report

Title of the report

Introduction

World Health Organization estimates 7 million that air pollution contributes to the death of 7 million people, anually (WHO 2020). For policymakers, it is thus impportant to understand which factors contribute to air quality, in order to target their efforts to improve it. This study aims to pinpoint the factors that contribute to air quality, using multiple regression. The factors considered in this study are rainfall, population density, income per capita, added value of companies and adjacency to coast. To find which combination of these variables best described their relationship on air quality, we use the better subset selection algorithm (Xiong 2014). Previous work showed this algorithm yields a better fit than a subset without optimization as the result of its monotonicity (Xiong 2014). Our data revealed that whether or not an area is coastal was the only statistically significant variable.

Data

Previous work has suggested the relation between the five chosen variables and air quality. Both natural and anthropogenic events attribute to air quality in the atmosphere. The distribution of air pollution mainly depends on the wind field (Leelőssy et al. 2014), which is quantified by the variable of adjacency to coast and rainfall in this study as they both reflect the wind field's condition. Air quality is also influenced by the production and consumption from society, leading to emmissions (Baklanov, Molina, and Gauss 2016). To account for this, our study includes variables on population density, income per capita and added values from companies. We use the econometrics dataset on air quality in California for 1972 (r-project 2020). The dependent variables under study are rainfall(inch), population density(per square mile), income per capita ($), added value of companies ($) and adjacency to coast(binary). The dataset has 30 set of observations, each being a different metropolitan area in California. As the unit of each variable is heterogenerous, we scaled all independent variables to follow a standard normal distribution, to ensure fair interpretation of the model coefficents. Scaling also ensures the MM algorithm converges faster.

Method

Result

We use the better subset method to select the best model for an M number of variables, with $M \in [1, 5]$. To assess the overall fit of each model, we look at adjusted R^2. We use this statistic because while it summarizes the % of the variance explained by the independent variables, it accounts for the fact that adding variables could improve the R^2 by random chance. The model that best explains the variance in air quality is the model in which all 5 variables are included, with an adjusted-R^2 of 25.4%.

	Dependent variable:				
	Air Quality				
	(1)	(2)	(3)	(4)	(5)
Coastal Area	-13.73^{***}	-13.83^{***}	-14.85^{***}	-14.98^{***}	-15.57^{***}
	(-2.54)	(-2.76)	(-2.98)	(-3.03)	(-3.19)
Value Added	-	9.26	3.63	3.33	4.090
		(0.86)	(0.34)	(0.31)	(0.39)
Median Income	-	-	6.3	7.2	6.9
			(0.58)	(0.67)	(0.65)
Population Density	-	-	-	-2.94	-3.04
				(-0.63)	(-0.66)
Rain	-	-	-	-	3.38
					0.72
Intercept	104.700^{***}	104.700^{***}	104.700^{***}	104.700^{***}	104.700^{***}
	(21.34)	(23.075)	(23.24)	(23.43)	(23.69)
Observations	30	30	30	30	30
R^2	0.240	0.349	0.359	0.369	0.383
Adjusted R^2	0.08	0.214	0.23	0.24	0.25
Note:					*p<0.1; **p<0.05; ***p<0.01

Reference

Functions

```
# calcRSS: Calculates the residual squared errors for a multiple regression of the form Y = XBeta + e
#
# Parameters:
#   mX: Matrix of n x p (n = observations, p = independent variables)
#   mY: Column matrix of n x 1 dependent variables (n = observations)
#   mBeta: Column Matrix of p x 1 coefficients
#
# Output:
#   ESquared: double, residual squared errors

calcRSS <- function(mX, mY,mBeta){

  # calculate the errors
  mE <-  mY - mX %*% mBeta

  # get errors squared
  ESquared <- t(mE) %*% mE

  # return the residual sum of squared errors
  return(ESquared[1,1])

}

# calcCovar: Calculates the covariance matrix
#
# Parameters:
#   RSS: Residual squared errors
#   mXtX: pxp matrix, created from independent variables (X), multiplied with itself
#   n: double, number of observations
#   p: double, number of variables
#
# Output:
#   Covar: matrix, covariance matrix

calcCovar <- function(RSS, mXtX,n, p){

  # est. for sigma squared
  SigmaSquared <- (RSS) / (n - p -1)

  Covar <- SigmaSquared * as.matrix(inv(mXtX))

  return(Covar)

}

# calcSignificance: Calculates the statistical significance of a set of beta's
#
# Parameters:
#   RSS: Residual squared errors
```

```
#   mXtX: pxp matrix, created from independent variables (X), multiplied with itself
#   n: double, number of observations
#   p: double, number of variables
#   mBetaEst: matrix of estimated Beta's
#
# Output:
#   dfSignificance: dataframe, containing the results on statistical signficance

calcSignificance <- function(RSS, mXtX, n,p, mBetaEst){

  # get covariance matrix
  mCovar <- calcCovar(RSS,mXtX,n,p)

  # calculate the standard deviations
  stdev <- sqrt(diag(mCovar))

  # define t, which is t-distributed with n-p-1 degrees of freedom
  t <- mBetaEst/stdev
  pval <- 2*pt(-abs(t),df=n-p-1)

  dfSignificance <- data.frame(BetaEst = mBetaEst,
                               stdev = stdev,
                               t = t,
                               pval = pval)

  return(dfSignificance)
}

# calcLargestEigen: Calculates the largest eigenvalue of an matrix of independent variables
#
# Parameters:
#   mX: Dataframe of n x p (n = observations, p = independent variables)
#
# Output:
#   LargestEigenval: float, largest eigenvalue of said matrix

calcLargestEigen <- function(mX){

  # get the eigenvalues of X
  EigenValX <- eigen(mX)$values

  # from these eigenvalues, get the largest one
  LargestEigenVal <- max(EigenValX, na.rm = TRUE)

  return(LargestEigenVal)

}

# CalcStepScore: Calculates the % improvement between the k-1th and kth set of beta's
#
# Parameters:
#   prevBeta: double, k-1th beta
#   currbeta: double, kth beta
```

```
#   mX: Dataframe of n x p (n = observations, p = independent variables)
#
# Output:
#   StepScore; double, % improvement between the RSS of the two sets of beta's

calcStepScore <- function(mX,mY, prevBeta, currBeta){

  # difference in RSS between previous and current set of beta's
  diffRSS <- (calcRSS(mX,mY,prevBeta) - calcRSS(mX,mY,currBeta))

  # divide difference with previous score to get % change
  StepScore <- diffRSS /calcRSS(mX,mY,prevBeta)

  return(StepScore)

}

# calcRsquared
#
# Calculates the r-squared
#
# Parameters:
#   Y: matrix, the true dependent variable
#   Yest: matrix, the predicted dependent variable
#   (optional) adjusted: if True, return adjusted r squared
#   (optional) p: if adjusted is calculated, add number of variables
#
# Output:
#   Rsquared: double, the Rsquared or adjusted Rsquared for a linear model

calcRsquared <- function(mY, mYest, adjusted = FALSE, p=0, n=0){

  # standardize Y, and Yest (mean of 0)
  mStandY = mY - mean(mY)
  mStandYest = mYest - mean(mYest)

  # calculate Rsquared
  numerator <- (t(mStandY) %*% mStandYest)^2
  denominator <- (t(mStandY) %*% mY) %*% (t(mStandYest) %*% mStandYest)
  resultRsquared <- (numerator/denominator)

  # if want adjusted R squared,
  if(adjusted){

    adjRsquared = 1 - (((1-resultRsquared)*(n - 1))/(n-p-1))
    resultRsquared <- adjRsquared
  }

  return(resultRsquared)

}

# calcModelMM
```

```
#
# Calculates a linear model, using the majorization in minimization (MM) algorithm
#
# Parameters:
#   X: Dataframe of n x p (n = observations, p = independent variables)
#   Y: Dataframe of n x 1 dependent variables (n = observations)
#   e: epsilon, parameter for threshold of improvement after which the algorithm should halt
#   nBeta: number of variables one wants to use
#
# Output:
#   result: dataframe with attributes of the model:
#       - Beta: dataframe, the calculated Beta's
#       - RSS: double, Sum of squared residuals
#       - Yest: dataframe, the predicted Y
#       - Rsquared: double, R^2 for the predicted Y
#       - AdjRsquared: Adjusted Rsquared
#       - Significance results: dataframe with significance results on the beta's
#       - Residuals: dataframe, Y - Yest.
#

calcModelMM <- function(mX,mY,e, nBeta){

  # get number of observations, and number of variables minues the intercept
  n <- nrow(mX)
  p <- ncol(mX) - 1

  # check the user has filled in an appropriate amount of beta's
  if(nBeta > p + 1){
    stop("You want to use more variables than there are in the dataset of independent variables")
  }

  # set the previous beta variable to initial, random beta's
  prevBeta <- runif(ncol(mX), min=0, max=1)

  # calculate X'X
  mXtX <- t(mX) %*% mX

  # get largest eigenvalue for the square of independent variables
  Lambda <- calcLargestEigen(mXtX)

  # set initial stepscore to 0, k to 1.
  StepScore <- 0
  k <- 1

  # run while, either if k is equal to 1, or the improvement between k-1th and kth set of beta's is sma
  while (k == 1 | StepScore > e ){

    # step to next k
    k <- k + 1

    # calculate beta's for this k
    BetaK <- prevBeta - ((1/Lambda) *  mXtX %*% prevBeta) + ((1/Lambda) * t(mX) %*% mY )
```

```
    # sort the beta's based on absolute value, remove the smallest ones to keep m
    absBetaKOrdered <- order(abs(BetaK[,1]), decreasing = T)
    BetaK[!BetaK %in% BetaK[absBetaKOrdered,][1:nBeta]] <- 0

    # new stepscore, % difference in RSS between new Beta's and previous beta's
    StepScore <- calcStepScore(mX,mY,prevBeta,BetaK)

    # assign current beta's to prevBeta variable for next iteration
    prevBeta <- BetaK


  }

  ## Calculate several attributes of the linear model, put in dataframes or doubles

  # final Beta's
  BetaFinal <- as.matrix(BetaK)

  # calculate the RSS of this final est.
  RSSBetaK <- calcRSS(mX,mY, BetaK)

  # get the est. dependent variables
  mYest <- mX %*% BetaFinal

  # get the r2 and adjusted r2
  Rsquared <- calcRsquared(mY, mYest)
  adjRsquared <- calcRsquared(mY,mYest, adjusted = T,  p, n)

  # get the residuals
  Resi <- mY - mYest

  # get the results on significance
  dfSignificance <- calcSignificance(RSSBetaK, mXtX, n, p, BetaFinal)

  # add these attributes together as a list to make it easily accessible
  result <- list(Beta = BetaFinal,
                 RSS = RSSBetaK,
                 Yest = mYest,
                 Rsquared = Rsquared,
                 adjRsquared = adjRsquared,
                 SignificanceResults = dfSignificance,
                 Residuals = Resi,
                 n = n,
                 p = p)


  return(result)

}

# findModelMM
#
# finds the best linear model, using the MM algorithm, by testing model with 1, 2...up to all variables
```

```
#
# Parameters:
#   mX: Matrix of n x p (n = observations, p = independent variables)
#   mY: Matrix of n x 1 dependent variables (n = observations)
#
# Output:
#   results: list with the results for each model version

findModelMM <- function(mX, mY, e){

  # get the number of independent variables used
  nIndVar = ncol(mX) - 1

  # start at m = 1, create empty list to be filled with results
  M = 1
  results <- list()

  # for each m, check the best model and save the results
  while(M <= nIndVar){

    M <- M + 1

    resultM <- calcModelMM(mX, mY, e, M)

    strSave <- paste0("Model with ", M-1, " variable(s)")
    results[[strSave]] <- resultM

  }

  return(results)

}
```

Analysis

```
library(matlib)
library(stargazer)
```

```
## Warning: package 'stargazer' was built under R version 4.0.3
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
library(sjPlot)
```

```
## Warning: package 'sjPlot' was built under R version 4.0.3
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car

## Install package "strengejacke" from GitHub (`devtools::install_github("strengejacke/strengejacke")`)
```

```r
# load the air quality data
load("Data/Airq_numeric.Rdata")

# set to dataframe
dfAirQ <- data.frame(Airq)

# select dependent variable of air quality
Yair = dfAirQ$airq

# select all other variables as independent variables
Xair = dfAirQ[,-1]

# scale the independent variables, and add an intercept to these
XairScaled <- scale(Xair)
XairIntercept <- cbind(intercept = 1, XairScaled)

# set the data to matrix format
mYair <- as.matrix(Yair)
mXairIntercept <- as.matrix(XairIntercept)

# set seed to ensure stability of results
set.seed(0)

# set e small
e <- 0.0000000001

# select the number of beta's you want to use in the model
nBeta <- ncol(mXairIntercept) - 1

# calculate the model using the MM algorithm, using the max (5) variables
modelMM <- calcModelMM(mXairIntercept, mYair, e, nBeta)

# calculate the model with MM, for 1-5 variables. This contains all the values shown in the paper
compareModelMM <- findModelMM(mXairIntercept, mYair, e)
```


Baklanov, Alexander, Luisa T Molina, and Michael Gauss. 2016. "Megacities, Air Quality and Climate." *Atmospheric Environment* 126: 235–49.

Leelőssy, Ádám, Ferenc Molnár, Ferenc Izsák, Ágnes Havasi, István Lagzi, and Róbert Mészáros. 2014. "Dispersion Modeling of Air Pollutants in the Atmosphere: A Review." *Central European Journal of Geosciences* 6 (3): 257–78.

r-project. 2020. "Ecdat: Data Sets for Econometrics." World Health Organization. https://cran.r-project.org/web/packages/Ecdat/index.html.

WHO. 2020. "Air Pollution." World Health Organization. https://www.who.int/health-topics/air-pollution#tab=tab_1.

Xiong, Shifeng. 2014. "Better Subset Regression." *Biometrika* 101 (1): 71–84.