In [ ]:
```
import sys
sys.path.append('..')
```

In [ ]:
```
from mesa import batch_run
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import sys
import contextlib
from competition.model import InnovationModel
from competition.agent import FirmAgent
```

Single Run

In this single run, I aim to demonstrate what a typical simulation would look like by varying my research variables of interest while keeping control variables fixed. The control variables such as the number of firms, baseline success probability, TAR increment, and success probability adjustment are held constant to maintain consistency across different scenarios. The primary focus is on three key variables: average node degree, network effect, and innovation gap. These variables are varied through low, average, and high versions to observe how they influence the number of firms innovating at each step and the overall steps count.

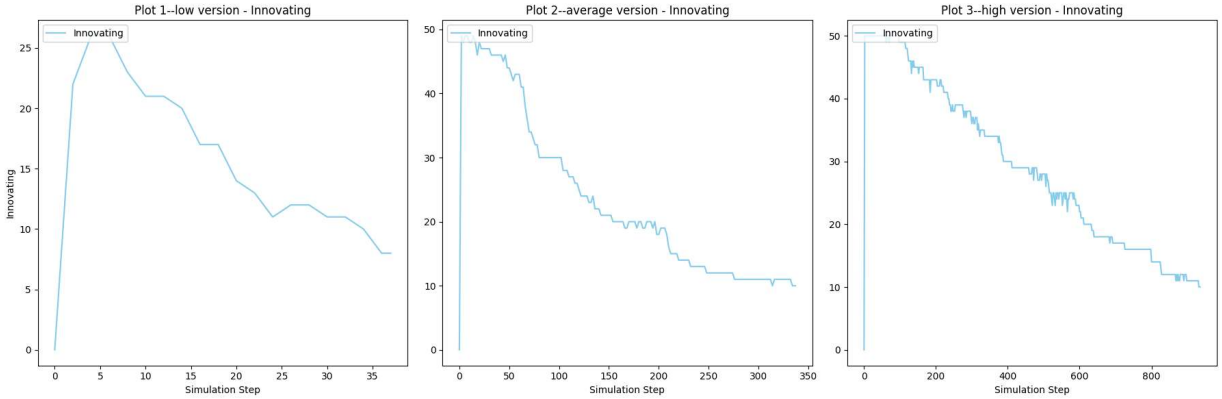
From the plots, we can observe distinct patterns across the different scenarios. In the low version scenario, the count of innovating firms peaks early but then declines sharply before all firms have had the opportunity to innovate. This scenario also has the fewest simulation steps, indicating a faster decline in innovation activity. As the parameters increase in the average and high versions, the number of simulation steps also increases. This is likely due to the larger innovation gap, which allows more firms to continue innovating and remain competitive in the market for a longer duration. The increase in network effect and average node degree in the higher versions may promotes sustained innovation activity, leading to a more gradual decline in the number of innovating firms.

In [ ]:
```
import os
import contextlib
import pandas as pd
import matplotlib.pyplot as plt
from mesa.batchrunner import batch_run
from competition.model import InnovationModel

@contextlib.contextmanager
def suppress_output():
    with open(os.devnull, "w") as devnull:
        old_stdout = os.sys.stdout
        os.sys.stdout = devnull
        try:
```

```
            yield
        finally:
            os.sys.stdout = old_stdout

parameter_sets = {
    'Plot 1--low version': {
        "num_firms": 50,
        "avg_node_degree": 3,
        "baseline_success_prob": 0.50,
        "innovation_gap": 10,
        "network_effect": 0.02,
        "distribution": "normal",
        "tar_gain": 6,
        "success_prob_adjustment": 0.08
    },
    'Plot 2--average version': {
        "num_firms": 50,
        "avg_node_degree": 5,
        "baseline_success_prob": 0.50,
        "innovation_gap": 35,
        "network_effect": 0.04,
        "distribution": "normal",
        "tar_gain": 6,
        "success_prob_adjustment": 0.08
    },
    'Plot 3--high version': {
        "num_firms": 50,
        "avg_node_degree": 8,
        "baseline_success_prob": 0.50,
        "innovation_gap": 60,
        "network_effect": 0.05,
        "distribution": "normal",
        "tar_gain": 6,
        "success_prob_adjustment": 0.08
    }
}

def run_and_plot_simulation():
    # Setup for subplots: 1 row, 3 columns
    fig, axes = plt.subplots(1, 3, figsize=(18, 6))

    for idx, (scenario, params) in enumerate(parameter_sets.items()):
        with suppress_output():
            batch = batch_run(
                InnovationModel,
                parameters=params,
                max_steps=1000,
                iterations=1,
                data_collection_period=2
            )

        results = pd.DataFrame(batch)

        # Plot Innovating
        axes[idx].plot(results['Step'], results['Innovating'], label='Innovating',
        axes[idx].set_title(f'{scenario} - Innovating')
```

```
        axes[idx].set_xlabel('Simulation Step')
        if idx == 0:
            axes[idx].set_ylabel('Innovating')
        axes[idx].legend(loc='upper left')

    plt.tight_layout()
    plt.show()

run_and_plot_simulation()
```

```
0%|          | 0/1 [00:00<?, ?it/s]
0%|          | 0/1 [00:00<?, ?it/s]
0%|          | 0/1 [00:00<?, ?it/s]
```



Batch Run

In [ ]:

```
# from contextlib import contextmanager

# @contextmanager
# def suppress_output():
#     with open(os.devnull, "w") as devnull:
#         old_stdout = os.sys.stdout
#         os.sys.stdout = devnull
#         try:
#             yield
#         finally:
#             os.sys.stdout = old_stdout

# params = {
#     """
#     Perform parameter sweeping to explore the effects of various parameters on th

#     Reasons for parameter sweeping selections could be checked in the paper.
#     """
#     "avg_node_degree": np.linspace(3, 6, 2),
#     "network_effect": np.linspace(0, 0.05, 5),
#     "innovation_gap": np.linspace(10, 61, 6),
#     "num_firms": [50],  # Fixed number of firms
#       "baseline_success_prob": [0.5],
#       "distribution": ["normal"],
#     "tar_gain": np.arange(2, 11, 4),
#     "success_prob_adjustment": np.linspace(0.02, 0.15, 3)
```

```
# }


# def run_and_collect_data():
#     with suppress_output():
#         results = batch_run(
#             InnovationModel,
#             parameters=params,
#             iterations=50,
#             number_processes=os.cpu_count() - 4
#         )
#     df = pd.DataFrame(results)
#     df.to_csv('batch_data2.csv', index=False)
#     return df


# results = run_and_collect_data()

# ## it takes 160 mins in total
```

```
  0%|          | 0/54000 [00:00<?, ?it/s]
```

Analysis & Plotting

1. Load Data

In [ ]:
```
df = pd.read_csv('batch_data2.csv')
df.info
```

Out[ ]: <bound method DataFrame.info of RunId iteration Step avg_node_degree network_effect \

0	6	0	18	3.0	0.00
1	6	0	18	3.0	0.00
2	6	0	18	3.0	0.00
3	6	0	18	3.0	0.00
4	6	0	18	3.0	0.00
...	...	...	...	...	...
2699995	53993	49	1000	8.0	0.05
2699996	53993	49	1000	8.0	0.05
2699997	53993	49	1000	8.0	0.05
2699998	53993	49	1000	8.0	0.05
2699999	53993	49	1000	8.0	0.05

	innovation_gap	num_firms	baseline_success_prob	distribution	\
0	10.0	50	0.5	normal	
1	10.0	50	0.5	normal	
2	10.0	50	0.5	normal	
3	10.0	50	0.5	normal	
4	10.0	50	0.5	normal	
...	...	...	...	...	
2699995	61.0	50	0.5	normal	
2699996	61.0	50	0.5	normal	
2699997	61.0	50	0.5	normal	
2699998	61.0	50	0.5	normal	
2699999	61.0	50	0.5	normal	

	tar_gain	success_prob_adjustment	Innovating	TAR Skewness	\
0	10	0.02	6	-0.073883	
1	10	0.02	6	-0.073883	
2	10	0.02	6	-0.073883	
3	10	0.02	6	-0.073883	
4	10	0.02	6	-0.073883	
...	...	...	...	...	
2699995	2	0.15	50	0.186575	
2699996	2	0.15	50	0.186575	
2699997	2	0.15	50	0.186575	
2699998	2	0.15	50	0.186575	
2699999	2	0.15	50	0.186575	

	0-25th Interval	25-50th Interval	50-75th Interval	\
0	3	3	0	
1	3	3	0	
2	3	3	0	
3	3	3	0	
4	3	3	0	
...	...	...	...	
2699995	13	16	11	
2699996	13	16	11	
2699997	13	16	11	
2699998	13	16	11	
2699999	13	16	11	

	75-100th Interval	AgentID	TAR	Interval
0	5	0	32.660035	0
1	5	1	103.099917	0

```
2                       5        2    31.051835         0
3                       5        3    80.447284         0
4                       5        4    59.733074         0
...                   ...      ...          ...       ...
2699995                10       45  1044.643290         1
2699996                10       46  1065.279403         1
2699997                10       47  1070.408027         2
2699998                10       48  1036.591795         0
2699999                10       49  1026.208971         0

[2700000 rows x 20 columns]>
```

2. Summary Stats Generating & Latex prep

In [ ]:
```
from tabulate import tabulate

## Generating a new var to better assess network influence based on the intensity(n
## quantity(avg_node_degree), which measures the average number of connection firms
df['network_influence']=df['network_effect']*df['avg_node_degree']


selected_columns = ['network_effect','avg_node_degree','network_influence', 'innova
summary_stats = df[selected_columns].describe()

# Convert to LaTeX table
latex_table = tabulate(summary_stats, headers='keys', tablefmt='latex')
print(latex_table)
```

```
\begin{tabular}{lrrrrrrrr}
\hline
       &   network\_effect &   avg\_node\_degree &   network\_influence &   innovati
on\_gap &        TAR &   tar\_gain &   success\_prob\_adjustment &       Step \\
\hline
 count &        2.7e+06   &           2.7e+06 &            2.7e+06  &          2.7e+
06 &    2.7e+06 &    2.7e+06 &                 2.7e+06   &    2.7e+06 \\
 mean  &        0.025     &           5.5     &            0.1375   &         35.5
&  569.143   &    6       &                 0.085     &  425.253   \\
 std   &        0.0176777 &           1.86339 &            0.112731 &         17.419
8  &  537.332   &    3.26599 &                 0.0530723 &  379.791   \\
 min   &        0         &           3       &            0        &         10
&    0       &    2       &                 0.02      &   14       \\
 25\%   &        0.0125    &           4.25    &            0.053125 &         20.2
&  127.82    &    2       &                 0.02      &   89       \\
 50\%   &        0.025     &           5.5     &            0.114583 &         35.5
&  387.947   &    6       &                 0.085     &  259       \\
 75\%   &        0.0375    &           6.75    &            0.208333 &         50.8
& 1028.82    &   10       &                 0.15      &  858       \\
 max   &        0.05      &           8       &            0.4      &         61
& 3226.08    &   10       &                 0.15      & 1000       \\
\hline
\end{tabular}
```

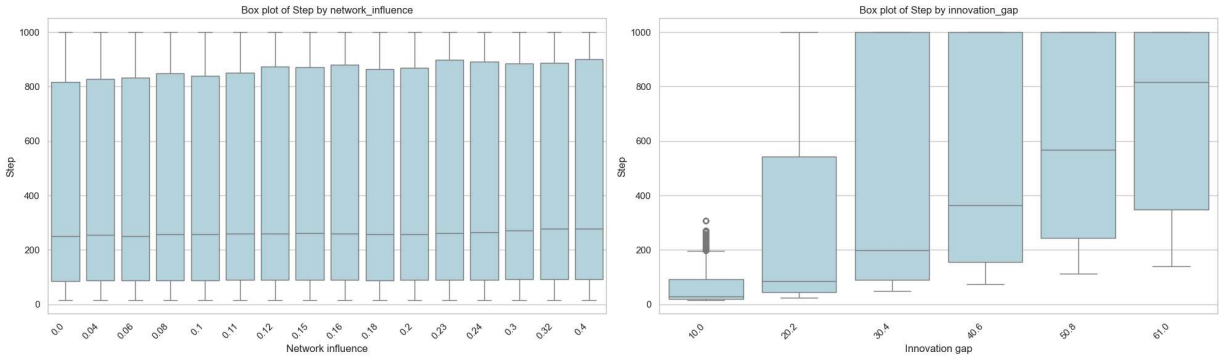
3. Box Plots by Steps and TAR

By Steps

Network Influence: The network_influence parameter does not seem to obviously affect the simulation steps, as indicated by the similar distributions across its values. This could mean that within the studied range, network_influence does not play a critical role in determining the time to reach the event of interest.

Innovation Gap: In contrast, innovation_gap has a significant impact on the number of steps. Higher values of innovation_gap lead to more steps, indicating that a larger gap between the firm's TAR and the market median TAR prolongs the process. This could imply that firms with greater abilty to catch up and greater tolerance with innovation gap are more likely to keep innovating and stay in the market.

In [ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

columns_to_plot = ['network_influence', 'innovation_gap']

# Set the Seaborn style for better aesthetics
sns.set(style="whitegrid")


plt.figure(figsize=(20, 6))

for idx, column in enumerate(columns_to_plot, 1):
    plt.subplot(1, 2, idx)

    if column == 'network_influence':
        # Round the network_influence values for better readability
        df[column] = df[column].round(2)

    sns.boxplot(x=column, y='Step', data=df, color='lightblue')
    plt.title(f'Box plot of Step by {column}')
    plt.xlabel(column.replace('_', ' ').capitalize())
    plt.ylabel('Step')

    plt.xticks(rotation=45, ha='right')

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

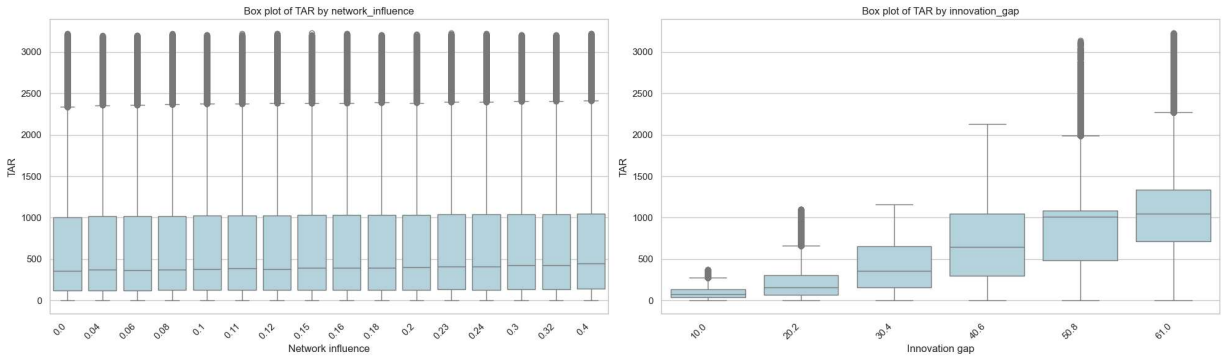


By TAR

Network Influence: From box plot of TAR by network influence, we can see there is no obvious effect on the TAR, as indicated by the similar distributions across its values.

Innovation Gap: There is a clear positive relationship between the Innovation Gap and the number of steps. Larger gaps lead to TAR and greater variability, suggesting that firms with higher innovation gaps allows firms more time to adjust and catch up, resulting in longer and more diverse simulation outcomes.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

columns_to_plot = ['network_influence', 'innovation_gap']

# Set the Seaborn style for better aesthetics
sns.set(style="whitegrid")


plt.figure(figsize=(20, 6))

for idx, column in enumerate(columns_to_plot, 1):
    plt.subplot(1, 2, idx)

    if column == 'network_influence':
        # Round the network_influence values for better readability
        df[column] = df[column].round(2)

    sns.boxplot(x=column, y='TAR', data=df, color='lightblue')
    plt.title(f'Box plot of TAR by {column}')
    plt.xlabel(column.replace('_', ' ').capitalize())
    plt.ylabel('TAR')

    plt.xticks(rotation=45, ha='right')

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



4. Regression Analysis

In the previous plots, I observed that there is no obvious impact of network influence on the simulation outcomes. It is possible that the effect of network influence is being confounded by other factors within the model. To better understand the impact of network influence on the Total Aggregate Revenue (TAR), I conducted a regression analysis.

By regressing the natural logarithm of TAR (lnTAR) on network influence while controlling for other relevant factors, I can isolate the effect of network influence. This approach allows me to account for the multiplicative nature of the relationships in the data and mitigate the impact of outliers or skewed distributions. The control variables included in the regression are innovation gap, tar gain, success probability adjustment, and simulation step. This analysis aims to provide a clearer understanding of how network influence affects TAR when other variables are held constant.

I aggregate the data by RunId. This is because TAR varies for each firm, but other variables, such as network influence, innovation gap, tar gain, success probability adjustment, and the simulation step, are collected at the end of the simulation and are constant for all firms within a given simulation run.

By grouping the data by RunId and aggregating the relevant variables, I can summarize the total TAR on market level, which provides a clearer picture of the overall effects.

In [ ]:
```
# Aggregate the total TAR for each simulation run
aggregated_data = df.groupby('RunId').agg({
    'network_influence': 'mean',
    'innovation_gap': 'mean',
    'tar_gain': 'mean',
    'success_prob_adjustment': 'mean',
    'TAR': 'sum',
    'Step': 'mean',
        'iteration':'mean'
}).reset_index()

# Renaming the columns for clarity
aggregated_data.rename(columns={'TAR': 'Total_TAR', 'Step': 'Step_count'}, inplace=
aggregated_data['lnTAR']=np.log(aggregated_data['Total_TAR'])
```

In [ ]:
```
import statsmodels.api as sm

# Select relevant features and the outcome variable
features = ['network_influence', 'innovation_gap', 'tar_gain','Step_count' ,'succes
X = aggregated_data[features]
y = aggregated_data['lnTAR']

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit(cov_type='cluster', cov_kwds={'groups': aggregated_data['i
```

```
# Print the summary of the model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   lnTAR   R-squared:                       0.934
Model:                             OLS   Adj. R-squared:                  0.934
Method:                  Least Squares   F-statistic:                 2.861e+06
Date:                 Wed, 22 May 2024   Prob (F-statistic):          1.23e-132
Time:                         18:02:35   Log-Likelihood:                734.25
No. Observations:                54000   AIC:                            -1457.
Df Residuals:                    53994   BIC:                            -1403.
Df Model:                            5
Covariance Type:              cluster
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          8.0914      0.002   4025.192      0.000       8.087       8.095
network_influence     0.1907      0.003     62.431      0.000       0.185       0.197
innovation_gap     0.0358   4.39e-05    815.803      0.000       0.036       0.036
tar_gain       0.0116      0.000     42.865      0.000       0.011       0.012
Step_count     0.0011   3.24e-06    325.149      0.000       0.001       0.001
success_prob_adjustment    -0.0024      0.008     -0.314      0.754      -0.017       0.012
==============================================================================
Omnibus:                     1619.305   Durbin-Watson:                   0.791
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1385.371
Skew:                          -0.327   Prob(JB):                    1.48e-301
Kurtosis:                       2.567   Cond. No.                     1.08e+04
==============================================================================

Notes:
[1] Standard Errors are robust to cluster correlation (cluster)
[2] The condition number is large, 1.08e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation of Regression Results:

Network Influence: The coefficient for network influence is 0.1907, which is positive and highly significant (p-value = 0.000). This indicates that for each unit increase in network influence, lnTAR increases by approximately 0.1907 units, holding other factors constant. This result is consistent with the expectation that network influence positively impacts technological advancement and innovation. The positive sign makes sense as network influence can facilitate the copying and sharing of technology among firms, leading to higher success probabilities in innovation and, consequently, greater technological advancements.

Innovation Gap, TAR Gain, Step Count: As expected, they all have positive significant coefficient, indicating that they are sinificantly contributing to tech advancement in the simulation.

Success Probability Adjustment: The coefficient for success probability adjustment is -0.0024, which is not statistically significant (p-value = 0.754). This indicates that variations in the adjustment of success probability do not have a significant impact on lnTAR in this model. This make sense as this adjustment is postive when firm successfully innovate but negative when fail to innovate.