The Convergence Hypothesis: A Bayesian Examination with Gibbs Sampling

Ilies El Jaouhari[1], Thomas Salomon[2], and Ismail Jamal-Eddine[3]

[1] ilies.eljaouhari@ensae.fr
[1] thomas.salomon@ensae.fr
[3] ismail.jamal-eddine@ensae.fr

Keywords:

1 Introduction

In this study, we investigate the complex dynamics of economic growth across a diverse array of nations, utilizing the extensive Barro and Lee dataset spanning the years 1960 to 1985. This comprehensive dataset, encompassing 138 countries, serves as a foundational resource for our investigation into the variations in national growth rates in GDP per capita, which we have chosen as our key dependent variable. Armed with an array of 62 covariates drawn from 90 complete data entries, our analysis is geared towards a critical examination of the convergence hypothesis, a cornerstone concept in the study of economic development.

The convergence hypothesis states that over time : lower-income economies are set to experience more rapid growth compared to their wealthier counterparts, potentially leading to a gradual equalization in income levels and developmental progress across nations. This hypothesis centers on the idea that poorer nations possess the potential to ”catch up” to more affluent nations in terms of economic output and living standards. A key implication of this hypothesis is the anticipated negative relationship between a country’s initial GDP level and its growth rate. However, initial empirical tests produce statistically insignificant results, suggesting a more nuanced relationship than the hypothesis straightforwardly predicts.

Our approach to testing the convergence hypothesis involves a selection of relevant variables that could significantly impact a nation’s economic growth trajectory through double-selection. Our analysis begins with a straightforward bivariate regression, examining the relationship between growth rates and initial GDP levels of countries. To bridge the gap between theory and empirical observations, we then embark on the double-selection, taking into account various country-specific characteristics. Through this comprehensive approach, we aim to shed light on the convergence hypothesis, offering fresh insights into one of the most debated topics in economic growth theory.

2 Data

Our analysis is based on a detailed dataset from the Barro and Lee study, covering the years 1960 to 1985. This dataset provides a comprehensive view of economic performance across 138 countries. Specifically, it includes 90 complete observations, each detailing a range of economic and development indicators. The dataset comprises 62 columns, encompassing various factors that could influence a country’s growth rate, which is our primary variable of interest. The columns include a diverse set of variables, such as `gdpsh465`, `bmp1l`, `freeop`, and `freetar`, which represent different economic and trade-related metrics. Additionally, there

are indicators related to health (h65, hm65, hf65), population demographics (p65, pm65, pf65), and education (teapri65, teasec65). Other notable variables include those related to fertility (fert65), mortality (mort65), life expectancy (lifee065), investment shares (invsh41), government spending (govwb1, govsh41), and trade (ex1, im1).

3 Question

We investigate the convergence hypothesis, which posits a negative relationship between a country's initial GDP level and its subsequent growth rate. In our study, we also focus on identifying robust conditioning variables, essential for a comprehensive verification of this hypothesis. These conditioning factors are crucial in fine-tuning our understanding of the dynamics at play, thereby enabling a more accurate and nuanced analysis of the convergence phenomenon.

Our analytical framework incorporates the spike-and-slab prior, which is particularly suited for high-dimensional settings as discussed in the course material on high-dimensional models. The spike-and-slab prior is a discrete mixture model that assigns a point mass at zero (referred to as the "spike") and an absolutely continuous density (referred to as the "slab") to each parameter in the regression model.

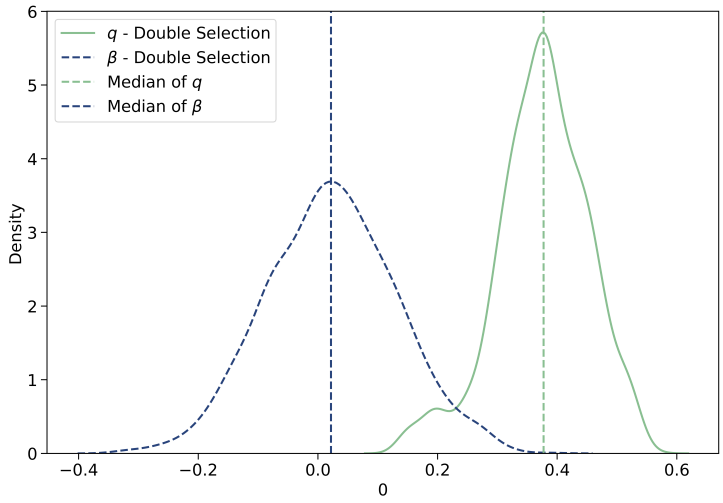
4 Results

In order to first tackle the question : "Do countries with initial low GDP witness a positive impact on their GDP growth ?", we will answer this question with both frequentist approach and a bayesian one. In order to have a first glance at the convergence hypothesis, a simple bivariate OLS of countries' initial GDP on the growth rate was implemented. We can observe the following results:

Variable	**Coefficient**	**Std. Error**	**P-value**	**Metric**	**Value**
Constant	0.0453	0.005	< 0.001	R-squared	0.000
GDP per capita	0.0012	0.006	**0.840**	Adj. R-squared	-0.011

As illustrated in the table above, the estimator for GDP is close to zero, suggesting that GDP does not quantitatively and significantly impact the growth rate. Furthermore, the statistical analysis leads to the rejection of the null hypothesis, indicating insufficient evidence to establish a meaningful relationship between GDP and its growth. This observation is consistent with our analysis of the posteriors for q and the estimator:

Figure 1: Posterior Distributions of q and β



Utilizing the code from the previous assignment, we observed that the distribution of our posterior q is significantly low-centered (since the regression is bivariate), as depicted above. We can also highlight a median value for β centered narrowly around 0. this underlines a quantitatively not significant relationship. This outcome challenges our initial assumptions and underscores the importance of considering conditional effects based on country-specific characteristics. When sampling our posterior q, a problem was encoutered around the 3000-level iterations of Gibbs, our algorithm was unable to sample the normal multivariate law because $\widetilde{\beta}$ was near zero. Consequently, both bayesian and frequentist approach lead to the same conclusion, the GDP initial level does not have any impact whatsoever on the growth rate aligning with the findings in [1]. This result appear to be in contradiction of the real convergence theory in the macroeconomic model Ramsey–Cass–Koopmans.

4.1 **Double-selection**

In our analysis, we utilized a double selection procedure employing a Bayesian Gibbs sampling approach (with 5000 iterations and 1000 for burn-in) to investigate the relationship between the initial level of Gross Domestic Product (GDP) (our independent variable, IV) and the growth rate (our dependent variable, DV). This method is particularly beneficial in scenarios with a large number of potential covariates, as it helps in selecting the most relevant ones for accurate estimation of the IV's effect on the DV. It has also be shown in [2] to be robust to Heteroskedasticity. The double selection process was carried out in two main stages:

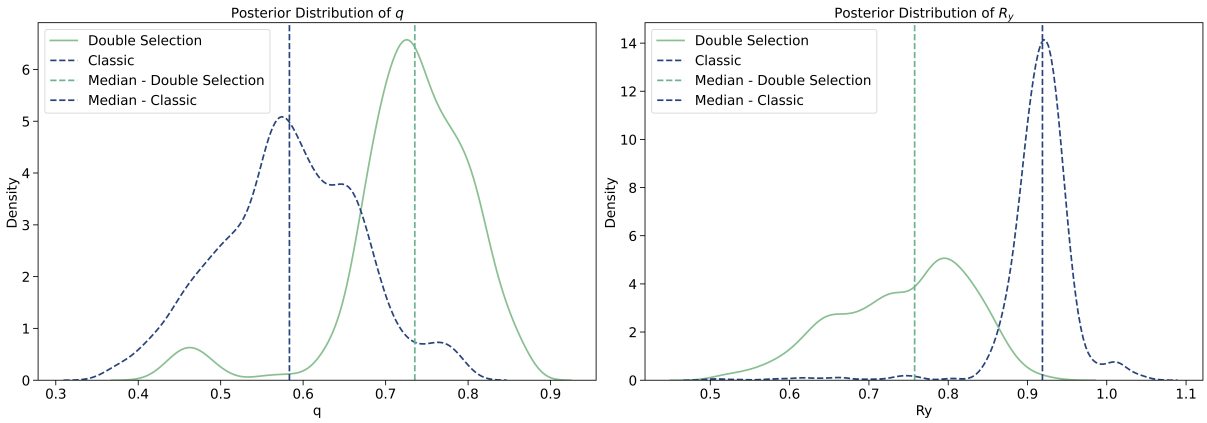
In the first stage, we regressed all covariates against the growth rate (DV) using the Gibbs sampling method. This step was crucial to identify covariates that have a direct relationship with the growth rate. The selection of covariates was based on the non-zero coefficients in the Gibbs sampling output, indicating a significant direct effect on the growth rate. The second stage involved regressing all covariates against the initial level of GDP (IV), again employing the Gibbs sampling method. This stage aimed to identify covariates that have a direct relationship with the initial GDP level. The selection criterion was similar to the first stage, focusing on non-zero coefficients. After completing both selection stages, we combined the covariates selected from each stage. This combined set of covariates was then utilized in a final regression model to estimate the effect of the initial GDP level on the growth rate.

The double selection approach is advantageous as it controls for both direct effects on the growth rate and potential confounding effects related to the initial GDP level. The Bayesian Gibbs sampling in the regression adds a probabilistic dimension, allowing for a more nuanced understanding of the variable relationships. In conclusion, this method enabled us to select relevant covariates for our final model in a statistically robust manner, enhancing the accuracy of our estimates for the impact of the initial GDP level on growth rates.

4.2 **Posterior distributions and sparsity**

The figure below present the kernel density estimate for the Gibbs sampling using the double-selection regression and the classical approach. Both were run during 5000 iterations with a 1000 for burn-in. We can immediately observe that the posterior from the classic approach has lower median for the probability of inclusion indicating higher sparsity. On the right, it is also clear we have a higher variability in the posterior for R_y after the covariate selection indicating that the classical approach exhibits a potentially over-fitted relationship.

Figure 2: Posteriors of (q, R_y) for the classic and the double-selection approaches



In order to verify the overfitting for the dataset, we divided it to conduct a 3-fold cross-validation. Here we present the MSE results for both the double selection (DB) and the classic (CL) approaches. The mean MSE for the DB approach was found to be 1.0055, indicating a more consistent performance across the folds compared to the CL approach, which had a higher mean MSE of 1.4503. This variation suggests that the CL approach may be more prone to overfitting, as evidenced by the significant fluctuation in its MSE values across different folds. Notably, the DB approach showed a lower MSE in two out of the three folds, reinforcing the efficiency of the double selection method in providing more stable and reliable predictions, especially in scenarios involving numerous covariates and complex variable relationships.

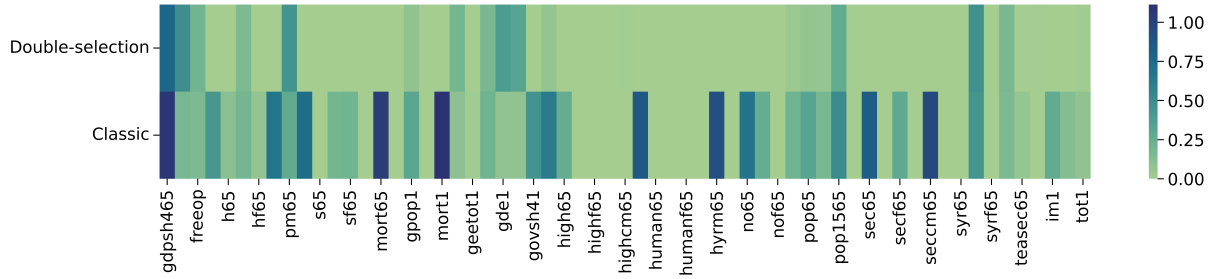
	Fold 1	**Fold 2**	**Fold 3**	**Mean**
MSE DB	1.4061	0.5808	1.0097	**1.0055**
MSE CL	1.3481	2.2169	0.7858	**1.4503**

Table 1: Comparison of MSE for Double Selection (DB) and Classic (CL) Approaches

4.3 Sparsity

The conclusions drawn for the observation of the posteriors for our probability of inclusion can also be retrieved from the following heatmap:

Figure 3: Values of β for the different approaches



It presents the absolute average value for our estimators computed over the last 100 iterations of the Gibbs sampling. We can see that the double selection approach outputs a highly sparser estimator that the classical one, as excepted by looking at the posterior for the inclusion. The next section explores this heatmap numerically in order to draw insights on the Solow-Swan-Ramsey growth model's convergence hypothesis.

4.4 Estimators

Our Bayesian Gibbs sampling analysis yielded estimators for key variables impacting the growth rate (DV) and initial GDP (IV). Non-zero coefficients guided our covariate selection, indicating statistically significant relationships.

	IV	bmp1l	syrm65	pm65	gde1	govwb1	pop1565	freeop
Value	**-0.779**	-0.488	0.477	0.454	0.390	-0.333	-0.272	0.205

Table 2: Median values for the last 100 Gibbs sampling iterations

The results, as outlined in Table 2, succinctly support the convergence hypothesis and the results both in [3] and [1]:

- A negative coefficient for `IV` suggests that countries with lower initial GDPs tend to grow faster, in line with convergence.
- Education variables `hm65`, `pm65`, and `syrm65` with positive coefficients, emphasize the growth-boosting effects of human capital.
- Government spending indicators `govwb1` and `gde1` show a potential drag on growth with their negative coefficients.
- Demographic factors `pop1565`, `pop65`, and `gpop1` are mostly positive, hinting at a demographic contribution to economic catch-up.
- Variables like `bmp1l`, `tot1`, `invsh41`, and `freeop` provide a nuanced view of trade and investment's role in convergence.

- The negative coefficient for political stability (`pinstab1`) indicates its positive influence on growth, aiding convergence.

For the purpose of being succinct, we refer the reader to Barro's research in [4] and [3] for more complete definition of the variables. Clearly, this methodological approach, contrasting with the simple bivariate model, offers a comprehensive affirmation of the convergence theory by integrating a broad spectrum of significant variables to condition on.

5 Conclusion

In conclusion, this report has helped us validate the convergence hypothesis and answer the initial question by identifying relevant conditioning covariates for the Solow-Swan-Ramsey growth model [5]. The use of Bayesian methods for nuanced understanding of the dataset's relationships provided significant advantages, notably in uncovering overfitted scenarios even before employing a test or credible set.

Achieving sparsity in our model was particularly crucial given the large number of covariates, as it ensures that only the most significant factors are considered, thereby enhancing the model's interpretability and efficiency. The cross-validation and MSE comparison yielded supporting evidence of consistent and reliable results. Furthermore, it managed to significantly improve the accuracy of the estimates (with a 45% increase in MSE). The results were interpreted in light of the economic model, enhancing the sparse structure of the estimators and underlining the relevance of certain covariates in economic growth. Notably, we identified factors related to trade openness and education that uniquely contribute to the economic growth narrative, compared to classical approaches.

6 Acknowledgements

We extend our sincere thanks to our Bayesian statistics teacher: Ms Anna Simoni, for the expert guidance and clarity in teaching the complex paradigm of Bayesian thinking and probability. Your dedication has significantly enriched our understanding and application of Bayesian methods in our professional pursuits. We deeply appreciate your support and inspiration throughout this course.

References

[1] Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference for high-dimensional sparse econometric models, 2011.

[2] Oleg Urminsky, Victor Chernozhukov, and Christian Hansen. Using double-lasso regression for principled variable selection, 2016.

[3] Robert J. Barro and Xavier Sala-i Martin. Technological diffusion, convergence, and growth. *National Bureau of Economic Research*, 1995.

[4] Robert J. Barro and Jong-Wha Lee. Sources of economic growth. *Carnegie-Rochester Conference Series on Public Policy*, 40:1–46, 1994.

[5] David Cass. Optimum growth in an aggregative model of capital accumulation. *The Review of Economic Studies*, 32(3):233–240, 1965. Accessed: 01/07/2011.

January 20, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from numba import jit

import matplotlib.pyplot as plt
import seaborn as sns
import scienceplots
```

[2]:
```
%matplotlib inline
plt.style.use(['nature', 'ieee', 'notebook'])
```

0.0.1 Data Import

[3]:
```
macro_proc = pd.read_excel('macro_excel.xlsx')
macro_proc.columns = macro_proc.columns.str.replace(' ', '')
macro_proc = (macro_proc - macro_proc.mean())/macro_proc.std()
```

0.0.2 Gibbs sampler

Same as in Assigment 2. Uses numba for precompilation and paralleling computations

Row processing Function *process_row* is the same as in Assignment 2

[ ]:
```
row = pd.Series([1, 1, 1, 1, None, None, X.values, y.values, None, None],␣
↪index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y', 'beta', 'epsilon'])
results = process_row(row)
```

0.0.3 Bivariate regression and statistical significance

Rajouter les résultats de l'OLS (p-value + valeur de l'estimateur) et c'est bon.

[7]:
```
y = macro_proc.loc[:, 'Outcome']
X = macro_proc.drop('Outcome', axis=1)
```

[21]:
```
X_biv = X.loc[:, "gdpsh465"].to_frame()
# X_biv['const'] = 1
```

[24]:
```
row = pd.Series([1, 1, 1, 1, None, None, X_biv.values, y.values, None, None],␣
 ↪index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y', 'beta', 'epsilon'])
results = process_row(row)
```

```
  0%|          | 0/2000 [00:00<?, ?it/s]
```

[25]:
```
betas = pd.DataFrame(results[2])
qs = pd.DataFrame(results[0])
```

[26]:
```
qs
```

[26]:

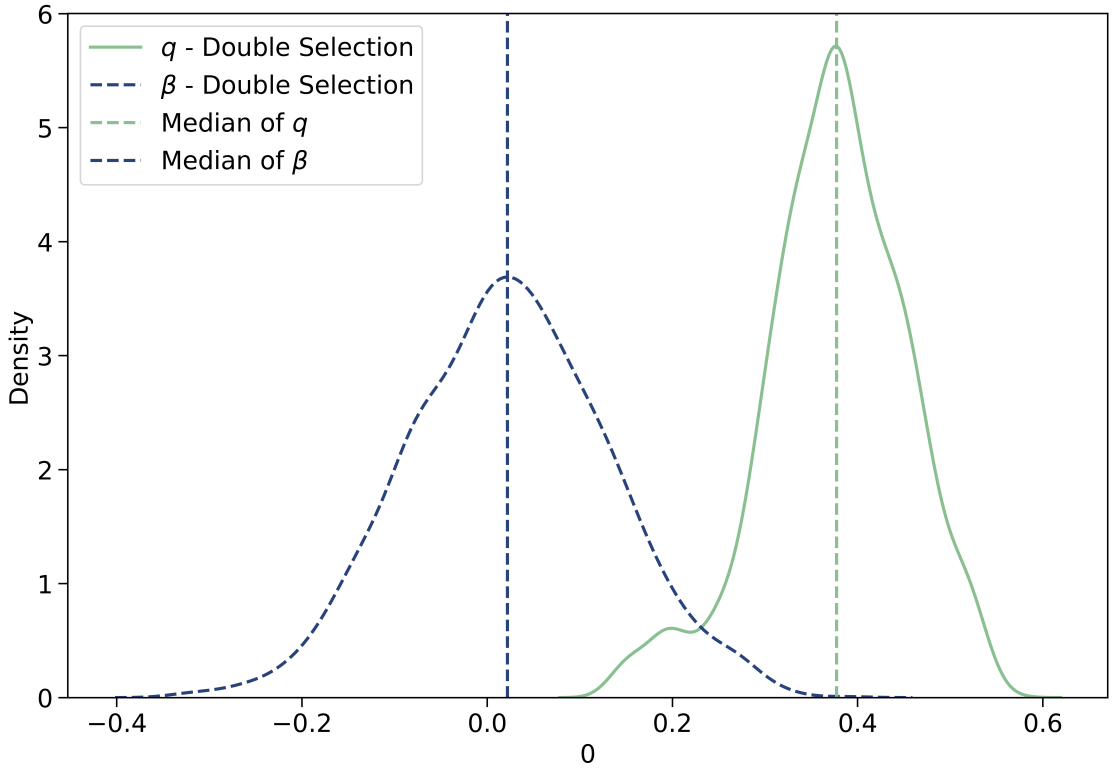
	0
0	0.500000
1	0.498629
2	0.506385
3	0.503511
4	0.485304
…	…
1995	0.211307
1996	0.195608
1997	0.183129
1998	0.185886
1999	0.174195

[2000 rows x 1 columns]

Posterior Distributions of q and β

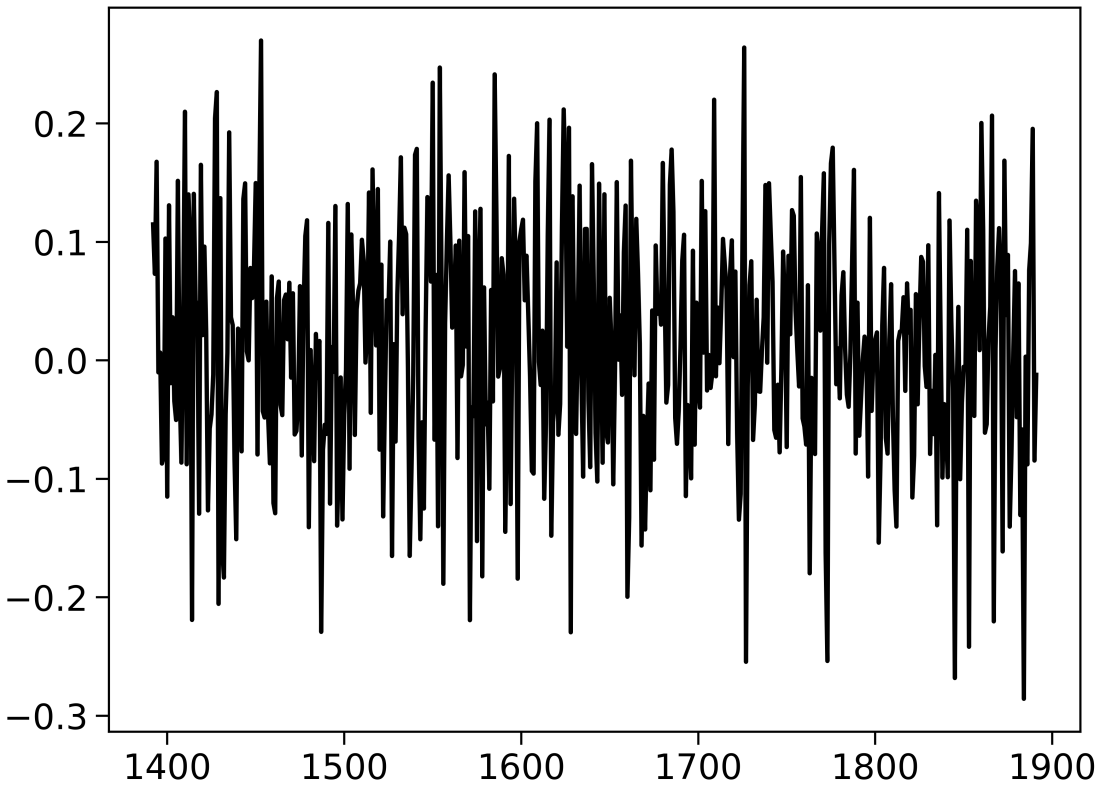
[33]:
```
# Define color map
crest = sns.color_palette("crest", as_cmap=True)

# Plotting
fig, ax = plt.subplots(figsize=(10, 7))

# Plot for 'q' column
sns.kdeplot(data=qs, x=0, ax=ax, label=r'$q$ - Double Selection', color=crest(0.
 ↪1), common_norm=False)
# Plot for 'beta' column
sns.kdeplot(data=betas, x=0, ax=ax, label=r'$\beta$ - Double Selection',␣
 ↪color=crest(0.9), common_norm=False)

# Adding medians
q_median = qs[0].median()
beta_median = betas[0].median()
ax.axvline(q_median, color=crest(0.1), linestyle='--', label=f'Median of $q$')
ax.axvline(beta_median, color=crest(0.9), linestyle='--', label=f'Median of␣
 ↪$\\beta$')
```

```
# Title and Legend
ax.legend()

# Adjust layout and display
plt.tight_layout()
plt.show()
```



[14]:
```
betas.iloc[-500:, 0].plot()
```

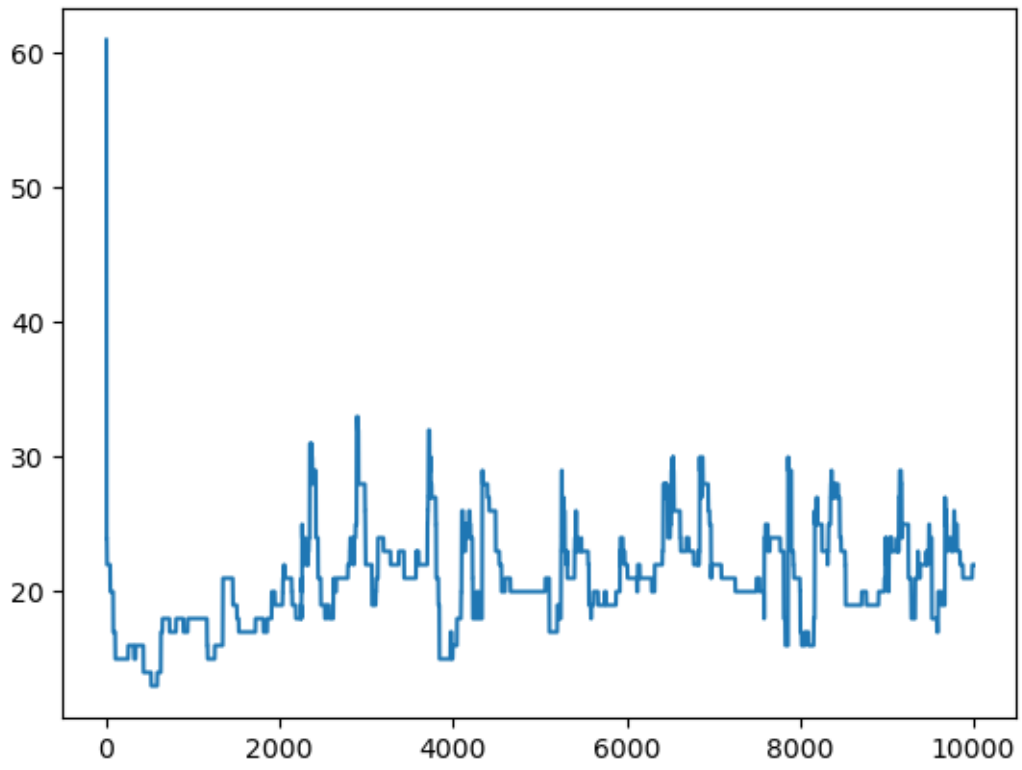
[14]: <Axes: >



0.0.4 Evolution of the sparsity of β at each iteration of the Gibbs sampling

[69]:
```
(pd.DataFrame(results[2]) == 0.).sum(axis=1).plot()
```

[69]: <Axes: >



0.1 Double selection with Gibbs Sampling

[34]:
```
def run_double_selection(X, y, IV):
    # First regression: Covariates on DV
    row_for_dv = pd.Series([1, 1, 1, 1, None, None, X.values, y.values, None,␣
 ↪None],
                           index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
 ↪'beta', 'epsilon'])
    results_dv = process_row(row_for_dv)
    selected_covariates_dv = identify_selected_covariates(results_dv, X.columns)

    # Second regression: Covariates on IV
    row_for_iv = pd.Series([1, 1, 1, 1, None, None, X.values, IV.values, None,␣
 ↪None],
                           index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
 ↪'beta', 'epsilon'])
    results_iv = process_row(row_for_iv)
    selected_covariates_iv = identify_selected_covariates(results_iv, X.columns)

    # Combine the selected covariates
```

```
    combined_covariates = selected_covariates_dv.
↪intersection(selected_covariates_iv)

    # Final regression: Combined Covariates and IV on DV
    X_final = X[combined_covariates]
    X_final['IV'] = IV  # Add the IV to the predictors
    final_row = pd.Series([1, 1, 1, 1, None, None, X_final.values, y.values,␣
↪None, None],
                          index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
↪'beta', 'epsilon'])

    final_results = process_row(final_row)
    final_covariates = X_final.columns

    # Prepare the output dictionary
    output = {
        "selected_covariates_dv": selected_covariates_dv,
        "selected_covariates_iv": selected_covariates_iv,
        "combined_covariates": combined_covariates,
        "intermediate_results_dv": results_dv,
        "intermediate_results_iv": results_iv,
        "final_results": final_results,
        "final_covariates": final_covariates
    }

    return output

def identify_selected_covariates(results, column_names):
    # Implement based on process_row output
    selected_covariates = {column_names[i] for i in range(len(column_names)) if␣
↪results[2][-1][i] != 0}
    return selected_covariates
```

[35]:
```
y = macro_proc.loc[:, 'Outcome']
X = macro_proc.drop('Outcome', axis=1)
```

[36]:
```
iv = X.loc[:, 'gdpsh465']
y = y
X = X.drop('gdpsh465', axis=1)
```

[37]:
```
final_results = run_double_selection(X, y, iv)
```

```
  0%|          | 0/2000 [00:00<?, ?it/s]
  0%|          | 0/2000 [00:00<?, ?it/s]
C:\Users\Hermes\AppData\Local\Temp\ipykernel_11340\3487297188.py:18:
FutureWarning: Passing a set as an indexer is deprecated and will raise in a
```

```
future version. Use a list instead.
  X_final = X[combined_covariates]
C:\Users\Hermes\AppData\Local\Temp\ipykernel_11340\3487297188.py:19:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_final['IV'] = IV  # Add the IV to the predictors

  0%|          | 0/2000 [00:00<?, ?it/s]
```

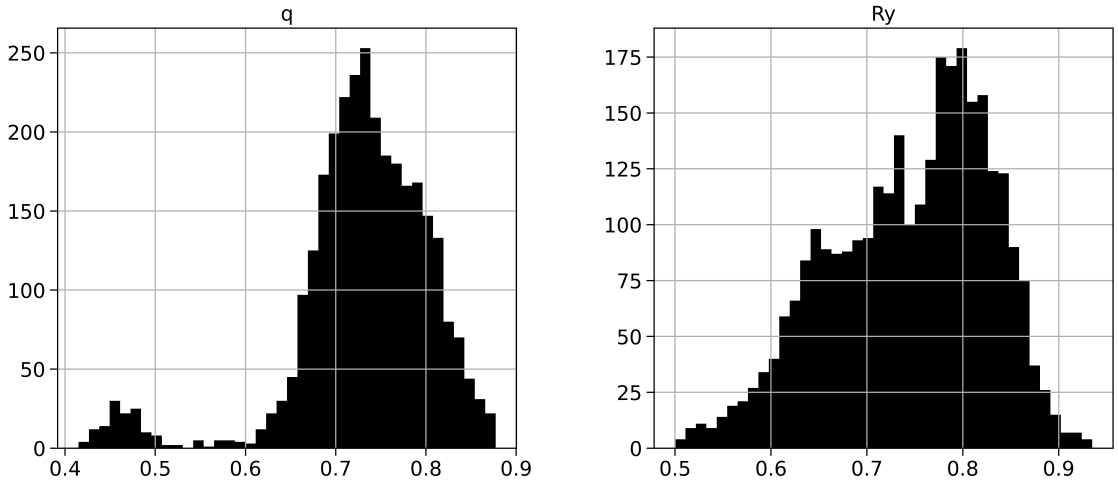
0.1.1 Posteriors

With double-selection Regression

[34]:
```
double_selection_posteriors = pd.DataFrame({"q":␣
↪final_results['final_results'][0], "Ry":final_results['final_results'][1]})
```

[40]:
```
double_selection_posteriors.hist(figsize=(15, 6), bins=40)
```

[40]:
```
array([[<Axes: title={'center': 'q'}>, <Axes: title={'center': 'Ry'}>]],
      dtype=object)
```



Classic way (w Gibbs)

[38]:
```
y = macro_proc.loc[:, 'Outcome']
X = macro_proc.drop('Outcome', axis=1)
```

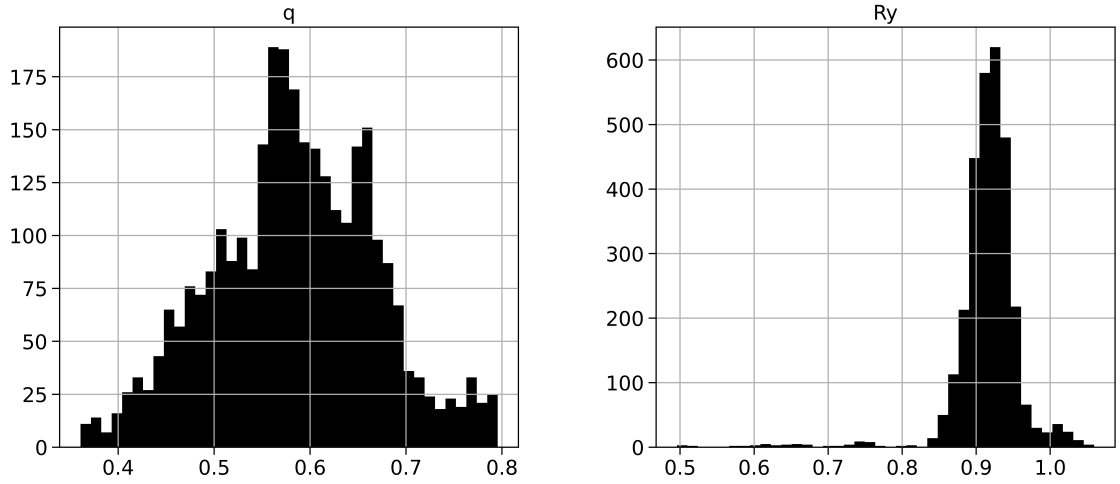
[ ]:
```
classic_row = final_row = pd.Series([1, 1, 1, 1, None, None, X.values, y.
↪values, None, None],
```

```
                              index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
 ↪'beta', 'epsilon'])

classic_results = process_row(classic_row)
```

[45]:
```
classic_posteriors = pd.DataFrame({"q": classic_results[0], "Ry":␣
 ↪classic_results[1]})
```

[46]:
```
classic_posteriors.hist(figsize=(15, 6), bins=40)
```

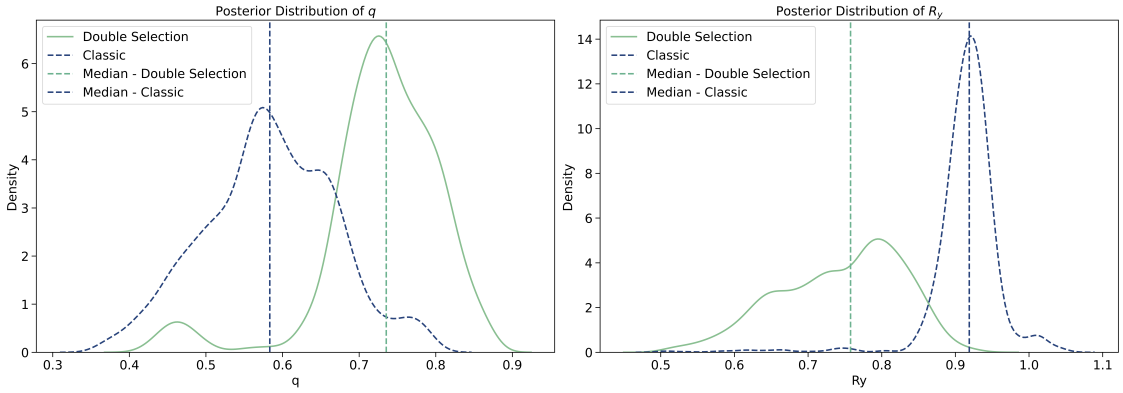
[46]:
```
array([[<Axes: title={'center': 'q'}>, <Axes: title={'center': 'Ry'}>]],
      dtype=object)
```



KDE Comparison

[159]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(20, 7))

# Define color map
crest = sns.color_palette("crest", as_cmap=True)

# Plot for 'q' column
sns.kdeplot(data=df1, x='q', ax=axs[0], label='Double Selection', color=crest(0.
 ↪1), common_norm=False)
sns.kdeplot(data=df2, x='q', ax=axs[0], label='Classic', color=crest(0.9),␣
 ↪common_norm=False)
axs[0].set_title(r'Posterior Distribution of $q$')
axs[0].axvline(np.median(df1['q']), color=crest(0.2), linestyle='--',␣
 ↪label='Median - Double Selection')
axs[0].axvline(np.median(df2['q']), color=crest(0.9), linestyle='--',␣
 ↪label='Median - Classic')
axs[0].legend()
```

```
# Plot for 'Ry' column
sns.kdeplot(data=df1, x='Ry', ax=axs[1], label='Double Selection',␣
 ↪color=crest(0.1), common_norm=False)
sns.kdeplot(data=df2, x='Ry', ax=axs[1], label='Classic', color=crest(0.9),␣
 ↪common_norm=False)
axs[1].set_title(r'Posterior Distribution of $R_y$')
axs[1].axvline(np.median(df1['Ry']), color=crest(0.2), linestyle='--',␣
 ↪label='Median - Double Selection')
axs[1].axvline(np.median(df2['Ry']), color=crest(0.9), linestyle='--',␣
 ↪label='Median - Classic')
axs[1].legend()

# Adjust layout and display
plt.tight_layout()
plt.show()
```



Cross-validation (3-fold)

[ ]:
```
iv = X.loc[:, 'gdpsh465']
y = y
X = X.drop('gdpsh465', axis=1)

final_results = run_double_selection(X, y, iv)['final_results']
db_estimators_last = final_results[2][-1]

y = macro_proc.loc[:, 'Outcome']
X = macro_proc.drop('Outcome', axis=1)

classic_row = final_row = pd.Series([1, 1, 1, 1, None, None, X.values, y.
 ↪values, None, None],
                                    index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
 ↪'beta', 'epsilon'])
```

```
classic_results = process_row(classic_row)
cl_estimators = final_results[2][-1]
```

[ ]:
```
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error

y = y
X_classic = X.copy()
iv = X.loc[:, 'gdpsh465']
X_double_selection = X.drop('gdpsh465', axis=1)

kf = KFold(n_splits=3, shuffle=True, random_state=42)

# Initialize lists to store MSE for each approach
mse_db = []
mse_cl = []

for train_index, test_index in kf.split(X):
    X_train_cl, X_test_cl = X_classic.iloc[train_index], X_classic.
 ↪iloc[test_index]
    X_train_db, X_test_db = X_double_selection.iloc[train_index],␣
 ↪X_double_selection.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    iv_train, iv_test = iv.iloc[train_index], iv.iloc[test_index]

    # Double Selection
    db_results = run_double_selection(X_train_db, y_train, iv_train)
    covar_double_selection = pd.Index(db_results['combined_covariates']).
 ↪append(pd.Index(['gdpsh465']))
    double_selection_estimators = pd.Series(0, index=X.columns)
    double_selection_estimators[covar_double_selection] =␣
 ↪db_results['final_results'][2][-1]
    y_pred_db = X_test_cl.dot(double_selection_estimators)
    mse_db.append(mean_squared_error(y_test, y_pred_db))

    # Classic Regression
    classic_row = pd.Series([1, 1, 1, 1, None, None, X_train_cl.values, y_train.
 ↪values, None, None],
                            index=['a', 'b', 'A', 'B', 's', 'Ry', 'X', 'y',␣
 ↪'beta', 'epsilon'])
    cl_results = process_row(classic_row)
    classic_estimators = pd.Series(cl_results[2][-1], index=X.columns)
    y_pred_cl = X_test_cl.dot(classic_estimators)
    mse_cl.append(mean_squared_error(y_test, y_pred_cl))

# MSE Results
```

```
for i in range(3):
    print(f"Fold {i+1} - MSE Double Selection: {mse_db[i]}, MSE Classic:␣
 ↪{mse_cl[i]}")
```

```
  0%|          | 0/2000 [00:00<?, ?it/s]

  0%|          | 0/2000 [00:00<?, ?it/s]

C:\Users\Hermes\AppData\Local\Temp\ipykernel_11340\3487297188.py:18:
FutureWarning: Passing a set as an indexer is deprecated and will raise in a
future version. Use a list instead.
  X_final = X[combined_covariates]
C:\Users\Hermes\AppData\Local\Temp\ipykernel_11340\3487297188.py:19:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_final['IV'] = IV  # Add the IV to the predictors

  0%|          | 0/2000 [00:00<?, ?it/s]

  0%|          | 0/2000 [00:00<?, ?it/s]

  0%|          | 0/2000 [00:00<?, ?it/s]

  0%|          | 0/2000 [00:00<?, ?it/s]
```

[45]:
```
pd.Index(db_results['combined_covariates'])
```

[45]:
```
Index(['humanm65', 'pinstab1', 'hm65', 'pf65', 'geetot1', 'tot1', 'geerec1',
       'highcf65', 'mort1', 'teasec65', 'pop65', 'hf65', 'invsh41', 'pop6565',
       'freetar', 'secc65', 'im1', 'teapri65', 'freeop', 'lifee065', 'pm65',
       'gde1', 'nof65', 'govwb1', 'worker65', 'xr65', 'sf65', 'bmp1l'],
      dtype='object')
```

0.1.2 Estimators

[162]:
```
final_estimators = pd.DataFrame({'value':
 ↪final_results['final_results'][2][-1]},␣
 ↪index=list(final_results['final_covariates']))
final_estimators.loc[:, 'value'] = final_estimators.loc[:, 'value'].apply(abs)
```

[169]:
```
print(final_estimators.sort_values('value', ascending=False).iloc[:8].T.
 ↪to_latex())
```

```
\begin{tabular}{lrrrrrrrr}
\toprule
{} &        IV &     bmp1l &    syrm65 &      pm65 &      gde1 &    govwb1 &
pop1565 &    freeop \\
```

```
\midrule
value &  0.779961 &  0.488257 &  0.477292 &  0.45439 &  0.389918 &  0.333146 &
0.272273 &  0.205588 \\
\bottomrule
\end{tabular}
```

```
C:\Users\Hermes\AppData\Local\Temp\ipykernel_11712\1444881185.py:1:
FutureWarning: In future versions `DataFrame.to_latex` is expected to utilise
the base implementation of `Styler.to_latex` for formatting and rendering. The
arguments signature may therefore change. It is recommended instead to use
`DataFrame.style.to_latex` which also contains additional functionality.
  print(final_estimators.sort_values('value',
ascending=False).iloc[:8].T.to_latex())
```

0.1.3 Sparsity heatmap

[112]:
```
final_results['final_results'][2][-1]
```

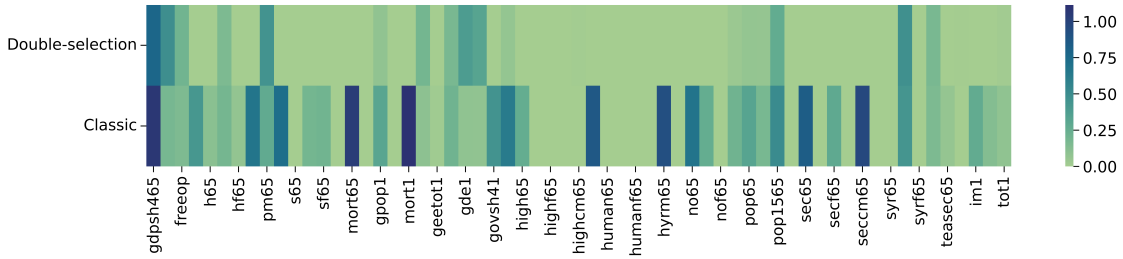
[112]:
```
array([ 0.06600297,  0.01412377, -0.00877303, -0.15838107,  0.17081984,
       -0.17135321,  0.        ,  0.        ,  0.        ,  0.06581704,
        0.47729192,  0.        , -0.04032498,  0.45439044,  0.        ,
        0.        , -0.48825674,  0.        ,  0.        , -0.06520572,
        0.        ,  0.19545861, -0.27227283,  0.        ,  0.3899185 ,
        0.00851085,  0.20558761,  0.08161609, -0.33314591, -0.77996058])
```

[134]:
```
covar_double_selection = final_results['final_covariates'].drop('IV').append(pd.
 ↪Index(['gdpsh465']))

double_selection_estimators = pd.Series(0, index=X.columns)
double_selection_estimators[covar_double_selection] =␣
 ↪final_results['final_results'][2][-1]

classic_estimators = pd.Series(classic_results[2][-1], index=X.columns)
```

[155]:
```
estimators = pd.DataFrame({'Double-selection': double_selection_estimators,␣
 ↪'Classic':classic_estimators})
```

[156]:
```
fig, ax =  plt.subplots()
fig.set_size_inches(20, 3)

sns.heatmap(estimators.T.apply(abs), ax=ax, cmap='crest')
```

[156]:
```
<Axes: >
```


Double-selection
Classic
gdpsh465
freeop
h65
hf65
pm65
s65
sf65
mort65
gpop1
mort1
geetot1
gde1
govsh41
high65
highf65
highcm65
human65
humanf65
hyrm65
no65
nof65
pop65
pop1565
sec65
secf65
seccm65
syr65
syrf65
teasec65
im1
tot1
1.00
0.75
0.50
0.25
0.00

[ ]: