MSBA 6440 Project Report

Spotify Sequential Skip Analysis

Group 13: Zhanglin Shangguan, Devansh Bhasin, Shravan Panneer, Yutong Liu

Executive Summary

Spotify is the largest music streaming service in the world as of the second quarter of 2021 with a market share of 31 percent. As of the first quarter of 2022, Spotify had 182 million premium subscribers up from 158 million in the corresponding quarter of 2021[1]. Spotify's subscriber base has increased dramatically in the last few years and has more than doubled since early 2017. Recommending relevant songs to users is a key component of Spotify's strategy to keep customers engaged on their platform. Spotify uses customized playlists called "Discover Weekly" to recommend songs to users tailor-made for them based on Spotify's recommendation system. Understanding customers is a key part of Spotify's customer engagement strategy and their overall success.

We decided to utilize Spotify's Skip Data Challenge to better understand the causal reasons behind why customers skip certain songs[2]. The dataset is composed of 'session' level data that contains the list of songs played in a single session by a user and which songs were skipped. We want to understand what impact being a 'premium user' has on the number of songs being skipped per session.

Business Problem

The causal question we want to answer is **"How does a Premium Subscription affect a user's total skip per session?"** The reason this is an important question to answer relates to the industry trends of the streaming service. From Amazon Prime Video to Netflix to Hulu, streaming platforms across the industry have a finely tuned customer recommendation system that ties into increasing customer engagement and customer satisfaction[3]. The idea is to keep users regularly returning to the platform regularly and increase view or listen times. This is done by recommending them content such as movies or songs that they would like based on various customer level metrics and what similar customers like and listen to. These so-called 'recommendation engines' rank content using a Machine Learning algorithm to accurately identify what songs a user prefers. It is constantly updated as the user continues to interact with the site and consume content. Understanding why customers skip certain songs can help us identify why users dislike songs that are recommended to them by Spotify. This can help the company improve its recommendation engine and tailor songs for Premium and Non-Premium users.

The Dataset

We used data from the Spotify Sequential Skip Prediction Challenge to address the causal relationship between treatment and number of skips. This data is observed over the year of 2019, and the entire dataset is roughly around 130 million records, but we are using a mini version of the original dataset with around 170K records. There are two parts to our dataset including track-level data and session-level data. However, we only take all session-level features into account because it contains user-behavior information, and each session id has information of corresponding 20 tracks. The feature Premium is the treatment, and the number of skips per session is the outcome variable for this study.

Session-level data includes the following features:

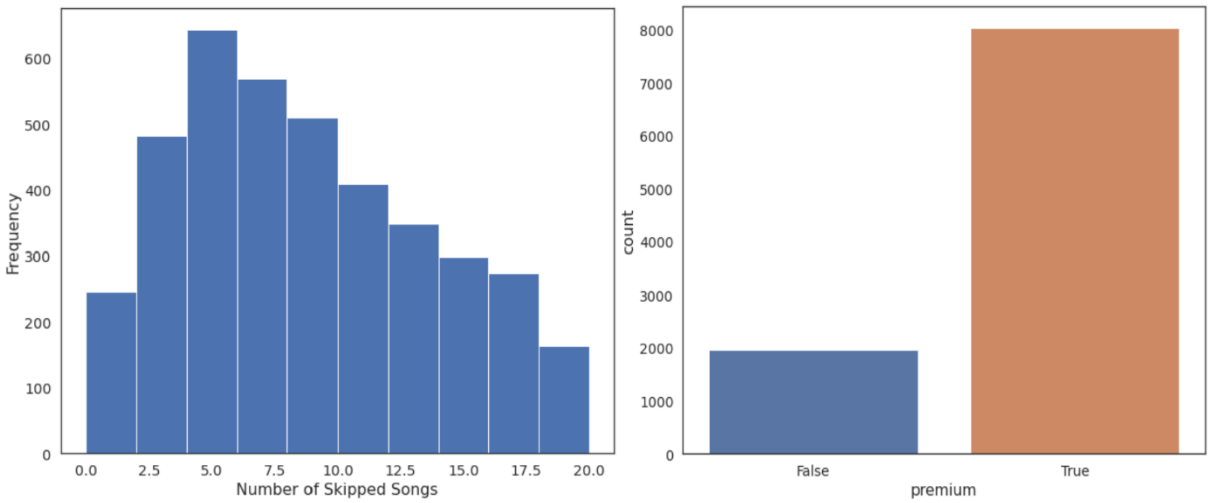
short_pause_before_play	Boolean indicating if there was a short pause between playback of the previous track and this track
long_pause_before_play	Boolean indicating if there was a long pause between playback of the previous track and this track
hist_user_behavior_n_seekfwd	Number of times the user did a seek forward within track
hist_user_behavior_n_seekback	Number of times the user did a seek back within track
hist_user_behavior_is_shuffle	Boolean indicating if the user encountered this track while shuffle mode was activated
hour_of_day	{0-23} - The hour of day
date	E.g. 2018-09-18 - The date
premium	Boolean indicating if the user was on premium or not. This has potential implications for skipping behavior.
context_type	E.g. editorial playlist - what type of context the playback occurred within
hist_user_behavior_reason_start	E.g. fwdbtn - the user action which led to the current track being played
hist_user_behavior_reason_end	E.g. trackdone - the user action which led to the current track playback ending

Data Description of Session Level Data

We performed some data cleaning, converting data types of variables into the right form for aggregation, and we used skip_2 as the outcome variable and dropped other skip columns. To aggregate for session-level data, we summed up the numerical variables including short_pause_number, long_pause_number, np_pause_number, n_seekfwd, n_seekback, and the number of skipped and kept track of the categorical variable including if_switch_playlist, hour_of_day, premium from session-level data of corresponding 20 tracks. Following are the aggregated features:

short_pause_number	Number of short pause songs
long_pause_number	Number of long pause songs
np_pause_number	Number of no pause songs
hist_user_behavior_n_seekfwd	Seek forward times within the session
hist_user_behavior_n_seekback	Seek backward times within the session
context_switch	If switched playlist during the session
skipped	If the song is being skipped
premium	If the user is premium

If we take a look at the distribution of the treatment and outcome variable, we can see that the number of skipped songs is very close to the normal distribution, and the premium session accounts for 80% of the data.

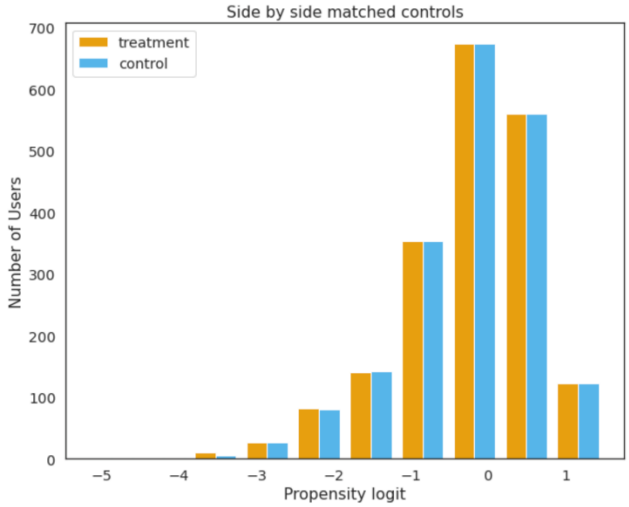


Data Distribution Visualizations

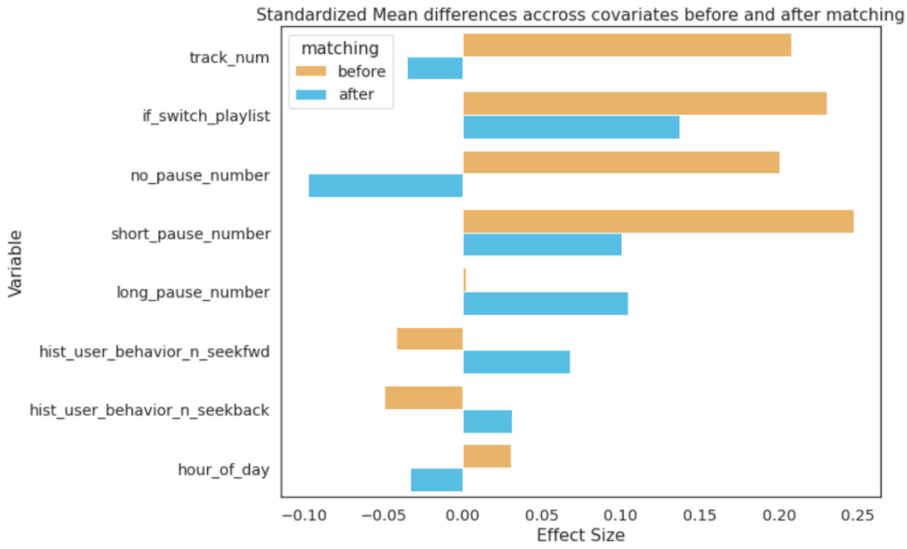
Methods and Main Analysis

For the inference method, we leveraged the propensity score matching to match the users/sessions based on the track number, if switched playlist during the session, number of no pause songs, number of short pause songs, number of long pause songs, seek forward times within the session, seekback times within the session, and hour of the day when listen to tracks during the session. Our treatment is whether the user is premium or not, and the outcome variable is the number of skipped songs in the session with the one session/user as an observation unit.

We used the 1:1 matching according to the propensity logit as the main setting. And our matching result is as below



.



We can observe that after matching the treatment and control group can be balanced within each propensity logit. Besides, the standardized mean difference across each covariate largely decreased after matching.

To obtain the effect of premium users on the skipped number within the matched data, we regress the skipped number on the if_premuim and get the following result:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.002
Model:	OLS	**Adj. R-squared:**	0.002
Method:	Least Squares	**F-statistic:**	8.396
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.00378
Time:	23:34:46	**Log-Likelihood:**	-10162.
No. Observations:	3361	**AIC:**	2.033e+04
Df Residuals:	3359	**BIC:**	2.034e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	8.6223	0.112	76.881	0.000	8.402	8.842
C(premium)[T.True]	-0.5052	0.174	-2.898	0.004	-0.847	-0.163

Omnibus:	595.010	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	193.918
Skew:	0.361	**Prob(JB):**	7.78e-43
Kurtosis:	2.070	**Cond. No.**	2.46

OLS Regression Result with Matched Data

We get the following result with unmatched data:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.3437
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.558
Time:	23:14:11	**Log-Likelihood:**	-30431.
No. Observations:	10000	**AIC:**	6.087e+04
Df Residuals:	9998	**BIC:**	6.088e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	8.6223	0.114	75.417	0.000	8.398	8.846
C(premium)[T.True]	0.0748	0.128	0.586	0.558	-0.175	0.325

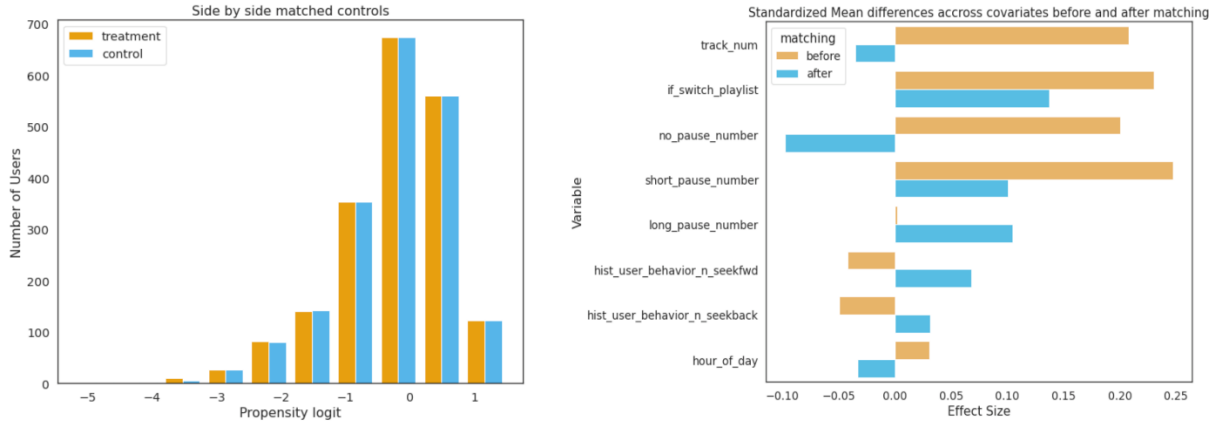
Omnibus:	2832.459	**Durbin-Watson:**	1.971
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	568.523
Skew:	0.274	**Prob(JB):**	3.52e-124
Kurtosis:	1.969	**Cond. No.**	4.30

OLS Regression Result with Unmatched Data

Sensitivity Analysis

To check the robustness of the inference method, we used different settings for the matching method.

1) Use propensity logit with replacement: Instead of 1 to 1 matching, we changed to 1 to many and still used the propensity score as the propensity function to match the data. As shown in the following graph, the data can still be balanced and the difference decreased after matching as well.

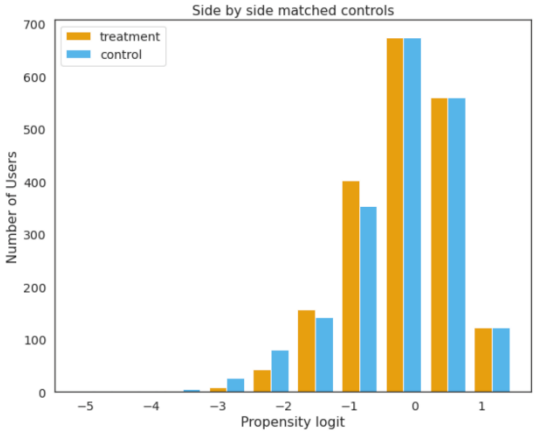


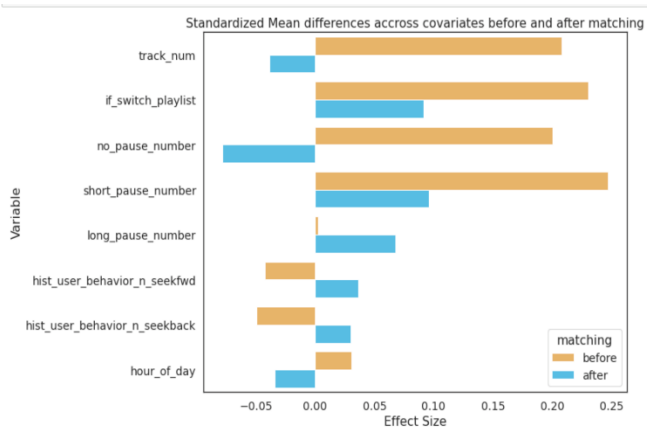
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.112	76.881	0.000	8.402	8.842
C(premium)[T.True]	-0.5052	0.174	-2.898	0.004	-0.847	-0.163

Omnibus:	595.010	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	193.918
Skew:	0.361	**Prob(JB):**	7.78e-43
Kurtosis:	2.070	**Cond. No.**	2.46

OLS Regression Result: Logit with Replacement

2) Use propensity score without replacement: After changing to using the propensity score instead of propensity logit to match the data, we obtained the following result, which is similar to the main setting.



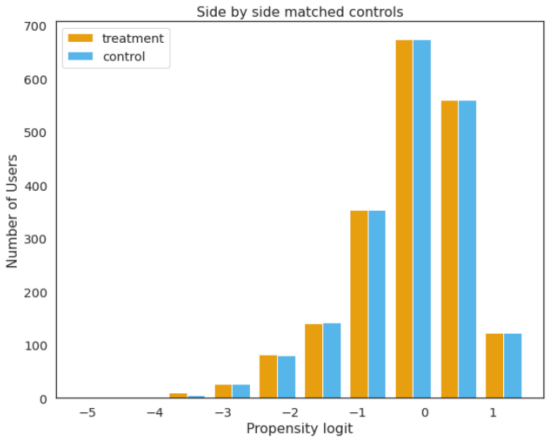


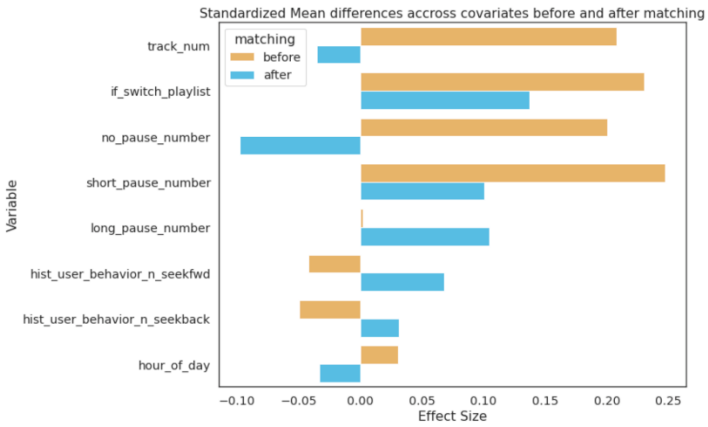
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.111	77.647	0.000	8.405	8.840
C(premium)[T.True]	-0.6076	0.157	-3.869	0.000	-0.916	-0.300

Omnibus:	606.994	**Durbin-Watson:**	1.951
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	224.719
Skew:	0.377	**Prob(JB):**	1.60e-49
Kurtosis:	2.106	**Cond. No.**	2.62

OLS Regression Result: Logit without Replacement

3) Use propensity logit with replacement and caliper=0.005: We changed the caliper to 0.005 instead of none, which can have limit on the matching scope. And the result is still stable as seen below:





	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.112	76.881	0.000	8.402	8.842
C(premium)[T.True]	-0.5052	0.174	-2.898	0.004	-0.847	-0.163

Omnibus:	595.010	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	193.918
Skew:	0.361	**Prob(JB):**	7.78e-43
Kurtosis:	2.070	**Cond. No.**	2.46

OLS Regression Result: Logit with Caliper = 0.005

Results and Conclusion

First, let's state the Null and Alternative Hypothesis for this problem:

Null Hypothesis: Premium and Non-Premium users skip the same number of songs on an average

Alternative Hypothesis: Premium and Non-Premium users skip different number of songs on an average

The results of our OLS Regression with and without matching can be summarized as below:

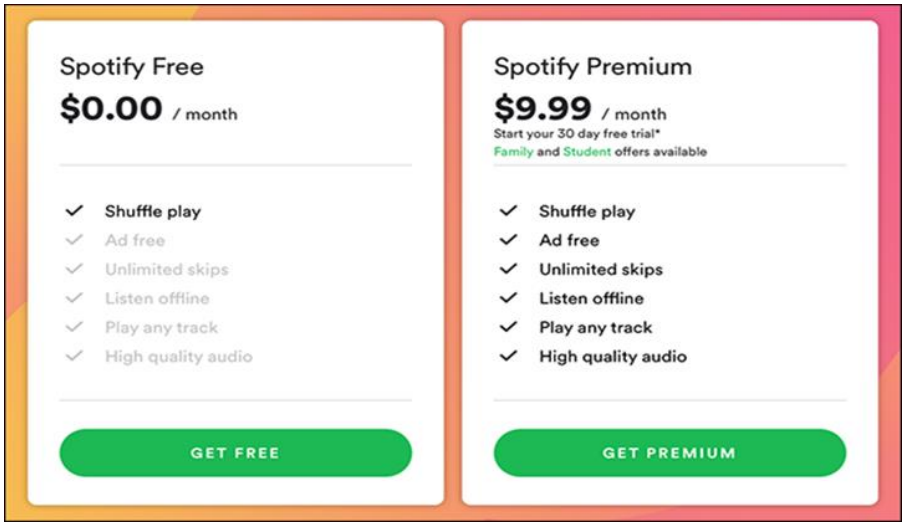
	Coefficient	**P-value**
With Matching	-0.505	0.004
Without Matching	0.0748	0.558

It can be observed that with matching our result is Statistically Significant (0.004 p-value), while the result is not Statistically Significant without matching (0.558 p-value.)

Interpretation: With matching, the coefficient observed is -0.505 which implies that users with Spotify Premium skip ~0.5 less songs compared to users without Premium on an average.

Hence, we can reject our Null Hypothesis and conclude that users with Spotify Premium users skip less songs compared to users without Premium. This is an interesting revelation because Premium users are offered unlimited skips on Spotify while Non-Premium users only get a limited number of skips.

Business Recommendation



Spotify Premium Flyer

Song recommendations for Spotify Premium users seem to work well as they skip less often than on average even though they have unlimited skips. Spotify can implement the recommendation engine and more features from Spotify Premium to Spotify Free if they want to increase their market share of free users. Also, they should remove 'Unlimited Skips' as an exclusive feature for Spotify Premium and provide it to all users to improve ratings and time spent on app by free users, which would increase their ad revenue.

Threats to Causal Inference

1) Omitted Variable Bias: We did not have user-level features such as age, gender, income and location to include in our analysis. This definitely limits the extent of our causal analysis and is unfortunately not something that we can solve with PSM. One of the assumptions of PSM is that we observe all relevant variables and in our case we may not be able to do that. However, the structure of our analysis can be reapplied with extra data and data-features which can help us deal with this weakness.

2) Sampling Error: The original dataset provided to us by Spotify is over 50GB so we had to use a 'mini-version' with 170K rows. However, this may not be a 'representative sample' of a typical Spotify session and may not generalize well to the population of Spotify users. We can re-run our analysis by randomly

sampling data or run it with vastly more data available to evaluate how well our results can be generalized.

3) Measurement Error: The original data set contained three 'skip' variables that corresponded to when the user skipped the song (in the first third, the second third or the last third). During our data-cleaning process we utilized 'skip_2' as our outcome variable as we felt this was an 'average' skip for a user. If a user skips too early it may be that the user did it by accident or is not listening to the song before skipping it (the reason for skip does not correlate to any observed factor). If the skip is late in the song it may be that the user is skipping only the last few seconds of the song and actually listened through the song and liked it. This introduces a 'measurement error' into our analysis about what we consider a 'skipped' song and how we evaluate the 'skip' variables in the data to define our target variable.

Limitations

1) Not Including Track Level Features: One of the main limitations of Propensity Score Matching is that it assumes all confounding variables are observed in the model. For this project, we are using only Session Level data for matching and not including Track Level features like acoustics, danceability, loudness, popularity etc. that can have an impact on the track being skipped but at the same time, make our matches more sparse and non intuitive.

2) Not matching on User Level: Matching on the users would have made sense here too as user demographics, psychographics and behavior can also have an effect on songs being skipped. However, since the database lacked this data, we chose to match on the session level.

3) Exploring more Propensity Functions and Matching Algorithms: For this project, we used only the Propensity Logit algorithm for matching. To evaluate the further robustness of our results, we could have tried different propensity functions like Probit and even other algorithms like Coarsened Exact Matching to check if it yields similar results as we did.

Sources

1. https://www.theverge.com/2022/1/20/22892939/music-streaming-services-market-share-q2-2021-spotify-apple-amazon-tencent-youtube#:~:text=Spotify%20was%20the%20largest%20music,third%20with%2013%20percent%20apiece.
2. https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge
3. https://www.appier.com/blog/what-is-a-recommendation-engine-and-how-does-it-work

Appendix

Setting Up

In [28]:

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", f
orce_remount=True).
```

In [29]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Preparation

In [31]:

```
# Load in the training set
training_set = pd.read_csv("log_mini.csv")
training_set.hist_user_behavior_reason_start = training_set.hist_user_behavior_reason_start.astype('category')
training_set.hist_user_behavior_reason_end = training_set.hist_user_behavior_reason_end.astype('category')
training_set.context_type = training_set.context_type.astype('category')
training_set.date = training_set.date.apply(pd.to_datetime)

# Using skip_2 as the ground truth
training_set['skipped'] = (training_set.skip_2 | training_set.skip_1).astype('int32')
training_set = training_set.drop(columns=['skip_1','skip_2','skip_3','not_skipped'])
training_set
```

Out[31]:

	session_id	session_position	session_length	track_id_clean	context_switch	no_pause_before_play	short_pause_before_play	long_paus
0	0_00006f66-33e5-4de7-a324-2d18e439fc1e	1	20	t_0479f24c-27d2-46d6-a00c-7ec928f2b539	0	0	0	
1	0_00006f66-33e5-4de7-a324-2d18e439fc1e	2	20	t_9099cd7b-c238-47b7-9381-f23f2c1d1043	0	1	0	
2	0_00006f66-33e5-4de7-a324-2d18e439fc1e	3	20	t_fc5df5ba-5396-49a7-8b29-35d0d28249e0	0	1	0	
3	0_00006f66-33e5-4de7-a324-2d18e439fc1e	4	20	t_23cff8d6-d874-4b20-83dc-94e450e8aa20	0	1	0	
4	0_00006f66-33e5-4de7-a324-2d18e439fc1e	5	20	t_64f3743c-f624-46bb-a579-0f3f9a07a123	0	1	0	
...	...	...	...	...	...	...	...	
167875	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	16	20	t_360910e8-2a84-42b0-baf1-59abcf96a1f2	0	1	0	
167876	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	17	20	t_aa2fff77-9b0a-4fa3-a685-ecef50310e8a	0	1	0	
167877	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	18	20	t_f673e1b7-4ebe-4fc1-ac24-a9f25de70381	0	1	0	
167878	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	19	20	t_e172e8e7-7161-42a9-acb0-d606346c8f87	0	1	0	
167879	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	20	20	t_77977dd6-597e-4425-8f8f-4efb32ecfba6	0	1	0	

167880 rows × 18 columns

Propensity Score Matching

In [32]:

```
# each session can have different context_type and context_switch means if the user change the playlist in the
training_set[training_set["session_id"]=="0_000641b5-a2ce-4803-bdeb-499c548ba50d"][["session_id","context_type'
```

Out[32]:

```
session_id                              context_type
0_000641b5-a2ce-4803-bdeb-499c548ba50d  catalog                  10
                                        charts                    0
                                        editorial_playlist        0
                                        personalized_playlist     0
                                        radio                    10
                                        user_collection           0
dtype: int64
```

In [33]:

```
session=training_set.groupby("session_id").agg(
     track_num=pd.NamedAgg(column="track_id_clean",aggfunc="count"),
     if_switch_playlist=pd.NamedAgg(column="context_switch",aggfunc=max),
     no_pause_number=pd.NamedAgg(column="no_pause_before_play",aggfunc=sum),
     short_pause_number=pd.NamedAgg(column="short_pause_before_play",aggfunc=sum),
     long_pause_number=pd.NamedAgg(column="long_pause_before_play",aggfunc=sum),
     hist_user_behavior_n_seekfwd=pd.NamedAgg(column="hist_user_behavior_n_seekfwd",aggfunc=sum),
     hist_user_behavior_n_seekback=pd.NamedAgg(column="hist_user_behavior_n_seekback",aggfunc=sum),
     hour_of_day=pd.NamedAgg(column="hour_of_day",aggfunc=max),
     premium=pd.NamedAgg(column="premium",aggfunc=max),
     skipped=pd.NamedAgg(column="skipped",aggfunc=sum)
).reset_index()
session.head()
```

Out[33]:

	session_id	track_num	if_switch_playlist	no_pause_number	short_pause_number	long_pause_number	hist_user_behavior_n_seekfwd	hist_u
0	0_00006f66-33e5-4de7-a324-2d18e439fc1e	20	0	18	1	1	0	
1	0_0000a72b-09ac-412f-b452-9b9e79bded8f	20	0	16	3	3	0	
2	0_00010fc5-b79e-4cdf-bc4c-f140d0f99a3a	20	0	14	3	5	0	
3	0_00016a3d-9076-4f67-918f-f29e3ce160dc	20	0	7	12	12	0	
4	0_00018b58-deb8-4f98-ac5e-d7e01b346130	11	1	9	1	1	0	

In [34]:

```
session
```

Out[34]:

	session_id	track_num	if_switch_playlist	no_pause_number	short_pause_number	long_pause_number	hist_user_behavior_n_seekfwd	hi
0	0_00006f66-33e5-4de7-a324-2d18e439fc1e	20	0	18	1	1	0	
1	0_0000a72b-09ac-412f-b452-9b9e79bded8f	20	0	16	3	3	0	
2	0_00010fc5-b79e-4cdf-bc4c-f140d0f99a3a	20	0	14	3	5	0	
3	0_00016a3d-9076-4f67-918f-f29e3ce160dc	20	0	7	12	12	0	
4	0_00018b58-deb8-4f98-ac5e-d7e01b346130	11	1	9	1	1	0	
...	...	...	...	...	...	...	...	
9995	0_0eac164c-f209-4590-8608-a56e67658952	20	0	18	1	1	0	
9996	0_0eacbee7-9868-48a0-9ab0-f86069329f50	20	0	16	3	3	0	
9997	0_0ead11fc-f32c-4eb4-8fc1-15b51432a404	20	0	11	8	8	0	
9998	0_0eae096d-ae4c-4556-8227-0f6294958330	12	0	9	1	2	0	
9999	0_0eaeef5d-25e9-4429-bd55-af15d3604c9f	20	0	17	1	2	0	

10000 rows × 11 columns

In [35]:

```
!pip install psmpy
from psmpy import PsmPy
from psmpy.functions import cohenD
from psmpy.plotting import *
```

```
Collecting psmpy
  Downloading psmpy-0.2.8-py3-none-any.whl (11 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from psmpy) (3.2.2)
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (from psmpy) (0.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from psmpy) (0.11.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from psmpy) (1.21.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from psmpy) (1.3.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from psmpy) (1.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packag
es (from matplotlib->psmpy) (3.0.8)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib-
>psmpy) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->psmpy)
(0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->ps
mpy) (1.4.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.
0.1->matplotlib->psmpy) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->m
atplotlib->psmpy) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->psmpy) (202
2.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from sklearn->psmpy) (1.
0.2)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->sklea
rn->psmpy) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-lear
n->sklearn->psmpy) (3.1.0)
Installing collected packages: psmpy
Successfully installed psmpy-0.2.8
```

In [36]:

```
psm = PsmPy(session, treatment='premium', indx='session_id', exclude = ["skipped"])
```

In [37]:

```
psm.logistic_ps(balance = True)
```

In [ ]:

```
psm.predicted_data.head()
```

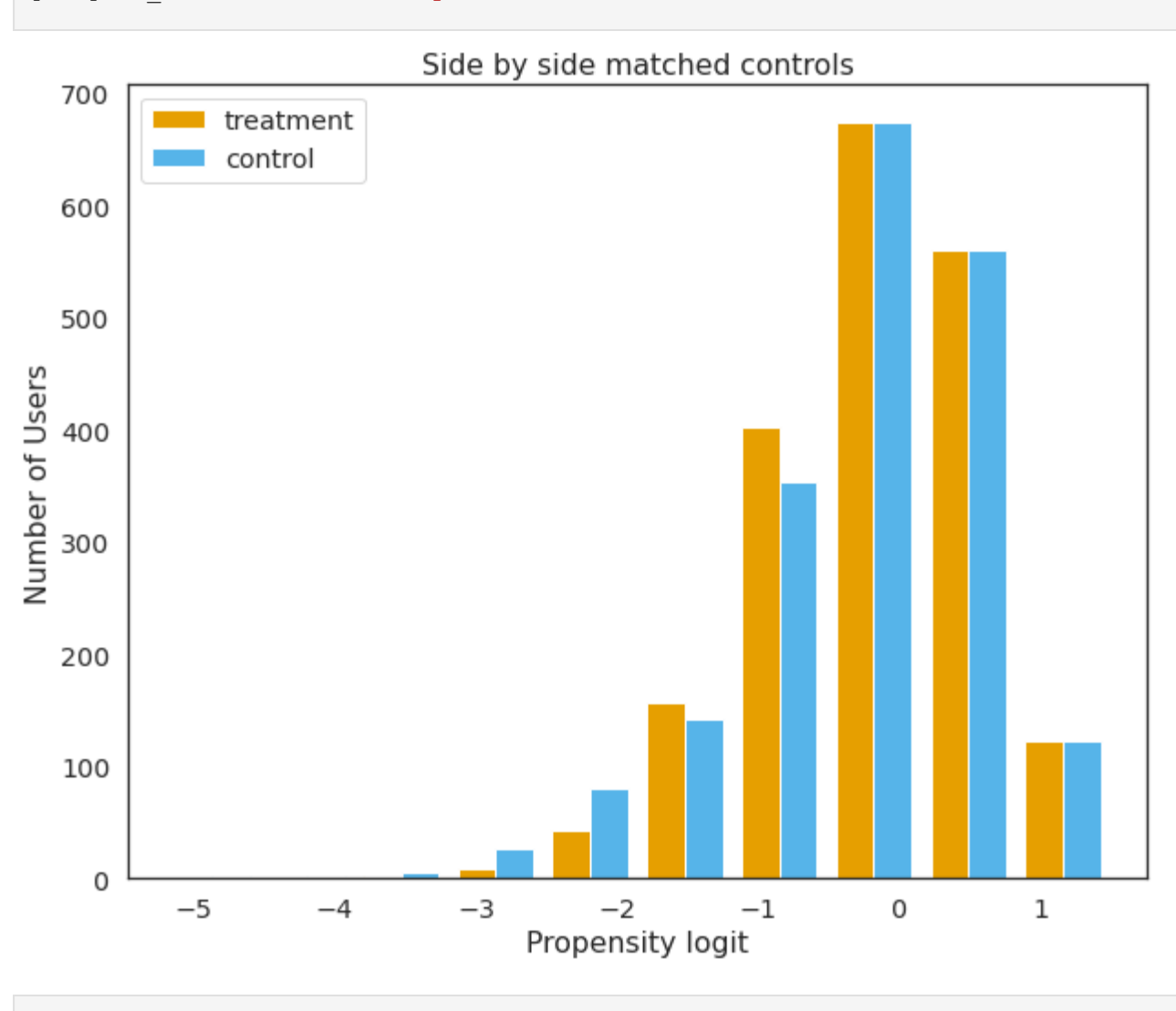
In [38]:

```
# use 1:1 KNN matching method, and no restriction to the matching neighbours
psm.knn_matched(matcher='propensity_logit', replacement=False)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but NearestNeighb
ors was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

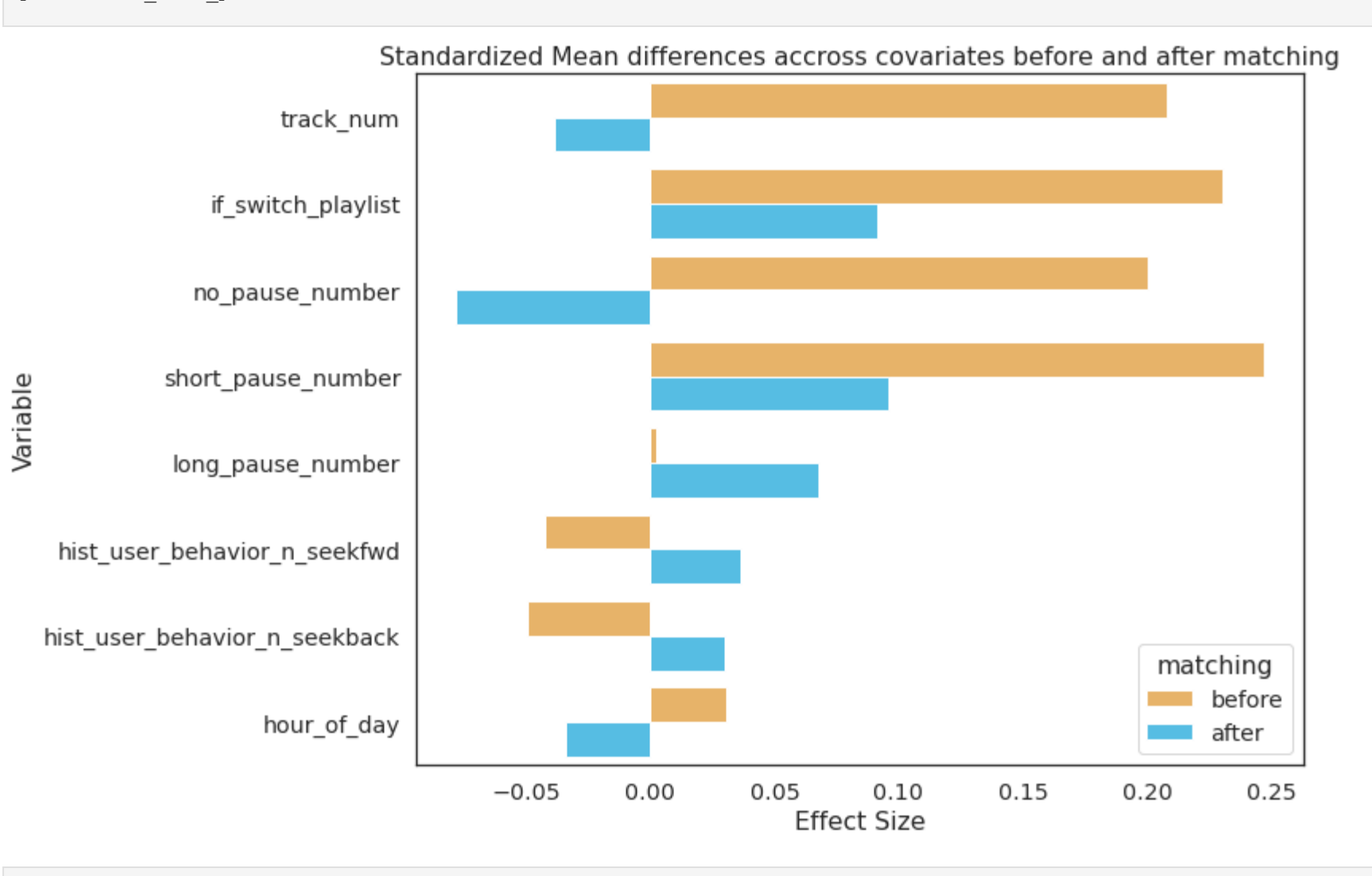
In [39]:

```
import seaborn as sns
sns.set(rc={'figure.figsize':(10,8)}, font_scale = 1.3)
psm.plot_match(Title='Side by side matched controls', Ylabel='Number of Users', Xlabel= 'Propensity logit', nar
```



In [40]:

```
psm.effect_size_plot(save=False)
```



In [41]:

```
psm.matched_ids
```

Out[41]:

	session_id	matched_ID
0	0_00030fe3-298d-4930-8534-fc5a9b82971b	0_09617d9c-2bac-4367-9702-689a8e15009b
1	0_000822a0-d9a2-4346-9fef-2b5d68e12bc2	0_01365638-6cba-4a6f-9188-128b93ff5593
2	0_0009bdeb-b5b7-4073-9b01-b72a32c77cda	0_037a0c8c-e0b1-4102-a8a6-c37d40d608a9
3	0_000a6cae-aef1-4a3e-b1bd-d1971edfcb41	0_0aea200a-57fb-4a62-8c55-39d2e352c54b
4	0_000cec57-79f9-4acf-87a1-9bb9063d012d	0_0a5da48b-0266-4e4a-ae10-41173da4e5e4
...	...	...
1965	0_0e9fc88b-f1d3-4d43-b17c-931c53e9b652	0_00fd371a-ee00-40e7-bcee-00fb67ecac89
1966	0_0ea14f73-464c-4ef5-86fe-8d6a842a031b	0_063822d1-e253-4178-a833-eea43aa4aadf
1967	0_0ea2015c-84ff-465e-871d-5be6cce7dc49	0_0bdd1377-1d09-4f69-993e-81110440b52f
1968	0_0ea75a3d-9151-444e-a7d2-df2b97d89e05	0_09c33f11-8fc4-413e-a2b4-63c8f0f5ddeb
1969	0_0ea956f2-3424-4b7b-8812-1da3344f077f	0_04e49d8b-e573-482e-8b74-f5fe92379b97

1970 rows × 2 columns

In [42]:

```
psm.effect_size
```

Out[42]:

	Variable	matching	Effect Size
0	track_num	before	0.208081
1	track_num	after	-0.038817
2	if_switch_playlist	before	0.230162
3	if_switch_playlist	after	0.091153
4	no_pause_number	before	0.200273
5	no_pause_number	after	-0.078122
6	short_pause_number	before	0.246944
7	short_pause_number	after	0.095726
8	long_pause_number	before	0.002231
9	long_pause_number	after	0.067736
10	hist_user_behavior_n_seekfwd	before	-0.042352
11	hist_user_behavior_n_seekfwd	after	0.036070
12	hist_user_behavior_n_seekback	before	-0.049656
13	hist_user_behavior_n_seekback	after	0.029953
14	hour_of_day	before	0.030624
15	hour_of_day	after	-0.034469

In [43]:

```
untreated_matched_data = session[session['session_id'].isin(psm.matched_ids['session_id']) == True]
```

In [44]:

```
treated_matched_data = session[session['session_id'].isin(psm.matched_ids['matched_ID']) == True]
```

In [45]:

```
all_matched = pd.concat([treated_matched_data, untreated_matched_data])
```

In [46]:

```
all_matched
```

Out[46]:

	session_id	track_num	if_switch_playlist	no_pause_number	short_pause_number	long_pause_number	hist_user_behavior_n_seekfwd	hi
2	0_00010fc5-b79e-4cdf-bc4c-f140d0f99a3a	20	0	14	3	5	0	
5	0_00027db1-ffe5-4b00-8eeb-8d8188451298	15	1	9	3	5	0	
9	0_0003ac37-c70a-47a6-b9dc-b8b163826082	20	0	13	5	6	0	
28	0_0008bd24-bbb0-4747-b777-7c97ce5ab3a9	20	1	12	7	7	0	
30	0_00094b63-0ef1-4cec-94e5-30879a3cce28	20	0	13	3	6	0	
...	...	...	...	...	...	...	...	
9967	0_0e9fc88b-f1d3-4d43-b17c-931c53e9b652	10	0	9	0	0	5	
9971	0_0ea14f73-464c-4ef5-86fe-8d6a842a031b	12	0	6	5	5	0	
9972	0_0ea2015c-84ff-465e-871d-5be6cce7dc49	10	0	8	0	1	0	
9985	0_0ea75a3d-9151-444e-a7d2-df2b97d89e05	20	0	11	7	8	0	
9988	0_0ea956f2-3424-4b7b-8812-1da3344f077f	20	0	15	4	4	0	

3940 rows × 11 columns

In [47]:

```
overview = all_matched[['skipped','premium']].groupby(by = ['premium']).aggregate([np.mean, np.var, np.std, 'cc
print(overview)
```

```
          skipped
             mean        var       std count
premium
False    8.622335  25.730329  5.072507  1970
True     8.035025  23.155197  4.811985  1970
```

In [48]:

```
treated_outcome = overview['skipped']['mean'][1]
treated_counterfactual_outcome = overview['skipped']['mean'][0]
```

In [49]:

```
ate = (treated_outcome - treated_counterfactual_outcome)
print('The Average Treatment Effect (ATE): {:.4f}'.format(ate))
```

```
The Average Treatment Effect (ATE): -0.5873
```

In [50]:

```
all_matched
```

Out[50]:

	session_id	track_num	if_switch_playlist	no_pause_number	short_pause_number	long_pause_number	hist_user_behavior_n_seekfwd	hi
2	0_00010fc5-b79e-4cdf-bc4c-f140d0f99a3a	20	0	14	3	5	0	
5	0_00027db1-ffe5-4b00-8eeb-8d8188451298	15	1	9	3	5	0	
9	0_0003ac37-c70a-47a6-b9dc-b8b163826082	20	0	13	5	6	0	
28	0_0008bd24-bbb0-4747-b777-7c97ce5ab3a9	20	1	12	7	7	0	
30	0_00094b63-0ef1-4cec-94e5-30879a3cce28	20	0	13	3	6	0	
...	...	...	...	...	...	...	...	
9967	0_0e9fc88b-f1d3-4d43-b17c-931c53e9b652	10	0	9	0	0	5	
9971	0_0ea14f73-464c-4ef5-86fe-8d6a842a031b	12	0	6	5	5	0	
9972	0_0ea2015c-84ff-465e-871d-5be6cce7dc49	10	0	8	0	1	0	
9985	0_0ea75a3d-9151-444e-a7d2-df2b97d89e05	20	0	11	7	8	0	
9988	0_0ea956f2-3424-4b7b-8812-1da3344f077f	20	0	15	4	4	0	

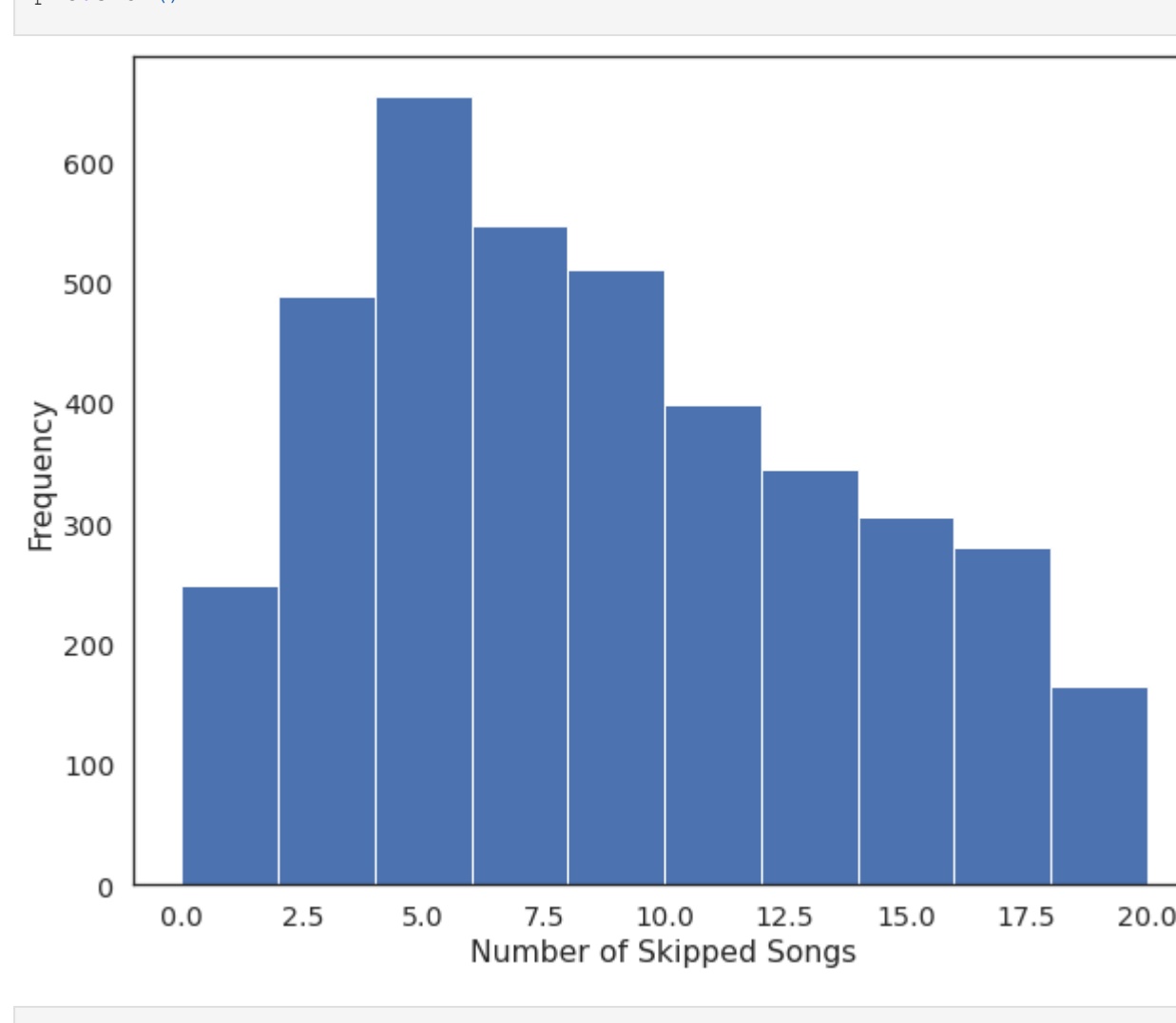
3940 rows × 11 columns

In [51]:

```
from statsmodels.formula.api import ols
```

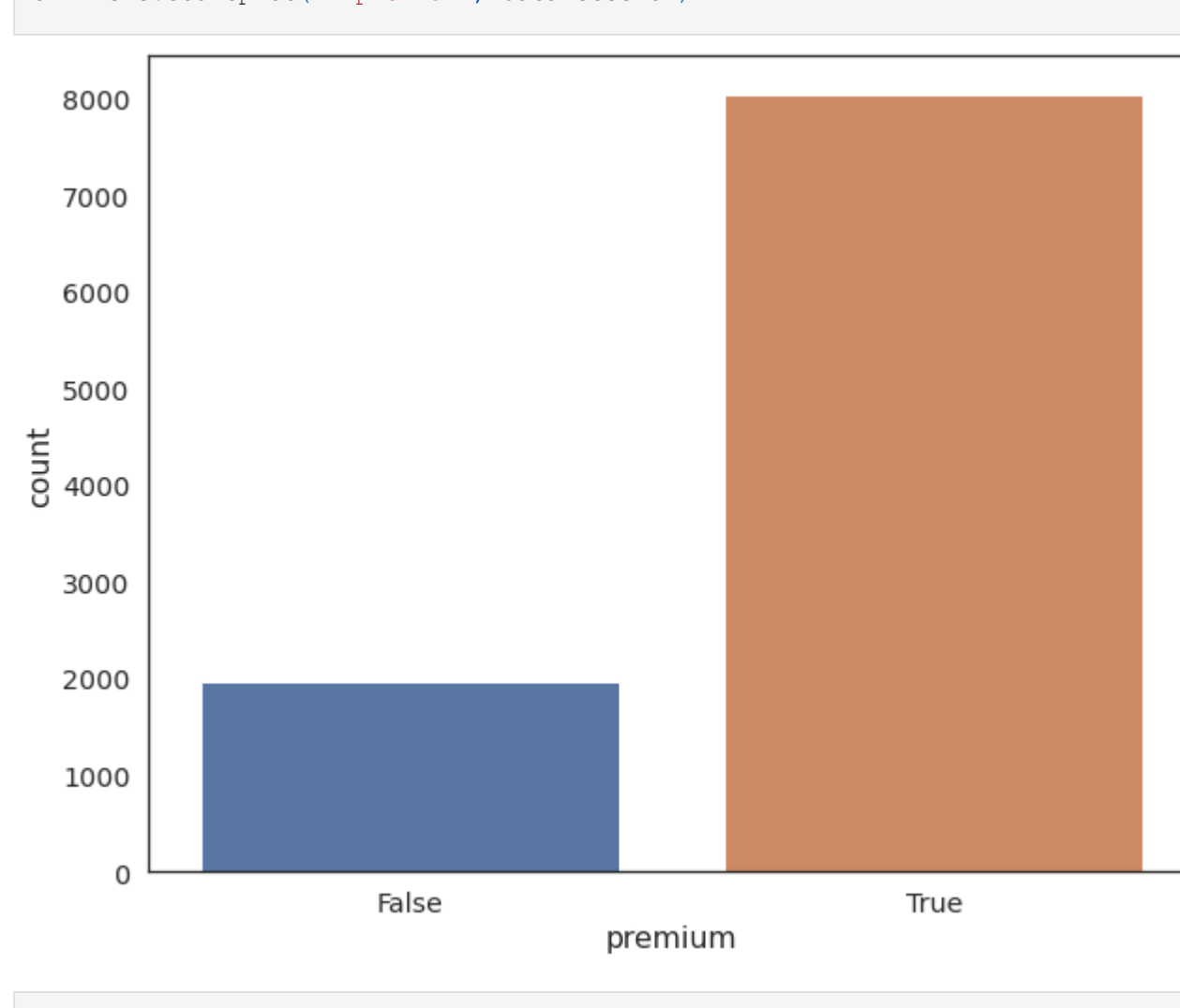
In [52]:

```
x = all_matched['skipped']
bins = np.linspace(0, 20, 11)
plt.hist(x, bins)
plt.xlabel('Number of Skipped Songs')
plt.ylabel('Frequency')
plt.show()
```



In [53]:

```
ax = sns.countplot(x="premium", data=session)
```



In [54]:

```
#effect with matching
fit = ols('skipped ~ C(premium)', data=all_matched).fit()
fit.summary()
```

Out[54]:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.004
Model:	OLS	**Adj. R-squared:**	0.003
Method:	Least Squares	**F-statistic:**	13.90
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.000195
Time:	23:14:11	**Log-Likelihood:**	-11886.
No. Observations:	3940	**AIC:**	2.378e+04
Df Residuals:	3938	**BIC:**	2.379e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.111	77.407	0.000	8.404	8.841
C(premium)[T.True]	-0.5873	0.158	-3.728	0.000	-0.896	-0.278

Omnibus:	623.019	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	227.507
Skew:	0.379	**Prob(JB):**	3.96e-50
Kurtosis:	2.100	**Cond. No.**	2.62

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

In [55]:

```
# effect without matching
fit = ols('skipped ~ C(premium)', data=session).fit()
fit.summary()
```

Out[55]:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.3437
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.558
Time:	23:14:11	**Log-Likelihood:**	-30431.
No. Observations:	10000	**AIC:**	6.087e+04
Df Residuals:	9998	**BIC:**	6.088e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.114	75.417	0.000	8.398	8.846
C(premium)[T.True]	0.0748	0.128	0.586	0.558	-0.175	0.325

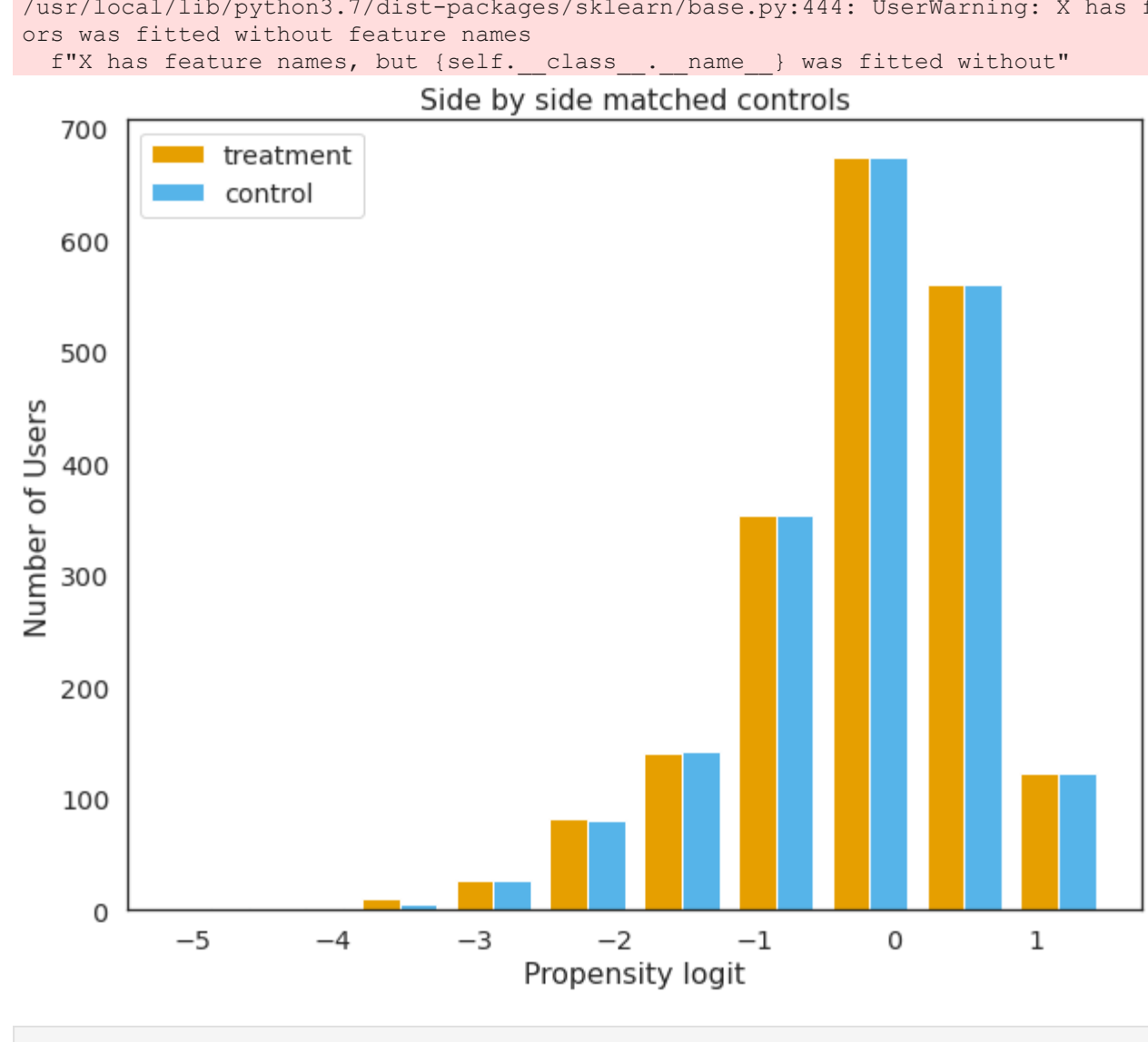
Omnibus:	2832.459	**Durbin-Watson:**	1.971
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	568.523
Skew:	0.274	**Prob(JB):**	3.52e-124
Kurtosis:	1.969	**Cond. No.**	4.30

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

In [57]:

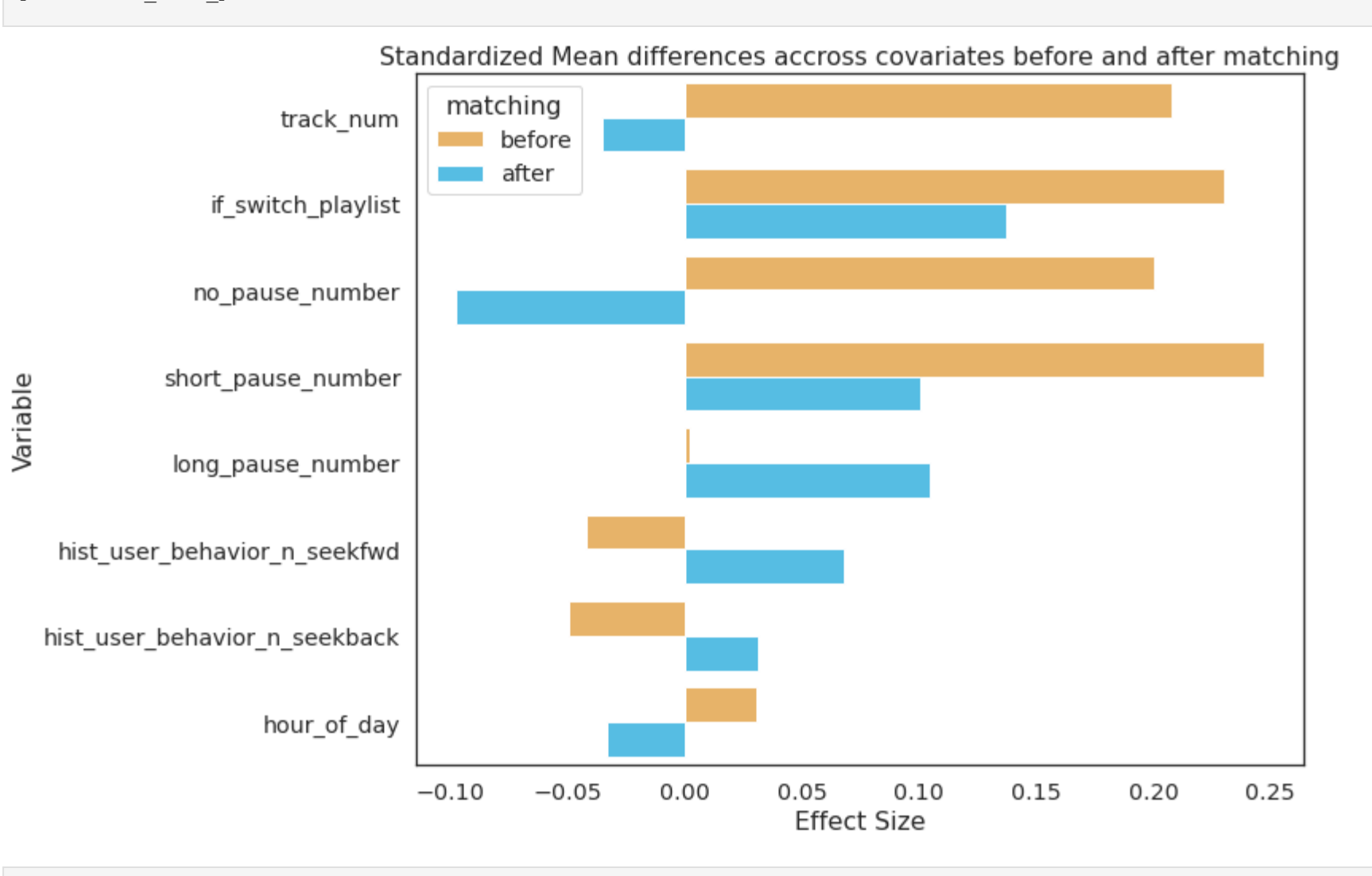
```
# sensitivity analysis
# change the matching method to with replacement
psm.knn_matched(matcher='propensity_logit', replacement=True)
sns.set(rc={'figure.figsize':(10,8)}, font_scale = 1.3)
psm.plot_match(Title='Side by side matched controls', Ylabel='Number of Users', Xlabel= 'Propensity logit', nar
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but NearestNeighb
ors was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```



In [58]:

```
psm.effect_size_plot(save=False)
```



In [59]:

```
untreated_matched_data = session[session['session_id'].isin(psm.matched_ids['session_id']) == True]
```

In [60]:

```
treated_matched_data = session[session['session_id'].isin(psm.matched_ids['matched_ID']) == True]
```

In [61]:

```
all_matched = pd.concat([treated_matched_data, untreated_matched_data])
```

In [62]:

```
fit = ols('skipped ~ C(premium)', data=all_matched).fit()
fit.summary()
```

Out[62]:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.002
Model:	OLS	**Adj. R-squared:**	0.002
Method:	Least Squares	**F-statistic:**	8.396
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.00378
Time:	23:34:46	**Log-Likelihood:**	-10162.
No. Observations:	3361	**AIC:**	2.033e+04
Df Residuals:	3359	**BIC:**	2.034e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.112	76.881	0.000	8.402	8.842
C(premium)[T.True]	-0.5052	0.174	-2.898	0.004	-0.847	-0.163

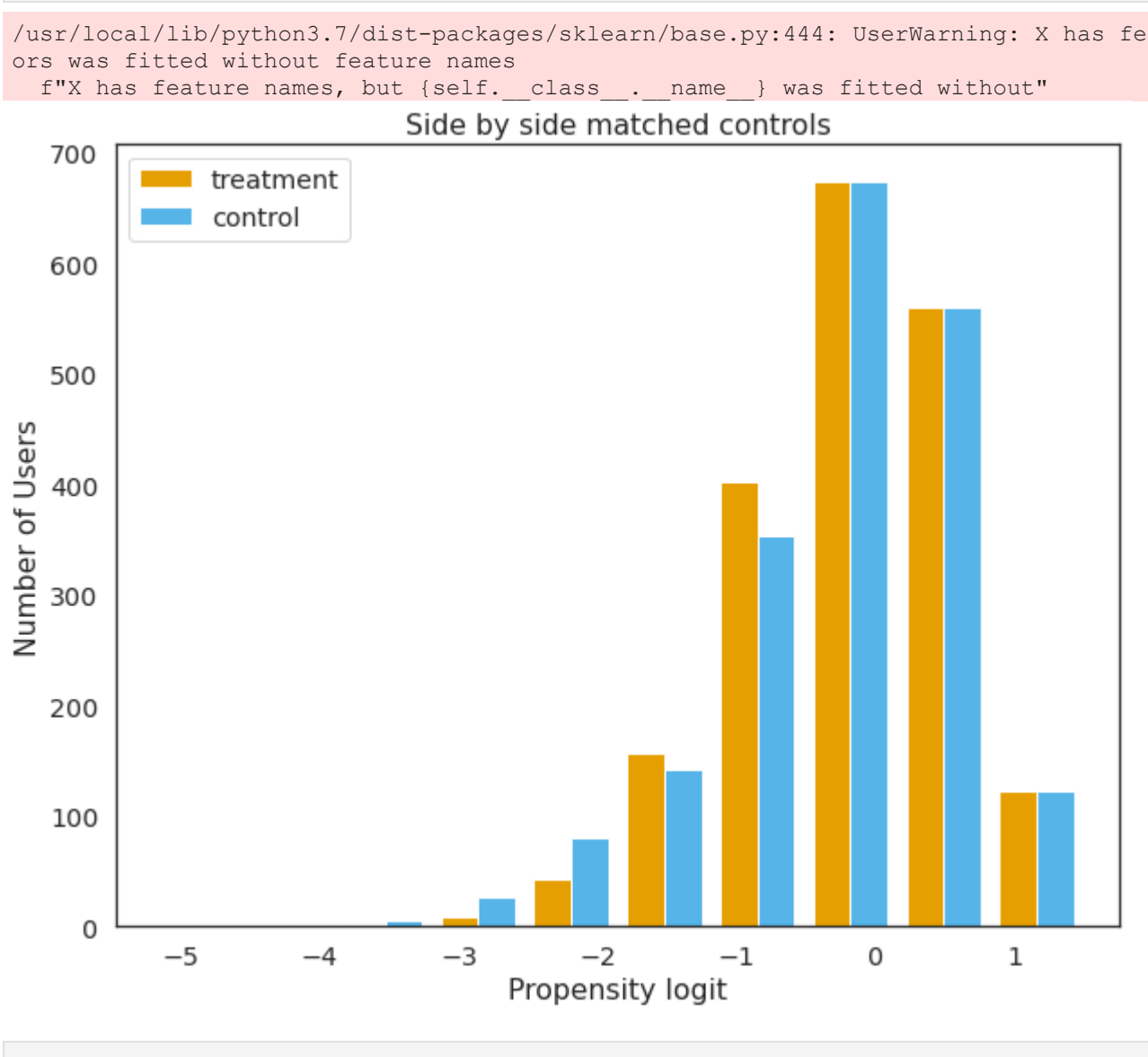
Omnibus:	595.010	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	193.918
Skew:	0.361	**Prob(JB):**	7.78e-43
Kurtosis:	2.070	**Cond. No.**	2.46

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

In [63]:

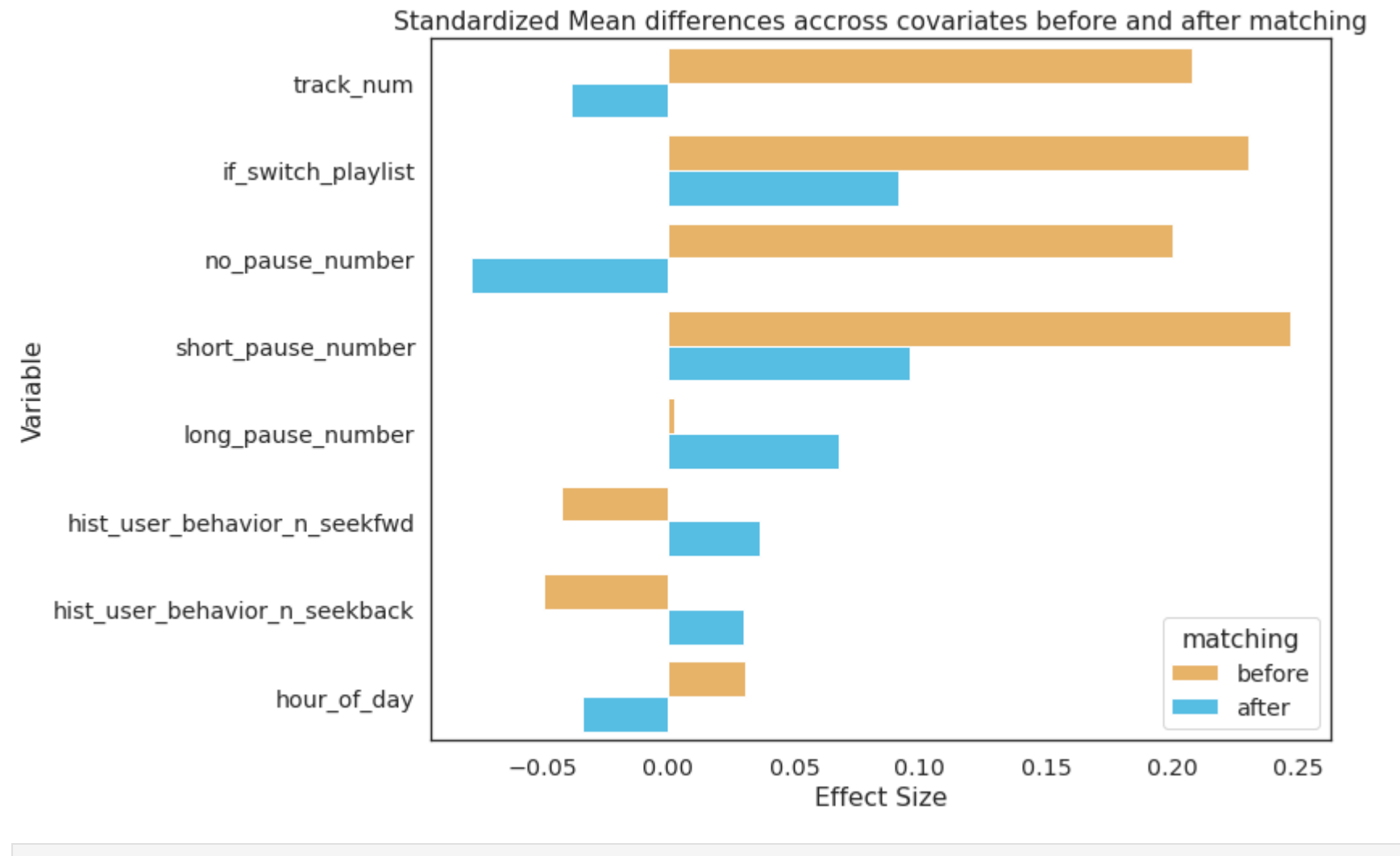
```
# change to match based on propensity score
psm.knn_matched(matcher='propensity_score', replacement=False)
sns.set(rc={'figure.figsize':(10,8)}, font_scale = 1.3)
psm.plot_match(Title='Side by side matched controls', Ylabel='Number of Users', Xlabel= 'Propensity logit', nar
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but NearestNeighb
ors was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```



In [64]:

```
psm.effect_size_plot(save=False)
```



In [65]:
```python
untreated_matched_data = session[session['session_id'].isin(psm.matched_ids['session_id']) == True]
```

In [66]:
```python
treated_matched_data = session[session['session_id'].isin(psm.matched_ids['matched_ID']) == True]
```

In [67]:
```python
all_matched = pd.concat([treated_matched_data, untreated_matched_data])
```

In [68]:
```python
fit = ols('skipped ~ C(premium)', data=all_matched).fit()
fit.summary()
```

Out[68]:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.004
Model:	OLS	**Adj. R-squared:**	0.004
Method:	Least Squares	**F-statistic:**	14.97
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.000111
Time:	23:39:35	**Log-Likelihood:**	-11874.
No. Observations:	3940	**AIC:**	2.375e+04
Df Residuals:	3938	**BIC:**	2.376e+04
Df Model:	1		
Covariance Type:	nonrobust		

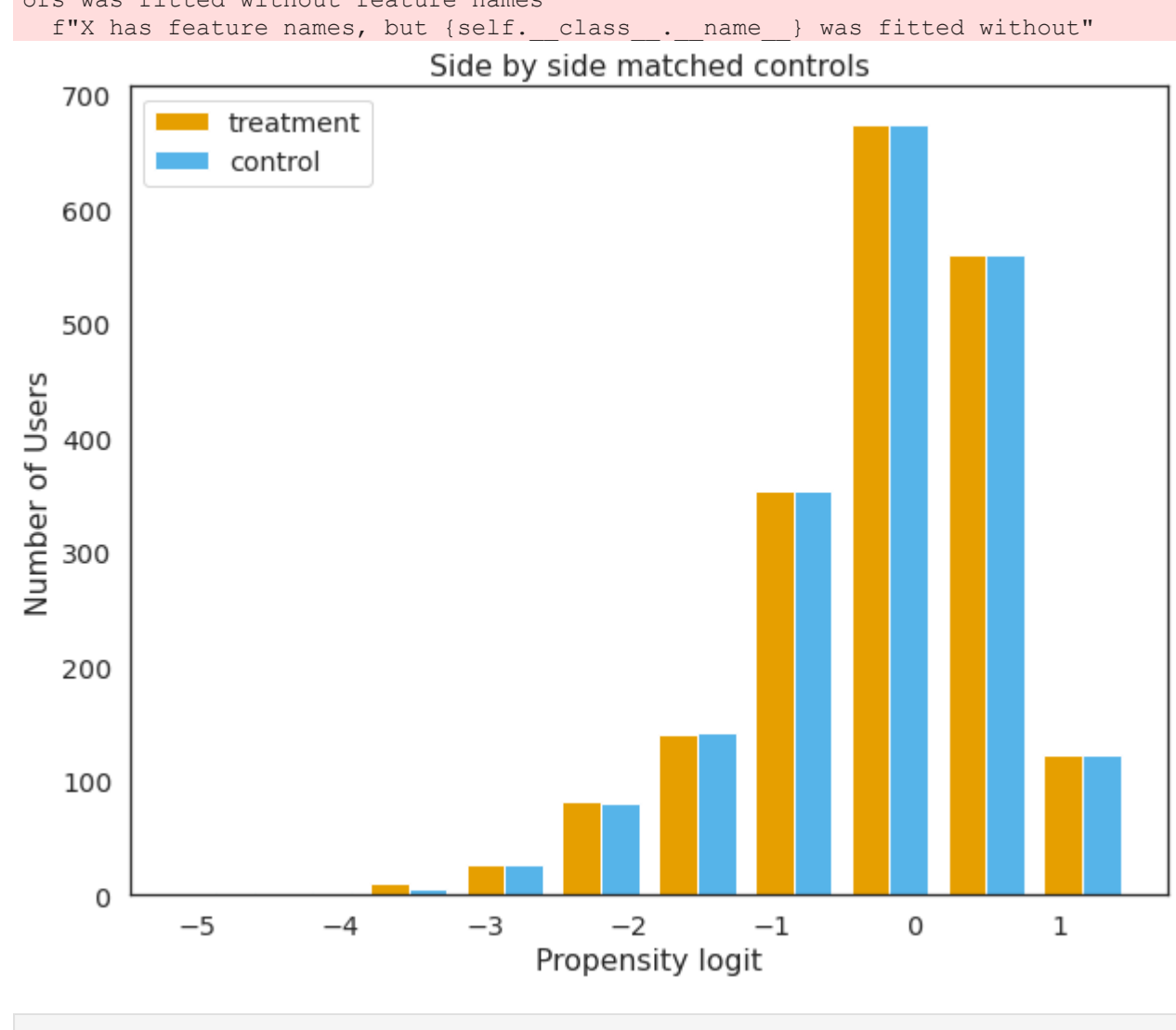
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.111	77.647	0.000	8.405	8.840
C(premium)[T.True]	-0.6076	0.157	-3.869	0.000	-0.916	-0.300

Omnibus:	606.994	**Durbin-Watson:**	1.951
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	224.719
Skew:	0.377	**Prob(JB):**	1.60e-49
Kurtosis:	2.106	**Cond. No.**	2.62

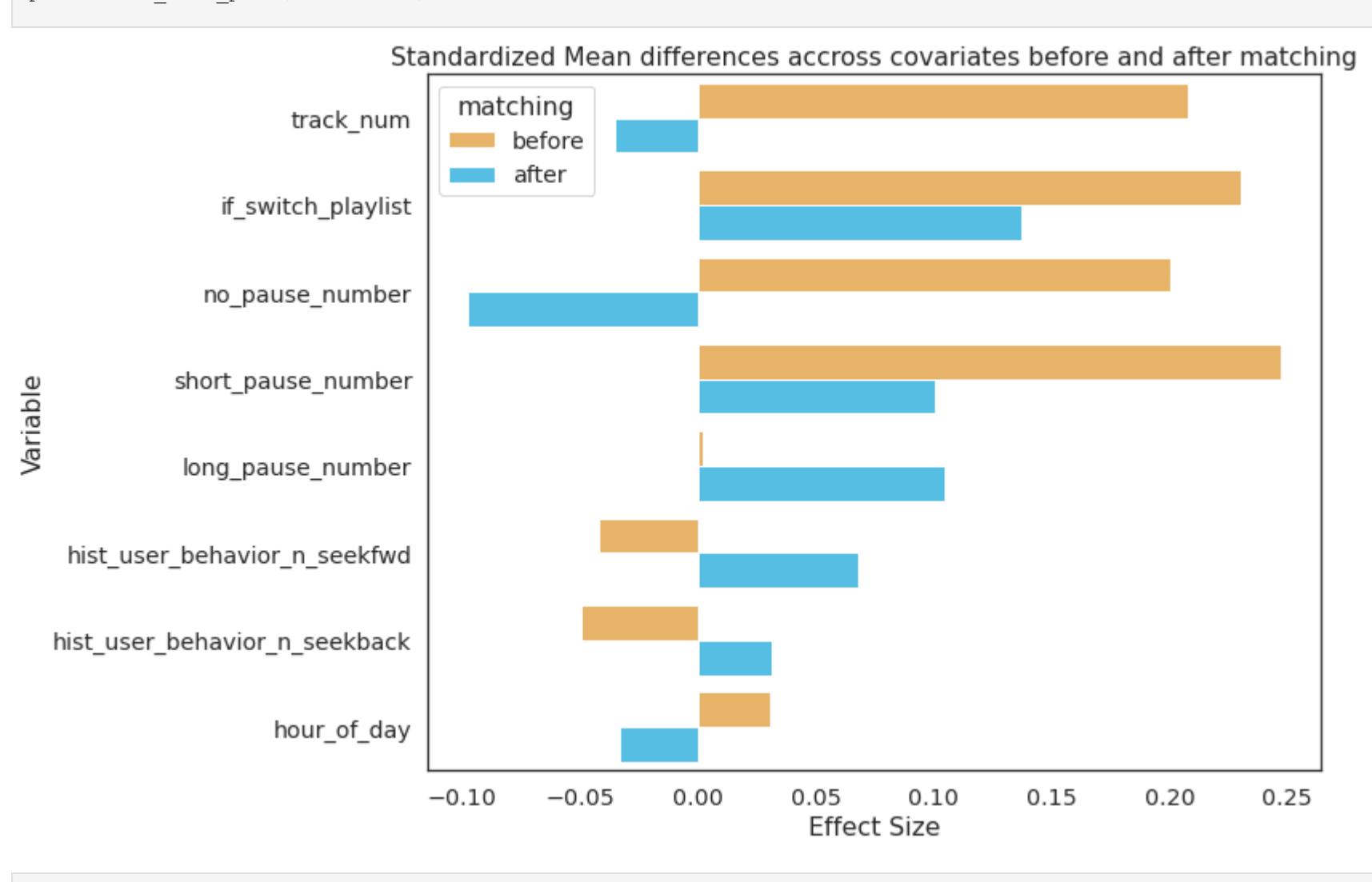
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [72]:
```python
# change to with caliper=0.005
psm.knn_matched(matcher='propensity_logit', replacement=True, caliper=0.005)
sns.set(rc={'figure.figsize':(10,8)}, font_scale = 1.3)
psm.plot_match(Title='Side by side matched controls', Ylabel='Number of Users', Xlabel= 'Propensity logit', nam
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but NearestNeighb
ors was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```



In [73]:
```python
psm.effect_size_plot(save=False)
```



In [74]:
```python
untreated_matched_data = session[session['session_id'].isin(psm.matched_ids['session_id']) == True]
```

In [75]:
```python
treated_matched_data = session[session['session_id'].isin(psm.matched_ids['matched_ID']) == True]
```

In [76]:
```python
all_matched = pd.concat([treated_matched_data, untreated_matched_data])
```

In [77]:
```python
fit = ols('skipped ~ C(premium)', data=all_matched).fit()
fit.summary()
```

Out[77]:

OLS Regression Results

Dep. Variable:	skipped	**R-squared:**	0.002
Model:	OLS	**Adj. R-squared:**	0.002
Method:	Least Squares	**F-statistic:**	8.396
Date:	Sat, 07 May 2022	**Prob (F-statistic):**	0.00378
Time:	23:42:54	**Log-Likelihood:**	-10162.
No. Observations:	3361	**AIC:**	2.033e+04
Df Residuals:	3359	**BIC:**	2.034e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.6223	0.112	76.881	0.000	8.402	8.842
C(premium)[T.True]	-0.5052	0.174	-2.898	0.004	-0.847	-0.163

Omnibus:	595.010	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	193.918
Skew:	0.361	**Prob(JB):**	7.78e-43
Kurtosis:	2.070	**Cond. No.**	2.46

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