Predicting life expectancy using PCA and LASSO regression

Brooke Stevens

2023-05-01

Table of Contents:

```r
# Load packages
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr)
library(ggplot2)
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rms)
```

```
## Loading required package: Hmisc
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following object is masked from 'package:psych':
##
##     describe
##
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
##
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
##
## Loading required package: SparseM
##
## Attaching package: 'SparseM'
##
## The following object is masked from 'package:base':
##
##     backsolve
```

```
library(bestNormalize)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-6
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Data

```
life_exp <- read.csv("../data/life_expectancy.csv")
glimpse(life_exp)
```

```
## Rows: 2,938
## Columns: 22
## $ Country                        <chr> "Afghanistan", "Afghanistan", "Afghani~
## $ Year                           <int> 2015, 2014, 2013, 2012, 2011, 2010, 20~
## $ Status                         <chr> "Developing", "Developing", "Developin~
## $ Life.expectancy                <dbl> 65.0, 59.9, 59.9, 59.5, 59.2, 58.8, 58~
## $ Adult.Mortality                <int> 263, 271, 268, 272, 275, 279, 281, 287~
## $ infant.deaths                  <int> 62, 64, 66, 69, 71, 74, 77, 80, 82, 84~
## $ Alcohol                        <dbl> 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.~
## $ percentage.expenditure         <dbl> 71.279624, 73.523582, 73.219243, 78.18~
## $ Hepatitis.B                    <int> 65, 62, 64, 67, 68, 66, 63, 64, 63, 64~
## $ Measles                        <int> 1154, 492, 430, 2787, 3013, 1989, 2861~
## $ BMI                            <dbl> 19.1, 18.6, 18.1, 17.6, 17.2, 16.7, 16~
## $ under.five.deaths              <int> 83, 86, 89, 93, 97, 102, 106, 110, 113~
## $ Polio                          <int> 6, 58, 62, 67, 68, 66, 63, 64, 63, 58,~
## $ Total.expenditure              <dbl> 8.16, 8.18, 8.13, 8.52, 7.87, 9.20, 9.~
```

```
## $ Diphtheria                      <int> 65, 62, 64, 67, 68, 66, 63, 64, 63, 58~
## $ HIV.AIDS                        <dbl> 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1~
## $ GDP                             <dbl> 584.25921, 612.69651, 631.74498, 669.9~
## $ Population                      <dbl> 33736494, 327582, 31731688, 3696958, 2~
## $ thinness..1.19.years            <dbl> 17.2, 17.5, 17.7, 17.9, 18.2, 18.4, 18~
## $ thinness.5.9.years              <dbl> 17.3, 17.5, 17.7, 18.0, 18.2, 18.4, 18~
## $ Income.composition.of.resources <dbl> 0.479, 0.476, 0.470, 0.463, 0.454, 0.4~
## $ Schooling                       <dbl> 10.1, 10.0, 9.9, 9.8, 9.5, 9.2, 8.9, 8~
```

The `life_expectancy.csv` dataset was obtained through Kaggle. This data was collected from the World Health Organization and United Nations website. Each row contains various immunization, mortality, economic, and social metrics for a country in a given year. The years 2000 through 2015 are represented in this dataset.

According to the publisher, this dataset aims to answer the following key questions: **(source: https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who)**

1. Do various predicting factors which have been chosen initially really affect life expectancy? What are the predicting variables actually affecting life expectancy?
2. Should a country having a lower life expectancy value (<65) increase its healthcare expenditure in order to improve its average lifespan?
3. How do infant and adult mortality rates affect life expectancy?
4. Does life expectancy have a positive or negative correlation with eating habits, lifestyle, exercise, smoking, drinking alcohol, etc.
5. What is the impact of schooling on life expectancy?
6. Does life expectancy have a positive or negative relationship with drinking alcohol?
7. Do densely populated countries tend to have lower life expectancy?
8. What is the impact of immunization coverage on life expectancy?

I am gong to use this dataset to **determine if life expectancy can be predicted using a LASSO linear regression model.**

Variables of Interest

Dependent Variable:

- `Life.expectancy`: Average life expectancy in years

Independent Variables:

- `Alcohol`: Alcohol, recorded per capita (15+) consumption (in liters of pure alcohol)
- `Hepatitis.B`: Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
- `Measles`: Number of reported measles cases per 1000 population
- `under.five.deaths`: Number of under-five deaths per 1000 population
- `Polio`: Polio (Pol3) immunization coverage among 1-year-olds (%)
- `Total.expenditure`: General government expenditure on health as a percentage of total government expenditure (%)
- `Diphtheria`: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
- `thinness.5.9.years`: Prevalence of thinness among children for age 5 to 9 (%)

- `HIV.AIDS`: Deaths per 1000 live births due to HIV/AIDS (0-4 years)
- `Adult.Mortality`: Adult mortality rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
- `GDP`: Gross domestic product per capita (in USD)
- `BMI`: Average body mass index of population
- `Schooling`: Number of years of schooling
- `Income.composition.of.resources`: Human development index in terms of income composition of resources (ranging from 0 to 1)
- `infant.deaths`: Number of infant deaths per 1000 population **(removed from PCA)**
- `percentage.expenditure`: Expenditure on health as a percentage of GDP per capita (%) **(removed from PCA)**
- `thinness..1.19.years`: Prevalence of thinness among children and adolescents for age 10 to 19 (%) **(removed from PCA)**

Research Question

Using this data, I want know if **life expectancy** can be predicted through a LASSO linear regression using the above independent variables after a PCA dimension reduction.

Null Hypothesis: Life expectancy is **not** related to the n number of principal components after the dimension reduction

Alternative Hypothesis: Life expectancy **is** related to the n number of principal components after the dimension reduction

Formal Model:

$$Life\,Expectancy = \beta_0 + \beta_1 * TC1 + \beta_2 * TC2 + \ ... \ + \beta_n * TCn + \varepsilon$$

Formal Hypotheses:

$$H_0\,(null) : \beta_1 = \beta_2 = \ ... \ = \beta_n = 0$$

$$H_A\,(alternative) : Any\,\beta_i \neq 0$$

Data Wrangling

Before I begin my analysis, I want to remove all variables from `life_exp` that will not be part of the analysis. This new dataset will be stored in the `life_exp_modified` variable. I would also like to remove all instances of `NA` from the dataset.

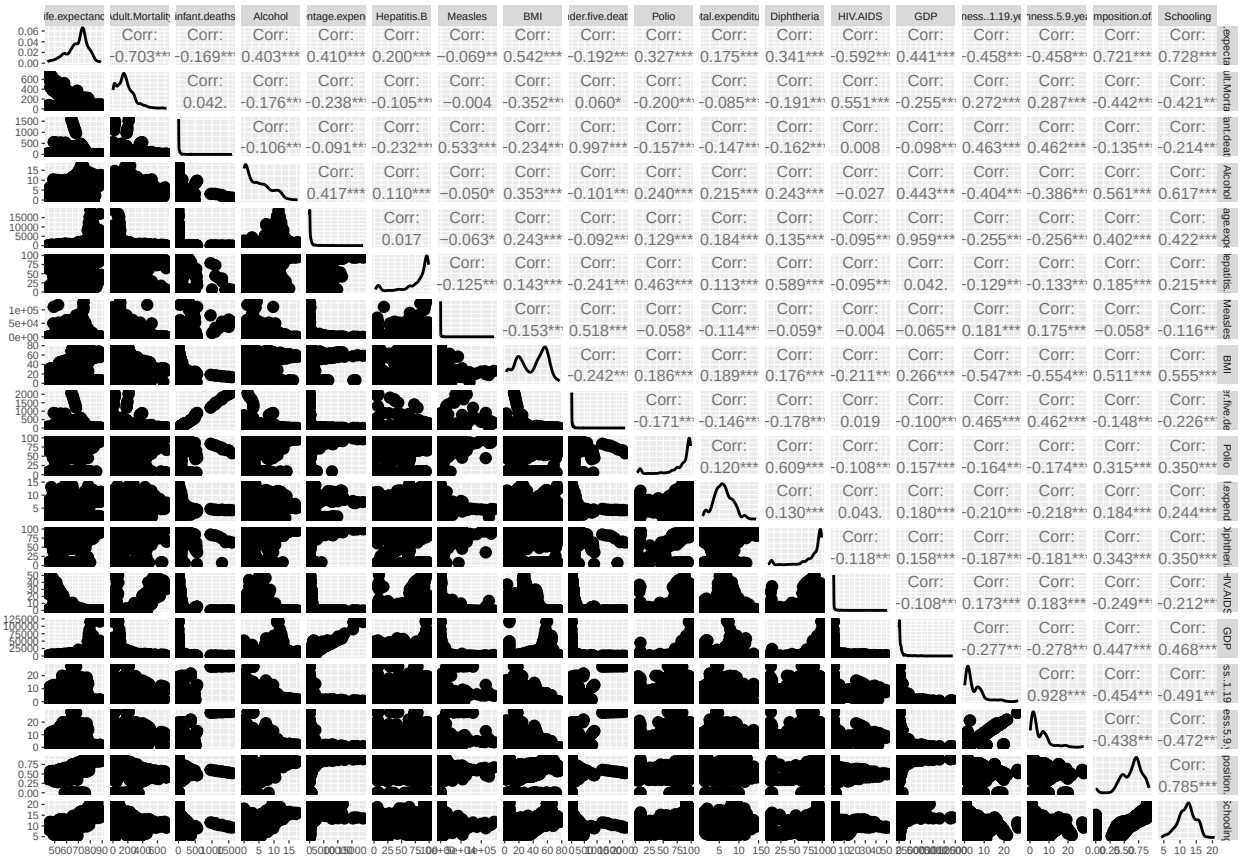
```
# Drop all instances of NA
life_exp <- life_exp %>%
  drop_na()

# Store life expectancy value for later
value <- life_exp$Life.expectancy

# Remove unused variables
life_exp_modified <- life_exp %>%
  select(-Country, -Year, -Status, -Population)
```

Visualizing Variables of Interest

I will use `{ggpairs}` to examine the scatterplots, distributions, and correlations between the variables within the dataset.

Because there are so many variables, I had to shrink the size of the font in order for the variable names and correlation values to be legible.

```
# Using ggpairs
ggpairs(life_exp_modified, upper = list(continuous = wrap("cor", size = 2))) +
  theme_grey(base_size = 5)
```



Most of these variables do not appear to be normally distributed. Some like `Adult.Mortality`, `infant.deaths`, and `percentage.expenditure` are right-skewed, while others like `Diphtheria`, `Polio`, and `Hepatitis.B` are left-skewed. The variables that appear the most normally distributed are `Total.expenditure` and `Schooling`. Nevertheless, it would still be beneficial to normalize all our variables of interest (see **Dimension Reduction**).

The highest correlation values are shared by `infant.deaths` and `under.five.deaths` (0.997), `percentage.expenditure` and `GDP` (0.995), and `thinness..1.19.years` and `thinness.5.9.years` (0.928). Because these correlations are greater than 0.90, they will likely be removed from the model in the dimension reduction process (see **Dimension Reduction**).

Dimension Reduction

Before creating a linear regression model, I want to know whether the number of components can be reduced through dimension reduction.

First, I will remove the dependent variable of the analysis, `Life.expectancy`, from `life_exp_modified`.

```r
life_exp_modified <- life_exp_modified %>%
  select(-Life.expectancy)
```

The steps I will take to perform principal component analysis (PCA) are as follows:

1. Check for multicollinearity (r > 0.90)
2. Scale variables
3. Build the PCA, visualize the data
4. Conduct Bartlett's test (use sample size)
5. Kaiser-Meyer-Olkin (KMO) on the data (look for variables below 0.50 and remove)
6. Check scree plot for eigenvalues greater than 1
7. Check that residuals are normally distributed using the Shapiro-Wilk test
8. Extracting PCA scores for the model

First, we may proceed with step 1 and **check the variables of interest for multicollinearity**.

```r
# Step 1: Check for multicollinearity

# Obtain correlations
correlations <- cor(life_exp_modified)

# Determine which variables are greater than 0.90
greater_than <- which(abs(correlations) >= 0.90, arr.ind = TRUE)

# Duplicate relationships happen because of symmetric matrix
# Also removes diagonal which equals 1
greater_than <- greater_than[
  greater_than[,"row"] < greater_than[,"col"],
]

# Replace indices with actual names
greater_than[,"row"] <- colnames(life_exp_modified)[
  greater_than[,"row"]
]
greater_than[,"col"] <- colnames(life_exp_modified)[
  as.numeric(greater_than[,"col"])
]

# Remove names for ease of interpretation
unname(greater_than)
```

```
##      [,1]                     [,2]
## [1,] "infant.deaths"          "under.five.deaths"
## [2,] "percentage.expenditure" "GDP"
## [3,] "thinness..1.19.years"   "thinness.5.9.years"
```

The variables `infant.deaths` and `under.five.deaths`, `percentage.expenditure` and `GDP`, and `thinness..1.19.years` and `thinness.5.9.years` have correlations greater than 0.9. I will therefore choose to remove the `infant.deaths`, `percentage.expenditure`, and `thinness..1.19.years` variables from the analysis.

```r
life_exp_modified <- life_exp_modified %>%
  select(-infant.deaths, -percentage.expenditure, -thinness..1.19.years)
```

We may proceed with step 2 and **scale the variables**.

```
# Step 2: Scale variables

## Apply to data
set.seed(1234)
life_exp_modified <- apply(
  X = life_exp_modified,
  2, # 1 = rows, 2 = columns
  FUN = function(x){
    bestNormalize(x)$x.t
  }
)
```

Now that the dataset is normalized, we can proceed with step 3, which involves **building and visualizing the PCA**.

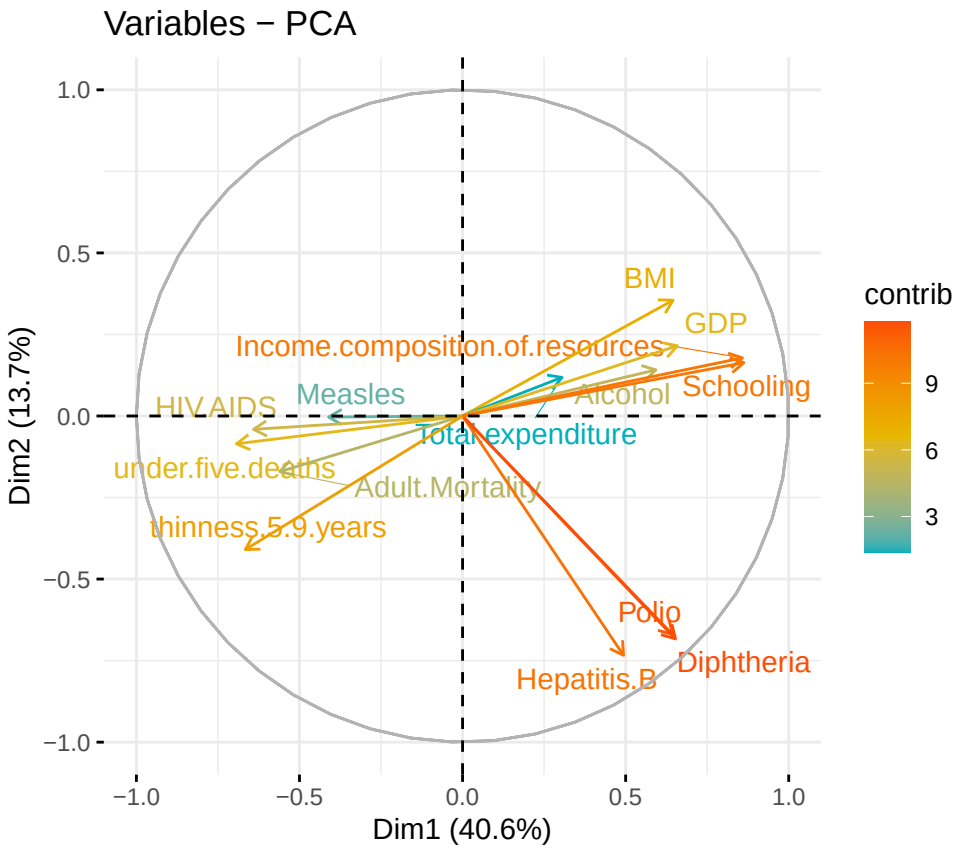
```
# Step 3: Visualize the data
## Building the PCA

# Perform PCA
life_exp_pca <- prcomp(life_exp_modified, center = TRUE, scale. = TRUE)

# Obtain summary
summary(life_exp_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.3828 1.3874 1.12299 1.06199 0.92468 0.80350 0.74389
## Proportion of Variance 0.4056 0.1375 0.09008 0.08056 0.06107 0.04612 0.03953
## Cumulative Proportion  0.4056 0.5430 0.63311 0.71367 0.77475 0.82086 0.86039
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.64841 0.62538 0.56970 0.55148 0.53768 0.34618 0.32461
## Proportion of Variance 0.03003 0.02794 0.02318 0.02172 0.02065 0.00856 0.00753
## Cumulative Proportion  0.89042 0.91836 0.94154 0.96326 0.98391 0.99247 1.00000
```

According to the table above, this PCA model can explain over 50% of the variance in the data with 2 components. This is very good!

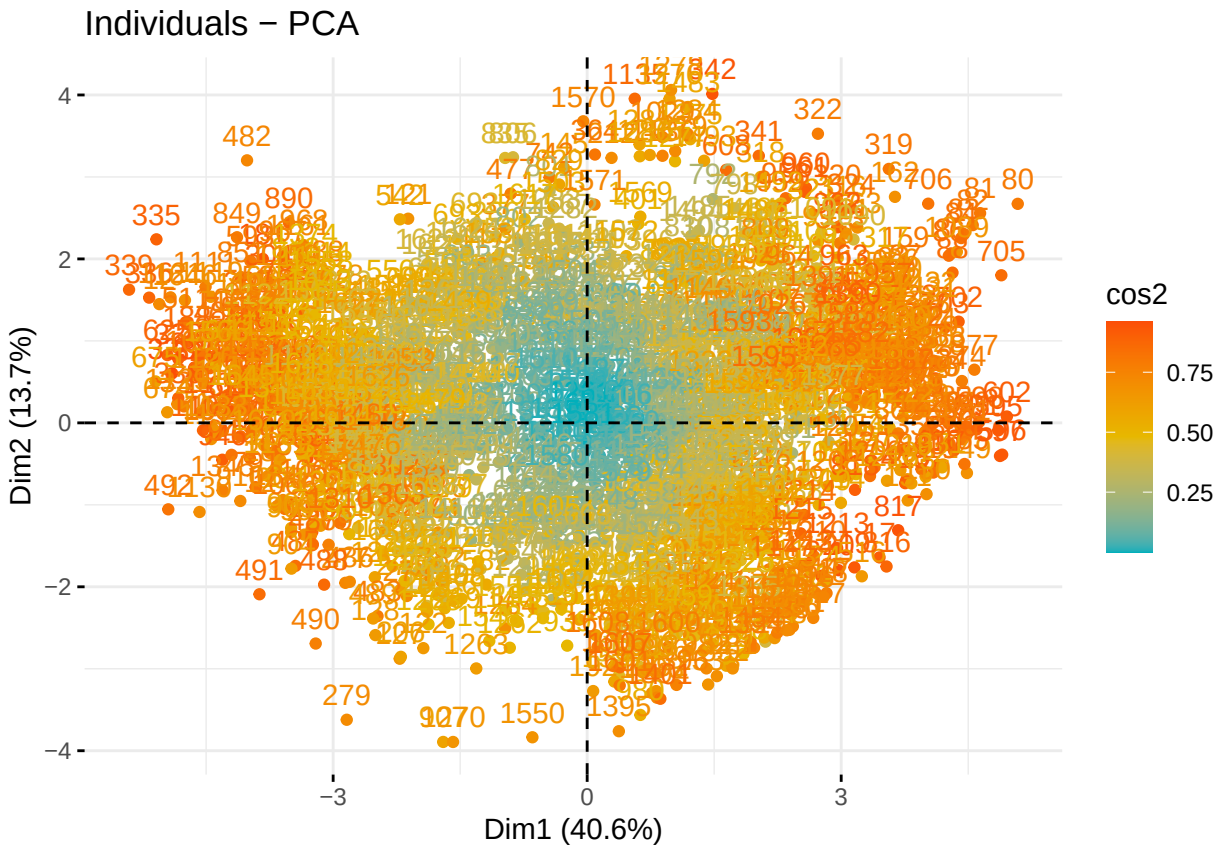
The following plot depicts all variables of the original PCA in a two-dimensional visualization:

```
# Step 3: Visualize the data

# Produce 2-dimensional plot
fviz_pca_var(
  life_exp_pca,
  col.var = "contrib", # Color by contributions to the PCA
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE #Avoid overlapping text if possible
)
```



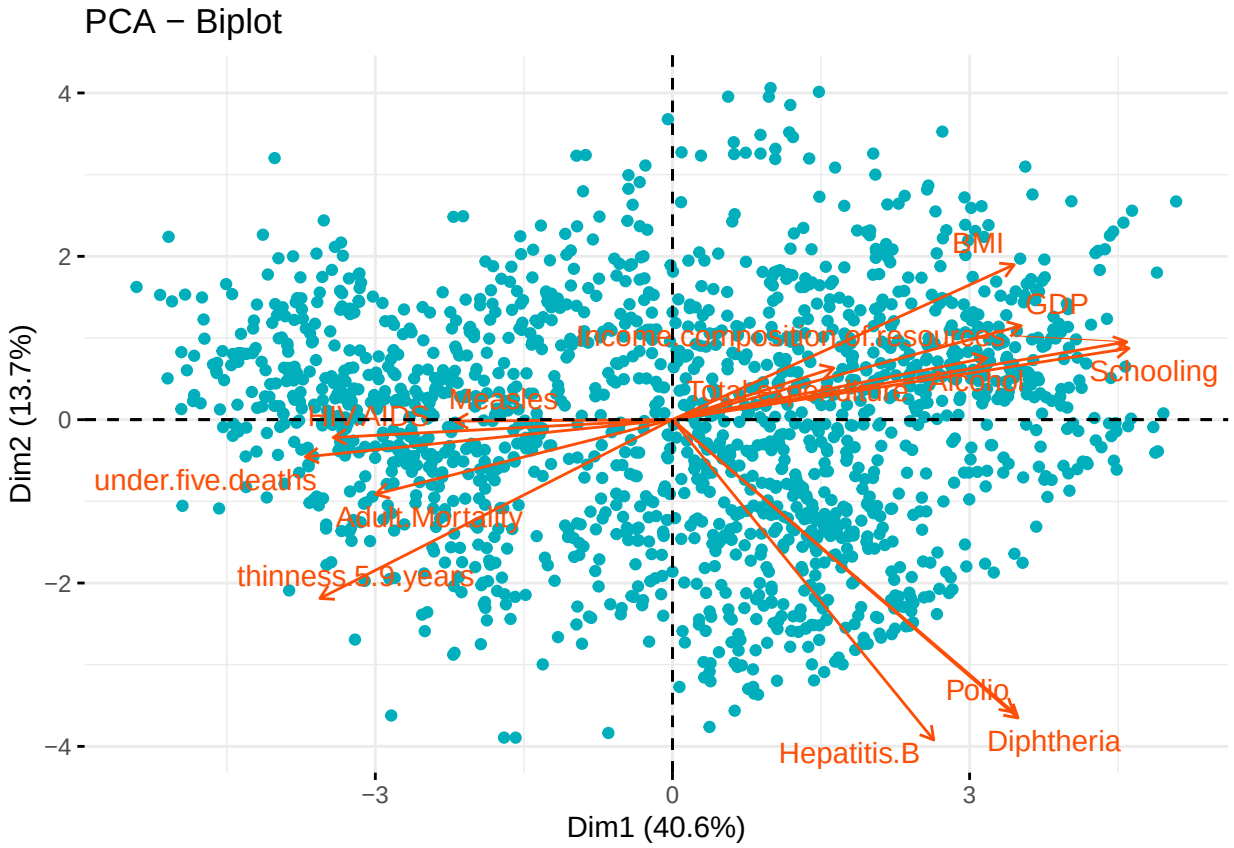
The following plot depicts all observations of the original PCA in a two-dimensional visualization:

```
# Step 3: Visualize the data

# Produce 2-dimensional plot
fviz_pca_ind(
  life_exp_pca,
  c = "point", # Observations
  col.ind = "cos2", # Quality of representation
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = FALSE
)
```



The following plot depicts a biplot of the observations and variables together:

```
# Step 3: Visualize the data

# Biplot
fviz_pca_biplot(
  life_exp_pca,
  repel = TRUE,
  col.var = "#FC4E07", # Variables color
  col.ind = "#00AFBB", # Individuals color
  label = "var" # Variables only
)
```



Now, we can proceed with step 4 and perform **Bartlett's test**. This value will help us determine whether PCA is appropriate for the data.

```
# Step 4: Perform Bartlett's Test

# Barlett's test
cortest.bartlett(life_exp_modified)
```

```
## R was not square, finding R from data
```

```
## $chisq
## [1] 13428.92
##
## $p.value
## [1] 0
##
## $df
## [1] 91
```

The Bartlett's test yields a p-value of 0. Because this is less than 0.05, we can say that PCA is appropriate for the data.

We can now proceed with step 5 and apply the **Kaiser-Meyer-Olkin (KMO)** process to the data. We should be on the lookout for KMO values less than 0.5, which would tell us that a variable is not appropriate for the analysis.

```
# Step 5: Kaiser-Meyer-Olkin (KMO) on the data

# KMO
KMO(life_exp_modified)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = life_exp_modified)
## Overall MSA =  0.86
## MSA for each item =
##                  Adult.Mortality                          Alcohol
##                             0.90                             0.91
##                      Hepatitis.B                          Measles
##                             0.87                             0.71
##                              BMI                 under.five.deaths
##                             0.92                             0.85
##                            Polio                Total.expenditure
##                             0.81                             0.89
##                       Diphtheria                         HIV.AIDS
##                             0.77                             0.87
##                              GDP                thinness.5.9.years
##                             0.96                             0.90
## Income.composition.of.resources                         Schooling
##                             0.86                             0.88
```

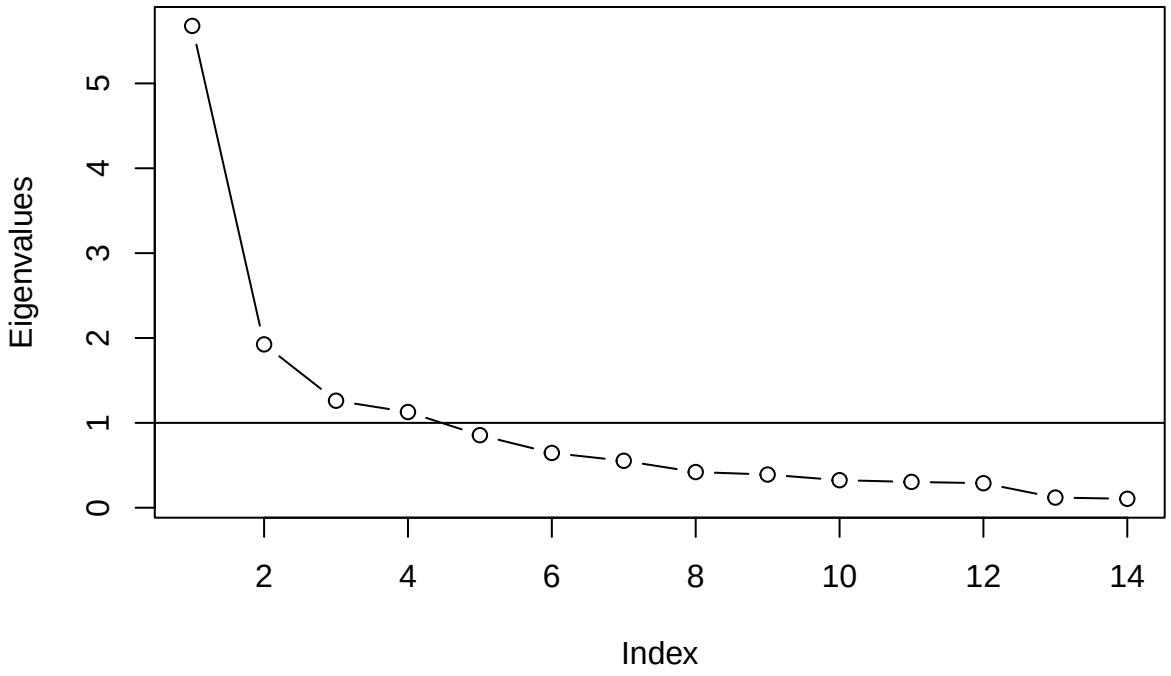
None of these KMO values are less than 0.5, so we are good to go!

Next, we will proceed with step 6 and determine the number of components to be used. We can do this by **checking the scree plot and parallel analysis plot** for eigenvalues greater than 1.

```
# Step 6: Determine the number of components

# PCA with {psych}
initial_pca <- principal(life_exp_modified, nfactors = ncol(life_exp_modified), rotate = "oblimin")
```

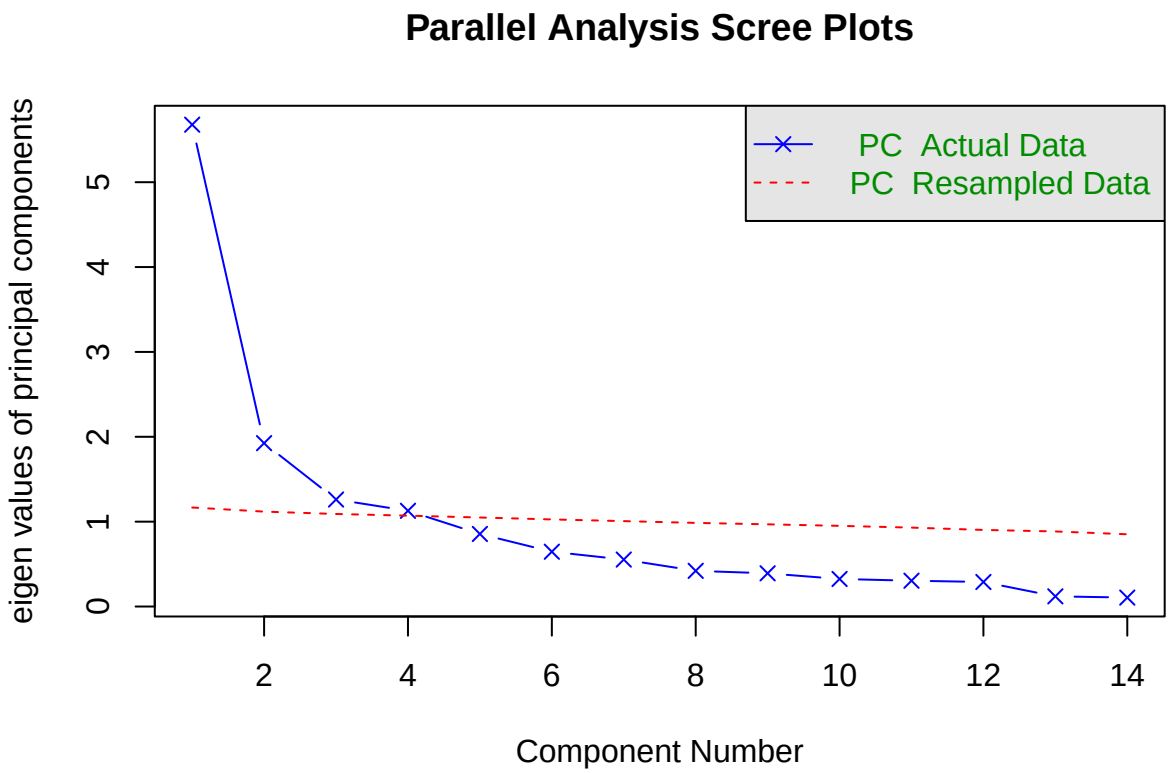
```
## Loading required namespace: GPArotation
```

```
# Plot
plot(initial_pca$values, type = "b", ylab = "Eigenvalues"); abline(h = 1)
```



According to the elbow method, the above scree plot suggests that **3 components** should be used.

```
# Step 6: Determine the number of components

# PCA with {psych}
parallel_pca <- fa.parallel(
  x = life_exp_modified,
  fa = "pc",
  sim = FALSE # ensures resampling
)
```




```
## Parallel analysis suggests that the number of factors =  NA  and the number of components =  4
```

The above parallel analysis plot suggests that **4 components** should be used.

We will instruct our final PCA to use **4 components** as suggested by the parallel analysis.

```
# Step 6: Determine the number of components

# PCA with {psych}
final_pca <- principal(
  r = life_exp_modified,
  nfactors = 4,
  rotate = "oblimin", # Correlated dimensions
  residuals = TRUE # Obtain residuals
)
```

We may now proceed with step 7 and **check the normality of the residuals** using the Shapiro-Wilk test.

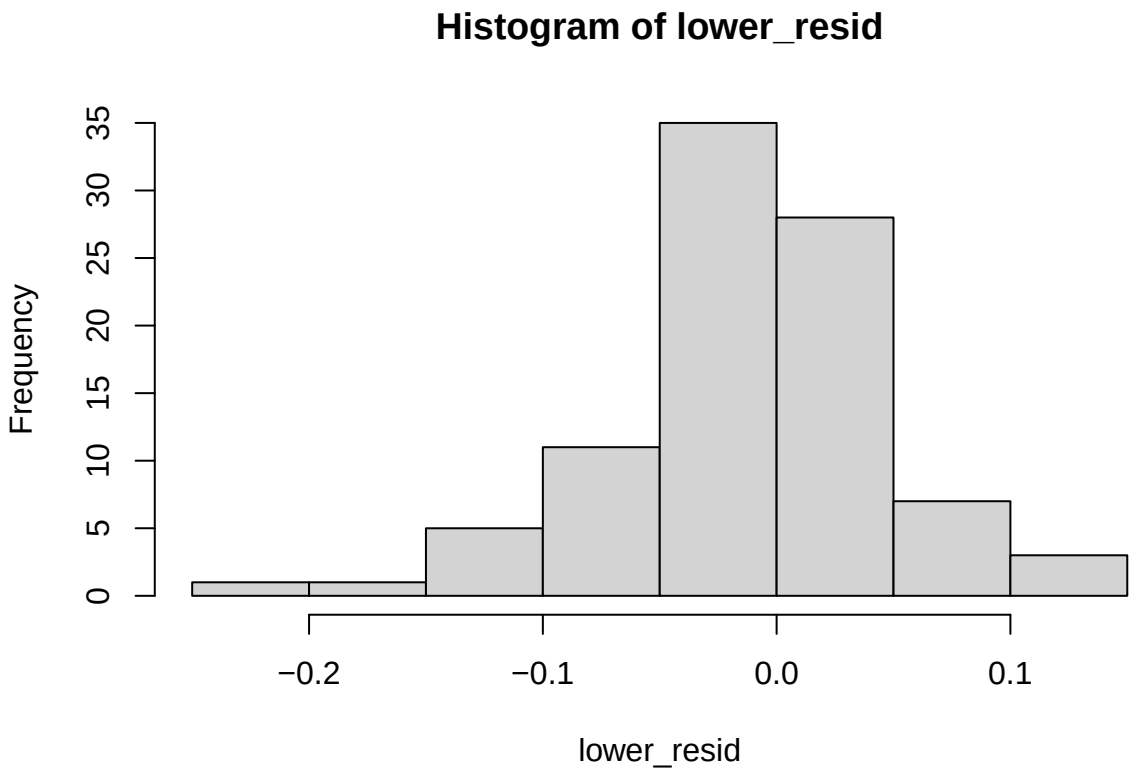
To check normality using the Shapiro-Wilk test:

- **Null Hypothesis:** Residuals **are** normally distributed ($p > 0.05$)
- **Alternative Hypothesis:** Residuals are **not** normally distributed ($p < 0.05$)

```r
# Step 7: Check residuals

resid <- final_pca$residual
lower_resid <- resid[lower.tri(resid)]

# Shapiro-Wilk
shapiro.test(lower_resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lower_resid
## W = 0.97563, p-value = 0.08539
```

```r
# Histogram
hist(lower_resid)
```



This p-value is greater than 0.05, so we can say these residuals are normally distributed.

Finally, we can perform step 8 and **compute scores for each PCA component** to determine the qualities/variables that make up each.

```r
# Step 8: Compute scores

# Check loadings
loadings <- round(final_pca$loadings[,1:4], 3)
loadings[abs(loadings) < 0.40] <- ""

# Sort TC1
loadings_ordered <- loadings[order(as.data.frame(loadings)$TC1),]
```

```
# View loadings
#View(loadings_ordered)
```

The `View(loadings_ordered)` command shows that **TC1** encapsulates:

- `thinness.5.9.years`
- `HIV.AIDS`
- `Adult.Mortality`
- `GDP`
- `BMI`
- `Schooling`
- `Income.composition.of.resources`

TC2 encapsulates:

- `Hepatitis.B`
- `Polio`
- `Diphtheria`

TC3 encapsulates:

- `Measles`
- `under.five.deaths`
- `thinness.5.9.years`

TC4 encapsulates:

- `Total.expenditure`
- `Alcohol`

With the help of ChatGPT, I've classified these four components as **Socioeconomic and Health Status (TC1)**, **Immunization Status (TC2)**, **Childhood Health and Mortality (TC3)**, and **Consumption and Substance Use (TC4)**.

ChatGPT provided the following reasons behind the selection of each name:

- **TC1:** "Based on these variable names, a possible name for the principal component could be"Socioeconomic and Health Status" or "Socioeconomic and Health Composite". This name reflects the fact that the variables included in the component relate to both socioeconomic and health factors, such as income, education, mortality rates, HIV/AIDS prevalence, and nutritional status."
- **TC2:** "Based on these variable names, a possible name for the principal component could be"Vaccine Preventable Diseases" or "VPD Composite". This name reflects the fact that the variables included in the component relate to diseases that are preventable by vaccination, such as hepatitis B, polio, and diphtheria. Alternatively, the component could be named "Immunization Status" or "Vaccine Coverage", to emphasize the importance of vaccination as a public health intervention."
- **TC3:** "Based on these variable names, a possible name for the principal component could be"Childhood Health and Mortality" or "Childhood Health Composite". This name reflects the fact that the variables included in the component relate to health and mortality outcomes in children, such as measles incidence and under-five mortality rates. Alternatively, the component could be named "Pediatric Health Status" or "Childhood Mortality and Morbidity", to emphasize the importance of addressing these issues in the context of improving overall pediatric health."

- **TC4:** "Based on these variable names, a possible name for the principal component could be"Spending and Substance Use" or "Expenditure and Alcohol Composite". This name reflects the fact that the variables included in the component relate to both spending patterns and alcohol consumption. Alternatively, the component could be named "Consumption and Substance Use" or "Alcohol and Expenditure Status", to emphasize the importance of understanding the relationship between these two factors in the context of public health and policy."

We can now extract scores from the PCA to be used in our linear model. These scores will be stored in the `scores_final` variable.

```
# Obtain scores
pca_scores <- final_pca$scores

# Change names
colnames(pca_scores) <- c(
  "socioeconomic_and_health_status",
  "immunization_status",
  "childhood_health_and_mortality",
  "consumption_and_substance_use"
)

## Make PCA scores a data frame
pca_scores <- as.data.frame(pca_scores)

## Add life expectancy value to PCA scores
pca_scores$life_expectancy <- value

## Remove NA cases
scores_final <- na.omit(pca_scores)
```

Linear Regression Model

Before running the regression, I will first **check for multicollinearity** between the principal components and life expectency (the dependent variable). This is one of the assumptions required for the construction of the model.

```
round(cor(pca_scores), 2)
```

```
##                                socioeconomic_and_health_status
## socioeconomic_and_health_status                            1.00
## immunization_status                                        0.29
## childhood_health_and_mortality                            -0.30
## consumption_and_substance_use                             -0.35
## life_expectancy                                            0.66
##                                immunization_status
## socioeconomic_and_health_status                0.29
## immunization_status                            1.00
## childhood_health_and_mortality                -0.22
## consumption_and_substance_use                 -0.22
## life_expectancy                                0.40
##                                childhood_health_and_mortality
## socioeconomic_and_health_status                          -0.30
## immunization_status                                      -0.22
## childhood_health_and_mortality                            1.00
## consumption_and_substance_use                             0.22
## life_expectancy                                          -0.39
```

```
##                                 consumption_and_substance_use life_expectancy
## socioeconomic_and_health_status                         -0.35            0.66
## immunization_status                                     -0.22            0.40
## childhood_health_and_mortality                           0.22           -0.39
## consumption_and_substance_use                            1.00           -0.77
## life_expectancy                                         -0.77            1.00
```

The highest correlation value is shared by `life_expectancy` and `consumption_and_substance_use`. While 0.77 is high, I don't believe it is a significant cause for concern.

We can now run a LASSO regression using a training and testing set.

First, I will separate the PCA scores data frame into matrices containing the predictor and outcome variables.

```
# Define outcome variable
outcome <- pca_scores$life_expectancy

# Define predictors
predictors <- pca_scores[, -which(colnames(pca_scores) %in% c("life_expectancy"))]

# Make matrix (required by {glmnet})
predictors <- as.matrix(predictors)
outcome <- as.matrix(outcome)
```

I will now scale the predictor variables using `{bestNormalize}`.

```
## Apply to data
set.seed(1234)
predictors <- apply(
  X = predictors,
  2, # 1 = rows, 2 = columns
  FUN = function(x){
    bestNormalize(x)$x.t
  }
)
```

I will now create an **80/20 train/test split** for the predictors and the outcome.

```
set.seed(1234)

predictor_train_idx = sort(sample(nrow(predictors), nrow(predictors)*.8))
predictor_train <- predictors[predictor_train_idx,]
predictor_test <- predictors[-predictor_train_idx,]

outcome_train_idx = sort(sample(nrow(outcome), nrow(outcome)*.8))
outcome_train <- outcome[outcome_train_idx,]
outcome_test <- outcome[-outcome_train_idx,]
```

I will now perform cross-validation to **determine the value of lambda that minimizes MSE**.

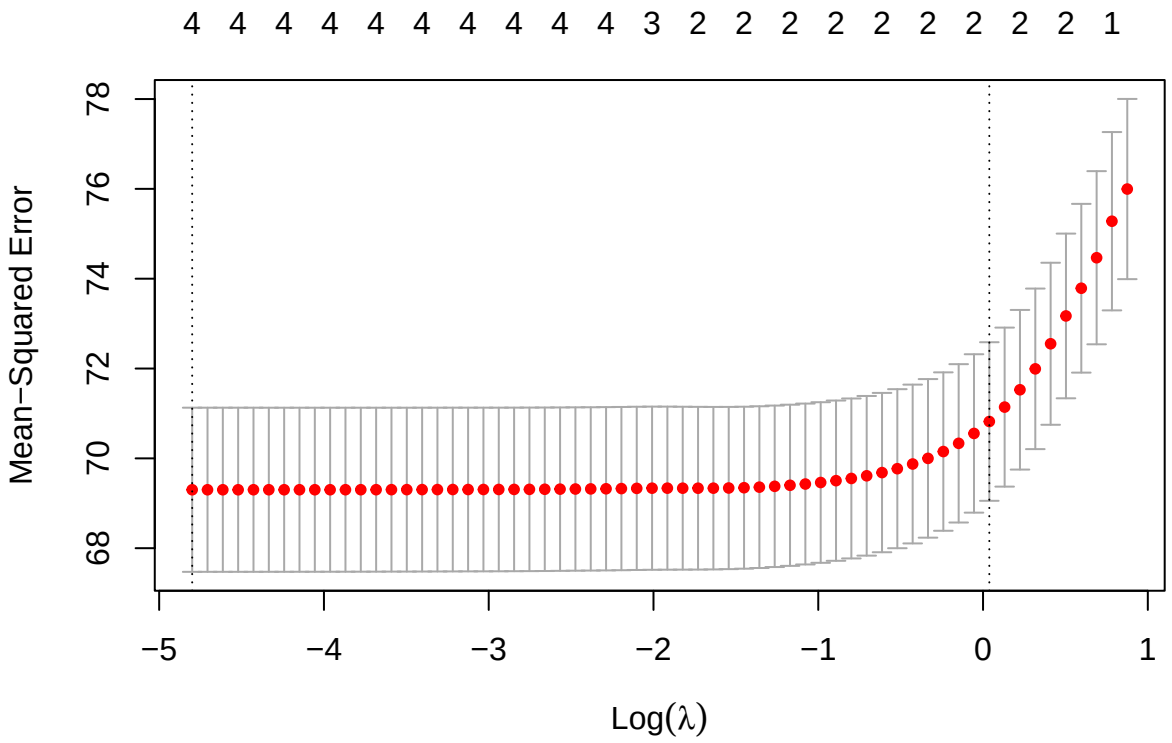
```
# Set seed for reproducibility
set.seed(1234)

# Perform cross-validation to determine best lambda
train_lasso <- cv.glmnet(
  x = predictor_train, # X as in our equation
  y = outcome_train, # Y as in our equation
  alpha = 1 # 0 = ridge, 1 = lasso
)
```

```
# Print performance
train_lasso
```

```
##
## Call:  cv.glmnet(x = predictor_train, y = outcome_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0082    62   69.30 1.827       4
## 1se 1.0388    10   70.82 1.766       2
```

Lambda and MSE can be visualized using the plot below:

```
# Plot output
plot(train_lasso)
```



To determine the lambda value that minimizes MSE, I will use the `lambda.min` command on `train_lasso`.

```
# Obtain best lambda
best_lambda <- train_lasso$lambda.min
best_lambda
```

```
## [1] 0.0082323
```

According to the above analysis, the lambda that should be used in the LASSO regression model is $\lambda = 0.0082323$.

I will use this lambda value to **build the final/best LASSO regression model** on the training set.

```
# Fit model with best lambda
best_lasso <- glmnet(
  x = predictor_train, # X as in our equation
```

```
  y = outcome_train, # Y as in our equation
  alpha = 1, # 0 = ridge, 1 = lasso
  lambda = best_lambda # set lambda
)
```

The **coefficients of the model** can now be obtained. This will inform whether to reject the Null Hypothesis.

```
# Obtain LASSO coefficients
lasso_coef <- unname(as.matrix(coef(best_lasso)))
lasso_coef
```

```
##            [,1]
## [1,] 69.4646094
## [2,]  1.2918031
## [3,] -0.2141833
## [4,]  0.1511653
## [5,] -1.9818861
```

The **formal model** can be written as follows:

$$Life\,Expectancy = \beta_0 + \beta_1 * Socioeconomic\,and\,Health\,Status + \beta_2 * Immunization\,Status$$
$$+\beta_3 * Childhood\,Health\,and\,Mortality + \beta_4 * Consumption\,and\,Substance\,Use + \varepsilon$$

Where:

- $\beta_0 = 69.4646094$
- $\beta_1 = 1.2918031$
- $\beta_2 = -0.2141833$
- $\beta_3 = 0.1511653$
- $\beta_4 = -1.9818861$

Because at least one of the $\beta_i s \neq 0$, the Alternative Hypothesis stated at the beginning of the report should be accepted.

To **determine significant variables**, I will examine the coefficients of the variables that are non-zero at the best lambda value. (Method obtained from ChatGPT)

```
# Determine significant variables
significant_vars <- which(coef(best_lasso, s = best_lambda) != 0)
significant_vars
```

```
## [1] 1 2 3 4 5
```

From this analysis, it can be said that the intercept and all variables are significant. Therefore, none should be removed the regression.

Now that the model has been built, I will **check the homoskedasticity and normality of the residuals** of the training and testing set.

Using the best model, I will obtain the predicted values on the training and testing data.

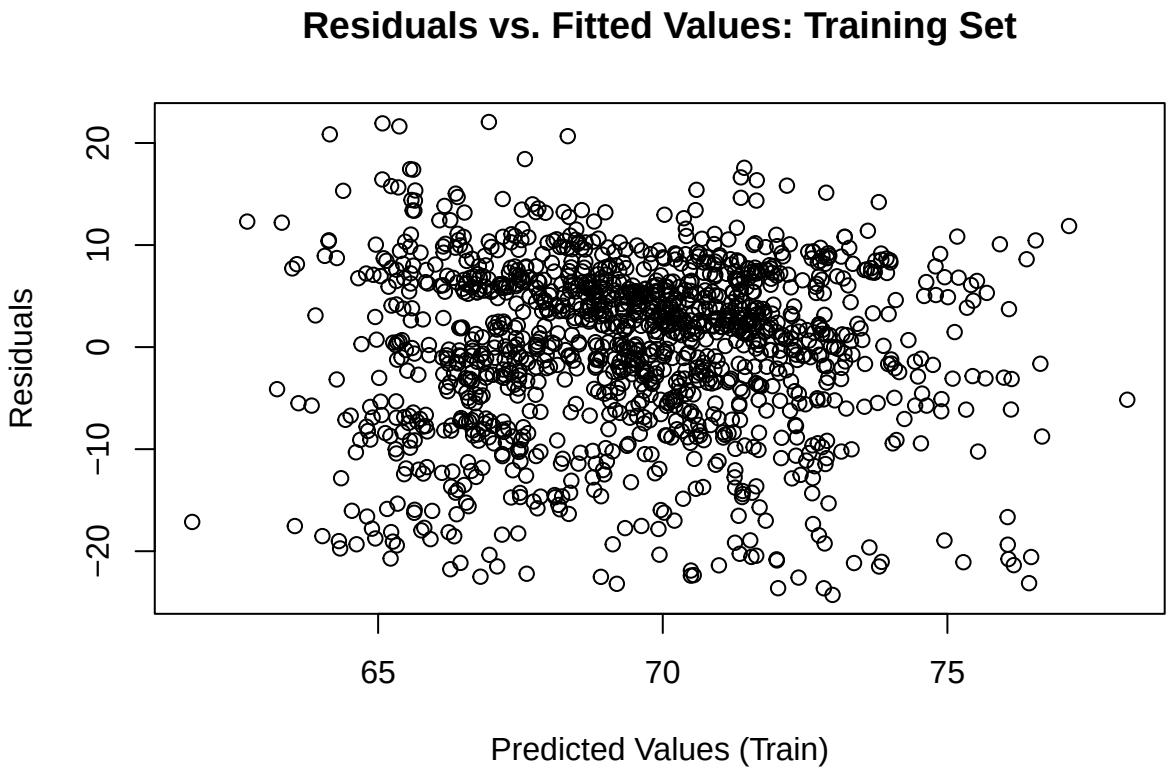
```
# Get predicted values
predicted_on_train <- predict(best_lasso, newx = predictor_train)
predicted_on_test <- predict(best_lasso, newx = predictor_test)
```

The residuals can be obtained by subtracting the predicted life expectancy values from the actual life expectancy values.

```r
# Get residuals
residuals_train <- outcome_train - predicted_on_train
residuals_test <- outcome_test - predicted_on_test
```

The train residuals may be plotted against the train predicted values to check for homoskedasticity.

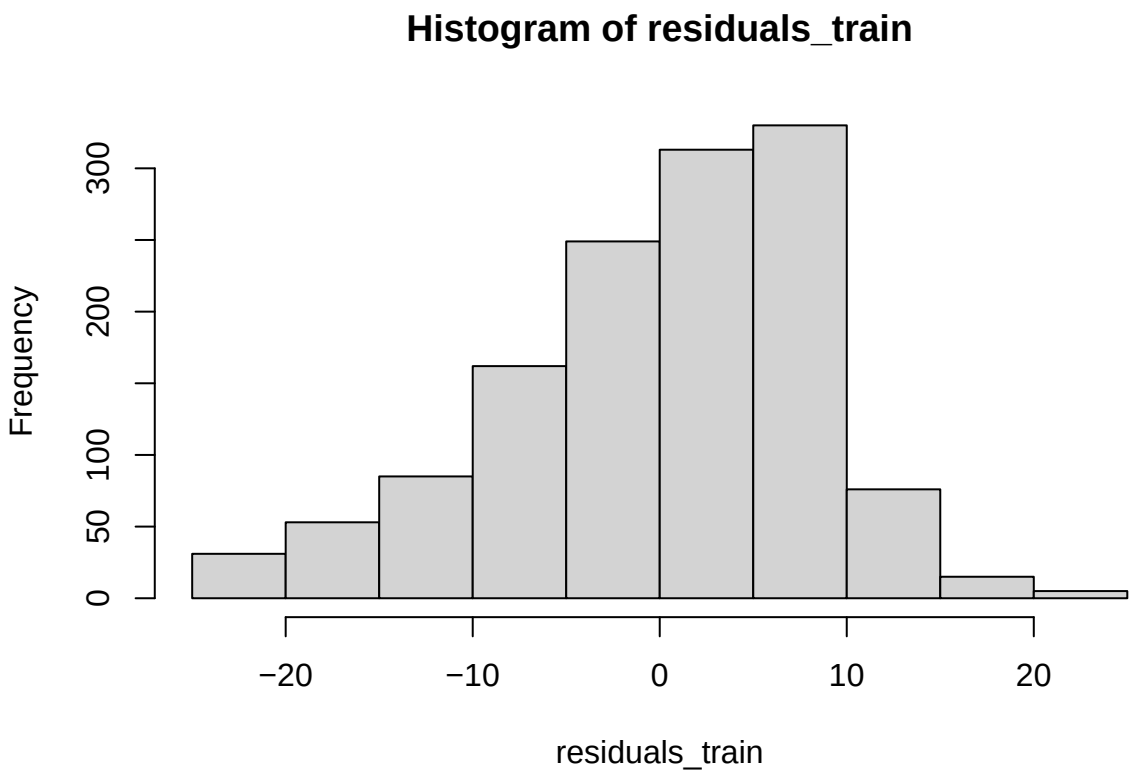
```r
# Plot residuals of training set
plot(predicted_on_train,
     residuals_train,
     main = "Residuals vs. Fitted Values: Training Set",
     xlab = "Predicted Values (Train)",
     ylab = "Residuals")
```



This residual plot looks pretty good! These values appear to be centered at 0, and the plot appears homoskedastic.

Now, let's examine a histogram of the train residuals to check whether they are normally distributed.

```r
hist(residuals_train)
```



At first glance, these residuals do not appear normally distributed. But let's investigate further.

I will check for normality using the Shapiro-Wilk test.

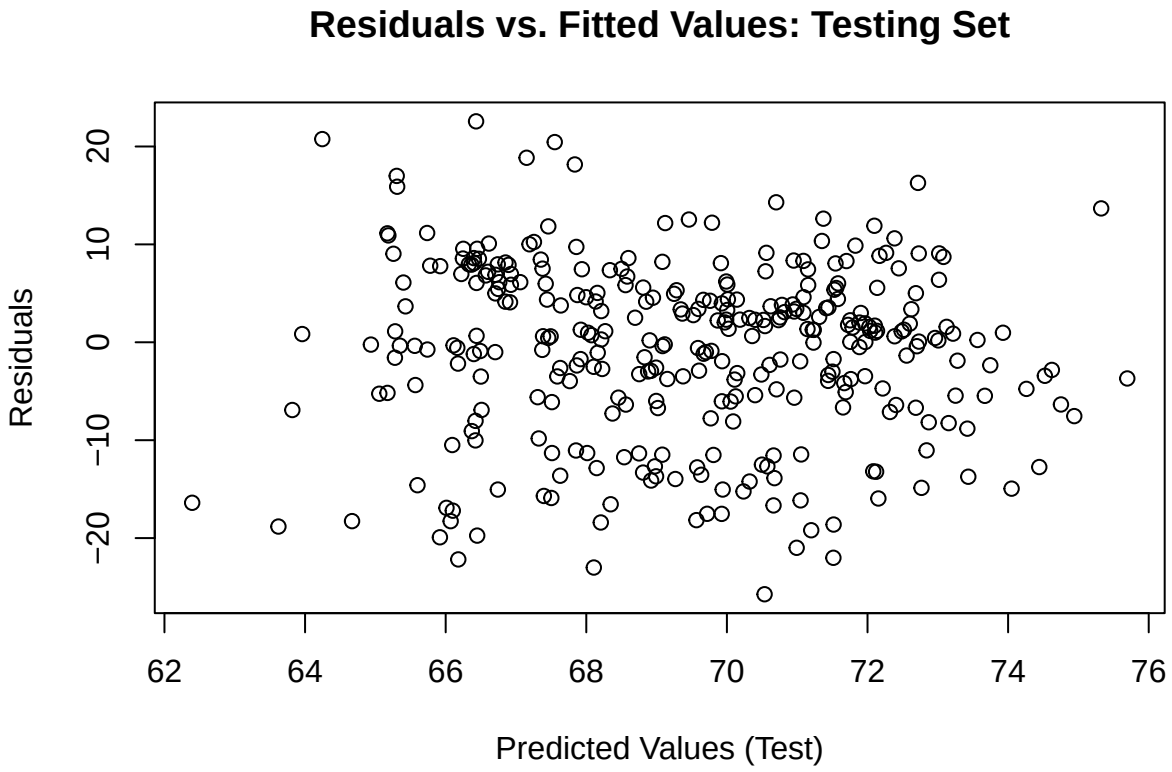
```
# Shapiro-Wilk test
shapiro.test(residuals_train)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_train
## W = 0.96801, p-value < 2.2e-16
```

This p-value is less than 0.05, indicating that the train residuals are not normally distributed.

We will do the same for the testing dataset.

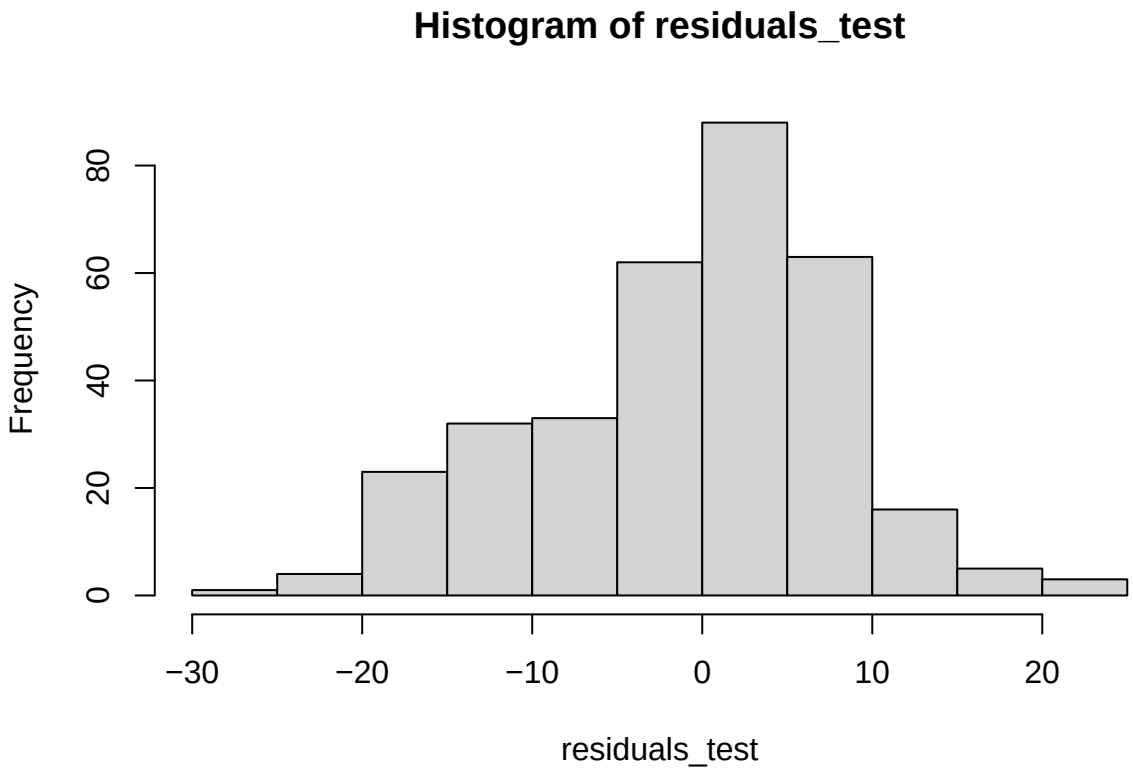
```
# Plot residuals of testing set
plot(predicted_on_test,
     residuals_test,
     main = "Residuals vs. Fitted Values: Testing Set",
     xlab = "Predicted Values (Test)",
     ylab = "Residuals")
```



This residual plot looks good as well. These values appear to be centered at 0, and the plot appears homoskedastic.

Now, let's examine a histogram of the test residuals to check whether they are normally distributed.

```
hist(residuals_test)
```



At first glance, these residuals do not appear normally distributed. But let's investigate further.

I will check for normality using the Shapiro-Wilk test.

```
# Shapiro-Wilk test
shapiro.test(residuals_test)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_test
## W = 0.97762, p-value = 5.158e-05
```

This p-value is less than 0.05, indicating that the test residuals are not normally distributed.

I will now **compute the R-squared value** of the model on both the training set and the testing set. A value close to 1 or -1 indicates a perfect correlation, while a value close to 0 indicates no correlation. (Method obtained from ChatGPT)

```
# Obtain train R-squared value
rss <- sum((outcome_train - predicted_on_train)^2)
tss <- sum((outcome_train - mean(outcome_train))^2)
r_squared <- 1 - rss/tss
r_squared
```

```
## [1] 0.09373124
```

The R-squared value of the training set was calculated to be **0.09373124**. This means that 9.37% of the variance in `Life.expectancy` can be explained by the four principal components. This is a **very** low value, meaning the linear model does not help much with the prediction. We would have liked to see an R-squared value of 0.5 or higher.

```
# Obtain test R-squared value
rss <- sum((outcome_test - predicted_on_test)^2)
tss <- sum((outcome_test - mean(outcome_test))^2)
r_squared <- 1 - rss/tss
r_squared
```

```
## [1] 0.03958735
```

The R-squared value of the testing set was calculated to be **0.03958735**. This means that 3.96% of the variance in `Life.expectancy` can be explained by the four principal components. Again, this is a **very** low value, and we would've liked to see something better.

Finally, I will **compute the RMSE value** of the model on both the **training data** and the **testing data**.

```
# Compute train RMSE
sqrt(mean(residuals_train^2))
```

```
## [1] 8.30322
```

The RMSE value of this model on the training set is **8.30322**. This means that on average, the model's life expectancy prediction on the training set is off by 8.30322 years.

```
# Compute test RMSE
sqrt(mean(residuals_test^2))
```

```
## [1] 8.864313
```

The RMSE value of this model on the testing set is **8.864313**. This means that on average, the model's life expectancy prediction on the testing set is off by 8.864313 years.

The model performs slightly better on the training set than the testing set. However, the fact that these R-squared and RMSE values are so close is a good sign that the model is not overfit, and that it would perform equally well on real-world data it has never seen before.

Discussion

The purpose of this analysis was to determine if **life expectancy** could be predicted through a LASSO linear regression using the independent variables in `life_expectancy.csv` after a PCA dimension reduction.

Four principal components were obtained through the PCA dimension reduction:

1. Socioeconomic and Health Status
2. Immunization Status
3. Childhood Health and Mortality
4. Consumption and Substance Use

Our Formal Null Hypothesis specified that the coefficients of these principal components were all equal to 0, while our Formal Alternative Hypothesis specified that at least one of these coefficients was not equal to 0. Through the analysis performed above, the coefficients were determined to be non-zero:

$$Life\,Expectancy = 69.4646094 + 1.2918031 * Socioeconomic\,and\,Health\,Status - 0.2141833 * Immunization\,Status$$

$$+ 0.1511653 * Childhood\,Health\,and\,Mortality - 1.9818861 * Consumption\,and\,Substance\,Use + \varepsilon$$

Therefore, we may accept the **Alternative Hypothesis** that **life expectancy is related to the principal components.**

The final model `best_lasso` was obtained on the training set by determining the lambda value that minimized MSE. Unfortunately, this best regression model still had many flaws. The residuals on the training and testing data were not normally distributed. Additionally, this model had an **R-squared value** of **0.09373124** on the training set and **0.03958735** on the testing set. These are both terrible R-squared values. An R-squared value of 0 indicates no correlation, so this model contributes very little to our understanding of the relationship in question. Finally, the model had an **RMSE** of **8.30322** on the training set and **8.864313** on the testing set, meaning that on average, the model's life expectancy prediction was off by about 8 years. In the context of predicting life expectancy, I don't think this is a good RMSE value. This yields a confidence interval of about 16 years, which is a wide range when considering the lifespan of a human being.

Despite the poor metrics listed above, it is somewhat promising to see that the R-squared and RMSE values on the training and testing sets are similar. This indicates that the model is not overfit, and that the model would perform equally well (though this is not "well" by any means) on data it has never seen before.

In summary, this model is not a good one, and it should not be deployed in a real-world setting. Perhaps if another model were constructed to predict life expectancy, it should use other variables or a different number of principal components.