1. Assignment

编程实现对以下案例中因果效应的估计。

1.1. 问题背景

估计钠摄入量对血压的影响。

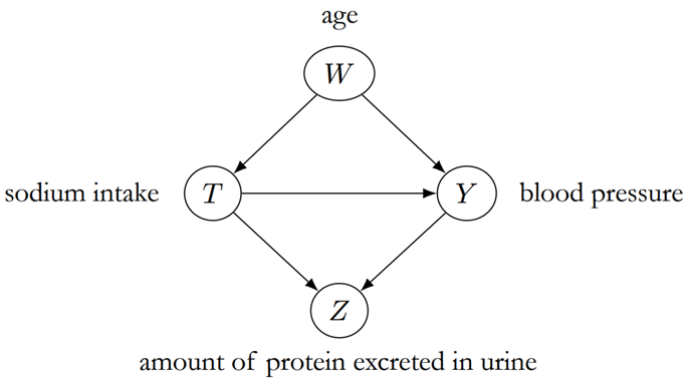
> 研究动机：快节奏的生活使得如今越来越多的人患上了高血压，血管壁长期承受着高于正常的压力会导致冠心病、脑卒中等严重疾病。

涉及的变量：

- Outcome Y: 血压
- Treatment T: 钠摄入量
- Covariates:
  - W: 年龄
  - Z: 尿液中蛋白质含量

通过模拟产生数据，所以我们知道"真实"ATE是1.05。

案例对应的因果图：



1.2. 任务

参考示例代码（见1.4），完成下列任务。

TODO 1:

实现`estimate_causal_effect`函数。

TODO 2:

思考以所有协变量作为调整集进行估计存在的问题，并根据上一任务实现的`estimate_causal_effect`函数，仅以年龄作为调整集进行估计（写出这样做的理由）。

TODO 3:

回忆条件结果模型存在的问题，有哪些进一步的改进方案？是如何改进的？

编程实现`estimate_causal_effect_with_GCOM`函数，使用GCOM(Grouped Conditional Outcome Modeling)进行估计。

TODO 4(Optional):

- 使用不同的编程语言或框架实现。
- 实现TARNet与X-Learner。

1.3. 提交

提交文件：学号_姓名.zip

- code文件夹
- 实验报告：学号_姓名.pdf（TODO 2、TODO 3中的相关问题解答）

提交邮箱：ruanzhh6@mail2.sysu.edu.cn

截止日期：2022.11.13 23:59

1.4. 示例代码

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression


def generate_data(n=1000, seed=0, beta1=1.05, alpha1=0.4, alpha2=0.3,
binary_treatment=True, binary_cutoff=3.5):
    """生成模拟数据"""
    np.random.seed(seed)
    age = np.random.normal(65, 5, n)
    sodium = age / 18 + np.random.normal(size=n)
    if binary_treatment:
        if binary_cutoff is None:
            binary_cutoff = sodium.mean()
        sodium = (sodium > binary_cutoff).astype(int)
    blood_pressure = beta1 * sodium + 2 * age + np.random.normal(size=n)
    proteinuria = alpha1 * sodium + alpha2 * blood_pressure +
np.random.normal(size=n)
    hypertension = (blood_pressure >= 140).astype(int)  # not used, but could be
used for binary outcomes
    return pd.DataFrame({'blood_pressure': blood_pressure, 'sodium': sodium,
                         'age': age, 'proteinuria': proteinuria})


def estimate_causal_effect(Xt, y, model=LinearRegression(), treatment_idx=0,
regression_coef=False):
    # TODO 1: 完成estimate_causal_effect函数
    model.fit(Xt, y)
    if regression_coef:
        # TODO
        return
    else:
        Xt1 = pd.DataFrame.copy(Xt)
        Xt1[Xt.columns[treatment_idx]] = 1
        Xt0 = pd.DataFrame.copy(Xt)
        Xt0[Xt.columns[treatment_idx]] = 0
        # TODO
        return

def estimate_causal_effect_with_GCOM():
    # TODO 3: 使用GCOM(Grouped Conditional Outcome Modeling)进行估计
```

```
    return


if __name__ == '__main__':
    binary_t_df = generate_data(beta1=1.05, alpha1=.4, alpha2=.3,
binary_treatment=True, n=10000000)
    continuous_t_df = generate_data(beta1=1.05, alpha1=.4, alpha2=.3,
binary_treatment=False, n=10000000)

    ate_est_naive = None
    ate_est_adjust_all = None
    ate_est_adjust_age = None

    for df, name in zip([binary_t_df, continuous_t_df],
                        ['Binary Treatment Data', 'Continuous Treatment Data']):
        print()
        print('### {} ###'.format(name))
        print()

        # Adjustment formula estimates
        ate_est_naive = estimate_causal_effect(df[['sodium']],
df['blood_pressure'], treatment_idx=0)
        ate_est_adjust_all = estimate_causal_effect(df[['sodium', 'age',
'proteinuria']],
                                                    df['blood_pressure'],
treatment_idx=0)
        # TODO 2: 仅以年龄作为调整集进行估计
        ate_est_adjust_age = estimate_causal_effect()
        print('# Adjustment Formula Estimates #')
        print('Naive ATE estimate:\t\t\t\t\t\t\t', ate_est_naive)
        print('ATE estimate adjusting for all covariates:\t',
ate_est_adjust_all)
        print('ATE estimate adjusting for age:\t\t\t\t', ate_est_adjust_age)
        print()

        # Linear regression coefficient estimates
        ate_est_naive = estimate_causal_effect(df[['sodium']],
df['blood_pressure'], treatment_idx=0,
                                               regression_coef=True)
        ate_est_adjust_all = estimate_causal_effect(df[['sodium', 'age',
'proteinuria']],
                                                    df['blood_pressure'],
treatment_idx=0,
                                                    regression_coef=True)
        # TODO 2: 仅以年龄作为调整集进行估计
        ate_est_adjust_age = estimate_causal_effect()
        print('# Regression Coefficient Estimates #')
        print('Naive ATE estimate:\t\t\t\t\t\t\t', ate_est_naive)
        print('ATE estimate adjusting for all covariates:\t',
ate_est_adjust_all)
        print('ATE estimate adjusting for age:\t\t\t\t', ate_est_adjust_age)
        print()
```