Amartya Kumar Sinha

Working document for Assignment 3 of International Climate Policy

February 11, 2024

CNETID: **amartyaksinha**

In [ ]:
```
import os
import kaleido
import pandas as pd
import plotly.io as pio
import plotly.express as px
import plotly.graph_objects as go
from sympy import symbols, Eq, solve, diff
pio.renderers.default = "notebook+pdf"

path = r'C:/Users/amart/OneDrive - The University of Chicago/IntlClimatePolicy_PPHA3993
```

I. Adaptation policy analysis

Creating visualizations of tables

In [ ]:
```
# Define the data from tables 1-5 as a dictionary
data = {
    'Comparison Point': [
        'Average Temperature', 'Minimum Temperature', 'Maximum Temperature',
        'Bias Adjusted TX35', 'Total Precipitation', 'Maximum 5-day Precipitation',
        'Consecutive Dry Days', 'Surface PM2.5'
    ],
    'Annual South Asia': [1.5, 1.8, 1.3, 23.6, 6.7, 10.2, 2.1, 10.1],
    'Annual Tibetan Plateau': [2.3, 2.5, 2.3, 2, 7.2, 10.3, -1.9, 2.5],
    'May-July South Asia': [1.4, 1.7, 1.3, 17.2, 6, 20.4, None, 10.4],
    'May-July Tibetan Plateau': [2.1, 2.4, 2.1, 0.8, 11, 19.8, None, 1.9],
    'Oct-Dec South Asia': [1.5, 1.9, 1.3, 4.5, 3, 24.7, None, 16.9],
    'Oct-Dec Tibetan Plateau': [2.5, 2.7, 2.5, 0.2, -3.2, 21.6, None, 3.5],
    'Mar-May South Asia': [1.5, 1.8, 1.4, 8.3, 2.9, 4.3, None, 7.8],
    'Mar-May Tibetan Plateau': [2.1, 2.4, 2.1, 0.8, 8.6, 9.3, None, 2.1],
    'Dec-Mar South Asia': [1.5, 1.8, 1.5, 5.6, -9, 30.8, None, 7.8],
    'Dec-Mar Tibetan Plateau': [2.4, 2.5, 2.4, 0.1, -0.5, 28.4, None, 2.5],
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Melt the DataFrame to make it suitable for Plotly
df_melted = df.melt(id_vars=['Comparison Point'], var_name='Category', value_name='Valu

display(df_melted)
```

	Comparison Point	Category	Value
0	Average Temperature	Annual South Asia	1.5
1	Minimum Temperature	Annual South Asia	1.8
2	Maximum Temperature	Annual South Asia	1.3
3	Bias Adjusted TX35	Annual South Asia	23.6
4	Total Precipitation	Annual South Asia	6.7
...	...	...	...
75	Bias Adjusted TX35	Dec-Mar Tibetan Plateau	0.1
76	Total Precipitation	Dec-Mar Tibetan Plateau	-0.5
77	Maximum 5-day Precipitation	Dec-Mar Tibetan Plateau	28.4
78	Consecutive Dry Days	Dec-Mar Tibetan Plateau	NaN
79	Surface PM2.5	Dec-Mar Tibetan Plateau	2.5

80 rows × 3 columns

In [ ]:

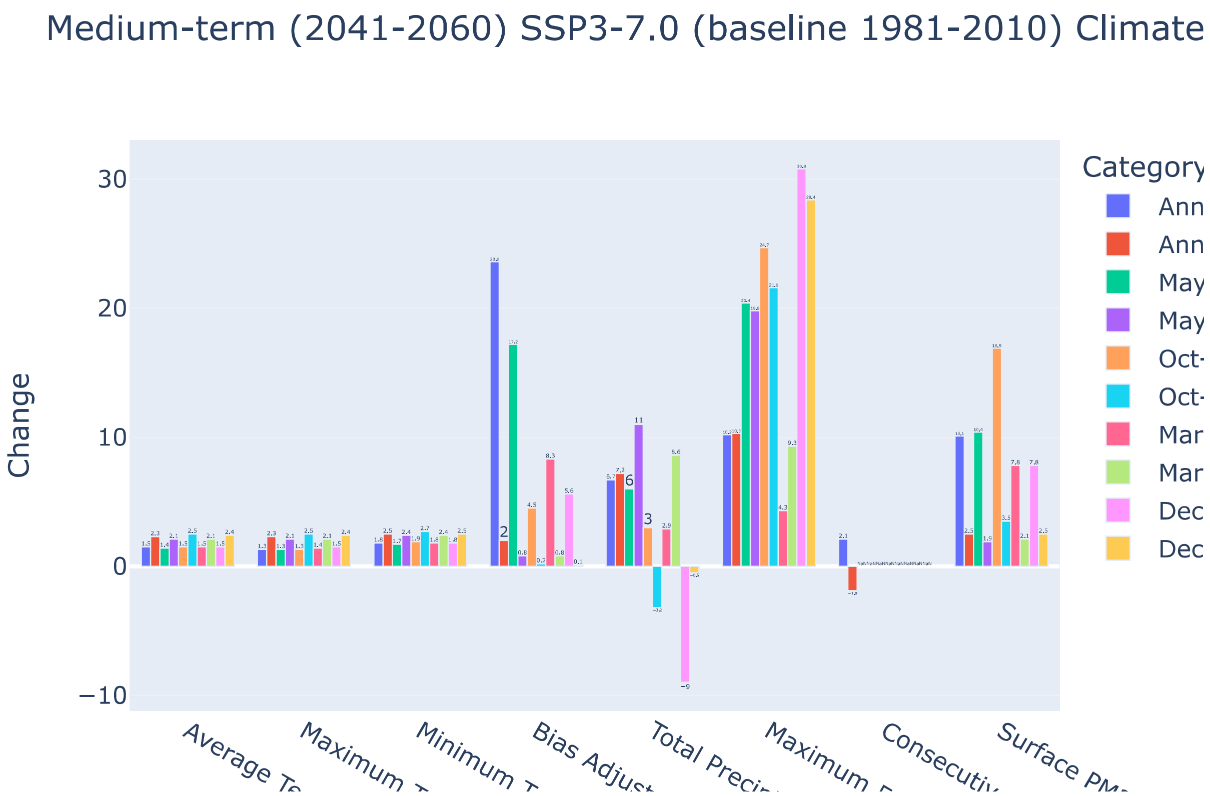
```
# Interactive Bar Graph
fig = px.bar(df_melted, x='Comparison Point', y='Value',
             color='Category', barmode='group',
             title='Medium-term (2041-2060) SSP3-7.0 (baseline 1981-2010) Climate Proje
             labels={'Value': 'Change', 'Comparison Point': 'Metric'})

order = ['Average Temperature', 'Maximum Temperature', 'Minimum Temperature',
         'Bias Adjusted TX35', 'Total Precipitation', 'Maximum 5-day Precipitation',
         'Consecutive Dry Days', 'Surface PM2.5']

fig.update_layout(xaxis={'categoryorder':'array', 'categoryarray': order})

fig.update_traces(texttemplate='%{y}', textposition='outside')

fig.show()
```



II. Mitigation Policy Copmarison

1. Command-and-control

In [ ]:
```
x = symbols('x')
A_values = [10, 20, 30, 40]
optimal_production_levels = {}

for A in A_values:
    # Define the profit function
    profit = A*x - x**2

    # Calculate the derivative of the profit function
    profit_prime = diff(profit, x)

    # Set the derivative to zero to find critical points
    critical_points = solve(Eq(profit_prime, 0), x)

    # Check which critical points are within the allowed range [0, 10]
    valid_points = [point.evalf() for point in critical_points if 0 <= point.evalf() <=

    if 10 not in valid_points and profit.subs(x, 10) > profit.subs(x, max(valid_points,
        optimal_production_levels[A] = 10
    else:
        optimal_production_levels[A] = max(valid_points, key=lambda point: profit.subs(
```

```
optimal_production_levels
```

Out[ ]: {10: 5.00000000000000, 20: 10.0000000000000, 30: 10, 40: 10}

2. Tax: Price control

In [ ]:
```
t = 10
optimal_levels_with_tax = {}

for A in A_values:
    # Profit function now includes the tax term
    profit_function_with_tax = A*x - x**2 - t*x
    derivative_profit_with_tax = diff(profit_function_with_tax, x)

    critical_points_with_tax = solve(derivative_profit_with_tax, x)

    # Since the problem is constrained to x ≥ 0, we take the positive critical point
    optimal_x_with_tax = max(critical_points_with_tax, key=lambda point: profit_functio

    optimal_levels_with_tax[A] = optimal_x_with_tax

optimal_levels_with_tax
```

Out[ ]: {10: 0, 20: 5, 30: 10, 40: 15}

3. Market for permits: Quantity control

In [ ]:
```
p = 10
N = 10
optimal_levels_with_permits = {}

for A in A_values:
    # Profit function now includes the permits trading term
    profit_function_with_permits = A*x - x**2 + p*(N - x)
    derivative_profit_with_permits = diff(profit_function_with_permits, x)

    critical_points_with_permits = solve(derivative_profit_with_permits, x)

    optimal_x_with_permits = max(critical_points_with_permits, key=lambda point: profit

    optimal_levels_with_permits[A] = optimal_x_with_permits

optimal_levels_with_permits
```

Out[ ]: {10: 0, 20: 5, 30: 10, 40: 15}

4. Examining reduced permit price

In [ ]:
```
p = 5
N = 10
optimal_levels_with_permits = {}

for A in A_values:
    # Profit function now includes the permits trading term
    profit_function_with_permits = A*x - x**2 + p*(N - x)
```

```python
    derivative_profit_with_permits = diff(profit_function_with_permits, x)

    critical_points_with_permits = solve(derivative_profit_with_permits, x)

    optimal_x_with_permits = max(critical_points_with_permits, key=lambda point: profit

    optimal_levels_with_permits[A] = optimal_x_with_permits

optimal_levels_with_permits
```

Out[ ]: {10: 5/2, 20: 15/2, 30: 25/2, 40: 35/2}