6372: Project 1

Megan Ball, Matt Farrow, Neddy Nyatome

# Introduction

Using the World Health Organization (WHO) data compiled by Kumar Rajarshi, Deeksha Russell, and Duan Wang, we developed three different models: - The first model was designed to be easily interpreted using linear regression. - The second model was designed to be used as a predictive tool using linear regression. - The third model was developed using non-parametric methods for prediction.

# Data Description

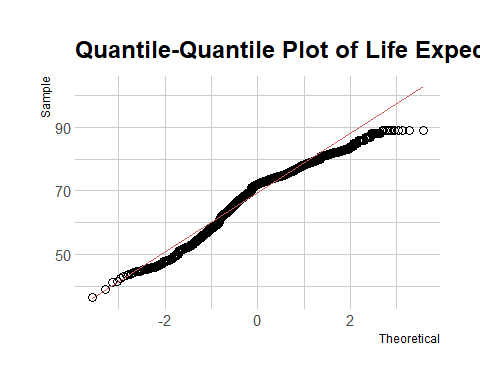
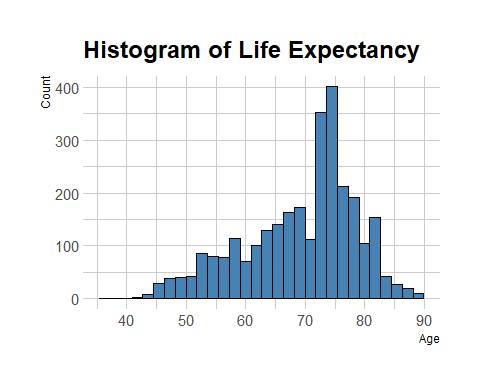
The description and context of the Life Expectancy (WHO) data set can be found [here](https://www.kaggle.com/kumarajarshi/life-expectancy-who). Data has been compiled from several different data sets into a final data set that represents health factors for 193 countries between the years of 2000-2015.

## Rows: 2,938  
## Columns: 22  
## $ country <chr> "Afghanistan", "Afghanistan", "Afgh...  
## $ year <dbl> 2015, 2014, 2013, 2012, 2011, 2010,...  
## $ status <chr> "Developing", "Developing", "Develo...  
## $ life\_expectancy <dbl> 65.0, 59.9, 59.9, 59.5, 59.2, 58.8,...  
## $ adult\_mortality <dbl> 263, 271, 268, 272, 275, 279, 281, ...  
## $ infant\_deaths <dbl> 62, 64, 66, 69, 71, 74, 77, 80, 82,...  
## $ alcohol <dbl> 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,...  
## $ percentage\_expenditure <dbl> 71.279624, 73.523582, 73.219243, 78...  
## $ hepatitis\_b <dbl> 65, 62, 64, 67, 68, 66, 63, 64, 63,...  
## $ measles <dbl> 1154, 492, 430, 2787, 3013, 1989, 2...  
## $ bmi <dbl> 19.1, 18.6, 18.1, 17.6, 17.2, 16.7,...  
## $ under\_five\_deaths <dbl> 83, 86, 89, 93, 97, 102, 106, 110, ...  
## $ polio <dbl> 6, 58, 62, 67, 68, 66, 63, 64, 63, ...  
## $ total\_expenditure <dbl> 8.16, 8.18, 8.13, 8.52, 7.87, 9.20,...  
## $ diphtheria <dbl> 65, 62, 64, 67, 68, 66, 63, 64, 63,...  
## $ hiv\_aids <dbl> 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, ...  
## $ gdp <dbl> 584.25921, 612.69651, 631.74498, 66...  
## $ population <dbl> 33736494, 327582, 31731688, 3696958...  
## $ thinness\_1\_19\_years <dbl> 17.2, 17.5, 17.7, 17.9, 18.2, 18.4,...  
## $ thinness\_5\_9\_years <dbl> 17.3, 17.5, 17.7, 18.0, 18.2, 18.4,...  
## $ income\_composition\_of\_resources <dbl> 0.479, 0.476, 0.470, 0.463, 0.454, ...  
## $ schooling <dbl> 10.1, 10.0, 9.9, 9.8, 9.5, 9.2, 8.9...

Looking at the data, there are 2,938 observations and 22 variables that cover various social, economic, and health-related factors.

# Exploratory Data Analysis

We’ll start by taking a closer look at life expectancy.



As we might expect, life expectancy tends to skew towards the older side, but for the purposes of this project we won’t worry about that. The Q-Q plot shows some slight deviations from normality towards the edges, but after trying various transformations we have decided to proceed with the original data.

## Check for Correlation

We next moved into looking at correlation (Figure @ref(fig:corr-1)). We began examining various correlation matrices to determine whether there were variable(s) that should be addressed, and started by removing redundant variables and picking ones with fewer NA values (Figure @ref(fig:corr-2).

Finally, we looked at what happens when we also remove population, since it has minimal correlation to life\_expectancy (Figure @ref(fig:corr-3).

We’ll proceed, dropping the columns we’ve looked at.

df\_clean <- df\_clean %>%   
 select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years, population))

## Deal with Missing Values

Our next task was to address the missing values in the data set (Figure @ref(fig:missing-1)). We dropped all rows where life expectancy is NA, and limited the scope of our analysis to not include these countries (Figure @ref(fig:missing-2)).

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

## NULL

## [1] Cook Islands Dominica Marshall Islands   
## [4] Monaco Nauru Niue   
## [7] Palau Saint Kitts and Nevis San Marino   
## [10] Tuvalu   
## 193 Levels: Afghanistan Albania Algeria Angola Antigua and Barbuda ... Zimbabwe

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

## NULL

We have now excluded these countries from our scope: Cook Islands, Dominica, Marshall Islands, Monaco, Nauru, Niue, Palau, Saint Kitts and Nevis, San Marino, and Tuvalu.

Hepatitis B is now our variable with the most NA’s at 553. Let’s see if there is either a year or a country that has the majority of the NA’s.

## # A tibble: 98 x 2  
## # Groups: country [98]  
## country missing  
## <fct> <int>  
## 1 Denmark 16  
## 2 Finland 16  
## 3 Hungary 16  
## 4 Iceland 16  
## 5 Japan 16  
## 6 Norway 16  
## 7 Slovenia 16  
## 8 Switzerland 16  
## 9 United Kingdom of Great Britain and Northern Ireland 16  
## 10 South Sudan 15  
## # ... with 88 more rows

In looking at the relationship between Hepatitis B and Life Expectancy (Figure @ref(fig:hep-life)), our options are to either drop all of the NA’s, impute the values, or fill with 0’s. For our interpretable model, we made the decision to drop the hepatitis\_b variable along with the remainder of the NA’s and we will revisit them for our predictive model (Figure @ref(fig:missing-3)).

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Saving 5 x 4 in image  
## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

## NULL

## Analysis of Years

Now that we’ve subsetted our variables and dealt with NA’s, we noticed that our feature engineering dropped almost all of the records from 2015. After a number of looks at the data, we decided to only use the most recent “good” sample size (2011-2014).

## # A tibble: 16 x 2  
## year n  
## <dbl> <int>  
## 1 2000 168  
## 2 2001 168  
## 3 2002 169  
## 4 2003 170  
## 5 2004 170  
## 6 2005 170  
## 7 2006 171  
## 8 2007 171  
## 9 2008 171  
## 10 2009 171  
## 11 2010 171  
## 12 2011 171  
## 13 2012 171  
## 14 2013 171  
## 15 2014 171  
## 16 2015 2

# Build the Model

Now that we’ve finished our feature engineering, let’s start building our interpretable model.

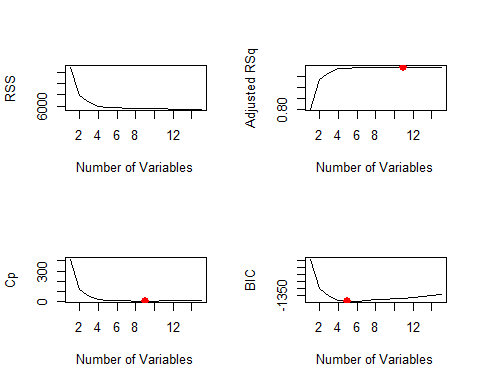
# Look at the names of reg\_summary  
names(reg\_summary)

## [1] "which" "rsq" "rss" "adjr2" "cp" "bic" "outmat" "obj"

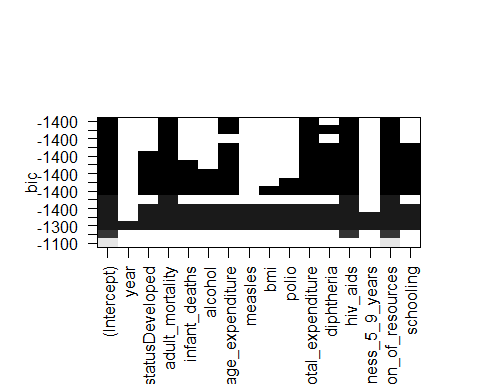
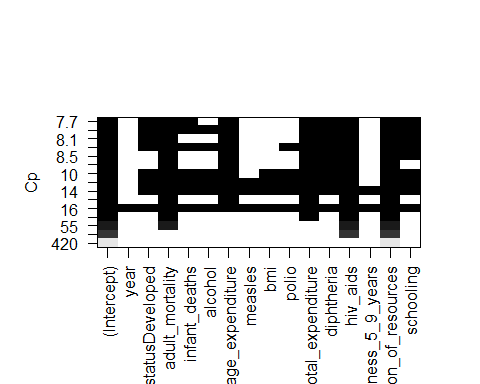
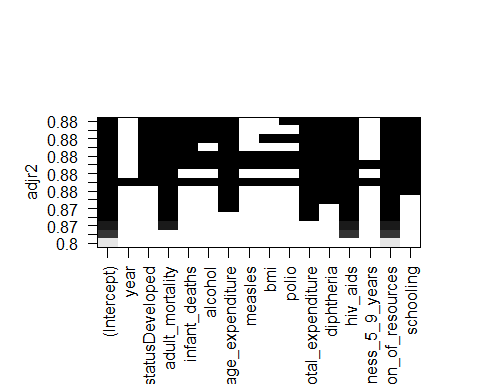
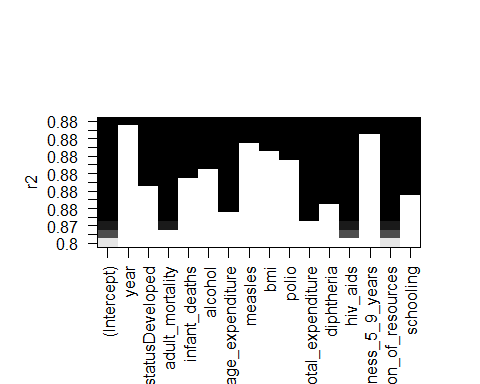
# What are the R-squared values?  
reg\_summary$rsq

## [1] 0.8008008 0.8543421 0.8672807 0.8740462 0.8754281 0.8765758 0.8771892  
## [8] 0.8776295 0.8780650 0.8783576 0.8786491 0.8786857 0.8787063 0.8787223  
## [15] 0.8787382

## Plot the RSS, adjusted , , and BIC for all Models



Now we’ll look at the selected variables for the best model with a given number of predictors.



The model with the lowest BIC is the five-variable model that contains adult\_mortality, percentage\_expenditure, total\_expenditure, hiv\_aids, and income\_composition\_of\_resources. We can use the coef() function to see the coefficient estimates associated with this model.

## (Intercept) adult\_mortality   
## 47.7674816027 -0.0128050404   
## percentage\_expenditure total\_expenditure   
## 0.0001382056 0.2529652954   
## hiv\_aids income\_composition\_of\_resources   
## -0.9321014506 35.8680622598

## Run Forward and Backward Stepwise Selection

The coefficients of the full, forward, and backward models all returned the same coefficients.

variables

full

fwd

bwd

(Intercept)

47.0500

47.0500

47.0500

adult\_mortality

-0.0128

-0.0128

-0.0128

total\_expenditure

0.2707

0.2707

0.2707

hiv\_aids

-0.9169

-0.9169

-0.9169

income\_composition\_of\_resources

36.9466

36.9466

36.9466

## Choose Using Validation Set

# Find the best model  
val\_errors

## [1] 15.170997 11.270808 10.081603 9.479321 9.470498 9.471813 9.558630  
## [8] 9.520908

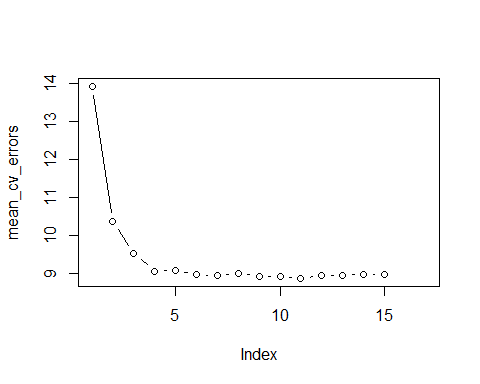
coef(regfit\_best, which.min(val\_errors))

## (Intercept) adult\_mortality   
## 47.5535106758 -0.0113716228   
## percentage\_expenditure total\_expenditure   
## 0.0001798031 0.2109324023   
## hiv\_aids income\_composition\_of\_resources   
## -0.9767312592 36.1681983992

## (Intercept) adult\_mortality   
## 47.7674816027 -0.0128050404   
## percentage\_expenditure total\_expenditure   
## 0.0001382056 0.2529652954   
## hiv\_aids income\_composition\_of\_resources   
## -0.9321014506 35.8680622598

## Choose Using Cross-Validation

## 1 2 3 4 5 6 7 8   
## 13.928763 10.368283 9.519365 9.063995 9.091764 8.965043 8.939566 9.008205   
## 9 10 11 12 13 14 15 16   
## 8.931195 8.921283 8.871960 8.943193 8.960145 8.971014 8.971249 NA   
## 17   
## NA



## (Intercept) adult\_mortality   
## 47.05001524 -0.01279054   
## total\_expenditure hiv\_aids   
## 0.27072370 -0.91691624   
## income\_composition\_of\_resources   
## 36.94661482

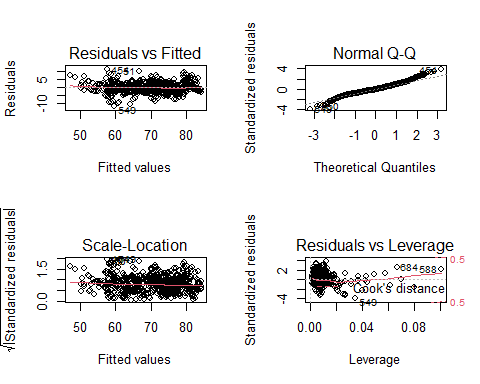
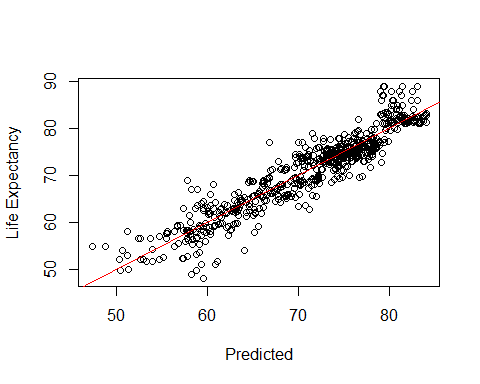
## Build and Inspect the Final Model

##   
## Call:  
## lm(formula = life\_expectancy ~ adult\_mortality + total\_expenditure +   
## hiv\_aids + income\_composition\_of\_resources, data = df\_interp)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -11.4387 -1.7704 -0.2258 1.6342 11.2075   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 47.050015 0.834957 56.350 < 2e-16 \*\*\*  
## adult\_mortality -0.012791 0.001558 -8.209 1.12e-15 \*\*\*  
## total\_expenditure 0.270724 0.044828 6.039 2.55e-09 \*\*\*  
## hiv\_aids -0.916916 0.074060 -12.381 < 2e-16 \*\*\*  
## income\_composition\_of\_resources 36.946615 0.991990 37.245 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.967 on 679 degrees of freedom  
## Multiple R-squared: 0.874, Adjusted R-squared: 0.8733   
## F-statistic: 1178 on 4 and 679 DF, p-value: < 2.2e-16

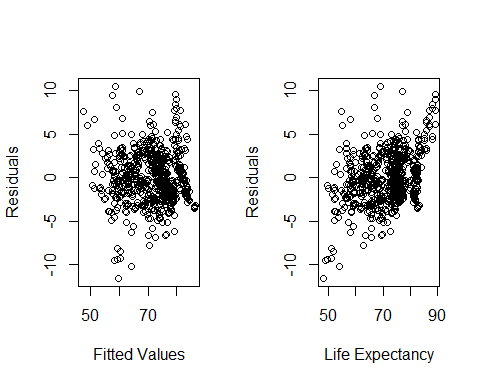
## 2.5 % 97.5 %  
## (Intercept) 45.41060756 48.689422926  
## adult\_mortality -0.01584978 -0.009731309  
## total\_expenditure 0.18270630 0.358741104  
## hiv\_aids -1.06233059 -0.771501898  
## income\_composition\_of\_resources 34.99887756 38.894352083

The MSE for our model is mean(final\_model\_summary$residuals ^ 2)

## Check Diagnostics



## Fit our Model to our Data Set



## Interpretation of Regression Coefficients

We are 95% confident that the model’s intercept is between (45.41, 48.689) and the true regression coefficient’s for the predicted variables are: adult mortality (-0.016, -0.01), total expenditure (0.183, 0.359), HIV/AIDS (-1.062, -0.772), and income composition of resources (34.999, 38.894).

# Linear Prediction Model

Let’s start again with the df\_clean data set to perform feature engineering. We have already removed at this point variables that are highly correlated and rows/countries with NA for life expectancy. This leaves us with 17 other potential predictors and still quite a few NA values. We know we cannot have NA values to perform Ridge Regression or LASSO so need to either remove them, impute them, or change them to 0.

## Deal with Missing Values

variable

missing

country

0

year

0

status

0

life\_expectancy

0

adult\_mortality

0

infant\_deaths

0

alcohol

193

percentage\_expenditure

0

hepatitis\_b

553

measles

0

bmi

32

polio

19

total\_expenditure

226

diphtheria

19

hiv\_aids

0

thinness\_5\_9\_years

32

income\_composition\_of\_resources

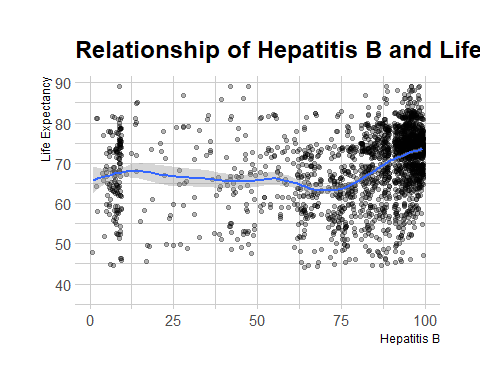
160

schooling

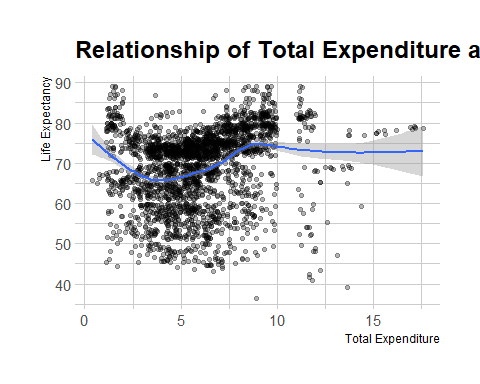
160

Let’s start by checking the relationships of each variable with > 100 NA’s to see if they appear significant.

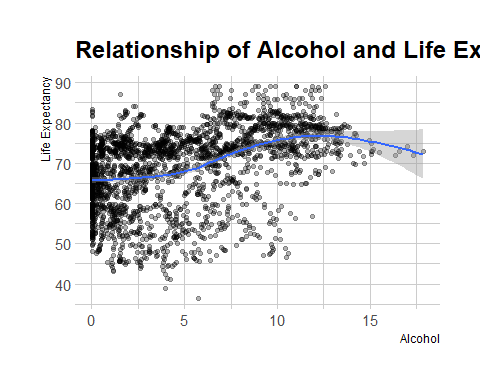
## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



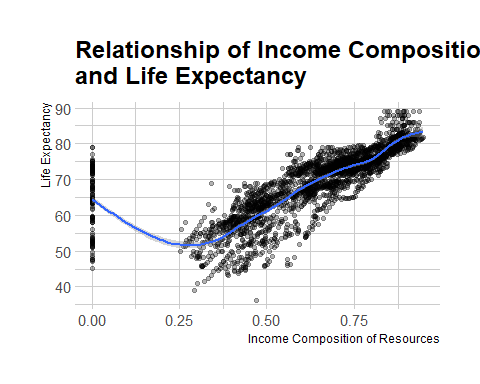
## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



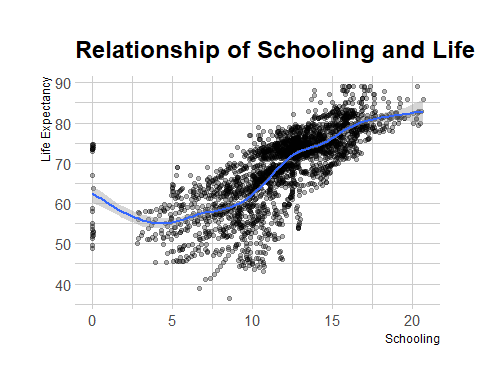
## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



The trend looks pretty flat for hepatitis\_b, total\_expenditure, and alcohol so I think it is safe for us to go ahead and remove these columns. However, there appears to be some relationship for schooling and possibly for income composition of resources as well, so instead of imputing these values let’s go ahead and remove the NA’s for these two variables.

variable

missing

country

0

year

0

status

0

life\_expectancy

0

adult\_mortality

0

infant\_deaths

0

percentage\_expenditure

0

measles

0

bmi

0

polio

0

diphtheria

0

hiv\_aids

0

thinness\_5\_9\_years

0

income\_composition\_of\_resources

0

schooling

0

We now have 2,728 observations and 15 variables. Let’s proceed to variable selection.

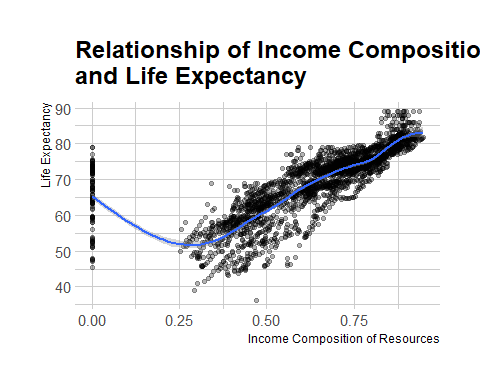
#replot after removing Na's  
  
# income composition of resources  
df\_predict %>%  
 ggplot(aes(x = income\_composition\_of\_resources, y = life\_expectancy)) +  
 # geom\_point() +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Income Composition of Resources\nand Life Expectancy",  
 x = "Income Composition of Resources",  
 y = "Life Expectancy") +  
 theme\_ipsum()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database

## Warning in grid.Call.graphics(C\_text, as.graphicsAnnot(x$label), x$x, x$y, :  
## font family not found in Windows font database

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database



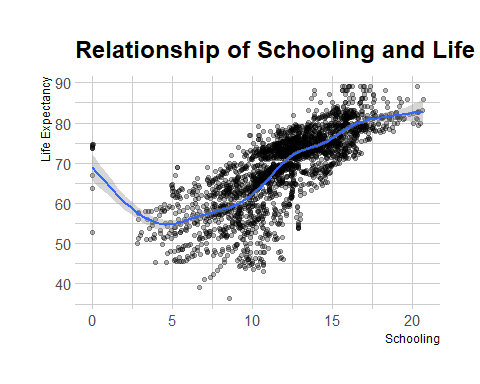
# schooling  
df\_predict %>% ggplot(aes(x = schooling, y = life\_expectancy)) +  
 # geom\_point() +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Schooling and Life Expectancy",  
 x = "Schooling",  
 y = "Life Expectancy") +  
 theme\_ipsum()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database

## Warning in grid.Call.graphics(C\_text, as.graphicsAnnot(x$label), x$x, x$y, :  
## font family not found in Windows font database

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database



## Ridge Regression

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.0.2

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.0-2

# Ridge regression and lasso require the format 'x matrix' and 'y'. The  
# model.matrix() function produces a matrix and automatically transforms  
# qualitative variables into dummy variables.  
x <- model.matrix(life\_expectancy ~ ., df\_predict[,-1])[, -1]  
y <- df\_predict$life\_expectancy  
  
# Run ridge regression  
ridge\_mod <- glmnet(x, y, alpha = 0)  
dim(coef(ridge\_mod))

## [1] 14 100

# Split data into training and testing sets  
set.seed(1)  
train <- sample(1:nrow(x), nrow(x)/2)  
test <- (-train)  
y\_test <- y[test]  
  
# Fit a ridge regression model on the training set, and evaluate its MSE on the  
# test set  
ridge\_mod <- glmnet(x[train, ], y[train], alpha = 0)  
ridge\_pred <- predict(ridge\_mod, s = 4, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)

## [1] 17.4224

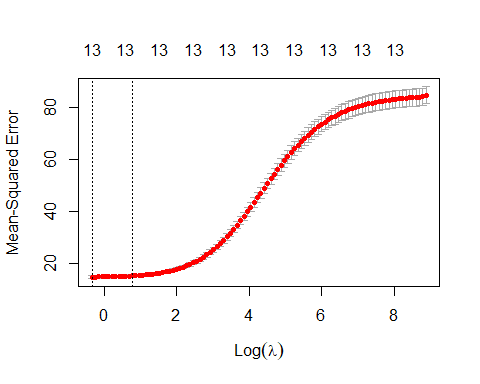
#[1] 17.4224  
  
# The test MSE is 17.4224. If we had simply fit a model with just an intercept,  
# we would have observed each test observation using the mean of the training  
# observations. In that case, we could compute the test set MSE like this:  
mean((mean(y[train]) - y\_test) ^ 2)

## [1] 89.45262

#[1] 89.45262  
  
# We could also get the same result by fitting a ridge regression model with a  
# very large value of ƛ.  
ridge\_pred <- predict(ridge\_mod, s = 1e10, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)

## [1] 89.45262

#[1] 89.45262  
  
# Use cross-validation to choose the tuning parameter ƛ.  
set.seed(1)  
cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 0)  
plot(cv\_out)



bestlambda <- cv\_out$lambda.min  
bestlambda

## [1] 0.715565

# What is the test MSE associated with bestlambda?  
ridge\_pred <- predict(ridge\_mod, s = bestlambda, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)

## [1] 16.2937

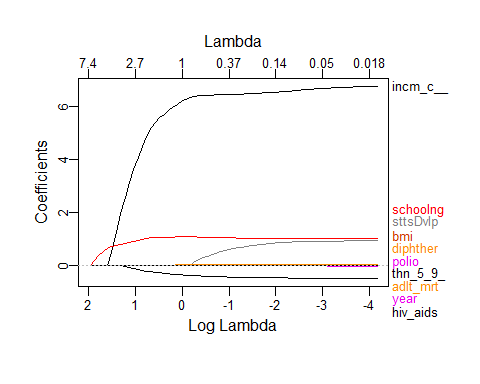
#[1] 16.2937  
  
# Refit the ridge regression model on the full data set using the value of ƛ  
# chosen by cross-validation  
out <- glmnet(x, y, alpha = 0)  
predict(out, type = "coefficients", s = bestlambda)[1:14,]

## (Intercept) year   
## 9.862342e+01 -2.329948e-02   
## statusDeveloped adult\_mortality   
## 1.075181e+00 -1.676717e-02   
## infant\_deaths percentage\_expenditure   
## -4.056071e-04 2.907730e-04   
## measles bmi   
## -1.903933e-05 3.792386e-02   
## polio diphtheria   
## 2.675770e-02 3.222588e-02   
## hiv\_aids thinness\_5\_9\_years   
## -4.664553e-01 -4.739236e-02   
## income\_composition\_of\_resources schooling   
## 8.028355e+00 7.932906e-01

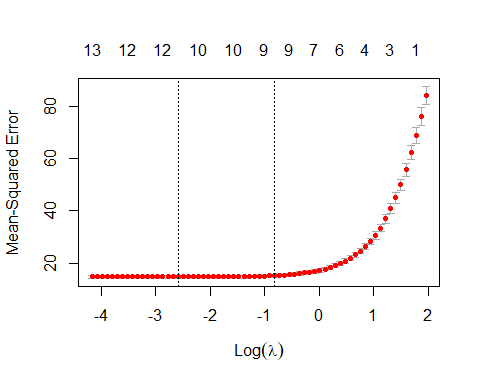
From ridge regression, we see the variables with the largest coefficients (in magnitude) are statusDeveloped, income\_composition\_of\_resources, schooling, and hiv\_aids. We will proceed with LASSO to determine another method of variable selection.

## Lasso

# Fit the lasso model  
lasso\_mod <- glmnet(x[train, ], y[train], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod)



# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 1)  
plot(cv\_out)



bestlambda <- cv\_out$lambda.min  
lasso\_pred <- predict(lasso\_mod, s = bestlambda, newx = x[test, ])  
mean((lasso\_pred - y\_test) ^ 2)

## [1] 16.41776

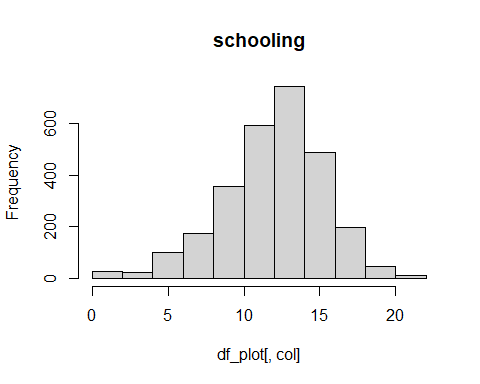
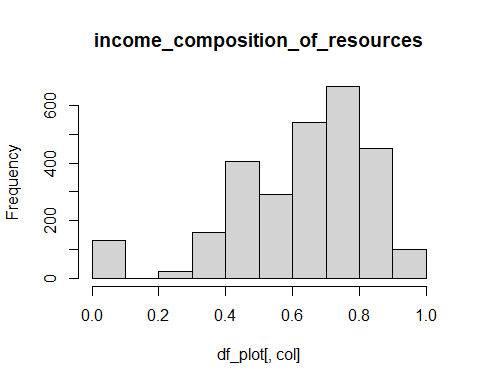
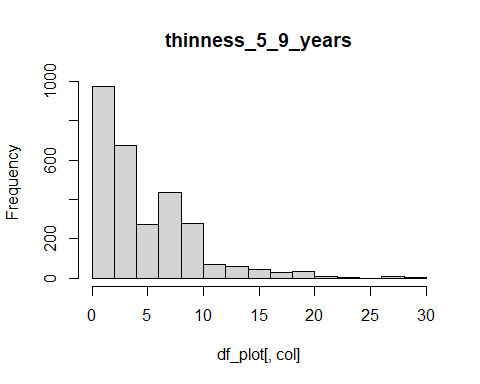
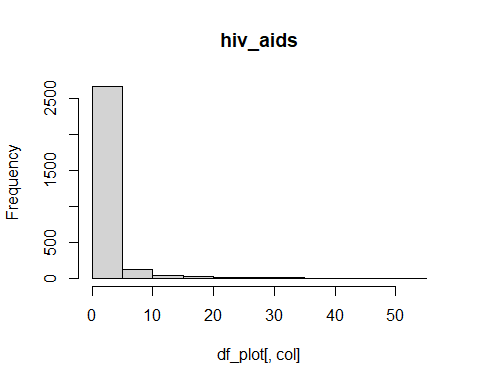
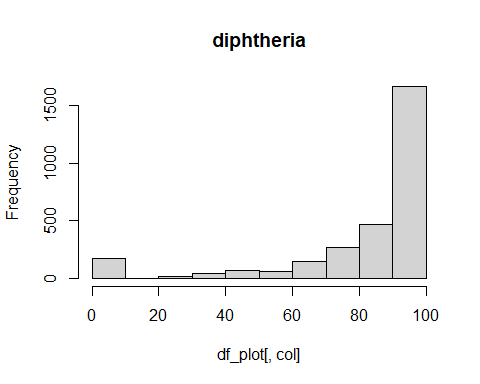
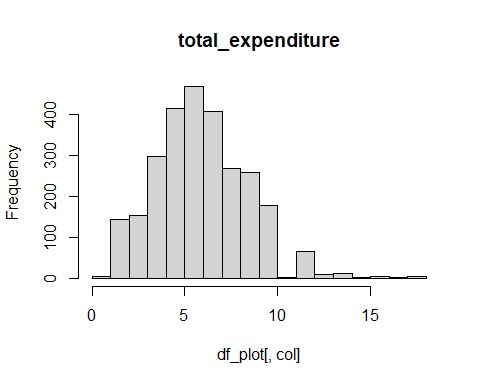
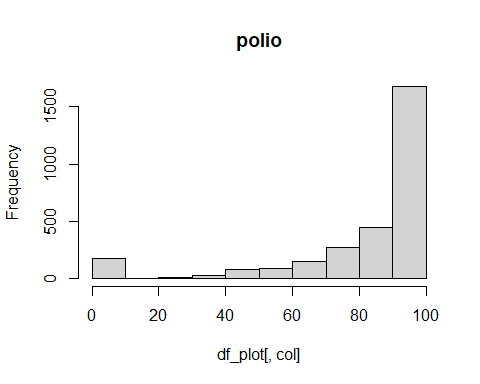
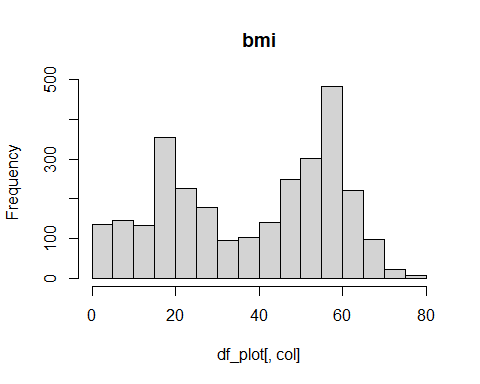
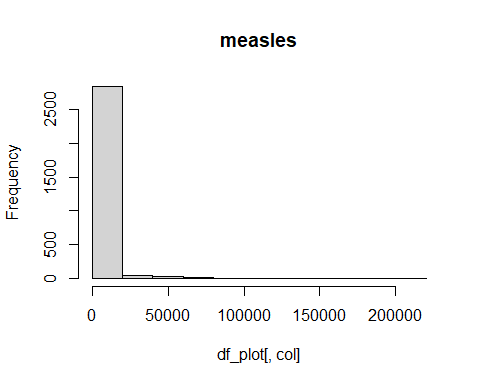
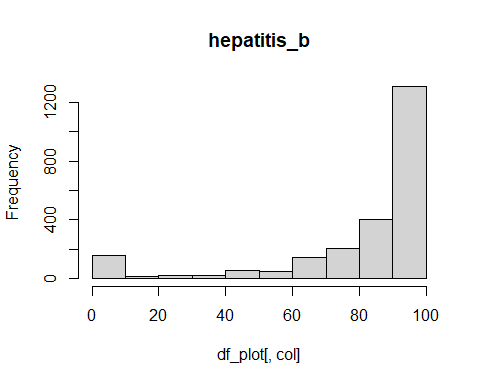
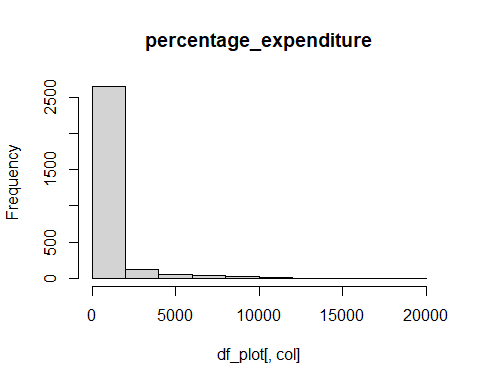
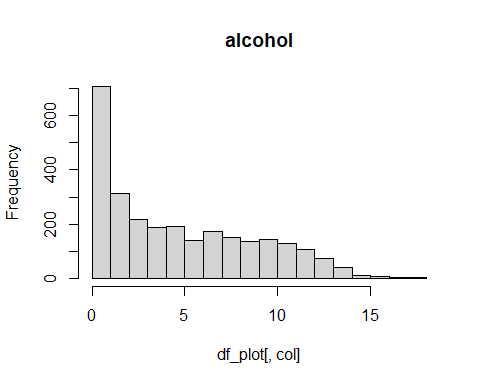
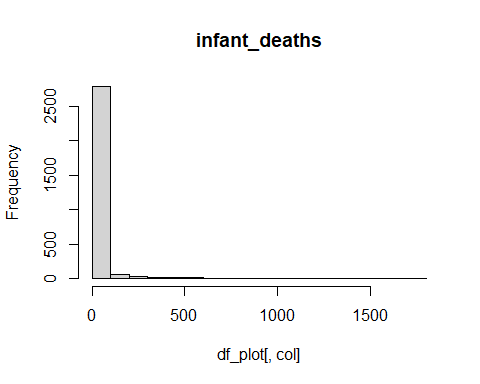
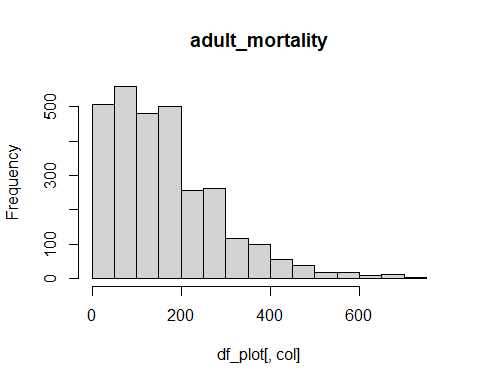
#[1] 16.41776  
  
# Compute lasso coefficients  
out <- glmnet(x, y, alpha = 1)  
lasso\_coef <- predict(out, type = "coefficients", s = bestlambda)[1:14,]  
lasso\_coef[lasso\_coef != 0]

## (Intercept) year   
## 8.898859e+01 -1.877109e-02   
## statusDeveloped adult\_mortality   
## 7.605516e-01 -1.700651e-02   
## infant\_deaths percentage\_expenditure   
## -1.898545e-04 2.696144e-04   
## measles bmi   
## -1.553535e-05 3.326365e-02   
## polio diphtheria   
## 2.416911e-02 3.019979e-02   
## hiv\_aids thinness\_5\_9\_years   
## -4.871484e-01 -2.907157e-02   
## income\_composition\_of\_resources schooling   
## 7.668078e+00 9.067694e-01

LASSO variable selection does not help us reduce our variables down, as it indicates all variables are non-zero, but does highlight the same top 4 as ridge regression: income\_composition\_of\_resources, schooling, statusDeveloping and hiv\_aids. Our ASE is comparable to ridge regression around 16.4, which is worse than our interpretable model (although our interpretable model is on a much smaller scale). Let’s proceed with imputation instead or removal of the NA’s on our data set to see if that improves performance, as our primary goal is just prediction.

## Impute Values

#plot distributions  
df\_plot <- as.data.frame(df\_clean)  
for (col in 5:ncol(df\_plot)) {  
 hist(df\_plot[,col], main=names(df\_plot[col]))  
}



# Copy the data  
df\_predict2 <- df\_clean %>%   
 select(-c(hepatitis\_b))  
  
# Impute the data set  
df\_predict2 <- df\_predict2 %>%  
 mutate(across(  
 c(polio, total\_expenditure, diphtheria),  
 ~ replace\_na(., median(.x, na.rm = TRUE))  
 )) %>%   
 mutate(across(  
 c(bmi, income\_composition\_of\_resources, schooling),  
 ~ replace\_na(., mean(.x, na.rm = TRUE))  
 )) %>%   
 mutate(across(  
 c(alcohol, thinness\_5\_9\_years),  
 ~ replace\_na(., 0)  
 ))  
  
  
#check for NA's  
tibble(variable = names(colSums(is.na(df\_predict2))),  
 missing = colSums(is.na(df\_predict2))) %>%   
 gt()

variable

missing

country

0

year

0

status

0

life\_expectancy

0

adult\_mortality

0

infant\_deaths

0

alcohol

0

percentage\_expenditure

0

measles

0

bmi

0

polio

0

total\_expenditure

0

diphtheria

0

hiv\_aids

0

thinness\_5\_9\_years

0

income\_composition\_of\_resources

0

schooling

0

Based on the histograms and distributions, we decided to impute the median for polio, total\_expenditure, and diptheria, the mean for bmi, income\_composition\_of\_resources, and schooling, and 0 for alcohol and thinness\_5\_9\_years. Now our data set has 2,928 observations with 18 variables including our response variable.

## Ridge Regression 2.0

# Ridge regression and lasso require the format 'x matrix' and 'y'. The  
# model.matrix() function produces a matrix and automatically transforms  
# qualitative variables into dummy variables.  
sum(is.na(df\_predict2))

## [1] 0

x2 <- model.matrix(life\_expectancy ~ ., df\_predict2[,-1])[, -1]  
y2 <- df\_predict2$life\_expectancy  
  
# Run ridge regression  
ridge\_mod2 <- glmnet(x2, y2, alpha = 0)  
dim(coef(ridge\_mod2))

## [1] 16 100

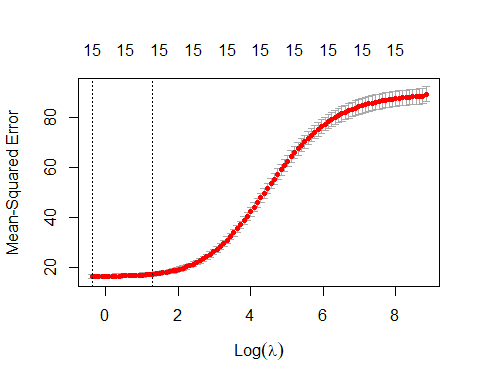
# Split data into training and testing sets  
set.seed(1)  
train2 <- sample(1:nrow(x2), nrow(x2)/2)  
test2 <- (-train2)  
y\_test2 <- y2[test2]  
  
# Fit a ridge regression model on the training set, and evaluate its MSE on the  
# test set  
ridge\_mod2 <- glmnet(x2[train2,], y2[train2], alpha = 0)  
ridge\_pred2 <- predict(ridge\_mod2, s = 4, newx = x2[test2, ])  
mean((ridge\_pred2 - y\_test2) ^ 2)

## [1] 19.39197

#[1] 19.39197  
  
# The test MSE is 19.39197. If we had simply fit a model with just an intercept,  
# we would have observed each test observation using the mean of the training  
# observations. In that case, we could compute the test set MSE like this:  
mean((mean(y2[train2]) - y\_test2) ^ 2)

## [1] 91.97787

#[1] 88.25497  
  
# Use cross-validation to choose the tuning parameter ƛ.  
set.seed(1)  
cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 0)  
plot(cv\_out2)



bestlambda2 <- cv\_out2$lambda.min  
bestlambda2

## [1] 0.6900917

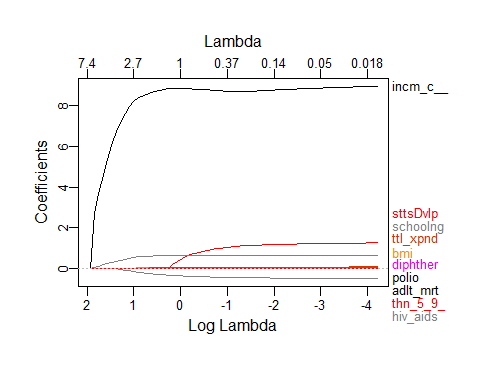
#[1] 0.6900917  
  
# What is the test MSE associated with bestlambda?  
ridge\_pred2 <- predict(ridge\_mod2, s = bestlambda2, newx = x2[test2, ])  
mean((ridge\_pred2 - y\_test2) ^ 2)

## [1] 18.68608

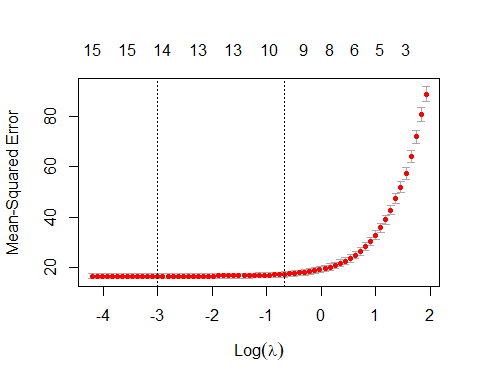
#[1] 18.68608  
  
# Refit the ridge regression model on the full data set using the value of ƛ  
# chosen by cross-validation  
out2 <- glmnet(x2, y2, alpha = 0)  
predict(out2, type = "coefficients", s = bestlambda2)[1:14,]

## (Intercept) year statusDeveloped   
## 5.821438e+01 -2.646849e-03 1.623178e+00   
## adult\_mortality infant\_deaths alcohol   
## -1.945724e-02 -5.222985e-04 2.206080e-02   
## percentage\_expenditure measles bmi   
## 2.647812e-04 -2.723483e-05 4.680687e-02   
## polio total\_expenditure diphtheria   
## 3.150963e-02 5.571625e-02 4.046722e-02   
## hiv\_aids thinness\_5\_9\_years   
## -4.579525e-01 -6.806821e-02

# Fit the lasso model  
lasso\_mod2 <- glmnet(x2[train2, ], y2[train2], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod2)



# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 1)  
plot(cv\_out2)



bestlambda2 <- cv\_out2$lambda.min  
lasso\_pred2 <- predict(lasso\_mod2, s = bestlambda2, newx = x2[test2, ])  
mean((lasso\_pred2 - y\_test2) ^ 2)

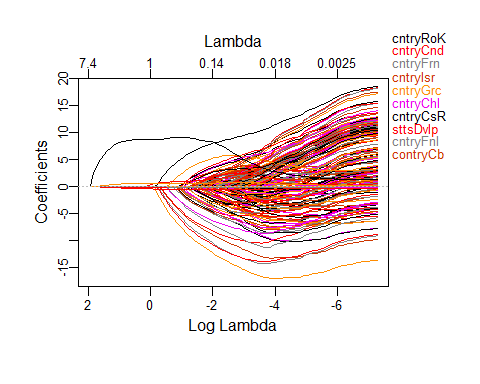
## [1] 18.74681

#[1] 18.74681  
  
# Compute lasso coefficients  
out2 <- glmnet(x2, y2, alpha = 1)  
lasso\_coef2 <- predict(out2, type = "coefficients", s = bestlambda2)[1:14,]  
lasso\_coef2[lasso\_coef2 != 0]

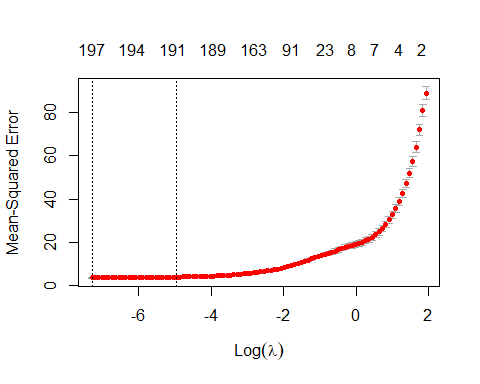
## (Intercept) year statusDeveloped   
## 5.479641e+01 -1.005116e-03 1.580627e+00   
## adult\_mortality infant\_deaths alcohol   
## -2.020382e-02 -2.027380e-04 4.316182e-03   
## percentage\_expenditure measles bmi   
## 2.482666e-04 -2.691257e-05 4.520631e-02   
## polio total\_expenditure diphtheria   
## 2.970608e-02 4.190417e-02 4.077462e-02   
## hiv\_aids thinness\_5\_9\_years   
## -4.719672e-01 -5.973889e-02

The top 4 variables using LASSO on the imputed data set are similar to the non-imputed data set: income\_composition\_of\_resources,statusDeveloped, schooling, and hiv\_aids

# Fit the lasso model with data set including country  
x3 <- model.matrix(life\_expectancy ~ ., df\_predict2)[, -1]  
y3 <- df\_predict2$life\_expectancy  
  
# Split data into training and testing sets  
set.seed(1)  
train3 <- sample(1:nrow(x3), nrow(x3)/2)  
test3 <- (-train3)  
y\_test3 <- y3[test3]  
  
lasso\_mod3 <- glmnet(x3[train3, ], y3[train3], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod3)



# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out3 <- cv.glmnet(x3[train3, ], y3[train3], alpha = 1)  
plot(cv\_out3)



bestlambda3 <- cv\_out3$lambda.min  
lasso\_pred3 <- predict(lasso\_mod3, s = bestlambda3, newx = x3[test3, ])  
mean((lasso\_pred3 - y\_test3) ^ 2)

## [1] 4.841146

#[1] 18.74681  
  
# Compute lasso coefficients  
out3 <- glmnet(x3, y3, alpha = 1)  
lasso\_coef3 <- predict(out3, type = "coefficients", s = bestlambda3)[1:14,]  
lasso\_coef3[lasso\_coef3 != 0]

## (Intercept) countryAlbania   
## -444.327458 12.688404   
## countryAlgeria countryAngola   
## 11.294685 -10.256412   
## countryAntigua and Barbuda countryArgentina   
## 13.405944 12.154597   
## countryArmenia countryAustralia   
## 11.203061 1.827414   
## countryAustria countryAzerbaijan   
## 2.293969 8.746186   
## countryBahamas countryBahrain   
## 12.214649 12.841279   
## countryBangladesh countryBarbados   
## 8.493312 11.756155

Based on these coefficients, it does appear that country is significant when predicting life expectancy. We removed it from our interpretable model as it would have been incredibly difficult to interpret a model with this many coefficients. However, since our goal is only prediction, let’s include country in our final predictive model along with the top 4 variables from LASSO: income\_composition\_of\_resources, statusDeveloped, schooling, and hiv\_aids.

#build a linear model using the top 3 variables from LASSO along with country for both imputed data set and removed NA data set  
  
# Build the final model using the best subset selection results on the imputed data set  
predict\_model1 <-  
 lm(  
 life\_expectancy ~   
 country +  
 income\_composition\_of\_resources +  
 status +  
 schooling +  
 hiv\_aids,  
 data = df\_predict2  
)  
  
# Final model summary  
predict\_model1\_sum <- summary(predict\_model1)  
predict\_model1\_sum

##   
## Call:  
## lm(formula = life\_expectancy ~ country + income\_composition\_of\_resources +   
## status + schooling + hiv\_aids, data = df\_predict2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -24.0639 -0.9750 -0.1676 0.6645 10.8615   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate  
## (Intercept) 52.07682  
## countryAlbania 13.75248  
## countryAlgeria 11.90512  
## countryAngola -8.15676  
## countryAntigua and Barbuda 16.28622  
## countryArgentina 10.85141  
## countryArmenia 12.24175  
## countryAustralia 15.00851  
## countryAustria 17.67152  
## countryAzerbaijan 9.82835  
## countryBahamas 12.63924  
## countryBahrain 12.84428  
## countryBangladesh 10.52951  
## countryBarbados 11.17295  
## countryBelarus 6.81456  
## countryBelgium 15.97391  
## countryBelize 7.86518  
## countryBenin -0.34510  
## countryBhutan 7.49481  
## countryBolivia (Plurinational State of) 5.42331  
## countryBosnia and Herzegovina 15.22741  
## countryBotswana 2.34978  
## countryBrazil 10.73964  
## countryBrunei Darussalam 13.52387  
## countryBulgaria 10.35305  
## countryBurkina Faso 0.46816  
## countryBurundi -0.60689  
## countryCôte d'Ivoire -8.61424  
## countryCabo Verde 11.65918  
## countryCambodia 5.34474  
## countryCameroon -1.96424  
## countryCanada 17.49773  
## countryCentral African Republic -4.20763  
## countryChad -4.30491  
## countryChile 16.10651  
## countryChina 13.42322  
## countryColombia 11.90371  
## countryComoros 2.65527  
## countryCongo 1.62014  
## countryCosta Rica 16.68773  
## countryCroatia 13.42754  
## countryCuba 14.92077  
## countryCyprus 17.08766  
## countryCzechia 15.67666  
## countryDemocratic People's Republic of Korea 8.10166  
## countryDemocratic Republic of the Congo -4.62142  
## countryDenmark 14.23972  
## countryDjibouti 6.11870  
## countryDominican Republic 11.23918  
## countryEcuador 12.85980  
## countryEgypt 10.41685  
## countryEl Salvador 10.27251  
## countryEquatorial Guinea -0.98786  
## countryEritrea 5.45041  
## countryEstonia 10.90581  
## countryEthiopia 3.16873  
## countryFiji 6.08134  
## countryFinland 15.70919  
## countryFrance 18.07337  
## countryGabon 4.42483  
## countryGambia 2.33131  
## countryGeorgia 11.86225  
## countryGermany 16.53599  
## countryGhana 2.56657  
## countryGreece 17.12839  
## countryGrenada 11.33046  
## countryGuatemala 12.18851  
## countryGuinea -0.45379  
## countryGuinea-Bissau -0.27600  
## countryGuyana 5.74541  
## countryHaiti 2.60293  
## countryHonduras 13.01140  
## countryHungary 10.33961  
## countryIceland 16.87945  
## countryIndia 5.84575  
## countryIndonesia 6.70754  
## countryIran (Islamic Republic of) 12.12146  
## countryIraq 10.52613  
## countryIreland 14.87405  
## countryIsrael 17.26191  
## countryItaly 18.05751  
## countryJamaica 13.32499  
## countryJapan 18.94060  
## countryJordan 10.85903  
## countryKazakhstan 4.10868  
## countryKenya 1.93634  
## countryKiribati 4.97214  
## countryKuwait 11.25059  
## countryKyrgyzstan 7.80325  
## countryLao People's Democratic Republic 3.33420  
## countryLatvia 10.00356  
## countryLebanon 12.47765  
## countryLesotho -0.59917  
## countryLiberia -0.61501  
## countryLibya 9.37581  
## countryLithuania 8.71491  
## countryLuxembourg 17.99706  
## countryMadagascar 3.94630  
## countryMalawi -1.96462  
## countryMalaysia 11.98459  
## countryMaldives 14.41428  
## countryMali -1.24370  
## countryMalta 17.35446  
## countryMauritania 5.48859  
## countryMauritius 10.42503  
## countryMexico 14.14803  
## countryMicronesia (Federated States of) 8.19707  
## countryMongolia 4.29111  
## countryMontenegro 14.31081  
## countryMorocco 12.22566  
## countryMozambique 0.63124  
## countryMyanmar 5.90736  
## countryNamibia 5.86182  
## countryNepal 6.73987  
## countryNetherlands 16.19905  
## countryNew Zealand 15.32176  
## countryNicaragua 13.02276  
## countryNiger 2.15685  
## countryNigeria -5.03028  
## countryNorway 16.52216  
## countryOman 13.37101  
## countryPakistan 6.98284  
## countryPanama 14.59984  
## countryPapua New Guinea 3.63197  
## countryParaguay 11.85483  
## countryPeru 11.65136  
## countryPhilippines 6.69789  
## countryPoland 12.05995  
## countryPortugal 16.00633  
## countryQatar 14.86535  
## countryRepublic of Korea 19.39541  
## countryRepublic of Moldova 8.88916  
## countryRomania 11.40881  
## countryRussian Federation 5.30728  
## countryRwanda 2.42644  
## countrySaint Lucia 11.99608  
## countrySaint Vincent and the Grenadines 11.66142  
## countrySamoa 11.99246  
## countrySao Tome and Principe 5.80479  
## countrySaudi Arabia 11.16181  
## countrySenegal 5.18839  
## countrySerbia 11.61335  
## countrySeychelles 10.55423  
## countrySierra Leone -11.29847  
## countrySingapore 18.51186  
## countrySlovakia 11.73696  
## countrySlovenia 15.26256  
## countrySolomon Islands 9.08017  
## countrySomalia -7.43729  
## countrySouth Africa 4.23853  
## countrySouth Sudan 2.18263  
## countrySpain 17.69984  
## countrySri Lanka 11.29350  
## countrySudan 4.75135  
## countrySuriname 9.80147  
## countrySwaziland 6.51446  
## countrySweden 18.33021  
## countrySwitzerland 18.38338  
## countrySyrian Arab Republic 10.40581  
## countryTajikistan 6.56099  
## countryThailand 11.59840  
## countryThe former Yugoslav republic of Macedonia 13.26088  
## countryTimor-Leste 4.76093  
## countryTogo -1.39009  
## countryTonga 9.88666  
## countryTrinidad and Tobago 9.66585  
## countryTunisia 11.82800  
## countryTurkey 12.19850  
## countryTurkmenistan 5.99827  
## countryUganda -0.58443  
## countryUkraine 7.20659  
## countryUnited Arab Emirates 13.58131  
## countryUnited Kingdom of Great Britain and Northern Ireland 19.70166  
## countryUnited Republic of Tanzania -1.81647  
## countryUnited States of America 16.97041  
## countryUruguay 12.62761  
## countryUzbekistan 7.27233  
## countryVanuatu 11.88650  
## countryVenezuela (Bolivarian Republic of) 11.53995  
## countryViet Nam 14.00140  
## countryYemen 5.32345  
## countryZambia -0.95815  
## countryZimbabwe 1.79749  
## income\_composition\_of\_resources 2.75868  
## statusDeveloped NA  
## schooling 0.61085  
## hiv\_aids -0.45566  
## Std. Error t value  
## (Intercept) 0.61043 85.312  
## countryAlbania 0.77902 17.653  
## countryAlgeria 0.78164 15.231  
## countryAngola 0.76697 -10.635  
## countryAntigua and Barbuda 0.76626 21.254  
## countryArgentina 0.81686 13.284  
## countryArmenia 0.77721 15.751  
## countryAustralia 0.86702 17.310  
## countryAustria 0.80566 21.934  
## countryAzerbaijan 0.77590 12.667  
## countryBahamas 0.78125 16.178  
## countryBahrain 0.79361 16.185  
## countryBangladesh 0.76679 13.732  
## countryBarbados 0.79946 13.976  
## countryBelarus 0.79785 8.541  
## countryBelgium 0.82052 19.468  
## countryBelize 0.78032 10.079  
## countryBenin 0.76654 -0.450  
## countryBhutan 0.78453 9.553  
## countryBolivia (Plurinational State of) 0.79119 6.855  
## countryBosnia and Herzegovina 0.78160 19.482  
## countryBotswana 0.82714 2.841  
## countryBrazil 0.79241 13.553  
## countryBrunei Darussalam 0.79467 17.018  
## countryBulgaria 0.78944 13.115  
## countryBurkina Faso 0.77215 0.606  
## countryBurundi 0.76780 -0.790  
## countryCôte d'Ivoire 0.78180 -11.018  
## countryCabo Verde 0.77818 14.983  
## countryCambodia 0.76809 6.959  
## countryCameroon 0.77290 -2.541  
## countryCanada 0.81110 21.573  
## countryCentral African Republic 0.78064 -5.390  
## countryChad 0.77145 -5.580  
## countryChile 0.79985 20.137  
## countryChina 0.77479 17.325  
## countryColombia 0.77875 15.286  
## countryComoros 0.77253 3.437  
## countryCongo 0.77334 2.095  
## countryCosta Rica 0.78333 21.304  
## countryCroatia 0.79144 16.966  
## countryCuba 0.79731 18.714  
## countryCyprus 0.79074 21.610  
## countryCzechia 0.77674 20.183  
## countryDemocratic People's Republic of Korea 0.77674 10.430  
## countryDemocratic Republic of the Congo 0.77781 -5.942  
## countryDenmark 0.82565 17.247  
## countryDjibouti 0.77995 7.845  
## countryDominican Republic 0.78248 14.364  
## countryEcuador 0.78360 16.411  
## countryEgypt 0.77645 13.416  
## countryEl Salvador 0.78084 13.156  
## countryEquatorial Guinea 0.77218 -1.279  
## countryEritrea 0.77384 7.043  
## countryEstonia 0.81038 13.458  
## countryEthiopia 0.76808 4.126  
## countryFiji 0.79257 7.673  
## countryFinland 0.82663 19.004  
## countryFrance 0.81077 22.292  
## countryGabon 0.79260 5.583  
## countryGambia 0.76656 3.041  
## countryGeorgia 0.78115 15.186  
## countryGermany 0.81867 20.198  
## countryGhana 0.76841 3.340  
## countryGreece 0.81061 21.130  
## countryGrenada 0.83954 13.496  
## countryGuatemala 0.76901 15.850  
## countryGuinea 0.76708 -0.592  
## countryGuinea-Bissau 0.77333 -0.357  
## countryGuyana 0.77209 7.441  
## countryHaiti 0.76718 3.393  
## countryHonduras 0.77172 16.860  
## countryHungary 0.80190 12.894  
## countryIceland 0.83818 20.138  
## countryIndia 0.76846 7.607  
## countryIndonesia 0.77510 8.654  
## countryIran (Islamic Republic of) 0.78175 15.506  
## countryIraq 0.77061 13.659  
## countryIreland 0.83140 17.890  
## countryIsrael 0.80900 21.337  
## countryItaly 0.81082 22.271  
## countryJamaica 0.78028 17.077  
## countryJapan 0.80258 23.600  
## countryJordan 0.78556 13.823  
## countryKazakhstan 0.79154 5.191  
## countryKenya 0.78207 2.476  
## countryKiribati 0.78194 6.359  
## countryKuwait 0.79034 14.235  
## countryKyrgyzstan 0.77865 10.021  
## countryLao People's Democratic Republic 0.76688 4.348  
## countryLatvia 0.80623 12.408  
## countryLebanon 0.79528 15.690  
## countryLesotho 0.85960 -0.697  
## countryLiberia 0.77050 -0.798  
## countryLibya 0.79826 11.745  
## countryLithuania 0.81206 10.732  
## countryLuxembourg 0.79372 22.674  
## countryMadagascar 0.76674 5.147  
## countryMalawi 0.81825 -2.401  
## countryMalaysia 0.78229 15.320  
## countryMaldives 0.77686 18.555  
## countryMali 0.76857 -1.618  
## countryMalta 0.79515 21.825  
## countryMauritania 0.76794 7.147  
## countryMauritius 0.78747 13.239  
## countryMexico 0.78048 18.127  
## countryMicronesia (Federated States of) 0.76999 10.646  
## countryMongolia 0.78093 5.495  
## countryMontenegro 0.77082 18.566  
## countryMorocco 0.76970 15.884  
## countryMozambique 0.78725 0.802  
## countryMyanmar 0.76640 7.708  
## countryNamibia 0.80850 7.250  
## countryNepal 0.76958 8.758  
## countryNetherlands 0.82400 19.659  
## countryNew Zealand 0.84860 18.055  
## countryNicaragua 0.77249 16.858  
## countryNiger 0.77996 2.765  
## countryNigeria 0.77130 -6.522  
## countryNorway 0.82951 19.918  
## countryOman 0.77959 17.151  
## countryPakistan 0.77105 9.056  
## countryPanama 0.78334 18.638  
## countryPapua New Guinea 0.76618 4.740  
## countryParaguay 0.77798 15.238  
## countryPeru 0.78570 14.829  
## countryPhilippines 0.77496 8.643  
## countryPoland 0.80328 15.013  
## countryPortugal 0.81025 19.755  
## countryQatar 0.78717 18.885  
## countryRepublic of Korea 0.77674 24.970  
## countryRepublic of Moldova 0.77674 11.444  
## countryRomania 0.79107 14.422  
## countryRussian Federation 0.78982 6.720  
## countryRwanda 0.77048 3.149  
## countrySaint Lucia 0.78121 15.356  
## countrySaint Vincent and the Grenadines 0.78449 14.865  
## countrySamoa 0.78096 15.356  
## countrySao Tome and Principe 0.76920 7.547  
## countrySaudi Arabia 0.78736 14.176  
## countrySenegal 0.76731 6.762  
## countrySerbia 0.78780 14.742  
## countrySeychelles 0.78276 13.483  
## countrySierra Leone 0.76645 -14.741  
## countrySingapore 0.79508 23.283  
## countrySlovakia 0.79527 14.758  
## countrySlovenia 0.81671 18.688  
## countrySolomon Islands 0.76611 11.852  
## countrySomalia 0.77708 -9.571  
## countrySouth Africa 0.84401 5.022  
## countrySouth Sudan 0.79961 2.730  
## countrySpain 0.81526 21.711  
## countrySri Lanka 0.78536 14.380  
## countrySudan 0.77018 6.169  
## countrySuriname 0.77762 12.604  
## countrySwaziland 0.94054 6.926  
## countrySweden 0.81107 22.600  
## countrySwitzerland 0.80741 22.768  
## countrySyrian Arab Republic 0.77225 13.475  
## countryTajikistan 0.77075 8.513  
## countryThailand 0.78083 14.854  
## countryThe former Yugoslav republic of Macedonia 0.78266 16.943  
## countryTimor-Leste 0.77047 6.179  
## countryTogo 0.77446 -1.795  
## countryTonga 0.79291 12.469  
## countryTrinidad and Tobago 0.78165 12.366  
## countryTunisia 0.79147 14.944  
## countryTurkey 0.78158 15.607  
## countryTurkmenistan 0.77930 7.697  
## countryUganda 0.78212 -0.747  
## countryUkraine 0.79672 9.045  
## countryUnited Arab Emirates 0.78654 17.267  
## countryUnited Kingdom of Great Britain and Northern Ireland 0.77674 25.364  
## countryUnited Republic of Tanzania 0.78774 -2.306  
## countryUnited States of America 0.77674 21.848  
## countryUruguay 0.80264 15.733  
## countryUzbekistan 0.77478 9.386  
## countryVanuatu 0.77376 15.362  
## countryVenezuela (Bolivarian Republic of) 0.78277 14.742  
## countryViet Nam 0.77441 18.080  
## countryYemen 0.76606 6.949  
## countryZambia 0.79935 -1.199  
## countryZimbabwe 0.85766 2.096  
## income\_composition\_of\_resources 0.47525 5.805  
## statusDeveloped NA NA  
## schooling 0.04126 14.805  
## hiv\_aids 0.01622 -28.096  
## Pr(>|t|)   
## (Intercept) < 2e-16 \*\*\*  
## countryAlbania < 2e-16 \*\*\*  
## countryAlgeria < 2e-16 \*\*\*  
## countryAngola < 2e-16 \*\*\*  
## countryAntigua and Barbuda < 2e-16 \*\*\*  
## countryArgentina < 2e-16 \*\*\*  
## countryArmenia < 2e-16 \*\*\*  
## countryAustralia < 2e-16 \*\*\*  
## countryAustria < 2e-16 \*\*\*  
## countryAzerbaijan < 2e-16 \*\*\*  
## countryBahamas < 2e-16 \*\*\*  
## countryBahrain < 2e-16 \*\*\*  
## countryBangladesh < 2e-16 \*\*\*  
## countryBarbados < 2e-16 \*\*\*  
## countryBelarus < 2e-16 \*\*\*  
## countryBelgium < 2e-16 \*\*\*  
## countryBelize < 2e-16 \*\*\*  
## countryBenin 0.652598   
## countryBhutan < 2e-16 \*\*\*  
## countryBolivia (Plurinational State of) 8.80e-12 \*\*\*  
## countryBosnia and Herzegovina < 2e-16 \*\*\*  
## countryBotswana 0.004532 \*\*   
## countryBrazil < 2e-16 \*\*\*  
## countryBrunei Darussalam < 2e-16 \*\*\*  
## countryBulgaria < 2e-16 \*\*\*  
## countryBurkina Faso 0.544359   
## countryBurundi 0.429347   
## countryCôte d'Ivoire < 2e-16 \*\*\*  
## countryCabo Verde < 2e-16 \*\*\*  
## countryCambodia 4.28e-12 \*\*\*  
## countryCameroon 0.011096 \*   
## countryCanada < 2e-16 \*\*\*  
## countryCentral African Republic 7.65e-08 \*\*\*  
## countryChad 2.64e-08 \*\*\*  
## countryChile < 2e-16 \*\*\*  
## countryChina < 2e-16 \*\*\*  
## countryColombia < 2e-16 \*\*\*  
## countryComoros 0.000597 \*\*\*  
## countryCongo 0.036263 \*   
## countryCosta Rica < 2e-16 \*\*\*  
## countryCroatia < 2e-16 \*\*\*  
## countryCuba < 2e-16 \*\*\*  
## countryCyprus < 2e-16 \*\*\*  
## countryCzechia < 2e-16 \*\*\*  
## countryDemocratic People's Republic of Korea < 2e-16 \*\*\*  
## countryDemocratic Republic of the Congo 3.18e-09 \*\*\*  
## countryDenmark < 2e-16 \*\*\*  
## countryDjibouti 6.15e-15 \*\*\*  
## countryDominican Republic < 2e-16 \*\*\*  
## countryEcuador < 2e-16 \*\*\*  
## countryEgypt < 2e-16 \*\*\*  
## countryEl Salvador < 2e-16 \*\*\*  
## countryEquatorial Guinea 0.200895   
## countryEritrea 2.36e-12 \*\*\*  
## countryEstonia < 2e-16 \*\*\*  
## countryEthiopia 3.81e-05 \*\*\*  
## countryFiji 2.32e-14 \*\*\*  
## countryFinland < 2e-16 \*\*\*  
## countryFrance < 2e-16 \*\*\*  
## countryGabon 2.60e-08 \*\*\*  
## countryGambia 0.002378 \*\*   
## countryGeorgia < 2e-16 \*\*\*  
## countryGermany < 2e-16 \*\*\*  
## countryGhana 0.000849 \*\*\*  
## countryGreece < 2e-16 \*\*\*  
## countryGrenada < 2e-16 \*\*\*  
## countryGuatemala < 2e-16 \*\*\*  
## countryGuinea 0.554176   
## countryGuinea-Bissau 0.721197   
## countryGuyana 1.33e-13 \*\*\*  
## countryHaiti 0.000702 \*\*\*  
## countryHonduras < 2e-16 \*\*\*  
## countryHungary < 2e-16 \*\*\*  
## countryIceland < 2e-16 \*\*\*  
## countryIndia 3.83e-14 \*\*\*  
## countryIndonesia < 2e-16 \*\*\*  
## countryIran (Islamic Republic of) < 2e-16 \*\*\*  
## countryIraq < 2e-16 \*\*\*  
## countryIreland < 2e-16 \*\*\*  
## countryIsrael < 2e-16 \*\*\*  
## countryItaly < 2e-16 \*\*\*  
## countryJamaica < 2e-16 \*\*\*  
## countryJapan < 2e-16 \*\*\*  
## countryJordan < 2e-16 \*\*\*  
## countryKazakhstan 2.25e-07 \*\*\*  
## countryKenya 0.013350 \*   
## countryKiribati 2.37e-10 \*\*\*  
## countryKuwait < 2e-16 \*\*\*  
## countryKyrgyzstan < 2e-16 \*\*\*  
## countryLao People's Democratic Republic 1.43e-05 \*\*\*  
## countryLatvia < 2e-16 \*\*\*  
## countryLebanon < 2e-16 \*\*\*  
## countryLesotho 0.485839   
## countryLiberia 0.424831   
## countryLibya < 2e-16 \*\*\*  
## countryLithuania < 2e-16 \*\*\*  
## countryLuxembourg < 2e-16 \*\*\*  
## countryMadagascar 2.84e-07 \*\*\*  
## countryMalawi 0.016417 \*   
## countryMalaysia < 2e-16 \*\*\*  
## countryMaldives < 2e-16 \*\*\*  
## countryMali 0.105732   
## countryMalta < 2e-16 \*\*\*  
## countryMauritania 1.13e-12 \*\*\*  
## countryMauritius < 2e-16 \*\*\*  
## countryMexico < 2e-16 \*\*\*  
## countryMicronesia (Federated States of) < 2e-16 \*\*\*  
## countryMongolia 4.27e-08 \*\*\*  
## countryMontenegro < 2e-16 \*\*\*  
## countryMorocco < 2e-16 \*\*\*  
## countryMozambique 0.422720   
## countryMyanmar 1.77e-14 \*\*\*  
## countryNamibia 5.39e-13 \*\*\*  
## countryNepal < 2e-16 \*\*\*  
## countryNetherlands < 2e-16 \*\*\*  
## countryNew Zealand < 2e-16 \*\*\*  
## countryNicaragua < 2e-16 \*\*\*  
## countryNiger 0.005724 \*\*   
## countryNigeria 8.24e-11 \*\*\*  
## countryNorway < 2e-16 \*\*\*  
## countryOman < 2e-16 \*\*\*  
## countryPakistan < 2e-16 \*\*\*  
## countryPanama < 2e-16 \*\*\*  
## countryPapua New Guinea 2.24e-06 \*\*\*  
## countryParaguay < 2e-16 \*\*\*  
## countryPeru < 2e-16 \*\*\*  
## countryPhilippines < 2e-16 \*\*\*  
## countryPoland < 2e-16 \*\*\*  
## countryPortugal < 2e-16 \*\*\*  
## countryQatar < 2e-16 \*\*\*  
## countryRepublic of Korea < 2e-16 \*\*\*  
## countryRepublic of Moldova < 2e-16 \*\*\*  
## countryRomania < 2e-16 \*\*\*  
## countryRussian Federation 2.21e-11 \*\*\*  
## countryRwanda 0.001655 \*\*   
## countrySaint Lucia < 2e-16 \*\*\*  
## countrySaint Vincent and the Grenadines < 2e-16 \*\*\*  
## countrySamoa < 2e-16 \*\*\*  
## countrySao Tome and Principe 6.05e-14 \*\*\*  
## countrySaudi Arabia < 2e-16 \*\*\*  
## countrySenegal 1.66e-11 \*\*\*  
## countrySerbia < 2e-16 \*\*\*  
## countrySeychelles < 2e-16 \*\*\*  
## countrySierra Leone < 2e-16 \*\*\*  
## countrySingapore < 2e-16 \*\*\*  
## countrySlovakia < 2e-16 \*\*\*  
## countrySlovenia < 2e-16 \*\*\*  
## countrySolomon Islands < 2e-16 \*\*\*  
## countrySomalia < 2e-16 \*\*\*  
## countrySouth Africa 5.44e-07 \*\*\*  
## countrySouth Sudan 0.006381 \*\*   
## countrySpain < 2e-16 \*\*\*  
## countrySri Lanka < 2e-16 \*\*\*  
## countrySudan 7.88e-10 \*\*\*  
## countrySuriname < 2e-16 \*\*\*  
## countrySwaziland 5.36e-12 \*\*\*  
## countrySweden < 2e-16 \*\*\*  
## countrySwitzerland < 2e-16 \*\*\*  
## countrySyrian Arab Republic < 2e-16 \*\*\*  
## countryTajikistan < 2e-16 \*\*\*  
## countryThailand < 2e-16 \*\*\*  
## countryThe former Yugoslav republic of Macedonia < 2e-16 \*\*\*  
## countryTimor-Leste 7.40e-10 \*\*\*  
## countryTogo 0.072778 .   
## countryTonga < 2e-16 \*\*\*  
## countryTrinidad and Tobago < 2e-16 \*\*\*  
## countryTunisia < 2e-16 \*\*\*  
## countryTurkey < 2e-16 \*\*\*  
## countryTurkmenistan 1.93e-14 \*\*\*  
## countryUganda 0.454979   
## countryUkraine < 2e-16 \*\*\*  
## countryUnited Arab Emirates < 2e-16 \*\*\*  
## countryUnited Kingdom of Great Britain and Northern Ireland < 2e-16 \*\*\*  
## countryUnited Republic of Tanzania 0.021189 \*   
## countryUnited States of America < 2e-16 \*\*\*  
## countryUruguay < 2e-16 \*\*\*  
## countryUzbekistan < 2e-16 \*\*\*  
## countryVanuatu < 2e-16 \*\*\*  
## countryVenezuela (Bolivarian Republic of) < 2e-16 \*\*\*  
## countryViet Nam < 2e-16 \*\*\*  
## countryYemen 4.58e-12 \*\*\*  
## countryZambia 0.230760   
## countryZimbabwe 0.036190 \*   
## income\_composition\_of\_resources 7.19e-09 \*\*\*  
## statusDeveloped NA   
## schooling < 2e-16 \*\*\*  
## hiv\_aids < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.166 on 2742 degrees of freedom  
## Multiple R-squared: 0.9516, Adjusted R-squared: 0.9483   
## F-statistic: 291.2 on 185 and 2742 DF, p-value: < 2.2e-16

# Get MSE  
mean(predict\_model1\_sum$residuals ^ 2)

## [1] 4.39226

#[1] 4.39226  
  
# Build the final model using the best subset selection results on removed NA data set  
predict\_model2 <-  
 lm(  
 life\_expectancy ~   
 country +  
 income\_composition\_of\_resources +  
 status +  
 schooling +  
 hiv\_aids,  
 data = df\_predict  
)  
  
# Final model summary  
predict\_model2\_sum <- summary(predict\_model2)  
predict\_model2\_sum

##   
## Call:  
## lm(formula = life\_expectancy ~ country + income\_composition\_of\_resources +   
## status + schooling + hiv\_aids, data = df\_predict)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -24.0690 -0.9110 -0.1506 0.6026 10.7040   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value  
## (Intercept) 50.97423 0.61296 83.160  
## countryAlbania 13.20822 0.76418 17.284  
## countryAlgeria 11.29213 0.76758 14.711  
## countryAngola -8.18463 0.74917 -10.925  
## countryAntigua and Barbuda 16.19334 0.74855 21.633  
## countryArgentina 9.74743 0.81048 12.027  
## countryArmenia 11.74323 0.76190 15.413  
## countryAustralia 13.44155 0.87063 15.439  
## countryAustria 16.69388 0.79703 20.945  
## countryAzerbaijan 9.38347 0.76007 12.345  
## countryBahamas 12.05355 0.76687 15.718  
## countryBahrain 12.03564 0.78220 15.387  
## countryBangladesh 10.45022 0.74905 13.951  
## countryBarbados 10.27136 0.78949 13.010  
## countryBelarus 5.93881 0.78746 7.542  
## countryBelgium 14.81911 0.81521 18.178  
## countryBelize 7.27831 0.76592 9.503  
## countryBenin -0.45758 0.74883 -0.611  
## countryBhutan 7.28352 0.76681 9.498  
## countryBolivia (Plurinational State of) 4.66529 0.77894 5.989  
## countryBosnia and Herzegovina 14.72136 0.76620 19.213  
## countryBotswana 1.52564 0.81324 1.876  
## countryBrazil 9.94328 0.78079 12.735  
## countryBrunei Darussalam 12.71006 0.78334 16.226  
## countryBulgaria 9.60273 0.77717 12.356  
## countryBurkina Faso 0.82139 0.75574 1.087  
## countryBurundi -0.55296 0.75007 -0.737  
## countryCabo Verde 11.12586 0.76315 14.579  
## countryCambodia 5.10671 0.75082 6.802  
## countryCameroon -2.17068 0.75530 -2.874  
## countryCanada 16.45378 0.80367 20.473  
## countryCentral African Republic -4.10616 0.76312 -5.381  
## countryChad -4.09782 0.75425 -5.433  
## countryChile 15.20110 0.78997 19.243  
## countryChina 12.97548 0.75901 17.095  
## countryColombia 11.35307 0.76398 14.860  
## countryComoros 2.45214 0.75503 3.248  
## countryCongo 1.27764 0.75641 1.689  
## countryCosta Rica 16.05122 0.76955 20.858  
## countryCroatia 12.65047 0.77957 16.228  
## countryCuba 14.05224 0.78681 17.860  
## countryCyprus 16.34817 0.77830 21.005  
## countryDenmark 13.03016 0.82145 15.862  
## countryDjibouti 6.49196 0.76363 8.501  
## countryDominican Republic 10.60048 0.76857 13.792  
## countryEcuador 12.20703 0.77002 15.853  
## countryEgypt 9.91471 0.76118 13.025  
## countryEl Salvador 9.67156 0.76661 12.616  
## countryEquatorial Guinea -1.11517 0.75439 -1.478  
## countryEritrea 5.86991 0.75789 7.745  
## countryEstonia 9.86711 0.80283 12.290  
## countryEthiopia 3.31051 0.75056 4.411  
## countryFiji 5.28414 0.78095 6.766  
## countryFinland 14.49229 0.82255 17.619  
## countryFrance 17.03039 0.80332 21.200  
## countryGabon 3.71499 0.77868 4.771  
## countryGambia 2.30101 0.74876 3.073  
## countryGeorgia 11.25713 0.76699 14.677  
## countryGermany 15.40135 0.81298 18.944  
## countryGhana 2.34457 0.75102 3.122  
## countryGreece 16.08610 0.80314 20.029  
## countryGrenada 10.49443 0.82707 12.689  
## countryGuatemala 11.94621 0.75179 15.890  
## countryGuinea -0.35267 0.74943 -0.471  
## countryGuinea-Bissau -0.33956 0.75544 -0.449  
## countryGuyana 5.36894 0.75565 7.105  
## countryHaiti 2.51718 0.74942 3.359  
## countryHonduras 12.63248 0.75533 16.725  
## countryHungary 9.40641 0.79249 11.869  
## countryIceland 15.55285 0.83640 18.595  
## countryIndia 5.59741 0.75129 7.450  
## countryIndonesia 6.23978 0.75950 8.216  
## countryIran (Islamic Republic of) 11.51166 0.76765 14.996  
## countryIraq 10.27404 0.75344 13.636  
## countryIreland 13.60852 0.82834 16.429  
## countryIsrael 16.24178 0.80113 20.274  
## countryItaly 17.01301 0.80340 21.176  
## countryJamaica 12.74324 0.76574 16.642  
## countryJapan 18.01162 0.79310 22.710  
## countryJordan 10.17424 0.77240 13.172  
## countryKazakhstan 3.32283 0.77979 4.261  
## countryKenya 1.54673 0.76514 2.021  
## countryKiribati 4.54861 0.76571 5.940  
## countryKuwait 10.49367 0.77817 13.485  
## countryKyrgyzstan 7.25049 0.76389 9.491  
## countryLao People's Democratic Republic 3.18664 0.74930 4.253  
## countryLatvia 9.01659 0.79773 11.303  
## countryLebanon 11.77891 0.78198 15.063  
## countryLesotho -1.32181 0.84353 -1.567  
## countryLiberia -0.85542 0.75314 -1.136  
## countryLibya 8.49508 0.78795 10.781  
## countryLithuania 7.65912 0.80481 9.517  
## countryLuxembourg 17.23426 0.78158 22.050  
## countryMadagascar 3.79160 0.74917 5.061  
## countryMalawi -2.51272 0.80158 -3.135  
## countryMalaysia 11.38000 0.76812 14.815  
## countryMaldives 13.90061 0.76170 18.249  
## countryMali -1.04347 0.75123 -1.389  
## countryMalta 16.52109 0.78412 21.070  
## countryMauritania 5.56939 0.75019 7.424  
## countryMauritius 9.70477 0.77478 12.526  
## countryMexico 13.57684 0.76593 17.726  
## countryMicronesia (Federated States of) 7.89236 0.75313 10.479  
## countryMongolia 3.68978 0.76672 4.812  
## countryMontenegro 11.18125 0.88679 12.609  
## countryMorocco 11.93520 0.75276 15.855  
## countryMozambique 0.46689 0.76937 0.607  
## countryMyanmar 5.87284 0.74860 7.845  
## countryNamibia 5.17050 0.79356 6.516  
## countryNepal 6.44775 0.75264 8.567  
## countryNetherlands 15.00664 0.81945 18.313  
## countryNew Zealand 13.90536 0.84877 16.383  
## countryNicaragua 12.62397 0.75629 16.692  
## countryNiger 2.68195 0.76495 3.506  
## countryNigeria -5.20417 0.75363 -6.906  
## countryNorway 15.27241 0.82616 18.486  
## countryOman 12.81480 0.76488 16.754  
## countryPakistan 7.15259 0.75345 9.493  
## countryPanama 13.96852 0.76950 18.153  
## countryPapua New Guinea 3.56193 0.74842 4.759  
## countryParaguay 11.31617 0.76308 14.830  
## countryPeru 10.95940 0.77260 14.185  
## countryPhilippines 6.23878 0.75929 8.217  
## countryPoland 11.10873 0.79418 13.988  
## countryPortugal 14.97109 0.80262 18.653  
## countryQatar 14.20998 0.77359 18.369  
## countryRomania 10.63208 0.77919 13.645  
## countryRussian Federation 4.55028 0.77763 5.851  
## countryRwanda 2.21454 0.75295 2.941  
## countrySaint Lucia 11.38867 0.76707 14.847  
## countrySaint Vincent and the Grenadines 10.99395 0.77106 14.258  
## countrySamoa 11.39114 0.76677 14.856  
## countrySao Tome and Principe 5.51346 0.75222 7.330  
## countrySaudi Arabia 10.46064 0.77441 13.508  
## countrySenegal 5.31275 0.74966 7.087  
## countrySerbia 10.88959 0.77516 14.048  
## countrySeychelles 9.91789 0.76898 12.897  
## countrySierra Leone -11.31615 0.74865 -15.115  
## countrySingapore 17.70841 0.78357 22.600  
## countrySlovakia 10.89963 0.78431 13.897  
## countrySlovenia 14.14993 0.81057 17.457  
## countrySolomon Islands 9.01499 0.74836 12.046  
## countrySouth Africa 3.28516 0.83138 3.951  
## countrySpain 16.60386 0.80879 20.529  
## countrySri Lanka 10.61177 0.77216 13.743  
## countrySuriname 9.32750 0.76193 12.242  
## countrySwaziland 5.63969 0.92419 6.102  
## countrySweden 17.28686 0.80363 21.511  
## countrySwitzerland 17.39576 0.79895 21.773  
## countrySyrian Arab Republic 10.02283 0.75593 13.259  
## countryTajikistan 6.21904 0.75412 8.247  
## countryThailand 11.00051 0.76656 14.351  
## countryThe former Yugoslav republic of Macedonia 12.73437 0.76746 16.593  
## countryTimor-Leste 4.51887 0.78416 5.763  
## countryTogo -1.76682 0.75777 -2.332  
## countryTonga 9.08545 0.78135 11.628  
## countryTrinidad and Tobago 9.07981 0.76720 11.835  
## countryTunisia 11.04818 0.77960 14.172  
## countryTurkey 11.58858 0.76749 15.099  
## countryTurkmenistan 5.83534 0.76151 7.663  
## countryUganda -1.02205 0.76555 -1.335  
## countryUkraine 6.34593 0.78601 8.074  
## countryUnited Arab Emirates 12.93228 0.77288 16.733  
## countryUruguay 11.68773 0.79331 14.733  
## countryUzbekistan 6.80784 0.75913 8.968  
## countryVanuatu 11.60229 0.75666 15.333  
## countryVenezuela (Bolivarian Republic of) 10.91158 0.76890 14.191  
## countryViet Nam 13.54982 0.75867 17.860  
## countryYemen 5.27551 0.74829 7.050  
## countryZambia -1.55971 0.78370 -1.990  
## countryZimbabwe 1.17639 0.84076 1.399  
## income\_composition\_of\_resources 2.94789 0.46601 6.326  
## statusDeveloped NA NA NA  
## schooling 0.73532 0.04336 16.958  
## hiv\_aids -0.43771 0.01602 -27.319  
## Pr(>|t|)   
## (Intercept) < 2e-16 \*\*\*  
## countryAlbania < 2e-16 \*\*\*  
## countryAlgeria < 2e-16 \*\*\*  
## countryAngola < 2e-16 \*\*\*  
## countryAntigua and Barbuda < 2e-16 \*\*\*  
## countryArgentina < 2e-16 \*\*\*  
## countryArmenia < 2e-16 \*\*\*  
## countryAustralia < 2e-16 \*\*\*  
## countryAustria < 2e-16 \*\*\*  
## countryAzerbaijan < 2e-16 \*\*\*  
## countryBahamas < 2e-16 \*\*\*  
## countryBahrain < 2e-16 \*\*\*  
## countryBangladesh < 2e-16 \*\*\*  
## countryBarbados < 2e-16 \*\*\*  
## countryBelarus 6.40e-14 \*\*\*  
## countryBelgium < 2e-16 \*\*\*  
## countryBelize < 2e-16 \*\*\*  
## countryBenin 0.541215   
## countryBhutan < 2e-16 \*\*\*  
## countryBolivia (Plurinational State of) 2.40e-09 \*\*\*  
## countryBosnia and Herzegovina < 2e-16 \*\*\*  
## countryBotswana 0.060770 .   
## countryBrazil < 2e-16 \*\*\*  
## countryBrunei Darussalam < 2e-16 \*\*\*  
## countryBulgaria < 2e-16 \*\*\*  
## countryBurkina Faso 0.277196   
## countryBurundi 0.461058   
## countryCabo Verde < 2e-16 \*\*\*  
## countryCambodia 1.28e-11 \*\*\*  
## countryCameroon 0.004087 \*\*   
## countryCanada < 2e-16 \*\*\*  
## countryCentral African Republic 8.09e-08 \*\*\*  
## countryChad 6.07e-08 \*\*\*  
## countryChile < 2e-16 \*\*\*  
## countryChina < 2e-16 \*\*\*  
## countryColombia < 2e-16 \*\*\*  
## countryComoros 0.001178 \*\*   
## countryCongo 0.091325 .   
## countryCosta Rica < 2e-16 \*\*\*  
## countryCroatia < 2e-16 \*\*\*  
## countryCuba < 2e-16 \*\*\*  
## countryCyprus < 2e-16 \*\*\*  
## countryDenmark < 2e-16 \*\*\*  
## countryDjibouti < 2e-16 \*\*\*  
## countryDominican Republic < 2e-16 \*\*\*  
## countryEcuador < 2e-16 \*\*\*  
## countryEgypt < 2e-16 \*\*\*  
## countryEl Salvador < 2e-16 \*\*\*  
## countryEquatorial Guinea 0.139468   
## countryEritrea 1.37e-14 \*\*\*  
## countryEstonia < 2e-16 \*\*\*  
## countryEthiopia 1.07e-05 \*\*\*  
## countryFiji 1.63e-11 \*\*\*  
## countryFinland < 2e-16 \*\*\*  
## countryFrance < 2e-16 \*\*\*  
## countryGabon 1.94e-06 \*\*\*  
## countryGambia 0.002141 \*\*   
## countryGeorgia < 2e-16 \*\*\*  
## countryGermany < 2e-16 \*\*\*  
## countryGhana 0.001817 \*\*   
## countryGreece < 2e-16 \*\*\*  
## countryGrenada < 2e-16 \*\*\*  
## countryGuatemala < 2e-16 \*\*\*  
## countryGuinea 0.637973   
## countryGuinea-Bissau 0.653117   
## countryGuyana 1.55e-12 \*\*\*  
## countryHaiti 0.000794 \*\*\*  
## countryHonduras < 2e-16 \*\*\*  
## countryHungary < 2e-16 \*\*\*  
## countryIceland < 2e-16 \*\*\*  
## countryIndia 1.27e-13 \*\*\*  
## countryIndonesia 3.31e-16 \*\*\*  
## countryIran (Islamic Republic of) < 2e-16 \*\*\*  
## countryIraq < 2e-16 \*\*\*  
## countryIreland < 2e-16 \*\*\*  
## countryIsrael < 2e-16 \*\*\*  
## countryItaly < 2e-16 \*\*\*  
## countryJamaica < 2e-16 \*\*\*  
## countryJapan < 2e-16 \*\*\*  
## countryJordan < 2e-16 \*\*\*  
## countryKazakhstan 2.11e-05 \*\*\*  
## countryKenya 0.043333 \*   
## countryKiribati 3.23e-09 \*\*\*  
## countryKuwait < 2e-16 \*\*\*  
## countryKyrgyzstan < 2e-16 \*\*\*  
## countryLao People's Democratic Republic 2.19e-05 \*\*\*  
## countryLatvia < 2e-16 \*\*\*  
## countryLebanon < 2e-16 \*\*\*  
## countryLesotho 0.117241   
## countryLiberia 0.256148   
## countryLibya < 2e-16 \*\*\*  
## countryLithuania < 2e-16 \*\*\*  
## countryLuxembourg < 2e-16 \*\*\*  
## countryMadagascar 4.47e-07 \*\*\*  
## countryMalawi 0.001740 \*\*   
## countryMalaysia < 2e-16 \*\*\*  
## countryMaldives < 2e-16 \*\*\*  
## countryMali 0.164950   
## countryMalta < 2e-16 \*\*\*  
## countryMauritania 1.54e-13 \*\*\*  
## countryMauritius < 2e-16 \*\*\*  
## countryMexico < 2e-16 \*\*\*  
## countryMicronesia (Federated States of) < 2e-16 \*\*\*  
## countryMongolia 1.58e-06 \*\*\*  
## countryMontenegro < 2e-16 \*\*\*  
## countryMorocco < 2e-16 \*\*\*  
## countryMozambique 0.544002   
## countryMyanmar 6.31e-15 \*\*\*  
## countryNamibia 8.69e-11 \*\*\*  
## countryNepal < 2e-16 \*\*\*  
## countryNetherlands < 2e-16 \*\*\*  
## countryNew Zealand < 2e-16 \*\*\*  
## countryNicaragua < 2e-16 \*\*\*  
## countryNiger 0.000463 \*\*\*  
## countryNigeria 6.29e-12 \*\*\*  
## countryNorway < 2e-16 \*\*\*  
## countryOman < 2e-16 \*\*\*  
## countryPakistan < 2e-16 \*\*\*  
## countryPanama < 2e-16 \*\*\*  
## countryPapua New Guinea 2.05e-06 \*\*\*  
## countryParaguay < 2e-16 \*\*\*  
## countryPeru < 2e-16 \*\*\*  
## countryPhilippines 3.29e-16 \*\*\*  
## countryPoland < 2e-16 \*\*\*  
## countryPortugal < 2e-16 \*\*\*  
## countryQatar < 2e-16 \*\*\*  
## countryRomania < 2e-16 \*\*\*  
## countryRussian Federation 5.49e-09 \*\*\*  
## countryRwanda 0.003299 \*\*   
## countrySaint Lucia < 2e-16 \*\*\*  
## countrySaint Vincent and the Grenadines < 2e-16 \*\*\*  
## countrySamoa < 2e-16 \*\*\*  
## countrySao Tome and Principe 3.08e-13 \*\*\*  
## countrySaudi Arabia < 2e-16 \*\*\*  
## countrySenegal 1.77e-12 \*\*\*  
## countrySerbia < 2e-16 \*\*\*  
## countrySeychelles < 2e-16 \*\*\*  
## countrySierra Leone < 2e-16 \*\*\*  
## countrySingapore < 2e-16 \*\*\*  
## countrySlovakia < 2e-16 \*\*\*  
## countrySlovenia < 2e-16 \*\*\*  
## countrySolomon Islands < 2e-16 \*\*\*  
## countrySouth Africa 7.98e-05 \*\*\*  
## countrySpain < 2e-16 \*\*\*  
## countrySri Lanka < 2e-16 \*\*\*  
## countrySuriname < 2e-16 \*\*\*  
## countrySwaziland 1.20e-09 \*\*\*  
## countrySweden < 2e-16 \*\*\*  
## countrySwitzerland < 2e-16 \*\*\*  
## countrySyrian Arab Republic < 2e-16 \*\*\*  
## countryTajikistan 2.57e-16 \*\*\*  
## countryThailand < 2e-16 \*\*\*  
## countryThe former Yugoslav republic of Macedonia < 2e-16 \*\*\*  
## countryTimor-Leste 9.27e-09 \*\*\*  
## countryTogo 0.019799 \*   
## countryTonga < 2e-16 \*\*\*  
## countryTrinidad and Tobago < 2e-16 \*\*\*  
## countryTunisia < 2e-16 \*\*\*  
## countryTurkey < 2e-16 \*\*\*  
## countryTurkmenistan 2.56e-14 \*\*\*  
## countryUganda 0.181976   
## countryUkraine 1.04e-15 \*\*\*  
## countryUnited Arab Emirates < 2e-16 \*\*\*  
## countryUruguay < 2e-16 \*\*\*  
## countryUzbekistan < 2e-16 \*\*\*  
## countryVanuatu < 2e-16 \*\*\*  
## countryVenezuela (Bolivarian Republic of) < 2e-16 \*\*\*  
## countryViet Nam < 2e-16 \*\*\*  
## countryYemen 2.29e-12 \*\*\*  
## countryZambia 0.046676 \*   
## countryZimbabwe 0.161874   
## income\_composition\_of\_resources 2.96e-10 \*\*\*  
## statusDeveloped NA   
## schooling < 2e-16 \*\*\*  
## hiv\_aids < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.115 on 2554 degrees of freedom  
## Multiple R-squared: 0.9518, Adjusted R-squared: 0.9485   
## F-statistic: 291.6 on 173 and 2554 DF, p-value: < 2.2e-16

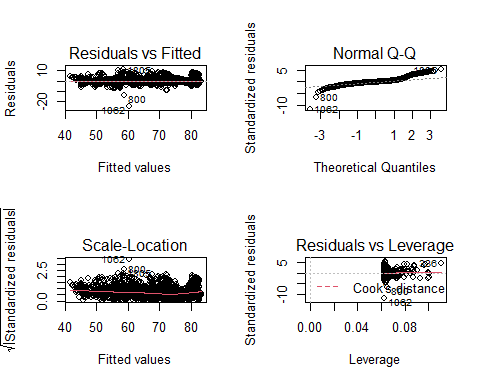
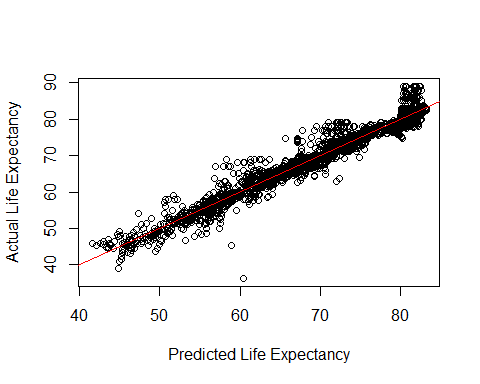
# Get MSE  
mean(predict\_model2\_sum$residuals ^ 2)

## [1] 4.189329

#[1] 4.189329

Make determination for which model to use above (imputed versus remove NA - metrics are not that different). What else can we do to avoid overfit?

## Check residuals just for fun



## KNN Model

We will proceed with the imputed data set we used in our predictive model above.

#copy and rename imputed data set for KNN models  
df\_knn <- df\_predict2  
  
#Make new data set that does not impute values for comparison  
df\_knn2 <- df\_clean  
df\_knn2 <- drop\_na(df\_knn2)  
  
  
# Set seed  
set.seed(123)  
  
# Standardize the data to prep for KNN first - everything except life expectancy  
preProcValues <- preProcess(df\_knn[, -4], method = c("scale"))  
df\_knn\_standard <- predict(preProcValues, df\_knn)  
  
# Split training/test data sets  
inTraining <-  
 createDataPartition(df\_knn\_standard$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train <- df\_knn\_standard[inTraining, ]

## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.  
## Convert to a vector.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

knn\_test <- df\_knn\_standard[-inTraining, ]  
  
# Perform same splits for data with NA's removed  
# Standardize the data to prep for KNN first - everything except life expectancy  
preProcValues\_2 <- preProcess(df\_knn2[, -4], method = c("scale"))  
df\_knn\_standard2 <- predict(preProcValues\_2, df\_knn2)  
  
# Split training/test data sets  
inTraining2 <-  
 createDataPartition(df\_knn\_standard2$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train\_na <- df\_knn\_standard2[inTraining2, ]  
knn\_test\_na <- df\_knn\_standard2[-inTraining2, ]

# Set seed  
set.seed(567)  
  
# Set train control: 5 repeat, 10-fold CV  
ctrl <-  
 trainControl(  
 method = "repeatedcv",  
 number = 10, # 10-fold CV  
 repeats = 5, #repeat 5 times  
 returnResamp = "all" #return all metrics  
 )   
  
# Run everything with the train control above  
knnFit <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10 #run for 10 different k's  
 )  
knnFit

## k-Nearest Neighbors   
##   
## 2198 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1977, 1979, 1977, 1980, 1977, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 2.727066 0.9187597 1.778652  
## 7 2.845302 0.9122791 1.906554  
## 9 2.964521 0.9055666 2.034328  
## 11 3.067507 0.8993824 2.146943  
## 13 3.155973 0.8938308 2.230669  
## 15 3.231355 0.8891469 2.309715  
## 17 3.290674 0.8854589 2.368762  
## 19 3.341312 0.8823133 2.411899  
## 21 3.397990 0.8785676 2.455751  
## 23 3.451031 0.8751930 2.493273  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

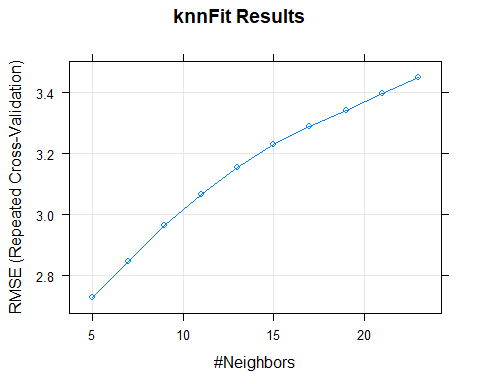
# Check the metrics on the test set  
Predictions\_knn5 <- predict(knnFit, newdata = knn\_test)  
ASE\_knn5 <- mean((Predictions\_knn5 - knn\_test$life\_expectancy)^2)  
ASE\_knn5

## [1] 5.437987

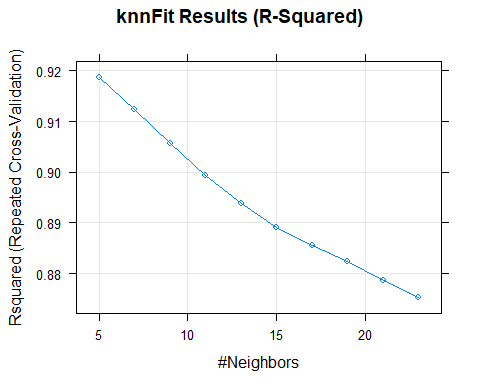
# Performance measurement  
postResample(knn\_test$life\_expectancy, Predictions\_knn5)

## RMSE Rsquared MAE   
## 2.331949 0.943103 1.551721

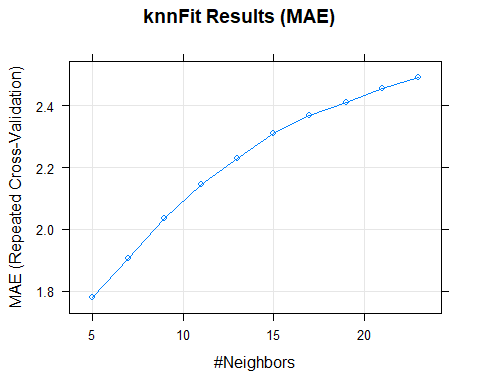
# Plotting  
plot(knnFit, main = "knnFit Results")



plot(knnFit, metric = "Rsquared", main = "knnFit Results (R-Squared)")



plot(knnFit, metric = "MAE", main = "knnFit Results (MAE)")



# Try another knn with a k value of less than 5  
knnFit2 <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneGrid = expand.grid(k = c(1, 3, 5)) #run only with k = 1, 3, 5  
 )  
knnFit2

## k-Nearest Neighbors   
##   
## 2198 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1978, 1977, 1979, 1980, 1978, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 1 2.779697 0.9149756 1.534585  
## 3 2.660422 0.9219317 1.629771  
## 5 2.720828 0.9190582 1.776862  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 3.

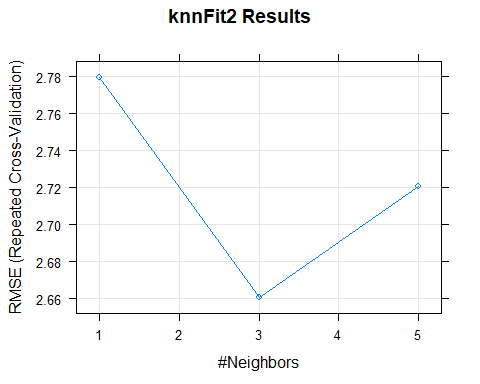
# Check the metrics on the test set  
Predictions\_knn3 <- predict(knnFit2, newdata = knn\_test)  
ASE\_knn3 <- mean((Predictions\_knn3 - knn\_test$life\_expectancy)^2)  
ASE\_knn3

## [1] 5.607542

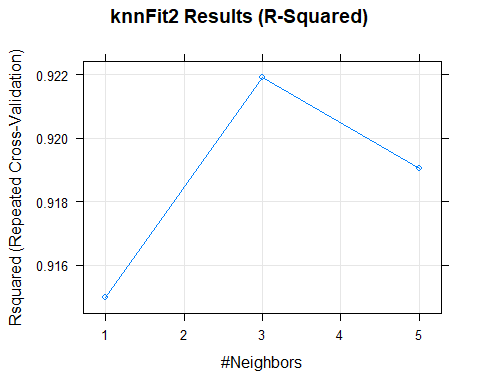
# Performance measurement  
postResample(knn\_test$life\_expectancy, Predictions\_knn3)

## RMSE Rsquared MAE   
## 2.3680249 0.9396708 1.4663470

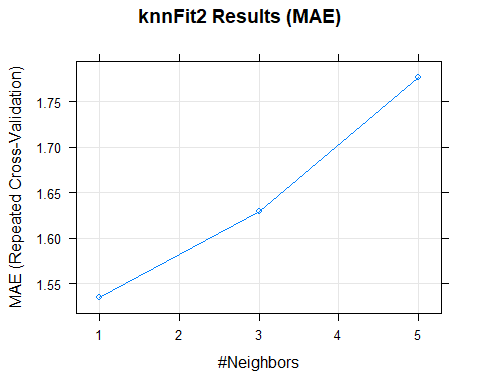
# Plotting  
plot(knnFit2, main = "knnFit2 Results")



plot(knnFit2, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")



plot(knnFit2, metric = "MAE", main = "knnFit2 Results (MAE)")



Based on our training data set, we have surprisingly good results with the imputed data set and all of the predictors. Our RMSE is around 3 and our Rsquared is in the 90’s. Since the Rsquared is so high, there is risk of over-fit. Let’s also run another model using our KNN data set where we just removed the NA values instead of imputing them.

# Run knn on data set with NA's removed  
knnFit\_na <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_na,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10 #run through 10 different k's  
 )  
knnFit\_na

## k-Nearest Neighbors   
##   
## 1565 samples  
## 17 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1409, 1409, 1409, 1409, 1409, 1408, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 2.729131 0.8955882 1.798072  
## 7 2.855972 0.8861574 1.937611  
## 9 2.971693 0.8774921 2.061289  
## 11 3.052713 0.8713865 2.151173  
## 13 3.114649 0.8672837 2.228349  
## 15 3.163130 0.8638952 2.283608  
## 17 3.203184 0.8611497 2.322736  
## 19 3.235289 0.8590118 2.363439  
## 21 3.273323 0.8564276 2.404058  
## 23 3.320029 0.8527544 2.442173  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

# Check the metrics on the test set  
Predictions\_knn\_na <- predict(knnFit\_na, newdata = knn\_test\_na)  
ASE\_knn\_na <- mean((Predictions\_knn\_na - knn\_test\_na$life\_expectancy)^2)  
ASE\_knn\_na

## [1] 6.425467

# Performance measurement  
postResample(knn\_test\_na$life\_expectancy, Predictions\_knn\_na)

## RMSE Rsquared MAE   
## 2.5348506 0.8996341 1.6095689

Even with a smaller data set, the metrics are still pretty good but we do see some decline in Rquared and an increase in RMSE and ASE.

## Variable Selection for KNN

Let’s now test KNN using the same 4 predictors from our first linear model and compare our results with all of the predictors.

knnFit3 <-  
 train(  
 life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit3

## k-Nearest Neighbors   
##   
## 2198 samples  
## 4 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1977, 1979, 1977, 1978, 1979, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 2.561113 0.9274475 1.737455  
## 7 2.648516 0.9226316 1.813755  
## 9 2.703079 0.9199069 1.856946  
## 11 2.753508 0.9173603 1.909651  
## 13 2.812952 0.9141397 1.951714  
## 15 2.877460 0.9106618 1.990644  
## 17 2.943589 0.9069184 2.036066  
## 19 3.012511 0.9028132 2.081130  
## 21 3.074444 0.8991037 2.116807  
## 23 3.120671 0.8964439 2.147088  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

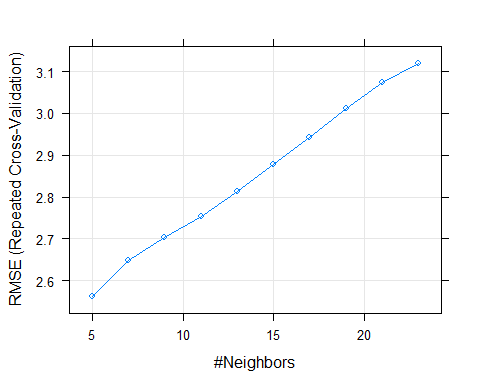
#check metrics on test set  
Predictions\_knnfit3 <- predict(knnFit3,newdata=knn\_test)  
#performance measurement  
postResample(knn\_test$life\_expectancy,Predictions\_knnfit3)

## RMSE Rsquared MAE   
## 2.3026406 0.9428581 1.6348539

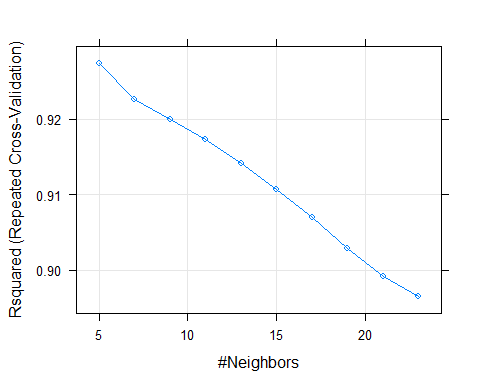
ASE\_knn3 <- mean((Predictions\_knnfit3 - knn\_test$life\_expectancy)^2)  
ASE\_knn3

## [1] 5.302154

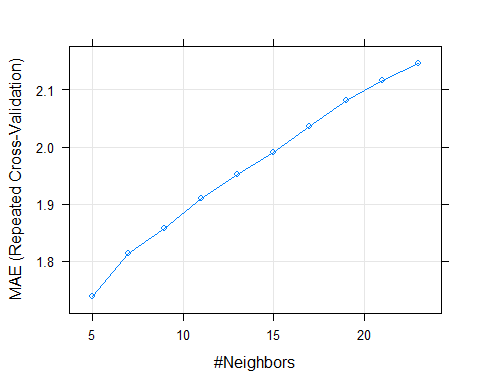
plot(knnFit3)



plot(knnFit3, metric = "Rsquared")



plot(knnFit3, metric = "MAE")



We see that using all of the predictors and then comparing with just the four ones we deemed important from our first model (which was on a much more limited data set) gives us approximately the same adj R2 value and RMSE. When applying to our test data set, it does not appear that our models are over-fitting as we do not see a big shift in our metrics, even though an adj R2 in the 90’s is normally a warning sign for an over-fitting model.

## Check for Over Fit

Let’s create one more train/test split for our KNN models to verify our metrics and confirm there is no over-fitting, as we are suspicious of the high Rsquared values which are typically indicative of overfit.

# Split training/test data sets - round 2  
set.seed(1)  
inTraining\_x <-  
 createDataPartition(df\_knn\_standard$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train\_x <- df\_knn\_standard[inTraining\_x, ]  
knn\_test\_x <- df\_knn\_standard[-inTraining\_x, ]  
  
# Run everything with the train control above  
knnFit\_x <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit\_x

## k-Nearest Neighbors   
##   
## 2198 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1978, 1978, 1978, 1978, 1979, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 2.758352 0.9178010 1.758206  
## 7 2.851920 0.9123058 1.875992  
## 9 2.945481 0.9071972 1.986478  
## 11 3.051277 0.9009312 2.095282  
## 13 3.137708 0.8955635 2.185419  
## 15 3.213433 0.8907567 2.257266  
## 17 3.273867 0.8870681 2.319674  
## 19 3.327547 0.8839221 2.372965  
## 21 3.380709 0.8807016 2.423853  
## 23 3.418812 0.8783644 2.461645  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

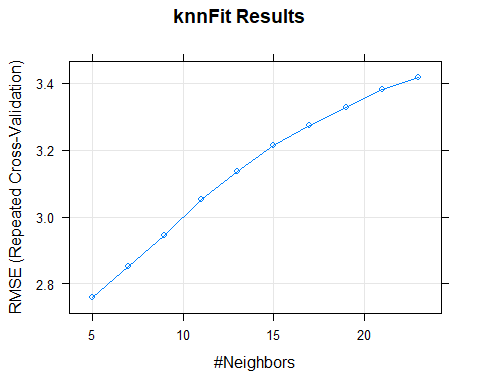
# Check the metrics on the test set  
Predictions\_knnFit\_x <- predict(knnFit\_x, newdata = knn\_test\_x)  
ASE\_x <- mean((Predictions\_knnFit\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_x

## [1] 6.218146

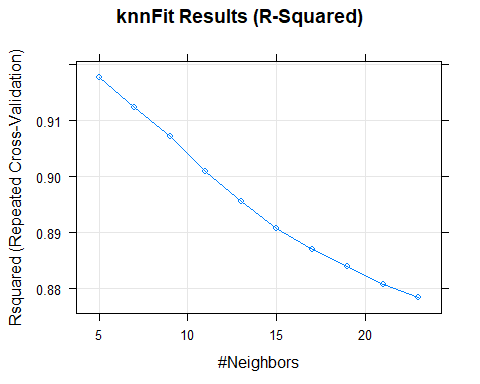
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit\_x)

## RMSE Rsquared MAE   
## 2.4936210 0.9296195 1.6599041

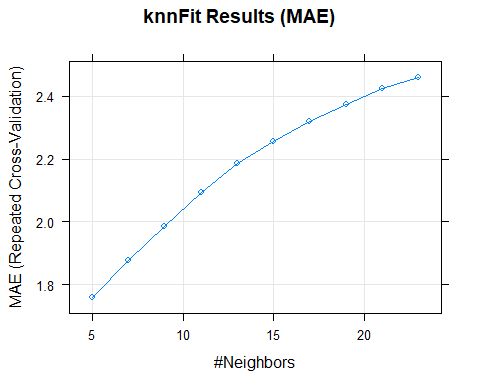
# Plotting  
plot(knnFit\_x, main = "knnFit Results")



plot(knnFit\_x, metric = "Rsquared", main = "knnFit Results (R-Squared)")



plot(knnFit\_x, metric = "MAE", main = "knnFit Results (MAE)")



# Try another knn with a k value of less than 5  
knnFit2\_x <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneGrid = expand.grid(k = c(1, 3, 5))  
 )  
knnFit2\_x

## k-Nearest Neighbors   
##   
## 2198 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1977, 1979, 1979, 1978, 1979, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 1 2.719049 0.9195747 1.492611  
## 3 2.676725 0.9217375 1.609224  
## 5 2.790398 0.9153602 1.769093  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 3.

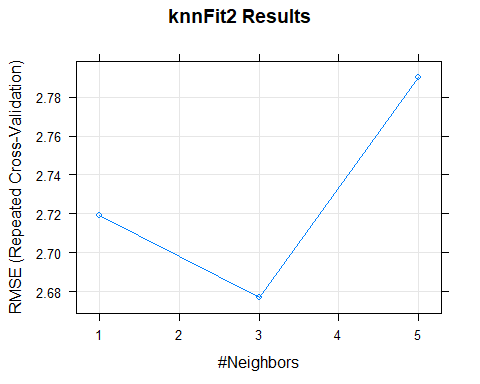
# Check the metrics on the test set  
Predictions\_knnFit2\_x <- predict(knnFit2\_x, newdata = knn\_test\_x)  
ASE\_x <- mean((Predictions\_knnFit2\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_x

## [1] 5.609656

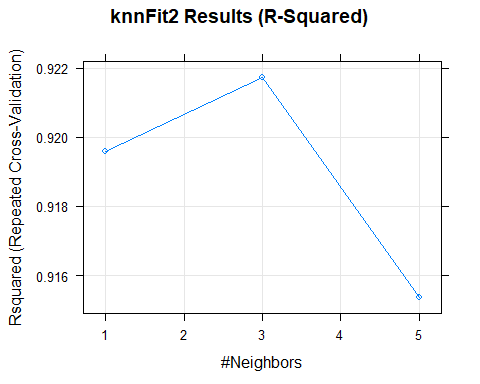
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit2\_x)

## RMSE Rsquared MAE   
## 2.3684712 0.9360244 1.5110502

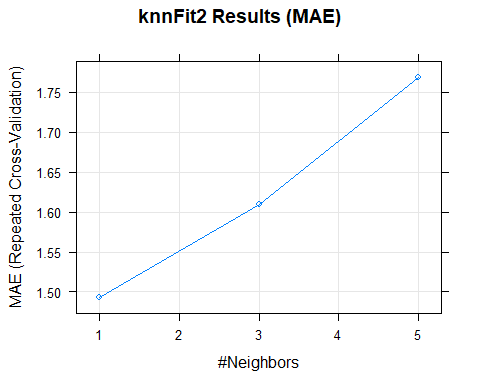
# Plotting  
plot(knnFit2\_x, main = "knnFit2 Results")



plot(knnFit2\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")



plot(knnFit2\_x, metric = "MAE", main = "knnFit2 Results (MAE)")



knnFit3\_x <-  
 train(  
 life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit3\_x

## k-Nearest Neighbors   
##   
## 2198 samples  
## 4 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 5 times)   
## Summary of sample sizes: 1978, 1978, 1978, 1979, 1978, 1978, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 2.553745 0.9287526 1.701945  
## 7 2.582380 0.9275443 1.761309  
## 9 2.609808 0.9264032 1.797818  
## 11 2.670744 0.9232022 1.846329  
## 13 2.742095 0.9192813 1.896727  
## 15 2.803642 0.9159492 1.942352  
## 17 2.884625 0.9113433 1.994559  
## 19 2.951274 0.9075646 2.038573  
## 21 3.010228 0.9042408 2.074177  
## 23 3.057312 0.9016441 2.103769  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

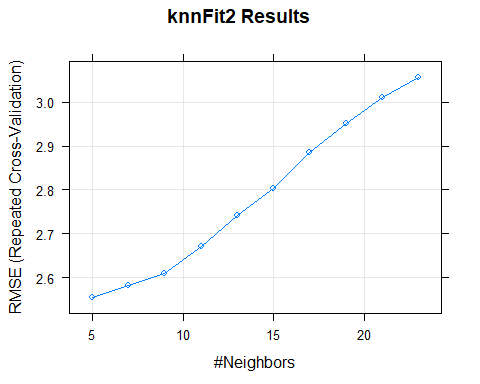
# Check the metrics on the test set  
Predictions\_knnFit3\_x <- predict(knnFit3\_x, newdata = knn\_test\_x)  
ASE\_3x <- mean((Predictions\_knnFit3\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_3x

## [1] 6.595381

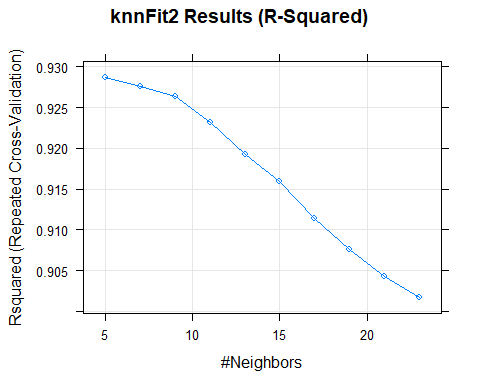
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit3\_x)

## RMSE Rsquared MAE   
## 2.5681474 0.9253835 1.6842420

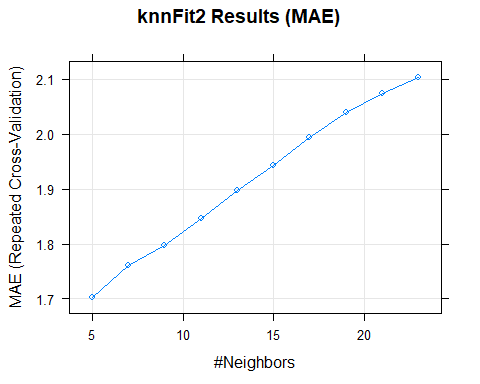
# Plotting  
plot(knnFit3\_x, main = "knnFit2 Results")



plot(knnFit3\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")



plot(knnFit3\_x, metric = "MAE", main = "knnFit2 Results (MAE)")



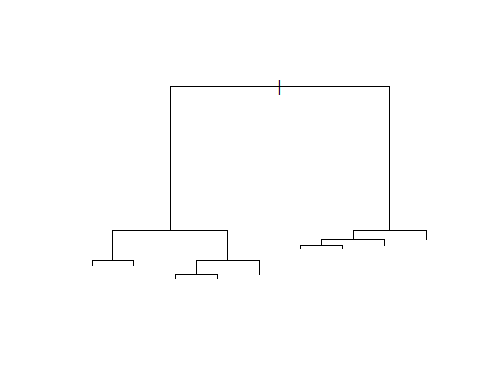
## Trees

Let’s run some tree models to see if we get better metrics that we think may be less susceptible to over-fitting.

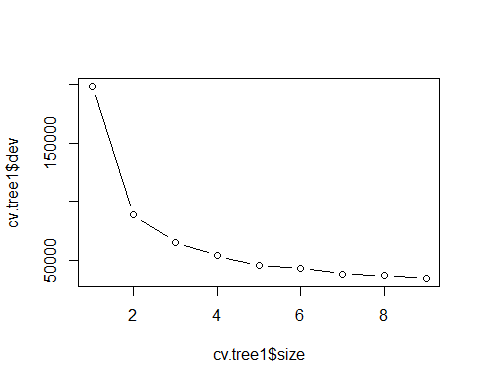
# Set seed  
set.seed(123)  
  
# Split training/test data sets  
inTraining <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)  
tree\_train <- df\_knn[inTraining,]  
tree\_test <- df\_knn[-inTraining,]  
  
# Remove country because this tree function has a maximum of 32 levels  
tree1 <- tree(life\_expectancy ~ ., data = tree\_train[, -1])  
summary(tree1)

##   
## Regression tree:  
## tree(formula = life\_expectancy ~ ., data = tree\_train[, -1])  
## Variables actually used in tree construction:  
## [1] "hiv\_aids" "income\_composition\_of\_resources"  
## [3] "adult\_mortality" "infant\_deaths"   
## Number of terminal nodes: 9   
## Residual mean deviance: 13.7 = 29990 / 2189   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -16.51000 -2.11000 -0.05483 0.00000 2.19300 16.19000

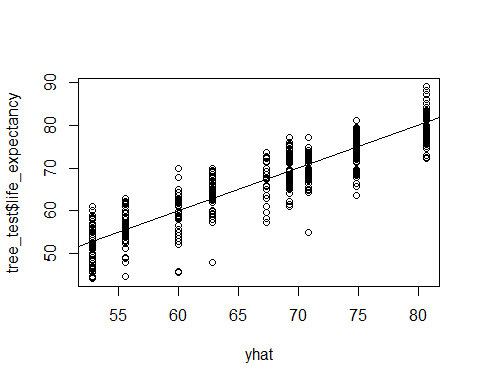
plot(tree1)



# Check tree performance  
cv.tree1 <- cv.tree(tree1)  
plot(cv.tree1$size, cv.tree1$dev, type = 'b')



# Check the predictions  
yhat = predict(tree1, newdata = tree\_test[, -1])  
plot(yhat, tree\_test$life\_expectancy)  
abline(0, 1)



mean((yhat - tree\_test$life\_expectancy) ^ 2)

## [1] 12.82312

# Try a random forest compared to a single tree model  
set.seed(1)  
  
# Remove country again  
rfFit <- randomForest(life\_expectancy ~ ., data = tree\_train[, -1])  
rfFit

##   
## Call:  
## randomForest(formula = life\_expectancy ~ ., data = tree\_train[, -1])   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 3.606142  
## % Var explained: 96

# Test using all predictors for each tree  
set.seed(1)  
  
# Have to remove country again - using top 4 predictors  
rfFit2 <- randomForest(life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources, data = tree\_train[,-1])  
rfFit2

##   
## Call:  
## randomForest(formula = life\_expectancy ~ adult\_mortality + total\_expenditure + hiv\_aids + income\_composition\_of\_resources, data = tree\_train[, -1])   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## Mean of squared residuals: 5.855018  
## % Var explained: 93.51

#compare y-hat for the two random forest models  
yhat.rf <- predict(rfFit, newdata = tree\_test[, -1])  
yhat.rf2 <- predict(rfFit2, newdata = tree\_test[, -1])  
  
# Performance measurement  
postResample(tree\_test$life\_expectancy, yhat.rf)

## RMSE Rsquared MAE   
## 1.6513999 0.9710845 1.1281576

postResample(tree\_test$life\_expectancy, yhat.rf2)

## RMSE Rsquared MAE   
## 2.0417042 0.9570357 1.5176295

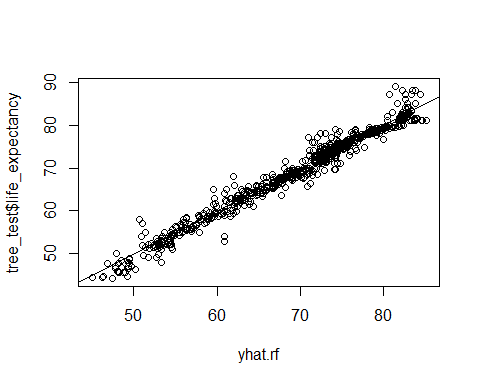
ASE\_rf <- mean((yhat.rf - tree\_test$life\_expectancy)^2)  
ASE\_rf

## [1] 2.727122

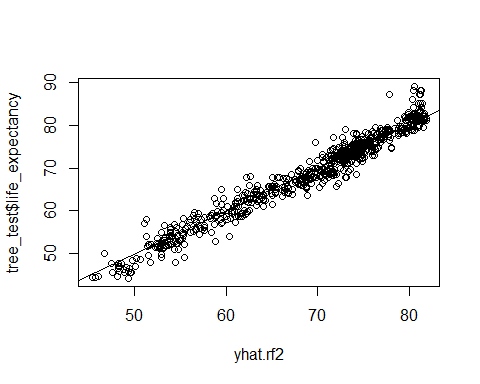
ASE\_rf2 <- mean((yhat.rf2 - tree\_test$life\_expectancy)^2)  
ASE\_rf2

## [1] 4.168556

plot(yhat.rf, tree\_test$life\_expectancy)  
abline(0,1)



plot(yhat.rf2,tree\_test$life\_expectancy)  
abline(0,1)



The random forest model that includes all predictors has smaller ASE than the random forest model that only includes the four important variables we used in our interpretable model.

## Check for overfit again

Let’s do another train/test split also for the random forest to check for overfitting, as the Rsquared values for RF are much higher than KNN.

# Set seed  
set.seed(1)  
  
# Split training/test data sets  
inTraining\_t <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)  
tree\_train\_x <- df\_knn[inTraining\_t,]  
tree\_test\_x <- df\_knn[-inTraining\_t,]  
  
# Try a random forest compared to a single tree model  
set.seed(567)  
  
# Remove country again  
rfFit\_x <- randomForest(life\_expectancy ~ ., data = tree\_train\_x[, -1])  
rfFit\_x

##   
## Call:  
## randomForest(formula = life\_expectancy ~ ., data = tree\_train\_x[, -1])   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 3.437878  
## % Var explained: 96.25

# Test using all predictors for each tree  
set.seed(5)  
  
# Have to remove country again - using top 4 predictors  
rfFit2\_x <- randomForest(life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources, data = tree\_train\_x[,-1])  
rfFit2\_x

##   
## Call:  
## randomForest(formula = life\_expectancy ~ adult\_mortality + total\_expenditure + hiv\_aids + income\_composition\_of\_resources, data = tree\_train\_x[, -1])   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## Mean of squared residuals: 5.480223  
## % Var explained: 94.02

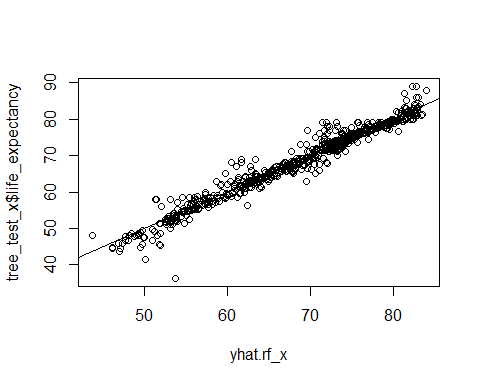
#compare y-hat for the two random forest models  
yhat.rf\_x <- predict(rfFit\_x, newdata = tree\_test\_x[, -1])  
yhat.rf2\_x <- predict(rfFit2\_x, newdata = tree\_test\_x[, -1])  
  
# Performance measurement  
postResample(tree\_test\_x$life\_expectancy, yhat.rf\_x)

## RMSE Rsquared MAE   
## 1.9159378 0.9585647 1.1869273

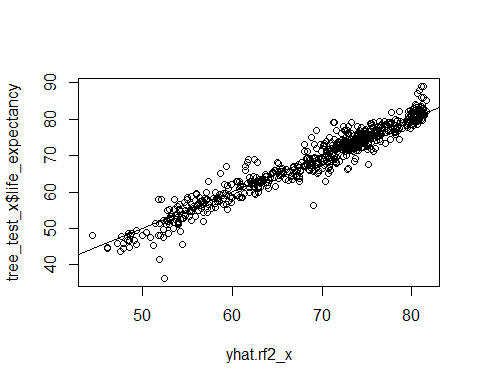
postResample(tree\_test\_x$life\_expectancy, yhat.rf2\_x)

## RMSE Rsquared MAE   
## 2.3898005 0.9360727 1.6652174

plot(yhat.rf\_x, tree\_test\_x$life\_expectancy)  
abline(0,1)



plot(yhat.rf2\_x,tree\_test\_x$life\_expectancy)  
abline(0,1)



## Objective 1

### Restatement of Problem and the Overall Approach to Solve It

### Model Selection

#### Type of Selection

* LASSO
* RIDGE
* ELASTIC NET
* Stepwise
* Forward
* Backward
* Manual / Intuition
* A mix of all of the above

#### Checking Assumptions

* Residual Plots
* Influential point analysis (Cook’s D and Leverage)

#### Compare Competing Models

Via: Training and test set split or CV Possible Metrics: ASE (Required), AIC, BIC, adj R2, are all welcome additions

#### Parameter Interpretation (Simple model only)

* Interpretation
* Confidence Intervals

#### Additional Details on a more complicated regression model

## Objective 2 Deliverable (see above)

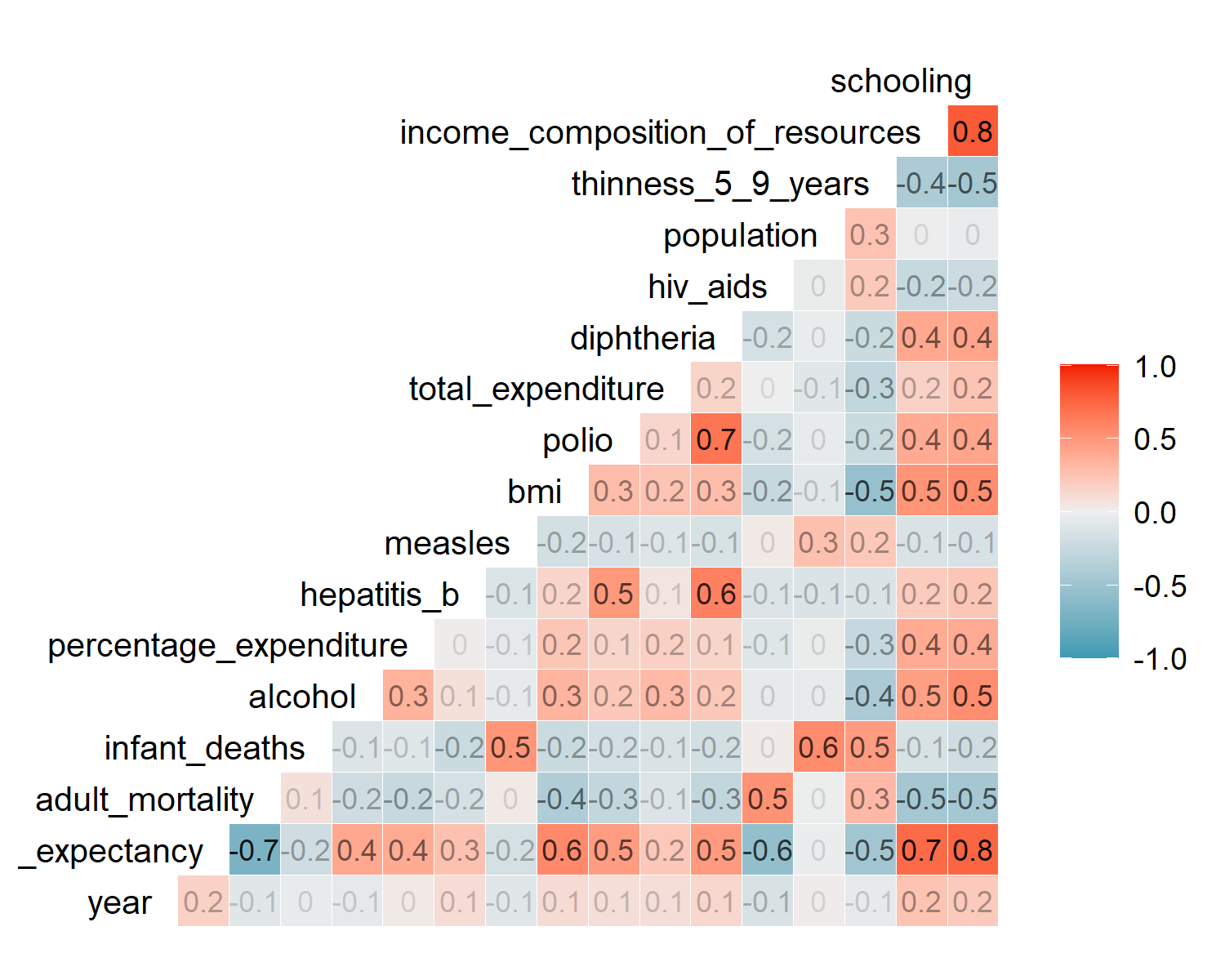
Final conclusions from the analyses of Objective 1’s interpretable model and include comments on what model would be recommended if prediction was the only goal (comparing all models considered).

# Appendix

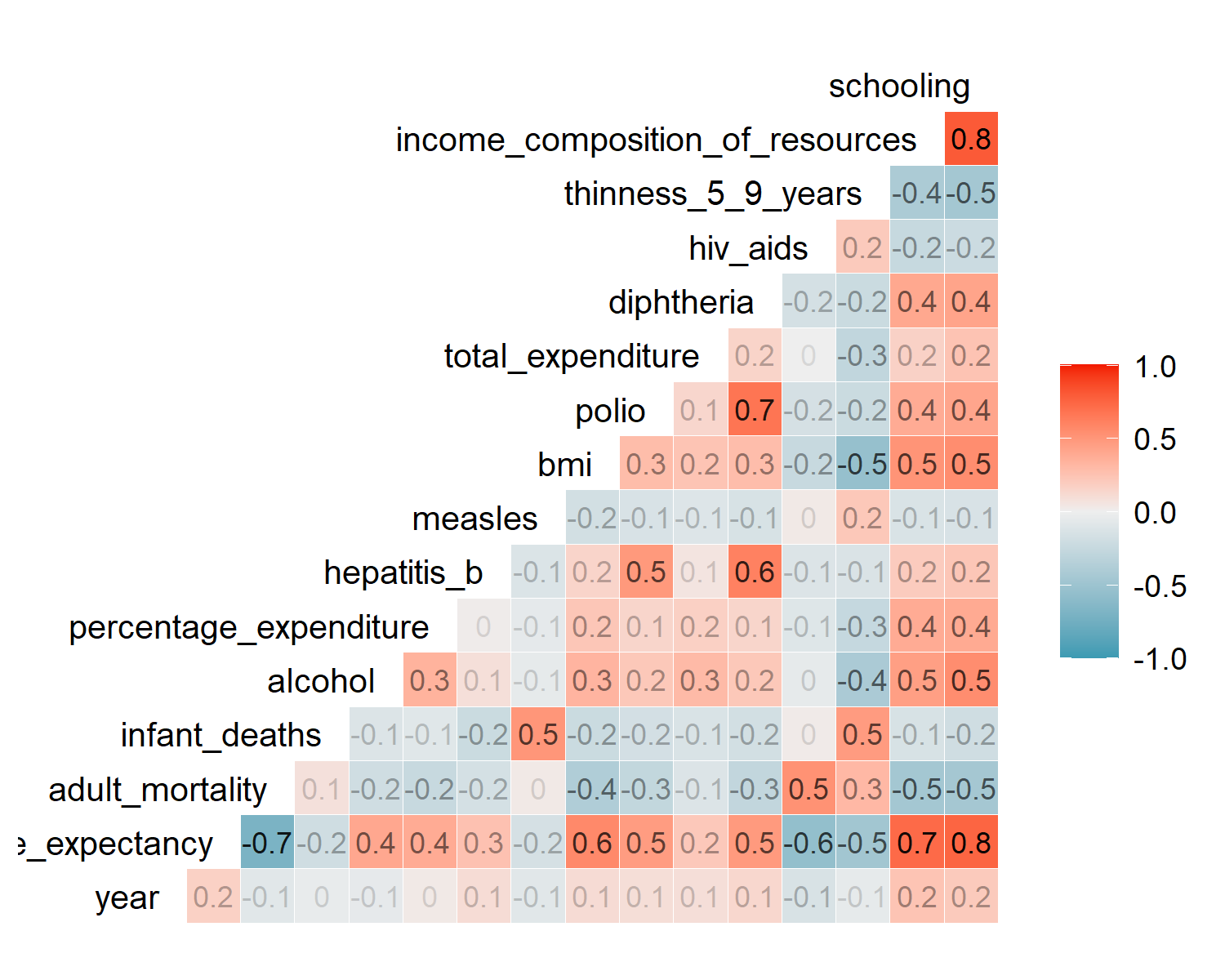
## Figures



Variable Correlation, Original



Variable Correlation, 1st Iteration



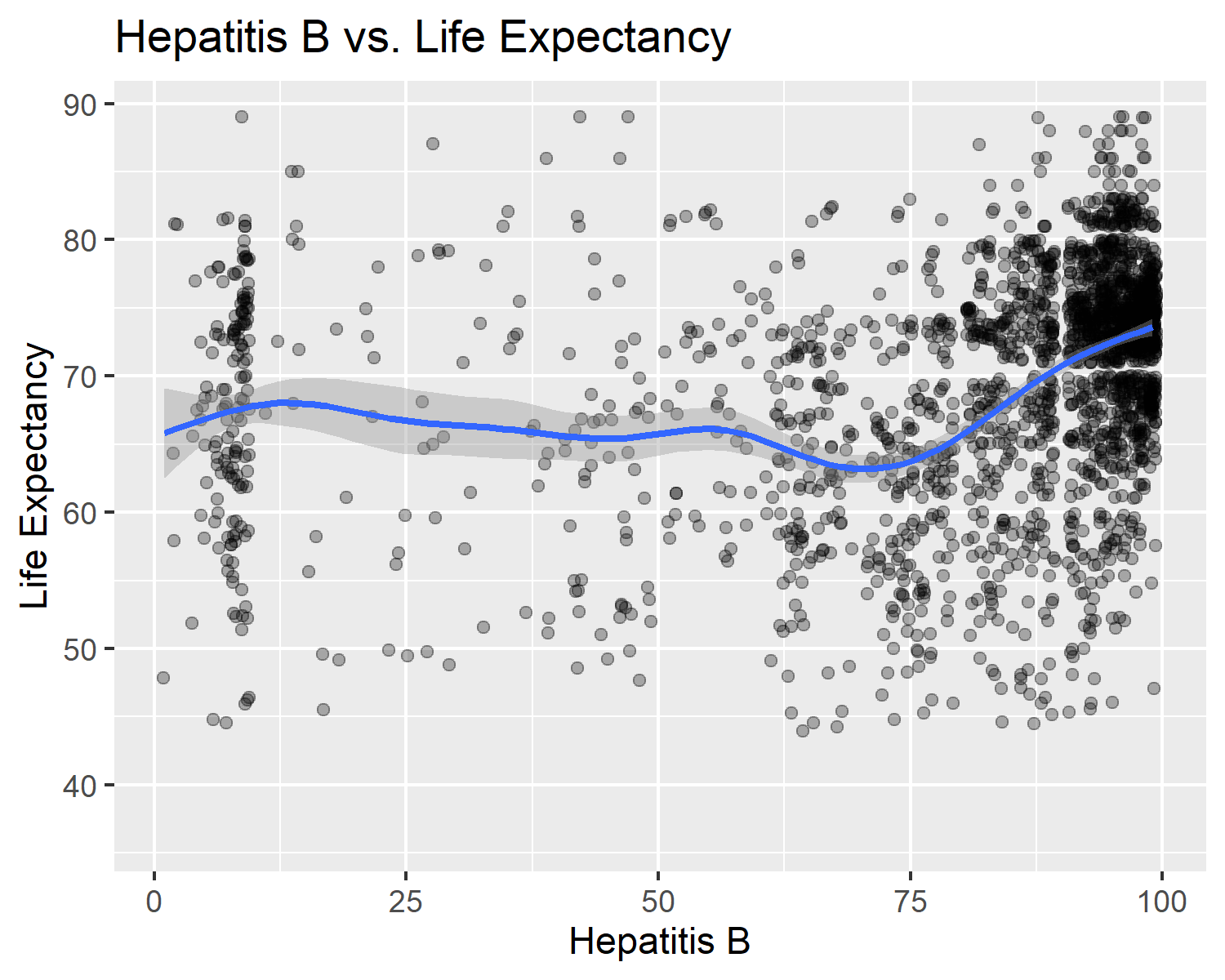
Variable Correlation, 2nd Iteration



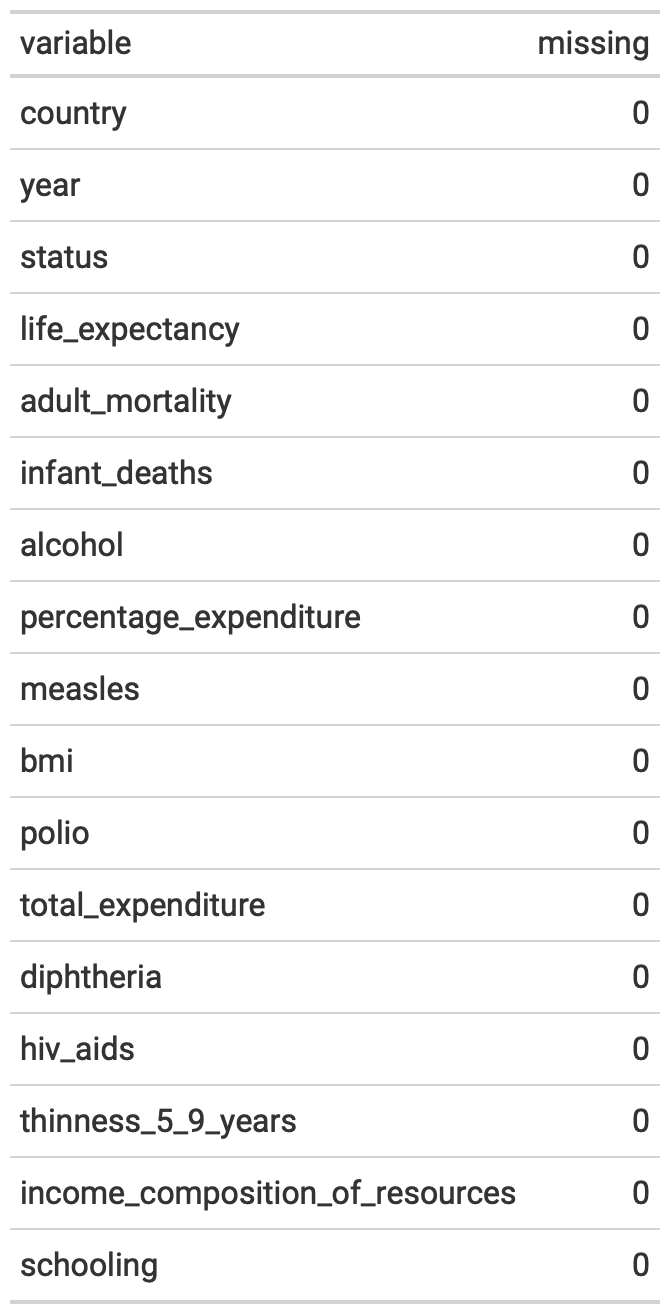
Missing Values Analysis, pt. 1



Missing Values Analysis, pt. 2



Relationship between Hepatitis B and Life Expectancy



Missing Values Analysis, pt. 3

## R Code

knitr::opts\_chunk$set(echo = TRUE)  
library(tidyverse)  
library(corrplot)  
library(GGally)  
library(gt)  
library(hrbrthemes)  
library(car)  
library(leaps)  
library(caret)  
library(tree)  
library(randomForest)  
library(plotmo)  
library(webshot)  
# Load data  
df <- read\_csv(here::here("data - raw", "Life Expectancy Data.csv"))  
  
# Clean up column names  
df\_clean <- janitor::clean\_names(df)  
  
# Look at the data  
glimpse(df\_clean)  
df\_clean %>%   
 ggplot(aes(life\_expectancy)) +  
 geom\_histogram(fill = "steelblue", color = "black") +  
 labs(title = "Histogram of Life Expectancy",  
 x = "Age",  
 y = "Count") +  
 theme\_ipsum()  
  
df\_clean %>%   
 ggplot(aes(sample = life\_expectancy)) +  
 geom\_qq(pch = 21, size = 3, na.rm = TRUE) +  
 geom\_qq\_line(color = "indianred", na.rm = TRUE) +  
 labs(title = "Quantile-Quantile Plot of Life Expectancy",  
 x = "Theoretical",  
 y = "Sample") +  
 theme\_ipsum()  
# Convert country and status to factors  
df\_clean$country <- as\_factor(df\_clean$country)  
df\_clean$status <- as\_factor(df\_clean$status)  
# Run correlations and save output as images to be used in the appendix.  
ggcorr(  
 df\_clean,  
 label = TRUE,  
 label\_alpha = TRUE,  
 label\_size = 3,  
 layout.exp = 2,  
 cex = 3.5,  
 hjust = 1  
)  
  
ggsave(here::here("images", "correlation 1.png"))  
  
df\_clean %>%   
 select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years)) %>%   
 ggcorr(  
 label = TRUE,  
 label\_alpha = TRUE,  
 label\_size = 3,  
 layout.exp = 2,  
 cex = 3.5,  
 hjust = 1  
)  
  
ggsave(here::here("images", "correlation 2.png"))  
  
df\_clean %>%   
 select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years, population)) %>%   
 ggcorr(  
 label = TRUE,  
 label\_alpha = TRUE,  
 label\_size = 3,  
 layout.exp = 2,  
 cex = 3.5,  
 hjust = 1  
)  
  
ggsave(here::here("images", "correlation 3.png"))  
df\_clean <- df\_clean %>%   
 select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years, population))  
# Check for missing values  
tibble(variable = names(colSums(is.na(df\_clean))),  
 missing = colSums(is.na(df\_clean))) %>%   
 gt() %>%   
 tab\_header(title = "Missing Values in Data") %>%   
 gtsave(here::here("images", "missing-1.png"))  
  
# Check which countries have NA rows for life expectancy  
df\_clean$country[which(is.na(df\_clean$life\_expectancy))]  
  
# Drop all rows where life expectancy is NA  
df\_clean <- df\_clean %>%   
 filter(!is.na(life\_expectancy))  
  
# Recheck missing value counts  
tibble(variable = names(colSums(is.na(df\_clean))),  
 missing = colSums(is.na(df\_clean))) %>%   
 gt() %>%   
 tab\_header(title = "Missing Values in Data") %>%   
 gtsave(here::here("images", "missing-2.png"))  
# Look at how many missing hepatitis measurements there are by country  
df\_clean %>%   
 group\_by(country) %>%   
 count(missing = is.na(hepatitis\_b)) %>%   
 filter(missing == TRUE) %>%   
 select(-missing) %>%   
 rename(missing = n) %>%   
 arrange(desc(missing))  
# Visualize the relationship to see if it looks significant  
df\_clean %>%   
 ggplot(aes(x = hepatitis\_b, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Hepatitis B vs. Life Expectancy",  
 x = "Hepatitis B",  
 y = "Life Expectancy")  
  
ggsave(here::here("images", "hepatitis-lifeexp.png"))  
  
# Drop hepatitis B variable   
df\_interp <- df\_clean %>%   
 select(-hepatitis\_b)  
  
# Drop remaining rows with NA's for the interpretable model  
df\_interp <- na.omit(df\_interp)  
  
# Final check of missing values  
tibble(variable = names(colSums(is.na(df\_interp))),  
 missing = colSums(is.na(df\_interp))) %>%   
 gt() %>%   
 gtsave(here::here("images", "missing-3.png"))  
df\_interp %>% count(year)  
df\_interp <- df\_interp %>% filter(year %in% 2011:2014)  
# Set the maximum number of variables to consider in the model. Although the  
# model can handle up to 20, the more we add, the less interpretable the final  
# model will be.  
consider <- 17  
  
# Fit the model. We'll remove country, but keep it in the data set for  
# interpretation.  
regfit\_full <-  
 regsubsets(life\_expectancy ~ .,  
 df\_interp[, -1],  
 nvmax = consider)  
  
# Store the regression summary  
reg\_summary <- summary(regfit\_full)  
# Look at the names of reg\_summary  
names(reg\_summary)  
  
# What are the R-squared values?  
reg\_summary$rsq  
par(mfrow = c(2, 2))  
plot(reg\_summary$rss,  
 xlab = "Number of Variables",  
 ylab = "RSS",  
 type = "l")  
plot(reg\_summary$adjr2,  
 xlab = "Number of Variables",  
 ylab = "Adjusted RSq",  
 type = "l")  
# which.max(reg\_summary$adjr2)  
points(  
 11,  
 reg\_summary$adjr2[which.max(reg\_summary$adjr2)],  
 col = "red",  
 cex = 2,  
 pch = 20  
)  
plot(reg\_summary$cp,  
 xlab = "Number of Variables",  
 ylab = "Cp",  
 type = "l")  
# which.min(reg\_summary$cp)  
points(  
 9,  
 reg\_summary$cp[which.min(reg\_summary$cp)],  
 col = "red",  
 cex = 2,  
 pch = 20  
)  
plot(reg\_summary$bic,  
 xlab = "Number of Variables",  
 ylab = "BIC",  
 type = "l")  
# which.min(reg\_summary$bic)  
points(  
 5,  
 reg\_summary$bic[which.min(reg\_summary$bic)],  
 col = "red",  
 cex = 2,  
 pch = 20  
)  
plot(regfit\_full, scale = "r2")  
plot(regfit\_full, scale = "adjr2")  
plot(regfit\_full, scale = "Cp")  
plot(regfit\_full, scale = "bic")  
coef(regfit\_full, 5)  
# Forward  
regfit\_fwd <-  
 regsubsets(  
 life\_expectancy ~ .,  
 data = df\_interp[,-1],  
 nvmax = 17,  
 method = "forward"  
 )  
# summary(regfit\_fwd)  
  
# Backward  
regfit\_bwd <-  
 regsubsets(  
 life\_expectancy ~ .,  
 data = df\_interp[,-1],  
 nvmax = 17,  
 method = "backward"  
 )  
# summary(regfit\_bwd)  
  
# How many variables do we want to use?  
x <- 4  
  
# Compare coefficients  
tibble(  
 variables = names(coef(regfit\_full, x)),  
 full = round(coef(regfit\_full, x), 4),  
 fwd = round(coef(regfit\_fwd, x), 4),  
 bwd = round(coef(regfit\_bwd, x), 4)  
) %>%  
 gt()  
# Set seed  
set.seed(123)  
  
# Build test and training data sets, dropping country from regfit\_best &  
# test\_mat because I was getting a warning that 1 linear dependencies found  
train <- sample(c(TRUE, FALSE), nrow(df\_interp), rep = TRUE)  
test <- (!train)  
regfit\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[train, -1])  
test\_mat <- model.matrix(life\_expectancy ~ ., data = df\_interp[test, -1])  
  
# Run a loop, and for each size `i`, extract the coefficients from `regfit\_best`  
# for the best model of that size, multiply them into the appropriate columns of  
# the test model matric to form the predictions, and compute the test MSE.  
val\_errors <- rep(NA, 8)  
for (i in 1:8) {  
 coefi <- coef(regfit\_best, id = i)  
 pred <- test\_mat[, names(coefi)] %\*% coefi  
 val\_errors[i] <- mean((df\_interp$life\_expectancy[test] - pred)^2)  
}  
# Find the best model  
val\_errors  
coef(regfit\_best, which.min(val\_errors))  
# Write a prediction function  
predict\_regsubsets <- function(object, newdata, id, ...) {  
 form <- as.formula(object$call[[2]])  
 mat <- model.matrix(form, newdata)  
 coefi <- coef(object, id = id)  
 xvars <- names(coefi)  
 mat[, xvars] %\*% coefi  
}  
  
# Perform best subset selection on the full data set, and select the best model.  
regfit\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[,-1])  
coef(regfit\_best, which.min(val\_errors))  
k <- 10  
set.seed(123)  
folds <- sample(1:k, nrow(df\_interp), replace = TRUE)  
cv\_errors <- matrix(NA, k, consider, dimnames = list(NULL, paste(1:consider)))  
  
predict.regsubsets <- function(object, newdata, id, ...) {  
 form <- as.formula(object$call[[2]])  
 mat <- model.matrix(form, newdata)  
 coefi <- coef(object, id = id)  
 mat[, names(coefi)] %\*% coefi  
}  
  
# Perform cross-validation  
for (j in 1:k) {  
 best\_fit <-  
 regsubsets(life\_expectancy ~ ., data = df\_interp[folds != j, -1], nvmax = consider)  
 for (i in 1:15) {  
 pred <- predict(best\_fit, df\_interp[folds == j, -1], id = i)  
 cv\_errors[j, i] <-  
 mean((df\_interp$life\_expectancy[folds == j] - pred) ^ 2)  
 }  
}  
  
# Use the apply function to average over the columns of the matrix in order to  
# obtain a vector for which the jth element is the cross-validation error for  
# the j-variable model.  
mean\_cv\_errors <- apply(cv\_errors, 2, mean)  
mean\_cv\_errors  
par(mfrow = c(1, 1))  
plot(mean\_cv\_errors, type = "b")  
  
# Perform best subset selection on the full data set in order to obtain the  
# variables for the final model  
reg\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[, -1])  
coef(reg\_best, 4) # the number sets the number of variables we want  
# Build the final model using the best subset selection results  
final\_model <-  
 lm(  
 life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources,  
 data = df\_interp  
)  
# Final model summary  
final\_model\_summary <- summary(final\_model)  
final\_model\_summary  
  
# Confidence intervals  
confint(final\_model)  
par(mfrow = c(1, 1))  
plot(  
 final\_model$fitted.values,  
 df\_interp$life\_expectancy,  
 xlab = "Predicted",  
 ylab = "Life Expectancy"  
)  
lines(c(0, 90), c(0, 90), col = "red")  
  
par(mfrow=c(2,2))  
plot(final\_model)  
par(mfrow = c(1, 2))  
test.model <- lm(life\_expectancy ~ ., df\_interp[, -1])  
plot(test.model$fitted.values,  
 test.model$residuals,  
 xlab = "Fitted Values",  
 ylab = "Residuals")  
plot(df\_interp$life\_expectancy,  
 test.model$residuals,  
 xlab = "Life Expectancy",  
 ylab = "Residuals")  
# Check for missing values  
tibble(variable = names(colSums(is.na(df\_clean))),  
 missing = colSums(is.na(df\_clean))) %>%   
 gt()  
#Visualize the relationship to see if it looks significant (plot again)  
  
# hepatitis b  
df\_clean %>%  
 ggplot(aes(x = hepatitis\_b, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Hepatitis B and Life Expectancy",  
 x = "Hepatitis B",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
  
# total expenditure  
df\_clean %>%  
 ggplot(aes(x = total\_expenditure, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Total Expenditure and Life Expectancy",  
 x = "Total Expenditure",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
  
# alcohol  
df\_clean %>%  
 ggplot(aes(x = alcohol, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Alcohol and Life Expectancy",  
 x = "Alcohol",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
  
# income composition of resources  
df\_clean %>%  
 ggplot(aes(x = income\_composition\_of\_resources, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Income Composition of Resources\nand Life Expectancy",  
 x = "Income Composition of Resources",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
  
# schooling  
df\_clean %>%  
 ggplot(aes(x = schooling, y = life\_expectancy)) +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Schooling and Life Expectancy",  
 x = "Schooling",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
#remove hepatitis b and total\_expenditure  
df\_predict <- df\_clean %>%   
 select(-c(hepatitis\_b, total\_expenditure, alcohol))  
  
#remove the remaining NA's  
df\_predict <- na.omit(df\_predict)  
  
#check for NA's  
tibble(variable = names(colSums(is.na(df\_predict))),  
 missing = colSums(is.na(df\_predict))) %>%   
 gt()  
#replot after removing Na's  
  
# income composition of resources  
df\_predict %>%  
 ggplot(aes(x = income\_composition\_of\_resources, y = life\_expectancy)) +  
 # geom\_point() +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Income Composition of Resources\nand Life Expectancy",  
 x = "Income Composition of Resources",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
  
# schooling  
df\_predict %>% ggplot(aes(x = schooling, y = life\_expectancy)) +  
 # geom\_point() +  
 geom\_jitter(alpha = 0.3) +  
 geom\_smooth() +  
 labs(title = "Relationship of Schooling and Life Expectancy",  
 x = "Schooling",  
 y = "Life Expectancy") +  
 theme\_ipsum()  
library(glmnet)  
  
# Ridge regression and lasso require the format 'x matrix' and 'y'. The  
# model.matrix() function produces a matrix and automatically transforms  
# qualitative variables into dummy variables.  
x <- model.matrix(life\_expectancy ~ ., df\_predict[,-1])[, -1]  
y <- df\_predict$life\_expectancy  
  
# Run ridge regression  
ridge\_mod <- glmnet(x, y, alpha = 0)  
dim(coef(ridge\_mod))  
  
# Split data into training and testing sets  
set.seed(1)  
train <- sample(1:nrow(x), nrow(x)/2)  
test <- (-train)  
y\_test <- y[test]  
  
# Fit a ridge regression model on the training set, and evaluate its MSE on the  
# test set  
ridge\_mod <- glmnet(x[train, ], y[train], alpha = 0)  
ridge\_pred <- predict(ridge\_mod, s = 4, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)  
#[1] 17.4224  
  
# The test MSE is 17.4224. If we had simply fit a model with just an intercept,  
# we would have observed each test observation using the mean of the training  
# observations. In that case, we could compute the test set MSE like this:  
mean((mean(y[train]) - y\_test) ^ 2)  
#[1] 89.45262  
  
# We could also get the same result by fitting a ridge regression model with a  
# very large value of ƛ.  
ridge\_pred <- predict(ridge\_mod, s = 1e10, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)  
#[1] 89.45262  
  
# Use cross-validation to choose the tuning parameter ƛ.  
set.seed(1)  
cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 0)  
plot(cv\_out)  
bestlambda <- cv\_out$lambda.min  
bestlambda  
  
# What is the test MSE associated with bestlambda?  
ridge\_pred <- predict(ridge\_mod, s = bestlambda, newx = x[test, ])  
mean((ridge\_pred - y\_test) ^ 2)  
#[1] 16.2937  
  
# Refit the ridge regression model on the full data set using the value of ƛ  
# chosen by cross-validation  
out <- glmnet(x, y, alpha = 0)  
predict(out, type = "coefficients", s = bestlambda)[1:14,]  
# Fit the lasso model  
lasso\_mod <- glmnet(x[train, ], y[train], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod)  
  
# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 1)  
plot(cv\_out)  
bestlambda <- cv\_out$lambda.min  
lasso\_pred <- predict(lasso\_mod, s = bestlambda, newx = x[test, ])  
mean((lasso\_pred - y\_test) ^ 2)  
#[1] 16.41776  
  
# Compute lasso coefficients  
out <- glmnet(x, y, alpha = 1)  
lasso\_coef <- predict(out, type = "coefficients", s = bestlambda)[1:14,]  
lasso\_coef[lasso\_coef != 0]  
#plot distributions  
df\_plot <- as.data.frame(df\_clean)  
for (col in 5:ncol(df\_plot)) {  
 hist(df\_plot[,col], main=names(df\_plot[col]))  
}  
  
# Copy the data  
df\_predict2 <- df\_clean %>%   
 select(-c(hepatitis\_b))  
  
# Impute the data set  
df\_predict2 <- df\_predict2 %>%  
 mutate(across(  
 c(polio, total\_expenditure, diphtheria),  
 ~ replace\_na(., median(.x, na.rm = TRUE))  
 )) %>%   
 mutate(across(  
 c(bmi, income\_composition\_of\_resources, schooling),  
 ~ replace\_na(., mean(.x, na.rm = TRUE))  
 )) %>%   
 mutate(across(  
 c(alcohol, thinness\_5\_9\_years),  
 ~ replace\_na(., 0)  
 ))  
  
  
#check for NA's  
tibble(variable = names(colSums(is.na(df\_predict2))),  
 missing = colSums(is.na(df\_predict2))) %>%   
 gt()  
  
# Ridge regression and lasso require the format 'x matrix' and 'y'. The  
# model.matrix() function produces a matrix and automatically transforms  
# qualitative variables into dummy variables.  
sum(is.na(df\_predict2))  
x2 <- model.matrix(life\_expectancy ~ ., df\_predict2[,-1])[, -1]  
y2 <- df\_predict2$life\_expectancy  
  
# Run ridge regression  
ridge\_mod2 <- glmnet(x2, y2, alpha = 0)  
dim(coef(ridge\_mod2))  
  
# Split data into training and testing sets  
set.seed(1)  
train2 <- sample(1:nrow(x2), nrow(x2)/2)  
test2 <- (-train2)  
y\_test2 <- y2[test2]  
  
# Fit a ridge regression model on the training set, and evaluate its MSE on the  
# test set  
ridge\_mod2 <- glmnet(x2[train2,], y2[train2], alpha = 0)  
ridge\_pred2 <- predict(ridge\_mod2, s = 4, newx = x2[test2, ])  
mean((ridge\_pred2 - y\_test2) ^ 2)  
#[1] 19.39197  
  
# The test MSE is 19.39197. If we had simply fit a model with just an intercept,  
# we would have observed each test observation using the mean of the training  
# observations. In that case, we could compute the test set MSE like this:  
mean((mean(y2[train2]) - y\_test2) ^ 2)  
#[1] 88.25497  
  
# Use cross-validation to choose the tuning parameter ƛ.  
set.seed(1)  
cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 0)  
plot(cv\_out2)  
bestlambda2 <- cv\_out2$lambda.min  
bestlambda2  
#[1] 0.6900917  
  
# What is the test MSE associated with bestlambda?  
ridge\_pred2 <- predict(ridge\_mod2, s = bestlambda2, newx = x2[test2, ])  
mean((ridge\_pred2 - y\_test2) ^ 2)  
#[1] 18.68608  
  
# Refit the ridge regression model on the full data set using the value of ƛ  
# chosen by cross-validation  
out2 <- glmnet(x2, y2, alpha = 0)  
predict(out2, type = "coefficients", s = bestlambda2)[1:14,]  
# Fit the lasso model  
lasso\_mod2 <- glmnet(x2[train2, ], y2[train2], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod2)  
  
# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 1)  
plot(cv\_out2)  
bestlambda2 <- cv\_out2$lambda.min  
lasso\_pred2 <- predict(lasso\_mod2, s = bestlambda2, newx = x2[test2, ])  
mean((lasso\_pred2 - y\_test2) ^ 2)  
#[1] 18.74681  
  
# Compute lasso coefficients  
out2 <- glmnet(x2, y2, alpha = 1)  
lasso\_coef2 <- predict(out2, type = "coefficients", s = bestlambda2)[1:14,]  
lasso\_coef2[lasso\_coef2 != 0]  
# Fit the lasso model with data set including country  
x3 <- model.matrix(life\_expectancy ~ ., df\_predict2)[, -1]  
y3 <- df\_predict2$life\_expectancy  
  
# Split data into training and testing sets  
set.seed(1)  
train3 <- sample(1:nrow(x3), nrow(x3)/2)  
test3 <- (-train3)  
y\_test3 <- y3[test3]  
  
lasso\_mod3 <- glmnet(x3[train3, ], y3[train3], alpha = 1)  
  
# Plot the lasso model  
plot\_glmnet(lasso\_mod3)  
  
# Run cross-validation and compute the associated test error  
set.seed(1)  
cv\_out3 <- cv.glmnet(x3[train3, ], y3[train3], alpha = 1)  
plot(cv\_out3)  
bestlambda3 <- cv\_out3$lambda.min  
lasso\_pred3 <- predict(lasso\_mod3, s = bestlambda3, newx = x3[test3, ])  
mean((lasso\_pred3 - y\_test3) ^ 2)  
#[1] 18.74681  
  
# Compute lasso coefficients  
out3 <- glmnet(x3, y3, alpha = 1)  
lasso\_coef3 <- predict(out3, type = "coefficients", s = bestlambda3)[1:14,]  
lasso\_coef3[lasso\_coef3 != 0]  
#build a linear model using the top 3 variables from LASSO along with country for both imputed data set and removed NA data set  
  
# Build the final model using the best subset selection results on the imputed data set  
predict\_model1 <-  
 lm(  
 life\_expectancy ~   
 country +  
 income\_composition\_of\_resources +  
 status +  
 schooling +  
 hiv\_aids,  
 data = df\_predict2  
)  
  
# Final model summary  
predict\_model1\_sum <- summary(predict\_model1)  
predict\_model1\_sum  
  
# Get MSE  
mean(predict\_model1\_sum$residuals ^ 2)  
#[1] 4.39226  
  
# Build the final model using the best subset selection results on removed NA data set  
predict\_model2 <-  
 lm(  
 life\_expectancy ~   
 country +  
 income\_composition\_of\_resources +  
 status +  
 schooling +  
 hiv\_aids,  
 data = df\_predict  
)  
  
# Final model summary  
predict\_model2\_sum <- summary(predict\_model2)  
predict\_model2\_sum  
  
# Get MSE  
mean(predict\_model2\_sum$residuals ^ 2)  
#[1] 4.189329  
par(mfrow = c(1, 1))  
plot(  
 predict\_model2$fitted.values,  
 df\_predict$life\_expectancy,  
 xlab = "Predicted Life Expectancy",  
 ylab = "Actual Life Expectancy"  
)  
lines(c(0, 90), c(0, 90), col = "red")  
  
par(mfrow=c(2,2))  
plot(predict\_model2)  
  
#copy and rename imputed data set for KNN models  
df\_knn <- df\_predict2  
  
#Make new data set that does not impute values for comparison  
df\_knn2 <- df\_clean  
df\_knn2 <- drop\_na(df\_knn2)  
  
  
# Set seed  
set.seed(123)  
  
# Standardize the data to prep for KNN first - everything except life expectancy  
preProcValues <- preProcess(df\_knn[, -4], method = c("scale"))  
df\_knn\_standard <- predict(preProcValues, df\_knn)  
  
# Split training/test data sets  
inTraining <-  
 createDataPartition(df\_knn\_standard$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train <- df\_knn\_standard[inTraining, ]  
knn\_test <- df\_knn\_standard[-inTraining, ]  
  
# Perform same splits for data with NA's removed  
# Standardize the data to prep for KNN first - everything except life expectancy  
preProcValues\_2 <- preProcess(df\_knn2[, -4], method = c("scale"))  
df\_knn\_standard2 <- predict(preProcValues\_2, df\_knn2)  
  
# Split training/test data sets  
inTraining2 <-  
 createDataPartition(df\_knn\_standard2$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train\_na <- df\_knn\_standard2[inTraining2, ]  
knn\_test\_na <- df\_knn\_standard2[-inTraining2, ]  
# Set seed  
set.seed(567)  
  
# Set train control: 5 repeat, 10-fold CV  
ctrl <-  
 trainControl(  
 method = "repeatedcv",  
 number = 10, # 10-fold CV  
 repeats = 5, #repeat 5 times  
 returnResamp = "all" #return all metrics  
 )   
  
# Run everything with the train control above  
knnFit <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10 #run for 10 different k's  
 )  
knnFit  
  
# Check the metrics on the test set  
Predictions\_knn5 <- predict(knnFit, newdata = knn\_test)  
ASE\_knn5 <- mean((Predictions\_knn5 - knn\_test$life\_expectancy)^2)  
ASE\_knn5  
  
# Performance measurement  
postResample(knn\_test$life\_expectancy, Predictions\_knn5)  
  
  
# Plotting  
plot(knnFit, main = "knnFit Results")   
plot(knnFit, metric = "Rsquared", main = "knnFit Results (R-Squared)")  
plot(knnFit, metric = "MAE", main = "knnFit Results (MAE)")  
  
# Try another knn with a k value of less than 5  
knnFit2 <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneGrid = expand.grid(k = c(1, 3, 5)) #run only with k = 1, 3, 5  
 )  
knnFit2  
  
# Check the metrics on the test set  
Predictions\_knn3 <- predict(knnFit2, newdata = knn\_test)  
ASE\_knn3 <- mean((Predictions\_knn3 - knn\_test$life\_expectancy)^2)  
ASE\_knn3  
  
# Performance measurement  
postResample(knn\_test$life\_expectancy, Predictions\_knn3)  
  
# Plotting  
plot(knnFit2, main = "knnFit2 Results")   
plot(knnFit2, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")  
plot(knnFit2, metric = "MAE", main = "knnFit2 Results (MAE)")  
  
# Run knn on data set with NA's removed  
knnFit\_na <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_na,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10 #run through 10 different k's  
 )  
knnFit\_na  
  
# Check the metrics on the test set  
Predictions\_knn\_na <- predict(knnFit\_na, newdata = knn\_test\_na)  
ASE\_knn\_na <- mean((Predictions\_knn\_na - knn\_test\_na$life\_expectancy)^2)  
ASE\_knn\_na  
  
# Performance measurement  
postResample(knn\_test\_na$life\_expectancy, Predictions\_knn\_na)  
  
knnFit3 <-  
 train(  
 life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources,  
 data = knn\_train,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit3  
  
#check metrics on test set  
Predictions\_knnfit3 <- predict(knnFit3,newdata=knn\_test)  
#performance measurement  
postResample(knn\_test$life\_expectancy,Predictions\_knnfit3)  
ASE\_knn3 <- mean((Predictions\_knnfit3 - knn\_test$life\_expectancy)^2)  
ASE\_knn3  
  
plot(knnFit3)   
plot(knnFit3, metric = "Rsquared")  
plot(knnFit3, metric = "MAE")  
# Split training/test data sets - round 2  
set.seed(1)  
inTraining\_x <-  
 createDataPartition(df\_knn\_standard$life\_expectancy,  
 p = 0.75,  
 list = FALSE)  
knn\_train\_x <- df\_knn\_standard[inTraining\_x, ]  
knn\_test\_x <- df\_knn\_standard[-inTraining\_x, ]  
  
# Run everything with the train control above  
knnFit\_x <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit\_x  
  
# Check the metrics on the test set  
Predictions\_knnFit\_x <- predict(knnFit\_x, newdata = knn\_test\_x)  
ASE\_x <- mean((Predictions\_knnFit\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_x  
  
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit\_x)  
  
# Plotting  
plot(knnFit\_x, main = "knnFit Results")   
plot(knnFit\_x, metric = "Rsquared", main = "knnFit Results (R-Squared)")  
plot(knnFit\_x, metric = "MAE", main = "knnFit Results (MAE)")  
  
# Try another knn with a k value of less than 5  
knnFit2\_x <-  
 train(  
 life\_expectancy ~ .,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneGrid = expand.grid(k = c(1, 3, 5))  
 )  
knnFit2\_x  
  
# Check the metrics on the test set  
Predictions\_knnFit2\_x <- predict(knnFit2\_x, newdata = knn\_test\_x)  
ASE\_x <- mean((Predictions\_knnFit2\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_x  
  
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit2\_x)  
  
# Plotting  
plot(knnFit2\_x, main = "knnFit2 Results")   
plot(knnFit2\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")  
plot(knnFit2\_x, metric = "MAE", main = "knnFit2 Results (MAE)")  
  
knnFit3\_x <-  
 train(  
 life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources,  
 data = knn\_train\_x,  
 method = "knn",  
 trControl = ctrl,  
 tuneLength = 10  
 )  
knnFit3\_x  
  
# Check the metrics on the test set  
Predictions\_knnFit3\_x <- predict(knnFit3\_x, newdata = knn\_test\_x)  
ASE\_3x <- mean((Predictions\_knnFit3\_x - knn\_test\_x$life\_expectancy)^2)  
ASE\_3x  
  
# Performance measurement  
postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit3\_x)  
  
  
# Plotting  
plot(knnFit3\_x, main = "knnFit2 Results")   
plot(knnFit3\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")  
plot(knnFit3\_x, metric = "MAE", main = "knnFit2 Results (MAE)")  
  
# Set seed  
set.seed(123)  
  
# Split training/test data sets  
inTraining <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)  
tree\_train <- df\_knn[inTraining,]  
tree\_test <- df\_knn[-inTraining,]  
  
# Remove country because this tree function has a maximum of 32 levels  
tree1 <- tree(life\_expectancy ~ ., data = tree\_train[, -1])  
summary(tree1)  
plot(tree1)  
  
# Check tree performance  
cv.tree1 <- cv.tree(tree1)  
plot(cv.tree1$size, cv.tree1$dev, type = 'b')  
  
# Check the predictions  
yhat = predict(tree1, newdata = tree\_test[, -1])  
plot(yhat, tree\_test$life\_expectancy)  
abline(0, 1)  
mean((yhat - tree\_test$life\_expectancy) ^ 2)  
  
# Try a random forest compared to a single tree model  
set.seed(1)  
  
# Remove country again  
rfFit <- randomForest(life\_expectancy ~ ., data = tree\_train[, -1])  
rfFit  
  
# Test using all predictors for each tree  
set.seed(1)  
  
# Have to remove country again - using top 4 predictors  
rfFit2 <- randomForest(life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources, data = tree\_train[,-1])  
rfFit2  
  
#compare y-hat for the two random forest models  
yhat.rf <- predict(rfFit, newdata = tree\_test[, -1])  
yhat.rf2 <- predict(rfFit2, newdata = tree\_test[, -1])  
  
# Performance measurement  
postResample(tree\_test$life\_expectancy, yhat.rf)  
postResample(tree\_test$life\_expectancy, yhat.rf2)  
  
ASE\_rf <- mean((yhat.rf - tree\_test$life\_expectancy)^2)  
ASE\_rf  
  
ASE\_rf2 <- mean((yhat.rf2 - tree\_test$life\_expectancy)^2)  
ASE\_rf2  
  
plot(yhat.rf, tree\_test$life\_expectancy)  
abline(0,1)  
  
plot(yhat.rf2,tree\_test$life\_expectancy)  
abline(0,1)  
# Set seed  
set.seed(1)  
  
# Split training/test data sets  
inTraining\_t <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)  
tree\_train\_x <- df\_knn[inTraining\_t,]  
tree\_test\_x <- df\_knn[-inTraining\_t,]  
  
# Try a random forest compared to a single tree model  
set.seed(567)  
  
# Remove country again  
rfFit\_x <- randomForest(life\_expectancy ~ ., data = tree\_train\_x[, -1])  
rfFit\_x  
  
# Test using all predictors for each tree  
set.seed(5)  
  
# Have to remove country again - using top 4 predictors  
rfFit2\_x <- randomForest(life\_expectancy ~ adult\_mortality +  
 total\_expenditure +  
 hiv\_aids +  
 income\_composition\_of\_resources, data = tree\_train\_x[,-1])  
rfFit2\_x  
  
#compare y-hat for the two random forest models  
yhat.rf\_x <- predict(rfFit\_x, newdata = tree\_test\_x[, -1])  
yhat.rf2\_x <- predict(rfFit2\_x, newdata = tree\_test\_x[, -1])  
  
# Performance measurement  
postResample(tree\_test\_x$life\_expectancy, yhat.rf\_x)  
postResample(tree\_test\_x$life\_expectancy, yhat.rf2\_x)  
  
plot(yhat.rf\_x, tree\_test\_x$life\_expectancy)  
abline(0,1)  
  
plot(yhat.rf2\_x,tree\_test\_x$life\_expectancy)  
abline(0,1)  
knitr::include\_graphics(here::here("images", "correlation 1.png"))  
knitr::include\_graphics(here::here("images", "correlation 2.png"))  
knitr::include\_graphics(here::here("images", "correlation 3.png"))  
knitr::include\_graphics(here::here("images", "missing-1.png"))  
knitr::include\_graphics(here::here("images", "missing-2.png"))  
knitr::include\_graphics(here::here("images", "hepatitis-lifeexp.png"))  
knitr::include\_graphics(here::here("images", "missing-3.png"))