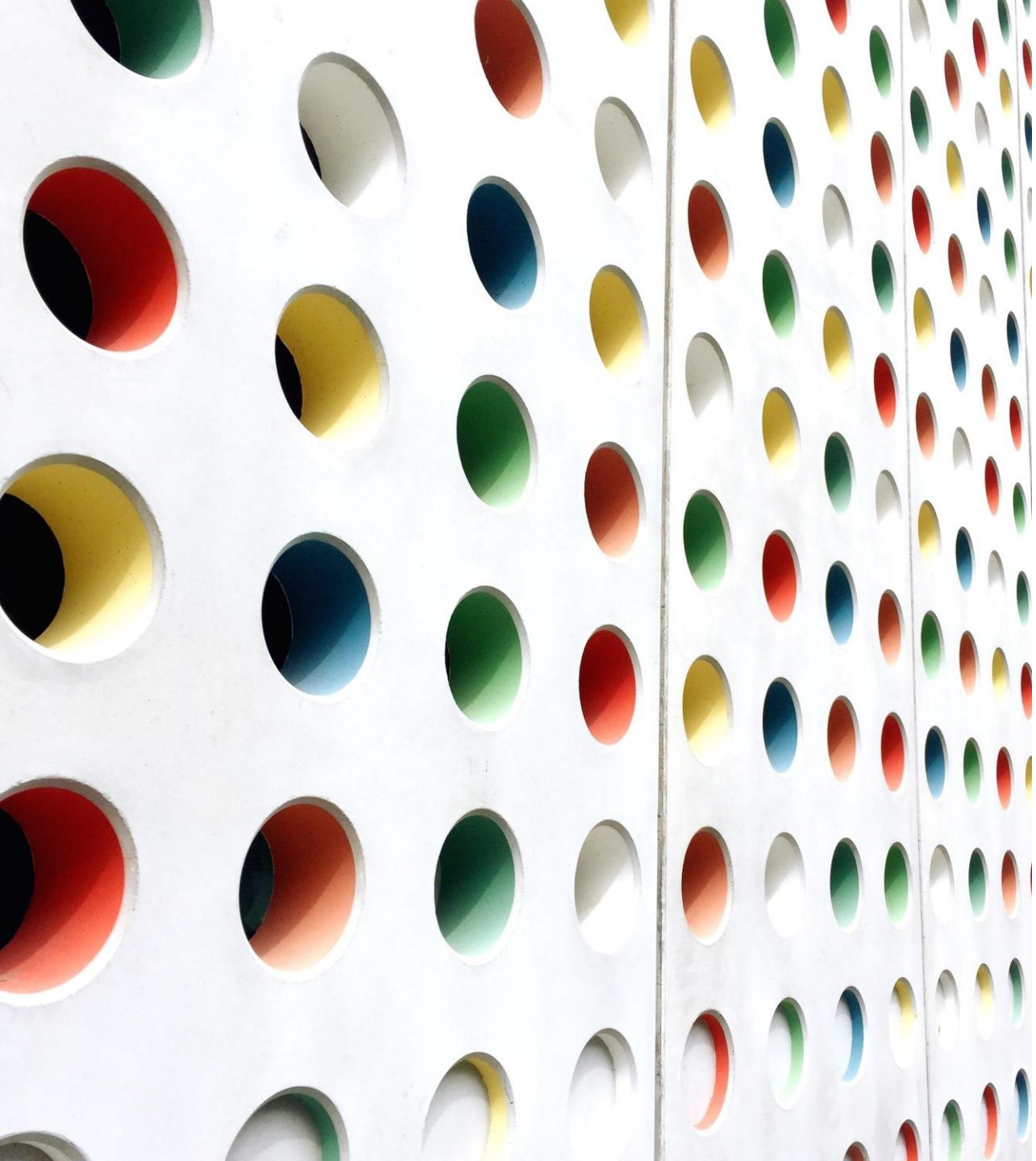


Sustainable Income Growth Rate

PRESENTED BY – NIKHIL KAROL
SAHIB GANDHI
VENKATA ADITYA REDDY CHOPPA
PRERNA MEHTA

SUSTAINABLE INCOME GROWTH RATE

Through our analysis of Household Economic Data of Canadaian households over the past 2 Decades, We made an attempt to define and estimate **Sustainable Income Growth Rate** , an annual rate at which a typical household should expect its income to grow given an anticipated rate of Inflation and historical trends of change in disposable income and household consumption expenditure.

HOUSEHOLD ECONOMIC DATA : ANALYSIS OF INCOME, INFLATION AND EXPENDITURE

Analysing historical trends in Disposable Income and Consumption Expenditure

```
# Import necessary libraries
import numpy as np
import pandas as pd
import hvplot.pandas
import seaborn as sns
import os
```

READING THE FILE

```
household_df = pd.read_csv("./Resources/canada_household_econimic_data.csv", index_col = 'Income, consumption and savings')
display(household_df.head())
display(household_df.tail())
```

Income, consumption and savings	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	...	2013	2014	2015
Household disposable income	615,286	642,584	663,239	695,238	721,510	774,375	816,321	862,404	884,783	928,091	...	1,046,791	1,081,654	1,129,875
Compensation of employees	574,278	596,782	620,600	656,154	692,609	737,321	782,564	818,108	810,556	835,940	...	959,181	996,247	1,024,742

Income, consumption and savings	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	...	2013	2014	1
Social transfers in kind (STiK): health	73,275	77,820	83,061	86,672	90,590	96,801	104,198	111,759	118,734	124,275	...	139,633	144,998	
Social transfers in kind (STiK): other	32,556	35,055	36,580	37,853	40,287	42,756	44,656	48,565	48,630	49,302	...	54,285	54,591	

CONVERTING STRING TO FLOAT TO COMBINE NUMERICAL DATA

```
# Define a function to remove commas and convert to float
def convert_to_float(value):
    return float(str(value).replace(',', ''))

# Apply conversion function to each specified column
for column in economic_data_df.columns:
    economic_data_df[column] = economic_data_df[column].apply(convert_to_float)

# Print the DataFrame with the updated columns
display(economic_data_df.head())

# Print the data types of the columns
economic_data_df.dtypes
```

Income, consumption and savings	Adjusted household disposable income	Household final consumption expenditure (HFCE)	Food and non-alcoholic beverages	Alcoholic beverages and tobacco	Clothing and footwear	Housing, water, electricity, gas and other fuels	Furnishings, household equipment and other goods and services related to the dwelling and property	Health	Transport	Communications	R
2001	775472.0	615180.0	59698.0	27713.0	30229.0	139341.0	35912.0	21548.0	90542.0	13672.0	
2002	812280.0	651605.0	62227.0	31002.0	31099.0	145400.0	38516.0	23418.0	97163.0	15134.0	

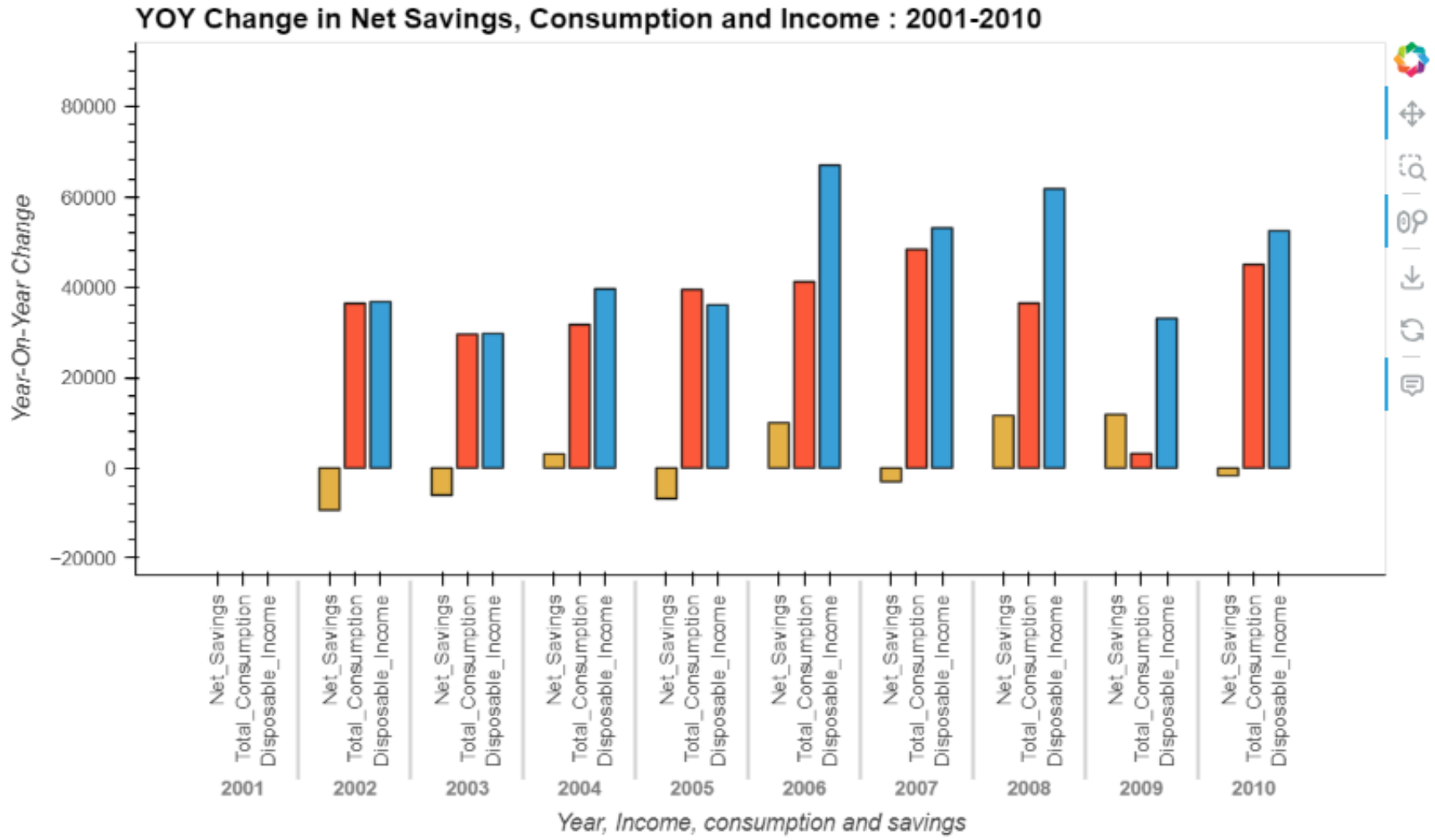
INCOME EXPENDITURE AND CONSUMPTION

```
# Income Expenditure and Consumption

inc_exp_saving_df = economic_data_df[['Disposable_Income','Total_Consumption','Net_Savings']].reset_index()

inc_exp_saving_df.rename(columns={'index':'Year'},inplace=True)

inc_exp_saving_df = inc_exp_saving_df.set_index('Year')

display(inc_exp_saving_df.head())
display(inc_exp_saving_df.tail())
```

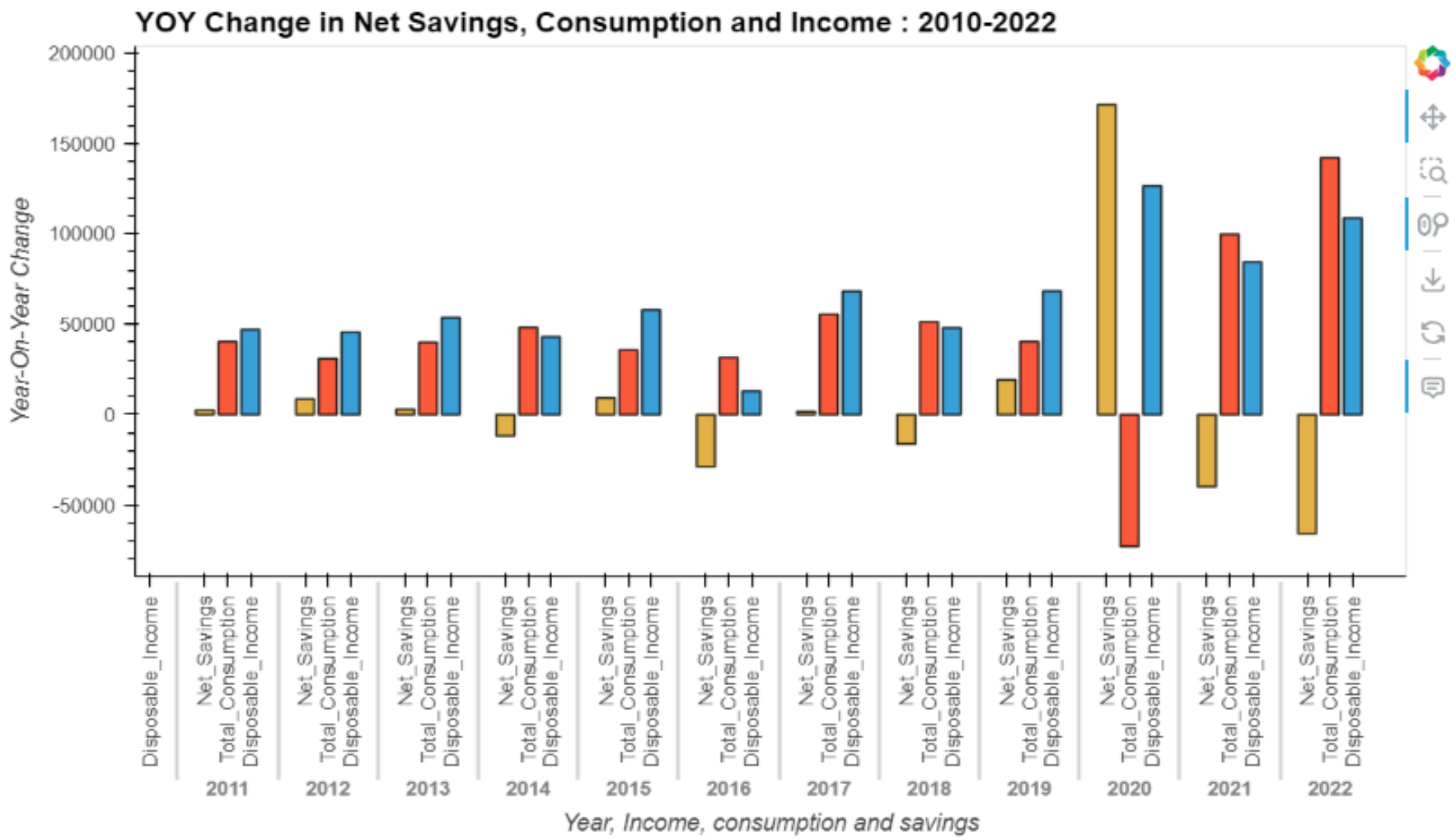
Income, consumption and savings	Disposable_Income	Total_Consumption	Net_Savings
Year			
2001	775472.0	615180.0	29749.0
2002	812280.0	651605.0	20423.0

Income, consumption and savings	Disposable_Income	Total_Consumption	Net_Savings
Year			
2018	1561880.0	1260445.0	7733.0
2019	1630217.0	1300816.0	27022.0

```
# Year-On-Year Change during 2001 to 2010
inc_exp_saving_df.loc['2001':'2010'].diff().hvplot(kind='bar',rot=90, width = 700,
                                                   ylabel = 'Year-On-Year Change',
                                                   frame_width = 700, frame_height = 300,
                                                   title = 'YOY Change in Net Savings, Consumption and Income : 2001-2010')
```



```
#  Year-On-Year Change during 2010 to 2022
inc_exp_saving_df.loc['2010':'2022'].diff().hvplot(kind='bar',rot=90, width = 700,
                                                   ylabel = 'Year-On-Year Change',yformatter = '%.0f',
                                                   frame_width = 700, frame_height = 300,
                                                   title = 'YOY Change in Net Savings, Consumption and Income : 2010-2022')
```



YOY PERCENTAGE CHANGE

```
yoy_pct_change = round(inc_exp_saving_df.pct_change()*100,2)

display(yoy_pct_change.head())
display(yoy_pct_change.tail())
```

Income, consumption and savings	Disposable_Income	Total_Consumption	Net_Savings
Year			
2001	NaN	NaN	NaN
2002	4.75	5.92	-31.35
2003	3.66	4.54	-29.45
2004	4.71	4.66	21.53
2005	4.09	5.54	-38.80

Income, consumption and savings	Disposable_Income	Total_Consumption	Net_Savings
Year			
2018	3.17	4.24	-67.70
2019	4.38	3.20	249.44
2020	7.76	-5.61	634.45
2021	4.81	8.12	-20.09
2022	5.90	10.70	-41.56

UNDERSTANDING THE IMPACT OF INFLATION ON THE HOUSEHOLD ECONOMIC DATA

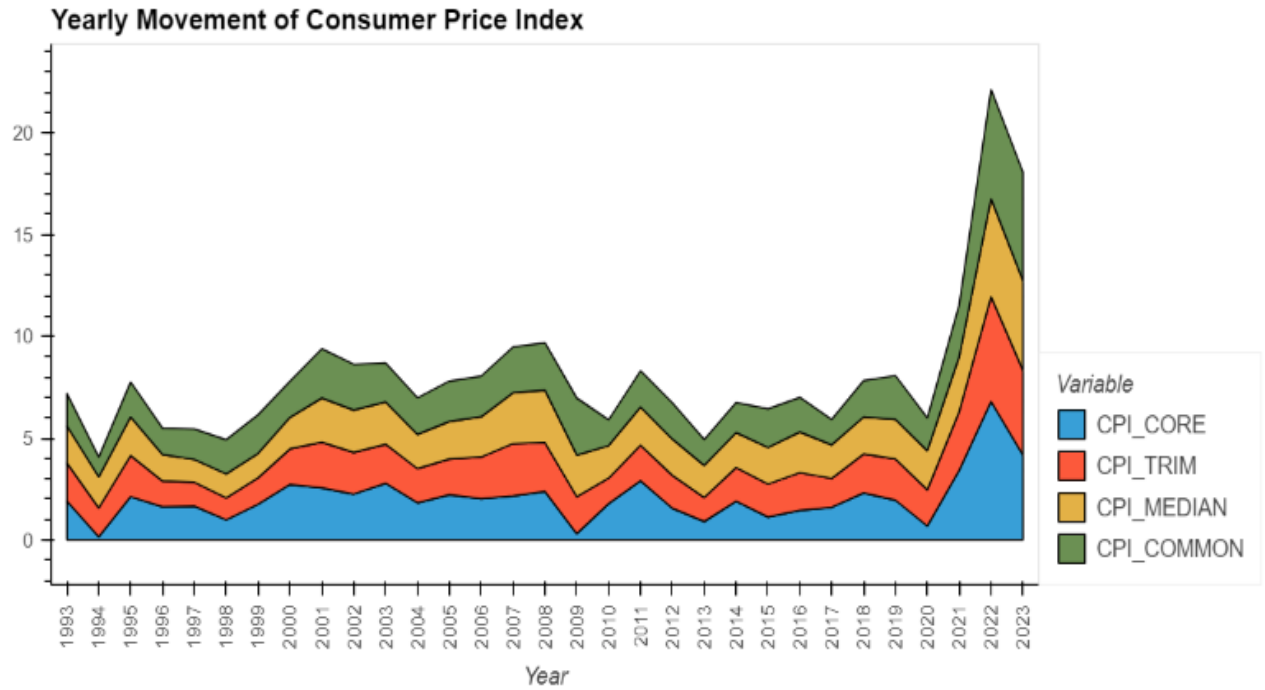
```
# Quarterly CPI Data

cpi_path = 'canada_CPI.csv'
cpi_df = pd.read_csv( cpi_path, skiprows=1)

cpi_df.head()
```

	date	INDINF_CPI_Q	INDINF_CPI_TRIM_Q	INDINF_CPI_MEDIAN_Q	INDINF_CPI_COMMON_Q
0	1993Q1	2.2	2.0	2.0	1.8
1	1993Q2	1.8	1.9	1.9	1.7
2	1993Q3	1.8	1.8	1.7	1.4
3	1993Q4	1.8	1.8	1.7	1.5
4	1994Q1	0.5	1.5	1.6	1.3

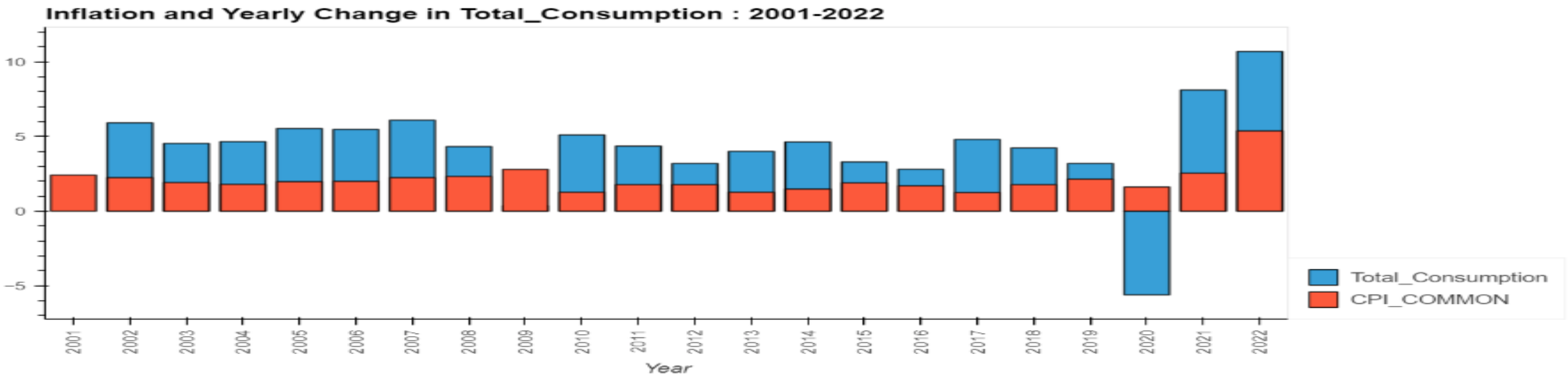
CALCULATING APPROXIMATE ANNUAL INFLATION AS MEAN OF QUARTERLY CPI

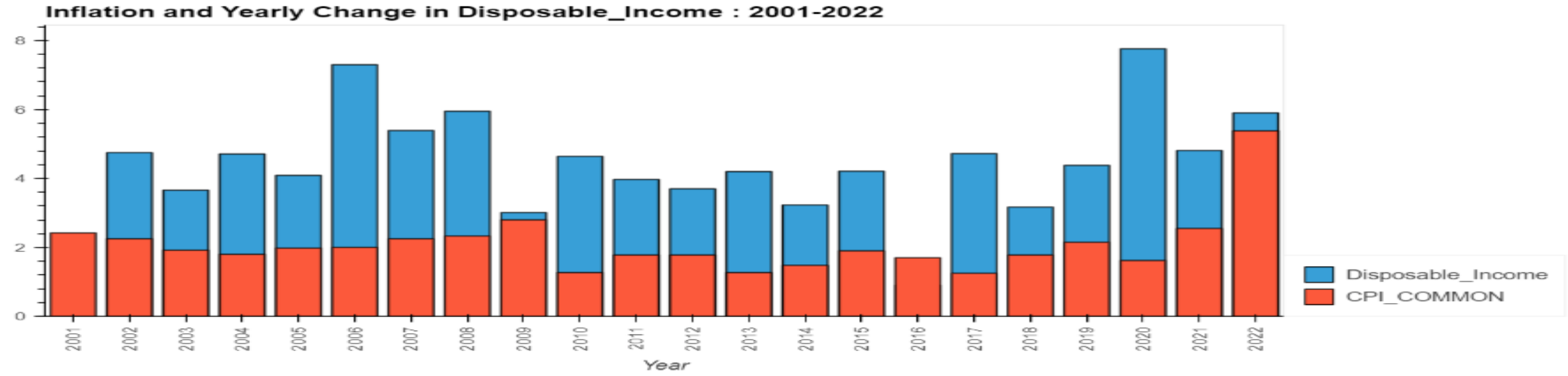
```
cpi_annual_df = round(cpi_df.groupby('Year').mean(),2)

display(cpi_annual_df.tail())

cpi_annual_df.hvplot.area(title='Yearly Movement of Consumer Price Index', rot = 90,frame_width = 600,
                          frame_height = 300)
```

Year	CPI_CORE	CPI_TRIM	CPI_MEDIAN	CPI_COMMON
2019	1.95	2.02	1.95	2.15
2020	0.68	1.78	1.92	1.62
2021	3.42	2.88	2.70	2.55
2022	6.80	5.15	4.82	5.38
2023	4.17	4.17	4.40	5.33



VISUALIZING THE IMPACT OF INFLATION ON TOTAL_CONSUMPTION & DISPOSABLE INCOME





```python
# Modeling Linear Regression fit for Target Income Investment Growth Rate
# based on Inflation and Historical Household Income & Consumption trends

import statsmodels.api as sm

inflation_X = sm.add_constant(target_growth_rate['Inflation'])
investment_growth_Y = target_growth_rate['Sustainable_Income_Growth']

model = sm.OLS(investment_growth_Y, inflation_X)
results = model.fit()

print(results.summary())
```

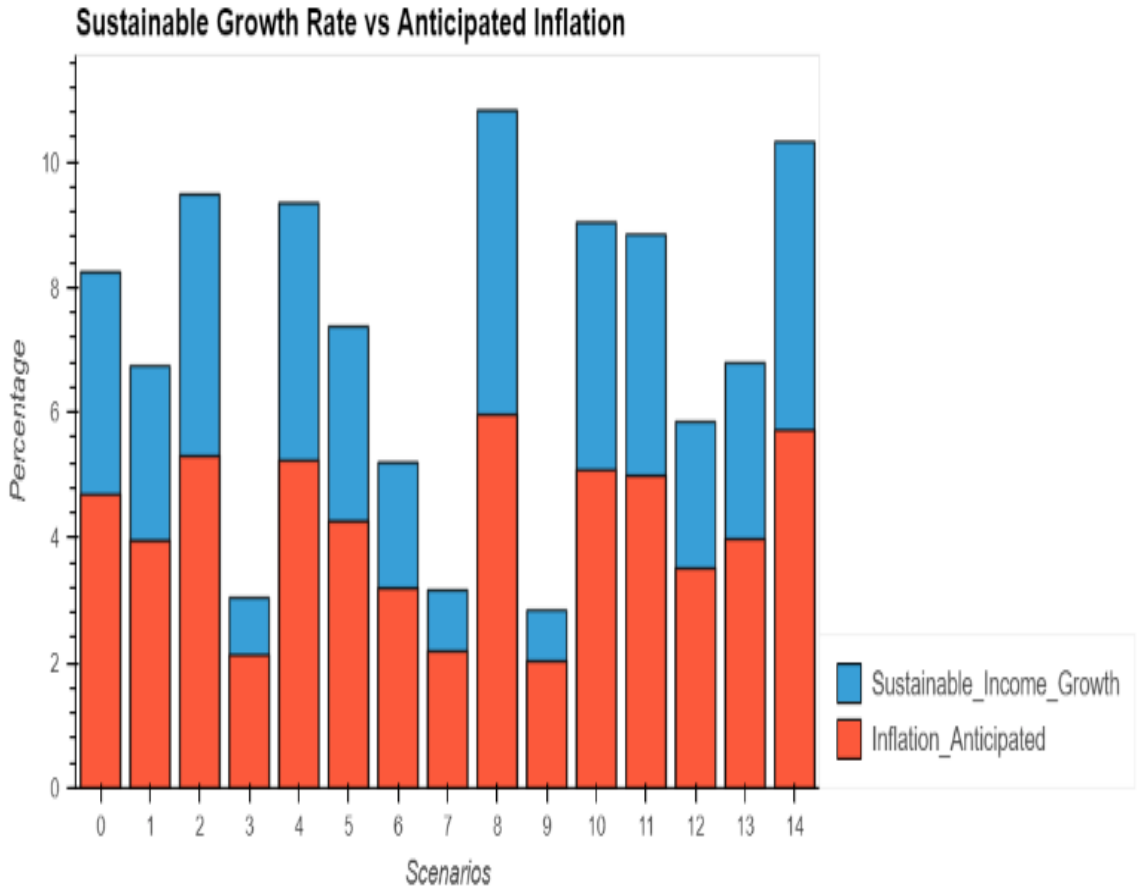
```
                                OLS Regression Results
==============================================================================
Dep. Variable:     Sustainable_Income_Growth   R-squared:                       0.801
Model:                                 OLS   Adj. R-squared:                  0.790
Method:                      Least Squares   F-statistic:                     76.39
Date:                     Sat, 27 Jan 2024   Prob (F-statistic):           4.40e-08
Time:                             19:55:58   Log-Likelihood:                -26.445
No. Observations:                       21   AIC:                             56.89
Df Residuals:                           19   BIC:                             58.98
Df Model:                                1
Covariance Type:                 nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.2802      0.517     -2.478      0.023      -2.362      -0.199
Inflation      2.0302      0.232      8.740      0.000       1.544       2.516
==============================================================================
Omnibus:                        0.952   Durbin-Watson:                   2.076
Prob(Omnibus):                  0.621   Jarque-Bera (JB):                0.602
Skew:                           0.404   Prob(JB):                        0.740
Kurtosis:                       2.813   Cond. No.                         6.92
==============================================================================
```

```python
# Linear Regression Line of Best Fit
print(f'Sustainable_Income_Growth = {round(results.params[1],2)}*Inflation' + f'{round(results.params[0],2)}')
```

```
Sustainable_Income_Growth = 2.03*Inflation-1.28
```

ESTIMATED TARGET INCOME INVESTMENT GROWTH, GIVEN INFLATION

```
np.random.seed(99)

scenario_target_growth = pd.DataFrame(np.random.uniform(2,6,size=(15,1)),columns=['Inflation_Anticipated'])

X = sm.add_constant(scenario_target_growth['Inflation_Anticipated'])

scenario_target_growth['Sustainable_Income_Growth']= round(results.predict(X),2)

plot_1 = scenario_target_growth['Inflation_Anticipated'].hvplot.bar(frame_width=500,frame_height=300)

plot_2 = scenario_target_growth['Sustainable_Income_Growth'].hvplot.bar(frame_width=500,frame_height=300)

target_growth_plot = (plot_2*plot_1).opts(xlabel = 'Scenarios',ylabel='Percentage',
                                          title='Sustainable Growth Rate vs Anticipated Inflation')

display(scenario_target_growth.head())

target_growth_plot
```



ETFS AND TICKER DATA FOR SECTORAL PERFORMANCE ANALYSIS

```
ticker_name_dict = {'XIU':['iShares S&P/TSX 60 Index ETF','TSX_60'],
                    'XST':['iShares S&P/TSX Capped Consumer Staples Index ETF','Consumer_Staples'],
                    'XRE':['iShares S&P/TSX Capped REIT Index ETF','Real_Estate'],
                    'XUT':['iShares S&P/TSX Capped Utilities Index ETF','Utilities'],
                    'XHC':['iShares Global Healthcare Index ETF','Health'],
                    'CEW':['iShares Equal Weight Banc & Lifeco ETF','Financial_Services'],
                    'TRVL':['Harvest Travel & Leisure Index ETF','Travel_Leisure'],
                    'XGD':['iShares S&P/TSX Global Gold Index ETF','Gold']}

ticker_df = pd.DataFrame.from_dict(ticker_name_dict,orient='index',
                                   columns=['ETF','Sector'])

ticker_df
```

	ETF	Sector
XIU	iShares S&P/TSX 60 Index ETF	TSX_60
XST	iShares S&P/TSX Capped Consumer Staples Index ETF	Consumer_Staples
XRE	iShares S&P/TSX Capped REIT Index ETF	Real_Estate
XUT	iShares S&P/TSX Capped Utilities Index ETF	Utilities
XHC	iShares Global Healthcare Index ETF	Health
CEW	iShares Equal Weight Banc & Lifeco ETF	Financial_Services
TRVL	Harvest Travel & Leisure Index ETF	Travel_Leisure
XGD	iShares S&P/TSX Global Gold Index ETF	Gold

USING FUNCTION TO EXTRACT DATA USING APIS AND SAVE AS CSV FILES

```
def extract_api_data(data_list):
    df_list = []
    for item in data_list:
        url = f'https://www.alphavantage.co/query?function=TIME_SERIES_MONTHLY&symbol={item}.TRT&outputsize=full&apikey={ap
        r = requests.get(url)
        df = pd.read_csv(url)
        df['timestamp'] = df['timestamp'].str[:4]                          # Extracting Year from timestamp
        item_df = df.rename(columns={'close': f'{item}','timestamp':'Year'})# Renaming Columns
        item_df = item_df.groupby('Year').mean()                           # Calculating Average Yearly Price
        df_list.append(item_df)

    raw_df = pd.concat(df_list, axis=1)
    return(raw_df)
```

Year	XIU	XST	XRE	XUT	XHC	CEW	TRVL	XGD
2005	57.660909	NaN	12.799045	NaN	NaN	NaN	NaN	51.521818
2006	68.781667	NaN	14.847333	NaN	NaN	NaN	NaN	75.980833
2007	79.343333	NaN	16.218783	NaN	NaN	NaN	NaN	74.636667
2008	54.714167	NaN	12.224200	NaN	NaN	7.15790	NaN	55.285833
2009	15.463333	NaN	9.569833	NaN	NaN	6.38115	NaN	20.005000

Year	XIU	XST	XRE	XUT	XHC	CEW	TRVL	XGD
2020	24.128333	63.537500	15.976717	27.167083	53.970833	11.792233	NaN	20.140833
2021	29.879167	70.055000	19.136183	29.904167	63.690833	15.790767	21.826364	18.213333
2022	30.756667	80.149167	18.015833	30.509167	65.453333	15.635833	18.695833	17.397500
2023	30.694167	86.050833	16.287500	26.570833	65.910000	15.675000	21.104167	17.889167
2024	32.280000	90.880000	15.920000	25.420000	68.290000	16.590000	23.760000	16.320000

ADDING DECSCRIPTION TO TICKER COLUMN NAMES

Year	XIU:TSX_60	XST:Consumer_Staples	XRE:Real_Estate	XUT:Utilities	XHC:Health	CEW:Financial_Services	TRVL:Travel_Leisure	XGD:Gc
2010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2011	6.01	NaN	17.11	NaN	NaN	2.84	NaN	6
2012	-6.91	10.76	12.00	-0.20	11.95	-5.62	NaN	-16
2013	6.07	25.94	-3.22	-4.29	28.29	22.34	NaN	-38
2014	15.53	24.96	0.56	2.36	23.30	17.69	NaN	-13
2015	-1.38	31.95	-1.09	0.80	16.73	-0.78	NaN	-15
2016	-0.85	11.93	-0.67	5.48	-6.57	4.25	NaN	43
2017	10.44	3.45	2.03	6.84	9.41	18.69	NaN	-4
2018	1.43	-0.09	4.93	-8.39	7.87	1.02	NaN	-11
2019	5.40	15.77	12.24	16.79	5.88	1.91	NaN	23
2020	-2.16	1.23	-17.52	10.86	10.33	-10.28	NaN	45
2021	23.83	10.26	19.78	10.07	18.01	33.91	NaN	-9
2022	2.94	14.41	-5.85	2.02	2.77	-0.98	-14.34	-4
2023	-0.20	7.36	-9.59	-12.91	0.70	0.25	12.88	2

```
correlation_df = combined_etf_inflation_df.corr()
display(correlation_df.tail())
sns.heatmap(correlation_df.iloc[:-1,-1:],annot=True,fmt='.2f')
```