causal_inference

December 5, 2022

1 Causal Inference

This notebook shows our work on our causal inference question: How does a mother's smoking behavior cause the change of her baby's birth weight?

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
from scipy.stats import beta, binom
import itertools
from ipywidgets import interact, interactive
import statsmodels.api as sm

import sklearn
from sklearn.linear_model import LogisticRegression as LR
```

[2]:
```
#load dataset
birth = pd.read_csv('subsampled_clean_data.csv')
birth
```

[2]:
```
      ATTEND  BFACIL   BMI  DBWT DMAR  FAGECOMB  FEDUC  FRACE6 LD_INDL  MAGER  \
0          1       1  31.4  3670    1        29      6       1       N     32
1          2       1  27.6  3494    1        34      4       1       Y     33
2          1       1  27.1  3374    2        43      2       1       N     29
3          1       1  26.8  3520    1        30      3       1       Y     28
4          1       1  21.3  3140    1        30      5       1       N     30
…        …      …    …   …   …        …     …       …      …      …
9995       1       1  35.9  3062    1        30      4       1       Y     30
9996       1       1  22.5  3855    1        30      3       1       Y     23
9997       1       1  20.4  2710    1        39      2       1       Y     32
9998       1       1  24.4  3118    1        35      2       1       Y     34
9999       1       1  24.6  3020    1        32      3       1       N     28

      …  PRIORLIVE  PRIORTERM  RDMETH_REC  RESTATUS  RF_CESAR  SEX  \
0     …      False      False           1         2         N    M
1     …       True      False           1         1         N    F
```

```
2     …      True      True          1        1       N    M
3     …     False      True          1        1       N    M
4     …     False     False          1        3       N    M
…   …        …       …        …       …       …  …
9995  …      True     False          1        1       N    M
9996  …     False     False          3        1       N    M
9997  …      True      True          1        2       N    M
9998  …     False     False          3        2       N    M
9999  …      True     False          3        1       N    M

      PREG_LEN  WTGAIN_PER    CIG  FIRST_BIRTH
0            9    0.000000  False         True
1            9    0.120482  False        False
2           10    0.061350   True        False
3            9    0.301282  False         True
4            9    0.208333  False         True
…          …         …     …          …
9995         9    0.173684  False        False
9996         9    0.263514  False         True
9997         9    0.388889   True        False
9998        10    0.147887  False         True
9999         9    0.192308   True        False

[10000 rows x 31 columns]
```

1.1 Causal Inference - Randomized Experiments

Although the data is from a observational study, we could try to assume it is a randomized experiment and conduct causal inference. It can be served as a comparsion with the later causal inference results when we use observational study techniques.

For randomized experiment technique, we use the Fisher Randomized Test (*i.e.*, Permutation Test) with the simple difference in means ($\hat{\tau} = \frac{1}{n_1}\sum_{i=1}^{n} Z_i Y_i - \frac{1}{n_0}\sum_{i=1}^{n}(1 - Z_i)Y_i$) as the test statistic. The null hypothesis we are testing is

$$H_0 : Y_i(1) = Y_i(0) \quad \forall i = 1, \dots, n$$

which is also known as *sharp/strong null hypothesis*. It is basically saying that the treatment and control outcomes come from the *same* distribution.

First we compute the observed test statistic.

```
[3]: observed_T = np.mean(birth[birth["CIG"] == True].DBWT) - np.
      ↪mean(birth[birth["CIG"] == False].DBWT)
     observed_T
```

```
[3]: -115.41768336628411
```

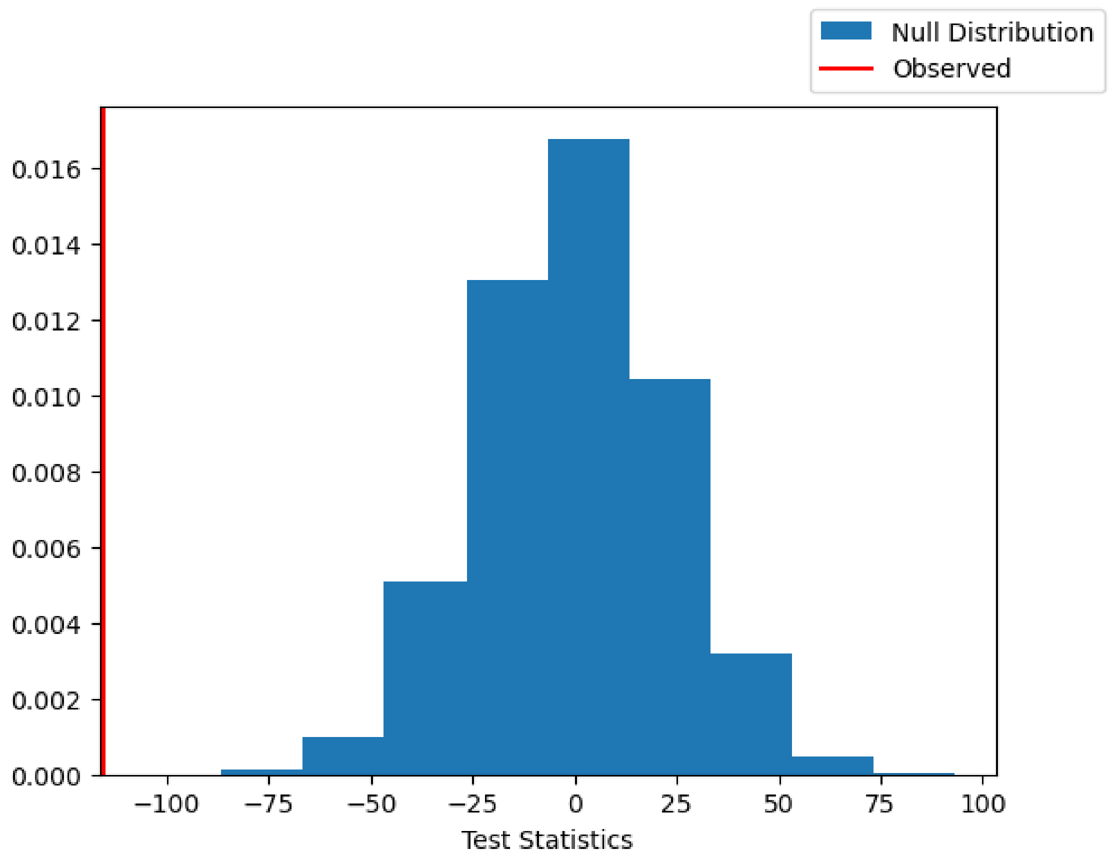
[4]:
```
birth['CIG'].sum()
```

[4]:
```
682
```

Since we have 10000 units with 682 treated units, going through all possible permutations – there are $\binom{10000}{682}$ different permutations) – will be too time-consuming. We can use Monte Carlo to approximate the true p-value.

[5]:
```
rng = np.random.default_rng(102)
R = 50000 # repetition times
Ts = np.zeros(R)
shuffled_birth = birth.copy()

for i in range(R):
    shuffled_birth['shuffled_CIG'] = rng.choice(birth['CIG'], size=birth.
  ↪shape[0], replace=False)
    Ts[i] = np.mean(shuffled_birth[shuffled_birth["shuffled_CIG"] == True].
  ↪DBWT) - np.mean(shuffled_birth[shuffled_birth["shuffled_CIG"] == False].DBWT)

p_val = np.sum(np.abs(Ts) >= np.abs(observed_T)) / R
print(f'The p-value is {p_val}')

fig, ax = plt.subplots()
ax.hist(Ts, density=True, label='Null Distribution')
ax.axvline(observed_T, color='r', label='Observed')
ax.set_xlabel('Test Statistics')
fig.legend()
fig.show()
```

```
The p-value is 0.0
```



Based on the approximated p-value, it is clear that we should reject the null hypothesis. Thus, we claim that a mother's smoking behavior will cause the change of her baby's birth weight. However, the permutation test cannot tell either qualitative (*i.e.*, whether smoking causes lower or higher birth weight) or quantitative (*i.e.*, how much birth weight increase or decrease can be caused by smoking) causal effect. To do so, we use the observational study techniques, and we will explain them next.

1.2 Causal Inference - Observational Studies

[6]:
```
causal_effect = np.mean(birth[birth["CIG"] == True].DBWT) - np.
  ↪mean(birth[birth["CIG"] == False].DBWT)
causal_effect
```

[6]:
```
-115.41768336628411
```

Using the simple difference in means, the causal effect of cigarettes on baby's birth weight is negative, meaning that smoking will lead to a lower birth weight.

1.3 Outcome Regression

We know that BMI and prenatal care may be confounders. Then by unconfoundedness, we could fit a linear model of the following form:

\$Birth Weights = * Z + a*BMI+ b*PRECARE \$

If we make two assumptions, then the estimated coefficient of treament from OLS, $\hat{\tau}$, will be an unbiased estimate of the ATE. The two assumptions are:

1. Assume unconfoundedness given `BMI` and `PRECARE`.
2. Assume this linear model correctly describes the interaction between the variables.

First, in order to fit the regression, we need to change categorical variable `CIG` to dummy variable `CIG_True`.

[7]:
```
#change categorical data to dummy variable
birth = pd.get_dummies(birth, columns=['CIG'], drop_first=True)
birth
```

[7]:
```
      ATTEND  BFACIL   BMI  DBWT DMAR  FAGECOMB  FEDUC  FRACE6 LD_INDL  MAGER  \
0          1       1  31.4  3670    1        29      6       1       N     32
1          2       1  27.6  3494    1        34      4       1       Y     33
2          1       1  27.1  3374    2        43      2       1       N     29
3          1       1  26.8  3520    1        30      3       1       Y     28
4          1       1  21.3  3140    1        30      5       1       N     30
…        …      …    …   …  …       …     …       …      …    …
9995       1       1  35.9  3062    1        30      4       1       Y     30
9996       1       1  22.5  3855    1        30      3       1       Y     23
9997       1       1  20.4  2710    1        39      2       1       Y     32
9998       1       1  24.4  3118    1        35      2       1       Y     34
9999       1       1  24.6  3020    1        32      3       1       N     28

      …  PRIORLIVE  PRIORTERM  RDMETH_REC  RESTATUS  RF_CESAR SEX  \
0     …      False      False           1         2         N   M
1     …       True      False           1         1         N   F
2     …       True       True           1         1         N   M
3     …      False       True           1         1         N   M
4     …      False      False           1         3         N   M
…   …          …          …           …         …       …  …
9995  …       True      False           1         1         N   M
9996  …      False      False           3         1         N   M
9997  …       True       True           1         2         N   M
9998  …      False      False           3         2         N   M
9999  …       True      False           3         1         N   M

      PREG_LEN  WTGAIN_PER  FIRST_BIRTH  CIG_True
0            9    0.000000         True         0
1            9    0.120482        False         0
2           10    0.061350        False         1
```

```
3             9    0.301282        True         0
4             9    0.208333        True         0
…        …        …          …        …
9995          9    0.173684       False         0
9996          9    0.263514        True         0
9997          9    0.388889       False         1
9998         10    0.147887        True         0
9999          9    0.192308       False         1

[10000 rows x 31 columns]
```

treatement (Z): `birth['CIG_True']`

outcome (Y): `birth['DBWT']`

confounder (X): `birth['BMI']`, `birth['PRECARE']`

units: baby's birth weights

[8]:
```
def fit_OLS_model(df, target_variable, explanatory_variables, intercept =␣
 ↪False):
    target = df[target_variable]
    inputs = df[explanatory_variables]
    if intercept:
        inputs = sm.add_constant(inputs)

    fitted_model = sm.OLS(target, inputs).fit()
    return(fitted_model)

def mean_squared_error(true_vals, predicted_vals):
    return np.mean((true_vals - predicted_vals) ** 2)
```

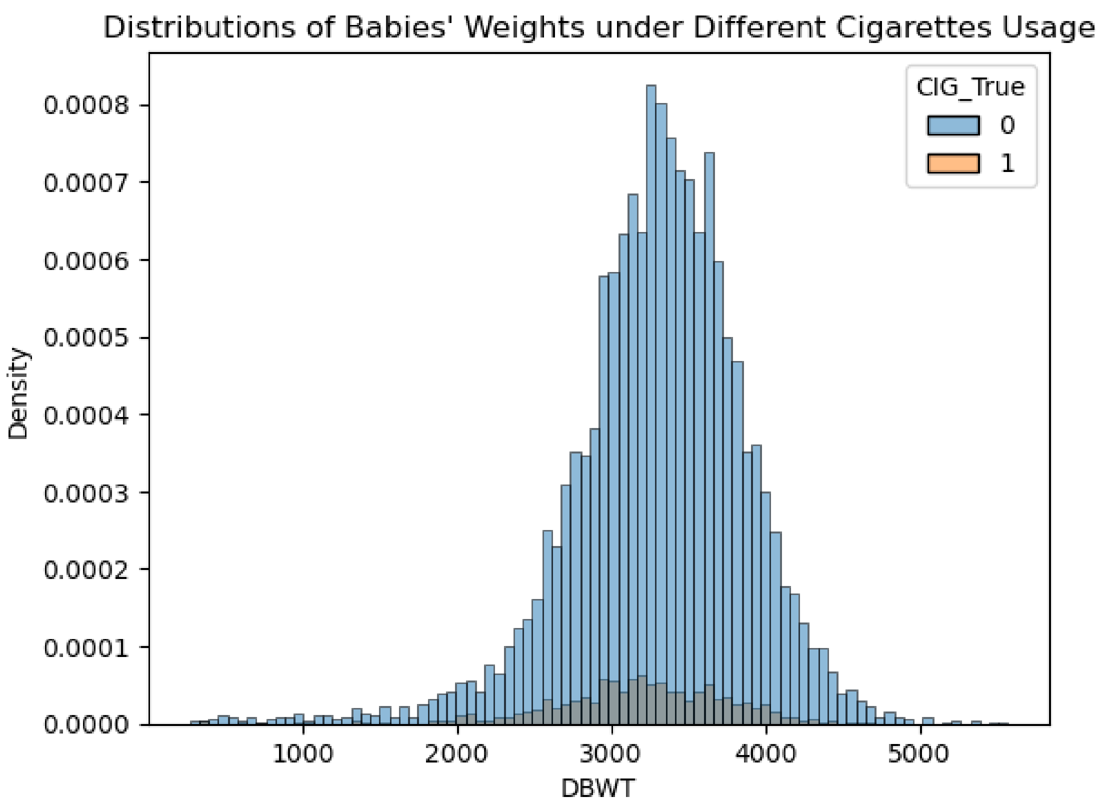
[9]:
```
full_linear_model = fit_OLS_model(birth, 'DBWT', ['CIG_True', 'BMI', 'PRECARE'])
print(full_linear_model.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                   DBWT   R-squared (uncentered):
0.933
Model:                            OLS   Adj. R-squared (uncentered):
0.933
Method:                 Least Squares   F-statistic:
4.611e+04
Date:                Mon, 05 Dec 2022   Prob (F-statistic):
0.00
Time:                        21:50:04   Log-Likelihood:
-81848.
No. Observations:               10000   AIC:
```

```
1.637e+05
Df Residuals:                    9997   BIC:
1.637e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
CIG_True    -104.8087     34.480     -3.040      0.002    -172.397     -37.221
BMI           88.6601      0.707    125.335      0.000      87.273      90.047
PRECARE      642.9650     15.157     42.421      0.000     613.254     672.675
==============================================================================
Omnibus:                     1056.169   Durbin-Watson:                   1.978
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1570.134
Skew:                          -0.797   Prob(JB):                         0.00
Kurtosis:                       4.107   Cond. No.                         111.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[10]:
```
sns.histplot(data=birth, x='DBWT', hue='CIG_True', stat='density')
plt.title("Distributions of Babies' Weights under Different Cigarettes Usage")
```

[10]:
```
Text(0.5, 1.0, "Distributions of Babies' Weights under Different Cigarettes
Usage")
```

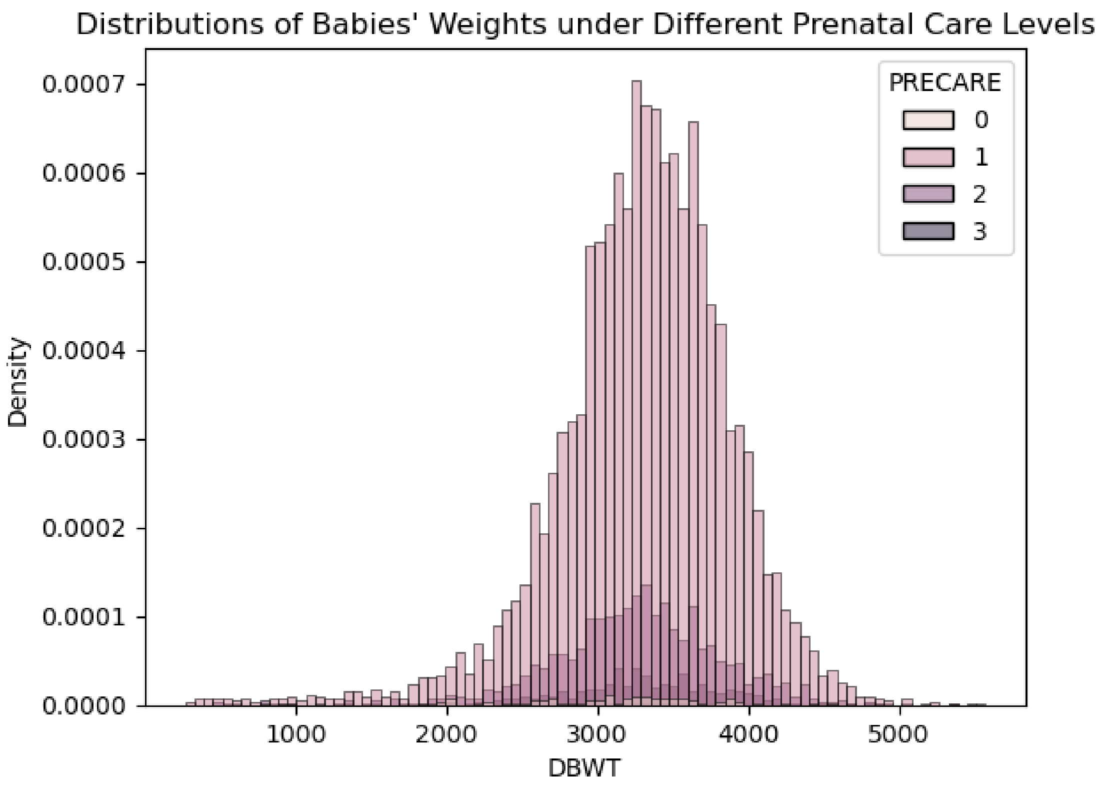


```
[11]: birth.groupby('CIG_True')['DBWT'].mean()
```

```
[11]: CIG_True
      0    3299.782786
      1    3184.365103
      Name: DBWT, dtype: float64
```

```
[12]: sns.histplot(data=birth, x='DBWT', hue='PRECARE', stat='density')
      plt.title("Distributions of Babies' Weights under Different Prenatal Care␣
       ↪Levels")
```

```
[12]: Text(0.5, 1.0, "Distributions of Babies' Weights under Different Prenatal Care
      Levels")
```



```
[13]: birth.groupby('PRECARE')['DBWT'].mean()
```

```
[13]: PRECARE
      0    3056.591398
      1    3301.014548
      2    3253.022464
      3    3295.043228
      Name: DBWT, dtype: float64
```

From the outcome regression results table, we can see that by using `BMI` and `PRECARE` as confounders, the causal coefficient of cigarettes consumption on babies' weights is around -104.8. This means that smoking an extra cigarette will likely decrease the baby's weight by 104.8 grams.

From the graphs above, we can also see that the distribution of babies' weights, whether mother smoke or not, are uniformly distributed. However, babies whose mothers don't smoke tend to weigh more than those whose mothers do, having a difference in mean around 115 grams.

We can see that the distribution of babies' weights with different levels of prenatal care are also uniformly distributed. Generally, more prenatal care will lead to higher babies weights, and level-3 precare has the highest average weight of 3295 grams. However, we can see that the average weights of babies under level-1 precare is actually higher for those under level-2. What causes this difference is unkown.

1.3.1 Inverse Propensity Score

In this section, we use inverse propensity weighting.

Propensity score calculates the probability that a unit was treated, conditioned on a particular set of confounders x:

$$e(x) = P(Z = 1|X = x)$$

For inverse propensity weighting (IPW). Let $n = n_0 + n_1$ be the total number of observations. The IPW estimator of the ATE is:

$$\hat{\tau}_{IPW} = \frac{1}{n}\sum_{i:Z_i=1}\frac{Y_i}{e(X_i)} - \frac{1}{n}\sum_{i:Z_i=0}\frac{Y_i}{1-e(X_i)}$$

```
[14]: Z = birth.CIG_True.values
      Y = birth.DBWT.values
      X = birth[['BMI', 'PRECARE']]

      lr = LR(max_iter=200, random_state=0)
      lr.fit(X, Z)
```

```
[14]: LogisticRegression(max_iter=200, random_state=0)
```

```
[15]: birth['pscore'] = lr.predict_proba(X)[:,1]
      birth
```

```
[15]:       ATTEND  BFACIL   BMI  DBWT DMAR  FAGECOMB  FEDUC  FRACE6 LD_INDL  MAGER  \
      0          1       1  31.4  3670    1        29      6       1       N     32
      1          2       1  27.6  3494    1        34      4       1       Y     33
      2          1       1  27.1  3374    2        43      2       1       N     29
      3          1       1  26.8  3520    1        30      3       1       Y     28
      4          1       1  21.3  3140    1        30      5       1       N     30
      …        …      …   …   …  …       …      …      …      …      …
      9995       1       1  35.9  3062    1        30      4       1       Y     30
      9996       1       1  22.5  3855    1        30      3       1       Y     23
      9997       1       1  20.4  2710    1        39      2       1       Y     32
      9998       1       1  24.4  3118    1        35      2       1       Y     34
      9999       1       1  24.6  3020    1        32      3       1       N     28

            …  PRIORTERM  RDMETH_REC  RESTATUS  RF_CESAR  SEX  PREG_LEN  \
      0     …      False           1         2         N    M         9
      1     …      False           1         1         N    F         9
      2     …       True           1         1         N    M        10
      3     …       True           1         1         N    M         9
      4     …      False           1         3         N    M         9
      …   …         …          …         …        …  …        …
```

```
9995  …     False          1         1         N    M          9
9996  …     False          3         1         N    M          9
9997  …      True          1         2         N    M          9
9998  …     False          3         2         N    M         10
9999  …     False          3         1         N    M          9

      WTGAIN_PER  FIRST_BIRTH  CIG_True    pscore
0       0.000000         True         0  0.068063
1       0.120482        False         0  0.063830
2       0.061350        False         1  0.063292
3       0.301282         True         0  0.062971
4       0.208333         True         0  0.057350
…            …           …       …        …
9995    0.173684        False         0  0.073410
9996    0.263514         True         0  0.058535
9997    0.388889        False         1  0.076169
9998    0.147887         True         0  0.060458
9999    0.192308        False         1  0.081689

[10000 rows x 32 columns]
```

[16]:
```
n = len(birth)
ipw = np.sum(birth[birth["CIG_True"] == 1].DBWT / birth[birth["CIG_True"] == 1].
 ↪pscore)/n - np.sum(birth[birth["CIG_True"] == 0].DBWT / (1 -␣
 ↪birth[birth["CIG_True"] == 0].pscore))/n
ipw
```

[16]:
```
-121.73829642696728
```

We can see that by applying inverse propensity weighting, the coefficient of cigarettes on babies' birth is around -121.74, meaning that smoking an extra cigarette will likely decrease the baby's weight by 121.74 grams.

Anomalies happens if some observations are rare in the treatment group (i.e. ( ) 0), which may cause the inverse propensity score, 1/ ( ) to be enormous. Therefore, we decide to only include points with propensity scores between 0.1 and 0.9 which accepts some bias to reduce the variance.

[17]:
```
birth_new = birth[(birth['pscore'] > 0.1) & (birth['pscore'] < 0.9)]
n = len(birth_new)
trimmed_ipw = np.sum(birth_new[birth_new["CIG_True"] == 1].DBWT /␣
 ↪birth_new[birth_new["CIG_True"] == 1].pscore)/n - np.
 ↪sum(birth_new[birth_new["CIG_True"] == 0].DBWT / (1 -␣
 ↪birth_new[birth_new["CIG_True"] == 0].pscore))/n
trimmed_ipw
```

[17]:
```
-593.0929729236877
```

Now, we can see that the coefficient of cigarettes on babies' birth becomes larger, around -593, meaning that smoking an extra cigarette will likely decrease the baby's weight by 593 grams.

prediction_glm

December 5, 2022

1 Prediction

This notebook shows our work on our prediction question: How to predict the birth weight?

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

[2]:
```
birth = pd.read_csv('subsampled_clean_data.csv')
```

[3]:
```
birth.head()
```

[3]:
```
   ATTEND  BFACIL   BMI  DBWT DMAR  FAGECOMB  FEDUC  FRACE6 LD_INDL  MAGER  \
0       1       1  31.4  3670    1        29      6       1       N     32
1       2       1  27.6  3494    1        34      4       1       Y     33
2       1       1  27.1  3374    2        43      2       1       N     29
3       1       1  26.8  3520    1        30      3       1       Y     28
4       1       1  21.3  3140    1        30      5       1       N     30

   …  PRIORLIVE  PRIORTERM  RDMETH_REC  RESTATUS  RF_CESAR  SEX  PREG_LEN  \
0  …      False      False           1         2         N    M         9
1  …       True      False           1         1         N    F         9
2  …       True       True           1         1         N    M        10
3  …      False       True           1         1         N    M         9
4  …      False      False           1         3         N    M         9

   WTGAIN_PER    CIG  FIRST_BIRTH
0    0.000000  False         True
1    0.120482  False        False
2    0.061350   True        False
3    0.301282  False         True
4    0.208333  False         True

[5 rows x 31 columns]
```

2 Research Question

Predicting a baby's birth weight from Mother's Single Years of Age and Number of Prenatal Visits, comparing GLMs to nonparametric methods.

3 Goal

We are trying to use various models including GLMs and Nonparametric methods to predict baby's birth weights. In the original EDA and using our domain knowledge on newborn health, we proposed two features that will be helpful for constructing the model, **Mother's Single Years of Age (MAGER)** and **Number of Prenatal Visits (PREVIS)**.
MAGER: Using our common knowledge, we think that younger mothers are likely to have healthier babies and thus higher birth weights.
PREVIS: Using our common knowledge, we think that more prenatal visits means that the family pays more attention to the pregnancy and thus is likely to have babies with higher birth weights.

4 Nonparametric Method-Decision Tree

Why Decision Tree
For nonparametric methed like decision tree, we do not need to consider about selecting the features. Since the impurity and impurity reduction will do the heavy lifting. The first few depths will select the best features and threshold that decrease the impurity the most, which help us to see which features are the best for the prediction.

Assumption 1. By using Decision Tree, we assume that the relationship is non linear and complex. 2. Since y is continuous, we will be using regression tree from CART, the DecisionTreeRegressor. We need to realize some mechanics of the model in order to make some assumption about our dataset: The model takes in X and y as input, and it tries to iterate through all the possible features in X and iterate thourgh all the values(threshold) that particular feature can be. And calculate the impurity reduction of such split, where

$$Impurity = \frac{1}{N_n(t)} \sum_{i: X_i \in R_i} (y_i - \mu_n(t))^2 \qquad (1)$$

and **Impurity reduction**

$$\Delta_I(t) = Impurity(t) - \frac{N_n(t^{left})}{N_n(t)} Impurity(t^{left}) - \frac{N_n(t^{right})}{N_n(t)} Impurity(t^{right}) \qquad (2)$$

Impurity calculates the weighted sum of difference between mean and each sample's label squared in a tree node, which basically tells how pure a particular node is. The Impurity reduction calculates the difference between the Impurity before split and the weighted sum of the Impurity of nodes after split. Since there's mean involved in the model, we need to make all the data to continuous, we will achieve this by performing one-hot encoding.

4.0.1 Baby Weights (y)

```
[4]: y = np.array(birth['DBWT'])
     y
```

```
[4]: array([3670, 3494, 3374, …, 2710, 3118, 3020])
```

4.0.2 Valid features to construct the tree (X)

```
[5]: # Drop irrelevant features and y
     X_drop = birth.drop(['DBWT', 'DMAR'], axis = 1)
     X_drop.head()
```

```
[5]:    ATTEND  BFACIL   BMI  FAGECOMB  FEDUC  FRACE6 LD_INDL  MAGER  MBSTATE_REC  \
     0       1       1  31.4        29      6       1       N     32            1
     1       2       1  27.6        34      4       1       Y     33            1
     2       1       1  27.1        43      2       1       N     29            1
     3       1       1  26.8        30      3       1       Y     28            1
     4       1       1  21.3        30      5       1       N     30            1

        MEDUC  …  PRIORLIVE  PRIORTERM  RDMETH_REC  RESTATUS  RF_CESAR  SEX  \
     0      6  …      False      False           1         2         N    M
     1      7  …       True      False           1         1         N    F
     2      2  …       True       True           1         1         N    M
     3      7  …      False       True           1         1         N    M
     4      4  …      False      False           1         3         N    M

        PREG_LEN  WTGAIN_PER    CIG  FIRST_BIRTH
     0         9    0.000000  False         True
     1         9    0.120482  False        False
     2        10    0.061350   True        False
     3         9    0.301282  False         True
     4         9    0.208333  False         True

     [5 rows x 29 columns]
```

```
[6]: # Get all the categorical data
     cat_cols = X_drop.select_dtypes(exclude=["number"]).columns
     cat_cols
```

```
[6]: Index(['LD_INDL', 'PRIORDEAD', 'PRIORLIVE', 'PRIORTERM', 'RF_CESAR', 'SEX',
            'CIG', 'FIRST_BIRTH'],
           dtype='object')
```

```
[7]: for c in cat_cols:
         encoded = pd.get_dummies(X_drop[c], prefix=c)
         X_drop = pd.concat([X_drop, encoded], axis='columns')
```

```
[8]: X_encoded = X_drop.drop(cat_cols, axis = 1)
```

```
[9]: X_encoded.head()
```

```
[9]:    ATTEND  BFACIL   BMI  FAGECOMB  FEDUC  FRACE6  MAGER  MBSTATE_REC  MEDUC  \
     0       1       1  31.4        29      6       1     32            1      6
     1       2       1  27.6        34      4       1     33            1      7
     2       1       1  27.1        43      2       1     29            1      2
     3       1       1  26.8        30      3       1     28            1      7
     4       1       1  21.3        30      5       1     30            1      4

        MRAVE6  …  PRIORTERM_False  PRIORTERM_True  RF_CESAR_N  RF_CESAR_Y  \
     0       1  …                1               0           1           0
     1       1  …                1               0           1           0
     2       1  …                0               1           1           0
     3       1  …                0               1           1           0
     4       1  …                1               0           1           0

        SEX_F  SEX_M  CIG_False  CIG_True  FIRST_BIRTH_False  FIRST_BIRTH_True
     0      0      1          1         0                  0                 1
     1      1      0          1         0                  1                 0
     2      0      1          0         1                  1                 0
     3      0      1          1         0                  0                 1
     4      0      1          1         0                  0                 1

     [5 rows x 37 columns]
```

4.0.3 Fit

```
[10]: X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.
       ↪01, random_state=0)
```

```
[11]: depths = [i for i in range(1, 10)]
      train_scores = np.ones(len(depths))
      test_scores = np.ones(len(depths))
```

Cross validation

```
[12]: for idx in range(len(depths)):
          clf = tree.DecisionTreeRegressor(max_depth = depths[idx])
          clf.fit(X_train, y_train)
          train_scores[idx] = clf.score(X_train, y_train)
          test_scores[idx] = clf.score(X_test, y_test)
          print("Max depths" ,depths[idx] , "will have train score" ,␣
       ↪train_scores[idx] , "and test score" , test_scores[idx])
```

```
Max depths 1 will have train score 0.15722212437225924 and test score
0.12774532540632055
```

```
Max depths 2 will have train score 0.23653748245476214 and test score
0.22971336641048823
Max depths 3 will have train score 0.26145146321359913 and test score
0.21841821841646225
Max depths 4 will have train score 0.2864851728757737 and test score
0.2692236761573288
Max depths 5 will have train score 0.30956300362718303 and test score
0.2591029708037196
Max depths 6 will have train score 0.33911101131747656 and test score
0.26877151389971643
Max depths 7 will have train score 0.3766485557353514 and test score
0.2162482576076865
Max depths 8 will have train score 0.42306787845459326 and test score
0.2156016222426992
Max depths 9 will have train score 0.47492748583367006 and test score
0.15871008237693107
```

According to the cross validation, max_depth = 4 looks promising, so that's what we will use in the prediction phase

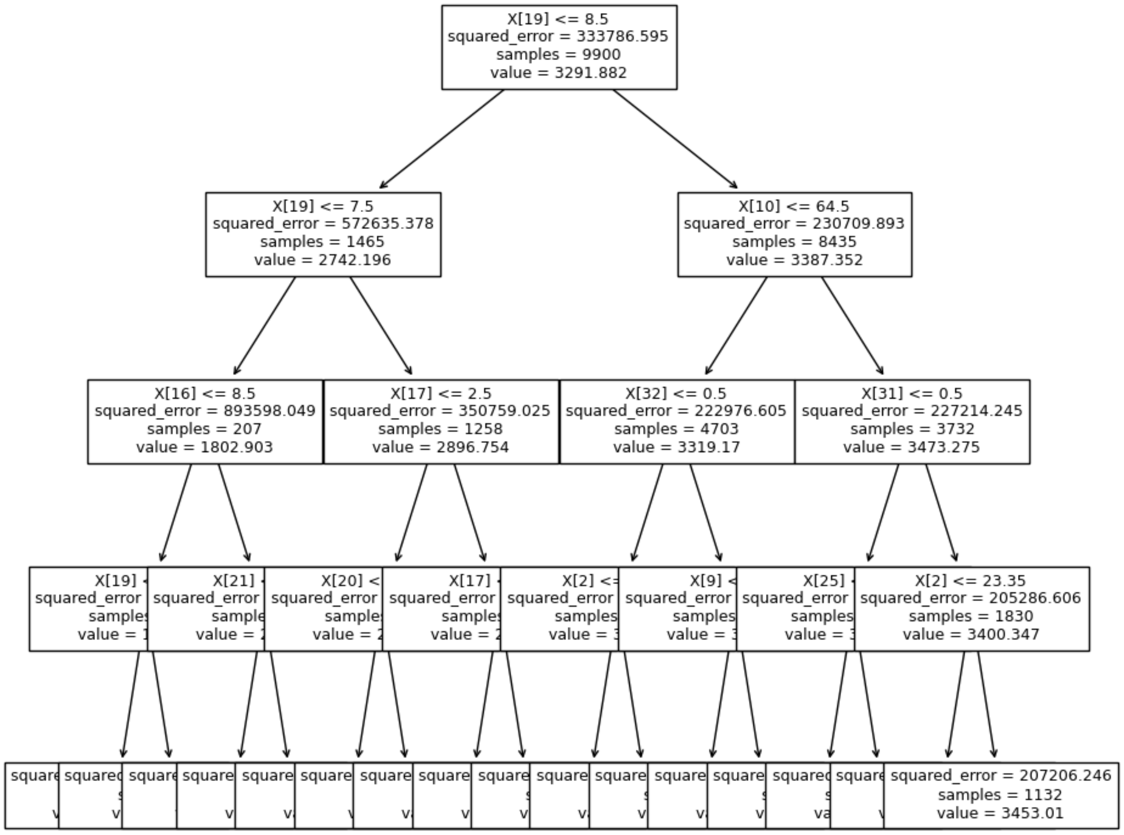
[13]:
```
clf = tree.DecisionTreeRegressor(max_depth = 4)
clf.fit(X_train, y_train)
```

[13]:
```
DecisionTreeRegressor(max_depth=4)
```

4.0.4 Summarize and interpret

Visualize tree

[14]:
```
plt.figure(figsize=(10,10))
tree.plot_tree(clf, fontsize=9)
plt.show()
```



```
[15]: clf.feature_names_in_[19], clf.feature_names_in_[10], clf.feature_names_in_[16]
```

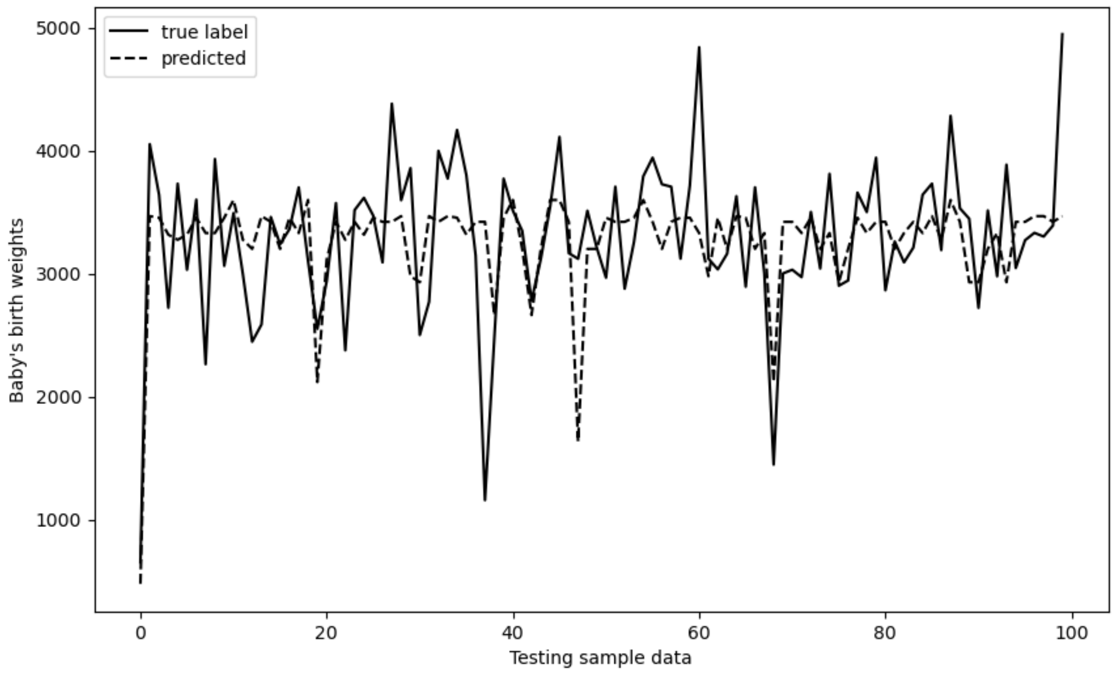
```
[15]: ('PREG_LEN', 'M_Ht_In', 'PREVIS')
```

According to the tree we got, the features like **number of pregnancy months (PREG_LEN)** and **Mother's height (M_HT_In)** helps the most. This makes sense because:
(1) the longer baby stay in mother's body, the more likely they can get more nutrition from nutrient transfer and thus more heavy
(2) the taller mother may have a higher chance to have a taller baby, and it will make the weights change

Ploting the true labels and predicted labels

```
[16]: plt.rcParams['figure.figsize'] = (10, 6)
x_ax = range(len(X_test))
plt.plot(x_ax, y_test, label = 'true label', color = 'k', linestyle = '-')
plt.plot(x_ax, clf.predict(X_test), label = 'predicted', color = 'k', linestyle␣
  ↪= '--')
```

```
plt.ylabel("Baby's birth weights")
plt.xlabel("Testing sample data")
plt.legend()
plt.show()
```



As the plot indicates, although the score of the model isn't the best, but it actually perform decently. The overlap between true label and predicted label is fair considering they are continuous.

5 Nonparametric Method-Random Forest

[17]:
```
clf1 = RandomForestRegressor(n_estimators= 200,max_depth=15)
```

[18]:
```
clf1.fit(X_train, y_train)
```

[18]:
```
RandomForestRegressor(max_depth=15, n_estimators=200)
```

[19]:
```
clf1.score(X_train, y_train)
```

[19]:
```
0.7983857863485958
```

[20]:
```
clf1.score(X_test, y_test)
```

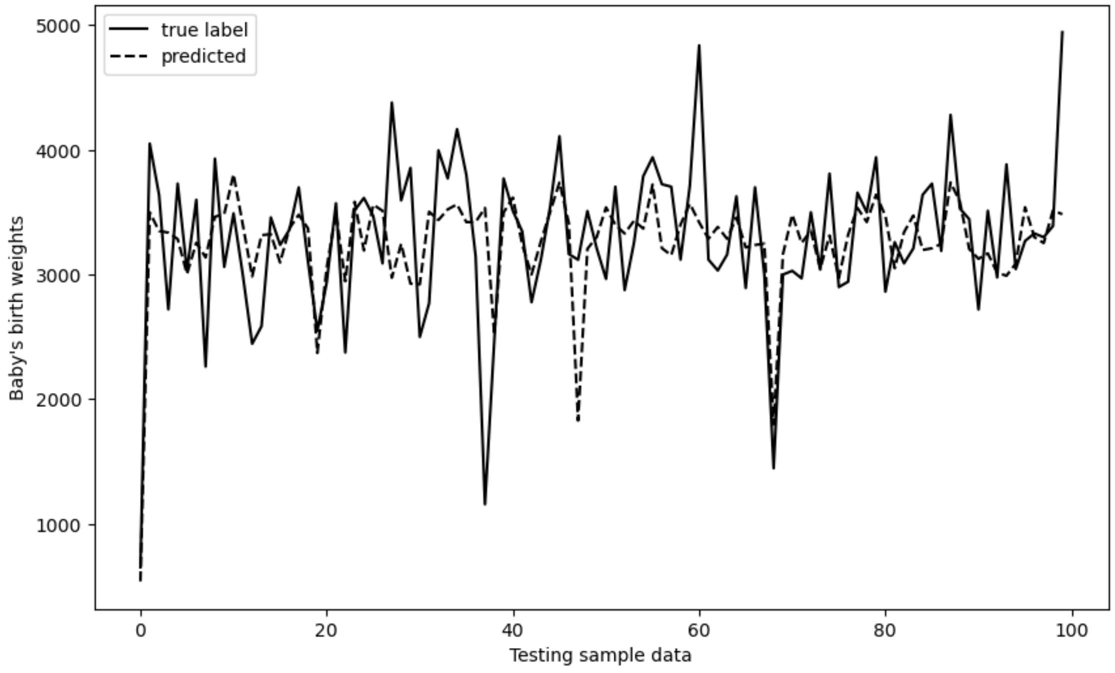
[20]:
```
0.32263016813649614
```

5.0.1 Summarize and interpret

We are unable to interpret the model since random forest is a ensemble algorithms that takes vote from hundreds of learners. Although the crowds' decision helps to lower the variance, it is unable to interpret anymore

Ploting the true labels and predicted labels

```
[21]: plt.rcParams['figure.figsize'] = (10, 6)
x_ax = range(len(X_test))
plt.plot(x_ax, y_test, label = 'true label', color = 'k', linestyle = '-')
plt.plot(x_ax, clf1.predict(X_test), label = 'predicted', color = 'k',␣
 ↪linestyle = '--')
plt.ylabel("Baby's birth weights")
plt.xlabel("Testing sample data")
plt.legend()
plt.show()
```



6 GLM

```
[22]: #import the pymc3 package
import pymc3 as pm
from pymc3 import glm
import statsmodels.api as sm
import arviz as az
```

6.1 Choice of Model - Linear Regression

From EDA, we see that DBWT, MAGER, and PREVIS plots are all roughly normal distributed, and we are predicting real-valued outputs, so the best choice of model here is Linear Regression.

Inverse Link Function: Identity
Likelihood: Gaussian

This means that we will model birth weight as $W_i \sim N(\beta_0 + \beta_1 M_i + \beta_2 P_i, \sigma^2 I_n)$ where M_i is MAGER and P_i is PREVIS.

6.2 Assumptions

For our model (Linear Regression), we are making the following assumptions: 1. There is a linear relationship between the response variable(DBWT) and explanatory variables (MAGER and PREVIS) 2. Constant Variance 3. Birth weights are assumed to be normally distirbuted (The histogram from EDA shows that this is valid). 4. The Birth weigths are independently distributed.

6.3 Frequentist Regression

[23]:
```
freq_model = sm.GLM(birth.DBWT, exog = sm.
 ↪add_constant(birth[['MAGER','PREVIS']]),
                  family=sm.families.Gaussian())
freq_res = freq_model.fit()
print(freq_res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                   DBWT   No. Observations:                10000
Model:                            GLM   Df Residuals:                     9997
Model Family:                Gaussian   Df Model:                            2
Link Function:               identity   Scale:                      3.2879e+05
Method:                          IRLS   Log-Likelihood:                -77704.
Date:                Mon, 05 Dec 2022   Deviance:                   3.2869e+09
Time:                        21:48:09   Pearson chi2:                 3.29e+09
No. Iterations:                     3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       3014.8158     34.136     88.318      0.000    2947.911    3081.721
MAGER          2.0604      1.031      1.998      0.046       0.039       4.081
PREVIS        18.5685      1.454     12.775      0.000      15.720      21.417
==============================================================================

/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

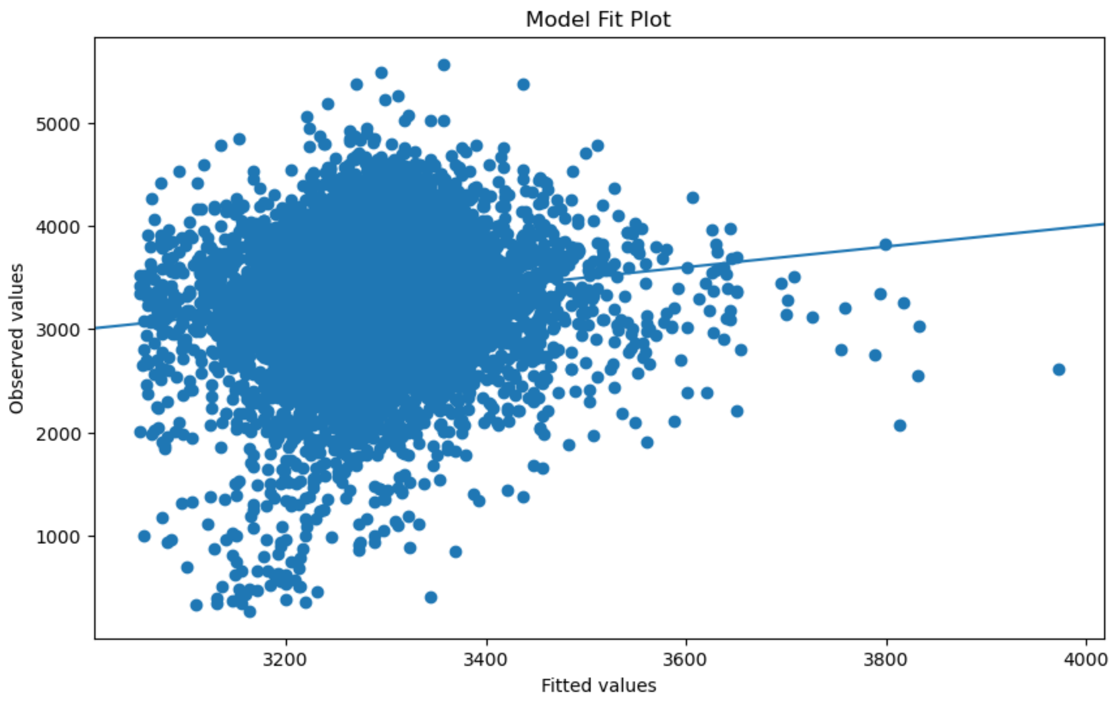
We see that the prediction is $W_i = 3014.8158 + 2.0604M_i + 18.5685P_i$. This means that older mothers have babies with higher birth weights, and mothers who go to more prenatal visists also have babies with higher birth weights.

6.3.1 Model Checking

```
[24]: from statsmodels.graphics.api import abline_plot
```

```
[25]: nobs = freq_res.nobs
      y = np.array(birth['DBWT'])
      yhat = freq_res.mu
```

```
[26]: fig, ax = plt.subplots()
      ax.scatter(yhat, y)
      line_fit = sm.OLS(y, sm.add_constant(yhat, prepend=True)).fit()
      abline_plot(model_results=line_fit, ax=ax)


      ax.set_title('Model Fit Plot')
      ax.set_ylabel('Observed values')
      ax.set_xlabel('Fitted values');
```

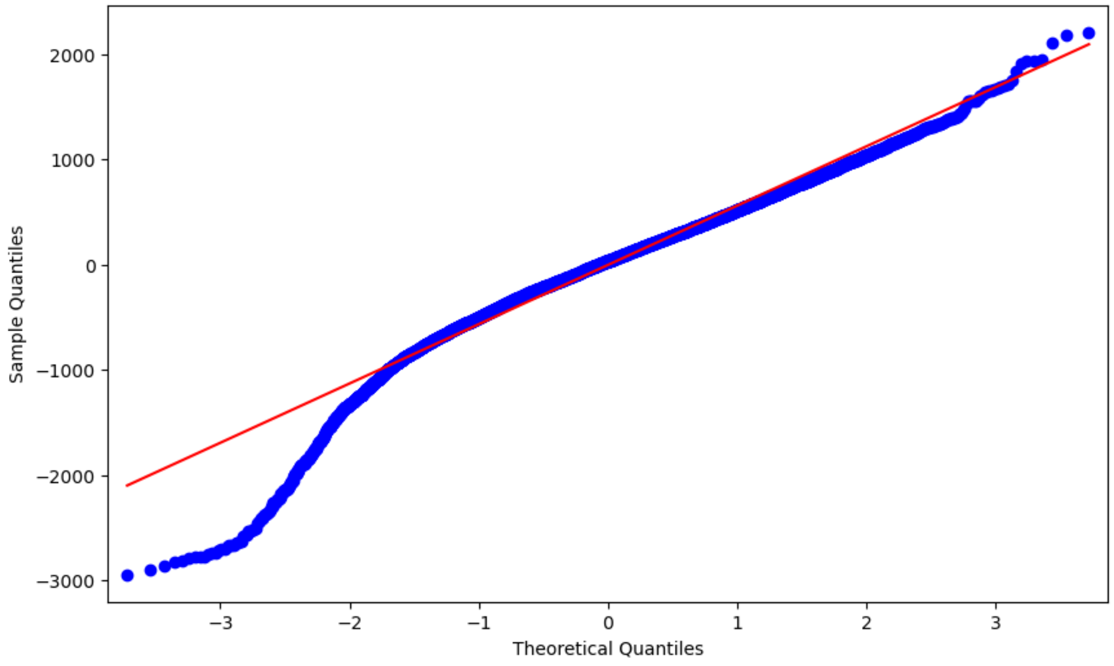


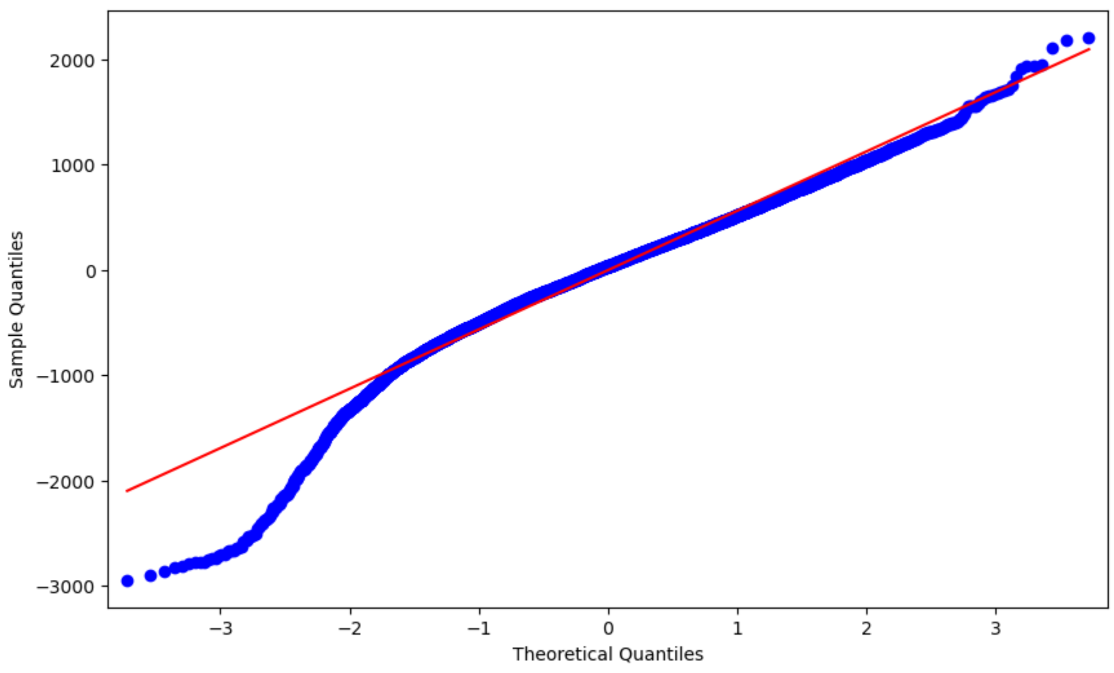
From the model fit plot above, the prediction it provides is fine but there's still a lot of room for improvement, so we might want to try a different set of features to see if it gives better predictions (For example, using 'PREG_LEN' and 'M_Ht_In', which are the best features from the decision tree).

[27]:
```
from statsmodels import graphics
resid = freq_res.resid_deviance.copy()
graphics.gofplots.qqplot(resid, line='r')
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993:
UserWarning: marker is redundantly defined by the 'marker' keyword argument and
the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```

[27]:





The Q-Q Plots is rougly linear, which menas that the assumption that the residuals are normally disbritued is satisfied.

6.3.2 Uncertainty Quantification

The parameter we are estimating is a fixed quantity but our estimate is random, because it depends on our data and the data is random. According to the frequentist model, the 95% confidence interval for the intercept is [2947.911,3081.721], for the coefficient of MAGER is [0.039 , 4.081], for the coefficient of PREVIS is [15.720 , 21.417]. The standard error for the intercept is 34.136, for the coefficient of MAGER is 1.031, for the coefficeint of PREVIS is 1.454.

6.4 Bayesian Regression

[28]:
```
dbwt = np.array(birth['DBWT'])
mager = np.array(birth['MAGER'])
previs = np.array(birth['PREVIS'])
```

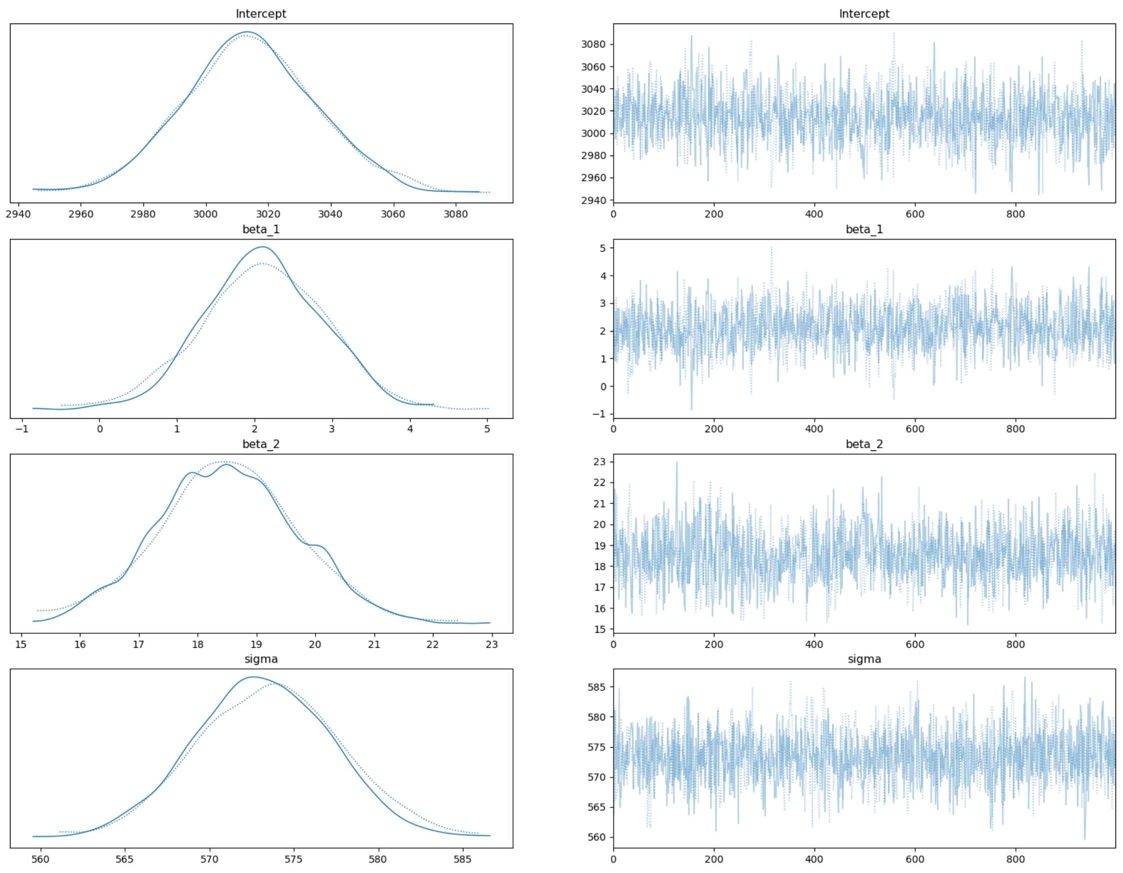
Choice of Priors:
For standard deviance, we set the prior to be exponential(0.01), because this must be nonnegative so exponential distribution is a good fit; according to the histogram we plotted in EDA, the SD of the birth weights is about 6, so the parameter 0.01 makes sense. For intercept and coefficients, we reference the result from the frequentist model and set relatively larger variances so that the posterior will be more dependent on the data instead of the priors.

[29]:
```
with pm.Model() as bayes_model:
    #define the priors
    sigma = pm.Exponential('sigma', lam=0.01)
    intercept = pm.Normal("Intercept", 3015, sigma=30)
    beta_1 = pm.Normal("beta_1", 2, sigma=3)
    beta_2 = pm.Normal("beta_2", 18, sigma=3)

    #likelihood
    likelihood = pm.Normal("y", mu = intercept + beta_1*mager + beta_2*previs,␣
  ↪sigma = sigma, observed = dbwt)

    #inference
    trace = pm.sample(1000, cores = 2, target_accept = 0.95,␣
  ↪return_inferencedata=True)
```

```
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (2 chains in 2 jobs)
NUTS: [beta_2, beta_1, Intercept, sigma]

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Sampling 2 chains for 1_000 tune and 1_000 draw iterations (2_000 + 2_000 draws
total) took 40 seconds.
```

[30]:
```
az.plot_trace(trace, figsize=(20, 15))
```

[30]:
```
array([[<AxesSubplot:title={'center':'Intercept'}>,
        <AxesSubplot:title={'center':'Intercept'}>],
       [<AxesSubplot:title={'center':'beta_1'}>,
        <AxesSubplot:title={'center':'beta_1'}>],
       [<AxesSubplot:title={'center':'beta_2'}>,
        <AxesSubplot:title={'center':'beta_2'}>],
       [<AxesSubplot:title={'center':'sigma'}>,
        <AxesSubplot:title={'center':'sigma'}>]], dtype=object)
```



[31]:
```
np.mean(trace.posterior["Intercept"].values)
```

[31]:
```
3014.142456984229
```

[32]:
```
np.mean(trace.posterior["beta_1"].values)
```

[32]:
```
2.1058263417649217
```

[33]:
```
np.mean(trace.posterior["beta_2"].values)
```

[33]:
```
18.516514695016042
```

We see that the posterior prediction given by the bayesian model is $W_i = 3015.256 + 2.086M_i + 18.471P_i$. This is very similar to the one given by the frequentist model. This again means that older mothers have babies with higher birth weights, and mothers who go to more prenatal visists also have babies with higher birth weights.

6.4.1 Model Checking

[34]:
```
with bayes_model:
    ppc = pm.sample_posterior_predictive(
        trace, var_names=["beta_1", "beta_2", "Intercept", "y"]
    )
```

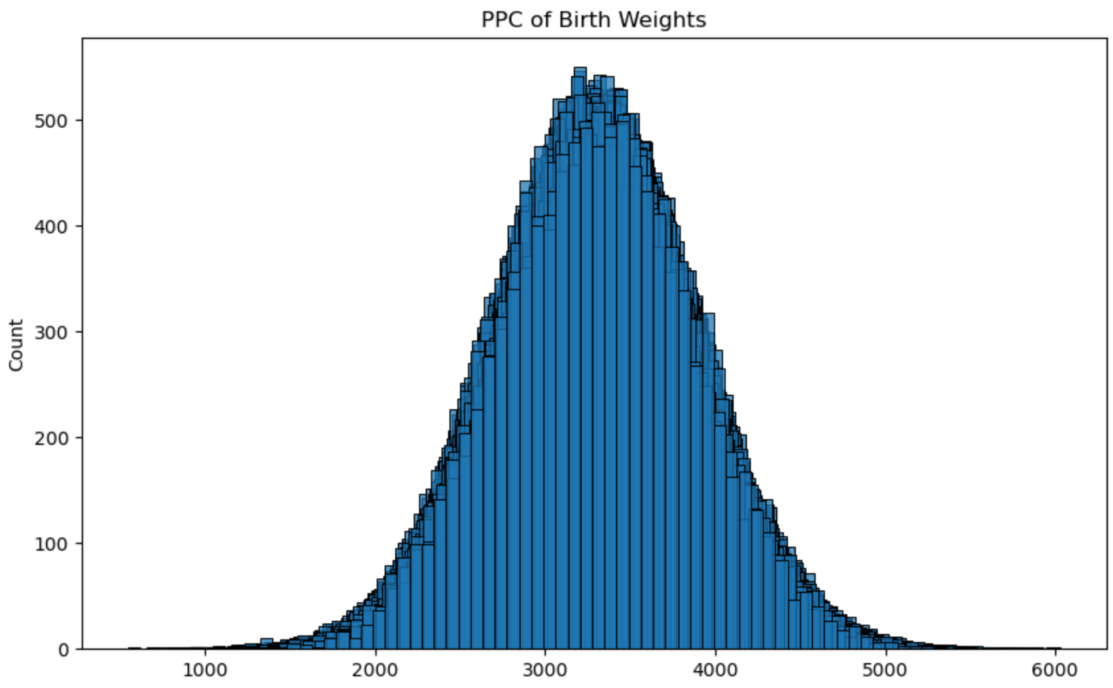
```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

[35]:
```
ppc
```

[35]:
```
{'beta_1': array([1.608638  , 3.11911254, 1.25982419, …, 2.1650516 ,
2.38930908,
        1.65561139]),
 'beta_2': array([19.25056496, 17.46079249, 19.94211726, …, 19.70850229,
        17.00464306, 17.67605704]),
 'Intercept': array([3010.62971253, 2999.66959585, 3021.44245843, …,
2997.22908972,
        3031.87069906, 3026.07162836]),
 'y': array([[2631.85897825, 2893.73461445, 3145.97074589, …, 2723.39482088,
         2480.1428695 , 4635.43100947],
        [2981.54851896, 3679.75829088, 3771.45585581, …, 3418.21215197,
         3699.64485004, 3141.13181795],
        [2845.71825798, 3153.07848564, 2422.53078112, …, 3773.20936441,
         4007.32236596, 3945.3082004 ],
        …,
        [4088.58951957, 3954.3746849 , 2068.08594291, …, 3078.62229034,
         3354.71237771, 3598.06527111],
        [3653.94283063, 3454.13215897, 4321.2614436 , …, 3852.82097892,
         3483.37595291, 2656.09877589],
        [3401.96189087, 3294.24525591, 3405.46915864, …, 3055.15488235,
         3489.73823729, 2898.96480928]])}
```

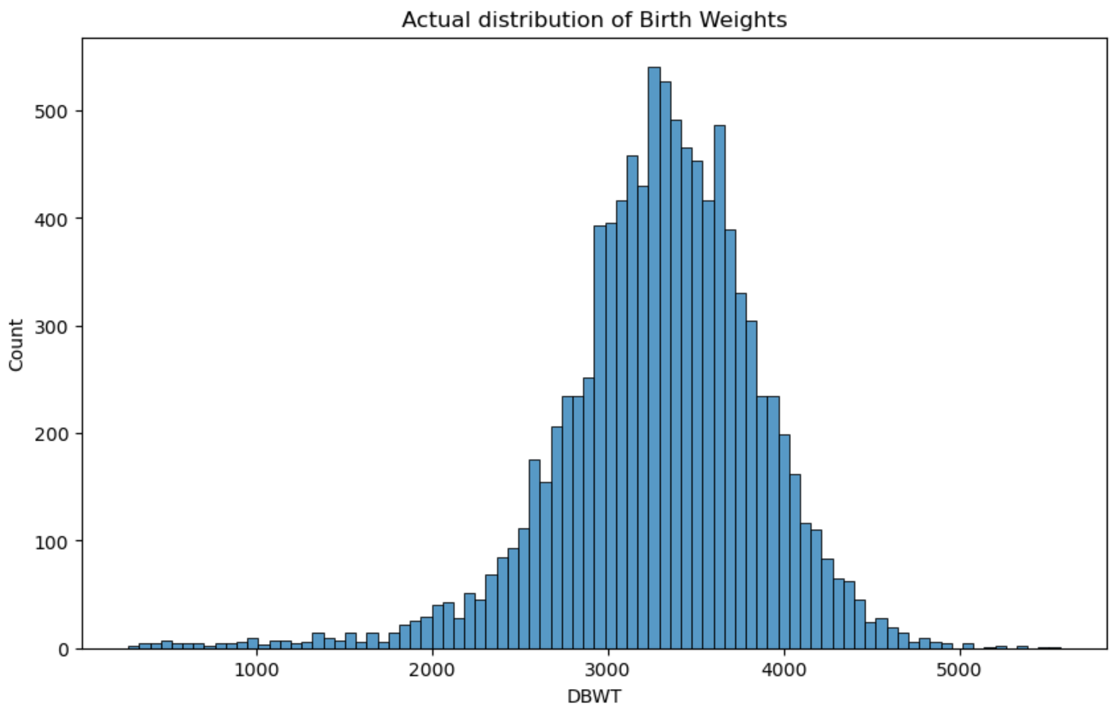
[36]:
```
for i in np.arange(100):
    sns.histplot(ppc['y'][i])
plt.title("PPC of Birth Weights")
```

[36]:
```
Text(0.5, 1.0, 'PPC of Birth Weights')
```



[37]:
```
#Actual birthweight histogram
sns.histplot(birth['DBWT'])
plt.title("Actual distribution of Birth Weights")
```

[37]:
```
Text(0.5, 1.0, 'Actual distribution of Birth Weights')
```

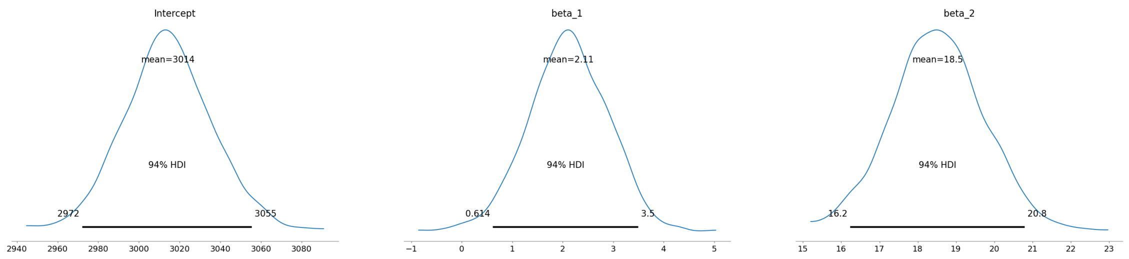


As we can see from the plots above, the PPC samples have a similar distribution as our data, so the model is a reasonable fit for the data.

6.4.2 Uncertainty Quantification

```
[38]: az.plot_posterior(trace, ['Intercept', 'beta_1', 'beta_2'], round_to = 3)
```

```
[38]: array([<AxesSubplot:title={'center':'Intercept'}>,
             <AxesSubplot:title={'center':'beta_1'}>,
             <AxesSubplot:title={'center':'beta_2'}>], dtype=object)
```



We see that the highest density interval (another term for credible interval) for the intercept is [2974, 3053]. The HDI for the coefficient of MAGER is [0.605, 3.41]. The HDI for the coefficeint of PREVIS is [16.1, 20.8].