Boombox exercise - Solution

In [ ]:
```
!pip install linearmodels

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.formula.api import ols
from linearmodels.panel import PanelOLS

# Set up modern styling
sns.set_theme(style="whitegrid")
```

```
Collecting linearmodels
  Downloading linearmodels-6.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl.metadata (7.9 kB)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.11/dist
-packages (from linearmodels) (1.26.4)
Requirement already satisfied: pandas>=1.4.0 in /usr/local/lib/python3.11/dist-pa
ckages (from linearmodels) (2.2.2)
Requirement already satisfied: scipy>=1.8.0 in /usr/local/lib/python3.11/dist-pac
kages (from linearmodels) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.0 in /usr/local/lib/python3.11/d
ist-packages (from linearmodels) (0.14.4)
Collecting mypy-extensions>=0.4 (from linearmodels)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: Cython>=3.0.10 in /usr/local/lib/python3.11/dist-p
ackages (from linearmodels) (3.0.12)
Collecting pyhdfe>=0.1 (from linearmodels)
  Downloading pyhdfe-0.2.0-py3-none-any.whl.metadata (4.0 kB)
Collecting formulaic>=1.0.0 (from linearmodels)
  Downloading formulaic-1.1.1-py3-none-any.whl.metadata (6.9 kB)
Collecting setuptools-scm<9.0.0,>=8.0.0 (from setuptools-scm[toml]<9.0.0,>=8.0.0-
>linearmodels)
  Downloading setuptools_scm-8.1.0-py3-none-any.whl.metadata (6.6 kB)
Collecting interface-meta>=1.2.0 (from formulaic>=1.0.0->linearmodels)
  Downloading interface_meta-1.3.0-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python
3.11/dist-packages (from formulaic>=1.0.0->linearmodels) (4.12.2)
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.11/dist-packa
ges (from formulaic>=1.0.0->linearmodels) (1.17.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.1
1/dist-packages (from pandas>=1.4.0->linearmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pac
kages (from pandas>=1.4.0->linearmodels) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-p
ackages (from pandas>=1.4.0->linearmodels) (2025.1)
Requirement already satisfied: packaging>=20 in /usr/local/lib/python3.11/dist-pa
ckages (from setuptools-scm<9.0.0,>=8.0.0->setuptools-scm[toml]<9.0.0,>=8.0.0->li
nearmodels) (24.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packa
ges (from setuptools-scm<9.0.0,>=8.0.0->setuptools-scm[toml]<9.0.0,>=8.0.0->linea
rmodels) (75.1.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-pac
kages (from statsmodels>=0.13.0->linearmodels) (1.0.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-package
s (from python-dateutil>=2.8.2->pandas>=1.4.0->linearmodels) (1.17.0)
Downloading linearmodels-6.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl (1.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 33.9 MB/s eta 0:00:00
Downloading formulaic-1.1.1-py3-none-any.whl (115 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.7/115.7 kB 7.2 MB/s eta 0:00:00
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pyhdfe-0.2.0-py3-none-any.whl (19 kB)
Downloading setuptools_scm-8.1.0-py3-none-any.whl (43 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.7/43.7 kB 2.9 MB/s eta 0:00:00
Downloading interface_meta-1.3.0-py3-none-any.whl (14 kB)
Installing collected packages: setuptools-scm, mypy-extensions, interface-meta, p
yhdfe, formulaic, linearmodels
Successfully installed formulaic-1.1.1 interface-meta-1.3.0 linearmodels-6.1 mypy
-extensions-1.0.0 pyhdfe-0.2.0 setuptools-scm-8.1.0
```

1. Import and format the dataset

In [ ]:
```
# Load and prepare data

file_path = 'https://www.dropbox.com/scl/fi/6hvfnl04zwik484zm3n2r/boombox.csv?rl

# Load the data
try:
    df = pd.read_csv(file_path)
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='latin-1')

df.head()
```

Out[ ]:

	curators_earned	ib_curators_earned	artists_earned	ib_artists_earned	active_artists_ea
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

5 rows × 42 columns

In [ ]:
```
# Convert week format (2019w33) to datetime
def convert_week_to_date(week_str):
    year = int(week_str[:4])
    week_num = int(week_str[5:])
    return pd.to_datetime(f"{year}-01-01") + pd.Timedelta(weeks=week_num-1)

# Apply conversion and create additional time-related columns
df['date'] = df['week'].apply(convert_week_to_date)
df['year'] = df['date'].dt.year
df['week_num'] = df['date'].dt.isocalendar().week

# Sort the dataframe by date
df = df.sort_values('date')

# Create a numeric week column for plotting
df['week_numeric'] = range(len(df))
```

In [ ]:
```
# Display basic information about the dataset
print("Dataset Overview:")
print(f"Number of rows: {len(df)}")
print(f"Number of unique countries: {df['country'].nunique()}")
print(f"Date range: from {df['date'].min()} to {df['date'].max()}")
df.describe()
```

```
Dataset Overview:
Number of rows: 21240
Number of unique countries: 177
Date range: from 2019-06-04 00:00:00 to 2022-06-25 00:00:00
```

Out[ ]:

	curators_earned	ib_curators_earned	artists_earned	ib_artists_earned	active_artis
count	21240.000000	21240.000000	21240.000000	21240.000000	212
mean	0.174953	8.201036	0.750989	15.491902	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	1.000000	0.000000	1.000000	
max	27.000000	768.000000	175.000000	4336.000000	
std	0.993312	45.607938	6.393790	149.178892	

8 rows × 44 columns

2. Counterfactual simulations

Create the different scenarios and log-transform the variables

In [ ]:

```
# First ensure the data is sorted by country and date
df = df.sort_values(['country', 'date'])

# Create artists and curators variables in scenarios with:
# 1. no earned media
df['artists_no_earned'] = df['artists'] - df['artists_earned']
df['curators_no_earned'] = df['curators'] - df['curators_earned']
df['ib_artists_no_earned'] = df['ib_artists'] - df['ib_artists_earned']
df['ib_curators_no_earned'] = df['ib_curators'] - df['ib_curators_earned']

# 2. no owned media
df['artists_no_owned'] = df['artists'] - df['artists_owned']
df['curators_no_owned'] = df['curators'] - df['curators_owned']
df['ib_artists_no_owned'] = df['ib_artists'] - df['ib_artists_owned']
df['ib_curators_no_owned'] = df['ib_curators'] - df['ib_curators_owned']

#3. no paid media
df['artists_no_paid'] = df['artists'] - df['artists_paid']
df['curators_no_paid'] = df['curators'] - df['curators_paid']
df['ib_artists_no_paid'] = df['ib_artists'] - df['ib_artists_paid']
```

```
df['ib_curators_no_paid'] = df['ib_curators'] - df['ib_curators_paid']

# Create lag variables (month-1 variables) using groupby and shift
df['artists-1'] = df.groupby('country')['artists'].shift(1)
df['artists_no_earned-1'] = df.groupby('country')['artists_no_earned'].shift(1)
df['artists_no_owned-1'] = df.groupby('country')['artists_no_owned'].shift(1)
df['artists_no_paid-1'] = df.groupby('country')['artists_no_paid'].shift(1)

df['curators-1'] = df.groupby('country')['curators'].shift(1)
df['curators_no_earned-1'] = df.groupby('country')['curators_no_earned'].shift(1
df['curators_no_owned-1'] = df.groupby('country')['curators_no_owned'].shift(1)
df['curators_no_paid-1'] = df.groupby('country')['curators_no_paid'].shift(1)

# Take log of dependent/independent variables (adding 1 to handle zeros)
log_vars = ['artists','curators','artists-1','curators-1','artists_no_earned','c
for var in log_vars:
    df[f'log_{var}'] = np.log(df[var] + 1)
```

Fixed effects regressions to estimate network effects for artists and curators

In [ ]:
```
# For panel regression, we need a multi-index with country and time
df.set_index(['country', 'date'], inplace=True)

# Artists model with fixed effects
artist_fe = PanelOLS(
    dependent=df['log_artists'],
    exog=df[['log_artists-1', 'log_ib_curators']],
    entity_effects=True,  # country fixed effects
    time_effects=True     # time fixed effects
).fit()

print("\nFixed Effects Results - Artists:")
print(artist_fe.summary.tables[1])

# Curators model with fixed effects
artist_fe = PanelOLS(
    dependent=df['log_curators'],
    exog=df[['log_curators-1', 'log_ib_artists']],
    entity_effects=True,  # country fixed effects
    time_effects=True     # time fixed effects
).fit()

print("\nFixed Effects Results - Curators:")
print(artist_fe.summary.tables[1])
```

```
/usr/local/lib/python3.11/dist-packages/linearmodels/panel/model.py:1260: Missing
ValueWarning:
Inputs contain missing values. Dropping rows with missing observations.
  super().__init__(dependent, exog, weights=weights, check_rank=check_rank)
```

```
Fixed Effects Results - Artists:
                                Parameter Estimates
==========================================================================
==
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper
CI
--------------------------------------------------------------------------
--
log_artists-1       0.6810     0.0051     133.08     0.0000      0.6710      0.69
10
log_ib_curators     0.1399     0.0044     31.469     0.0000      0.1312      0.14
86
==========================================================================
==
/usr/local/lib/python3.11/dist-packages/linearmodels/panel/model.py:1260: Missing
ValueWarning:
Inputs contain missing values. Dropping rows with missing observations.
  super().__init__(dependent, exog, weights=weights, check_rank=check_rank)
Fixed Effects Results - Curators:
                                Parameter Estimates
==========================================================================
=
                Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper C
I
--------------------------------------------------------------------------
-
log_curators-1     0.3127     0.0066     47.397     0.0000      0.2997      0.325
6
log_ib_artists     0.0511     0.0022     23.557     0.0000      0.0468      0.055
3
==========================================================================
=
```

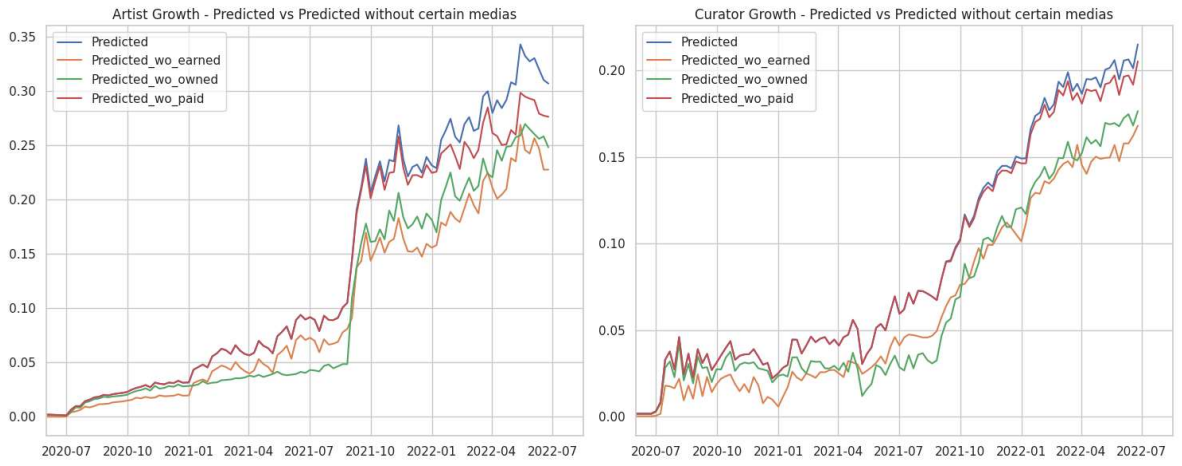
Use fixed-effects regressions estimates to predict/simulate counterfactual scenarios

In [ ]:
```
# Get artists predictions
df['predicted_artists'] = artist_fe.predict(df[['log_artists-1', 'log_ib_curator
df['predicted_artists_wo_earned'] = artist_fe.predict(df[['log_artists_no_earned
df['predicted_artists_wo_owned'] = artist_fe.predict(df[['log_artists_no_owned-1
df['predicted_artists_wo_paid'] = artist_fe.predict(df[['log_artists_no_paid-1',

# Get curators predictions
df['predicted_curators'] = artist_fe.predict(df[['log_curators-1', 'log_ib_artis
df['predicted_curators_wo_earned'] = artist_fe.predict(df[['log_curators_no_earn
df['predicted_curators_wo_owned'] = artist_fe.predict(df[['log_curators_no_owned
df['predicted_curators_wo_paid'] = artist_fe.predict(df[['log_curators_no_paid-1
```

Plot the results of the simulations

In [ ]:
```
# Reset index for plotting
df.reset_index(inplace=True)

# Plot actual vs predicted values
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
```

```
# Artists plot
monthly_artists = df.groupby('date')[['predicted_artists', 'predicted_artists_wo
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists'], label='Pre
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_earned'],
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_owned'], l
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_paid'], la
ax1.set_title('Artist Growth - Predicted vs Predicted without certain medias')
ax1.legend()
ax1.grid(True)
#change x axis to start june 2020
ax1.set_xlim(pd.to_datetime('2020-06-01'))

# Curators plot
monthly_curators = df.groupby('date')[['predicted_curators', 'predicted_curators
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators'], label='
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_earned'
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_owned']
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_paid'],
ax2.set_title('Curator Growth - Predicted vs Predicted without certain medias')
ax2.legend()
ax2.grid(True)
#change x axis to start june 2020
ax2.set_xlim(pd.to_datetime('2020-06-01'))

plt.tight_layout()
plt.show()
```

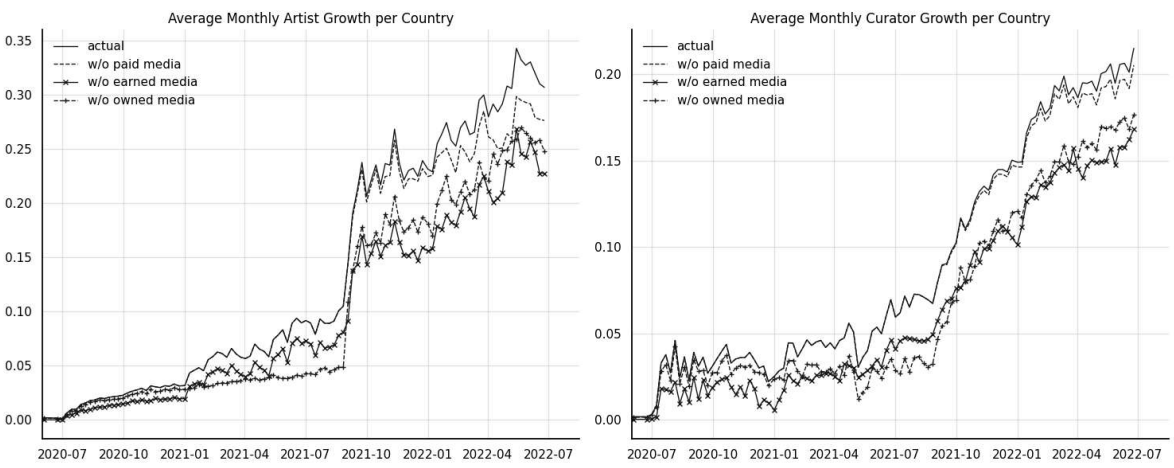


Customized plot style:

In [ ]:

```
# Reset index for plotting
df.reset_index(inplace=True)

plt.style.use('grayscale')

# Create the plot
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
fig.patch.set_facecolor('white')  # Set figure background to white
ax1.set_facecolor('white')  # Set plot background to white
ax2.set_facecolor('white')

# Artists plot
monthly_artists = df.groupby('date')[['predicted_artists', 'predicted_artists_wo
                                      'predicted_artists_wo_owned', 'predicted_art
```

```
# Plot with specific line styles
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists'],
         'k-', linewidth=1, label='actual')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_paid'],
         'k--', linewidth=1, label='w/o paid media')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_earned'],
         'k-', marker='x', linewidth=1, markersize=4, label='w/o earned media')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_owned'],
         'k--', marker='+', linewidth=1, markersize=4, label='w/o owned media')

# Customize grid and appearance
ax1.grid(True, color='grey', linestyle='-', alpha=0.2)
ax1.set_title('Average Monthly Artist Growth per Country')
ax1.set_xlim(pd.to_datetime('2020-06-01'))
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)

# Similar for curators plot
monthly_curators = df.groupby('date')[['predicted_curators', 'predicted_curators
                                       'predicted_curators_wo_owned', 'predicted_c

ax2.plot(monthly_curators.index, monthly_curators['predicted_curators'],
         'k-', linewidth=1, label='actual')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_paid'],
         'k--', linewidth=1, label='w/o paid media')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_earned'
         'k-', marker='x', linewidth=1, markersize=4, label='w/o earned media')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_owned']
         'k--', marker='+', linewidth=1, markersize=4, label='w/o owned media')

ax2.grid(True, color='grey', linestyle='-', alpha=0.2)
ax2.set_title('Average Monthly Curator Growth per Country')
ax2.set_xlim(pd.to_datetime('2020-06-01'))
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)

# Customize legends
ax1.legend(frameon=False)
ax2.legend(frameon=False)

plt.tight_layout()
plt.show()
```



Alternative specifications

In [ ]:
```
# For panel regression, we need a multi-index with country and time
df.set_index(['country', 'date'], inplace=True)

# Artists model with fixed effects (alt specification)
artist_fe = PanelOLS(
    dependent=df['log_artists'],
    exog=df[['log_ib_artists', 'log_ib_curators']],
    entity_effects=True,  # country fixed effects
    time_effects=True     # time fixed effects
).fit()

print("\nFixed Effects Results - Artists:")
print(artist_fe.summary.tables[1])


# Get predictions

df['predicted_artists'] = artist_fe.predict(df[['log_ib_artists', 'log_ib_curato
df['predicted_artists_wo_earned'] = artist_fe.predict(df[['log_ib_artists_no_ear
df['predicted_artists_wo_owned'] = artist_fe.predict(df[['log_ib_artists_no_owne
df['predicted_artists_wo_paid'] = artist_fe.predict(df[['log_ib_artists_no_paid'

# Curators model with fixed effects (alt specification)
artist_fe = PanelOLS(
    dependent=df['log_curators'],
    exog=df[['log_ib_curators', 'log_ib_artists']],
    entity_effects=True,  # country fixed effects
    time_effects=True     # time fixed effects
).fit()

print("\nFixed Effects Results - Curators:")
print(artist_fe.summary.tables[1])

# Get predictions
df['predicted_curators'] = artist_fe.predict(df[['log_ib_curators', 'log_ib_arti
df['predicted_curators_wo_earned'] = artist_fe.predict(df[['log_ib_curators_no_e
df['predicted_curators_wo_owned'] = artist_fe.predict(df[['log_ib_curators_no_ow
df['predicted_curators_wo_paid'] = artist_fe.predict(df[['log_ib_curators_no_pai

# Reset index for plotting
df.reset_index(inplace=True)

plt.style.use('grayscale')

# Create the plot
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
fig.patch.set_facecolor('white')  # Set figure background to white
ax1.set_facecolor('white')  # Set plot background to white
ax2.set_facecolor('white')

# Artists plot
monthly_artists = df.groupby('date')[['predicted_artists', 'predicted_artists_wo
                                      'predicted_artists_wo_owned', 'predicted_art

# Plot with specific line styles
```

```
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists'],
         'k-', linewidth=1, label='actual')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_paid'],
         'k--', linewidth=1, label='w/o paid media')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_earned'],
         'k-', marker='x', linewidth=1, markersize=4, label='w/o earned media')
ax1.plot(monthly_artists.index, monthly_artists['predicted_artists_wo_owned'],
         'k--', marker='+', linewidth=1, markersize=4, label='w/o owned media')

# Customize grid and appearance
ax1.grid(True, color='grey', linestyle='-', alpha=0.2)
ax1.set_title('Average Monthly Artist Growth per Country')
ax1.set_xlim(pd.to_datetime('2020-06-01'))
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)

# Similar for curators plot
monthly_curators = df.groupby('date')[['predicted_curators', 'predicted_curators
                                       'predicted_curators_wo_owned', 'predicted_c

ax2.plot(monthly_curators.index, monthly_curators['predicted_curators'],
         'k-', linewidth=1, label='actual')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_paid'],
         'k--', linewidth=1, label='w/o paid media')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_earned'
         'k-', marker='x', linewidth=1, markersize=4, label='w/o earned media')
ax2.plot(monthly_curators.index, monthly_curators['predicted_curators_wo_owned']
         'k--', marker='+', linewidth=1, markersize=4, label='w/o owned media')

ax2.grid(True, color='grey', linestyle='-', alpha=0.2)
ax2.set_title('Average Monthly Curator Growth per Country')
ax2.set_xlim(pd.to_datetime('2020-06-01'))
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)

# Customize legends
ax1.legend(frameon=False)
ax2.legend(frameon=False)

plt.tight_layout()
plt.show()
```

```
Fixed Effects Results - Artists:
                              Parameter Estimates
==============================================================================
==
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper
CI
------------------------------------------------------------------------------
--
log_ib_artists      0.5001     0.0042     118.29     0.0000      0.4918      0.50
84
log_ib_curators    -0.0010     0.0054    -0.1875     0.8513     -0.0116      0.00
96
==============================================================================
==

Fixed Effects Results - Curators:
                              Parameter Estimates
==============================================================================
==
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper
CI
------------------------------------------------------------------------------
--
log_ib_curators     0.1836     0.0035     51.758     0.0000      0.1767      0.19
06
log_ib_artists     -0.0180     0.0028    -6.4623     0.0000     -0.0234     -0.01
25
==============================================================================
==
```