Mid-Distance Performance Analysis

This is an analysis of the relationships between middle-distance events. I'm mostly interested in the 800, but the general idea is to be able to predict a certain race's outcome based on your performance in other events. Particularly, being able to predict your 800m time based off of your 400m and/or 1500m/mile time based off the current data.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from glob import glob
from model import BivariateModel, MultivariateModel, plot_bivariate_eda
import statsmodels.formula.api as smf

sns.set_theme(style='whitegrid')
```

Custom Functions

In [ ]:
```
def dist_viz(event: str):
    plt.figure(figsize=(10,5))
    plt.suptitle(f'{event.split("_")[1]}m Distributions')

    plt.subplot(1,2,1)
    plt.title(f'Raw Data\nn = {len(df_raw):,}')
    sns.histplot(df_raw[event],
                 kde=True,
                 bins=int(np.ceil(np.log2(len(df_raw)) + 1)),
                 line_kws={'linewidth': 2.5})

    plt.subplot(1,2,2)
    plt.title(f'Cleaned Data\nn = {len(df_cleaned):,}')
    sns.histplot(df_cleaned[event],
                 kde=True,
                 bins=int(np.ceil(np.log2(len(df_cleaned)) + 1)),
                 line_kws={'color': '#002051',
                         'linewidth': 2.5},
                 color='#002051',
                 label='Combined',
                 alpha=0.3)
    sns.histplot(df_f[event],
                 kde=True,
                 bins=int(np.ceil(np.log2(len(df_f)) + 1)),
                 line_kws={'color': '#fdea45',
                         'linewidth': 2.5},
                 color='#fdea45',
                 label='Women',
                 alpha=0.4)
    sns.histplot(df_m[event],
                 kde=True,
```

```
                bins=int(np.ceil(np.log2(len(df_m)) + 1)),
                line_kws={'color': '#7f7c75',
                          'linewidth': 2.5},
                color='#7f7c75',
                label='Men',
                alpha=0.4)
    plt.legend()
    plt.show()


def make_quantile_table(df: pd.DataFrame, col: str):

    quantiles = [0.05, 0.2, 0.4, 0.6, 0.8, 0.95]

    time_quantiles = np.quantile(df[col], q=0.05), \
        np.quantile(df[col], q=0.2), \
        np.quantile(df[col], q=0.4), \
        np.quantile(df[col], q=0.6), \
        np.quantile(df[col], q=0.8), \
        np.quantile(df[col], q=0.95)

    return pd.DataFrame({'quantile': quantiles,
                         'time_800m': time_quantiles}).style.format(precision=2)
```

TFRRS Data

The links in the following dictionary have been obtained by going to the TFRRS database archives, finding the men's 800m for the desired season/year, inspecting the table, and finding the src argument of the turbo-frame tag with id = 'list_data'. There are some patterns in the URLs, but the only consistent ones are the ones I've already automated. There are still random differences at times between divisions and between indoor and outdoor seasons.

Example:

> `<turbo-frame id="list_data" src="https://tf.tfrrs.org/list_data/3901?other_lists= https%3A%2F%2Fm.tfrrs.org%2Flists%2F3901%2F2022_2023_NCAA_Div_I_Indoor_ _FINAL&limit=2000&event_type=53&year=&gender=">`
> `... </turbo-frame>`

In [ ]:
```
di_dict = {
    'indoor_2022': 'https://tf.tfrrs.org/list_data/3492?other_lists=https%3A%2F%2Ft
    'outdoor_2022': 'https://tf.tfrrs.org/list_data/3711?other_lists=https%3A%2F%2F
    'indoor_2023': 'https://tf.tfrrs.org/list_data/3901?other_lists=https%3A%2F%2Fm
    'outdoor_2023': 'https://tf.tfrrs.org/list_data/4044?other_lists=https%3A%2F%2F
    'indoor_2024': 'https://tf.tfrrs.org/list_data/4364?other_lists=https%3A%2F%2Fm
}
```

```
def create_dx_dict(dict: dict, division: str) -> dict:
    '''Updates the values of the DI dictionary to correspond with DII and DIII URLs

    Parameters:
        dict (dict): the NCAA DI Dictionary of URLs
        division (str): 'II' or 'III' to match DII or DIII

    Returns:
        dx_dict (dict): a dict with updated values to match the corresponding NCAA

    dx_dict = {}
    for season in dict:
        new_url = dict[season].replace('_I_', f'_{division}_')

        table_id = dict[season].split('?')[0][-4:]
        new_table_id = str(int(table_id) + len(division))
        new_url = new_url.replace(table_id, new_table_id)

        dx_dict[season] = new_url

    return dx_dict


dii_dict = create_dx_dict(di_dict, 'II')
dii_dict['outdoor_2022'] = 'https://tf.tfrrs.org/list_data/3595?other_lists=https%3

diii_dict = create_dx_dict(di_dict, 'III')
diii_dict['indoor_2024'] = 'https://tf.tfrrs.org/list_data/4366?other_lists=https%3

naia_dict = {
    'indoor_2022': 'https://tf.tfrrs.org/list_data/3495?other_lists=https%3A%2F%2Fw
    'outdoor_2022': 'https://tf.tfrrs.org/list_data/3596?other_lists=https%3A%2F%2F
    'indoor_2023': 'https://tf.tfrrs.org/list_data/3904?other_lists=https%3A%2F%2Fw
    'outdoor_2023': 'https://tf.tfrrs.org/list_data/4046?other_lists=https%3A%2F%2F
    'indoor_2024': 'https://tf.tfrrs.org/list_data/4368?other_lists=https%3A%2F%2Fm
}

njcaa_dict = {
    'indoor_2022': 'https://tf.tfrrs.org/list_data/3496?other_lists=https%3A%2F%2Ft
    'outdoor_2022': 'https://tf.tfrrs.org/list_data/3717?other_lists=https%3A%2F%2F
    'indoor_2023': 'https://tf.tfrrs.org/list_data/3905?other_lists=https%3A%2F%2Fm
    'outdoor_2023': 'https://tf.tfrrs.org/list_data/4201?other_lists=https%3A%2F%2F
    'indoor_2024': 'https://tf.tfrrs.org/list_data/4367?other_lists=https%3A%2F%2Fm
}
```

800m and 400m + 1500m

In [ ]:

```
from scrape_tfrrs import TFRRSScraper, read_tfrrs_data
from glob import glob

if not glob(f'data/tfrrs*800_400_1500*.csv'):
    tf_scraper_di = TFRRSScraper(di_dict,
                                 division='di',
                                 outcome_event='800',
                                 predictor_events=['400', '1500'],
```

```
                            merge=True)
    tf_scraper_dii = TFRRSScraper(dii_dict,
                            division='dii',
                            outcome_event='800',
                            predictor_events=['400', '1500'],
                            merge=True)
    tf_scraper_diii = TFRRSScraper(diii_dict,
                            division='diii',
                            outcome_event='800',
                            predictor_events=['400', '1500'],
                            merge=True)
    tf_scraper_naia = TFRRSScraper(naia_dict,
                            division='naia',
                            outcome_event='800',
                            predictor_events=['400', '1500'],
                            merge=True)
    tf_scraper_njcaa = TFRRSScraper(njcaa_dict,
                            division='njcaa',
                            outcome_event='800',
                            predictor_events=['400', '1500'],
                            merge=True)

    tf_scraper_di()
    tf_scraper_dii()
    tf_scraper_diii()
    tf_scraper_naia()
    tf_scraper_njcaa()
else:
    print('[INFO] TFRRS data found, importing datasets.')
    df_tfrrs = read_tfrrs_data(events=['800', '400', '1500'], drop_na=True).reset_i
```

```
[INFO] TFRRS data found, importing datasets.
C:\Users\mitch\AppData\Local\Temp\ipykernel_3300\4149257923.py:38: DeprecationWarnin
g: Call to deprecated function (or staticmethod) read_tfrrs_data. (
    Not super functional right now, will figure something out eventually. Since I ad
ded the merge
    argument in the initializer, you don't really need this as much. Still useful fo
r combining
    different divisions, but use with caution.

    Issue #1: All data files with multiple events need to be specified in the same o
rder as they
    were saved in the name of the CSV.

    E.g. If you ran the TFRRSScraper with outcome = '800', and predictors = ['400',
'1500'], the
    CSV would be saved with the name: 'tfrrs_*_800_400_1500_*.csv'. To read that fil
e, the list
    provided must follow the same order: ['800', '400', '1500']. If ['800', '1500',
'400'] is
    entered, the files won't be read.
)
  df_tfrrs = read_tfrrs_data(events=['800', '400', '1500'], drop_na=True).reset_inde
x(drop=True)
```

In [ ]:
```
df_tfrrs
```

Out[ ]:

	athlete_team	sex	season	time_800	time_400	time_1500
0	Abdo, Nick | Fredonia	m	outdoor_2022	116.39	49.78	257.25
1	Alberti, Abby | Central College	f	indoor_2024	145.95	65.16	328.49
2	Albright, Cooper | Christopher Newport	m	outdoor_2022	114.93	50.65	252.87
3	Allard, Harrison | Wis.-Platteville	m	indoor_2024	121.20	55.68	280.25
4	Allen, Tommy | St. John's (Minn.)	m	outdoor_2022	114.56	52.59	232.03
...	...	...	...	...	...	...
2036	Woodard, Tiwan | Louisburg College	m	outdoor_2022	130.58	59.06	271.51
2037	Wright, Joseph Asahi | Iowa Central CC	m	outdoor_2023	114.92	53.69	240.53
2038	Yaley, Cassidy | Southeastern CC	f	outdoor_2023	155.70	68.58	327.31
2039	Young, Elijah | Glendale CC	m	outdoor_2022	121.11	52.65	271.35
2040	hernandez, brisa | North Central Texas College	f	outdoor_2023	172.34	71.90	342.27

2041 rows × 6 columns

In [ ]:
```
df_tfrrs.describe(percentiles=[.05, .25, .5, .75, .95]).style.format(precision=2)
```

Out[ ]:

	time_800	time_400	time_1500
count	2041.00	2041.00	2041.00
mean	133.61	59.11	294.15
std	18.17	7.38	42.30
min	105.82	46.30	221.28
5%	111.32	50.06	238.93
25%	118.74	53.11	261.00
50%	130.56	57.78	287.00
75%	144.91	64.13	321.26
95%	166.58	72.28	371.21
max	241.88	100.02	529.57

MileSplit Data

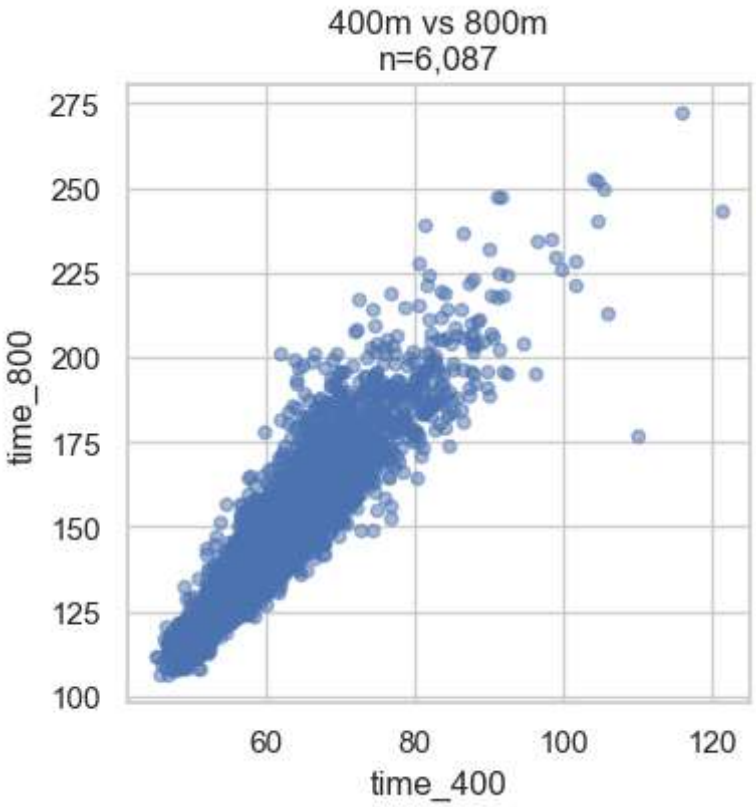
In [ ]:
```
from scrape_milesplit import MileSplitScraper
from glob import glob
import pandas as pd

if not glob('data/milesplit*.csv'):
    ms_1600 = MileSplitScraper(predictor_event='1600m')
    ms_1600.download_and_export(2020, 2024)
    ms_mile = MileSplitScraper(predictor_event='Mile')
    ms_mile.download_and_export(2020,2024)
    ms_400 = MileSplitScraper(predictor_event='400m')
    ms_400.download_and_export(2020, 2024)
else:
    print('[INFO] MileSplit data found in data directory')
    df_milesplit_400 = pd.read_csv('data/milesplit_indoor_2020-outdoor_2024_400m_20
    df_milesplit_mile = pd.read_csv('data/milesplit_indoor_2020-outdoor_2024_Mile_2
    df_milesplit_1600 = pd.read_csv('data/milesplit_indoor_2020-outdoor_2024_1600m_
```

```
[INFO] MileSplit data found in data directory
```

800m vs 400m

In [ ]:
```
df_milesplit_400.plot(
    kind='scatter',
    x='time_400',
    y='time_800',
    figsize=(4,4),
    alpha = 0.5,
    title = f'400m vs 800m\nn={len(df_milesplit_400):,}'
)
```

Out[ ]:
```
<Axes: title={'center': '400m vs 800m\nn=6,087'}, xlabel='time_400', ylabel='time_
800'>
```



800m vs 1500m

In [ ]:
```
df_milesplit_1500 = pd.merge(left=df_milesplit_1600,
                             right=df_milesplit_mile,
                             on=['athlete_team', 'time_800', 'season', 'sex'],
                             how='left',
                             suffixes=['_1600', '_mile'])

df_milesplit_1500['time_1500'] = df_milesplit_1500[['time_1500_1600', 'time_1500_mi
df_milesplit_1500 = df_milesplit_1500[['athlete_team', 'time_800', 'time_1500', 'se

df_milesplit_1500
```

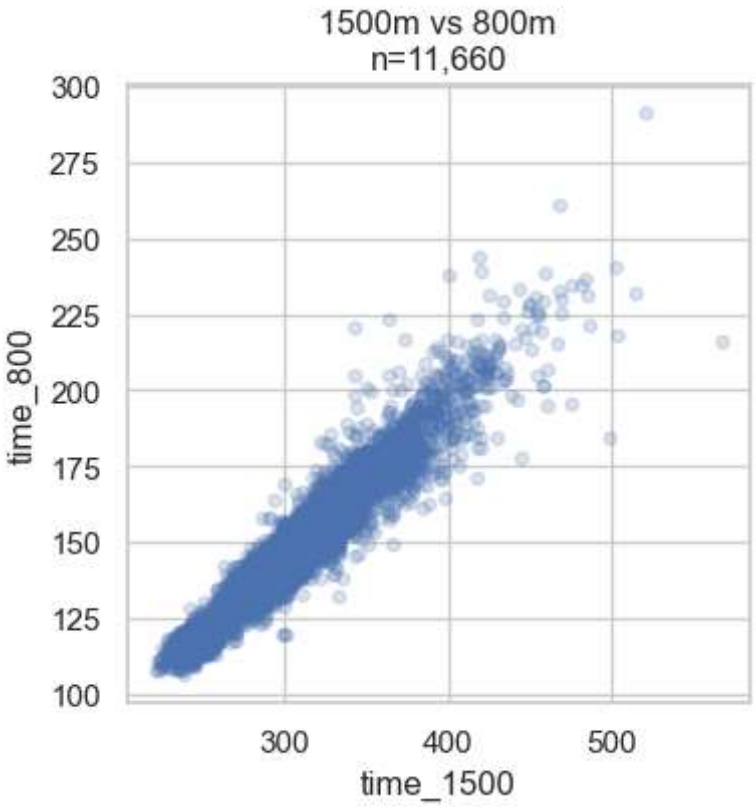
Out[ ]:

	athlete_team	time_800	time_1500	sex	season
0	Juliette Whittaker MD Mount De Sales Academy ...	123.01	264.60	f	indoor_2020
1	Michaela Rose VA Faith In Action Ministries A...	127.07	285.92	f	indoor_2020
2	Carina Napoleon OH Thom. Worthington 2021	130.59	278.10	f	indoor_2020
3	Lea Hatcher WV WV Flyers 2022	131.49	276.97	f	indoor_2020
4	Taylor James CO Niwot High School 2021	131.59	281.44	f	indoor_2020
...	...	...	...	...	...
11655	Mason Johnson TX Barbers Hill North MS 2028	144.54	302.74	m	outdoor_2024
11656	Conner Hill WA Toledo High School 2028	144.57	286.78	m	outdoor_2024
11657	Hayden Paqueno TX The Highlands MS nan	144.60	291.76	m	outdoor_2024
11658	Hedrick Johnathan OH Miami East JH 2028	144.63	288.28	m	outdoor_2024
11659	Miles Wharry OH Ottawa Hill MS 2028	144.63	288.69	m	outdoor_2024

11660 rows × 5 columns

In [ ]:
```
df_milesplit_1500.plot(
    kind='scatter',
    x='time_1500',
    y='time_800',
    figsize=(4,4),
    alpha = 0.2,
    title = f'1500m vs 800m\nn={len(df_milesplit_1500):,}'
)
```

Out[ ]:
```
<Axes: title={'center': '1500m vs 800m\nn=11,660'}, xlabel='time_1500', ylabel='time_800'>
```



800m vs 400m + 1500m

In [ ]:
```
df_ms = pd.merge(
    left=df_milesplit_400,
    right=df_milesplit_1500,
    how='inner',
    left_on=['athlete_team', 'sex', 'season', 'time_800'],
    right_on=['athlete_team', 'sex', 'season', 'time_800'],
)[['athlete_team', 'sex', 'season', 'time_800', 'time_400', 'time_1500']]

df_ms
```

Out[ ]:

	athlete_team	sex	season	time_800	time_400	time_1500
0	Claire Frazier Bolton AL McGill-Toolen Cathol...	f	indoor_2020	132.15	57.82	282.94
1	Erin Reidy IL Downers Grove (South) 2020	f	indoor_2020	132.43	62.00	287.81
2	Maddie Ullom OH Mason 2020	f	indoor_2020	132.85	60.70	298.51
3	Presley Miles AL Saint James 2021	f	indoor_2020	134.69	58.67	286.41
4	Lindsay Colflesh NJ Haddonfield Memorial HS 2021	f	indoor_2020	135.31	60.03	297.92
...	...	...	...	...	...	...
811	Ryan WULINSKY NC Marvin Ridge Middle School 2028	m	outdoor_2024	135.10	58.16	277.58
812	Wynn Andrews AL Pizitz 2028	m	outdoor_2024	135.99	58.23	267.28
813	Asa King AL Simmons Middle School 2028	m	outdoor_2024	137.57	58.35	283.21
814	Ethan Soper KS Lakewood Middle School 2028	m	outdoor_2024	138.09	58.58	287.53
815	Riley Sipe GA Sutton MS 2028	m	outdoor_2024	139.49	55.84	287.62

816 rows × 6 columns

EDA and Data Cleaning

Combined Data Sources

After doing some EDA, pretty apparent that Boosted Regression methods and/or analyzing the different sources separately will be necessary for better modelling of a wider range of speeds.

In [ ]:
```
df_raw = pd.concat([df_tfrrs, df_ms]).reset_index(drop=True)
# df_raw.to_csv('data/df_raw_800_400_1500_2024-04-14.csv', index=False)

df_f = df_raw.copy().loc[df_raw['sex'] == 'f'].reset_index(drop=True)
df_m = df_raw.copy().loc[df_raw['sex'] == 'm'].reset_index(drop=True)

df_raw
```

Out[ ]:

	athlete_team	sex	season	time_800	time_400	time_1500
0	Abdo, Nick | Fredonia	m	outdoor_2022	116.39	49.78	257.25
1	Alberti, Abby | Central College	f	indoor_2024	145.95	65.16	328.49
2	Albright, Cooper | Christopher Newport	m	outdoor_2022	114.93	50.65	252.87
3	Allard, Harrison | Wis.-Platteville	m	indoor_2024	121.20	55.68	280.25
4	Allen, Tommy | St. John's (Minn.)	m	outdoor_2022	114.56	52.59	232.03
...	...	...	...	...	...	...
2852	Ryan WULINSKY NC Marvin Ridge Middle School 2028	m	outdoor_2024	135.10	58.16	277.58
2853	Wynn Andrews AL Pizitz 2028	m	outdoor_2024	135.99	58.23	267.28
2854	Asa King AL Simmons Middle School 2028	m	outdoor_2024	137.57	58.35	283.21
2855	Ethan Soper KS Lakewood Middle School 2028	m	outdoor_2024	138.09	58.58	287.53
2856	Riley Sipe GA Sutton MS 2028	m	outdoor_2024	139.49	55.84	287.62

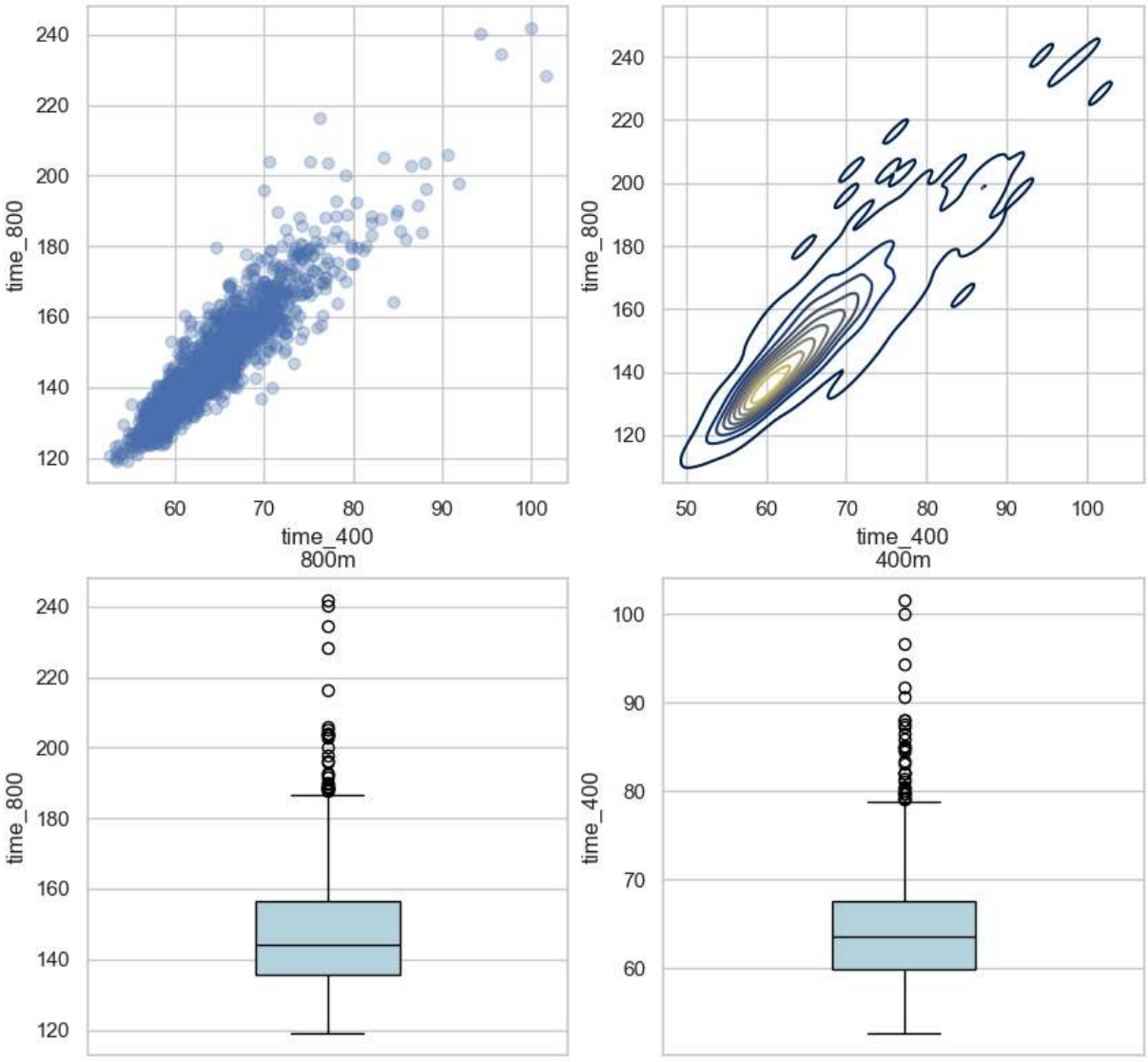
2857 rows × 6 columns

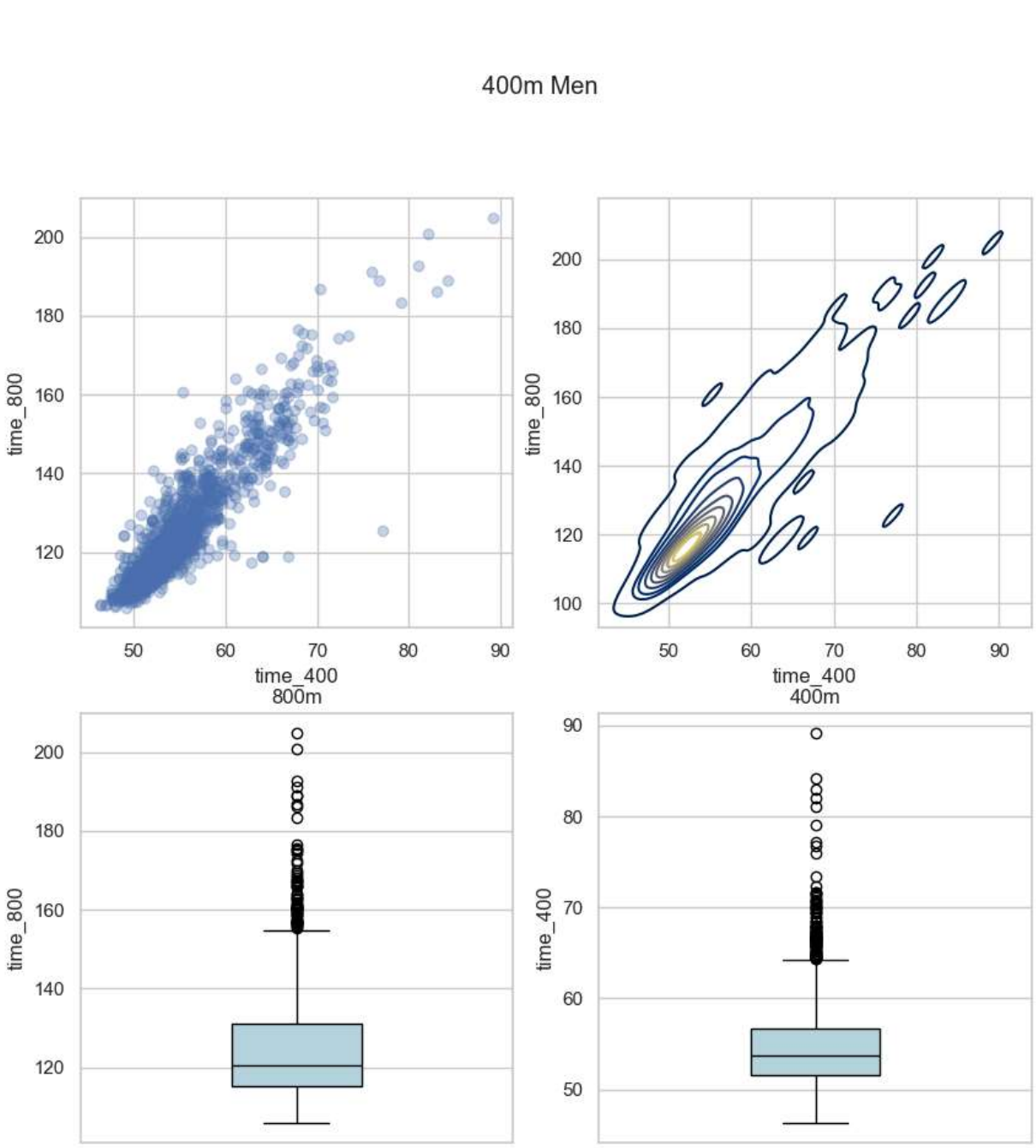
800m vs 400m

Before Cleaning

In [ ]:
```
plot_bivariate_eda(df_f, outcome_event=800, predictor_event=400, title='400m Women'
plot_bivariate_eda(df_m, outcome_event=800, predictor_event=400, title='400m Men')
```

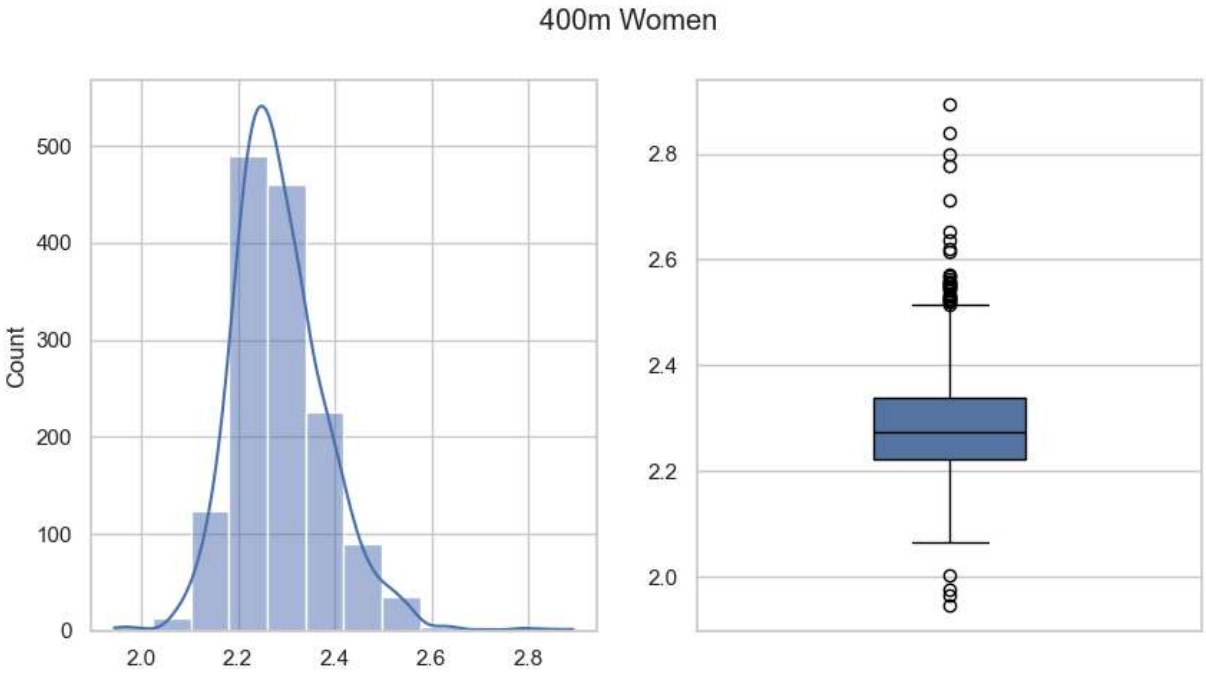
400m Women

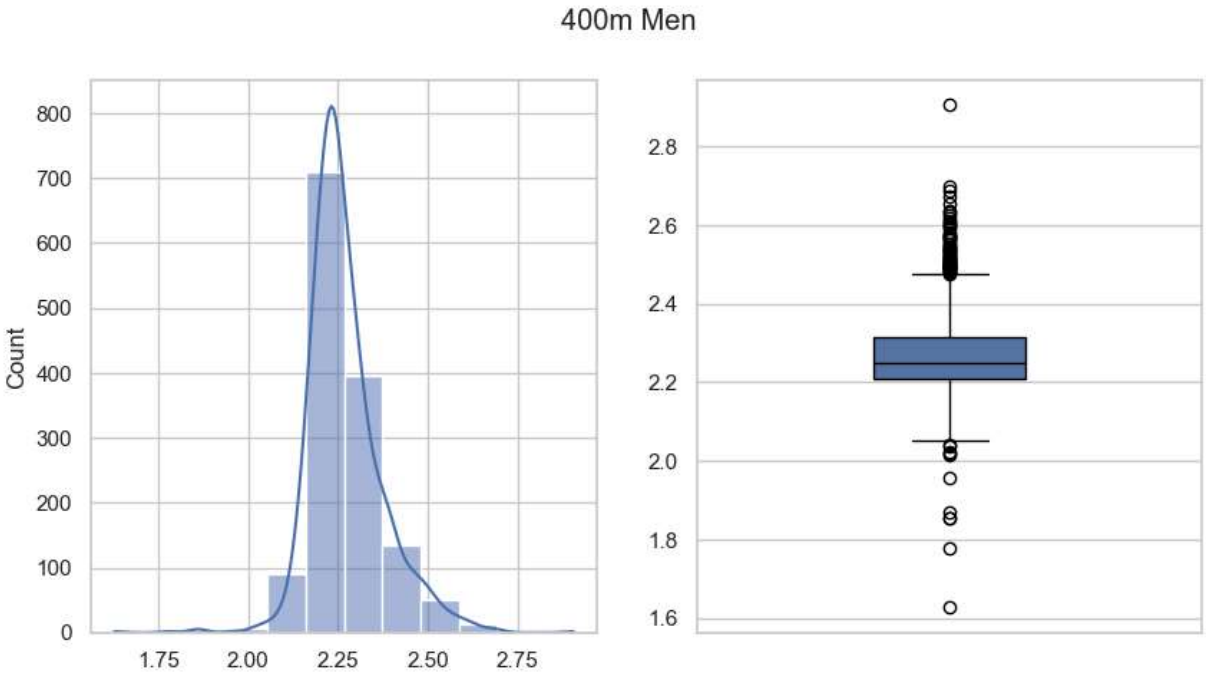

time_800
time_400
800m
time_800
time_400
400m
time_800
time_400



In [ ]:
```
# Find metric to describe outliers
metric_400_f = (df_f['time_800'] / df_f['time_400'])
metric_400_m = (df_m['time_800'] / df_m['time_400'])

plt.figure(figsize=(10,5))
plt.suptitle('400m Women')

plt.subplot(1,2,1)
sns.histplot(metric_400_f,
             bins=int(np.ceil(np.log2(len(metric_400_f)) + 1)),
             kde=True)

plt.subplot(1,2,2)
sns.boxplot(y=metric_400_f.values, width=0.3, linecolor='black')
plt.show()

plt.figure(figsize=(10,5))
```

```
plt.suptitle('400m Men')

plt.subplot(1,2,1)
sns.histplot(metric_400_m,
             bins=int(np.ceil(np.log2(len(metric_400_m)) + 1)),
             kde=True)

plt.subplot(1,2,2)
sns.boxplot(y=metric_400_m.values, width=0.3, linecolor='black')
plt.show()
```

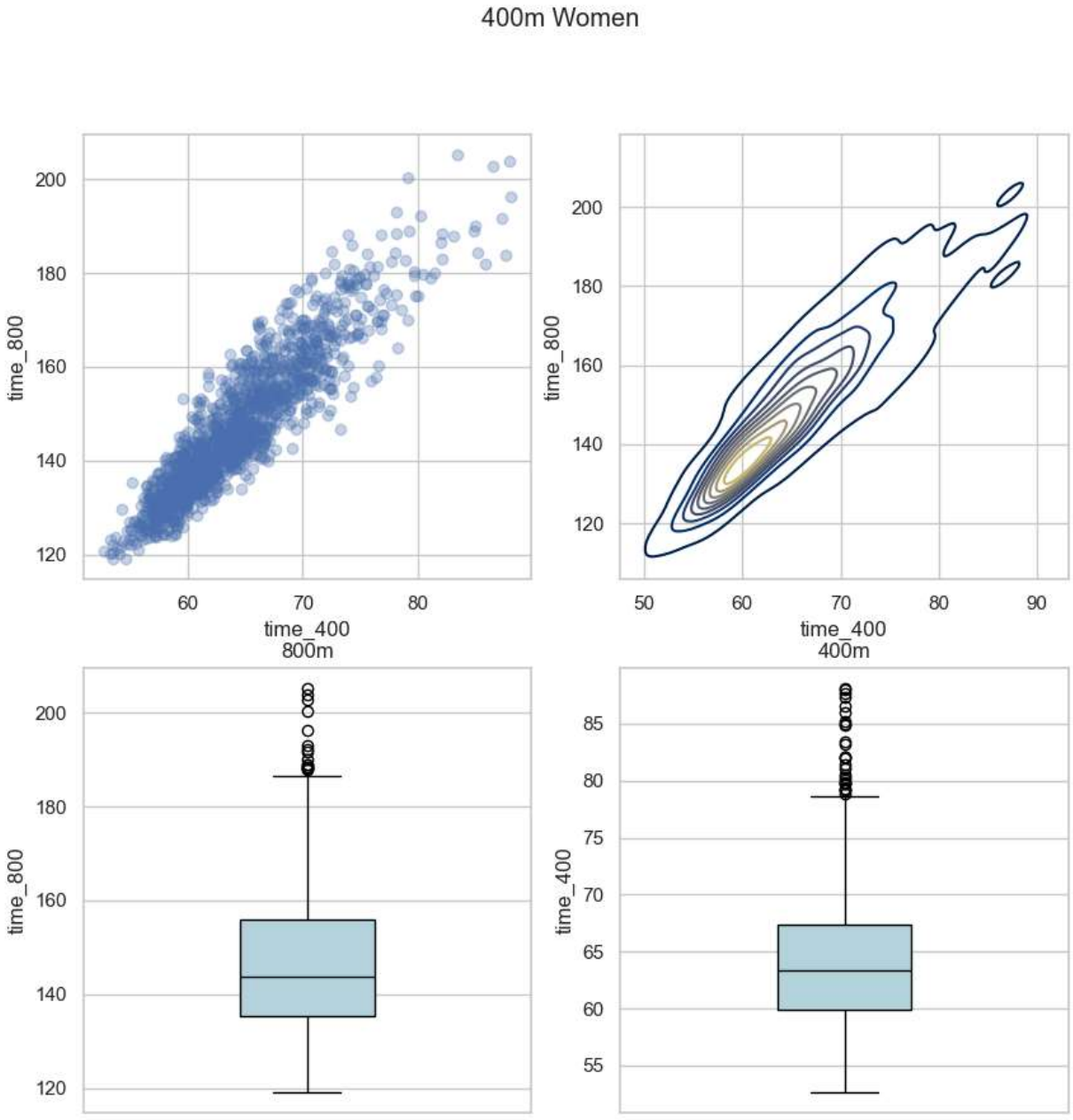


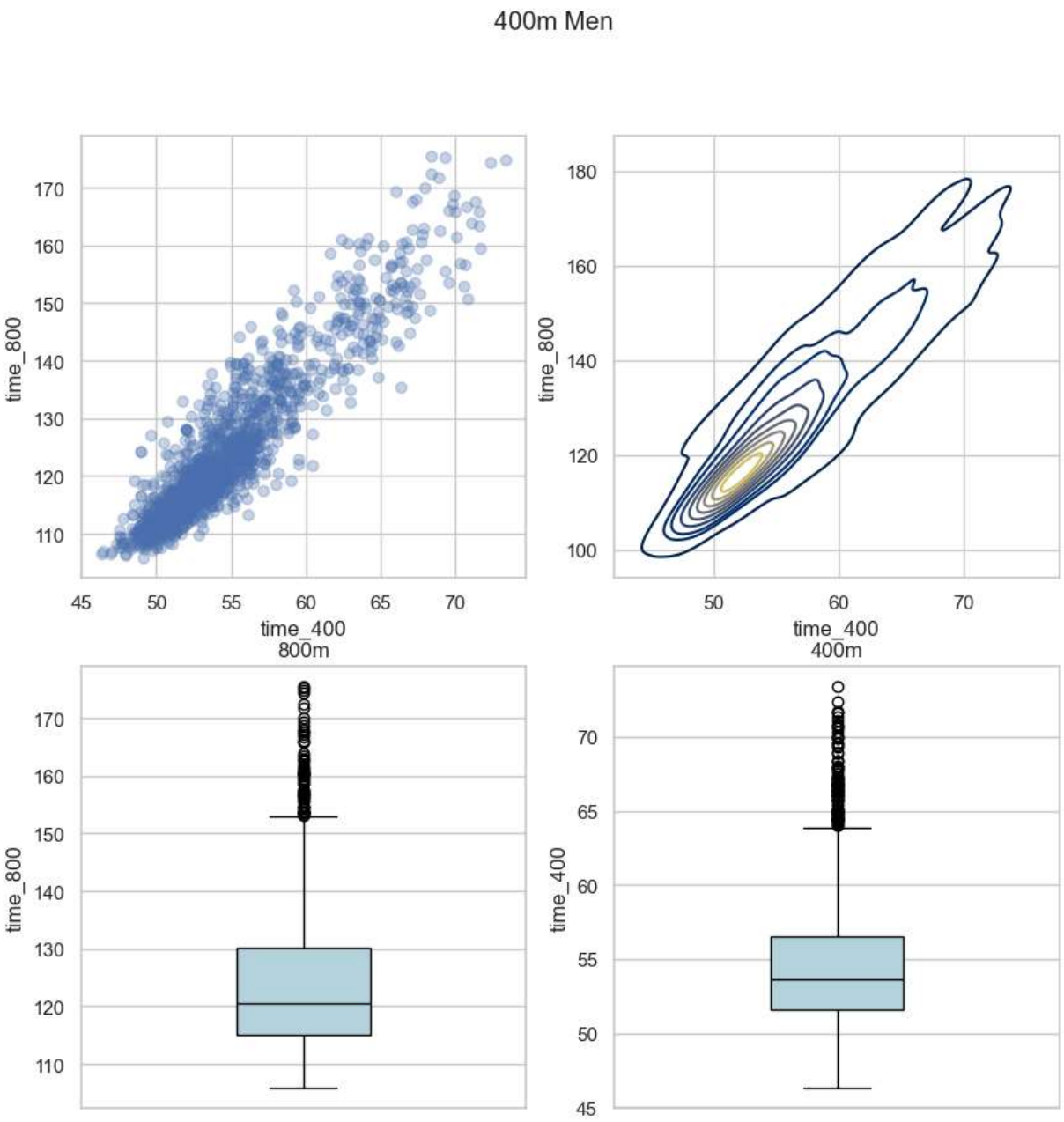


Cleaned Data

In [ ]:
```
# Remove tails that aren't representative of the rest of the data
df_f = df_f.loc[(df_f['time_800'] < 210) & (df_f['time_400'] < 90)]
df_m = df_m.loc[(df_m['time_800'] < 180) & (df_m['time_400'] < 75)]

# Remove unrealistic 400 vs 800 times/the new outlier metric
df_f = df_f.loc[(df_f['time_800'] / df_f['time_400'] > 2.0) & (df_f['time_800'] / d
df_m = df_m.loc[(df_m['time_800'] / df_m['time_400'] > 2.0) & (df_m['time_800'] / d
```

In [ ]:
```
# Visualize
plot_bivariate_eda(df_f, predictor_event=400, outcome_event=800, title='400m Women'
plot_bivariate_eda(df_m, predictor_event=400, outcome_event=800, title='400m Men')
```



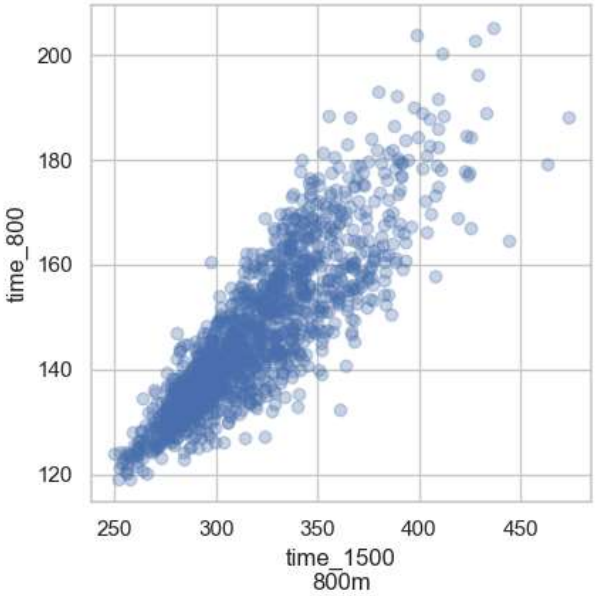


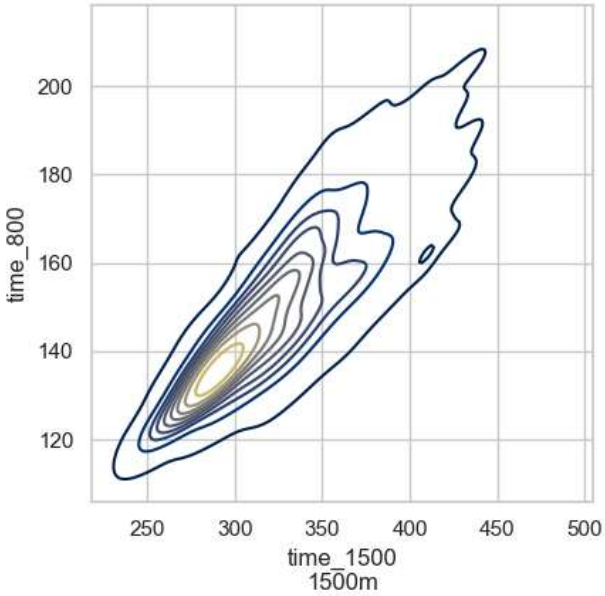
800m vs 1500m

Before Cleaning

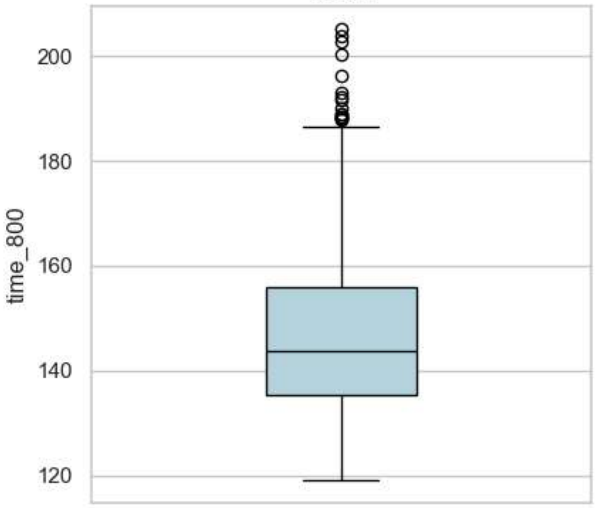
In [ ]:
```
plot_bivariate_eda(df_f, title='1500m Women', outcome_event=800, predictor_event=15
plot_bivariate_eda(df_m, title='1500m Men', outcome_event=800, predictor_event=1500
```

1500m Women

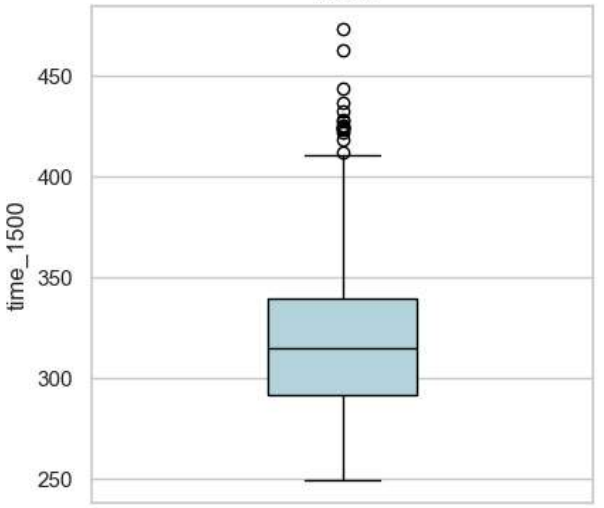


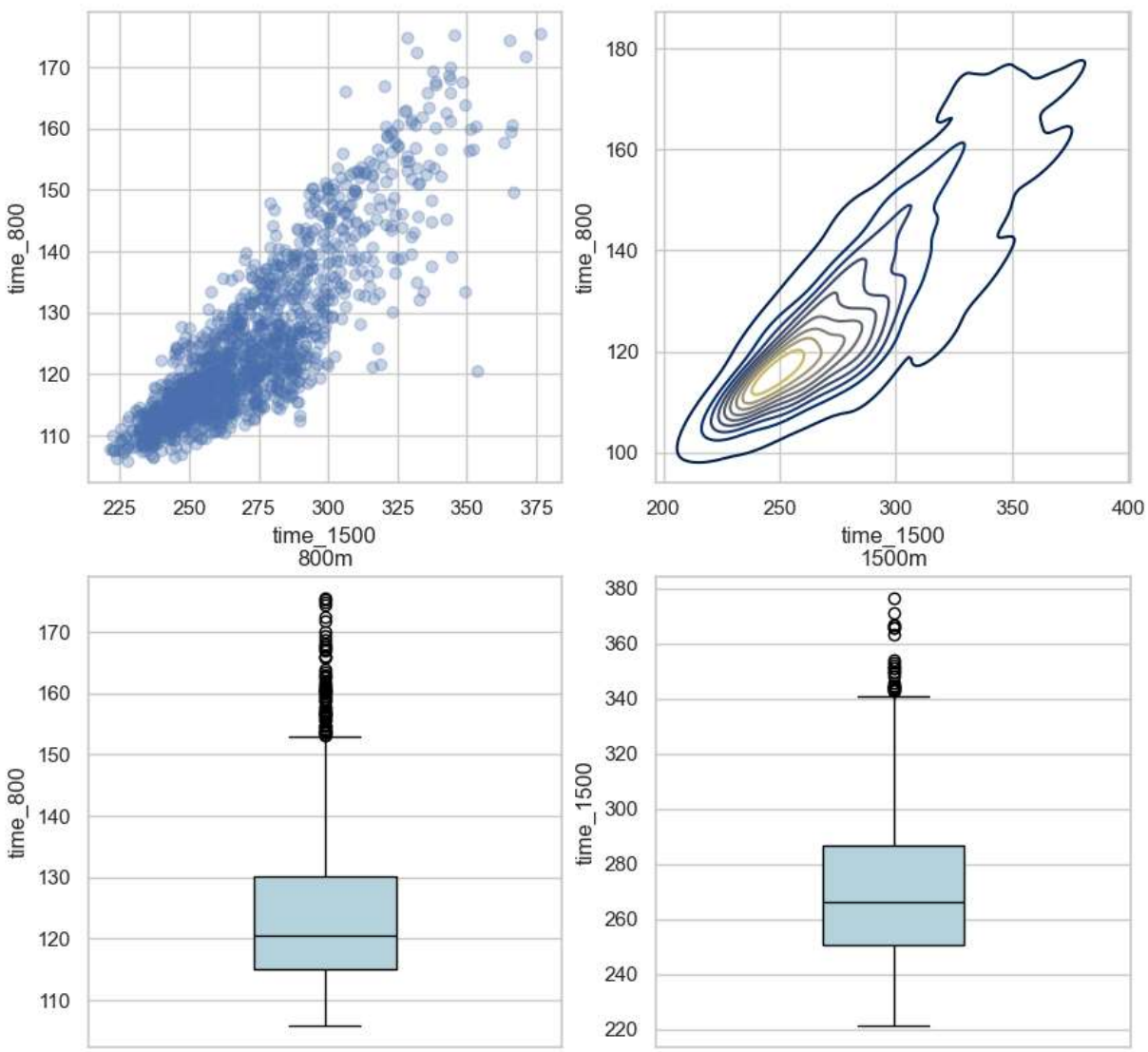


800m



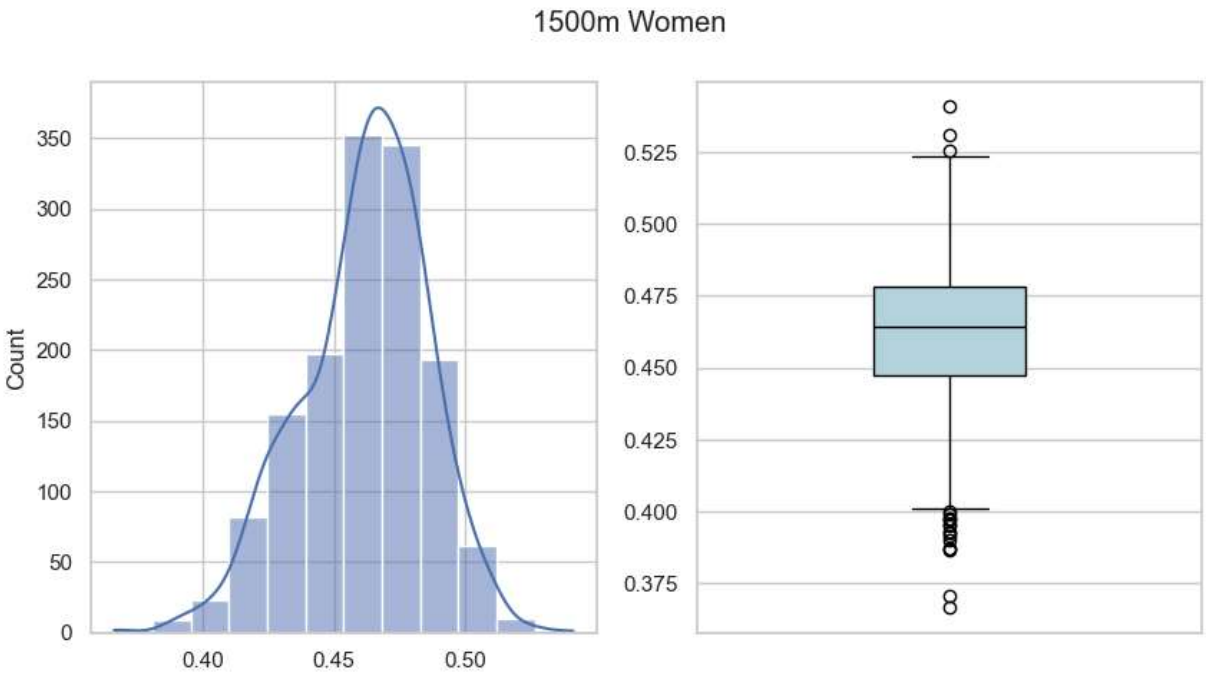
1500m

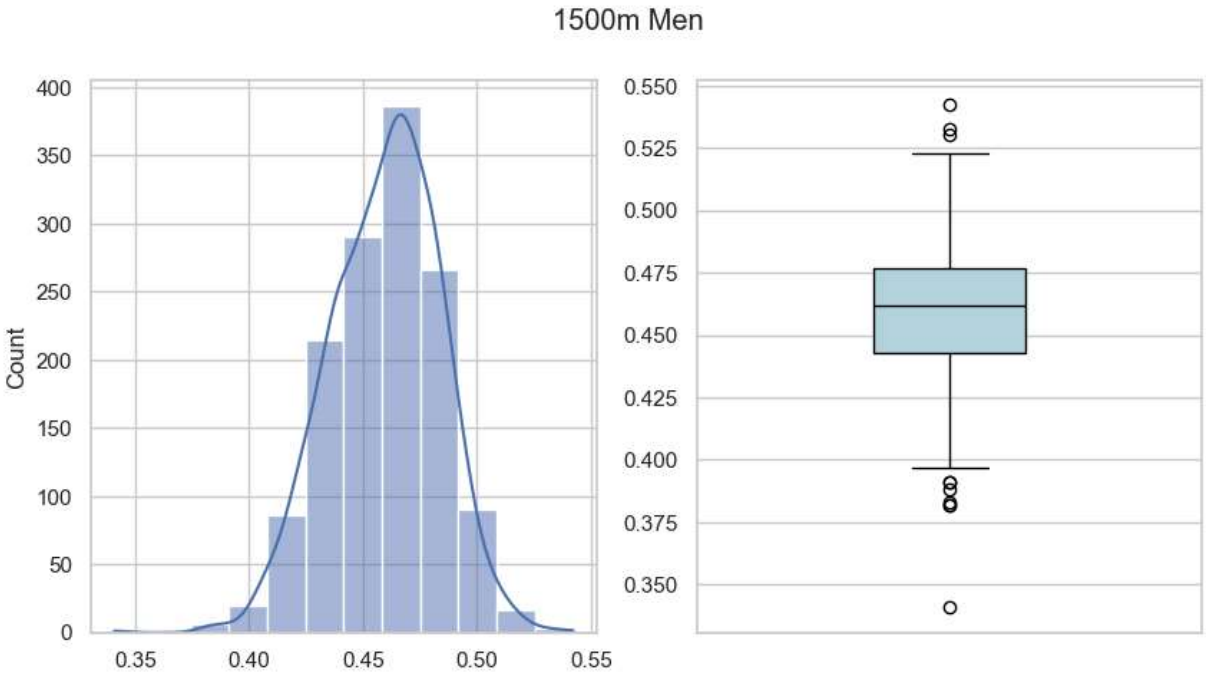




In [ ]:
```
# Find metric to describe outliers
metric_1500_f = (df_f['time_800'] / df_f['time_1500'])
metric_1500_m = (df_m['time_800'] / df_m['time_1500'])

plt.figure(figsize=(10,5))
plt.suptitle('1500m Women')

plt.subplot(1,2,1)
sns.histplot(metric_1500_f,
             bins=int(np.ceil(np.log2(len(metric_1500_f)) + 1)),
             kde=True)

plt.subplot(1,2,2)
sns.boxplot(y=metric_1500_f.values, width=0.3, color='lightblue', linecolor='black'
plt.show()

plt.figure(figsize=(10,5))
plt.suptitle('1500m Men')
```

```
plt.subplot(1,2,1)
sns.histplot(metric_1500_m,
             bins=int(np.ceil(np.log2(len(metric_1500_m)) + 1)),
             kde=True)

plt.subplot(1,2,2)
sns.boxplot(y=metric_1500_m.values, width=0.3, color='lightblue', linecolor='black'
plt.show()
```



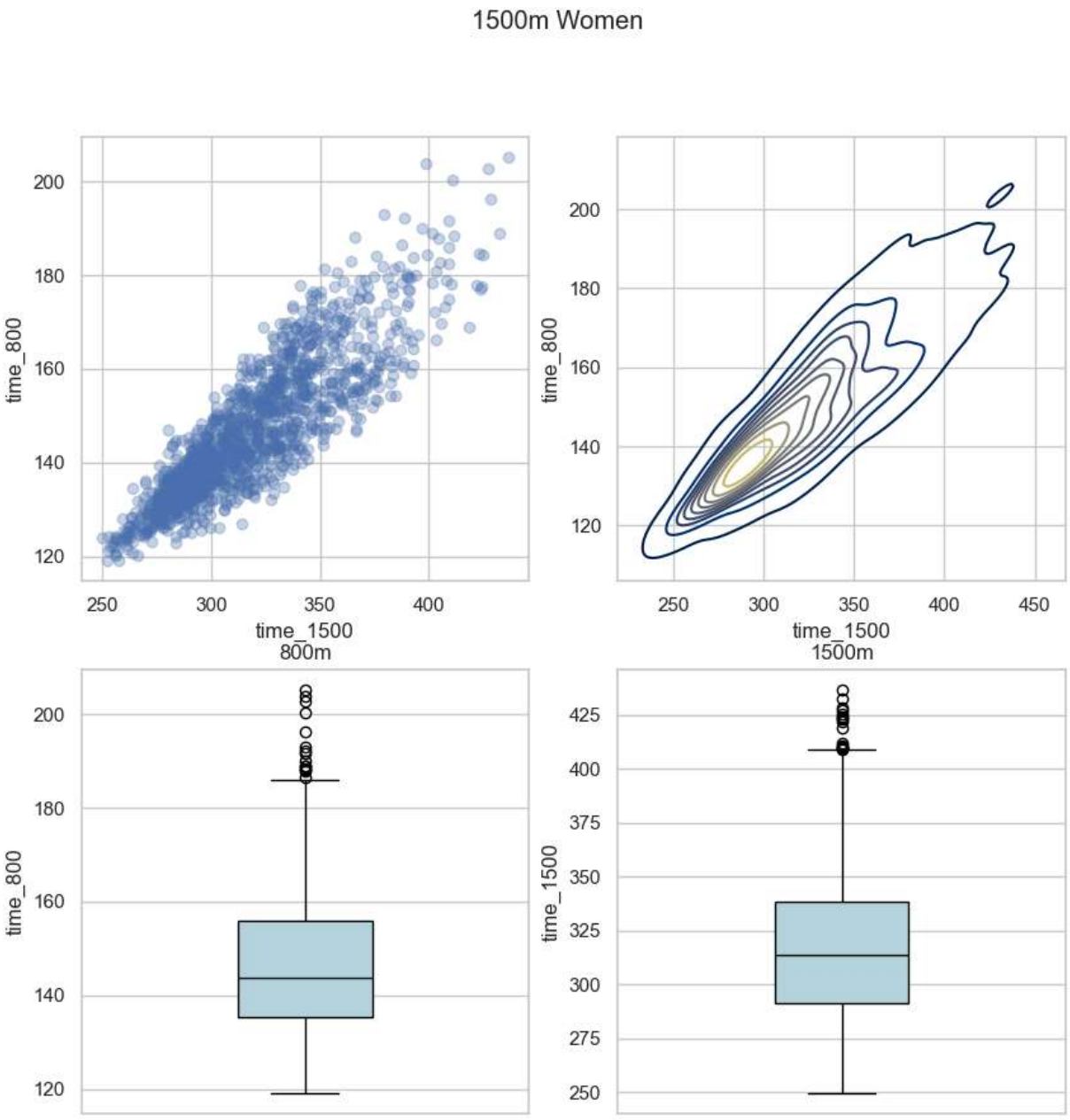


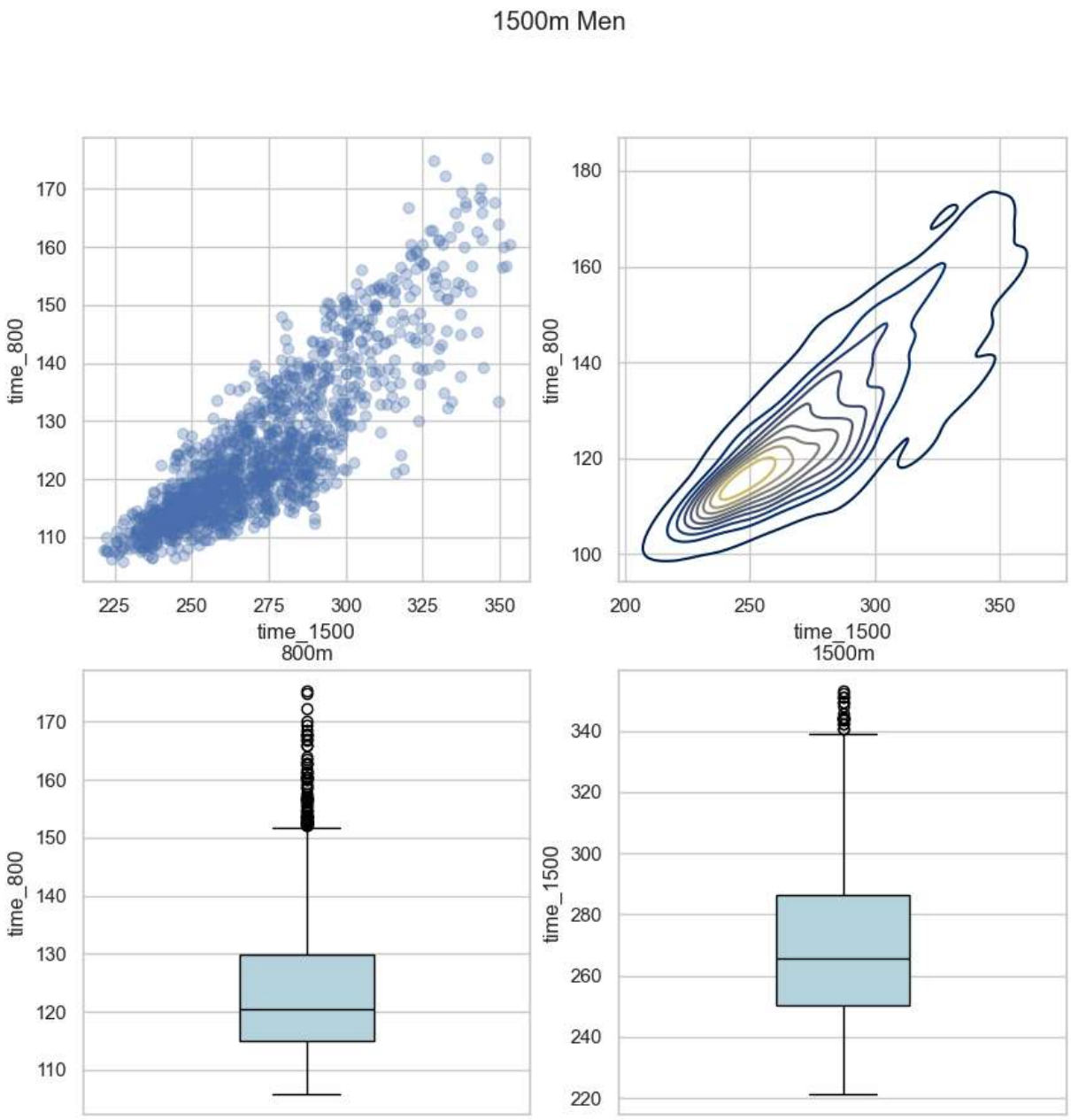
Cleaned Data

In [ ]:
```
# Remove tails that aren't representative of the rest of the data
df_f = df_f.loc[(df_f['time_800'] < 210) & (df_f['time_1500'] < 450)]
```

```
df_m = df_m.loc[(df_m['time_800'] < 180) & (df_m['time_1500'] < 360)]

# Remove unrealistic 400 vs 800 times/the new outlier metric
df_f = df_f.loc[(df_f['time_800'] / df_f['time_1500'] > 0.4) & (df_f['time_800'] /
df_m = df_m.loc[(df_m['time_800'] / df_m['time_1500'] > 0.38) & (df_m['time_800'] /
```

In [ ]:
```
# Visualize
plot_bivariate_eda(df_f, title='1500m Women', predictor_event=1500, outcome_event=8
plot_bivariate_eda(df_m, title='1500m Men', predictor_event=1500, outcome_event=800
```





800m vs 400m + 1500m

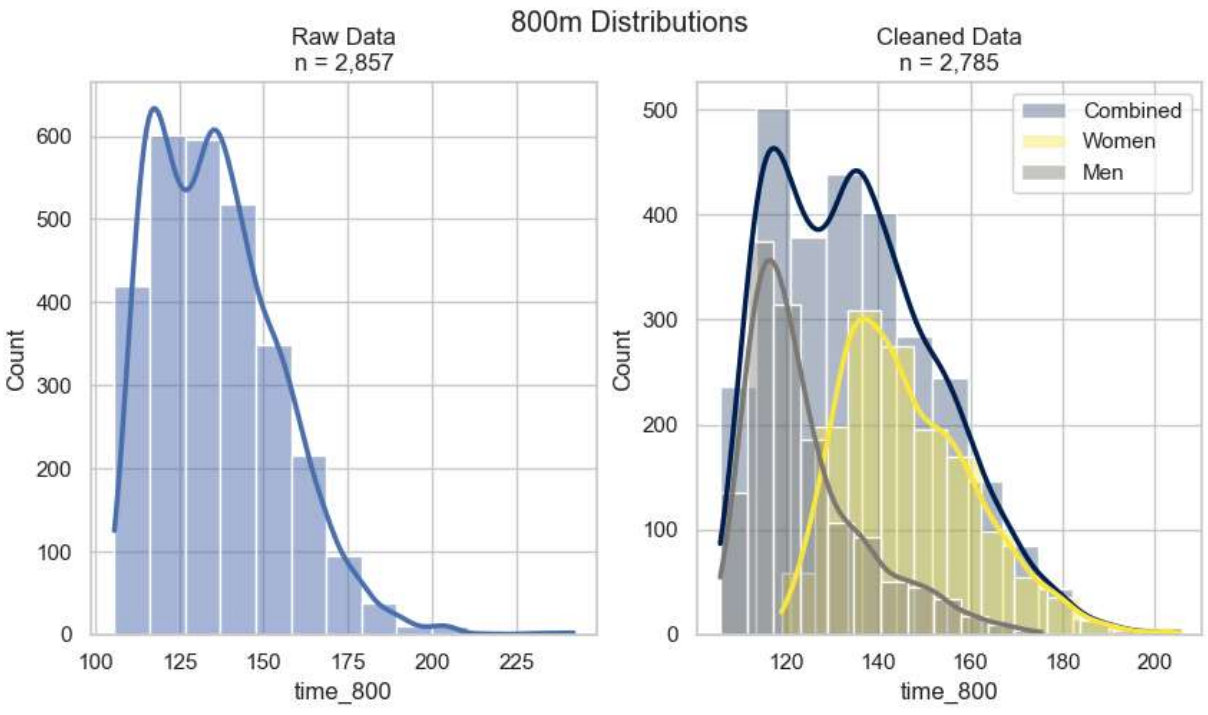
In [ ]:
```
# Rejoin cleaned data
df_cleaned = pd.concat([df_f, df_m]).reset_index(drop=True)
# df_cleaned.to_csv('data/df_cleaned_800_400_1500_2024-04-14.csv', index=False)

df_cleaned
```

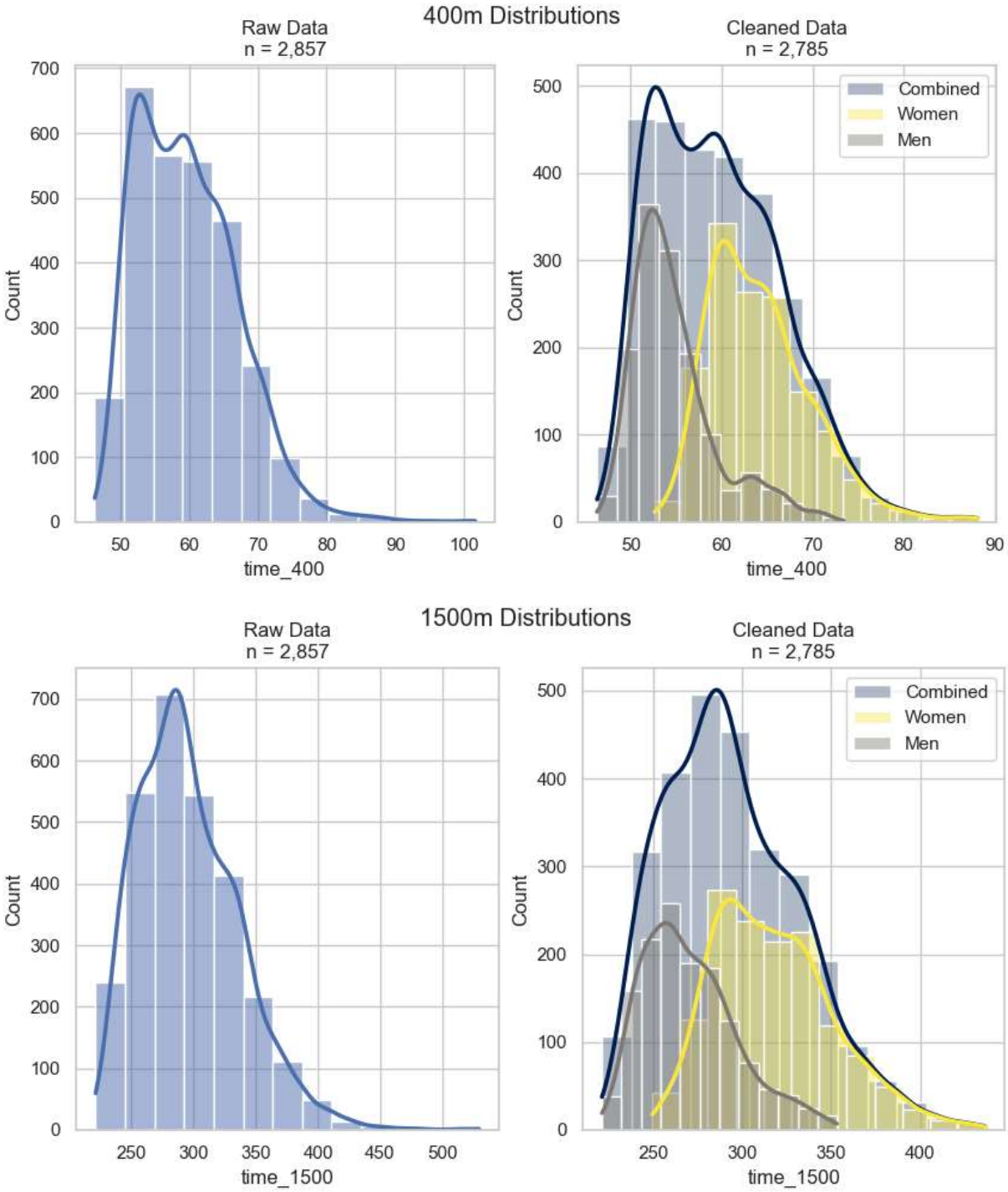
Out[ ]:

	athlete_team	sex	season	time_800	time_400	time_1500
0	Alberti, Abby | Central College	f	indoor_2024	145.95	65.16	328.49
1	Anstice, Sarah | Christopher Newport	f	outdoor_2022	168.55	71.78	348.13
2	Armijo-Vigil, Vivika | Park Gilbert	f	indoor_2022	179.84	75.33	389.81
3	Aspholm, Liv | Rose-Hulman	f	outdoor_2022	155.88	68.40	332.21
4	Bachman, Abby | Franklin & Marshall	f	indoor_2024	146.04	65.10	352.19
...	...	...	...	...	...	...
2780	Ryan WULINSKY NC Marvin Ridge Middle School 2028	m	outdoor_2024	135.10	58.16	277.58
2781	Wynn Andrews AL Pizitz 2028	m	outdoor_2024	135.99	58.23	267.28
2782	Asa King AL Simmons Middle School 2028	m	outdoor_2024	137.57	58.35	283.21
2783	Ethan Soper KS Lakewood Middle School 2028	m	outdoor_2024	138.09	58.58	287.53
2784	Riley Sipe GA Sutton MS 2028	m	outdoor_2024	139.49	55.84	287.62

2785 rows × 6 columns

In [ ]:
```
dist_viz('time_800')
dist_viz('time_400')
dist_viz('time_1500')
```





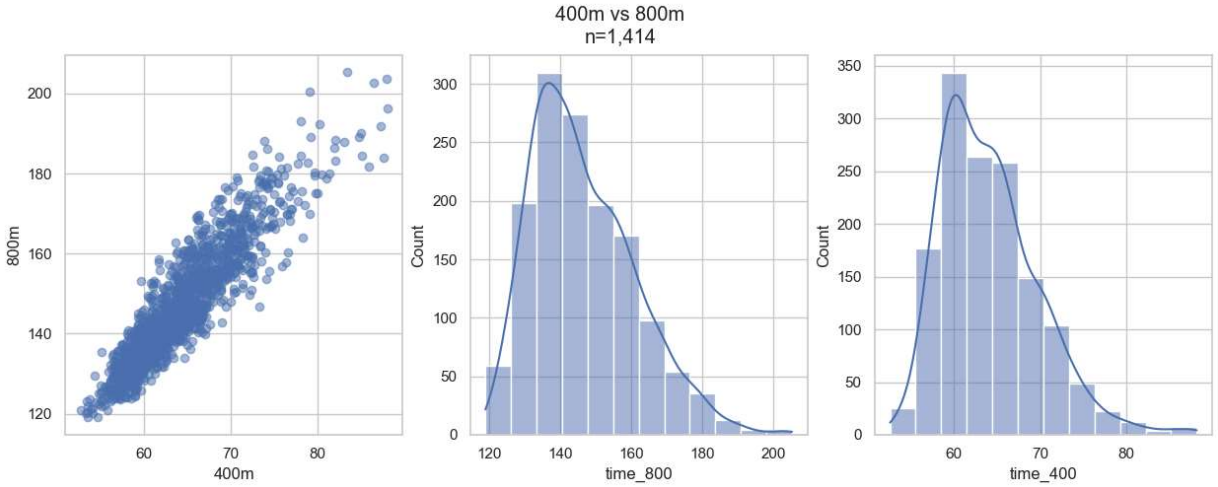
Modelling

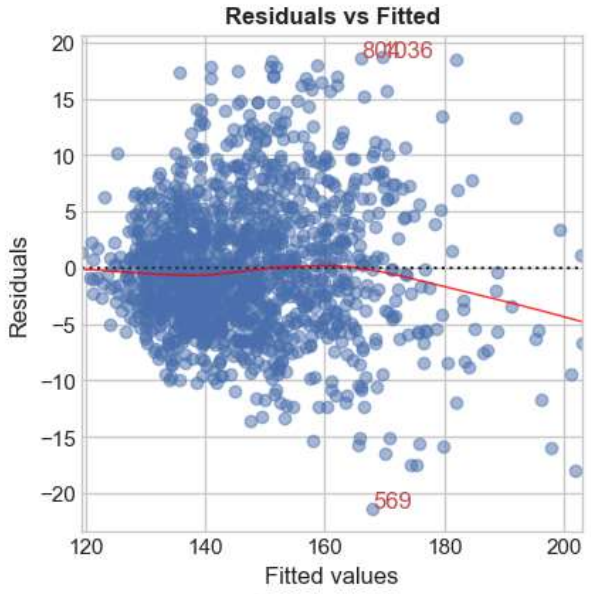
Women's Data

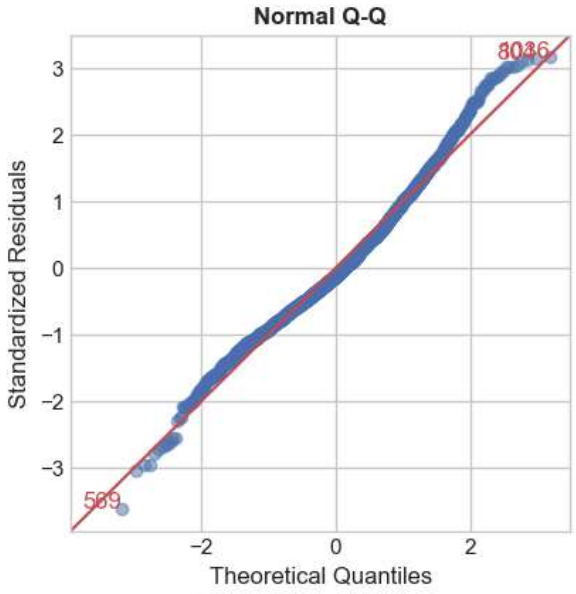
800m vs 400m

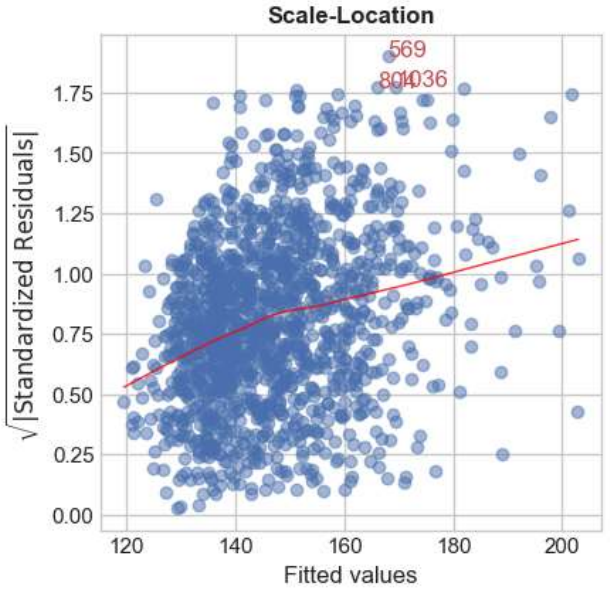
OLS

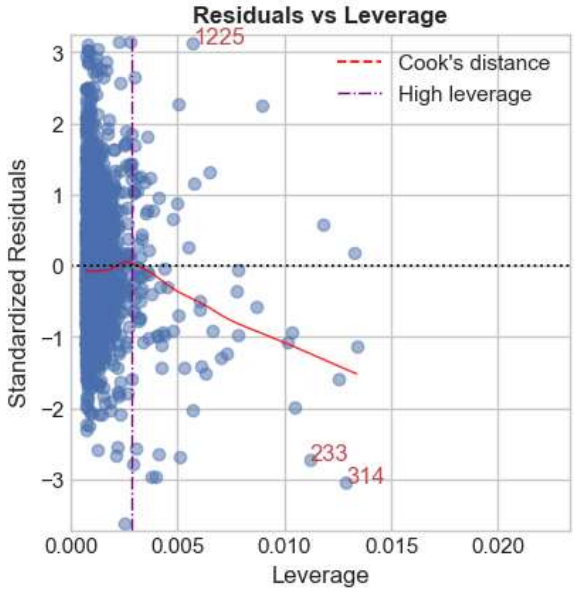
In [ ]:
```
model_400_f = BivariateModel(df_f, outcome_event=800, predictor_event=400, model_ty
model_400_f.plot_dist()
model_400_f.check_assumptions()
model_400_f.plot_conf_int()
```

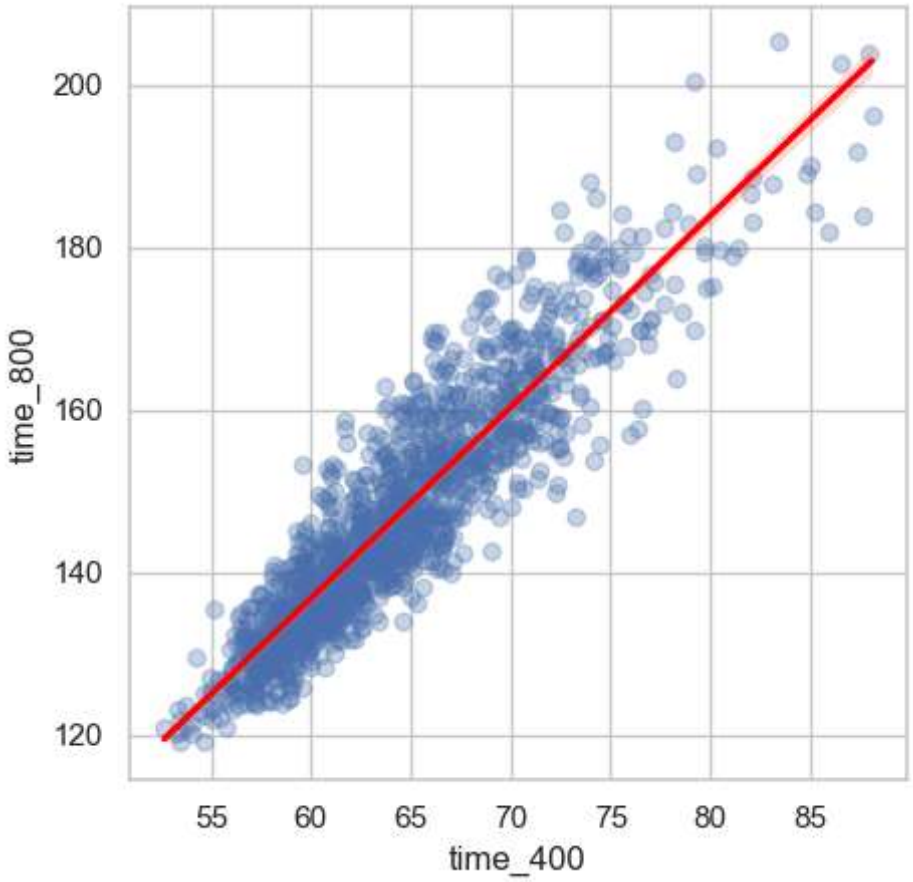












In [ ]:
```
model_400_f.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.835
Model:	OLS	**Adj. R-squared:**	0.835
Method:	Least Squares	**F-statistic:**	7141.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:23:27	**Log-Likelihood:**	-4520.4
No. Observations:	1414	**AIC:**	9045.
Df Residuals:	1412	**BIC:**	9055.
Df Model:	1		
Covariance Type:	nonrobust		

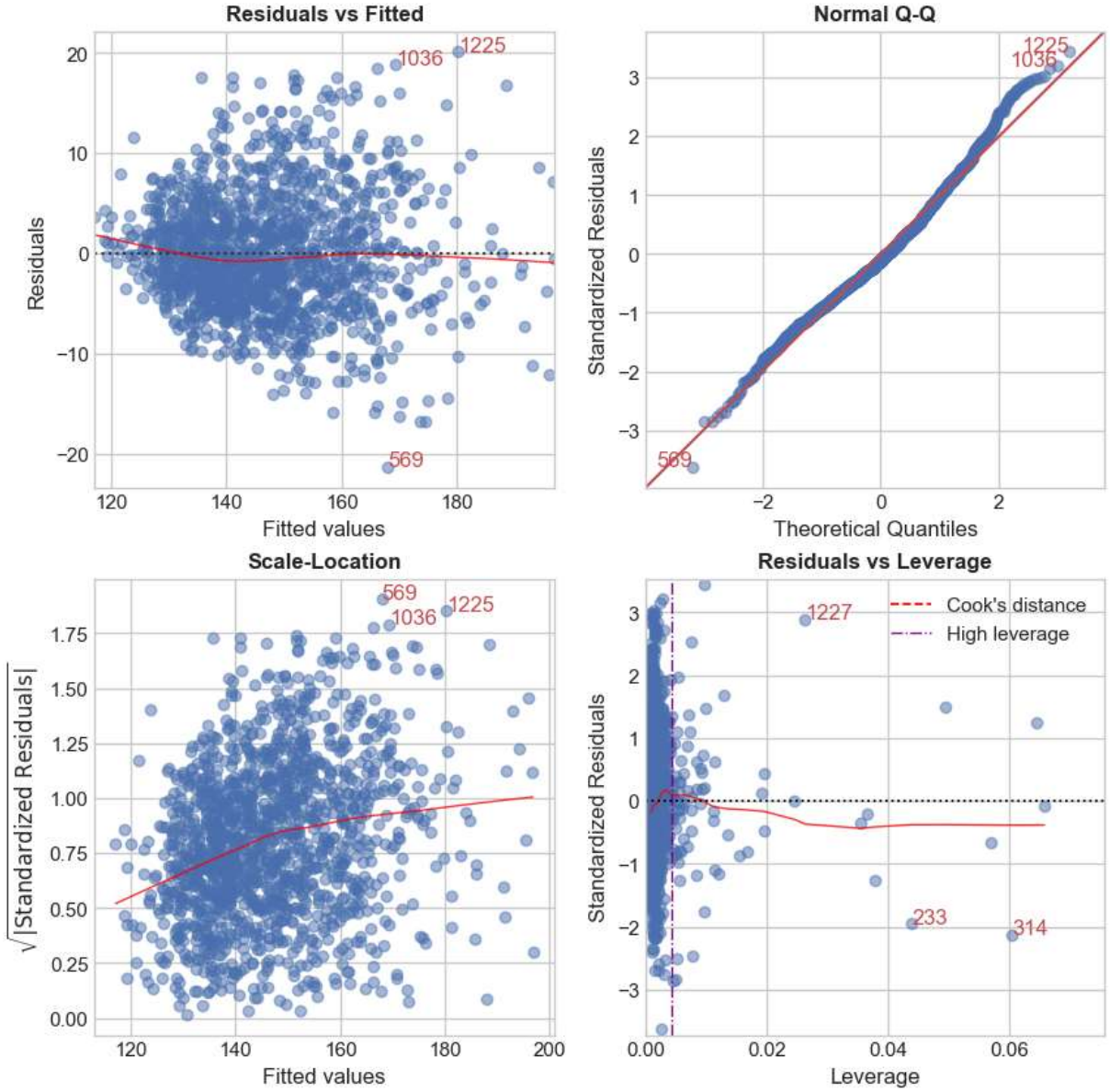
	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.4004	1.794	-2.453	0.014	-7.920	-0.881
time_400	2.3532	0.028	84.502	0.000	2.299	2.408

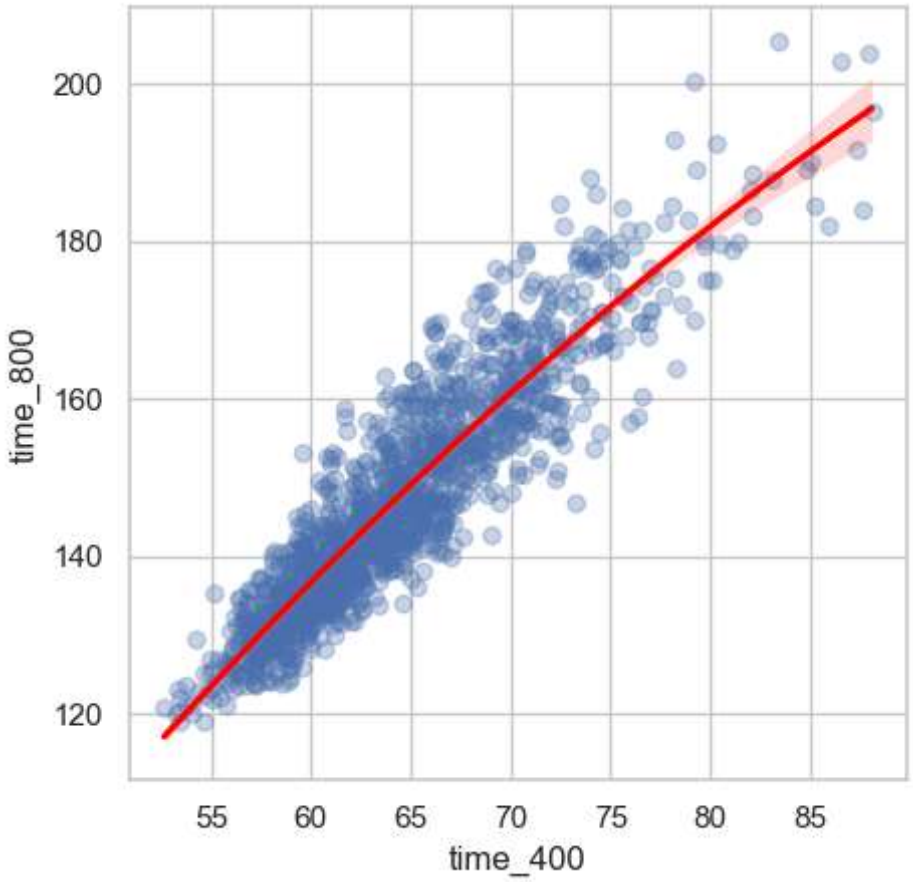
Omnibus:	51.562	**Durbin-Watson:**	1.490
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	60.919
Skew:	0.419	**Prob(JB):**	5.91e-14
Kurtosis:	3.576	**Cond. No.**	734.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Quadratic

In [ ]:
```
model_400_quad_f = BivariateModel(df_f, outcome_event=800, predictor_event=400, mod
model_400_quad_f.check_assumptions()
model_400_quad_f.plot_conf_int()
```


Residuals vs Fitted
Residuals
Fitted values
1036
1225
569
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
1225
1036
569
Scale-Location
√|Standardized Residuals|
Fitted values
569
1036
1225
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1227
233
314



In [ ]:
```
model_400_quad_f.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.837
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	3630.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:23:29	**Log-Likelihood:**	-4510.2
No. Observations:	1414	**AIC:**	9026.
Df Residuals:	1411	**BIC:**	9042.
Df Model:	2		
Covariance Type:	nonrobust		

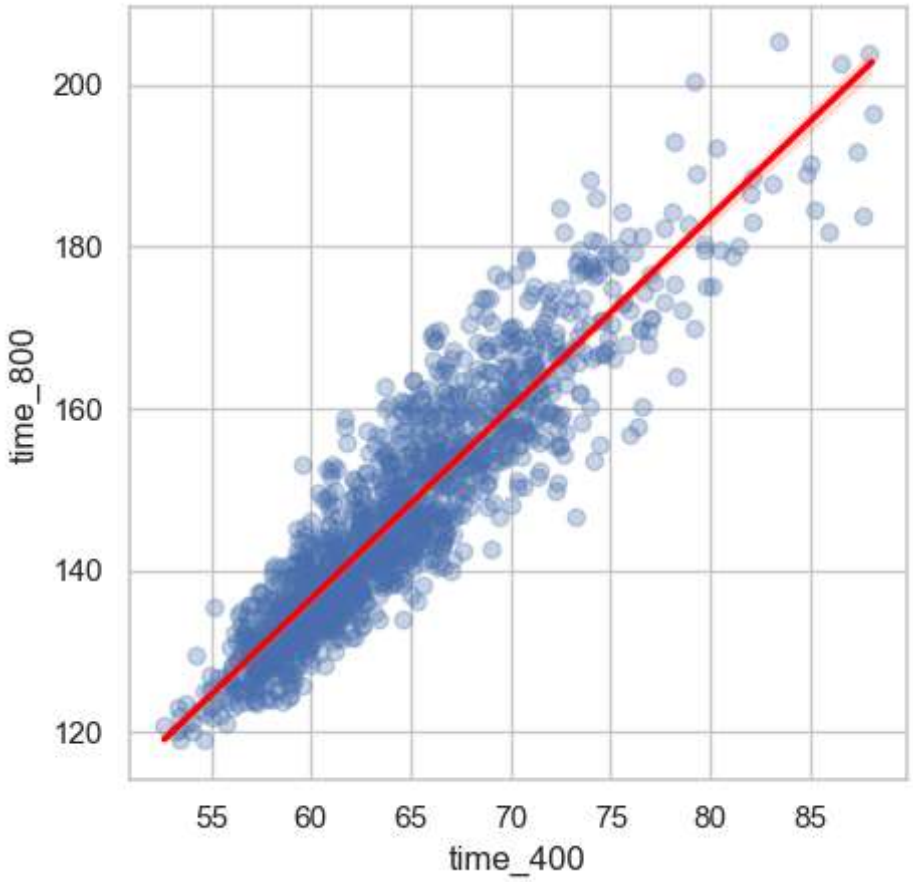
	coef	std err	t	P>|t|	[0.025	0.975]
const	-69.0427	14.362	-4.807	0.000	-97.216	-40.870
x1	4.3068	0.432	9.979	0.000	3.460	5.153
x2	-0.0146	0.003	-4.536	0.000	-0.021	-0.008

Omnibus:	45.793	**Durbin-Watson:**	1.502
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	52.374
Skew:	0.400	**Prob(JB):**	4.24e-12
Kurtosis:	3.498	**Cond. No.**	3.88e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.88e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

In [ ]:
```
model_400_robust_f = BivariateModel(df_f, outcome_event=800, predictor_event=400, m
model_400_robust_f.check_assumptions()
model_400_robust_f.plot_conf_int()
```



In [ ]:
```
model_400_robust_f.model_summary
```

Out[ ]:

Robust linear Model Regression Results

Dep. Variable:	time_800	**No. Observations:**	1414
Model:	RLM	**Df Residuals:**	1412
Method:	IRLS	**Df Model:**	1
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Sun, 14 Apr 2024		
Time:	16:23:40		
No. Iterations:	14		

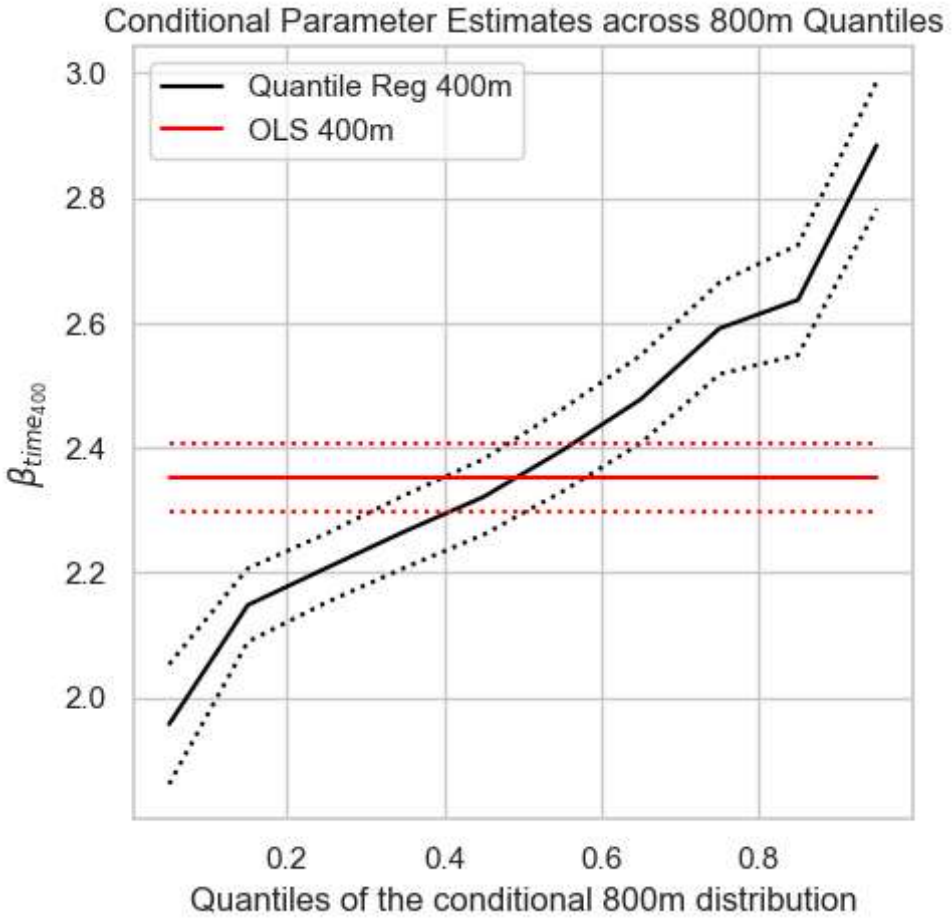
	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.9751	1.720	-2.893	0.004	-8.346	-1.604
time_400	2.3579	0.027	88.324	0.000	2.306	2.410

If the model instance has been used for another fit with different fit parameters, then the fit options might not be the correct ones anymore .

Quantile Regression

In [ ]:
```
model_400_quant_f = BivariateModel(df_f, outcome_event=800, predictor_event=400, mo
model_400_quant_f.plot_quantiles_by_parameter()
```

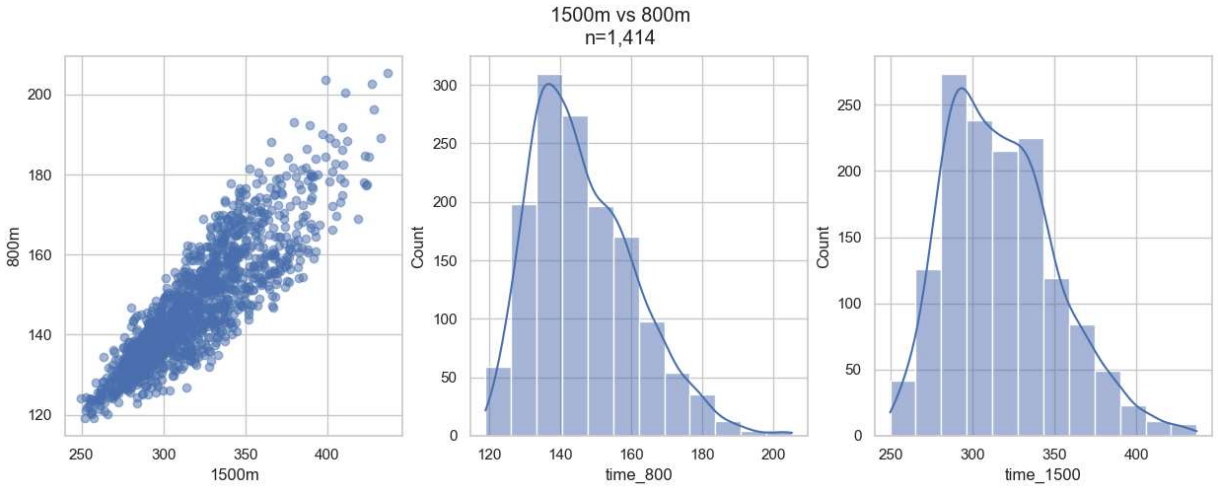
```
c:\Users\mitch\miniconda3\envs\webpy\Lib\site-packages\statsmodels\regression\quanti
le_regression.py:191: IterationLimitWarning: Maximum number of iterations (1000) rea
ched.
  warnings.warn("Maximum number of iterations (" + str(max_iter) +
c:\Users\mitch\miniconda3\envs\webpy\Lib\site-packages\statsmodels\regression\quanti
le_regression.py:191: IterationLimitWarning: Maximum number of iterations (1000) rea
ched.
  warnings.warn("Maximum number of iterations (" + str(max_iter) +
```

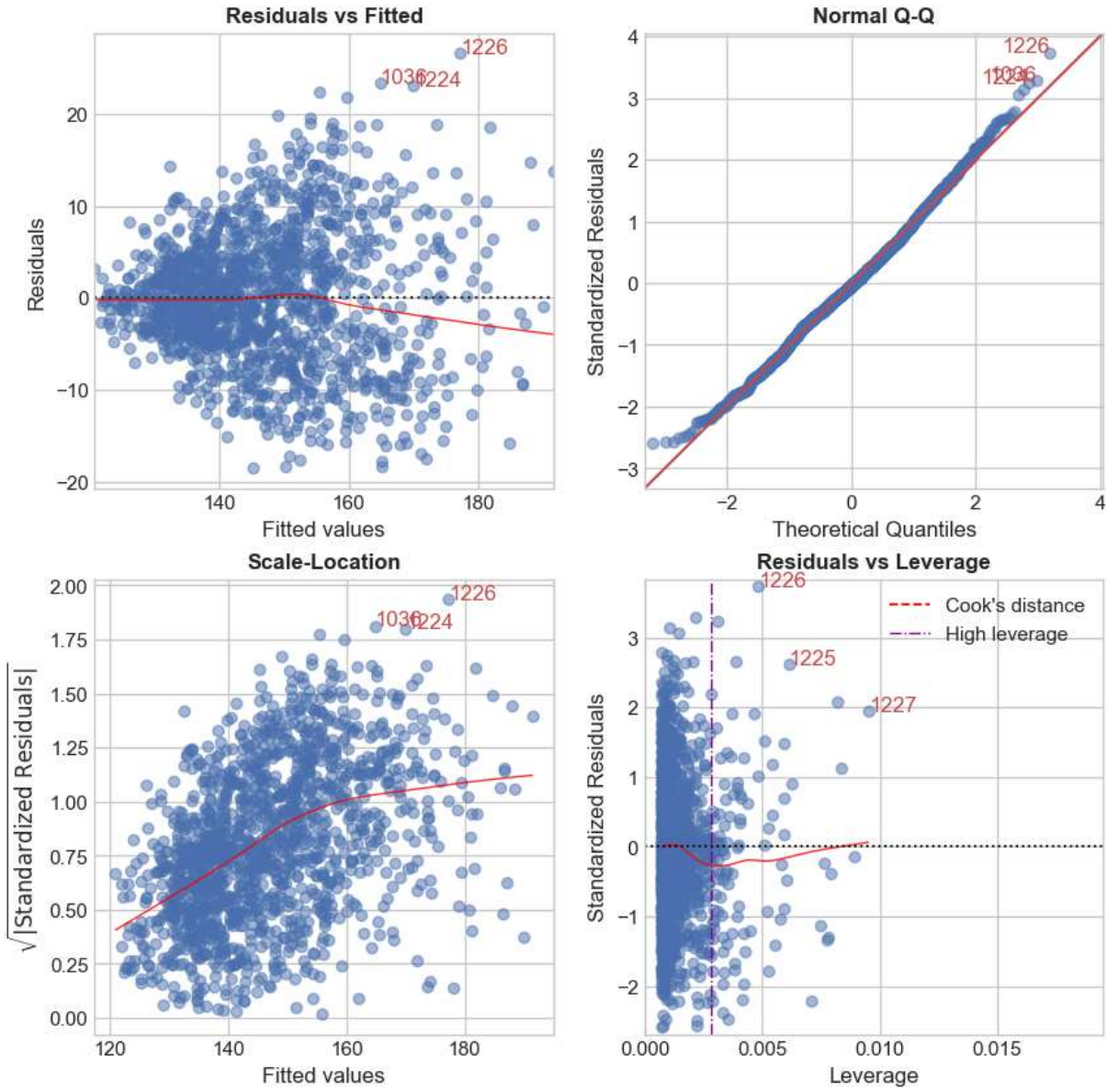


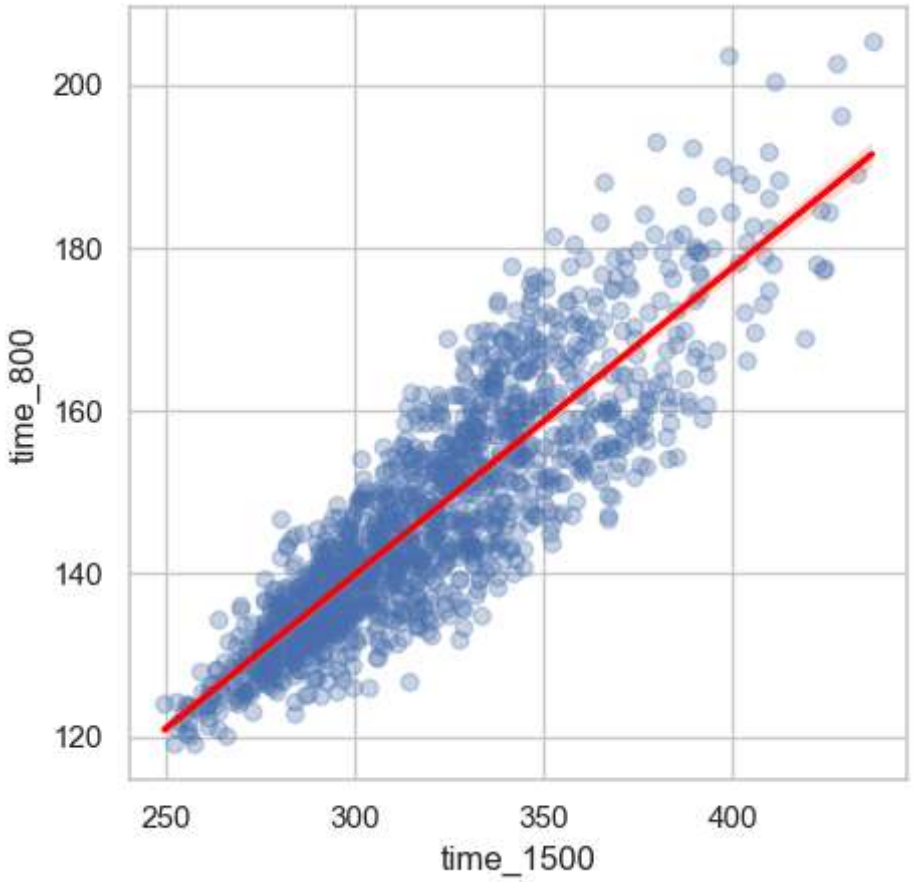
800m vs 1500m

OLS

In [ ]:
```
model_1500_f = BivariateModel(df_f, outcome_event=800, predictor_event=1500, model_
model_1500_f.plot_dist()
model_1500_f.check_assumptions()
model_1500_f.plot_conf_int()
```




Residuals vs Fitted
Residuals
Fitted values
1226
1036
1224
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
Scale-Location
√|Standardized Residuals|
Fitted values
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1225
1227



In [ ]:
```
model_1500_f.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.761
Model:	OLS	**Adj. R-squared:**	0.761
Method:	Least Squares	**F-statistic:**	4506.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:23:45	**Log-Likelihood:**	-4780.8
No. Observations:	1414	**AIC:**	9566.
Df Residuals:	1412	**BIC:**	9576.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	26.7135	1.796	14.873	0.000	23.190	30.237
time_1500	0.3773	0.006	67.127	0.000	0.366	0.388

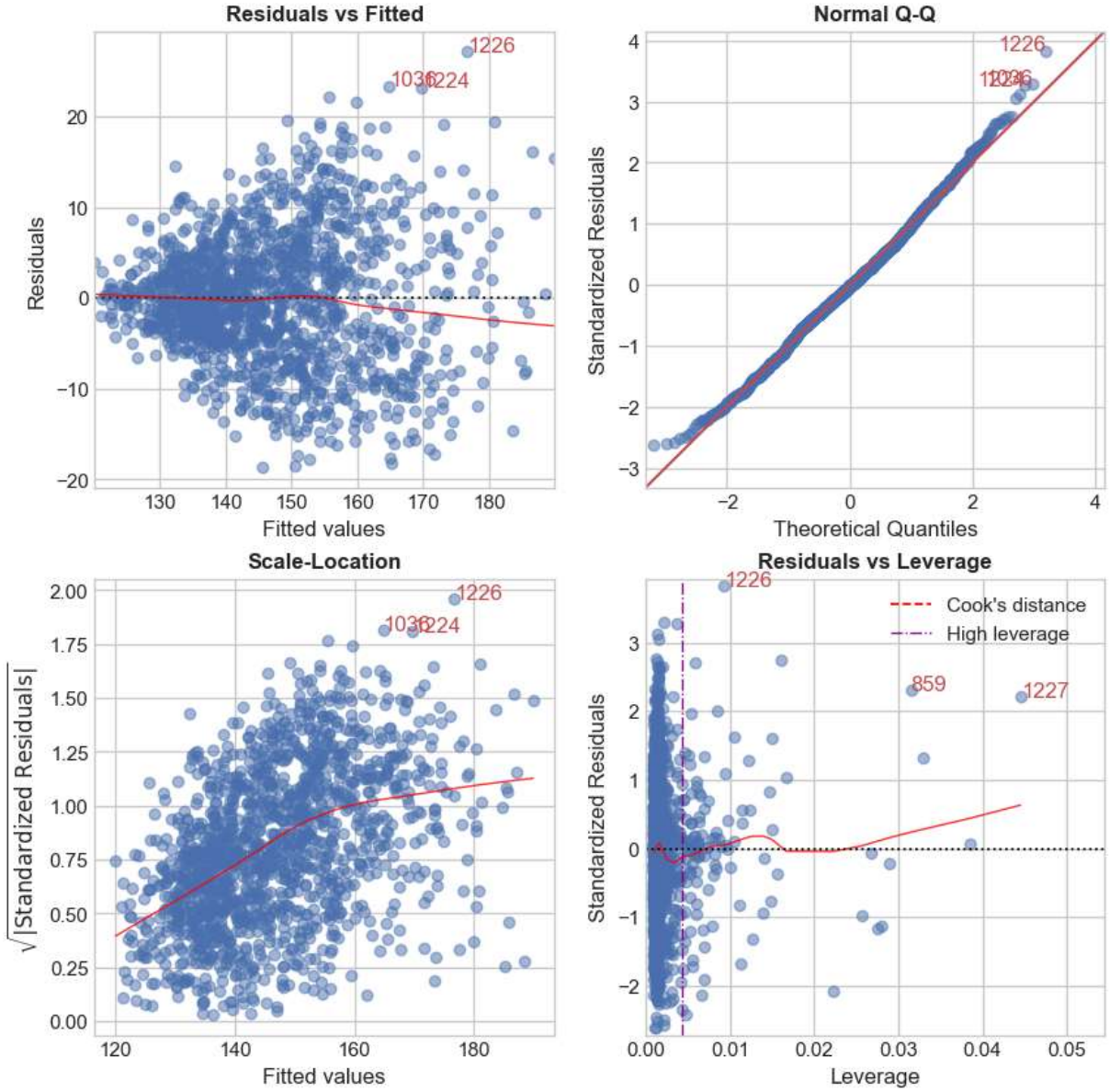
Omnibus:	11.149	**Durbin-Watson:**	1.343
Prob(Omnibus):	0.004	**Jarque-Bera (JB):**	11.164
Skew:	0.208	**Prob(JB):**	0.00376
Kurtosis:	3.130	**Cond. No.**	3.03e+03

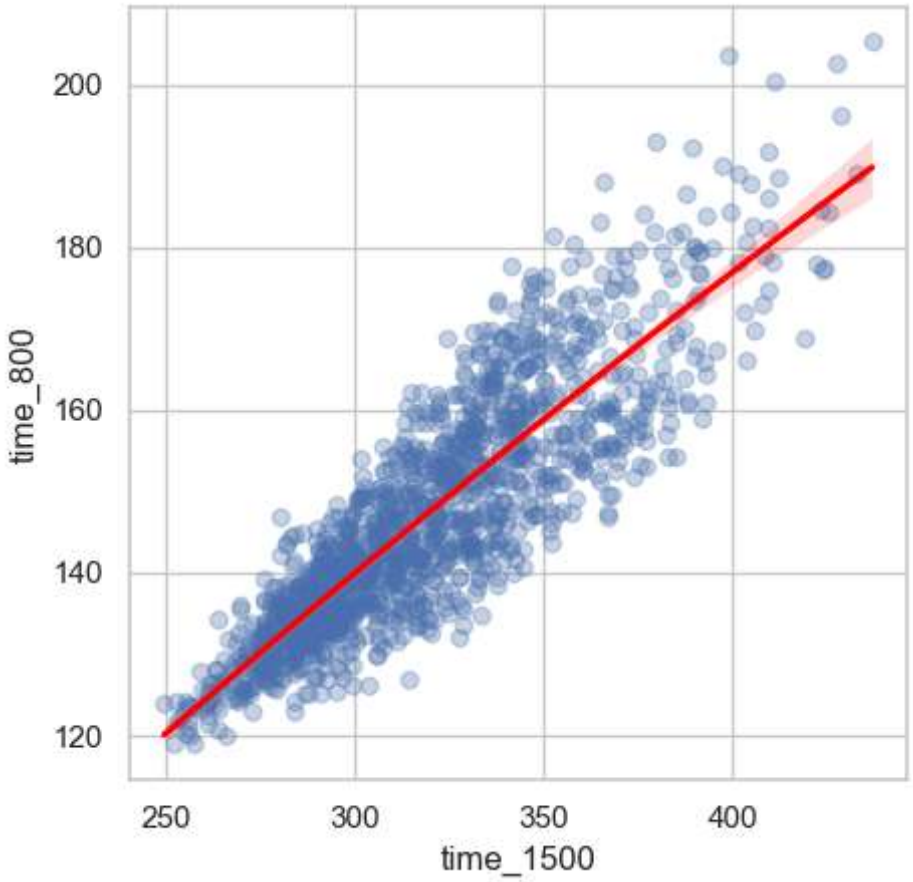
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.03e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Quadratic

In [ ]:

```
model_1500_quad_f = BivariateModel(df_f, outcome_event=800, predictor_event=1500, m
model_1500_quad_f.check_assumptions()
model_1500_quad_f.plot_conf_int()
```


Residuals vs Fitted
Residuals
Fitted values
1226
1036
1224
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
1226
Scale-Location
√|Standardized Residuals|
Fitted values
1226
1036
1224
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1226
859
1227



In [ ]:
```
model_1500_quad_f.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.762
Model:	OLS	**Adj. R-squared:**	0.761
Method:	Least Squares	**F-statistic:**	2255.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:23:48	**Log-Likelihood:**	-4780.0
No. Observations:	1414	**AIC:**	9566.
Df Residuals:	1411	**BIC:**	9582.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	10.0541	13.597	0.739	0.460	-16.618	36.726
x1	0.4800	0.083	5.765	0.000	0.317	0.643
x2	-0.0002	0.000	-1.236	0.217	-0.000	9.18e-05

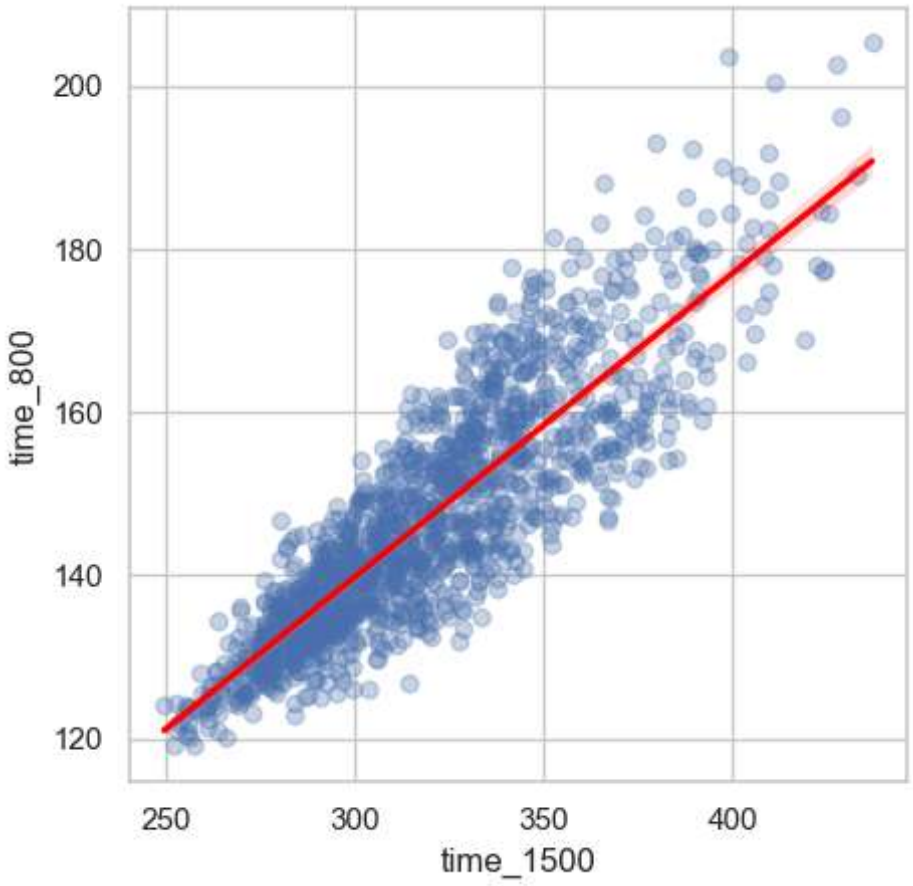
Omnibus:	10.328	**Durbin-Watson:**	1.351
Prob(Omnibus):	0.006	**Jarque-Bera (JB):**	10.332
Skew:	0.197	**Prob(JB):**	0.00571
Kurtosis:	3.144	**Cond. No.**	7.51e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.51e+06. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

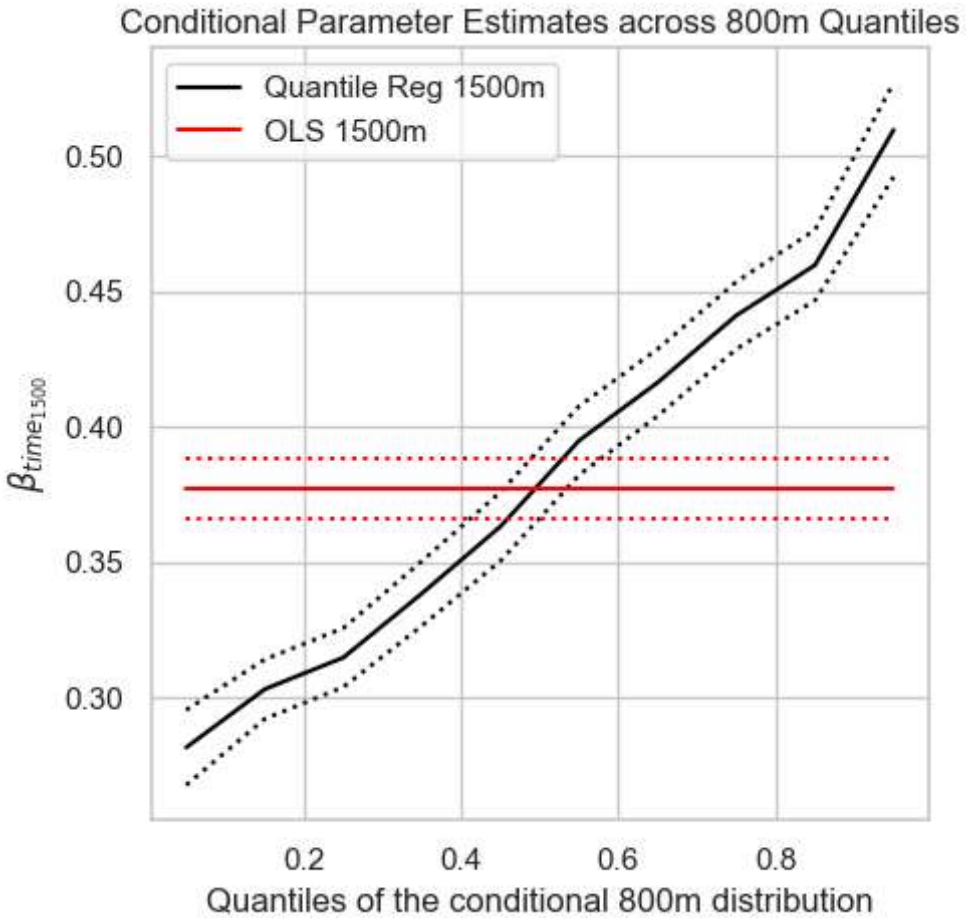
In [ ]:

```
model_1500_robust_f = BivariateModel(df_f, outcome_event=800, predictor_event=1500,
model_1500_robust_f.check_assumptions()
model_1500_robust_f.plot_conf_int()
```



Quantile Regression

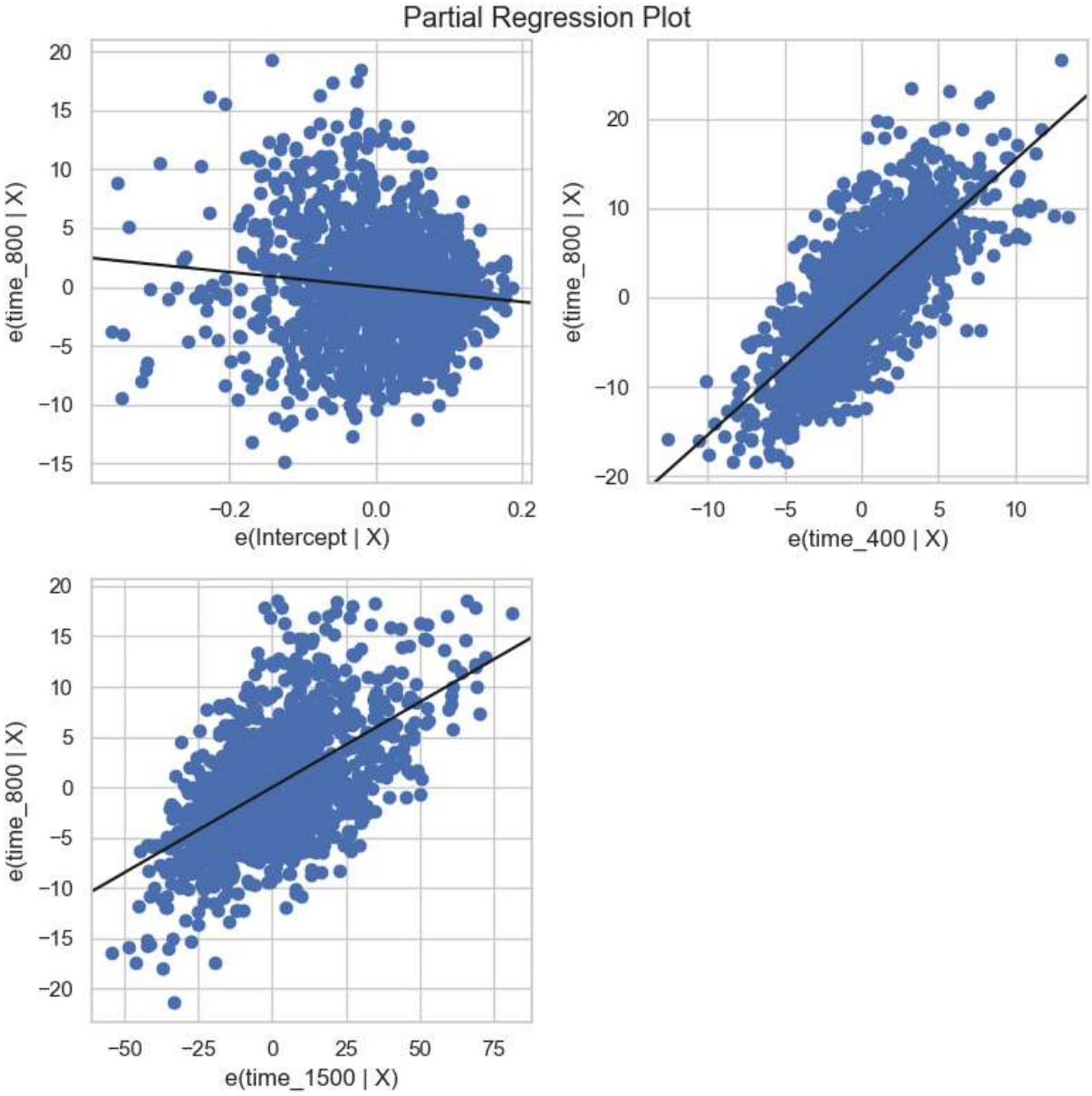
In [ ]:
```
model_1500_quant_f = BivariateModel(df_f, outcome_event=800, predictor_event=1500,
model_1500_quant_f.plot_quantiles_by_parameter()
```

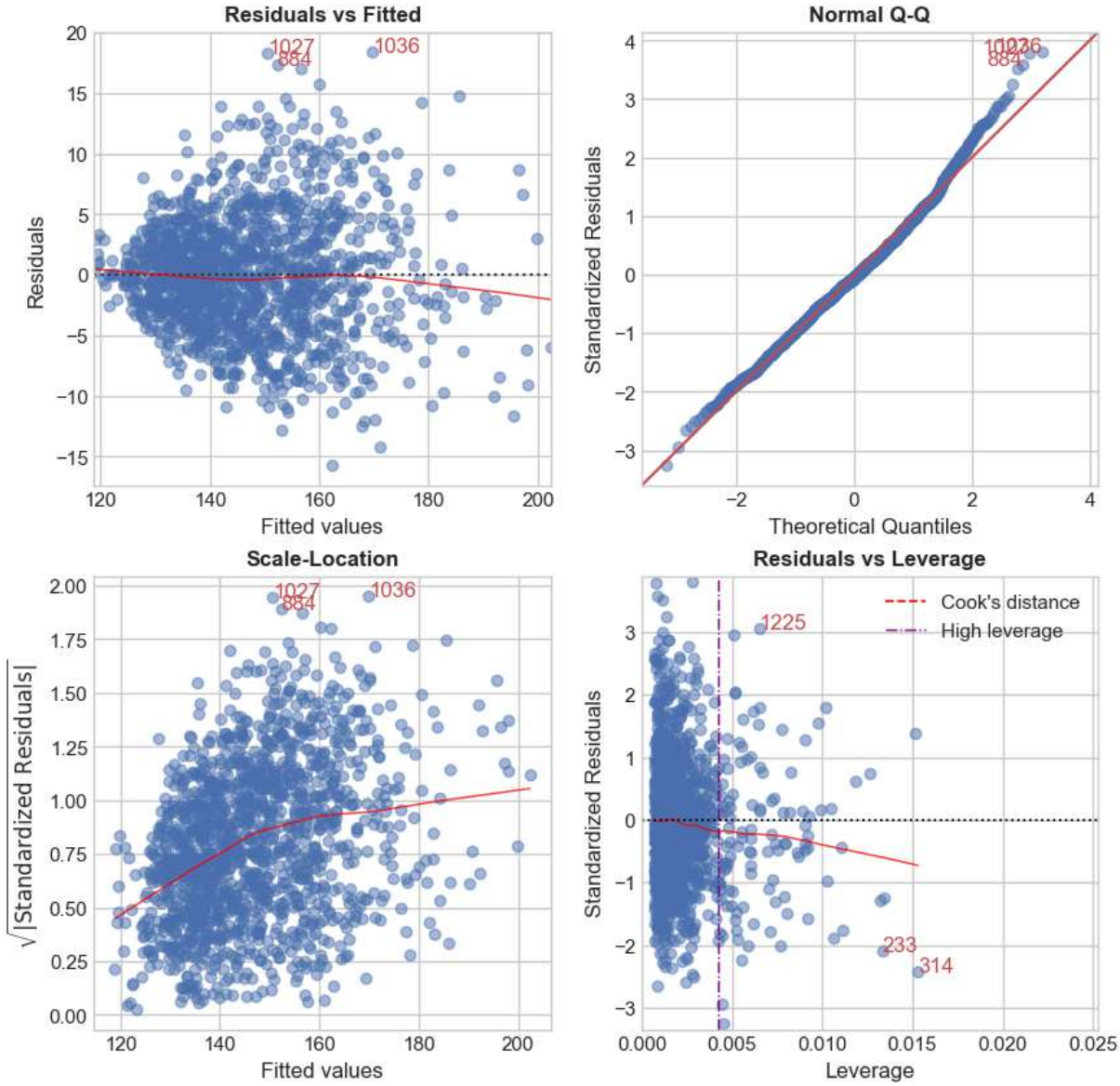


800m vs 400m + 1500m

OLS

In [ ]:
```
model_400_1500_f = MultivariateModel(df_f, outcome_event=800, predictor_events=[400
model_400_1500_f()
```


Partial Regression Plot
e(time_800 | X)
e(Intercept | X)
e(time_400 | X)
e(time_1500 | X)


Residuals vs Fitted
Residuals
Fitted values
1027
884
1036
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
Scale-Location
√|Standardized Residuals|
Fitted values
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1225
233
314

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.890
Model:	OLS	**Adj. R-squared:**	0.890
Method:	Least Squares	**F-statistic:**	5696.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:06	**Log-Likelihood:**	-4234.7
No. Observations:	1414	**AIC:**	8475.
Df Residuals:	1411	**BIC:**	8491.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-6.3241	1.468	-4.308	0.000	-9.204	-3.444
time_400	1.5439	0.038	40.545	0.000	1.469	1.619
time_1500	0.1695	0.006	26.511	0.000	0.157	0.182

Omnibus:	39.266	**Durbin-Watson:**	1.236
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	46.275
Skew:	0.348	**Prob(JB):**	8.94e-11
Kurtosis:	3.549	**Cond. No.**	3.72e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.72e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

In [ ]:
```
model_400_1500_robust_f = MultivariateModel(df_f, outcome_event=800, predictor_even
model_400_1500_robust_f()
```

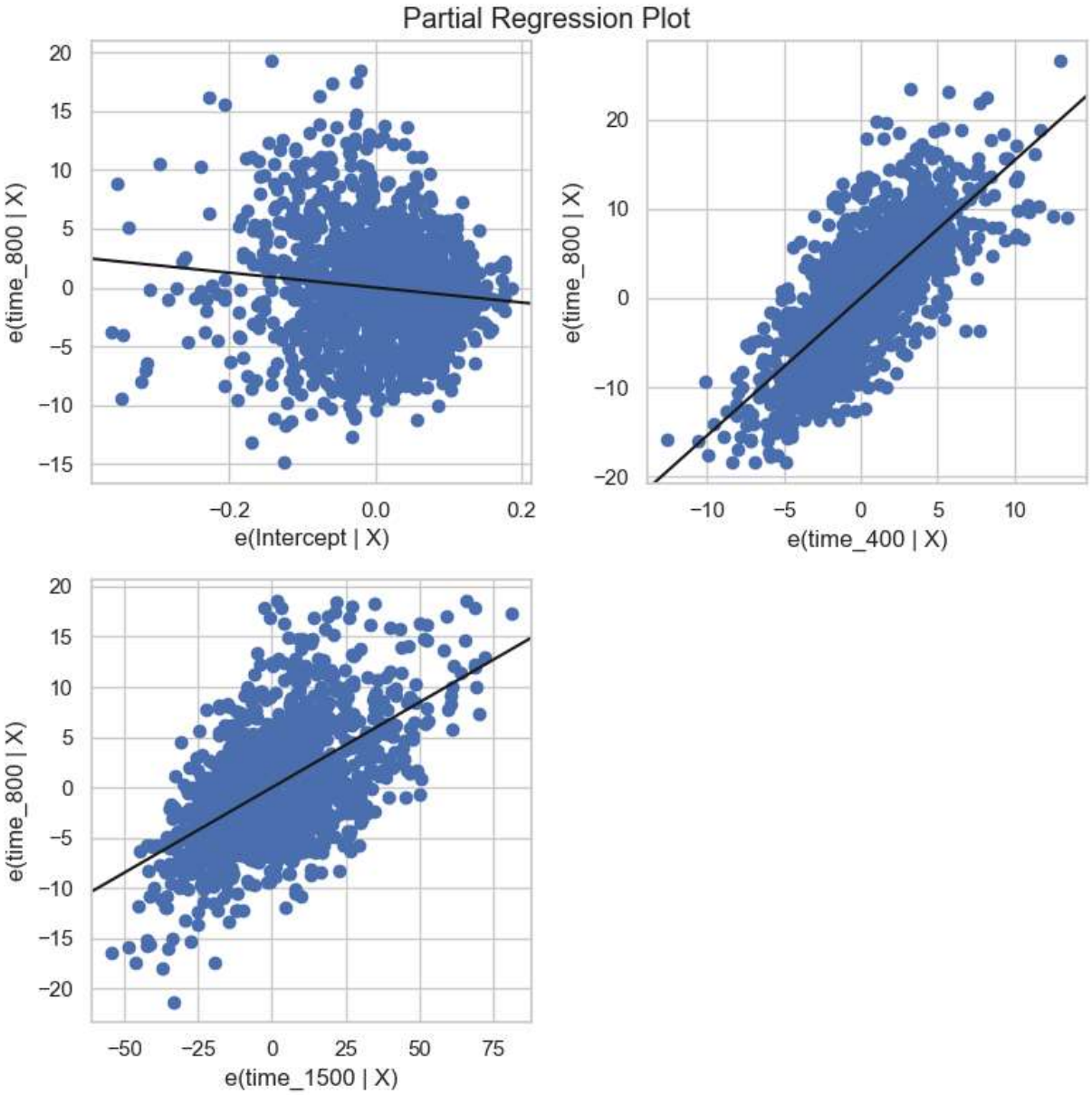
Out[ ]:

Robust linear Model Regression Results

Dep. Variable:	time_800	**No. Observations:**	1414
Model:	RLM	**Df Residuals:**	1411
Method:	IRLS	**Df Model:**	2
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Sun, 14 Apr 2024		
Time:	16:24:07		
No. Iterations:	19		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-5.6640	1.434	-3.949	0.000	-8.475	-2.853
time_400	1.5515	0.037	41.710	0.000	1.479	1.624
time_1500	0.1654	0.006	26.477	0.000	0.153	0.178

If the model instance has been used for another fit with different fit parameters, then the fit options might not be the correct ones anymore .

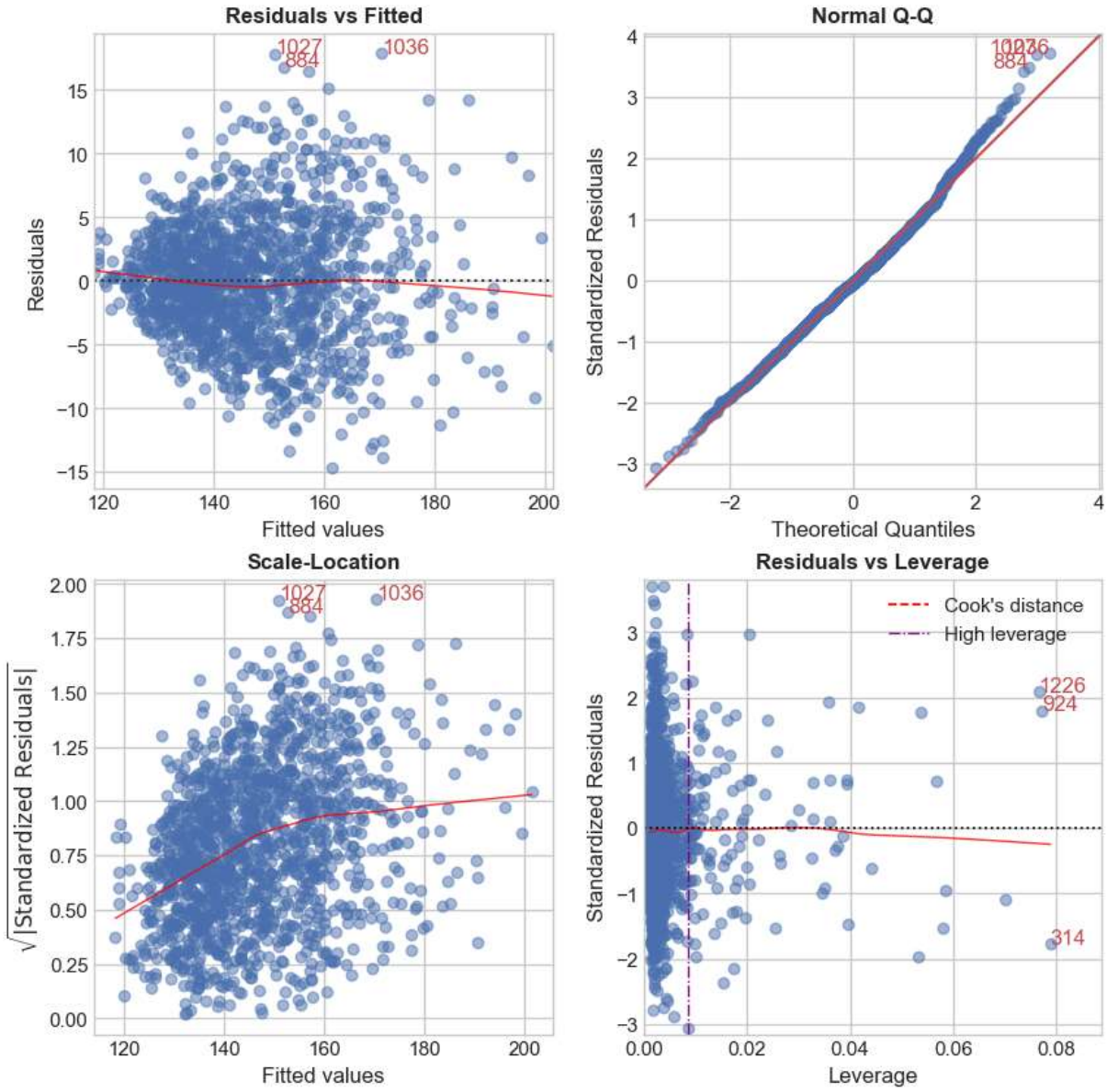


In [ ]:
```
model_400_1500_robust_f.predict_time(times = ['0:53.80', '4:18.40'], events=[400, 1
```

Out[ ]:
```
'800m Prediction: 117.87 seconds'
```

Quadratic

In [ ]:
```
model_400_1500_quad_f = MultivariateModel(df_f, outcome_event=800, predictor_events
model_400_1500_quad_f()
```


Residuals vs Fitted
Residuals
Fitted values
1027
884
1036
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
Scale-Location
√|Standardized Residuals|
Fitted values
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1226
924
314

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.891
Model:	OLS	**Adj. R-squared:**	0.891
Method:	Least Squares	**F-statistic:**	2305.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:10	**Log-Likelihood:**	-4226.1
No. Observations:	1414	**AIC:**	8464.
Df Residuals:	1408	**BIC:**	8496.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-23.2639	12.150	-1.915	0.056	-47.098	0.570
x1	2.5003	0.467	5.356	0.000	1.585	3.416
x2	0.0767	0.076	1.005	0.315	-0.073	0.226
x3	-0.0287	0.007	-3.952	0.000	-0.043	-0.014
x4	0.0086	0.002	3.753	0.000	0.004	0.013
x5	-0.0007	0.000	-3.274	0.001	-0.001	-0.000

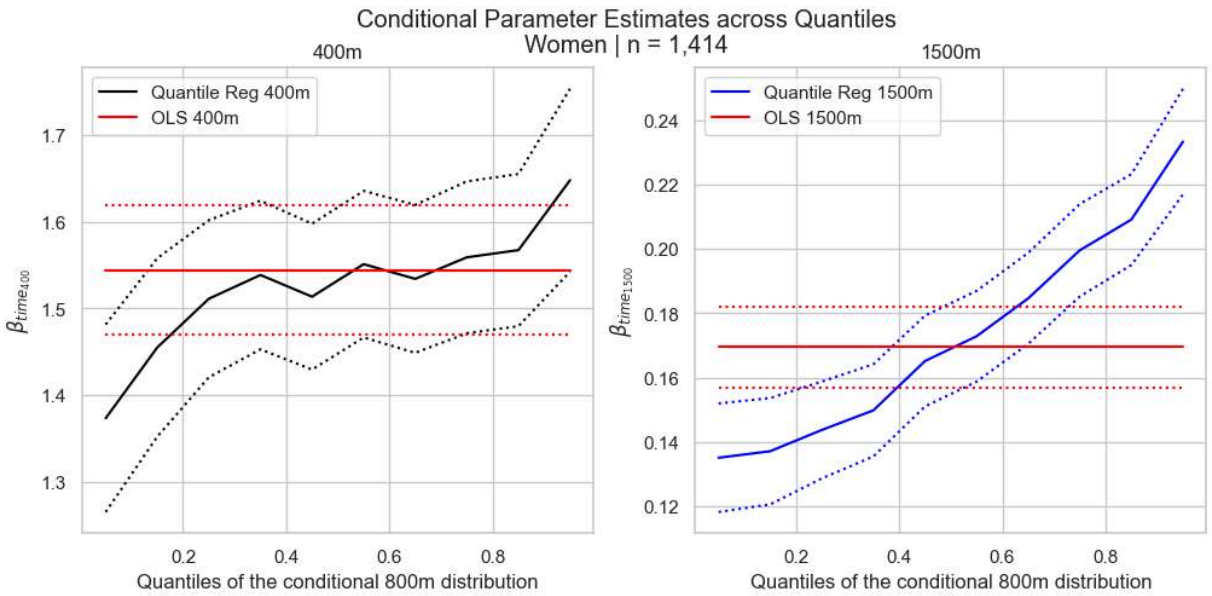
Omnibus:	28.726	**Durbin-Watson:**	1.261
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	32.639
Skew:	0.294	**Prob(JB):**	8.18e-08
Kurtosis:	3.457	**Cond. No.**	1.01e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.01e+07. This might indicate that there are strong multicollinearity or other numerical problems.

Quantile Regression

Cleaned Data

In [ ]:
```
model_400_1500_quant_f = MultivariateModel(df_f, outcome_event=800, predictor_event
model_400_1500_quant_f.plot_quantiles_by_parameter(title_specifier='Women')
```



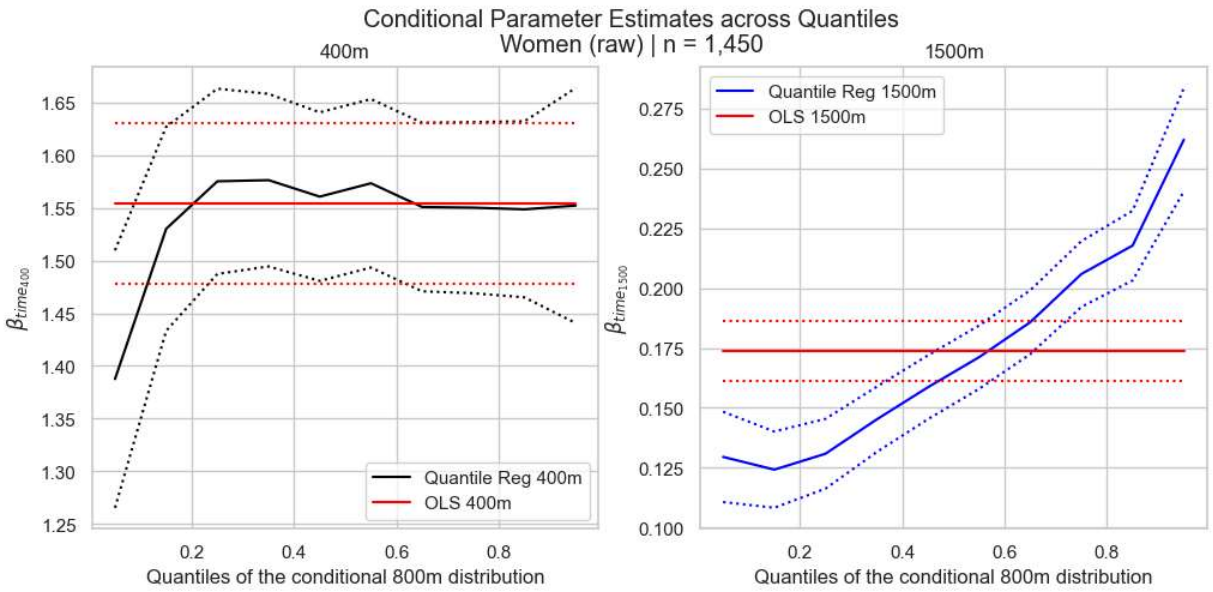
In [ ]:
```
make_quantile_table(df_f, 'time_800')
```

Out[ ]:

	quantile	time_800m
0	0.05	127.01
1	0.20	134.03
2	0.40	140.54
3	0.60	148.04
4	0.80	158.99
5	0.95	173.70

Raw Data

In [ ]:
```
model_raw_400_1500_f = MultivariateModel(df_raw.loc[df_raw['sex']=='f'], outcome_ev
model_raw_400_1500_f.plot_quantiles_by_parameter(title_specifier='Women (raw)')
```



In [ ]:
```
make_quantile_table(df_raw.loc[df_raw['sex']=='f'], 'time_800')
```

Out[ ]:

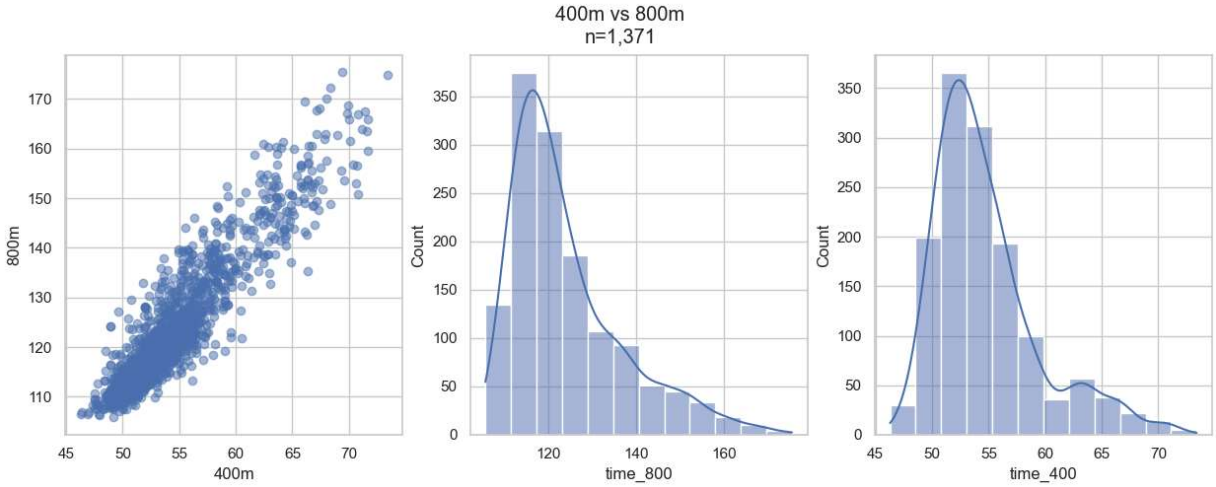
	quantile	time_800m
0	0.05	127.05
1	0.20	134.12
2	0.40	140.72
3	0.60	148.52
4	0.80	159.54
5	0.95	176.61

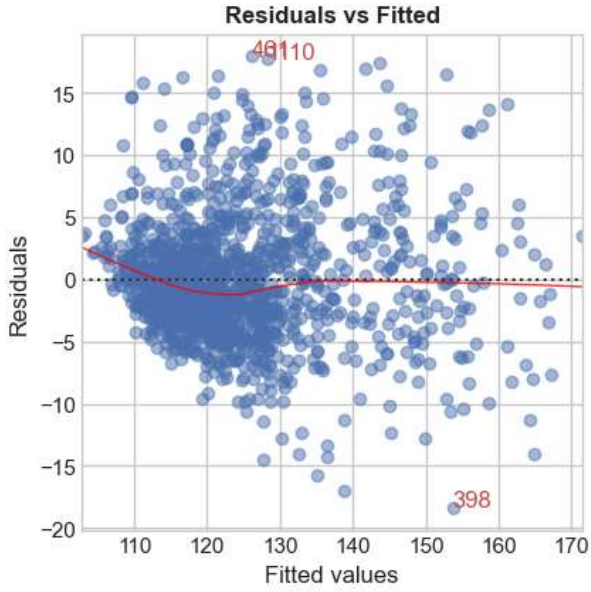
Men's Data

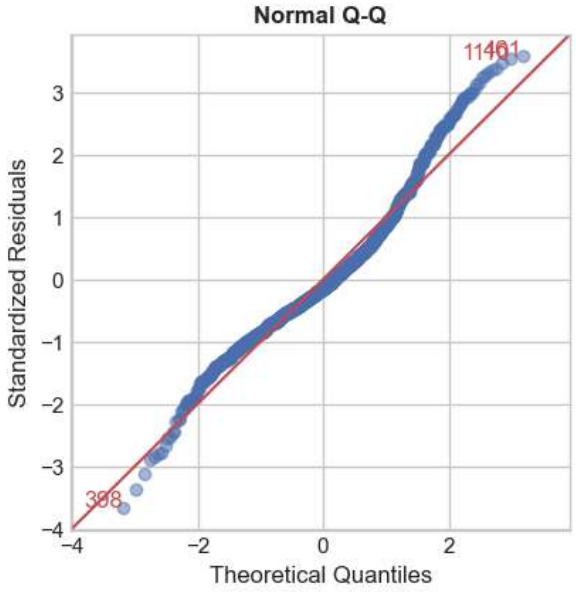
800m vs 400m

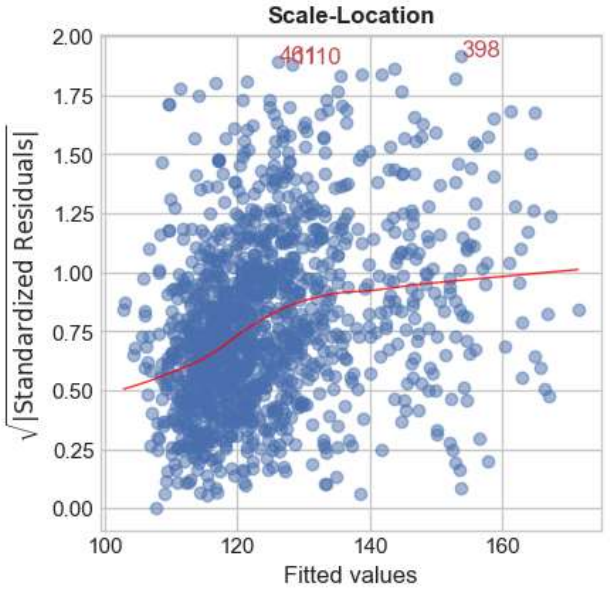
OLS

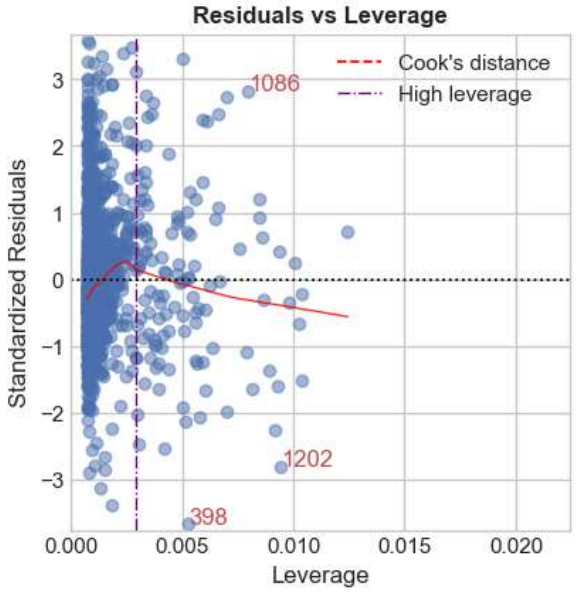
In [ ]:
```
model_400_m = BivariateModel(df_m, outcome_event=800, predictor_event=400, model_ty
model_400_m.plot_dist()
model_400_m.check_assumptions()
model_400_m.plot_conf_int()
```

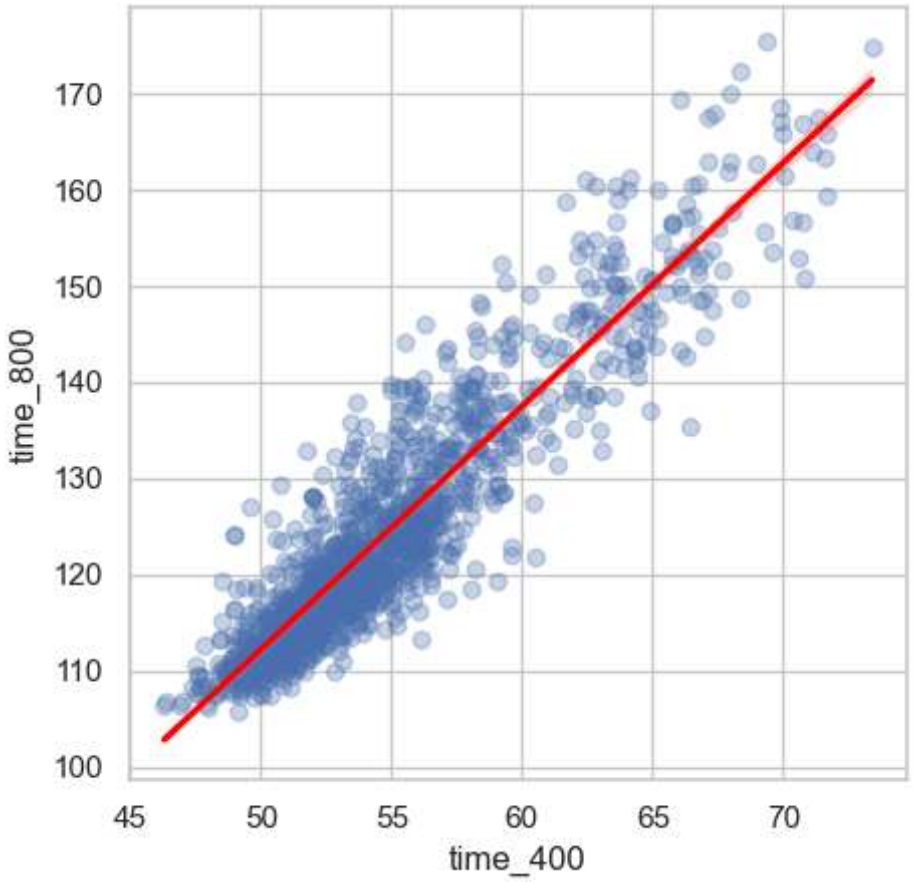

400m vs 800m
n=1,371
800m
400m
Count
time_800
Count
time_400


Residuals vs Fitted
Residuals
Fitted values
398


Normal Q-Q
Standardized Residuals
Theoretical Quantiles
398


Scale-Location
√|Standardized Residuals|
Fitted values
398


Residuals vs Leverage
Cook's distance
High leverage
Standardized Residuals
Leverage
1086
1202
398



In [ ]:
```
model_400_m.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.846
Model:	OLS	**Adj. R-squared:**	0.846
Method:	Least Squares	**F-statistic:**	7541.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:16	**Log-Likelihood:**	-4156.1
No. Observations:	1371	**AIC:**	8316.
Df Residuals:	1369	**BIC:**	8327.
Df Model:	1		
Covariance Type:	nonrobust		

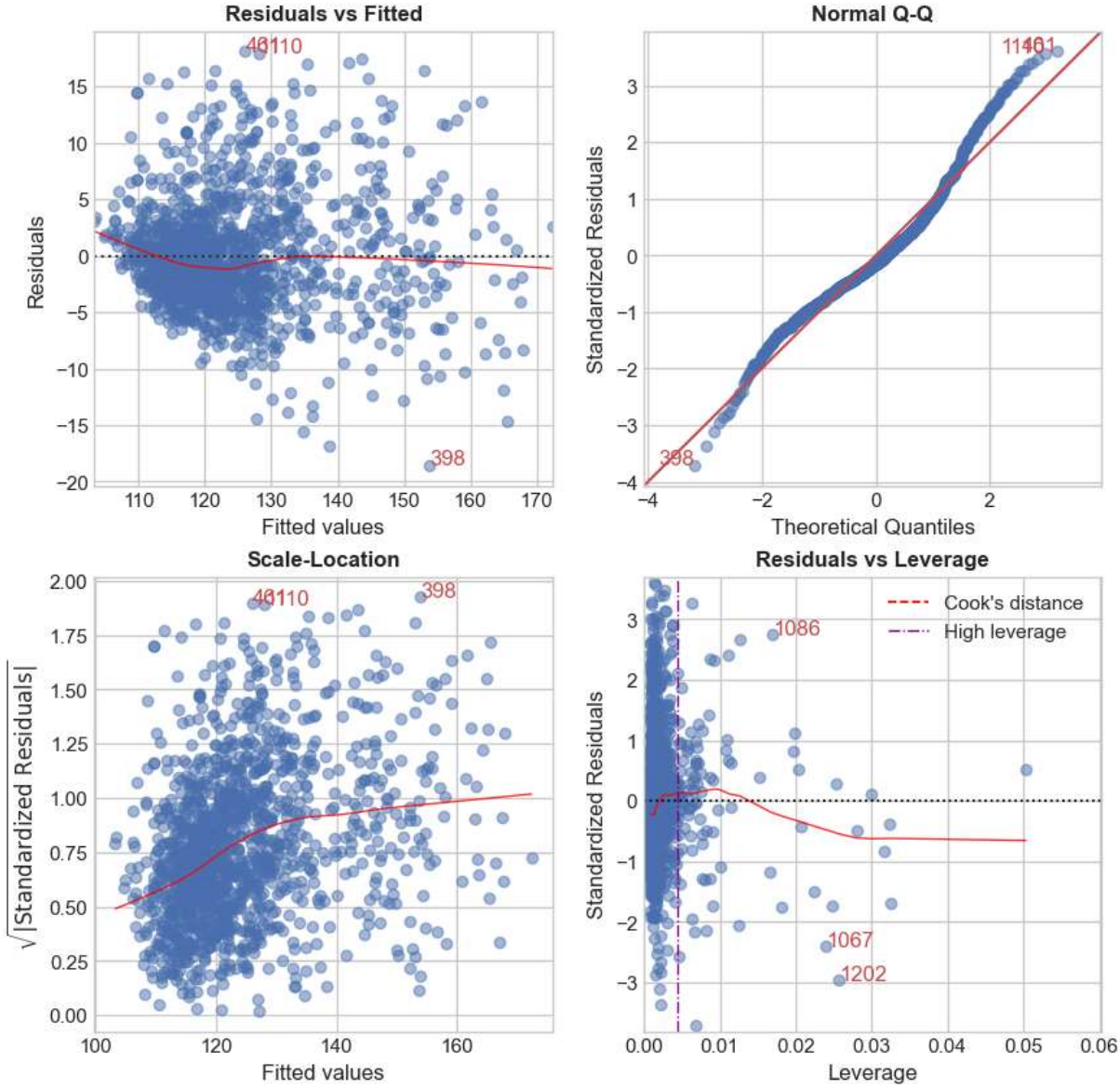
	coef	std err	t	P>|t|	[0.025	0.975]
const	-14.2515	1.600	-8.906	0.000	-17.391	-11.112
time_400	2.5296	0.029	86.840	0.000	2.472	2.587

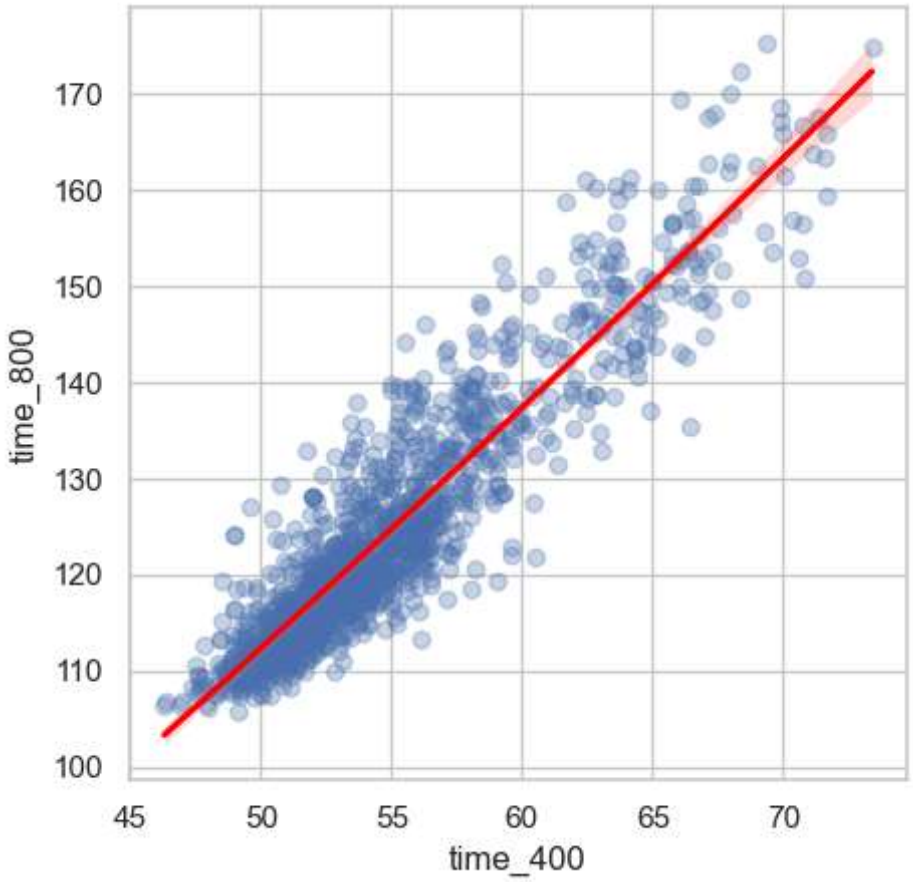
Omnibus:	121.579	**Durbin-Watson:**	1.545
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	191.748
Skew:	0.649	**Prob(JB):**	2.30e-42
Kurtosis:	4.293	**Cond. No.**	649.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Quadratic

In [ ]:
```
model_400_quad_m = BivariateModel(df_m, outcome_event=800, predictor_event=400, mod
model_400_quad_m.check_assumptions()
model_400_quad_m.plot_conf_int()
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
Scale-Location
√|Standardized Residuals|
Fitted values
Residuals vs Leverage
Cook's distance
High leverage
Standardized Residuals
Leverage
398
1086
1067
1202



In [ ]:
```
model_400_quad_m.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.846
Model:	OLS	**Adj. R-squared:**	0.846
Method:	Least Squares	**F-statistic:**	3771.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:19	**Log-Likelihood:**	-4155.6
No. Observations:	1371	**AIC:**	8317.
Df Residuals:	1368	**BIC:**	8333.
Df Model:	2		
Covariance Type:	nonrobust		

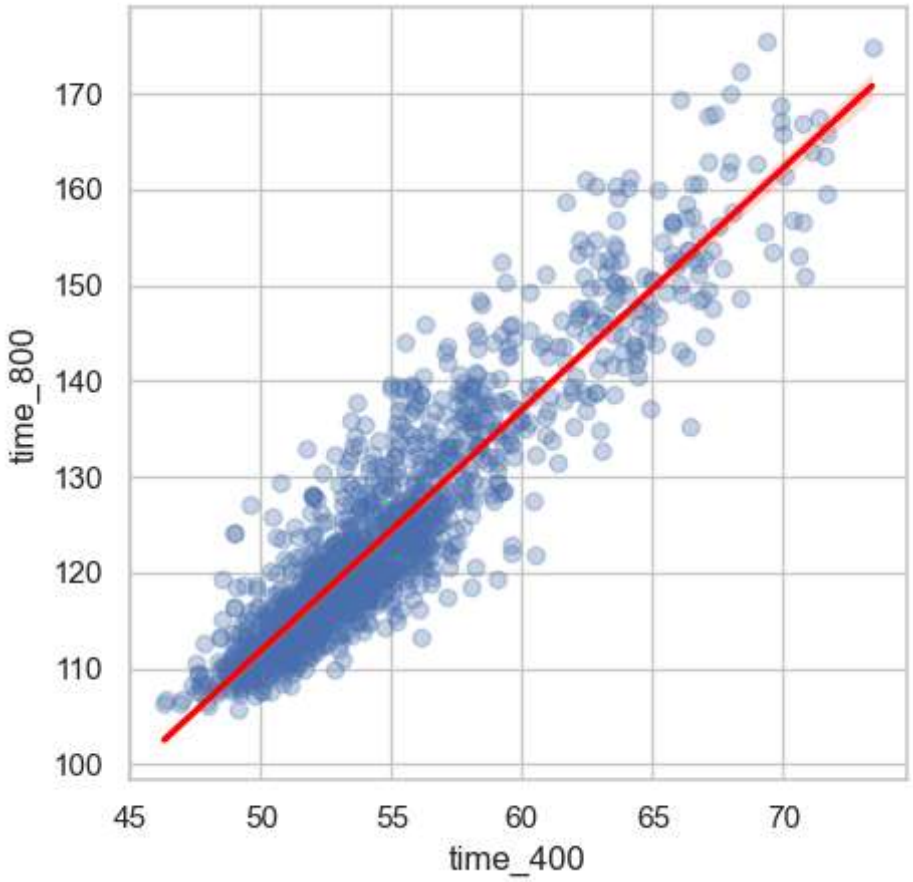
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7753	15.144	0.051	0.959	-28.932	30.483
x1	2.0042	0.527	3.801	0.000	0.970	3.039
x2	0.0045	0.005	0.998	0.319	-0.004	0.013

Omnibus:	124.950	**Durbin-Watson:**	1.549
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	199.148
Skew:	0.659	**Prob(JB):**	5.70e-44
Kurtosis:	4.322	**Cond. No.**	3.43e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.43e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

In [ ]:
```
model_400_robust_m = BivariateModel(df_m, outcome_event=800, predictor_event=400, m
model_400_robust_m.check_assumptions()
model_400_robust_m.plot_conf_int()
```



In [ ]:
```
model_400_robust_m.model_summary
```

Out[ ]:

Robust linear Model Regression Results

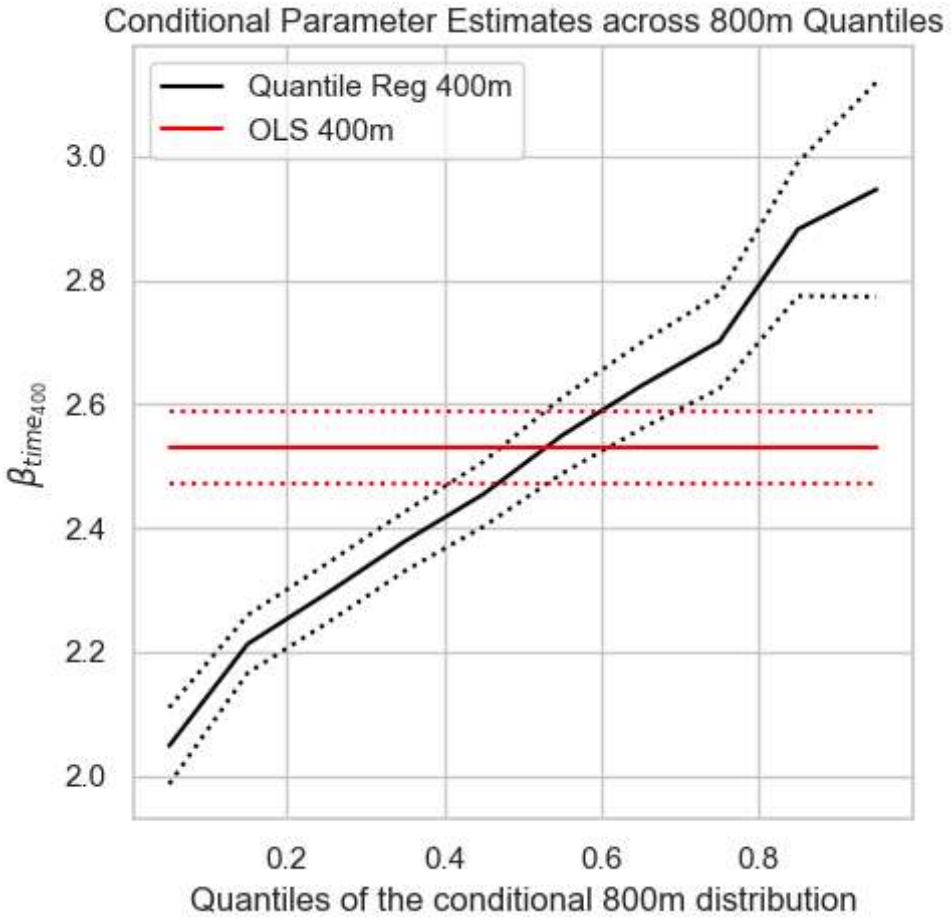
Dep. Variable:	time_800	**No. Observations:**	1371
Model:	RLM	**Df Residuals:**	1369
Method:	IRLS	**Df Model:**	1
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Sun, 14 Apr 2024		
Time:	16:24:34		
No. Iterations:	18		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-13.9552	1.406	-9.922	0.000	-16.712	-11.199
time_400	2.5170	0.026	98.312	0.000	2.467	2.567

If the model instance has been used for another fit with different fit parameters, then the fit options might not be the correct ones anymore .

Quantile Regression

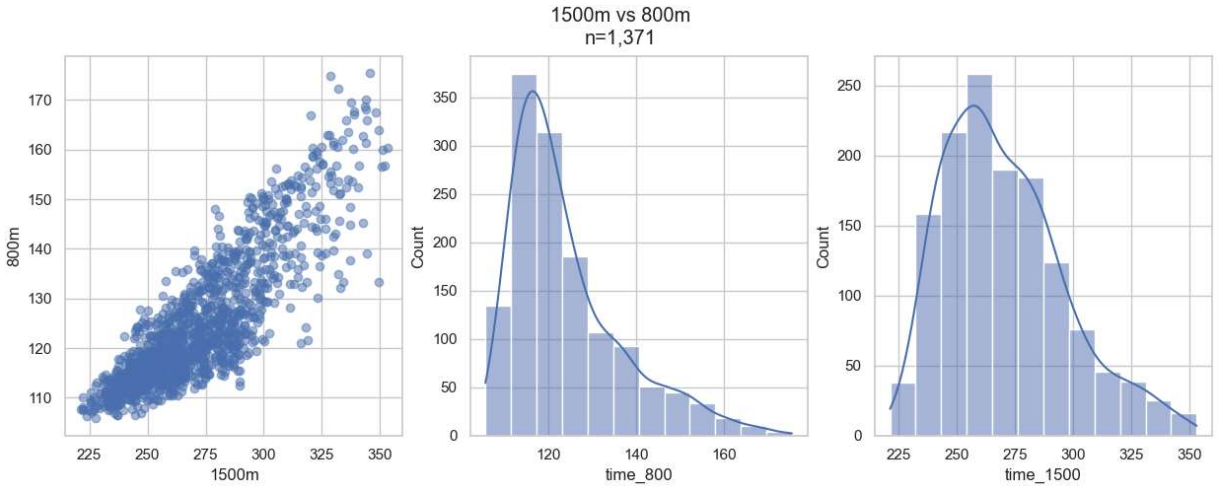
In [ ]:
```
model_400_quant_m = BivariateModel(df_m, outcome_event=800, predictor_event=400, mo
model_400_quant_m.plot_quantiles_by_parameter()
```

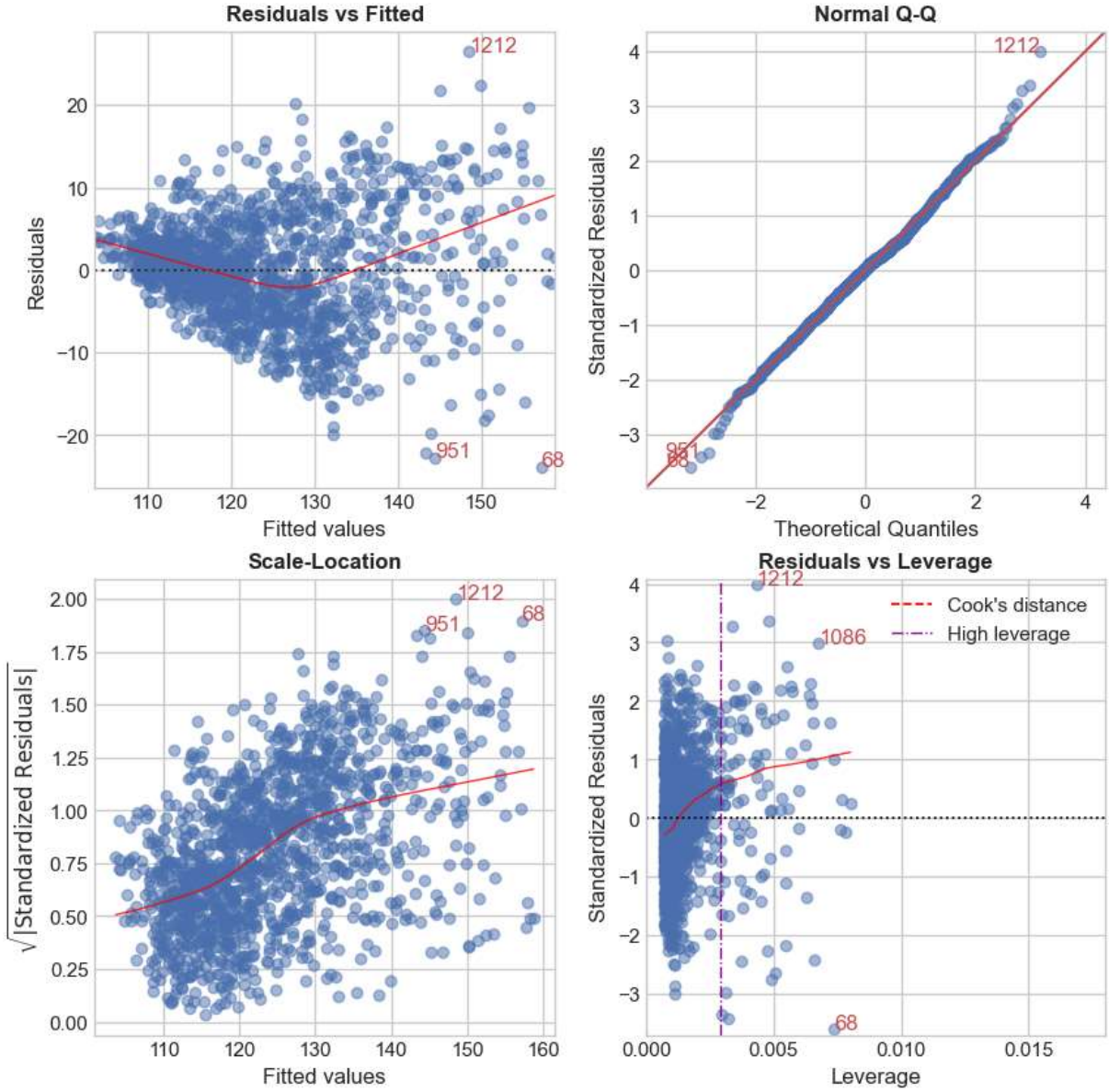


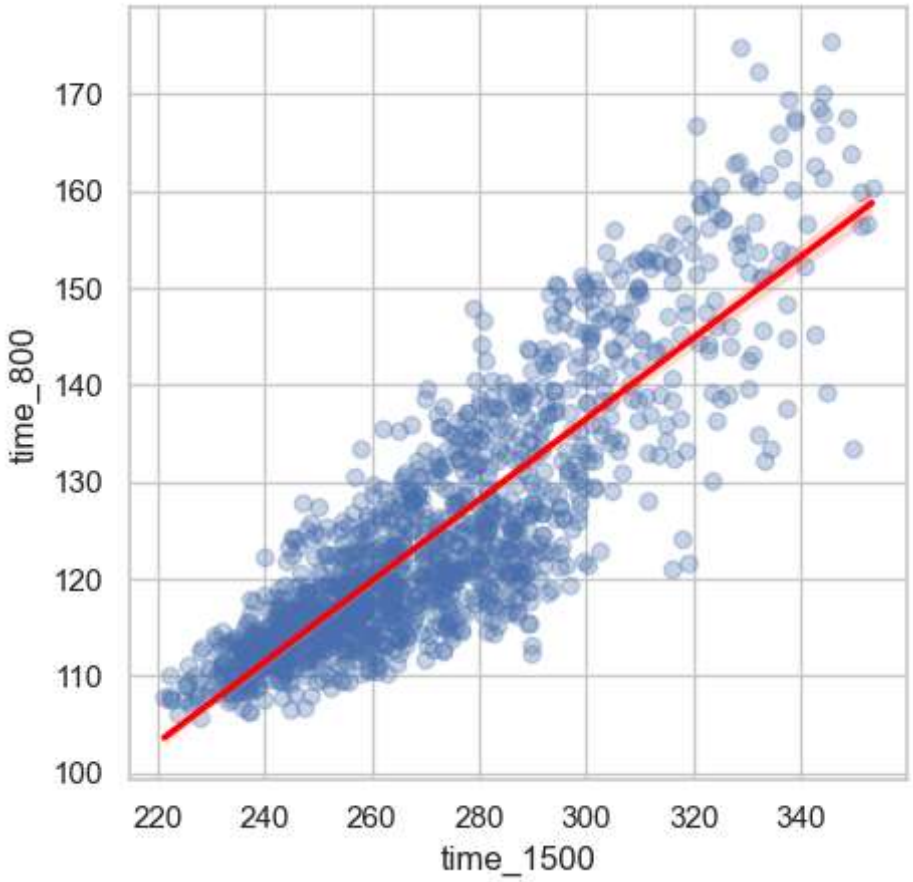
800m vs 1500m

OLS

In [ ]:
```
model_1500_m = BivariateModel(df_m, outcome_event=800, predictor_event=1500, model_
model_1500_m.plot_dist()
model_1500_m.check_assumptions()
model_1500_m.plot_conf_int()
```




Residuals vs Fitted
Residuals
Fitted values
1212
951
68
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
1212
951
68
Scale-Location
√|Standardized Residuals|
Fitted values
1212
951
68
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1212
1086
68



In [ ]:
```
model_1500_m.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.731
Model:	OLS	**Adj. R-squared:**	0.731
Method:	Least Squares	**F-statistic:**	3716.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:38	**Log-Likelihood:**	-4540.6
No. Observations:	1371	**AIC:**	9085.
Df Residuals:	1369	**BIC:**	9096.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	11.2320	1.862	6.032	0.000	7.579	14.885
time_1500	0.4177	0.007	60.959	0.000	0.404	0.431

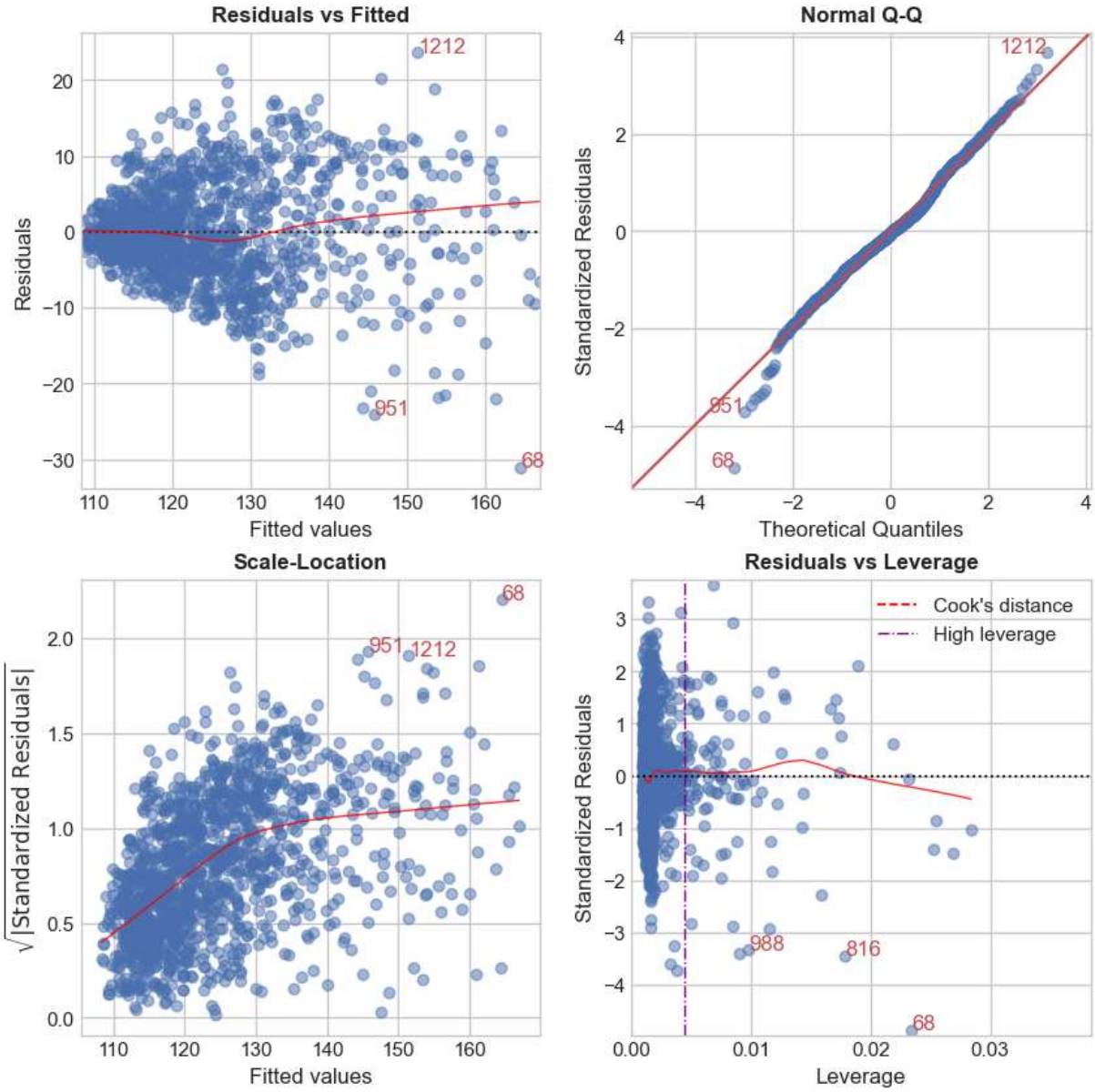
Omnibus:	7.039	**Durbin-Watson:**	1.387
Prob(Omnibus):	0.030	**Jarque-Bera (JB):**	8.913
Skew:	0.039	**Prob(JB):**	0.0116
Kurtosis:	3.387	**Cond. No.**	2.82e+03

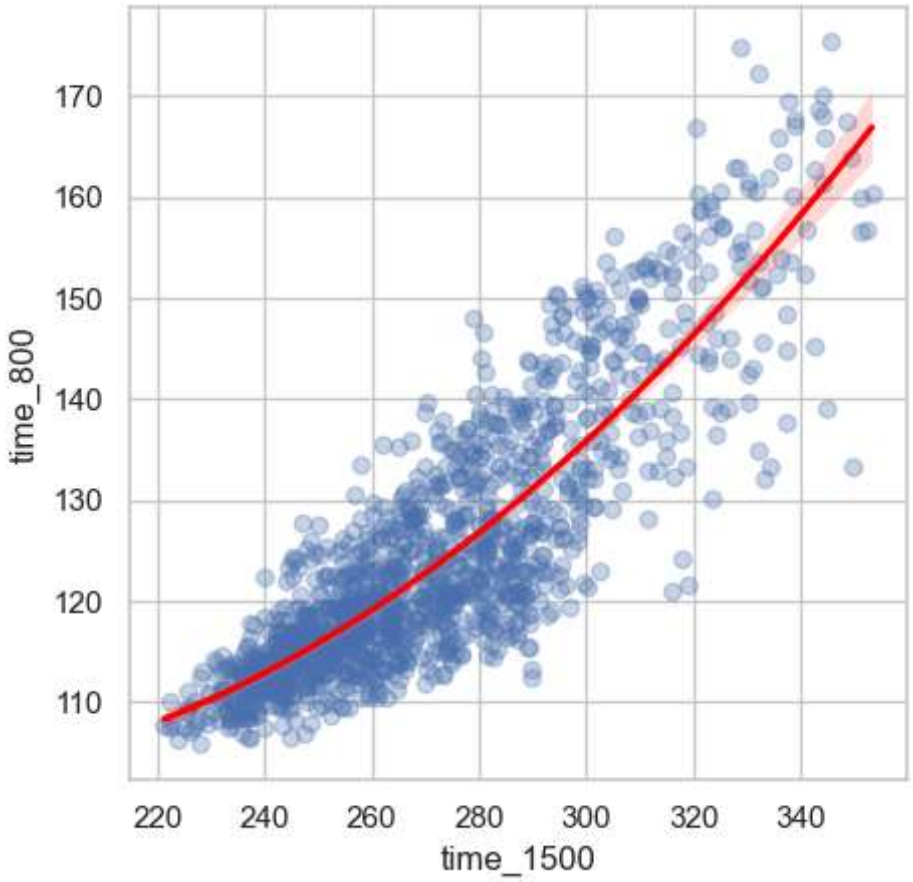
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.82e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Quadratic

In [ ]:

```
model_1500_quad_m = BivariateModel(df_m, outcome_event=800, predictor_event=1500, m
model_1500_quad_m.check_assumptions()
model_1500_quad_m.plot_conf_int()
```


Residuals vs Fitted
Residuals
Fitted values
1212
951
68
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
1212
951
68
Scale-Location
√|Standardized Residuals|
Fitted values
68
951
1212
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
988
816
68



In [ ]:
```
model_1500_quad_m.model_summary
```

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.745
Model:	OLS	**Adj. R-squared:**	0.745
Method:	Least Squares	**F-statistic:**	2001.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:24:41	**Log-Likelihood:**	-4502.7
No. Observations:	1371	**AIC:**	9011.
Df Residuals:	1368	**BIC:**	9027.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	148.0715	15.613	9.484	0.000	117.443	178.700
x1	-0.5701	0.112	-5.084	0.000	-0.790	-0.350
x2	0.0018	0.000	8.824	0.000	0.001	0.002

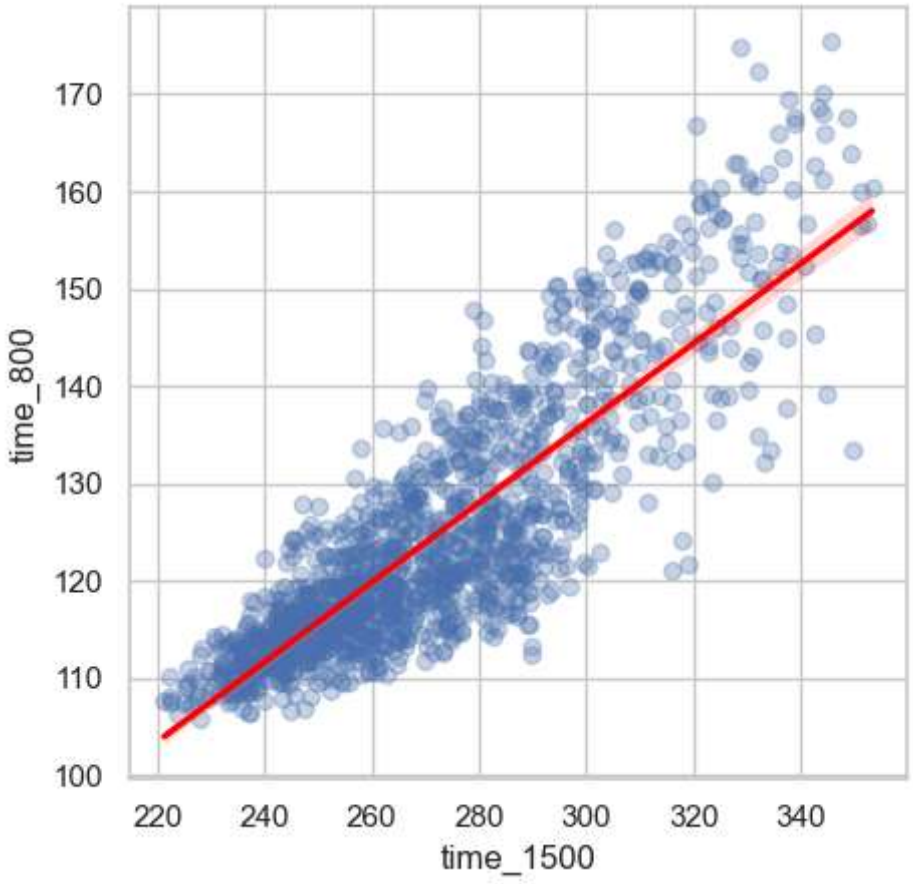
Omnibus:	23.960	**Durbin-Watson:**	1.436
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	45.010
Skew:	0.025	**Prob(JB):**	1.68e-10
Kurtosis:	3.886	**Cond. No.**	6.73e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.73e+06. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

In [ ]:

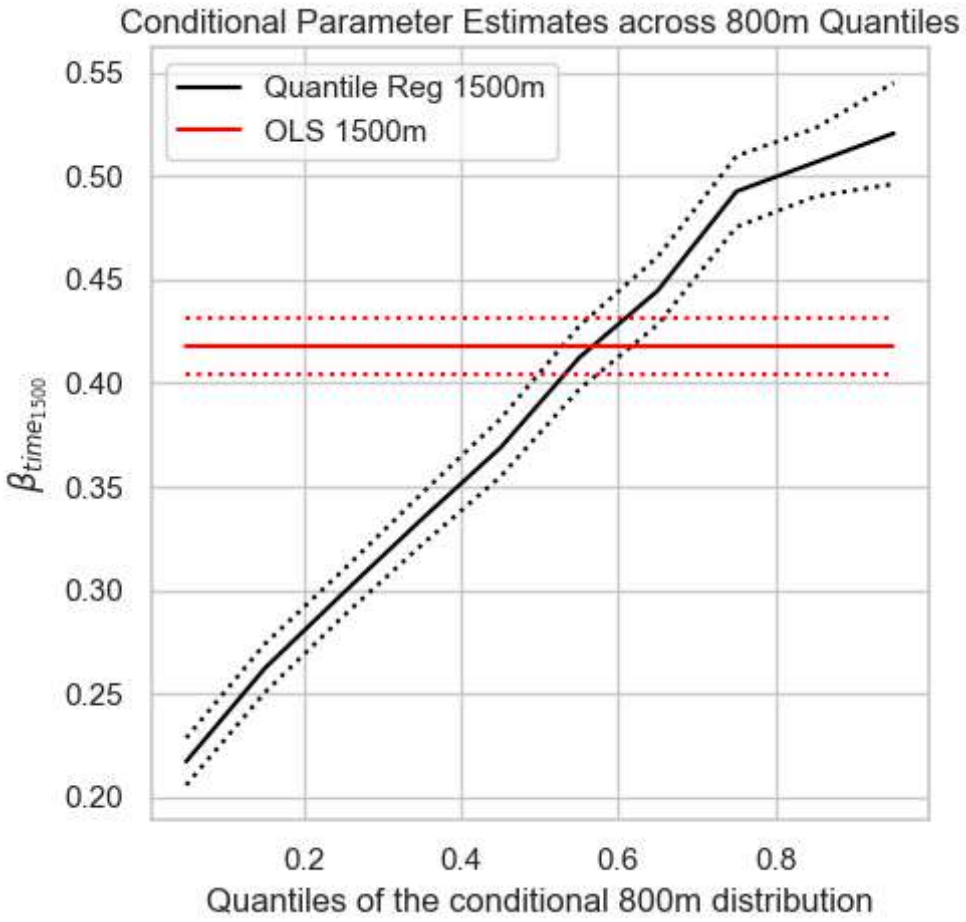
```
model_1500_robust_m = BivariateModel(df_m, outcome_event=800, predictor_event=1500,
model_1500_robust_m.check_assumptions()
model_1500_robust_m.plot_conf_int()
```



Quantile Regression

In [ ]:
```
model_1500_quant_m = BivariateModel(df_m, outcome_event=800, predictor_event=1500,
model_1500_quant_m.plot_quantiles_by_parameter()
```

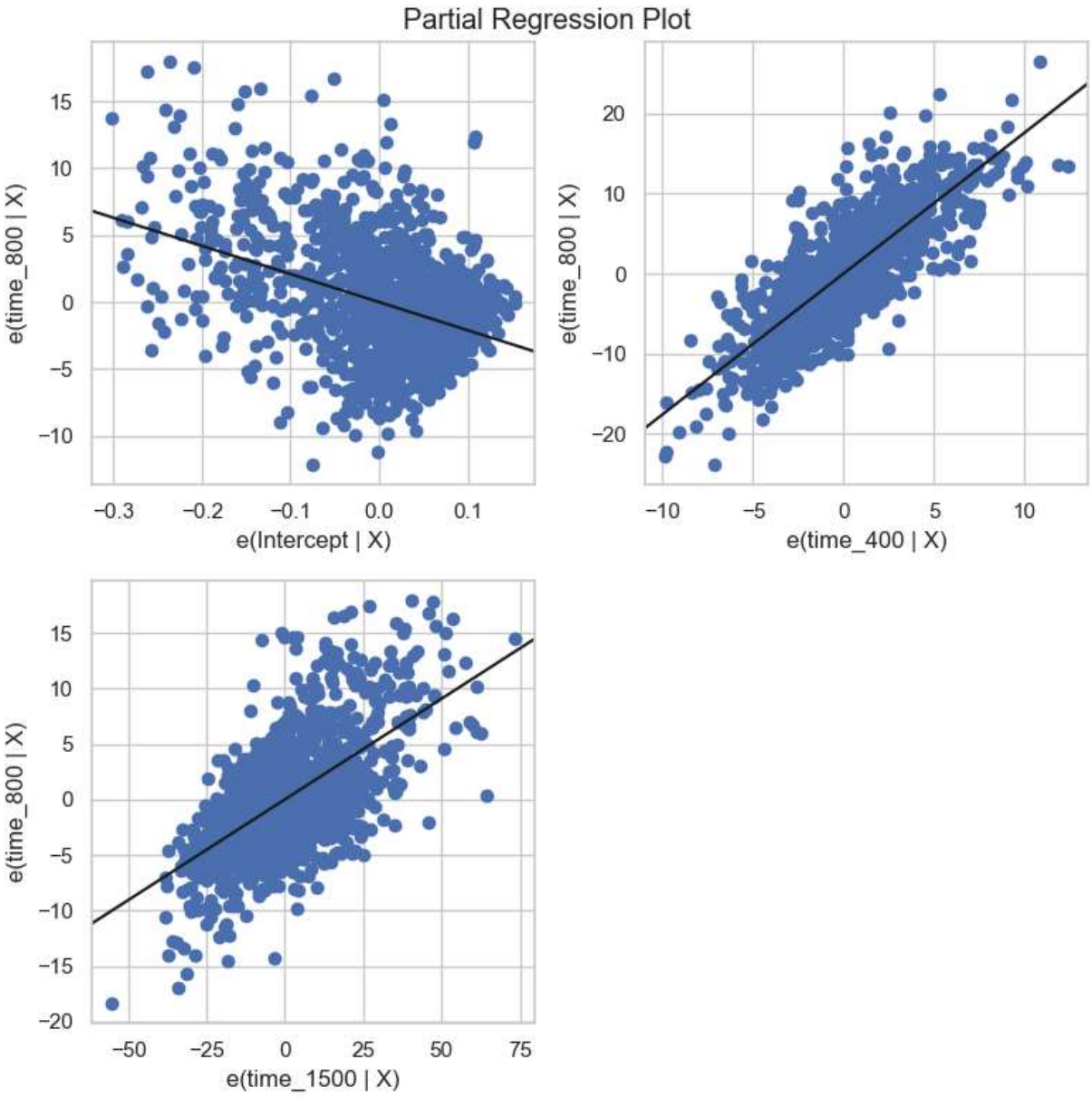
```
c:\Users\mitch\miniconda3\envs\webpy\Lib\site-packages\statsmodels\regression\quanti
le_regression.py:191: IterationLimitWarning: Maximum number of iterations (1000) rea
ched.
  warnings.warn("Maximum number of iterations (" + str(max_iter) +
c:\Users\mitch\miniconda3\envs\webpy\Lib\site-packages\statsmodels\regression\quanti
le_regression.py:191: IterationLimitWarning: Maximum number of iterations (1000) rea
ched.
  warnings.warn("Maximum number of iterations (" + str(max_iter) +
```

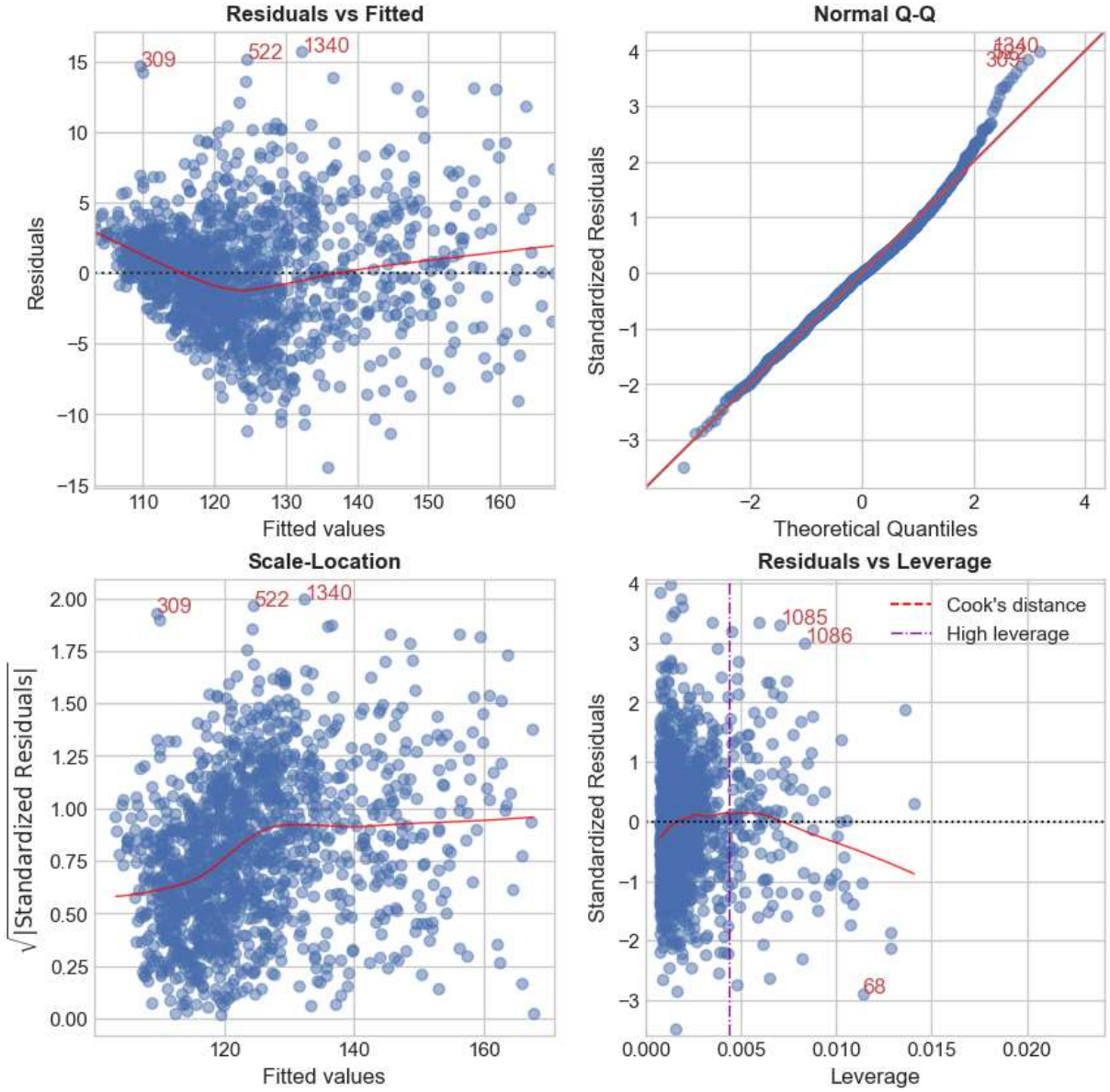


800m vs 400m + 1500m

OLS

In [ ]:
```
model_400_1500_m = MultivariateModel(df_m, outcome_event=800, predictor_events=[400
model_400_1500_m()
```


Partial Regression Plot
e(time_800 | X)
e(Intercept | X)
e(time_400 | X)
e(time_1500 | X)


Residuals vs Fitted
309
522
1340
Residuals
Fitted values
Normal Q-Q
1340
Standardized Residuals
Theoretical Quantiles
Scale-Location
309
522
1340
√|Standardized Residuals|
Fitted values
Residuals vs Leverage
Cook's distance
High leverage
1085
1086
68
Standardized Residuals
Leverage

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.905
Model:	OLS	**Adj. R-squared:**	0.905
Method:	Least Squares	**F-statistic:**	6543.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:25:01	**Log-Likelihood:**	-3824.0
No. Observations:	1371	**AIC:**	7654.
Df Residuals:	1368	**BIC:**	7670.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-21.0465	1.278	-16.471	0.000	-23.553	-18.540
time_400	1.7569	0.035	50.232	0.000	1.688	1.825
time_1500	0.1815	0.006	29.201	0.000	0.169	0.194

Omnibus:	50.332	**Durbin-Watson:**	1.540
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	73.146
Skew:	0.346	**Prob(JB):**	1.31e-16
Kurtosis:	3.895	**Cond. No.**	3.33e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.33e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Robust LM

In [ ]:
```
model_400_1500_robust_m = MultivariateModel(df_m, outcome_event=800, predictor_even
model_400_1500_robust_m()
```

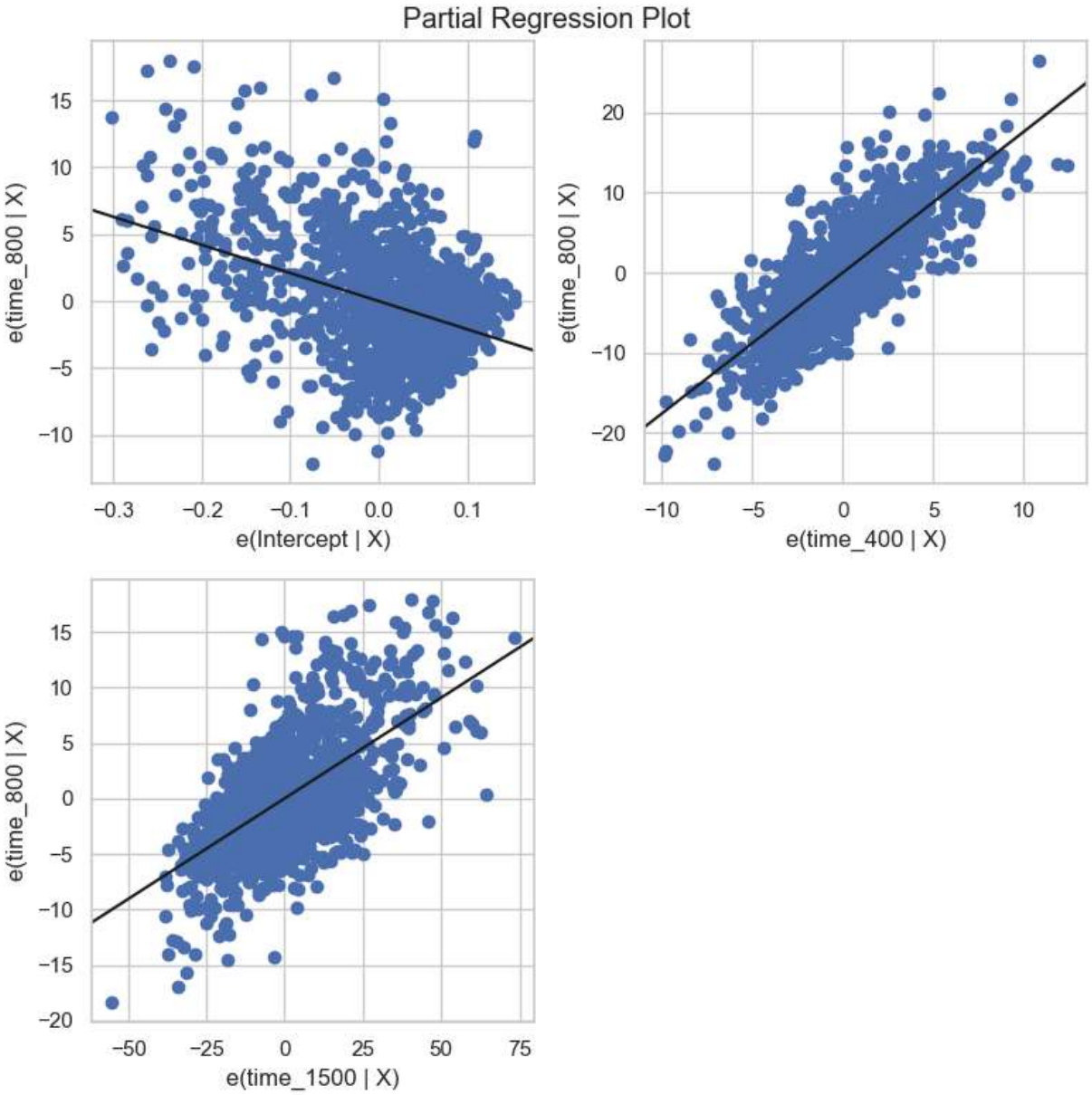
Out[ ]:

Robust linear Model Regression Results

Dep. Variable:	time_800	**No. Observations:**	1371
Model:	RLM	**Df Residuals:**	1368
Method:	IRLS	**Df Model:**	2
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Sun, 14 Apr 2024		
Time:	16:25:02		
No. Iterations:	24		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-19.9757	1.227	-16.284	0.000	-22.380	-17.571
time_400	1.7882	0.034	53.257	0.000	1.722	1.854
time_1500	0.1707	0.006	28.609	0.000	0.159	0.182

If the model instance has been used for another fit with different fit parameters, then the fit options might not be the correct ones anymore .

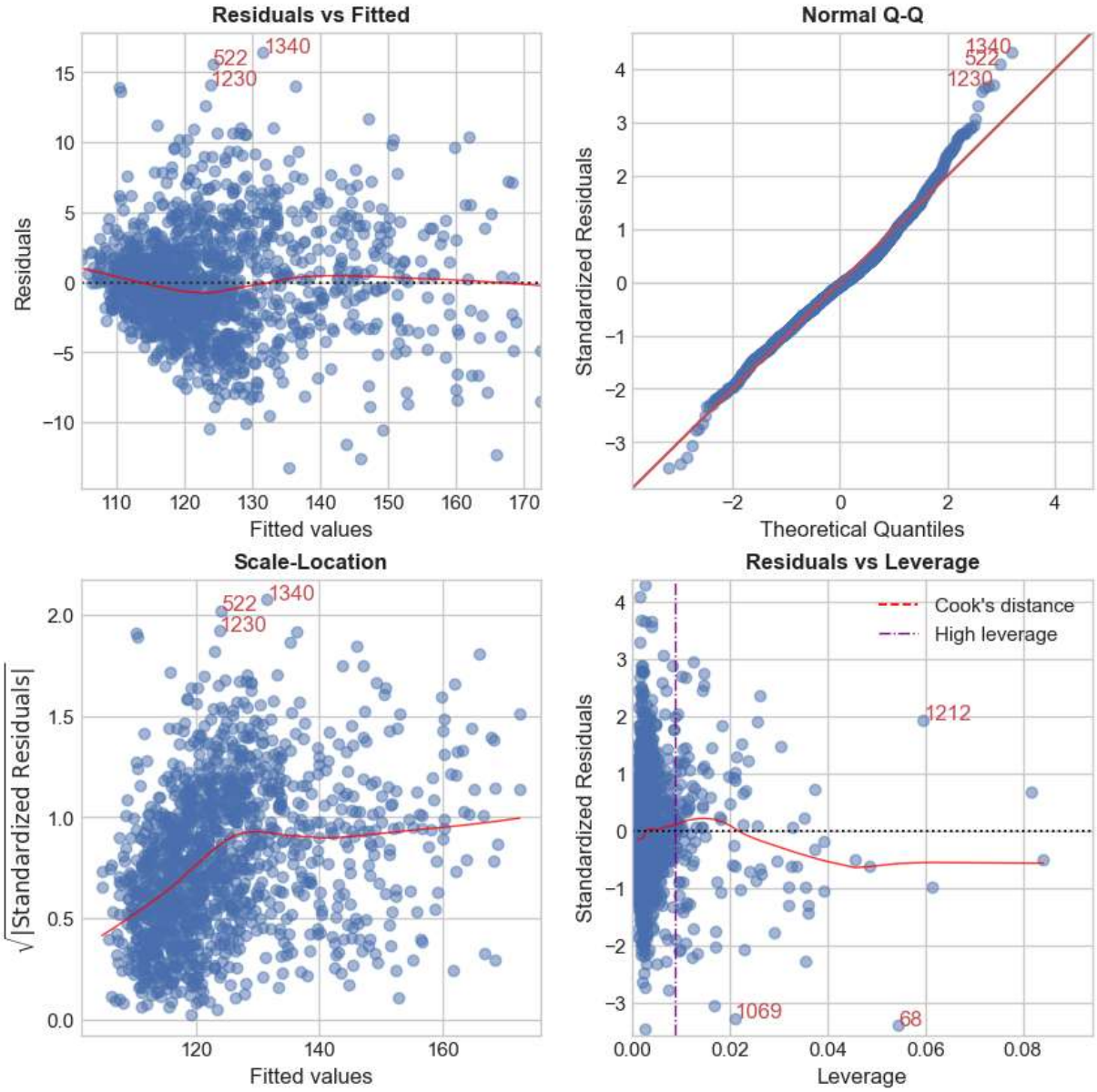


In [ ]:
```
model_400_1500_robust_m.predict_time(times = ['0:53.80', '4:18.40'], events=[400, 1
```

Out[ ]:
```
'800m Prediction: 117.59 seconds'
```

Quadratic

In [ ]:
```
model_400_1500_quad_m = MultivariateModel(df_m, outcome_event=800, predictor_events
model_400_1500_quad_m()
```


Residuals vs Fitted
Residuals
Fitted values
522
1340
1230
Normal Q-Q
Standardized Residuals
Theoretical Quantiles
1340
522
1230
Scale-Location
√|Standardized Residuals|
Fitted values
522
1340
1230
Residuals vs Leverage
Standardized Residuals
Leverage
Cook's distance
High leverage
1212
1069
68

Out[ ]:

OLS Regression Results

Dep. Variable:	time_800	**R-squared:**	0.912
Model:	OLS	**Adj. R-squared:**	0.911
Method:	Least Squares	**F-statistic:**	2815.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	16:25:05	**Log-Likelihood:**	-3777.3
No. Observations:	1371	**AIC:**	7567.
Df Residuals:	1365	**BIC:**	7598.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	64.5103	12.293	5.248	0.000	40.395	88.626
x1	1.7289	0.471	3.668	0.000	0.804	2.654
x2	-0.4305	0.080	-5.406	0.000	-0.587	-0.274
x3	-0.0322	0.007	-4.399	0.000	-0.047	-0.018
x4	0.0129	0.002	5.689	0.000	0.008	0.017
x5	-0.0002	0.000	-0.737	0.461	-0.001	0.000

Omnibus:	73.630	**Durbin-Watson:**	1.584
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	118.150
Skew:	0.432	**Prob(JB):**	2.21e-26
Kurtosis:	4.150	**Cond. No.**	9.17e+06

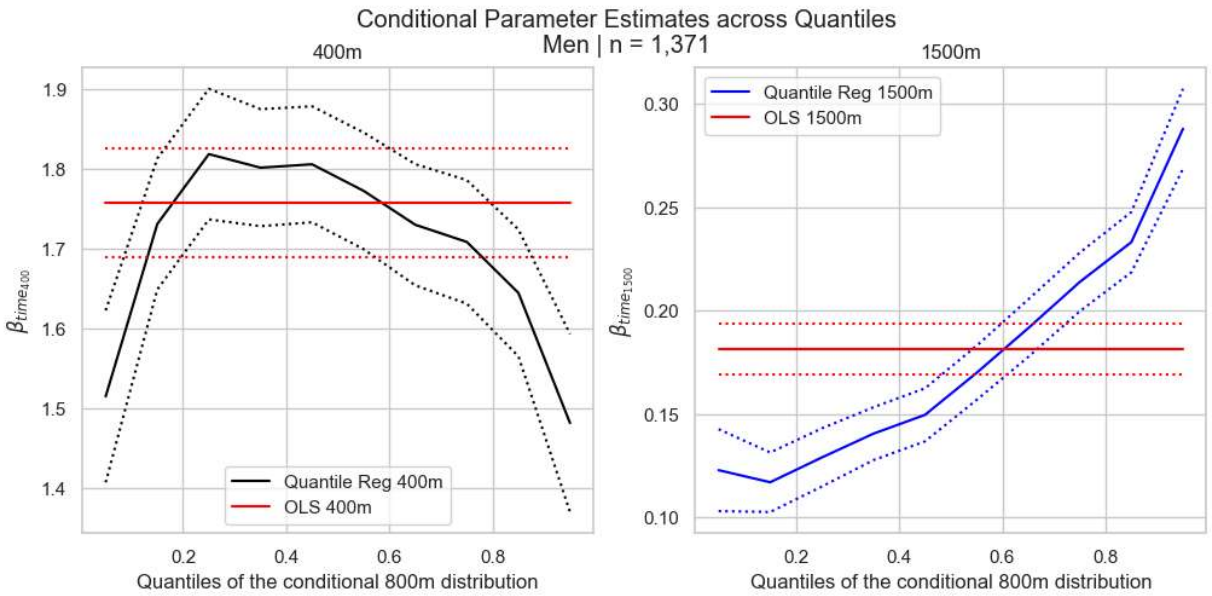
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.17e+06. This might indicate that there are strong multicollinearity or other numerical problems.

Quantile Regression

Cleaned Data

In [ ]:

```
model_400_1500_quant_m = MultivariateModel(df_m, outcome_event=800, predictor_event
model_400_1500_quant_m.plot_quantiles_by_parameter(title_specifier='Men')
```



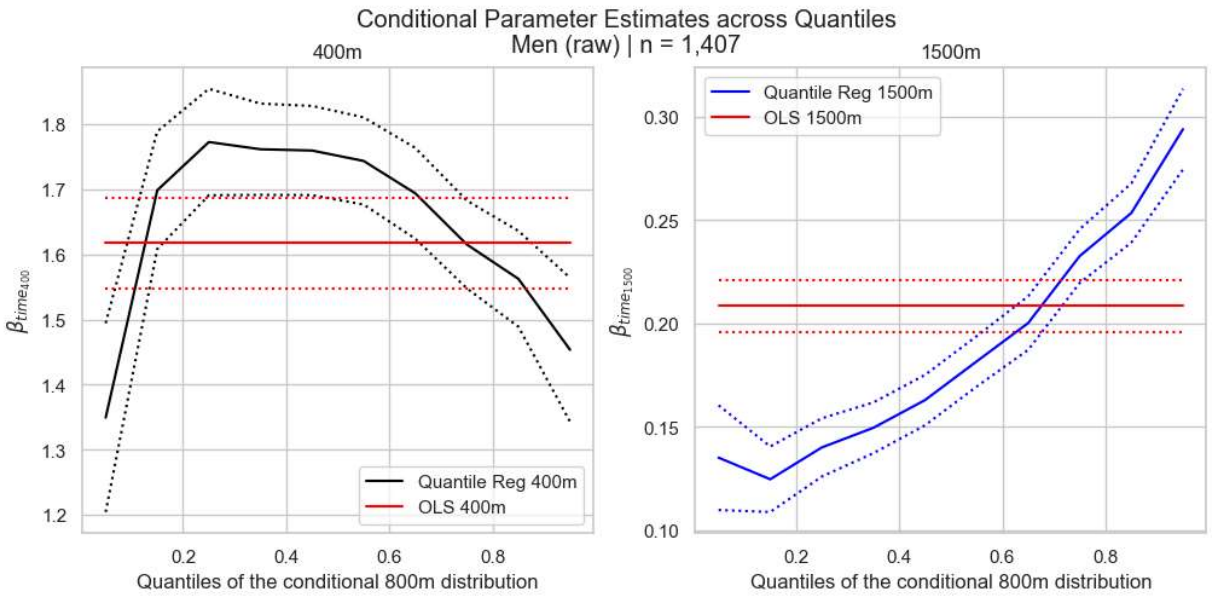
In [ ]:
```
make_quantile_table(df_m, 'time_800')
```

Out[ ]:

	quantile	time_800m
0	0.05	110.09
1	0.20	114.02
2	0.40	118.14
3	0.60	123.09
4	0.80	133.40
5	0.95	151.58

Raw Data

In [ ]:
```
model_raw_400_1500_m = MultivariateModel(df_raw.loc[df_raw['sex']=='m'], outcome_ev
model_raw_400_1500_m.plot_quantiles_by_parameter(title_specifier='Men (raw)')
```



In [ ]:
```
make_quantile_table(df_raw.loc[df_raw['sex']=='m'], 'time_800')
```

Out[ ]:

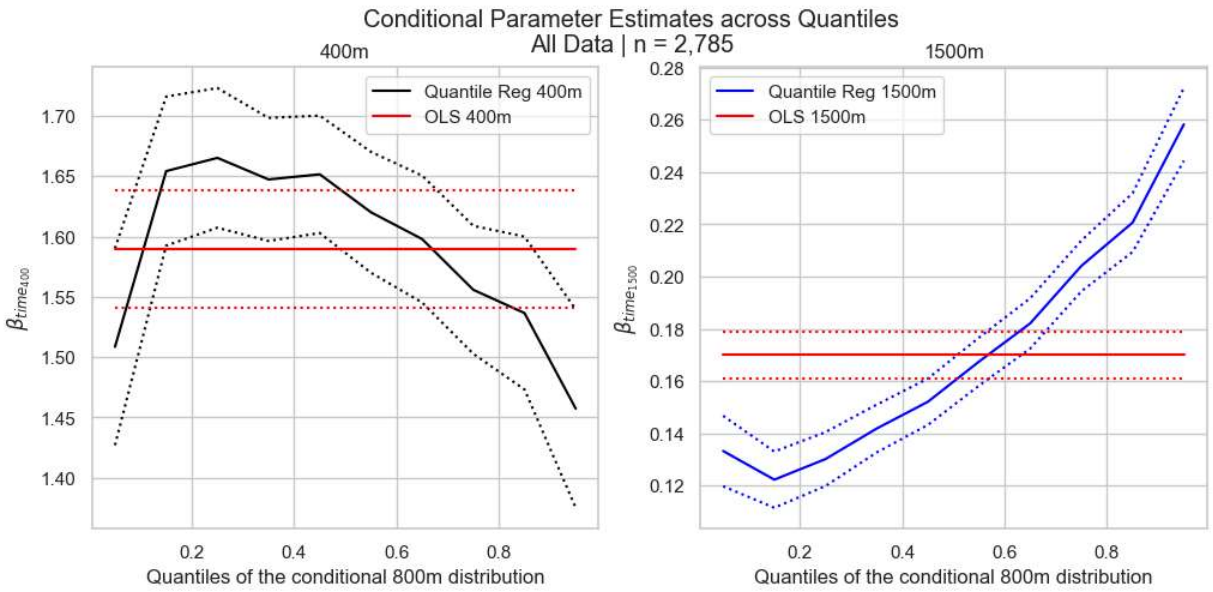
	quantile	time_800m
0	0.05	110.14
1	0.20	114.16
2	0.40	118.35
3	0.60	123.49
4	0.80	134.88
5	0.95	153.87

Both Sexes

800m vs 400m + 1500m

Quantile Regression

In [ ]:
```
model_400_1500 = MultivariateModel(df_cleaned, outcome_event=800, predictor_events=
model_400_1500.plot_quantiles_by_parameter(title_specifier='All Data')
```

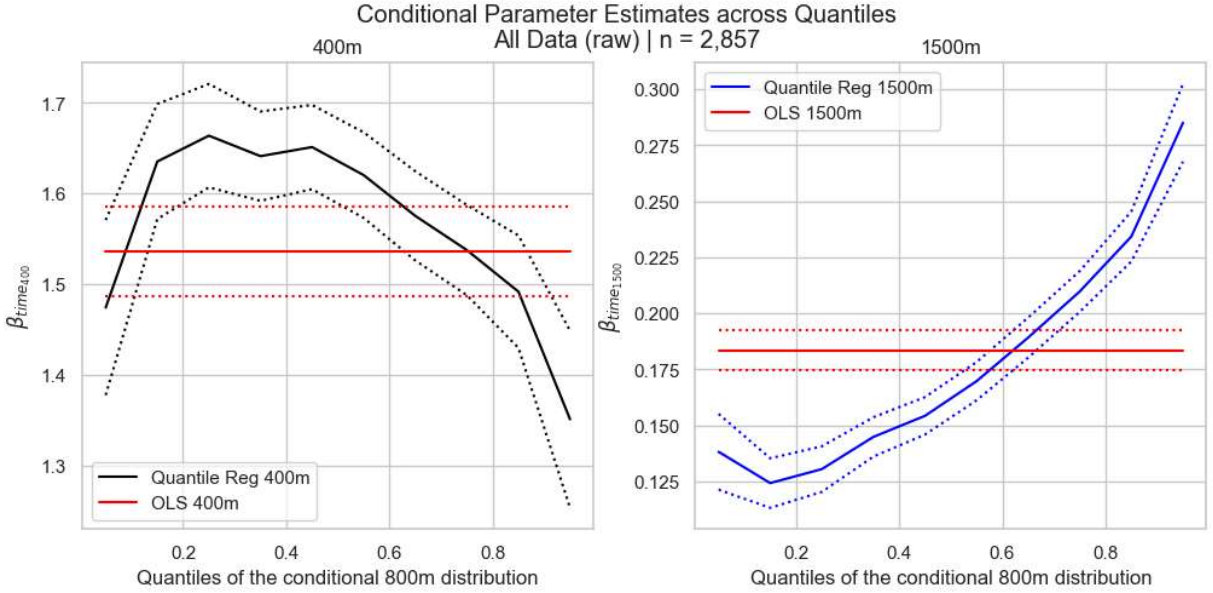


In [ ]:
```
make_quantile_table(df_cleaned, 'time_800')
```

Out[ ]:

	quantile	time_800m
0	0.05	111.69
1	0.20	118.28
2	0.40	128.69
3	0.60	138.62
4	0.80	151.38
5	0.95	167.74

In [ ]:
```
model_400_1500_raw = MultivariateModel(df_raw, outcome_event=800, predictor_events=
model_400_1500_raw.plot_quantiles_by_parameter(title_specifier = 'All Data (raw)')
```



In [ ]:
```
make_quantile_table(df_raw, 'time_800')
```

Out[ ]:

	quantile	time_800m
0	0.05	111.77
1	0.20	118.50
2	0.40	129.25
3	0.60	139.10
4	0.80	152.15
5	0.95	169.50

400m speed plays a relatively constant influence between 2:05 and 2:29, while working on aerobic fitness helps all the way down to 2:05. Going faster than 2:05 is when you really start to see 400m speed reduce 800m time. So, 400m time doesn't have a huge influence until you have the aerobic fitness to use that speed.