

데이터 처리

- 활용 : 대기업은 데이터웨어하우스와 데이터마트를 통해 분석 데이터를 가져와서 사용한다. **신규 시스템이나 DW에 포함되지 못한 자료의 경우, 기존 운영시스템이나 스테이징영역과 ODS에서 데이터를 가져와서 DW에서 가져온 내용과 결합하여 활용할 수 있다.** 하지만 운영시스템에 직접 접근해 데이터를 활용하는 것은 매우 위험한 일이므로 거의 이루어지지 않고 있으며, 스테이징 영역의 데이터는 운영시스템에서 임시로 저장된 데이터이기 때문에 가급적이면 클린징 영역인 ODS에서 데이터의 전처리를 해서 DW나 DM과 결합하여 활용하는 것이 가장 이상적이다.

Staging 로딩

Staging은 소스시스템으로부터 제공 받은 데이터를 아무런 변화 없이 그대로 로딩하는 저장공간이다. Temporary성격의 공간으로 데이터 로딩의 첫 작업이 시작 DataStage를 사용하여 ETT 구현

ODS 로딩

Staging의 데이터를 데이터 정합성을 보장하는 형태로 ~~변환~~ ODS에 적재한다. ODS로딩 과정은 Staging 작업이 완료된 시점에서 Staging 테이블 안의 데이터를 대상으로 한다. 로딩 순서는 코드성 데이터, 마스터 데이터, 연동 데이터 순서이다. DataStage를 사용하여 ETT 구현

FACT생성

ODS 데이터를 통해 분석용 통계정보의 생성 통계스크립트를 사용하여 데이터 생성.

- 정형화된 패턴으로 처리 : 비정형 데이터나 소셜 데이터(비정형 데이터에 포함됨)는 정형화한 패턴으로 처리해야 한다.
- 1. 비정형 데이터(문자) : DBMS에 저장됐다가 텍스트 마이닝을 거쳐 데이터 마트와 통합한다.
- 2. 소셜 데이터(관계형 데이터) : DBMS에 저장되어 사회 신경망분석을 거쳐 분석결과 통계값이 데이터 마트와 통합되어 활용된다.
- 시각화(시각화 그래프) : 시각화는 가장 낮은 수준의 분석이지만, 잘 사용하면 복잡한 분석보다도 더 효율적이다. 대용량 데이터를 다루는 빅데이터 분석에서 시각화는 필수이다. **탐색적 분석을 할 때 시각화는 필수이다.** 탐색적 분석을 할 때 시각화는 필수이다. 소셜네트워크분석을 할 때 자주 활용된다.
- 공간분석 : 공간적 차원과 관련된 속성들을 시각화하는 분석이다. 지도 위에 관련 속성들을 생성하고 크기, 모양, 선 굵기 등으로 구분하여 인사이트를 얻는다.
- 탐색적 자료 분석(EDA) : 탐색적 분석은 변수의 분포 등을 시각화하여 분석하는 방법으로, 데이터의 특징과 내재하는 구조적 관계를 알아내기 위한 기법들의 통칭이다. **어떤 목적을 가지고 데이터를 확보해서 분석하는 것이 아닌, 이미 주어져 있는 데이터 자체 내에서 분석을 하는 방법이다.** 줄기 잎 그림, 상자 그림, 산점도가 EDA의 대표적인 예이다.
- EDA의 4가지 주제 : **저항성의 강조, 잔차 계산, 자료변수의 재표현, 그래프를 통한 현시성**
 1. **저항성의 강조** : 이상치, 결측치, 입력 오류에 대한 저항성이 크다.
 2. **잔차 계산** : 잔차는 각 값들이 흐름(대표하는 값[평균, 중앙값])으로부터 얼마나 벗어나 있는지를 나타내는 값이다. **(즉, 분산은 잔차에 해당한다.)**

3. **자료변수의 재표현** : 데이터분석과 해석을 단순화할 수 있도록 원래 변수를 적당한 척도(로그 변환, 제곱근 변환, 역수 변환)로 바꾸는 것을 말한다. 이와 같은 변환을 통하여 분포의 대칭성, 선형성, 분산의 안정성, 관련변수의 가법성 등 데이터 구조파악과 해석에 도움을 얻는 경우가 많다.

4. **그래프를 통한 현시성** : 시각화

- 데이터 마이닝 : 대표적인 고급 데이터 분석법으로, 대용량의 자료로부터 정보를 요약하고 미래에 대한 예측을 목표로 자료에 존재하는 관계, 패턴, 규칙 등을 탐색하고 이를 모형화함으로써, 이전에 알려지지 않은 유용한 지식을 추출하는 분석 방법이다.
- 데이터 마이닝에서 활용하는 평가 기준 : 정확도, 리프트, 디텍트 레이트
- 시뮬레이션에서 활용하는 평가 기준 : Throughput, Average Waiting Time, Average Queue Length, Time in System

R기초와 데이터 마트

• 기본적인 자료형, 자료구조, 함수

1. R 자료형 : numeric(정수형 & 실수형), character(문자 & 문자열), logical(boolean)
2. R 자료구조 : 스칼라, 벡터, 행렬, 어레이, 데이터 프레임, 리스트, 요인(factor, 순서형 변수와 명목형 변수를 표현하는 자료형)
3. 벡터, 행렬 : 서로 동일한 유형의 데이터들을 원소로 가질 수 있다.
4. 리스트, 데이터 프레임 : 서로 다른 유형의 데이터들을 원소로 가질 수 있다.
5. c함수로 생성되는 벡터는 열벡터이다.
6. 행렬 생성 예시 : `x = matrix(c(1,2,3,4,5,6), ncol = 3, ncol = 2)`
(행렬은 기본적으로 열을 우선으로 값들을 채우는 방향으로 입력된다.)
7. 데이터 프레임은 데이터 구조가 행과 열로 이루어지지만, 행렬이라기보다는 리스트라고 볼 수 있다.
8. `dim()` : 고차원 텐서를 생성할 때 사용하는 함수(ex : `dim(a) = c(2,3,2)`)
9. `View()` : 데이터 프레임 자료구조를 좀 더 깔끔히 볼 수 있다.
10. `str()` : 변수형과 가지고 있는 변수들의 이름, 자료 값 일부를 확인할 수 있다.
11. `'**'` 연산자는 요소별 곱을 실시하고, `'%*%'` 연산자는 행렬곱을 실시한다.
12. `t()` : 전치행렬을 구할 수 있다.
13. `solve()` : 역행렬을 구할 수 있다.
14. `skewness()` : 왜도를 구해준다. 0보다 크면 왼쪽으로 치우친 분포이다.(정규분포처럼 대칭이면 왜도는 0이다.)
15. `kurtosis()` : 첨도를 구해준다. 3보다 크면 정규분포보다 뾰족한 모양이다.
16. `summary()` : 데이터셋 변수 내용을 요약('최소값, Q1, 중앙값, 평균, Q3, 최댓값'을 출력함)
17. 두 벡터의 길이가 서로 배수관계에 있을 때는 사칙연산이 가능하다. (만약 두 벡터의 길이가 서로 배수관계가 아닐 때 사칙연산을 수행하면, 경고 메시지와 함께 결과가 출력된다.)
18. 'NA'는 결측치를 의미하기 때문에, 'NA'에 어떤 연산을 수행해도 그 결과는 'NA'이다.
19. 벡터의 요소 개수를 반환하는 함수는 `length()`이고, 문자열의 길이를 반환하는 함수는 `nchar()`이다.
20. 벡터에 특정 연산이나 함수를 적용시키면, 벡터 내 모든 요소에 해당 연산 또는 함수가 적용되는 것이다.
21. `seq(start, end)` : 수열을 반환한다. 해당 반환값은 vector 자료구조이다.
22. `rep(반복될 대상, 반복 횟수)` : 특정 값이 반복되어있는 vector 자료구조를 반환한다.
23. `% %` : 나머지 연산, `%/%` : 몫 연산
24. 특이한 사칙연산 방법 : "사칙연산 기호"(인자1, 인자2)

• 벡터 핸들링

1. `b[-2]`는 2번째 원소만 제외하고 나머지 원소를 전부 불러온다. 이점을 제외하고 벡터를 다루는 것은 파이썬의 리스트를 다루는 것과 거의 동일하다.
2. 벡터와 리스트 내 요소들은 이름을 가질 수 있다. `names`함수로 벡터와 리스트 내 요소들에 이름을 부여할 수 있다.
(`names(V) = c("가", "나", "다")`)
3. 벡터에 데이터 추가 : `V = c(V, newvalues)`
4. 벡터에 데이터 삽입 : `append(V, newvalues, after = n)`
5. 두 벡터에 `cbind()`와 `rbind()`를 적용할 수 있다. 이때 반환값은 행렬이다.
6. NA를 요소로 가지는 벡터에 집계 함수(aggregate function)를 적용하면, NA를 반환한다.
7. `x = c(1,2,3,4)`

x > 2 -> FALSE FALSE TRUE TRUE

- 문자열 핸들링

1. paste함수는 문자열을 연결시킬 수 있는 함수이다. **둘 이상의 벡터를 연결시키면, 기본적으로 벡터의 원소 대 원소(element wise)를 기준으로 하여 문자열이 연결된다.** sep 매개변수가 지정되지 않는다면 기본적으로 공백이 구분자가 된다.
(paste(c("statistics", "R") -> "statistics R", paste(c("a", "b"), c("x", "y")) -> "a x", "b y")
2. paste의 collapse 매개변수를 설정하면, 반환된 결과물들을 이어 붙일 수 있다.
(paste(c("가", "나"), c("다", "라"), collapse = " and ") -> "가 나 and 다 라")
3. substr함수는 문자열 내 부분문자열을 추출한다. 호출 형식은 'substr(character, start, end)'이다.
(substr("가나다", 1, 2) -> "가나")

- 데이터 프레임 핸들링

1. cbind, rbind 함수는 두 데이터 프레임을 결합하는 함수이다. 해당 함수들로 기존의 행렬에 열과 행을 추가할 수 있다.(cbind는 행의 개수가 동일해야 하고, rbind는 열의 개수와 열의 이름이 동일해야 한다.)
2. **조건에 맞는 행, 열 조회 : subset(df, subset = (조건 기입), select = 열 지정)**
3. 데이터 프레임 인덱싱

```
> df
  이름 나이
1 최우석  26
2 최민규  30
> df[c(1,2), c("이름", "나이")]
  이름 나이
1 최우석  26
2 최민규  30
> df[1:2, c("이름", "나이")]
  이름 나이
1 최우석  26
2 최민규  30
```

4. 두 데이터 프레임을 inner join처럼 합치기 : merge(df1, df2, by = 기준이 되는 열 지정)
5. 이름으로 열 제거 : subset(df, select = -열이름)[열이름에 따옴표 안 붙이는게 핵심!]
6. 열 이름 바꾸기 : colnames(df) = c("이름1", "이름2", "이름3", ...)

- 자료형 변환

1. `as.character()` : 문자열로 변환
2. `as.complex()` : 복소수로 변환
3. `as.numeric()` or `as.double()` : 실수로 변환
4. `as.integer()` : 정수로 변환
5. `as.logical()` : 논리형으로 변환

- 자료구조 변환

1. `as.data.frame()` : 데이터 프레임으로 변환
2. `as.list()` : 리스트로 변환
3. `as.matrix()` : 행렬로 변환
4. `as.vector()` : 벡터로 변환(행렬에 `as.vector()`를 적용하면, 열 방향으로 1열 부터 차례로 원소를 나열하는 벡터가 생성된다.)

- apply함수

1. Python의 `map` 함수와 똑같은 기능을 한다.
2. `apply()` : 행렬의 행 또는 열 방향으로 특정 함수를 적용한다.(`apply` 함수의 단점은 `array`만 입력받을 수 있다는 것이다.)
3. `matrix`를 입력받는 `apply()` : `apply(matrix, 1 또는 2, function)`(1은 행 단위 연산이고 2는 열 단위 연산이다.)
4. `lapply()` : 결과값을 `list`로 반환한다.(`vector` 또는 `list`를 입력받는다. 이 점이 `apply` 함수의 단점을 극복한 것이라 볼 수 있다.)
5. `sapply()` : 결과값을 `vector` 또는 `matrix`로 반환한다.(`vector` 또는 `list`를 입력받는다. 이 점이 `apply` 함수의 단점을 극복한 것이라 볼 수 있다.)
6. 데이터 프레임에 `apply` 적용 : `lapply(df, func)` & `sapply(df, func)`(함수가 적용되는 변수에는 오류없이 적용되지만, 함수가 적용되지 않는 변수에는 `NA`값을 반환한다.), `apply(df, func)`(모든 변수가 동일한 유형일 때만 활용 가능하다.)
7. `tapply()` : 데이터 프레임 객체에 자주 사용되고, 범주형 변수의 `categori`로 데이터들을 묶은 후 지정한 함수를 적용한다.(`tapply(data, index(범주형 변수), function)`)

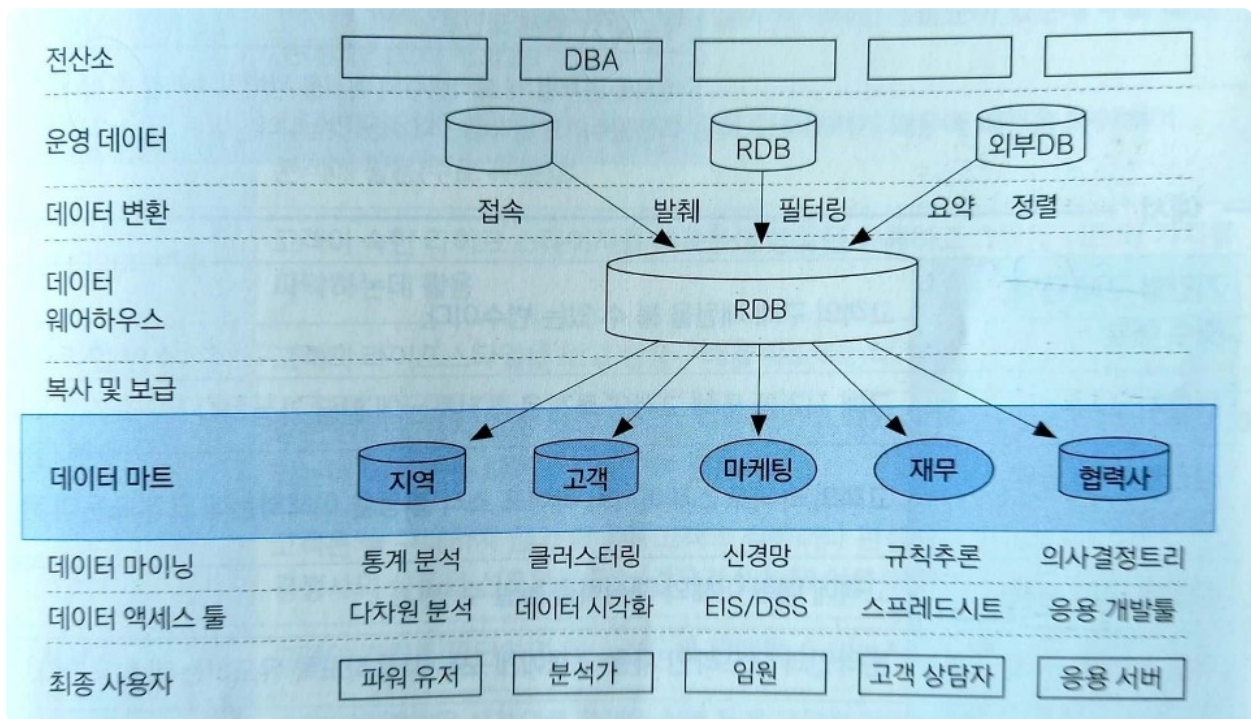
- 날짜 다루기

1. `Sys.Date()` : 현재 날짜를 반환한다.
2. `as.Date()` : 문자열을 날짜로 변환하는 함수이다. `as.Date()`는 문자열이 'yyyy-mm-dd'형식일 것이라 가정한다. 다른 날짜 형식을 띠는 문자열을 변환하려면, 'format' 매개변수를 통해 입력된 날짜의 형식을 지정해줘야 한다.
(`as.Date("01/13/2018", format = "%m/%d/%Y")`)

R reshape을 이용한 데이터 마트 개발

• 데이터 마트

1. 데이터 웨어하우스와 사용자 사이의 중간층에 위치한 것으로, 하나의 주제 또는 하나의 부서 중심의 데이터 웨어하우스라고 할 수 있다.
2. 데이터 마트 내 대부분의 데이터는 데이터웨어하우스로부터 복제되지만, 자체적으로 수집될 수도 있으며, 관계형 데이터 베이스나 다차원 데이터 베이스를 이용하여 구축한다.
3. **CRM 관련 업무 중에서 핵심 - 고객 데이터 마트 구축**
4. 동일한 데이터 셋을 활용할 경우 최신 분석기법들을 이용하면 분석가의 역량에서는 분석 효과가 크게 차이가 나지 않기 때문에, **데이터 마트를 어떻게 구축하느냐에 따라 분석 효과는 크게 차이 난다.**
5. 데이터 마트를 만들 때 가장 중요한 데이터들은 데이터 웨어하우스로부터 받아오는 데이터이다. **받아온 데이터를 처리과정을 통해 분석에 적절하게 활용할 수 있는 자료로 변환을 해야 한다. 이렇게 만들어진 변수는 요약변수와 파생변수로 나뉜다.**



X 데이터 베이스 → 데이터 웨어하우스 → 데이터 마트.

— '데이터 마트'에 존재

• 요약 변수

1. 수집된 정보를 분석에 맞게 **종합(aggregate)**한 변수이다.
2. 데이터 마트에서 가장 기본적인 변수로 **총 구매 금액, 횟수, 구매여부** 등 데이터 분석을 위해 만들어지는 변수이다.
3. 많은 모델을 공통으로 사용될 수 있어 재할용성이 높다.
4. **합계, 횟수와 같이 간단한 구조**이므로 자동화하여 상황에 맞게 또는 일반적인 자동화 프로그램으로 구축 가능하다.
5. 요약변수의 단점은 얼마 이상이면 구매하더라도 기준값의 의미 해석이 애매할 수 있다. 이러한 경우, 연속형 변수를 그룹핑해 사용하는 것이 좋다.
6. **기간별 구매 금액과 구매 횟수 여부, 위클리 쇼퍼, 상품별 구매 금액과 횟수 여부, 상품별 구매 순서, 유통 채널별 구매 금액, 단어 빈도(횟수), 초기 행동변수(고객 가입 또는 첫 거래 초기 1개월 간 거래 패턴에 대한 변수), 트렌드 변수(추이값을 나타내는 변수)** 등이 이에 해당된다.

• 파생변수 `데이터 파생'에 존재

1. **사용자(분석자)가 특정 조건을 만족하거나 특정 함수에 의해 값을 만들어 의미를 부여한 변수**이다.
2. 매우 주관적일 수 있으므로, 논리적 타당성을 갖추어 개발해야 한다.
3. 세분화, 고객행동 예측, 캠페인 반응 예측에 매우 잘 활용한다.
4. 파생변수는 상황에 따라 특정 상황에만 유7의미하지 않게 대표성을 나타나게 할 필요가 있다.

• 데이터 테이블

1. '데이터 테이블'자료구조는 '데이터 프레임'자료구조와 거의 똑같이 생긴 구조를 가지고 있지만(데이터 프레임에 적용할 수 있는 함수를 데이터 테이블에도 적용할 수 있음), **데이터 프레임에 없는 key값을 가지고 있다.**
2. 그래서 데이터 프레임 자료구조보다 월등히 빠른 데이터 연산 처리 속도를 자랑한다.
3. `setkey()`함수는 검색의 속도를 빠르게 하기 위해서, 특정한 변수를 기준으로 미리 정렬하고 정렬된 변수를 key로 지정한다.

```
# setkey(데이터명, key로 설정할 변수명)
```

```
setkey(waterstand_dt, FID)
```

 key 변수 설정.


```
# 조건(제약)으로 데이터를 선택
```

```
# 데이터명[J(조건), ]
```

```
# waterstand_dt의 FID 값이 1001인 데이터를 선택하기
```

```
# 이런 식으로 하면 검색 속도가 data.frame의 약 20배(?) 이상 빠르다고 한다.
```

```
waterstand_dt[J(1001), ]
```

 key로 지정된 변수 내 data value를 입력
→ 해당 data value를 가진 모든 data를
'data table' 자료구조로 출력!!

• 변수의 중요도

1. 개요 : 변수 선택법과 유사한 개념으로 모델을 생성하여 사용된 변수의 중요도를 살피는 과정이다.
(‘변수 선택법’은 선형 회귀분석에서 등장하는 개념이고, 변수 중요도는 범주형 변수(종속 변수)를 활용한 classification에서 등장하는 개념이다.)
2. klaR 패키지 : 특정 변수가 주어졌을 때, 클래스가 어떻게 분류되는지에 대한 어려움을 계산해주고 그래픽으로 결과를 보여주는 기능을 한다.
3. KlaR의 greedy.wilks() 함수를 이용하여, 종속변수에 가장 영향력을 미치는 변수를 wilks lambda를 활용하여 변수의 중요도를 정리(wilks lambda = within variance / total variance)

• R에서의 데이터 EDA(탐색적 분석)

1. 본격적인 데이터 분석에 앞서 전체적으로 데이터의 특징을 파악하고 데이터를 다양한 각도로 접근하다.
2. summary()를 이용하여 데이터의 기초통계량을 확인한다.

• 결측값 인식

1. 결측값은 **NA**, 9999999, ‘(공백)’, Unknown, Not Answer 등으로 표현되는 것으로, 결측값을 처리하기 위해서 시간을 많이 사용하는 것은 비효율적이다.
2. 결측값 자체의 의미가 존재하는 경우도 있다. 예를 들면, 쇼핑몰 가입자 중 특정 거래 자체가 존재하지 않는 경우와 인구통계학적 데이터에서 아주 부자이거나 아주 가난한 경우 자신의 정보를 잘 채워 넣지 않기 때문에 가입자의 특성을 유추하여 활용할 수 있다.
3. 결측값 처리는 전체 작업속도에 많은 영향을 준다.
4. 특정 함수 내 ‘na.rm’ 매개변수를 통해서, 결측치를 제거 한 후 해당 함수의 연산을 실시할 수 있다.

• 결측값 처리 방법 - 단순 대치법(대치[imputation] : 다른 것으로 바꾸어 놓음)

*completes analysis
평균대치법, 단순복구대치법.*

1. **completes analysis** : 결측값이 존재하는 레코드를 삭제한다.
2. **평균대치법** : 관측 또는 실험을 통해 얻어진 데이터의 평균으로 대치한다. 비조건부 평균 대치법(관측데이터의 평균으로 대치)과 조건부 평균 대치법(회귀분석을 활용한 대치)이 있다.

예2) 조건부 평균 대치법(Regression Imputation)

Y_1	Y_2	Y_3	Y_3
10	15	20	20
12	25	30	30
15	35	40	40
25	48	57	57
30	49	60	60
35	55	65	65
37	47	70	70
40	60	? ₁	76.89
42	65	? ₂	81.67
50	70	? ₃	92.39

sol> $Y_{3i} = \beta_0 + \beta_1 Y_{1i} + \beta_2 Y_{2i} + \varepsilon_i, i=1, \dots, 7$
 $\rightarrow \beta_0 = 3.69, \beta_1 = 0.099, \beta_2 = 0.56$
 $?_1 = 3.69 + 40 \cdot 0.99 + 60 \cdot 0.56 = 76.89$

3. **단순 확률 대체법** : 평균대치법에서 추정량 표준 오차의 과소 추정문제를 보완하고자 고안된 방법으로, Hot-dec, Nearest-Neighbour 방법 등이 있다. 기본적 아이디어는 평균대치법에서 관측된 자료를 토대로 추정된 통계량으로 결측값을 대체할 때, 어떤 적절한 확률값을 부여 한 후 대체하는 방법이다.

- 결측값 처리 방법 - **다중 대체법**

대치 · 분석 · 결합

1. 단순대치법을 한번하지 않고 m번의 대체를 통해 m개의 가상적 완전 자료를 만드는 방법이다.
2. **1단계 : 대체**(단순대치법을 여러번 수행) -> **2단계 : 분석** -> **3단계 : 결합**

- R에서 결측값 처리

1. **complete.cases()** : 데이터 내 레코드에 결측값이 있으면 FALSE, 없으면 TRUE를 반환 *결측값이 없으면 TRUE를 반환*
2. **is.na()** : 결측값을 NA로 인식하여, 결측값이 있으면 TRUE, 없으면 FALSE를 반환 *결측값이 있으면 TRUE를 반환*
3. DMwR 패키지의 **centralimputation()** : NA 값을 가운데 값으로 대체한다. 연속형 변수는 중앙값으로 범주형 변수는 최빈값으로 대체한다.
4. DMwR 패키지의 **knnimputation()** : NA 값을 k최근 이웃 분류 알고리즘을 사용하여 대체하는 것으로, k개 주변 이웃까지의 거리를 고려하여 가장 평균한 값을 사용
5. **Amelia** 패키지의 **amelia()** : time-series-cross-sectional dataset(여러 국가에서 매년 측정된 자료)을 활용

- 이상값(극단값, Outlier)이란?

1. **의도하지 않게 잘못 입력한 경우(bad data)**
2. **의도하지 않게 입력되었으나, 분석 목적에 부합되지 않아 제거해야 하는 경우(bad data)**
3. 의도하지 않은 현상이지만 분석에 포함해야 하는 경우
4. 의도된 이상값(fraud)인 경우
5. 이상값을 꼭 제거해야 하는 것은 아니기 때문에, 분석의 목적이나 종류에 따라 적절한 판단이 필요하다.

- 이상값의 인식 방법

1. **ESD(Extreme Studentized Deviation)** : 통상 평균으로부터 표준편차의 3배가 되는 점을 기준으로 이상치를 정의한다.
2. **‘기하평균 - 2.5 * 표준편차 < data < 기하평균 + 2.5 * 표준편차’**의 범위에서 벗어난 데이터를 이상치로 정의한다.

↑ 기하평균으로부터 표준편차의 2.5배가 되는 점을 기준으로

The Geometric Mean Formula

n : number of terms (x) that are multiplied

$$\sqrt[n]{X_1 \cdot X_2 \cdot X_3 \dots X_n}$$

3. 상자그림에서 fence 밖에 있는 값을 이상치로 정의한다.

- 이상값 절단 방법

1. 기하평균을 이용한 제거 : `geo_mean()`
2. 하단, 상단 % 이용한 제거 : 10% 절단(상하위 5%에 해당하는 데이터 제거)

- 이상값 조정 방법(이상값을 절단 하는 방법보다는 이상값을 조정하는 방법을 이용하는 것이 데이터 손실율도 적고 설명력도 높아진다.)

1. 상한값과 하한값을 벗어난 값들을 하한, 상한값으로 바꾸어 활용하는 방법이다.(즉, 상자그림에서 이상값들을 최대, 최소 수염값으로 바꾸는 방법이다.)

- R 그래픽 기능

1. `plot(x, y)` : x 변수와 y 변수에 대한 산점도 그래프를 그린다.
2. `pairs(data.frame)` : multipanel display에 변수 간의 산점도 그래프를 그린다.('panel = panel.smooth'를 추가하면, 산점도에 데이터 포인트들의 추세를 나타내는 smooth curve를 추가할 수 있다.)

