

---

# Smart Causal Inference

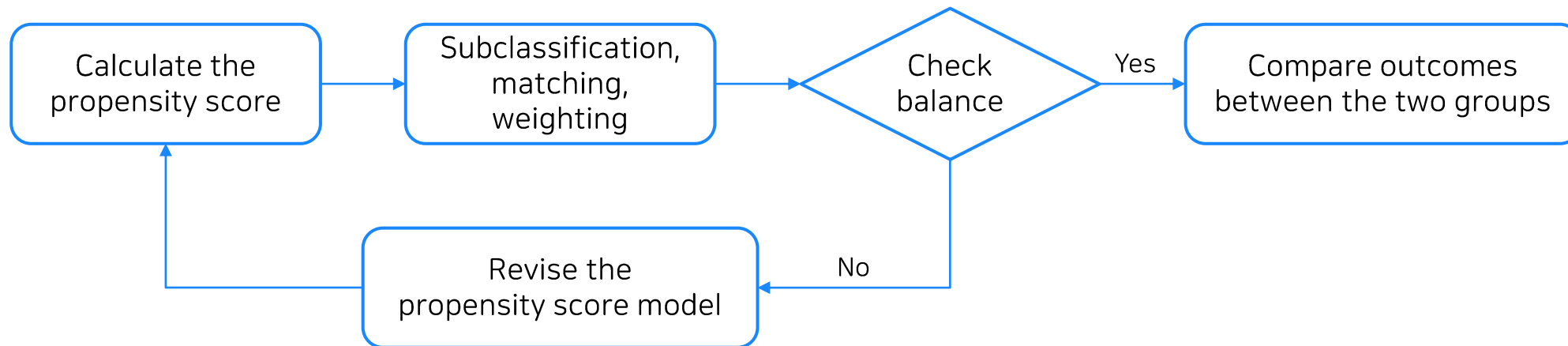
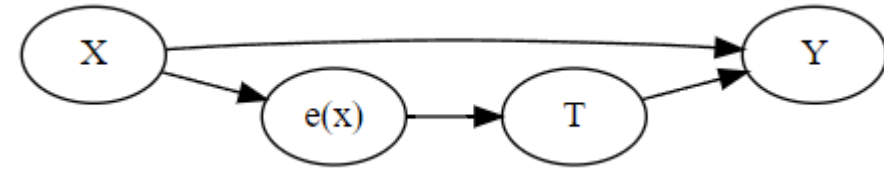
발표자: 이명석

## Causal Inference: Propensity score 성향점수 정의

특정 개체가 통제집단이 아닌 처치집단에 배치될 확률

$$e(X) = P(T|X),$$

where  $X = (X_1, \dots, X_p)$  is the collection of  $p$  covariates



Propensity Score Analysis Workflow

## Causal Inference: Propensity score

### 성향점수의 균형 특성

성향점수  $e(X)$  는 처치군 사이 모든  $X$  분산의 균형을 유지합니다.  $T \perp X \mid e(X)$

$E[T|X]$  를 추정하는 균형점수  $b(x)$  는 다음과 같은 공변량의 함수입니다.  $T \perp X \mid b(X)$

그래서 성향점수는 균형점수라고 부른다고 합니다. (?)

Rosenbaum와 Rubin(1983)은 성향점수  $e(x)$  가 가장 coarsest한 균형 점수라고 합니다.

## Causal Inference: Propensity score

# 성향점수 추정: 로지스틱 회귀

하지만 실제 성향점수는 알 수 없는 이상적인 값이므로 추정값으로 대체해야 합니다.

```
ps_model = smf.logit("""intervention ~
tenure + last_engagement_score + department_score
+ C(n_of_reports) + C(gender) + C(role)""", data=df).fit(dis=0)
```

```
data_ps = df.assign(
    propensity_score = ps_model.predict(df),
)

data_ps[["intervention", "engagement_score", "propensity_score"]].head()
```

	intervention	engagement_score	propensity_score
0	1	0.277359	0.596106
1	1	-0.449646	0.391138
2	1	0.769703	0.602578
3	1	-0.121763	0.580990
4	1	1.526147	0.619976

## Causal Inference: Propensity score

# 성향점수와 직교화

OLS 또한 성향점수 추정과 유사하게 처지 배정 매커니즘을 모델링합니다.

```
model = smf.ols("engagement_score ~ intervention + propensity_score",  
               data=data_ps).fit()  
model.params["intervention"]
```

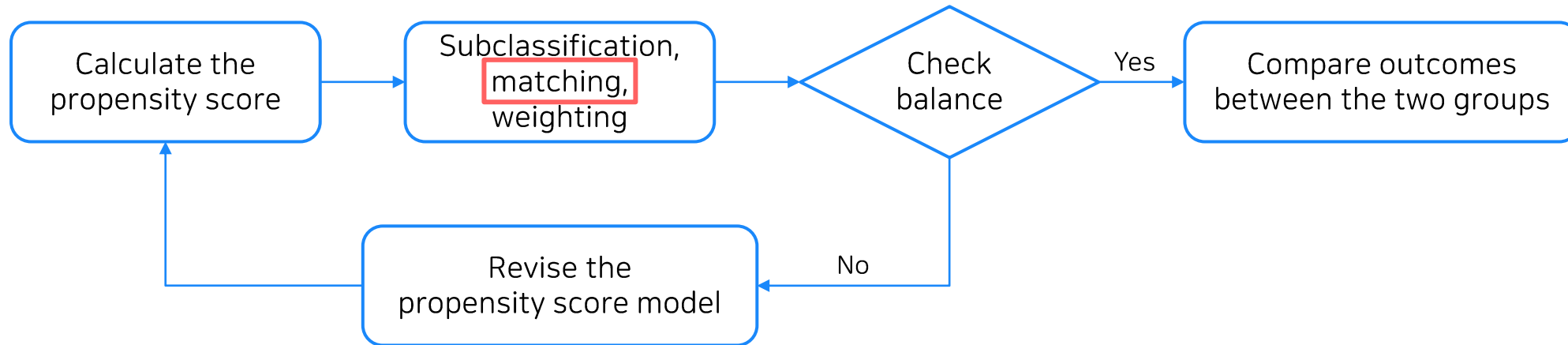
0.26331267490277066

## Causal Inference: Propensity score

### 성향점수 매칭

관찰 연구나 준실험에서 처리된 단위와 처리되지 않은 단위의 효과를 비교하여 처리의 효과를 평가

회귀에 비해 그룹 간 균형을 쉽게 평가할 수 있지만 성향 점수 추정에 민감하다는 한계 존재



## Causal Inference: Propensity score 성향점수 매칭 (KNN)

```
from sklearn.neighbors import KNeighborsRegressor

T = "intervention"
X = "propensity_score"
Y = "engagement_score"

treated = data_ps.query(f"{T}==1")
untreated = data_ps.query(f"{T}==0")

mt0 = KNeighborsRegressor(n_neighbors=1).fit(untreated[[X]],
                                             untreated[Y])

mt1 = KNeighborsRegressor(n_neighbors=1).fit(treated[[X]], treated[Y])

predicted = pd.concat([
    # find matches for the treated looking at the untreated knn model
    treated.assign(match=mt0.predict(treated[[X]])),

    # find matches for the untreated looking at the treated knn model
    untreated.assign(match=mt1.predict(untreated[[X]]))
])

predicted.head()

np.mean((predicted[Y] - predicted["match"])*predicted[T]
        + (predicted["match"] - predicted[Y])*(1-predicted[T]))
```

1. 성향 점수 계산
2. 이웃의 수 선택 (k=1)
3. 최근접 이웃 매칭
4. 실험군/대조군 짝 찾기

$$\widehat{ATE} = \frac{1}{N} \sum \{(Y_i - Y_{jm}(i))T_i + (Y_{jm}(i) - Y_i)(1 - T_i)\}$$

# Causal Inference: Propensity score

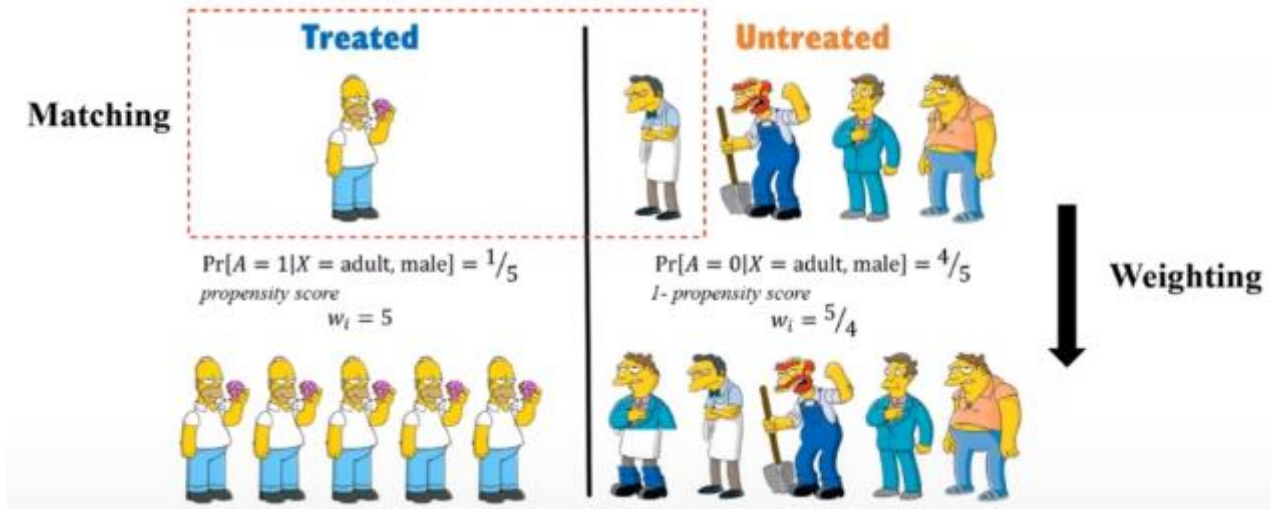
## 역확률 가중치

표본에  $1/P(T=t | X)$  가중치를 부여하여 모든 실험 대상이 처치  $t$ 를 받았을 경우와 같이 유사 모집단을 생성

$$E[Y_i] = E \left[ \frac{1(T = t)Y}{P(T = t | X)} \right]$$

$$ATE = E \left[ \frac{1(T = 1)Y}{P(T = 1 | X)} \right] - E \left[ \frac{1(T = 0)Y}{P(T = 0 | X)} \right]$$

$$= E \left[ Y \frac{T - e(x)}{e(x)(1 - e(x))} \right]$$





## Causal Inference: Propensity score

# 역확률 가중치의 분산

역확률 가중치의 표준오차를 계산하기 어려우므로 데이터를 반복 복원 추출하는 부트스트랩을 사용

```
from sklearn.linear_model import LogisticRegression * 패키지 변경으로 시간 절약
from patsy import dmatrix
```

```
# define function that computes the IPW estimator
def est_ate_with_ps(df, ps_formula, T, Y):

    X = dmatrix(ps_formula, df)
    ps_model = LogisticRegression(penalty="none",
                                  max_iter=1000).fit(X, df[T])
    ps = ps_model.predict_proba(X)[:, 1]

    # compute the ATE
    return np.mean((df[T]-ps) / (ps*(1-ps)) * df[Y])
```

```
formula = """tenure + last_engagement_score + department_score
+ C(n_of_reports) + C(gender) + C(role)"""
T = "intervention"
Y = "engagement_score"

est_ate_with_ps(df, formula, T, Y)
```

0.2659755621752663

# Causal Inference: Propensity score

## 역확률 가중치의 분산

```
from joblib import Parallel, delayed # for parallel processing
```

```
def bootstrap(data, est_fn, rounds=200, seed=123, pcts=[2.5, 97.5]): 추정값의 2.5번째와 97.5번째 백분위수를 계산하여 95% 신뢰구간 획득  
    np.random.seed(seed)
```

4개의 병렬처리

```
    stats = Parallel(n_jobs=4)(  
        delayed(est_fn)(data.sample(frac=1, replace=True)) 부트스트랩 표본 획득  
        for _ in range(rounds)  
    )
```

```
    return np.percentile(stats, pcts)
```

```
from toolz import partial
```

```
print(f"ATE: {est_ate_with_ps(df, formula, T, Y)}")
```

```
est_fn = partial(est_ate_with_ps, ps_formula=formula, T=T, Y=Y)  
print(f"95% C.I.: ", bootstrap(df, est_fn))
```

ATE: 0.2659755621752663

95% C.I.: [0.22654315 0.30072595]

```
def est_ate_with_ps(df, ps_formula, T, Y):
```

```
    X = dmatrix(ps_formula, df)  
    ps_model = LogisticRegression(penalty="none",  
                                  max_iter=1000).fit(X, df[T])  
    ps = ps_model.predict_proba(X)[:, 1]
```

```
    # compute the ATE  
    return np.mean((df[T]-ps) / (ps*(1-ps)) * df[Y])
```

# Causal Inference: Propensity score

## 안정된 성향점수 가중치

실험군과 대조군의 가중치 합을 원래 표본 크기와 비교하여 역확률 가중치 과정이 유사 모집단을 생성하는지 검증

```
print("Original Sample Size", data_ps.shape[0])
print("Treated Pseudo-Population Sample Size", sum(weight_t))
print("Untreated Pseudo-Population Sample Size", sum(weight_nt))
```

```
Original Sample Size 10391
Treated Pseudo-Population Sample Size 10435.089079197916
Untreated Pseudo-Population Sample Size 10354.298899788304
```

```
p_of_t = data_ps["intervention"].mean()

t1 = data_ps.query("intervention==1")
t0 = data_ps.query("intervention==0")

weight_t_stable = p_of_t/t1["propensity_score"]
weight_nt_stable = (1-p_of_t)/(1-t0["propensity_score"])

print("Treat size:", len(t1))
print("W treat", sum(weight_t_stable))

print("Control size:", len(t0))
print("W treat", sum(weight_nt_stable))
```

$$w = \frac{P(T = t)}{P(T = t | X)}$$

```
Treat size: 5611
W treat 5634.807508745978
Control size: 4780
W treat 4763.116999421415
```

```
nt = len(t1)
nc = len(t0)

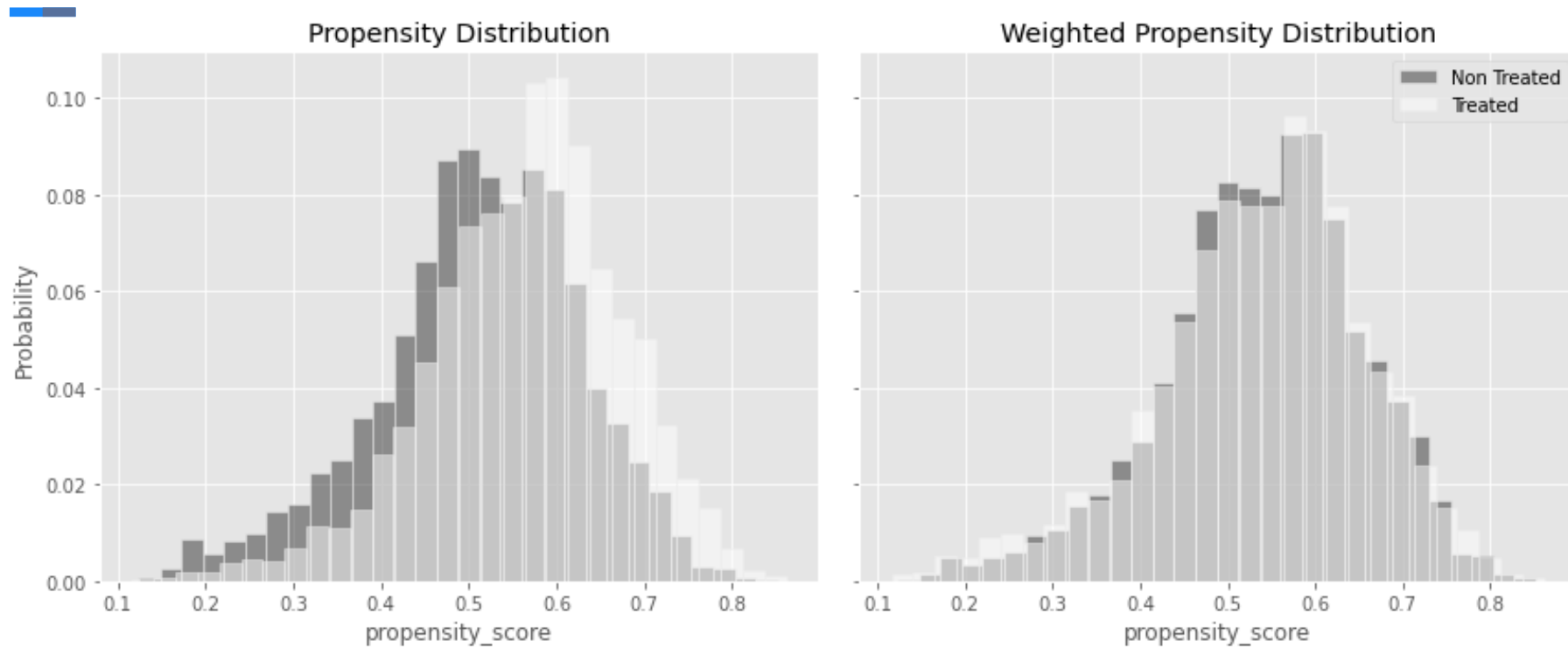
y1 = sum(t1["engagement_score"]*weight_t_stable)/nt
y0 = sum(t0["engagement_score"]*weight_nt_stable)/nc

print("ATE: ", y1 - y0)
```

```
ATE: 0.26597870880761176
```

# Causal Inference: Propensity score

## 유사 모집단



## Causal Inference: Propensity score 선택편향

역확률 가중치는 선택편향 보정에도 활용

- 고객 만족도 설문: 1~5점
- 무응답자로 인한 분석 결과 편향 (맘에 안들어서 응답 안함)
- 역확률 가중치 방법과 같이 응답자에게  $1/\hat{P}(R = 1)$ 만큼의 가중치 부여 (응답률 R)
- 그 결과 모두가 설문에 응답한 것과 같은 유사 모집단 생성

$$W = \frac{\hat{P}(T = t)}{\hat{P}(R = 1 | X) \hat{P}(T = t | X)}$$

## Causal Inference: Propensity score

### 편향-분산 트레이드오프

T가 매우 정확한 경우:

- $\hat{e}(x)$ 가 낮은 실험군이 없음 ( $Y_1 | T = 0$  을 추정할 수 없게 됨)

반대의 경우:

- $\hat{e}(x)$ 가 높은 대조군이 없음 ( $Y_0 | T = 1$  을 추정할 수 없게 됨)

# Causal Inference: Propensity score

## 편향-분산 트레이드오프



### ChatGPT

Inverse propensity weighting (IPW)의 편향-분산 균형은 추정자의 편향과 분산 사이의 균형을 나타내는 통계학 및 기계 학습의 기본적인 개념입니다. 이 균형은 편향을 줄이는 것이 분산을 증가시키는 경향이 있고 그 반대도 마찬가지입니다.

인과 추론에서 역 경향 가중치 (IPW)의 맥락에서 편향-분산 균형은 다음과 같은 방법으로 나타냅니다:

1. **편향:** 편향은 추정자의 기대값과 추정하려는 모수의 실제 값 간의 차이를 나타냅니다. IPW에서 편향은 경향 점수가 잘못 지정된 경우나 경향 점수를 모델링하는 데 사용된 기능 형태가 공변량과 처리 할당 간의 실제 관계를 충분히 포착하지 못하는 경우에 발생할 수 있습니다. 편향된 추정치는 처리 효과를 추정할 때 시스템적 오류를 초래할 수 있습니다.
2. **분산:** 분산은 동일한 모집단에서 다른 샘플들 사이에서 추정자의 변동성을 측정합니다. IPW에서 분산은 극단적인 경향 점수가 존재하거나 샘플 크기가 작은 경우에 증가할 수 있습니다. 극단적인 경향 점수는 불안정한 가중치를 야기하여 서로 다른 샘플에서 추정된 처리 효과의 높은 변동성을 초래할 수 있습니다.

편향과 분산 사이의 적절한 균형을 맞추는 것은 IPW에서 정확하고 신뢰할 수 있는 인과 효과의 추정치를 얻기 위해 중요합니다. 다음은 IPW에서 편향-분산 균형이 어떻게 작용하는지에 대한 설명입니다:

- 모델의 복잡성이나 유연성을 증가시키는 것(예: 경향 점수를 모델링하는 더 복잡한 기능 형태 사용)은 편향을 줄일 수 있지만 과적합으로 인해 분산이 증가할 수 있습니다.
- 반대로 모델을 단순화하는 것(예: 더 복잡한 모델 대신 선형 모델 사용)은 분산을 줄일 수 있지만 모델이 공변량과 처리 할당 간의 실제 관계를 포착하지 못하면 편향을 도입할 수 있습니다.

편향과 분산 사이의 적절한 균형을 찾는 것은 선택된 모델이 기본적인 관계를 충분히 포착하면서 과적합을 피하기 위해 조심스럽게 모델을 선택하고 정규화 기법(변수 선택이나 정규화 매개 변수와 같은) 및 검증 방법(교차 검증이나 부트스트래핑과 같은)을 사용하는 것을 의미합니다.

요약하면, 역 경향 가중치 (IPW)에서의 편향-분산 균형은 관측 연구에서 실제 처리 할당 메커니즘을 포착하는 것 (낮은 편향)과 안정적이고 신뢰할 수 있는 추정치 (낮은 분산)를 얻기 위한 필요성을 강조합니다.

결론:

편향을 통제할 수 있을 정도로만 정밀하게!

---

**End of Documentation**

---