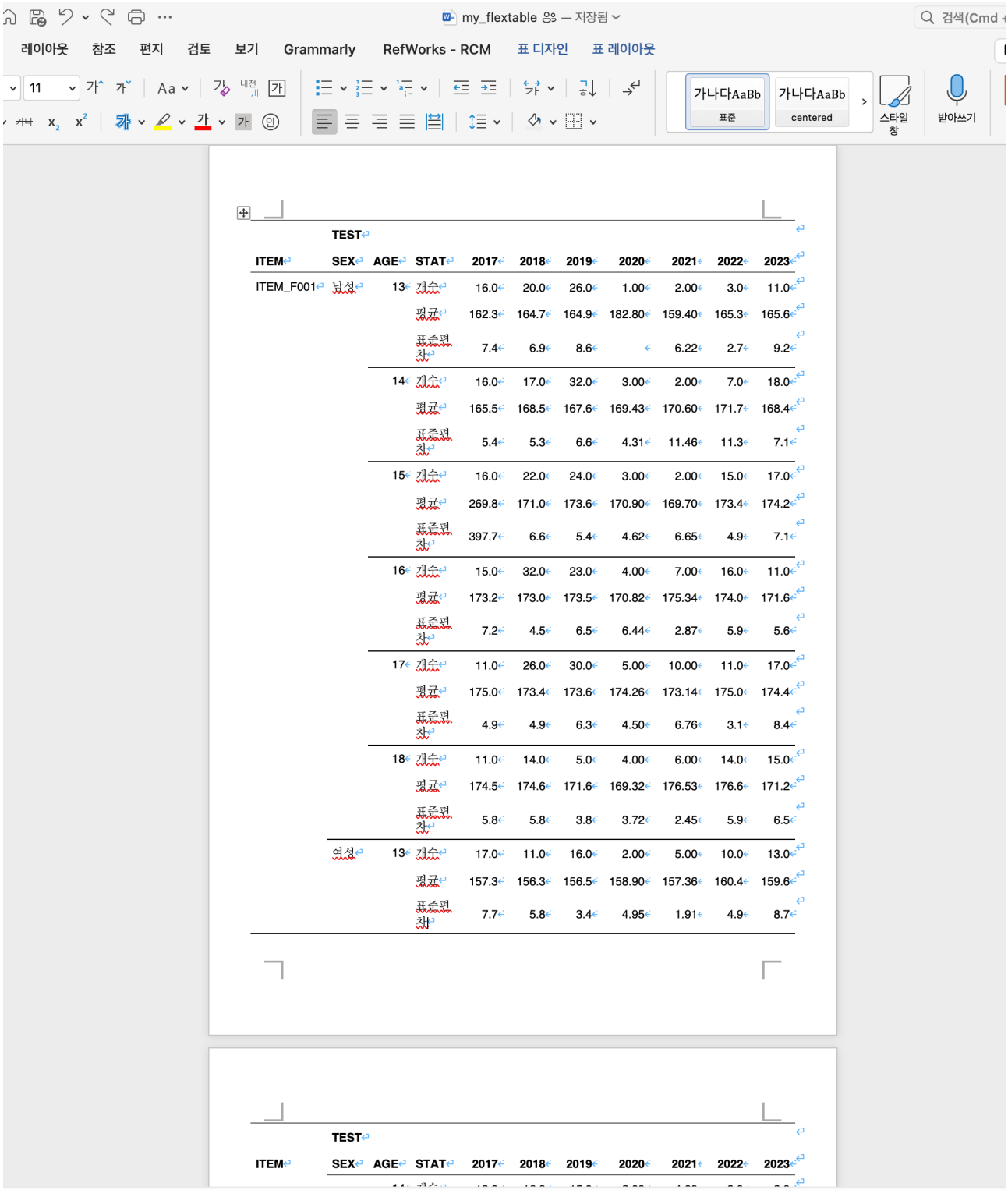
`mytable1`를 워드(`WORD`)로 내보내는 코드는 다음과 같다.

```
# 새로운 Word 문서 객체 생성
my_doc <- read_docx()

# 본문(body)에 flextable 삽입
my_doc <- body_add_flextable(my_doc, mytable1)

# docx 파일로 내보내기
print(my_doc, target = "my_flextable.docx")
```

코드를 실행한 다음 R작업폴더에 들어가 보면 내가 지정한 `my_flextable.docx`파일이 생긴것을 알수 있다.



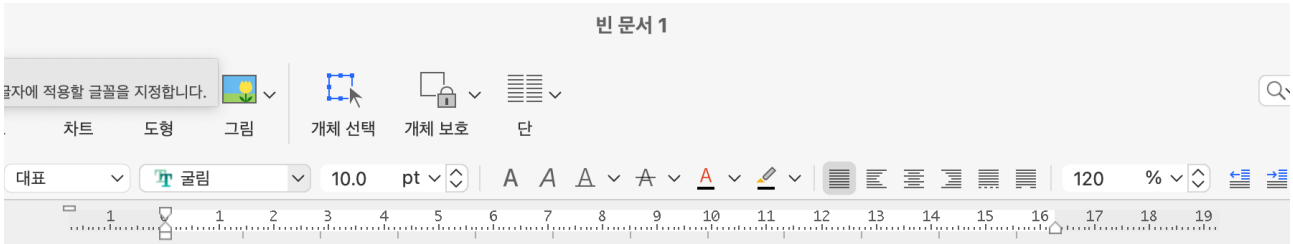
파일에 들어가보면 `flextable`로 만든 표가 있는 것을 확인 할수 있다.



		TEST							
ITEM	SEX	AGE	STAT	2017	2018	2019	2020	2021	2022	2023
ITEM_F001	남성	13	계수	16.0	20.0	26.0	1.00	2.00	3.0	11.0
			평균	162.3	164.7	164.9	182.80	159.40	165.3	165.6
			표준편차	7.4	6.9	8.6		6.22	2.7	9.2
		14	계수	16.0	17.0	32.0	3.00	2.00	7.0	18.0
			평균	165.5	168.5	167.6	169.43	170.60	171.7	168.4
			표준편차	5.4	5.3	6.6	4.31	11.46	11.3	7.1
		15	계수	16.0	22.0	24.0	3.00	2.00	15.0	17.0
			평균	269.8	171.0	173.6	170.90	169.70	173.4	174.2
			표준편차	397.7	6.6	5.4	4.62	6.65	4.9	7.1
		16	계수	15.0	32.0	23.0	4.00	7.00	16.0	11.0
			평균	173.2	173.0	173.5	170.82	175.34	174.0	171.6
			표준편차	7.2	4.5	6.5	6.44	2.87	5.9	5.6
		17	계수	11.0	26.0	30.0	5.00	10.00	11.0	17.0
			평균	175.0	173.4	173.6	174.26	173.14	175.0	174.4
			표준편차	4.9	4.9	6.3	4.50	6.76	3.1	8.4
		18	계수	11.0	14.0	5.0	4.00	6.00	14.0	15.0
			평균	174.5	174.6	171.6	169.32	176.53	176.6	171.2
			표준편차	5.8	5.8	3.8	3.72	2.45	5.9	6.5
	여성	13	계수	17.0	11.0	16.0	2.00	5.00	10.0	13.0
			평균	157.3	156.3	156.5	158.90	157.36	160.4	159.6
			표준편차	7.7	5.8	3.4	4.95	1.91	4.9	8.7

		TEST							
ITEM	SEX	AGE	STAT	2017	2018	2019	2020	2021	2022	2023

그리고 이 표를 복사해서 아래한글에 붙여넣기를 하면 아래한글로 표를 옮길수 있다.



ITEM	TEST SEX	AGE	STAT	2017	2018	2019	2020	2021	2022	2023
ITEM_F001	남성	13	개수	16.0	20.0	26.0	1.00	2.00	3.0	11.0
			평균	162.3	164.7	164.9	182.80	159.40	165.3	165.6
			표준편차	7.4	6.9	8.6		6.22	2.7	9.2
		14	개수	16.0	17.0	32.0	3.00	2.00	7.0	18.0
			평균	165.5	168.5	167.6	169.43	170.60	171.7	168.4
			표준편차	5.4	5.3	6.6	4.31	11.46	11.3	7.1
		15	개수	16.0	22.0	24.0	3.00	2.00	15.0	17.0
			평균	269.8	171.0	173.6	170.90	169.70	173.4	174.2
			표준편차	397.7	6.6	5.4	4.62	6.65	4.9	7.1
		16	개수	15.0	32.0	23.0	4.00	7.00	16.0	11.0
			평균	173.2	173.0	173.5	170.82	175.34	174.0	171.6
			표준편차	7.2	4.5	6.5	6.44	2.87	5.9	5.6
		17	개수	11.0	26.0	30.0	5.00	10.00	11.0	17.0
			평균	175.0	173.4	173.6	174.26	173.14	175.0	174.4
			표준편차	4.9	4.9	6.3	4.50	6.76	3.1	8.4
		18	개수	11.0	14.0	5.0	4.00	6.00	14.0	15.0
			평균	174.5	174.6	171.6	169.32	176.53	176.6	171.2
			표준편차	5.8	5.8	3.8	3.72	2.45	5.9	6.5
	여성	13	개수	17.0	11.0	16.0	2.00	5.00	10.0	13.0
			평균	157.3	156.3	156.5	158.90	157.36	160.4	159.6
			표준편차	7.7	5.8	3.4	4.95	1.91	4.9	8.7
		14	개수	13.0	16.0	15.0	2.00	4.00	8.0	8.0
			평균	159.9	159.3	161.0	155.60	158.32	161.9	159.0
			표준편차	7.4	4.7	4.5	7.78	5.31	5.0	6.9
		15	개수	18.0	13.0	22.0	2.00	1.00	8.0	18.0
			평균	162.3	162.4	160.6	159.95	162.00	160.7	160.4
			표준편차	5.8	6.9	5.1	3.18		3.7	3.9
		16	개수	14.0	24.0	21.0	2.00	3.00	7.0	19.0
			평균	159.3	160.9	159.7	158.65	165.07	159.1	161.4
			표준편차	5.4	5.1	6.0	0.49	0.85	7.6	4.3
		17	개수	15.0	19.0	17.0	3.00	6.00	8.0	14.0
			평균	161.0	162.0	159.3	161.00	158.73	162.4	162.4
			표준편차	4.0	3.5	6.2	1.56	3.75	5.0	5.1
		18	개수	6.0	7.0	5.0	2.00	3.00	11.0	12.0
			평균	161.2	160.5	163.1	164.50	156.20	162.9	160.4
			표준편차	6.3	6.3	4.9	5.66	3.59	7.1	8.4
ITEM_F002	남성	13	개수	16.0	20.0	26.0	1.00	2.00	3.0	11.0
			평균	56.9	64.3	63.7	97.50	72.85	60.9	64.4
			표준편차	15.9	15.5	16.6		36.42	9.2	20.2
		14	개수	16.0	17.0	32.0	3.00	2.00	7.0	18.0

5 부록

5.1 flextable 을 이용한 아래 한글 테이블 작성 요약

데이터가 요약된 결과를 flextable을 활용하여 한글(HWP) 문서로 출력하는 방법은 다음과 같이 세 단계로 이루어진다.

5.1.1 1. R에서 원하는 통계표 생성

먼저, R을 이용하여 원하는 통계 결과표를 만든다. 예를 들어, mtcars 데이터셋을 요약하여 표로 정리할 수 있다.

```
library(dplyr)
library(flextable)

# 데이터 요약
summary_table <- mtcars %>%
  group_by(cyl) %>%
  summarise(
    avg_mpg = mean(mpg),
    avg_hp = mean(hp),
    count = n()
  )

# `flextable`을 이용하여 보기 좋은 표로 변환
table_output <- flextable(summary_table)

# 기본 스타일 적용
table_output <- table_output %>%
  theme_vanilla() %>%
  autofit()

table_output
```

cyl	avg_mpg	avg_hp	count
4	26.66364	82.63636	11
6	19.74286	122.28571	7
8	15.10000	209.21429	14

5.1.2 2. MS Word 문서로 저장

한글 문서(.hwp)로 바로 저장하는 기능은 지원되지 않지만, 우선 MS Word(.docx) 파일로 저장한 후, 한글 문서로 변환하는 방식을 사용하면 된다.

```
library(officer)

# Word 파일로 저장
doc <- read_docx() %>%
  body_add_flextable(table_output) %>%
  body_add_par("통계표 예시", style = "heading 1")

print(doc, target = here("outputs", "statistics_table_by_flextable.docx"))
```

이렇게 생성된 statistics_table.docx 파일은 MS Word에서 열어볼 수 있으며, 이후 한글 파일로 변환하여 저장할 수 있다.

5.1.3 3. 한글(HWP) 파일로 변환

한컴 오피스에서 statistics_table.docx 파일을 열고 한글(HWP) 문서 형식인 .hwp 로 선택하여 저장한다.

5.2 요약 테이블을 만드는 R 패키지 비교

flextable을 사용하여 통계표를 생성하고 Word(.docx)로 저장하는 작업을 gtsummary와 arsenal 패키지를 사용하여 동일하게 수행하는 방법을 소개한다.

이 두 패키지는 의학 및 연구 분야에서 통계표를 손쉽게 생성하는 데 매우 유용하며, 특히 그룹별 기술 통계 및 비교 분석 결과를 깔끔한 형식으로 제공하는 장점이 있다.

하지만 flextable 보다 테이블을 다양하게 출력할 수 있는 선택사항과 유연성이 부족하다. 3개의 패키지의 특징을 다음 표와 같이 요약할 수 있다.

표 5.2: 테이블 작성 패키지의 비교

패키지	주요 기능	특징
flextable	데이터프레임을 표로 변환	다양한 스타일 지정 가능, 보고서용 고품질 테이블 제작
gtsummary	그룹별 기술 통계	평균, 표준편차, 중앙값 등 자동 계산, 깔끔한 보고서 스타일
arsenal	그룹 간 비교 및 통계 검정	t-검정, 카이제곱 검정 포함, 연구 분석에 적합

변수	**4** N = 11[1]	**6** N = 7[1]	**8** N = 14[1]
mpg	26.7 (4.5)	19.7 (1.5)	15.1 (2.6)
hp	83 (21)	122 (24)	209 (51)

[1]Mean (SD)

5.2.1 gtsummary 패키지를 사용한 통계표 생성

gtsummary는 그룹별 기술 통계를 자동으로 요약해 주는 패키지로, 특히 회귀 분석 결과와 그룹별 비교 통계를 자동으로 생성하는 기능이 강력하다.

다음은 tbl_summary 람수를 이용하요 mtcars 데이터에 대한 요약 테이블을 생성하는 과정이다.

```
library(gtsummary)

# 그룹별 요약 통계 생성
summary_table_gtsummary <- mtcars %>%
  select(cyl, mpg, hp) %>%  # 필요한 변수 선택
  tbl_summary(by = cyl,  # 그룹 변수 (cyl: 실린더 수)
              statistic = list(all_continuous() ~ "{mean} ({sd})"),  # 평균과 표준편차 표시
              missing = "no") %>%  # 결측값 처리
  modify_header(label = "변수") %>%  # 헤더 수정
  bold_labels()  # 변수명을 굵게 표시

# 출력
summary_table_gtsummary
```

위 코드는 mtcars 데이터셋에서 cyl(실린더 수)별로 mpg(연비)와 hp(마력)의 평균과 표준편차를 요약하여 테이블을 생성한다.

다음으로 gtsummary를 활용하여 Word(docx) 파일로 저장할 수 있다.

```
library(officer)

# Word 문서 생성 및 저장
summary_table_gtsummary %>%
  as_flex_table() %>%
  flextable::save_as_docx(path = here::here("outputs",  "statistics_table_by_gtsummary.docx"))
```

이제 gtsummary_statistics_table.docx 파일이 생성되며, MS Word에서 열어볼 수 있다.

5.2.2 arsenal 패키지를 활용한 그룹 비교 분석 테이블 생성

arsenal 패키지는 그룹 간 기술 통계를 자동으로 생성하고, 두 그룹을 비교하는 통계 검정을 함께 제공하는 기능을 갖추고 있다.

다음은 tableby()를 활용하여 그룹을 비교 분석하는 프로그램이다.

```r
library(arsenal)

# 그룹 비교 분석 (실린더 수에 따른 mpg, hp 차이 검정)
summary_table_arsenal <- tableby(
  cyl ~ mpg + hp,  # 그룹 변수: cyl, 비교할 변수: mpg, hp
  data = mtcars
)

# 요약 결과 출력
summary(summary_table_arsenal)
```

	4 (N=11)	6 (N=7)	8 (N=14)	Total (N=32)	p value
mpg					< 0.001
Mean (SD)	26.664 (4.510)	19.743 (1.454)	15.100 (2.560)	20.091 (6.027)	
Range	21.400 - 33.900	17.800 - 21.400	10.400 - 19.200	10.400 - 33.900	
hp					< 0.001
Mean (SD)	82.636 (20.935)	122.286 (24.260)	209.214 (50.977)	146.688 (68.563)	
Range	52.000 - 113.000	105.000 - 175.000	150.000 - 335.000	52.000 - 335.000	

위 코드는 `cyl`(실린더 수)에 따라 `mpg`(연비)와 `hp`(마력)의 기술 통계 및 그룹 간 차이에 대한 통계적 검정 결과를 제공한다.

다음은 `arsenal`의 분석 결과를 Word(`docx`) 파일로 저장하는 과정이다.

```r
# Word 문서로 저장
write2word(summary_table_arsenal, file =here::here("outputs", "statistics_table_by_arsenal.docx"))
```

```
  |
  |                                                                      |   0%
  |
  |......................................................................| 100%
```

```
/Applications/RStudio.app/Contents/Resources/app/quarto/bin/tools/aarch64/pandoc +RTS -K512m -RTS stati
```

References

Allaire, J. J., Charles Teague, Carlos Scheidegger, Yihui Xie, Christophe Dervieux, and Gordon Woodhull. 2024. “Quarto.” https://doi.org/10.5281/zenodo.5960048.