논문작성을 위한 R통계분석: Statistical analysis of papers WithR

Park Joonghee (Yonsei University coginitive Eng. lab)

2023-12-16

# 연구방법과 통계분석

**저자의 말**

이 책에서 다루고자 하는 것은 실제로 논문을 집필하거나 혹은 통계분석을 하고자 하는 경우에 실질적으로 필요한 부분과 집중해야할 부분을 담았다. 연구자로서 직접 분석을 하고 자신의 연구를 정밀하게 담고 싶은 마음은 정말 간절할 것이다. R은 무료로 이 모든 것을 가능하게 해주며, 배우는 것도 그리 어려운 것은 아니다. 그러나, R을 이용하여 통계분석을 하고자 하는 경우, 너무 기초에 매달리다보니 실제로 필요한 것을 익히는데 너무 올래걸리는 것이 현실이다. 연구자들에게 통계분석만을 목적으로 사용한다면 너무 어려운 것은 아님에도 불구하고 코드(syntax)방식이라는 이유로 벽에 부딪히는 경우가 많다. 그러나, 연구를 재현하는 관점에서 접근해보면 이러한 코드형태로 사용하는 방식은 매우 편리한 방법임에는 그 누구도 부인할 수 없다.  
본 책에서는 논문 통계분석의 절차에 따라서 사용하는 R함수와 라이브러리를 소개하여 분석하는 다양한 방법을 소개하면서 R을 익힐 수 있게 하고 있다. 함수를 일일이 기억하는 것은 매우 어려운 일이지만 상황에 맞추어서 최적의 함수를 사용하는 것은 경험으로서 기억이 잘 나고 찾아볼 수 있으므로 그러한 방식으로 분석방법을 소개하려고 한다. 분석후에는 간단하게라도 가능한 APA양식에 맞추어서 보고하는 방식을 소개하고 있다.

이 책에서는 필자가 개발한 R패키지도 사용하고 있다. jjstat라는 패키지를 이용하여 여러가지 분석의 방법과 요약기술등을 편리하게 하는 방법들을 제시하고 있다. 대체로 분석하는 것에 대한 함수는 잘 발달해있지만, 이를 정리하여 논문에 넣기위한 함수나 정리방법은 좀 미흡한 편이다. 따라서, 필자는 분석을 편리하게 하는 방법과 그리고, 결과물을 만들기 위해서 도입해야 하는 함수들을 모아서 jjstat라는 패키지로 담았다.

# 1. 데이터에 대한 이해

데이터를 이해한다는 것을 자신이 분석할 실태를 잘 이해하고 있다는 의미가 된다. 데이터를 이해한다는 것은 자신의 데이터의 열에 대한 것, 즉 변수가 무엇인지 정확히 이해하는 것이다. 그 변수의 형식과 내용에 대하여 명확히 이해하는 것이 중요하다. 보통은 행렬(matrix), 리스트(list), 배열(array), 데이터프레임(data.frame), 티블(tibble)데이터를 자주 사용한다. 이러한 형식과 변수의 형식, 그리고 변수를 리코딩하여 원하는 분석에 적용하는 방식을 잘 이해해야 한다. 이러한 과정을 배우는 것은 처음에 R을 배울 때 어렵게 만드는 경향이 있다. 따라서,

## 1. 논문을 위한 기본 개념

### 1.1. 연구주제의 선정

연구 주제 선정의 중요성  
논문 작성의 첫 걸음이자 성공적인 연구의 핵심 요소는 바로 연구 주제의 선정이다. 잘 선택된 주제는 연구의 방향성을 제시하고, 명확한 목표 설정을 가능하게 하며, 학문적 기여도를 높여주게 된다. 반대로, 부적절한 주제는 연구의 어려움, 시간 낭비, 학문적 가치 감소로 이어질 수 있다.

**연구 주제 선정을 위한 5가지 전략**

1. 관심 분야 및 전문성 고려:  
   자신의 관심 분야와 전문 지식을 기반으로 주제를 선정하면 연구에 대한 열정과 몰입도를 높일 수 있다. 이미 갖춘 지식을 바탕으로 연구를 시작하면 학습 곡선을 단축하고 연구 효율성을 높일 수 있다.
2. 최신 동향 및 학문적 격차 파악:  
   최신 학문적 트렌드와 연구 동향을 파악하여 미래 지향적인 주제를 발굴하고, 학문적 격차를 채울 수 있는 연구를 진행하는것이 좋다. 학술 논문, 학회 발표, 전문가 의견 등을 참고하여 최신 연구 동향을 파악하는 것도 매우 중요하다.
3. 연구 가능성 및 실현 가능성 평가:  
   연구 주제의 연구 가능성을 평가하여 성공적인 연구 수행 가능성을 높일 수 있는 방법을 찾아본다. 연구 자료, 방법론, 윤리적 문제 등을 고려하여 연구 가능성을 평가한다. 이때 중요한 것은, 실현 가능성을 위해 연구 시간, 예산, 인력, 장비 등 현실적인 제약 조건을 고려하는 것이 중요하다.
4. 학문적 기여도 및 사회적 영향력 검토:  
   연구 주제의 학문적 기여도와 사회적 영향력을 검토하여 연구의 가치를 높여야 한다. 연구 결과가 학문 분야에 어떤 기여를 할 수 있는지, 사회에 어떤 영향을 미칠 수 있는지 고려한다.
5. 명확한 연구 질문 및 가설 설정:  
   연구 주제를 바탕으로 명확한 연구 질문과 가설을 설정하여 연구 방향성을 명확하게 제시한다. 연구 질문은 연구의 핵심을 요약하고, 가설은 연구 결과를 예측하는 명제이므로 철저한 선행연구와 연구자의 깊은 고민을 통해서 가설에 대한 예측을 하는 것이 중요하다.

# 2. 기초통계분석

## 2.1. 연구에 관한 이해

이제 연구에 필요한 지식을 간단하게 이야기를 하고자 한다. 연구가설, 귀무가설, 대립가설, 연구가설의 유형, 1종오류와 2종오류, 연구모형 설정, 유의수준과 신뢰수준, 유의확률 및 통계적 의사결정에 대한 이해를 하고자 한다.

### 연구가설과 귀무가설

연구 가설은 연구 결과를 예측하는 명제이다. 연구가설은 연구 질문을 바탕으로 설정하며, 연구의 방향성을 제시하고 검증 가능성을 제공하므로, 명확하고 구체적인 가설 설정은 연구의 성공에 중요한 역할을 하게된다.

효과적인 연구 가설 설정 방법을 알아보자.

먼저, 명확하고 구체적인 표현을 해야 한다. 연구 질문에 대한 직접적인 답변을 제공하고, 체계적인 선행연구를 통하여 측정 가능한 변수를 포함해야 한다. 둘째, 검증이 가능해야 한다. 경험적 또는 논리적 방법으로 검증 가능한 연구여야 한다. 셋째, 관련성이 있어야 한다. 연구가설은 선행 연구와 이론적 근거를 바탕으로 설정해야 한다. 특히 조심해야 할 것은 연구자의 개인 경험에 의존해서는 안된다는 것이다. 대체로 선행연구가 부족한 경우 자신의 경험으로 모든 가설을 설정하는 경우가 있는데 이는 피해야 한다. 넷째, 단순해야 한다. 즉 이해하기 쉽고 불필요한 복잡성을 피해야 한다는 것이다. 다섯째, 파괴력이 있어야 한다. 기존 이론이나 가설에 대한 새로운 도전을 제시하는 것이 바람직하다.

연구가설의 예를 들어보자.  
연구 질문: 온라인 학습 참여도가 학습 성취도에 미치는 영향은 무엇인가?  
연구 가설: 온라인 학습 참여도가 높을수록 학습 성취도가 높아질 것이다.

**대립가설 (Alternative Hypothesis)** :

연구자가 검증하고자 하는 가설이다. 즉 연구 결과에 대한 예상이나 가정을 나타낸다. 보통 “H1”로 표기하며, 예를 들어 “A 치료법은 B 치료법보다 더 효과적이다”와 같은 가설이 연구가설을 말한다.

**귀무가설 (Null Hypothesis)**:

연구자가 연구가설을 검증하기 위해 반박하려는 가설이다. 통계적 검정에서 기본적으로 설정되는 가설이다. 보통 “H0”로 표기하며, 예를 들어 “A 치료법과 B 치료법은 효과적인 차이가 없다”와 같은 가설이 귀무가설이다. 연구가설과 귀무가설은 통계적 검정에서 중요한 역할을 하며, 연구 결과를 평가하고 결론을 도출하는데 사용된다.

**연구 가설의 유형**:  
연구가설의 유형으로는 다음과 같은 것들이 있다.  
- 서술적 가설: 두 변수 간의 관계를 단순히 서술하는 가설  
- 관계적 가설: 두 변수 간의 방향 또는 강도를 명시하는 가설  
- 원인적 가설: 한 변수가 다른 변수에 영향을 미치는 관계를 명시하는 가설

### 제 1종오류와 2종 오류

1종 오류와 2종 오류는 통계적 검정에서 중요한 개념이다.

**1종 오류 (Type I Error)**:  
- 의미: 귀무가설이 실제로 참인데도, 이에 불구하고 귀무가설을 기각하는 오류이다. 예를 들어보면, 의사가 환자에게 병이 없다고 진단했는데, 실제로는 병이 없는 경우이거나, 신약의 효과가 없는데 있다고 잘못 판단한 경우 등이다.

**2종 오류 (Type II Error)**:  
- 의미: 귀무가설이 실제로 거짓인데도, 이에 불구하고 귀무가설을 기각하지 못하는 오류이다. 예를 들면, 의사가 환자에게 병이 있다고 진단했는데, 실제로는 병이 있는 경우이다. 혹은 신약의 효과가 있는데 없다고 잘못 판단한 경우이다.  
이 두 오류는 통계적 검정에서 발생할 수 있으며, 연구 결과를 평가하고 결론을 도출하는데 중요한 역할을 한다. 1종 오류는 실제 음성인 것을 양성으로 판정하는 경우이고, 2종 오류는 실제 양성인 것을 음성으로 판정하는 경우이다.

### 연구모형의 설정

연구 모형은 연구 변수 간의 관계를 시각적으로 표현한 도식으로, 연구 가설을 검증하기 위한 구조를 제시하며, 연구 변수 간의 상호 작용을 이해하는 데 도움을 준다.

연구 모형 설정 방법은 다음과 같다. (1) 연구 가설을 기반으로 연구 가설을 검증하기 위한 변수와 그 관계를 명확하게 설정해야 한다. (2) 변수 선정시에는 연구 질문과 가설에 부합하는 주요 변수를 선정하고, 변수 간의 논리적 관계를 설정한다. (3) 변수 간의 관계를 화살표로 표현하여 나타낸다. 화살표 방향은 영향 관계를 나타낸다. (4) 모형 유형의 설정을 한다. 연구 목적에 따라 다양한 모형 유형을 선택할 수 있다. 예를 들어, 경로 분석, 구조 방정식 모델링, 회귀 분석 등이 있다.

### 연구 가설 및 연구 모형 설정 시 고려해야 할 사항

1. 연구의 목적과 범위:  
   연구 목적과 범위를 명확하게 설정해야 연구 가설과 연구 모형을 효과적으로 설정할 수 있다.
2. 선행 연구:  
   선행 연구 결과를 참고하여 연구 가설과 연구 모형을 설정하는 것이 바람직하다.
3. 통계적 검증 가능성:  
   연구 가설과 연구 모형은 통계적 방법으로 검증 가능해야 한다.
4. 윤리적 문제:  
   연구 가설과 연구 모형 설정 과정에서 윤리적 문제를 고려해야 합니다.

### 유의수준과 신뢰수준, 유의확률

1. 유의수준 (α):  
   귀무가설이 사실인데도 불구하고 기각할 확률을 의미한다. 일반적으로 0.05 또는 0.01로 설정한다.
2. 신뢰수준 (1-α):  
   귀무가설이 사실인 경우 기각하지 않을 확률을 의미한다. 즉, 연구 결과가 신뢰할 수 있는 정도를 나타낸다.
3. 유의확률 (p-value):  
   표본 데이터에서 관찰된 결과가 우연히 발생할 확률을 의미한다. 유의확률이 유의수준보다 작으면 귀무가설을 기각하고 대립가설을 채택한다.

### 통계적 의사결정

통계적 의사결정 과정은 데이터를 기반으로 합리적인 의사결정을 내리는 과정이라고 할 수 있다. 불확실성이 존재하는 상황에서 데이터 분석을 통해 가능성을 추정하고, 이를 바탕으로 최적의 선택을 하는 방법을 통계적 의사결정이라고 한다.

**통계적 의사 결정 단계**

통계적 의사결정 과정은 다음과 같은 단계로 구성됩니다.

가. 연구 목적 설정  
먼저, 연구 목적을 명확하게 설정해야 한다. 이를 통해 필요한 데이터를 수집하고 적절한 분석 방법을 선택할 수 있다.

나. 가설 설정  
연구 목적에 따라 귀무가설과 대립가설을 설정한다. 귀무가설은 일반적으로 유지하고자 하는 가설이며, 대립가설은 귀무가설을 기각할 때 채택되는 가설이다.

다. 데이터 수집 연구 목적에 맞는 데이터를 수집한다. 데이터 수집 방법에는 설문조사, 실험, 관찰 등이 있다. 물론 연구방법에 따라 양적연구와 질적연구의 수집방법에는 차이가 있다.

라. 데이터 분석  
수집된 데이터를 분석한다. 데이터 분석 방법에는 다양한 통계 검정 방법을 사용하며 특히 연구목적과 연구문제를 잘 설명해주고, 근거를 마련해줄 수 있는 통계적 검정 방법을 시행하여 분석한다. 특히 연구자는 자신이 분석할 연구에 대하여 명확히 이해하고 이 연구문제를 잘 설명해줄 적절한 통계기법을 사용해야 한다.

바. 결론 도출  
데이터 분석 결과를 바탕으로 귀무가설을 기각하거나 유지한다. 귀무가설을 기각하면 대립가설을 채택한다.

사. 의사결정  
데이터 분석 결과와 결론을 바탕으로 최종적인 의사결정을 내린다.

## 2.2. 데이터의 처리와 이해

이제 데이터를 처리하는 방법 및 변형하여 분석에 적합하게 구성하는 방법을 알아보고자 한다.

### 2.2.1. 데이터 사례 추출 및 변수 추출

데이터의 결합에는 열결합과 행결합, 공통열에 의한 결합등이 있다. 데이터 프레임을 다루다보면 행의 이름을 변수로 만들어야 하는 경우가 더러 생긴다. 이런 경우 빠르게 만드는 방법을 먼저 알아보자.

#### 2.2.1.1. 데이터에 행번호 붙이기

먼저 R에서 데이터를 다룰때 행이름을 열의 변수로만들어야 하는 경우가 자주 생긴다. 최근 R에서는 data.frame및 tibble데이터가 자주 사용되면서, 행이름(rownames)를 열 변수로 포함해서 분석해야 하는 경우가 자주 생기기 때문이다. mtcars데이터를 이용하여 이 과정을 살펴보자.

mtcars|>head()

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

데이터에서 행이름을 추출하는 방법은 rownames()를 이용한다.

rownames(mtcars)

## [1] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710"   
## [4] "Hornet 4 Drive" "Hornet Sportabout" "Valiant"   
## [7] "Duster 360" "Merc 240D" "Merc 230"   
## [10] "Merc 280" "Merc 280C" "Merc 450SE"   
## [13] "Merc 450SL" "Merc 450SLC" "Cadillac Fleetwood"   
## [16] "Lincoln Continental" "Chrysler Imperial" "Fiat 128"   
## [19] "Honda Civic" "Toyota Corolla" "Toyota Corona"   
## [22] "Dodge Challenger" "AMC Javelin" "Camaro Z28"   
## [25] "Pontiac Firebird" "Fiat X1-9" "Porsche 914-2"   
## [28] "Lotus Europa" "Ford Pantera L" "Ferrari Dino"   
## [31] "Maserati Bora" "Volvo 142E"

데이터의 열이름을 추출하는 방법은 colnames()를 이용한다.

colnames(mtcars)

## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"  
## [11] "carb"

행이름을 열에 포함하는 방법은 tibble::rownames\_to\_column()을 이용하는 것이다. 이함수에는 인수로는 data와 변수명을 정하는 var가 있다. 이 기능은 분석한 데이터를 정리해서 출력되는 표를 만들 때 상당히 유용한 기능이다.

mtcars |>   
 tibble::rownames\_to\_column(var = "car") |>  
 head()

## car mpg cyl disp hp drat wt qsec vs am gear carb  
## 1 Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## 2 Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## 3 Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## 4 Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## 5 Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## 6 Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

이러한 변환에 사용하는 함수로는 다음과 같은 것들이 있다.

has\_rownames(.data)   
remove\_rownames(.data)   
rownames\_to\_column(.data, var = "rowname")   
rowid\_to\_column(.data, var = "rowid")   
column\_to\_rownames(.data, var = "rowname")

#### 2.2.1.2. 데이터 선택과 추출

이 함수는 데이터의 일부를 추출하는 기능이다. dplyr패키지의 filter()와 select() 함수의 기능을 동시에 활용이 가능한 함수이다. 다만 사용성에서 dplyr보다는 직관적성은 떨어지지만, 매우 유용한 기능이다.

일반적으로 R에서의 데이터프레임은 인덱싱을 하는 방법이 여러가지가 있다. 먼저 $ 기호를 이용하여 변수를 추출한다. 변수를 추출할 때, 벡터로 추출하는 방법과 데이터프레임으로 추출하는 방법이 있다.  
##### (1) R-base방법

mtcars$mpg

## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
## [31] 15.0 21.4

두번째로는 [[]]기호를 이용하여 번호를 이용해서 추출한다.

mtcars[[1]]

## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
## [31] 15.0 21.4

데이터에 변수명을 넣어서 벡터로 추출할 수 있다. ’데이터[, 변수명]’으로 벡터를 추출할 수 있다.

mtcars[,"mpg"]

## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
## [31] 15.0 21.4

벡터로 추출된 형태에서 데이터프레임으로 만들어서 추출하려면 drop = FALSE 옵션을 사용한다.

mtcars[,"mpg", drop=FALSE] |> head()

## mpg  
## Mazda RX4 21.0  
## Mazda RX4 Wag 21.0  
## Datsun 710 22.8  
## Hornet 4 Drive 21.4  
## Hornet Sportabout 18.7  
## Valiant 18.1

기호를 사용할 때 []를 한번만 사용하면 데이터프레임으로 추출이 된다.

mtcars[1] |> head()

## mpg  
## Mazda RX4 21.0  
## Mazda RX4 Wag 21.0  
## Datsun 710 22.8  
## Hornet 4 Drive 21.4  
## Hornet Sportabout 18.7  
## Valiant 18.1

데이터[변수명]만 넣으면 데이터 프레임으로 추출이 된다.

mtcars["mpg"] |> head()

## mpg  
## Mazda RX4 21.0  
## Mazda RX4 Wag 21.0  
## Datsun 710 22.8  
## Hornet 4 Drive 21.4  
## Hornet Sportabout 18.7  
## Valiant 18.1

**조건에 맞는 데이터 추출**  
조건을 사용할 때에는 데이터[데이터$변수에 대한 조건 ,]으로 해서 열에 대한 성분을 추출한다. 예를 들어 mpg값이 30보다 큰 값을 추출해보자.

mtcars[mtcars$mpg > 30 ,1:8]

## mpg cyl disp hp drat wt qsec vs  
## Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1  
## Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1  
## Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1  
## Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1

범주형 변수에 대하여 이용하여 추출하면 다음과 같이 할 수 있다.

mtcars[mtcars$cyl == 6, 1:8]

## mpg cyl disp hp drat wt qsec vs  
## Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0  
## Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0  
## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1  
## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1  
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1  
## Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1  
## Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0

##### (2) subset을 이용한 추출

subset을 이용하면 인수는 데이터와 행을 선택하는 ‘subset= 행성분’ 인수와 열을 선택하는 select변수가 있다. 보통은 데이터에 대하여 subset을 이용하여 조건을 사용하여 추출이 가능하다. 함수명과 인수가 같은 용어라서 헷갈릴 수 있지만, 함수와 인수를 명확히 구분하면 사용이 불편하지 않을 것이다.

먼저, mtcars데이터에서 mpg값이 30이상이 되어야한다고 생각할 때, subset = mpg > 30으로 설정하여 추출한다.

mtcars[,1:5] |>   
 subset( subset = mpg > 30)

## mpg cyl disp hp drat  
## Fiat 128 32.4 4 78.7 66 4.08  
## Honda Civic 30.4 4 75.7 52 4.93  
## Toyota Corolla 33.9 4 71.1 65 4.22  
## Lotus Europa 30.4 4 95.1 113 3.77

동일한 방법으로 다음과 같이 인수를 select로 이용하여 열을 선택하는 방법도 있다. select를 사용할 때는 여러개의 변수를 선택해야 하므로 c()를 사용한다.

mtcars |>   
 subset( subset=(mpg) > 30, select=c(1:5))

## mpg cyl disp hp drat  
## Fiat 128 32.4 4 78.7 66 4.08  
## Honda Civic 30.4 4 75.7 52 4.93  
## Toyota Corolla 33.9 4 71.1 65 4.22  
## Lotus Europa 30.4 4 95.1 113 3.77

열변수를 선택할 때, 번호를 이용하여 선택해도 된다. 필요한 변수의 열의 위치를 알면 그 번호를 선택하여 변수를 선택할 수 있다.

mtcars |>   
 subset( select = c(1, 2)) |>   
 head()

## mpg cyl  
## Mazda RX4 21.0 6  
## Mazda RX4 Wag 21.0 6  
## Datsun 710 22.8 4  
## Hornet 4 Drive 21.4 6  
## Hornet Sportabout 18.7 8  
## Valiant 18.1 6

변수명을 직접 안다면 변수를 직접 설정해서 선택해도 된다. 이 역시도 c()을 이용하여 선택하면 된다.

mtcars |>   
 # subset( select = c("mpg", "cyl")) |>   
 subset( select = c("mpg", "cyl")) |>   
 head()

## mpg cyl  
## Mazda RX4 21.0 6  
## Mazda RX4 Wag 21.0 6  
## Datsun 710 22.8 4  
## Hornet 4 Drive 21.4 6  
## Hornet Sportabout 18.7 8  
## Valiant 18.1 6

##### (3) tidyverse의 select, filter이용한 추출

이번에는 tidyverse를 이용하여 mtcars데이터에서 mpg값이 30이상의 값을 추출해보자. 조건은 mpg > 30으로 설정하여 추출한다.

mtcars[,1:5] |>   
 filter( mpg > 30)

## mpg cyl disp hp drat  
## Fiat 128 32.4 4 78.7 66 4.08  
## Honda Civic 30.4 4 75.7 52 4.93  
## Toyota Corolla 33.9 4 71.1 65 4.22  
## Lotus Europa 30.4 4 95.1 113 3.77

mpg > 30 이상이면서 열을 1~5열까지를 추출해보자. v

mtcars |>   
 filter(mpg > 30) |>   
 select(1:5)

## mpg cyl disp hp drat  
## Fiat 128 32.4 4 78.7 66 4.08  
## Honda Civic 30.4 4 75.7 52 4.93  
## Toyota Corolla 33.9 4 71.1 65 4.22  
## Lotus Europa 30.4 4 95.1 113 3.77

select함수도 역시 변수명을 안다면 변수 이름을 직접 지정해서 추출할 수 있다. 다만, “cyl”와 같이 입력하면 오류가 나고 그냥 cyl을 입력해야 변수로 인식한다.

mtcars |>   
 select(mpg, cyl) |>  
 head()

## mpg cyl  
## Mazda RX4 21.0 6  
## Mazda RX4 Wag 21.0 6  
## Datsun 710 22.8 4  
## Hornet 4 Drive 21.4 6  
## Hornet Sportabout 18.7 8  
## Valiant 18.1 6

#### 2.2.1.3. 변수 생성 및 변수 변환

transform()을 이용하여 변수를 생성하거나 변환하는 방법은 dplyr::mutate()와 같은 역할을 한다. 인수로는 데이터을 넣고, 나머지 인수는 변환이나 생성에 관한 식을 넣으면 된다. 이 파트에서는 가능한 R-base에 존재하는 함수를 최대한 사용하여 분석하는 방법을 제시했다.

**변수리코딩**  
먼저 변수를 factor로 변경하는 변환을 해보자. 먼저 기본적인 base R에서 하는 방법과 transform을 이용한 방법을 사용해보자.

1. R-base에서 변수 리코딩

일단적으로 factor()처리를 하기 위해서 사용하는 방법을 아래에 설명했다.

#R- base  
data(mtcars)  
Mtcars <- mtcars  
# Engine (0 = V-shaped, 1 = straight)  
Mtcars$vs = factor(Mtcars$vs,   
 levels = c(0,1),   
 labels = c("V\_shaped"," straight") )  
  
Mtcars$vs

## [1] V\_shaped V\_shaped straight straight V\_shaped straight V\_shaped   
## [8] straight straight straight straight V\_shaped V\_shaped V\_shaped   
## [15] V\_shaped V\_shaped V\_shaped straight straight straight straight  
## [22] V\_shaped V\_shaped V\_shaped V\_shaped straight V\_shaped straight  
## [29] V\_shaped V\_shaped V\_shaped straight  
## Levels: V\_shaped straight

다음은 transform()을 이용하여 변수를 변형한 것이다.

1. transform()을 이용한 리코딩

#transform   
data(mtcars)  
Mtcars1 <- mtcars  
# Engine (0 = V-shaped, 1 = straight)  
Mtcars1 |>   
 transform(vs = factor(vs,  
 levels=c(0,1),  
 labels=c("V\_shaped"," straight")))|>  
 head(10)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 V\_shaped 1 4 4  
## Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 V\_shaped 1 4 4  
## Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 straight 1 4 1  
## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 straight 0 3 1  
## Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 V\_shaped 0 3 2  
## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 straight 0 3 1  
## Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 V\_shaped 0 3 4  
## Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 straight 0 4 2  
## Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 straight 0 4 2  
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 straight 0 4 4

1. dplyr::mutate()에서 변수 리코딩

dplyr::mutate()에서 변수를 factor로 만드는 과정을 다시 한번 해보도록 하자. dplyr::mutate는 library(tidyvers) 혹은 library(dplyr)을 수행한 후에는 그냥 mutate만 써도 된다. 독자들에게 어떤 패키지이지 정화히 알려주기 위해서 패키지명을 앞에 쓴 것이다.

data(mtcars)  
Mtcars2 <- mtcars  
  
Mtcars2 |>   
 dplyr::mutate(vs = factor(vs, levels=c(0,1), labels=c("V\_shaped"," straight"))) |>  
 head(10)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 V\_shaped 1 4 4  
## Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 V\_shaped 1 4 4  
## Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 straight 1 4 1  
## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 straight 0 3 1  
## Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 V\_shaped 0 3 2  
## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 straight 0 3 1  
## Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 V\_shaped 0 3 4  
## Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 straight 0 4 2  
## Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 straight 0 4 2  
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 straight 0 4 4

(4)새로운 변수의 생성

예륻 들어 새변수 두 변수의 평균 혹은 합을 이용하여 새로운 변수를 만든다고 하자. 그런 경우에는 R-base와 trandfomrm(), aplyr::mutate()를 사용하면 편리하게 만들 수 있다. 몇개이 함수의 합이나 평균을 구하거나 혹은 요인에 맞는 항목을 묶어 항목묶음(Item parceling)을 하는 경우에 새로운 변수를 만들어 내는 경우가 많다.

mtcars |>   
 transform(sum\_hp\_wt = hp + wt) |>  
 subset(select=c(mpg, hp, wt, sum\_hp\_wt)) |>  
 head(10)

## mpg hp wt sum\_hp\_wt  
## Mazda RX4 21.0 110 2.620 112.620  
## Mazda RX4 Wag 21.0 110 2.875 112.875  
## Datsun 710 22.8 93 2.320 95.320  
## Hornet 4 Drive 21.4 110 3.215 113.215  
## Hornet Sportabout 18.7 175 3.440 178.440  
## Valiant 18.1 105 3.460 108.460  
## Duster 360 14.3 245 3.570 248.570  
## Merc 240D 24.4 62 3.190 65.190  
## Merc 230 22.8 95 3.150 98.150  
## Merc 280 19.2 123 3.440 126.440

두 변수의 평균을 구해서 하나의 변수 생성하는 경우는 다음과 같이 직접 식을 입력한다.

mtcars |>   
 transform(mean\_hp\_wt = (hp + wt)/2 ) |>  
 subset(select=c(mpg, hp, wt, mean\_hp\_wt)) |>  
 head(10)

## mpg hp wt mean\_hp\_wt  
## Mazda RX4 21.0 110 2.620 56.3100  
## Mazda RX4 Wag 21.0 110 2.875 56.4375  
## Datsun 710 22.8 93 2.320 47.6600  
## Hornet 4 Drive 21.4 110 3.215 56.6075  
## Hornet Sportabout 18.7 175 3.440 89.2200  
## Valiant 18.1 105 3.460 54.2300  
## Duster 360 14.3 245 3.570 124.2850  
## Merc 240D 24.4 62 3.190 32.5950  
## Merc 230 22.8 95 3.150 49.0750  
## Merc 280 19.2 123 3.440 63.2200

**airquality 데이터를 이용하여 변수 생성 연습** airquality(New York Air Quality Measurements) 데이터는 1973년 5월부터 9월까지 뉴욕의 일일 대기질 측정한 데이터이다. 이 데이터를 이용하여 새로운 변수를 리코딩해보도록 하자. 먼저 오존(Ozone)데이터에는 부호를 바꾸고, 온도에 대한 단위를 변경해서 새롭게 데이터를 만들어보려고 한다.

airquality |> head()

## Ozone Solar.R Wind Temp Month Day  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8.0 72 5 2  
## 3 12 149 12.6 74 5 3  
## 4 18 313 11.5 62 5 4  
## 5 NA NA 14.3 56 5 5  
## 6 28 NA 14.9 66 5 6

airquality에 나타나는 Temp는 화씨() 로나타나있다.  
이것을 섭씨 로 바꾸는 것을 해보자. 바꾸는 공식은 다음과 같다.

이것을 새로운 변수를 변수를 만들 때 사용할 수 있다.  
원래 변수명을 입력하면 데이터의 내용이 교체되어 원본데이터가 없어지기 때문에 가능한 새로운 데이터를 만들어서 사용하는 것을 권장한다. C\_Temp라는 변수명을 만들어서 섭씨로 전환시킨 변수를 새롭게 하는 것은 아래 코드에서 확인할 수 있다.

# transform(airquality, Ozone = -Ozone)  
#Ozone은 new변수로 마이너스를 붙여서 생성   
# F(Fahrenheit ) -> C(Celsius) = (F-32)/1.8  
transform(airquality, new = -Ozone, C\_Temp = (Temp-32)/1.8) |>  
 head()

## Ozone Solar.R Wind Temp Month Day new C\_Temp  
## 1 41 190 7.4 67 5 1 -41 19.44444  
## 2 36 118 8.0 72 5 2 -36 22.22222  
## 3 12 149 12.6 74 5 3 -12 23.33333  
## 4 18 313 11.5 62 5 4 -18 16.66667  
## 5 NA NA 14.3 56 5 5 NA 13.33333  
## 6 28 NA 14.9 66 5 6 -28 18.88889

#### 2.2.1.4. 역코딩

**변수변환 처리 recode**  
다음은 일반적으로 변수를 리코딩, 즉 역코딩을 하는 방법이다.if문을 사용하면 너무 복자하게 된다. 따라서, 이련 경우는 dplyr::case\_when()을 이용하여 바꾸면 된다. 조건 ~ 내용형식으로 적용한다.

library(dplyr)  
# generate data  
y <- seq(1, 5, by = 1)  
## recoding variable   
data.frame(y) %>%   
 mutate(rev = case\_when(  
 y == 5 ~ 1,  
 y == 4 ~ 2,  
 y == 3 ~ 3,  
 y == 2 ~ 4,  
 y == 1 ~ 5  
 ))

## y rev  
## 1 1 5  
## 2 2 4  
## 3 3 3  
## 4 4 2  
## 5 5 1

그러나, 매번 역코딩에서는 이러한 방법으로 코딩하여 바꾸는 것은 불편한 일이다.  
따라서, 역코딩 기법을 이용하여 변수를 변환하는 것이 더 좋은 방법이 된다.  
역코딩 방법 데이터를 역코딩 하는 방법은 단순하다. 5점 척도라면, 다음과 같은 규칙으로 바꾸면 된다.

이렇게 만드는 방법은 5점 척도이면 로 하면 된다. 7점척도이면 8-변수로 하여 게산하면 자연습럽게 계산이 된다. 그렇다면 4,6,8로 구성된 변수라면 어떻게 하면될까? 생각해보면 간단한 문제이다. 12-변수를 하면 된다. 8->4로, 6->6으로, 4->8로 만들려면 가장 작은수(minimun value)와 가장 큰 수(Maximum value)를 더한 값에서 각각의 변수를 빼는 것이 역코딩의 원리이다.

그렇다면, mtcars에서 cyl변수를 역코딩해서 확인해보도록 하자. 이때 12(= 4+8)에서 빼게 되는 이유는 최솟값 4와 최대값 8의 합을 이용하여 빼야 반대로 나타나기 때문이다.

역코딩을 하기 위해서 사용하는 함수는 R-base에서는 transform()함수이다. transform(변형시키다 (=convert))라는 단어에서 볼 수 있듯이 이 함수는 새로운 변수를 만들거나 혹은 리코딩해서 변수를 수정해야 하는 경우 매우 편리한 함수이다. transform(Transform an Object, for Example a Data Frame)은 변환을 하는 함수이며, 데이터프레임을 받아서첫 번째 인수를 데이터프레임으로 변환하는 함수이다. 변환을 한 후에 subset(select = c(cyl, cyl\_reverse)을 사용한 것은 변환된 것만을 보이게 하기 위해서 변수를 선택한 것이다.

mtcars |>   
 transform(cyl\_reverse = 12- cyl) |>  
 subset(select = c(cyl, cyl\_reverse)) |> # 변수를 선택: 원래변수와 역코딩된 변수만 선택  
 head(n= 10)

## cyl cyl\_reverse  
## Mazda RX4 6 6  
## Mazda RX4 Wag 6 6  
## Datsun 710 4 8  
## Hornet 4 Drive 6 6  
## Hornet Sportabout 8 4  
## Valiant 6 6  
## Duster 360 8 4  
## Merc 240D 4 8  
## Merc 230 4 8  
## Merc 280 6 6

결과를 보면, 각각의 값이 역코딩 되어 들어가 있는 것을 확인할 수 있다. 그리고, 역코딩한 변수를 만들때에는 원래의 변수를 바꾸기 보다는 새로운 변수를 만드는 것을 추천한다. 분석을 하다가 원래것으로 다시 돌리는 일도 생기기 때문에 가능한 역코딩을 한 변수는 새로운 변수를 만드는 것이 낫다.

tidyverse 패키지를 이용하여 mutate를 이용하는 방법도 있다. 일반적으로 타이디버스(tidyverse)를 이용하는 방법이 더 만힝 알려져있고, mutate(;(새로운 형태로) 변형되다)라는 단어의 의미가 직관적이이기 때문에 함수를 기억하기 좋다는 장점이 있다. 또한 필자의 입장에서는 변수를 선택함에 있어서 select()도 subset()보다는 활용하기가 더 편리하다는 생각이 든다. 여하튼 tidyverse는 여러 가지 면에서 활용하기 좋은 패키지이다.

library(tidyverse)  
  
mtcars |>   
 mutate(cyl\_reverse = 12- cyl) |>  
 select(cyl, cyl\_reverse) |>   
 head(n= 10)

## cyl cyl\_reverse  
## Mazda RX4 6 6  
## Mazda RX4 Wag 6 6  
## Datsun 710 4 8  
## Hornet 4 Drive 6 6  
## Hornet Sportabout 8 4  
## Valiant 6 6  
## Duster 360 8 4  
## Merc 240D 4 8  
## Merc 230 4 8  
## Merc 280 6 6

**jjstat를 이용한 역코딩 방법: rc, rev\_coding, revcode**  
  
역코딩을 하는 것은 어려운 것은 아니지만, 역코딩을 직접적인 코딩을 하다보면 실수가 자주 생기곤 한다. 그래서 이러한 부분에서 함수를 이용하여 실수를 줄이는 것도 좋은 방법이 된다. 보통 교육학 연구나 심리학 및 경영학 연구에서는 여러 가지 면에서 주의(attention)을 주기 위해서 측정도구내에 역코딩을 하는 경우가 많다보니 변수의 역코딩을 자주 활용하게 된다.

이때 사용하면 좋은 것이 jjstat패키지의 rc() 혹은 rev\_coding()함수이다. 이를 이용하여 역코딩을 해보면 된다. 옵션으로 n은 리커트 척도에서 설정된 값을 최솟값+최댓값의 합을 입력해야 한다. 즉, 1~5점 척도는 n = 6이고, 1~7점 척도이면 n=8이다. 그리고, 인수중에서 check = TRUE로 하면 원래의 소스와 역코딩된 값을 함께 보여주어 제대로 잘 변환이 되었는지 점검을 할 수 있다.  
rc과 rev\_coding의 함수는 같은 함수인데, 하나는 이름을 명확하게 하기 위해 이름을 길게 둔 것이고 익숙해지면 쉽게 사용할 수 있도록 이름을 줄여놓은 함수이다.

jjstat::rc(mtcars$cyl, n=12)

## [1] 6 6 8 6 4 6 4 8 8 6 6 4 4 4 4 4 4 8 8 8 8 4 4 4 4 8 8 8 4 6 4 8

이번에는인수에서 check = TRUE로 설정한 경우를 보자. 결과값을 확인해보면, 원래의 입력된 소스와 역코딩된 결과를 확인할 수 있다. check는 제대로 코딩이 되었는지 점검용으로 확인하기 위한 인수이다.

jjstat::rc(mtcars$cyl, n = 12, check=TRUE) |> head(10)

## input direct reverse  
## 1 6 -> 6  
## 2 6 -> 6  
## 3 4 -> 8  
## 4 6 -> 6  
## 5 8 -> 4  
## 6 6 -> 6  
## 7 8 -> 4  
## 8 4 -> 8  
## 9 4 -> 8  
## 10 6 -> 6

이것을 데이터내에서 활용하려고 하면, tidyvers함수들과 함께 이용하려면 새로운 변수를 만들때 유용하다. 먼저 두가지 rc()함수나 rev\_coding()함수를 이용하면 된다.

1. rc함수이용

mtcars |>   
 mutate(cyl\_reverse = rc(cyl, 12)) |> # reverse code(간편한 이름 )  
 select(cyl, cyl\_reverse) |>   
 head()

## cyl cyl\_reverse  
## Mazda RX4 6 6  
## Mazda RX4 Wag 6 6  
## Datsun 710 4 8  
## Hornet 4 Drive 6 6  
## Hornet Sportabout 8 4  
## Valiant 6 6

(2) rev\_coding함수 이용

mtcars |>   
 mutate(cyl\_reverse = rev\_coding(cyl, 12)) |> # reverse code(명확한 이름 )  
 select(cyl, cyl\_reverse) |>   
 head()

## cyl cyl\_reverse  
## Mazda RX4 6 6  
## Mazda RX4 Wag 6 6  
## Datsun 710 4 8  
## Hornet 4 Drive 6 6  
## Hornet Sportabout 8 4  
## Valiant 6 6

그런데, 한 두개의 변수를 처리할 때에는 나름대로 괜찮지만, 데이터내에서 여러개의 변수를 역코딩히야 하는 경우는 이또한 상당히 불편해진다.  
따라서 데이터 내에서 이를 처리하는 것을 알아보자.

#### 2.2.1.5. 데이터내에서 한번에 처리하기

데이터 내에서 처리하려면 jjstat::revcode을 이용하면 된다. 인수로는 데이터를 받아서 n(= max + min)을 지정하고, 변수명을 선택하면 된다. cyl의 변수를 역코딩을 해보자. cyl은 4, 6,8로 구성이 되어있어서 4->8로 6->6으로 8->4로 바꾼다.  
데이터에 변환된 것을 쉽게 보이게 하기 위해서 select함수를 이용하여 필요한 변수를 선택하여보이게 하였다.  
먼저 데이터에서 tidyverse를 통해서 생성하는 것이 아니라 그냥 데이터 내에서 처리하는 방법이다. 아래의 코드는 데이터가 길어서 보기 힘들수 있어서 알아보기 쉽게 데이터를 구성하여 역코딩한 결과를 보여주었다.

MTCARS <- mtcars|> select(mpg, am, vs, gear, carb, cyl) #변수의 선택   
MTCARS |> jjstat::revcode(n=12, "cyl") |>  
  
 head()

## mpg am vs gear carb cyl cyl\_r  
## Mazda RX4 21.0 1 0 4 4 6 6  
## Mazda RX4 Wag 21.0 1 0 4 4 6 6  
## Datsun 710 22.8 1 1 4 1 4 8  
## Hornet 4 Drive 21.4 0 1 3 1 6 6  
## Hornet Sportabout 18.7 0 0 3 2 8 4  
## Valiant 18.1 0 1 3 1 6 6

revcode함수는 변수하나만을 할 수 있는 것이 아니라, 여러개의 변수를 바꿀 수도 있다. 이번에는 mtcars데이터의 vs, am은 0,1로 코딩된 데이터인데 이것을 역코딩해보자.

# MTCARS <- mtcars|> select(mpg, am, vs, gear, carb, cyl)   
jjstat::revcode(MTCARS, #변수의 선택   
 n = 1, #역코딩 계수   
 "am","vs") |> #역코딩할 변수   
 head()

## mpg am vs gear carb cyl am\_r vs\_r  
## Mazda RX4 21.0 1 0 4 4 6 0 1  
## Mazda RX4 Wag 21.0 1 0 4 4 6 0 1  
## Datsun 710 22.8 1 1 4 1 4 0 0  
## Hornet 4 Drive 21.4 0 1 3 1 6 1 0  
## Hornet Sportabout 18.7 0 0 3 2 8 1 1  
## Valiant 18.1 0 1 3 1 6 1 0

만약에 변수의 코딩을 해야 하는데, 척도가 다른 변수가 있다면 어떻게 해야할까? 이는 각각의 빼야할 n값을 달리해야할 것이다. mtcars데이터에서 cyl은 4,6,8이므로 역코딩을 하려면 12에서 빼야 하고, gear(3 ,4, 5)은 1~5로 되어있으니 n을 6로 해야한다. 그리고 crab는 1~8까지 척도로 되어있어서 역코딩을 하려면 n=9에서 빼야 한다. 이렇게 각각의 역코딩해야할 값이 다르다면 n값을 c()함수로 결합한 값으로 설정하면된다.

# MTCARS <- mtcars|> select(mpg, am, vs, gear, carb, cyl)   
jjstat::revcode(MTCARS,  
 n = c(12, 6, 9), # 역코딩 계수  
 "cyl","gear","carb") |> #역코딩할 변수  
 head(10)

## mpg am vs gear carb cyl cyl\_r gear\_r carb\_r  
## Mazda RX4 21.0 1 0 4 4 6 6 2 5  
## Mazda RX4 Wag 21.0 1 0 4 4 6 6 2 5  
## Datsun 710 22.8 1 1 4 1 4 8 2 8  
## Hornet 4 Drive 21.4 0 1 3 1 6 6 3 8  
## Hornet Sportabout 18.7 0 0 3 2 8 4 3 7  
## Valiant 18.1 0 1 3 1 6 6 3 8  
## Duster 360 14.3 0 0 3 4 8 4 3 5  
## Merc 240D 24.4 0 1 4 2 4 8 2 7  
## Merc 230 22.8 0 1 4 2 4 8 2 7  
## Merc 280 19.2 0 1 4 4 6 6 2 5

간단한 데이터를 다룰 때에는 불편함이 없을 수 있으나, 다루는 데이터가 많은 경우에는 revcode를 이용하여 처리하는 것이 시간을 절약해준다.  
  
이번에는 박중희(2020)의 스마트러닝 기술수용 모델 연구자료(sats)를 이용해보자. jjstat패키지에서 data(sats)로 불러올 수 있다. sats데이터에서 sux1~ sux5를 모두 역코딩해보자.이것은 1~7점 척도이다. 데이터가 변수가 많아 실제로 변경할 변수만 선택하여 결과를 비교해보자. 여기에서는 모든 변수가 1-7점 척도이므로 이때의 n값은 8만 하나만 입력해도 모든 변수에 적용이 된다. 입력되는 것은 변수명이므로 변수의 이름이 동일해야 한다. c()함수로 묶어도 되며, 입력된 순서대로 역코딩을 한다.아래의 코드에서는 sux3, sux2를 순서를 바꾸어서 입력했는데 나타난 결과도 순서가 바뀌어서 나타났다.

#변수선별  
SATS1 = sats|> select(sux1:sux5)  
  
#역코딩 진행: n=8을 모든 변수에 한번에 적용하는 방법   
revcode(SATS1, n=8, "sux1", "sux3", "sux2", "sux4", "sux5") |>  
 head(10)

## # A tibble: 10 × 10  
## sux1 sux2 sux3 sux4 sux5 sux1\_r sux3\_r sux2\_r sux4\_r sux5\_r  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7 7 7 2 6 1 1 1 6 2  
## 2 6 7 7 4 4 2 1 1 4 4  
## 3 3 3 3 2 3 5 5 5 6 5  
## 4 6 6 5 5 6 2 3 2 3 2  
## 5 6 6 6 3 3 2 2 2 5 5  
## 6 5 4 4 4 5 3 4 4 4 3  
## 7 7 7 7 7 7 1 1 1 1 1  
## 8 5 5 5 5 5 3 3 3 3 3  
## 9 2 4 5 1 2 6 3 4 7 6  
## 10 5 5 5 5 5 3 3 3 3 3

만약 변수명이 아니라 열번호를 선택한다면 열번호 명에 “\_r”을 붙여서 나타내게 된다. 입력하는 방식은 c(1:5)로 하거나 1:5도 가능하고, c(1, 3, 5) 나 1, 3, 5 등으로 어떤 방식으로 입력해도 가능하다.

#변수선별  
SATS1 = sats|> select(sux1:sux5)  
#역코딩 진행   
revcode(SATS1, n=8, 1,3,5 ) |>  
 head(10)

## # A tibble: 10 × 8  
## sux1 sux2 sux3 sux4 sux5 `1\_r` `3\_r` `5\_r`  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7 7 7 2 6 1 1 2  
## 2 6 7 7 4 4 2 1 4  
## 3 3 3 3 2 3 5 5 5  
## 4 6 6 5 5 6 2 3 2  
## 5 6 6 6 3 3 2 2 5  
## 6 5 4 4 4 5 3 4 3  
## 7 7 7 7 7 7 1 1 1  
## 8 5 5 5 5 5 3 3 3  
## 9 2 4 5 1 2 6 3 6  
## 10 5 5 5 5 5 3 3 3

### 2.2.2. 데이터 형태 변형

통계분석을 위해서는 데이터를 크게 2가지로 구분되는데, 롱포맷(long foramat)데이터와 와이드포맷(wide format)데이터로 구분할 수 있다.  
데이터 분석을 하다 보면 종종 데이터의 형식을 변환해야 할 때가 있다. 연구자가 사용하는 통계 도구에서 원하는 데이터의 모습이 아닌 경우가 많기 때문이다. 이때 롱포맷과 와이드포맷은 많이 사용되는 방법들이다. 롱포맷은 데이터를 행 기반으로 정리하는 형식이라고 할 수 있다. 여러 변수를 하나의 열로 정리하여 데이터를 구조화하는데, 주로 다양한 통계 분석 기법(다변량분석, 회귀분석, ANOVA 등)에 적합하다. 롱포맷은 매우 유연해서 변수를 추가하거나 삭제하거나 하는 재구성에 매우 유리하며, 시각화를 할때 적합한 형식이다.  
반면 와이드포맷은 데이터를 열 기반으로 정리하는 형식이다. 여러 관측치를 하나의 행으로 집계하여 데이터를 요약한다. 주로 데이터 저장과 기본적인 통계 분석에 유용하다. 데이터를 간결하게 표현하기에 적합한 만큼 변수가 적을 때 유리하다고 할 수 있다. 대체로 기본 통계분석에 유리한 형식이다.  
데이터 분석을 할 때는 연구 주제와 분석 목적에 맞게 데이터 형식을 변환해야 하는 경우가 많다. 이렇게 자유롭게 형식을 변경할 수 있으면 데이터를 효율적으로 다룰 수 있고 정확한 분석 결과를 얻을 수 있기 때문에 이러한 형식의 전환은 중요하다고 할 수 있다.  
이 절에서는 가능한 다양한 방법의 롱포맷과 와이드 포맷을 바꾸는 방법을 배우도록 하자. r-base에 있는 stats::reshape()함수를 이용하는 방법과 tidyverse패키지중 tidyr::gather, spread와 pivot\_longer, pivot\_wider 이용한 방법, 그리고, reshape2::melt, dcast를 이용한 방법을 익혀보도록 하자. 참고로 reshape::reshape()함수에서는 함수명이 동일하고, 긴데이터는 direction =“long”로 설정해야 하며, 넓은데이터는 direction =“wide”로 설정해서 변경한다. reshape패키지의 긴데이터로 변경할 때 melt는 특이점이 없으나, dcast는 formula를 이용하게 되므로 주의깊게 살펴볼 필요가 있다.

실제 분석을 하는 경우에는 각 상황에 잘 맞는 방법을 사용하면 될 것이기 때문에 상황에 맞게 익혀서 사용하면 된다.

**일반데이터를 긴데이터(long format)으로 변경**

일반데이터를 긴데이터로 변경하는 개념은 다음과 같다.  
일단 데이터내에서 유지해야하는 데이터를 선정하는 것과 길에 늘어서는 데이터의 변수들의 이름을 정하는 부분, 그리고 값들이 위치하는 곳의 이름을 정해주어야 하는 것이 변형시키는 기본적인 방법이라고 할 수 있다.  
  
이제 제시된 그림 1. 긴데이터로 변경에 대한 내용을 이해해보도록 하자. 일반적인 데이터(주로 넓은데이터)인 경우 봐야할 부분이 바로 열부분의 변수와 해당하는 값들에 대한 것들이다. 긴 데이터로 만들기 위해서는 다음과 같이 생각해보자.

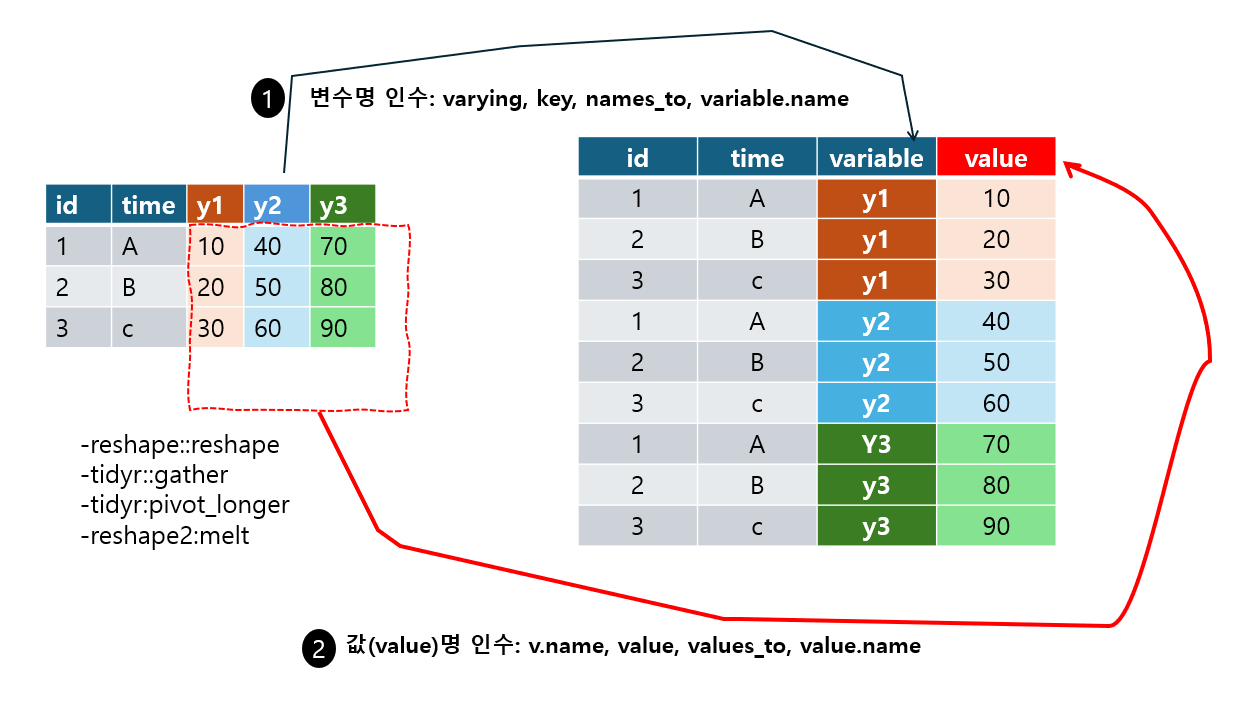


그림 1. 긴데이터(longformat)으로 변경

**긴 데이터로 변경하는 절차**

(1) 고정되어야 하는 열이 무엇인지 결정한다. 이 열은 위치가 고정된다.   
 그림 1.에서는 id, time변수가 고정되어야 하는 값이다.   
(2) 길게 변형되어야 하는 변수(열)이 무엇인지 결정한다.   
 그림1에서는 y1 ~ y3까지를 모두 이동시킬 예정이다.   
(3) 옮겨지는 부분의 열이름이 값들로 변화되므로 새롭게 만들어질 열이름을 설정한다.   
(4) 해당하는 값들이 들어가서 새롭 게 만들어지는 열의 이름을 결정한다.

긴데이터로 변경하는 경우 어떠한 함수를 사용하더라도 이러한 부분의 원칙은 유지된다. 고정되어야 하는 열과 이동되어야 하는 열을 결정하고, 이동되어야 하는 열은 열이름(colnames)가 값으로 들어가는 변수명을 정하고, 해당하는 값들이 들어가는 변수명을 정하면 된다.

**넓은데이터(wide format)으로 변경**

이제 넓은 데이터로 만드는 방법을 알아보자. 긴데이터를 만드는 방법이 이해되었다면 이제는 과정에 대한 것을 거꾸로 하는 것이므로 개념적인 것들은 이해하고 바라보면 이 방법도 간단하게 설명이 된다. 또한 이해해야 할 부분은 넓은 데이터로 변경할 때에는 넓은 데이터로 만드는 방법도 있고, 이러한 과정에서 집계함수의 기능을 수행하기도 한다. 와이드 데이터로 변경하는 함수들은 대부분 집계함수의 기능을 수행하는 경우가 많다. 그래서, 처음에 와이드 데이터로 만드는 함수가 복잡하게 느껴진다. 지금은 와이드한 데이터로 만드는 것이므로 복잡하게 생각하지 말고, 와이드 데이터로 만드는 과정을 이해하는 것이 중요하다.

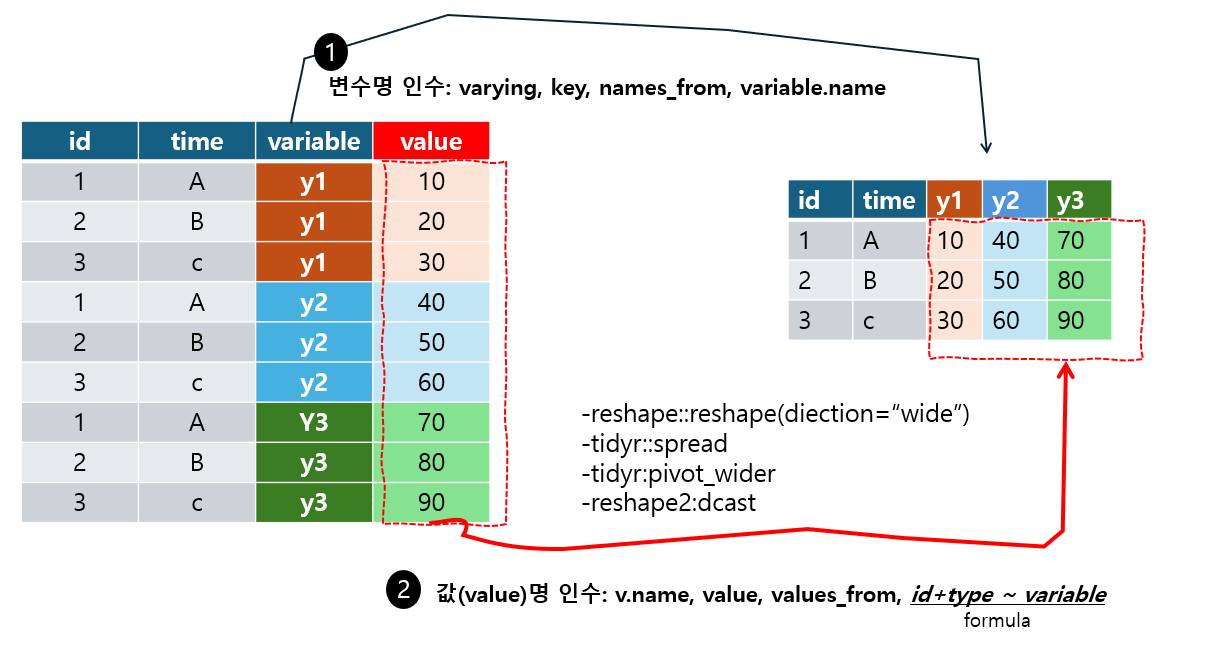


그림 2. 넓은데이터(wide format)으로 변경

데이터를 와이드하게 바꾸는 목적은 크게 2가지인데, 첫번째로 통계패키지에 맞게 사용하기 위함이다. SPSS나 JAMOVI, JASP, EXCEL 등은 모두 기본적으로 와이드한 데이터를 사용한다. 이렇게 변경하면 변수를 명확하여 사용하기가 편리해진다. 두번째로 집계 데이터를 생성하기 위함이다. 그룹별 기술통계나 기타 여러가지 집계하여 정리하기 위한 목적으로 사용된다.

독자들이 긴데이터와 넓은 데이터를 변경하는 방법이 헷갈리다고 말하는 경우가 많다. 대부분 연구자들이 학위논문이나 학술논문을 사용할 때, 설문지를 받는다면 대체로 와이드한 데이터이다. 따라서, 긴데이터로 변경하는 부분에 집중하여 알아보면 데이터를 사용하는 것이 편리해질 수 있다.

#### 2.2.2.1. stats::reshape 함수

데이터는 다음과 같이 생성했다. 독자도 같은 값을 얻으려면, set.seed(20240226)을 적용하여 부트스트랩의 시드(seed)값을 동일하게 해주면 된다.

set.seed(20240226)  
df <- data.frame(id = 1:12,  
 type = rep(c("mid","fin"), 6, each=TRUE),  
 y1 = sample(1:7, 12, replace=TRUE),  
 y2 = sample(1:7, 12, replace=TRUE),  
 y3 = sample(1:7, 12, replace=TRUE)  
 )  
  
# df  
  
# 다른 방법   
jjstat::df

## id type y1 y2 y3  
## 1 1 mid 1 3 1  
## 2 2 fin 4 3 2  
## 3 3 mid 3 7 7  
## 4 4 fin 1 5 6  
## 5 5 mid 4 7 5  
## 6 6 fin 4 3 3  
## 7 7 mid 7 2 5  
## 8 8 fin 5 5 2  
## 9 9 mid 6 1 2  
## 10 10 fin 6 5 7  
## 11 11 mid 1 3 7  
## 12 12 fin 4 7 6

롱데이터로 변경하는 방법은 r-base의 stats패키지의 reshaple()를 이용하는 방법이다. 함수의 기본 설정은 다음과 같다.

reshape(data, varying = NULL, v.names = NULL, timevar = "time",  
 idvar = "id", ids = 1:NROW(data),  
 times = seq\_along(varying[[1]]),  
 drop = NULL, direction, new.row.names = NULL,  
 sep = ".",  
 split = if (sep == "") {  
 list(regexp = "[A-Za-z][0-9]", include = TRUE)  
 } else {  
 list(regexp = sep, include = FALSE, fixed = TRUE)} )

**롱포맷(long format)데이터로 변경**  
reshape에서는 앞으로 소개된 롱포맷 변경함수들과 달리 변수의 앞부부을 없애고, 숫자형태로 변환한다. 예를들면 y1, y2, y3 이라는 변수가 롱포맷으로 변경될 결과를 보면 variable에는 1, 2, 3으로 변경이 되어있다. 이렇게 변경되는 것은 나중에 통계분석 활용시 다층분석(혼합모형분석)등에서 데이터 변환시 편리함을 주는 경우가 많다는 점을 참고하면 좋을듯 하다. 따라서 원래대로 돌릴 경우에는 변수명을 정확히 입력하여 변형해야 한다. 여기에서 중요한 것은 direction = ’long’라고 설정하는 부분인데, 이 옵션이 반드시 설정되어야 긴데이터로 변경이 된다.

df\_long1 = reshape(df,   
 idvar=c("id", "type"), #유지할 데이터  
 varying = list(3:5), # value가 될 변수   
 timevar = "variable", #value가 된 column  
 v.name = "score", # value의 명칭  
 direction ='long')   
df\_long1

## id type variable score  
## 1.mid.1 1 mid 1 1  
## 2.fin.1 2 fin 1 4  
## 3.mid.1 3 mid 1 3  
## 4.fin.1 4 fin 1 1  
## 5.mid.1 5 mid 1 4  
## 6.fin.1 6 fin 1 4  
## 7.mid.1 7 mid 1 7  
## 8.fin.1 8 fin 1 5  
## 9.mid.1 9 mid 1 6  
## 10.fin.1 10 fin 1 6  
## 11.mid.1 11 mid 1 1  
## 12.fin.1 12 fin 1 4  
## 1.mid.2 1 mid 2 3  
## 2.fin.2 2 fin 2 3  
## 3.mid.2 3 mid 2 7  
## 4.fin.2 4 fin 2 5  
## 5.mid.2 5 mid 2 7  
## 6.fin.2 6 fin 2 3  
## 7.mid.2 7 mid 2 2  
## 8.fin.2 8 fin 2 5  
## 9.mid.2 9 mid 2 1  
## 10.fin.2 10 fin 2 5  
## 11.mid.2 11 mid 2 3  
## 12.fin.2 12 fin 2 7  
## 1.mid.3 1 mid 3 1  
## 2.fin.3 2 fin 3 2  
## 3.mid.3 3 mid 3 7  
## 4.fin.3 4 fin 3 6  
## 5.mid.3 5 mid 3 5  
## 6.fin.3 6 fin 3 3  
## 7.mid.3 7 mid 3 5  
## 8.fin.3 8 fin 3 2  
## 9.mid.3 9 mid 3 2  
## 10.fin.3 10 fin 3 7  
## 11.mid.3 11 mid 3 7  
## 12.fin.3 12 fin 3 6

이 롱데이터를 와이드 데이터로 바꾸려면 다음과 같이 같은 함수를 사용한다. 다만 아래와 같은 방법으로 바꾸게 되면, 열(column)의 변수명이 자동으로 롱데이터에 있던 이름을 사용하게 된다.

**와이드포맷(wide format)데이터로 변경**   
앞에서 강조한 것처럼 롱포맷으로 변경된 데이터는 다시 와이드 데이터로 만들때, 컬럼명에 문자가 빠진 숫자만 있었기 때문에 열에 있던 score라는 이름을 이용하여 숫자를 붙이면서 돌아왔다. 따라서, 원래의 데이터명을 유지하려면 이러한 부분을 잘 고민해서 이름을 정하는 것이 좋다.

reshape(df\_long1,   
 idvar=c("id","type"), #유지될 데이터   
 timevar="variable", #column이 되는 변수   
 direction ="wide" )

## id type score.1 score.2 score.3  
## 1.mid.1 1 mid 1 3 1  
## 2.fin.1 2 fin 4 3 2  
## 3.mid.1 3 mid 3 7 7  
## 4.fin.1 4 fin 1 5 6  
## 5.mid.1 5 mid 4 7 5  
## 6.fin.1 6 fin 4 3 3  
## 7.mid.1 7 mid 7 2 5  
## 8.fin.1 8 fin 5 5 2  
## 9.mid.1 9 mid 6 1 2  
## 10.fin.1 10 fin 6 5 7  
## 11.mid.1 11 mid 1 3 7  
## 12.fin.1 12 fin 4 7 6

위 결과를 보면 처음에 이름을 정확히 하지 않았기 때문에 발생한 문제를 해결하는 방법은 변수명을 정확히 다시 입력해주면 되는데, 이에 대한 방법은 당므과 같다. 먼저 column의 이름을 명확히 하려면, varying이란 인수를 list로 변수명을 정확하게 입력해주면된다. 이렇게 하면 정확한 변수명이 나타나는 것을 확인할 수 있다.

reshape(df\_long1, idvar=c("id","type"), #유지될 데이터   
 timevar="variable",#column이 되는 변수   
 varying=list(c("y1","y2","y3")), #column의 정확한 변수명  
 direction ="wide" )

## id type y1 y2 y3  
## 1.mid.1 1 mid 1 3 1  
## 2.fin.1 2 fin 4 3 2  
## 3.mid.1 3 mid 3 7 7  
## 4.fin.1 4 fin 1 5 6  
## 5.mid.1 5 mid 4 7 5  
## 6.fin.1 6 fin 4 3 3  
## 7.mid.1 7 mid 7 2 5  
## 8.fin.1 8 fin 5 5 2  
## 9.mid.1 9 mid 6 1 2  
## 10.fin.1 10 fin 6 5 7  
## 11.mid.1 11 mid 1 3 7  
## 12.fin.1 12 fin 4 7 6

만약에 reshape()함수를 이용하여 일반데이터를 롱데이터로 만든 경우에는 rownames영역에 기록된 정보가 있어서 간단하게 원래대로 돌리는 것도 가능하지만, 정확한 함수의 사용법을 익히는 것이 중요하는 점을 강조하고 싶다. 아래와 같이 간단하게 변경이 되기도 하지만, 일반적으로 롱데이터를 와이드한 데이터로 바꾸는 방법을 정확히 익히도록 하자.

reshape(df\_long1,   
 direction ="wide" )

## id type y1 y2 y3  
## 1.mid.1 1 mid 1 3 1  
## 2.fin.1 2 fin 4 3 2  
## 3.mid.1 3 mid 3 7 7  
## 4.fin.1 4 fin 1 5 6  
## 5.mid.1 5 mid 4 7 5  
## 6.fin.1 6 fin 4 3 3  
## 7.mid.1 7 mid 7 2 5  
## 8.fin.1 8 fin 5 5 2  
## 9.mid.1 9 mid 6 1 2  
## 10.fin.1 10 fin 6 5 7  
## 11.mid.1 11 mid 1 3 7  
## 12.fin.1 12 fin 4 7 6

#### 2.2.2.2. tidyverse(tidyr::gather, tidyr::spread)

**gather함수를 이용한 롱데이터**

다음은 tidyverse에 포함되어 있는 tidyr패키지의 gather()와 spread()를 이용한 방법이다. 지금은 다음절에 소개된 pivot\_longer()와 pivot\_wider()를 더 추천한다고 하지만, 예전에 만들어진 대부분의 코드가 gather와 spread로 만들어거나 분석한 경우가 많아서 익혀야 하는 방법이기도 하다.  
gather 함수의 사용은 다음과 같다.

gather( data,  
 key = "key",  
 value = "value",  
 ...,  
 na.rm = FALSE,  
 convert = FALSE,  
 factor\_key = FALSE)

먼저 데이터는 앞에서 만들어진 df데이터를 사용하도록 하자. 롱포맷으로 바꾸는 것은 gather()라는 함수를 사용하는데, 이 단어의 의미대로 이해하면 될 것같다. key인수는 롱데이터로 만들면서 모이는 열변수들이 전체 이름을 정하는 것이고, value옵션은 데이터(값)들이 모인 열의 이름을 정하는 것이다. gather함수에서는 dplyr::select()함수가 자동으로 적용이 되어 ,(comma)를 이용하여 모여질 열들을 선택하면 된다. 함수를 사용할 때 ’tidyr::gather’와 같은 ’패키지명::함수’로 사용하는 것은 library(tidyverse)나 library(tidyr)을 임포트 하면 붙일 필요없으나, 본서에서는 각 함수가 어떤 패키지에서 왔는지 명확하게 알려주기 위해서 패키지명을 붙여서 사용하였다.

df\_long2 <-tidyr::gather(df,  
 key = "variable",   
 value= "score", y1, y2, y3)   
df\_long2

## id type variable score  
## 1 1 mid y1 1  
## 2 2 fin y1 4  
## 3 3 mid y1 3  
## 4 4 fin y1 1  
## 5 5 mid y1 4  
## 6 6 fin y1 4  
## 7 7 mid y1 7  
## 8 8 fin y1 5  
## 9 9 mid y1 6  
## 10 10 fin y1 6  
## 11 11 mid y1 1  
## 12 12 fin y1 4  
## 13 1 mid y2 3  
## 14 2 fin y2 3  
## 15 3 mid y2 7  
## 16 4 fin y2 5  
## 17 5 mid y2 7  
## 18 6 fin y2 3  
## 19 7 mid y2 2  
## 20 8 fin y2 5  
## 21 9 mid y2 1  
## 22 10 fin y2 5  
## 23 11 mid y2 3  
## 24 12 fin y2 7  
## 25 1 mid y3 1  
## 26 2 fin y3 2  
## 27 3 mid y3 7  
## 28 4 fin y3 6  
## 29 5 mid y3 5  
## 30 6 fin y3 3  
## 31 7 mid y3 5  
## 32 8 fin y3 2  
## 33 9 mid y3 2  
## 34 10 fin y3 7  
## 35 11 mid y3 7  
## 36 12 fin y3 6

지금은 y1, y2, y3이라는 열을 선택했는데, 거꾸로 id, type열을 빼내는 방법도 가능한데, 이때 사용하는 방법은 select()함수에 사용하는 것처럼 -(minus)를 붙이면 된다.

tidyr::gather(df,  
 key = "variable",   
 value= "score", -id, -type)

## id type variable score  
## 1 1 mid y1 1  
## 2 2 fin y1 4  
## 3 3 mid y1 3  
## 4 4 fin y1 1  
## 5 5 mid y1 4  
## 6 6 fin y1 4  
## 7 7 mid y1 7  
## 8 8 fin y1 5  
## 9 9 mid y1 6  
## 10 10 fin y1 6  
## 11 11 mid y1 1  
## 12 12 fin y1 4  
## 13 1 mid y2 3  
## 14 2 fin y2 3  
## 15 3 mid y2 7  
## 16 4 fin y2 5  
## 17 5 mid y2 7  
## 18 6 fin y2 3  
## 19 7 mid y2 2  
## 20 8 fin y2 5  
## 21 9 mid y2 1  
## 22 10 fin y2 5  
## 23 11 mid y2 3  
## 24 12 fin y2 7  
## 25 1 mid y3 1  
## 26 2 fin y3 2  
## 27 3 mid y3 7  
## 28 4 fin y3 6  
## 29 5 mid y3 5  
## 30 6 fin y3 3  
## 31 7 mid y3 5  
## 32 8 fin y3 2  
## 33 9 mid y3 2  
## 34 10 fin y3 7  
## 35 11 mid y3 7  
## 36 12 fin y3 6

**spread함수를 이용한 와이드데이터**

spread함수의 구조는 다음과 같다.

spread(data, key, value,   
 fill = NA, convert = FALSE,   
 drop = TRUE, sep = NULL)

롱포맷 데이터를 와이드한 데이터로 바꾸는 경우에 사용하는 spread함수는 key 인수에는 열의 변수로 돌아갈 이름이 들어있는 변수명을 택하고, value인수는 값으로 돌아갈 값들이 들어있는 변수를 택하면 된다.

tidyr::spread(df\_long2,   
 key = "variable", #열(column)으로 돌아갈 값들   
 value = "score") #각 열의 값이 되어야 하는 부분

## id type y1 y2 y3  
## 1 1 mid 1 3 1  
## 2 2 fin 4 3 2  
## 3 3 mid 3 7 7  
## 4 4 fin 1 5 6  
## 5 5 mid 4 7 5  
## 6 6 fin 4 3 3  
## 7 7 mid 7 2 5  
## 8 8 fin 5 5 2  
## 9 9 mid 6 1 2  
## 10 10 fin 6 5 7  
## 11 11 mid 1 3 7  
## 12 12 fin 4 7 6

#### 2.2.2.3. tidyr::pivot\_longer, pivot\_wider

tidyvers에서는 gather, spread를 솔리드버전(완전히 고정되어 없애지 않는 버전)으로 택하고, 좀 더 명확하게 이해되는 이름은 pivot\_longer와 pivot\_wider를 사용하는 것을 추천하고 있다. 특히 인수는 데이터를 어떻게 변형하는지 더 이해가 쉬운 인수명을 사용하고 있다. 또한 wide데이터로 변형하는 경우 더 다양한 집계함수와 기능을 추가하여 데이터를 분석하는데 유리한 것들을 제공하고 있다. 다만, 이를 자주 사용하는 경우 좀 더 완벽하게 익혀서 데이터 과학,데이터 분석에 효과적으로 사용하는 기능을 공부하는 부담이 좀 있지만 데이터를 분석하기 좋게 변경하는 기능에서는 복잡하지 않고 쉽게 사용이 가능하여 목적에 잘 맞게 사용하면 편리한 함수라고 보여진다.

**롱포맷으로 변경**

pivot\_longer의 기본적인 설정은 다음과 같다. 중요하게 설졍해야 하는 부분이 cols, names\_to,values\_to에 대한 부분이다.

함수의 구조는 다음과 같다.

pivot\_longer(  
 data,  
 cols,  
 ...,  
 cols\_vary = "fastest",  
 names\_to = "name",  
 names\_prefix = NULL,  
 names\_sep = NULL,  
 names\_pattern = NULL,  
 names\_ptypes = NULL,  
 names\_transform = NULL,  
 names\_repair = "check\_unique",  
 values\_to = "value",  
 values\_drop\_na = FALSE,  
 values\_ptypes = NULL,  
 values\_transform = NULL  
)

이제 df데이터를 사용하여 롱포맷으로 변경해보자.

df\_long3 = pivot\_longer(df,   
 cols = y1:y3,  
 names\_to="variable",   
 values\_to="score")  
df\_long3

## # A tibble: 36 × 4  
## id type variable score  
## <int> <chr> <chr> <int>  
## 1 1 mid y1 1  
## 2 1 mid y2 3  
## 3 1 mid y3 1  
## 4 2 fin y1 4  
## 5 2 fin y2 3  
## 6 2 fin y3 2  
## 7 3 mid y1 3  
## 8 3 mid y2 7  
## 9 3 mid y3 7  
## 10 4 fin y1 1  
## # ℹ 26 more rows

**와이드포맷으로 변경**

pivot\_wider는 data인수외에 주요한 인수가 names\_from과 values\_from인수이다. 즉 롱데이터를 와이드하게 변형할 때 변수가 될 것이 무엇인지와 값이 될 변수가 무엇인지 정하는것이다. 대체로 모든 함수들의 인수는 동일한 특성을 가지고 있으므로 이해하기 어렵기 않다고 생각된다.

함수의 구조는 다음과 같다.

pivot\_wider(  
 data,  
 ...,  
 id\_cols = NULL,  
 id\_expand = FALSE,  
 names\_from = name,  
 names\_prefix = "",  
 names\_sep = "\_",  
 names\_glue = NULL,  
 names\_sort = FALSE,  
 names\_vary = "fastest",  
 names\_expand = FALSE,  
 names\_repair = "check\_unique",  
 values\_from = value,  
 values\_fill = NULL,  
 values\_fn = NULL,  
 unused\_fn = NULL  
)

이제 데이터를 와이드 데이터로 변경해보자.

pivot\_wider(df\_long3,   
 names\_from="variable", #열이 될 이름   
 values\_from="score") #열변수의 값들

## # A tibble: 12 × 5  
## id type y1 y2 y3  
## <int> <chr> <int> <int> <int>  
## 1 1 mid 1 3 1  
## 2 2 fin 4 3 2  
## 3 3 mid 3 7 7  
## 4 4 fin 1 5 6  
## 5 5 mid 4 7 5  
## 6 6 fin 4 3 3  
## 7 7 mid 7 2 5  
## 8 8 fin 5 5 2  
## 9 9 mid 6 1 2  
## 10 10 fin 6 5 7  
## 11 11 mid 1 3 7  
## 12 12 fin 4 7 6

#### 2.2.2.4. reshape2::melt, dcast

reshape2패키지는 롱포맷과 와이드포맷을 변경하는 것을 문론 dcast함수는 집계함수로서도 동작한다. 집계함수의 기능은 추후에 소개하기로 하고 지금은 포맷을 변경하는 것에만 사용해보도록 하자.

아래의 데이터는 연습을 위해서 jjstat패키지에서 df데이터를 가져왔다.

# data 생성   
# set.seed(20240226)  
# df <- data.frame(id = 1:12,  
# type = rep(c("mid","fin"), 6, each=TRUE),  
# y1 = sample(1:7, 12, replace=TRUE),  
# y2 = sample(1:7, 12, replace=TRUE),  
# y3 = sample(1:7, 12, replace=TRUE)  
# )  
  
df <- jjstat::df

데이터를 롱포맷으로 바꾸는 방법을 위해서는 melt함수를 이해해야 한다. 함수의 구조는 다음과 같다.

melt( data, id.vars, measure.vars, variable.name = "variable",  
 ...,  
 na.rm = FALSE, value.name = "value", factorsAsStrings = TRUE)

기본 값이 variable.name = “variable”, value.name = “value”으로 설정이 되어있으므로 이 이름을 그대로 사용하려고 하면 굳이 바꿀 필요는 없다.

#롱데이터로 변경   
df\_long <- reshape2::melt(df, id.vars = c("id","type"))   
df\_long

## id type variable value  
## 1 1 mid y1 1  
## 2 2 fin y1 4  
## 3 3 mid y1 3  
## 4 4 fin y1 1  
## 5 5 mid y1 4  
## 6 6 fin y1 4  
## 7 7 mid y1 7  
## 8 8 fin y1 5  
## 9 9 mid y1 6  
## 10 10 fin y1 6  
## 11 11 mid y1 1  
## 12 12 fin y1 4  
## 13 1 mid y2 3  
## 14 2 fin y2 3  
## 15 3 mid y2 7  
## 16 4 fin y2 5  
## 17 5 mid y2 7  
## 18 6 fin y2 3  
## 19 7 mid y2 2  
## 20 8 fin y2 5  
## 21 9 mid y2 1  
## 22 10 fin y2 5  
## 23 11 mid y2 3  
## 24 12 fin y2 7  
## 25 1 mid y3 1  
## 26 2 fin y3 2  
## 27 3 mid y3 7  
## 28 4 fin y3 6  
## 29 5 mid y3 5  
## 30 6 fin y3 3  
## 31 7 mid y3 5  
## 32 8 fin y3 2  
## 33 9 mid y3 2  
## 34 10 fin y3 7  
## 35 11 mid y3 7  
## 36 12 fin y3 6

이제, 변수명을 바꾸어보자. variable.name=“New\_name”, value.name=“New\_score”으로 하여 name을 변경할 수 있다. 변수명이 어떻게 바뀌는지 확인해보도록 하자.

#새로운 변수명 설정   
reshape2::melt(df, id.vars = c("id","type"),   
 variable.name="New\_name",   
 value.name="New\_score") |>   
 head()

## id type New\_name New\_score  
## 1 1 mid y1 1  
## 2 2 fin y1 4  
## 3 3 mid y1 3  
## 4 4 fin y1 1  
## 5 5 mid y1 4  
## 6 6 fin y1 4

**와이드포맷으로 변경**

이제 dcast를 통하여 원래의 모습대로 만들어보자. dcast()에는 인수로 롱포맷데이터와 formula를 통해서 변경한다. id, type를 유지한채 variable변수와 value를 다시 넓게 펼치는 데이터로 변경하는 방법으로 하기 위해서는 id + type ~ variable라고 입력하는데, 왼쪽은 유지해야 할 변수이고, 오른쪽은 펼쳐야 할 변수이다.

reshape2::dcast(df\_long, id + type ~ variable)

## id type y1 y2 y3  
## 1 1 mid 1 3 1  
## 2 2 fin 4 3 2  
## 3 3 mid 3 7 7  
## 4 4 fin 1 5 6  
## 5 5 mid 4 7 5  
## 6 6 fin 4 3 3  
## 7 7 mid 7 2 5  
## 8 8 fin 5 5 2  
## 9 9 mid 6 1 2  
## 10 10 fin 6 5 7  
## 11 11 mid 1 3 7  
## 12 12 fin 4 7 6

**집계함수로의 dcast**

집계함수로서의 기능: dcast(데이터, formula, function)

dcast는 집계함수의 기능도 있다. 예를 들어 롱포맷으로 변경된 데이터에서 평균을 구해보자. type에 value에 대한 평균을 구해보자. 값을 구해보면 fin = 2.75, mid = 2.66으로 나타났다. type에 대한 평균을 구하기 위해서 tpye ~ 1 이라는 formula를 사용했다. ~1은 평균(혹은 절편)을 나타낸다.

type\_mean = reshape2::dcast(df\_long, type ~ 1, mean)  
type\_mean

## type 1  
## 1 fin 4.333333  
## 2 mid 4.000000

type에 value에 대한 평균을 구해보자.  
이 값에 대하여 집계함수를 사용해서 값을 찾아서 비교해보자.

# jjstat package  
jjstat::mysummaryBy(value ~ type, data = df\_long )[,1:2]

## # A tibble: 2 × 2  
## var Mean  
## <chr> <dbl>  
## 1 fin 4.33  
## 2 mid 4

만약에 varibale ~ type에 대한 것으로 각 그룹에 따른 평균을 구해보자. 결과는 행렬형태로 나타난다.

reshape2::dcast(df\_long, variable ~ type, mean)

## variable fin mid  
## 1 y1 4.000000 3.666667  
## 2 y2 4.666667 3.833333  
## 3 y3 4.333333 4.500000

다음은 집계하여 합을 구한 것이다.

reshape2::dcast(df\_long, variable ~ type, sum)

## variable fin mid  
## 1 y1 24 22  
## 2 y2 28 23  
## 3 y3 26 27

데이터의 포맷을 바꾸면서 집계함수를 직접적으로 사용하는 경우는 많지 않다. 다만, 연구자가 많이 사용하는 함수를 사용하여 집계를 하면 된다. 데이터의 형태에 따라 aggregate, apply, lapply, sapply, group\_by, summarise 등을 사용하기 때문에 각자에 맞는 집계함수를 사용하여 데이터를 구하면 된다.

#### 2.2.2.5. 변수선별하여 round처리

이 부분을 설명하는 것은 실제로 데이터를 처리하면서 자주 만나고, 반드시 해결해야 하는 중요한 처리 방법이다. round처리를 하는 경우 변수의 종류가 factor변수와 numeric변수가 섞여 있어서 이를 데이터에 처리하려면 dplyr::mutate\_if()를 사용하여 처리하면 된다. 이는 데이터를 처리할 때 중요한 기술이므로 잘 활용하는게 중요하다.

평균으로 집계하한 후에 muatate\_if를 이용하여 numeric변수만을 round처리

reshape2::dcast(df\_long, variable ~ type, mean) |>  
 dplyr::mutate\_if(is.numeric, round, 2)

## variable fin mid  
## 1 y1 4.00 3.67  
## 2 y2 4.67 3.83  
## 3 y3 4.33 4.50

평균으로 집계하한 후에 sapply를 이용하여 numeric변수만을 round처리

R-base에서 적용하는 방법은 sapply를 이용하는 방법이다. 단, 이 방법을 적용할 때에는 함수를 적용해야 한다. sapply(data, FUN)로 사용하기 때문에 FUN부분에 반복 적용되는 데이터의 변수에 대하여 numeric변수인지 확인하고 맞으면 ronud처리를 하고, 아니면 넘어가도록 설정하였다.

reshape2::dcast(df\_long, variable ~ type, mean) |>   
 sapply(function(x){if(is.numeric(x)){round(x, 2)}else{x}}) |>  
 data.frame()

## variable fin mid  
## 1 1 4.00 3.67  
## 2 2 4.67 3.83  
## 3 3 4.33 4.50

### 2.2.3. 데이터분리와 합치기

#### 2.2.3.1. separate를 이용한 데이터분리

데이터를 분리하는방법은 tidyr의 separate()사용할 수 있다. 데이터를 인수로 받아서 sep=“/”와 같이 구분자를 넣는다. 디폴트는 “/”으로 되어 있다. into=()를 이용하여 분리되는 변수명을 설정해주면 데이터가 분리된다.

separate( data, col, into,  
 sep = "[^[:alnum:]]+",  
 remove = TRUE,  
 convert = FALSE,  
 extra = "warn",  
 fill = "warn",  
 ...  
 )

데이터를 분리하는 연습을 위해서 jjstat패키지의 worldcup 데이터를 사용해보자. 이는 월드컵 개최국과 개회연도, 그리고 임의로 만들어진 골개수 데이터이다.

# worldcup data   
worldcup\_sep <- jjstat::worldcup |>   
 tidyr::separate(goal, into = c("vip\_goal", "total\_goal"), sep ="/")  
  
worldcup\_sep

## nation year vip\_goal total\_goal  
## 1 Russia 2018 46 189  
## 2 Brazil 2014 44 182  
## 3 South Africa 2010 47 142  
## 4 Germany 2006 30 186  
## 5 South Korea 2002 36 107  
## 6 Japan 1998 34 113  
## 7 France 1994 23 112  
## 8 United States 1990 40 168  
## 9 Italy 1986 18 151  
## 10 Qatar 2022 24 129

vip\_goal, total\_goal 변수를 보면 데이터가 numeric데이터임인데 분리해보니 character데이터로 나타닜다. 이때, convert = TRUE를 설정하면 데이터를 더 적합한 모습으로 변경해준다. 아래의 결과를 보면 변경된 데이터 int로 변경된 것을 알 수 있다.

jjstat::worldcup |>   
 tidyr::separate(goal, into = c("vip\_goal", "total\_goal"),   
 convert = TRUE)

## nation year vip\_goal total\_goal  
## 1 Russia 2018 46 189  
## 2 Brazil 2014 44 182  
## 3 South Africa 2010 47 142  
## 4 Germany 2006 30 186  
## 5 South Korea 2002 36 107  
## 6 Japan 1998 34 113  
## 7 France 1994 23 112  
## 8 United States 1990 40 168  
## 9 Italy 1986 18 151  
## 10 Qatar 2022 24 129

sep = 2 로하면 선택한 변수에 대하여 앞에서 부터(왼쪽부터) 두자리를 중심으로 분리한다. 변수중에서 year을 century와 year로 분리하였다. sep가 음수이면 오른쪽부터 분리된다.

jjstat::worldcup |>   
 tidyr::separate(year, into = c("century", "year"), sep = 2, convert = TRUE)

## nation century year goal  
## 1 Russia 20 18 46 / 189  
## 2 Brazil 20 14 44 / 182  
## 3 South Africa 20 10 47 / 142  
## 4 Germany 20 6 30 / 186  
## 5 South Korea 20 2 36 / 107  
## 6 Japan 19 98 34 / 113  
## 7 France 19 94 23 / 112  
## 8 United States 19 90 40 / 168  
## 9 Italy 19 86 18 / 151  
## 10 Qatar 20 22 24 / 129

#### 2.2.3.2. unite를 이용한 데이터 결합

이번에는 앞에서 만든 worldcup\_sep 데이터에서 vip\_goal과 totla\_goal을 결합해서 한개의 변수로 만들어 보자. sep =“*“에는 ’*’ 을 이용하여 변수를 결합하게 만들어 준다.

worldcup\_sep |>  
 unite(goal,vip\_goal, total\_goal, sep = "\_" )

## nation year goal  
## 1 Russia 2018 46 \_ 189  
## 2 Brazil 2014 44 \_ 182  
## 3 South Africa 2010 47 \_ 142  
## 4 Germany 2006 30 \_ 186  
## 5 South Korea 2002 36 \_ 107  
## 6 Japan 1998 34 \_ 113  
## 7 France 1994 23 \_ 112  
## 8 United States 1990 40 \_ 168  
## 9 Italy 1986 18 \_ 151  
## 10 Qatar 2022 24 \_ 129

### 2.2.4. 결측값

데이터

sleep |> str()

## 'data.frame': 20 obs. of 3 variables:  
## $ extra: num 0.7 -1.6 -0.2 -1.2 -0.1 3.4 3.7 0.8 0 2 ...  
## $ group: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ID : Factor w/ 10 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...

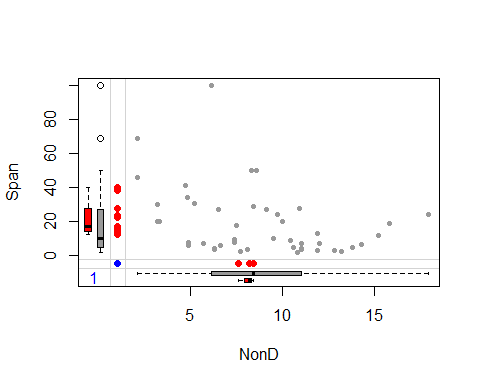
#### 2.2.4.1. 결측 확인하기

library(VIM)  
VIM::aggr(sleep, prop=FALSE, numbers = TRUE)



왼쪽 그래프는 각 변수별로 NA의 개수가 얼마나 되는지 보여주며, 오른쪽 그림은 NA의 조합이 어느 변수들의 조합으로 이루어진 것인지 보여주는 그림이다.  
누락된 자료가 없는 행은 42개라고 나타나있다. 붉은 색이 업는 것들이 NA가 없는 것이다.

VIM::marginplot(sleep[c("NonD","Span")], pch = 20,   
 col = c("gray60","red","blue"))



누락된 자료들 간의 상관관계보기

x = as.data.frame(abs(is.na(sleep)))   
# x  
y = apply(x, 2, function(x) sum(x) > 0)  
# y  
round( cor(x[y]), 2)

## NonD Dream Sleep Span Gest  
## NonD 1.00 0.91 0.49 0.02 -0.14  
## Dream 0.91 1.00 0.20 0.04 -0.13  
## Sleep 0.49 0.20 1.00 -0.07 -0.07  
## Span 0.02 0.04 -0.07 1.00 0.20  
## Gest -0.14 -0.13 -0.07 0.20 1.00

**MCAR (Missing Completely At Random), MAR (Missing At Random)**, 그리고 **MNAR (Missing Not At Random)**에 대한 개념을 이해해보자.

#### 2.2.4.2. 결측의 종류

**MCAR (Missing Completely At Random):**

MCAR는 데이터의 누락이 완전히 무작위로 발생하는 경우를 의미한다. 즉, 누락된 값이 해당 관측치의 다른 값이나 다른 변수와 관련이 없는 경우이다. 예를 들어, 설문조사에서 누락된 값이 응답자의 나이와 상관없이 무작위로 발생하는 경우가 MCAR이다. MCAR의 가정은 데이터를 삭제하거나 간단한 대체 방법으로 처리할 수 있다.

**MAR (Missing At Random):** MAR는 누락된 값이 다른 변수와 관련이 있지만 해당 관측치의 값과는 관련이 없는 경우를 의미한다. 즉, 누락된 값이 다른 변수의 값에 의해 결정되는 경우이다. 예를 들어, 연령대별로 누락된 값이 다르게 발생하는 경우가 MAR이다. 따라서, MAR의 경우 고급 대체 방법(예: MICE)을 사용하여 누락된 값을 처리해야 한다.

**MNAR (Missing Not At Random):** MNAR는 누락된 값이 관측치의 값과 관련이 있는 경우를 의미한다. 즉, 누락된 값이 다른 변수의 값에 의해 결정되는 경우이다. 예를 들어, 소득이 높은 사람들이 소득을 더 낮게 보고하는 경우가 MNAR이다. MNAR의 경우 추가적인 데이터 수집이 필요하거나 더 복잡한 대체 방법이 필요하다.

요약하면:

MCAR: 무작위로 누락됨. 간단한 대체 방법 사용 가능. MAR: 다른 변수와 관련 있으나 해당 관측치의 값과는 관련 없음. 고급 대체 방법 필요. MNAR: 누락된 값이 관측치의 값과 관련 있음. 추가 데이터 수집이 필요. 123

#### 2.2.4.3. listwise deletion 과 pariwise deletion

데이터에 NA가 있으면 결측으로 인하여 상관분석을 할 수 가 없다. 따라서 결측이 발생한 행을 지우게 된다. 이때 결측이 발생한 데이터를 제거하는 방법은 2가지가 있다. Listwise Deletion과 Pairwise Deletion은 누락된 데이터를 처리하는 방법이다. 각각의 개념과 차이점을 알아보자.

**(1) Listwise Deletion (전체 삭제):** Listwise Deletion은 누락된 값을 가진 전체 관측치를 분석에서 제외하는 방법이다. 즉, 하나 이상의 변수에서 누락된 값이 있는 경우 해당 관측치를 분석에서 제거하게 된다. 이때 분석은 데이터가 완전한 상태인 관측치에 대해서만 수행된다. 이 방법은 MCAR (Missing Completely At Random) 가정하에 사용된다.

**(2) Pairwise Deletion (부분 삭제):** Pairwise Deletion은 누락된 값을 가진 관측치를 일부만 분석에서 제외하는 방법이다. 통계적 절차는 누락된 데이터가 없는 변수를 사용하여 분석을 수행하게 된다. 예를 들어, 한 관측치에서 변수 A의 값이 누락되었더라도 변수 B와 C의 값은 분석에 활용되게 된다. 이 방법은 MAR (Missing At Random) 가정하에 사용되는 방법이다.

Listwise Deletion은 데이터를 완전히 삭제하므로 정보 손실이 크지만, Pairwise Deletion은 누락된 값을 최대한 활용하여 분석을 수행한다. 그러나 Pairwise Deletion은 각 계산된 통계량이 서로 다른 관측치의 부분 집합에 기반하여 계산되므로 주의해야 한다. 예를 들어, 상관 행렬은 Pairwise Deletion으로 계산할 경우 양의 준정부호성을 보장하지 않을 수 있다는 점을 유념해야 한다.

# Listwise Deletion   
sleep\_all = na.omit(sleep)  
cor\_all= cor(sleep\_all$BodyWgt, sleep\_all$NonD)  
nrow(sleep)

## [1] 62

nrow(sleep\_all)

## [1] 42

결측이 있으면 계산이 되지 않는다.

cor(sleep$BodyWgt, sleep$NonD)

## [1] NA

결측을 완전 제거한 경우

cor\_all= cor(sleep\_all$BodyWgt, sleep\_all$NonD)  
cor\_all

## [1] -0.3936373

결측을 부분 제거한 경우

sleep\_partial = na.omit(sleep[c("BodyWgt","NonD")])  
nrow(sleep\_partial)

## [1] 48

cor\_partial = cor(sleep\_partial$BodyWgt, sleep\_partial$NonD)  
cor\_partial

## [1] -0.3759462

cor\_all(-0.3936373), cor\_partial(-0.3759462)의 값을 비교해보면 금색 확인을 할 수 있다.

# Pairwise Deletion  
cbind.data.frame(  
 method =c("raw\_data","Listwise Deletion","Pairwise Deletion"),   
 N = c(nrow(sleep),nrow(sleep\_all) ,nrow(sleep\_partial)),  
 Cor = c("Failure to calculate.",  
 round(cor\_all, 3),   
 round(cor\_partial, 3) )  
)

## method N Cor  
## 1 raw\_data 62 Failure to calculate.  
## 2 Listwise Deletion 42 -0.394  
## 3 Pairwise Deletion 48 -0.376

#### 22.4.4. 다중 대입법(multi imputatin ,MI)

library(mice)  
sleep2 = mice(sleep, nedd=1234)

##   
## iter imp variable  
## 1 1 NonD Dream Sleep Span Gest  
## 1 2 NonD Dream Sleep Span Gest  
## 1 3 NonD Dream Sleep Span Gest  
## 1 4 NonD Dream Sleep Span Gest  
## 1 5 NonD Dream Sleep Span Gest  
## 2 1 NonD Dream Sleep Span Gest  
## 2 2 NonD Dream Sleep Span Gest  
## 2 3 NonD Dream Sleep Span Gest  
## 2 4 NonD Dream Sleep Span Gest  
## 2 5 NonD Dream Sleep Span Gest  
## 3 1 NonD Dream Sleep Span Gest  
## 3 2 NonD Dream Sleep Span Gest  
## 3 3 NonD Dream Sleep Span Gest  
## 3 4 NonD Dream Sleep Span Gest  
## 3 5 NonD Dream Sleep Span Gest  
## 4 1 NonD Dream Sleep Span Gest  
## 4 2 NonD Dream Sleep Span Gest  
## 4 3 NonD Dream Sleep Span Gest  
## 4 4 NonD Dream Sleep Span Gest  
## 4 5 NonD Dream Sleep Span Gest  
## 5 1 NonD Dream Sleep Span Gest  
## 5 2 NonD Dream Sleep Span Gest  
## 5 3 NonD Dream Sleep Span Gest  
## 5 4 NonD Dream Sleep Span Gest  
## 5 5 NonD Dream Sleep Span Gest

sleep2

## Class: mids  
## Number of multiple imputations: 5   
## Imputation methods:  
## BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred   
## "" "" "pmm" "pmm" "pmm" "pmm" "pmm" ""   
## Exp Danger   
## "" ""   
## PredictorMatrix:  
## BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger  
## BodyWgt 0 1 1 1 1 1 1 1 1 1  
## BrainWgt 1 0 1 1 1 1 1 1 1 1  
## NonD 1 1 0 1 1 1 1 1 1 1  
## Dream 1 1 1 0 1 1 1 1 1 1  
## Sleep 1 1 1 1 0 1 1 1 1 1  
## Span 1 1 1 1 1 0 1 1 1 1  
## Number of logged events: 10   
## it im dep meth out  
## 1 2 2 Span pmm Sleep  
## 2 2 2 Gest pmm Sleep  
## 3 3 5 Span pmm Sleep  
## 4 3 5 Gest pmm Sleep  
## 5 4 1 Span pmm Sleep  
## 6 5 2 Span pmm Sleep

실제 데이터의 자료를 누락시킨후에 mice을 이용하여 채운 값을 비교하여 어느정도의 차이가 발생하는지 확인하자.

#### 2.2.4.5. jjstat package missCheck, missfill

sleep |> jjstat::missCheck()

## Variable . Count  
## NonD NonD 48 14  
## Dream Dream 50 12  
## Sleep Sleep 58 4  
## Span Span 58 4  
## Gest Gest 58 4

NA가 잇는 변수의 값을 채워넣기  
다음은 median을 채운 것이다. 채원진 값에는 \* 표시가 붙는다. 예를 들면, data1에는 8.35\*라는 중앙값(median)이 채워진 것이다.

sleep$NonD |> jjstat::missfillin(fn= median)

## data1 data2 data3 data4 data5 data6 data7 data8 data9 data10 data11   
## 8.35\* 6.30 8.35\* 8.35\* 2.10 9.10 15.80 5.20 10.90 8.30 11.00   
## data12 data13 data14 data15 data16 data17 data18 data19 data20 data21 data22   
## 3.20 7.60 8.35\* 6.30 8.60 6.60 9.50 4.80 12.00 8.35\* 3.30   
## data23 data24 data25 data26 data27 data28 data29 data30 data31 data32 data33   
## 11.00 8.35\* 4.70 8.35\* 10.40 7.40 2.10 8.35\* 8.35\* 7.70 17.90   
## data34 data35 data36 data37 data38 data39 data40 data41 data42 data43 data44   
## 6.10 8.20 8.40 11.90 10.80 13.80 14.30 8.35\* 15.20 10.00 11.90   
## data45 data46 data47 data48 data49 data50 data51 data52 data53 data54 data55   
## 6.50 7.50 8.35\* 10.60 7.40 8.40 5.70 4.90 8.35\* 3.20 8.35\*   
## data56 data57 data58 data59 data60 data61 data62   
## 8.10 11.00 4.90 13.20 9.70 12.80 8.35\*

전체의 결측 채우기  
default는 mean으로 되어있다.

sleep |> jjstat::missfillinFor(fn=median ) |>  
 head(10)

## BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger  
## 1 6654.000 5712.0 8.35 1.8 3.3 38.6 645 3 5 3  
## 2 1.000 6.6 6.30 2.0 8.3 4.5 42 3 1 3  
## 3 3.385 44.5 8.35 1.8 12.5 14.0 60 1 1 1  
## 4 0.920 5.7 8.35 1.8 16.5 15.1 25 5 2 3  
## 5 2547.000 4603.0 2.10 1.8 3.9 69.0 624 3 5 4  
## 6 10.550 179.5 9.10 0.7 9.8 27.0 180 4 4 4  
## 7 0.023 0.3 15.80 3.9 19.7 19.0 35 1 1 1  
## 8 160.000 169.0 5.20 1.0 6.2 30.4 392 4 5 4  
## 9 3.300 25.6 10.90 3.6 14.5 28.0 63 1 2 1  
## 10 52.160 440.0 8.30 1.4 9.7 50.0 230 1 1 1

# 3. 기본 통계분석

기술통계 분석은 데이터를 수집하고 요약하여 데이터의 특징을 파악하는 분석 방법이다. 데이터의 중심 경향, 분포, 변동성 등을 파악하여 데이터를 이해하고 해석하는 데 반드시 필요하다.

기술통계 분석의 목적으로는 (1) 데이터를 요약하고 이해하기 위해서이다. 이는 방대한 데이터를 쉽게 이해하고 분석하기 위해 주요 특징을 요약하는 과정이다. (2)데이터의 분포 파악하기 위해서이다. 데이터가 어떻게 분포되어 있는지 파악하여 데이터의 특징을 이해할 수 있다. (3) 변수 간의 관계 파악을 위해서이다. 변수 간의 상관관계를 파악하여 데이터에 대한 통찰력을 얻을 수 있다. (4) 추론 및 가설 검정을 위해서이다. 데이터를 기반으로 추론하고 가설을 검정하는데 사용한다.

## 3.1. ㅣ기술통계(Descriptive statistics)

기본적인 분석은 가장 먼저 해야할 분석을 말한다.   
대푯값과 산포도를 중심으로 분석하게 된다.  
대표값에는 평균, 중앙값, 최빈값을 분석하게 되는데 평균은 mean(), 중앙값은 median(), 최빈값은 table()을 사용하여 분석한다.

**기술통계 분석의 종류**  
기술통계 분석을 살펴보면 다음과 같이 크게 4가지로 기술통계를 분석한다고 할 수 있다.

가. 중심 경향 측은 다음의 3가지를 주로 측정한다.  
(1)평균: 데이터 값들의 합을 데이터 개수로 나눈 값  
(2) 중앙값: 데이터 값들을 크기 순으로 정렬했을 때 중앙에 위치하는 값  
(3) 최빈값: 데이터 값들 중 가장 자주 나타나는 값

나. 분포 측정  
(1) 표준편차: 데이터 값들이 평균에서 얼마나 떨어져 있는지를 나타내는 값  
(2) 분포형: 데이터 값들이 어떤 형태로 분포되어 있는지를 나타내는 것  
(3) 사분위 범위: 데이터 값들을 4등분했을 때 각 그룹의 범위

다. 변동성 측정  
(1) 분산: 데이터 값들이 평균에서 얼마나 흩어져 있는지를 나타내는 값  
(2) 표준편차: 분산의 제곱근  
(3) 변동계수: 표준편차를 평균으로 나눈 값

라. 상관관계 측정  
(1) 상관계수: 두 변수 간의 선형적인 관계를 나타내는 값  
(2) 결정계수: 한 변수가 다른 변수를 얼마나 설명하는지를 나타내는 값

### 3.1.1. 빈도분석

빈도분석을 가장 먼저 소개하고 이야기를 하는 이유는 대체로 논문을 쓰는 과정에서 가장 먼저 대상자에 대한 것을 소개하거나 측정도구를 사용하여 분석한 내용을 소개하기 때문이다.

먼저 R에 기본으로 탑재된 mtcars 데이터를 이용하여 빈도분석을 해보자.  
데이터의 구조는 다음과 같다.  
이 데이터는 1974년 미국 Motor Trend US 잡지에서 추출된 자료로 32개의 자동차회사의 1973년~1974년 모델에 대한 자동차의 10가지의 디자인 및 성능에 대한 연료 소비량(mpg)를 나타낸 자료이다.

A data frame with 32 observations on 11 (numeric) variables.\  
 [, 1] mpg Miles/(US) gallon\  
 [, 2] cyl Number of cylinders\  
 [, 3] disp Displacement (cu.in.)\  
 [, 4] hp Gross horsepower\  
 [, 5] drat Rear axle ratio\  
 [, 6] wt Weight (1000 lbs)\  
 [, 7] qsec 1/4 mile time\  
 [, 8] vs Engine (0 = V-shaped, 1 = straight)\  
 [, 9] am Transmission (0 = automatic, 1 = manual)\  
 [,10] gear Number of forward gears\  
 [,11] carb Number of carburetors\

**데이터구조**  
데이터의 구조를 보는 좋은 방법중 하나가 str()함수를 사용하는 것이다.

library(tidyverse) # dplyr: %>% (pipe)  
data(mtcars)  
mtcars %>% str() # same as str(mtcars)

## 'data.frame': 32 obs. of 11 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

#### 3.1.1.1. table()

논문의 분석을 시작할 때, 항상 설문이나 실험참가자의 빈도분석을 통해서 제시하게 되어있는데, 이때 사용하는 것이 table함수이다.  
빈도분석은 보통 table()함수를 활용하는 경우가 많다. 빈도분석은 입력된 데이터들이 도수분포표(frequency table)상에서 어떠한 분포적 특성을 가지고 있는지 파악하는 데 사용된다.

mtcars의 am변수는 Transmission을 나타낸다 (0 = automatic, 1 = manual).  
따라서, 먼저 변수를 factor처리를 먼저해서 빈도를 분석해보자.

# mtcars$am  
data(mtcars)  
mtcars$am <- factor(mtcars$am,  
 level = c(0, 1),  
 labels = c("automatic", "manual" ))  
# frequency analysis  
table(mtcars$am)

##   
## automatic manual   
## 19 13

분석된 결과는 다음과 같이 설명한다.  
빈도분석결과, 자동차의 트랜스미션(transmission)에 대하여 오토매틱은 19개, 수동은 13개로 나타났다.  
이번에는 엔진(engine)에 대한 빈도분석을 해보자.

mtcars$vs <- factor(mtcars$vs, level=c(0, 1),  
 labels = c("V-shaped", "straight" ) )  
table(mtcars$vs)

##   
## V-shaped straight   
## 18 14

트랜스미션(transmission)별 엔진(Enigne)의 빈도분석을 해보자. 이는 table(변수1, 변수2)를 넣으면 분석이 된다. 행렬(matrix)의 형태로 빈도를 볼 수 있다.

table(mtcars$am, mtcars$vs)

##   
## V-shaped straight  
## automatic 12 7  
## manual 6 7

그런데 이러한 방법이 아닌 목록으로 정리하여 보고 싶다면 데이터프레임(data.frame)으로 변경하면 목록형태로 나타나서 논문표에서 넣기 더 편할 수 있다.

data.frame(table(mtcars$am, mtcars$vs))

## Var1 Var2 Freq  
## 1 automatic V-shaped 12  
## 2 manual V-shaped 6  
## 3 automatic straight 7  
## 4 manual straight 7

#### 3.1.1.2. count(), n()

R을 사용하는 경우 다양한 라이브러리(library)를 사용할 수 있기 때문에 분석하는 방법이 한가지로 고정되어 있는 것은 아니다. 사용하는 라이브러리를 이용하여 다양한 분석을 할 수 있다. 먼저. dplyr패키지를 이용하여 나타내는 방법도 매우 유용하다. dplyr::count()는 매우 편리한 함수이다. 사용방법은 count(데이터, 변수)로 입력하여 사용한다.

count(mtcars, am)

## am n  
## 1 automatic 19  
## 2 manual 13

# mtcars %>% count(am)

count(mtcars, am, vs)

## am vs n  
## 1 automatic V-shaped 12  
## 2 automatic straight 7  
## 3 manual V-shaped 6  
## 4 manual straight 7

좀 불편할 수 있으나, 실제 활용도가 높은 것중 하나가 n()이다. 빈도분석하는 과정에서 수시로 사용회는 함수인데, summarise()와 함께 사용하는 유용한 함수이다.

summarise()와 n()을 이용하여 다음과 같이 나타낼 수 있다.

mtcars %>%   
 group\_by(am)%>%   
 summarise(N=n())

## # A tibble: 2 × 2  
## am N  
## <fct> <int>  
## 1 automatic 19  
## 2 manual 13

#### 3.1.1.3. forcats::fct\_count()

이번에는 forcats라는 라이브러리의 fct\_count()이다. 필자의 경우도 매우 자주 사용하고 애용하는 라이브러리함수이다. 그 이유는 여러가지로 변형이 가능하기 때문이다. 그러나, fct\_count()는 vector형식으로 한가지 변수만 분석이 가능하다. 출력은 tibble데이터로 출력되어 table()와는 다른 형식으로 출력된다. fct\_count(데이터)를 입력하면 되고, 옵션으로는 sort를 이용한 정렬기능과 prop라는 비율을 나타내는 옵션이 있다.  
먼저 일반적인 빈도분석이다.

#frequency analysis : default option   
library(forcats)  
forcats::fct\_count(mtcars$am)

## # A tibble: 2 × 2  
## f n  
## <fct> <int>  
## 1 automatic 19  
## 2 manual 13

이번에는 각 빈도에 비율을 포함시켜보자.

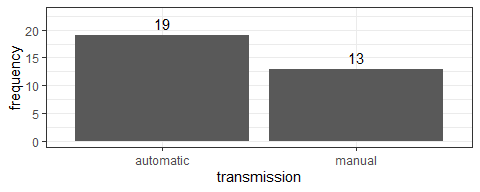
# add proportion   
forcats::fct\_count(mtcars$am, prop=TRUE)

## # A tibble: 2 × 3  
## f n p  
## <fct> <int> <dbl>  
## 1 automatic 19 0.594  
## 2 manual 13 0.406

**그래프의 시각화**

ggplot을 이용하여 그래프를 그린 후에 그래프위에 실제적인 값들을 표시하고자 할 때에는 geom\_text를 이용하여 나타낸다. geom\_text, geom\_label(텍스트를 박스에 표시)은 같은 기능을 수행한다.

library(forcats)  
  
fct\_count(mtcars$am) %>%   
 ggplot(aes(x=f, y=n))+  
 geom\_bar(stat="Identity")+  
 geom\_text(aes(label=n), vjust=-.5)+   
 labs(x="transmission", y="frequency")+  
 ylim(0, 23)+theme\_bw()



#### 3.1.1.4. 빈도의 수가 0인 경우 출력하는 방법

빈도수가 0인 것을 출력해야 하는 경우는 다양한 변수별로 빈도를 분석해서 데이터를 병합할 때 컬럼의 개수가 맞지 않으면 병합하기 어렵기 때문이다. 리커트 척도로 5개의 척도에 대하여 빈도를 분석한 경우, 0이라고 출력하지 않으면 어떤 것은 3개, 4개, 5개로 달라지기 때문에 병합할 수 없게 된다. 따라서, 빈도가 0으로 나와도 출력을 해주어야 하는 경우가 발생한다.

y <- c(0, 0, 0, 3, 4, 4, 4, 4)  
z <- c(0, 0, 1, 3, 4, 5, 5, 5)  
table(y)

## y  
## 0 3 4   
## 3 1 4

0에서 5까지의 변수중에서 개수를 센 상태인데, 2의 경우는 0개이다. 그런데 이 경우 2와 5가 0이라고 나타내고 싶은 경우는 factor로 처리하면서 2를 세도록 해야 한다.

table(factor(y, levels = 0:5))

##   
## 0 1 2 3 4 5   
## 3 0 0 1 4 0

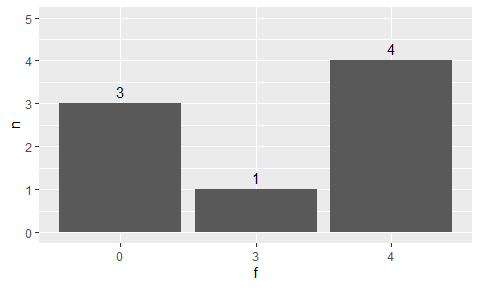
이것은 fct\_count()에서도 동일하다. 다만, fct\_count함수는 벡터에서만 사용한다.

y <- c(0, 0, 0, 3, 4, 4, 4, 4)  
#2가 0개에서 표시가 안된 경우   
fct\_count(factor(y))

## # A tibble: 3 × 2  
## f n  
## <fct> <int>  
## 1 0 3  
## 2 3 1  
## 3 4 4

2를 0개라고 표시한 경우이다.  
이를 시각화해보면 막대그래프가 4개가 그려지면서 실제 빈도수가 나타난다.

y <- c(0, 0, 0, 3, 4, 4, 4, 4)  
  
fct\_count(factor(y))%>%   
 ggplot(aes(x = f, y = n))+  
 geom\_bar(stat = "Identity")+  
 geom\_text(aes(label = n), vjust = -.5)+  
 ylim(0, 5)



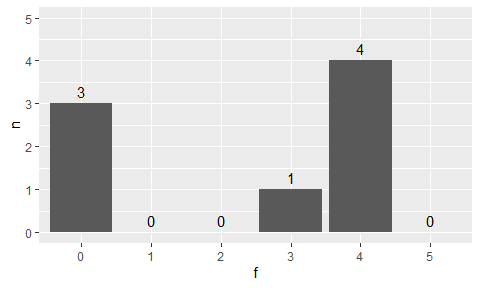
실제 개수가 없는 데이터를 표시하게 만들려면 factor에서 level을 모두 나타나게 해야 한다. 따라서 level = 0:5까지 모두 설정하면 빈도의 개수가 없는 값이 없는 것들이 개수도 나타난다.

# 개수가 없는 것은 0개로 표시한 경우   
fct\_count(factor(y, level=0:5))

## # A tibble: 6 × 2  
## f n  
## <fct> <int>  
## 1 0 3  
## 2 1 0  
## 3 2 0  
## 4 3 1  
## 5 4 4  
## 6 5 0

이것을 다시 시각화해보자. 시각화를 하면 존재하지 않는 level에는 0개로 개수가 나타난다.  
즉, 0은 3개, 3은 1개, 4는 4개가 있으나 1, 2, 5는 0개로 존재하지 않는다. 이것을 그래프로 시각화 한 것이다.  
실제로 논문에 들어가는 분석에는 0개인 것들을 표시해야 하는 경우가 많다.

fct\_count(factor(y, level = 0:5)) %>%   
 ggplot(aes(x = f, y = n))+  
 geom\_bar(stat = "Identity")+  
 geom\_text(aes(label = n), vjust = -.5)+  
 ylim(0, 5)



jjstat패키지의 Freq\_table을 이용하여 나타내보자.

jjstat::Freq\_table(factor(y))

## Variable Freq Prop  
## 1 0 3 37.5 %  
## 2 3 1 12.5 %  
## 3 4 4 50 %

이번에도 y 데이터에 대하여 factor()을 이용할 때 levels = 1:5을 이용하여 결측된 값도 표시되도록 하는 것은 앞에서도 설명을 하였다.

factor(y, levels = 1:5)

## [1] <NA> <NA> <NA> 3 4 4 4 4   
## Levels: 1 2 3 4 5

이제 levels를 표시한 데이터를 이용하여 다시한번 Freq\_table을 이용하여 나타내보자.

jjstat::Freq\_table(factor(y, levels = 1:5))

## Variable Freq Prop  
## 1 1 0 0 %  
## 2 2 0 0 %  
## 3 3 1 12.5 %  
## 4 4 4 50 %  
## 5 5 0 0 %

### 3.1.2 기술통계분석

#### 3.1.2.1 대푯값

대푯값에는 평균, 중앙값, 최빈값등이 있다.  
평균을 구할 때에는 mean(), 중앙값은 median(), 최빈값을 구할 때에는 table()을 사용하는 것이 일반적이다.

평균 :주어진 데이터의 합을 데이터 개수로 나눈 값으로, 데이터의 중심을 나타내는 대표값으로 통계적 의미는 데이터의 중심 위치를 파악하는데 사용된다.

mean(mtcars$mpg)

## [1] 20.09062

분산: 산포도를 나타내는 지표로서 편차의 제곱의 평균값이다. 분산은 데이터가 얼마나 넓게 퍼져있는지를 나타내며, 편차를 제곱하여 평균을 구한 값이다. 분산은 단위를 사용하지 않는다.

var(mtcars$mpg)

## [1] 36.3241

표준편차 :데이터가 평균에서 얼마나 퍼져있는지를 나타내는 값이며, 분산의 양의 제곱근으로 정의된다. 데이터의 산포도를 측정하며, 값이 작을수록 데이터가 평균 주변에 모여있음을 의미한다.

sd(mtcars$mpg) ; sd(mtcars$mpg)^2 #variance

## [1] 6.026948

## [1] 36.3241

**왜도**  
왜도는 데이터 그룹의 대칭성을 나타내는 척도이다. 왜도가 0이면 데이터는 대칭적이며, 왜도가 양수이면 오른쪽으로 치우쳐 있고,왜도가 음수이면 왼쪽으로 치우쳐 있다는 것을 의미한다. 왜도의 활용을 알아보면, 왜도는 데이터 분포 파악에 사용된다. 왜도를 통해 데이터 분포가 대칭적인지, 아니면 왼쪽 또는 오른쪽으로 치우쳐 있는지를 알 수 있다. 두번째로는 정규 분포 가정 검정한다. 많은 통계 분석 방법들은 데이터가 정규 분포를 따른다고 가정하는데, 왜도를 통해 데이터가 정규 분포를 따르는지 검정할 수 있다. 또한 **이상치 감지**에도 사용된다. 왜도가 크면 데이터 그룹에 이상치가 존재할 가능성이 높다고 추정할 수 있다. 또한 모델 선택 및 평가에도 사용되는데, 왜도를 통해 모델이 데이터를 잘 설명하는지 평가할 수 있다. 왜도는 데이터 그룹의 크기에 영향을 받는다. 즉 데이터 그룹의 크기가 작으면 왜도가 크게 나타날 수 있다는 점이다. 왜도는 이상치에 영향을 받기 때문에, 데이터 그룹에 이상치가 존재하면 왜도가 크게 나타날 수 있다 공식은 다음과 같다.

* : 각 값과 평균의 차이의 세제곱의 합
* : 데이터 그룹의 개수
* 표준편차: 데이터 그룹의 표준편차

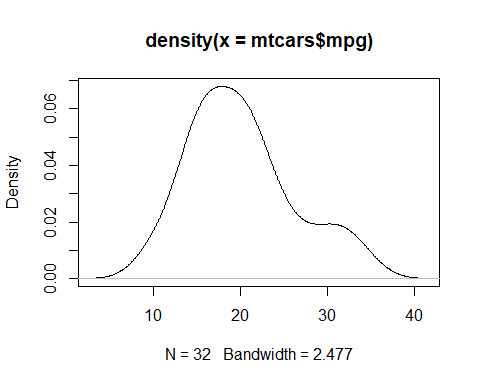
R-base에서는 왜도(skewness)를 직접 구하는 함수가 없어서, jjstat 패키지의 SKEW()함수를 이용하였다.

jjstat::SKEW(mtcars$mpg)

## [1] 0.610655

밀도 그래프를 그려보면 다음과 같다.

plot(density(mtcars$mpg))



**첨도**  
첨도는 분포의 정점 또는 편평도 정도를 측정하는 통계적 척도이다. 데이터 분포의 꼬리나 특이치의 존재 여부를 측정하느데, 높은 첨도는 뾰족한 분포를, 낮은 첨도는 평평한 분포를 나타낸다. 정규분포의 첨도는 보통 3으로 알려져 있다.

* 는 각 데이터 값,
* 은 평균,
* 는 표준편차,
* 은 데이터 개수

첨도(kurtosis)를 구하는 함수도 jjstat 패키지의 KURT()함수를 활용하여 구하였다.

jjstat::KURT(mtcars$mpg)

## [1] -0.372766

이상으로 기술통계에 적용하는 함수들을 알아보았다. 이 외에도 max, min, range 등이 있다.

max(mtcars$mpg); min(mtcars$mpg);range(mtcars$mpg)

## [1] 33.9

## [1] 10.4

## [1] 10.4 33.9

sprintf()를 이용하여 각각의 계산값을 나타내었다. sprintf()함수는 포맷팅을 하는 함수이므로 이 함수를 이용하여 문자와 함께 결과를 나타낼 때 편리하다.

sprintf("mtcars의 mpg데이터의 최댓값은 %.2f이다.",max(mtcars$mpg) )

## [1] "mtcars의 mpg데이터의 최댓값은 33.90이다."

sprintf("mtcars의 mpg데이터의 최솟은 %.2f이다.",min(mtcars$mpg) )

## [1] "mtcars의 mpg데이터의 최솟은 10.40이다."

sprintf("mtcars의 mpg데이터의 존재하는 범위는 [%.2f, %.2f]이다.", range(mtcars$mpg)[1], range(mtcars$mpg)[2])

## [1] "mtcars의 mpg데이터의 존재하는 범위는 [10.40, 33.90]이다."

**왜도와 첨도에 대한 패키지의 차이**

각 통계 패키지마다 첨도를 계산하는 공식의 차이가 있다. 예를 들어, 분모에 사용되는 표준편차의 정의, 편향 보정 여부 등이 다를 수 있다. 어떤 패키지는 기준값으로 0을 사용하고, 어떤 패키지는 3을 사용하는 등 기준값이 다를 수 있다. 왜도와 첨도를 계산할 때는 패키지와 함수를 선택할 때 주의해야 한다. 분야마다 기준이 다를 수 있으며, 통계 패키지마다 설정이 다르기 때문에 결과를 비교할 때 주의가 필요하다.

**왜도**

1. momnets 패키지의 경우 공식

moments 패키지의 왜도계산 공식

$$m\_r = \frac{{\sum\_i (x\_i - \mu)^r}}{{n}}$$  
여기서, $x\_i$는 비결측값인 데이터 요소, $n$은 데이터의 개수, $\mu$는 평균이다.

1. e1071 패키지의 경우

skewness()는 3가지의 계산 유형을 제공하고 있다.

유형 1:

이는 많은 오래된 교재에서 사용되는 정의이다.

유형 2:

SAS와 SPSS에서 사용되는 공식이다.

유형 3:

MINITAB과 BMDP에서 사용되는 공식이다.

# library(e1071)  
#library(moments)  
#유형1   
# skewness(mtcars$mpg, 1)  
sprintf("교과서에서 사용하는 왜도 : %.7f", e1071::skewness(mtcars$mpg, 1))

## [1] "교과서에서 사용하는 왜도 : 0.6106550"

#유형2   
# skewness(mtcars$mpg, 2)  
sprintf("SAS와 SPSS에서 사용되는 공식: %.7f", e1071::skewness(mtcars$mpg, 2))

## [1] "SAS와 SPSS에서 사용되는 공식: 0.6106550"

#유형3   
# skewness(mtcars$mpg, 3)  
sprintf("MINITAB과 BMDP에서 사용되는 공식: %.7f", e1071::skewness(mtcars$mpg, 3))

## [1] "MINITAB과 BMDP에서 사용되는 공식: 0.6106550"

#  
# moments::skewness(mtcars$mpg)  
sprintf("moments 패키지를 사용한 공식: %.7f", moments::skewness(mtcars$mpg))

## [1] "moments 패키지를 사용한 공식: 0.6404399"

# moments::skew(mtcars$mpg)  
sprintf("psych 패키지를 사용한 공식: %.7f", psych::skew(mtcars$mpg))

## [1] "psych 패키지를 사용한 공식: 0.6106550"

**첨도**

1. momnets 패키지의 경우 공식 moments 패키지의 첨도계산 공식

여기서, 는 비결측값인 데이터 요소, 은 데이터의 개수, 은 평균이다.

1. e1071, psych 패키지의 경우

유형 1: 오래된 정의

이는 많은 오래된 교재에서 사용되는 정의이다.

유형 2: 정규분포와 유사한 경우에 가장 적합

SAS와 SPSS에서 사용되는 공식이다.

유형 3: 정규분포가 아닌 경우에 더 좋은 성능

MINITAB과 BMDP에서 사용되는 공식이다.

# library(e1071)  
#library(moments)  
#유형1   
# kurtosis(mtcars$mpg, 1)  
sprintf("교과서에서 사용하는 왜도 : %.7f", e1071::kurtosis(mtcars$mpg, 1))

## [1] "교과서에서 사용하는 왜도 : -0.3727660"

#유형2   
# kurtosis(mtcars$mpg, 2)  
sprintf("SAS와 SPSS에서 사용되는 공식: %.7f", e1071::kurtosis(mtcars$mpg, 2))

## [1] "SAS와 SPSS에서 사용되는 공식: -0.3727660"

#유형3   
# kurtosis(mtcars$mpg, 3)  
sprintf("MINITAB과 BMDP에서 사용되는 공식: %.7f", e1071::kurtosis(mtcars$mpg, 3))

## [1] "MINITAB과 BMDP에서 사용되는 공식: -0.3727660"

#  
# moments::kurtosis(mtcars$mpg)  
sprintf("moments 패키지를 사용한 공식: %.7f", moments::kurtosis(mtcars$mpg))

## [1] "moments 패키지를 사용한 공식: 2.7994668"

# psych::kurtosi(mtcars$mpg)  
sprintf("psych 패키지를 사용한 공식: %.7f", psych::kurtosi(mtcars$mpg))

## [1] "psych 패키지를 사용한 공식: -0.3727660"

참고로 수치를 나타낼 때, comma등을 넣으려면 formmat함수를 활용하면 되는데, 이를 jjstat패키지에서 comma()함수를 활용해보자.

comma(1234567)

## [1] "1,234,567"

천단위의 숫자에 ‘,’(comma)를 사용하여 나타낼 수 있다. format함수를 이용하여 나타낼 수도 있다.

format(1234567, digits = 2, big.mark = ",")

## [1] "1,234,567"

#### 3.1.2.2. 집계함수에 대하여

다음은 집계함수들을 모아둔 것이다. 일반적으로 apply군이라고 불리며 사용법이 비슷하지만, 나름대로의 특징을 가지고 있다.  
apply, sapply, lapply, tapply, vapply 등 여러 가지 함수들은 복잡하게 만들어졌다기보다는 데이터와 상황에 맞게 사용할 수 있도록 만들어진 함수라는 점이다. 따라서 입력되는 데이터와 출력되는 데이터, 그리고 활용에 대한 이해가 필요하다.

표. apply함수군

| **함수** | **기능** | **사용법** | **입력 데이터** | **출력 데이터** |
| --- | --- | --- | --- | --- |
| **apply** | 배열, 행렬, 데이터프레임 등에서 함수 적용 | apply(X, MARGIN, FUN, ...) | 다차원 데이터 구조 | 배열, 행렬, 리스트 |
| **sapply** | 리스트나 벡터의 각 원소에 함수 적용 | sapply(X, FUN, ...) | 리스트, 벡터 | 벡터, 리스트 |
| **lapply** | 리스트의 각 원소에 함수 적용 | lapply(X, FUN, ...) | 리스트 | 리스트 |
| **tapply** | 요인(factor) 수준별로 함수 적용 | tapply(X, INDEX, FUN, ...) | 벡터, 리스트 | 리스트, 배열 |
| **vapply** | 함수를 적용할 데이터 유형 명시 | vapply(X, FUN, FUN.VALUE, ...) | 리스트, 벡터 | 벡터, 리스트 |

#### 3.1.2.3. apply의 기본이해와 활용

**apply**  
  
apply() 함수는 R에서 배열, 행렬, 데이터프레임과 같은 다차원 데이터 구조에서 함수를 적용하는 데 사용되는 벡터화된 함수이다. 이 함수를 사용하면 반복문을 사용하지 않고도 복잡한 작업을 수행할 수 있다는 장점이 있다.

apply() 주로 MARGIN 인수를 설정하여 행 또는 열에 함수를 적요한다. 기본 구문은 apply(X, MARGIN, FUN, …)로 활용하게 된다. X는 배열, 행렬, 데이터프레임으로 압력한다.  
MARGIN: 1 (행), 2 (열), 또는 c(1, 2) (행과 열 모두)이고, 3이면 table단위로 입력된다. FUN: 적용할 함수이다.  
기존에 존재하는 함수와 직접 함수를 정하며 사용할 수 있다.

1. 기본적인 사용

데이터는 연습을 위해서 stat\_onl데이터를 이용하여 해보자.

x\_df = stat\_onl|> select(5:7) |> slice(1:5)  
x\_df

## upgrade satisfy fatigue  
## 1 4 3 3  
## 2 6 4 1  
## 3 5 4 3  
## 4 7 7 2  
## 5 6 6 2

# apply(1, mean)

행단위로 집계를 하려면 MARGIN = 1로하면 된다. FUN = sum을 입력하면 행단위로 합을 구하게 된다. 이를 적용해서 합을 구해보자.  
MARJIN = 1

apply(x\_df, 1, sum)

## [1] 10 11 12 16 14

평균을 구할 때에는 FUN = mean으로 하면된다.

apply(x\_df, 1, mean)

## [1] 3.333333 3.666667 4.000000 5.333333 4.666667

MARGIN = 2로 하면 열단위로 집계한다.

apply(x\_df, 2, sum)

## upgrade satisfy fatigue   
## 28 24 11

이러한 부분을 실제 데이터내에서 활용하려면 tidyvers와 함께 사용하여 데이터를 변환하거나 조작할 때 사용하면 편리하다. 각종 기술통계 함수를 활용하여 데이터에 추가할 수 있다.

x\_df |>   
 mutate(Mean = apply(x\_df, 1, mean),  
 Sum = apply(x\_df, 1, sum),  
 Max = apply(x\_df, 1, max),  
 Min = apply(x\_df, 1, min),  
 Median = apply(x\_df, 1, median)  
 )

## upgrade satisfy fatigue Mean Sum Max Min Median  
## 1 4 3 3 3.333333 10 4 3 3  
## 2 6 4 1 3.666667 11 6 1 4  
## 3 5 4 3 4.000000 12 5 3 4  
## 4 7 7 2 5.333333 16 7 2 7  
## 5 6 6 2 4.666667 14 6 2 6

행에 합을 추가하는 방법은 rbind를 이용하여 하단에 합을 추가할 수 있다. 데이터에 적용하는 방법은 다른 함수와 함께 적욯하면 적절하게 데이터를 조절하고 정리할 수 있다.

x\_df |>   
 rbind(  
 apply(x\_df, 2, sum) )

## upgrade satisfy fatigue  
## 1 4 3 3  
## 2 6 4 1  
## 3 5 4 3  
## 4 7 7 2  
## 5 6 6 2  
## 6 28 24 11

**array에서의 활용**

y <- array(1:24, dim = c(4,3,2))  
#2nd menthod   
# y <- 1:24  
# dim(y)<-c(4,3,2)  
y

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 5 9  
## [2,] 2 6 10  
## [3,] 3 7 11  
## [4,] 4 8 12  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 13 17 21  
## [2,] 14 18 22  
## [3,] 15 19 23  
## [4,] 16 20 24

paste() 함수를 활용하여 데이터를 합쳐보았다.

apply(y, MARGIN = 1, FUN = paste)

## [,1] [,2] [,3] [,4]  
## [1,] "1" "2" "3" "4"   
## [2,] "5" "6" "7" "8"   
## [3,] "9" "10" "11" "12"  
## [4,] "13" "14" "15" "16"  
## [5,] "17" "18" "19" "20"  
## [6,] "21" "22" "23" "24"

합구하기

apply(y, MARGIN = 1, FUN = sum)

## [1] 66 72 78 84

이번에는 MARGIM =2로 하여 column을 중심으로 붙여보자.

apply(y, MARGIN = 2, FUN = paste)

## [,1] [,2] [,3]  
## [1,] "1" "5" "9"   
## [2,] "2" "6" "10"  
## [3,] "3" "7" "11"  
## [4,] "4" "8" "12"  
## [5,] "13" "17" "21"  
## [6,] "14" "18" "22"  
## [7,] "15" "19" "23"  
## [8,] "16" "20" "24"

apply(y, MARGIN = 2, FUN = sum)

## [1] 68 100 132

MARGIN = 3으로 하면 table단위로 합친다.

apply(y, MARGIN = 3, paste)

## [,1] [,2]  
## [1,] "1" "13"  
## [2,] "2" "14"  
## [3,] "3" "15"  
## [4,] "4" "16"  
## [5,] "5" "17"  
## [6,] "6" "18"  
## [7,] "7" "19"  
## [8,] "8" "20"  
## [9,] "9" "21"  
## [10,] "10" "22"  
## [11,] "11" "23"  
## [12,] "12" "24"

합치는 구조는 이해하기 어렵지 않다.  
이번에는 row끼지 collapse해서 합쳐보자. 마진을 1로 한 결과이다.

apply(y, 1, paste, collapse ="-")

## [1] "1-5-9-13-17-21" "2-6-10-14-18-22" "3-7-11-15-19-23" "4-8-12-16-20-24"

마진을 2로 하면 열끼지 합치게 된다.

#col 합치기   
apply(y, 2, paste, collapse ="-")

## [1] "1-2-3-4-13-14-15-16" "5-6-7-8-17-18-19-20" "9-10-11-12-21-22-23-24"

마진을 3으로 하여 table에 적용해보자.

# 3은 table에 적용   
apply(y, MARGIN = 3, paste, collapse ="-")

## [1] "1-2-3-4-5-6-7-8-9-10-11-12" "13-14-15-16-17-18-19-20-21-22-23-24"

#### 3.1.2.4. array 데이터 1:1 결합

행끼리 매칭하는 방법

#행끼리 1:1매칭   
apply(y, MARGIN = c(1,2), paste, collapse ="-")

## [,1] [,2] [,3]   
## [1,] "1-13" "5-17" "9-21"   
## [2,] "2-14" "6-18" "10-22"  
## [3,] "3-15" "7-19" "11-23"  
## [4,] "4-16" "8-20" "12-24"

열끼리 매칭하는 방법

#열끼지 1:1 매칭   
apply(y, MARGIN = c(2,1), paste, collapse ="-")

## [,1] [,2] [,3] [,4]   
## [1,] "1-13" "2-14" "3-15" "4-16"   
## [2,] "5-17" "6-18" "7-19" "8-20"   
## [3,] "9-21" "10-22" "11-23" "12-24"

#### 3.1.2.5. Titanic데이터를 이용한 범주형 자료분석

Titanic %>% str()

## 'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...  
## - attr(\*, "dimnames")=List of 4  
## ..$ Class : chr [1:4] "1st" "2nd" "3rd" "Crew"  
## ..$ Sex : chr [1:2] "Male" "Female"  
## ..$ Age : chr [1:2] "Child" "Adult"  
## ..$ Survived: chr [1:2] "No" "Yes"

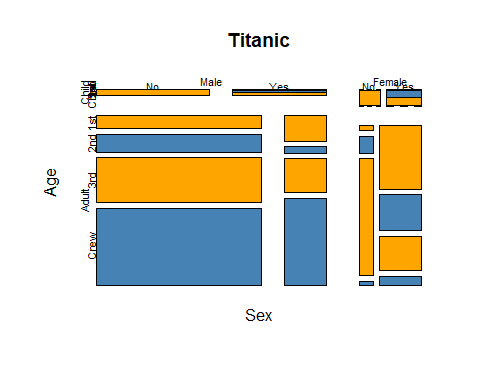
Titanic 데이터보기

Titanic

## , , Age = Child, Survived = No  
##   
## Sex  
## Class Male Female  
## 1st 0 0  
## 2nd 0 0  
## 3rd 35 17  
## Crew 0 0  
##   
## , , Age = Adult, Survived = No  
##   
## Sex  
## Class Male Female  
## 1st 118 4  
## 2nd 154 13  
## 3rd 387 89  
## Crew 670 3  
##   
## , , Age = Child, Survived = Yes  
##   
## Sex  
## Class Male Female  
## 1st 5 1  
## 2nd 11 13  
## 3rd 13 14  
## Crew 0 0  
##   
## , , Age = Adult, Survived = Yes  
##   
## Sex  
## Class Male Female  
## 1st 57 140  
## 2nd 14 80  
## 3rd 75 76  
## Crew 192 20

모자이크 플롯을 이용한 범주형 데이터 구조 파악

## Formula interface for tabulated data:  
mosaicplot(~ Sex + Age + Survived + Class, data = Titanic,   
 color =c("orange","steelblue") )



apply함수를 이용하여 탑장자를 분석해보자.

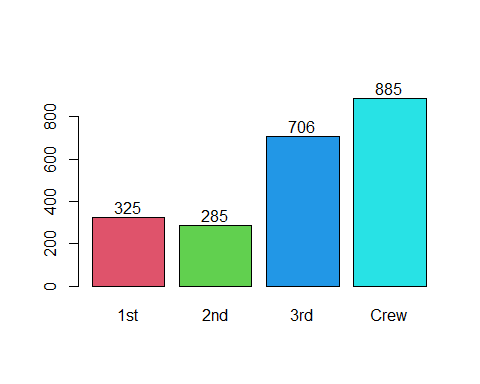
1. 탑승자분석

apply(Titanic, 1, sum)

## 1st 2nd 3rd Crew   
## 325 285 706 885

전체적으로 승무원과 3등석에 많이 탄 것을 알 수 있다.

apply(Titanic, 1, sum) %>%   
 barplot(col =2:5, ylim =c(0, 980))%>%   
 text(labels = apply(Titanic, 1, sum),  
 y = apply(Titanic, 1, sum) + 50)



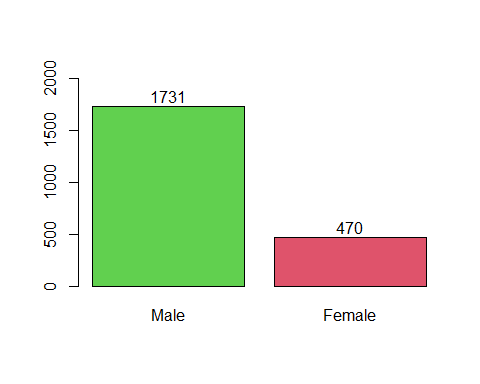
1. 성별 탑승자 분석

apply(Titanic, 2, sum)

## Male Female   
## 1731 470

시각화

apply(Titanic, 2, sum)%>%   
 barplot(col =3:2, ylim =c(0, 2000))%>%   
 text(labels = apply(Titanic, 2, sum),  
 y= apply(Titanic, 2, sum)+100, cex = 1)

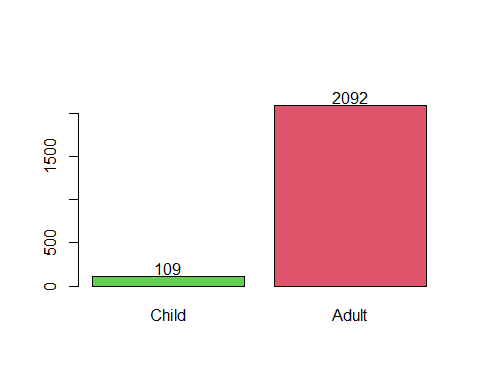


1. 연령별 탐승자 분석

apply(Titanic, 3, sum)

## Child Adult   
## 109 2092

apply(Titanic, 3, sum)%>%   
 barplot(col =3:2, ylim =c(0, 2400))%>%   
 text(labels = apply(Titanic, 3, sum),  
 y= apply(Titanic, 3, sum)+100, cex = 1)

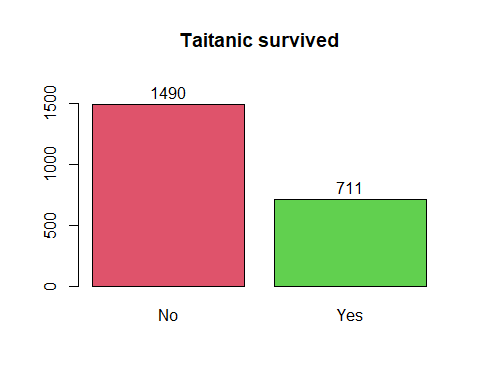


1. 생존자의 사망자 빈도

apply(Titanic, 4, sum)

## No Yes   
## 1490 711

apply(Titanic, 4, sum)%>%   
 barplot(col =2:3, ylim =c(0, 1700),  
 main="Taitanic survived")%>%   
 text(labels = apply(Titanic, 4, sum),  
 y = apply(Titanic, 4, sum)+100, cex = 1)

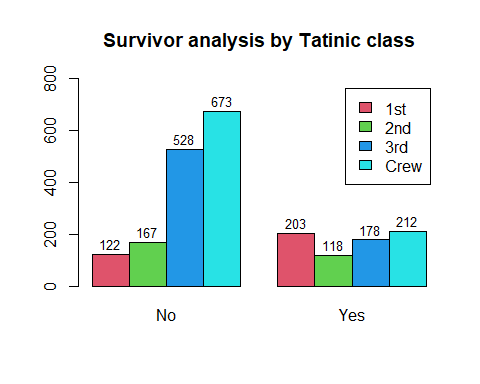


1. 좌석별 생존자 분석

# Class survived  
apply(Titanic, MARGIN = c(1,4), sum)

## Survived  
## Class No Yes  
## 1st 122 203  
## 2nd 167 118  
## 3rd 528 178  
## Crew 673 212

apply(Titanic, MARGIN = c(1,4), sum)%>%   
 barplot(col =2:5, ylim =c(0, 800), beside=TRUE,  
 main="Survivor analysis by Tatinic class",  
 legend.text = rownames(apply(Titanic,   
 MARGIN = c(1,4), sum)) )%>%   
 text(labels = apply(Titanic, c(1,4), sum),  
 y = apply(Titanic, c(1,4), sum)+ 40, cex = 0.8)

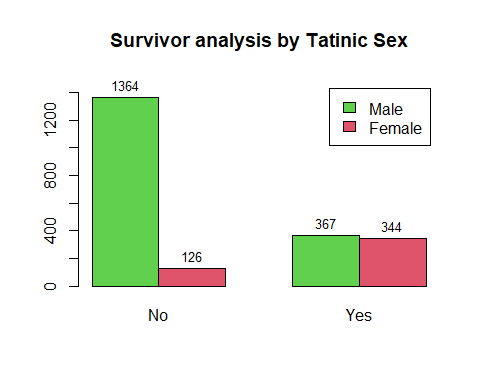


1. 성별 생존자 분석

apply(Titanic, c(2,4), sum)

## Survived  
## Sex No Yes  
## Male 1364 367  
## Female 126 344

apply(Titanic, c(2,4), sum)%>%   
 barplot(col = 3:2, ylim =c(0, 1500), beside=TRUE,  
 main="Survivor analysis by Tatinic Sex",  
 legend.text = rownames(apply(Titanic,   
 MARGIN = c(2,4), sum)) )%>%   
 text(labels = apply(Titanic, c(2,4), sum),  
 y = apply(Titanic, c(2,4), sum)+ 90, cex = 0.8)

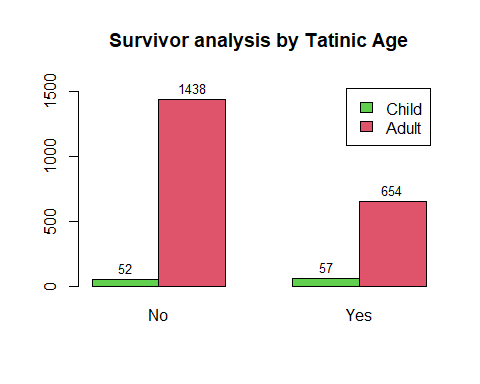


1. 연령별 생존자 분석

#Age survived  
apply(Titanic, c(3,4), sum)

## Survived  
## Age No Yes  
## Child 52 57  
## Adult 1438 654

apply(Titanic, c(3,4), sum)%>%   
 barplot(col = 3:2, ylim =c(0, 1600), beside=TRUE,  
 main="Survivor analysis by Tatinic Age",  
 legend.text = rownames(apply(Titanic,   
 MARGIN = c(3,4), sum)) )%>%   
 text(labels = apply(Titanic, c(3,4), sum),  
 y = apply(Titanic, c(3,4), sum)+ 90, cex = 0.8)



### 3.1.3. 정규성 검정

정규성 검정이란 데이터가 정규분포를 따르는지 확인하는 과정을 말한다. 정규분포는 평균을 중심으로 좌우 대칭인 분포를 말하는데, 정규성 검정은 모수적 방법과 비모수적 방법으로 수행되게 된다. 가설 검정을 수행하기 전에 데이터의 정규성을 확인하는 것이 일반적이다. 정규성 검정을 하는 이유는 데이터 정규성을 가정한 통계 검정들의 결과가 믿을 수 있는지 없는지를 체크하는 것인데, 즉 믿을만한 데이터인지 검정하는 과정이라고 볼 수 있다.  
정규성 검정 절차로 알려진 몇가지 유명한 것들이 있는데, 첫째, Shapiro-Wilk 검정이다. 작은 샘플 크기에서 사용되며, 정규성을 확인하는데 사용한다. 둘째, Kolmogorov-Smirnov 검정은 주로 큰 샘플 크기에서 사용되며, 분포의 차이를 확인하는데 사용한다. 셋째로 Anderson-Darling 검정: 다양한 분포에 대해 정규성을 검정한다고 알려져 있다.

#### 3.1.3.1. Shapiro-Wilk 검정

가설 검정을 수행하기 전에 데이터의 정규성을 확인하며, 샘플 크기가 작을 때 특히 유용하다. 정규성이 확보되면 t-검정이나 분산 분석(ANOVA)과 같은 모수적 방법을 사용하게 된다.

귀무가설 (H₀): 표본의 모집단이 정규분포를 따른다.   
대립가설 (H₁): 표본의 모집단이 정규분포를 따르지 않는다.

검정 통계량 (W)은 다음과 같이 계산한다.

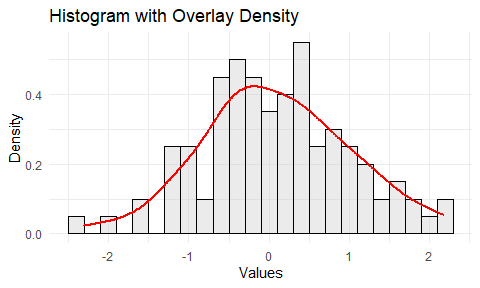
여기서 는 순서통계량(order statistic) 는 정규분포를 갖는 계수  
가설검정에서 중요한 것은 p> 0.05 일때 정규성이 확보된다는 점을 잊어서는 안되겠다.

# 데이터 생성  
set.seed(123)  
data <- rnorm(100)  
data |> head(30)

## [1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774 1.71506499  
## [7] 0.46091621 -1.26506123 -0.68685285 -0.44566197 1.22408180 0.35981383  
## [13] 0.40077145 0.11068272 -0.55584113 1.78691314 0.49785048 -1.96661716  
## [19] 0.70135590 -0.47279141 -1.06782371 -0.21797491 -1.02600445 -0.72889123  
## [25] -0.62503927 -1.68669331 0.83778704 0.15337312 -1.13813694 1.25381492

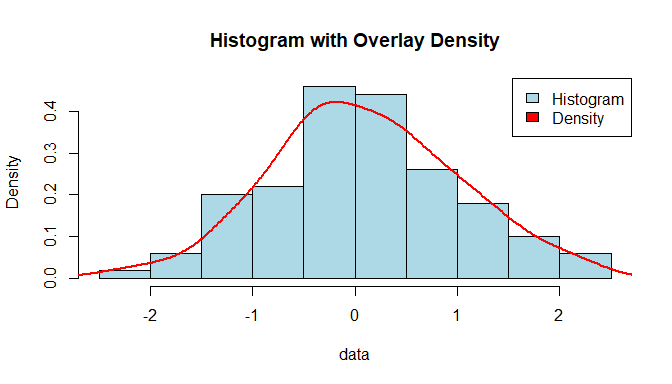
데이터의 정규성을 확인하기 위해서 히스토그램과 밀도그래프를 그렸다.

# density plot   
ggplot(data.frame(data), aes(x = data)) +  
 geom\_histogram(aes(y = ..density..), alpha = 0.4,  
 binwidth = 0.2, color = "black", fill = "gray80") +  
 geom\_density(color = "red", size = 1) +  
 labs(title = "Histogram with Overlay Density",  
 x = "Values", y = "Density") +  
 theme\_minimal()



R-base에서 그리는 방법도 있다. 독자의 편리성에 따라서 잘 익숙해지는 방법을 찾아서 그리면 될 것이다.

# 히스토그램 그리기  
hist(data, probability = TRUE,   
 col = "lightblue",   
 main = "Histogram with Overlay Density")  
# 밀도 그래프 그리기  
lines(density(data), col = "red", lwd = 2)  
# 범례 추가  
legend("topright",   
 legend = c("Histogram", "Density"),   
 fill = c("lightblue", "red"))



# 샤피로 윌크 검정 수행  
shapiro.test(data)

##   
## Shapiro-Wilk normality test  
##   
## data: data  
## W = 0.99388, p-value = 0.9349

분석결과를 보면 다음과 같다.

Shapiro-Wilk normality test  
  
 data: data  
 W = 0.99388, p-value = 0.9349

p-value = 0.9349로서 p > 0.05이므로 정규성이 나타나는 자료라고 할 수 있다. shapiro.test는 한번에 한개의 변수만을 검정할 수 있다.

다음은 jjstat::report\_stat함수를 이용하여 정규성 검정에 대한 해석결과를 보고자 한다. report\_stat()함수는 인수로 데이터를 받아서, type을 결정하면된다. type=“normality”로 되어있으면 정규성에 대한 검정결과를 보여준다. 특히 정규성을 검정하는 nomality는 기본설정(default)으로 되어있다.

shapiro.test(data) |>   
 jjstat::report\_stat("normality")

## [1] "Shapiro-Wilk normality test showed that the normality was secured by rejecting the alternative hypothesis because it was not statistically significant. statistic = 0.994, p = 0.935)"

#같은 방법   
# shapiro.test(data) |>   
# jjstat::report\_stat()

기본은 영어로 나타내게 되어있는데, 이를 한국어로 번역하여 나타내고 싶다면, trans옵션에서 TRUE로 두면 한국어로 번역해서 나타내준다. jjstat패키지의 번역프로그램을 이용하는 것이므로 우리말로는 자연습지 않을 수도 있으니 적절하게 이해해야 한다.

shapiro.test(data) |>   
 jjstat::report\_stat("normality", trans = TRUE)

##   
##   
## Shapiro-Wilk 정규성 검정결과 통계적으로 유의하지 않아 대립가설을 기각하여 정규성을 확보하였다. 통계 = 0.994, p = 0.935)

여러개의 변수에 대하여 정규성 검정을 해야 한다면 jjstat패키지의 shapiro\_test\_all()를 이용한다. 선택한 변수에 대한 정규성 검정을 해준다. 검정해야 할 변수가 많은 경우 이것을 활용하면 편리할 것이다.

data <- mtcars[,c("mpg","disp","hp","drat","wt","qsec")]  
data|>  
 jjstat::shapiro\_test\_all()

## variable W p.value hypotheses  
## W mpg 0.948 0.123 Normality \*   
## W1 disp 0.920 0.021 reject  
## W2 hp 0.933 0.049 reject  
## W3 drat 0.946 0.110 Normality \*   
## W4 wt 0.943 0.093 Normality \*   
## W5 qsec 0.973 0.594 Normality \*

#### 3.1.3.2. Kolmogorov-Smirnov 검정

Kolmogorov-Smirnov 검정은 데이터의 누적 분포 함수와 비교하고자 하는 분포의 누적 분포 함수 간의 최대 거리를 통계량으로 사용하는 가설 검정 방법이다. 주로 정규성 검정에서 사용되며, 다른 분포와도 비교할 수 있다는 장점이 있다. Kolmogorov-Smirnov 검정에서는 최대 거리를 검정통계량으로 사용한다. 최대 거리는 데이터의 누적 분포 함수와 비교하고자 하는 분포의 누적 분포 함수 간의 차이를 측정한다.

귀무가설 (H₀): 표본의 모집단이 정규분포를 따른다.  
대립가설 (H₁): 표본의 모집단이 정규분포를 따르지 않는다.

검정통계량 D는 다음과 같다.

여기서 는 데이터의 누적 분포 함수(CDF)이고, 은 정규분포의 CDF를 나타낸다.

데이터를 하나 생성해서 검정을 시행해보자.

ksdata<- c(6, 1, -4, 8, -2, 5, 0)  
mean(ksdata); sd(ksdata)

## [1] 2

## [1] 4.434712

**평균과 표준편차를 가정하여 정규분포와 비교하는 방법**

먼저, 평균과 표준편차를 가정하여 데이터가 정규분포를 따르는지 확인하는 방법이 있다. 이는 데이터의 분포를 미리 가정해서 확인하고 시행하는 방법이므로 강력한 방법이라고 할수 있다.

# Kolmogorov-Smirnov 검정 수행  
ks.test(ksdata, "pnorm",   
 mean = mean(ksdata),   
 sd = sd(ksdata))

##   
## Exact one-sample Kolmogorov-Smirnov test  
##   
## data: ksdata  
## D = 0.1792, p-value = 0.9494  
## alternative hypothesis: two-sided

**데이터의 분포를 정규분포와 비교하는 방법**  
반면에 평균과 표준편차를 가정하지 않는 경우는 데이터의 분포만을 기준하여 정규분포와 비교하는 방법으로 매우 유연한 방법으로 비교하는 것이라고 할 수 있다.

ksdata<- c(6, 1, -4, 8, -2, 5, 0)  
ks.test(ksdata, "pnorm")

##   
## Exact one-sample Kolmogorov-Smirnov test  
##   
## data: ksdata  
## D = 0.42857, p-value = 0.1106  
## alternative hypothesis: two-sided

jjstat패키지의 report\_stat()를 이용하여 결과를 확인해보자. 앞에서 보여주었던 것처럼 report\_stat()함수의 경우는 정규성 검정결과는 정규성에 맞게 디폴트 설정대로 사용하면 된다.

ks.test(ksdata, "pnorm") |>  
 jjstat::report\_stat()

## [1] "Exact one-sample Kolmogorov-Smirnov test showed that the normality was secured by rejecting the alternative hypothesis because it was not statistically significant. statistic = 0.429, p = 0.111)"

#### 3.1.3.3. Anderson-Darling 검정

Anderson-Darling 검정은 데이터의 적합도를 평가하는 데 사용된다. 특히 적합선(선택된 분포에 근거)과 비모수적 계단 함수(표시 점에 근거) 사이의 영역을 측정하는 방법을 사용하는데, Anderson-Darling 값이 작을수록 분포가 데이터를 보다 잘 적합시킨다고 할 수있다.

는 순서통계량(order statistic)이며, 는 해당 순서통계량의 누적 분포 함수(CDF) 값을 나타낸다.  
Anderson-Darling검정은 nortest 패키지의 ad.test를 사용한다. 인수는 벡터를 받는다. mtcars의 mpg데이터에 대한 정규성을 검정해보자.

# library(nortest)  
nortest::ad.test(mtcars$mpg)

##   
## Anderson-Darling normality test  
##   
## data: mtcars$mpg  
## A = 0.57968, p-value = 0.1207

비교를 위해서 ks.test에서 활용한 데이터도 한번 검정해보자.

ksdata<- c(6, 1, -4, 8, -2, 5, 0)  
 nortest::ad.test(ksdata)  
 "nortest::ad.test(ksdata)에서 다음과 같은 에러가 발생했습니다:sample size must be greater than 7"

적어도 7개이상은 되어야 한다는 메시지가 나타난다.  
데이터를 임의로 추가하여 검정해보자.

nortest::ad.test(c(6, 1, -4, 8, -2, 5, 1, 3))

##   
## Anderson-Darling normality test  
##   
## data: c(6, 1, -4, 8, -2, 5, 1, 3)  
## A = 0.16164, p-value = 0.912

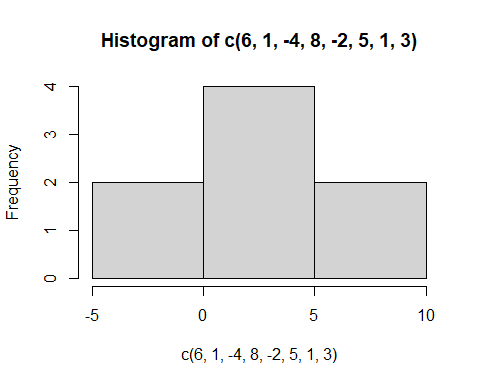
결과는 정규성이 있는 결과로 나타났다. 그래서 이를 히스토 그램에서 그려보자. 대략적인 모습을 보니 결과가 나온 이유가 추측이 된다. 히스토그램의 모양은 정규분포의 모습을 가지고 있다고 보여진다.

jjstat::report\_stat()를 이용하여 결과를 정리해보자.

nortest::ad.test(c(6, 1, -4, 8, -2, 5, 1, 3)) |>  
 jjstat::report\_stat(trans = FALSE)

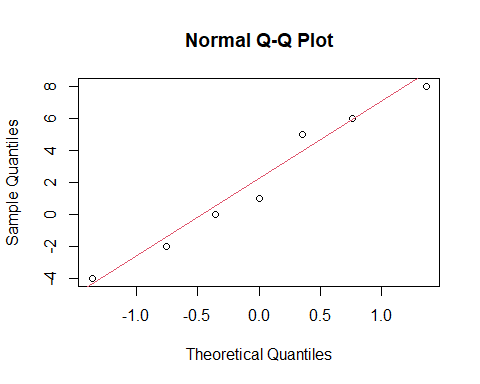
## [1] "Anderson-Darling normality test showed that the normality was secured by rejecting the alternative hypothesis because it was not statistically significant. statistic = 0.162, p = 0.912)"

#정밀하게 확인하기   
# jjstat::histogram(c(6, 1, -4, 8, -2, 5, 1, 3))  
hist(c(6, 1, -4, 8, -2, 5, 1, 3))



추가적으로 정규성을 검정하는 방법은 Q-Q (Quantile-Quantile) Plot을 이용하는 방법도 있다. Q-Q Plot을 그려보고, 데이터가 정규분포를 따르는지 확인하는 방법은 만약 점들의 분포가 45도 직선상에 모여 있게 되면, 정규분포를 따른다고 할 수 있다.  
이를 시각화 하면 다음과 같이 할 수 있다.

ksdata<- c(6, 1, -4, 8, -2, 5, 0)  
# Q-Q Plot 그리기  
qqnorm(ksdata) # 점 형태로 그리기  
qqline(ksdata, col = 2) # 근사선 그리기



## 3.2. 척도분석

이 부분은 처음 통계를 공부하거나 R을 사용하여 분석을 하는 경우 쉽지 않은 부분이다. 따라서, 이 부분을 과감히 넘어가는 것도 가능하다. 이 과정을 먼저 소개하는 이유는 실제로 논문을 쓸 때 측정도구(설문지)를 이용하여 측정된 내용에 대하여 문항에 대한 신뢰도와 타당도를 분석한 후에 통계분석에 들어가기 때문이다. 따라서, 실제 연구절차에 맞추어서 척도분석 파트를 먼저 소개하려고 한다. 사회과학 분야에서는 매우 중요하게 다루는 영역이기 때문이다(백영민, 2018).

척도 분석은 설문조사나 척도의 타당성과 신뢰성을 평가하는 과정이다. 척도 분석을 통해 척도가 연구 목적에 맞게 잘 구성되었는지,측정하고자 하는 개념을 정확하게 측정하는지 판단하거나 확인할 수 있다.  
척도 분석은 크게 타당성 분석과 신뢰도 분석으로 나눌 수 있다.

**가. 신뢰도 분석**

신뢰도 분석은 척도가 일관되고 정확하게 측정하는지 확인하는 과정이다. 신뢰도 분석에는 여러 가지 방법이 있으며, 대표적인 방법으로는 다음과 같은 것들이 있다.

1. 테스트-재테스트 신뢰도: 척도를 두 번 이상 측정하여 결과가 일관되는지 확인하는 방법이다.
2. 분할 신뢰도: 척도를 여러 개의 하위 척도로 나누어 각 하위 척도의 신뢰도를 확인하는 방법이다.
3. 내적 일관성 계수: 척도의 문항들이 서로 얼마나 일관성이 있는지 확인하는 방법이다. 대표적인 내적 일관성 계수로는 크론바흐 알파 (Cronbach’s alpha)가 있으며, 대부분의 논문에서는 크론바흐 알파를 사용한다.

**나. 타당성 분석**

타당성 분석은 척도가 측정하고자 하는 개념을 실제로 측정하는지 확인하는 과정이다. 타당성 분석에는 여러 가지 방법이 있으며, 대표적인 방법으로는 다음과 같은 것들이 있다.  
(1) 내적 타당성: 문항들이 측정하고자 하는 개념을 잘 나타내는지 확인하는 분석이다.  
(2) 구성 타당성: 척도의 요인 구조가 연구자가 가정한 모델과 일치하는지 확인하는 분석이다.  
(3) 차별 타당성: 척도가 다른 개념과 구별되는지 확인하는 분석이다.  
(4) 외부 타당성: 척도 결과가 다른 척도나 기준과 일치하는지 확인하는 분석이다.

척도 분석이 중요한 이유는 연구 결과의 신뢰성과 타당성을 확보할수 있기 때문이다. 척도 분석을 통해 척도의 문제점을 파악하고 개선하여 연구 결과의 질을 높일 수 있다는 점에서 매우 중요하다고 할 수 있다.

### 3.2.1. 신뢰도 분석

신뢰도 분석은 설문조사 또는 척도의 신뢰성을 평가하는 방법이다. 즉 신뢰성은 측정 도구가 얼마나 일관되고 정확하게 측정한 것인지를 나타내는 척도로 활용된다. 특히 가장 많이 사용되는 것이 **크론바흐 알파 (Cronbach’s alpha)**인데, 신뢰도 분석에 가장 많이 사용되는 지표이다. 크론바흐 알파는 복수의 변수들이 단일차원에 속해있는지 살펴볼 때 가장 많이 활용하는 통계치이기도 하다(백영민, 2018). 이 지표는 설문 문항 간의 내적 일관성을 측정하여 척도의 신뢰성을 평가하는데 자주 사용된다. 크론바흐 알파 값은 0에서 1 사이이며, 값이 높을수록 척도의 신뢰성이 높다는 것을 의미합하는데, 일반적으로 0.7 이상이면 신뢰성이 높다고 판단한다(Cronbach, 1951). 하지만, 연구 분야, 척도의 특성, 표본 크기 등에 따라 기준값은 달라질 수 있으니 0.7보다 작다고해서 무조건 삭제하거나 문제있다고 판단해서는 안된다. 크론바흐 알파는 척도의 신뢰성을 평가하는 데 유용한 지표이다. 하지만, 단독으로 사용하기보다는 다른 지표들과 함께 고려하여 척도의 신뢰성을 판단하는 것이 바람직하다고 할 수 있다.

: 문항 수  
: i번째 문항의 분산  
: 전체 문항들의 분산

#### 3.2.1.1. 신뢰도 분석

**신뢰도를 높이는 방법**

* 문항을 명확하게 작성하는 것이 중요하다. 참여자가 문항을 읽고 정확한 의미를 알수 있게 만들어야 신뢰도가 높아지는 것이다. 특히 한 문항안에 2가지 이상의 질문을 넣지 않도록 하는 것이 중요한데, 대체로 문항을 만들면서 자신도 모르게 여러가지를 물어보는 경우가 있다. 예를 들면, ’선생님은 친절하면서 잘 가르친다’라고 1~7점을 표시하라고 했다면, 이는 친절함을 측정하라는 것인지 잘 가르치는 것을 측정하라는 것인지 헷갈릴 수 있다는 것이다.
* 문항의 내용을 일관되게 유지한다.
* 불필요한 문항을 제거한다. 선행연구를 통해서 측정하고자 하는 문항만을 사용하는 것이 좋다.
* 예비 조사를 통해 문항을 수정한다.

wemen 데이터에 대해여 alpha함수를 적용해보자. 이는 설문데이터는 아니지만 간단히 설명하기 위해서 임의로 적용한 것이므로 독자들은 여기에서 나타난 숫자에는 큰 의미부여를 하지 않길 바란다.

women |>   
 psych::alpha() |>   
 summary()

##   
## Reliability analysis   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.69 1 1 1 442 0.0094 101 10 1

Raw Alpha값은 크론바흐 알파 계수는 0.693으로 나타났다. 이 값은 설문지의 항목들이 얼마나 일관되게 응답되었는지를 나타내는데, 이 값이 높다는 것은 설문지의 항목의 신뢰도가 높다는 것을 의미한다. Standardized Alpha (표준화된 알파) 계수는 0.998로 나타났다. 이 값은 설문지의 항목들이 얼마나 일관되게 응답되었는지를 표준화하여 나타낸 값이다. G6 (Guttman’s Lambda 6)계수는 0.995이었다. 이 값은 설문지의 항목들이 얼마나 일관되게 응답되었는지를 다른 방식으로 평가한 값이다. 선형회귀를 기준으로 계산된 과 비슷한 통계치이며, 활용하는 빈도는 높지 않은 편이다. Average Inter-Item Correlation (평균 항목간 상관계수)는 0.995인데, 이값은 설문지의 항목들이 얼마나 상호 연관되어 있는지를 나타내는 피어슨 상관계수의 평균값이다.  
S/N는 신호대비 잡음비(Signal-to-Noise Ratio)이며, S/N은 441.92로 나타났다. 설문지의 신호(일관성)와 노이즈(불일치성)의 비율을 나타내며 측정문항과 측정문항들 사이의 피어슨 상관계수를 이용하여 계산한 값인데, 자주 사용하지 않는다. Alpha Standard Error는 알파 계수의 표준 오차이며, Item Statistics는 항목 통계로 평균과 표준편차를 모두 나타낸다. Non-Missing Response Frequency는 응답 빈도, 각 항목에 대한 응답 빈도를 나타낸다. check.keys = TRUE라는 옵션은 간혹 나타나는 역코딩되어야할 변수에 대한 것을 점검해주고, 그것을 감안해서 측정해주는 옵션이다.

**데이터로 이해하는 신뢰도 분석**

이번에는 jjstat패키지의 stat\_onl(박중희, 2023)데이터의 일부를 이용하여 신뢰도를 분석해보자. 여기에서 학습앱만족도에 관한 요소로 upgrade(사용향상감), satisfy(사용만족도), fatigue(사용피로감)에 대한 측정 결과이다. 이중에서 fatigue요소는 역코딩을 해야하는 변수인데, 분석의 절차를 이해하고 확인하기 위해서 원래의 변수를 먼저 그대로 넣어서 분석을 해보도록 하자.

stat\_onl |>  
 dplyr::select(upgrade, satisfy, fatigue) |>   
 psych::alpha()|>  
 summary()

## Some items ( fatigue ) were negatively correlated with the first principal component and   
## probably should be reversed.   
## To do this, run the function again with the 'check.keys=TRUE' option  
## Reliability analysis   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.23 0.078 0.38 0.027 0.084 0.14 3.9 0.9 -0.29

이 결과를 보면 다음과 같은 메지시가 출력된다.

Warning message:  
 In psych::alpha(dplyr::select(stat\_onl, upgrade, satisfy, fatigue)) :  
 Some items were negatively correlated with the first principal component and probably should be reversed.To do this, run the function again with the 'check.keys=TRUE' option

신뢰도 분석 결과 매우 낮은 값인 0.2345379이라는 값이 나타났다.  
일부 항목에서 주요 구성요소와 부적 상관이 나타나고 있으니 ‘check.keys=TRUE’ 라는 옵션을 추가하라는 것이 함수를 다시 해보라는 것이다. 아마도 처음에 말한 것처럼 역코딩 되어야 할 변수 사용피로감(fatigue)의 문제로 보인다. 이번에는 check.keys=TRUE라는 옵션을 넣어서 분석해보자.

stat\_onl |>  
 dplyr::select(upgrade, satisfy, fatigue) |>   
 psych::alpha(check.keys=TRUE)|>  
 summary()

##   
## Reliability analysis   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.7 0.078 0.38 0.027 0.084 0.059 4.6 1.1 -0.29

이번에는 신뢰도 값이 0.7(=0.6994)로 높은 값으로 변하였다. 출력되는 메시지도 변경되었다.

Some items were negatively correlated with the first principal component and were automatically reversed. This is indicated by a negative sign for the variable name.

함수에서 알아서 fatigue변수를 역코딩을 해서 출력했다는 의미이다. 그러나, 크론바흐 알파를 분석하면서 임의로 역코딩해서 분석한 것이지 실제로 역코딩이 된 것은 아니다. 그렇다면 summary된 결과가 아니라 실제 세부적인 것들을 점검할 필요가 있으므로 한번 확인해보자.

stat\_onl |>  
 dplyr::select(upgrade, satisfy, fatigue) |>   
 psych::alpha(check.keys=TRUE)

##   
## Reliability analysis   
## Call: psych::alpha(x = dplyr::select(stat\_onl, upgrade, satisfy, fatigue),   
## check.keys = TRUE)  
##   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.7 0.078 0.38 0.027 0.084 0.059 4.6 1.1 -0.29  
##   
## 95% confidence boundaries   
## lower alpha upper  
## Feldt 0.55 0.7 0.80  
## Duhachek 0.58 0.7 0.81  
##   
## Reliability if an item is dropped:  
## raw\_alpha std.alpha G6(smc) average\_r S/N alpha se var.r med.r  
## upgrade 0.43 -0.81 -0.29 -0.29 -0.45 0.130 NA -0.29  
## satisfy 0.46 -0.89 -0.31 -0.31 -0.47 0.127 NA -0.31  
## fatigue- 0.80 0.81 0.68 0.68 4.20 0.047 NA 0.68  
##   
## Item statistics   
## n raw.r std.r r.cor r.drop mean sd  
## upgrade 68 0.86 0.77 0.77 0.65 4.3 1.5  
## satisfy 68 0.87 0.78 0.78 0.62 4.4 1.6  
## fatigue- 68 0.61 0.23 -0.44 0.32 5.2 1.1  
##   
## Non missing response frequency for each item  
## 1 2 3 4 5 6 7 miss  
## upgrade 0.04 0.06 0.18 0.24 0.29 0.12 0.07 0  
## satisfy 0.09 0.03 0.15 0.21 0.24 0.22 0.07 0  
## fatigue 0.10 0.25 0.50 0.09 0.03 0.00 0.03 0

fatigue- 로 표시가바뀌면서 갑은 0.609로 나타났는데, 이는 alpha함수에서 알아서 역코딩 처리를 한후에 분석했다는 의미이다. upgrade도 0.86, satisfy도 0.87로 비교적 높은 값을 나타내고 있다. 여기에서 확인해보아야 하는 것은 Reliability if an item is dropped항목이다.

Reliability if an item is dropped:  
 raw\_alpha std.alpha G6(smc) average\_r S/N alpha se var.r med.r  
upgrade 0.43 -0.81 -0.29 -0.29 -0.45 0.130 NA -0.29  
satisfy 0.46 -0.89 -0.31 -0.31 -0.47 0.127 NA -0.31  
fatigue- 0.80 0.81 0.68 0.68 4.20 0.047 NA 0.68

이것은 upgrade를 제거(drop)을 시키면, 전체 신뢰도는 0.43으로 변한다는 의미지고, fatigue- 를 제외(drop)시키면 크론바흐 알파값은 0.80으로 변한다는 의미이다.  
예를 들어, fatigue(사용피로감)변수를 제외한 상태에서 분석해보자.

#fatiguefmf 제거(drop)한 후 분석  
jjstat::stat\_onl |>  
 dplyr::select(upgrade, satisfy) |>   
 psych::alpha(check.keys = TRUE) |>  
 summary()

##   
## Reliability analysis   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.8 0.81 0.68 0.68 4.2 0.047 4.4 1.4 0.68

여기에 나타난 값은 크론바흐 알파값이 0.8049024라는 높은 값으로 변경되었다.

**신뢰도가 좋지 않은 경우 연구자가 고민할 것은…**  
  
이쯤 되면 연구자 혹은 독자들은 유혹에 빠지기 쉽다. 신뢰도를 높이기 위해서 신뢰도를 낮추는 문항들을 사정없이 숙청(drop)시키면서 제외시키는 행동을 하기 쉽다는 것이다. 그러나, 잊지 말아야 할 것은 수치적으로 신뢰도를 높이는 것은 위험한 행동이다. 즉, 연구목적을 위해 임의로 설정한 변수, 혹시 몰라 추가로 확인하는 변수들이 있을 수 있는데, 이는 처음부터 제외할 수 도 있는 변수들이 있다는 것이다. 이러한 것들이 신뢰도에 영향을 주면 빼내도 된다. 그러나, 그것이 아니라 반드시 사용해야 하는 변수라면, 마구 지워서는 안된다. 신뢰도가 잘못된 중요한 이유를 먼저 설명한 것들이 있는데, 이를 참고하여 신뢰도를 높이는 연구를 해야한다. 측정도구(설문지)부터 재점검하고 제대로 다시 측정하는 것이 좋다. 따라서, 점검이 안된 설문지를 본 조사에 사용하는 것은 지극히 좋지 않은 행동이다. 따라서, 측정도구를 완성할 때 좀 더 정밀하게 만들어야 하고, 선행연구를 통해서 미리 연구하는 것이 중요하다.

이제 fatigue변수를 역코딩하여 넣어보자. 역코딩은 2.2.1.4.절에서 이미 이야기를 하였다. 리커트 척도의 제일 큰값에서 1을 더한후에 변수의 값을 빼면 된다.

먼저 fatigue값을 보자.이 값은 1~7점으로 된 리커트 척도이다.

stat\_onl$fatigue

## [1] 3 1 3 2 2 2 4 3 5 3 3 2 2 4 3 3 3 3 5 4 3 3 3 2 3 3 3 2 2 2 3 2 1 2 2 3 3 2  
## [39] 1 1 3 3 3 3 2 1 3 3 3 2 7 3 3 3 4 1 3 2 3 4 3 7 4 3 3 1 3 2

이를 역코딩하는 값은 다음과 같다.

fatigue\_reverse = 8 - stat\_onl$fatigue  
fatigue\_reverse

## [1] 5 7 5 6 6 6 4 5 3 5 5 6 6 4 5 5 5 5 3 4 5 5 5 6 5 5 5 6 6 6 5 6 7 6 6 5 5 6  
## [39] 7 7 5 5 5 5 6 7 5 5 5 6 1 5 5 5 4 7 5 6 5 4 5 1 4 5 5 7 5 6

stat\_onl데이터에는 이를 미리 계산하여 둔 fatigue\_inverse 변수가 미리 계산되어 있어서 이것을 사용하여 분석해보면 된다.

데이터는 upgrade, satisfy, fatigue\_inverse(역코딩)데이터를 이용하여 신뢰도를 분석해보았다.

stat\_onl |>  
 dplyr::select(upgrade, satisfy, fatigue\_inverse) |>   
 psych::alpha()|> summary()

##   
## Reliability analysis   
## raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
## 0.7 0.69 0.65 0.42 2.2 0.059 4.6 1.1 0.31

이제 데이터를 정리하는 방법을 알아보자.

stat\_onl |>  
 dplyr::select(upgrade, satisfy, fatigue\_inverse) |>   
 jjstat::alpha\_table()

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 .1 alpha = 0.699, 95%CI[0.55, 0.8] upgrade 0.4263113  
## 2 .2 alpha = 0.699, 95%CI[0.55, 0.8] satisfy 0.4593506  
## 3 .3 alpha = 0.699, 95%CI[0.55, 0.8] fatigue\_inverse 0.8049024

다음과 같이 표로 만들어서 활용할 수 있다. 약간의 편집을 통해서 논문에 넣을 수 있는 표로 만들어질 수 있다.  
  
  
표. 온라인앱 사용만족도 신뢰도 분석

| Var | alpha\_95%CI | subfactor | cronbach alpha |
| --- | --- | --- | --- |
| .1 |  | upgrade | 0.4263113 |
| .2 | alpha = 0.699, 95%CI[0.55, 0.8] | satisfy | 0.4593506 |
| .3 |  | fatigue\_inverse | 0.8049024 |

논문에 넣을 때 필요한 부분은 표로 정리한 모습이다. 이러한 부분에서 편리함을 제공한다.  
그런데 이렇게 한개의 요인만 하는 것이 아니라 여러 개의 요인을 분석하여 정리하는 경우는 매번 분석한 내용을rbid혹은 bind\_rows()함수를 이용하여 결합해야 한다.

결과를 보면, 아래와 같이 결합할 수 있다.그런데 생각보다 코드는 복잡한 모습을 가지고 있다.

dplyr::bind\_rows(  
stat\_onl %>% select(S\_Review, S\_Add\_learn, S\_Feedback) %>%  
 jjstat::alpha\_table(show="data", variable = "A" ),  
stat\_onl %>% select(SE\_Place, SE\_Time) %>%  
 jjstat::alpha\_table(show="data", variable = "B"),  
stat\_onl %>% select(On\_Satisfy, On\_Joy, On\_Easy, On\_Engage) %>%  
 jjstat::alpha\_table(show="data", variable = "C"),  
stat\_onl %>% select(upgrade,satisfy,fatigue\_inverse) %>%  
 jjstat::alpha\_table(show="data", variable = "D")  
 )

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 A1 alpha = 0.84, 95%CI[0.76, 0.9] S\_Review 0.7923222  
## 2 A2 alpha = 0.84, 95%CI[0.76, 0.9] S\_Add\_learn 0.7705651  
## 3 A3 alpha = 0.84, 95%CI[0.76, 0.9] S\_Feedback 0.7708075  
## 4 B1 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Place 0.8558135  
## 5 B2 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Time 0.7227740  
## 6 C1 alpha = 0.877, 95%CI[0.82, 0.92] On\_Satisfy 0.8238116  
## 7 C2 alpha = 0.877, 95%CI[0.82, 0.92] On\_Joy 0.8292648  
## 8 C3 alpha = 0.877, 95%CI[0.82, 0.92] On\_Easy 0.8653063  
## 9 C4 alpha = 0.877, 95%CI[0.82, 0.92] On\_Engage 0.8491109  
## 10 D1 alpha = 0.699, 95%CI[0.55, 0.8] upgrade 0.4263113  
## 11 D2 alpha = 0.699, 95%CI[0.55, 0.8] satisfy 0.4593506  
## 12 D3 alpha = 0.699, 95%CI[0.55, 0.8] fatigue\_inverse 0.8049024

사실 복잡하기 보다는 같은 것이 반복된 것뿐인데, 복잡해보인다. 그러나, 이러한 부분에서 신뢰도 테이블을 빠르게 만들고 논문에 넣는 것은 jjstat패키지를 이용하면 된다.  
jjstat::bind\_alpha\_table() 함수를 이용하면이러한 부분에서 편리하게 결합된 내용을 만들 수 있다. 예를 들어보면 다음과 같다.

## select variable   
bind\_alpha\_table(data = stat\_onl,   
 c("S\_Review", "S\_Add\_learn", "S\_Feedback")) # 변수 선택

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 a1 alpha = 0.84, 95%CI[0.76, 0.9] S\_Review 0.7923222  
## 2 a2 alpha = 0.84, 95%CI[0.76, 0.9] S\_Add\_learn 0.7705651  
## 3 a3 alpha = 0.84, 95%CI[0.76, 0.9] S\_Feedback 0.7708075

여기에서 주의할 것은 bind\_alpha\_table(data = stat\_onl, c(S\_Review, S\_Add\_learn, S\_Feedback))와 같은 방법은 열을 찾아낼 수 없고, “S\_Review”와 같이 따옴표를 넣어서 해주어야 한다.  
만약에 변수의 열 번호를 알고 있다면 변수의 열번호를 입력하여 구할 수도 있다. 변수명이 아니라 열번호에 어떤 변수가 있는지 안다면 입력을 쉽게하여 분석할 수도 있다.

#select col number   
bind\_alpha\_table(data = stat\_onl, c(9:11))

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 a1 alpha = 0.84, 95%CI[0.76, 0.9] S\_Review 0.7923222  
## 2 a2 alpha = 0.84, 95%CI[0.76, 0.9] S\_Add\_learn 0.7705651  
## 3 a3 alpha = 0.84, 95%CI[0.76, 0.9] S\_Feedback 0.7708075

# 같은 방법   
# bind\_alpha\_table(data = stat\_onl, c(9, 10, 11))

다변량 데이터를 한번에 하려면 모든 변수를 순서대로 입력하면된다. 예를 들면 다음과 같다. 데이터를 입력하고 c()함수를 이용하여 한번에 묶어서 요인으로 크론바흐 알파값을구할 것들을 묶어서 입력하면 된다. 결과는 아래와 같다.

# multivariate data  
jjstat::bind\_alpha\_table( stat\_onl,   
 c("S\_Review", "S\_Add\_learn", "S\_Feedback"),  
 c("SE\_Place", "SE\_Time"),  
 c("On\_Satisfy", "On\_Joy", "On\_Easy", "On\_Engage"),  
 c("upgrade","satisfy","fatigue\_inverse")  
)

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 a1 alpha = 0.84, 95%CI[0.76, 0.9] S\_Review 0.7923222  
## 2 a2 alpha = 0.84, 95%CI[0.76, 0.9] S\_Add\_learn 0.7705651  
## 3 a3 alpha = 0.84, 95%CI[0.76, 0.9] S\_Feedback 0.7708075  
## 4 b1 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Place 0.8558135  
## 5 b2 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Time 0.7227740  
## 6 c1 alpha = 0.877, 95%CI[0.82, 0.92] On\_Satisfy 0.8238116  
## 7 c2 alpha = 0.877, 95%CI[0.82, 0.92] On\_Joy 0.8292648  
## 8 c3 alpha = 0.877, 95%CI[0.82, 0.92] On\_Easy 0.8653063  
## 9 c4 alpha = 0.877, 95%CI[0.82, 0.92] On\_Engage 0.8491109  
## 10 d1 alpha = 0.699, 95%CI[0.55, 0.8] upgrade 0.4263113  
## 11 d2 alpha = 0.699, 95%CI[0.55, 0.8] satisfy 0.4593506  
## 12 d3 alpha = 0.699, 95%CI[0.55, 0.8] fatigue\_inverse 0.8049024

물론 이함수에는 변수의 위치, 즉 열번호를 알면 열번호를 혼합해서 입력해도 같은 결과를 나타낸다.

# Multivariate data, and the column number and variable name are mixed  
bind\_alpha\_table( stat\_onl,   
 c("S\_Review", "S\_Add\_learn", "S\_Feedback"),  
 c("SE\_Place", "SE\_Time"),  
 c("On\_Satisfy", "On\_Joy", "On\_Easy", "On\_Engage"),  
 c(5:6, 21)  
)

## Var alpha\_95%CI subfactor cronbach alpha  
## 1 a1 alpha = 0.84, 95%CI[0.76, 0.9] S\_Review 0.7923222  
## 2 a2 alpha = 0.84, 95%CI[0.76, 0.9] S\_Add\_learn 0.7705651  
## 3 a3 alpha = 0.84, 95%CI[0.76, 0.9] S\_Feedback 0.7708075  
## 4 b1 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Place 0.8558135  
## 5 b2 alpha = 0.879, 95%CI[0.8, 0.93] SE\_Time 0.7227740  
## 6 c1 alpha = 0.877, 95%CI[0.82, 0.92] On\_Satisfy 0.8238116  
## 7 c2 alpha = 0.877, 95%CI[0.82, 0.92] On\_Joy 0.8292648  
## 8 c3 alpha = 0.877, 95%CI[0.82, 0.92] On\_Easy 0.8653063  
## 9 c4 alpha = 0.877, 95%CI[0.82, 0.92] On\_Engage 0.8491109  
## 10 d1 alpha = 0.699, 95%CI[0.55, 0.8] upgrade 0.4263113  
## 11 d2 alpha = 0.699, 95%CI[0.55, 0.8] satisfy 0.4593506  
## 12 d3 alpha = 0.699, 95%CI[0.55, 0.8] fatigue\_inverse 0.8049024

##|> knitr::kable(align="cccc", digits = 3, booktabs = TRUE,  
 # caption ="Table. 신뢰도 분석 결과")

신뢰도를 분석하는 과정은 R에서는 생각보다 간단하다. psych:alpha함수를 이용하여 세밀하게 확인하는 과정과 jjstat패키지의 alpha\_table과 bind\_alpha\_table을 이용하여 원하는 표를 빠르게 완성하는 것도 좋은 활용이 될 것이다.

다음의 표는 신뢰도 분석 결과의 일부를 편집한 것이다.

| Var | alpha\_95%CI | subfactor | cronbach alpha |
| --- | --- | --- | --- |
| a1 |  | S\_Review | 0.7923222 |
| a2 | alpha = 0.84, 95%CI[0.76, 0.9] | S\_Add\_learn | 0.7705651 |
| a3 |  | S\_Feedback | 0.7708075 |
| b1 | alpha = 0.879, 95%CI[0.8, 0.93] | SE\_Place | 0.8558135 |
| b2 |  | SE\_Time | 0.7227740 |
| c1 |  | On\_Satisfy | 0.8238116 |
| c2 | alpha = 0.877, 95%CI[0.82, 0.92] | On\_Joy | 0.8292648 |
| c3 |  | On\_Easy | 0.8653063 |
| c4 |  | On\_Engage | 0.8491109 |
| d1 |  | upgrade | 0.4263113 |
| d2 | alpha = 0.699, 95%CI[0.55, 0.8] | satisfy | 0.4593506 |
| d3 |  | fatigue\_inverse | 0.8049024 |

#### 3.2.1.2. 이분형 변수의 신뢰도 측정

측정한 항목들이 연속형 변수가 아니라 이분형변수 (0, 1)로 코딩된 변수일 경우에는 KR20(Kuder-Richardson formula 20)이라고 한다. 측정문항들이 이론적으로 위계적 관계를 가지는 경우 뢰빙거의 H(Loveinger’s coefficient H)를 사용한다(Mokken, 1971; Sijtsma & Molenaar, 2002; Sijtsma & Van der Ark, 2017).

mtcars의 am, vs을 위계적 데이터라고 가정하고, 이 데이터를 사용하여 분석해보자.

library(mokken)  
MTC <- mtcars #분석데이터  
MTC$am <- as.numeric(MTC$am ) # numeric변수로 변환   
MTC$vs <- as.numeric(MTC$vs ) # numeric변수로 변환  
  
# mokken::coefH(MTC[,c("am","vs")], results = FALSE)$H  
mokken::coefH(MTC[,c("am","vs")])

## $Hij  
## am se vs se   
## am 0.179 (0.191)  
## vs 0.179 (0.191)   
##   
## $Hi  
## Item H se   
## am 0.179 (0.191)  
## vs 0.179 (0.191)  
##   
## $H  
## Scale H se   
## 0.179 (0.191)

분석결과는 $H부분을 확인하면 된다. 결과값은 .179로 나타난 것을 확인할 수 있다.

### 3.2.2. 요인 분석(factor Analysis)

요인분석은 다변량 통계 기법 중 하나로, 관측된 변수들 간의 상관성을 고려하여 적은 수의 잠재변수 (요인)로 설명하는 방법이다. 이때, 요인분석의 접근방식에는 PCFA(principal component factor analysis)와 MLFA(maximum likelihood factor analysis)와 같은 2가지 방식이 존재한다.

요인분석에서 **MLFA (Maximum Likelihood Factor Analysis)**와 **PCFA (Principal Component Factor Analysis)**는 각각 다른 접근 방식을 가지고 있습니다. 이 두 방법의 장단점을 살펴보자

MLFA (Maximum Likelihood Factor Analysis):

\* 장점:  
 - 모델 기반: 최대우도법을 기반으로 하기 때문에 통계적으로 더 강력한 모델을 제공   
 - 정확한 추정: 데이터의 분포를 가정하고 모수를 추정하기 때문에 정확한 추정이 가능  
 - 변수 간 상관성 고려: 변수들 간의 상관성을 고려하여 요인을 추출  
   
\* 단점:  
 - 계산량: 최대우도법은 계산량이 많아서 큰 데이터셋에서 시간이 오래 걸릴 수 있다.   
 - 모델 가정: 데이터의 분포를 가정하고 모델을 설정해야 하므로 가정이 잘못된 경우 결과가 왜곡될 수 있다.

PCFA (Principal Component Factor Analysis):

\*장점:  
 - 빠른 계산: 주성분분석을 통해 차원을 축소하면서 요인분석을 수행하므로 계산량이 적다.   
 - 간단한 모델: 주성분분석은 간단한 선형 변환을 통해 요인을 추출하므로 모델 가정이 단순.  
  
\* 단점:  
 -상관성 무시: 주성분분석은 변수들 간의 상관성을 고려하지 않고 분산을 최대화하는 요인을 추출  
 - 모델 가정 부족: 데이터의 분포를 가정하지 않기 때문에 모델 가정이 부족할 수 있다.

요인분석을 선택할 때는 연구 목적, 데이터의 특성, 계산량, 모델 가정 등을 고려하여 적절한 방법을 선택해야한다.

#### 3.2.2.1. 주성분 요인분석(PCFA; Principal Component Factor Analysis)

PCFA (Principal Component Factor Analysis)는 PCFA는 주성분분석(PCA)과 요인분석을 결합한 방법이다. 주성분분석을 통해 원변수들을 선형 변환하여 새로운 변수들을 생성한 후, 이를 요인분석에 활용하는 방법으로 사용된다. 주성분분석은 원변수들의 분산을 최대화하는 새로운 변수들을 찾는 기법인데, PCFA는 데이터의 차원을 축소하면서 요인분석을 수행하므로, 변수들 간의 상관성을 고려하면서 데이터를 설명하는 잠재적인 요인들을 추출하게 된다. R에서는 psych 패키지의 principal() 함수를 사용하여 PCFA를 수행할 수 있다.

먼저 사용해볼 데이터는 MVT 패키지의 examScor데이터이다.

# install.package("MVN")  
library(MVT)  
data(examScor)  
examScor%>% str()

## 'data.frame': 88 obs. of 5 variables:  
## $ mechanics : num 77 63 75 55 63 53 51 59 62 64 ...  
## $ vectors : num 82 78 73 72 63 61 67 70 60 72 ...  
## $ algebra : num 67 80 71 63 65 72 65 68 58 60 ...  
## $ analysis : num 67 70 66 70 70 64 65 62 62 62 ...  
## $ statistics: num 81 81 81 68 63 73 68 56 70 45 ...

데이터의 일부를보면 다음과 같다.

examScor%>% head()

## mechanics vectors algebra analysis statistics  
## 1 77 82 67 67 81  
## 2 63 78 80 70 81  
## 3 75 73 71 66 81  
## 4 55 72 63 70 68  
## 5 63 63 65 70 63  
## 6 53 61 72 64 73

이제 데이텨를 통하여 표준화 한 후에 상관계수를 구해보자.  
표준화(scale)를 하는 이유는 각 과목의 난이도와 수준, 척도가 다르기 때문이다.

X = examScor #data  
Z = scale(X, scale=T) #표준화  
  
R = cor(X) # correlation matrix  
R

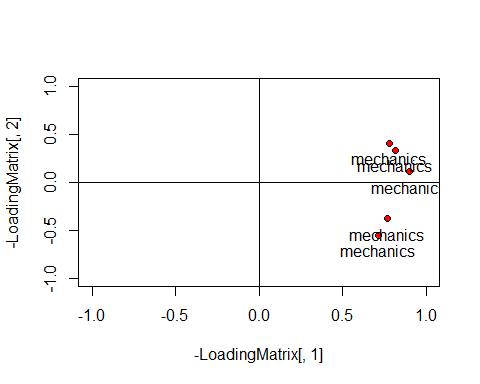
## mechanics vectors algebra analysis statistics  
## mechanics 1.0000000 0.5534052 0.5467511 0.4093920 0.3890993  
## vectors 0.5534052 1.0000000 0.6096447 0.4850813 0.4364487  
## algebra 0.5467511 0.6096447 1.0000000 0.7108059 0.6647357  
## analysis 0.4093920 0.4850813 0.7108059 1.0000000 0.6071743  
## statistics 0.3890993 0.4364487 0.6647357 0.6071743 1.0000000

psych::principal(R, nfactor = 2)

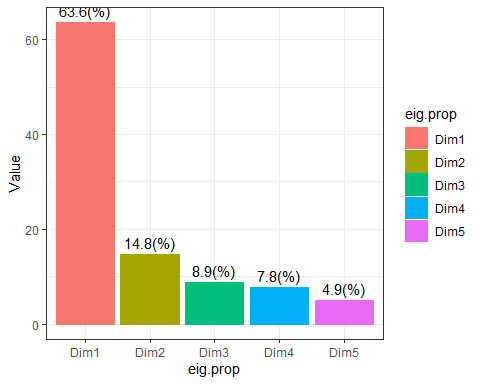
## Principal Components Analysis  
## Call: psych::principal(r = R, nfactors = 2)  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## RC1 RC2 h2 u2 com  
## mechanics 0.19 0.88 0.82 0.18 1.1  
## vectors 0.35 0.78 0.74 0.26 1.4  
## algebra 0.76 0.49 0.82 0.18 1.7  
## analysis 0.84 0.27 0.78 0.22 1.2  
## statistics 0.86 0.19 0.77 0.23 1.1  
##   
## RC1 RC2  
## SS loadings 2.18 1.74  
## Proportion Var 0.44 0.35  
## Cumulative Var 0.44 0.78  
## Proportion Explained 0.56 0.44  
## Cumulative Proportion 0.56 1.00  
##   
## Mean item complexity = 1.3  
## Test of the hypothesis that 2 components are sufficient.  
##   
## The root mean square of the residuals (RMSR) is 0.09   
##   
## Fit based upon off diagonal values = 0.97

jjstat패키지의 pcfa함수를 통해서 분석을 해볼 수 있다.

jjstat::pcfa(R, nfactor = 2)



## $data  
## mechanics vectors algebra analysis statistics  
## mechanics 1.0000000 0.5534052 0.5467511 0.4093920 0.3890993  
## vectors 0.5534052 1.0000000 0.6096447 0.4850813 0.4364487  
## algebra 0.5467511 0.6096447 1.0000000 0.7108059 0.6647357  
## analysis 0.4093920 0.4850813 0.7108059 1.0000000 0.6071743  
## statistics 0.3890993 0.4364487 0.6647357 0.6071743 1.0000000  
##   
## $propDim  
## eig.prop Value  
## 1 Dim1 63.619603  
## 2 Dim2 14.791437  
## 3 Dim3 8.899303  
## 4 Dim4 7.757848  
## 5 Dim5 4.931810  
##   
## $communality  
## mechanics vectors algebra analysis statistics  
## mechanics 0.8160687 0.7591497 0.5781263 0.3957286 0.3324899  
## vectors 0.7591497 0.7362142 0.6485274 0.5004880 0.4478032  
## algebra 0.5781263 0.6485274 0.8180380 0.7686419 0.7465051  
## analysis 0.3957286 0.5004880 0.7686419 0.7756105 0.7720366  
## statistics 0.3324899 0.4478032 0.7465051 0.7720366 0.7746206  
##   
## $specific\_variance  
## mechanics vectors algebra analysis statistics   
## 0.1839313 0.2637858 0.1819620 0.2243895 0.2253794   
##   
## $residual\_matrix  
## mechanics vectors algebra analysis statistics  
## mechanics 0.00000000 -0.20574447 -0.03137518 0.01366338 0.05660943  
## vectors -0.20574447 0.00000000 -0.03888276 -0.01540672 -0.01135454  
## algebra -0.03137518 -0.03888276 0.00000000 -0.05783606 -0.08176935  
## analysis 0.01366338 -0.01540672 -0.05783606 0.00000000 -0.16486231  
## statistics 0.05660943 -0.01135454 -0.08176935 -0.16486231 0.00000000  
##   
## $factorloadings  
## PC1 PC2  
## mechanics 0.7127067 -0.5550836  
## vectors 0.7694490 -0.3796873  
## algebra 0.8976179 0.1109959  
## analysis 0.8150623 0.3335926  
## statistics 0.7816221 0.4045832  
##   
## $prop\_graph



#### 3.2.2.1. 최대우도 요인분석(MLFA : Maximum Likelihood Factor Analysis)

MLFA는 최대우도법을 기반으로 하는 요인분석 방법이다. 최대우도법은 주어진 데이터를 가장 잘 설명하는 모수를 추정하는 기법으로, 요인분석에서는 잠재변수들의 모수를 추정한다. 또한 MLFA는 데이터의 분포를 가정하고 최대우도법을 적용하여 요인들을 추정하는 방법을 사용한다. factanal()함수를 사용하거나 psych 패키지의 fa()를 사용할 수 있다. 특히, fa()함수는 PCFA와 MLFA를 모두 지원하며, 요인의 수, 회전 방법, 요인부하량 등을 설정하여 요인분석을 수행할 수 있다.

#factor analysis tools : factanal  
mlfa <- factanal(Z, factors = 2, rotation = "varimax")  
mlfa

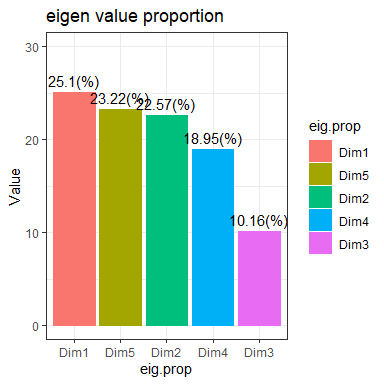
##   
## Call:  
## factanal(x = Z, factors = 2, rotation = "varimax")  
##   
## Uniquenesses:  
## mechanics vectors algebra analysis statistics   
## 0.466 0.419 0.189 0.352 0.431   
##   
## Loadings:  
## Factor1 Factor2  
## mechanics 0.265 0.681   
## vectors 0.356 0.674   
## algebra 0.740 0.514   
## analysis 0.738 0.322   
## statistics 0.696 0.290   
##   
## Factor1 Factor2  
## SS loadings 1.774 1.370  
## Proportion Var 0.355 0.274  
## Cumulative Var 0.355 0.629  
##   
## Test of the hypothesis that 2 factors are sufficient.  
## The chi square statistic is 0.07 on 1 degree of freedom.  
## The p-value is 0.785

mlfa$loadings

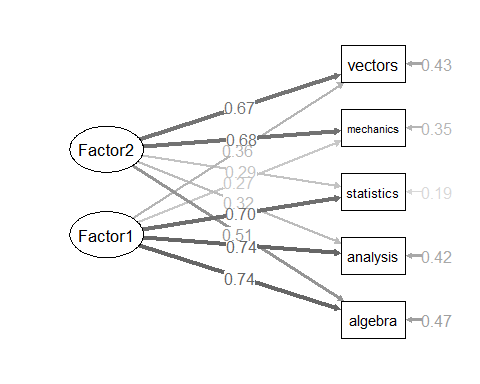
##   
## Loadings:  
## Factor1 Factor2  
## mechanics 0.265 0.681   
## vectors 0.356 0.674   
## algebra 0.740 0.514   
## analysis 0.738 0.322   
## statistics 0.696 0.290   
##   
## Factor1 Factor2  
## SS loadings 1.774 1.370  
## Proportion Var 0.355 0.274  
## Cumulative Var 0.355 0.629

mlfa |>   
 jjstat::pcfa\_porp\_plot(prop = F, ymax = 30)

## [[1]]  
## mechanics vectors algebra analysis statistics   
## 0.4658969 0.4190553 0.1885694 0.3517932 0.4310210   
##   
## [[2]]  
## eig.prop Value  
## mechanics Dim1 25.09766  
## vectors Dim2 22.57432  
## algebra Dim3 10.15815  
## analysis Dim4 18.95095  
## statistics Dim5 23.21892  
##   
## [[3]]



# library(semPlot)  
#factanal diagram  
semPlot::semPaths(mlfa, what="est", posCol="gray40",fade=T,  
rotation = 2,style="lisrel",  
sizeLat = 16, sizeLat2 = 10,  
sizeMan = 14, sizeMan2 = 8,  
nCharNodes=10, edge.label.cex=1.5)



# library(psych)  
# library(MVT)  
library(GPArotation)  
  
data(examScor)  
 X = examScor  
 psych::fa(X, nfactor=2)

## Factor Analysis using method = minres  
## Call: psych::fa(r = X, nfactors = 2)  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## MR1 MR2 h2 u2 com  
## mechanics -0.03 0.75 0.53 0.47 1.0  
## vectors 0.09 0.69 0.58 0.42 1.0  
## algebra 0.73 0.22 0.81 0.19 1.2  
## analysis 0.84 -0.05 0.65 0.35 1.0  
## statistics 0.80 -0.06 0.57 0.43 1.0  
##   
## MR1 MR2  
## SS loadings 1.96 1.18  
## Proportion Var 0.39 0.24  
## Cumulative Var 0.39 0.63  
## Proportion Explained 0.62 0.38  
## Cumulative Proportion 0.62 1.00  
##   
## With factor correlations of   
## MR1 MR2  
## MR1 1.00 0.75  
## MR2 0.75 1.00  
##   
## Mean item complexity = 1  
## Test of the hypothesis that 2 factors are sufficient.  
##   
## df null model = 10 with the objective function = 2.3 with Chi Square = 194.46  
## df of the model are 1 and the objective function was 0   
##   
## The root mean square of the residuals (RMSR) is 0   
## The df corrected root mean square of the residuals is 0.01   
##   
## The harmonic n.obs is 88 with the empirical chi square 0.03 with prob < 0.87   
## The total n.obs was 88 with Likelihood Chi Square = 0.08 with prob < 0.78   
##   
## Tucker Lewis Index of factoring reliability = 1.051  
## RMSEA index = 0 and the 90 % confidence intervals are 0 0.186  
## BIC = -4.4  
## Fit based upon off diagonal values = 1  
## Measures of factor score adequacy   
## MR1 MR2  
## Correlation of (regression) scores with factors 0.94 0.89  
## Multiple R square of scores with factors 0.88 0.79  
## Minimum correlation of possible factor scores 0.76 0.58

#### 3.2.2.2. 탐색적 요인 분석

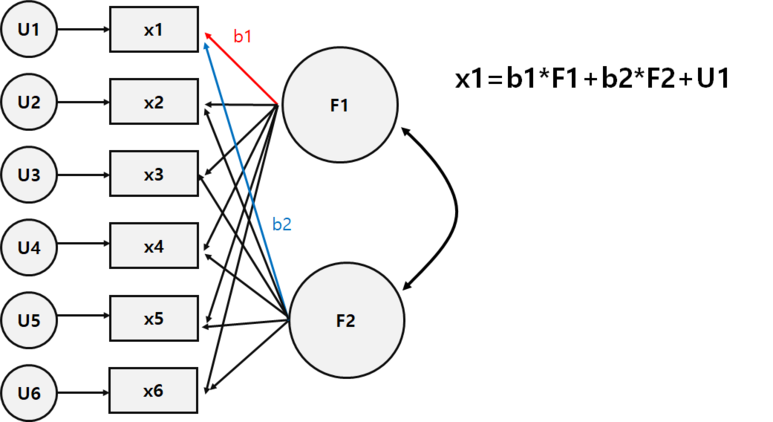


그림 요인분석

# install.packages("ade4")  
library(ade4)  
data(olympic)  
str(olympic)

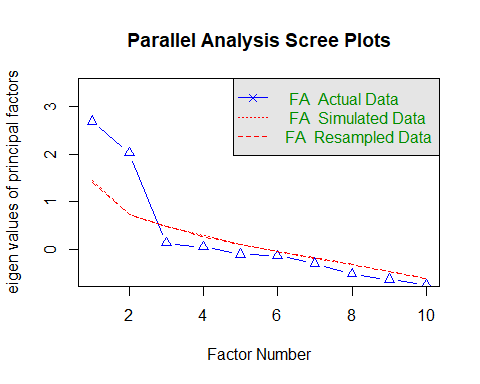
## List of 2  
## $ tab :'data.frame': 33 obs. of 10 variables:  
## ..$ 100 : num [1:33] 11.2 10.9 11.2 10.6 11 ...  
## ..$ long: num [1:33] 7.43 7.45 7.44 7.38 7.43 7.72 7.05 6.95 7.12 7.28 ...  
## ..$ poid: num [1:33] 15.5 15 14.2 15 12.9 ...  
## ..$ haut: num [1:33] 2.27 1.97 1.97 2.03 1.97 2.12 2.06 2 2.03 1.97 ...  
## ..$ 400 : num [1:33] 48.9 47.7 48.3 49.1 47.4 ...  
## ..$ 110 : num [1:33] 15.1 14.5 14.8 14.7 14.4 ...  
## ..$ disq: num [1:33] 49.3 44.4 43.7 44.8 41.2 ...  
## ..$ perc: num [1:33] 4.7 5.1 5.2 4.9 5.2 4.9 5.7 4.8 4.9 5.2 ...  
## ..$ jave: num [1:33] 61.3 61.8 64.2 64 57.5 ...  
## ..$ 1500: num [1:33] 269 273 263 285 257 ...  
## $ score: num [1:33] 8488 8399 8328 8306 8286 ...

**요인이 개수를 찾는 방법**

요인분석에서 중요한 것은 추출할 요인의 개수를 정하는 것이 가장 중요한데 이는 선행연구를 통해서 밝혀내는 것이 원칙이다. 밝혀낸다는 의미는 연구자가 미리 이론에 대한 연구, 선행연구 문헌들을 고찰하여 자신의 연구에 맞게 이론을 선정하고 연구모형을 만들고, 요인을 설정해야 한다. 그럼에도 불구하고 통계적으로 요인을 검토하는 방법들은 존재한다.  
요인개수를 선정하는 방법중 가장 유명하고 가장 많이 활용하는 것은 고유값(eigenvalue)이 1보다 큰 것을 찾아내는 카이저규칙이다. 분산이 1보다 작은 것은 변수 자신도 설명하지 못하는 것이기 때문이다.

1. 스크리 도표를 이용하는 방법  
   fm= ml: maximum likeihood = 최대우도법  
   실제 고유값을 바탕으로 하는 스크리테스트(scree test)결과는 실선으로 그려졌고, 가상의 무작위 데이터를 기반으로 계산한 고유값 평균은 점선으로 나타냈다. 그림의 결과를 보면 요인은 2개가 적달하다.

psych::fa.parallel(olympic$tab, fm="ml", fa="fa", n.iter=100)



## Parallel analysis suggests that the number of factors = 2 and the number of components = NA

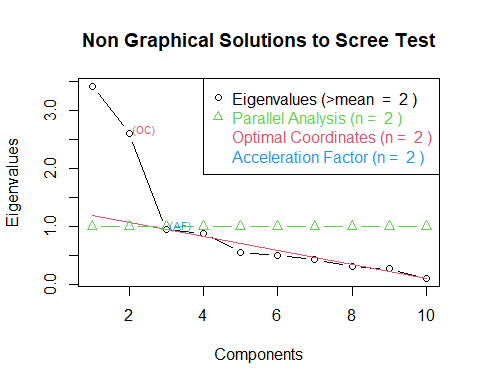
1. nFactors 패키지의 nScree()를 이용하여 적정 요인 개수를 추천받기

library(nFactors)  
nFactors::nScree(olympic$tab)

## noc naf nparallel nkaiser  
## 1 2 2 2 2

4개의 방법 모두 2개가 적절하다고 추천하고 있다. 연구자가 요인을 얼마로 설정해야 할지 판단이 서지 않을때 활용하면 효과적이다.

plotnScree(nScree(olympic$tab))



1. eigen value를 이용하는 방법

먼저 cor()을 이용하여 상관행렬 만든 후에 eigen()을 이용하여 분해한 값중에서 1보다 큰 값을 찾는 방법이다. 현재 데이터에서는 3.4182381 2.6063931 0.9432964, ….으로 가고 있으므로 1보다 큰 값은 2개이며, 이 데이터에 적합한 요인은 2개라고 판단할 수있다.

eigen(cor(olympic$tab))

## eigen() decomposition  
## $values  
## [1] 3.4182381 2.6063931 0.9432964 0.8780212 0.5566267 0.4912275 0.4305952  
## [8] 0.3067981 0.2669494 0.1018542  
##   
## $vectors  
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0.4158823 0.1488081 -0.26747198 0.08833244 0.442314456 0.03071237  
## [2,] -0.3940515 -0.1520815 -0.16894945 0.24424963 -0.368913901 -0.09378242  
## [3,] -0.2691057 0.4835374 0.09853273 0.10776276 0.009754680 0.23002054  
## [4,] -0.2122818 0.0278985 -0.85498656 -0.38794393 0.001876311 0.07454380  
## [5,] 0.3558474 0.3521598 -0.18949642 -0.08057457 -0.146965351 -0.32692886  
## [6,] 0.4334816 0.0695682 -0.12616012 0.38229029 0.088802794 0.21049130  
## [7,] -0.1757923 0.5033347 0.04609969 -0.02558404 -0.019358607 0.61491241  
## [8,] -0.3840821 0.1495820 0.13687235 -0.14396548 0.716743474 -0.34776037  
## [9,] -0.1799436 0.3719570 -0.19232803 0.60046566 -0.095582043 -0.43744387  
## [10,] 0.1701426 0.4209653 0.22255233 -0.48564231 -0.339772188 -0.30032419  
## [,7] [,8] [,9] [,10]  
## [1,] 0.2543985 0.663712826 -0.10839531 -0.10948045  
## [2,] 0.7505343 0.141264141 0.04613910 -0.05580431  
## [3,] -0.1106637 0.072505560 0.42247611 -0.65073655  
## [4,] -0.1351242 -0.155435871 -0.10206505 -0.11941181  
## [5,] 0.1413388 -0.146839303 0.65076229 0.33681395  
## [6,] 0.2725296 -0.639003579 -0.20723854 -0.25971800  
## [7,] 0.1439726 0.009400445 -0.16724055 0.53450315  
## [8,] 0.2732665 -0.276873049 -0.01766443 0.06589572  
## [9,] -0.3419099 0.058519366 -0.30619617 0.13093187  
## [10,] 0.1868704 0.007310045 -0.45688227 -0.24311846

**factanal을 이용한 요인분석**

factanal(Perform maximum-likelihood factor analysis on a covariance matrix or data matrix) 함수는 R에서 데이터의 상관 구조를 고려하여 적은 수의 잠재변수(요인)로 설명하는 기법으로의 요인분석을 수행하는 함수이다. 이 함수는 최대우도법을 기반으로 하여 공분산 행렬이나 데이터 행렬에서 요인분석을 수행한다.

다음은 factanal 함수를 사용할 때 알아야 하는 주요 인자이다.

factanal(x, factors, data = NULL, covmat = NULL, n.obs = NA,  
 subset, na.action, start = NULL,  
 scores = c("none", "regression", "Bartlett"),  
 rotation = "varimax", control = NULL, ...)

* x는 공식 또는 숫자 행렬 또는 숫자 행렬로 변환 가능한 객체이다.
* factors: 추출할 요인의 수를 지정한다.
* data: 데이터 프레임 또는 유사한 객체로, x가 공식일 경우에만 사용된다.
* covmat: 공분산 행렬 또는 cov.wt 함수로 반환된 공분산 리스트이다.

fa <- factanal(olympic$tab, factors=2, scores="regression")  
fa

##   
## Call:  
## factanal(x = olympic$tab, factors = 2, scores = "regression")  
##   
## Uniquenesses:  
## 100 long poid haut 400 110 disq perc jave 1500   
## 0.396 0.547 0.022 0.933 0.287 0.420 0.318 0.575 0.638 0.631   
##   
## Loadings:  
## Factor1 Factor2  
## 100 0.772   
## long -0.672   
## poid -0.155 0.977   
## haut -0.241   
## 400 0.814 0.226   
## 110 0.739 -0.185   
## disq 0.826   
## perc -0.505 0.413   
## jave 0.600   
## 1500 0.488 0.362   
##   
## Factor1 Factor2  
## SS loadings 2.834 2.401  
## Proportion Var 0.283 0.240  
## Cumulative Var 0.283 0.523  
##   
## Test of the hypothesis that 2 factors are sufficient.  
## The chi square statistic is 19.12 on 26 degrees of freedom.  
## The p-value is 0.831

요인을 회전시킨 결과

fa.p <- factanal(olympic$tab, factors=2, scores="regression", rotation ="promax")  
fa.p

##   
## Call:  
## factanal(x = olympic$tab, factors = 2, scores = "regression", rotation = "promax")  
##   
## Uniquenesses:  
## 100 long poid haut 400 110 disq perc jave 1500   
## 0.396 0.547 0.022 0.933 0.287 0.420 0.318 0.575 0.638 0.631   
##   
## Loadings:  
## Factor1 Factor2  
## 100 0.762   
## long -0.668   
## poid 0.982   
## haut -0.230   
## 400 0.840 0.232   
## 110 0.719 -0.182   
## disq 0.100 0.832   
## perc -0.458 0.413   
## jave 0.603   
## 1500 0.529 0.367   
##   
## Factor1 Factor2  
## SS loadings 2.805 2.431  
## Proportion Var 0.281 0.243  
## Cumulative Var 0.281 0.524  
##   
## Factor Correlations:  
## Factor1 Factor2  
## Factor1 1.000 0.118  
## Factor2 0.118 1.000  
##   
## Test of the hypothesis that 2 factors are sufficient.  
## The chi square statistic is 19.12 on 26 degrees of freedom.  
## The p-value is 0.831

직접 oblimin으로 사각회전한 값을 알고 싶다면 GPArotation을 이용해서 회전시켜야 한다. factanal에는 회전하는 기능이 없기 때문이다.

library(GPArotation)  
fa.o <- factanal(olympic$tab, factors=2, scores="regression", rotation = "oblimin")  
fa.o

##   
## Call:  
## factanal(x = olympic$tab, factors = 2, scores = "regression", rotation = "oblimin")  
##   
## Uniquenesses:  
## 100 long poid haut 400 110 disq perc jave 1500   
## 0.396 0.547 0.022 0.933 0.287 0.420 0.318 0.575 0.638 0.631   
##   
## Loadings:  
## Factor1 Factor2  
## 100 0.761 -0.108   
## long -0.667   
## poid 0.983   
## haut -0.231 0.101   
## 400 0.835 0.207   
## 110 0.719 -0.203   
## disq 0.828   
## perc -0.461 0.426   
## jave 0.602   
## 1500 0.524 0.351   
##   
## Factor1 Factor2  
## SS loadings 2.790 2.429  
## Proportion Var 0.279 0.243  
## Cumulative Var 0.279 0.522  
##   
## Factor Correlations:  
## Factor1 Factor2  
## Factor1 1.0000 0.0784  
## Factor2 0.0784 1.0000  
##   
## Test of the hypothesis that 2 factors are sufficient.  
## The chi square statistic is 19.12 on 26 degrees of freedom.  
## The p-value is 0.831

‘Loadings’ 이란 무엇인가?  
‘Loadings’ is a term from factor analysis, but because factor analysis and principal component analysis (PCA) are often conflated in the social science literature, it was used for PCA by SPSS and hence by princomp in S-PLUS to help SPSS users.  
Small loadings are conventionally not printed (replaced by spaces), to draw the eye to the pattern of the larger loadings. The print method for class “factanal” calls the “loadings” method to print the loadings, and so passes down arguments such as cutoff and sort.  
The signs of the loadings vectors are arbitrary for both factor analysis and PCA  
이 값은 요인적재값이라고 하는데, 요인적재값은 0.1보다 작으면 출력되지 않는다.

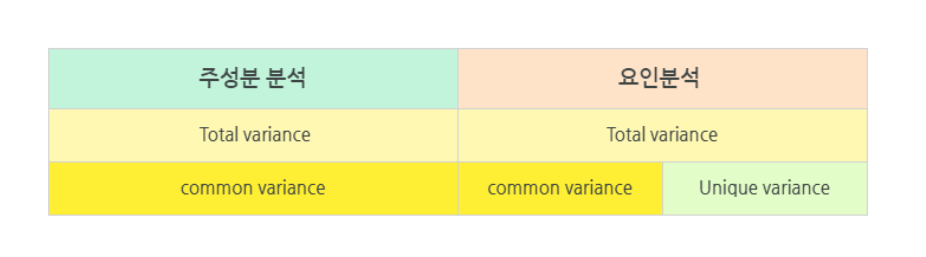
print(fa$loadings, cutoff=0.001)

##   
## Loadings:  
## Factor1 Factor2  
## 100 0.772 -0.089   
## long -0.672 0.041   
## poid -0.155 0.977   
## haut -0.241 0.095   
## 400 0.814 0.226   
## 110 0.739 -0.185   
## disq 0.007 0.826   
## perc -0.505 0.413   
## jave -0.049 0.600   
## 1500 0.488 0.362   
##   
## Factor1 Factor2  
## SS loadings 2.834 2.401  
## Proportion Var 0.283 0.240  
## Cumulative Var 0.283 0.523

Loadings:  
Factor1 Factor2  
100 0.772 -0.089

이것을 해석하는 방법은 아래와 같다. Factor1과 Factor2의 선형결합으로 표현이 된다.

100이라는 변수의 분산은 Factor1에 의해서는 0.772의제곱인 59.6%가 설명이 되고 Factor2에 의해서는 (-.089)^2=0.8%가 설명된다는 의미이다. 통계는 분산의 마법과 같은 것이므로. 모든 것이 분산으로 설명된다고 생각하면 된다. 여기에서 두개를 합흔 60.4%가 두 요인에 의해 설명되는 변수100의 분산비율이다. 이것을 변수 100의 공통성(communality)라고 부른다. 공통성(communality)이라는 것은 공통요인에 의해서 섦여되는 분산의 비율이다. 값은 0~1사이의 값을 갖게 되는데, 공통성이 1이면 변수의 분산을 100%를 설명한다. 이 공통요인으로 설명이 안되는 것은 변수의 고유요인에 의해서 설명되는 것으로 가정하게 된다. 요인분석에서는 총분산에는 공통분산과 고유분산으로 나누어서 분석을 하기 때문이다. 고유요인에 의해서 설명되는 분산은 uniquenesses들어있다. 아래 그림을 참고해보자.



print(fa$loadings, cutoff=0.5, sort = TRUE)

##   
## Loadings:  
## Factor1 Factor2  
## 100 0.772   
## long -0.672   
## 400 0.814   
## 110 0.739   
## perc -0.505   
## poid 0.977   
## disq 0.826   
## jave 0.600   
## haut   
## 1500   
##   
## Factor1 Factor2  
## SS loadings 2.834 2.401  
## Proportion Var 0.283 0.240  
## Cumulative Var 0.283 0.523

공통분산= 1 - 고유 분산

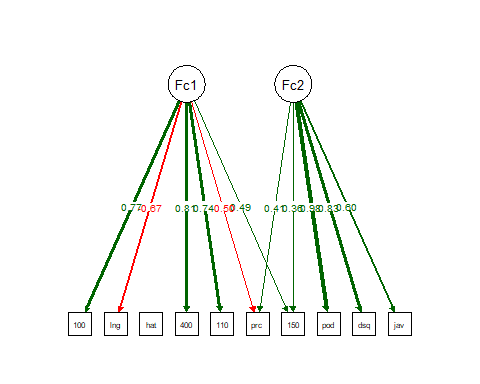
round(1 - fa$uniquenesses, 3)

## 100 long poid haut 400 110 disq perc jave 1500   
## 0.604 0.453 0.978 0.067 0.713 0.580 0.682 0.425 0.362 0.369

요인분석의 시각화

# factor.plot(fa, labels=colnames(olympic$tab), pch=20, pos=4, title="Factor Plot")

library(semPlot)  
semPaths(fa, what="est", residuals=FALSE,  
 cut=0.3, posCol=c("white", "darkgreen"), negCol=c("white", "red"),  
 edge.label.cex = 1, title=TRUE)



# 4. 차이분석

## 4.1. t-test

t-test는 두 그룹의 평균을 비교하는 통계적 검정 방법이다. 주로 두 그룹 간의 차이가 우연히 발생한 것인지, 실제로 효과가 있는지를 확인하는데 사용한다.

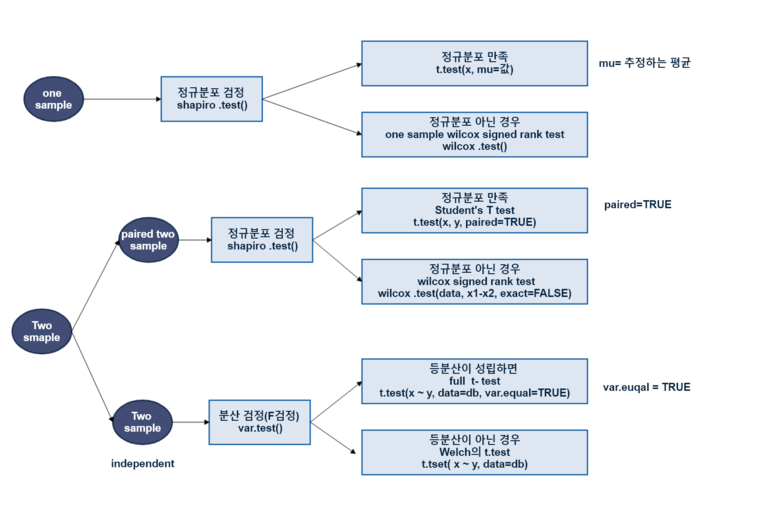


fig. t-test process

### 4.1.1. 일표본 t검정(onesample t.test)

하나의 표본을 이용하여 모집단의 특정한 값과 같은지 추론하는 방법이다. 만약에 음료수를 사먹는데, 평균용량이 1L라고 할 때, 진짜 평균적으로 1L인지 샘플을 추출하여 평균을 구하여 일표본 t검정을 수행하면 된다.

검정에 필요한 t값은 다음과 같이 계산한다.

는 평균, :샘플사이즈, :샘플의 표준편차

결과를 해석할 때에는 p값을 기준하여 설명하면 된다. 이면 귀무가설을 기각하고 대립가설을 받아들인다. 즉, 표본평균은 이론적 평균과 유의하게 다르다는 결론을 내릴 수 있다.

예) 학생들의 기말고사 성적은 95%의 신뢰수준에서 볼때 24점보다 유의하게 높은 가?  
귀무가설: 학생들의 성적은 24점과 같다.

대립가설: 학생들의 성적은 24점보다 높다

*사용하는 방법*

t.test(관측치, alternative = 판별 방향, mu=특정기준, conf.level = 신뢰수준)

#단일표본 t test  
final <- c(19, 22, 24, 24, 25, 25, 26, 26, 28, 32)  
shapiro.test(final)

##   
## Shapiro-Wilk normality test  
##   
## data: final  
## W = 0.95546, p-value = 0.7332

t.test(final, mu=24, alternative="greater", conf.level = .95)

##   
## One Sample t-test  
##   
## data: final  
## t = 1.0093, df = 9, p-value = 0.1696  
## alternative hypothesis: true mean is greater than 24  
## 95 percent confidence interval:  
## 23.10218 Inf  
## sample estimates:  
## mean of x   
## 25.1

결과를 요약하면 다음과 같다. 정규성 검정(Shapiro-Wilk normality test)결과 정규성은 확보되었다( = 0.96, = 0.7332)

The One Sample t-test testing the difference between final (mean = 25.10) and mu = 24 suggests that the effect is positive, statistically not significant,and small (difference = 1.10, 95% CI [23.10, Inf], t(9) = 1.01, p = 0.170.

일표본 t검정 결과 중간고사 성적은 24점보다 높다고 말할 수 없다( (9) = 1.01, = 0.170).

t.test(final, mu=24, alternative="greater", conf.level = .95) |>  
 report\_stat("t.test", trans = FALSE)

## [1] "One Sample t-test showed that It was not statistically significant. t = 1.009, p = 0.17)"

### 4.1.2. 독립표본 t검정

* 두변수간 평균 비교
* 두 집단간 평균 비교

#### 4.1.2.1. 등분산성 검정

영가설: 등분산을 이루고 있다.   
대립가설: 등분산을 이루지 않는다

등분산인 경우와 등분산이 아닌 경우에 사용하는 t값은 다르게 사용된다.

로 사용한다.이때 공통분산(pooled variance)는

이다.

만약에 등분산이 아니라면 t값은 다음과 같이 사용한다.

자유도가 많이 복잡한 상태로 변경되어 사용된다.

###### (1) var.test()

var.test는 F검정(F Test to Compare Two Variances )이라고도 한다.  
Performs an F test to compare the variances of two samples from normal populations.

var.test(mpg ~ vs, data = mtcars)

##   
## F test to compare two variances  
##   
## data: mpg by vs  
## F = 0.51515, num df = 17, denom df = 13, p-value = 0.1997  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.1714935 1.4353527  
## sample estimates:  
## ratio of variances   
## 0.5151485

F검정결과, 현재의 vs데이터에 대한 등분산 검정은 p값이 0.05보다 크므로, 귀무가설을 기각할 수 없다. 즉, 등분산을 가정할 수 있다. 만약에 이 결과를 표로 만들어서 논문에 넣으려고 한다면 다음과 같이 tidy()함수를 이용하여 정리된 형태를 저장하여 편집하면 된다.

var.test(mpg ~ vs, data = mtcars) |>  
 broom::tidy()

## # A tibble: 1 × 9  
## estimate num.df den.df statistic p.value conf.low conf.high method alternative  
## <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <chr> <chr>   
## 1 0.515 17 13 0.515 0.200 0.171 1.44 F tes… two.sided

jjstat패키지에서 var.test에 대한 분석결과를 간략하게 해석해줄 수 있다.

var.test(mpg ~ vs, data = mtcars)|>  
 jjstat::report\_stat("var.test", trans= TRUE, show="data")

##   
##   
## 두 변량을 비교하기 위한 F검정 결과 통계적으로 유의하지 않은 것으로 나타났으며, 귀무가설을 채택하여 동일한 분산을 가정할 수 있다. F = 0.515, p = 0.2)

###### (2)bartlett.test

bartlett.test는 모수적 등분산 검정 방법이다. 비모수적 방법으로는fliger.test()를 사용한다.  
Bartlett 검정은 두 개 이상의 모집단의 분산이 동일한지 검정하는 모수적 검정 방법이다. 일반적으로 ANOVA 등 분산 분석을 수행하기 전에 가정하는 등분산성을 검증하는 데 사용된다. Bartlett검정의 일반적인 가정은 각 모집단은 정규 분포를 따른다는 것과 각 모집단의 변수는 서로 독립적이라는 사실이다.

Bartlett 검정의 검정 통계량은 다음과 같이 계산된다.

: 전체 표본 크기  
: i번째 그룹의 표본 크기  
: 전체 표본의 분산  
: i번째 그룹의 분산

Bartlett 검정의 가설은 다음과 같다.  
- 귀무가설 (): 모든 모집단의 분산은 동일합니다. - 대립가설 (): 모든 모집단의 분산은 동일하지 않습니다.

Bartlett 검정의 p-값은 검정 통계량 T가 분포에서 자유도를 가지고 발생할 확률이다. p 값이 유의수준 ()보다 작으면 귀무가설을 기각하고 대립가설을 채택한다.  
표본 크기가 작거나 모집단의 분포가 정규 분포를 따르지 않는 경우 Bartlett 검정의 결과가 정확하지 않을 수 있다.

bartlett.test(mpg~vs,mtcars)

##   
## Bartlett test of homogeneity of variances  
##   
## data: mpg by vs  
## Bartlett's K-squared = 1.5849, df = 1, p-value = 0.2081

bartlett.test결과, p값이 0.05보다 커서 귀무가설을 기각할 수 없다. 따라서, 두 변수는 등분산이라고 할 수 있다.

###### (3) leveneTest()

car 패키지에는 Computes Levene’s test for homogeneity of variance across groups라고 설명되어 있다. 르벤테스트는 SPSS에서 사용하고 있어서 유명한 등분산 검정이다. Levene 검정은 두 개 이상의 모집단의 분산이 동일한지 검정하는 비모수적 검정 방법이다. ANOVA 등 분산 분석을 수행하기 전에 가정하는 등분산성을 검증하는 데 사용된다. Bartlett 검정과 달리 모집단의 분포에 대한 가정을 하지 않아 더 넓은 범위의 상황에서 사용할 수 있다.

Levene 검정의 검정 통계량은 다음과 같이 계산됩니다.

: 전체 표본 크기 : 그룹 수 : i번째 그룹의 표본 크기 : i번째 그룹의 평균 : i번째 그룹의 j번째 데이터 : 전체 표본의 평균

Levene의 가설  
귀무가설 (): 모든 모집단의 분산은 동일합니다.  
대립가설 (): 모든 모집단의 분산은 동일하지 않습니다.

Levene 검정의 p-값은 검정 통계량 W가 F 분포에서 , 자유도를 가지고 발생할 확률이다. p값이 유의수준 ()보다 작으면 귀무가설을 기각하고 대립가설을 채택한다.

mtcars에서 vs에 따른 mpg의 등분산 검정

car::leveneTest(mpg ~ factor(vs),mtcars)

## Levene's Test for Homogeneity of Variance (center = median)  
## Df F value Pr(>F)  
## group 1 1.5922 0.2167  
## 30

##### (4) 비모수검정Fligner-Killeen 검정 (Fligner-Killeen test)

Fligner-Killeen 검정은 두 개 이상의 모집단의 분산이 동일한지 검정하는 비모수적 검정 방법이다. Levene 검정과 유사하지만, 순위 통계를 사용하여 검정 통계량을 계산하기 때문에 더 정확한 결과를 제공할 수 있다. 특히, 표본 크기가 작거나 모집단의 분포가 정규 분포를 따르지 않는 경우 유용한 방법이다. 비모수 검정을 사용하는 경우 chi-square(카이제곱)으로 통계치가 나타난다.

Fligner-Killeen 검정의 검정 통계량은 다음과 같이 계산된다.

: 전체 표본 크기 : 그룹 수 : i번째 그룹의 표본 크기 : i번째 그룹의 j번째 데이터의 순위

귀무가설 (): 모든 모집단의 분산은 동일합니다.  
대립가설 (): 모든 모집단의 분산은 동일하지 않습니다.

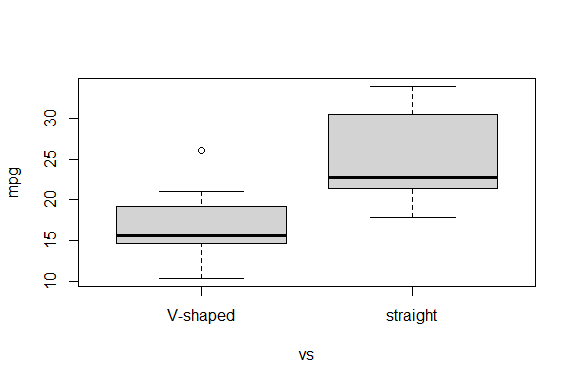
Fligner-Killeen 검정의 p-값은 검정 통계량 F가 χ^2 분포에서 자유도를 가지고 발생할 확률이다. p값이 유의수준 ()보다 작으면 귀무가설을 기각하고 대립가설을 채택한다.  
Fligner-Killeen 검정은 Levene 검정과 유사하지만, 순위 통계를 사용하기 때문에 더 정확한 결과를 제공할 수 있다. 표본 크기가 작거나 모집단의 분포가 정규 분포를 따르지 않는 경우 Fligner-Killeen 검정을 사용하는 것이 좋다.

fligner.test(mpg ~ vs,data= mtcars)

##   
## Fligner-Killeen test of homogeneity of variances  
##   
## data: mpg by vs  
## Fligner-Killeen:med chi-squared = 1.1765, df = 1, p-value = 0.2781

두변수의 차이를 시각화해보자. Straigth가 V-shaped보다 높은 값을 나타내고 있다.

plot(mpg ~ vs, data= mtcars)



### 4.1.3. 독립표본 t검정

등분산 검정(동질성 검정)결과, vs의 두 개의 레벨은 등분산이므로 var.equal = TRUE로 해서 분석을 한다.

t.test(mpg ~ vs, data= mtcars, var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: mpg by vs  
## t = -4.8644, df = 30, p-value = 3.416e-05  
## alternative hypothesis: true difference in means between group V-shaped and group straight is not equal to 0  
## 95 percent confidence interval:  
## -11.274221 -4.606732  
## sample estimates:  
## mean in group V-shaped mean in group straight   
## 16.61667 24.55714

보고서 작성

t.test(mpg ~ vs, data= mtcars, var.equal = TRUE) %>%   
 jjstat::report\_ttest( trans =TRUE)

##   
##   
## 종속변수 'mpg'에 대한 독립변수 'vs'의 두 집단 'gV-shaped(16.62)'와 'gstraight(24.56)'간의 두 표본 t-test 결과는 통계적으로 유의한 것으로 나타났다( t(30) = -4.86, p < .001).   
##   
##   
##

##### 4.1.3.1. 등분산이 아닌 경우

등분산이 아닌 경우에는 var.equal = FALSE로 하게 되는데 이 경우에는 Two sample t-test가 아닌 Whelch Two t-test를 수행한다. 자유도가 바뀌어서 나타나게 된다.

t.test(mpg ~ vs, data= mtcars, var.equal = FALSE)

##   
## Welch Two Sample t-test  
##   
## data: mpg by vs  
## t = -4.6671, df = 22.716, p-value = 0.0001098  
## alternative hypothesis: true difference in means between group V-shaped and group straight is not equal to 0  
## 95 percent confidence interval:  
## -11.462508 -4.418445  
## sample estimates:  
## mean in group V-shaped mean in group straight   
## 16.61667 24.55714

다음은 jjstat패키지의 report\_ttest를 이용하여 보고서를 작성한 경우를 나타낸 것이다.

t.test(mpg ~ vs, data= mtcars, var.equal = FALSE) %>%   
 jjstat::report\_ttest(trans = TRUE)

##   
##   
## 종속변수 'mpg'에 대한 독립변수 'vs'의 'gV-shaped(16.62)'와 'gstraight(24.56)' 두 집단 간의 t-test 결과는 통계적으로 유의한 것으로 나타났다( t(22.72) = -4.67, p < .001).   
##   
##   
##

ex) 오렌지 주스와 비타민을 복용시킨 2개의 그룹에서 기니피그의 이빨길이는 통계적으로 유의미한 차이가 있는가?

A data frame with 60 observations on 3 variables.  
[,1] len numeric Tooth length  
[,2] supp factor Supplement type (VC or OJ).  
[,3] dose numeric Dose in milligrams/day

# Two-samples unpaired test  
data("ToothGrowth")  
ToothGrowth %>%str()

## 'data.frame': 60 obs. of 3 variables:  
## $ len : num 4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...  
## $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...  
## $ dose: num 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...

**1. 등분산 검정**

var.test(len ~ supp, ToothGrowth)

##   
## F test to compare two variances  
##   
## data: len by supp  
## F = 0.6386, num df = 29, denom df = 29, p-value = 0.2331  
## alternative hypothesis: true ratio of variances is not equal to 1  
## 95 percent confidence interval:  
## 0.3039488 1.3416857  
## sample estimates:  
## ratio of variances   
## 0.6385951

귀무가설:H0:등분산이다.  
F검정결과 두 변수는 등분산을 충족하였다, *F*(29) = 0.64, *p* = 0.23.

t.test(len ~ supp, data = ToothGrowth, var.equal=TRUE)

##   
## Two Sample t-test  
##   
## data: len by supp  
## t = 1.9153, df = 58, p-value = 0.06039  
## alternative hypothesis: true difference in means between group OJ and group VC is not equal to 0  
## 95 percent confidence interval:  
## -0.1670064 7.5670064  
## sample estimates:  
## mean in group OJ mean in group VC   
## 20.66333 16.96333

**APA보고**  
기니피그의 supplement(오랜지주스와 비타민C)이 차이에 따른 이빨길이의 차이에 대하여 독립표본t검정을 시행한 결과 유의하지않았다, *t*(58) = 1.92, *p* =.06.

**report package**를 이용한 결과

t.test(len ~ supp, data = ToothGrowth, var.equal=TRUE) %>%   
 jjstat::report\_ttest()

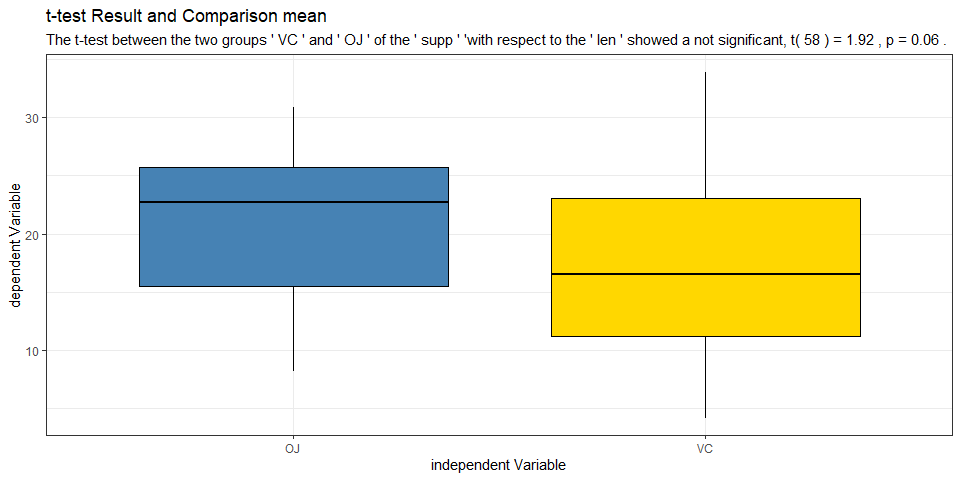
##   
## Two Sample t-test results between the two groups 'gOJ(20.66)' and 'gVC(16.96)' of the independent variable 'supp' with respect to the dependent variable 'len' showed a not statistically significant( t(58) = 1.92,p = 0.06)

##### 4.1.3.2. t.test 요약보고 함수

이번에는 필자가 개발한 jjstat package에 있는 t\_test\_report()함수를 이용하여 독립표본 t검정을 진행해보자. 이 함수는 먼저 var.test를 통하여 분산의 동질성 검정을 시행한다. 그리고 var.test의 결과에 대한 결과보고를 해준다. 이후에는 t.test()를 수행한 후에 t.test의 결과에 대한 보고를 해준다. 그리고 평균을 비교하는 것을 시각화하기 위해서 boxplot을 그려서 보여주는 함수이다.  
위에서 배운 개념을 모두 활용하는 것들을 모아서 보여주는 함수이다. 분석을 하면서 여러가지를 편리하게 만들어주는데 중요한 것은 t.test에 대한 깊은 이해를 한 후에 나타나는 결과는 매우 편리하게 만들어주지만, 정확한 이해없이 사용하면 그저 복잡한 결과만 출력되는 것이 된다. 따라서, 명확하게 인식하고 사용하는 것이 중요하다.

jjstat::t\_test\_report(data = ToothGrowth,   
 iv= "supp",  
 dv="len",   
 translate = FALSE)

## $var\_test\_result\_tibble  
## # A tibble: 2 × 4  
## Var df f\_value p\_value  
## <fct> <int> <dbl> <dbl>  
## 1 VC 29 0.639 0.233  
## 2 OJ 29 0.639 0.233  
##   
## $var\_test\_report  
## [1] "The t-test results between the two groups 'VC' and 'OJ' of the independent variable 'supp' with respect to the dependent variable 'len' showed anot statistically significant. F(29,29) = 0.64, p = 0.23. This indicates that the variances of the two groups are equal."  
##   
## $t\_test\_result\_tibble  
## # A tibble: 1 × 9  
## DV IV IV.1 IV.2 df t.value p.value sig test\_type   
## <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <chr> <chr>   
## 1 len supp VC(20.66) OJ(16.96) 58 1.92 0.0604 ns student's t.test  
##   
## $t\_test\_report  
## [1] "The t-test results between the two groups 'VC' and 'OJ' of the independent variable ' supp ''with respect to the dependent variable 'len' showed anot statistically significant. t(58) = 1.92,p = 0.06."  
##   
## $descriptive  
## # A tibble: 2 × 2  
## supp len[,"n"] [,"mean"] [,"sd"]  
## <fct> <dbl> <dbl> <dbl>  
## 1 OJ 30 20.7 6.61  
## 2 VC 30 17.0 8.27  
##   
## $boxplot



### 4.1.4. 대응표본 t검정

대응표본 t검정은 주로 실험에서 많이 쓰이는데, 어떤 처치에 대한 효과를 알아보고자 할 때 사용하는 경우가 많다.

예를 들면, 아래의 데이터는 시험데이터이다.  
학생이 중간고사를 본 후에 학원을 다녔다. 이후에 기말고사를 보았다. 학원을 다닌 후에 성적의 차이가 유의하게 나타났는지 검정해보고자 하였다.

대응표본 t-test는 분산의 동질성 검정을 하지 않는다. 같은 그룹에서 표집을 진행했기 때문이다.

표준오차는

으로 계산한다. t의 임계값을 구하여 유의성을 검정하거나 p값을 구하여 통계적 유의성을 검정한다.

ex) 시험성적은 학원을 다닌후에 유의하게 향상되었을까?

| mid | final |
| --- | --- |
| 16 | 19 |
| 20 | 22 |
| 21 | 24 |
| 22 | 24 |
| 23 | 25 |
| 22 | 25 |
| 27 | 26 |
| 25 | 26 |
| 27 | 28 |
| 28 | 32 |

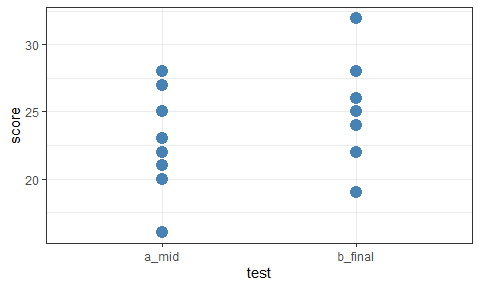
이 데이터를 롱포맷으로 만들어서 분석해보자.

exam\_long = data.frame(  
 test = c(rep("a\_mid", 10), rep("b\_final", 10)),  
 score = c(16, 20, 21, 22, 23, 22, 27, 25, 27, 28,  
 19, 22, 24, 24, 25, 25, 26, 26, 28, 32)  
)  
exam\_long

## test score  
## 1 a\_mid 16  
## 2 a\_mid 20  
## 3 a\_mid 21  
## 4 a\_mid 22  
## 5 a\_mid 23  
## 6 a\_mid 22  
## 7 a\_mid 27  
## 8 a\_mid 25  
## 9 a\_mid 27  
## 10 a\_mid 28  
## 11 b\_final 19  
## 12 b\_final 22  
## 13 b\_final 24  
## 14 b\_final 24  
## 15 b\_final 25  
## 16 b\_final 25  
## 17 b\_final 26  
## 18 b\_final 26  
## 19 b\_final 28  
## 20 b\_final 32

이를 시각화하여 그림으로 나타내보자. 각 성적의 변화를 보자.

exam\_long %>%   
 ggplot(aes(x=test, y= score))+  
 geom\_point(size=4, color= "steelblue")+  
 theme\_bw()



t.test(score ~ test, data= exam\_long, paired=TRUE)

##   
## Paired t-test  
##   
## data: score by test  
## t = -4.4721, df = 9, p-value = 0.00155  
## alternative hypothesis: true mean difference is not equal to 0  
## 95 percent confidence interval:  
## -3.0116674 -0.9883326  
## sample estimates:  
## mean difference   
## -2

# exam\_long

paired t.test결과, 학원을 다닌후에 성적을 유의하게 올랐다고 할 수 있다, t(9) =-4.47, p = .002.

데이터가 롱포맷이면 수식을 이용하여 분석한다. 종속변수 ~ 독립변수로 입력하면된다.

library(report)  
t.test(score ~ test, data= exam\_long, paired=TRUE) %>%   
 report\_ttest()

##   
## Paired t-test results between the two groups 'mean difference(-2)' and 'NA(NA)' of the independent variable 'test' with respect to the dependent variable 'score' showed a statistically significant( t(9) = -4.47,p = 0)

##### 데이터가 와이드 포맷인 경우

데이터가 와이드포맷일 때에는 변수명을 선택하여 분석하면 된다.

#데이터 생성   
exam = cbind.data.frame(  
 mid = c(16, 20, 21, 22, 23, 22, 27, 25, 27, 28),  
 final = c(19, 22, 24, 24, 25, 25, 26, 26, 28, 32) )  
exam

## mid final  
## 1 16 19  
## 2 20 22  
## 3 21 24  
## 4 22 24  
## 5 23 25  
## 6 22 25  
## 7 27 26  
## 8 25 26  
## 9 27 28  
## 10 28 32

t.test(변수1, 변수2, paired=TRUE)의 형식으로 분석하면 된다.

t.test(exam$mid, exam$final, paired= TRUE)

##   
## Paired t-test  
##   
## data: exam$mid and exam$final  
## t = -4.4721, df = 9, p-value = 0.00155  
## alternative hypothesis: true mean difference is not equal to 0  
## 95 percent confidence interval:  
## -3.0116674 -0.9883326  
## sample estimates:  
## mean difference   
## -2

paired t.test결과, 학원을 다닌후에 성적을 유의하게 올랐다고 할 수 있다, (9) = -4.47, = .002.

## 4.2. oneway ANOVA

**일원 분산 분석 (One-way ANOVA)**은 세 개 이상의 그룹 간 평균 차이를 통계적으로 분석하는 방법이다. 이 분석은 하나의 독립 변수 또는 요인만 고려할 때 사용되는 방법으로 그룹 간 변동과 그룹 내 변동을 비교하여 해당 요인이 종속 변수에 미치는 영향을 조사할 때 사용한다. 일원 분산 분석은 귀무 가설 ()과 대립 가설 ()을 검정한다. 는 모든 그룹 평균이 동일하다는 가설이며, 는 적어도 하나의 그룹 평균이 전체 평균과 다르다는 가설이다. 통계적 유의성 검정은 p-값을 통해 진행하며, 이면 평균이 서로 다르다는 근거로 사용된다. 즉 0.05보다 작은 p-값은 귀무 가설을 기각하며, 평균이 서로 다르다고 판단한다.

**일원분산분석 가정**

* 독립성 가정: 각 그룹의 데이터는 서로 독립적이어야 한다.
* 정규성 가정: 각 그룹의 데이터는 정규 분포를 따라야 한다.
* 등분산 가정: 그룹 내 변동은 그룹 전반에서 유사해야 한다.

### 4.2.1. oneway ANOVA

일원배치 분산분석을 실행하면서 분석의 기법를 이해보도록 하자.

**데이터**  
데이터를 잘 이해하는 것은 분석할 때 매우 유리하다. 그룹은 총 4개로 하여 성적을 측정하였다.

# 01 oneway anova ------------------------------------------------------------  
fuel = data.frame(  
 trt = c(1L,1L,1L,1L,1L,1L,2L,  
 2L,2L,2L,2L,2L,3L,3L,3L,3L,3L,3L,4L,4L,4L,  
 4L,4L,4L),  
 y = c(64L,72L,68L,77L,56L,95L,  
 78L,91L,97L,82L,85L,77L,75L,93L,78L,71L,63L,  
 76L,55L,66L,49L,64L,70L,68L)  
 )  
  
fuel |> head()

## trt y  
## 1 1 64  
## 2 1 72  
## 3 1 68  
## 4 1 77  
## 5 1 56  
## 6 1 95

#### 4.2.1.1. 데이터의 구조보기

fuel$trt <- as.factor(fuel$trt) #변수의 factor처리   
fuel %>% str()

## 'data.frame': 24 obs. of 2 variables:  
## $ trt: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 2 2 2 2 ...  
## $ y : int 64 72 68 77 56 95 78 91 97 82 ...

데이터를 wide format으로 변경하여 보자. 일반적으로 프로그램에서 보는 형태이다.

# wideData  
fuel %>%   
 mutate(rowid= rep(1:6,4)) %>%  
 pivot\_wider(values\_from = y, names\_from = trt)

## # A tibble: 6 × 5  
## rowid `1` `2` `3` `4`  
## <int> <int> <int> <int> <int>  
## 1 1 64 78 75 55  
## 2 2 72 91 93 66  
## 3 3 68 97 78 49  
## 4 4 77 82 71 64  
## 5 5 56 85 63 70  
## 6 6 95 77 76 68

#### 4.2.1.2. 기술통계 분석

기술통계를 분석하는 여러가지 방법이 있지만, 분석을 하다보면 가장 빠르고 편리하게 사용하는 방법이어서 소개한다.

#statistical describe   
fuel %>%  
 group\_by(trt) %>%   
 summarise(M=mean(y), SD=sd(y),N=n(),  
 Median=median(y), Max=max(y),Min=min(y),  
 skewness=SKEW(y),  
 kurtosis=KURT(y),  
 SE=sd(y)/sqrt(n()),  
 lowerCI = mean(y)-1.96\*SE,  
 upperCI = mean(y)+1.96\*SE  
 ) #%>% kable("pandoc", digits = 2, caption="기술통계분석")

## # A tibble: 4 × 12  
## trt M SD N Median Max Min skewness kurtosis SE lowerCI  
## <fct> <dbl> <dbl> <int> <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 1 72 13.3 6 70 95 56 0.535 -1.16 5.45 61.3  
## 2 2 85 7.77 6 83.5 97 77 0.377 -1.69 3.17 78.8  
## 3 3 76 9.88 6 75.5 93 63 0.449 -1.03 4.03 68.1  
## 4 4 62 8.22 6 65 70 49 -0.522 -1.66 3.36 55.4  
## # ℹ 1 more variable: upperCI <dbl>

참고로 jjstat 패키지를 사용하면 다음과 같은 방법을 사용하기도 한다.mysummaryBy()는 그룹별 기술통계를 나태는 주는 함수로 jjstat::mysummaryBy(data= fuel, y ~ trt, … )와 같이 계속해서 formula를 입력하면 누적하여 분석하는 함수이다.

#the other packages psych   
jjstat::mysummaryBy( y ~ trt, data = fuel)

## # A tibble: 4 × 8  
## var Mean SD N Min Max Skew Kurt  
## <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 72 13.3 6 56 95 0.535 -1.16  
## 2 2 85 7.77 6 77 97 0.377 -1.69  
## 3 3 76 9.88 6 63 93 0.449 -1.03  
## 4 4 62 8.22 6 49 70 -0.522 -1.66

## same method   
# fuel |> mysummaryBy(formula = y~ trt)  
  
#다른방법 psych패키지 적용  
# psych::describeBy(fuel ~ trt)

### 4.2.1. ANOVA process

ANOVA를 하는 방법은 매우 간단하다. 이렇게 간단해도 될 정도인가?라고 생각이 들정도로 사용방법은 너무나 간단하다. 따라서, 코드를 명확히 익히고 분석하는 것이 필요하고, 무엇보다 분산분석표를 이해하는 것이 매우 중요하다.

aov(y~ trt, data=fuel) %>%   
 summary()

## Df Sum Sq Mean Sq F value Pr(>F)   
## trt 3 1636 545.5 5.406 0.00688 \*\*  
## Residuals 20 2018 100.9   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

H0: 4개 그룹의 평균은 같다.  
H1: not H0  
일원분산 분석결과, trt의 4가지 그룹은 통계적으로 유의한 차이가 나타났다, F(3, )= 5.41, p =.01.

aov(y~ trt, data=fuel) |>  
 report\_aov()

## one-way analysis of variance result for dv[y]~[trt], It was statistically significant(F(3, 20) = 5.41, p = 0.01).   
##

<**참고**>  
lm()함수와 anova()함수를 이용하여 분석하는 방법도 있다. lm()은 회귀분석(regression)을 수행하는 함수이다. 그러나 독립변수가 명목형 변수인 경우에는 일반화 선형 모형에 근거하면 aov()와 같음 함수가 된다. 이 부분은 별도의 부록에 설명해두었다.

lm(y ~ trt, data=fuel) %>%   
 anova()

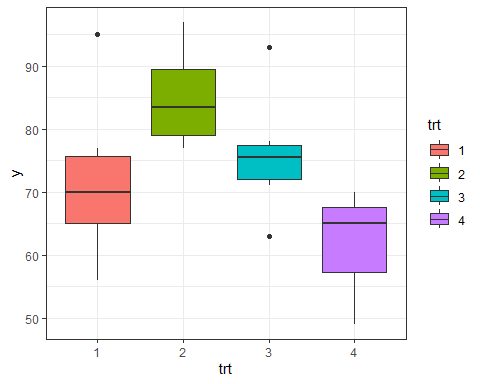
## Analysis of Variance Table  
##   
## Response: y  
## Df Sum Sq Mean Sq F value Pr(>F)   
## trt 3 1636.5 545.5 5.4063 0.006876 \*\*  
## Residuals 20 2018.0 100.9   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

**평균비교 시각화**

평균을 비교할 때, 가장 눈에 쉽게 비교해주는 방법은 시각화하는 방법이다.

A boxplot is a standardized way of displaying the dataset based on the five-number summary: the minimum, the maximum, the sample median, and the first and third quartiles. Minimum (Q0 or 0th percentile): the lowest data point in the data set excluding any outliers(from WIKI).

# boxplot visualization   
fuel %>%   
 ggplot(aes(x=trt, y=y, fill=trt))+  
 geom\_boxplot()+  
 theme\_bw()



### 4.2.1. Post Hoc(사후분석석)

ANOVA가 통계적으로 유의하였으므로 이제는 어떤 그룹간 차이가 발생했지는 확인해봐야 한다.쌍으로 비교하여 실제로 평균의 차이가 어디에서 발생했는지 확인한다.  
이러한 과정을 사후분석(POST HOC)이라고 한다.

#post HOC  
aov(y ~ trt, data=fuel) %>%   
 TukeyHSD()

## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = y ~ trt, data = fuel)  
##   
## $trt  
## diff lwr upr p adj  
## 2-1 13 -3.232221 29.232221 0.1461929  
## 3-1 4 -12.232221 20.232221 0.8998057  
## 4-1 -10 -26.232221 6.232221 0.3378150  
## 3-2 -9 -25.232221 7.232221 0.4270717  
## 4-2 -23 -39.232221 -6.767779 0.0039064  
## 4-3 -14 -30.232221 2.232221 0.1065573

이 결과를 좀더 tidy하게 정리해보면 다음과 같다.  
분석을 정리한 후에 유의성을 표시하였다.

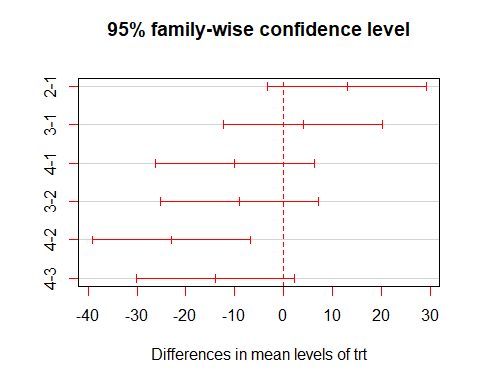
#broom packages의 tidy  
aov(y~ trt, data=fuel) %>%   
 TukeyHSD() %>%   
 broom::tidy() %>%   
 dplyr::select(2,4,5,6,7) %>%   
 rename(p.value=adj.p.value) %>%  
 p\_mark\_sig() %>%   
 tibble()

## # A tibble: 6 × 6  
## contrast estimate conf.low conf.high p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 2-1 13.0 -3.23 29.2 0.146 ns   
## 2 3-1 4.00 -12.2 20.2 0.900 ns   
## 3 4-1 -10.0 -26.2 6.23 0.338 ns   
## 4 3-2 -9 -25.2 7.23 0.427 ns   
## 5 4-2 -23 -39.2 -6.77 0.00391 \*\*   
## 6 4-3 -14 -30.2 2.23 0.107 ns

그룹 4와 2가 통계적으로 유의한 차이가 나타났다, diff = -23, 95%CI[-39.23,-6.76], *p* = .003.

보통은 이렇게 정리하지 않고 시각화 하여 유의성을 보기도 한다. 이때 plot()만 추가하면 된다.  
신뢰구간을 이용하여 나타낸 것이므로 0을 신뢰구간이 0을 포함하지 않는 경우에 유의하다고 볼 수 있다.

#plot   
aov(y ~ trt, data=fuel) %>%   
 TukeyHSD() %>%  
 plot(col="red")



### 4.2.2. Planned Contrast

사후분석을 위해서는 보통 emmeans를 사용하는 경우가 많다. 가장 일반적이라고 할 수 있다. 물론 방법은 다양하지만 기억하기 쉽고 사용하기가 쉽다. 먼저 marginal mean을 구한다. 그후에 대비계수를 만들고, 대비검정을 수행하면 된다(대비검정이나 계획비교를 혼용하여 사용하는 중인데, 같은 의미로 보면 된다).

#marginal mean   
library(emmeans)  
  
aov(y~ trt, data=fuel) %>%   
 emmeans("trt")

## trt emmean SE df lower.CL upper.CL  
## 1 72 4.1 20 63.4 80.6  
## 2 85 4.1 20 76.4 93.6  
## 3 76 4.1 20 67.4 84.6  
## 4 62 4.1 20 53.4 70.6  
##   
## Confidence level used: 0.95

# lm(y~ trt, data=fuel) %>% emmeans("trt") #same method

계획해서 비교하고 싶은 것은 그룹2 vs 그룹1, 3, 4의 합이다. 그리고, 그룹1+그룹1 vs 그룹3+그룹4를 비교해보자.  
그런데, TukeyHSD를 함께 보여주기 위해서 contrast계수를 모두 설정했다. 대비계수를 설정할 때 중요한 것은 수직대비(Othogonal cantrast)가 되어야 한다.

contrast\_coef = list(  
 "2-1"= c(-1, 1, 0, 0),  
 "3-1"= c(-1, 0, 1, 0),  
 "4-1"= c(-1, 0, 0, 1),  
 "3-2"= c( 0,-1, 1, 0),  
 "4-2"= c( 0,-1, 0, 1),  
 "4-3"= c( 0, 0,-1, 1),  
 "m1-m2+m3+m4"= c(1,-3, 1,1), # 검증하고자 하는 가설 1  
 "m1+m2-m3-m4"= c(1,1,-1,-1) # 검증하고자 하는 가설 2   
)  
contrast\_coef

## $`2-1`  
## [1] -1 1 0 0  
##   
## $`3-1`  
## [1] -1 0 1 0  
##   
## $`4-1`  
## [1] -1 0 0 1  
##   
## $`3-2`  
## [1] 0 -1 1 0  
##   
## $`4-2`  
## [1] 0 -1 0 1  
##   
## $`4-3`  
## [1] 0 0 -1 1  
##   
## $`m1-m2+m3+m4`  
## [1] 1 -3 1 1  
##   
## $`m1+m2-m3-m4`  
## [1] 1 1 -1 -1

#planned contrast analysis   
aov(y~ trt, data=fuel) %>%   
 emmeans("trt") %>%   
 contrast(contrast\_coef,  
 adjust="bonferroni") %>%  
 p\_mark\_sig()

## # A tibble: 8 × 7  
## contrast estimate SE df t.ratio p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 2-1 13.0 5.80 20 2.24 0.292 ns   
## 2 3-1 4.00 5.80 20 0.690 1 ns   
## 3 4-1 -10.0 5.80 20 -1.72 0.801 ns   
## 4 3-2 -9.00 5.80 20 -1.55 1 ns   
## 5 4-2 -23 5.80 20 -3.97 0.00610 \*\*   
## 6 4-3 -14 5.80 20 -2.41 0.204 ns   
## 7 m1-m2+m3+m4 -45.0 14.2 20 -3.17 0.0387 \*   
## 8 m1+m2-m3-m4 19 8.20 20 2.32 0.250 ns

4-2에 관한 차이가 유의했고, 가설1은 통계적으로 유의하였다.

그리고 p값에 대한 adjust가 적용되지 않으면 유의한 것들이 더 생겨나게 된다. 그렇다면 채택되지 말아야 할 귀무가설이 채택되는지 확인해보자. p값에 대한 보정을 하지 않고 “none”로 한 경우이다. 4-3도 유의하게 나타났고, 가설2도 통계적으로 유의하게 나타났다. bonferroni 교정의 경우는 유의수준을 총 검정의 수(N)로 나누어서 개별 검정을 보정를 하는 방법이다.

#planned contrast analysis   
aov(y~ trt, data=fuel) %>%  
 emmeans("trt") %>%   
 contrast(contrast\_coef,  
 adjust="none") %>%  
 p\_mark\_sig()

## # A tibble: 8 × 7  
## contrast estimate SE df t.ratio p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 2-1 13.0 5.80 20 2.24 0.0365 \*   
## 2 3-1 4.00 5.80 20 0.690 0.498 ns   
## 3 4-1 -10.0 5.80 20 -1.72 0.100 ns   
## 4 3-2 -9.00 5.80 20 -1.55 0.136 ns   
## 5 4-2 -23 5.80 20 -3.97 0.000762 \*\*\*   
## 6 4-3 -14 5.80 20 -2.41 0.0255 \*   
## 7 m1-m2+m3+m4 -45.0 14.2 20 -3.17 0.00484 \*\*   
## 8 m1+m2-m3-m4 19 8.20 20 2.32 0.0313 \*

#### 4.2.2.1. 95%신뢰구간 산정과 시각화

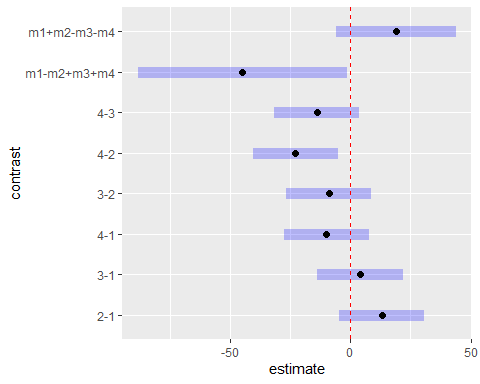
신뢰구간과 유의성을 표시해준다.

#confidence interval   
aov(y~ trt, data=fuel) %>%   
 emmeans("trt") %>%   
 contrast(contrast\_coef,  
 adjust="bonferroni", ) %>%   
 confint()

## contrast estimate SE df lower.CL upper.CL  
## 2-1 13 5.8 20 -4.72 30.72  
## 3-1 4 5.8 20 -13.72 21.72  
## 4-1 -10 5.8 20 -27.72 7.72  
## 3-2 -9 5.8 20 -26.72 8.72  
## 4-2 -23 5.8 20 -40.72 -5.28  
## 4-3 -14 5.8 20 -31.72 3.72  
## m1-m2+m3+m4 -45 14.2 20 -88.40 -1.60  
## m1+m2-m3-m4 19 8.2 20 -6.06 44.06  
##   
## Confidence level used: 0.95   
## Conf-level adjustment: bonferroni method for 8 estimates

#### 4.2.2.2. 신뢰구간의 유의성

#plotting   
aov(y~ trt, data=fuel) %>%   
 emmeans("trt") %>%   
 contrast(contrast\_coef,  
 adjust="bonferroni", ) %>%   
 plot()+  
 geom\_vline(xintercept = 0, col= "red", linetype=2)



#### 4.2.2.3. 계획비교를 TukeyHSD처럼 하고 싶을 때, pairs()

TukeyHSD()는 쌍으로 두개의 변수끼리 차를 이용하여 비교하는 것이다. 이러한 방법을 emmeans에서도 사용할 수 있는데 이때 사용하는 함수가 pairs()이다.

# similarity of TukeyHSD  
aov(y~ trt, data=fuel) %>%   
 emmeans("trt") %>%   
 pairs(simple = "trt") %>%   
 p\_mark\_sig()

## # A tibble: 6 × 7  
## contrast estimate SE df t.ratio p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 trt1 - trt2 -13.0 5.80 20 -2.24 0.146 ns   
## 2 trt1 - trt3 -4.00 5.80 20 -0.690 0.900 ns   
## 3 trt1 - trt4 10.0 5.80 20 1.72 0.338 ns   
## 4 trt2 - trt3 9.00 5.80 20 1.55 0.427 ns   
## 5 trt2 - trt4 23 5.80 20 3.97 0.00391 \*\*   
## 6 trt3 - trt4 14 5.80 20 2.41 0.107 ns

이 방법을 이용하여 쉽게 TukeyHSD와 같은 비교를 할 수 있다. TukeyHSD()가 p값을 adjust가 되지 않고 자동 적용이 된다. 이런 경우 pairs()를 이용하면 p값을 조절해서 확인해볼 수 있다.

# similarity of TukeyHSD  
aov(y~ trt, data=fuel) %>%   
 emmeans("trt") %>%   
 pairs(simple = "trt", adjust="none") %>%  
 p\_mark\_sig() %>%   
 tibble()

## # A tibble: 6 × 7  
## contrast estimate SE df t.ratio p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 trt1 - trt2 -13.0 5.80 20 -2.24 0.0365 \*   
## 2 trt1 - trt3 -4.00 5.80 20 -0.690 0.498 ns   
## 3 trt1 - trt4 10.0 5.80 20 1.72 0.100 ns   
## 4 trt2 - trt3 9.00 5.80 20 1.55 0.136 ns   
## 5 trt2 - trt4 23 5.80 20 3.97 0.000762 \*\*\*   
## 6 trt3 - trt4 14 5.80 20 2.41 0.0255 \*

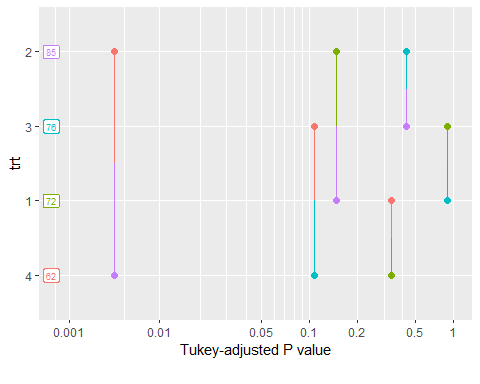
**사후분석을 행렬로 표현** 사후분석을 행렬 형태로 분석하는 방법은 누군가에게는 익숙할 수 있다. 그 이유는 예전 사후분석 패키지의 방식이 행렬을 통해서 나타내는 경우가 종종 있었기 때문이다.  
행렬의 하단에는 diff값을 나타내고, 상단에는 유의성을 나타내는 p값이 있다.

aov(y~ trt, data=fuel) %>%  
 emmeans("trt") %>%  
 pwpm()

## 1 2 3 4  
## 1 [72] 0.1462 0.8998 0.3378  
## 2 -13 [85] 0.4271 0.0039  
## 3 -4 9 [76] 0.1066  
## 4 10 23 14 [62]  
##   
## Row and column labels: trt  
## Upper triangle: P values adjust = "tukey"  
## Diagonal: [Estimates] (emmean)   
## Lower triangle: Comparisons (estimate) earlier vs. later

**시각화**  
그림을 보면 다음과 같다. x축에는 p값을 나타냈고, y축에는 각 그룹의 평균과 연결된 부분을 나타냈다. 각 선 그래프는 p값의 위치가 어디에서 나타나는지 구체적으로 나타내고 있어서 이해하기 쉽다는 장점이 있다.

aov(y~ trt, data=fuel) %>%  
 emmeans("trt") %>%   
 pwpp()



###4.2.3.multicomp::glht()를 이용한 계획비교

이번에는 다른 방법을 더 소개한다. glht(general linear hypothesis test)를 사용하여 계획비교를 하는 방법을 알아보고자 한다. 계획비교를 하는 방법을 먼저 one way ANOVA를 설명하면서 emmeans를 이용하는 방법을 설명하였다. 이번에는 좀 다른 방법을 설명하고자 한다. 전체적인 맥락은 비슷하기 때문에 어려움은 없을 것으로 예상된다.

사용할 데이터는 기존과 동일하다.

# fuel = data.frame(  
# trt = c(1L,1L,1L,1L,1L,1L,2L,  
# 2L,2L,2L,2L,2L,3L,3L,3L,3L,3L,3L,4L,4L,4L,  
# 4L,4L,4L),  
# y = c(64L,72L,68L,77L,56L,95L,  
# 78L,91L,97L,82L,85L,77L,75L,93L,78L,71L,63L,  
# 76L,55L,66L,49L,64L,70L,68L)  
# )  
fuel <- jjstat::fuel  
  
# factor processing of variable  
fuel$trt <- as.factor(fuel$trt)  
fuel %>% str()

## 'data.frame': 24 obs. of 2 variables:  
## $ trt: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 2 2 2 2 ...  
## $ y : int 64 72 68 77 56 95 78 91 97 82 ...

분산분석의 실행  
분산분석을 통해서 값을 정확히 알아보자

library(multcomp) #gjht  
aov(y~ trt, data=fuel) |>   
 summary()

## Df Sum Sq Mean Sq F value Pr(>F)   
## trt 3 1636 545.5 5.406 0.00688 \*\*  
## Residuals 20 2018 100.9   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

**대비계수의 설정**  
여기에서는 대비계수를 설정할 때 행렬로 만든다. rbind()이용하여 결합해둔다. (참고로 emmeans에서는 list()데이터로 만들었었다)

# define contrast matrix directly  
contrast\_coef2 = rbind(  
 "2-1"= c(-1, 1, 0, 0),  
 "3-1"= c(-1, 0, 1, 0),  
 "4-1"= c(-1, 0, 0, 1),  
 "3-2"= c( 0,-1, 1, 0),  
 "4-2"= c( 0,-1, 0, 1),  
 "4-3"= c( 0, 0,-1, 1),  
 "m1-m2+m3+m4"= c(1,-3, 1,1),  
 "m1+m2-m3-m4"= c(1,1,-1,-1)  
) %>% `colnames<-`(c("m1","m2","m3","m4"))  
  
contrast\_coef2

## m1 m2 m3 m4  
## 2-1 -1 1 0 0  
## 3-1 -1 0 1 0  
## 4-1 -1 0 0 1  
## 3-2 0 -1 1 0  
## 4-2 0 -1 0 1  
## 4-3 0 0 -1 1  
## m1-m2+m3+m4 1 -3 1 1  
## m1+m2-m3-m4 1 1 -1 -1

<*참고설명*> colnames<-(c(“m1”,“m2”,“m3”,“m4”))부분은 컬럼명을 설정하는 방법이다. %>%를 사용할 때 이 방법이 생각보다 더 유용한 방법이 된다. 이 함수는 R의 base함수중 하나이다.

**계획비교(Contrast)**  
분산분석후 glht()함수를 실행한다. linfct옵션에 mcp()에 factor변수 = 대비계수를 설정해주는 방법으로 설정해주면 된다.

aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = contrast\_coef2)) %>%   
 summary()

##   
## Simultaneous Tests for General Linear Hypotheses  
##   
## Multiple Comparisons of Means: User-defined Contrasts  
##   
##   
## Fit: aov(formula = y ~ trt, data = fuel)  
##   
## Linear Hypotheses:  
## Estimate Std. Error t value Pr(>|t|)   
## 2-1 == 0 13.000 5.799 2.242 0.15506   
## 3-1 == 0 4.000 5.799 0.690 0.90395   
## 4-1 == 0 -10.000 5.799 -1.724 0.35076   
## 3-2 == 0 -9.000 5.799 -1.552 0.44033   
## 4-2 == 0 -23.000 5.799 -3.966 0.00452 \*\*  
## 4-3 == 0 -14.000 5.799 -2.414 0.11385   
## m1-m2+m3+m4 == 0 -45.000 14.206 -3.168 0.02544 \*   
## m1+m2-m3-m4 == 0 19.000 8.202 2.317 0.13598   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## (Adjusted p values reported -- single-step method)

4-2의 차이가 유의하게 나타났으며, 가설1번(m1-m2+m3+m4 == 0 )이 유의하게 나타났다.

자유도를 구해보면, 20이 되었다.

# fuel의 개수 24, trt factor variable level은 4개(1,2,3,4)  
nrow(fuel)-length(levels(fuel$trt))

## [1] 20

이는 상당히 유용한 정보이다. 무슨 말이냐면, 두 개씩 비교하는 경우(t.test)의 경우는 자유도가 5일수 밖에 없다. 그러나 지금은 자유도가 높게 나타난다는 것은 표준오차의 공식에서 에서 n값이 커졌다는 의미가 된다. 이는 표준오차를 작아지게 만든다는 것을 알려주기 때문이다.

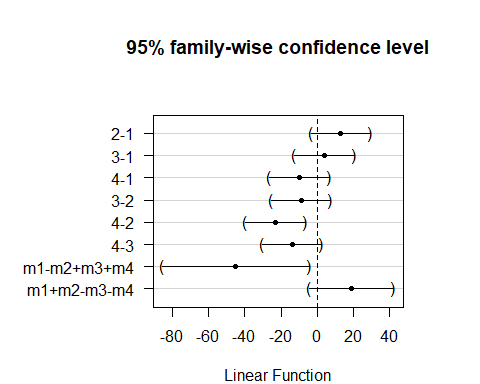
신뢰구간 을 나타내기

#신뢰구간   
aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = contrast\_coef2)) %>%   
 confint()

##   
## Simultaneous Confidence Intervals  
##   
## Multiple Comparisons of Means: User-defined Contrasts  
##   
##   
## Fit: aov(formula = y ~ trt, data = fuel)  
##   
## Quantile = 2.8387  
## 95% family-wise confidence level  
##   
##   
## Linear Hypotheses:  
## Estimate lwr upr   
## 2-1 == 0 13.0000 -3.4627 29.4627  
## 3-1 == 0 4.0000 -12.4627 20.4627  
## 4-1 == 0 -10.0000 -26.4627 6.4627  
## 3-2 == 0 -9.0000 -25.4627 7.4627  
## 4-2 == 0 -23.0000 -39.4627 -6.5373  
## 4-3 == 0 -14.0000 -30.4627 2.4627  
## m1-m2+m3+m4 == 0 -45.0000 -85.3252 -4.6748  
## m1+m2-m3-m4 == 0 19.0000 -4.2818 42.2818

시각화

par(mar=c(4,8,6,4))  
aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = contrast\_coef2)) %>% plot()



**Tukey를 사용하는 방법**

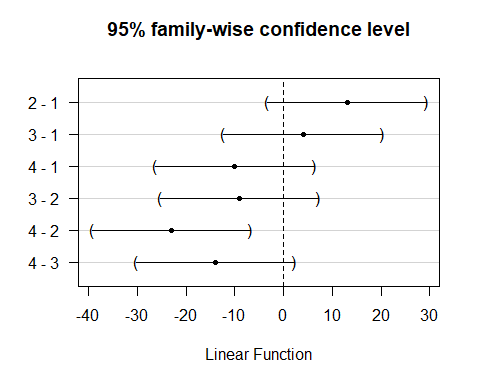
glht()에서는 Tukey를 사용하는 방법이 mcp()에서 명목변수인 trt에 “Tukey”를 설정하는 방법이다.

# Tukey   
aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = "Tukey")) %>%  
 summary()

##   
## Simultaneous Tests for General Linear Hypotheses  
##   
## Multiple Comparisons of Means: Tukey Contrasts  
##   
##   
## Fit: aov(formula = y ~ trt, data = fuel)  
##   
## Linear Hypotheses:  
## Estimate Std. Error t value Pr(>|t|)   
## 2 - 1 == 0 13.000 5.799 2.242 0.14619   
## 3 - 1 == 0 4.000 5.799 0.690 0.89982   
## 4 - 1 == 0 -10.000 5.799 -1.724 0.33797   
## 3 - 2 == 0 -9.000 5.799 -1.552 0.42719   
## 4 - 2 == 0 -23.000 5.799 -3.966 0.00391 \*\*  
## 4 - 3 == 0 -14.000 5.799 -2.414 0.10649   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## (Adjusted p values reported -- single-step method)

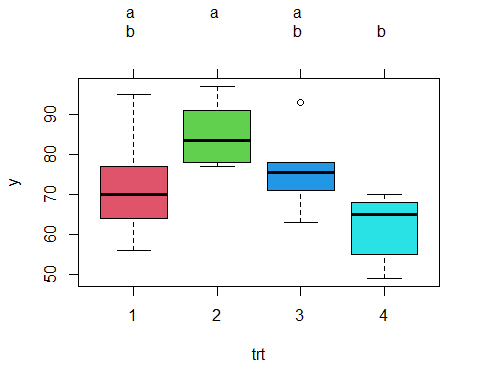
Tukey 사후분석을 이용한 경우의 신뢰구간의 시각화

aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = "Tukey")) %>% plot()



Boxplot을 이용한 사후분석 결과 표시 이 방법도 매우 유용하며 의미가 있다.  
Boxplot에서 유의한 것이 무엇인지 보여주는 방식이다. cld()와 plot()을 이용하여 나타낸다.

#bocplot   
aov(y~ trt, data=fuel) %>%   
 glht(linfct = mcp(trt = "Tukey")) %>%   
 cld(level = 0.05) %>%   
 plot(col=c(2:5))



여기에서는 유의성 표시를 a, b 등을 이용하여 나타낸 것이다. ab는 서로 구분되지 않는다는 의미이므로 2는 a로 나타났고, 4는 b로 나타났으므로 이 둘은 유의미한 차이가 난다는 의미이다. 아마도 전통적으로 많이 사용해온 방식이라 이것이 더 익숙할 수도 있다고 보여진다.

###4.2.4.oneway ANOVA 수기계산

이번에는 Wiki에 나온 데이터를 가지고 일원배치 분산분석(oneway ANOVA)에 대한 수기계산 공식을 확인하려고 한다. 데이터를 직접 계산하게 되면 전체적인 이해도가 높아지게 된다.

# df1 = jjstat::df1  
df1 = data.frame(a1 = c(6, 8, 4, 5, 3, 4),  
 a2 = c(8, 12, 9, 11, 6, 8),  
 a3 = c(13, 9 ,11, 8, 7, 12))  
df1

## a1 a2 a3  
## 1 6 8 13  
## 2 8 12 9  
## 3 4 9 11  
## 4 5 11 8  
## 5 3 6 7  
## 6 4 8 12

#### 4.2.4.1. ANOVA 수기계산

주어진 데이터를 롱포맷으로 변형하자.

df\_long <- df1 |>  
 pivot\_longer(cols = c(a1,a2,a3)) |>  
 arrange(name)  
df\_long

## # A tibble: 18 × 2  
## name value  
## <chr> <dbl>  
## 1 a1 6  
## 2 a1 8  
## 3 a1 4  
## 4 a1 5  
## 5 a1 3  
## 6 a1 4  
## 7 a2 8  
## 8 a2 12  
## 9 a2 9  
## 10 a2 11  
## 11 a2 6  
## 12 a2 8  
## 13 a3 13  
## 14 a3 9  
## 15 a3 11  
## 16 a3 8  
## 17 a3 7  
## 18 a3 12

데이터에 추가하는 방식으로 계산하면 다음과 같이 할 수 있다.

anovaData = df\_long %>% mutate(  
 grp\_mean = ave(df\_long$value, df\_long$name), #그룹별 평균  
 grand\_mean = mean(df\_long$value), #전체 평균   
 n = ave(df\_long$value, df\_long$name, FUN = function(x) length(x)), #그룹별샘플수  
 Sb = sum(unique(n\*(grp\_mean - grand\_mean)^2)), #between sum of square  
 dofb = length(unique(df\_long$name)) -1, #degree of freedom between   
 MSb = Sb/dofb, # mean of square bewteen   
 Sw = sum((value - grp\_mean)^2), #winthin sum of square   
 dofw = length(unique(df\_long$name))\*(n-1), #degree of freedom within  
 MSw = Sw/dofw, #mean of square within   
 F.value = MSb/MSw,   
 p.value = 1-pf(F.value, dofb, dofw)  
 )  
anovaData[,c(1,2,4,6,8,11,12,13)]

## # A tibble: 18 × 8  
## name value grand\_mean Sb MSb MSw F.value p.value  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 a1 6 8 84 42 4.53 9.26 0.00240  
## 2 a1 8 8 84 42 4.53 9.26 0.00240  
## 3 a1 4 8 84 42 4.53 9.26 0.00240  
## 4 a1 5 8 84 42 4.53 9.26 0.00240  
## 5 a1 3 8 84 42 4.53 9.26 0.00240  
## 6 a1 4 8 84 42 4.53 9.26 0.00240  
## 7 a2 8 8 84 42 4.53 9.26 0.00240  
## 8 a2 12 8 84 42 4.53 9.26 0.00240  
## 9 a2 9 8 84 42 4.53 9.26 0.00240  
## 10 a2 11 8 84 42 4.53 9.26 0.00240  
## 11 a2 6 8 84 42 4.53 9.26 0.00240  
## 12 a2 8 8 84 42 4.53 9.26 0.00240  
## 13 a3 13 8 84 42 4.53 9.26 0.00240  
## 14 a3 9 8 84 42 4.53 9.26 0.00240  
## 15 a3 11 8 84 42 4.53 9.26 0.00240  
## 16 a3 8 8 84 42 4.53 9.26 0.00240  
## 17 a3 7 8 84 42 4.53 9.26 0.00240  
## 18 a3 12 8 84 42 4.53 9.26 0.00240

anovaTableData <-anovaData %>%   
 dplyr::select(6:13) %>% unique()  
anovaTableData

## # A tibble: 1 × 8  
## Sb dofb MSb Sw dofw MSw F.value p.value  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 84 2 42 68 15 4.53 9.26 0.00240

분산분석표

rbind.data.frame(  
anovaTableData[1,c(1,2,3,7,8)] %>%   
 `colnames<-`(c("Sum\_sq","Df", "Mean\_sq","F","p")),  
anovaTableData[1,c(4,5,6,7,8)] %>%  
 `colnames<-`(c("Sum\_sq","Df", "Mean\_sq","F","p"))) %>%  
 as.data.frame() %>% `row.names<-`(c("between","within")) %>% round(3)

## Sum\_sq Df Mean\_sq F p  
## between 84 2 42.000 9.265 0.002  
## within 68 15 4.533 9.265 0.002

이것을 표로 나타내면 다음과 같이 나타낼 수 있다.

|  | Sum\_sq | Df | Mean\_sq | F | p |
| --- | --- | --- | --- | --- | --- |
| between | 84 | 2 | 42.000 | 9.265 | 0.002 |
| within | 68 | 15 | 4.533 |  |  |

분산분석 결과, 각 그룹은 통계적으로 유의한 차이가 나타났다, F(2, 15 ) = 9.26, p = 0.002.

각각의 과정을 살펴보자.

각 그룹의 평균

#group mean   
grp\_mean = ave(df\_long$value, df\_long$name) %>%   
 unique()%>%   
 data.frame() %>%   
 `rownames<-`(c("a1\_mean","a2\_mean","a3\_mean")) %>%   
 `colnames<-`(c("mean.value"))  
grp\_mean

## mean.value  
## a1\_mean 5  
## a2\_mean 9  
## a3\_mean 10

전체평균

#ground\_mean  
ground\_mean = mean(df\_long$value)  
ground\_mean

## [1] 8

그룹별샘플수

# n = df\_long %>%   
# group\_by(name) %>%   
# summarize(N= n())  
# n  
   
n = aggregate(data=df\_long, value ~ name, length)  
n

## name value  
## 1 a1 6  
## 2 a2 6  
## 3 a3 6

sum of square between

n[,2]\*(grp\_mean - ground\_mean)^2

## mean.value  
## a1\_mean 54  
## a2\_mean 6  
## a3\_mean 24

sum of square between

# sum of square between  
Sb= sum(n[,2]\*(grp\_mean - ground\_mean)^2)  
Sb

## [1] 84

degree of freedoom between

dofb = length(unique(df\_long$name)) -1 #degree of freedom between   
dofb

## [1] 2

sum of square within

# sum of squae within  
Sw = sum((df\_long$value - ave(df\_long$value, df\_long$name))^2)  
Sw

## [1] 68

degree of freedoom within

dofw = length(unique(df\_long$name))\*((n[1,2]) %>%   
 unlist()-1) #degree of freedom within  
dofw<- dofw %>% as.vector()  
dofw

## [1] 15

mean sum of square

MSb = Sb/dofb %>% as.vector() # mean of square bewteen   
MSw = Sw/dofw %>% as.vector() #mean of square within   
F.value = MSb/MSw   
  
MSb

## [1] 42

MSw

## [1] 4.533333

F.value

## [1] 9.264706

통계적 유의성 검정 여기에서 임계치의 F값을 구해보자. qf()를 이용하여 구하게 된다.

qf(1-.05, dofb, dofw)

## [1] 3.68232

현재 나타난 F값은 9.26이고 임계치는 3.68이므로 F값(9.26) > 임계치 F값(3.68)이므로 통계적으로 유의하다. 다른 방법으로는 p값을 구하여 확인하는 방법이 있다.

p value

#p  
p.value = 1-pf(F.value, dofb, dofw)  
p.value

## [1] 0.002398777

p< 0.001이므로 귀무가설을 기각하고 대립가설을 선택하면 된다.

이렇게 분해해서 값을 구하면 생각보다 일이 좀 많다. 그래서 데이터내에서 계산을 마치는 것이 더 낫다는 것이다. 이렇게 계산하면서 일원배치 분산분석(Oneway ANOVA)에 대한 이해를 높였을 것으로 생각도니다.

#### 4.2.4.2. aov()

‘aov(value ~ factor(name), df\_long) %>% summary()’ R의 oneway-ANOVA 를 실행하는 함수는 aov()이다. 이를 이용하면 한줄로 간단하게 분산분석을 마무리할 수 있다.

aov(value ~ name, df\_long) %>%   
 summary()

## Df Sum Sq Mean Sq F value Pr(>F)   
## name 2 84 42.00 9.265 0.0024 \*\*  
## Residuals 15 68 4.53   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

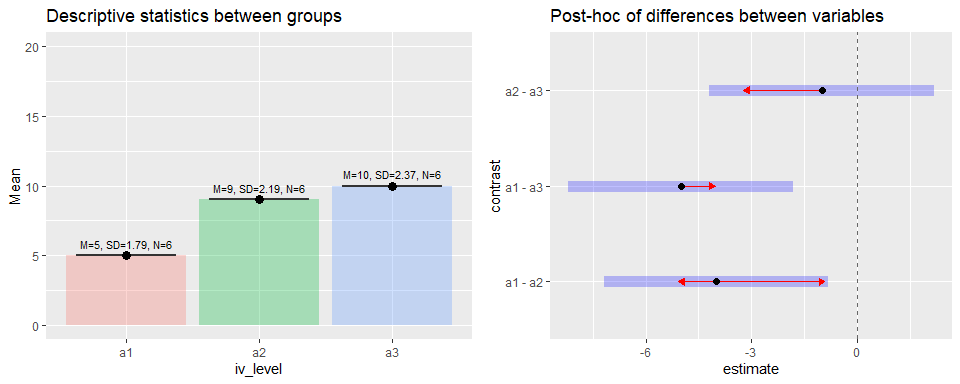
**APA결과 보고**  
report 패키지를 이용하여 기본적 보고 형식을 이해할 수 있다.

#결과보고   
  
aov(value ~ factor(name), df\_long) %>%   
 report\_aov()

## one-way analysis of variance result for dv[value]~[factor(name)], It was statistically significant(F(2, 15) = 9.26, p = 0).   
##

##### 4.2.5. jjstat::oneway\_aov oneway-ANOVA를 이용한 전체적인 과정은 jjstat::oneway\_aov()를 이용하여 분석할 수 있다. 일원분산 분석결과와 miginal 평균을 구하고, 사후분석을 쌍으로 비교하여 진행한다. 시각화하여 제공하는 것은 기술통계에 대한 bar그래프와 사후 분석의 신뢰구간의 유의성을 검정한 결과를 제공한다.

df1\_long <- jjstat::df1\_long  
jjstat::oneway\_aov(value ~ name, df1\_long)



## $ANOVA\_Table  
##   
##   
## term df sumsq meansq statistic p.value sig   
## ---------- --- ------ ------- ---------- -------- ----  
## name 2 84 42.00 9.26 0 \*\*   
## Residuals 15 68 4.53   
##   
## $Descriptives  
## # A tibble: 3 × 4  
## iv\_level Mean SD N  
## <chr> <dbl> <dbl> <dbl>  
## 1 a1 5 1.79 6  
## 2 a2 9 2.19 6  
## 3 a3 10 2.37 6  
##   
## $posthoc  
## # A tibble: 3 × 7  
## contrast estimate SE df t.ratio p.value sig   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 a1 - a2 -4 1.23 15 -3.25 0.0139 \*   
## 2 a1 - a3 -5 1.23 15 -4.07 0.00273 \*\*   
## 3 a2 - a3 -1 1.23 15 -0.813 0.701 ns   
##   
## $plot  
## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

## 4.3. Twoway ANOVA

## 4.4. 반복측정 분산 분석

## 4.5. MANOVA(다변량 분석)

## 4.6. ANCOVA 공분산 분석

## 4.7. Chi-square test

독립성 검정과 동질성 검정

# 5. 연관성을 이용한 분석

## 5.1. Correlation

상관분석은 두 연속변수가 서로 관련이 있는지 검정하는 통계분석이다. 상관분석을 통해서 두 변수가 어느정도의 관계가 있는지 파악할 수 있다. 상관계수의 값은 -1 ~ 1사이의 값을 가ㅣ져 1에 가까울 수록 관련성이 높은 것으로 본다. 값이 양수이면 정비례, 음수이면 반비례 관계를 나타낸다.

### 5.1.1. 상관계수

상관계수는 cor()을 이용하여 상관계수를 찾는다.  
두 개의 변수를 지정하면 두 변수 사이의 관계가 나타난다.

cor(mtcars$mpg, mtcars$hp)

## [1] -0.7761684

변수를 3개로 하면 3개의 변수가 나타난다. R의 베이스에서 변수 선택방벙을 이용하여 나타내었다.

#R base를 이용한 상관계수   
mtcars[, c("mpg","hp","wt")] %>%   
 cor()

## mpg hp wt  
## mpg 1.0000000 -0.7761684 -0.8676594  
## hp -0.7761684 1.0000000 0.6587479  
## wt -0.8676594 0.6587479 1.0000000

tidyverse이 방법을 사용하면 다음과 같다. 익숙해지면 이 방법이 더 편리한 것으로 생각 된다.

#tidyverse를 이용  
mtcars %>%  
 dplyr::select(mpg, hp, wt) %>%  
 cor()

## mpg hp wt  
## mpg 1.0000000 -0.7761684 -0.8676594  
## hp -0.7761684 1.0000000 0.6587479  
## wt -0.8676594 0.6587479 1.0000000

더 많은 변수로 구성된 경우를 상관행렬을 얻어보자.

#다변량으로 구성된 경우   
mtcars[,1:7] %>%   
 cor() %>%   
 round(3)

## mpg cyl disp hp drat wt qsec  
## mpg 1.000 -0.852 -0.848 -0.776 0.681 -0.868 0.419  
## cyl -0.852 1.000 0.902 0.832 -0.700 0.782 -0.591  
## disp -0.848 0.902 1.000 0.791 -0.710 0.888 -0.434  
## hp -0.776 0.832 0.791 1.000 -0.449 0.659 -0.708  
## drat 0.681 -0.700 -0.710 -0.449 1.000 -0.712 0.091  
## wt -0.868 0.782 0.888 0.659 -0.712 1.000 -0.175  
## qsec 0.419 -0.591 -0.434 -0.708 0.091 -0.175 1.000

많은 변수의 상관을 구할 수 있다. cor()함수는 상관계수만을 구해주는 것이며, 유의성 검정을 해주지는 않는다. 따라서, 유의성 검정은 별도로 해야 한다.

### 5.1.2. 상관계수의 유의성 검정

두 변수와의 상관관계는 통계적으로 유의한 것인지 알아보자.  
이때 사용하는 함수는 cor.test()함수이다.  
이 함수의 경우는 2개의 변수만을 입력해야 한다.

cor.test(변수1, 변수2)

이러한 형태로 입력하면 된다.

cor.test(mtcars$mpg, mtcars$hp)

##   
## Pearson's product-moment correlation  
##   
## data: mtcars$mpg and mtcars$hp  
## t = -6.7424, df = 30, p-value = 1.788e-07  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.8852686 -0.5860994  
## sample estimates:  
## cor   
## -0.7761684

mpg(연비)와 마력(hp)의 상관은 -0.78이고, 통게적으로 유의하다(p < .001).

The Pearson’s product-moment correlation between mtcarshp is negative, statistically significant, and very large (r = -0.78, 95% CI [-0.89, -0.59], t(30) = -6.74, p < .001).

이 결과는 report package를 이용하여 보고서를 작성한 것이다.

#report   
cor.test(mtcars$mpg, mtcars$hp) %>%   
 report::report()

## Effect sizes were labelled following Funder's (2019) recommendations.  
##   
## The Pearson's product-moment correlation between mtcars$mpg and mtcars$hp is  
## negative, statistically significant, and very large (r = -0.78, 95% CI [-0.89,  
## -0.59], t(30) = -6.74, p < .001)

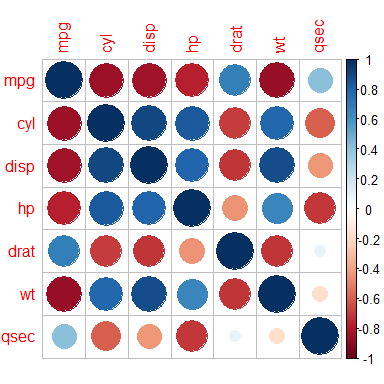
### 5.1.3. 상관계수의 시각화

보통 상관계수를 시각화하면 여러가지 면에서 빠른 파악이 된다. 간단하게 시각화하여 내용을 파악하는 방법과 응용하는 방법을 알아보자.

#### 5.1.3.1 히트맵을 그리는 corrplot

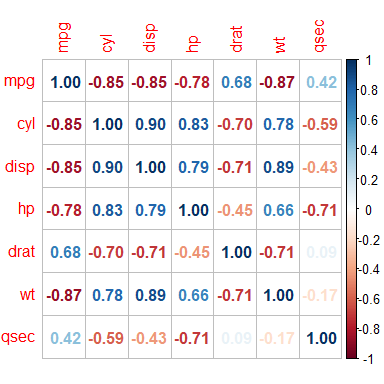
히트맵을 이욯하여 시각화하면 상관의 정도를 파악하는데 유리하다. corrplot패키지를 이용하여 시각화하면 화려한 모습을 나타나게 된다.

#heatmap   
library(corrplot)  
cor(mtcars[,1:7]) %>%  
 corrplot()



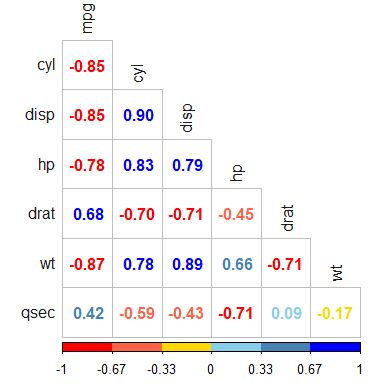
동그라미의 크기로 상관을 파악하고 색으로 판단이 된다. 그러나 정확한 수치(number)를 아는 것이 더 편리할 것이다.  
method = “number”로 하면 숫치로 나타나게 된다.

cor(mtcars[,1:7]) %>%  
 corrplot(method ="number")



이를 좀 더 다듬에서 행렬의 하위 부분에 나타내고, 대각선 부분의 1은 없앤 상태로 나타내 보자. 색은 음의 상관이면 “red”쪽으로 나타나고, 양의 상관은 “blue”쪽으로 나타나도록 설정하였다.

cor(mtcars[,1:7]) %>%   
 corrplot(method ="number", #숫자로 표시   
 type = "lower", #아래 행렬만 표현   
 tl.col = "gray10", #변수명 색 지정  
 diag = FALSE,  
 addCoef.col = "black",  
 col=c("red","tomato","gold","skyblue","steelblue","blue")  
 )



#### 5.1.3.2. rstatix패키지를 이용한 시각화(cor\_plot)

rstatix를 이용하는 것도 실은 corrplot을 이용한 것이나 마찬가지이다. 이를 기반으로 만들어진 것이라서 그런 것인데, rstatix패키지는는 실제 논문에서 적용하기 좋은 형태로 분석의 결과를 제시해주기 때문에 매우 의미있는 패키지이다.

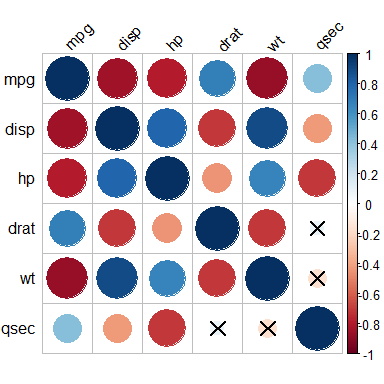
**cor\_mat()**  
먼저 데이터에서 cor\_mat()를 이용하여 상관행렬을 만들어내는 것이 중요하고 이를 이용하여 분석을 한다. 데이터를 저장하기 보다는 최대한 데이터에서 상관행렬을 만드는 것들을 보여주면서 설명하고 있다.

library(rstatix)  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat()

## # A tibble: 6 × 7  
## rowname mpg disp hp drat wt qsec  
## \* <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 mpg 1 -0.85 -0.78 0.68 -0.87 0.42   
## 2 disp -0.85 1 0.79 -0.71 0.89 -0.43   
## 3 hp -0.78 0.79 1 -0.45 0.66 -0.71   
## 4 drat 0.68 -0.71 -0.45 1 -0.71 0.091  
## 5 wt -0.87 0.89 0.66 -0.71 1 -0.17   
## 6 qsec 0.42 -0.43 -0.71 0.091 -0.17 1

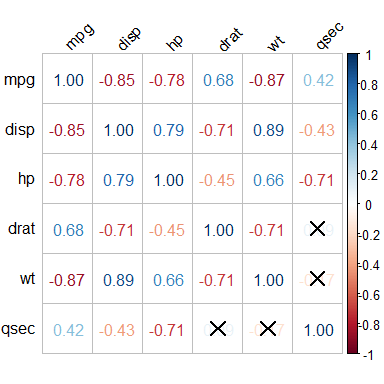
**(1)시각화**  
corrplot과의 차이점은 유의하지 않은 곳은 ’x’로 표시를 해주고 있다.  
사용은 cor\_plot()만 입력하면 되는데, corrplot과도 옵션이 동일하다.

#cor\_plot  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%   
 cor\_plot()



**(2)상관게수를 수치로 나타낸 것** 이번에는 상관계수를 수치로 나타낸 것이다. method = “number”를 설정하면된다. 또한 변수명등 심미적으로 더 보기좋게 나타나고 있다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%   
 cor\_plot(method = "number")



**(3)lower로 나타내기**  
rstatix::cor\_plot() 먼저 행렬을 lower부분만을 나타내려면 type =“lower”로 표시하면 된다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat()%>%   
 cor\_plot(method = "number", type = "lower")



#diag =FALSE옵션은 수행되지 않음

다른 방법으로는 pull\_lower\_triangel()함수를 이용하여 실제로 행렬을 하위부분만 나타낸 후에 cor\_plot()을 적용하는 방법도 있다.

먼저 pull\_lower\_triangel()을 적용하면 다음과 같다. 상관행렬을 보기 좋게 만들어준다.

# pull lower triangle and visualize  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 pull\_lower\_triangle()

## rowname mpg disp hp drat wt qsec  
## 1 mpg   
## 2 disp -0.85   
## 3 hp -0.78 0.79   
## 4 drat 0.68 -0.71 -0.45   
## 5 wt -0.87 0.89 0.66 -0.71   
## 6 qsec 0.42 -0.43 -0.71 0.091 -0.17

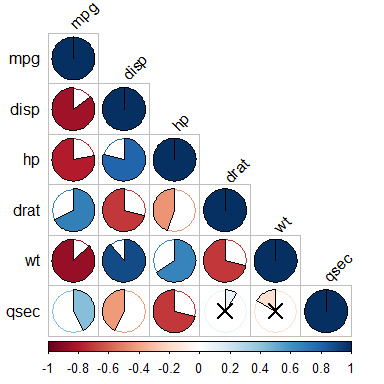
cor+plot()에서 diag=FALSE 옵션은 수행되지 않는다. 따라서, 행렬을 수정하여 상관행렬 히트맵을 그리는 방법이 가능하다. diagonal = TRUE 옵션을 켜서 위에 보여준 히트맵과 동일하게 설정하였다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 pull\_lower\_triangle(diagonal = T)%>% # lower행렬   
 cor\_plot(method="number") #시각화



**(4) pie를 이욯하여 나타내기**  
상관계수가 아닌 pie를 이용하여 나타낼 수도 있다. method =“pie”로 설정해주면 된다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 pull\_lower\_triangle(diagonal=T)%>%   
 cor\_plot(method="pie")



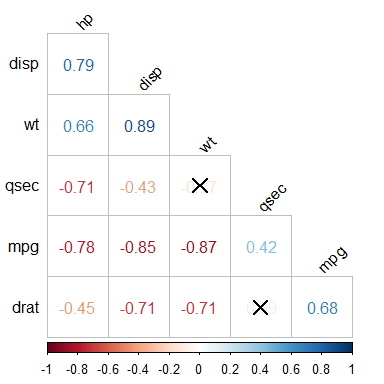
**(5)상관의 크기대로 정렬렬**  
만약에 변수를 정렬해서 나타내고자 한다면 cor\_reorder()를 활용한다.  
상관계수가 큰 것부터 나타났다.  
특별한 경우 상관계수를 정렬해서 보여주어야할 때 필요할 것으로 생각이 든다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 cor\_reorder() %>%  
 pull\_lower\_triangle()

## rowname hp disp wt qsec mpg drat  
## 1 hp   
## 2 disp 0.79   
## 3 wt 0.66 0.89   
## 4 qsec -0.71 -0.43 -0.17   
## 5 mpg -0.78 -0.85 -0.87 0.42   
## 6 drat -0.45 -0.71 -0.71 0.091 0.68

이렇게 상관의 크기대로 정렬하여 그려보자.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 cor\_reorder() %>%  
 pull\_lower\_triangle()%>%   
 cor\_plot(method="number")



### 5.1.4. 상관계수의 데이터 처리와 활용

1. 먼저 상관행렬을 보자.

여기까지는 cor()을 이용하는 것이나 큰 차이가 없다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%   
 rstatix::cor\_mat()

## # A tibble: 6 × 7  
## rowname mpg disp hp drat wt qsec  
## \* <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 mpg 1 -0.85 -0.78 0.68 -0.87 0.42   
## 2 disp -0.85 1 0.79 -0.71 0.89 -0.43   
## 3 hp -0.78 0.79 1 -0.45 0.66 -0.71   
## 4 drat 0.68 -0.71 -0.45 1 -0.71 0.091  
## 5 wt -0.87 0.89 0.66 -0.71 1 -0.17   
## 6 qsec 0.42 -0.43 -0.71 0.091 -0.17 1

1. 상관계수의 유의성 검정

상관계수의 유의성 검정을 위해 p값을 확인하여 검정해보자.  
각 위치에 적절하게 p값을 나타내고 있다.

# p value  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%   
 rstatix::cor\_mat() %>%   
 cor\_get\_pval()

## # A tibble: 6 × 7  
## rowname mpg disp hp drat wt qsec  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 mpg 0 9.38e-10 0.000000179 0.0000178 1.29e- 10 0.0171   
## 2 disp 9.38e-10 0 0.0000000714 0.00000528 1.22e- 11 0.0131   
## 3 hp 1.79e- 7 7.14e- 8 0 0.00999 4.15e- 5 0.00000577  
## 4 drat 1.78e- 5 5.28e- 6 0.00999 0 4.78e- 6 0.62   
## 5 wt 1.29e-10 1.22e-11 0.0000415 0.00000478 2.27e-236 0.339   
## 6 qsec 1.71e- 2 1.31e- 2 0.00000577 0.62 3.39e- 1 0

1. 롱포맷데이터로 변경

이번에는 이렇게 만들어진 상관행렬의 값을 롱포맷으로 만등어서 변수간 상관계수와 유의성을 나타내는 방법도 있다. 이경우는 cor\_gather()함수를 사용한다.  
일단은 모든 변수를 나타내도록 설정하여 만들었다.

# Gather/collapse correlation matrix into long format  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%   
 rstatix::cor\_mat() %>%   
 cor\_gather() %>% print(n=36)

## # A tibble: 36 × 4  
## var1 var2 cor p  
## <chr> <chr> <dbl> <dbl>  
## 1 mpg mpg 1 0   
## 2 disp mpg -0.85 9.38e- 10  
## 3 hp mpg -0.78 1.79e- 7  
## 4 drat mpg 0.68 1.78e- 5  
## 5 wt mpg -0.87 1.29e- 10  
## 6 qsec mpg 0.42 1.71e- 2  
## 7 mpg disp -0.85 9.38e- 10  
## 8 disp disp 1 0   
## 9 hp disp 0.79 7.14e- 8  
## 10 drat disp -0.71 5.28e- 6  
## 11 wt disp 0.89 1.22e- 11  
## 12 qsec disp -0.43 1.31e- 2  
## 13 mpg hp -0.78 1.79e- 7  
## 14 disp hp 0.79 7.14e- 8  
## 15 hp hp 1 0   
## 16 drat hp -0.45 9.99e- 3  
## 17 wt hp 0.66 4.15e- 5  
## 18 qsec hp -0.71 5.77e- 6  
## 19 mpg drat 0.68 1.78e- 5  
## 20 disp drat -0.71 5.28e- 6  
## 21 hp drat -0.45 9.99e- 3  
## 22 drat drat 1 0   
## 23 wt drat -0.71 4.78e- 6  
## 24 qsec drat 0.091 6.2 e- 1  
## 25 mpg wt -0.87 1.29e- 10  
## 26 disp wt 0.89 1.22e- 11  
## 27 hp wt 0.66 4.15e- 5  
## 28 drat wt -0.71 4.78e- 6  
## 29 wt wt 1 2.27e-236  
## 30 qsec wt -0.17 3.39e- 1  
## 31 mpg qsec 0.42 1.71e- 2  
## 32 disp qsec -0.43 1.31e- 2  
## 33 hp qsec -0.71 5.77e- 6  
## 34 drat qsec 0.091 6.2 e- 1  
## 35 wt qsec -0.17 3.39e- 1  
## 36 qsec qsec 1 0

여기에서 자세히보면 qsec 와 mpg의 상관계수가 0.42인데, 뒷부분에 보면 mpg와 qsec의 상관계수가 0.42로 중복해서 나타나고 있다. 이런 경우는 행렬에서 upper부분을 없애고 다시 cor\_gather()을 하면 된다.  
이때 upper부분을 없애버리거나 lower부분만을 남겨서 해보자. 방법은 두가지가 있는데, pull\_lower\_triangle()를 이용하여 lower부분만을 남기는 것이다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat()%>%   
 pull\_lower\_triangle()

## rowname mpg disp hp drat wt qsec  
## 1 mpg   
## 2 disp -0.85   
## 3 hp -0.78 0.79   
## 4 drat 0.68 -0.71 -0.45   
## 5 wt -0.87 0.89 0.66 -0.71   
## 6 qsec 0.42 -0.43 -0.71 0.091 -0.17

여기에 cor\_gather()적용하여 보자. 이렇게 하면 6개의 변수에 대하여 6C2 = 15개의 pairwise list가 나타나게된다.

mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat()%>%   
 pull\_lower\_triangle()%>%   
 cor\_gather() %>%  
 tibble()

## # A tibble: 15 × 4  
## var1 var2 cor p  
## <chr> <chr> <dbl> <dbl>  
## 1 disp mpg -0.85 9.38e-10  
## 2 hp mpg -0.78 1.79e- 7  
## 3 drat mpg 0.68 1.78e- 5  
## 4 wt mpg -0.87 1.29e-10  
## 5 qsec mpg 0.42 1.71e- 2  
## 6 hp disp 0.79 7.14e- 8  
## 7 drat disp -0.71 5.28e- 6  
## 8 wt disp 0.89 1.22e-11  
## 9 qsec disp -0.43 1.31e- 2  
## 10 drat hp -0.45 9.99e- 3  
## 11 wt hp 0.66 4.15e- 5  
## 12 qsec hp -0.71 5.77e- 6  
## 13 wt drat -0.71 4.78e- 6  
## 14 qsec drat 0.091 6.2 e- 1  
## 15 qsec wt -0.17 3.39e- 1

1. 다시 행렬로 만들기

이것을 다시 원래대로 돌려놓으려면 cor\_spread(value=“cor”)를 활용하면 된다. NA(결측)이 나타나게되면 replace\_upper\_triangle(by = ““)를 이용하여 NA를 공백으로 바꾸면 된다.

# spredd  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat()%>%   
 pull\_lower\_triangle()%>%   
 cor\_gather() %>%  
 cor\_spread(value="cor") %>%   
 replace\_upper\_triangle(by = "") #NA를 제거

## rowname mpg disp hp drat wt  
## 1 disp   
## 2 hp -0.78   
## 3 drat 0.68 -0.71   
## 4 wt -0.87 0.89 0.66   
## 5 qsec 0.42 -0.43 -0.71 0.091

1. 상관행렬에 유의성 표시

논문에 넣을 때 보통 p값을 근거로 하여 ’\*’로 유의성을 표시하는 경우가 많다. 이런 경우에는 cor\_mark\_significant()를 사용하면 이 부분을 좀 더 편리하게 사용할 수 있다.

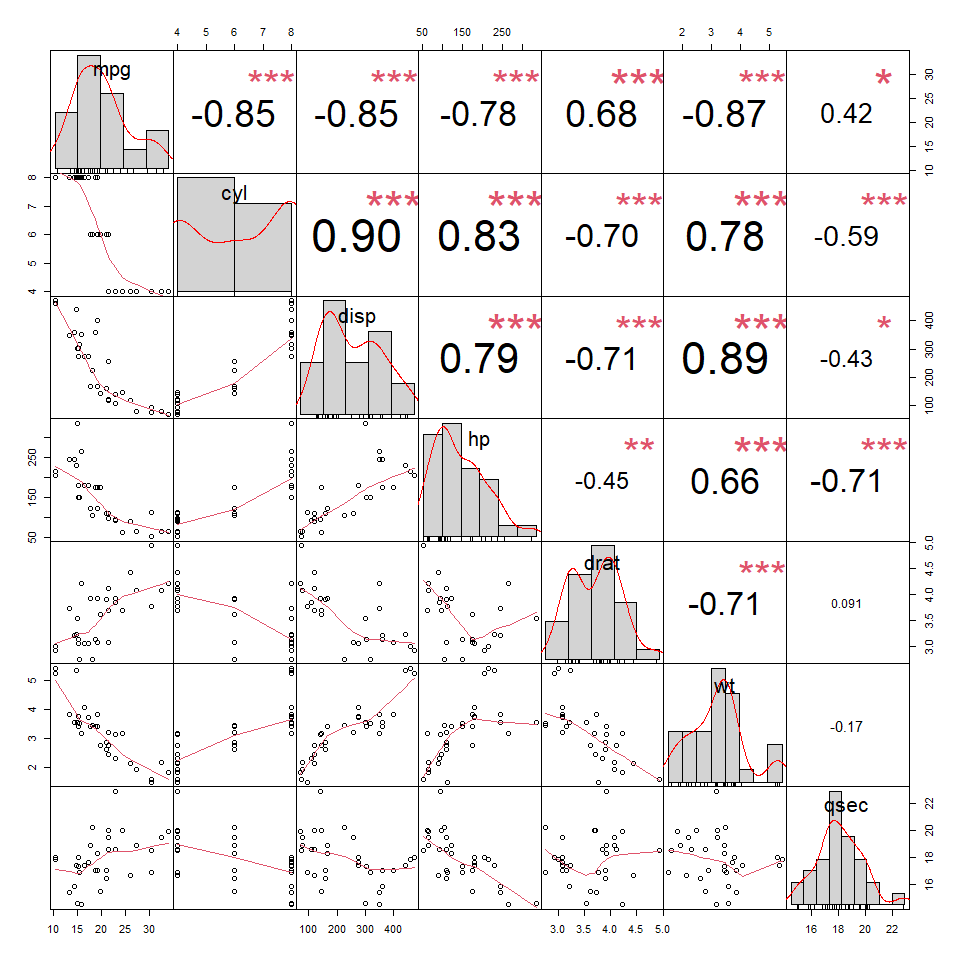
# Add Significance Levels To a Correlation Matrix  
mtcars %>%  
 dplyr::select(mpg, disp, hp, drat, wt, qsec) %>%  
 rstatix::cor\_mat() %>%  
 cor\_mark\_significant()

## rowname mpg disp hp drat wt qsec  
## 1 mpg   
## 2 disp -0.85\*\*\*\*   
## 3 hp -0.78\*\*\*\* 0.79\*\*\*\*   
## 4 drat 0.68\*\*\*\* -0.71\*\*\*\* -0.45\*\*   
## 5 wt -0.87\*\*\*\* 0.89\*\*\*\* 0.66\*\*\*\* -0.71\*\*\*\*   
## 6 qsec 0.42\* -0.43\* -0.71\*\*\*\* 0.091 -0.17

### 5.1.5. 응용패키지를 이용한 상관분석(데이터, 시각화 )

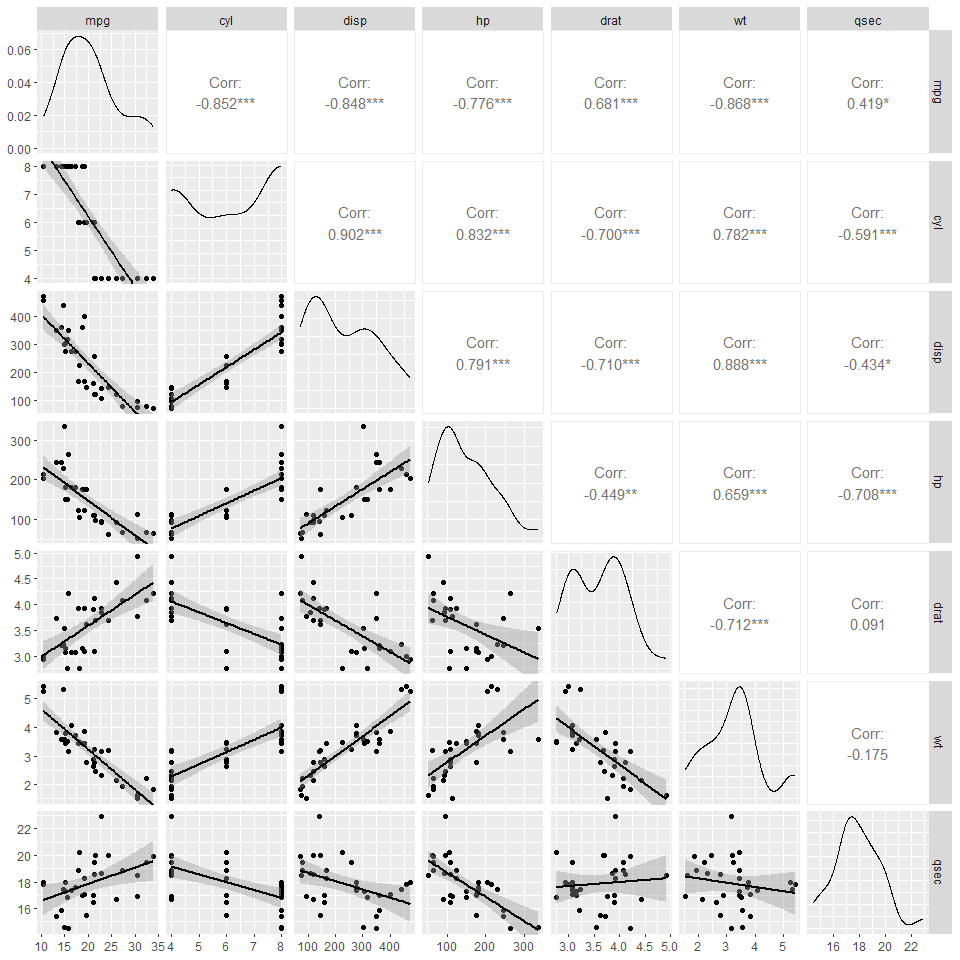
**PerformanceAnalytics::chart.Correlation()**  이 패키지는 옵션이 간단하고, 사용법도 편리하다. 하단에는 산점도와 회귀직성, 대각선에는 히스토그램과 밀도그래프, 상단에는 상관계와 유의성을 표시해준다.  
  
  
시각적으로도 보기 좋아 활용도가 매우 높은 패키지이다.

library(PerformanceAnalytics)  
  
 chart.Correlation(mtcars[,1:7])



**GGally:ggpairs()** 이 패키지도 논문등에 자주 사용되는 패키지이다. 옵션설정도 다양하고, 기본적인 것만을 사용해도 보기좋은 그래프와 상관분석 결과가 얻어진다.

library(GGally)  
ggpairs(mtcars[,1:7],  
 lower = list(continuous = "smooth"))



이러한 심미적인 그림을 제공하는 패키지는 논문을 집필할 때 활용도가 매우높다. 단점이라고 한다면 코드를 기억하기 쉽지 않다는 점이다. 따라서, 본인이 찾기좋은 곳에 메모해두거나 혹은 확인하기 좋은 곳에 적어두면 도움이 될 것이다.

## 5.2. Regression

### 5.2.1. OLS 회귀분석

### 5.2.2.로지스틱 회귀분석

# 6. 응용통계분석(1)

## 6.1. 경로 분석

## 6.2. 매개효과 분석

## 6.3. 조절효과 분석

## 6.4. Process Macro

# 7. 구조방정식 모형

# 8. PLS 구조 방정식 모형

# 9. 혼합모형(다층모형) 분석

# 10. 다변량 분석

## 10.1.주성분 분석

## 10.2.인자분석

## 10.3. 다차원 척도법

## 10.4. 대응일치분석

## 10.5. 군집 분석

## 10.6. 잠재프로파일 분석

## 10.7. 형상 분석

## 10.8. 소셜네트워크 분석

## 10.9. 연관분석

# Reference

1. Creswell, J. W. (2014). Research design: Qualitative, quantitative, and mixed methods approaches (4th ed.). Thousand Oaks, CA: Sage publications.
2. Hair, J. F., Jr., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2017). A primer on partial least squares structural equation modeling (PLS-SEM). Thousand Oaks, CA: Sage publications.
3. Levene, H. (1960). Robust tests for equality of variances. Contributions to probability and statistics, 278-292.
4. Fligner, M. A., & Killeen, T. J. (1976). Distribution of the F statistic for the Kruskal-Wallis one-way analysis of variance. Journal of the American Statistical Association, 71(353), 207-210.
5. Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. Psychometrika, 16(3), 297-334.
6. Tavakol, M., & Dennick, R. (2011). Making sense of Cronbach’s alpha. International journal of medical education, 2, 53.
7. Razali, N. M., & Wah, Y. B. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. Journal of statistical modeling and analytics, 2(1), 21-33.
8. Dawson, Robert J. MacG. (1995), The ‘Unusual Episode’ Data Revisited. Journal of Statistics Education, 3. <doi:10.1080/10691898.1995.11910499>.
9. British Board of Trade (1990), Report on the Loss of the ‘Titanic’ (S.S.). British Board of Trade Inquiry Report (reprint). Gloucester, UK: Allan Sutton Publishing.
10. D. N. Joanes and C. A. Gill (1998), Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183–189.
11. Bartlett, M. S. (1937). The statistical conception of mental factors. British Journal of Psychology, 28, 97–104. <doi:10.1111/j.2044-8295.1937.tb00863.x>.
12. Bartlett, M. S. (1938). Methods of estimating mental factors. Nature, 141, 609–610. <doi:10.1038/141246a0>.
13. Jöreskog, K. G. (1963). Statistical Estimation in Factor Analysis. Almqvist and Wicksell.
14. Lawley, D. N. and Maxwell, A. E. (1971). Factor Analysis as a Statistical Method. Second edition. Butterworths.  
      
    Thomson, G. H. (1951). The Factorial Analysis of Human Ability. London University Press.