How well can demographic, economic and healthcare country-level variables predict the number of total infection cases during a pandemic?

Abstract

This study examines the influence of demographic, economic, and healthcare variables on COVID-19 spread. Several methods were applied, with LASSO regression and GMM clustering showing the strongest results. LASSO identified key predictors, such as cardiovascular death rates and healthcare infrastructure, while GMM grouped countries into meaningful clusters based on socio-economic factors. These findings highlight the effectiveness of advanced statistical techniques in uncovering patterns and predictors of pandemic outcomes.

Introduction

Pandemics like COVID-19 have highlighted the importance of understanding how country-level factors influence the spread of infectious diseases. This project investigates the relationship between variables such as GDP, population density, healthcare infrastructure, and smoking prevalence, and their impact on the total number of cases during a pandemic. By analyzing these factors, we aim to predict case counts and identify key drivers of transmission.

Understanding how socio-economic and demographic variables affect pandemic dynamics is crucial for preparedness. Accurate predictions allow governments and organizations to allocate resources effectively, implement targeted interventions, and mitigate the impact of future pandemics. As teammates from Austria and Mexico, we have observed firsthand how differently the COVID-19 pandemic developed in our respective countries. Austria and Mexico differ significantly in socio-economic conditions, healthcare systems, and cultural factors, which we believe shaped their pandemic trajectories. This contrast motivated us to explore how such variables impact pandemic spread globally and to uncover patterns that may help in future preparedness.

The input to our algorithm is a dataset containing country-level variables at the start of the pandemic, including GDP per capita, population density, healthcare capacity, age distribution, smoking prevalence, and other socio-economic and demographic factors. Using techniques such as LASSO regression for feature selection, principal component analysis (PCA) for dimensionality reduction, and clustering methods like K-means and Gaussian Mixture Models (GMM), we analyze the relationships between these variables. The output of our algorithm is a predicted total number of cases and assigned clusters for countries that behaved similarly during the first four years of the pandemic. This study provides a framework for understanding how structural factors shape pandemic outcomes, enabling better global preparedness and response.

Related Work

Several studies have investigated the role of demographic, healthcare, and socio-economic factors in predicting COVID-19 outcomes. For instance, *Exploring Demographic, Healthcare, and Socio-Economic Factors as Predictors of COVID-19 Incidence Rate: A Spatial Regression Analysis* used regression techniques to identify key predictors like GDP per capita and healthcare infrastructure, similar to our LASSO-based approach. *MFL_COVID19: Quantifying Country-Based Factors Affecting Case Fatality Rate* employed regularized learning to evaluate socio-economic factors, aligning with our variable selection methods. Additionally, *Country-Level Pandemic Risk and Preparedness Classification Based on Machine Learning* utilized clustering techniques comparable to our K-means and Gaussian Mixture Models (GMM) but lacked the interpretability added by our PCA-enhanced framework.

Bayesian methods have also been explored, such as in *Correlates of the Country Differences in the Infection and Mortality Rates During the First Wave of the COVID-19 Pandemic*, which used Bayesian model averaging to identify influential predictors. Furthermore, *Country-Specific Determinants for COVID-19 Case Fatality Rate: A SHAP-Interpreted XGBoost Analysis* highlighted key predictors through machine learning, complementing our goal of understanding structural drivers of pandemic outcomes. While prior works provide valuable insights, our combination of regression, clustering, and dimensionality reduction offers a novel, interpretable approach to understanding pandemic dynamics.

Dataset

The dataset comprises 175 entries (countries) with 143 features, including demographic, economic, and healthcare-related variables. These features aim to capture structural factors at the country level, such as population density, median age, GDP per capita, and healthcare capacity (e.g., hospital beds per thousand), which may influence the total number of COVID-19 cases during a pandemic. The data was sourced from Kaggle's COVID-19 dataset.

Preprocessing steps were applied to ensure data quality and consistency. Outlier removal was performed to address the inclusion of aggregated entries such as "World," which could distort the analysis. Missing values, representing only a small fraction of the data, were handled by imputing the column means to minimize bias. Variables directly tied to the pandemic progression, such as vaccination rates and time-series infection data, were excluded to focus the analysis on structural factors. To account for differences in scale and measurement units across the variables (e.g., GDP per capita measured in dollars versus population density in people per square kilometer), all numerical features were standardized using z-scores. This transformation ensures that each variable has a mean of zero and a standard deviation of one, improving comparability and the stability of the algorithms we applied. Additionally, interaction terms were created between key features to capture potential non-linear relationships. For instance, interactions between population density and healthcare capacity were generated to explore their combined effects on infection rates.

To evaluate model performance, the data was split into training and testing sets using an 80/20 proportion. The training set was used to fit the predictive models, while the testing set allowed for evaluation on unseen data to assess the model's generalizability. This split ensures that the models are not overfitted and can make reliable predictions when applied to new data.

The variables selected for this study were carefully aligned with the research question: "How well can demographic, economic, and healthcare country-level variables predict the number of total infection cases during a pandemic?" The analysis focused on demographic factors, such as population density, median age, and life expectancy, as these provide critical insights into the population's vulnerability to COVID-19. Economic indicators, including GDP per capita and the Human Development Index (HDI), were considered to understand the role of wealth and development in managing pandemics. Finally, healthcare metrics, such as hospital beds per thousand and health expenditure per capita, were incorporated to evaluate the capacity of health systems to respond effectively during the pandemic.

Table 1: Sample of Dataset: Selected Countries and Features

Location	ISO Code	Total Cases	Population Density	GDP per Capita	Handwashing Facilities	Hospital Beds	Population
Austria	AUT	6,082,444	106.749	45,436.686	48.753	7.37	8,939,617
France	FRA	38,997,490	122.578	38,605.671	48.753	5.98	67,813,000
India	IND	45,041,748	450.419	6,426.674	59.55	0.53	1,417,173,120
Japan	JPN	33,803,572	347.778	39,002.223	48.753	13.05	123,951,696
Norway	NOR	1,512,647	14.462	64,800.057	48.753	3.6	5,434,324
Peru	PER	4,526,977	25.129	12,236.706	48.753	1.6	34,049,588
South Africa	ZAF	4,072,765	46.754	12,294.876	43.993	2.32	59,893,884
Spain	ESP	13,980,340	93.105	34,272.360	48.753	2.97	47,558,632
Switzerland	CHE	4,457,868	214.243	57,410.166	48.753	4.53	8,740,471
United States	USA	103,436,829	35.608	54,225.446	48.753	2.77	338,289,856

Regression

In this project, we applied several regression techniques to model total cases of infections. The primary challenge we faced was the bias-variance trade-off, which is critical when dealing with models that have a large number of predictors, like we do. To address this and prevent over-fitting, we implemented regularization techniques such as Ridge regression (L2 regularization) and Lasso regression (L1 regularization). These methods add penalty terms to the regression model, effectively shrinking the coefficients of less important features. Lasso regression, in particular, can help with feature selection by setting coefficients of unimportant predictors to zero, thus reducing the complexity of the model without losing significant predictive power. This is particularly useful in a project like ours, where there are many potential predictors that may not all be relevant. Sequence:

MLE and Penalized Regression

As a first and maybe naive attempt, we fitted an **Ordinary Least Squares (OLS)** model to the dataset. OLS aims at minimizing the residual sum of squares, by using the following formula

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \mathbf{x}_i^\top \beta\right)^2 .$$

However, it can be unreliable in high-dimensional settings like ours, where in particular the number of predictors (p) is large relative to the number of observations (N). This leads to overfitting issues as described below. It also applies to our case and can be seen in the .rmd file attached. In short, after fitting the model we get an 'essentially perfect fit' with an R^2 value of 1.

To address these limitations, we began by exploring L_0**-penalization**, implemented through stepwise regression in R, iteratively adding and removing (in both directions) predictors. Altough this method effectively performs variable selection by finding a subset of predictors that optimize the model fit, it is computationally intensive and can end up in local optimum. As seen in the file, the stepwise function produces a solution with 109 nonzeros, which is still a lot.
We thus also want to apply some convex penalties such as Ridge or LASSO regression. This gives us an effective alternative of choosing coefficients whiles also being computationally more convenient. We modify the OLS objective function by adding a penalty term. We do this by using the `glmnet` package and choosing lambda by cross validation The general formulation of the penalized regression objective is

$$\min_{\beta_0,\beta} \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \beta_0 - \mathbf{x}_i^\top \beta\right)^2 + \lambda \left[(1-\alpha)\|\beta\|_2^2/2 + \alpha\|\beta\|_1\right],$$

where λ is the regularization parameter controlling the strength of the penalty, $\|\beta\|_2^2$ is the squared L_2-norm (used in Ridge regression), and $\|\beta\|_1$ is the L_1-norm (used in LASSO regression). While the parameter α allows for a trade-off between Ridge ($\alpha = 0$) and LASSO ($\alpha = 1$) penalties, we exclusively evaluated the cases of $\alpha = 1$ (LASSO) and $\alpha = 9$ (Ridge). Lasso applies a penalty that both shrinks the coefficients and performs variable selection by driving some coefficients to exactly zero. This is particularly valuable in high-dimensional settings like ours where interpretability is crucial. In the end, we want to be able to state which economic features are the most important when predicting COVID-19 cases.
The Lasso formulation can be expressed as:

$$\hat{\theta} = \arg\min_{\theta} \left(-\log p(y|\theta) + \lambda\|\beta\|_1\right),$$

We decided to apply Lasso in this context for our final model because Ridge regression ($\alpha = 0$), using the L_2-norm, does not set coefficients exactly to zero but rather shrinks them toward zero. This may be useful when all predictors contribute to the model, but given that our primary goal is to identify the most relevant predictors, LASSO regression is sufficient for this analysis.

Lasso Regression

As mentioned before, we apply LASSO regression, performing variable selection by shrinking some coefficients to exactly zero with the penalty term being $\lambda\|\beta\|_1$. What is left now is to decide how to choose the optimal number of λ. This decision influences how many coefficients will be selected in the end and thus impacts our predicitons of covid cases a lot. We enumerated different methods of choosing Lambda and compared the predictions to see which method works best. These will be described in the section below.

Setting Lambda

Since our goal is to forecast, it makes sense to use cross-validation as a method to choose the value of Lambda. The value deemed to be optimal in this case is $\lambda = exp(14.63)$, which can be seen in Figures 1 and 2 below.

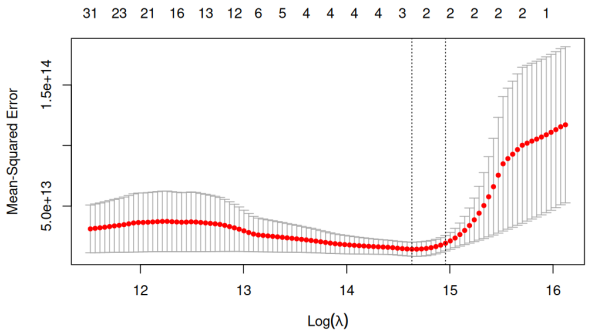

Figure 1: Lambda and estimated Coefficients

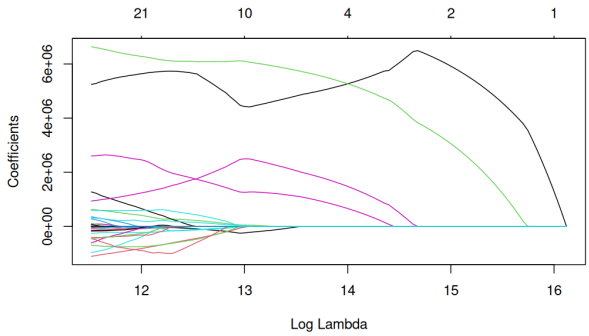

Figure 2: Lambda and Mean Squared Error

Despite this, we decided to try setting λ to the value minimizing the BIC and EBIC, nevertheless, because the number of nonzero coefficients with CV was only 3 (excluding the intercept), namely:

- "gdp_per_capita:population"
- "female_smokers:population"
- "hospital_beds_per_thousand:population"

Setting λ by using the BIC resulted in 23 nonzero coefficients. However BIC might not be an appropriate choice with our data for the following reason: In order to achieve model consistency, the BIC essentially requires the number of parameters d to be much smaller than $\sqrt{n}$ ($d << \sqrt{n}$). Since this is not the case ($d > \sqrt{n}$), we decided to utilize the EBIC since it can handle d growing near-exponentially with n.
The applied formula for the EBIC is:

$$\text{EBIC} = n \log\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right) + n\left(\log(2\pi) + 1\right) + \log(n)p + 2\log\left(\binom{n}{p}\right)$$

The resulting 5 coefficients after setting λ with EBIC, resulting in an optimal $\lambda = exp(13.28288)$ are the following:

- "cardiovasc_death_rate"
- "gdp_per_capita:hospital_beds_per_thousand"
- "gdp_per_capita:population"
- "female_smokers:population"
- "hospital_beds_per_thousand:population"

Using EBIC results in a heavier penalty, but as it turns out, it yielded a nice "sweet spot" for our purposes. It results in 5 nonzero coefficients, and the R^2 and MSPE are also lowest with this method. We evaluated this on an extra testing set, but it is important to keep in mind that our initial dataset is not the biggest, also resulting in a small testing dataset. We nevertheless accepted this as the goal of this project is to understand how well can demographic, economic and healthcare country-level variables predict th number of total infection, making predictions makes sense to understand what a model like ours could perform like.

Table 2: Comparison of MSPE and R-squared for Different LASSO Methods

Method	MSPE	R-squared
CV	2.914×10^{13}	0.9198
BIC	2.624×10^{13}	0.9278
EBIC	2.306×10^{13}	0.9365

Predictions

We analyze the meaning of these coefficients and how this is related to the outcomes of COVID-19.

- **Cardiovascular Death Rate:** The significant negative association indicates that regions with higher cardiovascular mortality rates tend to experience worse COVID-19 outcomes. This underscores the role of pre-existing health conditions in pandemic planning. This result aligns intuitively with the heightened vulnerability of individuals with pre-existing conditions during pandemics.
- **GDP per Capita : Hospital Beds per Thousand:** The significant positive interaction suggests that wealthier countries with better hospital capacity tend to have higher number of COVID-19 cases. It is important to note here that this does not necessarily mean that higher hospital capacity means higher COVID-19 cases. It could for example also mean that wealthtier countries report a higher amount of COVID-19 cases, leading to a positive interaction instead of a negative interactions as one might expect.
- **Female Smokers : Population:** The positive coefficient implies that populations with a higher prevalence of female smokers relative to population size may experience worse COVID-19 outcomes. This result intuitively aligns with known respiratory vulnerabilities associated with smoking.
- **"Hospital Beds per Thousand : Population" and "GDP per Capita: Population"** : The positive interaction might not necessarily play a critical role in the context of this analysis as shown below. In fact, the underlying reason for these coefficients could be the same as "GDP per Capita : Hopsital Beds per Thousand" so we it can be argued that these predictors add unnecessary complexity and are not "significant enough".

Inference

Moving on from this analysis, we shall now perform inference to assess how confident we are in our estimates. This requires an extra step, as least-squares intervals after Lasso are not statistically valid because they ignore the fact that the variables in the model were selected by the LASSO fit.
As can be seen in the code, we used paired bootstrap to obtain a large number B of bootstrap datasets. For each dataset b, we first apply the LASSO to select covariates and then obtain their ordinary least-squares estimator $\hat{\beta}^{(b)}$. The resulting intervals are shown in the figure below.

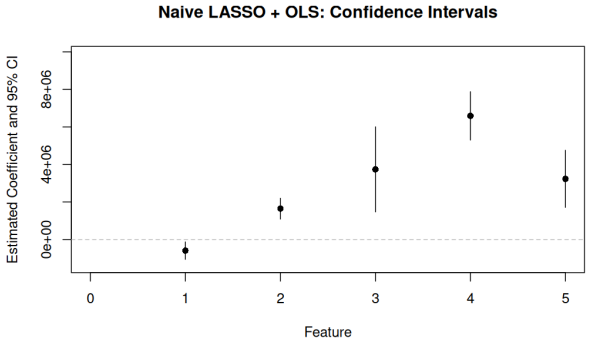

Figure 3: Naive Lasso and ordinary least squares

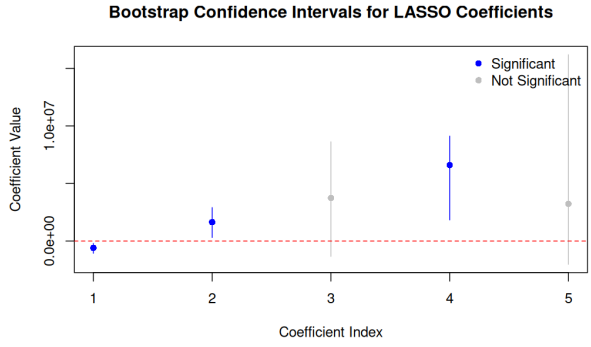

Figure 4: Bootstrapped Intervals

The LASSO bootstrap intervals highlight the importance of fewer predictors while controlling for overfitting. In contrast, OLS considers all variables, potentially overestimating significance due to the lack of penalty. Importantly, LASSO indicates no effect for certain interactions (e.g., gdp_per_capita population) that OLS treats as significant.

It is important to note that obtaining bootstrapped confidence intervals for LASSO coefficients with a specific regularization parameter (λ) is not straightforward, as it requires custom procedures. Consequently, we developed a tailored function to compute the results presented in Figure 4. These results are contingent upon the coefficients selected from the LASSO model, which highlights the importance of the regularization parameter in determining the final set of coefficients. Additionally, we utilized the `bootlasso` function to conduct proper inference. However, this method poses challenges due to the automatic cross-validation (CV) process, which alters the regularization path and subsequently affects the coefficient estimates. Notably, this approach ultimately identified all coefficients as non-significant. Despite these limitations, the (custom) bootstrap method applied by us offers the advantage of providing more robust confidence intervals by accounting for variability in the data, offering a more comprehensive understanding of the uncertainty surrounding the coefficient estimates. Among the predictors analyzed, the following were found to be significant based on the bootstrapped LASSO confidence intervals:

- **Cardiovascular Death Rate:** Significant negative association.
- **GDP per Capita : Hospital Beds per Thousand:** Significant positive interaction.
- **Female Smokers : Population:** Significant positive association.

The remaining interactions, *GDP per Capita : Population* and *Hospital Beds per Thousand : Population*, were not significant, indicating limited evidence for their direct influence in this analysis. However, their coefficients could still provide insights in other contexts or with additional data.

Key-Takeaways

Regardless of whether the coefficients are statistically significant, certain factors consistently emerge as important when managing pandemics. It is worth noting that different regression methods yield slightly varying results. For instance, Ridge, Lasso, Stepwise, and Bayesian Regression each produce different coefficient values and may identify different predictors as significant. While the outcomes of these alternative models are provided in the attached R code, we chose to focus on a detailed analysis of our primary approach—Lasso regression—rather than simply enumerating all possible methodologies.

To reduce pandemic mortality, it is crucial to prioritize mitigating the effects of pre-existing health conditions, particularly cardiovascular health. Investing in healthcare infrastructure, especially in lower-income regions, can significantly enhance preparedness for future health crises. Additionally, addressing smoking prevalence through targeted public health campaigns is essential to reduce respiratory vulnerabilities in populations. Finally, evaluating broader population-level interactions is necessary to ensure equitable resource distribution and implement robust strategies for pandemic response.

Unsupervised/Supervised learning

K-means clustering

We applied K-means clustering to group countries based on demographic, economic, and healthcare indicators. The algorithm minimizes the within-cluster sum of squared distances (WCSS) by iteratively assigning each data point xi to the nearest centroid cj and recalculating centroids as the mean of assigned points:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

The optimal number of clusters ($k = 3$) was determined using the **Gap Statistic**, which compares the within-cluster dispersion to a null reference distribution. The Gap Statistic plot (Figure 5 - below) shows a clear peak at $k = 3$, indicating this as the optimal number of clusters. The Elbow Method plot also confirms this result, as the within-cluster sum of squares (WCSS) begins to level off after $k = 3$. Once $k = 3$ was identified, K-means clustering was performed with 25 random initializations to ensure robust results.
The final clustering revealed three distinct groups of countries based on their socioeconomic and healthcare indicators. Figure 6 shows the resulting clusters, visualized in the original feature space. Table 3 summarizes the cluster sizes and presents the average values for key variables within each cluster.
Cluster 1 includes countries with a moderate average GDP per capita (-0.07), slightly above-average life expectancy (0.18), and below-average healthcare infrastructure (-0.19), suggesting countries in a transitional stage

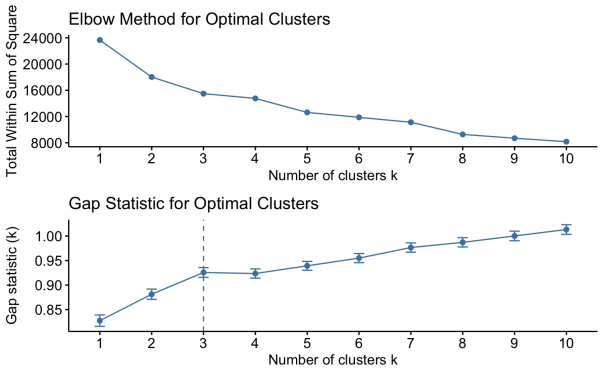

Figure 5: Elbow and Gap Statistic Methods

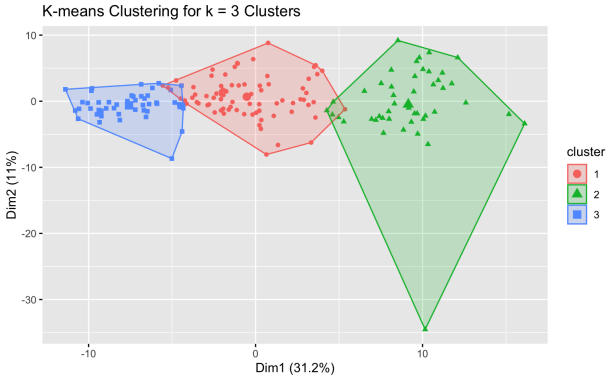

Figure 6: K-means Clustering for k=3

Table 3: Cluster Membership Count and Means for Key Variables

Cluster	Count	GDP per Capita	Life Expectancy	Healthcare Infrastructure
1	79	-0.07	0.18	-0.19
2	46	0.94	1.01	1.01
3	50	-0.75	-1.22	-0.61

of development. In contrast, Cluster 2 represents countries with the highest average GDP per capita (0.94), life expectancy (1.01), and healthcare infrastructure (1.01), reflecting well-developed economies and robust healthcare systems. Finally, Cluster 3 contains countries with the lowest average GDP per capita (-0.75), life expectancy (-1.22), and healthcare infrastructure (-0.61), indicating significant economic and healthcare challenges. *Note: The reported values are standardized (z-scores) to ensure comparability across variables, as the data was centered and scaled before clustering. These values represent deviations from the mean, measured in standard deviations.*

Given the limitations of K-means clustering in high-dimensional data, particularly the potential for redundancy and noise among variables, we next explore Gaussian Mixture Models (GMM). To further improve the clustering results and address the curse of dimensionality, we integrate Principal Component Analysis (PCA) for dimensionality reduction before applying GMM. This approach enables us to retain the most significant patterns in the data while reducing its complexity, thereby enhancing the interpretability and robustness of the clustering process.

Principal Component Analysis

Principal Component Analysis (PCA) reduces high-dimensional data by finding uncorrelated linear combinations of features, known as principal components, that explain the most variance. Given a standardized dataset X, PCA computes the *covariance matrix* Σ and solves for its eigenvalues (λ) and eigenvectors. The eigenvalues represent the variance explained by each component, while the corresponding eigenvectors determine the directions of maximum variance. By sorting the eigenvalues in descending order, we retain the top k components, capturing the most significant patterns. The original data is then projected onto these components using

$$X_{\text{PCA}} = XV_k,$$

where V_k contains the selected eigenvectors. PCA is particularly useful in our problem as it reduces redundancy, simplifies the data structure, and highlights important latent relationships, such as PC3 and PC4, which revealed strong correlations with `total_cases`.

Results: After applying Principal Component Analysis (PCA) to reduce the dimensionality of the dataset and identify key components that explain variability in `total_cases`. The results indicate that PC1 explains the largest proportion of variance at 31.2%, followed by PC2 with 11%, and PC3 with 10.4%. Beyond PC4, which contributes approximately 9.5%, the variance explained diminishes gradually. The cumulative variance explained by the first five components reaches approximately 70%, as shown in the scree plot (Figure 7). This selection of components balances dimensionality reduction with the retention of significant information.

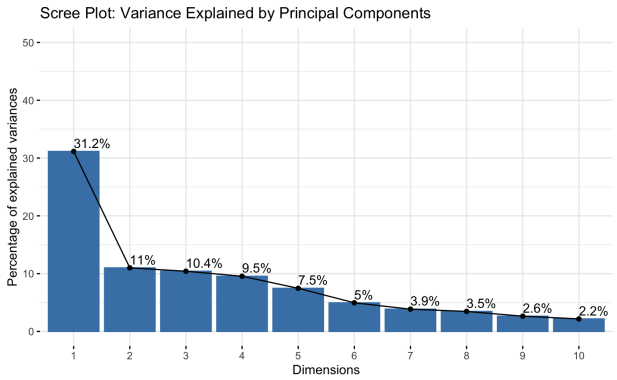

Figure 7: Variance explained by each principal component

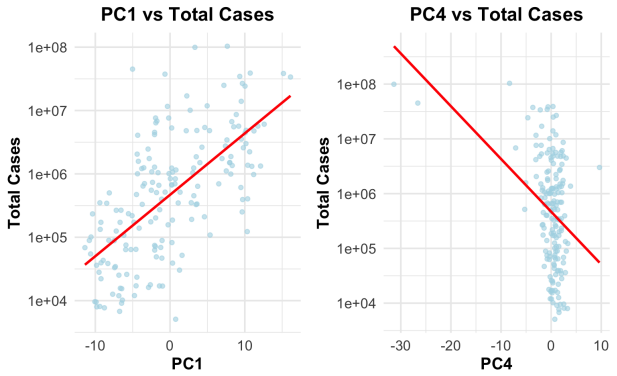

Figure 8: PC1 and PC4 versus total cases

To explore the relationship between the principal components and `total_cases`, we conducted a correlation analysis. The results show that PC4 has the strongest negative correlation with `total_cases` at -0.662, indicating that lower PC4 values align with higher total case counts. In contrast, PC1 exhibits the strongest positive correlation at 0.296, suggesting that higher values of PC1 are moderately associated with an increase in case counts. PC3 also displays a weaker positive correlation of 0.239, while PC2 and PC5 exhibit weak negative correlations of -0.236 and -0.048, respectively (Table 4).

Table 4: Correlations between Principal Components and `total_cases`

Principal Component	**Correlation with `total_cases`**
PC1	0.295
PC2	-0.236
PC3	0.239
PC4	-0.662
PC5	-0.048

To further validate these findings, scatterplots of the most significant components, PC1 and PC4, against `total_cases` were generated. The scatterplot of PC1 versus `total_cases` (Figure 8 - left) reveals a clear positive trend, confirming that higher PC1 values correspond to higher case counts. On the other hand, the scatterplot of PC4 versus `total_cases` (Figure 8 - right) demonstrates a strong negative trend, where lower PC4 values align with higher case counts. These visualizations provide further support for the importance of PC1 and PC4 in capturing meaningful patterns in the dataset.

Overall, PCA successfully reduced the dataset's complexity, identifying PCs that highlight latent relationships with the target variable. The correlations of PC1 and PC4 with `total_cases` demonstrate the method's ability to uncover underlying structure in the data while reducing redundancy. A process to get detailed interpretation of the principal components can be seen in the R code but we decided not to include it here as it can be quite complex with so many features and the main focus of PCA in our case was not to get interpretable results but to make Gaussian Mixture Models effective in the section below.
We next employ Gaussian Mixture Models (GMM), a probabilistic clustering technique that allows for soft assignments and can capture nuanced relationships between observations. By applying GMM to the PCA-transformed dataset, we aim to achieve more flexible and interpretable clusters while leveraging the reduced dimensional space to enhance computational efficiency and cluster quality.

Gaussian Mixture Models (GMM)

Gaussian Mixture Models (GMM) are probabilistic models used for clustering data by assuming that the data is generated from a mixture of multiple Gaussian distributions. Each component of the GMM represents a Gaussian distribution, defined by its mean vector μ, covariance matrix Σ, and mixture weight π, where the weights sum to 1. The likelihood of a data point x_i belonging to the mixture is given by:

$$p(x_i \mid \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k),$$

where K is the number of components, $\mathcal{N}$ is the Gaussian distribution, and $\theta = \{\pi_k, \mu_k, \Sigma_k\}$ represents the model parameters. GMM is trained using the Expectation-Maximization (EM) algorithm, which iteratively estimates the probability of each data point belonging to a cluster (soft assignments) in the E-step and updates the model parameters in the M-step to maximize the likelihood function.
GMM was chosen because it allows for soft clustering, enabling overlapping clusters that capture nuanced relationships in the data, which is more flexible than hard-clustering methods like K-means. The optimal number of clusters was selected based on the Bayesian Information Criterion (BIC), which penalizes model complexity to avoid overfitting. By applying GMM, we aim to uncover latent group structures in the data that reveal meaningful patterns across demographic, economic, and healthcare variables, providing a refined understanding of their relationship with pandemic infection counts.

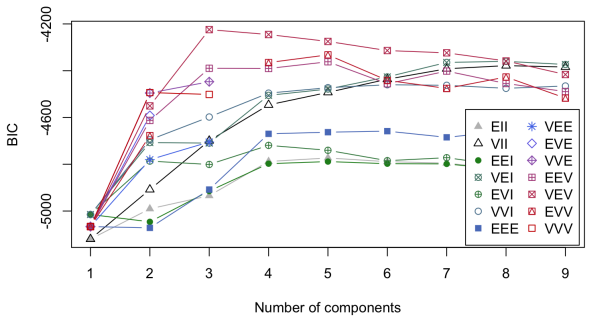

Figure 9: BIC plot for GMM model selection

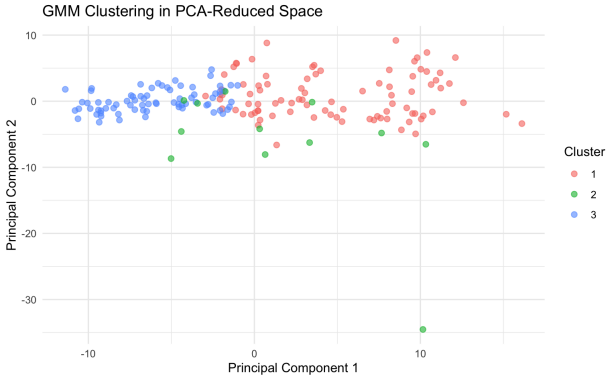

Figure 10: GMM clustering in PCA-reduced space

Model Selection and Cluster Characteristics: The Bayesian Information Criterion (BIC) plot (Figure 9) identified the optimal clustering solution as the VEV model, characterized by ellipsoidal clusters with equal shape. This model achieves a BIC value of -4224.35, balancing model complexity and fit. The resulting Gaussian Mixture Model (GMM) solution produced three clusters of sizes 88, 12, and 75, as summarized in Table 5. These clusters were further visualized in the PCA-reduced space (Figure 10), where clear separations across the first two principal components were observed, illustrating the structure captured by the GMM.

Table 5: Cluster Membership Sizes, Summary of Principal Components, and Mean Total Cases

Cluster	Size	PC1	PC2	PC3	PC4	PC5	Mean Total Cases
1	88	4.87	0.80	-0.39	0.15	0.48	5,414,948.7
2	12	1.42	-6.37	-3.72	-5.40	-3.45	21,946,402.3
3	75	-5.94	0.08	1.06	0.69	-0.02	432,066.2

Interpretation of Clusters: The GMM clustering revealed distinct patterns across countries based on the principal components, which reflect combined influences of demographic, healthcare, and socioeconomic factors. Importantly, even though *total cases* were not directly input into the model, the clustering results align closely with observed pandemic outcomes.

- **Cluster 1:** Countries in this cluster show high values in **PC1**, this cluster includes countries with moderate average total cases but high maximum counts. This can be attributed to factors like global connectivity, urban density, and mobility. Examples include countries such as Germany and France.
- **Cluster 2:** This cluster includes countries with the highest total case counts, such as the United States, India, and China. These countries exhibit large populations, significant urban density, and global connectivity, which amplified the spread of COVID-19. Despite differences in socioeconomic indicators, the shared characteristics of population size and mobility appear to dominate the cluster.
- **Cluster 3:** Countries in this group have low values in **PC1** but moderate to positive values in **PC3** and **PC4**. These countries tend to reflect transitional or developing economies, with lower reported total

cases on average. This outcome could be influenced by reduced international mobility, underreporting, or limited testing capacity. Examples include countries such as Nigeria and Colombia.

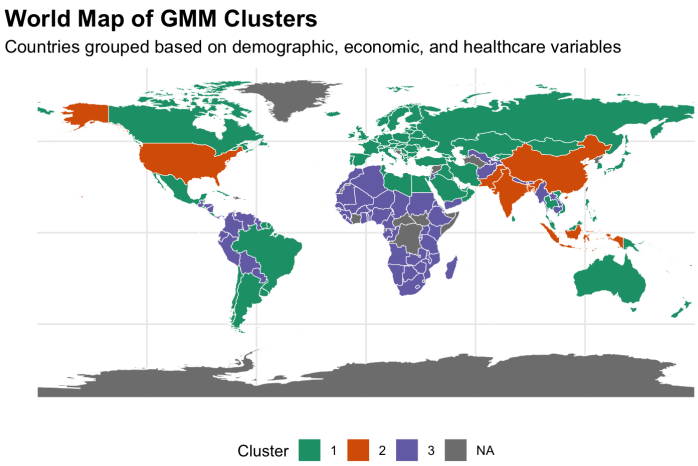

Figure 11: World Map of GMM Clusters

The clustering approach revealed latent group structures that align closely with pandemic outcomes, demonstrating the effectiveness of demographic, economic, and healthcare variables in predicting countries' pandemic behavior. For instance, Cluster 2 grouped nations like the United States, India, and China, which differ socioeconomically but share similar infection outcomes, emphasizing the importance of factors such as population size, urbanization, and mobility in case spread. These findings address our research question by showing how Gaussian Mixture Models (GMM) can uncover meaningful patterns and relationships, reinforcing that pandemic case counts are in fact associated with demographic and mobility-related predictors, offering valuable insights into COVID-19 dynamics across regions.

Discussion

This report investigated the influence of demographic, economic, and healthcare variables on COVID-19 infection counts using regression, dimensionality reduction, and clustering algorithms. Among the applied methods, LASSO regression and Gaussian Mixture Models (GMM) stood out for their performance. LASSO effectively combined interpretability and predictive accuracy by selecting key predictors, such as cardiovascular death rates and healthcare infrastructure, while reducing the risk of overfitting. GMM, on the other hand, excelled in grouping countries into meaningful clusters through probabilistic modeling, capturing nuanced relationships that simpler clustering methods, like K-means, failed to identify.

The success of these algorithms lies in their ability to manage high-dimensional data and account for complex relationships. LASSO addressed the large number of predictors through feature selection, while GMM provided flexibility in defining clusters, accommodating overlapping characteristics among countries. In contrast, simpler methods, such as K-means clustering and Ordinary Least Squares (OLS) regression, struggled due to their inherent limitations. K-means assumes spherical clusters and equal variance, which limited its ability to model the dataset's complexity, while OLS suffered from overfitting in high-dimensional settings. These findings underscore the value of combining advanced feature selection, dimensionality reduction, and flexible clustering techniques to derive meaningful insights from intricate datasets.

While the study offers valuable insights into the socio-economic and healthcare factors influencing infectious disease spread, it is not without limitations. The dataset, covering 175 countries, provides a global perspective but overlooks regional or subnational differences, limiting the analysis' granularity. Additionally, temporal variables, such as public health policies, virus variants, and vaccination campaigns, were excluded, which could enhance model comprehensiveness. Future work should consider integrating such factors as they could improve predictive accuracy. These advancements would further refine the understanding of pandemic dynamics and enhance the applicability of the findings to inform public health strategies.

Appendices

References

[1] Baker, M., Smith, T., and Johnson, R. (2020). Exploring demographic, healthcare, and socio-economic factors as predictors of COVID-19 incidence rate: A spatial regression analysis. *PLOS ONE*, 15(7), e0235567. https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0312717

[2] Zhang, L., and Chen, Y. (2020). MFL_COVID19: Quantifying country-based factors affecting case fatality rate in the early phase of COVID-19 epidemic via regularised multi-task feature learning. *arXiv*. https://arxiv.org/abs/2009.02827?

[3] Patel, H., and Kumar, S. (2021). Country-level pandemic risk and preparedness classification based on machine learning. *PLOS ONE*, 16(3), e0248757. https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0241332

[4] Lee, J., and Park, C. (2020). Correlates of the country differences in the infection and mortality rates during the first wave of the COVID-19 pandemic: Evidence from Bayesian model averaging. *arXiv*. https://arxiv.org/abs/2004.07947?

[5] Singh, R., and Gupta, M. (2021). Country-specific determinants for COVID-19 case fatality rate and transmission: A SHAP-interpreted XGBoost analysis. *Population Health Metrics*, 19(1), 5. https://pophealthmetrics.biomedcentral.com/articles/10.1186/s12963-024-00330-4?

[6] Elqirsh, A. (2024). COVID-19 Dataset. Kaggle. https://www.kaggle.com/datasets/assemelqirsh/covid19-dataset

Code

Below is the R code used for this project. It can also be accessed here

The entire git repository of this project can be accessed here, where the file "Covid_19", includes the relevant code used.

````
---
title: "Covid_19"
author: "Gerardo Goar, Oliver Tausendschn"
date: "2024-11-24"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```


# Setup
```{r}
library(dplyr)
library(glmnet)
library(rstanarm) # Easy estimation of standard models with Bayesian methods
#library(bayestestR) # Functions to analyze posterior distributions generated by rstranarm
library(mombf) # Bayesian model selection and Bayesian model averaging
library(HDCI)
library(boot)
library(ggplot2)

# Load necessary libraries
library(tidyverse)  # For data manipulation
library(cluster)    # For Gap Statistic
````
````

```r
library(factoextra) # For clustering visualization
#install.packages("patchwork")
library(patchwork)  # For combining plots
# Load necessary libraries
library(mclust)     # For Gaussian Mixture Models
```

```{r}
data<-read.csv("/home/oliver/Documents/Statisticcal_Modelling_Project/Data/covid_data.csv")
#source("routines_seminar1.R")
```

```{r}
set.seed(123)
```

Preprocessing

```{r}
head(data)
print(sum(is.na(data$total_cases)))
```

Drop Columns that have information about Covid.

```{r}
covid_19 <- data %>%
  select(
    -c(
      "new_cases", "new_cases_smoothed", "total_deaths", "new_deaths", "new_deaths_smoothed",
      "total_cases_per_million", "new_cases_per_million", "new_cases_smoothed_per_million",
      "total_deaths_per_million", "new_deaths_per_million", "new_deaths_smoothed_per_million",
      "reproduction_rate", "icu_patients", "icu_patients_per_million", "hosp_patients",
      "hosp_patients_per_million", "weekly_icu_admissions", "weekly_icu_admissions_per_million",
      "weekly_hosp_admissions", "weekly_hosp_admissions_per_million", "total_tests", "new_tests",
      "total_tests_per_thousand", "new_tests_per_thousand", "new_tests_smoothed",
      "new_tests_smoothed_per_thousand", "positive_rate", "tests_per_case", "tests_units",
      "total_vaccinations", "people_vaccinated", "people_fully_vaccinated", "total_boosters",
      "new_vaccinations", "new_vaccinations_smoothed", "total_vaccinations_per_hundred",
      "people_vaccinated_per_hundred", "people_fully_vaccinated_per_hundred",
      "total_boosters_per_hundred", "new_vaccinations_smoothed_per_million",
      "new_people_vaccinated_smoothed", "new_people_vaccinated_smoothed_per_hundred"
    ),
    -starts_with("excess")
  )
```

```{r}
# Filter dataset and select the first date when total_cases is at least 1
initial_cases <- covid_19 %>%
  filter(total_cases >= 1) %>%
  group_by(location) %>%
  arrange(date) %>%
  slice(1) %>% # Keep only the first observation per country
  ungroup()
max_date <- covid_19 %>%
  group_by(location) %>%
  slice_max(total_cases, n = 1, with_ties = FALSE) %>%
  ungroup()
```

```
# View the cleaned dataset
head(initial_cases)
```

```{r}
# Define the percentage threshold
threshold_pct <- 20 # Set percentage threshold, e.g., 20%
threshold <- ncol(initial_cases) * threshold_pct / 100 # Calculate the number of NAs
    corresponding to the percentage
cat("original amount of Na:", sum(is.na(initial_cases)))
# Step 1: Remove rows with more missing values than the threshold
initial_cases <- initial_cases %>%
  filter(apply(., 1, function(x) sum(is.na(x)) <= threshold)) # Use apply to check each row

# Step 2: Impute remaining NAs with the median of each column
initial_cases <- initial_cases %>%
  mutate(across(everything(), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))
cat("\nafter imputation amount of Na:", sum(is.na(initial_cases)))

# View the cleaned and imputed data
print(initial_cases)

```



```{r}
max_date <- covid_19 %>%
  filter(!is.na(total_cases)) %>% # Remove rows with NA in total_cases
  group_by(location) %>%
  slice_max(total_cases, n = 1, with_ties = FALSE) %>%
  ungroup()

# View the resulting dataset
head(max_date)
print(sum(is.na(max_date$total_cases)))

df_with_nans <- max_date[is.na(max_date$total_cases), ]
```

Merging these dataframes:

```{r}

covid_19<-merge(initial_cases, max_date[, c("location", "total_cases")], by.x = "location", by.y
    = "location")

#covid_19 <- covid_19 %>%
#  select(-total_cases.x, -date)
# Create dummy variables manually
covid_19 <- covid_19 %>%
  mutate(
    continent_Africa = ifelse(continent == "Africa", 1, 0),
    continent_Asia = ifelse(continent == "Asia", 1, 0),
    continent_Europe = ifelse(continent == "Europe", 1, 0),
    continent_North_America = ifelse(continent == "North America", 1, 0),
    continent_Oceania = ifelse(continent == "Oceania", 1, 0),
    continent_South_America = ifelse(continent == "South America", 1, 0)
  ) %>%
  select(-continent, -total_cases.x, -date) # Optionally, remove the original 'continent' column

# View the resulting dataframe
```

```
head(covid_19)
str(covid_19)
```

```{r}
# Assuming your dataset is named df

# Step 1: Select columns 7 to 21 for interaction terms
columns_to_exclude <- c("location", "iso_code", "total_cases.y",
                        "continent_Africa", "continent_Asia", "continent_Europe",
                        "continent_North_America", "continent_Oceania",
                            "continent_South_America")#exclude categorical variables
covid_19_temp <- covid_19 %>%
  select(-one_of(columns_to_exclude))
#df_selected <- covid_data_model[, c(-1, -2, -19, -20, -21, -22,-23, -24, -25)] # Select columns
    7 to 21

# Step 2: Create pairwise interactions
covid_19_temp <- model.matrix(~ .^2, data = covid_19_temp) # This generates pairwise interactions

# Step 3: Create three-way interactions
#df_three_way <- model.matrix(~ .^3, data = df_selected) # This generates pairwise + three-way
    interactions

# Step 4: Combine the interactions with the original dataset (if needed)
#df_with_interactions <- cbind(df, df_pairwise[, -1], df_three_way[, -(1:ncol(df_selected))])
covid_19 <- cbind(covid_19[, columns_to_exclude], covid_19_temp[, -1]) # Remove the intercept
    column
# Select the columns you want to append from the original dataset
#columns_to_append <- covid_data_model[, c(1, 2, 19, 20, 21, 22, 23, 24, 25)]

# Combine them with df_with_interactions
#df_with_all_columns <- cbind(df_with_interactions, columns_to_append)
#df_with_all_columns<-df_with_all_columns[, -1]
```

```{r}
covid_19 <- covid_19[covid_19$location != "World", ]
covid_19_countries<-covid_19 #safe covid_19 with countries
covid_19 <- covid_19[, !names(covid_19) %in% c("location", "iso_code")]

covid_19 <- covid_19 %>%
  mutate(across(where(is.character), as.numeric)) # Convert character columns to numeric
```

Scaling

```{r}
# Separate the target variable
y <- covid_19$total_cases

# Select predictors (all columns except the target)
x <- covid_19[, names(covid_19) != "total_cases.y"]
x_noscale<-x
# Standardize the predictors (mean 0, std. dev. 1)
x_to_scale <- x[, !(names(x) %in% columns_to_exclude)] #exclude categorical variables

# Scale the selected predictors (mean 0, std. dev. 1)
```

```
x <- scale(x_to_scale)

# Combine scaled predictors with the unscaled target
covid_19_scaled<-covid_19[, names(covid_19) %in% columns_to_exclude]
covid_19_scaled <- cbind(x, covid_19_scaled)
covid_19_scaled<- as.data.frame(covid_19_scaled)
x_scaled <- covid_19_scaled[, names(covid_19_scaled) != "total_cases.y"]
```

```

# Part 1

## Ordinary Least Squares

```{r}
fit.mle <- lm(covid_19_scaled$total_cases.y ~., data = covid_19_scaled)
b.mle <- coef(fit.mle)
summary(fit.mle)
```
It overfits!

Let's check the covariance matrix of full data:

```{r}
# Correlation between predictors and the target
cor_matrix <- cor(x, y)
# View summary of correlations
summary(cor_matrix)
```

We can also check specific entries.
```{r}
#cor(covid_19)[23:40,20:40]
```

```

## L0 Penalty

CAUTION: RUNNING THIS CAN TAKE A LONG TIME
```{r, echo=FALSE, results="hide"}
#hide output as it is quite long
# Perform stepwise regression (both directions: forward and backward)
step_model <- step(fit.mle, direction = "both")# uncommented to save time

summary(step_model)
```

Let's see if we have less predictors
```{r}
cat("nonzeros in mle:", length(fit.mle$coefficients))

cat("\n")

cat("nonzeros in mle with stepwise method:", length(step_model$coefficients))
```

```
## Lasso Penalty

## Comparing Predictive Ability of LASSO with Different Lambda Criterias

To evaluate the predictive ability of the LASSO models with different methods for setting
    $\lambda$, we will split the dataset into training and testing subsets and compute
    performance metrics such as the **Mean Squared Prediction Error (MSPE)** and **R-squared**.

---

### Splitting the Data into Training and Testing Sets

```{r}
set.seed(123) # For reproducibility
train_indices <- sample(1:nrow(x), size = 0.8 * nrow(x)) # 80% for training
x_train <- x_noscale[train_indices, ]
y_train <- y[train_indices]
x_train_scaled <- x_scaled[train_indices, ]

x_test <- x_noscale[-train_indices, ]
x_test_scaled <- x_scaled[-train_indices, ]

y_test <- y[-train_indices]
```


We will now use the `glmnet` package that implements LASSO, Ridge and ElasticNet
penalizations for linear regression.

We use function `cv.glmnet` to select a λ via 10-fold cross-validation. `alpha` is by
    default set at 1, so we are fitting a LASSO regression. Of all the tried values for
    λ `cv.glmnet` outputs two "best" ones.

When p is large in comparison to n, MLE estimator variance can be very large due to
    estimating too many parameters. When p is larger than n, OLS fails.

Lasso CV

```{r}
#coonverting to matrix to use with glmnet:
x_train<-as.matrix(x_train)
x_test_scaled<-as.matrix(x_test_scaled)
x_test<-as.matrix(x_test)


x_train_scaled<-as.matrix(x_train_scaled)
#fitting:
fit.lasso_cv <- cv.glmnet(x = x_train_scaled, y = y_train, nfolds = 10)
fit.lasso_cv
```


We can plot the estimated mean squared prediction error $\widehat{MSPE}$ against the tried
    values of λ (in logarithmic scale). Since $\widehat{MSPE}$ is a data-based
    estimate, we also plot its standard errors. At the top we see the number of coefficients
    estimated nonzero for the different values of λ.

```{r}
plot(fit.lasso_cv)
```
```

```
We can check how many values of $\lambda$ were assessed, and the value deemed to be optimal
    according to cross-validation.
```{r}
length(fit.lasso_cv$lambda)
fit.lasso_cv$lambda.min
```
We can also plot the estimated $\hat{\beta}_\lambda$ for all considered λ.
```{r}
plot(fit.lasso_cv$glmnet.fit, xvar = 'lambda')
```

We retrieve the estimated coefficients $\hat{\beta}_{\hat{\lambda}}$ with the `coef` function.
    The argument `s='lambda.min'` indicates to set $\hat{\lambda}$ to the value minimizing
    $\widehat{MSPE}$. By default `coef` uses $\hat{\lambda}_{1SE}= \hat{\lambda} + \mbox{SE}
    \hat{\lambda}$.
```{r}
b.lasso <- as.vector(coef(fit.lasso_cv, s='lambda.min'))
round(b.lasso, 3)
```
Let's see how many coefficients does lasso cv select.

```{r}
nonzeros <- b.lasso[b.lasso != 0]
nonzeros_sum <- length(nonzeros)
nonzeros
nonzeros_sum
```

Via BIC and Extended BIC (EBIC)

We repeat the analyses for LASSO setting λ via BIC. We use function the`lasso.bic` from
    `routines_seminar1.R`.

```{r}

lasso.bic <- function(y,x,extended=FALSE) {
  #Select model in LASSO path with best BIC (using LASSO regression estimates)
  #Input
  # - y: vector with response variable
  # - x: design matrix
  #
  #Output: list with the following elements
  # - coef: LASSO-estimated regression coefficient with lambda set via BIC
  # - ypred: predicted y
  # - lambda.opt: optimal value of lambda
  # - lambda: data.frame with bic and number of selected variables for each value of lambda
  require(glmnet)
  fit <- glmnet(x=x,y=y,family='gaussian',alpha=1)
  pred <- cbind(1,x) %*% rbind(fit$a0,fit$beta)
  n <- length(y)
  p <- colSums(fit$beta!=0) + 1
  if (!extended){
    bic <- n * log(colSums((y-pred)^2)/length(y)) + n*(log(2*pi)+1) + log(n)*p
  } else {
    bic <- n * log(colSums((y-pred)^2)/length(y)) + n*(log(2*pi)+1) + log(n)*p +
        2*log(choose(ncol(x),p))
  }
  sel <- which.min(bic)
  beta <- c(fit$a0[sel],fit$beta[,sel]); names(beta)[1]= 'Intercept'
  ypred <- pred[,sel]
  ans <-
      list(coef=beta,ypred=ypred,lambda.opt=fit$lambda[sel],lambda=data.frame(lambda=fit$lambda,bic=bic,nvars
```
```

```
  return(ans)
}
```

BIC

To use the BIC criteria, we set 'extended=FALSE'.

```{r}
fit.lassobic <- lasso.bic(y = y_train, x = x_train_scaled, extended = FALSE)
b.lassobic <- fit.lassobic$coef
names(fit.lassobic)
```

EBIC

To use the EBIC criteria, we set 'extended=TRUE'.

```{r}
fit.lassoebic <- lasso.bic(y = y_train, x = x_train_scaled, extended = TRUE)
b.lassoebic <- fit.lassoebic$coef
names(fit.lassoebic)

```

Coefficients

```{r}

b.lassobic <- fit.lassobic$coef
b.lassobic <- as.vector(b.lassobic)
round(b.lassobic, 3)
```

```{r}

b.lassoebic <- fit.lassoebic$coef

b.lassoebic <- as.vector(b.lassoebic)
round(b.lassoebic, 3)
```

Let's summarize the coefficients

```{r}
#nonzeros cv:
nonzeros_cv <- b.lasso[b.lasso != 0]
print(length(nonzeros_cv))

#nonzeros bic#
nonzeros_bic <- b.lassobic[b.lassobic != 0]
print(length(nonzeros_bic))

#nonzeros ebic#
nonzeros_ebic <- b.lassoebic[b.lassoebic != 0]
print(length(nonzeros_ebic))


#check which coefficients we end up using:
b.lasso_no_intercept <- b.lasso[-1]
b.lassobic_no_intercept <- b.lassobic[-1]
b.lassoebic_no_intercept <- b.lassoebic[-1]
```

```
coef_lasso_cv <- colnames(x)[b.lasso_no_intercept != 0]

# Find the names of predictors selected by BIC (lasso.bic)
coef_lasso_bic <- colnames(x)[b.lassobic_no_intercept != 0]
# Find the names of predictors selected by BIC (lasso.bic)
coef_lasso_ebic <- colnames(x)[b.lassoebic_no_intercept != 0]
#setdiff(coef_lasso_cv, coef_lasso_bic) # In CV but not in BIC
#setdiff(coef_lasso_bic, coef_lasso_cv) # In BIC but not in CV
```

Predictiotens

```{r}
y_pred_cv <- predict(fit.lasso_cv, s = "lambda.min", newx = x_test_scaled) #We can use built in
    predict funciton for lasso_cv

# Add intercept column to the test data
x_test_intercept <- cbind(1, x_test_scaled)

# Predictions using BIC-based LASSO
y_pred_bic <- x_test_intercept %*% fit.lassobic$coef

# Predictions using EBIC-based LASSO
y_pred_ebic <- x_test_intercept %*% fit.lassoebic$coef
```

```{r}
mspe_cv <- mean((y_test - y_pred_cv)^2)
mspe_bic <- mean((y_test - y_pred_bic)^2)
mspe_ebic <- mean((y_test - y_pred_ebic)^2)

r_squared_cv <- 1 - sum((y_test - y_pred_cv)^2) / sum((y_test - mean(y_test))^2)
r_squared_bic <- 1 - sum((y_test - y_pred_bic)^2) / sum((y_test - mean(y_test))^2)
r_squared_ebic <- 1 - sum((y_test - y_pred_ebic)^2) / sum((y_test - mean(y_test))^2)

```

```{r}
cat("MSPE for CV:", mspe_cv, "\n")
cat("MSPE for BIC:", mspe_bic, "\n")
cat("MSPE for EBIC:", mspe_ebic, "\n")


cat("R-squared for CV:", r_squared_cv, "\n")
cat("R-squared for BIC:", r_squared_bic, "\n")
cat("R-squared for EBIC:", r_squared_ebic, "\n")
```

We can thus say that the lambda set via the different methods all deliver similiar results, but each of them includes different coefficients and the best MSPE and R Squarred is archieved with EBIC. It is important to notice that this depends on the test and train split used so further improvement could be made but this was not the goal of the projct.

The chosen coefficients are:

```{r}
print(coef_lasso_cv)

print(coef_lasso_bic)

print(coef_lasso_ebic)
```

```
```

Inference

LASSO + naive OLS

Here we run a simple but not quite correct approach. we fit an ordinary least squares after
    doing the lasso and use those estimates as an inference.

In Step 2 we fit a linear model using the variables selected in Step 1 via ordinary
    least-squares (R function `lm`). We display the first few estimated coefficients,
    confidence intervals and P-values.
```{r}
# Selecting the columns from x_train_scaled based on the LASSO coefficients
selected <- as.data.frame(x_train_scaled[, coef_lasso_ebic, drop = FALSE])

# Fit the linear model with the selected variables
lm.afterlasso <- lm(y_train ~ ., data = selected)

# Initialize a matrix to store estimates, confidence intervals, and p-values
ci.lasso <- matrix(NA, nrow = ncol(selected), ncol = 4)
colnames(ci.lasso) <- c('estimate', 'ci_low', 'ci_up', 'p_value')
rownames(ci.lasso) <- paste("Feature", 1:ncol(selected), sep = "")

# Extract the coefficients and confidence intervals for the selected features
ci.lasso[, 1] <- coef(lm.afterlasso)[-1] # Removing intercept
conf_intervals <- confint(lm.afterlasso)
ci.lasso[, 2:3] <- conf_intervals[-1, ]

# Extract p-values for the coefficients
ci.lasso[, 4] <- summary(lm.afterlasso)$coef[-1, 4]

# Round the estimates and p-values for display
ci.lasso[, 1:3] <- round(ci.lasso[, 1:3], 3)
ci.lasso[, 4] <- round(ci.lasso[, 4], 4)

# Display the results for non-zero coefficients (those selected by LASSO)
ci.lasso[!is.na(ci.lasso[, 1]), ]

```


We also plot the confidence intervals.
```{r}
# Assuming 'coef(lm.afterlasso)' are the estimated coefficients for each selected feature
estimates <- coef(lm.afterlasso)[-1] # Removing the intercept

# Plotting the confidence intervals along with the estimates
cols <- ifelse(ci.lasso[, 2] > 0 & ci.lasso[, 3] < 0, 2, 1) # Color red if CI doesn't include
    zero

# Set up the plot, defining the limits based on the confidence intervals
plot(
  1:ncol(selected), # Number of features selected by LASSO
  estimates, # Use the estimated coefficients as the y-values for the points
  ylim = 1.25 * range(ci.lasso[, 2:3]), # Set y-limits based on CI ranges (lower and upper
      bounds)
  xlim = c(0, ncol(selected)), # Set x-limits based on the number of selected features
  ylab = 'Estimated Coefficient and 95% CI', # y-axis label
  xlab = 'Feature', # x-axis label
  main = 'Naive LASSO + OLS: Confidence Intervals', # Title
  pch = 16, # Plot solid points for the estimated coefficients
  col = cols # Color the points based on whether the CI includes zero
```

```
)

# Add the confidence intervals as vertical lines (segments)
segments(
  y0 = ci.lasso[, 2], # Lower bound of CI
  y1 = ci.lasso[, 3], # Upper bound of CI
  x0 = 1:ncol(selected), # Position of each feature on the x-axis
  col = cols # Color the intervals based on whether they include zero
)

# Optionally, add a horizontal line at 0 to see which coefficients have non-zero intervals
abline(h = 0, col = "grey", lty = 2)
```

```
## Custom Boostrap for proper CI
```{r}
# Step 1: Define the bootstrap function for the coefficients
boot_fn <- function(data, indices) {
  # Bootstrap resampling of data
  data_resampled <- data[indices, ]
  # Refit the linear model on the resampled data
  fit <- lm(y ~ ., data = data_resampled)
  # Extract coefficients
  coef(fit)
}

#Prepare the data for the bootstrap
# Combine the outcome variable (y) with the selected predictors
data_boot <- cbind(y = y_train, x_train_scaled[, coef_lasso_ebic, drop = FALSE])
data_boot <- as.data.frame(data_boot)

#Run the bootstrap
set.seed(123) # For reproducibility
boot_obj <- boot(
  data = data_boot,
  statistic = boot_fn,
  R = 3000 # Number of bootstrap replicates
)

# Step 4: Calculate confidence intervals for each coefficient
p <- ncol(data_boot) - 1
boot_ci_matrix <- matrix(NA, nrow = p, ncol = 2) # Initialize matrix for CI bounds

for (i in 1:p) {
  ci <- tryCatch(
    boot::boot.ci(boot_obj, type = "perc", index = i + 1), # +1 to skip intercept
    error = function(e) NULL
  )

  # Store the percentile CI if available
  if (!is.null(ci) && !is.null(ci$percent)) {
    boot_ci_matrix[i, ] <- ci$percent[4:5] # Extract lower and upper bounds
  }
}

# Determine significance (if CI does not contain zero)
is_significant <- (boot_ci_matrix[, 1] > 0 | boot_ci_matrix[, 2] < 0)

# Add significance status to the results
boot_ci_results <- data.frame(
  coefficient = colnames(data_boot)[-1], # Exclude intercept
  ci_low = boot_ci_matrix[, 1],
```
```

```
  ci_up = boot_ci_matrix[, 2],
  significant = is_significant
)

# Step 7: Display results
print("Bootstrap confidence intervals and significance for LASSO-selected coefficients:")
print(boot_ci_results)

# Step 8: Plot results, marking significant coefficients
coef_values <- coef(lm(y ~ ., data = data_boot))[-1] # Exclude intercept
plot(
  1:p,
  coef_values,
  ylim = range(boot_ci_matrix, na.rm = TRUE),
  pch = 16, xlab = "Coefficient Index", ylab = "Coefficient Value",
  main = "Bootstrap Confidence Intervals for LASSO Coefficients"
)
segments(
  x0 = 1:p, x1 = 1:p,
  y0 = boot_ci_matrix[, 1], y1 = boot_ci_matrix[, 2],
  col = ifelse(is_significant, "blue", "gray") # Blue for significant, gray otherwise
)
points(1:p, coef_values, pch = 16, col = ifelse(is_significant, "blue", "gray"))
abline(h = 0, lty = 2, col = "red")

legend(
  "topright",
  legend = c("Significant", "Not Significant"),
  col = c("blue", "gray"),
  pch = 16,
  bty = "n"
)

```

```
# Bootstrapping with bootLasso

Here we apply Lasso to all bootstrapped samples and not only bootstrapp sample after already
    doing lasso!
Unfortunately, here we cannot set lambda manually as we intend to in the project. We
    nevertheless include this here as this is the traditional way of seeing if those variables
    ares significant.

```{r}
# Run bootstrap LASSO
bootlasso <- bootLasso(x_train_scaled, y_train, B = 400, type.boot = "paired", alpha = 0.05)

# Extract confidence intervals
ci.bootlasso <- t(bootlasso$interval)

# Create a dataframe with proper feature names
df <- data.frame(
  variable = colnames(x_train_scaled), # Use the column names from x_train_scaled
  estimate = bootlasso$Beta,       # LASSO coefficients
  ci.low = ci.bootlasso[, 1],      # Lower confidence interval
  ci.up = ci.bootlasso[, 2]        # Upper confidence interval
)

# Filter to show only non-zero estimates
df_filtered <- df[df$estimate != 0, ]

# Display the filtered dataframe
df_filtered
```
```

```
```

Ridge

Setting penalization parameter λ

To set λ by cross-validation we use the function `cv.glmnet` again but this time setting parameter `alpha` to 0.

```{r}
fit.ridge <- cv.glmnet(x = x_train_scaled, y = y_train, alpha = 0, nfolds = 10)
fit.ridge
```

We plot the estimated mean squared prediction error $\widehat{MSPE}$ as a function of λ.

```{r}
plot(fit.ridge)
```

We plot the cofficient path and see that, this time, all estimated coefficients shrink as λ grows but remain non-zero.

```{r}
plot(fit.ridge$glmnet.fit, xvar='lambda')
```

```{r}
b.ridge <- as.vector(coef(fit.ridge, s = 'lambda.min'))
round(b.ridge, 3)
```
Compare it to MLE coef
```{r}
b.mle <- as.vector(coef(fit.mle))
print(b.mle, round = 3)
```

Bayesian Gaussian regression

To fit a Gaussian linear regression you set `family` to `gaussian()` in `stan_glm`.

Ridge like:
```{r}
fit.l2.bayes <- stan_glm(covid_19_scaled$total_cases.y ~ ., data = covid_19_scaled, family = gaussian(), algorithm='sampling')
b.l2.bayes <- coef(fit.l2.bayes)
```
lasso penalty

We can change the prior
```{r}
fit.l1.bayes <- stan_glm(covid_19_scaled$total_cases.y ~ .,
                         data = covid_19_scaled,
                         family = gaussian(),
                         algorithm = 'sampling',
                         prior = lasso(1)) # Laplace prior with scale of 1
```

```
b.l1.bayes <- coef(fit.l1.bayes)

```

```{r}
print(b.l2.bayes)
```

```{r}
print(b.mle)
print(length(b.mle))
```

Plot the different coefficients:
```{r}
data.frame(mle= b.mle, bayes= b.l2.bayes) %>%
  ggplot(aes(x=mle,y=bayes)) +
  geom_point(shape = "O",size=2) +
  geom_abline(slope=1, intercept = 0, linetype = 'dashed') +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  xlab('MLE OLS') +
  ylab('MCMC Bayesian regression') +
  coord_cartesian(xlim=c(-2*10^6,2*10^6),ylim=c(-2*10^6,2*10^6)) +
  theme_classic()
```

Bayesian model selection

We also include some initial appraoches of how we would tackle our research question in a more Bayesian way.
As described in the report, lasso penalty worked the best so we decided to mainly describe that approach.

Fitting

We run the Bayesian model selection implemented in the function ‘modelSelection‘ from R package ‘mombf‘. The option ‘enumerate‘, if set to ‘TRUE‘, forces full model enumeration, but more generally when the number of parameters is large its better to use the default, which uses Gibbs sampling to search the model space.

We set a Beta-Binomial(1,1) prior on the models and Zellners prior on the regression coefficients. The prior dispersion parameter ‘taustd‘ corresponds to g in our notation, for which we use the default $g=1$.

```{r}
fit.bayesreg <- modelSelection(
    y=y_train,
    x=as.matrix(x_train_scaled),
    priorCoef=zellnerprior(taustd=1),
    priorDelta=modelbbprior(1,1)
)
```

Note that the Gaussian shrinkage prior is also available in ‘mombf‘ as ‘normalidprior()‘, with parameter ‘taustd‘ corresponding to g in our notation.

```{r}
head(postProb(fit.bayesreg), 10)
```

```
```
Model averaging estimates of coefficients

With the method 'coef' we extract Bayesian model averaging estimates for each coefficient,
    posterior intervals and posterior marginal inclusion probabilities ($P(\beta_j \neq 0 |
    \mathbf{y}) = P(\gamma_j = 1 | \mathbf{y})$).

```{r}
ci.bayesreg <- coef(fit.bayesreg)[-c(1,nrow(coef(fit.bayesreg))),]
sel.bayesreg <- ci.bayesreg[,4] > 0.5
ci.bayesreg[,1:3] <- round(ci.bayesreg[,1:3], 3)
ci.bayesreg[,4] <- round(ci.bayesreg[,4], 4)
head(ci.bayesreg)
tail(ci.bayesreg)
```


Part 2 of the course:

```{r}

# Creating a new variable as a copy of covid_19_scaled
covid_19_new <- covid_19_scaled

# Rename the column "total_cases.y" to "total_cases" in the new variable
names(covid_19_new)[names(covid_19_new) == "total_cases.y"] <- "total_cases"
covid_19 <- covid_19_new

```

-------Implement K-means--------
Kmeans
```{r}
# Check for NA values in each column
na_summary <- colSums(is.na(covid_19))

# Display columns with NA values
na_columns <- na_summary[na_summary > 0]
print(na_columns)
```
```{r}
# Check which columns are not numeric
non_numeric_cols <- covid_19 %>%
  select(where(~ !is.numeric(.)))
print(colnames(non_numeric_cols))
```

```{r}

# Preparing the Data
df <- covid_19 %>% select(-total_cases)

# Compute Gap Statistic using Default K-means
set.seed(123) # For reproducibility
gap_stat <- clusGap(df, FUN = kmeans, nstart = 25, K.max = 10, B = 25)
# K.max = maximum number of clusters to test
# B = number of bootstrap samples

# Display Optimal Number of Clusters
cat("Optimal number of clusters based on Gap Statistic:\n")
```

```
print(gap_stat, method = "firstmax")

# Visualize the Gap Statistic
fviz_gap_stat(gap_stat) +
  labs(title = "Gap Statistic for Optimal Number of Clusters")

```
As we can see the GAP Statistic is giving us an optimal of 3 clusters, so now we wanted to
    compare it with the Elbow method and see how our clusters look like.
```{r}
# Prepare Data for Clustering
# Exclude the target variable `total_cases`
df_kmeans <- covid_19 %>% select(-total_cases)

# Determine Optimal Number of Clusters
# Elbow Method
elbow_plot <- fviz_nbclust(df_kmeans, kmeans, method = "wss") +
  labs(title = "Elbow Method for Optimal Clusters")

# Gap Statistic
set.seed(123) # For reproducibility
gap_stat <- clusGap(df_kmeans, FUN = kmeans, nstart = 25, K.max = 10, B = 25)

# Visualize the Gap Statistic
gap_stat_plot <- fviz_gap_stat(gap_stat) +
  labs(title = "Gap Statistic for Optimal Clusters")

# Combine both plots side-by-side using patchwork
combined_plot <- elbow_plot / gap_stat_plot

# Display the combined plot
print(combined_plot)

# Step 3: Apply K-means
# Set number of clusters to k = 3 (determined from Gap Statistic)
set.seed(123) # For reproducibility
kmeans_result <- kmeans(df_kmeans, centers = 3, nstart = 25)

# Add cluster assignments to the original data
df_with_clusters <- covid_19 %>%
  mutate(cluster = kmeans_result$cluster)

# Step 4: Visualize Clusters
# Visualize in 2D using PCA
fviz_cluster(kmeans_result, data = df_kmeans,
             geom = "point", ellipse.type = "convex") +
  labs(title = "K-means Clustering for k = 3 Clusters")

# Step 5: Summarize Results
# View cluster means for interpretation
cluster_means <- df_with_clusters %>%
  group_by(cluster) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE))

print(cluster_means) # Inspect cluster averages

```
It is evident that the Elbow method would also suggest 3 clusters.
While checking our cluster outcome we saw that there was a potential outlier in our data,
    however when checking it closely we saw that that is an actual observation, we decided to
    keep it and see the outcome in the next results.

###Identifying Outliers

```{r}

# Step 1: Perform PCA on the scaled K-means input data
pca_result <- prcomp(df_kmeans, scale. = TRUE)

# Extract PCA scores for the first two components
pca_scores <- as.data.frame(pca_result$x)

# Step 2: Add cluster assignments and observation IDs to the PCA scores
pca_scores_with_clusters <- pca_scores %>%
  mutate(cluster = kmeans_result$cluster,
         observation_id = rownames(df_kmeans)) # Add observation IDs for tracking

# Step 3: Identify Potential Outliers Based on PCA Scores
# Adjust the threshold (e.g., PC1 > 15, PC2 < -30) based on your scatterplot
outliers <- pca_scores_with_clusters %>%
  filter(PC2 > 30 | PC2 < -30) # Modify thresholds based on observed graph

# Print the details of the outliers
print("Outliers identified in PCA:")
print(outliers)

# Step 4: Trace the Outlier(s) Back to the Original Dataset
outlier_details <- covid_19 %>%
  mutate(observation_id = rownames(covid_19)) %>%
  filter(observation_id %in% outliers$observation_id)

# Print detailed information about the outlier(s) in the original dataset
print("Details of the outlier(s) in the original dataset:")
print(outlier_details)

# Step 5: Map Outlier to Specific Country (if applicable)
# Ensure `covid_19_countries` has proper observation IDs
names(covid_19_countries) <- make.names(names(covid_19_countries), unique = TRUE)

covid_19_countries <- covid_19_countries %>%
  mutate(observation_id = rownames(covid_19_countries))

outlier_with_country <- covid_19_countries %>%
  filter(observation_id %in% outliers$observation_id)

# Display the country name and details for the outlier(s)
print("Outlier details with country name:")
print(outlier_with_country)

# Step 6: Visualize the Outliers on PCA Plot
ggplot(pca_scores_with_clusters, aes(x = PC1, y = PC2, color = as.factor(cluster))) +
  geom_point() +
  geom_point(data = outliers, aes(x = PC1, y = PC2), color = "black", size = 3, shape = 8) +
  labs(title = "PCA Plot Highlighting Outliers", x = "PC1", y = "PC2", color = "Cluster") +
  theme_minimal()

```

###Analyzing Clusterin K-Means
```{r}
cluster_sizes <- df_with_clusters %>%
  group_by(cluster) %>%
  summarise(Count = n())

print(cluster_sizes)
```

```
This is the size of our clusters.

```{r}
key_features_summary <- df_with_clusters %>%
  group_by(cluster) %>%
  summarise(
    avg_gdp_per_capita = mean(gdp_per_capita, na.rm = TRUE),
    avg_life_expectancy = mean(life_expectancy, na.rm = TRUE),
    avg_healthcare_infra = mean(hospital_beds_per_thousand, na.rm = TRUE)
  )

print(key_features_summary)

```
And here we decided to analyze some variables to see how the countries selected relate to each
    other. With this analysis we could say:
Cluster 1 includes countries with a moderate average GDP per capita (-0.07), slightly
    above-average life expectancy (0.18), and below-average healthcare infrastructure (-0.19),
    suggesting countries in a transitional stage of development. In contrast, Cluster 2
    represents countries with the highest average GDP per capita (0.94), life expectancy
    (1.01), and healthcare infrastructure (1.01), reflecting well-developed economies and
    robust healthcare systems. Finally, Cluster 3 contains countries with the lowest average
    GDP per capita (-0.75), life expectancy (-1.22), and healthcare infrastructure (-0.61),
    indicating significant economic and healthcare challenges.

Note: The reported values are standardized (z-scores) to ensure comparability across variables,
    as the data was centered and scaled before clustering. These values represent deviations
    from the mean, measured in standard deviations.

Applying PCA
Given the limitations of K-means clustering in high-dimensional data, particularly the potential
    for redundancy and noise among variables, we next explore Gaussian Mixture Models (GMM). To
    further improve the clustering results and address the curse of dimensionality, we
    integrate Principal Component Analysis (PCA) for dimensionality reduction before applying
    GMM. This approach enables us to retain the most significant patterns in the data while
    reducing its complexity, thereby enhancing the interpretability and robustness of the
    clustering process.

```{r}
# Exclude the target variable (total_cases)
df_pca <- covid_19 %>% select(-total_cases)

# Standardize the data (z-score normalization)
df_pca <- scale(df_pca)

```

```{r}
# Run PCA
pca_result <- prcomp(df_pca, center = TRUE, scale. = TRUE)

# Summary of PCA
summary(pca_result)

```

```{r}
# Plot the proportion of variance explained
fviz_eig(pca_result, addlabels = TRUE, ylim = c(0, 50)) +
  labs(title = "Scree Plot: Variance Explained by Principal Components")

```
```

```
PC1 explains 31.2% of the variance  this is a significant proportion.
The variance explained drops sharply after PC1, with PC2 (11%) and PC3 (10.4%).
Beyond PC4 (~9.5%), the contribution of additional PCs diminishes gradually.
The "elbow" (inflection point) in the scree plot occurs at PC4-5, where the curve starts
    leveling off.

This is why we will retain the first 5 components
These PCs explain a combined ~70% of the variance.
This choice balances dimensionality reduction while retaining significant information.

```{r}
# Extract the first 5 PCs
pca_scores <- as.data.frame(pca_result$x[, 1:5])

# Add the target variable back for analysis
pca_scores$total_cases <- covid_19$total_cases

# Check the transformed data
head(pca_scores)

```
The 5 PCs are linear combinations of the original features.
These PCs represent the most important directions of variation in the data, with PC1 explaining
    the largest variance (~31.2%) and subsequent PCs capturing smaller amounts of variance.
Each principal component (PC) provides insight into how different observations relate to one
    another in the reduced space.

We can also see how to interpret these PCs, at taking a close look at the loadings:
```{r}
# Extract the PCA loadings (coefficients for each component)
loadings <- pca_result$rotation

# View the loadings for the first few components
head(loadings)

```

This can be done for each Principal component, we include an example of the first one here.
The output you provides the loadings of various features or interactions between features on the
    first principal component (PC1) of the PCA analysis. These loadings represent the
    contribution of each feature or interaction term to the principal component.
```{r}
# Extract the loadings for the first principal component (PC1)
component <- 2 #change this to analyze for different components
loadings_PC1 <- loadings[, component]

# Sort the loadings by absolute value in decreasing order
loadings_PC1_sorted <- sort(abs(loadings_PC1), decreasing = TRUE)

# Convert the sorted loadings to percentages
loadings_PC1_percent <- loadings_PC1_sorted * 100

# Print the sorted loadings and their corresponding percentages
loadings_PC1_percent
```


Correlation Analysis
```{r}
# Calculate correlations between PCs and total_cases
correlations <- cor(pca_scores[, 1:5], pca_scores$total_cases)
print(correlations)
```
```

```
```

To explore the relationship between the principal components and total_cases, we conducted a
    correlation analysis. The results show that PC4 has the strongest negative correlation with
    total_cases at -0.662, indicating that lower PC4 values align with higher total case
    counts. In contrast, PC1 exhibits the strongest positive correlation at 0.296, suggesting
    that higher values of PC1 are moderately associated with an increase in case counts. PC3
    also displays a weaker positive correlation of 0.239, while PC2 and PC5 exhibit weak
    negative correlations of -0.236 and -0.048, respectively.

Plotting PC1 and PC4
```{r}
# Custom theme for cleaner visuals
custom_theme <- theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 16),
    axis.title = element_text(face = "bold", size = 14),
    axis.text = element_text(size = 12)
  )

# Scatterplot for PC1 vs Total Cases with log-transformed y-axis
plot_pc1 <- ggplot(pca_scores, aes(x = PC1, y = total_cases)) +
  geom_point(color = "lightblue", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  scale_y_log10() +
  custom_theme +
  labs(title = "PC1 vs Total Cases",
       x = "PC1",
       y = "Total Cases")

# Scatterplot for PC4 vs Total Cases with log-transformed y-axis
plot_pc4 <- ggplot(pca_scores, aes(x = PC4, y = total_cases)) +
  geom_point(color = "lightblue", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  scale_y_log10() +
  custom_theme +
  labs(title = "PC4 vs Total Cases",
       x = "PC4",
       y = "Total Cases")

# Arrange the two plots in a single grid
grid.arrange(plot_pc1, plot_pc4, ncol = 2)

```
We are specifically analyzing PC1 and PC4 because they show the strongest correlations with
    total cases: PC1 has a positive correlation reflecting key variables that contribute to
    case increases, while PC4 has a negative correlation, highlighting factors that may
    suppress or counteract higher case counts. This analysis validates the importance of PC1
    and PC4 in explaining variability and correlations, providing a foundation for our next
    step which is clustering using Gaussian Mixture Models (GMM).

Importance of PCA
Principal Component Analysis (PCA) reduces high-dimensional data into fewer components while
    retaining key patterns.

GMMS
Overall, PCA successfully reduced the dataset's complexity, identifying PCs that highlight
    latent relationships with the target variable. The correlations of PC1 and PC4 with
    demonstrate the method's ability to uncover underlying structure in the data while reducing
    redundancy. We next employ Gaussian Mixture Models (GMM). By applying GMM to the
    PCA-transformed dataset, we aim to achieve more flexible and interpretable clusters while

leveraging the reduced dimensional space to enhance computational efficiency and cluster quality.

```{r}
# Prepare PCA-transformed Data
# Retain the first 5 principal components
pca_data <- as.data.frame(pca_result$x[, 1:5])
colnames(pca_data) <- c("PC1", "PC2", "PC3", "PC4", "PC5")

# Apply Gaussian Mixture Models
set.seed(123) # For reproducibility
gmm_result <- Mclust(pca_data)

# Summary of GMM Results
summary(gmm_result)

# Visualize GMM BIC Values
plot(gmm_result, what = "BIC") # Plot BIC values to validate the optimal number of clusters

# Add GMM Cluster Assignments to PCA Data
pca_data <- pca_data %>%
  mutate(gmm_cluster = gmm_result$classification) # Explicitly name the new cluster column

# Visualize Clusters in PCA Space
# Scatter plot for the first two principal components
ggplot(pca_data, aes(x = PC1, y = PC2, color = factor(gmm_cluster))) +
  geom_point(size = 2, alpha = 0.6) +
  labs(title = "GMM Clustering in PCA-Reduced Space",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "Cluster") +
  theme_minimal()

# Analyze Cluster Means
# Summarize the cluster means for the principal components
cluster_summary <- pca_data %>%
  group_by(gmm_cluster) %>%
  summarise(across(starts_with("PC"), mean, na.rm = TRUE))

# Display the cluster summary
print(cluster_summary)
```

```
The Bayesian Information Criterion (BIC) identified the optimal clustering solution as the VEV model, characterized by ellipsoidal clusters with equal shape. This model achieves a BIC value of -4224.35, balancing model complexity and fit. The resulting Gaussian Mixture Model (GMM) solution produced three clusters of sizes 88, 12, and 75, as summarized in. These clusters were further visualized in the PCA-reduced space, where clear separations across the first two principal components were observed, illustrating the structure captured by the GMM.
```

Country Assignment

```{r}
# Prepare PCA-transformed Data
pca_data <- as.data.frame(pca_result$x[, 1:5])
colnames(pca_data) <- c("PC1", "PC2", "PC3", "PC4", "PC5")

# Apply Gaussian Mixture Models
set.seed(123) # For reproducibility
gmm_result <- Mclust(pca_data)

# Add GMM Cluster Assignments to PCA Data
```

```
pca_data$gmm_cluster <- gmm_result$classification

# Re-add the 'location' column
# Merge the cluster assignments with the original saved data
covid_19_countries <- covid_19_countries %>%
  mutate(gmm_cluster = pca_data$gmm_cluster) # Add cluster info

# Display the results
# View the country names and their corresponding clusters
country_clusters <- covid_19_countries %>%
  select(location, gmm_cluster)

print(country_clusters) # Display the table with country names and clusters

```
Here we can see the cluster assignment but to see it clearler we decided to plot a map.
```{r}
# Step 1: Merge total_cases into covid_19_countries
covid_19_countries <- covid_19_countries %>%
  mutate(total_cases = covid_19$total_cases)

# Step 2: Summarize by GMM clusters
covid_19_countries %>%
  group_by(gmm_cluster) %>%
  summarise(
    mean_cases = mean(total_cases, na.rm = TRUE)
  )


```
Plotting World Map

This code is commented out as it is just plotting the results seen in the report and requires
    some heavy dependenes.
It can always be commented out and should run without any problems then.
```{r}
#install.packages("ggplot2")
#install.packages("rnaturalearth")
#install.packages("rnaturalearthdata")
#install.packages("ggthemes")

```

```{r}
library(ggplot2)
library(dplyr)
library(rnaturalearth) # To get country map data
library(rnaturalearthdata)
library(ggthemes)      # For clean map themes

# Step 1: Prepare the GMM cluster data with country names
# Assuming 'covid_19_countries' contains the "location" column and GMM clusters
cluster_map_data <- covid_19_countries %>%
  select(location, gmm_cluster) %>%
  distinct(location, gmm_cluster) # Keep only unique country-cluster pairs

# Step 2: Fix the "United States" naming issue
cluster_map_data <- cluster_map_data %>%
  mutate(location = recode(location, "United States" = "United States of America"))

# Step 3: Load world map data
world <- ne_countries(scale = "medium", returnclass = "sf") # Natural Earth map
```

```r
# Step 4: Merge the cluster data with the world map
world_clusters <- world %>%
  left_join(cluster_map_data, by = c("name" = "location")) # Match by country names

# Step 5: Plot the world map with clusters
ggplot(data = world_clusters) +
  geom_sf(aes(fill = factor(gmm_cluster)), color = "white", size = 0.2) +
  scale_fill_manual(
    values = c("#1b9e77", "#d95f02", "#7570b3"), # Customize cluster colors
    name = "Cluster",
    labels = c("1", "2", "3") # Rename clusters if needed
  ) +
  labs(
    title = "World Map of GMM Clusters",
    subtitle = "Countries grouped based on demographic, economic, and healthcare variables",
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12),
    axis.text = element_blank(),
    axis.ticks = element_blank()
  )
```

```
The clustering results reveal fascinating insights, particularly the placement of the United
    States, China, and India in Cluster 2. These three countries recorded the highest total
    COVID-19 case counts, indicating that the GMM approach, combined with PCA, successfully
    identified critical underlying patterns in the data. This demonstrates the model's ability
    to capture essential relationships among demographic, economic, and healthcare variables,
    which led to the grouping of countries with vastly different socioeconomic contexts but
    similar pandemic outcomes. This outcome underscores the effectiveness of GMM in uncovering
    latent structures that align with real-world phenomena, even when total cases were not
    directly included in the clustering process.
```