

In [1]:

```
# 导入所需要的函数模块
from marvel.AntPrint import ant_read_data, ant_write_data # 读取数据模块
from marvel.AntPrint import ant_print_all
from marvel.AntPrint import result_printer
from marvel.AntPrint.model import PSM
# 导入建模所需要的包

from scipy import stats
import seaborn as sns
import numpy as np
import pandas as pd
from scipy.stats import norm
import statsmodels.api as sm
from scipy.stats.mstats import winsorize
from linearmodels.iv.absorbing import AbsorbingLS, Interaction
from linearmodels import PanelOLS
from marvel.AntPrint import *
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
from statsmodels.formula.api import poisson
import patsy
from itertools import product
from sklearn.metrics import pairwise_distances
from statsmodels.stats.sandwich_covariance import cov_cluster
import itertools
```

Last executed in 6 s, finished in 21:06:04 2025-06-30

In [2]:

```
df = ant_read_data('workdata', mid_table = True)
```

Last executed in 12 s, finished in 21:06:16 2025-06-30

In [3]:

```
df['报告期1'] = df['报告期'].astype('str')
df['报告期1'] = pd.to_datetime(df['报告期1'])
df['year_month'] = df['报告期1'].apply(lambda x: x.strftime("%Y-%m"))

# 固定效应
df['type'] = df['period_length'].astype('category')
df['year_month_cat'] = df['year_month'].astype('category')
df['citycode_cat'] = df['citycode'].astype('category')
df['time_city'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'],
row['citycode_cat']), axis=1).astype('category')
df['time_type'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'],
row['type']), axis=1).astype('category')
df['用户id_rank'] = df['匿名化用户id_rank'].astype('category')

# 聚类层级
df['cluster_id'] = df['匿名化用户id_rank'].astype('category')
```

Last executed in 5 s, finished in 21:06:21 2025-06-30

In [4]:

```
regdata = df.copy()
```

Last executed in 0 s, finished in 21:06:21 2025-06-30

表1 描述性统计

In [5]:

```
ant_print_all(regdata[['new_use', 'period_length', '性别', '是否有医保', '是否有社保',
'年龄', 'n_module_consumpt', 'all_consumpt', 'online_consumpt',
'offline_consumpt', 'num_consumpt', '是否基金交易', '余额宝余额', '基金余额']],
model_method="describe")
```

describe

	new_use	period_length	性别	是否有医保	是否有社保 \
count	60060.000000	60060.000000	60060.000000	60060.000000	60060.000000
mean	0.739011	6.640293	0.400733	0.433333	0.332967
std	0.439178	5.084558	0.490051	0.495540	0.471279
min	0.000000	3.000000	0.000000	0.000000	0.000000
25%	0.000000	3.000000	0.000000	0.000000	0.000000
50%	1.000000	4.000000	0.000000	0.000000	0.000000
75%	1.000000	8.000000	1.000000	1.000000	1.000000
max	1.000000	21.000000	1.000000	1.000000	1.000000

	年龄	n_module_consumpt	all_consumpt	online_consumpt \
count	60060.000000	60060.000000	60060.000000	60060.000000
mean	37.105128	5.145022	3929.583921	868.356031
std	12.558085	3.652004	8664.305087	1852.267457
min	18.000000	0.000000	0.000000	0.000000
25%	28.000000	2.000000	176.922500	30.000000
50%	34.000000	5.000000	1010.210000	250.950000
75%	45.000000	8.000000	3338.412500	809.402500
max	87.000000	21.000000	58067.040000	12774.180000

	offline_consumpt	num_consumpt	是否基金交易	余额宝余额 \
count	60060.000000	60060.000000	60038.000000	60060.000000
mean	2911.814177	30.225108	0.234518	5013.346014
std	7609.723784	43.815044	0.423701	16153.969313
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	4.000000	0.000000	0.000000
50%	327.000000	16.000000	0.000000	16.010000
75%	1985.160000	41.000000	0.000000	1073.037500
max	52202.970000	2121.000000	1.000000	108925.310000

基金余额

count	60060.000000
mean	960.688443
std	5751.150588
min	0.000000
25%	0.000000
50%	0.000000
75%	0.100000
max	48361.160000

Last executed in 0 s, finished in 21:06:21 2025-06-30

附录一 ¶

表I 1 更换依赖程度的代理变量 ¶

In [6]:

```
interact = Interaction(pd.DataFrame(regdata["year_month_cat"]), regdata[
"lnavg_pv"])
lnty_list = ['lnt_module_consumpt', 'lnt_all_consumpt', 'lntnum_consumpt']

if enviroir()== '仿真环境':
    pass
else:
    cats = regdata[['用户id_rank', 'time_city']]
    cluster_id = regdata['cluster_id']
    x = regdata[["new_use"]]

    models = []
    results = []
    for i in lnty_list:
        y = regdata[i]
        model = AbsorbingLS(y, x, absorb=cats, interactions = interact)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names =lnty_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"], float_format='% .3f')
```

```
=====
              lnn_module_consumpt lnall_consumpt lnnum_consumpt
-----
new_use    0.035***              0.104**          0.049**
           (0.011)              (0.044)          (0.021)
rsquared    0.750                0.707            0.766
count      60060.000            60060.000        60060.000
=====
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****
```

Last executed in 3 s, finished in 21:06:25 2025-06-30

表15 对数转换的估计偏误 ¶

In [7]:

```
for i in ['all_consumpt', 'num_consumpt', 'n_module_consumpt']:
    regdata[i + '_dum'] = regdata[i].apply(lambda x: 1 if x > 0 else 0)
    regdata['log' + i] = np.where(regdata[i] > 0, np.log(regdata[i]), 0)
```

Last executed in 0 s, finished in 21:06:25 2025-06-30

In [8]:

```
# intensive
lnty_list = ['logn_module_consumpt', 'logall_consumpt', 'lognum_consumpt']
if regdata.shape[0]>200:
    cats = regdata[['用户id_rank', 'time_city', 'time_type']]
    cluster_id = regdata['cluster_id']
    x = regdata["new_use"]

    models = []
    results = []
    for i in lnty_list:
        y = regdata[i]
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names =lnty_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"], float_format='% .3f')
```

```
=====
              logn_module_consumpt logall_consumpt lognum_consumpt
-----
new_use    0.041***                0.145***                0.068***
           (0.012)                 (0.044)                 (0.022)
rsquared    0.747                  0.710                  0.765
count       60060.000              60060.000              60060.000
=====
```

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Last executed in 4 s, finished in 21:06:29 2025-06-30

In [9]:

```
# extensive
df_list = ['all_consumpt']
for i in df_list:
    regdata[i+"_dum"] = np.where(regdata[i]>0,1,0)

dum_list = ['all_consumpt_dum']
if regdata.shape[0]>200:
    cats = regdata[['用户id_rank','time_city','time_type']]
    cluster_id = regdata['cluster_id']
    x = regdata["new_use"]

    models = []
    results = []
    for i in dum_list:
        y = regdata[i]
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

    # 结果输出
    ant_print_all(result_printer, model_method="summary_col",
                    results=results, model_names =dum_list,
                    regressor_order = ["new_use"], drop_omitted=True,
                    info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"],float_format='% .3f')
```

```
=====
all_consumpt_dum
-----
```

```
new_use    0.029***
           (0.006)
rsquared    0.580
count       60060.000
=====
```

Standard errors in
parentheses.

* p<.1, ** p<.05,

***p<.01

Last executed in 1 s, finished in 21:06:30 2025-06-30

表I 6 疫情与消费选择的稳健性检验



In [10]:

```
regdata['newuse_p_case'] = regdata['new_use'] * regdata['p_case_人口平减']

reg = regdata.dropna(subset=['p_case_人口平减'])

covid_var = ["p_case"]
cats = reg[['用户id_rank', 'time_city', 'time_type']]
cluster_id = reg['cluster_id']

if envir()=='仿真环境':
    pass
else:
    for j in covid_var:
        interact = f'newuse_{j}'
        x = reg[["new_use", interact]]

        y_list = ['complement_dum', 'substitutue_dum'] # 是否互补消费 是否替代消费
        models = []
        results = []
        for i in y_list:
            y = reg[i]
            model = AbsorbingLS(y, x, absorb=cats)
            result = model.fit(method="lsmr", cov_type="clustered", clusters
=cluster_id)
            models.append(model)
            results.append(result)

        # 结果输出
        ant_print_all(result_printer, model_method="summary_col",
                      results=results, model_names = y_list,
                      regressor_order = ["new_use",interact], drop_omitted=
True,
                      info_list=["rsquared"], info_dict={"count": lambda x:
x.nobs}, info_order=["rsquared", "count"],float_format='% .3f')
```



```

*****

=====
                        complement_dum  substitue_dum
-----
new_use          0.024***          0.017**
                  (0.009)          (0.009)
newuse_p_case   -0.143***          0.010
                  (0.055)          (0.056)
rsquared         0.553             0.470
count           58762.000          58762.000
=====

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****

```

Last executed in 4 s, finished in 21:06:34 2025-06-30

附录II

表II 1 排除调节变量的主导作用

In [11]:

```
adjust = regdata.dropna(subset=['残疾干部残联比例'])
model_list = ['lnn_module_consumpt'] * 5

if regdata.shape[0] > 200:
    cats = adjust[['用户id_rank', 'time_type']]
    cluster_id = adjust['cluster_id']

    models = []
    results = []
    regressor_order = []

    for col in ["残疾干部残联比例", "村级残协", "残疾证", "建设面积", "县域政务网站无障碍比例"]:
        x = adjust['new_use']
        y = adjust['lnn_module_consumpt']
        interact = Interaction(adjust['year_month_cat'], adjust[col])
        model = AbsorbingLS(y, x, absorb=cats, interactions=interact)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=cluster_id)
        models.append(model)
        results.append(result)

        # 将当前的'newuse_{col}' 和'new_use' 加入 regressor_order 列表
        regressor_order.append(f'newuse_{col}')

    # 最后, 将'new_use' 加入 regressor_order
    regressor_order.append('new_use')

    # 结果输出
    ant_print_all(result_printer, model_method="summary_col",
                  results=results, model_names=model_list,
                  regressor_order=regressor_order, drop_omitted=True,
                  info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs},
                  info_order=["rsquared", "count"], float_format='%.3f')
```

=====

lnn_module_consumpt I lnn_module_consumpt II lnn_module_consumpt
III lnn_module_consumpt IIII lnn_module_consumpt IIIII

new_use	0.054***	0.054***	0.054***
	0.054***	0.055***	
	(0.011)	(0.011)	(0.011)
	(0.011)	(0.011)	
rsquared	0.723	0.723	0.723
	0.723	0.723	
count	60038.000	60038.000	60038.000
	60038.000	60038.000	

=====

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

Last executed in 8 s, finished in 21:06:41 2025-06-30

表II 2 线下无障碍环境建设的替代效应检验 

In [12]:

```
model_list = ['all_consumpt_dum', 'all_consumpt_dum']

if regdata.shape[0] > 200:
    cats = adjust[['用户id_rank', 'time_city', 'time_type']]
    cluster_id = adjust['cluster_id']

    models = []
    results = []
    regressor_order = []

    for col in ['残疾证', '建设面积']:
        x = adjust[[f'newuse_{col}', f'newuse_{col}_平方', 'new_use']]
        y = adjust['all_consumpt_dum']
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

        # 将当前的'newuse_{col}' 和'new_use' 加入 regressor_order 列表
        regressor_order.append(f'newuse_{col}')
        regressor_order.append(f'newuse_{col}_平方')

    # 最后, 将'new_use' 加入 regressor_order
    regressor_order.append('new_use')

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names=model_list,
               regressor_order=regressor_order, drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs},
               info_order=["rsquared", "count"], float_format='%.6f')
```

```

*****

=====
all_consumpt_dum I all_consumpt_dum II
-----
newuse_残疾证 0.008785*** -
(0.003032) -
newuse_残疾证_平方 -0.000133*** -
(0.000046) -
newuse_建设面积 - 0.010529***
- (0.003570)
newuse_建设面积_平方 - -0.000172***
- (0.000057)
new_use 0.000082 -0.001098
(0.010755) (0.010952)
rsquared 0.580911 0.580921
count 60038.000000 60038.000000
=====
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****

```

Last executed in 5 s, finished in 21:06:46 2025-06-30

附录III

助盲功能与线上线下消费提升

In [13]:

```
# 线上
lnty_list = ['lntonline_modules', 'lntonline_consumpt', 'lntonline_consumpt_num']

if regdata.shape[0]>20000:
    cats = regdata[['用户id_rank', 'time_city', 'time_type']] # 固定效应
    cluster_id = regdata['cluster_id'] # 聚类层级
    x = regdata["new_use"] # 解释变量

    models = []
    results = []
    for i in lnty_list:
        y = regdata[i] # 被解释变量
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names =lnty_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"],float_format='% .3f')
```

```
=====
               lntonline_modules lntonline_consumpt lntonline_consumpt_num
-----
new_use    0.031***           0.128***           0.053***
           (0.010)           (0.038)           (0.017)
rsquared    0.733             0.704             0.741
count      60060.000          60060.000          60060.000
=====
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****
```

Last executed in 4 s, finished in 21:06:50 2025-06-30

In [14]:

```
# 线下
lnty_list = ['lnoffline_modules', 'lnoffline_consumpt', 'lnoffline_consumpt_num']

if regdata.shape[0]>20000:
    cats = regdata[['用户id_rank', 'time_city', 'time_type']] # 固定效应
    cluster_id = regdata['cluster_id'] # 聚类层级
    x = regdata["new_use"] # 解释变量

    models = []
    results = []
    for i in lnty_list:
        y = regdata[i] # 被解释变量
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names =lnty_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"],float_format='% .3f')
```

```
=====
lnoffline_modules lnoffline_consumpt lnoffline_consumpt_num
-----
new_use  0.031***          0.121**          0.047**
          (0.011)          (0.053)          (0.023)
rsquared 0.738            0.689            0.762
count    60060.000        60060.000        60060.000
=====
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****
```

Last executed in 4 s, finished in 21:06:54 2025-06-30

表A2 视觉无障碍功能与消费便利性提升-补充分析

In [15]:

```
categories = ['医疗健康', '公共服务', '教育培训']
dummy_list = [f'{board}_dum' for board in categories]
cats = regdata[['用户id_rank', 'time_city', 'time_type']]
cluster_id = regdata['cluster_id']

# 取对数
if regdata.shape[0]>20000:
    x = regdata["new_use"]
    models = []
    results = []
    for i in dummy_list:
        y = regdata[i]
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names=dummy_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"], float_format='%.3f')
```

```
=====
               医疗健康_dum   公共服务_dum   教育培训_dum
-----
new_use   0.006         -0.003         -0.003
          (0.007)       (0.002)       (0.003)
rsquared  0.411         0.234         0.267
count     60060.000    60060.000    60060.000
=====
```

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Last executed in 4 s, finished in 21:06:58 2025-06-30

附录I 样本选择偏误 表I3

In [16]:

```
df = ant_read_data('data_heckman', mid_table = True)

df['报告期1'] = df['报告期'].astype('str')
df['报告期1'] = pd.to_datetime(df['报告期1'])
df['year_month'] = df['报告期1'].apply(lambda x: x.strftime("%Y-%m"))

df['provcde'] = df.groupby(['常驻省份'])['常驻省份'].ngroup()
df['citycode'] = df.groupby(['常驻省份', '常驻城市'])['常驻省份', '常驻城市'].ngroup()
df['countycode'] = df.groupby(['常驻省份', '常驻城市', '常驻区县'])['常驻省份', '常驻城市', '常驻区县'].ngroup()
```

Last executed in 9 s, finished in 21:07:07 2025-06-30

In [17]:

```
df['year_month_cat'] = df['year_month'].astype('category')
df['citycode_cat'] = df['citycode'].astype('category')
df['time_city'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'], row['citycode_cat']), axis=1).astype('category')
df['time_type'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'], row['type']), axis=1).astype('category')
df['用户id_rank'] = df['匿名化用户id_rank'].astype('category')
df['cluster_id'] = df['匿名化用户id_rank'].astype('category')

df = df.rename(columns={"yb状态": "是否医保", "rs状态": "是否社保"})
```

Last executed in 4 s, finished in 21:07:11 2025-06-30

In [18]:

```
# 一阶段
# y 是否有观测值的虚拟变量, y=0 表示在使用时长小于3个月的样本中消费全为0的个体, 即视为
# 有开通但未使用视觉无障碍功能的用户
# x_select是选择方程中的自变量
y = df['y']
x_select = df[['age_45', '是否医保', '是否社保', 'new_use']]
ant_print_all(x_select, model_method="describe")
if df.shape[0]>200:
    model = sm.Probit(df['y'], x_select)
    cluster_id = df['cluster_id']
    result = model.fit(cov_type="cluster", cov_kwds={'groups': cluster_id})
    ant_print_all(result, model_method="summary")
    df['y_hat'] = result.predict()
    # 计算概率密度函数(pdf) 和累积分布函数(cdf)
    df['pdf'] = norm.pdf(df['y_hat'])
    df['cdf'] = norm.cdf(df['y_hat'])

    # 计算逆米尔斯比率(IMR)
    df['imr'] = df['pdf'] / df['cdf']
```

```
*****
describe
      age_45      是否医保      是否社保      new_use
count 66110.000000 66110.000000 66110.000000 66110.000000
mean   0.715474    0.403661    0.306822    0.718817
std    0.451192    0.490635    0.461178    0.449580
min    0.000000    0.000000    0.000000    0.000000
25%    0.000000    0.000000    0.000000    0.000000
50%    1.000000    0.000000    0.000000    1.000000
75%    1.000000    1.000000    1.000000    1.000000
max    1.000000    1.000000    1.000000    1.000000
*****
```

Probit Regression Results

=====

Dep. Variable: y No. Observations: 66110

Model: Probit Df Residuals: 66106

Method: MLE Df Model: 3

Date: Mon, 30 Jun 2025 Pseudo R-squ.: 0.1430

Time: 13:07:14 Log-Likelihood: 17338. -

converged: True LL-Null: 20231. -

Covariance Type: cluster LLR p-value: 0.000

=====

	coef	std err	z	P> z	[0.025	0.975]

--						
age_45	0.6828	0.052	13.165	0.000	0.581	
0.784						
是否医保	0.8605	0.091	9.444	0.000	0.682	1.039
是否社保	0.8763	0.128	6.863	0.000	0.626	1.127
new_use	0.7310	0.042	17.454	0.000	0.649	
0.813						
=====						

Last executed in 3 s, finished in 21:07:14 2025-06-30

In [19]:

```
# 二阶段
regdata = df[df['y']==1]
Heckman_list = ['lnn_module_consumpt', 'lnall_consumpt', 'lnnum_consumpt' ]

if regdata.shape[0]>200:
    cats = regdata[['用户id_rank', 'time_city', 'time_type']]
    cluster_id = regdata['cluster_id']
    x = regdata[['new_use', 'imr']]

    models = []
    results = []
    for i in Heckman_list:
        y = regdata[i]
        model = AbsorbingLS(y, x, absorb=cats)
        result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
        models.append(model)
        results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names = Heckman_list,
               regressor_order = ["new_use", "imr"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.
nobs}, info_order=["rsquared", "count"], float_format='% .3f')
```

```
=====
lnn_module_consumpt lnall_consumpt lnnum_consumpt
-----
new_use  0.060***          0.223***          0.105***
          (0.015)          (0.059)          (0.029)
imr       0.254            1.630*           0.659
          (0.233)          (0.930)          (0.446)
rsquared  0.753            0.711            0.769
count     60060.000        60060.000        60060.000
=====
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
*****
```

Last executed in 5 s, finished in 21:07:19 2025-06-30

附录I 样本选择偏误表I4

In [20]:

```
df = ant_read_data('psm_data', mid_table = True)
```

Last executed in 699 s, finished in 21:18:58 2025-06-30

In [21]:

```
## 时间变量
df['报告期1'] = df['报告期'].astype('str')
df['报告期1'] = pd.to_datetime(df['报告期1'])
df['year_month'] = df['报告期1'].apply(lambda x: x.strftime("%Y-%m"))
df['type'] = df['period_length'].astype('category')
```

Last executed in 83 s, finished in 21:20:20 2025-06-30

In [22]:

```
# 固定效应相关变量
df['year_month_cat'] = df['year_month'].astype('category')
df['citycode_cat'] = df['citycode'].astype('category')
df['time_city'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'],
row['citycode_cat']), axis=1).astype('category')
df['time_type'] = df.apply(lambda row: '{}_{}'.format(row['year_month_cat'],
row['type']), axis=1).astype('category')
df['用户id_rank'] = df['匿名化用户id_rank'].astype('category')

# 聚类
df['cluster_id'] = df['匿名化用户id_rank'].astype('category')
```

Last executed in 583 s, finished in 21:30:03 2025-06-30

In [23]:

```
df_t= df[df['是否视障用户'] == 1]
df_c = df[df['是否视障用户'] == 0]

#保留视障开始记录当月没有使用的个体
df_t['tag'] = 0
df_t.loc[((df_t['报告期']== 20200331) & (df_t['累计pv'] == 0)), 'tag'] = 1
df_temp = df_t[df_t['tag']== 1 ]
df_temp = df_temp['匿名化用户id_rank'].unique()
df_t = df_t[df_t['匿名化用户id_rank'].isin(df_temp)]

df_t = df_t[(df_t['period_length'] >= 3)]
df = pd.concat([df_t,df_c],ignore_index=True)

# 过渡 便于之后再用
df_original = df
```

Last executed in 11 s, finished in 21:30:14 2025-06-30

In [24]:

```
# 根据事前的消费数据选出特征相似的个体
df = df[df['报告期'] < 20200331]
## 计算事前的消费特征, 用均值匹配
df['all_consumpt_mean'] = df.groupby('匿名化用户id_rank')['all_consumpt'].transform('mean')
df['num_consumpt_mean'] = df.groupby('匿名化用户id_rank')['num_consumpt'].transform('mean')
df['n_module_consumpt_mean'] = df.groupby('匿名化用户id_rank')['n_module_consumpt'].transform('mean')
```

Last executed in 7 s, finished in 21:30:21 2025-06-30

In [25]:

```
xvar = ["all_consumpt_mean", "num_consumpt_mean", "n_module_consumpt_mean"]
test = df[df['是否视障用户'] == 1]
test = test[["匿名化用户id_rank", "all_consumpt_mean", "num_consumpt_mean",
"n_module_consumpt_mean", "是否视障用户"]]
test = test.drop_duplicates(subset="匿名化用户id_rank")

control = df[df['是否视障用户'] == 0]
control = control[["匿名化用户id_rank", "all_consumpt_mean",
"num_consumpt_mean", "n_module_consumpt_mean", "是否视障用户"]]
control = control.drop_duplicates(subset="匿名化用户id_rank")

# 初始化模型并计算倾向分
model = PSM(test=test, control=control, xvar=xvar, yvar="是否视障用户")
# 支持probit/logit
model.calculate_pscore(covariates=xvar, method="probit")
# 打印probit的模型拟合结果
ant_print_all(model.pscore_model, model_method="summary")

model.match(nmatches=1, replacement=False, radius=None, measure='covariates',
, scale=True, seed=0)
```

Probit Regression Results

```
=====
Dep. Variable:          y      No. Observations:
148545
Model:                  Probit   Df Residuals:
148541
Method:                 MLE      Df Model:
3
Date:                   Mon, 30 Jun 2025   Pseudo R-squ.:
0.01041
Time:                   13:30:28   Log-Likelihood:      -
13474.
converged:              True      LL-Null:      -
13615.
Covariance Type:        nonrobust   LLR p-value:      3.762e-
61
=====
```

	coef	std err	z	P> z	
[0.025 0.975]					

all_consumpt_mean	1.781e-07	3.69e-07	0.482	0.630	-5.46e-
07 9.02e-07					
num_consumpt_mean	0.0019	0.000	8.250	0.000	
0.001 0.002					
n_module_consumpt_mean	-0.0611	0.004	-16.954	0.000	-
0.068 -0.054					
const	-1.8360	0.017	-106.576	0.000	-
1.870 -1.802					

=====

Last executed in 11 s, finished in 21:30:32 2025-06-30

In [26]:

```
# 获取权重, 权重使用前标准化
test_weight, control_weight = model.test_weight.copy(), model.control_weight
.copy()
test_weight /= test_weight.sum()
control_weight /= control_weight.sum()

# 获取匹配后的被用到的样本点的位置索引, 被多次使用的样本点会出现多次
test_indices, control_indices = model.get_indices(stage="after")
selected_test = test.iloc[test_indices]
selected_control = control.iloc[control_indices]

df_PSM = pd.concat([selected_test, selected_control], ignore_index=True)
df_temp = df_PSM['匿名化用户id_rank'].unique()
df_original = df_original[df_original['报告期'] >= 20200331]
regdata = df_original[df_original['匿名化用户id_rank'].isin(df_temp)]
```

Last executed in 2 s, finished in 21:30:34 2025-06-30

In [27]:

```
regdata['all_consumpt']=winsorize(regdata['all_consumpt'],limits=[0, 0.01],
nan_policy='omit')

for i in ['all_consumpt', 'num_consumpt', 'n_module_consumpt']:
    regdata['ln' + i] = np.log1p(regdata[i])
```

Last executed in 0 s, finished in 21:30:34 2025-06-30

In [28]:

```
cats = regdata[['用户id_rank', 'time_city', 'time_type']]
cluster_id = regdata['cluster_id']
y_list = ['lnn_module_consumpt', 'lnall_consumpt', 'lnnum_consumpt' ]
x = regdata["new_use"]
models = []
results = []
for i in y_list:
    y = regdata[i]
    model = AbsorbingLS(y, x, absorb=cats)
    result = model.fit(method="lsmr", cov_type="clustered", clusters=
cluster_id)
    models.append(model)
    results.append(result)

# 结果输出
ant_print_all(result_printer, model_method="summary_col",
               results=results, model_names =y_list,
               regressor_order = ["new_use"], drop_omitted=True,
               info_list=["rsquared"], info_dict={"count": lambda x: x.nobs},
               info_order=["rsquared", "count"], float_format='%.3f')
```

```
=====
lnn_module_consumpt lnall_consumpt lnnum_consumpt
-----
new_use  0.050***          0.157***          0.080***
          (0.011)          (0.044)          (0.021)
rsquared 0.745            0.709            0.757
count    120120.000       120120.000       120120.000
=====
```

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Last executed in 8 s, finished in 21:30:42 2025-06-30