| | |
|---|---|
| **TITLE:** | **Agency for Healthcare Research & Quality (AHRQ) Data Science Challenge – Project Summary** |
| **PROJECT LEAD:** | **Daniel Linnen, PhD (DL)** |
| **DATA SCIENTIST:** | **Austin Powell, MS (AP)** |
| **DEVELOPER:** | **Richard Peute (RP)** |
| **SUBMITTED ON:** | **6/20/2019** |
| **PURPOSE:** | **Describe DS Challenge project steps and code used to develop predictions** |
| **OUTPUT TO:** | **AHRQ at https://cma.ahrq.gov/cma/registration.jsp?eventid=85** |

## OVERVIEW

Our hypothesis was that county or state characteristics of economic desolation (e.g., poverty, crime, provider shortages, rurality) would be significant features when predicting measures of hospital utilization. Plausibly, a lower percentage of individuals with insurance may be associated with emergency department crowding, or a financially depressed county may witness more issues with patient discharges because receiving facilities may be understaffed or may not exist in adequate numbers.

We downloaded data from various public data sources and performed joins to derive a singular .csv dataset for analysis. To make county names unique, we generated a new variable **countystate** for each dataset with county-level data. **countystate** combines county name and state abbreviation into one (no space: e.g., **alamedaca**). In section 2, we provide (pp.3-5) an overview of the datasets. The joined pre-analysis dataset had the same number and sequence of rows (counties) as the original dataset provided by AHRQ. DL identified the datasets and provided join specifications to RP. Most datasets were joined against this main dataset either on county or the state abbreviation using Microsoft Access. Where specified, RP generated summary variables at the state level in Python (see section 3). Some variables were collected and/or generated at the state-level.

## 1. Submission files

- DL developed the file **"AHRQ Linnen datasets summary 2019-0430.csv"**, which describes the public datasets used for this challenge
- RP developed **"AHRQ Linnen Data Science Challenge.accdb",** which contains these datasets. It is available upon request (approx. 30MB)
- RP developed **"AHRQ Linnen joins python.pdf",** which contains code for additional joins and data transformations in Python
- RP saved the joined dataset as **"AHRQ challenge Linnen 2019-05-08.csv"**.
- DL and AP developed **"AHRQ PPL 2019-0620.R",** which contains R code to generate additional variables, wrangle the data and derive predictions using random forest and cross validation
- DL and AP saved 2016 and 2017 prediction values in **"AHRQ PPL predictions.csv".**
    - Predicted values of the number of hospital inpatient discharges and the mean length of stay for selected counties in the U.S. for year 2016 = n_2016_preds and avglos_2016_preds in csv file
    - Predicted values of the total number of hospital inpatient discharges and the mean length of stay for selected counties in the U.S. for year 2017 = n_2017_preds and avglos_2017_preds in csv file

We describe all code in section 3.

## 2. Datasets

Below, we describe the datasets used for this submission. The file "AHRQ Linnen datasets summary 2019-0430.csv" provides the same overview in more detail.

| **Dataset working name** | **Full dataset name** | **Federal Agency** | **Description** | **Join** |
|---|---|---|---|---|
| utilization_main | Predictive Analytic Challenge Updated 2019-04-11 | AHRQ | Main competition dataset with hospital discharges and average length of stay at the county level | primary key: **countystate** (new variable; composite of cntyname and state [no space]) for example: alamedaca |
| hospitals | Hospital General Information | Centers for Medicare & Medicaid Services | A list of all hospitals that have been registered with Medicare. The list includes addresses, phone numbers, hospital type, and overall hospital rating. | left join utilization_main and hospitals on **countystate** (generate new variable; composite of County and State [no space]) |
| hac | Hospital-Acquired Condition Reduction Program | Centers for Medicare & Medicaid Services | In October 2014, CMS began reducing Medicare payments for subsection (d) hospitals that rank in the worst-performing quartile with respect to hospital-acquired conditions (HACs). | left join hospitals and hac on **Provider ID** |

| | | | | |
|---|---|---|---|---|
| age | US Census county age | United States Census Bureau | Age distributions and % by US county | left join utilization_main and age on **countystate** (generate new variable; composite of County and State [no space]) |
| sex | US census county sex | United States Census Bureau | Sex distributions by US county | left join utilization_main and sex on **countystate** (generate new variable; composite of County and State [no space]) |
| poor | US census county poverty | United States Census Bureau | Families with Income Below 1-2 Times the U.S. Federal Poverty Level Percent of Total | left join utilization_main and poor on **countystate** (generate new variable; composite of County and State [no space]) |
| crime | state crime: uniform crime reporting statistics | Federal Bureau of Investigations | data are state-level | left join utilization_main and crime on **state** |
| rn_supply | Rn supply vs demand at state level | U.S. Department of Health and Human Services Health Resources and Services Administration; Bureau of Health | data are state-level; Table 5, p.17 | left join utilization_main and rn_supply on **state/state abb** |

| | | | | |
|---|---|---|---|---|
| | | Professions; National Center For Health Workforce Analysis | | |
| urban_rural | NCHS Urban-Rural Classification Scheme for Counties | National Center for Health Statistics | a county's urban-rural designation across 6 categories ranging from metropolitan center to rural | left join utiliozation_main and urban-rural on **fipsstco/FIPS code** |
| underserved | Health profession shortage and Medically Underserved Populations | Health Resources & Services Administration | State-level HPSA scores for mental health, dental health, primary care: This attribute represents the Health Professional Shortage Area (HPSA) Score developed by the National Health Service Corps (NHSC) in determining priorities for assignment of clinicians. The scores range from 0 to 26 where the higher the score, the greater the priority | left join utilization_main and underserved on **countystate** (generate new variable; composite of County and State [no space]) |
| beds | Beds per 1000 persons per state | American Hospital Association | data are state-level | left join utilization_main and rn_supply on **state/state abb** |

## 3. Code

RP joined the data in MS Access (see "AHRQ Linnen Data Science Challenge.accdb") and used Python to flatten county level data to state level for select variables.

### 3.1. Python code (same as "AHRQ Linnen joins python.pdf")

```
# coding: utf-8

# In[29]:


import datetime


# In[18]:


import pyodbc
import pandas as pd

db_file = r'C:\Users\xxxx\Code\data science challenge\Data Science
Challenge.accdb'
user = 'admin'
password = ''

cnxn = pyodbc.connect('DRIVER={{Microsoft Access Driver
(*.mdb, *.accdb)}};DBQ={};Uid={};Pwd={};'.format(db_file,
user, password))


# In[3]:


qryHospitals = "SELECT countystate, [Provider ID], [Hospital Type],
[Meets criteria for meaningful use of EHRs], [Hospital overall
rating], [Mortality national comparison], [Safety of care national
comparison], [Readmission national comparison], [Patient experience
national comparison], [Timeliness of care national comparison] FROM
hospitals_fk"
dfHospitals = pd.read_sql(qryHospitals, cnxn, index_col='Provider ID')


# In[4]:


qryHac = "SELECT [Provider ID], CLABSI_W_Z_SCORE, CAUTI_W_Z_SCORE,
SSI_W_Z_SCORE,
```

```
TOTAL_HAC_SCORE FROM hac"
dfHac = pd.read_sql(qryHac, cnxn, index_col='Provider ID')


# In[5]:


qryUtilization_main_county_state = 'SELECT * FROM
utilization_main_county_state ORDER By countystate'
dfUtilization_main_county_state =
pd.read_sql(qryUtilization_main_county_state, cnxn,
index_col='countystate')


# In[19]:


qryUnderservedMeans = "SELECT [Primary State Abbreviation] AS
State, [HPSA Discipline Class], AVG([HPSA Score]) AS [Mean (HPSA
Score) State Lvl] FROM underserved_fk WHERE [HPSA Status] =
'Withdrawn' GROUP BY [Primary State Abbreviation], [HPSA
Discipline Class]"
dfUnderservedMeans = pd.read_sql(qryUnderservedMeans, cnxn,
index_col="State")


# In[20]:


cnxn.close()


# In[8]:


dfHospitals_hac = dfHospitals.join(dfHac, rsuffix='hac')
dfHospitals_hac.sort_values(by='countystate', inplace=True)


# In[9]:


for col in dfHospitals_hac.columns:
    print(col)


# In[10]:
```

```
print(dfHospitals_hac)


# In[11]:


hospitals_hacColumns = dfHospitals_hac.columns
lastCountystate = ''
row_dict = {}
data_list = []
first_row = True

for row in dfHospitals_hac.itertuples():
    if lastCountystate == row[1]:
        counter += 1
        colCount = 1
        col_name = 'Provider ID_%04d' % counter
        row_dict[col_name] = row[0]
        for col in hospitals_hacColumns:
            if col != 'countystate':
                col_name = col + "_%04d" % counter
                row_dict[col_name] = row[colCount]
            colCount += 1
    else:
        counter = 0
        colCount = 1
        if row_dict:
            data_list.append(row_dict)
            row_dict = {}
            row_dict['countystate'] = row[1]
        if first_row:
            row_dict['countystate'] = row[1]
            first_row = False
        col_name = 'Provider ID_%04d' % counter
        row_dict[col_name] = row[0]
        for col in hospitals_hacColumns:
            if col != 'countystate':
                col_name = col + "_%04d" % counter
                row_dict[col_name] = row[colCount]
            colCount += 1
    lastCountystate = row[1]
data_list.append(row_dict)
dfHospitals_hac_flat = pd.DataFrame(data_list)
dfHospitals_hac_flat.set_index('countystate', inplace=True)
print(dfHospitals_hac_flat)
```

```
# In[12]:


for col in dfHospitals_hac_flat.columns: print(col)


41: In[21]:


underservedMeansCols = dfUnderservedMeans.columns
lastState = ''
row_dict = {}
data_list = []
first_row = True

for row in dfUnderservedMeans.itertuples():
    if lastState == row[ø]:
        col_name = underservedMeansCols[1] + " - " + row[1]
        row_dict[col_name] = row[2]
    else:
        if row_dict:
            data_list.append(row_dict)
            row_dict = {}
            row_dict['State'] = row[ø]
        if first_row:
            row_dict['State'] = row[ø]
            first_row = False
        col_name = underservedMeansCols[1] + " - " + row[1]
        row_dict[col_name] = row[2]
        lastState = row[ø]
data_list.append(row_dict)
41:print(data_list)
dfUnderservedMeans_flat = pd.DataFrame(data_list)
dfUnderservedMeans_flat.set_index('State', inplace=True)
print(dfUnderservedMeans_flat)


41: In[16]:


dfUtilization_main_county_state_hosp_hac =
dfUtilization_main_county_state.merge(dfHospitals_hac_flat,
how='left', on='countystate', suffixes=(False, 'hosp'))
print(dfUtilization_main_county_state_hosp_hac)
```

```
41: In[27]:


dfUtilization_main_county_state_hosp_hac_underserved =
dfUtilization_main_county_state_hosp_hac.merge(dfUnderservedMeans_flat,
how='left', left_on='state', right_on='State', suffixes=(False,
'hosp'))
for col in
    dfUtilization_main_county_state_hosp_hac_underserved.column
    s: print(col)
dfUtilization_main_county_state_hosp_hac_underserved.sort_values(by
 =[ state , cntyname ], inplace=True)
print(dfUtilization_main_county_state_hosp_hac_underserved)


# In[37]:


version = 0


# In[38]:


version += 1
if version ==1:
    strVersion =
else:
    strVersion = v%s % str(version)
now = datetime.datetime.now()
year = {:02d} .format(now.year)
month = {:02d} .format(now.month)
day = {:02d} .format(now.day)
todays_date = {}-{}-{} .format(year, month, day)
csv_file = r C:\Users\xxxx\Code\data science challenge\AHRQ challenge
Linnen
%s%s.csv % (todays_date, strVersion)
dfUtilization_main_county_state_hosp_hac_underserved.to_csv(path_or_bu
f=csv_file,
index=False)
```

### 3.2 R code: Data wrangling, variable generation, model derivation and validation

github: data2vizdom/ahrq

Per the DUA, we set the github project to private. We will grant AHRQ collaborator access upon request, and we will delete the original AHRQ data on 8/1/2019

Steps 1 and 2 code by DL
Remaining code by AP

```
#################################################
# AHRQ 2019 Data Science Challenge              #
# Daniel Linnen, Austin Powell, Richard Peute   #
# San Francisco Bay Area, CA  |  May-June 2019   #
#################################################


# We extracted data from public agency sources, then joined and
transformed the dataset using Python
# in a seperate file (part of submission). We performed all additional
work in this R script.



rm(list=ls()) # clear the environment

##################### LOAD PACKAGES ############################

library(dplyr);library(ggplot2);library(caret);library(tidyverse);
library(caret)

#################################################################



##################### SET DIRECTORY ############################

setwd("~/competition_archive/ahrq")
```

```
#####################################################################
# STEP 1: DATA PREP
#####################################################################


#import dataset

joined_dataset <- read.csv("C:/users/p104425/Desktop/r_projects/AHRQ
challenge Linnen 2019-05-08.csv",

                           header=TRUE, na.strings=c("","  ","."))

View(joined_dataset)

a <- ncol(joined_dataset)


# retain only variables with missing data < 10%

clean_data <-
joined_dataset[colSums(is.na(joined_dataset))/nrow(joined_dataset) <
.1]

View(clean_data)

b <- ncol(clean_data)

a-b

sum(is.na(clean_data)) # (We removed 1042 variables with incomplete
data and retained a dataset with only 1 NA)

summary(clean_data)

rm(a,b)



# remove joined_dataset from environment

rm(joined_dataset)



# inspect remaining 83 variables, then remove variables that do not
meet project needs

colnames(clean_data)

clean_data2 <- clean_data[, -c(16:29, 51:53, 59, 61:69)]
```

```
colnames(clean_data2)

ncol(clean_data2) # we removed an additional 27 variables



# the following measurement categories remained before data
exploratiom, variable generation and analysis:

# county-level: hospital utilization, demographics, NCHS urban-rural
class, poverty

# state-level: registered nurse supply/demand, health professional
shortage (HPSA), hospital beds, crime



# Variable generation

# produce 3 age groups: % young, % middle-aged, % old

pct_young <- clean_data2$Age.17.Years.or.Fewer.Percent.of.Total

pct_old <- clean_data2$Age.65.Years.or.Greater.Percent.of.Total

pct_mid_aged <- 100 - (pct_young + pct_old)



# calculate a young-to-old ratio

y2o <- pct_young / pct_old



# calculate a female-to-male ratio

female <- clean_data2$Total.Population.Female

male <- clean_data2$Total.Population.Male

f2m <- female / male



# generate a new variable metro_flag: flags NCHS class 1 and 2 urban
metropolitan counties

metro_flag <- clean_data2$X2013.code  < 3
```

```
# generate a new variable metro_sum: sum of metro_flag by state using
dplyr

clean_data3 <- clean_data2 %>% group_by(state) %>%  mutate(

  metro_sum = sum(metro_flag)

)



# generate a new variable desolate_flag: flags counties that are non-
metro, poor, and high-crime (1/0)

hist(clean_data2$Families.with.Income.Below.100.Times.the.US.Federal.P
overty.L)

boxplot(clean_data2$Families.with.Income.Below.100.Times.the.US.Federa
l.Poverty.L)

colnames(clean_data3)[colnames(clean_data3)=="Families.with.Income.Bel
ow.100.Times.the.US.Federal.Poverty.L"] <- "pct_poor"

#75th percentile cutoff for poor is ~14%

hist(clean_data2$Violent.Crime.rate)

boxplot(clean_data2$Violent.Crime.rate) #75th percentile cutoff for
violent crime is ~400 per 100,000 inhabitants)



clean_data3 <- clean_data3 %>% mutate(

  desolate_flag = case_when(X2013.code >= 3 & pct_poor >= 14 &
Violent.Crime.rate >= 400 ~1,

                            TRUE ~0)

)



# generate a new variable desolate_sum: state level sum of
desolate_flag using dplyr

clean_data3 <- clean_data3 %>% group_by(state) %>% mutate(

  desolate_sum = sum(desolate_flag)

)
```

```
colnames(clean_data3)

clean_data3[,c("X2013.code","pct_poor","Violent.Crime.rate","desolate_
flag")] #checking logic is correct



#remove age variables not used in analysis

final_data <- clean_data3[, -c(16:19, 21:26, 28, 29)]

colnames(final_data)[colnames(final_data)=="sex_fk_Total.Population"]
<- "pop_size"



#remove data frames

rm(clean_data)

rm(clean_data2)

rm(clean_data3)



#save dataframe and in-memory values

save(final_data, file="ahrq_data.rda")



# STEP 2: DATA EXPLORATION
################################################################


# explore data for normality: smooth density plots


final_data %>%

  keep(is.numeric) %>%                     # Keep only numeric columns

  gather() %>%                             # Convert to key-value
pairs

  ggplot(aes(value)) +                     # Plot the values

  facet_wrap(~ key, scales = "free") +     # In separate panels
```

```
  geom_density()

# density plots saved seperately and is part of submission


#########################################################################
#   BASE MODEL
#########################################################################

# Run training

#########################################################################
# STEP 3: MODELING
#########################################################################

# Outcome 1: 2016 Total Discharges (county)

# Outxome 2: 2016 Average length of stay  (county)


# convert to dataframe

df = data.frame(final_data)


# create feature for weighting the scores

s = as.data.frame(df %>%

                    group_by(cntyname) %>%

                    summarise(countySum = sum(County.2012.pop)))

df = merge(s,df,by="cntyname")

# create county weighted sum proportion

df$countySumProportion = df$countySum/sum(df$County.2012.pop)


#################### Metric Functions ############################

R2 <- function (x, y) cor(x, y) ^ 2

# Competition scoring function

arhqScoreFunction <- function(predCol,preds) sum(abs(df[,predCol] -
preds) / df[,predCol] * df$countySumProportion)
```

```
#################################################################
#################### LOAD DATA ##################################

# This data has already been cleaned and prepped


df$X2013.code <- as.factor(df$X2013.code)

df$state <- as.factor(df$state)

#################################################################



## Summary information

print(dim(df))

print(colnames(df))




#################### SET OUTCOMES ###############################

# Outcome 1

totalDischargeCols <- c("n_2011","n_2012","n_2013","n_2014" ,"n_2015",
"n_2016" )

# Outcome 2

avlosCols <-
c("avglos_2011","avglos_2012","avglos_2013","avglos_2014","avglos_2015
","avglos_2016")

#################################################################

## prediction columns

#predCols <- totalDischargeCols



# Set seed for prediction consistancy

set.seed(3456)
```

```
#################### FEATURE SELECTION PROCESS #################

# Choice of features was primarily based off of validation of score
improvement with

# cross-validation. Through exploration of different features, it was
apparent that

# many variables were quit correlated with each other and hurt the
score validation.

################################################################



# This columns are used in cross-validation stage

X <-
c("Burglary.rate","Population","Legacy.rape.rate..1","X2016.Beds.per.1
.000.Persons","Excess.or.Shortage..Supply.Less.Demand.......Shortage."
,"state",'fipsstco')



# outcome 1: 'discharge'

# outcome 2: 'avglos'

# "Mean ahrqScore 7.78699023182547 Mean rmse 38316.2435638335 Mean R2
0.112036499308017"

CrossValidation <- function(outcome){


  if (outcome == 'discharge'){

    # Outcome 1

    cvYears = c("n_2011","n_2012","n_2013","n_2014" ,"n_2015",
"n_2016")

    cvYears2 =
c("avglos_2011","avglos_2012","avglos_2013","avglos_2014","avglos_2015
","avglos_2016")

  } else if (outcome == 'avglos'){

    cvYears =
c("avglos_2011","avglos_2012","avglos_2013","avglos_2014","avglos_2015
","avglos_2016")
```

```
    cvYears2 = c("n_2011","n_2012","n_2013","n_2014" ,"n_2015",
"n_2016")

  }

  # Create empty vectors to store scores from each data split

  ahrqScores <- c()

  rmseScores <- c()

  R2Scores <- c()




  for (i in 1:4){

    print(paste("Iteration:",i))


    # Iteration 1

    if (i == 1){

      X_train <- X

      predCol <- cvYears[i]

    }

    # Iteration 2

    else if (i == 2){

      X_train <- c(X,cvYears[1],cvYears2[1])

      predCol <- cvYears[2]

    }

    # Iteration 3

    else if (i == 3){

      X_train <- c(X,cvYears[1:2],cvYears2[1:2])

      predCol <- cvYears[3]
```

```
    }
    # Iteration 4
    else if (i == 4){
      X_train <- c(X,cvYears[1:3],cvYears2[1:3])
      predCol <- cvYears[4]
    }

    print("Training on columns:")
    print(X_train)

    # train
    fit <-  train(as.formula(paste(predCol, "~ ",X_train)), data = df,
                  method = 'rf',
                  verbose = FALSE)
    # preds
    preds_rf <- predict(fit, data=df[which(colnames(df) %in%
X_train)])

    preds <- preds_rf

    print("Scoring on")
    print(cvYears[i+1])
    ahrqScore <- arhqScoreFunction(cvYears[i+1],preds)
    rmse = Metrics::rmse(df[,cvYears[i+2]],preds)
    r2 = R2(df[,cvYears[i+2]],preds)
    ## Append scores to to list of scores
```

```
    ahrqScores = c(ahrqScores,ahrqScore)

    rmseScores = c(rmseScores,rmse)

    R2Scores = c(R2Scores,r2)



print("_____________________________________________________________________
_____________")

  }

  print(paste('Mean ahrqScore',mean(ahrqScores),

              'Mean rmse',mean(rmseScores),

              'Mean R2',mean(R2Scores))

  )

}

CrossValidation('avglos')



###################### CREATE VALIDATION ########################

# set aside validation data (2016 discharges and 2016 average length
of stay)

## Validation

preds_2016 = list()

for (columnSet in list(avlosCols,totalDischargeCols)){




  ValCols <- columnSet

  X_val_train <- c(X,ValCols[1:4])

  y_val_train <- ValCols[5]


  X_val_test <- c(X,ValCols[2:5])
```

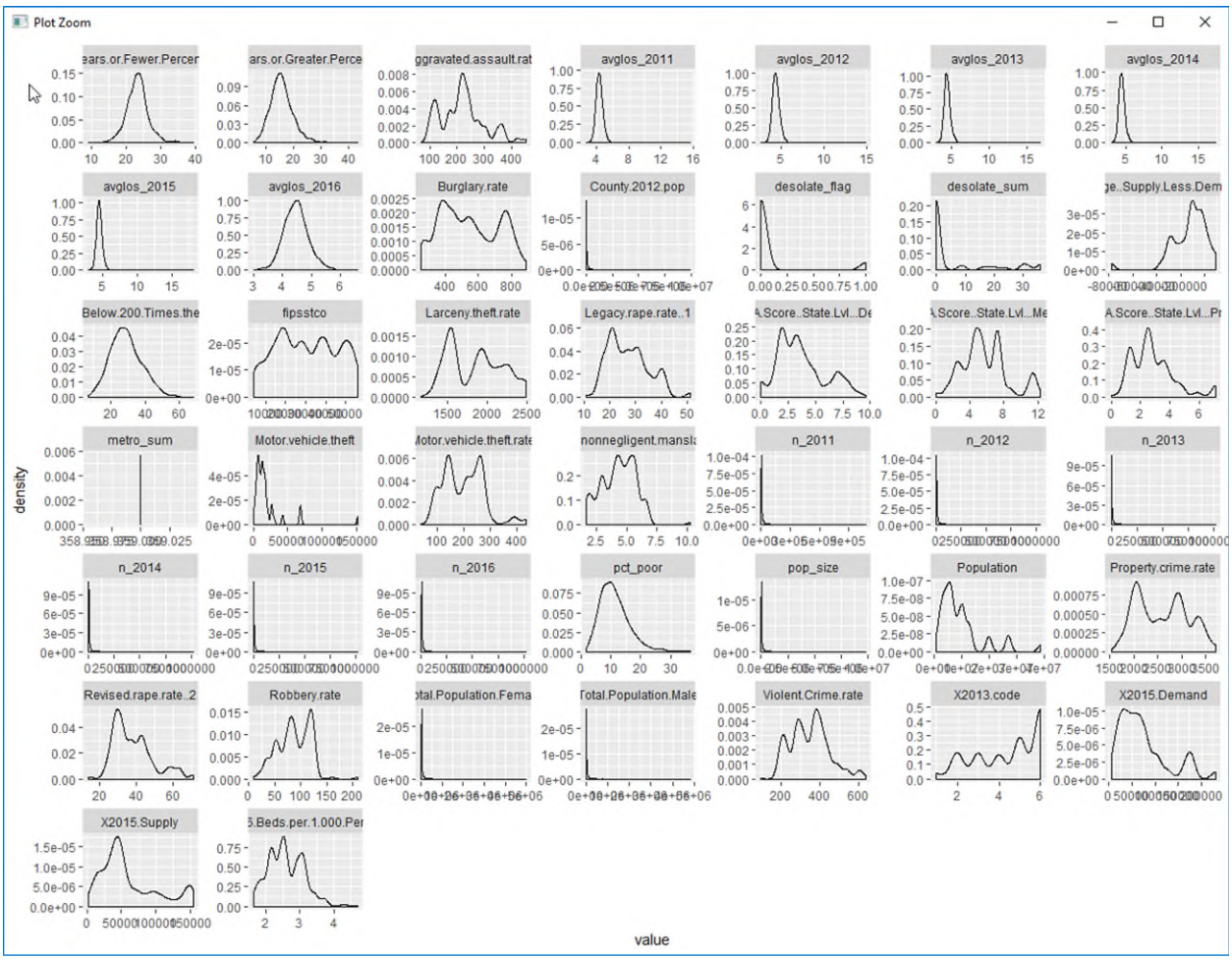
```
  y_val_test <- ValCols[6]


  valFit <-  train(y=df[,y_val_train],x=df[,which(colnames(df) %in%
X_val_train)], data = df,
                   method = 'rf', importance=FALSE)


  preds = predict(valFit, data=df[which(colnames(df) %in%
X_val_test)])
  preds_2016[[paste0(ValCols[6],"_preds")]] = preds
  ## Print out validation set metrics
  print(arhqScoreFunction("n_2016",preds))
  print(R2(df[,ValCols[6]],preds))
}


#########################################################################
#  Variable Importance Determined by validation dataset
#########################################################################
## Uncomment to run (takes a while to run)
## variable can be set to either for variable importance
# y_val_train <- "n_2016"
#y_val_train <= "avglos_2017"


# valFit <-  train(y=df[,y_val_train],x=df[,which(colnames(df) %in%
X_val_train)], data = df,
#                   method = 'rf', importance=TRUE)
#
# valImp <- varImp(valFit, scale = FALSE)
# valImp
```

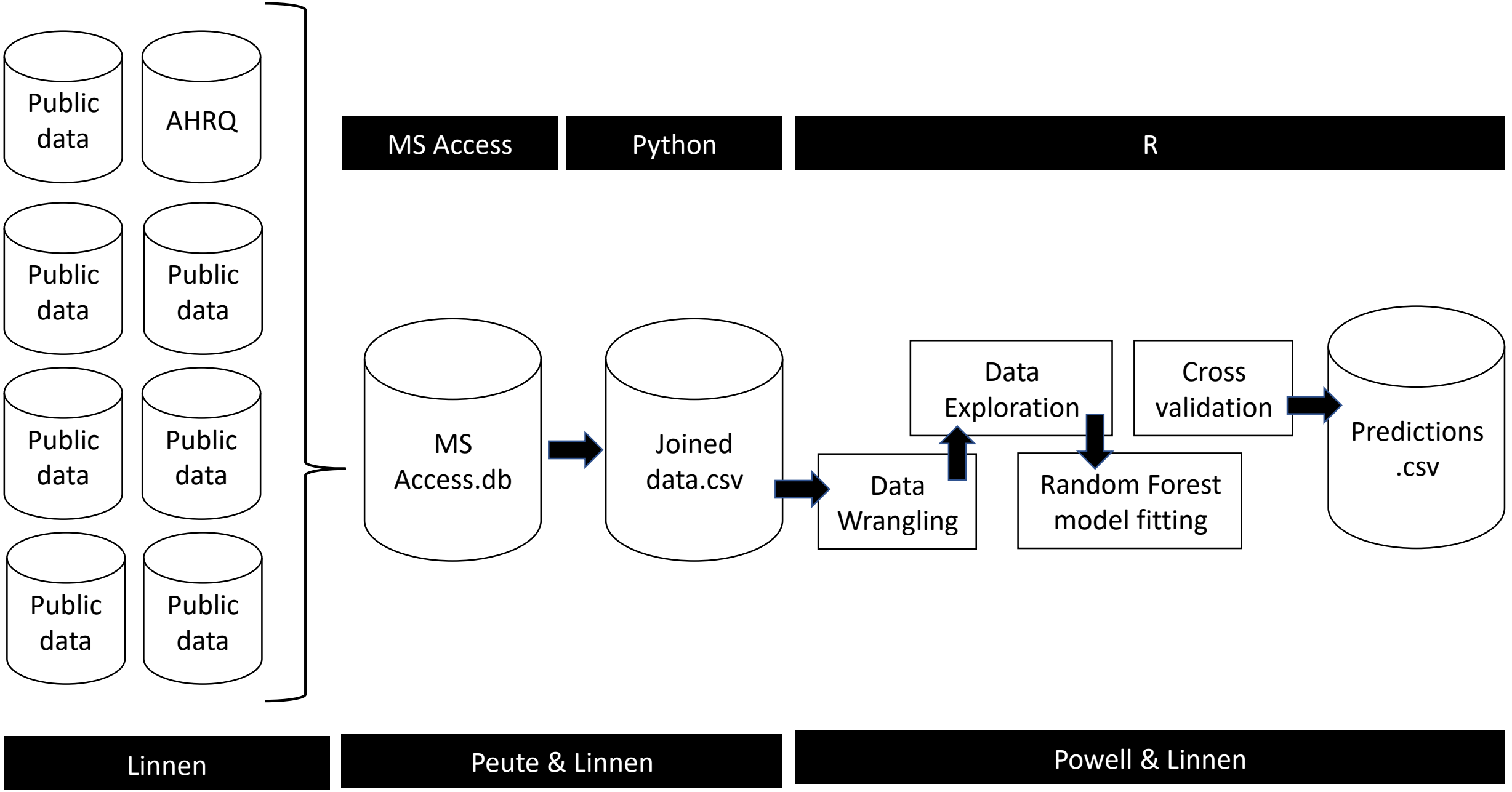
```
#########################################################################
########################### CREATE SUBMISSION ###########################



# set outcomes

outcomes <- list(n_2017_preds = totalDischargeCols, avglos_2017_preds
= avlosCols)

# Create dataframe for submission

n <- dim(df)[1]

df_preds <- data.frame(index=matrix(seq(1,n), nrow=n, ncol=1))

x <- c("index")

colnames(df_preds) <- x

df_preds$cntyname <- df$cntyname

df_preds$state <- df$state



## Append the 2016 preds

df_preds$n_2016_preds = preds_2016[['n_2016_preds']]

df_preds$avglos_2016_preds = preds_2016[['avglos_2016_preds']]



for (idx in range(c(1,2))){

  submissionCols <- outcomes[[idx]]



  X_sub_train <- c(X,submissionCols[1:5])

  y_sub_train <-  submissionCols[6]



  subFit <-  train(as.formula(paste(y_sub_train, "~ ",X_sub_train)),
data = df,

                   method = 'rf',
```

```
                    verbose = FALSE)


  X_sub_test <- c(X,submissionCols[2:6])

  preds = predict(subFit, data=df[which(colnames(df) %in%
X_sub_test)])

  df_preds[,names(outcomes[idx])] <- preds

}



head(df_preds)

tail(df_preds)



# save predictions to .csv

write.csv(df_preds, file="C:/Users/p104425/Desktop/r_projects/AHRQ PPL
predictions.csv", row.names=FALSE)



########################## END ###################################
##########################################################################
```

## Appendix.

## Smooth density plots (Step 2)




Public data
AHRQ
Public data
Public data
Public data
Public data
Public data
Public data
MS Access
Python
R
MS Access.db
Joined data.csv
Data Wrangling
Data Exploration
Random Forest model fitting
Cross validation
Predictions .csv
Linnen
Peute & Linnen
Powell & Linnen