Question 1 Outline

This notebook outlines the steps for parts (a) through (g) of Question 1. Each section contains a brief prompt followed by starter code cells.

(a) Lasso for `mret` and OLS with five non-zero topics

> Using the mret (for market return) column from the macro.csv file, fit lasso for a range of penalty parameters to the topics data. Select the penalty that yields five non-zero coefficients. Then run OLS with these five topics. What is the R^2 ? Interpret the topics selected.

In [9]:
```
# (a) Setup: load data and libraries
import pandas as pd
from sklearn.linear_model import LassoCV, Lasso
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression  # added for OLS replacement


# define reusable functions
def load_data(topic_path='topics.csv', macro_path='macro.csv'):
    topics = pd.read_csv(topic_path, index_col=0)
    macro = pd.read_csv(macro_path, index_col=0)
    return topics, macro

def run_lasso_path(X, y, alphas):
    nnz, models = [], []
    for alpha in alphas:
        l = Lasso(alpha=alpha, max_iter=10000)
        l.fit(X, y)
        nnz.append((l.coef_ != 0).sum())
        models.append(l)
    return nnz, models

def find_alpha_for_nnz(nnz, alphas, target=5):
    idxs = [i for i, n in enumerate(nnz) if n == target]
    if not idxs:
        raise ValueError(f"No alpha yields {target} non-zero coefficients.")
    idx = idxs[0]
    return alphas[idx], idx

def find_alpha_binary(X, y, target=5, min_exp=-6.5, max_exp=-3.5, tol=1e-4, max_iter=40):
    lower, upper = 10**min_exp, 10**max_exp
    for _ in range(max_iter):
        mid = (lower * upper) ** 0.5
        l = Lasso(alpha=mid, max_iter=10000).fit(X, y)
        nnz = (l.coef_ != 0).sum()
        if nnz > target:
            lower = mid
        elif nnz < target:
            upper = mid
        else:
            return mid
    return mid

def get_selected_features(model, feature_names):
    return feature_names[model.coef_ != 0]

def run_ols_and_r2(X, y):
    model = LinearRegression().fit(X, y)
    return model, model.score(X, y)

# Load data
topics, macro = load_data()

# display dataset heads
print("Topics")
display(topics.describe())
print("Macro")
display(macro.describe())

y = macro['mret']
X = topics

# Ensure X and y have overlapping indices
common_idx = X.index.intersection(y.index)
y = y.loc[common_idx]
X = X.loc[common_idx]

# Perform train-test split
split_idx = int(len(y) * 0.8)  # 80% train, 20% test
train_idx, test_idx = y.index[:split_idx], y.index[split_idx:]
y_train, y_test = y.loc[train_idx], y.loc[test_idx]
```

```python
X_train, X_test = X.loc[train_idx], X.loc[test_idx]

min_exp, max_exp = -6.5, -3.5  # exponent bounds for alpha=10**exp

# compute sparsity path via binary search
max_nnz = X.shape[1]
alphas_bs = []
nnz_bs = []
for target in range(1, 50):
    try:
        alpha_t = find_alpha_binary(X, y, target=target, min_exp=min_exp, max_exp=max_exp)
        alphas_bs.append(alpha_t)
        nnz_bs.append(target)
    except ValueError:
        pass

# plot binary-search sparsity path

alpha5 = find_alpha_binary(X_train, y_train, target=5, min_exp=min_exp, max_exp=max_exp)
model5 = Lasso(alpha=alpha5, max_iter=10000).fit(X_train, y_train)
selected = get_selected_features(model5, X_train.columns)
# Fit linear regression on selected train data and get in-sample R²
model_ols, r2_train = run_ols_and_r2(X_train[selected], y_train)
# Compute out-of-sample R² using the trained OLS model
r2_test = model_ols.score(X_test[selected], y_test)

plt.figure(figsize=(8,4))
plt.plot(alphas_bs, nnz_bs, marker='o')
plt.xscale('log')
plt.xlabel('alpha')
plt.ylabel('non-zero coefficients')
plt.title('Binary-search Lasso sparsity path for mret')
plt.show()
print(f"Alpha with 5 non-zero coefficients: {alpha5}\n")
print("Selected topics:", list(selected))
print(f"In-sample R-squared: {r2_train:.4f}")
print(f"Out-of-sample R-squared: {r2_test:.4f}")
```

Topics

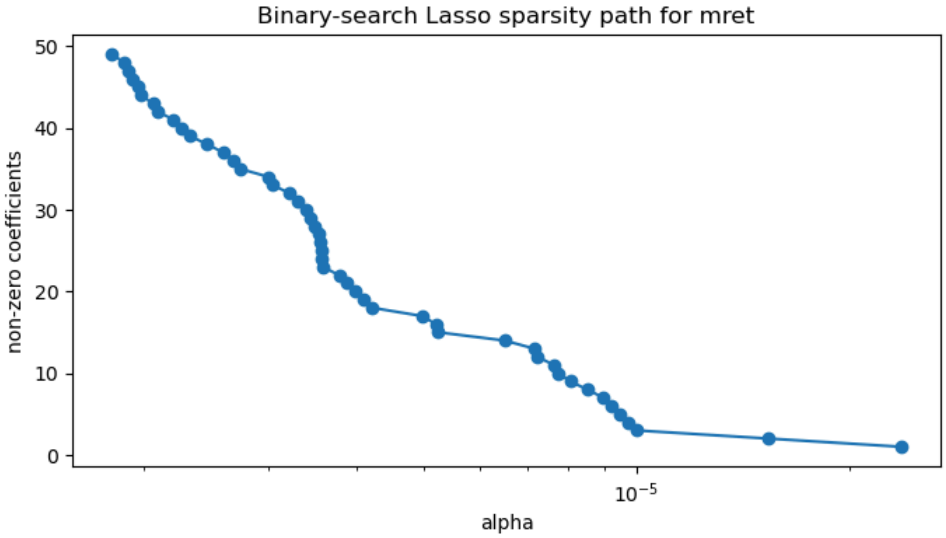
	Natural disasters	Internet	Soft drinks	Mobile devices	Profits	M&A	Changes	Police/crime	Research	Executive pay	...	European politics	Size	
count	402.000000	402.000000	402.000000	402.000000	402.000000	402.000000	402.000000	402.000000	402.000000	402.000000	...	402.000000	402.000000	402
mean	0.005643	0.006742	0.004111	0.005085	0.006347	0.005167	0.004893	0.006631	0.005062	0.004618	...	0.008523	0.005126	0
std	0.003141	0.004806	0.000669	0.003626	0.002424	0.001422	0.000842	0.003086	0.000720	0.001437	...	0.002859	0.000736	0
min	0.002469	0.000883	0.002527	0.001309	0.001858	0.002948	0.002953	0.002722	0.003421	0.002852	...	0.003965	0.003205	0
25%	0.004014	0.001260	0.003618	0.002034	0.004454	0.004175	0.004223	0.004562	0.004551	0.003601	...	0.006429	0.004539	0
50%	0.004650	0.007077	0.004011	0.004007	0.006011	0.004804	0.004897	0.005298	0.004964	0.004151	...	0.007724	0.005181	0
75%	0.006457	0.010746	0.004581	0.007206	0.008092	0.005861	0.005550	0.007933	0.005516	0.005163	...	0.010196	0.005636	0
max	0.046310	0.022412	0.006500	0.016698	0.014725	0.009903	0.006767	0.021215	0.007120	0.011017	...	0.020162	0.007116	0

8 rows × 180 columns

Macro

	vol	mret	indpro	indprol1	Agric_vol	Food_vol	Soda_vol	Beer_vol	Smoke_vol	Toys_vol	...	Boxes_vol	Trans_vol	Whl
count	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	...	444.000000	444.000000	444.0
mean	-2.002548	0.009959	0.001427	0.001402	-0.064404	0.220549	0.067294	0.019036	0.103020	0.021602	...	0.077420	0.016718	-0.1
std	0.446540	0.044050	0.009981	0.009949	0.807006	0.533525	0.748195	0.496363	0.648011	0.482159	...	0.553324	0.424029	0.3
min	-2.899441	-0.226844	-0.143656	-0.143656	-2.770550	-0.911250	-1.499983	-1.506372	-1.232753	-0.869406	...	-1.663027	-0.984501	-0.9
25%	-2.308861	-0.015274	-0.001721	-0.001721	-0.618365	-0.152218	-0.489776	-0.259147	-0.392418	-0.348695	...	-0.283901	-0.247010	-0.4
50%	-2.078115	0.014378	0.002115	0.002115	-0.068246	0.137506	0.015400	0.013587	0.068764	-0.049276	...	0.006124	0.025805	-0.2
75%	-1.743734	0.038414	0.005348	0.005348	0.448401	0.554391	0.585416	0.329703	0.502695	0.342609	...	0.421119	0.254431	0.0
max	-0.375508	0.128439	0.062986	0.062986	2.316249	2.694422	2.195218	1.967359	1.944101	1.484560	...	2.423346	1.578573	1.3

8 rows × 53 columns




```
Alpha with 5 non-zero coefficients: 1.0554496008786032e-05

Selected topics: ['Small caps', 'Recession', 'Accounting', 'Bear/bull market', 'Elections']
In-sample R-squared: 0.1146
Out-of-sample R-squared: -0.5361
```

(b) Repeat for other outcome variables

> Repeat this procedure for `vol`, `indpro`, `indprol1` (industrial production growth one period in the future), and each of the `indvol` columns. Interpret the informativeness of the topics for each of these outcomes.

OOS `$R^2$` are largely negative when trying to predict most outcomes except:

Outcome	R2_in_sample	R2_out_of_sample	Selected Topics
`Oil_vol`	0.457	0.239	`Treasury bonds`, `Financial crisis`, `International exchanges`, `Oil market`, `Bush/Obama/Trump`
`Aero_vol`	0.259	0.05	`Small caps`, `Treasury bonds`, `Reagan`, `Bush/Obama/Trump`, `Elections`
`Whlsl_vol`	0.465	0.048	`Small caps`, `Financial crisis`, `International exchanges`, `Reagan`, `Private equity/hedge funds`
`indprol1`	0.277	0.028	`Health insurance`, `Recession`, `Clintons`, `Oil market`, `Elections`
`Other_vol`	0.389	0.012	`Internet`, `Treasury bonds`, `International exchanges`, `Elections`, `Private equity/hedge funds`

This result is interesting given that same-period prediction for industrial production growth showed negative R^2

In [2]:
```python
# (b) Loop over outcomes

outcomes = ['vol', 'indpro', 'indprol1'] + [col for col in macro.columns if col.endswith('_vol')]
results = []
for outcome in outcomes:
  # Subset outcome and form train/test
  y_out = macro[outcome].loc[common_idx]
  y_train_out, y_test_out = y_out.loc[train_idx], y_out.loc[test_idx]
  # Find alpha via binary search on training data
  alpha5_out = find_alpha_binary(X_train, y_train_out, target=5, min_exp=min_exp, max_exp=max_exp)
  # Fit Lasso on train, select topics
  model5_out = Lasso(alpha=alpha5_out, max_iter=10000).fit(X_train, y_train_out)
  selected_out = get_selected_features(model5_out, X_train.columns)
  # Fit OLS on train and compute R-squared in and out of sample
  model_ols_out, r2_in = run_ols_and_r2(X_train[selected_out], y_train_out)
  r2_out = model_ols_out.score(X_test[selected_out], y_test_out)
  # Append results with both metrics
  results.append({
    'Outcome': outcome,
    'R2_in_sample': round(r2_in, 3),
    'R2_out_of_sample': round(r2_out, 3),
    'Selected Topics': list(selected_out),
  })
results_df = pd.DataFrame(results)
display(results_df.sort_values('R2_out_of_sample', ascending=False))
```

	Outcome	R2_in_sample	R2_out_of_sample	Selected Topics
32	Oil_vol	0.457	0.239	[Treasury bonds, Financial crisis, Internation...
26	Aero_vol	0.259	0.050	[Small caps, Treasury bonds, Reagan, Bush/Obam...
44	Whlsl_vol	0.465	0.048	[Small caps, Financial crisis, International e...
2	indprol1	0.277	0.028	[Health insurance, Recession, Clintons, Oil ma...
51	Other_vol	0.389	0.012	[Internet, Treasury bonds, International excha...
1	indpro	0.222	-0.019	[Financial crisis, Recession, Clintons, Oil ma...
28	Guns_vol	0.291	-0.042	[US defense, Small caps, International exchang...
23	Mach_vol	0.104	-0.043	[China, Terrorism, Earnings, Private equity/he...
16	Chems_vol	0.350	-0.057	[Internet, Small caps, Treasury bonds, Iraq, N...
8	Toys_vol	0.294	-0.079	[Small caps, International exchanges, Converti...
33	Util_vol	0.399	-0.093	[Small caps, Treasury bonds, Financial crisis,...
47	Banks_vol	0.602	-0.167	[Small caps, Russia, Financial crisis, Interna...
5	Soda_vol	0.407	-0.168	[Treasury bonds, Financial crisis, Internation...
31	Coal_vol	0.545	-0.169	[Treasury bonds, China, Southeast Asia, Reagan...
36	BusSv_vol	0.379	-0.201	[Accounting, Clintons, Mortgages, Reagan, Iraq]
6	Beer_vol	0.438	-0.206	[Profits, Treasury bonds, China, Elections, Pr...
21	Steel_vol	0.470	-0.206	[Profits, China, Clintons, Bear/bull market, P...
12	Clths_vol	0.370	-0.217	[Small caps, Financial crisis, Phone companies...
46	Meals_vol	0.378	-0.248	[Treasury bonds, Financial crisis, Clintons, R...
11	Hshld_vol	0.249	-0.339	[Internet, Small caps, Takeovers, Reagan, Iraq]
0	vol	0.547	-0.356	[Internet, Treasury bonds, Financial crisis, R...
42	Boxes_vol	0.379	-0.403	[Treasury bonds, International exchanges, Reag...
15	Drugs_vol	0.482	-0.408	[Federal Reserve, Health insurance, Treasury b...
14	MedEq_vol	0.291	-0.408	[International exchanges, Convertible/preferre...
38	Softw_vol	0.387	-0.416	[Treasury bonds, International exchanges, Chin...
9	Fun_vol	0.307	-0.417	[Savings & loans, Small caps, European soverei...
50	Fin_vol	0.473	-0.462	[Financial crisis, Clintons, Reagan, Taxes, Pr...
19	BldMt_vol	0.193	-0.463	[Internet, Small caps, Reagan, Iraq, Elections]
20	Cnstr_vol	0.423	-0.488	[Small caps, Treasury bonds, International exc...
40	LabEq_vol	0.692	-0.521	[Small caps, Russia, Health insurance, Financi...
25	Autos_vol	0.270	-0.529	[Internet, Small caps, Treasury bonds, Reagan,...
3	Agric_vol	0.478	-0.540	[Small caps, Financial crisis, Mortgages, Elec...
7	Smoke_vol	0.371	-0.600	[Small caps, Financial crisis, International e...
24	ElcEq_vol	0.353	-0.617	[Treasury bonds, International exchanges, Clin...
43	Trans_vol	0.158	-0.618	[Internet, International exchanges, Terrorism,...
35	PerSv_vol	0.278	-0.761	[Treasury bonds, Financial crisis, Terrorism, ...
41	Paper_vol	0.382	-0.813	[Internet, Small caps, Bush/Obama/Trump, Earni...
34	Telcm_vol	0.326	-0.890	[Natural disasters, Internet, Profits, Small c...
29	Gold_vol	0.190	-1.026	[Small caps, European sovereign debt, Currenci...
10	Books_vol	0.525	-1.121	[Internet, Small caps, Financial crisis, Inter...
45	Rtail_vol	0.199	-1.142	[Small caps, European sovereign debt, Financia...
27	Ships_vol	0.463	-1.250	[Savings & loans, Small caps, Treasury bonds, ...
37	Hardw_vol	0.504	-1.254	[Internet, Small caps, Financial crisis, Inter...
13	Hlth_vol	0.267	-1.330	[Small caps, International exchanges, China, E...
22	FabPr_vol	0.120	-1.606	[Profits, Small caps, Takeovers, Terrorism, Ta...
18	Txtls_vol	0.459	-1.875	[Small caps, Clintons, Terrorism, Bush/Obama/T...
4	Food_vol	0.396	-1.978	[Drexel, Treasury bonds, Financial crisis, Ele...
30	Mines_vol	0.658	-1.998	[Federal Reserve, Small caps, Financial crisis...
17	Rubbr_vol	0.245	-2.008	[Financial crisis, International exchanges, Cl...

	Outcome	R2_in_sample	R2_out_of_sample	Selected Topics
49	RIEst_vol	0.503	-2.122	[Small caps, Financial crisis, International e...
48	Insur_vol	0.449	-2.492	[Financial crisis, International exchanges, Re...
39	Chips_vol	0.658	-9.351	[Internet, Small caps, Financial crisis, Inter...

(c) Real-time forecasting of industrial production growth

> Using what you learned in the first problem set, let's now try our best to forecast industrial production growth in real time. Provide some reasoning for your modeling decisions.

Tried lots of things:

1. **Dimension reduction** via PCA / PLS / Agglomerative Clustering & summing
2. **Non-linearity** - Tried introduction via Gradient Boosted Trees and RBF
3. **Preprocessing** - Started with simple scaling, went down a rabbit hole and ended on needing to use the inverse-softmax to get from the simplex back to the reals. I think there's more here to think about to keep the compositional data linearly independent, which sounds helpful for these linear methods. (CLR vs ILR). Tried different variations of feature selection using lasso regularization then feeding it to more complex models.
4. **Lag data** - Tried to include several lags, but doing so with raw compositional data didn't seem to help. After moving to level space including diffs made the largest impact.

It was difficult to initally move into the positive R^2 with any model, after moving away from the compositional space, we were able to get into higher R^2 in the `0.09` to `0.13` range grabbing between `5` and `20` columns with our last-step lasso.

Here are the 20 selected Lasso features out of 360 (CLR-topics + diffs):

- `Cable`
- `Problems`
- `Health insurance`
- `Recession`
- `Product prices`
- `Humor/language`
- `Agriculture`
- `Mortgages`
- `Watchdogs`
- `Oil market`
- `Music industry_diff`
- `Chemicals/paper_diff`
- `Unions_diff`
- `Russia_diff`
- `Bankruptcy_diff`
- `Software_diff`
- `US Senate_diff`
- `Futures/indices_diff`
- `Gender issues_diff`
- `National security_diff`

In [8]:
```python
# (c) Real-time forecasting of industrial production growth
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.linear_model import Lasso, LassoCV
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.linalg import svd

# Hyper-parameter grids
param_grid = {
  'compositional_transform__method': ['clr', 'ilr', None],
  'innovation__run': [True, False],
}

class AgglomerativeClusterSummarizer(BaseEstimator, TransformerMixin):
    def __init__(self, n_clusters=10):
        self.n_clusters = n_clusters

    def fit(self, X, y=None):
        # Always fit clustering based on *columns*, not rows
        X = pd.DataFrame(X)
        X_T = X.T  # transpose: now rows = topics, cols = time points
        self.clusterer_ = AgglomerativeClustering(n_clusters=self.n_clusters)
        self.cluster_labels_ = self.clusterer_.fit_predict(X_T)
        self.feature_names_in_ = X.columns.to_list()
```

```
        return self

    def transform(self, X):
        X = pd.DataFrame(X, columns=self.feature_names_in_)
        assert X.shape[1] == len(self.cluster_labels_), "Column mismatch at transform!"
        X_clustered = pd.DataFrame({
            f'cluster_{i}': X.iloc[:, self.cluster_labels_ == i].sum(axis=1)
            for i in np.unique(self.cluster_labels_)
        }, index=X.index)
        return X_clustered

class LagExpansion(BaseEstimator, TransformerMixin):
  def __init__(self, n_lags=1):
    self.n_lags = n_lags

  def fit(self, X, y=None):
    return self

  def transform(self, X):
    if isinstance(X, pd.DataFrame):
      df = X.copy()
      for lag in range(1, self.n_lags + 1):
        df = df.join(X.shift(lag).add_suffix(f'_lag{lag}'))
      return df.fillna(0)
    elif isinstance(X, np.ndarray):
      n_samples, n_features = X.shape
      lagged_data = [X]
      for lag in range(1, self.n_lags + 1):
        lagged = np.zeros_like(X)
        lagged[lag:] = X[:-lag]
        lagged_data.append(lagged)
      return np.hstack(lagged_data)
    else:
      raise ValueError("Input X must be either a pandas DataFrame or a numpy ndarray.")

class LassoSelector(BaseEstimator, TransformerMixin):
    def __init__(self, n_nonzero=5, min_exp=-6.5, max_exp=-3.5, tol=1e-4, max_iter=40):
        self.n_nonzero = n_nonzero
        self.min_exp = min_exp
        self.max_exp = max_exp
        self.tol = tol
        self.max_iter = max_iter

    def fit(self, X, y):
        # binary-search for alpha yielding target nonzeros
        alpha = find_alpha_binary(
            X, y,
            target=self.n_nonzero,
            min_exp=self.min_exp,
            max_exp=self.max_exp,
            tol=self.tol,
            max_iter=self.max_iter
        )
        model = Lasso(alpha=alpha, max_iter=10000).fit(X, y)
        if isinstance(X, pd.DataFrame):
            self.features_ = X.columns[model.coef_ != 0]
        else:
            self.features_ = np.where(model.coef_ != 0)[0]
        return self

    def transform(self, X):
      if isinstance(X, pd.DataFrame):
        return X.loc[:, self.features_]
      elif isinstance(X, np.ndarray):
        return X[:, self.features_]
      else:
        raise ValueError("Input X must be either a pandas DataFrame or a numpy ndarray.")

class CompositionalTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, method=None):
        self.method = method

    def fit(self, X, y=None):
        X = self._validate_input(X)
        self.n_features_in_ = X.shape[1]
        return self

    def transform(self, X):
      X = self._validate_input(X)

      if self.method == 'clr':
        # centered log-ratio
        gm = np.exp(np.mean(np.log(X), axis=1))[:, None]
        return np.log(X / gm)

      elif self.method == 'ilr':
        # first compute the clr
        gm = np.exp(np.mean(np.log(X), axis=1))[:, None]
```

```
        clr = np.log(X / gm)
        # build centering operator
        n = self.n_features_in_
        H = np.eye(n) - np.ones((n, n)) / n
        # orthonormal basis for the clr-space via SVD of H
        _, _, vh = svd(H)
        # take first n-1 columns
        V = vh.T[:, :-1]
        # project clr onto that orthonormal basis
        return clr @ V

      elif self.method is None:
        return X

      else:
        raise ValueError("Invalid method. Choose from {'clr', 'ilr', None}.")

    def _validate_input(self, X):
        if isinstance(X, pd.DataFrame):
            X = X.values
        elif not isinstance(X, np.ndarray):
            raise ValueError("Input must be a pandas DataFrame or numpy array.")
        if np.any(X <= 0):
            raise ValueError("All input values must be positive for log transforms.")
        return X

X_raw = topics
y_raw = macro['indprol1']
common_idx = X_raw.index.intersection(y_raw.index)
y_raw = y_raw.loc[common_idx]
X_raw = X_raw.loc[common_idx]

# CV Pipeline
tscv = TimeSeriesSplit(n_splits=5)
class InnovationTransformer(BaseEstimator, TransformerMixin):
  def __init__(self, run=True):
    self.run = run

  def fit(self, X, y=None):
    return self

  def transform(self, X):
    X = X*100 #sloppy rescaling
    if not self.run:
      return X

    # DataFrame path
    if isinstance(X, pd.DataFrame):
      df = X.copy()
      df_diff = df.diff().fillna(0)
      return pd.concat([df, df_diff], axis=1)

    # ndarray path
    elif isinstance(X, np.ndarray):
      diffs = np.vstack([np.zeros((1, X.shape[1])), np.diff(X, axis=0)])
      return np.hstack([X, diffs])

    else:
      raise ValueError("Input must be DataFrame or ndarray")

pipeline = Pipeline([
  ('compositional_transform', CompositionalTransformer()),
  ('innovation', InnovationTransformer()),
  ('scaler', StandardScaler()),
  ('lasso', LassoCV(max_iter=10000,
            alphas=np.logspace(-5, -2.5, 25),
            n_jobs=-1,
            random_state=42)),
])
# Grid search
param_grid = {p: v for p, v in param_grid.items() if p in pipeline.get_params()}

grid = GridSearchCV(pipeline, param_grid, cv=tscv, scoring='r2', n_jobs=-1)
grid.fit(X_raw, y_raw)
# Evaluate final model OOS
split_idx = int(len(X_raw) * 0.8)
X_train, X_test = X_raw.iloc[:split_idx], X_raw.iloc[split_idx:]
y_train, y_test = y_raw.iloc[:split_idx], y_raw.iloc[split_idx:]
best_params = grid.best_params_ if hasattr(grid, 'best_params_') else {}
pipeline.set_params(
 **best_params
)
pipeline.fit(X_train, y_train)
r2_oos = pipeline.score(X_test, y_test)

lasso_cv = pipeline.named_steps['lasso']
print(f"Best parameters: {best_params}")
print(f"{(lasso_cv.coef_ != 0).sum()} non-zero coefficients")
```

```
cols = np.array(list(X_raw.columns) + list(X_raw.columns + '_diff'))
selected_features = cols[lasso_cv.coef_ != 0]
print(f"Selected features: {selected_features}")
print(f'Out-of-sample R2: {r2_oos:.4f}')
```

```
/Users/potter/miniconda3/envs/ds/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descent.py:681: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.0039966971608569e-06, tolerance: 5.228596487302508e-07
  model = cd_fast.enet_coordinate_descent_gram(
/Users/potter/miniconda3/envs/ds/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descent.py:681: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1410299470280064e-06, tolerance: 5.228596487302508e-07
  model = cd_fast.enet_coordinate_descent_gram(
Best parameters: {'compositional_transform__method': 'clr', 'innovation__run': True}
20 non-zero coefficients
Selected features: ['Cable' 'Problems' 'Health insurance' 'Recession' 'Product prices'
 'Humor/language' 'Agriculture' 'Mortgages' 'Watchdogs' 'Oil market'
 'Music industry_diff' 'Chemicals/paper_diff' 'Unions_diff' 'Russia_diff'
 'Bankruptcy_diff' 'Software_diff' 'US Senate_diff' 'Futures/indices_diff'
 'Gender issues_diff' 'National security_diff']
Out-of-sample R2: 0.1270
```

(d) Document-term matrix for WSJ headlines

> Next, download the articles.pq file from canvas. This file contains headlines from the Wall Street Journal. Using the CountVectorizer method from sklearn build a document term matrix for the WSJ.

In [20]:
```
# (d) Build document-term matrix
from sklearn.feature_extraction.text import CountVectorizer
import pyarrow.parquet as pq

# Load WSJ headlines
articles = pq.read_table('articles.pq').to_pandas()

# turn display_date into a month index
articles['date'] = articles['display_date'].dt.to_period('M').dt.to_timestamp()
vectorizer = CountVectorizer(max_features=10000)  # adjust as needed

# collapse all headlines in each month into one big string
monthly_docs = articles.groupby('date')['headline'].apply(' '.join)

# now vectorize per-month
X_counts = vectorizer.fit_transform(monthly_docs)
feature_names = vectorizer.get_feature_names_out()

# optional: put into a DataFrame aligned with macro
X_counts_df = pd.DataFrame(
  X_counts.toarray(),
  index=monthly_docs.index,
  columns=feature_names
)

print("monthly DTM shape:", X_counts_df.shape)
print("Top 25 words in DTM:\n" ,X_counts_df.sum().sort_values(ascending=False).head(25))
```

```
monthly DTM shape: (408, 10000)
Top 25 words in DTM:
 by           8452
of           5977
the          5647
to           5398
street       3868
in           3810
journal      3671
wall         3633
staff        3091
and          2783
reporter     2552
for          2106
on           1994
news         1625
brief        1521
business     1430
is           1409
as           1098
new           765
with          677
world         639
at            598
inc           598
from          534
reporters     531
dtype: int64
```

(e) Contemporaneous exercises with raw counts

> Next, repeat the contemporaneous exercises from part (a) and (b) using the counts. How many non-zero counts do you need to recover the same R2 ? What does that say about the informativeness of the counts vs. topics?

In [5]:
```
# (e) Lasso + OLS with counts
# TODO: find count predictors matching R2 from part (a)
```

(f) Forecasting with counts

> Using the counts attempt to form the best forecasting model for industrial production growth. How well can you do relative to the topics?

In [6]:
```
# (f) Forecast with raw counts
# TODO: forecasting pipeline using X_counts
```

(g) TF-IDF vs raw counts

> Convert the raw counts into tf-idf and repeat the exercises from part (e) and (d). Summarize the differences between the tf-idf and raw count approaches. Which terms are most important in either approach?

In [7]:
```
# (g) TF-IDF analysis
from sklearn.feature_extraction.text import TfidfTransformer
```