ADP 정리 노트

2025-09-27

확률과 통계

통계학

- **불확실한 상황** 하에서 데이터에 근거하여 **과학적인 의사결정**을 도출하기 위한 이론과 방법의 체계
- **모집단**으로 부터 수집된 **데이터**(sample)를 기반으로 모집단의 **특성을 추론**하는 것을 목표로 한다.

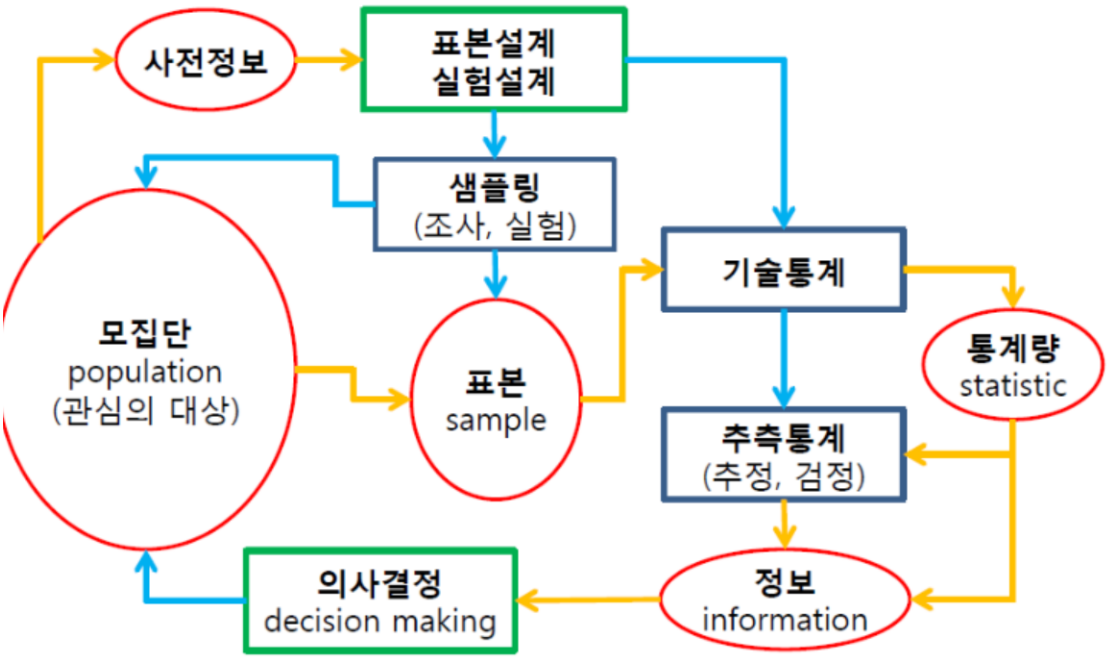


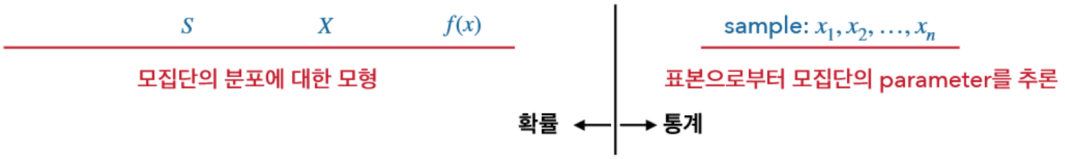
Figure 1: 통계적 의사결정 과정

확률

- 고전적 의미: 표본공간에서 특정 사건이 차지하는 비율
- 통계적 의미: 특정 사건이 발생하는 **상대도수의 극한**
  - 각 원소의 발생 가능성이 동일하지 않아도 무한한 반복을 통해 수렴하는 값을 구할 수 있다.

확률 분포 정의 단계

- 확률실험 → 표본공간 → 확률변수 → 확률분포 → 표본의 분포 → 통계적 추론 (추정, 검정)



- **Experiment(확률실험)**: 동일한 조건에서 독립적으로 반복할 수 있는 실험이나 관측
- **Sample space(표본공간)**: 모든 simple event의 집합
- **Event(사건)**: 실험에서 발생하는 결과 (부분 집합)
- **Simple event(단순사건)**: 원소가 하나인 사건
- **확률 변수**: 확률실험의 결과를 수치로 나타낸 변수

확률 분포

이산 확률 분포

이산 표본 공간, 연속 표본공간에서 정의 가능포

- **베르누이 시행**: 각 시행은 서로 **독립적**이고, 실패와 성공 **두 가지 결과만 존재**.
  - 단 **모집단의 크기가 충분히 크고**, **표본(시행)의 크기가 충분히 작다면 비복원 추출**에서도 **유효**
  - 평균: p
  - 분산: p(1-p)
- **이항 분포**: n번의 독립적인 **베르누이 시행**을 수행하여 성공 횟수를 측정
  - X ~ B(n, p), $f(x) = \binom{n}{x}p^x(1-p)^{n-x}$
  - 평균: np
  - 분산: np(1-p)
  - n이 매우 크고, p가 매우 작을 때, **포아송 분포로 근사**할 수 있다. (λ = np)
- **음이항 분포**
  - 정의: n번의 독립적인 **베르누이 시행**을 수행하여 k번 성공하고, r번 실패한 경우 (n = k + r)
    1. r번의 실패가 나오기 전까지, 성공한 횟수 x
       * X ~ NB(r, p), $f(x) = \binom{x+r-1}{x}p^x(1-p)^r$
       * 평균: $\frac{rp}{1-p}$
       * 분산: $\frac{rp}{(1-p)^2}$
    2. r번의 실패가 나오기 전까지, 시행한 횟수 x
       * 4번에서 성공을 실패로 바꿈
    3. k번의 성공이 나오기 전까지, 실패한 횟수 x
       * 1번에서 실패를 성공으로 바꿈
    4. k번의 성공이 나오기 전까지, 시행한 횟수 x
       * $f(x) = \binom{x-1}{k-1}p^k(1-p)^{x-k}$
       * k가 1일 때 기하분포와 동일
    5. n번의 시행 횟수에서, k번 성공 또는 r번 실패한 경우: 이항분포
- **기하 분포**:
  - 정의:
    1. 성공 확률이 p인 **베르누이 시행**에서 첫 성공까지의 시행 횟수
       * X ~ G(p), $f(x) = (1-p)^{x-1}p, x = 1, 2, 3, ...$

        * 평균: $\frac{1}{p}$
        * 분산: $\frac{1-p}{p^2}$
    2. 성공 확률이 p인 **베르누이 시행**에서 첫 성공까지의 실패 횟수
        * X ~ G(p), $f(x) = (1-p)^x p, x = 0, 1, 2, ...$
        * 평균: $\frac{1-p}{p}$
        * 분산: $\frac{1-p}{p^2}$
  - **비기억 특성**: $P(X > n + k | X > n) = P(X > k)$
- **초기하 분포**: **베르누이 시행이 아닌 시행**에서 성공하는 횟수
  - X ~ H(n, N, k), $f(x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$
  - 평균: $\frac{nK}{N}$
  - 분산: $\frac{nK(N-K)(N-n)}{N^2(N-1)}$
- **포아송 분포**: **임의의 기간**동안 **어떤 사건이 간헐적**으로 발생할 때, 동일한 길이의 기간동안 실제 사건이 발생하는 횟수
  - X ~ Poisson(λ), $f(x) = \frac{e^{-\lambda}\lambda^x}{x!}, \lambda > 0$
  - 평균: λ
  - 분산: λ

연속 확률 분포

연속 표본 공간에서 정의 가능

- **균일 분포**
  - $f(x) = \frac{1}{b-a}, a \leq x \leq b$
  - 평균: $\frac{a+b}{2}$
  - 분산: $\frac{(b-a)^2}{12}$
- **정규 분포**
  - $X + Y \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$
  - 선형 변환: $Y = aX + b \sim N(a\mu + b, a^2\sigma^2)$
- **t 분포**
  - 자유도가 커질수록 표준 정규분포에 근사함.
  - $\frac{Z}{\sqrt{V/n}} \sim t(n)$, Z: 표준정규분포, V: 자유도가 n인 카이제곱분포
- **f 분포**
  - $F = \frac{X_1/\nu_1}{X_2/\nu_2}, X_1 \sim \chi^2(\nu_1), X_2 \sim \chi^2(\nu_2)$, X1과 X2는 서로 독립
- **감마 분포**
  - α: 분포의 형태 결정, θ: 분포의 크기 결정
  - 평균: αθ
  - 분산: $\alpha\theta^2$
  - **카이제곱 분포**: α = v/2, θ = 2 인 감마분포
    * $Z_i \sim N(0, 1)$일 때, $Z_1^2 + Z_2^2 + ... + Z_n^2 \sim \chi^2(n)$

    - ⋆ X_i가 서로 독립이고, 자유도가 ν_i인 카이제곱분포를 따른다면, $X_1 + X_2 + ... + X_n \sim x^2(\nu_1 + \nu_2 + ... + \nu_n)$
    - ⋆ 자유도가 커질수록 기댓값을 중심으로 모이고, 대칭에 가까워진다.
  - **지수 분포**: α = 1, θ = 1/λ 인 감마분포
    - ⋆ $X\ Exp(\lambda = \frac{1}{\theta})$, f(x) = $\lambda e^{-\lambda x}, x > 0$
    - ⋆ θ: 평균 사건 발생 간격, λ: 단위 시간당 사건 발생 횟수
    - ⋆ 포아송 분포에서 사건 발생 간격의 분포
    - ⋆ $\sum_{i=1}^{n} X_i \sim \Gamma(n, \theta), \theta = 1/\lambda$
    - ⋆ 비기억 특성을 가진다: $p(X > s + t|X > s) = p(X > t) = e^{-\lambda t}$
    - ⋆ 독립적으로 동일한 지수분포를 따르는 확률변수 n개의 합은 $\alpha = n, \theta = \frac{1}{\lambda}$인 감마분포를 따른다.

다변량 분포

- **다항 분포**: n번의 독립적인 **베르누이 시행**을 수행하여 k개의 범주로 분류
  - X ~ M(n, p1, p2, ..., pk), $f(x_1, x_2, ..., x_k) = \frac{n!}{x_1!x_2!...x_k!}p_1^{x_1}p_2^{x_2}...p_k^{x_k}$
  - 평균: $[np_1, np_2, ..., np_k]$
  - 분산: $[np_1(1 - p_1), np_2(1 - p_2), ..., np_k(1 - p_k)]$
  - 공분산: $-np_ip_j(i \neq j)$
  - 독립인 변수의 갯수는 k-1개 (k개의 사건)

샘플링

분포의 동질성 검정

- 연속형
  - 이표본 검정: 콜모고로프-스미르노프 검정 사용
  - 일표본 검정:
    - ⋆ 정규분포, 지수분포: 앤더슨-달링 검정 사용
    - ⋆ 그 외: 몬테카를로 방법 사용
- 이산형
  - 이표본: 카이제곱 독립성 검정
  - 일표본: 카이제곱 동질성 검정

표본의 분포

- 샘플링에 따라 통계량이 다른 값을 가질 수 있다. 따라서 통계량의 분포를 이용한 통계적 추론이 가능하다.
- 통계량: 표본의 특성을 나타내는 값

- 추정량: 아래의 조건을 만족하는 통계량
  - 불편성: 추정량의 기대값이 추정하려는 모수와 같아야 한다.
  - 효율성: 분산이 작아야 한다. 표본의 갯수가 많아질수록 분산이 작아져야 한다.

표본 평균의 분포

- 모집단의 분포와 관계없이, 모집단의 평균이 μ이고, 분산이 σ^2이면, $\bar{X}$의 평균은 μ이고, 분산은 σ^2/n인 정규분포를 따른다.
  - 단 모집단의 분포에 따라 표본의 크기가 충분히 커야함. (중심극한정리[1])
- 만약 모집단의 분산을 모를 경우, σ를 s로 대체하여, t분포를 따르는 표본 평균의 분포를 구할 수 있다.
  - 단 이때는 **모집단이 정규분포를 따라야 한다.**

표본 분산의 분포

- **정규 모집단으로 부터 나온 표본**의 분산 S에 대하여, $\frac{(n-1)S^2}{\sigma^2}$은 자유도가 n-1인 카이제곱 분포를 따른다.
  - 모집단이 정규분포를 따르지 않을 경우, 비모수적인 방법을 사용해야 한다.
- 두 정규 모집단으로부터 계산되는 표본분산의 비율은 f-분포를 따른다.

추정

- 통계적 추론: **모집단에서 추출된 표본**의 **통계량**으로부터 **모수**를 추론하는 것
  - 추정
    - 점추정
    - 구간추정
  - 가설 검정

점 추정

- 불편성
  - $E(\hat{\theta}) = \theta$
  - bias = $E(\hat{\theta}) - \theta$
    - 보통 sample size가 커질수록 bias는 0에 수렴
  - $\bar{X}, X_n$은 μ의 불편추정량이다.
- 최소분산
  - $Var(\bar{X})$가 $Var(X_n)$보다 분산이 작아서 더 좋은 추정량

---

[1] 모집단의 분포와 상관 없이, 표본의 평균은 정규분포에 수렴한다는 정리. 이항분포의 경우, P(X=c) ~ P(c - 0.5 < X < c + 0.5)로 근사 가능하다는 라플라스의 정리를 일반화한 것

- $MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = Var(\hat{\theta}) + bias^2$
  - ⋆ 큰 오차에 더 큰 페널티를 주기 위해 제곱

	모수 θ	표본크기	추정량 $\hat{\theta}$	기대값 $E(\hat{\theta})$	표준오차 $\sigma_{\hat{\theta}}$
모평균	μ	n	$\overline{X}$	μ	$\frac{\sigma}{\sqrt{n}}$
모비율	p	n	$\hat{p} = X/n$	p	$\sqrt{\frac{p(1-p)}{n}}$
모평균차이	$\mu_1 - \mu_2$	n_1, n_2	$\overline{X}_1 - \overline{X}_2$	$\mu_1 - \mu_2$	$\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$
모비율차이	$p_1 - p_2$	n_1, n_2	$\hat{p}_1 - \hat{p}_2$	$p_1 - p_2$	$\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}$

Figure 2: 대표적인 불편추정량

- **전부 중심극한의정리를 적용**할 수 있다. (비율은 0과 1의 평균이므로)
- 모평균, 모비율의 차이는 서로 독립이라는 가정이 필요하다.

구간 추정

- α: 유의수준
- 1 - α: 신뢰수준[2]
- (θ_L, θ_U) = (1 - α) × 100% 신뢰구간

1. (θ_L, θ_U) 이 충분이 높은 가능성으로 미지의 모수 θ를 포함해야 한다
2. 구간이 충분히 좁아야 한다
   - 표준 정규분포에서 0을 중심으로 대칭일 때 길이가 짧다.
   - 고로 신뢰구간이 대칭임

표본의 크기 결정

특정 오차 아래로 하는 표본의 수 구하는 법

- 그냥 표본오차가 목표 오차보다 작게 하는 값을 구하면 됨.
- **모비율**을 모를 때는 일단 **0.5로 보수적으로 놓고 계산**

모분산 추정

- 카이제곱 분포는 가장 짧은 신뢰구간을 구하기 쉽지 않음
  - 그냥 쉽게 구하기 위해 $(x^2_{\alpha/2}, x^2_{1-\alpha/2})$를 사용

[2] 샘플링을 무한히 반복했을 때, **이들의 신뢰 구간 중 95%의 구간이 실제 모수를 포함**한다. 즉, 구간이 확률변수이다.

- 모분산의 신뢰구간: $\left(\frac{(n-1)s^2}{x^2_{(1-\alpha)/2}(n-1)}, \frac{(n-1)s^2}{x^2_{\alpha/2}(n-1)}\right)$
- 표본의 수가 적을수록, 카이제곱 분포의 신뢰구간은 더 길어진다.

가설 검정

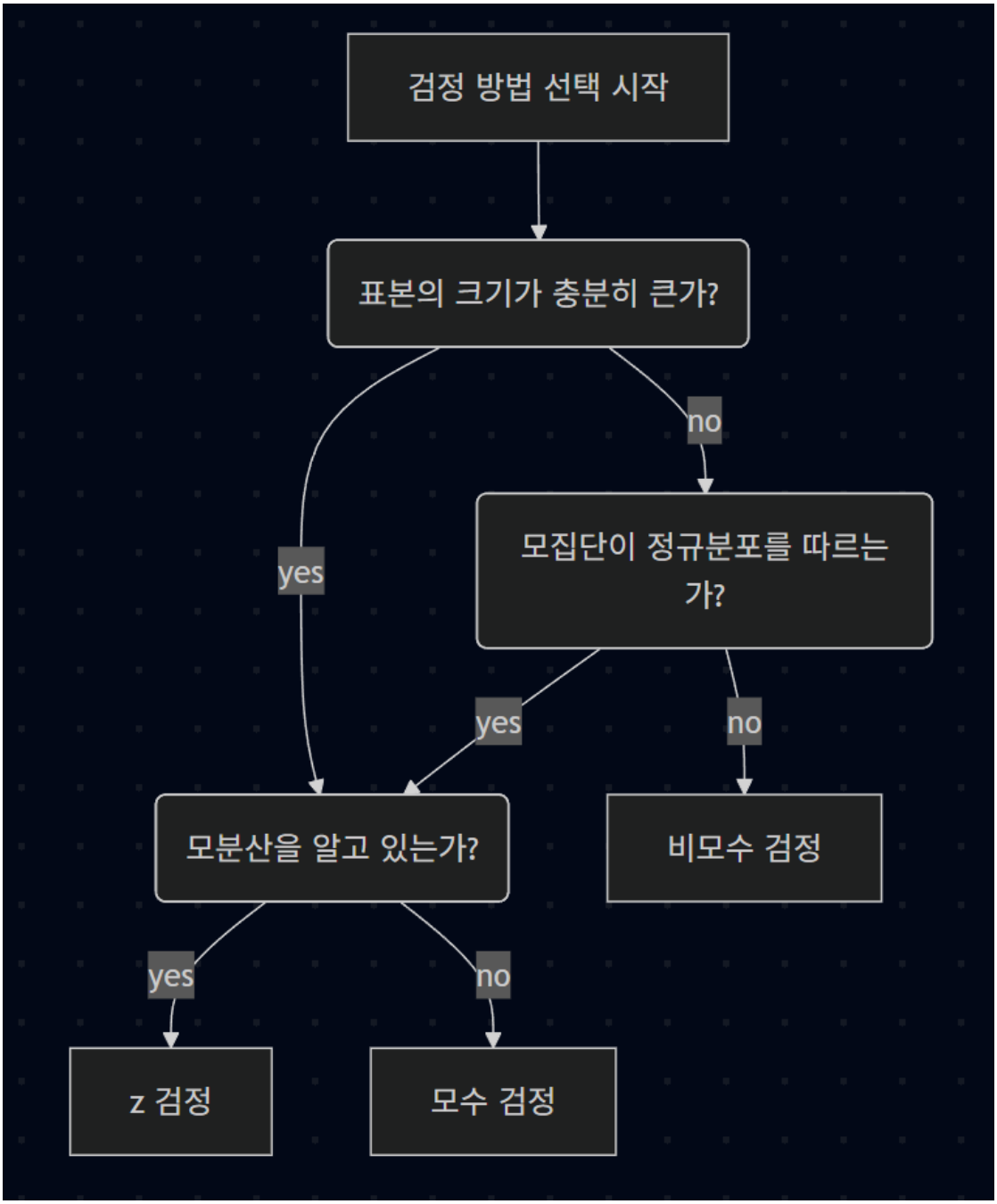


Figure 3: 검정 방법 선택 기준

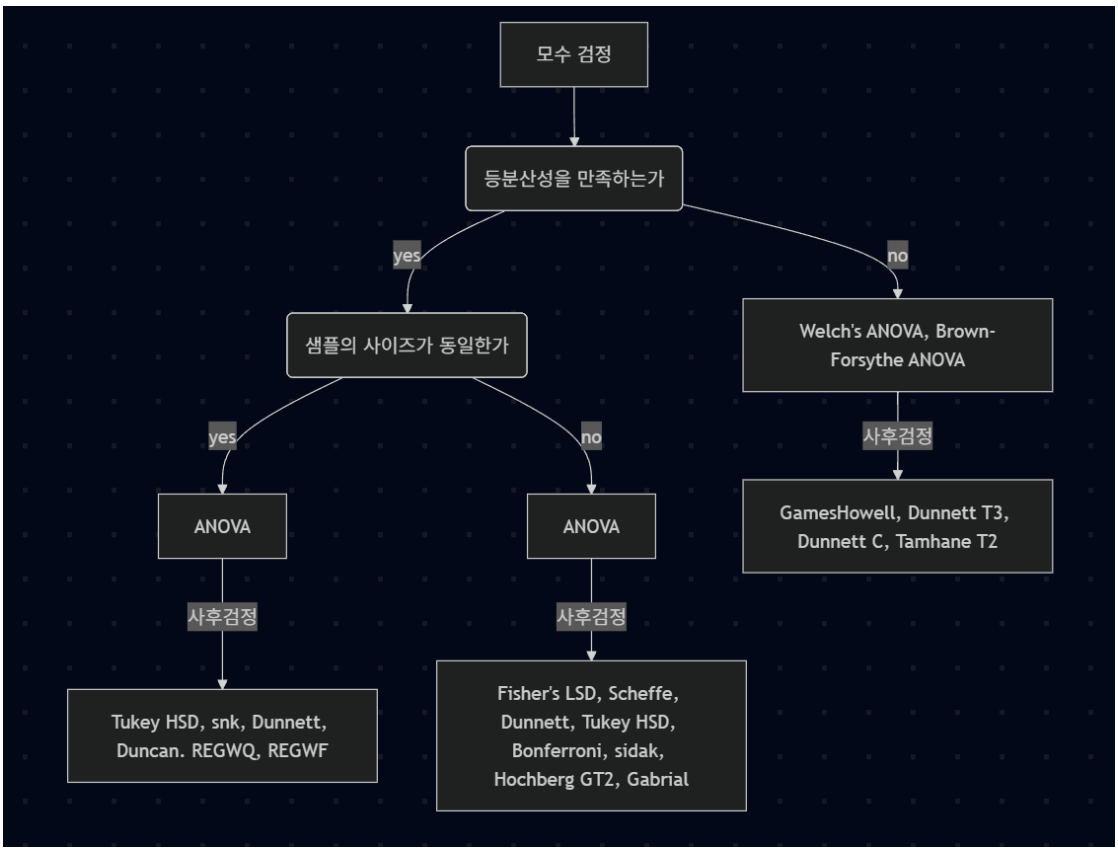


Figure 4: 모수 검정 방법 선택 기준

- 관측치 간에 독립이 아닌 경우(시간: 자기 상관이 존재, 공간: 패널, 계층, …), 각 케이스에 맞는 모형을 사용해야 함.

정규성 검정

- 표본이 정규분포를 따르는지 검정.
- 따르지 않더라도 중심극한정리에 의해 **표본의 크기가 충분히 크면** 모수 검정을 사용할 수 있다.
- **shapiro wilk 검정**: 표본의 크기가 3-5000개인 데이터에 사용. 동일한 값이 많은 경우 성능이 떨어질 수 있음
  - H0: 데이터가 정규분포를 따른다.
  - H1: 데이터가 정규분포를 따르지 않는다.
- **jarque-Bera**: 대표본에 사용.
  - H0: 데이터가 정규분포를 따른다.
  - H1: 데이터가 정규분포를 따르지 않는다.
- **Q-Q plot**: x축이 이론적 분위수, y축이 표본 분위수

Listing 0.1 `normality test`

```
from scipy.stats import shapiro, jarque_bera, zscore, probplot

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
stat, p = shapiro(data)

print(f"Shapiro-Wilk Test: stat={stat:.3f}, p={p:.3f}")

stat, p = jarque_bera(data)
print(f"Jarque-Bera Test: stat={stat:.3f}, p={p:.3f}")

import matplotlib.pyplot as plt
import seaborn as sns

zdata = zscore(data)
fig, ax = plt.subplots(1, 2, figsize=(10, 3))

(osm, odr), (slope, intercept, r)) = probplot(zdata, plot=ax[0])
ax[0].set_title("Q-Q Plot")

sns.histplot(data, kde=True, ax=ax[1])
ax[1].set_title("Histogram")

plt.show()
```

등분산성 검정

- **Barlett 검정**: **정규성을 만족하는 경우에만** 사용 가능
  - H0: $\sigma_1^2 = \sigma_2^2 = \ldots = \sigma_k^2$
  - H1: $\sigma_i \neq \sigma_j$ for some i, j
- **Levene 검정**: **정규성을 만족하지 않는 경우에도** 사용 가능
  - H0: $\sigma_1^2 = \sigma_2^2 = \ldots = \sigma_k^2$
  - H1: $\sigma_i \neq \sigma_j$ for some i, j

Listing 0.2 `equal variance test`

```
from scipy.stats import bartlett, levene

group1 = [1, 2, 3, 4, 5]
group2 = [2, 3, 4, 5, 6]
group3 = [3, 4, 5, 6, 7]

stat, p = bartlett(group1, group2, group3)
print(f"Bartlett's Test: stat={stat:.3f}, p={p:.3f}")

stat, p = levene(group1, group2, group3)
print(f"Levene's Test: stat={stat:.3f}, p={p:.3f}")
```

분산분석

표본	개수	비모수 검정		모수 검정	
		서열척도	명목척도	등분산성 o	등분산성 x
단일 표본	1개	부호검정, 부호순위검정	적합성 검정, Run 검정	일표본 t-검정	
대응 표본	2개	부호검정, 부호순위검정	McNemar 검정	대응표본 t-검정	
	K개	Friedman 검정	Cochran Q 검정	반복측정 분산분석	
독립 표본	2개	순위합 검정, 만위트니U 검정	독립성 검정, 동질성 검정	독립표본 t-검정	Welch's t-검정
	K개	Kruskal-Wallis 검정		일원배치 분산분석	Welch's ANOVA

회귀분석

회귀분석을 위한 가정

- **선형성**: 종속변수와 독립변수 간의 관계는 선형이다.
- **정규성**: 종속변수 **잔차들의 분포**는 정규분포이다.
- **등분산성**: 종속변수 **잔차들의 분포**는 동일한 분산을 갖는다.
- **독립성**: 모든 **잔차값**은 서로 독립이다.

검정

- 영향치 처리: 레버러지(변수 내 다른 관측치들이랑 떨어진 정도) * 잔차
  - Cook's distance
  - DFBETAS
  - DFFITS
  - Leverage
- 다중공산성 검정
- 선형성 검정
  - 잔차도 검정: 잔차 vs 예측값 산점도
  - 잔차도가 어떠한 패턴도 보이지 않아야 한다.
- 정규성 검정: 분산분석이랑 동일
- 등분산성 검정:
  - 잔차도 검정: 잔차 vs 예측값 산점도
  - 잔차도가 일정한 폭을 가져야 한다.
- 독립성은 검정은 연구자 주관에 판단하는 것이 일반적이라고 한다..
  - 더빈-왓슨 검정을 사용할 수도 있지만, 1차 자기상관만 검정 가능하다.

Listing 0.3 `linear regression test`

```
impot statsmodels.api as sm

Xc = sm.add_constant(X)
model = sm.OLS(y, Xc).fit()
resid = model.resid

print(model.summary())
```

전처리

1. 범주형 변수 처리:
   - 더미 변수화: 기준이 되는 범주를 하나 정하고, 나머지 범주를 0과 1로 표현

    - 각 범주의 회귀계수는 기준 범주와의 차이를 의미
2. 이상치 / 영향점: 관측값 제거
3. 선형성 위반: 독립변수 변환, GAM
4. 정규성 / 등분산성 위반: 종속변수 변환, GLM, GAM
5. 다중공산성 위반: 다중공산성 파트 참고
    - 혹은 변수 선택법을 사용

- 가정 만족할 때까지 검정, 전처리 계속 반복

변수 변환

λ	−2	−1	−0.5	0*	0.5	1	2
변환	$\frac{1}{x^2}$	$\frac{1}{x}$	$\frac{1}{\sqrt{x}}$	$\ln(x)$	$\sqrt{x}$	x (무변환)	x^2

Figure 5: 회귀 모델 수정 λ

TABLE 5.1 Useful Variance-Stabilizing Transformations

Relationship of σ^2 to $E(y)$	Transfonnation
$\sigma^2 \propto$ constant	$y' = y$ (no transformation)
$\sigma^2 \propto E(y)$	$y' = \sqrt{y}$ (square root; Poisson data)
$\sigma^2 \propto E(y)[1 - E(y)]$	$y' = \sin^{-1}(\sqrt{y})$ (arcsin; binomial proportions $0 \le y_i \le 1$)
$\sigma^2 \propto [E(y)]^2$	$y' = \ln(y)$(log)
$\sigma^2 \propto [E(y)]^3$	$y' = y^{-1/2}$ (reciprocal square root)
$\sigma^2 \propto [E(y)]^4$	$y' = y^{-1}$(reciprocal)

Figure 6: 경험적인 적절한 λ

- 최적의 λ는 최대 우도 추정법으로 구할 수 있다.
- 변수 변환은 예측력은 높일 수 있지만, 해석이 어려워질 수 있다.
- 일반적으로 box tidwell 검정을 사용하여 변환을 수행할 수 있지만 파이썬에서는 제공하는 라이브러리가 없다.
    - 아마 양수 변수만 사용 가능한 단점과 다른 방법들이 많아서 그런 것 같다.
    - 통계적 검정은 아니지만 box cox 변환을 사용하여 최적의 λ를 찾을 수 있다.

Listing 0.4 `box cox transformation`

```
from scipy.stats import boxcox

y_transformed, best_lambda = boxcox(y)
print(f"Best lambda: {best_lambda:.3f}")
```

변수 선택법

- 전진 선택법
- 후진 선택법
- 단계적 선택법
- 최적조합 선택법: 모든 조합 다 해봄
- 기준
  - R2, Adj R2
  - AIC(Akaike Information Criterion): 모델에 변수를 추가할 수록 불이익을 주는 오차 측정법
  - BIC(Bayesian Information Criterion): 변수 추가에 더 강한 불이익을 줌
  - Mallows' Cp

종류

- 최대한 단순한 모델을 사용하는 것이 일반화에 좋다.

단순

- 그냥 선형 회귀

규제 선형 회귀

- 지나치게 많은 독립변수를 갖는 모델에 패널티를 부과하는 방식으로 간명한 모델을 만듦
- 독립변수에 대한 scaling이 선행되어야 함 (큰 변수에만 과하게 패널티가 부과될 수 있어서)
  - 일반적으로는 scale을 하든 안하든 r square에 차이가 없다.

1. 릿지회귀
   - $\Sigma(y_i - \hat{y_i})^2 + \lambda\Sigma\beta_j^2$
   - 회귀계수 절댓값을 0에 가깝게 함
   - 하지만 0으로 만들지는 않음
   - 작은 데이터셋에서는 선형 회귀보다 점수가 더 좋지만, 데이터가 충분히 많아지면 성능이 비슷해짐.
   - 회귀계수가 모두 비슷한 크기를 가질 때 라쏘보다 성능이 좋음
2. 라쏘회귀:
   - $\Sigma(y_i - \hat{y_i})^2 + \lambda\Sigma|\beta_j|$
   - 회귀계수를 0으로 만들 수 있음
   - 변수 선택 효과
   - 릿지보다 해석이 쉬움
   - 일부 독립계수가 매우 큰 경우 릿지회귀보다 성능이 좋음
3. 엘라스틱넷 회귀

- $\Sigma(y_i - \hat{y_i})^2 + \lambda_1 \Sigma|\beta_j| + \lambda_2 \Sigma \beta_j^2$
- 릿지와 라쏘의 장점을 모두 가짐
- 변수 선택 효과도 있고, 회귀계수를 0에 가깝게 만듦
- 독립변수 간에 상관관계가 있을 때, 그룹으로 선택하는 경향이 있음

일반화 선형 회귀(GLM)

- 종속변수가 이항분포를 따르거나 포아송 분포를 따르는 경우
  - 이항분포: 평균이 np, 분산이 np(1-p), 즉 평균과 분산 사이에 관계가 존재하여 등분산성 가정을 만족하기 어렵다.
  - 포아송 분포: 평균과 분산이 같아서 등분산성 가정을 만족하기 어렵다.
  - 따라서 위와 같은 경우에 종속변수에 적절한 함수를 적용하여 등분산성 가정을 만족시킨다.
- 종속변수가 범주형일 경우: logistic 회귀 분석
  - $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$
  - $p = \frac{1}{1+e^{-z}}$
  - 오즈: $\frac{p}{1-p} = e^z$
  - 오즈비: 독립변수 k 단위 변화에 따른 오즈(양성 vs 음성)의 변화 비율
    - $(e^{\beta_k})^k$
- 종속변수가 count 데이터일 경우: 포아송 회귀 분석

이상치에 강한 선형 회귀

- Robust 회귀
- Quantile 회귀

상관분석

상관계수

- 두 변수 간의 선형적 관계의 강도와 방향을 나타내는 척도

질적 변수

- **스피어만** 상관계수: 서열척도 vs 서열척도. 확률분포에 대한 가정 필요 없음.
- **켄달의 타우**: 서열척도 vs 서열척도.
  - 둘 중 하나가 연속형이여도 스피어만, 켄달의 타우 중 하나를 사용.
  - 샘플이 적거나, 이상치, 동점이 많은 경우 켄달의 타우를 주로 사용.
  - **두 변수의 크기는 같아야함**.
- phi 계수: 명목척도 vs 명목척도
  - 두 변인 모두 level이 2개일 때 사용
  - 두 변수를 0과 1로 바꾼 후 pearson 상관계수 계산
- 크래머 v: 명목척도 vs 명목척도.
  - 적어도 하나의 변수가 3개 이상의 level을 가지면 사용
  - 범위는 0~1. 0.2 이하면 서로 연관성이 약하고, 0.6 이상이면 서로 연관성이 높음.

양적 변수

- 피어슨 상관계수: 연속형 vs 연속형
  - 두 변수 간의 선형적 관계를 측정
  - -1 ~ 1 사이의 값
  - 0: 독립, 1: 완전한 양의 상관관계, -1: 완전한 음의 상관관계
  - 이상치에 민감

다중 공산성

- 독립 변수 집합 X 가 본래는 서로 독립적(직교적)이어야 한다는 가정에서 벗어난 정도.

문제

- 회귀계수 **추정치의 분산이 커지고**, 결과적으로 **추정이 매우 불안정**해짐.
- 특정 독립변수가 설명하는 **효과**를 다른 변수와 **구분하기 어려워짐**.
- 예측력이나 설명력이 높게 나올 수 있지만, **모형의 구조적 해석은 신뢰할 수 어려움**.
- 물론 이런 문제들은 **중요한 변수**에 영향을 줄 때 생김.

해결 방법

잘못된 예시

1. 그냥 둔다
2. 직교화
    - PCA(주성분 분석)나 요인분석 등을 사용하여 독립변수들을 직교화
    - 요인이 해석 불가능한 경우가 아니면 좋지 않음
3. 규칙 기반 접근: 상관계수가 0.8이 넘는걸 제거하거나, 종속변수의 상관계수보다 높은 변수 제거
    - 직관에만 의존하고, 잘못된 결론을 낳을 수 있음

좋은 예시

다중 공산성의 문제를 세부적으로 진단하고 각각에 대해 해결 방법을 다음과 같이 제시한다.

1. 전 변수 집합 대상:
    - 독립변수의 전체 차원이 부족한 경우
    - 표본을 더 모으거나 새로운 변수를 도입
2. 개별 변수의 계수 추정이 불안정한 경우(표본 오차가 큰 경우)
    - 특정 변수가 다른 변수들의 선형 결합으로 표현될 수 있는 경우
    - **VIF가 10**을 넘을 경우
    - 덜 중요하다면 제거
    - 중요하다면 변수에 대한 독립적 정보 보강(세분화, …)
3. 두 변수의 상관계수가 높은 경우
    - 둘의 관계를 설명하는 제 3의 변수 도입(요인 분석 등)
    - 둘 중 하나를 제거

Listing 0.5 `multicollinearity test`

```
cor = df.corr()
cond_num = np.linalg.cond(cor)
print("Condition Number:", cond_num)
```

- **30을 초과**하면 다중공선성이 높다고 판단한다.
- 선형 종속 가능성을 봄.
- scaling이 선행되어야 함.

Listing 0.6 `VIF test`

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

def check_vif(X, y):
    X = sm.add_constant(X)
    model = sm.OLS(y, X)
    model.fit()
    vif_df = pd.DataFrame(columns=['feature', 'VIF'])
    for i in range(1, len(model.exog_names)):
        vif_df.loc[i, 'feature'] = model.exog_names[i]
        vif_df.loc[i, 'VIF'] = variance_inflation_factor(model.exog, i)
    return vif_df.sort_values(by='VIF', ascending=False)

print(check_vif(X, y))
```

Listing 0.7 `correlation heatmap`

```
import seaborn as sns

fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(cor, annot=True, ax=ax)
```

시계열 분석

베이지안 통계

머신러닝

전처리

결측치 처리

- 결측치가 발생하는 원인
  - **무작위 결측**(Missing Completely at Random, MCAR): 결측치가 발생할 확률이 관측된 데이터와 무관
    - 이 경우, 결측치를 제거하거나 대체해도 무방
  - **조건부 무작위 결측**(Missing at Random, MAR): 결측치가 발생할 확률이 관측된 데이터에만 의존
    - 이 경우, 관측된 데이터로 결측치를 예측하여 대체하는 방법이 유용
  - **비무작위 결측**(Missing Not at Random, MNAR): 결측치가 발생할 확률이 관측된 데이터와도 관련이 없음
    - 통계적 방법으로 해결하기 어려움
- 결측치 처리 방법
  - 제거: 결측치가 적거나 무작위 결측일 때 사용
  - 대치(대체):
    - 일반적인 방법
      - 시계열 데이터 o: 이전 값, 이후 값, 선형 보간법
      - 시계열 데이터 x: 평균, 중앙값, 최빈값
    - 고급 대치법(과적합 발생 가능성 유의)
      - KNN 대치: 유사한 관측치의 값을 사용하여 결측치를 대체. 결측치가 없는 데이터로 예측
      - 다중 대치: 결측치를 여러 번 대체하여 불확실성을 반영

Listing 0.8 `imputer`

```
from sklearn.impute import SimpleImputer, KNNImputer

# KNN
imputer = KNNImputer(n_neighbors=5)
imputed_X = imputer.fit_transform(X)

# MICE
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

imputer = IterativeImputer(max_iter=10)
imputed_X = imputer.fit_transform(X)
```

이상치 처리

불균형 처리

ensemble

- stacking:
  - 여러 모델을 학습시킨 후, 각 모델의 예측 결과를 입력으로 하는 메타 모델을 학습시킨다.
  - 메타 모델은 다른 모델들의 예측 결과를 종합하여 최종 예측을 수행한다.
- voting, averaging:
  - 여러 모델을 학습시킨 후, 각 모델의 예측 결과를 투표(voting)하거나 평균(averaging)하여 최종 예측을 수행한다.
  - 분류 문제에서는 다수결 투표(hard voting) 또는 확률 평균(soft voting)을 사용하고, 회귀 문제에서는 단순 평균 또는 가중 평균을 사용한다.
- bagging
  - vs cross validation:
    - cross validation은 이미 생성된 모델을 검증하기 위한 방법. 모델 구축 방법은 아님
    - bagging은 분산을 줄이기 위해 사용함
  - **bagging의 voting, averaging은 unsupervised learning**
- boosting: sequentially 학습
  - 이전 모델의 오차를 보완하는 방식으로 학습한다.

Listing 0.9 `voting`

```
#| eval: false

from sklearn.ensemble import VotingClassifier, VotingRegressor

voting_lf = VotingClassifier(estimators=[
    ('lr', LogisticRegression()),
    ('dt', DecisionTreeClassifier()),
    ('rf', RandomForestClassifier())
], voting='soft') # or hard


voting_rf = VotingRegressor(estimators=[
    ('lr', LinearRegression()),
    ('dt', DecisionTreeRegressor()),
    ('rf', RandomForestRegressor())
], weights=[1, 1, 2])


voting_lf.fit(X_train, y_train)
voting_rf.fit(X_train, y_train)
voting_lf.predict(X_test)
voting_rf.predict(X_test)
```

군집분석

전제 조건

1. scalability
2. 다양한 타입의 속성을 처리해야 함
    - k-means는 수치형만 처리 가능
3. 인위적인 형상의 군집도 발견할 수 있어야 함
    - k-means는 non-convex 형태는 잘 못찾음
4. 파라미터 설정에 전문지식을 요하지 않아야함
5. noise와 outliers를 처리해야 함
6. 데이터가 입력되는 순서에 민감하면 안됨
7. 차원 수가 높아도 잘 처리할 수 있어야함
8. 사용자 정의 제약조건도 수용할 수 있어야함
9. 해석과 사용이 용이해야함
10. scaling, one hot encoding 등의 전처리가 필요하다.

```
### model


- Distance-based methods
  - Partitioning methods
    - k-means:
      - polinominal
      - noise, outlier
      -
      - non-convex
    - k-modes:          .
    - k-prototype:   ,
    - k-medoids:               outlier
      - PAM: Partitioning Around Medoids
        - scalability
      - CLARA: sampling    PAM  scalability
        -       biased
      - CLARANS: medoid
    - k-means++:   centroids
  - Hierarchical methods
    - top-down: divisive, dia
    - bottom-up: agglomerative
        - ward's distance:
```

            - ESS:
- Density-based methods
    -
    - noise, outlier
    - DBSCAN:
        1. core point (eps   minPts   )
        1. core point
            - core point
        -
        -
    - OPTICS: DBSCAN
        -
        -
        - eps, minPts
- Grid-based methods: (grid)
    -
- Model-based clustering methods

-       :
    -
    -
    -

###

- silhuette score: $\frac{\sum_{i=1}^{n} s(i)}{n}$
  - s(i): $\frac{b(i) - a(i)}{max((a(i), b(i)))}$
    - a(i):
    - b(i):
  - 1

생존분석

OR

- pulp 이용해서 푼다.
- 제약 함수, 결정 변수, 목표 함수만 잘 설정하면 풀 수 있을듯

Listing 0.10 `pulp example`

```
import pulp

prob = pulp.LpProblem("Problem_name", pulp.LpMinimize) #

# 2.       ( ,  ,  ,    )
x_A1 = pulp.LpVariable("A_to_1", 0, None, pulp.LpInteger)
x_A2 = pulp.LpVariable("A_to_2", 0, None, pulp.LpInteger)
x_A3 = pulp.LpVariable("A_to_3", 0, None, pulp.LpInteger)
x_B1 = pulp.LpVariable("B_to_1", 0, None, pulp.LpInteger)
x_B2 = pulp.LpVariable("B_to_2", 0, None, pulp.LpInteger)
x_B3 = pulp.LpVariable("B_to_3", 0, None, pulp.LpInteger)

# 3.       (    )
prob += 2*x_A1 + 4*x_A2 + 5*x_A3 + 3*x_B1 + 1*x_B2 + 6*x_B3, "obj_name"

# 4.
prob += x_A1 + x_A2 + x_A3 <= 100, "Factory_A_Supply"
prob += x_B1 + x_B2 + x_B3 <= 200, "Factory_B_Supply"
prob += x_A1 + x_B1 == 80, "Warehouse_1_Demand"
prob += x_A2 + x_B2 == 90, "Warehouse_2_Demand"
prob += x_A3 + x_B3 == 130, "Warehouse_3_Demand"

# 5.
prob.solve()

# 6.
print("Status:", pulp.LpStatus[prob.status])
print("\nOptimal Transportation Plan:")
for v in prob.variables():
    print(v.name, "=", v.varValue)

print("\nTotal Minimum Cost = ", pulp.value(prob.objective))
```